Chapter 4
Network Layer

Reti degli Elaboratori
Canale AL
Prof.ssa Chiara Petrioli
a.a. 2014/2015
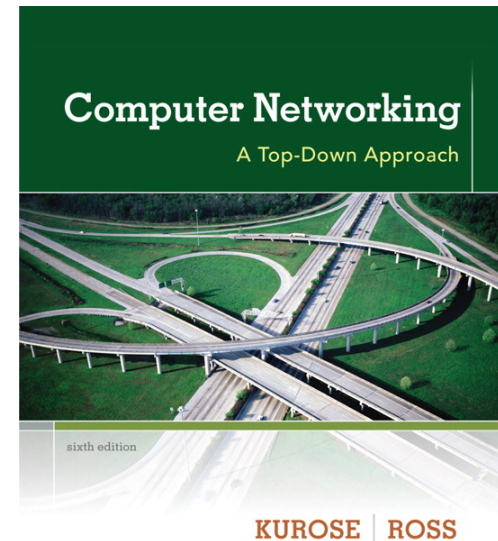
Computer Networking: A Top Down Approach
6th edition
Jim Kurose, Keith Ross
Addison-Wesley
March 2012

Chapter 4: outline

ICMP: internet control message protocol

❖ **used by hosts & routers to communicate network-level information**
  - error reporting: unreachable host, network, port, protocol
 - echo request/reply (used by ping)

❖ **network-layer "above" IP:**
  - ICMP msgs carried in IP datagrams

❖ **ICMP message:** type, code plus first 8 bytes of IP datagram causing error

Type	Code	description
0	0	echo reply (ping)
3	0	dest. network unreachable
3	1	dest host unreachable
3	2	dest protocol unreachable
3	3	dest port unreachable
3	6	dest network unknown
3	7	dest host unknown
4	0	source quench (congestion control - not used)
8	0	echo request (ping)
9	0	route advertisement
10	0	router discovery
11	0	TTL expired
12	0	bad IP header

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 nth 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

3 probes   3 probes

3 probes
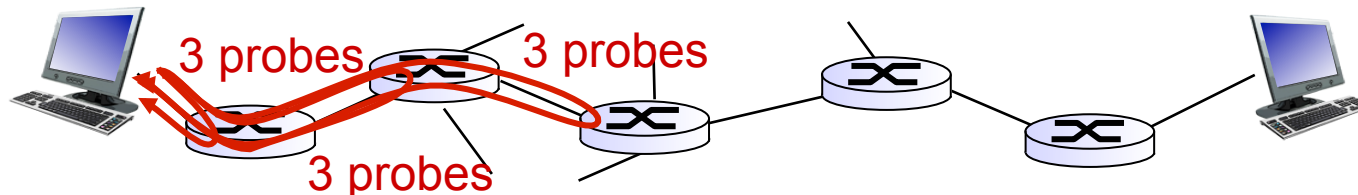
IPv6: motivation

❖ *initial motivation:* **32**-bit address space soon to be completely allocated.

❖ additional motivation:
  - header format helps speed processing/forwarding
  - header changes to facilitate QoS

IPv6 datagram format:
  - fixed-length 40 byte header
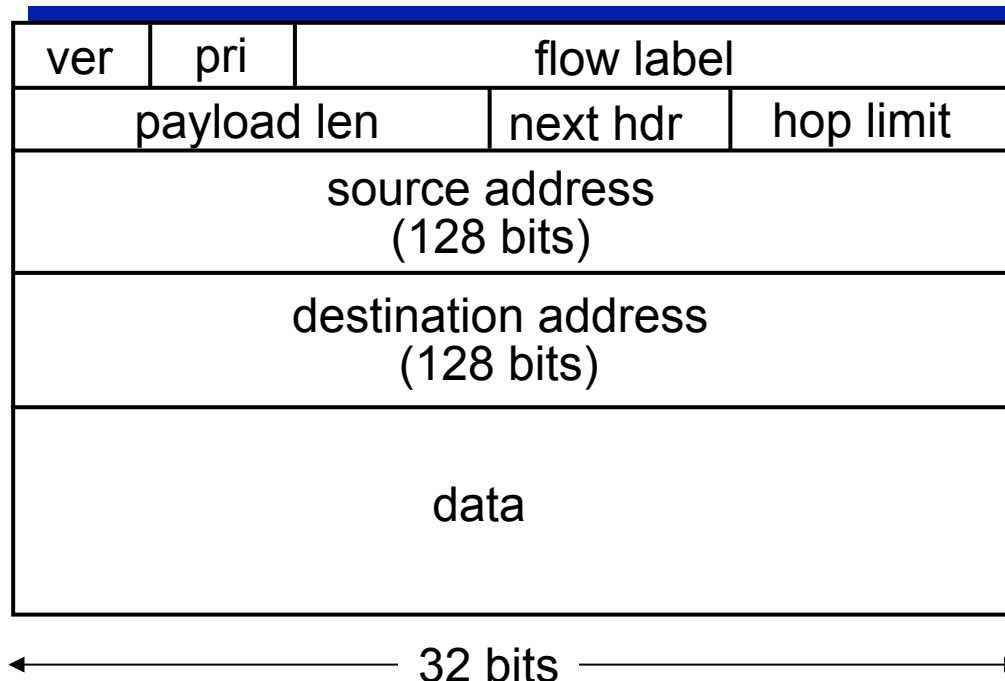 - no fragmentation allowed

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

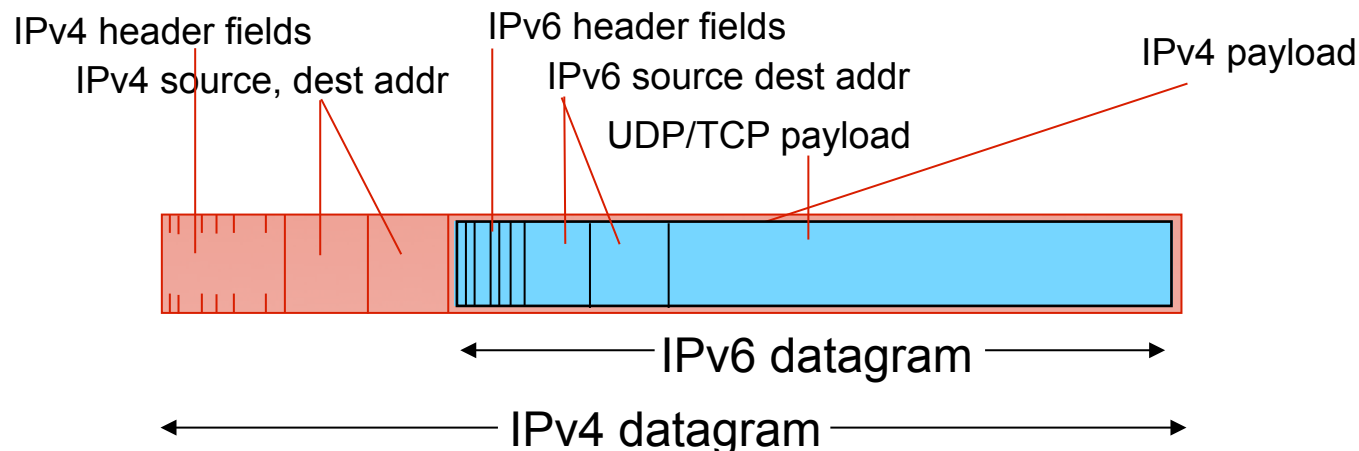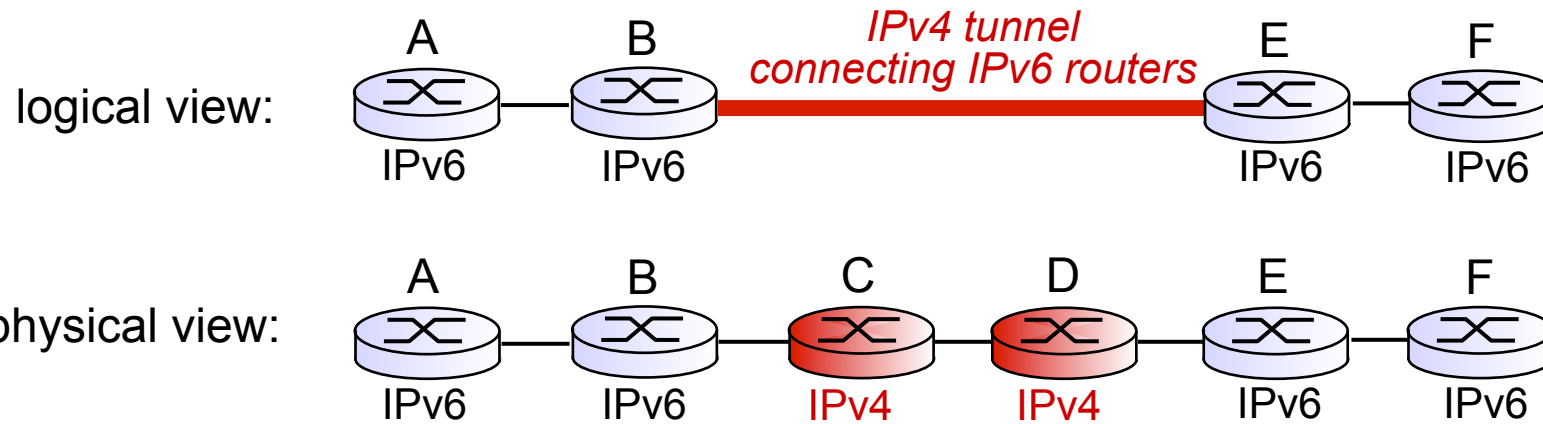| ver	pri	flow label		
payload len			next hdr	hop limit
source address (128 bits)				
destination address (128 bits)				
data				

← 32 bits →

Other changes from IPv4

- ❖ *checksum:* removed entirely to reduce processing time at each hop

- ❖ *options:* allowed, but outside of header, indicated by "Next Header" field

- ❖ *ICMPv6:* new version of ICMP
  - ▪ additional message types, e.g. "Packet Too Big"
  - ▪ multicast group management functions

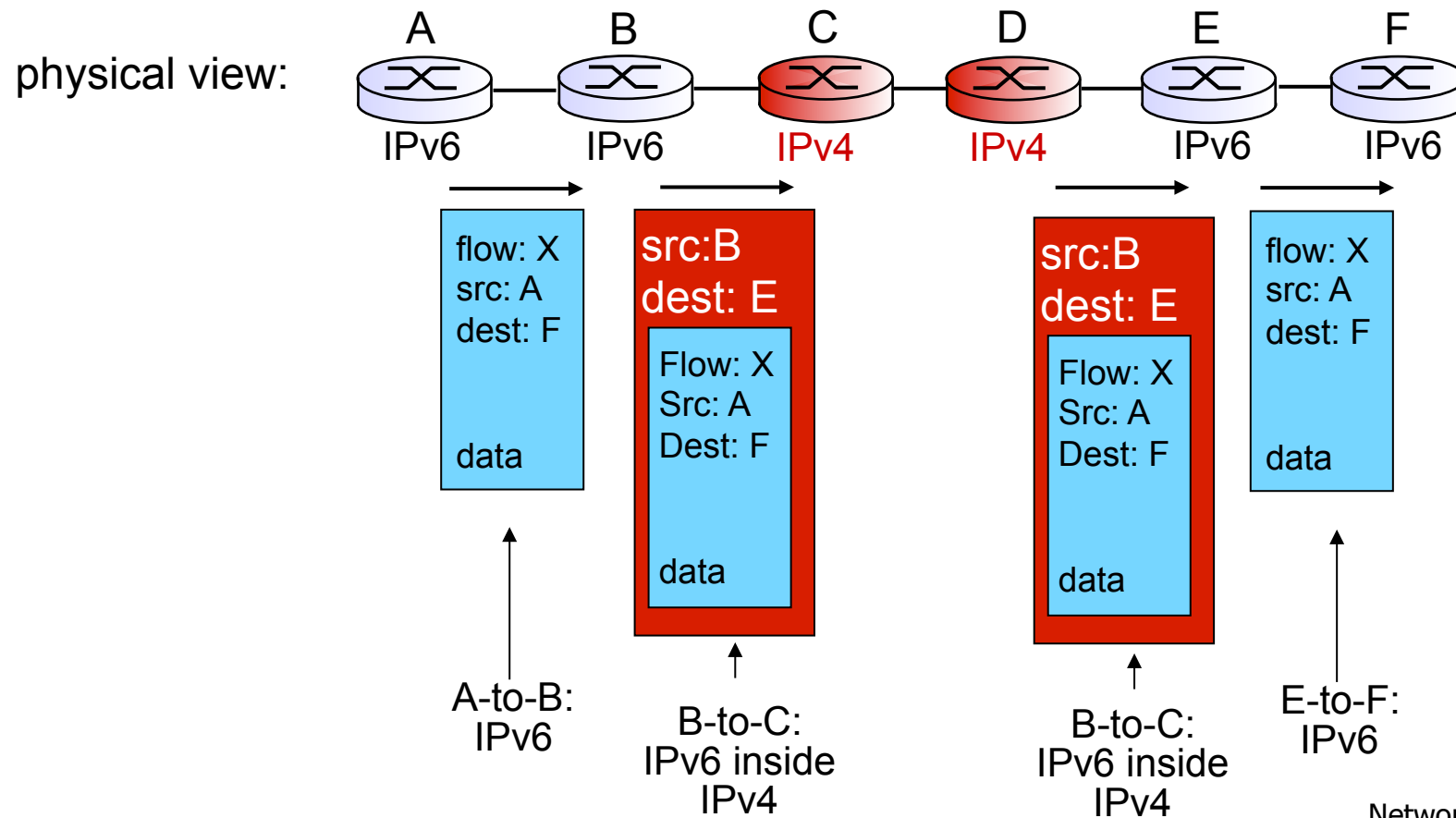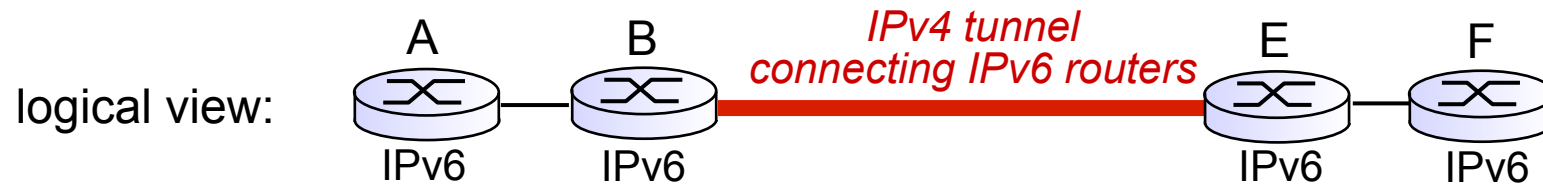# Transition from IPv4 to IPv6

❖ **not all routers can be upgraded simultaneously**
  - no "flag days"
  - how will network operate with mixed IPv4 and IPv6 routers?

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

IPv4 header fields
IPv4 source, dest addr

IPv6 header fields
IPv6 source dest addr
UDP/TCP payload

IPv4 payload

IPv6 datagram

IPv4 datagram

Tunneling

logical view:

A — B ----IPv4 tunnel connecting IPv6 routers---- E — F
IPv6   IPv6   IPv6   IPv6

physical view:

A — B — C — D — E — F
IPv6   IPv6   IPv4   IPv4   IPv6   IPv6

Tunneling

logical view:

A   B   *IPv4 tunnel connecting IPv6 routers*   E   F

IPv6   IPv6   IPv6   IPv6

physical view:

A   B   C   D   E   F

IPv6   IPv6   **IPv4**   **IPv4**   IPv6   IPv6

flow: X
src: A
dest: F

data

src:B
dest: E

Flow: X
Src: A
Dest: F

data

src:B
dest: E

Flow: X
Src: A
Dest: F

data

flow: X
src: A
dest: F

data

A-to-B:
IPv6

B-to-C:
IPv6 inside
IPv4

B-to-C:
IPv6 inside
IPv4

E-to-F:
IPv6

Chapter 4: outline

4.1 introduction

4.2 virtual circuit and datagram networks

4.3 what's inside a router

4.4 IP: Internet Protocol
- datagram format
- IPv4 addressing
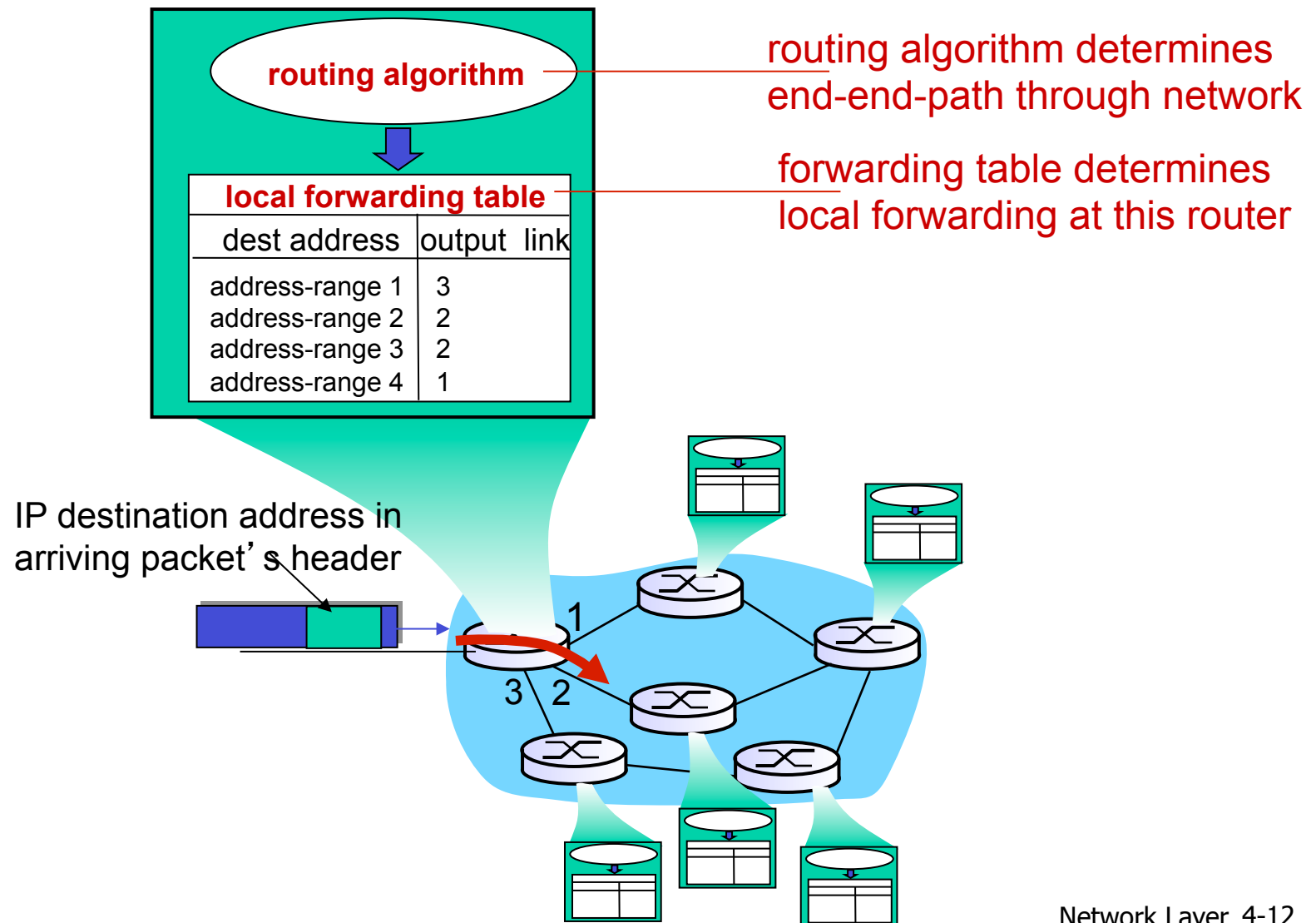- ICMP
- IPv6

4.5 routing algorithms
- link state
- distance vector
- hierarchical routing
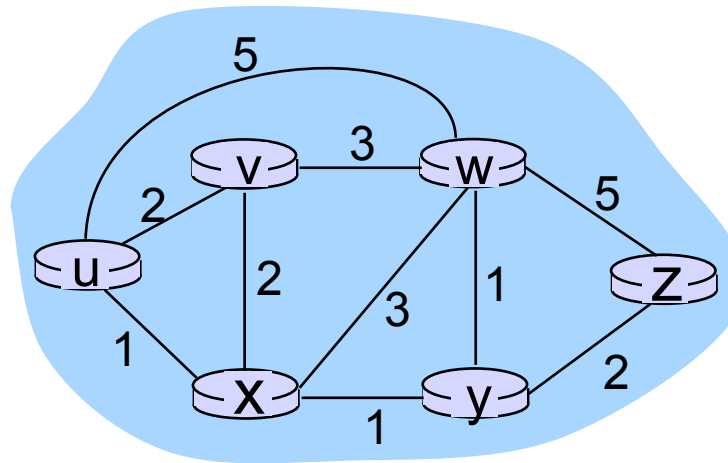
4.6 routing in the Internet
- RIP
- OSPF
- BGP

4.7 broadcast and multicast routing

Interplay between routing, forwarding

routing algorithm

routing algorithm determines
end-end-path through network

local forwarding table

forwarding table determines
local forwarding at this router

dest address	output link
address-range 1	3
address-range 2	2
address-range 3	2
address-range 4	1

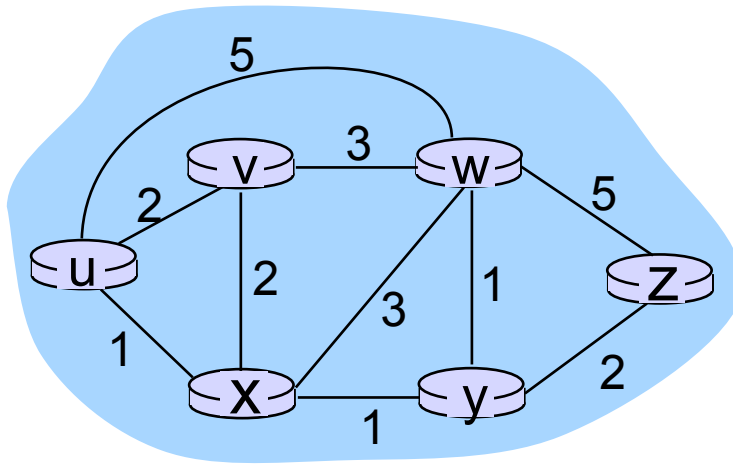IP destination address in
arriving packet's header

Graph abstraction

graph: G = (N,E)

N = set of routers = { u, v, w, x, y, z }

E = set of links ={ (u,v), (u,x), (v,x), (v,w), (x,w), (x,y), (w,y), (w,z), (y,z) }

aside: graph abstraction is useful in other network contexts, e.g., P2P, where *N* is set of peers and *E* is set of TCP connections

Graph abstraction: costs

c(x,x') = cost of link (x,x')
    e.g., c(w,z) = 5

cost could always be 1, or inversely related to bandwidth, or inversely related to congestion

cost of path $(x_1, x_2, x_3,\ldots, x_p)$ = $c(x_1,x_2) + c(x_2,x_3) + \ldots + c(x_{p-1},x_p)$

key question: what is the least-cost path between u and z ?
routing algorithm: algorithm that finds that least cost path

Routing algorithm classification

Q: global or decentralized information?

global:

❖ all routers have complete topology, link cost info

❖ "link state" algorithms

decentralized:

❖ router knows physically-connected neighbors, link costs to neighbors

❖ iterative process of computation, exchange of info with neighbors

❖ "distance vector" algorithms

Q: static or dynamic?

static:

❖ routes change slowly over time

dynamic:

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

Chapter 4: outline

4.1 introduction

4.2 virtual circuit and
    datagram networks

4.3 what's inside a router

4.4 IP: Internet Protocol
- datagram format
- IPv4 addressing
- ICMP
- IPv6

4.5 routing algorithms
- link state
- distance vector
- hierarchical routing

4.6 routing in the Internet
- RIP
- OSPF
- BGP

4.7 broadcast and multicast
    routing

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 dest.'s

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

```
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'
```
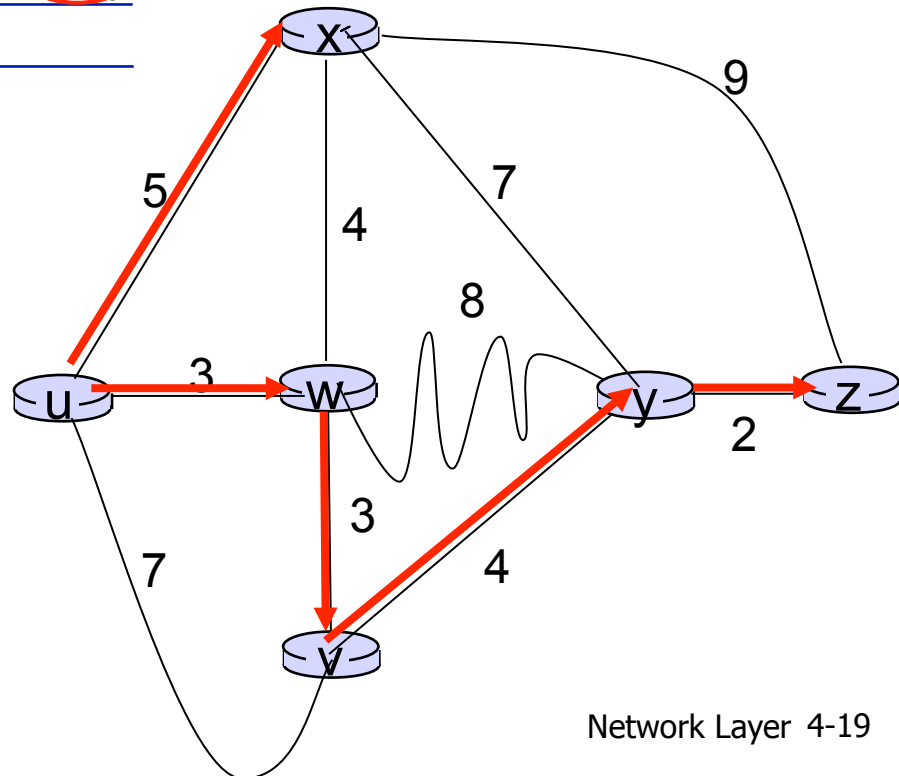
Dijkstra's algorithm: example

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					

notes:

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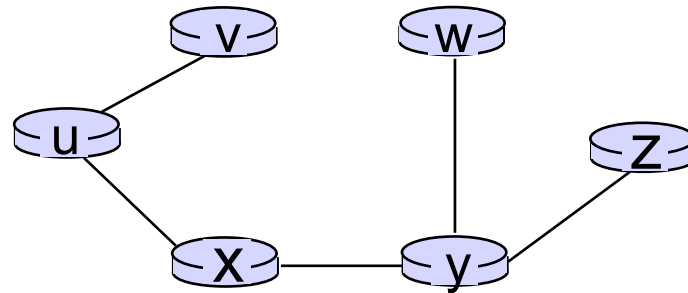
❖ ties can exist (can be broken arbitrarily)

Dijkstra's algorithm: another example

Step	N'	D(v),p(v)	D(w),p(w)	D(x),p(x)	D(y),p(y)	D(z),p(z)
0	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					
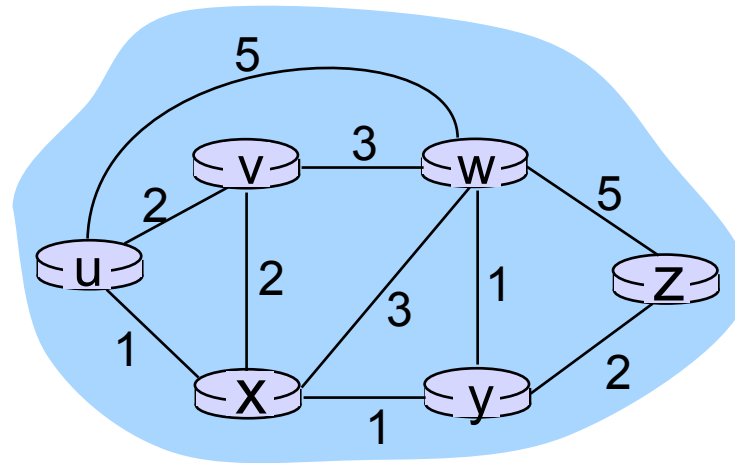
Dijkstra's algorithm: example (2)

resulting shortest-path tree from u:

resulting forwarding table in u:

destination	link
v	(u,v)
x	(u,x)
y	(u,x)
w	(u,x)
z	(u,x)

Correttezza

Se eseguiamo l'algoritmo di Dijkstra su un grafo pesato diretto G=(N,E) con pesi non negativi, sorgente u, e funzione peso c allora alla terminazione D(v)=δ(u,v), per ogni nodo v in N. (dove δ(u,v) indica la lunghezza del cammino di peso minimo tra u e v).

Dim.

D(v) non è più aggiornato nel momento in cui v è inserito in N'. Dovremmo quindi mostrare che D(v)=δ(u,v) nel momento in cui v è inserito in N', per ogni v. Ragioniamo per assurdo. Sia x il primo nodo (nell'ordine di inserimento in N') per cui vale D(x)!=δ(u,x) al momento in cui x è inserito nell'insieme N' (linea 10 dell'algoritmo). x!=u dato che u, nodo sorgente, è inserito nella fase di inizializzazione e per lui vale D(u)=δ(u,v)=0. Inoltre deve esistere un percorso di costo non infinito da u a x dato che altrimenti varrebbe che il valore a cui D(x) è inizializzato (infinito) sarebbe uguale a δ(u,v). Quindi esiste un percorso di costo minimo p=u...v$\rightarrow$y...x dove y è il primo nodo sul percorso di costo minimo NON in N' (quindi u....v sono TUTTI in N'). Il percorso p può quindi essere diviso in due percorsi: p1 che va da u a y e p2 che va da y a x.

Da notare che il percorso p1 è anch'esso il percorso di costo minimo che unisce u a y (se non lo fosse e ci fosse un percorso p3 che unisce u a y di costo < del costo di p1, allora la concatenazione di p2 e p2 sarebbe un percorso p' da u a x di costo < di p,contro l'assunto che p sia un percorso di costo minimo).

…Correttezza

Se eseguiamo l'algoritmo di Dijkstra su un grafo pesato diretto G=(N,E) con pesi non negativi, sorgente u, e funzione peso c allora alla terminazione D(v)=δ(u,v), per ogni nodo v in N. (dove δ(u,v) indica la lunghezza del cammino di peso minimo tra u e v).

Dim (…continua).

Quando x è inserito in N' D(y)=δ(u,y). Ifnatti in quel momento v è stato già inserito in N' e dopo il suo inserimento y ha ricalcolato D(y)=D(v)+c(v,y)=δ(u,v)+c(v,y) (dato che per ipotesi x è il primo nodo per cui all'inserimento in N' la stima dei costi non corrisponde al percorso di costo minimo)=δ(u,y).

Dato che y precede x sul percorso minimo ed i pesi sugli archi sono non negativi vale che:

δ(u,x)>=δ(u,y)=D(y)

e quindi anche che

D(x)>=δ(u,x)>=δ(u,y)=D(y)

D'altro canto dato che x viene inserito in N' prima di y vale che

δ(u,x)<=D(x)<=D(y)=δ(u,y)

Quindi

δ(u,x)=D(x)=D(y)=δ(u,y)
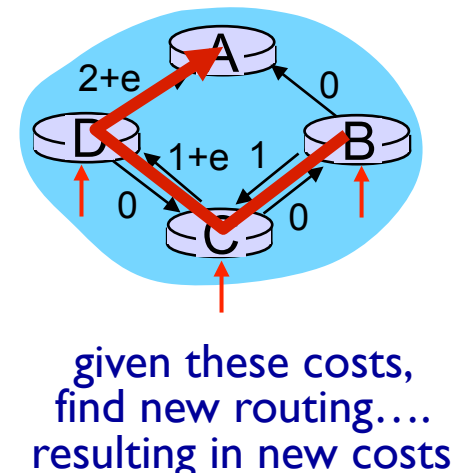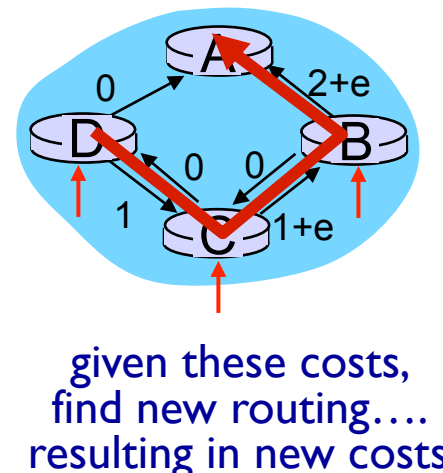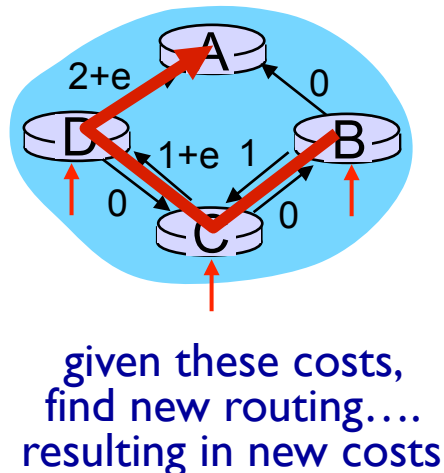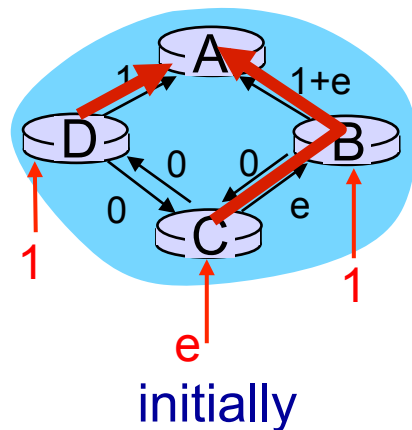
Cosa che porta alla contraddizione.

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: $O(n\log n)$

oscillations possible:

❖ e.g., support link cost equals amount of carried traffic:

initially

given these costs,
find new routing….
resulting in new costs

given these costs,
find new routing….
resulting in new costs

given these costs,
find new routing….
resulting in new costs

Chapter 4: outline

4.1 introduction

4.2 virtual circuit and datagram networks

4.3 what's inside a router

4.4 IP: Internet Protocol
  - datagram format
  - IPv4 addressing
  - ICMP
  - IPv6

4.5 routing algorithms
  - link state
  - distance vector
  - hierarchical routing

4.6 routing in the Internet
  - RIP
  - OSPF
  - BGP

4.7 broadcast and multicast routing

Bellman-Ford

Given a graph G=(N,E) and a node *s* finds the shortest path from *s* to every node in N.

A shortest walk from *s* to i subject to the constraint that the walk contains at most h arcs and goes through node *s* only once, is denoted shortest(<=h) walk and its length is D^h_i.

Bellman-Ford rule:

Initiatilization D^h_s=0, for all h; $c_{i,k}$ = infinity if (i,k) NOT in E; $c_{k,k}$ =0; **D^0_i=infinity for all i!=s.**

Iteration:

$$D^{h+1}_i = \min_k [c_{i,k} + D^h_k]$$

Assumption: non negative cycles **(this is the case in a network!!)**

The Bellman-Ford algorithm first finds the one-arc shortest walk lengths, then the two-arc shortest walk length, then the three-arc…etc. →distributed version used for routing

Distance vector algorithm

Bellman-Ford equation (dynamic programming)

let
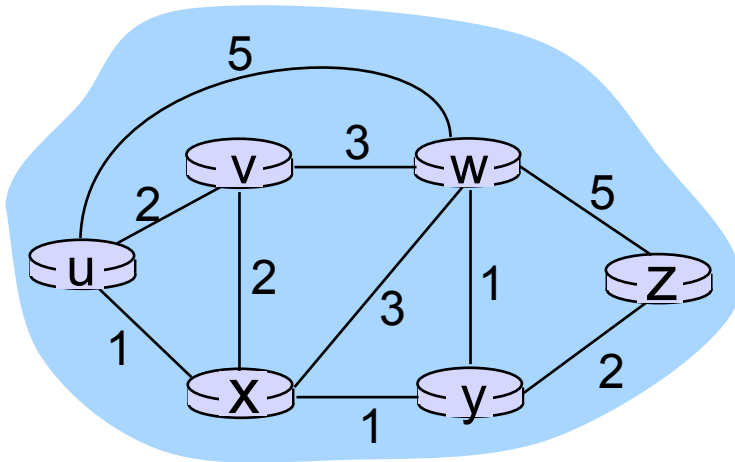 $d_x(y) :=$ cost of least-cost path from x to y
then
$$d_x(y) = \min_v \{c(x,v) + d_v(y)\}$$

cost from neighbor v to destination y

cost to neighbor v

min taken over all neighbors v of x

Bellman-Ford example

clearly, $d_v(z) = 5$, $d_x(z) = 3$, $d_w(z) = 3$

B-F equation says:

$$d_u(z) = \min \{ c(u,v) + d_v(z),$$
$$c(u,x) + d_x(z),$$
$$c(u,w) + d_w(z) \}$$
$$= \min \{2 + 5,$$
$$1 + 3,$$
$$5 + 3\} \; = 4$$

node achieving minimum is next
hop in shortest path, used in forwarding table

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:

At all nodes, X:

1 Initialization:
2   for all adjacent nodes v:
3   D^X(*,v) = infinity   /* the * operator means "for all rows" */
4   D^X(v,v) = c(X,v)
5   for all destinations, y
6   send $min_w D^X$(y,w) to each neighbor /* w over all X's neighbors */

From the node to whatever destination going through v

Distance Vector Algorithm (cont.):

```
8  loop
9    wait (until I see a link cost change to neighbor V
10        or until I receive update from neighbor V)
11
12   if (c(X,V) changes by d)
13     /* change cost to all dest's via neighbor v by d */
14     /* note: d could be positive or negative */
15     for all destinations y:  D^X(y,V) =  D^X(y,V) + d
16
17   else if (update received from V wrt destination Y)
18     /* shortest path from V to some Y has changed  */
19     /* V has sent a new value for its  min_w DV(Y,w) */
20     /* call this received new value is "newval"     */
21     for the single destination y: D^X(Y,V) = c(X,V) + newval
22
23   if we have a new min_w D^X(Y,w) for any destination Y
24     send new value of min_w D^X(Y,w) = of D_x(Y) to all neighbors
25
26 forever
```

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* $d_x(y)$
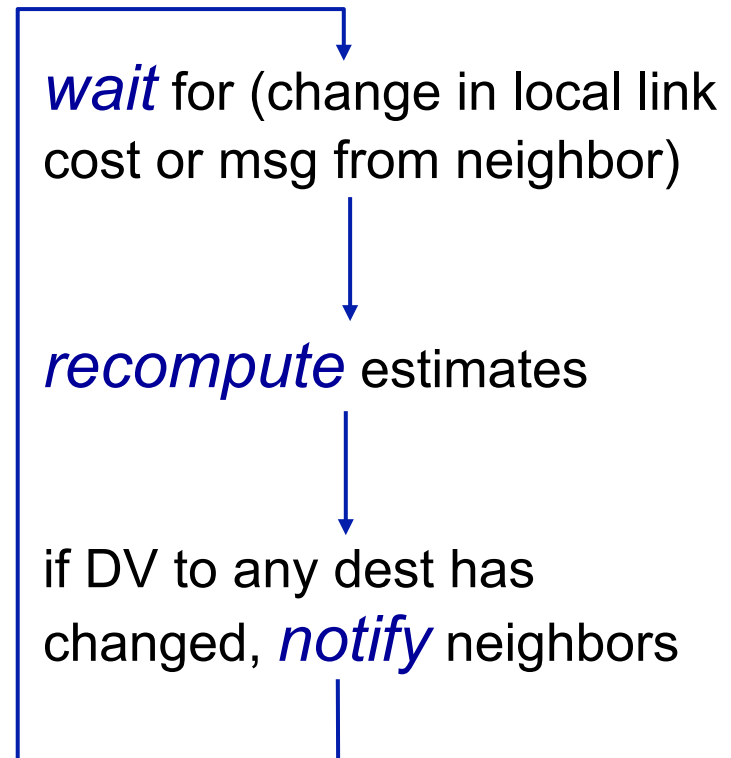
Distance vector algorithm

iterative, asynchronous:
each local iteration caused by:

- local link cost change
- DV update message from neighbor

distributed:

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

each node:

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

↓

recompute estimates

↓

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

Previous lecture. Summary:

Distributed Belman Ford

- Based on Distributed Bellman Ford Equation

$$D^X(Y,Z) = \text{distance from X to Y, via Z as next hop}$$

$$= c(X,Z) + \min_w \{D^Z(Y,w)\}$$

- $D^x(Y,Z)$ re-computed:
  - Upon reception of updates from the neighbors
  - Upon link cost change
- $\min_z D^X(Y,Z)$ communicated to the neighbors whenever its value changes, or periodically
- How long does it take for the algorithm to converge? 'good news travel fast, bad news may not→count to infinity'

Distance Vector: link cost changes

Subtitle: Distributed Bellman Ford converges *but* how fast?

Link cost changes:

❒ node detects local link cost change

❒ updates distance table (line 15)

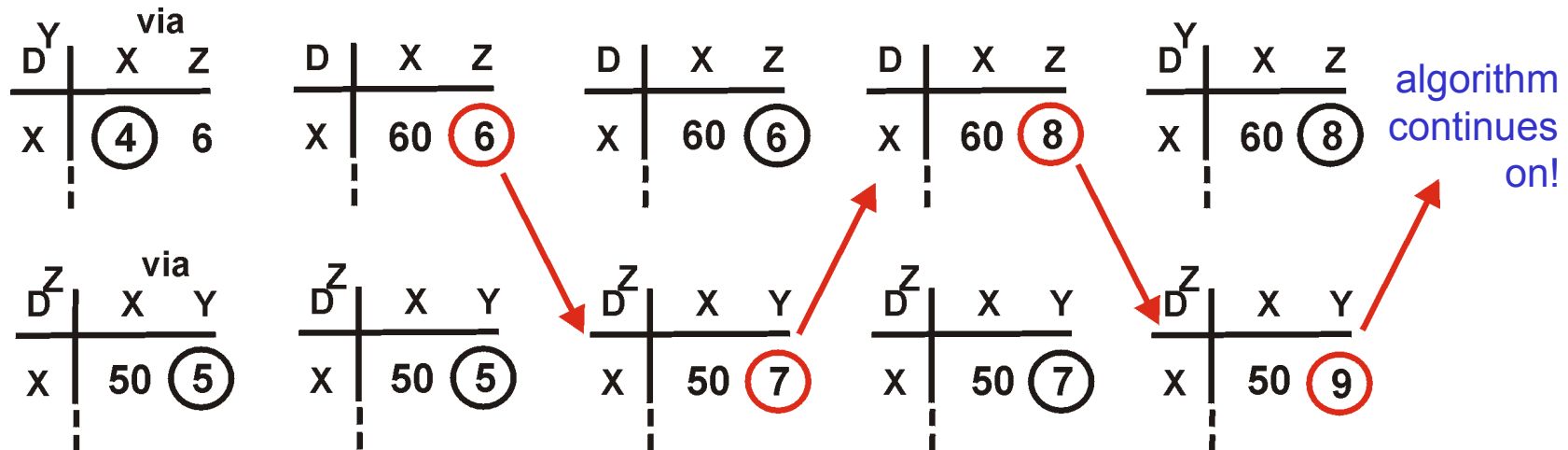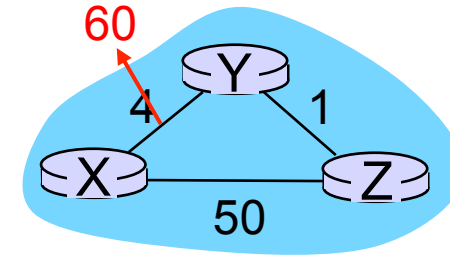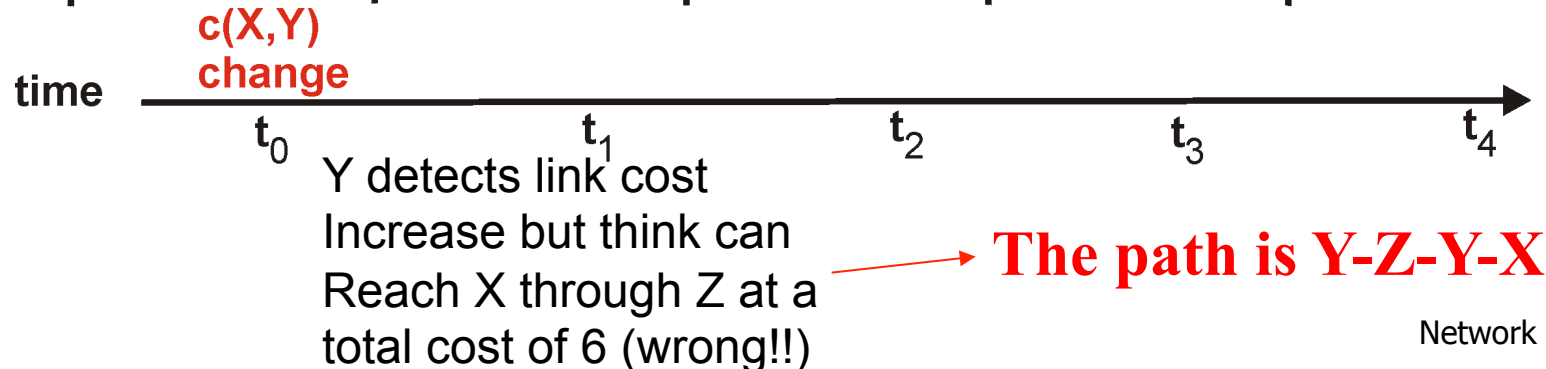❒ if cost change in least cost path, notify neighbors (lines 23,24)

"good news travels fast"

Y D	via X	Z
X	(4)	6

Y D	X	Z
X	(1)	6

Y D	X	Z
X	(1)	6

Y D	X	Z
X	(1)	3

algorithm terminates

Z D	·via X	Y
X	50	(5)

Z D	X	Y
X	50	(5)

Z D	X	Y
X	50	(2)

Z D	X	Y
X	50	(2)

**c(X,Y)
change**

time ——— t_0 ——————— t_1 ——————— t_2 ———

Ydetects change
informs neighbors

Z receives update
updates table, new least
Cost→informs neighbors
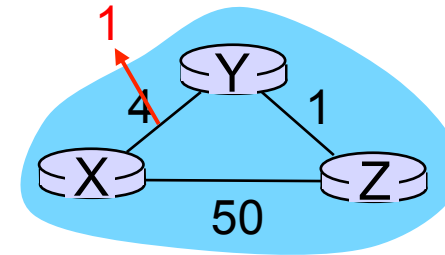
Y receives new least
cost; no modification
in the routing table,
No updates sent
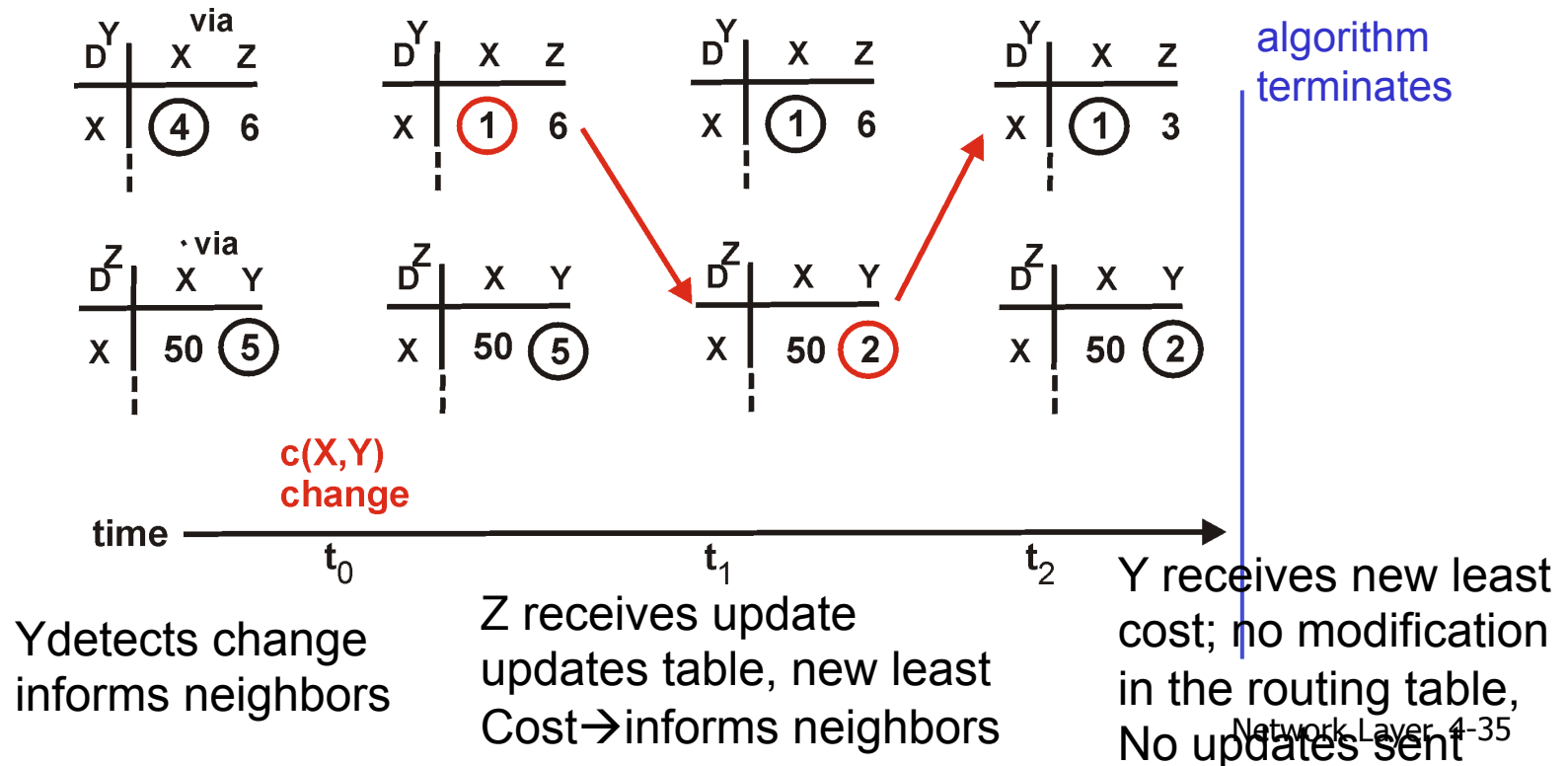
Distance Vector: link cost changes

Link cost changes:

☐ good news travels fast

☐ *bad news travels slow* - "count to infinity" problem!

Y	via	
D	X	Z
X	④	6

D	X	Z
X	60	⑥

D	X	Z
X	60	⑥

D	X	Z
X	60	⑧

Y		
D	X	Z
X	60	⑧

algorithm continues on!

Z	via	
D	X	Y
X	50	⑤

Z		
D	X	Y
X	50	⑤

Z		
D	X	Y
X	50	⑦

Z		
D	X	Y
X	50	⑦

Z		
D	X	Y
X	50	⑨

$c(X,Y)$ change

time

t_0 t_1 t_2 t_3 t_4

Y detects link cost
Increase but think can
Reach X through Z at a
total cost of 6 (wrong!!)

The path is Y-Z-Y-X

Count-to-infinity –an everyday life example

Which is the problem here?
the info exchanged by the protocol!! 'the best route to X I have has the following cost…' (no additional info on the route)

A Roman example…
-assumption: there is only one route going from Colosseo to Altare della Patria: Via dei Fori Imperiali. Let us now consider a network, whose nodes are Colosseo., Altare della Patria, Piazza del Popolo

Count-to-infinity –everyday life example (2/2)

Colosseo —1Km— Al.Patria —1Km— P.Popolo

The Colosseo. and Alt. Patria nodes exchange the following info
• Colosseo says 'the shortest route from me to P. Popolo is 2 Km'
• Alt. Patria says 'the shortest path from me to P. Popolo is 1Km'

Based on this exchange from Colosseo you go to Al. Patria, and from there to Piazza del Popolo OK Now **due to the big dig they close Via del Corso (Al. Patria—P.Popolo)**

• Al. Patria thinks 'I have to find another route from me to P.Popolo. Look there is a route from Colosseo to P.Popolo that takes 2Km, I can be at Colosseo in 1Km → I have found a 3Km route from me to P.Popolo!!' Communicates the new cost to Colosseo that updates 'OK I can go to P.Popolo via Al. Patria in 4Km'
VERY WRONG!! Why is it so? I didn't know that the route from Colosseo to P.Popolo was going through Via del Corso from Al.Patria to P.Popolo (which is closed)!!

Distance Vector: poisoned reverse

If Z routes through Y to get to X :

❑ Z tells Y its (Z's) distance to X is infinite (so Y won't route to X via Z)

❑ will this completely solve count to infinity problem?
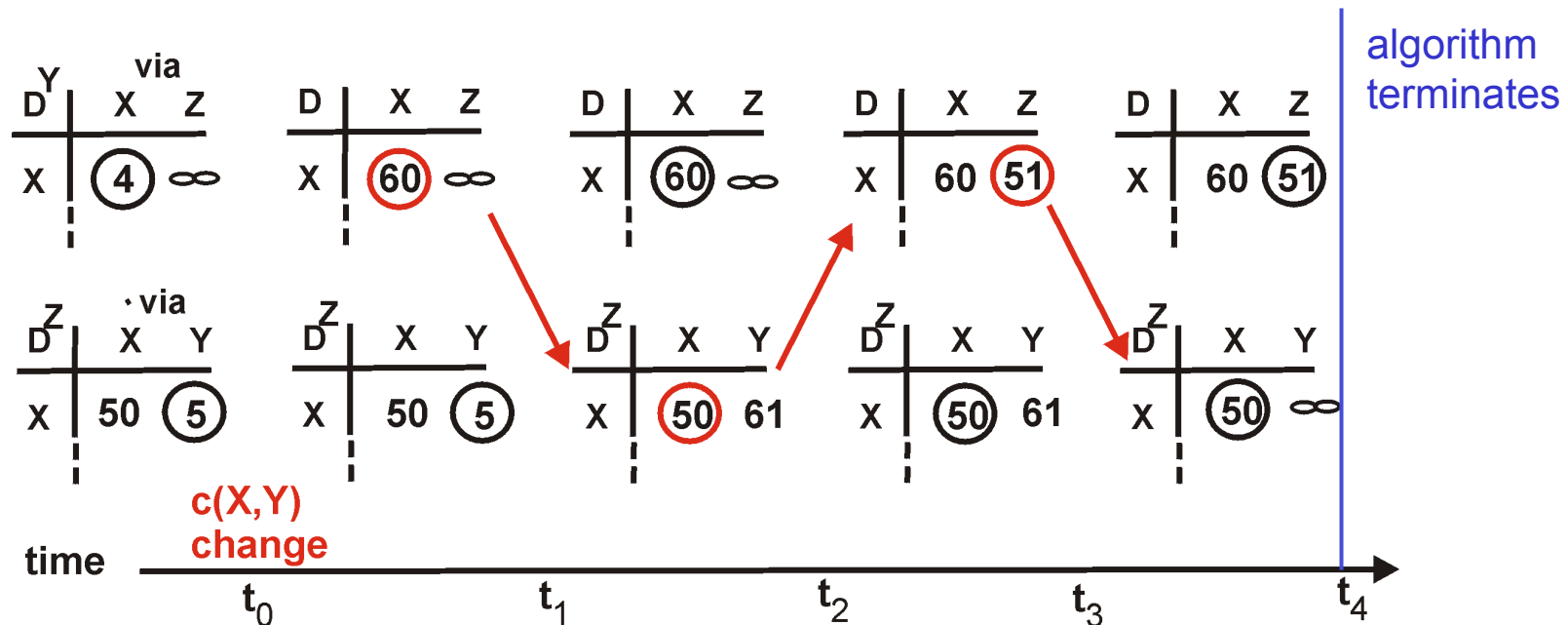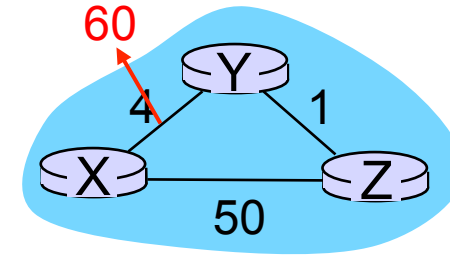
Infinity is advertized by Y (poisoned reverse)

Split horizon poison reverse failure

A —— B

C

D
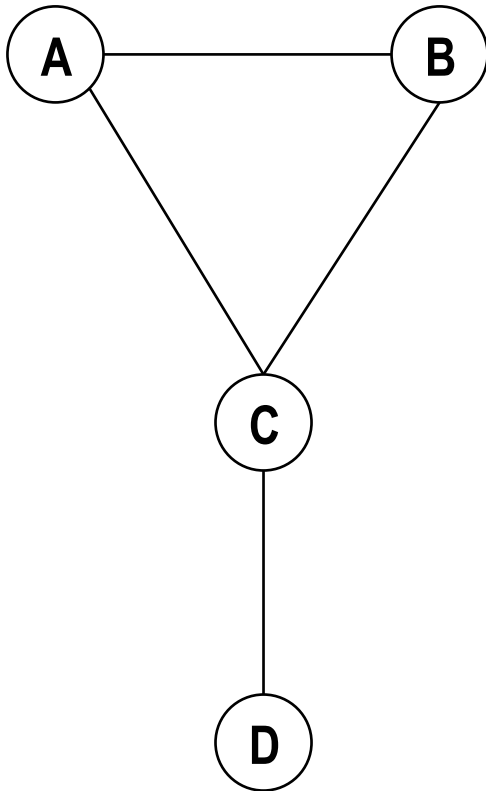
Line CD goes down…

1) because of split horizon rule, A and B tell C that dist(D)=inf

2) C concludes that D is unreachable and reports this to A and B

3) but A knows from B that dist(D)=2, and sets its dist=3

4) similarly, B knows from A distance from D… C estimates new value 4; A and B again through C estimate a value of 5….then again 1) … etc until distance = infinite

Regardless the hack used, there is always a network topology that makes the trick fail!

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(n²) 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

Chapter 4: outline

4.1 introduction

4.2 virtual circuit and
datagram networks

4.3 what's inside a router

4.4 IP: Internet Protocol
- datagram format
- IPv4 addressing
- ICMP
- IPv6

4.5 routing algorithms
- link state
- distance vector
- hierarchical routing

4.6 routing in the Internet
- RIP
- OSPF
- BGP

4.7 broadcast and multicast
routing

Hierarchical routing

our routing study thus far - idealization

❖ all routers identical

❖ network "flat"

… *not* true in practice

scale: with 600 million destinations:

❖ can't store all dest's in routing tables!

❖ routing table exchange would swamp links!

administrative autonomy

❖ internet = network of networks

❖ each network admin may want to control routing in its own network

Hierarchical routing

❖ **aggregate routers into regions, "autonomous systems" (AS)**

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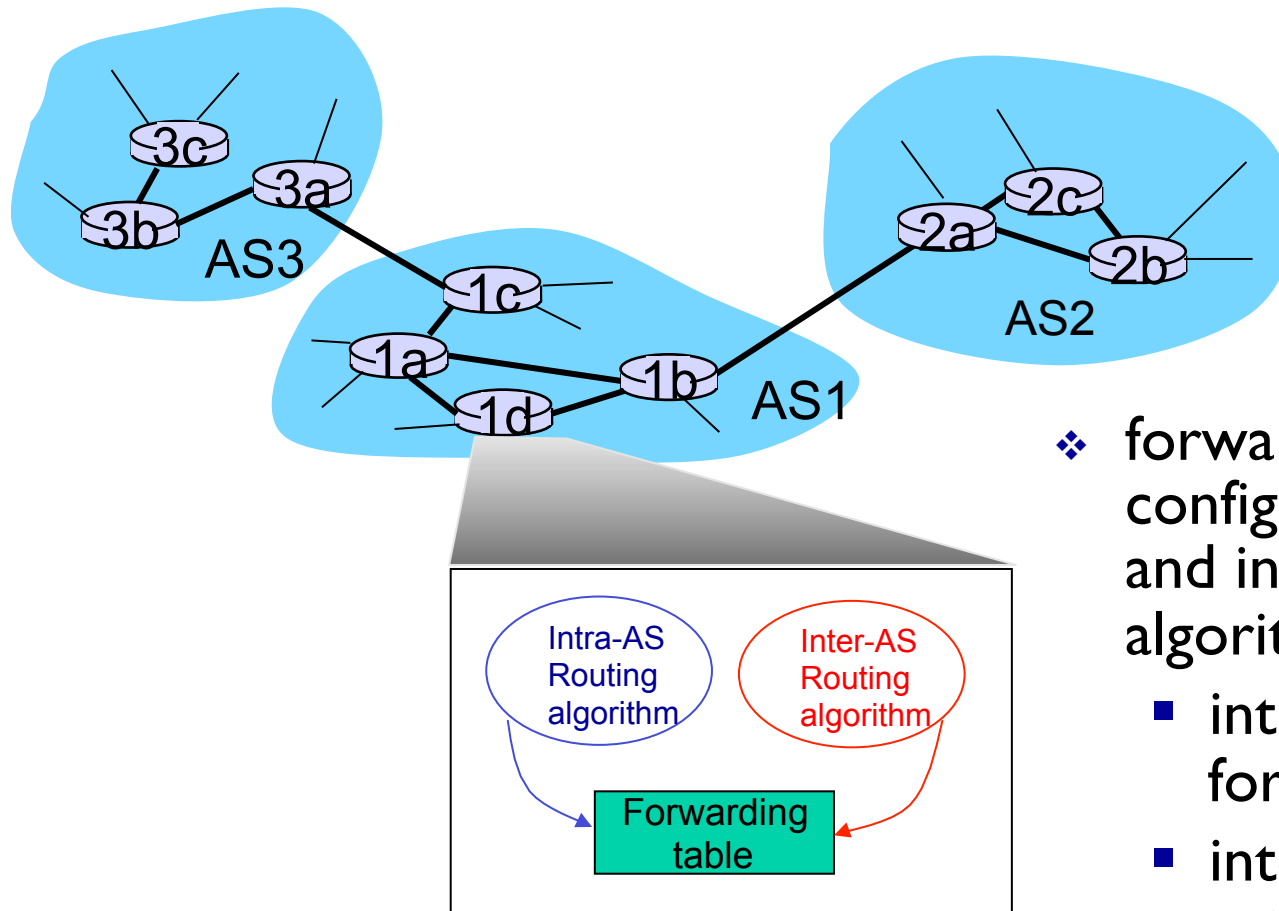
- ❖ forwarding table configured by both intra- and inter-AS routing algorithm
  - ▪ intra-AS sets entries for internal dests
  - ▪ inter-AS & intra-AS sets entries for external dests

Inter-AS tasks

❖ suppose router in AS1 receives datagram destined outside of AS1:

  ▪ router should forward packet to gateway router, but which one?

AS1 must:

1. learn which dests are reachable through AS2, which through AS3
2. propagate this reachability info to all routers in AS1

job of inter-AS routing!
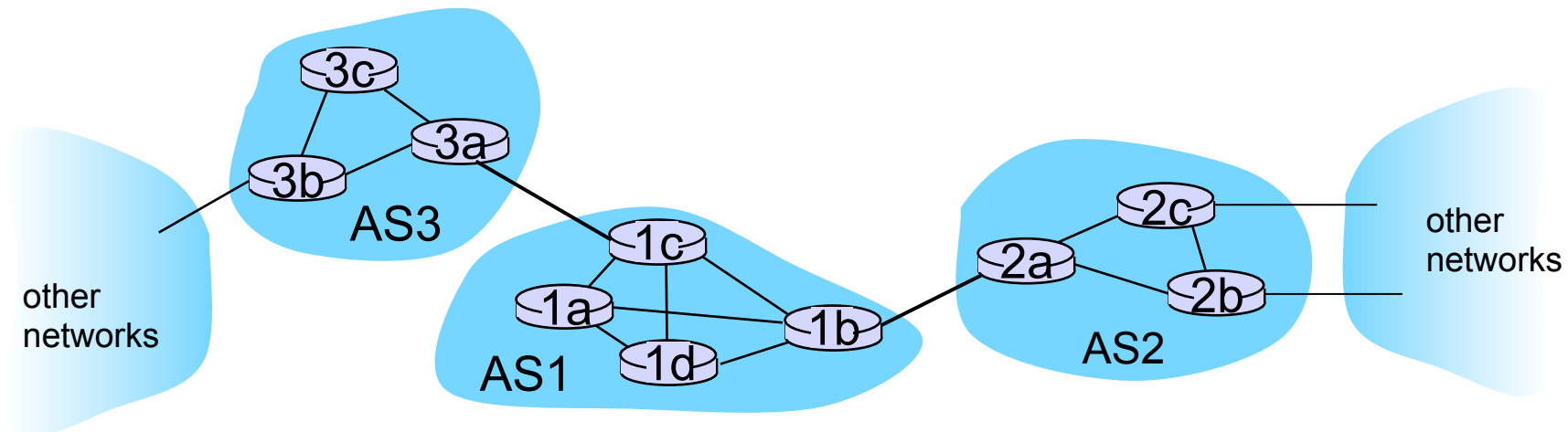
Example: setting forwarding table in router 1d

- ❖ suppose AS1 learns (via inter-AS protocol) that subnet *x* reachable via AS3 (gateway 1c), but not via AS2
    - ▪ inter-AS protocol propagates reachability info to all internal routers
- ❖ router 1d determines from intra-AS routing info that its interface *I* is on the least cost path to 1c
    - ▪ installs forwarding table entry *(x,I)*
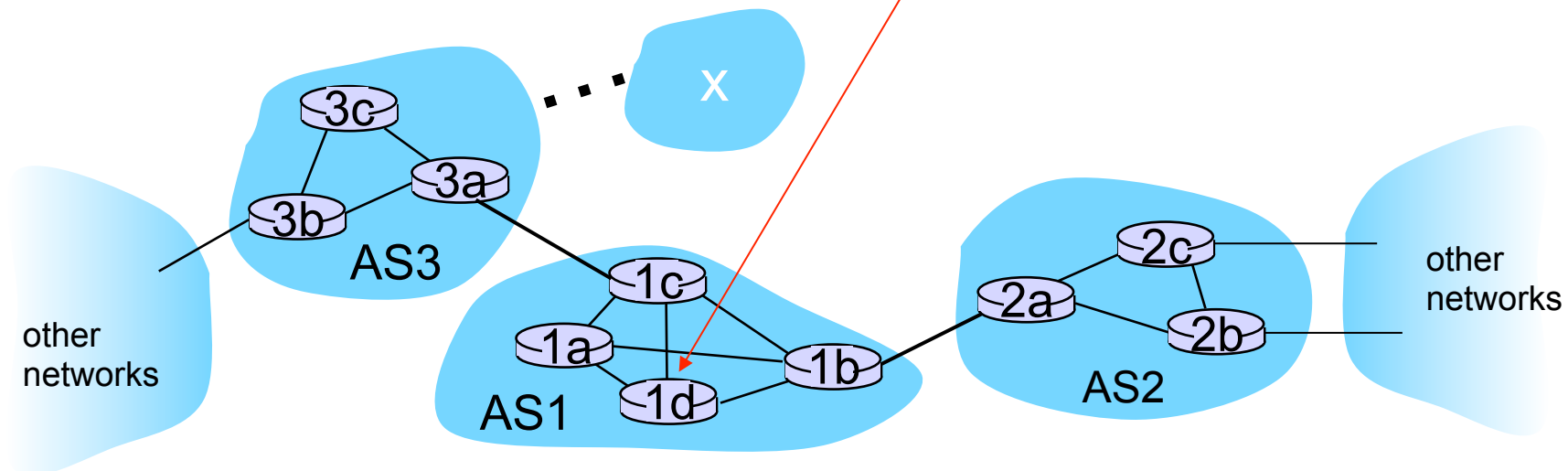
Example: choosing among multiple ASes

❖ now suppose AS1 learns from inter-AS protocol that subnet x is reachable from AS3 *and* from AS2.

❖ to configure forwarding table, router 1d must determine which gateway it should forward packets towards for dest x
  ▪ this is also job of inter-AS routing protocol!
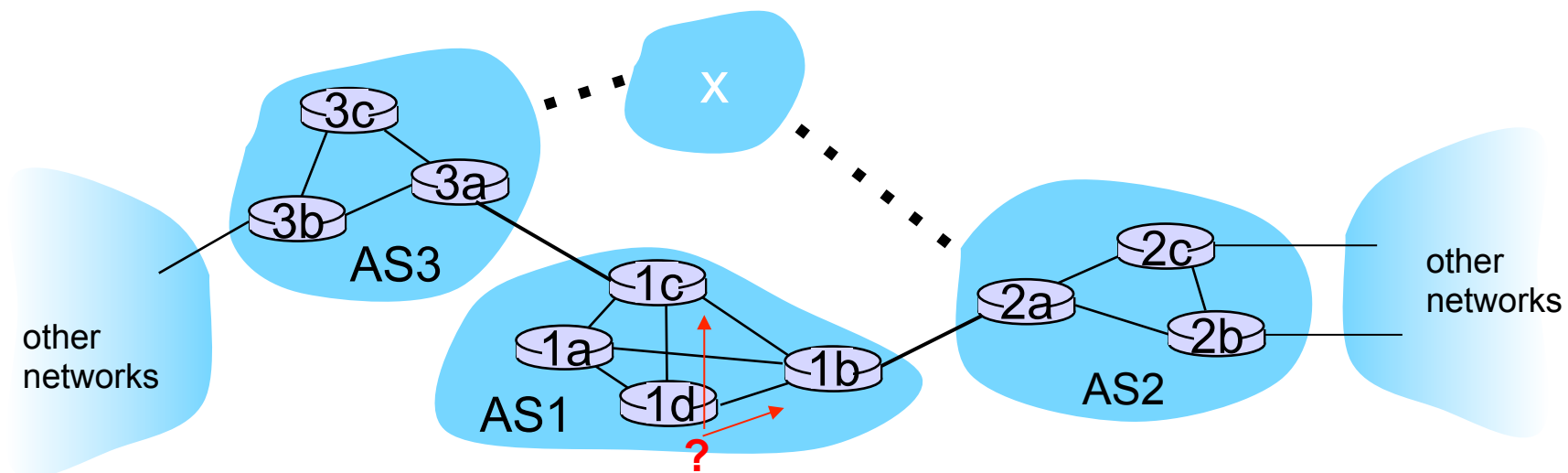
Example: choosing among multiple ASes

❖ now suppose AS1 learns from inter-AS protocol that subnet *x* is reachable from AS3 *and* from AS2.

❖ to configure forwarding table, router 1d must determine towards which gateway it should forward packets for dest *x*

  ▪ this is also job of inter-AS routing protocol!

❖ *hot potato routing: send* packet towards closest of two routers.
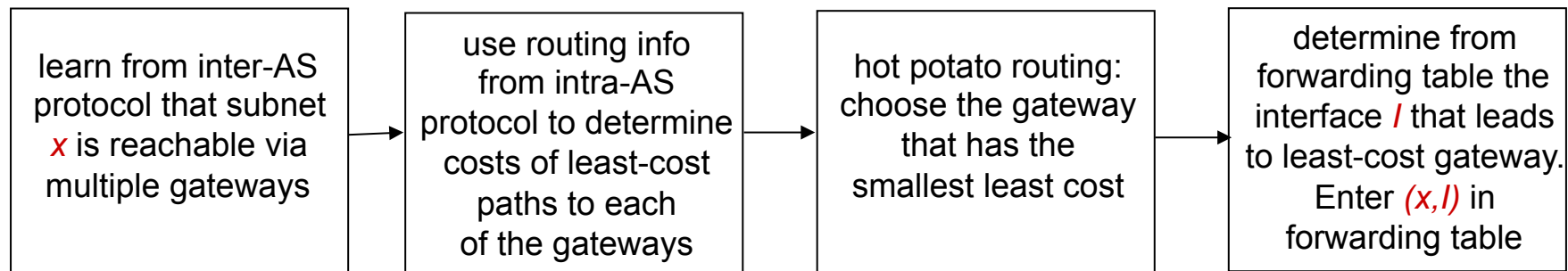
learn from inter-AS protocol that subnet *x* is reachable via multiple gateways	→	use routing info from intra-AS protocol to determine costs of least-cost paths to each of the gateways	→	hot potato routing: choose the gateway that has the smallest least cost	→	determine from forwarding table the interface *I* that leads to least-cost gateway. Enter *(x,I)* in forwarding table

Chapter 4: outline

4.1 introduction

4.2 virtual circuit and datagram networks

4.3 what's inside a router

4.4 IP: Internet Protocol
- datagram format
- IPv4 addressing
- ICMP
- IPv6

4.5 routing algorithms
- link state
- distance vector
- hierarchical routing

4.6 routing in the Internet
- RIP
- OSPF
- BGP

4.7 broadcast and multicast routing

Intra-AS Routing

❖ also known as *interior gateway protocols (IGP)*

❖ most common intra-AS routing protocols:

- RIP: Routing Information Protocol

- OSPF: Open Shortest Path First

- IGRP: Interior Gateway Routing Protocol (Cisco proprietary)

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
  ▪ DVs exchanged with neighbors every 30 sec in response message (aka advertisement)
  ▪ each advertisement: list of up to 25 destination *subnets (in IP addressing sense)*
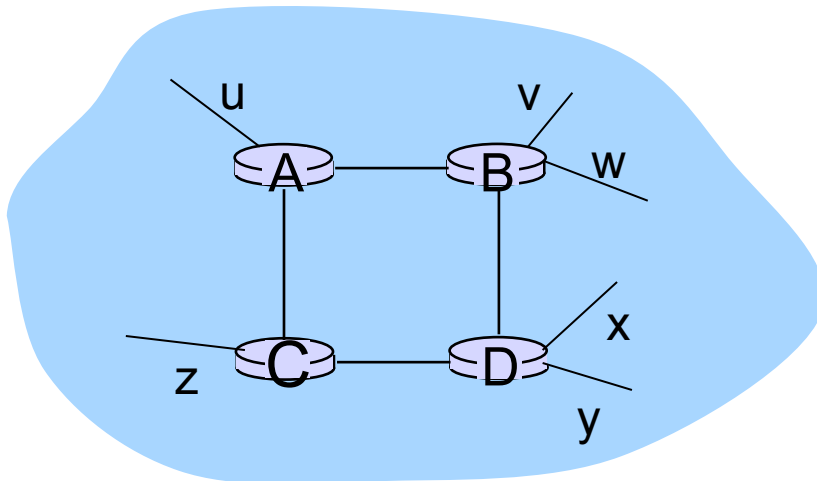
from router A to destination *subnets:*

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

RIP: example
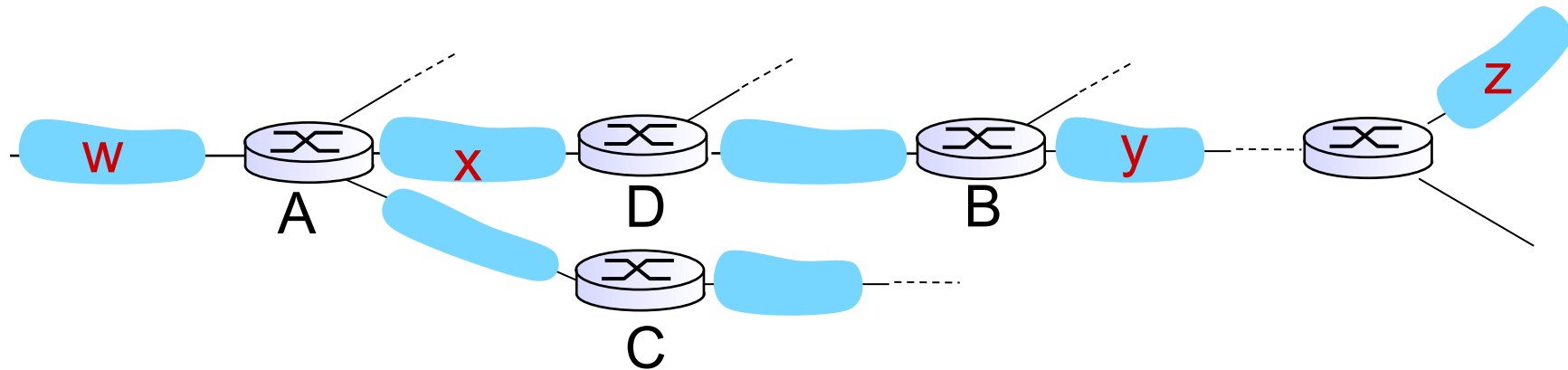
routing table in router D

destination subnet	next router	# hops to dest
w	A	2
y	B	2
z	B	7
x	--	1
....	....	....

RIP: example

A-to-D advertisement

dest	next	hops
w	-	1
x	-	1
z	C	4
....	....	....

w   A   x   D   B   y   z

C
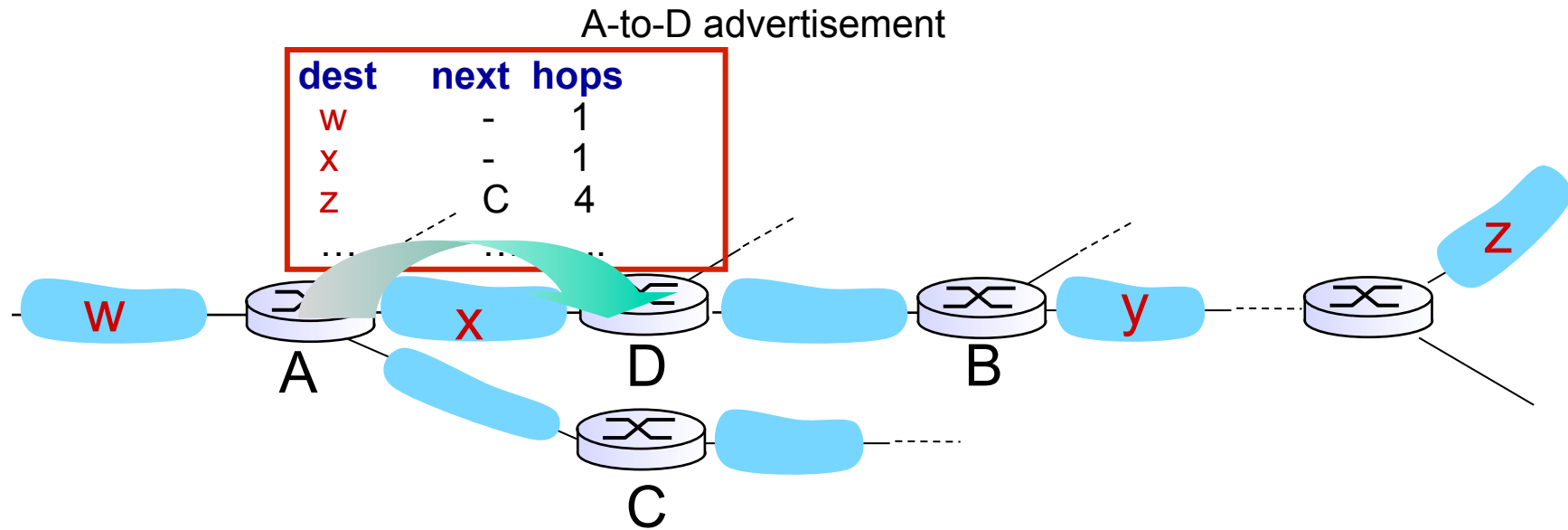
routing table in router D

destination subnet	next router	# hops to dest
w	A	2
y	B	2
z	B A	7 5
x	--	1
....	....	....

RIP: link failure, recovery

if no advertisement heard after 180 sec --> neighbor/
link declared dead

- routes via neighbor invalidated
- new advertisements sent to neighbors
- neighbors in turn send out new advertisements (if tables changed)
- link failure info quickly (?) propagates to entire net
- *poison reverse* used to prevent ping-pong loops (infinite distance = 16 hops)

RIP table processing

- ❖ RIP routing tables managed by *application-level* process called route-d (daemon)
- ❖ advertisements sent in UDP packets, periodically repeated

routed				routed
transport (UDP)				transprt (UDP)
network (IP)	*forwarding table*		*forwarding table*	network (IP)
link				link
physical				physical
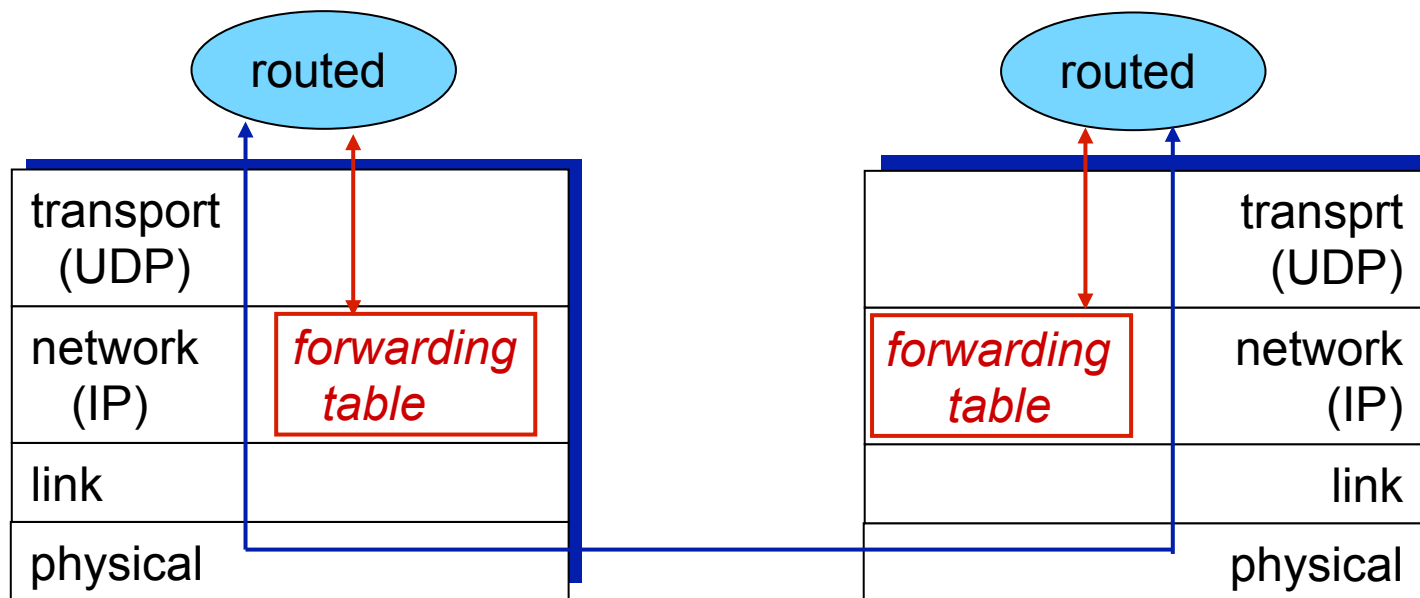
OSPF (Open Shortest Path First)

- ❖ "open": publicly available
- ❖ uses link state algorithm
  - ■ LS packet dissemination
  - ■ topology map at each node
  - ■ route computation using Dijkstra's algorithm
- ❖ OSPF advertisement carries one entry per neighbor
- ❖ advertisements flooded to *entire* AS
  - ■ carried in OSPF messages directly over IP (rather than TCP or UDP
- ❖ *IS-IS routing* protocol: nearly identical to OSPF

OSPF "advanced" features (not in RIP)

❖ *security:* all OSPF messages authenticated (to prevent malicious intrusion)

❖ multiple same-cost paths allowed (only one path in RIP)

❖ for each link, multiple cost metrics for different TOS (e.g., satellite link cost set "low" for best effort ToS; high for real time ToS)

❖ integrated uni- and multicast support:

  ▪ Multicast OSPF (MOSPF) uses same topology data base as OSPF

❖ hierarchical OSPF in large domains.

Hierarchical OSPF

boundary router

backbone router

backbone

area border routers

area 1

area 2

area 3

internal routers
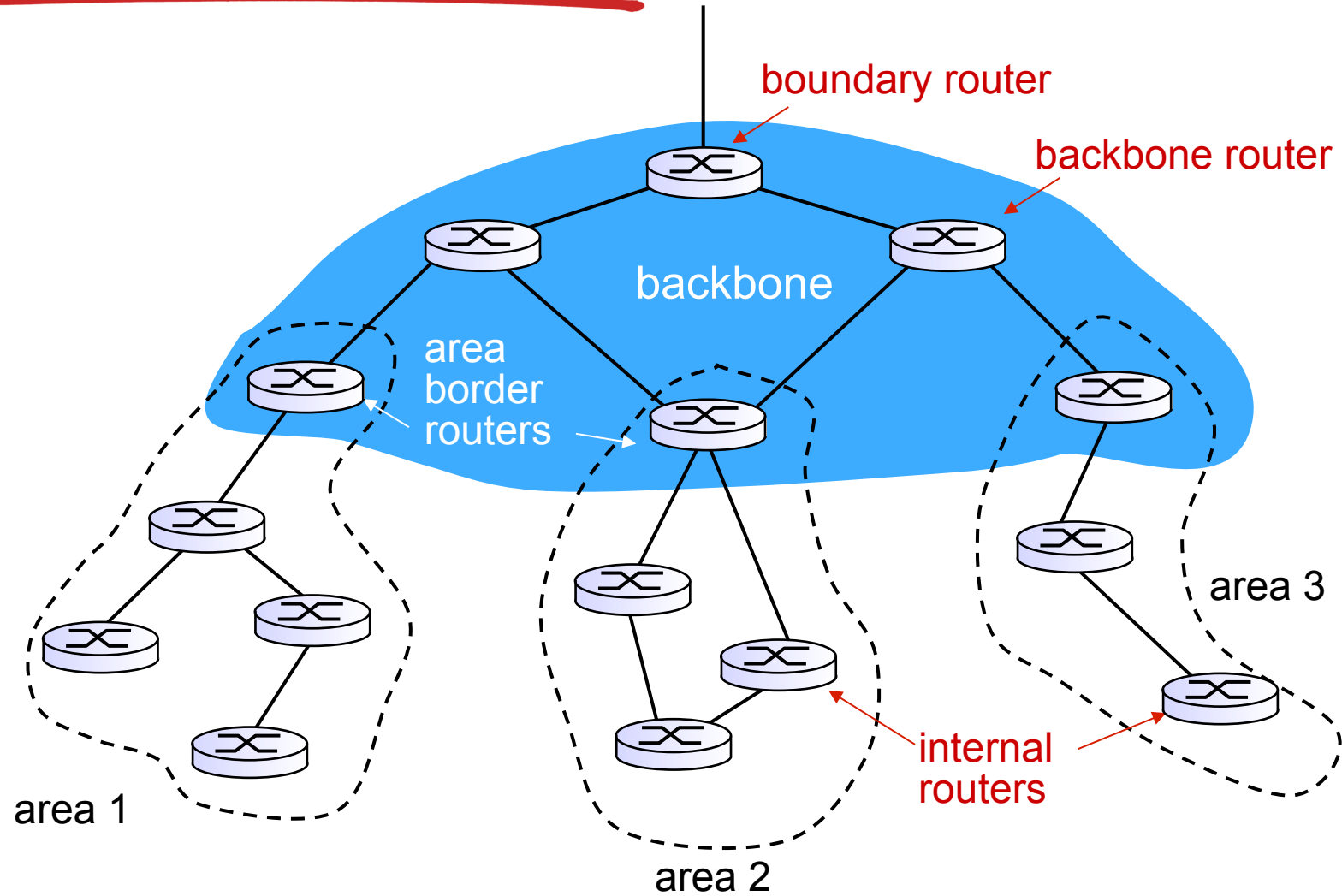
Hierarchical OSPF

❖ *two-level hierarchy:* local area, backbone.
  ▪ link-state advertisements only in area
  ▪ each nodes has detailed area topology; only know direction (shortest path) to nets in other areas.

❖ *area border routers:* "summarize" distances to nets in own area, advertise to other Area Border routers.

❖ *backbone routers:* run OSPF routing limited to backbone.

❖ *boundary routers:* connect to other AS's.

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:
    - ▪ **eBGP:** obtain subnet reachability information from neighboring ASs.
    - ▪ **iBGP:** propagate reachability information to all AS-internal routers.
    - ▪ determine "good" routes to other networks based on reachability information and policy.
- ❖ 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

- ❖ when AS3 advertises a prefix to AS1:
  - AS3 *promises* it will forward datagrams towards that prefix
  - AS3 can aggregate prefixes in its advertisement
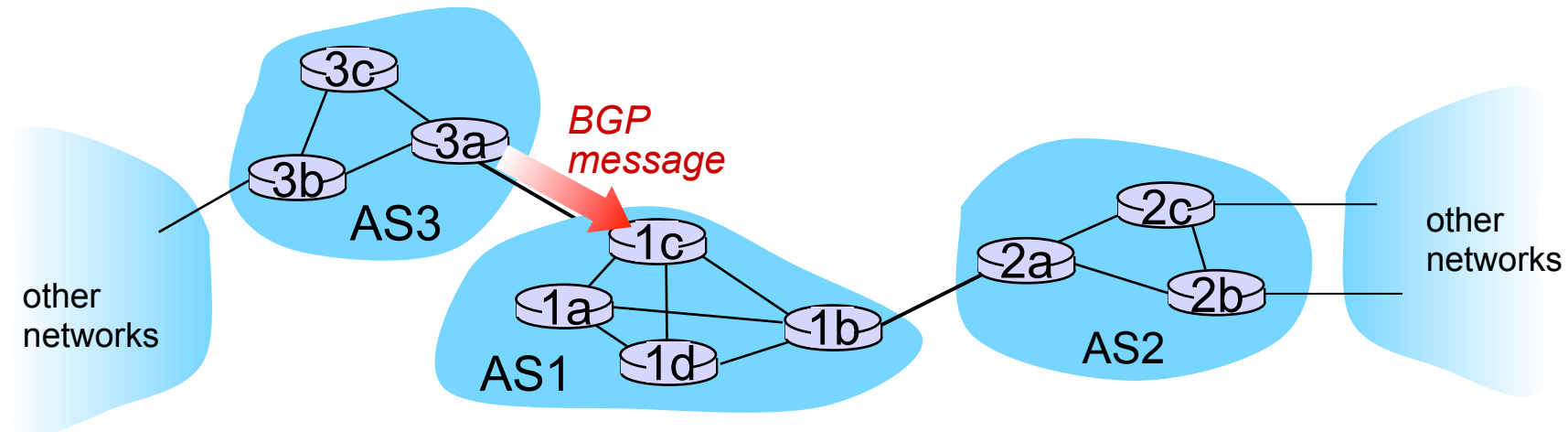
BGP basics: distributing path information

❖ using eBGP session between 3a and 1c, AS3 sends prefix reachability info to AS1.

  ▪ 1c can then use iBGP do 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

❖ when router learns of new prefix, it creates entry for prefix in its forwarding table.

eBGP session

iBGP session

3a

3b

AS3

1c

1a

1d

1b

2a

2c

2b

AS1

AS2

other networks

other networks
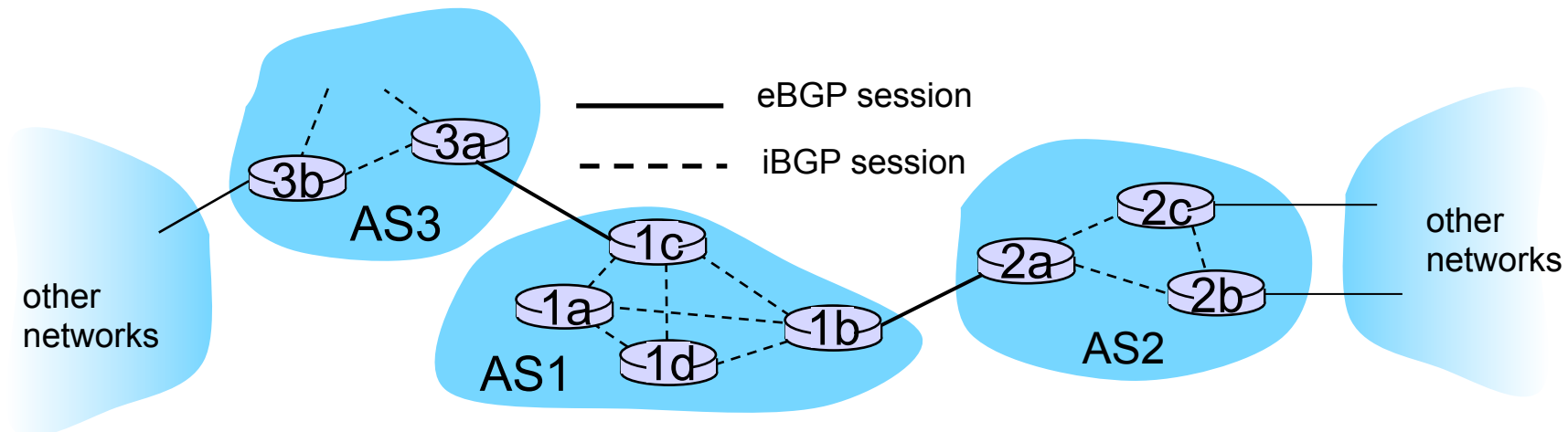
Path attributes and BGP routes

- ❖ **advertised prefix includes BGP attributes**
    - ▪ prefix + attributes = "route"
- ❖ **two important attributes:**
    - ▪ AS-PATH: contains ASs through which prefix advertisement has passed: e.g., AS 67, AS 17
    - ▪ NEXT-HOP: indicates specific internal-AS router to next-hop AS. (may be multiple links from current AS to next-hop-AS)
- ❖ **gateway router receiving route advertisement uses import policy to accept/decline**
    - ▪ e.g., never route through AS x
    - ▪ *policy-based* routing

BGP route selection

❖ router may learn about more than 1 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

BGP messages

❖ BGP messages exchanged between peers over TCP connection

❖ BGP messages:

- **OPEN:** opens TCP connection to peer and authenticates sender
- **UPDATE:** advertises new path (or withdraws old)
- **KEEPALIVE:** keeps connection alive in absence of UPDATES; also ACKs OPEN request
- **NOTIFICATION:** reports errors in previous msg; also used to close connection

BGP routing policy
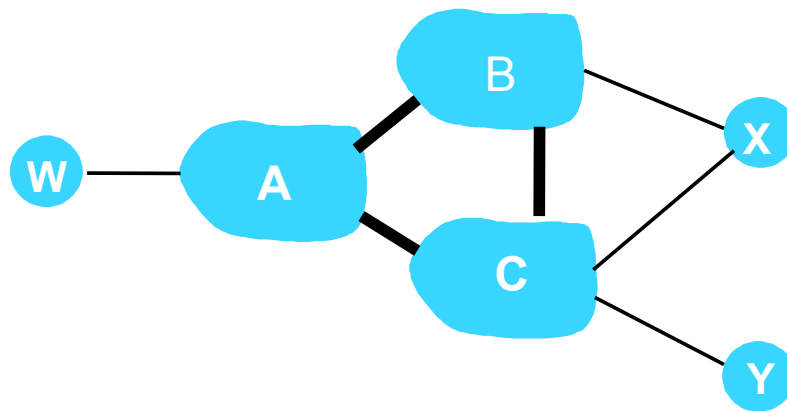
legend:

provider network

customer network:

- A,B,C are *provider networks*
- X,W,Y are customer (of provider networks)
- X is *dual-homed:* attached to two networks
  - X does not want to route from B via X to C
  - .. so X will not advertise to B a route to C

BGP routing policy (2)

legend:

provider
network
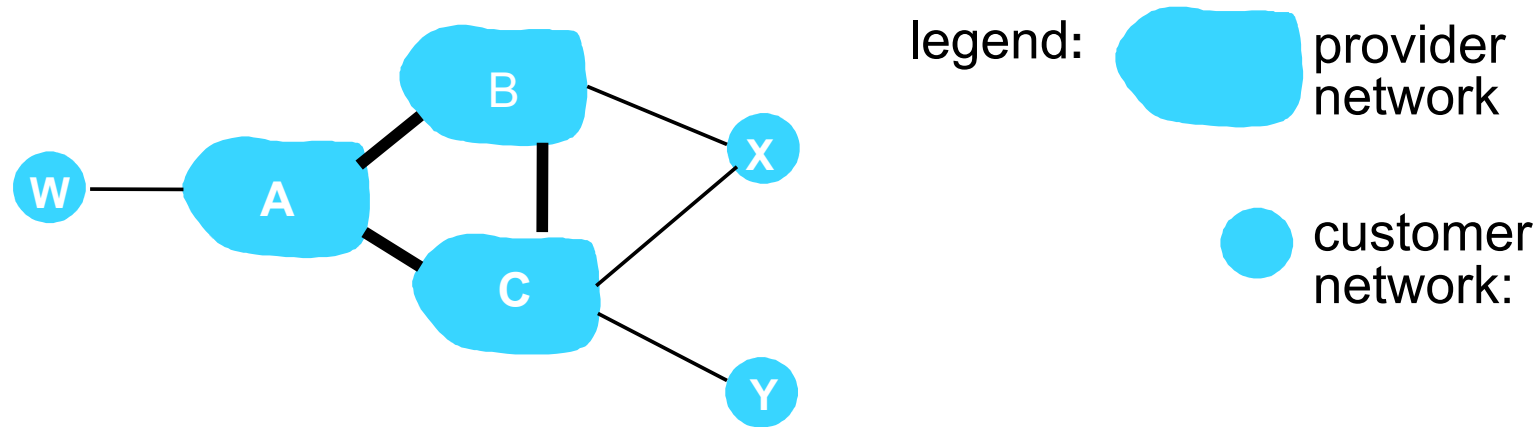
customer
network:

- ❖ A advertises path AW to B
- ❖ B advertises path BAW to X
- ❖ Should B advertise path BAW to C?
  - No way! B gets no "revenue" for routing CBAW since neither W nor C are B's customers
  - B wants to force C to route to w via A
  - B wants to route *only* to/from its customers!

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

Chapter 4: outline

Broadcast routing

❖ deliver packets from source to all other nodes

❖ source duplication is inefficient:

duplicate

R1

duplicate
creation/transmission

R1

duplicate

R2

R2

duplicate

R3   R4

R3   R4

source
duplication

in-network
duplication
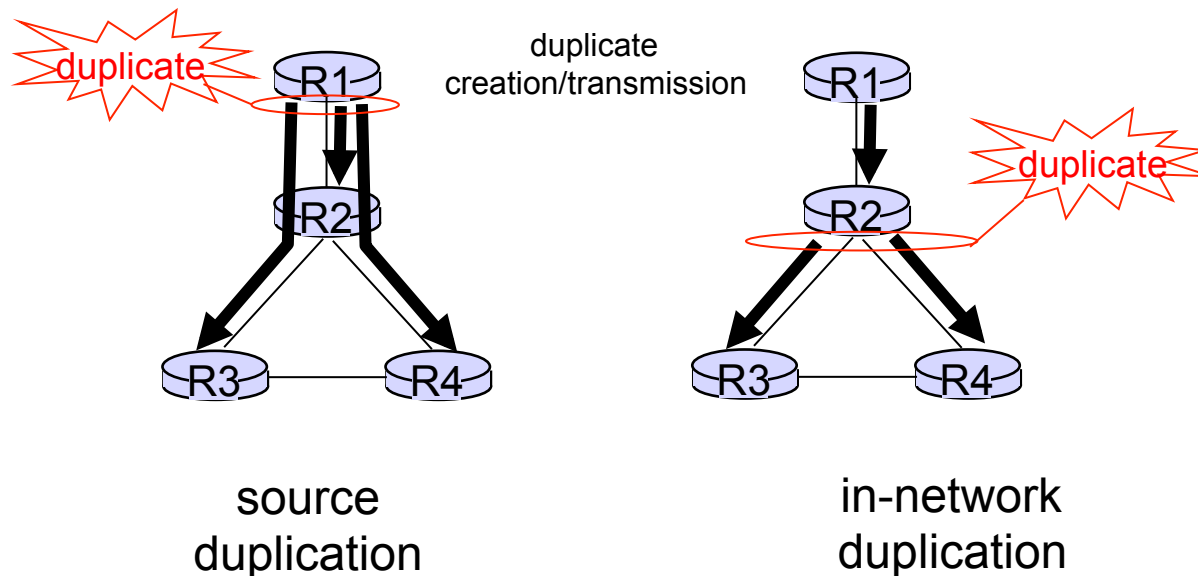
❖ source duplication: how does source determine
recipient addresses?

In-network duplication

❖ *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
  - 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

❖ first construct a spanning tree
❖ nodes then forward/make copies only along spanning tree
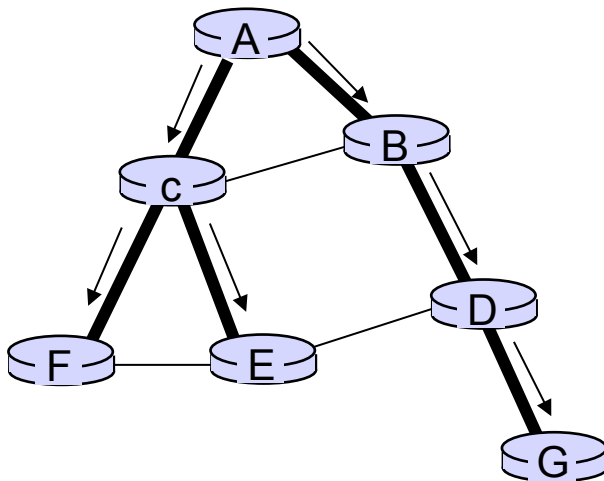
(a) broadcast initiated at A   (b) broadcast initiated at D
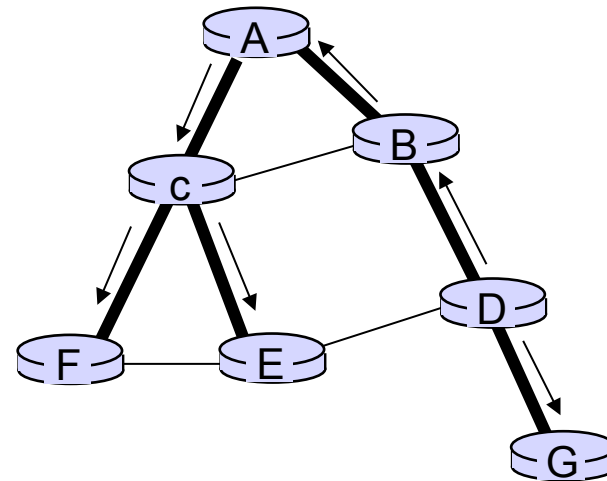
Spanning tree: creation

❖ center node

❖ each node sends unicast join message to center node

  ▪ message forwarded until it arrives at a node already belonging to spanning tree
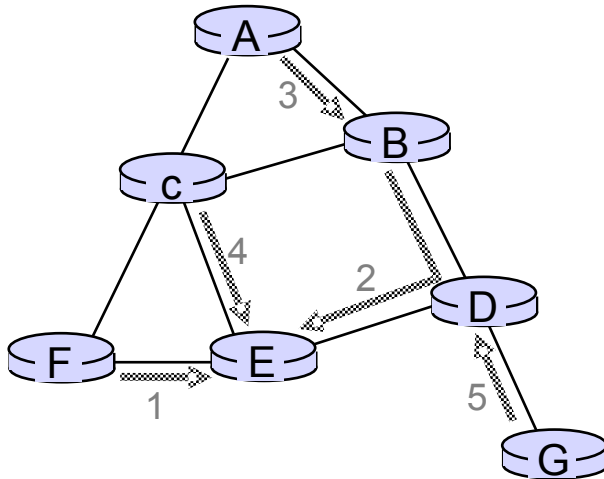
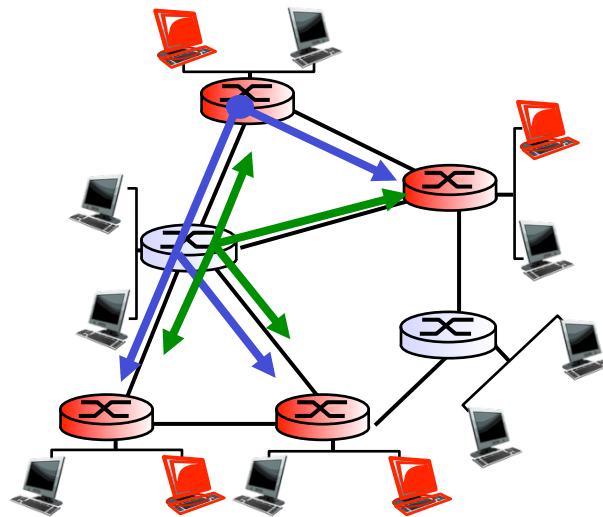(a) stepwise construction of spanning tree (center: E)
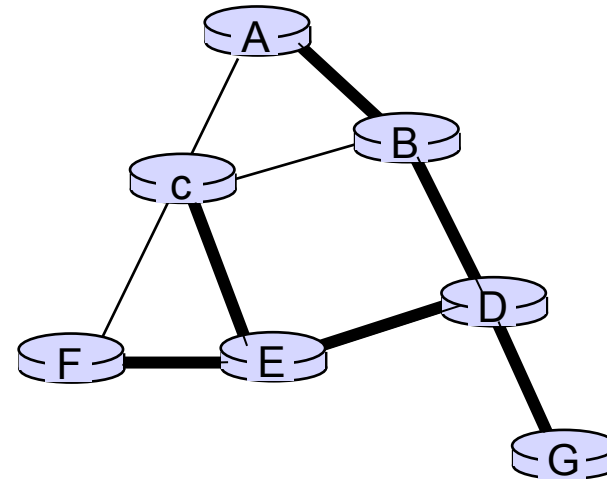
(b) constructed spanning tree

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

legend

group member

not group member

router with a group member

router without group member

shared tree

source-based trees

Approaches for building mcast trees

approaches:

❖ *source-based tree:* one tree per source
  - shortest path trees
  - reverse path forwarding

❖ *group-shared tree:* group uses one tree
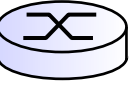 - minimal spanning (Steiner)
  - center-based trees

…we first look at basic approaches, then specific protocols adopting these approaches

Shortest path tree

- ❖ mcast forwarding tree: tree of shortest path routes from source to all receivers
 - ▪ Dijkstra's algorithm
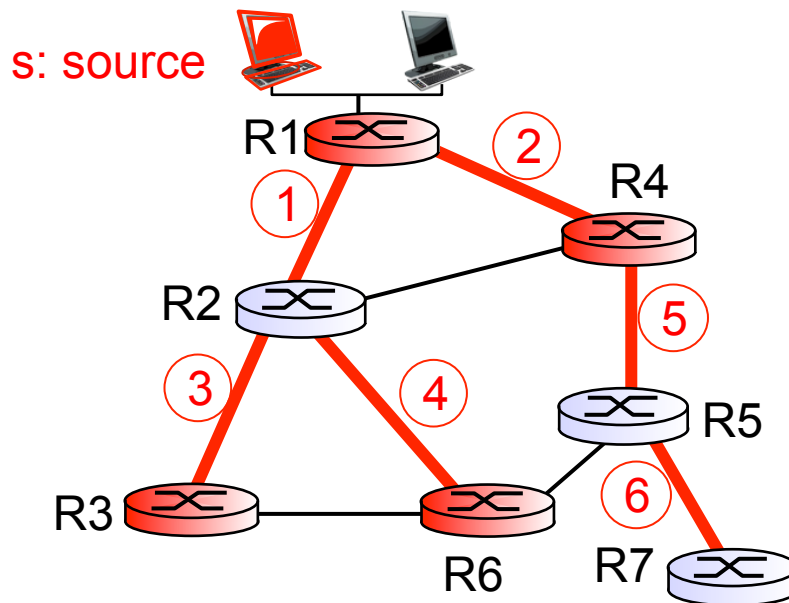
s: source

LEGEND

router with attached group member

router with no attached group member

(i) link used for forwarding, i indicates order link added by algorithm

Reverse path forwarding

❖ 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 center)
  then flood datagram onto all outgoing links
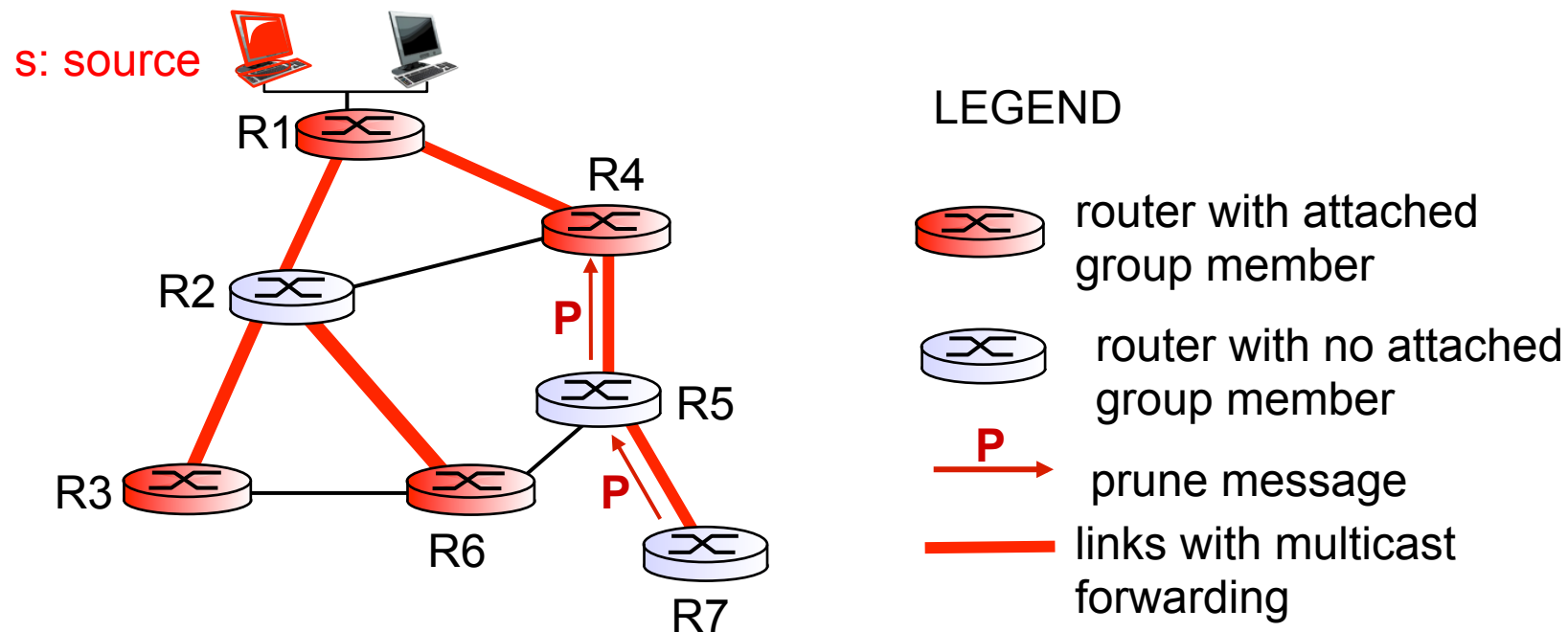 **else** ignore datagram

Reverse path forwarding: example

s: source

LEGEND

router with attached
group member

router with no attached
group member

→ datagram will be forwarded

⊣ datagram will not be
forwarded
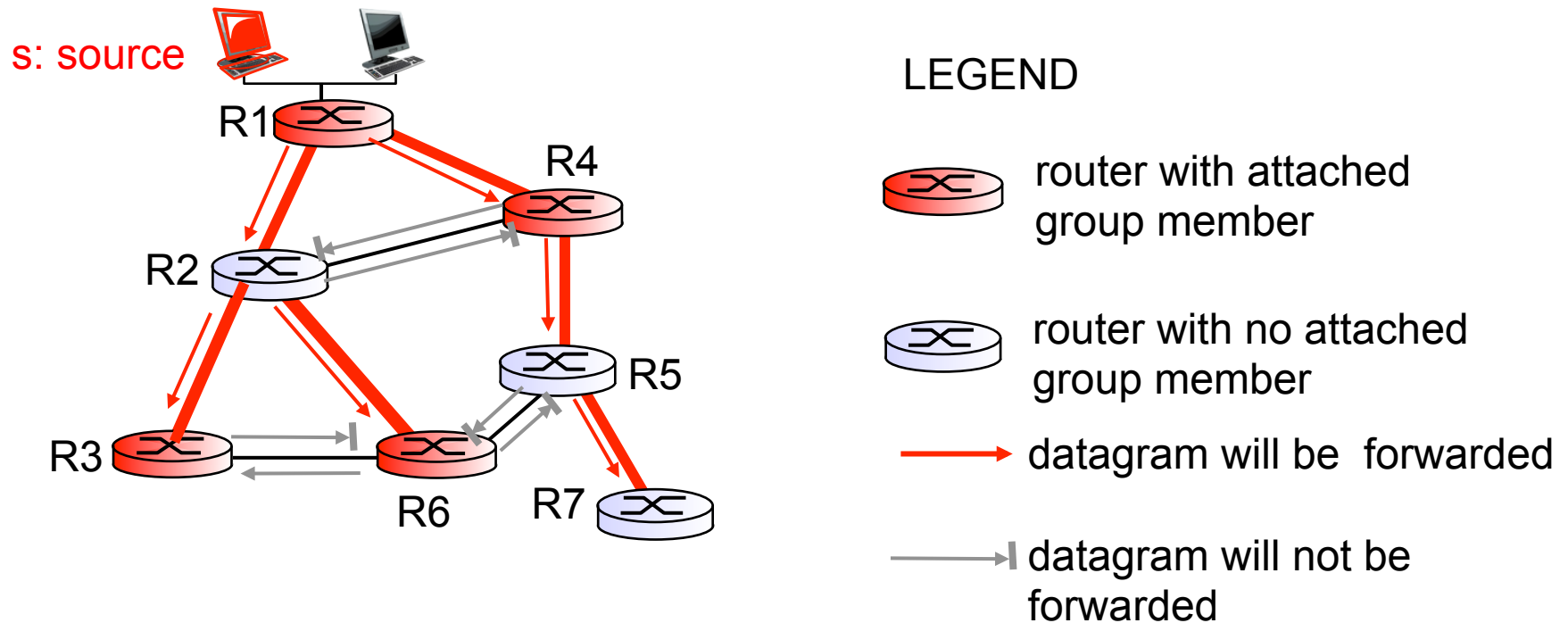
❖ result is a source-specific *reverse* SPT

  ▪ may be a bad choice with asymmetric links

Reverse path forwarding: pruning

❖ forwarding tree contains subtrees with no mcast group members
  ▪ no need to forward datagrams down subtree
  ▪ "prune" msgs sent upstream by router with no downstream group members

s: source

LEGEND

R1
R2
R3
R4
R5
R6
R7

P
P

router with attached group member

router with no attached group member

P → prune message

links with multicast forwarding

Shared-tree: steiner tree

❖ *steiner tree:* minimum cost tree connecting all routers with attached group members

❖ problem is NP-complete

❖ excellent heuristics exists

❖ not used in practice:
  ▪ computational complexity
  ▪ information about entire network needed
  ▪ monolithic: rerun whenever a router needs to join/ leave

Center-based trees

❖ single delivery tree shared by all

❖ 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

Center-based trees: example

suppose R6 chosen as center:
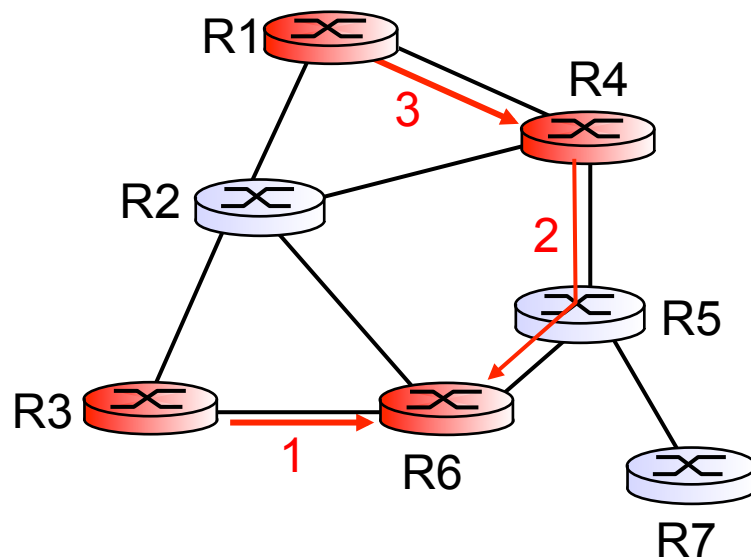
LEGEND

router with attached group member

router with no attached group member

1 → path order in which join messages generated

Internet Multicasting Routing: DVMRP

❖ DVMRP: distance vector multicast routing protocol, RFC1075

❖ *flood and prune:* reverse path forwarding, source-based tree

- RPF tree based on DVMRP's own routing tables constructed by communicating DVMRP routers
- no assumptions about underlying unicast
- initial datagram to mcast group flooded everywhere via RPF
- routers not wanting group: send upstream prune msgs

DVMRP: continued…

- *soft state:* DVMRP router periodically (1 min.) "forgets" branches are pruned:
  - mcast data again flows down unpruned branch
  - downstream router: reprune or else continue to receive data
- routers can quickly regraft to tree
  - following IGMP join at leaf
- odds and ends
  - commonly implemented in commercial router

Tunneling

Q: how to connect "islands" of multicast routers in a "sea" of unicast routers?
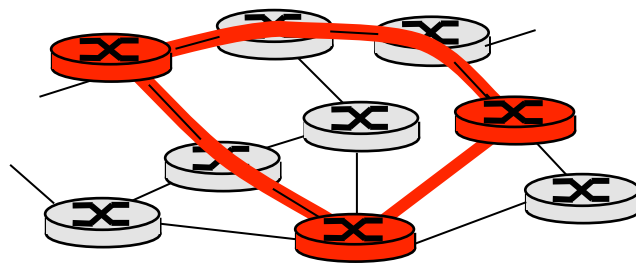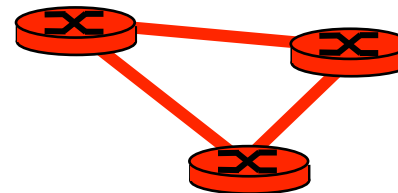
physical topology   logical topology

❖ mcast datagram encapsulated inside "normal" (non-multicast-addressed) datagram

❖ normal IP datagram sent thru "tunnel" via regular IP unicast to receiving mcast router (recall IPv6 inside IPv4 tunneling)

❖ receiving mcast router unencapsulates to get mcast datagram

PIM: Protocol Independent Multicast

❖ not dependent on any specific underlying unicast routing algorithm (works with all)

❖ two different multicast distribution scenarios :

dense:

❖ group members densely packed, in "close" proximity.

❖ bandwidth more plentiful

sparse:

❖ # networks with group members small wrt # interconnected networks

❖ group members "widely dispersed"

❖ bandwidth not plentiful

Consequences of sparse-dense dichotomy:

dense

- ❖ group membership by routers *assumed* until routers explicitly prune
- ❖ *data-driven* construction on mcast tree (e.g., RPF)
- ❖ bandwidth and non-group-router processing *profligate*

sparse:

- ❖ no membership until routers explicitly join
- ❖ *receiver- driven* construction of mcast tree (e.g., center-based)
- ❖ bandwidth and non-group-router processing *conservative*

PIM- dense mode

flood-and-prune RPF: similar to DVMRP but…

- ❖ underlying unicast protocol provides RPF info for incoming datagram

- ❖ less complicated (less efficient) downstream flood than DVMRP reduces reliance on underlying routing algorithm

- ❖ has protocol mechanism for router to detect it is a leaf-node router

PIM - sparse mode

- ❖ center-based approach
- ❖ router sends *join* msg to rendezvous point (RP)
  - ▪ intermediate routers update state and forward *join*
- ❖ after joining via RP, router can switch to source-specific tree
  - ▪ increased performance: less concentration, shorter paths

R1

R4

join

R2

join

R5

join

R3

join

R6

all data multicast from rendezvous point

R7

rendezvous point
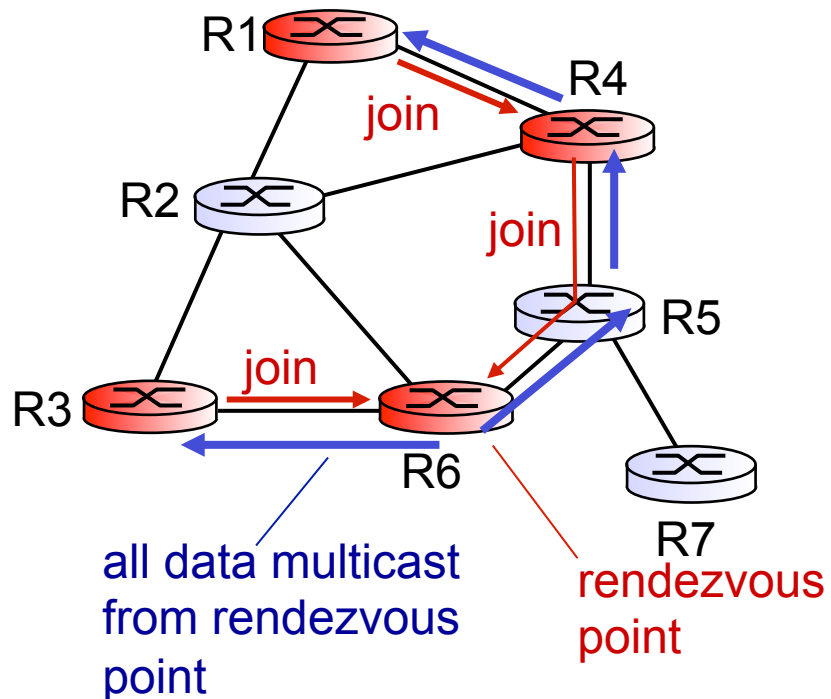
PIM - sparse mode

sender(s):

❖ unicast data to RP, which distributes down RP-rooted tree

❖ RP can extend mcast tree upstream to source

❖ RP can send *stop* msg if no attached receivers

▪ "no one is listening!"

R1

R4

join

R2

join

R5

R3

join

R6

R7

all data multicast from rendezvous point

rendezvous point
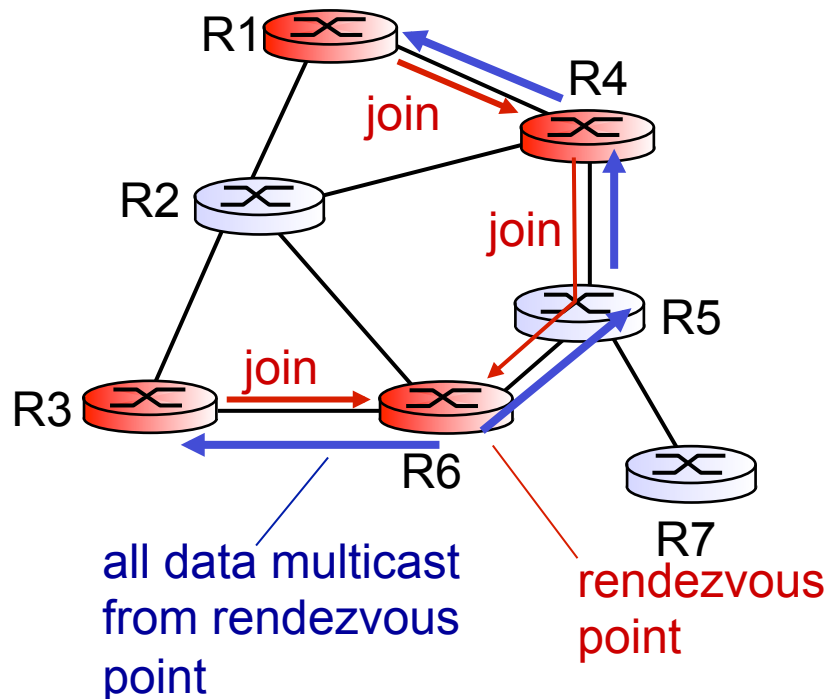
Chapter 4: *done!*

4.1 introduction

4.2 virtual circuit and datagram networks

4.3 what's inside a router

4.4 IP: Internet Protocol
- datagram format, IPv4 addressing, ICMP, IPv6

4.5 routing algorithms
- link state, distance vector, hierarchical routing

4.6 routing in the Internet
- RIP, OSPF, BGP

4.7 broadcast and multicast routing

❖ understand principles behind network layer services:
- network layer service models, forwarding versus routing how a router works, routing (path selection), broadcast, multicast

❖ instantiation, implementation in the Internet