CS 43: Computer Networks
Switches and LANs

Kevin Webb
Swarthmore College
December 3, 2015
“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: 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 reliable delivery (e.g., TCP), otherwise dropped data lost
- Ethernet’s MAC protocol:
  
  *CSMA/CD with binary exponential backoff*
802.3 Ethernet standards: link & physical layers

- **Many** different Ethernet standards
  - Common MAC protocol and frame format
  - Speeds: 2 Mbps, 10 Mbps, 100 Mbps, 1Gbps, 10Gbps
  - Physical layer media: fiber, copper cable
**Ethernet frame structure**

Sender encapsulates IP datagram (or other network layer protocol packet) in **Ethernet frame**

<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>type</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**preamble:**

- 7 bytes with pattern 10101010 followed by one byte with pattern 10101011
Clock Synching

- Bits represented as voltages, either low or high
- We will read one bit per clock cycle
Clock Synching

• Bits represented as voltages, either low or high
• We will read one bit per clock cycle

Ideal receiver: Sample signal at regular interval.

For 1 Gbps Ethernet, ~1 nanosecond interval.
Clock Synching

- Bits represented as voltages, either low or high
- We will read one bit per clock cycle

Problem: receiver clock may not agree with sender!

Preamble let’s receiver see several 0 -> 1 -> 0 -> ... transitions.
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
A quick lab note...

• You will NOT see the preamble in the frames you receive.
  – (It also doesn’t count as part of the 1500 byte MTU)

• There are header structs defined in sr_protocol.h.

• First task upon receiving a packet: “Is this for me?”
  – Compare dest address of packet against address of interface that received it.
  – Function already exists for this (ether_to_me)

<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></td>
<td></td>
<td></td>
<td>type</td>
</tr>
</tbody>
</table>
MAC Addresses

• 32-bit IP address:
  – *network-layer* address for interface
  – used by network layer for end-to-end routing

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

  hexadecimal (base 16) notation
  (each digit represents 4 bits)
MAC Addresses

Each interface/adapter on LAN has unique *MAC* address

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

• 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 & LAN communication

• A wants to send datagram to B. A knows B’s IP address.
  – 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, most ignore it

• B receives ARP packet, replies to A with its (B's) MAC address
  – frame sent to A’s MAC address (unicast)

• A caches IP-to-MAC address pair in its ARP table until timeout
  – soft state: times out unless refreshed, can be reacquired
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

— Note: there’s a router here, these are separate subnets
Addressing: routing to another LAN

Walkthrough: send datagram from A to B via R

—Who do we address the datagram to (IP destination)?

—Who do we forward it to on the first hop?
How does A learn the IP address of R?

A. ARP
B. DHCP
C. IP
D. Routing protocol
How does A learn the MAC address of R?

A. ARP
B. DHCP
C. IP
D. Routing protocol
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

MAC src: 1A-23-F9-CD-06-9B
MAC dest: 49-BD-D2-C7-56-2A
IP src: 111.111.111.111
IP dest: 222.222.222.222

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

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

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

IP
Eth
Phy
Physical Topology: Bus

- **Bus**: popular through mid 90s
  - all nodes in same collision domain (transmissions collide with each other)

**bus**: coaxial cable
Physical Topology: Star

- **Hub** in the center:
  - broadcasts all messages to all hosts
  - retransmits on collisions
  - often considered a physical layer device (like a bus wire)
Physical Topology: Star (Switched)

- **Switch**: prevails today
  - each “spoke” runs a (separate) Ethernet protocol (nodes do not collide with each other)
  - Full duplex: No collisions on spoke
Institutional Network (Tree)

To external network

router

mail server

web server

IP subnet
Ethernet switch

- link-layer device: takes an *active* role
  - store, forward Ethernet frames
  - examines incoming frame’s MAC address, *selectively* forwards 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-D and B-to-E can transmit simultaneously, without collisions

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

Q: how does switch know D reachable via interface 4, E reachable via interface 5?

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

Switch with six interfaces (1, 2, 3, 4, 5, 6)
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)
Self-learning, forwarding: example

- frame destination, D, location unknown:
- destination A location known: selectively send on just one link

```
 MAC addr | interface | TTL  
 A        | 1         | 60   
 D        | 4         | 60   
```

Switch table (initially empty)
Suppose the switch receives a packet from A to G. (Assume it knows what interface both A and G are on.) It should...

A. Flood the packet
B. Throw the packet away
C. Send the packet out on interface 1
D. Do something else
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 {
   if destination on segment from which frame arrived
   drop frame
   else
   forward frame on interface indicated by entry
}
else flood  /* forward on all interfaces except arriving interface */
Interconnecting switches

- Switches often connected to form trees.
Sending from A to G - how does $S_1$ know to forward frame destined to G via $S_4$ and $S_3$?

A. A network administrator will need to configure this.

B. $S_1$ will automatically learn the entire path.

C. $S_1$ will learn to send packets to G on the interface that leads to $S_4$. 
Eve wants to snoop and read all of the frames being sent to anyone on the LAN. She will NOT be able to do this on a

A. Bus

B. Hub

C. Switch

D. She can do this on all of these
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
Reading

• Security
  – Sections 1.6, 8.2.2