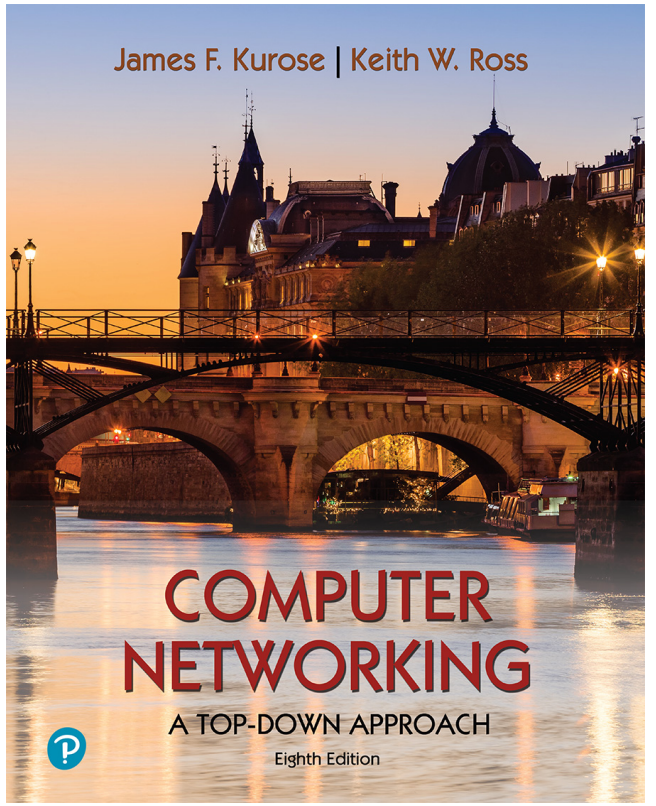
Midterm Exam 2024 Questions (Selected)



Computer Networking: A Top-Down Approach
8th edition
Jim Kurose, Keith Ross
Pearson, 2020

Question 1.3-1 1.3-3

- 1.3-1 Routing versus forwarding. Choose one of the following two definitions that makes the correct distinction between routing versus forwarding.
- ANS: Forwarding is the local action of moving arriving packets from router's input link to appropriate router output link, while routing is the global action of determining the source-destination paths taken by packets.

- 1.3-3 Packet switching versus circuit switching (2). Which of the characteristics below are associated with the technique of circuit switching? Select all correct answers. [Hint: more than one of the answers is correct].
- ANS: Reserves resources needed for a call from source to destination.
- Frequency Division Multiplexing (FDM) and Time Division Multiplexing (TDM) are two approaches for implementing this technique.

Question 1.4-01

- 1.4-01 Components of packet delay. Match the description of each component of packet delay to its name in the pull down list.

Time needed to perform an integrity check, lookup packet information in a local table and move the packet from an input link to an output link in a router.	Processing delay
Time spent waiting in packet buffers for link transmission.	Queueing delay
Time spent transmitting packets bits into the link.	Transmission delay
Time need for bits to physically propagate through the transmission medium from end one of a link to the other.	Propagation delay

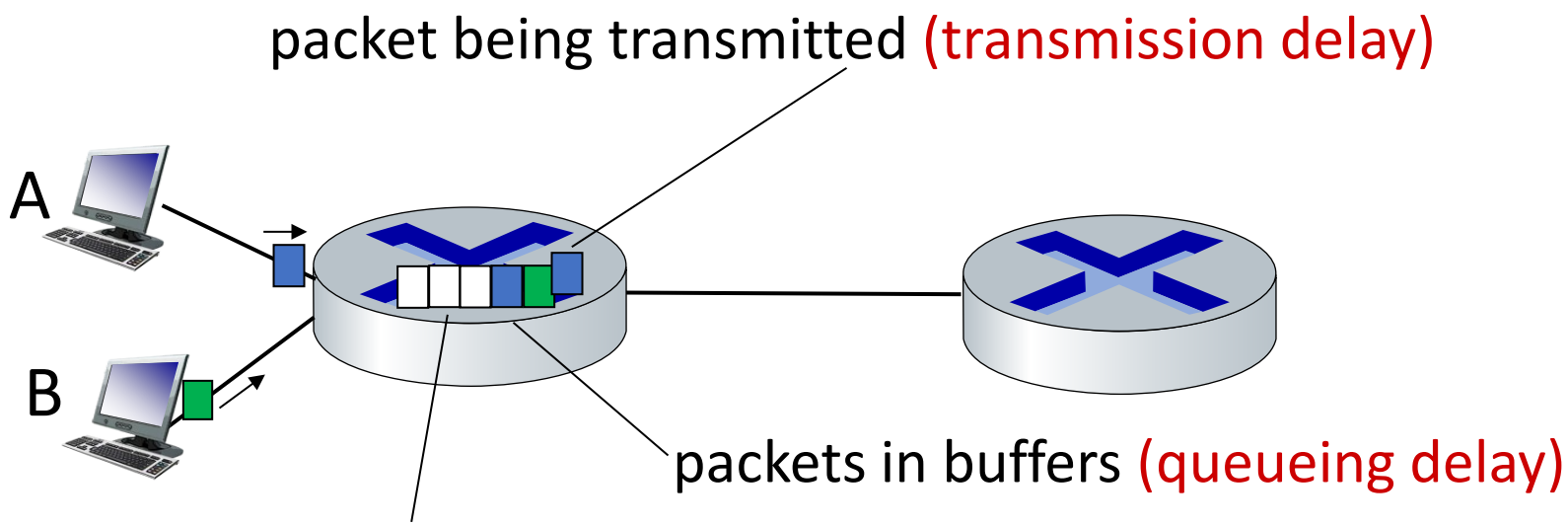
Chapter 1: roadmap

- What *is* the Internet?
- What *is* a protocol?
- Network edge: hosts, access network, physical media
- Network core: packet/circuit switching, internet structure
- Performance: loss, delay, throughput
- Security
- Protocol layers, service models

IMPORTANT

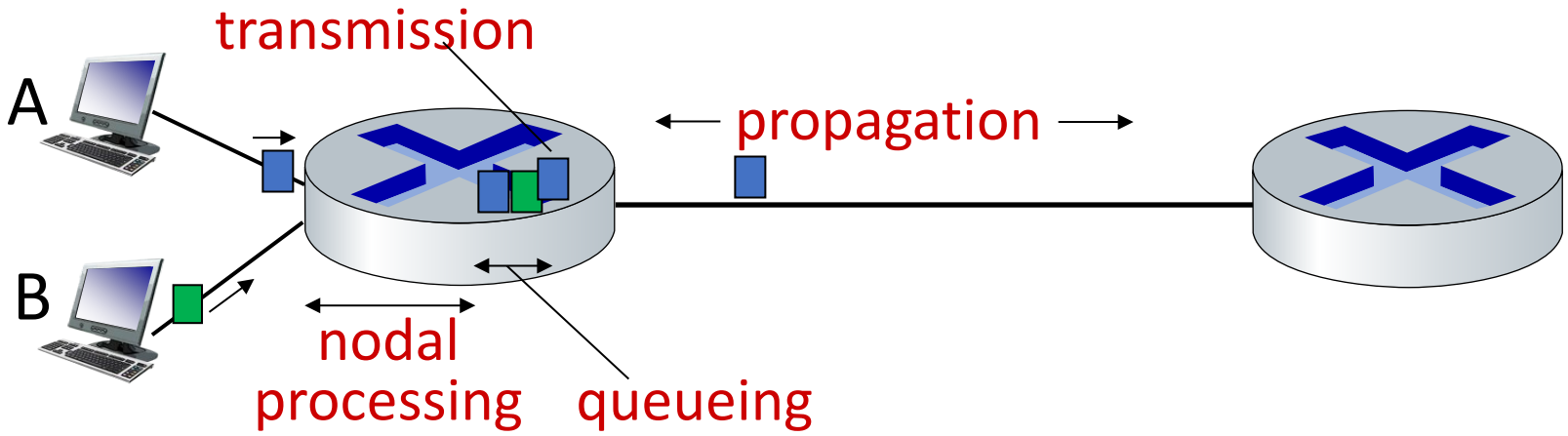
How do packet delay and loss occur?

- packets *queue* in router buffers, waiting for turn for transmission
  - queue length grows when arrival rate to link (temporarily) exceeds output link capacity
- packet *loss* occurs when memory to hold queued packets fills up



IMPORTANT

Packet delay: four sources



$$d_{nodal} = d_{proc} + d_{queue} + d_{trans} + d_{prop}$$

d_{proc}: nodal processing

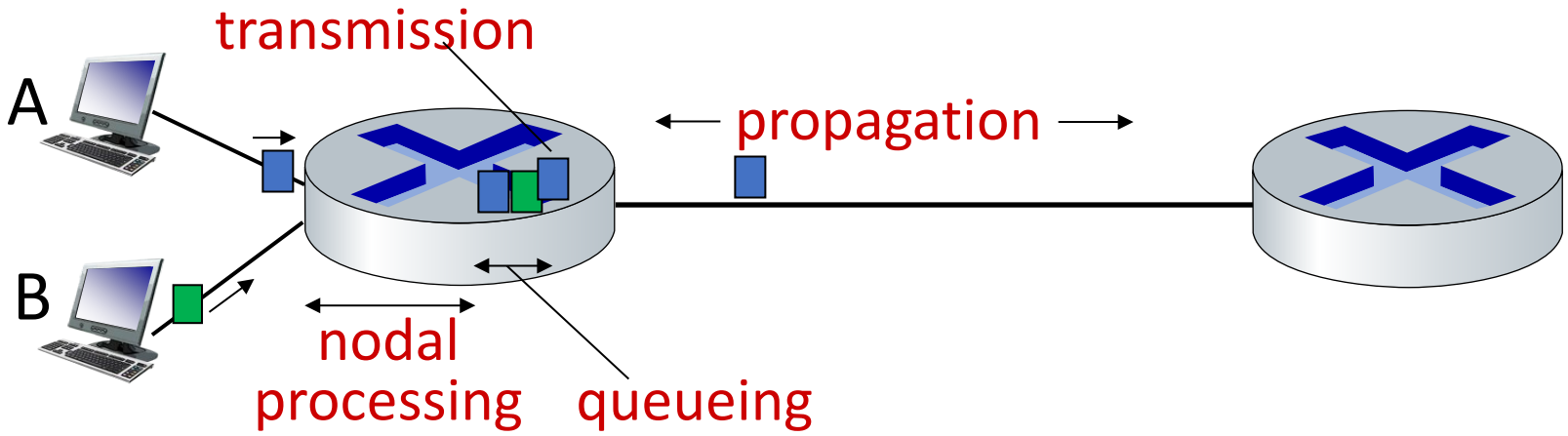
- check bit errors
- determine output link
- typically < microsecs

d_{queue}: queueing delay

- time waiting at output link for transmission
- depends on congestion level of router

IMPORTANT

Packet delay: four sources



$$d_{nodal} = d_{proc} + d_{queue} + d_{trans} + d_{prop}$$

d_{trans}: transmission delay:

- L: packet length (bits)
- R: link *transmission rate (bps)*
- $d_{trans} = L/R$

d_{prop}: propagation delay:

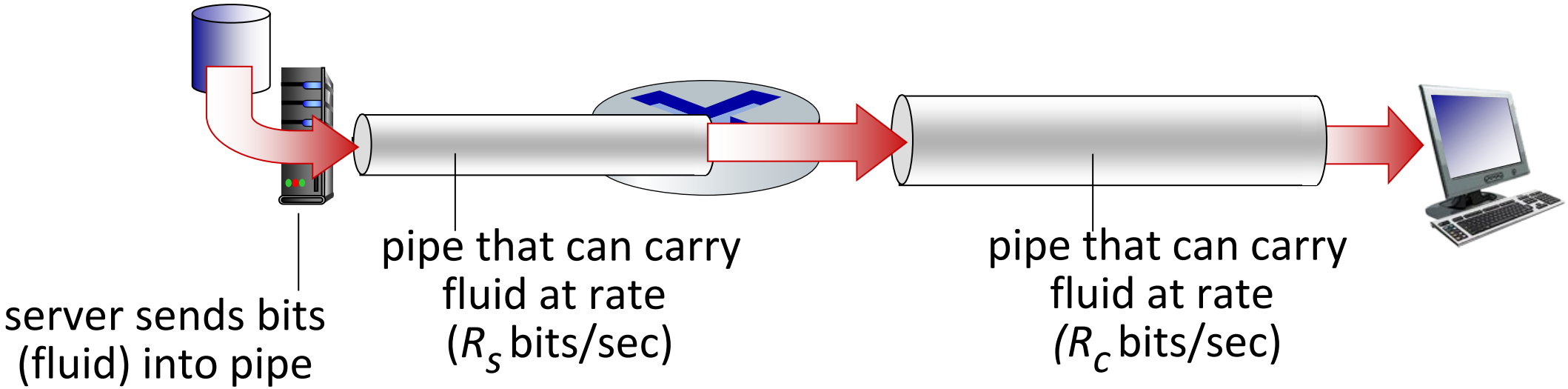
- d: length of physical link
- s: propagation speed (~2×10^8 m/sec)
- $d_{prop} = d/s$

d_{trans} and d_{prop} *very* different

IMPORTANT

Throughput

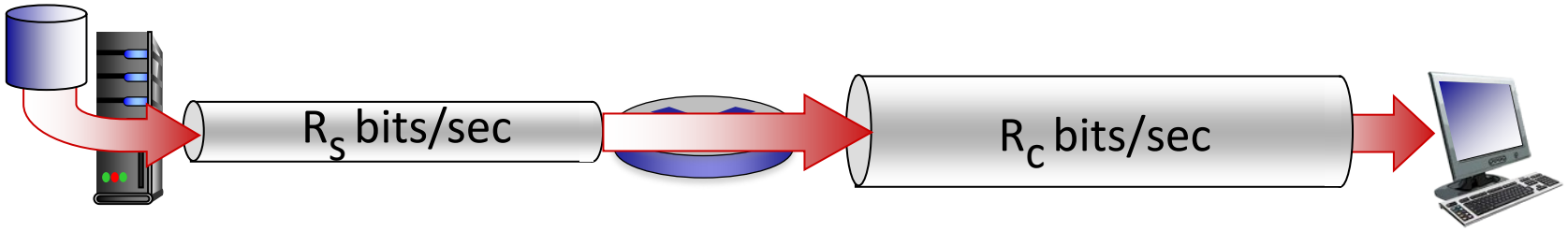
- *throughput:* rate (bits/time unit) at which bits are being sent from sender to receiver
  - *instantaneous:* rate at given point in time
  - *average:* rate over longer period of time



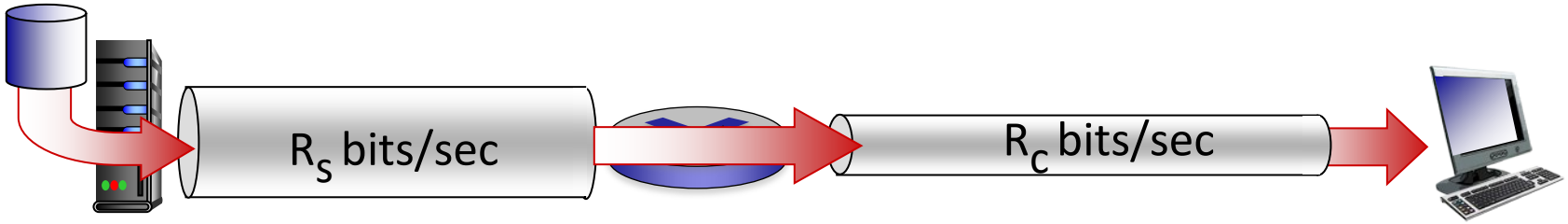
IMPORTANT

Throughput

$R_s < R_c$ What is average end-end throughput?



$R_s > R_c$ What is average end-end throughput?

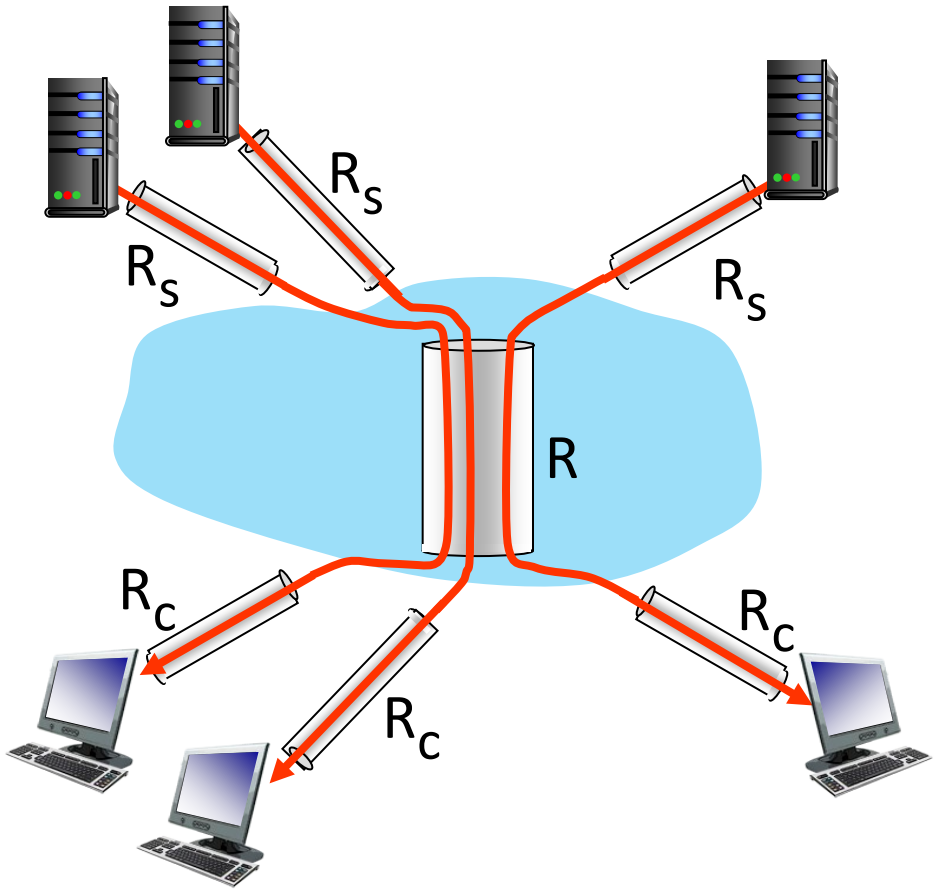


bottleneck link

link on end-end path that constrains end-end throughput

IMPORTANT

Throughput: network scenario



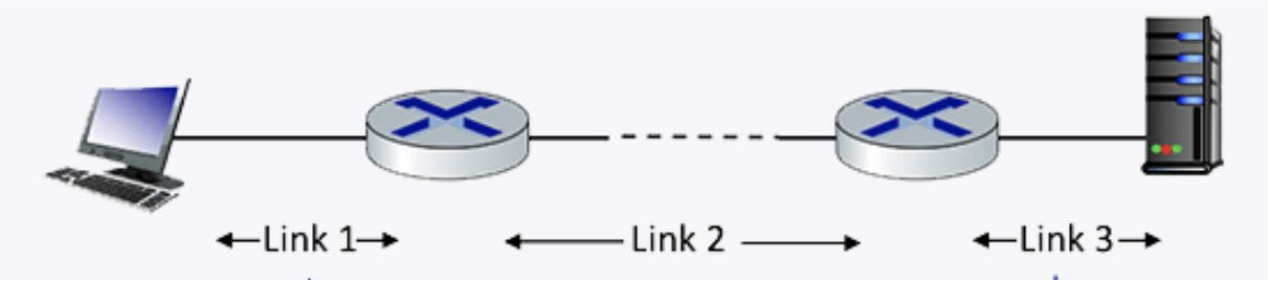
10 connections (fairly) share backbone bottleneck link *R* bits/sec

- per-connection end-end throughput: $min(R_c, R_s, R/10)$
- in practice: R_c or R_s is often bottleneck
- Link utilization: used bandwidth/available bandwidth. For the three links:
  - $min(R_c, R_s, R/10)/R_s$
  - $min(R_c, R_s, R/10)/(R/10)$
  - $min(R_c, R_s, R/10)/R_c$

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

Question 1.4-01a

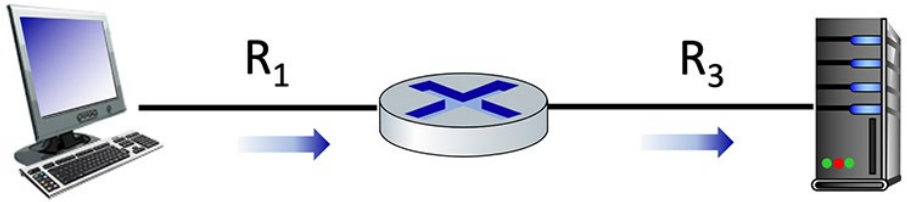
- 1.4-01a. Performance: Delay. Consider the network shown in the figure below, with three links, each with a transmission rate of 1 Mbps, and a propagation delay of 1 msec per link. Assume the length of a packet is 1000 bits.
- What is the end-end delay of a packet from when it first begins transmission on link 1, until is it received in full by the server at the end of link 3. You can assume that queueing delays and packet processing delays are zero, but make sure you include packet transmission time delay on all links. Assume store-and forward packet transmission.



- Each link has transmission delay 1000 bits/1 Mbps = 1ms. So each link has total delay of transmission + propagation = 1+1=2 ms.
- For the three links, the total delay = 2 + 2 + 2 = 6 ms. `

Question 1.4-01b

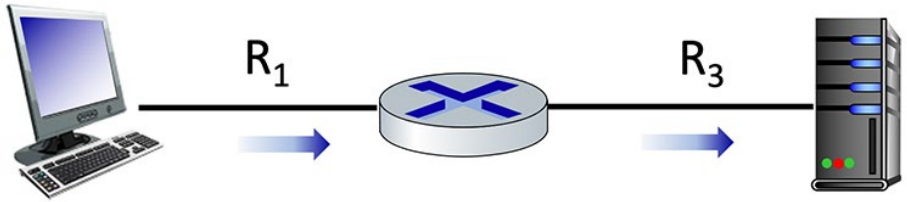
- Maximum end-end throughput. Consider the scenario shown below, with a single source client sending to a server over two links of capacities R1=100 Mbps and R3=10 Mbps.
- What is the maximum achievable end-end throughput (in Mbps, give an integer value) for the client-to-server pair, assuming that the client is trying to send at its maximum rate?



- min (R1, R3) = min (100, 10) = 10 Mpbs.

Question 1.4-01c

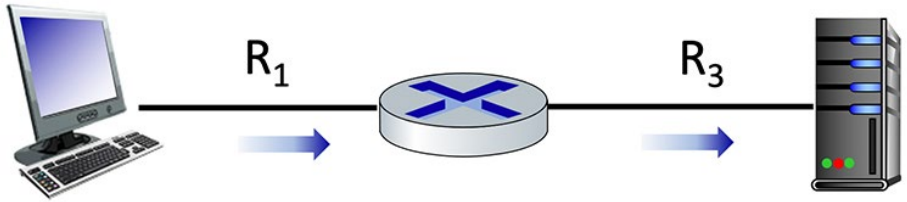
- Performance: Maximum end-end throughput. Consider the scenario shown below, with a single source client sending to a server over two links of capacities R1=10 Mbps and R3=100 Mbps.
- What is the maximum achievable end-end throughput (in Mbps, give an integer value) for the client-to-server pair, assuming that the client is trying to send at its maximum rate?



- min (R1, R3) = min (10, 100) = 10 Mpbs.

Question 1.4-01c

- Performance: Maximum end-end throughput. Consider the scenario shown below, with a single source client sending to a server over two links of capacities R1=10 Mbps and R3=100 Mbps.
- What is the maximum achievable end-end throughput (in Mbps, give an integer value) for the client-to-server pair, assuming that the client is trying to send at its maximum rate?

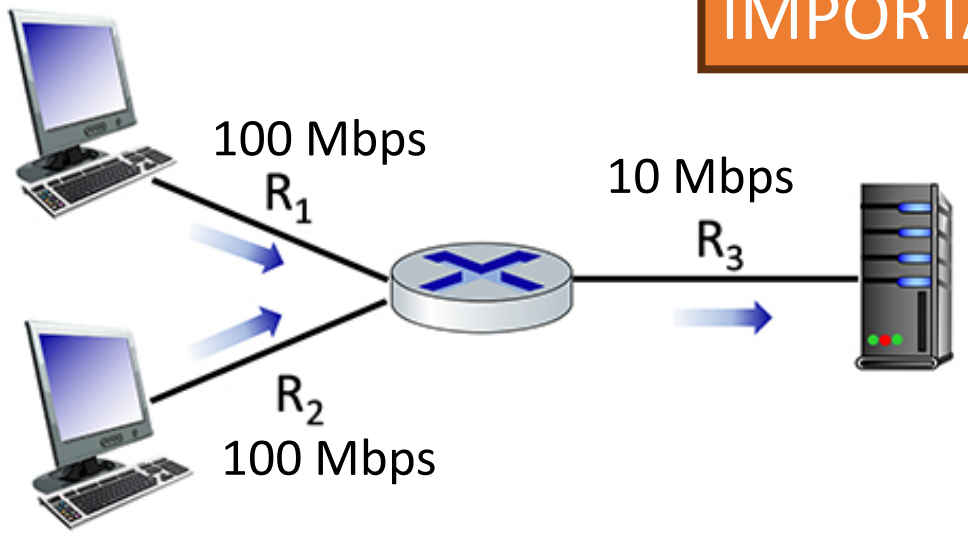


- min (R1, R3) = min (10, 100) = 10 Mpbs.

IMPORTANT

Question 1.4-01d

- 1.4-01e. Performance: Maximum end-end throughput. Consider the scenario shown below, with two clients sending to a server. The links attached to clients each have a capacity of R1= R2 = 100 Mbps. The link from the router to the server has a capacity of and R3 = 10 Mbps, which is shared evenly between the two sources when they are each sending at their maximum rate.
- What is the maximum achievable end-end throughput (in Mbps, give an integer value) for each client-to-server pair, assuming that the client is trying to send at its maximum rate?

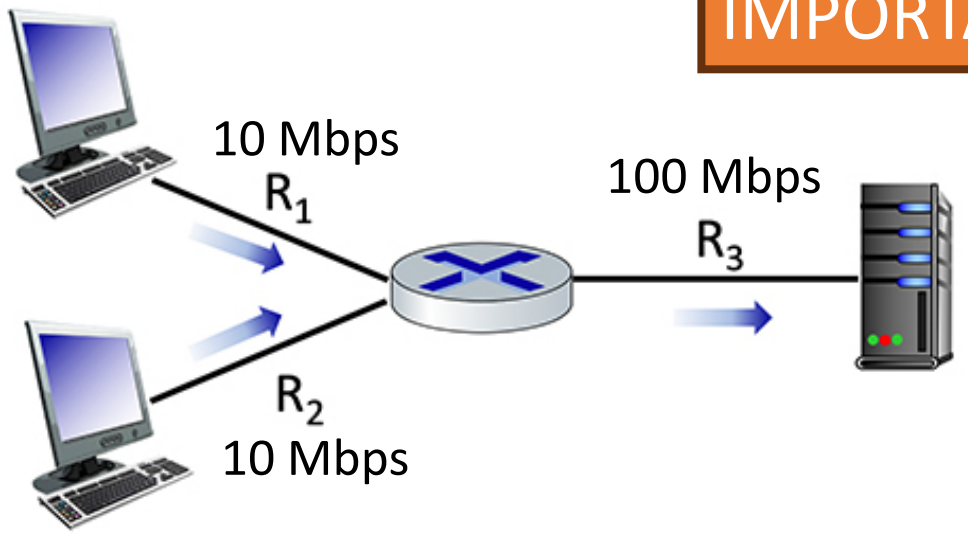


- The maximum achievable end-end throughput is min (R1, R3/2) = min (R2, R3/2) = min (100, 10/2) = 5 Mbps.
- The shared 100 Mbps link is fairly shared among two end-to-end connections (R1, R3) and (R2, R3), hence each end-to-end connection gets 10/2=5 Mbps.

IMPORTANT

Question 1.4-01e

- 1.4-01e. Performance: Maximum end-end throughput. Consider the scenario shown below, with two clients sending to a server. The links attached to clients each have a capacity of R1= R2 = 10 Mbps. The link from the router to the server has a capacity of and R3 = 100 Mbps, which is shared evenly between the two sources when they are each sending at their maximum rate.
- What is the maximum achievable end-end throughput (in Mbps, give an integer value) for each client-to-server pair, assuming that the client is trying to send at its maximum rate?

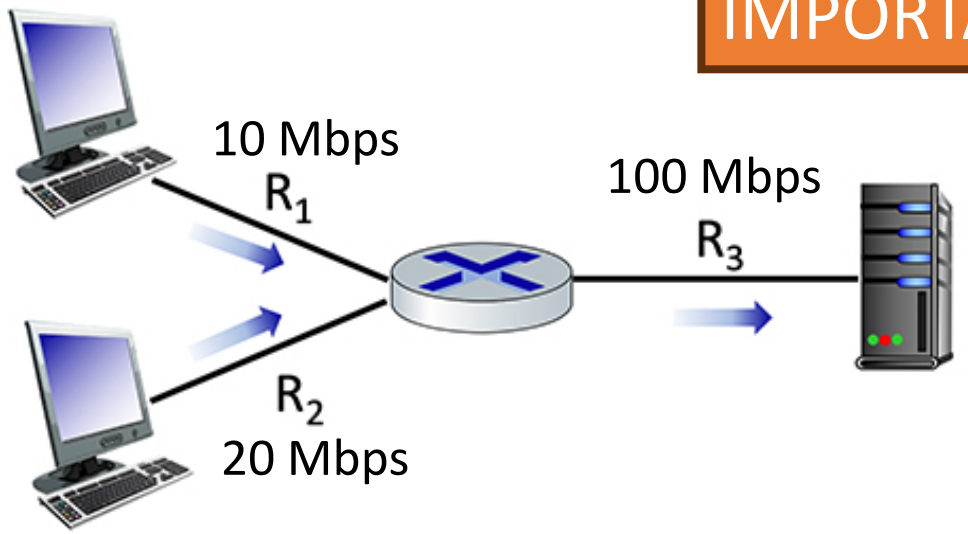


- The maximum achievable end-end throughput is min (R1, R3/2) = min (R2, R3/2) = min (10, 100/2) = 10 Mbps.
- The shared 100 Mbps link is fairly shared among two end-to-end connections (R1, R3) and (R2, R3), hence each end-to-end connection gets 100/2=50 Mbps.
- Note that min (R1, R2, R3/2) is incorrect, since R1 and R2 are on deferent end-to-end connections.
- Remember that parallel links should never be put into the min() formula, only sequential links forming the same end-to-end connection should be in the same min() formula.

Question 1.4-01e variations

IMPORTANT

- 1.4-01e. Performance: Maximum end-end throughput. Consider the scenario shown below, with two clients sending to a server. The links attached to clients each have a capacity of R1 = 10 Mbps, R2 = 20 Mbps. The link from the router to the server has a capacity of and R3 = 100 Mbps, which is shared evenly between the two sources when they are each sending at their maximum rate.
- What is the maximum achievable end-end throughput (in Mbps, give an integer value) for each client-to-server pair, assuming that the client is trying to send at its maximum rate?



- For client on top, the maximum achievable end-end throughput is min (R1, R3/2) = min (10, 100/2) = 10 Mbps.
- For client on bottom, the maximum achievable end-end throughput is min (R1, R3/2) = min (20, 100/2) = 20 Mbps.
- The shared 100 Mbps link is fairly shared among two end-to-end connections (R1, R3) and (R2, R3), hence each end-to-end connection gets 100/2=50 Mbps.

Question 1.4-02a

- Consider the scenario shown in Figure 1 in which a server is connected to a router by a 100Mbps link with a 50ms propagation delay. Initially this router is also connected to two routers, each over a 50Mbps link with a 200ms propagation delay. A 1Gbps link connects a host and a cache (if present) to each of these routers and we assume that this link has 0 propagation delay. All packets in the network are 20,000 bits long.
- What is the end-to-end delay (in msec) from when a packet is transmitted by the server to when it is received by the client? In this case, we assume there are no caches, there's no queuing delay at the routers, and the packet processing delays at routers and nodes are all 0.

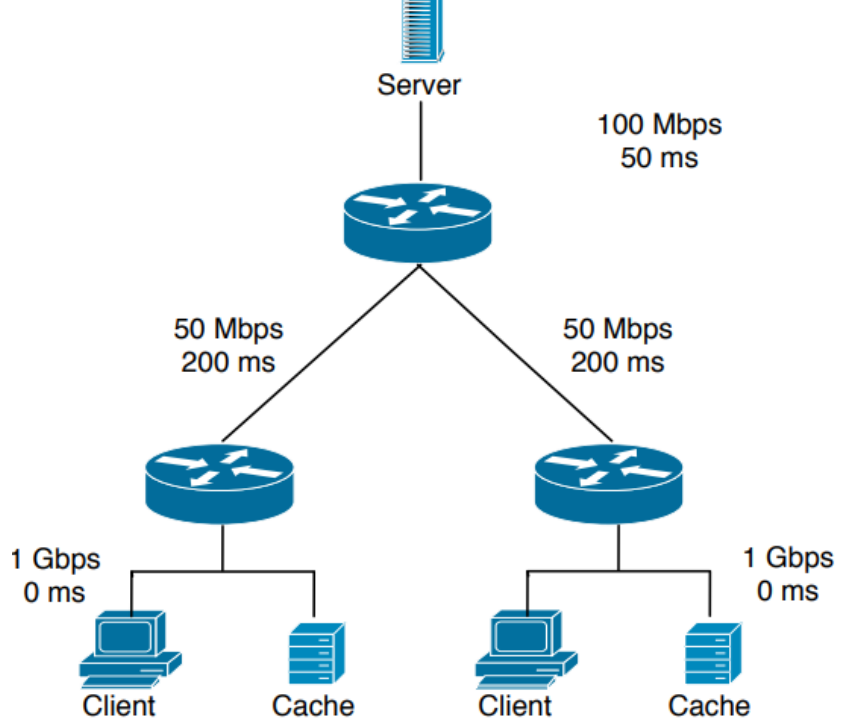


Figure 1

- ANS: If all packets are 20,000 bits long it takes 200 usec to send the packet over the 100Mbps link, 400 usec to send over the 50Mbps link, and 20 usec to send over the 1Gbps link.
- Sum of the three-link transmission is 620 usec. Thus, the total end-to-end delay is 250 ms + 620 usec = 250.62 msec.

Question 1.4-02b

- Here we assume that client hosts send requests for files directly to the server (caches are not used or off in this case). What is the maximum rate at which the server can deliver data to a single client if we assume no other clients are making requests?

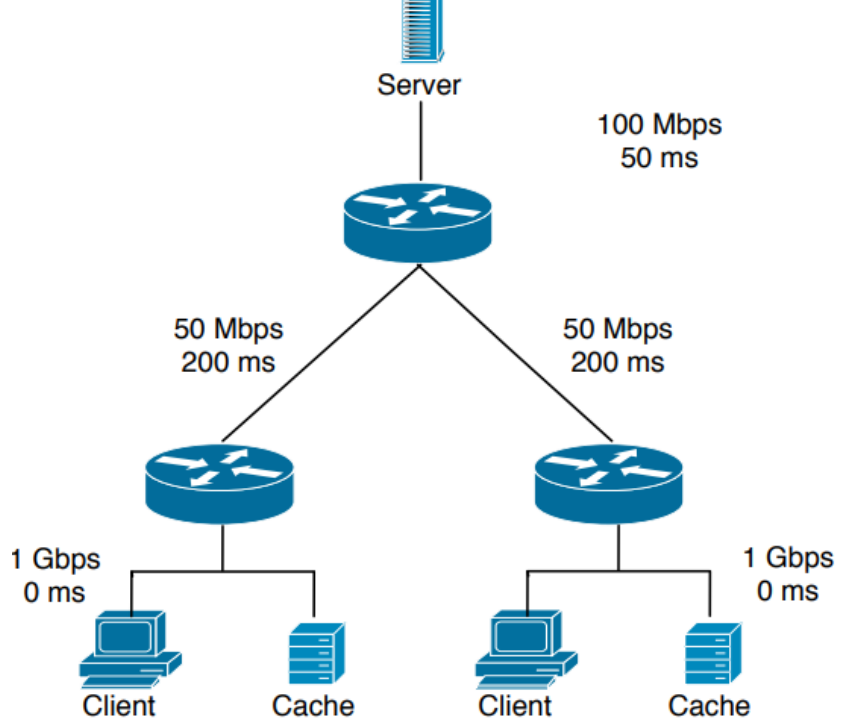


Figure 1

- Server can send at the max of the bottleneck link bandwidth min(100/2 Mbps, 50 Mbps, 1 Gbps) = 50 Mbps, since the top link is shared and the other two links are separate for each client and not shared.
- Suppose we had 10 clients on the bottom, then the bottleneck link bandwidth min(100/10 Mbps, 50 Mbps, 1 Gbps) = 10 Mbps.

Question 1.4-02c

- Again we assume only one active client but in this case the caches are on and behave like HTTP caches. A client's HTTP GET is always first directed to its local cache. 60% of the requests can be satisfied by the local cache. What is the average rate at which the client can receive data in this case?

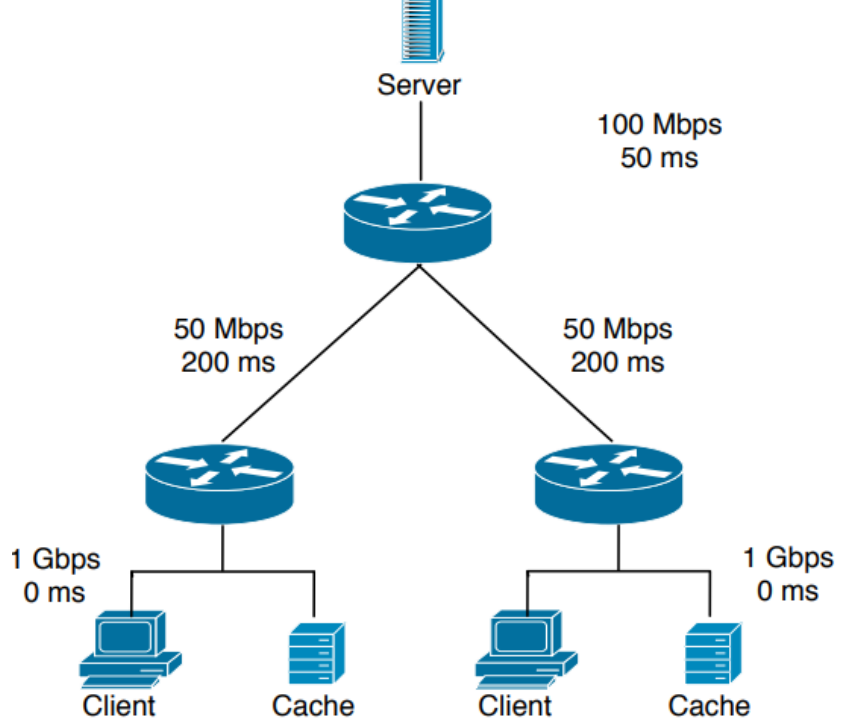


Figure 1

- 60% of the requests can be satisfied by the local cache, with bandwidth of 1 Gbps = 1000 Mpbs.
- 40% of the requests go to the remote server, with bandwidth of min (100 Mbps, 50 Mbps, 1Gbps) = 50 Mpbs.
  - Since we assume only one active client, we do not have 100/2 as the first term.
- The average rate at which the client can receive data is .4*50+.6*1000=620 Mbps

Question 1.4-02d

- Now clients in both LANs are active and the both caches are on. 60% of the requests can be satisfied by the local caches. What is the average rate at which each client can receive data?

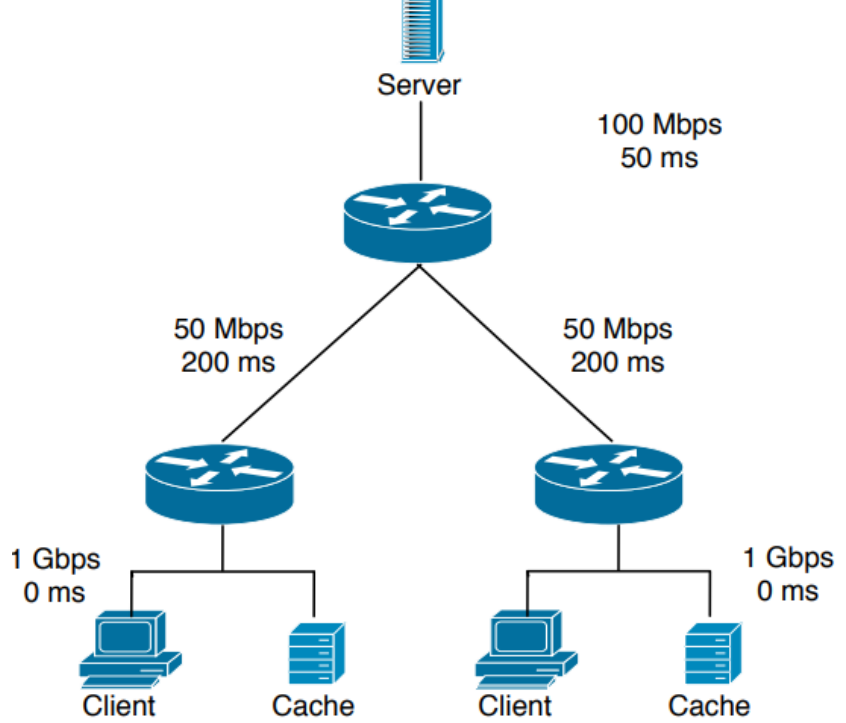
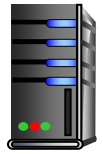


Figure 1

- 60% of the requests can be satisfied by the local cache, with bandwidth of 1 Gbps = 1000 Mpbs.
- 40% of the requests go to the remote server, with bandwidth of min (100/2 Mbps, 50 Mbps, 1Gbps) = 50 Mpbs.
  - Since we both clients are active, we have 100/2 as the first term.
- The average rate at which the client can receive data is .4*50+.6*1000=620 Mbps

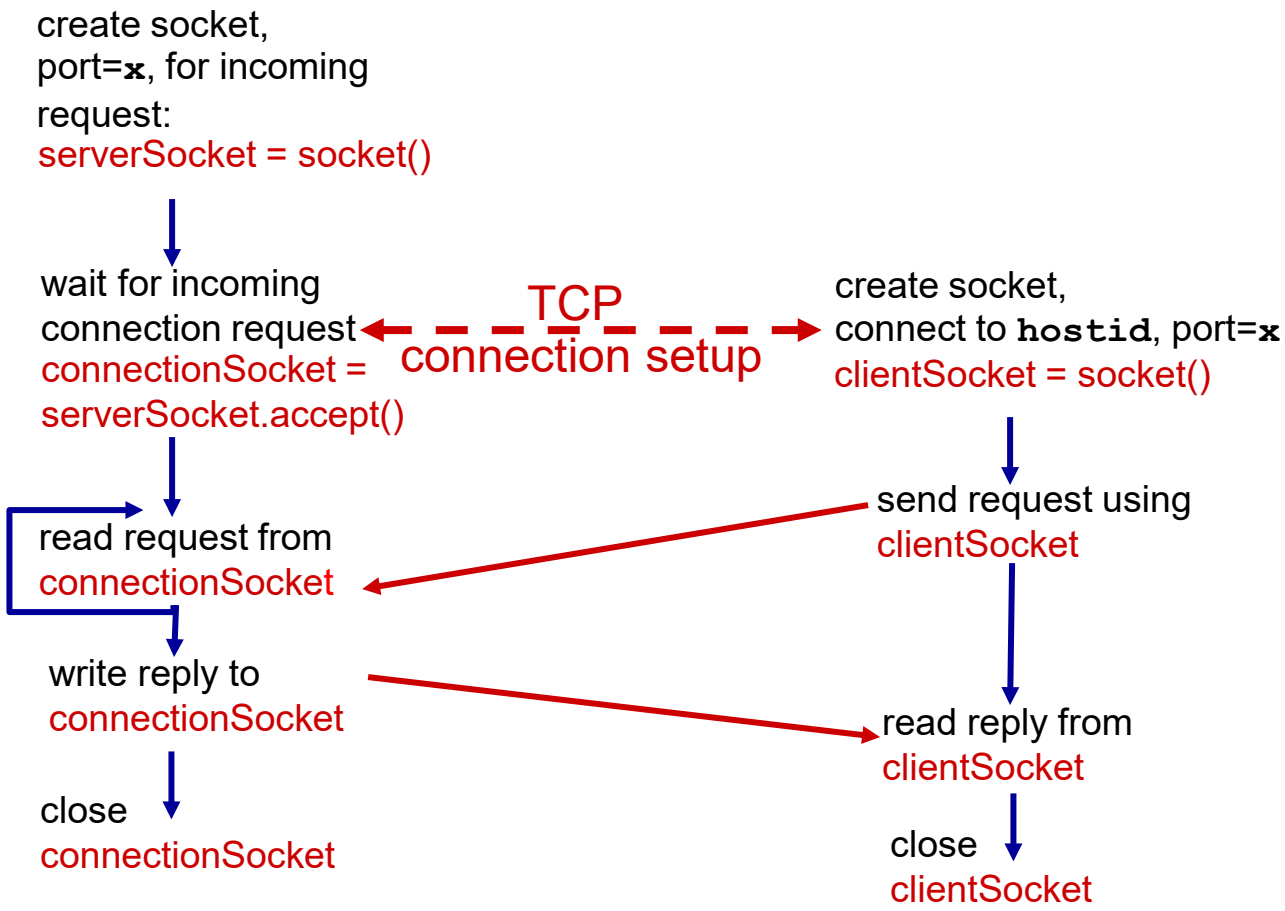
IMPORTANT

Client/server socket interaction: TCP



server (running on hostid)

client 



IMPORTANT

Example app: TCP client

Python TCPClient

```
from socket import *
serverName = 'servername'
serverPort = 12000
clientSocket = socket(AF_INET, SOCK_STREAM)
clientSocket.connect((serverName,serverPort))
sentence = input('Input lowercase sentence:')
clientSocket.send(sentence.encode())
modifiedSentence = clientSocket.recv(1024)
print ('From Server:', modifiedSentence.decode())
clientSocket.close()
```

create TCP socket for server, remote port 12000

No need to attach server name, port

Note: this code update (2023) to Python 3

IMPORTANT

Example app: TCP server

Python TCPServer

```
from socket import *
serverPort = 12000
serverSocket = socket(AF_INET,SOCK_STREAM)
serverSocket.bind(('',serverPort))
serverSocket.listen(1)
print('The server is ready to receive')
while True:
    connectionSocket, addr = serverSocket.accept()

    sentence = connectionSocket.recv(1024).decode()
    capitalizedSentence = sentence.upper()
    connectionSocket.send(capitalizedSentence.
                          encode())
    connectionSocket.close()
```

create TCP welcoming socket

server begins listening for incoming TCP requests

loop forever

server waits on accept() for incoming requests, new socket created on return

read bytes from socket (but not address as in UDP)

close connection to this client (but *not* welcoming socket)

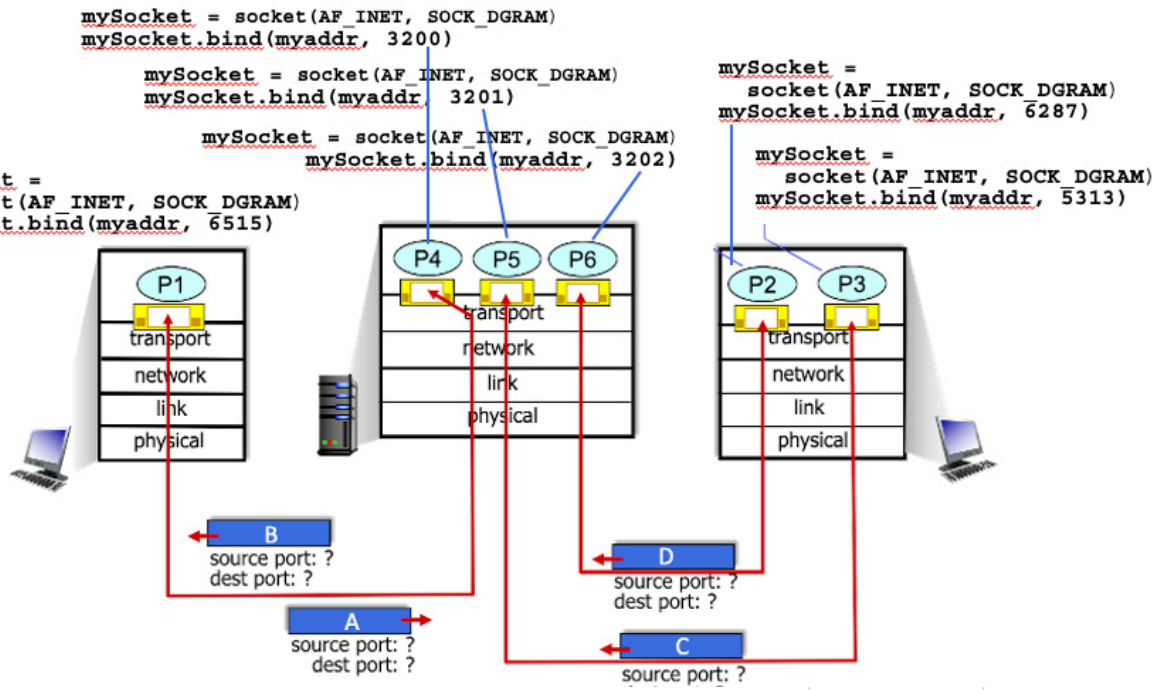
Note: this code update (2023) to Python 3

Question 2.7-4

- **2.7-4 How many sockets?** Suppose a Web server has *five* ongoing connections that use TCP receiver port 80, and assume there are no other TCP connections (open or being opened or closed) at that server. How many TCP sockets are in use at this server?
- ANS: 6.
- 1 serverSocket and 5 connectionSockets

Question 3.2-01

- **3.2-01. UDP multiplexing and demultiplexing.** Consider the figure below, with 6 sockets shown across the network, and the corresponding Python code at each host. There are four UDP segments in flight. Match the source and destination port numbers for each segment with a value below.



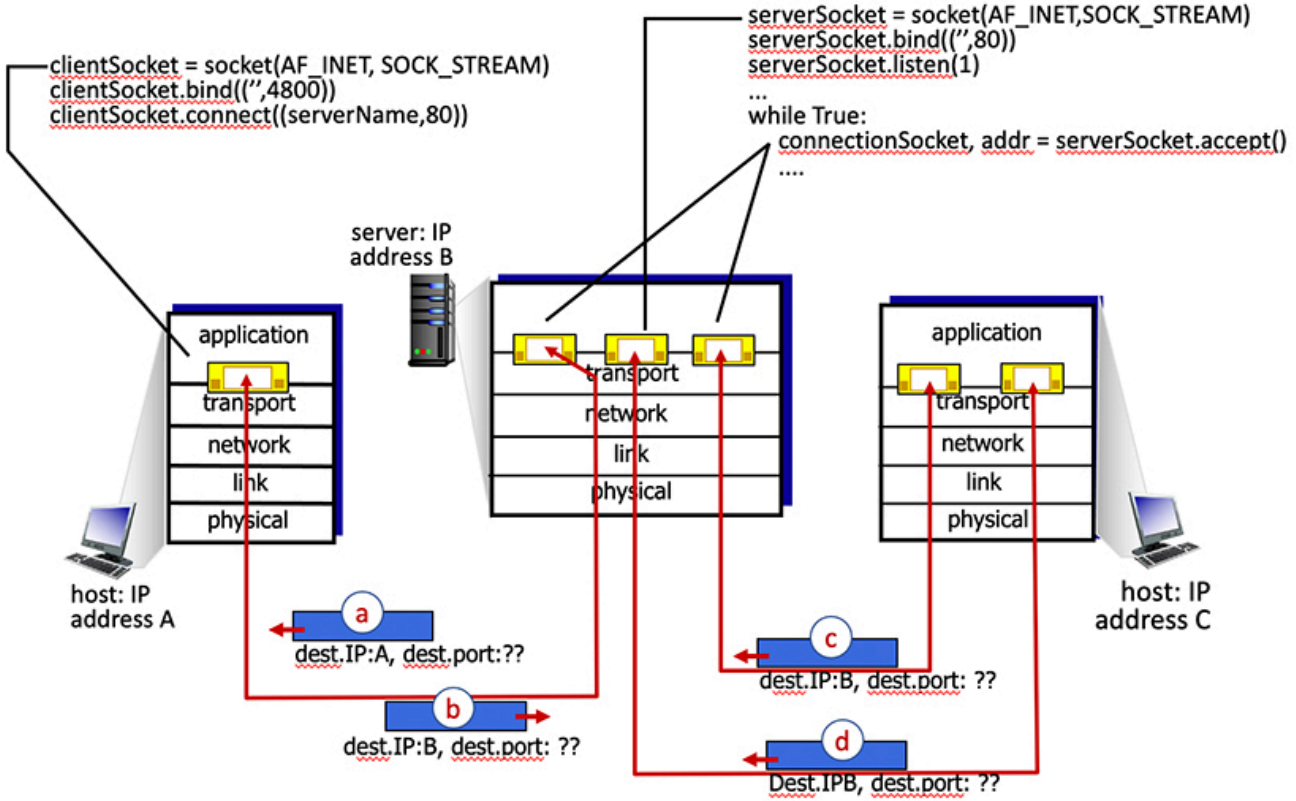
Segment A source port #	6515
...	
Segment B source port #	3200
...	
Segment C source port #	5313
...	
Segment C destination port #	3201
...	
Segment D source port #	6287
...	
Segment D destination port #	3202

Question 3.2-04

- **3.2-04. TCP demultiplexing.** Which of the following datagram and segment header fields are used, when demultiplexing data up to a TCP socket?
- ANS: Source and destination IP addresses, and source and destination port numbers.

Question 3.2-05

- **3.2-05a. TCP multiplexing/demultiplexing and connection management.** Consider the scenario in the figure below, with two client hosts – client A (with one TCP socket) and client B (with two TCP sockets) exchanging packets with server B (with three TCP sockets). The Python code that created the three sockets on the server and the TCP code that created the one socket at the left client is shown in the diagram, with lines indicating the line of executed Python code that *created* each of the sockets. Which of the packets a, b, c, d have destination port 80?
- ANS: Packets b, c and d



IMPORTANT

Internet checksum

Goal: detect errors (*i.e.,* flipped bits) in transmitted segment

sender:

- treat contents of UDP segment (including UDP header fields and IP addresses) as sequence of 16-bit integers
- checksum: addition of segment content, then flip all the bits (one's complement sum)
- checksum value put into UDP checksum field

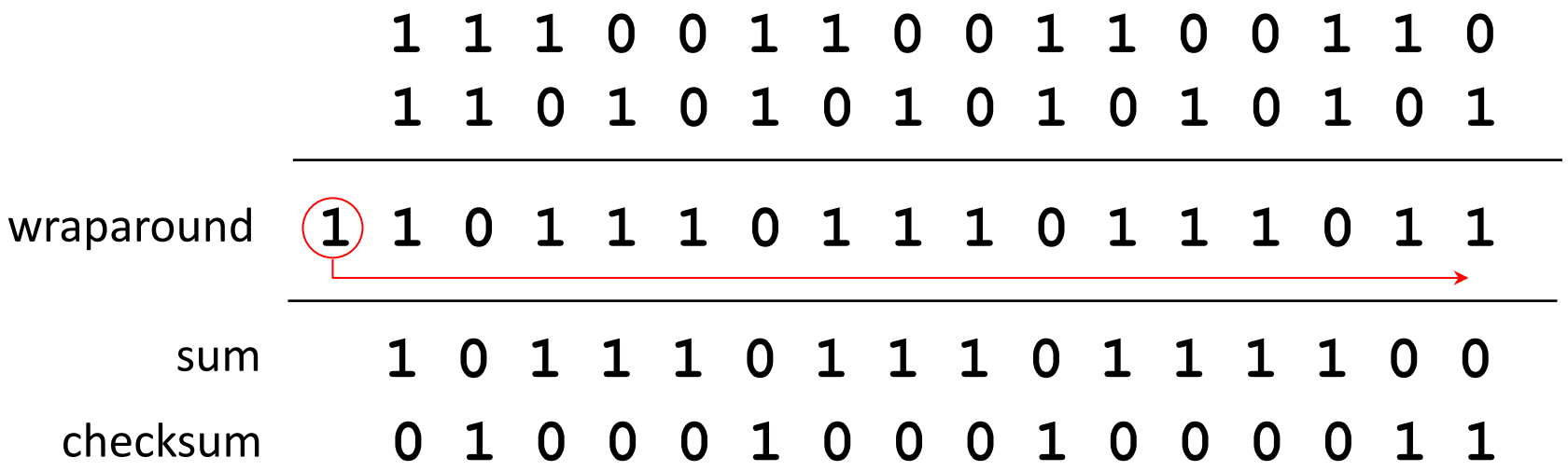
receiver:

- compute checksum of received segment
- check if computed checksum equals checksum field value:
  - not equal - error detected
  - equal - no error detected. *But maybe errors nonetheless?* More later ….

IMPORTANT

Internet checksum: an example

example: add two 16-bit integers



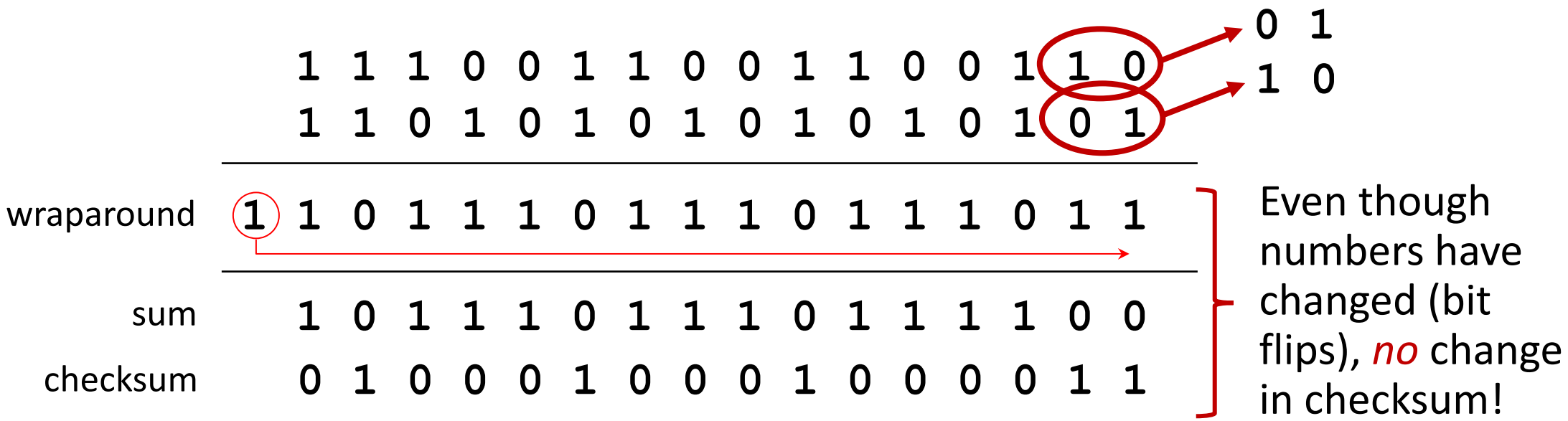
Note: when adding numbers, a carryout from the most significant bit needs to be added to the result

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

IMPORTANT

Internet checksum: weak protection!

example: add two 16-bit integers



Question 3.3-1

- **3.3-1. Internet Checksum.** Consider the two sixteen bit numbers:

  ```
  10110100 01000110
  01001000 01101111
  ```

  Compute the Internet Checksum of these two values

  Enter the 2 bytes each as an 8-bit number with only 0's and 1's, and make a single blank space between the two 8-bit numbers (e.g., 01010101 00101000).

- ANS: adding up the two numbers we have: `11111100 10110101`
- Taking ones complement we have: `00000011 01001010`

Network layer: “data plane” roadmap

- Network layer: overview
  - data plane
  - control plane
- What’s inside a router
  - input ports, switching, output ports
  - buffer management, scheduling
- IP: the Internet Protocol
  - datagram format
  - addressing
  - network address translation
  - IPv6

- Generalized Forwarding, SDN
  - Match+action
  - OpenFlow: match+action in action
- Middleboxes

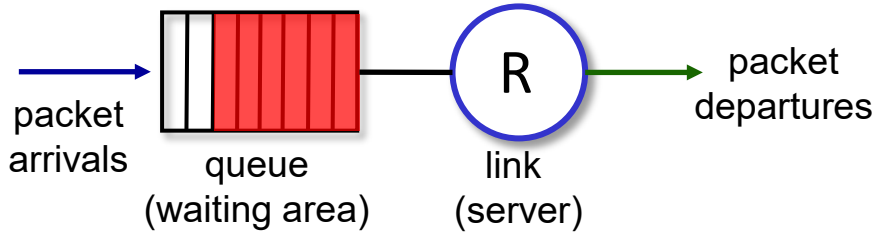
IMPORTANT

Packet Scheduling: FCFS

packet scheduling: deciding which packet to send next on link

- first come, first served
- priority
- round robin
- weighted fair queueing

Abstraction: queue



FCFS: packets transmitted in order of arrival to output port

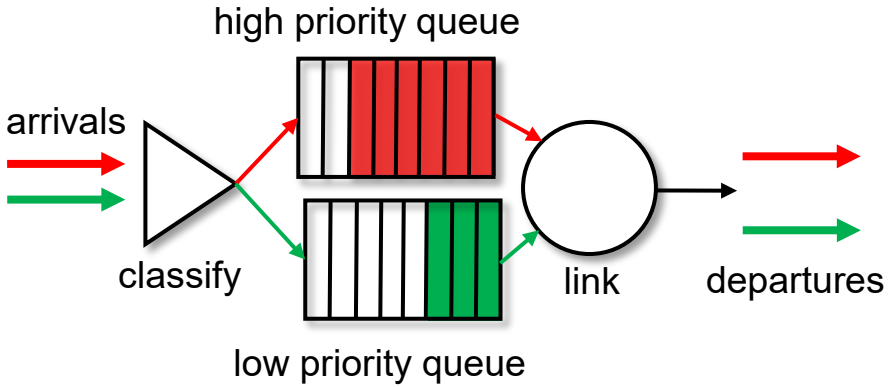
- also known as: First-in-first-out (FIFO)
- real world examples?

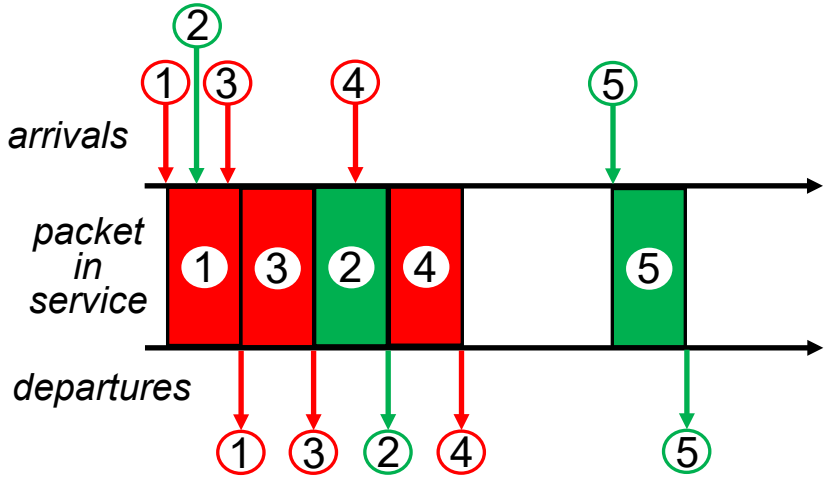
IMPORTANT

Scheduling policies: priority

Priority scheduling:

- arriving traffic classified, queued by class
  - any header fields can be used for classification
- send packet from highest priority queue that has buffered packets
  - FCFS within priority class



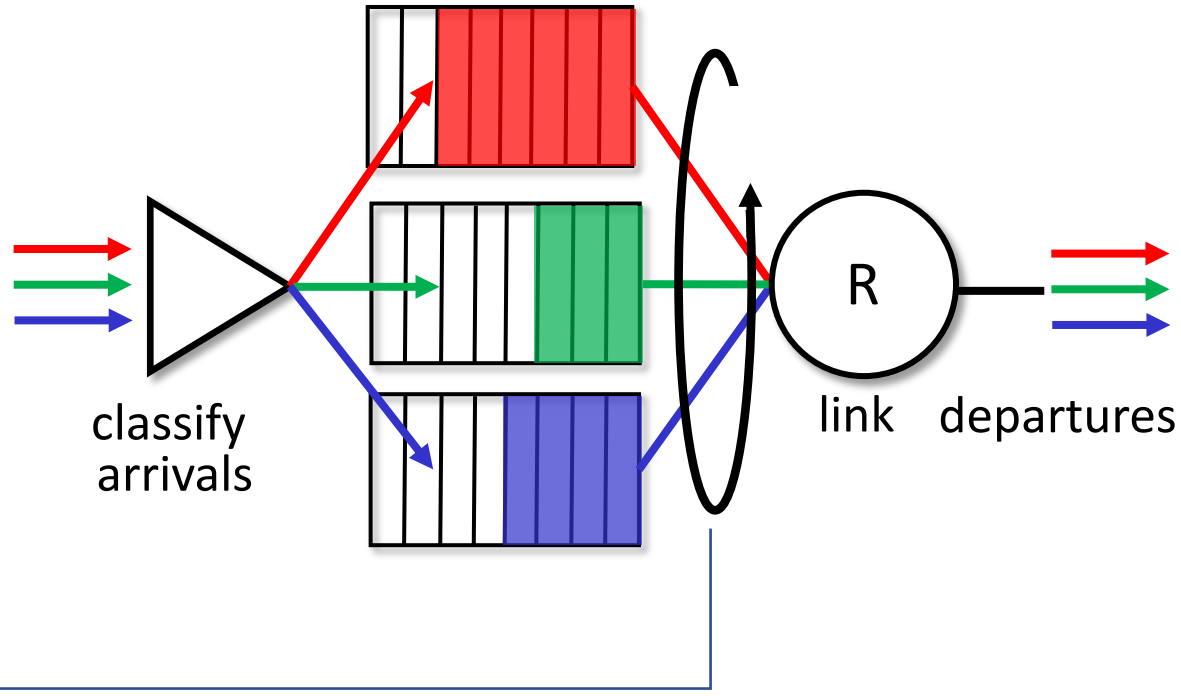


IMPORTANT

Scheduling policies: round robin

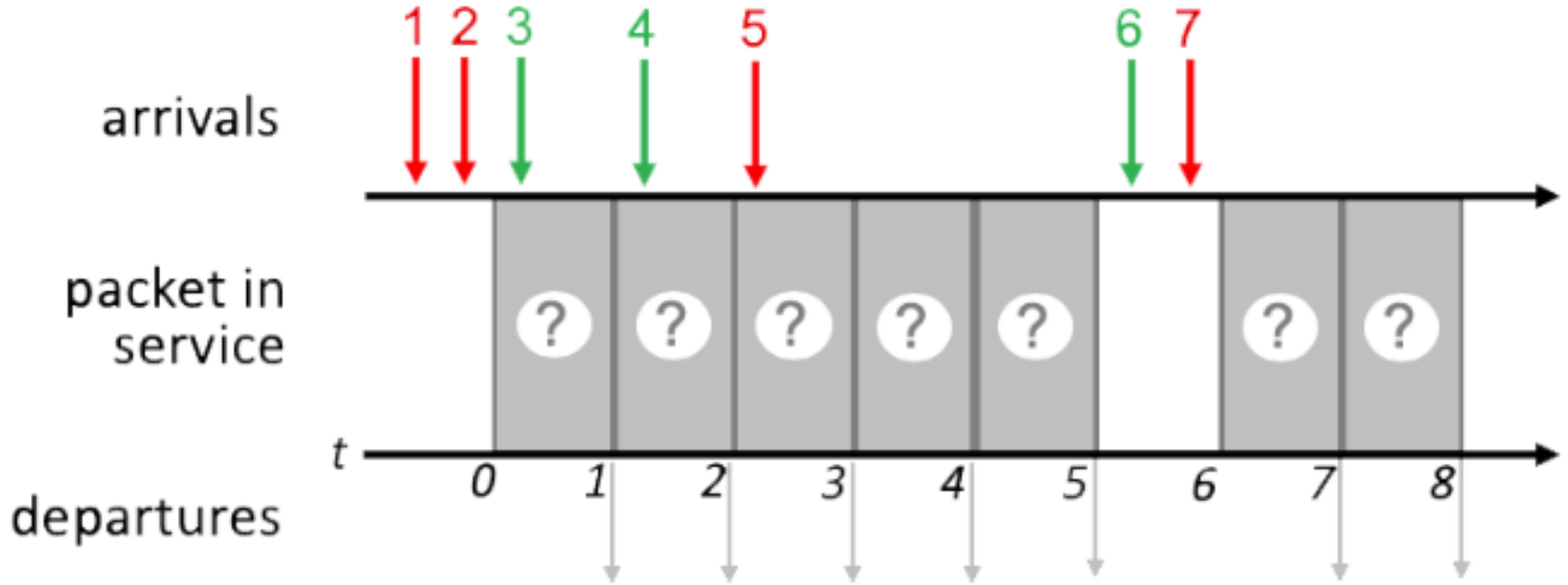
Round Robin (RR) scheduling:

- arriving traffic classified, queued by class
  - any header fields can be used for classification
- server cyclically, repeatedly scans class queues, sending one complete packet from each class (if available) in turn

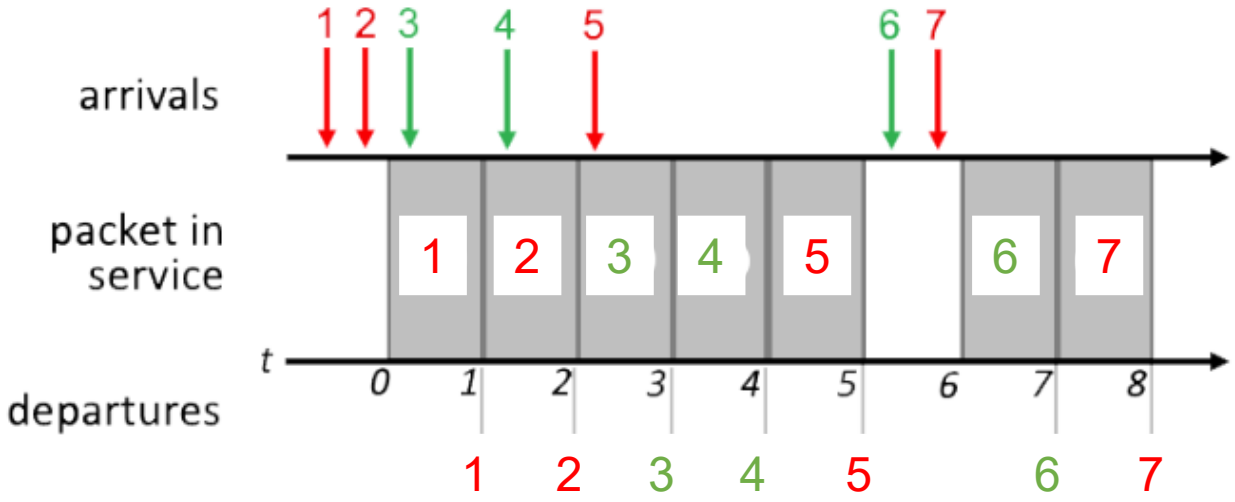


Question 4.2-7

- Packet scheduling. Consider the pattern of red and green packet arrivals to a router's output port queue, shown below. Suppose each packet takes one time slot to be transmitted, and can only begin transmission at the beginning of a time slot after its arrival. Indicate the sequence of departing packet numbers (at t = 1, 2, 3, 4, 5, 7, 8) under FCFS, priority, round-robin scheduling.



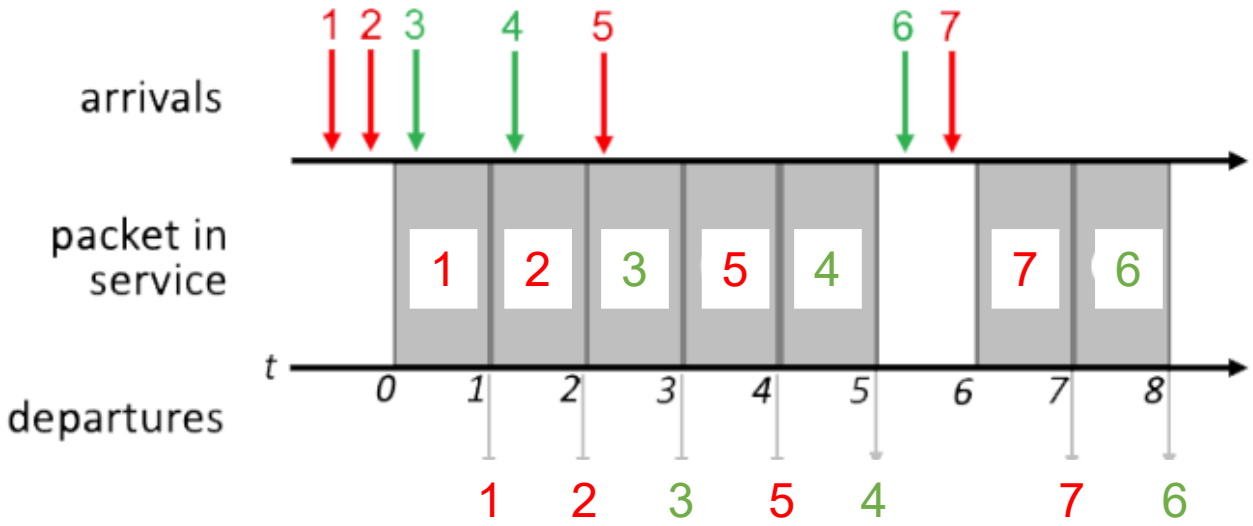
Question 4.2-7a FCFS Scheduling



- Transmit order the same as packet arrival order of 1 2 3 4 5 6 7

1 2 3 4 5 6 7

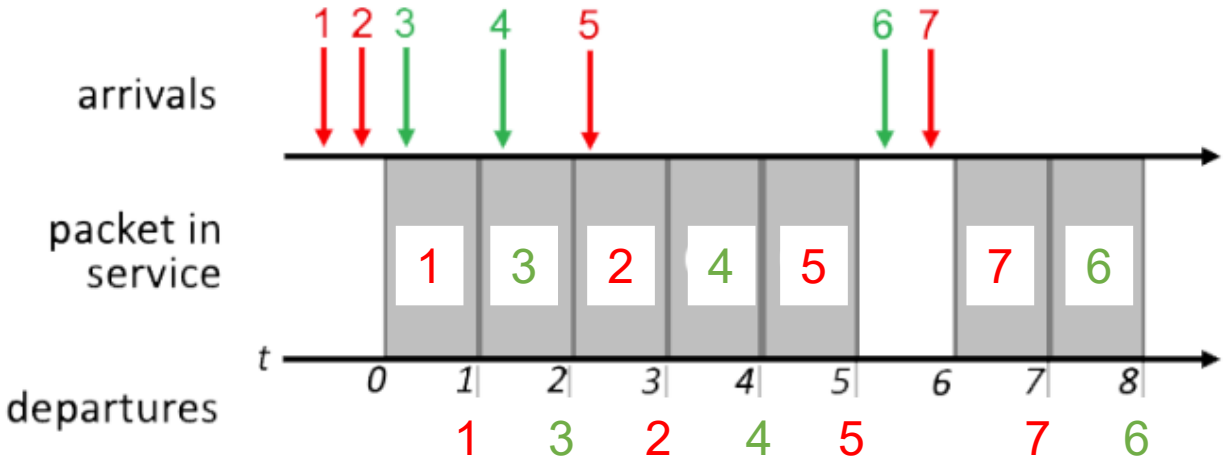
Question 4.2-7b Priority Scheduling



1 2 3 5 4 7 6

- Time 0: 1, 2 in queue, transmit 1
  - FCFS within same priority
- Time 1: 2, 3 in queue, transmit 2
- Time 2: 3, 4 in queue, transmit 3
  - FCFS within same priority
- Time 3: 4, 5 in queue, transmit 5
- Time 4: 4 in queue, transmit 4
- Time 6: 6, 7 in queue, transmit 7
- Time 7: 6 in queue, transmit 6

Question 4.2-7c Round Robin Scheduling



1 3 2 4 5 7 6

- Assume a round-robin scheduling cycle begins with red packets. i.e., (red, green) in each round.
- Time 0: 1, 2 in queue, transmit 1
  - 1st round of (red, green)
- Time 1: 2, 3 in queue, transmit 3
  - 1st round of (red, green)
- Time 2: 2. 4 in queue, transmit 2
  - 2nd round of (red, green)
- Time 3: 4, 5 in queue, transmit 4
  - 2nd round of (red, green)
- Time 4: 5 in queue, transmit 5
  - 3RD round of (red, green). Since there is no green packet ready, this round is (red, null)
- Time 6: 6, 7 in queue, transmit 7
  - 4th round of (red, green)
- Time 7: 6 in queue, transmit 6
  - 4th round of (red, green)
- Summary:
- Times 0-1: 1st round: (1, 3)
- Times 2-3: 2nd round: (2, 4)
- Time 4: 3rd round: (5, null)
  - No green packets ready
- Times 6-7: 4th round: (7, 6)

Network layer: “data plane” roadmap

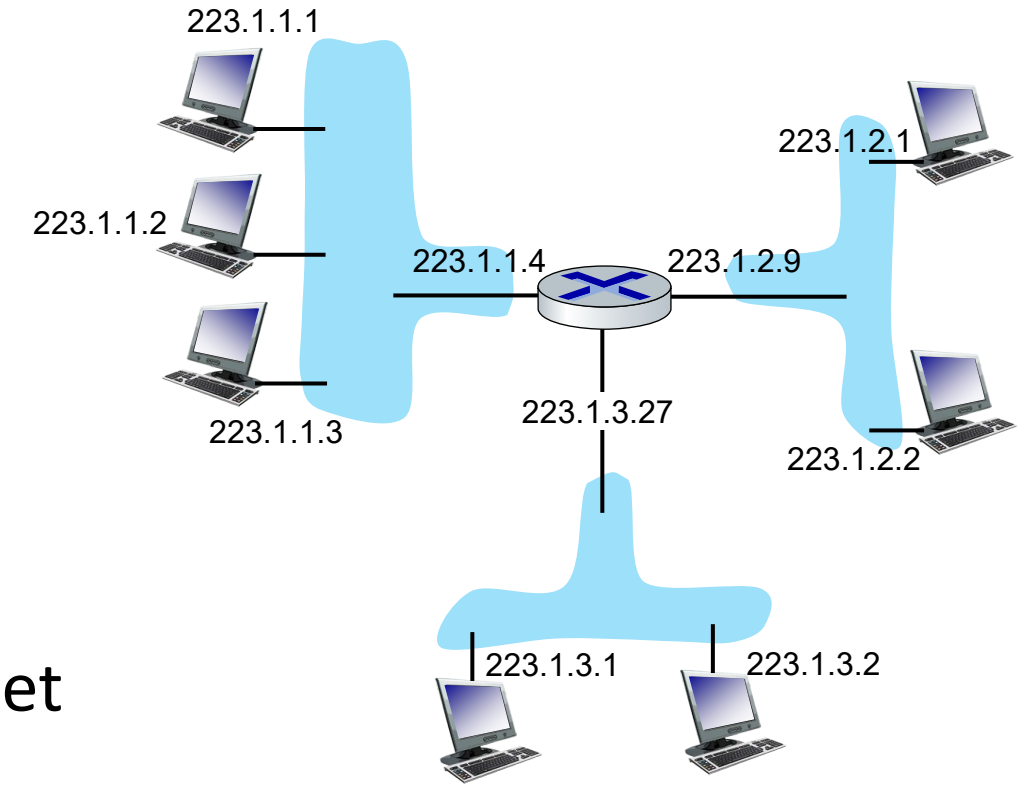
- Network layer: overview
  - data plane
  - control plane
- What’s inside a router
  - input ports, switching, output ports
  - buffer management, scheduling
- IP: the Internet Protocol
  - datagram format
  - addressing
  - network address translation
  - IPv6

- Generalized Forwarding, SDN
  - match+action
  - OpenFlow: match+action in action
- Middleboxes

IMPORTANT

Subnets

- *What's a subnet ?*
  - device interfaces that can physically reach each other without passing through an intervening router
- IP addresses have structure:
  - subnet part: devices in same subnet have common high order bits
  - host part: remaining low order bits



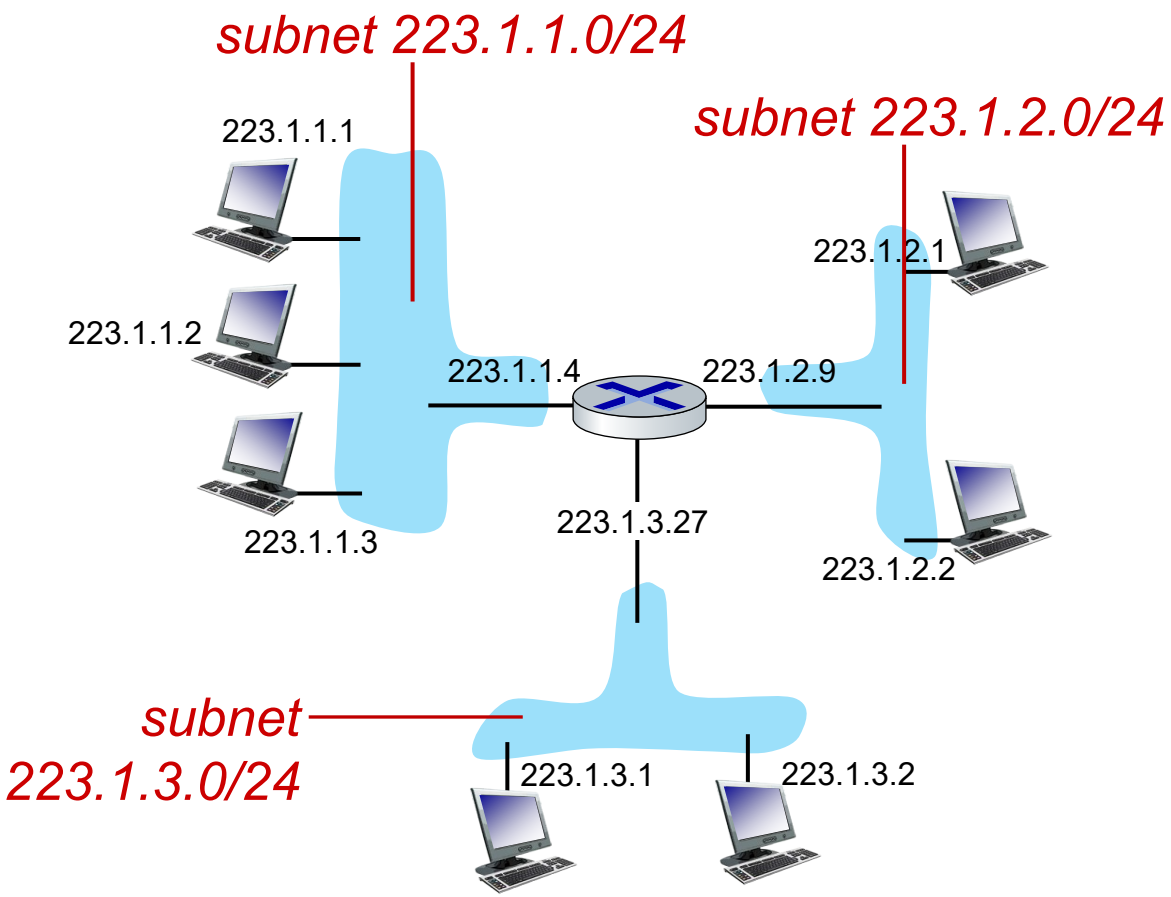
network consisting of 3 subnets

IMPORTANT

Subnets

Recipe for defining subnets:

- detach each interface from its host or router, creating "islands" of isolated networks
- each isolated network is called a *subnet*

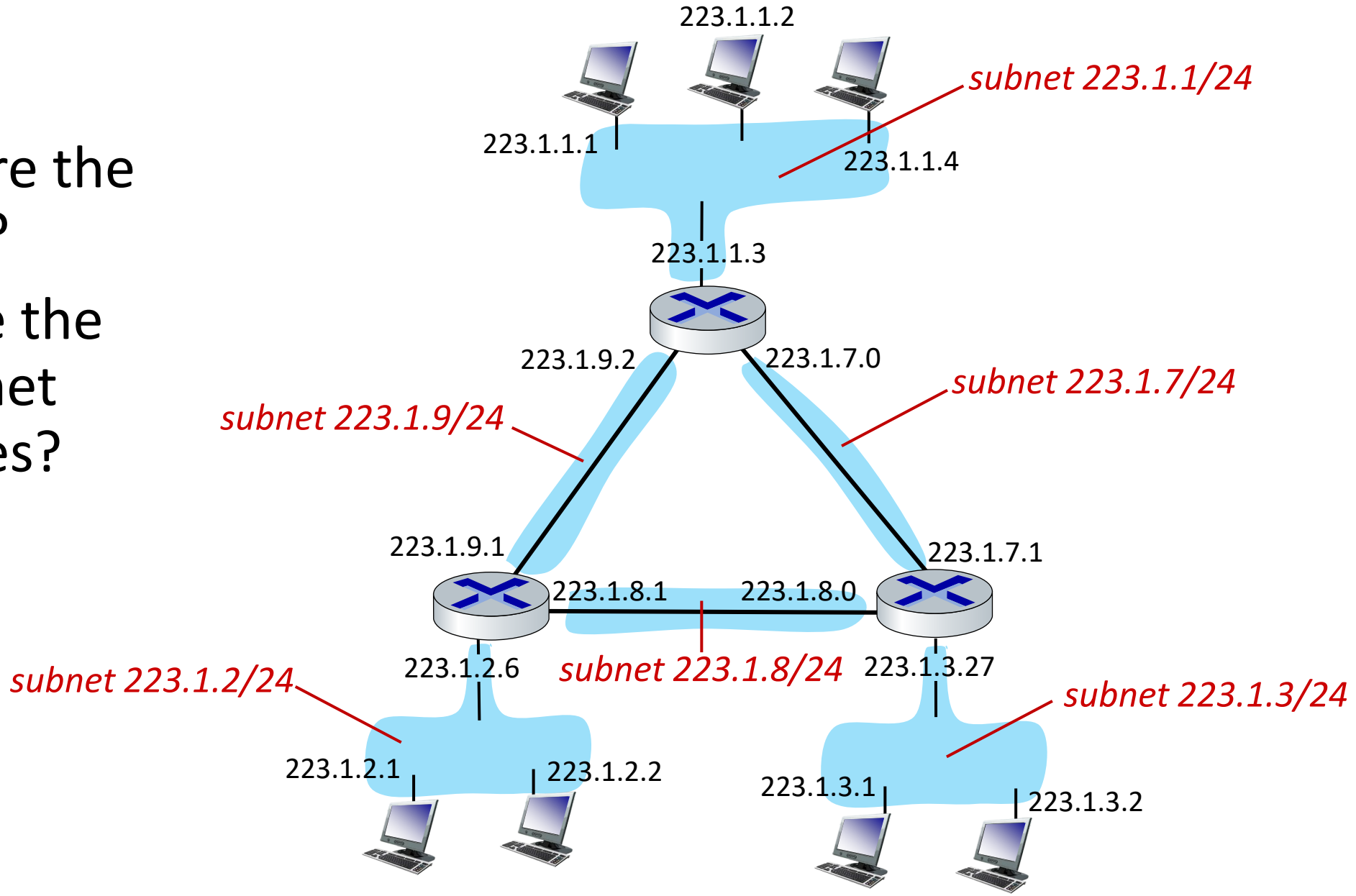


subnet mask: /24
(high-order 24 bits: subnet part of IP address)

IMPORTANT

Subnets

- where are the subnets?
- what are the /24 subnet addresses?

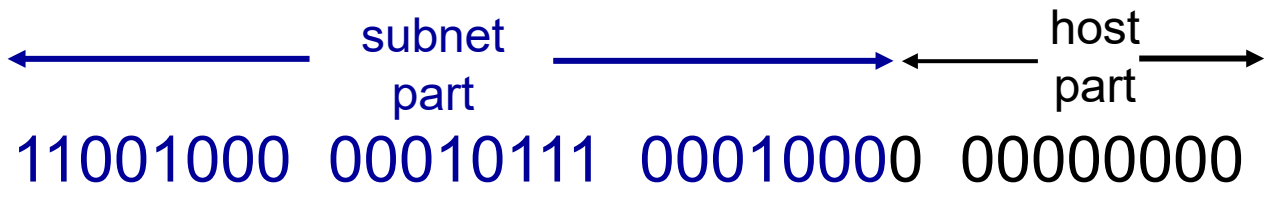


IMPORTANT

IP addressing: CIDR

CIDR: Classless InterDomain Routing (pronounced “cider”)

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



200.23.16.0/23

Question 4.3-04

- 4.3-04. What is a subnet? What is meant by an IP subnet? (Check zero, one or more of the following characteristics of an IP subnet).
  - A A set of device interfaces that can physically reach each other without passing through an intervening router.
    - Correct
  - B A set of devices that always have a common first 16 bits in their IP address.
    - Incorrect as the subnet mask may not be 16 bits
  - C A set of devices that have a common set of leading high order bits in their IP address.
    - Correct
  - D A set of devices all manufactured by the same equipment maker/vendor.
    - Incorrect

Question 4.3-05ab

- 4.3-05. Subnetting(a). Consider the three subnets in the diagram below.
- 1. What is the maximum # of interfaces in the 223.1.2/24 network?
- 2. What is the maximum # of interfaces in the 223.1.3/29 network?

- 1 ANS: In 223.1.2/24

Network, since 32-24

=8 bits for host address,

Hence maximum # of

Interfaces = 2^8=256

- 2 ANS: In 223.1.3/29

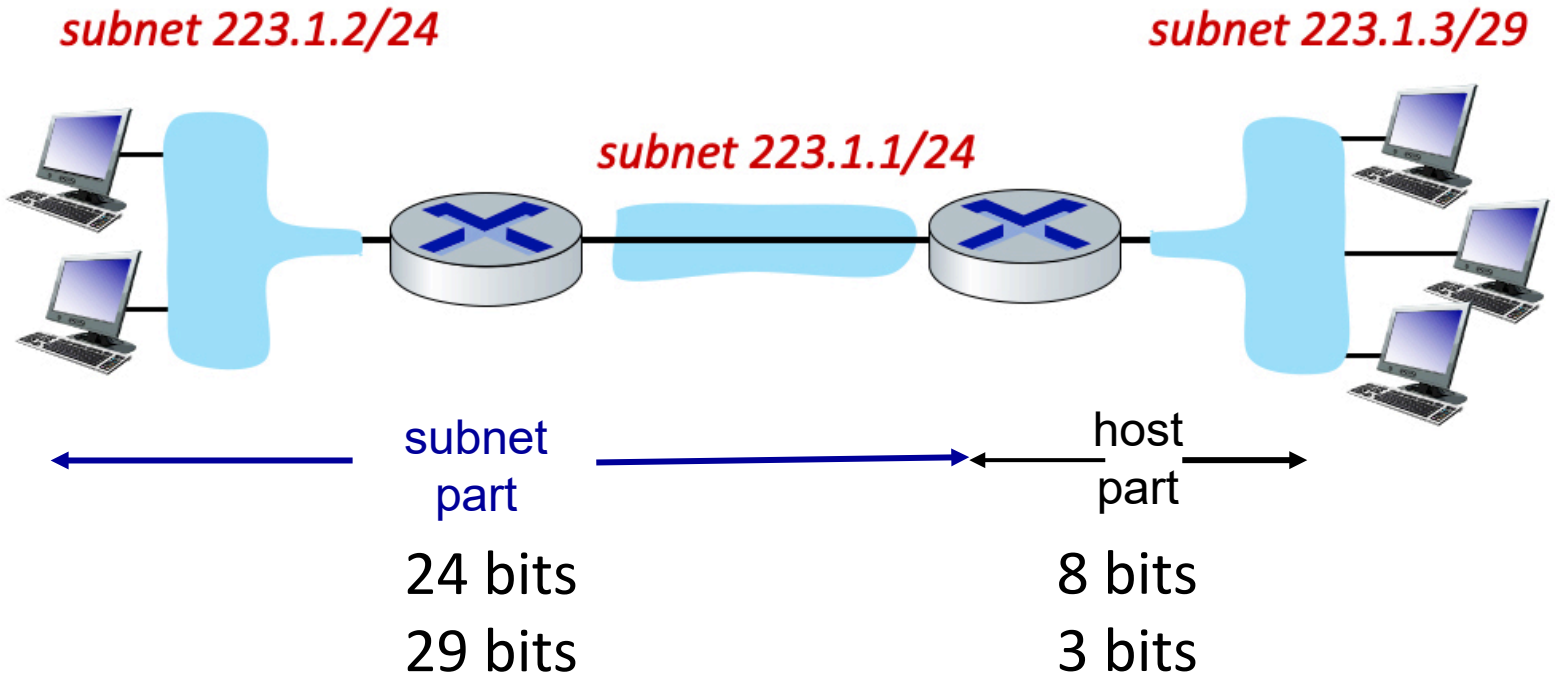
Network, since 32-29

=3 bits for host address,

Hence maximum # of

Interfaces = 2^3=8

3 ANS:



Question 4.3-05c

- 4.3-05. Subnetting(a). Consider the three subnets in the diagram below.
- 3. Which of the following addresses can not be used by an interface in the 223.1.3/29 network? Check all that apply.
  - 223.1.3.6, 223.1.3.2, 223.1.3.16, 223.1.2.6, 223.1.3.28

- 3 ANS: In 223.1.3/29

Network, since 32-29

=3 bits for host address,

Hence maximum # of

Interfaces = 2^3=8, ranging

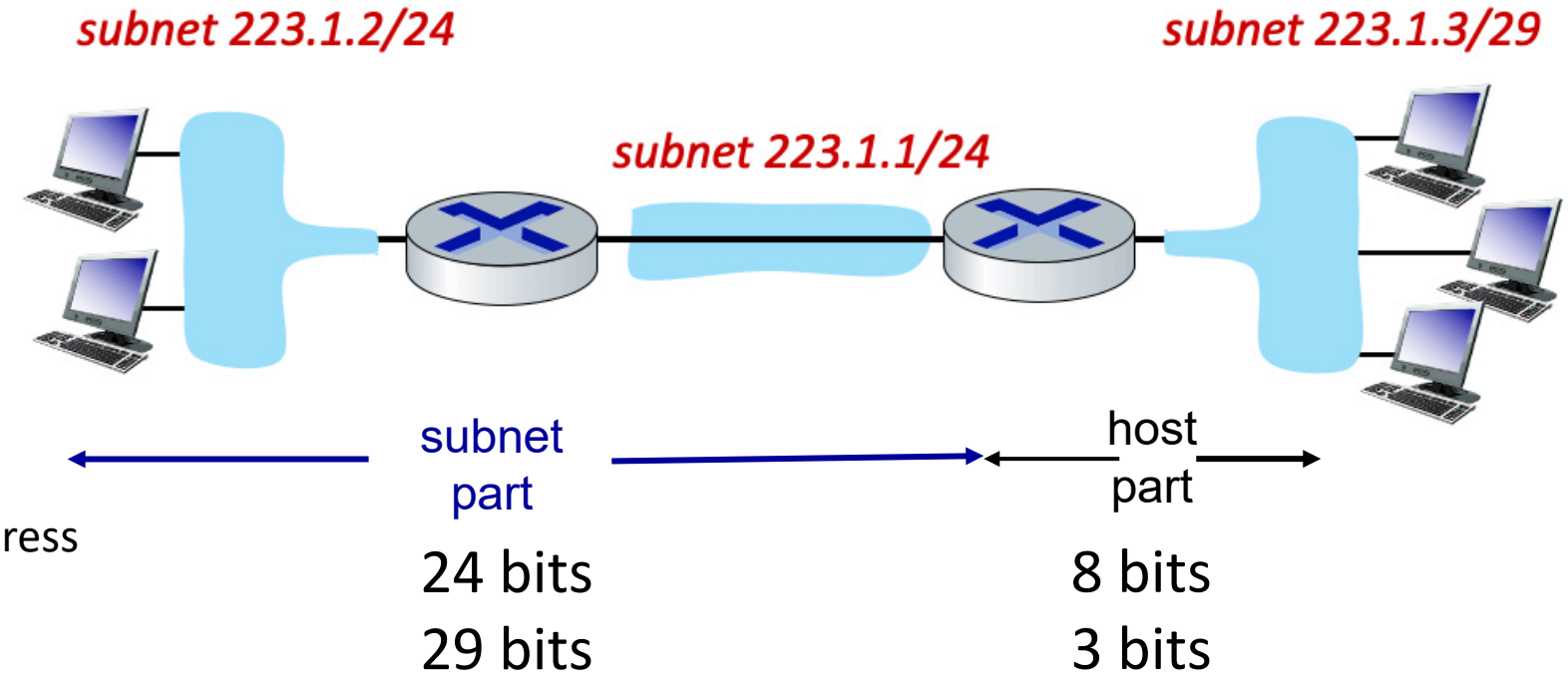
From 0 to 7. Hence

- 223.1.3.6, 223.1.3.2 are OK
- 223.1.3.16, 223.1.3.28 are not OK

since the host ID exceeds 7

- 223.1.2.6 is not OK since network address

223.1.2 does not match 223.1.3



Network layer: “data plane” roadmap

- Network layer: overview
  - data plane
  - control plane
- What’s inside a router
  - input ports, switching, output ports
  - buffer management, scheduling
- IP: the Internet Protocol
  - datagram format
  - addressing
  - network address translation
  - IPv6

- Generalized Forwarding, SDN
  - match+action
  - OpenFlow: match+action in action
- Middleboxes

IMPORTANT

IP addresses: how to get one?

That's actually two questions:

1. Q: How does a *host* get IP address within its network (host part of address)?
2. Q: How does a *network* get IP address for itself (network part of address)

How does *host* get IP address?

- hard-coded by sysadmin in config file (e.g., /etc/rc.config in UNIX)
- DHCP: Dynamic Host Configuration Protocol: dynamically get address from as server
  - "plug-and-play"

IMPORTANT

DHCP: Dynamic Host Configuration Protocol

goal: host *dynamically* obtains IP address from network server when it "joins" network

- can renew its lease on address in use
- allows reuse of addresses (only hold address while connected/on)
- support for mobile users who join/leave network

DHCP overview:

- host broadcasts DHCP discover msg [optional]
- DHCP server responds with DHCP offer msg [optional]
- host requests IP address: DHCP request msg
- DHCP server sends address: DHCP ack msg

Question 4.3-06

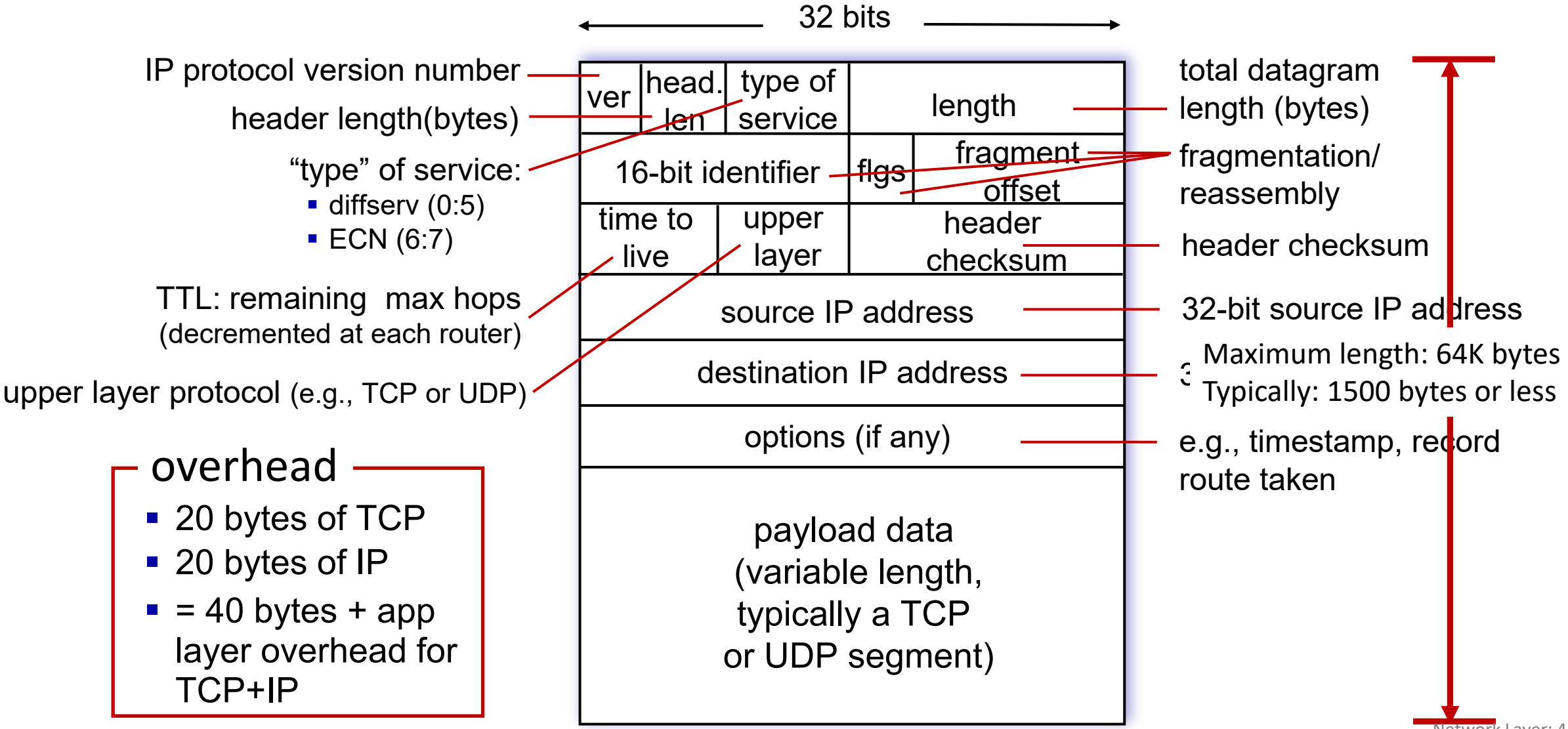
- 4.3-06. Plug-and-play. What is meant by saying that DHCP is a "plug and play" protocol?
- The host needs to “plug” (by wire or wirelessly) into the local network in order to access (“play” in) the Internet
- No manual configuration is needed for the host to join the network., (Correct answer)
- The network provides an Ethernet jack for a host’s Ethernet adapter., (Incorrect answer)

IMPORTANT

IPv6: motivation

- initial motivation: 32-bit IPv4 address space would be completely allocated
- additional motivation:
  - speed processing/forwarding: 40-byte fixed length header
  - enable different network-layer treatment of “flows”

IP Datagram format



overhead

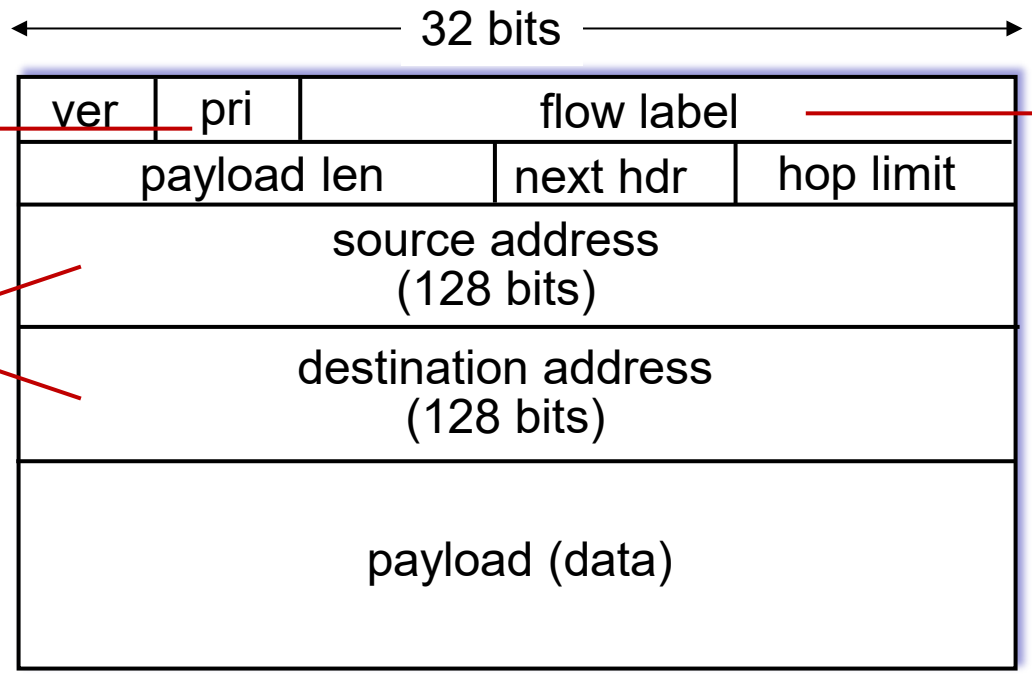
- 20 bytes of TCP
- 20 bytes of IP
- = 40 bytes + app layer overhead for TCP+IP

IMPORTANT

IPv6 datagram format

priority: identify priority among datagrams in flow

128-bit IPv6 addresses



flow label: identify datagrams in same "flow." (concept of "flow" not well defined).

What's missing (compared with IPv4):

- no checksum (to speed processing at routers)
- no fragmentation/reassembly
- no options (available as upper-layer, next-header protocol at router)

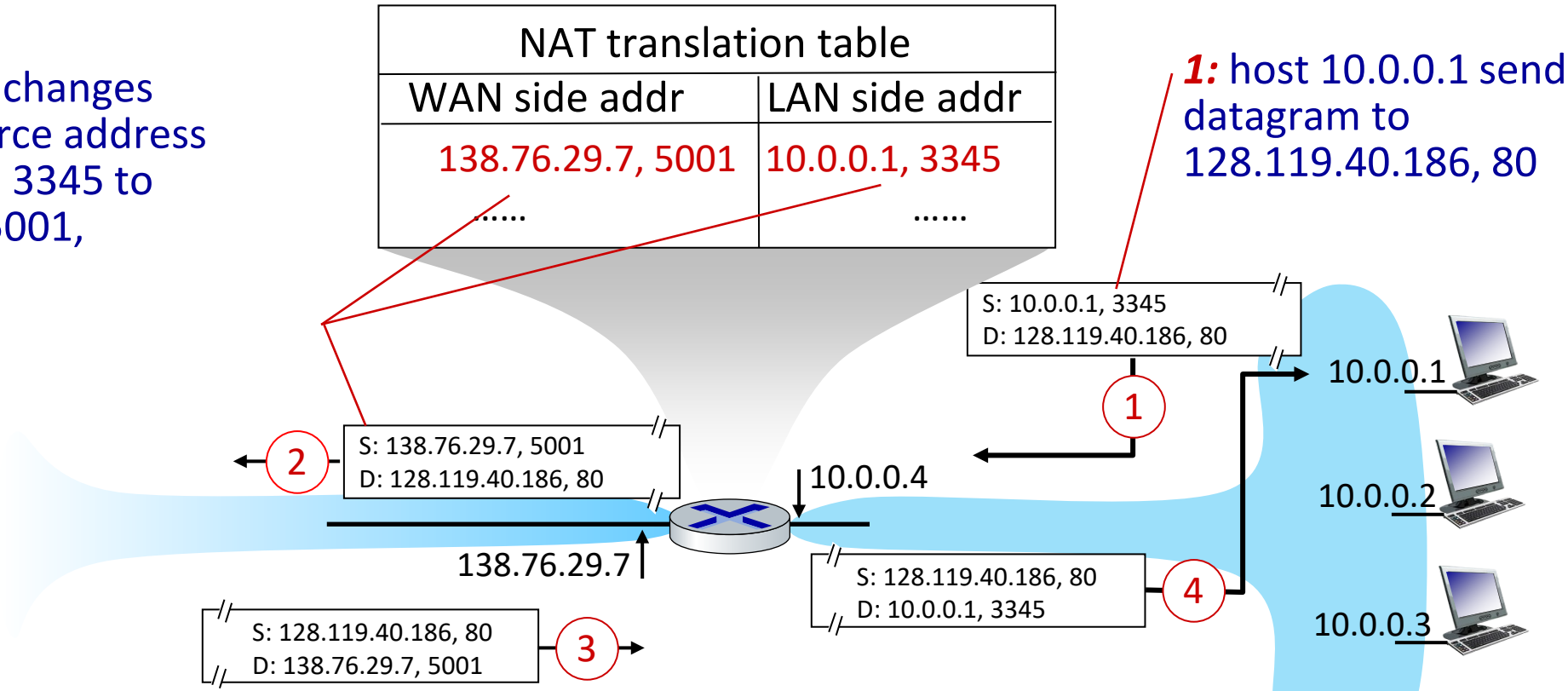
Question 4.3-08

- **4.3-08. IPv4 versus IPv6.** Which of the following fields occur ***ONLY*** in the IPv6 datagram header (i.e., appear in the IPv6 header but not in the IPv4 header)? Check all that apply.
- Correct: 128-bit source and destination IP addresses.
- The IP version number field.
- The time-to-live (or hop limit) field.
- The header checksum field.
- Correct: The flow label field.
- The header length field.
- The options field.
- The upper layer protocol (or next header) field.

IMPORTANT

NAT: network address translation

2: NAT router changes datagram source address from 10.0.0.1, 3345 to 138.76.29.7, 5001, updates table



3: reply arrives, destination address: 138.76.29.7, 5001

Question 4.3.10

- **4.3.10. Network Address Translation (NAT).** Which one of the following operations is *not* performed by NAT.?
- A Generating ACKs back to the TCP sender and then taking responsibility for reliably delivery the segment to its destination, possibly using a non-TCP reliable data transfer protocol.
- B On an outgoing datagram, changing the transport-layer port number of the transport-layer segment inside a datagram received from the LAN side of the NAT.
- C On an incoming datagram from the public Internet side of a NAT, changing the destination IP address of a datagram to a new destination IP address that is looked up in the NAT table, and (possibly after other actions), sending that IP datagram on to the LAN side of the NAT.
- D On an outgoing datagram, changing the source IP address of a datagram received from the LAN side of the NAT
- ANS: A

IMPORTANT

OpenFlow abstraction

- match+action: abstraction unifies different kinds of devices

Router

- *match:* longest destination IP prefix
- *action:* forward out a link

Switch

- *match:* destination MAC address
- *action:* forward or flood

Firewall

- *match*: IP addresses and TCP/UDP port numbers
- *action:* permit or deny

NAT

- *match:* IP address and port
- *action:* rewrite address and port

IMPORTANT

Generalized forwarding: summary

- "match plus action" abstraction: match bits in arriving packet header(s) in any layers, take action
  - matching over many fields (link-, network-, transport-layer)
  - local actions: drop, forward, modify, or send matched packet to controller
  - "program" n*etwork-wide* behaviors
- simple form of "network programmability"
  - programmable, per-packet "processing"
  - *historical roots:* active networking
  - *today:* more generalized programming: P4 (see p4.org).

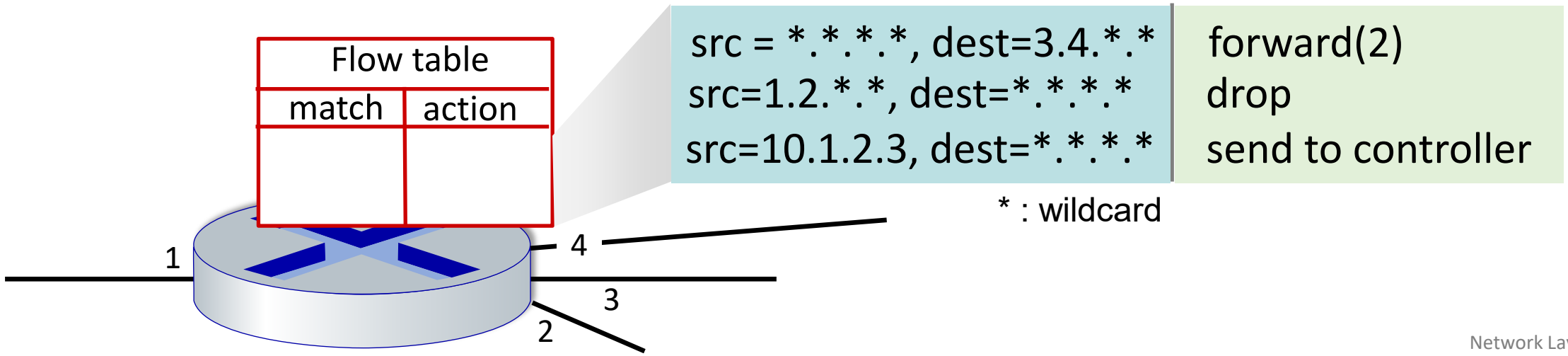
Question 4.4-2

- **4.4-2. Generalized match+action.** Which of the following match+actions can be taken in the generalized OpenFlow 1.0 match+action paradigm that we studied in Section 4.4? Check all that apply.
- (Correct)... after matching on the destination IP address in the datagram header, the action taken is to forward the datagram to the output port associated with that destination IP address.
- (Correct)... after matching on the destination IP address in the datagram header, the action taken is to decide whether or not to drop that datagram.
- (Correct)... after matching on the port number in the segment's header, the action taken is to decide whether or not to drop that datagram containing that segment.
- (Correct)... after matching on the port number in the segment's header, the action taken is to forward the datagram to the output port associated with that destination IP address.
- (Correct)... after matching on the 48-bit link-layer destination MAC address, the action taken is to forward the datagram to the output port associated with that link-layer address.
- (InCorrect)... after matching on the URL contained in an HTTP GET request in the TCP segment within the IP datagram, the action taken is to determine the IP address of the server associated with that URL, and to forward the datagram to the output port associated with that destination IP address.
  - At the IP layer, cannot see any application layer information such as the URL contained in an HTTP GET request
- (Correct)... after matching on the source and destination IP address in the datagram header, the action taken is to forward the datagram to the output port associated with that source and destination IP address pair.

IMPORTANT

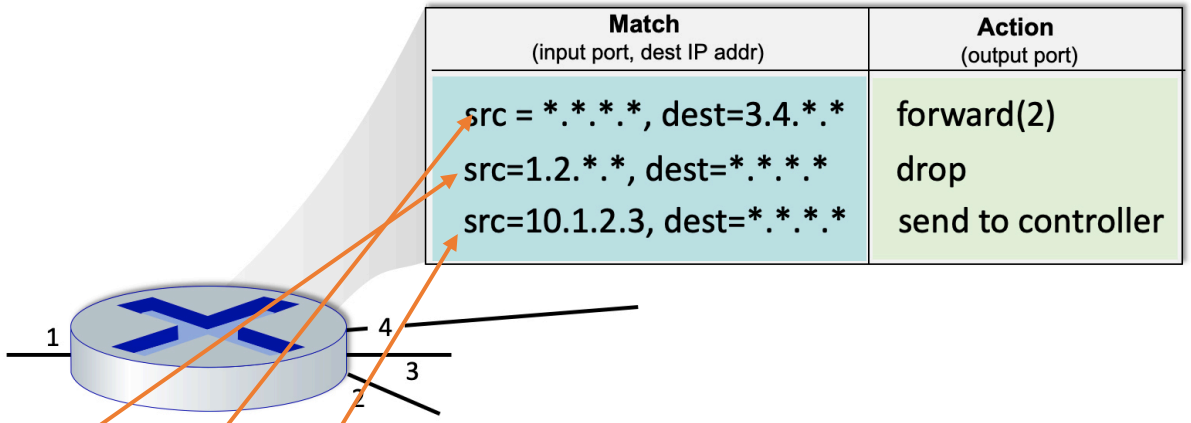
Flow table abstraction

- flow: defined by header fields
- generalized forwarding: simple packet-handling rules
  - match: pattern values in packet header fields
  - actions: for matched packet: drop, forward, modify, matched packet or send matched packet to controller
  - priority: disambiguate overlapping patterns
  - counters: #bytes and #packets



Question 4.4-4

- **4.4-4. Match+action in Openflow 1.0.** Consider the figure below that shows the generalized forwarding table in a router. Recall that a * represents a wildcard value. Now consider an arriving datagram with the IP source and destination address fields indicated below. For each source/destination IP address pair, indicate which rule is matched. Note: assume that a rule that is earlier in the table takes priority over a rule that is later in the table and that a datagram that matches none of the table entries is dropped.



Source: 1.2.56.32 Destination:128.116.40.186	Rule 2, with action drop
...	
Source: 65.92.15.27 Destination: 3.4.65.76	Rule 1, with action forward(2)
...	
Source: 10.1.2.3 Destination: 7.8.9.2	Rule 3, with action send to controller
...	
Source: 10.1.34.56 Destination: 54.72.29.90	Nomatch to any rule.

Network layer: “data plane” roadmap

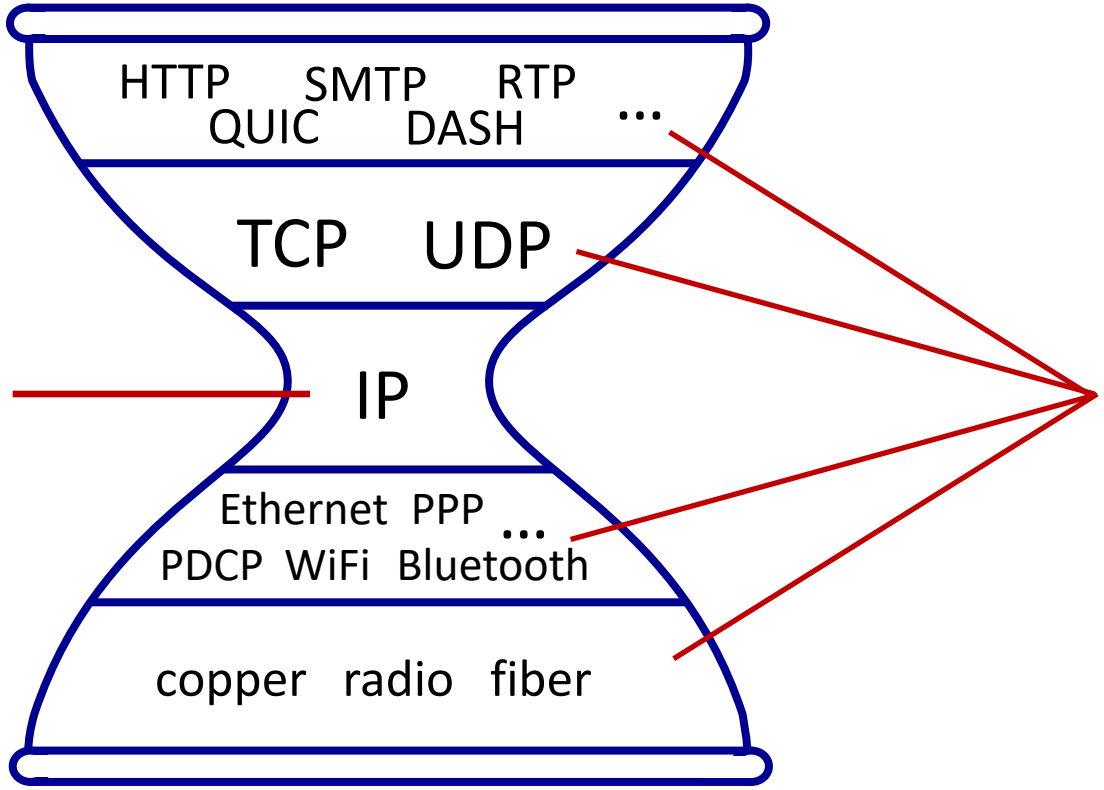
- Network layer: overview
- What’s inside a router
- IP: the Internet Protocol
- Generalized Forwarding
- Middleboxes
  - middlebox functions
  - evolution, architectural principles of the Internet

IMPORTANT

The IP hourglass

Internet's "thin waist":

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



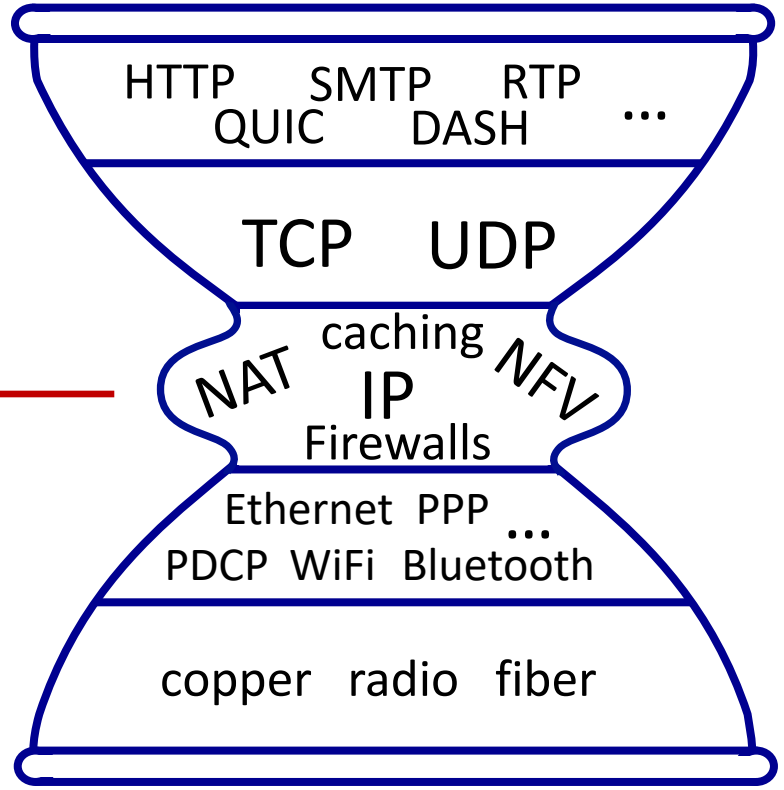
many protocols in physical, link, transport, and application layers

IMPORTANT

The IP hourglass, at middle age

Internet's middle age "love handles"?

- middleboxes, operating inside the network



Question 4.5-2

- **4.5-2. The "thin waist" of the Internet.** What protocol (or protocols) constitutes the "thin waist" of the Internet protocol stack?
- ANS: IP