CSCI-1680
Building Links and (Local) Networks

Nick DeMarinis

Based partly on lecture notes by Rodrigo Fonseca, David Mazières, Phil Levis, John Jannotti

Administrivia

- Snowcast due Monday (9/25) by 11:59pm EDT
- I'm debugging a few tester issues now
  - See FAQ post on Ed for known issues!
  - As long as manual testing works with the reference, you're fine
    => Document problems in your README
- Look for update on HW1 on Friday

Administrivia

- Tuesday 9/26: IP project out
  - You will work in groups of two
  - We will send a form today/tomorrow where you can specify your group, or ask to be matched to a group
  - Matching happens based on language, in-person/remote, etc.

Fill out form by Tuesday, 9/26!

Today

Last time: how to send over a link

Today: how to build a network with links?

- Sharing links
- Case study: Ethernet (and Wifi)
  - *Network interfaces:* How you interact with the link layer
- How switching works

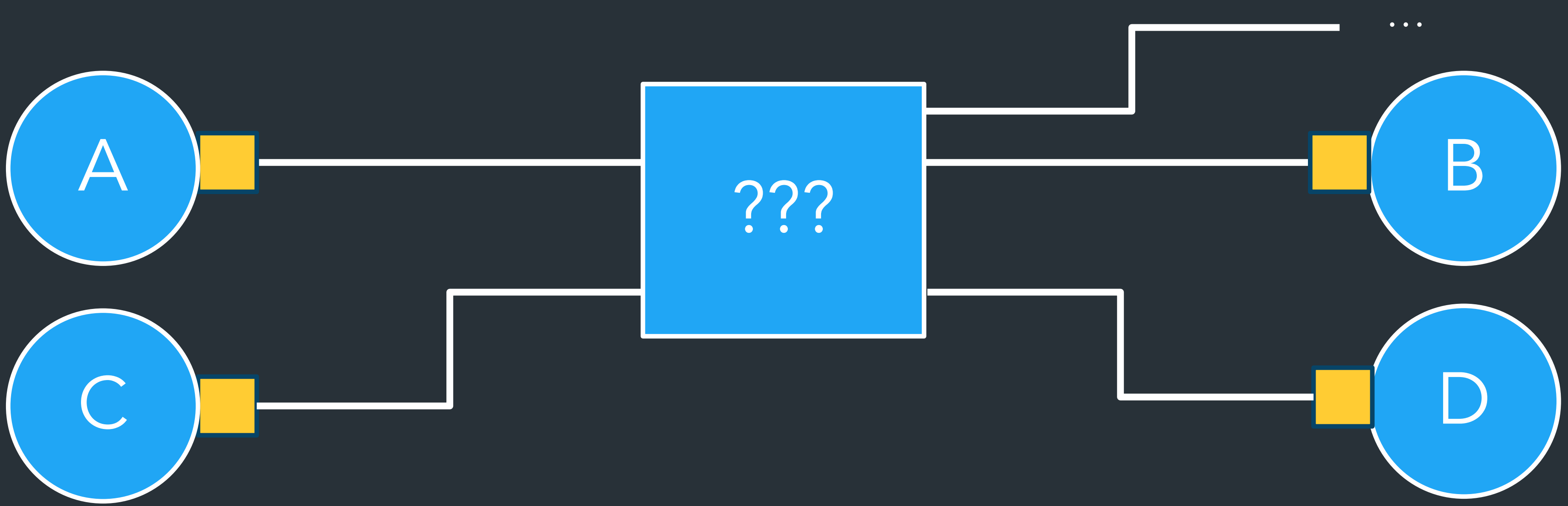
What does “link layer” mean?

Application	Service: user-facing application. Application-defined messages
Transport	Service: multiplexing applications Reliable byte stream to other node (TCP), Unreliable datagram (UDP)
Network	Service: move packets to any other node in the network Internet Protocol (IP)
Link	**Service: move frames to other node across link. May add reliability, medium access control**
Physical	Service: move bits to other node across link

The main idea

Sending bits over a channel....

What does "link layer" mean?



- Multiple hosts => shared channel
- Need ways to allow "small" number of hosts to communicate

"Small" => Within a building, floor of office, etc
Related term: Local Area Network (LAN)

How to share the channel?

Medium Access Control (MAC)

Medium Access Control

Idea: Control access to shared physical medium
=> No more than one device can be "talking" at one time

Need a protocol for "who can talk when?"

An example of *multiplexing* => sharing the channel among multiple devices

High-level: MAC approaches

Partitioned Access: divide the channel into fixed slots

- Time Division Multiple Access (TDMA)
- Frequency Division Multiple Access (FDMA)
- Code Division Multiple Access (CDMA)

Problems?

⇒ Hard to maximize channel utilization
(eg. what happens if only one person is talking?)

High-level: MAC approaches

Random Access: no fixed slots: "ask" to talk, or just talk and hope for the best

- Carrier Sense Multiple Access / Collision Detection (CSMA/CD)
- Carrier Sense Multiple Access / Collision Avoidance (CSMA/CA)
- RTS/CTS (Request to Send/Clear to Send)
- Token-based

Problems?

⇒ Hard to maintain "fairness"
(eg. one host dominating channel)

Why does this matter?

Different types of links solve these problems differently

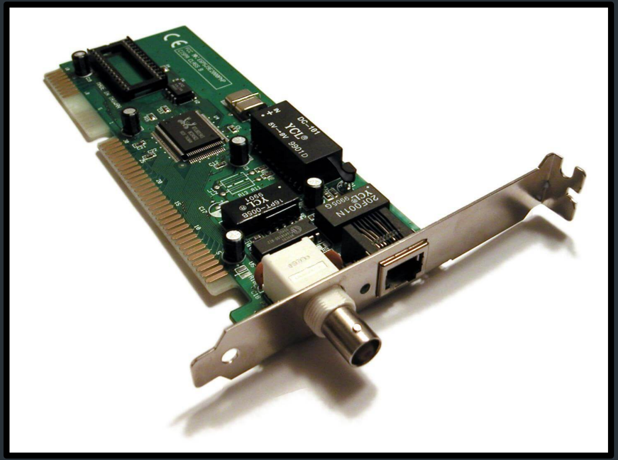
- Ethernet (wired) vs. Wifi (wireless)
- Affects throughput, reliability, etc.

Understand why different links operate differently
=> How we build the Internet from them

Interface: device that connects something to a network

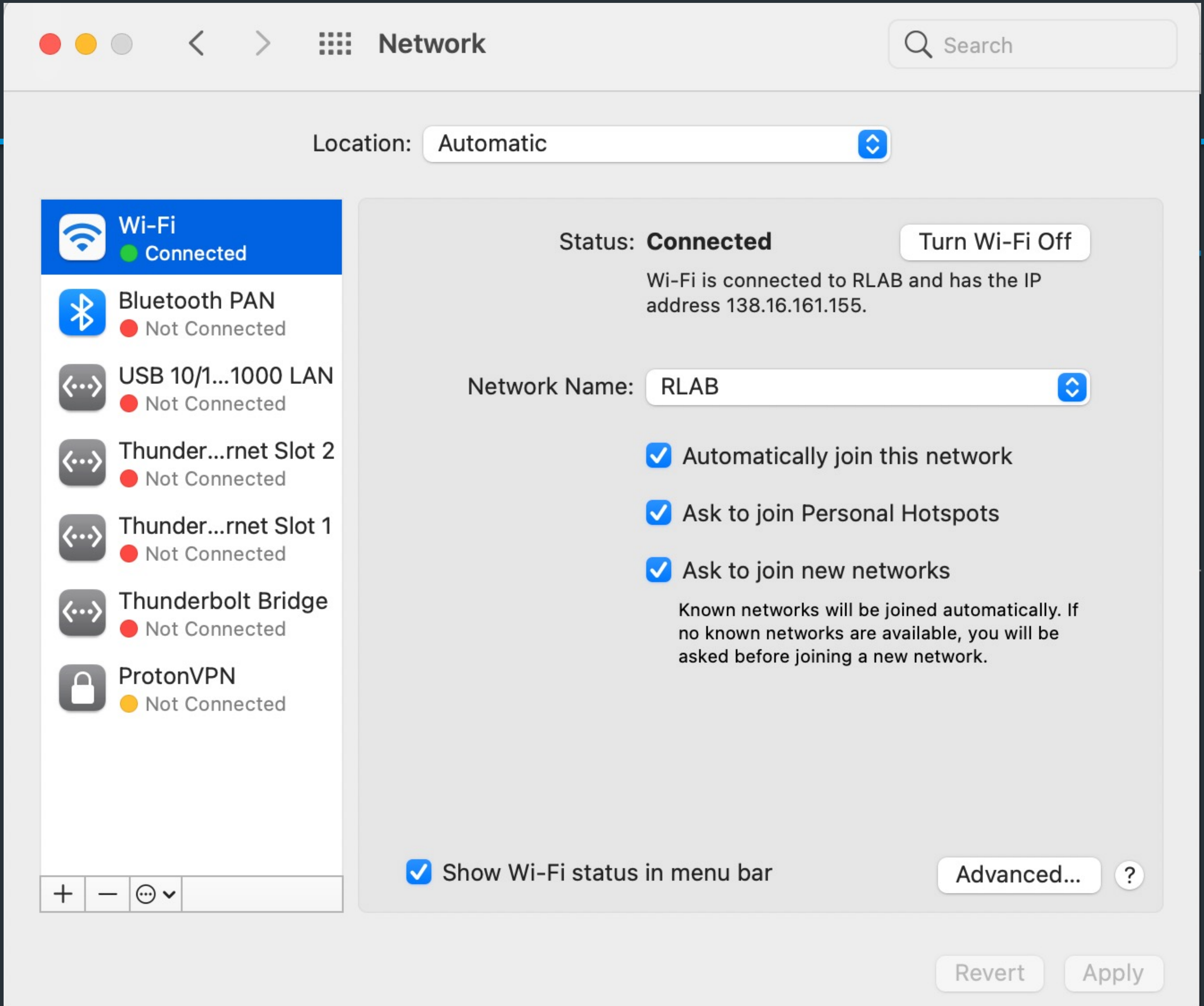
- OS abstraction for a network device
- Physical hardware that does the "talking"

  => Network Interface Card (NIC)



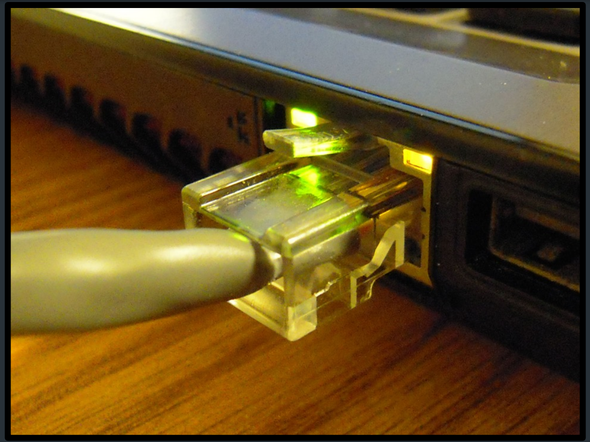
<u>Common interfaces</u>

- Loopback: Virtual, only for local host
- Wifi, Ethernet, Bluetooth, …


Network
Search
Location: Automatic
Wi-Fi
Connected
Bluetooth PAN
Not Connected
USB 10/1...1000 LAN
Not Connected
Thunder...rnet Slot 2
Not Connected
Thunder...rnet Slot 1
Not Connected
Thunderbolt Bridge
Not Connected
ProtonVPN
Not Connected
Status: Connected
Turn Wi-Fi Off
Wi-Fi is connected to RLAB and has the IP address 138.16.161.155.
Network Name: RLAB
Automatically join this network
Ask to join Personal Hotspots
Ask to join new networks
Known networks will be joined automatically. If no known networks are available, you will be asked before joining a new network.
Show Wi-Fi status in menu bar
Advanced...
?
Revert
Apply

Example: Ethernet

Ethernet



Dominant wired LAN technology, has evolved significantly over time

- Original version (1983): 10Mbps
- Now (commonly): 1Gbps
- Also: 10Gbps, 40Gbps, …

New developments in physical media, encodings, hardware => higher speeds over time

Ethernet: software viewpoint

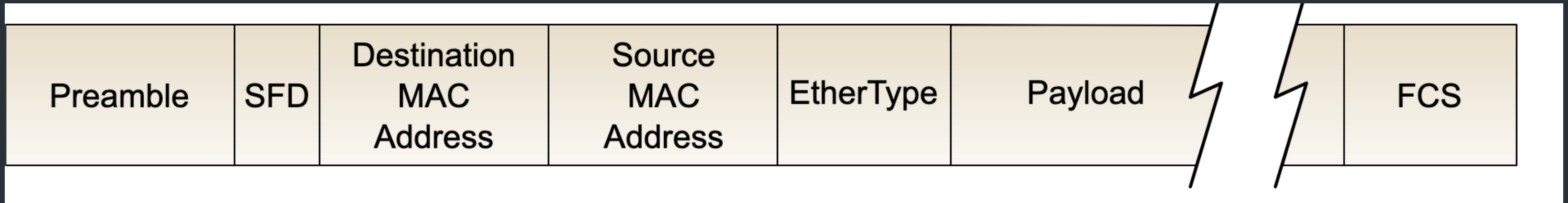
- All hosts have an “ethernet address”

 => Globally-unique identifier

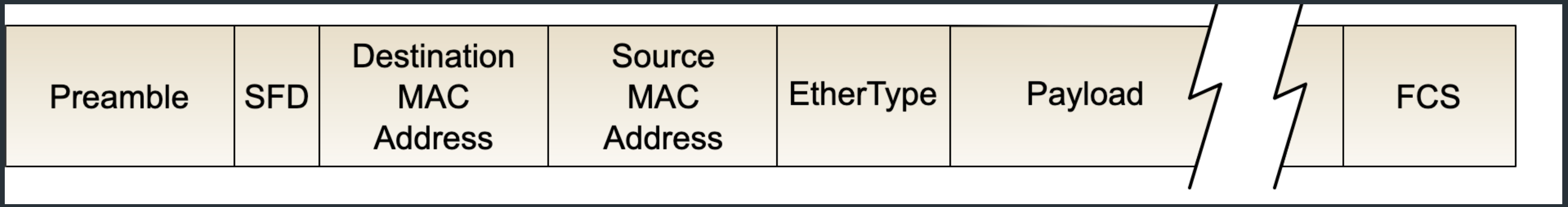
- Logically all hosts are connected to each other

- If you know a host’s ethernet address, you can send to it

Ethernet: the header



Ethernet: the header

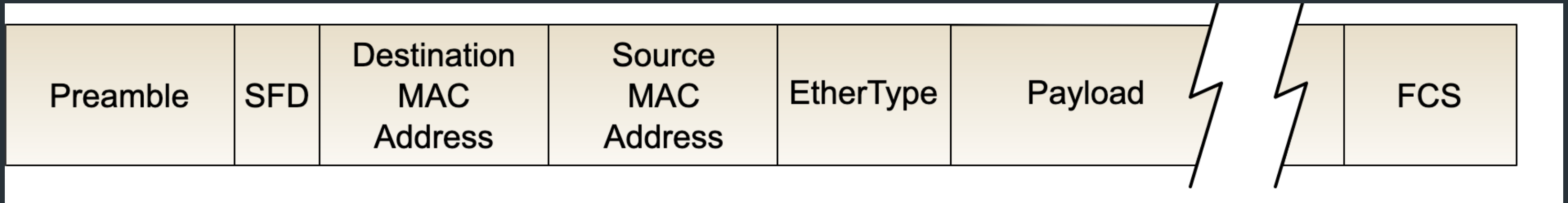


- Source address: where packet is from
- Destination address: where packet is going
  - ⇒ Devices ask: "Is this my packet?" "Where should I send this packet?"

Other stuff

- Preamble: when a packet starts
- FCS: Frame Check sequence (checksum)

Ethernet Addressing



Globally unique, 48-bit unicast address per adapter

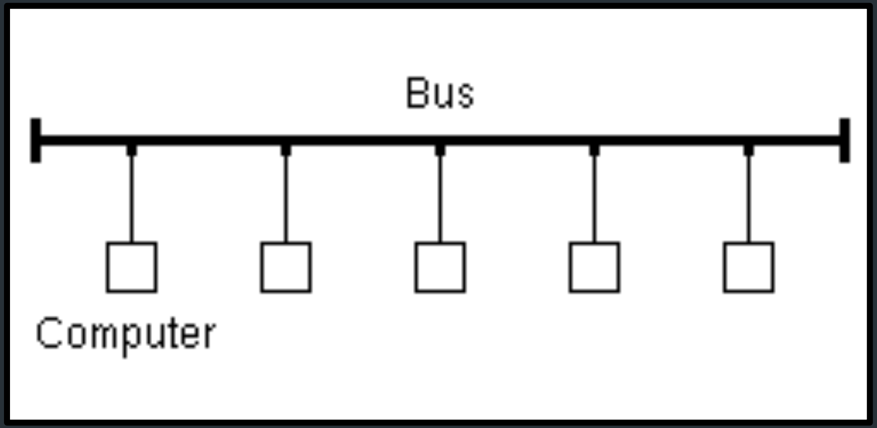
- Example: 00:1c:43:00:3d:09 (Samsung adapter)
- First 24 bits: Registered to manufacturers

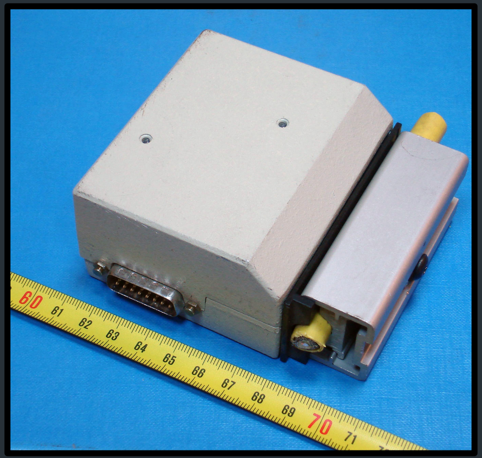
Other protocols have adopted this address format (eg. Wifi, Bluetooth, …)

=> Nowadays, we call them "mac addresses" or "hardware addresses"

Ethernet's evolution

Originally, a shared medium with all hosts





- Basic idea: all hosts can see all frames, read a frame if it matches your hardware address
- Implications?

=>Can have collisons!

Classical Ethernet: Problems

- Problem: shared medium, all hosts in the same "collision domain"
- Transmit algorithm
  - If line is idle, transmit immediately
  - Upper bound message size of 1500 bytes
  - If line is busy: wait until idle and transmit immediately
- Generally possible to <u>detect</u> collisions, deal with it

CSMA/CD: Carrier Sense Multiple Access / Collision Detection

When to transmit again?

- Delay and try again: exponential backoff
- *n*th time: $k \times 51.2\mu s$, for $k = U\{0..(2^{\min(n,10)}-1)\}$
  - 1st time: 0 or 51.2μs
  - 2nd time: 0, 51.2, 102.4, or 153.6μs
- Give up after several times (usually 16)

- Exponential backoff is a useful, general technique

Does this scale?

Ethernet Recap

- Service provided: send frames among stations with specific addresses
- All nodes in the same “collision domain”

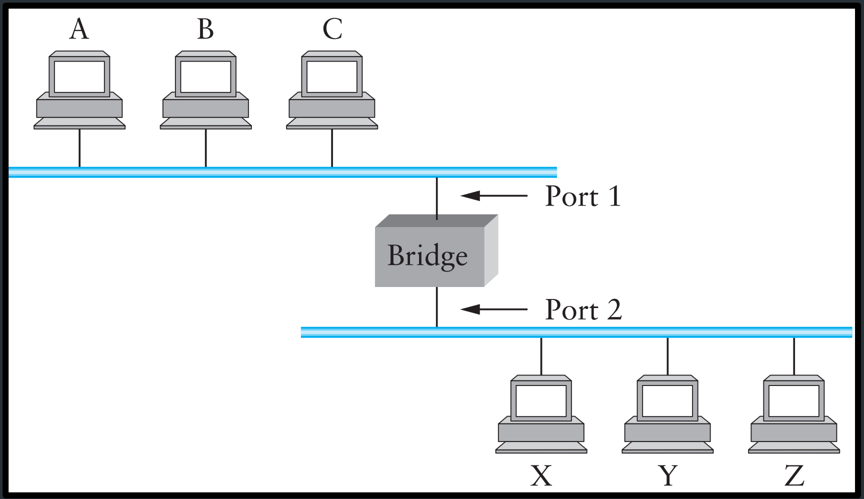
Avoiding collisions

- Early method: bridging

Avoiding collisions

Add some hardware to the network that can separate collision domains

Original way (1990s): bridges



Modern way: switching

Switch: network device that forwards frames (packets) between *ports*

- All hosts connect to a switch
- Collision domain is host-switch
- Switch buffers packets, forwards to destination when its port is idle

How to know which devices is on which port?

MAC learning, how it works

MAC Learning

- Switches “learn” which host lives on which port by watching traffic

  =>Keeps table of <destination addr => port>

- If you don’t know, flood to all ports!

MAC Learning

- Switches "learn" which host lives on which port by watching traffic
- If you don't know, flood to all ports!

MAC learning is just an optimization vs. old version
(but a pretty good one…)

MAC table example

```
R6#sh mac-address-table
EHWIC: 0
Destination Address Address Type VLAN Destination Port
------------------- ------------ ---- ------------------
5c45.27e0.8383            Dynamic          1 GigabitEthernet0/1/3
7641.7b63.584a            Dynamic         20 GigabitEthernet0/1/3
5c45.27e0.8381            Dynamic         10 GigabitEthernet0/1/3
0000.5e00.0101            Dynamic         10 GigabitEthernet0/0/1
ca3f.aee3.e3e6            Dynamic         20 GigabitEthernet0/1/3
644b.f012.7f75.           Dynamic         20 GigabitEthernet0/1/3
f018.9815.8eb8            Dynamic         20 GigabitEthernet0/1/3
ecb5.fa13.4677            Dynamic         20 GigabitEthernet0/0/2
a0a4.c5c2.4165            Dynamic         20 GigabitEthernet0/0/1
4c71.0c92.4f10            Dynamic         10 GigabitEthernet0/1/3
12d3.acae.bbc0            Dynamic         20 GigabitEthernet0/0/1
04d4.c448.9cf7            Dynamic         20 GigabitEthernet0/1/3
```

What can go wrong?

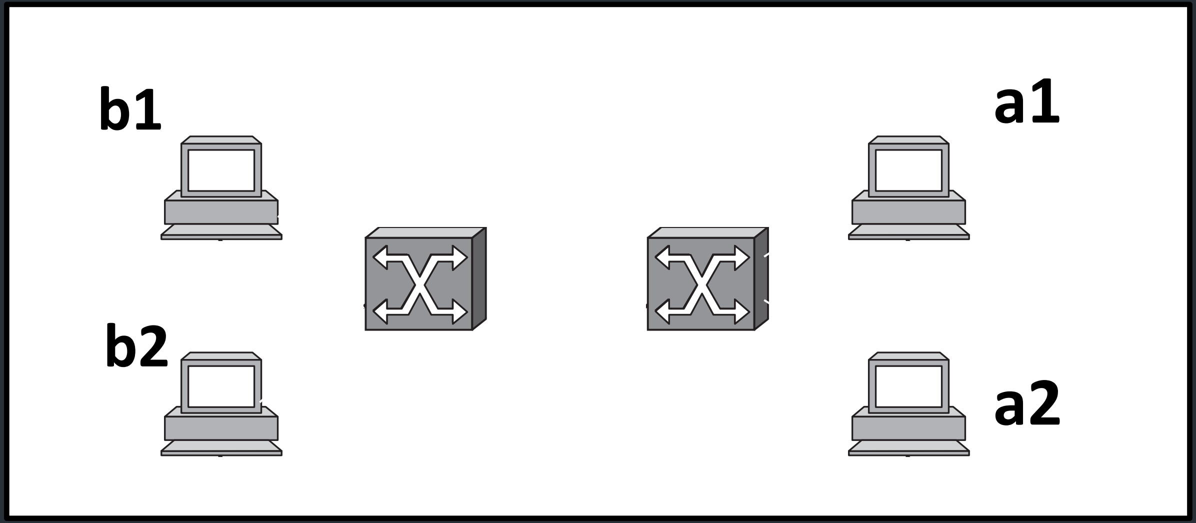
Attack on a Learning Switch

- Eve: wants to sniff all packets sent to Bob
- Same segment: easy (shared medium)
- Different segment on a learning bridge: hard
  - Once bridge learns Bob's port, stop broadcasting
- How can Eve force the bridge to keep broadcasting?
  - Flood the network with frames with spoofed src addr!

Also: VLANs

Consider: Company network, A and B departments

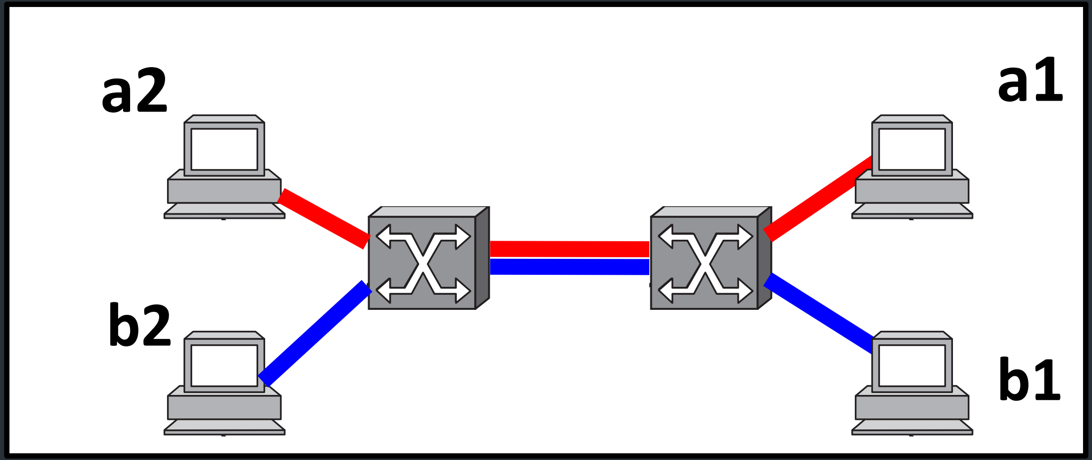
- Broadcast traffic does not scale
- May not *want* traffic between the two departments
- What if employees move between offices?



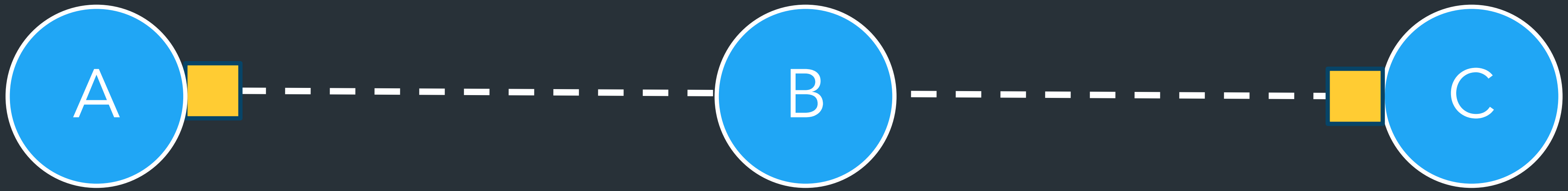
VLANs

- Solution: Virtual LANs
  - Assign switch ports to a VLAN ID (color)
  - Isolate traffic: only same color
  - Some links may belong to multiple VLANs

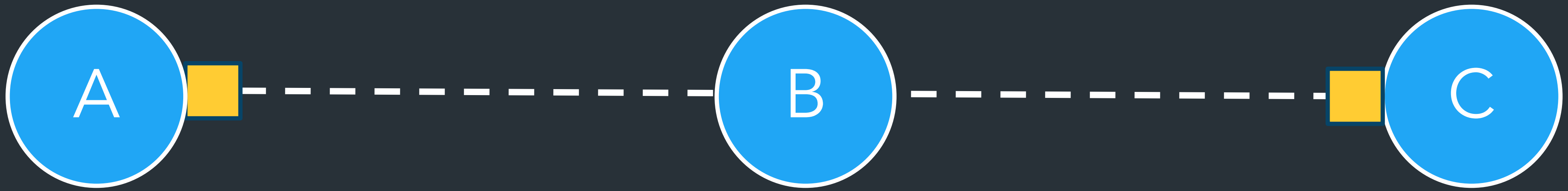
=> Easy to change, no need to rewire



How does this all change with wifi?



How does this all change with wifi?



Can’t detect collisions anymore!

=> Carrier Sense Multiple Access / Collision Avoidance

=> Try to send: if you don’t hear back, assume collision (and maybe retry)

Extra material

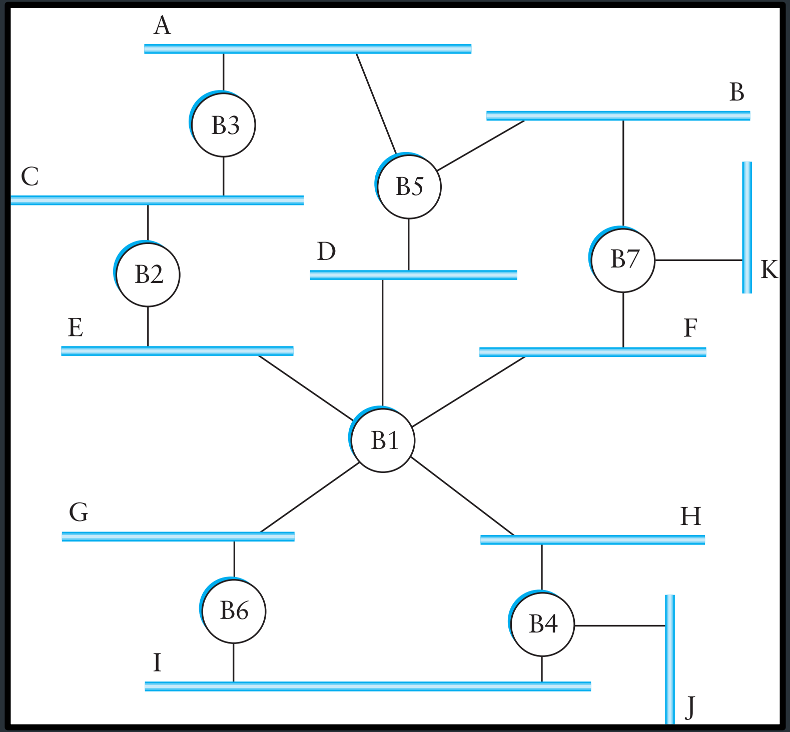
Coming Up

- Connecting multiple networks: IP and the Network Layer

Dealing with Loops

Problem: people may create loops in LAN!

- Accidentally, or to provide redundancy
- Don’t want to forward packets indefinitely

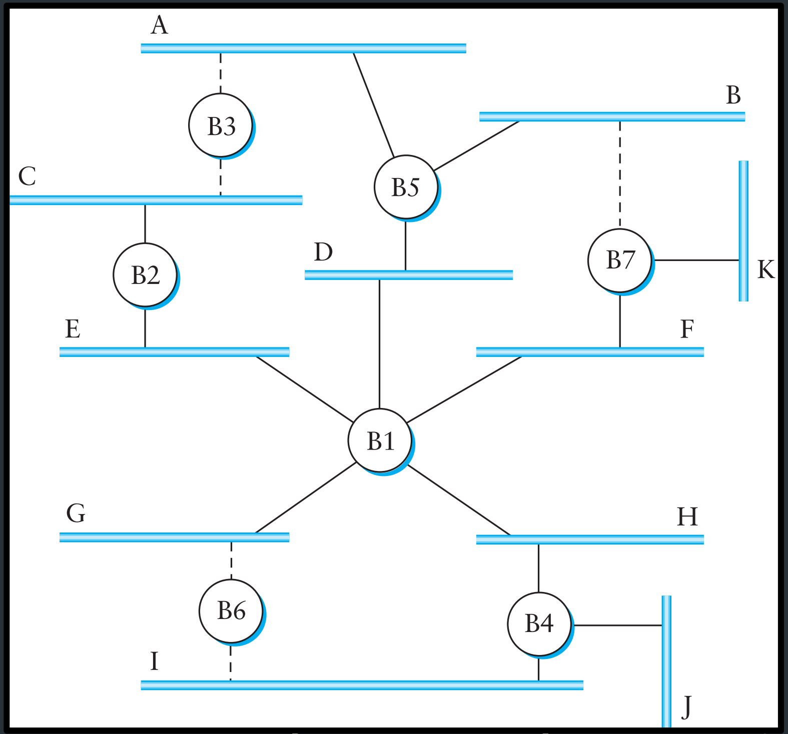


Enter Radia Perlman

"…we have designed an algorithm that allows the extended network to consist of an arbitrary topology. (…) The algorithm (…) computes a subset of the topology that connects all LANs yet is loop-free (a spanning tree)."

Perlman, Radia (1985). "An Algorithm for Distributed Computation of a Spanning Tree in an Extended LAN". *ACM SIGCOMM Computer Communication Review*. **15** (4): 44–53. doi:10.1145/318951.319004

Spanning Tree



- Need to disable ports, so that no loops in network
- Like creating a spanning tree in a graph
  - View switches and networks as nodes, ports as edges

Distributed Spanning Tree Algorithm

- Every bridge has a unique ID (Ethernet address)
- Goal:
  - Bridge with the smallest ID is the root
  - Each segment has one designated bridge, responsible for forwarding its packets towards the root
    - Bridge closest to root is designated bridge
    - If there is a tie, bridge with lowest ID wins

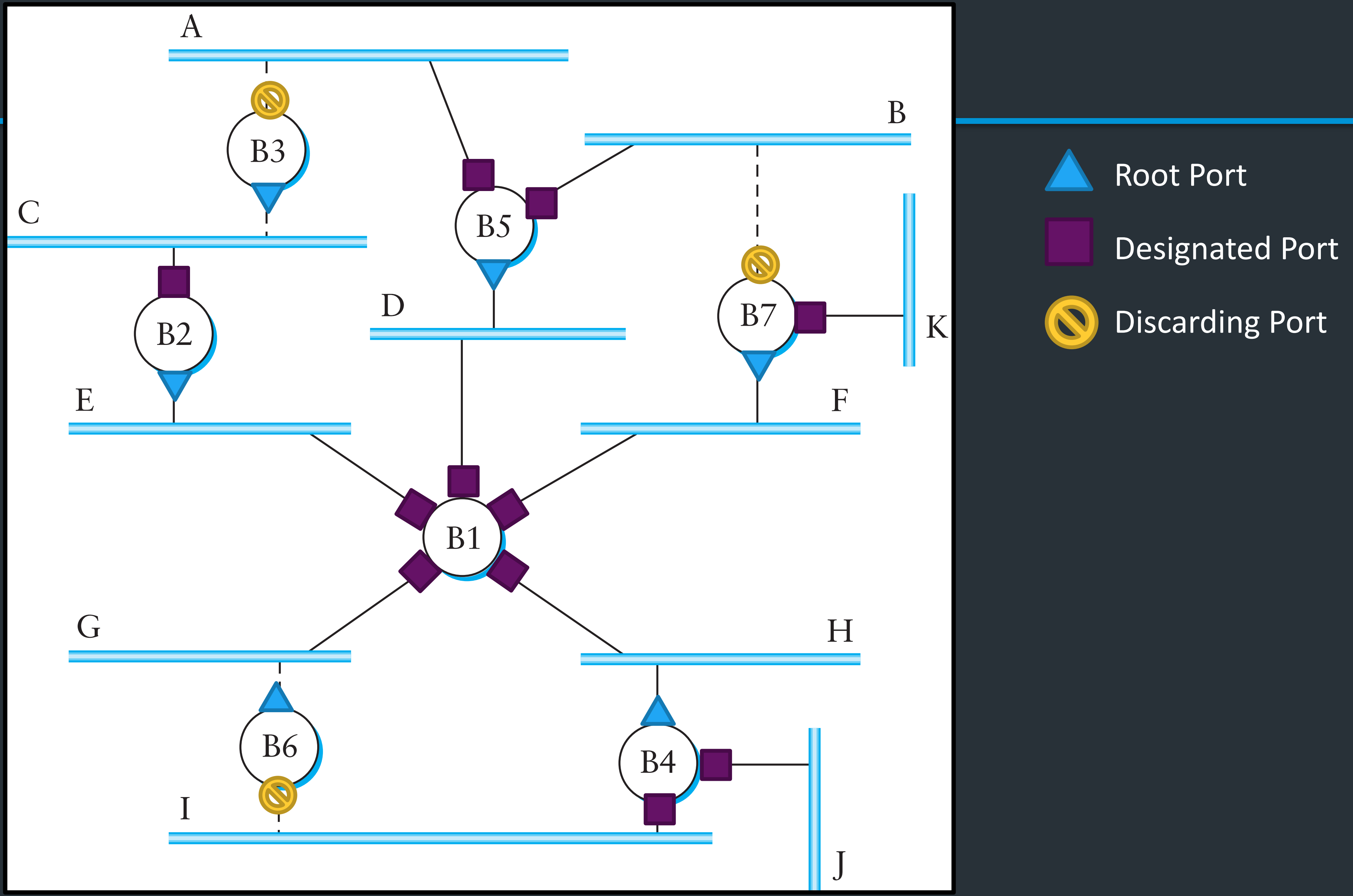
Spanning Tree Protocol

- Send message when you think you are the root
- Otherwise, forward messages from best known root
  - Add one to distance before forwarding
  - Don’t forward over *discarding ports* (see next slide)
- Spanning Tree messages contain:
  - ID of bridge sending the message
  - ID sender believes to be the root
  - Distance (in hops) from sender to root
- Bridges remember best config msg on each port
- In the end, only root is generating messages

Spanning Tree Protocol (cont.)

- Forwarding and Broadcasting
- Port states*:
  - Root port: a port the bridge uses to reach the root
  - Designated port: the lowest-cost port attached to a single segment
  - If a port is not a root port or a designated port, it is a discarding port.

* In a later protocol RSTP, there can be ports configured as backups and alternates.



A
B
C
D
E
F
G
H
I
J
K
B1
B2
B3
B4
B5
B6
B7
Root Port
Designated Port
Discarding Port

Algorhyme

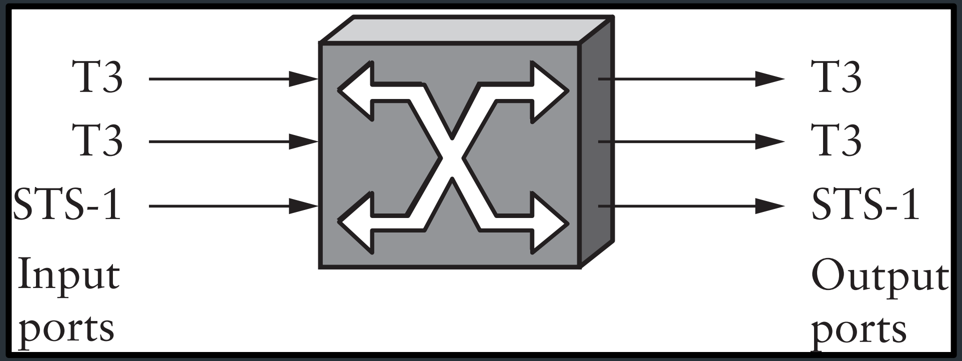
I think that I shall never see
a graph more lovely that a tree.
A tree whose crucial property
is loop-free connectivity.
A tree that must be sure to span
so packet can reach every LAN.
First the root must be selected.
By ID, it is elected.
Least cost paths from root are traced.
In the tree, these paths are placed.
A mesh is made by folks like me,
then bridges find a spanning tree.

Radia Perlman

Limitations of Bridges

- Scaling
  - Spanning tree algorithm doesn't scale
  - Broadcast does not scale
  - No way to route around congested links, *even if path exists*
- May violate assumptions
  - Could confuse some applications that assume single segment
    - Much more likely to drop packets
    - Makes latency between nodes non-uniform
  - Beware of transparency

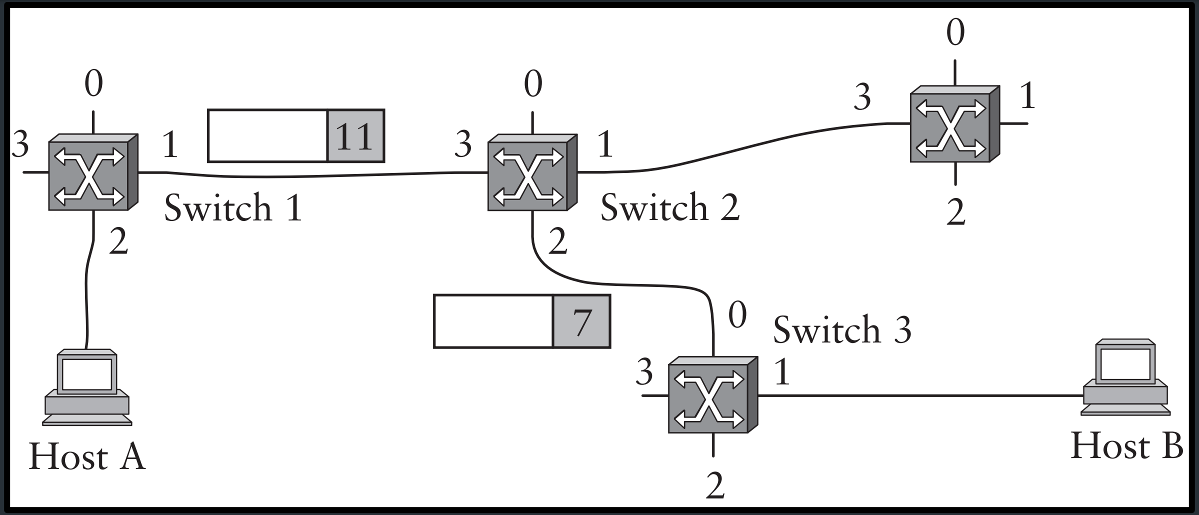
Switching



Switches must be able to, given a packet, determine the outgoing port

- 3 ways to do this:
  - Virtual Circuit Switching
  - Datagram Switching
  - Source Routing

Virtual Circuit Switching



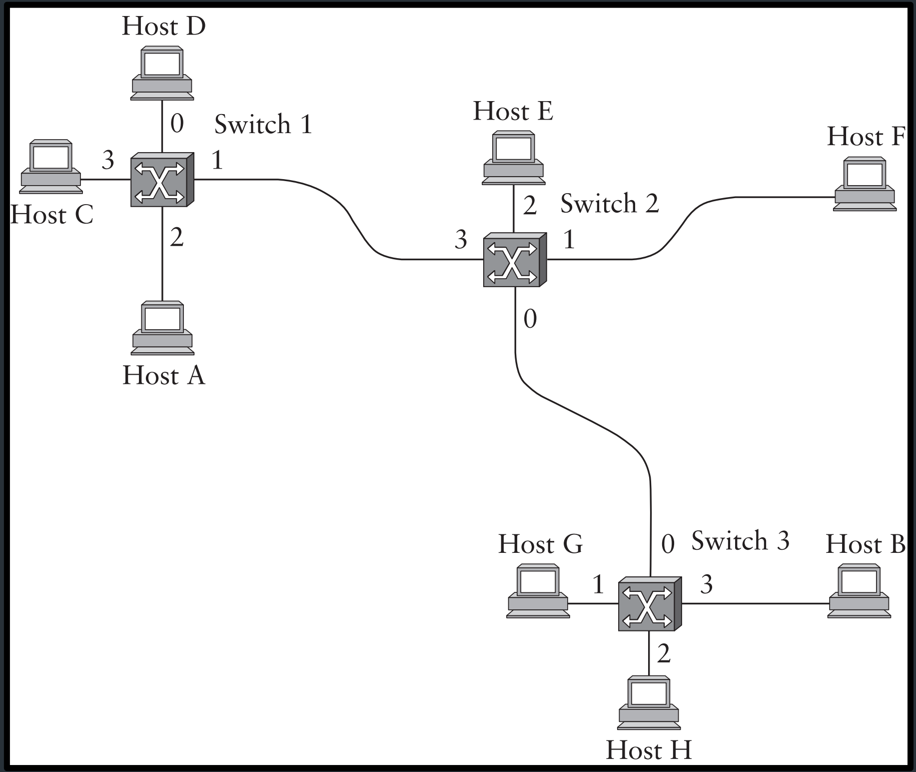
- Explicit set-up and tear down phases
  - Establishes Virtual Circuit Identifier on each link
  - Each switch stores VC table
- Subsequent packets follow same path
  - Switches map [in-port, in-VCI] : [out-port, out-VCI]
- Also called *connection-oriented* model

Virtual Circuit Model

- Requires one RTT before sending first packet
- Connection request contain full destination address, subsequent packets only small VCI
- Setup phase allows reservation of resources, such as bandwidth or buffer-space
  - Any problems here?
- If a link or switch fails, must re-establish whole circuit
- Example: ATM, MPLS

Datagram Switching

- Each packet carries destination address
- Switches maintain address-based tables
  - Maps [destination address]:[out-port]
- Also called *connectionless* model



Switch 2

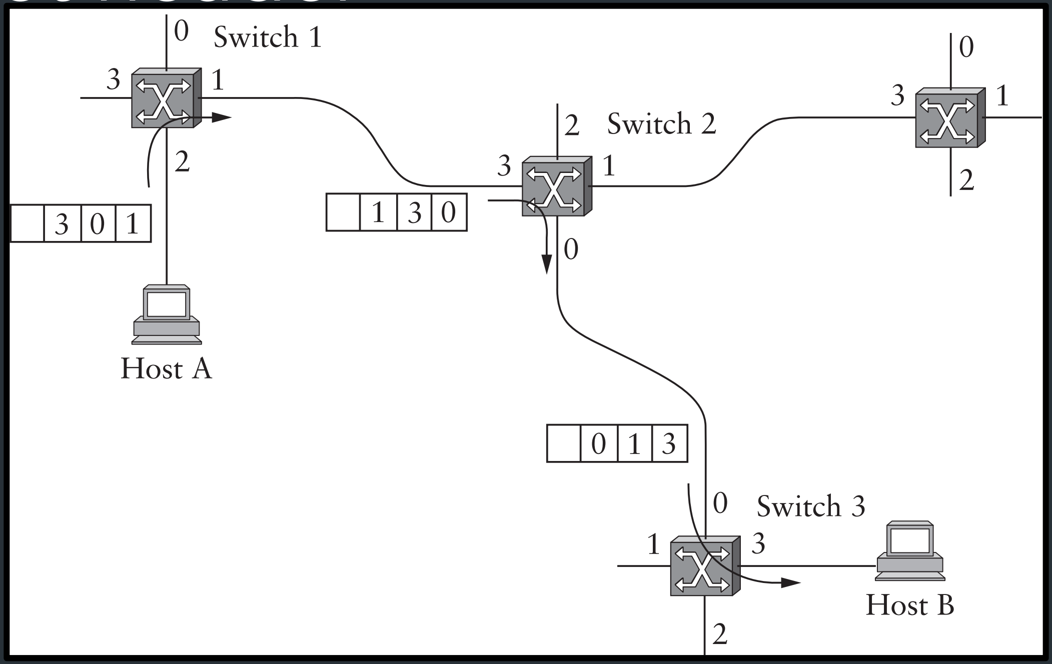
Addr	Port
A	3
B	0
C	3
D	3
E	2
F	1
G	0
H	0

Datagram Switching

- No delay for connection setup
- Source can’t know if network can deliver a packet
- Possible to route around failures
- Higher overhead per-packet
- Potentially larger tables at switches

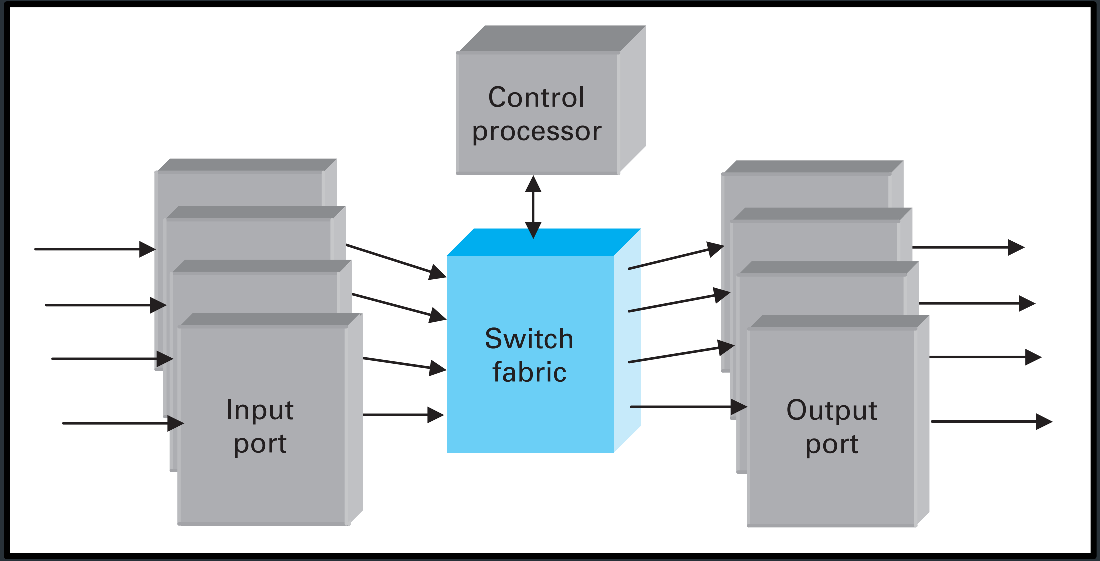
Source Routing

- Packets carry entire route: ports
- Switches need no tables!
  - But end hosts must obtain the path information
- Variable packet header



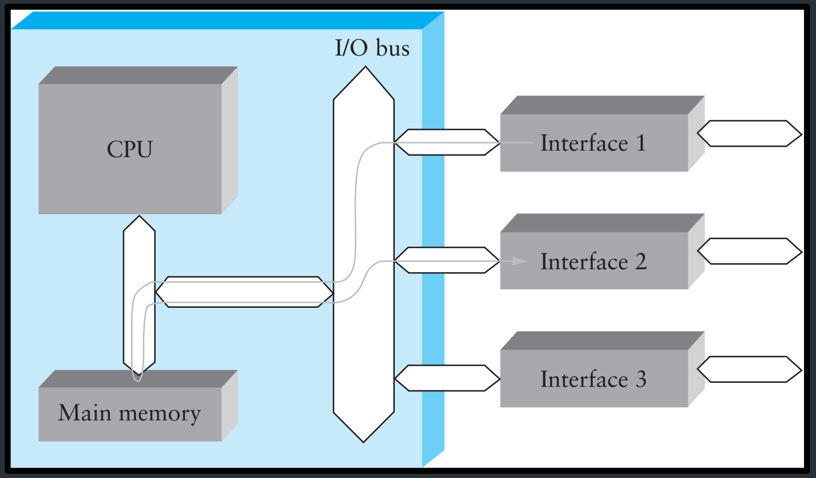
Generic Switch Architecture

- Goal: deliver packets from input to output ports
- Three potential performance concerns:
  - Throughput in bytes/second
  - Throughput in packets/second
  - Latency



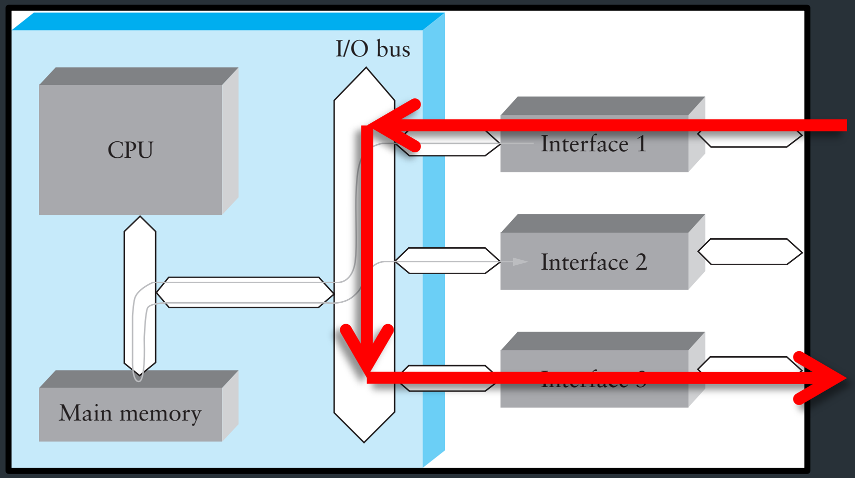
Shared Memory Switch

- 1^{st} Generation – like a regular PC
  - NIC DMAs packet to memory over I/O bus
  - CPU examines header, sends to destination NIC
  - I/O bus is serious bottleneck
  - For small packets, CPU may be limited too
  - Typically < 0.5 Gbps



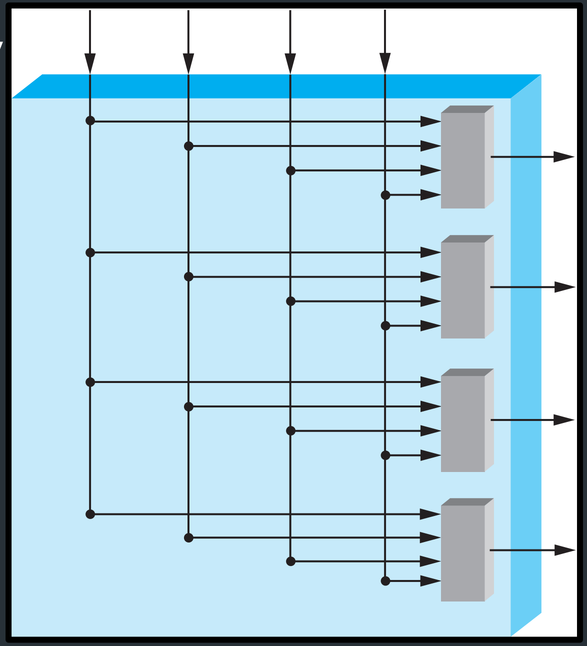
Shared Bus Switch

- 2st Generation
  - NIC has own processor, cache of forwarding table
  - Shared bus, doesn't have to go to main memory
  - Typically limited to bus bandwidth
    - (Cisco 5600 has a 32Gbps bus)



Point to Point Switch

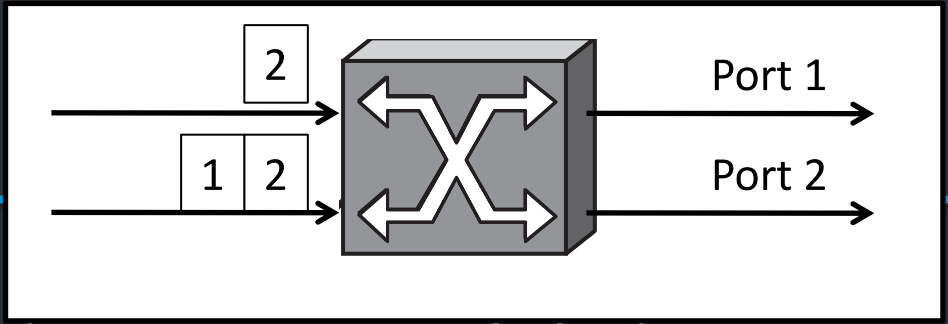
- 3rd Generation: overcomes single-bus bottleneck
- Example: Cross-bar switch
  - Any input-output permutation
  - Multiple inputs to same output requires trickery
  - Cisco 12000 series: 60Gbps



Cut through vs. Store and Forward

- Two approaches to forwarding a packet
  – Receive a full packet, then send to output port
  – Start retransmitting as soon as you know output port, before full packet
- Cut-through routing can greatly decrease latency
- Disadvantage
  – Can waste transmission (classic *optimistic* approach)
    - CRC may be bad
    - If Ethernet collision, may have to send runt packet on output link

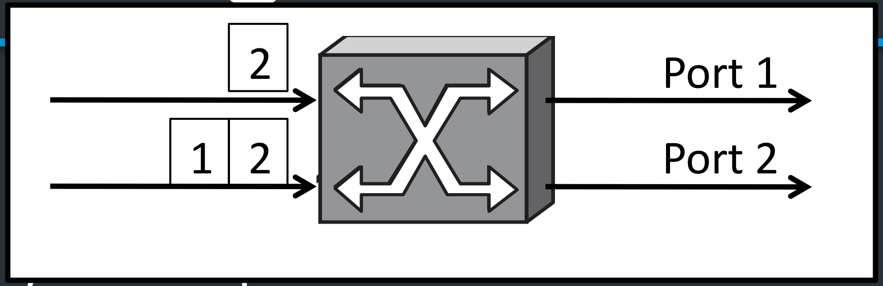
Buffering



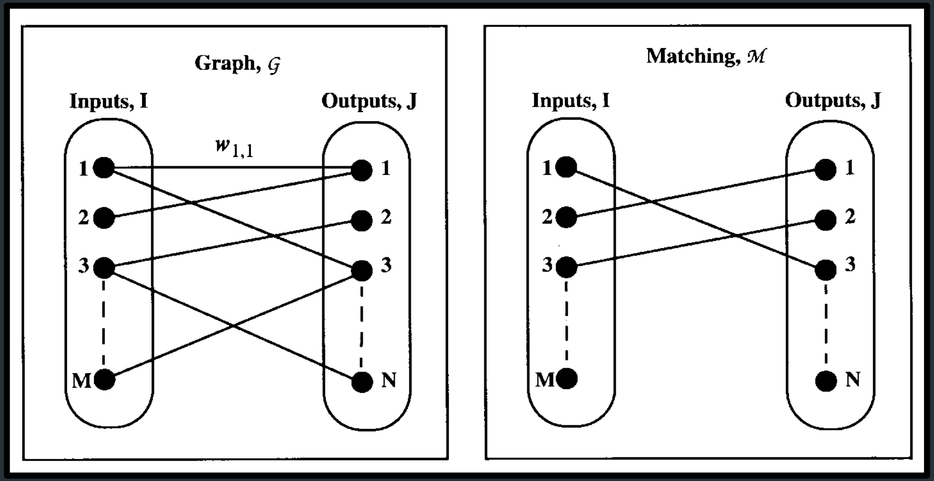
- Buffering of packets can happen at input ports, fabric, and/or output ports
- Queuing discipline is very important
- Consider FIFO + input port buffering
  - Only one packet per output port at any time
  - If multiple packets arrive for port 2, they may block packets to other ports that are free
  - *Head-of-line blocking:* can limit throughput to ~ 58% under some reasonable conditions*

* For independent, uniform traffic, with same-size frames

Head-of-Line Blocking



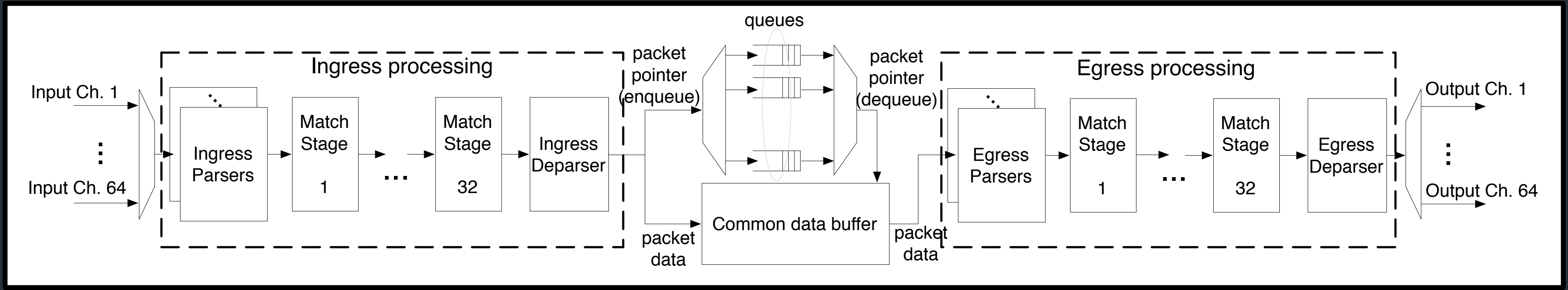
- Solution: Virtual Output Queueing
  - Each input port has *n* FIFO queues, one for each output
  - Switch using matching in a bipartite graph
  - Shown to achieve 100% throughput*



*MCKEOWN *et al.*: ACHIEVING 100% THROUGHPUT IN AN INPUT-QUEUED SWITCH, 1999

Current Developments

- Switches are becoming programmable
  - Match-action paradigm
  - Custom protocols, encapsulation, metering, monitoring



- Current speeds reach 12.8Tbps (32x400Gbps or 256x50Gbps) on a single programmable switching chip