Computer Networking

Ethernet
Hubs and Switches

Ethernet
“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

Metcalfe’s Ethernet sketch
Star topology

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

![Bus vs. Star Topology Diagram]

Ethernet Frame Structure

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

![Ethernet Frame Structure Diagram]

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 bytes
  - 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:** checked at receiver, if error is detected, frame is dropped

![Ethernet Frame Structure Diagram]

**Ethernet: Unreliable, connectionless**

- **connectionless:** No handshaking between sending and receiving NICs
- **unreliable:** receiving NIC doesn’t send acks or nacks to sending NIC
  - stream of datagrams passed to network layer can have gaps (missing datagrams)
  - gaps will be filled if app is using TCP
  - otherwise, app will see gaps
- **Ethernet’s MAC protocol:** unslotted CSMA/CD
Ethernet CSMA/CD algorithm

1. NIC receives datagram from network layer, creates frame
2. If NIC senses channel idle, starts frame transmission If NIC senses channel busy, waits until channel idle, then transmits
3. If NIC transmits entire frame without detecting another transmission, NIC is done with frame!
4. If NIC detects another transmission while transmitting, aborts and sends jam signal
5. After aborting, NIC enters **exponential backoff**: after mth collision, NIC chooses \( K \) at random from \( \{0,1,2,...,2^m-1\} \). NIC waits \( K \cdot 512 \) bit times, returns to Step 2

Ethernet’s CSMA/CD (more)

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

**Bit time:** .1 microsec for 10 Mbps Ethernet; for \( K=1023 \), wait time is about 50 msec

**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 \( K \cdot 512 \) bit transmission times
- after second collision: choose \( K \) from \( \{0,1,2,3\}... \)
- after ten collisions, choose \( K \) from \( \{0,1,2,3,4,...,1023\} \)
802.3 Ethernet Standards: Link & Physical Layers

- *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

**Manchester encoding**

- used in 10BaseT
- each bit has a transition
- allows clocks in sending and receiving nodes to synchronize to each other
  - no need for a centralized, global clock among nodes!
- Hey, this is physical-layer stuff!
Hubs

... physical-layer ("dumb") repeaters:

- bits coming in one link go out all other links at same rate
- all nodes connected to hub can collide with one another
- no frame buffering
- no CSMA/CD at hub: host NICs detect collisions

Switch

- link-layer device: smarter than hubs, take 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: allows *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' simultaneously, without collisions
  - Not possible with dumb hub

Switch Table

- **Q:** How does switch know that 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:

1. record link associated with sending host
2. index switch table using MAC dest address
3. if entry found for destination
   then {
     if dest on segment from which frame arrived
     then drop the frame
     else forward the frame on interface indicated
   }
else flood

forward on all but the interface on which the frame arrived
Self-learning, forwarding: example

- frame destination unknown: flood
- destination A location known: selective send

<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 S₁ know to forward frame destined to F via S₄ and S₃?
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

Ethernet; hubs & switches

Institutional network

Ethernet; hubs & switches
Switches vs. Routers

- both store-and-forward devices
  - routers: network layer devices (examine network layer headers)
  - switches are link layer devices
- routers maintain routing tables, implement routing algorithms
- switches maintain switch tables, implement filtering, learning algorithms

<table>
<thead>
<tr>
<th>Traffic isolation</th>
<th>Hubs</th>
<th>Routers</th>
<th>Switches</th>
</tr>
</thead>
<tbody>
<tr>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Plug &amp; Play</th>
<th>Hubs</th>
<th>Routers</th>
<th>Switches</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Optimal Routing</th>
<th>Hubs</th>
<th>Routers</th>
<th>Switches</th>
</tr>
</thead>
<tbody>
<tr>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Cut through</th>
<th>Hubs</th>
<th>Routers</th>
<th>Switches</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td></td>
</tr>
</tbody>
</table>