Chapter V: Link Layer

UG3 Computer Communications & Networks (COMN)

Myungjin Lee
myungjin.lee@ed.ac.uk

Slides copyright of Kurose and Ross
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 (at transport layer)!
  – 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
Where is the link layer implemented?

- in each and every host
- link layer implemented in “adaptor” (aka network interface card NIC) or on a chip
  - Ethernet card, 802.11 card; Ethernet chipset
  - implements link, physical layer
- attaches into host’s system buses
- combination of hardware, software, firmware
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
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

**Desired properties:**

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 \( \frac{R}{M} \)
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

FDM cable
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
**Slotted ALOHA**

**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
- clock synchronization
**Slotted ALOHA: efficiency**

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

- **suppose**: $N$ nodes with many frames to send, each transmits in slot with probability $p$
- prob that given node has success in a slot $= p(1-p)^{N-1}$
- prob 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:
  
  \[
  \text{max efficiency} = \frac{1}{e} = .37
  \]

**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.
CSMA collisions

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

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

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

![Spatial layout of nodes](image)
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
CSMA/CD (collision detection)

spatial layout of nodes

time

t_0

t_1

collision detect/abort time
Ethernet CSMA/CD algorithm

Create a frame with a datagram from Network layer

Channel idle?

Yes
Start to transmit the frame

No collision?

No
Abort transmission

Yes

Binary (exponential) backoff

After $m$th collision, NIC chooses $K$ at random from $\{0, 1, 2, \ldots, 2^m - 1\}$. NIC waits $K$ time units (one time unit = 512 bit times).

Longer backoff interval with more collisions
CSMA/CD efficiency

- $T_{prop} = \text{max prop delay between 2 nodes in LAN}$
- $t_{trans} = \text{time to transmit max-size frame}$

\[
\text{efficiency} = \frac{1}{1 + 5 \frac{t_{prop}}{t_{trans}}}
\]

- Efficiency goes to 1
  - as $t_{prop}$ goes to 0
  - as $t_{trans}$ goes to infinity
- Better performance than ALOHA: simple, cheap, decentralized!
“Taking turns” MAC protocols

channel partitioning MAC protocols:
  – share channel *efficiently* and *fairly* 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!
“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)
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
MAC addresses and ARP

• 32-bit IP address:
  – network-layer address for interface
  – used for layer 3 (network layer) forwarding

• MAC (or LAN or physical or Ethernet) address:
  – used ‘locally’ to get frame from one interface to another physically-connected interface (same network, in IP-addressing sense)
  – 48 bit MAC address (for most LANs) burned in NIC ROM, also sometimes software settable
  – e.g.: 1A-2F-BB-76-09-AD

  hexadecimal (base 16) notation
  (each “number” represents 4 bits)
LAN addresses and ARP

each adapter on LAN has unique *LAN* (MAC) address

LAN (wired or wireless)

adapter

1A-2F-BB-76-09-AD

71-65-F7-2B-08-53

58-23-D7-FA-20-B0

0C-C4-11-6F-E3-98

32
LAN addresses (more)

- MAC address allocation administered by IEEE
- manufacturer buys portion of MAC address space (to assure uniqueness)
- analogy:
  - MAC address: like Social Security Number
  - IP address: like postal address
- **MAC flat address** ➔ portability
  - can move LAN card from one LAN to another
- **IP hierarchical address** ➔ *not* portable
  - address depends on IP subnet to which node is attached
ARP: address resolution protocol

Question: how to determine interface’s MAC address, knowing its IP address?

ARP table: each IP node (host, router) on LAN has table

- IP/MAC address mappings for some LAN nodes:
  
  < IP address; MAC address; TTL>

- TTL (Time To Live): time after which address mapping will be forgotten (typically 20 min)
ARP protocol: same LAN

- A wants to send datagram to B
  - B’s MAC address not in A’s ARP table
- A broadcasts ARP query packet, containing B's IP address
  - dest MAC address = FF-FF-FF-FF-FF-FF
  - all nodes on LAN receive ARP query
- B receives ARP packet, replies to A with its (B's) MAC address
  - frame sent to A’s MAC address (unicast)

- A caches (saves) IP-to-MAC address pair in its ARP table until information becomes old (times out)
  - soft state: information that times out (goes away) unless refreshed

- ARP is “plug-and-play”:
  - nodes create their ARP tables without intervention from net administrator
“dominant” wired LAN technology:

- cheap $20 for NIC
- first widely used LAN technology
- simpler, cheaper than token LANs and ATM
- kept up with speed race: 10 Mbps – 10 Gbps
Ethernet: physical topology

- **bus**: popular through mid 90s
  - all nodes in same collision domain (can collide with each other)
- **star**: prevails today
  - active *switch* in center
  - each “spoke” runs a (separate) Ethernet protocol (nodes do not collide with each other)
Ethernet frame structure

sending adapter encapsulates IP datagram (or other network layer protocol packet) in Ethernet frame

**preamble:**
- 7 bytes with pattern 10101010 followed by one byte with pattern 10101011
- used to synchronize receiver, sender clock rates
Ethernet frame structure (more)

- **addresses**: 6 byte source, destination MAC addresses
  - if adapter receives frame with matching destination address, or with broadcast address (e.g., ARP packet), it passes data in frame to network layer protocol
  - otherwise, adapter discards frame
- **type**: indicates higher layer protocol (mostly IP but others possible, e.g., Novell IPX, AppleTalk)
- **CRC**: cyclic redundancy check at receiver
  - error detected: frame is dropped

<table>
<thead>
<tr>
<th>preamble</th>
<th>dest. address</th>
<th>source address</th>
<th>data (payload)</th>
<th>CRC</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>dest. address</td>
<td>source address</td>
<td>data (payload)</td>
<td>CRC</td>
</tr>
</tbody>
</table>
Ethernet: unreliable, connectionless

- **connectionless**: no handshaking between sending and receiving NICs
- **unreliable**: receiving NIC doesn’t send acks or nacks to sending NIC
  - data in dropped frames recovered only if initial sender uses higher layer rdt (e.g., TCP), otherwise dropped data lost
- Ethernet’s MAC protocol: unslotted *CSMA/CD with binary backoff*
many different Ethernet standards
- common MAC protocol and frame format
- different speeds: 2 Mbps, 10 Mbps, 100 Mbps, 1Gbps, 10G bps
- different physical layer media: fiber, cable
Ethernet switch

• link-layer device: takes an active role
  – store, forward Ethernet frames
  – examine incoming frame’s MAC address, selectively forward frame to one-or-more outgoing links when frame is to be forwarded on segment (link), uses CSMA/CD to access segment

• transparent
  – hosts are unaware of presence of switches

• plug-and-play, self-learning
  – switches do not need to be configured
Switch: *multiple* simultaneous transmissions

- hosts have dedicated, direct connection to switch
- switches buffer packets
- Ethernet protocol used on *each* incoming link, but no collisions; full duplex
  - each link is its own collision domain
- **switching**: A-to-A’ and B-to-B’ can transmit simultaneously, without collisions

---

Switch with six interfaces (1,2,3,4,5,6)
**Switch forwarding table**

**Q:** how does switch know A’ reachable via interface 4, B’ reachable via interface 5?

- **A:** each switch has a switch table, each entry:
  - (MAC address of host, interface to reach host, time stamp)
  - looks like a routing table!

**Q:** how are entries created, maintained in switch table?

- something like a routing protocol?
Switch: self-learning

- switch *learns* which hosts can be reached through which interfaces
  - when frame received, switch "learns" location of sender: incoming LAN segment
  - records sender/location pair in switch table

<table>
<thead>
<tr>
<th>MAC addr</th>
<th>interface</th>
<th>TTL</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1</td>
<td>60</td>
</tr>
</tbody>
</table>

Switch table *(initially empty)*
Switch: frame filtering/forwarding

when frame received at switch:

1. record incoming link, MAC address of sending host
2. index switch table using MAC destination address
3. if entry found for destination
   then {
     if destination on segment from which frame arrived
     then drop frame
     else forward frame on interface indicated by entry
   }
else flood  /* forward on all interfaces except arriving interface */
Self-learning, forwarding: example

- frame destination, A', location unknown: "flood"
- destination A location known: "selectively send on just one link"

Switch table (initially empty)

<table>
<thead>
<tr>
<th>MAC addr</th>
<th>interface</th>
<th>TTL</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1</td>
<td>60</td>
</tr>
<tr>
<td>A'</td>
<td>4</td>
<td>60</td>
</tr>
</tbody>
</table>
Interconnecting switches

• switches can be connected together

Q: sending from A to G - how does $S_1$ know to forward frame destined to G via $S_4$ and $S_3$?

A: self learning! (works exactly the same as in single-switch case!)
Self-learning multi-switch example

Suppose C sends frame to I, I responds to C

Q: show switch tables and packet forwarding in S₁, S₂, S₃, S₄
Switches vs. routers

Both are store-and-forward:

- **routers**: network-layer devices (examine network-layer headers)
- **switches**: link-layer devices (examine link-layer headers)

Both have forwarding tables:

- **routers**: compute tables using routing algorithms, IP addresses
- **switches**: learn forwarding table using flooding, learning, MAC addresses
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

---

**Parity Checking Example:**

```
<table>
<thead>
<tr>
<th>1 0 1 0 1</th>
<th>1 0 1 0 1</th>
<th>1 0 1 0 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 1 1 0 0</td>
<td>1 0 1 0 0</td>
<td>1 0 1 0 0</td>
</tr>
<tr>
<td>0 1 1 1 0 1</td>
<td>0 1 1 1 0 1</td>
<td>0 1 1 1 0 1</td>
</tr>
<tr>
<td>0 0 1 0 1 0</td>
<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)

\[ D \times 2^r \text{ XOR } R \]  
\( \text{bit pattern} \)

\( D: \) data bits to be sent  
\( R: \) CRC bits
Cyclic redundancy check

- All CRC calculations are done in modulo-2 arithmetic without carries in addition or borrows in subtraction

\[
1011 \ XOR \ 0101 = 1110 \quad \text{equivalent to} \quad 1011 \ − \ 0101 = 1110
\]

\[
\text{equivalent to} \quad 1011 \ + \ 0101 = 1110
\]

- Multiplication and division are the same as in base-2 arithmetic, except that any required addition or subtraction is done without carries and borrows
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:

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