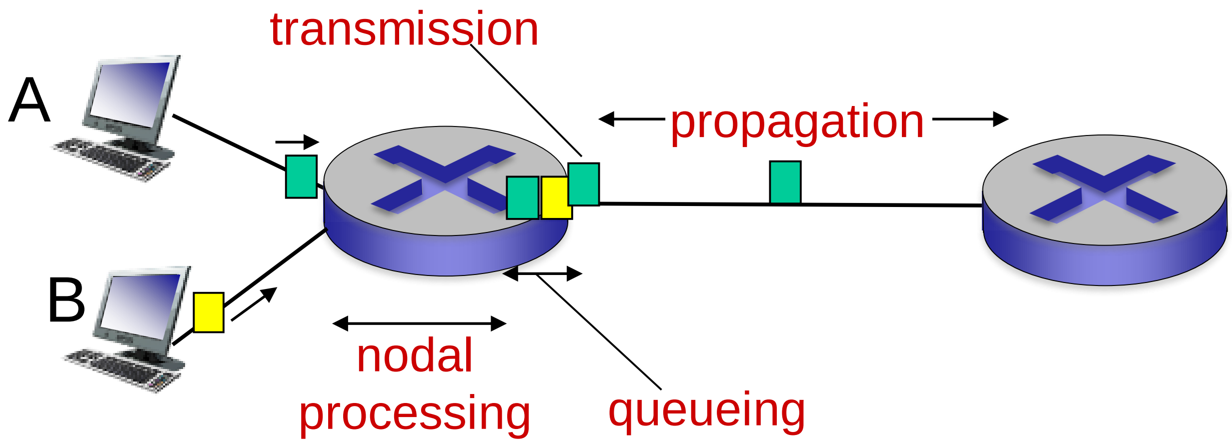
CS 3800: Computer Networks

Final Exam Review

Instructor: John Korah

Four sources of packet delay



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

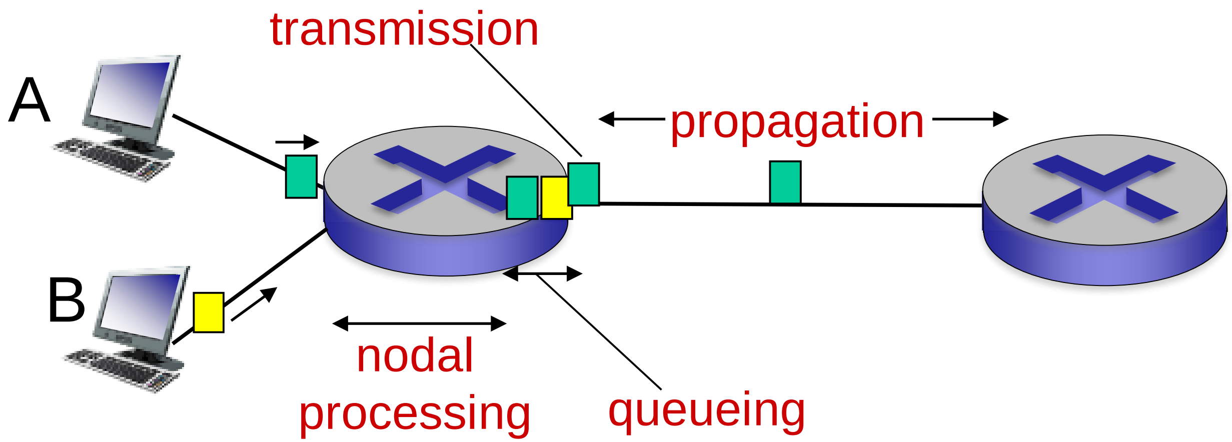
d_{proc}: nodal processing

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

d_{queue}: queueing delay

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

Four sources of packet delay



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

d_{trans}: transmission delay:

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

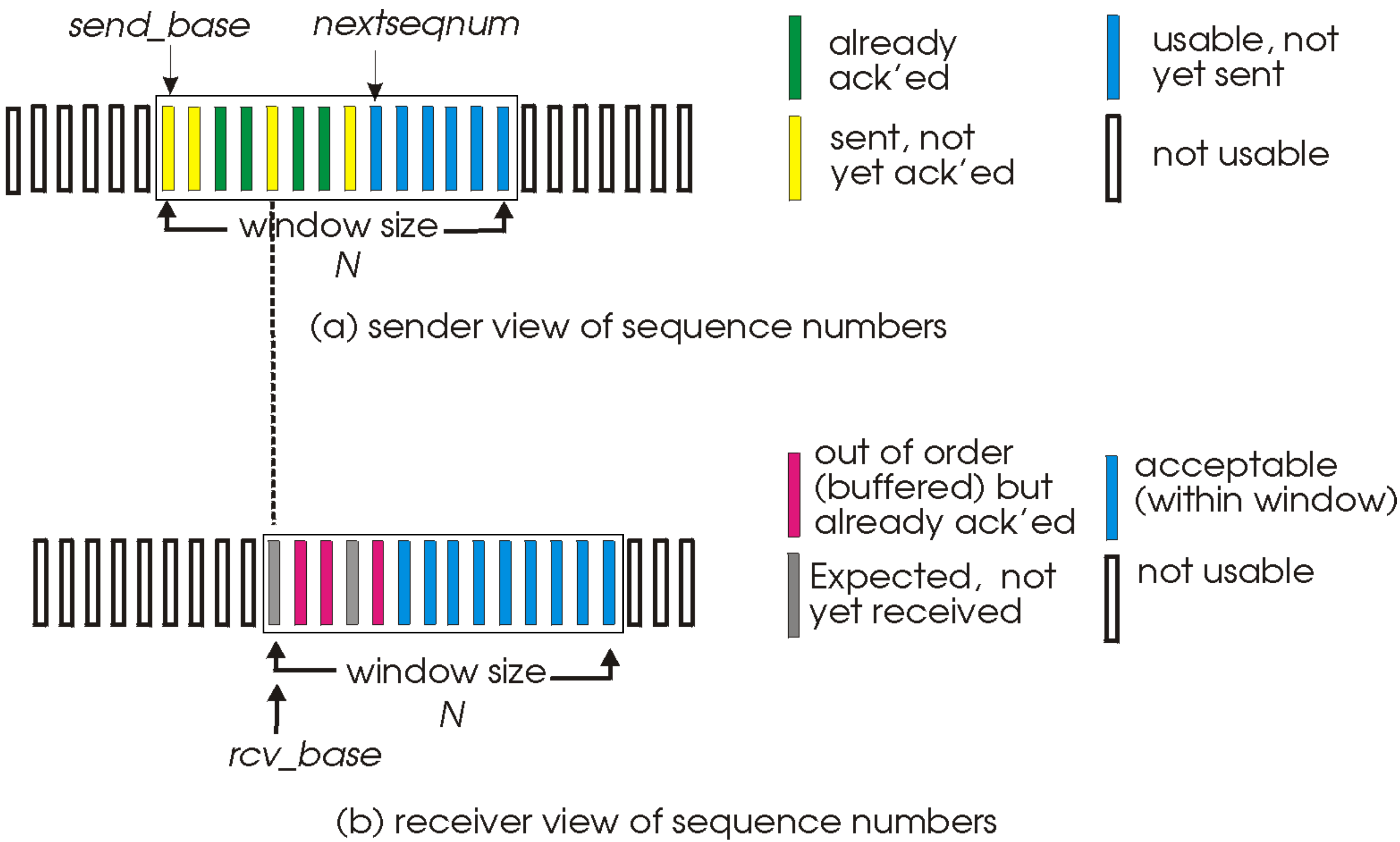
d_{trans} and d_{prop} *very* different

d_{prop}: propagation delay:

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

- Check out the Java applet for an interactive animation on trans vs. prop delay here

Selective repeat: sender, receiver windows



Selective repeat

sender

data from above:

- if next available seq # in window, send pkt

timeout(n):

- resend pkt n, restart timer

ACK(n) in [sendbase,sendbase+N]:

- mark pkt n as received
- if n smallest unACKed pkt, advance window base to next unACKed seq #

receiver

pkt n in [rcvbase, rcvbase+N-1]

- send ACK(n)
- out-of-order: buffer
- in-order: deliver (also deliver buffered, in-order pkts), advance window to next not-yet-received pkt

pkt n in [rcvbase-N,rcvbase-1]

- ACK(n)

otherwise:

- ignore

TCP seq. numbers, ACKs

sequence numbers:

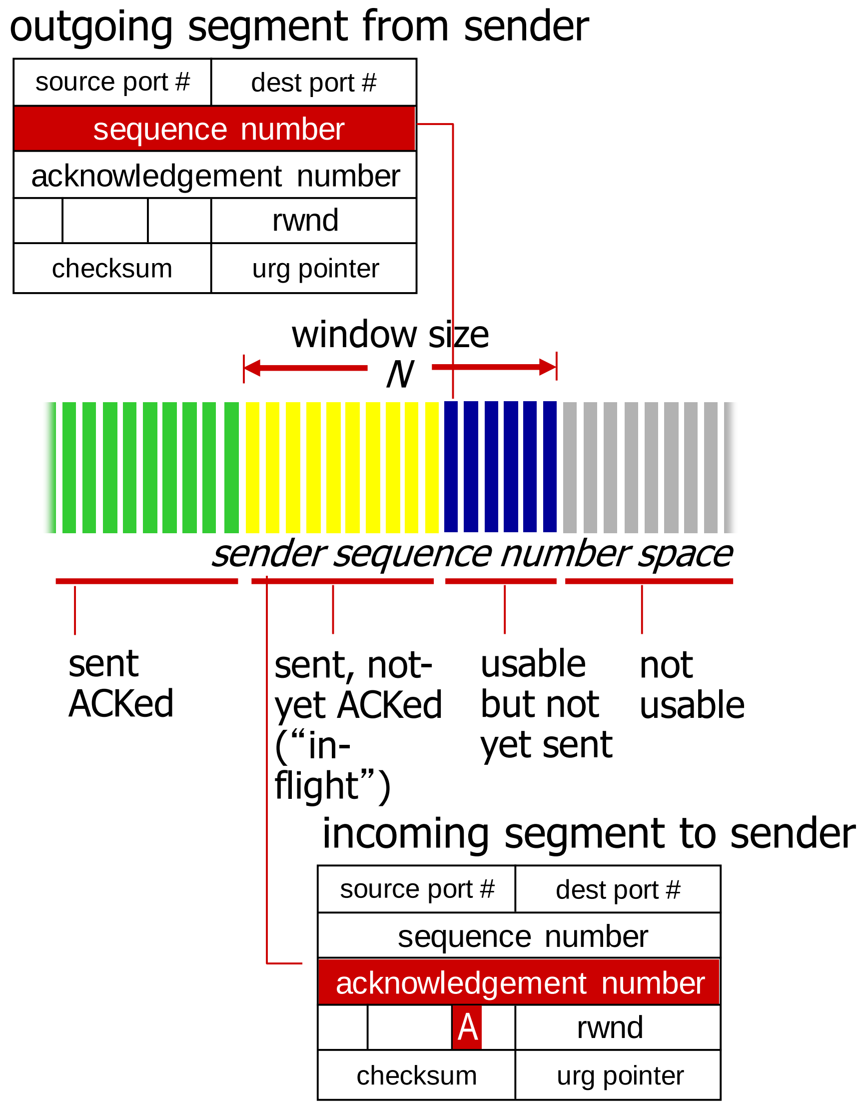
- byte stream "number" of first byte in segment's data

acknowledgements:

- seq # of next byte expected from other side
- cumulative ACK

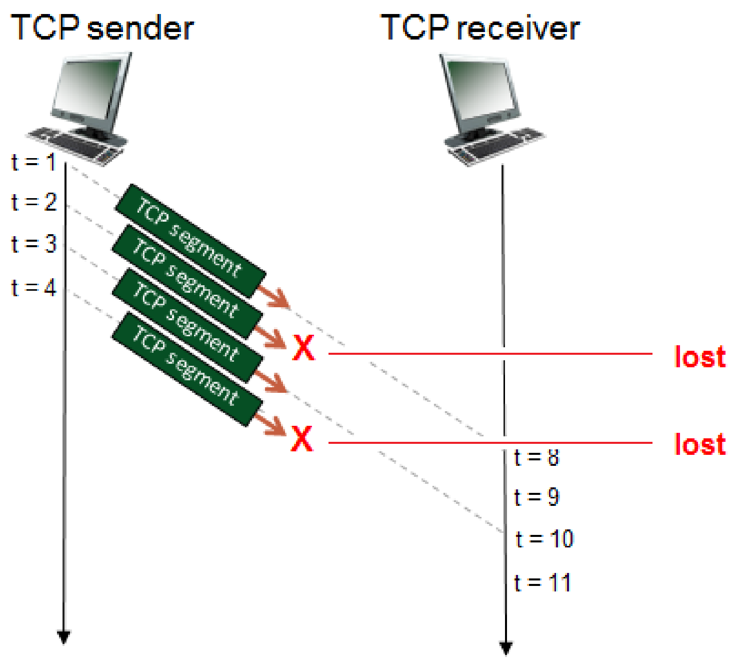
Q: how receiver handles out-of-order segments

- A: TCP spec doesn't say, - up to implementor



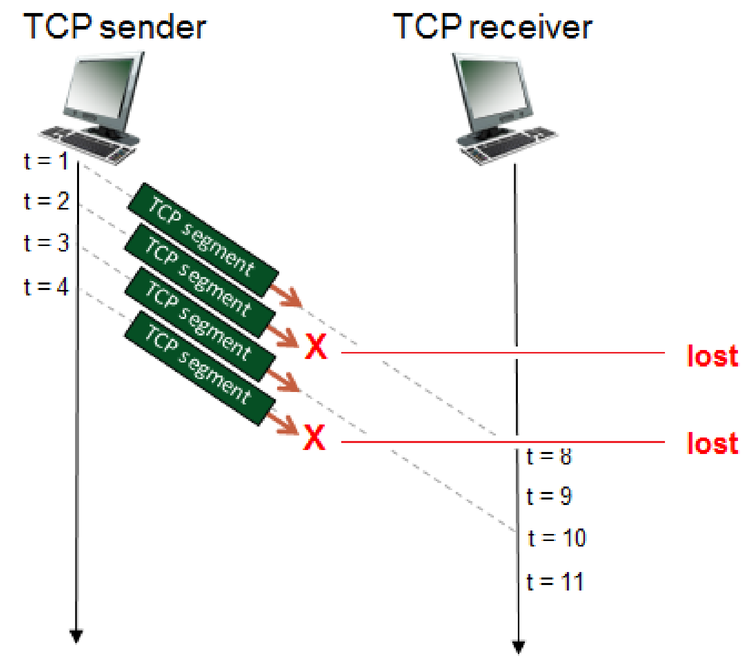
Problem

- Consider the figure below in which TCP a sender and receiver communicate over a connection in which the sender-to-receiver segments may be lost.



- The TCP sender sends initial window of four segments at t=1,2,3,4, respectively. Suppose the initial value of the sender-to-receiver sequence number is 123 and the first four segments *each* contain 575 bytes. The delay between the sender and the receiver is 7 time units, and so the first segment arrives at the receiver at t=8. As shown in the figure, two of the four segment(s) are lost between the sender and the receiver.

Problem



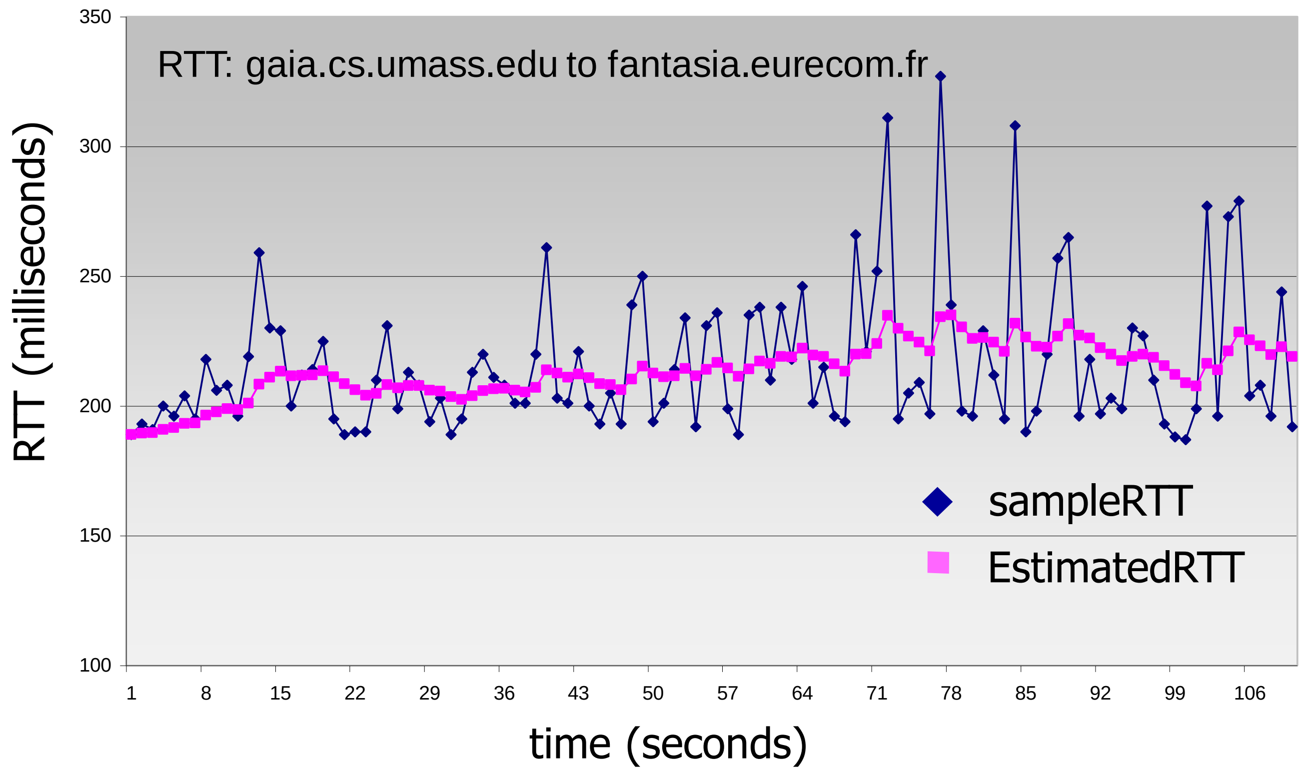
- Consider the figure below in which TCP a sender and receiver communicate over a connection in which the sender-to-receiver segments may be lost.

- Answer the following questions:
  - Give the sequence numbers associated with each of the four segments sent by the sender
  - List the sequence of acknowledgements transmitted by the TCP receiver in response to the receipt of the segments actually received. In particular, give the value in the acknowledgement field of each receiver-to-sender acknowledgement, and give a brief explanation as to why that particular acknowledgement number value is being used

TCP round trip time, timeout

`EstimatedRTT = (1- α)*EstimatedRTT + α*SampleRTT`

- exponential weighted moving average
- influence of past sample decreases exponentially fast
- typical value: $\alpha = 0.125$



TCP round trip time, timeout

- timeout interval: **`EstimatedRTT`** plus "safety margin"
  - large variation in **`EstimatedRTT`** **`->`** larger safety margin
- estimate SampleRTT deviation from EstimatedRTT:

```
DevRTT = (1-β)*DevRTT +
              β*|SampleRTT-EstimatedRTT|

              (typically, β = 0.25)
```

```
TimeoutInterval = EstimatedRTT + 4*DevRTT
```

estimated RTT

"safety margin"

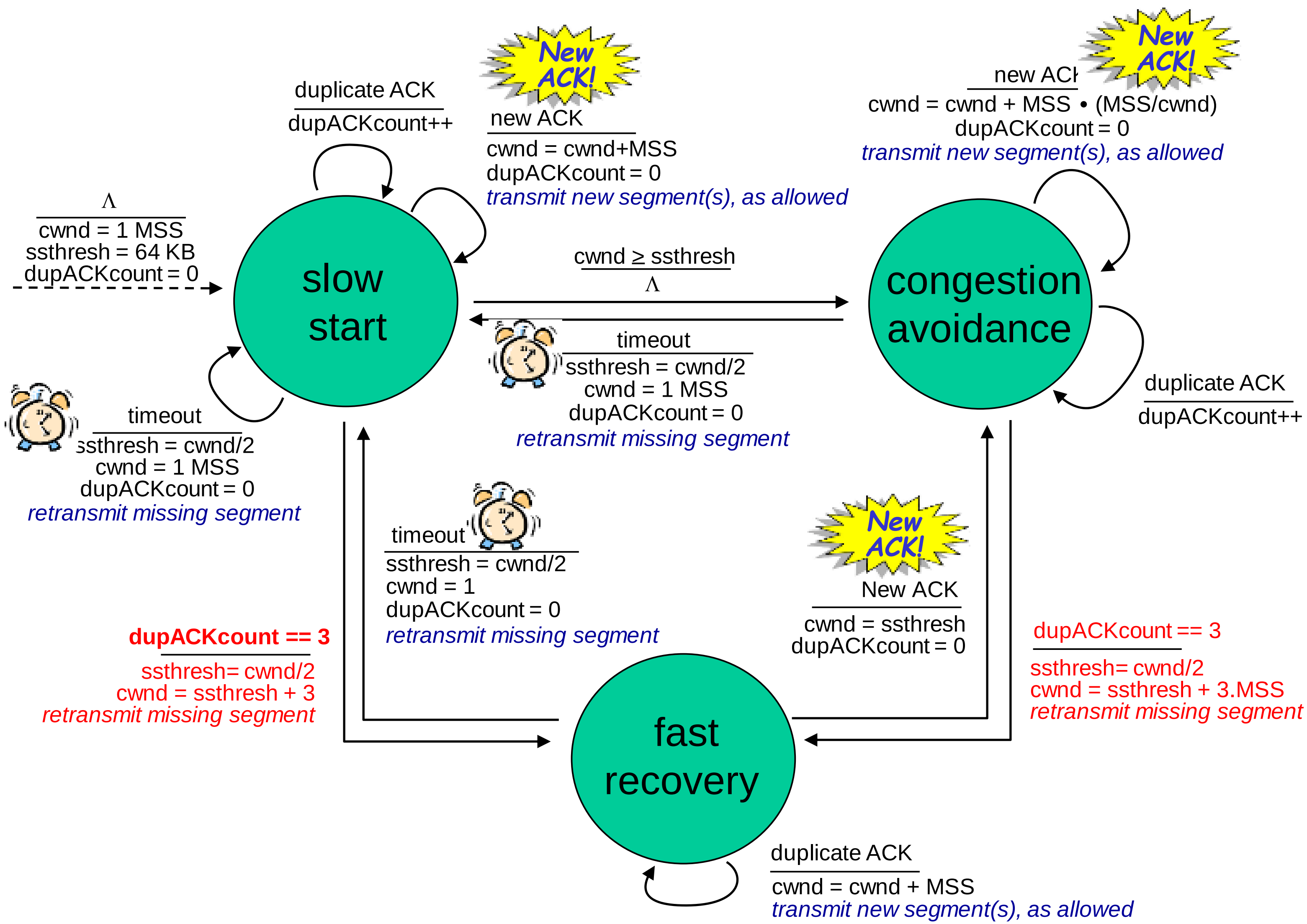
* Check out the online interactive exercises for more examples: http://gaia.cs.umass.edu/kurose_ross/interactive/

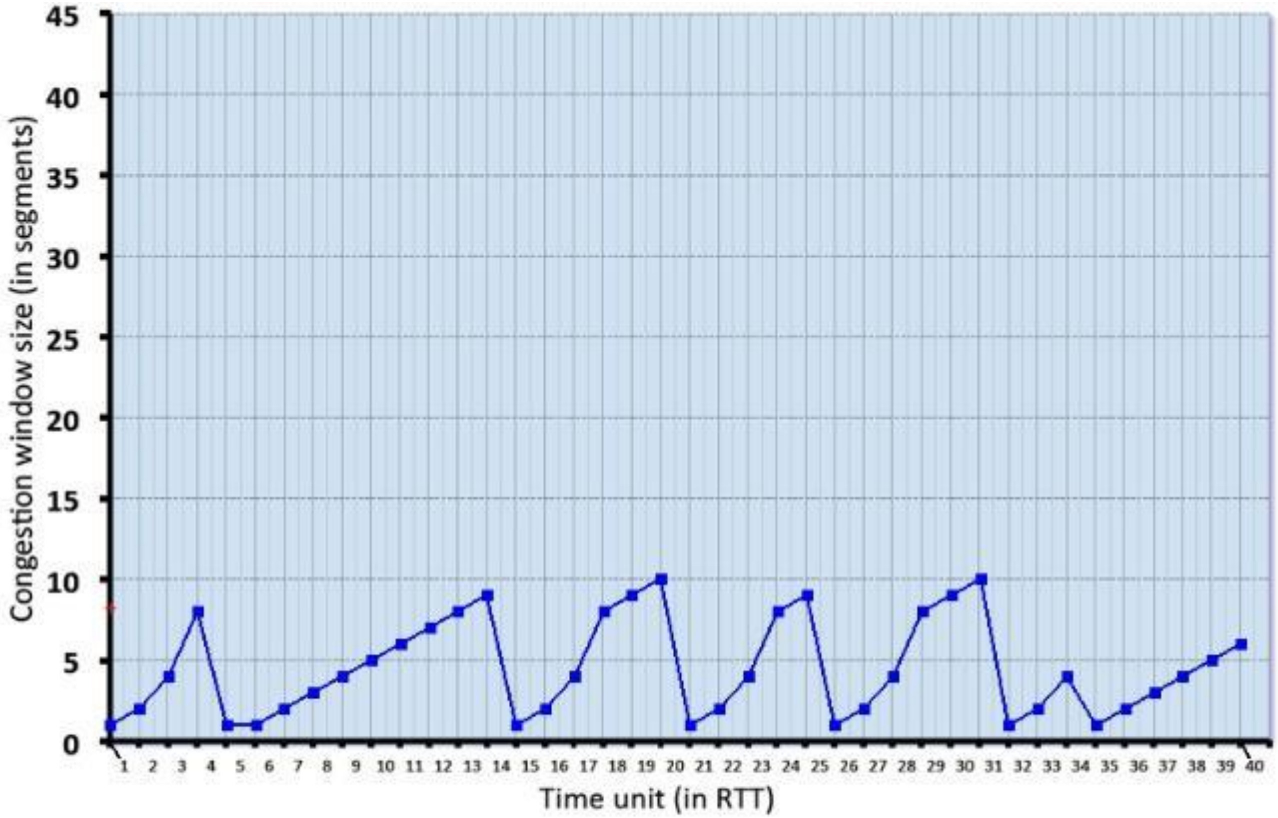
TCP round trip time, timeout

Suppose that TCP's current estimated values for the round trip time (`estimatedRTT`) and deviation in the RTT (`DevRTT`) are 360 msec and 39 msec, respectively. Suppose that the next three measured values of the RTT are 260, 340, and 260 respectively.

Compute TCP's new value of `estimatedRTT, DevRTT`, and the TCP timeout value after each of these three measured RTT values is obtained.Use the values of $\alpha = 0.125$ and $\beta = 0.25$.

Recap: TCP Congestion Control



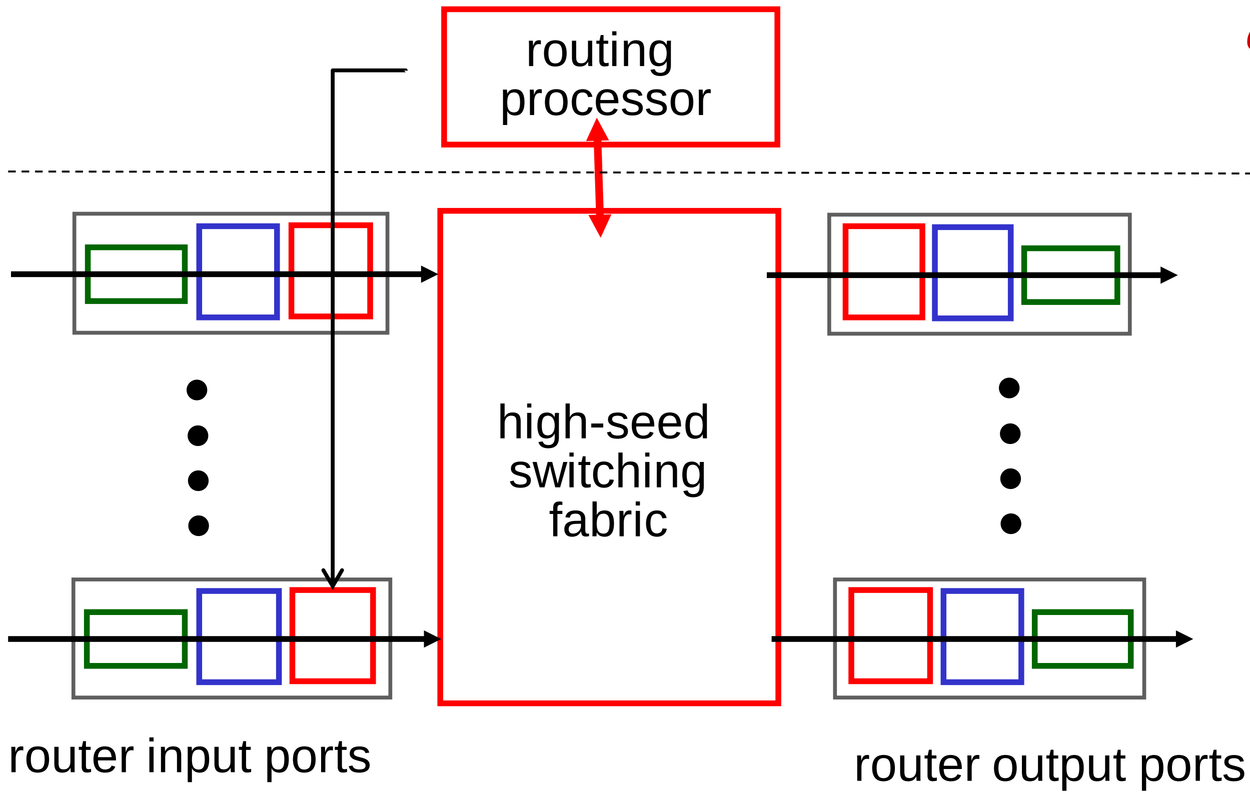


The result of sending that flight of packets is that either *(i)* all packets are ACKed at the end of the time unit, *(ii)* there is a timeout for the first packet, or *(iii)* there is a triple duplicate ACK for the first packet.

- Give the times at which TCP is in slow start, congestion avoidance and fast recovery at the start of a time slot, when the flight of packets is sent.
- Give the times at which the first packet in the sent flight of packets is lost, and indicate whether that packet loss is detected via timeout, ot by triple duplicate ACKs

Router architecture overview

- high-level view of generic router architecture:



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

forwarding data plane (hardware) operates in nanosecond timeframe

Longest prefix matching

longest prefix matching

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

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

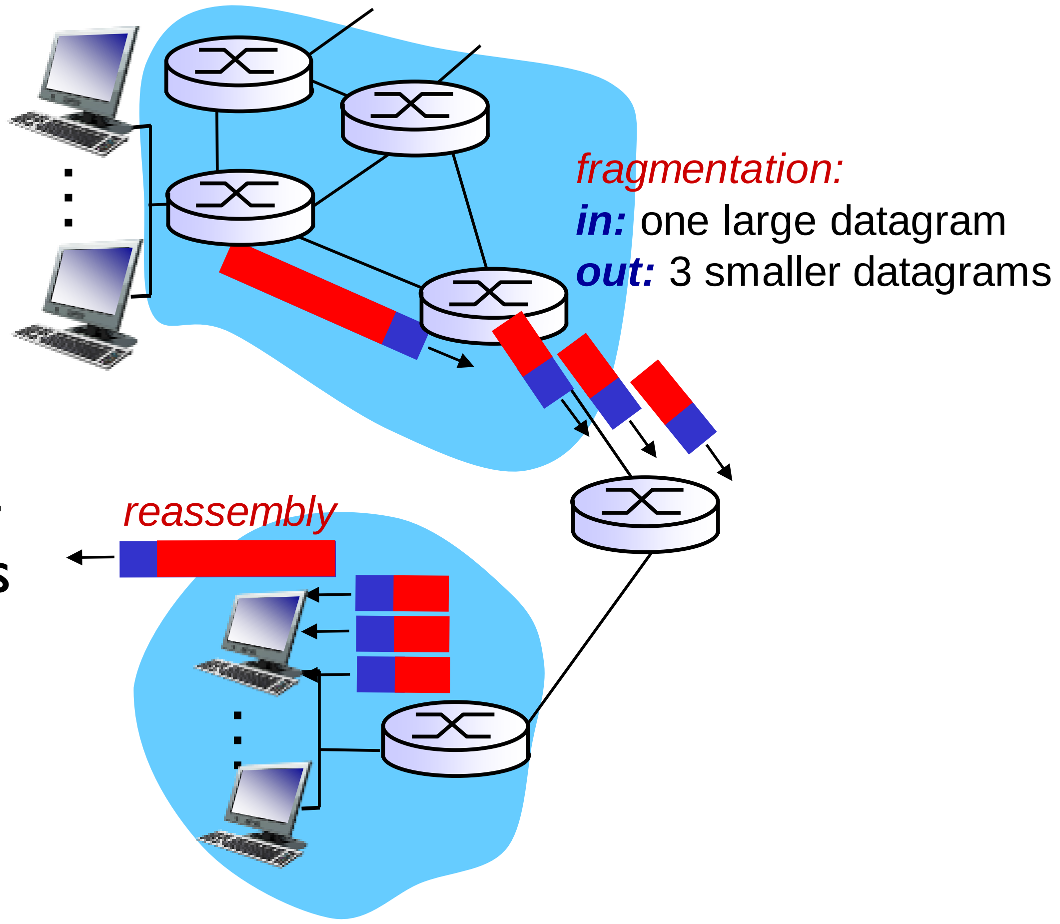
examples:

DA: 11001000 00010111 00010110 10100001 which interface?

DA: 11001000 00010111 00011000 10101010 which interface?

IP fragmentation, reassembly

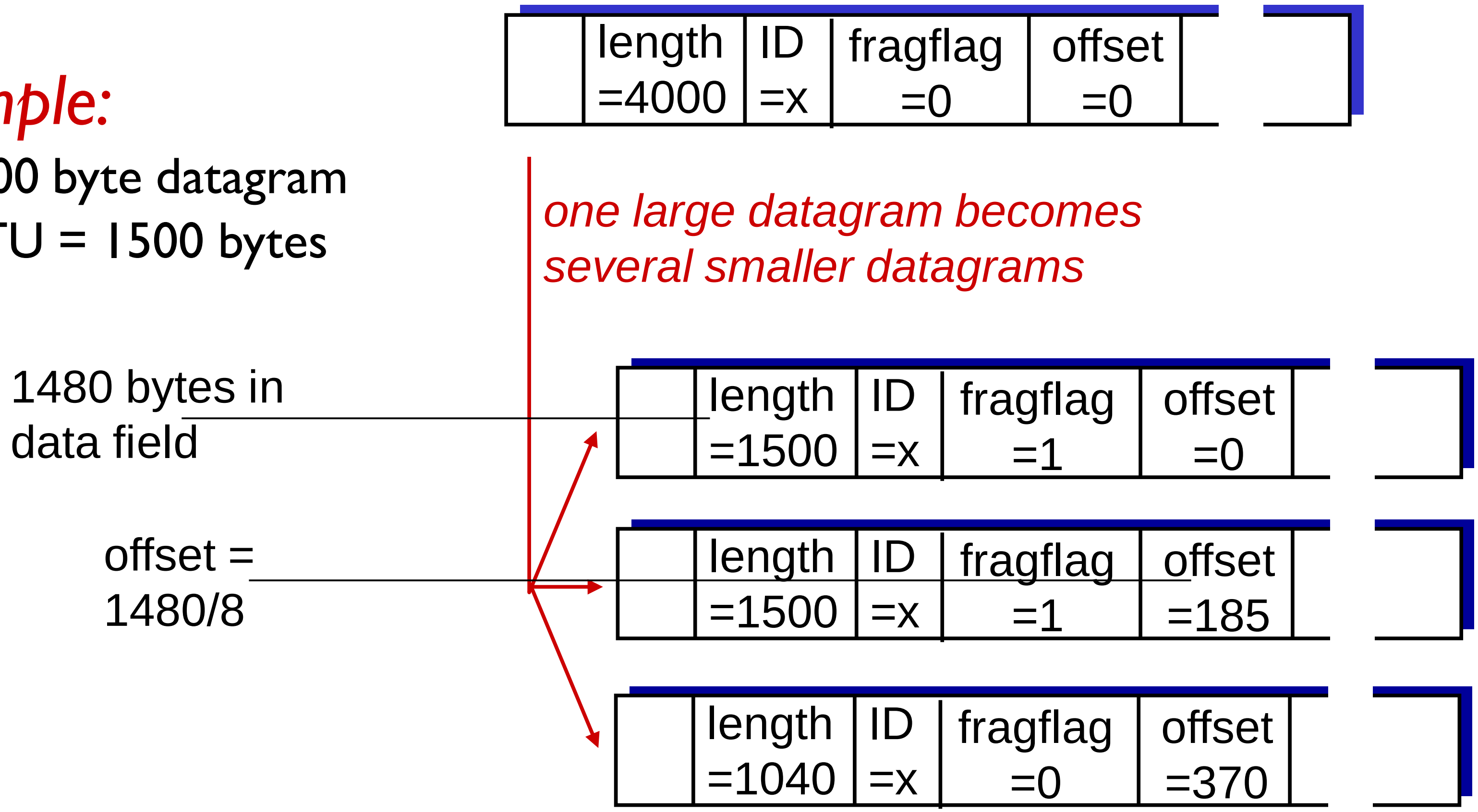
- network links have MTU (max.transfer size) - largest possible link-level frame
  - different link types, different MTUs
- large IP datagram divided ("fragmented") within net
  - one datagram becomes several datagrams
  - **"reassembled" only at final destination**
  - IP header bits used to identify, order related fragments



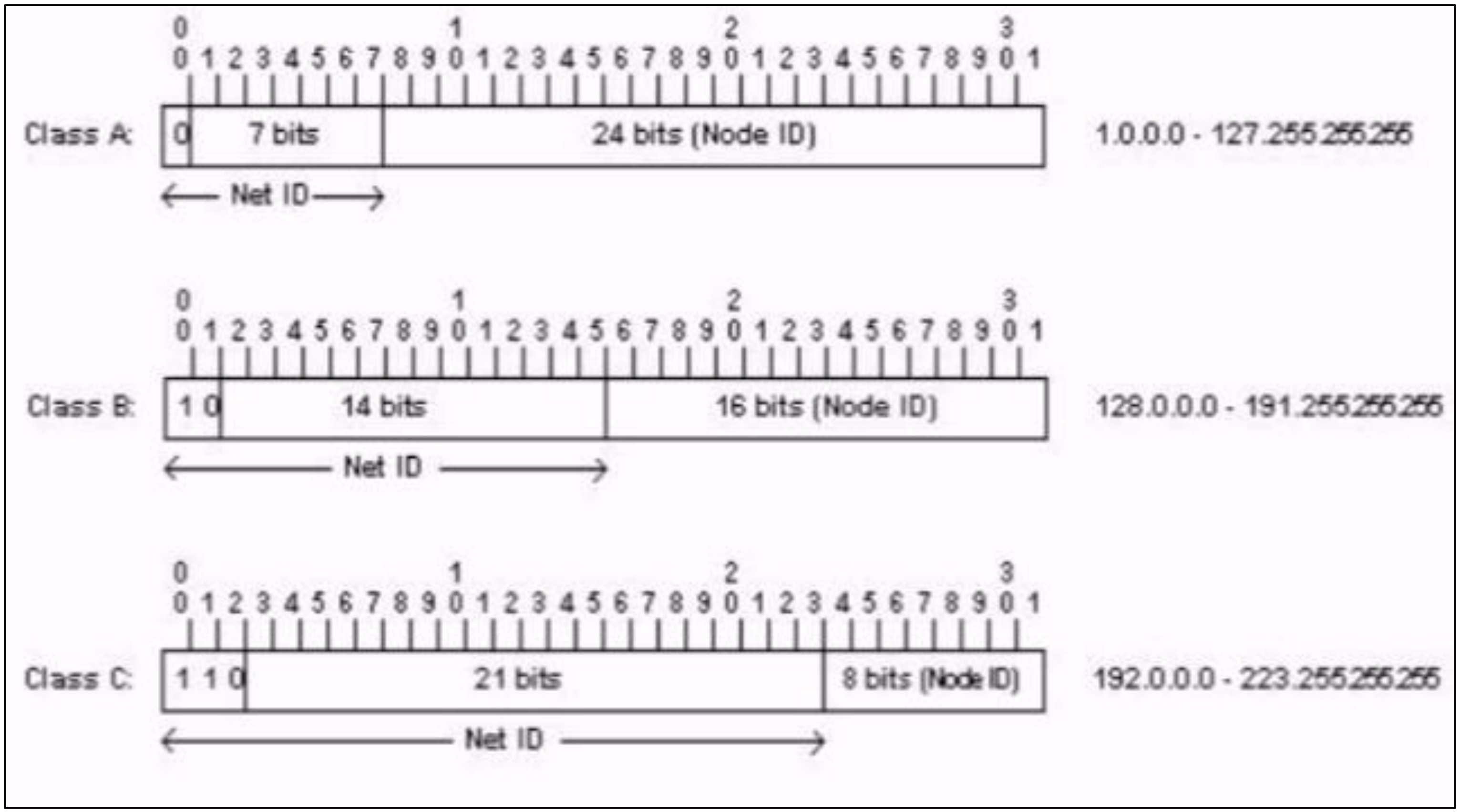
IP fragmentation, reassembly

example:

- 4000 byte datagram
- MTU = 1500 bytes



IP Address Classes



IP addressing: CIDR

CIDR: Classless InterDomain Routing

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

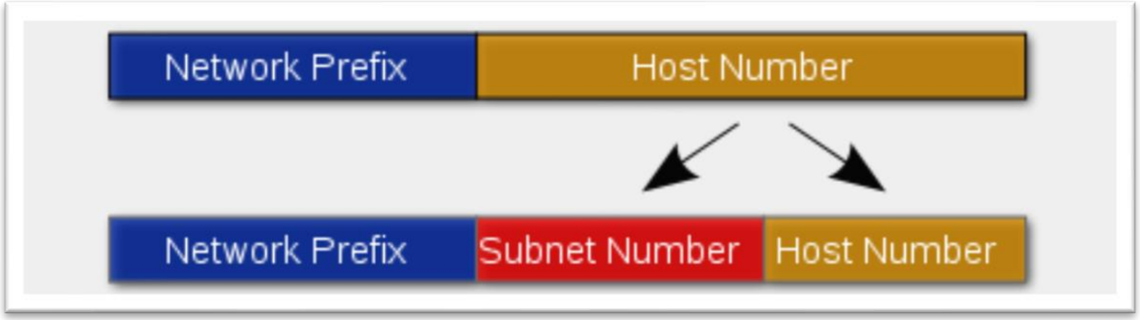
subnet part | host part

11001000 00010111 00010000 00000000

200.23.16.0/23

Subnetted Networks

- The network portion of the address is **<u>extended</u>** by splitting up the host number

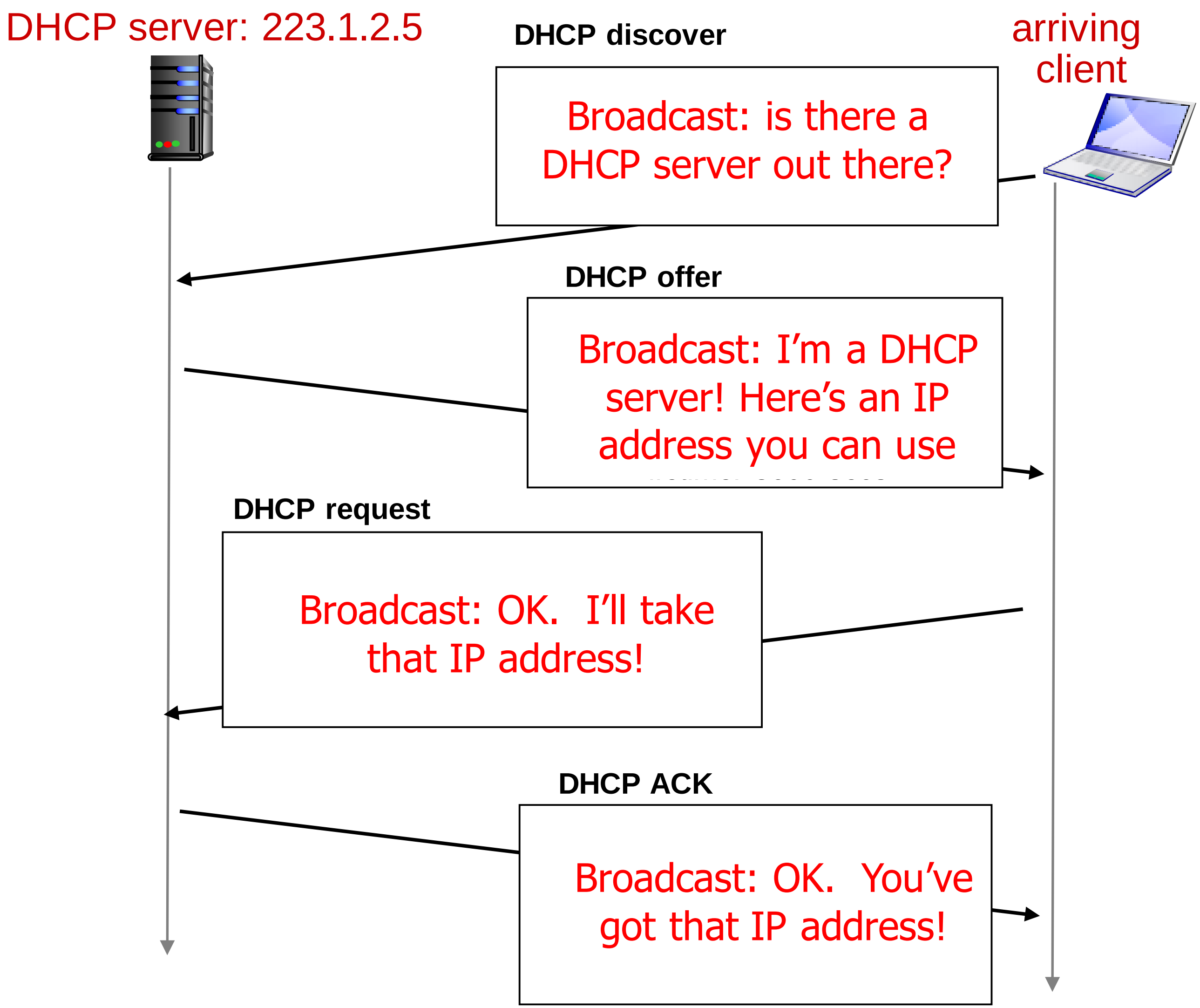


- Borrowing 1 or more bits from the host bit portion

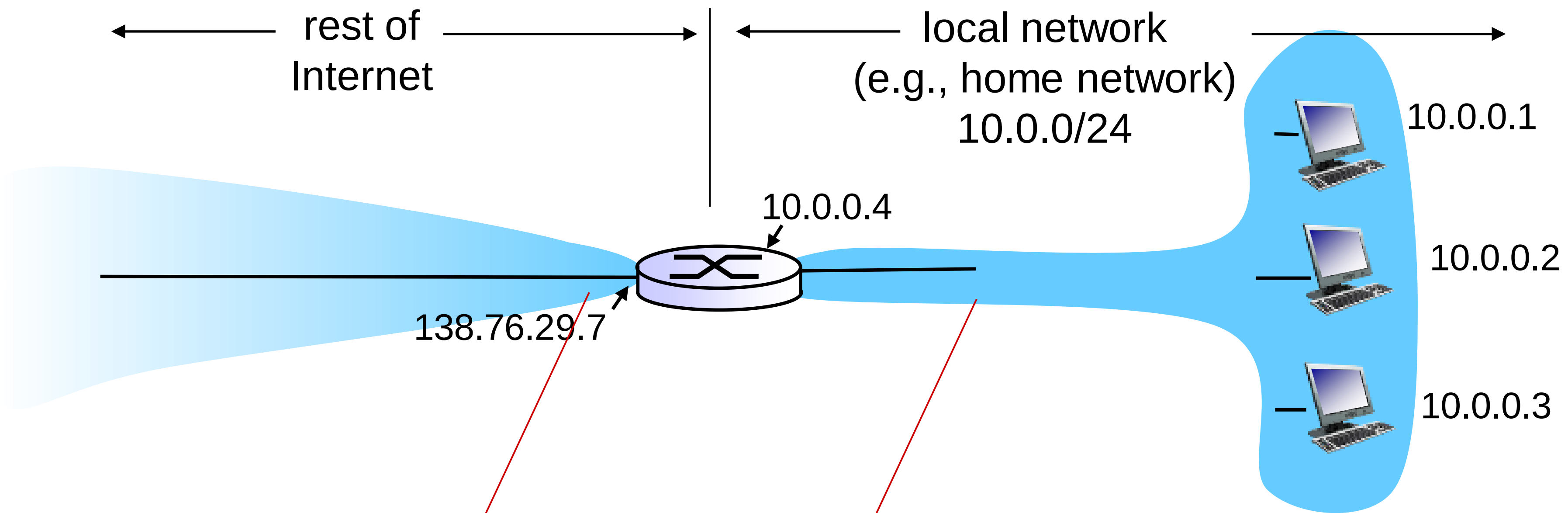
Class C Subnetting

# of Subnets	# of Hosts/Subnet	NetMask	4th Octet	CIDR Notation
2	126	255.255.255.128	10000000	/25
4	62	255.255.255.192	11000000	/26
8	30	255.255.255.224	11100000	/27
16	14	255.255.255.240	11110000	/28
32	6	255.255.255.248	11111000	/29
64	2	255.255.255.252	11111100	/30

DHCP client-server scenario



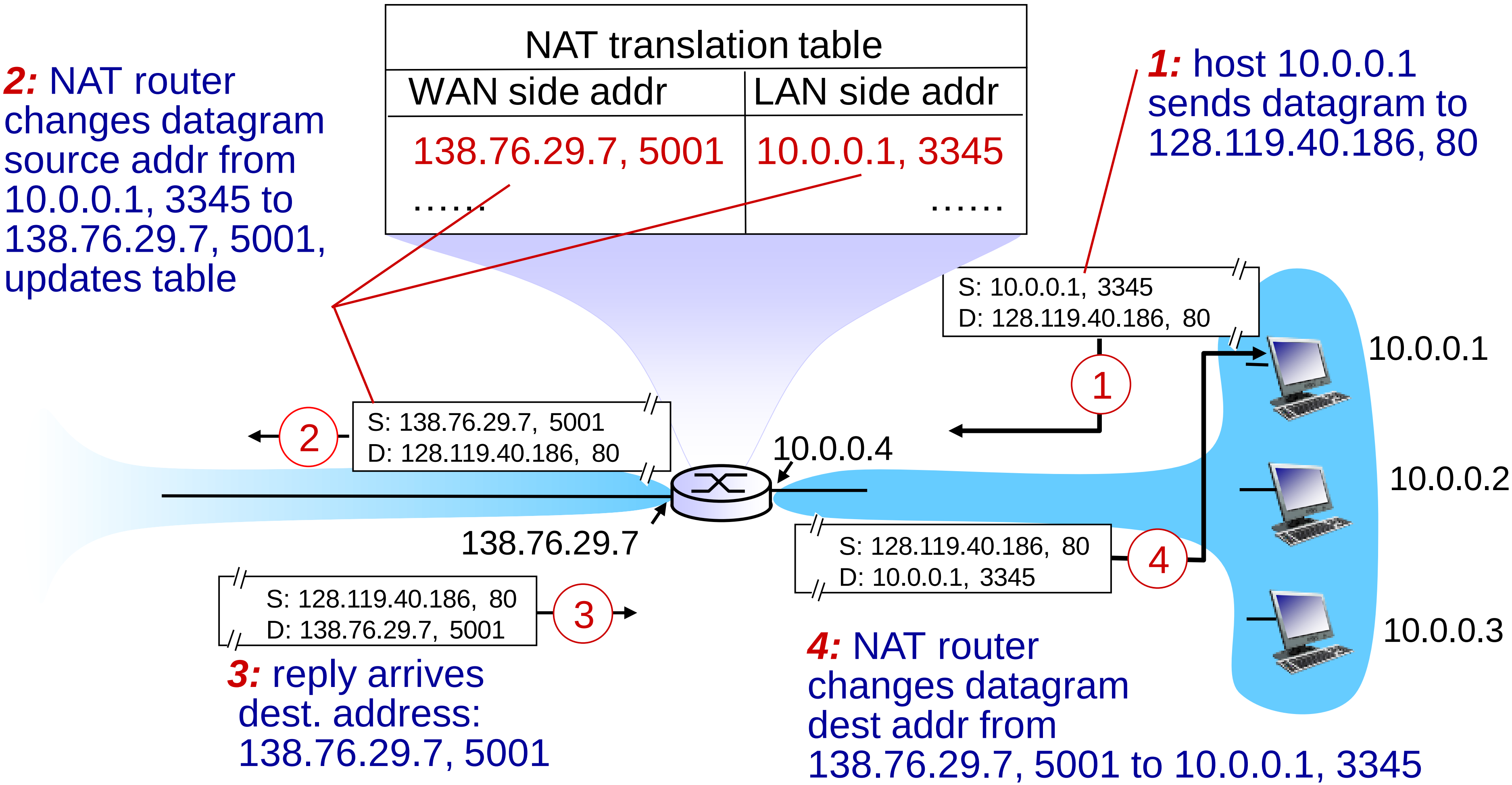
NAT: network address translation



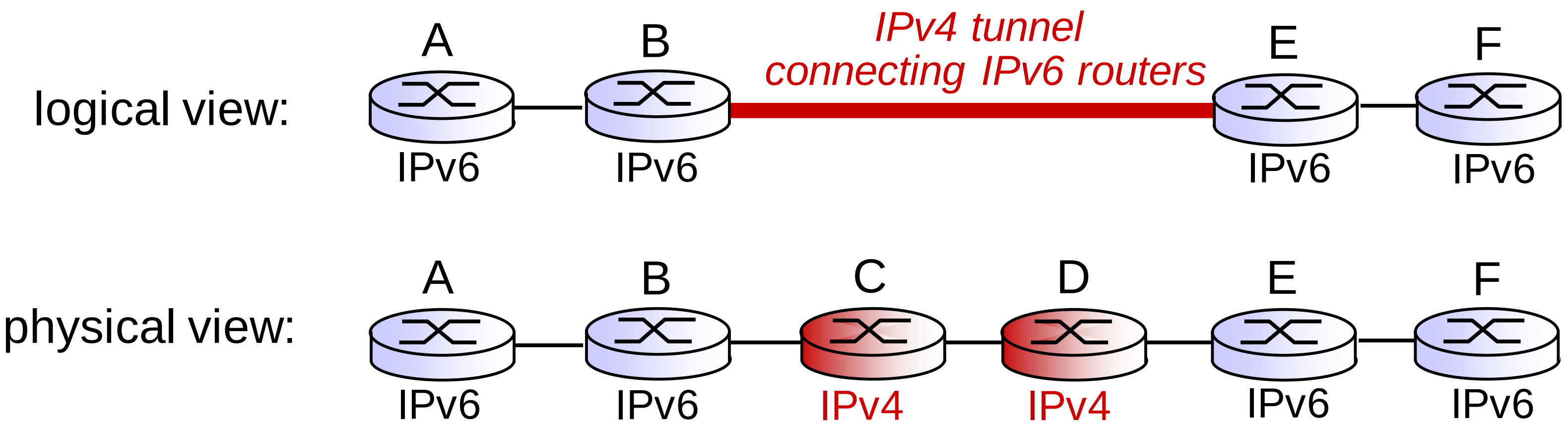
all datagrams *leaving* local network have *same* single source NAT IP address: 138.76.29.7,different source port numbers

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

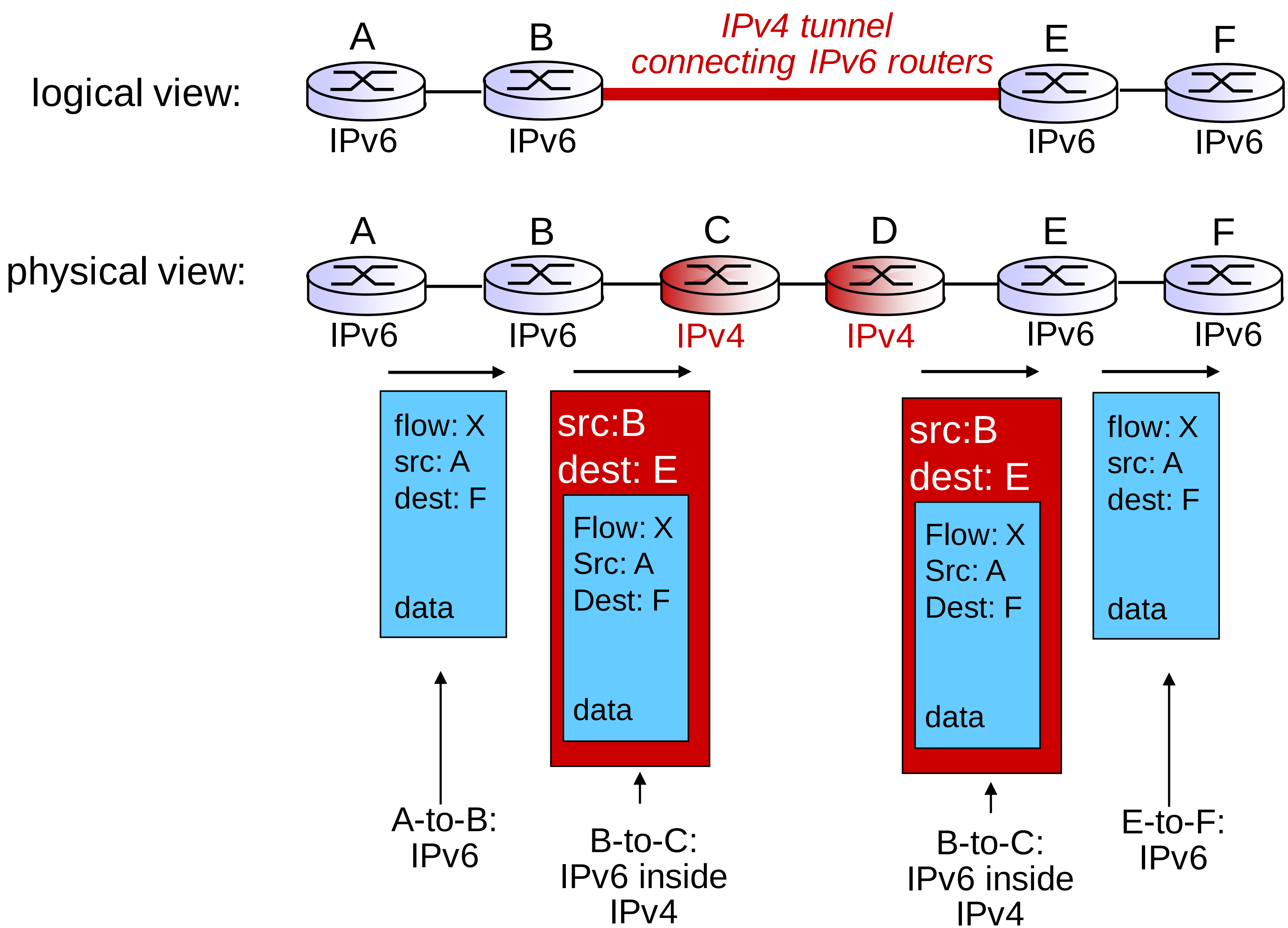
NAT: network address translation



Tunneling


logical view:
A
B
IPv4 tunnel
connecting IPv6 routers
E
F
IPv6
IPv6
IPv6
IPv6
physical view:
A
B
C
D
E
F
IPv6
IPv6
IPv4
IPv4
IPv6
IPv6

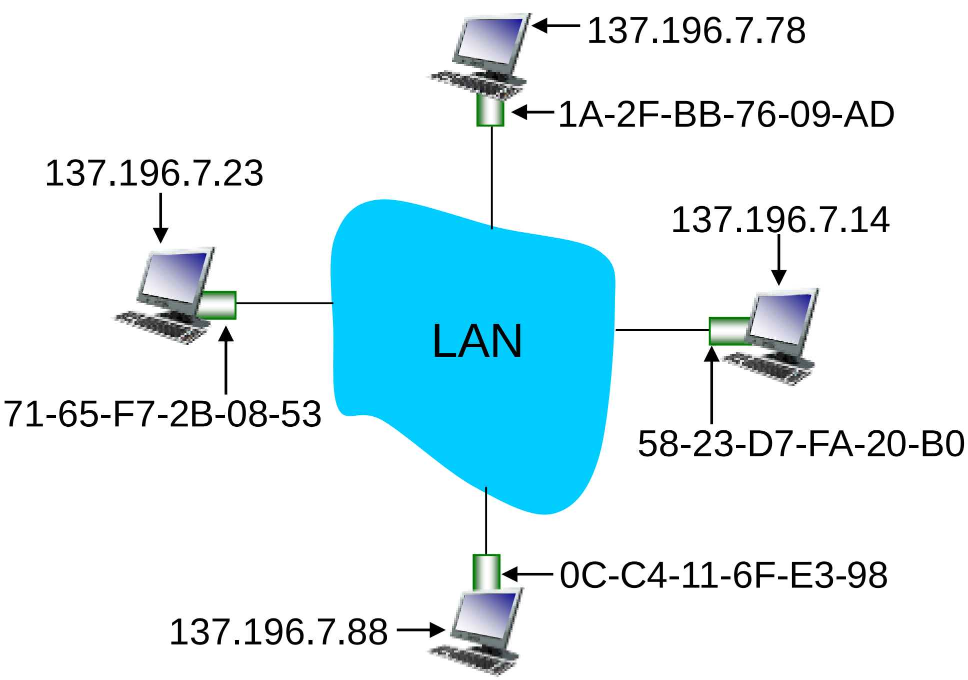
Tunneling



logical view:
A
B
IPv4 tunnel
connecting IPv6 routers
E
F
IPv6
IPv6
IPv6
IPv6
physical view:
A
B
C
D
E
F
IPv6
IPv6
IPv4
IPv4
IPv6
IPv6
flow: X
src: A
dest: F
data
src:B
dest: E
Flow: X
Src: A
Dest: F
data
src:B
dest: E
Flow: X
Src: A
Dest: F
data
flow: X
src: A
dest: F
data
A-to-B:
IPv6
B-to-C:
IPv6 inside
IPv4
B-to-C:
IPv6 inside
IPv4
E-to-F:
IPv6

Address Resolution Protocol (ARP)

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



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

- IP/MAC address mappings for some LAN nodes:

  < IP address; MAC address; TTL>

- TTL (Time To Live): time after which address mapping will be forgotten (typically 20 min)

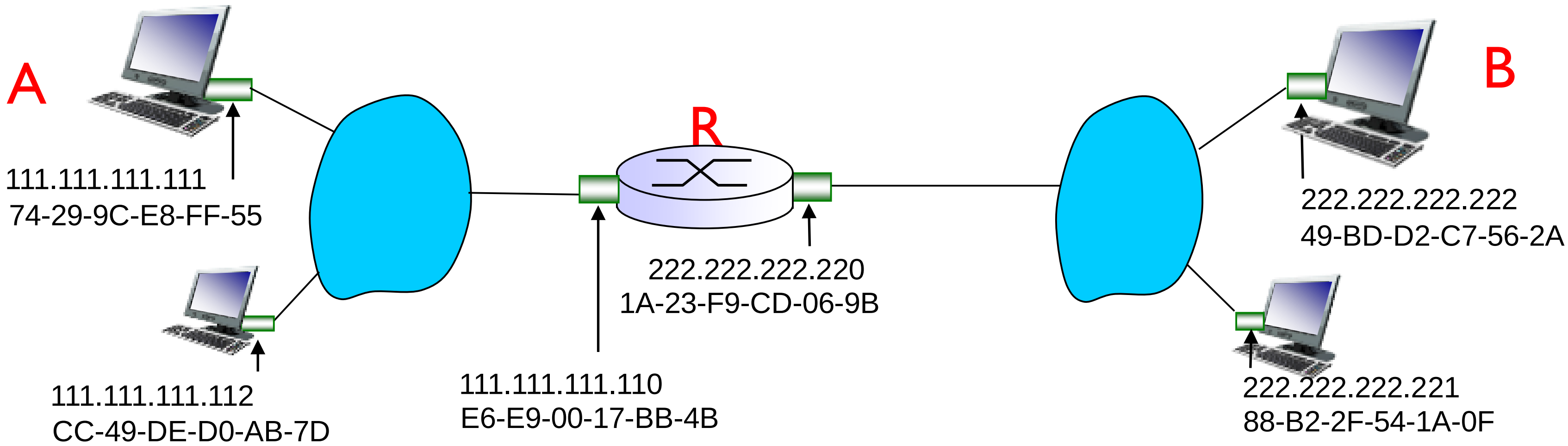
ARP protocol: same LAN

- A wants to send datagram to B
  - B's MAC address not in A's ARP table.
- A broadcasts ARP query packet, containing B's IP address
  - destination MAC address = FF-FF-FF-FF-FF-FF
  - all nodes on LAN receive ARP query
- B receives ARP packet, replies to A with its (B's) MAC address
  - frame sent to A's MAC address (unicast)
- A caches (saves) IP-to-MAC address pair in its ARP table until information becomes old (times out)
  - soft state: information that times out (goes away) unless refreshed
- ARP is "plug-and-play":
  - nodes create their ARP tables *without intervention from net administrator*

Addressing: routing to another LAN

walkthrough: 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?)



CSMA (carrier sense multiple access)

CSMA: listen before transmit:

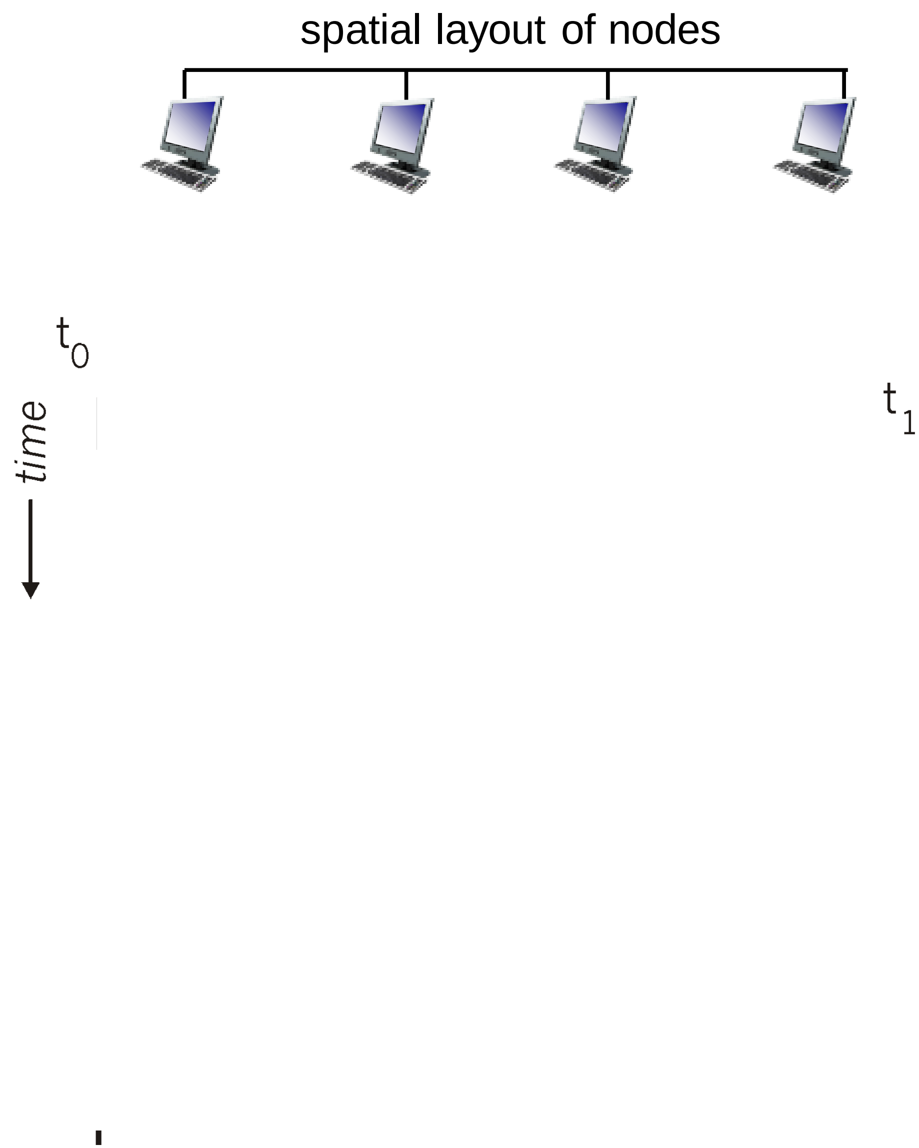
if channel sensed idle: transmit entire frame

- if channel sensed busy, defer transmission

- human analogy: don't interrupt others!

CSMA collisions

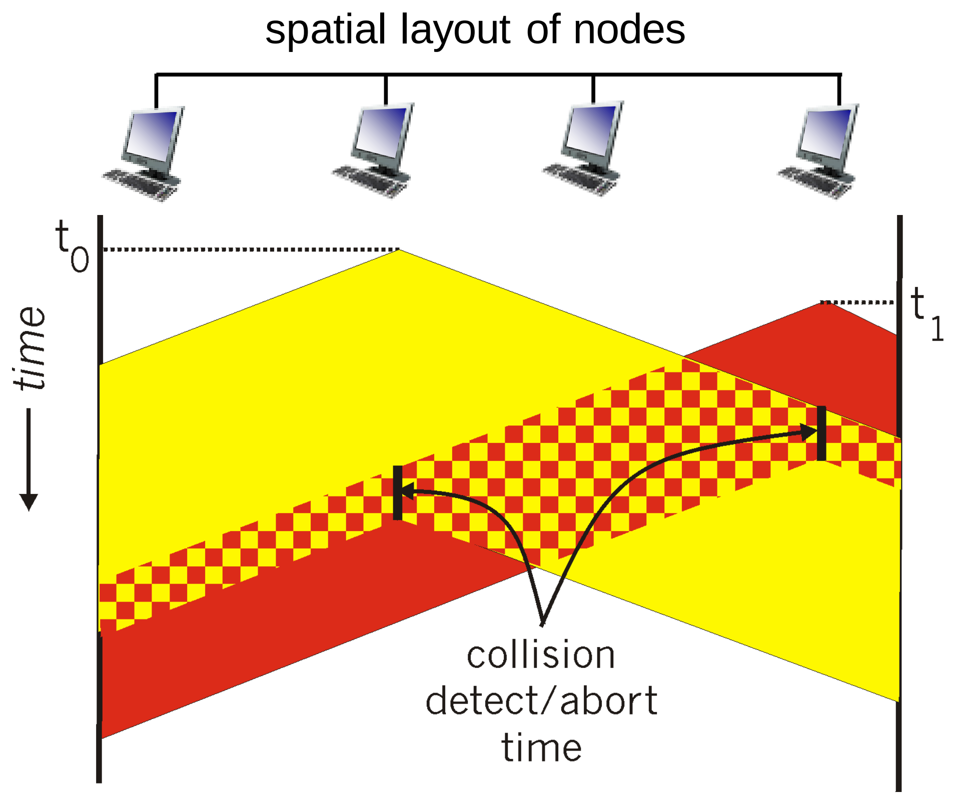
- collisions *can* still occur: propagation delay means two nodes may not hear each other's transmission
- collision: entire packet transmission time wasted
  - distance & propagation delay play role in in determining collision probability



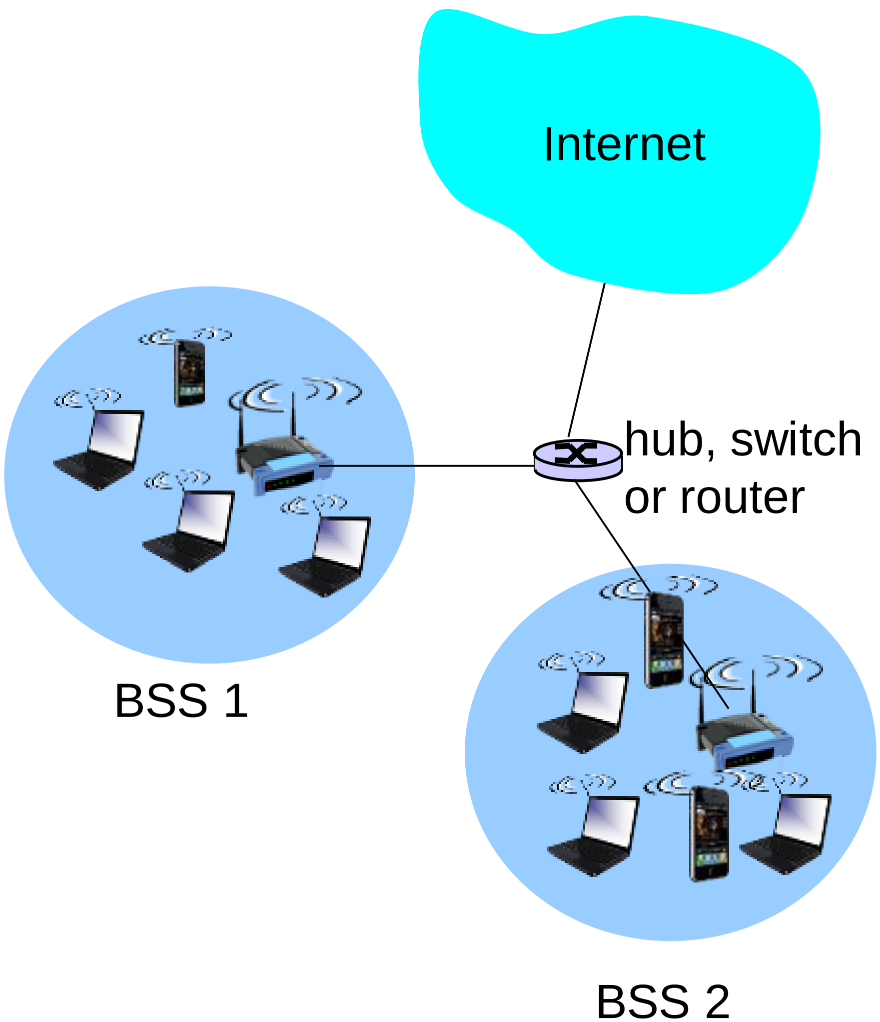
CSMA/CD (collision detection)

CSMA/CD: carrier sensing, deferral as in CSMA
  - collisions *detected* within short time
  - colliding transmissions aborted, reducing channel wastage
- collision detection:
  - easy in wired LANs: measure signal strengths, compare transmitted, received signals
  - difficult in wireless LANs: received signal strength overwhelmed by local transmission strength
- human analogy: the polite conversationalist

CSMA/CD (collision detection)

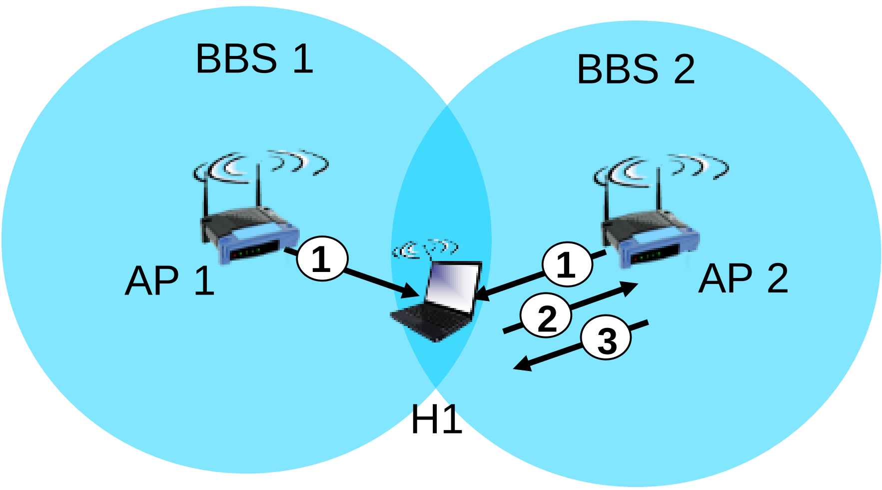


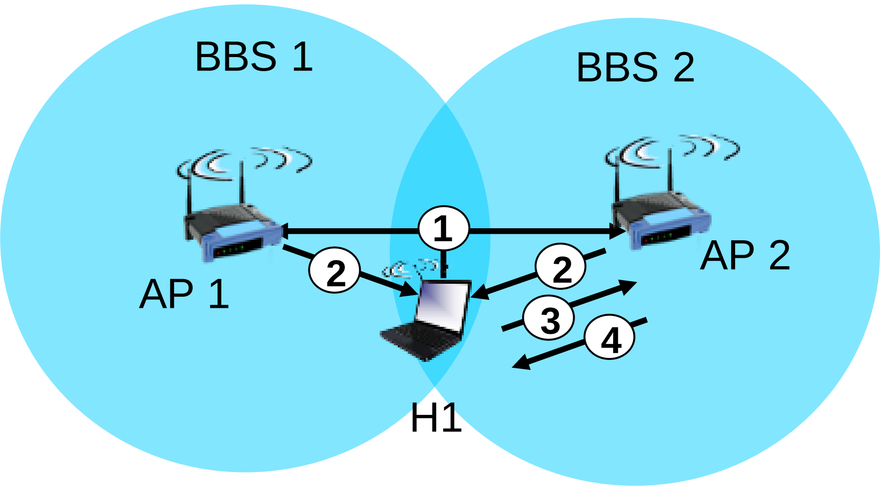
802.11 LAN architecture



- 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

802.11: passive/active scanning





passive scanning:

(1) beacon frames sent from APs

(2) association Request frame sent: H1 to selected AP

(3) association Response frame sent from selected AP to H1

active scanning:

(1) Probe Request frame broadcast from H1

(2) Probe Response frames sent from APs

(3) Association Request frame sent: H1 to selected AP

(4) Association Response frame sent from selected AP to H1

IEEE 802.11 MAC Protocol: CSMA/CA

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

timer counts down while channel idle

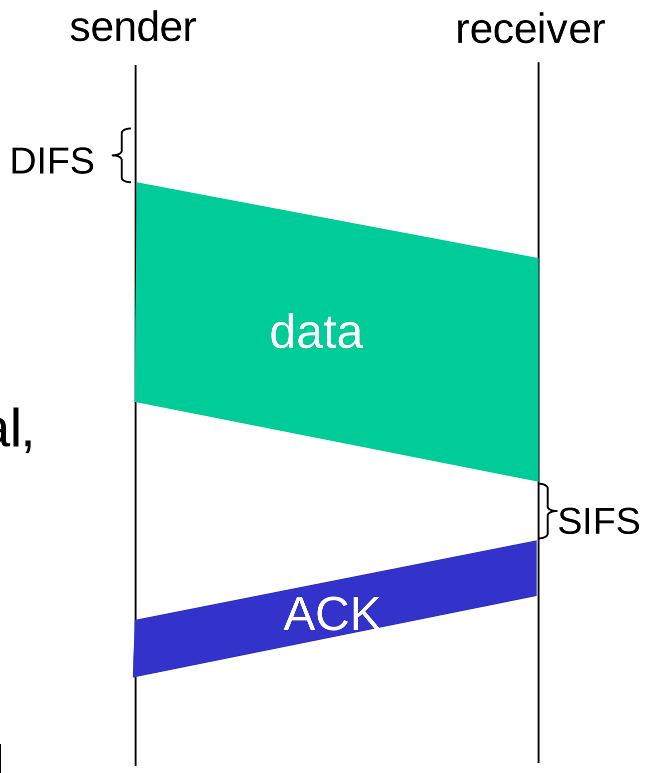
transmit when timer expires

if no ACK, increase random backoff interval, repeat 2

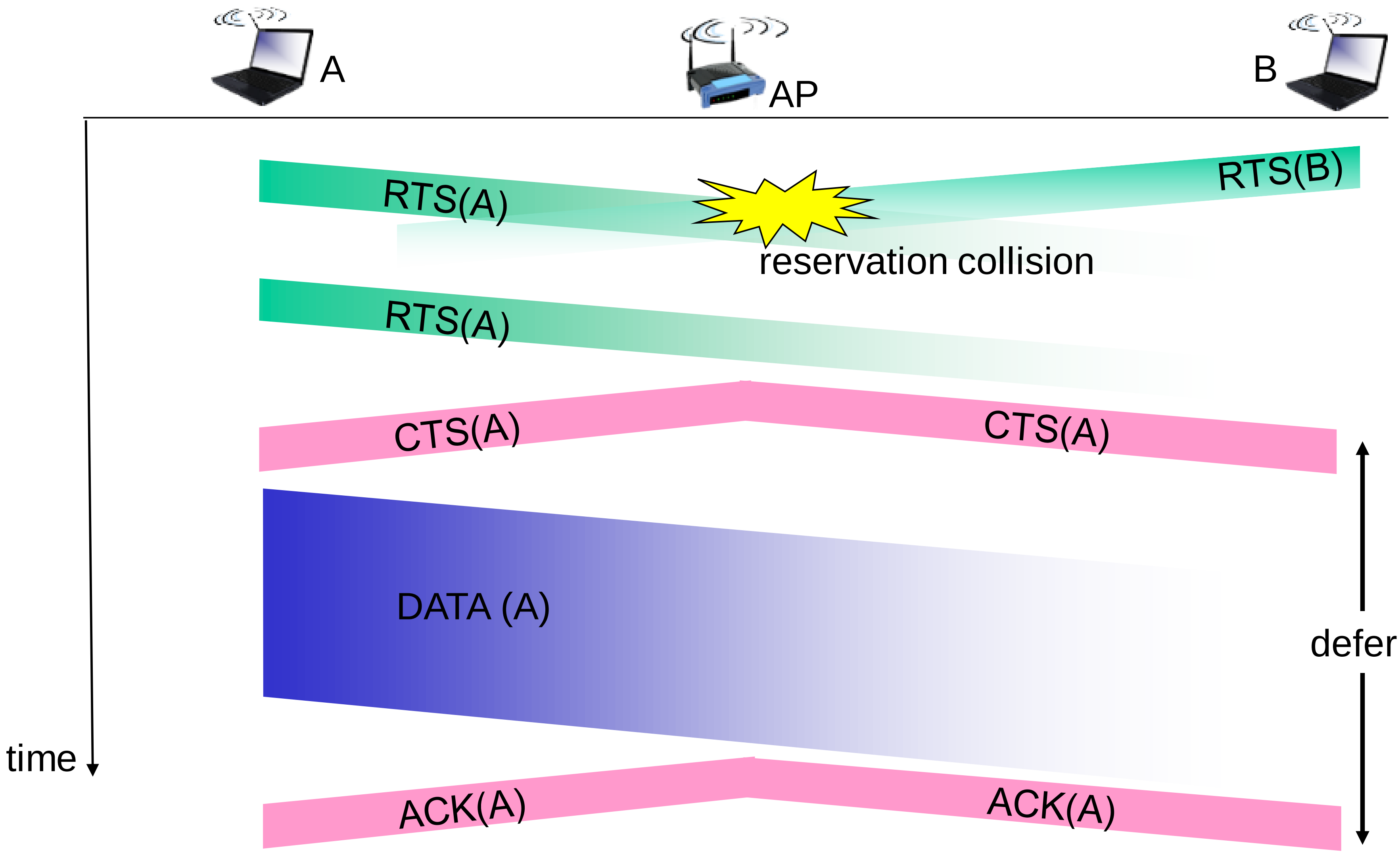
802.11 receiver

- if frame received OK

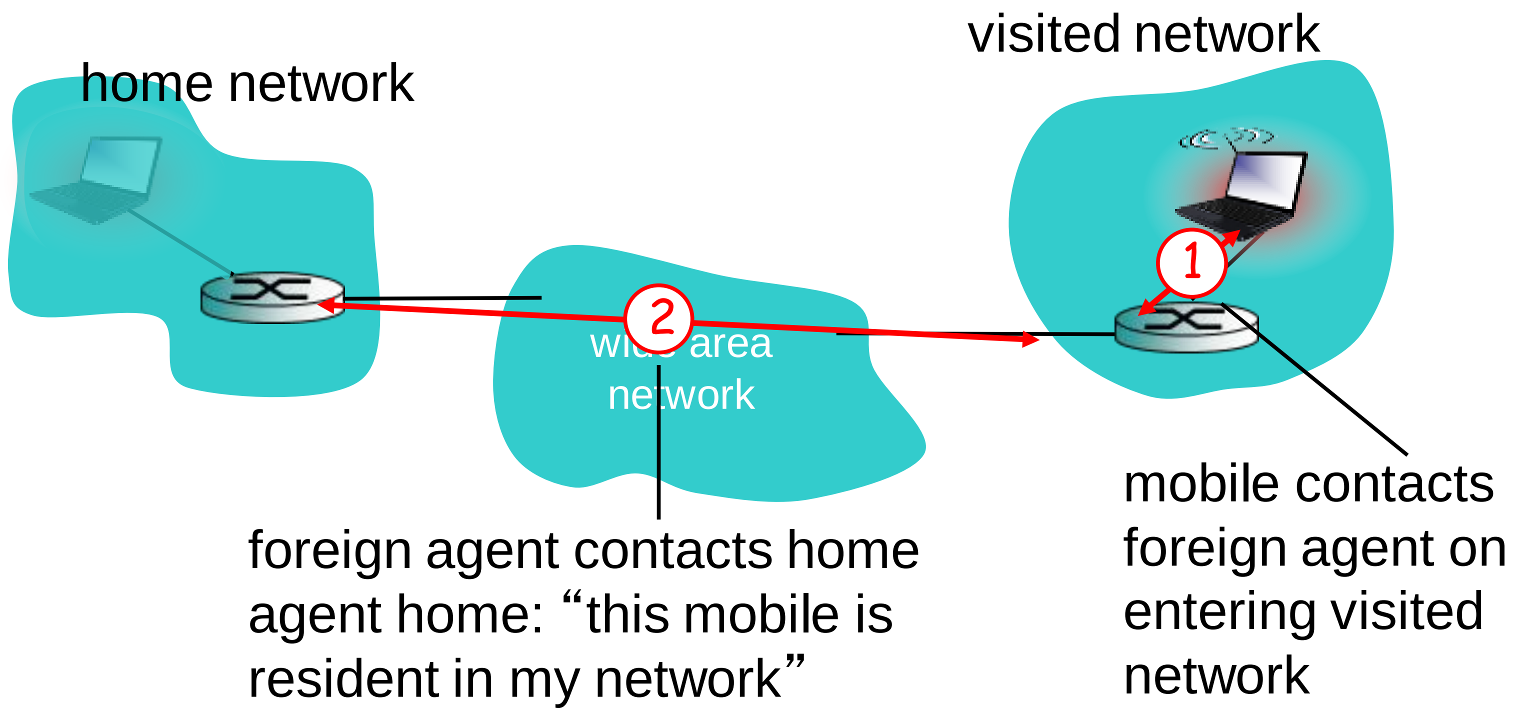
return ACK after Short Inter-frame Spacing (**SIFS)** (ACK needed due to hidden terminal problem)



Collision Avoidance: RTS-CTS exchange



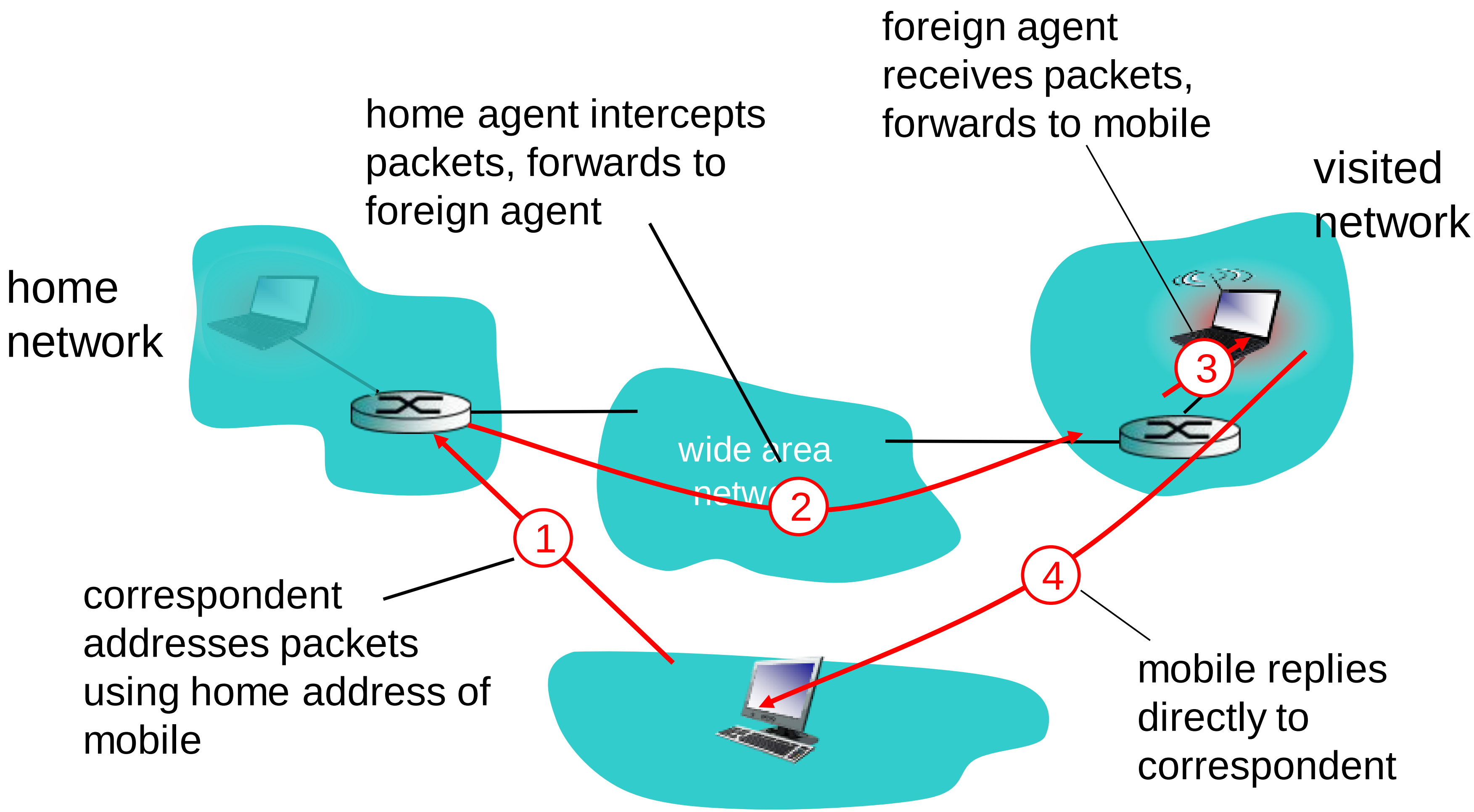
Mobility: registration



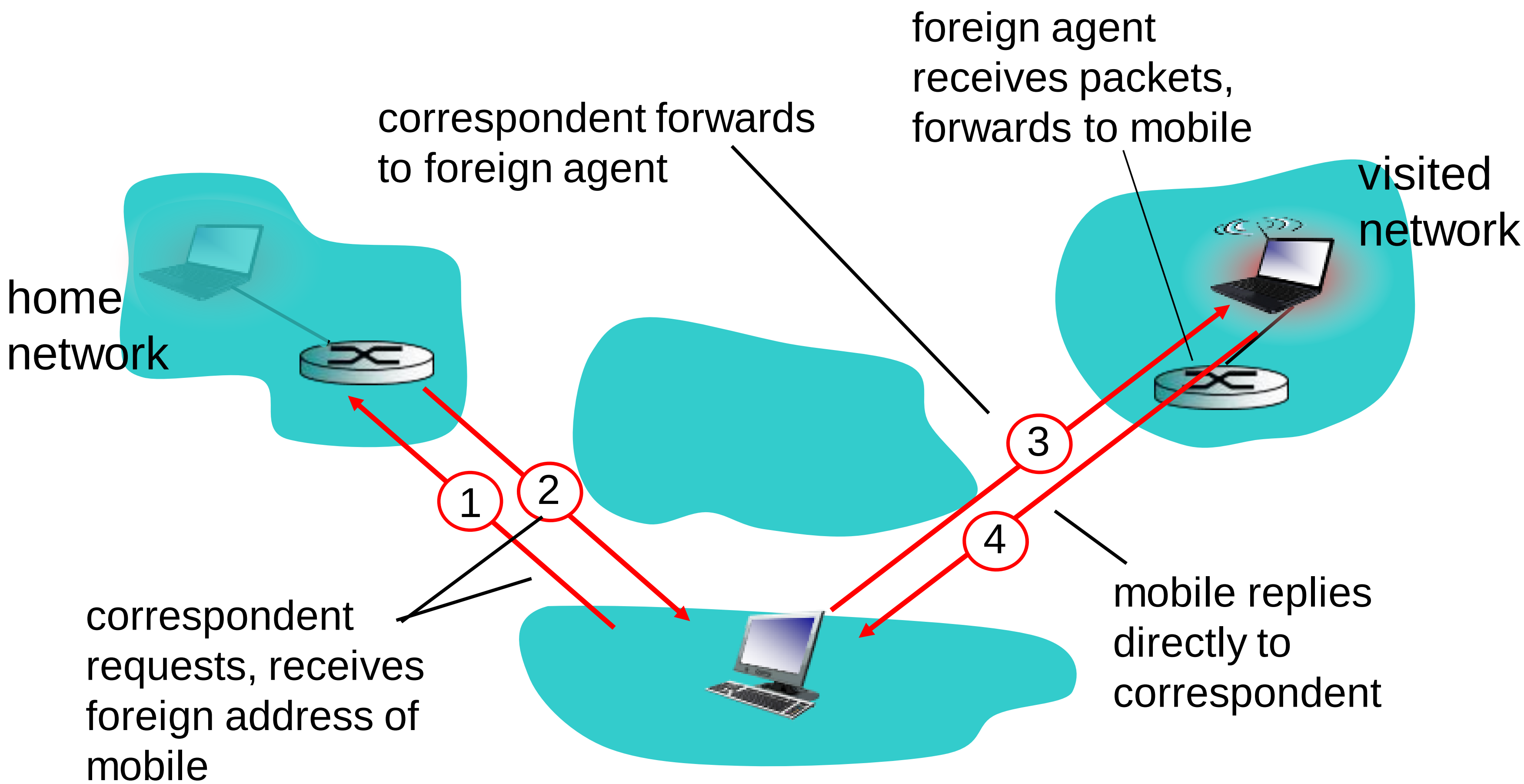
end result:

- foreign agent knows about mobile
- home agent knows location of mobile

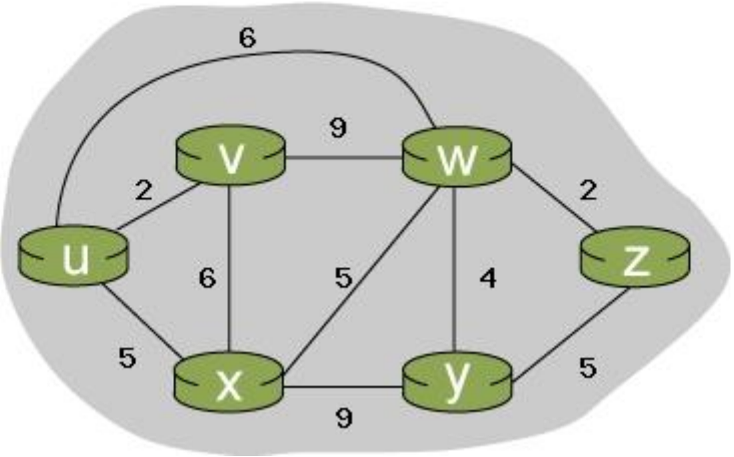
Mobility via indirect routing



Mobility via direct routing

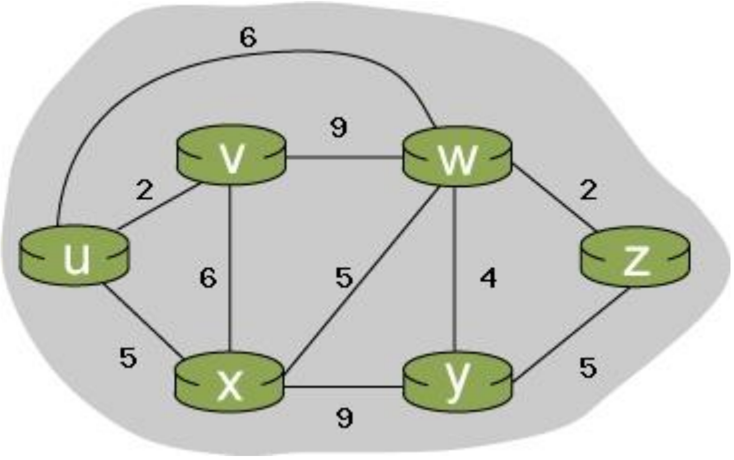


Dijkstra’s algorithm: another example



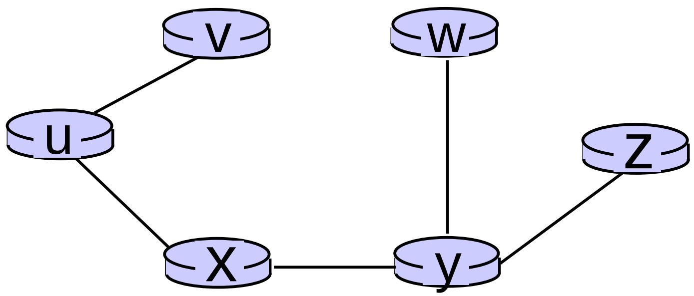
Dijkstra's algorithm: another example

Step	N'	D(v),p(v)	D(w),p(w)	D(x),p(x)	D(y),p(y)	D(z),p(z)
0						
1						
2						
3						
4						
5						



Dijkstra’s algorithm: example (2)

resulting shortest-path tree from u:



resulting forwarding table in u:

destination	link
v	
x	
y	
w	
z	

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:

wait for (change in local link cost or msg from neighbor)

recompute estimates

if DV to any dest has changed, *notify* neighbors

Distance Vector algorithm

Let

$d_x(y)$:= cost of least-cost path from x to y

then

$$d_x(y) = \min_v \{c(x,v) + d_v(y)\}$$

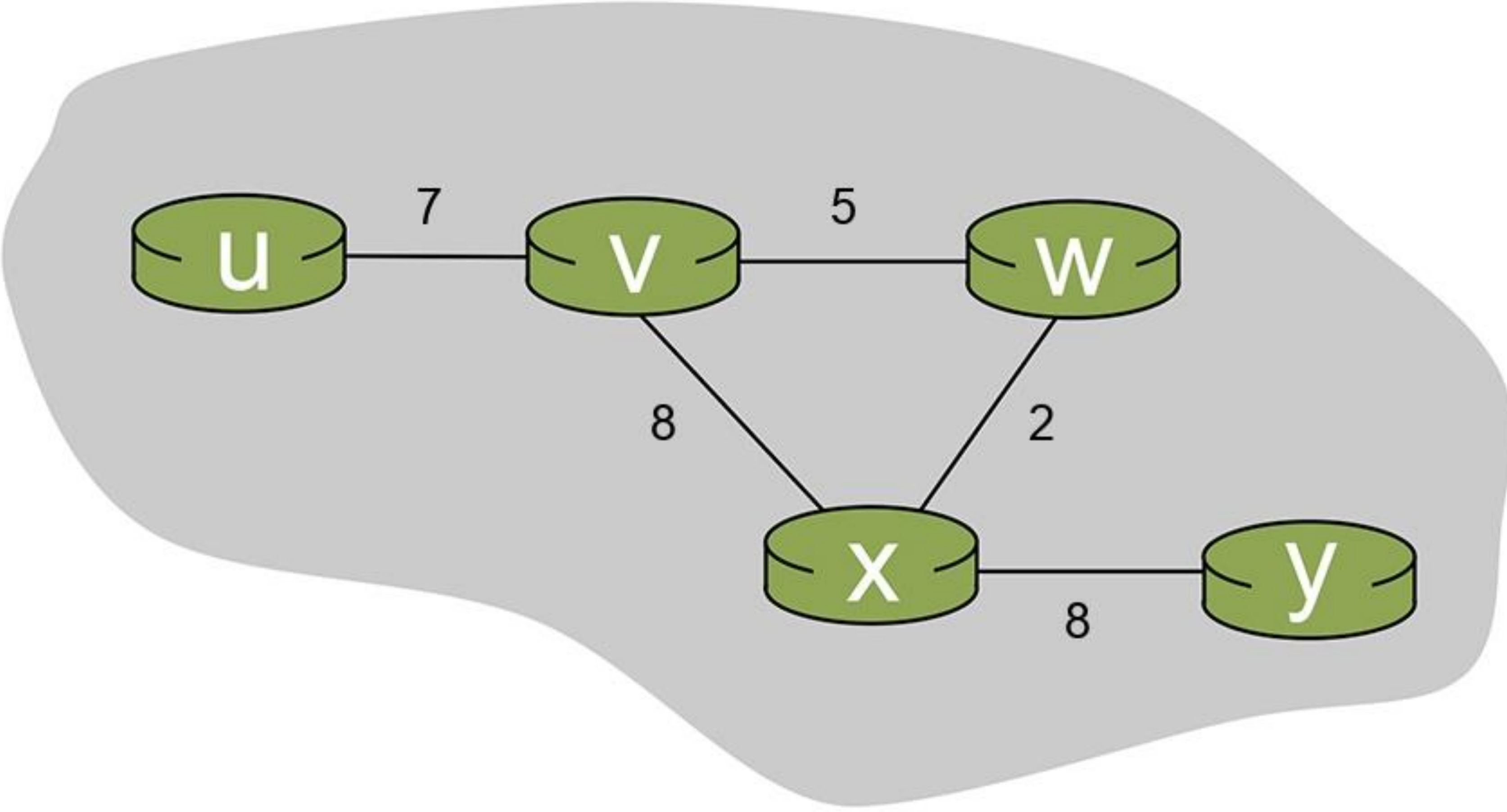
cost from neighbor v to destination y

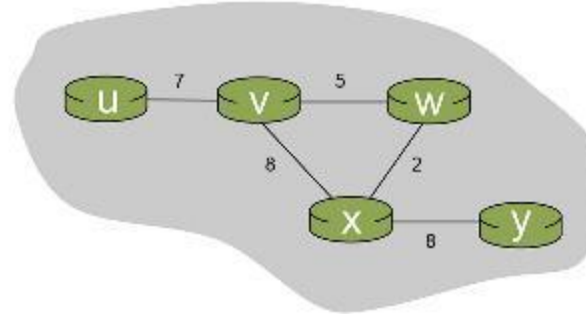
cost to neighbor v

min taken over all neighbors v of x

Also called the Bellman-Ford equation (dynamic programming)

DVR Example





U	U	V	W	X	Y
U					
V					

V	U	V	W	X	Y
U					
V					
W					
X					

W	U	V	W	X	Y
V					
W					
X					

X	U	V	W	X	Y
V					
W					
X					
Y					

Y	U	V	W	X	Y
X					
Y					

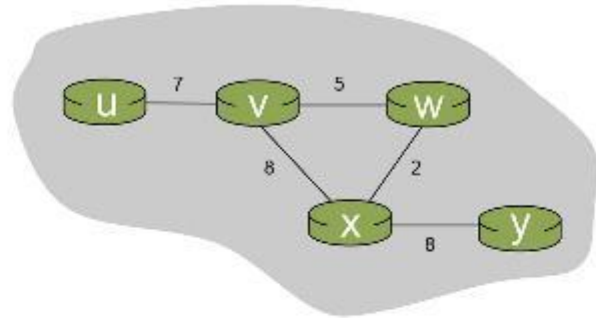
U	U	V	W	X	Y
U					
V					

V	U	V	W	X	Y
U					
V					
W					
X					

W	U	V	W	X	Y
V					
W					
X					

X	U	V	W	X	Y
V					
W					
X					
Y					

Y	U	V	W	X	Y
X					
Y					



U	U	V	W	X	Y
U					
V					

V	U	V	W	X	Y
U					
V					
W					
X					

W	U	V	W	X	Y
V					
W					
X					

X	U	V	W	X	Y
V					
W					
X					
Y					

Y	U	V	W	X	Y
X					
Y					

U	U	V	W	X	Y
U					
V					

V	U	V	W	X	Y
U					
V					
W					
X					

W	U	V	W	X	Y
V					
W					
X					

X	U	V	W	X	Y
V					
W					
X					
Y					

Y	U	V	W	X	Y
X					
Y					

Comparison of LS and DV algorithms

Communication costs

- ***LS:*** broadcast link info to all
- ***DV:*** exchange between neighbors only
  - convergence time varies

robustness: what happens if router malfunctions?

LS:

- node can advertise incorrect *link* cost
- each node computes only its *own* table

DV:

- DV node can advertise incorrect *path* cost
- each node's table used by others
- error propagate thru network