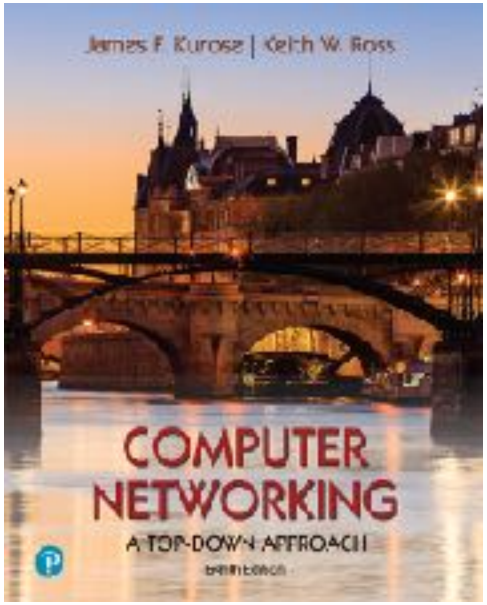
Chapter 4
Network Layer
Data Plane



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

Day 18: Recapping the Edge and Diving into the Core

CSEE 4119
Computer Networks
Ethan Katz-Bassett

COLUMBIA UNIVERSITY
IN THE CITY OF NEW YORK

Slides adapted from (and often identical to) slides from Kurose and Ross.

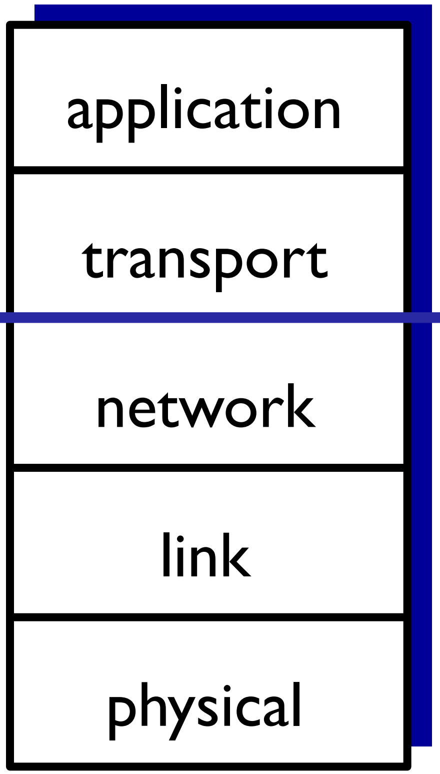
Recall: Internet protocol stack (5 layer)

- *application:* supporting network applications
  - DNS, SMTP, HTTP
- *transport:* process-process data transfer
  - TCP, UDP

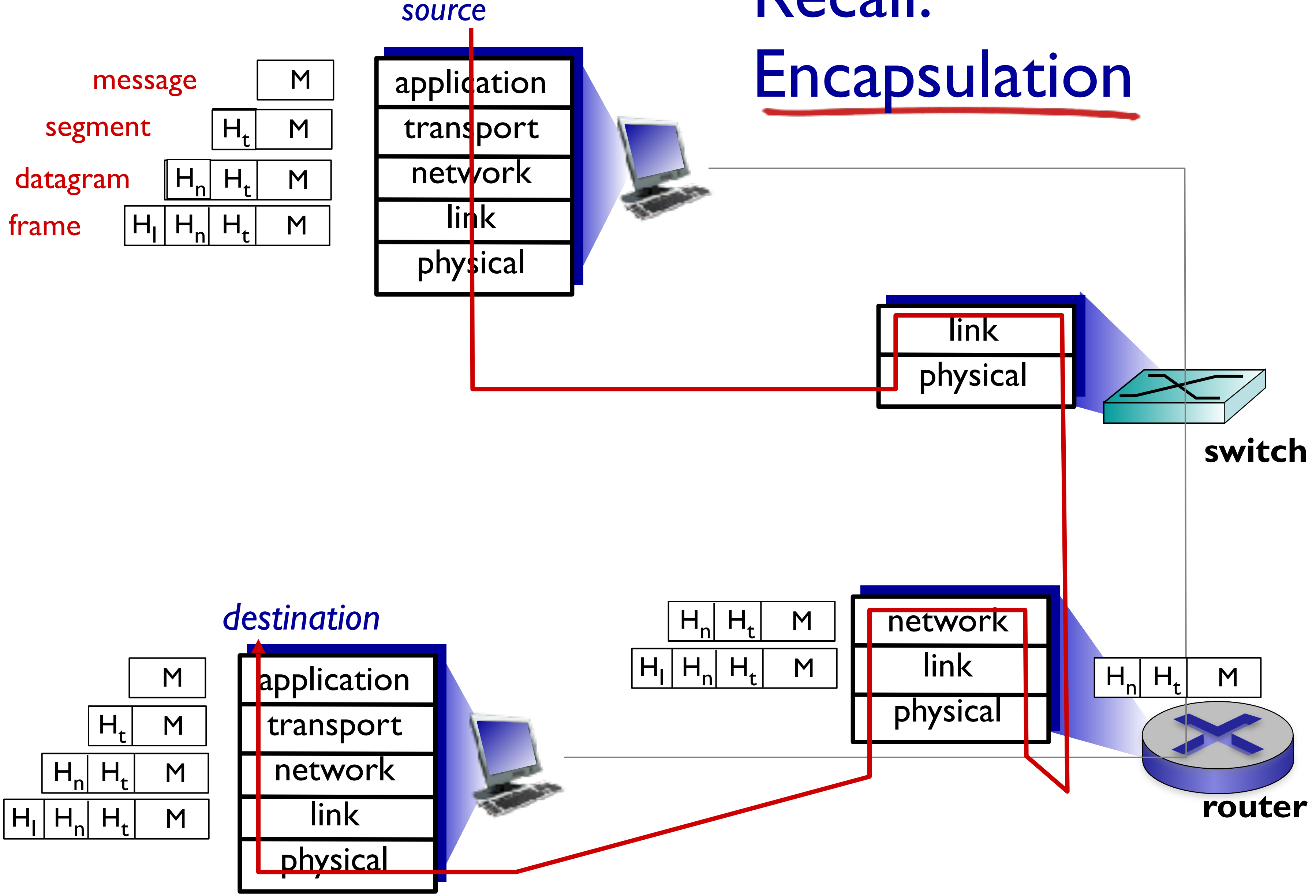
⬆ what we've done so far

⬇ where we are going next

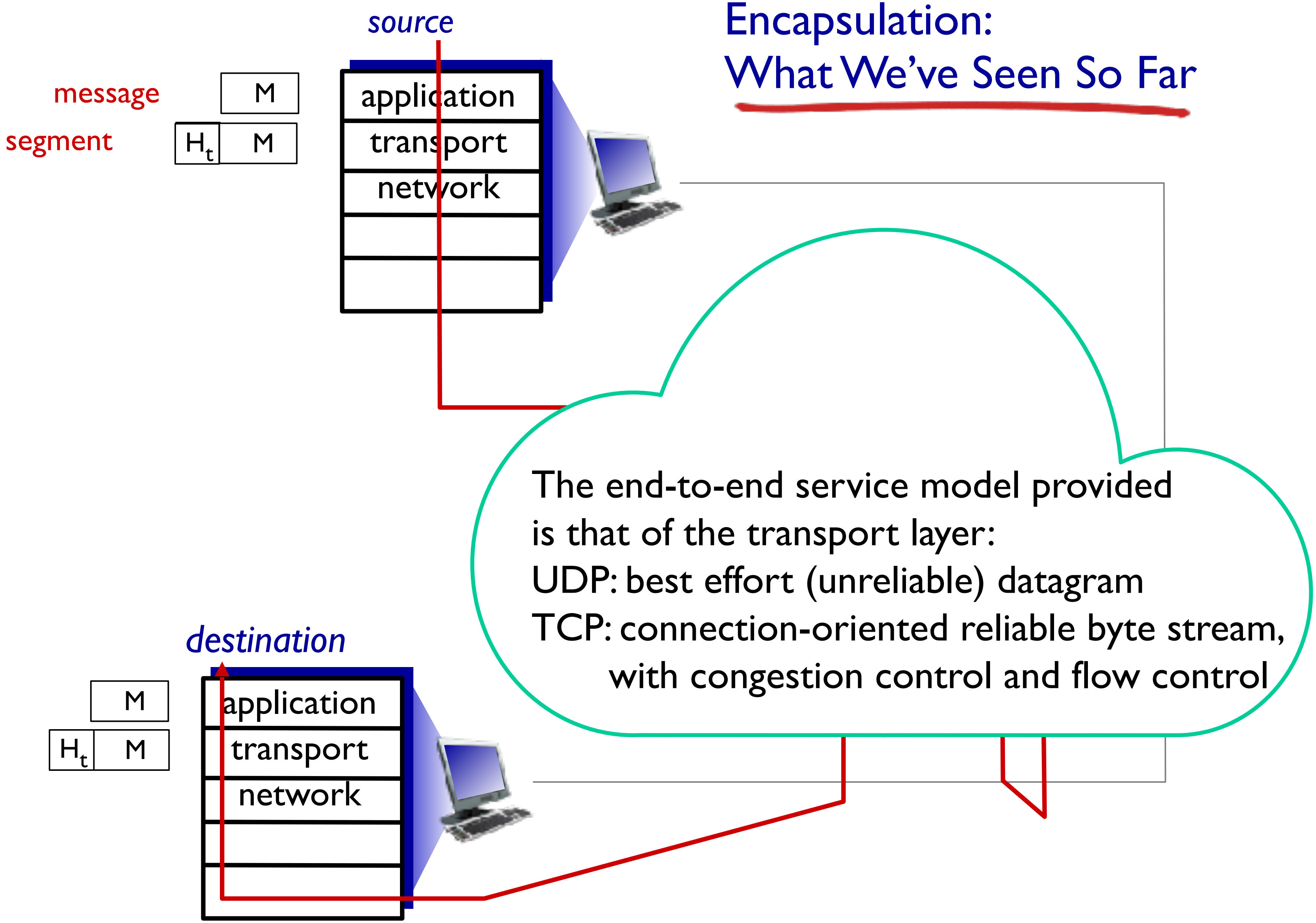
- *network:* routing of datagrams from source to destination
  - IP, routing protocols
- *link:* data transfer between neighboring network elements
  - Ethernet, 802.111 (WiFi), PPP
- *physical:* bits "on the wire"



Recall: Encapsulation



Encapsulation: What We've Seen So Far



The end-to-end service model provided is that of the transport layer:

UDP: best effort (unreliable) datagram

TCP: connection-oriented reliable byte stream, with congestion control and flow control

(half) a day in the life of a web request

6.1 introduction, services

6.2 error detection, correction

6.3 multiple access protocols

64 LANs

- addressing, ARP
- Ethernet
- switches
- VLANS

6.5 link virtualization: MPLS

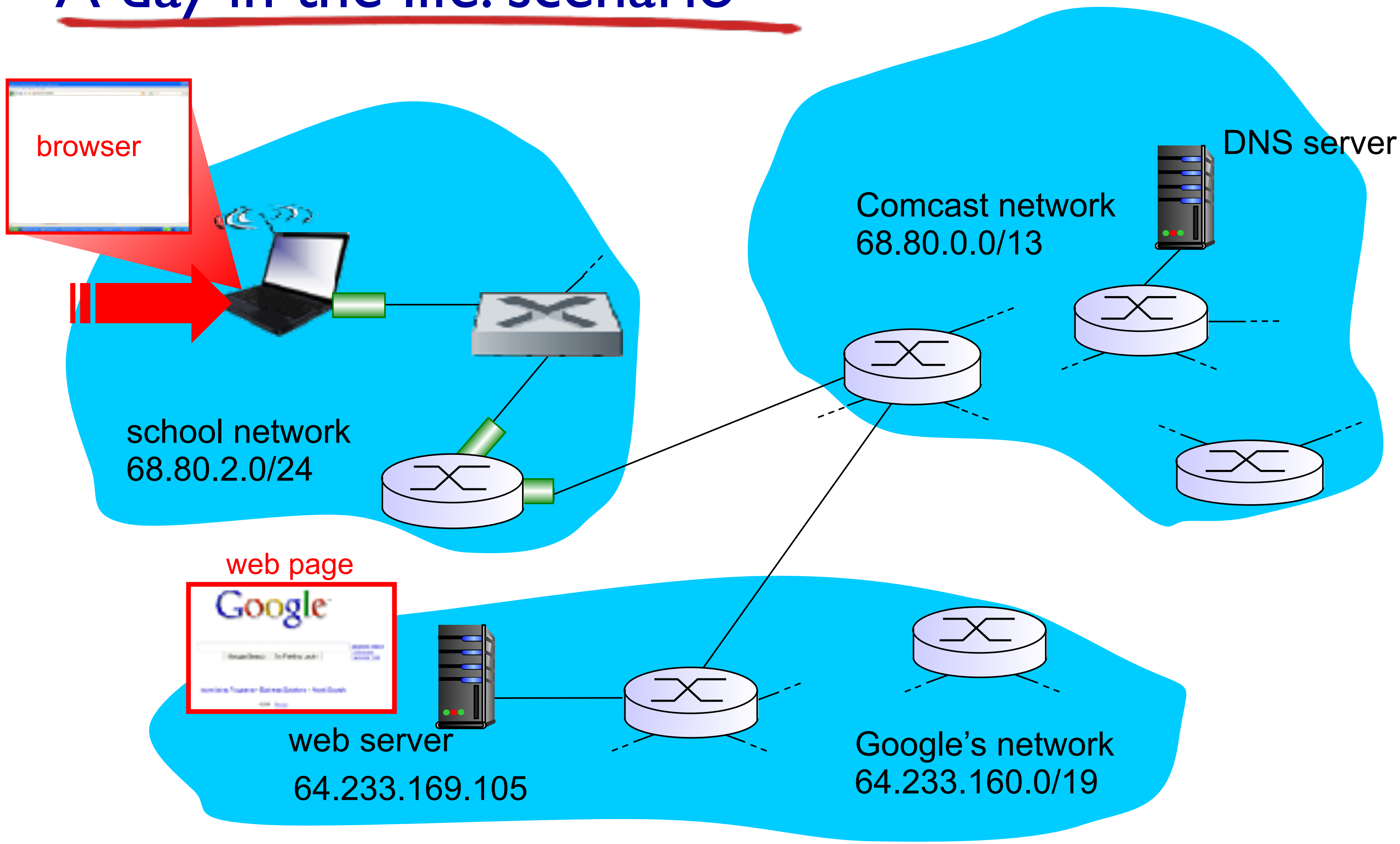
6.6 data center networking

6.7 a day in the life of a web request

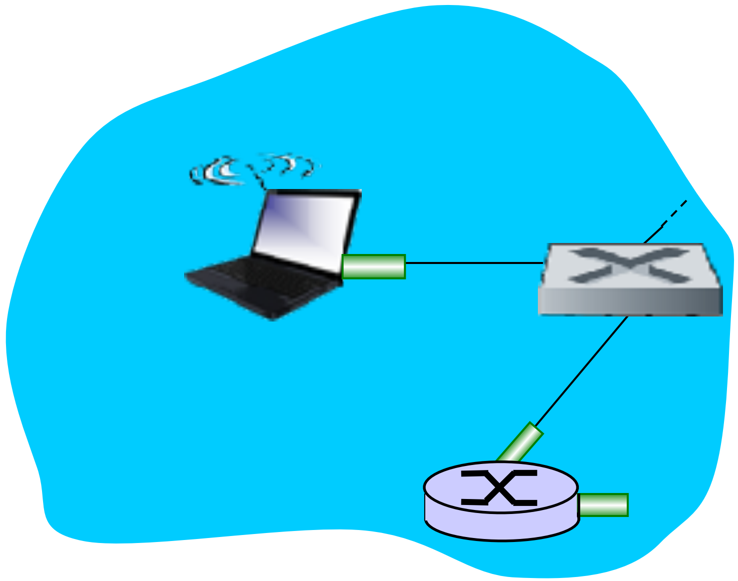
Synthesis: a day in the life of a web request

- journey down protocol stack 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
  - But also think about:
    - how does this relate to the steps in Project 1?

A day in the life: scenario

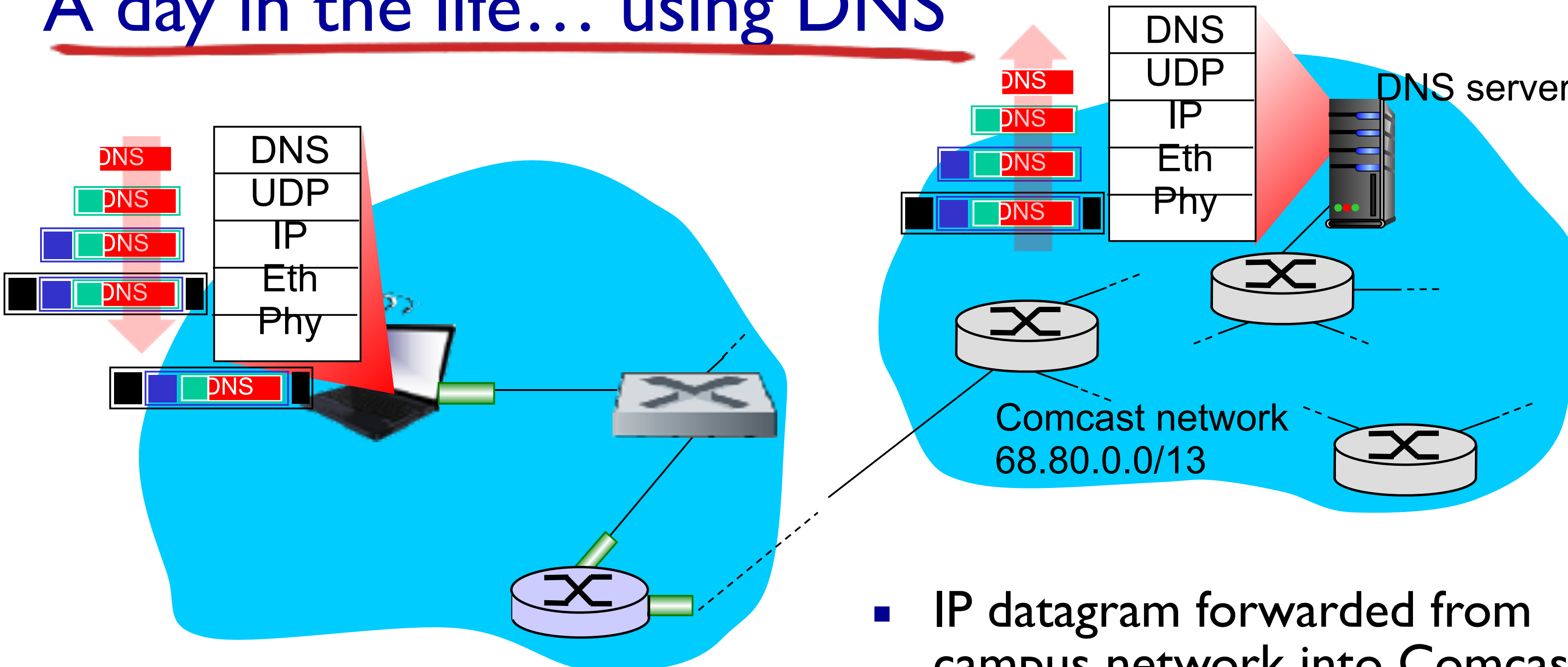


A day in the life… connecting to the Internet



- When a laptop connects to the network, it needs to:
  - Get an IP address to use
  - Learn which DNS server to use
  - Learn a "gateway" to the Internet
- Later in the semester, we will learn how DHCP enables this
- We will also learn how the laptop knows how to address packets and add headers at the network and link layers

A day in the life… using DNS



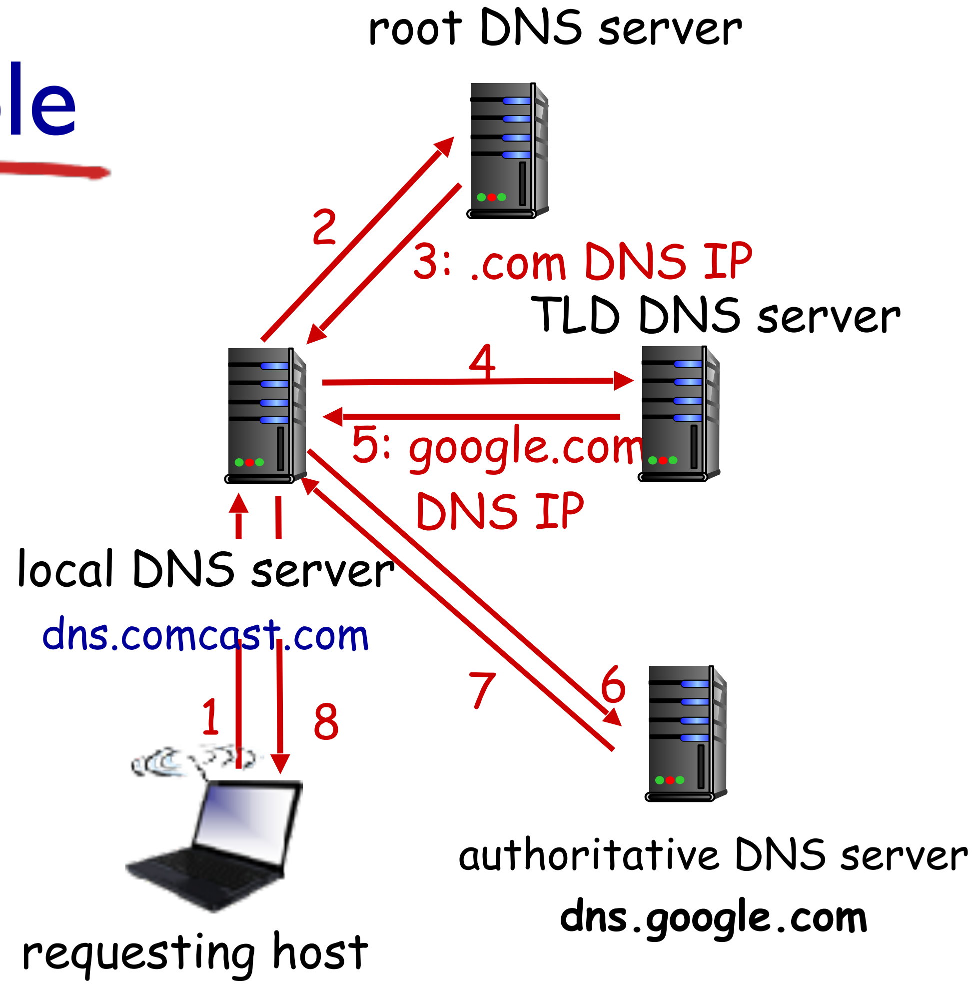
- IP datagram forwarded from campus network into Comcast network, routed (tables created by RIP, OSPF, IS-IS and/or BGP routing protocols) to DNS server
- demuxed to DNS server
- DNS server replies to client with IP address of www.google.com

DNS name resolution example

- laptop connecting via Comcast wants IP address for www.google.com

iterated query:

- contacted server replies with name of server to contact
- "I don't know this name, but ask this server"



DNS records

DNS: distributed database storing resource records (RR)

RR format: `(name, value, type, ttl)`

type=A

- **`name`** is hostname
- **`value`** is IP address

type=NS

- **`name`** is domain (e.g., foo.com)
- **`value`** is hostname of authoritative name server for this domain

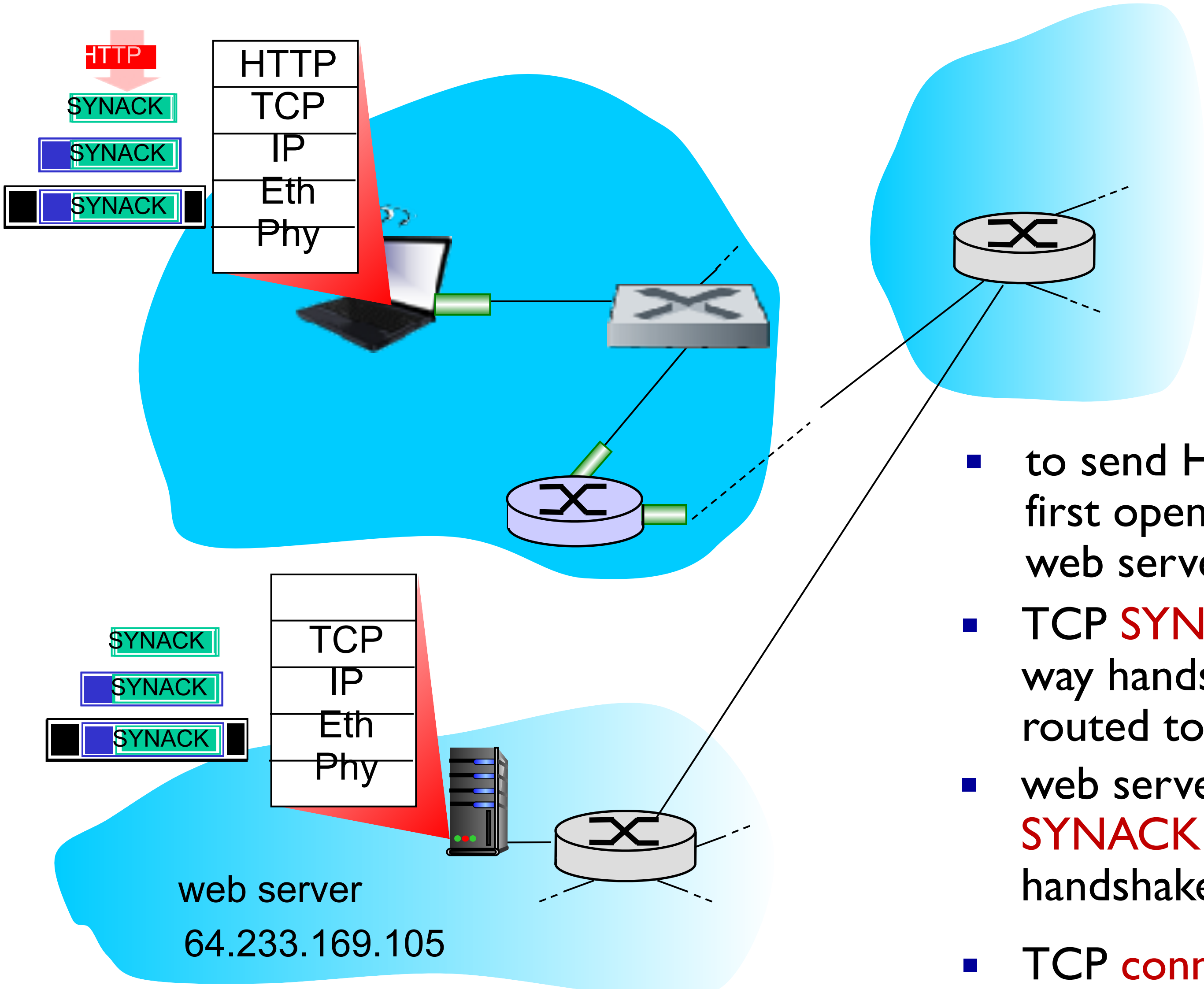
type=CNAME

- **`name`** is alias name for some "canonical" (the real) name
- **`www.ibm.com`** is really **`servereast.backup2.ibm.com`**
- **`value`** is canonical name

type=MX

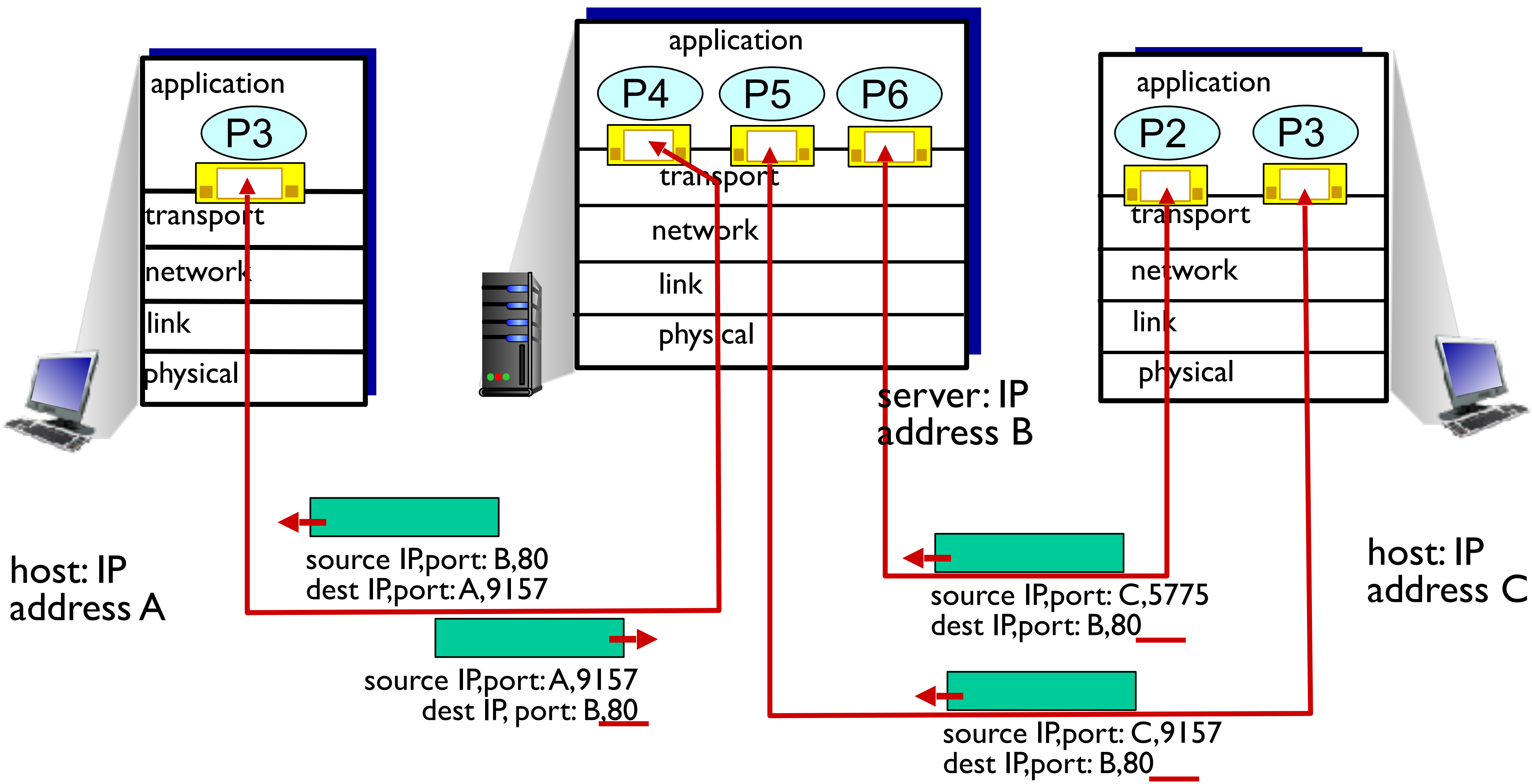
- **`value`** is name of mailserver associated with **`name`**

A day in the life…TCP connection carrying HTTP



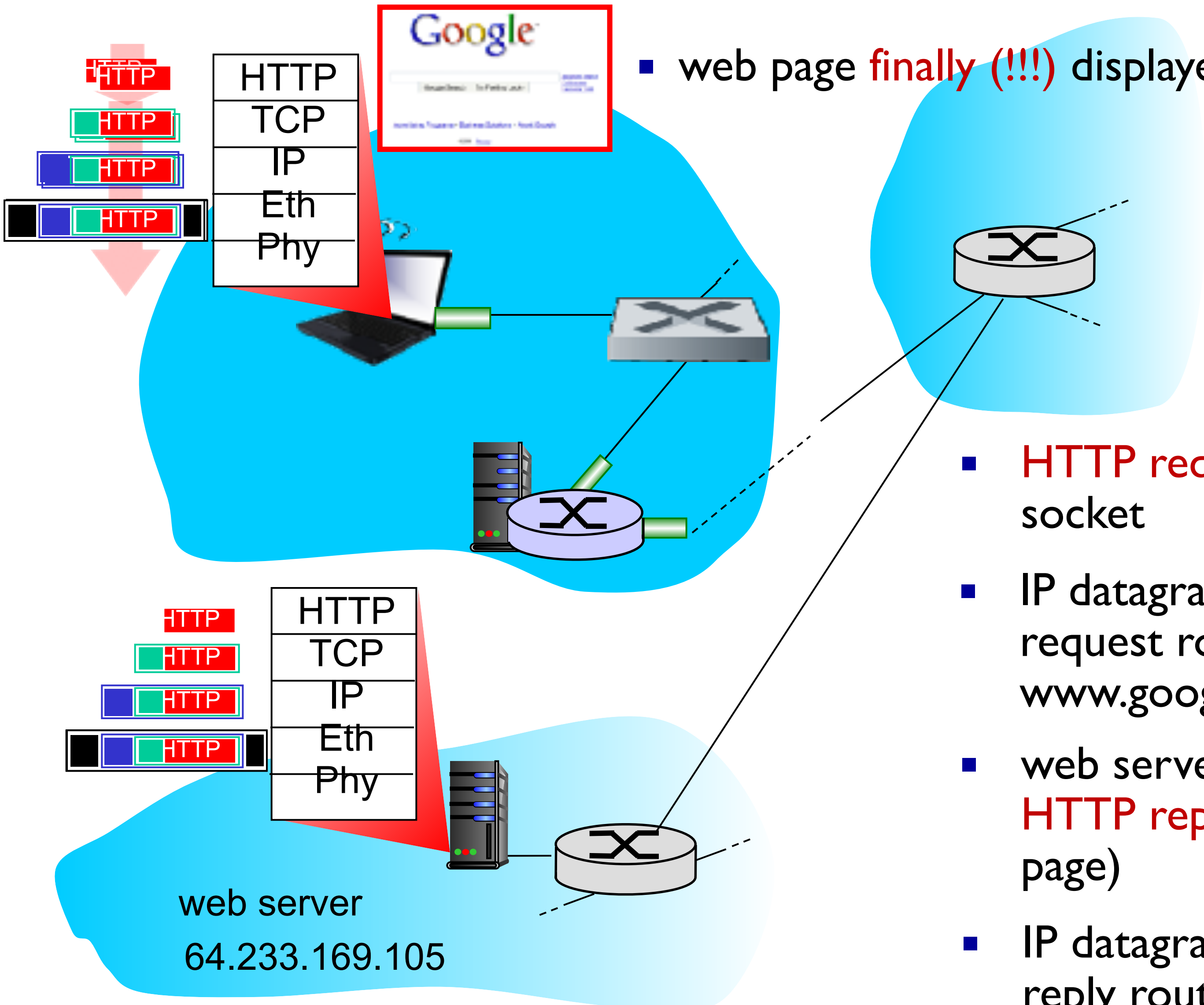
- to send HTTP request, client first opens TCP socket to web server
- TCP SYN segment (step 1 in 3-way handshake) inter-domain routed to web server
- web server responds with TCP SYNACK (step 2 in 3-way handshake)
- TCP connection established!

Connection-oriented demux: example



three segments, all destined to IP address: B,
dest port: 80 are demultiplexed to *different* sockets

A day in the life… HTTP request/reply

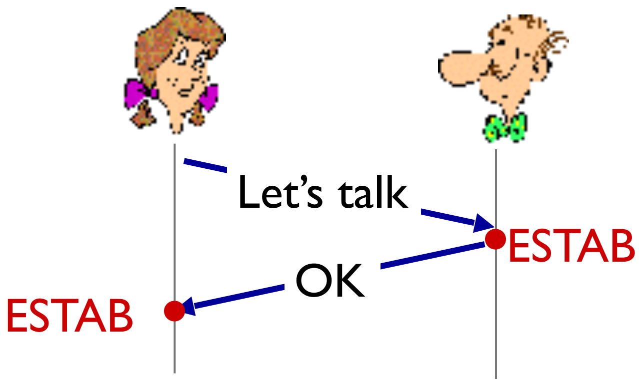


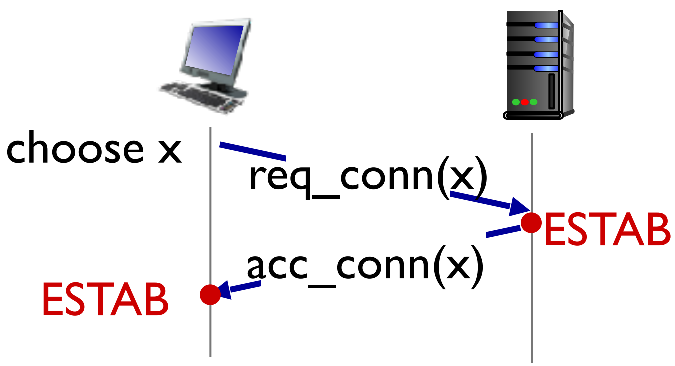
- web page finally (!!!) displayed

- 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

Agreeing to establish a connection

2-way handshake:



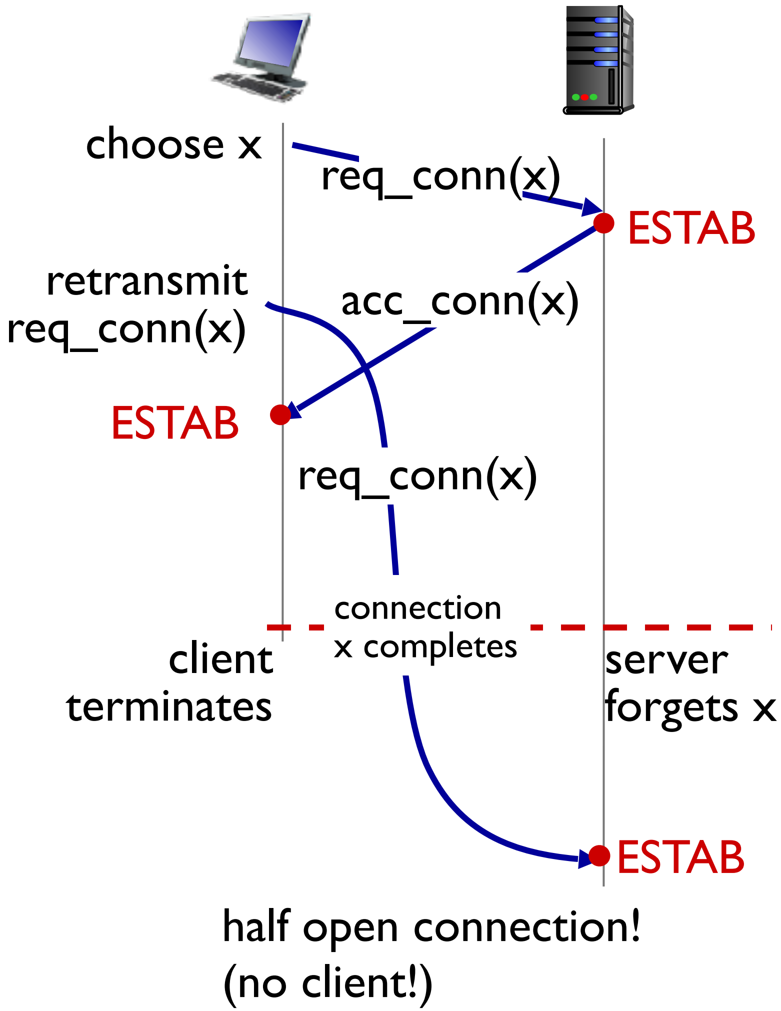


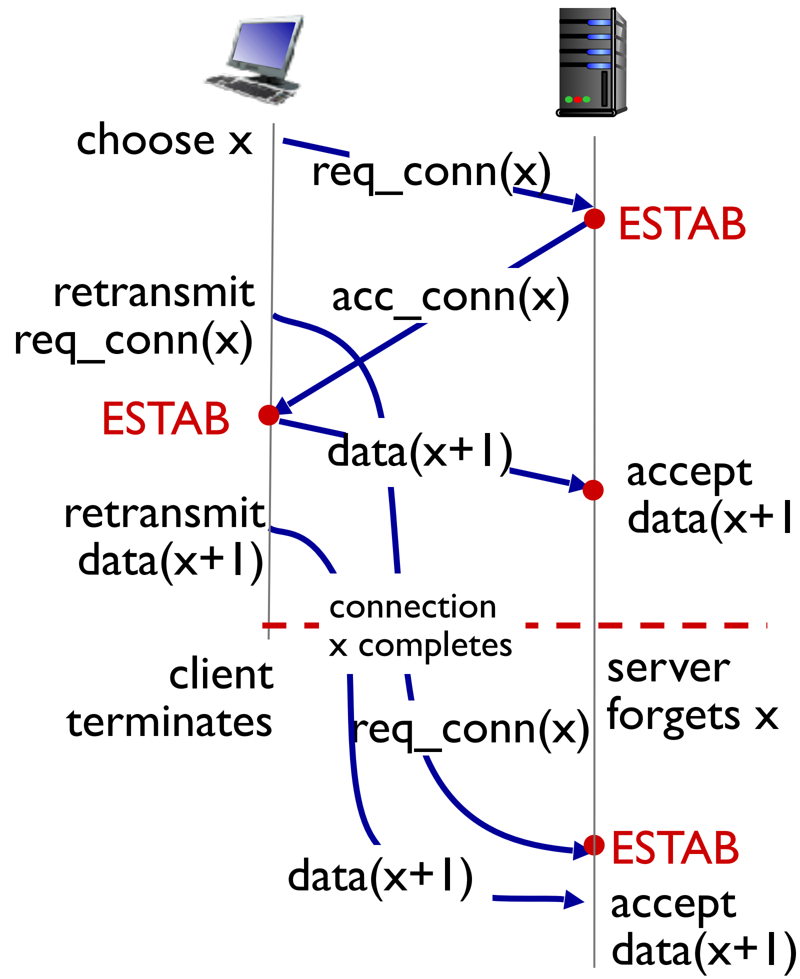
Q: will 2-way handshake always work in network?

- variable delays
- retransmitted messages (e.g. req_conn(x)) due to message loss
- message reordering
- can't "see" other side

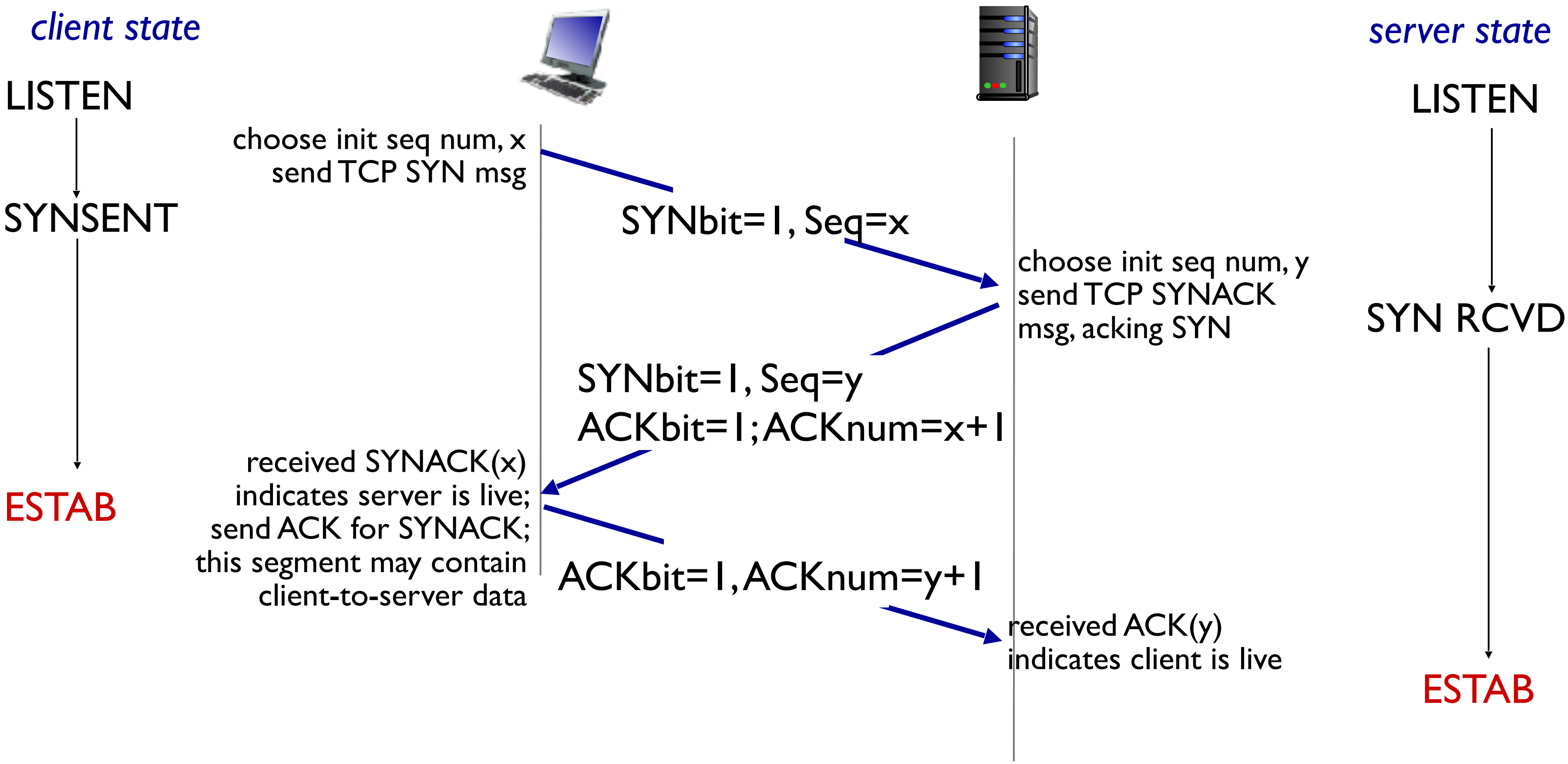
Agreeing to establish a connection

2-way handshake failure scenarios:

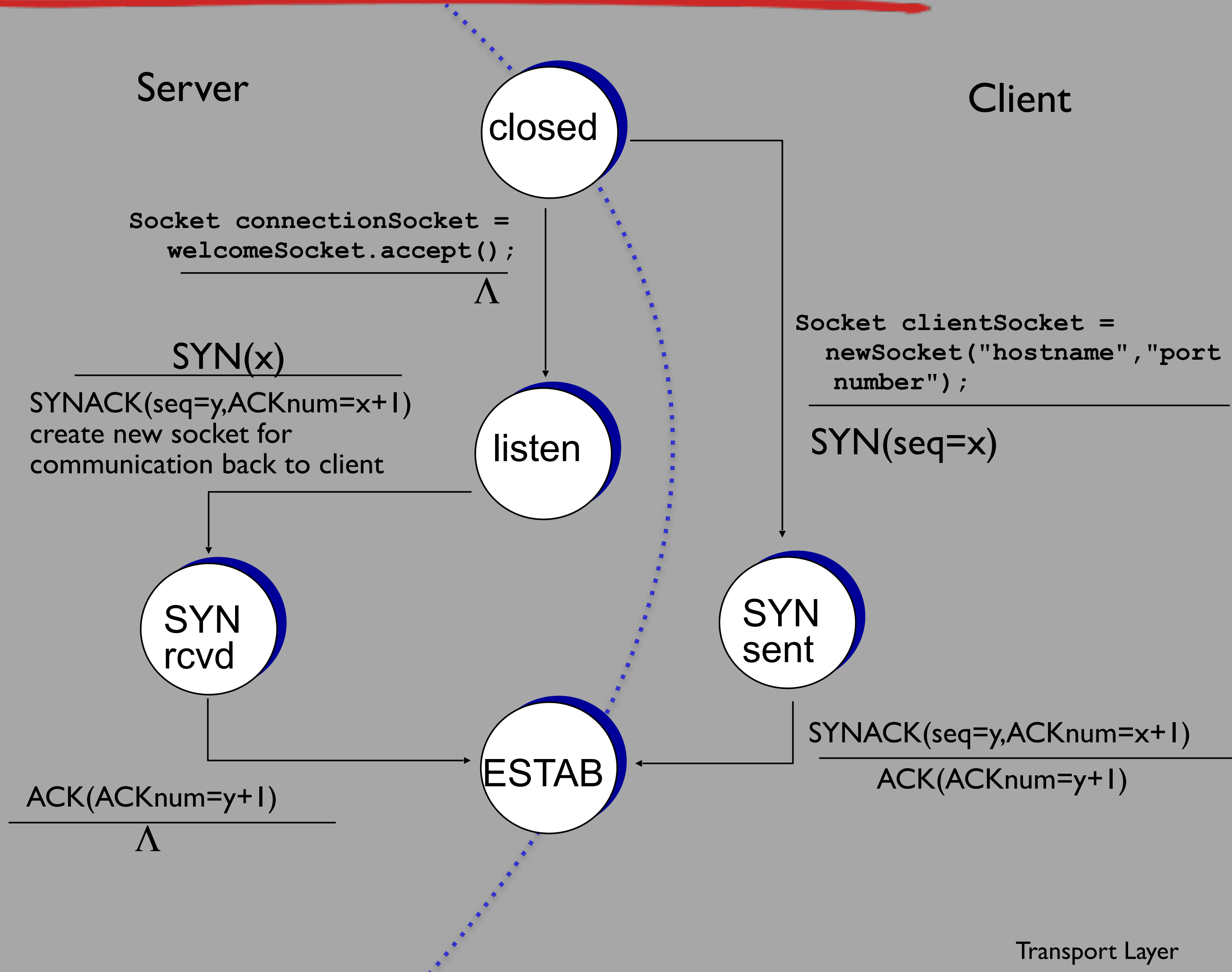




TCP 3-way handshake



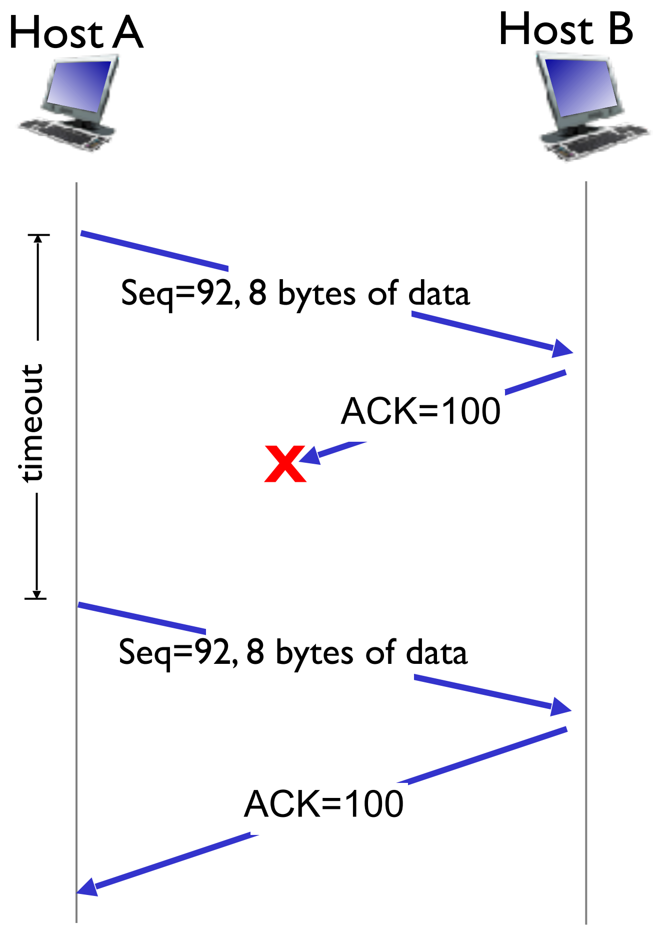
TCP 3-way handshake: FSM



Synthesis: Other pieces

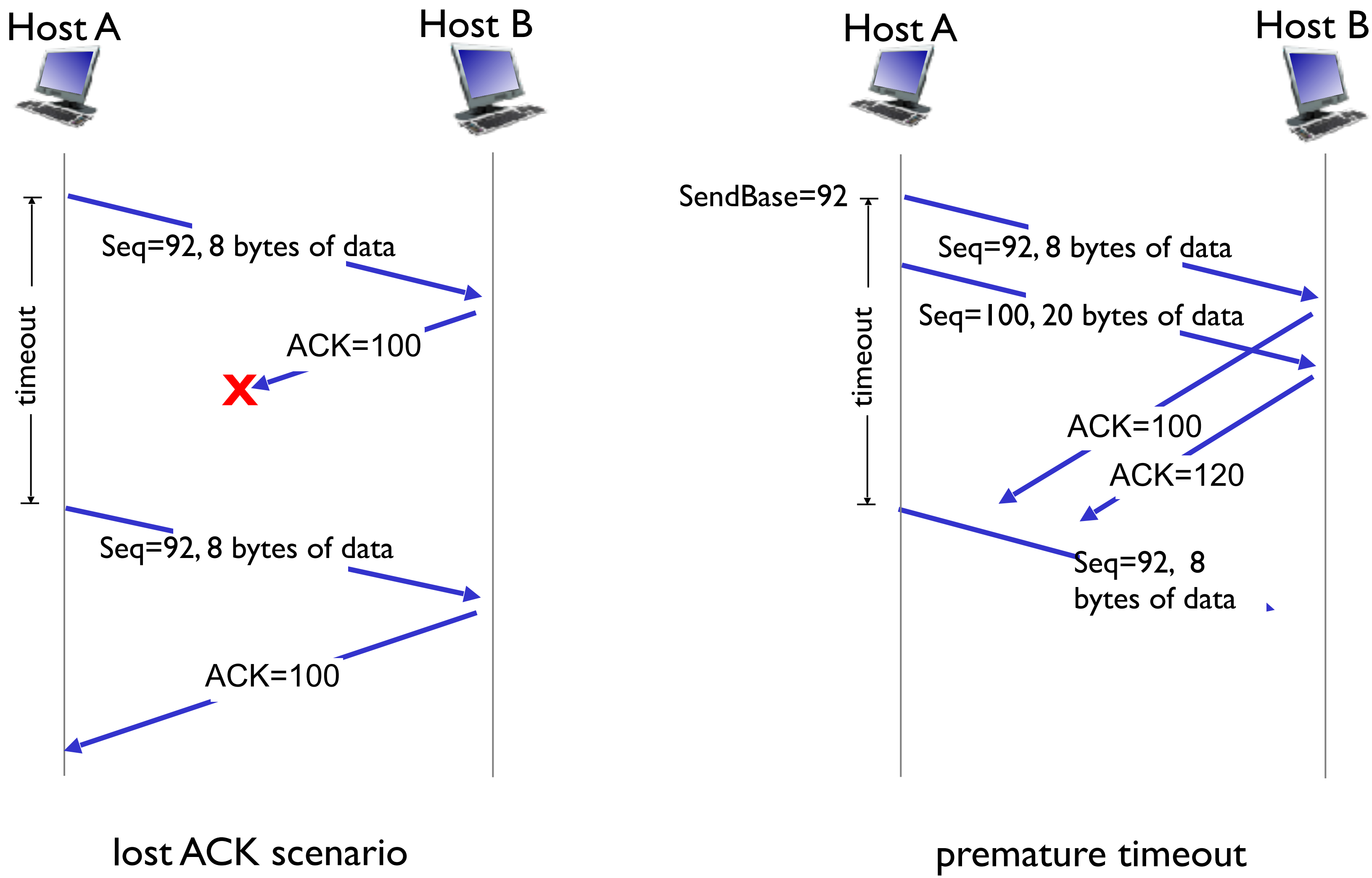
- TCP
  - reliable delivery

TCP: retransmission scenarios

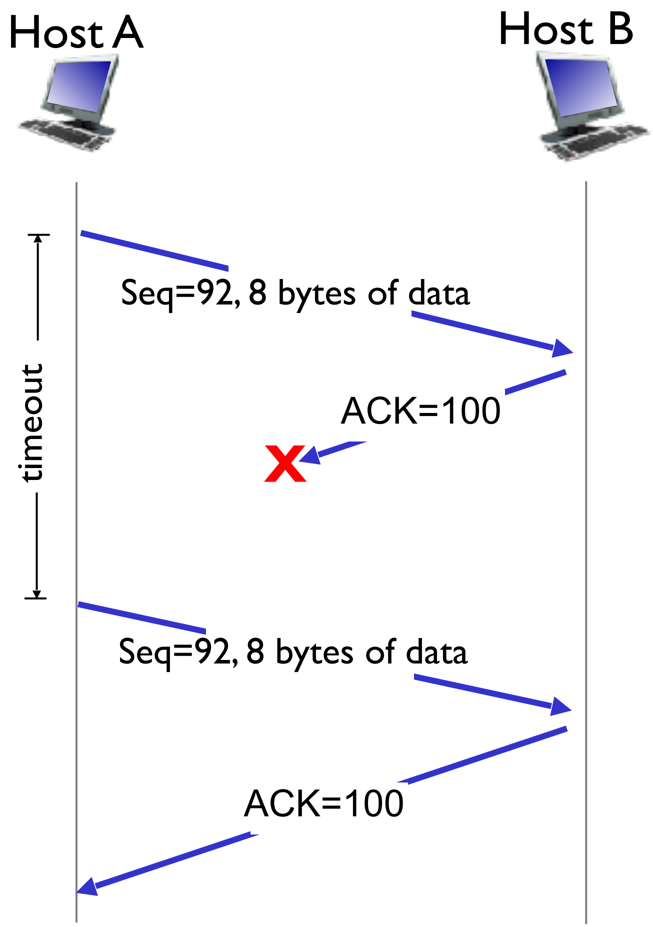


lost ACK scenario

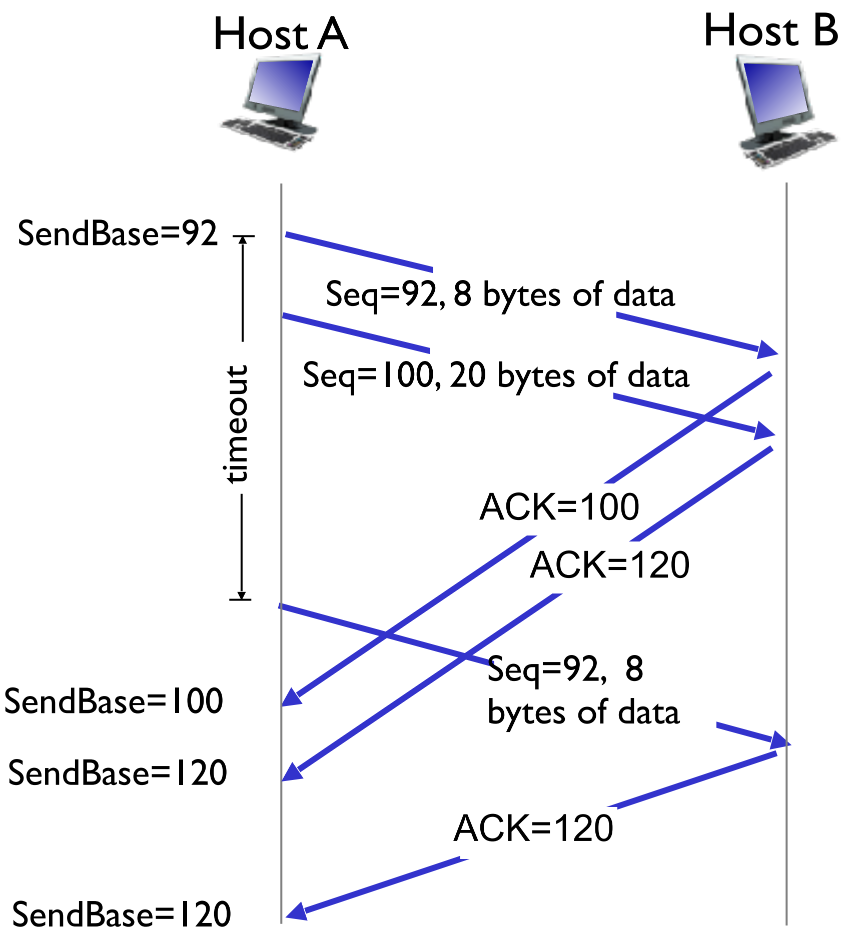
TCP: retransmission scenarios



TCP: retransmission scenarios

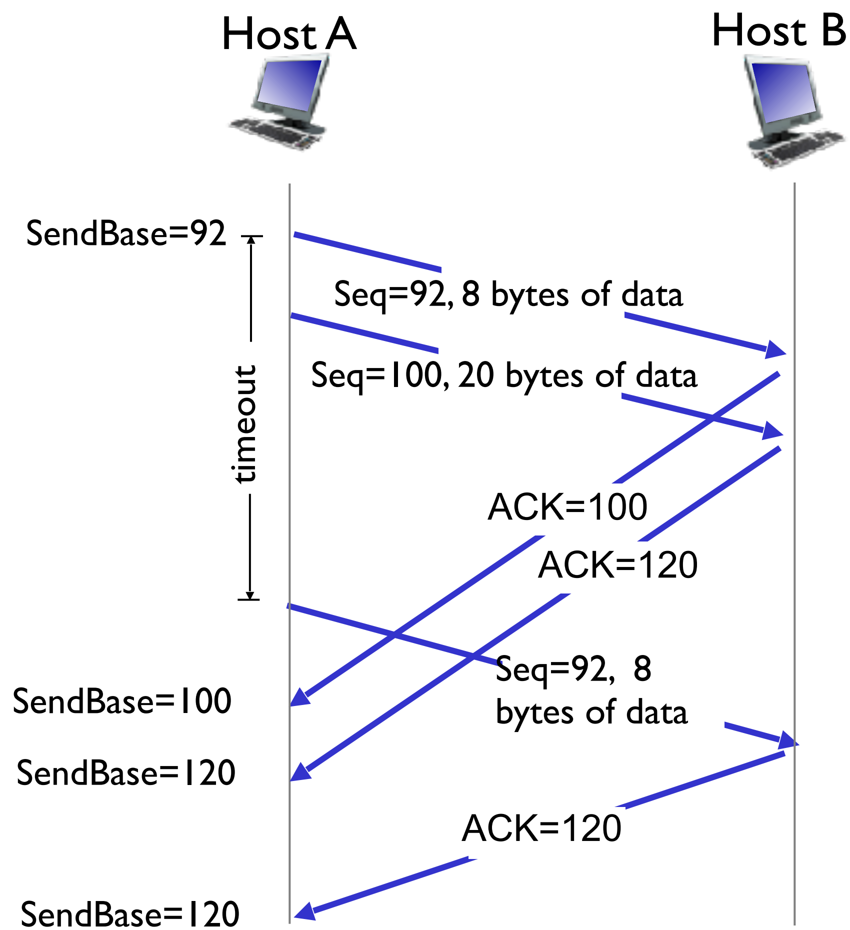


lost ACK scenario



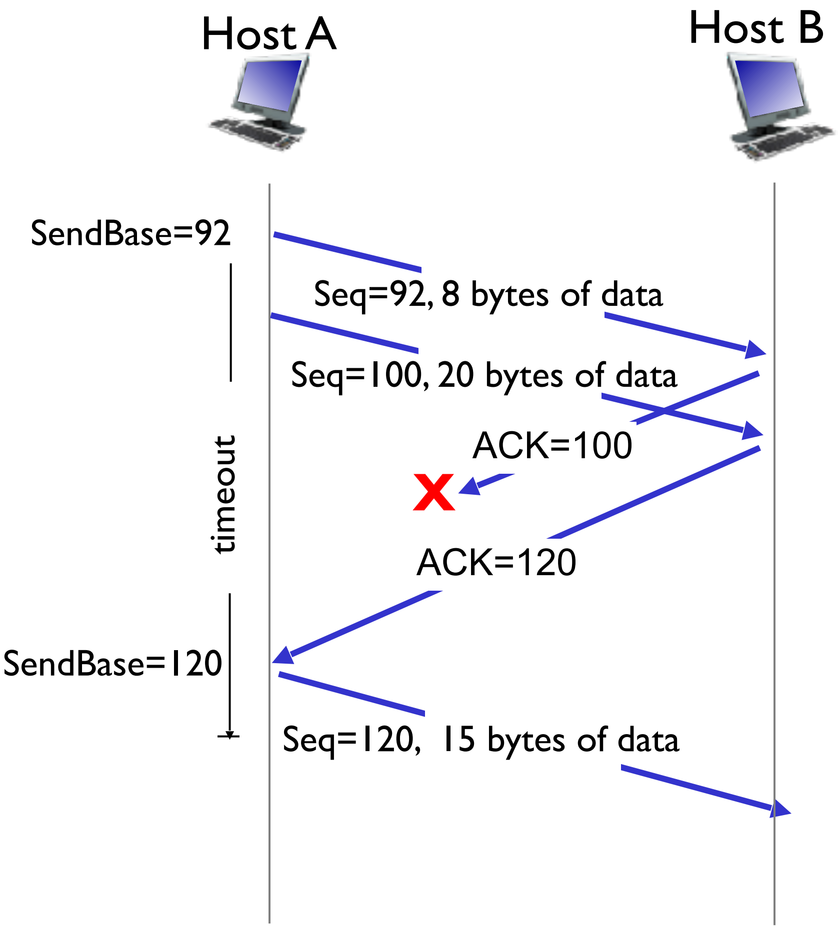
premature timeout

TCP: retransmission scenarios

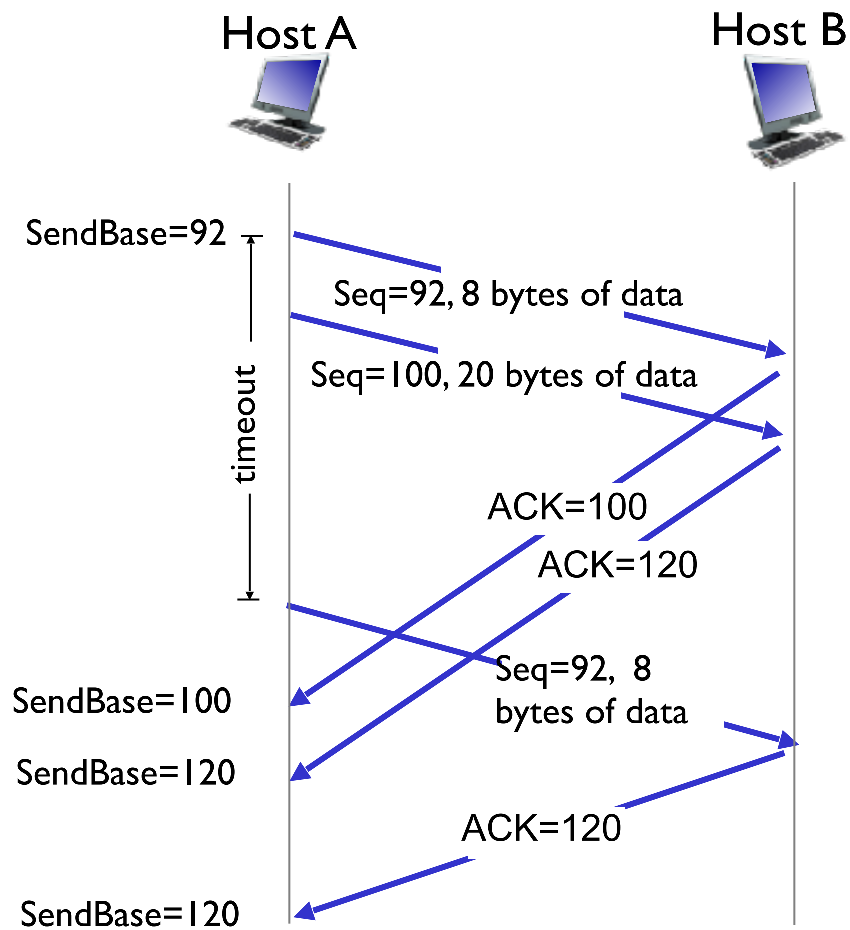


premature timeout

TCP: retransmission scenarios



cumulative ACK



premature timeout

TCP ACK generation [RFC 1122, RFC 2581]

event at receiver	*TCP receiver action*
arrival of in-order segment with expected seq #. All data up to expected seq # already ACKed	delayed ACK. Wait up to 500ms for next segment. If no next segment, send ACK
arrival of in-order segment with expected seq #. One other segment has ACK pending	immediately send single cumulative ACK, ACKing both in-order segments
arrival of out-of-order segment higher-than-expect seq. # . Gap detected	immediately send *duplicate ACK,* indicating seq. # of next expected byte
arrival of segment that partially or completely fills gap	immediate send ACK, provided that segment starts at lower end of gap

TCP fast retransmit

- time-out period often relatively long:
  - long delay before resending lost packet
- detect lost segments via duplicate ACKs.
  - sender often sends many segments back-to-back
  - if segment is lost, there will likely be many duplicate ACKs.

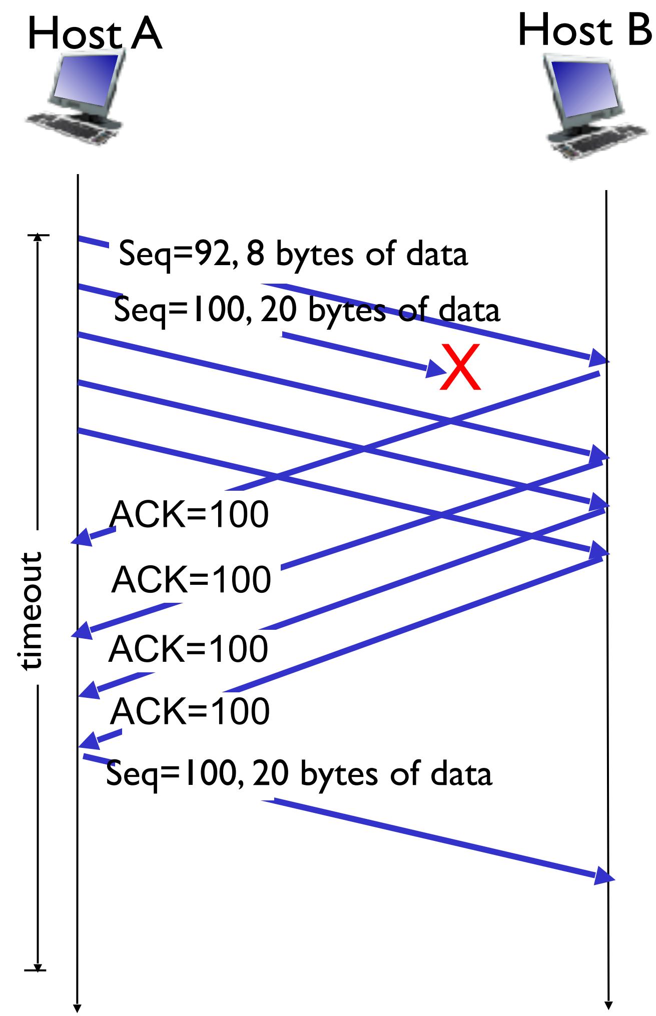
TCP fast retransmit

if sender receives 3 ACKs for same data

("triple duplicate ACKs"), resend unacked segment with smallest seq #

- likely that unacked segment lost, so don't wait for timeout

TCP fast retransmit



fast retransmit after sender receipt of triple duplicate ACK

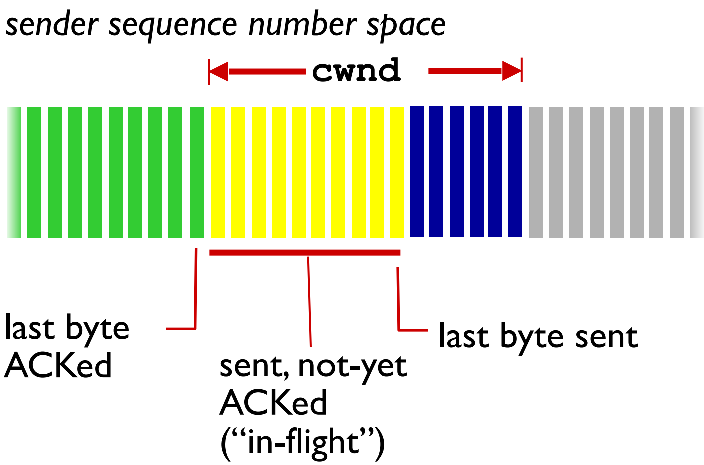
Synthesis: Other pieces

- TCP
  - reliable delivery
  - why flow control?

Synthesis: Other pieces

- TCP
  - reliable delivery
  - why flow control?
  - why congestion control?

TCP Congestion Control: details



- sender limits transmission:

$$\texttt{LastByteSent-}\ \texttt{LastByteAcked} \leq \texttt{cwnd}$$

- **`cwnd`** is dynamic, function of perceived network congestion

TCP sending rate:

- *roughly:* send cwnd bytes, wait RTT for ACKS, then send more bytes

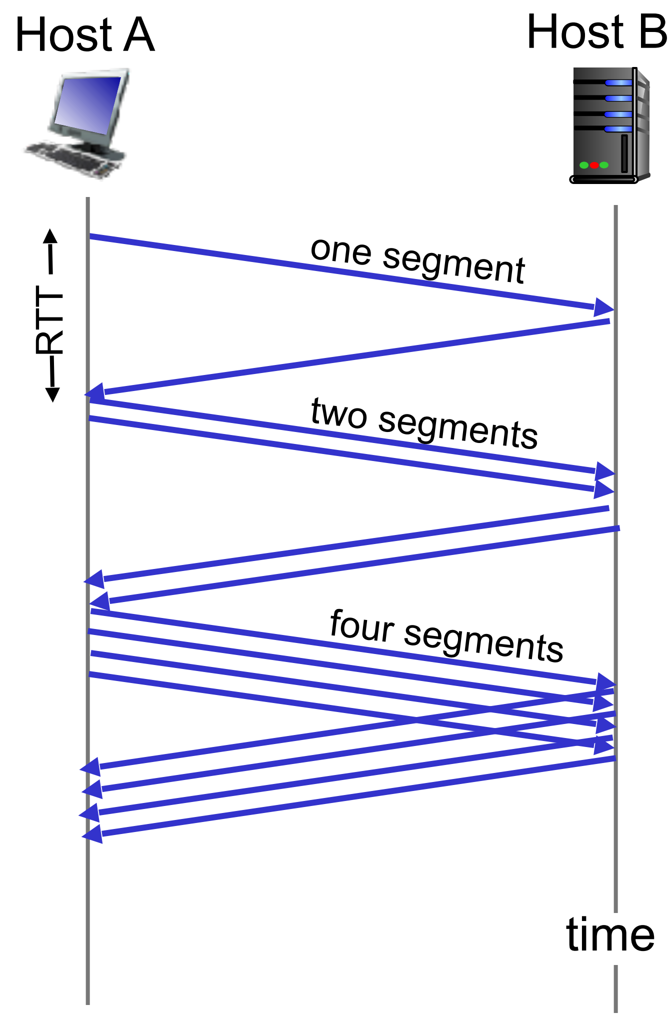
$$\text{rate} \approx \frac{\text{cwnd}}{\text{RTT}} \text{ bytes/sec}$$

TCP Stages

- slow start
- congestion avoidance
- fast recovery

TCP Slow Start

- when connection begins, increase rate exponentially until first loss event:
  - initially **`cwnd`** = 1 MSS
  - double **`cwnd`** every RTT
  - done by incrementing **`cwnd`** for every ACK received
- *summary:* initial rate is slow but ramps up exponentially fast



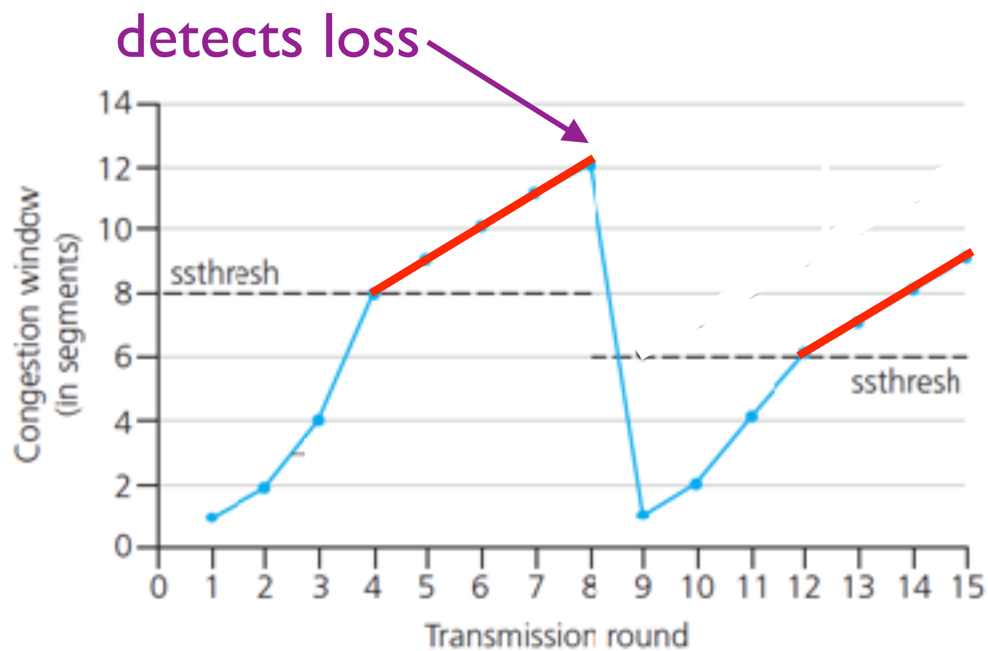
TCP: detecting, reacting to loss

- on loss:
  - **`cwnd`** set to 1 MSS

Switching from Slow Start (exponential) to Congestion Avoidance (linear)

Q: when should the **exponential** increase switch to **linear**?

A: when `cwnd` gets to 1/2 of its value before timeout.



Implementation:

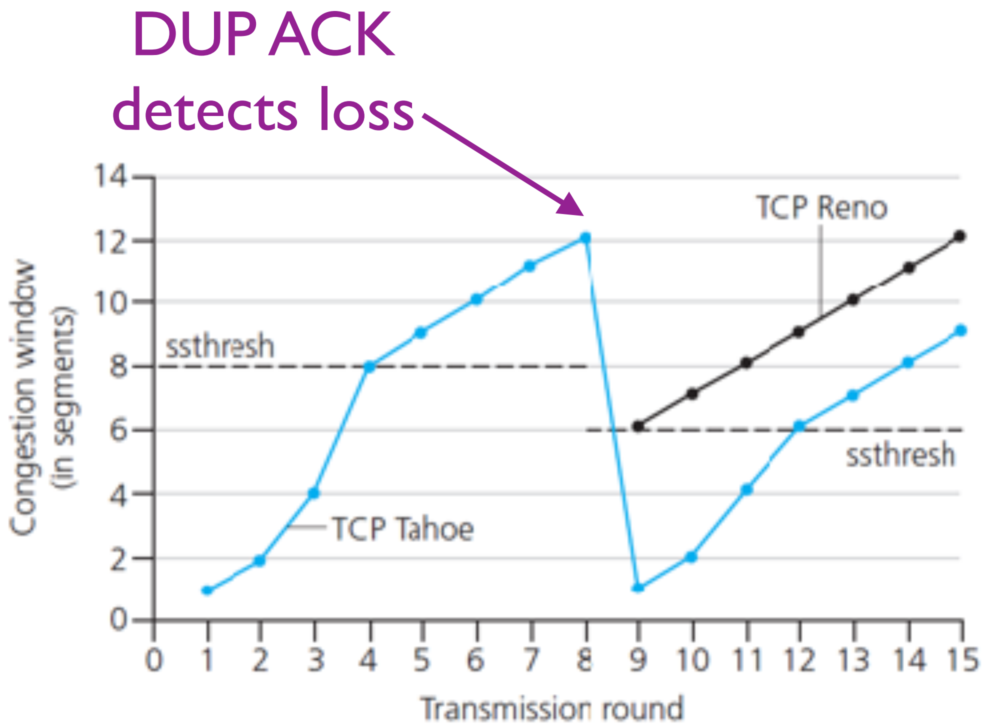
- variable `ssthresh`
- on loss event, `ssthresh` is set to 1/2 of `cwnd` just before loss event

* Check out the online interactive exercises for more examples: http://gaia.cs.umass.edu/kurose_ross/interactive/

TCP: detecting, reacting to loss

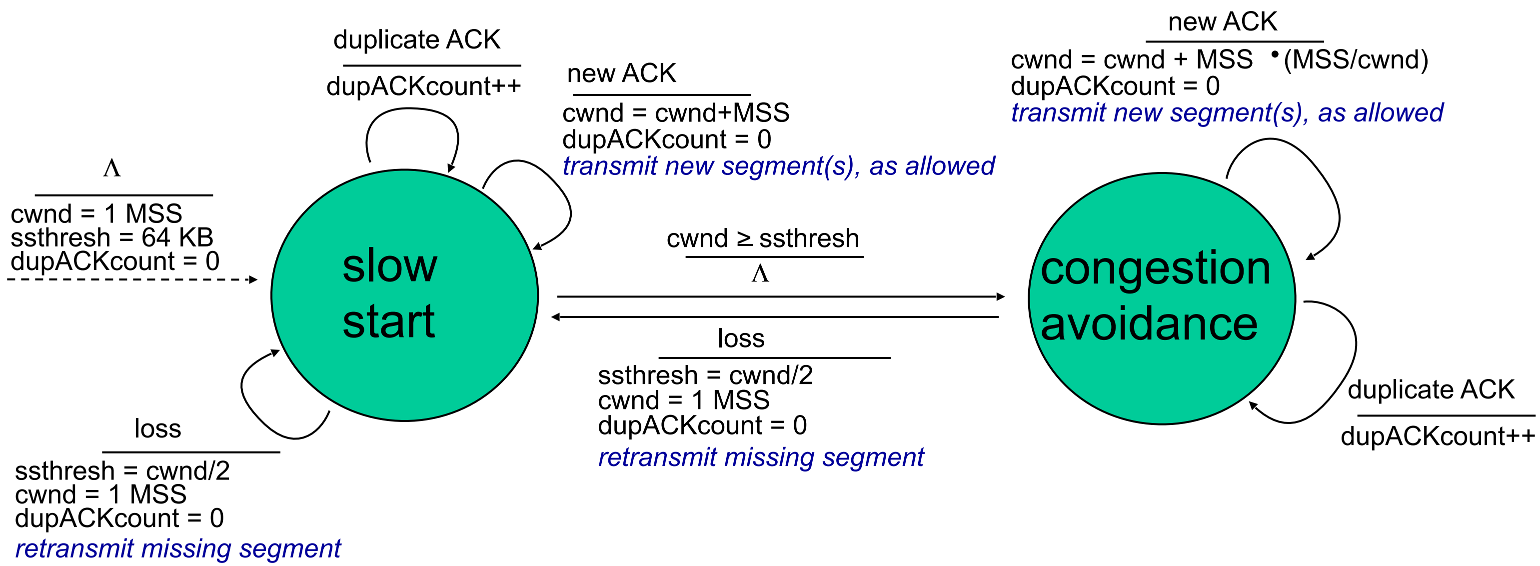
- on loss indicated by timeout:
  - **`ssthresh`** set to half of **`cwnd`**
  - **`cwnd`** set to 1 MSS
  - *slow start* (exponentially growth) up to **`ssthresh`**, then *congestion avoidance* (linear growth)
- loss indicated by 3 duplicate ACKs:
  - dup ACKs indicate network delivering some segments
  - TCP Tahoe: same as for timeouts above
  - TCP Reno's *fast recovery*:
    - **`cwnd`** is only cut in half
  - why 3 dup ACKs? Why not 1? Why not 5?

TCP Reno's *Fast Recovery*

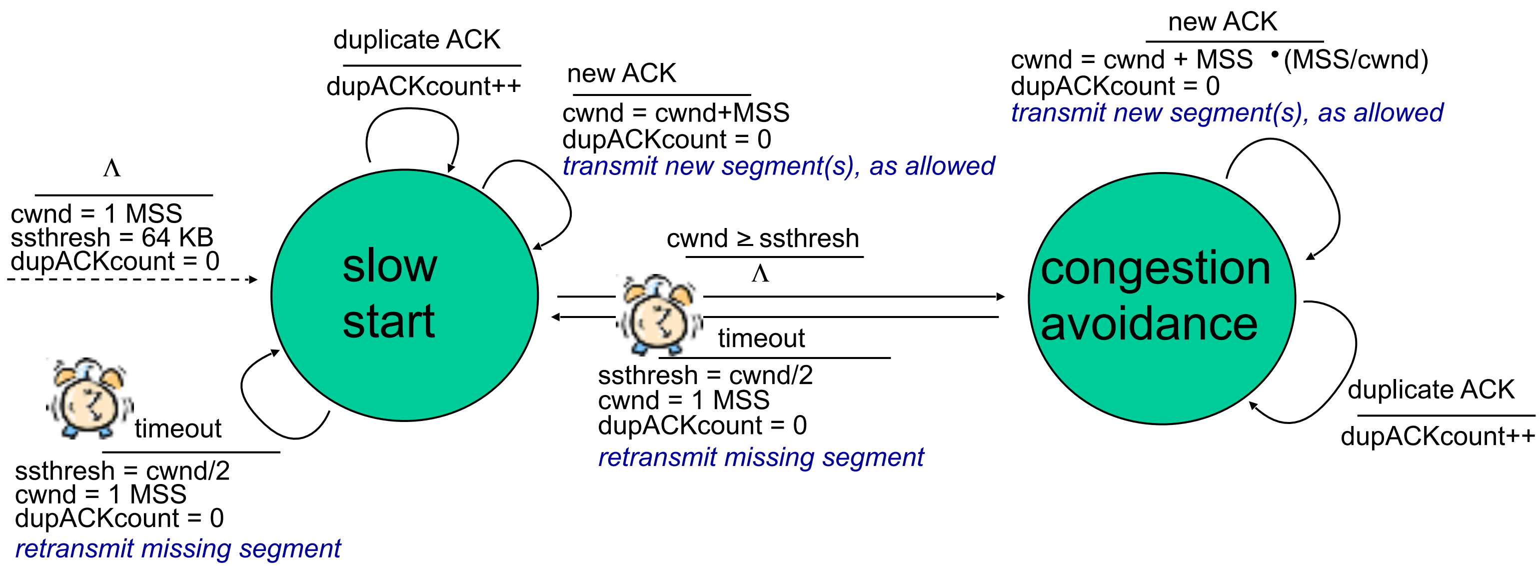


* Check out the online interactive exercises for more examples: http://gaia.cs.umass.edu/kurose_ross/interactive/

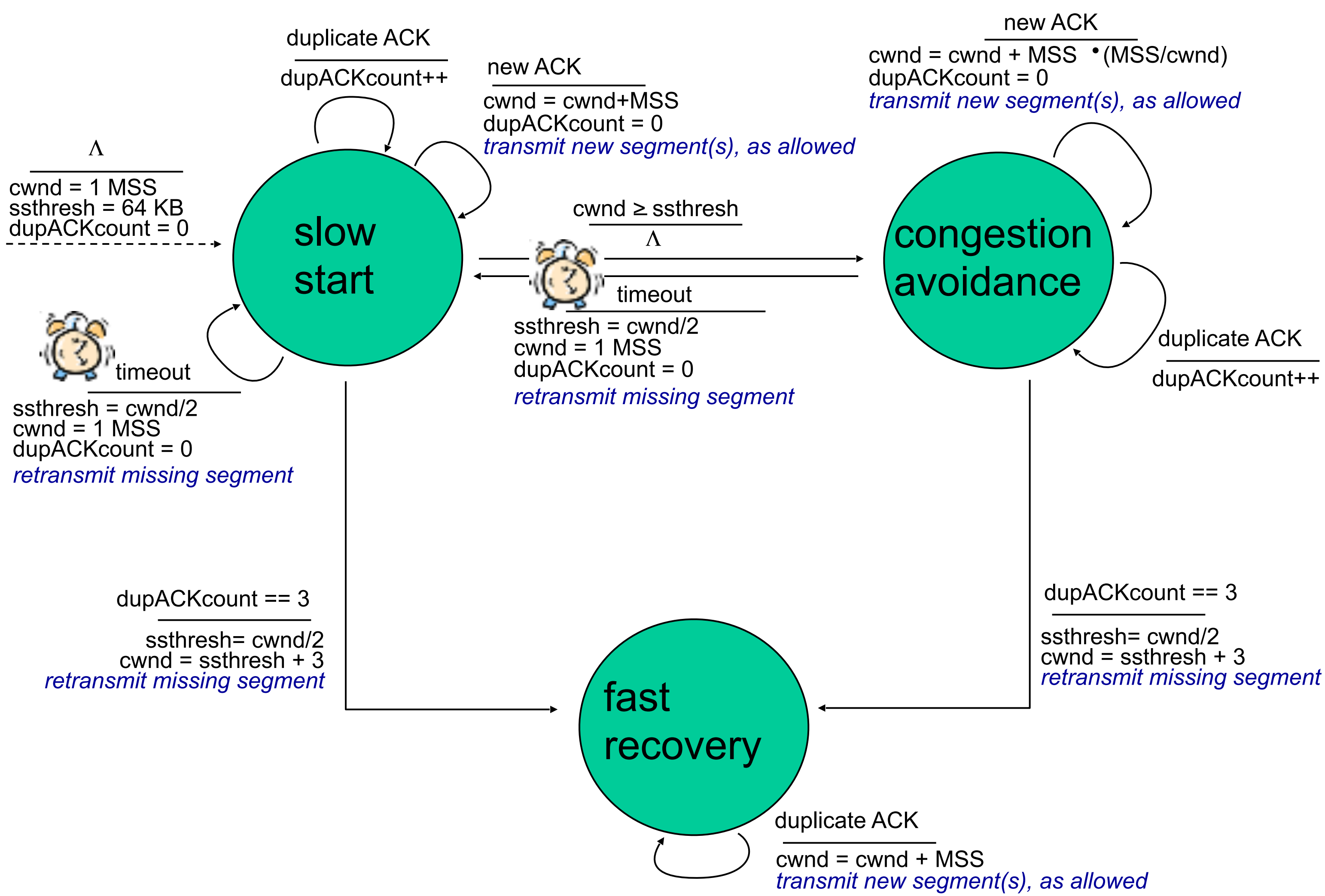
Summary: TCP Congestion Control



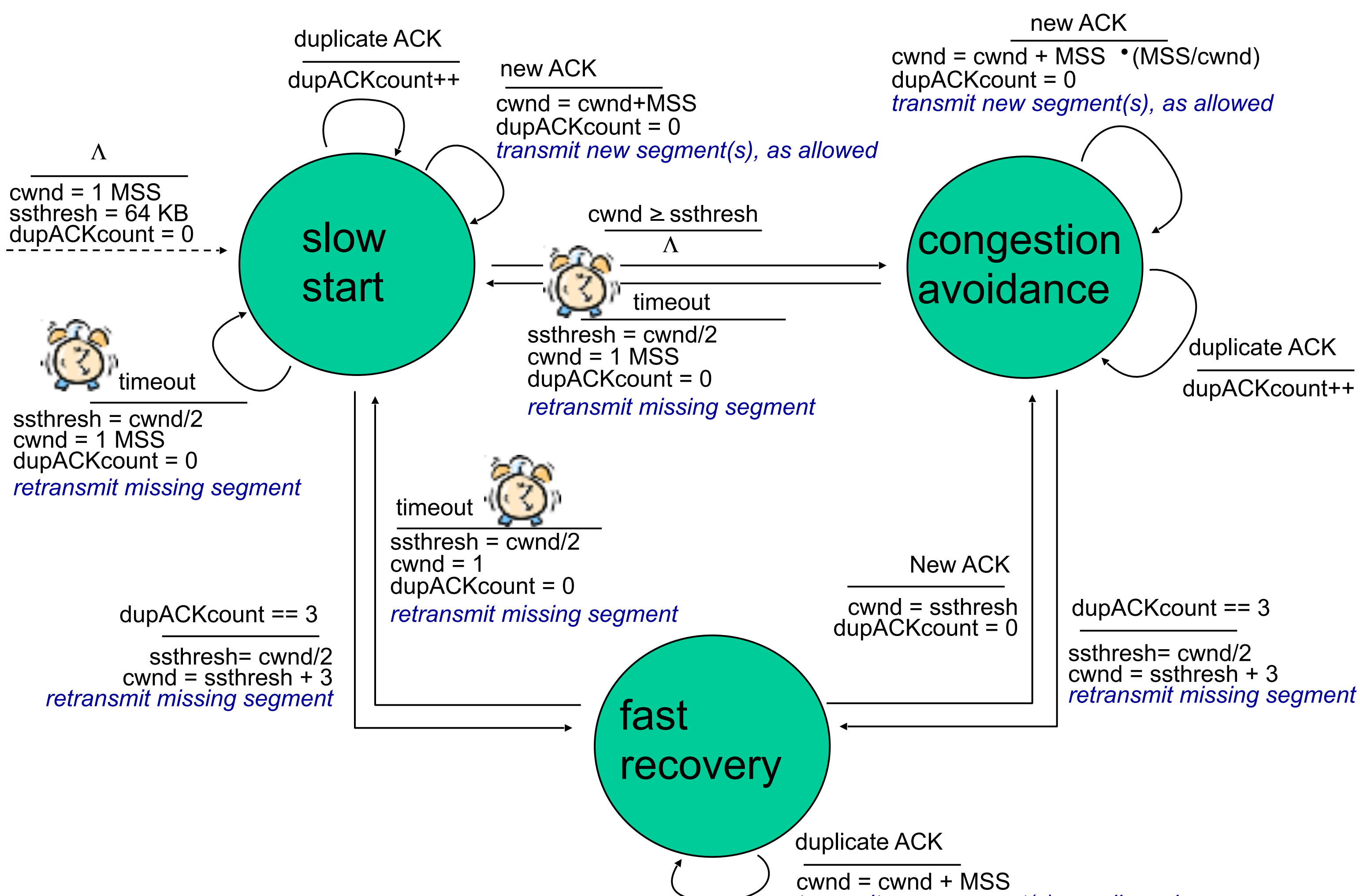
Summary: TCP Congestion Control



Summary: TCP Congestion Control

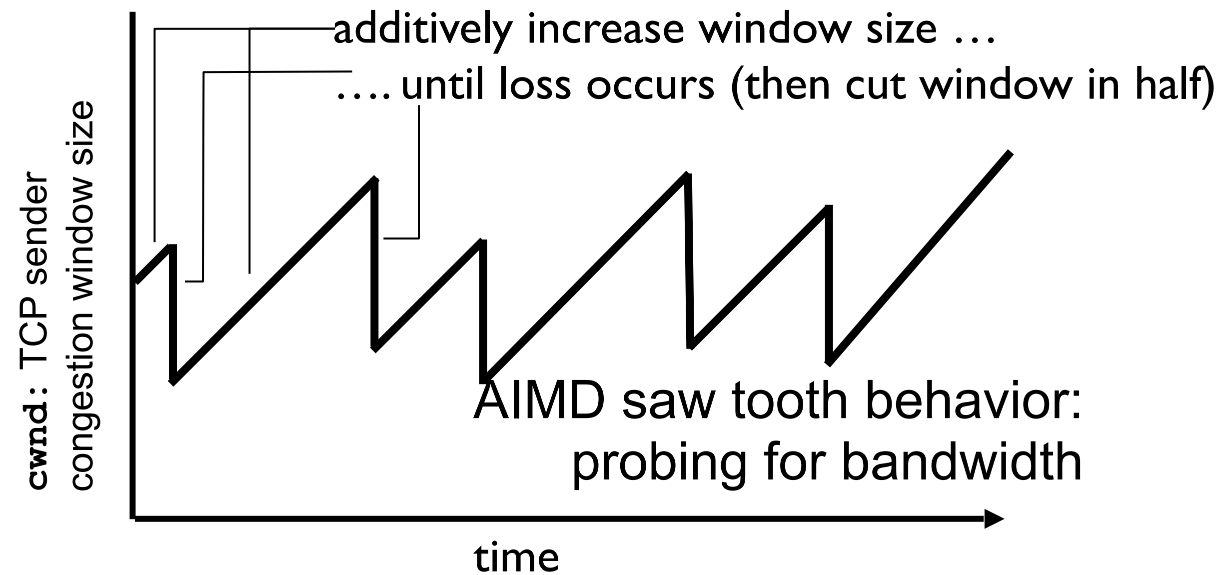


Summary: TCP Congestion Control



TCP congestion control: additive increase multiplicative decrease

- *approach:* sender increases rate (window size), probing for usable bandwidth, until loss occurs
  - *additive increase:* increase **`cwnd`** by 1 MSS every RTT until loss detected
  - *multiplicative decrease*: cut **`cwnd`** in half after loss



Synthesis: Other pieces

- TCP
  - reliable delivery
  - why flow control?
  - why congestion control?
- Video
  - why ABR?
  - why CDN?

Day 19: Network Layer Overview and Addresses

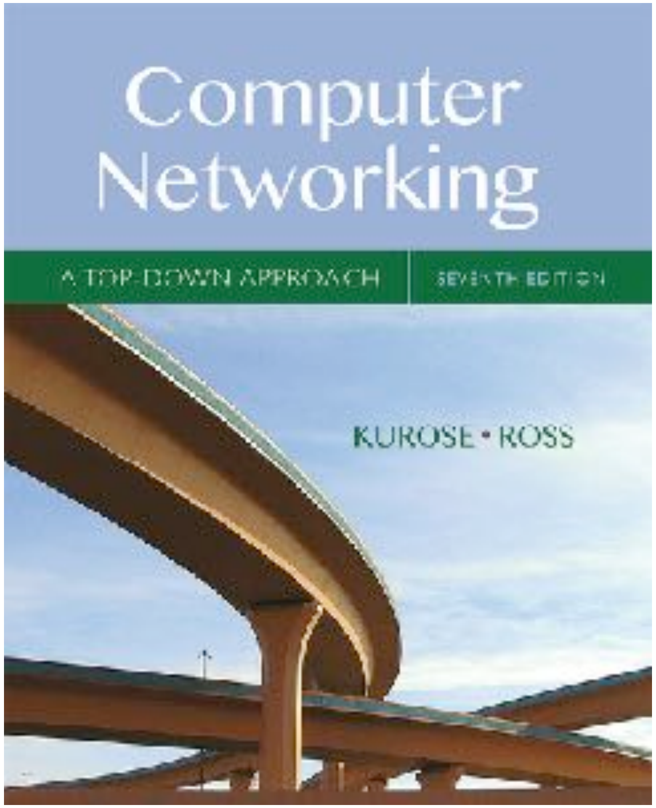
CSEE 4119
Computer Networks
Ethan Katz-Bassett

COLUMBIA UNIVERSITY
IN THE CITY OF NEW YORK

Chapter 4
Network Layer: The Data Plane

CSEE 4119
Computer Networks
Ethan Katz-Bassett

Adapted by Ethan Katz-Bassett from slides provided by Kurose/Ross



Computer Networking: A Top Down Approach

7th edition
Jim Kurose, Keith Ross
Pearson/Addison Wesley
April 2016

Chapter 4: network layer

chapter goals:

- understand principles behind network layer services, focusing on data plane:
  - network layer service models
  - forwarding versus routing
  - how a router works
  - generalized forwarding
  - Internet architecture
- instantiation, implementation in the Internet
  - IP protocol
  - NAT, middleboxes

Chapter 4: outline

Next

4.1 Overview of Network layer
- data plane
- control plane

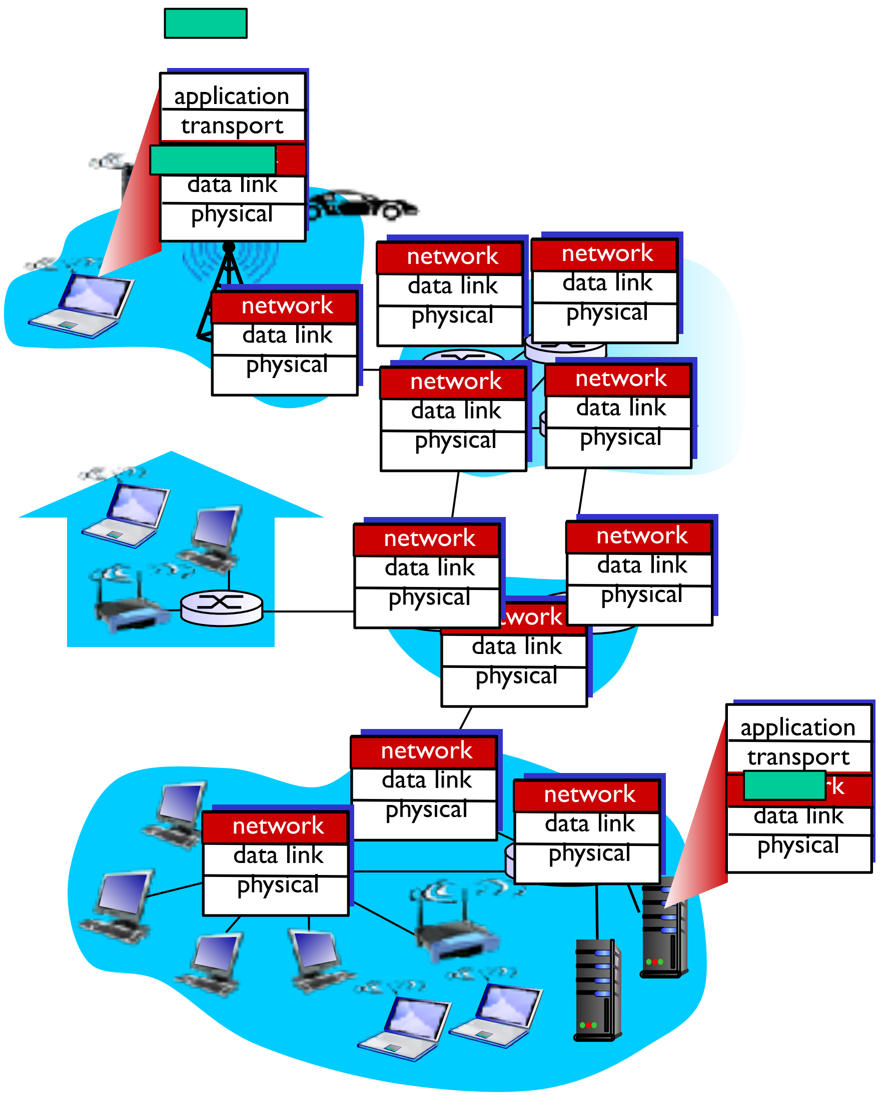
4.2 What's inside a router

4.3 IP: Internet Protocol
- datagram format
- fragmentation
- IPv4 addressing
- network address translation
- IPv6

4.4 Generalized Forward and SDN
- match
- action
- OpenFlow examples of match-plus-action in action

Network layer

- transport segment from sending to receiving host
  - sender: network layer encapsulates segments into datagrams, passes to link layer
  - receiver: it delivers segments to transport layer
- network layer protocols in *every* host *and every* router (but not in switches)
- router:
  - examines header fields in all IP datagrams passing through it
  - moves datagrams from input ports to output ports to transfer datagrams along end-end path



Two key network-layer functions

network-layer functions:

- *forwarding:* move packets from router's input to appropriate router output
- *routing:* determine route taken by packets from source to destination
  - *routing algorithms*

analogy: taking a trip

- *forwarding:* process of getting through single interchange

forwarding

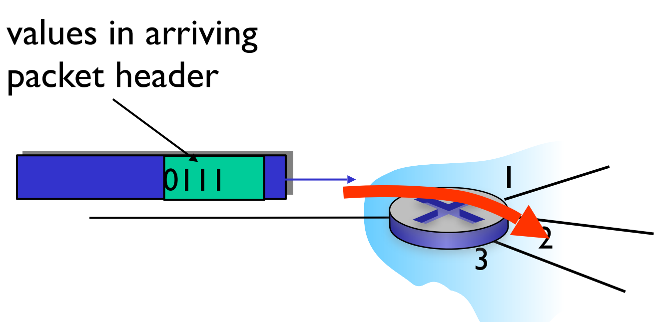
- *routing:* process of planning trip from source to destination

routing

Network layer: data plane, control plane

Data plane

- local, per-router function
- determines how datagram arriving on router input port is forwarded to router output port
- forwarding function

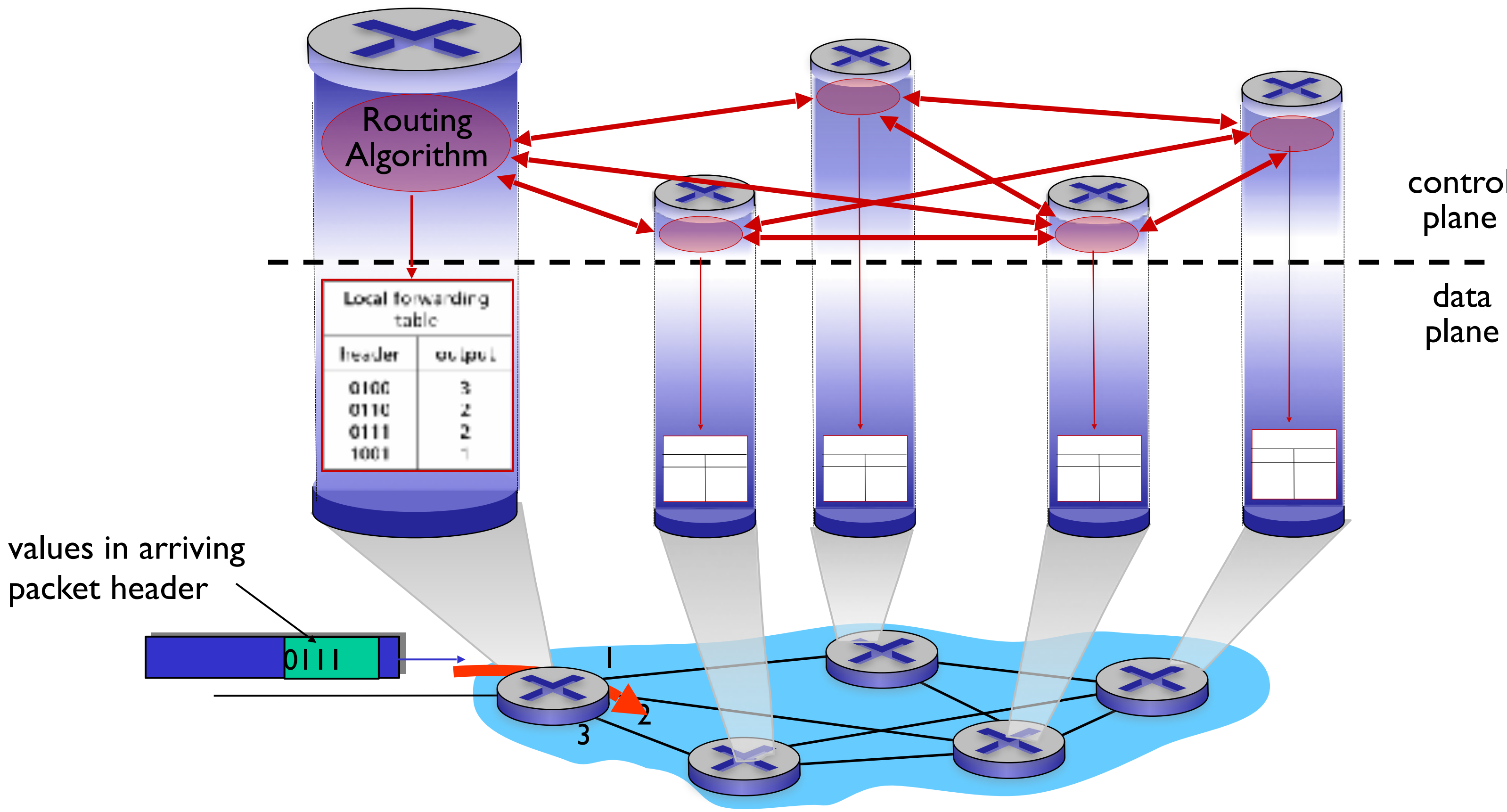


Control plane

- network-wide logic
- determines how datagram is routed among routers along end-end path from source host to destination host
- two control-plane approaches:
  - *traditional routing algorithms:* implemented in routers
  - *software-defined networking (SDN)*: implemented in (remote) servers

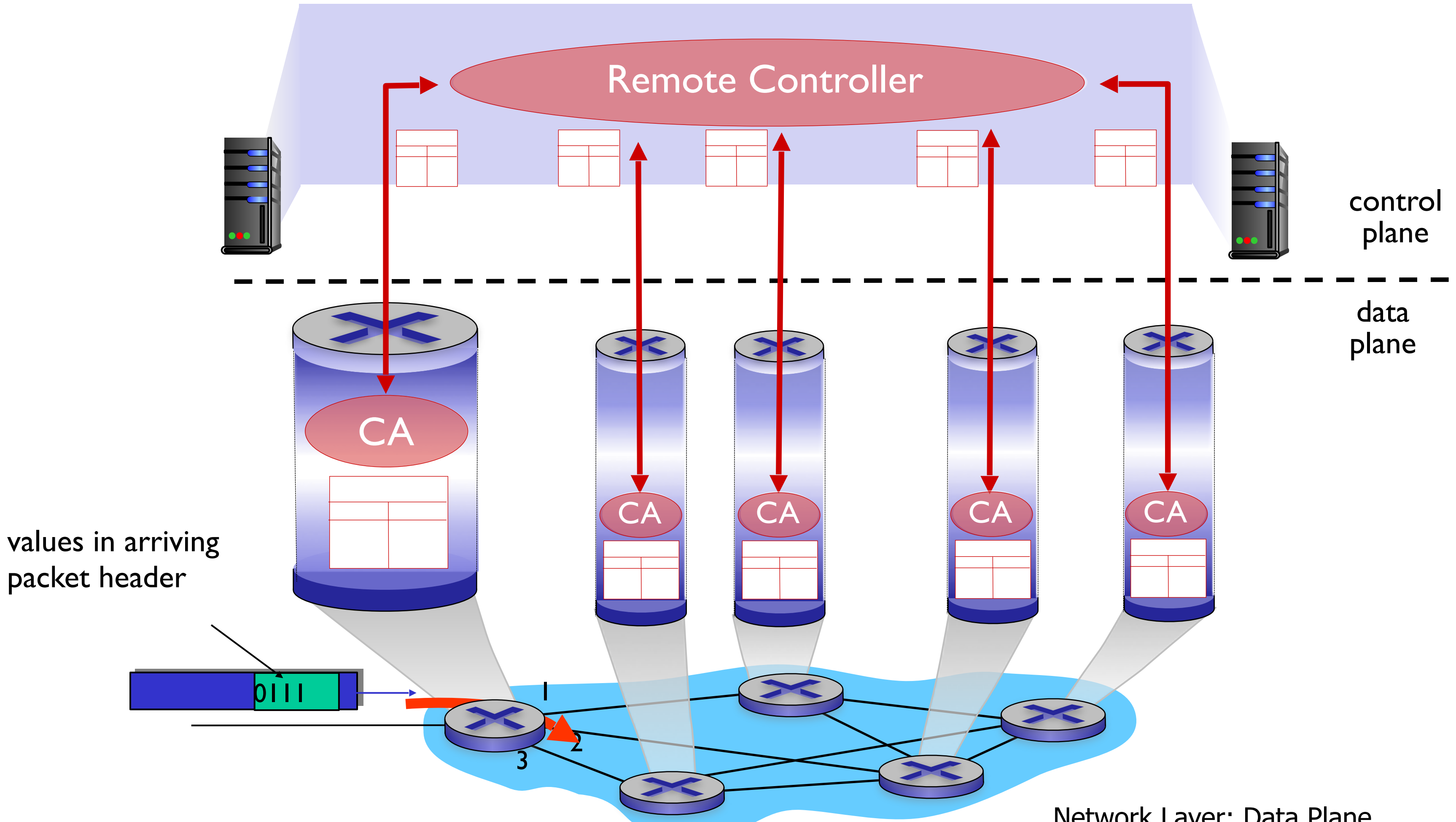
Per-router control plane

Individual routing algorithm components *in each and every router* interact in the control plane



Software-Defined Networking (SDN) Logically centralized control plane

A distinct (typically remote) controller interacts with local control agents (CAs)



Network service model

Q: What *service model* for "channel" transporting datagrams from sender to receiver?

example services for individual datagrams:

- guaranteed delivery
- guaranteed delivery with less than 40 msec delay

example services for a flow of datagrams:

- in-order datagram delivery
- guaranteed minimum bandwidth to flow
- restrictions on changes in inter-packet spacing

Network layer service models:

Network Architecture	Service Model	Guarantees ? Bandwidth	Loss	Order	Timing	Congestion feedback
Internet	best effort	none	no	no	no	no (inferred via loss)

Internet "best effort" service model

No guarantees on:

i. successful datagram delivery to destination
ii. timing or order of delivery
iii. bandwidth available to end-end flow

Network layer service models:

Network Architecture	Service Model	Guarantees ? Bandwidth	Loss	Order	Timing	Congestion feedback
Internet	best effort	none	no	no	no	no (inferred via loss)
ATM	Constant Bit Rate	constant rate	yes	yes	yes	no congestion
ATM	Variable Bit Rate	guaranteed rate	yes	yes	yes	no congestion
ATM	Available Bit Rate	guaranteed minimum	no	yes	no	yes
ATM	Unspecified Bit Rate	none	no	yes	no	no
Internet	Intserv (RFC 1633)	yes	yes	yes	yes	
Internet	Diffserve (RFC 2475)	possible	possibly	possibly	possibly	

Reflections on best-effort service:

- simplicity of mechanism has allowed Internet to be widely deployed adopted
- sufficient provisioning of bandwidth allows performance of real-time applications (e.g., interactive voice, video) to be “good enough” for “most of the time”
- replicated, application-layer distributed services (datacenters, content distribution networks) connecting close to clients’ networks, allow services to be provided from multiple locations
- congestion control of “elastic” services helps

It’s hard to argue with success of best-effort service model

Day 19 Pt. 2: Addresses

CSEE 4119
Computer Networks
Ethan Katz-Bassett

COLUMBIA UNIVERSITY
IN THE CITY OF NEW YORK

Chapter 4: outline

4.1 Overview of Network layer
- data plane
- control plane

4.2 What's inside a router

4.3 IP: Internet Protocol
- datagram format
- fragmentation
- IPv4 addressing
- network address translation
- IPv6

4.4 Generalized Forward and SDN
- match
- action
- OpenFlow examples of match-plus-action in action

Chapter 4: outline

4.1 Overview of Network layer
- data plane
- control plane

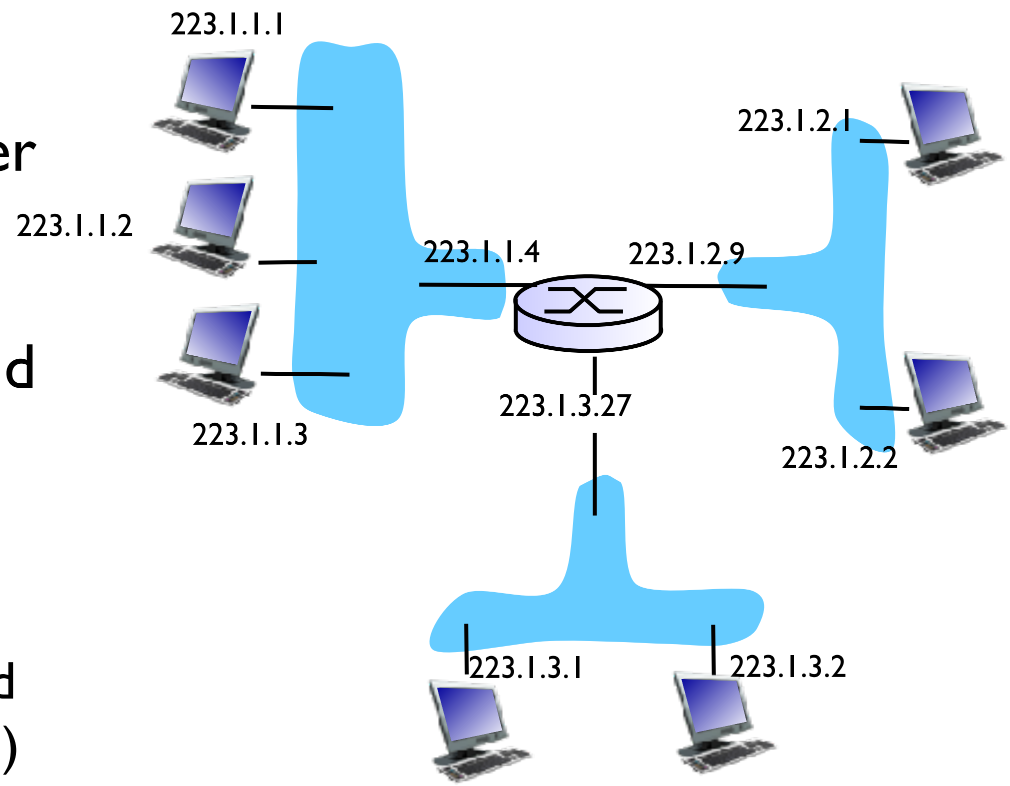
4.2 What's inside a router

4.3 IP: Internet Protocol
- datagram format
- fragmentation
- IPv4 addressing
- network address translation
- IPv6

4.4 Generalized Forward and SDN
- match
- action
- OpenFlow examples of match-plus-action in action

IP addressing: introduction

- *IP address:* 32-bit identifier for host, router *interface*
- *interface:* connection between host/router and physical link
  - router's typically have multiple interfaces
  - host typically has one or two interfaces (e.g., wired Ethernet, wireless 802.11)
- *IP addresses associated with each interface*



$223.1.1.1 = \underbrace{11011111}_{223}\ \underbrace{00000001}_{1}\ \underbrace{00000001}_{1}\ \underbrace{00000001}_{1}$

Classful Routing
(Internet Classic—changed in 1993)

- take a 32-bit IP address, get 4 classes

High Order Bits	Format	Class
0	7 bits of net, 24 bits of host	a
10	14 bits of net, 16 bits of host	b
110	21 bits of net, 8 bits of host	c
111	escape to extended addressing mode	

- A: 128 networks @ 16M hosts each
- B: 16k networks @ 64k hosts each
- C: 2M networks @ 256 hosts each

And Subnets (new in 1985)

- Not just network and host, but another level of division, the *subnet*
  - network: 128.9.0.0 --- class B, 64k hosts
  - subnet: 128.9.240.0, mask 255.255.240.0 --- part of it, 4k hosts

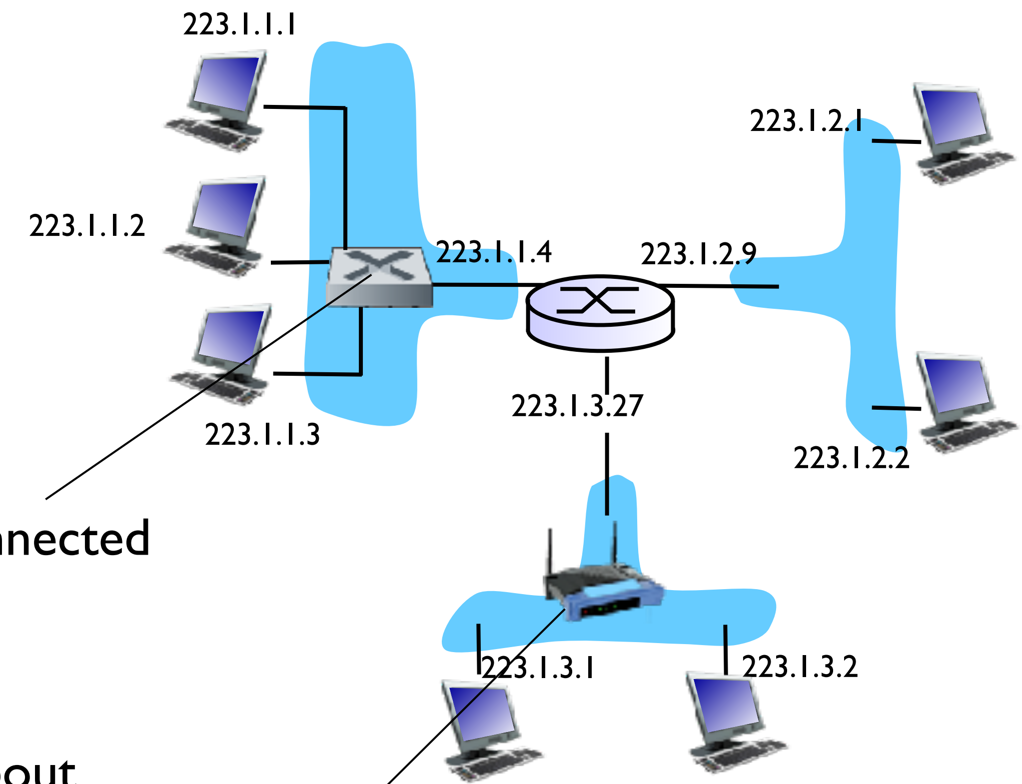
network 128.9.	host

network 128.9.	.240. subnet	host

IP addressing: introduction

Q: how are interfaces actually connected?

A: we'll learn about that in chapter 6, 7.



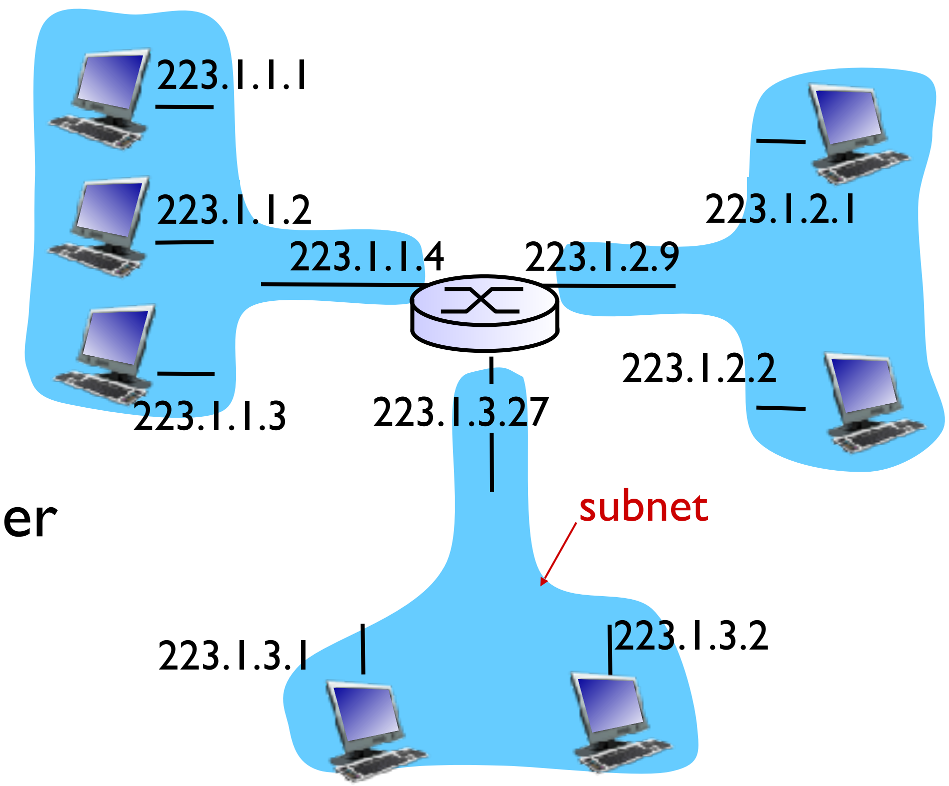
A: wired Ethernet interfaces connected by Ethernet switches

For now: don't need to worry about how one interface is connected to another (with no intervening router)

A: wireless WiFi interfaces connected by WiFi base station

Subnets

- IP address:
  - subnet part - high order bits
  - host part - low order bits
- *what's a subnet ?*
  - device interfaces with same subnet part of IP address
  - can physically reach each other *without intervening router*

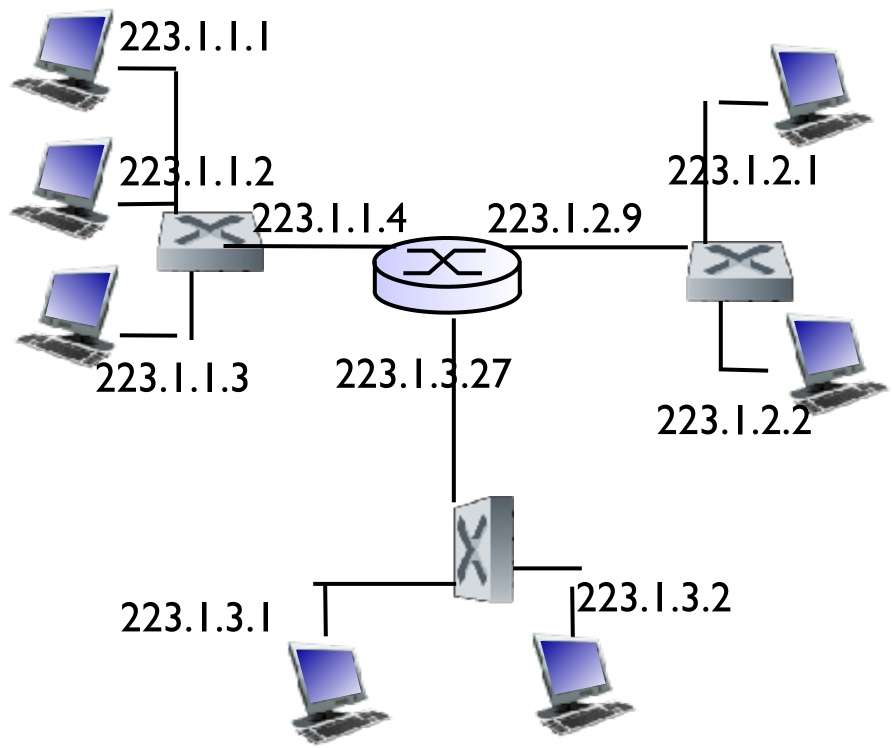


network consisting of 3 subnets

Subnets

recipe

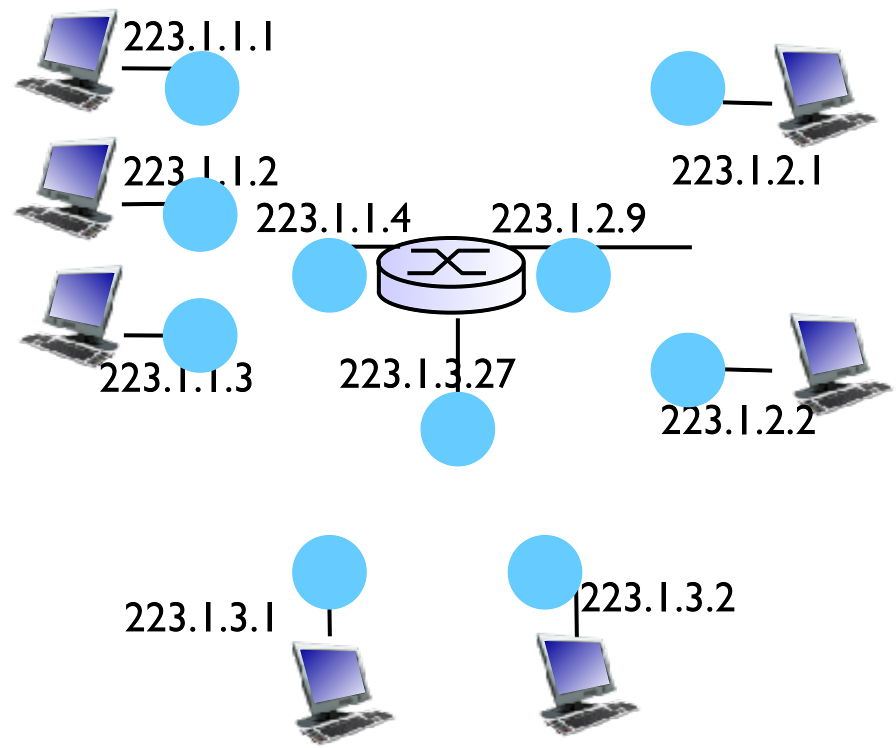
- to determine the subnets, detach each interface from its host or router, creating islands of isolated networks
- isolated network terminates at each interface
- each isolated network is called a *subnet*



Subnets

recipe

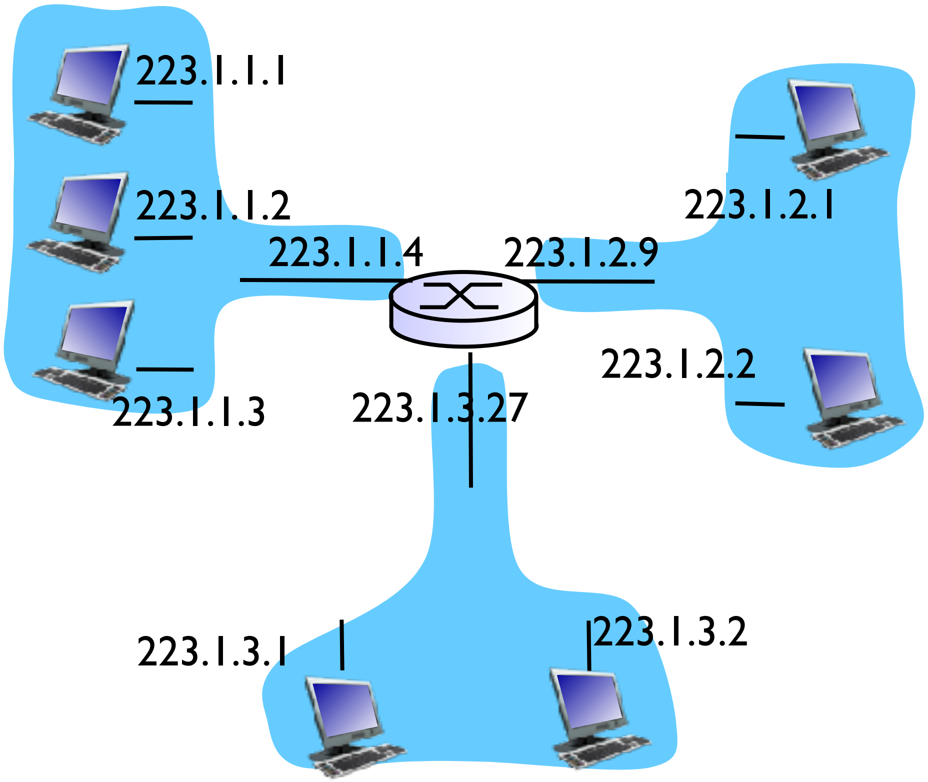
- to determine the subnets, detach each interface from its host or router, creating islands of isolated networks
- isolated network terminates at each interface
- each isolated network is called a *subnet*



Subnets

recipe

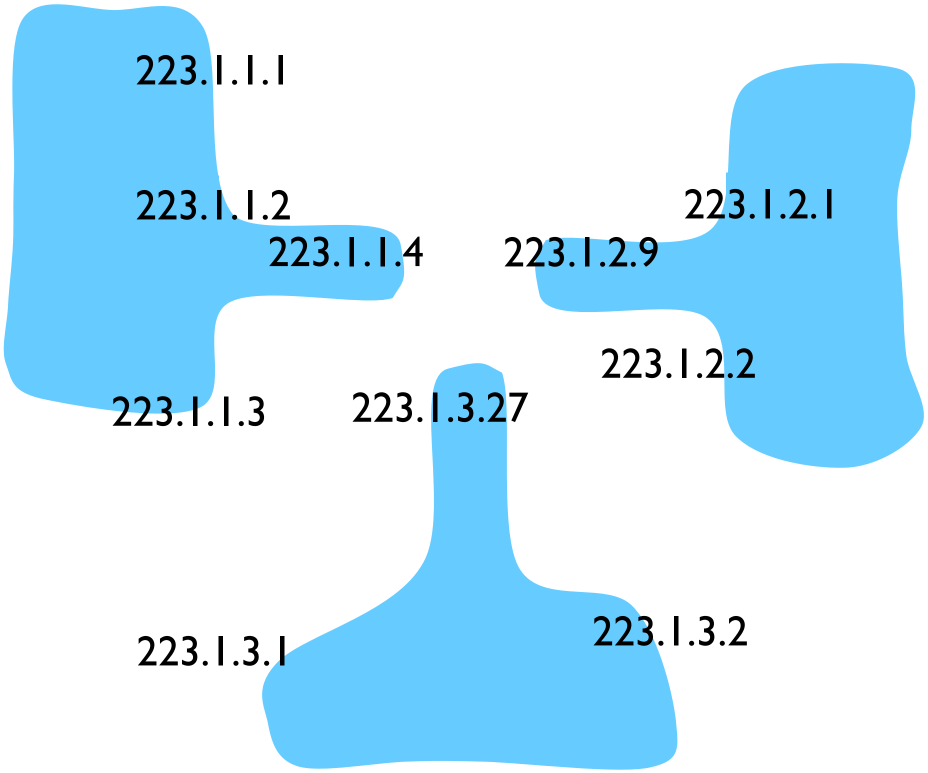
- to determine the subnets, detach each interface from its host or router, creating islands of isolated networks
- isolated network terminates at each interface
- each isolated network is called a *subnet*



Subnets

recipe

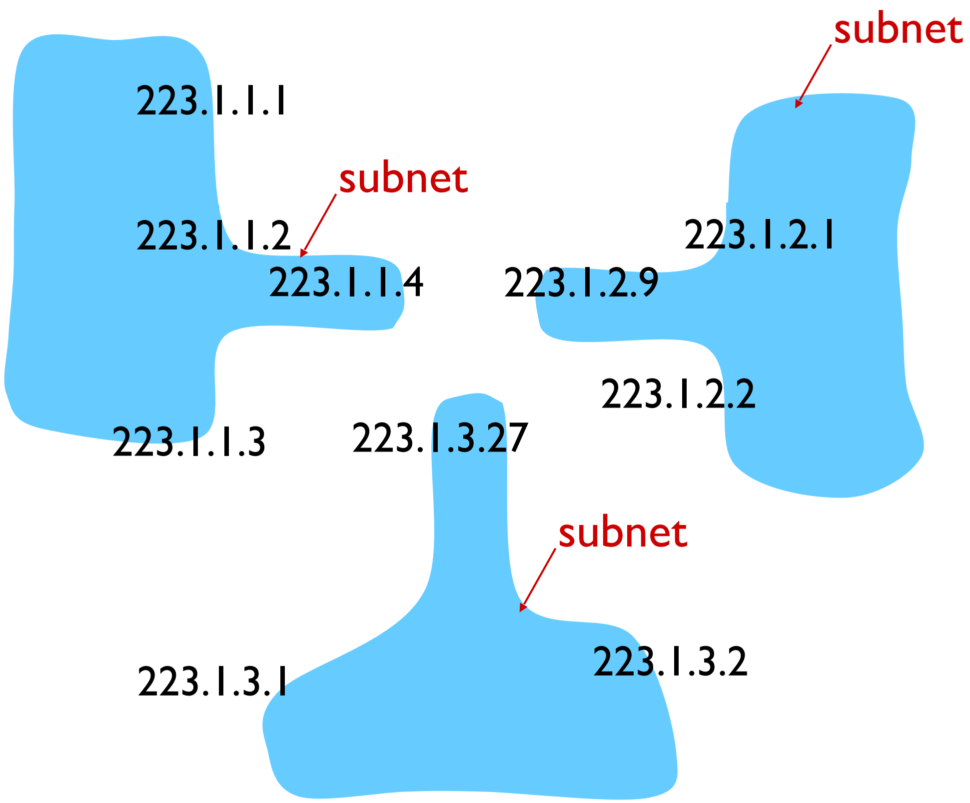
- to determine the subnets, detach each interface from its host or router, creating islands of isolated networks
- isolated network terminates at each interface
- each isolated network is called a *subnet*



Subnets

recipe

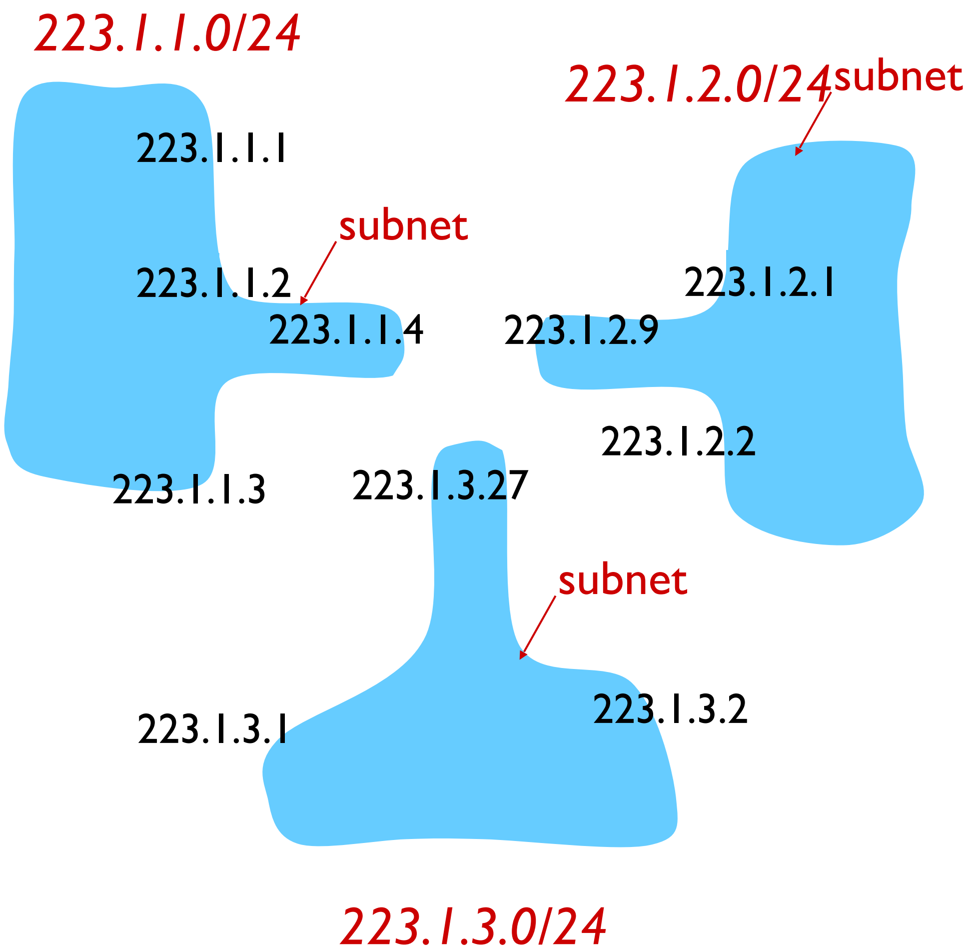
- to determine the subnets, detach each interface from its host or router, creating islands of isolated networks
- isolated network terminates at each interface
- each isolated network is called a *subnet*



Subnets

recipe

- to determine the subnets, detach each interface from its host or router, creating islands of isolated networks
- isolated network terminates at each interface
- each isolated network is called a *subnet*

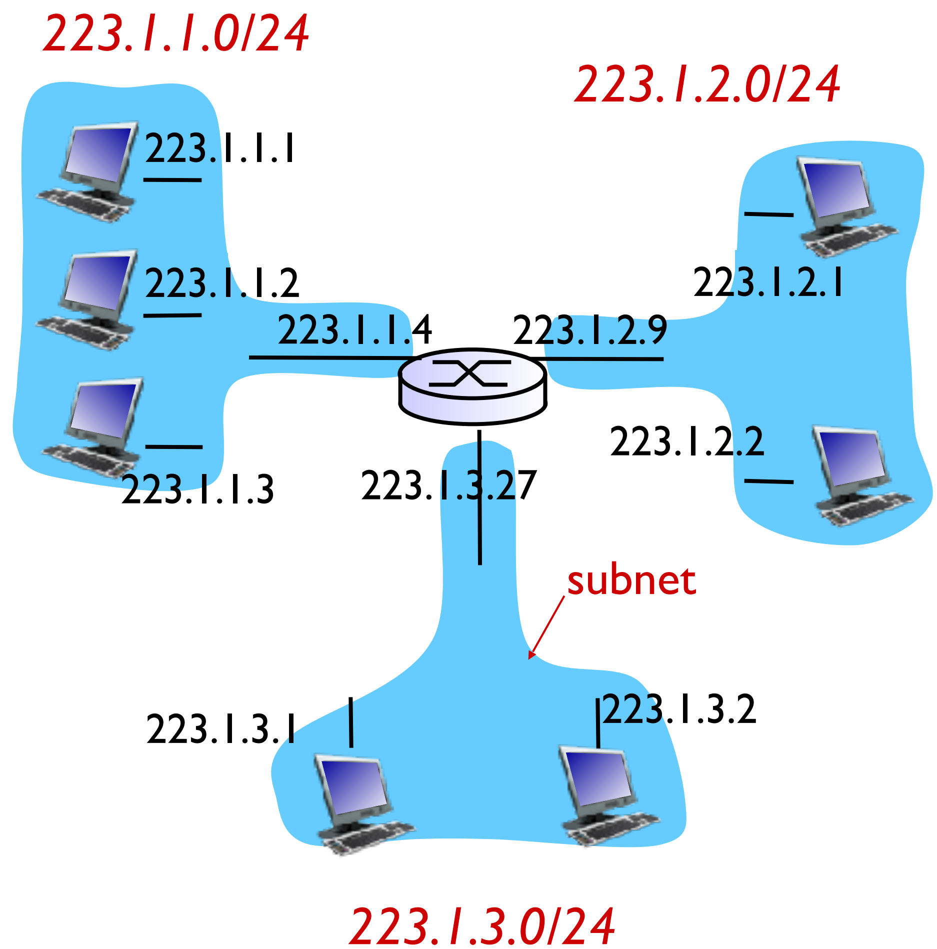


subnet mask: /24

Subnets

recipe

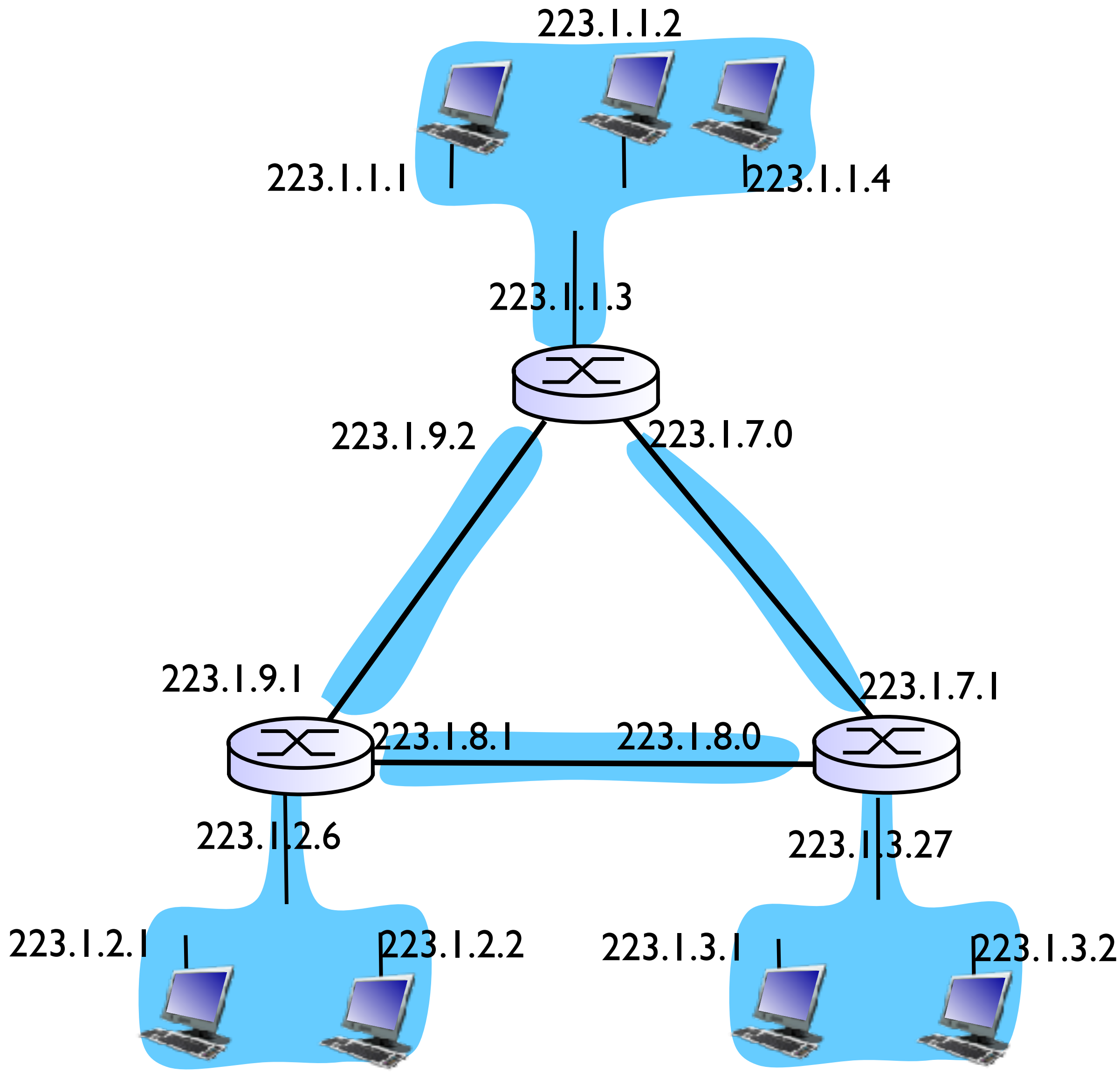
- to determine the subnets, detach each interface from its host or router, creating islands of isolated networks
- isolated network terminates at each interface
- each isolated network is called a *subnet*



subnet mask: /24

Subnets

how many?



Classful Routing

(Internet Classic—changed in 1993)

- take a 32-bit IP address, get 4 classes

High Order Bits	Format	Class
0	7 bits of net, 24 bits of host	a
10	14 bits of net, 16 bits of host	b
110	21 bits of net, 8 bits of host	c
111	escape to extended addressing mode	

- A: 128 networks @ 16M hosts each
- B: 16k networks @ 64k hosts each
- C: 2M networks @ 256 hosts each

Classful Addressing Trade-offs

pros:

-

cons:

-

Classful Addressing Trade-offs

pros:

- easy to identify what the network is just by looking at the address
- Simple to build fast routers
- Can support needs of different size networks

cons:

- Can waste address space: big gaps in size from A to B to C. Not flexible, uneven allocation
- 2M class C-> large routing tables, lots of memory
- Too few mid-sized networks

Problems (early 90's)

- But fixed classes are a poor match for the growing Internet:
  - many groups needed >256 but <<64k hosts (unfortunately, more than 16k of them)
  - AND 128+16k+2M ~= 2M networks >> router routing table capacities (~200k)

Uploading course materials to sites such as CourseHero, Chegg or Github is academic misconduct at Columbia (see pg 10 of Columbia guide).

Day 19: Network Layer Overview and Addresses

CSEE 4119
Computer Networks
Ethan Katz-Bassett

COLUMBIA UNIVERSITY
IN THE CITY OF NEW YORK

Recap: Classful Routing

(Internet Classic—changed in 1993)

- take a 32-bit IP address, get 4 classes

High Order Bits	Format	Class
0	7 bits of net, 24 bits of host	a
10	14 bits of net, 16 bits of host	b
110	21 bits of net, 8 bits of host	c
111	escape to extended addressing mode	

- A: 128 networks @ 16M hosts each
- B: 16k networks @ 64k hosts each
- C: 2M networks @ 256 hosts each

Recap: Problems (early 90's)

- But fixed classes are a poor match for the growing Internet:
  - many groups needed >256 but <<64k hosts (unfortunately, more than 16k of them)
  - AND 128+16k+2M ~= 2M networks >> router routing table capacities (~200k)

Solutions

- Short-term: Classless Internet Domain Routing, CIDR
  - make better use of existing space
  - divide addresses on *any* bit boundary
- Long-term: IPv6
  - increase address space (to 128-bits)

IP addressing: CIDR

CIDR: Classless InterDomain Routing

- subnet portion of address of arbitrary length
- address format: a.b.c.d/x, where x is # bits in subnet portion of address

subnet part ⟵⟶ host part

11001000 00010111 00010000 00000000

200.23.16.0/23

Day 17: Address Exhaustion, NATs, IPv6, and Middleboxes

CSEE 4119
Computer Networks
Ethan Katz-Bassett

COLUMBIA UNIVERSITY
IN THE CITY OF NEW YORK

IP addresses: how to get one? (1)

Q: how does an ISP get block of addresses?

A: IANA: Internet Assigned Numbers Authority

– Oversees global address allocation, via Regional Internet Registries
  - African Network Information Center (AFRINIC)
  - American Registry for Internet Numbers (ARIN)
  - Asia-Pacific Network Information Centre (APNIC)
  - Latin America and Caribbean Network Information Centre (LACNIC)
  - Réseaux IP Européens Network Coordination Centre (RIPE NCC)

– Also autonomous system number allocation, DNS root zones, etc

– History:
  - pre 1998: Jon Postel and Joyce Reynolds ran IANA at USC/ISI
  - 1998: Transfers to Internet Corp. for Assigned Names & Numbers (ICANN)
  - Sept 2016: IANA to privatize global Internet community, ending contract with US NTIA. Has more to do with DNS (especially administering root zone file) rather than address allocation.

IPv4 Exhaustion

- Jan 2011: IANA allocated last 2 unreserved /8s to APNIC,| then allocated 1 reserved /8 to each RIR
- All 5 Regional Internet Registries have exhausted their blocks (except those reserved for IPv6 transition)
  - April 2011: Asia-Pacific
  - June 2014: Latin America and the Caribbean
  - September 2015: North America
  - April 2017: Africa
  - November 2019: Europe, Middle East, and Central Asia

"*Who the hell knew how much address space we needed?*"
Vint Cerf (reflecting on decision to make IPv4 address 32 bits long)

Chapter 4: outline

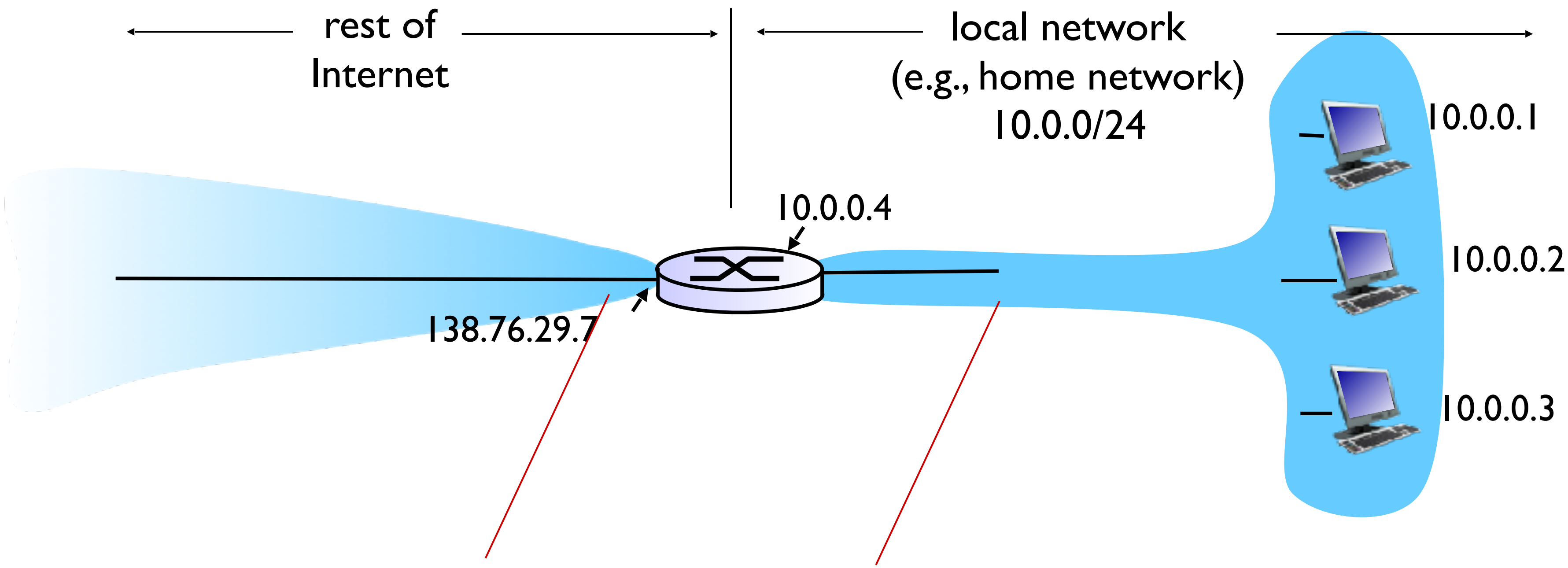
4.1 Overview of Network layer
- data plane
- control plane

4.2 What's inside a router

4.3 IP: Internet Protocol
- datagram format
- fragmentation
- IPv4 addressing
- network address translation
- IPv6

4.4 Generalized Forward and SDN
- match
- action
- OpenFlow examples of match-plus-action in action

NAT: network address translation



all datagrams *leaving* local network have *same* single source NAT IP address: 138.76.29.7,different source port numbers

datagrams with source or destination in this network have 10.0.0/24 address for source, destination (as usual)

NAT: network address translation

motivation: local network uses just one IP address as far as outside world is concerned:

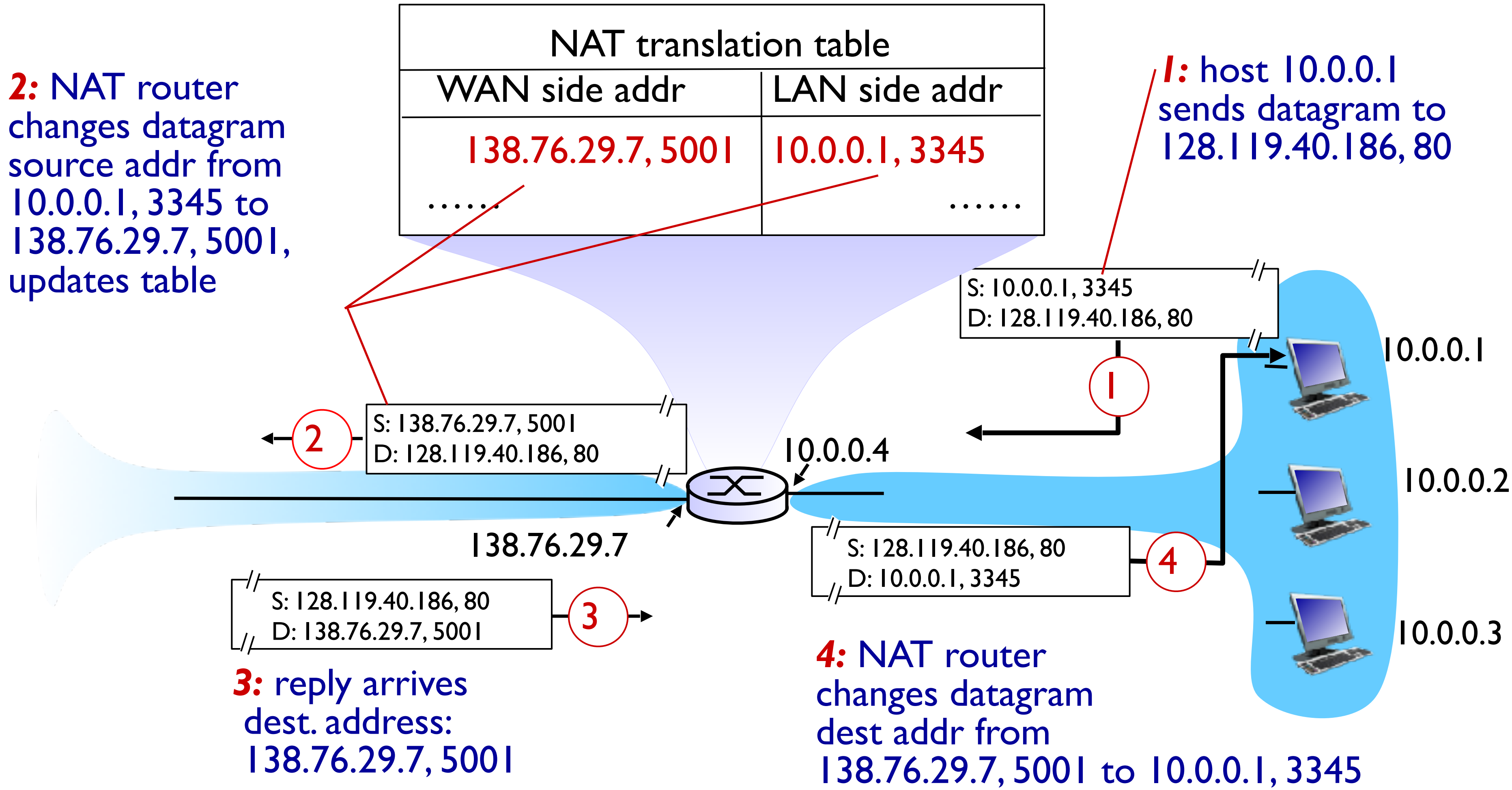
- range of addresses not needed from ISP: just one IP address for all devices
- helps sidestep IPv4 exhaustion
- can change addresses and/or devices in local network without notifying outside world
- can change ISP without changing addresses of devices in local network
- security: devices inside local net not explicitly addressable, visible by outside world (a security plus)

NAT: network address translation

implementation: NAT router must:

- *outgoing datagrams: replace* (source IP address, port #) of every outgoing datagram to (NAT IP address, new port #)
  . . . remote clients/servers will respond using (NAT IP address, new port #) as destination addr
- *remember (in NAT translation table)* every (source IP address, port #) to (NAT IP address, new port #) translation pair
- *incoming datagrams: replace* (NAT IP address, new port #) in dest fields of every incoming datagram with corresponding (source IP address, port #) stored in NAT table

NAT: network address translation



* Check out the online interactive exercises for more examples: http://gaia.cs.umass.edu/kurose_ross/interactive/

NAT Implications to the Stack

- to the Internet architecture?
  –
- to transport-layer protocols (TCP, UDP)?
  –
- to applications?
  –
- to users?
  –
- general concerns
  –

NAT Implications to the Stack

- to the Internet architecture?
  - Breaks global connectivity: not everyone can address everyone
- to transport-layer protocols (TCP, UDP)?
  - NAT designed for TCP: TCP setup sets up NAT mapping
  - UDP: make implicit connection setup
  - new protocols? out of luck… slows progress: problem when transport layers leak into higher layers, like with port numbers
- to applications?
  - Apps have to be flexible about port #s
- to users?
  - Easy to add hosts to home network
  - Harder to run applications, P2P

NAT: network address translation

- 16-bit port-number field:
  - 60,000 simultaneous connections with a single LAN-side address!
- often use ***private*** reserved addresses behind NAT
  - RFC1918: 10.0.0.0/8, 172.16.0.0/12, 192.168.0.0/16
  - only have meaning within local address, not globally
- NAT is controversial:
  - routers "should" only process up to layer 3
  - address shortage "should" be solved by IPv6
  - violates end-to-end argument
    - NAT possibility must be taken into account by app designers, e.g., P2P applications
  - NAT traversal: what if client wants to connect to server behind NAT?
- but NAT is here to stay:
  - extensively used in home/institutional nets, 4G/5G networks

Chapter 4: outline

4.1 Overview of Network layer
- data plane
- control plane

4.2 What's inside a router

4.3 IP: Internet Protocol
- datagram format
- fragmentation
- IPv4 addressing
- network address translation
- IPv6

4.4 Generalized Forward and SDN
- match
- action
- OpenFlow examples of match-plus-action in action

IPv6: motivation

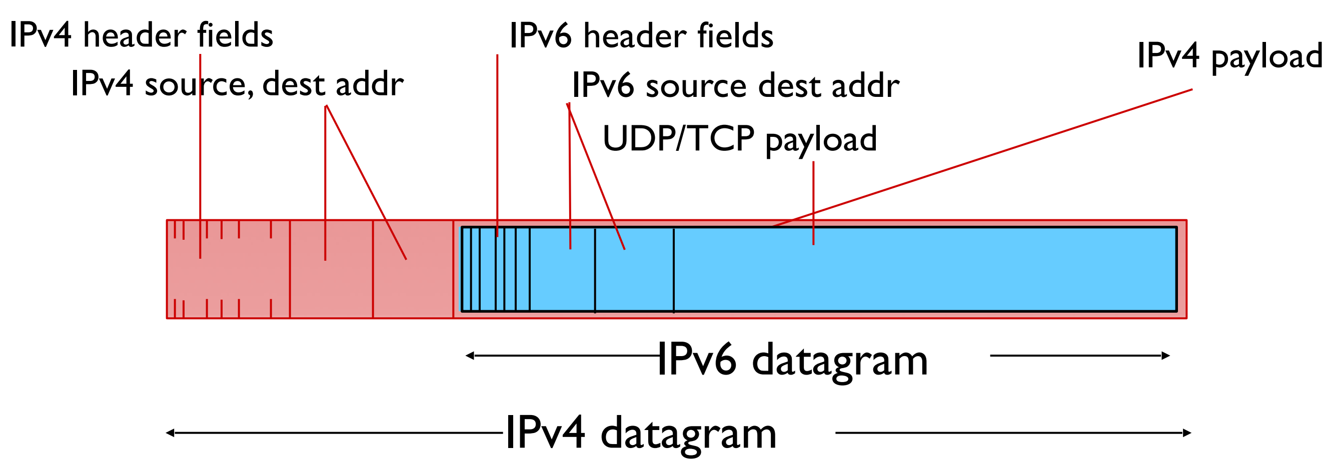
- *initial motivation:* 32-bit address space soon to be completely allocated.
- additional motivation:
  - header format helps speed processing/forwarding
  - header changes to facilitate QoS
  - enable different network-layer treatment of "flows"

IPv6 datagram format:

- fixed-length 40 byte header
- no fragmentation allowed

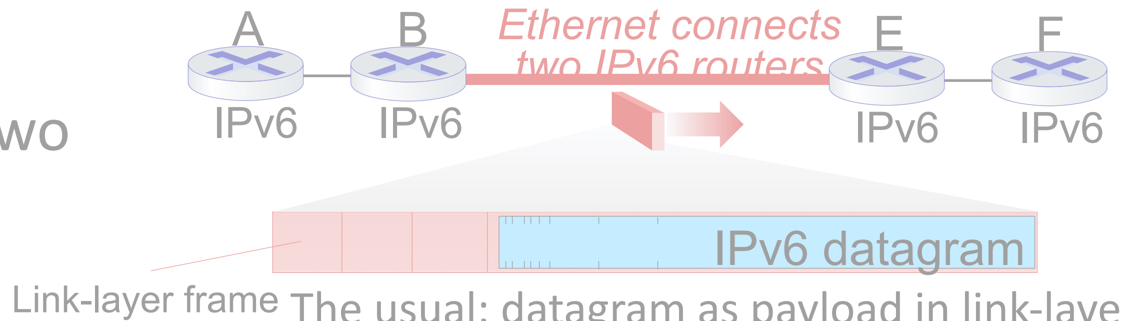
Transition from IPv4 to IPv6

- not all routers can be upgraded simultaneously
  - no "flag days"
  - how will network operate with mixed IPv4 and IPv6 routers?
- *tunneling:* IPv6 datagram carried as *payload* in IPv4 datagram among IPv4 routers



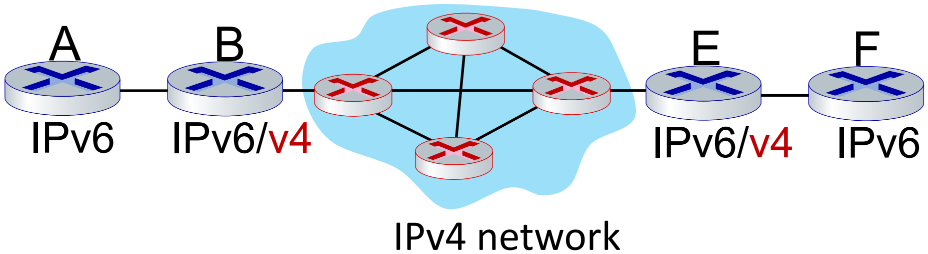
Tunneling and encapsulation

Ethernet connecting two IPv6 routers:



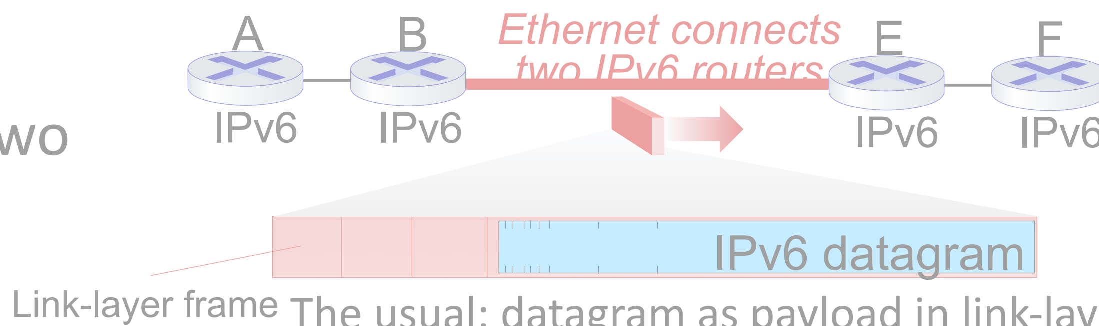
The usual: datagram as payload in link-layer frame

IPv4 network connecting two IPv6 routers



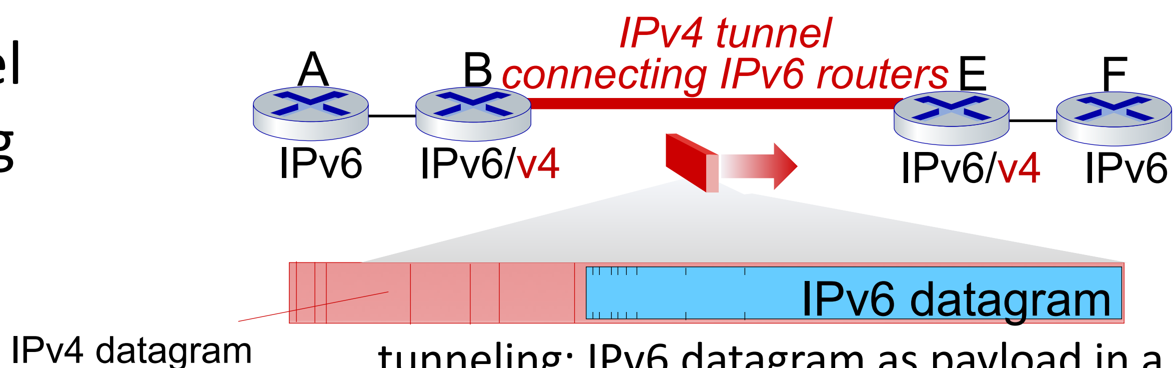
Tunneling and encapsulation

Ethernet connecting two IPv6 routers:



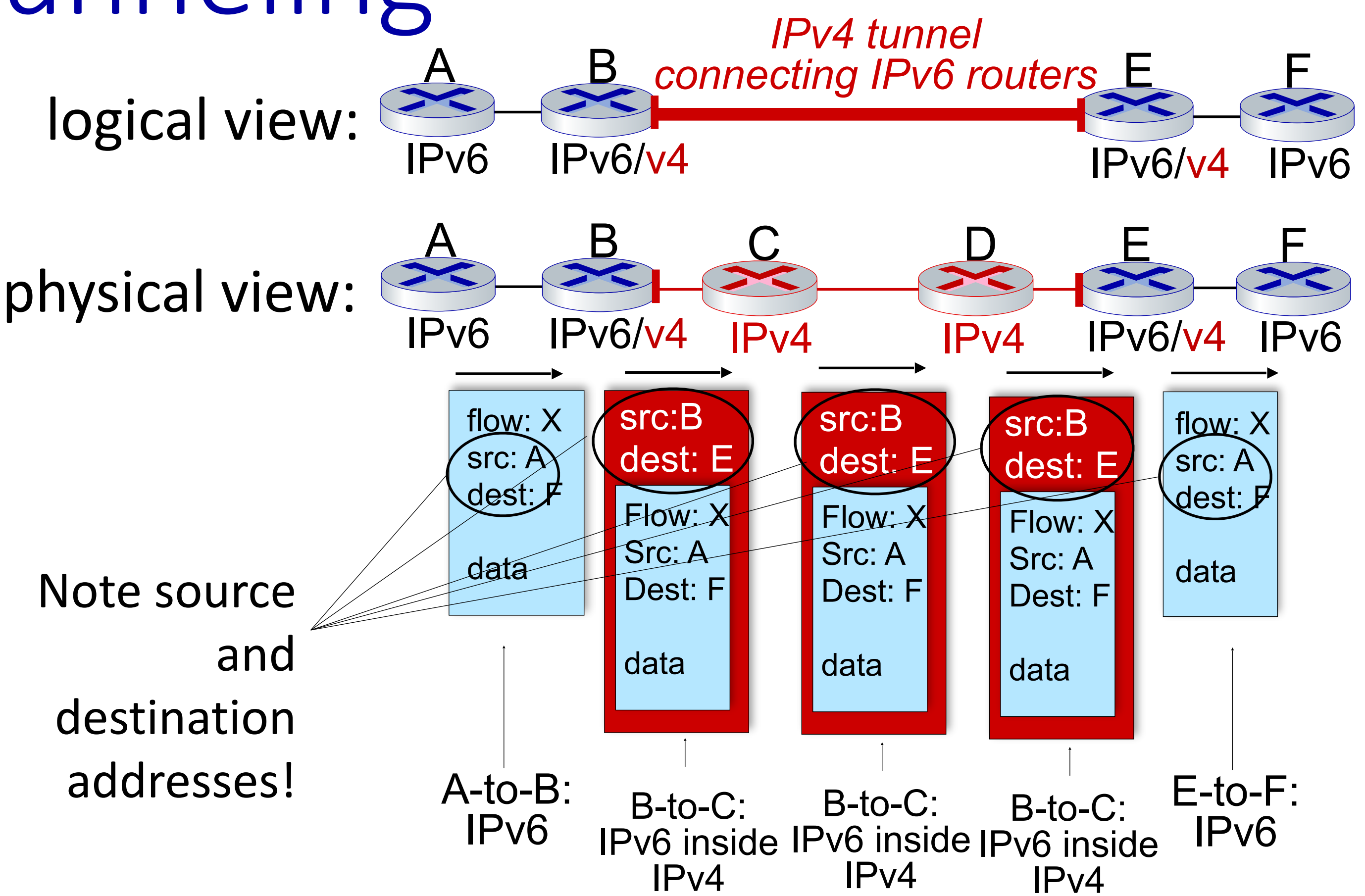
The usual: datagram as payload in link-layer frame

IPv4 tunnel connecting two IPv6 routers



tunneling: IPv6 datagram as payload in a IPv4 datagram

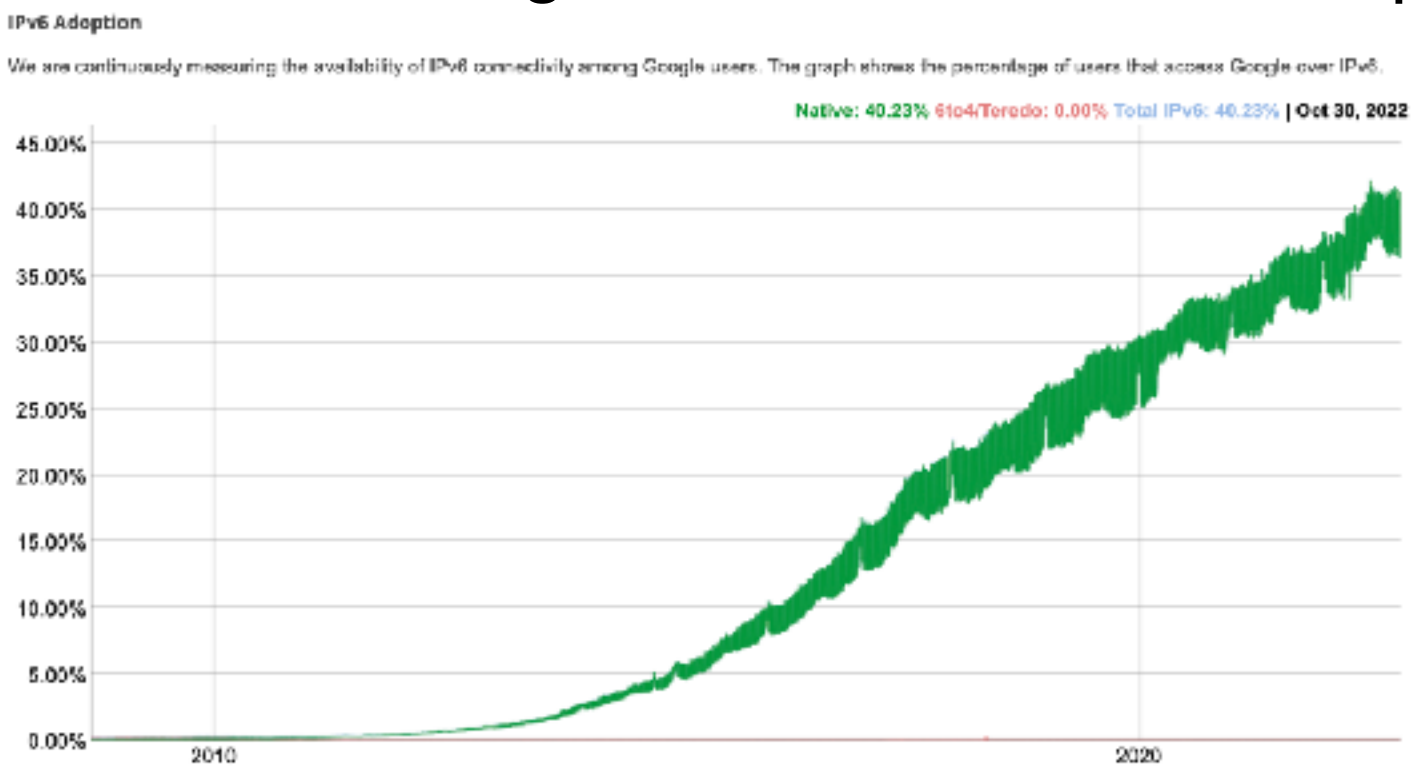
Tunneling



logical view:
A
B
IPv4 tunnel
connecting IPv6 routers
E
F
IPv6
IPv6/v4
IPv6/v4
IPv6
physical view:
A
B
C
D
E
F
IPv6
IPv6/v4
IPv4
IPv4
IPv6/v4
IPv6
flow: X
src: A
dest: F
data
src:B
dest: E
Flow: X
Src: A
Dest: F
data
src:B
dest: E
Flow: X
Src: A
Dest: F
data
src:B
dest: E
Flow: X
Src: A
Dest: F
data
flow: X
src: A
dest: F
data
Note source and destination addresses!
A-to-B: IPv6
B-to-C: IPv6 inside IPv4
B-to-C: IPv6 inside IPv4
B-to-C: IPv6 inside IPv4
E-to-F: IPv6

IPv6: adoption

- Google: 40% of clients access services via IPv6
- NIST: 1/3 of all US government domains IPv6 capable



https://www.google.com/intl/en/ipv6/statistics.html

IPv6: adoption

- Google: 40% of clients access services via IPv6
- NIST: 1/3 of all US government domains IPv6 capable

- *Long (long!) time for deployment, use*
  - 24 years and counting!
  - think of application-level changes in last 20 years: WWW, social media, streaming media, gaming, telepresence, …
  - *Why?*

Chapter 4: outline

4.1 Overview of Network layer
- data plane
- control plane

4.2 What's inside a router

4.3 IP: Internet Protocol
- datagram format
- fragmentation
- IPv4 addressing
- network address translation
- IPv6

~~4.4 Generalized Forward and SDN~~
- ~~match -> action~~
- ~~OpenFlow examples of match-plus-action in action~~

Deferring generalized forwarding (SDN data plane) to talk about later with SDN control plane

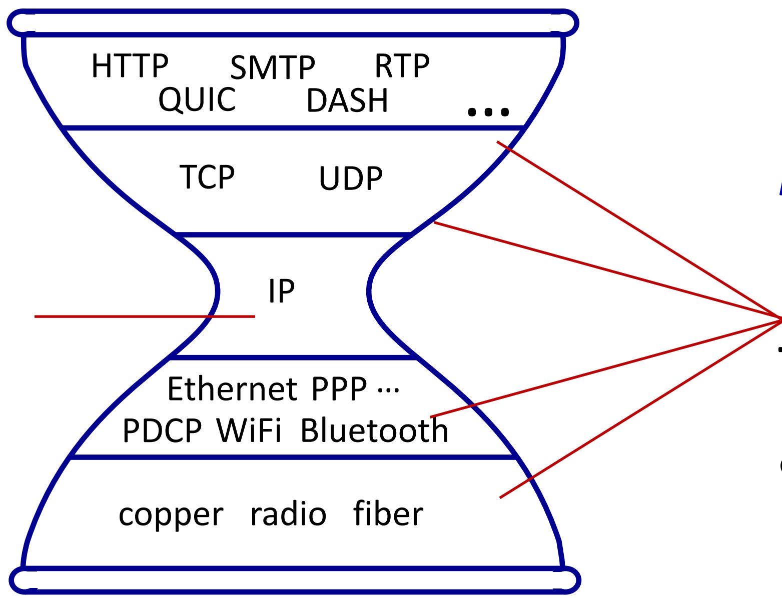
Network layer: Internet Architecture

- Network layer: overview
- What's inside a router
- IP: the Internet Protocol
- ~~Generalized Forwarding~~
- Net Neutrality
- Middleboxes
  - middlebox functions
  - evolution, architectural principles of the Internet

The IP hourglass

Internet's "thin waist":

- *one* network layer protocol: IP
- *must* be implemented by every (billions) of Internet-connected devices



many protocols in physical, link, transport, and application layers

Network Neutrality

What is network neutrality?

- *technical:* how an ISP should share/allocation its resources
  - packet scheduling, buffer management are the *mechanisms*
- *social, economic* principles
  - protecting free speech
  - encouraging innovation, competition
- enforced *legal* rules and policies

Different countries have different "takes" on network neutrality

Network Neutrality

2015 US FCC *Order on Protecting and Promoting an Open Internet:*

three "clear, bright line" rules:

- no blocking … "shall not block lawful content, applications, services, or non-harmful devices, subject to reasonable network management."
- no throttling … "shall not impair or degrade lawful Internet traffic on the basis of Internet content, application, or service, or use of a non-harmful device, subject to reasonable network management."
- no paid prioritization. … "shall not engage in paid prioritization"

ISP: telecommunications or information service?

Is an ISP a "telecommunications service" or an "information service" provider?

- the answer *really* matters from a regulatory standpoint!

US Communications Act of 1934 and Telecommunications Act of 1996:

- *Title II:* imposes "common carrier duties" on *telecommunications services*: reasonable rates, non-discrimination, and *requires regulation* (by FCC)
- *Title I:* applies to *information services:*
  - no common carrier duties (*not regulated*, would give FCC little control)
  - but grants FCC authority "… as may be necessary in the execution of its functions"
- (Tele)communications were not updated to account for ISPs, so FCC can designate they are Title I or Title II

Network Neutrality in the US

2015 US FCC *Order on Protecting and Promoting an Open Internet:*

three "clear, bright line" rules:

- no blocking … "shall not block lawful content, applications, services, or non-harmful devices, subject to reasonable network management."
- no throttling … "shall not impair or degrade lawful Internet traffic on the basis of Internet content, application, or service, or use of a non-harmful device, subject to reasonable network management."
- no paid prioritization. … "shall not engage in paid prioritization"

Repealed in 2018. In 2021, Executive Order instructed FCC to restore the net neutrality rules that had been undone in 2018.

Network Neutrality at Columbia

Columbia plays an active role in net neutrality debate

- Tim Wu (law) popularized the concept and coined the term "net neutrality" as an extension to the concept of a common carrier
- Vishal Misra (CS):
  - Defines neutrality as "Internet is a platform where ISPs provide no competitive advantage to specific apps/services, either through pricing or QoS"
  - Research on net neutrality
  - Presented his views to Indian Parliament, shaping regulations
- Henning Schulzrinne (CS) was CTO at FCC 2011-2014
- For more info, check out the video of a panel I hosted in 2017, in advance of the FCC vote. The panel included Vishal and Henning.

Middleboxes

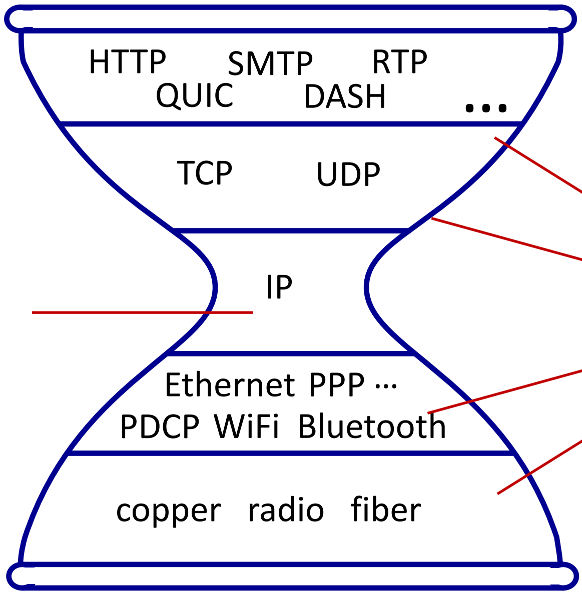
Middlebox (RFC 3234)

"any intermediary box performing functions apart from normal, standard functions of an IP router on the data path between a source host and destination host"

The IP hourglass

Internet's "thin waist":

- *one* network layer protocol: IP
- *must* be implemented by every (billions) of Internet-connected devices

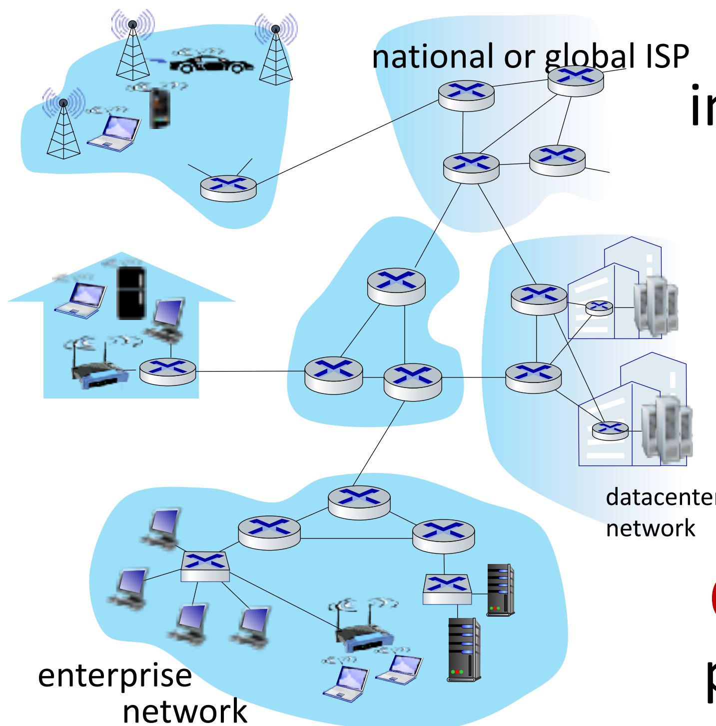


many protocols in physical, link, transport, and application layers

Middleboxes everywhere!

NAT: home, cellular, institutional

Application-specific: service providers, institutional, CDN



Firewalls, IDS: corporate, institutional, service providers, ISPs

Load balancers: corporate, service provider, data center, mobile nets

Caches: service provider, mobile, CDNs

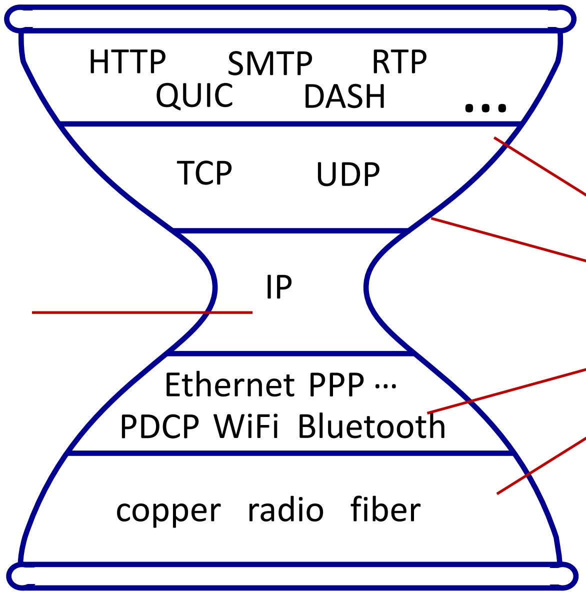
Middleboxes

- initially: proprietary (closed) hardware solutions
- move towards "whitebox" hardware implementing open API
  - move away from proprietary hardware solutions
  - programmable local actions via match+action
  - move towards innovation/differentiation in software
- SDN: (logically) centralized control and configuration management often in private/public cloud
- network functions virtualization (NFV): programmable services over white box networking, computation, storage

The IP hourglass

Internet's "thin waist":

- *one* network layer protocol: IP
- *must* be implemented by every (billions) of Internet-connected devices

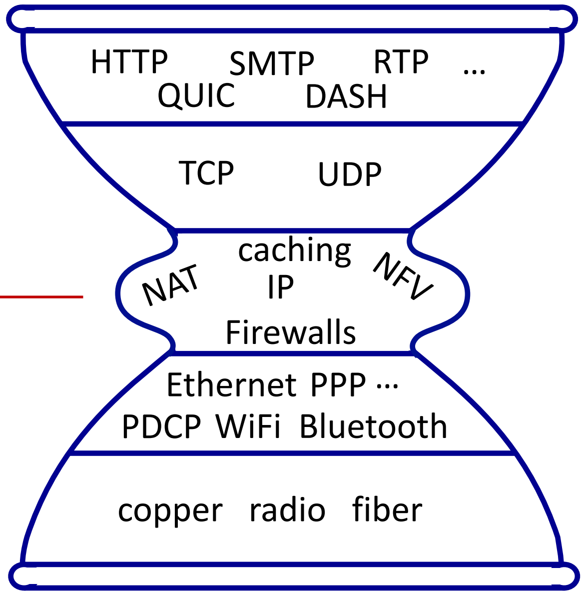


many protocols in physical, link, transport, and application layers

The IP hourglass, at middle age

Internet's middle age "love handles"?

- middleboxes, operating inside the network



Architectural Principles of the Internet

RFC 1958 - Architectural Principles of the Internet (1996)

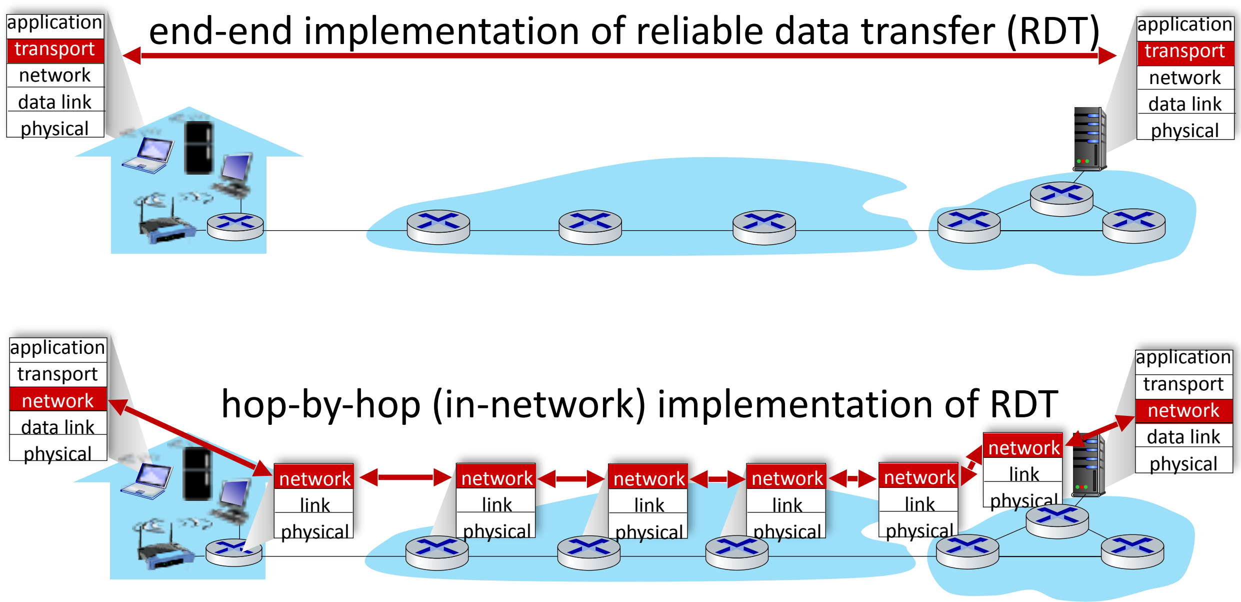
"Many members of the Internet community would argue that there is no architecture, but only a tradition, which was not written down for the first 25 years (or at least not by the IAB). However, in very general terms, the community believes that the goal is connectivity, the tool is the Internet Protocol, and the intelligence is end to end rather than hidden in the network."

Three cornerstone beliefs:

- simple connectivity
- IP protocol: that narrow waist
- intelligence, complexity at network edge

The end-to-end argument

- some functionality (e.g., reliable data transfer, congestion control) can be implemented in network *or/and* at network edge



The end-to-end argument

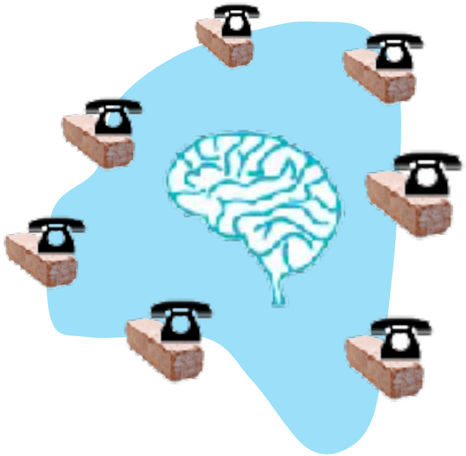
- some functionality (e.g., reliable data transfer, congestion control) can be implemented in network *or/and* at network edge

"The function in question can completely and correctly be implemented only with the knowledge and help of the application standing at the end points of the communication system." "Functions placed at low levels of a system may be redundant or of little value when compared with the cost of providing them at that low level." "(Sometimes an incomplete version of the function provided by the communication system may be useful as a performance enhancement.)"

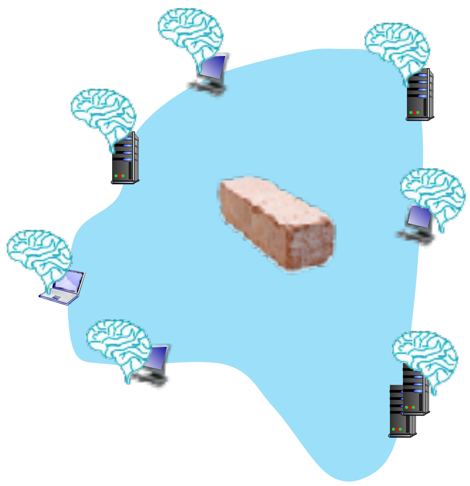
We call this line of reasoning against low-level function implementation the "***end-to-end argument***."

Saltzer, Reed, Clark 1981

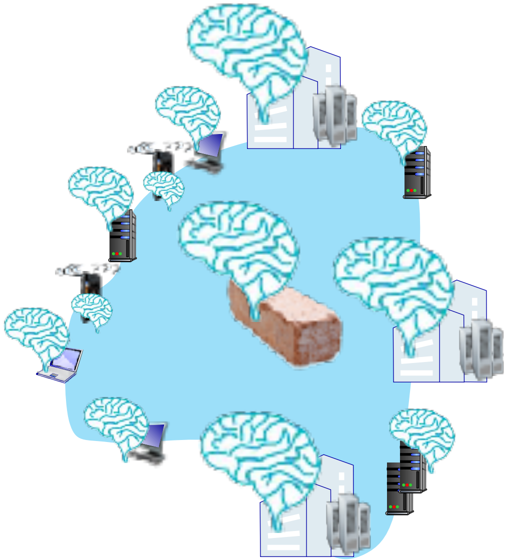
Where’s the intelligence?



20th century phone net:
- intelligence/computing at network switches



Internet (pre-2005)
- intelligence, computing at edge



Internet (post-2005)
- programmable network devices
- intelligence, computing, massive application-level infrastructure at edge

Chapter 4: *done!*

4.1 Overview of Network layer: data plane and control plane

4.2 What's inside a router

4.3 IP: Internet Protocol

- datagram format
- fragmentation
- IPv4 addressing
- NAT
- IPv6

- Internet architecture
  - Net neutrality
  - Middleboxes
  - End-to-end argument

~~4.4 Generalized Forward and SDN~~

- ~~match plus action~~
- ~~OpenFlow example~~

Deferring generalized forwarding (SDN data plane) to talk about later with SDN control plane

Question: how are forwarding tables (destination-based forwarding) or flow tables (generalized forwarding) computed?

Answer: by the control plane (next chapter)

Chapter 4: outline

4.1 Overview of Network layer
- data plane
- control plane

4.2 What's inside a router

4.3 IP: Internet Protocol
- datagram format
- fragmentation
- IPv4 addressing
- network address translation
- IPv6

4.4 Generalized Forward and SDN
- match
- action
- OpenFlow examples of match-plus-action in action

Chapter 4: outline

4.1 Overview of Network layer
- data plane
- control plane

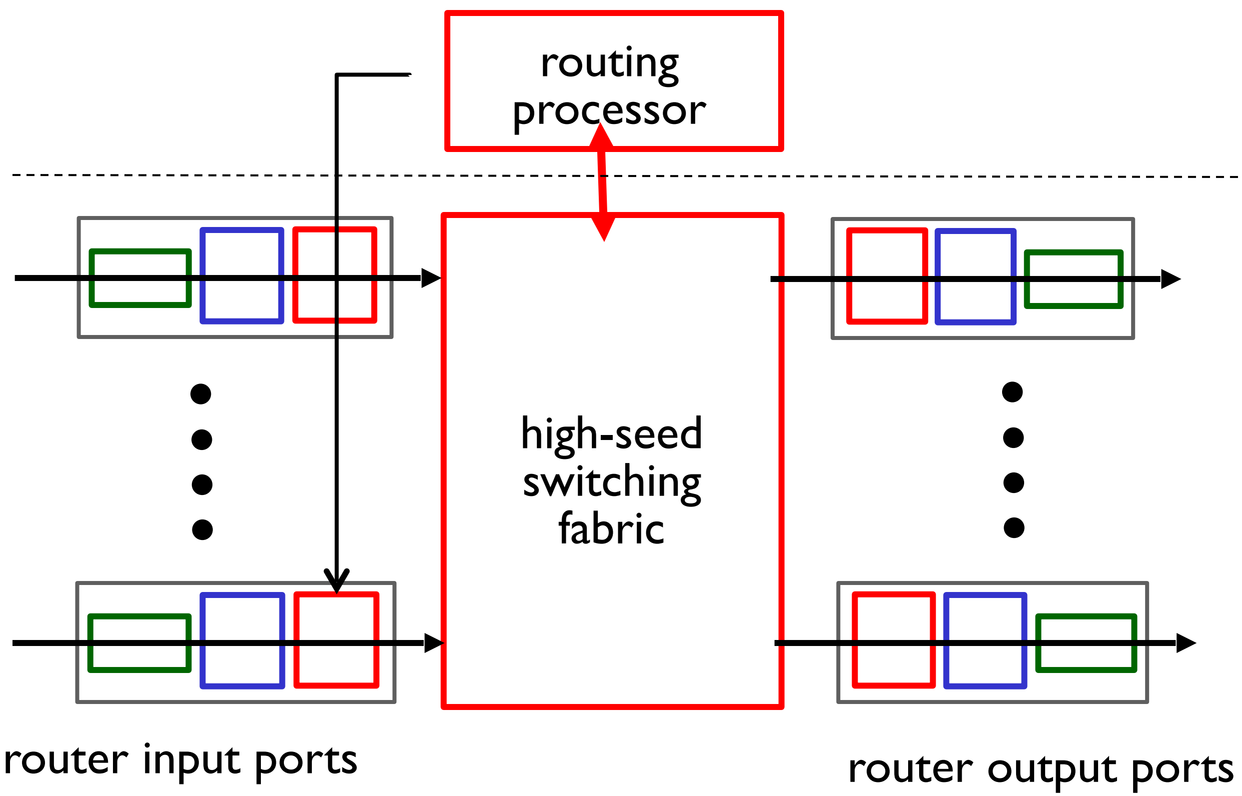
4.2 What's inside a router

4.3 IP: Internet Protocol
- datagram format
- fragmentation
- IPv4 addressing
- network address translation
- IPv6

4.4 Generalized Forward and SDN
- match
- action
- OpenFlow examples of match-plus-action in action

Router architecture overview

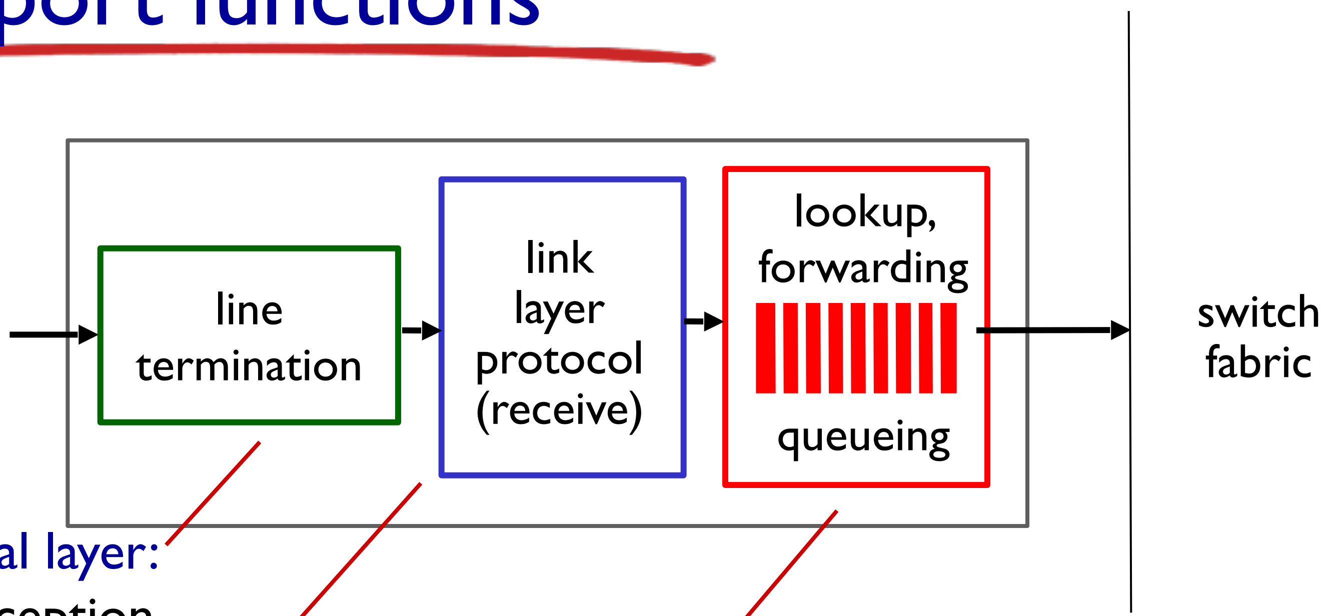
- high-level view of generic router architecture:



routing, management control plane (software) operates in millisecond time frame

forwarding data plane (hardware) operates in nanosecond timeframe

Input port functions



physical layer:
bit-level reception

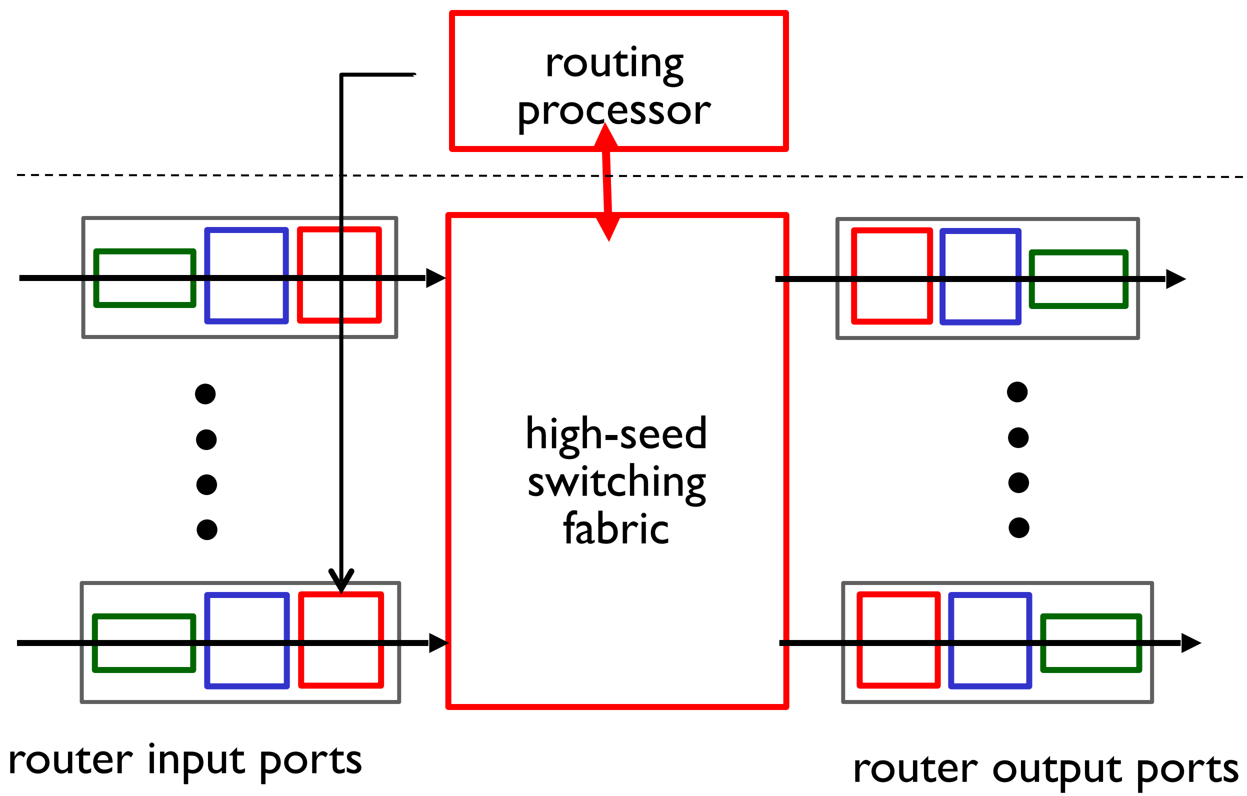
data link layer:
e.g., Ethernet
see chapter 6

decentralized switching:

- using header field values, lookup output port using forwarding table in input port memory *("match plus action")*
- goal: complete input port processing at 'line speed'
- queuing: if datagrams arrive faster than forwarding rate into switch fabric

Router architecture overview

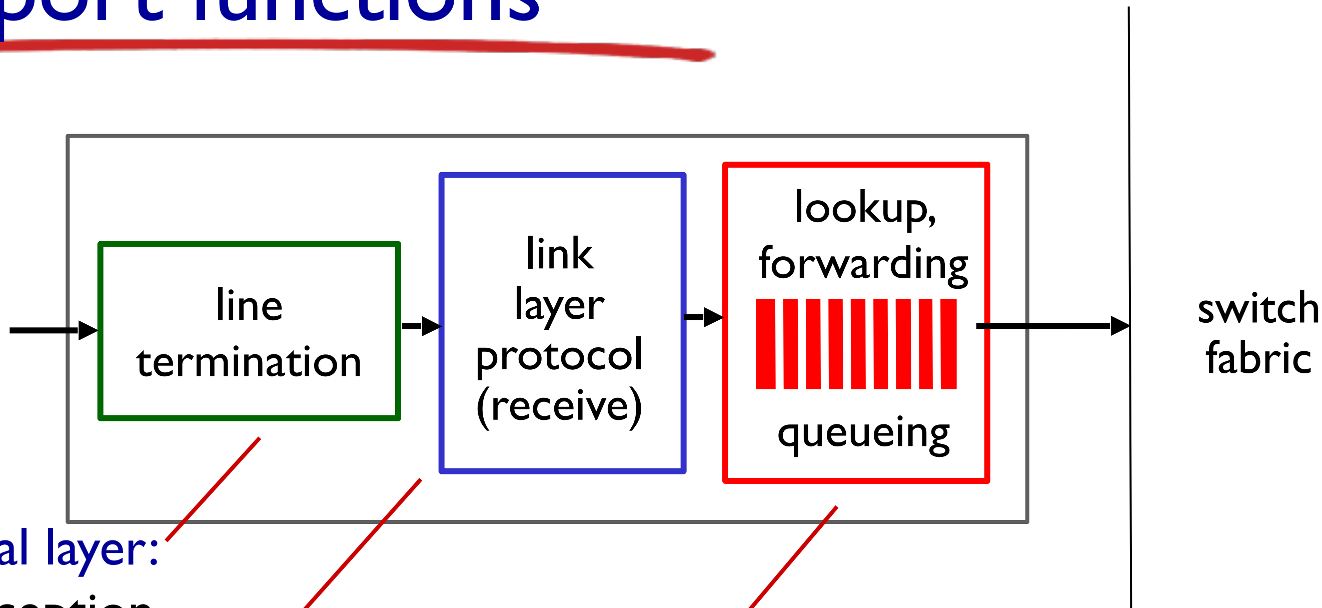
- high-level view of generic router architecture:



routing, management
control plane (software)
operates in millisecond
time frame

forwarding data plane
(hardware) operates in
nanosecond timeframe

Input port functions



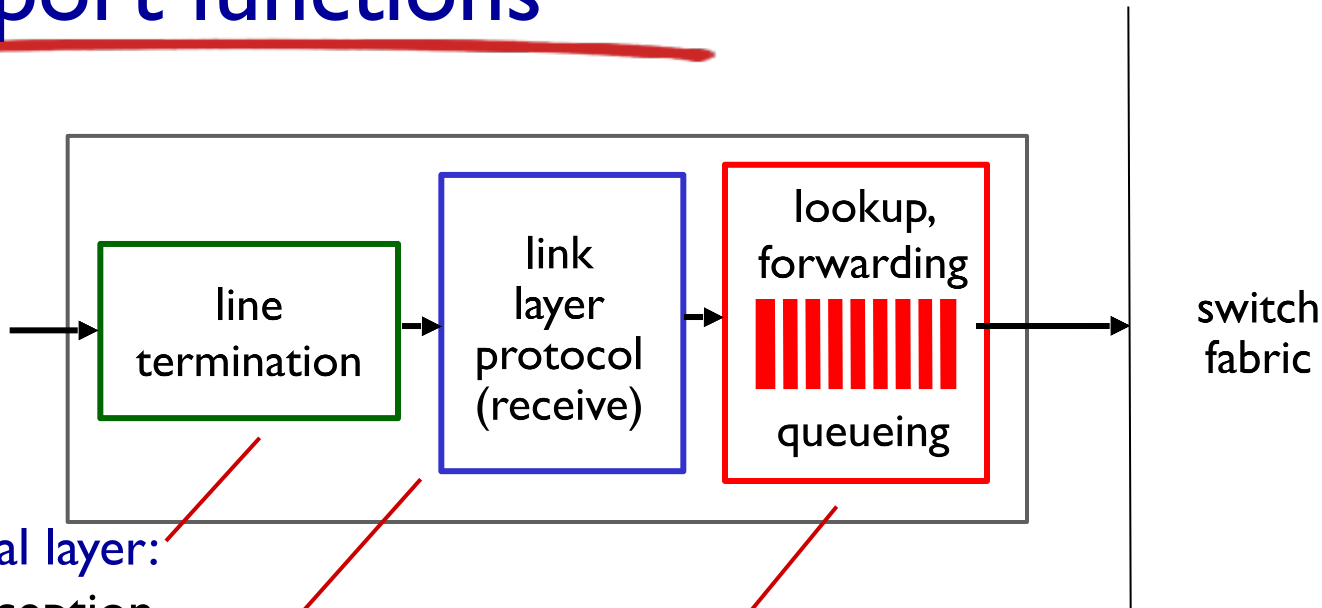
physical layer:
bit-level reception

data link layer:
e.g., Ethernet
see chapter 5

decentralized switching:

- using header field values, lookup output port using forwarding table in input port memory *("match plus action")*
- goal: complete input port processing at 'line speed'
- queuing: if datagrams arrive faster than forwarding rate into switch fabric

Input port functions



physical layer:
bit-level reception

data link layer:
e.g., Ethernet
see chapter 5

decentralized switching:

- using header field values, lookup output port using forwarding table in input port memory *("match plus action")*
- *destination-based forwarding:* forward based only on destination IP address (traditional)
- *generalized forwarding:* forward based on any set of header field values

Destination-based forwarding

forwarding table

Destination Address Range	Link Interface
11001000 00010111 00010000 00000000 through 11001000 00010111 00010111 11111111	0
11001000 00010111 00011000 00000000 through 11001000 00010111 00011000 11111111	1
11001000 00010111 00011001 00000000 through 11001000 00010111 00011111 11111111	2
otherwise	3

forwarding table

Destination Address Range	Link Interface
11001000 00010111 00010000 00000000 through **11001000 00010111 00010111 11111111**	0
11001000 00010111 00011000 00000000 through **11001000 00010111 00011000 11111111**	1
11001000 00010111 00011001 00000000 through **11001000 00010111 00011111 11111111**	2
otherwise	3

Destination Address Range	Link interface
11001000 00010111 00010*** *********	0
11001000 00010111 00011000 *********	1
11001000 00010111 00011*** *********	2
otherwise	3

DA: 11001000 00010111 00010110 10100001 matches 1 and 2

Destination Address Range	Link interface
11001000 00010111 00010*** *********	0
11001000 00010111 00011000 *********	1
11001000 00010111 00011*** *********	2
otherwise	3

Longest prefix matching

longest prefix matching

when looking for forwarding table entry for given destination address, use *longest* address prefix that matches destination address.

Destination Address Range	Link interface
11001000 00010111 00010*** *********	0
11001000 00010111 00011000 *********	1
11001000 00010111 00011*** *********	2
otherwise	3

examples:

DA: 11001000 00010111 00010110 10100001 which interface?

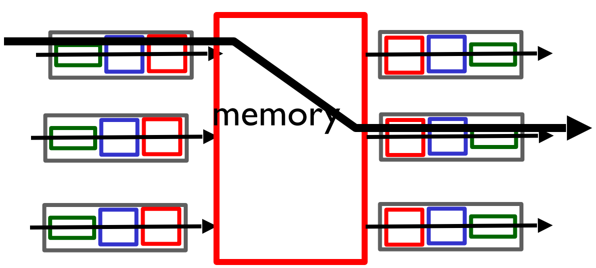
DA: 11001000 00010111 00011000 10101010 which interface?

Longest prefix matching

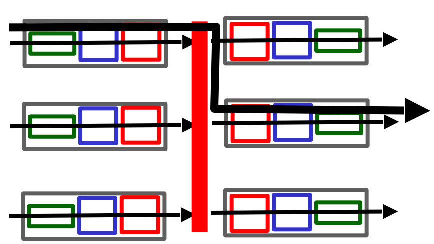
- we'll see *why* longest prefix matching is used shortly, when we study addressing
- longest prefix matching: often performed using ternary content addressable memories (TCAMs)
  - *content addressable:* present address to TCAM: retrieve address in one clock cycle, regardless of table size
  - Cisco Catalyst: up ~1M routing table entries in TCAM

Switching fabrics

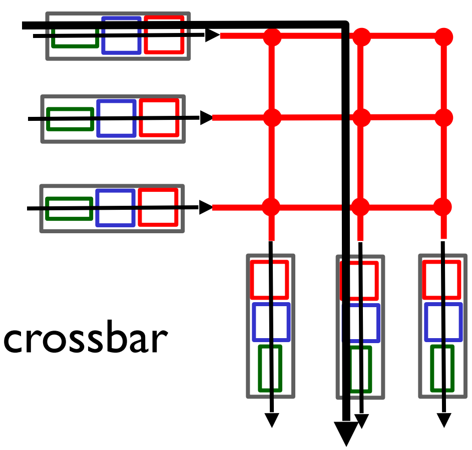
- transfer packet from input buffer to appropriate output buffer
- switching rate: rate at which packets can be transfer from inputs to outputs
  - often measured as multiple of input/output line rate
  - N inputs: switching rate N times line rate desirable
- three types of switching fabrics



memory



bus

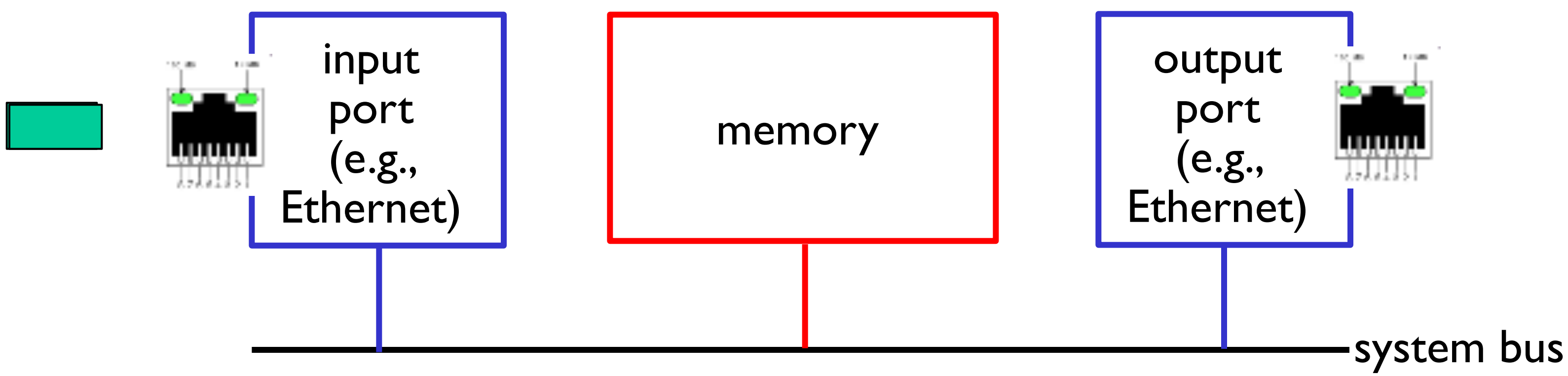


crossbar

Switching via memory

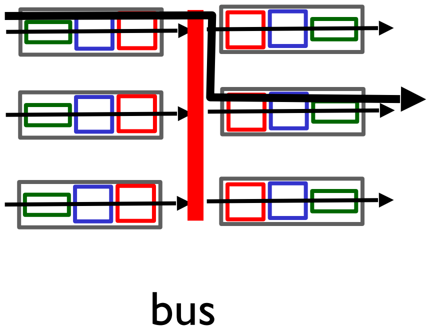
first generation routers:

- traditional computers with switching under direct control of CPU
- packet copied to system's memory
- speed limited by memory bandwidth (2 bus crossings per datagram)



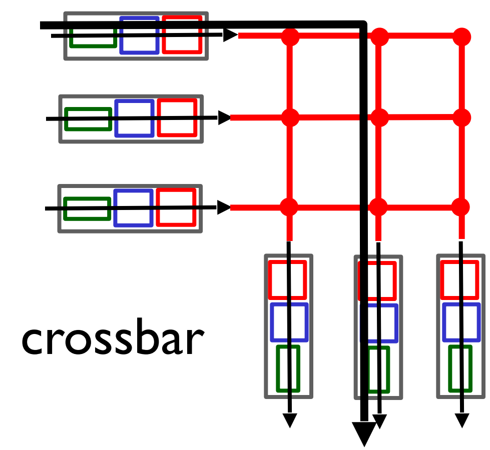
Switching via a bus

- datagram from input port memory
  to output port memory via a shared bus
- *bus contention:* switching speed limited by bus bandwidth
- 32 Gbps bus, Cisco 5600: sufficient speed for access and enterprise routers

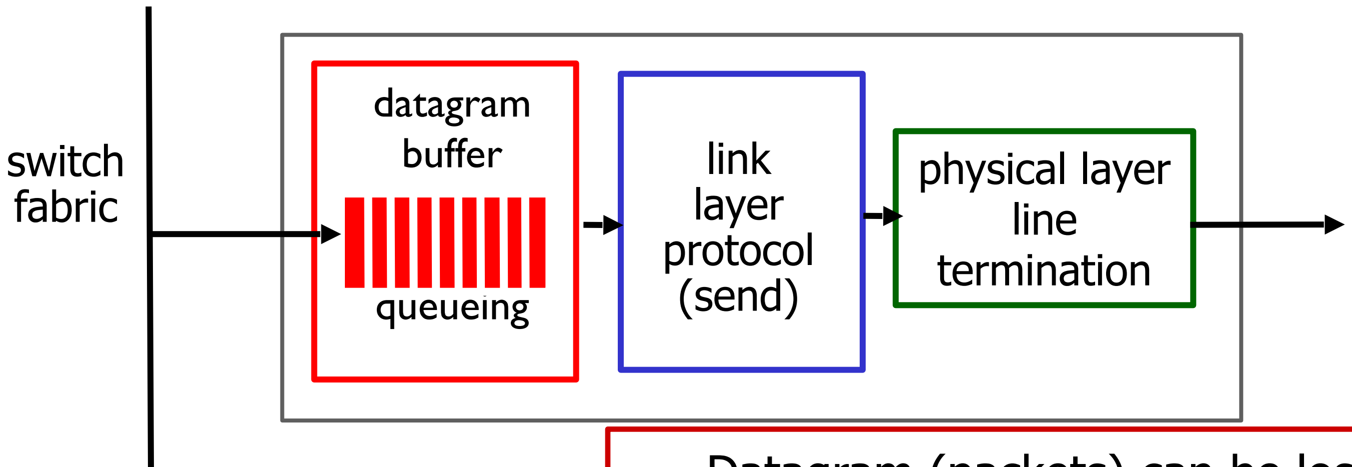


Switching via interconnection network

- overcome bus bandwidth limitations
- banyan networks, crossbar, other interconnection nets initially developed to connect processors in multiprocessor
- advanced design: fragmenting datagram into fixed length cells, switch cells through the fabric.
- Cisco 12000: switches 60 Gbps through the interconnection network



Output ports



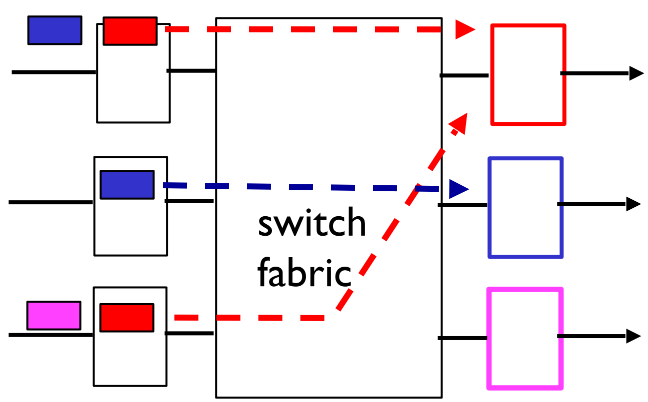
Datagram (packets) can be lost due to congestion, lack of buffers

- *buffering* required when datagrams arrive from fabric faster than the transmission rate
- *scheduling discipline* chooses among queued datagrams for transmission

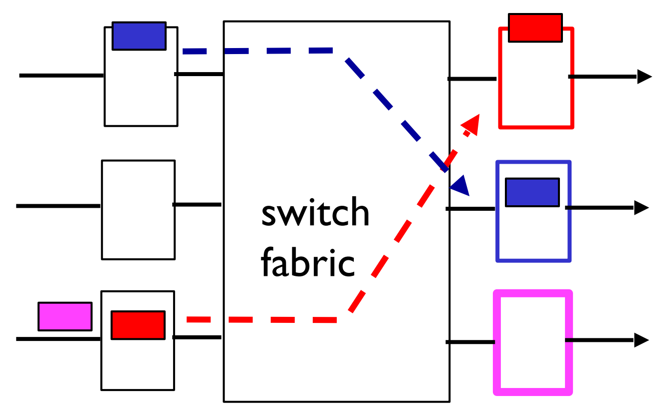
Priority scheduling – who gets best performance, network neutrality

Input port queuing

- fabric slower than input ports combined -> queueing may occur at input queues
  - *queueing delay and loss due to input buffer overflow!*
- Head-of-the-Line (HOL) blocking: queued datagram at front of queue prevents others in queue from moving forward

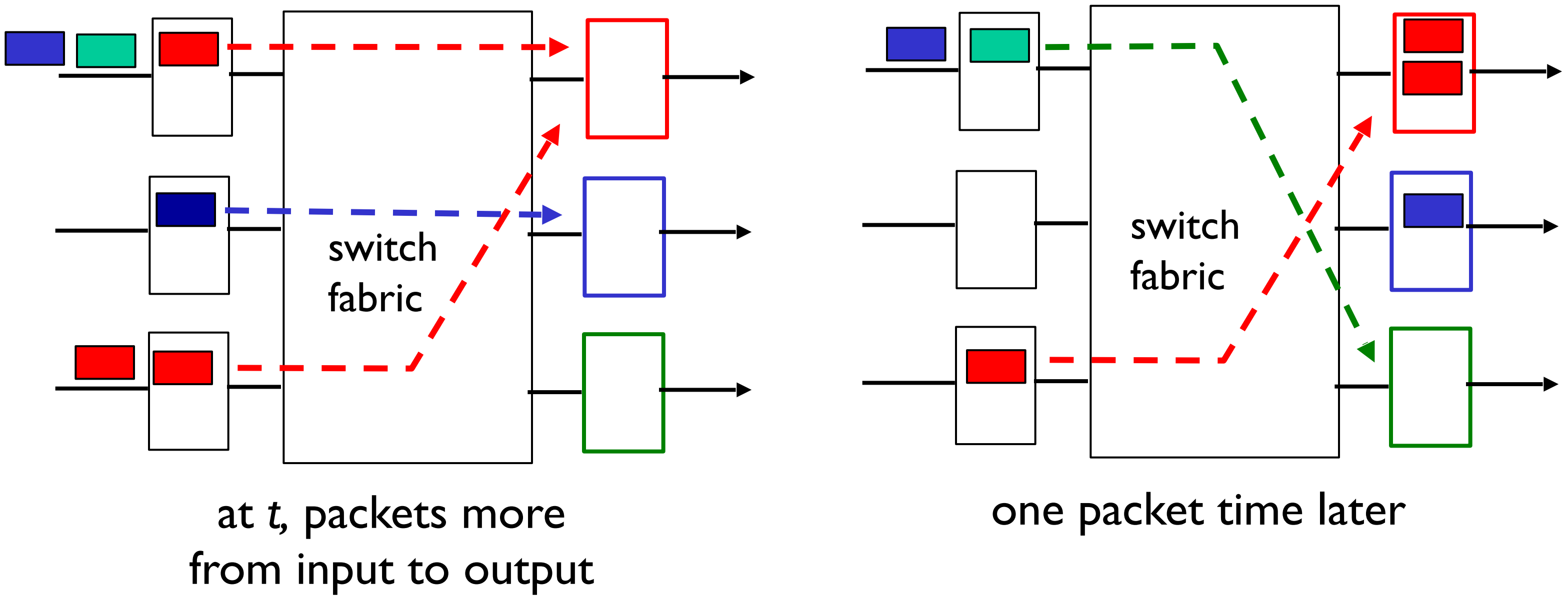


output port contention:
only one red datagram can be transferred.
lower red packet is blocked



one packet time later: magenta packet experiences Head-of-Line (HOL) blocking, even though magenta output is free

Output port queueing



- buffering when arrival rate via switch exceeds output line speed
- *queueing (delay) and loss due to output port buffer overflow!*

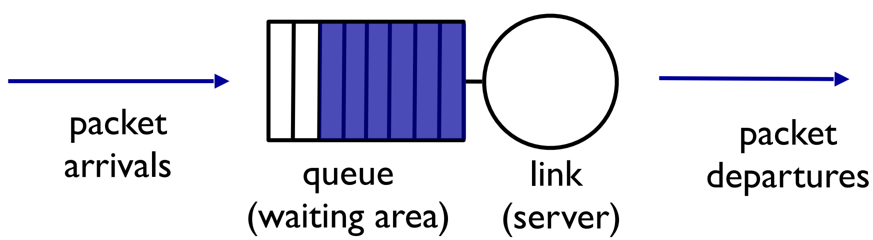
How much buffering?

- RFC 3439 rule of thumb: average buffering equal to "typical" RTT (say 250 msec) times link capacity C
  - e.g., C = 10 Gpbs link: 2.5 Gbit buffer
- recent recommendation: for *N* flows, buffering equal to

$$\frac{RTT \cdot C}{\sqrt{N}}$$

Scheduling mechanisms

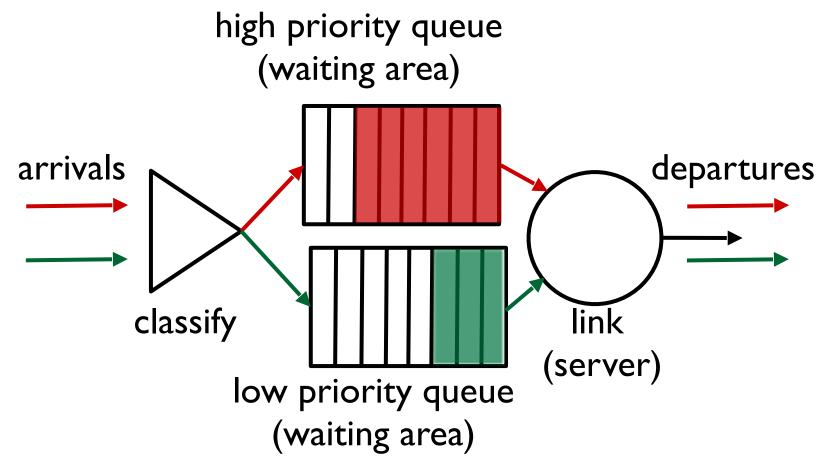
- *discard policy:* if packet arrives to full queue: which to discard?
  - *tail drop:* drop arriving packet
  - *priority:* drop/remove on priority basis
  - *random:* drop/remove randomly
- *scheduling:* choose next packet to send on link
  - *FIFO (first in first out) scheduling:* send in order of arrival to queue
    - real-world example?
  - *more on next bunch of slides*

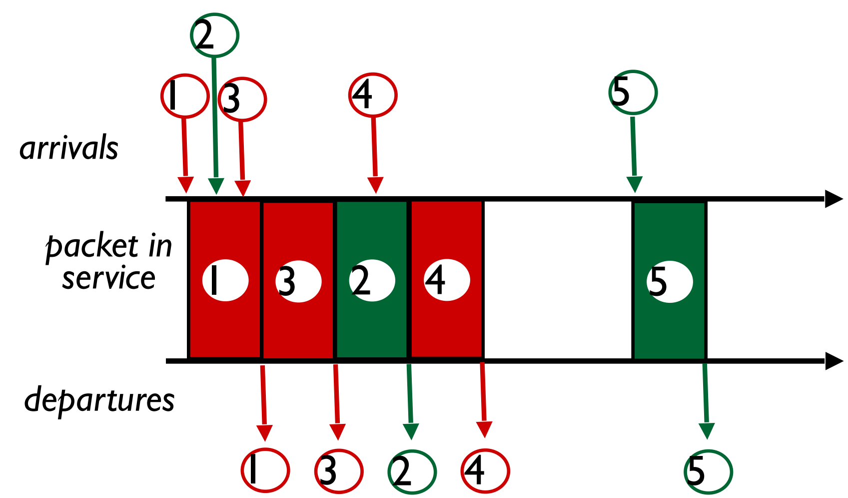


Scheduling policies: priority

priority scheduling: send highest priority queued packet

- multiple *classes*, with different priorities
  - class may depend on marking or other header info, e.g. IP source/dest, port numbers, etc.
  - real world example?

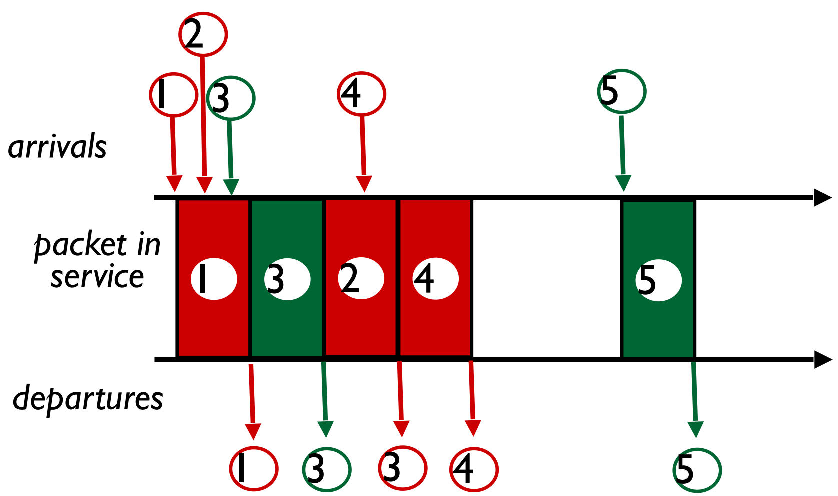




Scheduling policies: still more

Round Robin (RR) scheduling:

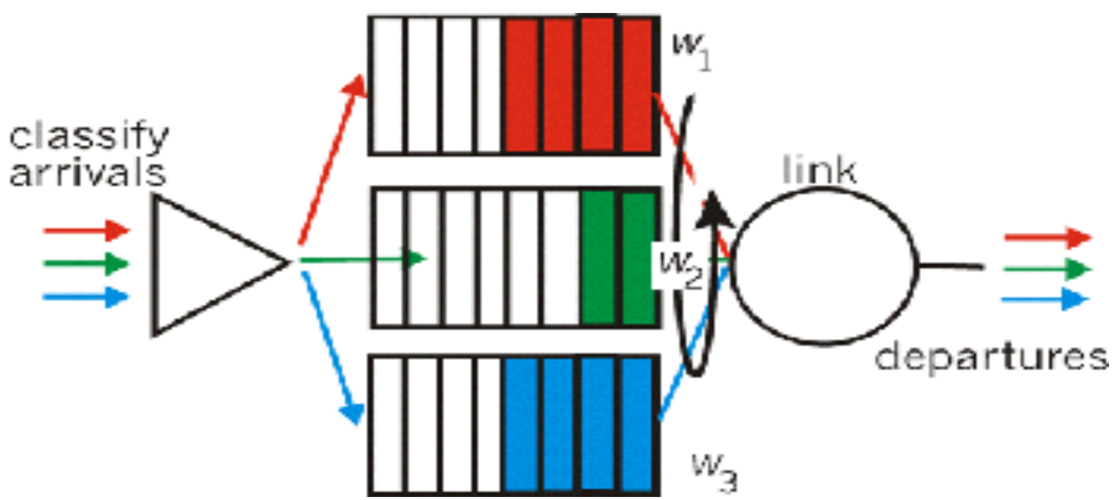
- multiple classes
- cyclically scan class queues, sending one complete packet from each class (if available)
- real world example?



Scheduling policies: still more

Weighted Fair Queuing (WFQ):

- generalized Round Robin
- each class gets weighted amount of service per cycle
- real-world example?



Uploading course materials to sites such as CourseHero, Chegg or Github is academic misconduct at Columbia (see pg 10 of Columbia guide).