Part 5: Data Link Layer

CSE 3461/5461
Reading: Chapter 5, Kurose and Ross (≤ 6th ed.);
Chapter 6, Kurose and Ross (7th ed.)
Part 5: 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
  - MPLS
  - Datacenter networking
Outline

• Overview: Link Layer Services
• Error Detection and Correction
• Multiple Access Control (MAC) Sublayer
• Ethernet
• Switches and Switch Self-Learning
• Multiprotocol Label Switching (MPLS)
• Datacenter Networking
• Synthesis: Requesting a Web Page
• Tools of the Trade
Link Layer: Setting the Context (1)
Link Layer: Setting the Context (2)

- Two *physically connected* devices:
  - Host-router, router-router, host-host, router-switch, etc.
- Unit of data: *frame*
Link Layer Services (1)

• **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 (2)

• **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
Outline

• Overview: Link Layer Services
• Error Detection and Correction
• Multiple Access Control (MAC) Sublayer
• Ethernet and LAN Technologies
• Switches and Switch Self-Learning
• Multiprotocol Label Switching (MPLS)
• Datacenter Networking
• Synthesis: Requesting a Web Page
• Tools of the Trade
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

| 0111000110101011 | 0 |

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

<table>
<thead>
<tr>
<th>d₁₁,₁</th>
<th>⋯</th>
<th>d₁₁,j</th>
</tr>
</thead>
<tbody>
<tr>
<td>d₂₁,₁</td>
<td>⋯</td>
<td>d₂₁,j</td>
</tr>
<tr>
<td>⋯</td>
<td>⋯</td>
<td>⋯</td>
</tr>
<tr>
<td>dᵢ,₁</td>
<td>⋯</td>
<td>dᵢ,j</td>
</tr>
<tr>
<td>dᵢ₊₁,₁</td>
<td>⋯</td>
<td>dᵢ₊₁,j</td>
</tr>
</tbody>
</table>

#### Data Bits

- d₁₁,₁
- d₂₁,₁
- ⋯
- dᵢ,₁
- dᵢ₊₁,₁

#### Parity Bits

- d₁,₁
- d₂,₁
- ⋯
- dᵢ,₁
- dᵢ₊₁,₁

#### Row Parity

- d₁, j+1
- d₂, j+1
- ⋯
- dᵢ, j+1
- dᵢ₊₁, j+1

#### Column Parity

- d₁,₁
- d₂,₁
- ⋯
- dᵢ,₁
- dᵢ₊₁,₁

### Examples

#### No Errors

| 1 0 1 0 1 1 |
| 1 1 1 1 0 0 |
| 0 1 1 1 0 1 |
| 0 0 1 0 1 0 |

#### Single Bit Error

- Data: 101011
- Parity: 101011
- Error: 01100
- Corrected: 01100

- Data: 01100
- Parity: 01100
- Error: 10101
- Corrected: 01010
**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 ….
Checksum: 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
  - $\langle D, R \rangle$ exactly divisible by $G$ (modulo 2)
  - Receiver knows $G$, divides $\langle D, R \rangle$ by $G$. If non-zero remainder: error detected!
  - Can detect all burst errors less than $r + 1$ bits
- Widely used in practice (ATM, HDCL)

\[ D \text{: data bits to be sent} \quad R \text{: CRC bits} \]

\[ D \ast 2^r \text{ XOR } R \quad \text{mathematical formula} \]
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 reminder \( R \)

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

• Overview: Link Layer Services
• Error Detection and Correction
• Multiple Access Control (MAC) Sublayer
• Ethernet and LAN Technologies
• Switches and Switch Self-Learning
• Multiprotocol Label Switching (MPLS)
• Datacenter Networking
• Synthesis: Requesting a Web Page
• Tools of the Trade
Multiple Access Links & Protocols

Three types of “links”:

• **Point-to-point** (single wire, e.g. PPP, SLIP)
• **Broadcast** (shared wire or medium; e.g., Ethernet, Wi-Fi, etc.)
• **Switched** (e.g., switched Ethernet)
Multiple Access (MAC) 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
MAC Protocols: A Taxonomy

Three broad classes:

• **Channel Partitioning**
  – TDMA: time division multiple access
  – FDMA: frequency division multiple access
  – CDMA (Code Division Multiple Access) Read (§6.2.1)

• **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

![Diagram of TDMA access]

- 6-slot frame
- 1, 3, 4
- 1, 3, 4
- 1, 3, 4
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

![Diagram of frequency bands and time]
Random Access Protocols

• When node has packet to send
  – Transmit at full channel data rate $R$.
  – No \textit{a priori} coordination among nodes

• Two or more transmitting nodes $\Rightarrow$ “collision”,

• \textbf{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 and ALOHA
  – CSMA and CSMA/CD
Slotted ALOHA (1)

**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 probability $p$ 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
- Nodes may be able to detect collision in less than time to transmit packet
- Clock synchronization
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/Slotted ALOHA efficiency

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

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Efficiency</th>
</tr>
</thead>
<tbody>
<tr>
<td>Slotted ALOHA</td>
<td>1/e</td>
</tr>
<tr>
<td>Pure ALOHA</td>
<td>1/2e</td>
</tr>
</tbody>
</table>

The derivation is given in the textbook [Sect. 5.3 (5th, 6th ed.); Sect. 6.3 (7th ed.)]; it is a homework problem.

**Hint**: \( \lim_{N \to \infty} (1 - 1/N)^N = 1/e \)
**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!
Collisions can still occur:
Propagation delay means two nodes may not year hear each other’s transmission

Collision:
enentire packet transmission time wasted

Note:
role of distance and propagation delay in determining collision probability

Spatial layout of nodes along Ethernet
CSMA/CD (Collision Detection) (1)

**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 (2)
“Taking Turns” MAC Protocols (1)

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!
“Taking Turns” MAC Protocols (2)

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)
Summary of MAC Protocols

• What do you do with a shared medium?
  – Channel partitioning via 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
Outline

• Overview: Link Layer Services
• Error Detection and Correction
• Multiple Access Control (MAC) Sublayer
• **Ethernet and LAN Technologies**
• Switches and Switch Self-Learning
• Multiprotocol Label Switching (MPLS)
• Datacenter Networking
• Synthesis: Requesting a Web Page
• Tools of the Trade
LAN Technologies

Data link layer so far:
– Services, error detection/correction, multiple access

Next: LAN technologies
– Addressing
– Ethernet
– Hubs, bridges, switches
– 802.11
– PPP
– ATM
LAN Addresses and ARP

32-bit IP address:
• *Network-layer* address
• Used to get datagram to destination network (recall IP network definition)

LAN (or MAC or physical) address:
• Used to get datagram from one interface to another physically-connected interface (same network)
• 48 bit MAC address (for most LANs) burned in the adapter ROM
LAN Addressing (1)

Each adapter on LAN has unique LAN address
LAN Addressing (2)

- MAC address allocation administered by IEEE
- Manufacturer buys portion of MAC address space (to assure uniqueness)
- Analogy:
  (a) MAC address: like Social Security Number
  (b) IP address: like postal address
- MAC flat address $\Rightarrow$ portability
  - Can move LAN card from one LAN to another
- IP hierarchical address NOT portable
  - Depends on network to which one attaches
Recall Earlier Routing Discussion

Starting at $A$, given IP datagram addressed to $B$:

- Look up net. address of $B$, find $B$ on same net. as $A$
- **Link layer sends datagram to $B$ inside link-layer frame**
ARP: Address Resolution Protocol (1)

Question: How to determine MAC address of B given B’s IP address?

- Each IP node (Host, Router) on LAN has **ARP** module, table
- **ARP Table**: IP/MAC address mappings for some LAN nodes
  $< \text{IP address; MAC address; TTL}>$
  $< \text{.................................}>$
  - **TTL** (Time To Live): time after which address mapping will be forgotten (typically 20 min)
ARP (2)

- $A$ knows $B$’s IP address, wants to learn $B$’s physical address
- $A$ broadcasts ARP query pkt containing $B$’s IP address
  - All machines on LAN receive ARP query
- $B$ receives ARP packet, replies to $A$ with its ($B$’s) physical layer address
- $A$ caches (saves) IP-to-physical address pairs until information becomes old (times out)
  - *Soft state*: information that times out (goes away) unless refreshed
Routing to Another LAN (1)

Walkthrough: routing from $A$ to $B$ via $R$

- In routing table at source Host, find router 111.111.111.110
- In ARP table at source, find MAC address E6-E9-00-17-BB-4B, etc.
Routing to Another LAN (2)

- $A$ creates IP packet with source $A$, destination $B$
- $A$ uses ARP to get $R$’s physical layer address for 111.111.111.110
- $A$ creates Ethernet frame with $R$’s physical address as dest, Ethernet frame contains $A$-to-$B$ IP datagram
- $A$’s data link layer sends Ethernet frame
- $R$’s data link layer receives Ethernet frame
- $R$ removes IP datagram from Ethernet frame, sees it’s destined to $B$
- $R$ uses ARP to get $B$’s physical layer address
- $R$ creates frame containing $A$-to-$B$ IP datagram, sends it to $B$
“Dominant” LAN technology (aka IEEE 802.3):
• Cheap $20 for 100Mbps!
• First wildly used LAN technology
• Simpler, cheaper than token LANs and ATM
• Kept up with speed race: 10, 100, Mbps; 1, 10, 40, 100 Gbps
Ethernet Frame Structure (1)

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 (2)

- **Addresses:** 6 bytes, frame is received by all adapters on a LAN and dropped if address does not match
- **Type:** indicates the higher layer protocol, mostly IP but others may be supported such as Novell IPX and AppleTalk)
- **CRC:** checked at receiver, if error is detected, the frame is simply dropped
Ethernet’s CSMA/CD (1)

1: **START:** Sense channel
2: **if** channel is idle **then**
3: transmit and monitor the channel
4: **if** another transmission is detected **then**
5: abort and send jam signal
6: update \textit{collisions}
7: delay as required by exponential backoff algorithm
8: \textit{goto} START
9: **else**
10: \textit{collisions} \leftarrow 0 \qquad // done with transmission
11: **end if**
12: **else**
13: wait until ongoing transmission is over
14: \textit{goto} START
15: **end if**
Ethernet’s CSMA/CD (2)

Jam Signal: make sure all other transmitters are aware of collision; 48 bits

Exponential Backoff:

• **Goal**: adapt retransmission attempts to estimated current load
  – Heavy load: random wait will be longer
• First collision: choose $k$ from $\{0, 1\}$; delay is $k \times 512$ bit transmission times
• After second collision: choose $k$ from $\{0, 1, 2, 3\}$…
• After ten or more collisions, choose $k$ from $\{0, 1, \ldots, 1023\}$

In general: After $k$-th collision, choose $k$ from $\{0, \ldots, 2^k - 1\}$, where $1 \leq k \leq 10$; stations give up after 10 collisions.
Ethernet Technologies: 10Base2

- **10**: 10 Mbps; **2**: under 200 meters max cable length
- Thin coaxial cable in a bus topology

- Repeaters used to connect up to multiple segments
- Repeater repeats bits it hears on one interface to its other interfaces: physical layer device only!
10BaseT and 100BaseT (1)

- 10/100 Mbps rate; latter called “Fast Ethernet”
- T stands for Twisted Pair
- Hub to which nodes are connected by twisted pair, thus “star topology”
- CSMA/CD implemented at hub
10BaseT and 100BaseT (2)

- Max distance from node to Hub is 100 meters
- Hub can disconnect “jabbering adapter”
- Hub can gather monitoring information, statistics for display to LAN administrators
Hubs (1)

- Physical Layer devices: essentially repeaters operating at bit levels: repeat received bits on one interface to all other interfaces

- Hubs can be arranged in a hierarchy (or multi-tier design), with backbone hub at its top
Hubs (2)

- Each connected LAN referred to as LAN **segment**
- Hubs **do not isolate** collision domains: node may collide with any node residing at any segment in LAN
- Hub Advantages:
  - Simple, inexpensive device
  - Multi-tier provides graceful degradation: portions of the LAN continue to operate if one hub malfunctions
  - Extends maximum distance between node pairs (100 m per hub)
Hub Limitations

• Single collision domain results in no increase in max throughput
  – Multi-tier throughput same as single segment throughput
• Individual LAN restrictions pose limits on number of nodes in same collision domain and on total allowed geographical coverage
• Cannot connect different Ethernet types (e.g., 10BaseT and 100baseT)
Token Passing: IEEE 802.5 Standard (1)

- 4 Mbps
- Max token holding time: 10 ms, limiting frame length

```
SD  AC  FC
```
```
SD  AC  FC  dest addr  src addr  data  checksum  ED  FS
```

- **SD, ED** mark start, end of packet
- **AC**: access control byte:
  - **Token bit**: value 0 means token can be seized, value 1 means data follows FC
  - **Priority bits**: priority of packet
  - **Reservation bits**: station can write these bits to prevent stations with lower priority packet from seizing token after token becomes free
Token Passing: IEEE 802.5 Standard (2)

- **FC**: frame control used for monitoring and maintenance
- **Source, destination address**: 48 bit physical address, as in Ethernet
- **Data**: packet from network layer
- **Checksum**: CRC
- **FS**: frame status: set by destination, read by sender
  - Set to indicate destination up, frame copied OK from ring
  - DLC-level ACKing
Outline

• Overview: Link Layer Services
• Error Detection and Correction
• Multiple Access Control (MAC) Sublayer
• Ethernet and LAN Technologies
• Switches and Switch Self-Learning
• Multiprotocol Label Switching (MPLS)
• Datacenter Networking
• Synthesis: Requesting a Web Page
• Tools of the Trade
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, 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 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 dest. A’, location unknown: **flood**
- Destination A location known: **selectively send on just one link**

<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>

Switch table (initially empty)
Interconnecting Switches

• Switches can be connected together

Q: Sending from A to G – how does $S_1$ know to forward frame destined to F 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.
Institutional Network

To external network

Router

Mail server

Web server

IP subnet
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
VLANs: Motivation

Consider:

- CS user moves office to EE, but wants connect to CS switch?
- Single broadcast domain:
  - All layer-2 broadcast traffic (ARP, DHCP, unknown location of destination MAC address) must cross entire LAN
  - Security/privacy, efficiency issues
VLANs

Virtual Local Area Network

Switch(es) supporting VLAN capabilities can be configured to define multiple virtual LANS over single physical LAN infrastructure.

Port-Based VLAN: switch ports grouped (by switch management software) so that *single* physical switch …

… operates as *multiple* virtual switches
Port-Based VLANs

- **Traffic isolation:** frames to/from ports 1-8 can only reach ports 1-8
  - Can also define VLAN based on MAC addresses of endpoints, rather than switch port

- **Dynamic membership:** ports can be dynamically assigned among VLANs

- **Forwarding between VLANs:** done via routing (just as with separate switches)
  - In practice, vendors sell combined switches plus routers
VLANs Spanning Multiple Switches

- **Trunk port**: carries frames between VLANS defined over multiple physical switches
  - Frames forwarded within VLAN between switches can’t be vanilla 802.1 frames (must carry VLAN ID info)
  - 802.1q protocol adds/removed additional header fields for frames forwarded between trunk ports

Electrical Engineering (VLAN ports 1-8)  
Computer Science (VLAN ports 9-15)  
Ports 2,3,5 belong to EE VLAN  
Ports 4,6,7,8 belong to CS VLAN
802.1Q VLAN Frame Format

Note: Frame fields are not to scale.
Outline

• Overview: Link Layer Services
• Error Detection and Correction
• Multiple Access Control (MAC) Sublayer
• Ethernet and LAN Technologies
• Switches and Switch Self-Learning
• Multiprotocol Label Switching (MPLS)
• Datacenter Networking
• Synthesis: Requesting a Web Page
• Tools of the Trade
Interconnecting LANs

**Q:** Why not just one big LAN?

- Limited amount of supportable traffic: on single LAN, all stations must share bandwidth
- Limited length: 802.3 specifies maximum cable length
- Large “collision domain” (can collide with many stations)
- Limited number of stations: 802.5 have token passing delays at each station
Multiprotocol Label Switching (MPLS)

- Initial goal: high-speed IP forwarding using fixed length label (instead of IP address)
  - Fast lookup using fixed length identifier (rather than shortest prefix matching)
  - Borrowing ideas from Virtual Circuit (VC) approach
  - But IP datagrams still keep their IP addresses!

<table>
<thead>
<tr>
<th>PPP or Ethernet header</th>
<th>MPLS header</th>
<th>IP header</th>
<th>remainder of link-layer frame</th>
</tr>
</thead>
<tbody>
<tr>
<td>Label</td>
<td>Exp</td>
<td>S</td>
<td>TTL</td>
</tr>
<tr>
<td>20</td>
<td>3</td>
<td>1</td>
<td>5</td>
</tr>
</tbody>
</table>
MPLS-Capable Routers

• A.k.a. label-switched router
• Forward packets to outgoing interface based only on label value (*don’t inspect IP address*)
  – MPLS forwarding table distinct from IP forwarding tables
• **Flexibility:** MPLS forwarding decisions can *differ* from those of IP
  – Use destination *and* source addresses to route flows to same destination differently (traffic engineering)
  – Re-route flows quickly if link fails: pre-computed backup paths (useful for VoIP)
**MPLS vs. IP Paths (1)**

- **IP routing:** Path to destination determined by destination address alone

![Diagram of IP routing](image)
MPLS vs. IP Paths (2)

- **IP routing:** path to destination determined by destination address alone

- **MPLS routing:** path to destination can be based on source *and* dest. address
  - *Fast reroute:* precompute backup routes in case of link failure
MPLS Signaling

- Modify OSPF, IS-IS link-state flooding protocols to carry info used by MPLS routing,
  - e.g., link bandwidth, amount of “reserved” link bandwidth
- Entry MPLS router uses RSVP-TE signaling protocol to set up MPLS forwarding at downstream routers
MPLS Forwarding Tables

<table>
<thead>
<tr>
<th>In Label</th>
<th>Out Label</th>
<th>Dest</th>
<th>Out Interface</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>A</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>D</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>A</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>In Label</th>
<th>Out Label</th>
<th>Dest</th>
<th>Out Interface</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>6</td>
<td>A</td>
<td>1</td>
</tr>
<tr>
<td>12</td>
<td>9</td>
<td>D</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>In Label</th>
<th>Out Label</th>
<th>Dest</th>
<th>Out Interface</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>6</td>
<td>A</td>
<td>0</td>
</tr>
</tbody>
</table>

In         Out                 Out
Label     Label   Dest    Interface

R6

R5

R4

R3

R2

D

A
Outline

• Overview: Link Layer Services
• Error Detection and Correction
• Multiple Access Control (MAC) Sublayer
• Ethernet and LAN Technologies
• Switches and Switch Self-Learning
• Multiprotocol Label Switching (MPLS)
• Datacenter Networking
• Synthesis: Requesting a Web Page
• Tools of the Trade
Datacenter Networks (1)

• 10,000s–100,000s of thousands of hosts, often closely coupled, in close proximity:
  – E-business (e.g. Amazon)
  – Content servers (e.g., YouTube, Akamai, Apple, Microsoft)
  – Search engines, data mining (e.g., Google)

• Challenges:
  – Multiple applications, each serving massive numbers of clients
  – Managing/balancing load, avoiding processing, networking, data bottlenecks

Inside a 40-ft Microsoft container,
Chicago data center
Datacenter Networks (2)

- **Load balancer: application-layer routing**
  - Receives external client requests
  - Directs workload within data center
  - Returns results to external client (hiding datacenter internals from client)
Datacenter Networks (3)

- Rich interconnection among switches, racks:
  - Increased throughput between racks (multiple routing paths possible)
  - Increased reliability via redundancy
Outline

• Overview: Link Layer Services
• Error Detection and Correction
• Multiple Access Control (MAC) Sublayer
• Ethernet and LAN Technologies
• Switches and Switch Self-Learning
• Multiprotocol Label Switching (MPLS)
• Datacenter Networking
• Synthesis: Requesting a Web Page
• Tools of the Trade
Synthesis: A Day in the Life of a Web Request

• Journey down protocol stack complete!
  – Application, transport, network, link

• Putting-it-all-together: synthesis!
  – Goal: identify, review, understand protocols (at all layers) involved in seemingly simple scenario: requesting WWW page
  – Scenario: student attaches laptop to campus network, requests/receives www.google.com
A Day in the Life: Scenario

School network 68.80.2.0/24

Comcast network 68.80.0.0/13

Google’s network 64.233.160.0/19

DNS server

Web server 64.233.169.105

web page

browser
A Day in the Life… Connecting to the Internet (1)

- Connecting laptop needs to get its own IP address, addr of first-hop router, addr of DNS server: use **DHCP**

- DHCP request **encapsulated** in **UDP**, encapsulated in **IP**, encapsulated in **802.3** Ethernet

- Ethernet frame **broadcast** (dest: FFFFFFFF000000000) on LAN, received at router running **DHCP** server

- Ethernet **demuxed** to IP demuxed, UDP demuxed to DHCP
A Day in the Life… Connecting to the Internet (2)

• DHCP server formulates **DHCP ACK** containing client’s IP address, IP address of first-hop router for client, name & IP address of DNS server

• Encapsulation at DHCP server, frame forwarded (**switch learning**) through LAN, demultiplexing at client

• DHCP client receives DHCP ACK reply

Client now has IP address, knows name & addr of DNS server, IP address of its first-hop router
A Day in the Life… ARP (Before DNS, HTTP)

- Before sending **HTTP** request, need IP address of `www.google.com`: **DNS**
- DNS query created, encapsulated in UDP, encapsulated in IP, encapsulated in Eth. To send frame to router, need MAC address of router interface: **ARP**
  - **ARP query** broadcast, received by router, which replies with **ARP reply** giving MAC address of router interface
  - Client now knows MAC address of first hop router, so can now send frame containing DNS query
A Day in the Life… Using DNS

- IP datagram containing DNS query forwarded via LAN switch from client to 1st hop router
- IP datagram forwarded from campus network into Comcast network, routed (tables created by RIP, OSPF, IS-IS and/or BGP routing protocols) to DNS server
- Demuxed to DNS server
- DNS server replies to client with IP address of www.google.com
A Day in the Life…TCP Connection Carrying HTTP

- To send HTTP request, client first opens **TCP socket** to web server
- **TCP SYN segment** (step 1 in 3-way handshake) inter-domain routed to web server
- Web server responds with **TCP SYNACK** (step 2 in 3-way handshake)
- **TCP connection established!**
A Day in the Life… HTTP Request/Reply

- Web page **finally (!!!) displayed**

- **HTTP request** sent into TCP socket
- IP datagram containing HTTP request routed to **www.google.com**
- Web server responds with **HTTP reply** (containing web page)
- IP datagram containing HTTP reply routed back to client
Outline

• Overview: Link Layer Services
• Error Detection and Correction
• Multiple Access Control (MAC) Sublayer
• Ethernet and LAN Technologies
• Switches and Switch Self-Learning
• Multiprotocol Label Switching (MPLS)
• Datacenter Networking
• Synthesis: Requesting a Web Page
• Tools of the Trade
Tools of the Trade: Linux (1)

- **arp**: View, manage ARP table (net-tools package, [https://sourceforge.net/projects/net-tools/](https://sourceforge.net/projects/net-tools/))

/home/champion.17 # On zeta.cse.ohio-state.edu
% /sbin/arp -a
? (164.107.113.3) at 00:de:fb:58:f8:c1 [ether] on eth0
cse-sl7.coeit.osu.edu (164.107.113.26) at 00:50:56:95:41:f6 [ether] on eth0
? (164.107.113.18) at 00:13:72:54:cc:4f [ether] on eth0
cse-sl3.coeit.osu.edu (164.107.113.12) at 00:50:56:95:55:53 [ether] on eth0
cse-epsilon.coeit.osu.edu (164.107.113.21) at 00:13:72:54:c2:5b [ether] on eth0
? (164.107.113.1) at 00:00:0c:9f:f0:01 [ether] on eth0
cse-sl6.coeit.osu.edu (164.107.113.15) at 00:50:56:95:6c:57 [ether] on eth0
cse-sl4.coeit.osu.edu (164.107.113.13) at 00:50:56:95:20:aa [ether] on eth0
cse-sl2.coeit.osu.edu (164.107.113.11) at 00:50:56:95:1e:80 [ether] on eth0
cse-eta.coeit.osu.edu (164.107.113.23) at 00:13:72:54:c8:38 [ether] on eth0
? (164.107.113.2) at 00:de:fb:58:f8:41 [ether] on eth0
cse-sl11.coeit.osu.edu (164.107.113.10) at 00:50:56:95:13:58 [ether] on eth0
Tools of the Trade: Linux (2)

- **ethtool**: View, set Ethernet interface configuration

  ```bash
/home/champion.17 # Also on zeta
% /sbin/ethtool eth0
Settings for eth0:
  Supported ports: [ TP ]
  Supported link modes: 10baseT/Half 10baseT/Full 100baseT/Half 100baseT/Full 1000baseT/Full
  Supported pause frame use: No
  Supports auto-negotiation: Yes
  Advertised link modes: 10baseT/Half 10baseT/Full 100baseT/Half 100baseT/Full 1000baseT/Full
  Advertised pause frame use: No
  Advertised auto-negotiation: Yes
  Speed: 1000Mb/s
  Duplex: Full
  Port: Twisted Pair
  PHYAD: 0
  Transceiver: internal
  Auto-negotiation: on
  MDI-X: on (auto) . . .
```
Tools of the Trade: Linux (3)

• vconfig: Add, remove VLANs via Ethernet interfaces

/home/champion.17 # Also on zeta
% /sbin/vconfig
Expecting argc to be 3-5, inclusive.  Was: 1

Usage: add [interface-name] [vlan_id]
rem [vlan-name]
set_flag [interface-name] [flag-num] [0 | 1]
set_egress_map [vlan-name] [skb_priority] [vlan_qos]
set_ingress_map [vlan-name] [skb_priority] [vlan_qos]
set_name_type [name-type]

* The [interface-name] is the name of the ethernet card that hosts the VLAN you are talking about.
* The vlan_id is the identifier (0-4095) of the VLAN you are operating on.
* skb_priority is the priority in the socket buffer (sk_buff).
* vlan_qos is the 3 bit priority in the VLAN header
* name-type: . . .
* bind-type: . . .
* FLAGS: . . .
Tools of the Trade: Linux (4)

- Data center mgmt: Apache Mesos, https://mesos.apache.org/
- Distributed computing frameworks for clusters:
- Other resources:
Part 5: Summary

• Principles behind data link layer services:
  – Error detection, correction
  – Sharing a broadcast channel: multiple access
  – Link layer addressing, ARP

• Various link layer technologies
  – Ethernet
  – hubs, bridges, switches
  – IEEE 802.11 LANs
  – PPP
  – ATM
  – X.25, Frame Relay
  – MPLS
  – Datacenter Networking

• Journey down the protocol stack now **OVER!**