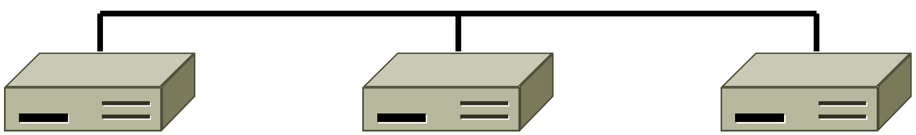
Computer Networks

Randomized Multiple Access

Topic

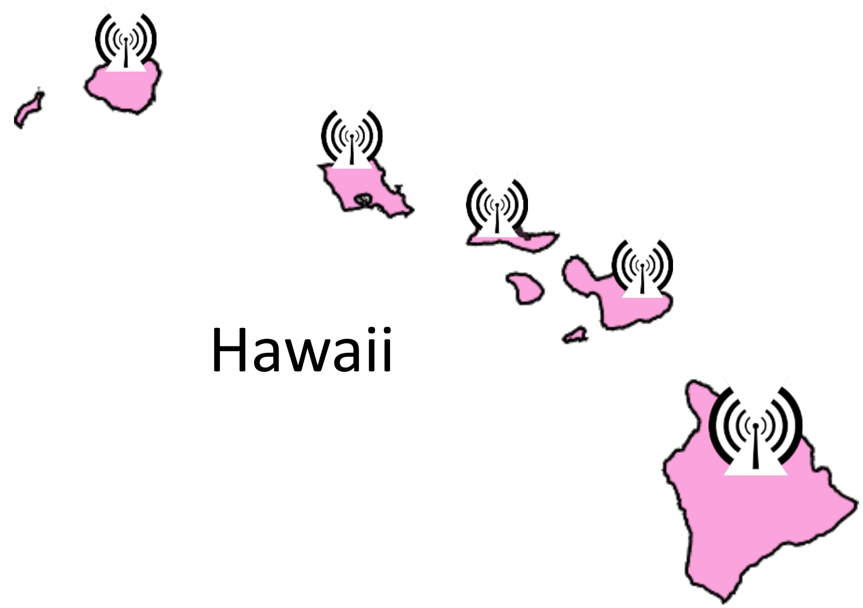
- How do nodes share a single link? Who sends when, e.g., in WiFI?
  - Explore with a simple model



- Assume no-one is in charge; this is a distributed system

ALOHA Network

- Seminal computer network connecting the Hawaiian islands in the late 1960s
  - When should nodes send?
  - A new protocol was devised by Norm Abramson …



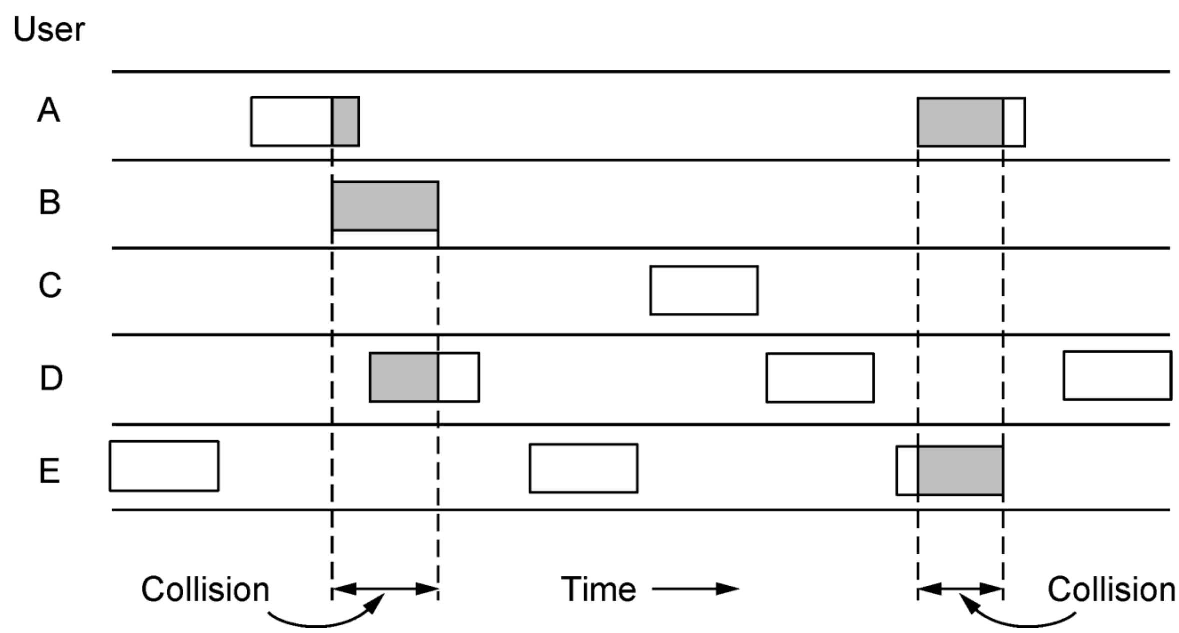
ALOHA Protocol

- Simple idea:
  - Node just sends when it has traffic.
  - If there was a collision (no ACK received) then wait a random time and resend
- That's it!

ALOHA Protocol (2)

- Some frames will be lost, but many may get through…

- Good idea?

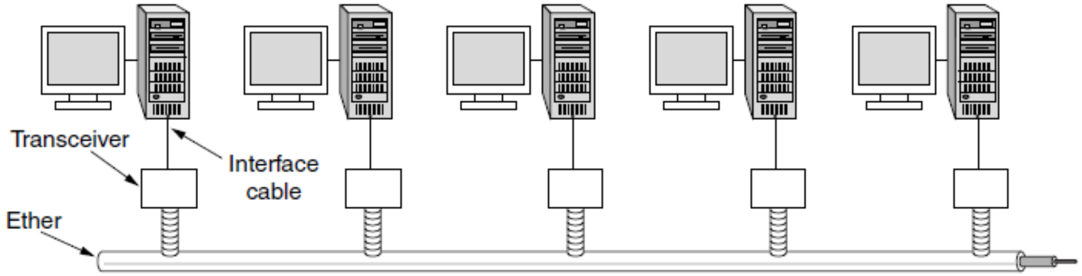


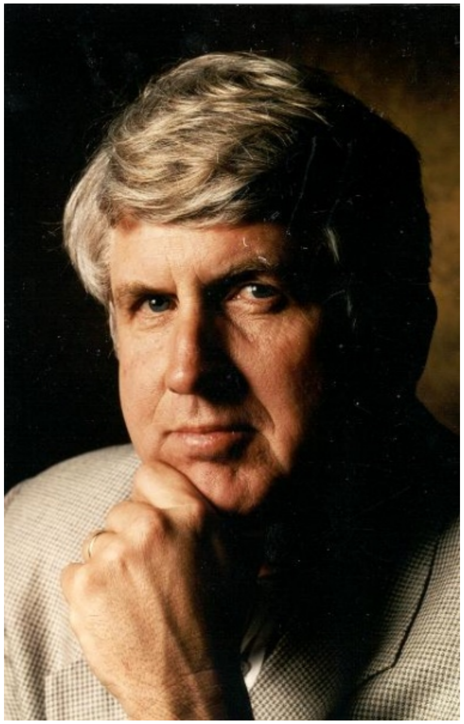
ALOHA Protocol (3)

- Simple, decentralized protocol that works well under low load!
- Not efficient under high load
  - Analysis shows at most 18% efficiency
  - Improvement: divide time into slots and efficiency goes up to 36%
- We'll look at other improvements

Classic Ethernet

- ALOHA inspired Bob Metcalfe to invent Ethernet for LANs in 1973
  - Nodes share 10 Mbps coaxial cable
  - Hugely popular in 1980s, 1990s





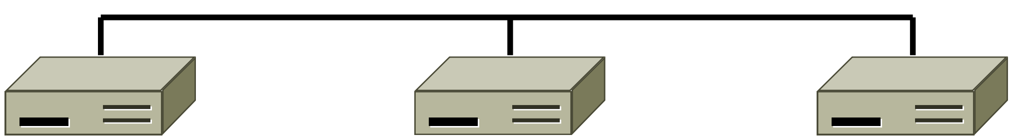
: © 2009 IEEE

CSMA (Carrier Sense Multiple Access)

- Improve ALOHA by listening for activity before we send (Doh!)
  - Can do easily with wires, not wireless
- So does this eliminate collisions?
  - Why or why not?

CSMA (2)

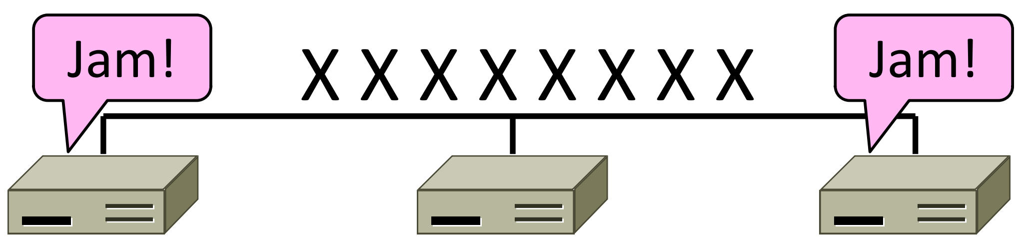
- Still possible to listen and hear nothing when another node is sending because of delay



- CSMA is a good defense against collisions only when BD is small

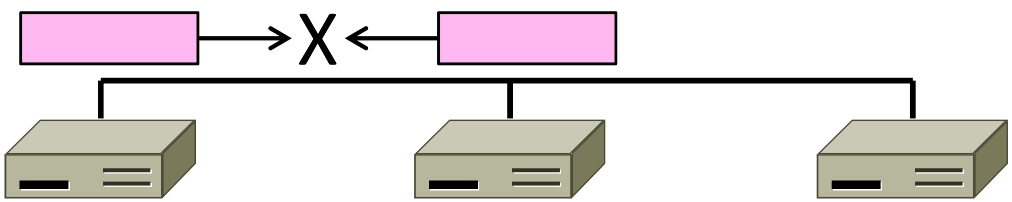
CSMA/CD (with Collision Detection)

- Can reduce the cost of collisions by detecting them and aborting (Jam) the rest of the frame time
  - Again, we can do this with wires



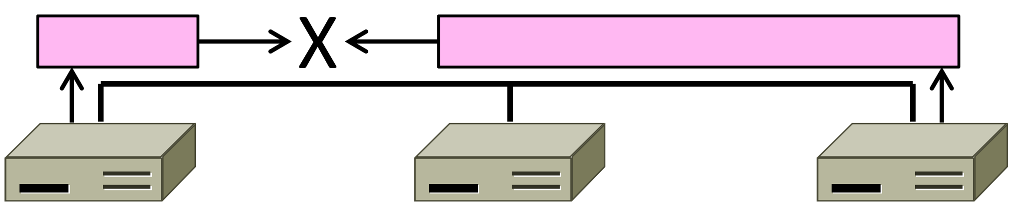
CSMA/CD Complications

- Want everyone who collides to know that it happened
  - Time window in which a node may hear of a collision is 2D seconds



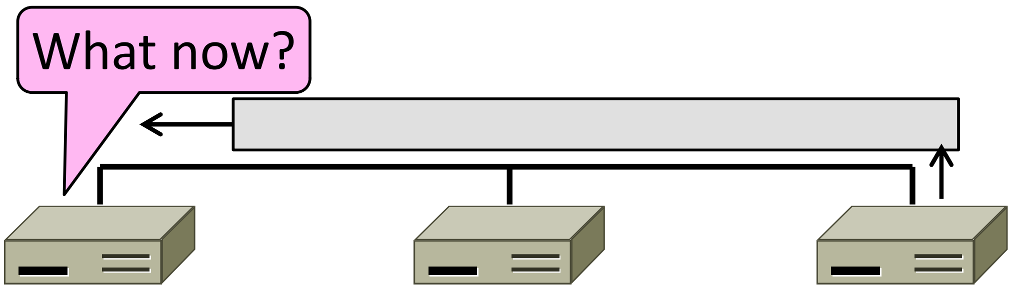
CSMA/CD Complications (2)

- Impose a minimum frame size that lasts for 2D seconds
  - So node can't finish before collision
  - Ethernet minimum frame is 64 bytes



CSMA "Persistence"

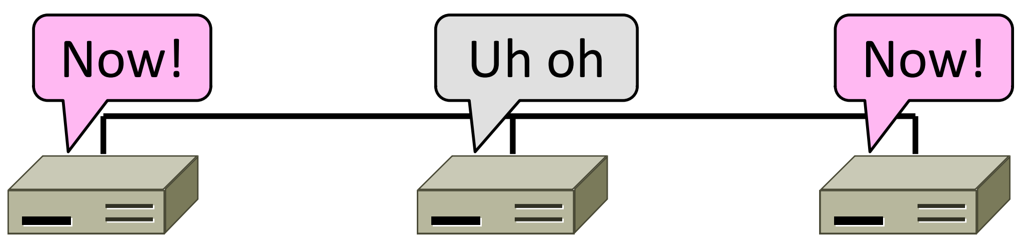
- What should a node do if another node is sending?



- Idea: Wait until it is done, and send

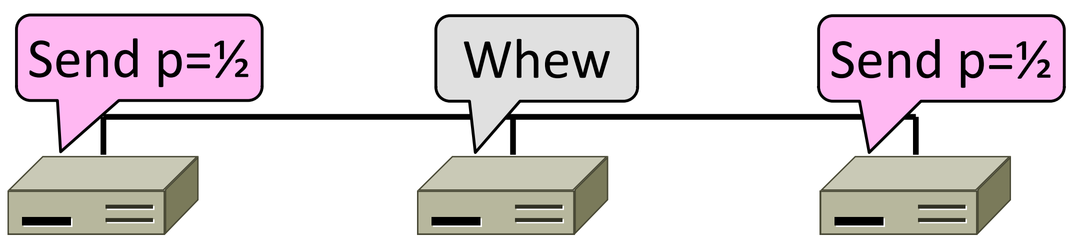
CSMA "Persistence" (2)

- Problem is that multiple waiting nodes will queue up then collide
  - More load, more of a problem



CSMA “Persistence” (3)

- Intuition for a better solution
  - If there are N queued senders, we want each to send next with probability 1/N



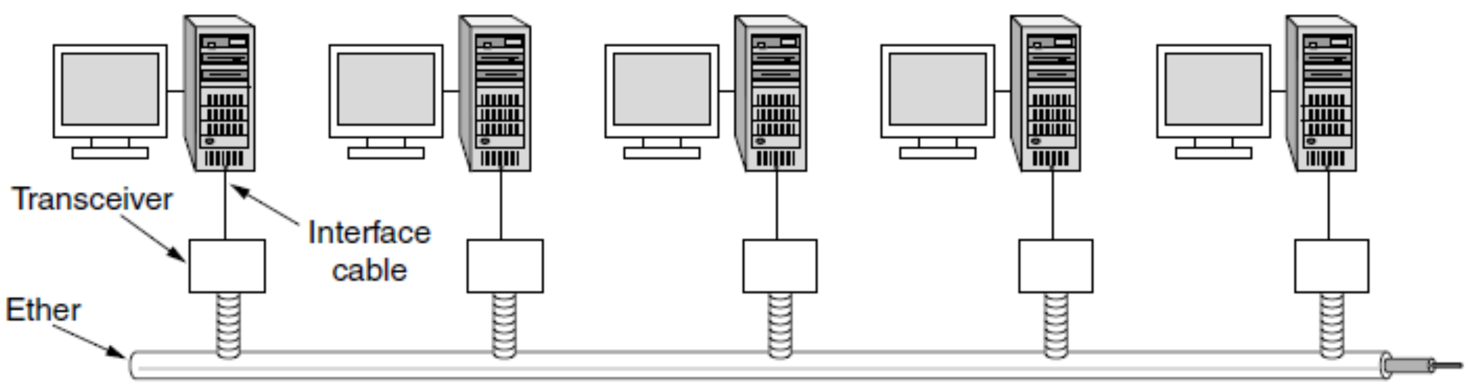
Binary Exponential Backoff (BEB)

- Cleverly estimates the probability
  - 1st collision, wait 0 or 1 frame times
  - 2nd collision, wait from 0 to 3 times
  - 3rd collision, wait from 0 to 7 times …

- BEB doubles interval for each successive collision
  - Quickly gets large enough to work
  - Very efficient in practice

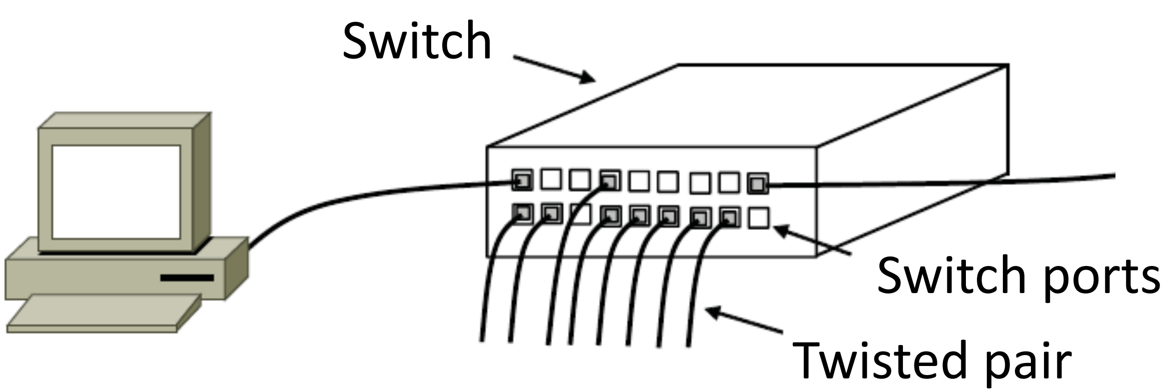
Classic Ethernet, or IEEE 802.3

- Most popular LAN of the 1980s, 1990s
  - 10 Mbps over shared coaxial cable, with baseband signals
  - Multiple access with “1-persistent CSMA/CD with BEB”



Modern Ethernet

- Based on switches, not multiple access, but still called Ethernet



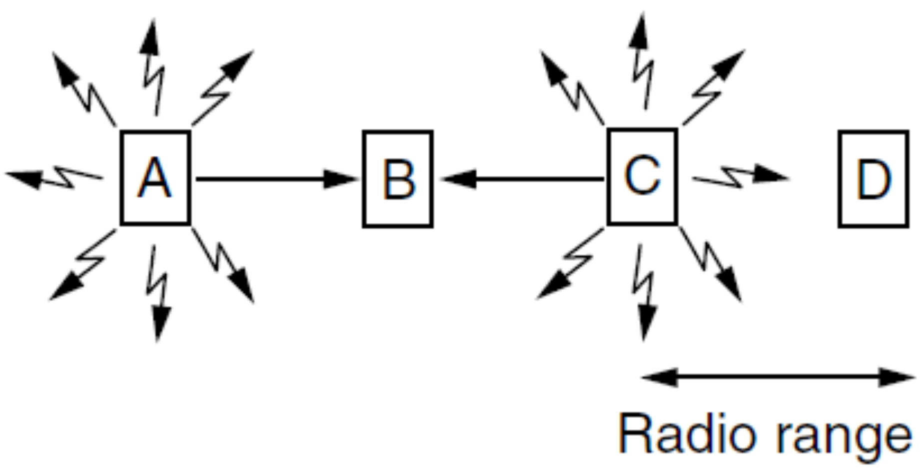
Wireless Complications

- Wireless is more complicated than the wired case (No Surprise!)
  1. Nodes may have different areas of coverage – doesn't fit Carrier Sense »
  2. Nodes can't hear while sending – can't Collision Detect »

 ≠ CSMA/CD

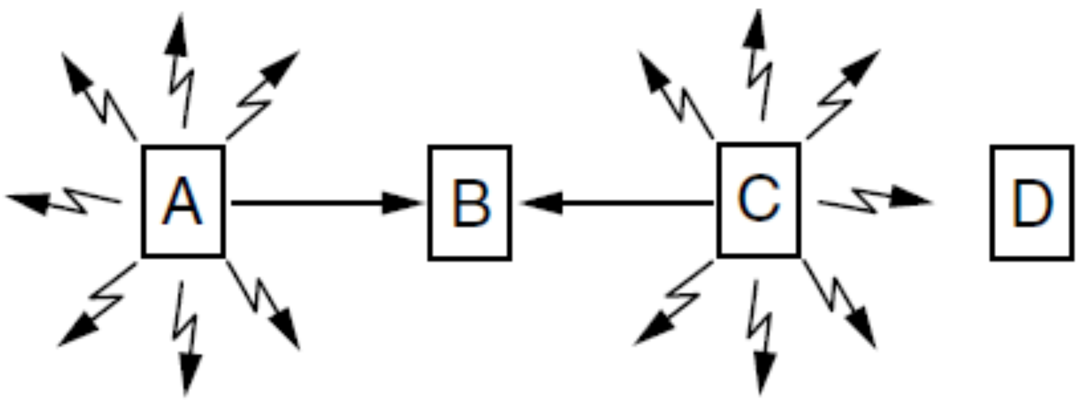
Different Coverage Areas

- Wireless signal is broadcast and received nearby, where there is sufficient SNR



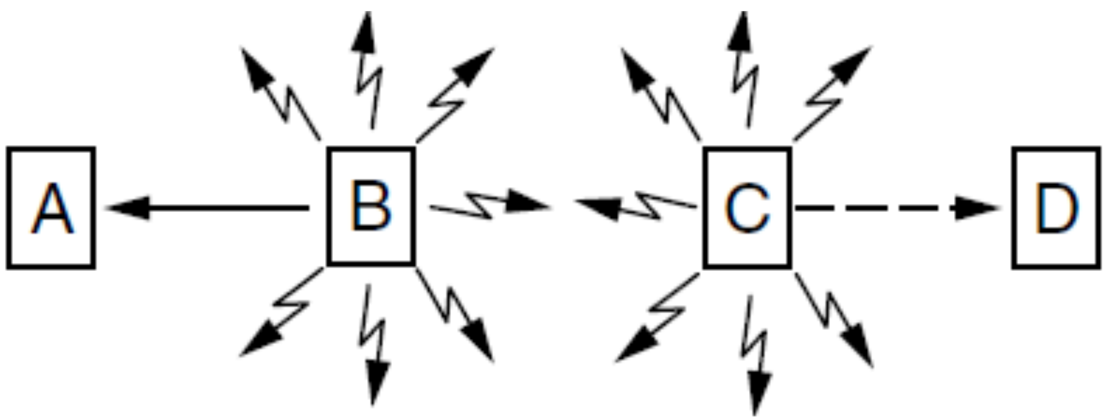
Hidden Terminals

- Nodes A and C are <u>hidden terminals</u> when sending to B
  - Can't hear each other (to coordinate) yet collide at B
  - We want to avoid the inefficiency of collisions



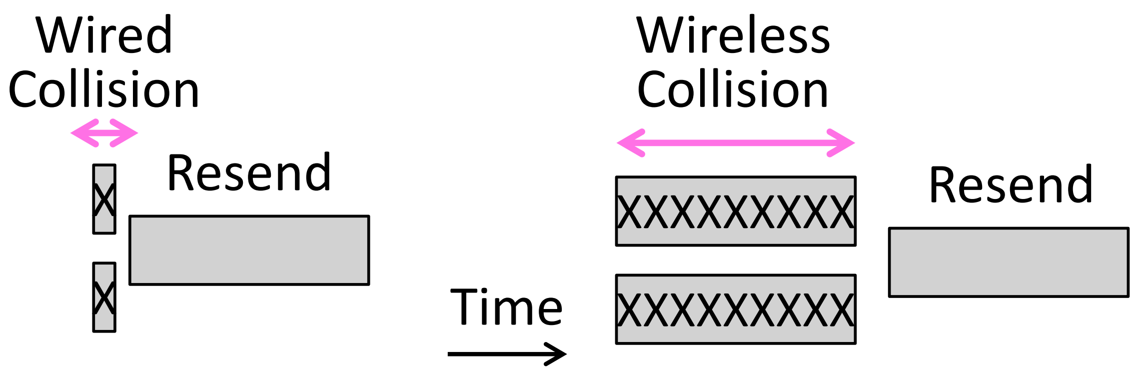
Exposed Terminals

- B and C are exposed terminals when sending to A and D
  - Can hear each other yet don't collide at receivers A and D
  - We want to send concurrently to increase performance



Nodes Can't Hear While Sending

- With wires, detecting collisions (and aborting) lowers their cost
- More wasted time with wireless



Possible Solution: MACA

- MACA uses a short handshake instead of CSMA (Karn, 1990)
  – 802.11 uses a refinement of MACA (later)

- Protocol rules:
  1. A sender node transmits a RTS (Request-To-Send, with frame length)
  2. The receiver replies with a CTS (Clear-To-Send, with frame length)
  3. Sender transmits the frame while nodes hearing the CTS stay silent

  – Collisions on the RTS/CTS are still possible, but less likely

MACA – Hidden Terminals

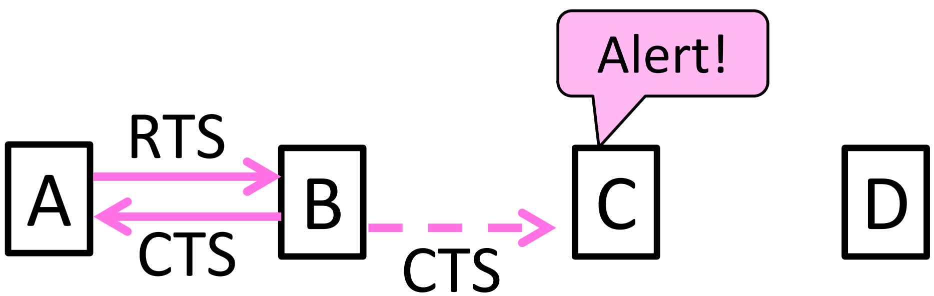
- A→B with hidden terminal C
  1. A sends RTS, to B

MACA – Hidden Terminals (2)

- A→B with hidden terminal C
    2. B sends CTS, to A, and C too

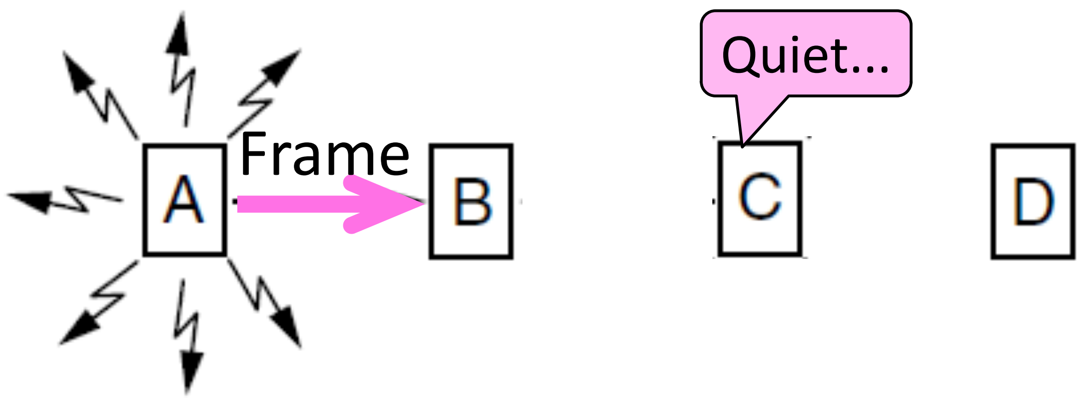
MACA – Hidden Terminals (3)

- A→B with hidden terminal C
  2. B sends CTS, to A, and C too



MACA – Hidden Terminals (4)

- A→B with hidden terminal C
  3. A sends frame while C defers

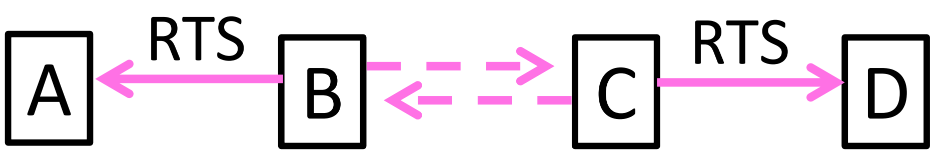


MACA – Exposed Terminals

- B→A, C→D as exposed terminals
  - B and C send RTS to A and D

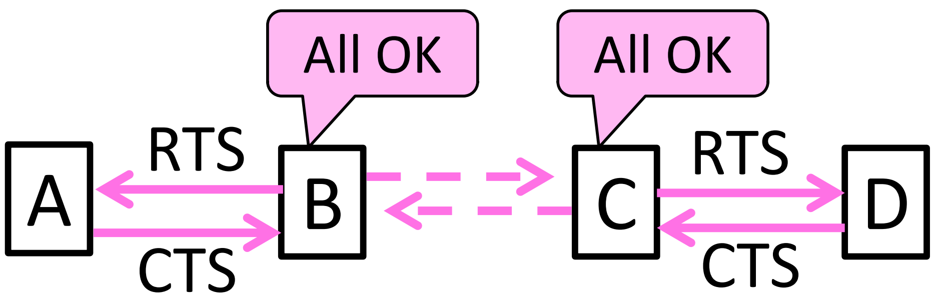
MACA – Exposed Terminals (2)

- B→A, C→D as exposed terminals
  - A and D send CTS to B and C



MACA – Exposed Terminals (3)

- B→A, C→D as exposed terminals
  - A and D send CTS to B and C

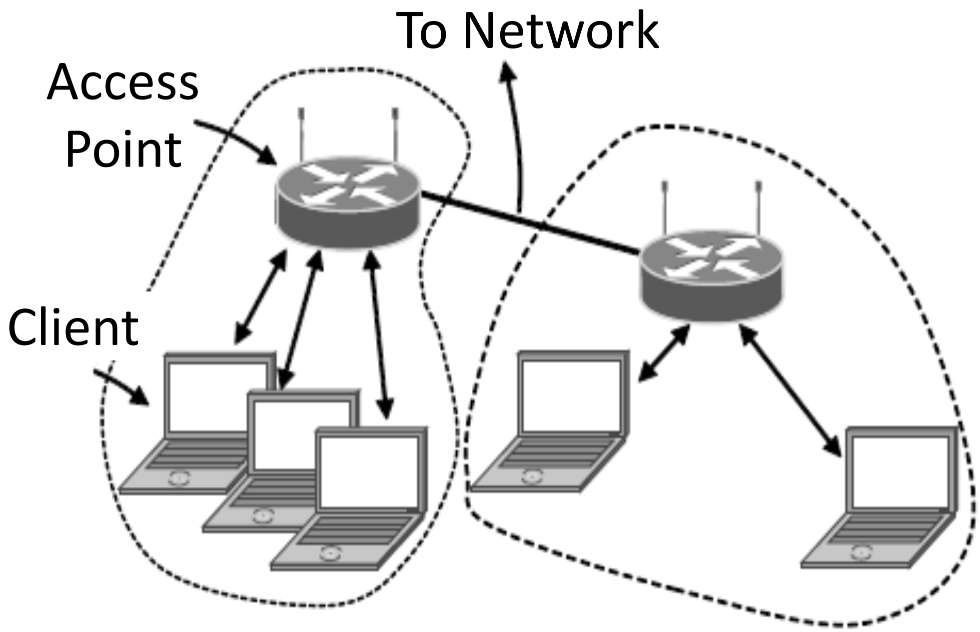


MACA – Exposed Terminals (4)

- B→A, C→D as exposed terminals
  - A and D send CTS to B and C

802.11, or WiFi

- Very popular wireless LAN started in the 1990s
- Clients get connectivity from a (wired) AP (Access Point)
- It's a multi-access problem ☺
- Various flavors have been developed over time
  - Faster, more features



802.11 Physical Layer

- Uses 20/40 MHz channels on ISM bands
  - 802.11b/g/n on 2.4 GHz
  - 802.11 a/n on 5 GHz
- OFDM modulation (except legacy 802.11b)
  - Different amplitudes/phases for varying SNRs
  - Rates from 6 to 54 Mbps plus error correction
  - 802.11n uses multiple antennas; see “802.11 with Multiple Antennas for Dummies”

802.11 Link Layer

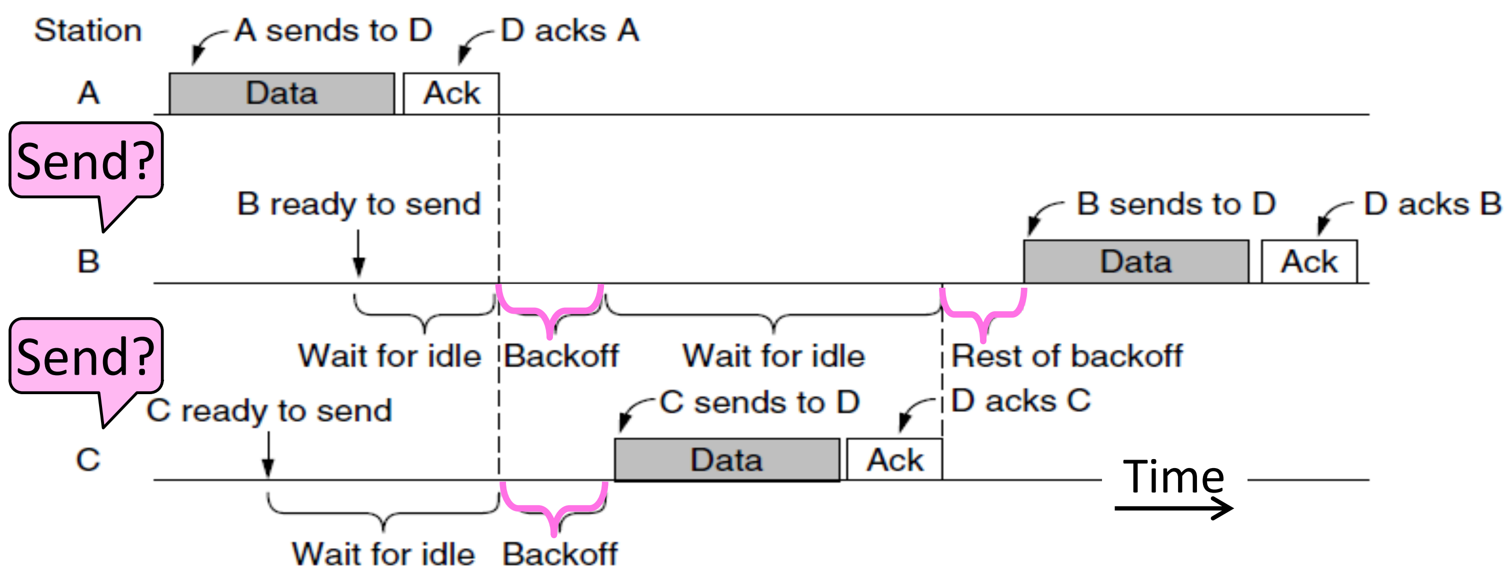
- Multiple access uses CSMA/CA; RTS/CTS optional
- Frames are ACKed and retransmitted with ARQ
- Funky addressing (three addresses!) due to AP
- Errors are detected with a 32-bit CRC
- Many, many features (e.g., encryption, power save)

Packet from Network layer (IP)

	Frame control	Duration	Address 1 (recipient)	Address 2 (transmitter)	Address 3	Sequence	Data	Check sequence
Bytes	2	2	6	6	6	2	0–2312	4

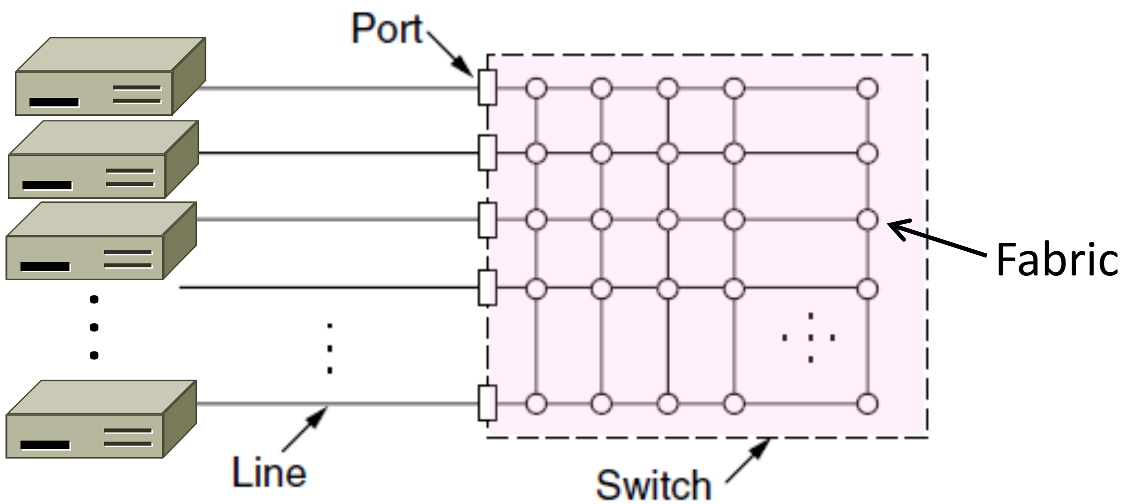
802.11 CSMA/CA for Multiple Access

- Sender avoids collisions by inserting small random gaps
  - E.g., when both B and C send, C picks a smaller gap, goes first



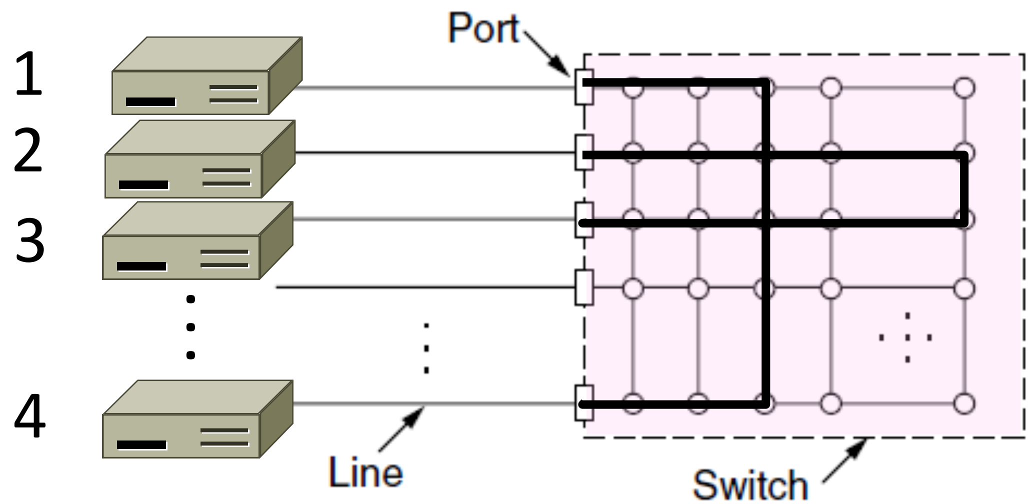
Inside a Switch

- Uses frame addresses to connect input port to the right output port; multiple frames may be switched in parallel



Inside a Switch (2)

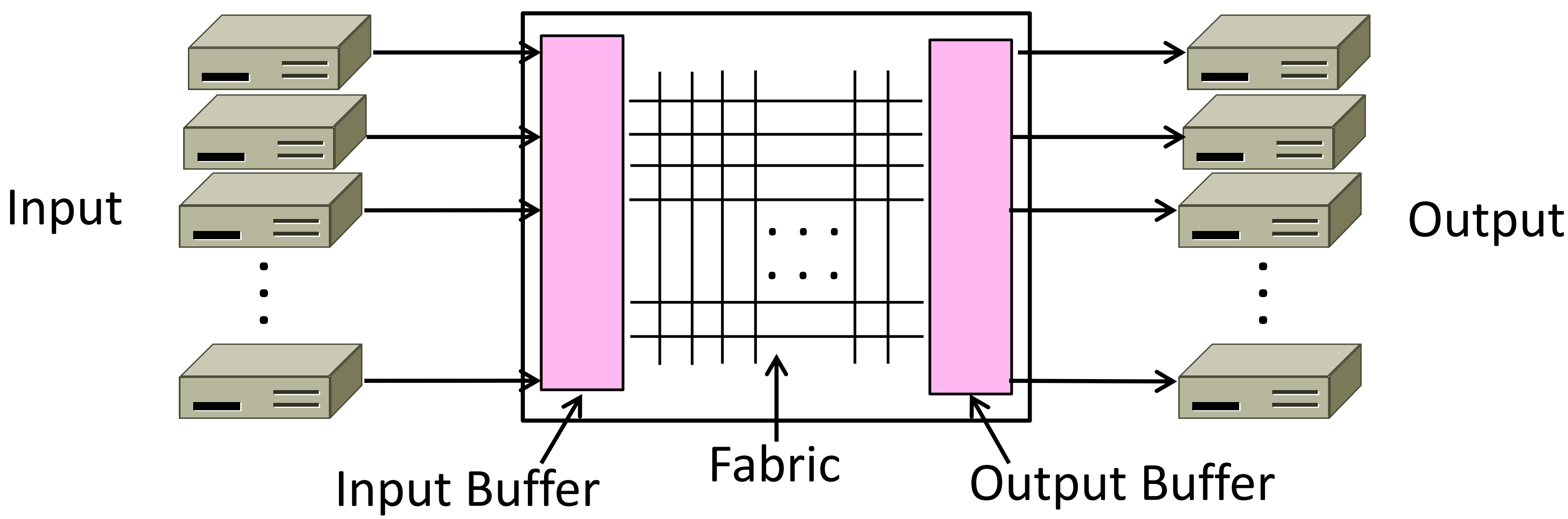
- Port may be used for both input and output (full-duplex)
  - Just send, no multiple access protocol



1 → 4
and
2 → 3

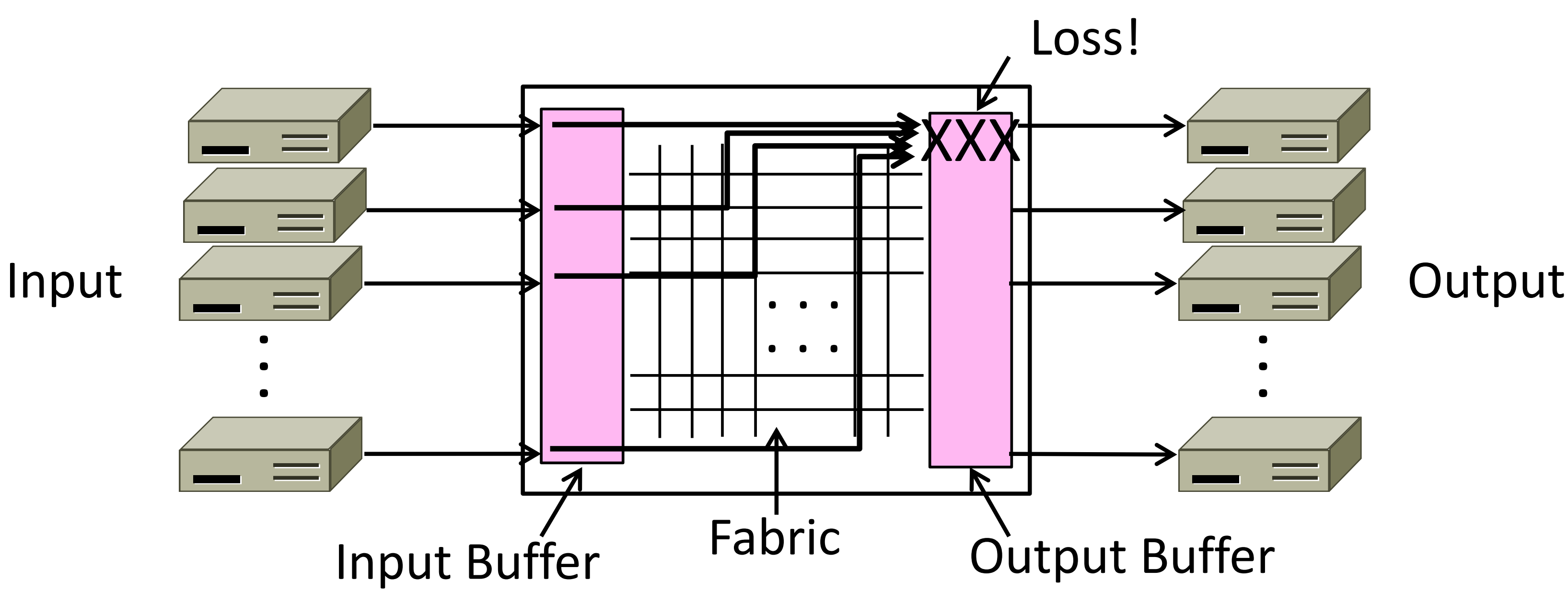
Inside a Switch (3)

- Need buffers for multiple inputs to send to one output



Inside a Switch (4)

- Sustained overload will fill buffer and lead to frame loss



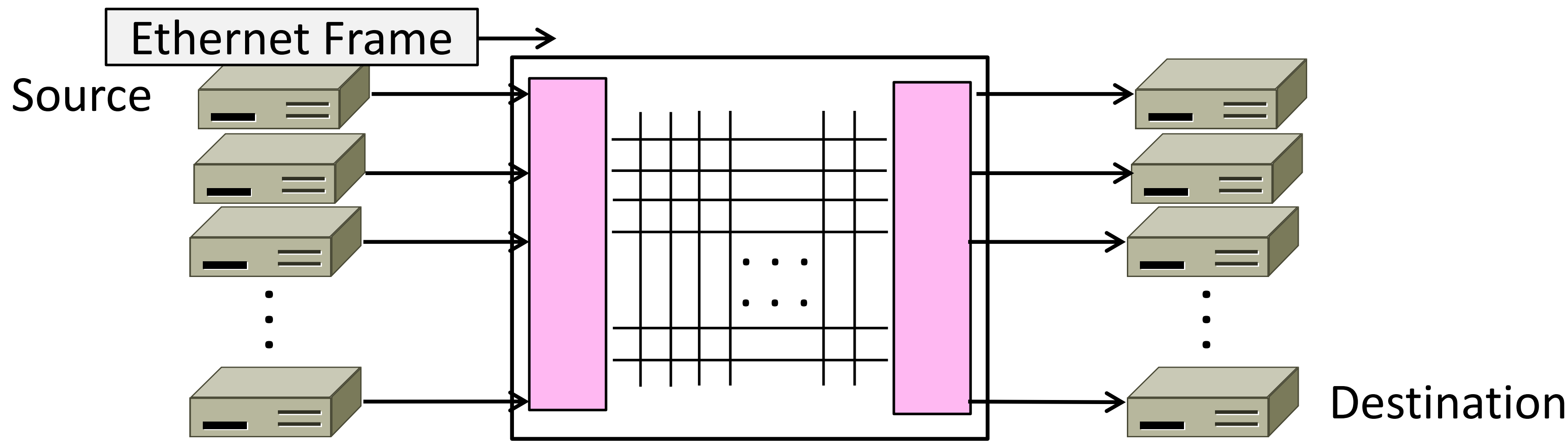
Advantages of Switches

- Switches and hubs have replaced the shared cable of classic Ethernet
  - Convenient to run wires to one location
  - More reliable; wire cut is not a single point of failure that is hard to find

- Switches offer scalable performance
  - E.g., 100 Mbps per port instead of 100 Mbps for all nodes of shared cable / hub

Switch Forwarding

- Switch needs to find the right output port for the destination address in the Ethernet frame. How?
  – Want to let hosts be moved around readily; don't look at IP



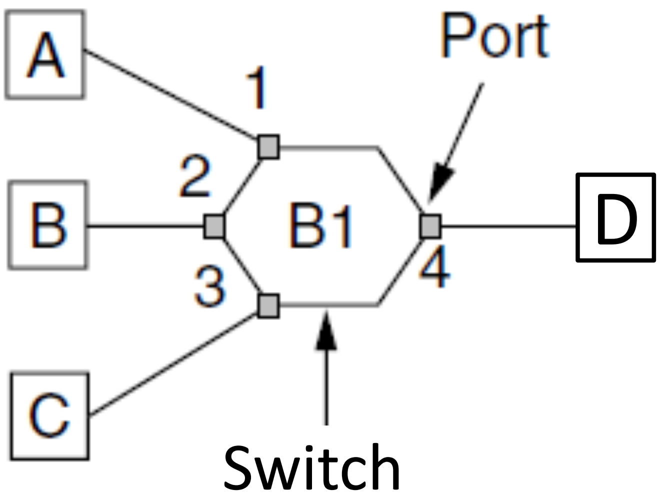
- How can we enable plug-n-play for ethernet switches?

Backward Learning

- Switch forwards frames with a port/address table as follows:
  1. To fill the table, it looks at the source address of input frames
  2. To forward, it sends to the port, or else broadcasts to all ports

Backward Learning (2)

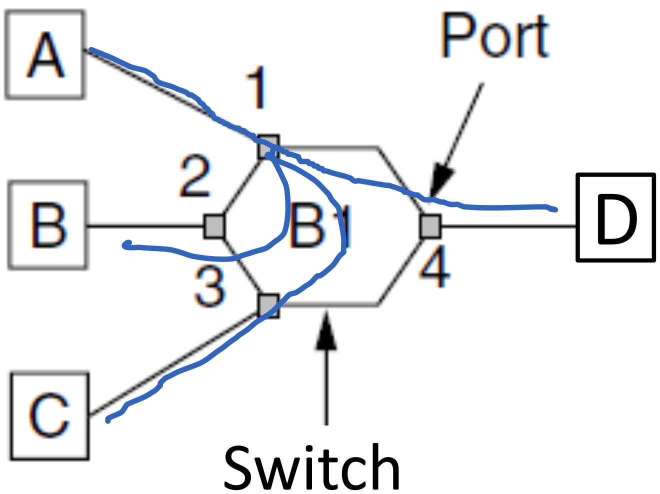
- 1: A sends to D



Address	Port
A	
B	
C	
D	

Backward Learning (3)

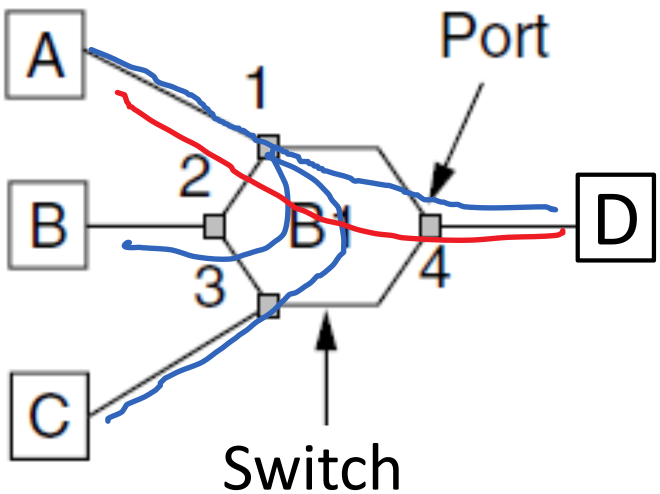
- 2: D sends to A



Address	Port
A	1
B	
C	
D	

Backward Learning (4)

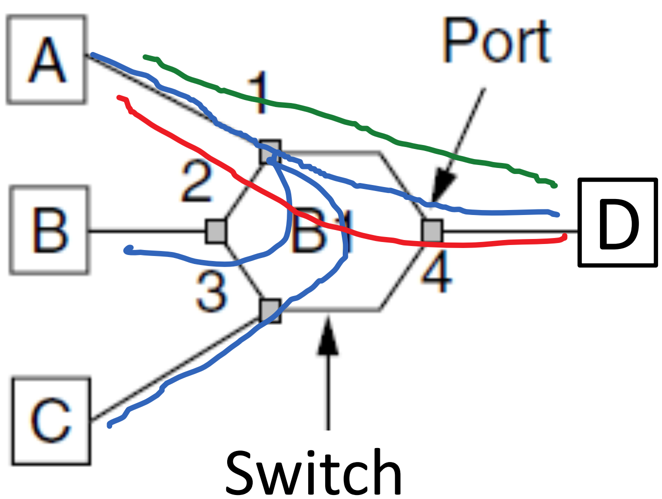
- 3: D sends to A



Address	Port
A	1
B	
C	
D	4

Backward Learning (5)

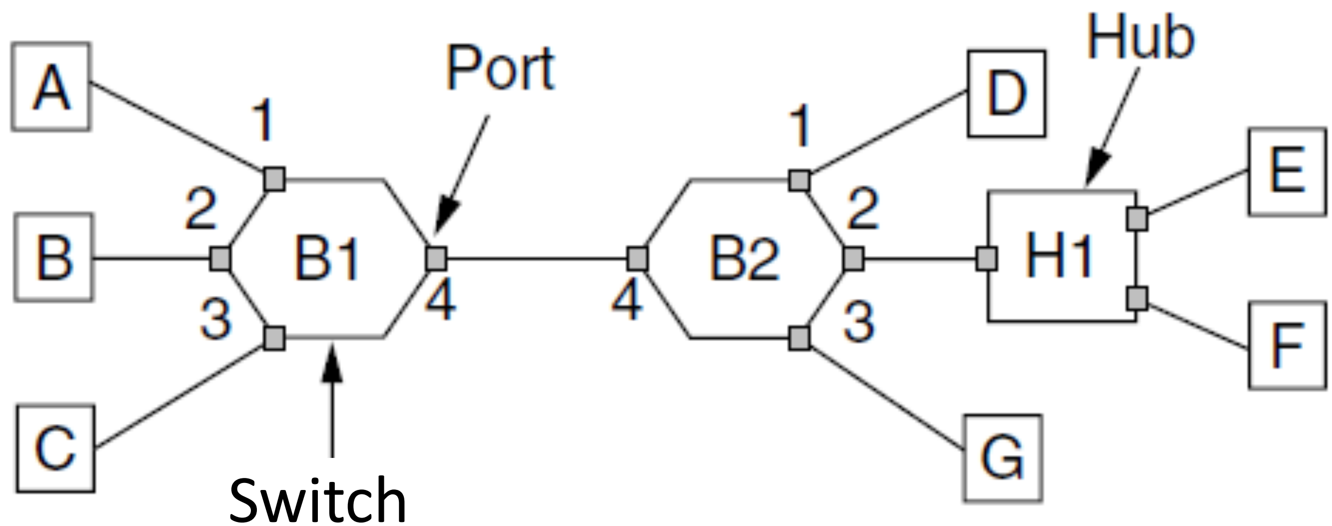
- 3: D sends to A



Address	Port
A	1
B	
C	
D	4

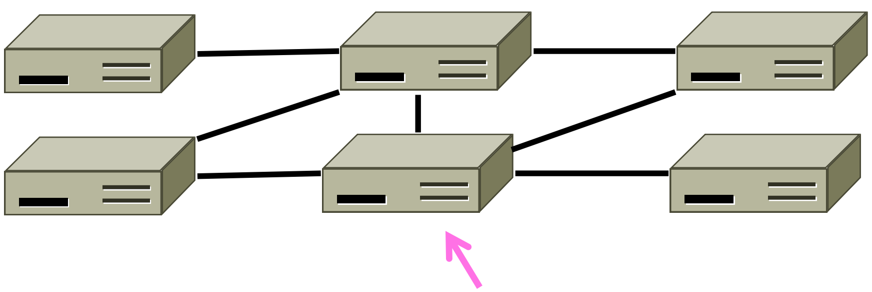
Learning with Multiple Switches

- Just works with multiple switches and a mix of hubs, *assuming no loops* in the topology, E.g., A sends to D



Topic

- How can we connect switches in any topology so they just work
  - This is part 2 of switched Ethernet

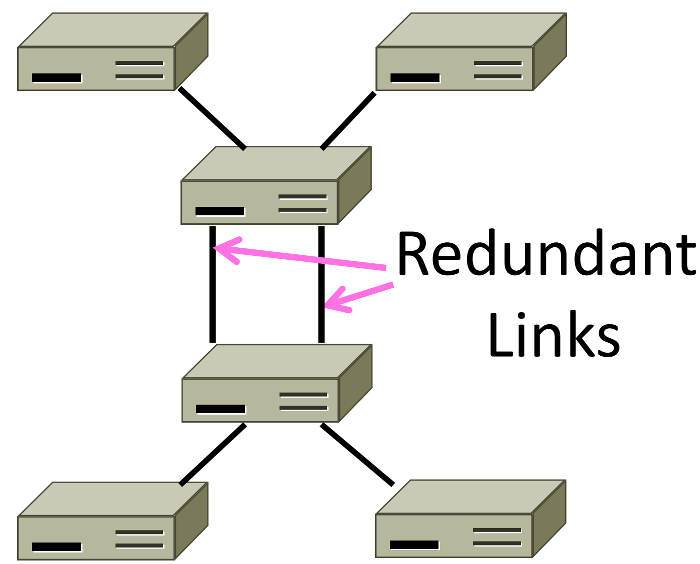


Loops – yikes!

Problem – Forwarding Loops

- May have a loop in the topology
  - Redundancy in case of failures
  - Or a simple mistake

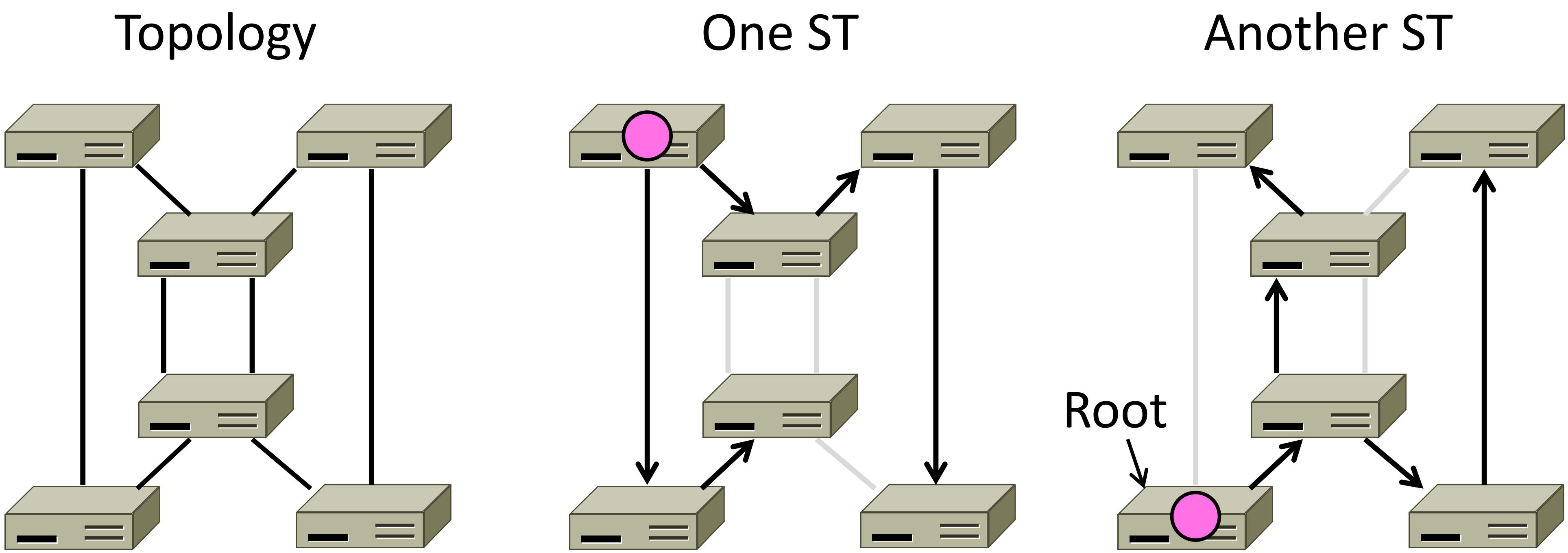
- Want LAN switches to "just work"
  - Plug-and-play, no changes to hosts
  - But loops cause a problem …



Spanning Tree Solution

- Switches collectively find a spanning tree for the topology
  - A subset of links that is a tree (no loops) and reaches all switches
  - Then switches forward as normal on the spanning tree
  - Broadcasts will go up to the root of the tree and down all the branches

Spanning Tree



Spanning Tree Algorithm

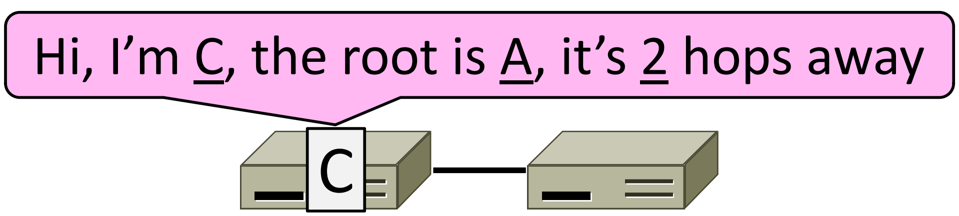
- Rules of the distributed game:
  - All switches run the same algorithm
  - They start with no information
  - Operate in parallel and send messages
  - Always search for the best solution
- Ensures a highly robust solution
  - Any topology, with no configuration
  - Adapts to link/switch failures, …

Spanning Tree Algorithm (2)

- Outline:
  1. Elect a root node of the tree (switch with the lowest address)
  2. Grow tree as shortest distances from the root (using lowest address to break distance ties)
  3. Turn off ports for forwarding if they aren't on the spanning tree

Spanning Tree Algorithm (3)

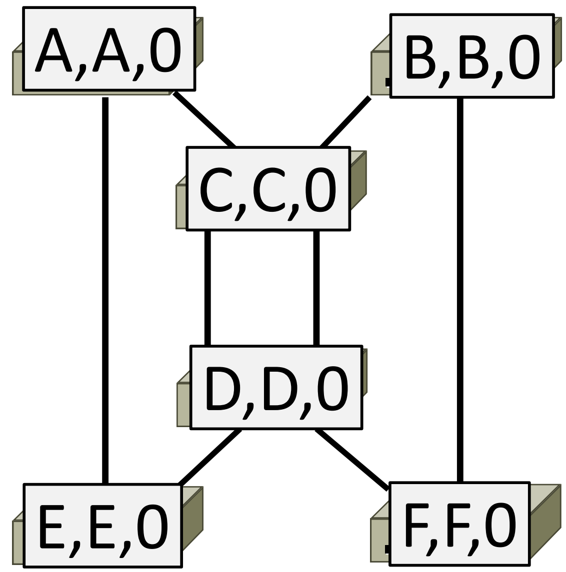
- Details:
  - Each switch initially believes it is the root of the tree
  - Each switch sends periodic updates to neighbors with:
    - Its address, address of the root, and distance (in hops) to root
  - Switches favors ports with shorter distances to lowest root
    - Uses lowest address as a tie for distances



or (C, A, 2)

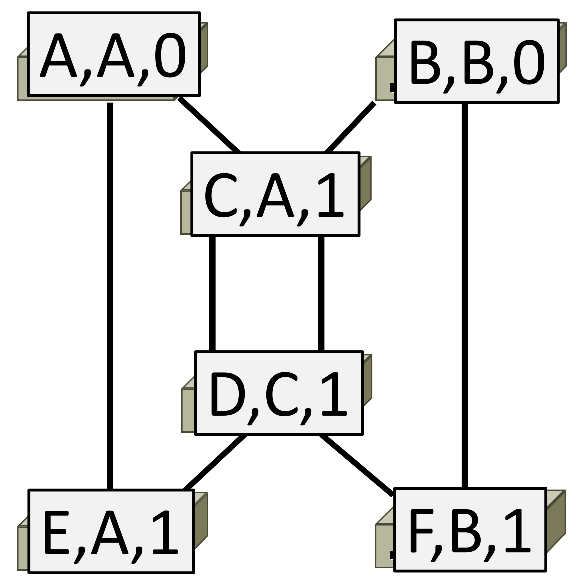
Spanning Tree Example

- 1st round, sending:
  - A sends (A, A, 0) to say it is root
  - B, C, D, E, and F do likewise
- 1st round, receiving:
  - A still thinks is it (A, A, 0)
  - B still thinks (B, B, 0)
  - C updates to (C, A, 1)
  - D updates to (D, C, 1)
  - E updates to (E, A, 1)
  - F updates to (F, B, 1)



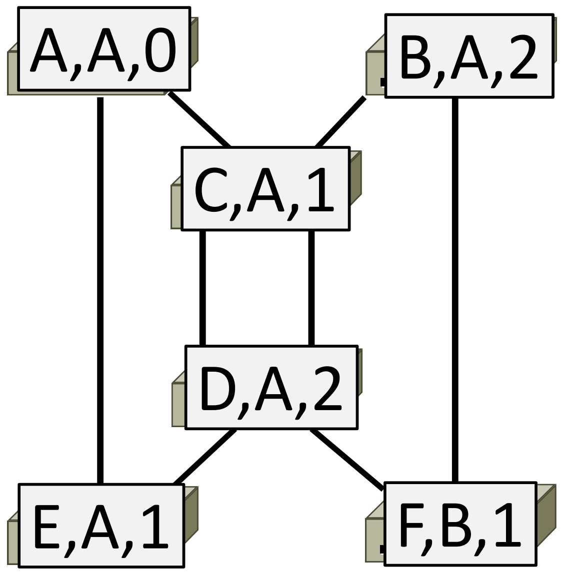
Spanning Tree Example (2)

- 2nd round, sending
  - Nodes send their updated state
- 2nd round receiving:
  - A remains (A, A, 0)
  - B updates to (B, A, 2) via C
  - C remains (C, A, 1)
  - D updates to (D, A, 2) via C
  - E remains (E, A, 1)
  - F remains (F, B, 1)



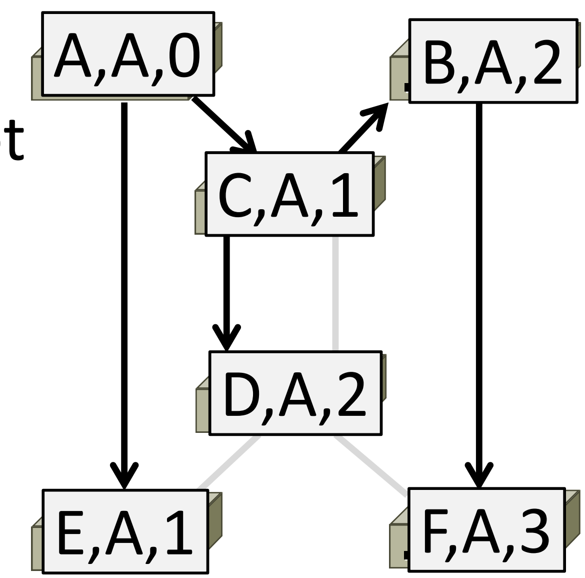
Spanning Tree Example (3)

- 3^{rd} round, sending
  - Nodes send their updated state
- 3^{rd} round receiving:
  - A remains (A, A, 0)
  - B remains (B, A, 2) via C
  - C remains (C, A, 1)
  - D remains (D, A, 2) via C-left
  - E remains (E, A, 1)
  - F updates to (F, A, 3) via B



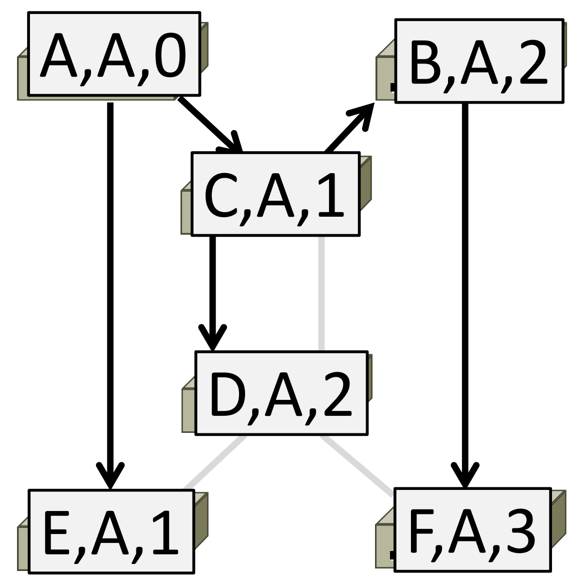
Spanning Tree Example (4)

- 4^{th} round
  - Steady-state has been reached
  - Nodes turn off forwarding that is not on the spanning tree
- Algorithm continues to run
  - Adapts by timing out information
  - E.g., if A fails, other nodes forget it, and B will become the new root



Spanning Tree Example (5)

- Forwarding proceeds as usual on the ST
- Initially D sends to F:



- And F sends back to D:

Computer Networks

IP Forwarding

IP Addresses

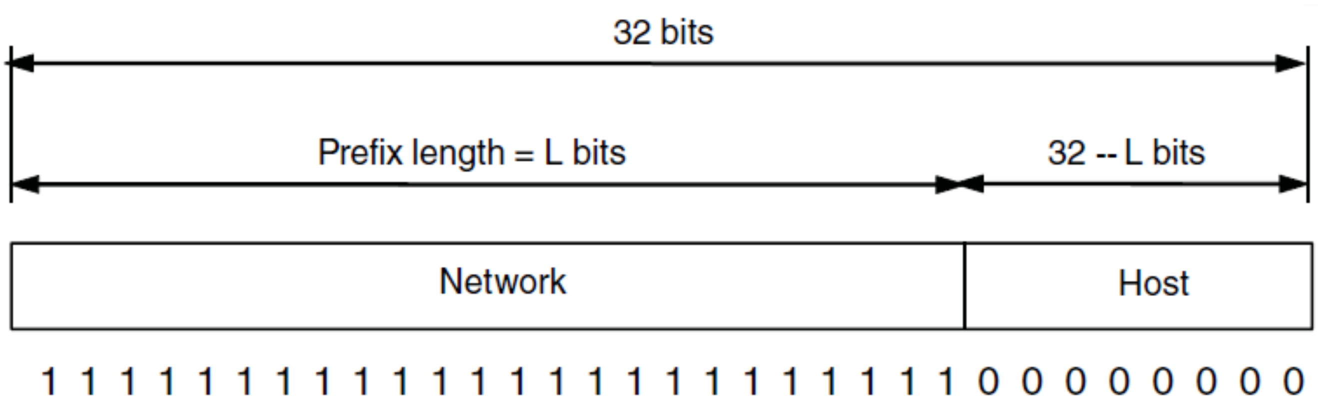
- IPv4 uses 32-bit addresses
  - IPv6 uses 128-bit addresses
- Written in "dotted quad" notation
  - Four 8-bit numbers separated by dots

$$\overbrace{\texttt{aaaaaaaa}}^{\text{8 bits}}\overbrace{\texttt{bbbbbbbb}}^{\text{8 bits}}\overbrace{\texttt{cccccccc}}^{\text{8 bits}}\overbrace{\texttt{dddddddd}}^{\text{8 bits}} \;\leftrightarrow\; \texttt{A.B.C.D}$$

$$\texttt{00010010} \mid \texttt{00011111} \mid \texttt{00000000} \mid \texttt{00000001} \;\leftrightarrow$$

IP Prefixes

- Addresses are allocated in blocks called <u>prefixes</u>
  - Addresses in an L-bit prefix have the same top L bits
  - There are 2^{32-L} addresses aligned on 2^{32-L} boundary



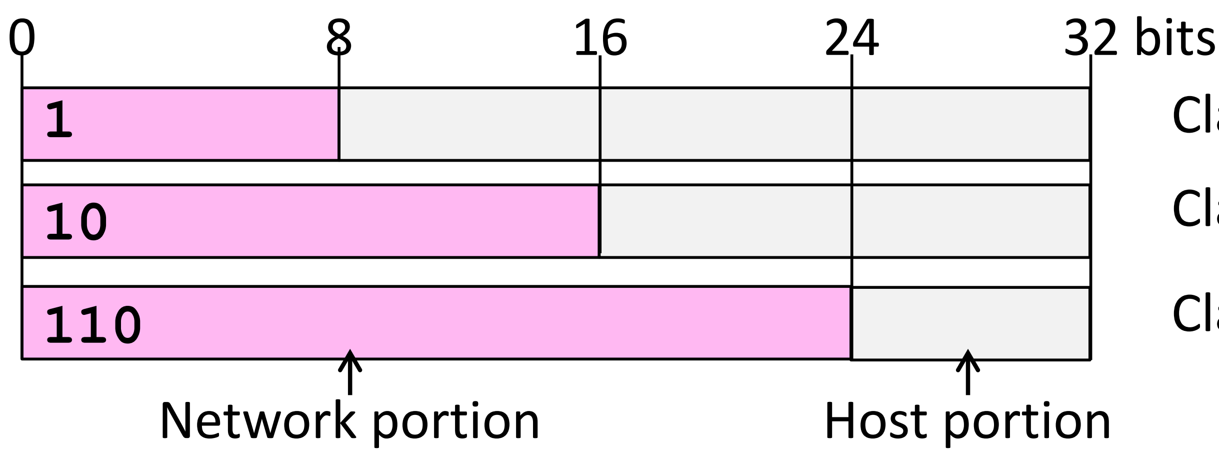
IP Prefixes (2)

- Written in "address/length" notation
  - Address is lowest address in the prefix, length is prefix bits
  - E.g., 128.13.0.0/16 is 128.13.0.0 to 128.13.255.255
  - So a /24 ("slash 24") is 256 addresses, and a /32 is one address

```
00010010|00011111|00000000|xxxxxxxx  ↔
        |        |        |
        |        |        |           ↔ 128.13.0.0/16
```

Classful IP Addressing

- Originally, IP addresses came in fixed size blocks with the class/size encoded in the high-order bits
  - They still do, but the classes are now ignored



Class A, 2^{24} addresses

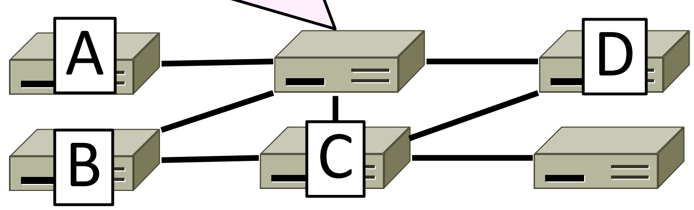
Class B, 2^{16} addresses

Class C, 2^{8} addresses

IP Forwarding

- All addresses on one network belong to the same prefix
- Node uses a table that lists the next hop for prefixes

Prefix	Next Hop
192.24.0.0/19	D
192.24.12.0/22	B



Longest Matching Prefix

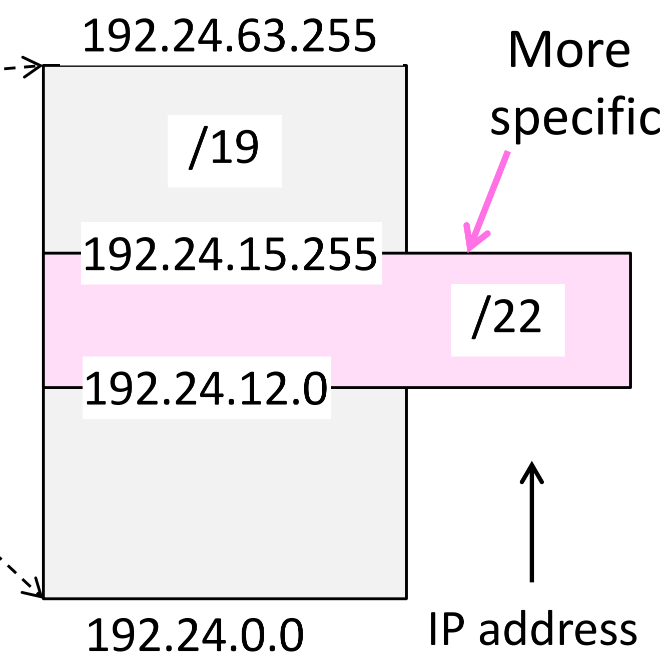
- Prefixes in the table might overlap!
  - Combines hierarchy with flexibility

- Longest matching prefix forwarding rule:
  - For each packet, find the longest prefix that contains the destination address, i.e., the most specific entry
  - Forward the packet to the next hop router for that prefix

Longest Matching Prefix (2)

Prefix	Next Hop
192.24.0.0/19	D
192.24.12.0/22	B

192.24.6.0 →
192.24.14.32 →
192.24.16.32 →



Flexibility of Longest Matching Prefix

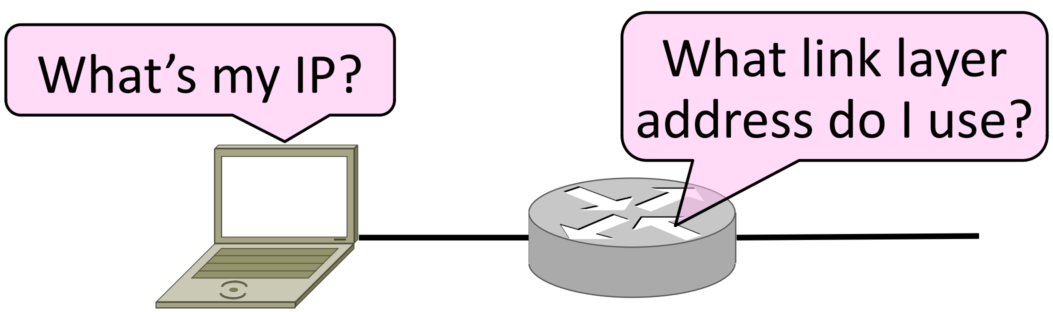
- Can provide default behavior, with less specifics
  - To send traffic going outside an organization to a border router
- Can special case behavior, with more specifics
  - For performance, economics, security, …

Performance of Longest Matching Prefix

- Uses hierarchy for a compact table
  - Relies on use of large prefixes
- Lookup more complex than table
  - Used to be a concern for fast routers
  - Not an issue in practice these days

Topic

- Filling in the gaps we need to make for IP forwarding work in practice
  - Getting IP addresses (DHCP) »
  - Mapping IP to link addresses (ARP) »

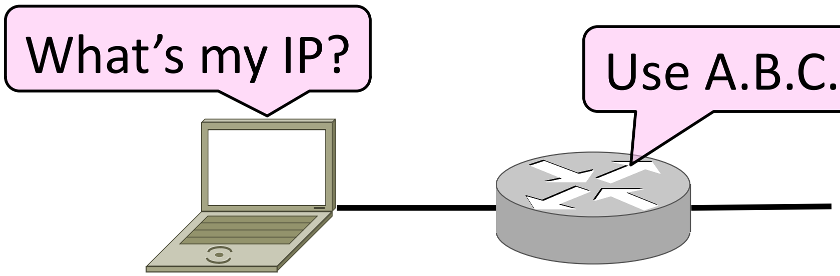


Getting IP Addresses

- Problem:
  - A node wakes up for the first time …
  - What is its IP address? What’s the IP address of its router? Etc.
  - At least Ethernet address is on NIC

Getting IP Addresses (2)

1. Manual configuration (old days)
   - Can't be factory set, depends on use
2. A protocol for automatically configuring addresses (DHCP)
   - Shifts burden from users to IT folk



DHCP

- DHCP (Dynamic Host Configuration Protocol), from 1993, widely used
- It leases IP address to nodes
- Provides other parameters too
  - Network prefix
  - Address of local router
  - DNS server, time server, etc.

DHCP Protocol Stack

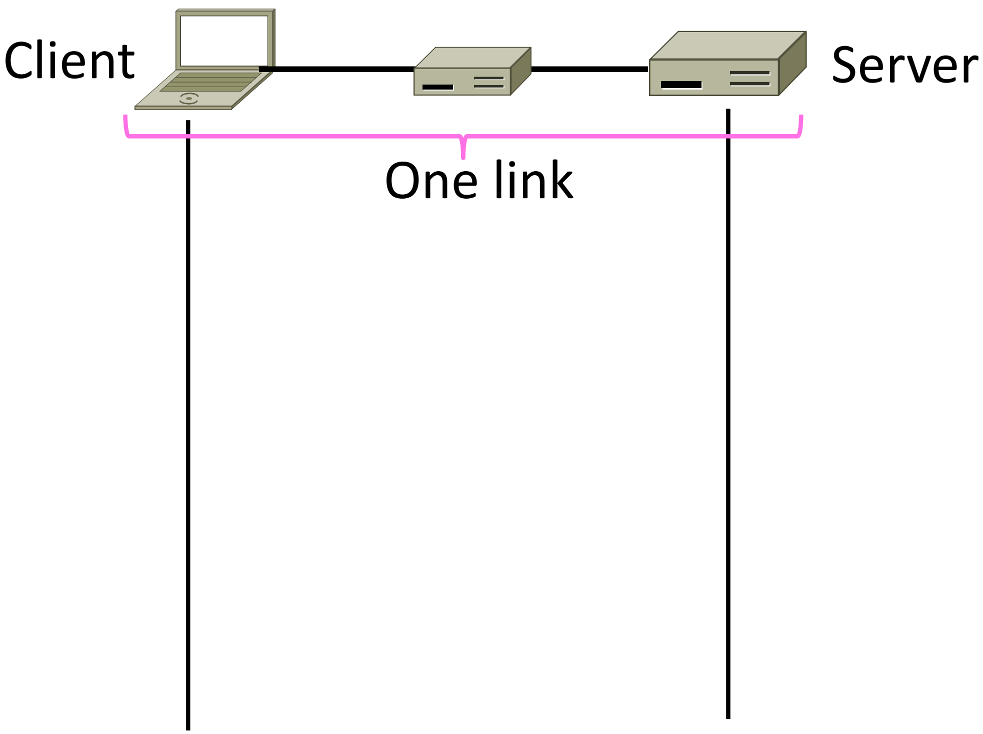
- DHCP is a client-server application
  - Uses UDP ports 67, 68

DHCP
UDP
IP
Ethernet

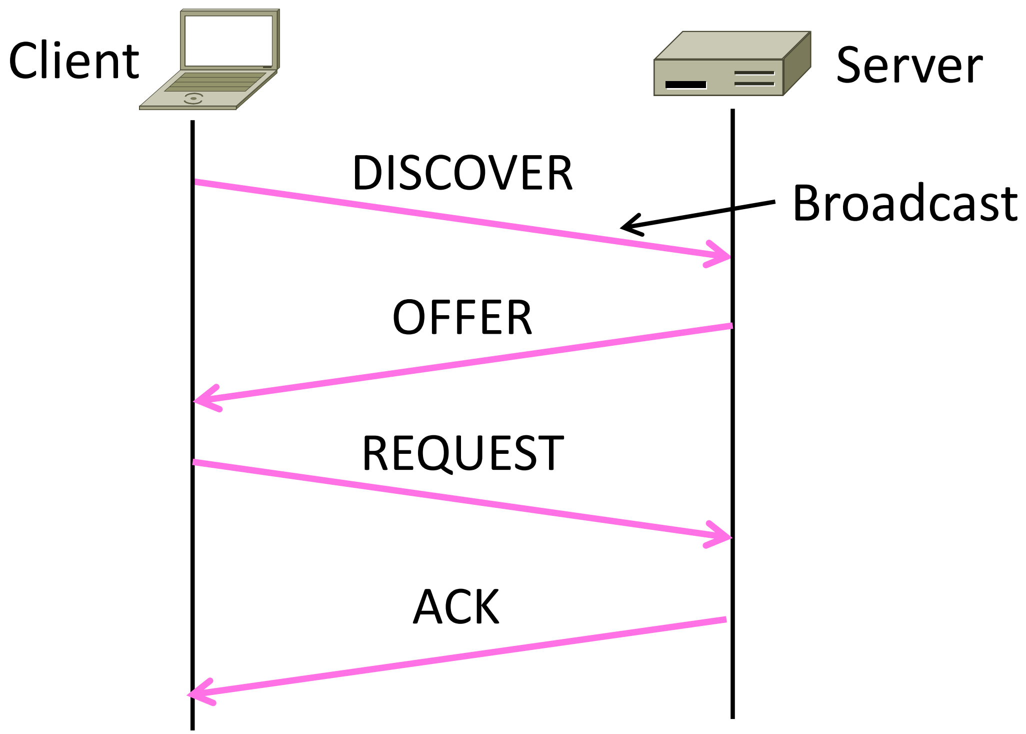
DHCP Addressing

- Bootstrap issue:
  - How does node send a message to DHCP server before it is configured?
- Answer:
  - Node sends broadcast messages that delivered to all nodes on the network
  - Broadcast address is all 1s
  - IP (32 bit): 255.255.255.255
  - Ethernet (48 bit): ff:ff:ff:ff:ff:ff

DHCP Messages



DHCP Messages (2)



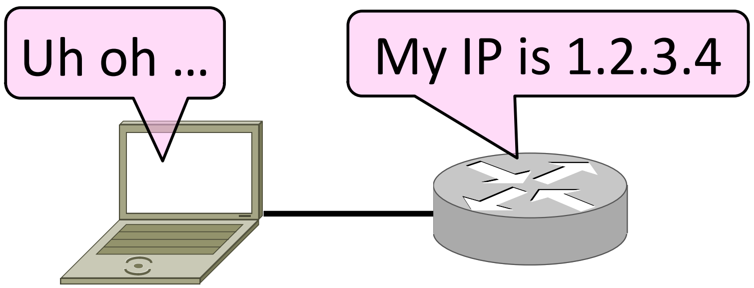
DHCP Messages (3)

- To renew an existing lease, an abbreviated sequence is used:
  - REQUEST, followed by ACK

- Protocol also supports replicated servers for reliability

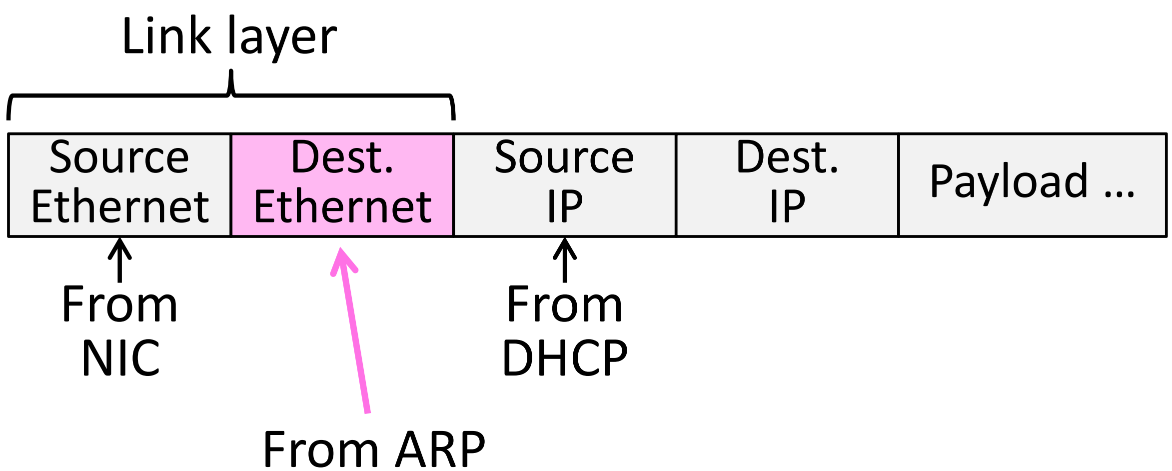
Sending an IP Packet

- Problem:
  - A node needs Link layer addresses to send a frame over the local link
  - How does it get the destination link address from a destination IP address?



ARP (Address Resolution Protocol)

- Node uses to map a local IP address to its Link layer addresses

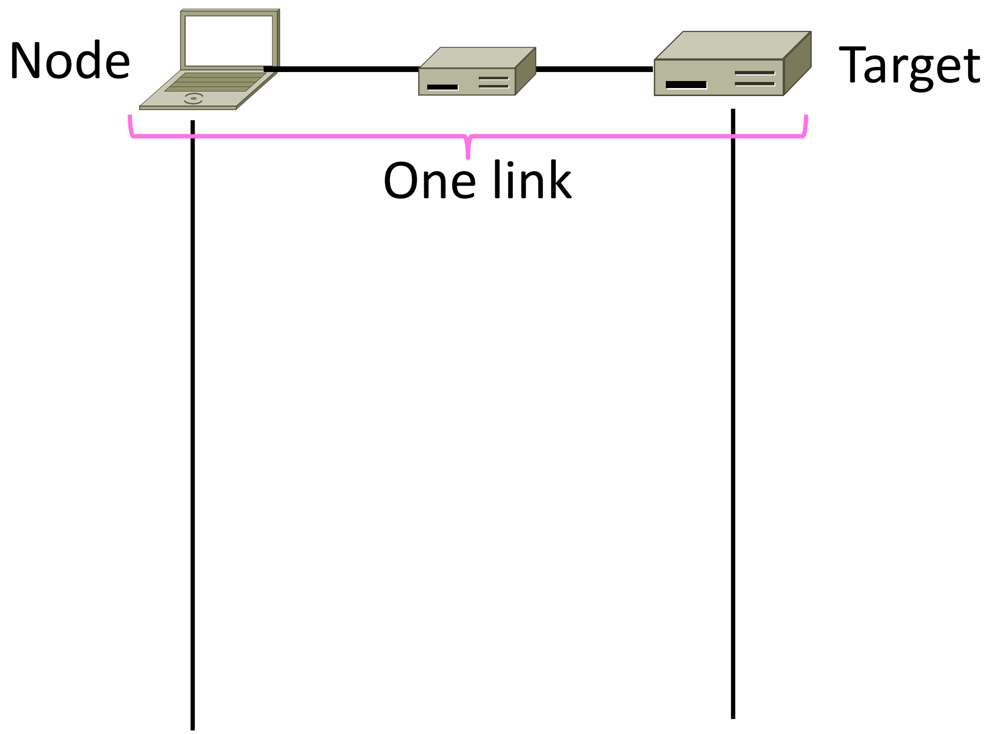


ARP Protocol Stack

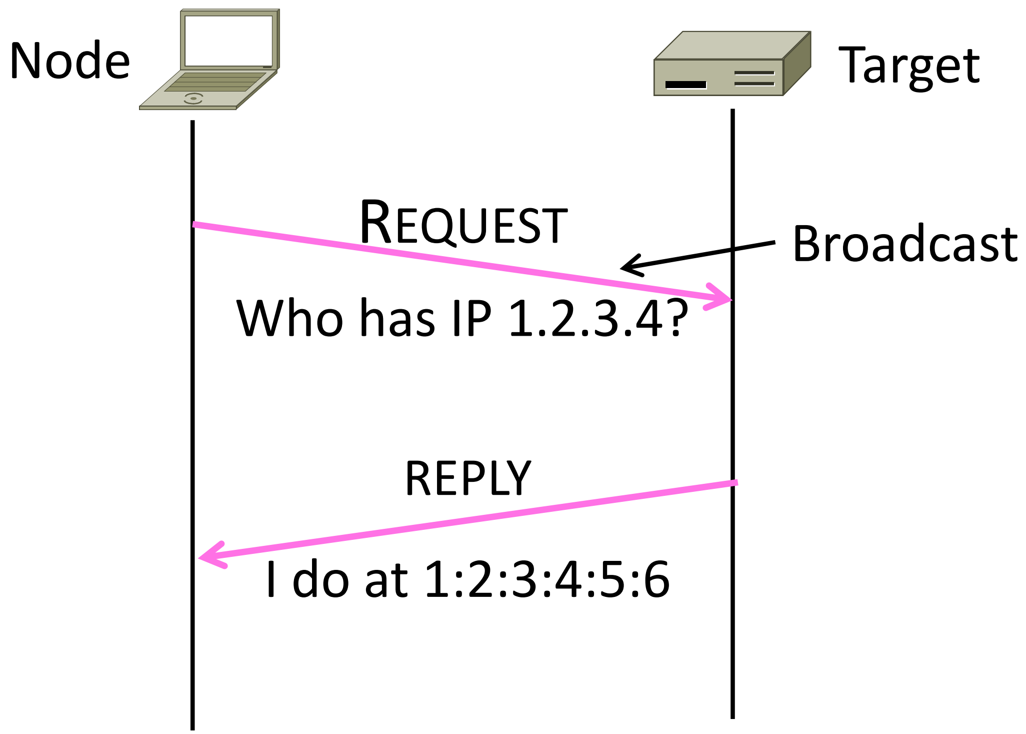
- ARP sits right on top of link layer
  - No servers, just asks node with target IP to identify itself
  - Uses broadcast to reach all nodes

ARP
Ethernet

ARP Messages

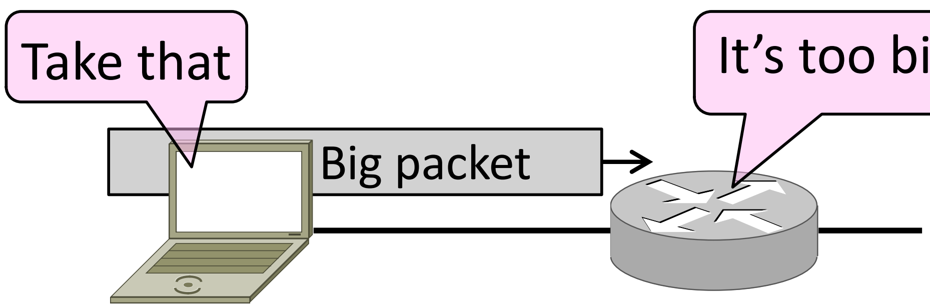


ARP Messages (2)



Topic

- How do we connect networks with different maximum packet sizes?
  - Need to split up packets, or discover the largest size to use



Packet Size Problem

- Different networks have different maximum packet sizes
  - Or MTU (Maximum Transmission Unit)
  - E.g., Ethernet 1.5K, WiFi 2.3K
- Prefer large packets for efficiency
  - But what size is too large?
  - Difficult because node does not know complete network path

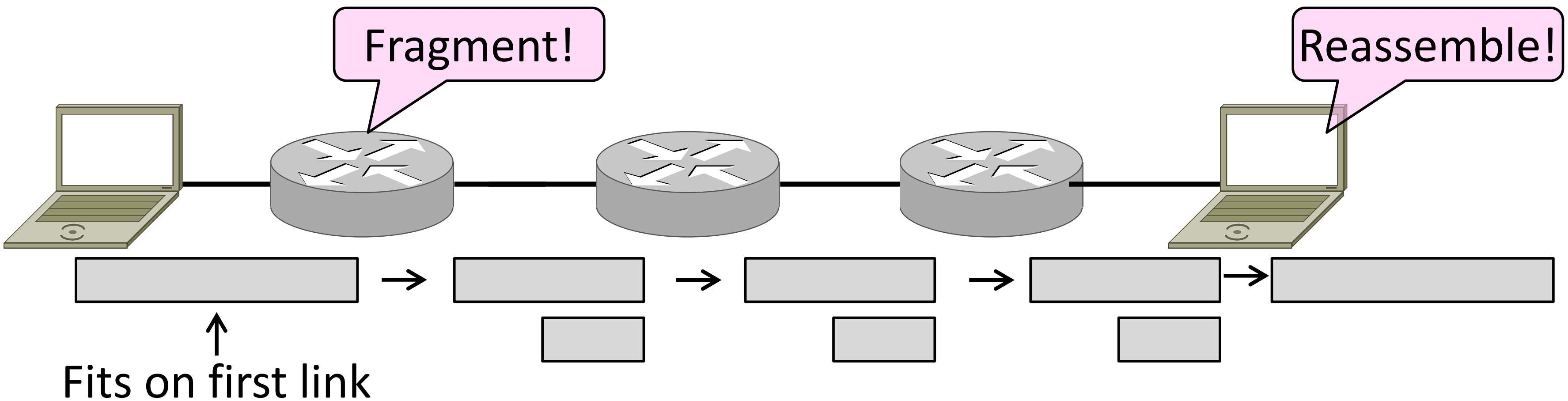
Packet Size Solutions

- Fragmentation (now)
  - Split up large packets in the network if they are too big to send
  - Classic method, dated

- Discovery (next)
  - Find the largest packet that fits on the network path and use it
  - IP uses today instead of fragmentation

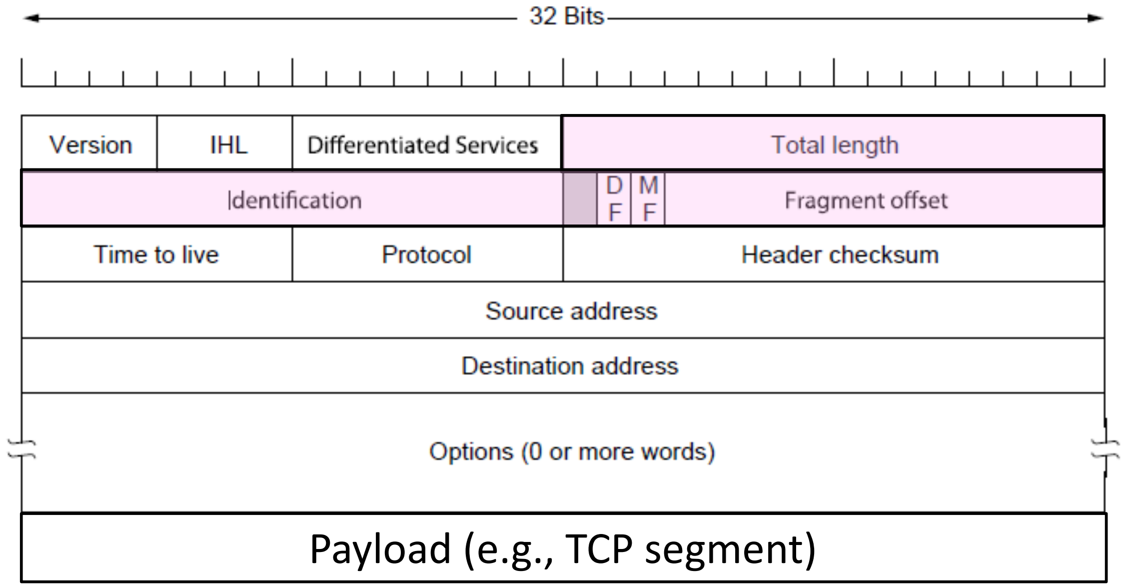
IPv4 Fragmentation

- Routers fragment packets that are too large to forward
- Receiving host reassembles to reduce load on routers



IPv4 Fragmentation Fields

- Header fields used to handle packet size differences
  - Identification, Fragment offset, MF/DF control bits



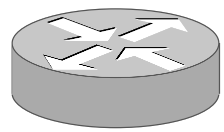
IPv4 Fragmentation Procedure

- Routers split a packet that is too large:
  - Typically break into large pieces
  - Copy IP header to pieces
  - Adjust length on pieces
  - Set offset to indicate position
  - Set MF (More Fragments) on all pieces except last
- Receiving hosts reassembles the pieces:
  - Identification field links pieces together, MF tells receiver when it has all pieces

IPv4 Fragmentation (3)

Before
MTU = 2300

ID = 0x12ef
Data Len = 2300
Offset = 0
MF = 0



After
MTU = 1500

ID = 0x12ef
Data Len = 1500
Offset = 0
MF = 1

ID = 0x12ef
Data Len = 800
Offset = 1500
MF = 0

IPv4 Fragmentation (4)

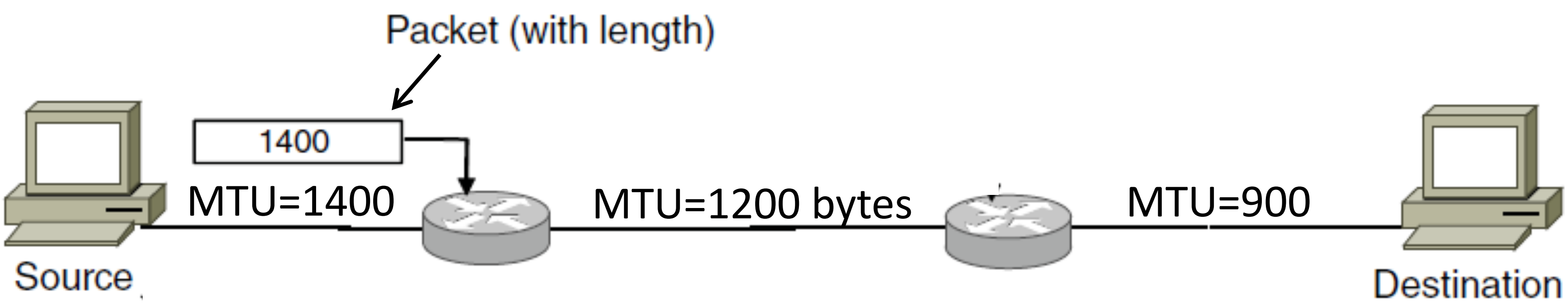
- It works!
  - Allows repeated fragmentation
- But fragmentation is undesirable
  - More work for routers, hosts
  - Tends to magnify loss rate
  - Security vulnerabilities too

Path MTU Discovery

- Discover the MTU that will fit
  - So we can avoid fragmentation
  - The method in use today

- Host tests path with large packet
  - Routers provide feedback if too large; they tell host what size would have fit

Path MTU Discovery (2)



Path MTU Discovery (3)