The Network Layer: Part II

These slides are adapted from those provided by Jim Kurose and Keith Ross with their book "Computer Networking: A Top-Down Approach (6th edition)."

Outline

- ✓ Network layer functions, mainly forwarding and routing
- ✓ Network layer services
- ✓ Datagram vs. Virtual circuit networks
- ✓ Router architectures and design issues
- ✓ IPv4 (incl. fragmentation)
- ✓ Internet addressing, DHCP and NAT
- ❖ IPv6
- ❖ ICMP
- ❖ Routing algorithms (link state, distance vector, hierarchical)
- ❖ Routing in the Internet (OSPF, BGP)

IPv6 Motivations

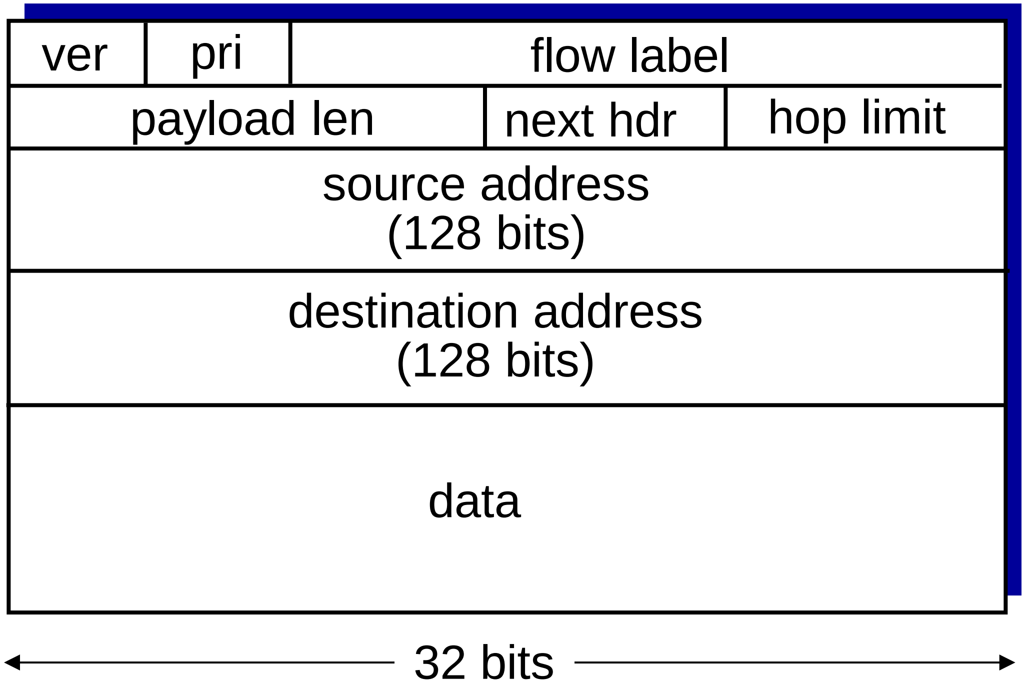
- *initial motivation:* 32-bit IPv4 address space was getting used up quickly.

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

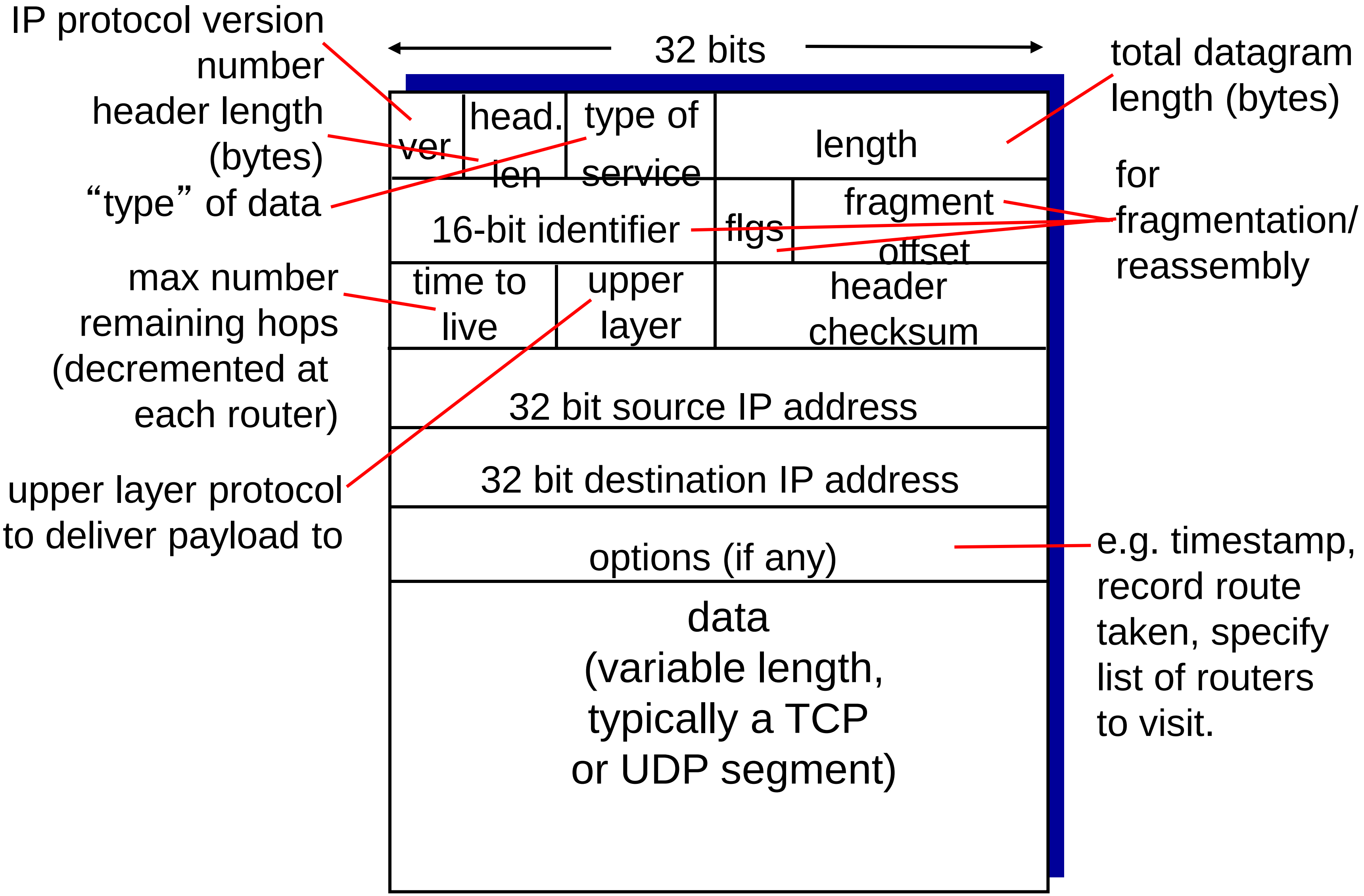
➔ *IPv6:*
  - IP address size increased from 32 bits to 128 bits
  - fixed-length 40 byte header
  - fragmentation not allowed, no header checksum, options left out of the standard header
  - flow labels and priorities

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



Cf. IPv4 Datagram Format



IP protocol version number

header length (bytes)

“type” of data

max number remaining hops (decremented at each router)

upper layer protocol to deliver payload to

32 bits

ver | head. len | type of service | length

16-bit identifier | flgs | fragment offset

time to live | upper layer | header checksum

32 bit source IP address

32 bit destination IP address

options (if any)

data (variable length, typically a TCP or UDP segment)

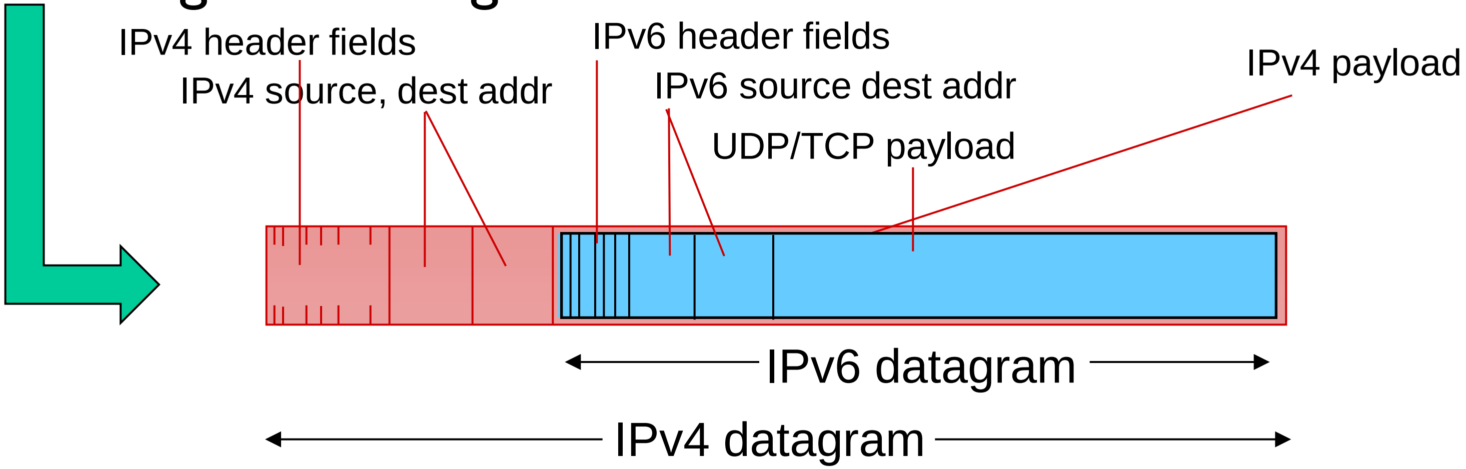
total datagram length (bytes)

for fragmentation/ reassembly

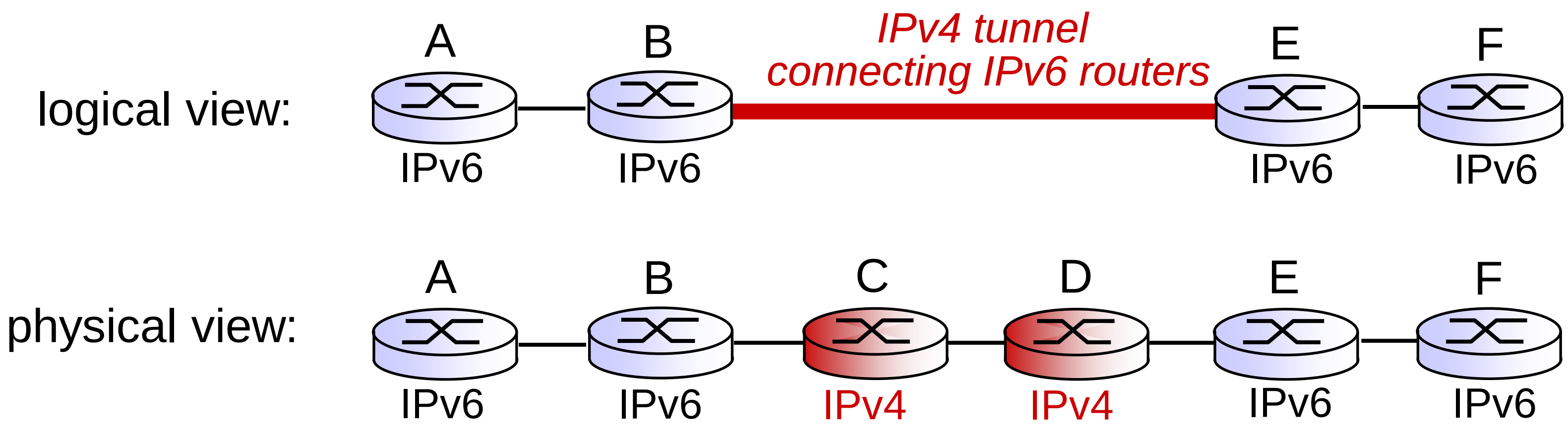
e.g. timestamp, record route taken, specify list of routers to visit.

Transition from IPv4 to IPv6

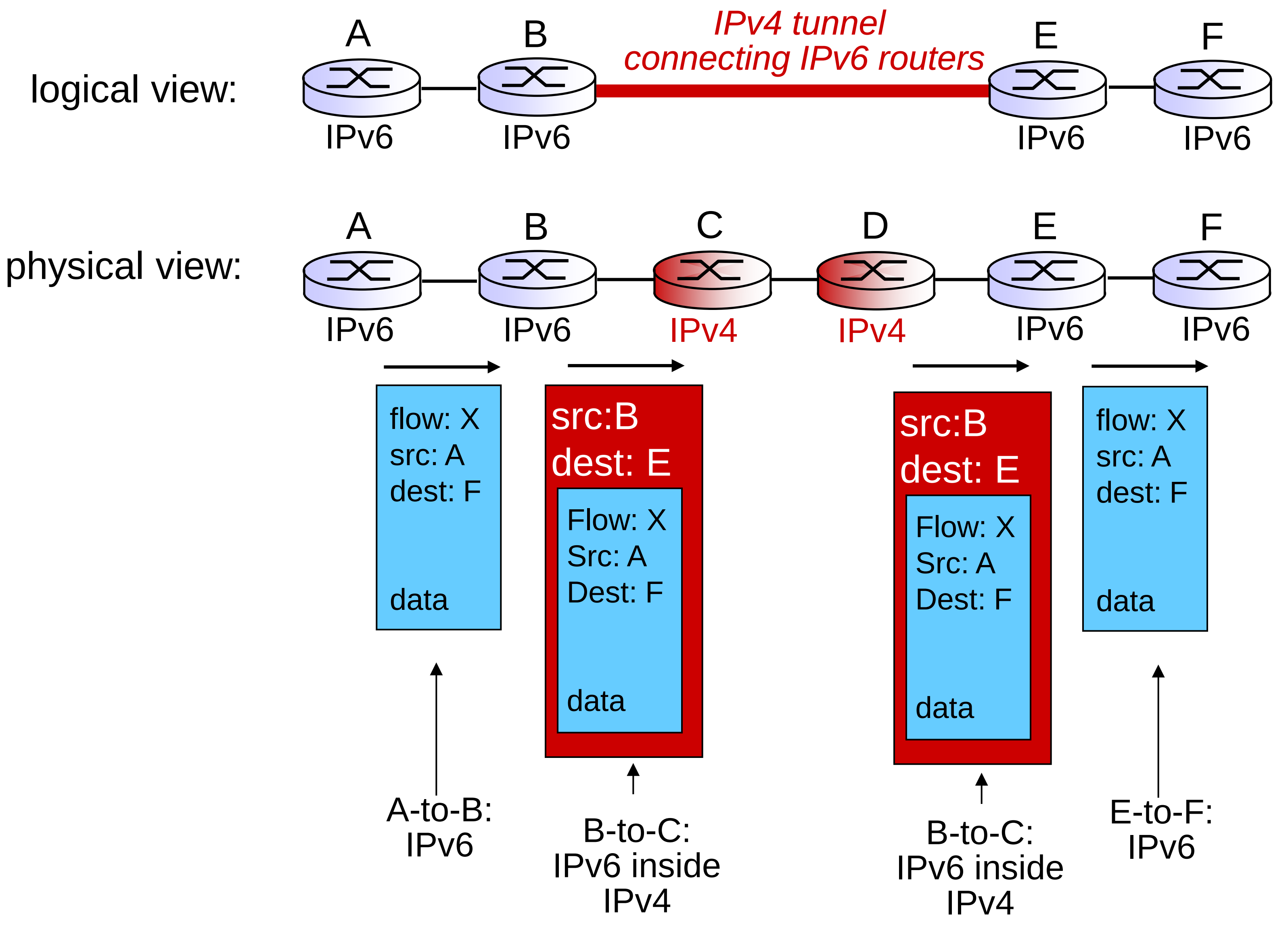
- declaring a flag day to switch all routers from IPv4 to IPv6 impractical
- how will network operate with mixed IPv4 and IPv6 routers?
  1. *dual-stack approach:* IPv6 capable routers also support IPv4
     - Shortcoming: protocol conversion between IPv6 and IPv4 packets causes loss of header fields
  2. *tunneling approach:* IPv6 datagram carried as *payload* in IPv4 datagram among IPv4 routers



Tunneling Illustrated



Tunneling Illustrated



Outline

- ✓ Network layer functions, mainly forwarding and routing
- ✓ Network layer services
- ✓ Datagram vs. Virtual circuit networks
- ✓ Router architectures and design issues
- ✓ IPv4 (incl. fragmentation)
- ✓ Internet addressing, DHCP and NAT
- ✓ IPv6
- ❖ ICMP
- ❖ Routing algorithms (link state, distance vector, hierarchical)
- ❖ Routing in the Internet (OSPF, BGP)

Internet Control Message Protocol (ICMP)

- used by hosts & routers to communicate network-layer information
  - Typically for error reporting (e.g., unreachable host/network/port/protocol)
  - But other uses too (e.g., ping via echo request/reply)
- network-layer "above" IP:
  - ICMP messages carried in IP datagrams
- ICMP message: type, code plus first 8 bytes of IP datagram causing error

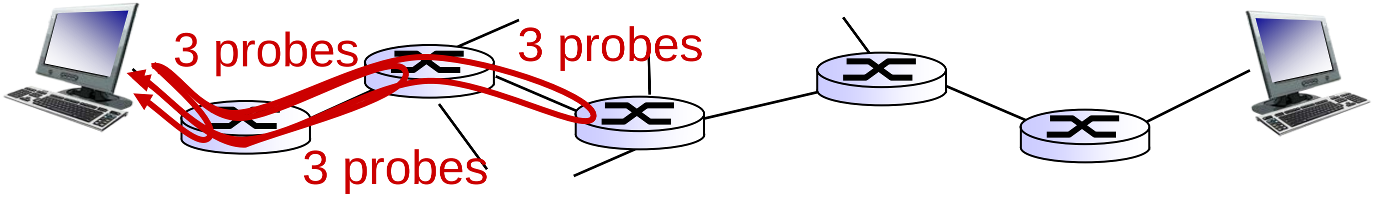
Type	Code	description
0	0	echo reply (ping)
3	0	dest. network unreachable
3	1	dest host unreachable
3	2	dest protocol unreachable
3	3	dest port unreachable
3	6	dest network unknown
3	7	dest host unknown
4	0	source quench (congestion control - not used)
8	0	echo request (ping)
9	0	route advertisement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

ICMP Example: Traceroute

- source sends series of UDP segments to dest using an unlikely dest port number
  - first set has TTL=1
  - second set has TTL=2, and so on.
- when *n*th set of datagrams arrives at *n*th router:
  - router discards datagrams
  - sends back ICMP "TTL expired" messages (type 11, code 0) to source
  - ICMP messages include 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



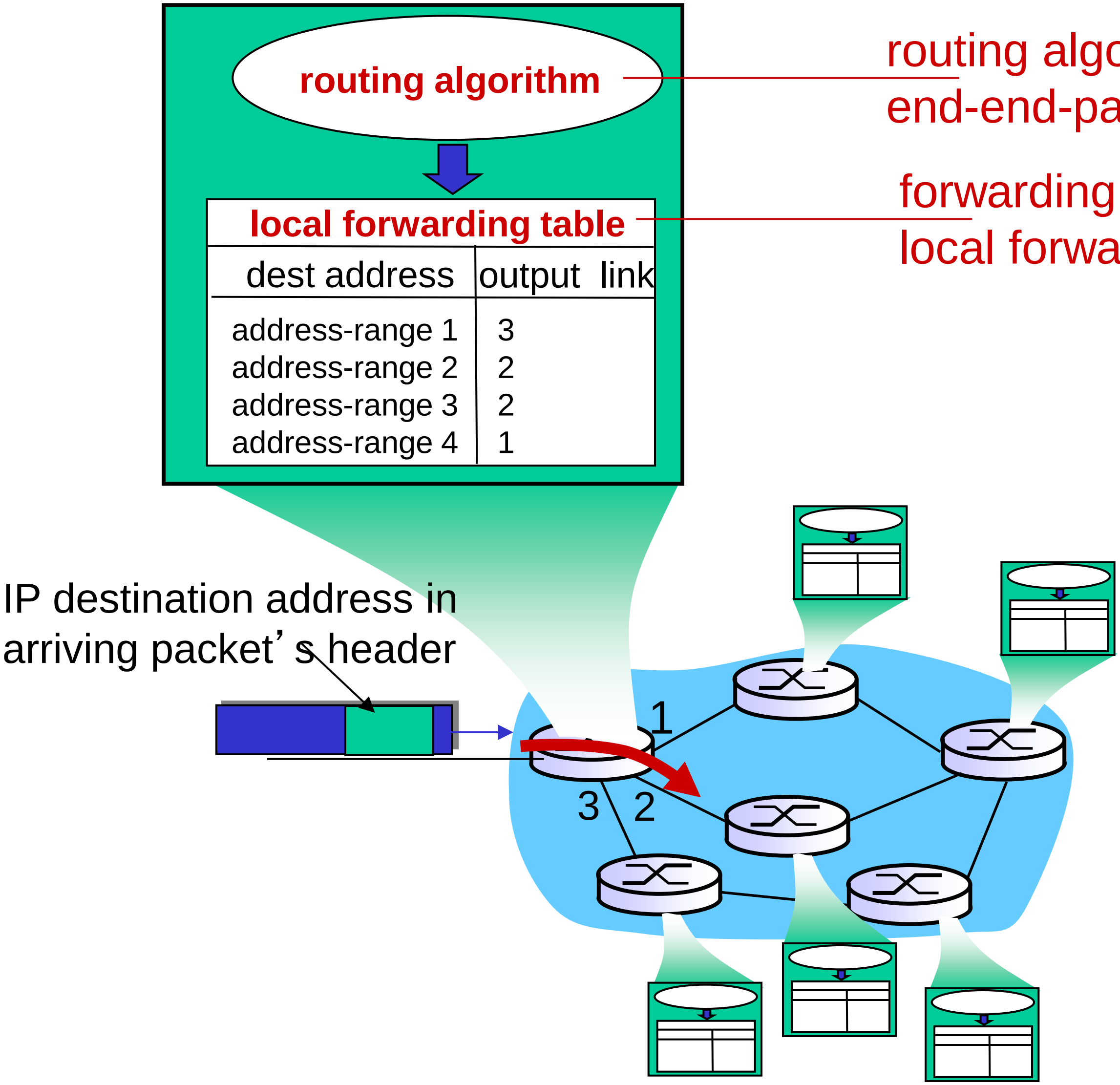
ICMPv6

- New version of ICMP for IPv6
  - added new message types, e.g., “Packet Too Big”
  - includes multicast group management functions that were previously part of Internet Group Management Protocol (IGMP)

Outline

- ✓ Network layer functions, mainly forwarding and routing
- ✓ Network layer services
- ✓ Datagram vs. Virtual circuit networks
- ✓ Router architectures and design issues
- ✓ IPv4 (incl. fragmentation)
- ✓ Internet addressing, DHCP and NAT
- ✓ IPv6
- ✓ ICMP
- ❖ Routing algorithms (link state, distance vector, hierarchical)
- ❖ Routing in the Internet (OSPF, BGP)

Interplay between Routing & Forwarding



routing algorithm

local forwarding table

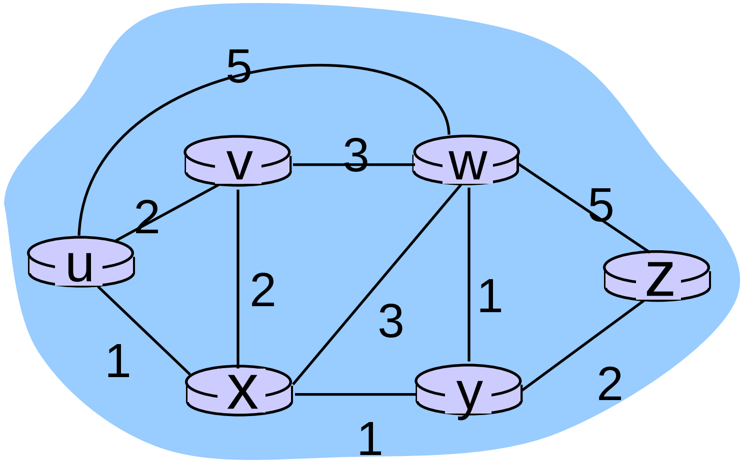
dest address	output link
address-range 1	3
address-range 2	2
address-range 3	2
address-range 4	1

routing algorithm determines end-end-path through network

forwarding table determines local forwarding at this router

IP destination address in arriving packet's header

Graph Abstraction

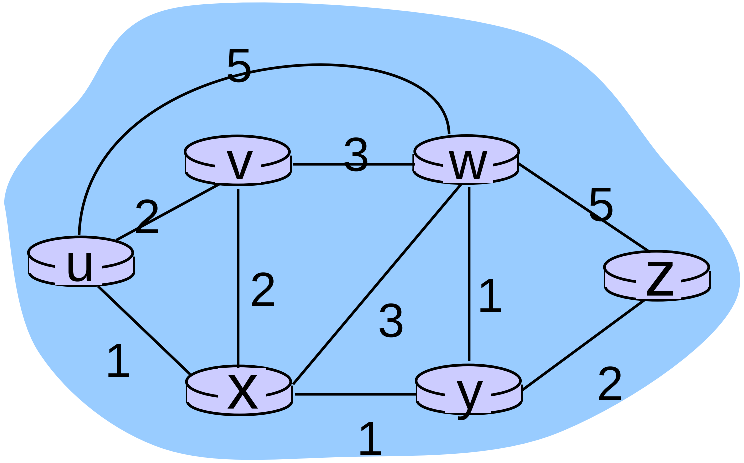


graph: G = (N,E)

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) }

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
or inversely related to congestion, or
some other metric or combination thereof

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 (more generally, between a given pair of nodes/routers)?
Routing algorithm: algorithm that finds that least cost path

Routing Algorithms: Various Classifications

Global vs. Decentralized state/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
- e.g., "distance vector" algorithms

Static vs. Dynamic

static:

- manual set routes, assume routes change at human timescales

dynamic:

- routes change more quickly in response to topology or load changes
  - periodic
  - event-driven, e.g., in response to link cost changes

Load sensitive vs. load insensitive

Link-State Routing Algorithms

- net topology, link costs known to all nodes
    - accomplished via "link state broadcast"
    - all nodes have same info
- each node computes least cost paths from itself ("source") to all other nodes
    - gives *forwarding table* for that node
- *Dijkstra's algorithm:* iterative, i.e., 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

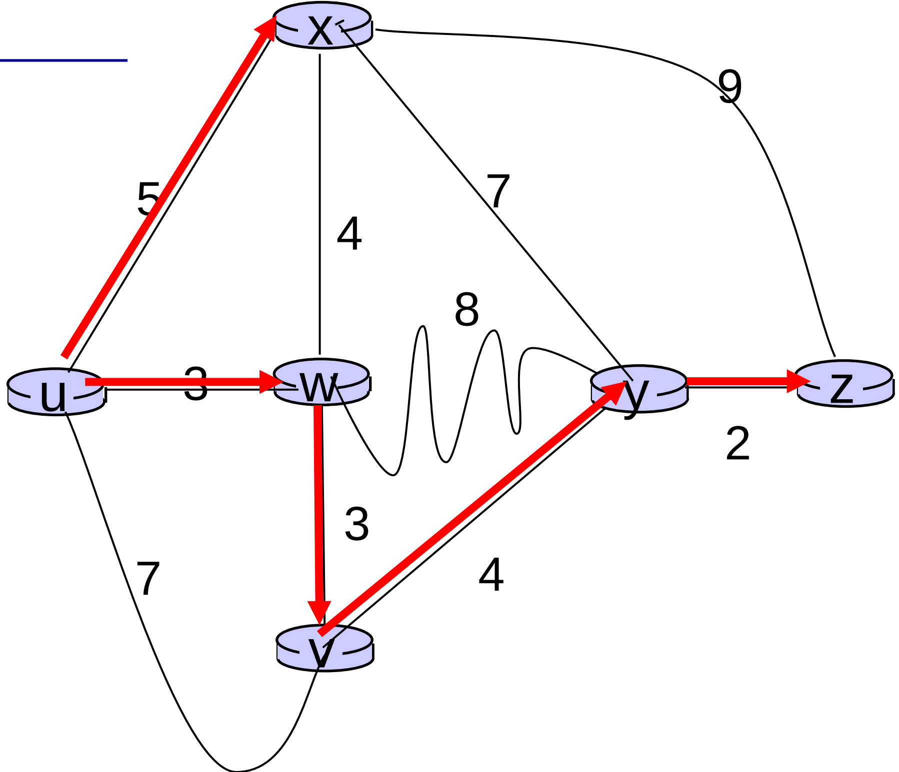
```
Initialization:
  N' = {u}
  for all nodes v
    if v adjacent to u
        then $D(v) = c(u,v)$
    else $D(v) = \infty$

Loop
  find w not in N' such that $D(w)$ is a minimum
  add w to N'
  update $D(v)$ for all v adjacent to w and not in N':
      $D(v) = \min( D(v), D(w) + c(w,v) )$
  /* new cost to v is either old cost to v or known
   shortest path cost to w plus cost from w to v */
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					

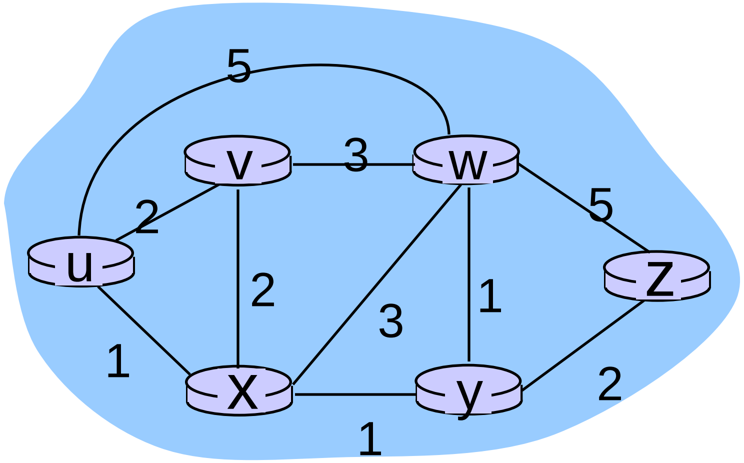
notes:

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



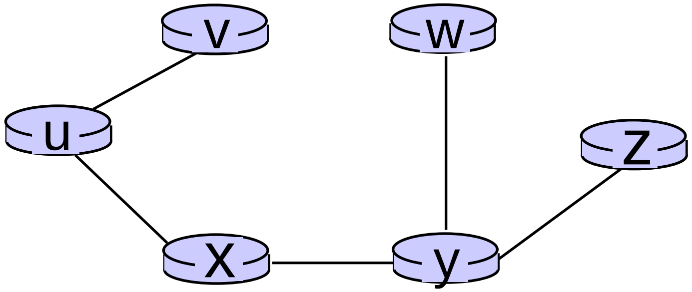
Dijkstra's Algorithm: Another Example

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: Another 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)

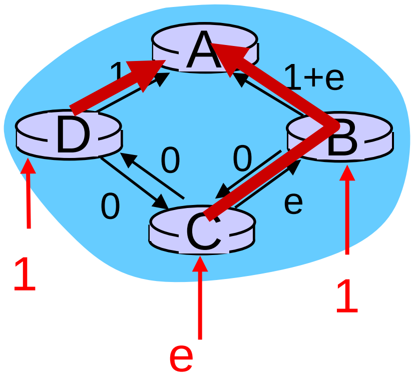
Dijkstra's Algorithm: Discussion

algorithm complexity: n nodes

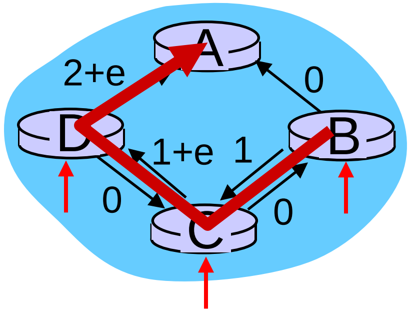
- each iteration: need to check all nodes, w, not in N'
- $n(n+1)/2$ comparisons: $O(n^2)$
- more efficient implementations possible: $O(nlogn)$

oscillations possible (not a unique problem with LS/Dijkstra though):

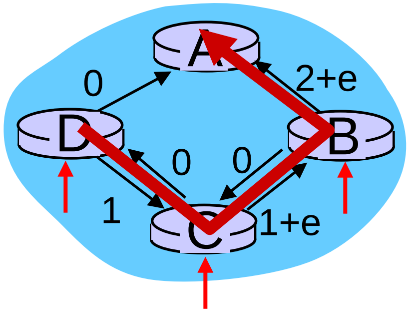
- e.g., suppose link cost equals amount of carried traffic:



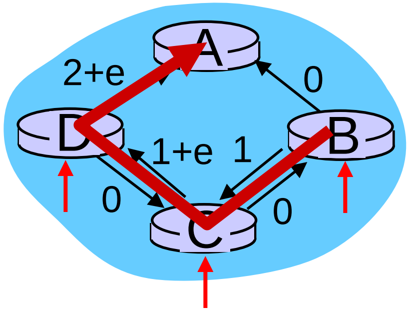
initially



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

Distance Vector Algorithms

A class of decentralised routing algorithms that are based on *Bellman-Ford equation (dynamic programming)*

let

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

then

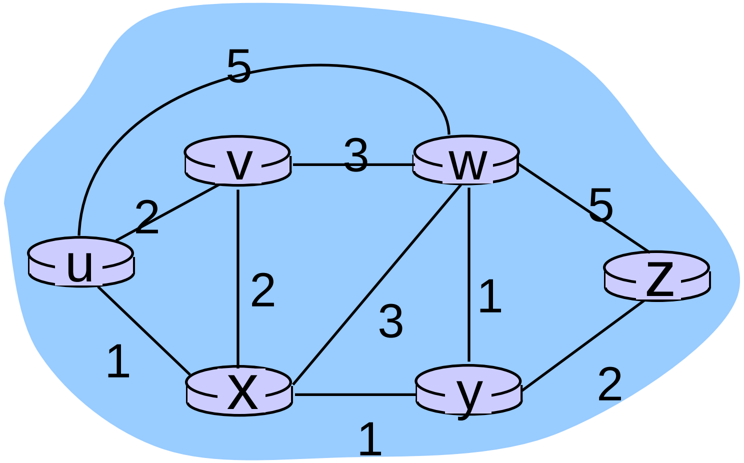
$$d_x(y) = \min_{v} \{c(x,v) + d_v(y) \}$$

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:

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

neighbour providing minimum distance estimate is chosen as the next hop and used in forwarding table

Distance Vector Algorithm

- node x:
  - knows cost to each neighbor v: $c(x,v)$
  - maintains its neighbours' distance vectors. For each neighbour v, x maintains $\mathbf{D}_v = [D_v(y): y \in N]$
- $D_x(y)$ = estimate of least cost from x to y
  - x computes distance vector $\mathbf{D}_x = [D_x(y): y \in N]$ based on $c(x,v)$ and D_v from all neighbours v using the Bellman-Ford equation

Distance Vector Algorithm (2)

key idea:

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

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

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

Distance Vector Algorithm (3)

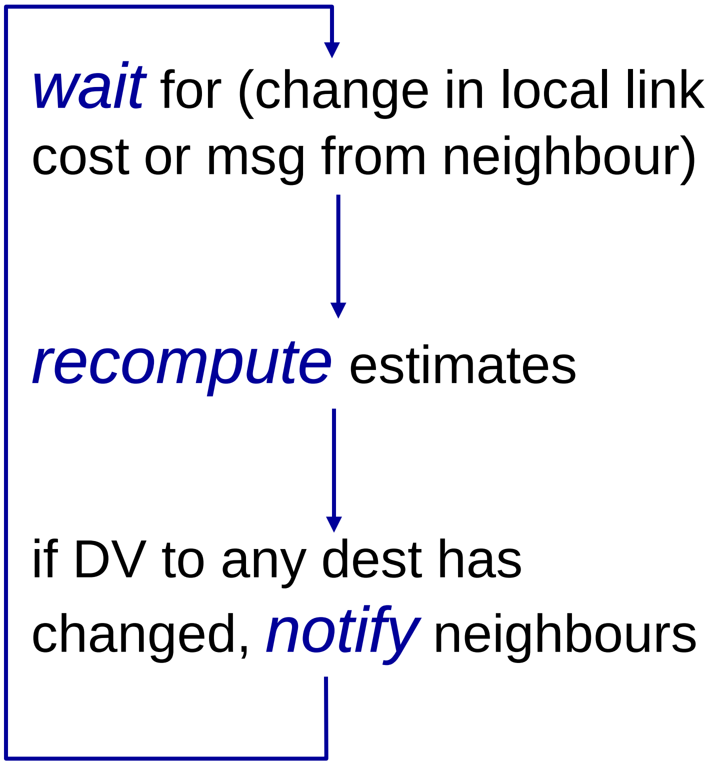
iterative, asynchronous: each local iteration caused by:

- local link cost change
- DV update message from neighbour

distributed:

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

each node:



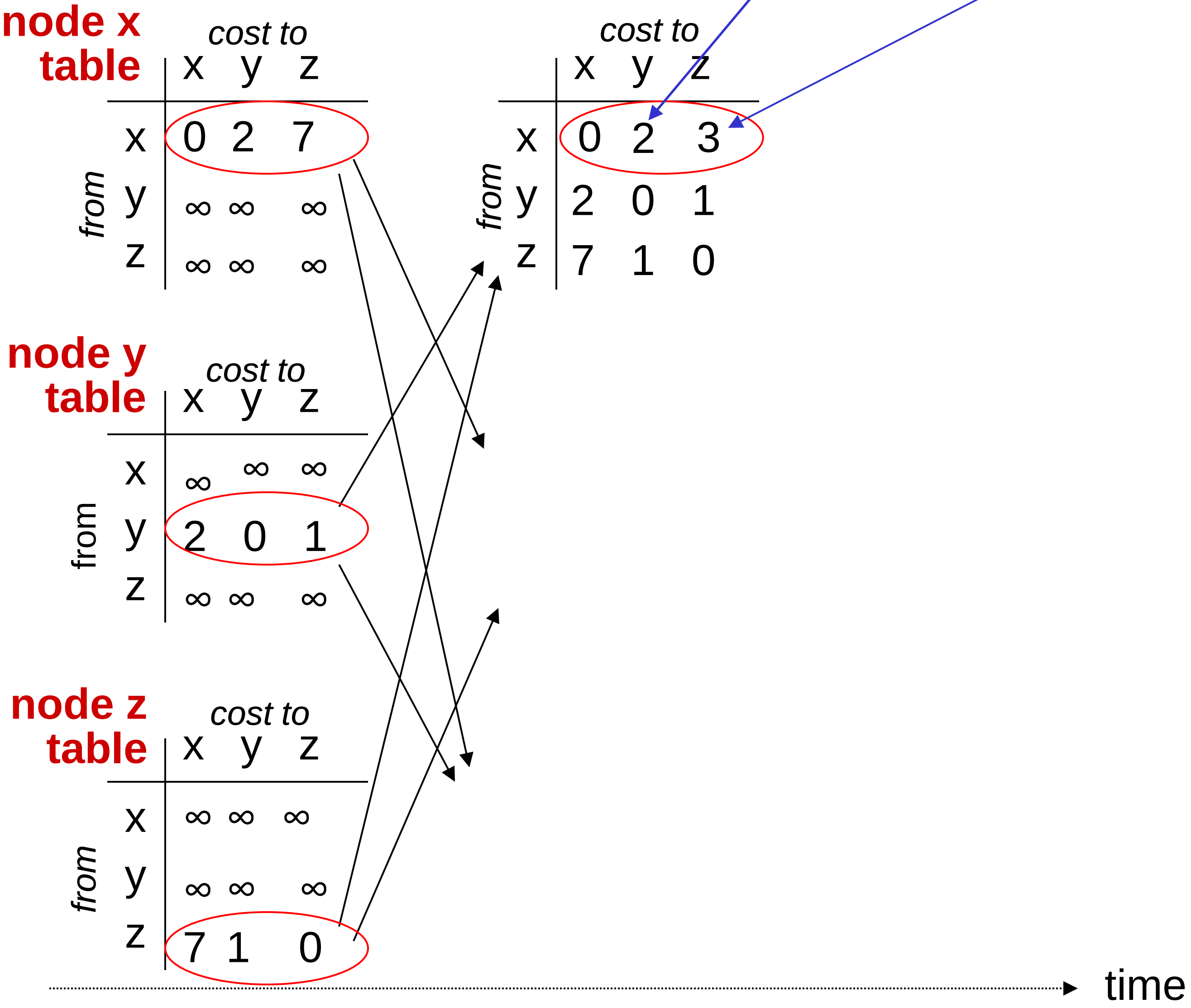
wait for (change in local link cost or msg from neighbour)

recompute estimates

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

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

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



node x table

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

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

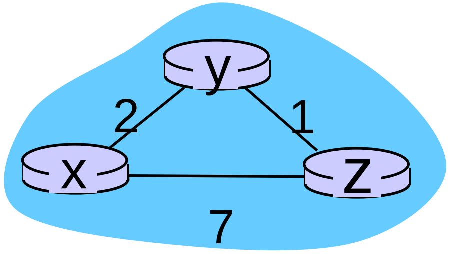
node y table

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

node z table

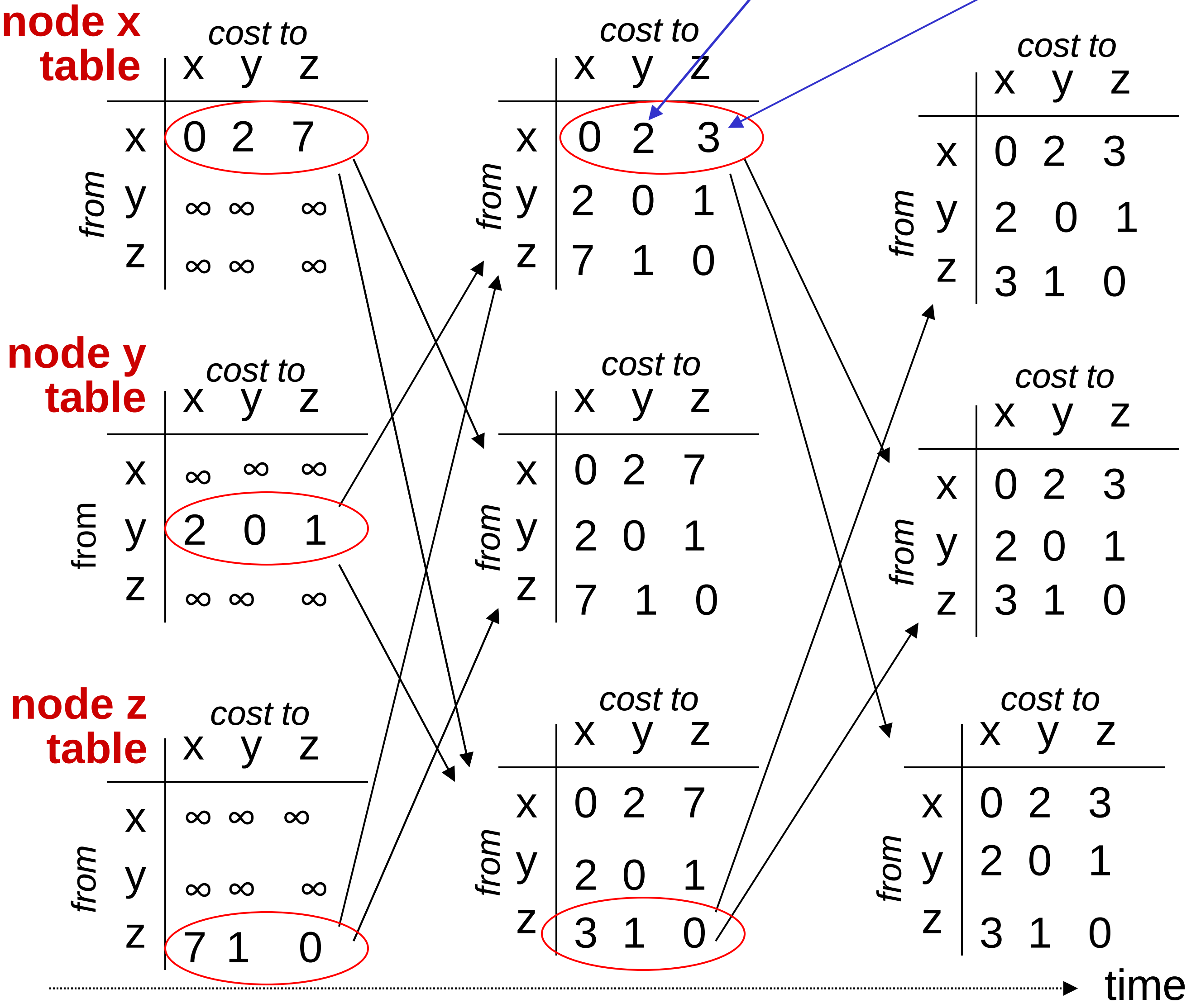
	cost to		
from	x	y	z
x	∞	∞	∞
y	∞	∞	∞
z	7	1	0

time



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

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



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

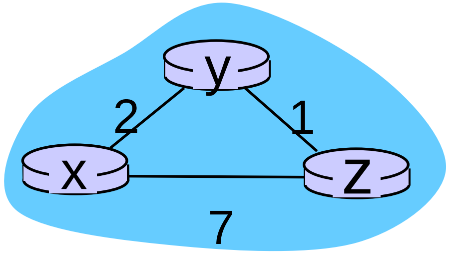
node z table

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

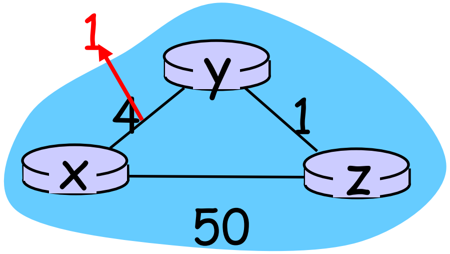
time



Distance Vector: Link Cost Changes

link cost changes:

- node detects local link cost change
- updates routing info, recalculates distance vector
- if DV changes, notifies neighbours



"good news travels fast"

t_0 : *y* detects link-cost change, updates its DV, informs its neighbours.

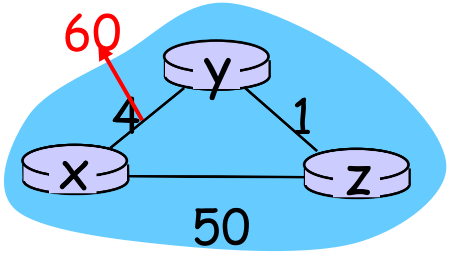
t_1 : *z* receives update from *y*, updates its table, computes new least cost to *x*, sends its neighbours its DV.

t_2 : *y* receives *z*'s update, updates its distance table. *y*'s least costs do *not* change, so *y* does *not* send a message to *z*.

Distance Vector: Link Cost Changes

link cost changes:

- node detects local link cost change
- *bad news travels slow* – "counting to infinity" problem!
- 44 iterations before algorithm stabilizes for the example on the right



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 counting to infinity problem?

Comparison of LS and DV Algorithms

message complexity

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

speed of convergence

- ***LS:*** O(n^2) algorithm requires O(nE) msgs
  - may have oscillations if metric load sensitive (same applies for vanilla DV too)
- ***DV:*** convergence time varies
  - routing loops possible, e.g., counting-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 → errors propagate through the network

Hierarchical Routing

our routing study thus far assumes:

- all routers identical
- network structure "flat"

… *not* true in practice

scale: with 600 million destinations:

- can't store all destinations 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

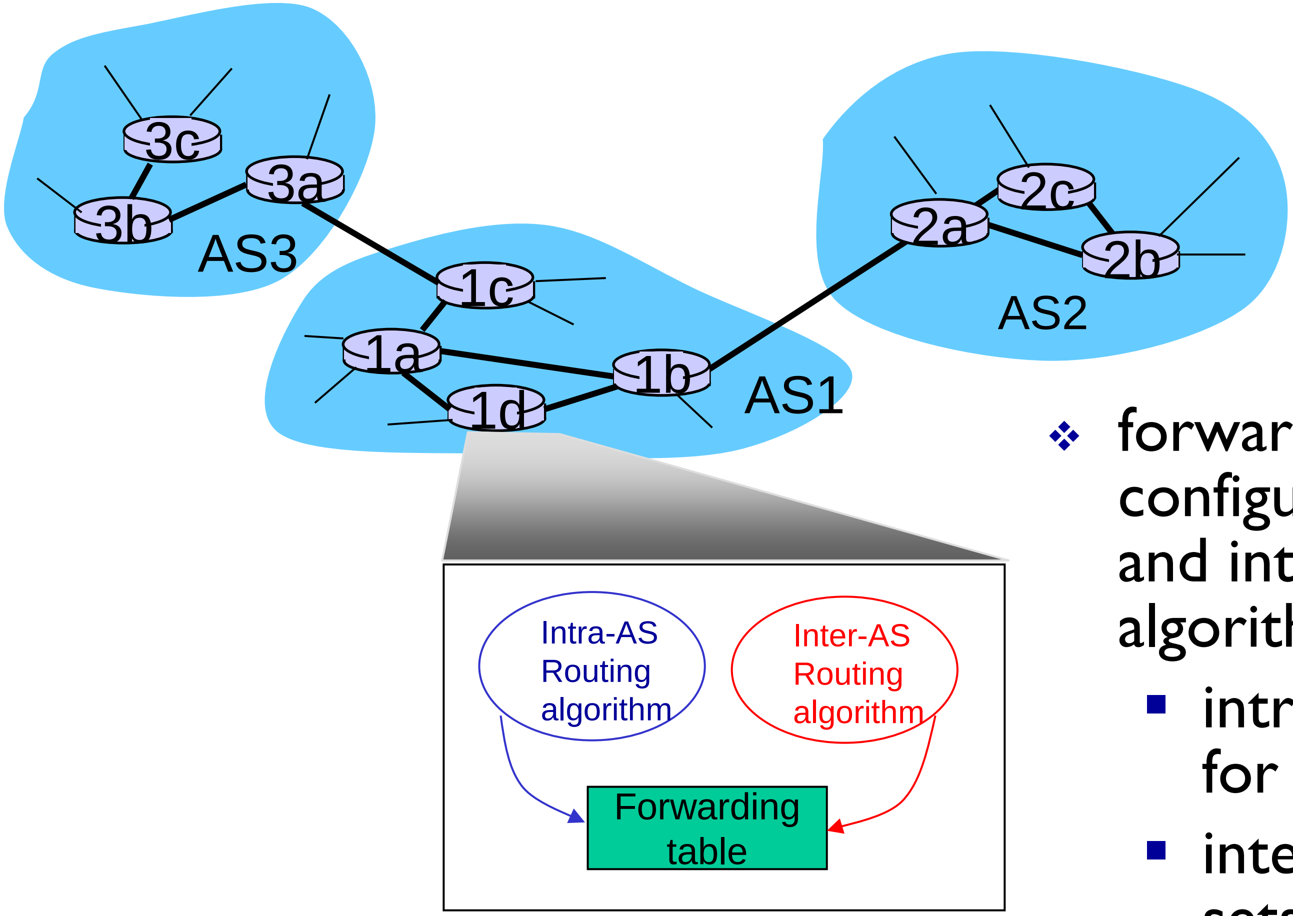
Hierarchical Routing

- 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 protocols

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

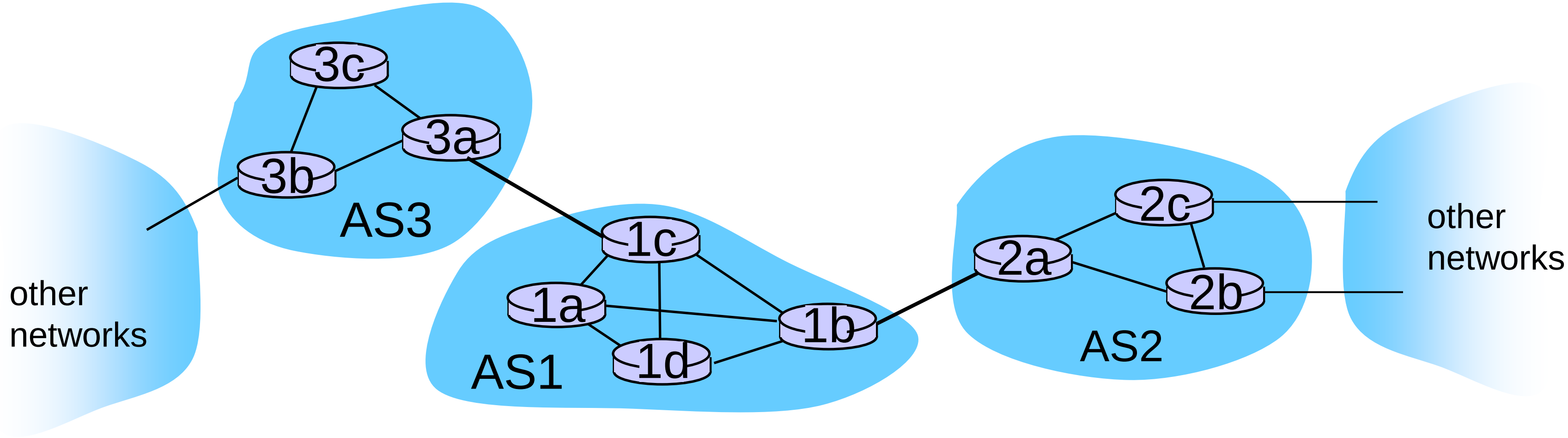
Inter-AS Tasks

- suppose router in AS1 receives datagram destined outside of AS1:
  - router should forward packet to gateway router, but which one?

AS1 must:

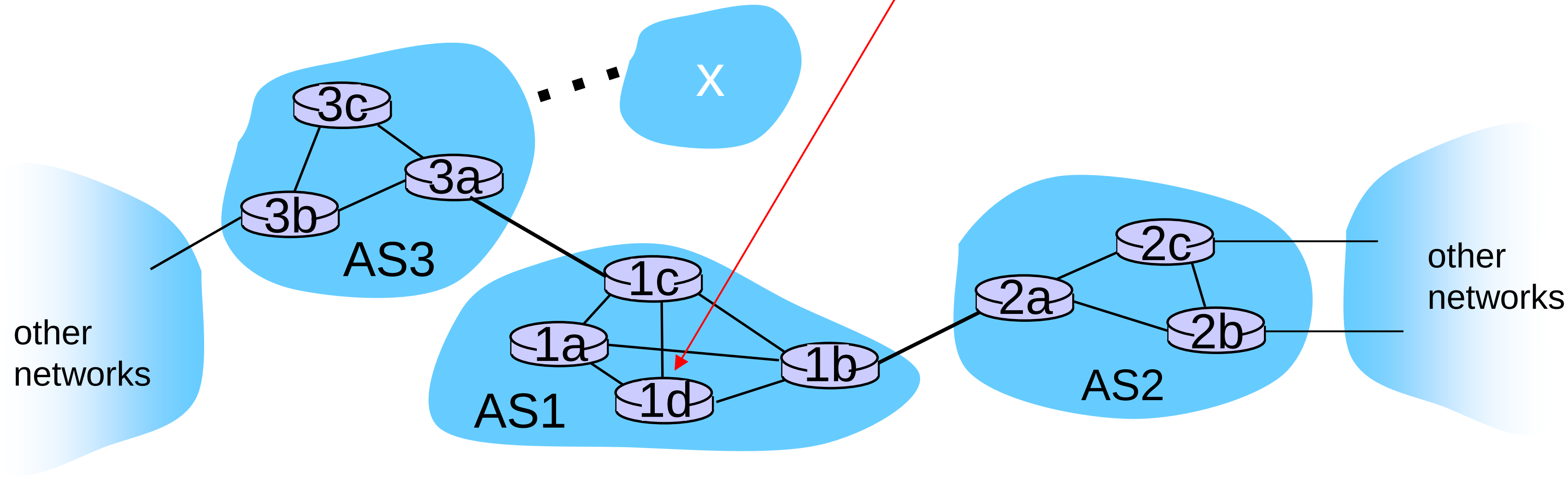
1. learn which dests are reachable through AS2, which through AS3
2. propagate this reachability info to all routers in AS1

job of inter-AS routing!



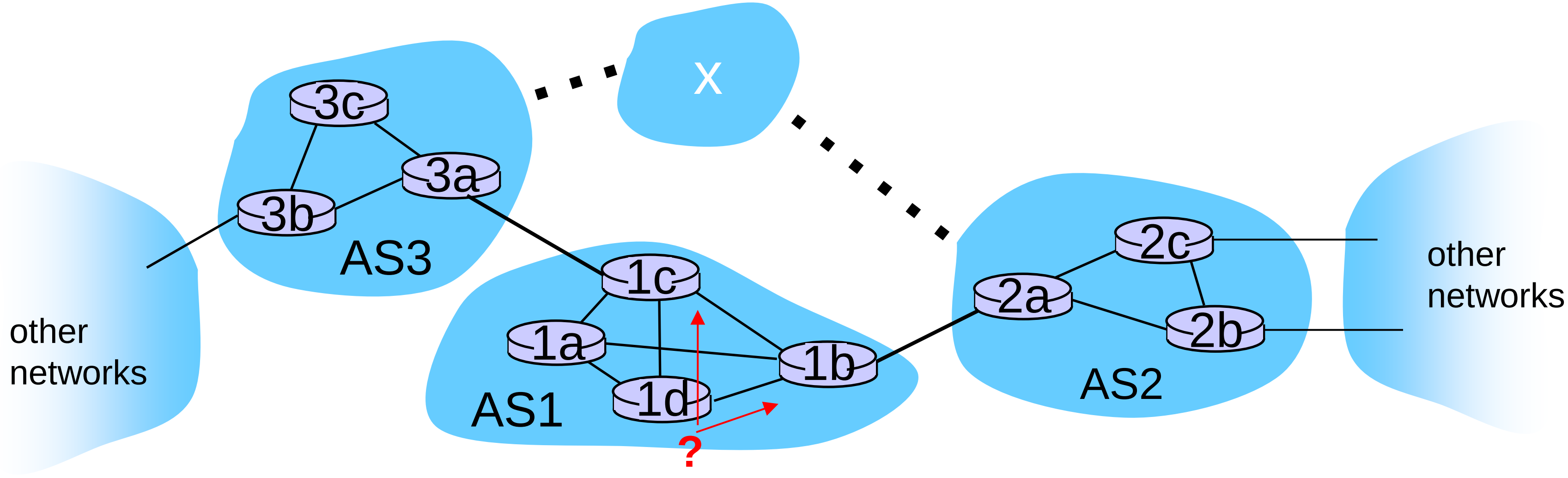
Example: setting forwarding table in router 1d

- 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

Outline

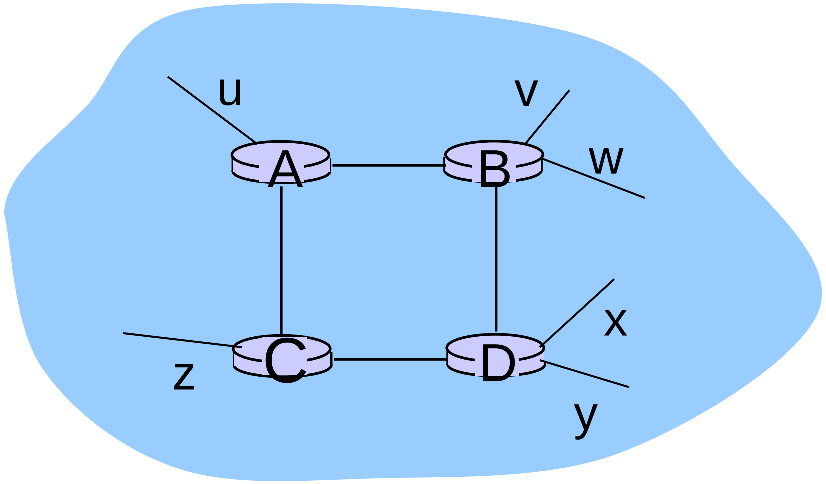
- ✓ Network layer functions, mainly forwarding and routing
- ✓ Network layer services
- ✓ Datagram vs. Virtual circuit networks
- ✓ Router architectures and design issues
- ✓ IPv4 (incl. fragmentation)
- ✓ Internet addressing, DHCP and NAT
- ✓ IPv6
- ✓ ICMP
- ✓ Routing algorithms (link state, distance vector, hierarchical)
- ❖ Routing in the Internet (RIP, OSPF, BGP)

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)

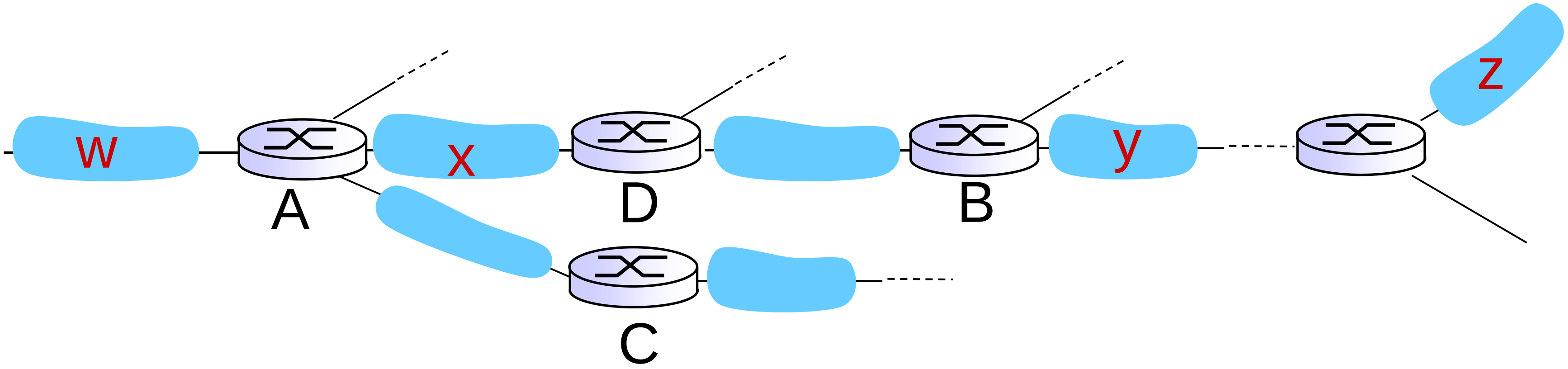
- included in BSD-UNIX distribution in 1982
- distance vector algorithm
  - distance metric: # hops (max = 15 hops), each link has cost 1
  - DVs exchanged with neighbors every 30 sec in response message (aka advertisement)
  - each advertisement: list of up to 25 destination *subnets* *(in IP addressing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



from router A to destination *subnets:*

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

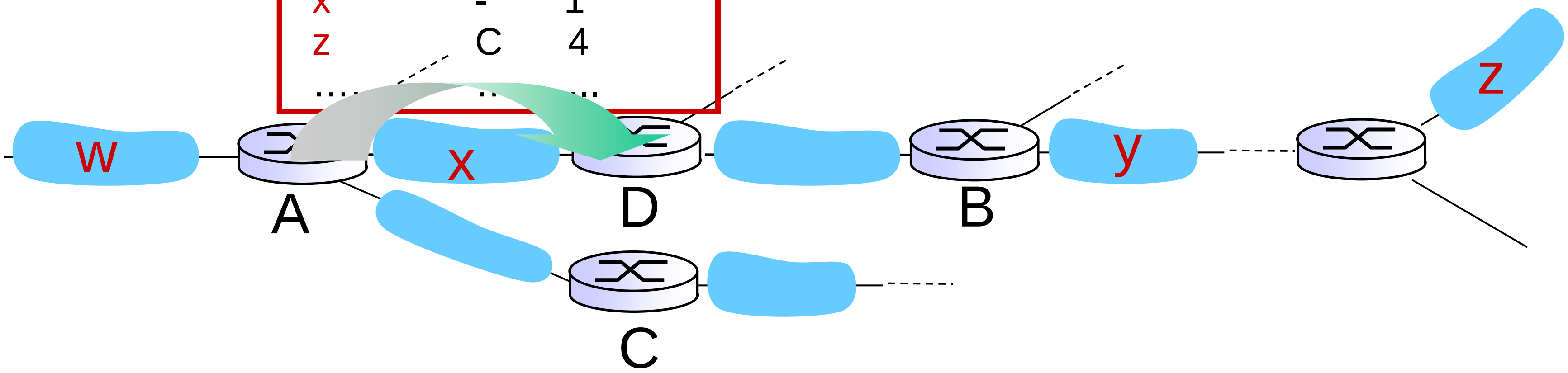
RIP: example



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
….	…	….

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