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

Overview:
- link layer services
- error detection, correction
- multiple access protocols and LANs
- link layer addressing, ARP
- specific link layer technologies:
  - Ethernet
  - hubs, bridges, switches
  - IEEE 802.11 LANs
  - PPP
  - ATM

Link Layer: setting the context
Link Layer: setting the context

- two physically connected devices:
  - host-router, router-router, host-host
- unit of data: frame

Link Layer Services

- Framing, link access:
  - encapsulate datagram into frame, adding header, trailer
  - implement channel access if shared medium,
  - ‘physical addresses’ used in frame headers to identify source, dest
    - different from IP address!
- Reliable delivery between two physically connected devices:
  - 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 sender and receivers

- **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

Link Layer: Implementation

- implemented in “adapter”
  - e.g., PCMCIA card, Ethernet card
  - typically includes: RAM, DSP chips, host bus interface, and link interface

![Diagram of link layer components]
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

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

<table>
<thead>
<tr>
<th>Column parity</th>
<th>d_{i,1}</th>
<th>\ldots</th>
<th>d_{i,j}</th>
<th>\ldots</th>
<th>d_{i+1,1}</th>
<th>\ldots</th>
<th>d_{i+1,j}</th>
</tr>
</thead>
<tbody>
<tr>
<td>d_{1,1}</td>
<td>d_{1,2}</td>
<td>\ldots</td>
<td>d_{1,j}</td>
<td>\ldots</td>
<td>d_{2,1}</td>
<td>\ldots</td>
<td>d_{2,j}</td>
</tr>
<tr>
<td>d_{2,1}</td>
<td>\ldots</td>
<td>d_{2,j}</td>
<td>\ldots</td>
<td>d_{3,1}</td>
<td>\ldots</td>
<td>d_{3,j}</td>
<td></td>
</tr>
<tr>
<td>\ldots</td>
<td>\ldots</td>
<td>\ldots</td>
<td>\ldots</td>
<td>\ldots</td>
<td>\ldots</td>
<td>\ldots</td>
<td>\ldots</td>
</tr>
<tr>
<td>d_{i,1}</td>
<td>\ldots</td>
<td>d_{i,j}</td>
<td>\ldots</td>
<td>d_{i+1,1}</td>
<td>\ldots</td>
<td>d_{i+1,j}</td>
<td></td>
</tr>
<tr>
<td>d_{i+1,1}</td>
<td>\ldots</td>
<td>d_{i+1,j}</td>
<td>\ldots</td>
<td>d_{i+2,1}</td>
<td>\ldots</td>
<td>d_{i+2,j}</td>
<td></td>
</tr>
<tr>
<td>\ldots</td>
<td>\ldots</td>
<td>\ldots</td>
<td>\ldots</td>
<td>\ldots</td>
<td>\ldots</td>
<td>\ldots</td>
<td>\ldots</td>
</tr>
<tr>
<td>d_{n,1}</td>
<td>\ldots</td>
<td>d_{n,j}</td>
<td>\ldots</td>
<td>d_{n+1,1}</td>
<td>\ldots</td>
<td>d_{n+1,j}</td>
<td></td>
</tr>
</tbody>
</table>

- 10101 10101
- 111100 111100
- 011101 011101
- 001010 001010

**Correctable:**
Single bit error

**No errors:**
Parity 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...

Checksumming: Cyclic Redundancy Check

- 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 (ATM, HDCL)

\[ D \cdot 2^r \text{ XOR } R \]

---

5: DataLink Layer 5a-9

5: DataLink Layer 5a-10
CRC Example

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

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

 CRC Implementation (cont)

- The sender carries out on-line, in hardware the division of the string \( D \) by the polynomial \( G \) and appends the remainder \( R \) to it
- The receiver divides \( <D, R> \) by \( G \); if the remainder is non-zero, the transmission was corrupted
- International standards for \( G \) polynomials of degrees 8, 12, 16 and 32 have been defined
  - CRC-12: \( X^{12} + X^{11} + X^9 + X^2 + X + 1 \)
  - CRC-16: \( X^{16} + X^{15} + X^2 + 1 \)
  - CRC-CCITT: \( X^{16} + X^{12} + X^5 + 1 \)
  - CRC-32: \( X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} + X^{10} + X^8 + X^7 + X^5 + X^4 + X^2 + X + 1 \)
- ARPANET was using a 24 bit CRC for the alternating bit link protocol
- ATM is using a 32 bit CRC in ALL 5 (CRC-32)
- HDLC uses a 16 bit CRC (CRC-16)
Multiple Access Links and Protocols

Three types of “links”:
- point-to-point (single wire, e.g. PPP, SLIP)
- broadcast (shared wire or medium; e.g, Ethernet, Wavelan, etc.)
- switched (e.g., switched Ethernet, ATM etc)

Multiple Access protocols

- single shared communication channel
- two or more simultaneous transmissions by nodes: interference
  - only one node can send successfully at a time
- multiple access protocol:
  - distributed algorithm that determines how stations share channel, i.e., determine when station can transmit
  - communication about channel sharing must use channel itself!
  - what to look for in multiple access protocols:
    - synchronous or asynchronous
    - information needed about other stations
    - robustness (e.g., to channel errors)
    - performance
Multiple Access protocols

- claim: humans use multiple access protocols all the time
- class can "guess" multiple access protocols
  - multiaccess protocol 1:
  - multiaccess protocol 2:
  - multiaccess protocol 3:
  - multiaccess protocol 4:

MAC Protocols: a taxonomy

Three broad classes:
- Channel Partitioning
  - divide channel into smaller "pieces" (time slots, frequency)
  - allocate piece to node for exclusive use
- Random Access
  - allow collisions
  - "recover" from collisions
- "Taking turns"
  - tightly coordinate shared access to avoid collisions

Goal: efficient, fair, simple, decentralized
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

![TDMA Diagram](image)

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

![FDMA Diagram](image)
Channel Partitioning (CDMA)

CDMA (Code Division Multiple Access)
- unique "code" assigned to each user; ie, code set partitioning
- used mostly in wireless broadcast channels (cellular, satellite, etc)
- all users share same frequency, but each user has own "chipping" sequence (ie, code) to encode data
- encoded signal = (original data) X (chipping sequence)
- decoding: inner-product of encoded signal and chipping sequence
- allows multiple users to "coexist" and transmit simultaneously with minimal interference (if codes are "orthogonal")

CDMA Encode/Decode

![CDMA Encode/Decode Diagram]
**CDMA:** two-sender interference

**senders**

- data bits
- code

**channel, \( z_{1,m} \)**

**receivers**

- slot 1 received input
- slot 0 received input

\[
\begin{align*}
z_{1,m} &= d_1^1 c_1^m \\
&= d_2^1 c_2^m
\end{align*}
\]

**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 and CSMA/CD
**Slotted Aloha**

- time is divided into equal size slots (= pkt trans. time)
- node with new arriving pkt: transmit at beginning of next slot
- if collision: retransmit pkt in future slots with probability p, until successful.

![Slotted Aloha diagram]

**Slotted Aloha efficiency**

**Q:** what is max fraction slots successful?

**A:** Suppose N stations have packets to send
- each transmits in slot with probability p
- prob. successful transmission S is:

- by single node: \( S = p (1-p)^{N-1} \)
- by any of N nodes
  \[ S = \text{Prob (only one transmits)} = N \cdot p \cdot (1-p)^{N-1} \]
  ... choosing optimum p as \( n \to \infty \) ...
  \[ = 1/e = .37 \text{ as } N \to \infty \]

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

- unslotted Aloha: simpler, no synchronization
- pkt needs transmission:
  - send without awaiting for beginning of slot
- collision probability increases:
  - pkt sent at $t_0$ collide with other pkts sent in $[t_0-1, t_0+1]$

**Pure Aloha (cont.)**

\[ P(\text{success by given node}) = P(\text{node transmits}) \cdot P(\text{no other node transmits in } [p_0-1,p_0]) \cdot P(\text{no other node transmits in } [p_0-1,p_0]) = p \cdot (1-p) \cdot (1-p) \]

\[ P(\text{success by any of } N \text{ nodes}) = N \cdot p \cdot (1-p) \cdot (1-p) \]

... choosing optimum $p$ as $n \rightarrow \infty$ ...

\[ = \frac{1}{2e} = .18 \]
**CSMA: Carrier Sense Multiple Access**

**CSMA**: listen before transmit:
- If channel sensed idle: transmit entire pkt
- If channel sensed busy, defer transmission
  - Persistent CSMA: retry immediately with probability p when channel becomes idle (may cause instability)
  - Non-persistent CSMA: retry after random interval
- Human analogy: don’t interrupt others!

---

**CSMA collisions**

Collisions can occur: propagation delay means two nodes may not year hear each other’s transmission

Collision: entire packet transmission time wasted

Note: role of distance and propagation delay in determining collision prob.
**CSMA/CD (Collision Detection)**

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

- collision detection:
  - easy in wired LANs: measure signal strengths, compare transmitted, received signals
  - difficult in wireless LANs: receiver shut off while transmitting

- human analogy: the polite conversationalist

---

**CSMA/CD collision detection**

![Diagram of CSMA/CD collision detection](attachment:image.png)
“Taking Turns” MAC protocols

channel partitioning MAC protocols:
- share channel efficiently at high load
- inefficient at low load: delay in channel access, 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!

Polling:
- master node “invites” slave nodes to transmit in turn
- Request to Send, Clear to Send msgs
- concerns:
  - polling overhead
  - latency
  - single point of failure (master)

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

Distributed Polling:
- time divided into slots
- begins with N short reservation slots
  - reservation slot time equal to channel end-end propagation delay
  - station with message to send posts reservation
  - reservation seen by all stations
- after reservation slots, message transmissions ordered by known priority

Summary of MAC protocols

- What do you do with a shared media?
  - Channel Partitioning, by time, frequency or code
    - Time Division, Code Division, Frequency Division
  - Random partitioning (dynamic),
    - ALOHA, S-ALOHA, CSMA, CSMA/CD
    - carrier sensing: easy in some technologies (wire), hard in others (wireless)
    - CSMA/CD used in Ethernet
  - Taking Turns
    - polling from a central cite, token passing