CS 305: Computer Networks

Fall 2022

Lecture 15: Summary and Review

Ming Tang

Department of Computer Science and Engineering
Southern University of Science and Technology (SUSTech)

Chapter 3 outline

3.1 transport-layer services
3.2 multiplexing and demultiplexing
3.3 connectionless transport: UDP
3.4 principles of reliable data transfer
3.5 connection-oriented transport: TCP
- segment structure
- reliable data transfer
- flow control
- connection management

3.6 principles of congestion control
3.7 TCP congestion control

Transport services and protocols

- Provide *logical communication* between app processes running on different hosts (how about network layer protocol?)
- Transport protocols run in end systems (how about network layer protocol?)
  - send side: breaks app messages into *segments*, passes to network layer (Multiplexing?)
  - rcv side: reassembles segments into messages, passes to app layer (demultiplexing?)

Multiplexing at sender:
handle data from multiple sockets, add transport header (later used for demultiplexing)

Demultiplexing at receiver:
use header info to deliver received segments to correct socket

- sockets have unique identifiers
- each segment have special fields that indicate the socket to which the segment is to be delivered.

Host uses *IP addresses & port numbers* to direct segment to appropriate socket

Transport vs. network layer

- The services that a transport protocol can provide are often constrained by the underlying network-layer protocol.
  - delay or bandwidth guarantees
- Certain services can be offered by a transport protocol even when the underlying network protocol doesn't offer the corresponding service at the network layer.
  - Reliable data transfer; security

Network layer: Internet protocol (IP) is a best effort delivery service, unreliable

UDP: unreliable, unordered delivery

TCP: reliable, in-order delivery

- congestion control
- slow control
- connection setup

Services not available:

- delay, bandwidth guarantees

UDP: Connectionless demux

UDP socket is fully identified by **a two-tuple** consisting of a destination IP address and a destination port number.

when host receives UDP segment:

- directs UDP segment to socket with that port #

IP datagrams with *same dest. port #,* but different source IP addresses and/or source port numbers will be directed to *same socket* at destination

TCP socket identified by 4-tuple:

- source IP address, source port number, dest IP address, dest port number

- Demux: receiver uses all four values to direct segment to appropriate socket
- Server host may support many simultaneous TCP sockets:
  - each socket identified by its own 4-tuple

Connectionless transport: UDP

- "No frills," "bare bones" Internet transport protocol
  - Multiplexing/demultiplexing; light error checking (UDP header?)
- "Best effort" service, UDP segments may be:
  - Lost, delivered out-of-order to app
- *connectionless:*
  - No handshaking between UDP sender, receiver
  - Each UDP segment handled independently of others
  - No congestion control

Advantage?

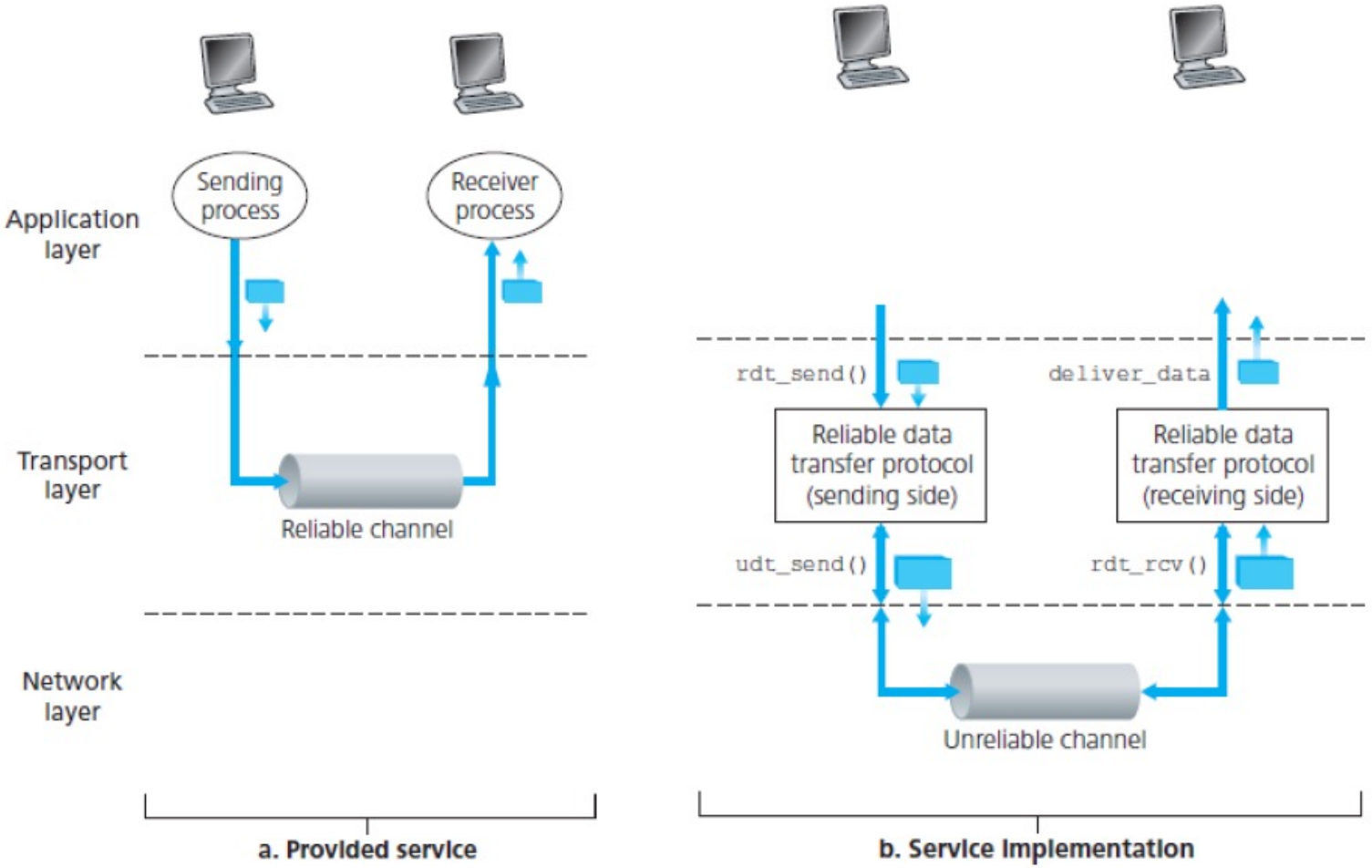
- No congestion control: Immediately pass the segment to network layer
- No connection-establish delay
- No connection state: server can support more clients
- Smaller packet overhead

Disadvantage?

- No congestion control: congestion, overflow, fairness
- Not reliable

reliable transfer over UDP:
add reliability at application layer
application-specific error recovery!

Reliable Data Transfer (rdt)



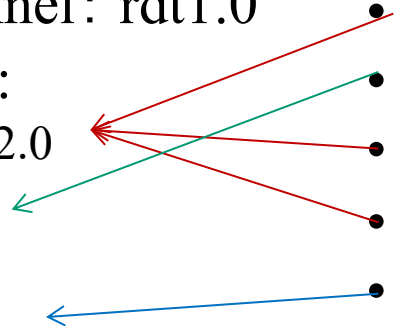
Reliable Data Transfer (rdt)

Roadmap:

- Perfectly reliable channel: rdt1.0
- Channel with bit error:
  - bit error in packet: rdt 2.0
  - bit error in ACK: 2.1
  - NAK-free: 2.2
- Lossy channel: rdt 3.0

Summary of Techniques

- Checksum
- Sequence number
- ACK packets
- Retransmission
- Timeout



You do not need to remember the details of each rdt protocols

But you do need to know how to read a finite state machine (FSM), and based on the FSM, be able to figure out

- The problems (e.g., bit error, lossy channel) that the FSM can address
- Potential problems of the FSM (e.g., cannot handle corrupted ACKs)

Piplined Protocols

Pipelined protocols (Why? How? How to compute utilization?)

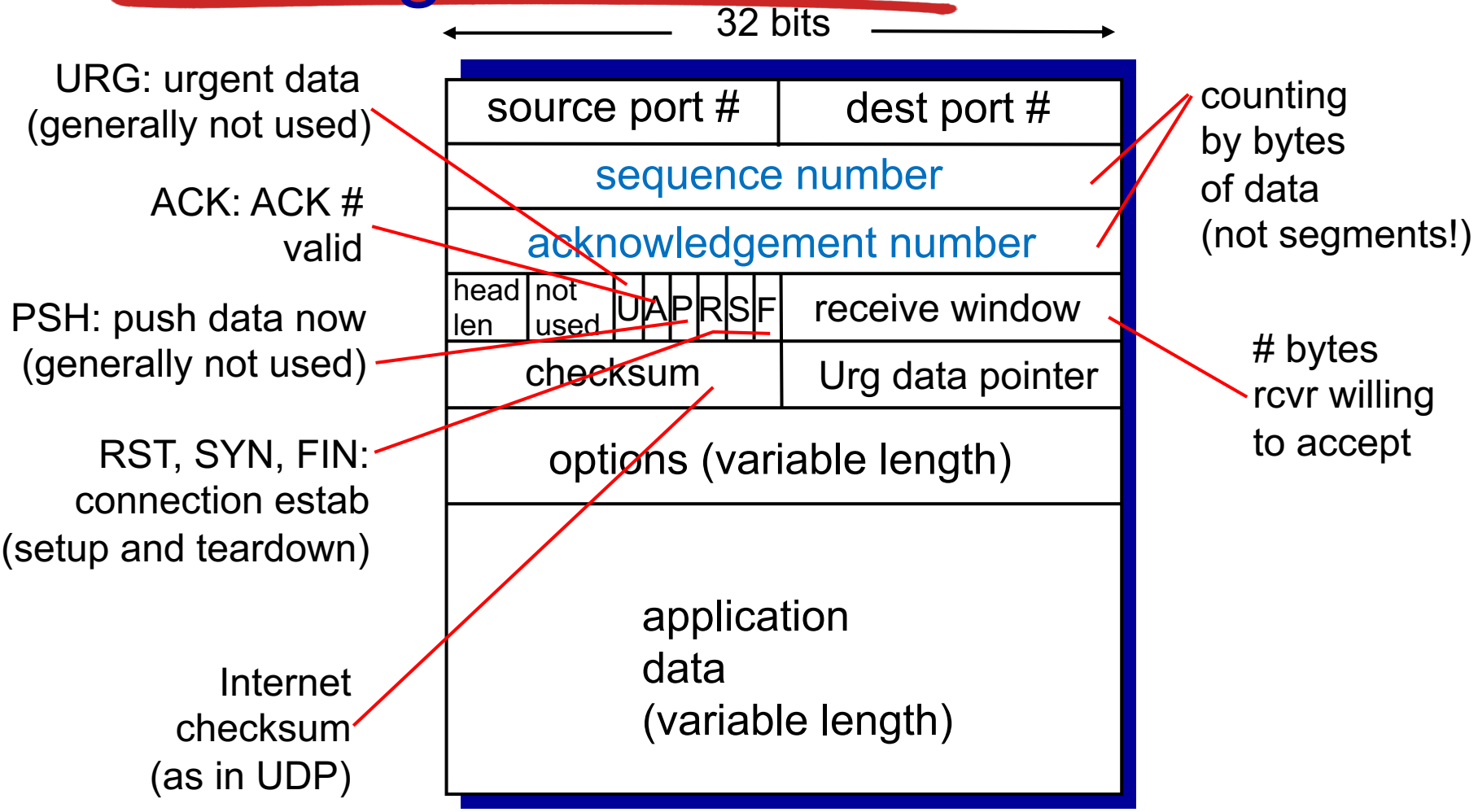
- Go-Back-N
  - Timer for the oldest unACKed packet
  - Cumulative ACK
  - Retransmit all packets in the window
- Selective repeat
  - Timer for each packet in window
  - Individual ACK for each correctly received packets
  - Retransmit only those packets that might be lost or corrupted

Selective repeat dilemma: The window size must be less than or equal to half the size of the sequence number space for SR protocols.

TCP: Overview

- point-to-point:
  - one sender, one receiver
  - No buffers or variables are allocated to network elements
- reliable, in-order *byte stream:*
  - no "message boundaries"
  - Seq # and Ack # are in unit of byte, rather than pkt
  - Seq #: byte stream "number" of first byte in segment's data
  - Ack #: seq # of next byte expected from other side
  - The case study of Telnet? Does a segment without data has a Seq #?
- pipelined:
  - TCP congestion and flow control set window size
- connection-oriented:
  - handshaking (exchange of control msgs) initiates sender and receiver state before data exchange
- flow control (how?) and congestion control (how?) (difference?)

TCP segment structure



When a host receives a TCP segment whose port numbers or source IP address do not match with any of the ongoing sockets.

- RST flag bit is set to 1.

TCP reliable data transfer

- TCP creates rdt service on top of IP's unreliable service
  - pipelined segments: window size, SendBase
  - cumulative acks
  - single retransmission timer (for oldest unacked segment)
- retransmissions triggered by:
  - timeout events
  - Three duplicate acks (fast retransmit)

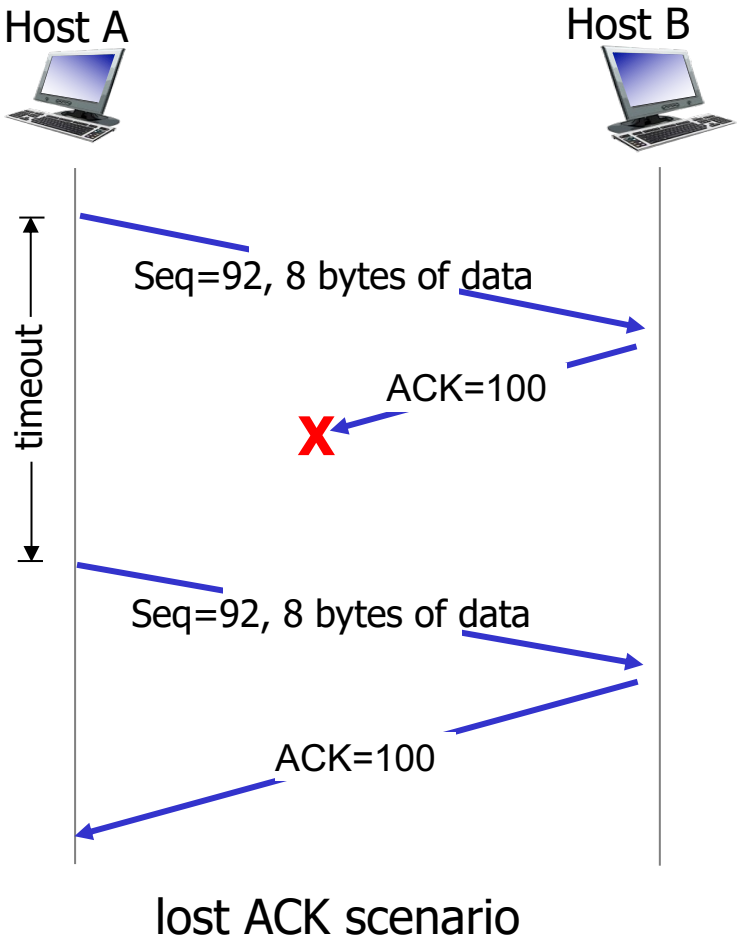
Q: How to set TCP timeout value?
A: longer than RTT

TCP reliable data transfer

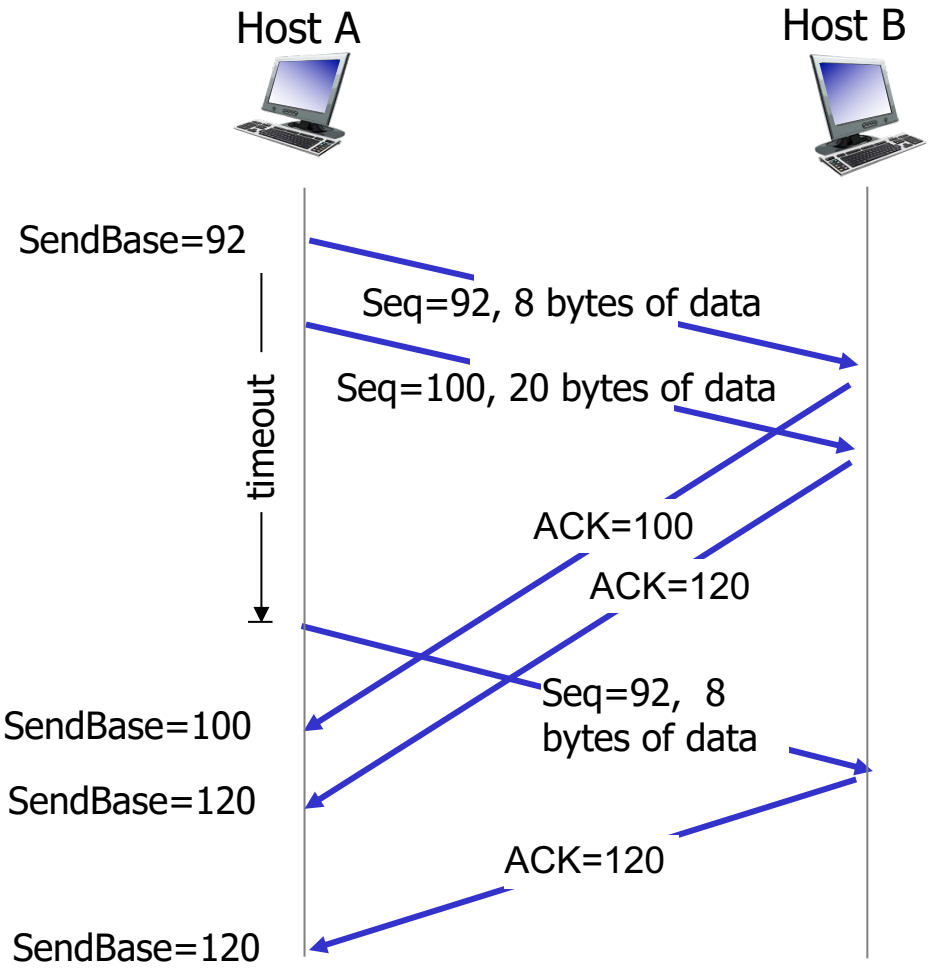
```
NextSeqNum=InitialSeqNumber
SendBase=InitialSeqNumber

loop (forever) {
    switch(event)

        event: data received from applicatio
            create TCP segment with sequence
            if (timer currently not running)
                start timer
            pass segment to IP
            NextSeqNum=NextSeqNum+length(data)
            break;

        event: timer timeout
            retransmit not-yet-acknowledged segment
                smallest sequence number
            start timer
            break;

        event: ACK received, with ACK field value of y
            if (y > SendBase) {
                SendBase=y
                if (there are currently any not-yet-acknowledged segments)
                    start timer
                }
            break;

    } /* end of loop forever */
```

```
event: ACK received, with ACK field value of y
            if (y > SendBase) {
            SendBase=y
            if (there are currently any not yet
                        acknowledged segments)
               start timer
            }
            else {/* a duplicate ACK for already ACKed
                    segment */
               increment number of duplicate ACKs
                    received for y
               if (number of duplicate ACKS received
                    for y==3)
                    /* TCP fast retransmit */
                    resend segment with sequence number y
               }
          break;
```

TCP: retransmission scenarios

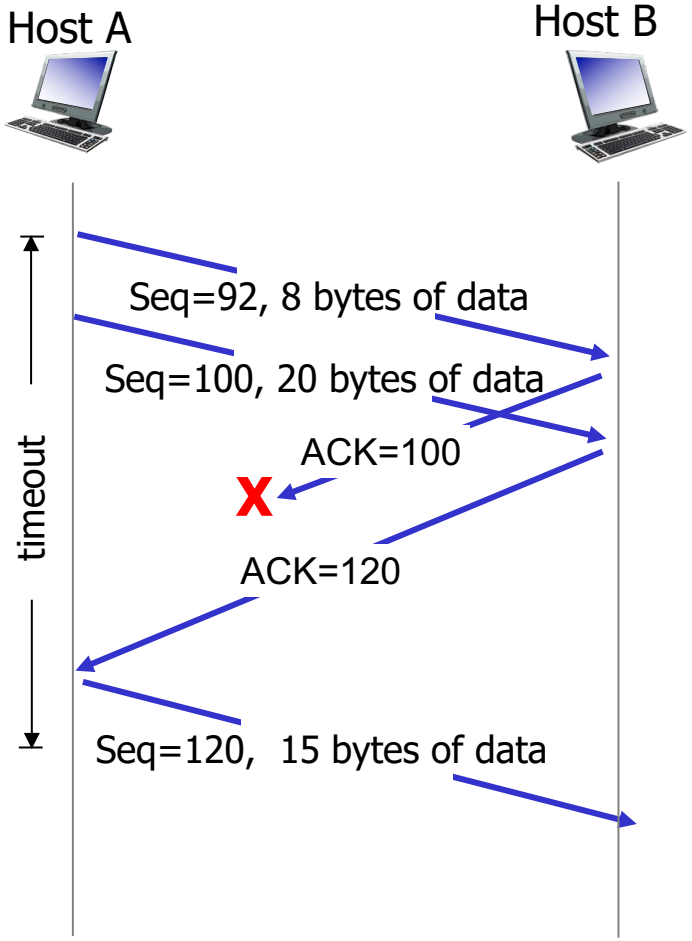


lost ACK scenario

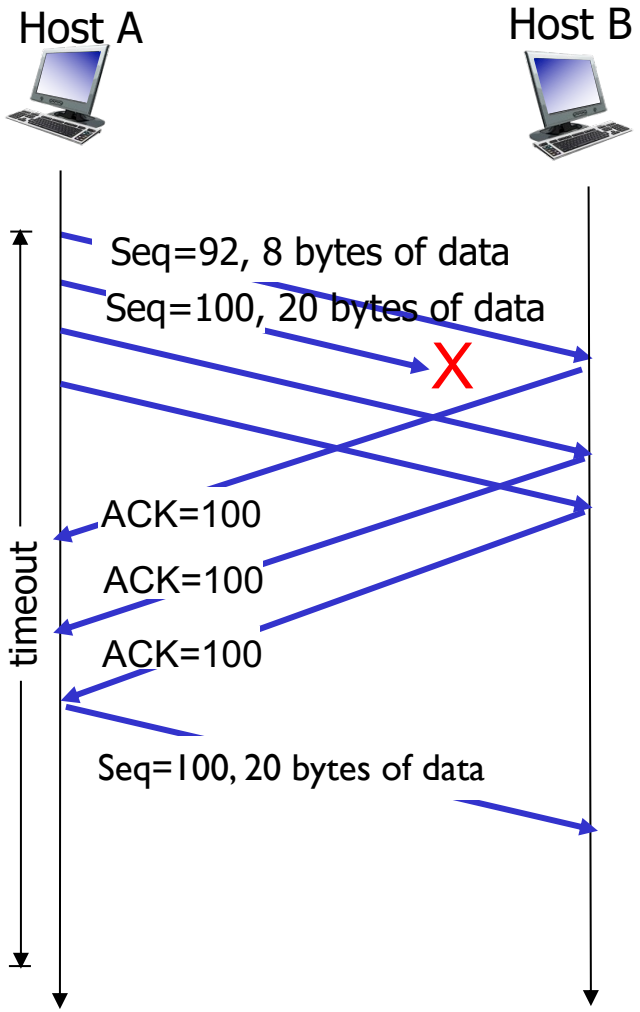


premature timeout

TCP: retransmission scenarios

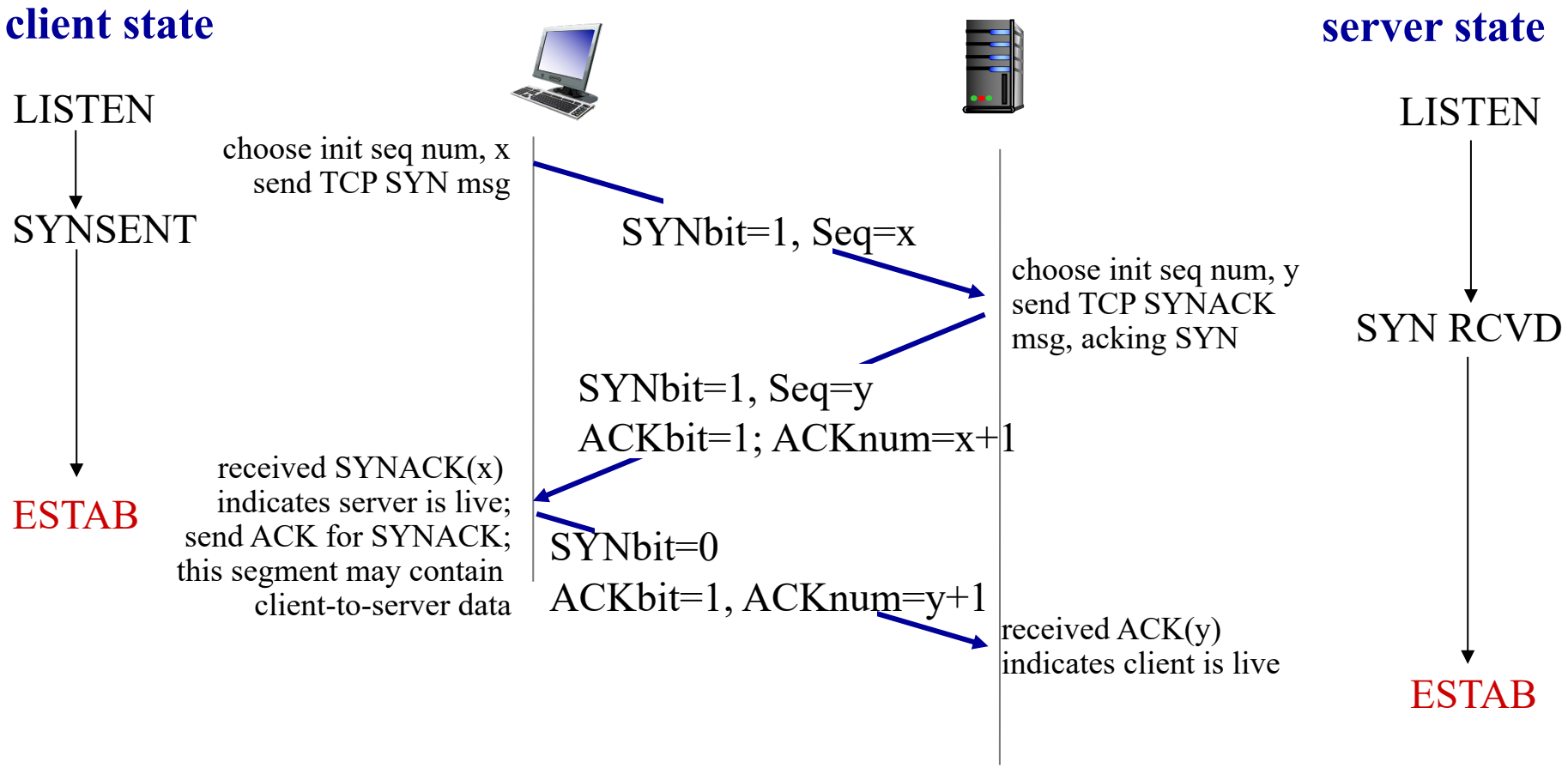


cumulative ACK



fast retransmit after sender receipt of triple duplicate ACK

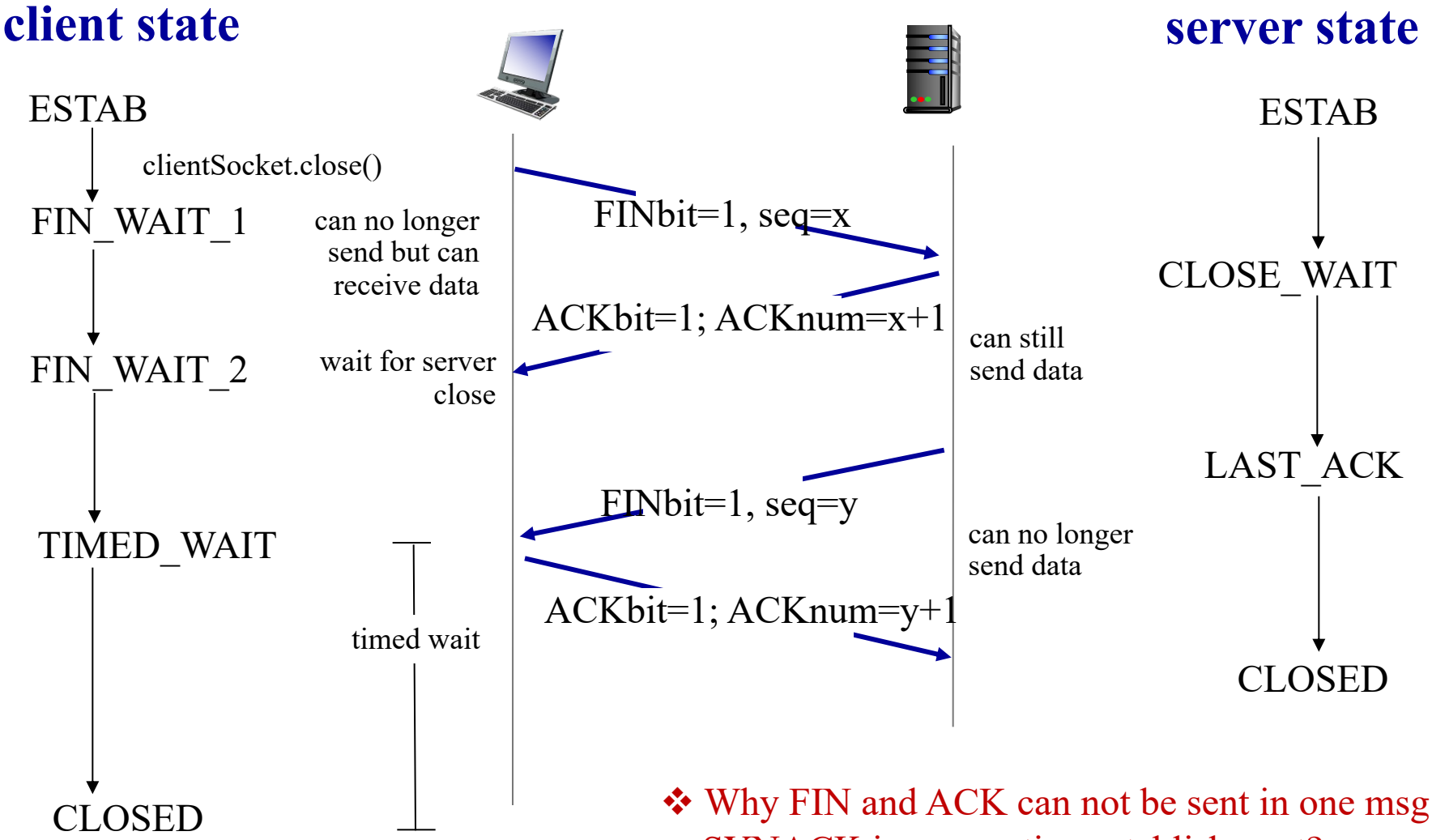
TCP 3-way handshake



Once these three steps have been completed, the client and server hosts can send segments containing data to each other.

- In each of these future segments, SYNbit=0

TCP: closing a connection



- Why FIN and ACK can not be sent in one msg as SYNACK in connection establishment?
- Why TIME_WAIT?

Cause and Cost of Congestion

Cause

- Shared link; limited link capacity
- Sending at a high rate

Cost of Congestion

- Delay and packet lost
- Retransmission
- Unneeded retransmission: waste
- "upstream" transmission capacity was wasted

The four Cause/Cost of Congestion scenarios

End-to-end congestion control:

- TCP segment loss or round-trip segment delay
- TCP decreases its window size accordingly

Network-assisted congestion control:

- routers provide feedback to the sender and/or receiver
- a single bit indicating congestion at a link; the maximum host sending rate the router can support

TCP congestion control

Questions for achieving congestion control:

Q1: How does a TCP sender limit the rate at which it sends traffic into its connection?

Congestion window: **cwnd**

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

LastByteSent – LastByteAcked ≤ min{cwnd, rwnd}

Q2: How does a TCP sender perceive that there is congestion on the path between itself and the destination?

Timeout; three duplicate ACKs

TCP congestion control

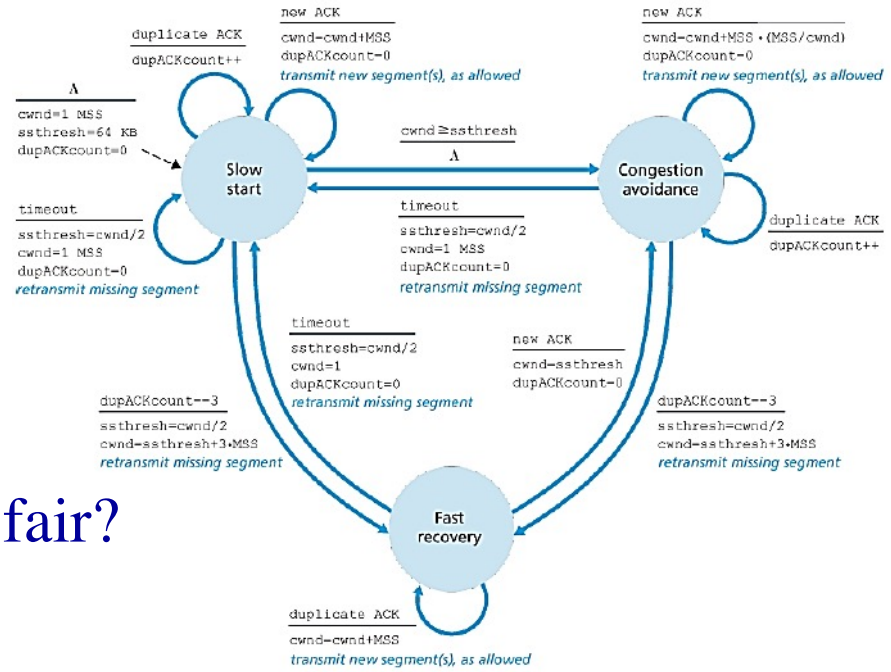
Q3: What algorithm should the sender use to change its send rate as a function of perceived end-to-end congestion?

- A lost segment → congestion → decrease rate
- An acknowledged segment → the network is fine → increase rate
- Bandwidth probing: network condition may change
- additive increase multiplicative decrease

$$\text{avg TCP thruput} = \frac{3}{4}\frac{W}{RTT} \text{ bytes/sec}$$

Finite state machine

- Slow start
- Congestion avoidance
- Fast recovery



Why is TCP fair? Why is TCP not fair?
UDP v.s. fairness?

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

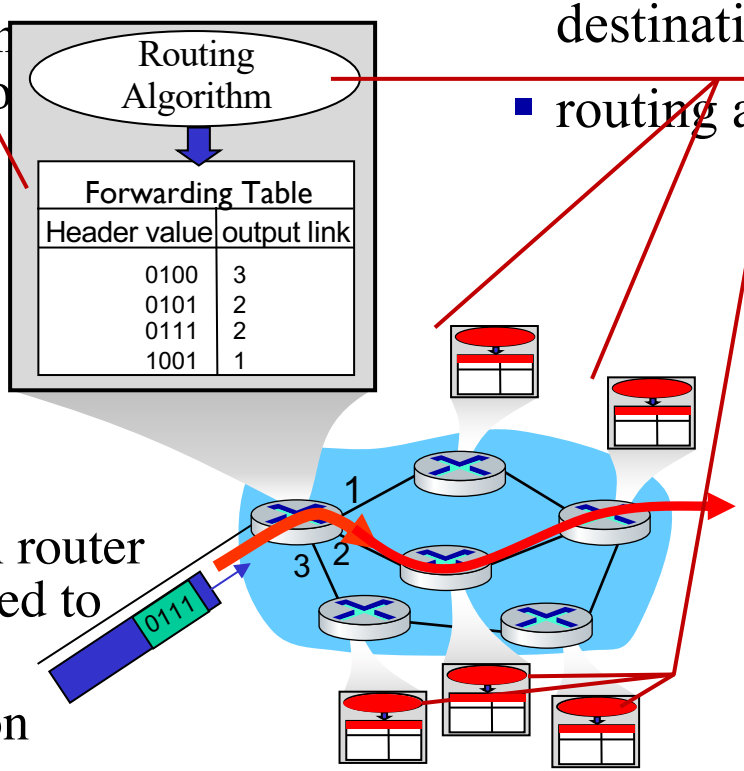
Network layer: data plane, control plane

Forwarding:

local action: move arriving packets from router's input link to appropriate router output link

Routing:

- global action: determine source-destination paths taken by packets
- routing algorithms



Data plane

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

Control plane

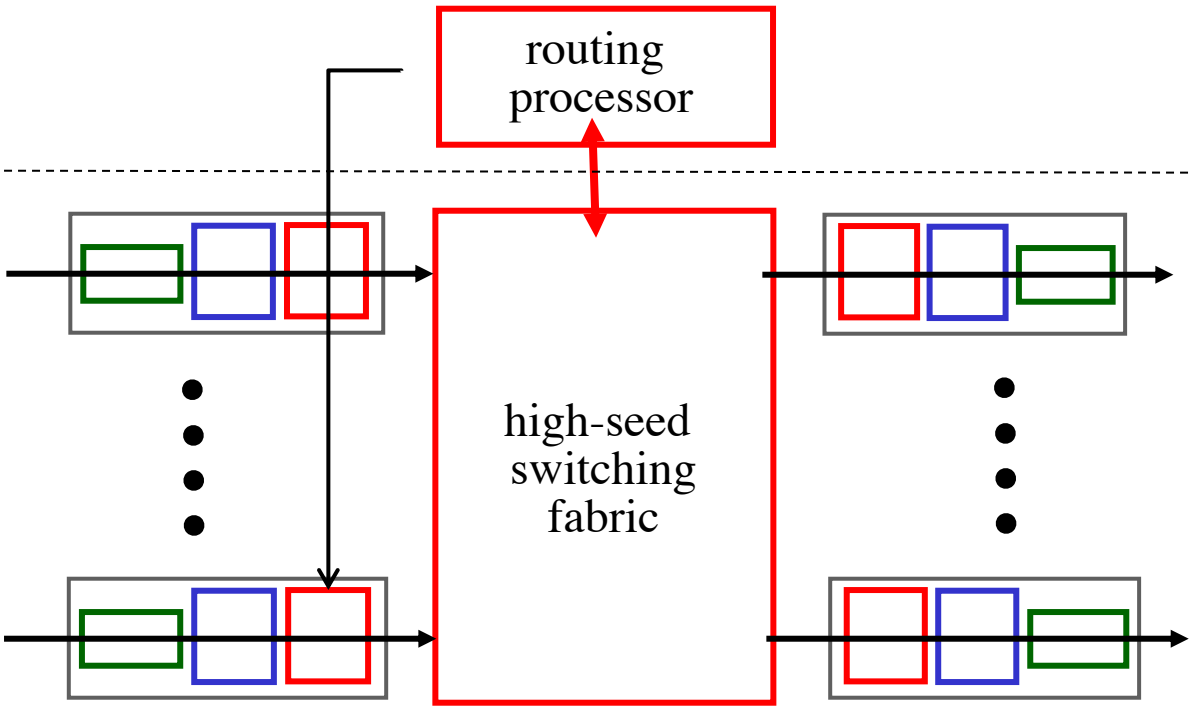
- network-wide logic, softward
- determines how datagram is routed among routers along end-end path
- *traditional routing algorithms*: in routers
- *software-defined networking (SDN)*: in (remote) servers

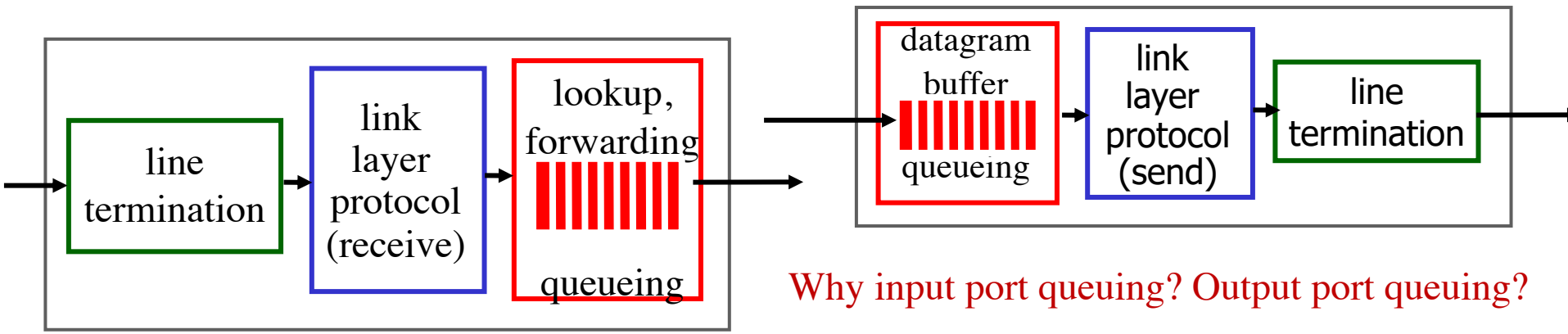
Internet service model provide "best effort" service, no guarantee on bandwidth, loss, order or timing.

Router architecture

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

forwarding data plane (hardware) operates in nanosecond timeframe





Why input port queuing? Output port queuing?

Destination-based forwarding

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

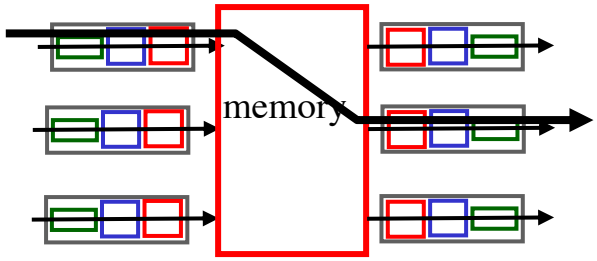
examples:

DA: 11001000 00010111 00010110 10100001 which interface?

DA: 11001000 00010111 00011000 10101010 which interface?

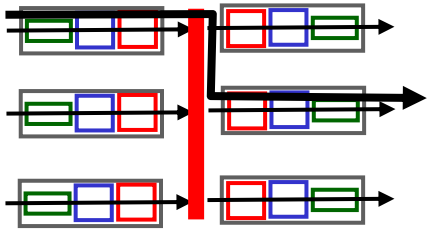
Switching fabrics and output ports

- switching rate: rate at which packets can be transferred from inputs to outputs
  - often measured as multiple of input/output line rate
  - N inputs: switching rate N times line rate desirable



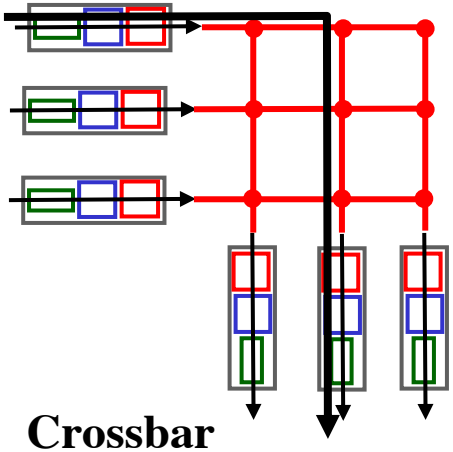
Switch via memory

- Interrupt; write and read
- Two packets cannot be forwarded at the same time



Switch via bus

- Broadcast; label
- One packet can cross at a time

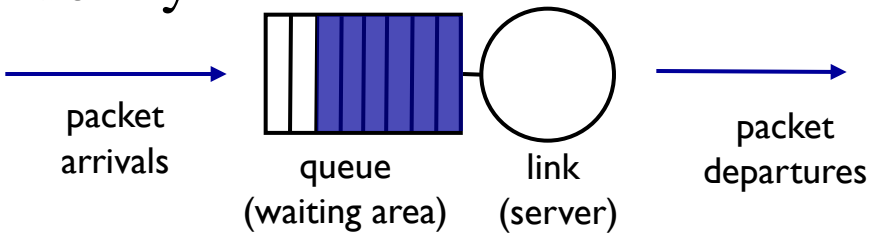


Crossbar

- Multiple packets in parallel
- Non-blocking

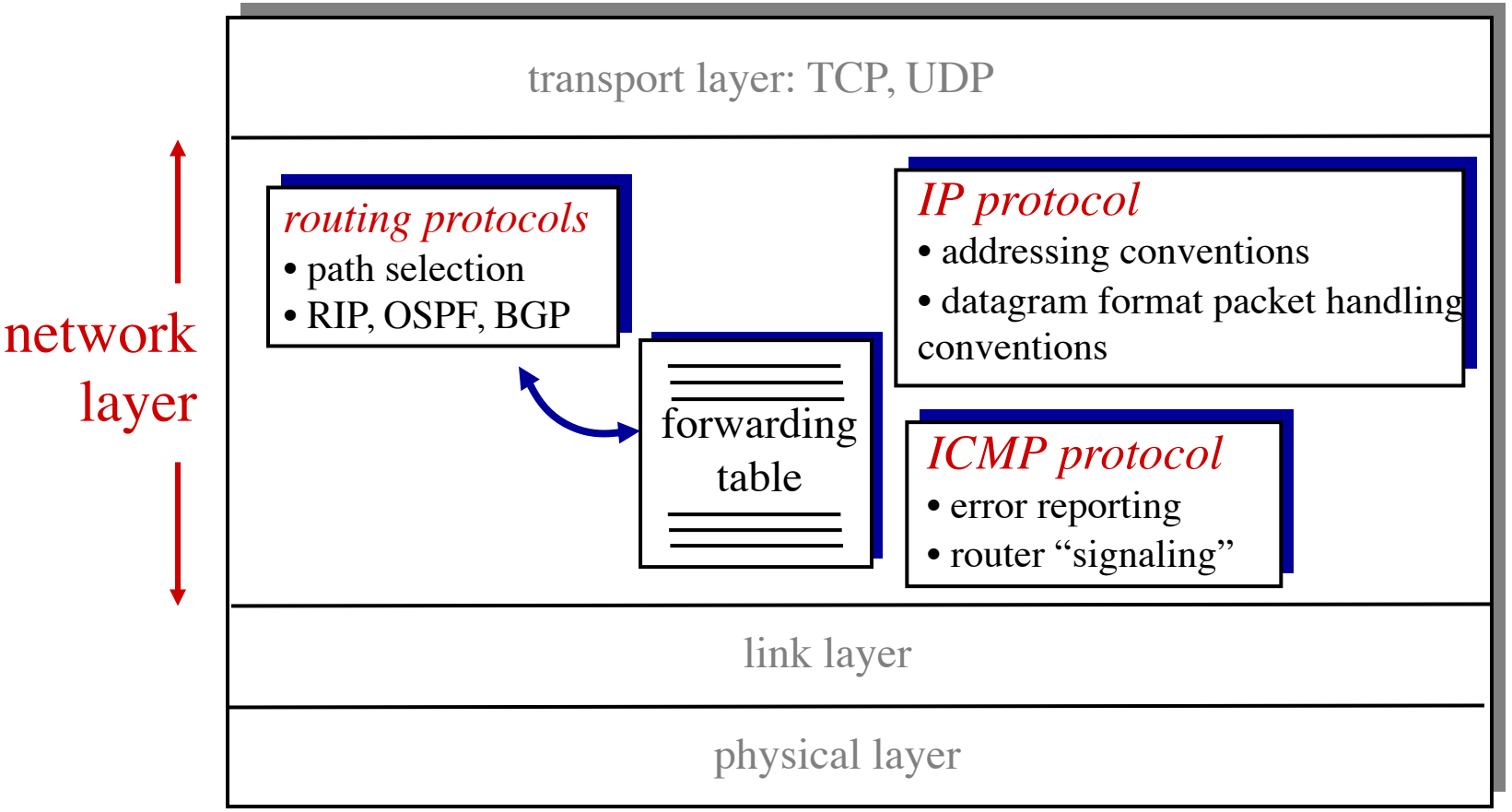
Scheduling mechanisms

- *Scheduling at one output port:* choose next packet to send on link
  - Priority queuing; non-preemptive
  - Round robin scheduling
  - Weighted fair queuing (how to compute the throughput?)
- *FIFO (first in first out) queuing:* send in order of arrival to queue
- *discard policy:* if packet arrives to full queue: who to discard?
  - *tail drop:* drop arriving packet
  - *priority:* drop/remove on priority basis
  - *random:* drop/remove randomly



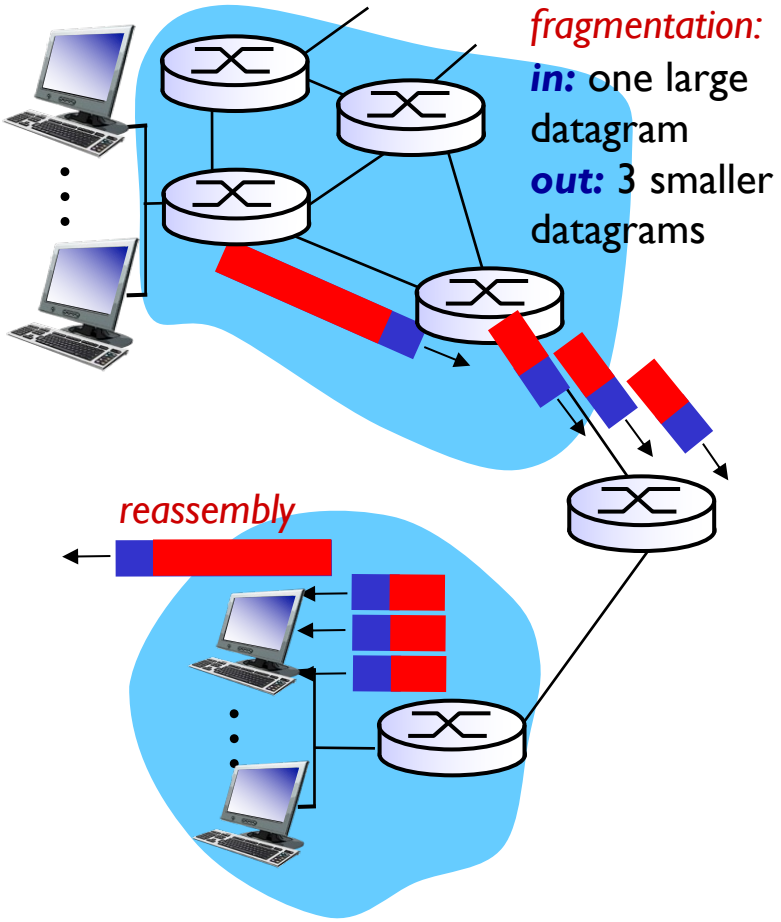
The Internet network layer

host, router network layer functions:



IP format and fragmentation

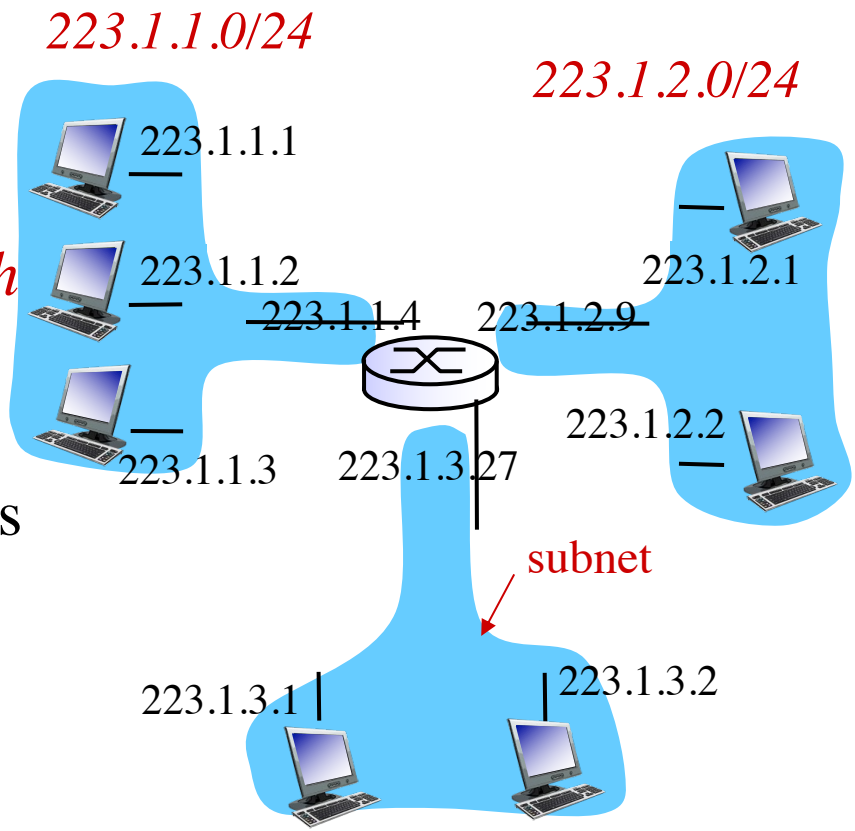
- IPv4 and IPv6 format (meaning of each field? Difference?)
- network links have MTU (max. transfer size) - largest possible link-level frame
  - different link types, different MTUs
- large IP datagram divided ("fragmented") within net
- "reassembled" only at final destination
  - IP header bits used to identify, order related fragments



The IP fragmentation and reassembly example: value of each field?

IP addressing: introduction

- *IP address:* 32-bit identifier for interface of hosts and routers
- *IP addresses associated with each interface*
- IP address:
  - subnet part - high order bits
  - host part - low order bits
- *what's a subnet ?*
  - can physically reach each other *without intervening router*
  - device interfaces with same subnet part of IP address



223.1.1.1 = 11011111 00000001 00000001 00000001

223 1 1 1

How many subnets? Subnet address? Subnet mask?

Obtaining a Block of Address

Q: how does *network* get subnet part of IP addr?

A: gets allocated portion of its provider ISP's address space. Hierarchical addressing. Why?

Suppose all of the interfaces in each of these three subnets are required to have the prefix 223.1.17.0/24.

- Subnet 1 is required to support at least 60 interfaces
- Subnet 2 is to support at least 90 interfaces
- Subnet 3 is to support at least 12 interfaces

$60 \leq 64 = 2^6$

$90 \leq 128 = 2^7$

$12 \leq 16 = 2^4$

Provide three network addresses (of the form a.b.c.d/x) that satisfy these constraints

11011111 00000001 00010001 00000000

Subnet 2: 11011111 00000001 00010001 00000000 223.1.17.0/25

Subnet 1: 11011111 00000001 00010001 10000000 223.1.17.128/26

Subnet 3: 11011111 00000001 00010001 11000000 223.1.17.192/28

Obtaining a Host Address

DHCP: Dynamic Host Configuration Protocol: dynamically get address from as server

- "plug-and-play"
- Same IP each time, or temporary IP addresses
- Why? allows reuse of addresses; support for mobile users
- Also, can return the address of first-hop router for client; name and IP address of DNS sever; network mask

A Client-Server Protocol:

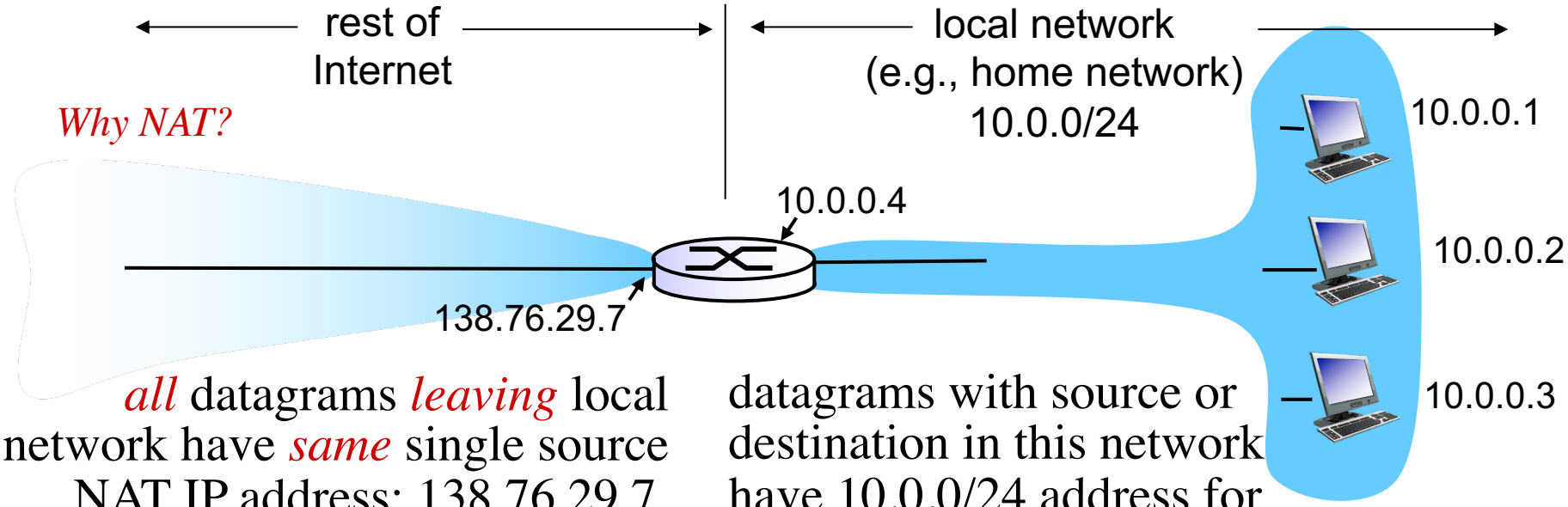
- host broadcasts "DHCP discover" msg
- DHCP server responds with "DHCP offer" msg
- host requests IP address: "DHCP request" msg (why?)
- DHCP server sends address: "DHCP ack" msg

- Given a particular message, can you explain what happens?

NAT: network address translation

Huge number of hosts and LAN … → NAT

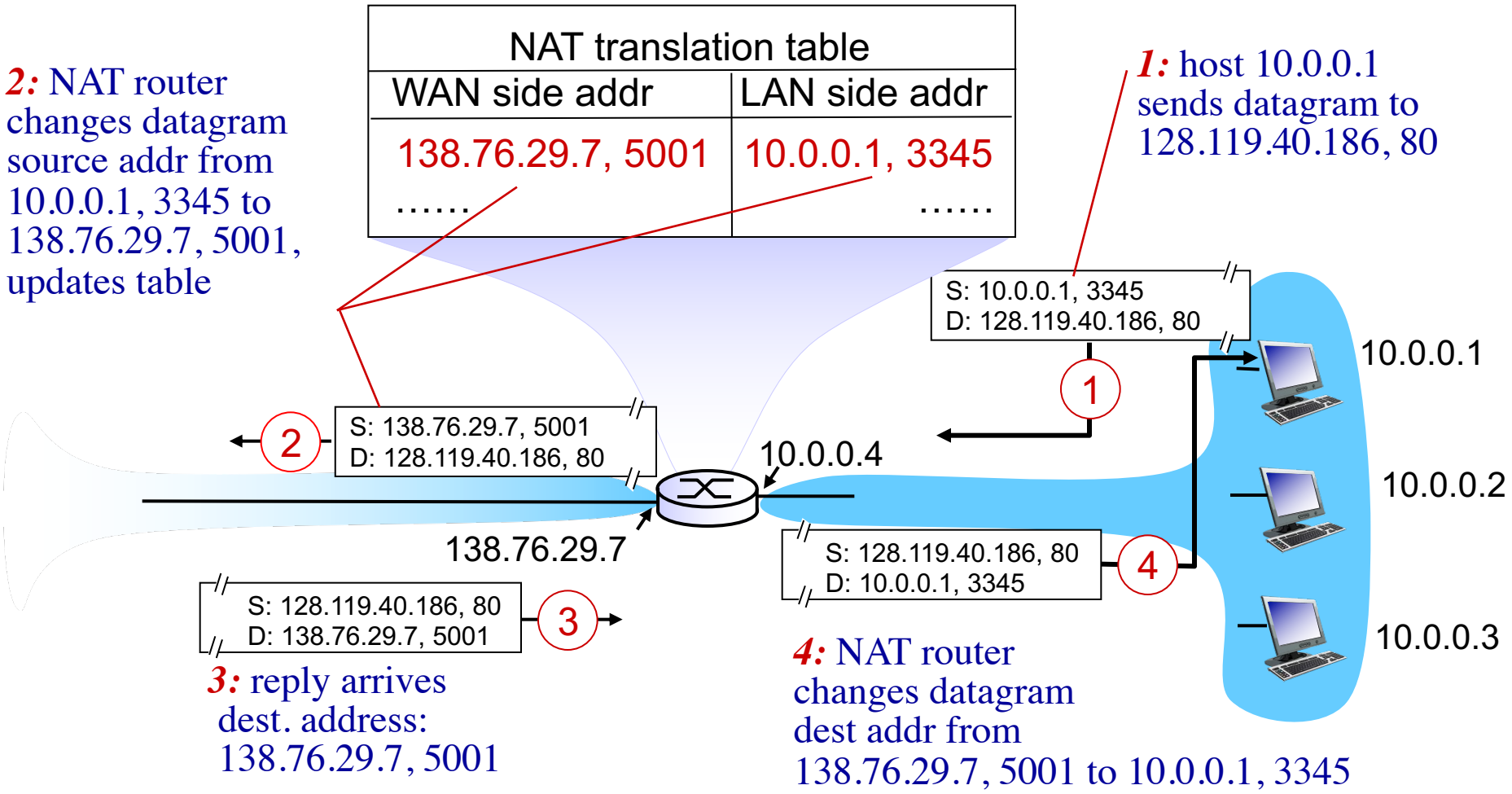
RFC 1918 name	IP address range	Number of addresses	Largest CIDR block (subnet mask)	Host ID size	Mask bits	*Classful* description[Note 1]
24-bit block	10.0.0.0 – 10.255.255.255	16 777 216	10.0.0.0/8 (255.0.0.0)	24 bits	8 bits	single class A network
20-bit block	172.16.0.0 – 172.31.255.255	1 048 576	172.16.0.0/12 (255.240.0.0)	20 bits	12 bits	16 contiguous class B networks
16-bit block	192.168.0.0 – 192.168.255.255	65 536	192.168.0.0/16 (255.255.0.0)	16 bits	16 bits	256 contiguous class C networks



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



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

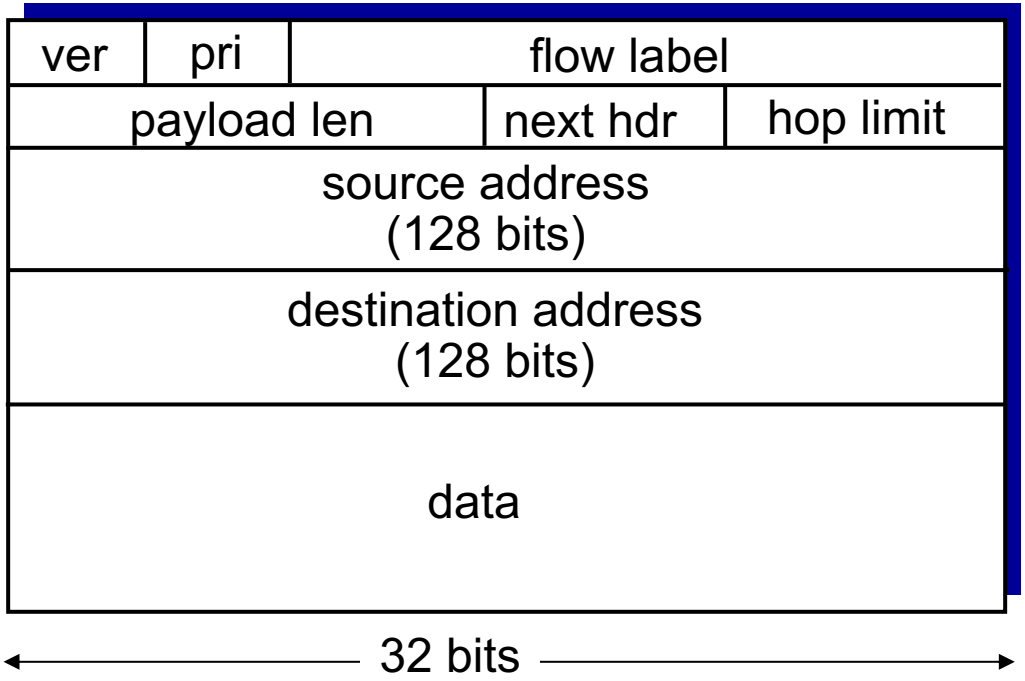
How many simultaneous connections?

IPv6: motivation

Why Ipv6?

IPv6 datagram format (differences with IPv4?):

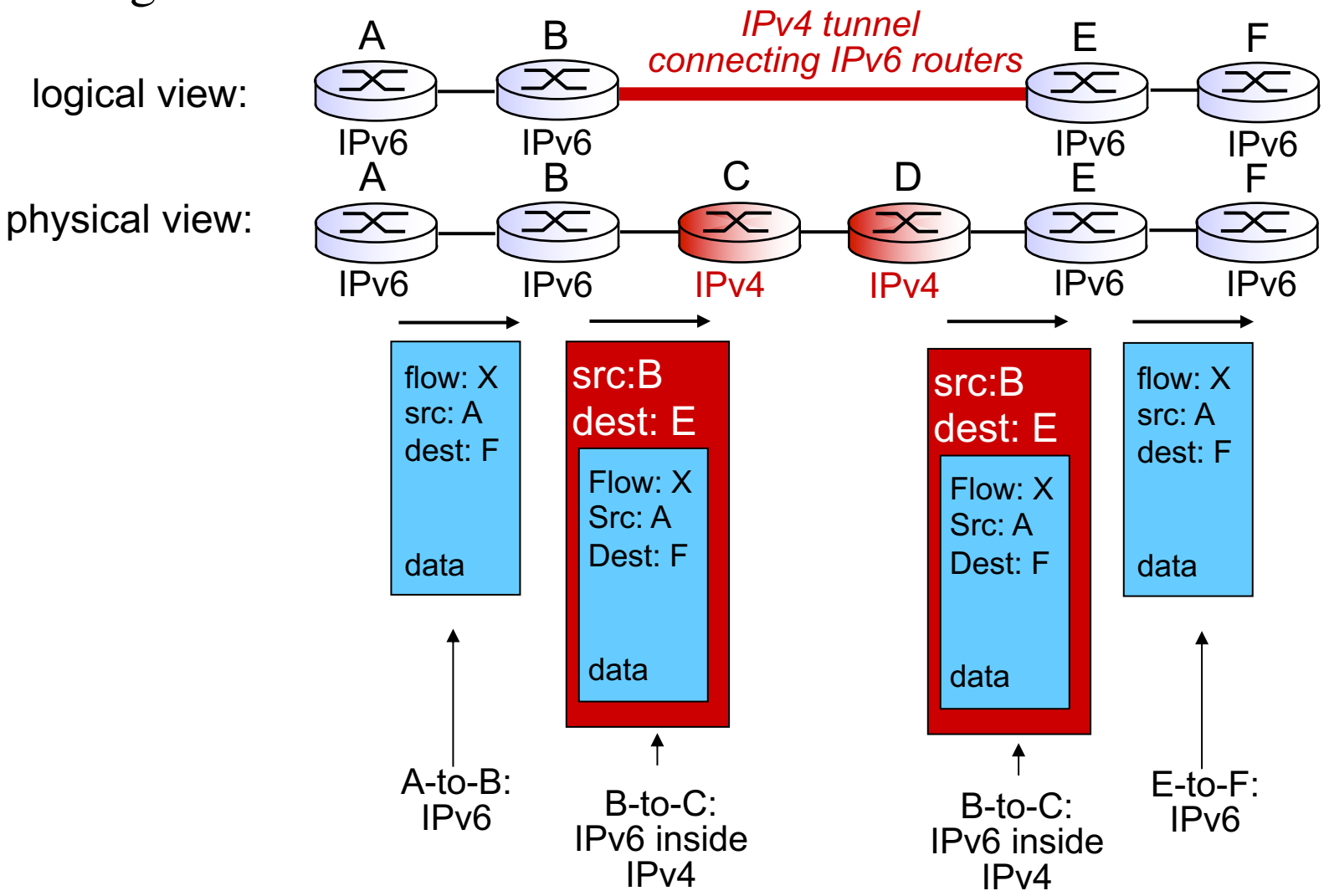
- 128-bit address space: why?
- fixed-length 40 byte header; why?
- no fragmentation allowed; why?



- *checksum*: removed entirely to reduce processing time at each hop; why?
- *options:* allowed, but outside of header, indicated by "Next Header" field
- *no fragmentation:*
  - too large to be forwarded over the outgoing link
  - the router simply drops the datagram and sends a "Packet Too Big" ICMP error message

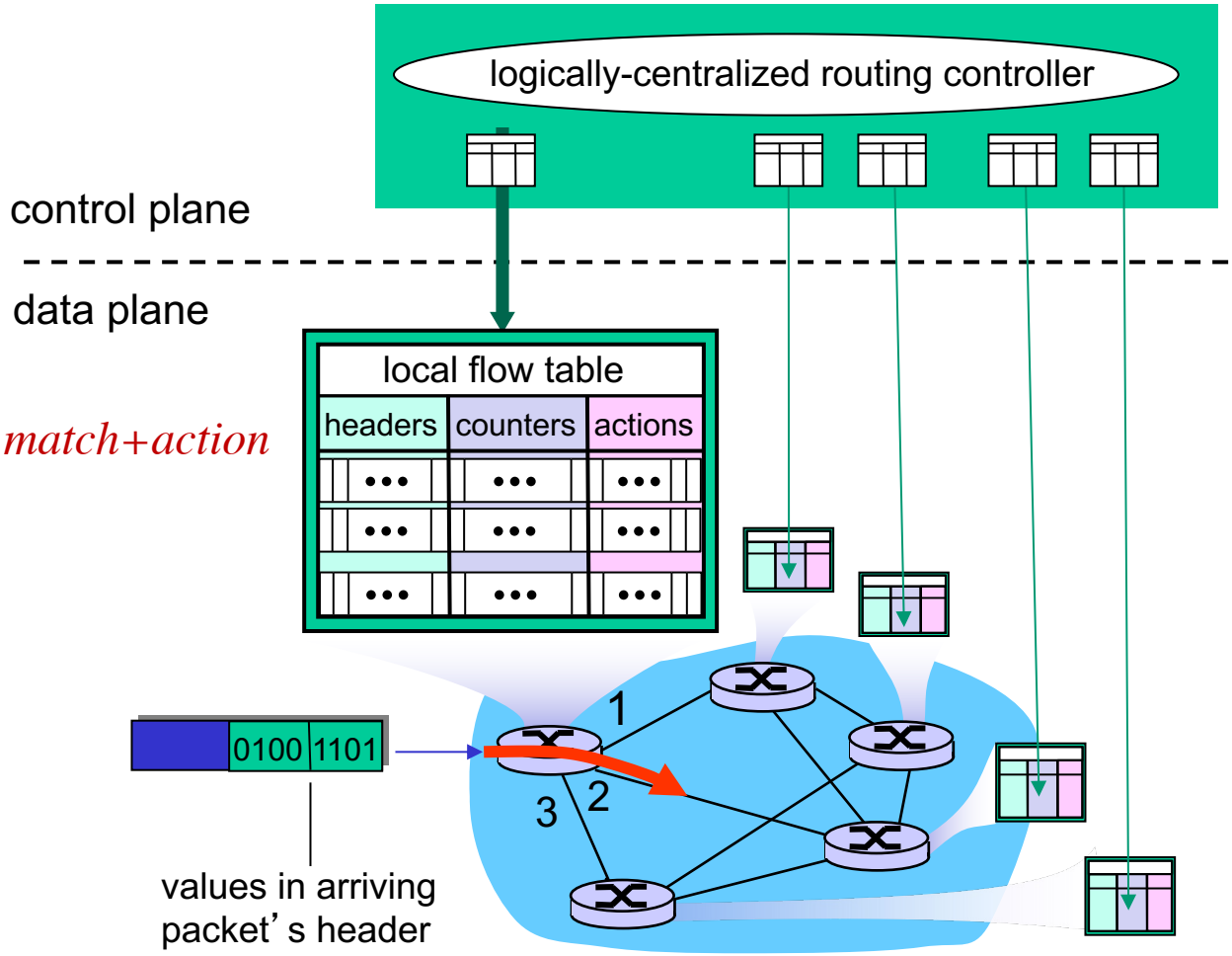
Tunneling

tunneling: IPv6 datagram carried as *payload* in IPv4 datagram among IPv4 routers

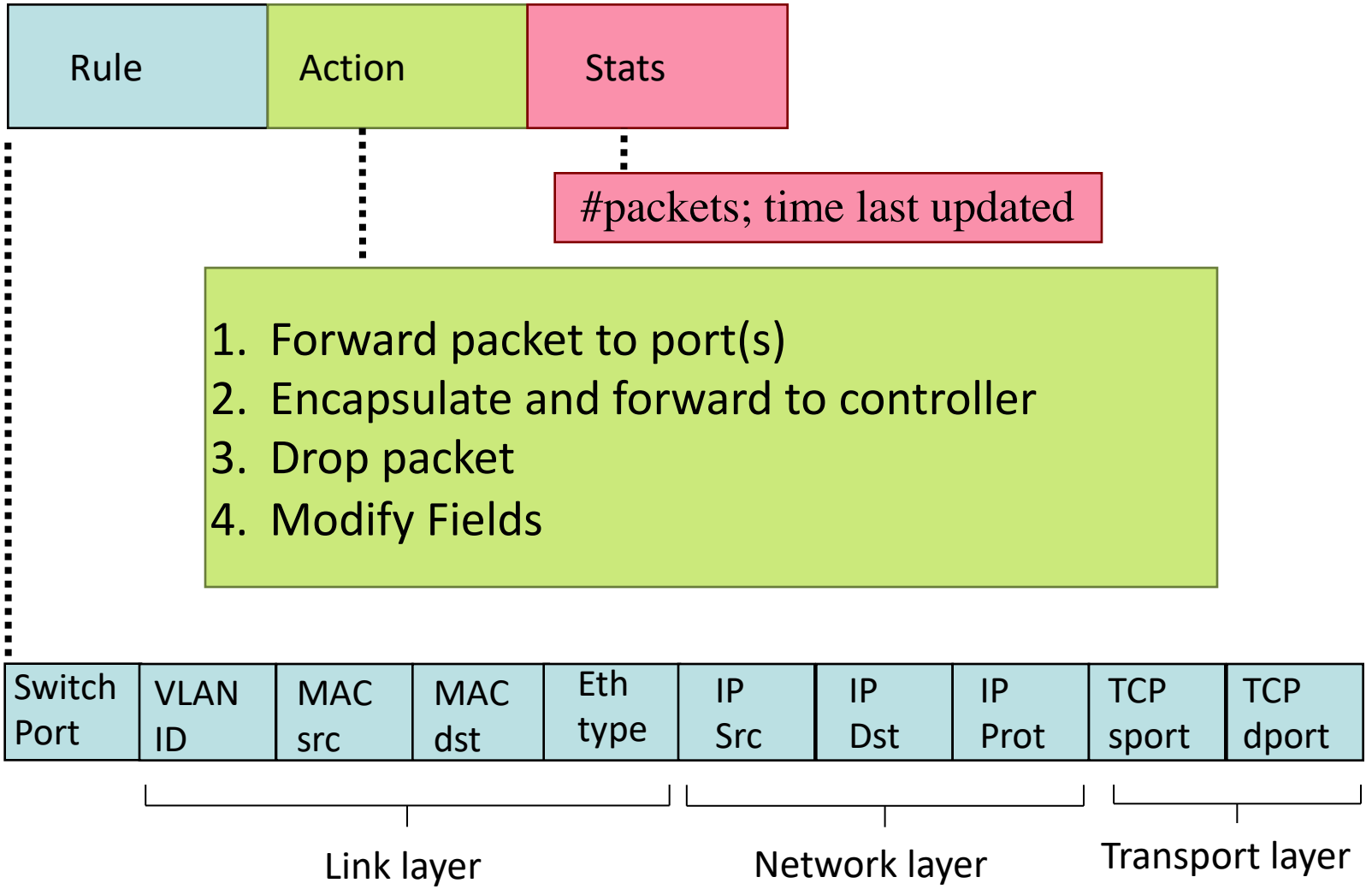


Generalized Forwarding and SDN

Each router contains a *flow table* that is computed and distributed by a *logically centralized* routing controller



OpenFlow: Flow Table Entries



How to fill in the flow table when considering destination-based forwarding? Firewall? Load balancing?

Chapter 5: outline

5.1 introduction (control plane: tranditional; SDN)

5.2 routing protocols

- link state
- distance vector

5.3 intra-AS routing in the Internet: OSPF

5.4 routing among the ISPs: BGP

5.5 The SDN control plane

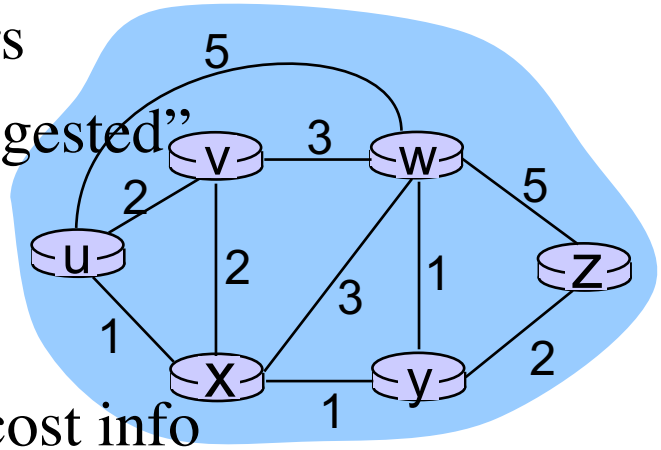
5.6 ICMP: The Internet Control Message Protocol

5.7 Network management and SNMP

Routing protocols

Routing protocol goal: determine "good" paths from sending hosts to receiving host, through network of routers

- "good": least "cost", "fastest", "least congested"



global:

- all routers have complete topology, link cost info
- "link state" algorithms -> *Dijkstra's algorithm*
- *Given the result, how to construct the forwarding table?*

decentralized:

- router knows physically-connected neighbors, link costs to neighbors
- iterative process of computation, exchange of info with neighbors
- "distance vector" algorithms

Distance vector algorithm

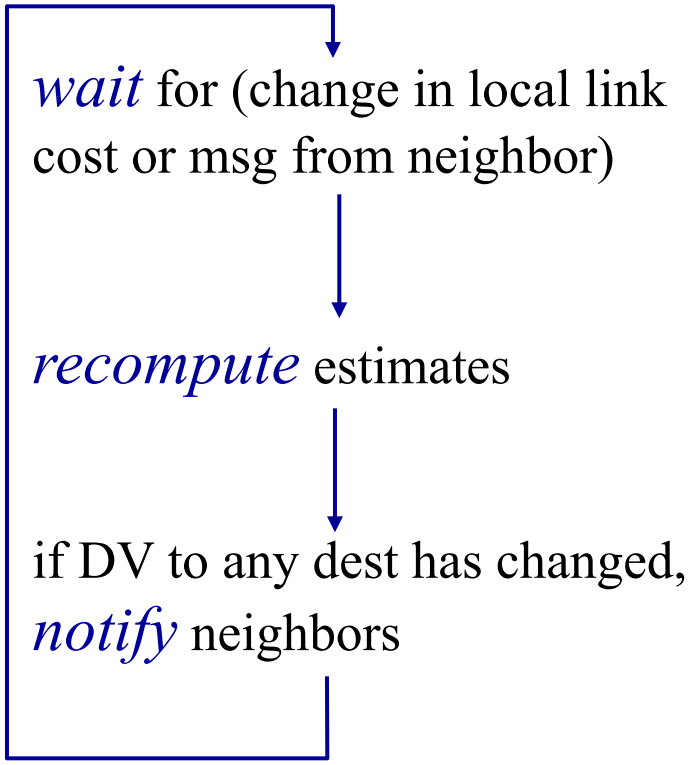
iterative, asynchronous:
each local iteration caused by:

- local link cost change
- DV update message from neighbor

distributed:

- each node notifies neighbors *only* when its DV changes
  - neighbors then notify their neighbors if necessary

each node:



Bellman-Ford equation:

$$D_x(y) \leftarrow \min_v\{c(x,v) + D_v(y)\} \quad \text{for each node } y \in N$$

$D_y(x)=\min\{c(y,x)+D_x(x),c(y,z)+D_z(x)\}$

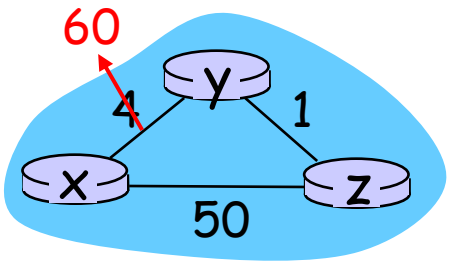
$D_z(x)=\min(c(z,x)+D_x(x),c(z,y)+D_y(x)\}$

node x table

from \ cost to	x	y	z
x	0	4	~~5~~ 50
y	4	0	1
z	5	1	0

from \ cost to	x	y	z
x			
y			
z			

from \ cost to	x	y	z
x			
y			
z			



Detect c(x,y)=c(y,x)=60 !

node y table

from \ cost to	x	y	z
x	0	4	5
y	~~4~~ 6	0	1
z	5	1	0

from \ cost to	x	y	z
x	0	51	50
y	6	0	1
z	5	1	0

from \ cost to	x	y	z
x	0	51	50
y	8	0	1
z	7	1	0

node z table

from \ cost to	x	y	z
x	0	4	5
y	4	0	1
z	5	1	0

from \ cost to	x	y	z
x	0	51	50
y	6	0	1
z	7	1	0

from \ cost to	x	y	z
x			
y			
z			

time

loop will persist for 44 iterations until z eventually computes the cost of its path via y to be greater than 50.

Poisoned reverse:

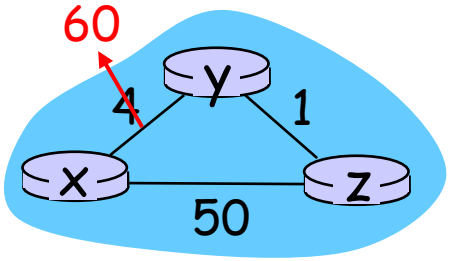
- If Z routes through Y to get to X :
  - Z tells Y its (Z's) distance to X is infinite (so Y won't route to X via Z)

node x table

from \ cost to	x	y	z
x	0	4	5
y	4	0	1
z	5	1	0

from \ cost to	x	y	z
x			
y			
z			

from \ cost to	x	y	z
x			
y			
z			



Detect c(x,y)=c(y,x)=60 !

node y table

from \ cost to	x	y	z
x	0	4	∞
y	4	0	1
z	∞	1	0

from \ cost to	x	y	z
x	0	51	50
y	6	0	1
z	5	1	0

from \ cost to	x	y	z
x	0	51	50
y	51	0	1
z	50	1	0

node z table

from \ cost to	x	y	z
x	0	4	5
y	4	0	1
z	5	1	0

from \ cost to	x	y	z
x	0	51	50
y	60	0	1
z	50	1	0

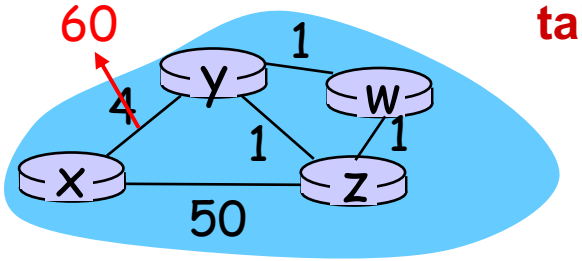
from \ cost to	x	y	z
x			
y			
z			

time

will this completely solve count to infinity problem?

No, when the loops involves three or more nodes

Distance vector: link cost changes



node x table

from \ cost to	x	y	z	w
x	0	4	5	5
y	4	0	1	1
z	5	1	0	1

node y table

from \ cost to	x	y	z	w
x	0	4	∞	∞
y	4	0	1	1
z	∞	1	0	1
w	∞	1	1	0

60

node z table

from \ cost to	x	y	z	w
x	0	4	5	5
y	4	0	1	1
z	5	1	0	1
w	5	1	1	0

node z table

from \ cost to	x	y	z	w
x	0	4	5	5
y	60	0	1	1
z	6	1	0	1
w	5	1	1	0

node w table

from \ cost to	x	y	z	w
x				
y	60	0	1	1
z	5	1	0	1
w	6	1	1	0

node w table

from \ cost to	x	y	z	w
y	4	0	1	1
z	5	1	0	1
w	5	1	1	0

node y table

from \ cost to	x	y	z	w
x	0	4	∞	∞
y	7	0	1	1
z	6	1	0	1
w	6	1	1	0

- Knows only one-hop neighbors' information
- Sends infinity to only the first-hop node along the path

Making routing scalable

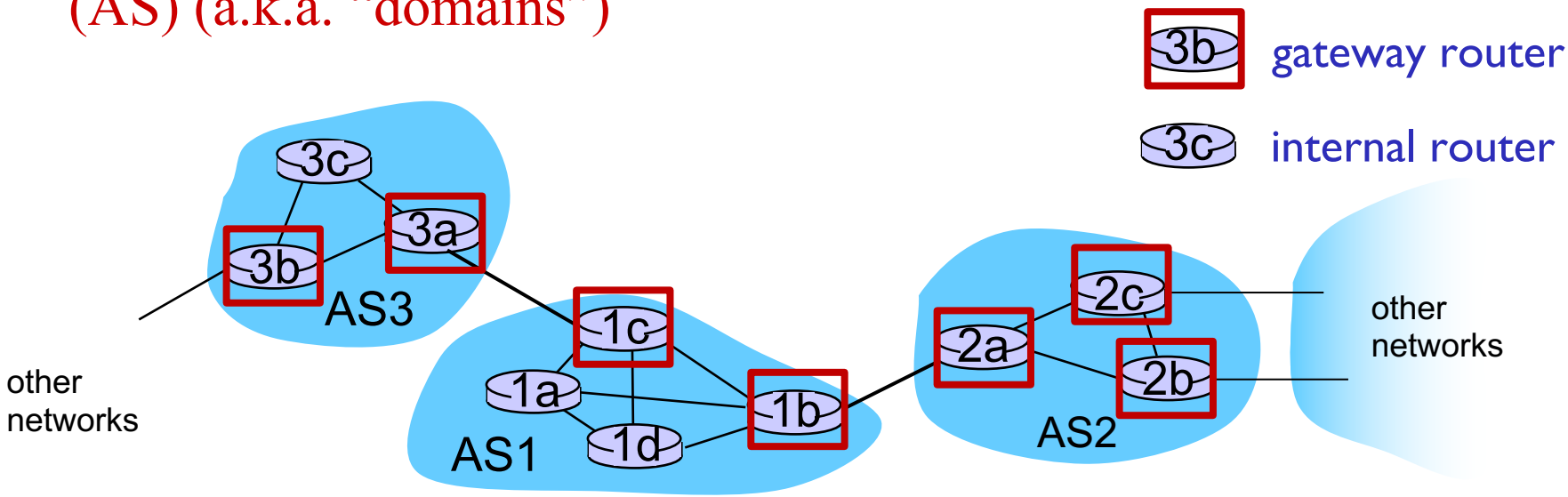
The link state and distance vector routing studies far is idealized

- all routers identical
- network "flat"

... *not* true in practice (why?)

- *scale:* with billions of destinations; storage; links
- *administrative autonomy*

Aggregate routers into regions known as "autonomous systems" (AS) (a.k.a. "domains")



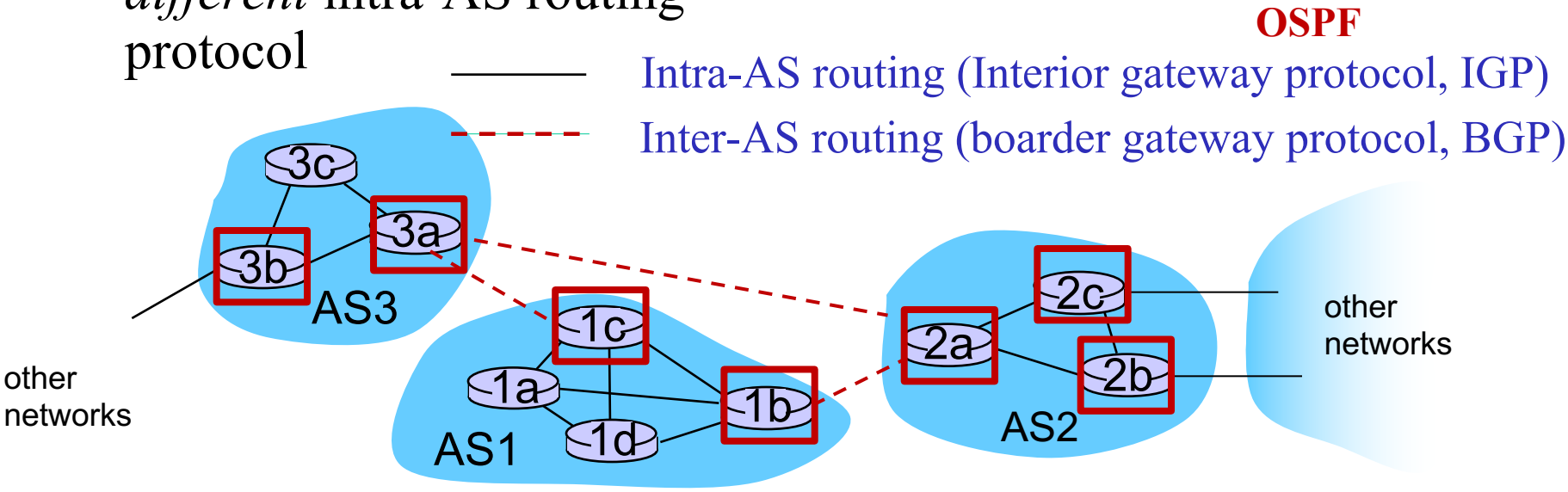
Internet approach to scalable routing

intra-AS routing

- routing among hosts, routers in same AS ("network")
- all routers in AS must run *same* intra-domain protocol
- routers in *different* AS can run *different* intra-AS routing protocol

inter-AS routing

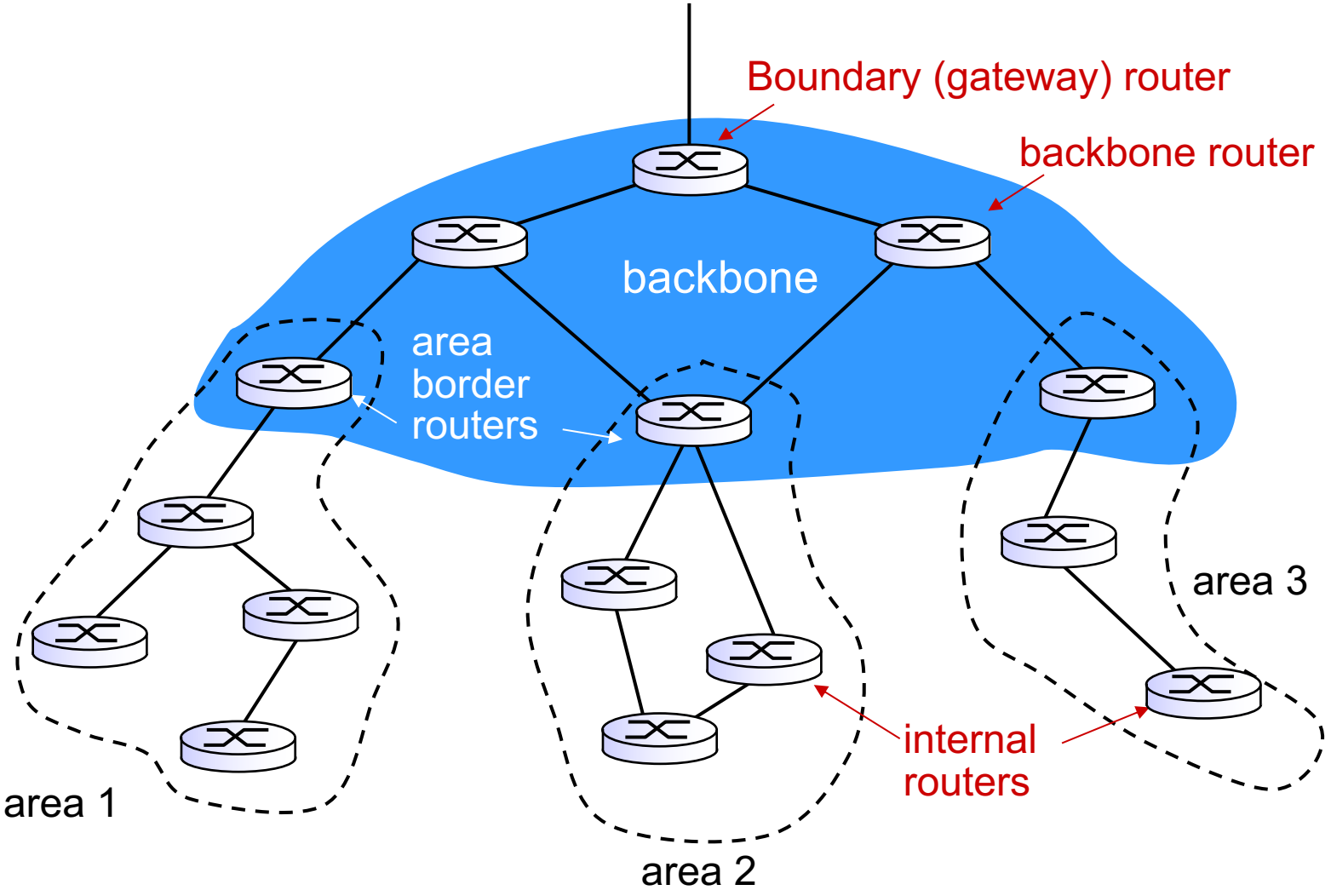
- routing among AS'es
- gateways perform inter-AS routing (as well as intra-AS routing)



OSPF (Open Shortest Path First)

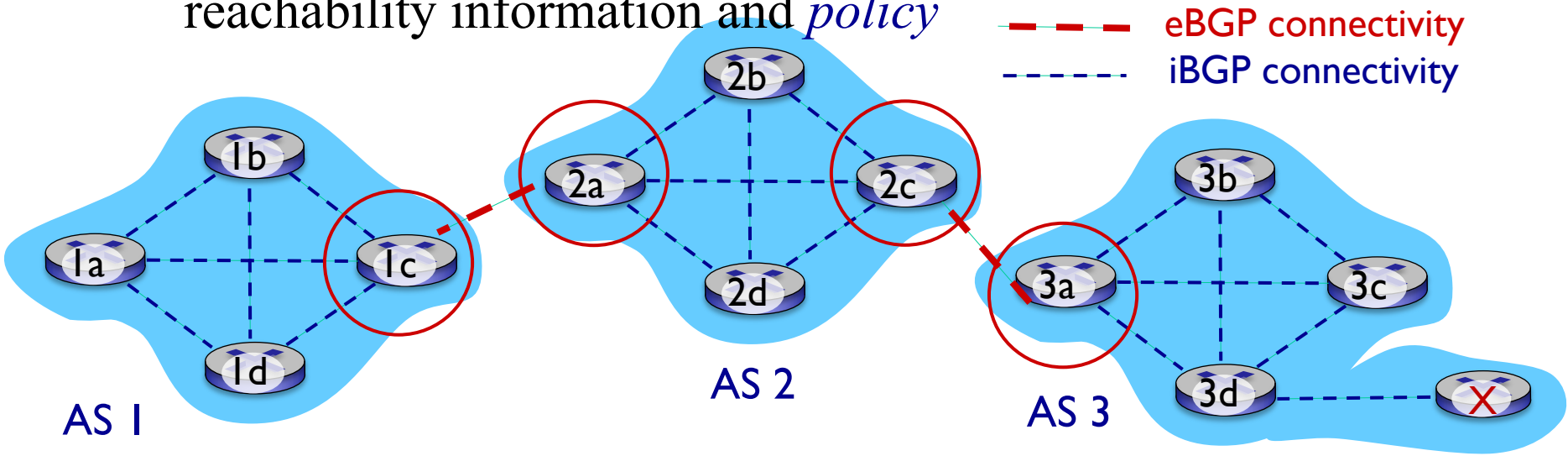
- Message format, routing algorithms, link-state broadcast…
- uses link-state algorithm
  - link state packet dissemination (Dijkstra's algorithm)
  - topology map at each node
- router floods OSPF link-state advertisements to all other routers in *entire* AS
  - carried in OSPF messages directly over IP
  - Reliable message transfer, link-state broadcast
- multiple same-cost paths allowed
- *two-level hierarchy:* local area, backbone
  - link-state advertisements only in area
  - each nodes has detailed area topology; only know direction (shortest path) to nets in other areas.

Hierarchical OSPF



Internet inter-AS routing: BGP

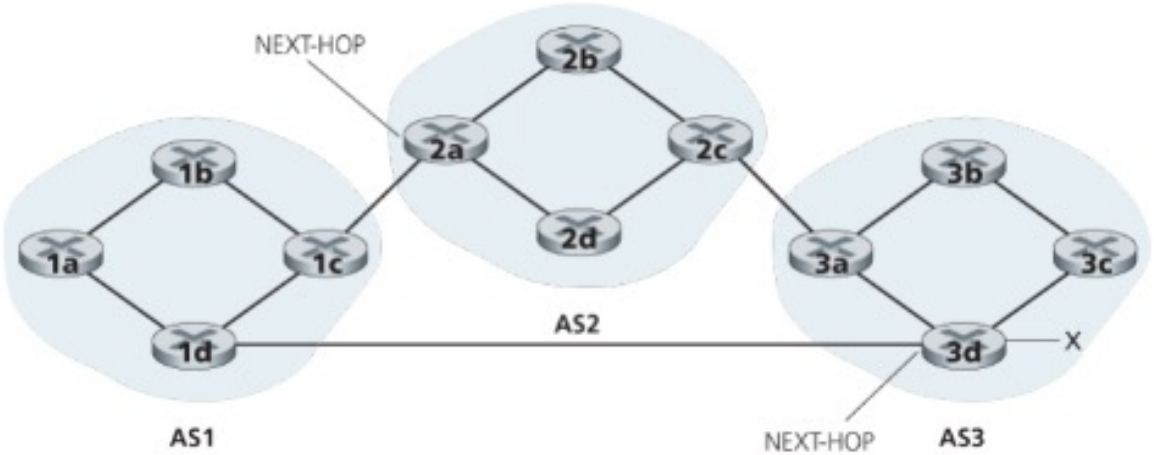
- BGP (Border Gateway Protocol): inter-domain routing protocol
  - Decentralized, asynchronous, distance-vector
- Main functions BGP provides:
  - allows subnet to advertise its existence to rest of Internet:
    - subnet reachability information from neighboring: eBGP
    - reachability information to all AS-internal routers: iBGP
  - determine "good" routes to other networks based on reachability information and *policy*



Path attributes and BGP routes

- advertised prefix includes BGP attributes
  - Prefix (destination) + attributes = "route"
- two important attributes:
  - AS-PATH: list of ASes through which the advertisement has passed, e.g., AS2 AS3
    - Advertisement; prevent loops
  - NEXT-HOP: IP address of the router interface that begins the AS-PATH

Q: how does router set forwarding table entry to distant prefix?



IP address of leftmost interface for router 2a; AS2 AS3; x

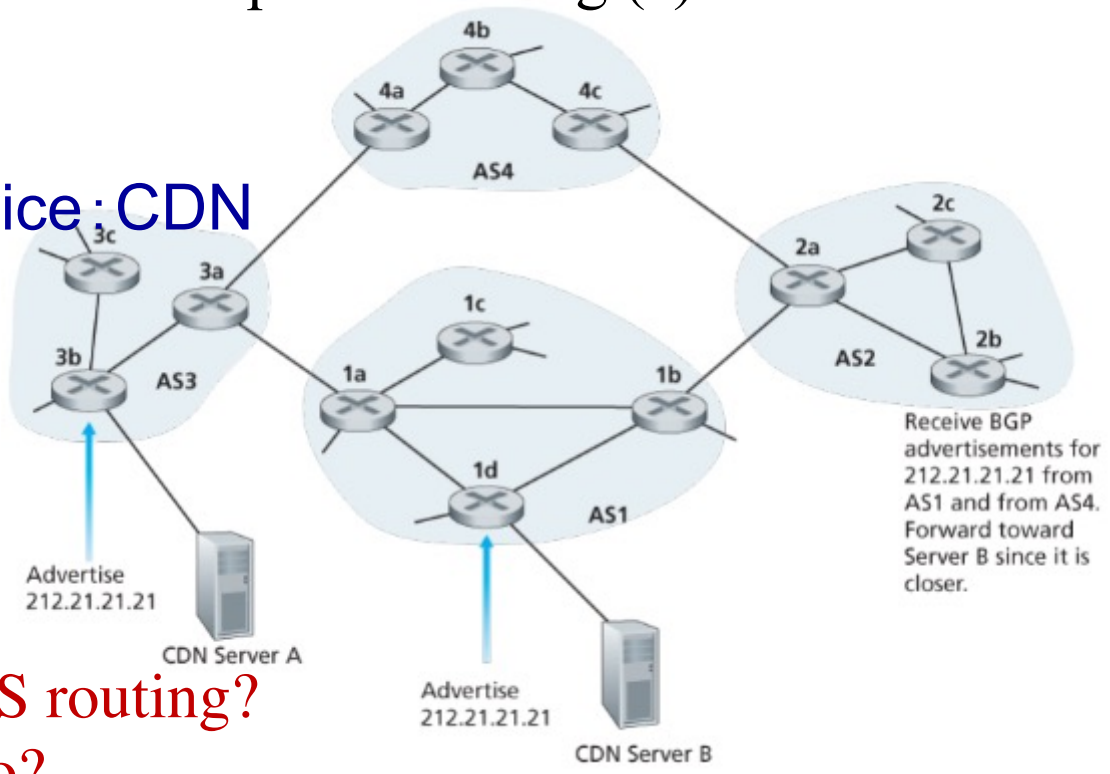
IP address of leftmost interface of router 3d; AS3; x

BGP route selection

Router may learn about more than one route to destination AS, selects route based on:

1. local preference value attribute: policy decision
2. shortest AS-PATH
3. closest NEXT-HOP router: hot potato routing (?)
4. additional criteria

IP-Anycast Service : CDN



Why different Intra-, Inter-AS routing?
Difference between these two?

SDN perspective: data plane switches

- *SDN v.s. traditional approach? Logically centralized*? Why?

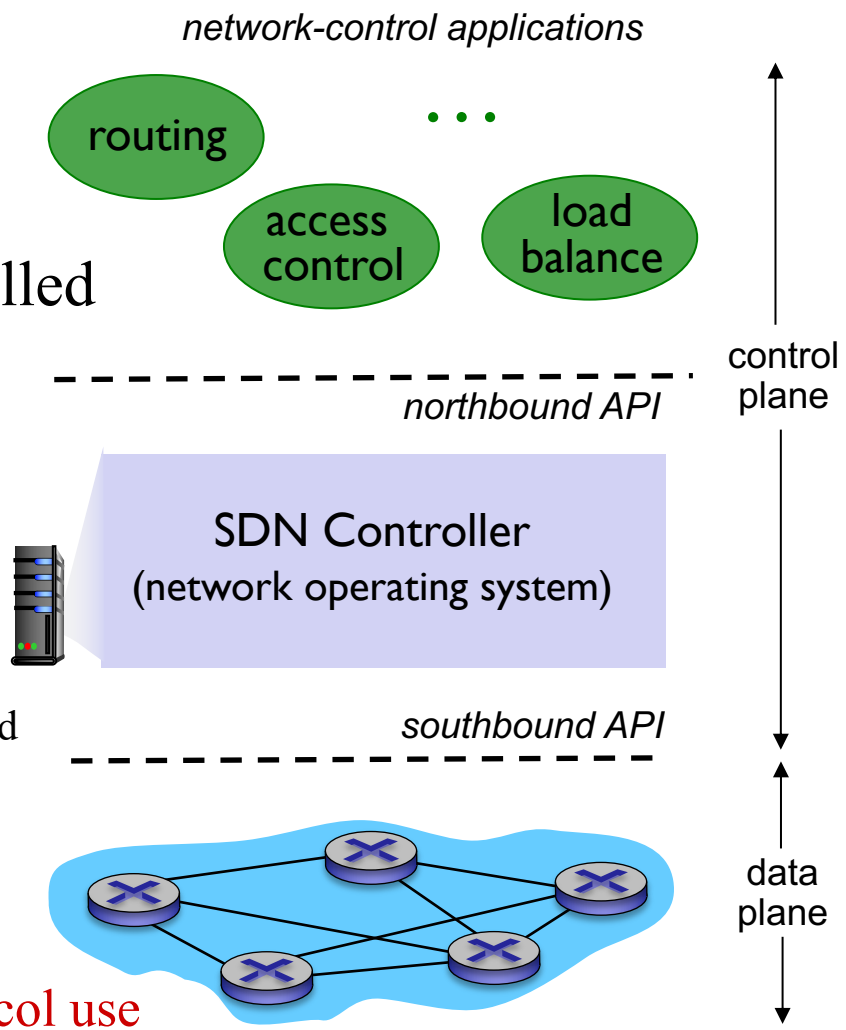
Data plane switches

- switch flow table computed, installed by controller; OpenFlow for communication and control

SDN controller (network OS):

Network-control apps:

- "brains" of control
- *unbundled:* can be provided by 3rd party:



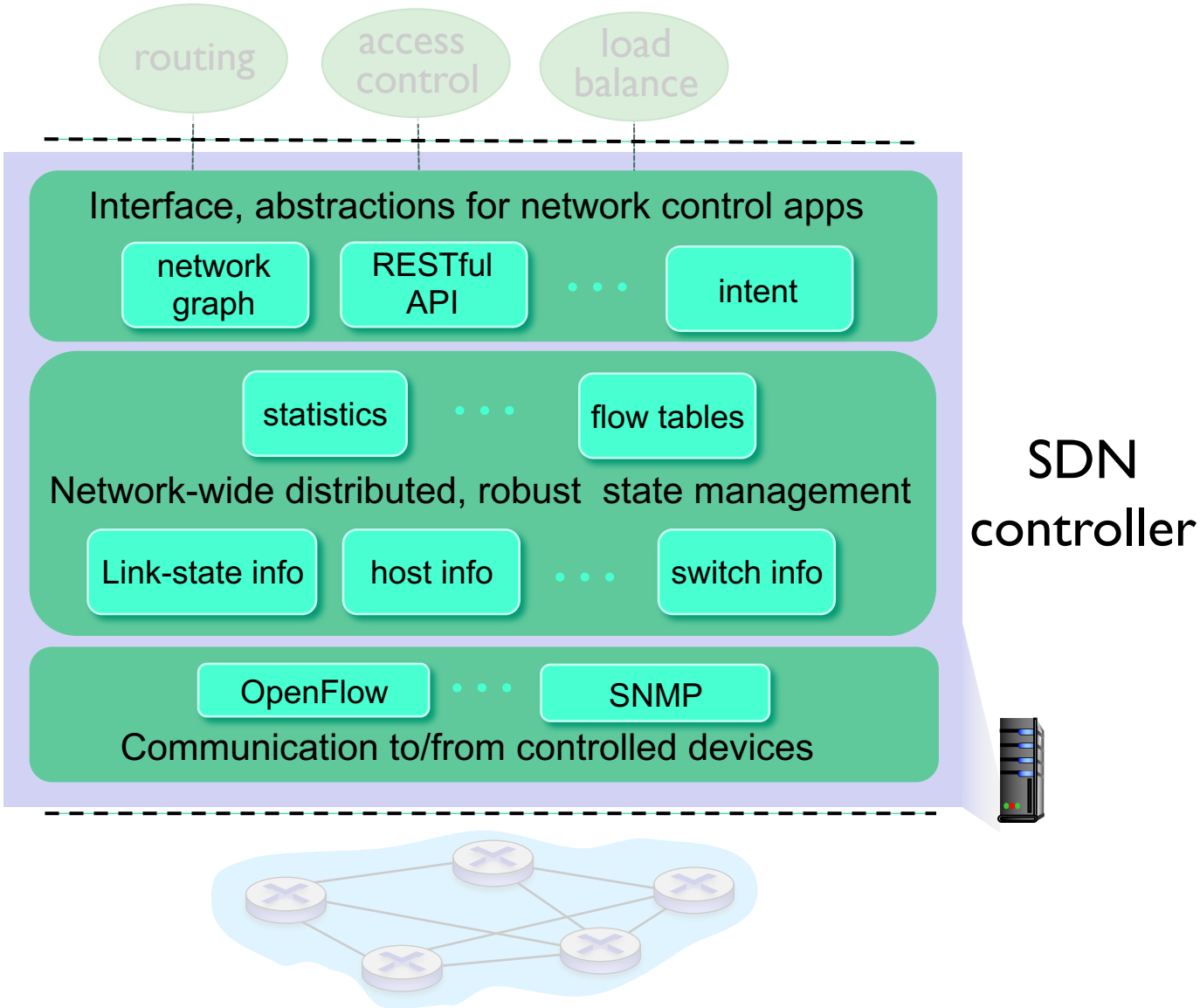
OpenFlow is the standard southbound protocol used between the SDNcontroller and the switch.

Components of SDN controller

Interface layer to network control apps: abstractions API

Network-wide state management layer: state of networks links, switches, services: a *distributed database*

communication layer: communicate between SDN controller and controlled switches



OpenFlow protocol

- operates between controller and switch
- TCP used to exchange messages

controller-to-switch messages

- *configure:* controller queries/sets switch configuration parameters
- *modify-state:* add, delete, modify flow entries in the OpenFlow tables
- *Read-state:* collect statistics and counter values from the switch's flow table and ports
- *packet-out:* controller can send this packet out of specific switch port

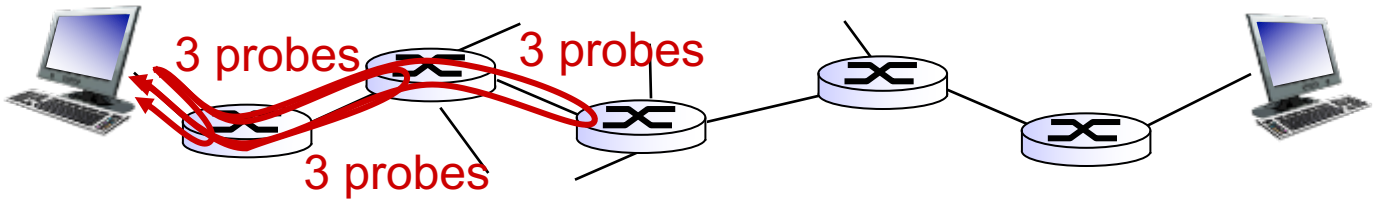
Key switch-to-controller messages

- *packet-in:* transfer packet (and its control) to controller. See packet-out message from controller
- *flow-removed:* flow table entry deleted at switch
- *port status:* inform controller of a change on a port.

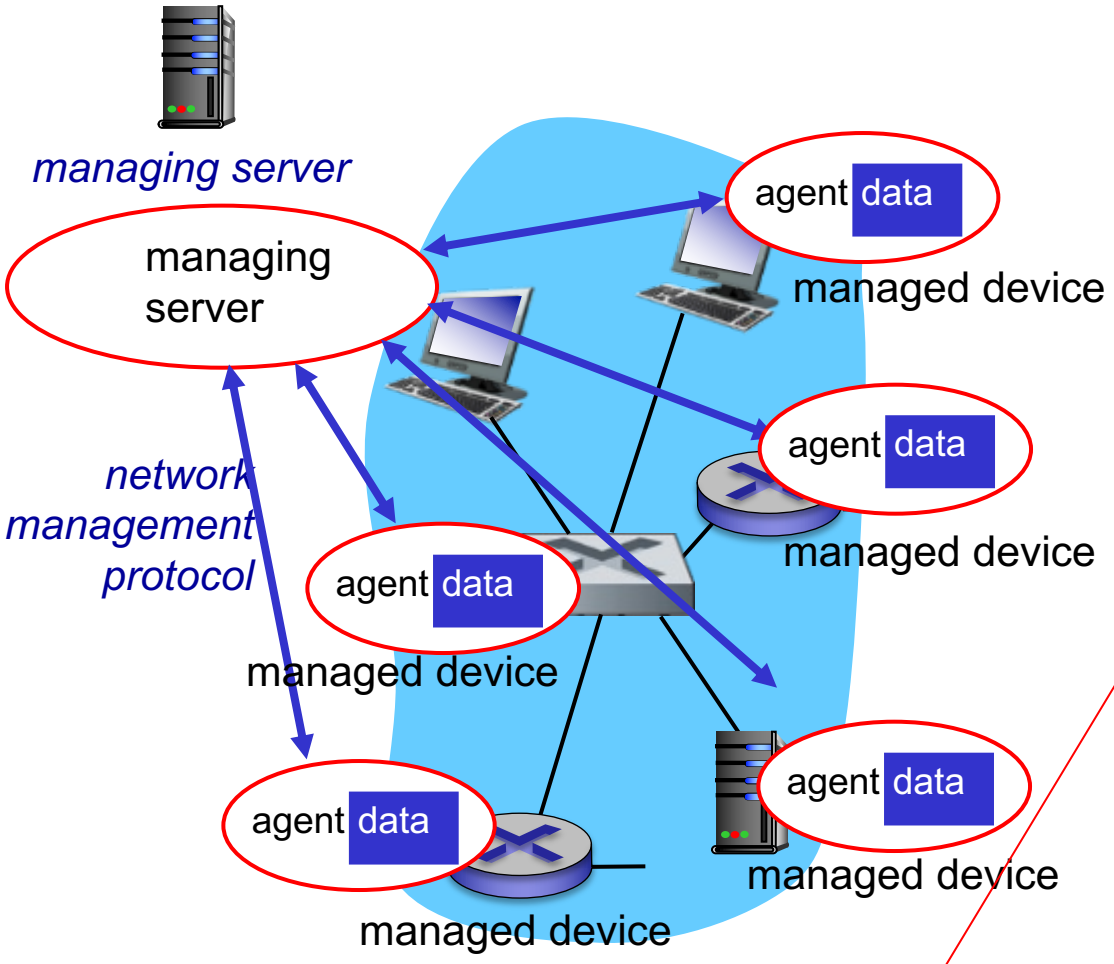
ICMP: internet control message protocol

- used by hosts & routers to communicate network-level information
  - error reporting: unreachable host, network, port, protocol
  - echo request/reply (used by ping)
- network-layer "above" IP:
  - ICMP msgs carried in IP datagrams
- ICMP message:
  - Type + code + the header and the first 8 bytes of IP datagram causing error

The traceroute example?



Network management



host, router, switch, middlebox, modem, or other network-connected device

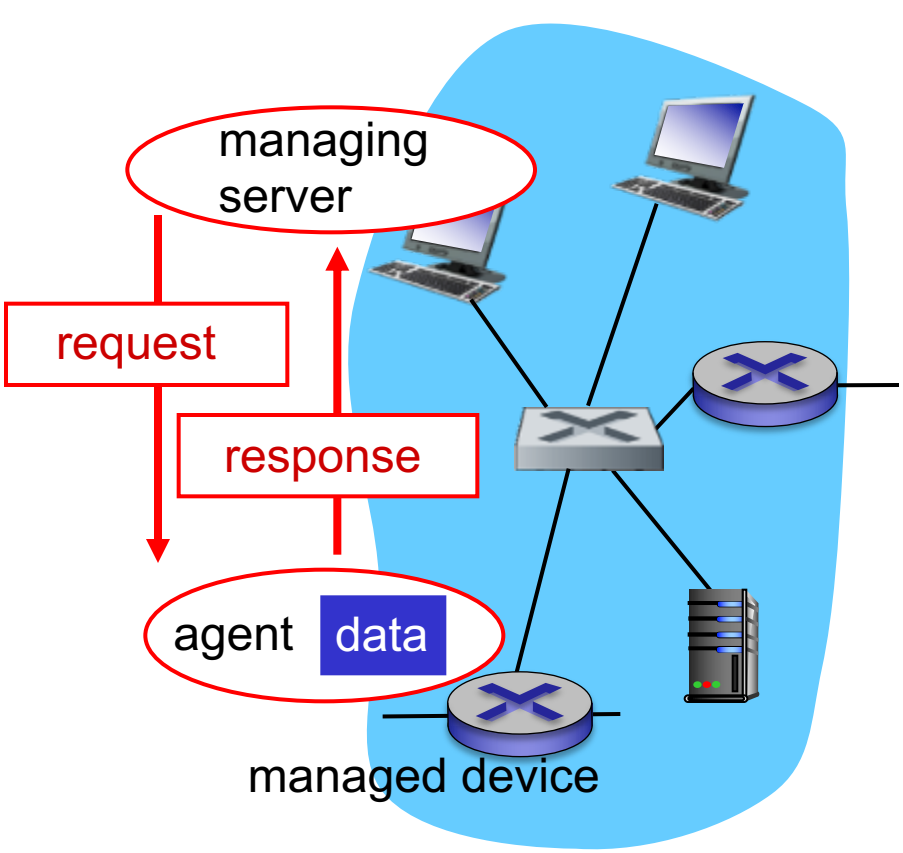
managed devices contain *managed objects* whose data is gathered into a *Management Information Base (MIB)*

the actual pieces of hardware within the managed device and configuration parameters for these hardware and software components

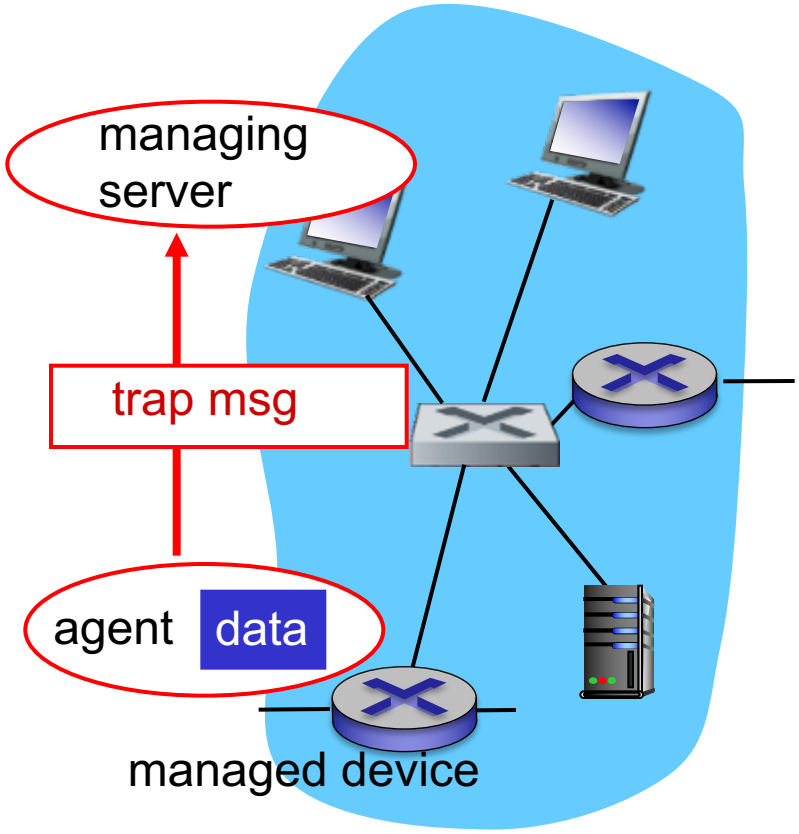
a counter, descriptive information, or protocol-specific information

Simple Network Management Protocol (SNMP)

Two usages of SNMP



request/response mode



trap mode

Link layer, LANs: outline

6.1 introduction, services

6.2 error detection, correction

6.3 multiple access protocols

6.4 LANs

- addressing, ARP
- Ethernet
- switches
- VLANS

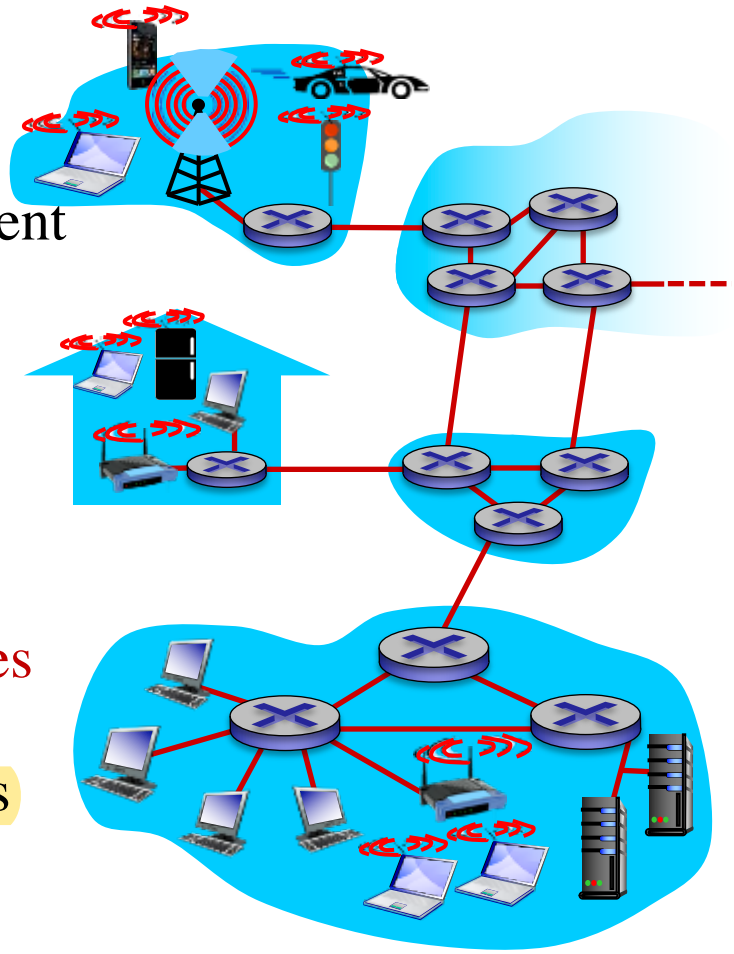
6.6 data center networking

6.7 a day in the life of a web request

Procedure? Protocols?

Link layer: introduction

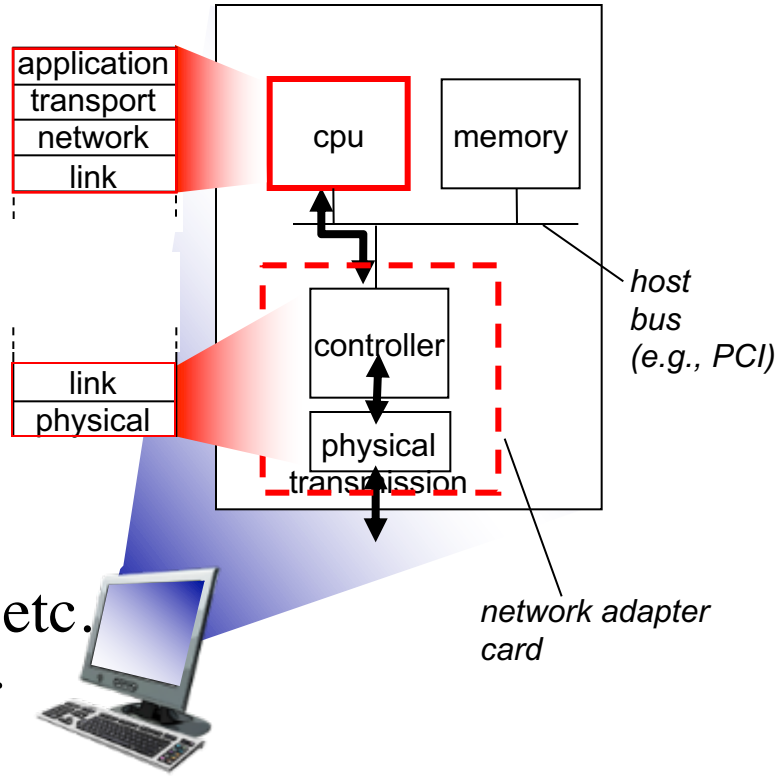
link layer has responsibility of transferring datagram from one node to *physically adjacent* node over a link



- communication channels that connect adjacent nodes along communication path:
  - wired links
  - wireless links
- datagram transferred by different link protocols over different links:
- each link protocol provides different services
  - Framing， channel access （shared）
  - reliable delivery between adjacent nodes (why? 802.11? Ethernet?)
  - error detection/correction
  - Flow control
  - Half-duplex and full-duplex

Where is the link layer implemented?

- link layer implemented in "adaptor" (aka *network interface card* NIC) or on a chip
  - Ethernet card, 802.11 card; Ethernet chipset
  - implements link, physical layer
  - MAC address
- sending side:
  - encapsulates datagram in frame
  - adds error checking bits, rdt, flow control, etc.
- receiving side
  - looks for errors, rdt, flow control, etc.
  - extracts datagram, passes to upper layer at receiving side



Error detection

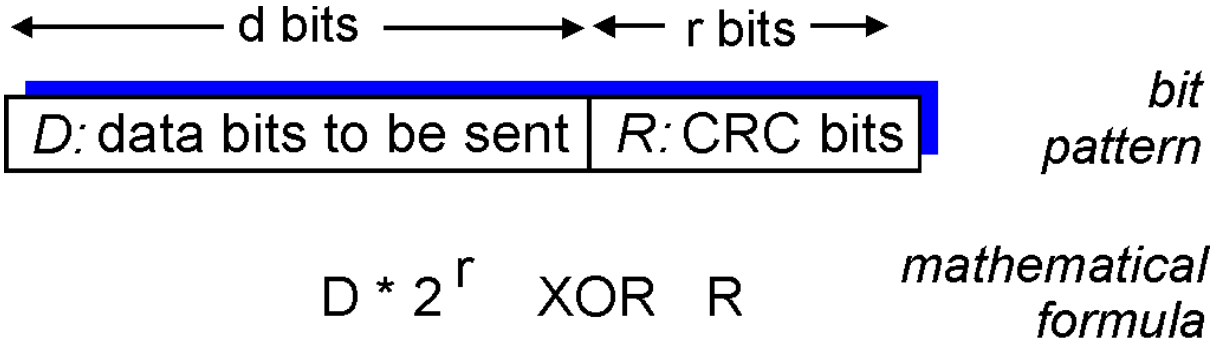
Parity checks

- Single bit parity: How to compute? Can detect any error?
- Two-dimensional bit parity: How to compute? How to correct error?

Check-summing methods

Cyclic-redundancy checks

- How to compute?
- modulo-2 arithmetic division



Multiple access protocols

- single shared broadcast channel
- two or more simultaneous transmissions by nodes: interference
  - *collision* if node receives two or more signals at the same time

Desired properties:

1. when one node wants to transmit, it can send at rate R.
2. when M nodes want to transmit, each can send at average rate R/M
3. fully decentralized:
   - no special node to coordinate transmissions
   - no synchronization of clocks, slots
4. simple

MAC protocols: taxonomy

three broad classes:

- *channel partitioning*
  - divide channel into smaller "pieces": time slots, frequency, code (how to compute?)
  - when M nodes want to transmit, each can send at average rate R/M
- *random access* channel not divided, allow collisions
  - "recover" from collisions: Access with probability; waiting
  - when one node wants to transmit, it can send at rate R
- *"taking turns"*
  - nodes take turns, but nodes with more to send can take longer turns
  - Polling, token passing
  - Disadvantage: Overhead; single point of failure
  - If the overhead is negligible:
    - when one node wants to transmit, it can send at rate R
    - when M nodes want to transmit, each can send at average rate R/M

Random access protocols

- when node has packet to send
  - transmit at full channel data rate R.
  - no *a priori* coordination among nodes
- Slotted ALOHA
  - Slots; transmit at the slot beginning
  - when node obtains fresh frame, transmits in next slot
    - *if no collision:* node can send new frame in next slot
    - *if collision:* node retransmits frame in each subsequent slot with prob. p until success
  - Pros: single node->full rate; decentralized; simple
  - Cons: collisions, wasting slots; idle slots; synchronization
- ALOHA
  - No time slot; transmit immediately when frame arrives
  - Efficiency when compared with Slotted ALOHA?
- CSMA

CSMA (carrier sense multiple access)

CSMA: listen before transmit:

- if channel sensed idle: transmit entire frame
- if channel sensed busy, defer transmission
  Collisions *can* still occur (why?): propagation delay means two nodes may not hear each other's transmission

CSMA/CD: CSMA+CD (collision detection)

- collision detection → stop talking:
  - collisions *detected* within short time
  - colliding transmissions are aborted, reducing channel wastage
- easy in wired LANs (why?), difficult in wireless LANs (why? Fading, multi-path)

Ethernet CSMA/CD algorithm

- If NIC senses channel idle, starts frame transmission. If NIC senses channel busy, waits until channel idle, then transmits.
- If NIC transmits entire frame without detecting another transmission, NIC is done with frame!
- If NIC detects another transmission while transmitting, aborts and sends jam signal

5. After aborting, NIC enters *binary (exponential) backoff:*
   - after *m*th collision, NIC chooses *K* at random from $\{0,1,2, \ldots, 2^m-1\}$.
   - NIC waits K·512 bit times, returns to Step 2
   - longer backoff interval with more collisions

MAC addresses and ARP

- MAC (or LAN or physical or Ethernet) address:
  - Adapter (network interface) rather than host or routers
  - Link-layer switches do NOT have MAC addresses
  - function: *used "locally" to get frame from one interface to another physically-connected interface (same network, in IP-addressing sense)*
  - 48 bit MAC address (for most LANs) burned in NIC ROM
  - e.g.: 1A-2F-BB-76-09-AD
  - Difference between MAC address and IP address?
    - 48 bit; fixed; flat address

LAN addresses and ARP

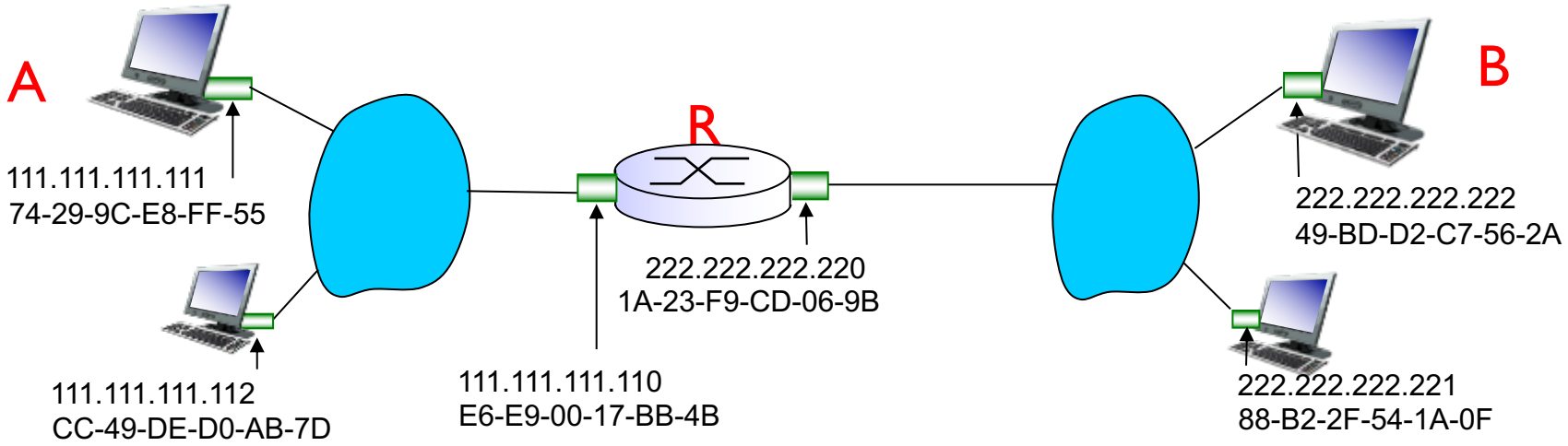
- inserts the destination adapter's MAC address into the frame and then sends the frame into the LAN
- an adapter receive a frame
  - If there is a match, extracts the enclosed datagram and passes the datagram up the protocol stack ;
  - If there isn't a match, discards
- MAC broadcast address FF-FF-FF-FF-FF-FF

- ARP: IP -> MAC (how?)
  - A broadcasts ARP query with B's IP address
  - B replies to A with MAC address
- ARP table: < IP address; MAC address; TTL>

Addressing: routing to another LAN

Send datagram from A to B via R

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



The detailed procedure?

Ethernet

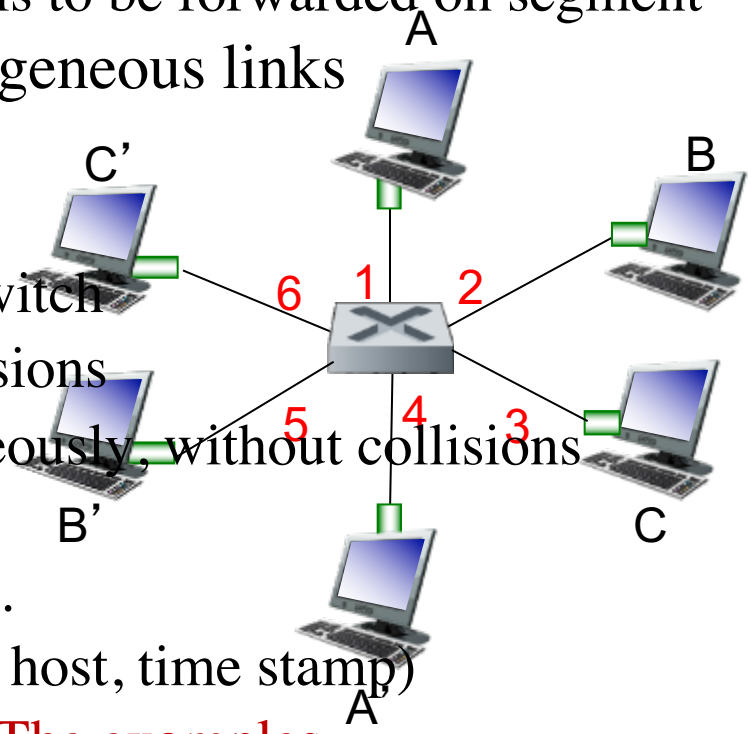
"dominant" wired LAN technology:

- Topology
  - Bus and star (pros of star topology?)
- Format

  - Preamble: why? How does it work?
- Unreliable (why?), connectionless
- Ethernet's MAC protocol: unslotted *CSMA/CD with binary backoff*

Ethernet switch

- link-layer device
  - store, forward Ethernet frames
  - examine incoming frame's MAC address, selectively forward frame to one-or-more outgoing links when frame is to be forwarded on segment
  - isolates one link from another; heterogeneous links
- *Transparent*
- *plug-and-play, self-learning*
- hosts have dedicated, direct connection to switch
- switches buffer packets; elimination of collisions
- A-to-A' and B-to-B' can transmit simultaneously, without collisions
- each switch has a switch table
  - no entry, entry with interface x, with y …
  - (MAC address of host, interface to reach host, time stamp)
  - How does switch learn the reachability? The examples
  - Interconnecting switches



Switches vs. routers?

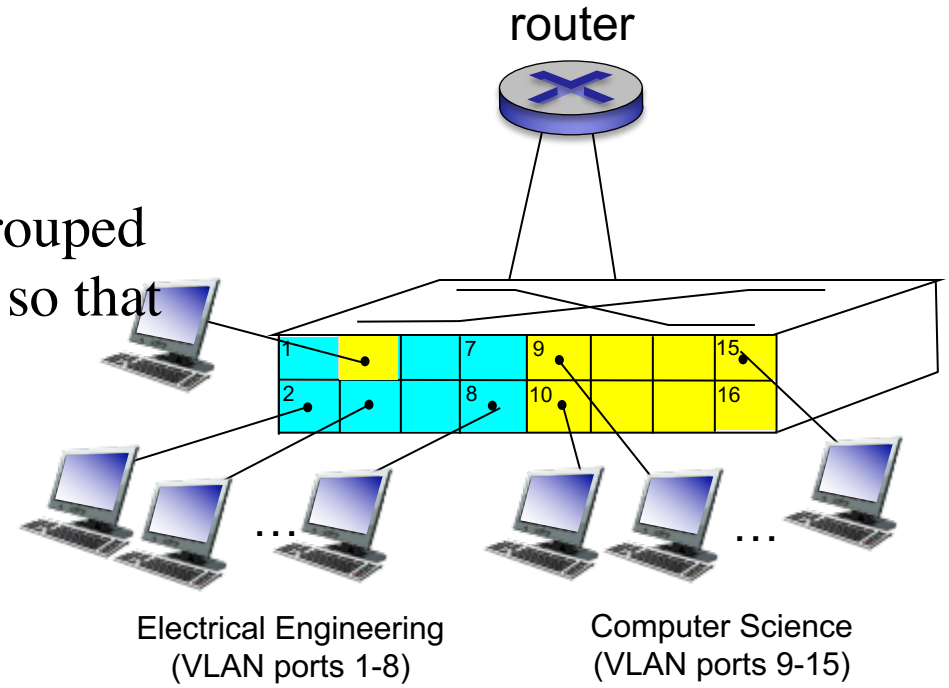
VLANs

Motivation:

- CS user moves office to EE, but wants to connect to CS switch
- Inefficient use of switches
- single broadcast domain

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

- traffic isolation
- Dynamic membership
- Forwarding between VLANs



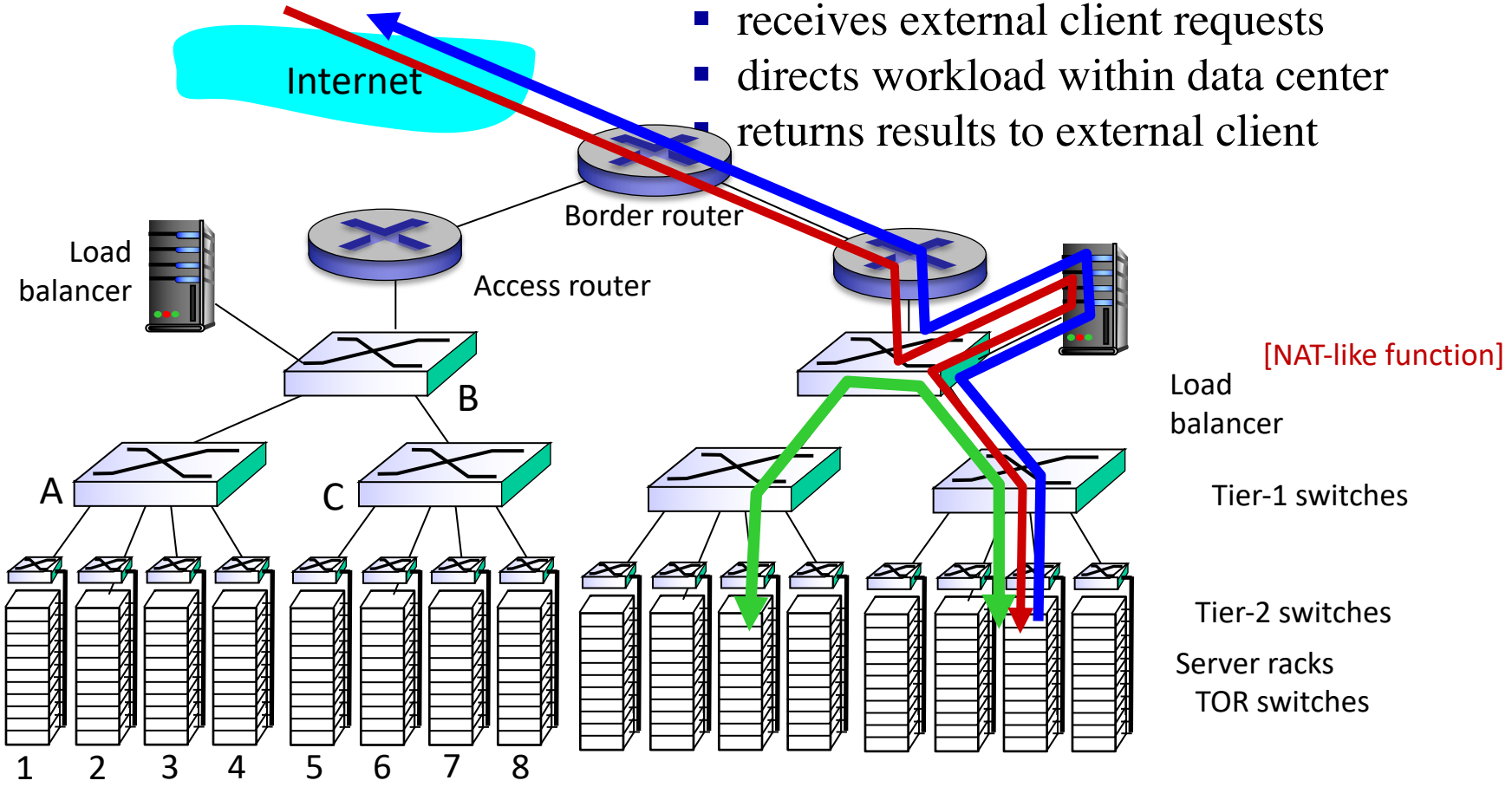
trunk port: carries frames between VLANS defined over multiple physical switches

- frames forwarded within VLAN between switches can't be vanilla 802.1 frames (must carry VLAN ID info)

Data center networks

load balancer: application-layer routing

- receives external client requests
- directs workload within data center
- returns results to external client



- Rich interconnection among switches, racks: increased throughput between racks; increased reliability via redundancy

Chapter 7 outline

7.1 Introduction

Wireless

7.2 Wireless links, characteristics

- CDMA

7.3 IEEE 802.11 wireless LANs (“Wi-Fi”)

7.4 Cellular Internet Access

- architecture
- standards (e.g., 3G, LTE)

Mobility

7.5 Principles: addressing and routing to mobile users

Elements of a wireless network

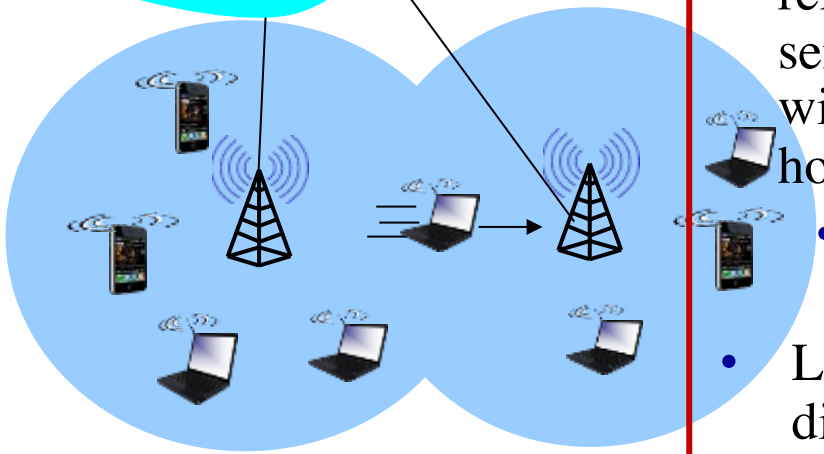
wireless hosts

- run applications
- may be stationary (non-mobile) or mobile
  - wireless does *not* always mean mobility

wireless link

- typically used to connect mobile(s) to base station
- also used as backbone link
- multiple access protocol coordinates link access
- various data rates, transmission distance

network infrastructure



base station

- typically connected to wired network
- relay - responsible for sending packets between wired network and wireless host(s) in its "area"
  - e.g., cell towers, 802.11 access points
- Link layer relay; but different from switches

Infrastructure mode

ad hoc mode

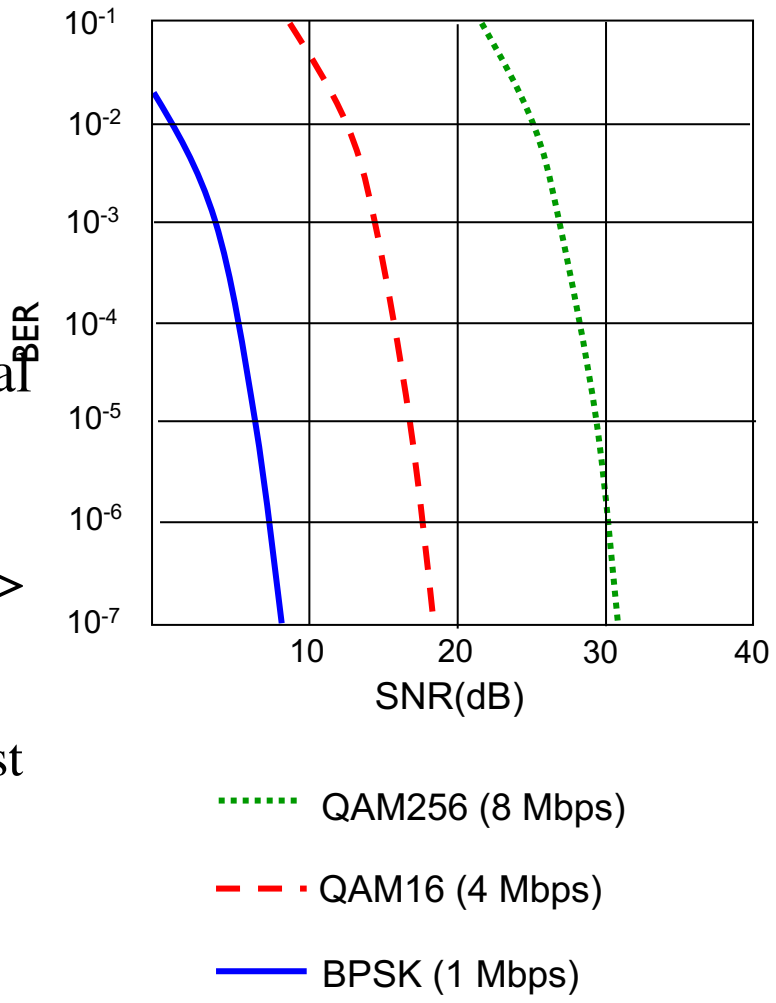
Wireless Link Characteristics

important differences from wired link ….

- *decreased signal strength*
- *interference from other sources*
- *multipath propagation*

- SNR: signal-to-noise ratio
  - larger SNR – easier to extract signal from noise (a "good thing")
- *SNR versus BER tradeoffs*
  - *given physical layer:* increase power -> increase SNR->decrease BER
  - *given SNR:* choose physical layer that meets BER requirement, giving highest thruput

Hidden terminal problem: obstacle, fading -> CDMA/CA



Code Division Multiple Access (CDMA)

Medium access protocols:

Channel partitioning: CDMA, FDMA, TDMA, random access, taking turns

Unique "code" assigned to each user; i.e., code set partitioning

- all users share same frequency, but each user has own "chipping" sequence (i.e., code) to encode data
- allows multiple users to "coexist" and transmit simultaneously with minimal interference (if codes are "orthogonal")

- *encoded signal* = (original data) X (chipping sequence)
- *decoding:* inner-product of encoded signal and chipping sequence

How to compute?

802.11 LAN architecture

- wireless host communicates with base station
  - base station = access point (AP)
- Basic Service Set (BSS) (aka "cell") in infrastructure mode contains:
  - wireless hosts
  - access point (AP): base station
  - ad hoc mode: hosts only
- 802.11b: 2.4GHz-2.485GHz spectrum divided into 11 channels at different frequencies
  - AP admin chooses frequency for AP
  - interference possible: channel can be same as that chosen by neighboring AP!
- host: must *associate* with an AP
  - Passive scanning, active scanning
  - Once associated with an AP, the device will want to join the subnet through sending a DHCP discovery message via the AP.

IEEE 802.11 MAC Protocol: CSMA/CA

802.11: *no* collision detection (why?)

*802.11CSMA/CA protocol: (1) Congestion avoidance (*once start transmission, no turning back*), (2) reliable*

802.11 sender

1 if sense channel idle for distributed inter-frame space (**DIFS**) then transmit entire frame (no CD)

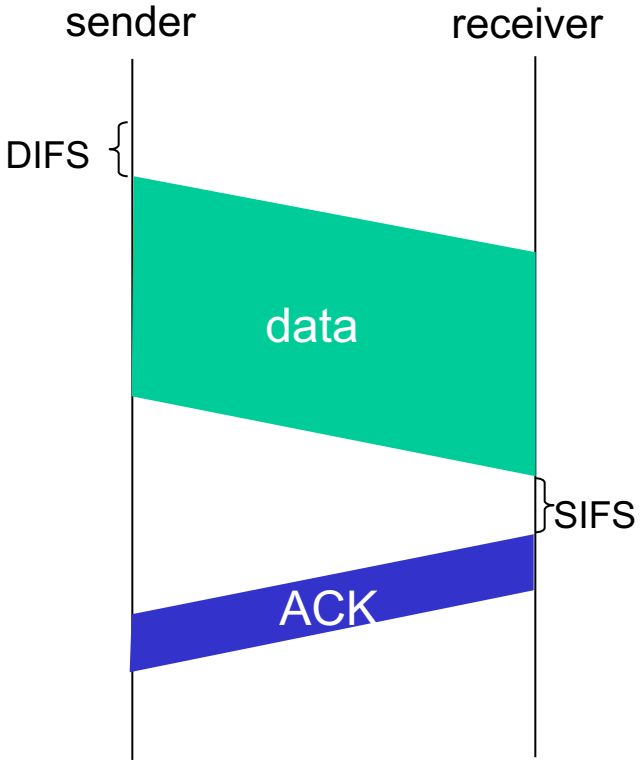
2 if sense channel busy then

start random backoff time (why?)

timer counts down while channel idle (why?)

transmit when timer expires and wait for ACK

if no ACK, increase random backoff interval, repeat 2



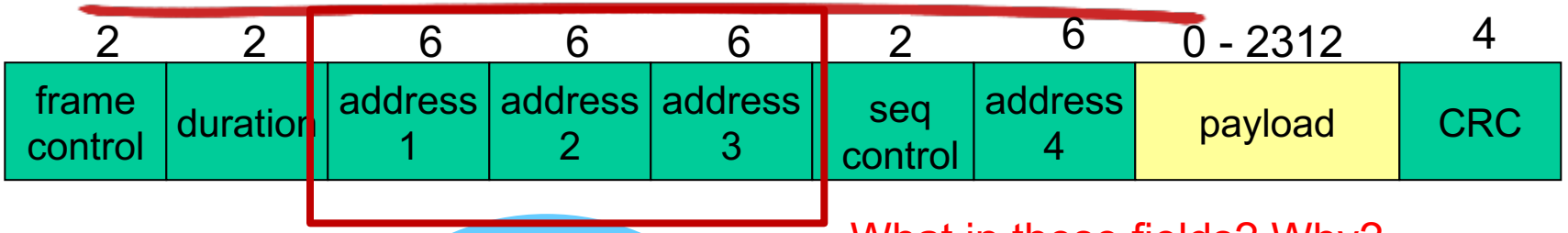
802.11 receiver

- if frame received OK

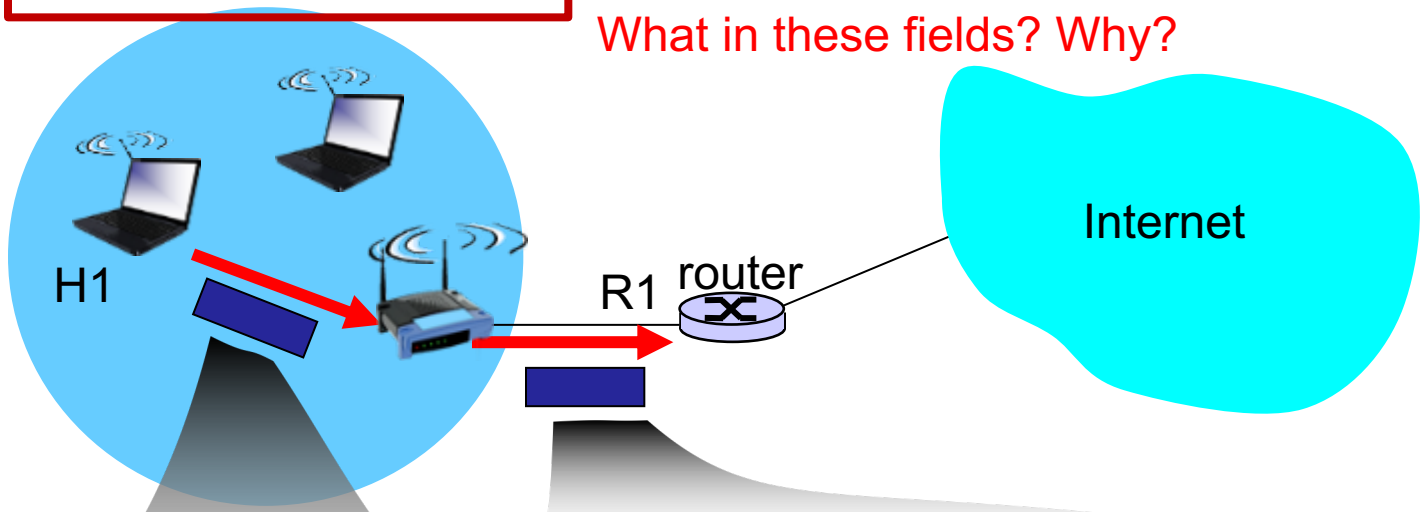
return ACK after **SIFS** (ACK needed due to hidden terminal problem)

May still exist collision: why? how to address? detailed procedure?

802.11 frame: addressing



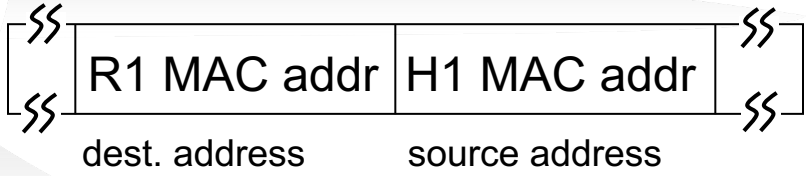
What in these fields? Why?



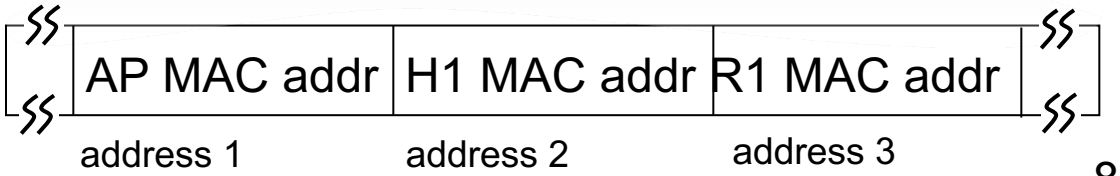
Mobility with same subnet (how switch updates?)

Rate adaptation (how to read figure?)

Power management



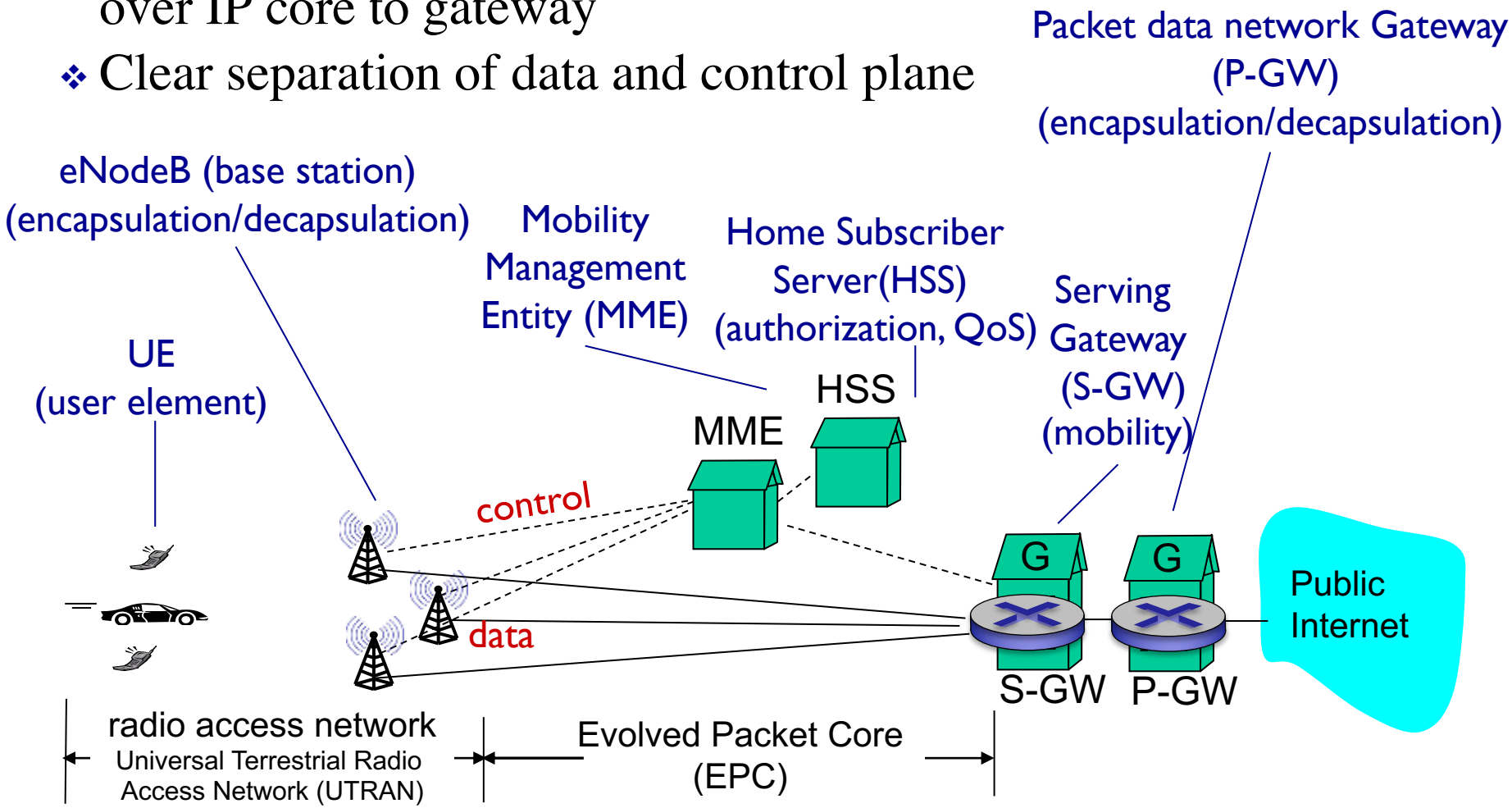
802.3 (Ethernet) frame



802.**11** frame

4G and differences from 3G

- no separation between voice and data – all traffic carried over IP core to gateway
- Clear separation of data and control plane



Mobility

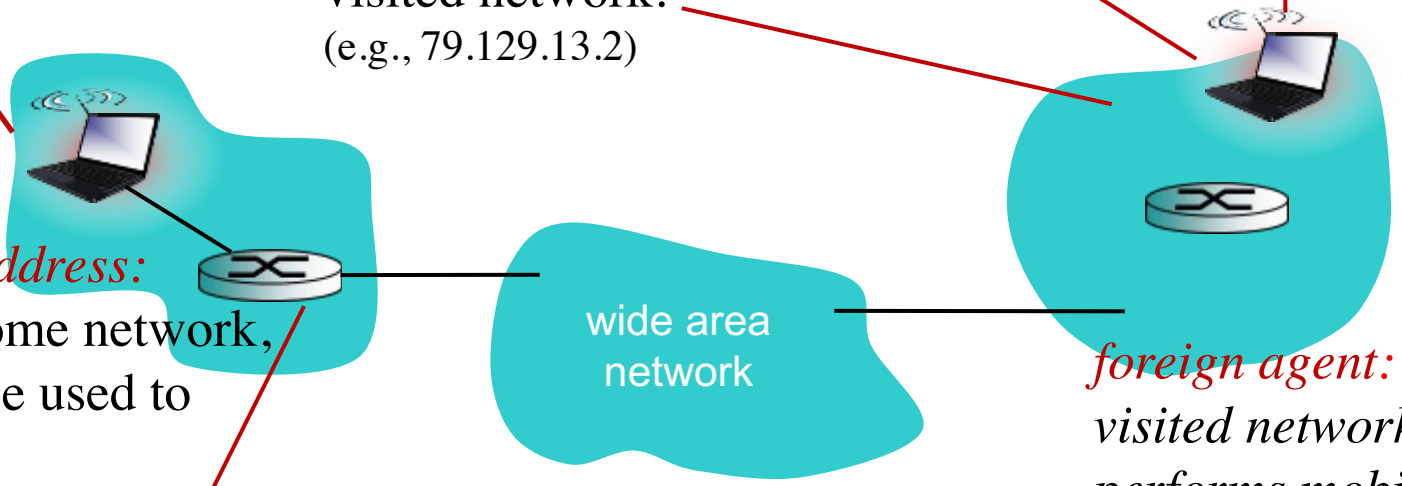
What is mobility, from the *network* perspective?

permanent address: remains constant (e.g., 128.119.40.186)

visited network: network in which mobile currently resides (e.g., 79.129.13/24)

home network: permanent "home" of mobile (e.g., 128.119.40/24)

care-of-address: address in visited network. (e.g., 79.129.13.2)



permanent address: address in home network, *can always* be used to reach mobile e.g., 128.119.40.186

foreign agent: *entity in visited network that performs mobility functions on behalf of mobile.*

home agent: *entity that will perform mobility functions on behalf of mobile, when mobile is remote*

correspondent: *wants to communicate with mobile*

Mobility: approaches

- ***let routing handle it:*** foreign agent advertises permanent address of mobile-nodes-in-residence via usual routing table exchange.
  - routing tables indicate where each mobile located
  - no changes to end-systems
  - Drawbacks?
- ***let end-systems handle it:***
  - *indirect routing:* communication from correspondent to mobile goes through home agent, then forwarded to remote (detailed procedure?)
    - Drawbacks? triangle routing
  - *direct routing:* correspondent gets foreign address of mobile, sends directly to mobile (detailed procedure?)
    - what if mobile changes visited network?