Ethernet

EE 122: Intro to Communication Networks

Fall 2010 (MW 4-5:30 in 101 Barker)

Scott Shenker

TAs: Sameer Agarwal, Sara Alspaugh, Igor Ganichev, Prayag Narula

http://inst.eecs.berkeley.edu/~ee122/

Materials with thanks to Jennifer Rexford, Ion Stoica, Vern Paxson and other colleagues at Princeton and UC Berkeley

Announcements

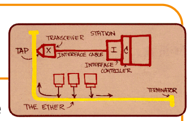
- HW#2 and Project 1A due today
- Midterm next Monday
- Review next lecture
- Extended office hours on Today/Wednesday
  - I'll be available as long as line lasts
- Change in lecture schedule
  - Control protocols moved to after midterm….

Goals of Today's Lecture

- Single-segment Ethernet
  - Review some of the basics
  - Fun and games with backoff functions
- Multi-segment Ethernet
  - Hubs/repeaters vs switches/bridges vs routers
  - Spanning Tree
- Two nontrivial algorithms: (finally!)
  - Backoff algorithms
  - Spanning tree

Ethernet (Single Segment)

Ethernet: CSMA/CD Protocol



- **Carrier sense**: wait for link to be idle
- **Collision detection**: listen while transmitting
  - No collision: transmission is complete
  - Collision: abort transmission & send **jam** signal
- **Random access**: binary exponential back-off
  - After collision, wait a random time before trying again
  - After m^{th} collision, choose K randomly from {0, …, 2^m-1}
  - … and wait for K*512 bit times before trying again
    - o Using min packet size as "slot"
    - o If transmission occurring when ready to send, wait until end of transmission (CSMA)

Binary Exponential Backoff (BEB)

- Think of time as divided in slots
- After each collision, pick a slot randomly within next 2^m slots
  - Where m is the number of collisions since last successful transmission
- Questions:
  - Why backoff?
  - Why random?
  - Why 2^m?
  - Why not listen while waiting?

Behavior of BEB Under Light Load

Look at collisions between two nodes

- First collision: pick one of the next two slots
  - Chance of success after first collision: 50%
  - Average delay 1.5 slots
- Second collision: pick one of the next four slots
  - Chance of success after second collision: 75%
  - Average delay 2.5 slots
- In general: after m^{th} collision
  - Chance of success: $1-2^{-m}$
  - Average delay (in slots): $½ + 2^{(m-1)}$

BEB: Reality vs Theory

- In reality, binary exponential backoff (BEB)
  - Performs well (far from optimal, but no one cares)
    - *Large data packets are ~23 times as large as minimal slot*
  - Is mostly irrelevant
    - *Almost all current ethernets are* ***switched***
- In theory, a very interesting algorithm
  - Stability of algorithm for finite N only proved in 1985
    - Ethernet can handle nonzero traffic load without collapse (duh!)
  - All backoff algorithms unstable for infinite N (1985)
    - Poisson model: infinite user pool, whose total demand is finite
    - Not of practical interest

MAC "Channel Capture" in BEB

- Two hosts, each with infinite packets to send
- With BEB, there is a finite chance that the first one to have a successful transmission will never relinquish the channel
  - The other host will *never* send a packet

Example

- Two hosts, each with infinite packets to send
  - Slot 1: collision
  - Slot 2: each resends with prob ½
    - Assume host A sends, host B does not
  - Slot 3: A and B both send (collision)
  - Slot 4: A sends with probability ½, B with prob. ¼
    - Assume A sends, B does not
  - Slot 5: A definitely sends, B sends with prob. ¼
    - Assume collision
  - Slot 6: A sends with probability ½, B with prob. 1/8
- Conclusion: if A gets through first, the prob. of B sending successfully halves with each collision

Insight

- Σ ProbSendInNextSlot(after k collisions):
  - Sum of probabilities of success for "losing" host
    - Will it resend on first slot? If not, it will lose again
  - If sum is infinite, then losing host will eventually win
  - If sum is finite, then losing host might never win
- Let F(i) = DelayBeforeSend(after i collisions)
  - (Σ F(i))/F(k) is ratio of number of successes for winning host before the k^{th} collision vs average delay for losing host after the k^{th} solution (before trying to send)
    - If diverges, then percentage of wasted time waiting for losing host to start up after winner finishes emptying queue is small

Necessary Mathematical Facts….

- $\Sigma\ 2^{-i}$ is finite
- $\Sigma\ i^{-p}$ is finite for $p > 1$
- $\Sigma\ i^{-p}$ is infinite for $p \leq 1$

More Mathematical Facts….

Sums are from i=1 to i=k…….

- $(\Sigma\ 2^i)/2^k$ remains finite k grows

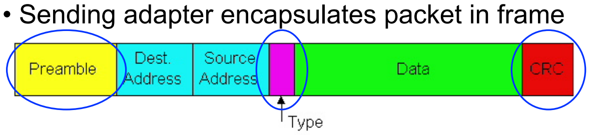
- $(\Sigma\ i^p)/k^p$ diverges as k grows

Different Backoff Functions

- Exponential: backoff ~ a^i
  - Channel capture *(loser might not send until winner idle)*
  - Efficiency less than 1 *(time wasted waiting for loser to start)*
- Superlinear polynomial: backoff ~ i^p p>1
  - Channel capture
  - Efficiency is 1 (for any finite N)
- Sublinear polynomial: backoff ~ i^p p≤1
  - No channel capture *(loser not shut out)*
  - Efficiency is less than 1 (and goes to zero for large N)
    - *Time wasted resolving collisions*

Ethernet Frame Structure

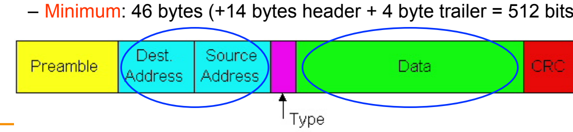
- Sending adapter encapsulates packet in frame



- Preamble: synchronization
  - Seven bytes with pattern **10101010**, followed by one byte with pattern **10101011**
  - Used to synchronize receiver & sender
- Type: indicates the higher layer protocol
  - Usually IP (but also Novell IPX, AppleTalk, …)
- CRC: cyclic redundancy check
  - Receiver checks & simply drops frames with errors

Ethernet Frame Structure (Continued)

- Addresses: 48-bit source and destination MAC addresses
  - Receiver's adaptor passes frame to network-level protocol
    - If destination address matches the adaptor's
    - Or the destination address is the broadcast address (**ff:ff:ff:ff:ff:ff**)
    - Or the destination address is a multicast group receiver belongs to
    - Or the adaptor is in promiscuous mode
  - Addresses are **globally unique**
    - Assigned by NIC vendors (top three octets specify vendor)
      - During any given week, > 500 vendor codes seen at LBNL
- Data:
  - Maximum: 1,500 bytes
  - Minimum: 46 bytes (+14 bytes header + 4 byte trailer = 512 bits)



Ethernet, con't

- Connectionless
  - No handshaking between sending and receiving adapter
- Unreliable
  - Receiving adapter doesn't send ACKs or NACKs
  - Packets passed to network layer can have gaps
  - Gaps will be filled if application is using TCP
  - Otherwise, application will see the gaps
- 2,700 page IEEE 802.3 standardization
  - http://standards.ieee.org/getieee802/802.3.html
- Note, "classical" Ethernet has no length field …
  - … instead, sender pauses 9.2 μsec when done
  - 802.3 shoehorns in a length field

Benefits of Ethernet

- Easy to administer and maintain
- Inexpensive
- Increasingly higher speed
- Evolvable!

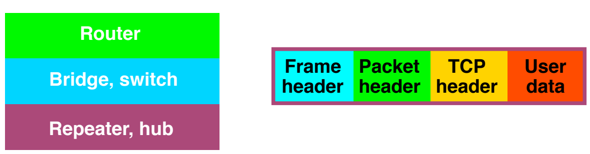
Evolution of Ethernet

- Changed everything except the frame format
  - From single coaxial cable to hub-based star
  - From shared media to switches
  - From electrical signaling to optical
- Lesson #1
  - The right interface can accommodate many changes
  - Implementation is hidden behind interface
- Lesson #2
  - Really hard to displace the dominant technology
  - Slight performance improvements are not enough

Ethernet (Multiple Segments)

Shuttling Data at Different Layers

- Different devices switch different things
  - Physical layer: electrical signals (repeaters and hubs)
  - Link layer: frames (bridges and switches)
  - Network layer: packets (routers)

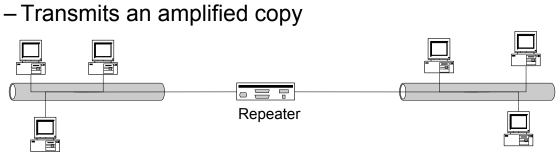


Key Distinction

- Routers: forward based on IP headers
- Switches/Bridges: forward based on MAC addresses
- Repeaters/Hubs: broadcast all bits

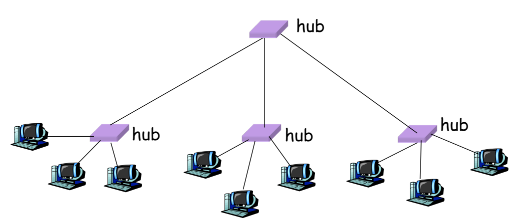
Physical Layer: Repeaters

- Distance limitation in local-area networks
  - Electrical signal becomes weaker as it travels
  - Imposes a limit on the length of a LAN
    - o In addition to limit imposed by collision detection
- Repeaters join LANs together
  - Analog electronic device
  - Continuously monitors electrical signals on each LAN
  - Transmits an amplified copy



Physical Layer: Hubs

- Joins multiple input lines electrically
  - Do not necessarily amplify the signal
- Very similar to repeaters
  - Also operates at the physical layer

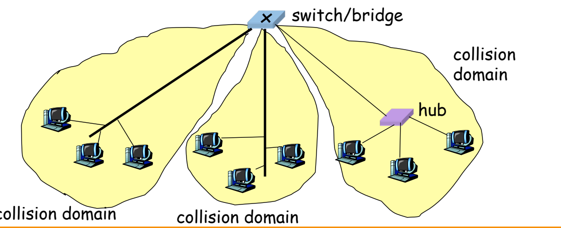


Limitations of Repeaters and Hubs

- One large collision domain
  - Every bit is sent everywhere
  - So, aggregate throughput is limited
  - E.g., three departments each get 10 Mbps independently
  - … and then if connect via a hub must share 10 Mbps
- Cannot support multiple LAN technologies
  - Repeaters/hubs do not buffer or interpret frames
  - So, can't interconnect between different rates or formats
  - E.g., no mixing 10 Mbps Ethernet & 100 Mbps Ethernet
- Limitations on maximum nodes and distances
  - Does not circumvent limitations of shared media
  - E.g., still cannot go beyond 2500 meters on Ethernet

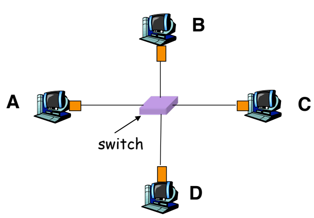
Link Layer: Switches / Bridges

- Connect two or more LANs at the link layer
  - Extracts destination address from the frame
  - Looks up the destination in a table
  - Forwards the frame to the appropriate LAN segment
    - o Or point-to-point link, for higher-speed Ethernet
- Each segment is its own collision domain (if not just a link)



Switches & Concurrent Comunication

- Host A can talk to C, while B talks to D



- If host has (dedicated) point-to-point link to switch:
  - Full duplex: each connection can send in both directions
  - Completely avoids collisions
    - o No need for carrier sense, collision detection, and so on
    - o Complete change in nature of multiple access, but same framing

Advantages Over Hubs & Repeaters

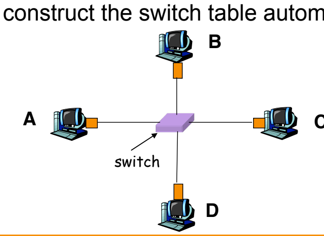
- Only forwards frames as needed
  - Filters frames to avoid unnecessary load on segments
  - Sends frames only to segments that need to see them
- Extends the geographic span of the network
  - Separate collision domains allow longer distances
- Improves privacy by limiting scope of frames
  - Hosts can "snoop" the traffic traversing their segment
  - … but not all the rest of the traffic
- Applies CSMA/CD in segment (not whole net)
  - Smaller collision domain
- Joins segments using different technologies

Disadvantages Over Hubs & Repeaters

- Higher cost
  - More complicated devices that cost more money
- Delay in forwarding frames
  - Bridge/switch must receive and parse the frame
  - … and perform a look-up to decide where to forward
  - Introduces store-and-forward delay
    - o Can ameliorate using *cut-through switching*
      - Start forwarding after only header received
- Need to learn where to forward frames
  - Bridge/switch needs to construct a *forwarding table*
  - Ideally, without intervention from network administrators
  - Solution: self-learning

Motivation For Self Learning

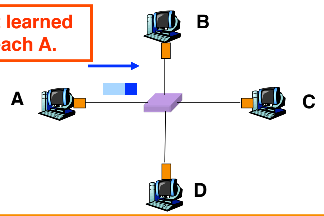
- Large benefit if switch/bridge forward frames only on segments that need them
  - Allows concurrent use of other links
- Switch table
  - Maps destination MAC address to outgoing interface
  - Goal: construct the switch table automatically



Self Learning: Building the Table

- When a frame arrives
  - Inspect *source* MAC address
  - Associate address with the *incoming* interface
  - Store mapping in the switch table
  - Use time-to-live field to eventually forget the mapping
    - ***Soft state***

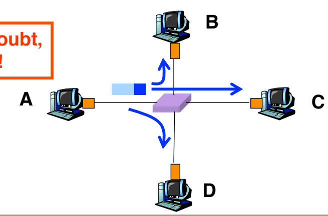
Switch just learned how to reach A.



Self Learning: Handling Misses

- When frame arrives with unfamiliar destination
  - Forward the frame out **all** of the interfaces ("flooding")
    - … except for the one where the frame arrived
  - Hopefully, this case won't happen very often
  - When destination replies, switch learns that node, too

When in doubt, shout!



Switch Filtering / Forwarding

When switch receives a frame:

index the switch table using MAC dest address

if entry found for destination **{**

  if dest on segment from which frame arrived
    then drop frame

    else forward frame on interface indicated

}

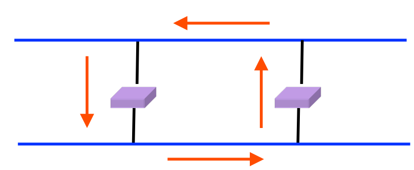
  else flood

Problems?

forward on all but the interface on which the frame arrived

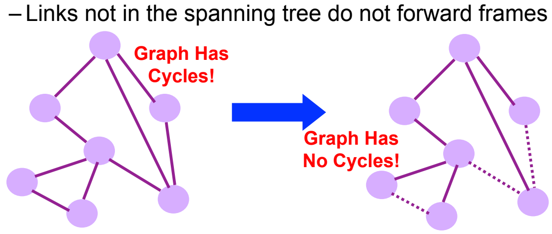
Flooding Can Lead to Loops

- Switches sometimes need to broadcast frames
  - Upon receiving a frame with an unfamiliar destination
  - Upon receiving a frame sent to the broadcast address
  - Implemented by flooding
- Flooding can lead to forwarding loops
  - E.g., if the network contains a cycle of switches
    - Either accidentally, or by design for higher reliability
  - "Broadcast storm"



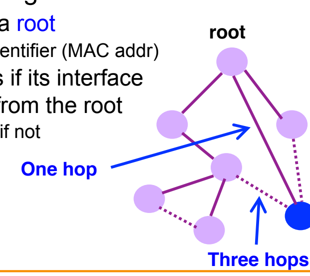
Solution: Spanning Trees

- Ensure the forwarding topology has no loops
  - Avoid using some of the links when flooding
  - … to prevent loop from forming
- Spanning tree (K&R pp. 411-413)
  - Sub-graph that covers all vertices but *contains no cycles*
  - Links not in the spanning tree do not forward frames



Constructing a Spanning Tree

- Need a distributed algorithm
  - Switches cooperate to build the spanning tree
  - … and adapt automatically when failures occur
- Key ingredients of the algorithm
  - Switches need to elect a root
    - The switch w/ smallest identifier (MAC addr)
  - Each switch determines if its interface is on the shortest path from the root
    - Excludes it from the tree if not
  - Messages (Y, d, X)
    - From node X
    - Proposing Y as the root
    - And the distance is d

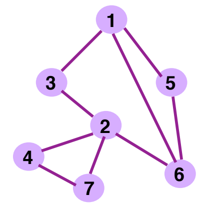


Steps in Spanning Tree Algorithm

- Initially, each switch proposes itself as the root
  - Switch sends a message out every interface
  - … proposing itself as the root with distance 0
  - Example: switch X announces (X, 0, X)
- Switches update their view of the root
  - Upon receiving message (Y, d, Z) from Z, check Y's id
  - If new id smaller, start viewing that switch as root
- Switches compute their distance from the root
  - Add 1 to the distance received from a neighbor
  - Identify interfaces not on shortest path to the root
  - … and exclude them from the spanning tree
- If root or shortest distance to it changed, flood updated message (Y, d+1, X)

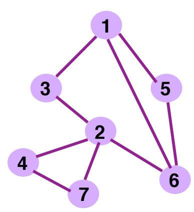
Example From Switch #4's Viewpoint

- Switch #4 thinks it is the root
  - Sends (4, 0, 4) message to 2 and 7
- Then, switch #4 hears from #2
  - Receives (2, 0, 2) message from 2
  - … and thinks that #2 is the root
  - And realizes it is just one hop away
- Then, switch #4 hears from #7
  - Receives (2, 1, 7) from 7
  - And realizes this is a longer path
  - So, prefers its own one-hop path
  - And removes 4-7 link from the tree



Example From Switch #4's Viewpoint

- Switch #2 hears about switch #1
  - Switch 2 hears (1, 1, 3) from 3
  - Switch 2 starts treating 1 as root
  - And sends (1, 2, 2) to neighbors
- Switch #4 hears from switch #2
  - Switch 4 starts treating 1 as root
  - And sends (1, 3, 4) to neighbors
- Switch #4 hears from switch #7
  - Switch 4 receives (1, 3, 7) from 7
  - And realizes this is a longer path
  - So, prefers its own three-hop path
  - And removes 4-7 link from the tree



Robust Spanning Tree Algorithm

- Algorithm must react to failures
  - Failure of the root node
    - Need to elect a new root, with the next lowest identifier
  - Failure of other switches and links
    - Need to recompute the spanning tree
- Root switch continues sending messages
  - Periodically reannouncing itself as the root (1, 0, 1)
  - Other switches continue forwarding messages
- Detecting failures through timeout (soft state)
  - If no word from root, times out and claims to be the root
  - Delay in reestablishing spanning tree is ***major problem*** in modern datacenters
  - Work on rapid spanning tree algorithms…

Moving From Switches to Routers

- Advantages of switches over routers
  - Plug-and-play
  - Fast filtering and forwarding of frames
- Disadvantages of switches over routers
  - Topology restricted to a spanning tree
  - Large networks require large ARP tables
  - Broadcast storms can cause the network to collapse
  - Can't accommodate non-Ethernet segments (why not?)

Comparing Hubs, Switches & Routers

	hubs	switches	routers
traffic isolation	no	yes	yes
plug & play	yes	yes	no
optimized routing	no	no	yes

Summary

- Ethernet as an exemplar of link-layer technology
- Simplest form, single segment:
  - *Carrier sense*, *collision detection*, and random access
- Extended to span multiple segments:
  - Hubs & repeaters: physical-layer interconnects
  - Bridges / switches: link-layer interconnects
- Key ideas in switches
  - Self learning of the switch table
  - Spanning trees
- Next time: midterm review