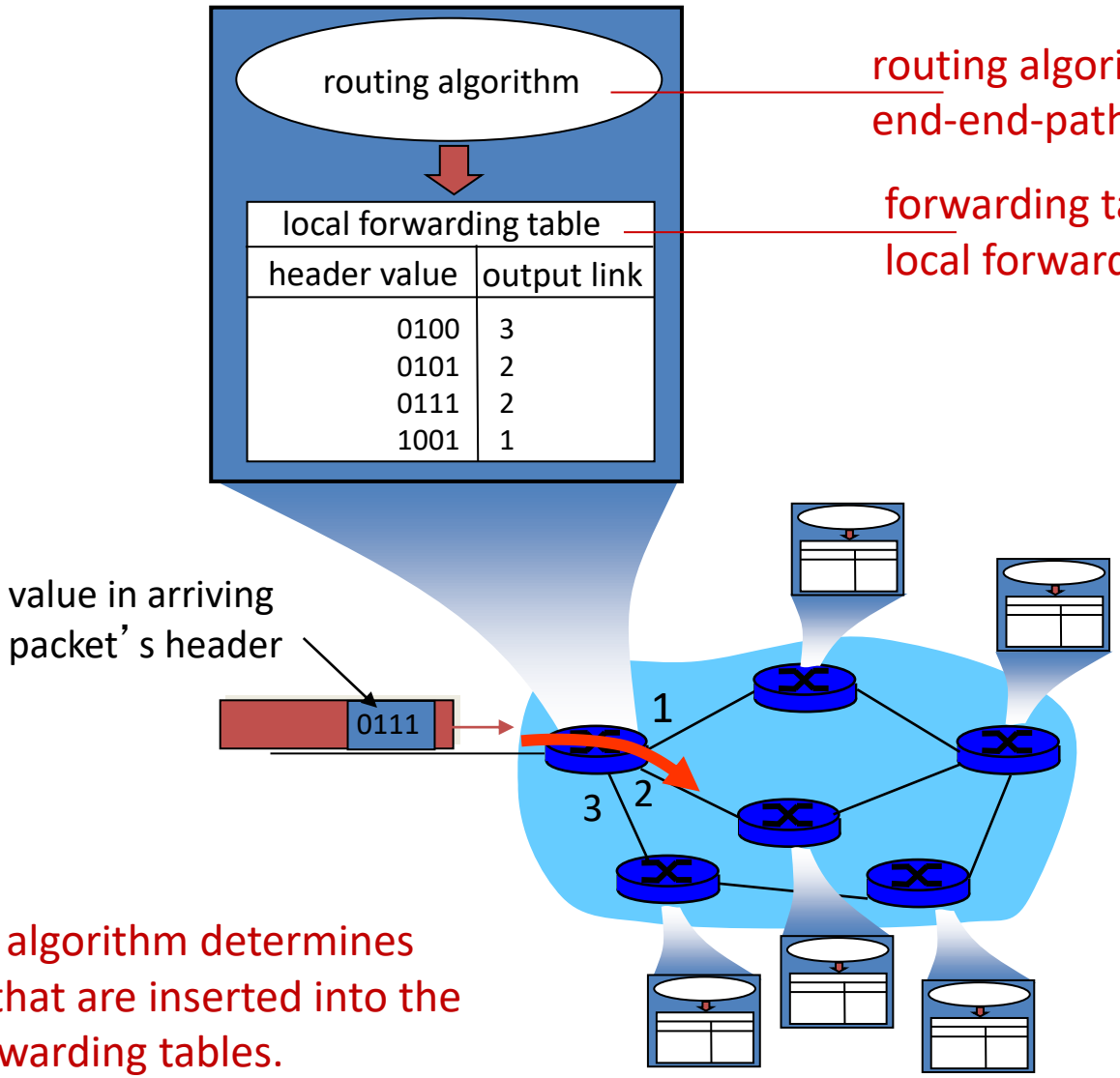
Interplay between routing and forwarding



the routing algorithm determines the values that are inserted into the routers' forwarding tables.

Connection, connection-less service

- *Datagram (数据报)* network provides network-layer *connectionless* service
- *virtual-circuit (虚电路)* network provides network-layer *connection* service
- analogous to TCP/UDP connection-oriented / connectionless transport-layer services, but:
  - *service:* host-to-host (not process to process)
  - *no choice:* network provides one or the other, but not both
  - *implementation:* in network core, fundamentally different

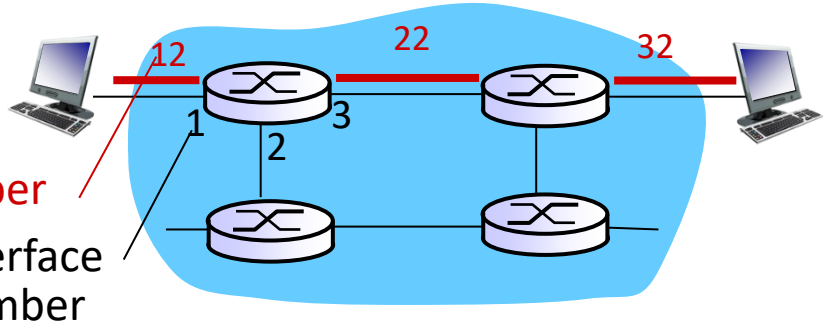
Virtual circuits

"source-to-dest path behaves much like telephone circuit"

- performance-wise
- network actions along source-to-dest path

- call setup, teardown for each call *before* data can flow
- each packet carries VC identifier (not destination host address)
- *every* router on source-dest path maintains "state" for each passing connection
- link, router resources (bandwidth, buffers) may be *allocated* to VC (dedicated resources = predictable service)

VC forwarding table



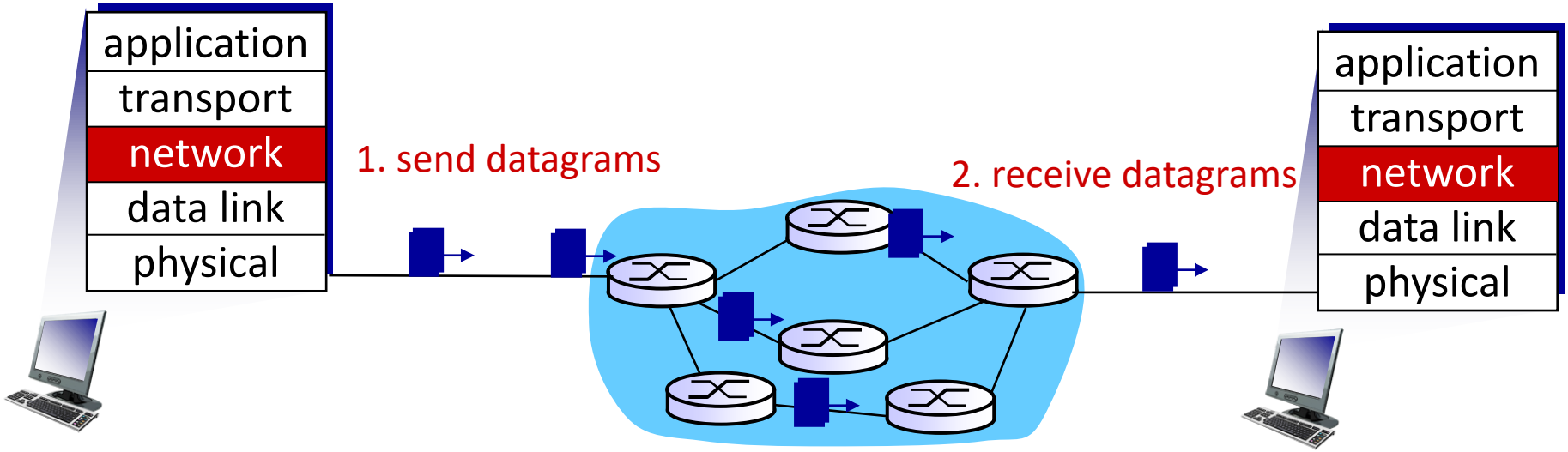
forwarding table in northwest router:

Incoming interface	Incoming VC #	Outgoing interface	Outgoing VC #
1	12	3	22
2	63	1	18
3	7	2	17
1	97	3	87
…	…	…	…

VC routers maintain connection state information!

Datagram networks

- no call setup at network layer
- routers: no state about end-to-end connections
  - no network-level concept of “connection”
- packets forwarded using destination host address



Longest prefix matching

longest prefix matching（最长前缀匹配）

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?

Datagram or VC network: why?

Internet (datagram)

- data exchange among computers
  - "elastic" service, no strict timing req.
- many link types
  - different characteristics
  - uniform service difficult
- "smart" end systems (computers)
  - can adapt, perform control, error recovery
  - ***simple inside network, complexity at "edge"***

ATM (VC)

- evolved from telephony
- human conversation:
  - strict timing, reliability requirements
  - need for guaranteed service
- "dumb" end systems
  - telephones
  - ***complexity inside network***

Router architecture overview

two key router functions:

- run routing algorithms/protocol (RIP, OSPF, BGP)
- *forwarding* datagrams from incoming to outgoing link

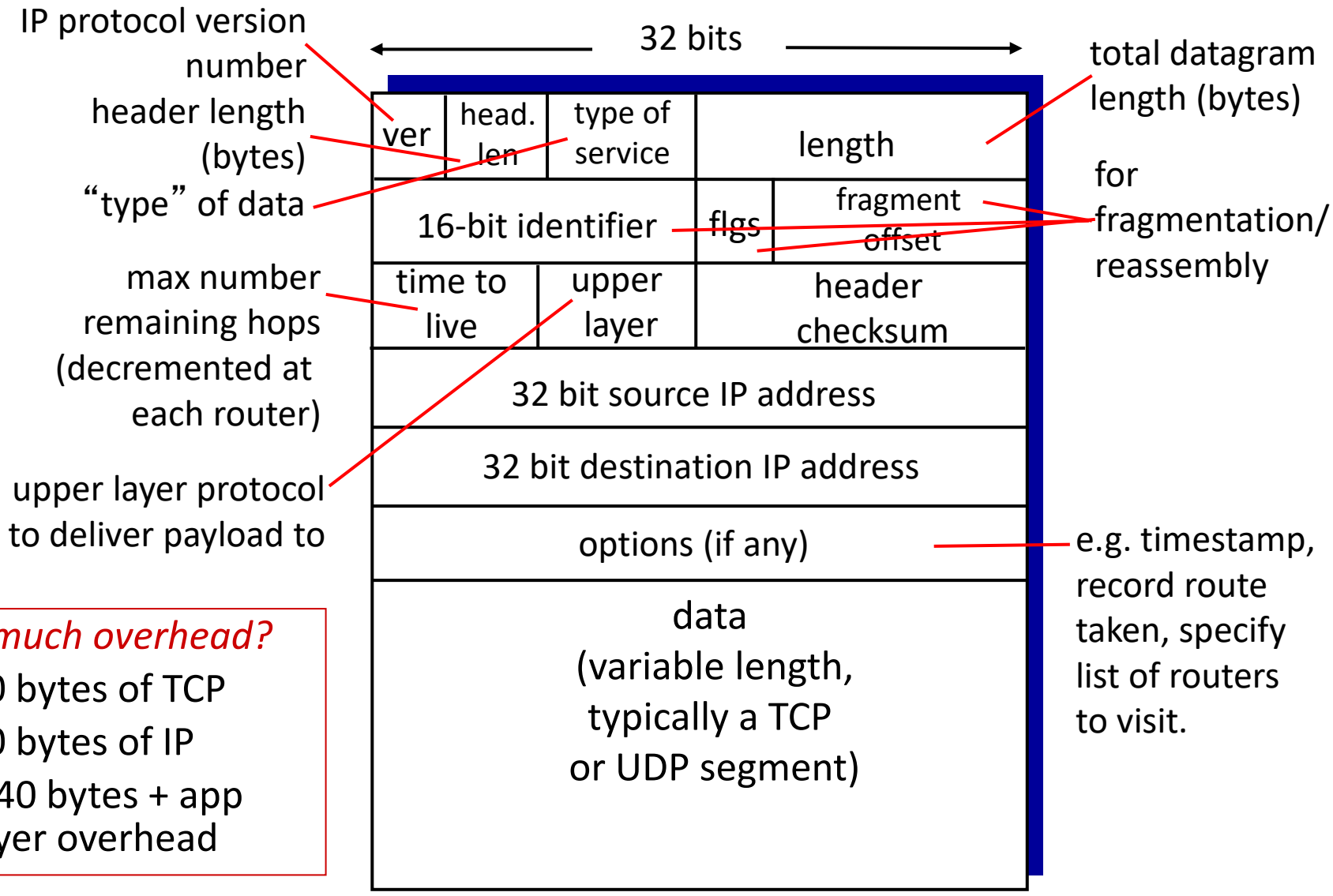
Four components:

- Input ports: physical-layer function; link-layer function; look up the forwarding table; forward the control packet to the routing processor
- Switching fabric: connects input ports to output ports
- Output ports: buffer; link-layer and physical-layer functions
- Routing processor: executes routing protocols

Router architecture overview

- A router's input ports, output ports, and switching fabric collectively referred to as the router forwarding plane
  - Almost always implemented in hardware
  - 10Gbps port ~ 51.2 ns to process an IP datagram
- Router' control functions—executing the routing protocols, responding to attached links that go up or down, and performing management functions, are referred to as the router control plane
  - Software running on CPU
  - Much slower

IP datagram format

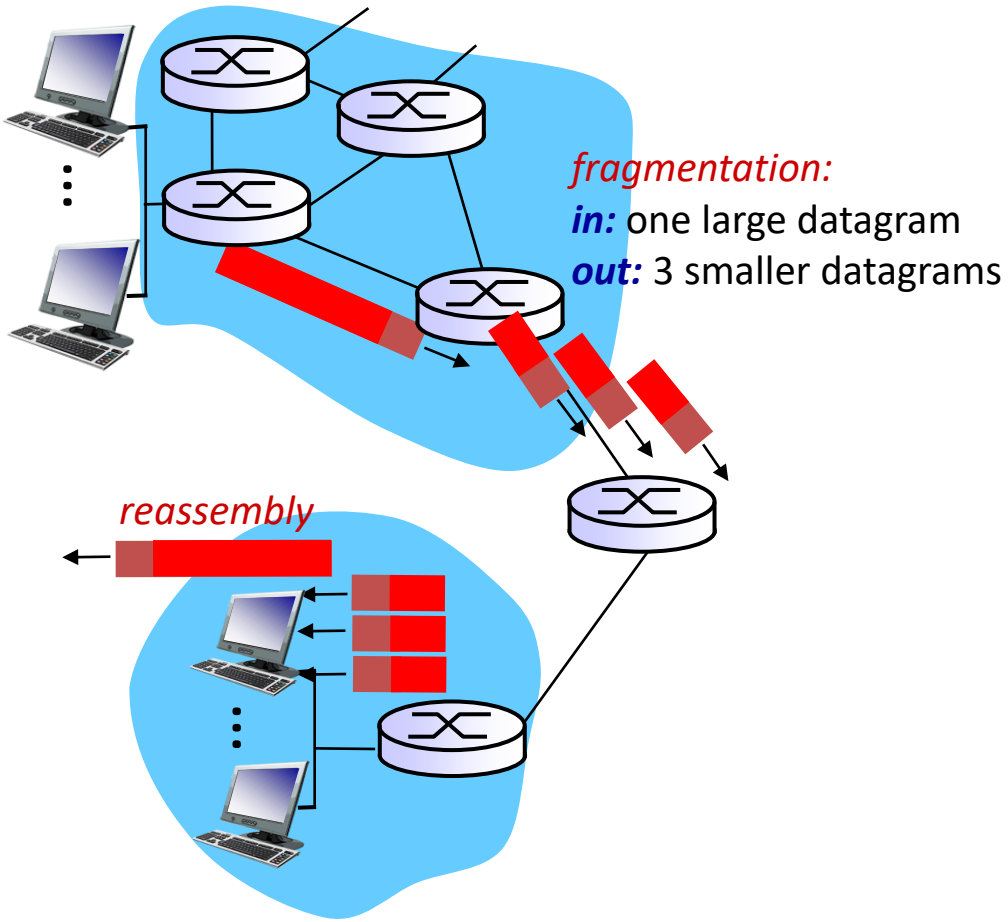


how much overhead?

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

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



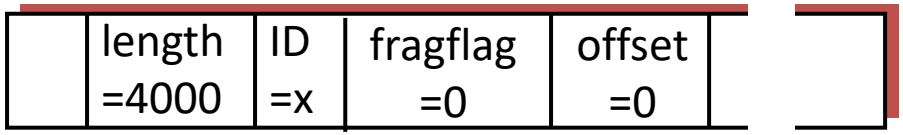
IP fragmentation, reassembly

- Identification: stamped by the sending host
  - Increase for every datagram sent by the host
  - Fragments have a same identification as the original datagram
- Multiple fragments for one original datagram, the last fragment has a flag bit set to 0, whereas all the other fragments have this flag bit set to 1.
- The offset field is used to specify where the fragment fits within the original IP datagram.

IP fragmentation, reassembly

example:

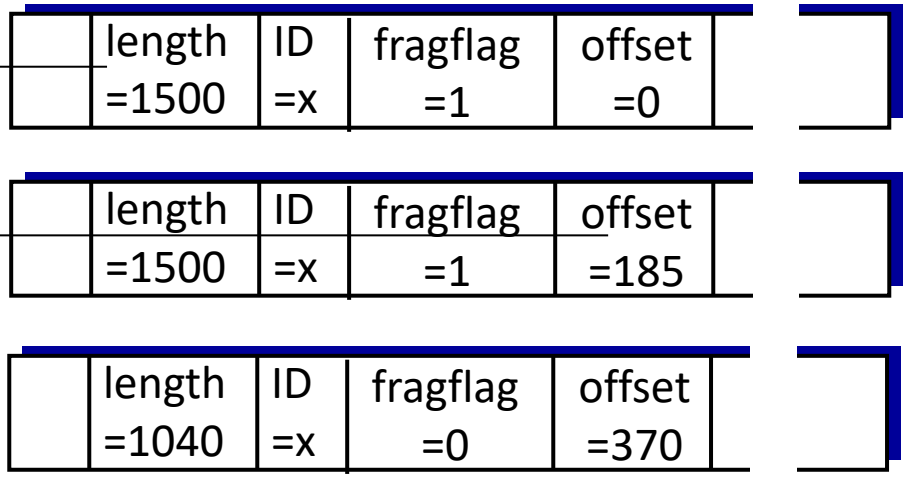
- 4000 byte datagram
- MTU = 1500 bytes



one large datagram becomes several smaller datagrams

1480 bytes in data field

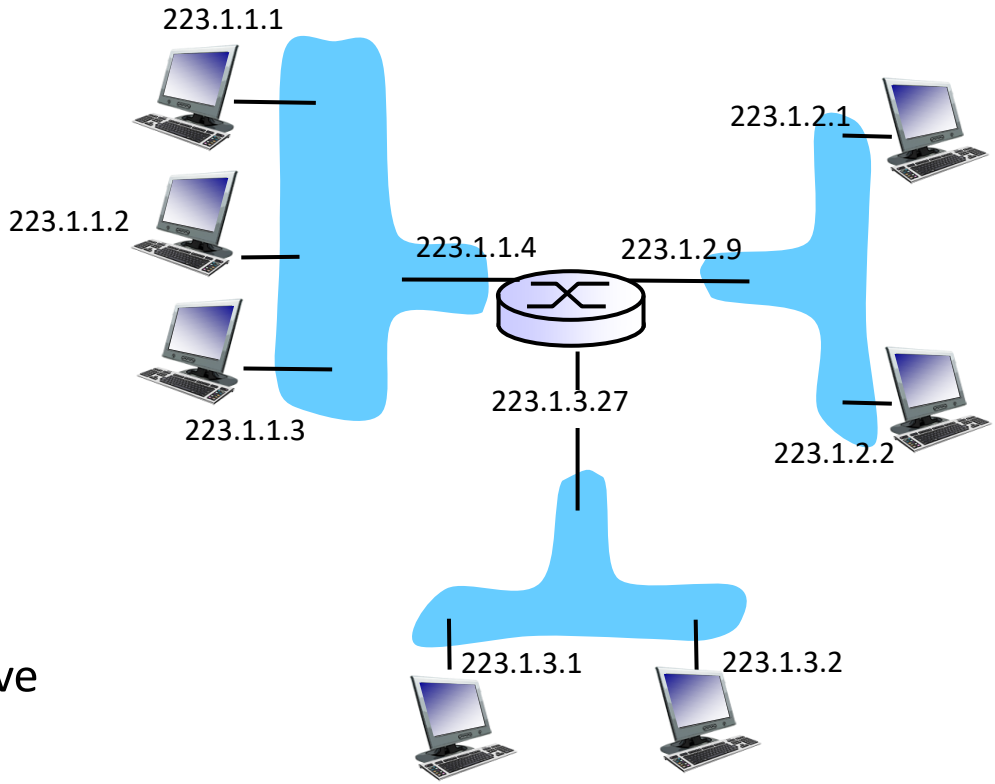
offset = 1480/8 meaning the data should be inserted beginning at byte 1,480.



- Cost of fragmentation:
  - Complication at routers and hosts
  - DoS attack

IP addressing: introduction

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

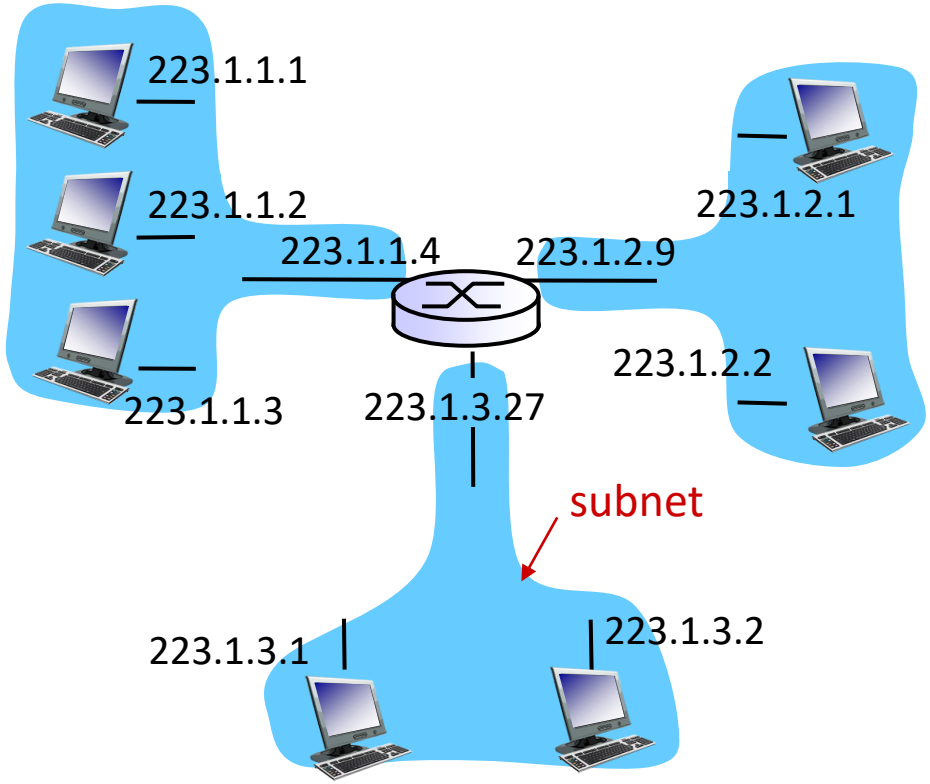


223.1.1.1 = 11011111 00000001 00000001 00000001

223 1 1 1

Subnets

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



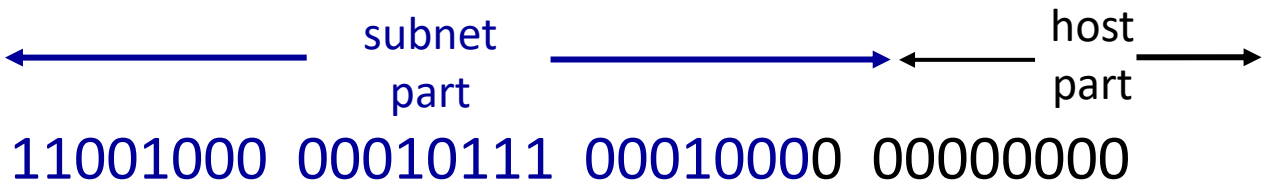
network consisting of 3 subnets

逻辑上、物理上都有要求

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
- The x most significant bits of an address is called a *prefix*

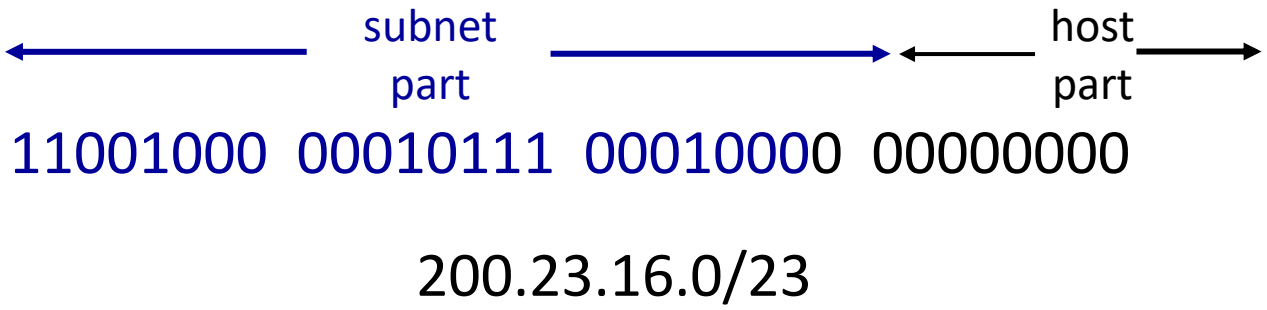


200.23.16.0/23

200.23.16.0 --- 200.23.17.255

IP addressing: CIDR

- The lower-order bits may (or may not) have an additional subnetting structure
  - 200.23.16.0/23 may contain two subnets: 200.23.16.0/24 and 200.23.17.0/24.
- Broadcast address: 255.255.255.255
  - Delivered to all hosts in same subnet



DHCP: Dynamic Host Configuration Protocol

goal: allow host to *dynamically* obtain its IP address from network server when it joins network

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

DHCP overview:

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

DHCP: more than IP addresses

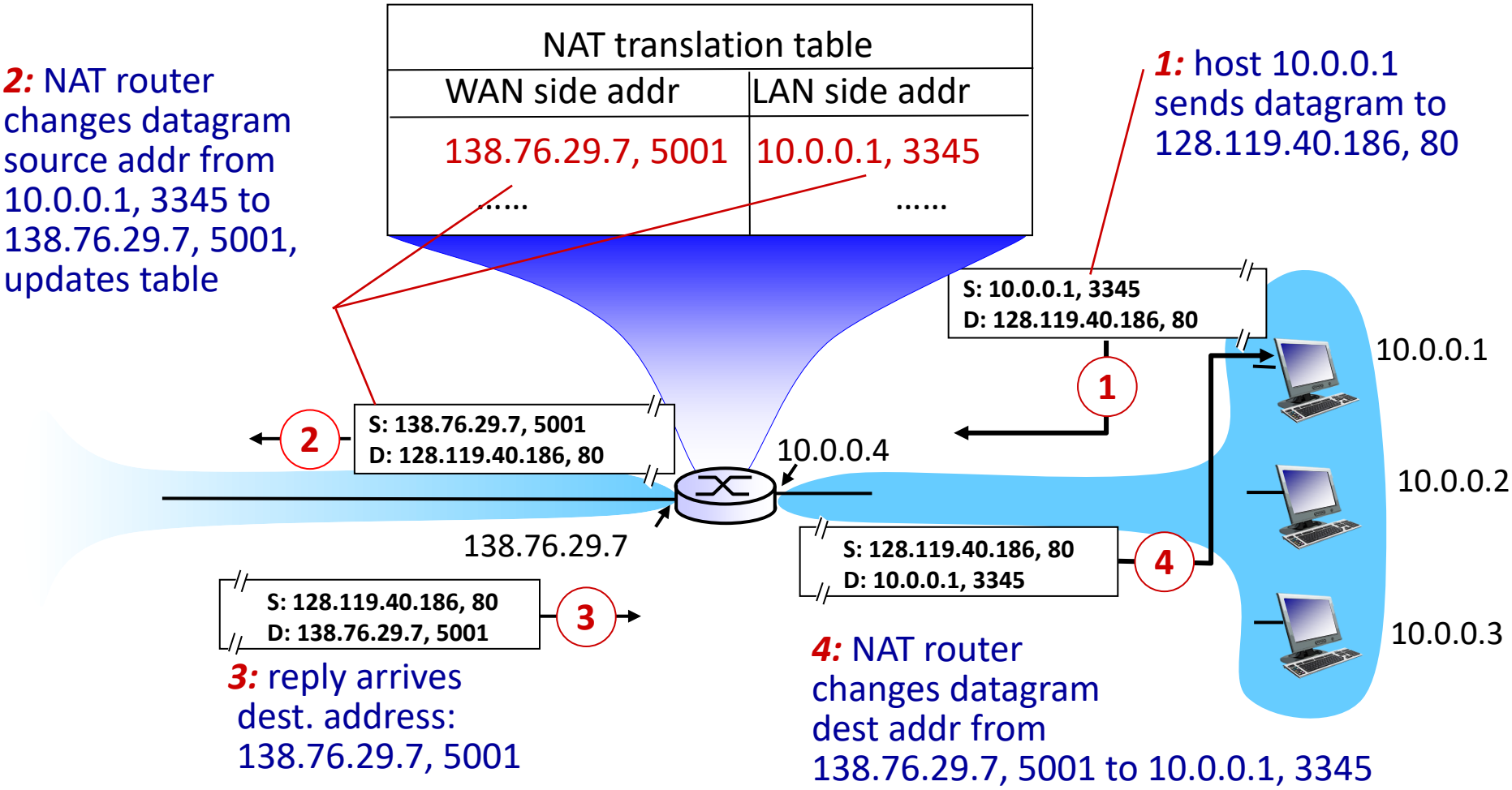
DHCP returns:

- IP address
- address of first-hop router for client (gateway address)
- name and IP address of DNS sever
- network mask (indicating network versus host portion of address)

Mobility issue:

- Obtain new address when moving to a new subnet
- Can not maintain TCP connections
- Mobile IP

NAT: network address translation

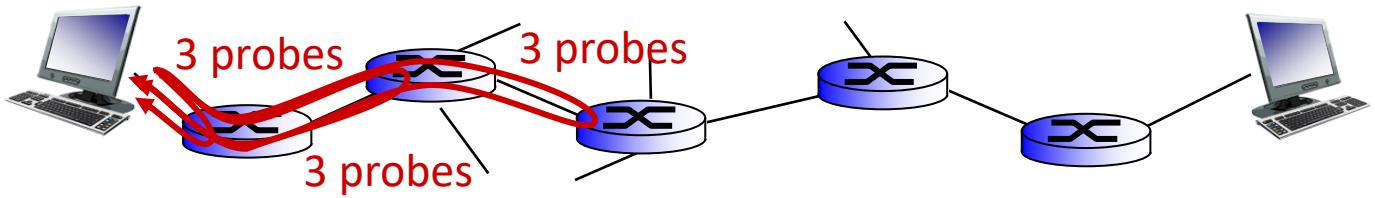


Traceroute and ICMP

- source sends series of UDP segments to dest
  - first set has TTL =1
  - second set has TTL=2, etc.
  - unlikely port number
- when *n*th set of datagrams arrives to *n*th router:
  - router discards datagrams
  - and sends source ICMP messages (type 11, code 0)
  - ICMP messages includes name of router & IP address
- when ICMP messages arrives, source records RTTs

stopping criteria:

- UDP segment eventually arrives at destination host
- destination returns ICMP "port unreachable" message (type 3, code 3)
- source stops

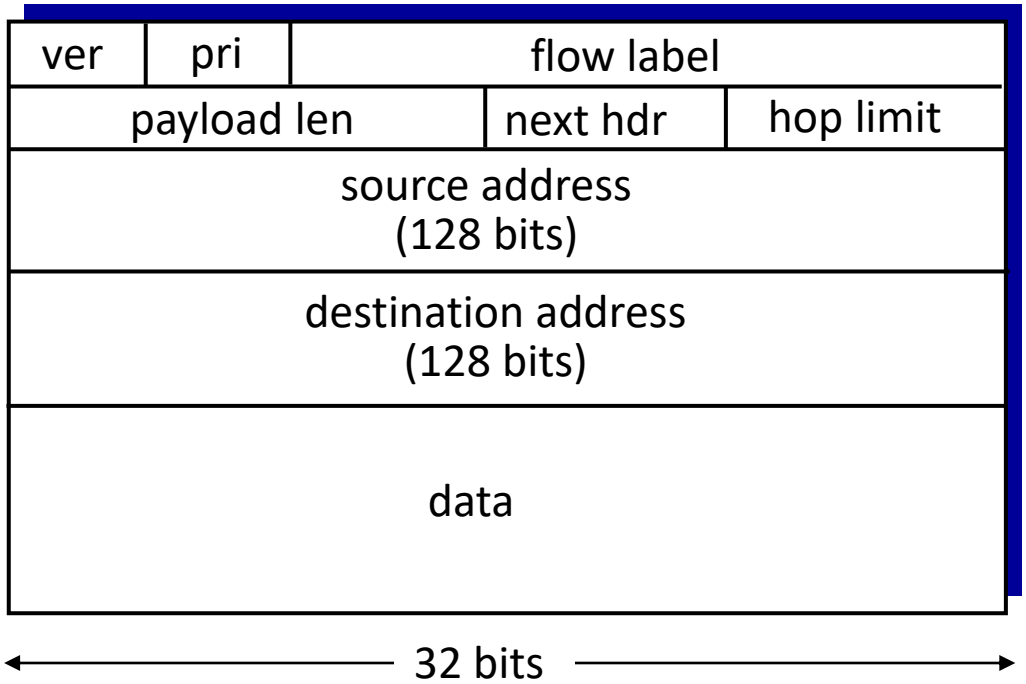


IPv6 datagram format

priority: identify priority among datagrams in flow

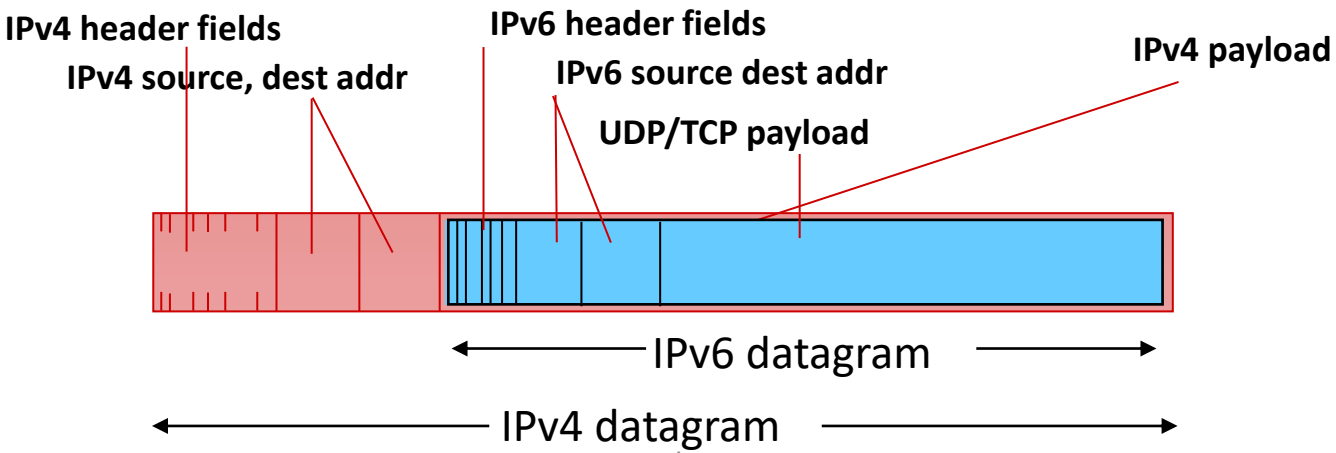
flow Label: identify datagrams in same "flow."
(concept of"flow" not well defined).

next header: identify upper layer protocol for data

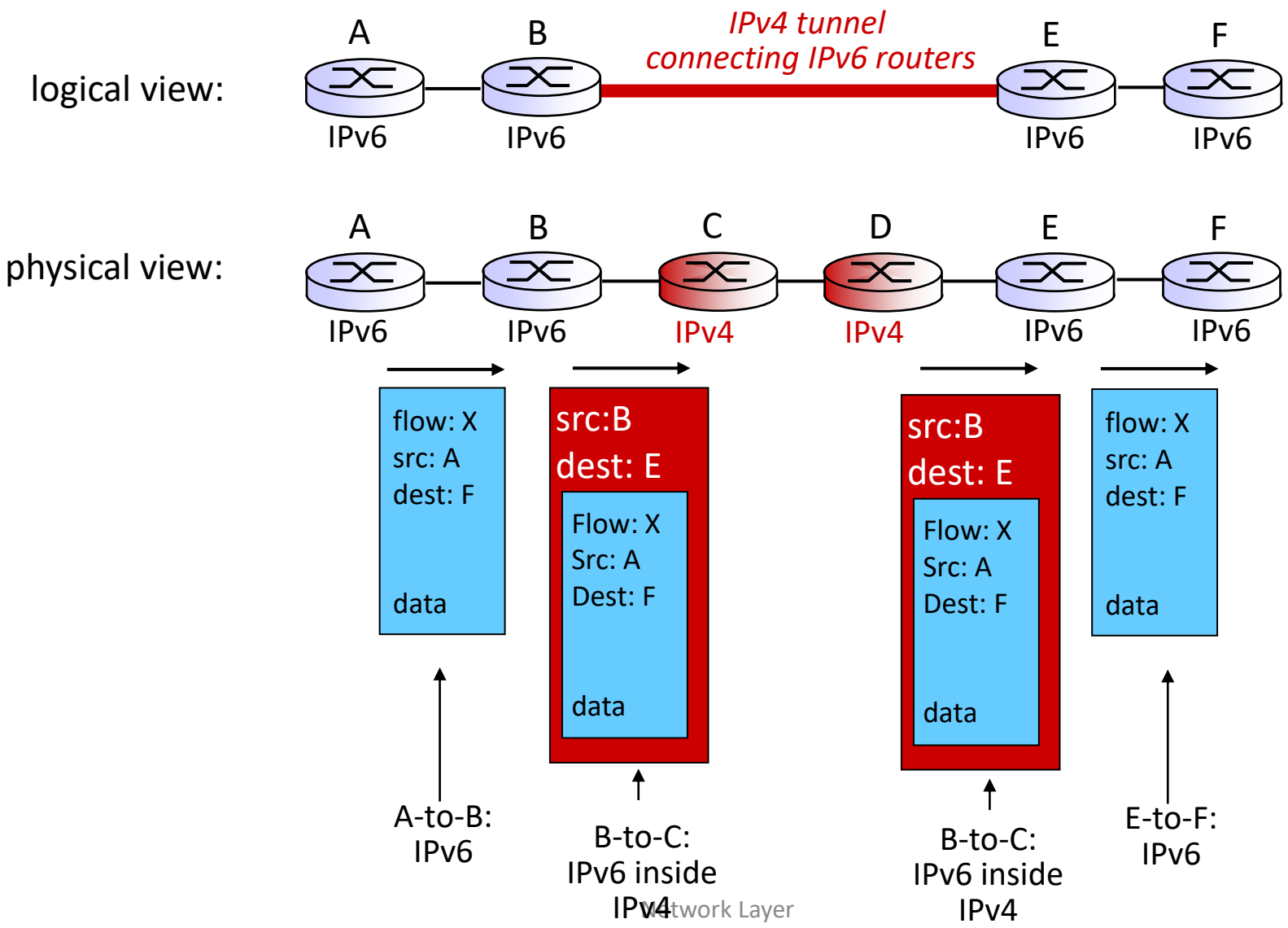


Transition from IPv4 to IPv6

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

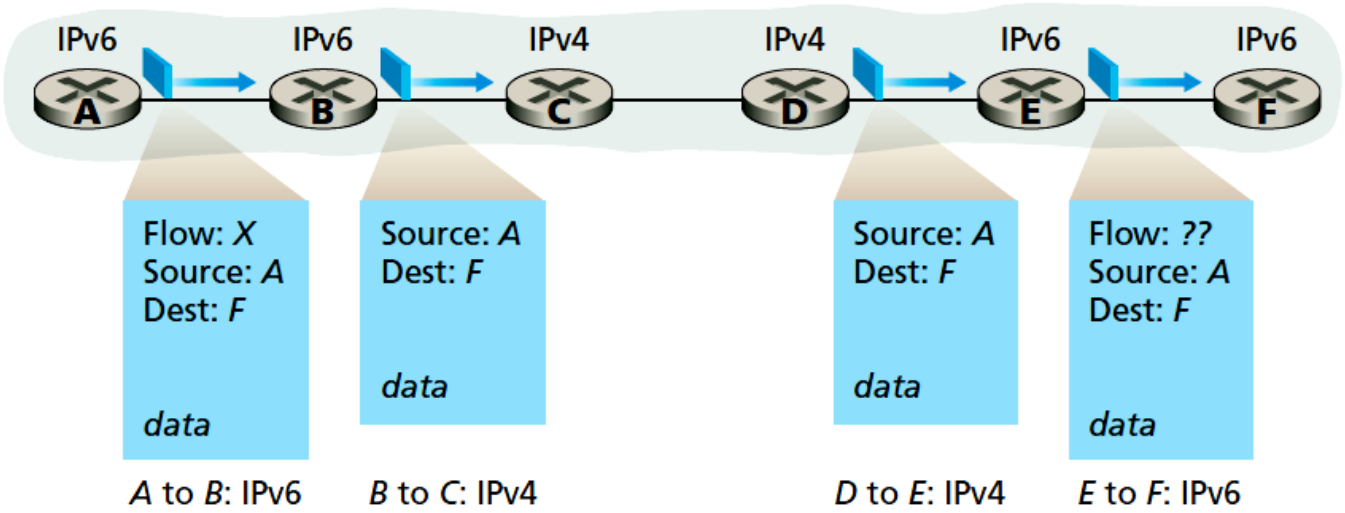


Tunneling


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

Dual-stack

- Router has the ability to send and receive both IPv4 and IPv6 datagrams
- IPv4 to IPv4 transition loses v6 field values



even though E and F can exchange IPv6 datagrams, the arriving IPv4 datagrams at E from D do not contain all of the fields that were in the original IPv6 datagram sent from A.

Routing algorithm classification

Q: global or decentralized information?

global:

- all routers have complete topology, link cost info
- "link state（链路状态）" algorithms

decentralized:

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

Q: static or dynamic?

static:

- routes change slowly over time

dynamic:

- routes change more quickly
  - periodic update
  - in response to link cost changes

Q: load sensitive or insensitive:

- Today's Internet routing is load insensitive

A Link-State Routing Algorithm

Dijkstra's algorithm

- net topology, link costs known to all nodes
  - accomplished via "link state broadcast"
  - all nodes have same info
- computes least cost paths from one node ("source") to all other nodes
  - gives *forwarding table* for that node
- iterative: after k iterations, know least cost path to k destinations

notation:

- c(x,y): link cost from node x to y; = ∞ if not direct neighbors
- D(v): current value of cost of path from source to dest. v
- p(v): predecessor node along path from source to v
- N': set of nodes whose least cost path definitively known

Dijsktra's Algorithm

Algorithm executed on node u

```
1  Initialization:
2    N' = {u}
3    for all nodes v
4      if v adjacent to u
5          then D(v) = c(u,v)
6      else D(v) = ∞
7
8  Loop
9     find w not in N' such that D(w) is a minimum
10    add w to N'
11    update D(v) for all v adjacent to w and not in N' :
12       D(v) = min( D(v), D(w) + c(w,v) )
13    /* new cost to v is either old cost to v or known
14     shortest path cost to w plus cost from w to v */
15 until all nodes in N'
```

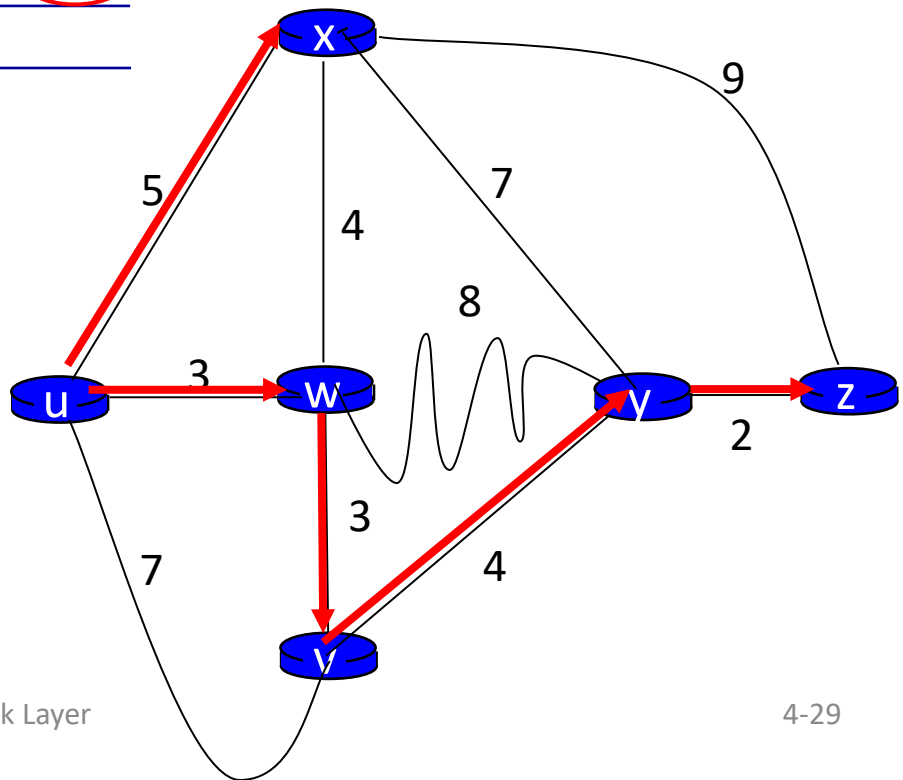
Dijkstra's algorithm: example

Step	N'	D(v) p(v)	D(w) p(w)	D(x) p(x)	D(y) p(y)	D(z) p(z)
0	u	7,u	3,u	5,u	∞	∞
1	uw	6,w		5,u	11,w	∞
2	uwx	6,w			11,w	14,x
3	uwxv				10,v	14,x
4	uwxvy					12,y
5	uwxvyz					

$$e.g.,\ D(v) = \min(D(v), D(w) + c(w,v))$$
$$= \min\{7, 3+3\} = 6$$

notes:

- construct shortest path tree by tracing predecessor nodes
- ties can exist (can be broken arbitrarily)



Dijkstra's algorithm, discussion

algorithm complexity: n nodes

- each iteration: need to check all nodes, w, not in N'
- n(n+1)/2 comparisons: $O(n^2)$
- more efficient implementations possible: O(nlogn)

Distance vector algorithm

Bellman-Ford equation (dynamic programming)

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

Distance vector algorithm

- $D_x(y)$ = estimate of least cost from x to y
  - x maintains distance vector $\mathbf{D}_x = [D_x(y): y \in N]$
- node x:
  - knows cost to each neighbor v: $c(x,v)$
  - maintains its neighbors' distance vectors. For each neighbor v, x maintains $\mathbf{D}_v = [D_v(y): y \in N]$

Distance vector algorithm

key idea:

- from time-to-time, each node sends its own distance vector estimate to neighbors
- when x receives new DV estimate from neighbor, it updates its own DV using B-F equation:

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

- under minor, natural conditions, the estimate $D_x(y)$ *converge to the actual least cost* $d_x(y)$

Distance vector algorithm

```
1  Initialization:
2     for all destinations y in N:
3        D_x(y) = c(x,y)    /* if y is not a neighbor then c(x,y) = ∞ */
4     for each neighbor w
5        D_w(y) = ? for all destinations y in N
6     for each neighbor w
7        send distance vector D_x = [D_x(y): y in N] to w
8
9  loop
10    wait (until I see a link cost change to some neighbor w or
11          until I receive a distance vector from some neighbor w)
12
13    for each y in N:
14       D_x(y) = min_v{c(x,v) + D_v(y)}
15
16    if D_x(y) changed for any destination y
17       send distance vector D_x = [D_x(y): y in N] to all neighbors
18
19 forever
```

$D_x(y) = \min\{c(x,y) + D_y(y), c(x,z) + D_z(y)\}$
$= \min\{2+0 , 7+1\} = 2$

$D_x(z) = \min\{c(x,y) + D_y(z), c(x,z) + D_z(z)\}$
$= \min\{2+1 , 7+0\} = 3$

node x table

from \ cost to	x	y	z
x	0	2	7
y	∞	∞	∞
z	∞	∞	∞

from \ cost to	x	y	z
x	0	2	3
y	2	0	1
z	7	1	0

from \ cost to	x	y	z
x	0	2	3
y	2	0	1
z	3	1	0

node y table

from \ cost to	x	y	z
x	∞	∞	∞
y	2	0	1
z	∞	∞	∞

from \ cost to	x	y	z
x	0	2	7
y	2	0	1
z	7	1	0

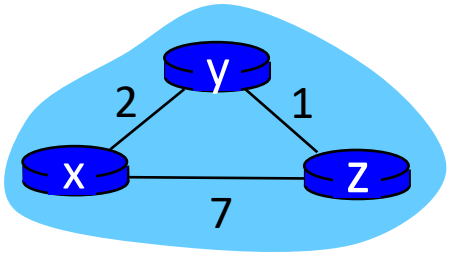
from \ cost to	x	y	z
x	0	2	3
y	2	0	1
z	3	1	0

node z table

from \ cost to	x	y	z
x	∞	∞	∞
y	∞	∞	∞
z	7	1	0

from \ cost to	x	y	z
x	0	2	7
y	2	0	1
z	3	1	0

from \ cost to	x	y	z
x	0	2	3
y	2	0	1
z	3	1	0



time

Comparison of LS and DV algorithms

message complexity

- ***LS:*** with n nodes, E links, O(nE) msgs sent
- ***DV:*** exchange between neighbors only
  - convergence time varies

speed of convergence

- ***LS:*** $O(n^2)$ algorithm requires O(nE) msgs
  - may have oscillations
- ***DV:*** convergence time varies
  - may be routing loops
  - count-to-infinity problem

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

Hierarchical routing

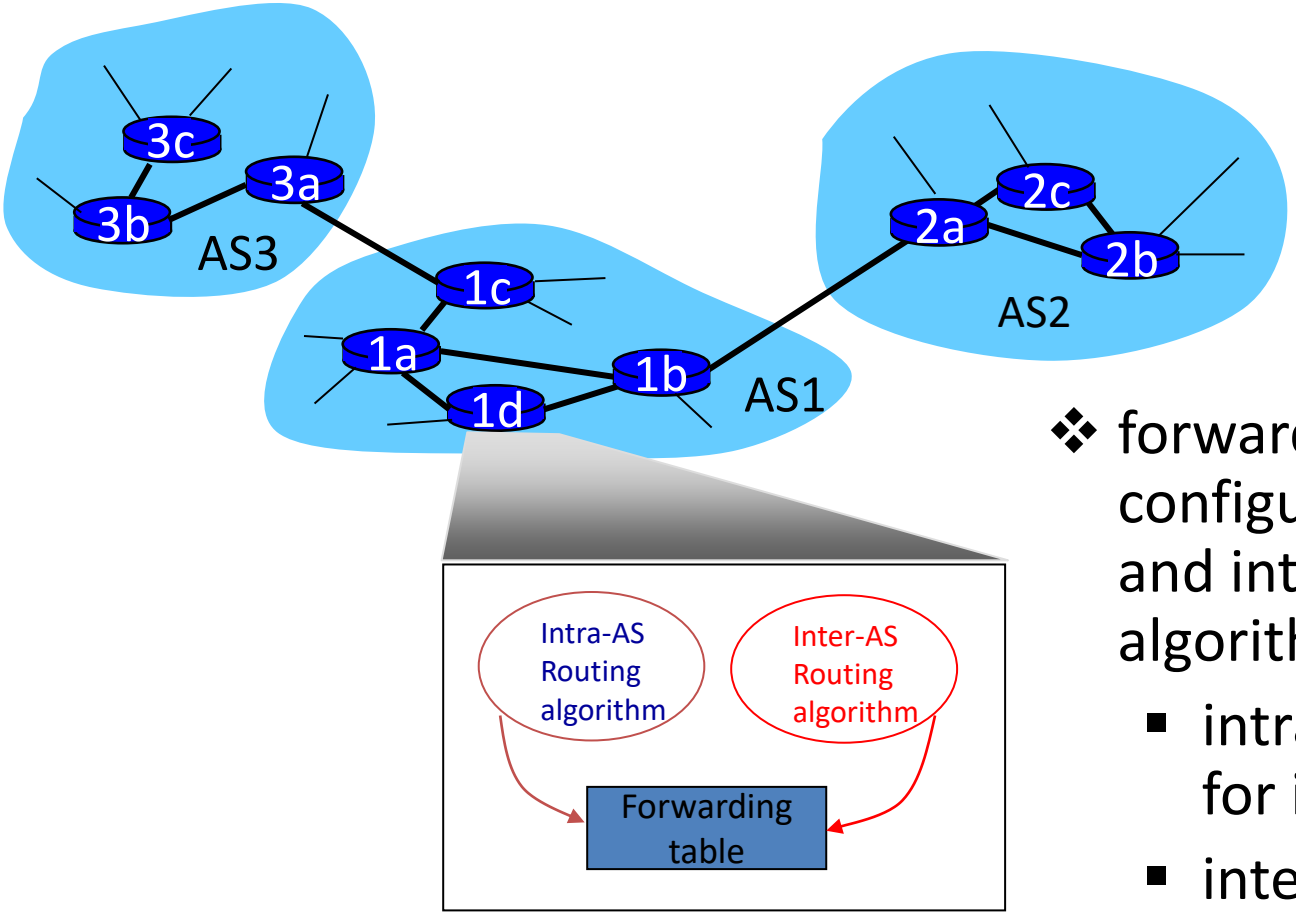
- collect routers into regions, "autonomous systems" (AS)
- Each AS within an ISP
  - ISP may consist of one or more ASes

- routers in same AS run same routing protocol
  - "intra-AS" routing protocol
  - routers in different AS can run different intra-AS routing protocol

gateway router:

- at "edge" of its own AS
- has link to router in another AS

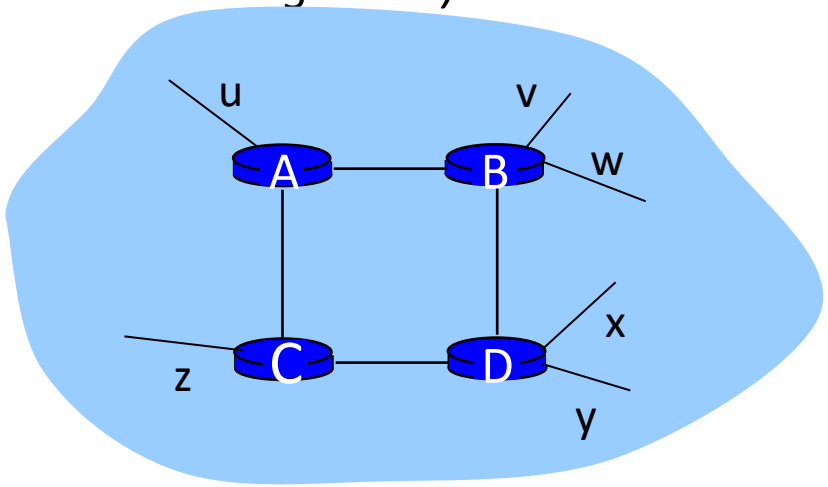
Interconnected ASes



- forwarding table configured by both intra- and inter-AS routing algorithm
  - intra-AS sets entries for internal dests
  - inter-AS & intra-AS sets entries for external dests

RIP (Routing Information Protocol)

- included in BSD-UNIX distribution in 1982
- distance vector algorithm
  - distance metric: # hops (max = 15 hops), each link has cost 1
  - AS should have diameter less than 15 hops
  - DVs exchanged with neighbors every 30 sec in RIP Response Message (aka advertisement)
  - each advertisement: list of up to 25 destination *subnets (in IP addressing sense)*



from router A to destination *subnets:*

subnet	hops
u	1
v	2
w	2
x	3
y	3
z	2

OSPF (Open Shortest Path First)

- "open": publicly available
  - rfc 2328
- uses link state algorithm
  - LS packet dissemination
  - topology map at each node
  - Shortest-path tree to all subnets locally computed by router using Dijkstra's algorithm
  - How to set link weight is up to the administrator

OSPF (Open Shortest Path First)

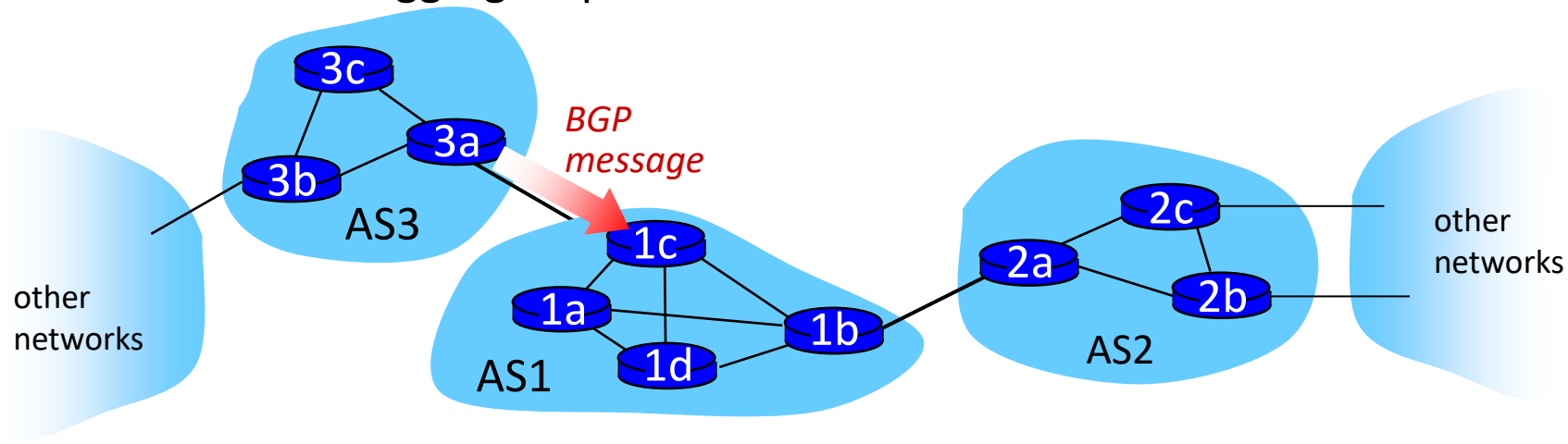
- advertisements flooded to *entire* AS
  - carried in OSPF messages directly over IP (rather than TCP or UDP)
  - Upper layer protocol 89
- A router broadcast link-state information when
  - A link's state changes
  - Periodically (e.g., 30 min)
- Check links with HELLO messages
- *IS-IS routing* protocol: nearly identical to OSPF

Internet inter-AS routing: BGP

- BGP (Border Gateway Protocol): *the* de facto inter-domain routing protocol
  - "glue that holds the Internet together"
- BGP provides each AS a means to:
  - obtain subnet reachability information from neighboring AS's: eBGP
  - propagate reachability information to all AS-internal routers: iBGP
  - determine "good" routes to other networks based on reachability information and policy.
- Basically, allows subnet to advertise its existence to rest of Internet: *"I am here"*

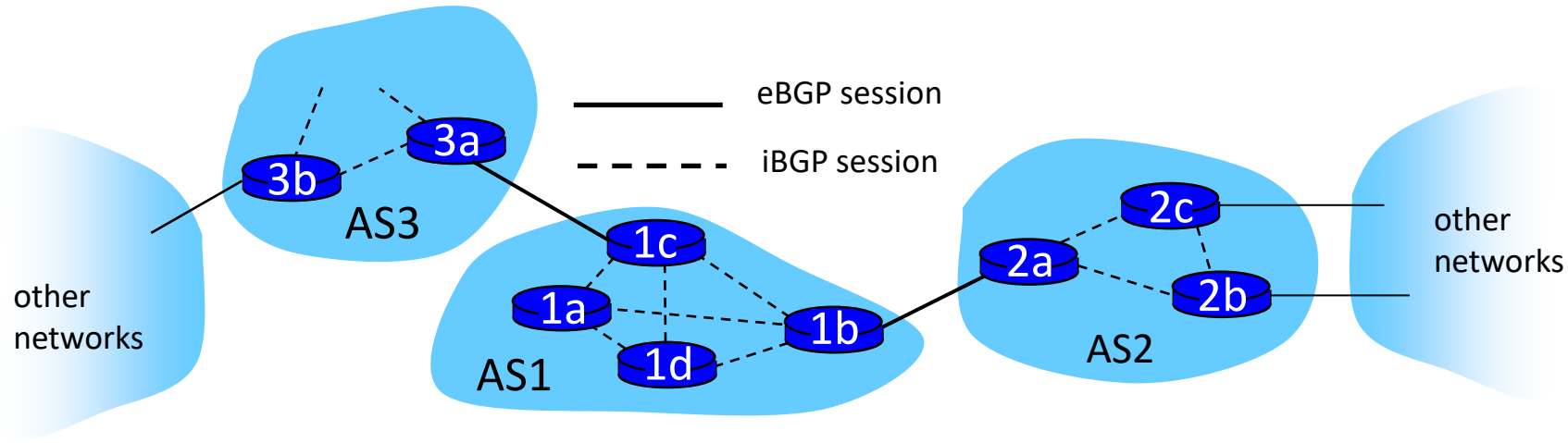
BGP basics

- BGP session: two BGP routers ("peers") exchange BGP messages:
  - advertising *paths* to different destination network prefixes ("path vector" protocol)
  - exchanged over semi-permanent TCP connections, port 179
- when AS3 advertises a prefix to AS1:
  - For example **138.16.64/22 ; AS-PATH: AS3 AS131**
  - AS3 *promises* it will forward datagrams towards that prefix
  - AS3 can aggregate prefixes in its advertisement



BGP basics: distributing path information

- using eBGP session between 3a and 1c, AS3 sends prefix reachability info to AS1.
  - 1c can then use iBGP sessions to distribute new prefix info to all routers in AS1
  - 1b can then re-advertise new reachability info to AS2 over 1b-to-2a eBGP session (**138.16.64/22 ; AS-PATH: AS1 AS3 AS131**)
- when router learns of new prefix, it creates entry for prefix in its forwarding table.



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
- These rules are applied *sequentially*

How does entry get in forwarding table?

Summary

1. Router becomes aware of prefix
   - via BGP route advertisements from other routers
2. Determine router output port for prefix
   - Use BGP route selection to find best inter-AS route
   - Use OSPF to find best intra-AS route leading to best inter-AS route
   - Router identifies router port for that best route
3. Enter prefix-port entry in forwarding table

Why different Intra-, Inter-AS routing ?

policy:

- inter-AS: admin wants control over how its traffic routed, who routes through its net.
- intra-AS: single admin, so no policy decisions needed

scale:

- hierarchical routing saves table size, reduced update traffic

performance:

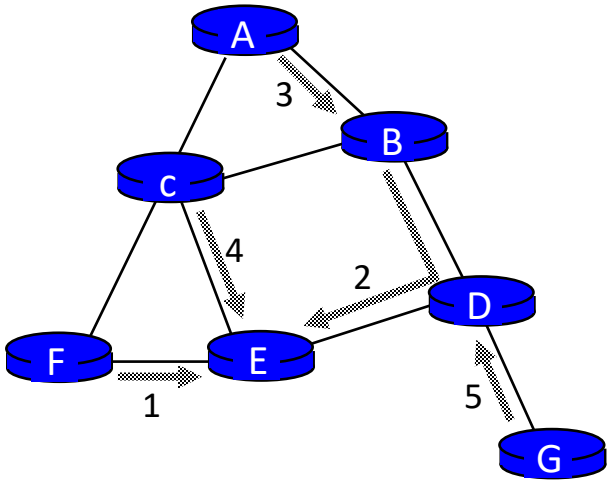
- intra-AS: can focus on performance
- inter-AS: policy may dominate over performance

In-network duplication

- *Uncontrolled flooding:* when node receives broadcast packet, sends copy to all neighbors
  - problems: cycles & broadcast storm
- *controlled flooding:* node only broadcasts pkt if it hasn't broadcast same packet before
  - node keeps track of packet ids already broadacsted (src. addr. + seq. num)
  - or reverse path forwarding (RPF): only forward packet if it arrived on shortest path between node and source
- *spanning tree:*
  - no redundant packets received by any node

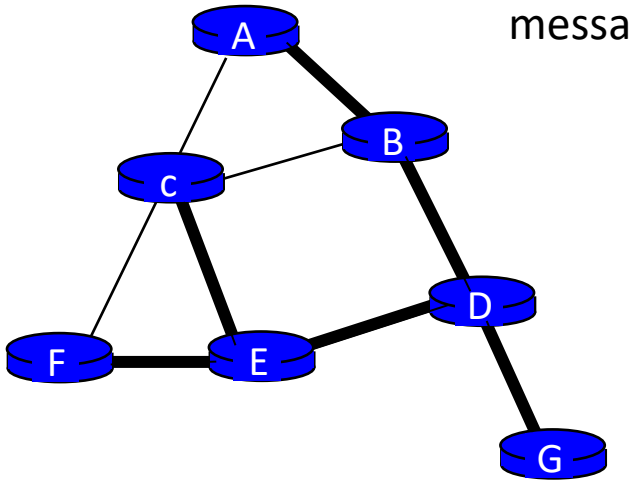
Spanning tree: creation

- Define a center node
- each node sends unicast join message to center node
  - message forwarded until it arrives at a node already belonging to spanning tree



(a) stepwise construction of spanning tree (center: E)

D joins by B's message



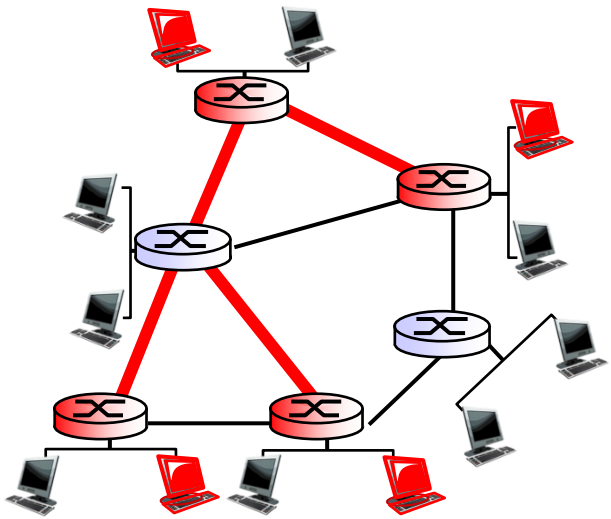
(b) constructed spanning tree

Multicast routing: problem statement

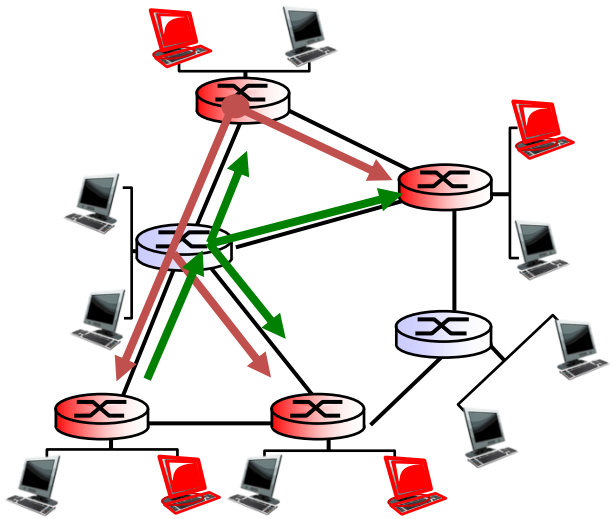
goal: find a tree (or trees) connecting routers having local mcast group members

tree: not all paths between routers used

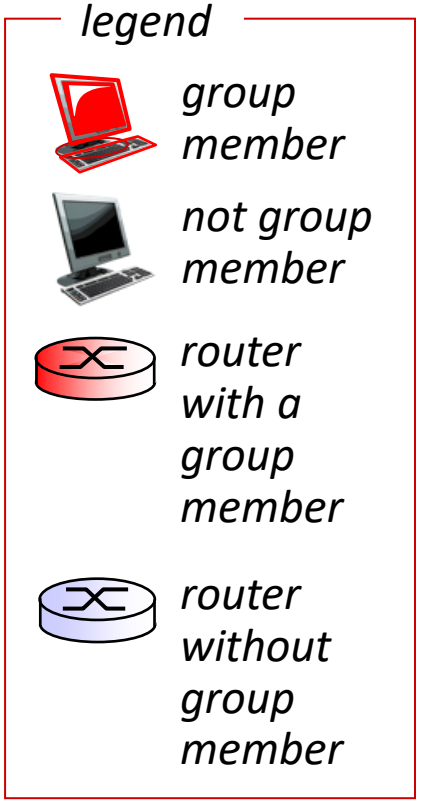
- *shared-tree:* same tree used by all group members
- *source-based:* different tree from each sender to rcvrs



shared tree



source-based trees



Constructing group-shared tree

- *Center-based approach*
- one router identified as *"center"* of tree
- to join:
  - edge router sends unicast *join-msg* addressed to center router
  - *join-msg* "processed" by intermediate routers and forwarded towards center
  - *join-msg* either hits existing tree branch for this center, or arrives at center
  - path taken by *join-msg* becomes new branch of tree for this router

Constructing source-based tree

RFP (reverse path forwarding) algorithm

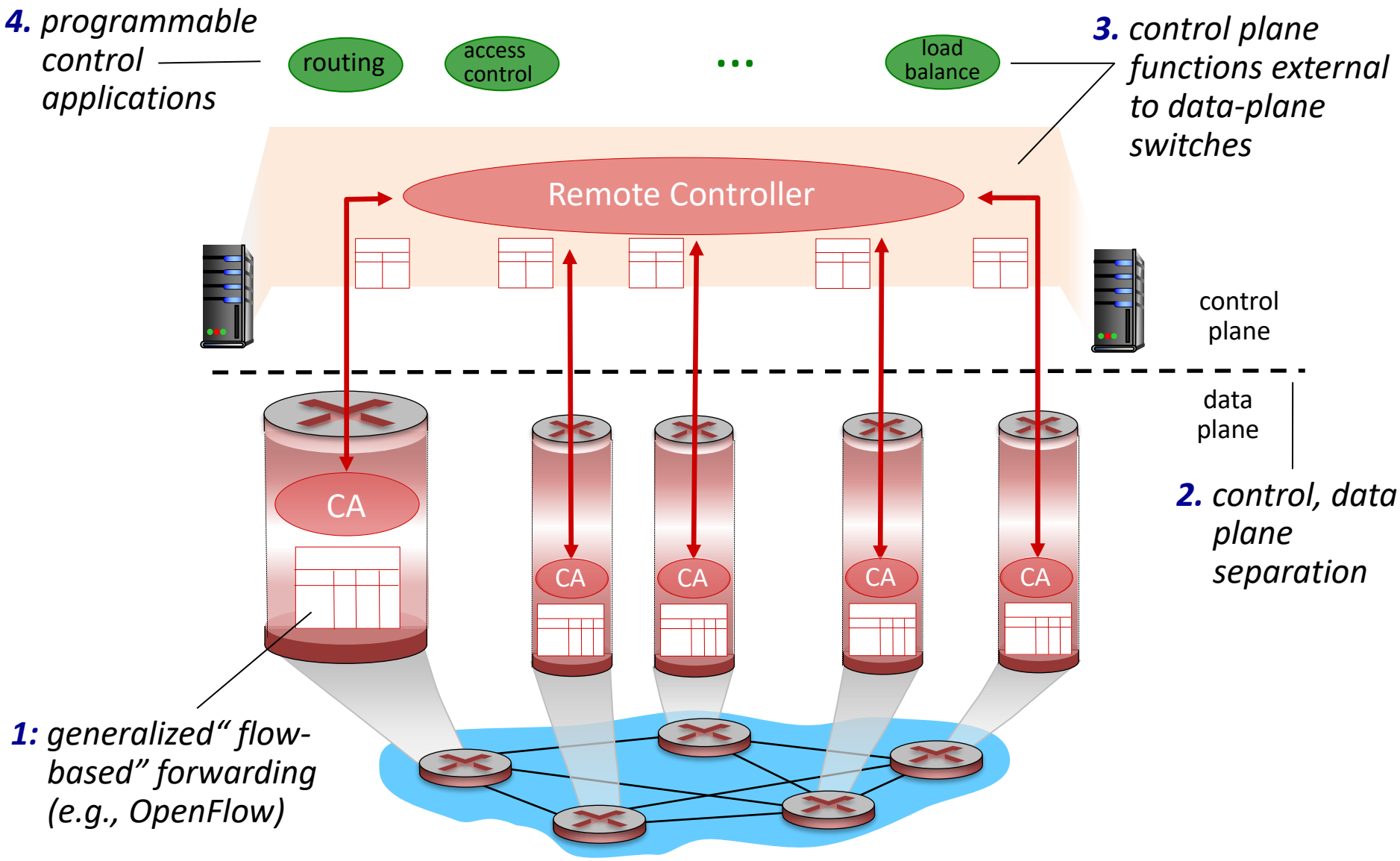
- rely on router's knowledge of unicast shortest path from it to sender
- each router has simple forwarding behavior:

if (mcast datagram received on incoming link on shortest path back to source)

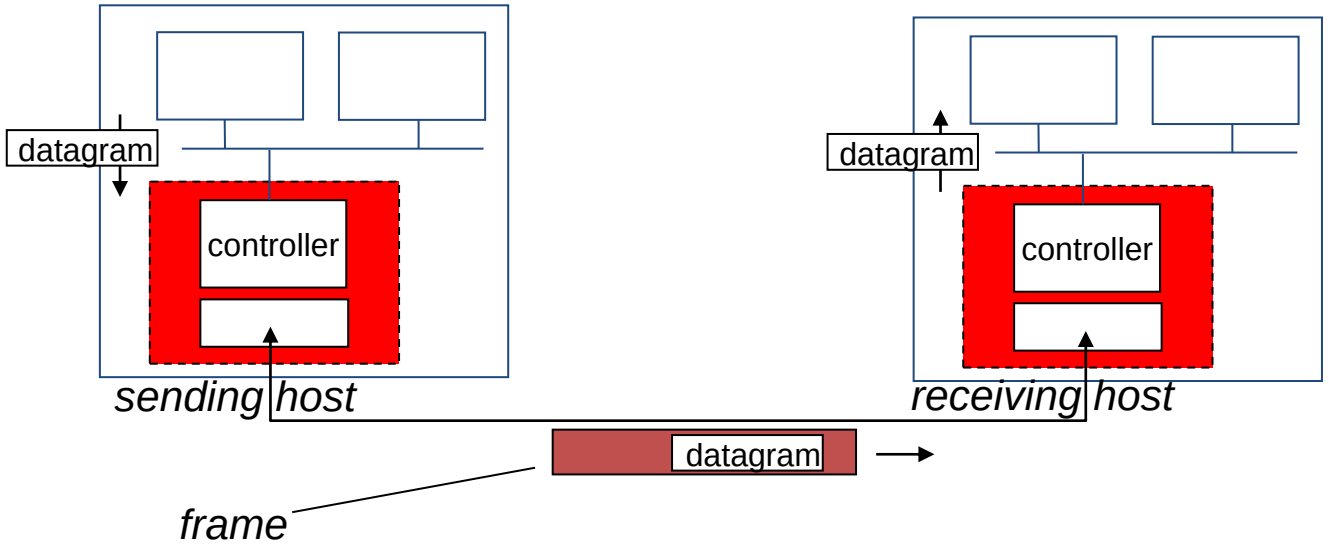
then flood datagram onto all outgoing links

else ignore datagram

Software defined networking (SDN)



Adaptors communicating

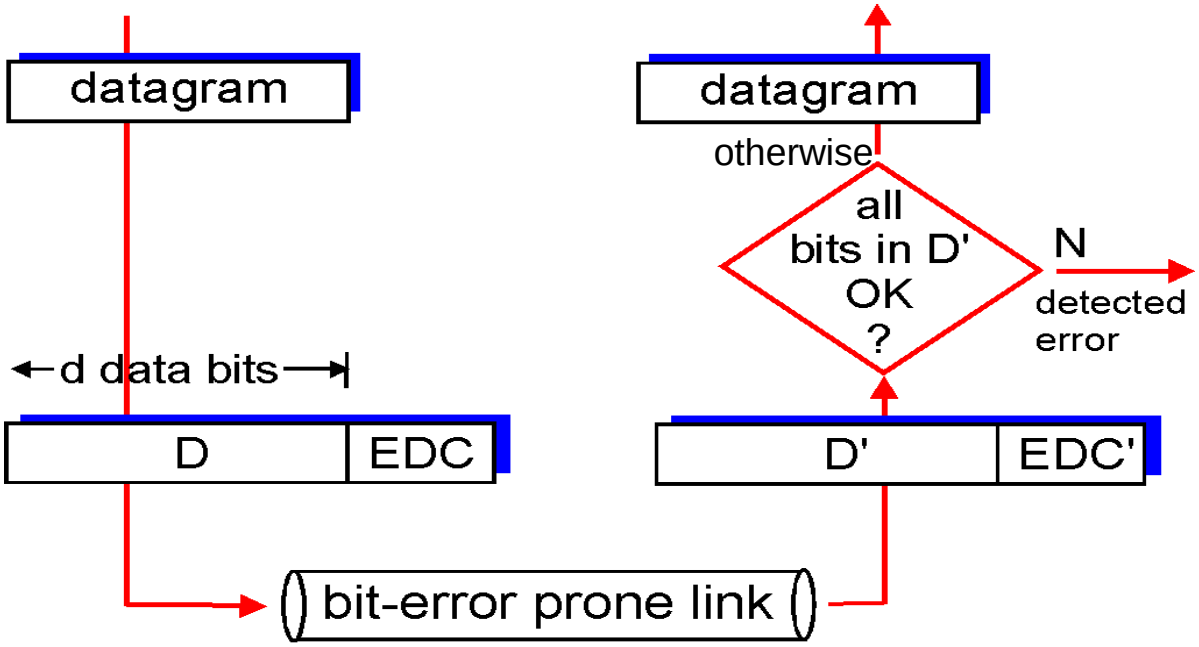


- sending side:
  - encapsulates datagram in frame
  - adds error checking bits, rdt, flow control, etc.
- receiving side
  - looks for errors, rdt, flow control, etc.
  - extracts datagram, passes to upper layer at receiving side

Error detection

EDC= Error Detection and Correction bits (redundancy)
D = Data protected by error checking, may include header fields

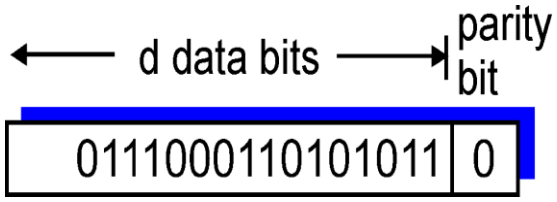
- Error detection not 100% reliable!
  - protocol may miss some errors, but rarely
  - larger EDC field yields better detection and correction



Parity checking

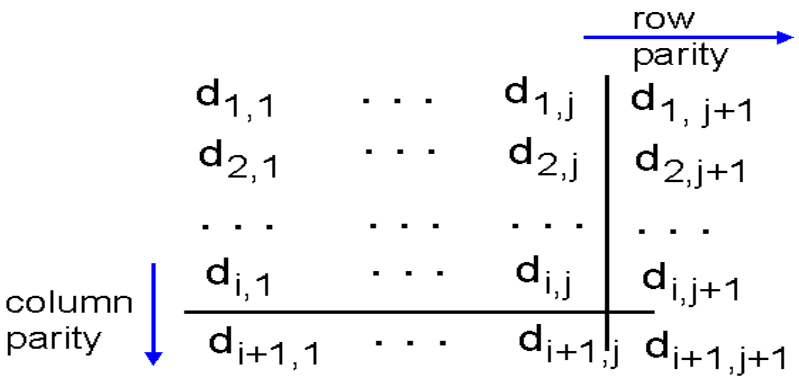
single bit parity:

- *d*etect single bit errors



two-dimensional bit parity:

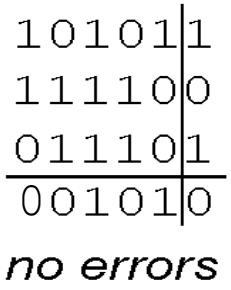
- detect and correct single bit errors

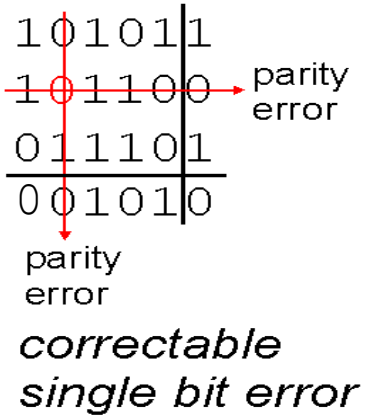


Even/odd parity: parity bit is set to 1 if there is an odd/even number of one bits

偶校验：确保1 bit数是偶数

奇校验：确保1 bit数是奇数

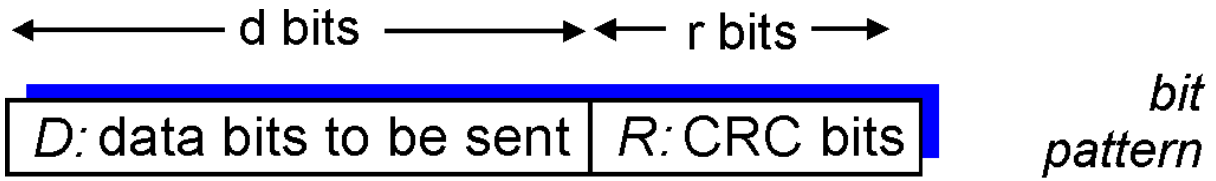




* Check out the online interactive exercises for more examples: http://gaia.cs.umass.edu/kurose_ross/interactive/

Cyclic redundancy check

- view data bits, D, as a binary number
- choose r+1 bit pattern (generator), G
- goal: choose r CRC bits, R, such that
  - <D,R> exactly divisible by G (modulo 2)
  - receiver knows G, divides <D,R> 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, ATM)



$D * 2^r$ XOR R

mathematical formula

CRC example

want:

$D \cdot 2^r$ XOR R = nG

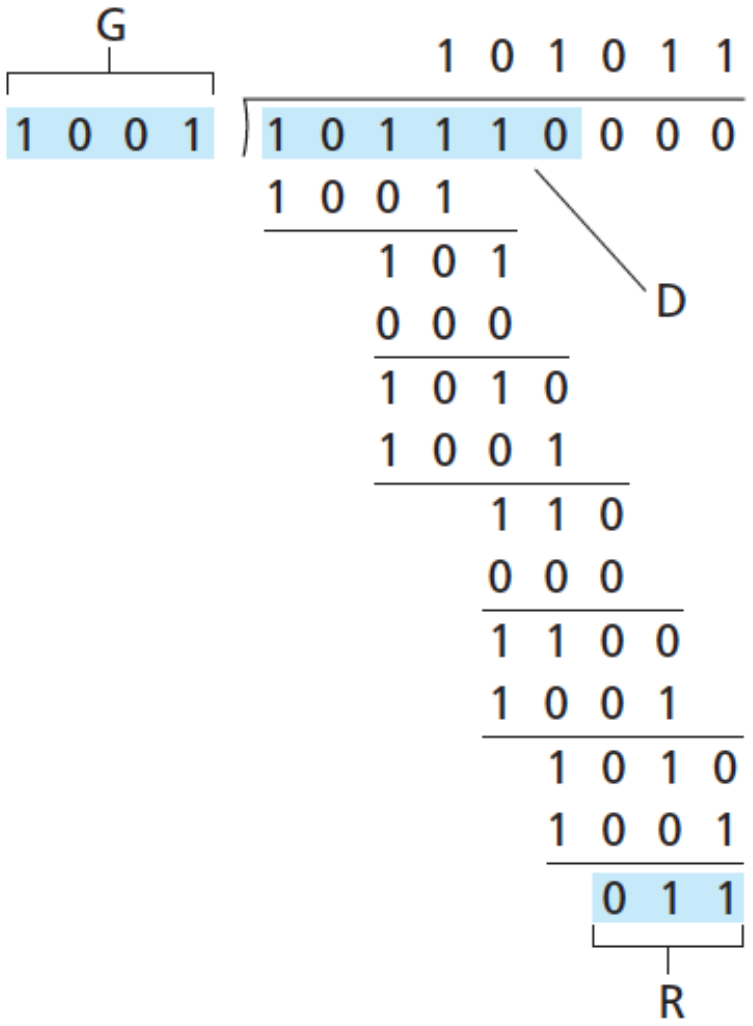
equivalently:

$D \cdot 2^r$ = nG XOR R

equivalently:

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

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



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

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

multiple access protocol

- distributed algorithm that determines how nodes share channel, i.e., determine when node can transmit
- communication about channel sharing must use channel itself!
  - no out-of-band channel for coordination

MAC protocols: taxonomy

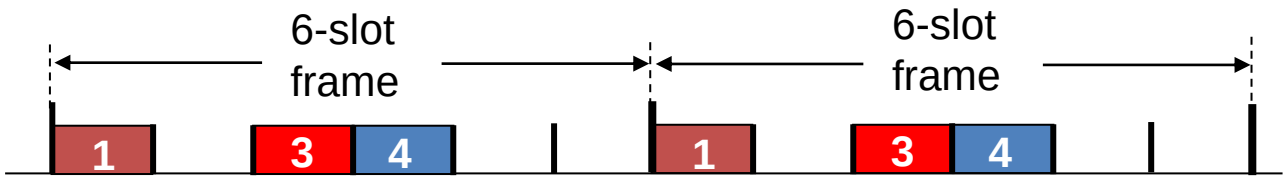
three broad classes:

- *channel partitioning*（信道划分）
  - divide channel into smaller "pieces" (time slots, frequency, code)
  - allocate piece to node for exclusive use
- *random access*（随机接入）
  - channel not divided, allow collisions
  - "recover" from collisions
- *"taking turns"*（轮询）
  - nodes take turns, but nodes with more to send can take longer turns

Channel partitioning MAC protocols: TDMA

TDMA: time division multiple access

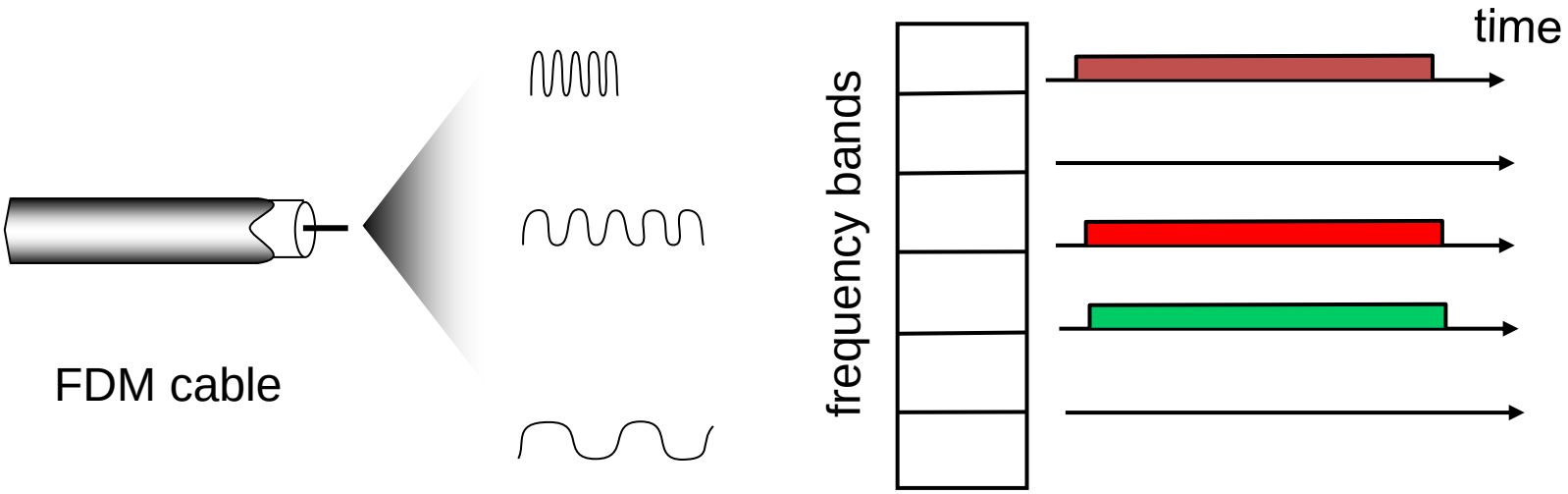
- access to channel in "rounds"
- each station gets fixed length slot (length = packet transmission time) in each round
- unused slots go idle
- example: 6-station LAN, 1,3,4 have packets to send, slots 2,5,6 idle



Channel partitioning MAC protocols: FDMA

FDMA: frequency division multiple access

- channel spectrum divided into frequency bands
- each station assigned fixed frequency band
- unused transmission time in frequency bands go idle
- example: 6-station LAN, 1,3,4 have packet to send, frequency bands 2,5,6 idle



Random access protocols

- when node has packet to send
  - transmit at full channel data rate R.
  - no *a priori* coordination among nodes
- two or more transmitting nodes ➜ "collision",
- random access MAC protocol specifies:
  - how to detect collisions
  - how to recover from collisions (e.g., via delayed retransmissions)
- examples of random access MAC protocols:
  - slotted ALOHA, ALOHA
  - CSMA, CSMA/CD, CSMA/CA

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 prob. p until success

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:

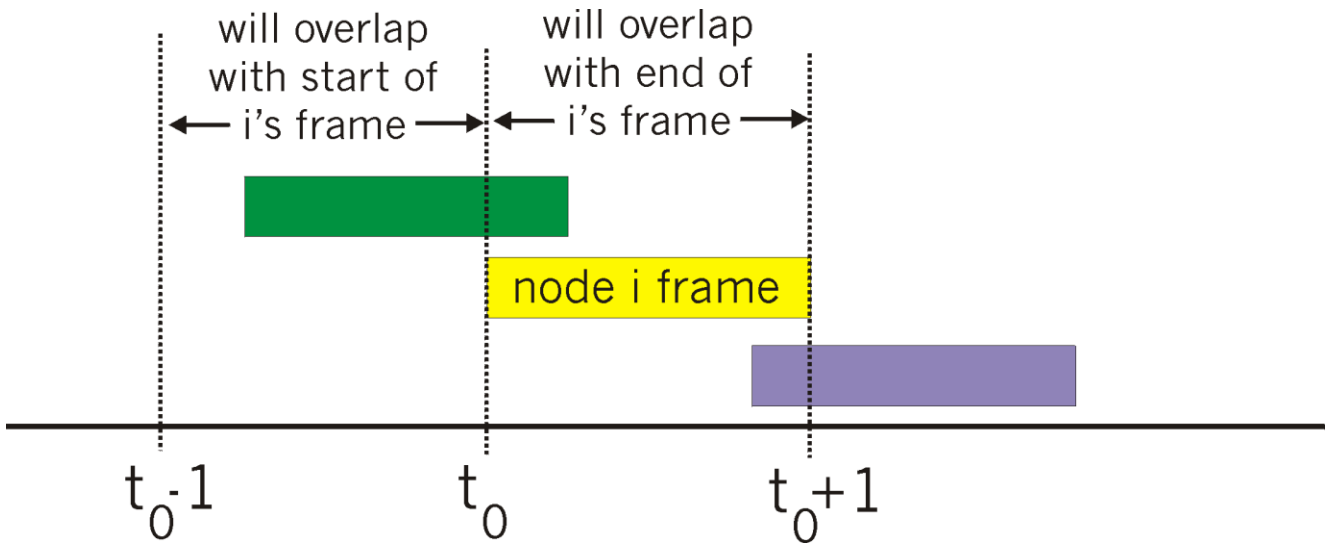
max efficiency $= 1/e = .37$

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

!

Pure (unslotted) ALOHA

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



Pure ALOHA efficiency

P(success by given node) = P(node transmits) ·

P(no other node transmits in $[t_0\text{-}1, t_0]$ ·

P(no other node transmits in $[t_0, t_0+1]$

$= p \cdot (1-p)^{N-1} \cdot (1-p)^{N-1}$

$= p \cdot (1-p)^{2(N-1)}$ $\longrightarrow \infty$

… choosing optimum p and then letting *n*

$= 1/(2e) = .18$

even *worse* than slotted Aloha!

CSMA (carrier sense multiple access)

CSMA: listen before transmit:

if channel sensed idle: transmit entire frame

- if channel sensed busy, defer transmission

Called *carrier sensing* *（载波侦听）*

- human analogy: don't interrupt others!

Ethernet CSMA/CD algorithm

1. NIC receives datagram from network layer, creates frame
2. If NIC senses channel idle, starts frame transmission. If NIC senses channel busy, waits until channel idle, then transmits.
3. If NIC transmits entire frame without detecting another transmission, NIC is done with frame !
4. If NIC detects another transmission while transmitting, aborts and sends jam signal
   - Jam signal, inform other stations must not transmit
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 $K \cdot 512$ bit times, returns to Step 2
   - longer backoff interval with more collisions

CSMA/CD efficiency

- t_{prop} = max prop delay between 2 nodes in LAN
- t_{trans} = time to transmit max-size frame

$$efficiency = \frac{1}{1 + 5t_{prop}/t_{trans}}$$

- efficiency goes to 1
  - as t_{prop} goes to 0 (abort immediately without wasting the channel)
  - as t_{trans} goes to infinity (never release the channel)
- better performance than ALOHA: and simple, cheap, decentralized!
  - Reason: listen and abort

MAC addresses and ARP

- 32-bit IP address:
  - *network-layer* address for interface
  - used for layer 3 (network layer) forwarding
- MAC (or LAN or physical or Ethernet) address:
  - 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, also sometimes software settable
  - e.g.: 1A-2F-BB-76-09-AD

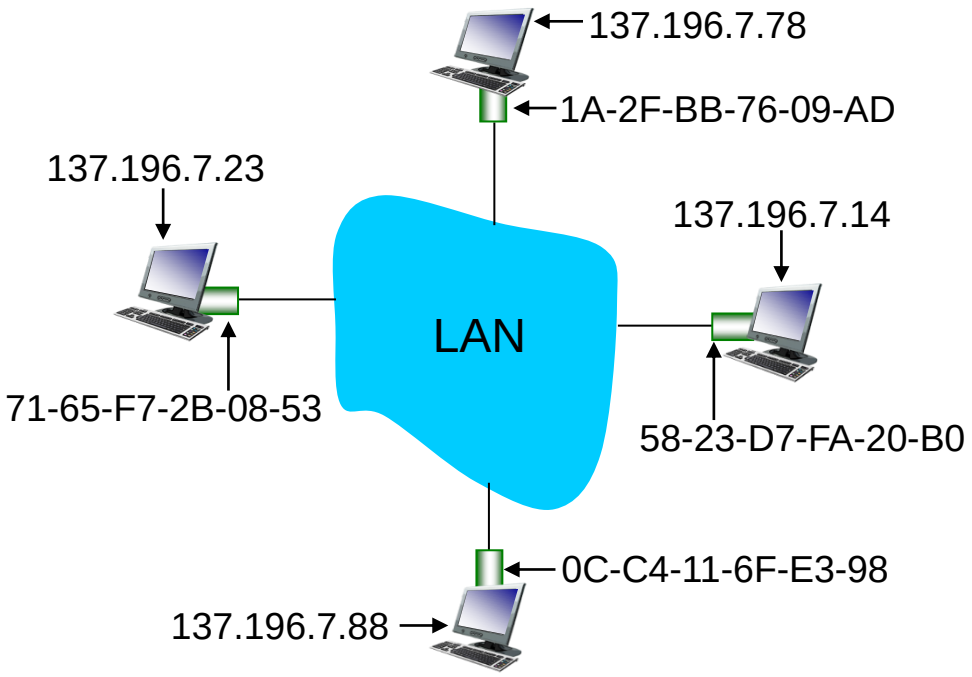
hexadecimal (base 16) notation
(each "numeral" represents 4 bits)

LAN addresses (more)

- MAC flat address ➜ portability
  - can move LAN card from one LAN to another
- IP hierarchical address *not* portable
  - address depends on IP subnet to which node is attached
- Sending adapter: inserts the destination adapter' MAC address into the frame then sends to the LAN
- Receiving adapter: Check to see whether the destination MAC matches its own MAC address
- Broadcast address: FF-FF-FF-FF-FF-FF

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>

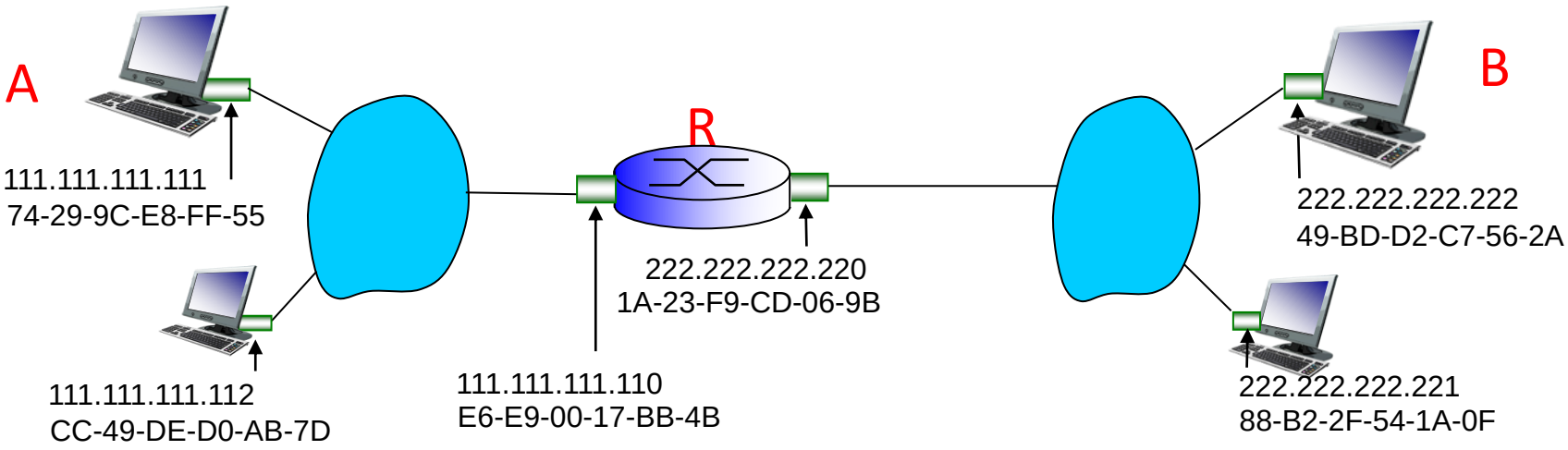
- TTL (Time To Live): time after which address mapping will be forgotten (typically 20 min)

Addressing: routing to another LAN

Can A put B's MAC address in the frame? No. Why?

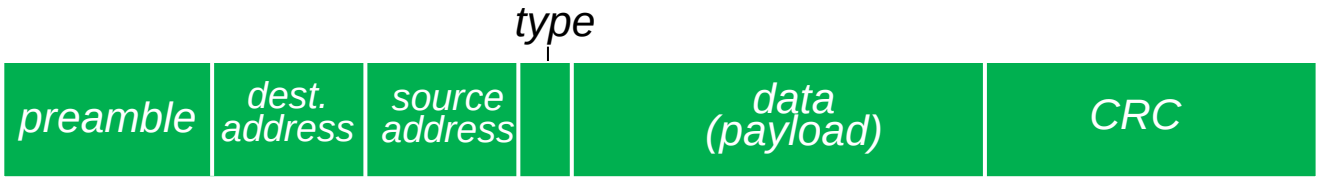
walkthrough: send datagram from A to B via R

- 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? DHCP)
- assume A knows R's MAC address (how? ARP)



Ethernet frame structure (more)

- *addresses:* 6 byte source, destination MAC addresses
  - if adapter receives frame with matching destination address, or with broadcast address (e.g. ARP packet), it passes data in frame to network layer protocol
  - otherwise, adapter discards frame
- *type:* indicates higher layer protocol (mostly IP but others possible, e.g., Novell IPX, AppleTalk)
- *CRC:* cyclic redundancy check at receiver
  - error detected: frame is dropped



Ethernet: unreliable, connectionless

- *connectionless:* no handshaking between sending and receiving NICs
- *unreliable:* receiving NIC doesn't send acks or nacks to sending NIC
  - data in dropped frames recovered only if initial sender uses higher layer rdt (e.g., TCP), otherwise dropped data lost
- Ethernet's MAC protocol: unslotted *CSMA/CD with binary backoff*

Ethernet switch

- link-layer device: takes an *active* role
  - 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, uses CSMA/CD to access segment
- *transparent*
  - hosts are unaware of presence of switches
- *plug-and-play, self-learning*
  - switches do not need to be configured

Forwarding and filtering

Two basic functionalities:

- *Filtering*: determines whether a frame should be forwarded to some interface or should just be dropped.
- *Forwarding*: determines the interfaces to which a frame should be directed, and then moves the frame to those interfaces.
- Done with *switch table*.

Q: how are entries created, maintained in switch table?

- something like a routing protocol?

Switch: frame filtering/forwarding

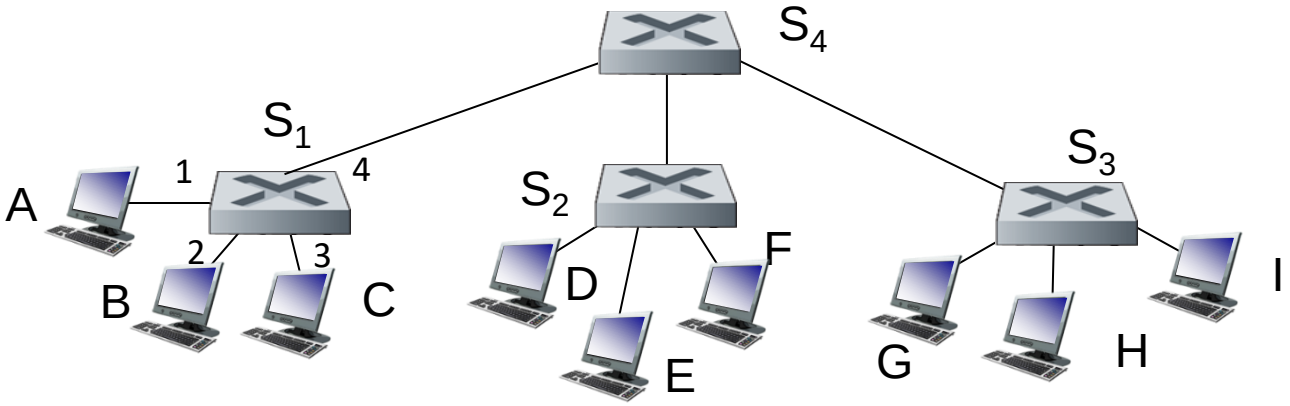
when frame received at switch:

1. record incoming link, source MAC address of sending host
2. index switch table using destination MAC address
3. if entry found for destination

```
then {
if destination on segment from which frame arrived
    then drop frame
    else forward frame on interface indicated by entry
}
else flood  /* forward on all interfaces except arriving
                interface */
```

Self-learning multi-switch example

Suppose C sends frame to I, I responds to C



- Q: show switch tables and packet forwarding in S_1, S_2, S_3, S_4

MAC addr.	Interface	TTL
A	1	
B	2	
C	3	
D、E、F、G、H、I	4	

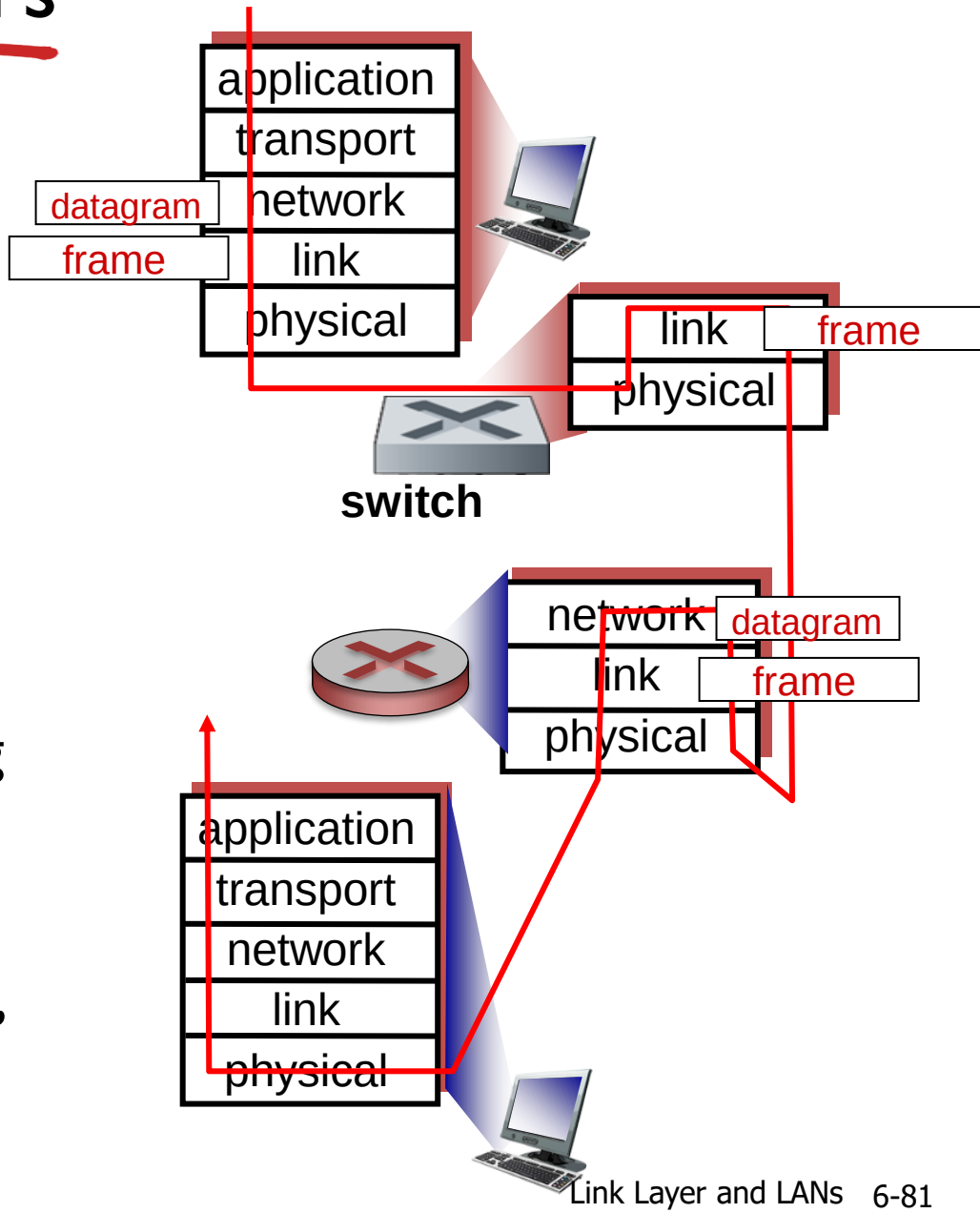
Switches vs. routers

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:

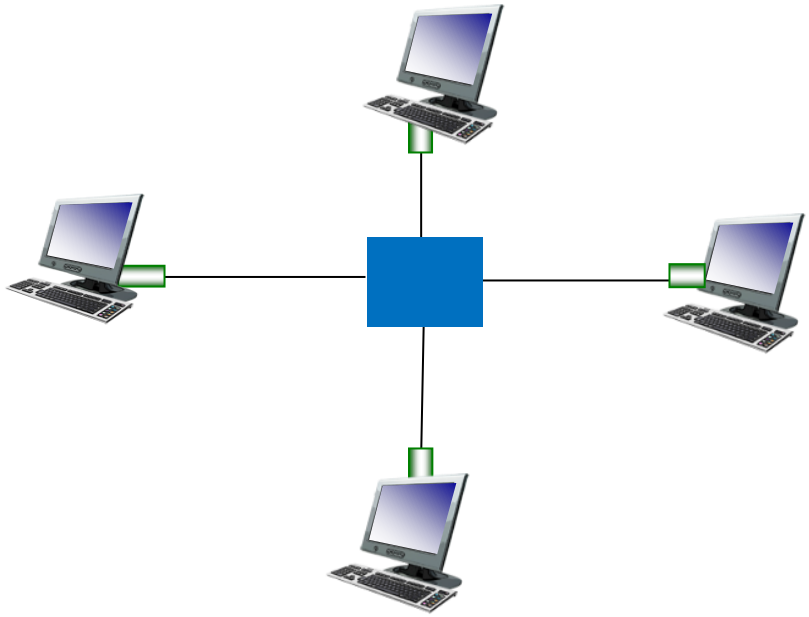
- *routers:* compute tables using routing algorithms, IP addresses
- *switches:* learn forwarding table using flooding, learning, MAC addresses



Switches vs. routers

- Both are interconnecting devices

	Hubs	Routers	Switches
Traffic isolation	No	Yes	Yes
Plug and play	Yes	No	Yes
Optimal routing	No	Yes	No

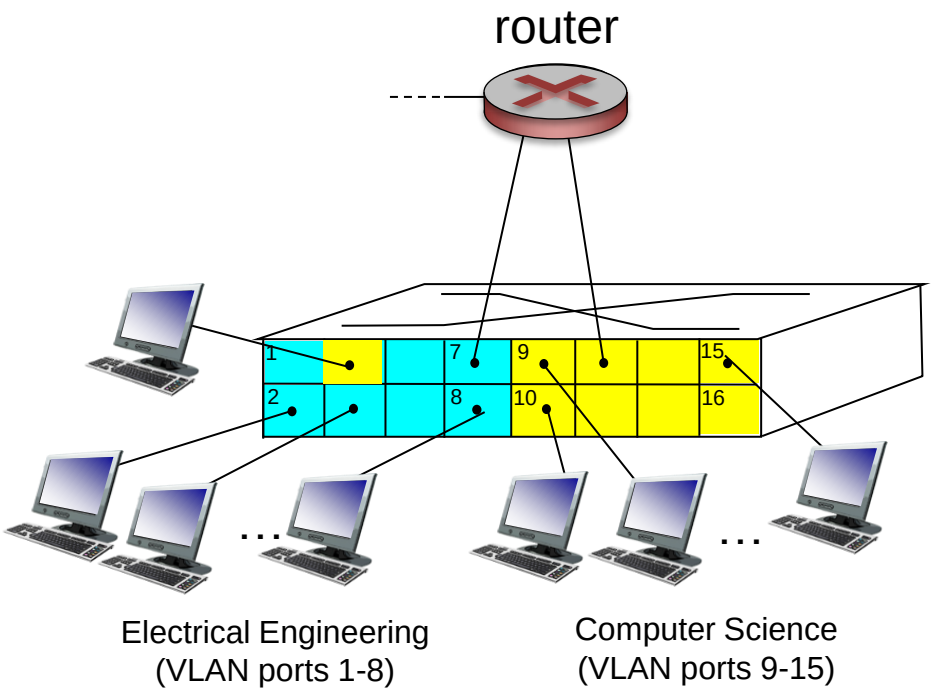


Port-based VLAN

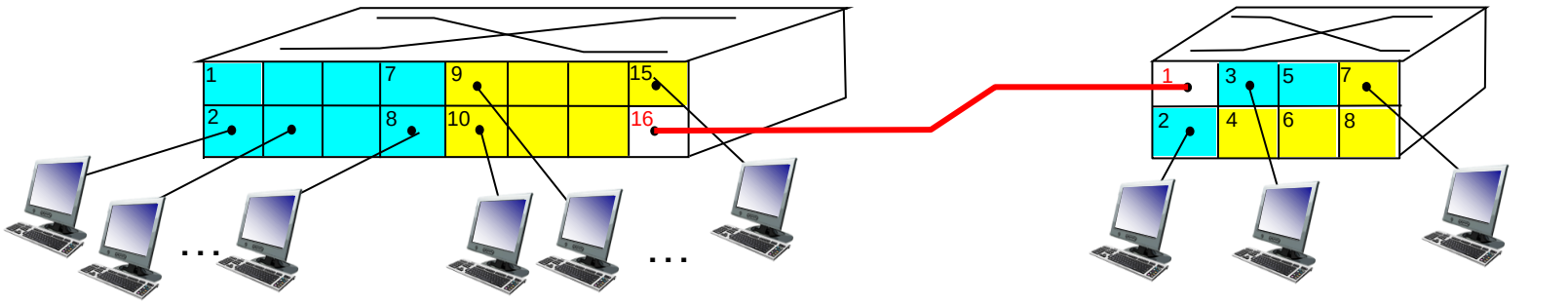
- *traffic isolation:* frames to/from ports 1-8 can *only* reach ports 1-8
  - can also define VLAN based on MAC addresses of endpoints, rather than switch port

- dynamic membership: ports can be dynamically assigned among VLANs

- forwarding between VLANS: done via routing (just as with separate switches)
  - in practice vendors sell combined switches plus routers

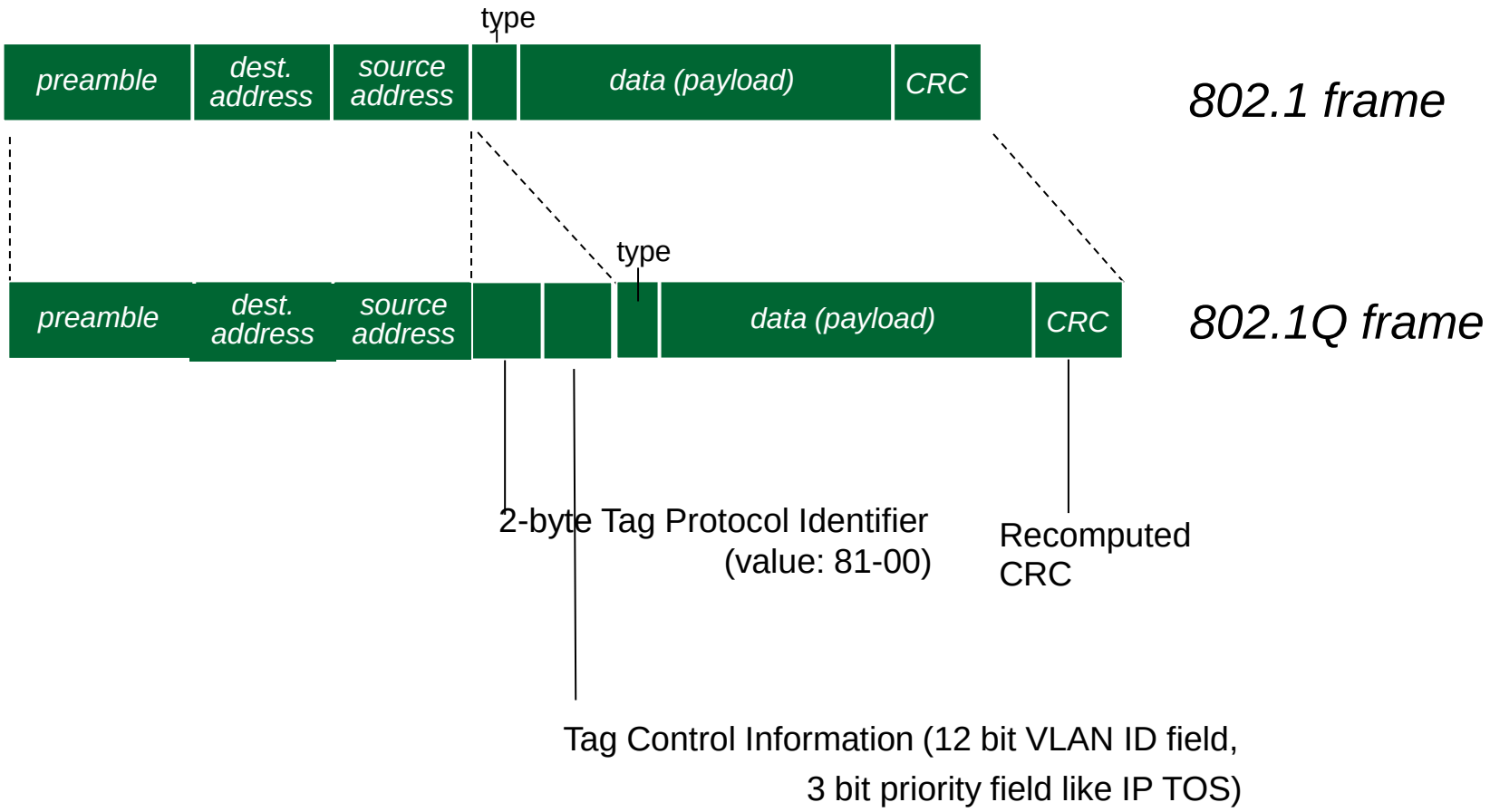


VLANS spanning multiple switches

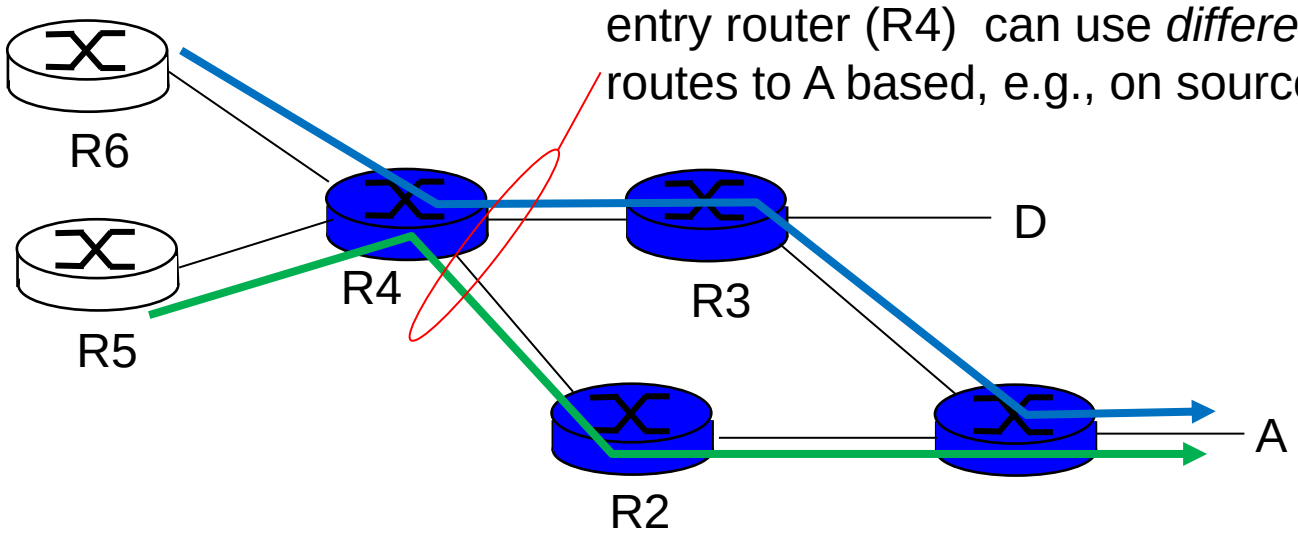


- *trunk port:* carries frames between VLANS defined over multiple physical switches
  - frames forwarded within VLAN between switches can't be vanilla 802.1 frames (must carry VLAN ID info)
  - 802.1q protocol adds/removed additional header fields for frames forwarded between trunk ports

802.1Q VLAN frame format

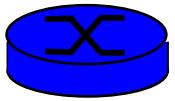


MPLS versus IP paths



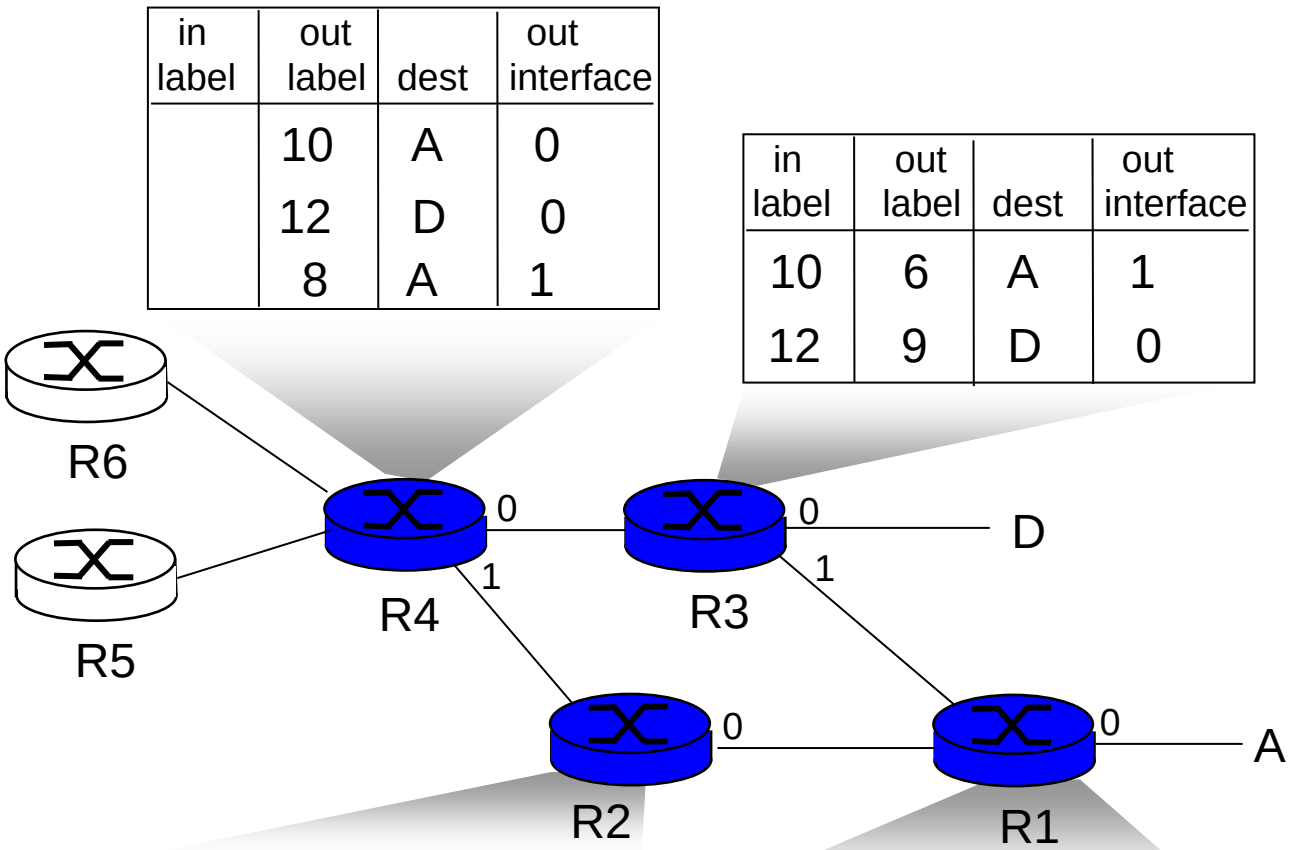
- *IP routing: path to destination determined by destination address alone*
- *MPLS routing:* path to destination can be based on source *and* destination address
  - *fast reroute:* precompute backup routes in case of link failure

IP-only router



MPLS and IP router

MPLS forwarding tables



Wireless network taxonomy

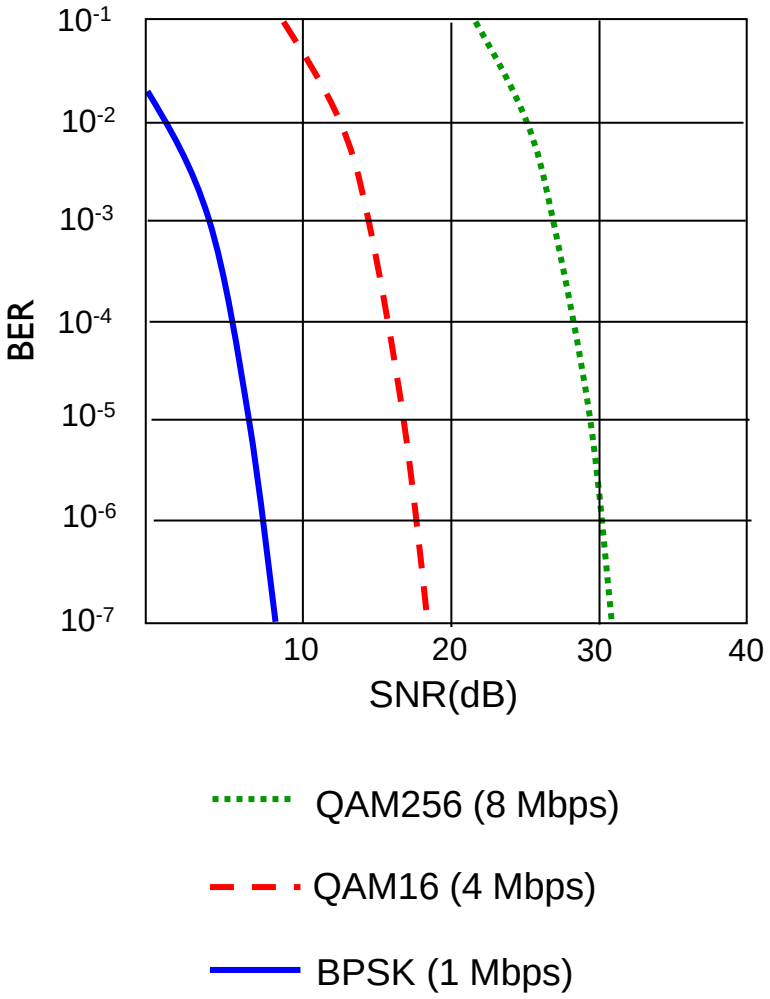
	single hop	multiple hops
infrastructure (e.g., APs)	host connects to base station (WiFi, WiMAX, cellular) which connects to larger Internet	host may have to relay through several wireless nodes to connect to larger Internet: *mesh net*
no infrastructure	no base station, no connection to larger Internet (Bluetooth, ad hoc nets)	no base station, no connection to larger Internet. May have to relay to reach other a given wireless node MANET, VANET

Wireless Link Characteristics (2)

- SNR: signal-to-noise ratio
  - larger SNR – easier to extract signal from noise (a "good thing")

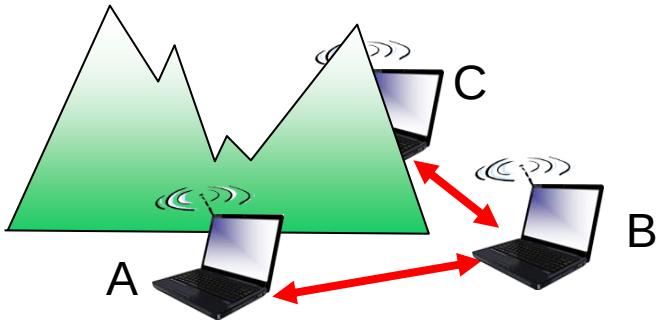
$$\text{SNR(dB)} = 10 \log_{10} \left(\frac{P_{\text{signal}}}{P_{\text{noise}}} \right)$$

- *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
    - SNR may change with mobility: dynamically adapt physical layer (modulation technique, rate)



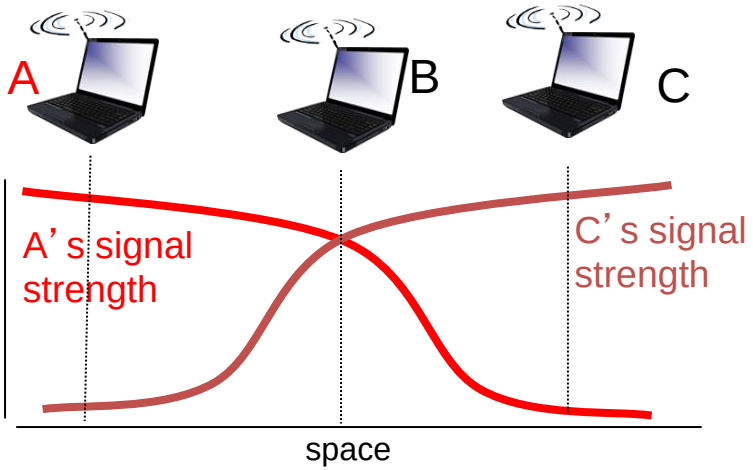
Wireless network characteristics

Multiple wireless senders and 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:

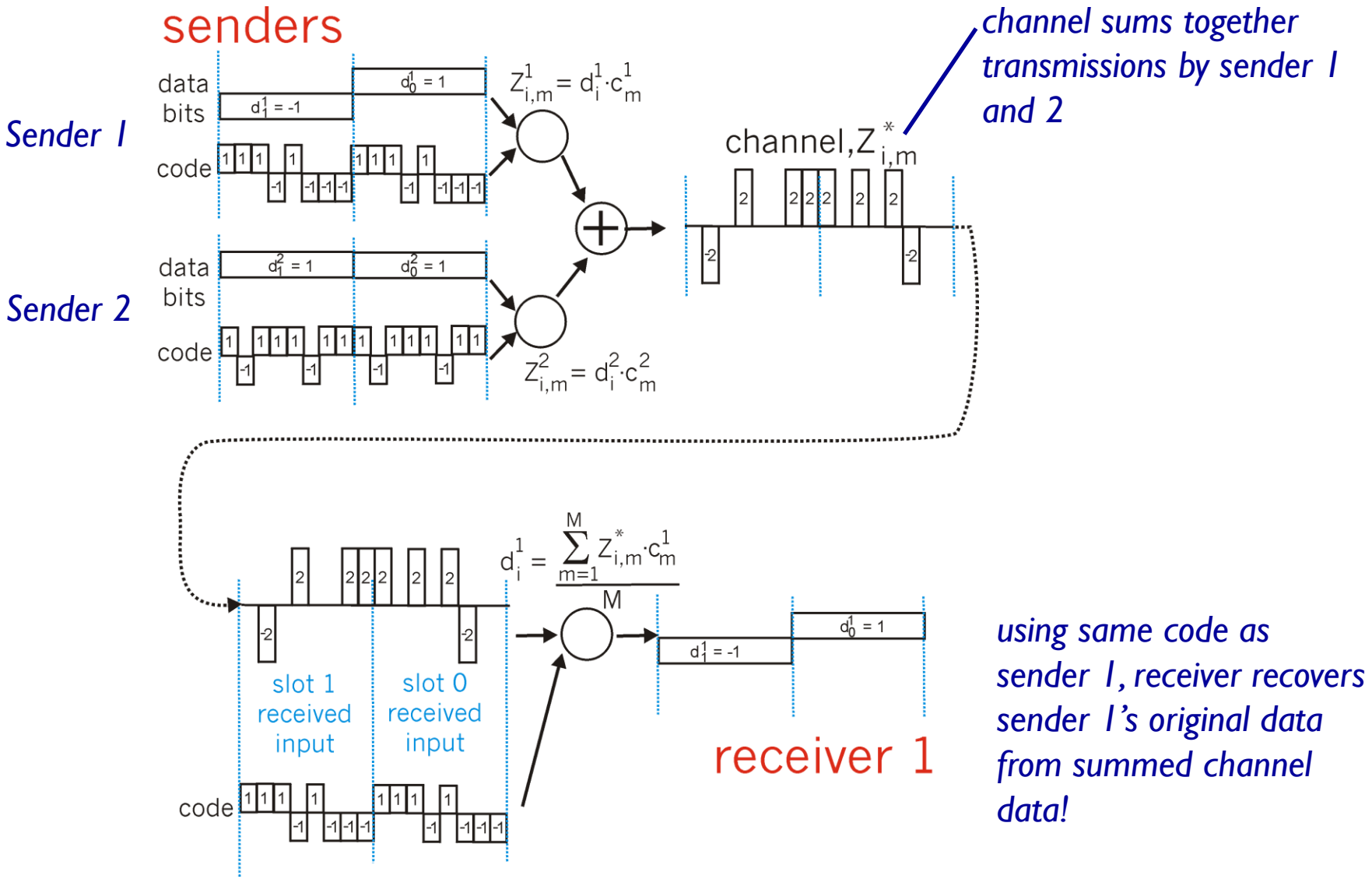
- B, A hear each other
- B, C hear each other
- A, C can not hear each other interfering at B

CSMA/CD not work!

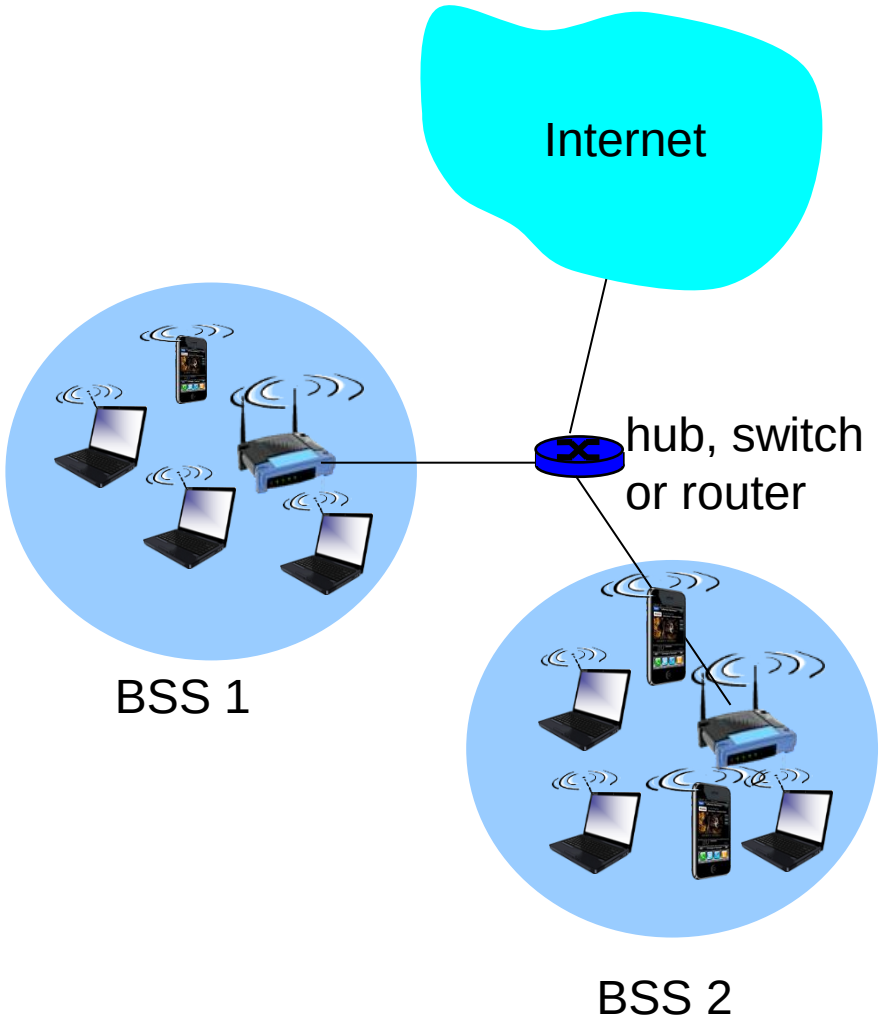
Code Division Multiple Access (CDMA)

- 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., CDMA code) to encode data
  - allows multiple users to "coexist" and transmit simultaneously with minimal interference (if codes are "orthogonal")
- *encoded signal* = (original data) X (CDMA code)
- *decoding:* inner-product of encoded signal and CDMA code

CDMA: two-sender interference



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

IEEE 802.11 MAC Protocol: CSMA/CA

802.11 sender

1 if sense channel idle for **DIFS** then

transmit entire frame (no CD)

2 if sense channel busy then

start random backoff timer value using binary exponential backoff (as in 802.3)

timer counts down while channel idle / freeze while channel busy

3 if timer reach zero

transmit and wait for ACK

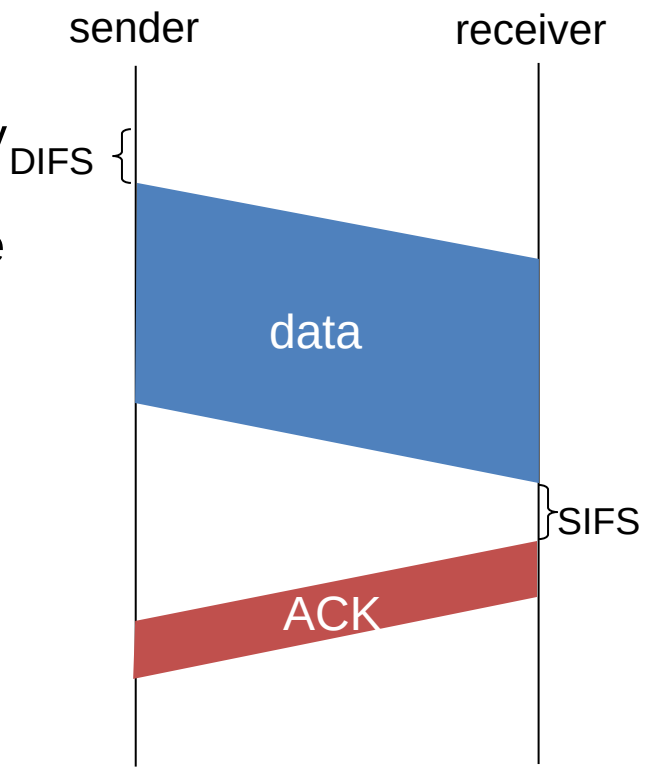
4 if no ACK

increase random backoff interval, repeat 2

802.11 receiver

- if frame received OK

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



Avoiding collisions (more)

idea: allow sender to "reserve" channel rather than random access of data frames: avoid collisions of long data frames

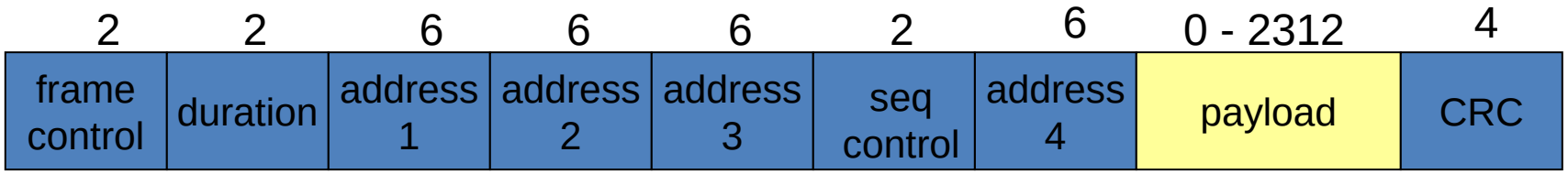
- sender first transmits *small* request-to-send (RTS) packets to BS using CSMA
  - RTSs may still collide with each other (but they're short)
- BS broadcasts clear-to-send CTS in response to RTS
- CTS heard by all nodes
  - sender transmits data frame
  - other stations defer transmissions

avoid data frame collisions completely using small reservation packets!

Collision Avoidance: RTS-CTS

- Avoid data frame collisions completely
  - The hidden station problem is mitigated, since a long DATA frame is transmitted only after the channel has been reserved.
  - A collision involving an RTS or CTS frame will last only for the duration of the short RTS or CTS frame. Once the RTS and CTS frames are correctly transmitted, the following DATA and ACK frames should be transmitted without collisions.
- RTS/CTS is only used when transmitting long DATA frame

802.11 frame: addressing



Address 1: MAC address of wireless host or AP to receive this frame

Address 2: MAC address of wireless host or AP transmitting this frame

Address 3: MAC address of router interface to which AP is attached

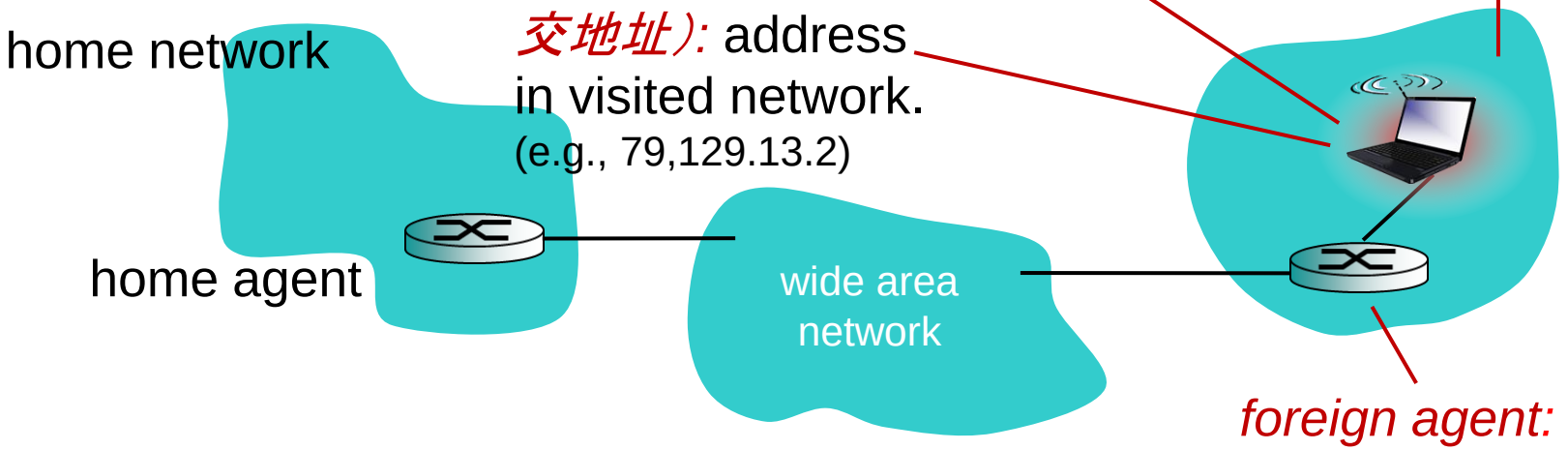
Address 4: used only in ad hoc mode

Mobility: more vocabulary

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

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

care-of-address (转交地址): address in visited network. (e.g., 79,129.13.2)

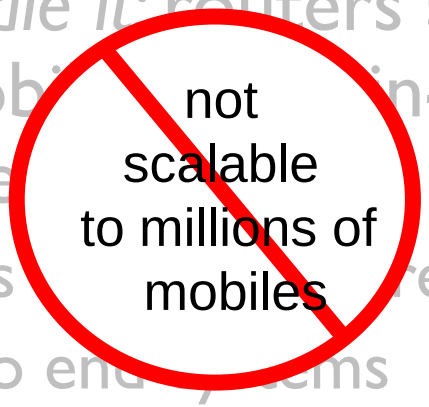


correspondent: wants to communicate with mobile

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

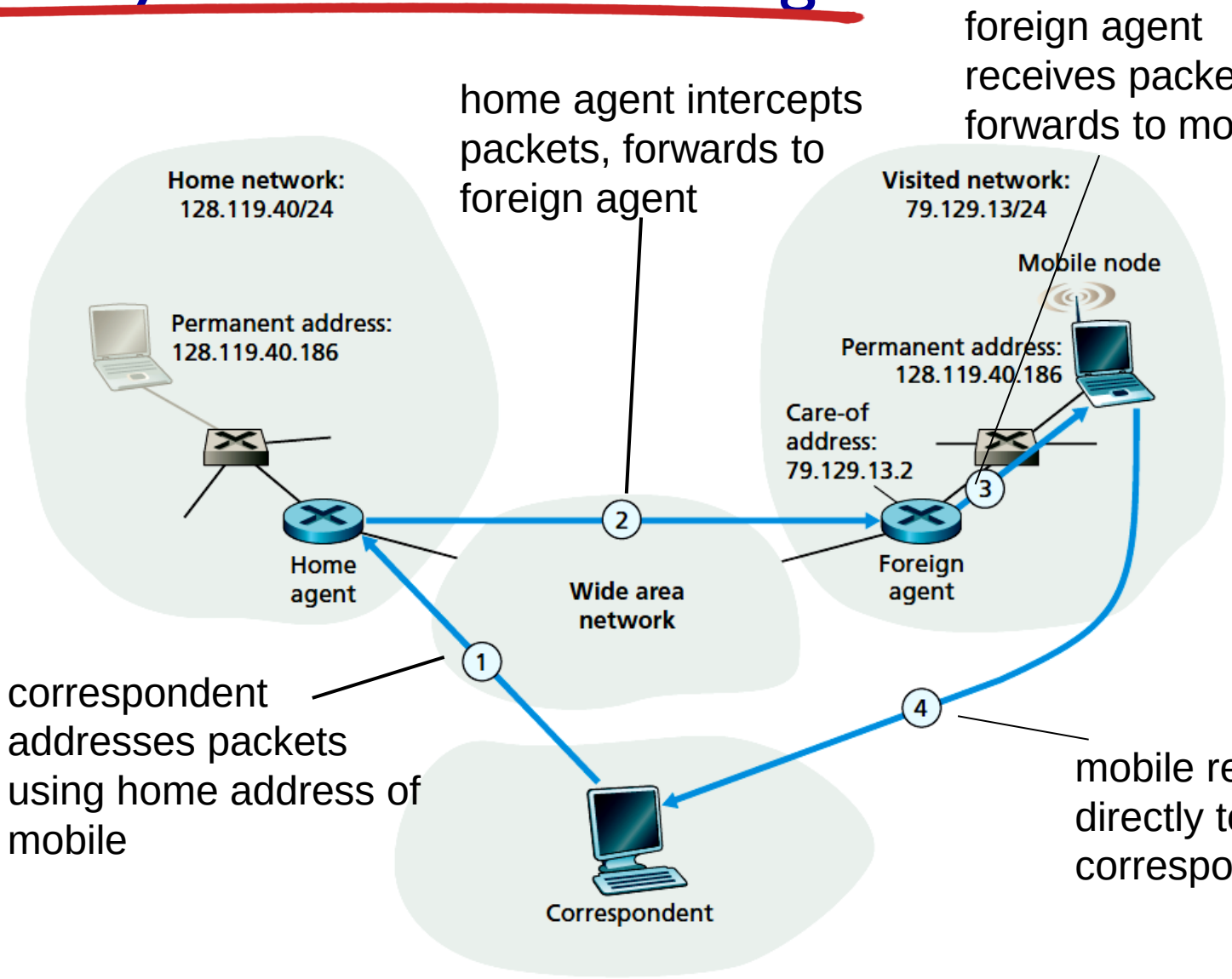
Mobility: approaches

- *let routing handle it:* routers advertise permanent address of mobile-in-residence via usual routing table exchange
  - routing tables indicate where each mobile located
  - no changes to end-systems

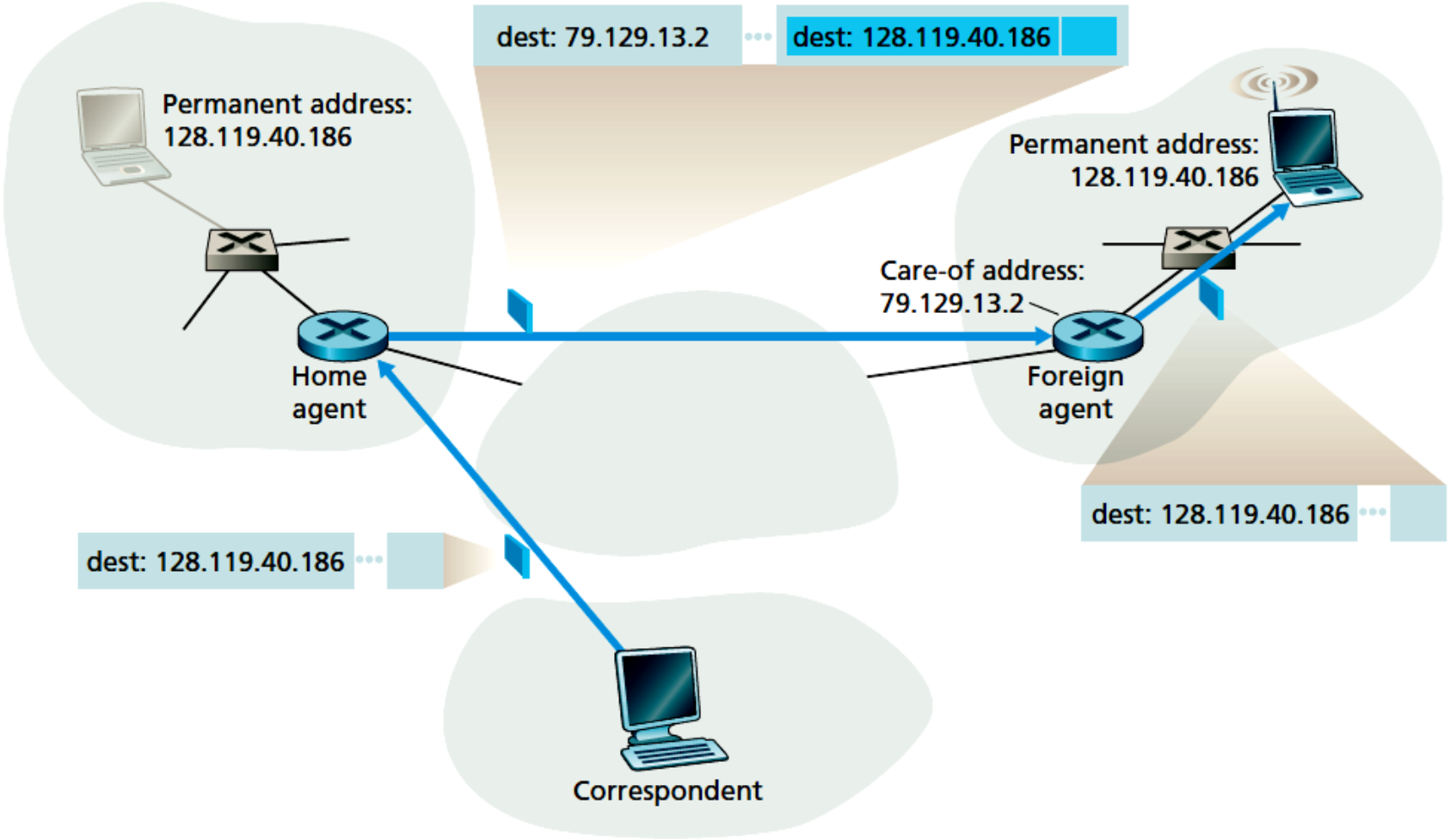


- *let end-systems handle it:*
  - *indirect routing:* communication from correspondent to mobile goes through home agent, then forwarded to remote
  - *direct routing:* correspondent gets foreign address of mobile, sends directly to mobile

Mobility via indirect routing

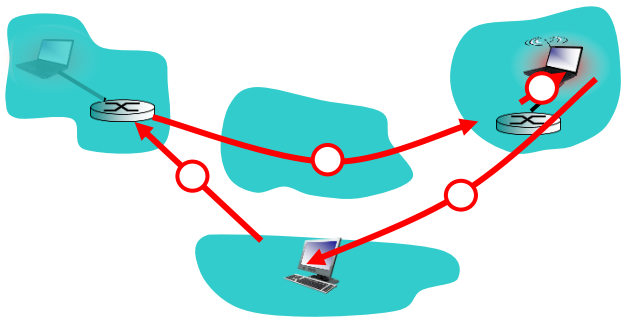


Datagram forwarding

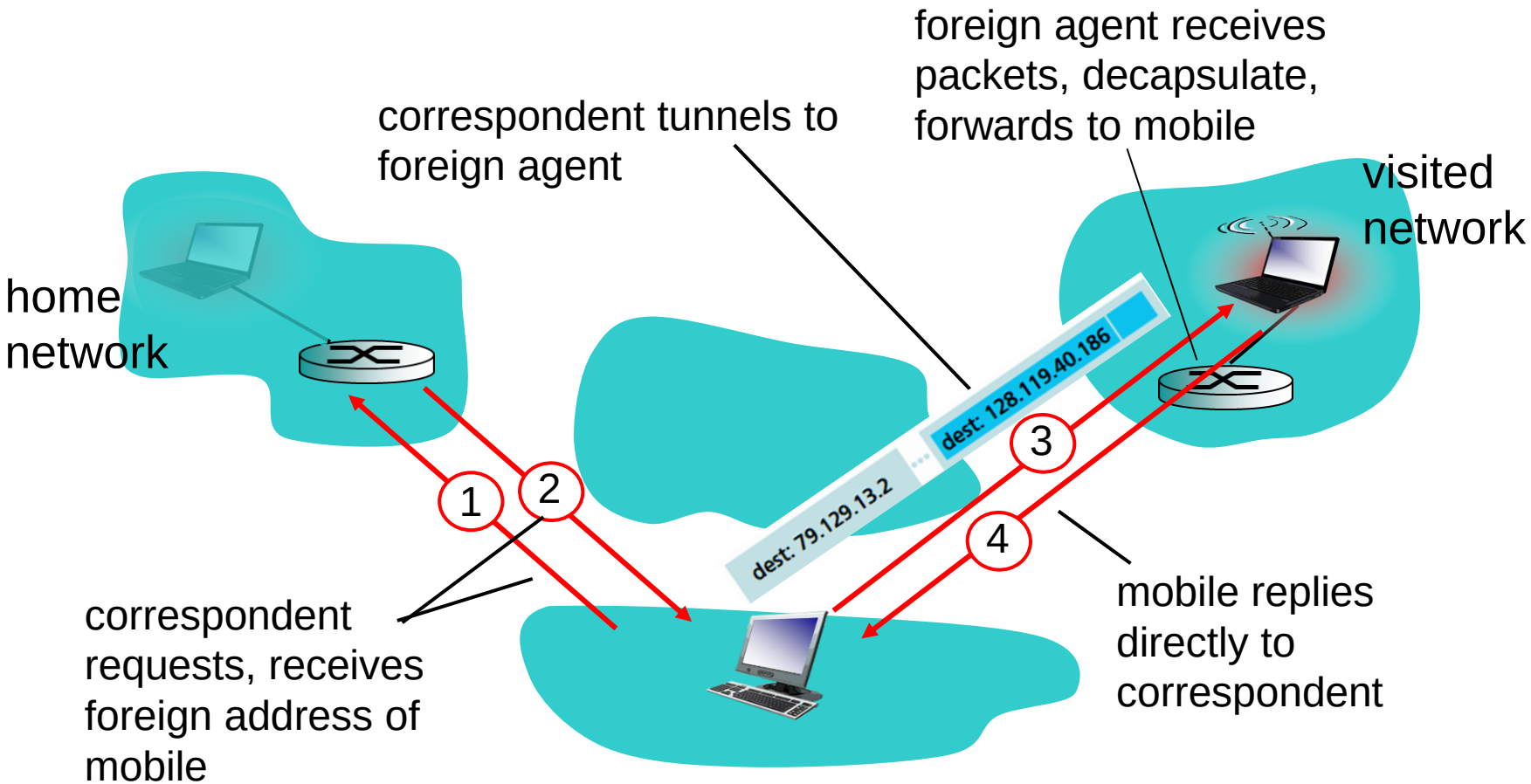


Indirect Routing: comments

- mobile uses two addresses:
  - permanent address: used by correspondent (hence mobile location is *transparent* to correspondent)
  - care-of-address: used by home agent to forward datagrams to mobile
- foreign agent functions may be done by mobile itself
- triangle routing: correspondent-home-network-mobile
  - inefficient when correspondent, mobile are in same network

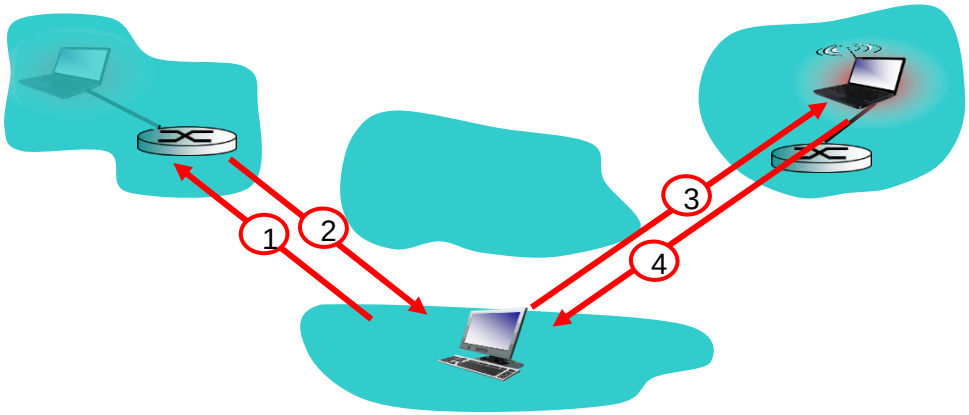


Mobility via direct routing



Mobility via direct routing: comments

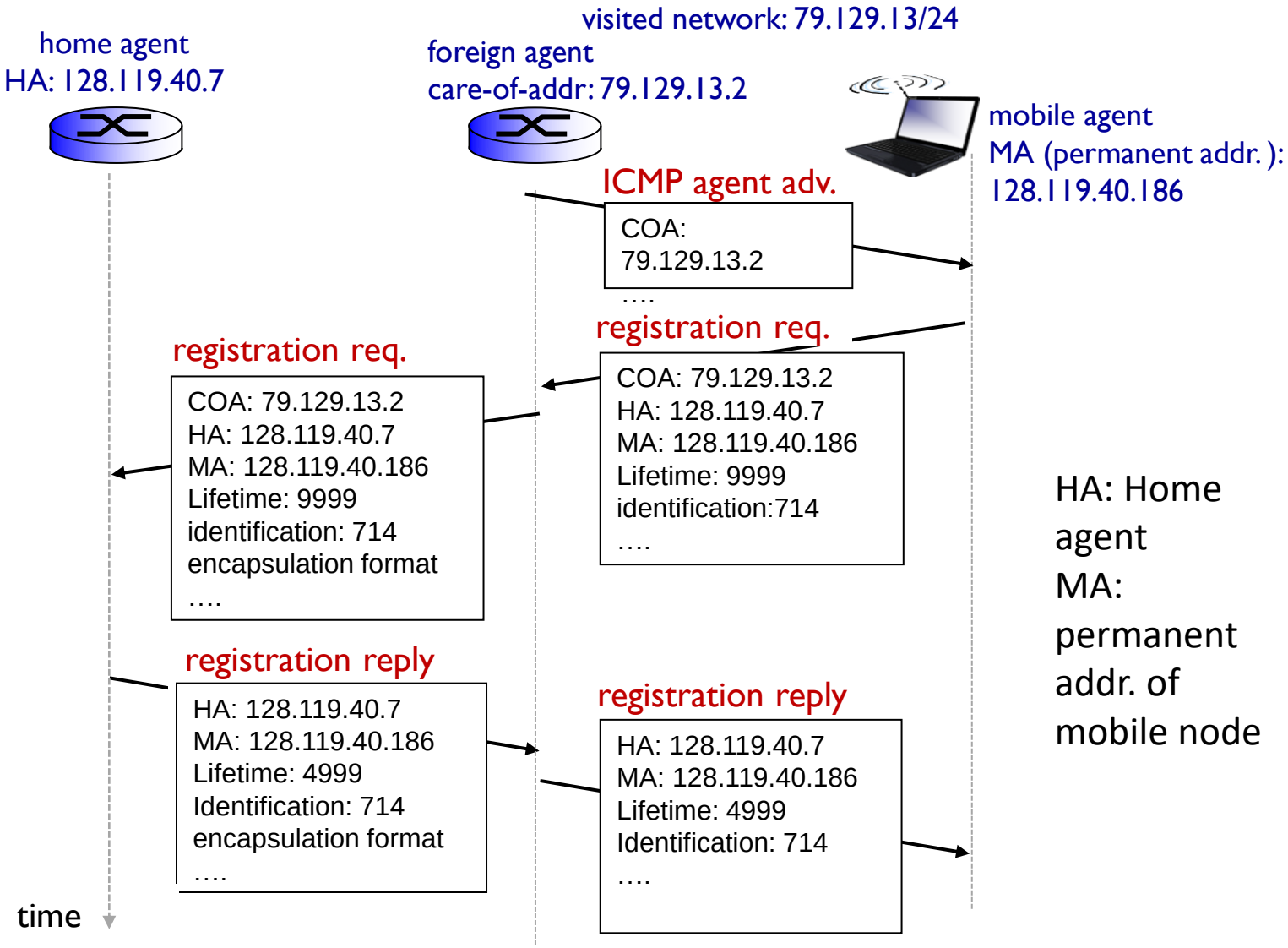
- Correspondent learns mobile node's COA
- Correspondent tunnels to mobile node's COA
- overcome triangle routing problem
- *non-transparent to correspondent:* correspondent must get care-of-address from home agent
  - what if mobile changes visited network?



Indirect routing vs. direct routing

- Transparent to correspondent
  - Indirect: yes, only need to know permanent address
  - Direct: no, need to tunnel.
- Efficiency:
  - Indirect: low, triangle routing
  - Direct: high
- Mobility within session:
  - Indirect: Simple, just update home agent with new COA
  - Direct: Complicated

Mobile IP: registration example



Registration messages are in UPD, sent to port 434

HA: Home agent
MA: permanent addr. of mobile node

What is network security?

Confidentiality: only sender, intended receiver should "understand" message contents

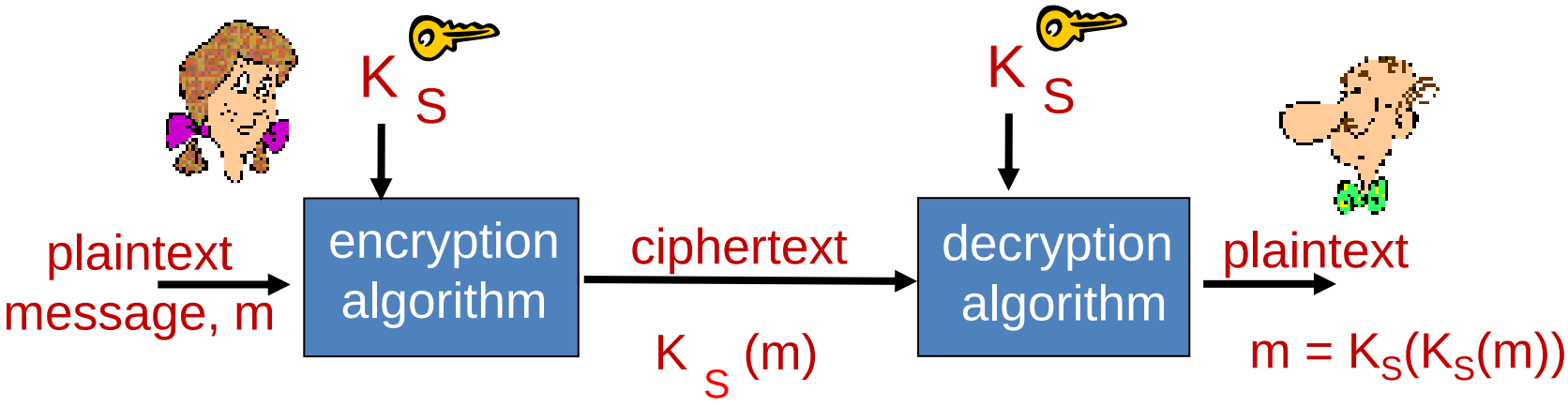
- sender encrypts message
- receiver decrypts message

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_S

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

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

Block cipher

- Message is encrypted in blocks of *k* bits.

k=3

input	output	input	output
000	110	100	011
001	111	101	010
010	101	110	000
011	100	111	001

- Large *k* values against brute-force attack
  - 2^k possible mapping
  - E.g, k=1024

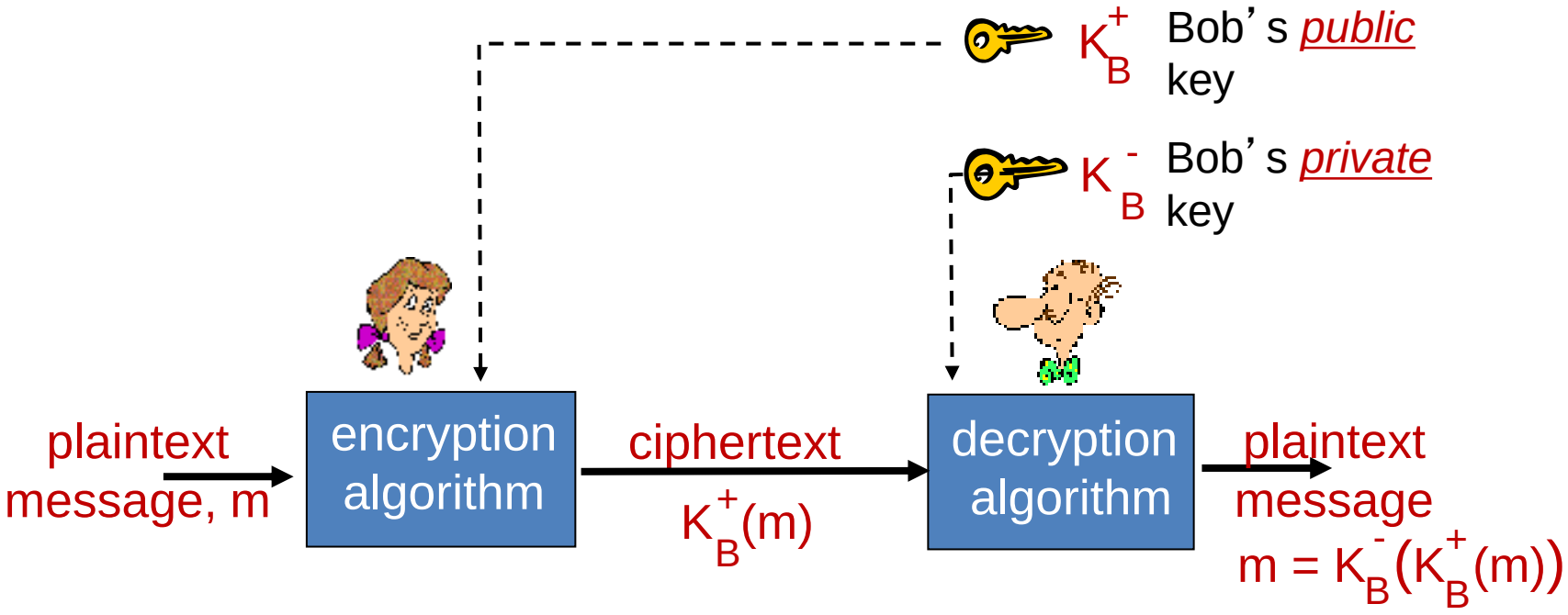
Cipher-block chaining

- The sender creates a random *k*-bit number r(i) for the *i*th block and calculates
  c(i) = $K_S(m(i) \oplus r(i))$. ($\oplus$ means exclusive or （异或）)
  - A new *k*-bit random number is chosen for each block.
- Receiver receives c(i) and r(i), it can recover each block of the plaintext by computing $m(i)=K_S(c(i)) \oplus r(i)$.
- Example:
  - k=3, plain text 010 010 010
  - r(1)=001 r(2)=111 r(3)=100
  - Produce different ciphertext for same plaintext

Cipher-block chaining

- Problem: need to transmit *r*(i) from sender to receiver
  - Transmit twice as many bits as before
- Cipher-block chaining (CBC)
  - Sender generates a k-bit string, Initialization Vector (IV), denoted as c(0), sent IV to receiver.
  - Runs block cipher algorithm $c(1)=K_s(m(1)\oplus c(0))$, send c(1) to receiver.
  - $c(i)=K_s(m(i)\oplus c(i-1))$
  - Receiver: knows c(i-1), obtain $m(i)=K_s(c(i))\oplus c(i-1)$

Public key cryptography



RSA: Creating public/private key pair

1. choose two large prime numbers（质数） *p*, *q*. (e.g., 1024 bits each)
2. compute *n* = *pq*, *z* = *(p-1)(q-1)*
3. choose *e* (with *e*<*n*) that has no common factors with z (e, z are "relatively prime（互质）").
4. choose *d* such that *ed-1* is exactly divisible by *z*. (in other words: *ed* mod *z* = *1*).
5. *public* key is *(n,e)*. *private* key is *(n,d)*.

K_B^+ K_B^-

RSA: encryption, decryption

0. given (n,e) and (n,d) as computed above

1. to encrypt message m ($<n$), compute

$c = m^e \bmod n$

2. to decrypt received bit pattern, c, compute

$m = c^d \bmod n$

magic happens!

$$m = \underbrace{(m^e \bmod n)}_{c}{}^d \bmod n$$

RSA: another important property

The following property will be *very* useful later:

$$K_B^-(K_B^+(m)) = m = K_B^+(K_B^-(m))$$

use public key first, followed by private key

use private key first, followed by public key

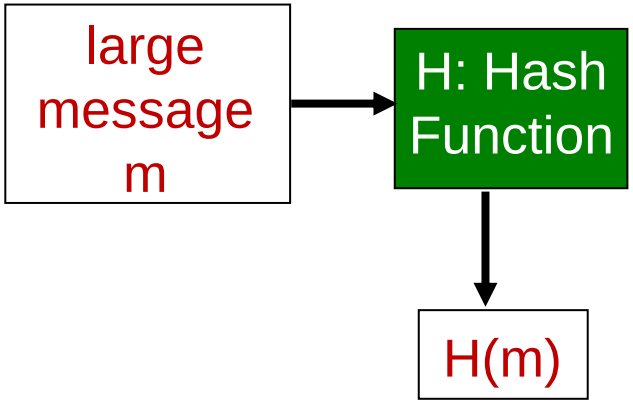
result is the same!

Message digests

computationally expensive to public-key-encrypt long messages

goal: fixed-length, easy- to-compute digital "fingerprint"

- apply hash function H to *m*, get fixed size message digest（报文摘要）, *H(m)*.



Hash function properties:

- many-to-1
- produces fixed-size msg digest (fingerprint)
- given message digest x, computationally infeasible to find m such that x = H(m)

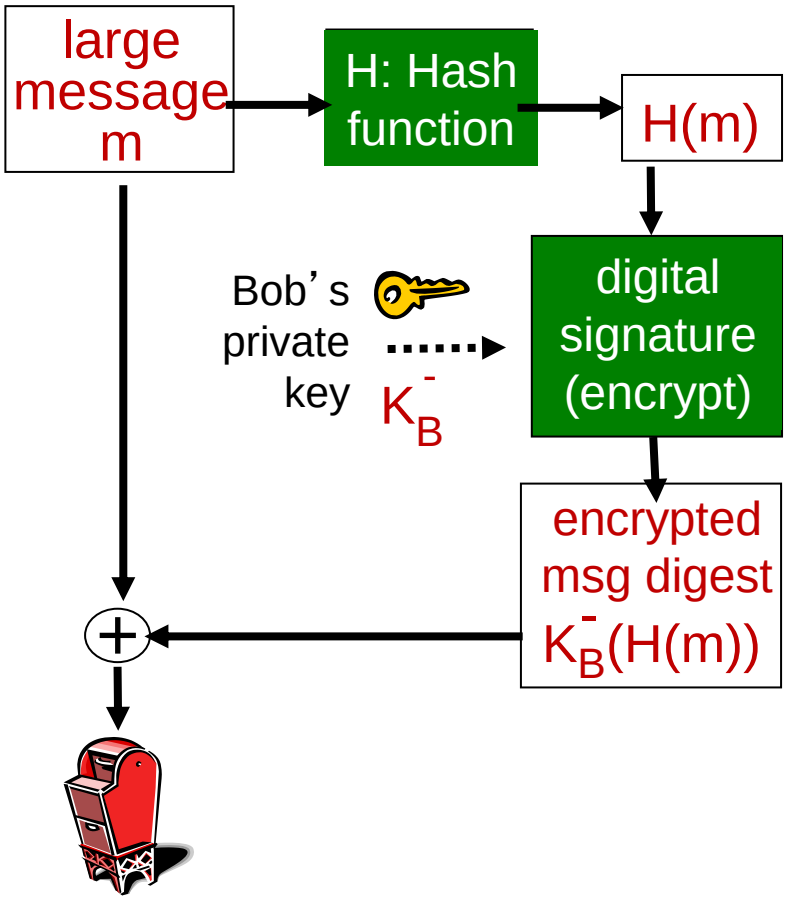
Message authentication code

- Alice and Bob will need a shared secret s, called the *authentication key*.

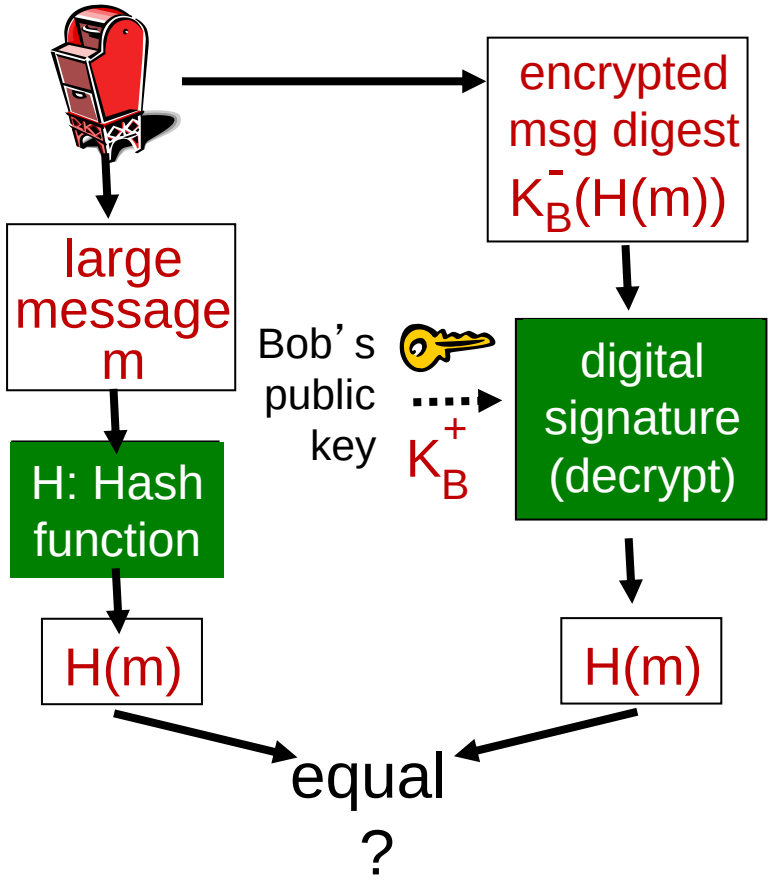
1. Alice concatenates s with m to create m + s, and calculates the hash H(m + s) (for example with SHA-1).
   - H(m + s) is called the message authentication code (MAC).
2. Alice then appends the MAC to the message m, creating an extended message (m, H(m + s)), and sends to Bob.
3. Bob receives an extended message (m, h) and knowing s, calculates the MAC H(m + s).
   - If H(m + s) = h, Bob concludes that everything is fine.

Digital signature = signed message digest

Bob sends digitally signed message:

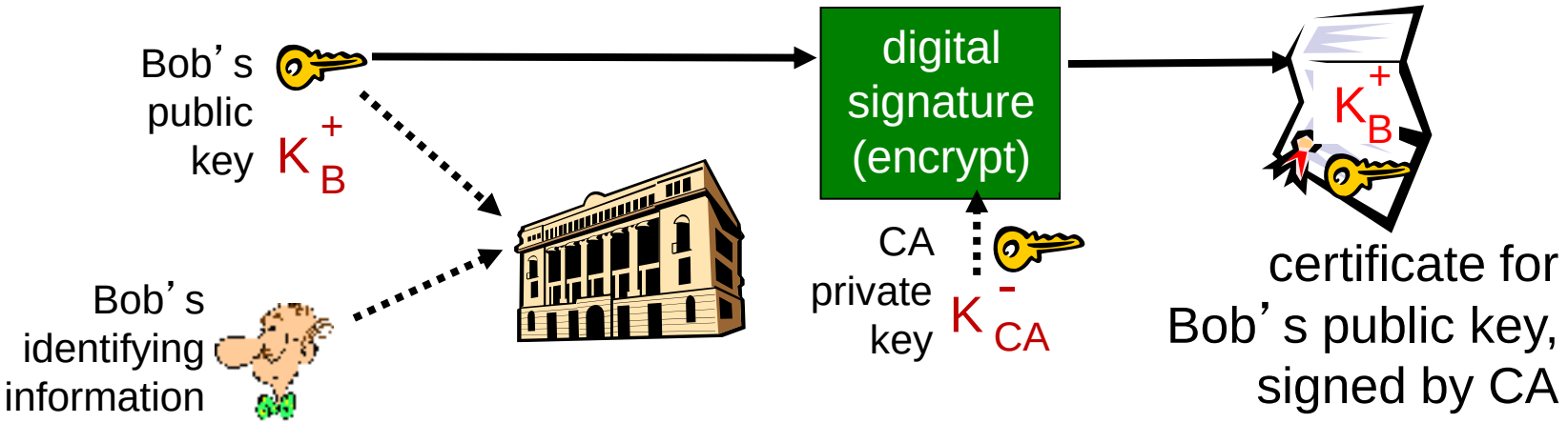


Alice verifies signature, integrity of digitally signed message:



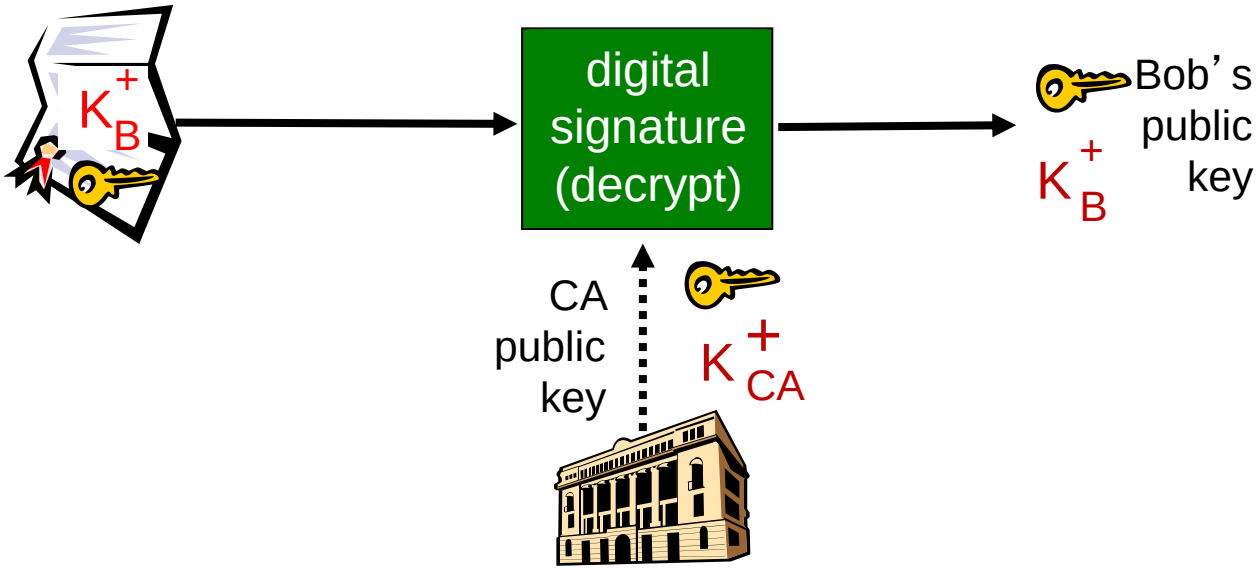
Certification authorities

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



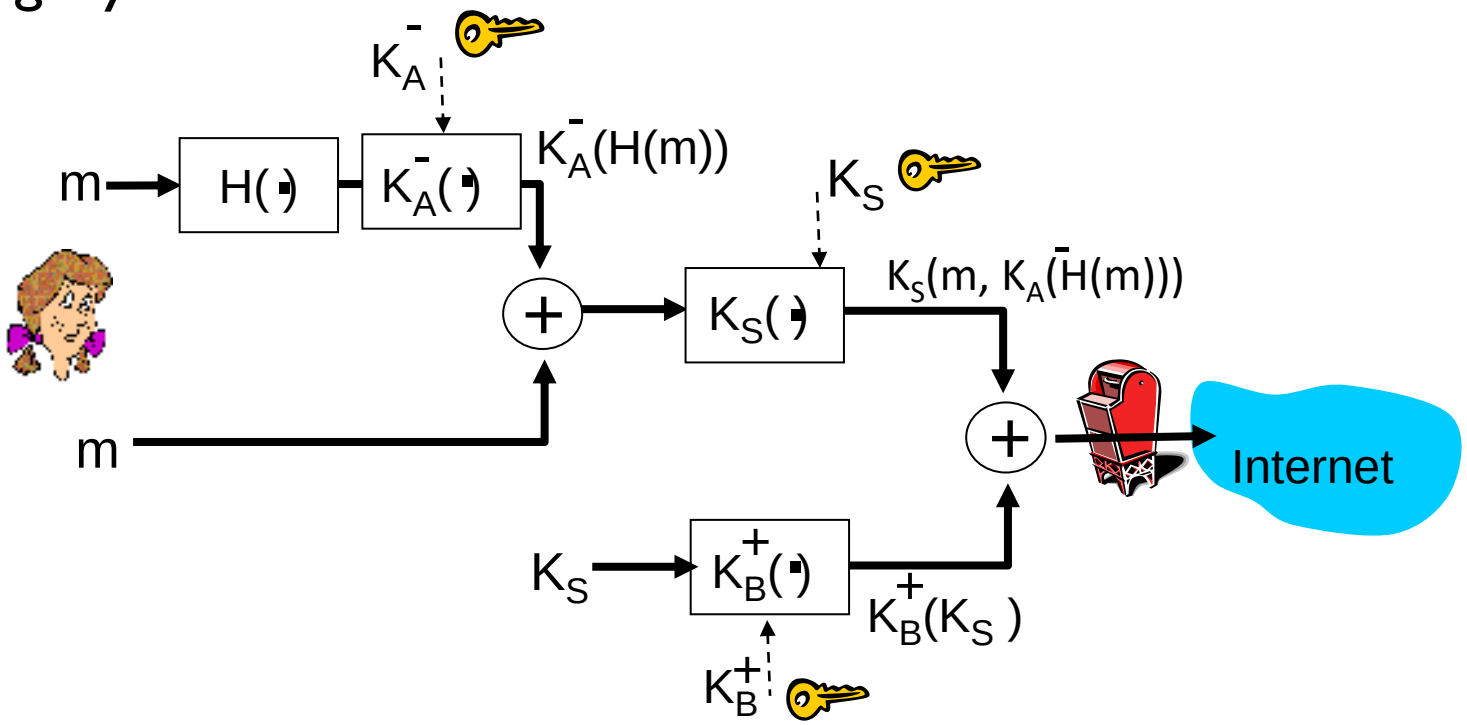
Certification authorities

- 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



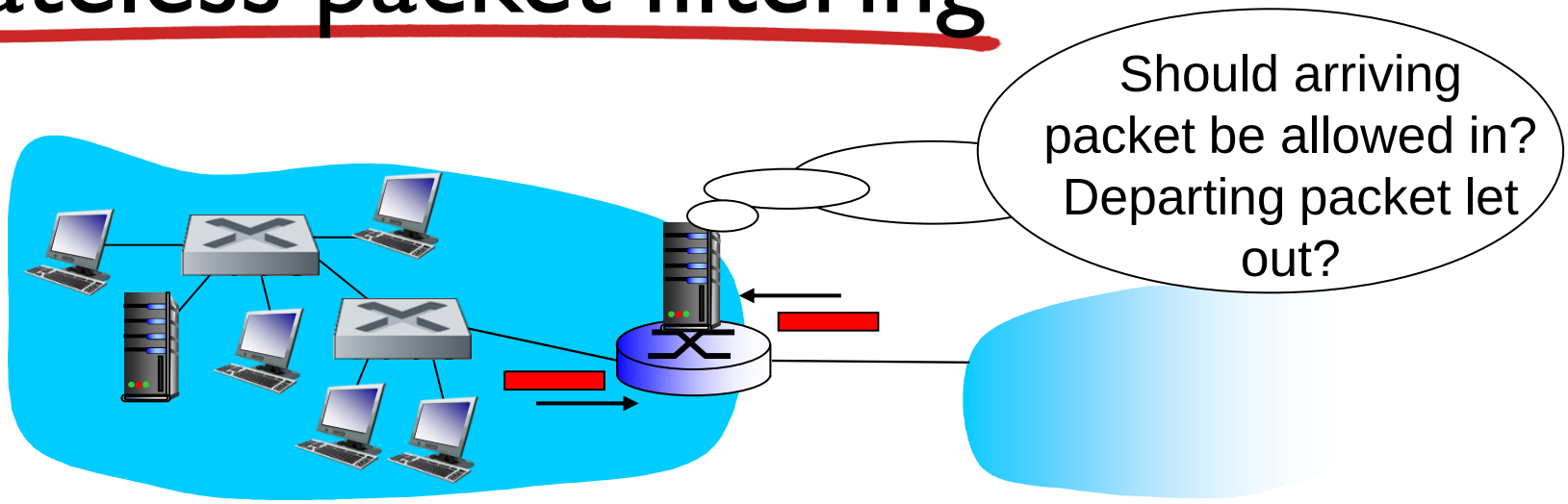
Secure e-mail (continued)

Alice wants to provide secrecy, sender authentication, message integrity.



Alice uses three keys: her private key, Bob's public key, newly created symmetric key

Stateless packet filtering



- internal network connected to Internet via *router firewall*
- router *filters packet-by-packet,* decision to forward/drop packet based on:
  - source IP address, destination IP address
  - TCP/UDP source and destination port numbers
  - ICMP message type
  - TCP SYN and ACK bits

Stateful packet filtering

- *stateless packet filter:* heavy handed tool
  - admits packets that "make no sense," e.g., dest port = 80, ACK bit set, even though no TCP connection established:

action	source address	dest address	protocol	source port	dest port	flag bit
allow	outside of 222.22/16	222.22/16	TCP	80	> 1023	ACK

- *stateful packet filter:* track status of every TCP connection
  - track connection setup (SYN), teardown (FIN): determine whether incoming, outgoing packets "makes sense"
  - timeout inactive connections at firewall: no longer admit packets