15-441/641: Computer Networks
Intradomain Routing

15-441 Fall 2019

Profs Peter Steenkiste & **Justine Sherry**

Carnegie Mellon University

Talk with a friend…

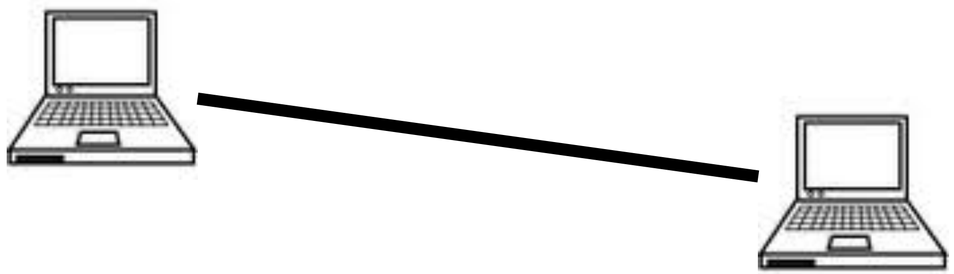
- Leonard has been lost on a desert island and doesn't know much about technology. How would you explain what a "protocol" is to Leonard?
- Name four communication mediums used in computer networks
    - Name one communication medium used by non-computer communication networks
- Identify one TA and when their office hours are.

Project 1

- HANDING OUT NOW!
- TAs will give a tutorial on getting started and how to succeed with the very first checkpoint.
    - Bring your laptop to recitation tomorrow so that you can follow along with the how-to's.
- START. EARLY.

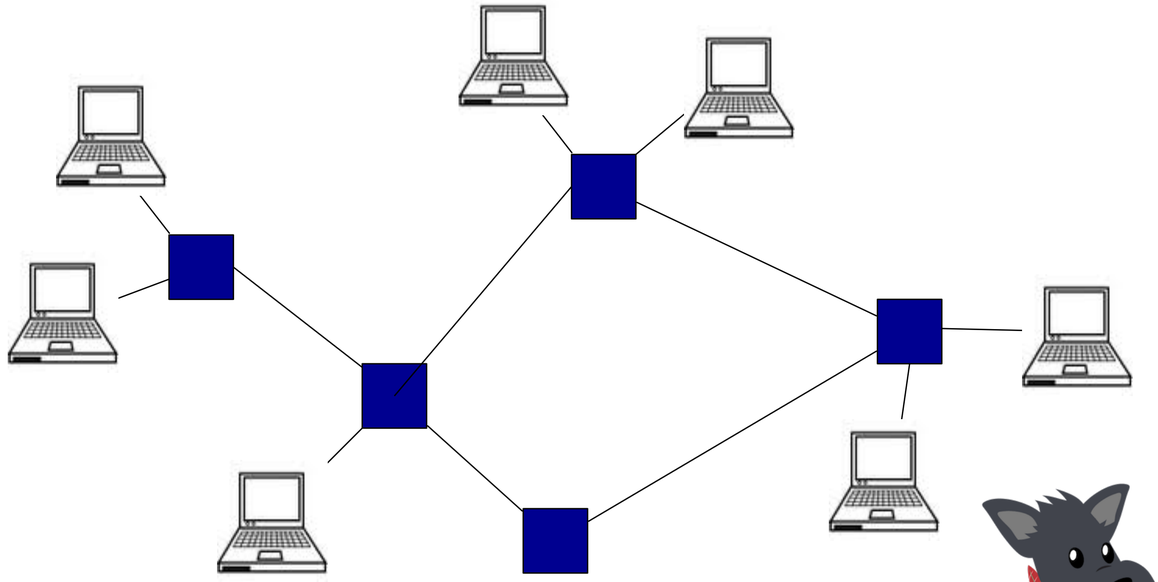
Start Early. Start Early. Start Early.
Start Early. Start Early. Start Early.
Start Early. Start Early. Start Early.
Start Early. Start Early. Start Early.
Start Early. Start Early. Start Early.
Start Early. Start Early. Start Early.
Start Early. Start Early. Start Early.
Start Early. Start Early. Start Early.

How do we go from here to here?



“Point-to-Point Network”

“Local Area Network”



What Prof. Steenkiste said

- “When you have networks with lots of nodes talking to each other, we think it’s better to have packet-switched rather than circuit switched networks.”

- What I didn’t answer:

  - In those networks with lots of nodes talking to each other, how do we identify each node?

  - And how do the messages get routed from one node to another? How does the network *know* where to steer the packets?

Today we will fill in these gaps

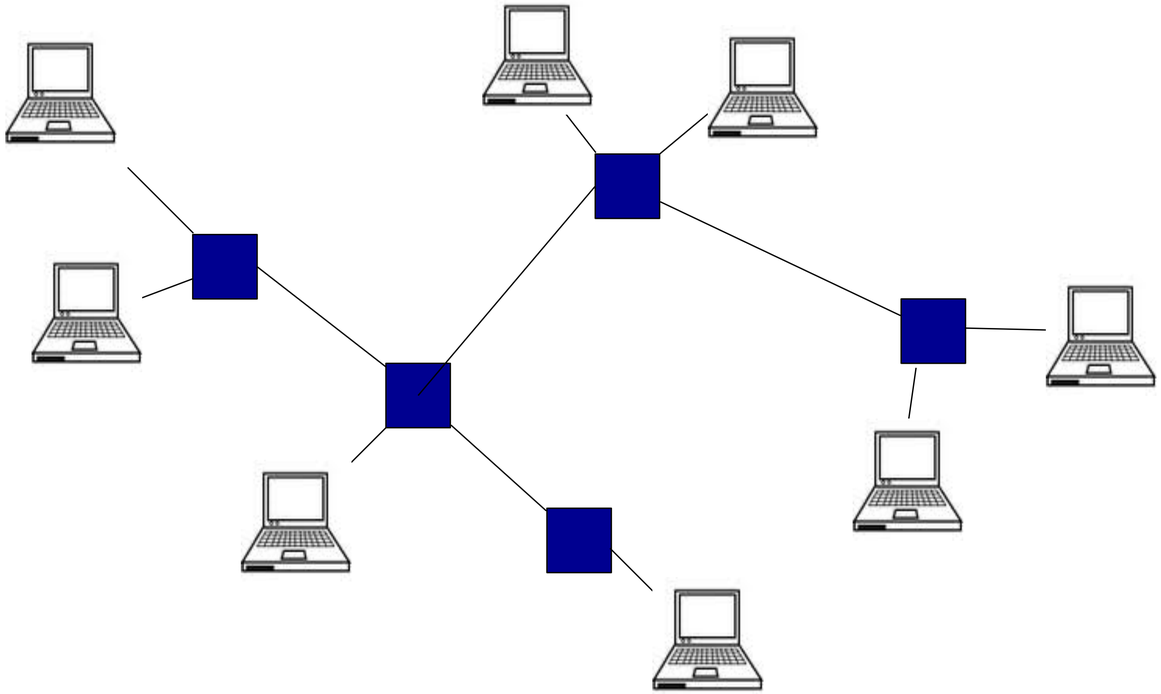
- We will talk about Local Area Networks
  - How do we identify hosts/nodes/servers on Local Area Networks?
  - How does data reach its destination in a Local Area Network?
- We will then talk about Wide Area Networks
  - What is a Wide Area Network???
- We still won't talk about the Internet! Just the insides of one single network.

One Single Network

EVIL CORP

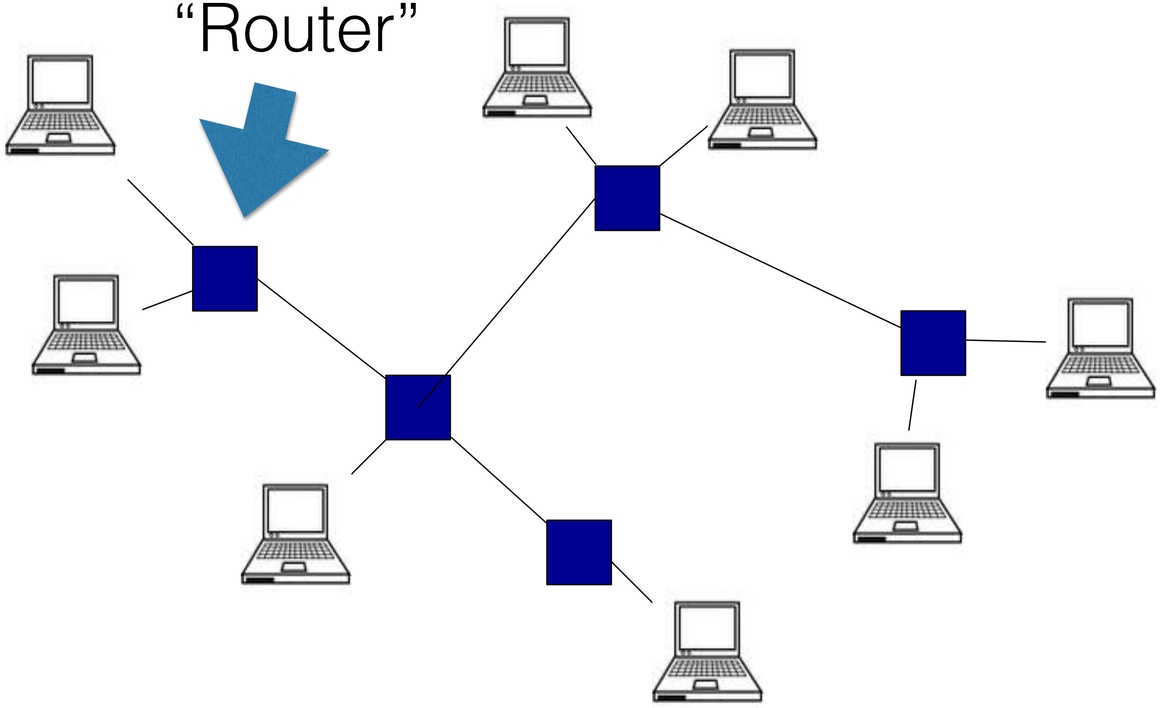
Basic Vocab

“End Host”
“Host”
“Computer”
“Device”



Basic Vocab

“Switch”
“Bridge”
“Router”

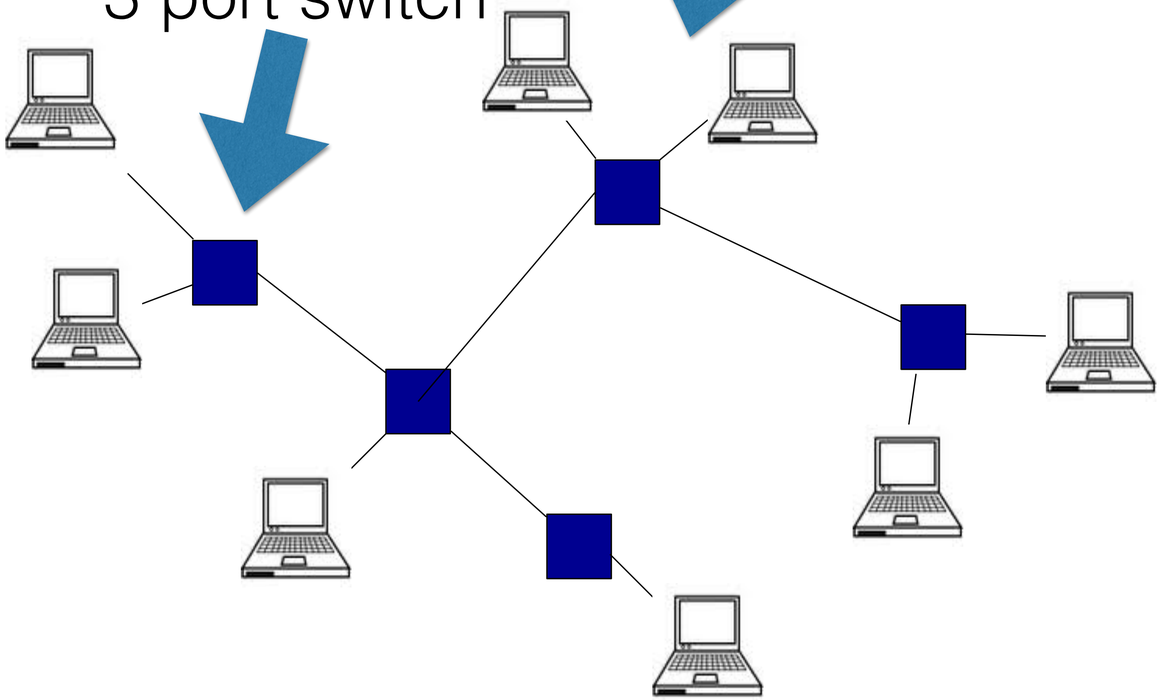


Basic Vocab

5 port switch

4 port switch

3 port switch



So You Want To Build a Local Area Network

Step 1: Choose an *addressing scheme*

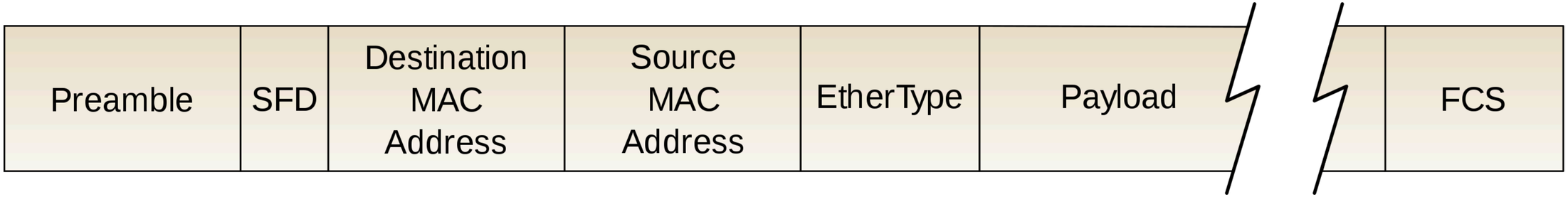
Today, we'll talk about one addressing scheme: MAC addresses.

There are others we'll learn about next week.

MAC Addresses

- “Media Access Control Address”
- 48 bits long, written as a sequence of hexadecimal numbers
  - e.g. 34:f3:e4:ae:66:44
- Quick — how many possible MAC addresses are there?
  - 281,474,976,710,656
- Used as part of a protocol called *Ethernet*.

An Ethernet Packet



Aka “Frame”

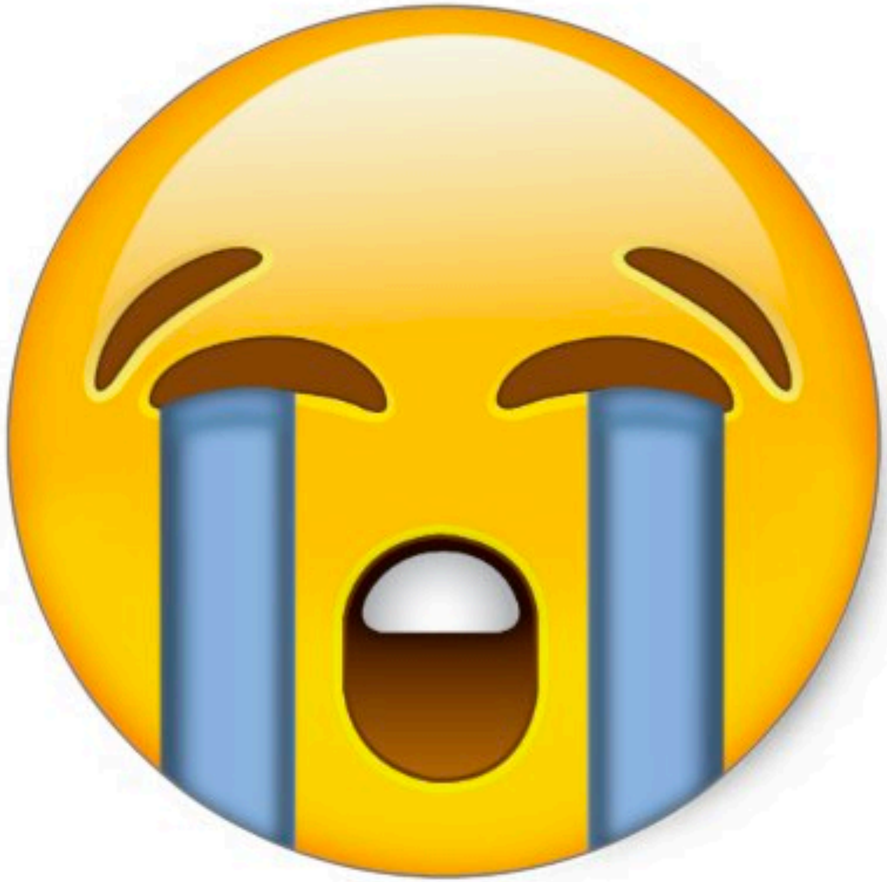
A,aka “Datagram”

NEVER: PACKAGE

Package

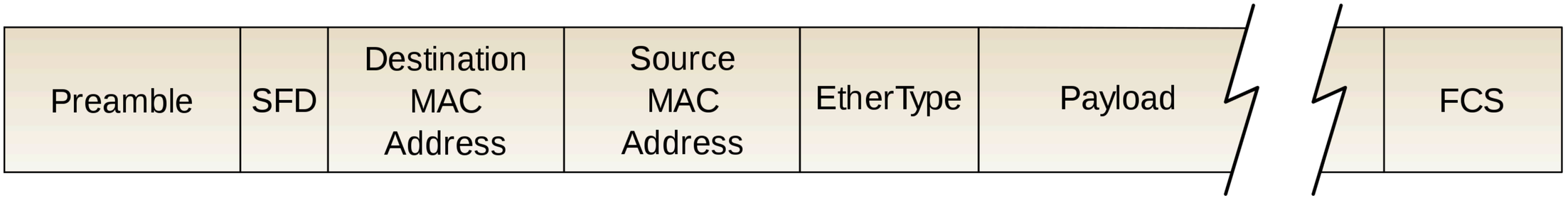
Packages are sent through the mail.

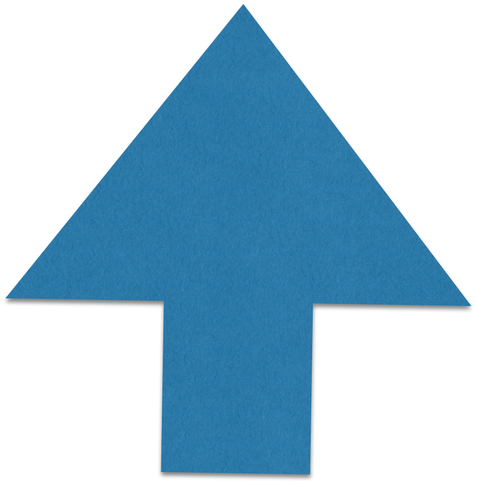
<u>“Packets” are what we send over networks</u> and it makes me cry when students switch these two words up.



me irl

Back to Ethernet

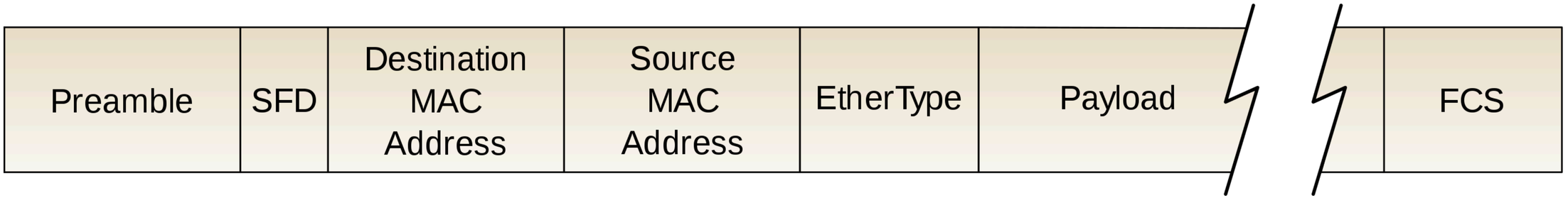


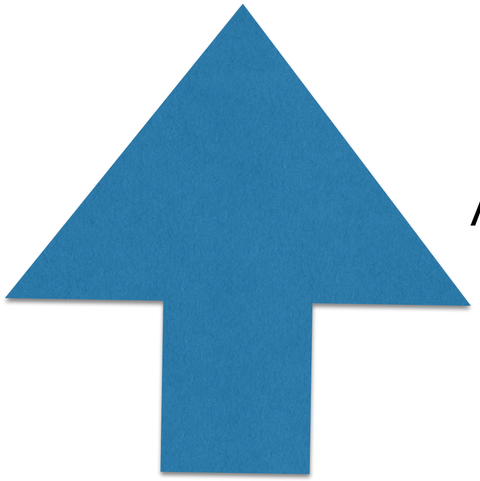


Preamble:
Always 10101010101 repeating, 56 times
SFD:
10101011

WHY???
Ask your friend.

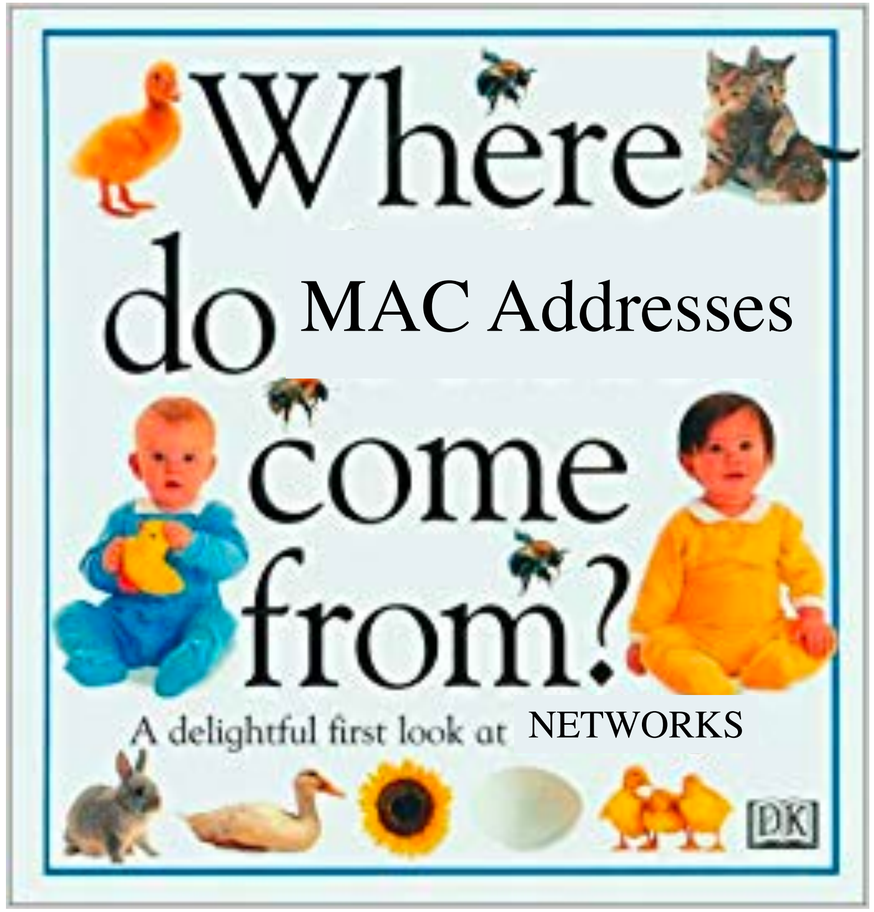
Back to Ethernet





Address of the host to send the packet to

MAC Addresses

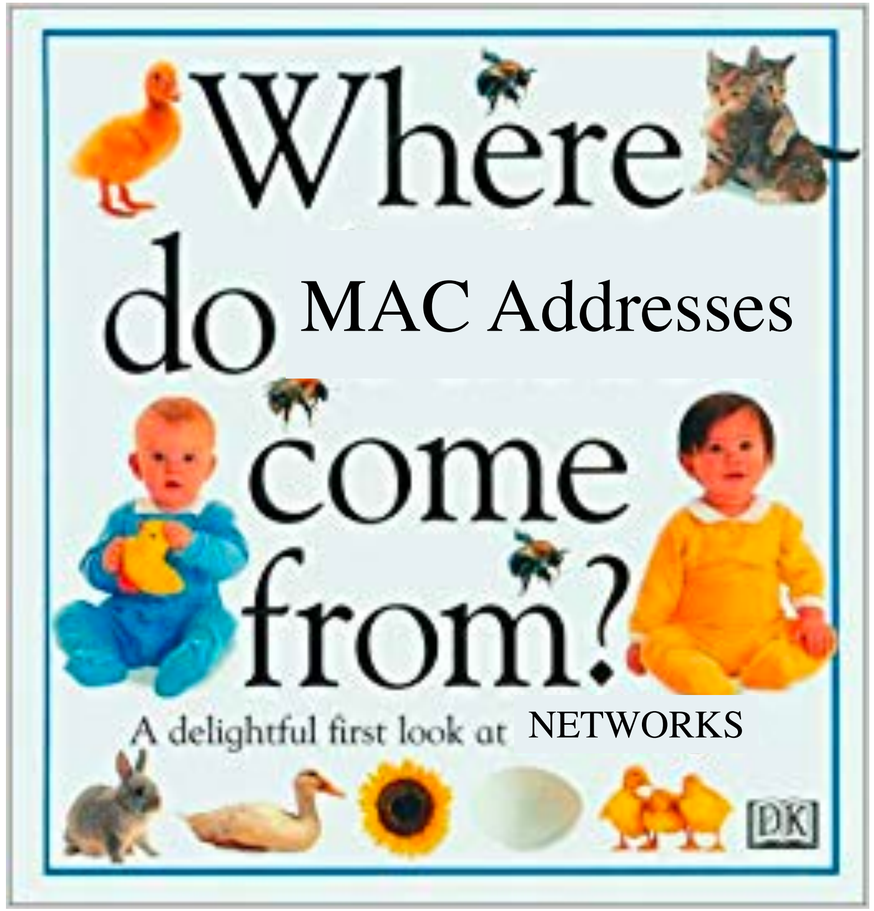


All MAC addresses are assigned by your device's manufacturer.

Every wireless adaptor, ethernet port, bluetooth connector you have has a unique number assigned to it by the manufacturer.

I found this crazy when I learned this!

MAC Addresses

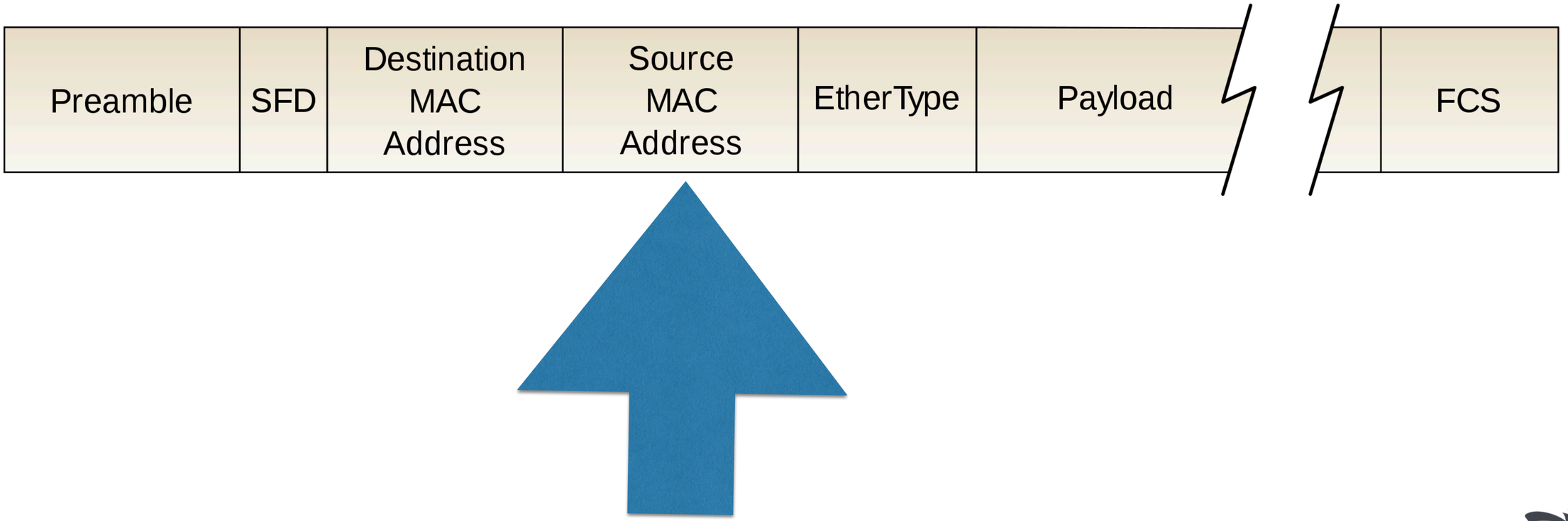


How do I learn someone's MAC address so I can send to them?

I'll tell you when you're older.

(For now, just pretend you MAC addresses are like phone numbers — you have to ask someone for their number in the real world.)

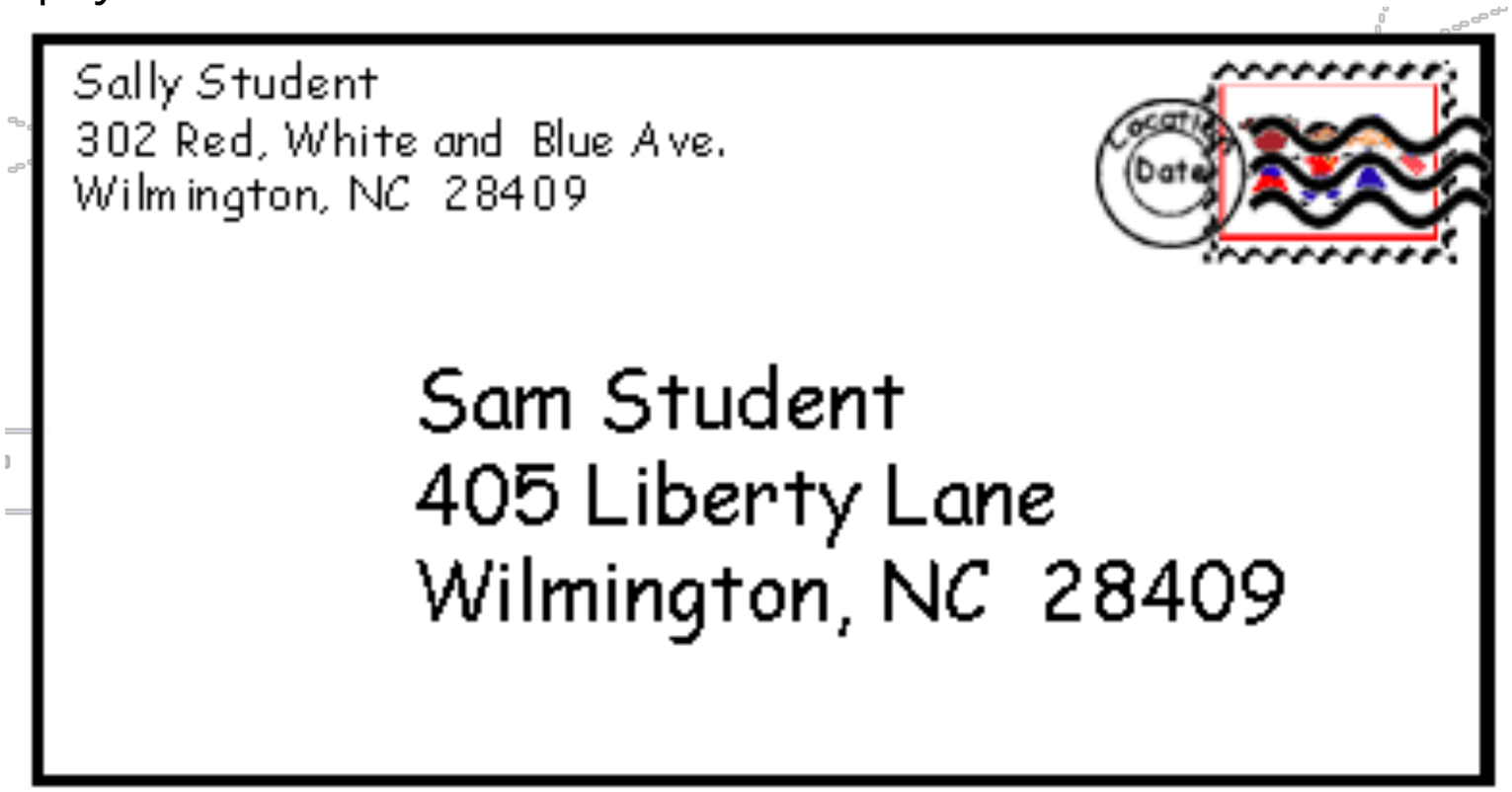
Back to Ethernet



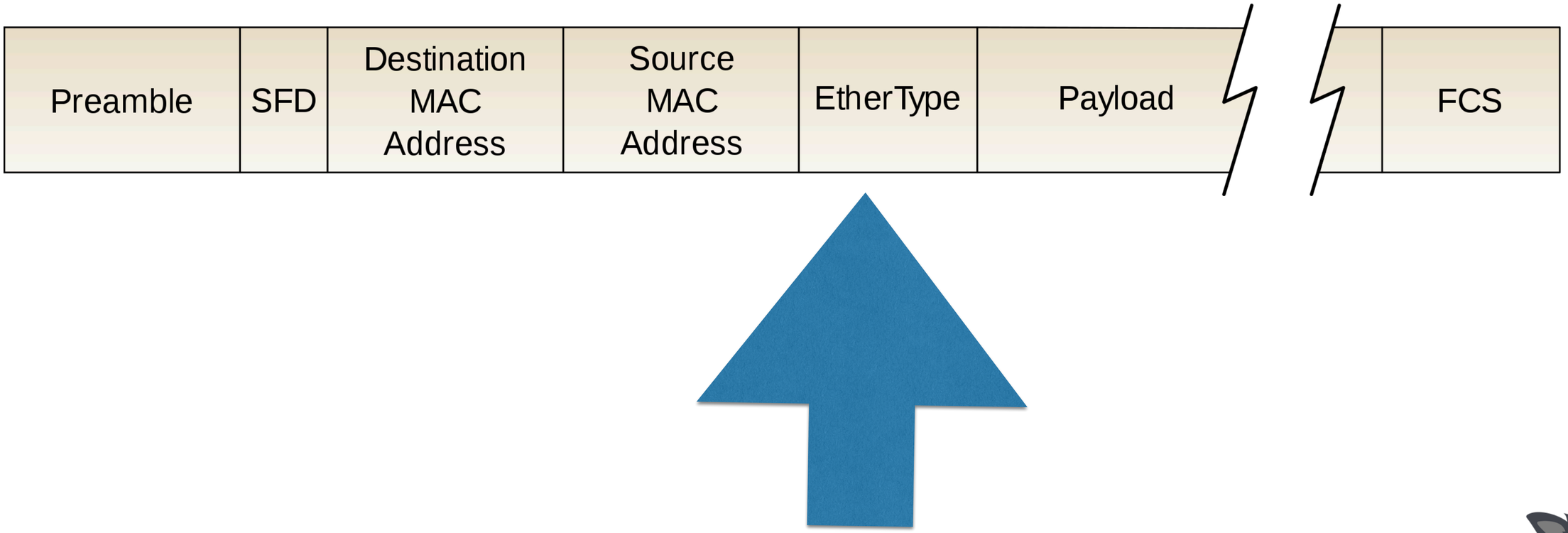
Address of the host who is sending the packet

Why would we need the source address?

Reply-to….


Sally Student
302 Red, White and Blue Ave.
Wilmington, NC 28409
Date
Sam Student
405 Liberty Lane
Wilmington, NC 28409

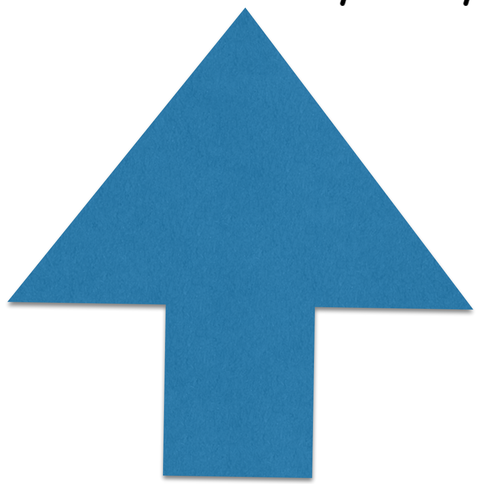
Back to Ethernet



Type of data inside the packet (more on this next week)

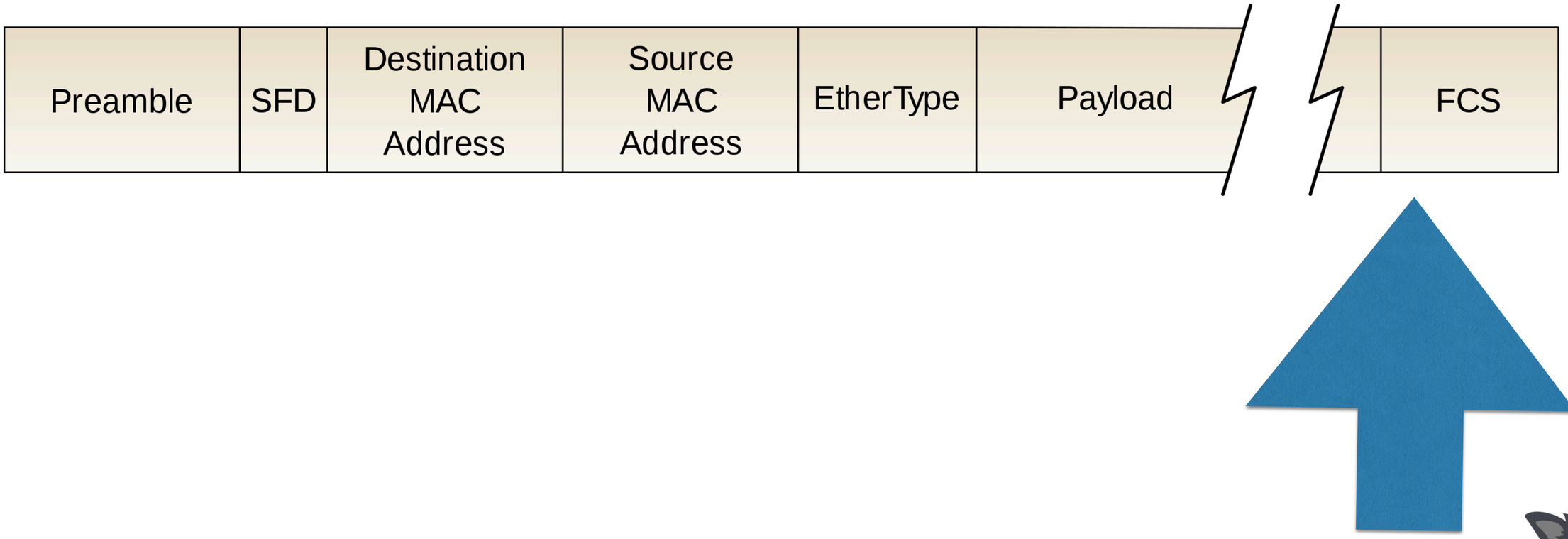
Back to Ethernet

Preamble	SFD	Destination MAC Address	Source MAC Address	EtherType	Payload	FCS



Finally! The data you want to send!

Back to Ethernet



Frame check sequence

(We'll talk about this later too — it helps you detect errors in the packet)

So You Want To Build a Local Area Network

1. Choose an addressing scheme

    - Great! Using Ethernet, we have MAC addresses that let us know where to send packets, and that let the receiver know where to reply-to.

2. Choose a *routing algorithm* to deliver your packets to the right place.

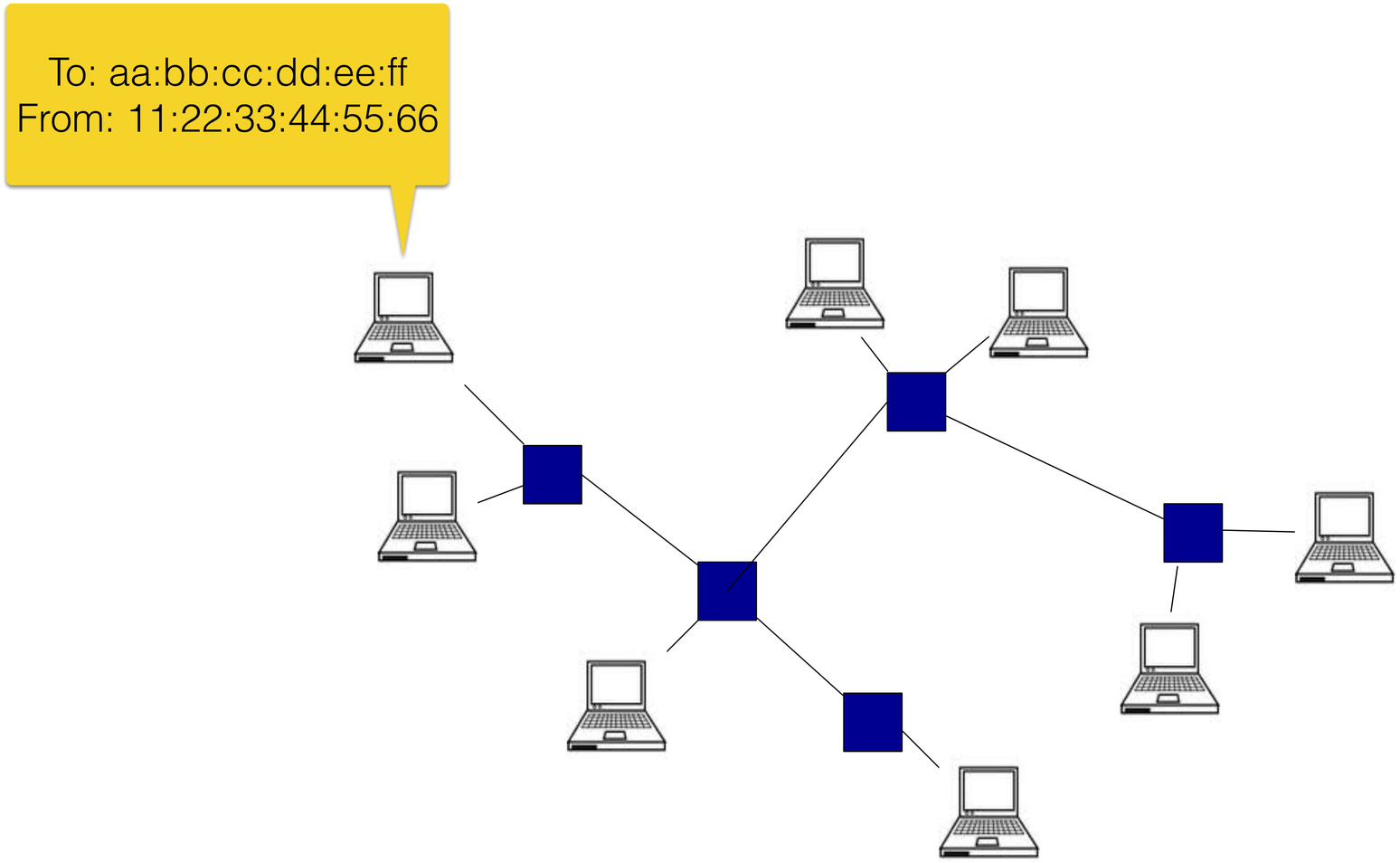
    - Literally the entire rest of this lecture… and 2 of the next 4 lectures too.

Cheater Solution

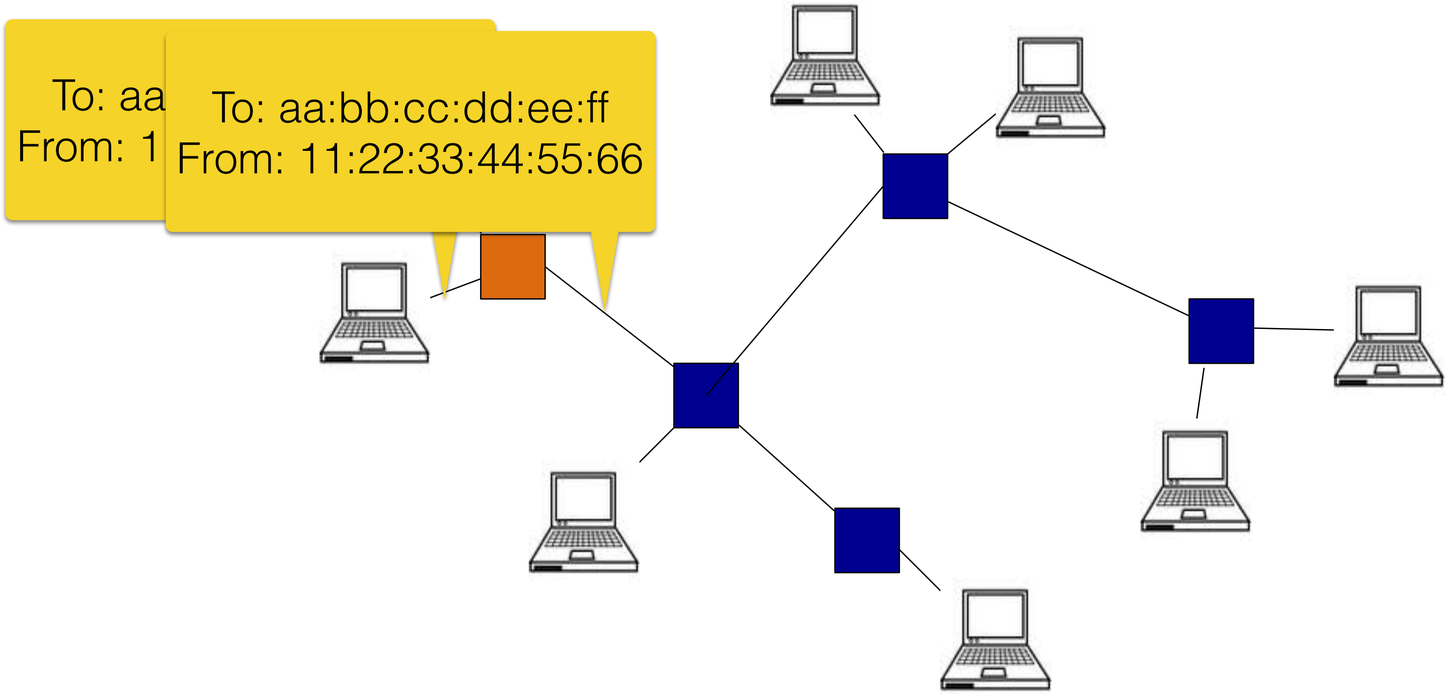
- Hard-Code Everything

- Better Solution: Plug and Play
  - Network should self-update if a link or a router goes down.
  - New computers should be able to join the network easily
    - Or move around!
  - The network should be able to grow without having to update all the old infrastructure

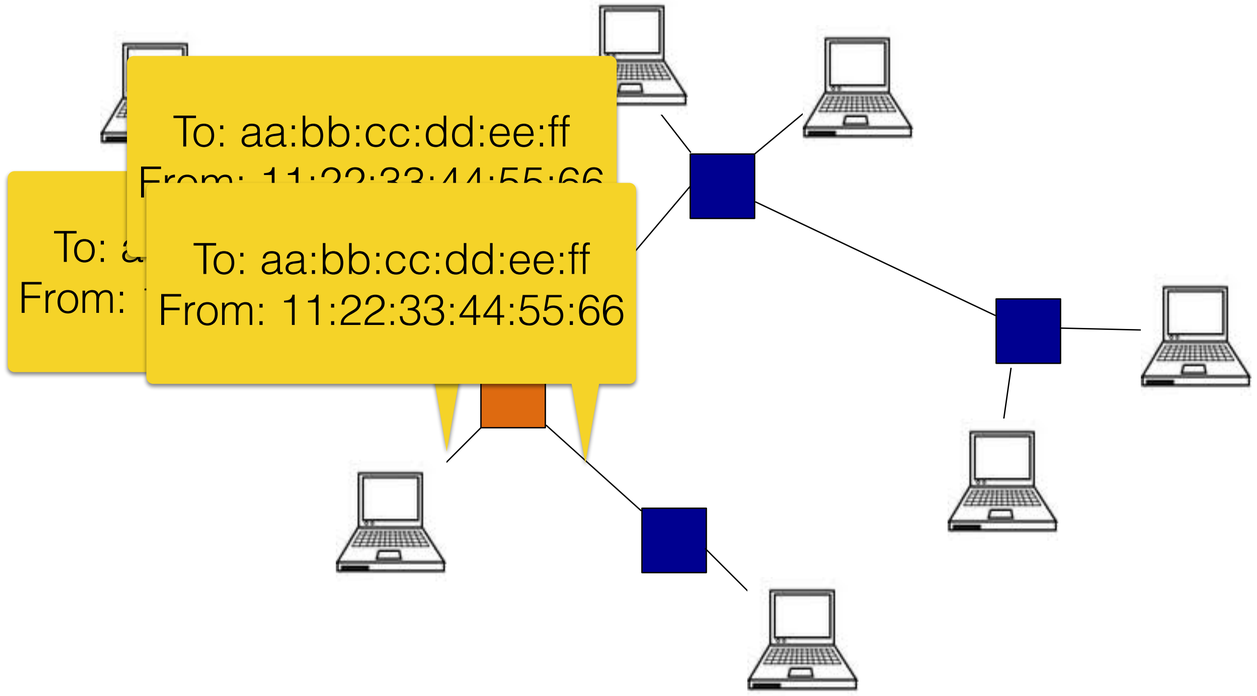
Routing, Generation 1: Broadcast



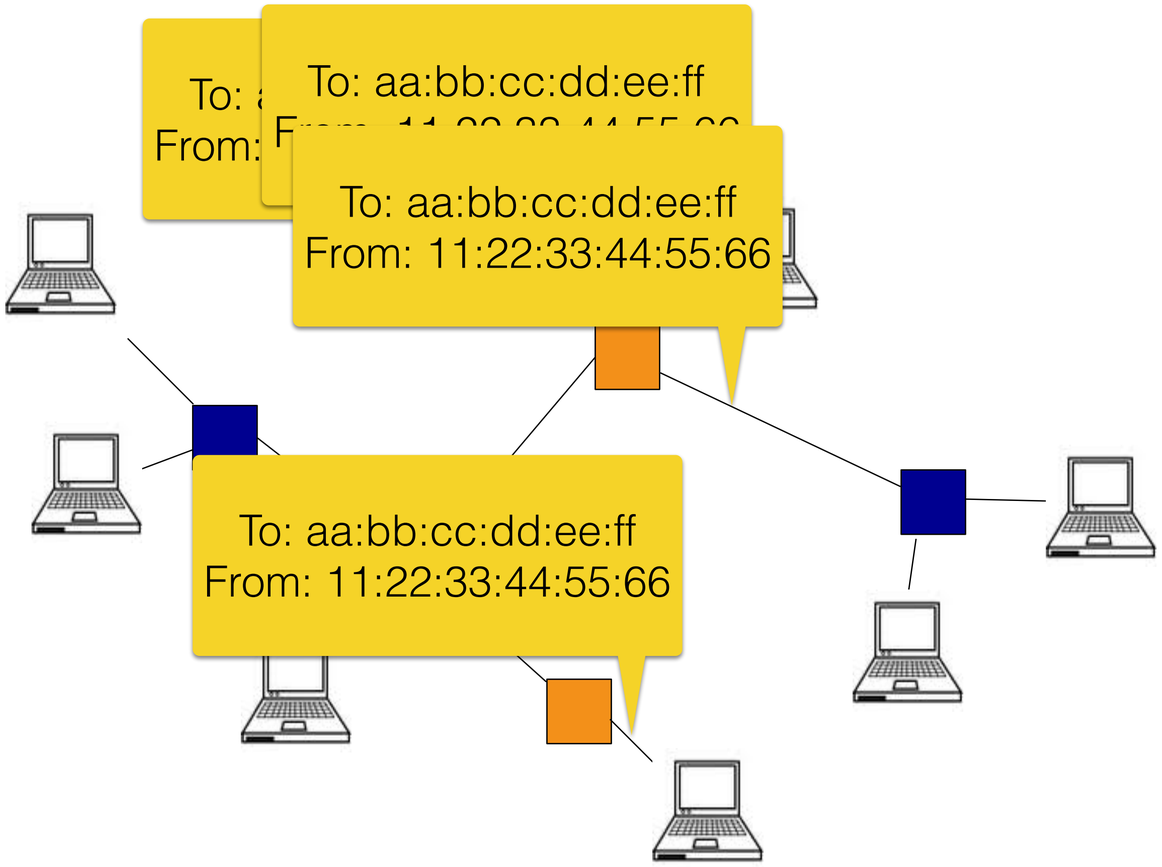
Routing, Generation 1: Broadcast



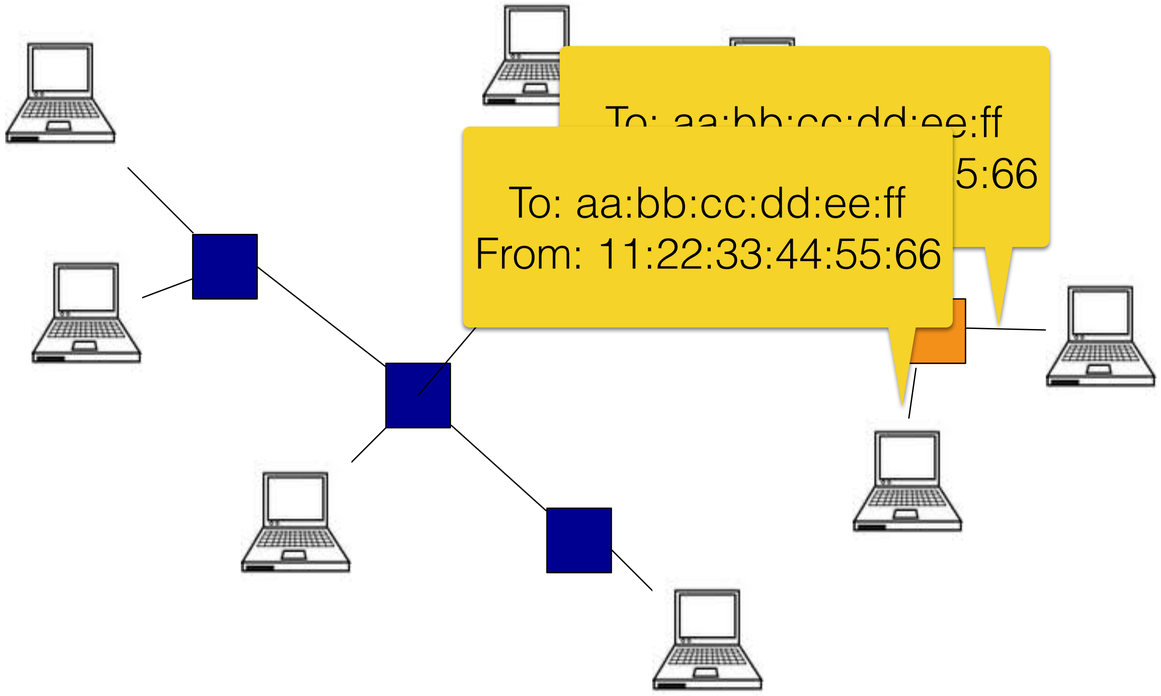
Routing, Generation 1: Broadcast



Routing, Generation 1: Broadcast



Routing, Generation 1: Broadcast



Result:

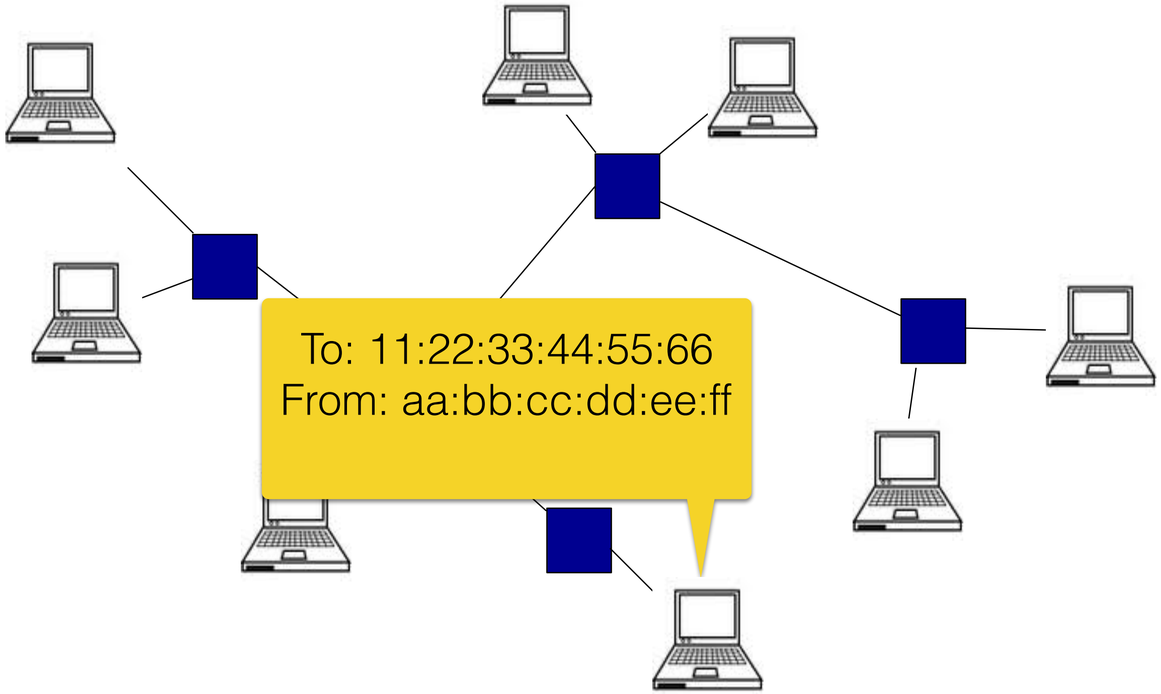
Everyone receives the packet!

Only the receiver responds.
(Anything wrong with this?)

Fun Fact

- You can often listen in to Ethernet packets being sent around when you're sharing a network with others.
  - On WiFi…
  - Plugged in to an Ethernet port at CMU…
  - All you need to do is put your network interface in "promiscuous" mode.
- We will show you how to do this later this semester.
  - Use your powers for good (debugging) not evil!

The Only Smart Thing In Gen 1: Learning Bridges

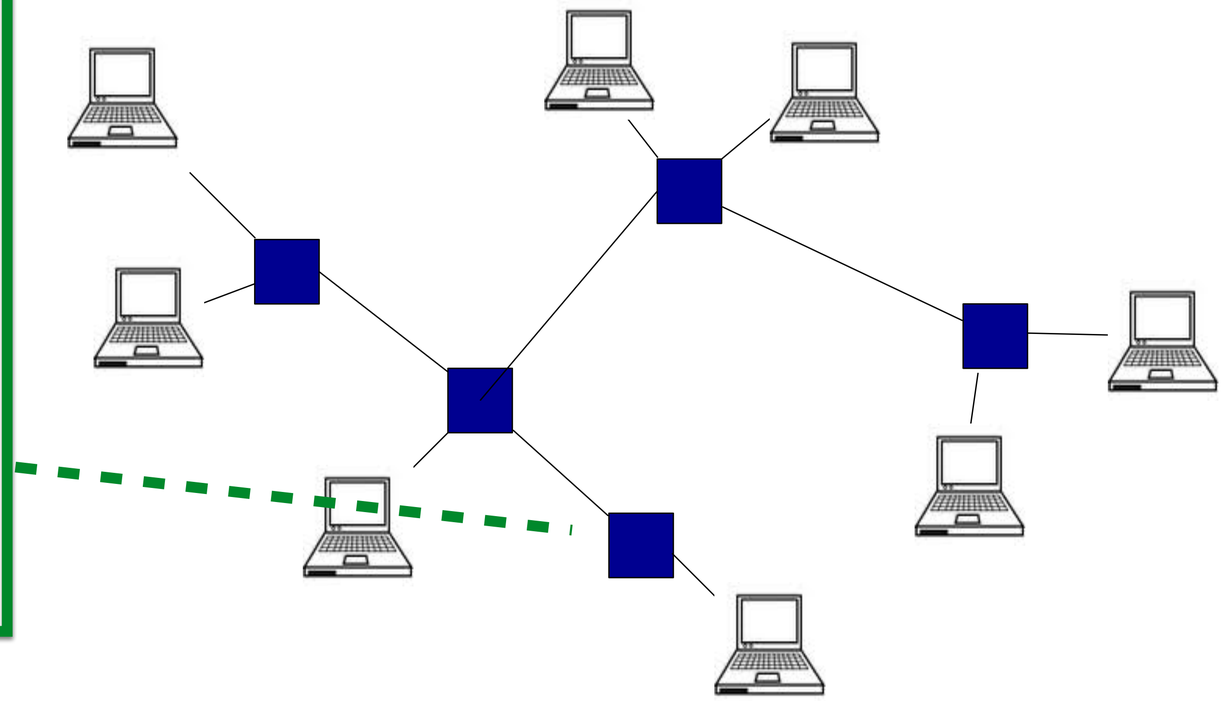


What Happens When aa:bb:cc:dd:ee:ff replies? Does everyone hear it again?

The Only Smart Thing In Gen 1: Learning Bridges

Every switch maintains a table: “If you want to send a packet to this MAC address, send it on this port.”

MAC Address	Port	Age
A21032C9A591	1	36
99A323C90842	2	01
112233445566	2	15
301B2369011C	2	16
AABBCCDDEEFF	1	11



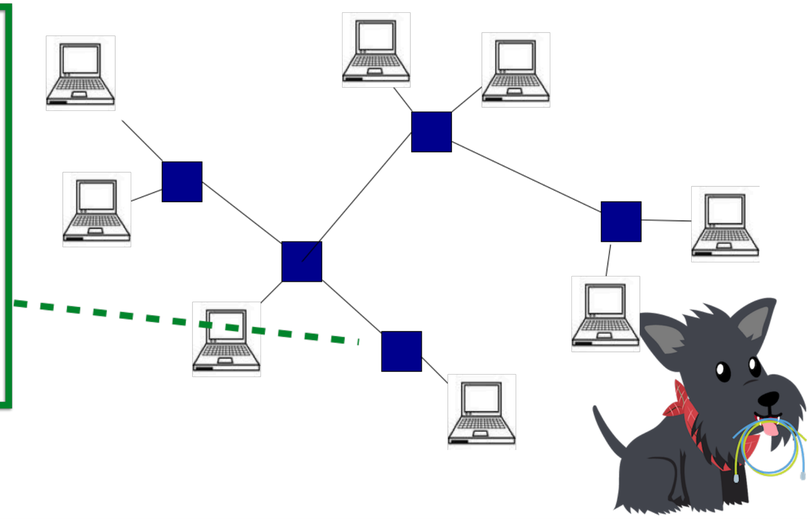
Learning Switch Algorithm

Receive_Packet (packet, ingress_port, time_now):
  Is Source MAC Address in Table?
    If no, insert source (address, ingress_port, time) into table

  Is Destination MAC address in Table?
    If no, send out all ports (except the one it came in on)!!
      (except ingress_port)
    If yes, just send out the port from table

Clean_Up(time_now):
  foreach entry:
    if (time_now - entry.time) is big
      delete entry

MAC Address	Port	Age
A21032C9A591	1	36
99A323C90842	2	01
112233445566	2	15
301B2369011C	2	16
AABBCCDDEEFF	1	11



Why do we have this age/ timeout stuff?

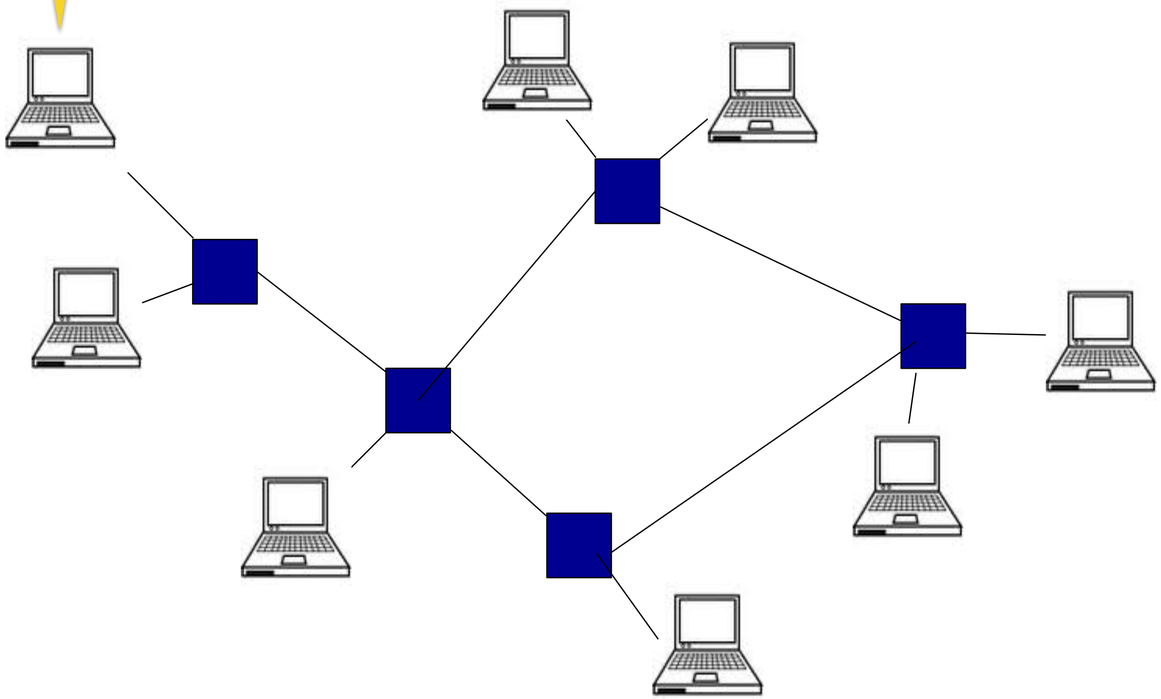
Broadcast + Learning Briges, Recap

- Pros:
  - Self-organizing — “plug and play”!
  - Hardware and algorithms are fairly simple.
  - State: Each switch maintains O(number of hosts) state.
- Cons?

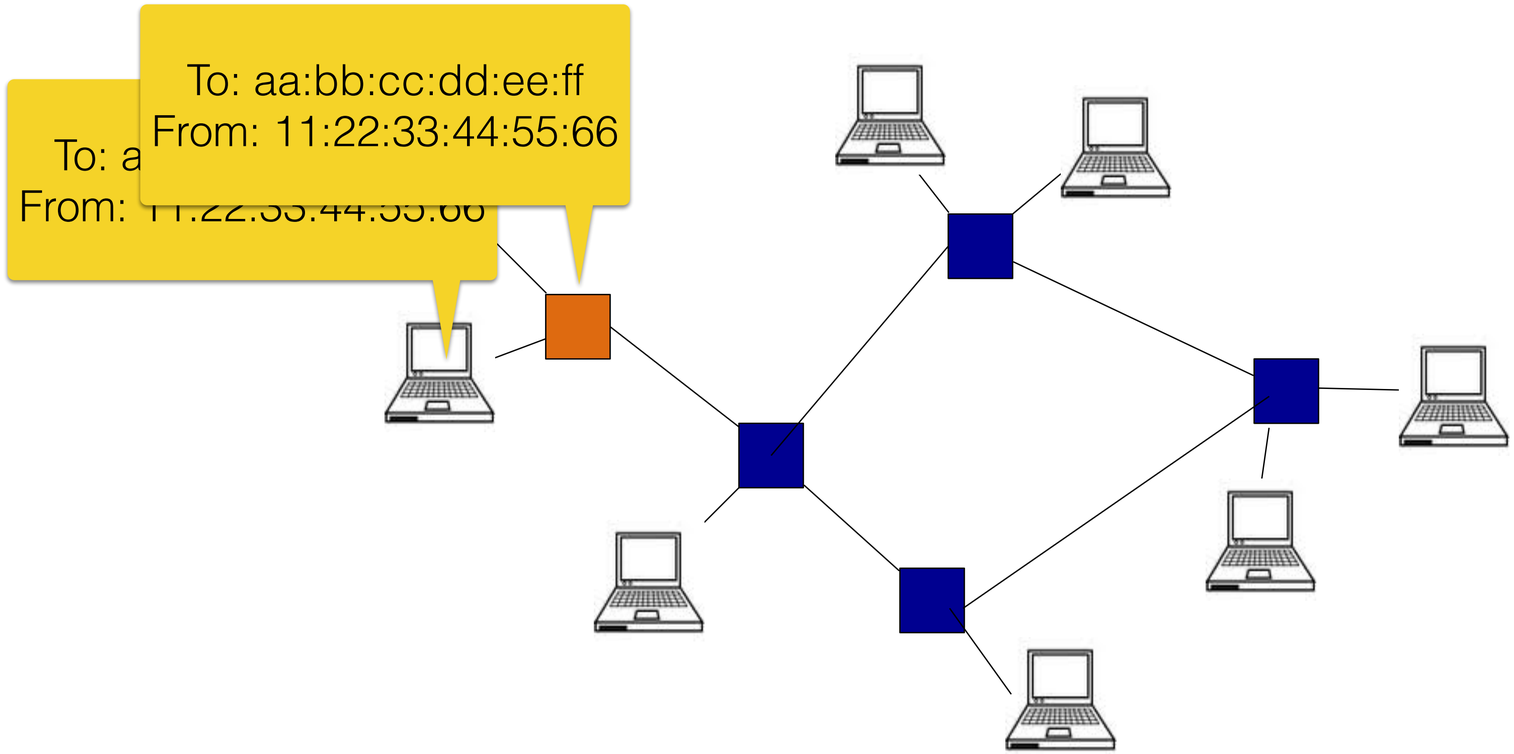
There’s a major problem…!

Routing, Generation 1: Broadcast

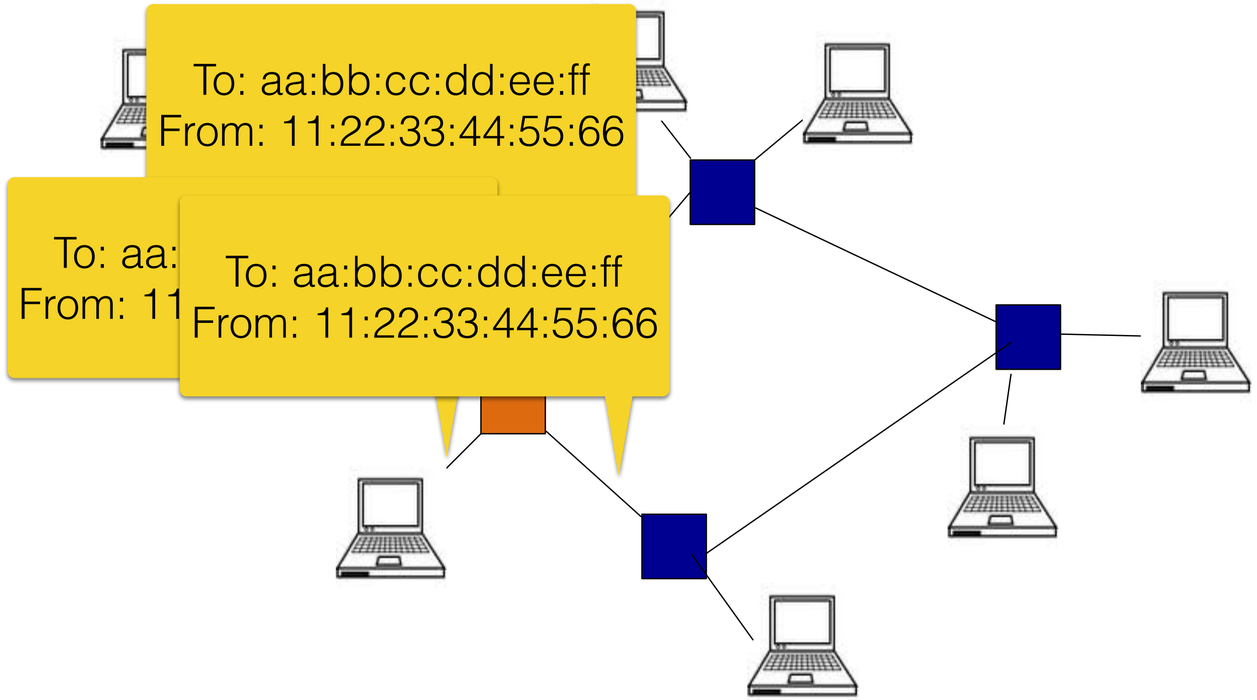
To: aa:bb:cc:dd:ee:ff
From: 11:22:33:44:55:66



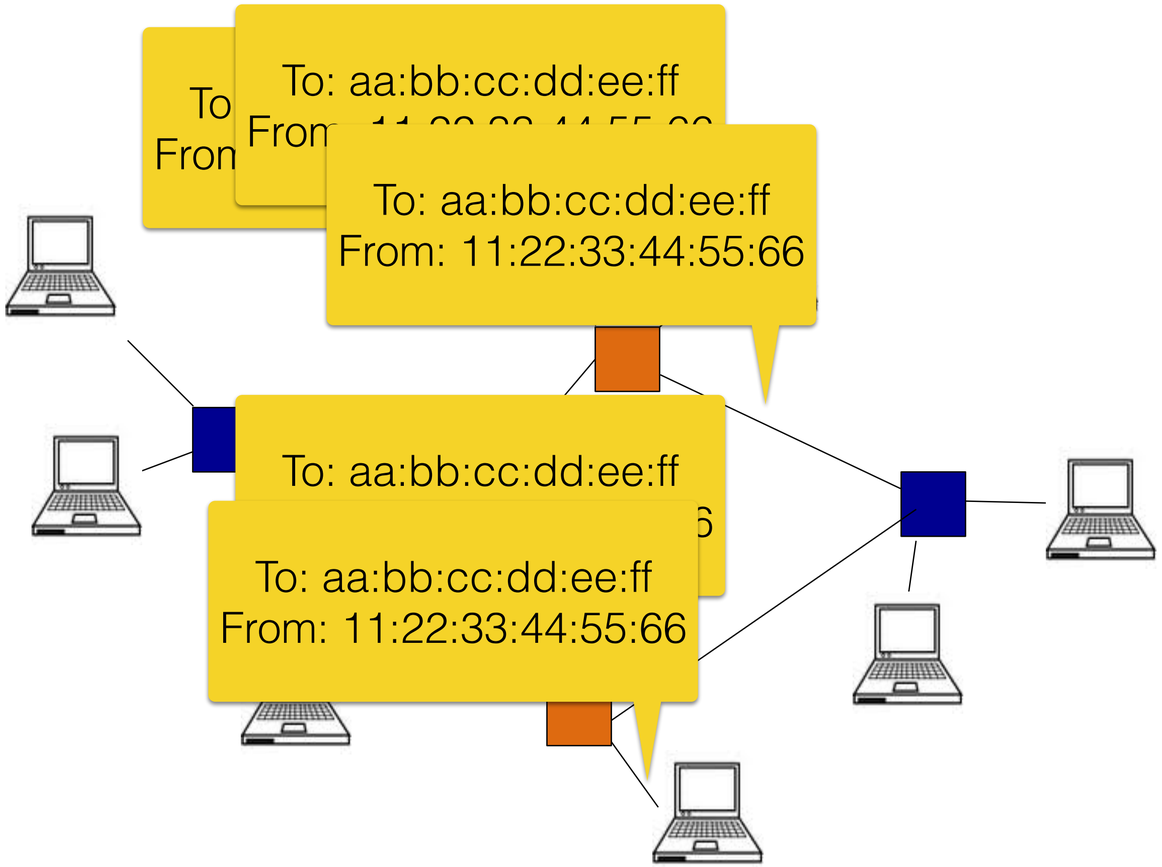
Routing, Generation 1: Broadcast



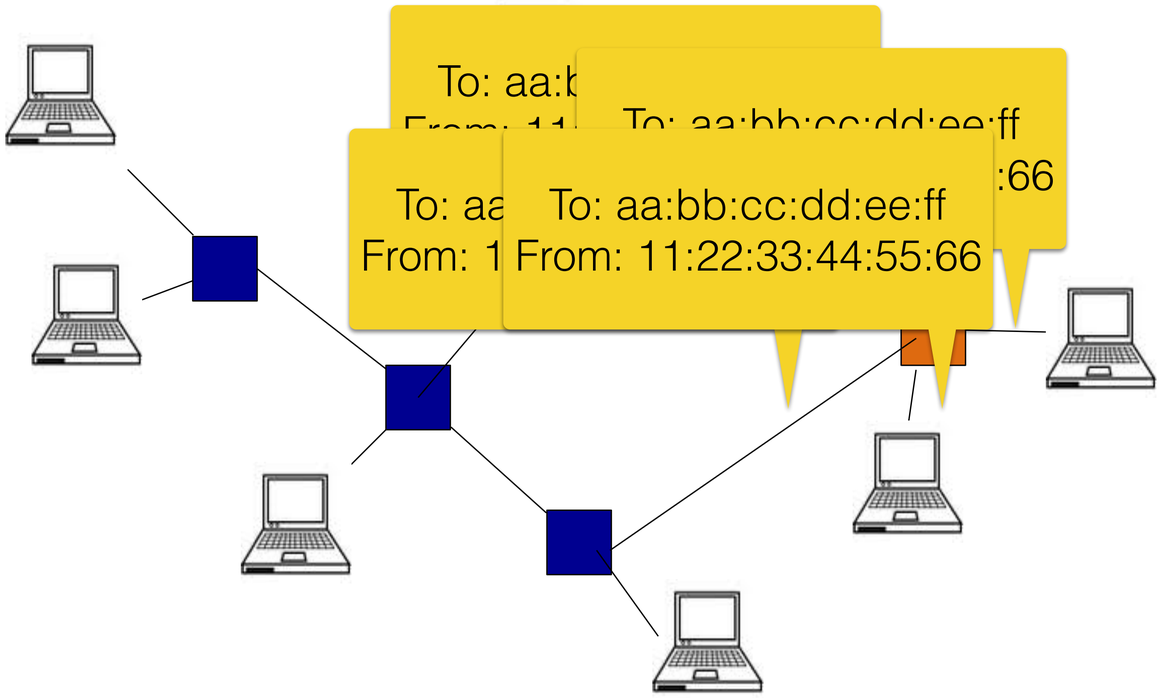
Routing, Generation 1: Broadcast



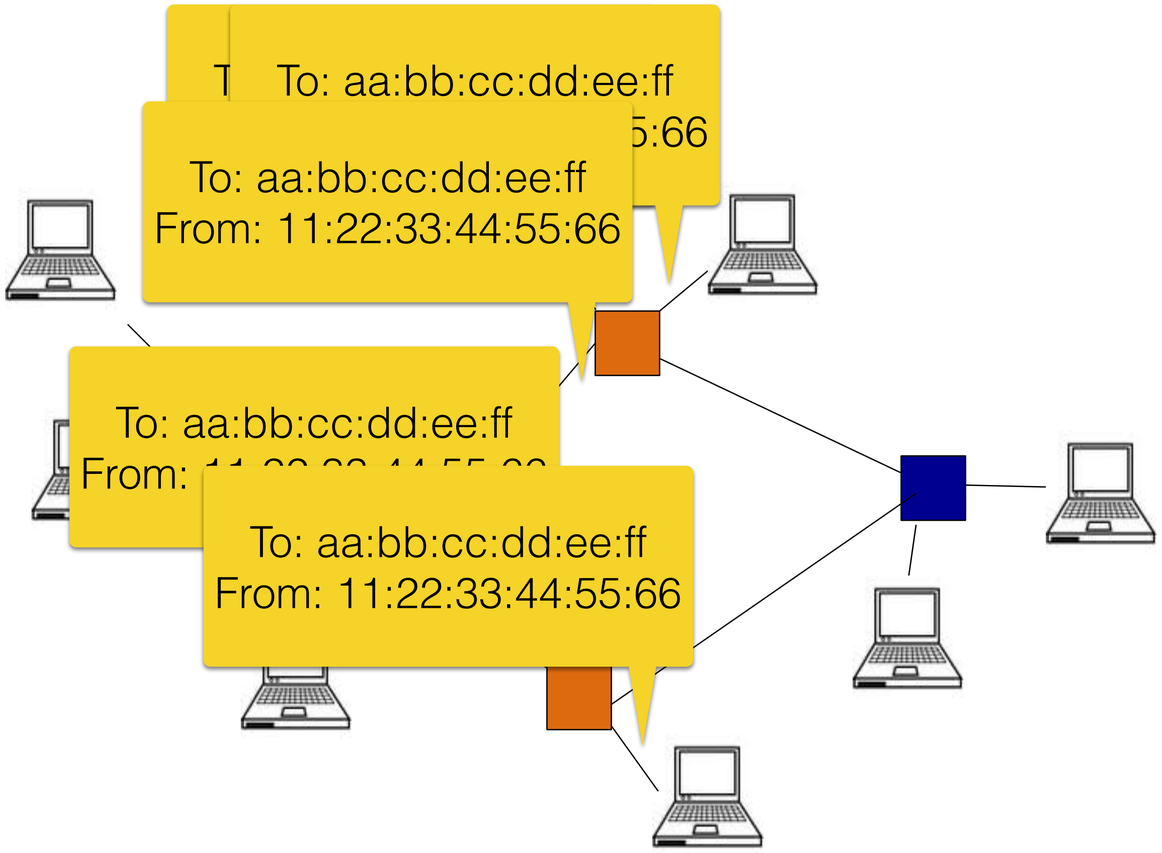
Routing, Generation 1: Broadcast



Routing, Generation 1: Broadcast



Routing, Generation 1: Broadcast



…forever

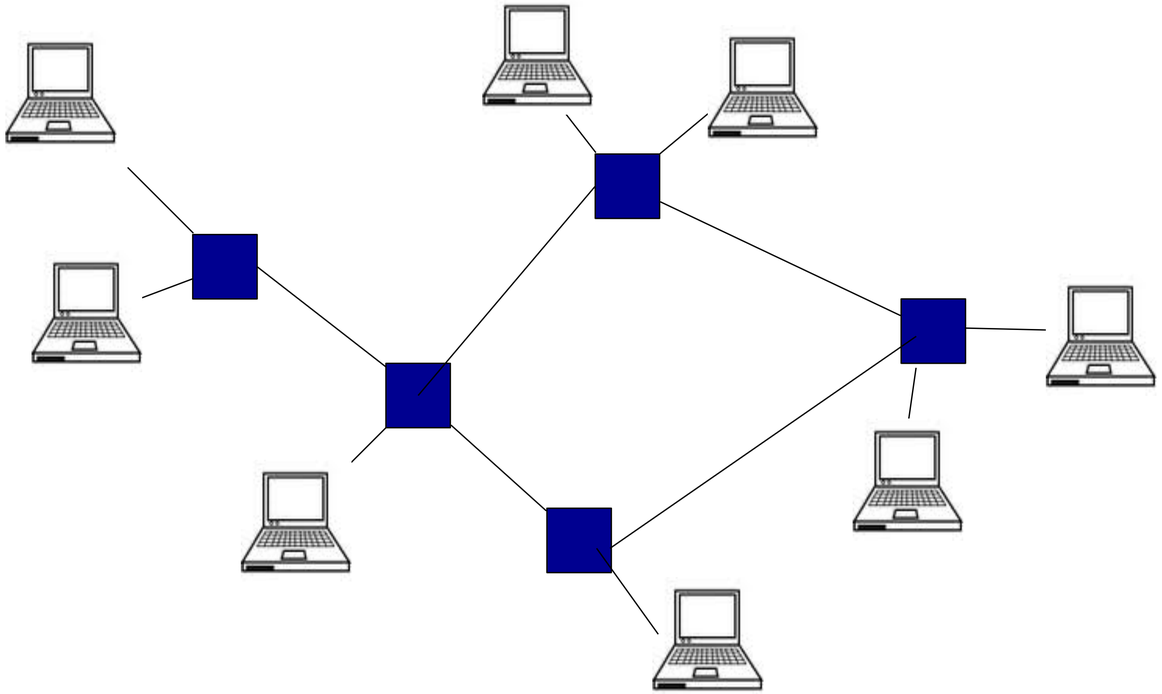
This is called a “broadcast storm.”

Solutions?

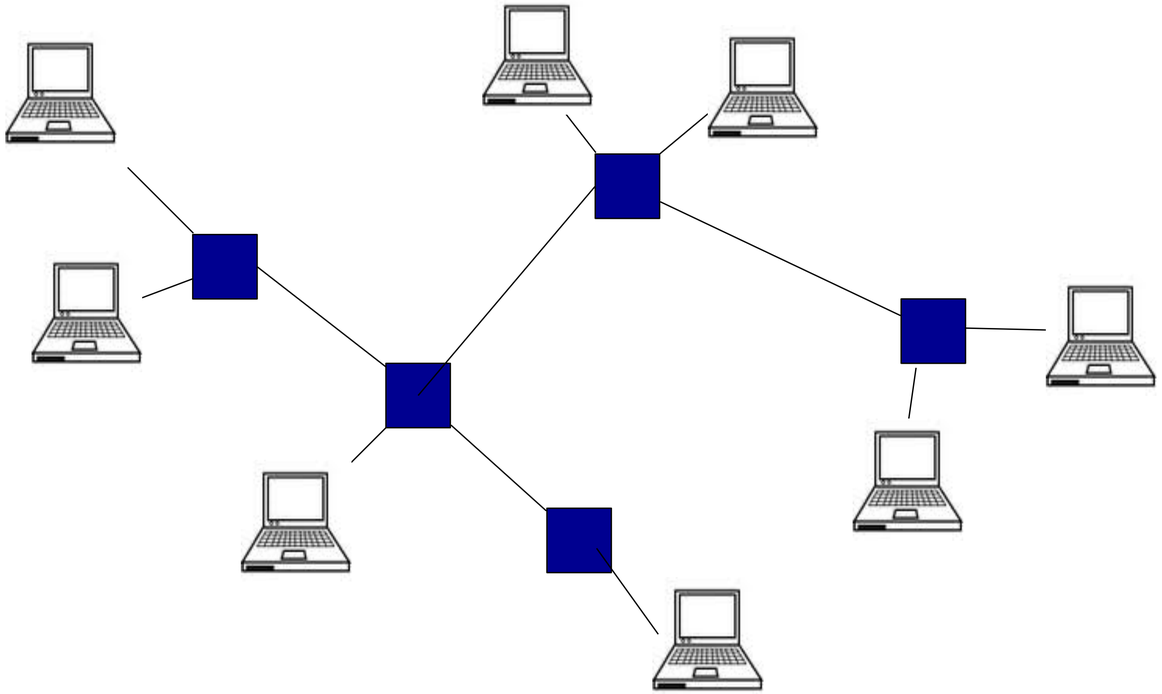
Just make your network be a tree.

Just don’t plug in anything that makes a loop?

How do we turn this…



…into this?



Spanning Tree Protocol

I think that I shall never see
A graph more lovely than a tree.
A tree whose crucial property
Is loop-free connectivity.
A tree that must be sure to span
So packets 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

What is a Spanning Tree?

- Reduce our topology graph to a tree:
  - Make sure there are no loops in the topology
  - All LAN segments are still connected to the LAN and can receive messages
- Main idea: Bridges choose the ports over which they have to forward frames.

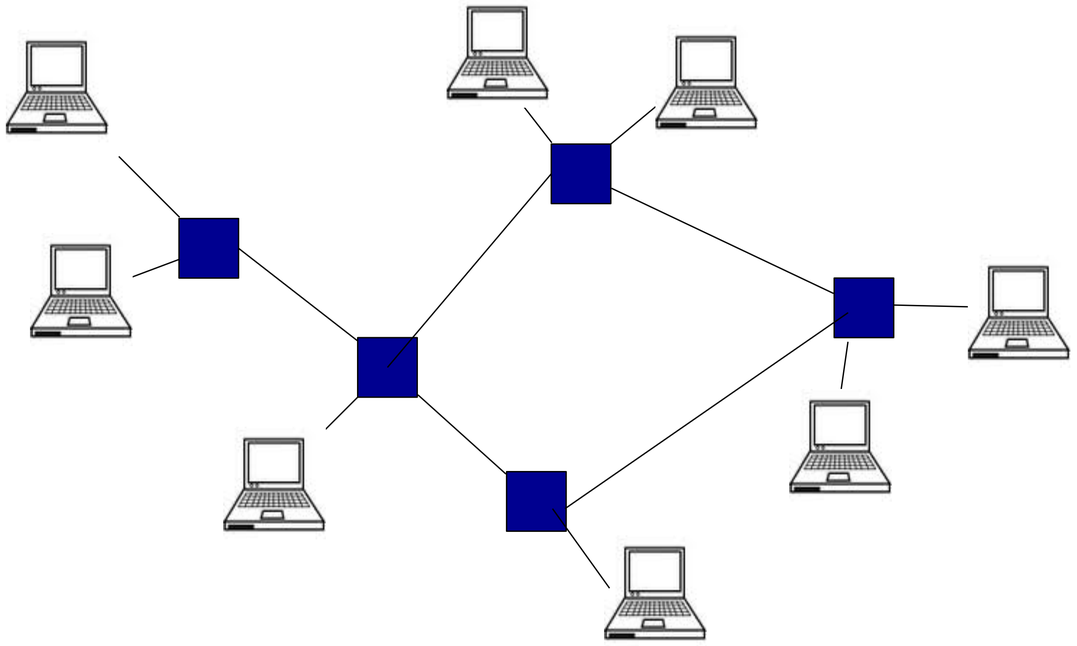
Distributed Spanning Tree Overview

Embed a tree that provides a single unique default path to each destination:

- Bridges designate ports over which they will or will not forward frames
- By removing ports, networks is reduced to a tree
- Addresses the broadcast storm; but tree is not resilient
    - When switch/link fails, rerun protocol to converge to new tree

Distributed Spanning Tree Protocol

- Bridge with lowest ID (MAC address) is the “root”
  - All ports are part of tree
- Each bridge finds shortest path to the root.
  - Remembers port that is on the shortest path



Everyone keeps a simple data structure

- (Root, Path Length, Next Hop)
  - Root: the root of the tree
  - Path Length: the number of switches you have to go to to reach the root
  - Next Hop: The switch you should forward packets to for them to reach the root.

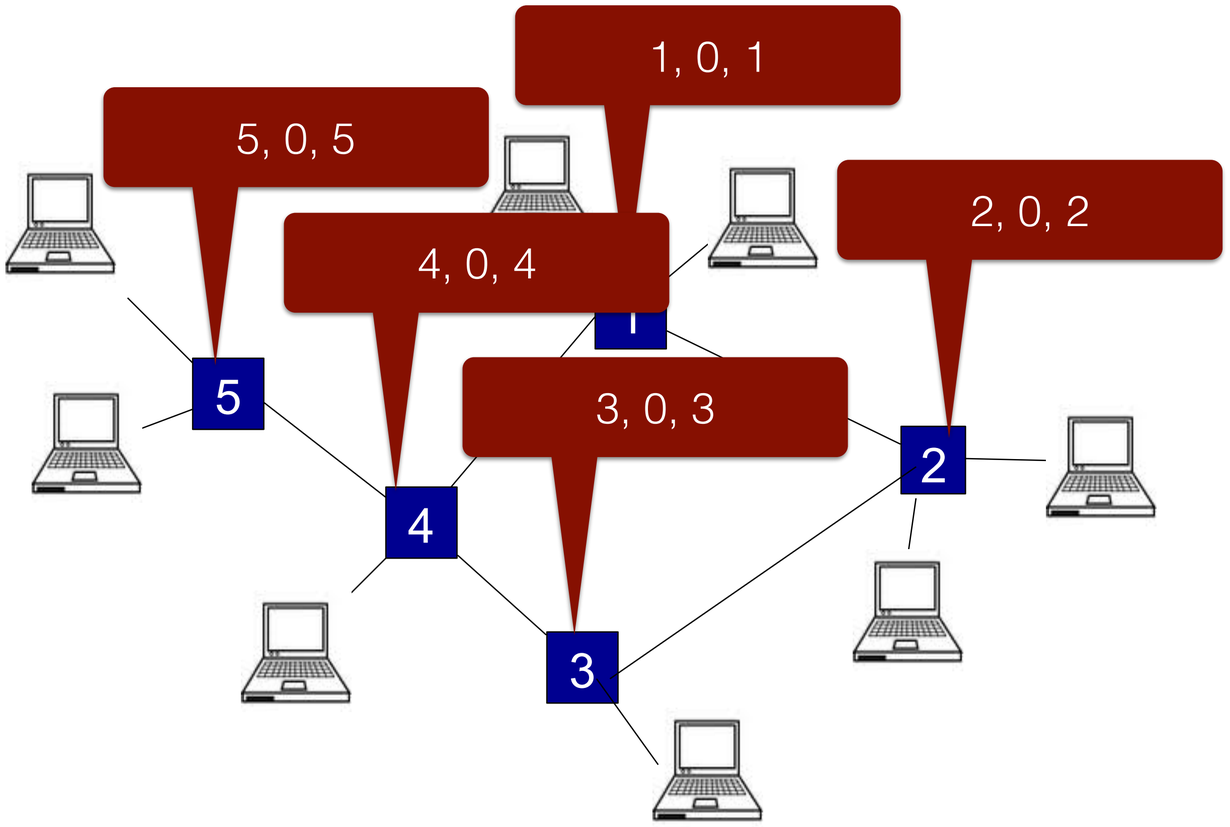
Basic Algorithm, if you are a switch

- Look at your ID (MAC address). That is your ID. Assume you are the root. Store (Me, 0, Me) in your data structure.
- do{
  - Tell your neighbors (root, pathLength, yourID)
  - Listen to your neighbors when they tell you their path to the root.
    - If their ID number is smaller, replace root/path with their root/path, incrementing PathLength by one
    - If their ID number is the same but their path length is shorter, replace your root/path with theirs, incrementing PathLength by 1
    - If neighbor A and neighbor B both tell you the same ID and path length, choose to route through A since A is lower than B.

  while(I keep getting new updates)

Basic Algorithm, if you are a switch

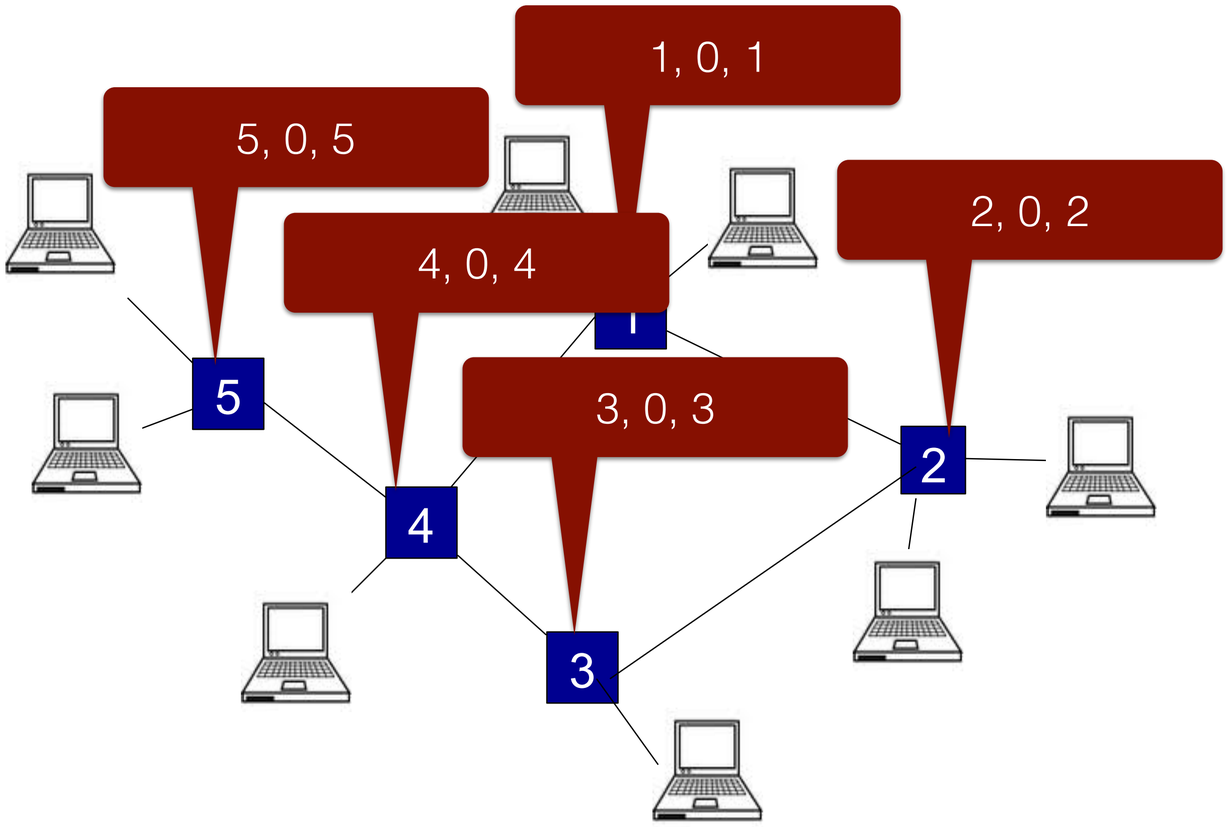
- Now that you know where the root is and how to get it:
- Disable all ports that do not
    - (a) Connect you to the root
    - (b) Connect someone to you who uses you to get to the root.

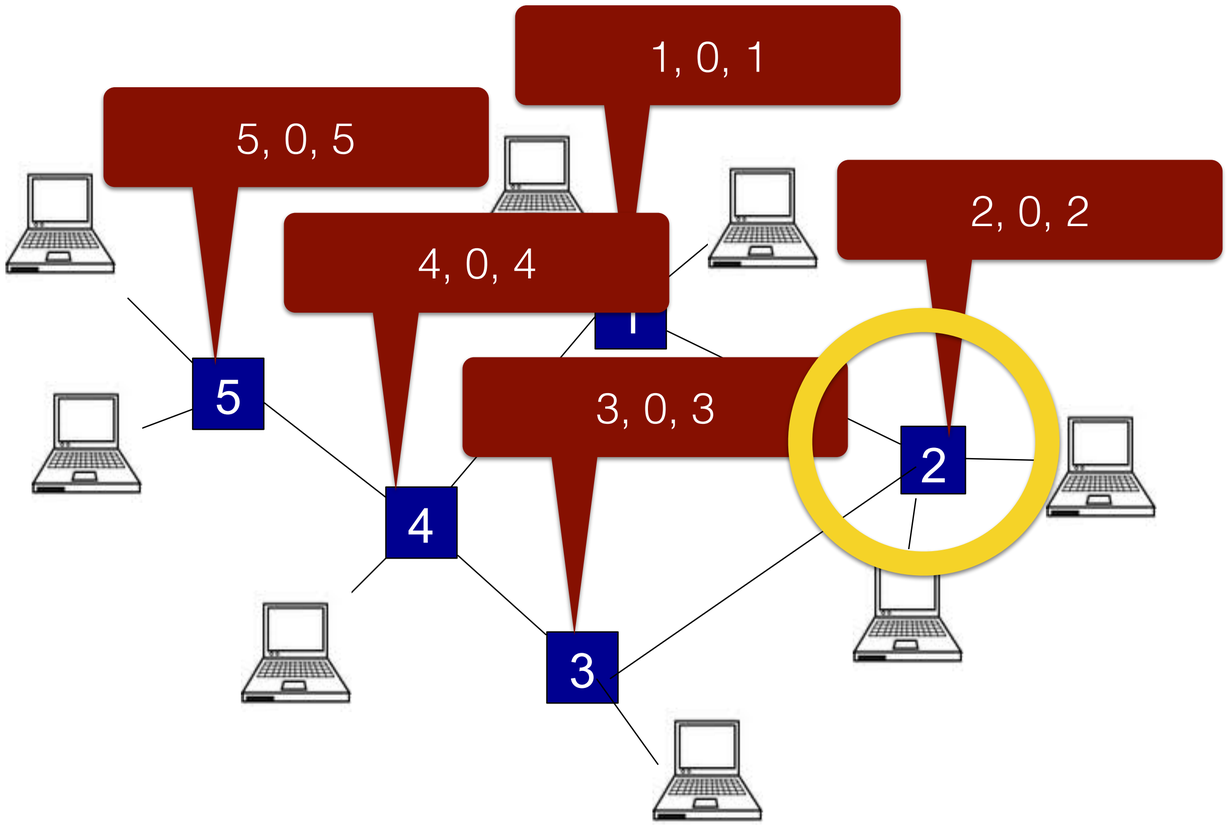


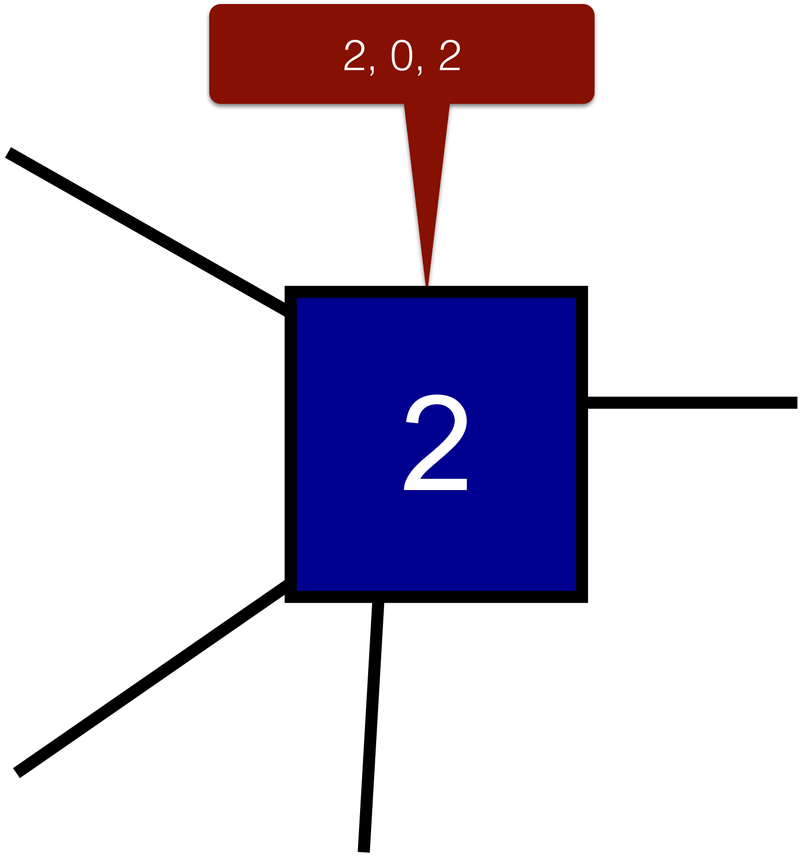
Note: these are special *control messages* which are not broadcast

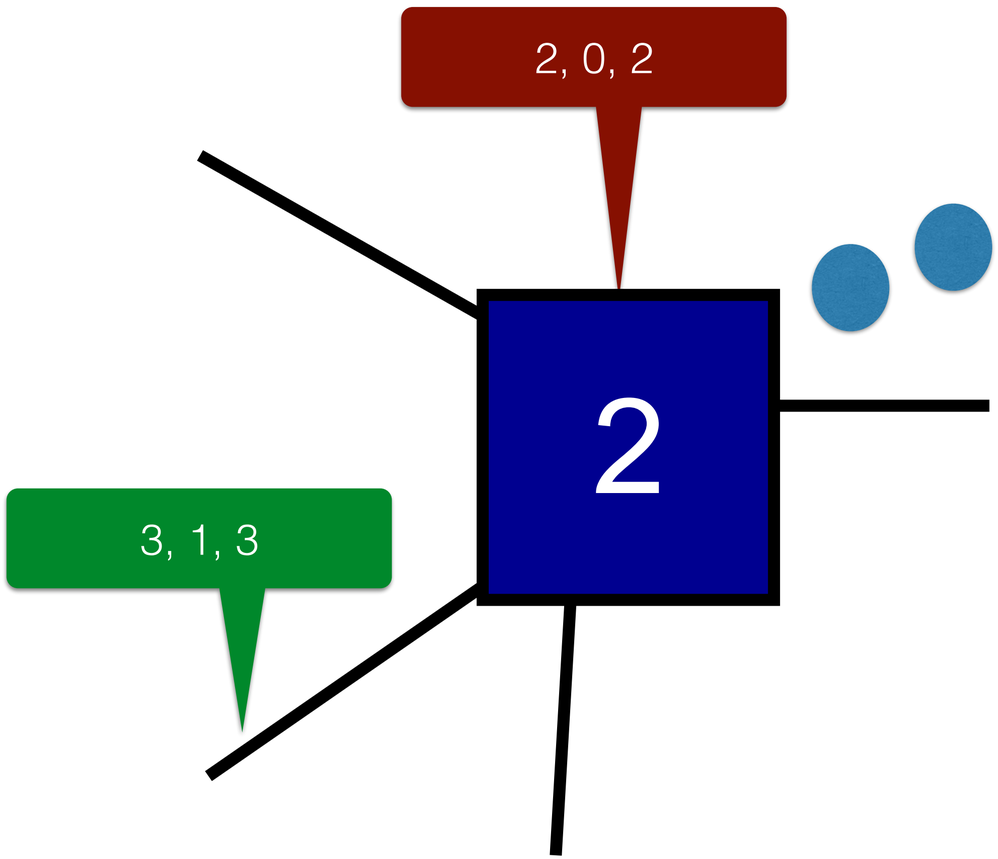
Things are about to get weird

- This is a *distributed algorithm*
  - That means that all of the nodes operate on their own time scales, at the same time
- This makes it hard to reason about the order things happen in across the whole system
  - It's easiest to think about the system just one node at a time.

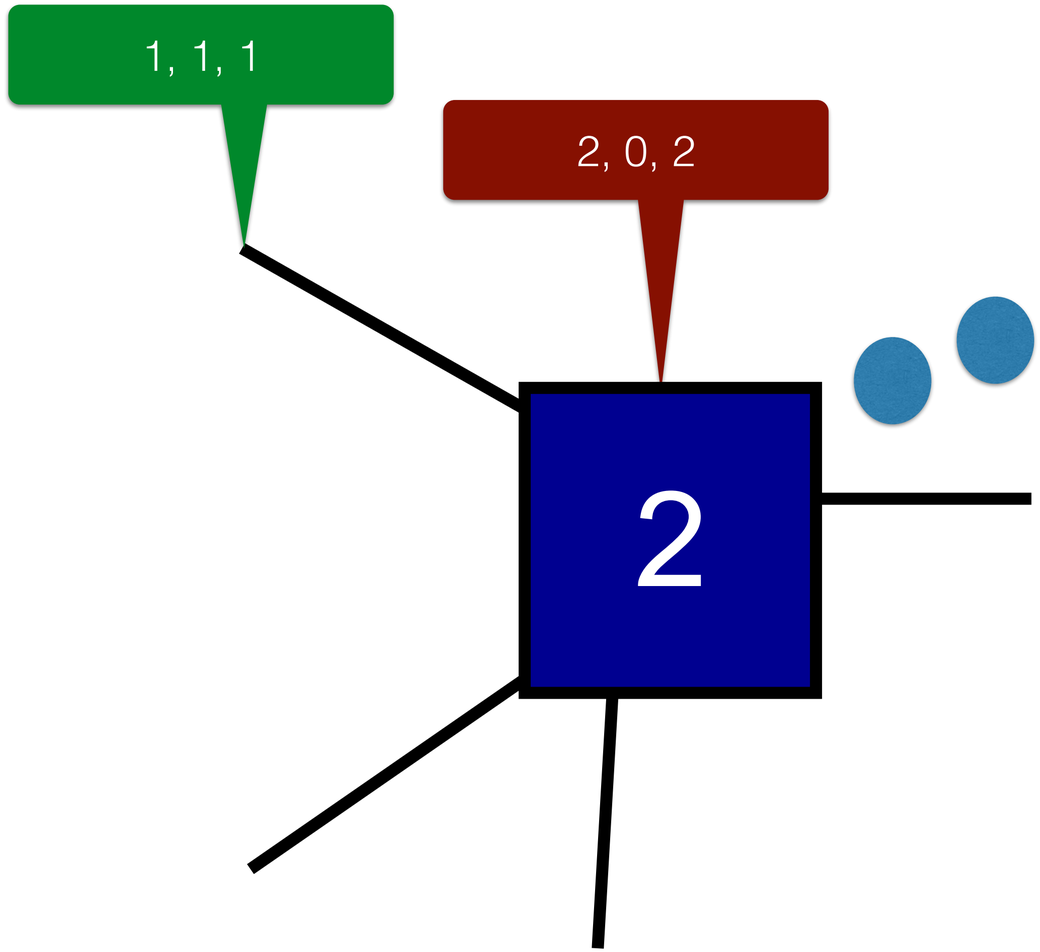

1, 0, 1
5, 0, 5
4, 0, 4
2, 0, 2
3, 0, 3
1
5
4
2
3


1, 0, 1
5, 0, 5
4, 0, 4
2, 0, 2
3, 0, 3
1
5
4
2
3

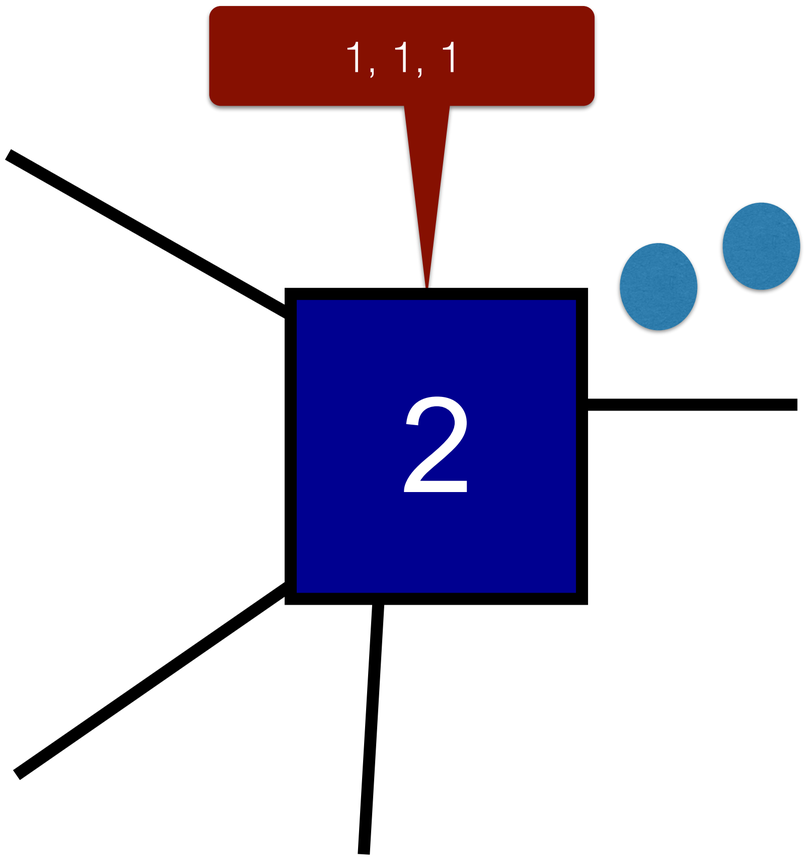

2, 0, 2
2



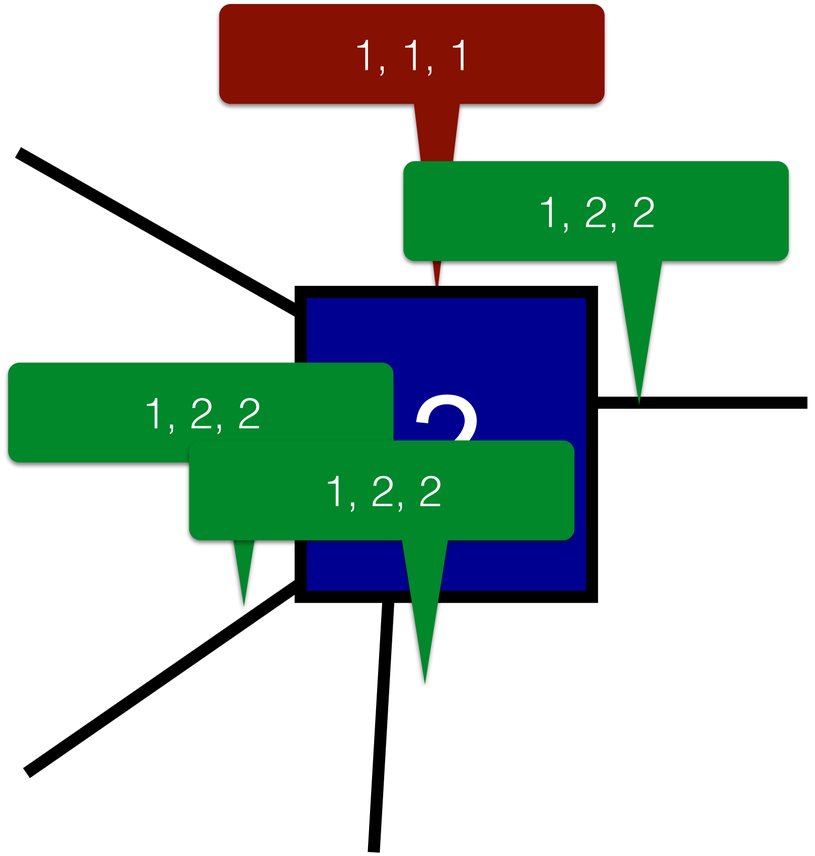
Root node ID for this new route is **higher** than the current node ID. I should keep my old route.

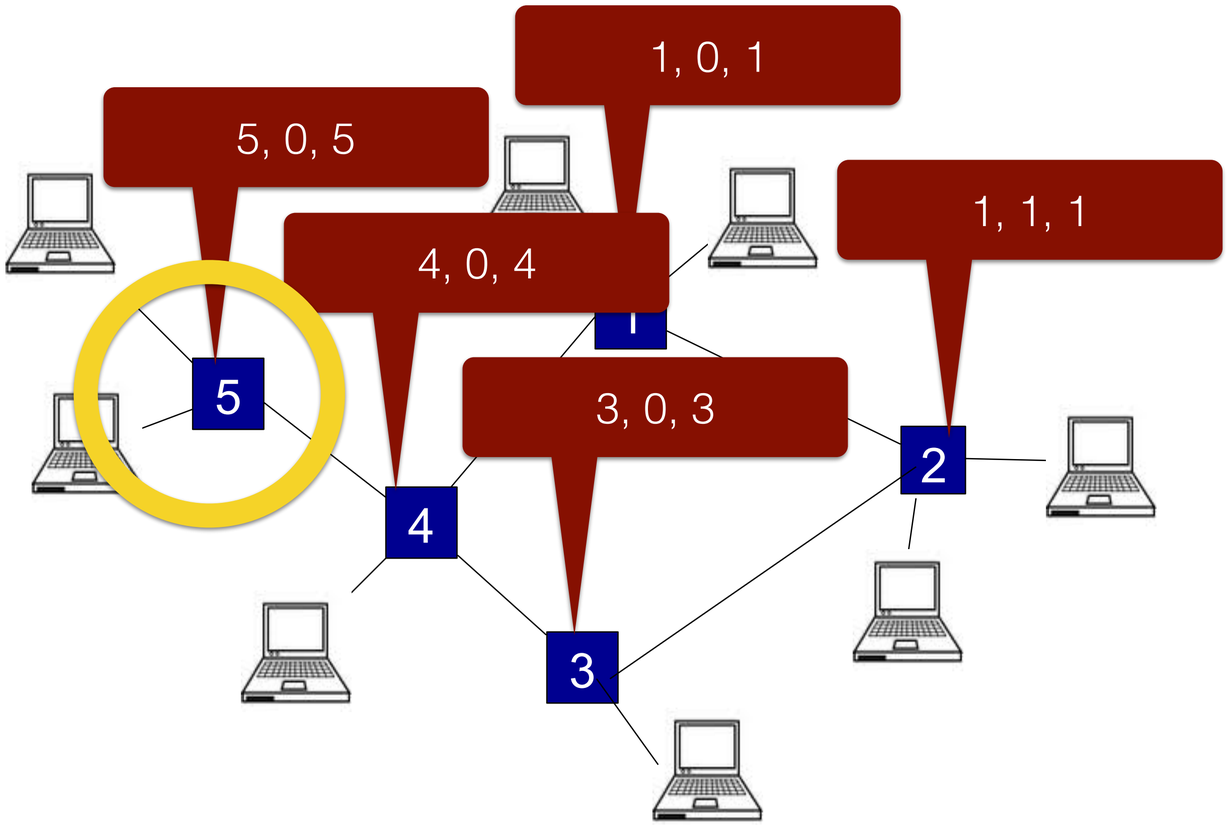


Root node ID for this new route is lower than the current node ID. I should update my route!

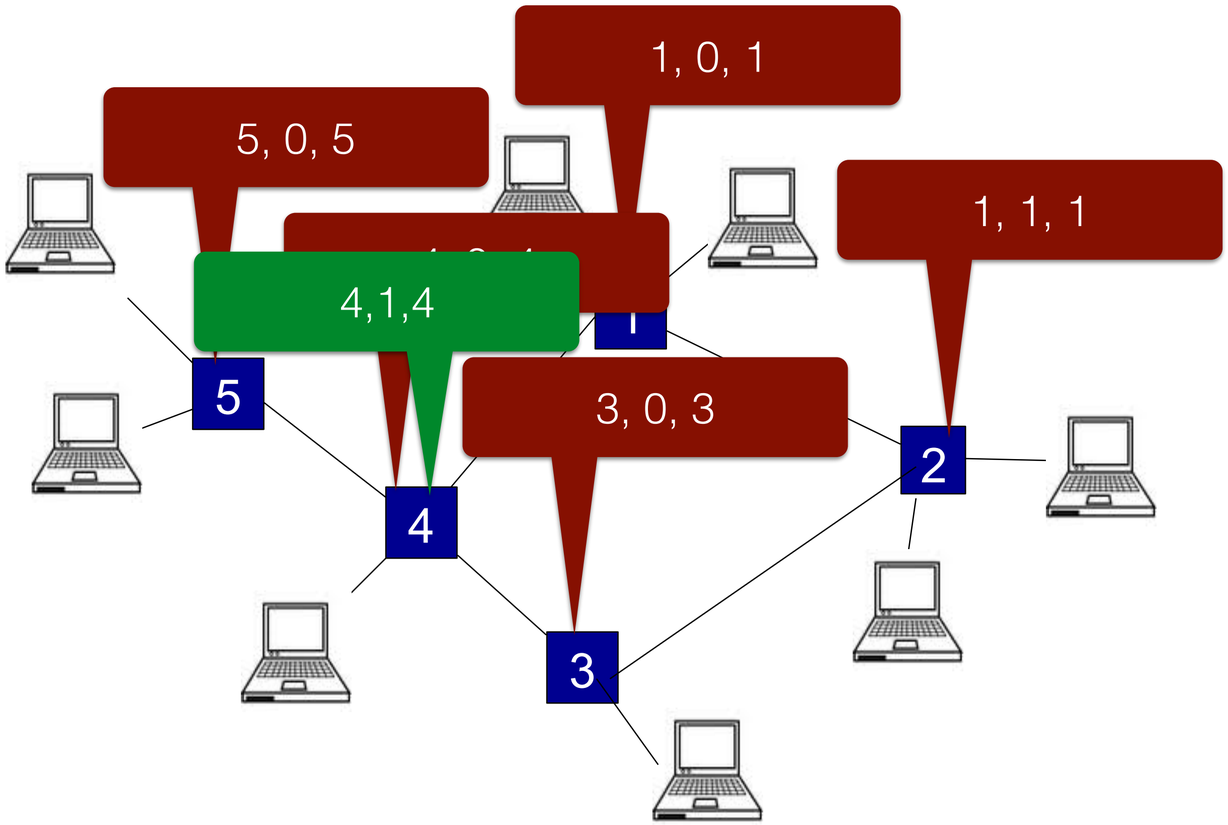


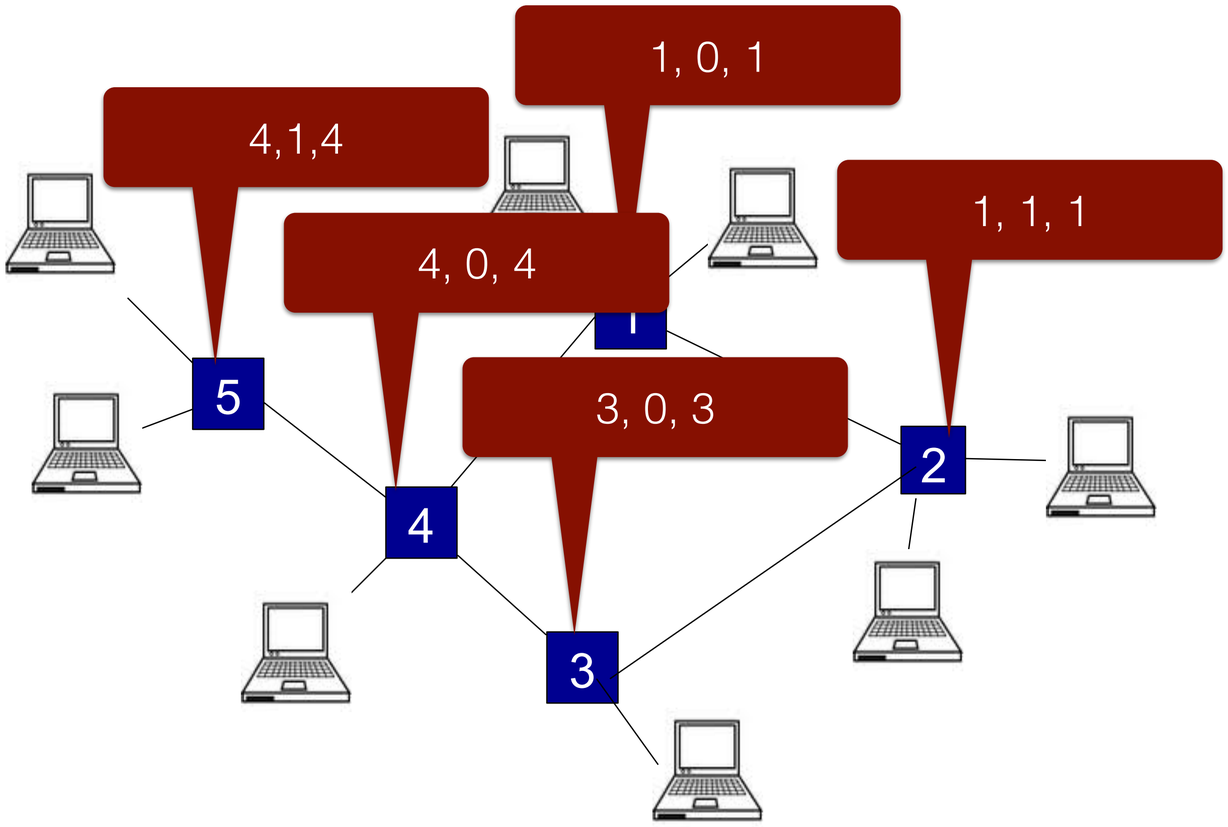
I
should tell
my neighbors about
the change!!

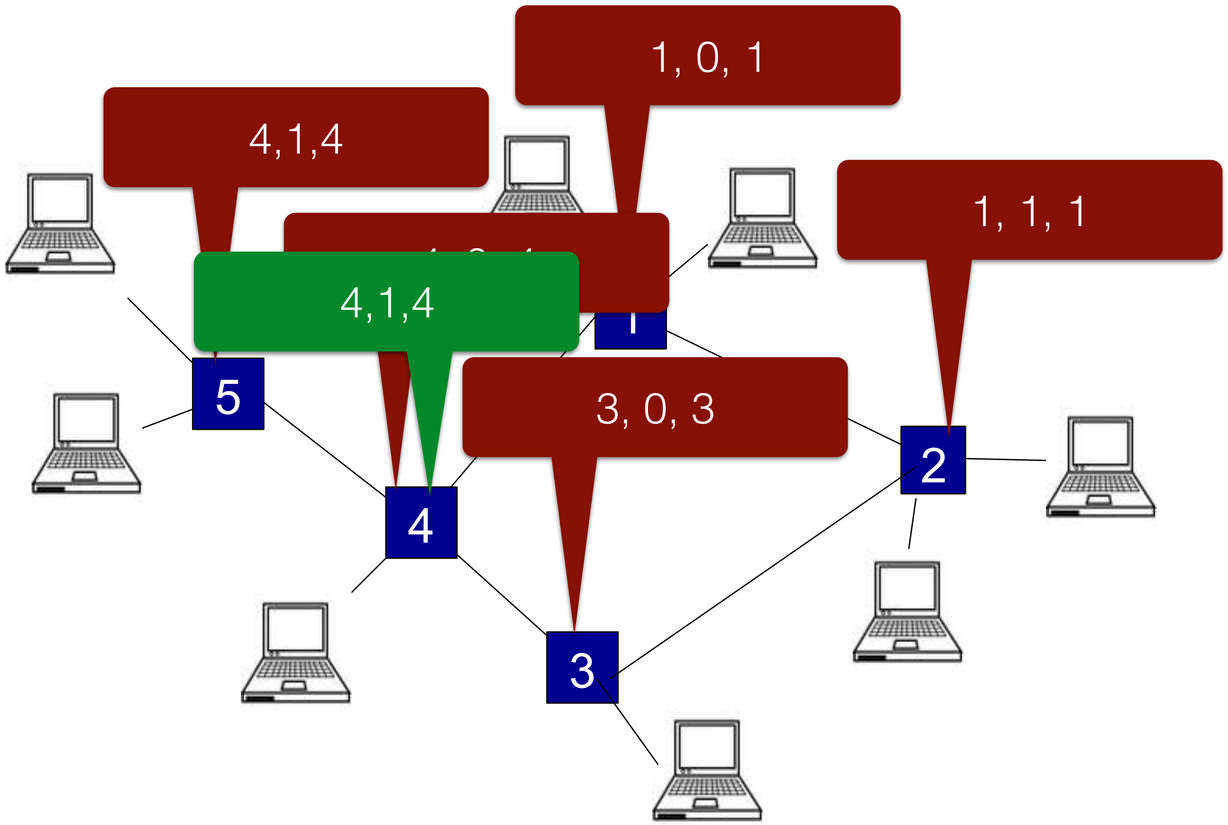

1, 1, 1
1, 2, 2
1, 2, 2
1, 2, 2

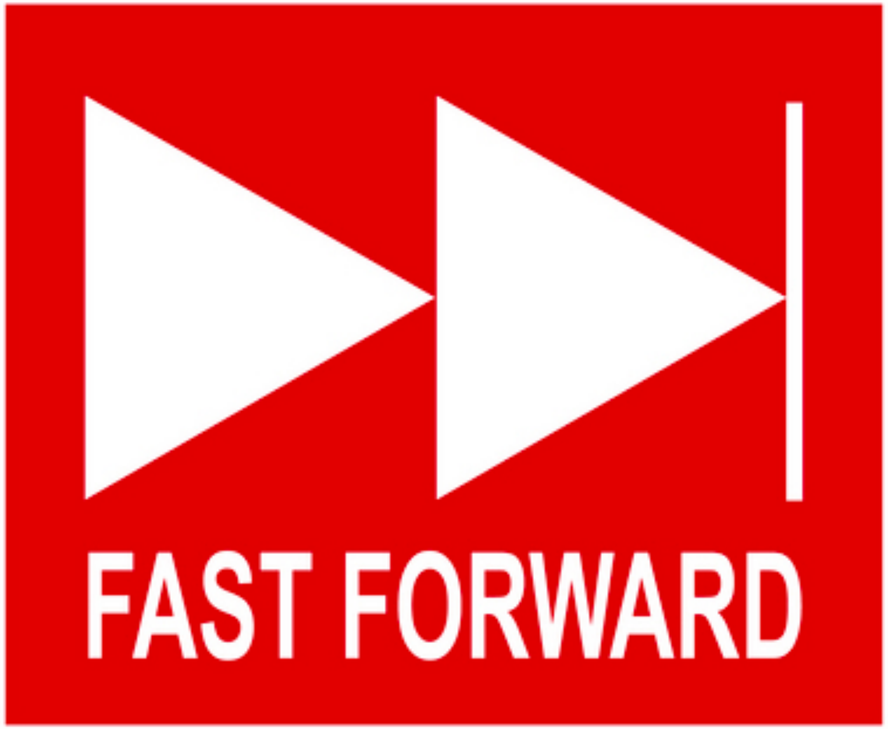


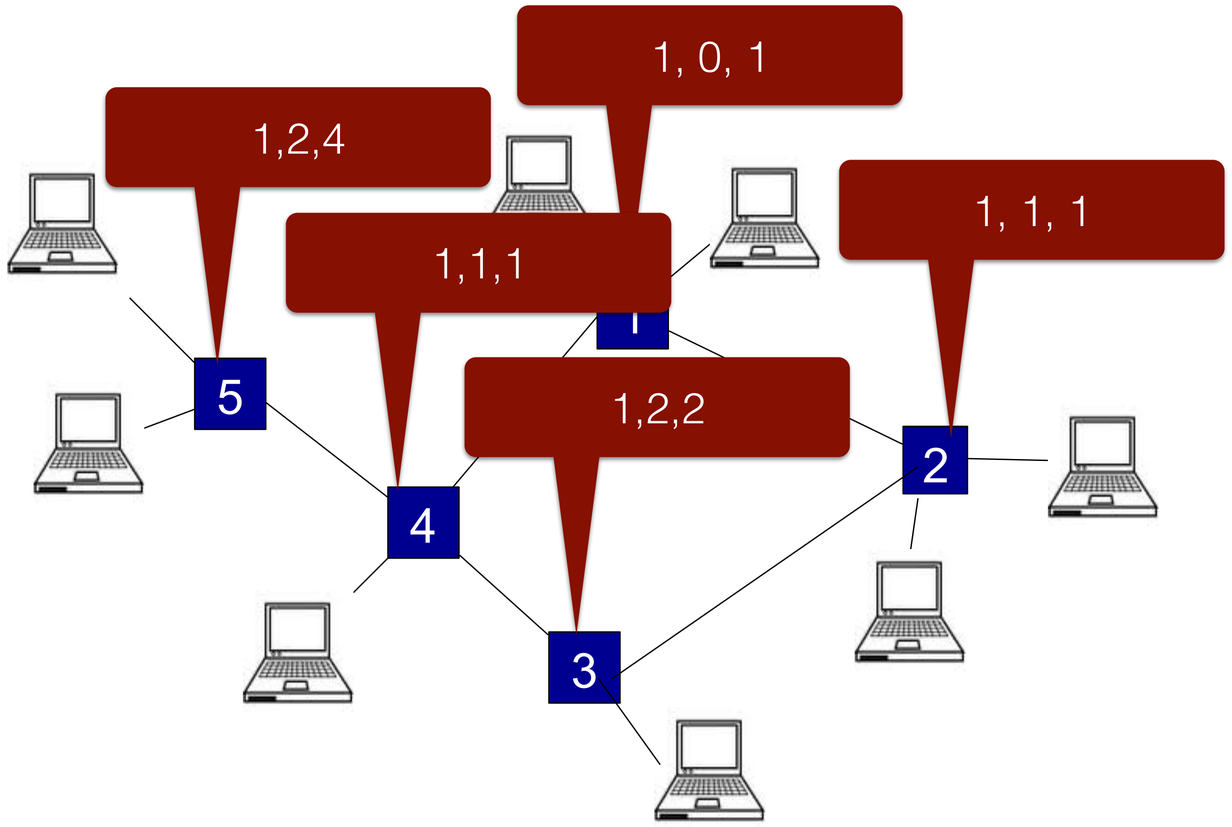
It's hard to predict what order things will happen in: everyone is sending and updating at the same time — the only place it is easy to reason about order is at an individual node!


1, 0, 1
5, 0, 5
1, 1, 1
4,1,4
3, 0, 3
1
2
3
4
5

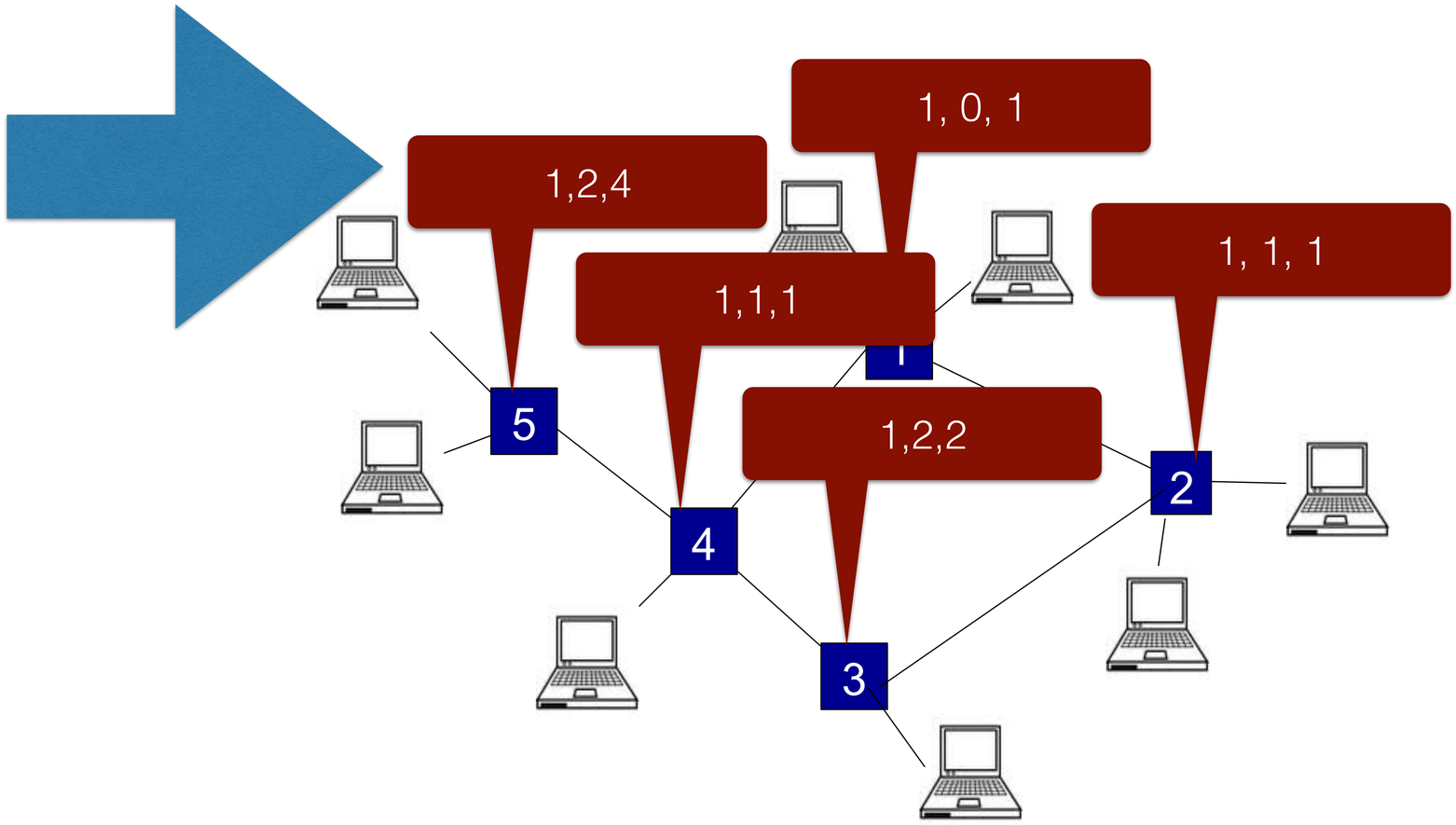

1, 0, 1
4,1,4
1, 1, 1
4, 0, 4
1
5
3, 0, 3
2
4
3


1, 0, 1
4,1,4
1, 1, 1
4,1,4
1
5
3, 0, 3
2
4
3


FAST FORWARD


1, 0, 1
1,2,4
1, 1, 1
1,1,1
1,2,2
1
2
3
4
5

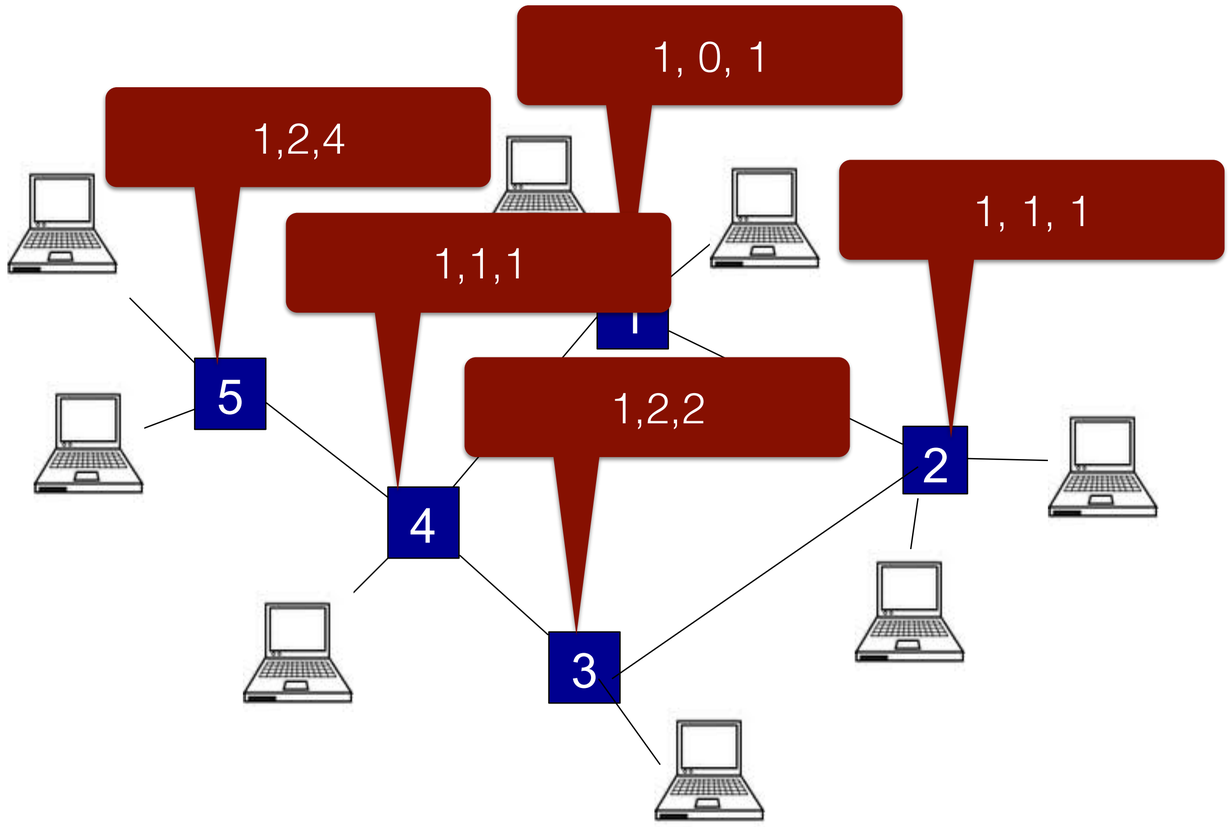
What does this mean?



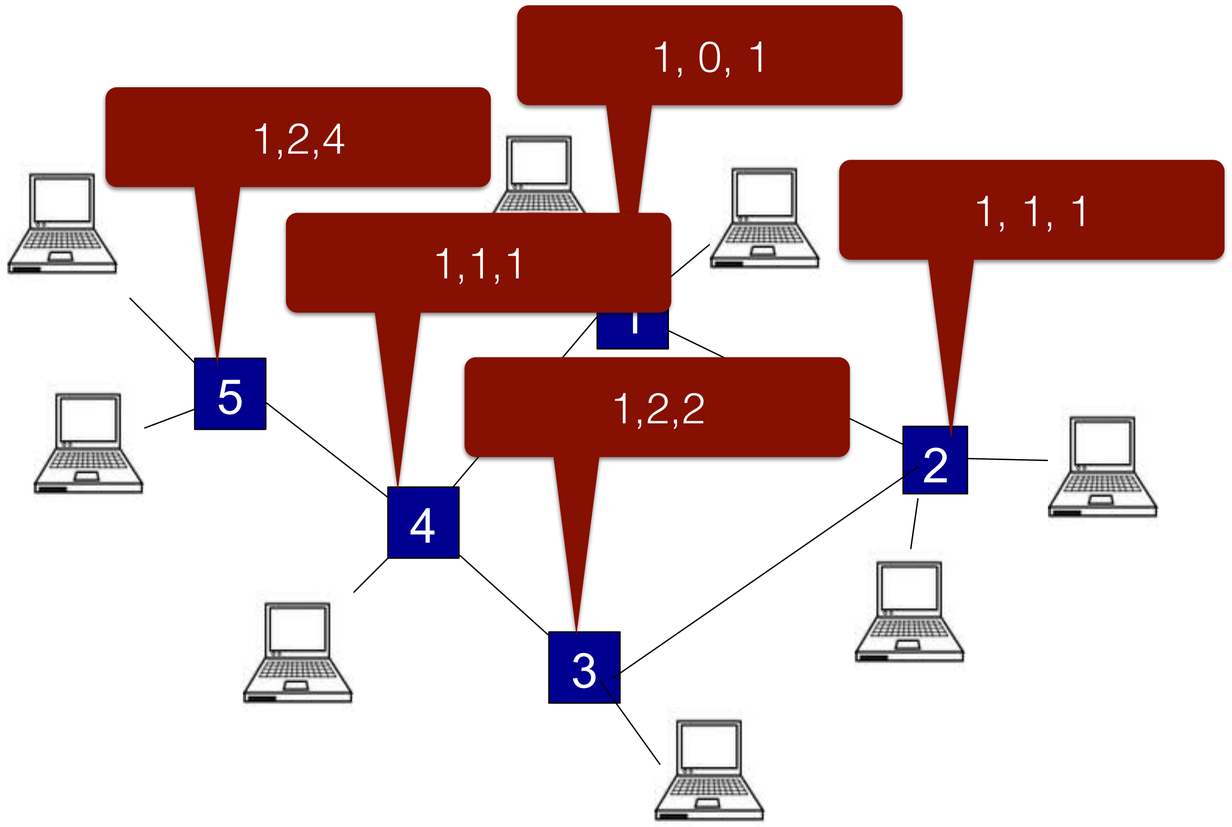
Eventually…

- We stop receiving “new” updates
  - We say that the protocol has *converged.*
- Now: remove all links that don’t connect someone on their path to the root node.

What should I remove?



What should I remove?



Let’s try it now

- Your node ID is your full name.
- Your neighbors are your neighbors.
- Quiet down when you think you have converged.

Who is the root?

Trade-Offs

	Broadcast Network w/ Learning Switches	Broadcast Network w/ Learning Switches and Spanning Tree
Resilience	If there is a route, the packet will reach dest!	Need to recompute spanning tree if failure

Resilience: the ability to provide and maintain an acceptable level of service in the face of faults and challenges to normal operation

Trade-Offs

	Broadcast Network w/ Learning Switches	Broadcast Network w/ Learning Switches and Spanning Tree
Resilience	If there is a route, the packet will reach dest!	Need to recompute spanning tree if failure
Fully Distributed	Yes	Yes

Fully Distributed: does not assume the previous existence of a central coordinator.

Trade-Offs

	Broadcast Network w/ Learning Switches	Broadcast Network w/ Learning Switches and Spanning Tree
Resilience	If there is a route, the packet will reach dest!	Need to recompute spanning tree if failure
Fully Distributed	Yes	Yes
State per Node	Learning Switch: O(#nodes)	Learning Switch: O(#nodes) + Path to Root: O(constant)

State: The amount of memory each node uses

Trade-Offs

	Broadcast Network w/ Learning Switches	Broadcast Network w/ Learning Switches and Spanning Tree
Resilience	If there is a route, the packet will reach dest!	Need to recompute spanning tree if failure
Fully Distributed	Yes	Yes
State per Node	Learning Switch: O(#nodes)	Learning Switch: O(#nodes) + Path to Root: O(constant)
Convergence	No setup time at all!	Need to run spanning tree protocol before routing

We will only talk about convergence in “big picture” terms — but analyzing convergence for routing protocols and other distributed algorithms is a fascinating area of theoretical computer science. If you’re curious about this stuff, take a class from Prof. Haeupler

Convergence: the process of routers/switches agreeing on optimal routes for forwarding packets and thereby completing the updating of their routing table

Trade-Offs

	Broadcast Network w/ Learning Switches	Broadcast Network w/ Learning Switches and Spanning Tree
Resilience	If there is a route, the packet will reach dest!	Need to recompute spanning tree if failure
Fully Distributed	Yes	Yes
State per Node	Learning Switch: O(#nodes)	Learning Switch: O(#nodes) + Path to Root: O(constant)
Convergence	No setup time at all!	Need to run spanning tree protocol before routing
Routing Efficiency	Broadcast Storms	Still sends new connections everywhere.

Do the packets go where they need to get efficiently — without wasting resources at switches?

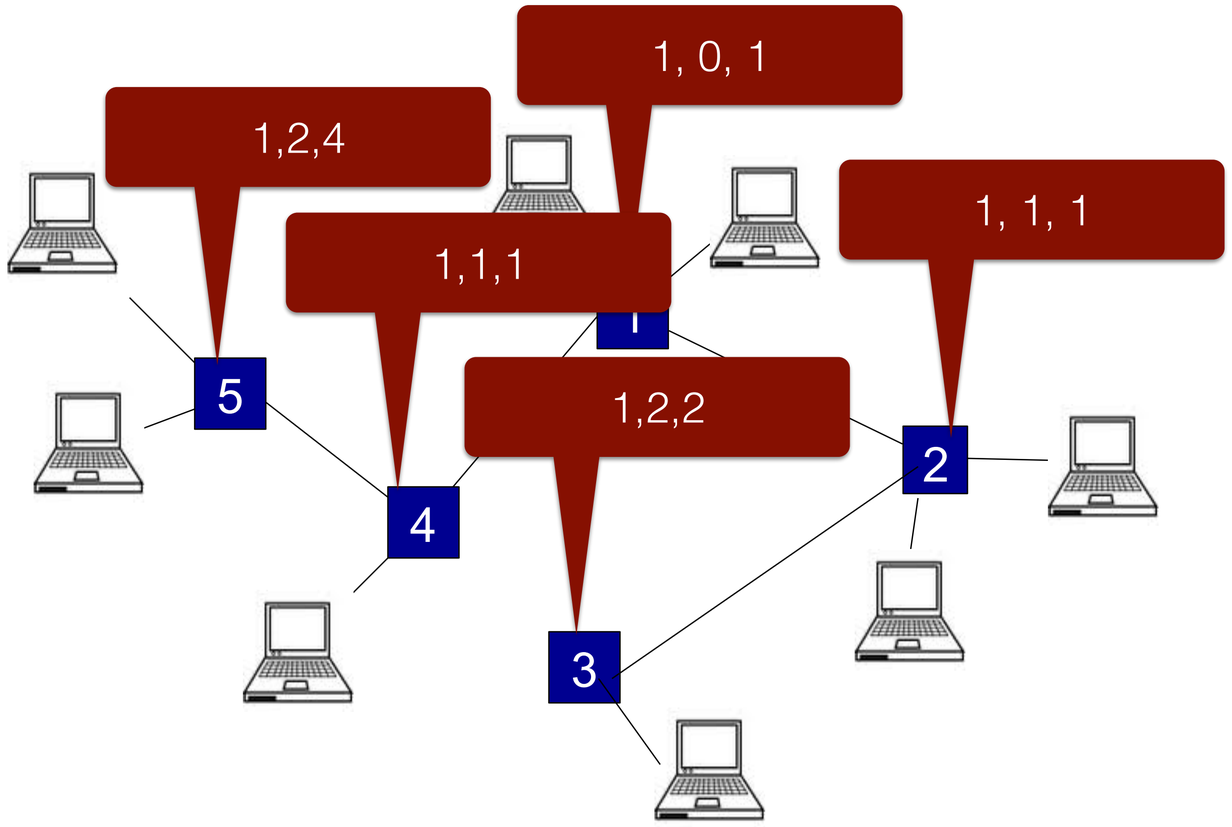
Trade-Offs

	Broadcast Network w/ Learning Switches	Broadcast Network w/ Learning Switches and Spanning Tree
Resilience	If there is a route, the packet will reach dest!	Need to recompute spanning tree if failure
Fully Distributed	Yes	Yes
State per Node	Learning Switch: O(#nodes)	Learning Switch: O(#nodes) + Path to Root: O(constant)
Convergence	No setup time at all!	Need to run spanning tree protocol before routing
Routing Efficiency	Broadcast Storms	Still sends new connections everywhere.
Shortest Path?	Not Necessarily…	Not Necessarily…

We know packets will reach their destination… but do they take the shortest path to get there?

What if 3 wants to communicate with 4?

What if 5 wants to communicate with 3?



Time check…

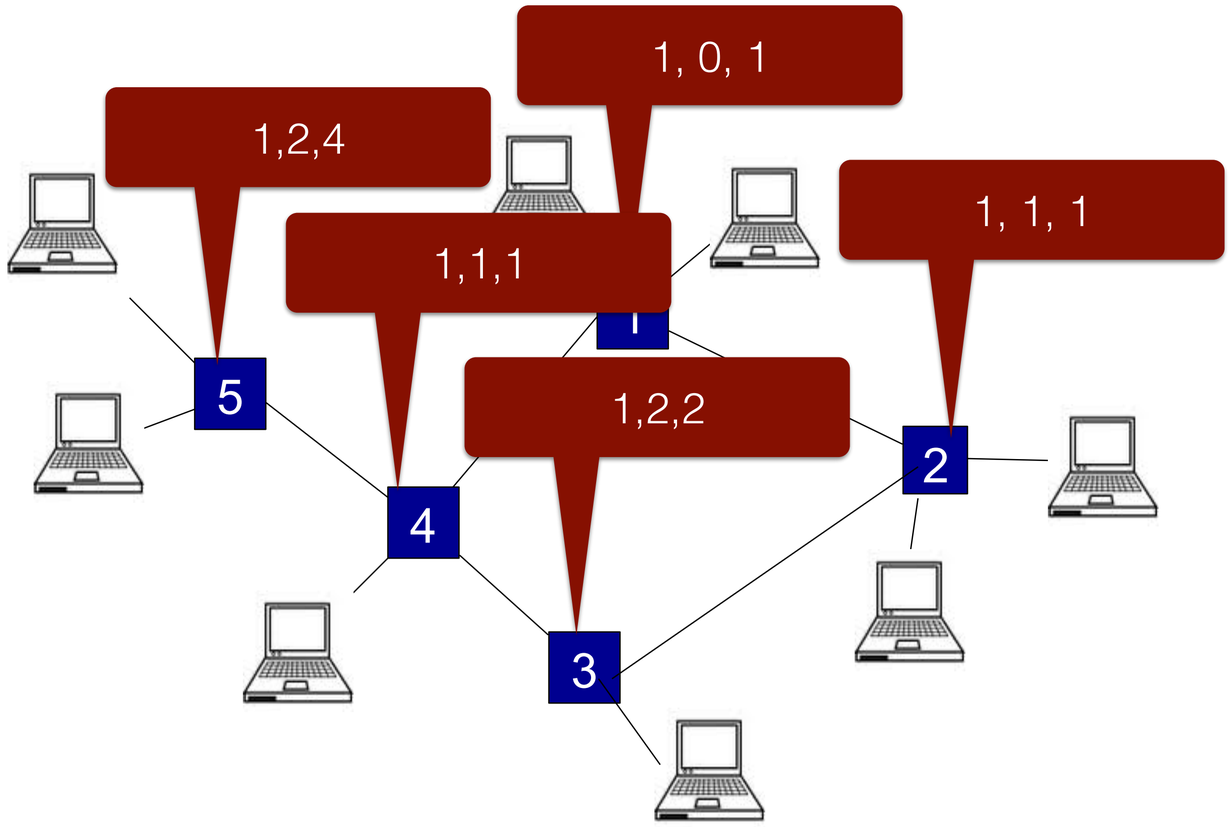
Real World

- We only use broadcast routing in very small networks.
  - One rack in the machine room.
  - A wing of one floor in GHC.
- To orchestrate the bigger network — across campus — we use other algorithms.
  - Why do you think that is?

	Broadcast Network w/ Learning Switches	Broadcast Network w/ Learning Switches and Spanning Tree	Distance Vector e.g RIP
Resilience	If there is a route, the packet will reach dest!	Need to recompute spanning tree if failure	
Fully Distributed	Yes	Yes	
State per Node	Learning Switch: O(#nodes)	Learning Switch: O(#nodes) + Path to Root: O(constant)	
Convergence	No setup time at all!	Need to run spanning tree protocol before routing	
Routing Efficiency	Broadcast Storms	Still sends new connections everywhere.	Packets sent directly to their destination.
Shortest Path?	Not Necessarily…	Not Necessarily…	Yes

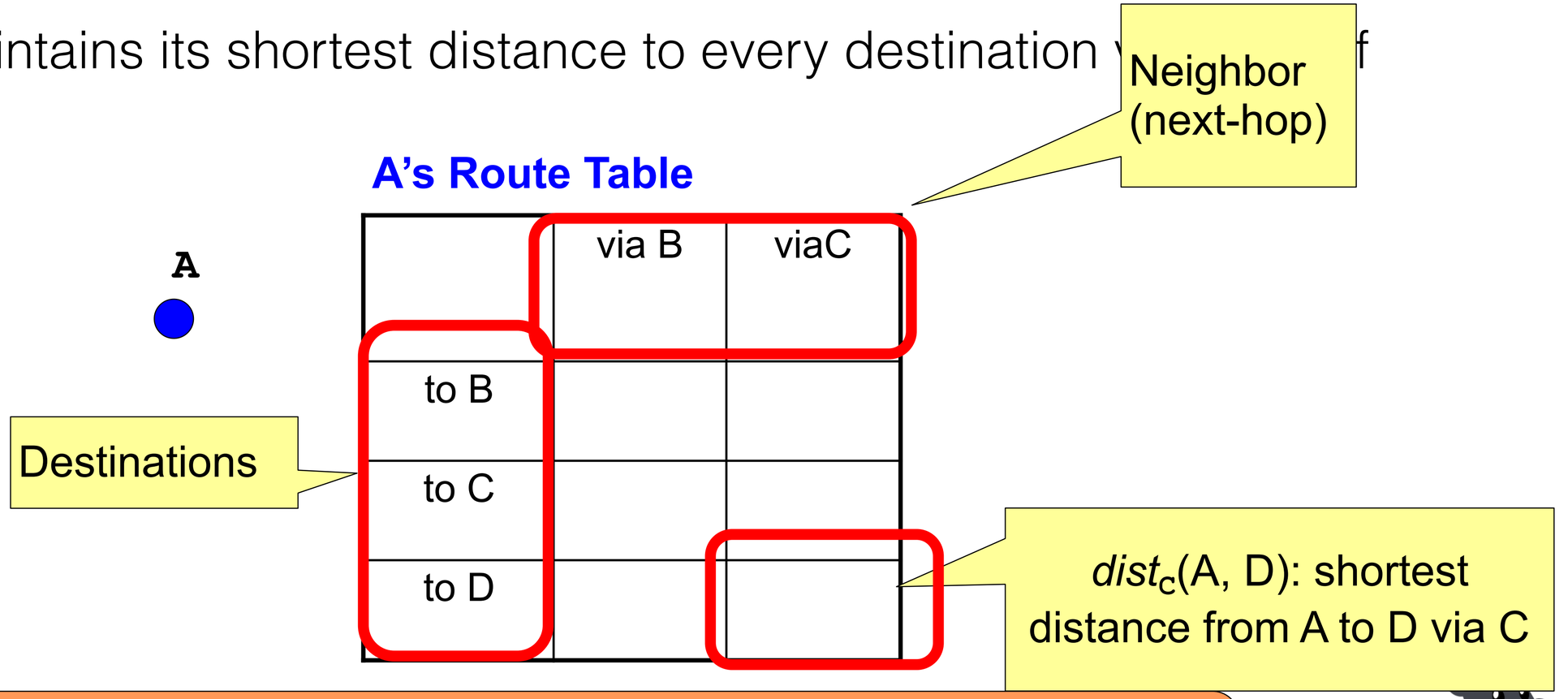
Recall: Spanning Tree

There is exactly one node that every *does* have the shortest path to.



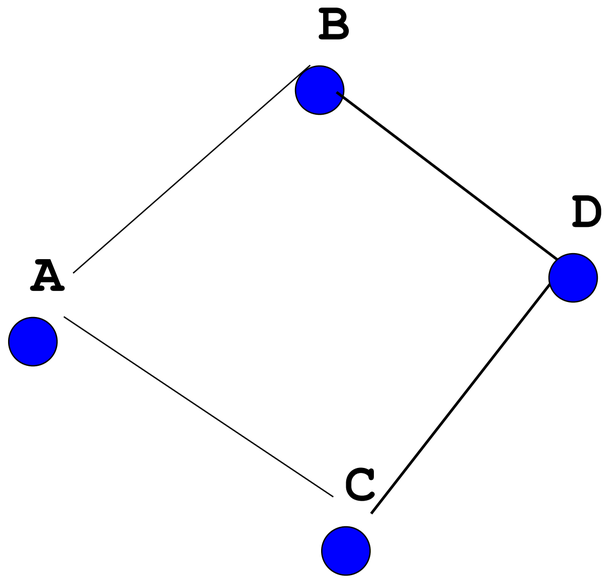
How Distance-Vector (DV) Works

Each router maintains its shortest distance to every destination v f its neighbors



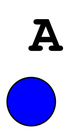
Each router computes its shortest distance to every destination via <u>any</u> of its neighbors

How Distance-Vector (DV) Works



How Distance-Vector (DV) Works

Each router maintains its shortest distance to every destination via each of its neighbors

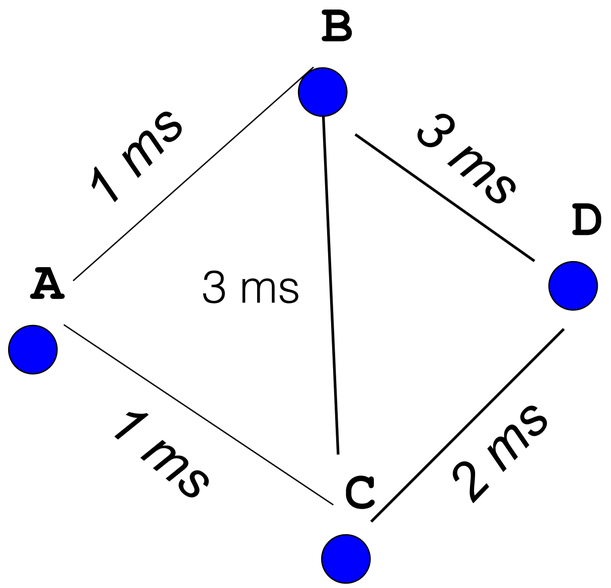


A's Route Table

	via B	viaC
to B	1	
to C		1
to D		

Each router computes its shortest distance to every destination via <u>any</u> of its neighbors

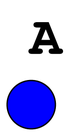
How Distance-Vector (DV) Works



Link distance doesn’t have to be 1! Could be some other value — e.g., latency of the link

How Distance-Vector (DV) Works

Each router maintains its shortest distance to every destination via each of its neighbors



A's Route Table

	via B	viaC
to B	1	4
to C	4	1
to D	4	3

Each router computes its shortest distance to every destination via <u>any</u> of its neighbors

How Distance-Vector (DV) Works

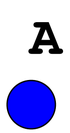
Routers send a summary of their tables to their neighbors.

This summary is called a “distance vector”

A’s distance vector (DV)

	min dist
to A	
to B	
to C	
to D	

A’s Route Table



	via B	viaC
to B	1	4
to C	2	1
to D	4	3

Each router computes its shortest distance to every destination via <u>any</u> of its neighbors

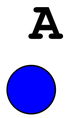
How Distance-Vector (DV) Works

A's distance vector (DV)

	min dist
to A	0
to B	
to C	
to D	

A's Route Table

	via B	viaC
to B	1	4
to C	2	1
to D	4	3



Update route to min(all of my B routes)

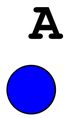
How Distance-Vector (DV) Works

A's distance vector (DV)

	min dist
to A	0
to B	1
to C	1
to D	

A's Route Table

	via B	viaC
to B	1	4
to C	2	1
to D	4	3



Update route to min(all of my C routes)

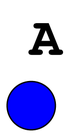
How Distance-Vector (DV) Works

A's distance vector (DV)

	min dist
to A	0
to B	1
to C	1
to D	3

A's Route Table

	via B	viaC
to B	1	4
to C	2	1
to D	4	3

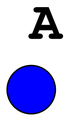


Update route to min(all of my D routes)

How Distance-Vector (DV) Works

B's DV

A's Route Table



	via B	viaC
to B	1	∞
to C	∞	1
to D	∞	∞

But, when we start the table is mostly empty…

We have to learn by receiving DV's from others.

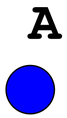
How Distance-Vector (DV) Works

B's DV

	mindist
to A	1
to C	3
to D	2

A's Route Table

	via B	viaC
to B	1	∞
to C	∞	1
to D	∞	∞



But, when we start the table is mostly empty…
We have to learn by receiving DV's from others.

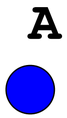
How Distance-Vector (DV) Works

B's DV

	mindist
to A	1
to C	3
to D	2

A's Route Table

	via B	viaC
to B	1	∞
to C	4	1
to D	3	∞



But, when we start the table is mostly empty…
We have to learn by receiving DV's from others.

Distance Vector Routing: Summary

- Each router knows the links to its neighbors
- Each router has provisional “shortest path” to every other router -- its distance vector (DV)
- Routers exchange this DV with their neighbors
- Routers look over the set of options offered by their neighbors and select the best one
- Iterative process converges to set of shortest paths

Tricky Question

- Let's assume our DV algorithm runs in "rounds"
  - In lock-step, all routers send out a DV to their neighbors
  - Then they update their tables — all at the same time! — with the new information they have received.
  - Then, in lock-step, they all send out a DV at the same time. (Repeat)
- **Q: How many "rounds" will it take for the DV algorithm to *converge*?**

Intuition

- Initial state: best one-hop paths
- One simultaneous round: best two-hop paths
- Two simultaneous rounds: best three-hop paths
- …
- Kth simultaneous round: best (k+1) hop paths
- Must eventually converge
  - as soon as it reaches longest best path

	Broadcast Network w/ Learning Switches	Broadcast Network w/ Learning Switches and Spanning Tree	Distance Vector e.g RIP
Resilience	If there is a route, the packet will reach dest!	Need to recompute spanning tree if failure	
Fully Distributed	Yes	Yes	
State per Node	Learning Switch: O(#nodes)	Learning Switch: O(#nodes) + Path to Root: O(constant)	
Convergence	No setup time at all!	Need to run spanning tree protocol before routing	
Routing Efficiency	Broadcast Storms	Still sends new connections everywhere.	Packets sent directly to their destination.
Shortest Path?	Not Necessarily…	Not Necessarily…	Yes

	Broadcast Network w/ Learning Switches	Broadcast Network w/ Learning Switches and Spanning Tree	Distance Vector e.g RIP
Resilience	If there is a route, the packet will reach dest!	Need to recompute spanning tree if failure	
Fully Distributed	Yes	Yes	
State per Node	Learning Switch: O(#nodes)	Learning Switch: O(#nodes) + Path to Root: O(constant)	
Convergence	No setup time at all!	Need to run spanning tree protocol before routing	Need to run DV before routing — takes length of longest best path time.
Routing Efficiency	Broadcast Storms	Still sends new connections everywhere.	Packets sent directly to their destination.
Shortest Path?	Not Necessarily…	Not Necessarily…	Yes

	Broadcast Network w/ Learning Switches	Broadcast Network w/ Learning Switches and Spanning Tree	Distance Vector e.g RIP
Resilience	If there is a route, the packet will reach dest!	Need to recompute spanning tree if failure	
Fully Distributed	Yes	Yes	
State per Node	Learning Switch: O(#nodes)	Learning Switch: O(#nodes) + Path to Root: O(constant)	O(# switches * max node degree)
Convergence	No setup time at all!	Need to run spanning tree protocol before routing	Need to run DV before routing — takes length of longest best path time.
Routing Efficiency	Broadcast Storms	Still sends new connections everywhere.	Packets sent directly to their destination.
Shortest Path?	Not Necessarily…	Not Necessarily…	Yes

	Broadcast Network w/ Learning Switches	Broadcast Network w/ Learning Switches and Spanning Tree	Distance Vector e.g RIP
Resilience	If there is a route, the packet will reach dest!	Need to recompute spanning tree if failure	
Fully Distributed	Yes	Yes	Yes
State per Node	Learning Switch: O(#nodes)	Learning Switch: O(#nodes) + Path to Root: O(constant)	O(# switches * max node degree)
Convergence	No setup time at all!	Need to run spanning tree protocol before routing	Need to run DV before routing — takes length of longest best path time.
Routing Efficiency	Broadcast Storms	Still sends new connections everywhere.	Packets sent directly to their destination.
Shortest Path?	Not Necessarily…	Not Necessarily…	Yes

	Broadcast Network w/ Learning Switches	Broadcast Network w/ Learning Switches and Spanning Tree	Distance Vector e.g RIP
Resilience	If there is a route, the packet will reach dest!	Need to recompute spanning tree if failure	**I have some bad news.**
Fully Distributed	Yes	Yes	Yes
State per Node	Learning Switch: O(#nodes)	Learning Switch: O(#nodes) + Path to Root: O(constant)	O(# switches * max node degree)
Convergence	No setup time at all!	Need to run spanning tree protocol before routing	Need to run DV before routing — takes length of longest best path time.
Routing Efficiency	Broadcast Storms	Still sends new connections everywhere.	Packets sent directly to their destination.
Shortest Path?	Not Necessarily…	Not Necessarily…	Yes

Stay Tuned

- We will come back to this on Tuesday.
- Recitations: TOMORROW!
  - You can go to any section, even if you are not registered for that section.
  - You are not required to attend, but are highly recommended to do so.
  - ***Reminder: homework is out!***
  - ALL COURSE INFO IS AT www.myheartisinthenetwork.com

On Your Way Our

- Anonymous feedback cards
  - Technical questions
  - Questions about the course
  - Feedback & recommendations
    - RECOMMENDATIONS!