Putting The Pieces Together

Ethernet, DHCP, ARP, etc.

Today in Internet history...

- April 14, 1998 (22 years ago)…
- Netflix website launched!
- 925 movies
- .. mailed to you on DVD; no streaming until 2007 (nine years later)
- Pay-per-movie; subscription started the following year
- A year after that, it offers itself to Blockbuster for $50 million
  - .. Blockbuster probably should have taken them up on that
  - 2019 Netflix: $20 billion gross, $1.86 billion net
  - 2020 Blockbuster: One remaining store in Bend, Oregon… maybe? 😕

Putting The Pieces Together

Ethernet, DHCP, ARP, etc.

In the past…

- We've talked a lot about L3; specifically IP!
  - Common routing
    - Intradomain (D-V and L-S)
    - Interdomain (BGP)
  - Addresses
    - Structure, properties (CIDR, aggregatable, etc.)

- We've talked some about L2; mostly Ethernet
  - Common routing
    - L-S
    - Learning switches and STP
  - Addresses
    - … ?

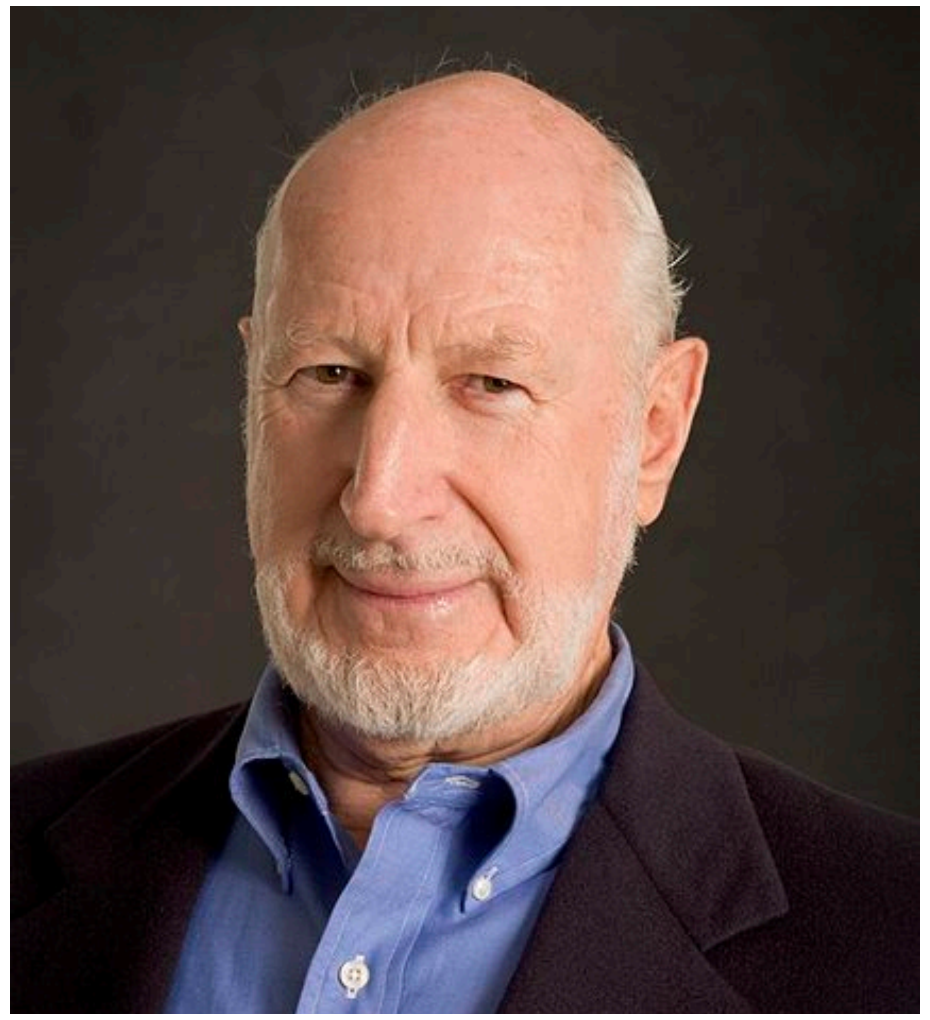
Today…

- Fill in some gaps!
  - Bias towards Ethernet (L2) and IPv4 (L3)
  - Generally similarities with other L2/L3 (e.g., WiFi and IPv6)
- Ethernet
  - History and background:
    - Multiple access, ALOHA, CSMA, CSMA/CD, and exponential backoff
  - Addresses, broadcast and multicast service types
  - Modern Ethernet
- How do L2 and L3 really fit together?
  - Routing
  - Addresses (ARP)
  - How does a host know its own IP address? (DHCP)
- Example — all together now!
- Bonus topic: Network Address Translation (NAT)

Digging into Ethernet

Our L2 technology of choice

ALOHA



- In 1968, Norman Abramson had a problem at the University of Hawaii…
  - How to allow people on the other islands access to the U of H computer?

- His solution: ALOHAnet
  - Additive Links On-line Hawaii Area
  - Wireless connection from terminals on the other islands!
  - *Hugely* influential!

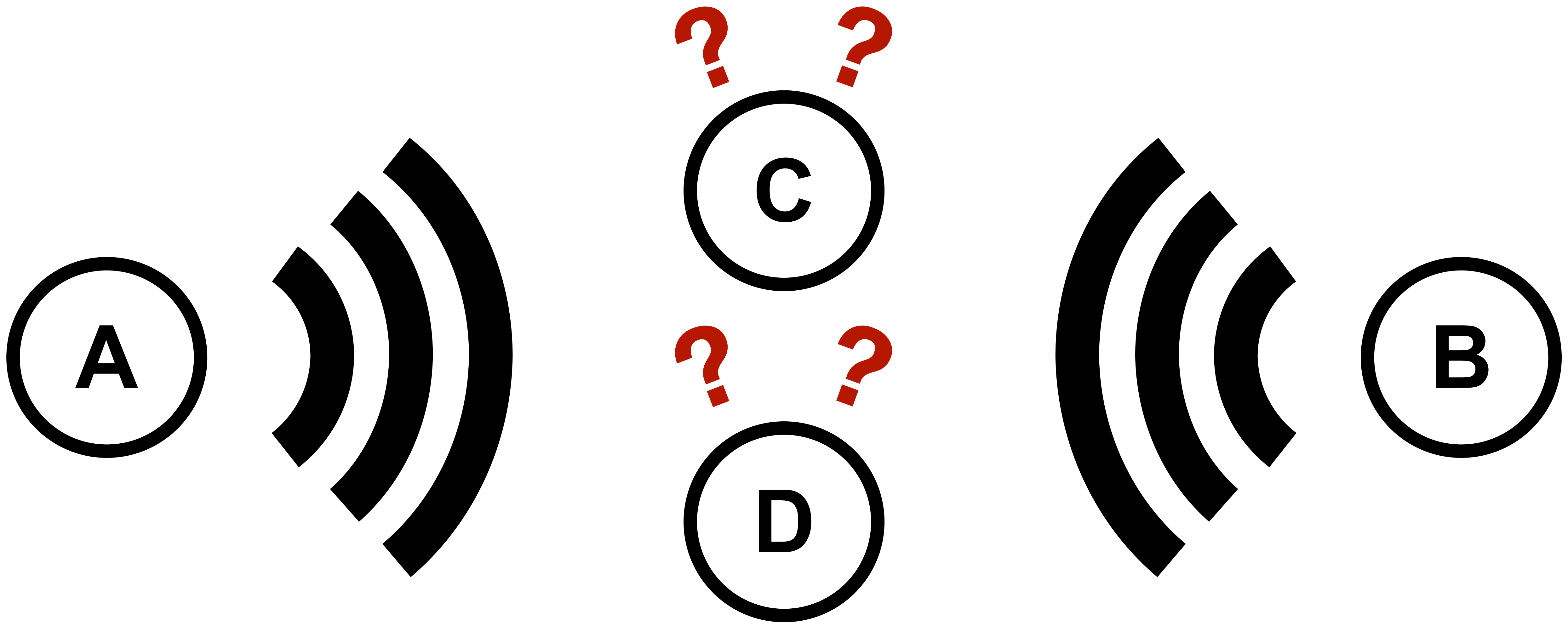
  - We'll return to ALOHA; first let's talk about *shared media*
    (And no, I don't mean BitTorrenting the full works of Abba)

Shared Media

- In a radio network, nodes utilize a *shared medium* (electromagnetic spectrum in some locality)

- Transmissions from different nodes may interfere or *collide* with each other!

- We need a system for allocating the medium to everyone wanting to use it
  - .. a *multiple access protocol*



Common Multiple Access Protocol approaches

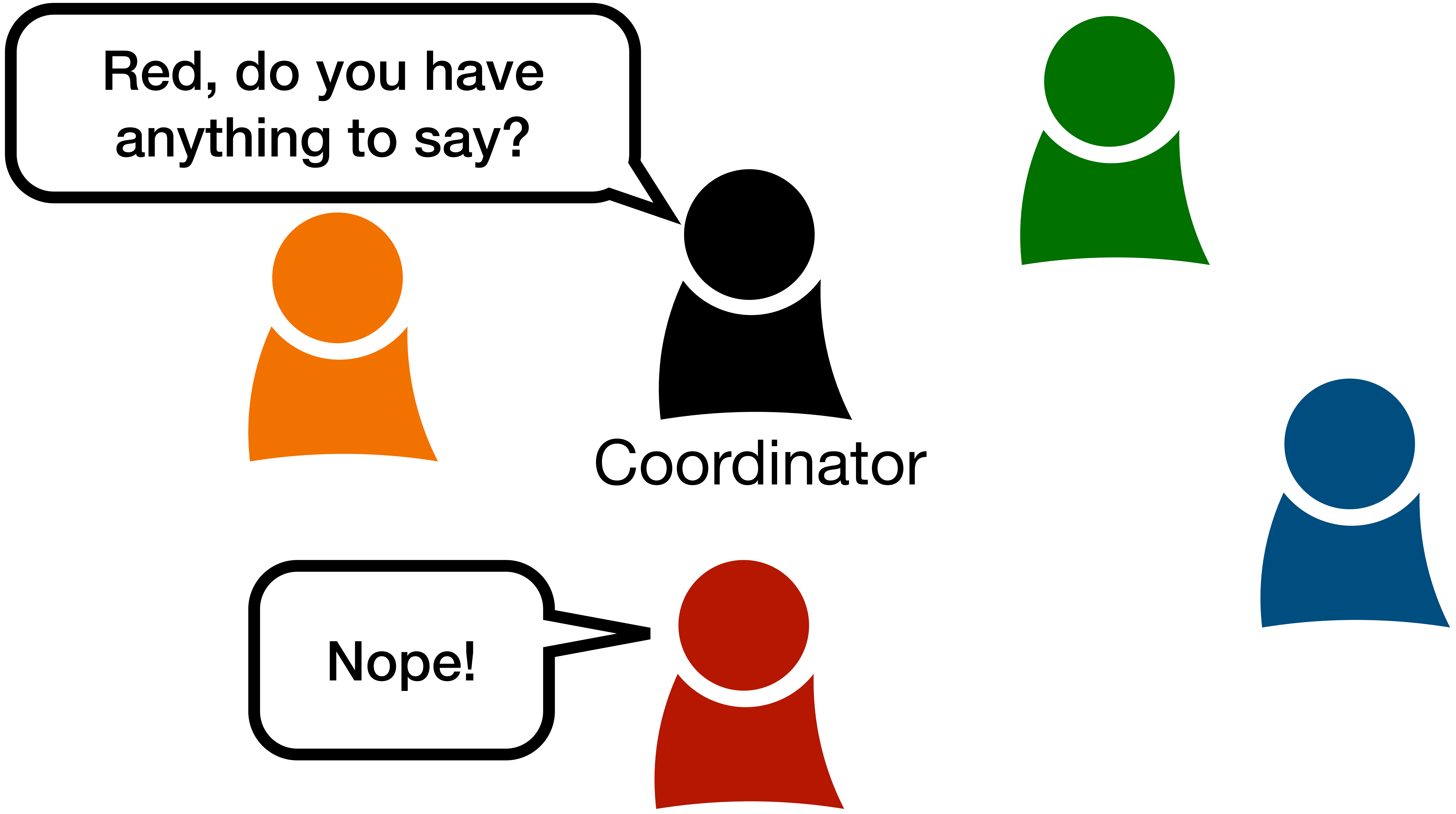
- **Divide medium up by frequency** (*Frequency Division Multiplexing*)
  - Can be wasteful! Only so much EM spectrum to go around, and many frequencies likely to be idle often (traffic is bursty)

Partitioning approaches

- **Divide medium up by time** (several ways)
  - Divide time into fixed-sized “slots”, each sender gets their own slot (*Time Division Multiplexing*); same drawback as FDM
  - Take turns…

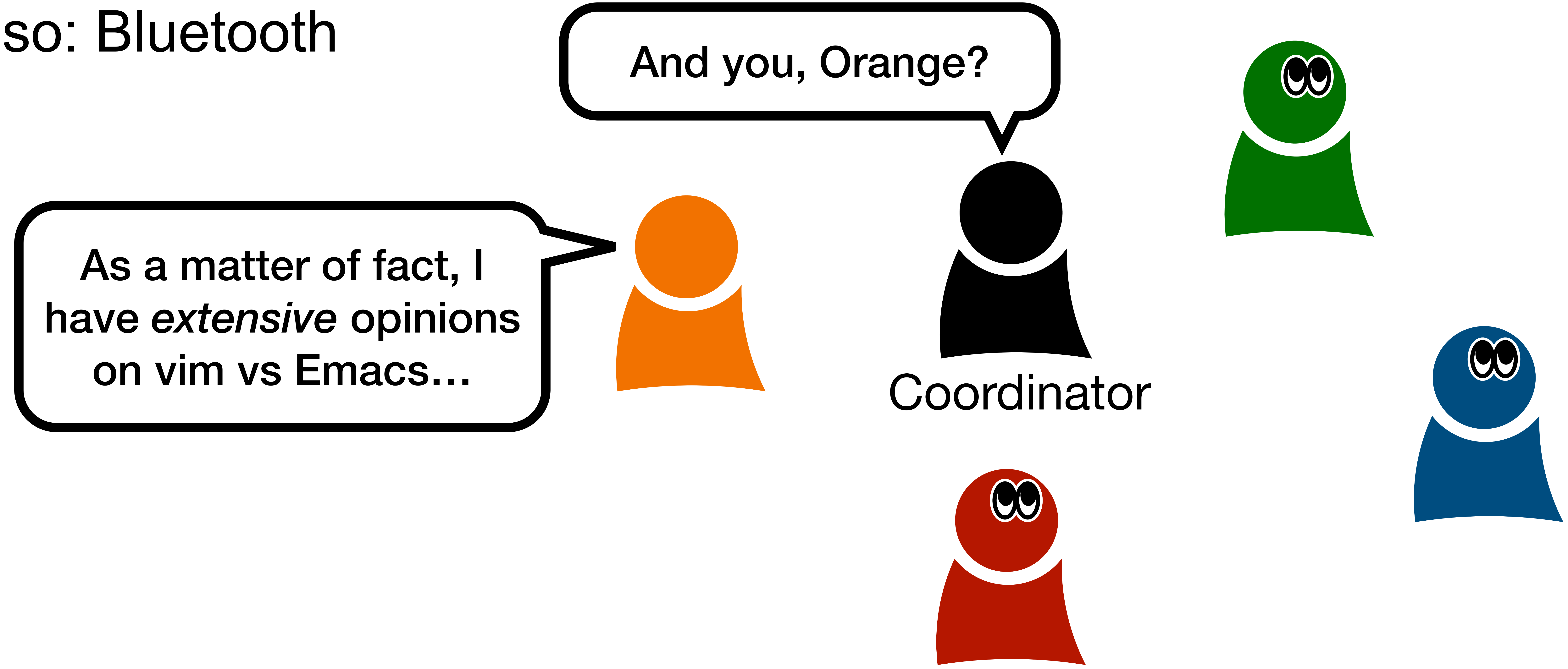
Turn-Taking schemes

- Polling protocols
  - A coordinator decides who gets to speak when
  - Like congress: “The Chair recognizes the Senator from California…”
  - Also: Bluetooth



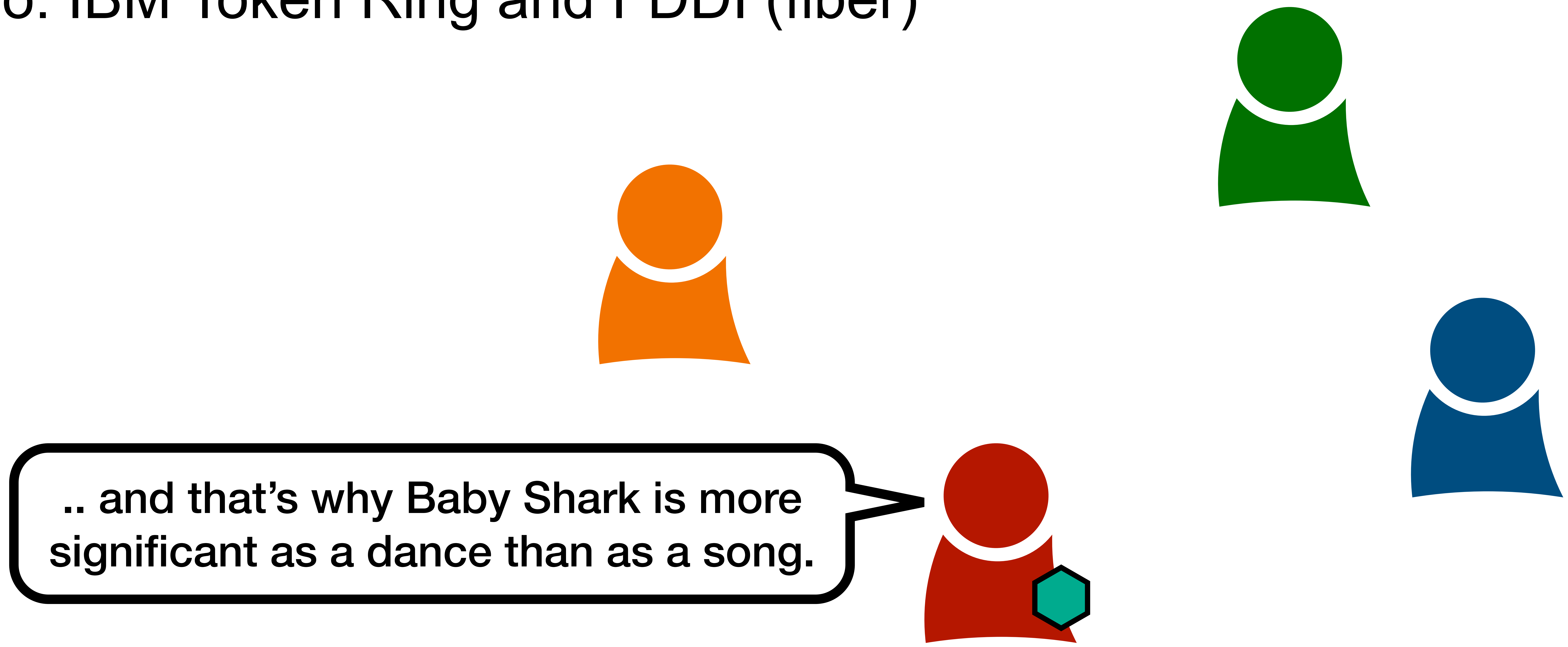
Turn-Taking schemes

- Polling protocols
  - A coordinator decides who gets to speak when
  - Like congress: “The Chair recognizes the Senator from California…”
  - Also: Bluetooth



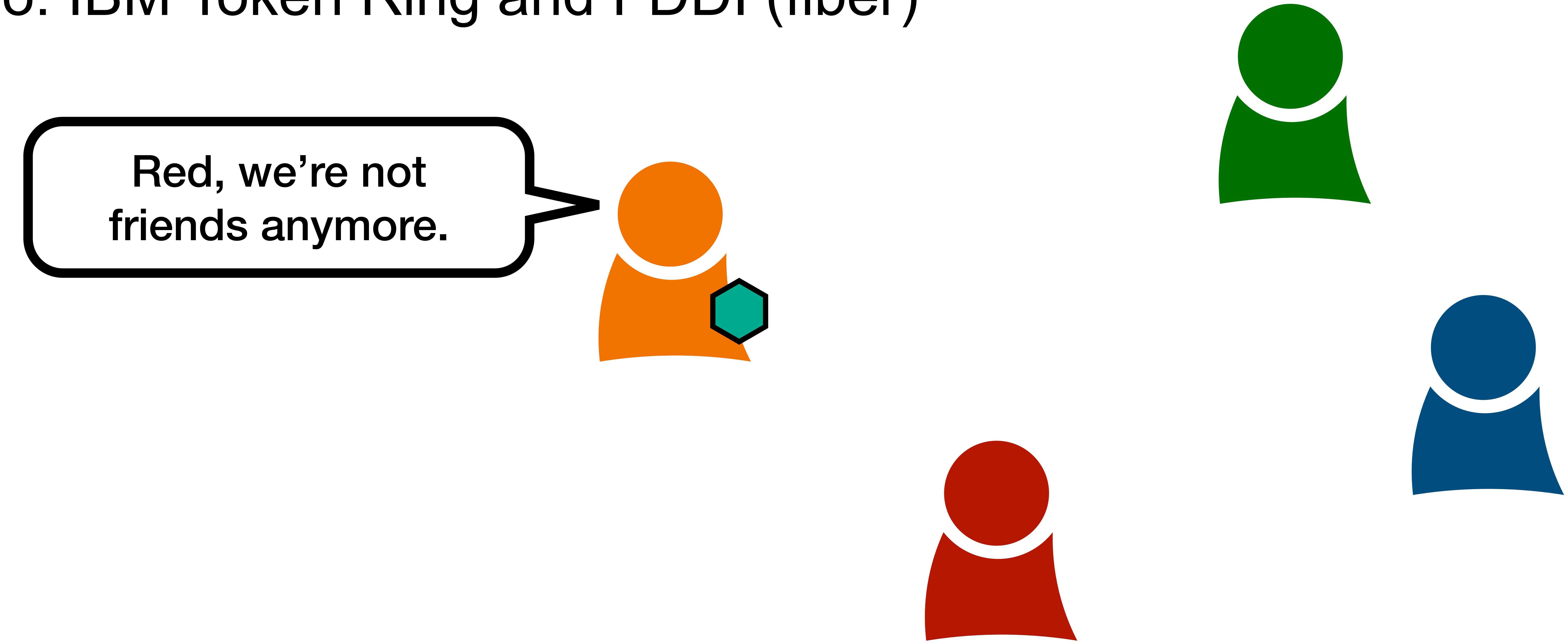
Turn-Taking schemes

- Token-passing
  - Virtual “token” passed around, only holder can transmit
  - Like a “talking stick”
  - Also: IBM Token Ring and FDDI (fiber)



Turn-Taking schemes

- Token-passing
  - Virtual "token" passed around, only holder can transmit
  - Like a "talking stick"
  - Also: IBM Token Ring and FDDI (fiber)



Common Multiple Access Protocol approaches

- **Divide medium up by frequency** (*Frequency Division Multiplexing*)
  - Can be wasteful! Only so much EM spectrum to go around, and many frequencies likely to be idle often (traffic is bursty)

Partitioning approaches

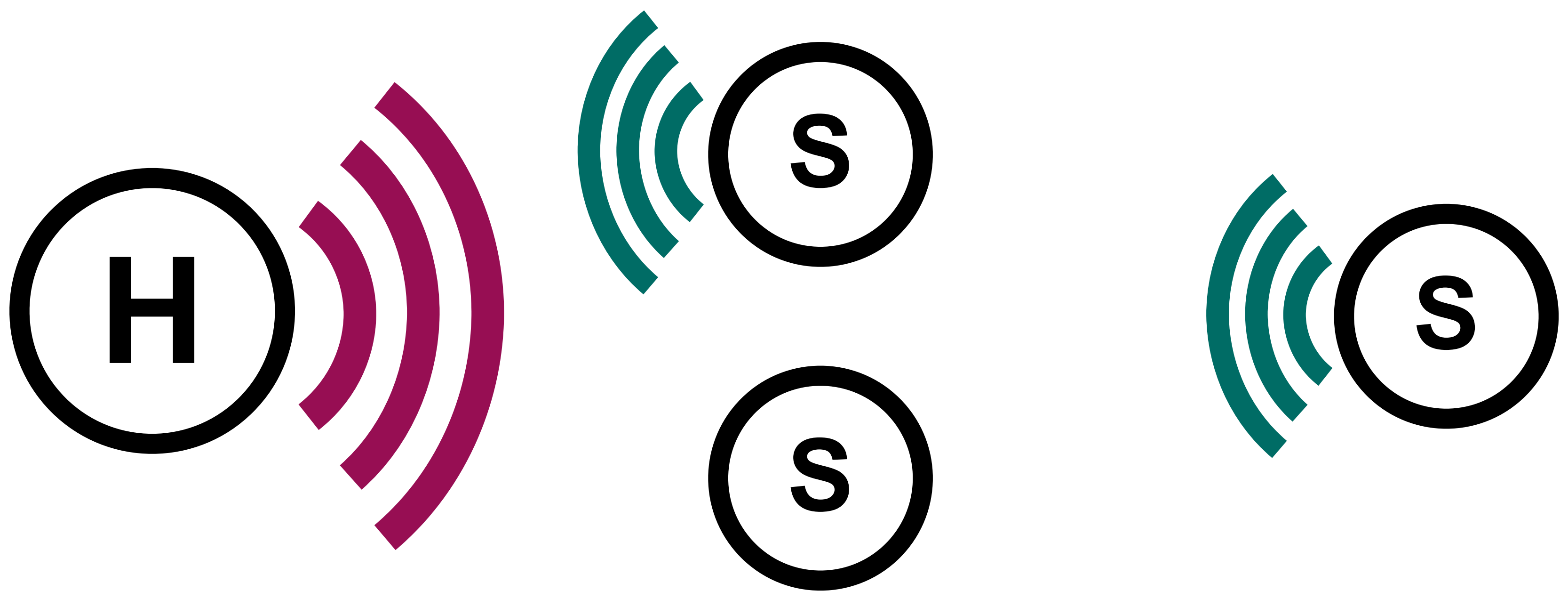
- **Divide medium up by time** (several ways)
  - Divide time into fixed-sized "slots", each sender gets their own slot (*Time Division Multiplexing*); same drawback as FDM
  - Take turns
    - e.g., by *polling* or *token-passing*
  - Random access
    - Introduced by *ALOHA*
    - Also used by *CSMA, CSMA/CD, ….*

ALOHAnet: Context

- "Hub" node on Oahu
- "Remote" nodes across Hawaii

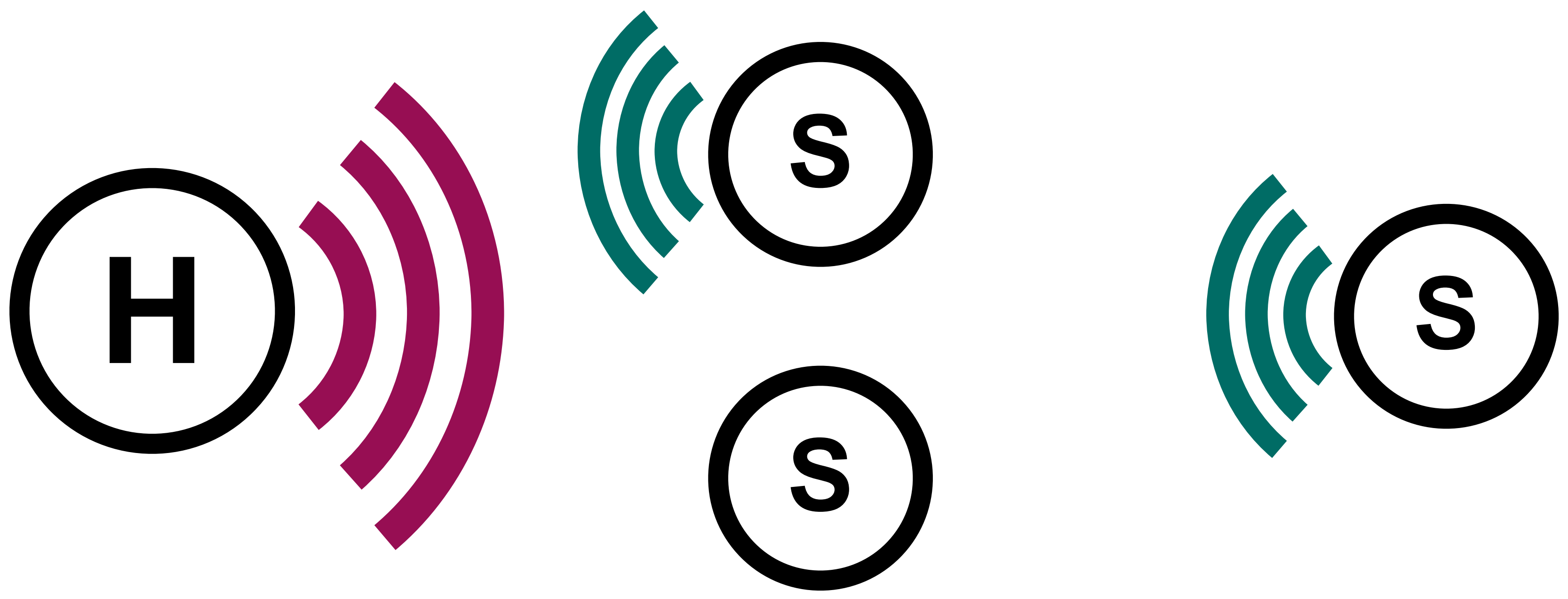
- Two frequencies:
  - Hub transmits on its own frequency
    - Only one sender — no collisions
    - (All remotes listen to it)

  - All remote sites transmit on shared frequency
    - May collide — use random access scheme
    - (Only hub listens to it)



ALOHAnet: "Pure ALOHA" random access scheme

- If remote has a packet — just send it
  - No *a priori* coordination among remote sites



- When hub gets a packet — send ack

- If two remote sites transmitted at once, collision will have garbled packet…
  - .. hub will not send an ack!

- If remote does not get expected ack…
  - Wait *a random amount of time*
  - Then resend — probably won't collide this time!

  - .. it's so simple!

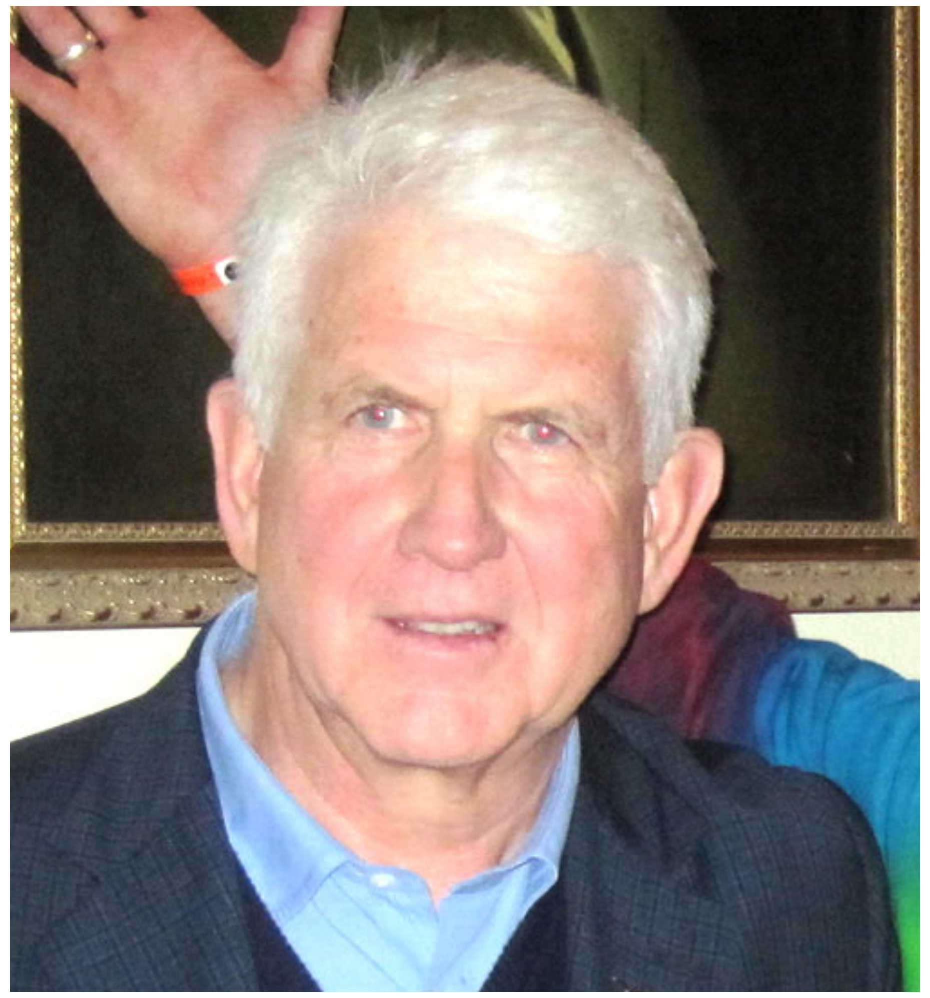
That’s all great, but…

aren’t we supposed to be talking about

Ethernet

?

From ALOHA to Ethernet

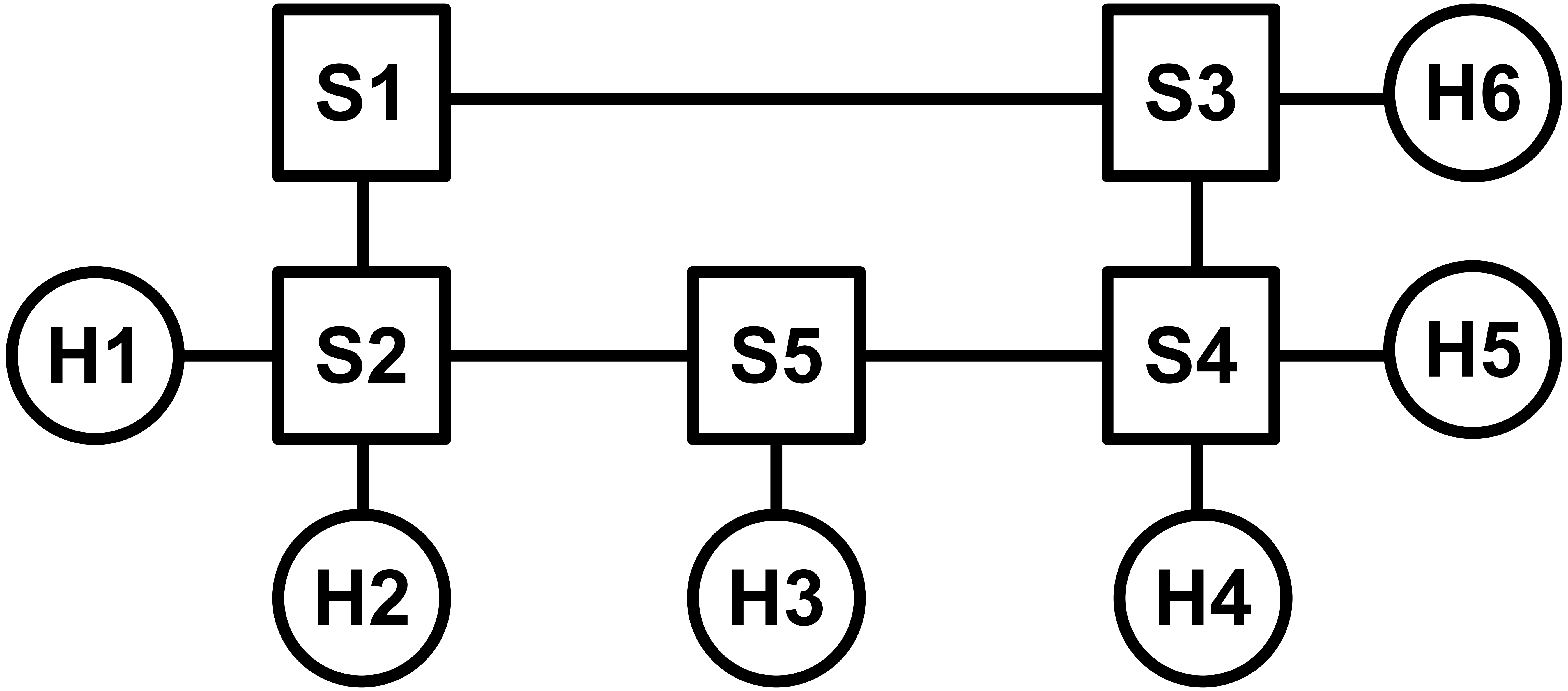


- Robert “Bob” Metcalfe worked at Xerox in 1972 while Xerox was developing the Xerox Alto computer
  - This was a totally groundbreaking computer — the first attempt at a “personal” computer or workstation

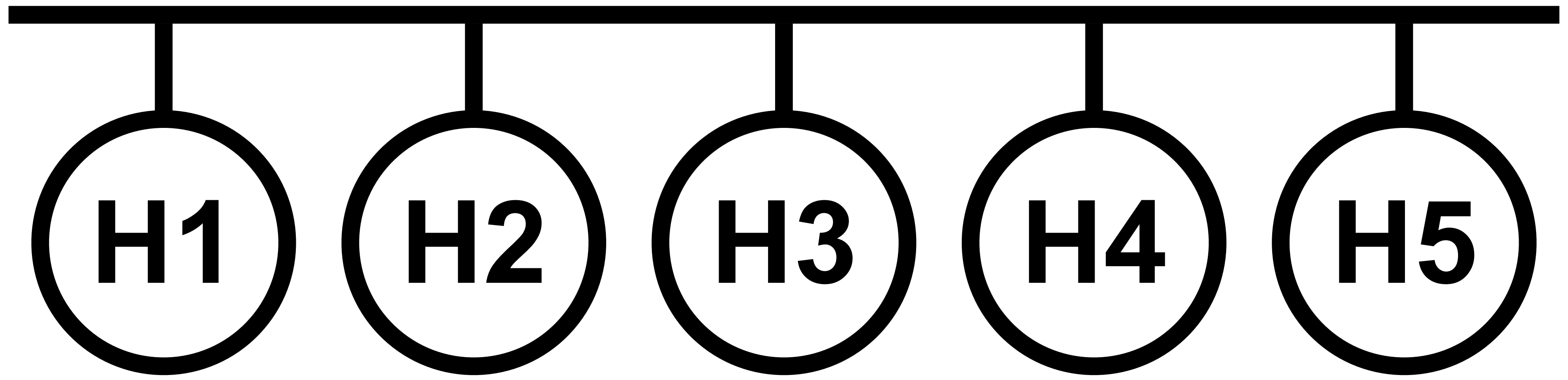
  - When you’ve got tens (hundreds?!) of computers in one building, how do you connect them all?!
  - Didn’t want a centralized “rat’s nest” of cables in a wiring closet
  - Wanted something “maximally distributed”… and cheap
  - .. just run *one* two-conductor cable; connect *all* the computers to it!
    - .. a shared medium!
    - Part of his PhD thesis was on ALOHAnet — used similar ideas

Ethernet

- Early ARPANET (and almost everything we've looked at this semester) were all point-to-point links with switches:



- Bob Metcalfe's Ethernet looked like this:



Ethernet: CSMA

- Refined ALOHA multiple access protocol:

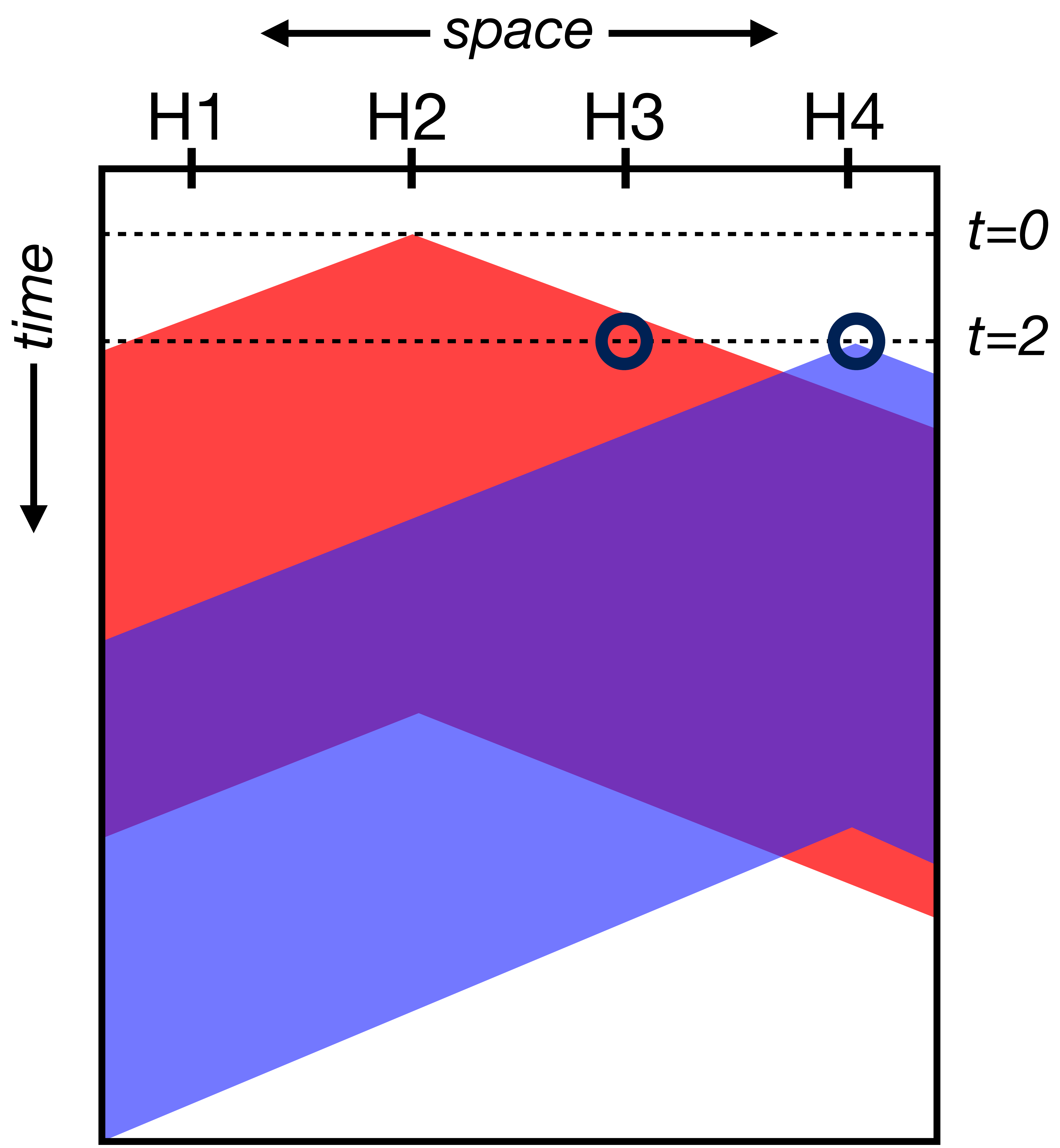
- *Carrier Sense Multiple Access* - CSMA
  - ALOHA is “rude” — nodes just start talking; figure out collisions later
  - CSMA is “polite” — listen first, start talking when it’s quiet
    - Listen = *sense* the signal (*carrier*)

  - .. this is a nice improvement but doesn’t completely avoid collisions
  - .. why not?

      - Propagation delay!

Ethernet: CSMA and propagation delay

- At t=0…
  - H2 transmits
    - Signal propagates as time goes by
- At t=2…
  - H3 has heard it; won't transmit
  - H4 has no idea yet; starts transmitting
    - Signal propagates as time goes by
    - .. and collides with H2's signal!
- Solution: CSMA/CD



Ethernet: CSMA/CD

- *Carrier Sense Multiple Access with Collision Detection (CSMA/CD)*
  - Listen *while* you talk
    - If you start hearing someone else while you're talking, shut up (Detect the collision)
      - Don't bother continuing to transmit the whole packet!
    - .. there's a bit more to it, but this is the basic idea

Ethernet: A final word on retransmission

- After a collision, we wait a random amount of time and retransmit
- If link has high contention (many wanting to send), may keep colliding
- Use randomized *binary exponential backoff*
  - If retransmit after collision also collides, wait up to twice as long
  - Continue doubling for every subsequent collision
  - Retransmits fast when possible, slows down when necessary

Ethernet: Summary so far

- Ethernet
  - Used a shared medium (coaxial cable)
  - .. with a random medium access protocol (CSMA/CD)
  - .. inspired by ALOHA

- Key ideas:
  - Carrier sense
    - **Listen before speaking, and don't interrupt**
    - Check if someone else is already sending data; waiting for them to finish
  - Collision detection
    - **If someone else starts talking at the same time, stop**
    - Realizing when two nodes are transmitting at once (detect data on wire is garbled)
  - Retransmission randomness
    - **Don't start talking again right away**
    - Waiting a random amount of time
  - Exponential backoff
    - **When link is highly contended, be increasingly conservative**
    - On subsequent collisions, upper bound of random wait gets longer and longer (doubles)

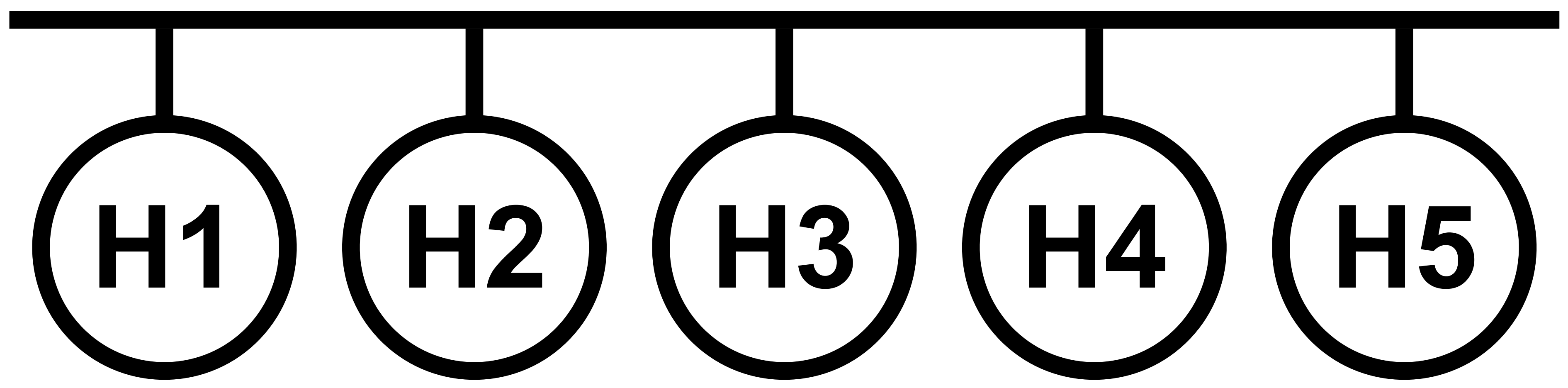
Questions?

Ethernet Addresses
&
Service Types

Ethernet: Addresses and Service Types

- On this shared medium, if you transmit, everyone receives!
  - L2 “address” is not useful as a *locator*
    - .. but it’s useful as an *identifier*
  - We’ve used postal metaphor in this class
    - But there’s no need to find the right street or anything here
    - It’s like everyone is in the same room — you talk, they’ll hear
    - But you still need to say their name so they know who you mean

    - .. there’s no routing or aggregation here — *flat* addresses



Ethernet: Addresses

- 48 bits
- Usually shown as six two-digit hex numbers with colons (or dashes)
- Typically *stored permanently in network interface hardware (“burned in”)*
  - Can often be overridden by software
  - Often found printed on the device

- Structure (simplified)
  - Two bits of flags (we won’t discuss)
  - 22 bits identifying company/organization (e.g. device manufacturer)
  - 24 bits identifying device

- Usually supposed to be globally unique
  - Not because they’re all reachable as in IP!
  - .. just because they’re hardcoded and you don’t know if they will be or not

Ethernet: Service Types

- We've talked about two service types:
  - *Unicast* — send to one recipient
  - *Anycast* — send to any one member of a group

Ethernet: Service Types

- We've talked about two service types:
  - *Unicast* — send to one recipient
  - *Anycast* — send to any one member of a group

- On classic Ethernet, it is trivial to support:
  - *Broadcast* — send to everyone

Ethernet: Broadcast

- Broadcast — send to everyone
  - Specifically, we mean everyone in the specific Ethernet network
    - .. everyone on the same cable!
  - The packet already reaches them, they just need to listen!
  - Implemented using all-ones address:
    - `FF:FF:FF:FF:FF:FF`
  - In classic Ethernet, only really influences receiver
    - It’s just listening to something besides just its normal address
    - Network itself behaves just the same

Ethernet: Service Types

- We've talked about two service types:
  - *Unicast* — send to one recipient
  - *Anycast* — send to any one member of a group

- On classic Ethernet, it is trivial to support:
  - *Broadcast* — send to everyone
  - *Multicast* — send to all members of a group

Ethernet: Multicast

- Multicast — send to all members of a group
  - Again, trivial on classic Ethernet
    - .. just a matter of whether you’re listening for it or not
  - Implemented by setting one of the flags in address to 1:
    - 01:00:00:00:00:00 (the one here is the flag bit)
    - Thus, in all normal addresses, first byte is *even*
    - This is actually the first bit on the wire; bytes are sent low bit first
    - .. note that broadcast is really just a special case

Etherr

- Multic
  - Ag
    - ..
  - Imp
    - 0
    - T
    - T … bit first
    - ..
  - Mu

Real-world multicast example

How does a Mac know when there are things around to AirPlay to? Or network printers nearby?

They're all communicating via multicast using multicast Ethernet address 01:00:5E:00:00:FB !

Your computer sends queries to that address ("I'm looking for printers!").
Relevant devices are listening on that address and answer back ("I'm a printer named foo!").

These messages are formatted as DNS records (PTR, SRV, TXT).
But there's no central server! Each device responds when it sees a query relevant to it.

(Windows does similar using 01:00:5E:00:00:FC.)

Ethernet: Multicast

- Multicast — send to all members of a group
  - Again, trivial on classic Ethernet
    - .. just a matter of whether you're listening for it or not
  - Implemented by setting one of the flags in address to 1:
    - 01:00:00:00:00:00 (the one here is the flag bit)
    - Thus, in all normal addresses, first byte is *even*
    - This is actually the first bit on the wire; bytes are sent low bit first
    - .. note that broadcast is really just a special case
  - Multicast in IP is much more complex to implement!

Ethernet: Service Types

- We've talked about two service types:
  - *Unicast* — send to one recipient
  - *Anycast* — send to any one member of a group

- On classic Ethernet, it is trivial to support:
  - *Broadcast* — send to everyone
  - *Multicast* — send to all members of a group

  - .. basically just a matter of receiver listening to broadcast/multicast addresses and not just their own address

Ethernet: Service Types

- We've talked about two service types:
  - *Unicast* — send to one recipient
  - *Anycast* — send to any one member of a group

- On classic Ethernet, it is trivial to support:
  - *Broadcast* — send to everyone
  - *Multicast* — send to all members of a group
  - .. basically ast/multicast addresses

Quiz!

Does Ethernet support unicast? (Yes)

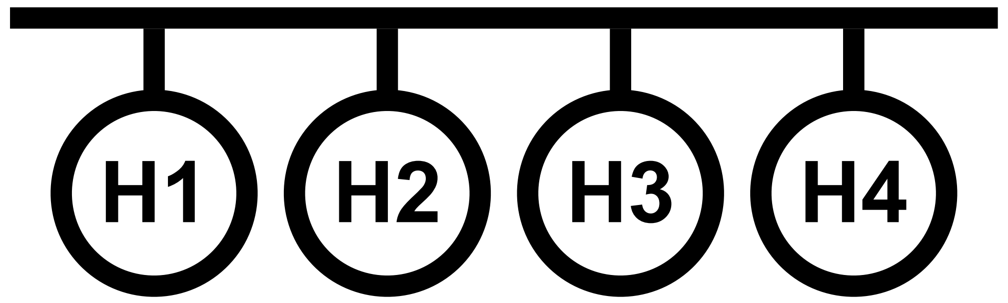
Does Ethernet support anycast? (Not directly)

Questions?

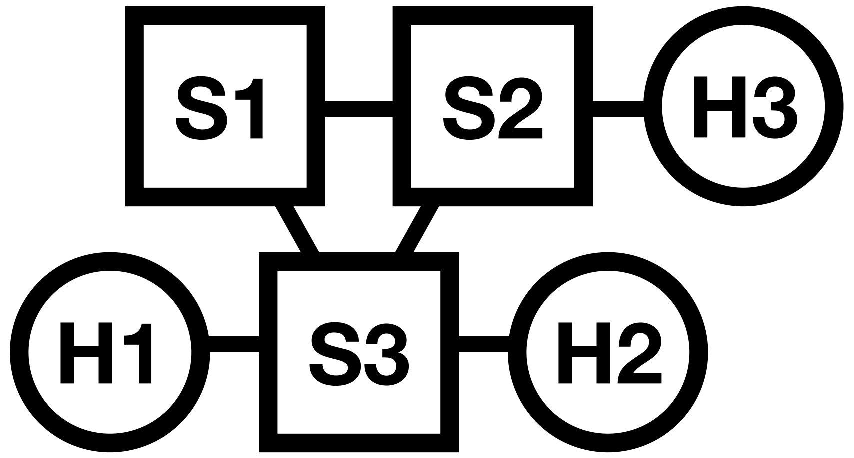
Modern Ethernet

Ethernet: From classic to modern

- I've been sort of hedging here, talking about "classic Ethernet"
  - Shared media with CSMA/CD



- Modern Ethernet rarely uses shared media — "switched Ethernet"
  - Links have exactly two nodes
    - *Nodes transmit on separate wires*
      - It's actually like two unidirectional links
    - No possibility of collision on a single link



  - And no collisions at switches; they queue packets from each link

- But switched Ethernet still mostly *acts like* shared media Ethernet

Ethernet: From classic to modern

- Classic Ethernet
  - Infrastructure is a single cable
  - You send a packet, and everyone gets it

- Switched Ethernet:
  - Essential primitive: **flooding**
  - You send a packet, and everyone gets it

  - Same basic model meant easy transition from single-cable Ethernet
    - No big new element required (e.g., address assignment, routing…)

  - Learning switches are just an optimization:
    - Once you learn where an address is, don't flood for that address

Ethernet: From classic to modern

- Classic Ethernet
  - Infrastructure is a single cable
  - You send a packet, and everyone gets it

- Switched Ethernet:
  - Essential primitive: **flooding**
  - You send a packet, and everyone gets it

  - Sa… …et
    - N… …)

  - Lea…
    - C… …s

Quiz!

Broadcast/multicast on classic Ethernet:
Just send the packet

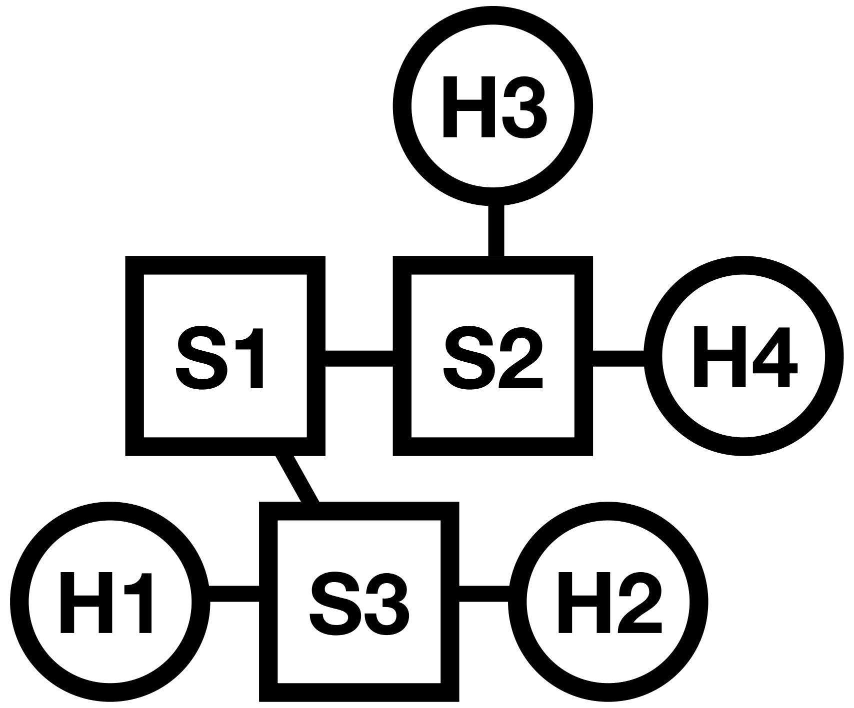
How do you support broadcast/multicast on switched Ethernet?
Just flood it

Questions?

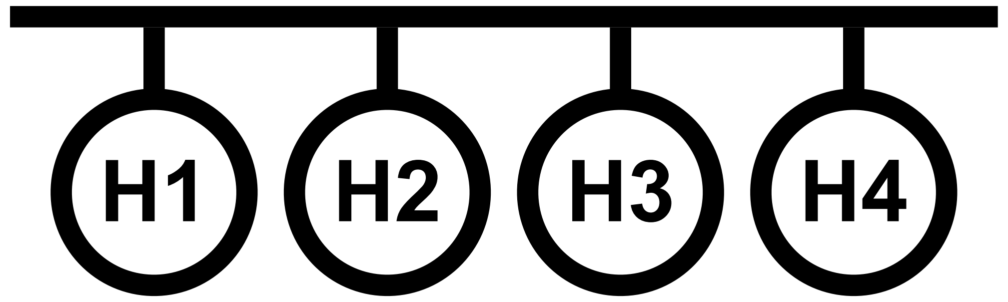
The Interplay of L2 and L3

A note on notation

- Super important note!
- If the switches in this diagram are all L2 switches…



- Then this network is logically equivalent to…



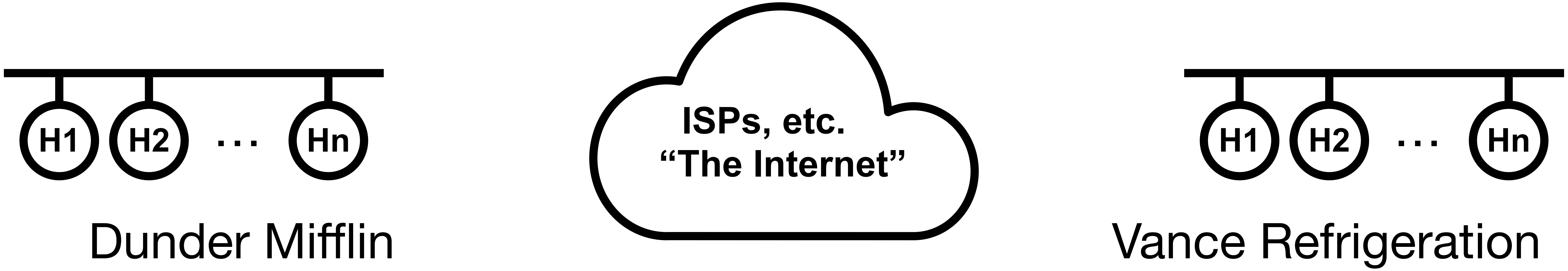
- Right???

L2 and L3 together

- Remember that IP is the *Internet Protocol*
  - Its purpose is to compose many networks into one Internet!
  - What are those networks?

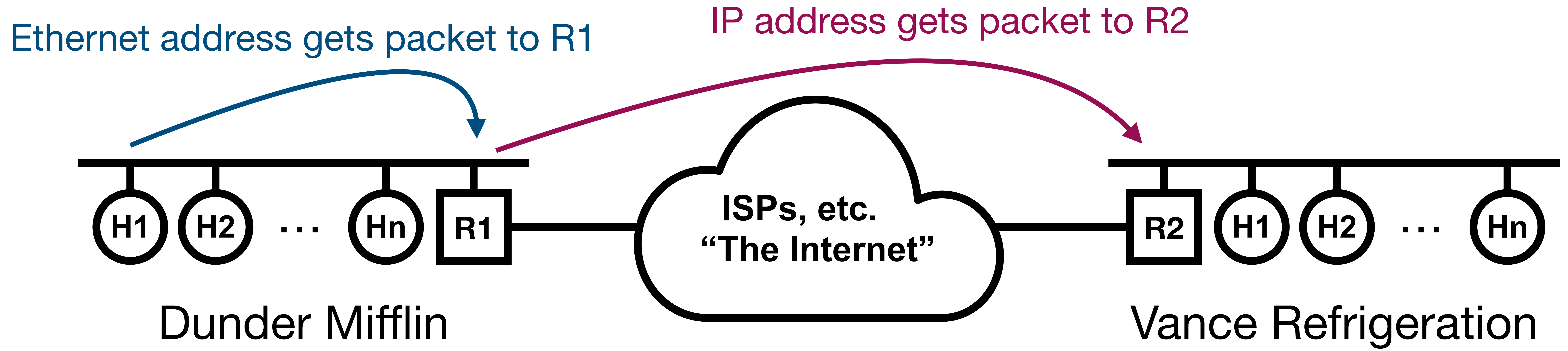
In the context of IP, these are often referred to as *subnets*

    - Many are local networks built with Ethernet! (Or some other L2)



L2 and L3 together

- Remember that IP is the *Internet Protocol*
  - Its purpose is to compose many networks into one Internet!
  - What are those networks?
    - Many are local networks built with Ethernet! (Or some other L2)

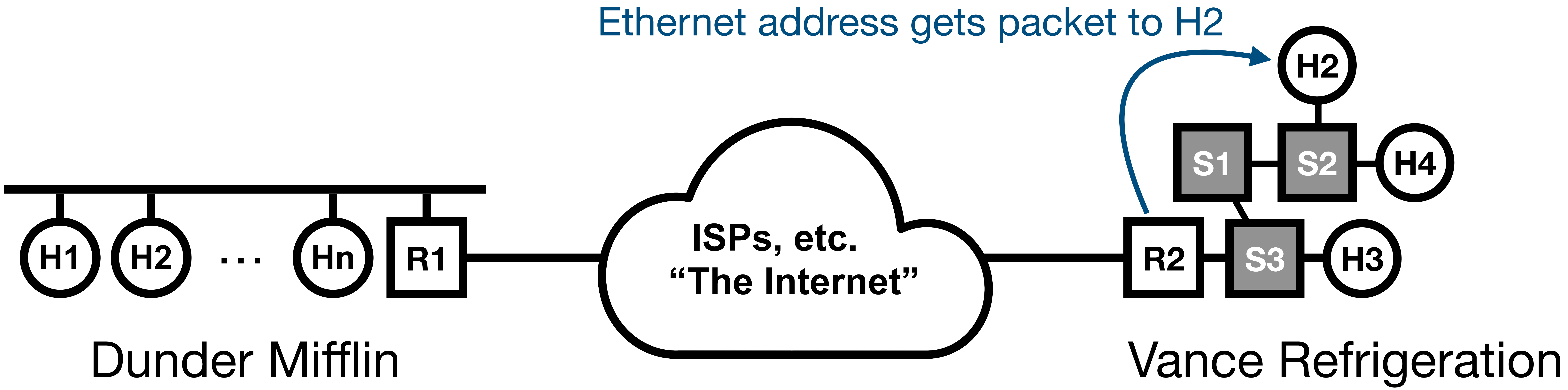


L2 and L3 together

- Remember that IP is the *Internet Protocol*
  - Its purpose is to compose many networks into one Internet!

  - What are those networks?

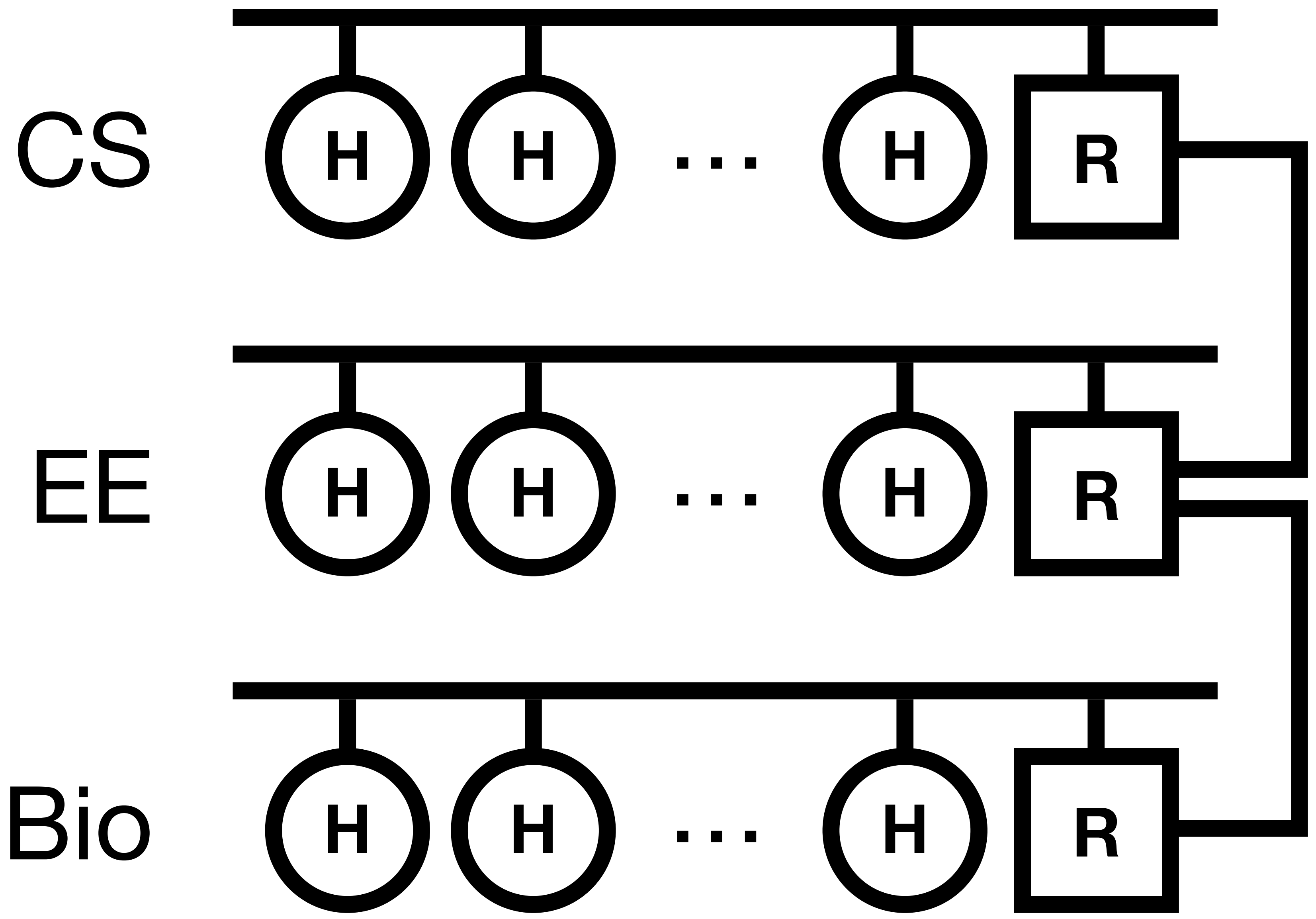
    - Many are local networks built with Ethernet! (Or some other L2)

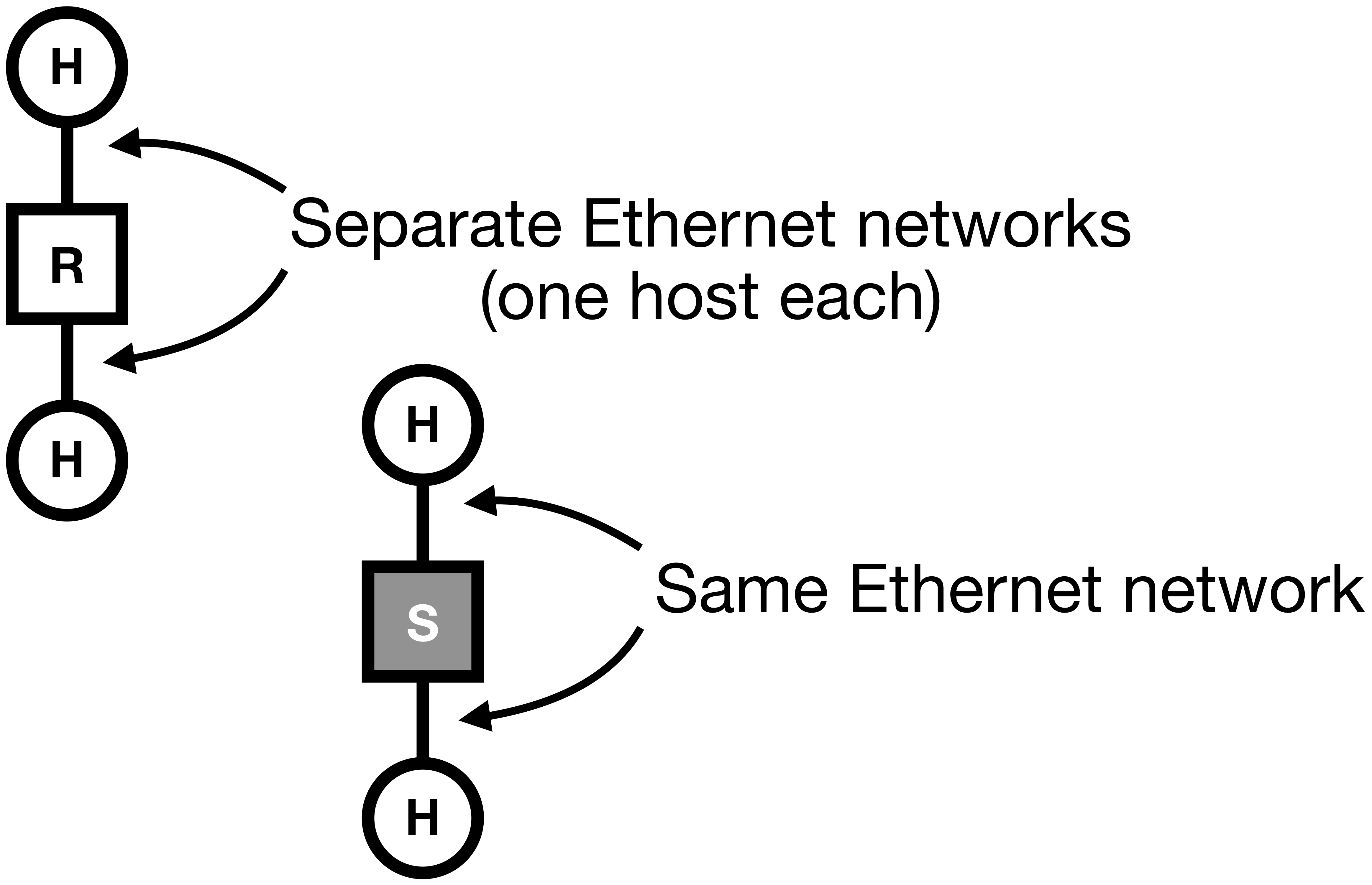


L2 and L3 together

- Note: no reason you can't use IP routers to connect Ethernets in a private part of network (without going through public Internet)!

- Note: no reason your Ethernet needs to have more than two nodes!

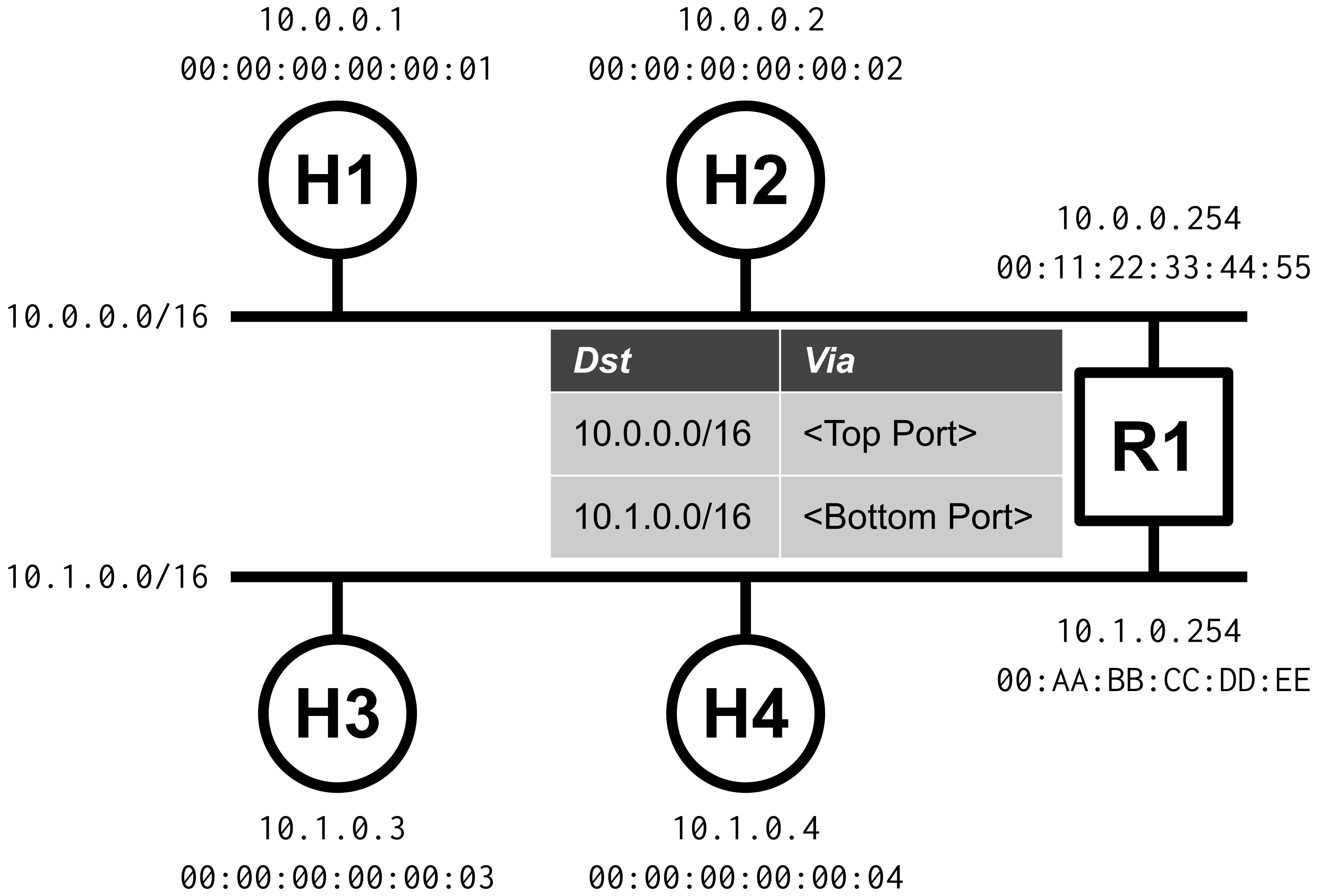




Questions?

L2 and L3 together: sending packets

- Two subnets connected by IP router
- Subnets use different IP prefixes
- IP table populated with static routes
- Router has appropriate IP address for each port
- Note: real Ethernet addresses would be very arbitrary!
(Assigned by manufacturer)



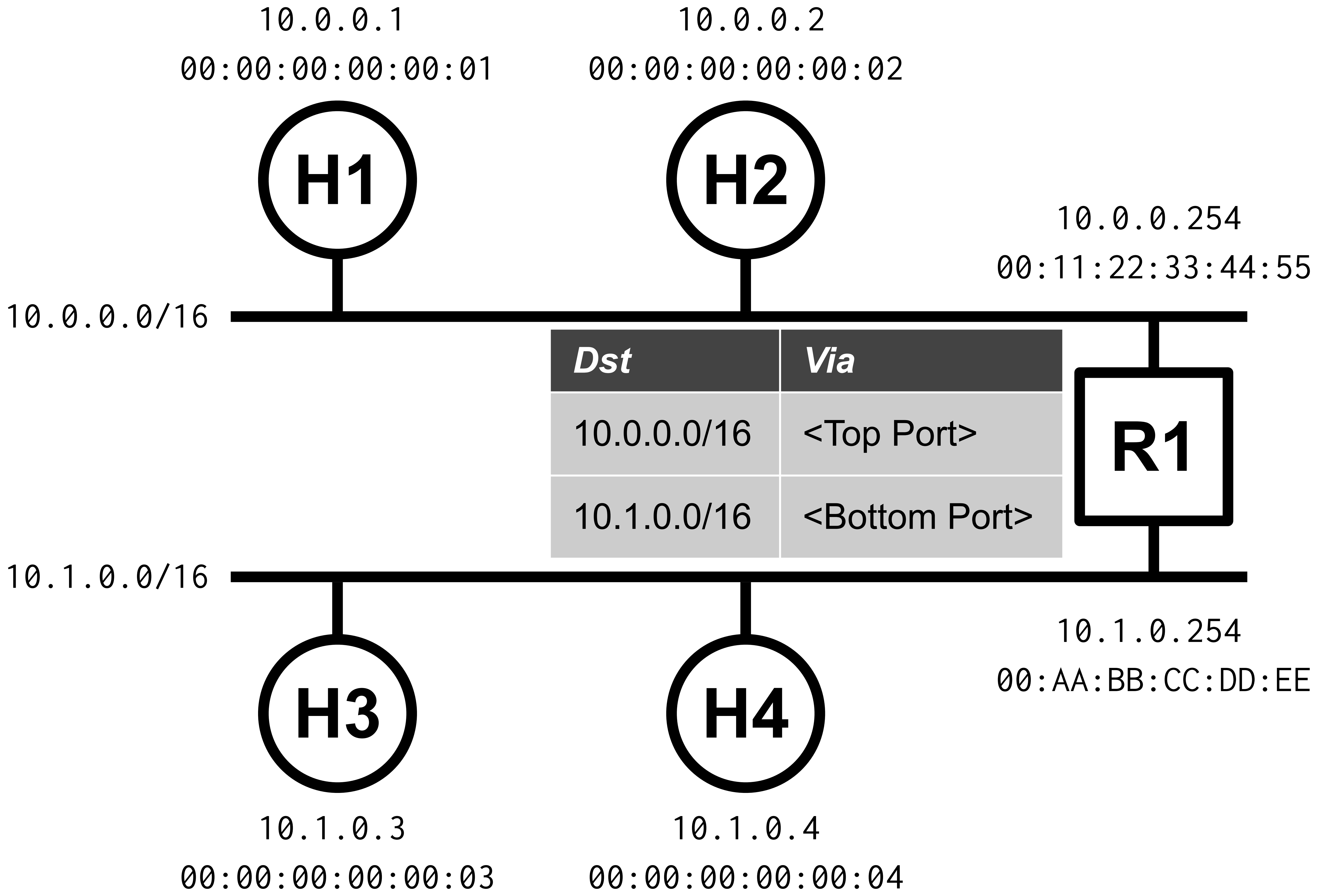
Dst	*Via*
10.0.0.0/16	<Top Port>
10.1.0.0/16	<Bottom Port>

L2 and L3 together: sending packets

- Ex: H1 is sending an IP packet to H2

- They're on the same subnet, so H1 can just put the packet to `10.0.0.2` on the wire, and it'll get to H2

- Is it that easy? Missing something?
  - What Ethernet address should it use?
  - .. without right one, H2 will ignore it!
  - Option 1: `FF-FF-FF-FF-FF-FF`
    - Doesn't allow learned paths
    - Annoys other nodes on network
    - Doesn't always work!
  - Option 2: `00-00-00-00-00-02`
    - But how do we find that?
    - ARP!

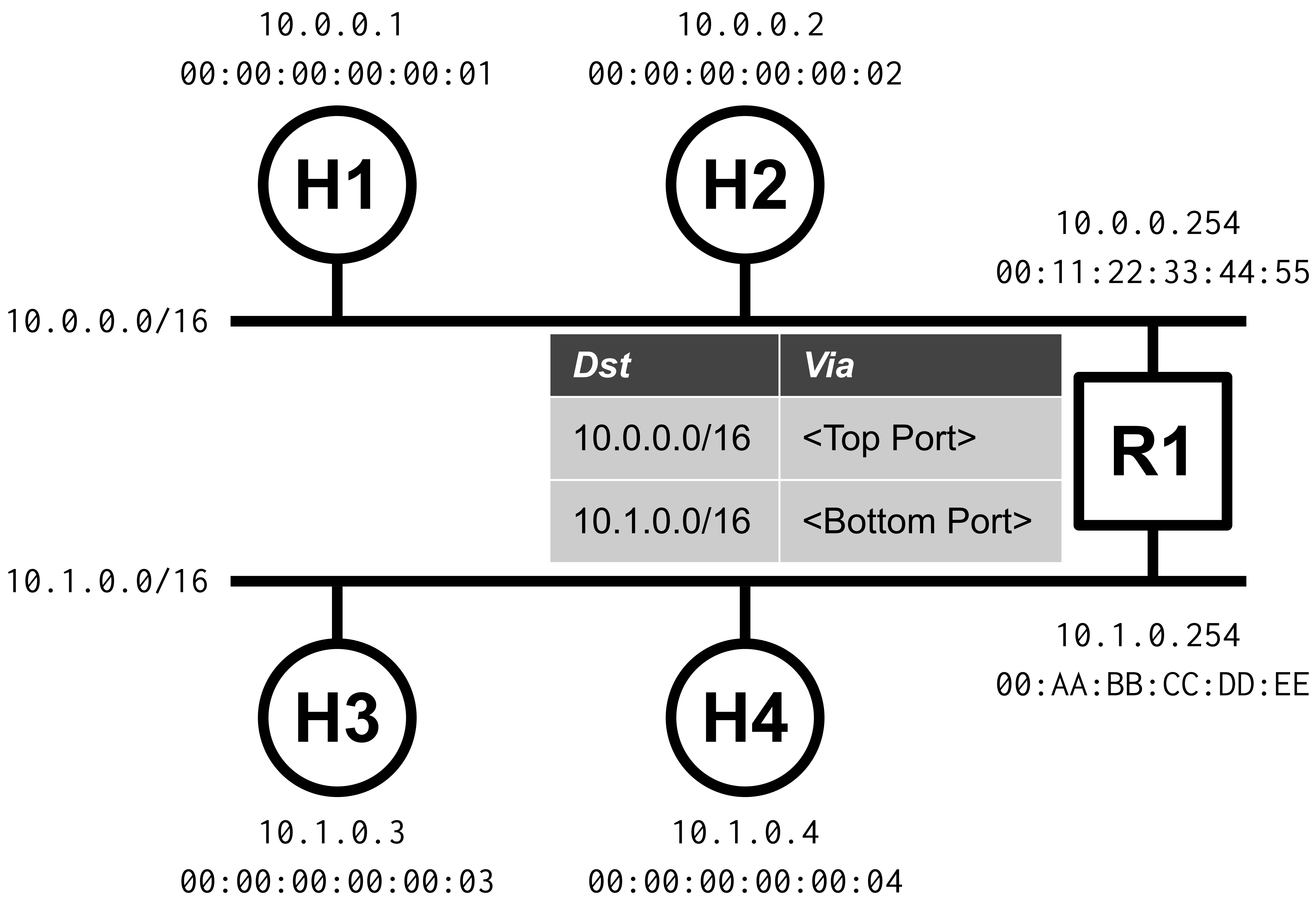


ARP: the Address Resolution Protocol

- Given an IP address, want to know corresponding Ethernet address
- ARP runs directly atop L2 (not part of IP!)
- Host *broadcasts* query:
  - **Who has IP address `w.x.y.z`?**
- Host with address `w.x.y.z` hears query and responds (unicast):
  - **I am `w.x.y.z`, and my Ethernet address is `a1:b2:c3:d4:e5`.**
- Hosts cache results in “ARP table” / “neighbor table”
  - Refresh occasionally (resend queries)

L2 and L3 together: sending packets

- Ex: H1 is sending an IP packet to H2
- They’re on the same subnet, so H1 can just put the packet to `10.0.0.2` on the wire, and it’ll get to H2
- Use ARP to find Ethernet address
- How do we know H2 is on same subnet?
  - Check netmask/prefix:
    - `10.0.0.0/16 = 10.0.0.0/255.255.0.0`
    - `(10.0.0.2 & 255.255.0.0)`
`==`
`(10.0.0.1 & 255.255.0.0)`
- How did we know our netmask?
  - Hold that thought…



Dst	*Via*
10.0.0.0/16	<Top Port>
10.1.0.0/16	<Bottom Port>

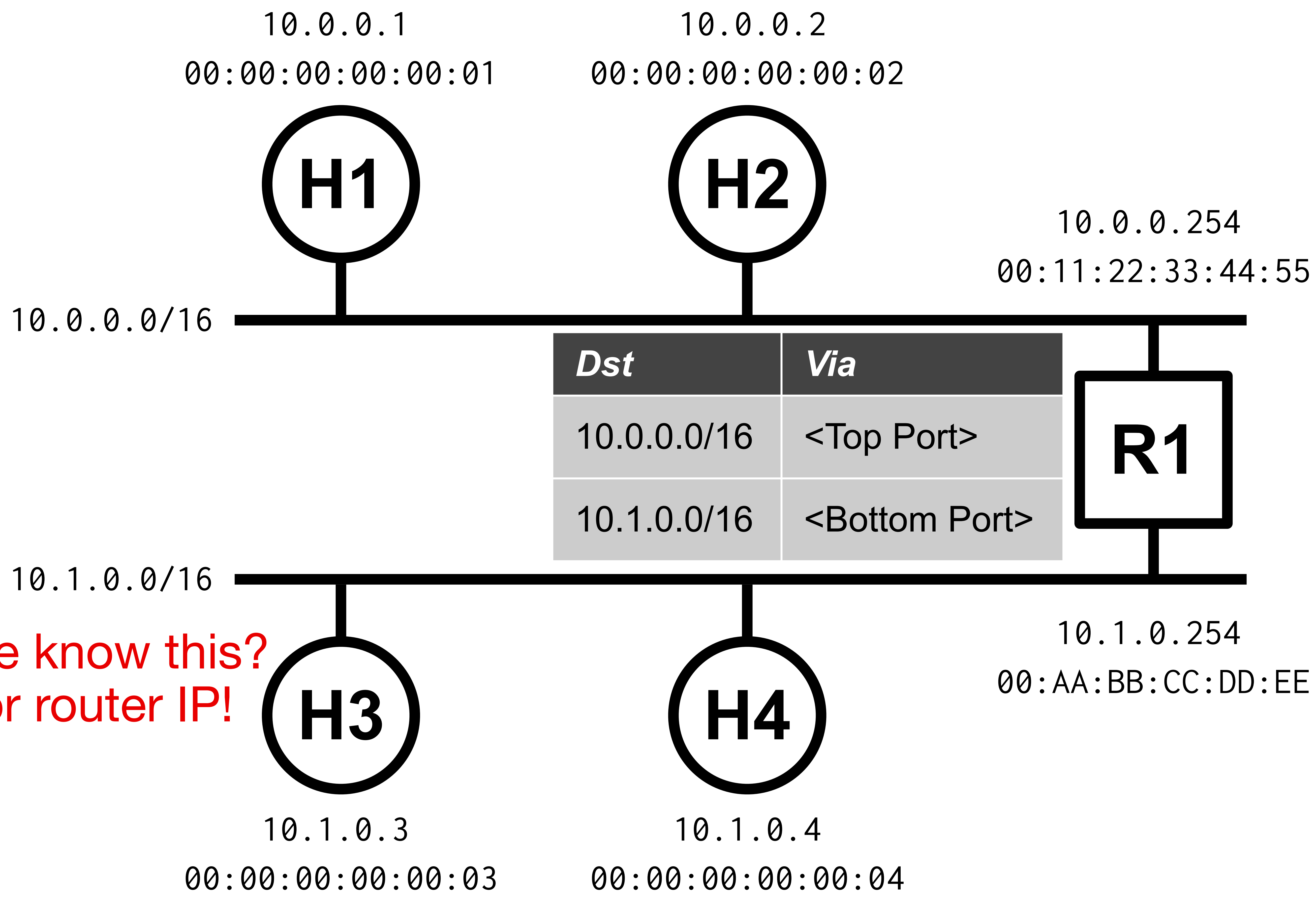
L2 and L3 together: sending packets

- Ex: H1 is sending an IP packet to H3

- Not on the same subnet
  - We must be sending via a router!
  - Assume host knows router's IP

- Packet headers when H1 sends it…
  - src IP: 10.0.0.1
  - src Eth: 00:00:00:00:00:01
  - dst IP: 10.1.0.3
  - dst Eth: 00:11:22:33:44:55
- Packet headers when R1 sends it…
  - src IP: 10.0.0.1
  - src Eth: 00:AA:BB:CC:DD:EE
  - dst IP: 10.1.0.3
  - dst Eth: 00:00:00:00:00:03

How did we know this?
ARPed for router IP!



Dst	*Via*
10.0.0.0/16	<Top Port>
10.1.0.0/16	<Bottom Port>

L2 and L3 together: things a host must know…

- Its own IP address
- Subnet mask (network size) of directly attached network
  - So that we know if another host is directly reachable (at L2) or needs to be reached via router
- IP address of router
  - We didn't need this directly…
  - .. but we used it to get Ethernet address of router
- We're about to discuss how we know all this, but first…

Questions?

DHCP

How to know the things you need to know.

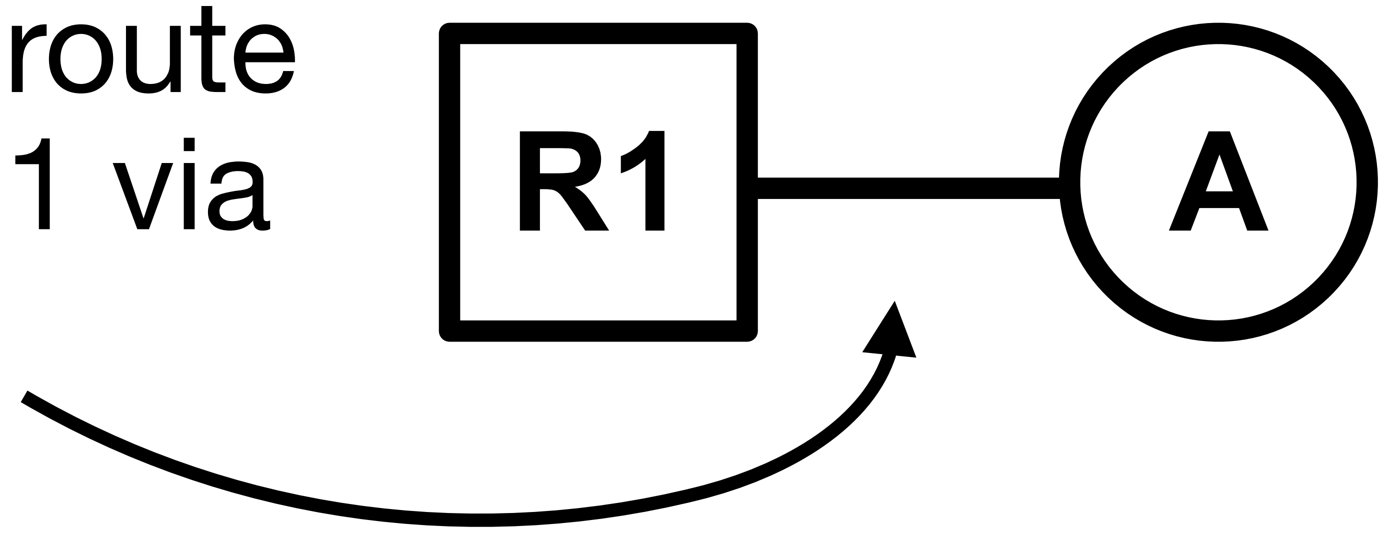
(Assuming you’re a host.)

IP Addresses

- The source of “ground truth” for Ethernet addresses is that addresses are burned into the hardware!
  - **Switch state / routing adapts to hosts (learning).**

- What’s the source of ground truth for IP addresses?
  - Answer 1: Static routes on routers (from network designer/operator)
  - Answer 2: Allocation of addresses from a registrars, e.g., ARIN)
  - **Hosts must adapt to switch state / routing / network authority.**

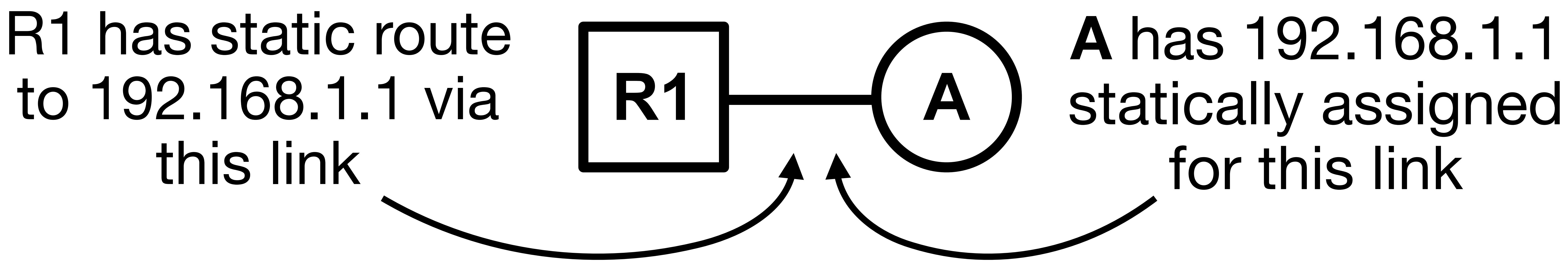
R1 has static route to 192.168.1.1 via this link



But how does **A** know that it is 192.168.1.1 ?!

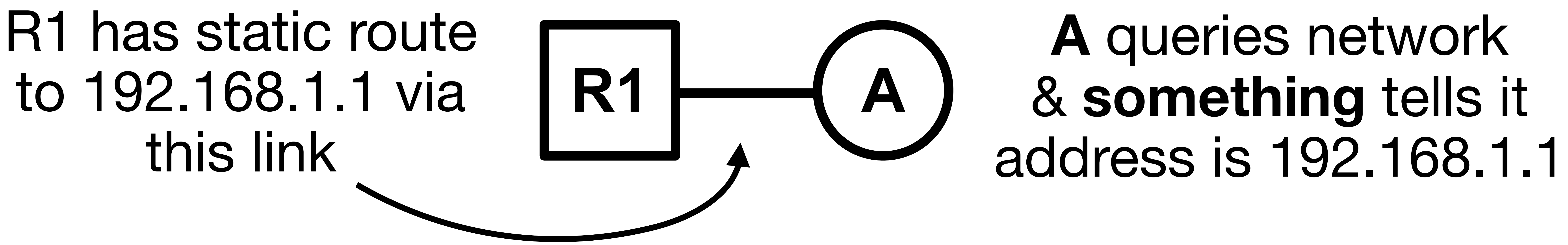
IP Addresses

- Possible solution 1:
  - Manual — statically assign address to hosts
  - Static works well for networks (that don't move/change much)
  - Static worked fine for hosts when computers were big and few
  - .. works less well today
    - Discounting COVID-19, we often move our hosts around several times per day!
    - Doing it manually would be a pain!



IP Addresses

- Possible solution 2:
  - Observation: “The network” already knows valid addresses
    - .. operators got the block of addresses from ARIN or whoever
    - .. operators configured routers with those addresses
  - So… design a protocol so that the network can tell the hosts!
    - DHCP!



IP Addresses: DHCP

- DHCP is the *Dynamic Host Configuration Protocol*
  - Provides a way for hosts to query "the network" for local configuration information
    - Crucial IP configuration stuff:
      - **IP address**
      - **Netmask**
      - "Default gateway" = **first hop router**
    - Also important:
      - **Local DNS resolving server**
    - Much less important: lots of other assorted stuff (all optional)

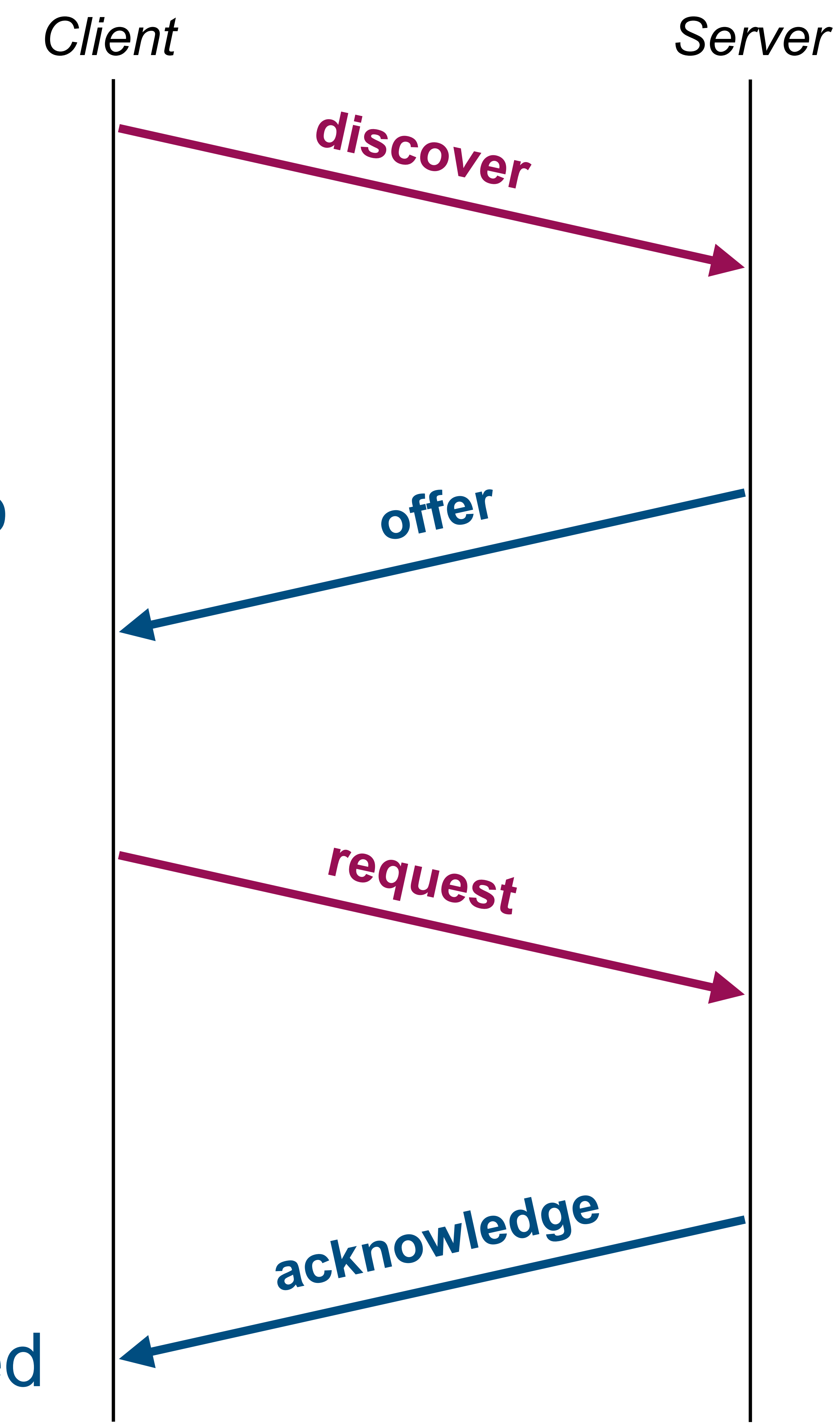
Exactly the three things we said we wanted to know a moment ago

DHCP

- One or more *DHCP servers* are added to network
  - Can be separate machine or built into a router (e.g., your wifi router)
  - Listen on well-known UDP port 67
  - Configured with required information
    - First hop router address, local DNS server
    - A *pool* of usable IP addresses
  - Servers *lease* hosts an IP address
    - Only valid for a limited time (often hours or a day)
    - Host must renew if it wants to keep it
    - Server won't offer it to another host if it's currently leased!

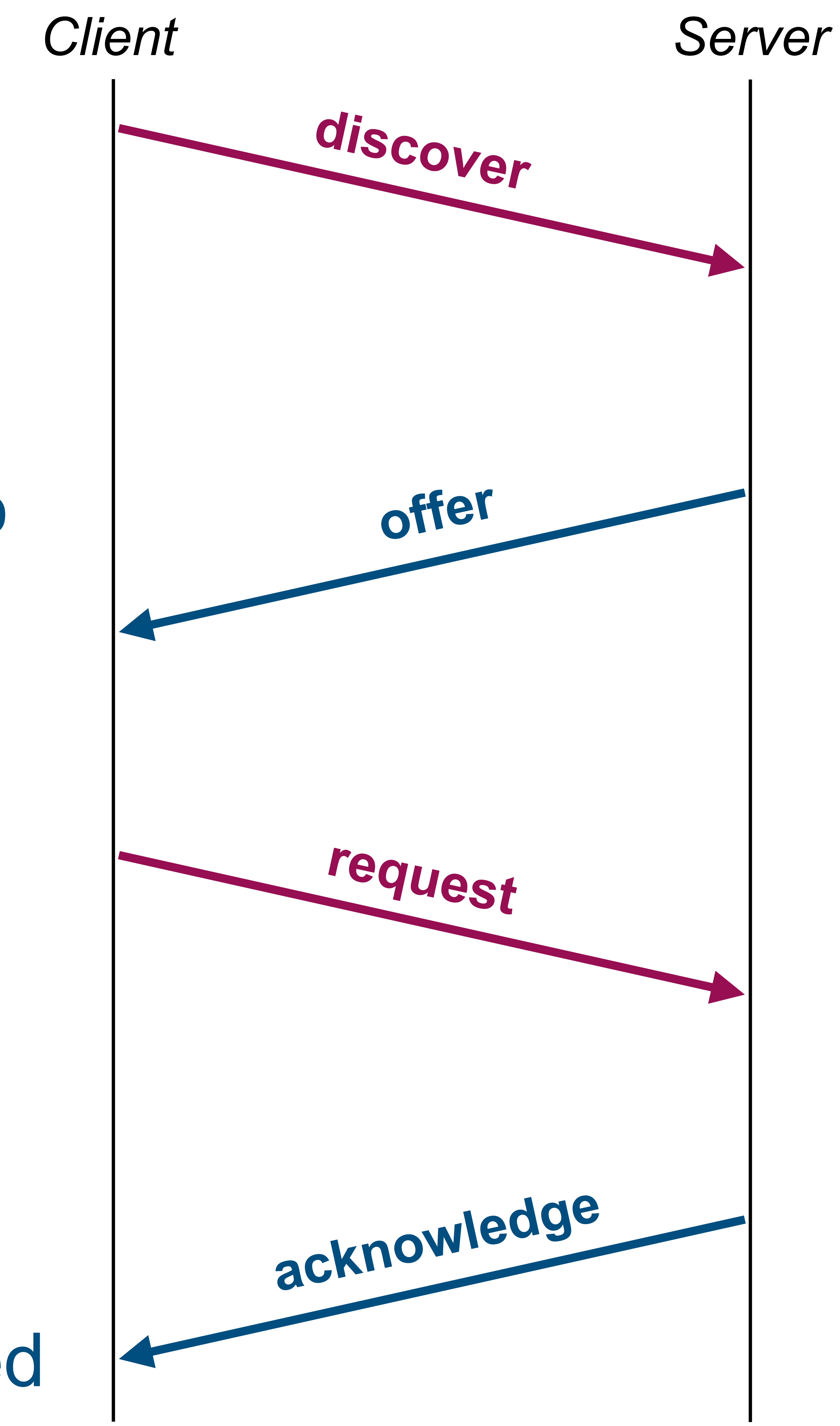
DHCP

- Client sends a **discover message** — asks for config info
- Server(s) send(s) **offer message** with config info (e.g., particular IP)
- Client sends **request message** to accept a particular offer
- Server sends **acknowledge message** to confirm request granted



DHCP

- Client sends a **discover message** — asks for config info
- Server(s) send(s) **offer message** with config info (e.g., particular IP)
- Client sends **request message** to accept a particular offer
- Server sends **acknowledge message** to confirm request granted



- DHCP built on UDP (built on IP)…
- How does client know server IP?
  - It might not!
  - What do you do about that?
  - **Broadcast** messages to it
  - Eth/L2 - `FF:FF:FF:FF:FF:FF`
  - IP/L3 - `255.255.255.255`
- What IP does server use for client?
  - Client doesn't have one yet!
  - **Broadcast** messages to it
- Source IP in packets from client?
  - `0.0.0.0`

DHCP

- Client sends a **discover message** — asks for config info
- Server(s) send **message** with (e.g., particula
- Client sends **r message** to a particular offer
- Server sends **acknowledge** to confirm request granted

Client

Server

- DHCP built on UDP (built on IP)…
- w server IP?
- out that?
- ges to it
- FF:FF:FF
- 5.255
- use for client?
- one yet!
- ges to it
- from client?
  - 0.0.0.0

Quiz!

Q: What does broadcasting imply about the location of the DHCP server?

A: It's got to be available on the L2 network (within "broadcast range" of the client)! Broadcast doesn't generally extend beyond that!

DHCP relays (generally part of a router) can do special forwarding across L2 networks if necessary.

Questions?

DHCP

- Final DHCP question:
  - Why doesn't DHCP just give us the router's Ethernet address?
  - .. did we actually need the router's IP address?
  - It's cleaner — IP configuration all in terms of IP
    - There must be some mechanism for mapping from L3 to L2 addr
      - Just use it, whatever it is
    - Means that DHCP (and IP config in general) is the same even when used with different L2s

Everything together now!

We’ve been building toward
this all semester!

First a quick recap...

- Hosts know their Ethernet address…
  - .. because it's *burnt in* to hardware
- Hosts know their IP address…
  - .. via DHCP
- Hosts learn mapping from IP to Ethernet addresses…
  - .. via ARP

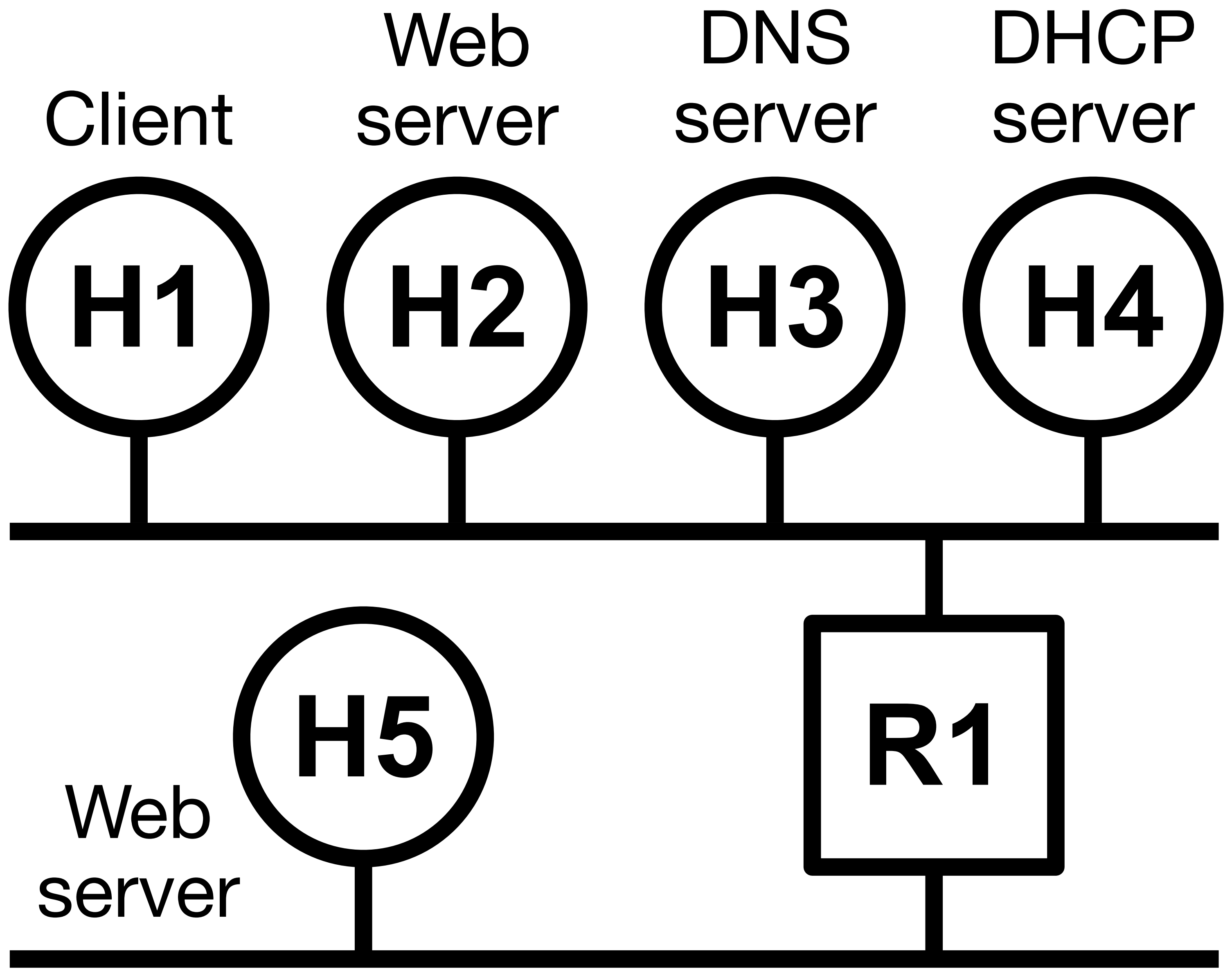
- Other things you learn from DHCP…
  - Subnet mask
  - First hop router IP address
  - Local DNS resolving server

- DHCP and ARP use a lot of *broadcast*
  - Scalability is okay, because only broadcasts to local L2 network
  - Solves chicken/egg addressing problems (i.e., don't know who ask so ask everyone)

The Setup

- Scenario:
  - Two subnets connected by router R1
  - Host H1…
    - Boots up (all state cleared)
    - Fetches a small file from H5.com
    - Goes idle for five minutes
    - Fetches two small files from H2.com

- The Task:
  - List (in order) the packets H1 sends/receives



Assumptions

- HTTP uses persistent connections
  - Browser times out after one minute
  - Server times out after two minutes
- HTTP requests/responses fit in single packets
- No TCP “piggybacking” — i.e., no data on returning ACKs (next slide)

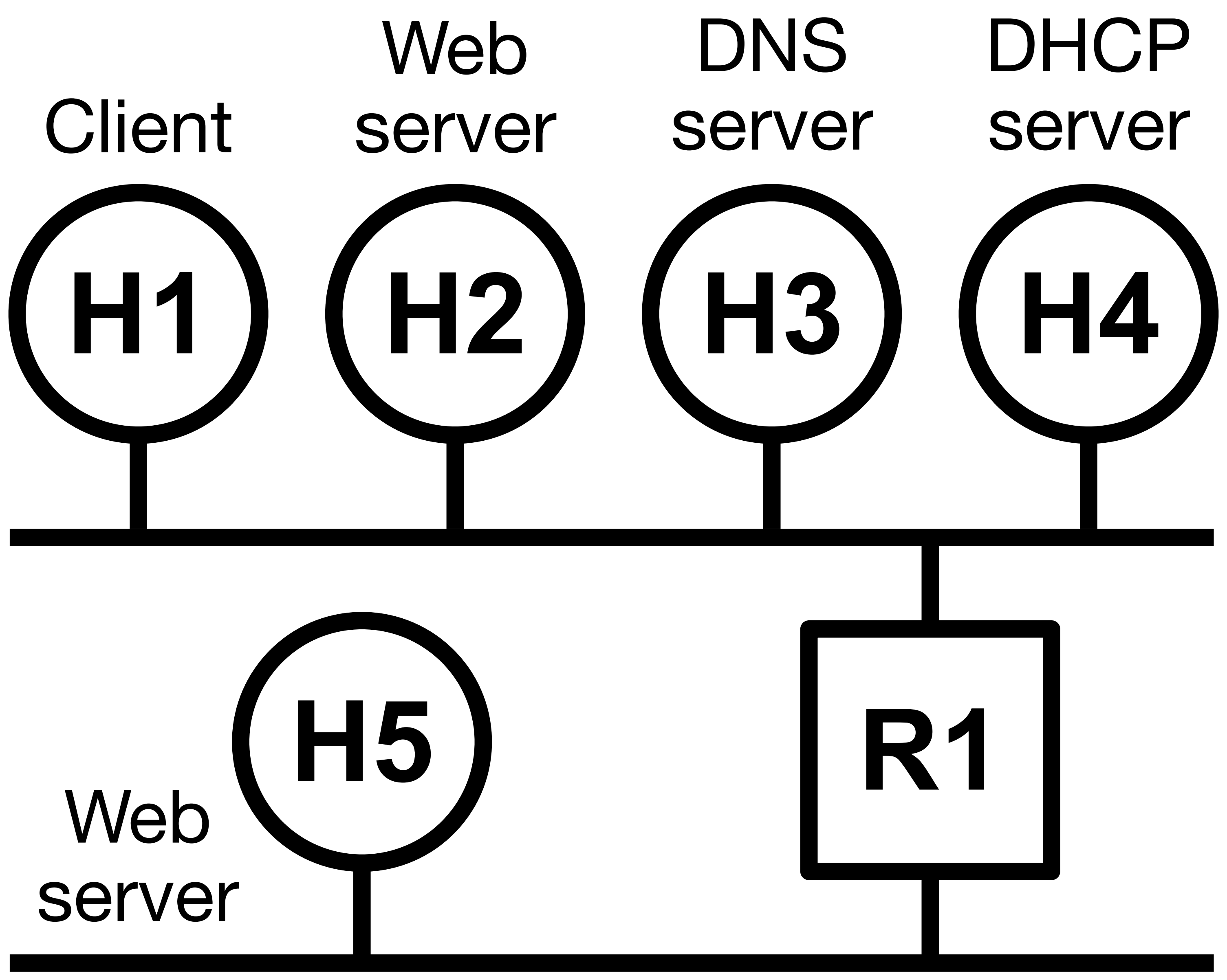
When to piggyback?

- TCP implementation is typically in kernel
  - Returning ACKs are generated in kernel
- Applications (HTTP and above) are in userspace
  - Application responses generated in userspace
- ACKs are often generated before application has chance to respond
  - Kernel creates ACK and *schedules application to run later*
  - Exception if kernel is delaying ACKs (which is a thing, but not in our example)
- Similar with TCP close:
  - Generally see FIN, ACK, FIN, ACK — not FIN, FIN+ACK, ACK
  - Generally, one application side sees other side close, then closes its side
- In what follows, do not use any piggybacking…
  - .. except in SYN+ACK (because done in kernel)

The Setup

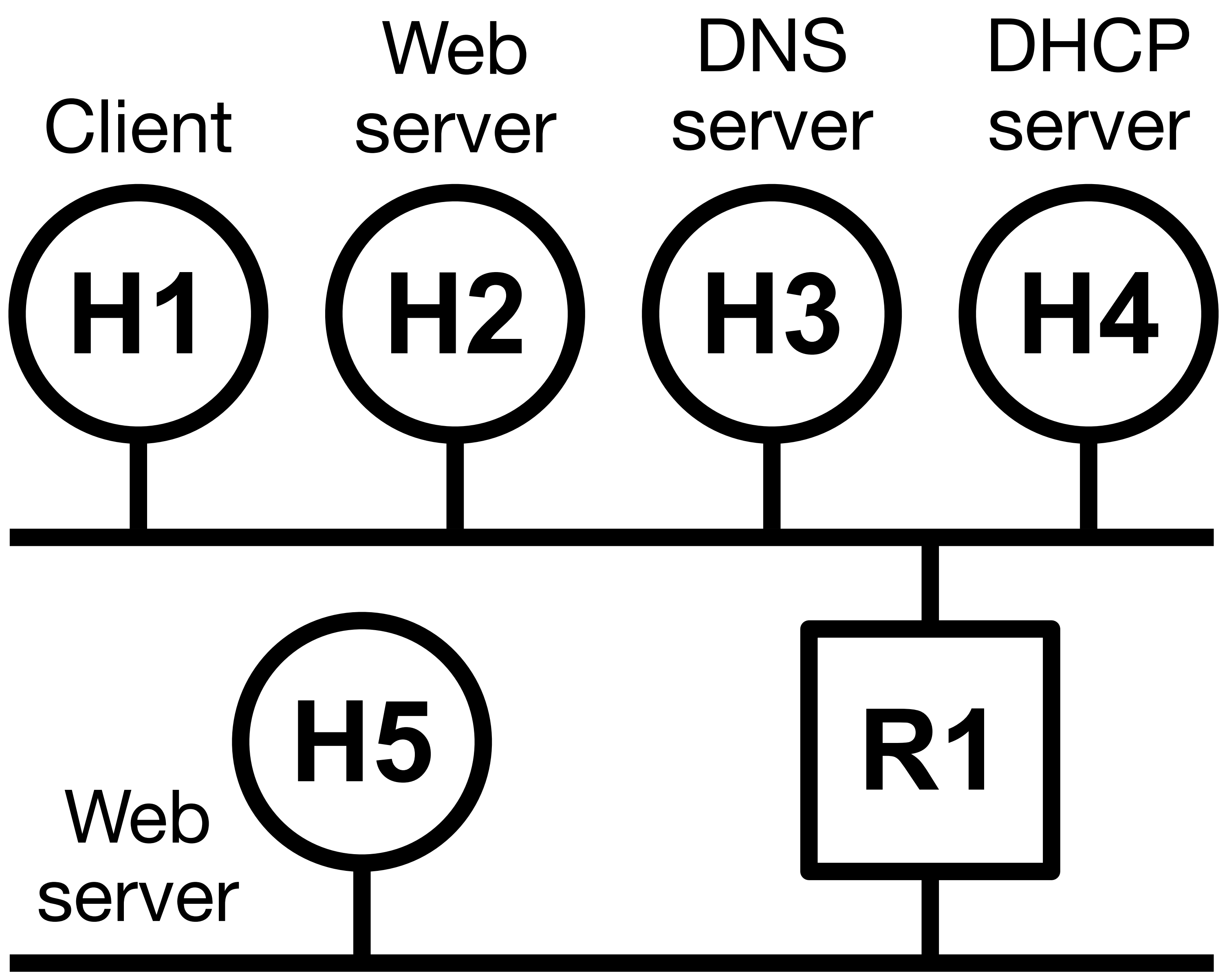
- Host H1…
  - Boots up (all state cleared)
  - Fetches a small file from H5.com
  - Goes idle for five minutes
  - Fetches two small files from H2.com

- To do list:
  - DHCP (get configured)
  - ARP for DNS server
  - Resolve H5.com
  - ARP for R1
  - TCP connection to H5
  - HTTP request to H5
  - TCP disconnect from H5
  - Resolve H2.com
  - ARP for H2
  - TCP connection to H2
  - HTTP request to H2
  - HTTP request to H2
  - TCP disconnect from H2



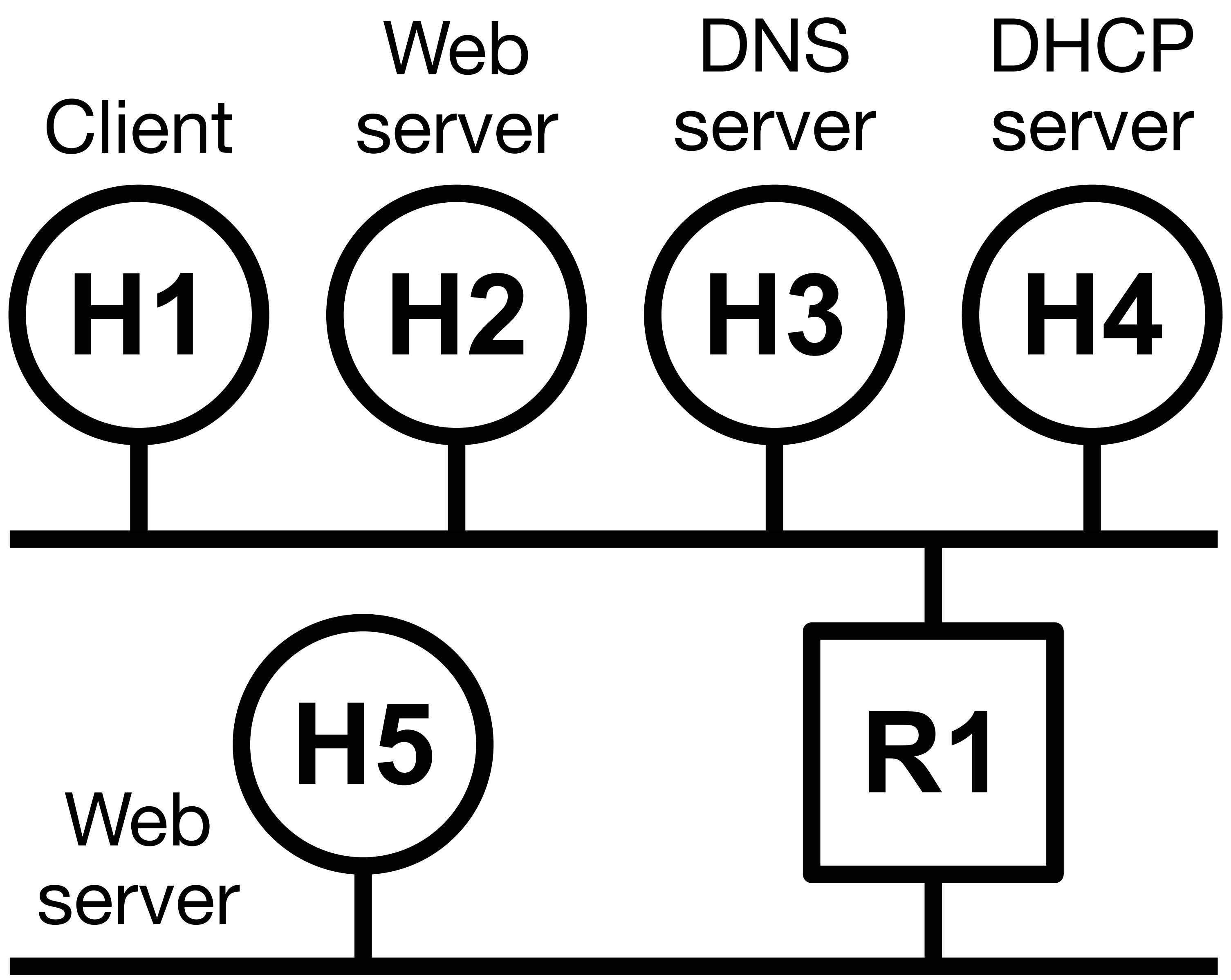
Dir	O	Trnsp	Message
⇉	*	UDP	DHCP discover
⇇	H4	UDP	DHCP offer
⇉	*	UDP	DHCP request
⇇	H4	UDP	DHCP acknowledge
⇉	*	-	ARP request for H3
←	H3	-	ARP response from H3
→	H3	UDP	DNS request for H5.com
←	H3	UDP	DNS response for H5.com
⇉	*	-	ARP request for R1
←	R1	-	ARP response from R1
→	H5	TCP	SYN
←	H5	TCP	SYN+ACK
→	H5	TCP	ACK
→	H5	TCP	HTTP GET
←	H5	TCP	ACK
←	H5	TCP	HTTP response
→	H5	TCP	ACK
→	H5	TCP	FIN
←	H5	TCP	ACK
←	H5	TCP	FIN
→	H5	TCP	ACK

✔ 1. DHCP (get configured)
✔ 2. ARP for DNS server
✔ 3. Resolve H5.com
✔ 4. ARP for R1
✔ 5. TCP connection to H5
✔ 6. HTTP request to H5
✔ 7. TCP disconnect from H5
8. Resolve H2.com
9. ARP for H2
10. TCP connection to H2
11. HTTP request to H2
12. HTTP request to H2
13. TCP disconnect from H2



Dir	O	Trnsp	Message
→	H3	UDP	DNS request for H2.com
←	H3	UDP	DNS response for H2.com
⇉	*	-	ARP request for H2
←	R1	-	ARP response from H2
→	H5	TCP	SYN
←	H5	TCP	SYN+ACK
→	H5	TCP	ACK
→	H5	TCP	HTTP GET
←	H5	TCP	ACK
←	H5	TCP	HTTP response
→	H5	TCP	ACK
→	H5	TCP	HTTP GET
←	H5	TCP	ACK
←	H5	TCP	HTTP response
→	H5	TCP	ACK
→	H5	TCP	FIN
←	H5	TCP	ACK
←	H5	TCP	FIN
→	H5	TCP	ACK

- ✔ 1. DHCP (get configured)
- ✔ 2. ARP for DNS server
- ✔ 3. Resolve H5.com
- ✔ 4. ARP for R1
- ✔ 5. TCP connection to H5
- ✔ 6. HTTP request to H5
- ✔ 7. TCP disconnect from H5
- ✔ 8. Resolve H2.com
- ✔ 9. ARP for H2
- ✔ 10. TCP connection to H2
- ✔ 11. HTTP request to H2
- ✔ 12. HTTP request to H2
- ✔ 13. TCP disconnect from H2



Questions?

Thank you!

Good luck on the project and final!

Attributions

Norman Abramson

Public Domain, https://commons.wikimedia.org/wiki/File:NormanAbramson.jpg

Kris Krug, Bob Metcalfe and Tim Berners Lee (Cropped), by Shashi Bellamkonda

CC-BY 2.0, https://www.flickr.com/photos/drbeachvacation/8543090629

Many slides borrowed/adapted from earlier CS168/EE122, with thanks to Nick McKeown, Sylvia Ratnasamy, Jennifer Rexford, Scott Shenker, Ion Stoica, and others