

Final review

CS 455
Parth Pathak

Final exam

- Tuesday (5/16) from 10:30 am to 1:15 pm
 - In class, in person
- Closed book, closed notes, no electronic devices
 - No cheatsheet
 - Bring your own calculator, can't share it
- 15% of your grade
- Comprehensive
 - Questions will be on topics discussed after midterm
 - But you are expected to know the topics before midterm
- Chapters 4, 5, 6, 7, 8 (until slide 53 in 8-security.ppt deck)
- Multiple choice just like the midterm

Network-layer functions

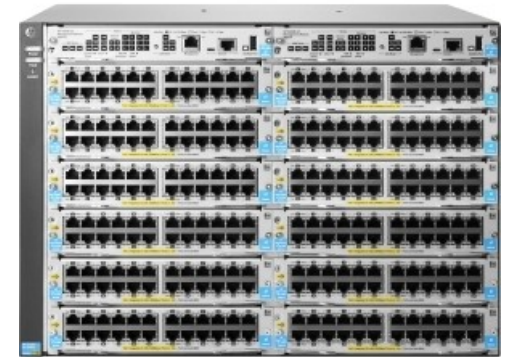
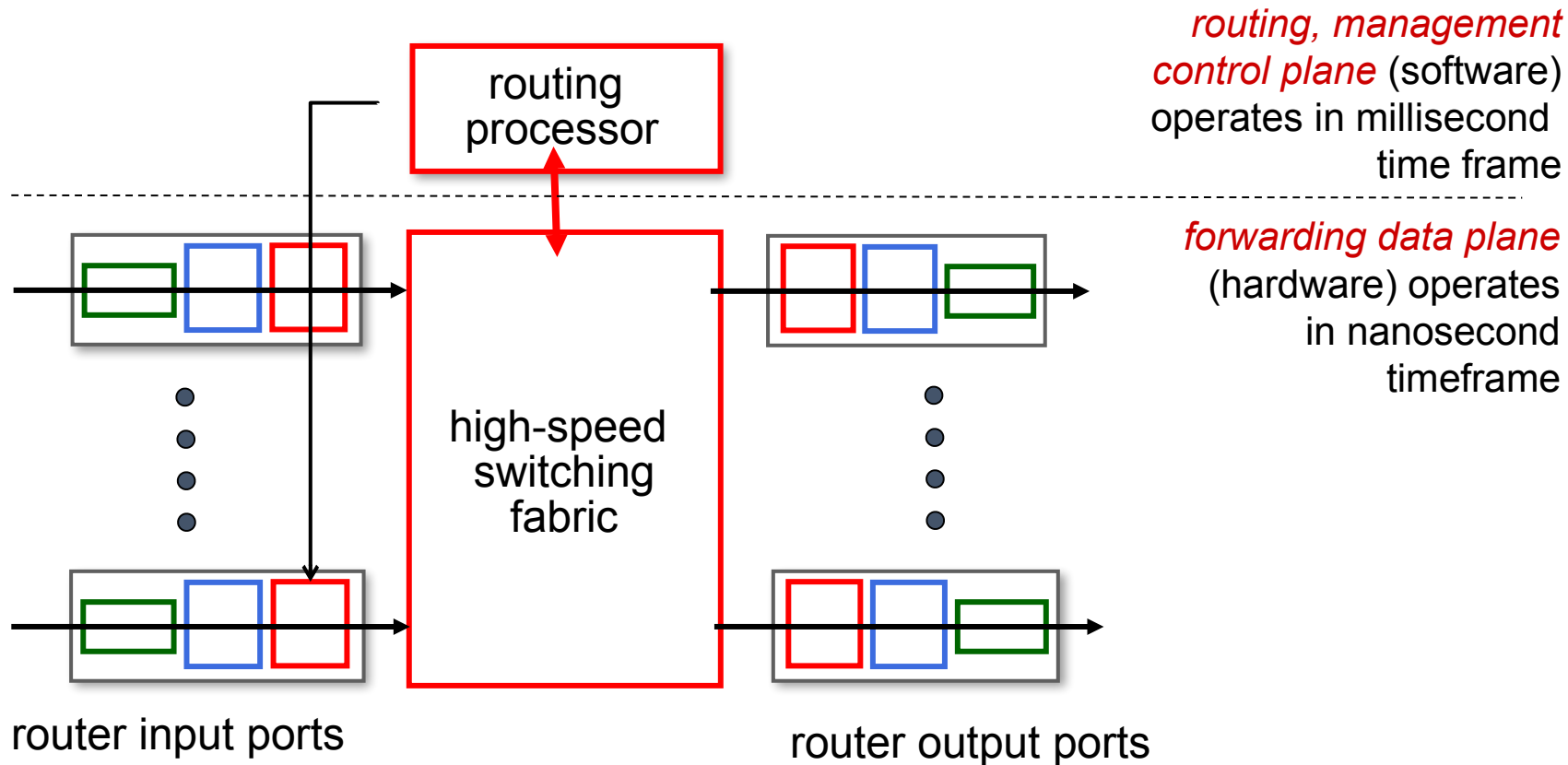
- **forwarding:** move packets from router's input to appropriate router output *data plane*
- **routing:** determine route taken by packets from source to destination *control plane*

Two approaches to structuring network control plane:

- per-router control (traditional)
- logically centralized control (software defined networking)

Router architecture overview

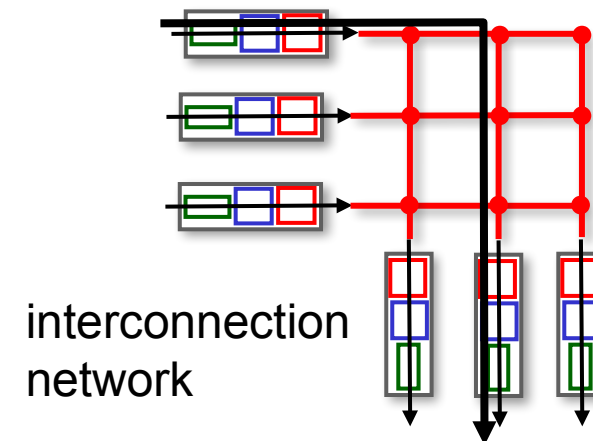
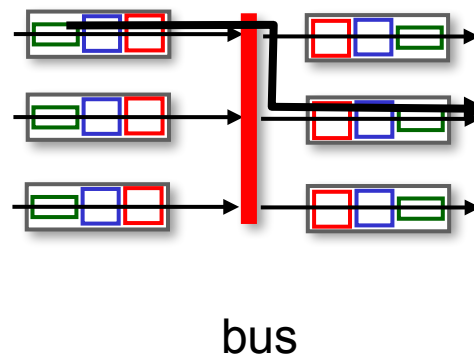
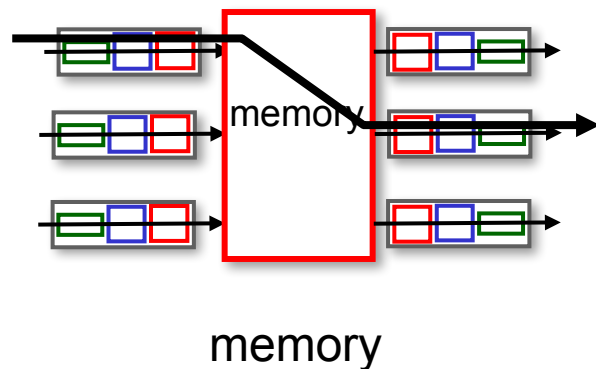
high-level view of generic router architecture:



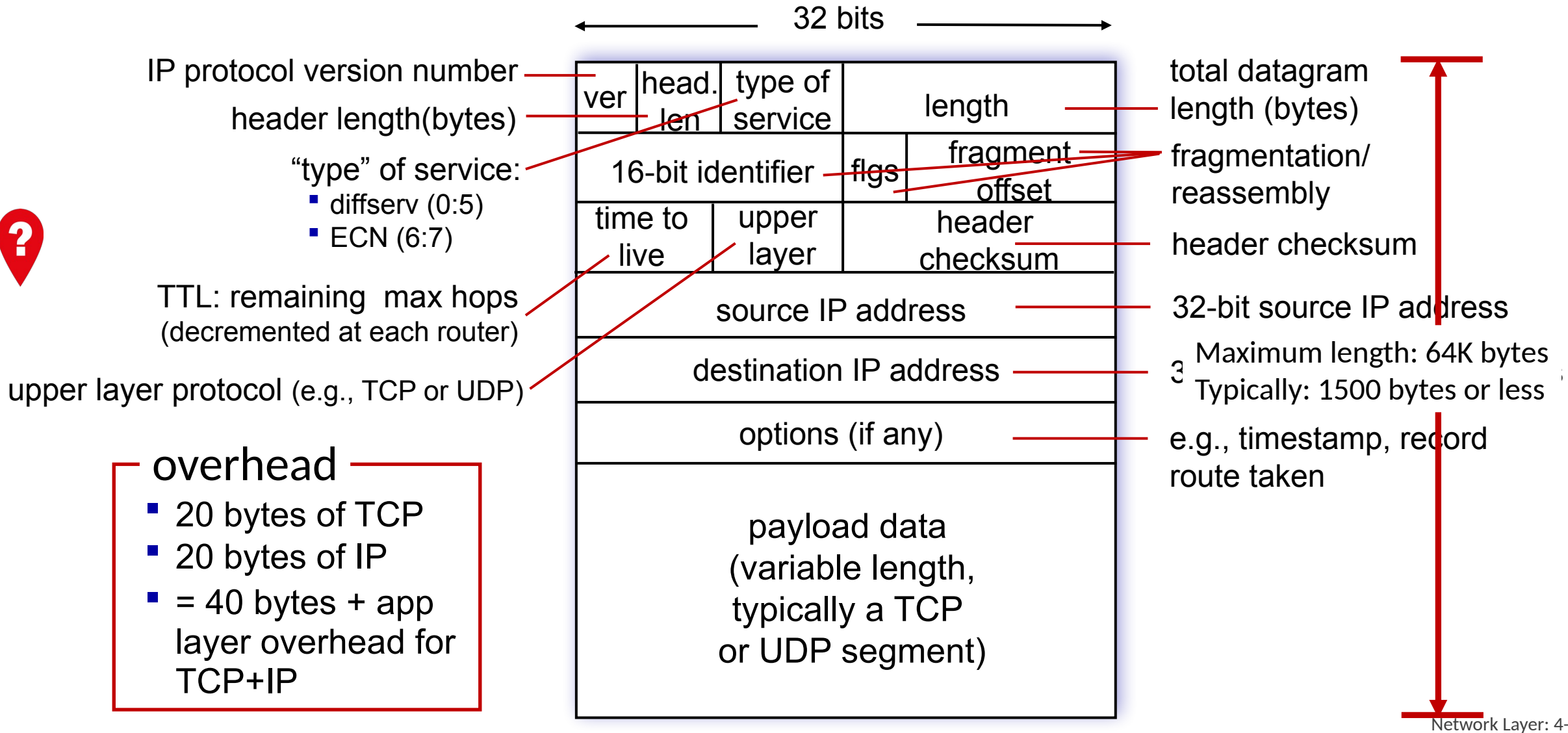
**HPE J9822A
5412R**

Switching fabrics

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



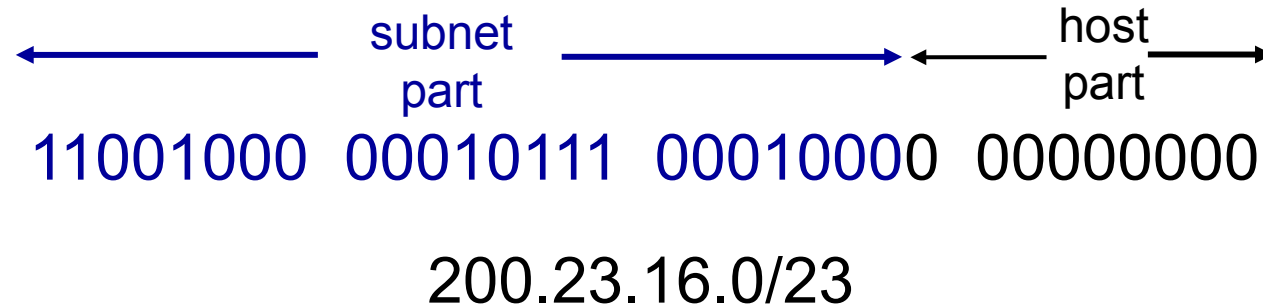
IP Datagram format



IP addressing: CIDR

CIDR: Classless **I**nter**D**omain **R**outing (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



DHCP client-server scenario

DHCP server: 223.1.2.5



DHCP discover

Broadcast: is there a
DHCP server out
there?

Arriving client



DHCP offer

Broadcast: I'm a DHCP
server! Here's an IP
address you can use

DHCP request

Broadcast: OK. I would
like to use this IP
address!

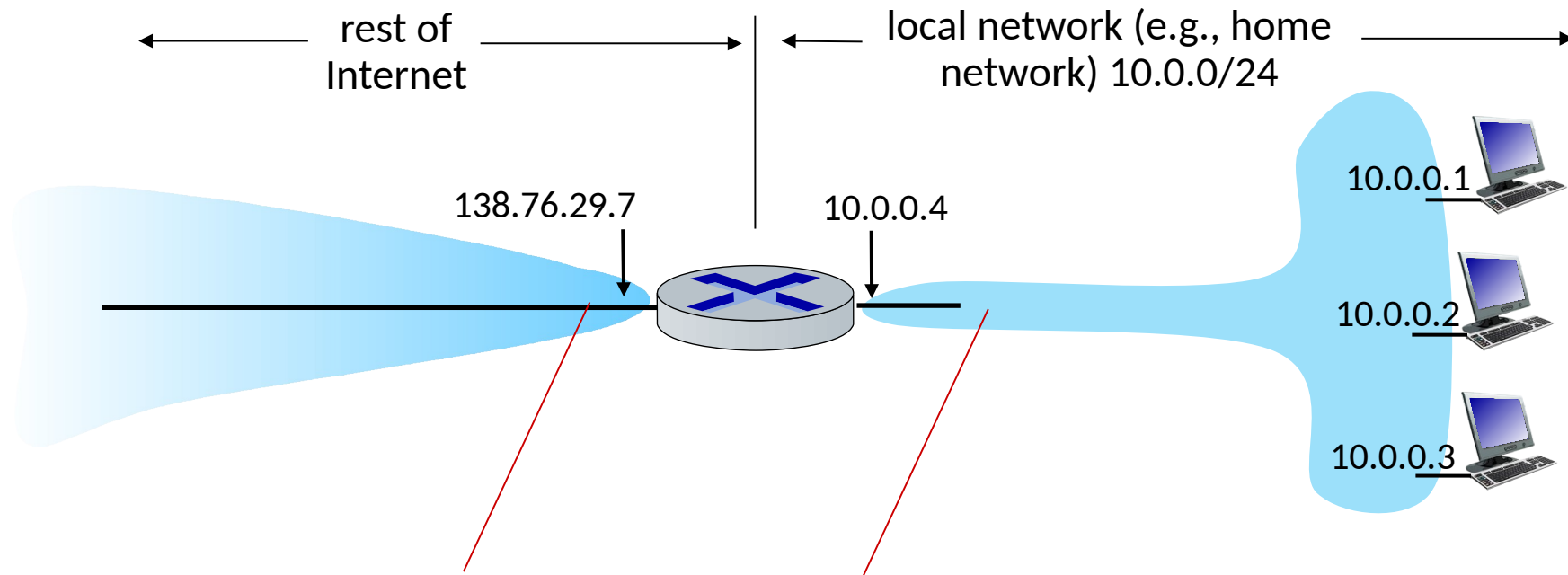
DHCP ACK

Broadcast: OK. You've
got that IP address!

The two steps above can
be skipped "if a client
remembers and wishes to
reuse a previously
allocated network address"
[RFC 2131]

NAT: network address translation

NAT: all devices in local network share just **one** IPv4 address as far as outside world is concerned

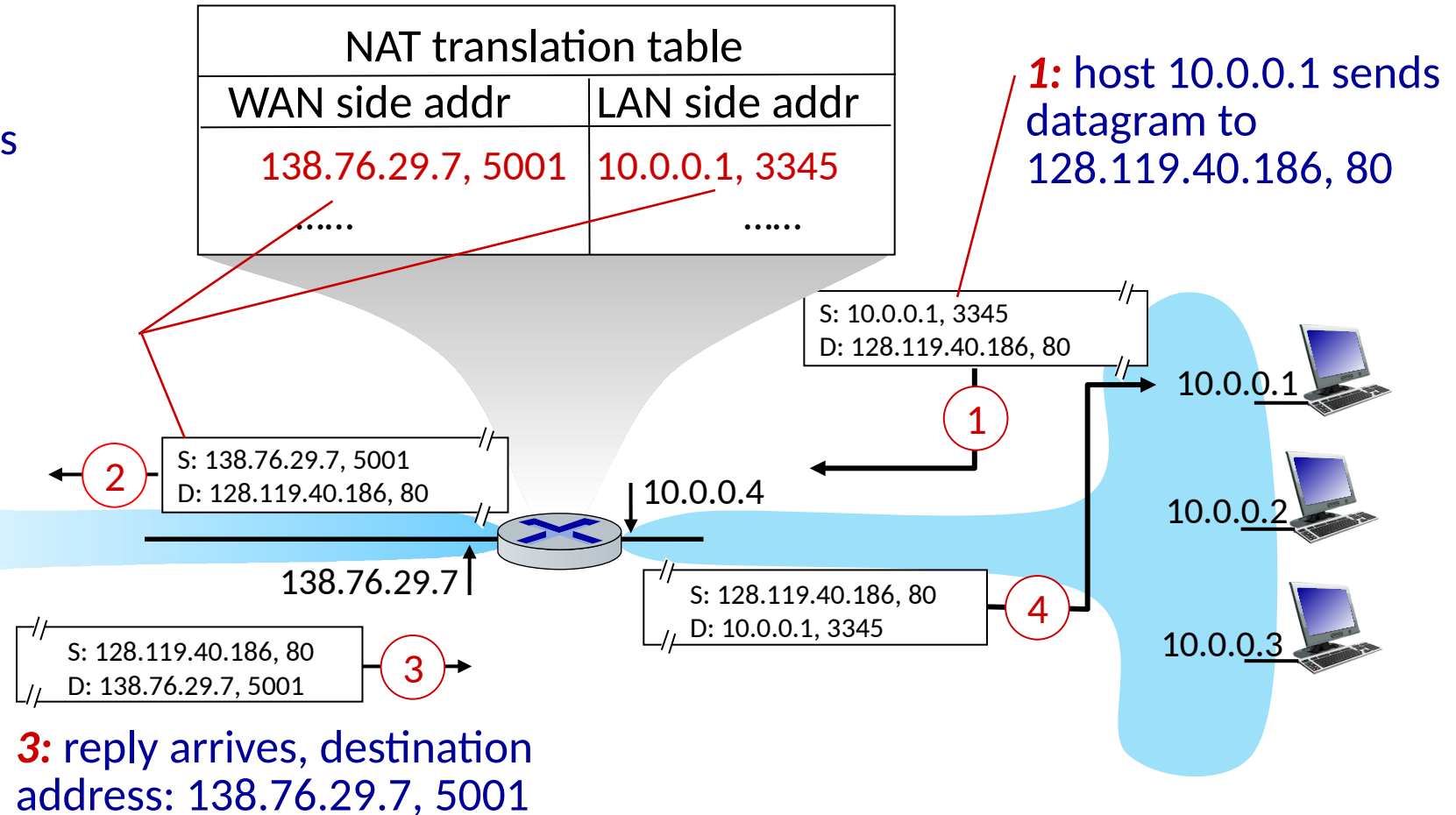


all datagrams *leaving* local network have *same* source NAT IP address: 138.76.29.7, but *different* source port numbers

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

NAT: network address translation

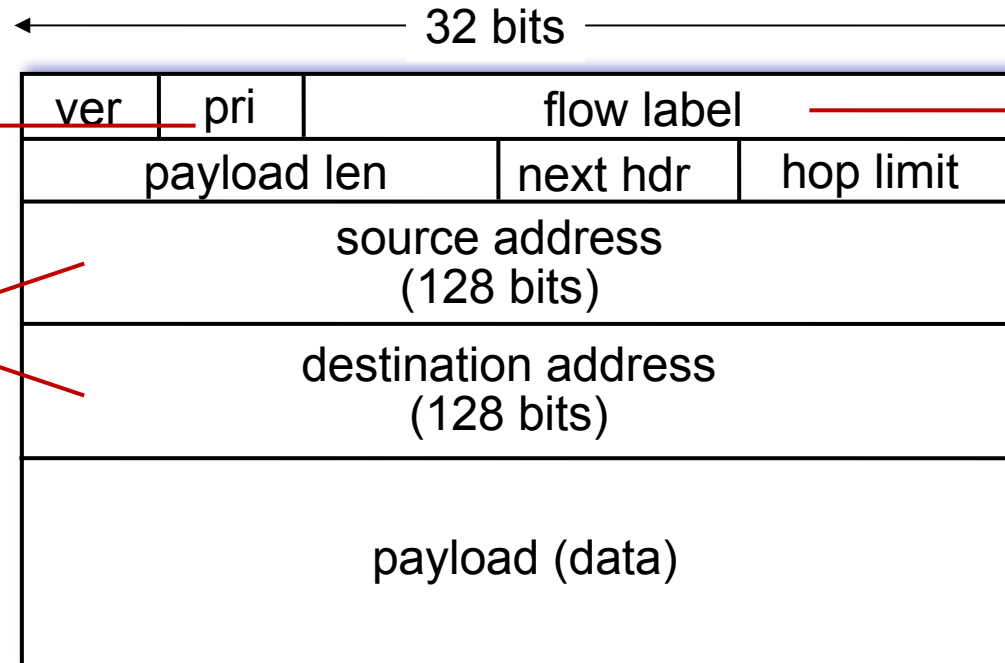
2: NAT router changes datagram source address from 10.0.0.1, 3345 to 138.76.29.7, 5001, updates table



IPv6 datagram format

priority: identify
priority among
datagrams in flow

128-bit
IPv6 addresses

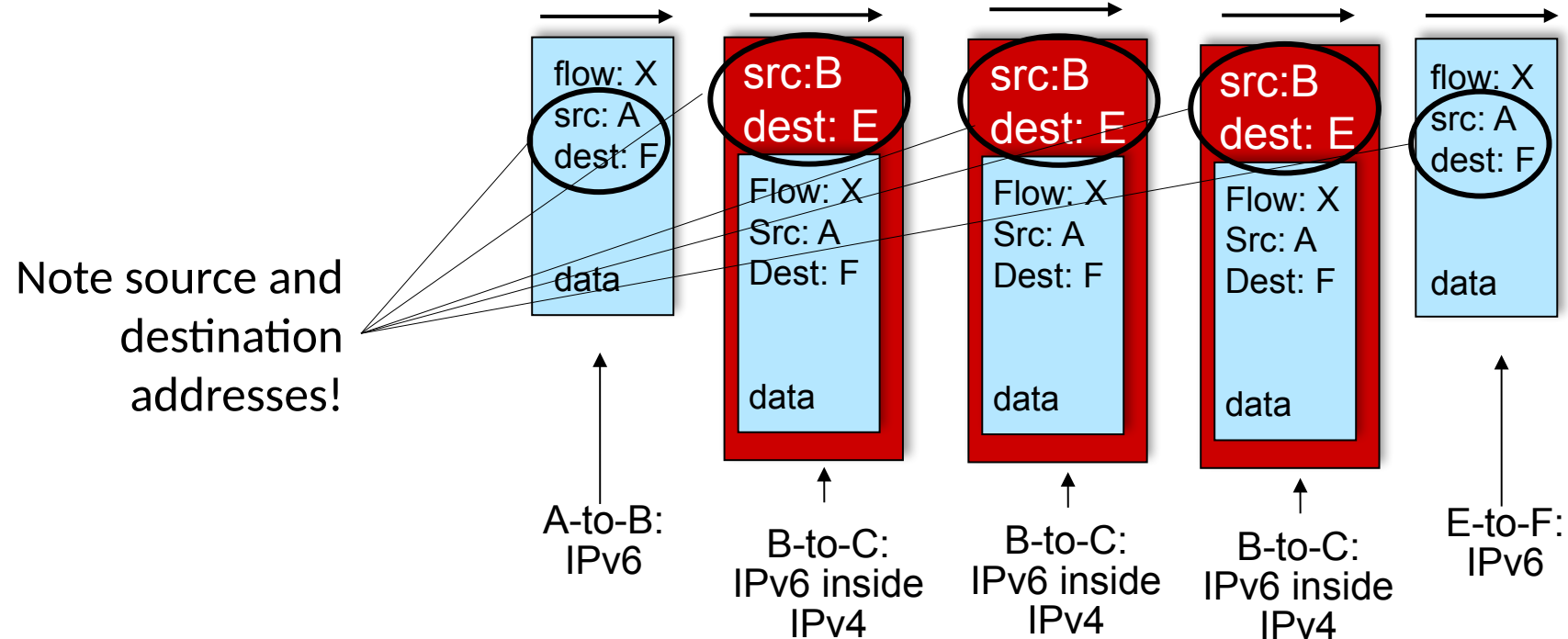
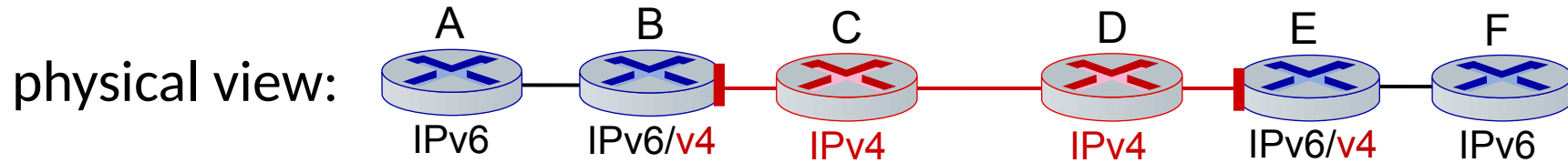


flow label: identify
datagrams in same
"flow." (concept of
"flow" not well defined).

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)

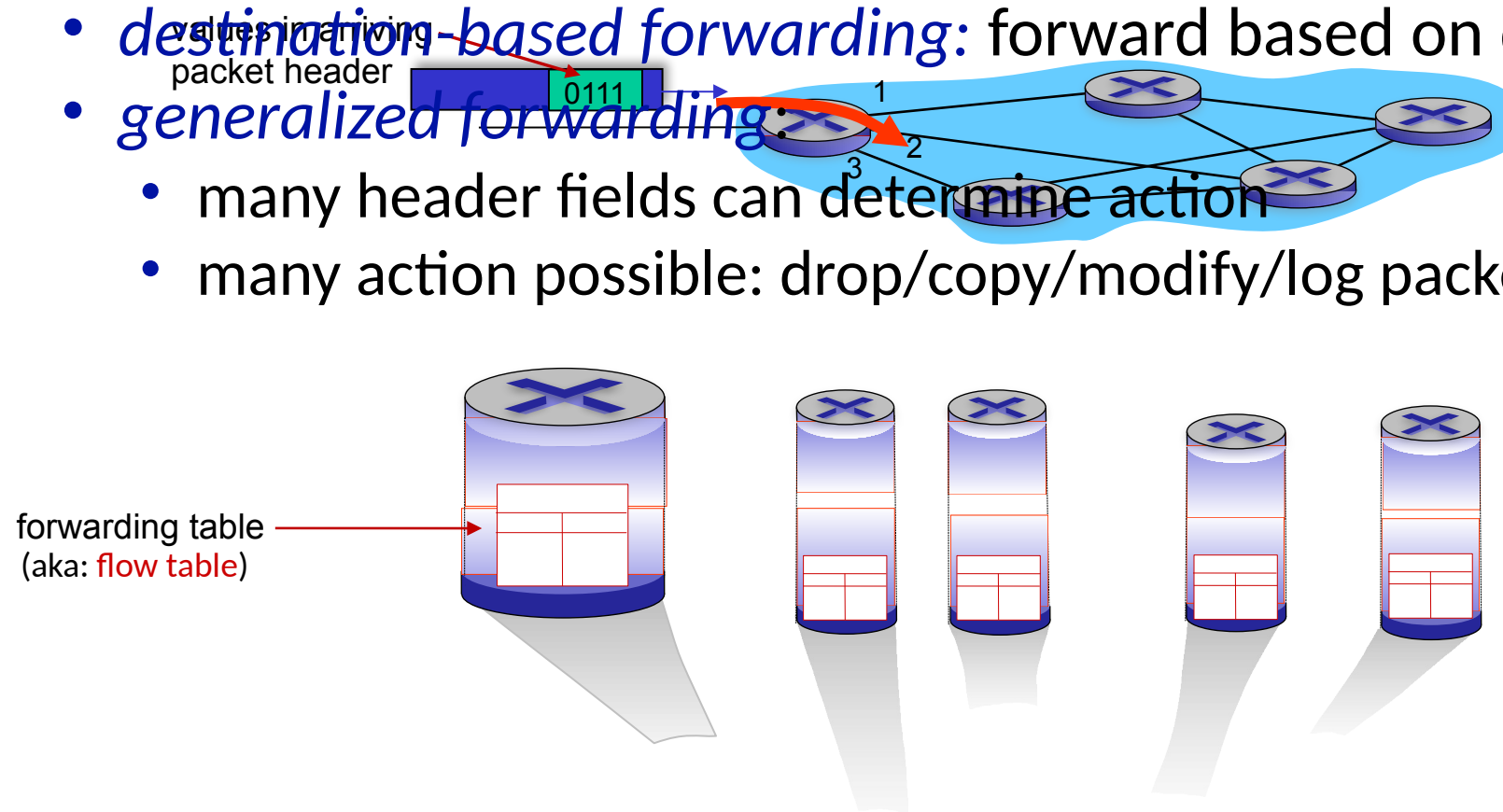
Tunneling



Generalized forwarding: match plus action

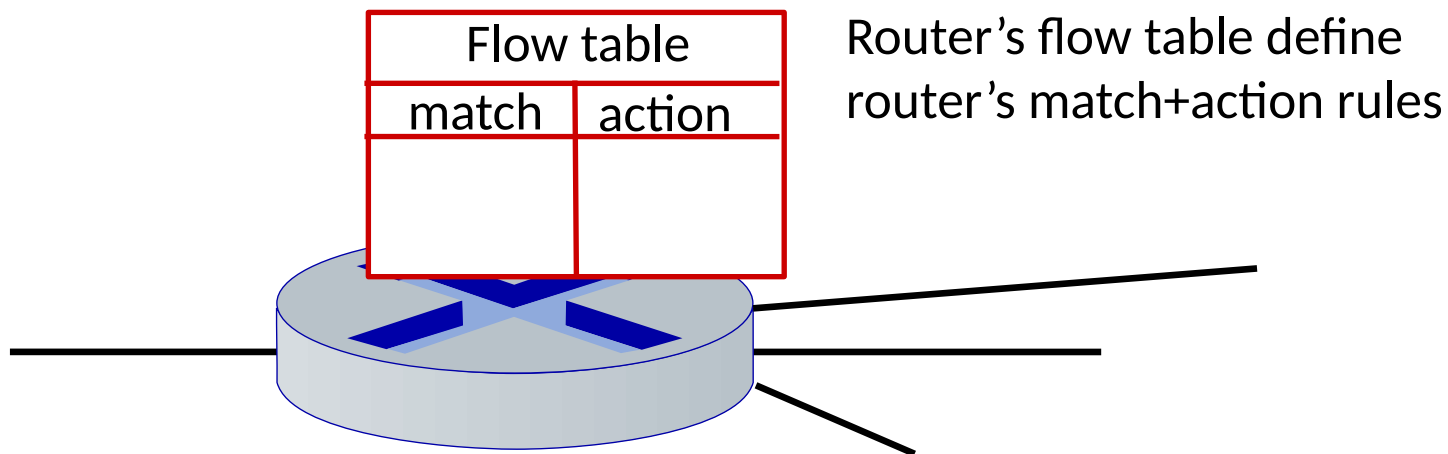
Review: each router contains a **forwarding table** (aka: **flow table**)

- “**match plus action**” abstraction: match bits in arriving packet, take action
- **destination-based forwarding**: forward based on dest. IP address
- **generalized forwarding**:
 - many header fields can determine action
 - many action possible: drop/copy/modify/log packet



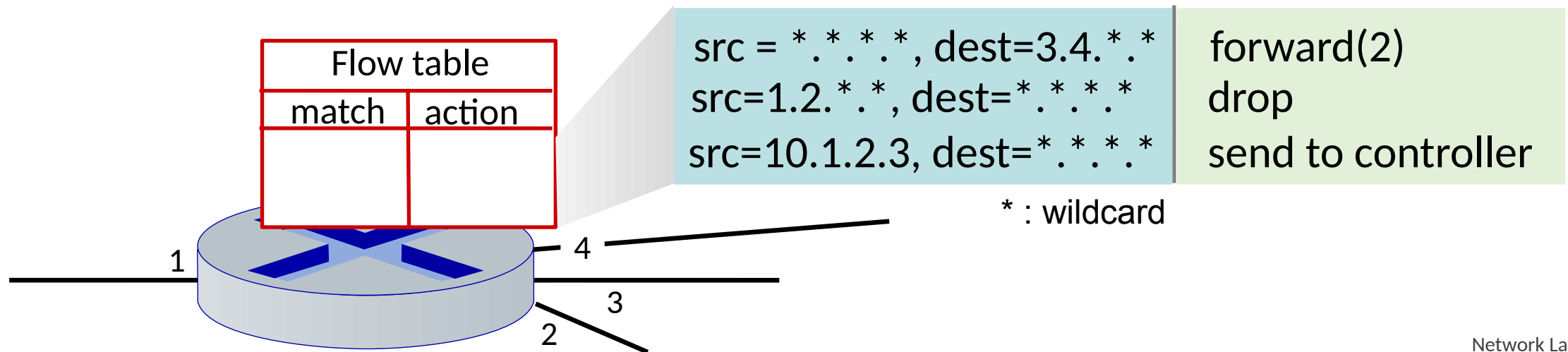
Flow table abstraction

- **flow**: defined by header field values (in link-, network-, transport-layer 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



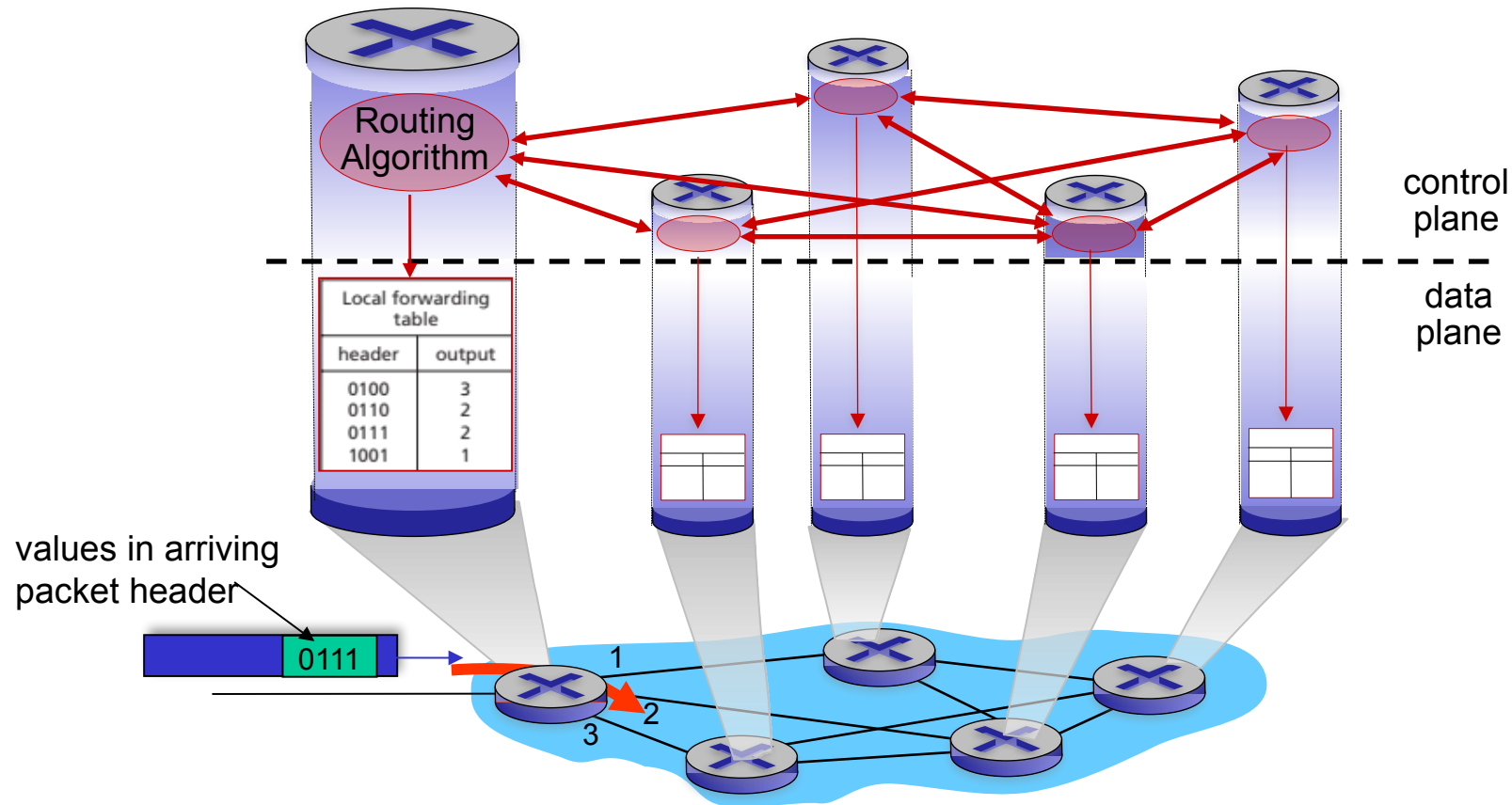
Flow table abstraction

- **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



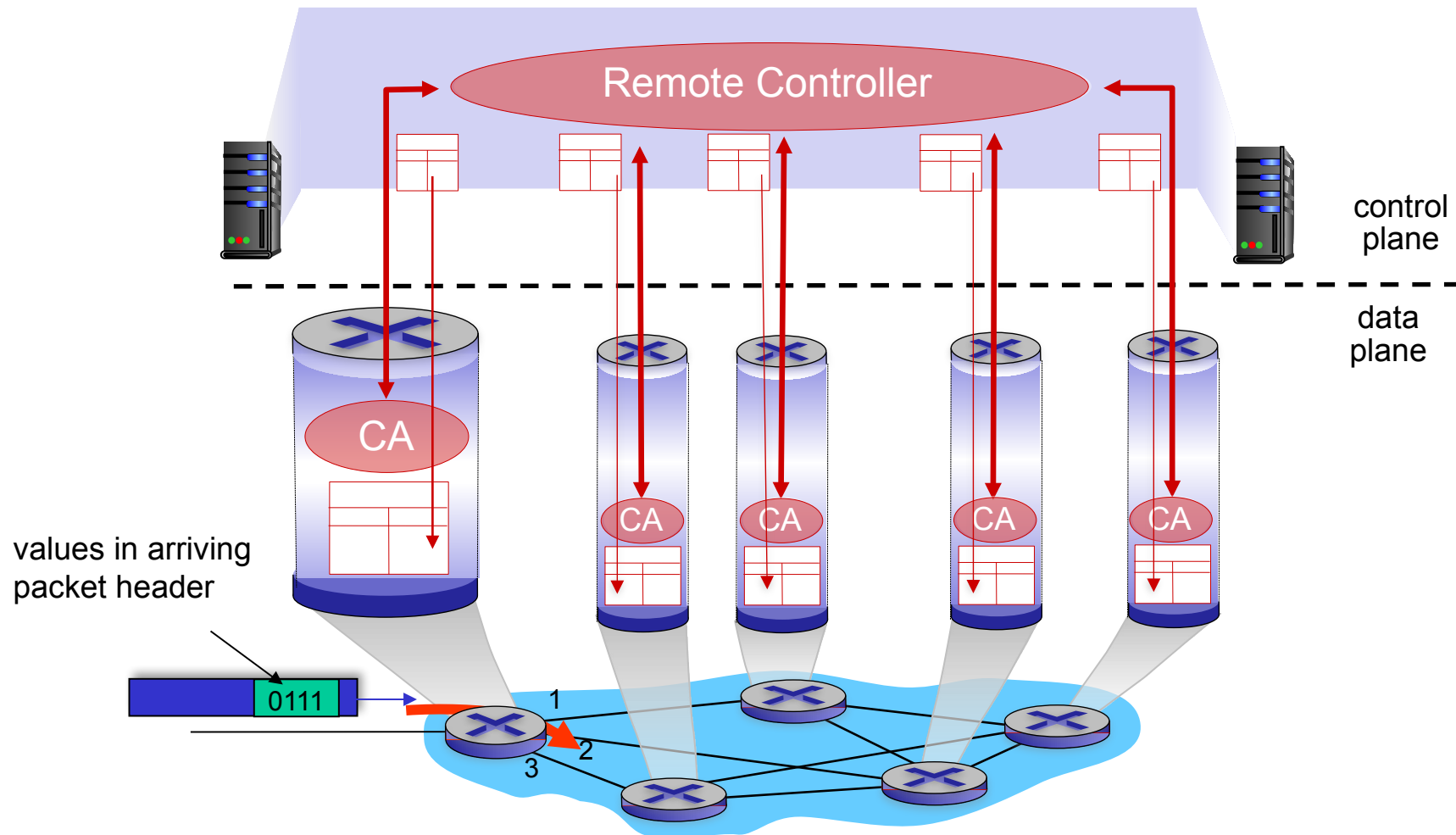
Per-router control plane

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

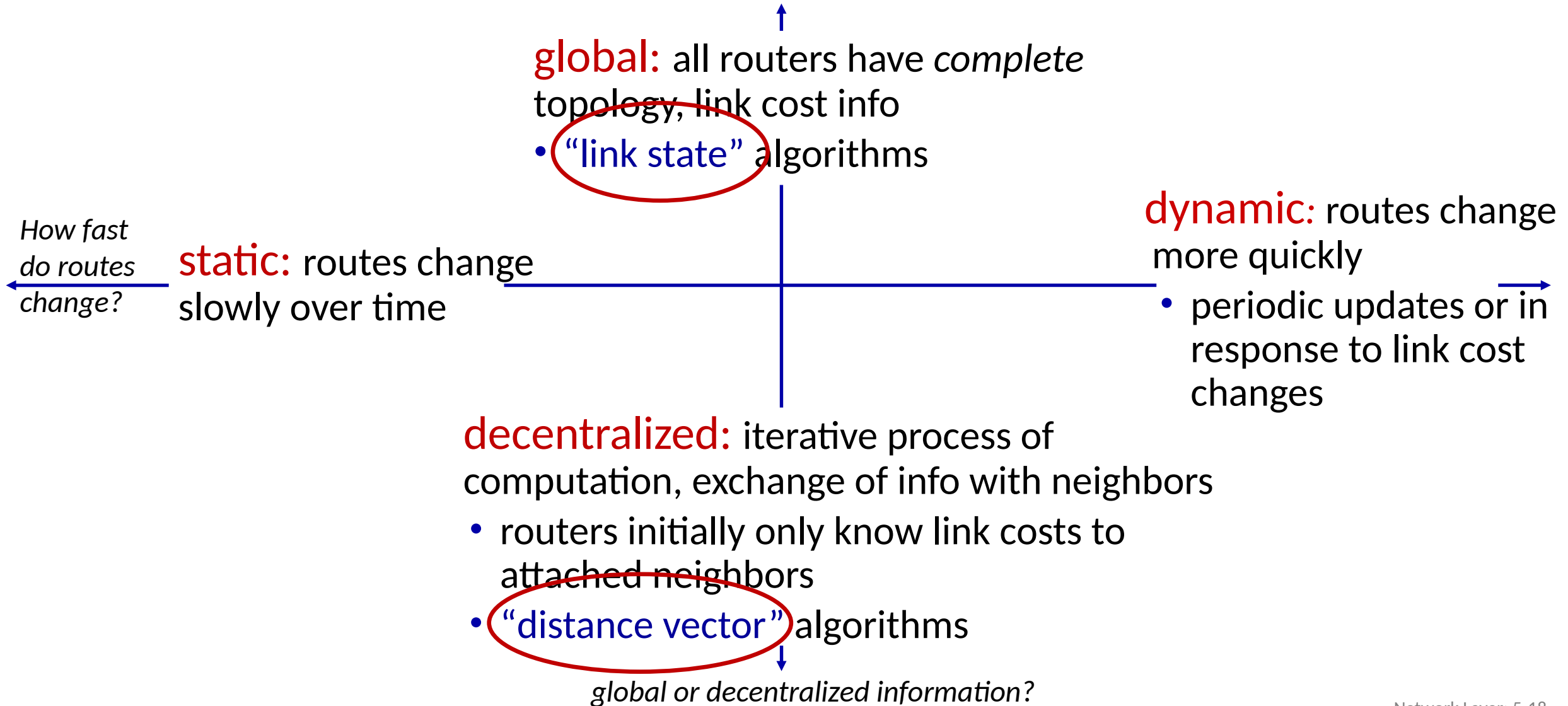


Software-Defined Networking (SDN) control plane

Remote controller computes, installs forwarding tables in routers

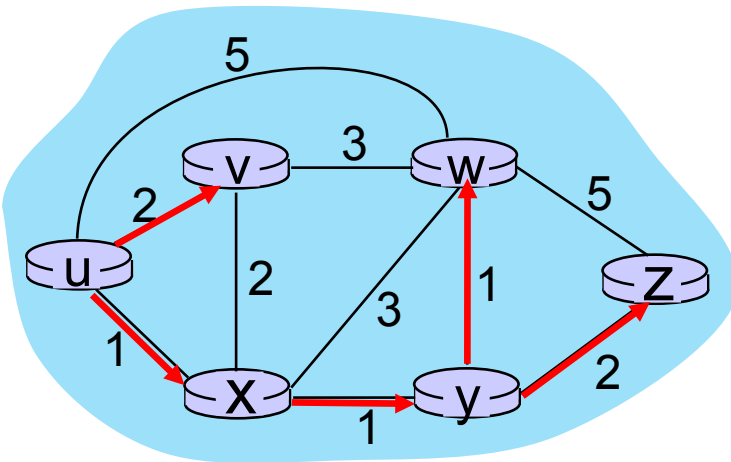


Routing algorithm classification



Dijkstra's algorithm: an example

Step	N'	$D(v), p(v)$	$D(w), p(w)$	$D(x), p(x)$	$D(y), p(y)$	$D(z), p(z)$
0	u	2, u	5, u	1, u	∞	∞
1	ux	2, u	4, x		2, x	∞
2	uxy	2, u	3, y			4, y
3	uxyv		3, y			4, y
4	uxyvw					4, y
5	uxyvwz					



Initialization (step 0): For all a : if a adjacent to then $D(a) = c_{u,a}$

find a not in N' such that $D(a)$ is a minimum

add a to N'

update $D(b)$ for all b adjacent to a and not in N' :

$$D(b) = \min (D(b), D(a) + c_{a,b})$$

Distance vector algorithm

Based on *Bellman-Ford* (BF) equation (dynamic programming):

Bellman-Ford equation

Let $D_x(y)$: cost of least-cost path from x to y .

Then:

$$D_x(y) = \min_v \{ c_{x,v} + D_v(y) \}$$

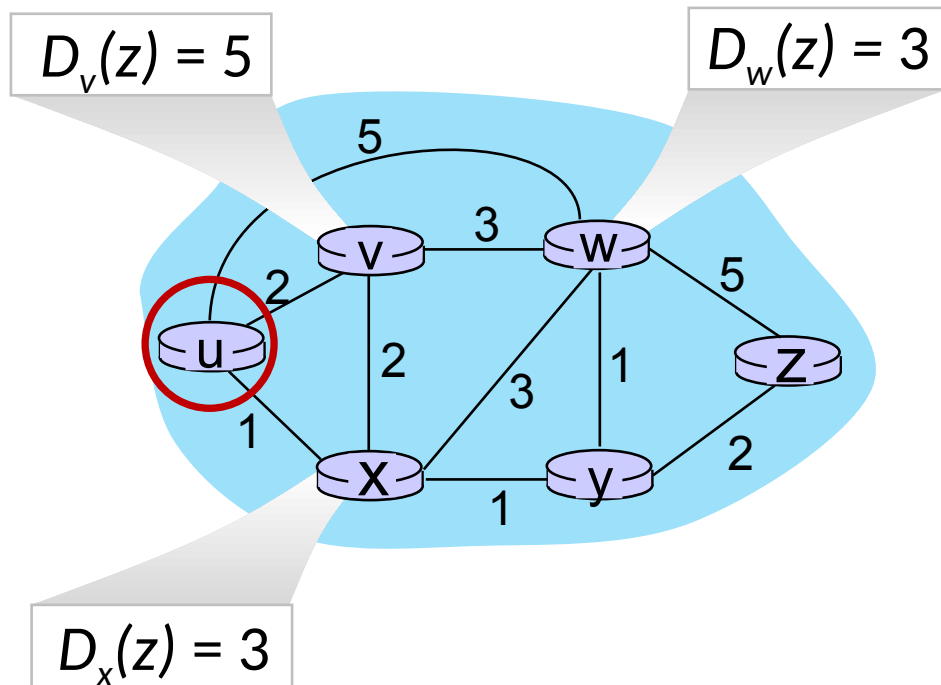
v 's estimated least-cost-path cost to y

\min taken over all neighbors v of x

direct cost of link from x to v

Bellman-Ford Example

Suppose that u 's neighboring nodes, x, v, w , know that for destination z :



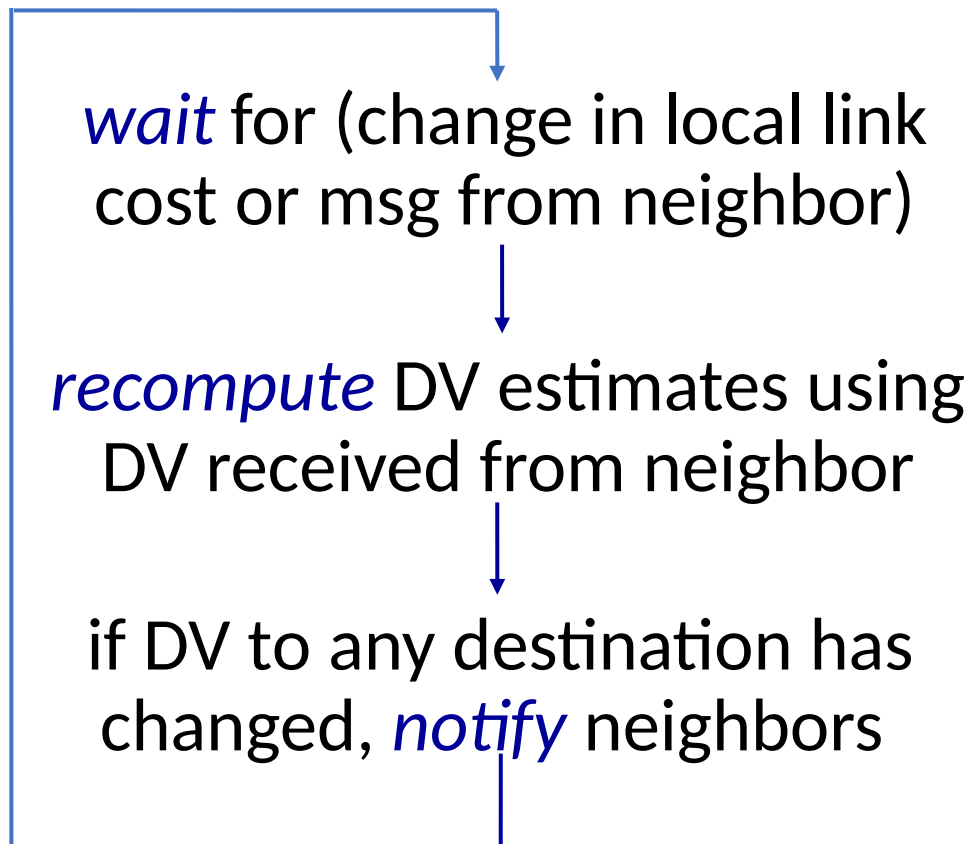
Bellman-Ford equation says:

$$\begin{aligned} D_u(z) &= \min \{ c_{u,v} + D_v(z), \\ &\quad c_{u,x} + D_x(z), \\ &\quad c_{u,w} + D_w(z) \} \\ &= \min \{ 2 + 5, \\ &\quad 1 + 3, \\ &\quad 5 + 3 \} = 4 \end{aligned}$$

node achieving minimum (x) is next hop on estimated least-cost path to destination (z)

Distance vector algorithm:

each node:



iterative, asynchronous: each local iteration caused by:






- local link cost change
- DV update message from neighbor

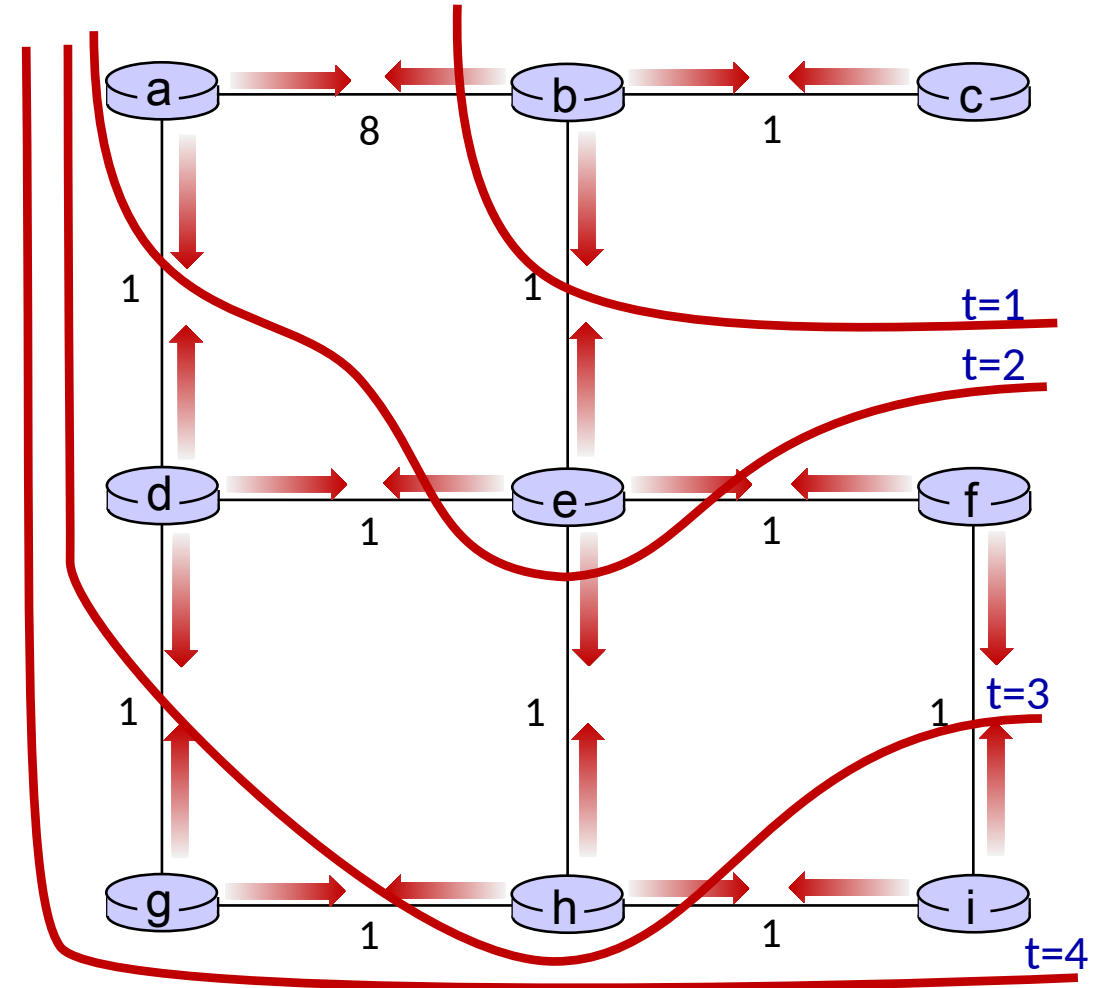
distributed, self-stopping: each node notifies neighbors *only* when its DV changes

- neighbors then notify their neighbors – *only if necessary*
- no notification received, no actions taken!

Distance vector: state information diffusion

Iterative communication, computation steps diffuses information through network:

-  $t=0$ c's state at $t=0$ is at c only
-  $t=1$ c's state at $t=0$ has propagated to b, and may influence distance vector computations up to **1** hop away, i.e., at b
-  $t=2$ c's state at $t=0$ may now influence distance vector computations up to **2** hops away, i.e., at b and now at a, e as well
-  $t=3$ c's state at $t=0$ may influence distance vector computations up to **3** hops away, i.e., at b,a,e and now at c,f,h as well
-  $t=4$ c's state at $t=0$ may influence distance vector computations up to **4** hops away, i.e., at b,a,e, c, f, h and now at g,i as well



Comparison of LS and DV algorithms

message complexity

LS: n routers, $O(n^2)$ messages sent

DV: exchange between neighbors;
convergence time varies

speed of convergence

LS: $O(n^2)$ algorithm, $O(n^2)$ messages

- may have oscillations

DV: convergence time varies

- may have routing loops
- count-to-infinity problem

robustness: what happens if router malfunctions, or is compromised?

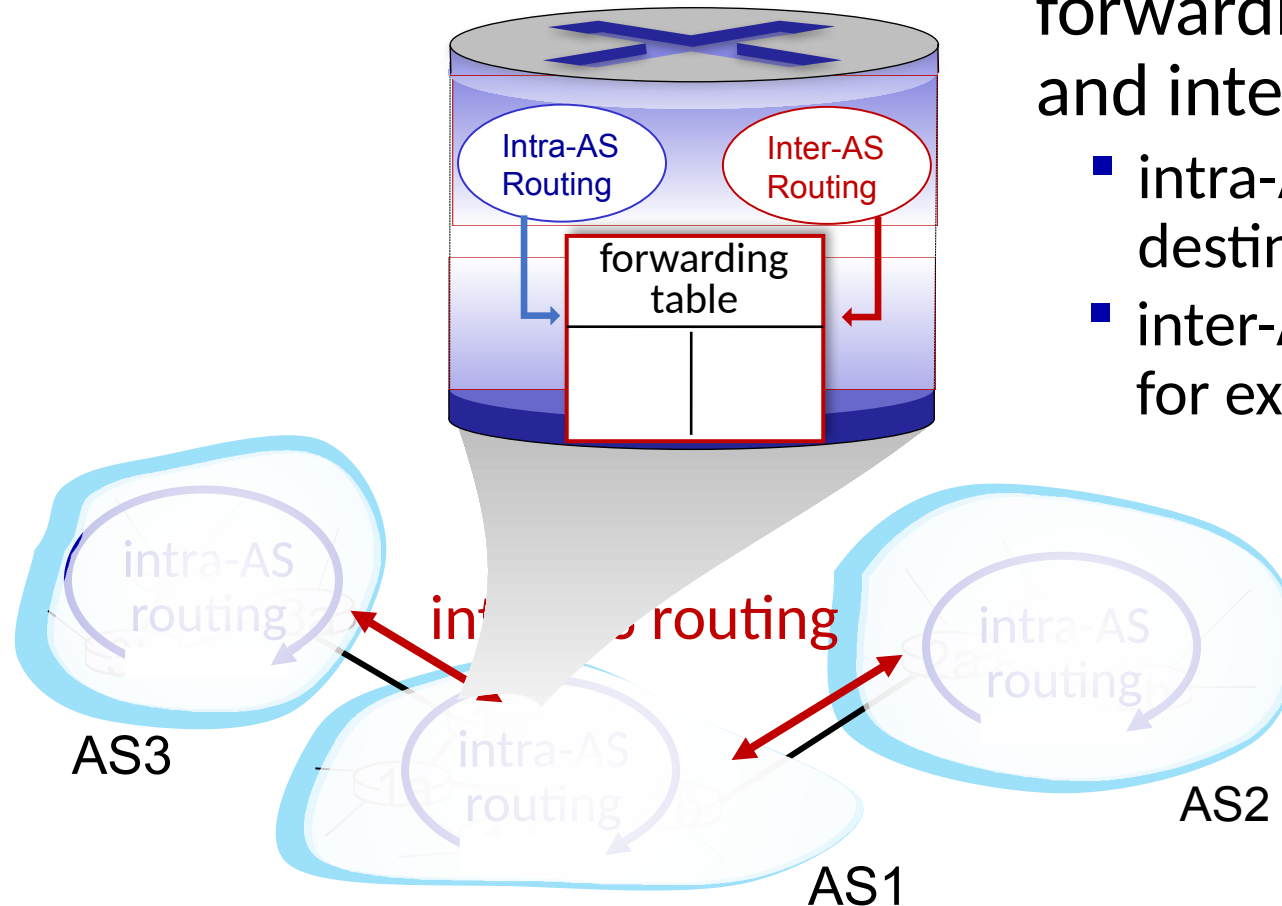
LS:

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

DV:

- DV router can advertise incorrect *path* cost (“I have a *really* low cost path to everywhere”): black-holing
- each router’s table used by others: error propagate thru network

Interconnected ASes



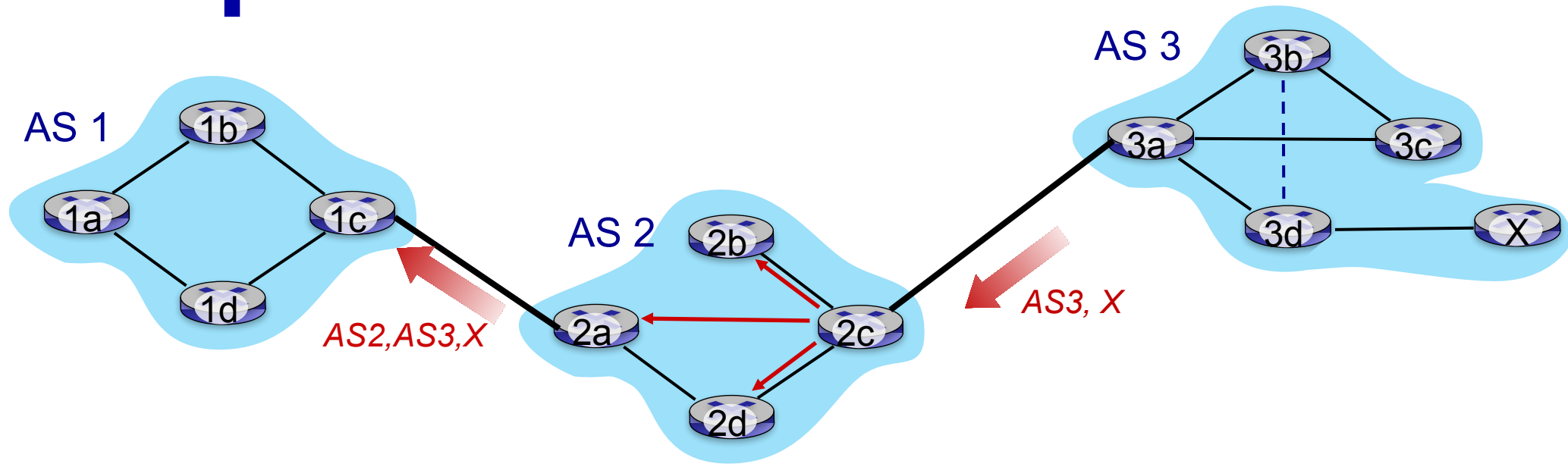
forwarding table configured by intra- and inter-AS routing algorithms

- intra-AS routing determine entries for destinations within AS
- inter-AS & intra-AS determine entries for external destinations

OSPF (Open Shortest Path First) routing

- “open”: publicly available
- classic link-state
 - each router floods OSPF link-state advertisements (directly over IP rather than using TCP/UDP) to all other routers in entire AS
 - multiple link costs metrics possible: bandwidth, delay
 - each router has full topology, uses Dijkstra’s algorithm to compute forwarding table
- *security*: all OSPF messages authenticated (to prevent malicious intrusion)

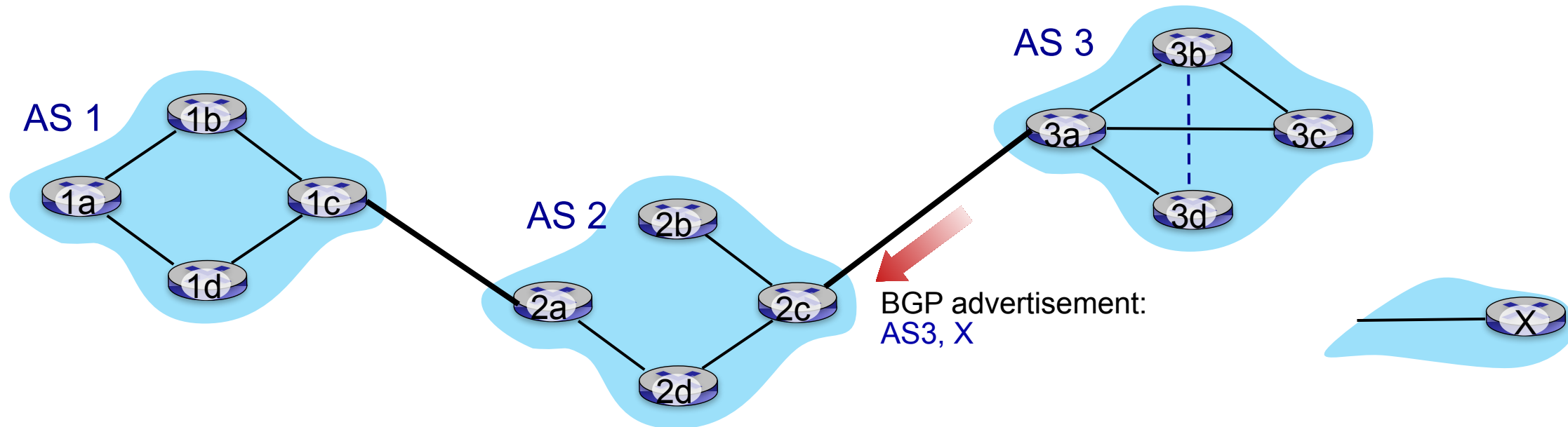
BGP path advertisement



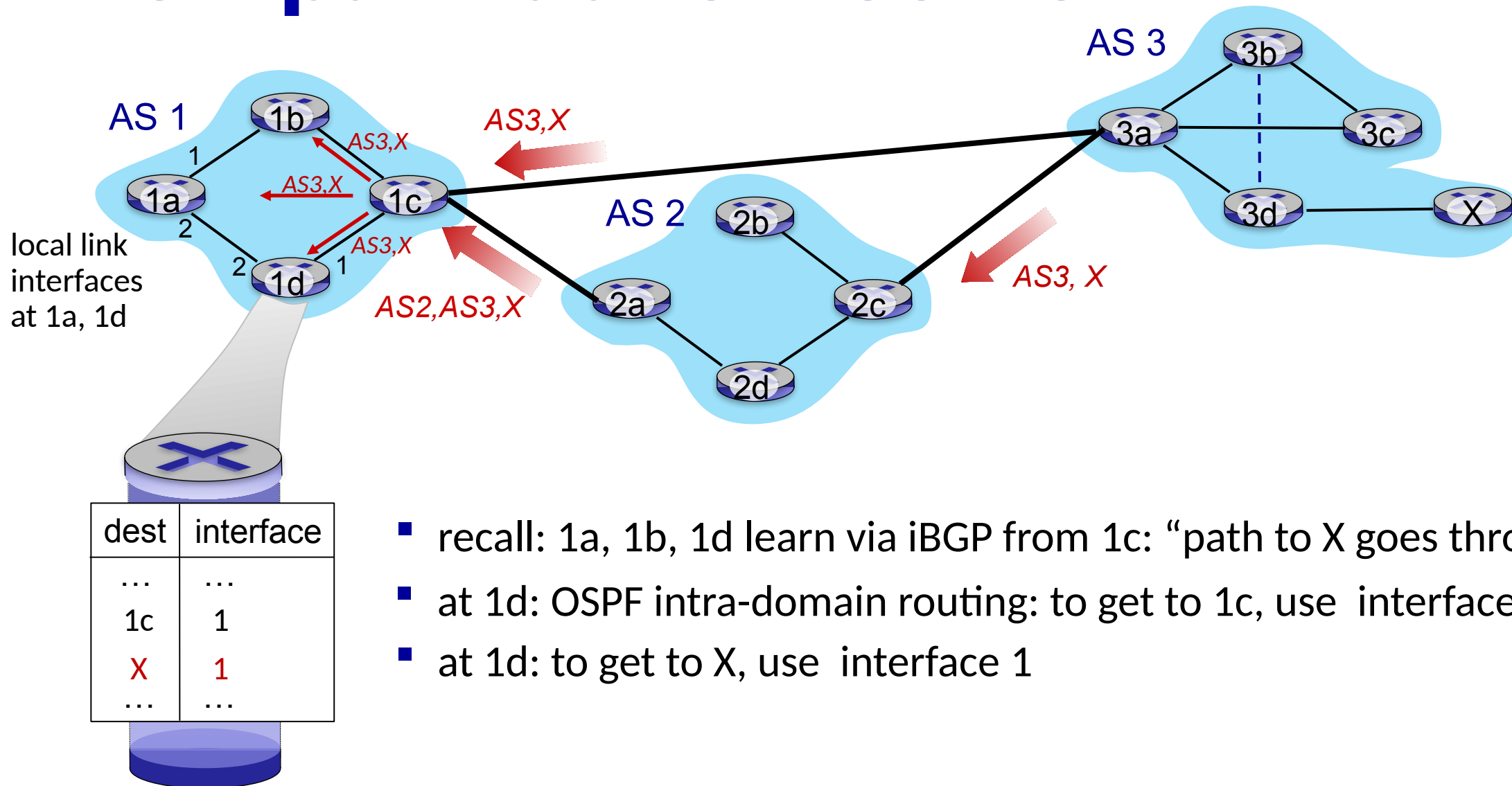
- AS2 router 2c receives path advertisement **AS3, X** (via eBGP) from AS3 router 3a
- based on AS2 policy, AS2 router 2c accepts path AS3, X, propagates (via iBGP) to all AS2 routers
- based on AS2 policy, AS2 router 2a advertises (via eBGP) path **AS2, AS3, X** to AS1 router 1c

BGP basics

- **BGP session:** two BGP routers (“peers”) exchange BGP messages over semi-permanent TCP connection:
 - advertising *paths* to different destination network prefixes (BGP is a “path vector” protocol)
- when AS3 gateway 3a advertises *path AS3,X* to AS2 gateway 2c:
 - AS3 *promises* to AS2 it will forward datagrams towards X

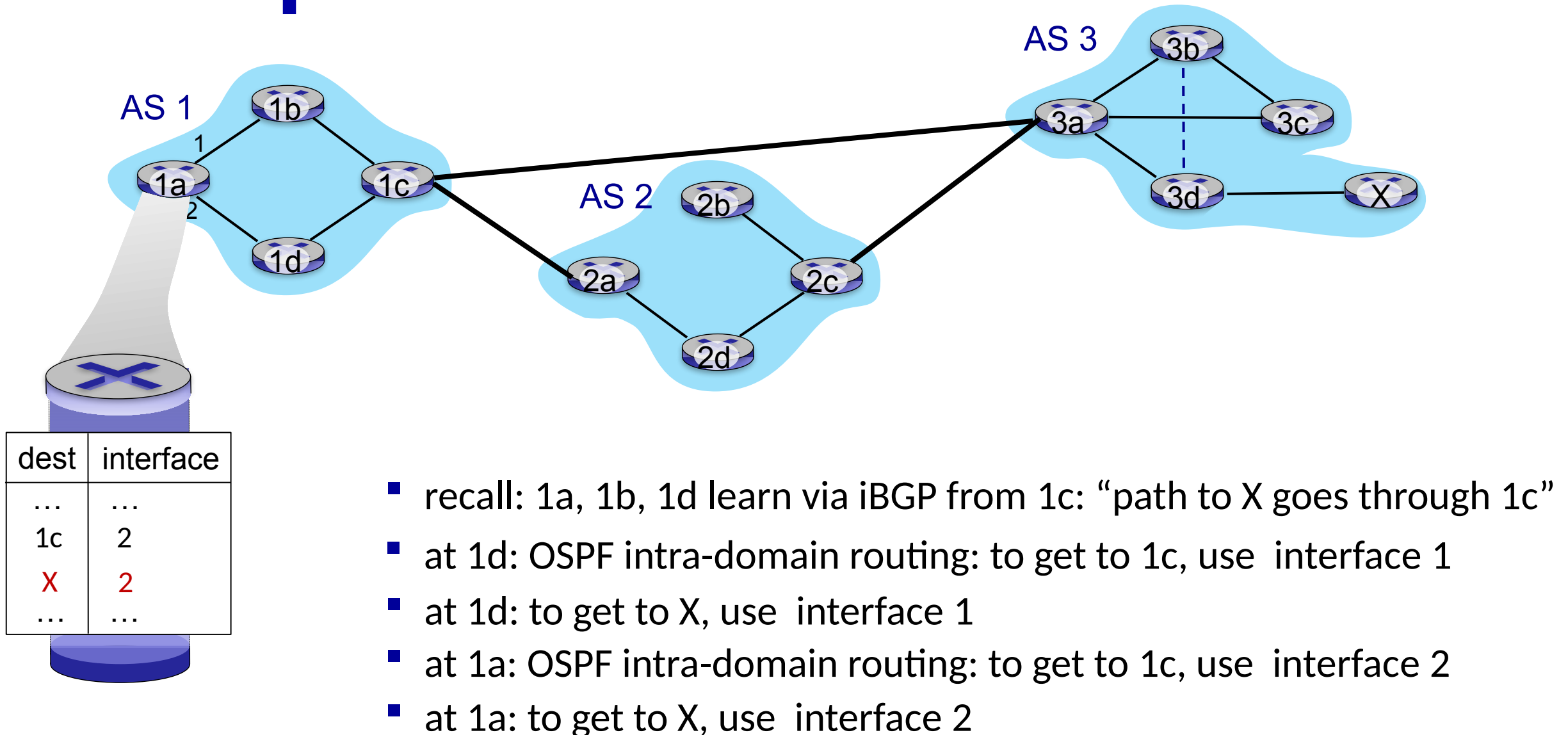


BGP path advertisement

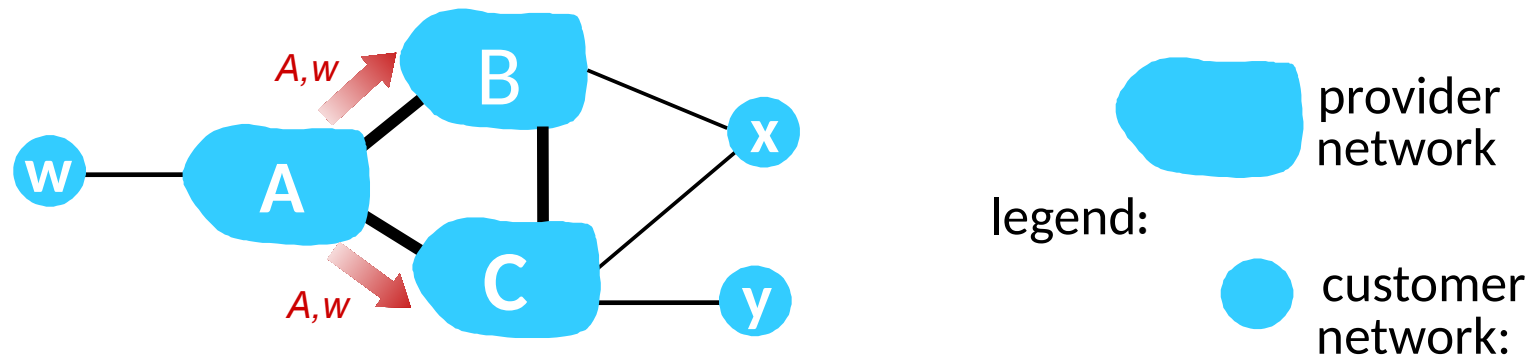


- recall: 1a, 1b, 1d learn via iBGP from 1c: “path to X goes through 1c”
- at 1d: OSPF intra-domain routing: to get to 1c, use interface 1
- at 1d: to get to X, use interface 1

BGP path advertisement



BGP: achieving policy via advertisements



ISP only wants to route traffic to/from its customer networks (does not want to carry transit traffic between other ISPs – a typical “real world” policy)

- A advertises path Aw to B and to C
- B *chooses not to advertise* BA_w to C!
 - B gets no “revenue” for routing CBA_w, since none of C, A, w are B’s customers
 - C does *not* learn about CBA_w path
- C will route CA_w (not using B) to get to w

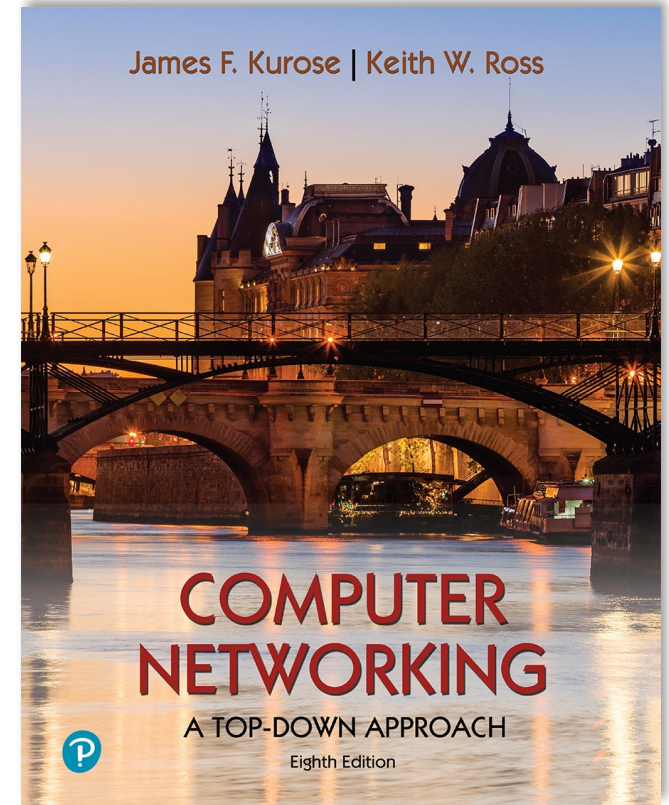
Chapter 6

The Link

Layer

and LANs

CS 455/555
Parth Pathak



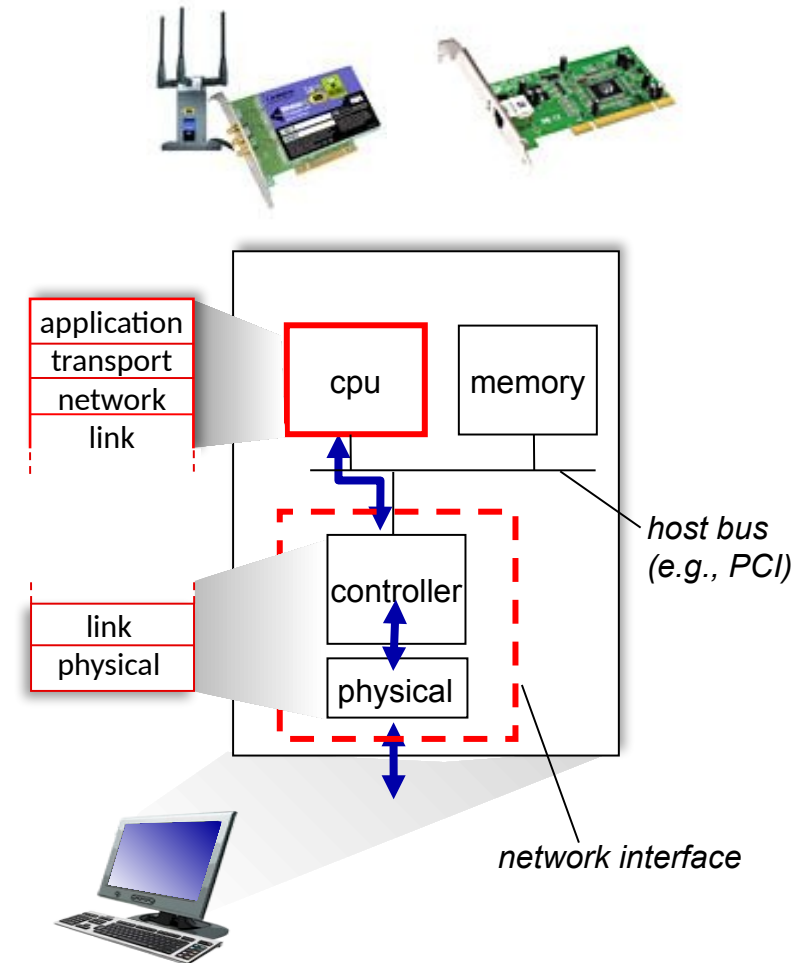
Computer Networking: A Top-Down Approach

8th edition

Jim Kurose, Keith Ross
Pearson, 2020

Where is the link layer implemented?

- in each-and-every host
- link layer implemented in *network interface card* (NIC) or on a chip
 - Ethernet, WiFi card or chip
 - implements link, physical layer
- attaches into host's system buses
- combination of hardware, software, firmware



Internet checksum (review)

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

sender:

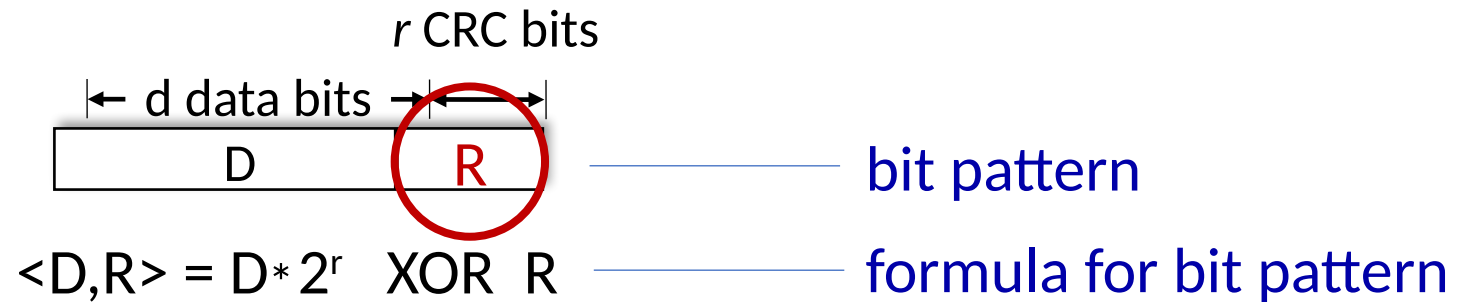
- treat contents of UDP segment (including UDP header fields and IP addresses) as sequence of 16-bit integers
- **checksum:** addition (one's complement sum) of segment content
- checksum value put into UDP checksum field

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*

Cyclic Redundancy Check (CRC)

- more powerful error-detection coding
- **D**: data bits (given, think of these as a binary number)
- **G**: bit pattern (generator), of $r+1$ bits (given)



goal: choose r CRC bits, **R**, such that $\langle D, R \rangle$ exactly divisible by $G \pmod{2}$

- receiver knows G , divides $\langle D, R \rangle$ by G . If non-zero remainder: error detected!
- can detect all burst errors less than $r+1$ bits
- widely used in practice (Ethernet, 802.11 WiFi)

Cyclic Redundancy Check (CRC): example

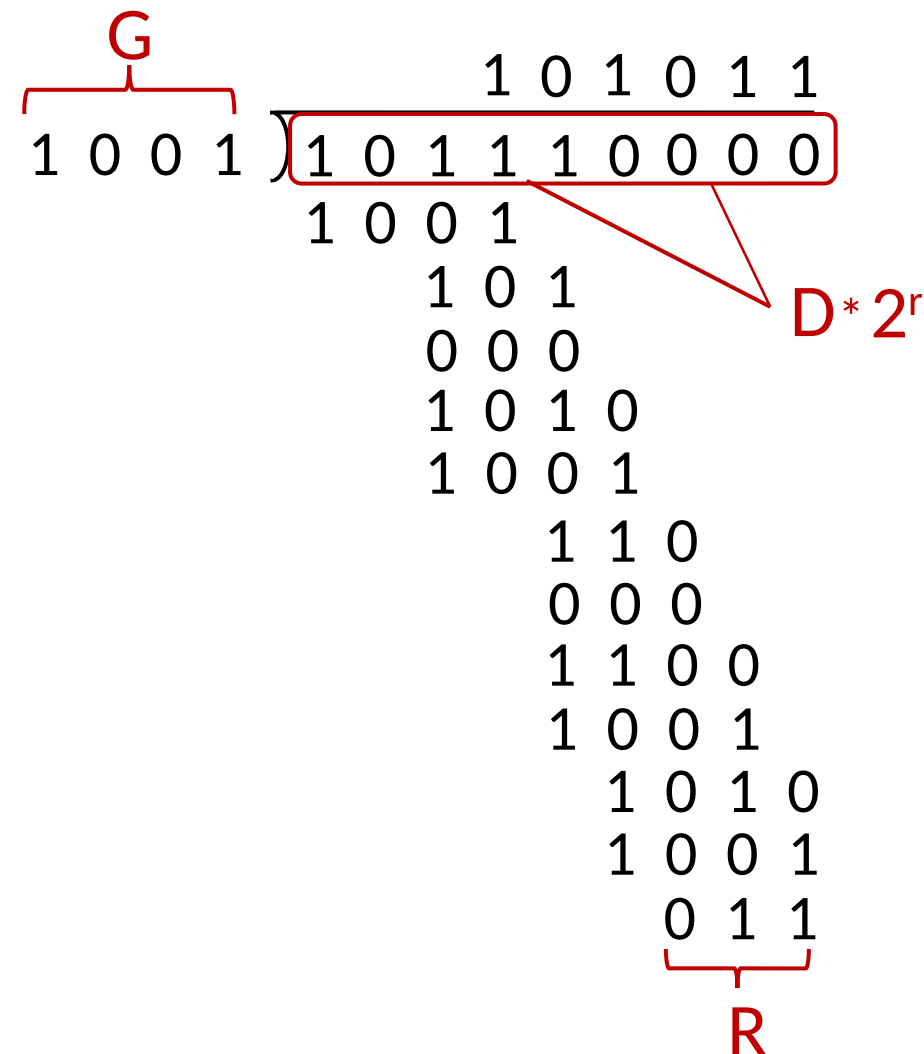
We want:

$$D \cdot 2^r \text{ XOR } R = nG$$

or equivalently:

if we divide $D \cdot 2^r$ by G , want remainder R to satisfy:

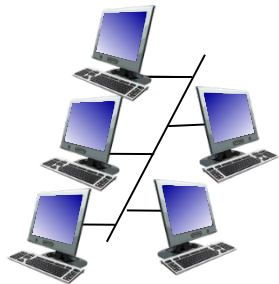
$$R = remainder \left[\frac{D \cdot 2^r}{G} \right]$$



Multiple access links, protocols

two types of “links”:

- point-to-point
 - point-to-point link between Ethernet switch, host
 - PPP for dial-up access
- **broadcast (shared wire or medium)**
 - old-fashioned Ethernet
 - upstream HFC in cable-based access network
 - 802.11 wireless LAN, 4G/4G. satellite



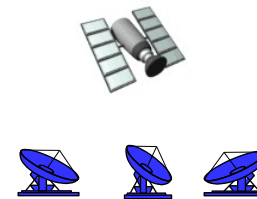
shared wire (e.g.,
cabled Ethernet)



shared radio: 4G/5G



shared radio: WiFi



shared radio: satellite



humans at a cocktail party
(shared air, acoustical)

MAC protocols: taxonomy

three broad classes:

- **channel partitioning**

- divide channel into smaller “pieces” (time slots, frequency, code)
- allocate piece to node for exclusive use

- ***random access***

- channel not divided, allow collisions
- “recover” from collisions

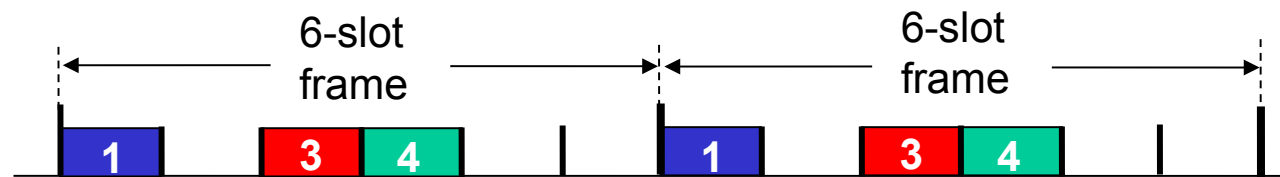
- **“taking turns”**

- nodes take turns, but nodes with more to send can take longer turns

Channel partitioning MAC protocols: TDMA

TDMA: time division multiple access

- access to channel in “rounds”
- each station gets fixed length slot (length = packet transmission time) in each round
- unused slots go idle
- example: 6-station LAN, 1,3,4 have packets to send, slots 2,5,6 idle



Slotted ALOHA

assumptions:

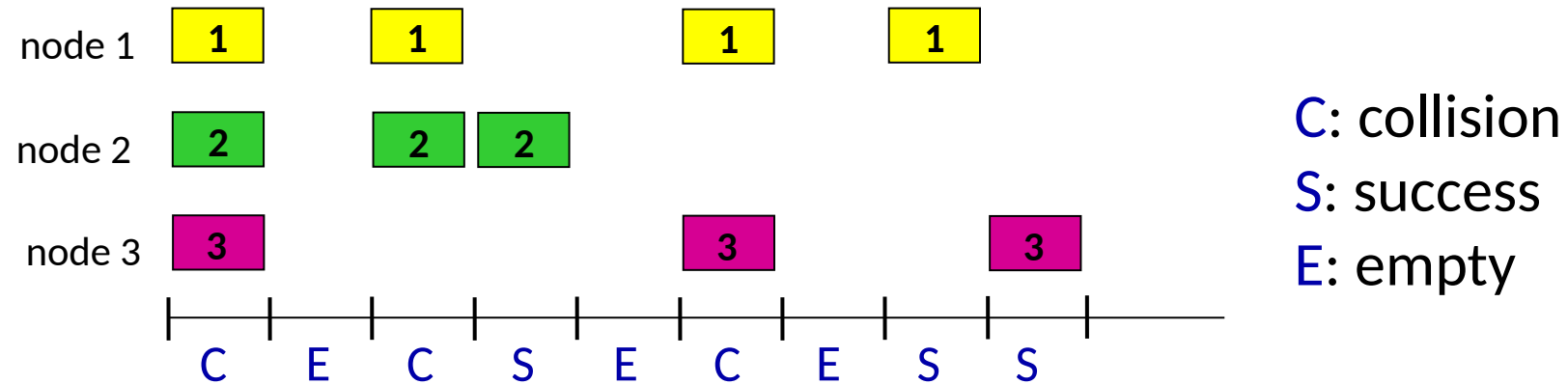
- all frames same size
- time divided into equal size slots (time to transmit 1 frame)
- nodes start to transmit only slot beginning
- nodes are synchronized
- if 2 or more nodes transmit in slot, all nodes detect collision

operation:

- 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 probability p until success

randomization – why?

Slotted ALOHA



Pros:

- single active node can continuously transmit at full rate of channel
- highly decentralized: only slots in nodes need to be in sync
- simple

Cons:

- collisions, wasting slots
- idle slots
- nodes may be able to detect collision in less than time to transmit packet
- clock synchronization

Slotted ALOHA: efficiency

efficiency: long-run fraction of successful slots (many nodes, all with many frames to send)

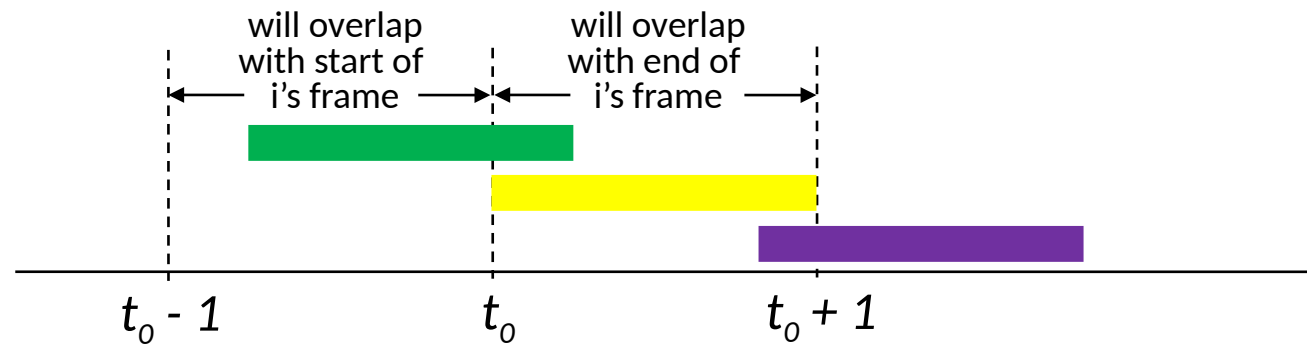
- *suppose:* N nodes with many frames to send, each transmits in slot with probability p
 - prob that given node has success in a slot $= p(1-p)^{N-1}$
 - prob that *any* node has a success $= Np(1-p)^{N-1}$
 - max efficiency: find p^* that maximizes $Np(1-p)^{N-1}$
 - for many nodes, take limit of $Np^*(1-p^*)^{N-1}$ as N goes to infinity, gives:

max efficiency $= 1/e = .37$

- *at best:* channel used for useful transmissions 37% of time!

Pure ALOHA

- unslotted Aloha: simpler, no synchronization
 - when frame first arrives: transmit immediately
- collision probability increases with no synchronization:
 - frame sent at t_0 collides with other frames sent in $[t_0-1, t_0+1]$



- pure Aloha efficiency: 18% !

CSMA (carrier sense multiple access)

simple **CSMA**: listen before transmit:

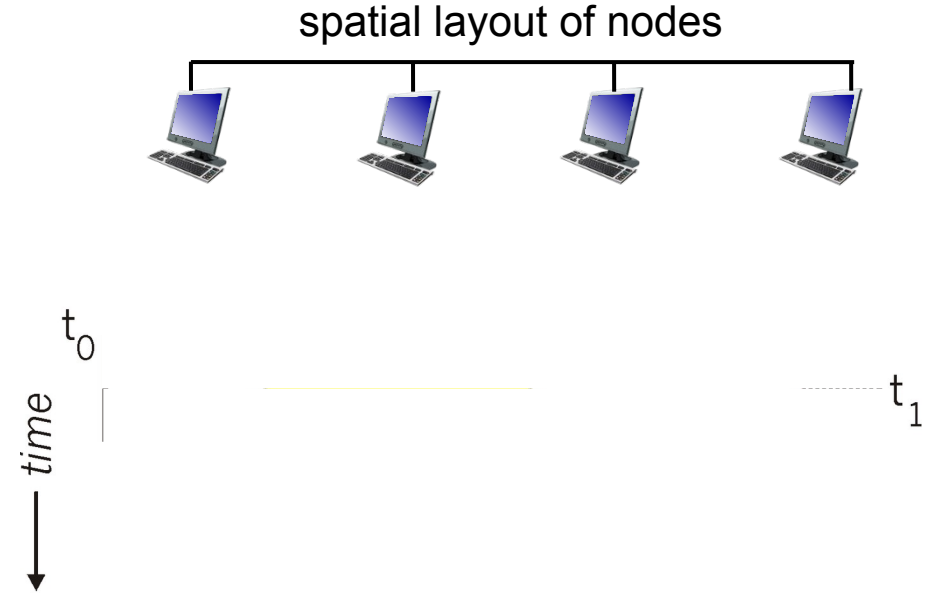
- if channel sensed **idle**: transmit entire frame
 - if channel sensed **busy**: defer transmission
- human analogy: don't interrupt others!

Can collisions still occur?



CSMA: collisions

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



CSMA (carrier sense multiple access)

simple **CSMA**: listen before transmit:

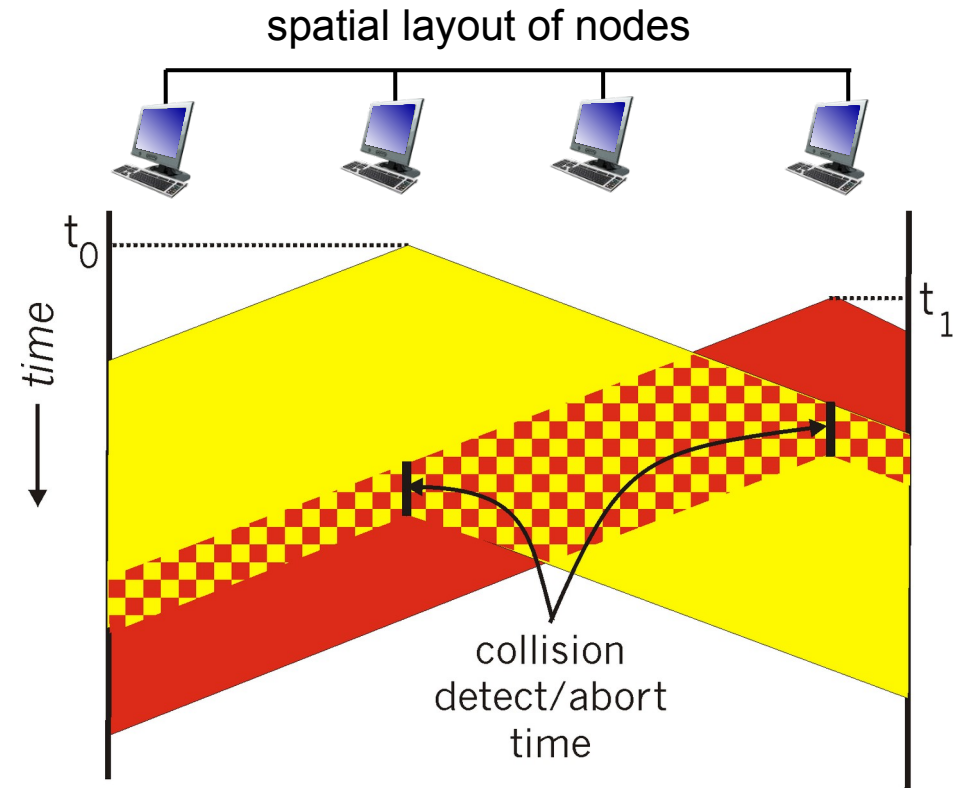
- if channel sensed **idle**: transmit entire frame
 - if channel sensed **busy**: defer transmission
- human analogy: don't interrupt others!

CSMA/CD: CSMA with *collision detection*

- collisions *detected* within short time
 - colliding transmissions aborted, reducing channel wastage
 - collision detection easy in wired, difficult with wireless
- human analogy: the polite conversationalist

CSMA/CD:

- CSMA/CS reduces the amount of time wasted in collisions
 - transmission aborted on collision detection



Ethernet CSMA/CD algorithm

1. NIC receives datagram from network layer, creates frame
2. If NIC senses channel:
 - if **idle**: start frame transmission.
 - if **busy**: wait until channel idle, then transmit
3. If NIC transmits entire frame without collision, NIC is done with frame !
4. If NIC detects another transmission while sending: abort, send jam signal, got to step 1



Any possible issues
executing this algorithm?

Ethernet CSMA/CD algorithm

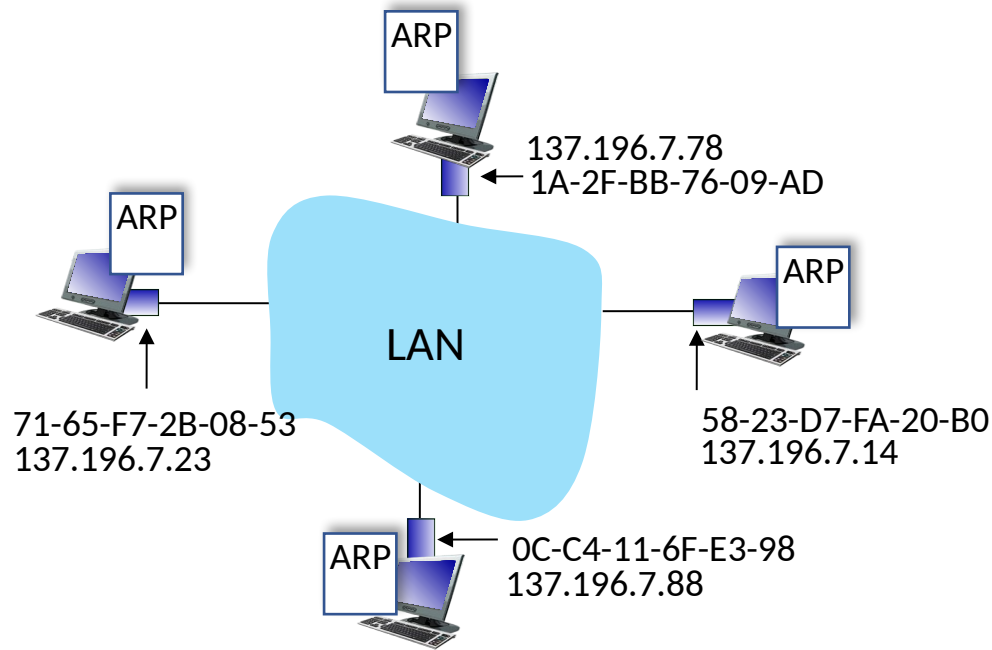
1. NIC receives datagram from network layer, creates frame
2. If NIC senses channel:
 - if **idle**: start frame transmission.
 - if **busy**: wait until channel idle, then transmit
3. If NIC transmits entire frame without collision, NIC is done with frame !
4. If NIC detects another transmission while sending: abort, send jam signal
5. After aborting, NIC enters *binary (exponential) backoff*:
 - after m th collision, NIC chooses K at random from $\{0, 1, 2, \dots, 2^m - 1\}$. NIC waits $K \cdot 512$ bit times, returns to Step 2
 - more collisions: longer backoff interval

MAC addresses

- 32-bit IP address:
 - *network-layer* address for interface
 - used for layer 3 (network layer) forwarding
 - e.g.: 128.119.40.136
- MAC (or LAN or physical or Ethernet) address:
 - function: used “locally” to get frame from one interface to another physically-connected interface (same subnet, in IP-addressing sense)
 - 48-bit MAC address (for most LANs) burned in NIC ROM, also sometimes software settable
 - e.g.: 1A-2F-BB-76-09-AD
 - hexadecimal (base 16) notation
(each “numeral” represents 4 bits)

ARP: address resolution protocol

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 in action

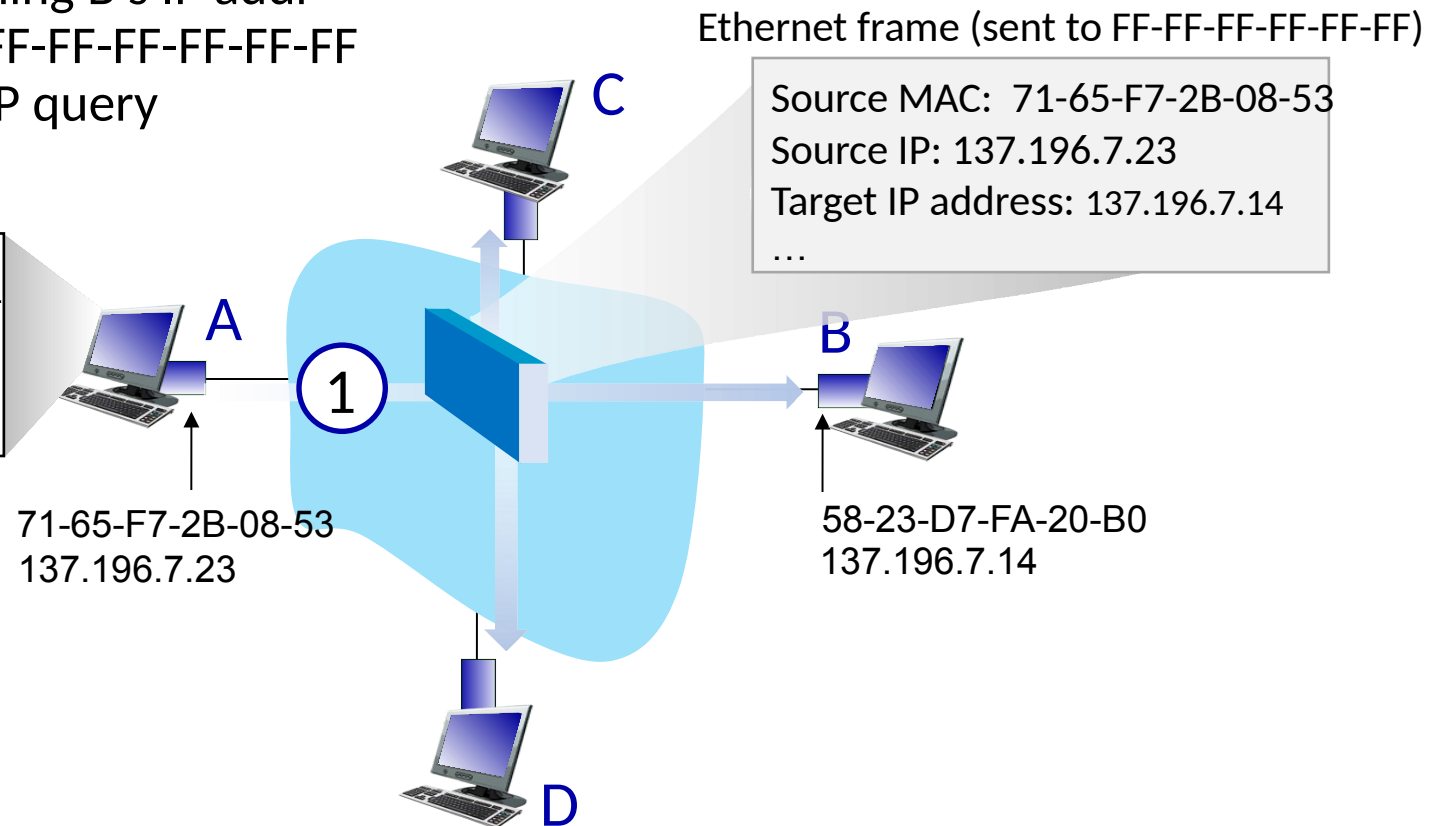
example: A wants to send datagram to B

- B's MAC address not in A's ARP table, so A uses ARP to find B's MAC address

- 1 A broadcasts ARP query, containing B's IP addr
- destination MAC address = FF-FF-FF-FF-FF-FF
 - all nodes on LAN receive ARP query

ARP table in A

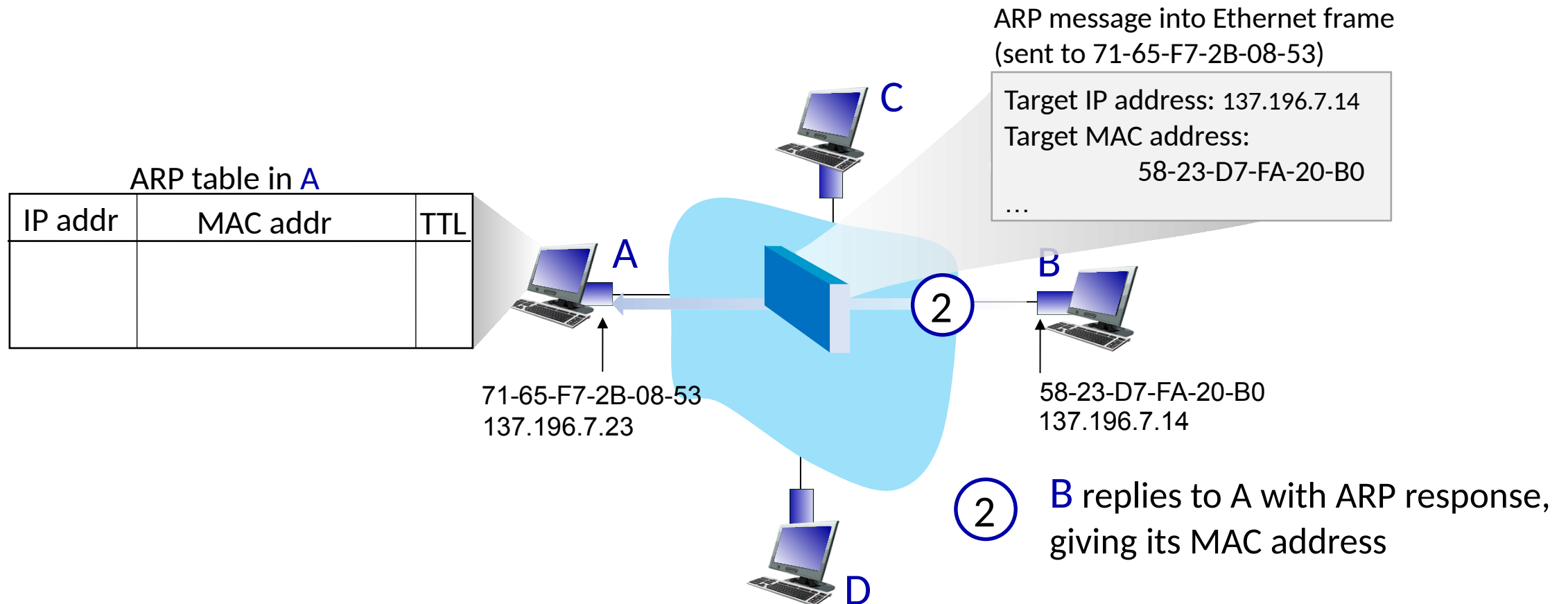
IP addr	MAC addr	TTL



ARP protocol in action

example: A wants to send datagram to B

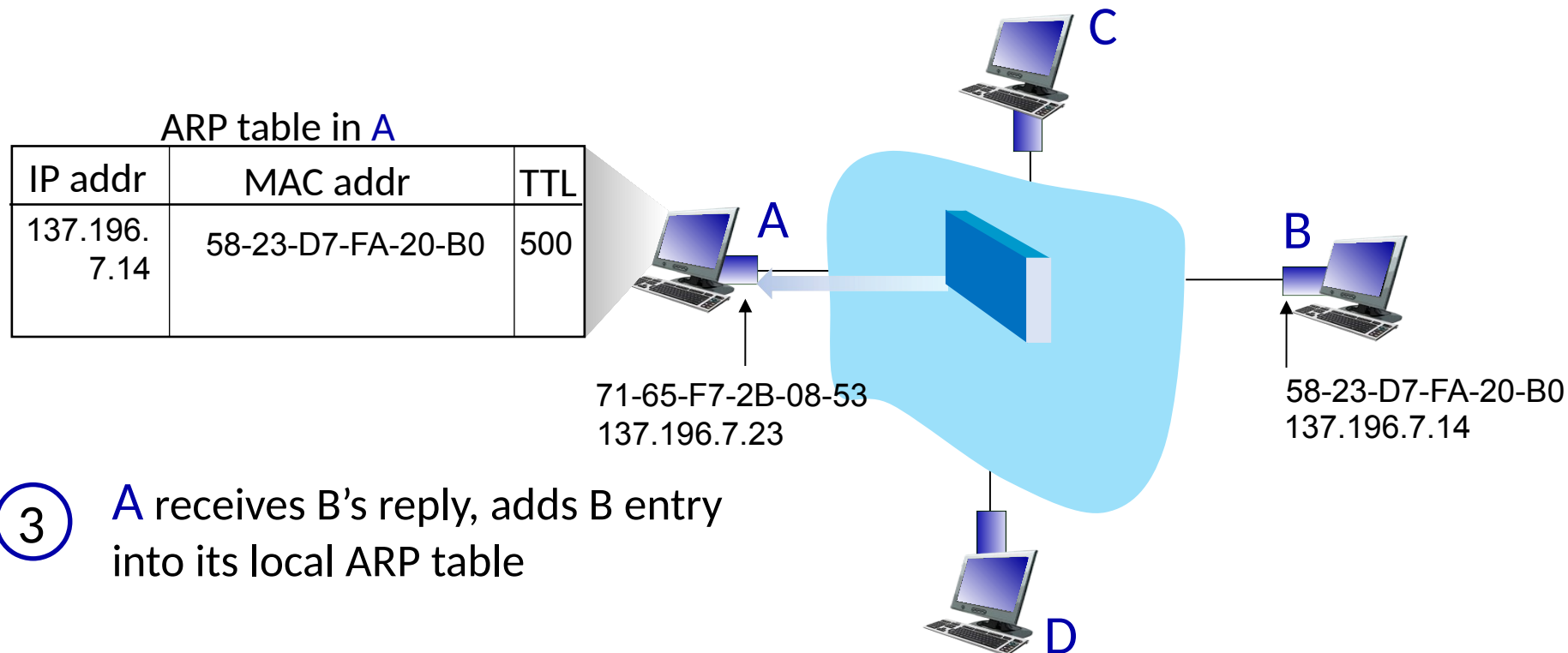
- B's MAC address not in A's ARP table, so A uses ARP to find B's MAC address



ARP protocol in action

example: A wants to send datagram to B

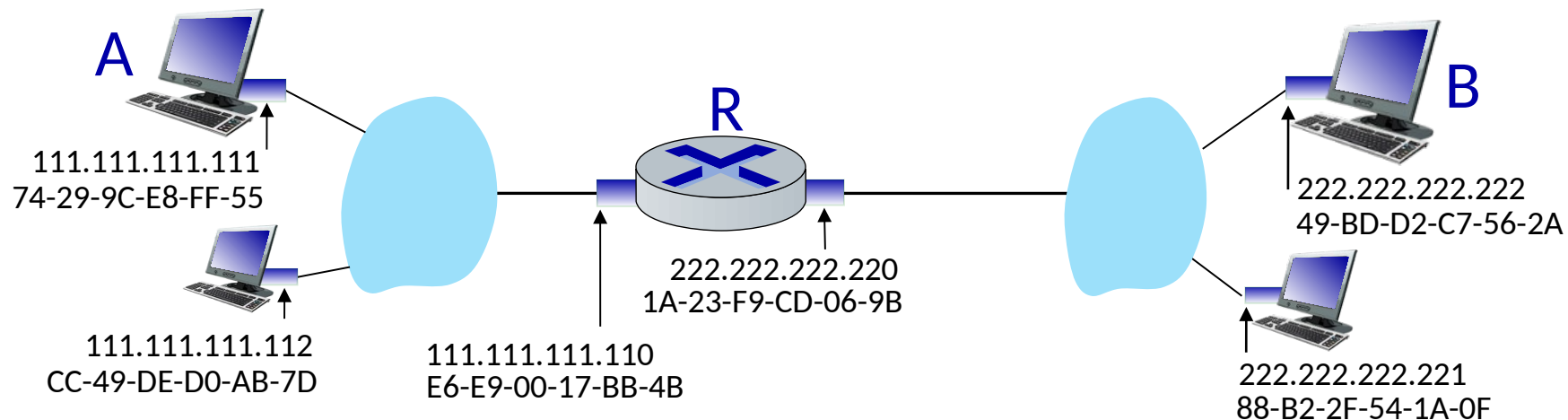
- B's MAC address not in A's ARP table, so A uses ARP to find B's MAC address



Routing to another subnet: addressing

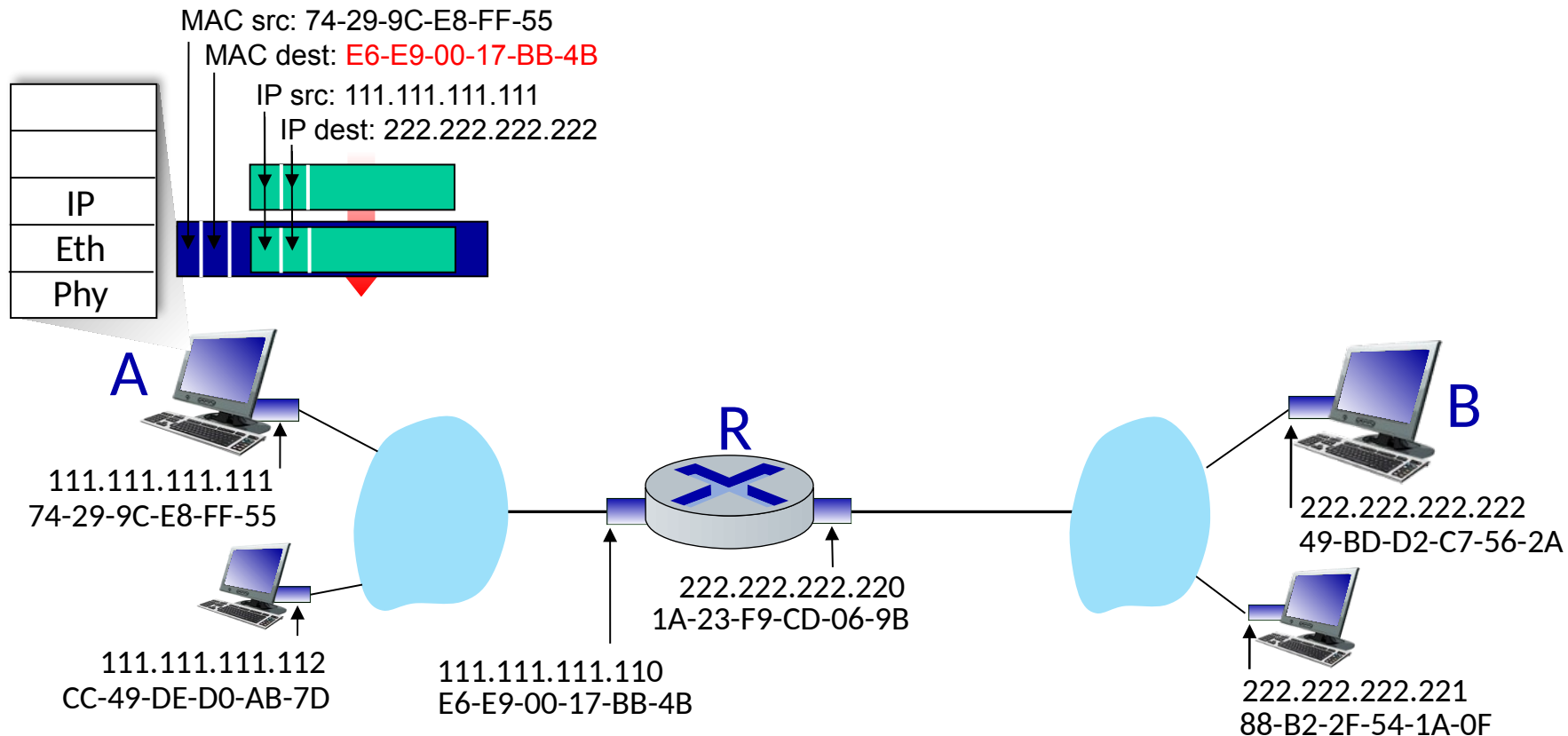
walkthrough: sending a datagram from A to B via R

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



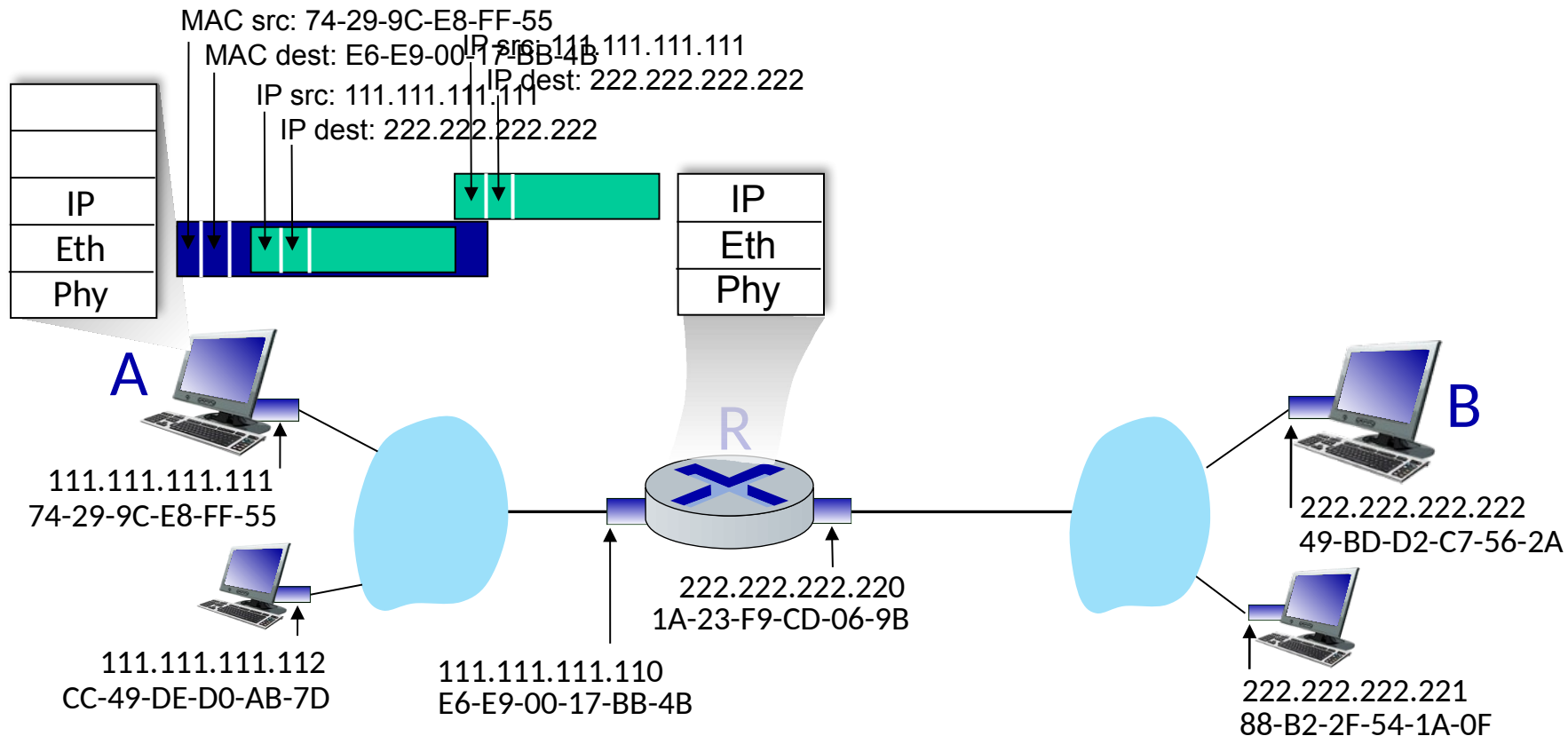
Routing to another subnet: addressing

- A creates IP datagram with IP source A, destination B
- A creates link-layer frame containing A-to-B IP datagram
 - **R's** MAC address is frame's destination



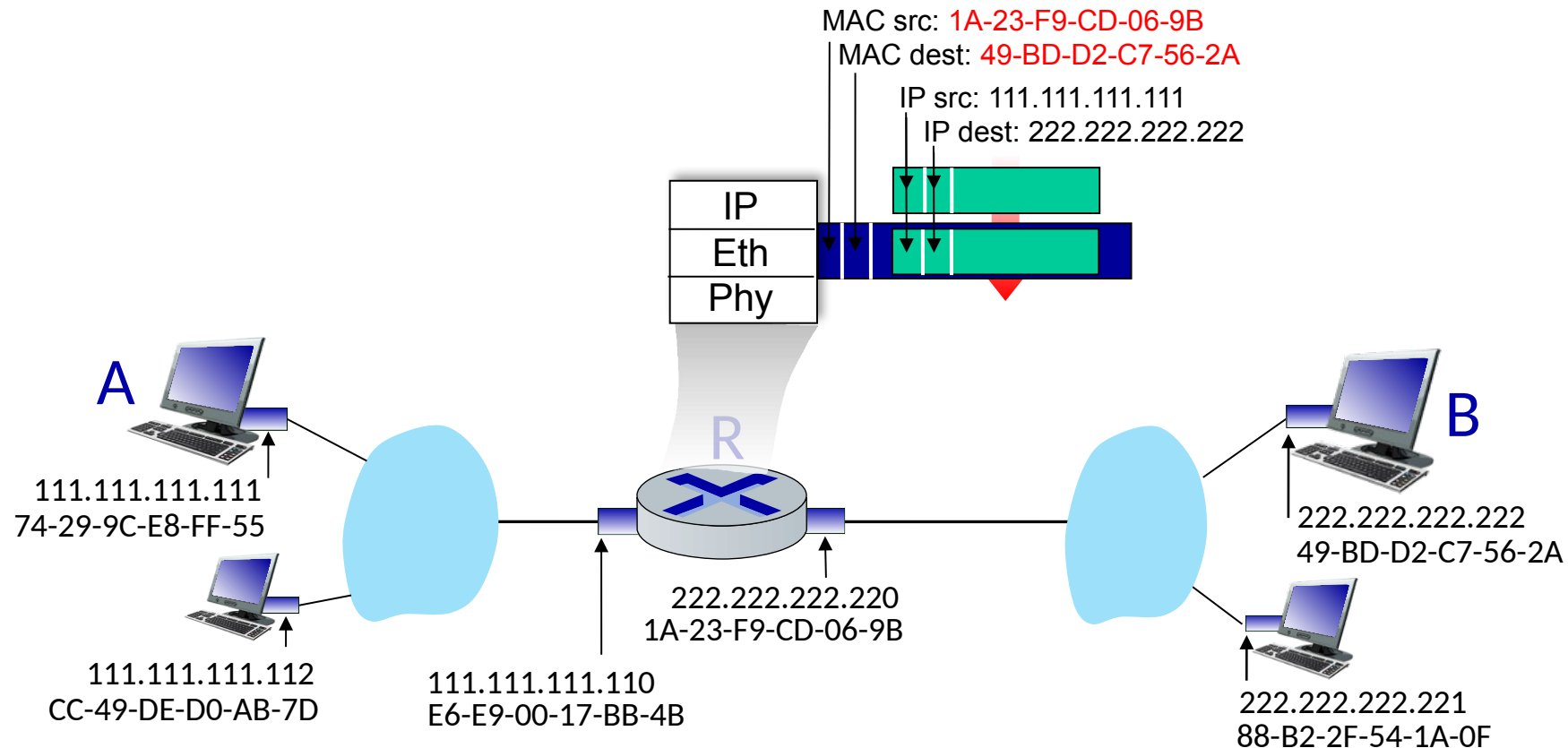
Routing to another subnet: addressing

- frame sent from A to R
- frame received at R, datagram removed, passed up to IP



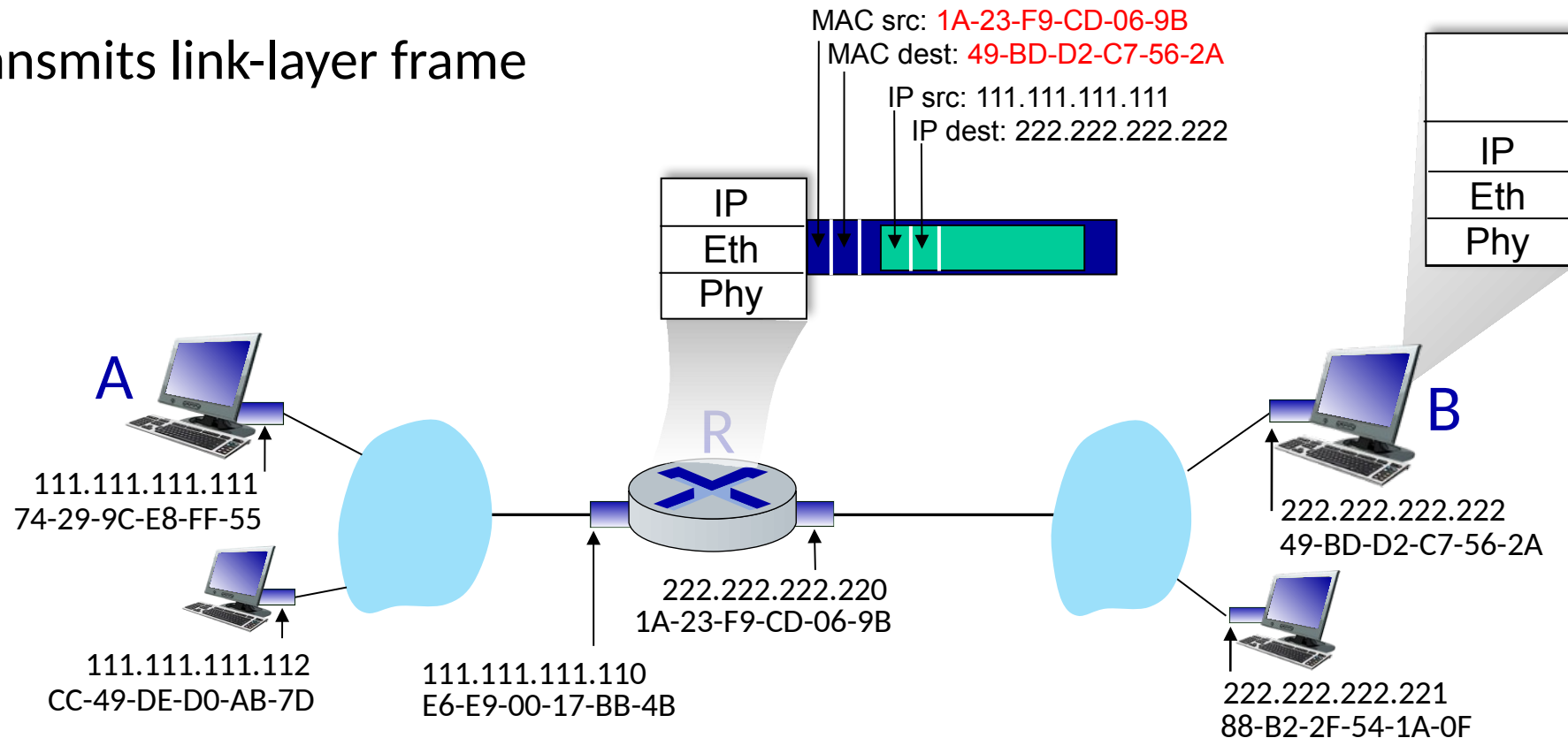
Routing to another subnet: addressing

- R determines outgoing interface, passes datagram with IP source A, destination B to link layer
- R creates link-layer frame containing A-to-B IP datagram. Frame destination address: B's MAC address



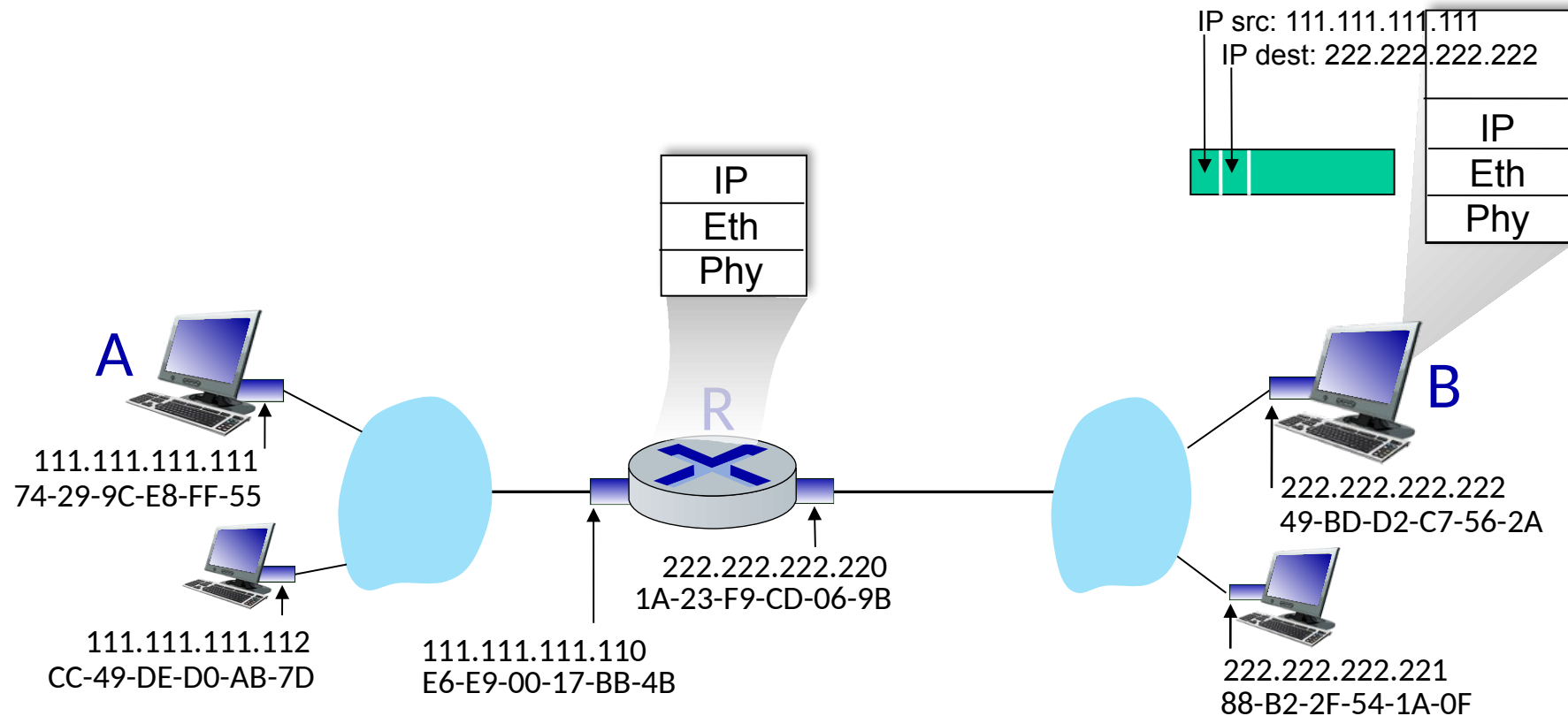
Routing to another subnet: addressing

- R determines outgoing interface, passes datagram with IP source A, destination B to link layer
- R creates link-layer frame containing A-to-B IP datagram. Frame destination address: B's MAC address
- transmits link-layer frame



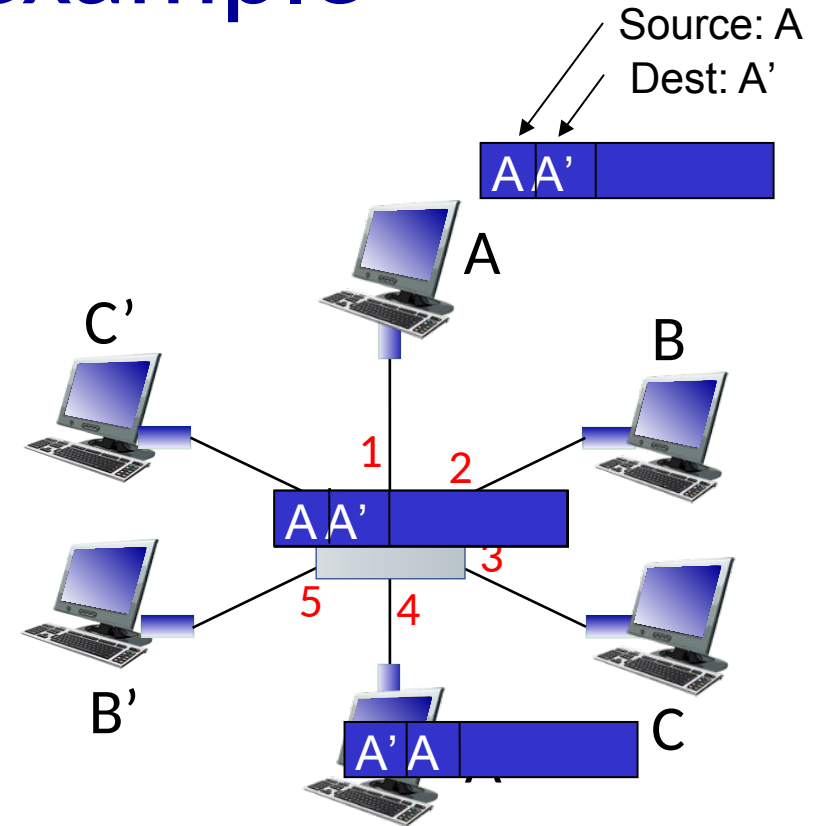
Routing to another subnet: addressing

- B receives frame, extracts IP datagram destination B
- B passes datagram up protocol stack to IP



Self-learning, forwarding: example

- frame destination, A',
location unknown: **flood**
- destination A location
known: **selectively send**
on just one link



MAC addr	interface	TTL
A	1	60
A'	4	60

*switch table
(initially empty)*

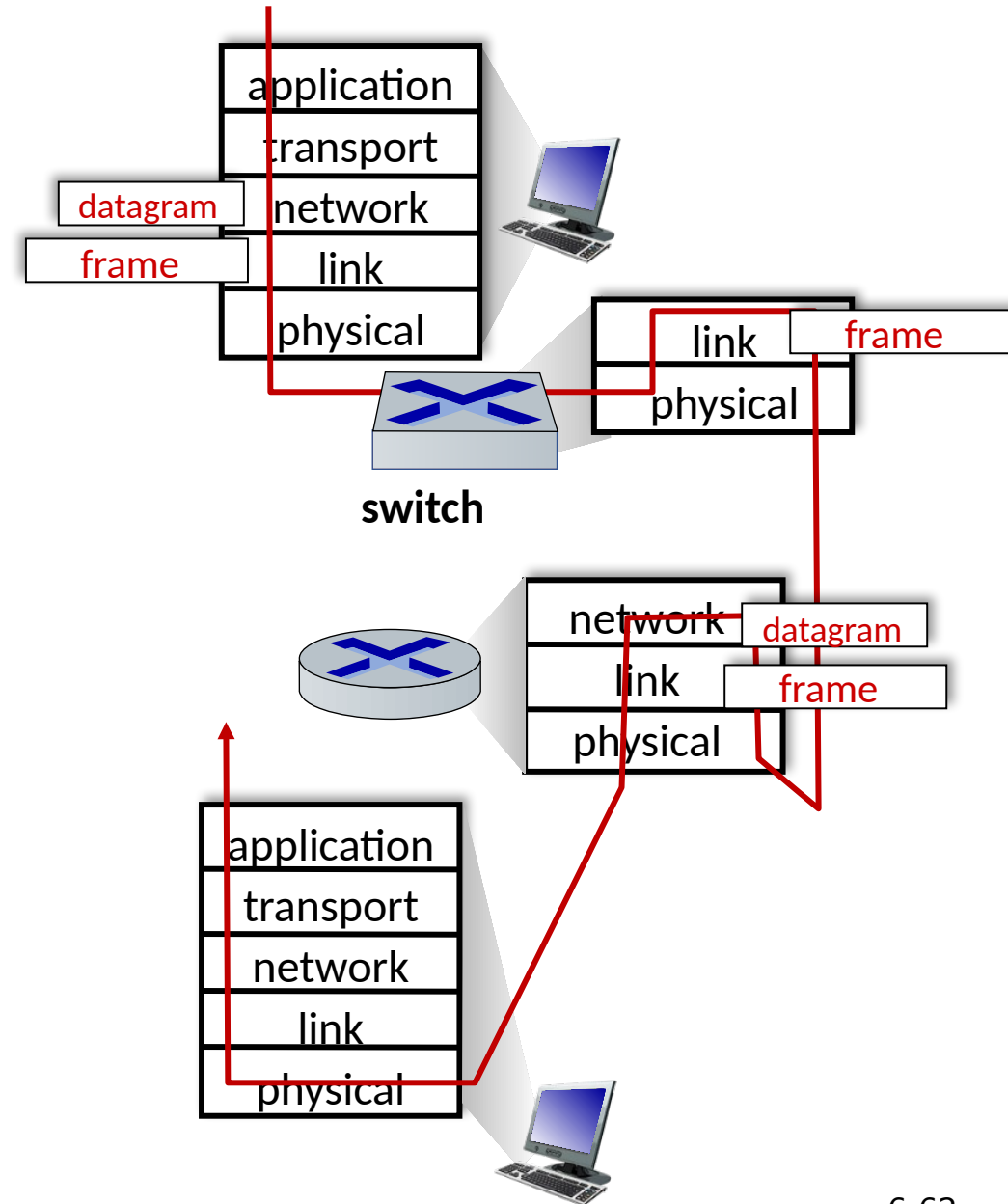
Switches vs. routers

both are store-and-forward:

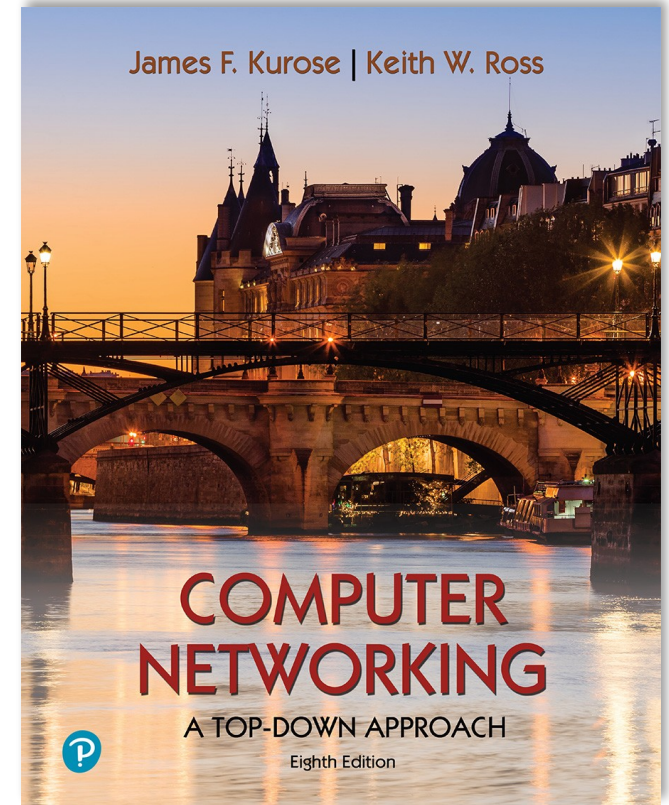
- *routers*: network-layer devices (examine network-layer headers)
- *switches*: link-layer devices (examine link-layer headers)

both have forwarding tables:

- *routers*: compute tables using routing algorithms, IP addresses
- *switches*: learn forwarding table using flooding, learning, MAC addresses



Chapter 7 Wireless and Mobile Networks



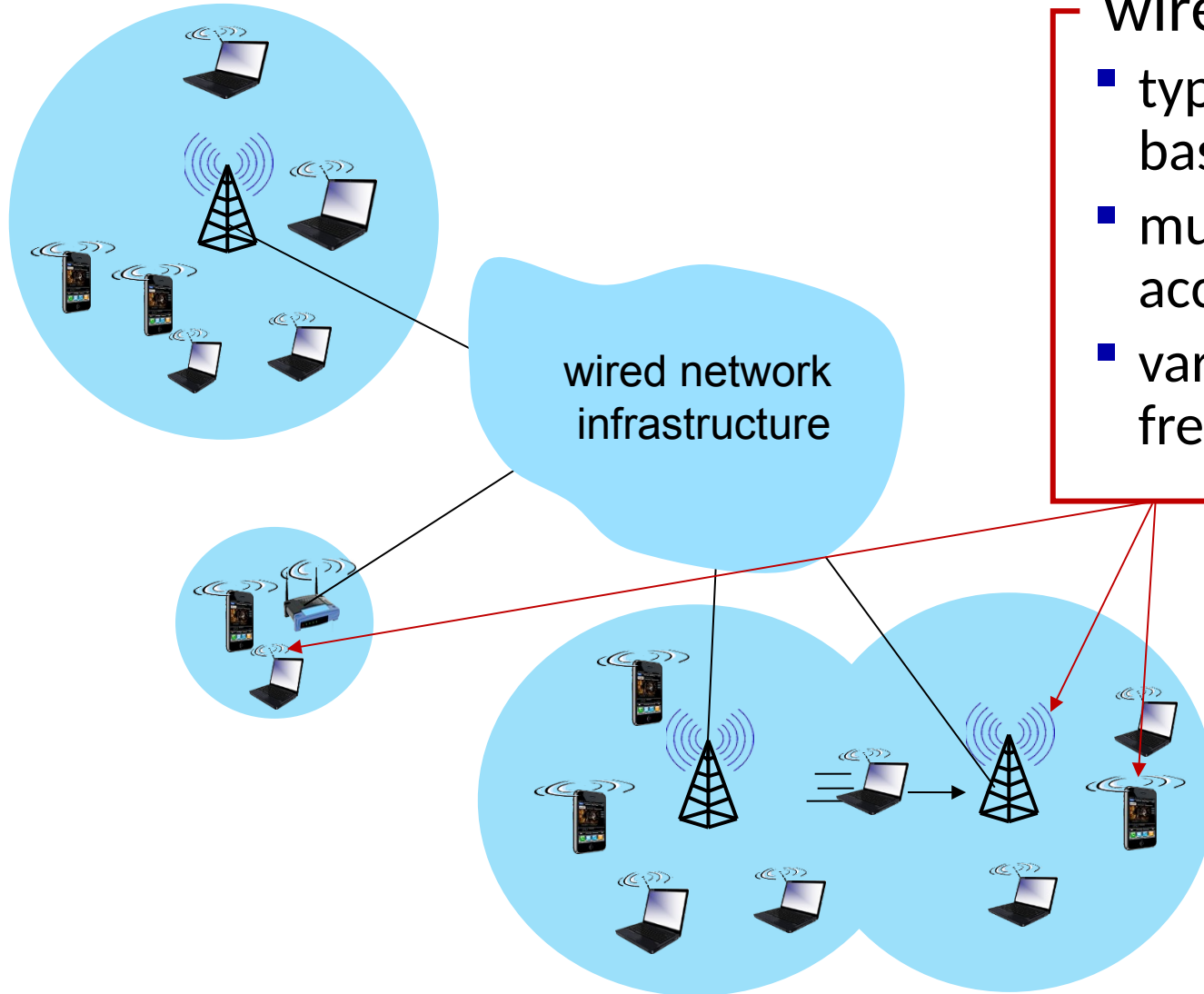
Computer Networking: A Top-Down Approach

8th edition

Jim Kurose, Keith Ross

Pearson, 2020

Elements of a wireless network



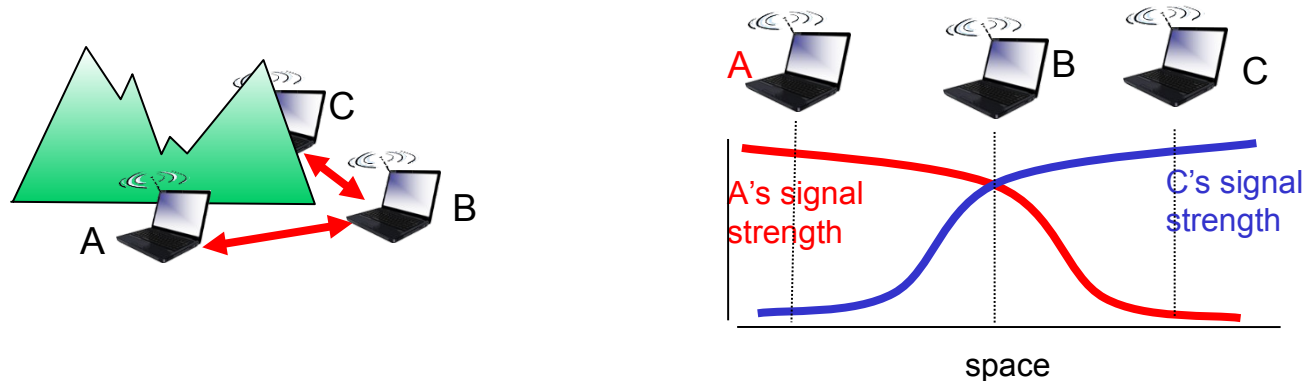
wireless link



- typically used to connect mobile(s) to base station, also used as backbone link
- multiple access protocol coordinates link access
- various transmission rates and distances, frequency bands

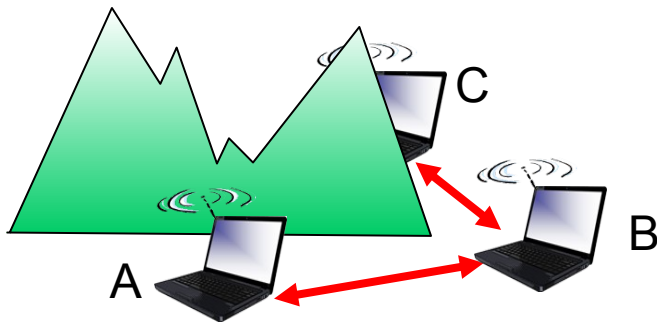
IEEE 802.11: multiple access

- avoid collisions: 2+ nodes transmitting at same time
- 802.11: CSMA - sense before transmitting
 - don't collide with detected ongoing transmission by another node
- 802.11: *no* collision detection!
 - difficult to sense collisions: high transmitting signal, weak received signal due to fading
 - can't sense all collisions in any case: hidden terminal, fading
 - goal: *avoid collisions*: CSMA/CollisionAvoidance



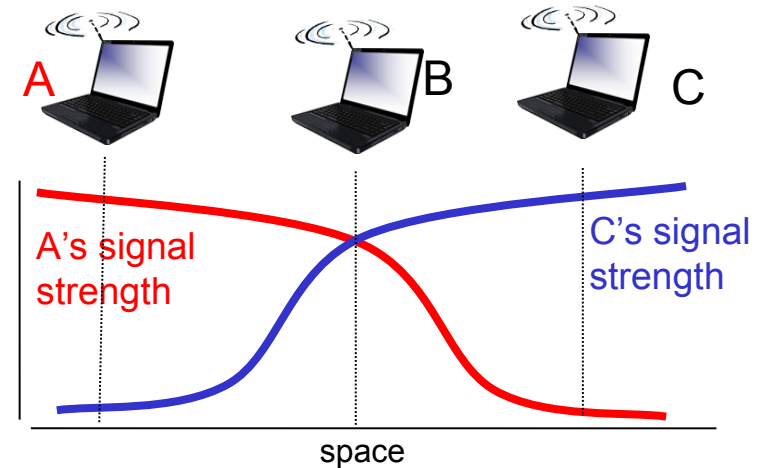
Wireless link characteristics (3)

Multiple wireless senders, receivers create additional problems (beyond multiple access):



Hidden terminal problem

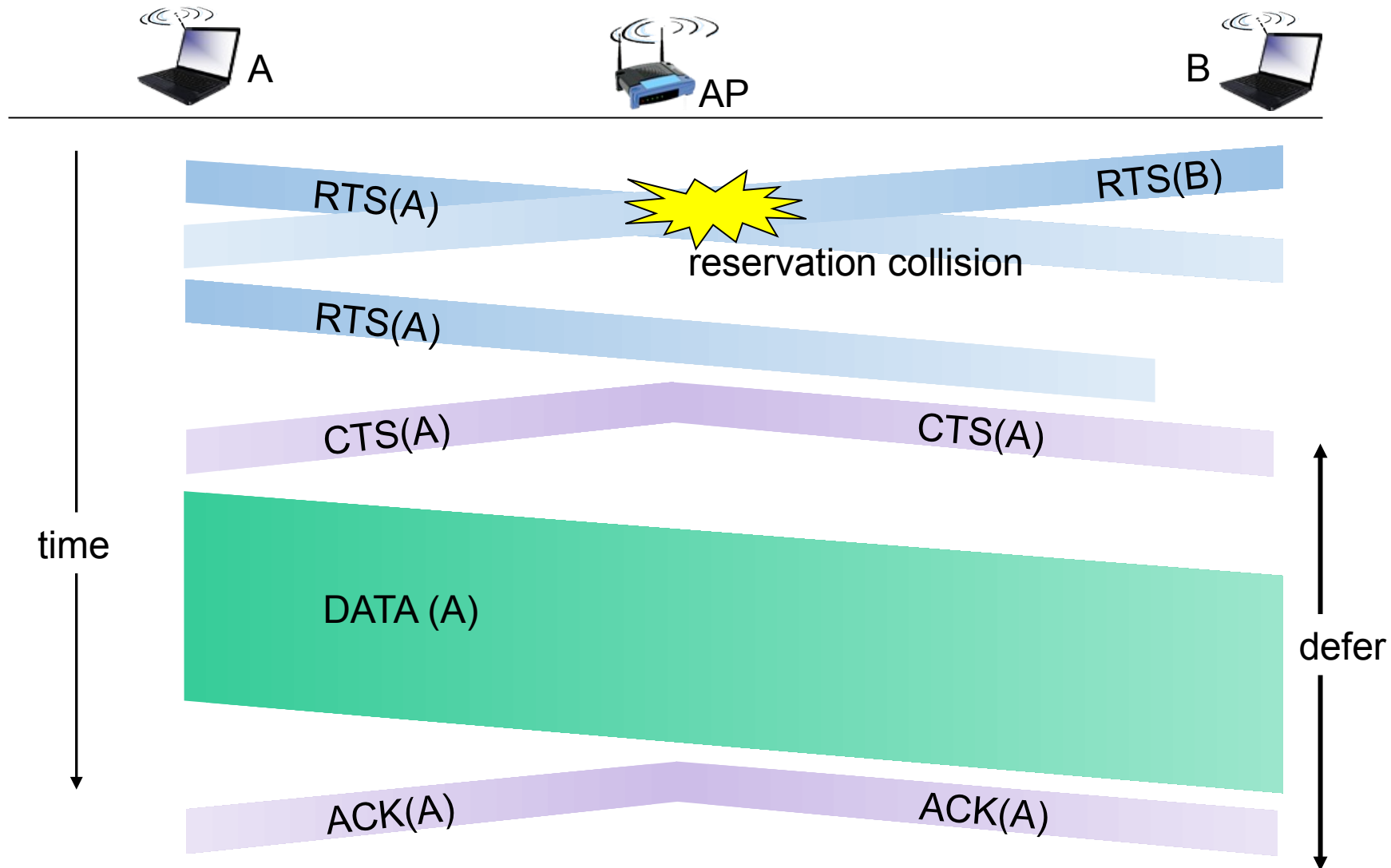
- B, A hear each other
- B, C hear each other
- A, C can not hear each other means A, C unaware of their interference at B



Signal attenuation:

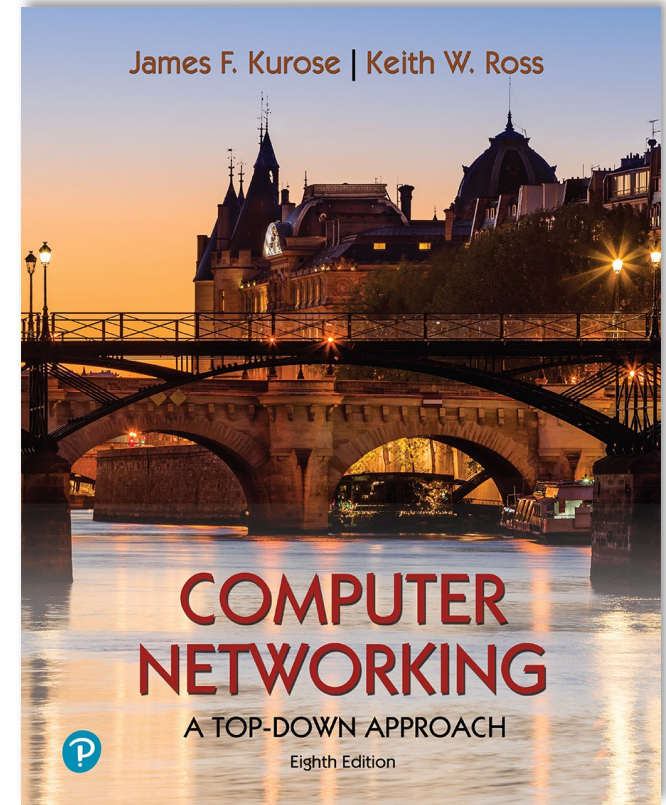
- B, A hear each other
- B, C hear each other
- A, C can not hear each other interfering at B

Collision Avoidance: RTS-CTS exchange



Chapter 8

Security



Computer Networking: A Top-Down Approach

8th edition

Jim Kurose, Keith Ross

Pearson, 2020

What is network security?

confidentiality: only sender, intended receiver should “understand” message contents

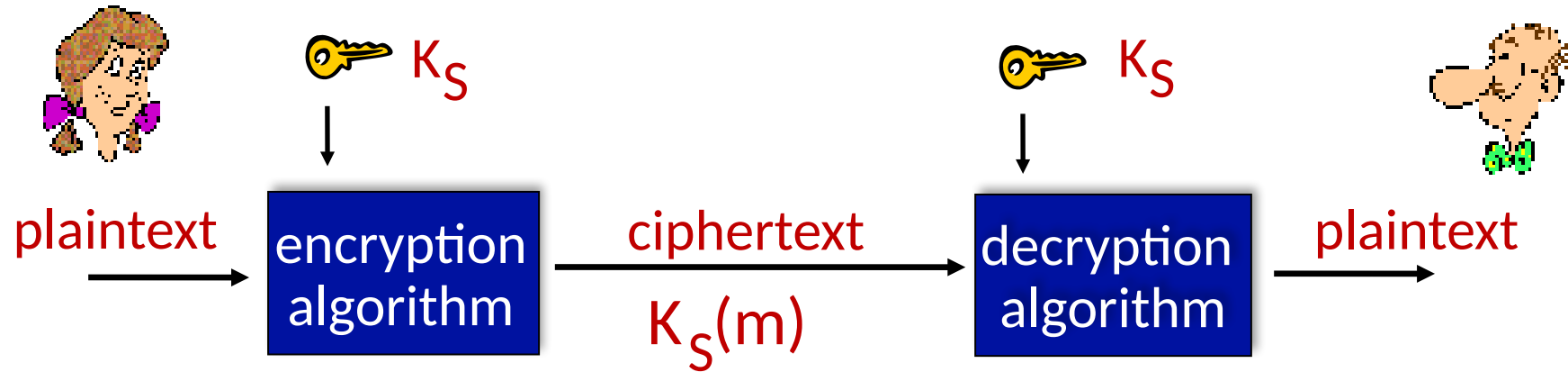
- sender encrypts message
- receiver decrypts message

authentication: sender, receiver want to confirm identity of each other

message integrity: sender, receiver want to ensure message not altered (in transit, or afterwards) without detection

access and availability: services must be accessible and available to users

Symmetric key cryptography

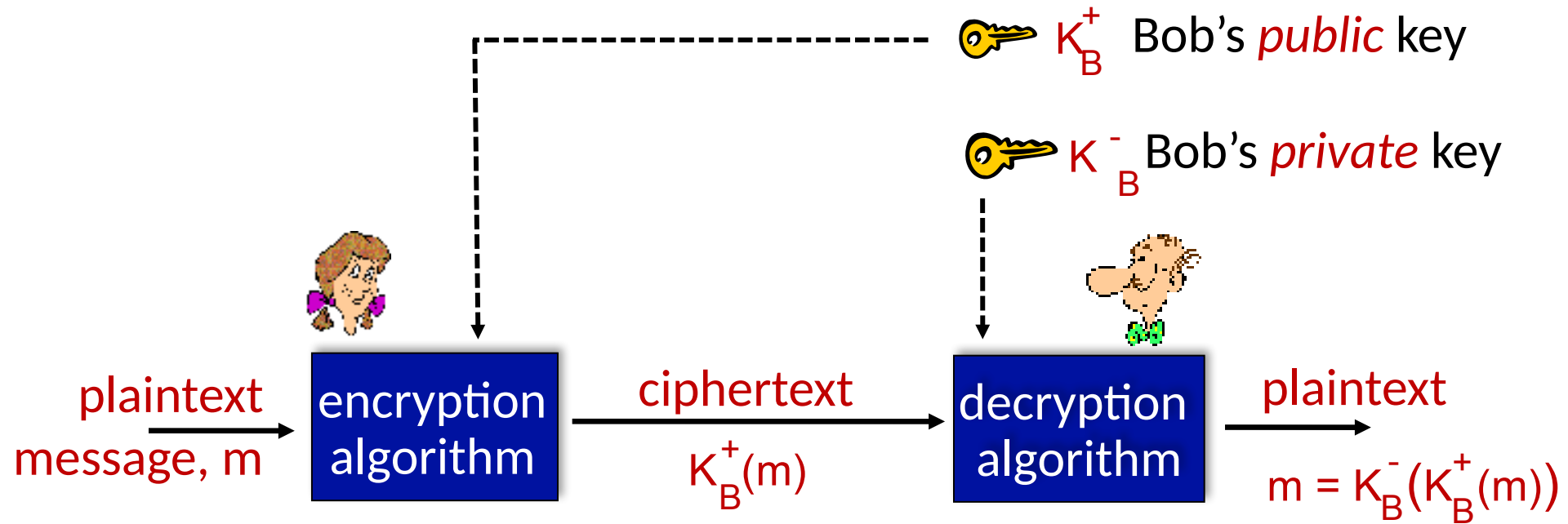


symmetric key crypto: Bob and Alice share same (symmetric) key: K

- e.g., key is knowing substitution pattern in mono alphabetic substitution cipher

Q: how do Bob and Alice agree on key value?

Public Key Cryptography



Wow - public key cryptography revolutionized 2000-year-old (previously only symmetric key) cryptography!

- similar ideas emerged at roughly same time, independently in US and UK (classified)

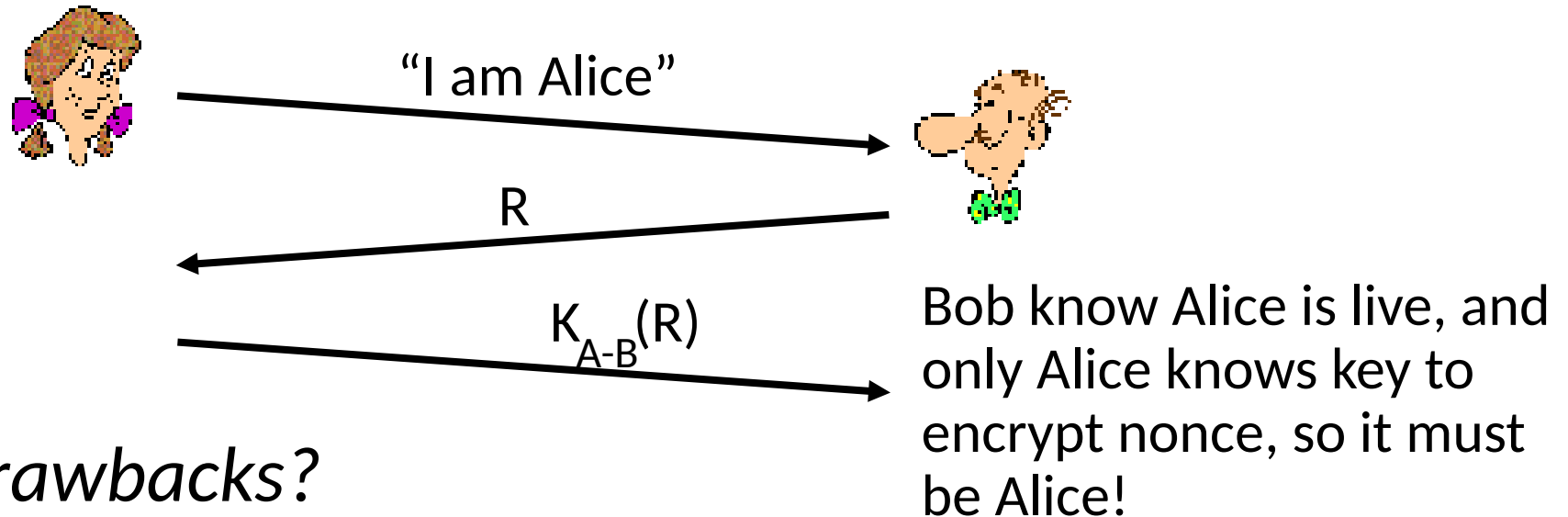
Authentication: a fourth try

Goal: avoid playback attack

nonce: number (R) used only **once-in-a-lifetime**

protocol ap4.0: to prove Alice “live”, Bob sends Alice nonce, R

- Alice must return R, encrypted with shared secret key

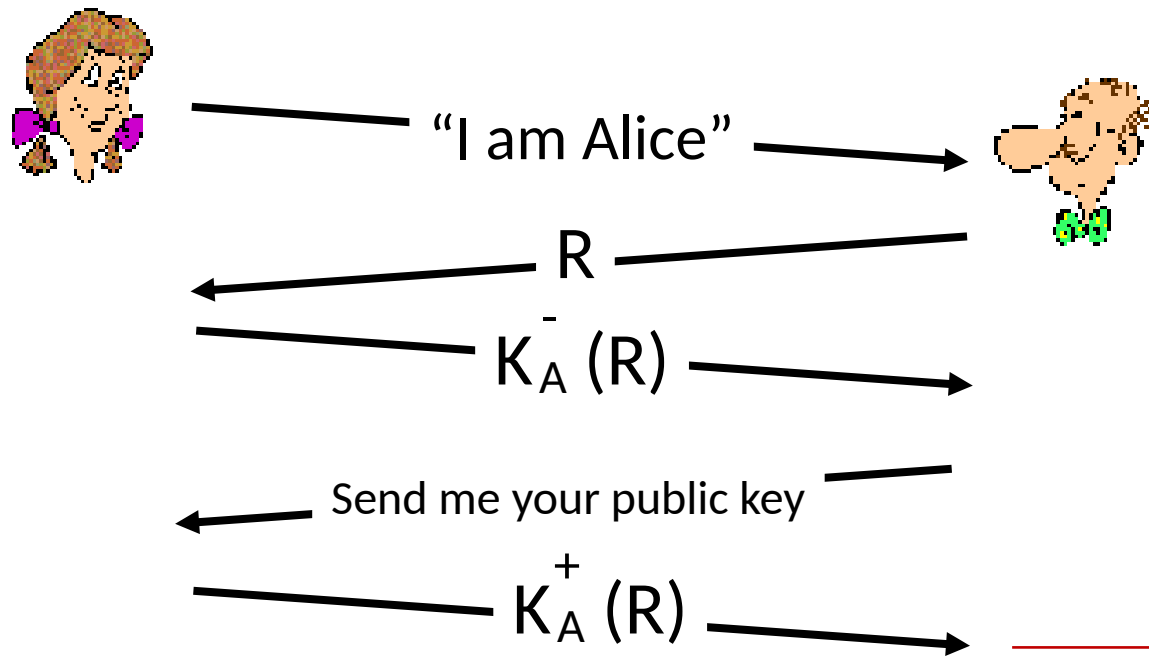


Failures, drawbacks?

Authentication: ap5.0

ap4.0 requires shared symmetric key - can we authenticate using public key techniques?

ap5.0: use nonce, public key cryptography



Bob computes

$$K_A^+ (K_A^-(R)) = R$$

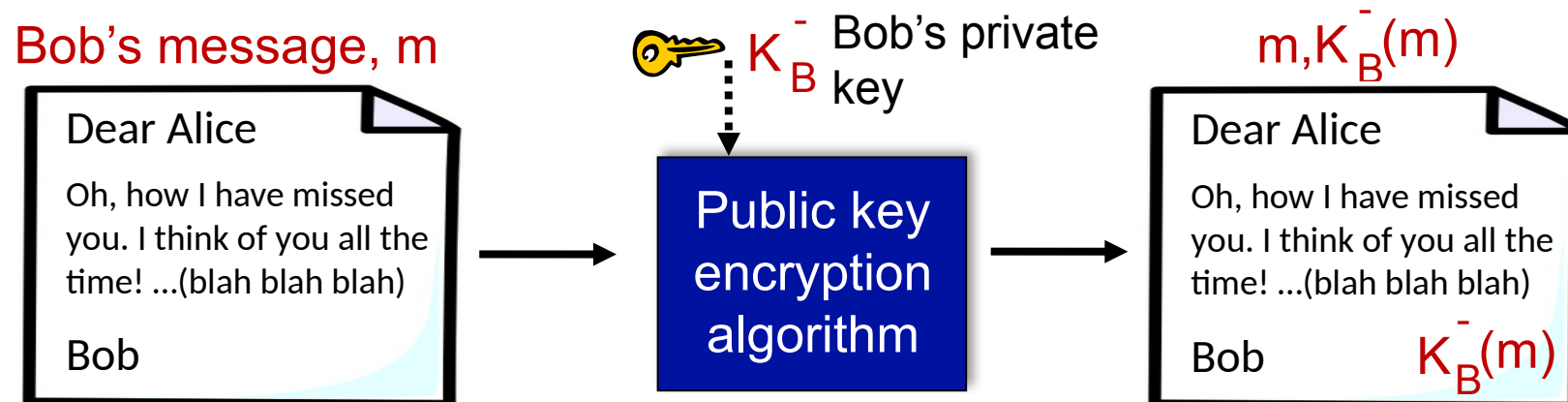
and knows only Alice could have the private key, that encrypted R such that

$$K_A^+ (K_A^-(R)) = R$$

Digital signatures

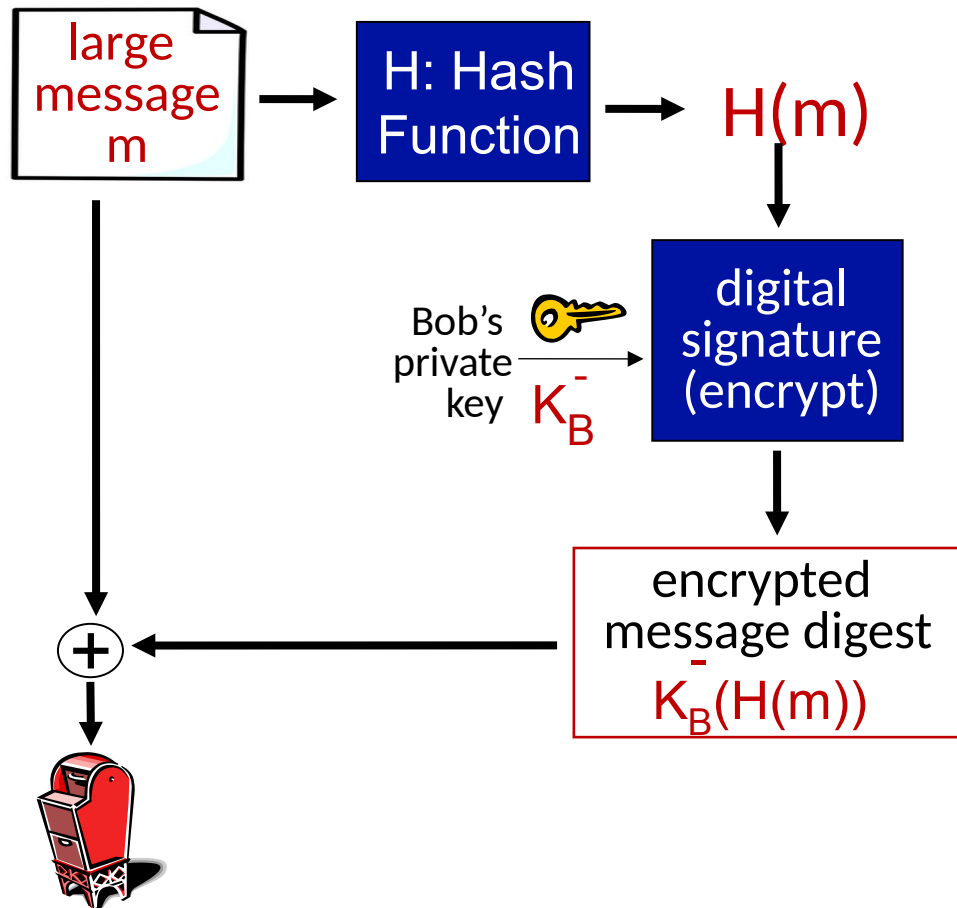
cryptographic technique analogous to hand-written signatures:

- sender (Bob) digitally signs document: he is document owner/creator.
- *verifiable, nonforgeable*: recipient (Alice) can prove to someone that Bob, and no one else (including Alice), must have signed document
- simple digital signature for message m :
 - Bob signs m by encrypting with his private key K_B , creating “signed” message, $K_B^-(m)$

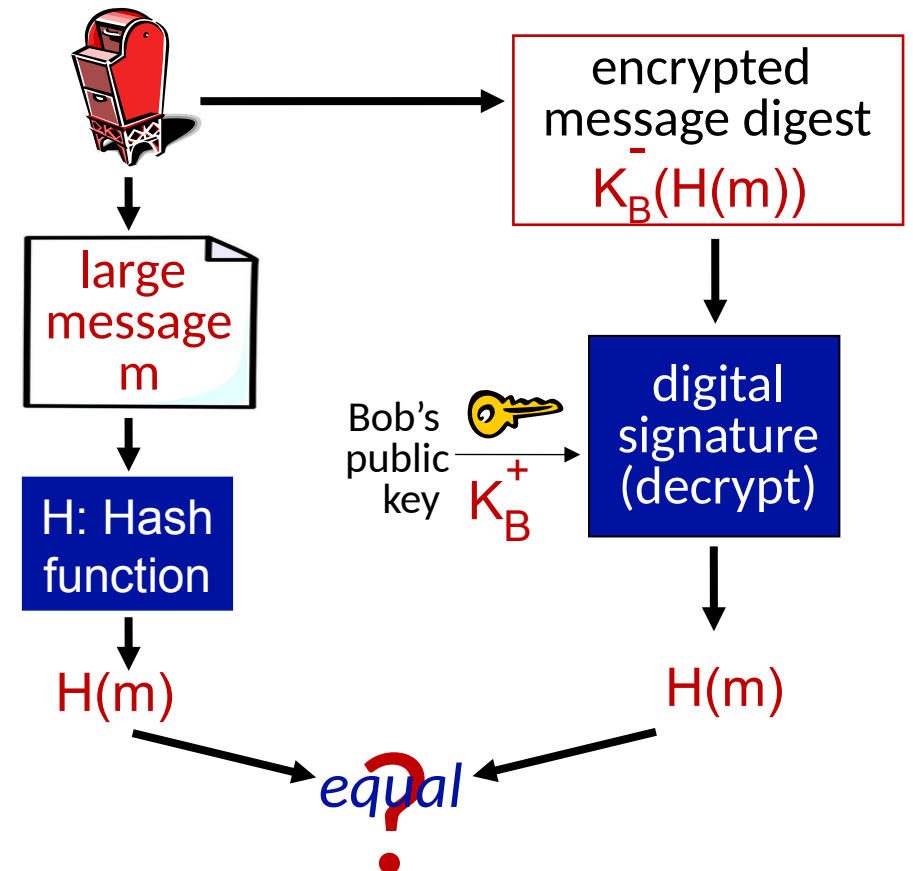


Digital signature = signed message digest

Bob sends digitally signed message:

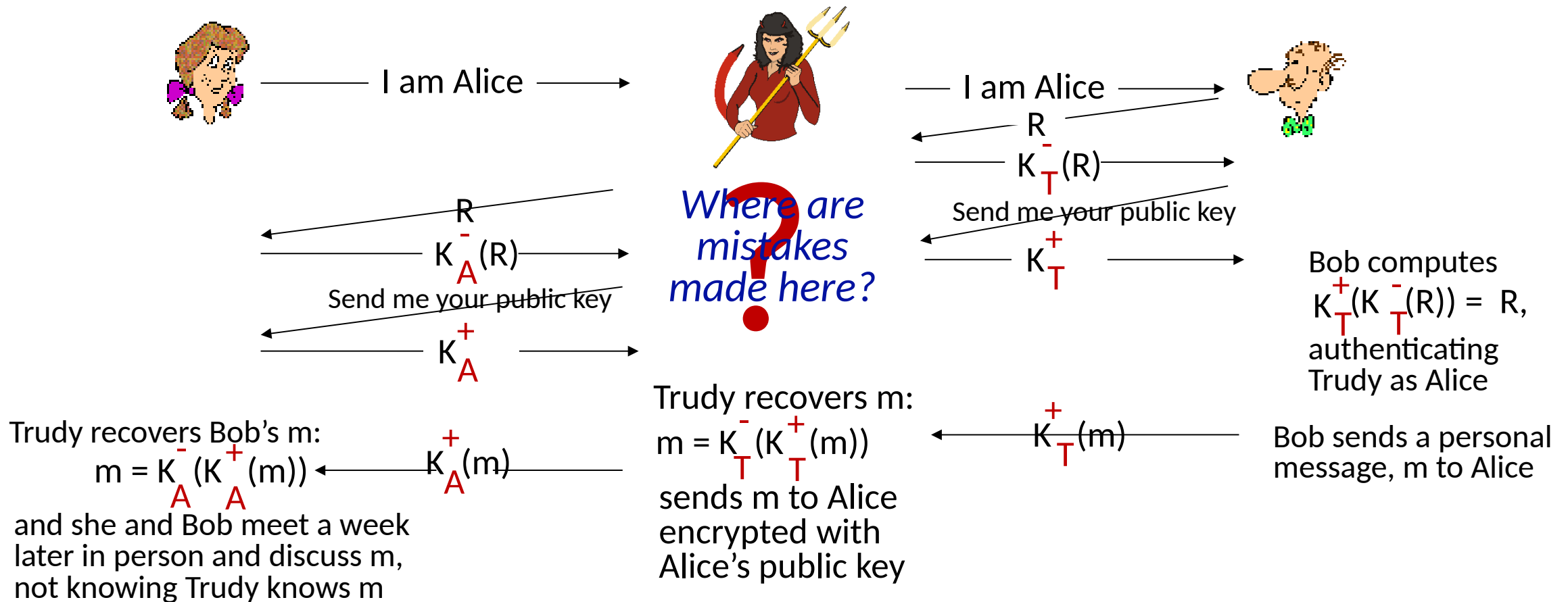


Alice verifies signature, integrity of digitally signed message:



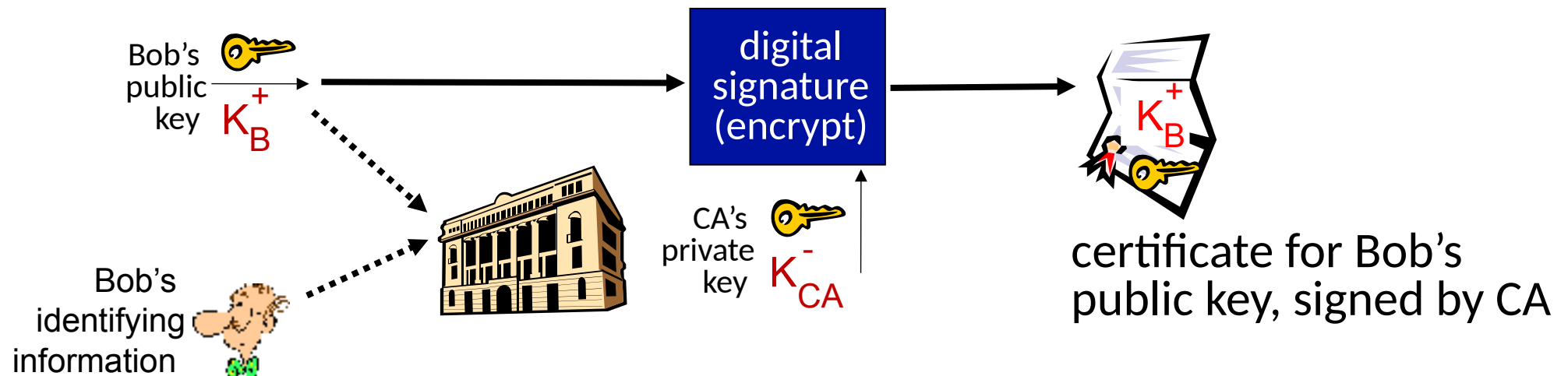
Authentication: ap5.0 – let's fix it!!

Recall the problem: Trudy poses as Alice (to Bob) and as Bob (to Alice)



Public key Certification Authorities (CA)

- **certification authority (CA):** binds public key to particular entity, E
- entity (person, website, router) registers its public key with CE provides “proof of identity” to CA
 - CA creates certificate binding identity E to E’s public key
 - certificate containing E’s public key digitally signed by CA: CA says “this is E’s public key”



Public key Certification Authorities (CA)

- when Alice wants Bob's public key:
 - gets Bob's certificate (Bob or elsewhere)
 - apply CA's public key to Bob's certificate, get Bob's public key

