

# Network-layer functions

*Recall: two network-layer functions:*

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

*data plane*

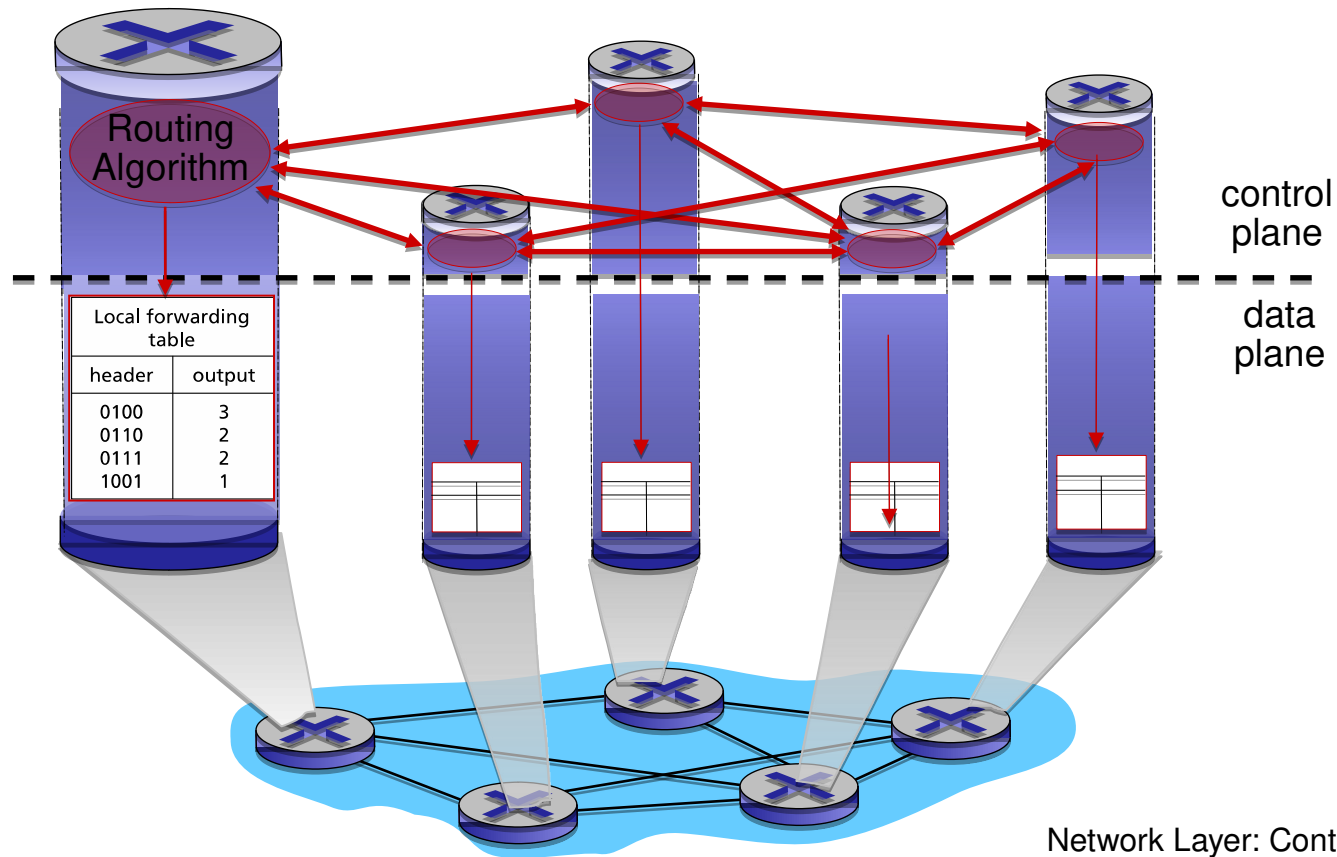
*control plane*

*Two approaches to structuring network control plane:*

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

# Per-router control plane

Individual routing algorithm components *in each and every router* interact with each other in control plane to compute forwarding tables

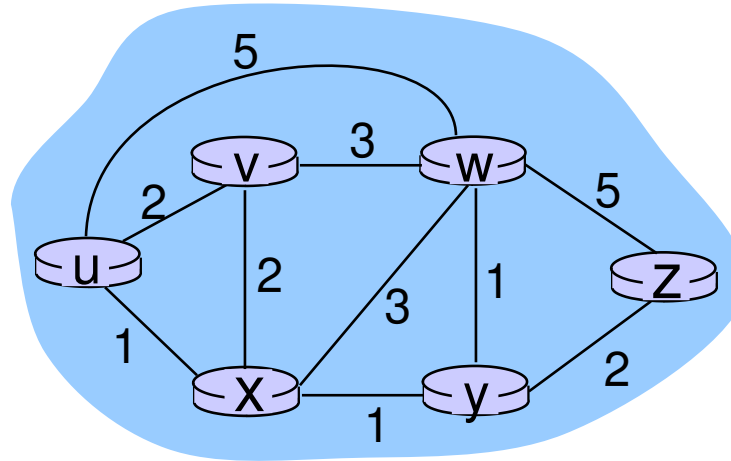


# Routing protocols

*Routing protocol goal:* determine “good” paths (equivalently, routes), from sending hosts to receiving host, through network of routers

- path: sequence of routers packets will traverse in going from given initial source host to given final destination host
- “good”: least “cost”, “fastest”, “least congested”
- routing: a “top-10” networking challenge!

# Graph abstraction of the network



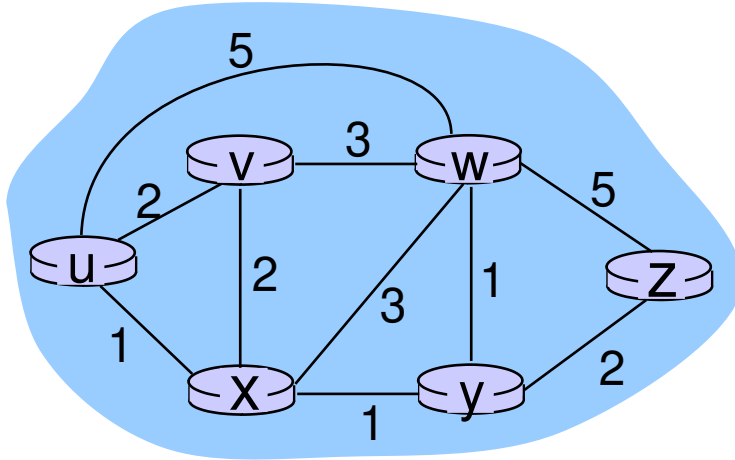
graph:  $G = (N, E)$

$N$  = 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



$c(x, x') = \text{cost of link } (x, x')$   
e.g.,  $c(w, z) = 5$

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

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

**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

# 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;  $= \infty$  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

# Dijkstra'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  $D(v) = \infty$

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

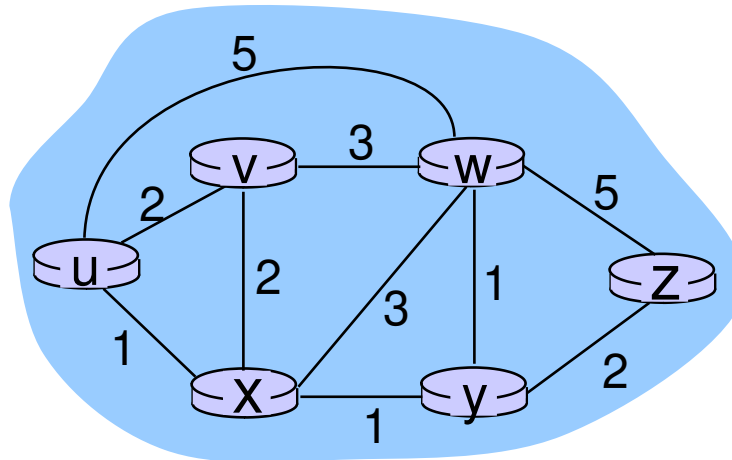
14 shortest path cost to  $w$  plus cost from  $w$  to  $v$  \*/

15 **until all nodes in  $N'$**



# Dijkstra's algorithm: example

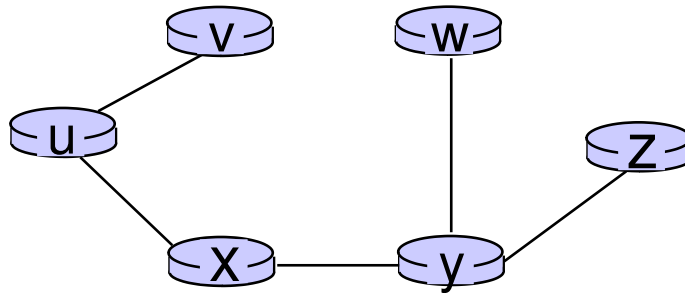
Step	N'	D(v),p(v)	D(w),p(w)	D(x),p(x)	D(y),p(y)	D(z),p(z)
0	u	2,u	5,u	1,u	$\infty$	$\infty$
1	ux	2,u	4,x		2,x	$\infty$
2	uxy	2,u	3,y			4,y
3	uxyv		3,y			4,y
4	uxyvw					4,y
5	uxyvwz					



\* Check out the online interactive exercises for more examples: [http://gaia.cs.umass.edu/kurose\\_ross/interactive/](http://gaia.cs.umass.edu/kurose_ross/interactive/)

# 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)

# Distance vector algorithm

*Bellman-Ford equation (dynamic programming)*

let

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

then

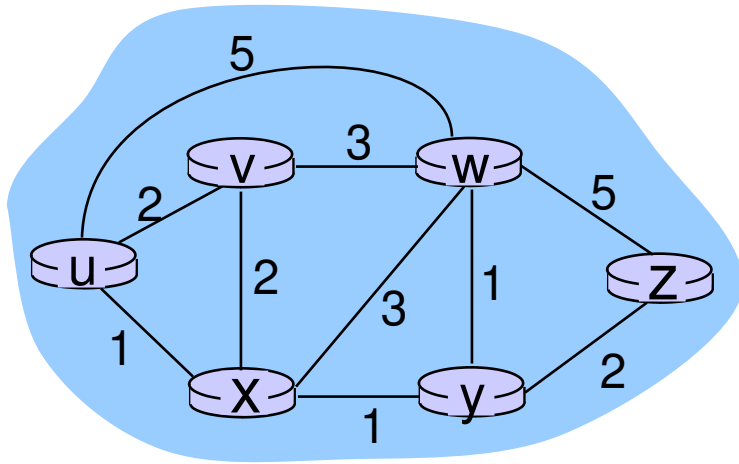
$$d_x(y) = \min_v \{ c(x,v) + d_v(y) \}$$

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:

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

node achieving minimum 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

## *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,*

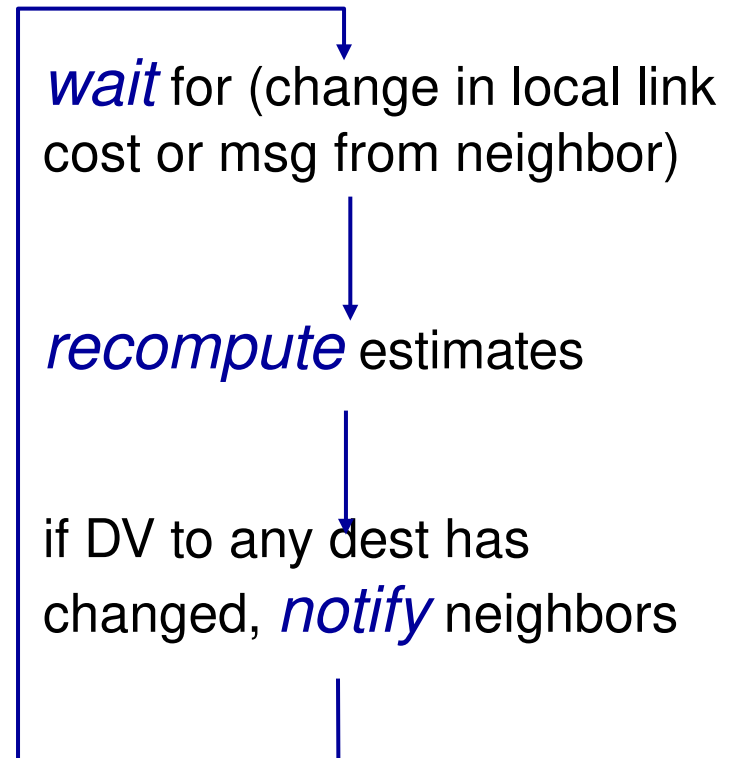
*asynchronous:* each local iteration caused by:

- 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:*



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

$$= \min\{2+0, 7+1\} = 2$$

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

$$= \min\{2+1, 7+0\} = 3$$

**node x  
table**

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

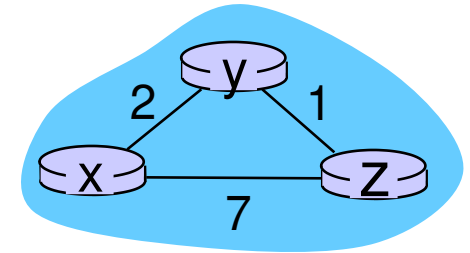
		cost to		
		x	y	z
from	x	0	2	3
	y	2	0	1
	z	7	1	0

**node y  
table**

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

**node z  
table**

		cost to		
		x	y	z
from	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**

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

**node y  
table**

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

**node z  
table**

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

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

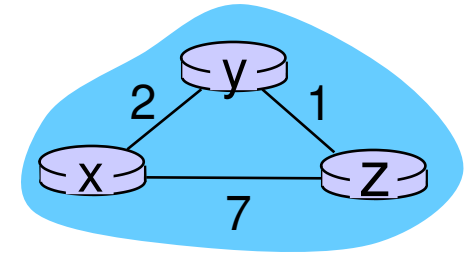
		cost to		
		x	y	z
from	x	0	2	7
	y	2	0	1
	z	7	1	0

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

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

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

		cost to		
		x	y	z
from	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
  - *errors propagate through network*

# Making routing scalable

our routing study thus far - idealized

- all routers identical
- network “flat”

... *not* true in practice

*scale:* with billions of 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

# Internet approach to scalable routing

aggregate routers into regions known as  
“autonomous systems” (AS) (a.k.a. “domains”)

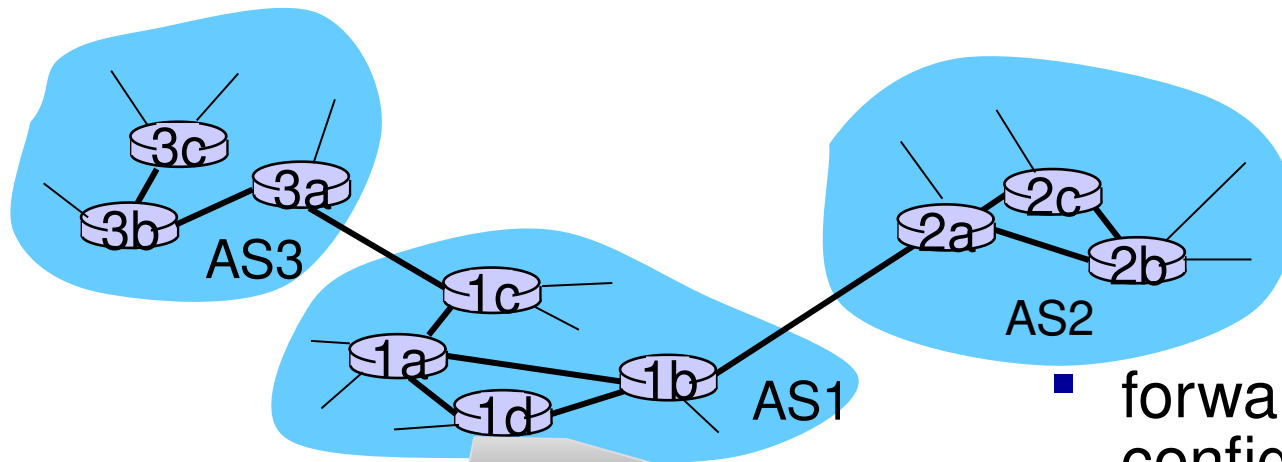
## intra-AS routing

- routing among hosts, routers in same AS (“network”)
- all routers in AS must run *same* intra-domain protocol
- routers in *different* AS can run *different* intra-domain routing protocol
- gateway router: at “edge” of its own AS, has link(s) to router(s) in other AS'es

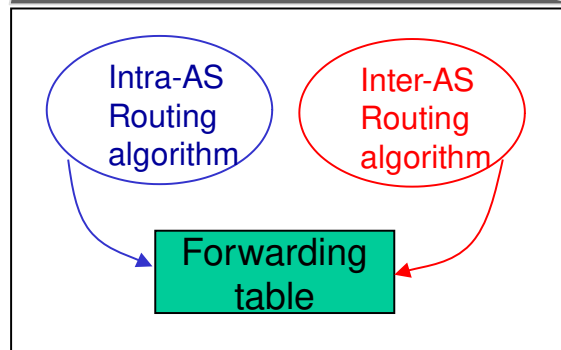
## inter-AS routing

- routing among AS'es
- gateways perform inter-domain routing (as well as intra-domain routing)

# Interconnected ASes



- forwarding table configured by both intra- and inter-AS routing algorithm
  - intra-AS routing determine entries for destinations within AS
  - inter-AS & intra-AS determine entries for external destinations



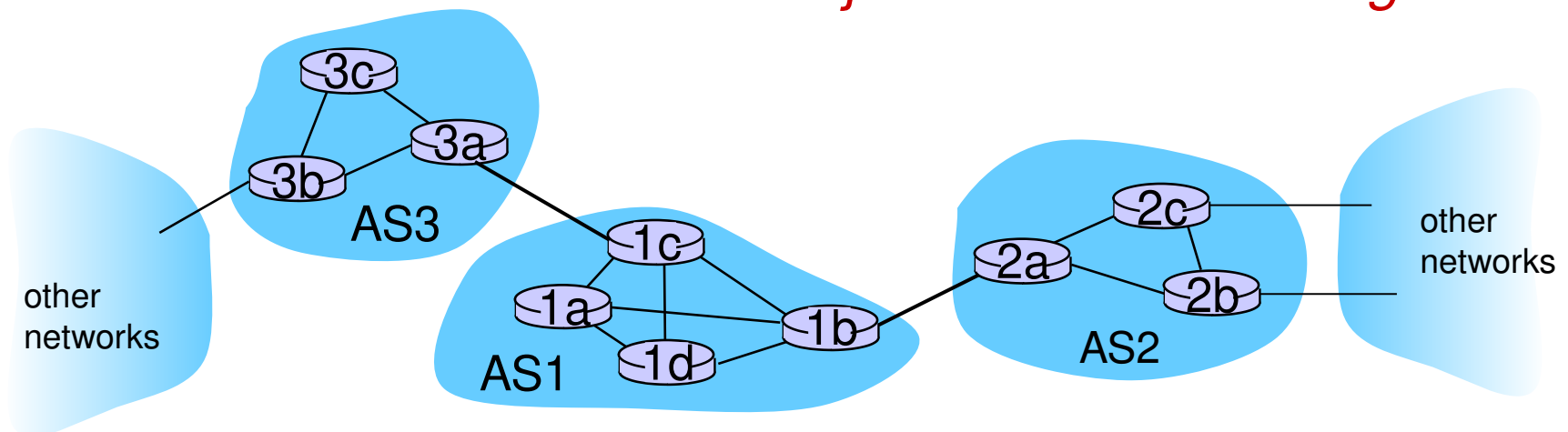
# Inter-AS tasks

- 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 destds are reachable through AS2, which through AS3
2. propagate this reachability info to all routers in AS1

*job of inter-AS 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 (IS-IS protocol essentially same as OSPF)
  - IGRP: Interior Gateway Routing Protocol (Cisco proprietary for decades, until 2016)

# 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 ASes
  - **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”*



# 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 5

## Network Layer:

### The Control Plane

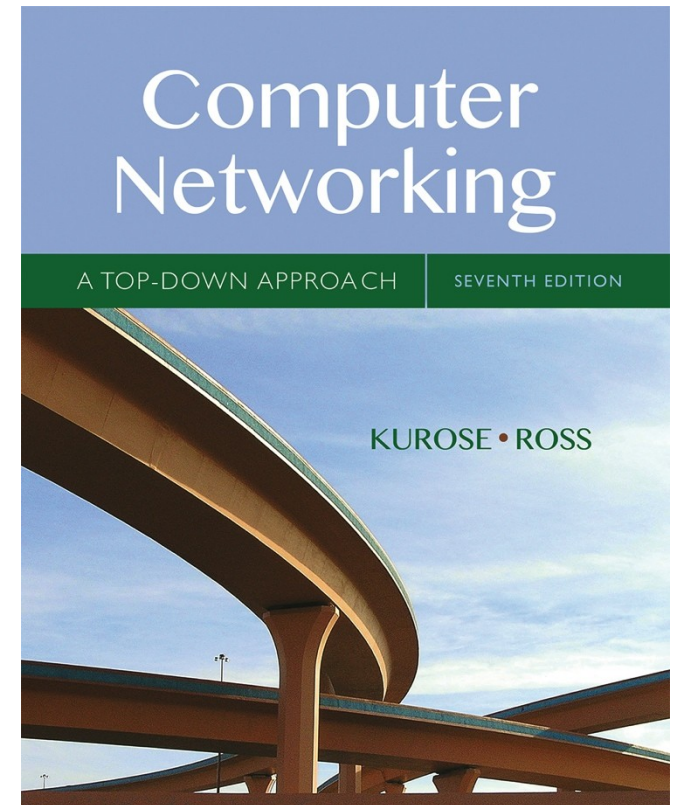
#### A note on the use of these Powerpoint slides:

We're making these slides freely available to all (faculty, students, readers). They're in PowerPoint form so you see the animations; and can add, modify, and delete slides (including this one) and slide content to suit your needs. They obviously represent a *lot* of work on our part. In return for use, we only ask the following:

- If you use these slides (e.g., in a class) that you mention their source (after all, we'd like people to use our book!)
- If you post any slides on a www site, that you note that they are adapted from (or perhaps identical to) our slides, and note our copyright of this material.

Thanks and enjoy! JFK/KWR

© All material copyright 1996-2016  
J.F Kurose and K.W. Ross, All Rights Reserved



## *Computer Networking: A Top Down Approach*

7<sup>th</sup> edition

Jim Kurose, Keith Ross  
Pearson/Addison Wesley  
April 2016