Welcome to

**CS 3516:**

Advanced Computer Networks

Prof. Yanhua Li

*Time: 9:00am - 9:50am M, T, R, and F*

*Location: Fuller 320*

*Fall 2016 A-term*

Office hours today
10-10:30AM AK 130
1-5PM TA office by John
Sample questions for final will be ready by tomorrow morning

We will review it next Tuesday

Final exam, next Thursday
Next Monday Quiz 8

*Topic will be announced by tomorrow morning*

Next Tuesday three deadlines

*Project 2*
*Project 3*
*Lab 3*
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

5.6 data center networking
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:
  - function: 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 address

- 1A-2F-BB-76-09-AD
- 71-65-F7-2B-08-53
- 58-23-D7-FA-20-B0
- 0C-C4-11-6F-E3-98
LAN addresses (more)

- MAC address allocation administered by IEEE
- manufacturer buys portion \(2^{24}\) of MAC address space (to assure uniqueness)
- analogy:
  - MAC address: like Social Security Number
  - IP address: like postal address
  - Domain Name: Person name
- 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-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*
Addressing: routing to another LAN

walkthrough: send datagram from A to B via R

- focus on addressing – at IP (datagram) and MAC layer (frame)
- assume A knows B’s IP address
- assume A knows IP address of first hop router, R (how?)
- assume A knows R’s MAC address (how?)

[Diagram showing network topology with IP and MAC addresses]

A

111.111.111.111
74-29-9C-E8-FF-55

111.111.111.112
CC-49-DE-D0-AB-7D

R

222.222.222.220
1A-23-F9-CD-06-9B

B

222.222.222.222
49-BD-D2-C7-56-2A

222.222.222.221
88-B2-2F-54-1A-0F

Link Layer 5-11
Addressing: routing to another LAN

- A creates IP datagram with IP source A, destination B
- A creates link-layer frame with R's MAC address as dest, frame contains A-to-B IP datagram
Addressing: routing to another LAN

- frame sent from A to R
- frame received at R, datagram removed, passed up to IP
Addressing: routing to another LAN

- R forwards datagram with IP source A, destination B
- R creates link-layer frame with B's MAC address as dest, frame contains A-to-B IP datagram
Addressing: routing to another LAN

- R forwards datagram with IP source A, destination B
- R creates link-layer frame with B's MAC address as dest, frame contains A-to-B IP datagram
Addressing: routing to another LAN

- R forwards datagram with IP source A, destination B
- R creates link-layer frame with B's MAC address as dest, frame contains A-to-B IP datagram
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
5.6 data center networking
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

**ATM: Asynchronous Transfer Mode**
**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**: (2 bytes) indicates higher layer protocol (mostly IP but others possible, e.g., Novell IPX, AppleTalk)

- **CRC**: (4 bytes) 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><strong>type</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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
5.5 link virtualization: MPLS
5.6 data center networking
5.7 a day in the life of a web request
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
- 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)*

Source: A
Dest: A'
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*

<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 $S_1, S_2, S_3, S_4$
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
Questions?