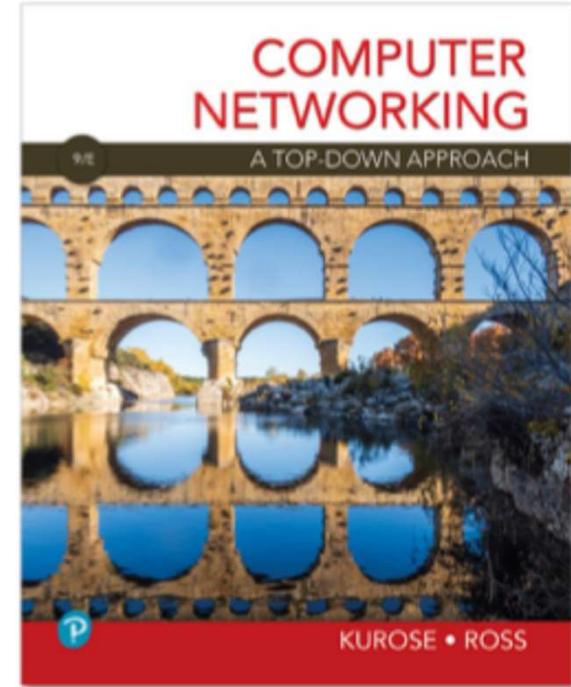


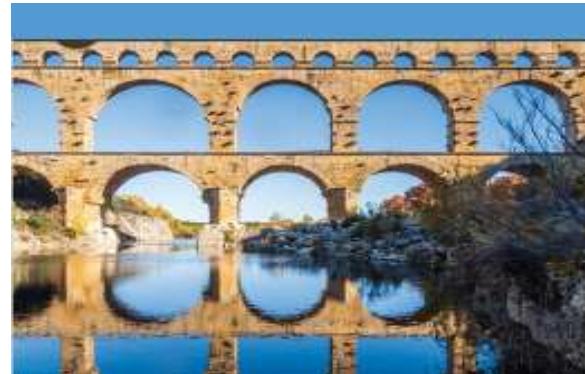
The Link Layer and LANs



*Computer Networking: A
Top-Down Approach*
9th edition
Jim Kurose, Keith Ross
Pearson, 2025

Link layer, LANs: roadmap

- LANs
 - addressing, ARP
 - switches
 - VLANs
- link virtualization: MPLS



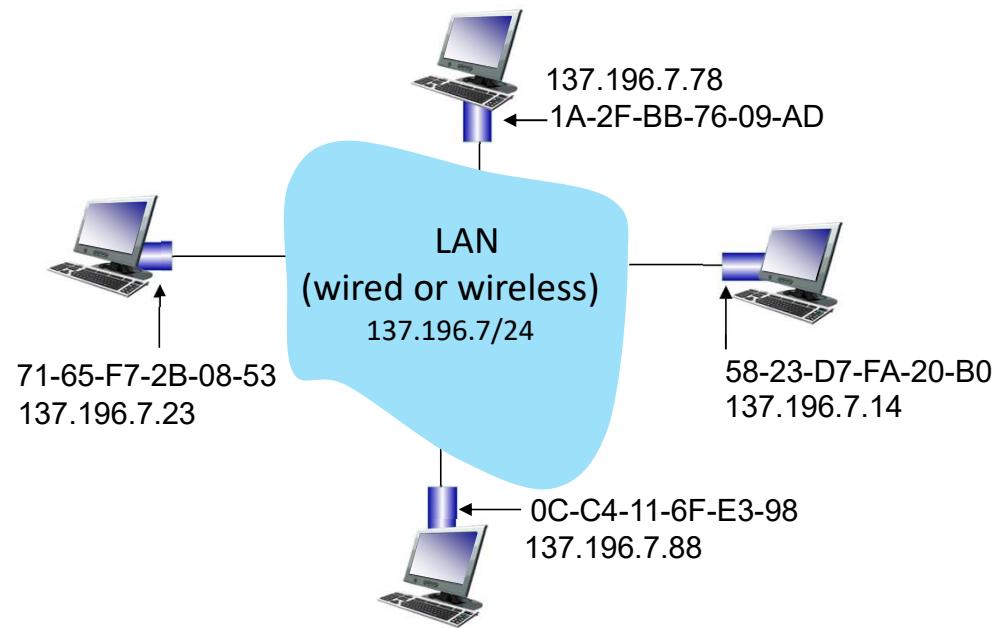
MAC addresses

- 32-bit IP address:
 - *network-layer* address for interface
 - used for layer 3 (network layer) forwarding
 - e.g.: 128.119.40.136
 - MAC (or LAN or physical or Ethernet) address:
 - function: used “locally” to get frame from one interface to another physically-connected interface (same subnet, in IP-addressing sense)
 - 48-bit MAC address (for most LANs) burned in NIC ROM, also sometimes software settable
 - e.g.: 1A-2F-BB-76-09-AD
- hexadecimal (base 16) notation
(each “numeral” represents 4 bits)*

MAC addresses

each interface on LAN

- has unique 48-bit MAC address
- has a locally unique 32-bit IP address (as we've seen)

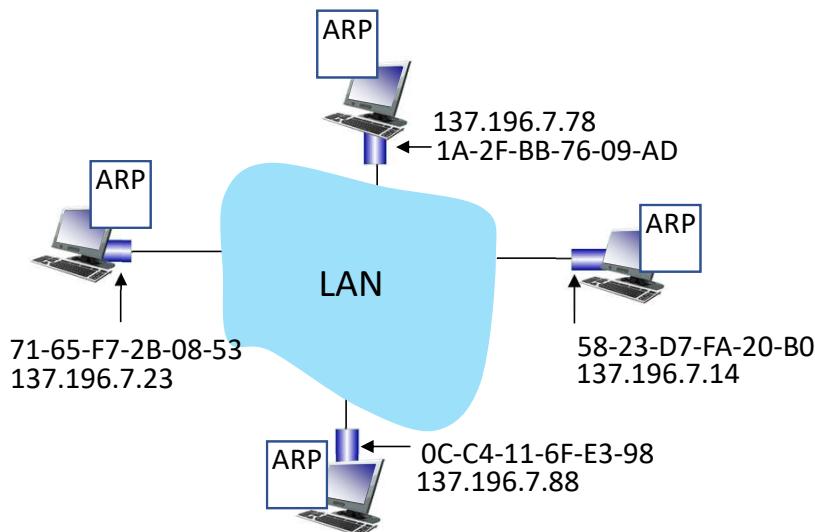


MAC addresses

- MAC address allocation administered by IEEE
- manufacturer buys portion of MAC address space (to assure uniqueness)
- analogy:
 - MAC address: like Social Security Number
 - IP address: like postal address
- MAC flat address: portability
 - can move interface from one LAN to another
 - recall IP address *not* portable: depends on IP subnet to which node is attached

ARP: address resolution protocol

Question: how to determine interface's MAC address, knowing its IP address?



ARP table: each IP node (host, router) on LAN has table

- IP/MAC address mappings for some LAN nodes:
<IP address; MAC address; TTL>
- TTL (Time To Live): time after which address mapping will be forgotten (typically 20 min)

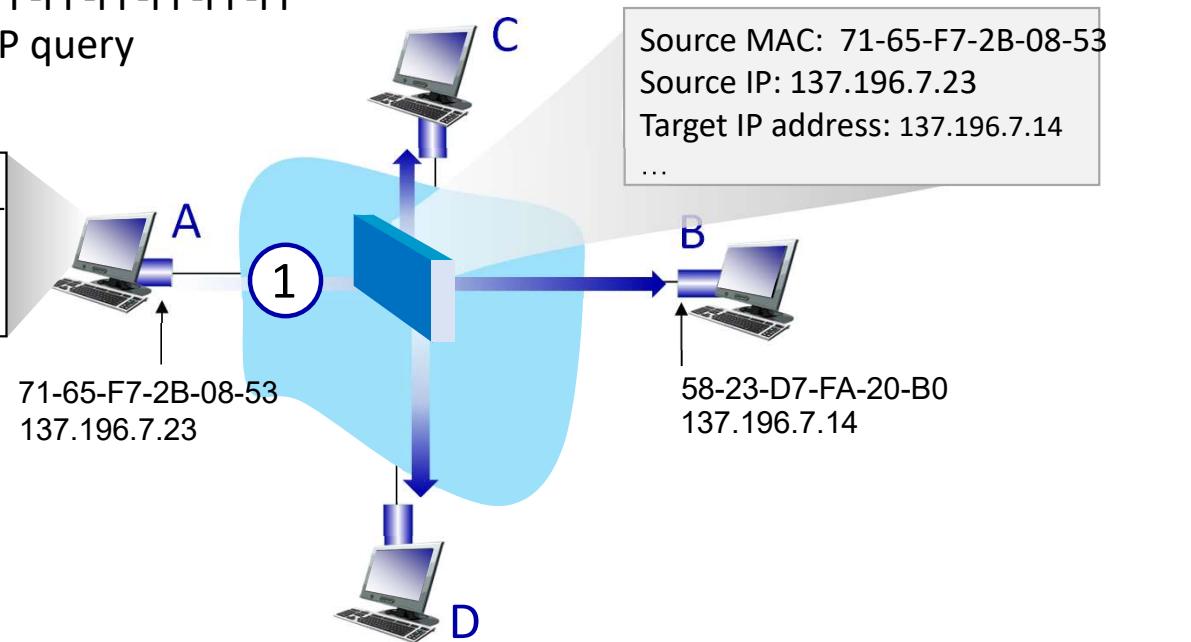
ARP protocol in action

example: A wants to send datagram to B

- B's MAC address not in A's ARP table, so A uses ARP to find B's MAC address

- ① A broadcasts ARP query, containing B's IP addr
- destination MAC address = FF-FF-FF-FF-FF-FF
 - all nodes on LAN receive ARP query

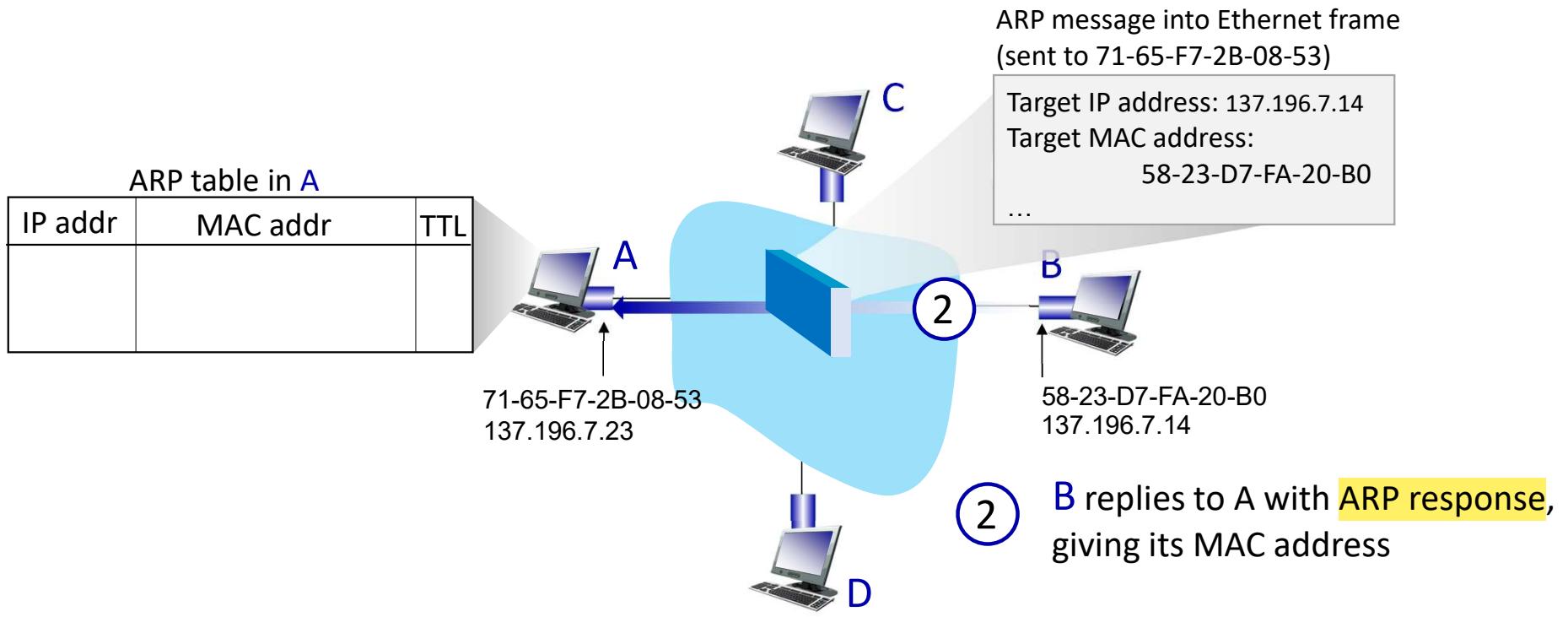
IP addr	MAC addr	TTL



ARP protocol in action

example: A wants to send datagram to B

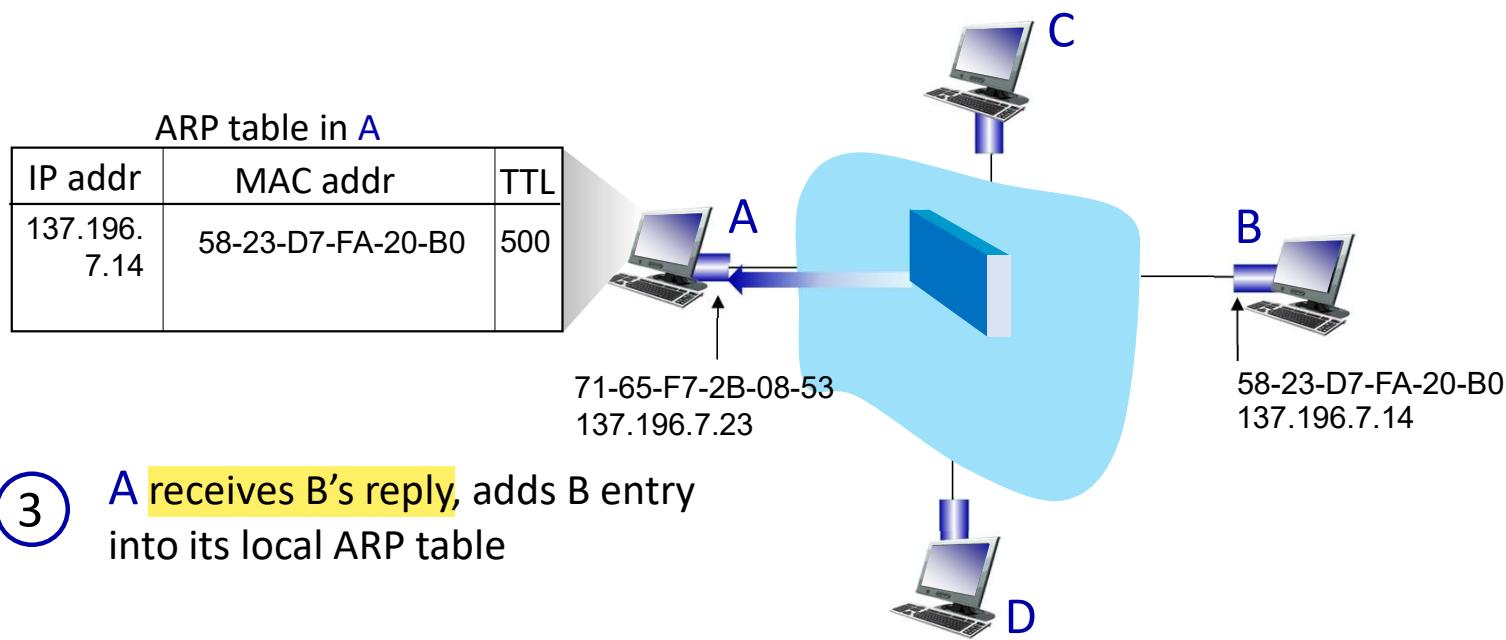
- B's MAC address not in A's ARP table, so A uses ARP to find B's MAC address



ARP protocol in action

example: A wants to send datagram to B

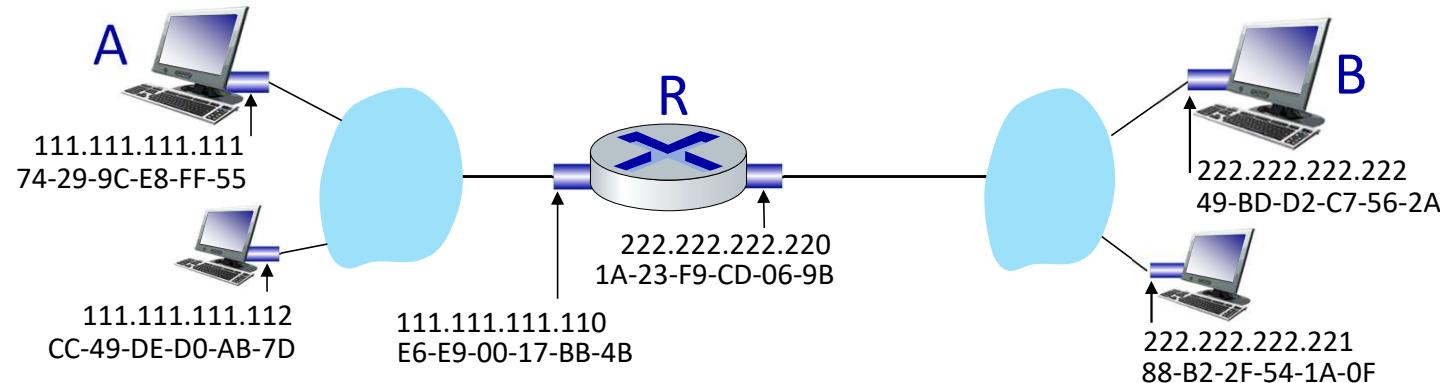
- B's MAC address not in A's ARP table, so A uses ARP to find B's MAC address



Routing to another subnet: addressing

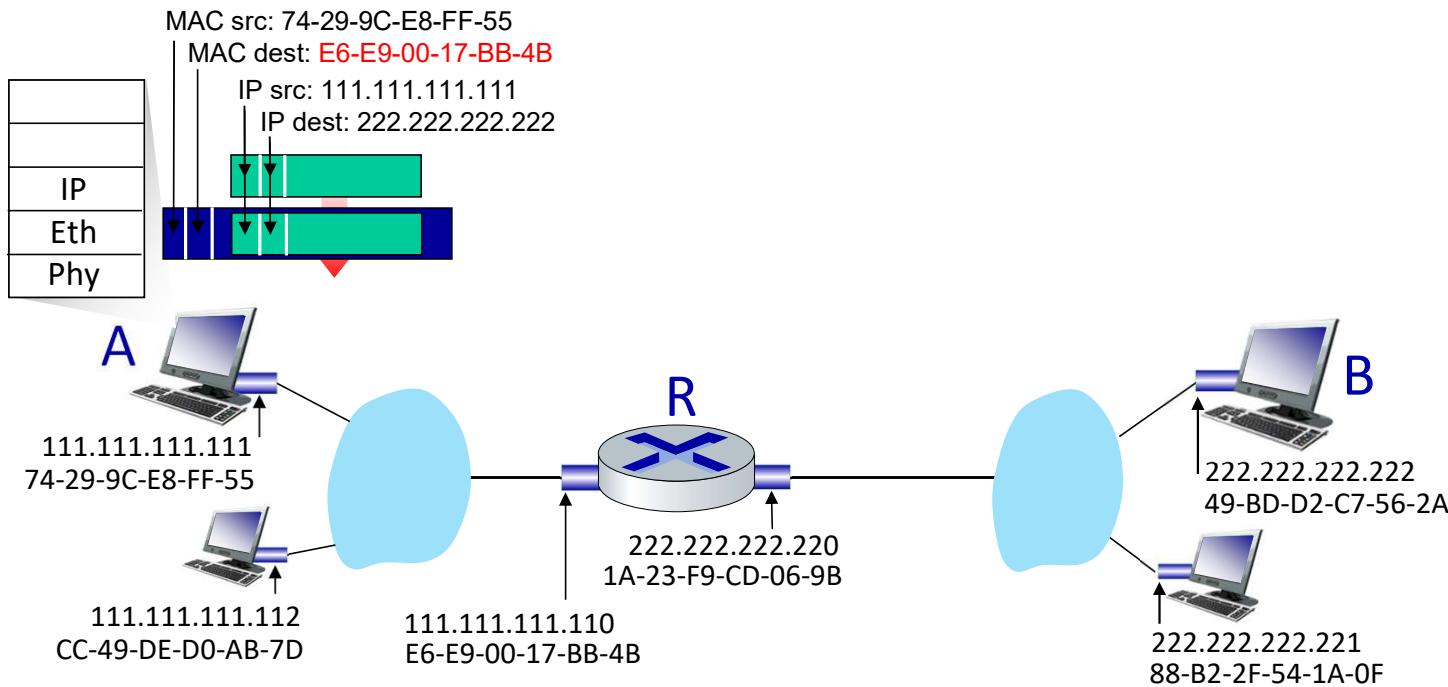
walkthrough: sending a datagram from A to B via R

- focus on addressing – at IP (datagram) and MAC layer (frame) levels
- assume that:
 - A knows B's IP address
 - A knows IP address of first hop router, R (how?)
 - A knows R's MAC address (how?)



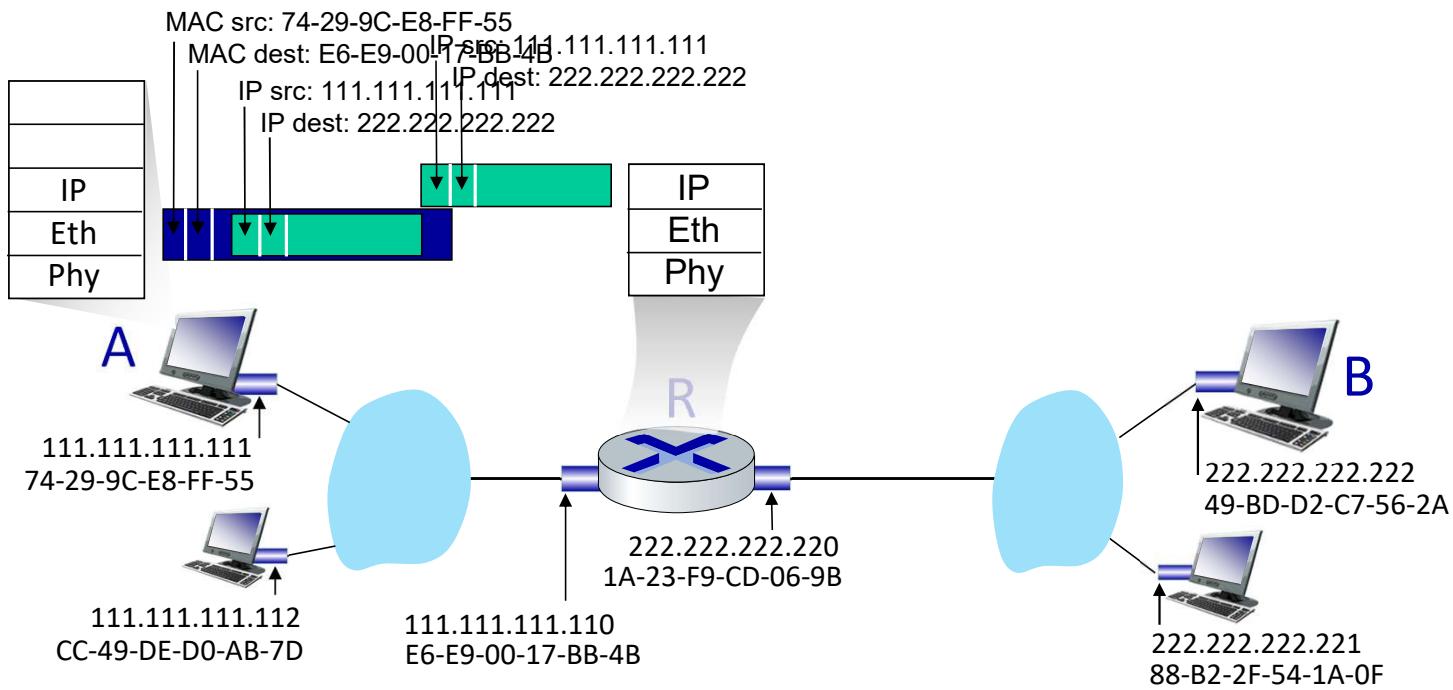
Routing to another subnet: addressing

- A creates IP datagram with IP source A, destination B
- A creates link-layer frame containing A-to-B IP datagram
 - R's MAC address is frame's destination



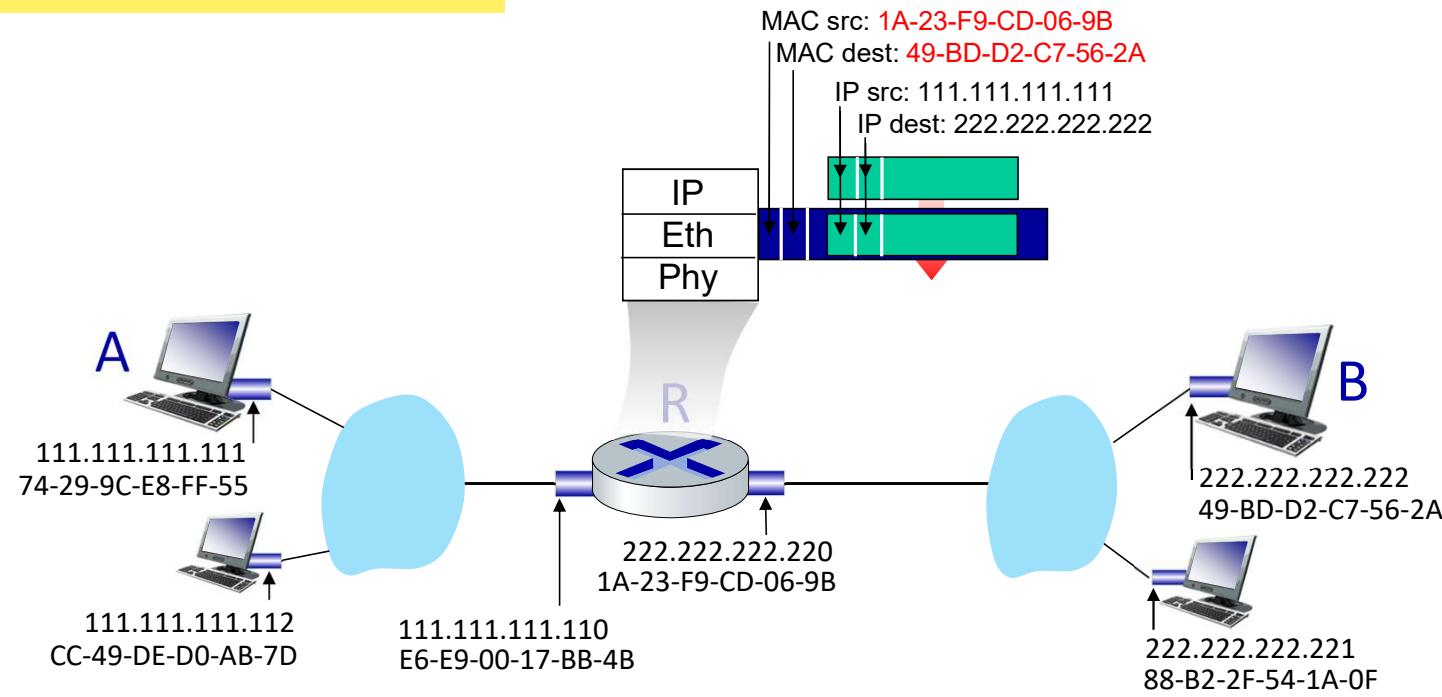
Routing to another subnet: addressing

- frame sent from A to R
- frame received at R, datagram removed, passed up to IP



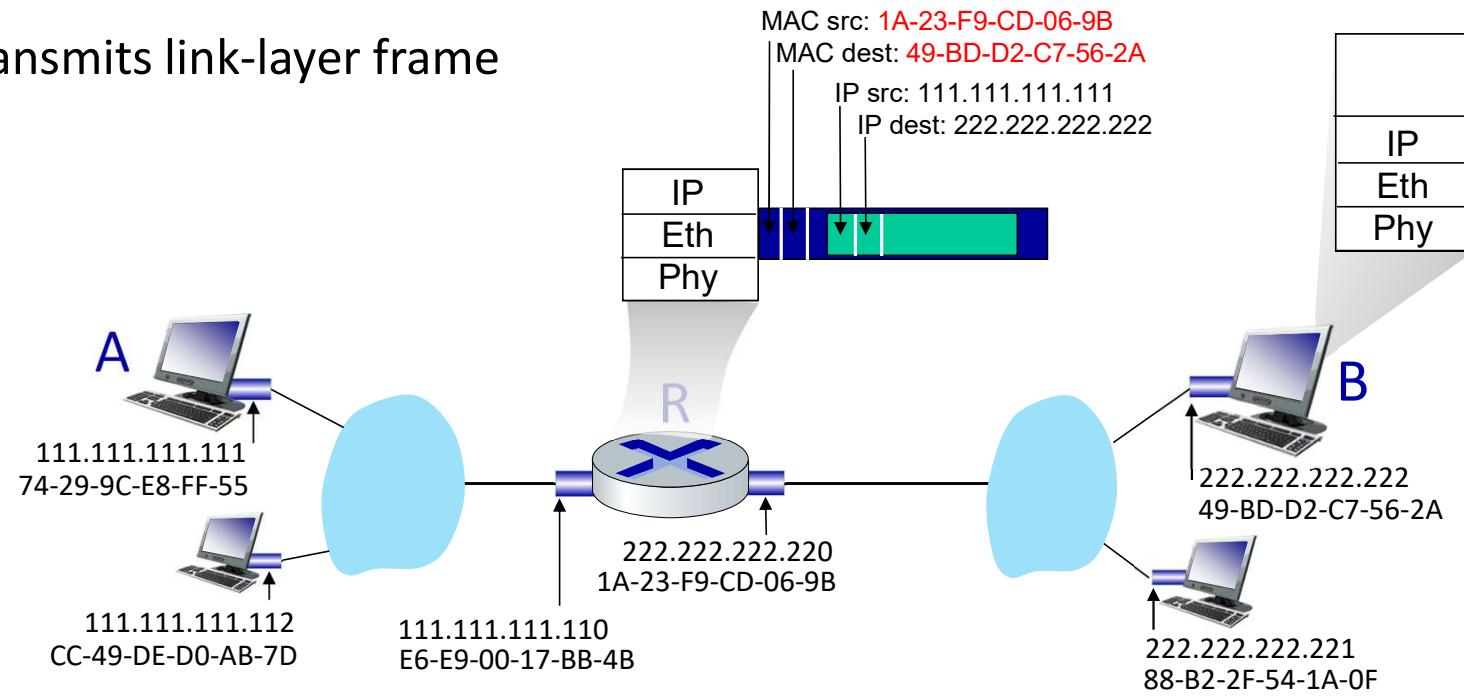
Routing to another subnet: addressing

- R determines outgoing interface, passes datagram with IP source A, destination B to link layer
- R creates link-layer frame containing A-to-B IP datagram. **Frame destination address: B's MAC address**



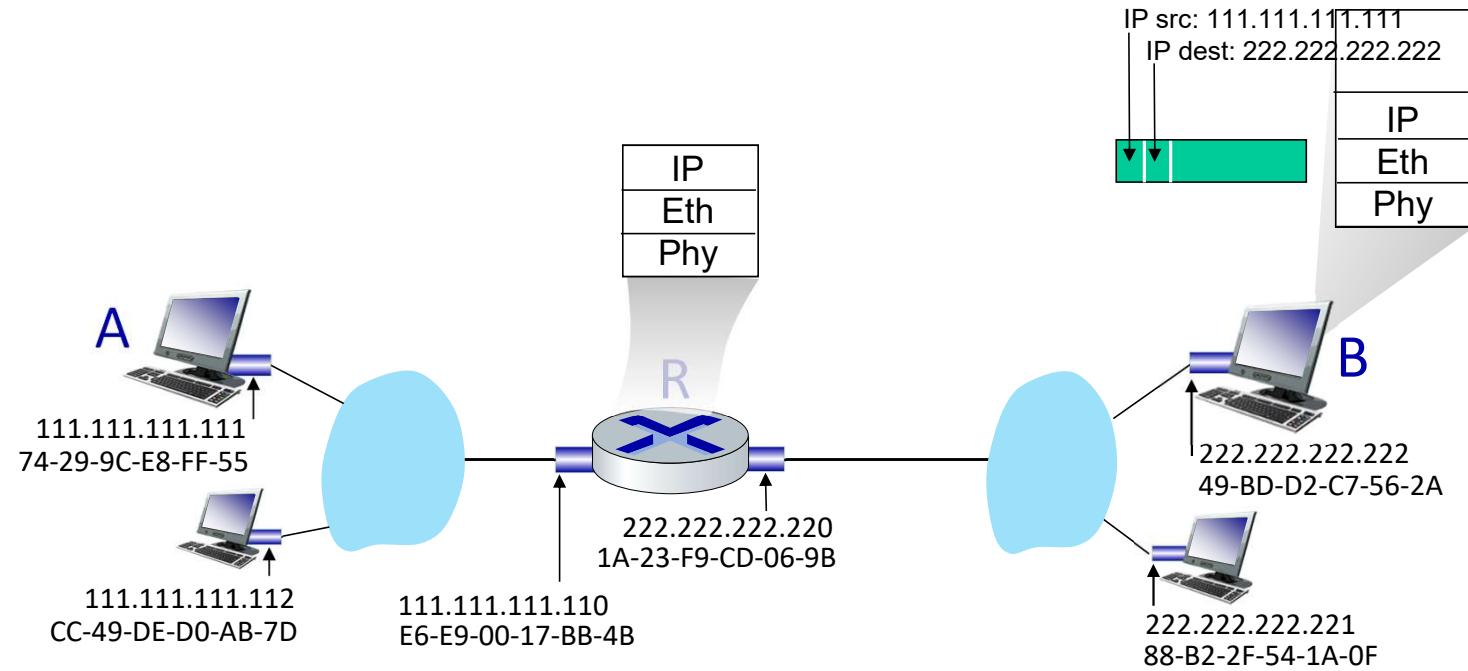
Routing to another subnet: addressing

- R determines outgoing interface, passes datagram with IP source A, destination B to link layer
- R creates link-layer frame containing A-to-B IP datagram. Frame destination address: B's MAC address
- transmits link-layer frame



Routing to another subnet: addressing

- B receives frame, extracts IP datagram destination B
- B passes datagram up protocol stack to IP



Link layer, LANs: roadmap

■ LANs

- addressing, ARP
- Ethernet
- switches
- VLANs



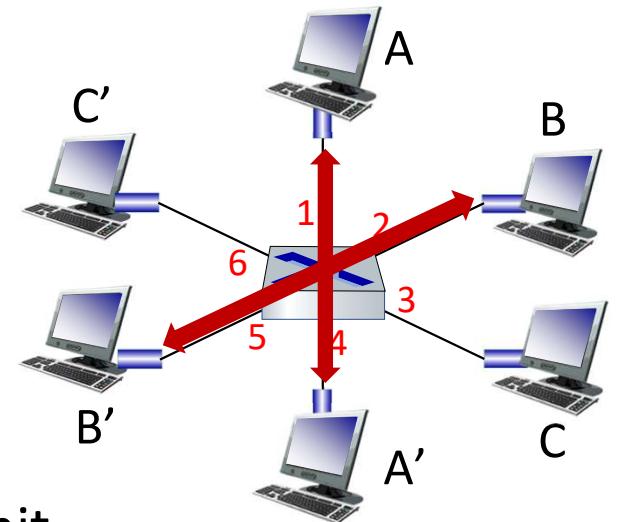
- a day in the life of a web request

Switch

- Switch is a **link-layer** device: takes an *active* role
 - store, forward Ethernet (or other type of) frames
 - examine incoming frame's MAC address, *selectively forward frame to one-or-more outgoing links* when frame is *to be forwarded on segment*, uses *CSMA/CD* to access segment
- **transparent:** hosts *unaware* of presence of switches
- **plug-and-play, self-learning**
 - switches do not need to be configured

Switch: multiple simultaneous transmissions

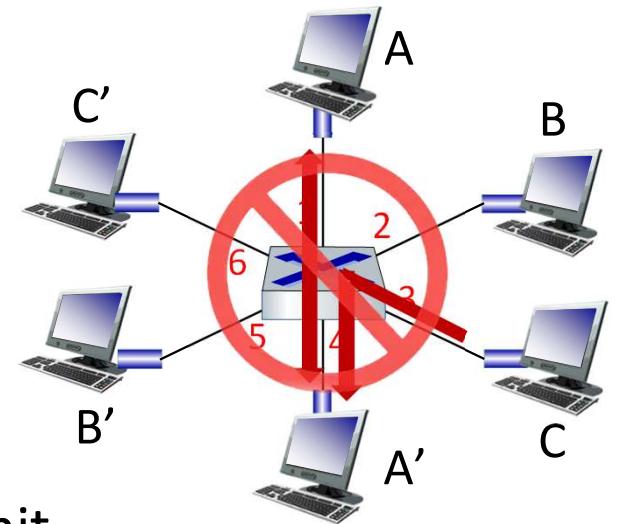
- hosts have **dedicated, direct connection to switch**
- switches **buffer packets**
- Ethernet protocol used on *each* incoming link, so:
 - no collisions; **full duplex**
 - **each link is its own collision domain**
- **switching:** A-to-A' and B-to-B' can transmit simultaneously, without collisions



switch with six
interfaces (1,2,3,4,5,6)

Switch: multiple simultaneous transmissions

- hosts have dedicated, direct connection to switch
- switches buffer packets
- Ethernet protocol used on *each* incoming link, so:
 - no collisions; full duplex
 - each link is its own collision domain
- **switching:** A-to-A' and B-to-B' can transmit simultaneously, without collisions
 - but A-to-A' and C to A' can *not* happen simultaneously



switch with six
interfaces (1,2,3,4,5,6)

Switch forwarding table

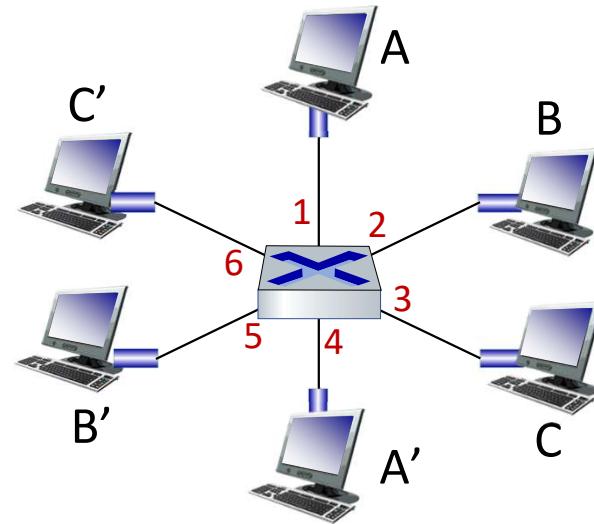
Q: how does switch know A' reachable via interface 4, B' reachable via interface 5?

A: each switch has a **switch table**, each entry:

- (MAC address of host, interface to reach host, time stamp)
- looks like a routing table!

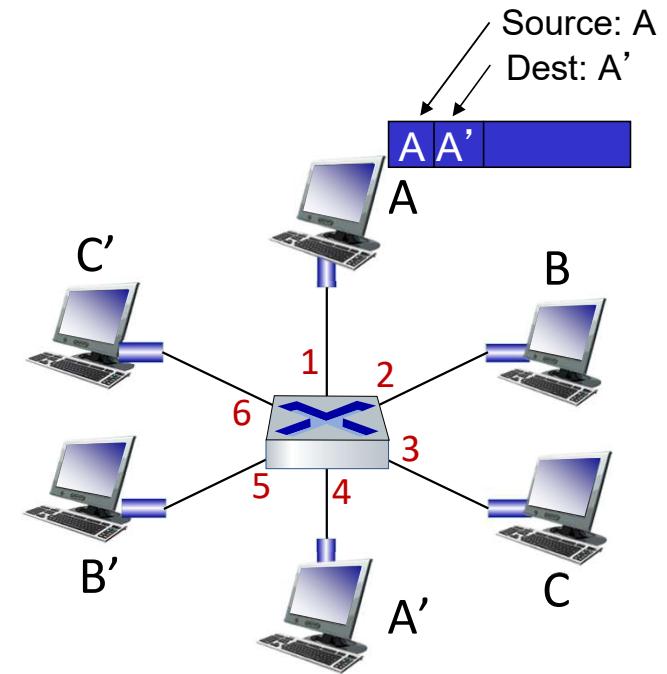
Q: how are entries created, maintained in switch table?

- something like a routing protocol?



Switch: self-learning

- switch **learns** which hosts can be reached through which interfaces
 - when frame received, switch “learns” location of sender: incoming LAN segment
 - records sender/location pair in switch table



MAC addr	interface	TTL
A	1	60

*Switch table
(initially empty)*

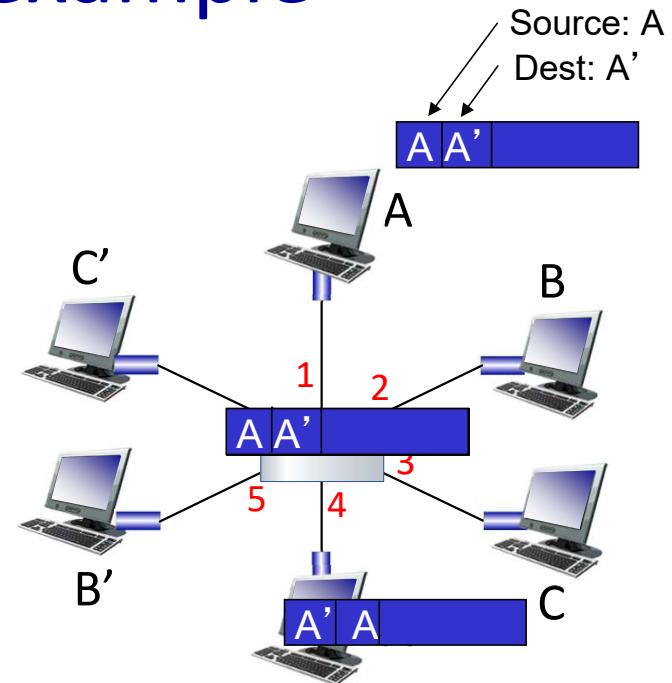
Switch: frame filtering/forwarding

when frame received at switch:

1. record incoming link, MAC address of sending host
2. index switch table using MAC destination address
3. if entry found for destination
 then {
 if destination on segment from which frame arrived
 then drop frame
 else forward frame on interface indicated by entry
 }
 else flood /* forward on all interfaces except arriving interface */

Self-learning, forwarding: example

- frame destination, A', location unknown: **flood**
- destination A location known: **selectively send on just one link**

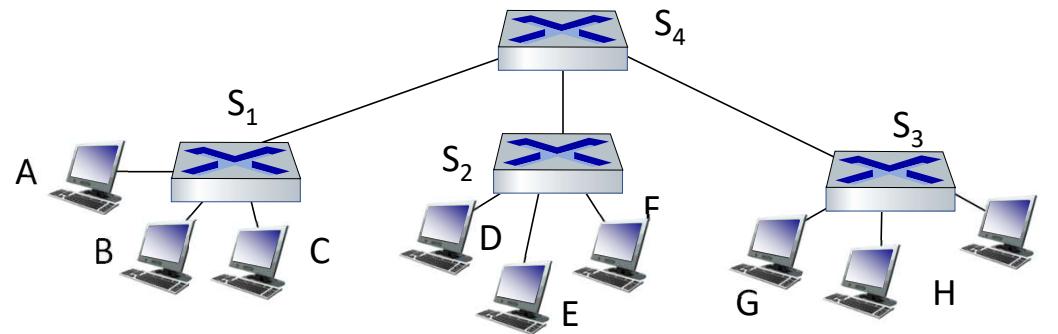


MAC addr	interface	TTL
A	1	60
A'	4	60

*switch table
(initially empty)*

Interconnecting switches

self-learning switches can be connected together:

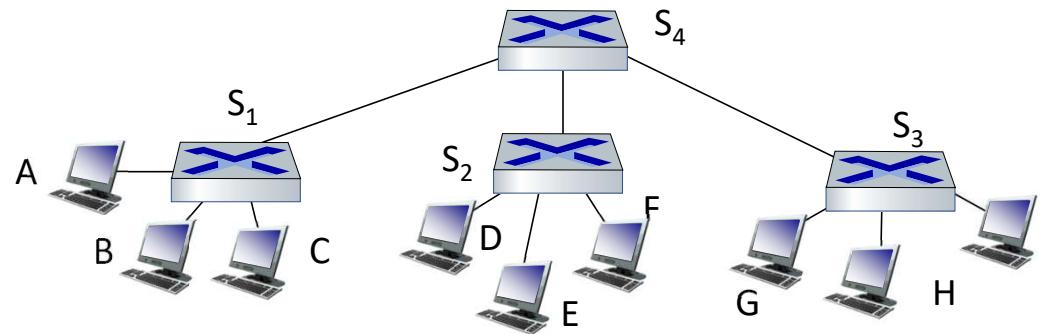


Q: sending from A to G - how does S_1 know to forward frame destined to G via S_4 and S_3 ?

- **A:** self learning! (works exactly the same as in single-switch case!)

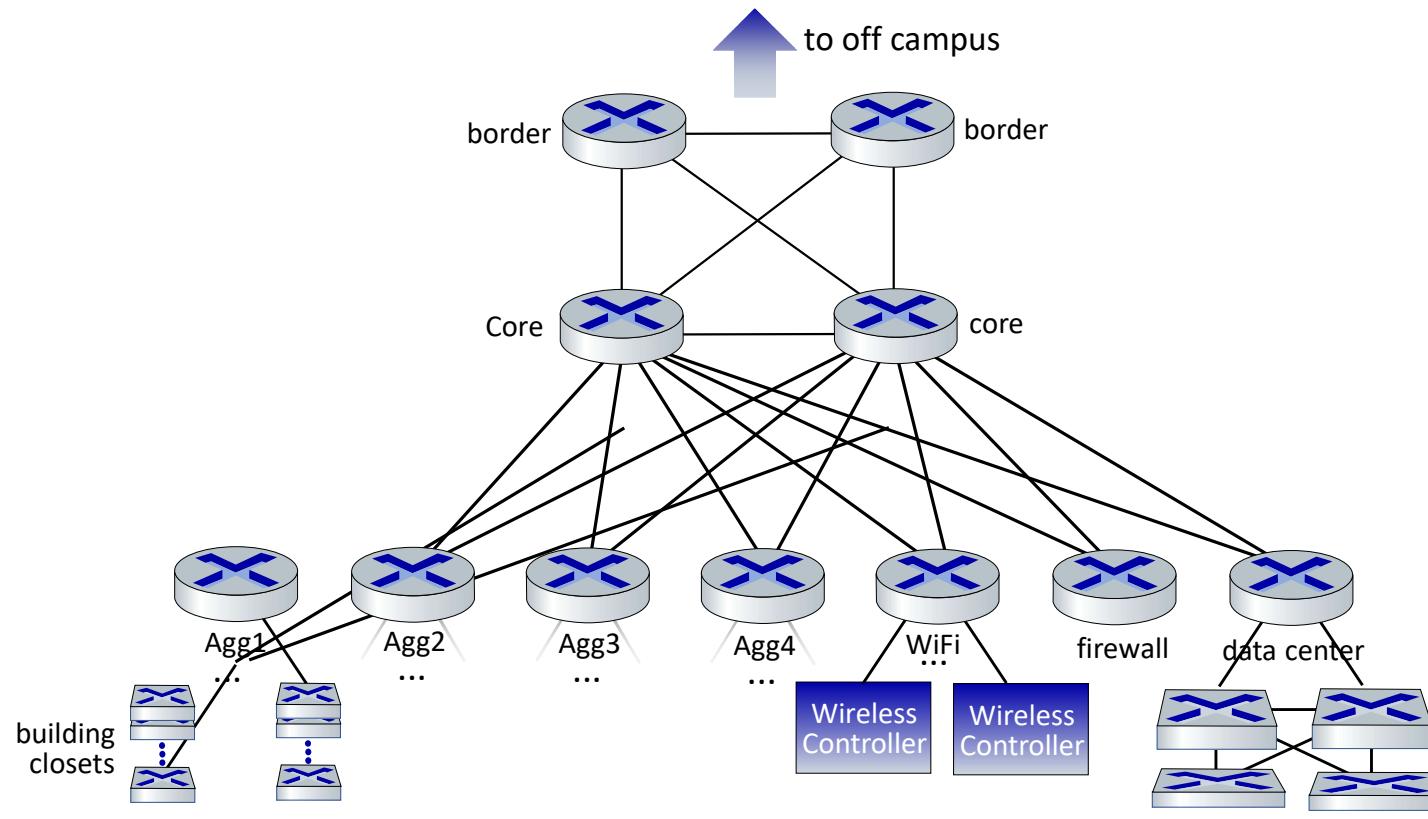
Self-learning multi-switch example

Suppose C sends frame to I, I responds to C



Q: show switch tables and packet forwarding in S_1, S_2, S_3, S_4

UMass Campus Network - Detail

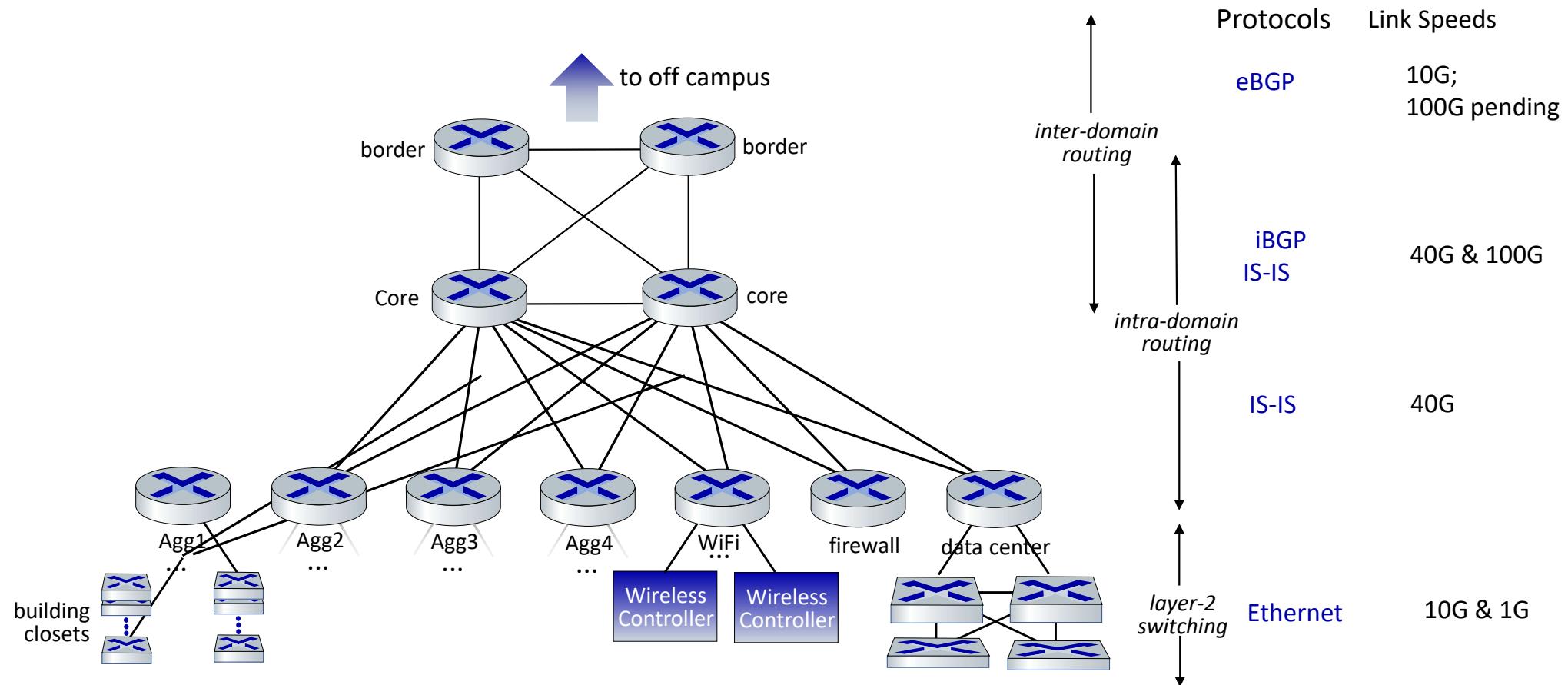


UMass network:

- 4 firewalls
- 10 routers
- 2000+ network switches
- 6000 wireless access points
- 30000 active wired network jacks
- 55000 active end-user wireless devices

... all built,
operated,
maintained by ~15
people

UMass Campus Network - Detail



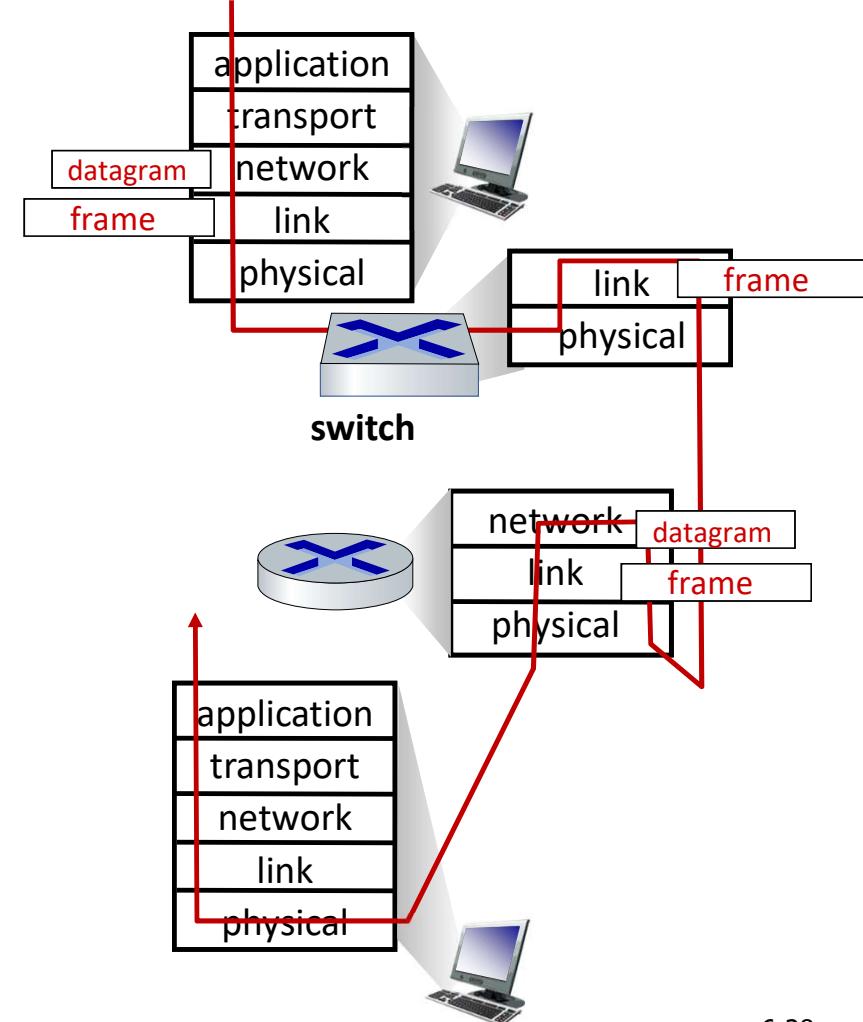
Switches vs. routers

both are **store-and-forward**:

- *routers*: network-layer devices (examine network-layer headers)
- *switches*: link-layer devices (examine link-layer headers)

both have **forwarding tables**:

- *routers*: compute tables using **routing algorithms, IP addresses**
- *switches*: learn forwarding table using **flooding, learning, MAC addresses**



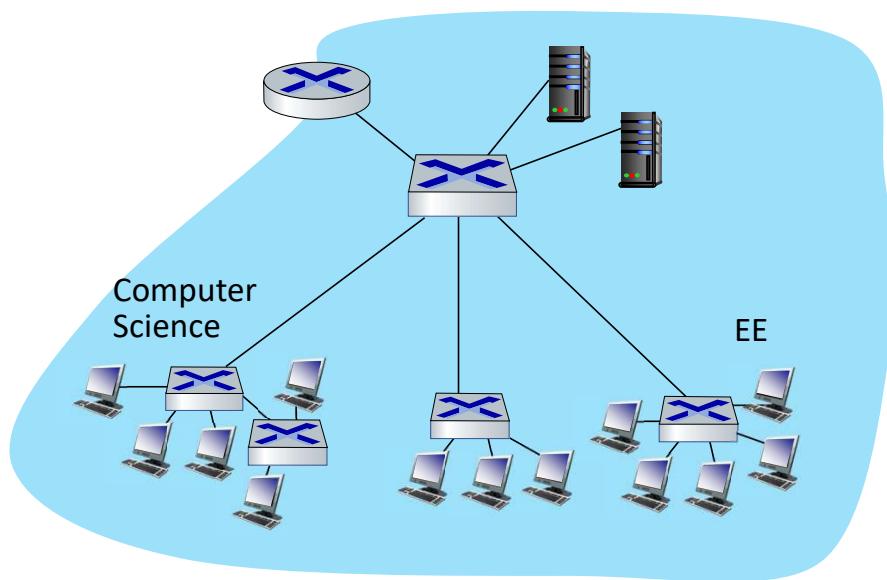
Link layer, LANs: roadmap

- LANs
 - addressing, ARP
 - switches
 - VLANs
- link virtualization: MPLS



Virtual LANs (VLANs): motivation

Q: what happens as LAN sizes scale, users change point of attachment?

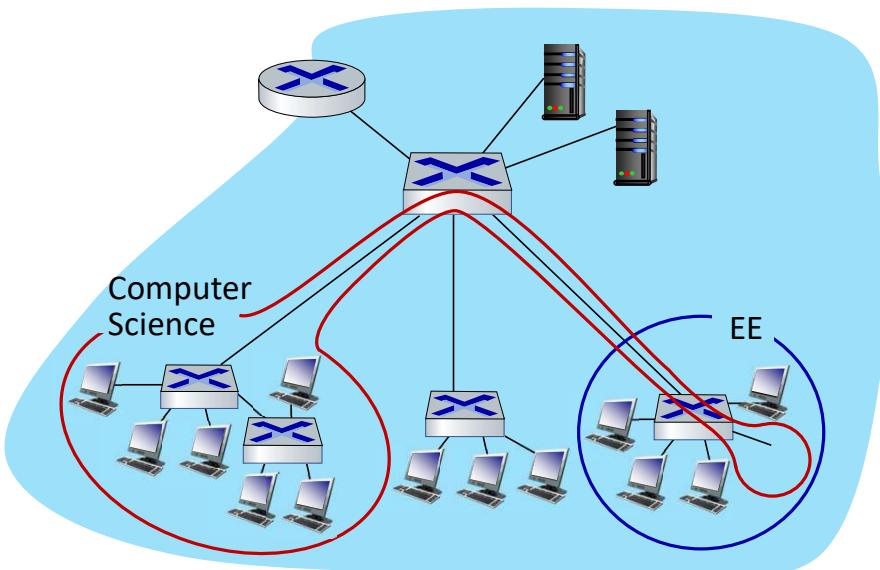


single broadcast domain:

- *scaling*: all layer-2 broadcast traffic (ARP, DHCP, unknown MAC) must cross entire LAN
- efficiency, security, privacy issues

Virtual LANs (VLANs): motivation

Q: what happens as LAN sizes scale, users change point of attachment?



single broadcast domain:

- *scaling*: all layer-2 broadcast traffic (ARP, DHCP, unknown MAC) must cross entire LAN
- efficiency, security, privacy, efficiency issues

administrative issues:

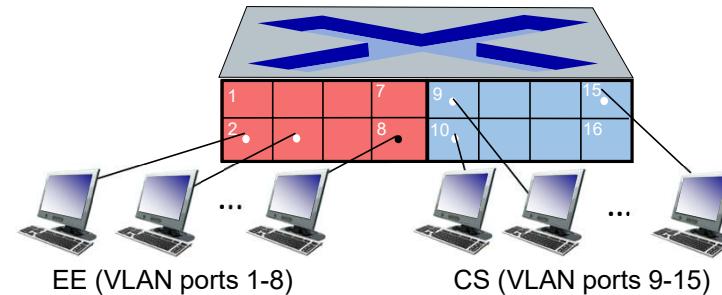
- CS user moves office to EE - *physically* attached to EE switch, but wants to remain *logically* attached to CS switch

Port-based VLANs

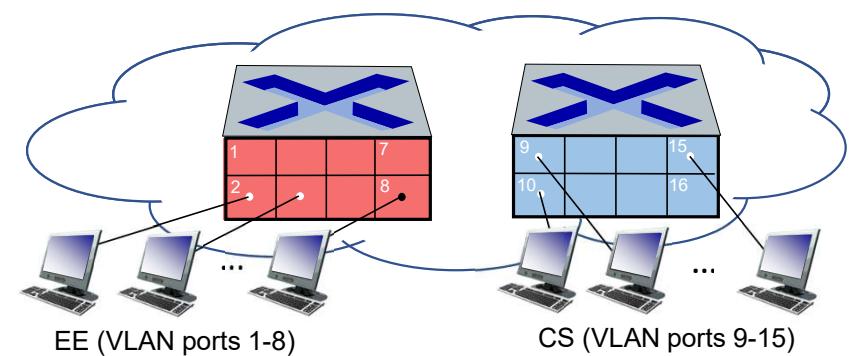
Virtual Local Area Network (VLAN)

switch(es) supporting VLAN capabilities can be configured to define multiple *virtual LANs* over single physical LAN infrastructure.

port-based VLAN: switch ports grouped (by switch management software) so that *single* physical switch

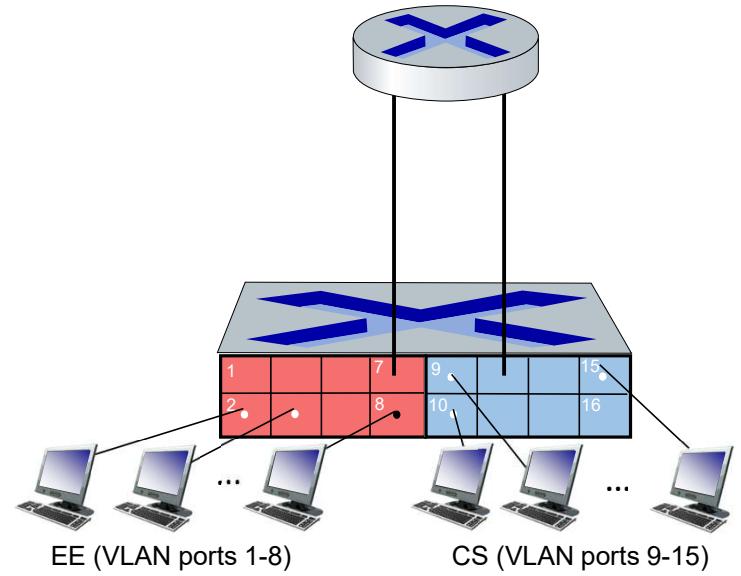


... operates as **multiple virtual switches**

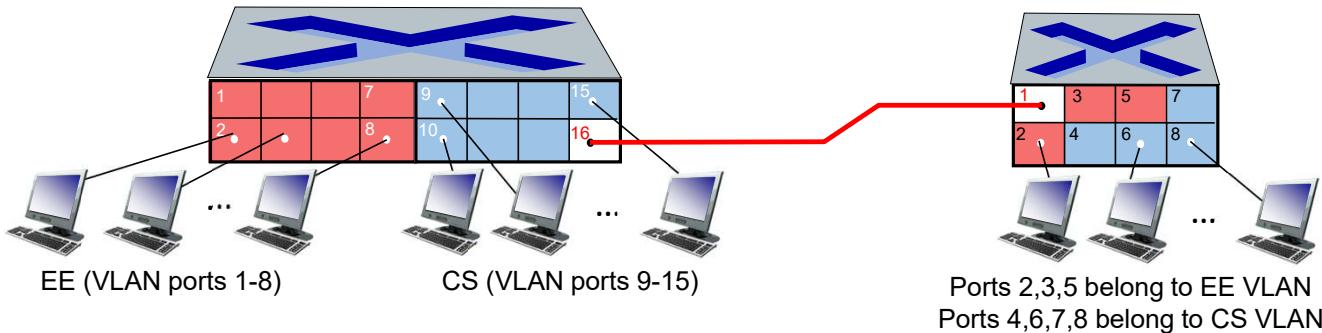


Port-based VLANs

- **traffic isolation:** frames to/from ports 1-8 can *only* reach ports 1-8
 - can also define VLAN based on MAC addresses of endpoints, rather than switch port
- **dynamic membership:** ports can be dynamically assigned among VLANs
- **forwarding between VLANs:** done via routing (just as with separate switches)
 - in practice vendors sell combined switches plus routers



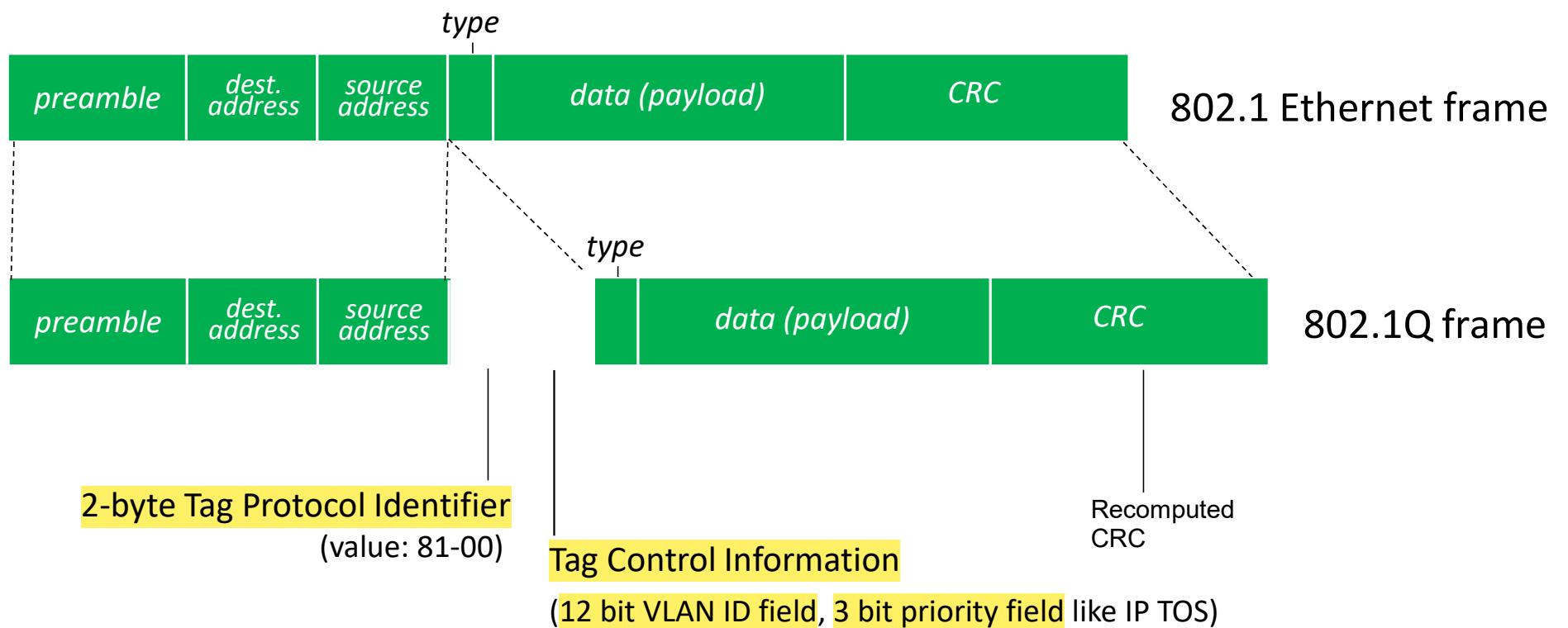
VLANS spanning multiple switches



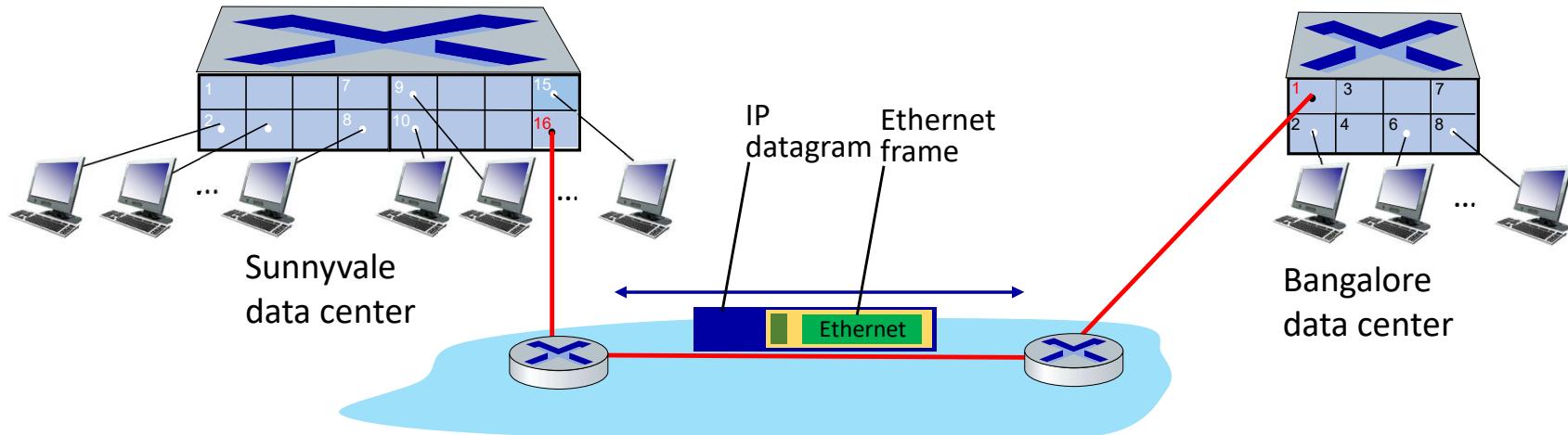
trunk port: carries frames between VLANS defined over multiple physical switches

- frames forwarded within VLAN between switches can't be vanilla 802.1 frames (must carry VLAN ID info)
- 802.1q protocol adds/removed additional header fields for frames forwarded between trunk ports

802.1Q VLAN frame format



EVPN: Ethernet VPNs (aka VXLANs)



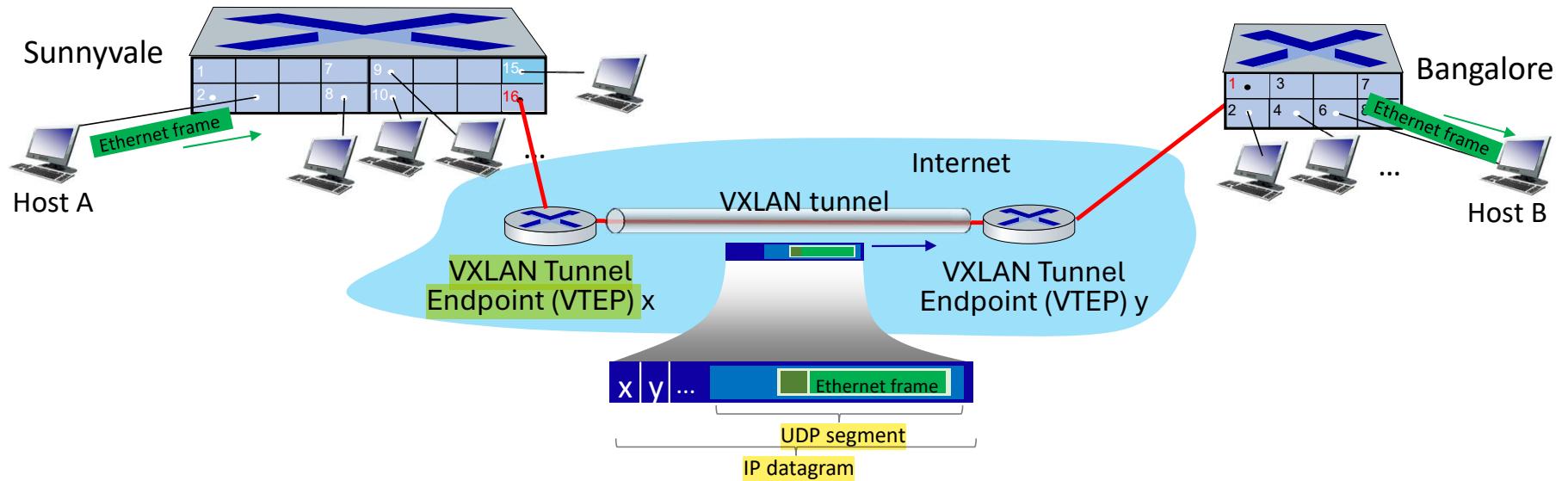
Layer-2 Ethernet switches *logically* connected to each other (e.g., using IP as an *underlay*)

- Ethernet frames carried *within* IP datagrams between sites
- “*tunneling* scheme to *overlay Layer 2 networks on top of Layer 3 networks* ... runs over the existing networking infrastructure and provides a means to “*stretch*” a Layer 2 network.” [RFC 7348]

EVPN: Ethernet VPNs (aka VXLANs)

Layer-2 Ethernet switches *logically* connected to each other (using IP as an *underlay*)

- Ethernet frames carried *within* IP datagrams between sites
- “*tunneling* scheme to *overlay Layer 2 networks on top of Layer 3 networks* ... runs over existing networking infrastructure ... provides a means to “stretch” a Layer 2 network.” [RFC 7348]



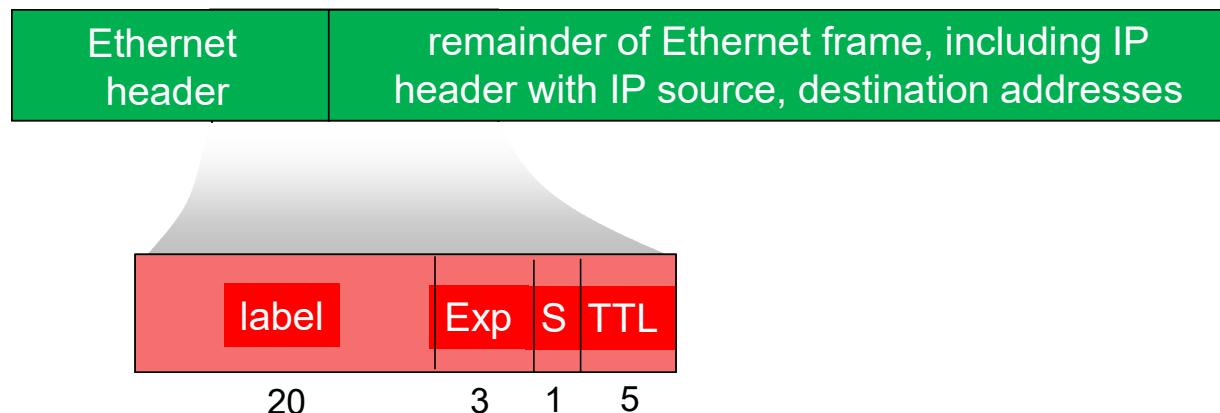
Link layer, LANs: roadmap

- LANs
 - addressing, ARP
 - switches
 - VLANs
- link virtualization: MPLS



Multiprotocol label switching (MPLS)

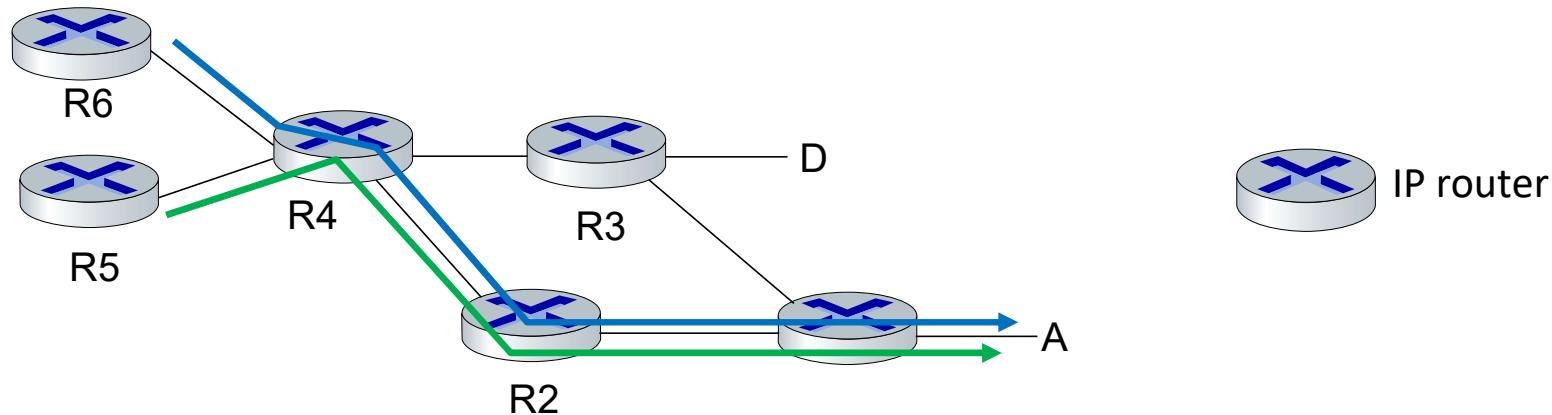
- goal: high-speed IP forwarding among network of MPLS-capable routers, using fixed length label (instead of shortest prefix matching)
 - faster lookup using fixed length identifier
 - borrowing ideas from Virtual Circuit (VC) approach
 - but IP datagram still keeps IP address!



MPLS capable routers

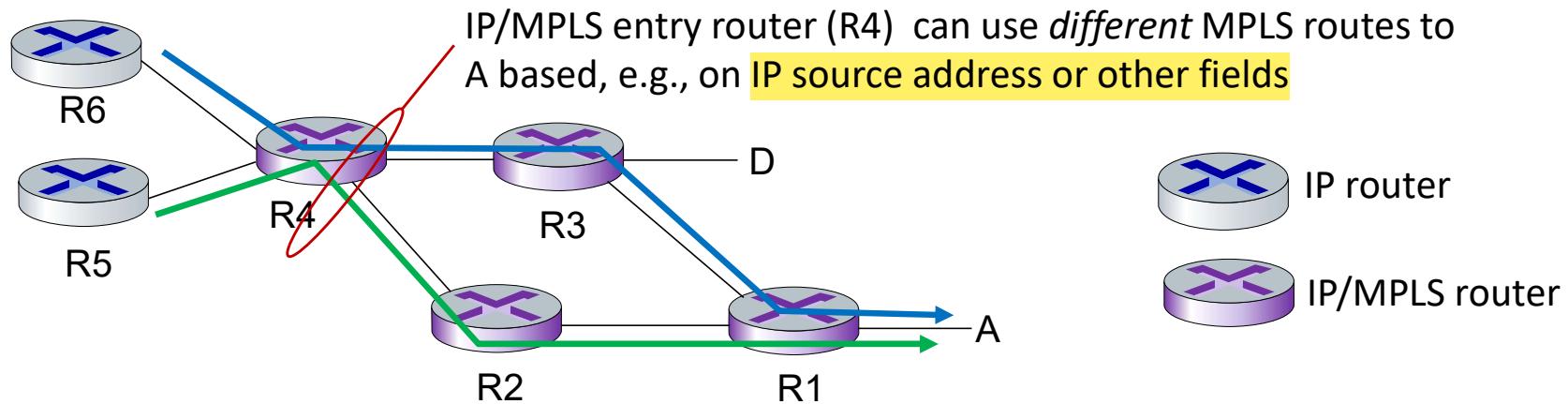
- a.k.a. label-switched router
- forward packets to outgoing interface based only on label value (*don't inspect IP address*)
 - MPLS forwarding table distinct from IP forwarding tables
- *flexibility*: MPLS forwarding decisions can *differ* from those of IP
 - use destination *and* source addresses to route flows to same destination differently (*traffic engineering*)
 - re-route flows quickly if link fails: pre-computed backup paths

MPLS versus IP paths



- **IP routing:** path to destination determined by destination address alone

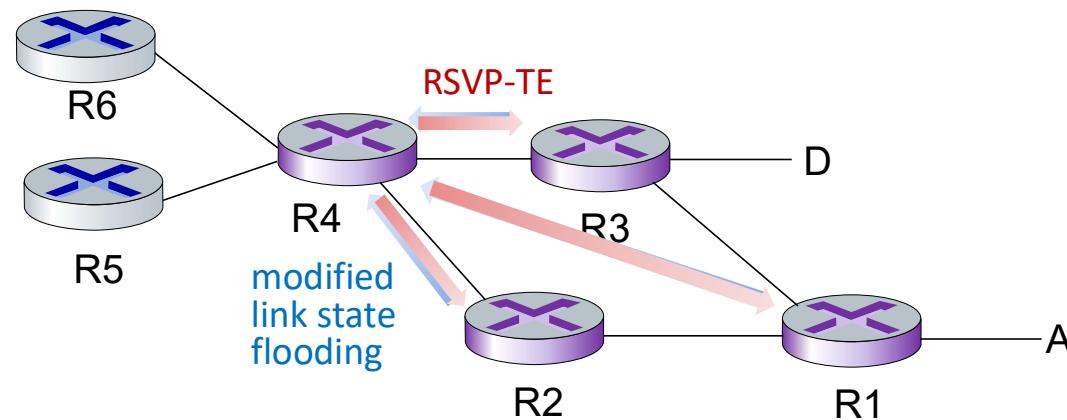
MPLS versus IP paths



- **IP routing:** path to destination determined by destination address alone
- **MPLS routing:** path to destination can be based on source *and* destination address
 - flavor of generalized forwarding (MPLS 10 years earlier)
 - *fast reroute:* precompute backup routes in case of link failure

MPLS signaling

- modify OSPF, IS-IS link-state flooding protocols to carry info used by MPLS routing:
 - e.g., link bandwidth, amount of “reserved” link bandwidth
- entry MPLS router uses RSVP-TE signaling protocol to set up MPLS forwarding at downstream routers



MPLS forwarding tables

