Chapter 6
The Link Layer and LANs

A note on the use of these PowerPoint slides:

We're making these slides freely available to all (faculty, students, readers). They're in PowerPoint form so you see the animations; and can add, modify, and delete slides (including this one) and slide content to suit your needs. They obviously represent a *lot* of work on our part. In return for use, we only ask the following:

- If you use these slides (e.g., in a class) that you mention their source (after all, we'd like people to use our book!)
- If you post any slides on a www site, that you note that they are adapted from (or perhaps identical to) our slides, and note our copyright of this material.

For a revision history, see the slide note for this page.

Thanks and enjoy! JFK/KWR
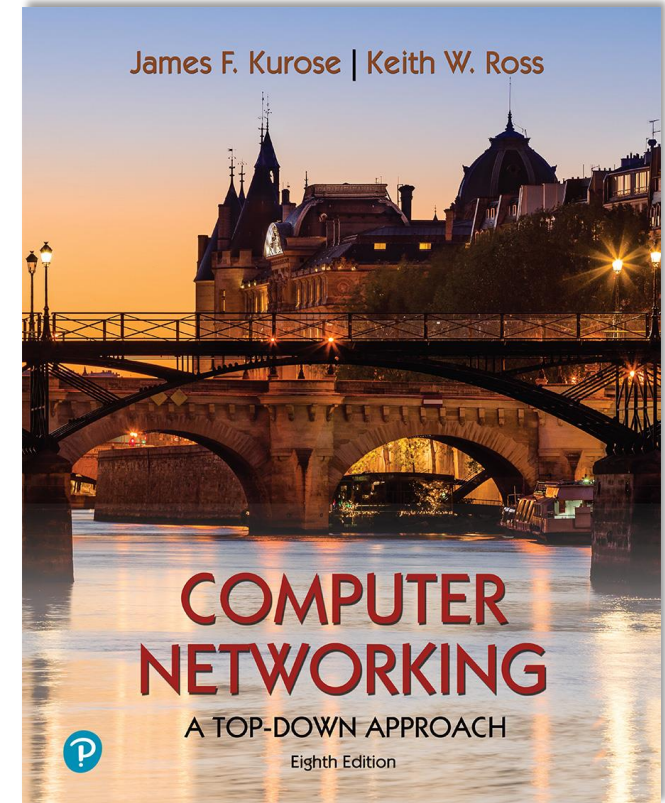
Computer Networking: A Top-Down Approach

8th edition
Jim Kurose, Keith Ross
Pearson, 2020

Link layer and LANs: our goals

- **understand principles behind link layer services:**
  - error detection, correction
  - sharing a broadcast channel: multiple access
  - link layer addressing
  - local area networks: Ethernet, VLANs
- **datacenter networks**

- **instantiation, implementation of various link layer technologies**

Link layer, LANs: roadmap

- introduction
- error detection, correction
- multiple access protocols
- **LANs**
  - addressing, ARP
  - Ethernet
  - **switches**
  - VLANs
- link virtualization: MPLS
- data center networking

- a day in the life of a web request

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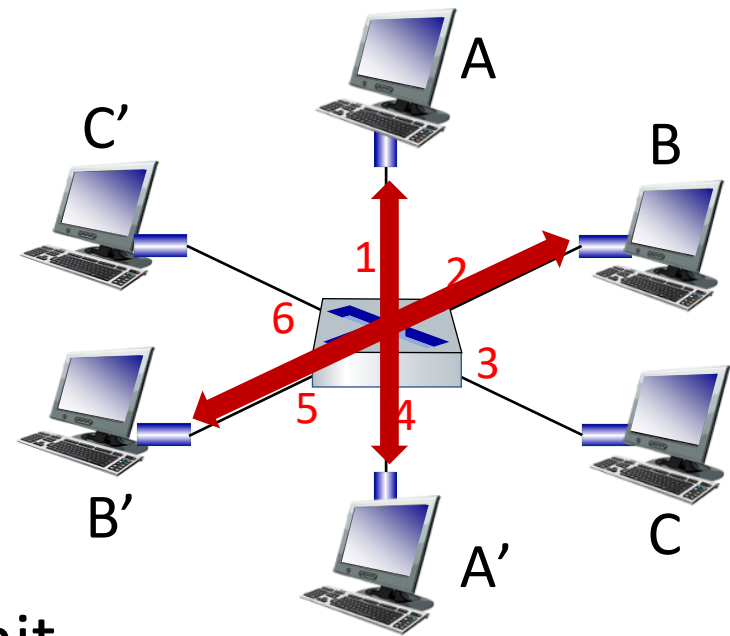
switch with six interfaces (1,2,3,4,5,6)

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

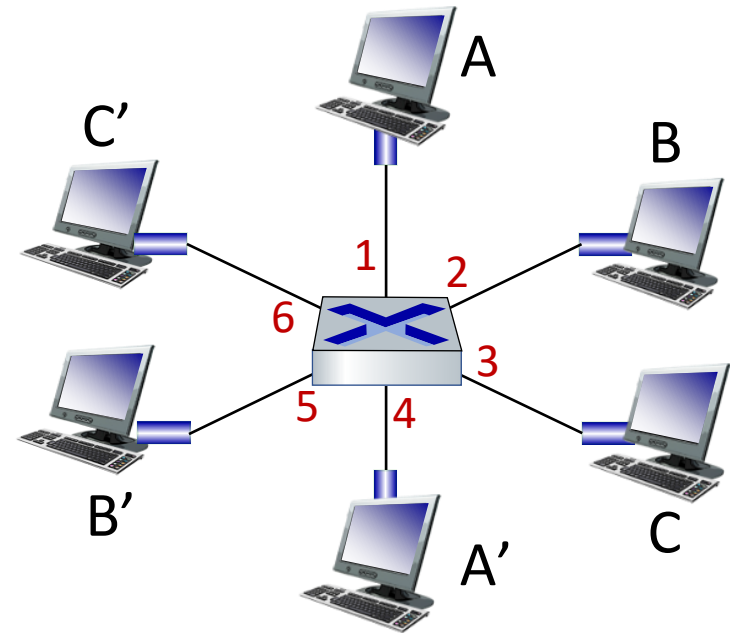switch with six interfaces (1,2,3,4,5,6)
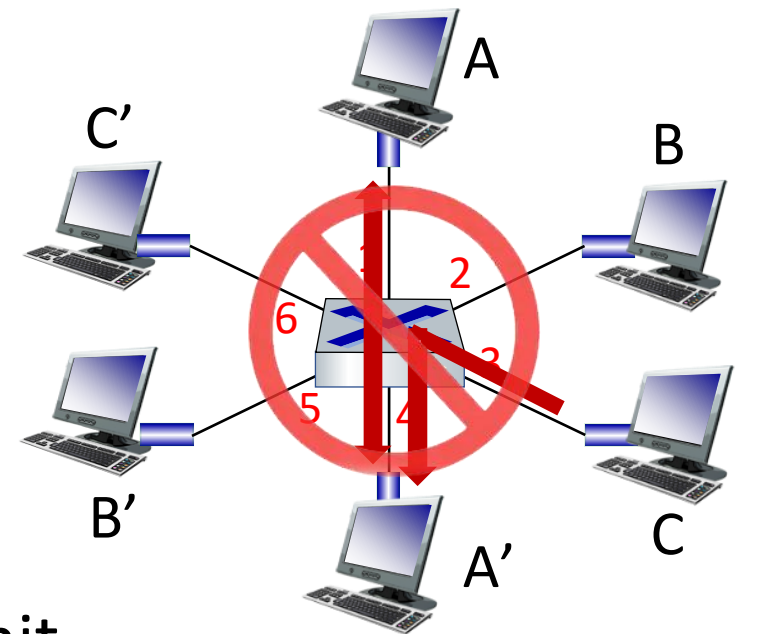
Switch forwarding table

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)
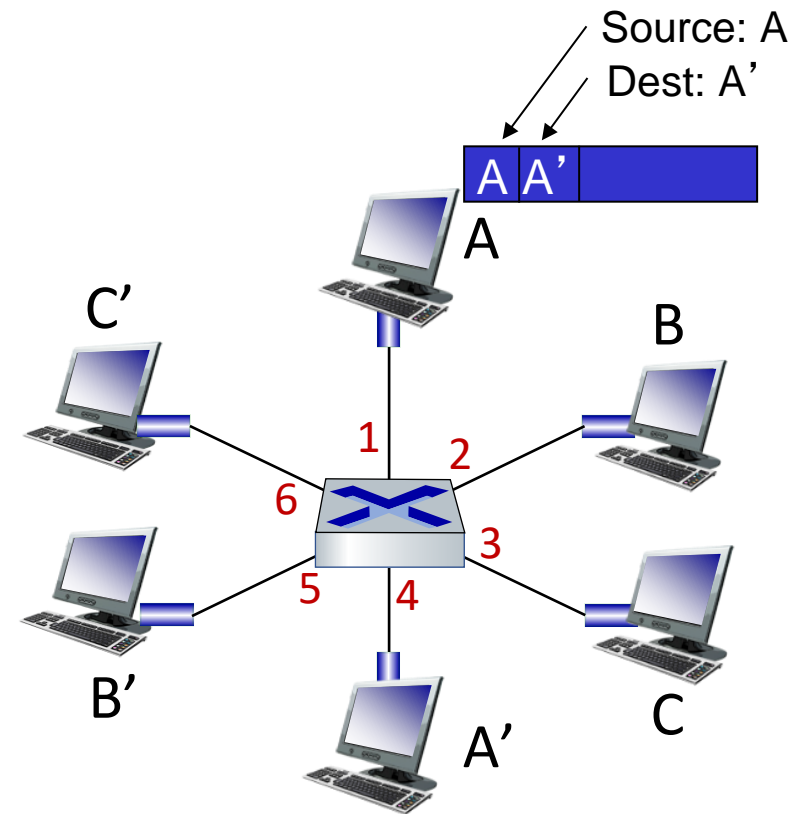- looks like a routing table!

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

Source: A
Dest: A'

A A'

A

C'

B

1 2

6

3

5 4

B'

C

A'

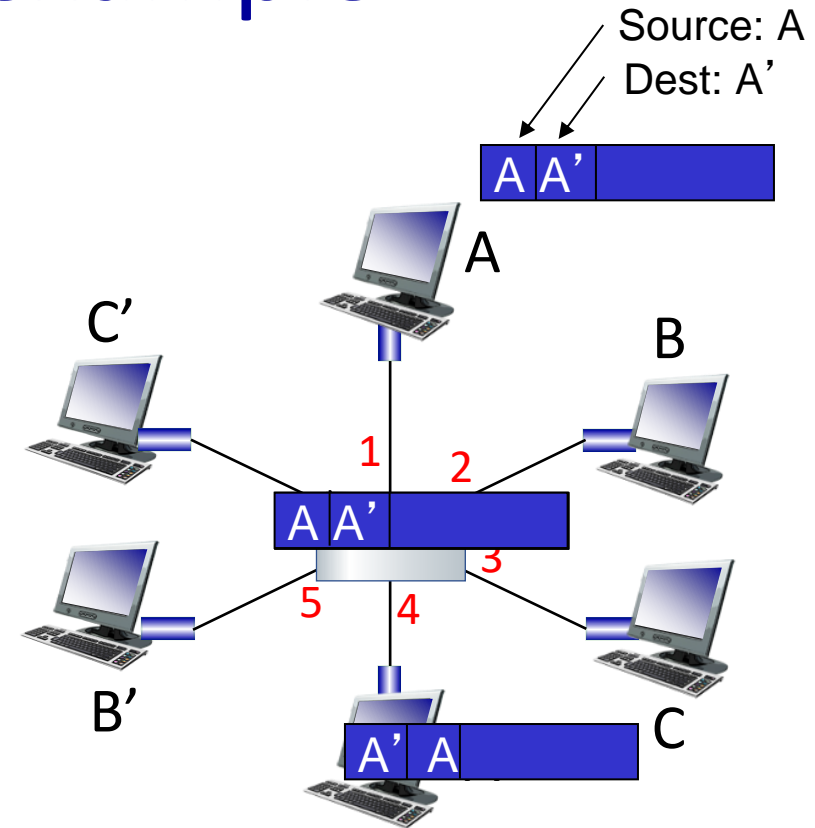| MAC addr	interface	TTL
A	1	60

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

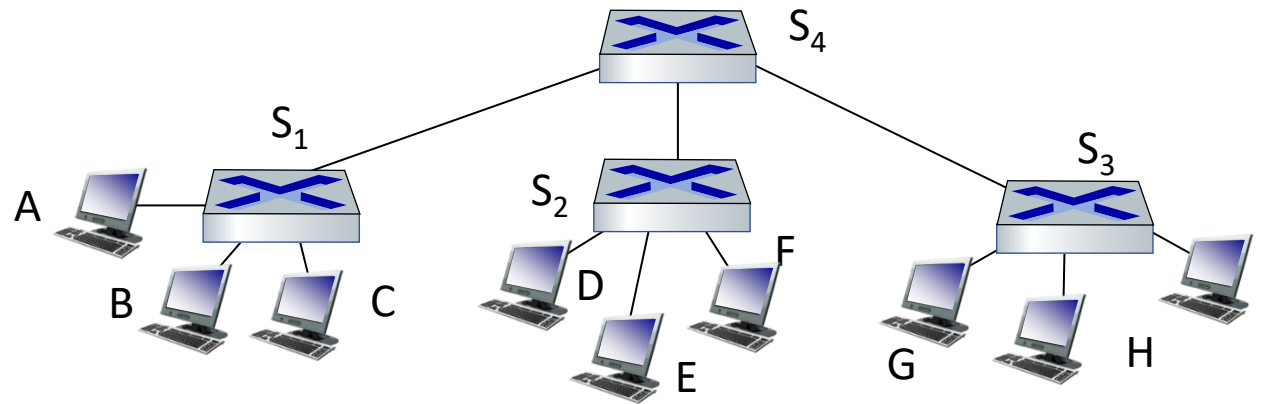- destination A location known: selectively send on just one link

Source: A
Dest: A'

MAC addr	interface	TTL
A	1	60
A'	4	60

switch table (initially empty)

Interconnecting switches

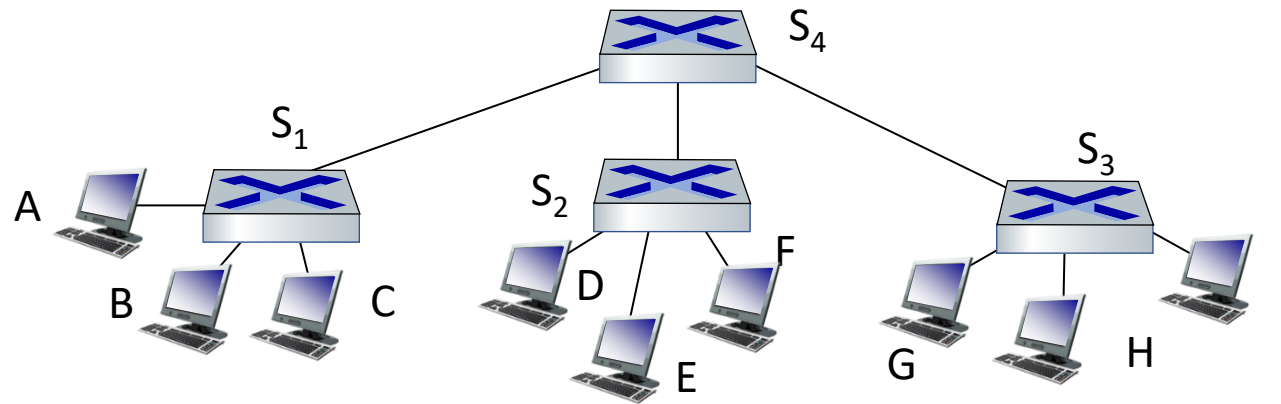self-learning switches can be connected together:

$Q:$ sending from A to G - how does S_1 know to forward frame destined to G via S_4 and S_3?
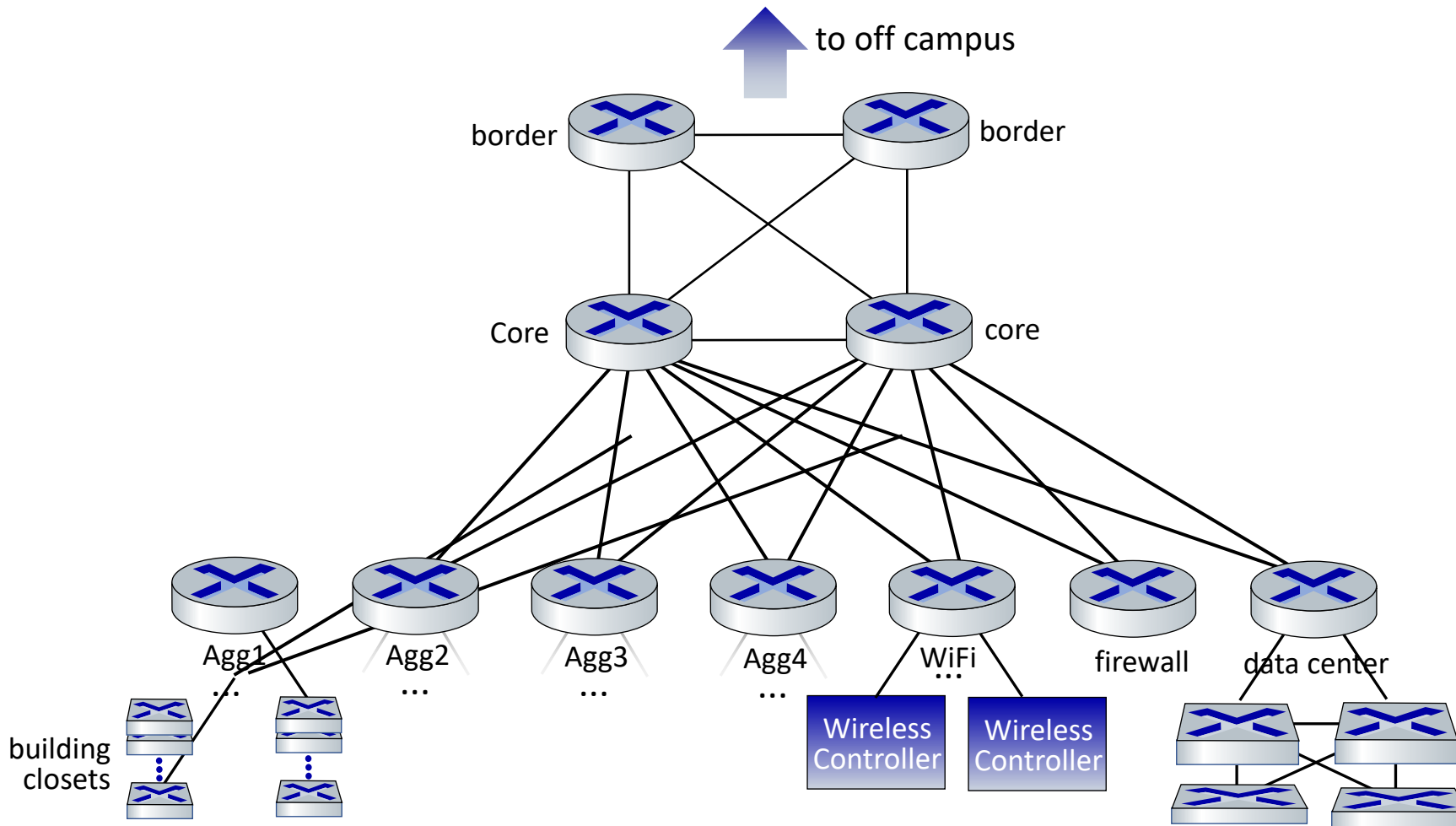
- $A:$ self learning! (works exactly the same as in single-switch case!)

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
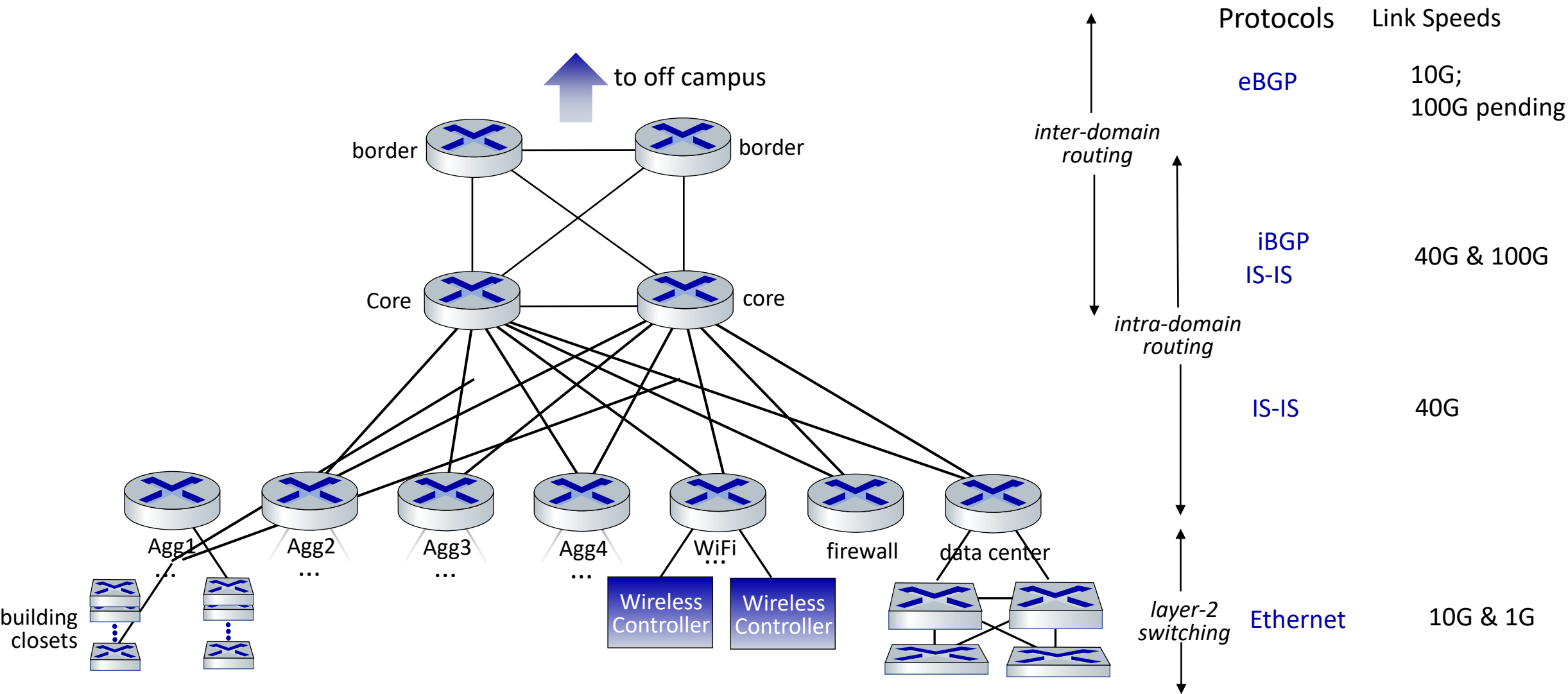
UMass Campus Network - Detail

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
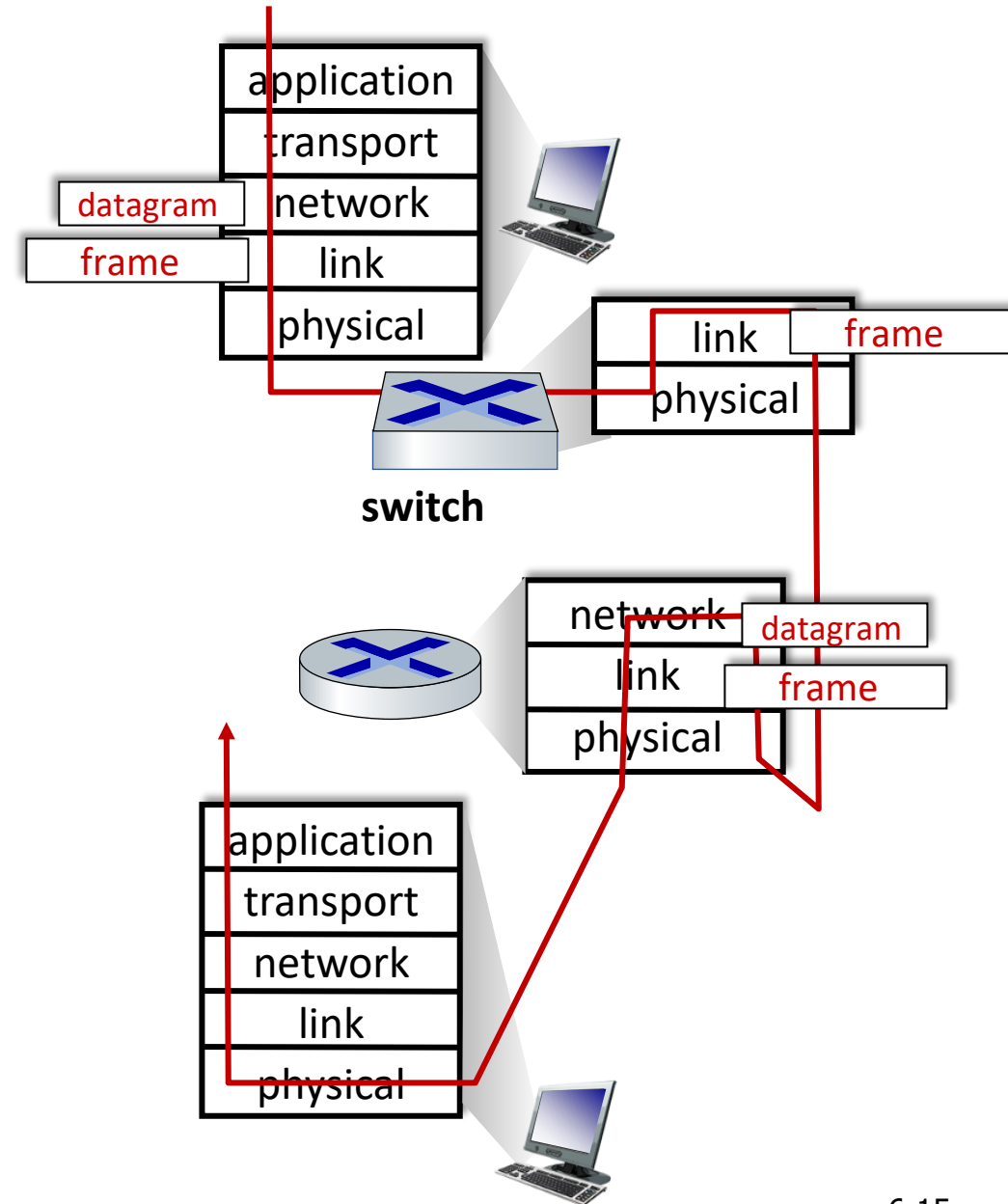
UMass Campus Network - Detail

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
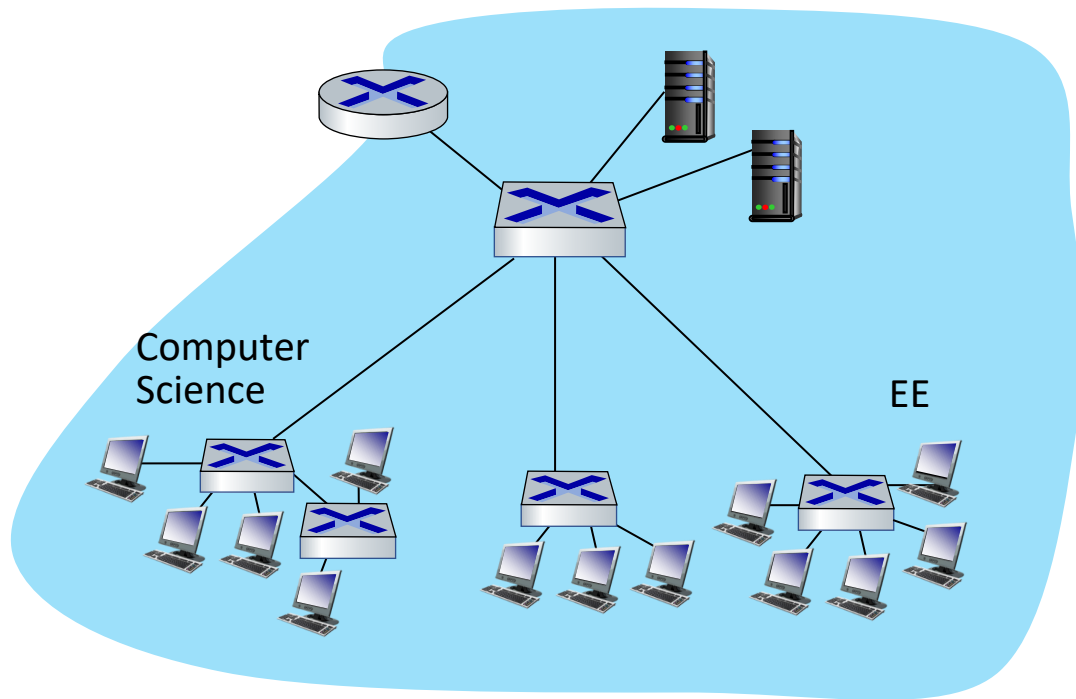
Link layer, LANs: roadmap

- introduction
- error detection, correction
- multiple access protocols
- **LANs**
  - addressing, ARP
  - Ethernet
  - switches
  - VLANs
- link virtualization: MPLS
- data center networking

- a day in the life of a web request

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?
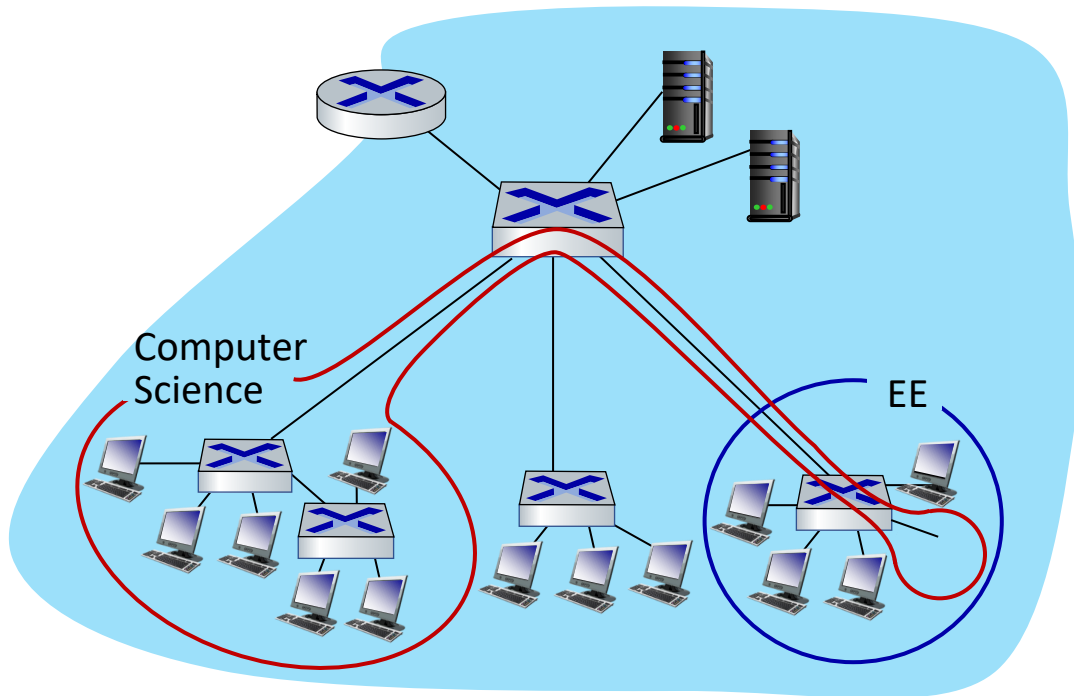
single broadcast domain:

- *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

Computer Science

EE

Port-based VLANs

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

EE (VLAN ports 1-8)   CS (VLAN ports 9-15)
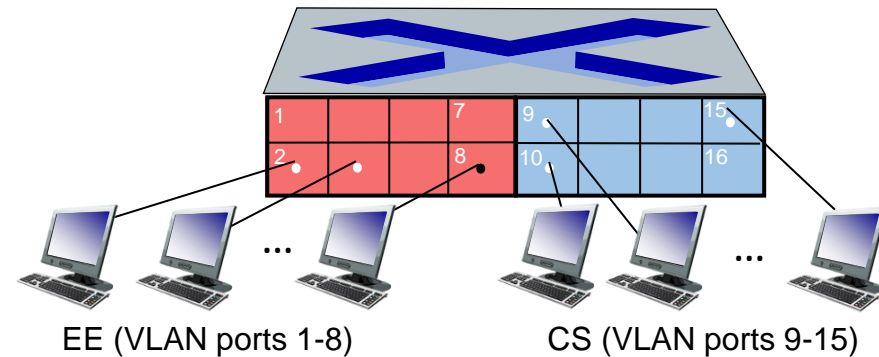
... operates as **multiple** virtual switches

EE (VLAN ports 1-8)   CS (VLAN ports 9-15)
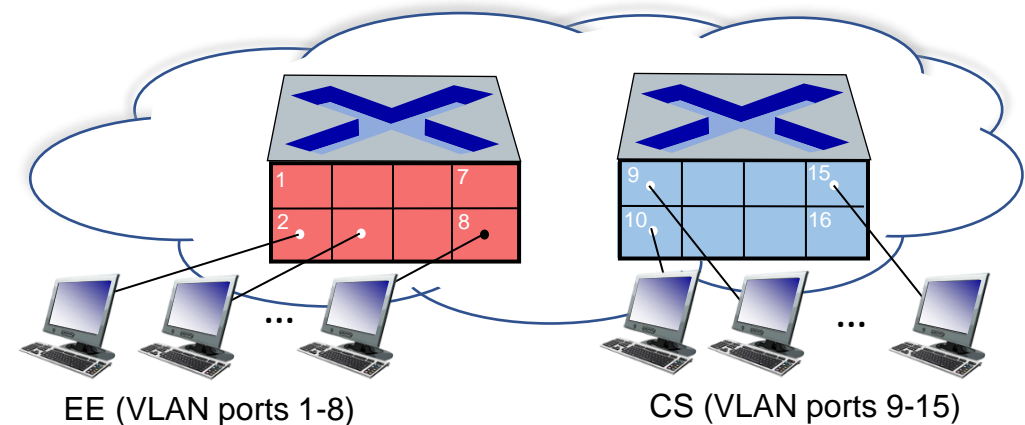
Port-based VLANs

- **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

EE (VLAN ports 1-8)   CS (VLAN ports 9-15)
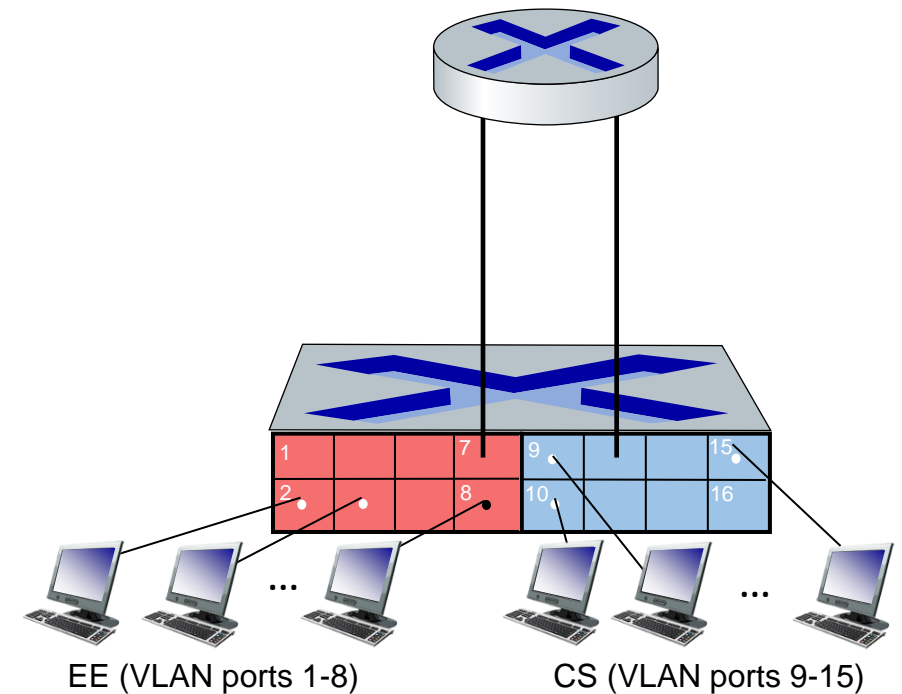
VLANS spanning multiple switches

EE (VLAN ports 1-8)

CS (VLAN ports 9-15)

Ports 2,3,5 belong to EE VLAN
Ports 4,6,7,8 belong to CS VLAN
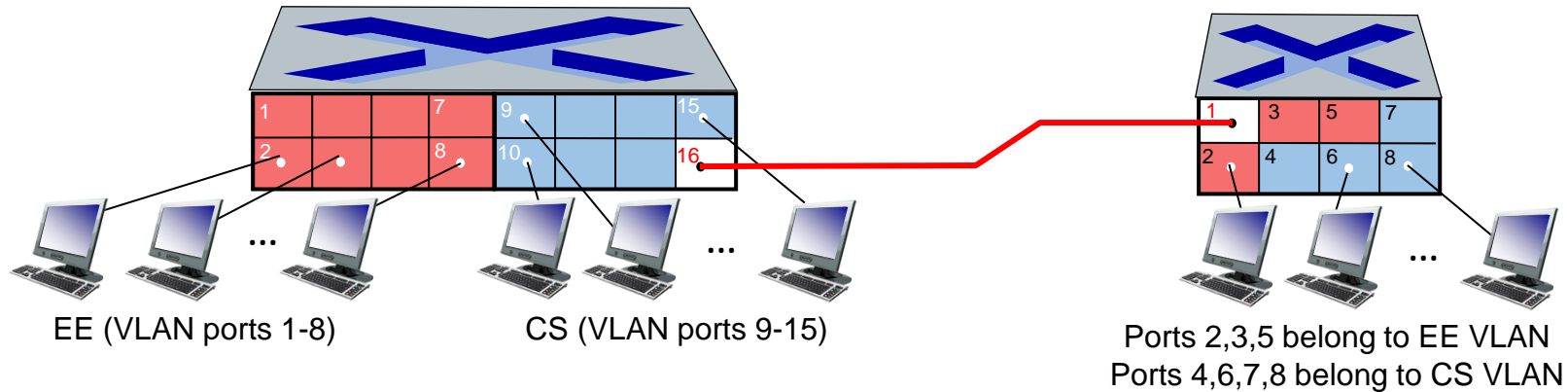
trunk port: carries frames between VLANS defined over multiple physical switches

- 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

type

| preamble | dest. address | source address | | data (payload) | CRC | 802.1 Ethernet frame |

type

| preamble | dest. address | source address | | | data (payload) | CRC | 802.1Q frame |

2-byte Tag Protocol Identifier
(value: 81-00)

Tag Control Information

(12 bit VLAN ID field, 3 bit priority field like IP TOS)

Recomputed CRC
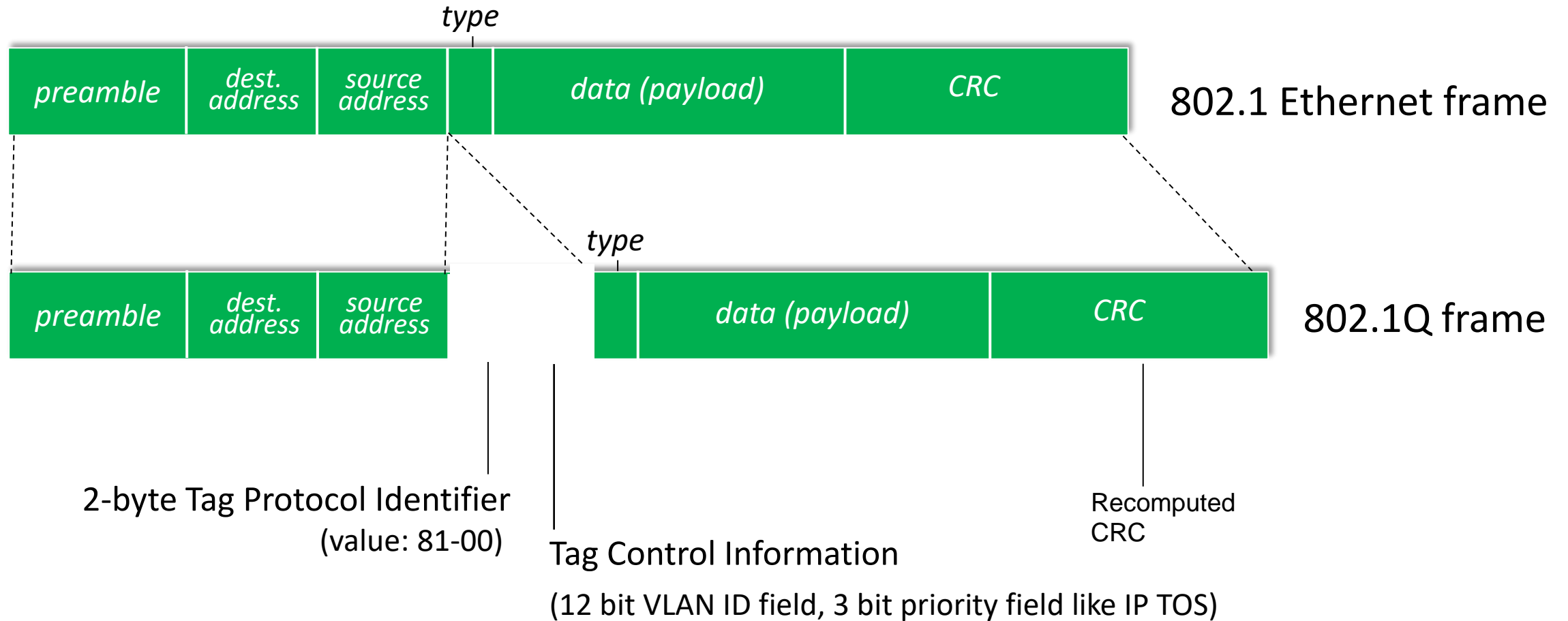
EVPN: Ethernet VPNs (aka VXLANs)
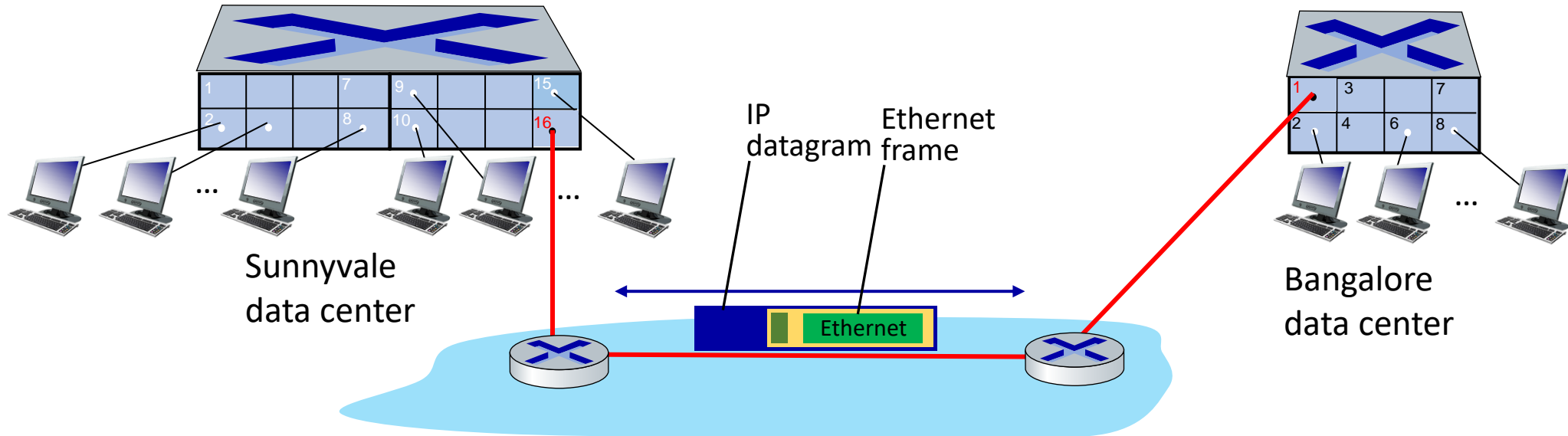
Layer-2 Ethernet switches *logically* connected to each other (e.g., using IP as an *underlay*)

- 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]
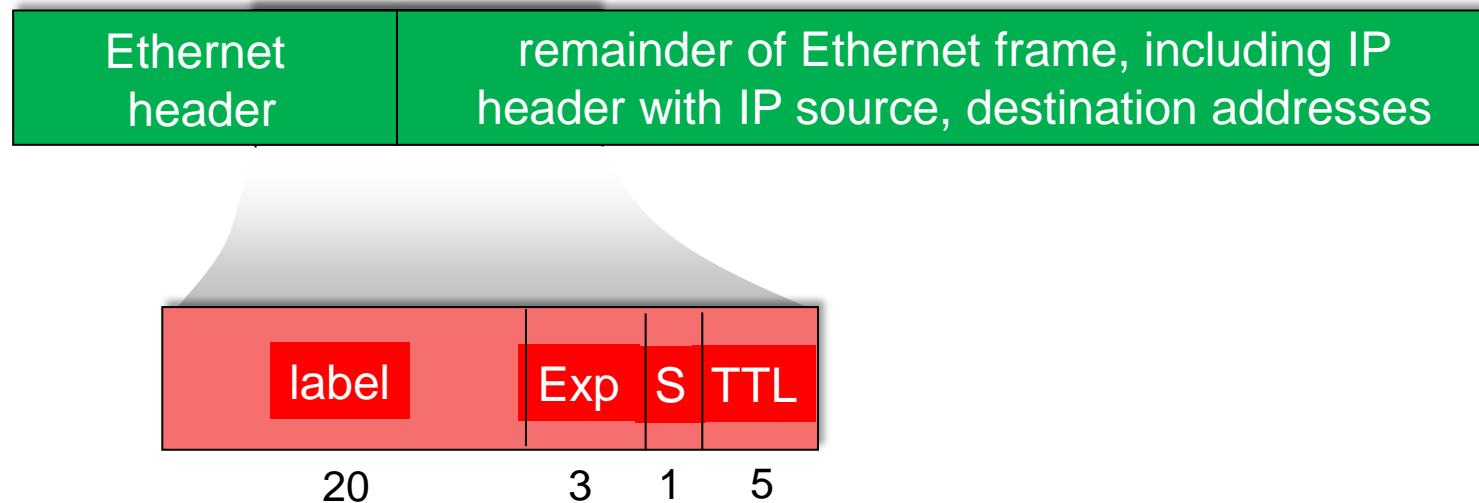
Link layer, LANs: roadmap

- introduction
- error detection, correction
- multiple access protocols
- LANs
  - addressing, ARP
  - Ethernet
  - switches
  - VLANs
- **link virtualization: MPLS**
- data center networking

- a day in the life of a web request

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!

Ethernet header	remainder of Ethernet frame, including IP header with IP source, destination addresses

label		Exp	S	TTL
20		3	1	5

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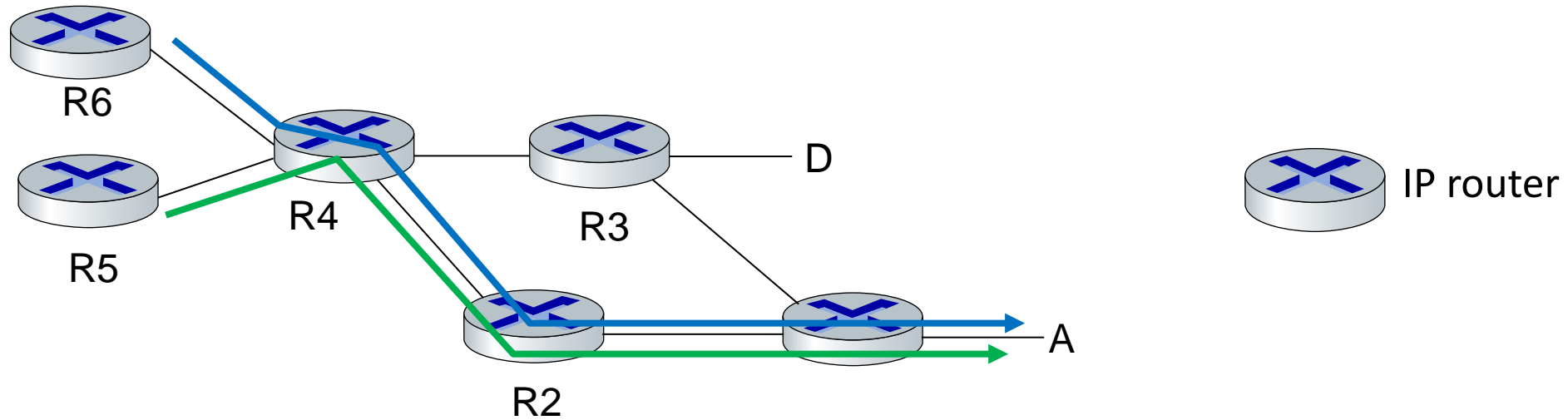
- **IP routing:** path to destination determined by destination address alone

MPLS versus IP paths

IP/MPLS entry router (R4) can use *different* MPLS routes to A based, e.g., on IP source address or other fields

R6

R5

R4

R3

D

R2

R1

A

IP router
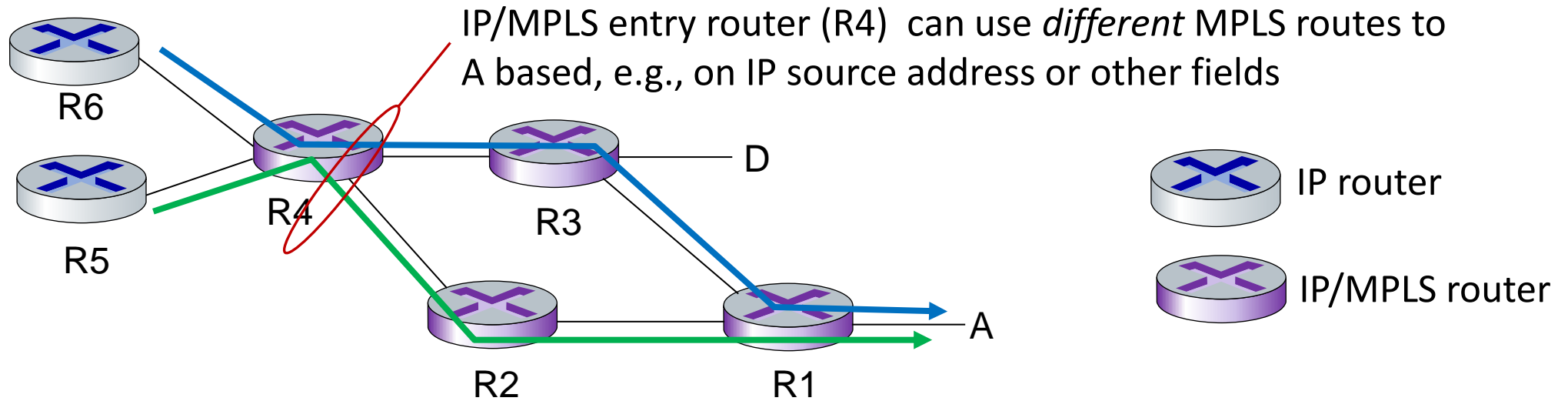
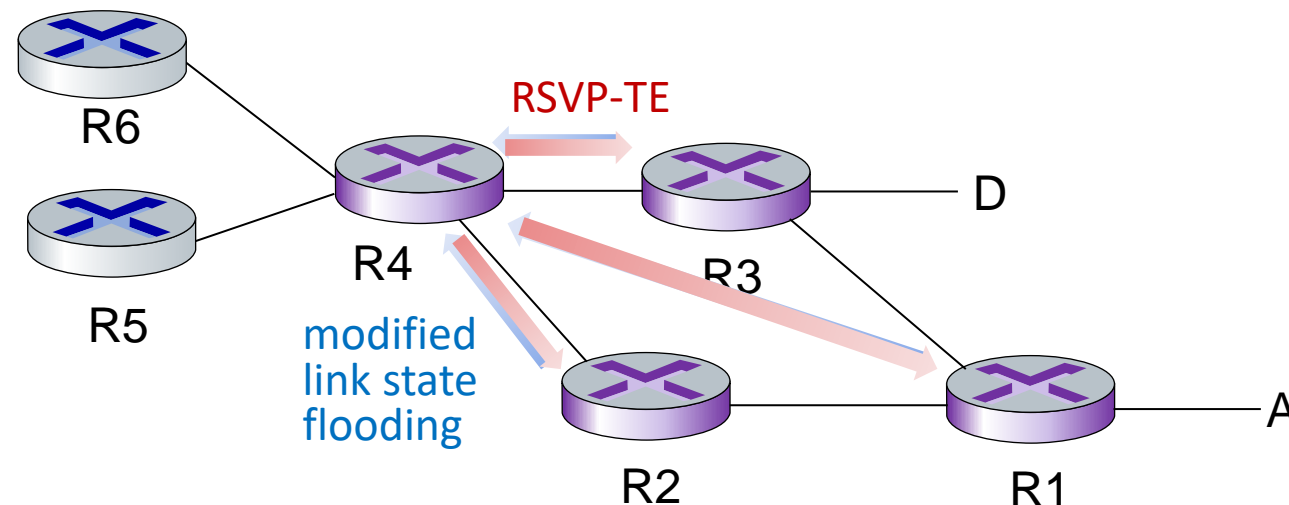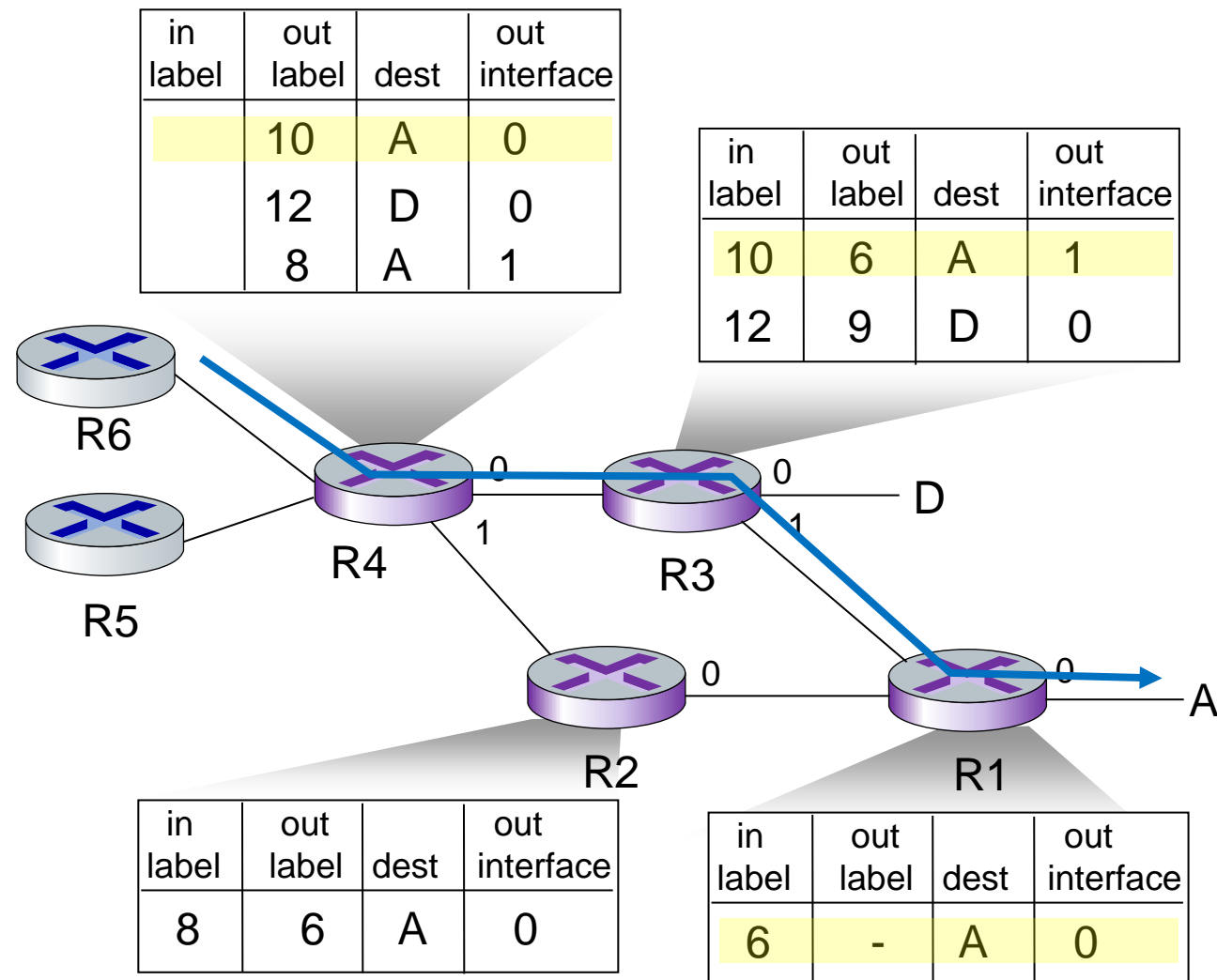IP/MPLS router

- **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

in label	out label	dest	out interface
10	A	0	
12	D	0	
8	A	1	

in label	out label	dest	out interface
10	6	A	1
12	9	D	0

in label	out label	dest	out interface
8	6	A	0

in label	out label	dest	out interface
6	-	A	0

Link layer, LANs: roadmap

- introduction
- error detection, correction
- multiple access protocols
- LANs
  - addressing, ARP
  - Ethernet
  - switches
  - VLANs
- link virtualization: MPLS
- **data center networking**

- a day in the life of a web request

Datacenter networks

10's to 100's of thousands of hosts, often closely coupled, in close proximity:

- e-business (e.g. Amazon)
- content-servers (e.g., YouTube, Akamai, Apple, Microsoft)
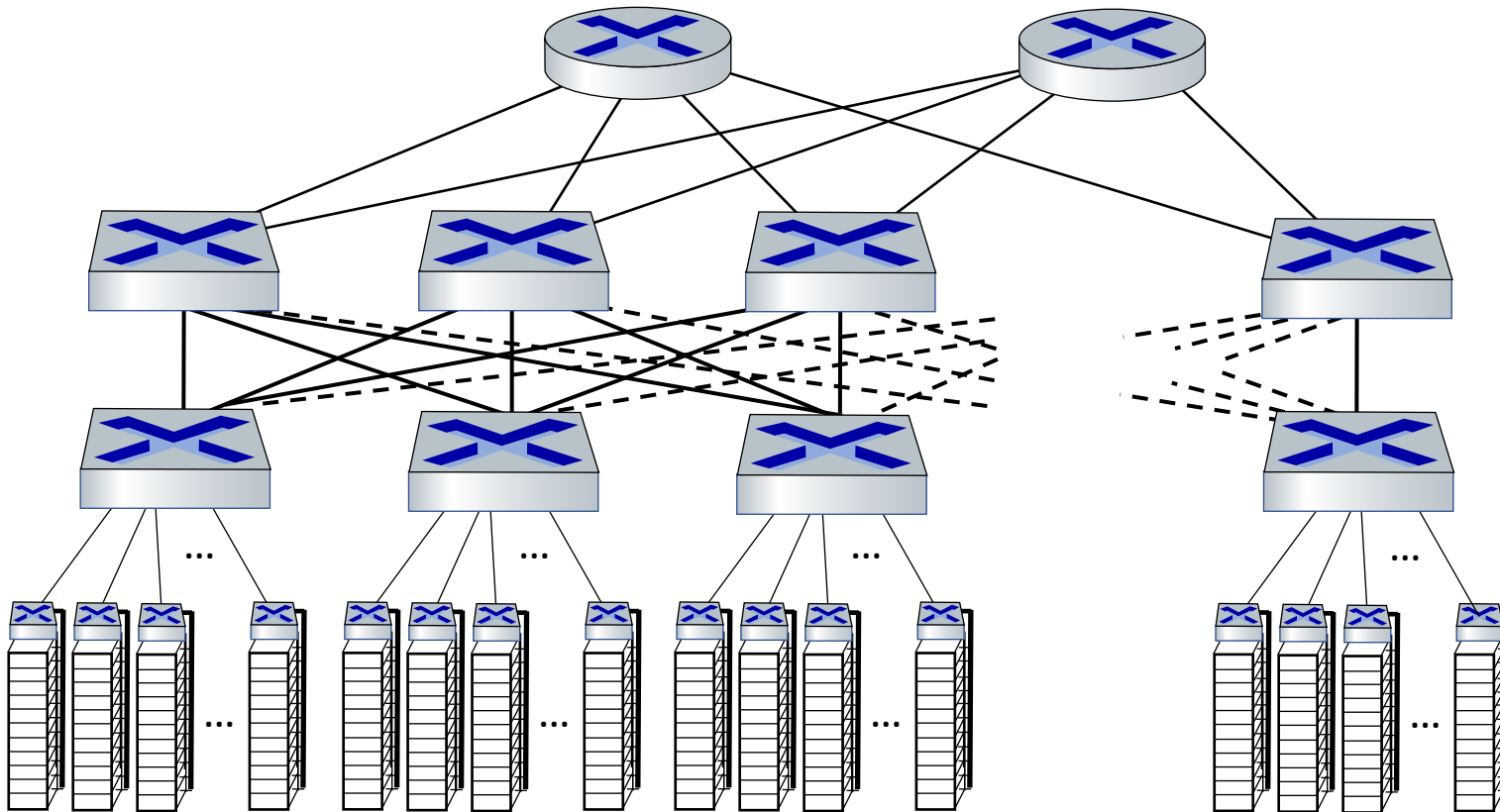- search engines, data mining (e.g., Google)

challenges:

- multiple applications, each serving massive numbers of clients

- reliability

- managing/balancing load, avoiding processing, networking, data bottlenecks

Inside a 40-ft Microsoft container, Chicago data center

Datacenter networks: network elements

Border routers
- connections outside datacenter

Tier-1 switches
- connecting to ~16 T-2s below

Tier-2 switches
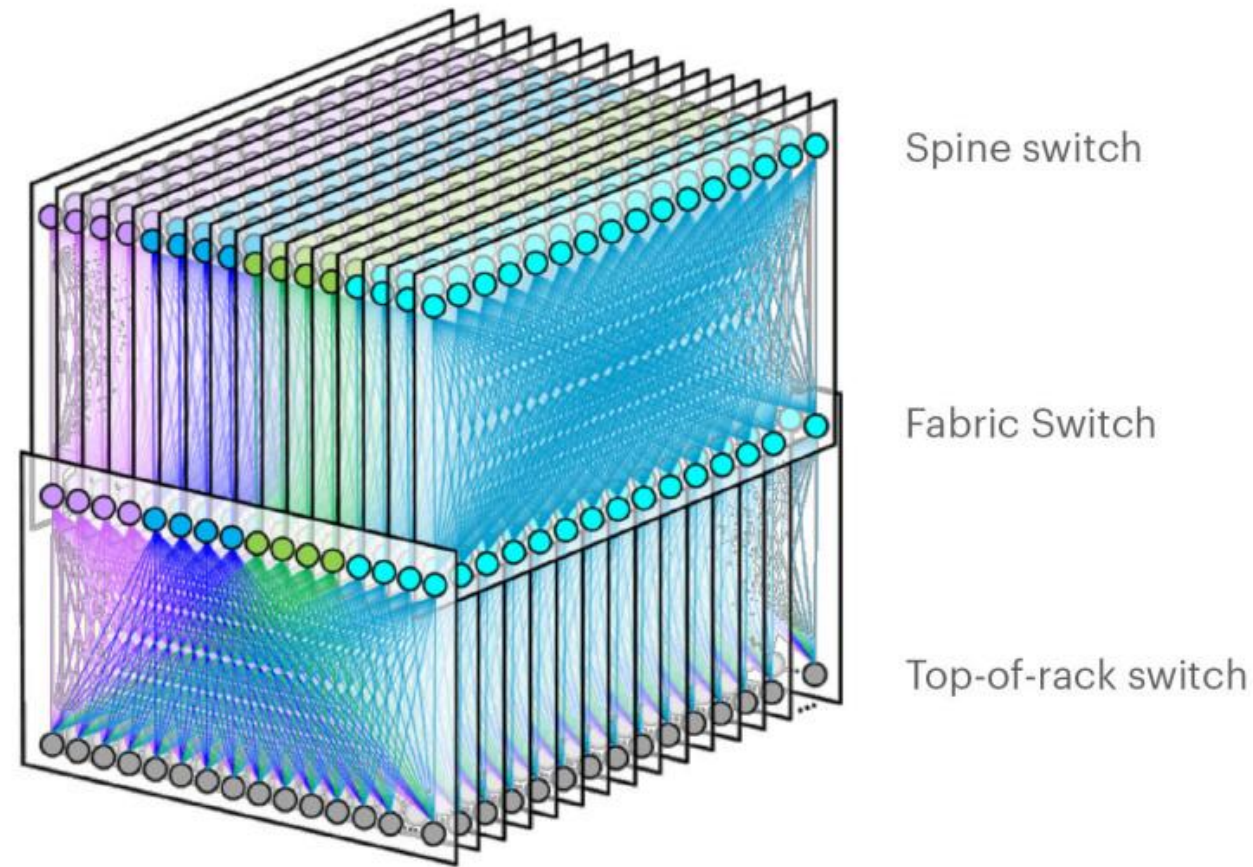- connecting to ~16 TORs below

Top of Rack (TOR) switch
- one per rack
- 100G-400G Ethernet to blades

Server racks
- 20- 40 server blades: hosts
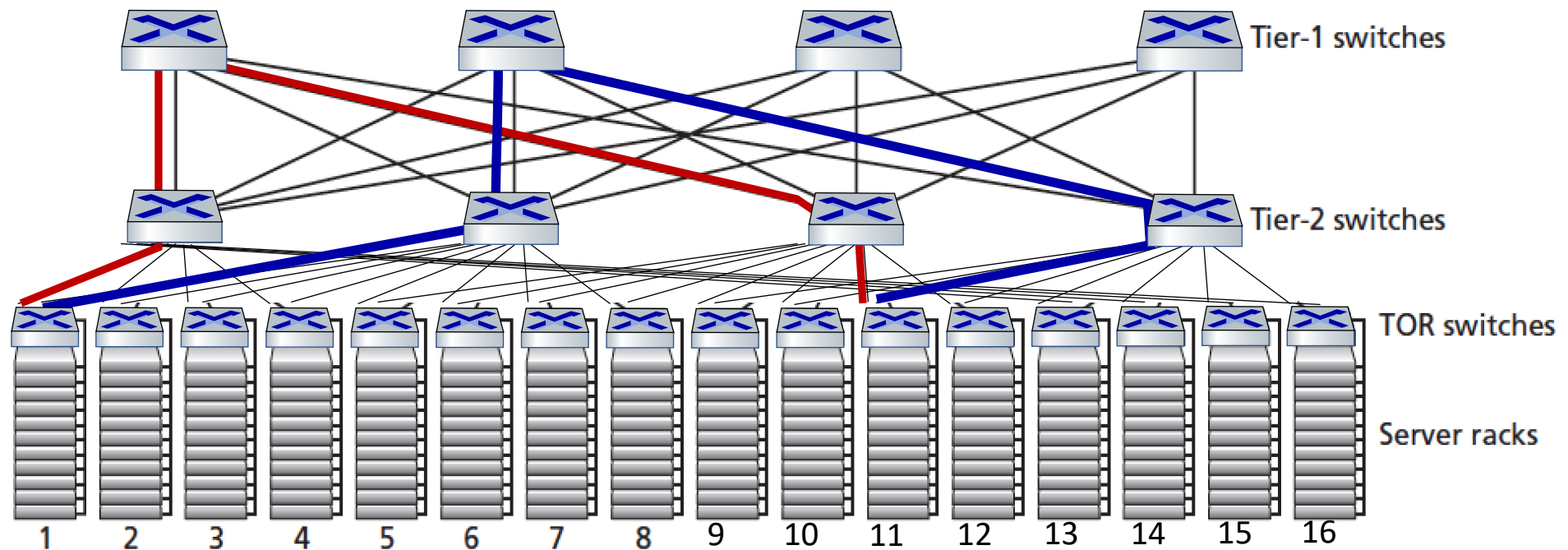
Datacenter networks: network elements

Facebook F16 data center network topology:

Spine switch

Fabric Switch

Top-of-rack switch

https://engineering.fb.com/data-center-engineering/f16-minipack/   (posted 3/2019)
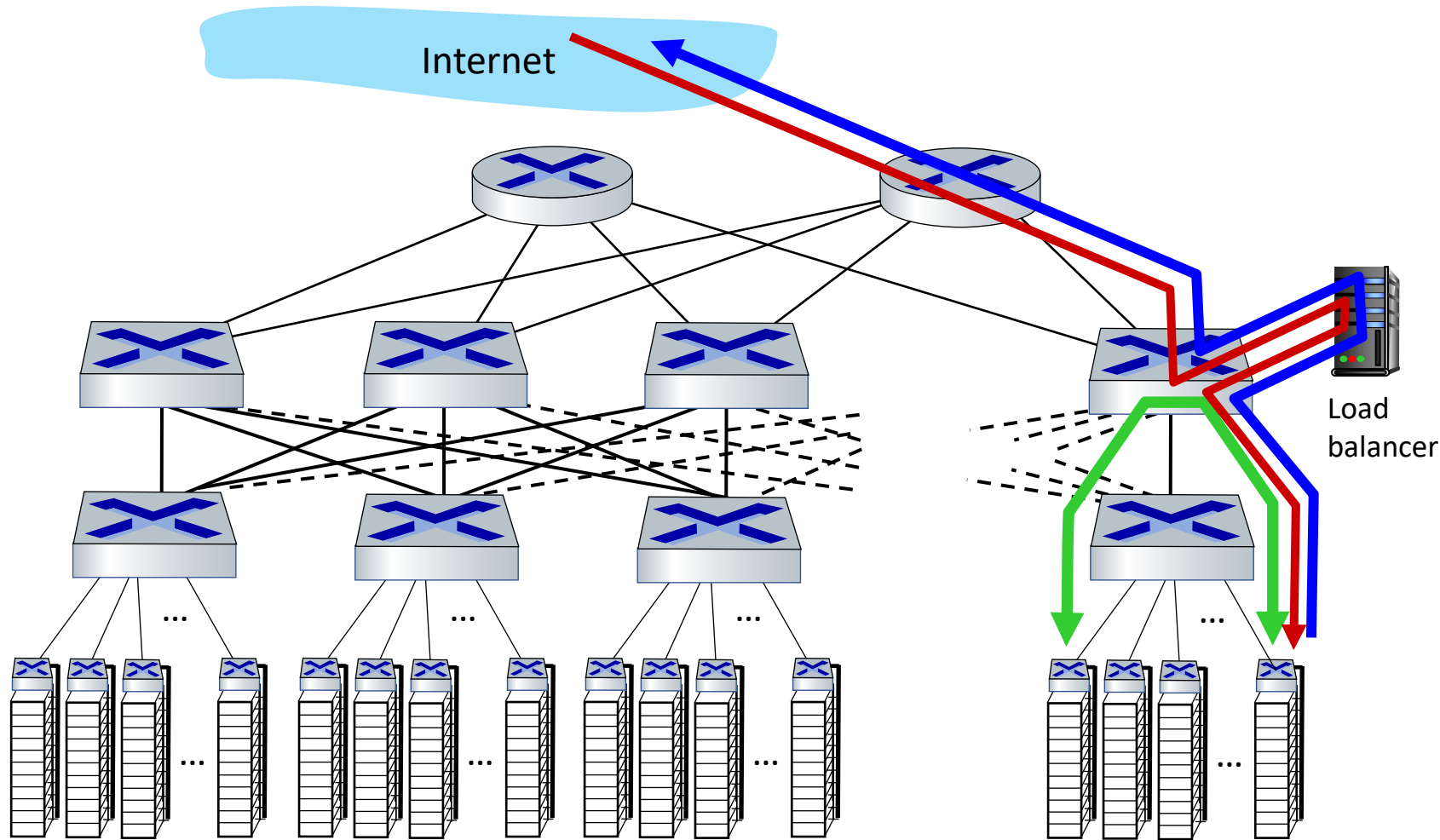
Datacenter networks: multipath

- rich interconnection among switches, racks:
  - increased throughput between racks (multiple routing paths possible)
  - increased reliability via redundancy

two disjoint paths highlighted between racks 1 and 11

Datacenter networks: application-layer routing

Internet

load balancer: application-layer routing

- receives external client requests
- directs workload within data center
- returns results to external client (hiding data center internals from client)

Load balancer

Link layer, LANs: roadmap

- introduction
- error detection, correction
- multiple access protocols
- LANs
  - addressing, ARP
  - Ethernet
  - switches
  - VLANs
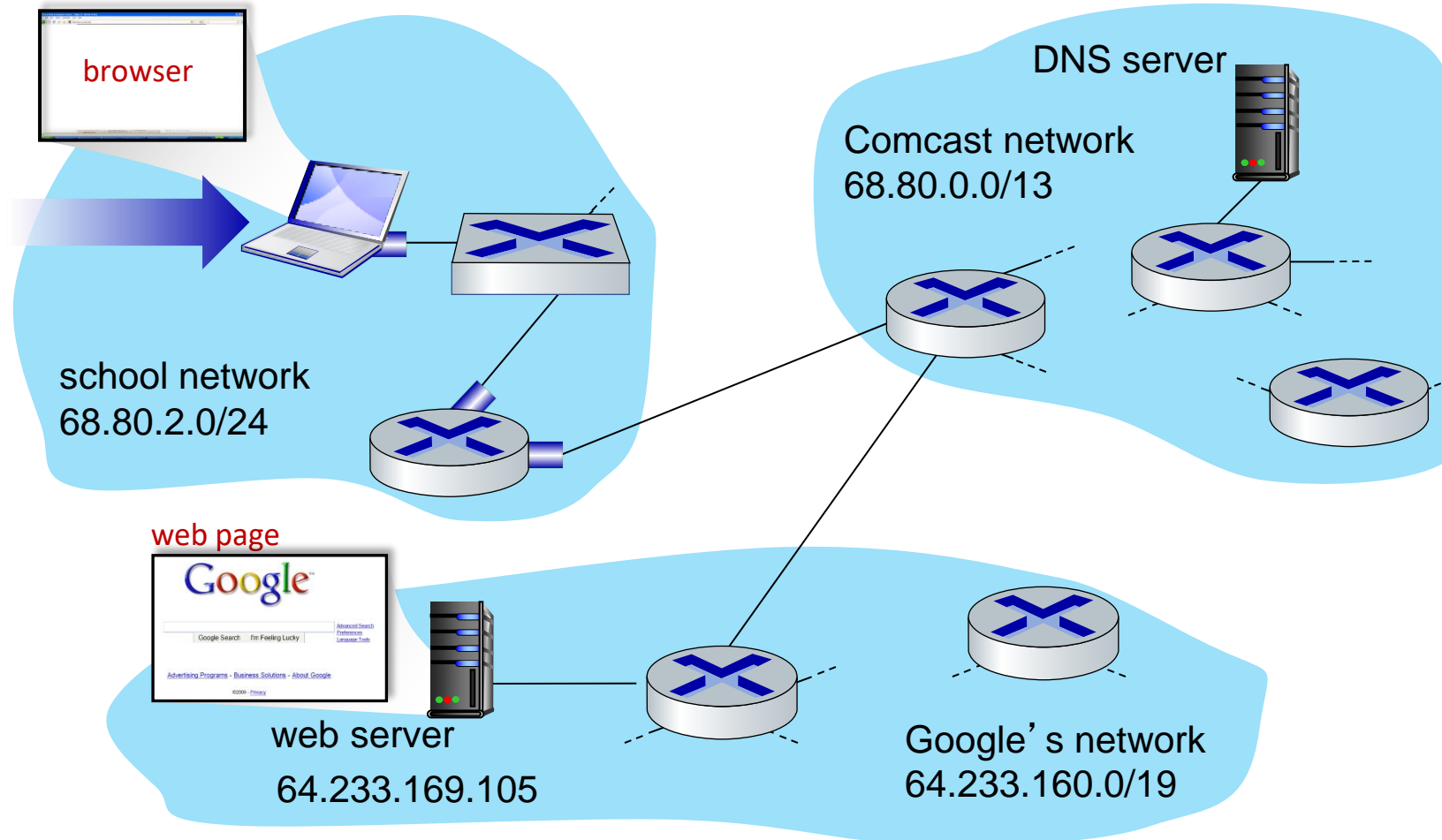- link virtualization: MPLS
- data center networking

- **a day in the life of a web request**

Synthesis: a day in the life of a web request

▪ our journey down the protocol stack is now complete!
- application, transport, network, link

▪ putting-it-all-together: synthesis!
- *goal:* identify, review, understand protocols (at all layers) involved in seemingly simple scenario: requesting www page
- *scenario:* student attaches laptop to campus network, requests/receives www.google.com

A day in the life: scenario

browser

school network
68.80.2.0/24

DNS server

Comcast network
68.80.0.0/13

web page

Google

web server
64.233.169.105

Google's network
64.233.160.0/19

scenario:

- arriving mobile client attaches to network ...

- requests web page: www.google.com

Sounds simple!

A day in the life: connecting to the Internet

arriving mobile: DHCP client

router has DHCP server
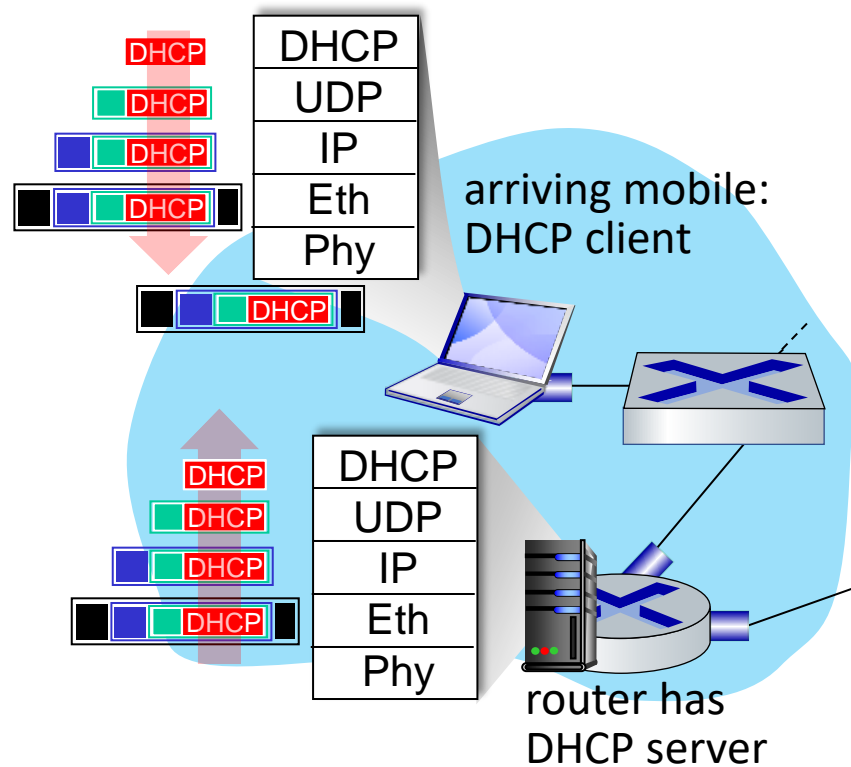
- connecting laptop needs to get its own IP address, addr of first-hop router, addr of DNS server: use DHCP

- DHCP request encapsulated in UDP, encapsulated in IP, encapsulated in 802.3 Ethernet

- Ethernet frame broadcast (dest: FFFFFFFFFFFF) on LAN, received at router running DHCP server

- Ethernet de-muxed to IP de-muxed, UDP de-muxed to DHCP

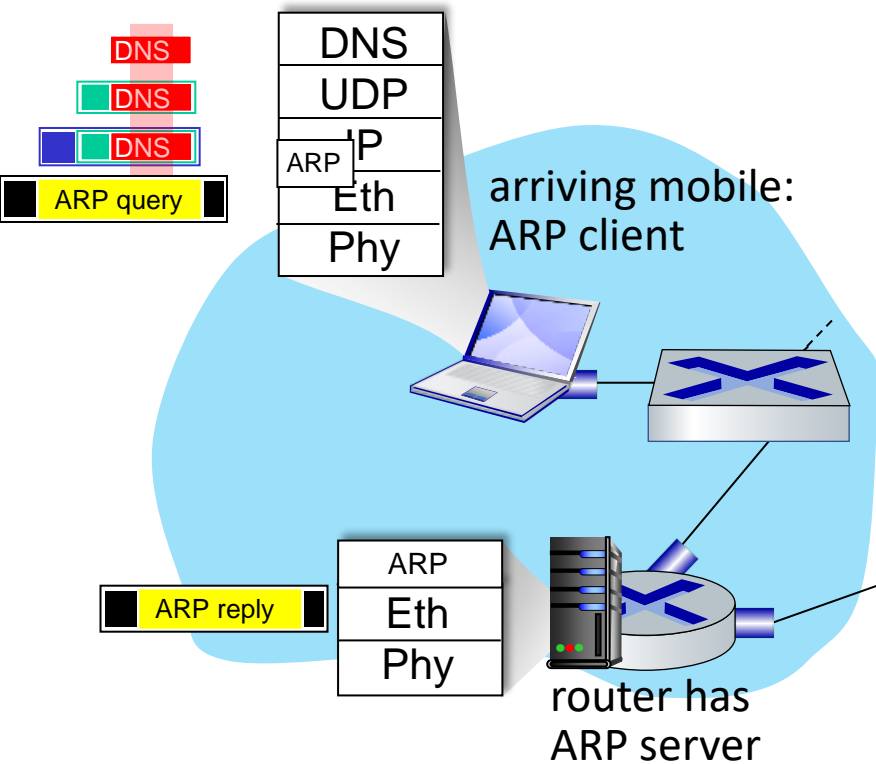# A day in the life: connecting to the Internet

arriving mobile: DHCP client

router has DHCP server
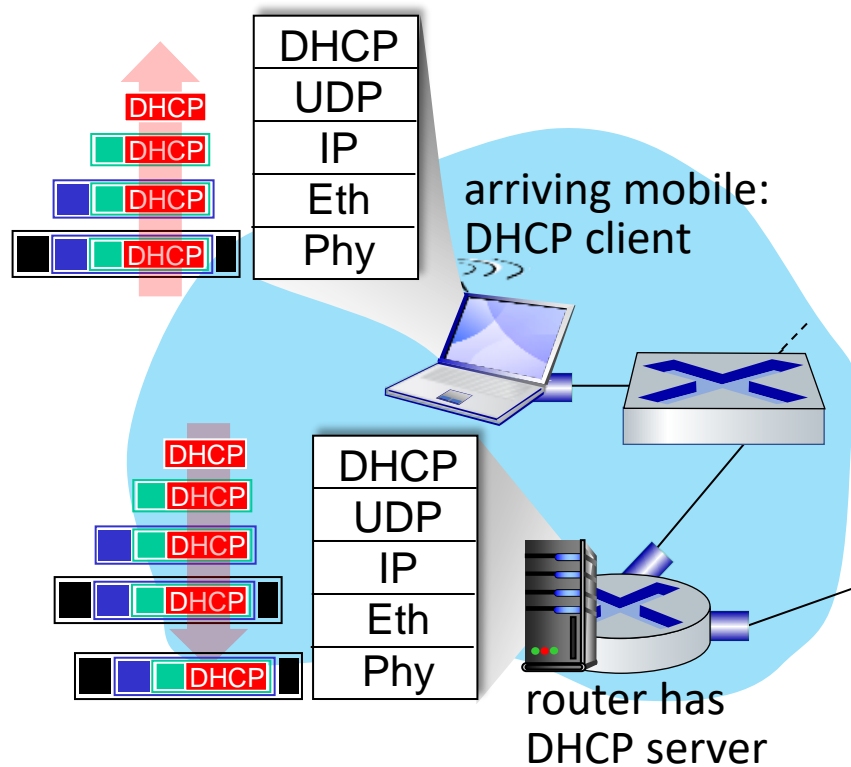
- DHCP server formulates DHCP ACK containing client's IP address, IP address of first-hop router for client, name & IP address of DNS server

- encapsulation at DHCP server, frame forwarded (switch learning) through LAN, demultiplexing at client
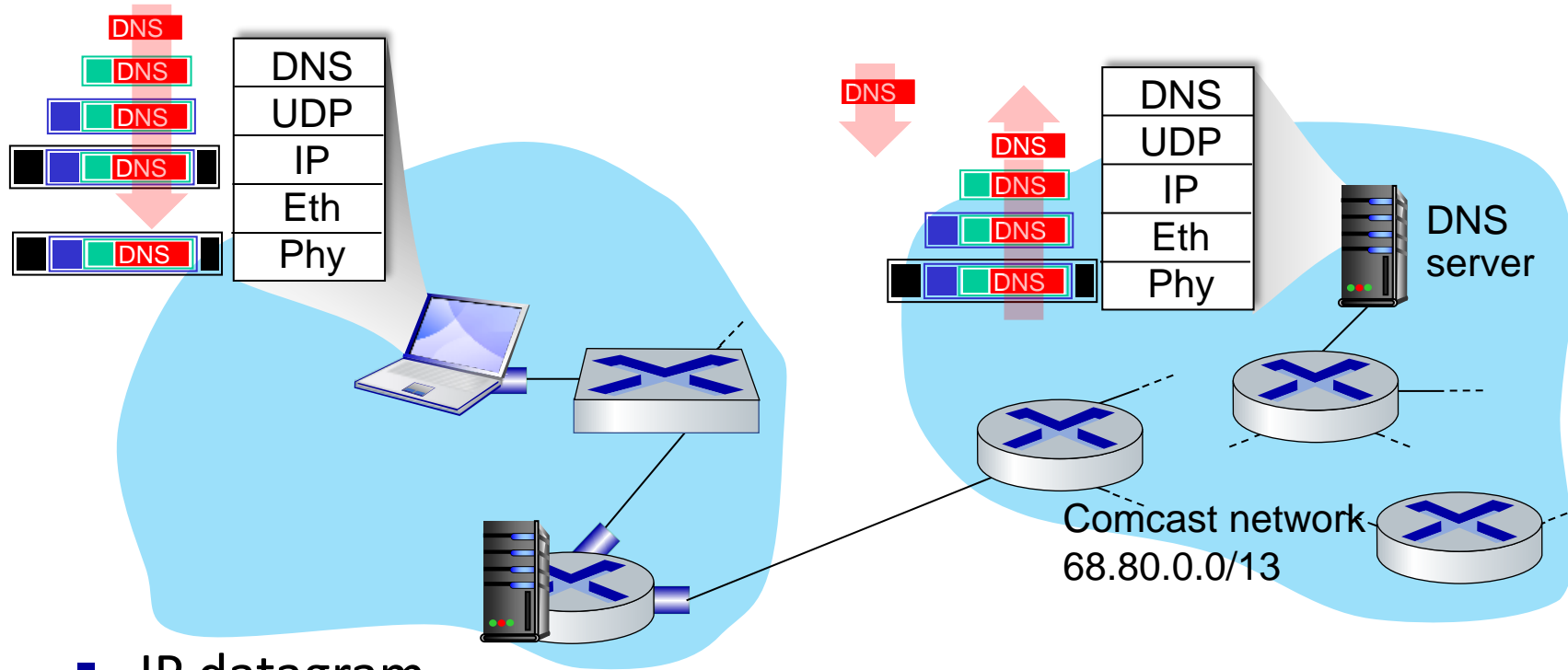
- DHCP client receives DHCP ACK reply

Client now has IP address, knows name & addr of DNS server, IP address of its first-hop router

A day in the life... ARP (before DNS, before HTTP)

DNS
DNS
DNS
ARP query

DNS
UDP
IP
ARP
Eth
Phy

arriving mobile:
ARP client

ARP reply

ARP
Eth
Phy

router has
ARP server

- before sending HTTP request, need IP address of www.google.com: DNS

- DNS query created, encapsulated in UDP, encapsulated in IP, encapsulated in Eth. To send frame to router, need MAC address of router interface: ARP

- ARP query broadcast, received by router, which replies with ARP reply giving MAC address of router interface

- client now knows MAC address of first hop router, so can now send frame containing DNS query

A day in the life... using DNS

DNS

| DNS |
| UDP |
| IP |
| Eth |
| Phy |

- de-muxed to DNS
- DNS replies to client with IP address of www.google.com

DNS

| DNS |
| UDP |
| IP |
| Eth |
| Phy |

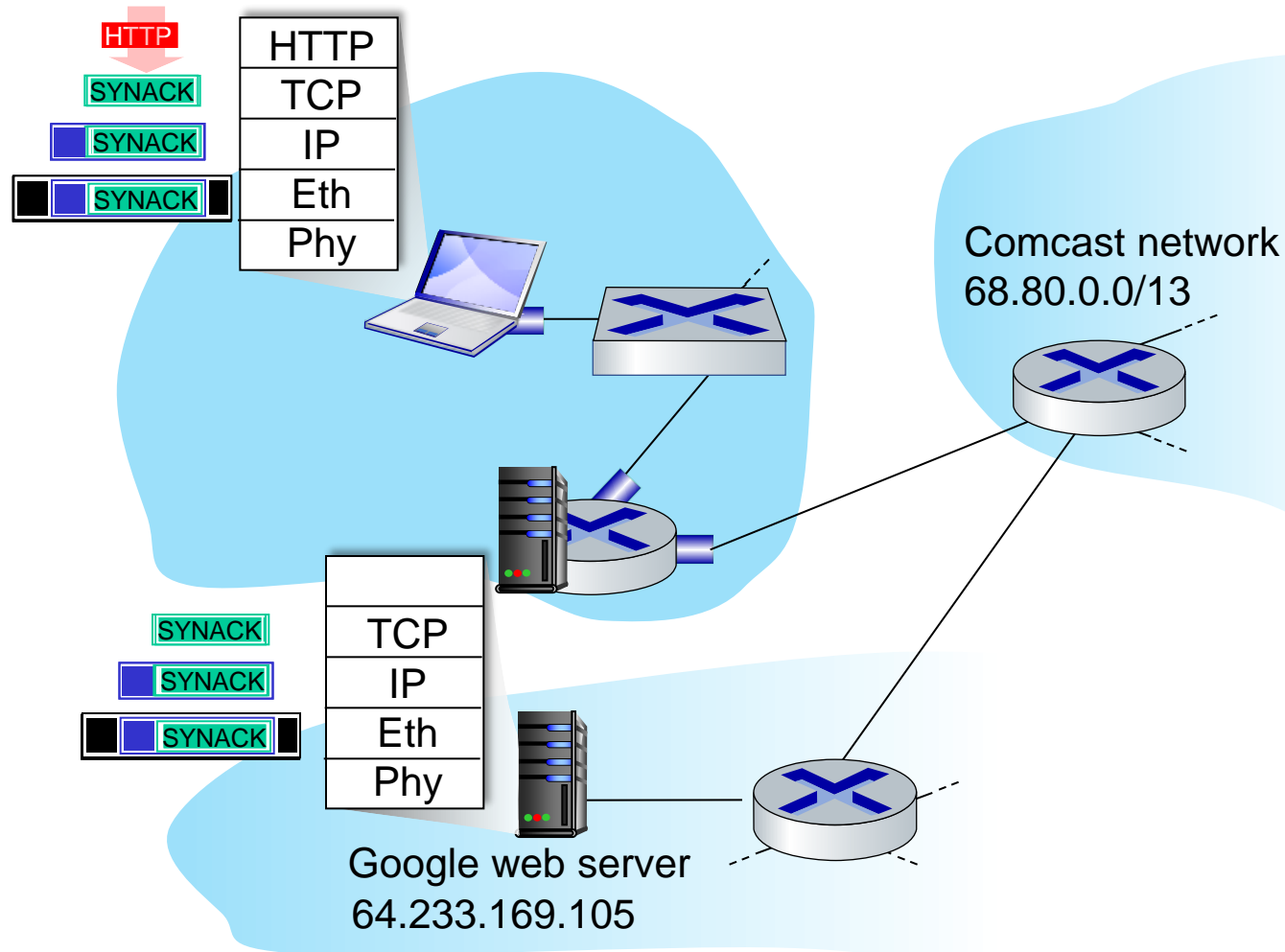DNS server

Comcast network
68.80.0.0/13

- IP datagram containing DNS query forwarded via LAN switch from client to 1st hop router

- IP datagram forwarded from campus network into Comcast network, routed (tables created by RIP, OSPF, IS-IS and/or BGP routing protocols) to DNS server

# A day in the life…TCP connection carrying HTTP

- to send HTTP request, client first opens TCP socket to web server

- TCP SYN segment (step 1 in TCP 3-way handshake) inter-domain routed to web server

- web server responds with TCP SYNACK (step 2 in TCP 3-way handshake)

- TCP connection established!

A day in the life… HTTP request/reply

web page *finally (!!!)* displayed

Comcast network 68.80.0.0/13

Google web server 64.233.169.105
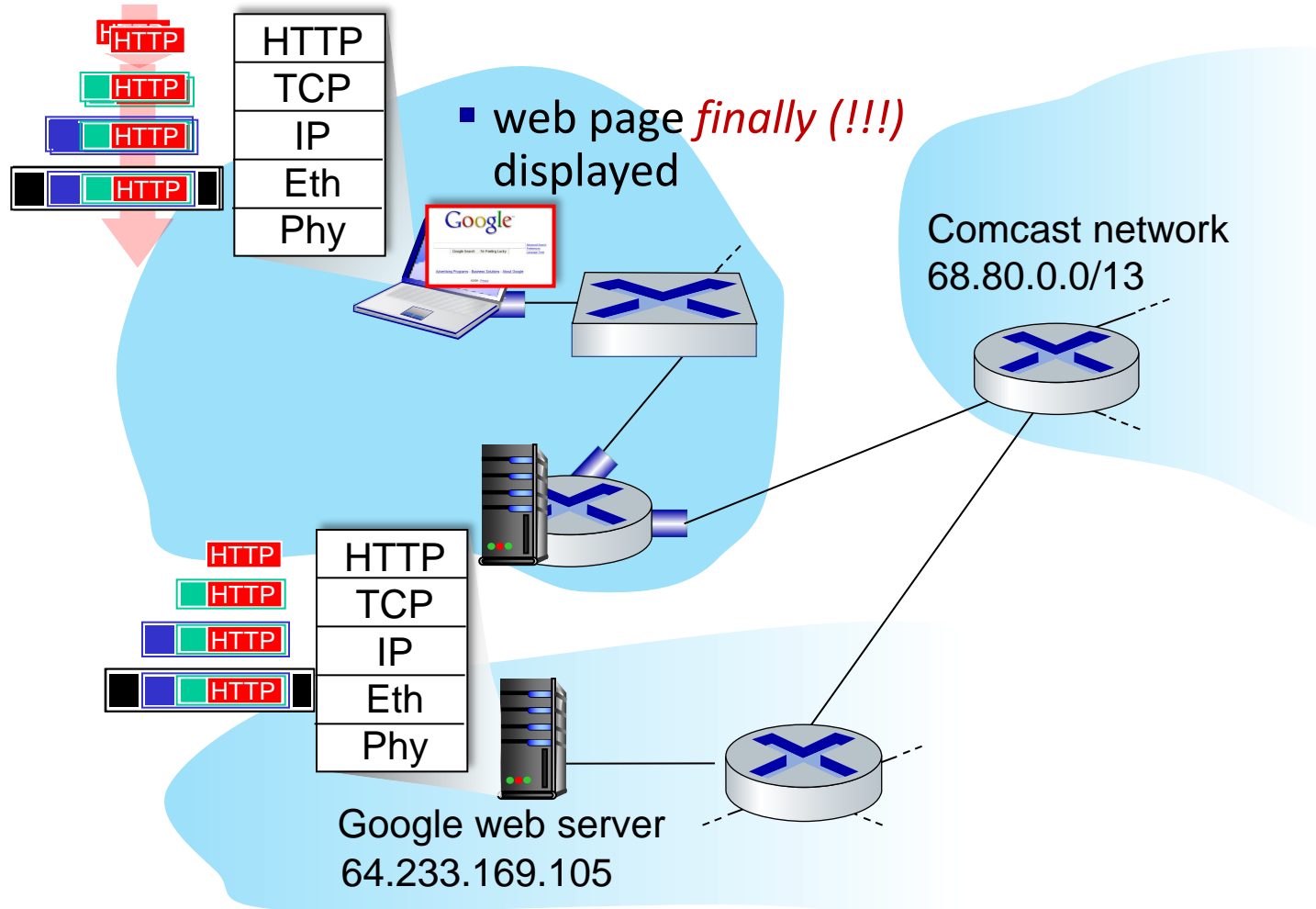
- **HTTP request** sent into TCP socket

- IP datagram containing HTTP request routed to www.google.com

- web server responds with **HTTP reply** (containing web page)

- IP datagram containing HTTP reply routed back to client

Chapter 6: Summary

- principles behind data link layer services:
  - error detection, correction
  - sharing a broadcast channel: multiple access
  - link layer addressing
- instantiation, implementation of various link layer technologies
  - Ethernet
  - switched LANS, VLANs
  - virtualized networks as a link layer: MPLS
- synthesis: a day in the life of a web request