CSCI-1680
Building Links and (Local) Networks

TODAY
- CHANNELS
- LATENCY
- LINK LAYER + SWITCHING

SNOWCAST DUE
TUES 2/10
LOOK FOR FEEDBACK
+ GRADESCOPE SOON

IP PARTNER FORM
OUT ON TUESDAY
DUE THURS

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

Administrivia

- Snowcast due Tuesday (2/10) by 11:59pm EST

- Look for announcement on Gradescope/testing soon
  - See our FAQ post for testing resources & common issues!

Next week: we start IP

Tuesday, 2/10
- Start of IP lectures
- Form for selecting your team
- Pre-release IP handout available

Thursday, 2/12
- Team form due
- IP project official release
- Gearup 5-7pm, CIT 165

Today

Last time: how to send over a link

Today: how to build *small* network?
- Link mechanics and sharing
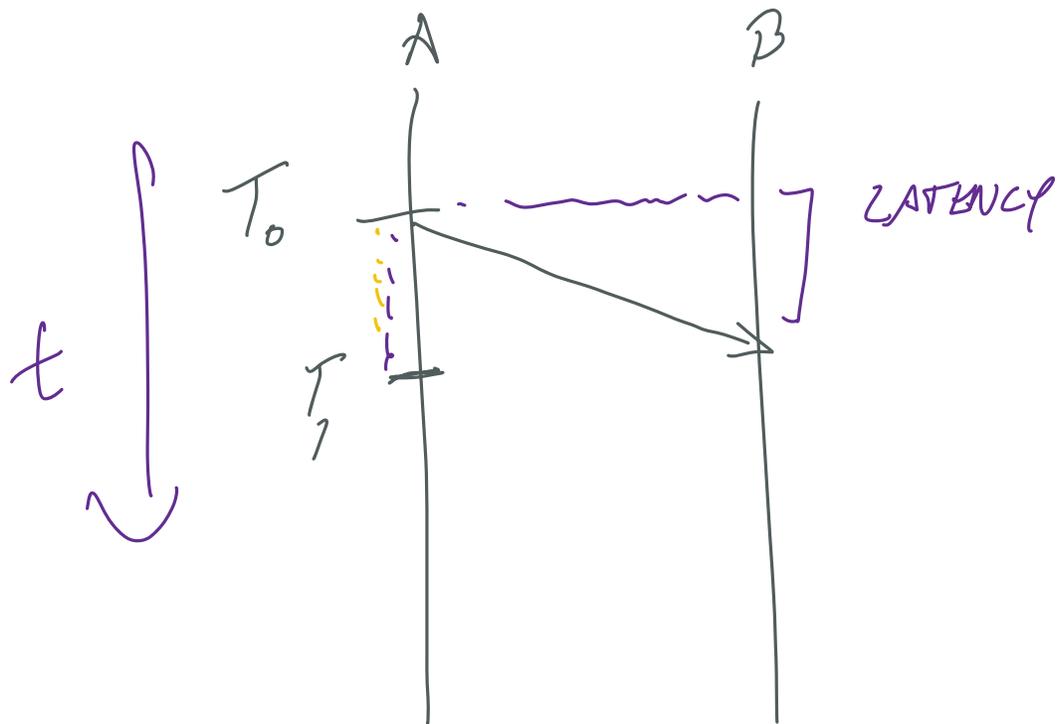- Case study/fundamental terms: Ethernet (and Wifi)
- How switching works

Fun systems facts: channels

Key physical layer properties

Bandwidth: range of frequencies used by a channel (Hz)

Throughput: amount of data delivered per time (bits/sec)

Latency: time between sending data and when it arrives

A B
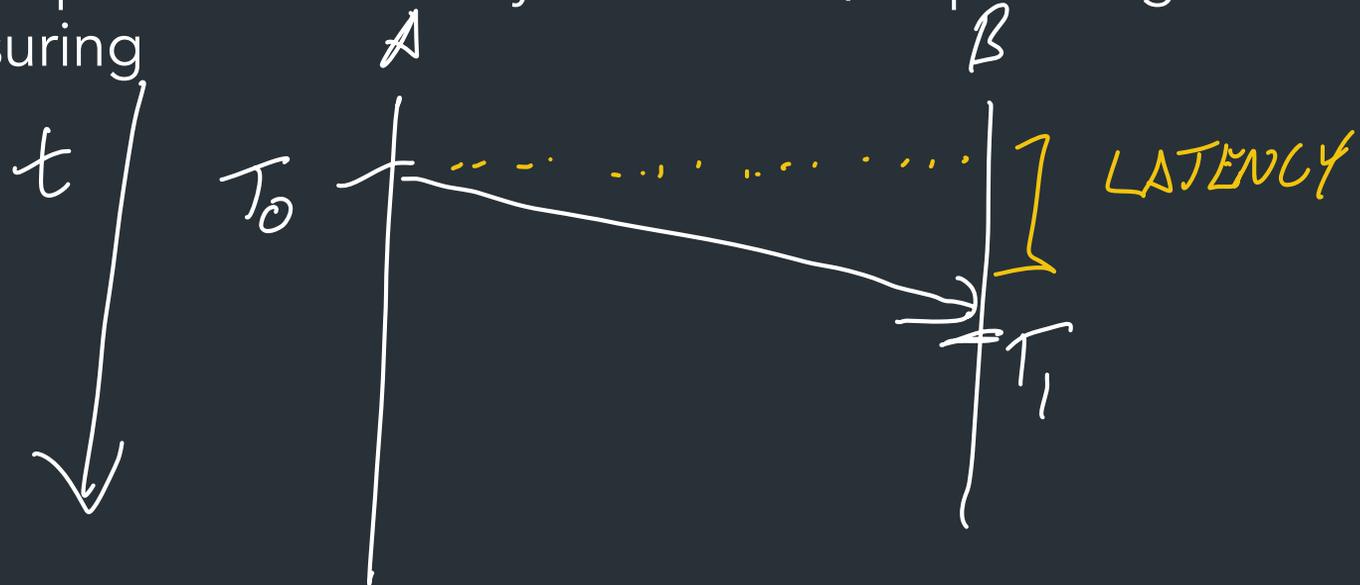
T_0

t

T_1

LATENCY

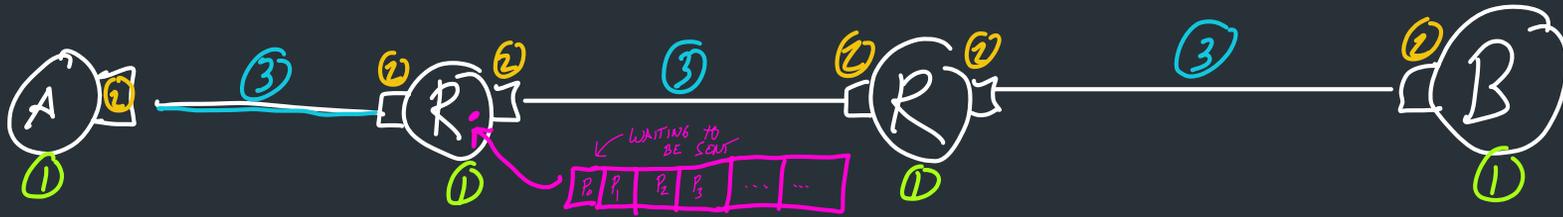# Sending data takes time!

- Latency: time between sending data and when data arrives (somewhere)

  Sending data is not instantaneous (for many reasons)

- Multiple components => many definitions, depending on what we're measuring

How to think about latency

1. Processing delay: Time for data to be sent by a node
   => Time for OS to decide to send data

2. Transmission delay: Time for transceiver to actually send bits on wire

3. Propagation delay: Time for data to propagate across wire => speed of light

4. Queuing delay: time data spends in buffers waiting to be sent (due to congestion, etc.)
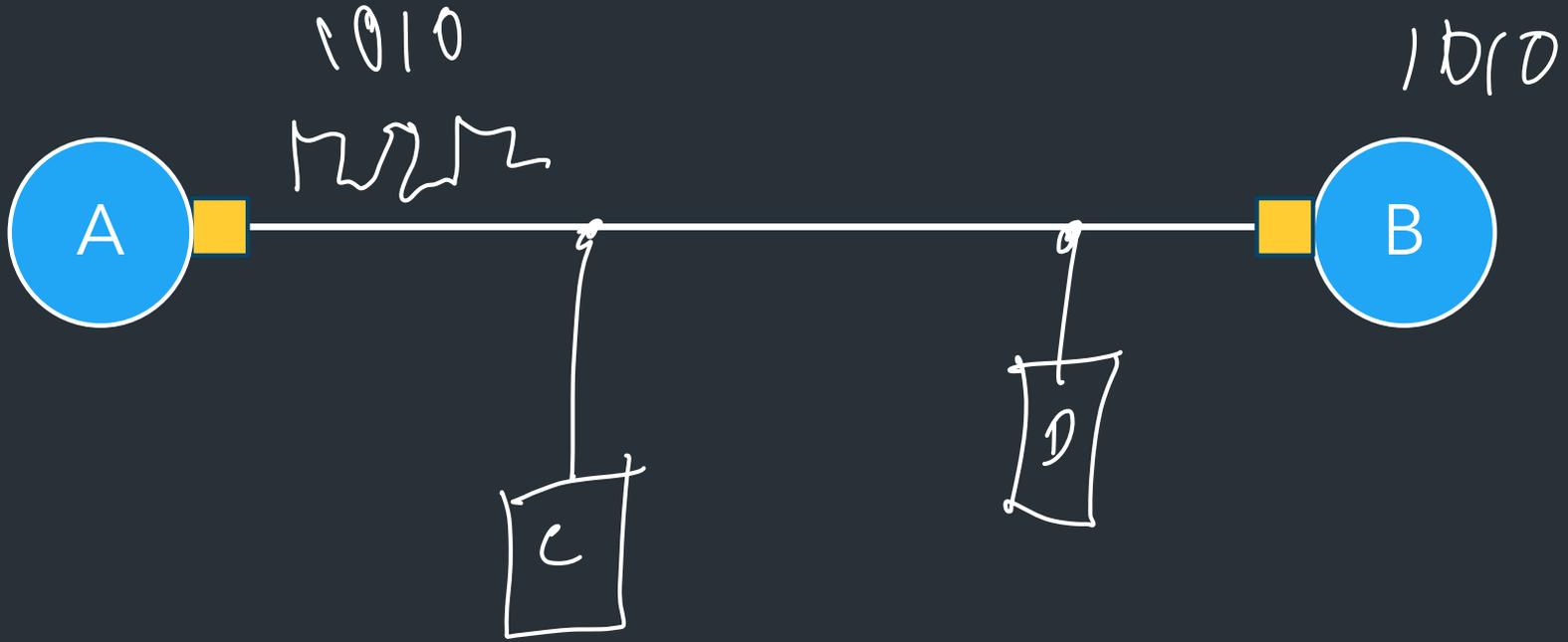
Round trip time (RTT): time between request and response

When we design protocols,
can think about performance
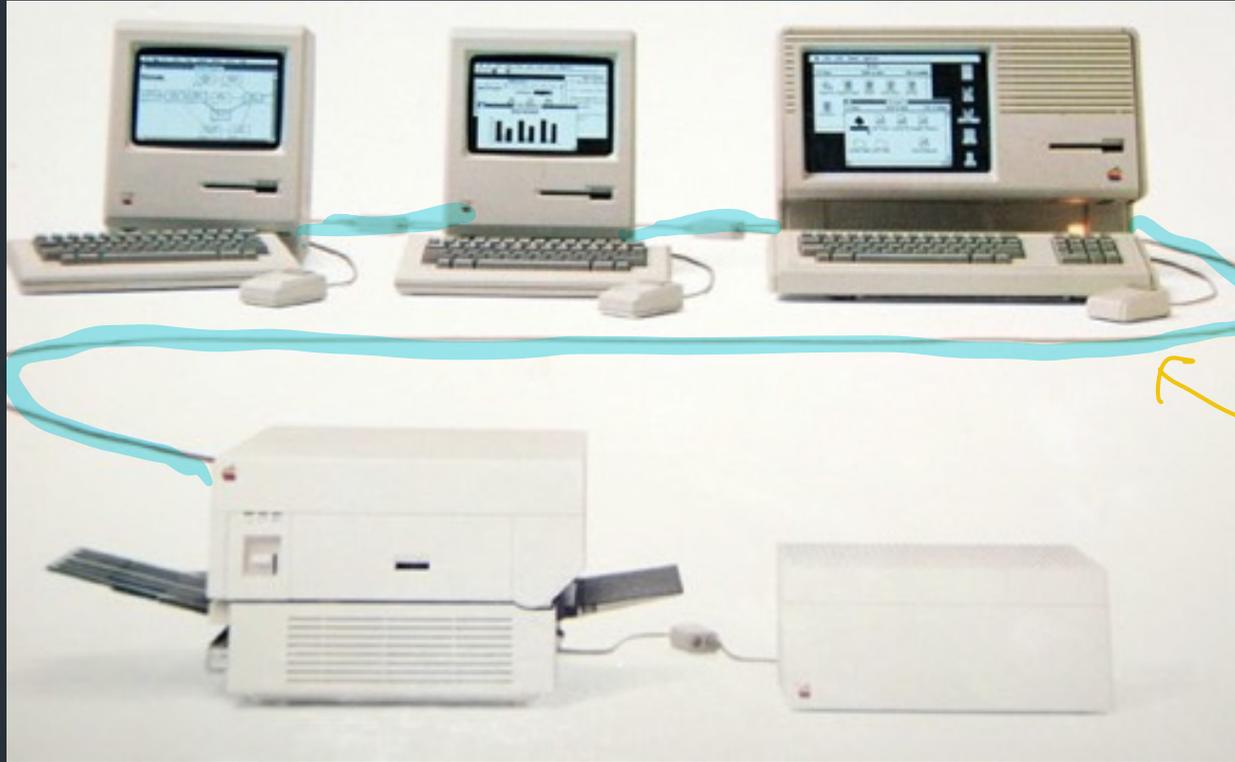based on number of RTTs
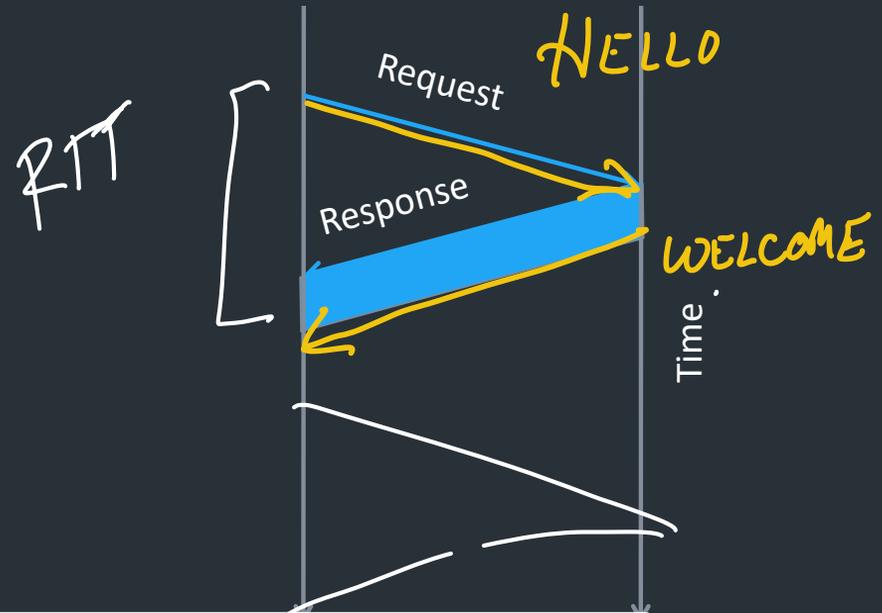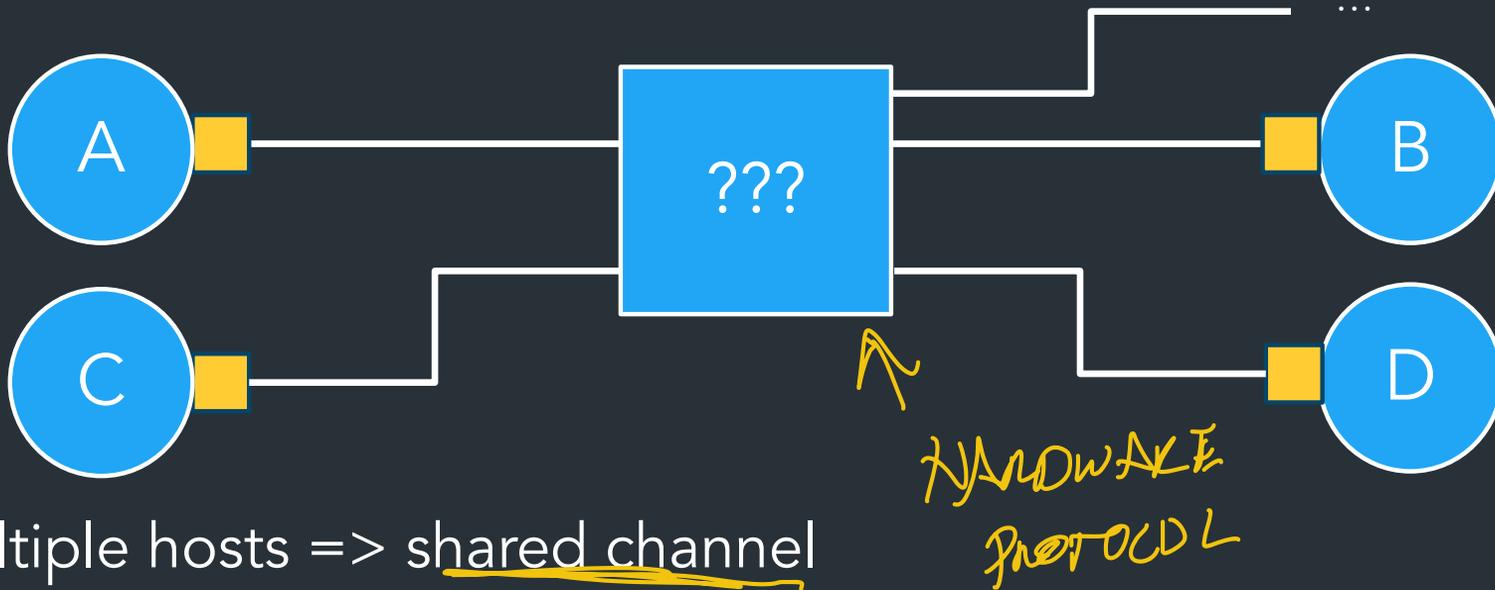
Link layer

The main idea

Setting the scene

An [AppleTalk](#) network (1980s)

ONE SHARED WIRE!

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

What does "link layer" mean?

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

How to share the channel?

=> **Media access control: mechanism/protocol to share the channel**
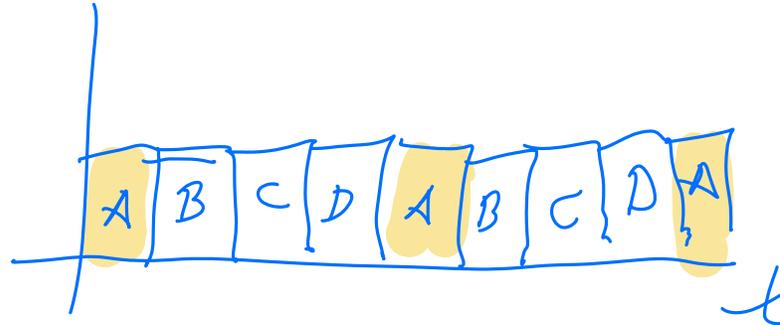
MAC strategies (the gist)

Idea: no more than one device can be "Talking" at a time
Protocol: "who can talk when?"

Two main types

1. **Partitioned access**: divide up the channel into fixed "slots"
 eg. Time slices, small ranges of frequencies, ...

=> Fair to all hosts, but not efficient,
doesn't have high utilization unless all
hosts are using channel

2. Random access
=> No fixed slots, but some way to mediate multiple hosts talking

 - 'Listen' for when other devices are talking, and wait for channel to be free
 - "request to send" => send a tiny signal to ask to use the channel (usually a "leader")
 - "Token-based"

 => Can achieve higher utilization, but not necessarily fair to all hosts (some ways to mitigate this)

High-level: MAC approaches *(Some examples)*

Partitioned Access: divide the channel into fixed slots

- Time Division Multiple Access (TDMA)
- Frequency Division Multiple Access (FDMA)
- …

Problems?

$\Rightarrow$ Hard to maximize channel *utilization*
(eg. what happens if only one person is talking?)

High-level: MAC approaches

(Some examples)

<u>Random Access</u>: 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?

$\Rightarrow$ Hard to maintain "fairness"
(eg. one host dominating channel)

Why does this matter?

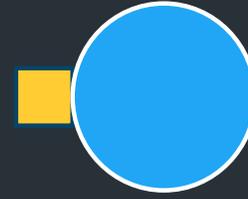Different types of links solve these problems differently
- – Ethernet (wired) vs. Wifi (wireless)
- – Affects throughput, reliability, etc.

Understand why different links operate differently
=> How we build the Internet from them
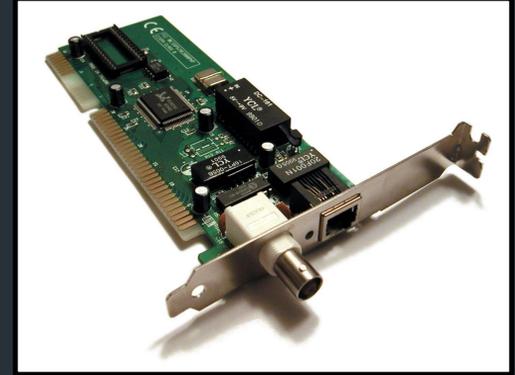
How does a device use a link?

Interface: device that connects something to the network

=> OS abstraction for a link
=> Hardware in the interface actually "talks" on the channel
=> Network Interface Card (NIC)

Examples:
- Loopback interface (lo): Virtual interface, for localhost
- Physical: Wifi, ethernet, bluetooth, etc.

Network

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

Search
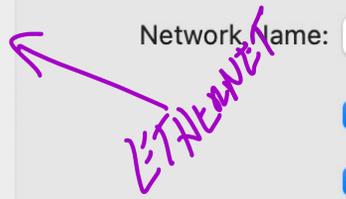
Status: **Connected**   Turn Wi-Fi Off

Wi-Fi is connected to RLAB and has the IP
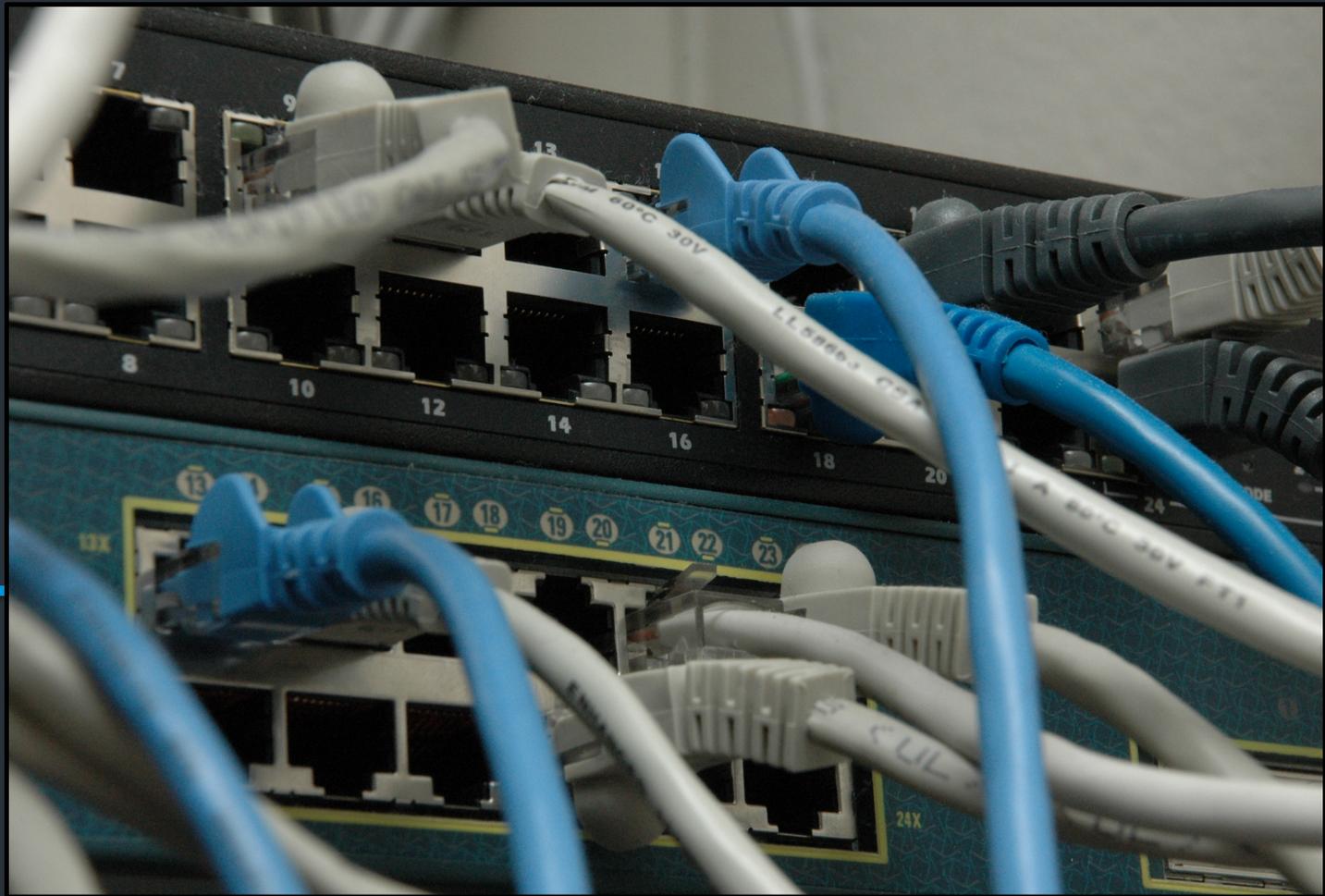address 138.16.161.155.

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.

ETHERNET

SOFTWARE

(LOOPBACK)

☑ Show Wi-Fi status in menu bar   Advanced...   ?
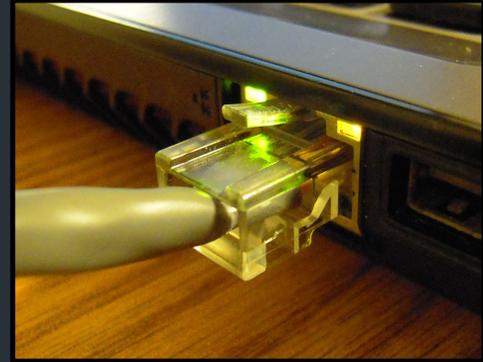
Revert   Apply

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: Historical version

"BUS"

A   B   C   D
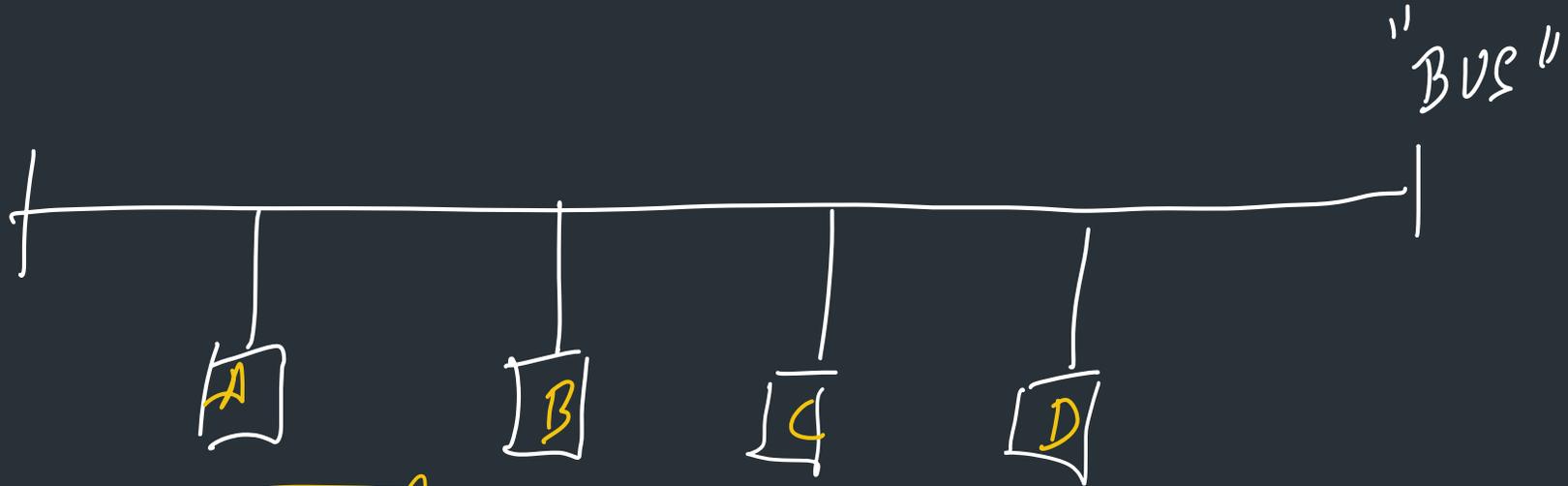
Every host has an address => Every host is connected to every other host"
When you want to send, send packet on bus
Every host can see signals on the wire PROBLEM? YUP.
If you see a packet for your address, process packet, otherwise ignore

# Ethernet: software viewpoint

- <u>Logically</u> all hosts are connected to each other

- All hosts have an "ethernet address" ("mac address")
        => Globally-unique identifier

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

This page and the next page talk about the Ethernet packet format. We didn't talk about this much in class, but we'll discuss it further and see more examples later!
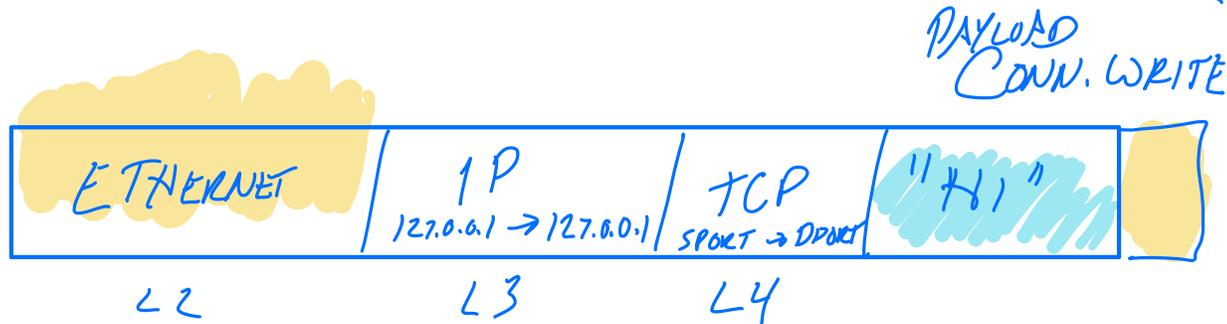
Preamble	SFD	Destination MAC Address	Source MAC Address	EtherType	Payload		FCS
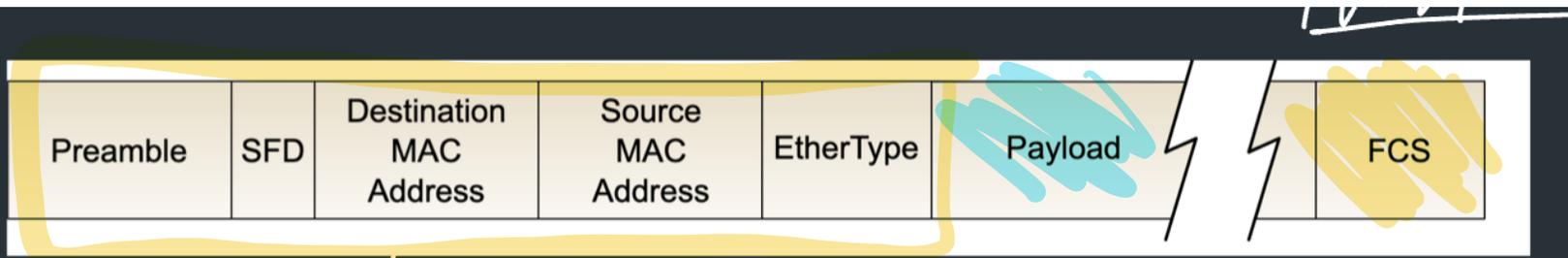
Destination address: where packet is going
Source address: where packet is coming from

EtherType: type of data that's in the payload

Payload: rest of the packet
FCS (Frame Check Sequence): a checksum

("HI")

PAYLOAD
CONN. WRITE

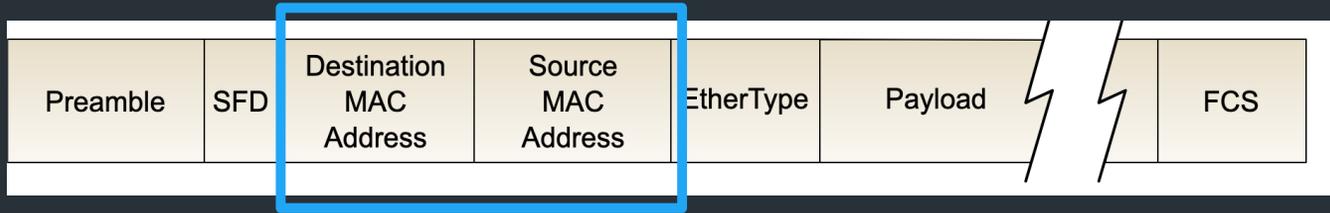| ETHERNET	IP 127.0.0.1 → 127.0.0.1	TCP SPORT → DPORT	"HI"	
L2	L3	L4		

What's a checksum? Helps check for errors
 - Sender compute small value, like hash of packet => send this with the packet
 - Receiver hashes packet on arrival. If receiver's hash != hash in packet, message was corrupted in transit

Ethernet Addresses (mac addresses)

Globally unique, 48-bit address per interface
00:1c:43:00:3d:09 (Samsung adapter)

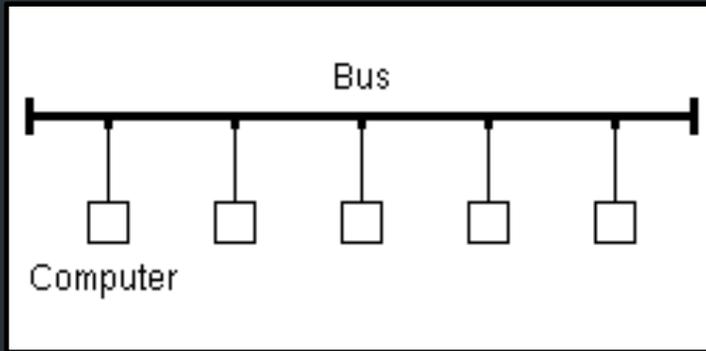First 24 bits: Registered to manufacturers

=> 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?

RANDOM ACCESS

=>Can have collisons!

Classical Ethernet: Problems

Problem: all hosts in the same "collision domain"

=> All hosts could cause collisions with all others

Transmit algorithm

- If line is idle, transmit immediately

- Max message size: 1500 bytes

- If line is busy: wait until idle and transmit immediately

> **"Wait until idle" =>** wait for some time t, if line is busy after t,
> wait 2*t, keep increasing exponentially until time out
> **=> Exponential backoff**

Does this scale?

Ethernet: the header

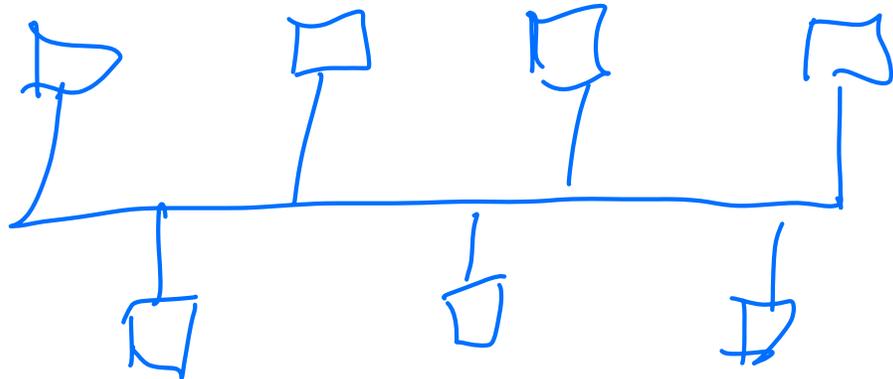| Preamble	SFD	Destination MAC Address	Source MAC Address	EtherType	Payload		FCS
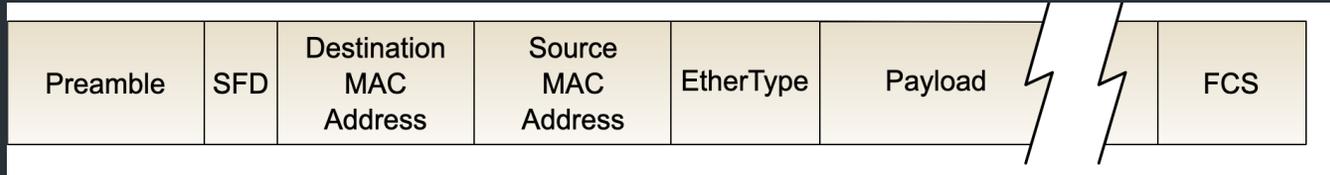
- Source address: where packet is from
- Destination address: where packet is going
  - ⇒ Devices ask: "Is this my packet?" "Where should I send this packet?"

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

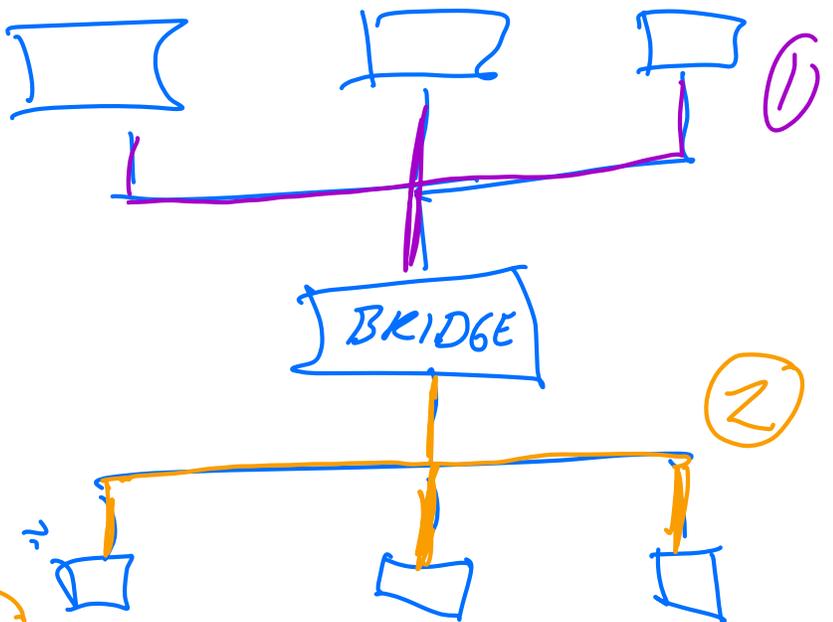EARLY ETHERNET:

ALL HOSTS ON SAME COLLISION DOMAIN

- DOESN'T SCALE

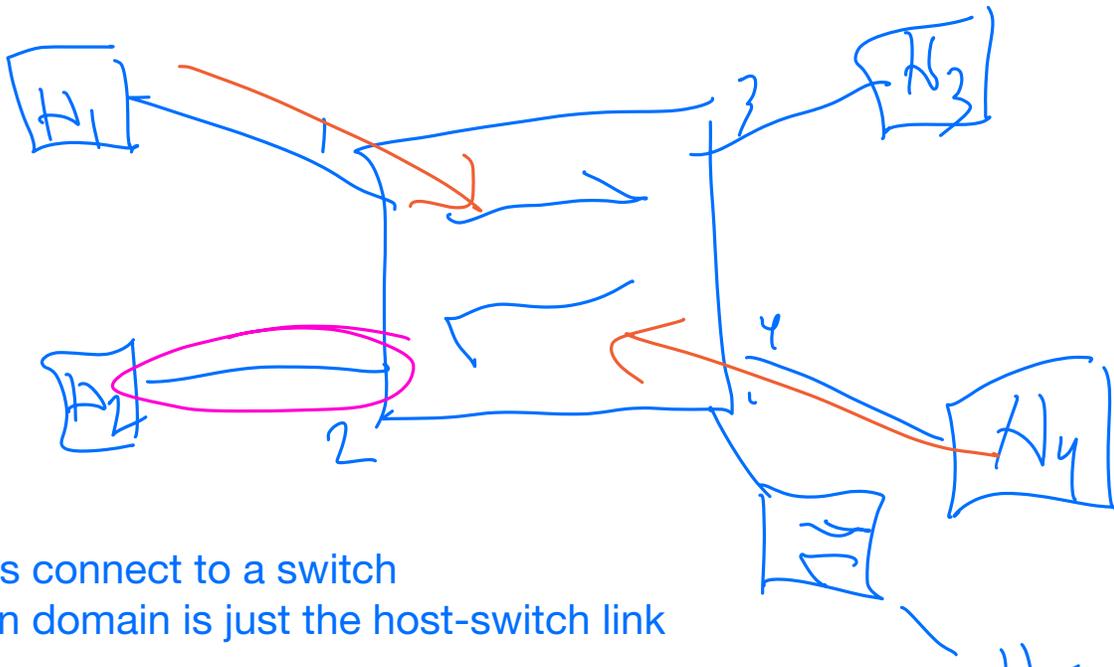- WANT ABILITY TO SEPARATE COLLISION DOMAIN WHILE STILL BEING ABLE TO TALK TO EVERYONE.

EARLY WAY: BRIDGES
 - CONNECT TWO NETWORKS
 - ONLY FORWARD BETWEEN ①⇔② IF NECESSARY

HERE: ONE LAN, 2 COLLISION DOMAINS ① ②

BRIDGE

Modern way: an Ethernet switch

H1   N3
H2   H4
H5

All hosts connect to a switch
Collision domain is just the host-switch link

 => In modern times, both sides of one link can
transmit/receive at the same time (full duplex)
 => Really hard to have a collision

Switch receives frames, looks at destination
address
 - Decides which port to send packet to
 - Queue up packets until the destination port is
idle
  => Lots of engineering involved in how to design
a switch to do this quickly, queue efficiently

DEST	PORT
H1	1
H4	4

| SRC MAC | DST MAC | PAYLOAD | IP HDR | ..-~ |
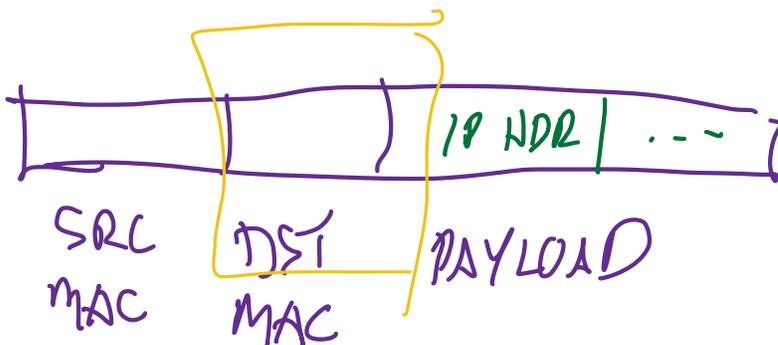
MAC learning: as packets are sent,
switch builds a table of mac
addresses it has seen before,
remembers the port used

(More info in next couple of pages)

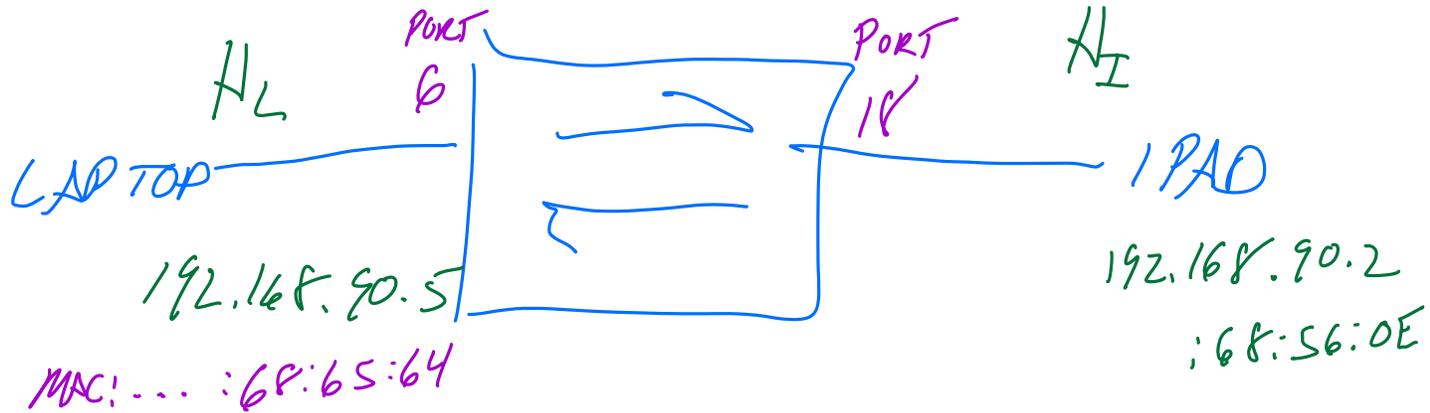# MAC table example

```
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
```

(handwritten annotations: "PORTS ON SWITCH" pointing to Destination Port column, "MAC" pointing to Destination Address column)

Example network: Nick's lecturing setup (with a switch!)

H_L — PORT 6 ——[switch]—— PORT 18 — H_I

LAPTOP

192.168.90.5

MAC: ... :68:65:64

IPAD

192.168.90.2

:68:56:0E

MAC	PORT
LAPTOP	6
IPAD	18

What the switch does:

Switch "learns" which host is on which port

Switch has table: <mac address, port>

Fills the table based on packets that it sees
 => Source address on packet tells it what to put in table

When switch doesn't know what to do, it sends packets to all ports ("flood")

If there's an entry in the table, switch can send packet to only the port associated with that host

Extra content and additional notes after this point

What if you WANT to talk to multiple hosts?

This is still possible! Separating collision domains means we
don't need to do this unless necessary.

There are special Ethernet addresses to reach multiple hosts
 - Broadcast address: send to all hosts
 - Multicast: send to a certain group of hosts

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

- Switches "learn" which host lives on which port by watching traffic

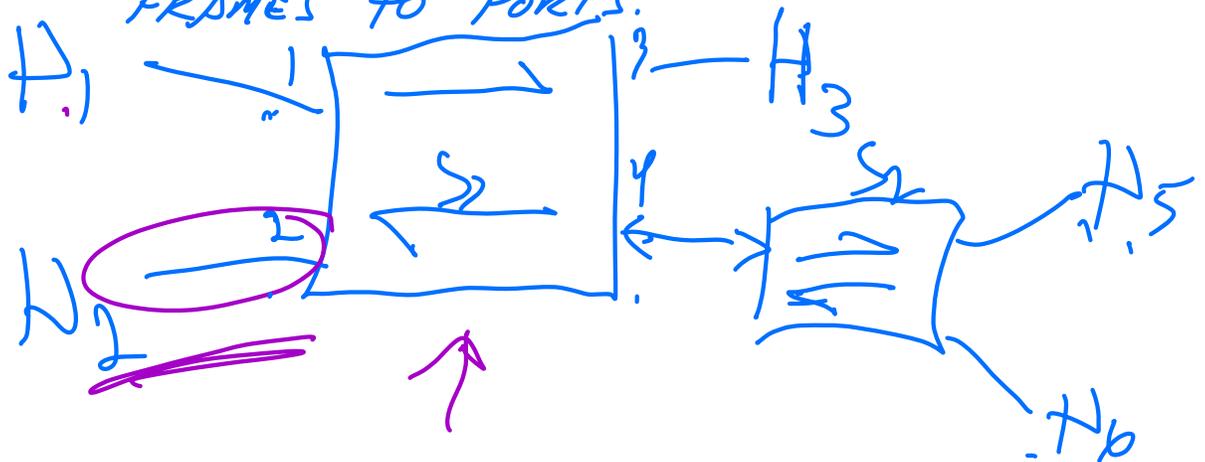- If you don't know, flood to all ports!

SOME SECURITY PROBLEMS
— CAN "SPOOF" (FORGE) ADDRESS
— CAN FILL TABLE.

REALLY JUST A CACHE!

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

MODERN ETHERNET: SWITCHES

SWITCH: NETWORK DEVICE THAT FORWARDS FRAMES TO PORTS.

- ALL HOSTS CONNECT TO A SWITCH

- COLLISION DOMAIN IS JUST THE LINK OF HOST — SWITCH!

  - IN MODERN TIMES, BOTH SIDES OF A LINK CAN TRANSMIT AT SAME TIME.
  - SWITCH CAN STORE FRAMES + ONLY FORWARD TO A PORT WHEN IT'S IDLE.
    ⟹ REALLY HARD TO HAVE A COLLISION

- SWITCH RECEIVES FRAME, LOOKS AT DEST ADDRESS, DECIDES WHICH PORT ON WHICH TO SEND FRAME
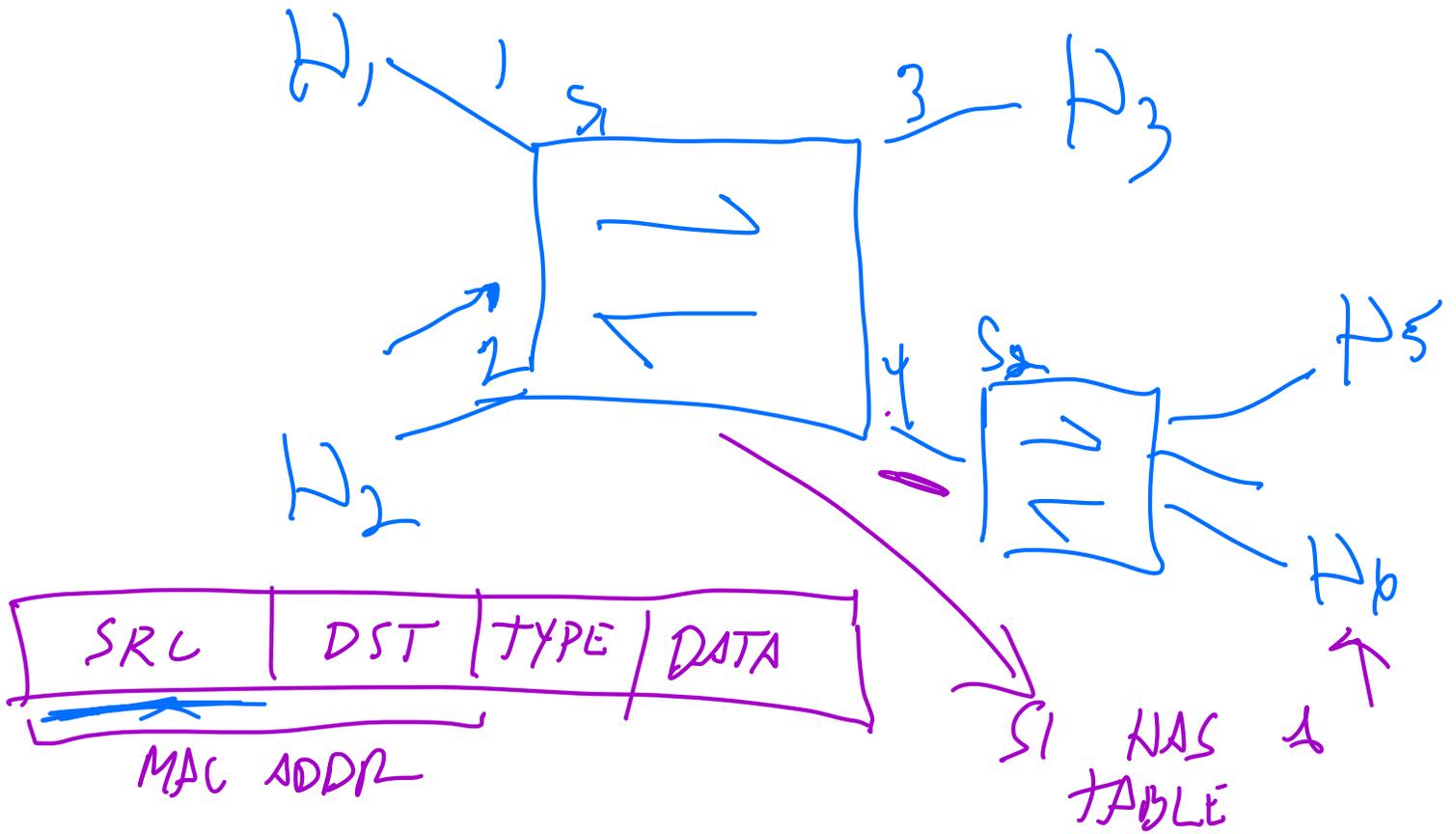
"MAC LEARNING:"
  - SWITCHES "LEARN" WHICH HOST(S) LIVES ON WHICH PORT

  - IF DON'T KNOW, FLOOD TO ALL PORTS!
    ⟹ JUST AN OPTIMIZATION VS. OLD VERSION!

MAC LEARNING: How IT WORKS.

SRC	DST	TYPE	DATA

MAC ADDR

$H_2 \to H_3$ ON PORT 2

IF TABLE IS EMPTY
$\Rightarrow$ FLOOD

$H_5 \to H_2 \Rightarrow$ SEND TO PORT 2

S1 HAS A
TABLE

| MAC ADDR $\to$	PORT
H_2	2
H_5	4
H_6	4

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?
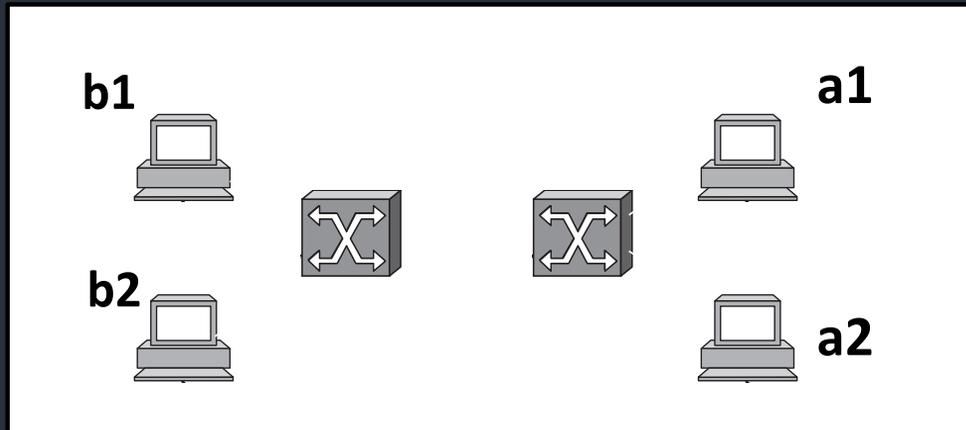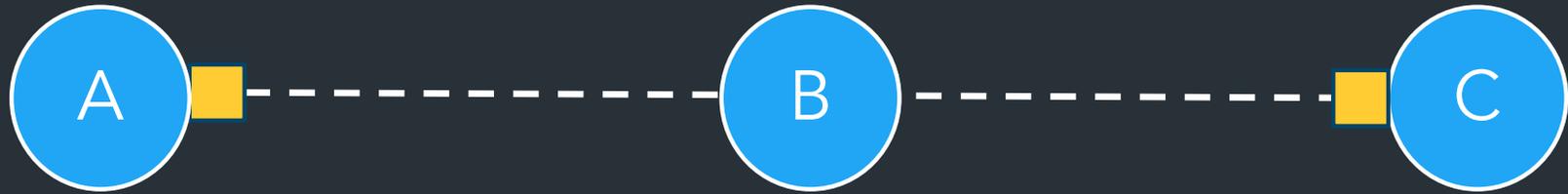
VLANs

- 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
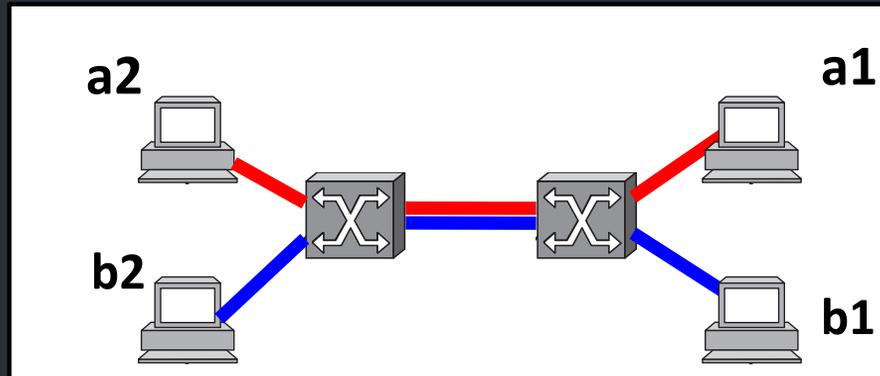
How does this all change with wifi?

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)