

# CMPE 150/L : Introduction to Computer Networks

Chen Qian

Computer Engineering  
UCSC Baskin Engineering

Lecture 14

# Two notes on routing algorithm

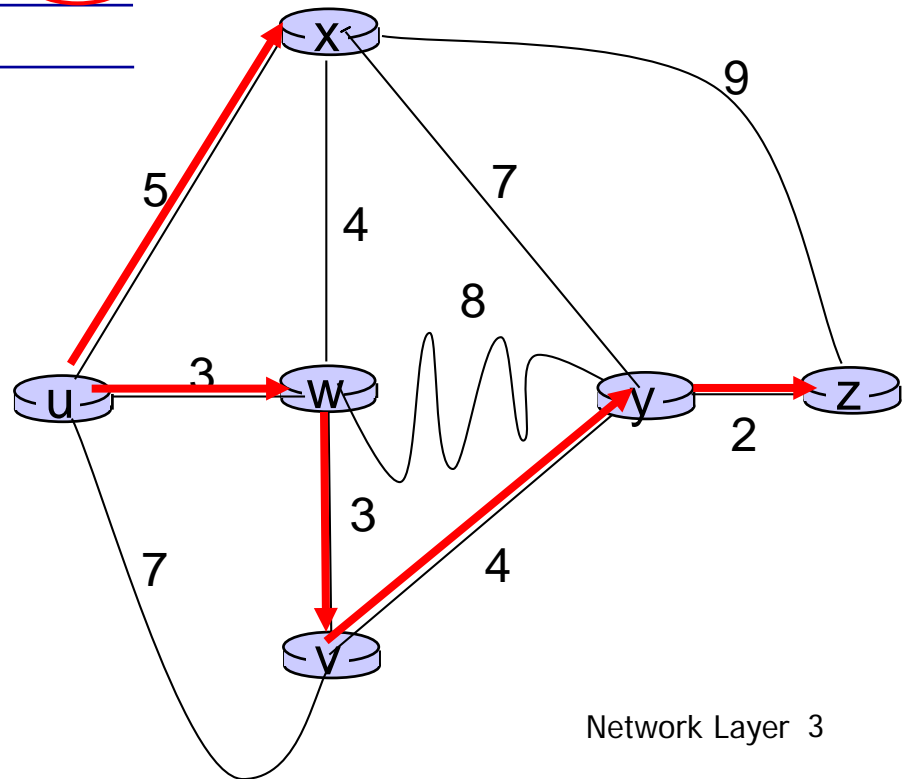
- ❑ Do not believe you can understand any routing protocol, e.g., link-state or distance vector, just by attending the lectures.
  - ❖ I didn't understand them when I was an undergrad.
  
- ❑ Best approach to learn: Try to write your own program to simulate the protocols.

# 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	$\infty$	$\infty$
1	uw	6,w		5,u	11,w	$\infty$
2	uwx	6,w			11,w	14,x
3	uwxv				10,v	14,x
4	uwxvy				12,y	
5	uwxvyz					

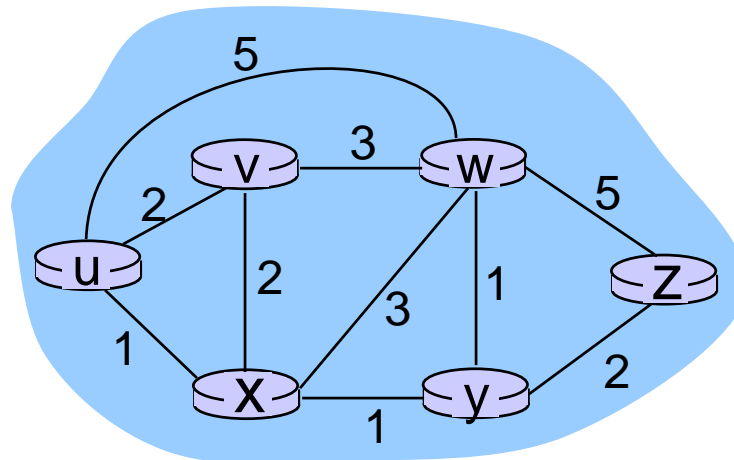
## notes:

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



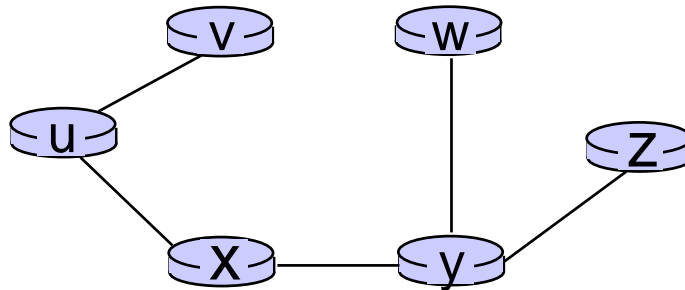
# Dijkstra's algorithm: another example

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	$\infty$	$\infty$
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					



# 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)

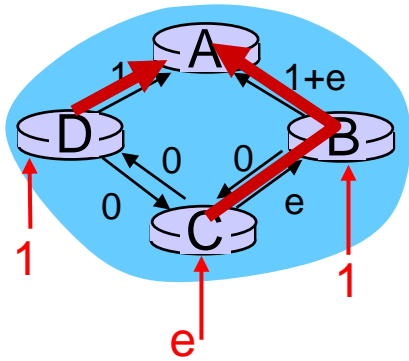
# Dijkstra's algorithm, discussion

*algorithm complexity:* n nodes

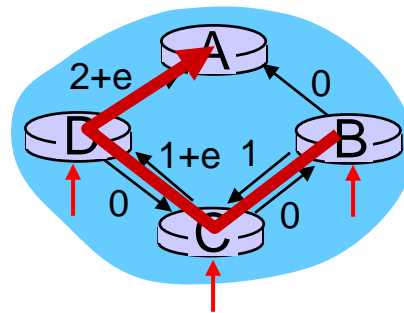
- ❖ each iteration: need to check all nodes, w, not in N
- ❖  $n(n+1)/2$  comparisons:  $O(n^2)$
- ❖ more efficient implementations possible:  $O(n \log n)$

*oscillations possible:*

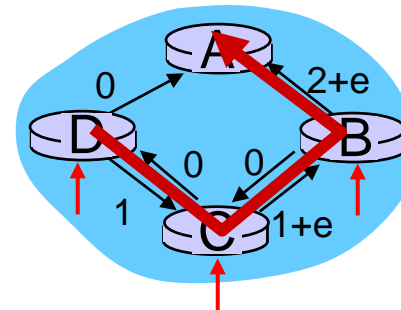
- ❖ e.g., support link cost equals amount of carried traffic:



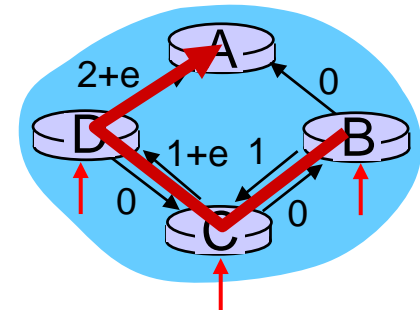
initially



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

# Distance vector algorithm

*Bellman-Ford equation (dynamic programming)*

let

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

then

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

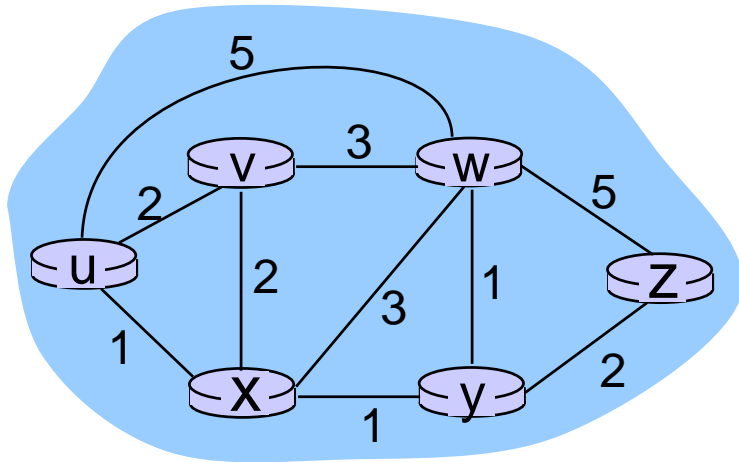
cost from neighbor  $v$  to destination  $y$

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:*

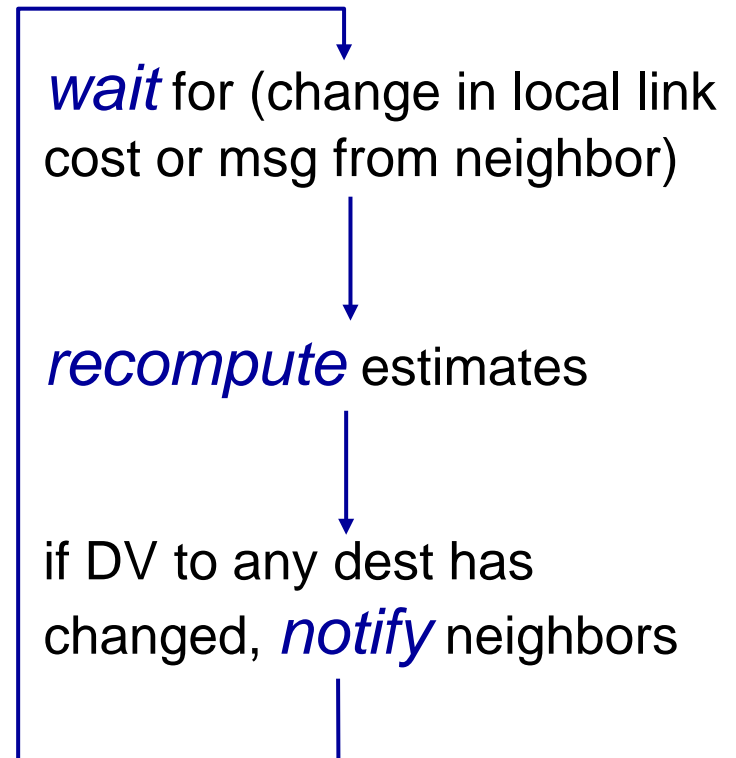
each local iteration  
caused by:

- ❖ 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	$\infty$	$\infty$	$\infty$
	z	$\infty$	$\infty$	$\infty$

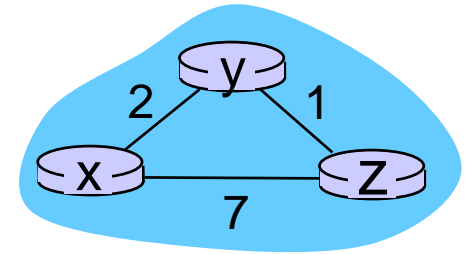
		cost to		
		x	y	z
from	x	0	2	3
	y	2	0	1
	z	7	1	0

**node y table**

		cost to		
		x	y	z
from	x	$\infty$	$\infty$	$\infty$
	y	2	0	1
	z	$\infty$	$\infty$	$\infty$

**node z table**

		cost to		
		x	y	z
from	x	$\infty$	$\infty$	$\infty$
	y	$\infty$	$\infty$	$\infty$
	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	$\infty$	$\infty$	$\infty$
	z	$\infty$	$\infty$	$\infty$

**node y table**

		cost to		
		x	y	z
from	x	$\infty$	$\infty$	$\infty$
	y	2	0	1
	z	$\infty$	$\infty$	$\infty$

**node z table**

		cost to		
		x	y	z
from	x	$\infty$	$\infty$	$\infty$
	y	$\infty$	$\infty$	$\infty$
	z	7	1	0

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

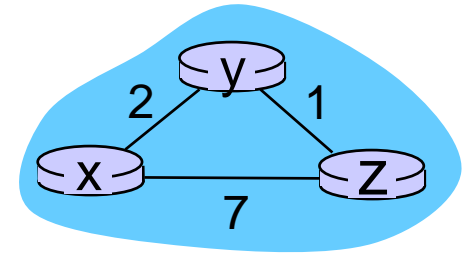
		cost to		
		x	y	z
from	x	0	2	7
	y	2	0	1
	z	7	1	0

		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

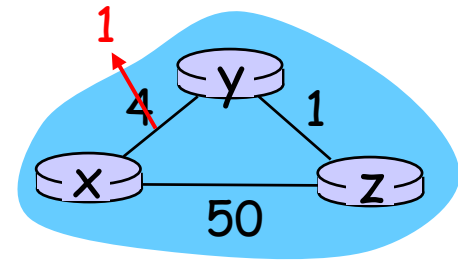


time

# Distance vector: link cost changes

## *link cost changes:*

- ❖ node detects local link cost change
- ❖ updates routing info, recalculates distance vector
- ❖ if DV changes, notify neighbors



“good  
news  
travels  
fast”

$t_0$ :  $y$  detects link-cost change, updates its DV, informs its neighbors.

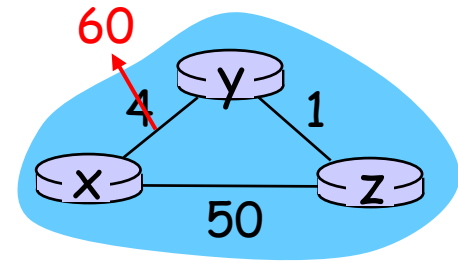
$t_1$ :  $z$  receives update from  $y$ , updates its table, computes new least cost to  $x$ , sends its neighbors its DV.

$t_2$ :  $y$  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* - “count to infinity” problem!
- ❖ 44 iterations before algorithm stabilizes: see text



## *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)



# 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

# 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

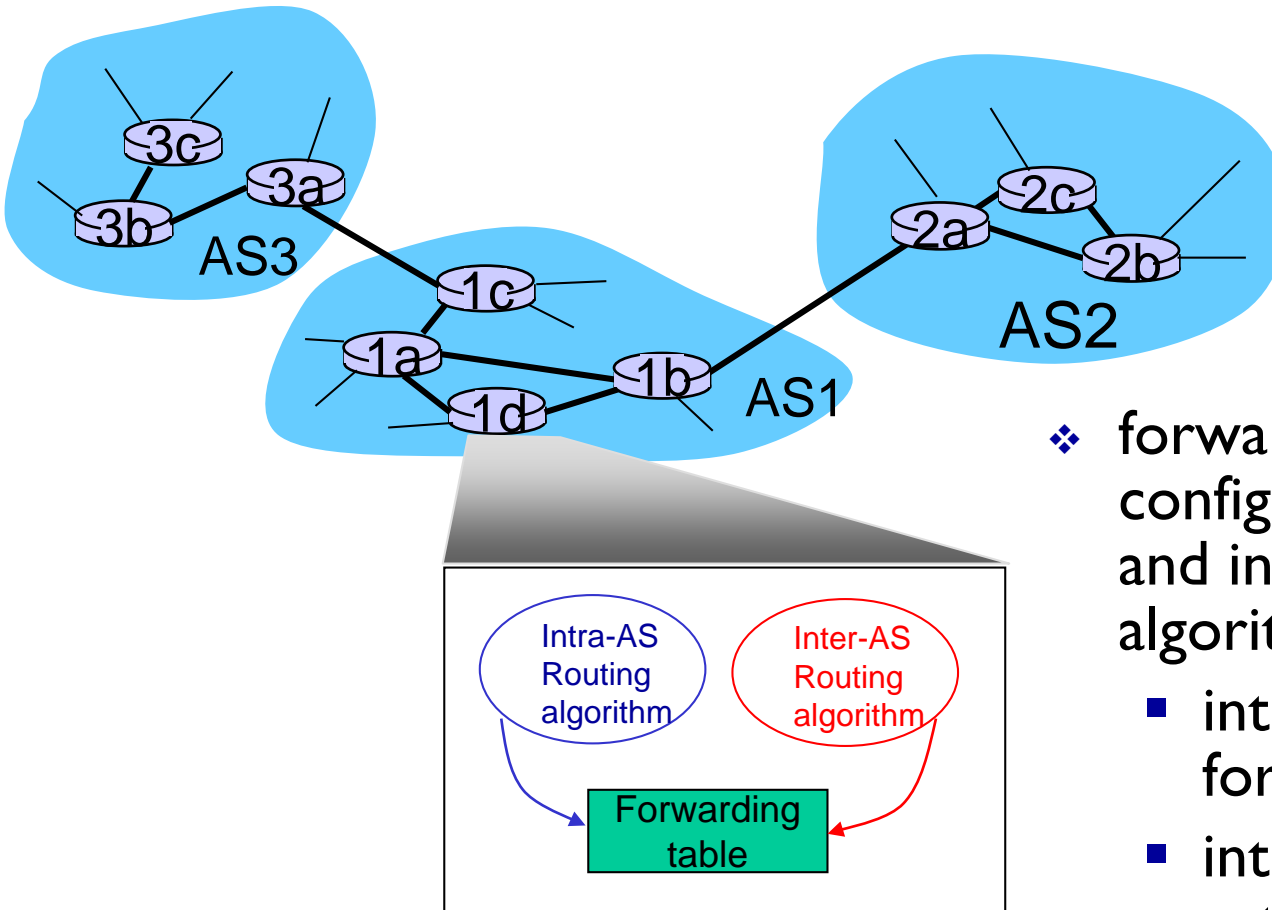
# Hierarchical routing

- ❖ 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

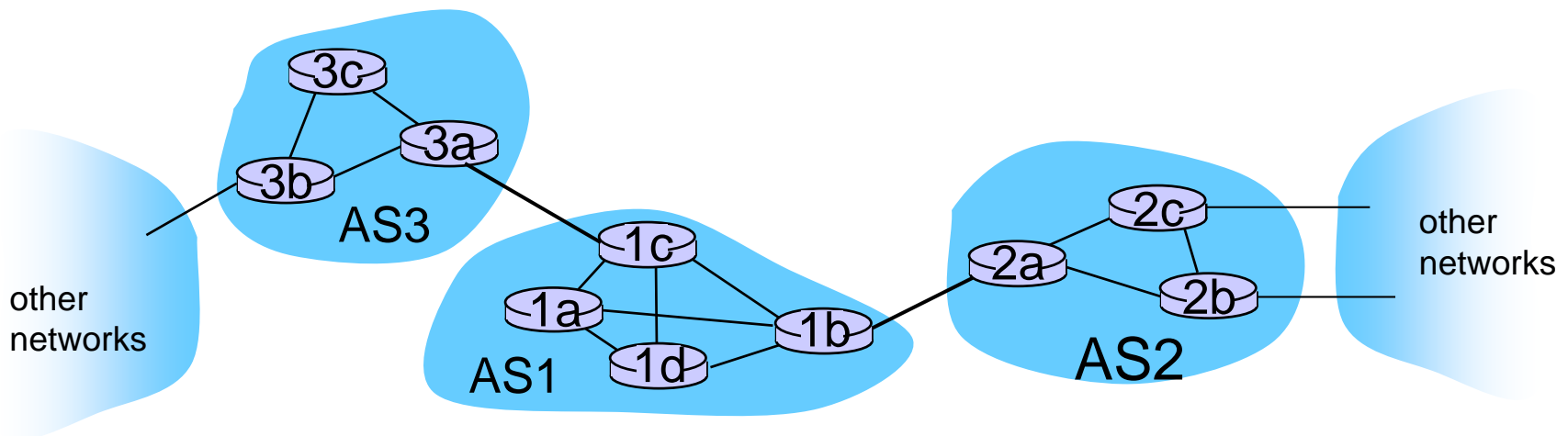
# Inter-AS tasks

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

*AS1 must:*

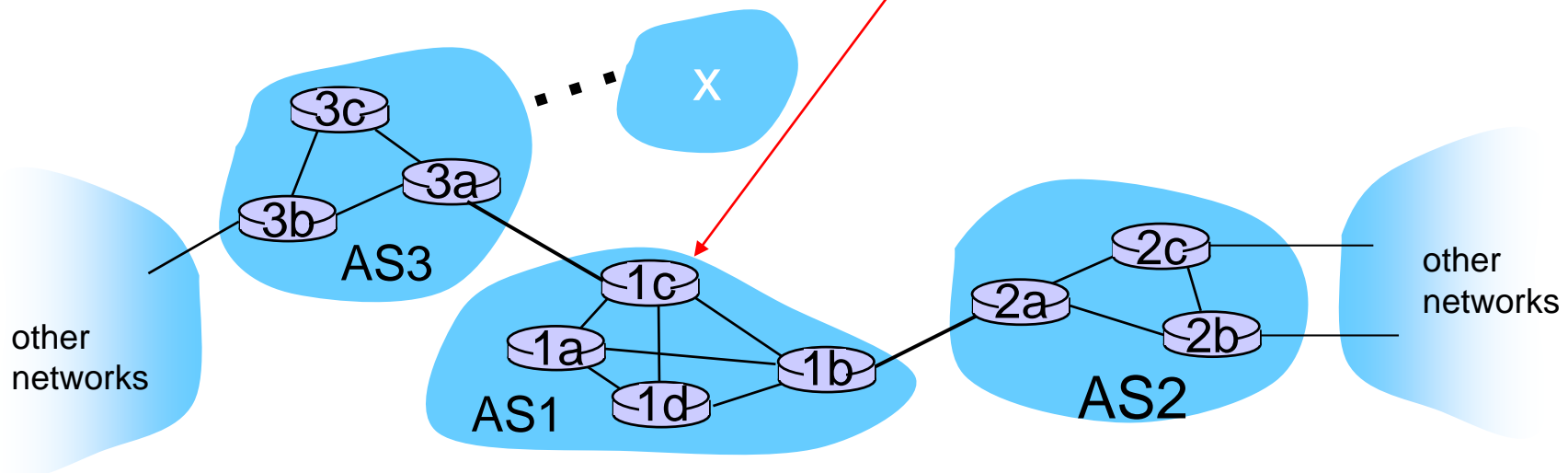
1. learn which destds are reachable through AS2, which through AS3
2. propagate this reachability info to all routers in AS1

*job of inter-AS routing!*



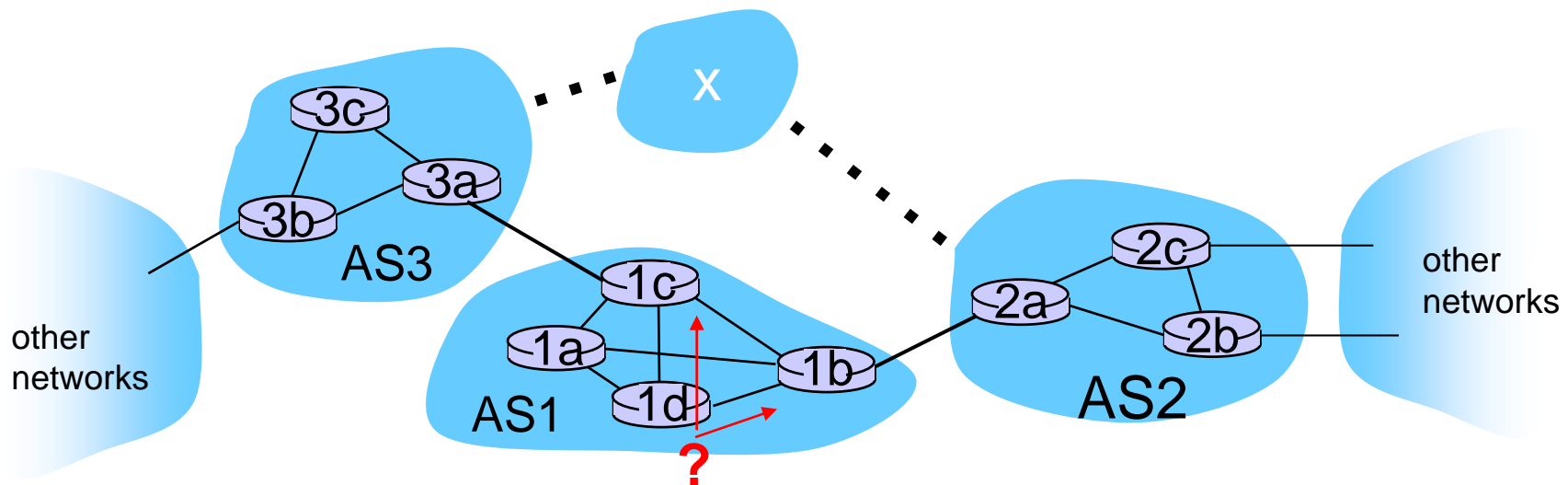
# Example: setting forwarding table in router 1d

- ❖ 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  $l$  is on the least cost path to 1c
  - installs forwarding table entry  $(x, l)$



# 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!



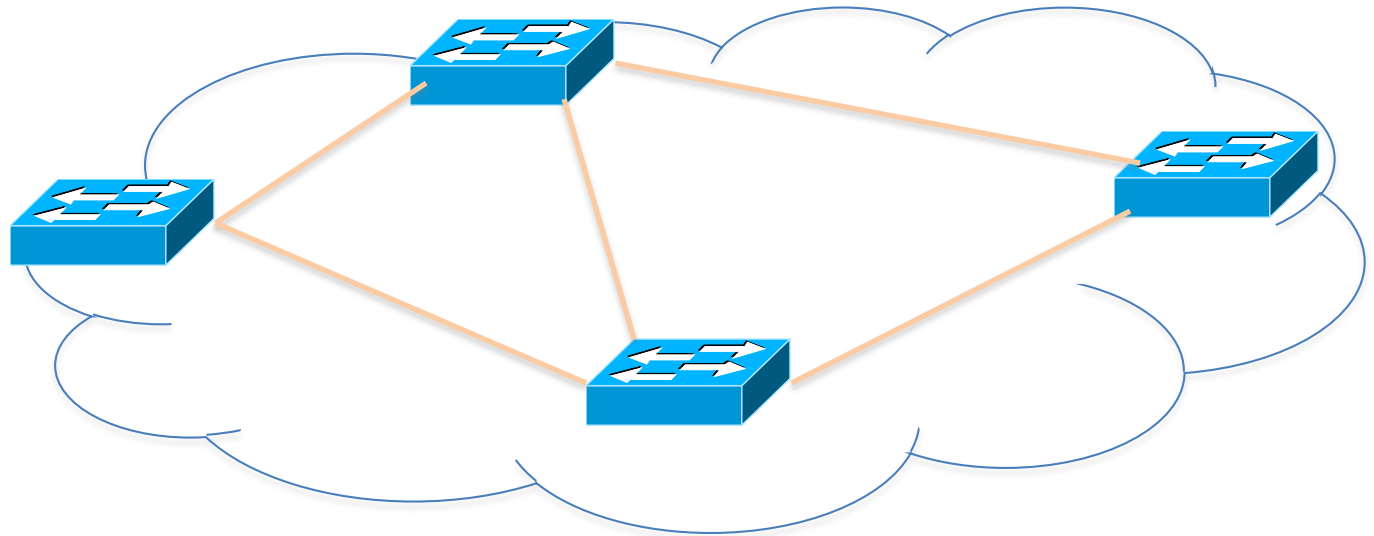


# Example: choosing among multiple ASes

- ❖ now suppose AS1 learns from inter-AS protocol that subnet *x* is reachable from AS3 *and* from AS2.
- ❖ to configure forwarding table, router Id 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.

# Traditional Computer Networks

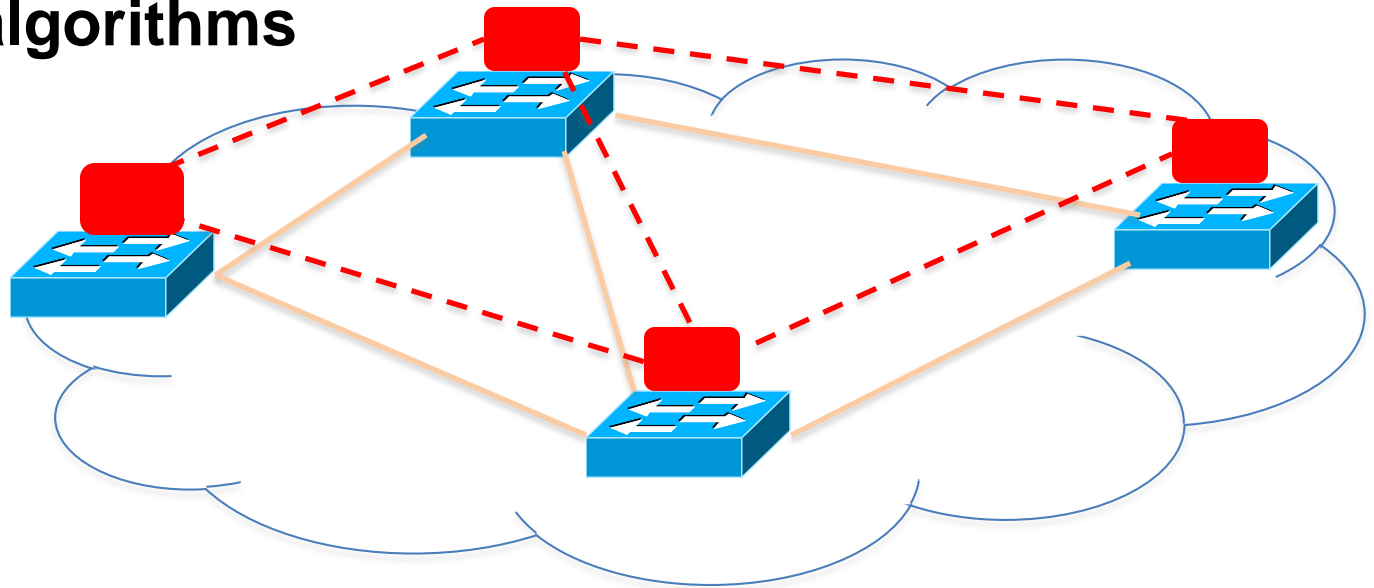
**Data plane:**  
**Packet  
streaming**



**Forward, filter, buffer, mark,  
rate-limit, and measure packets**

# Traditional Computer Networks

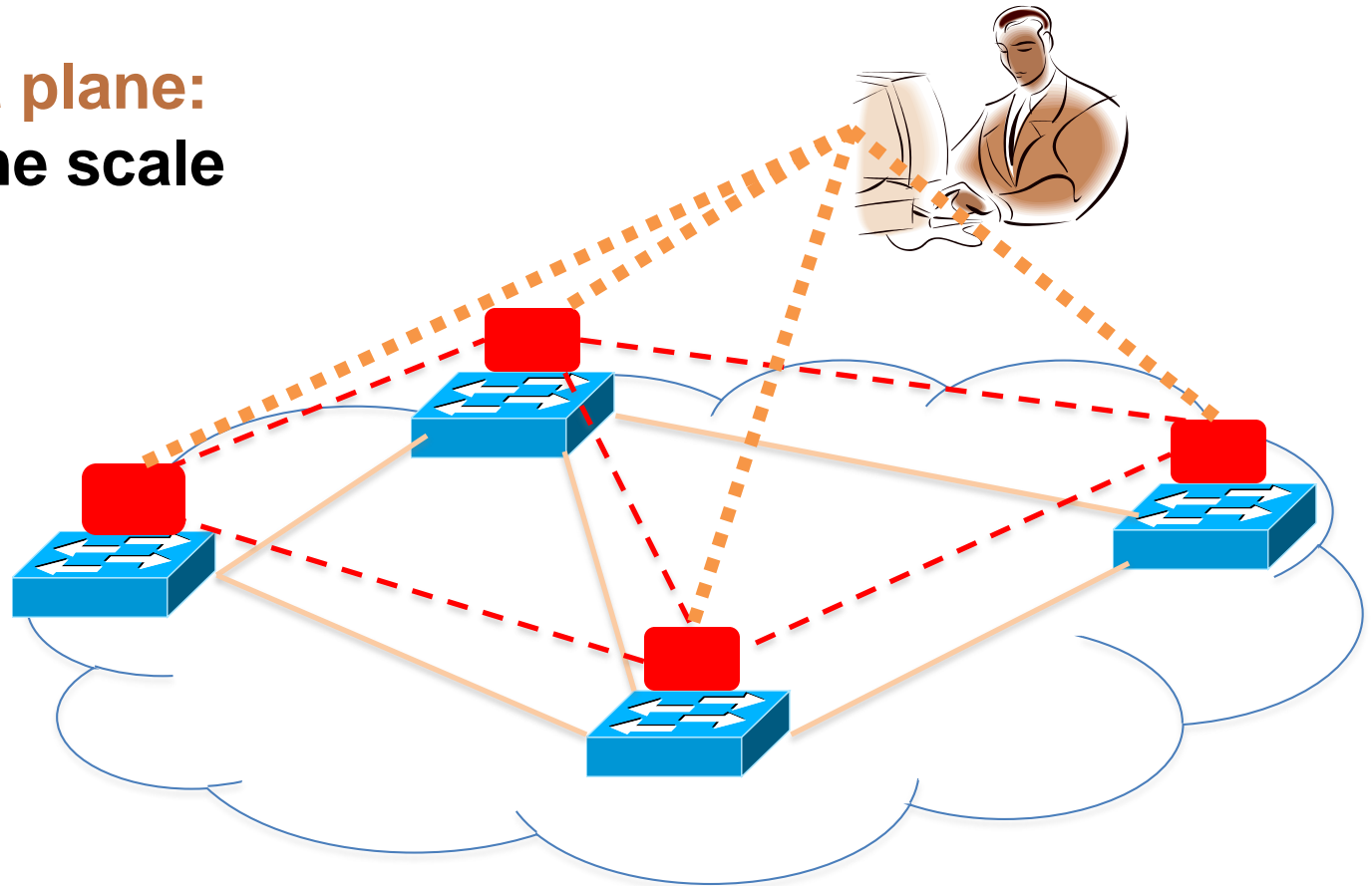
**Control plane:**  
**Distributed algorithms**



**Track topology changes, compute routes, install forwarding rules**

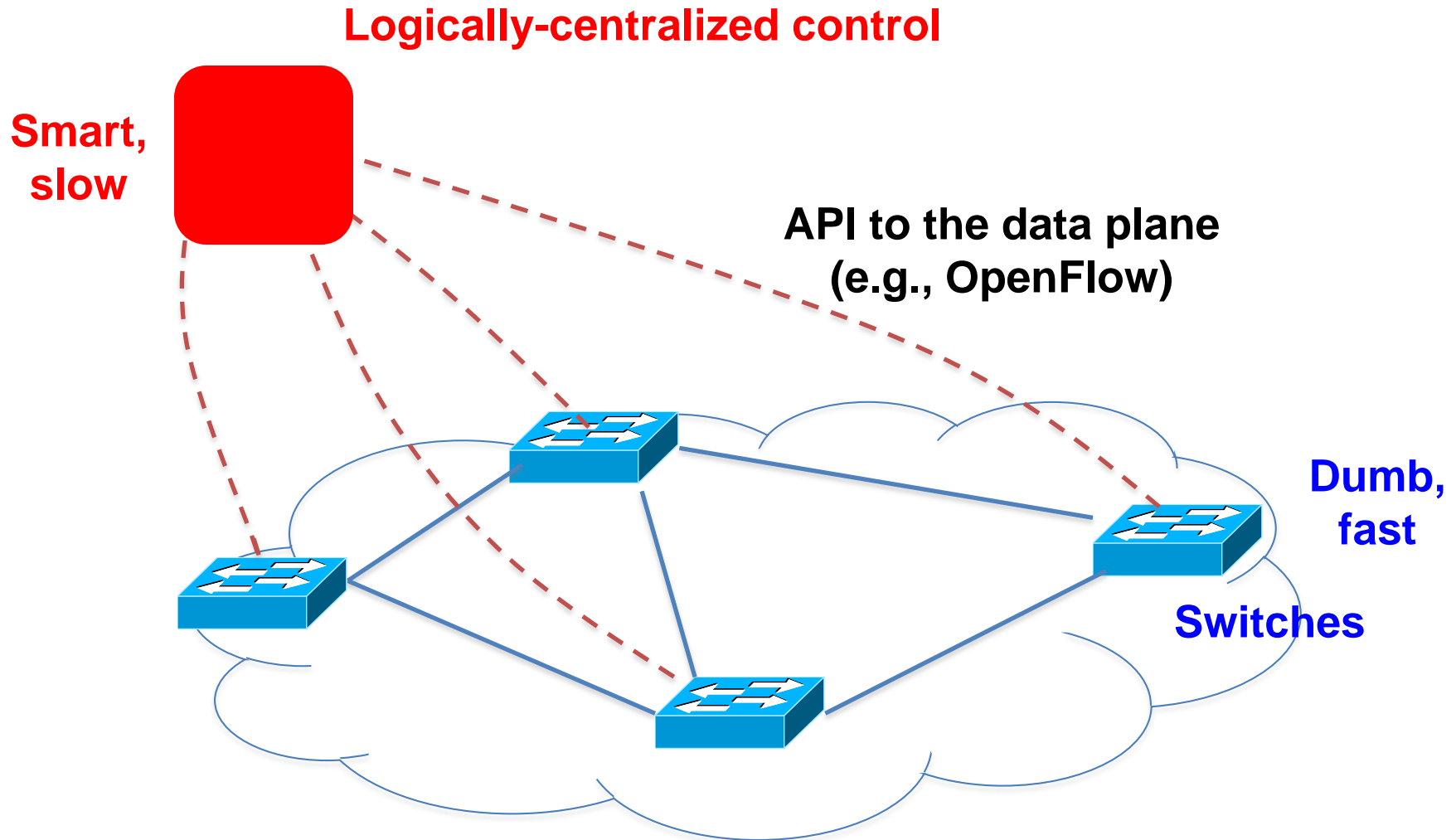
# Traditional Computer Networks

**Management plane:**  
**Human time scale**



**Collect measurements and  
configure the equipment**

# Software Defined Networking (SDN)



# OpenFlow Networks

# Data-Plane: Simple Packet Handling

- Simple packet-handling rules



- Pattern: match packet header bits
- Actions: drop, forward, modify, send to controller
- Priority: disambiguate overlapping patterns
- Counters: #bytes and #packets



1. `src=1.2.*.*`, `dest=3.4.5.*` → drop
2. `src = *.*.*.*`, `dest=3.4.*.*` → forward(2)
3. `src=10.1.2.3`, `dest=*.*.*.*` → send to controller

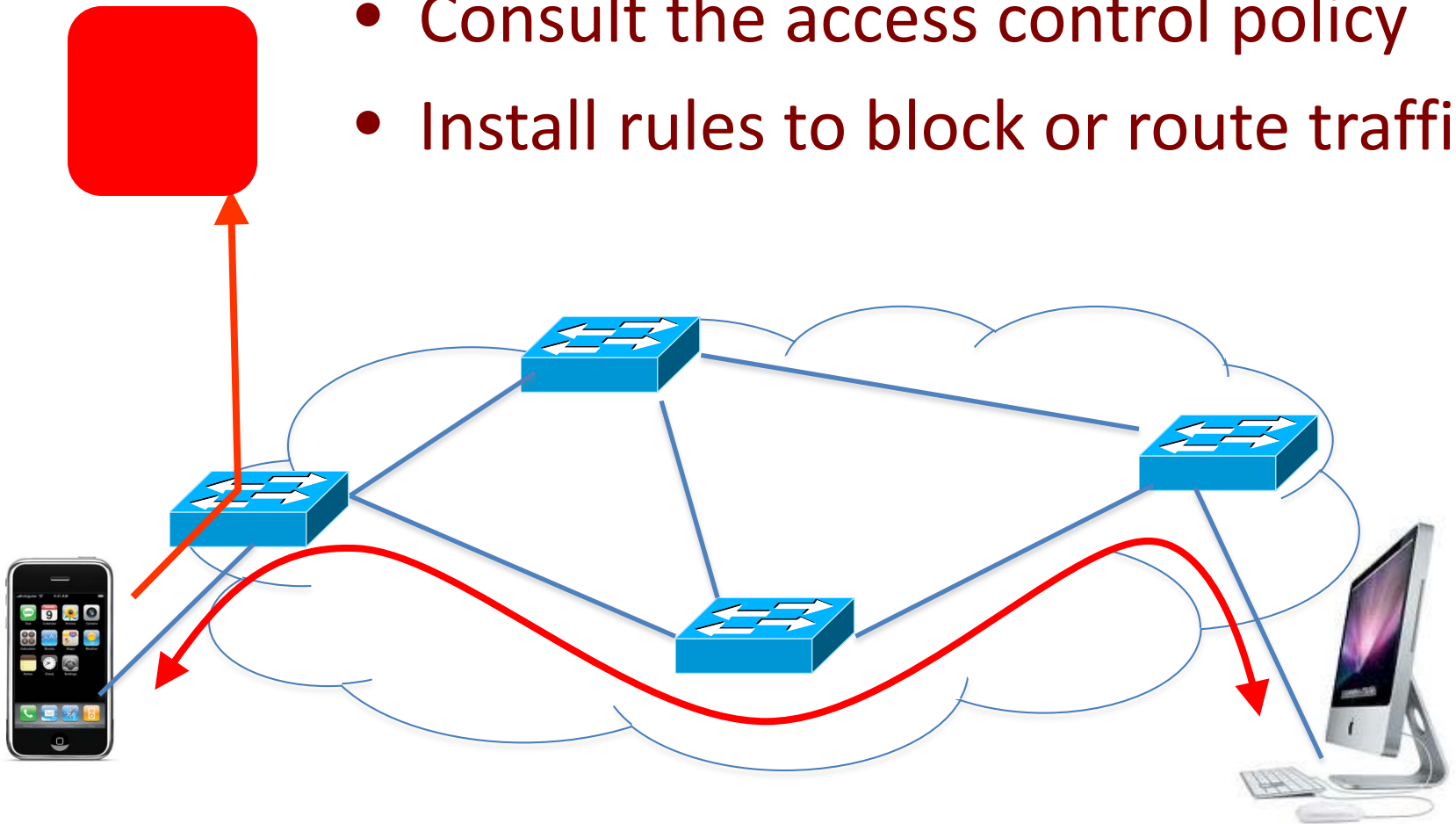
# Unifies Different Kinds of Boxes

- **Router**
  - Match: longest destination IP prefix
  - Action: forward out a link
- **Switch**
  - Match: destination MAC address
  - Action: forward or flood
- **Firewall**
  - Match: IP addresses and TCP/UDP port numbers
  - Action: permit or deny
- **NAT**
  - Match: IP address and port
  - Action: rewrite address and port

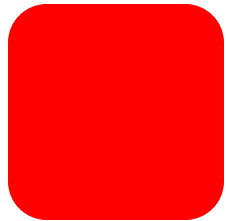


# E.g.: Dynamic Access Control

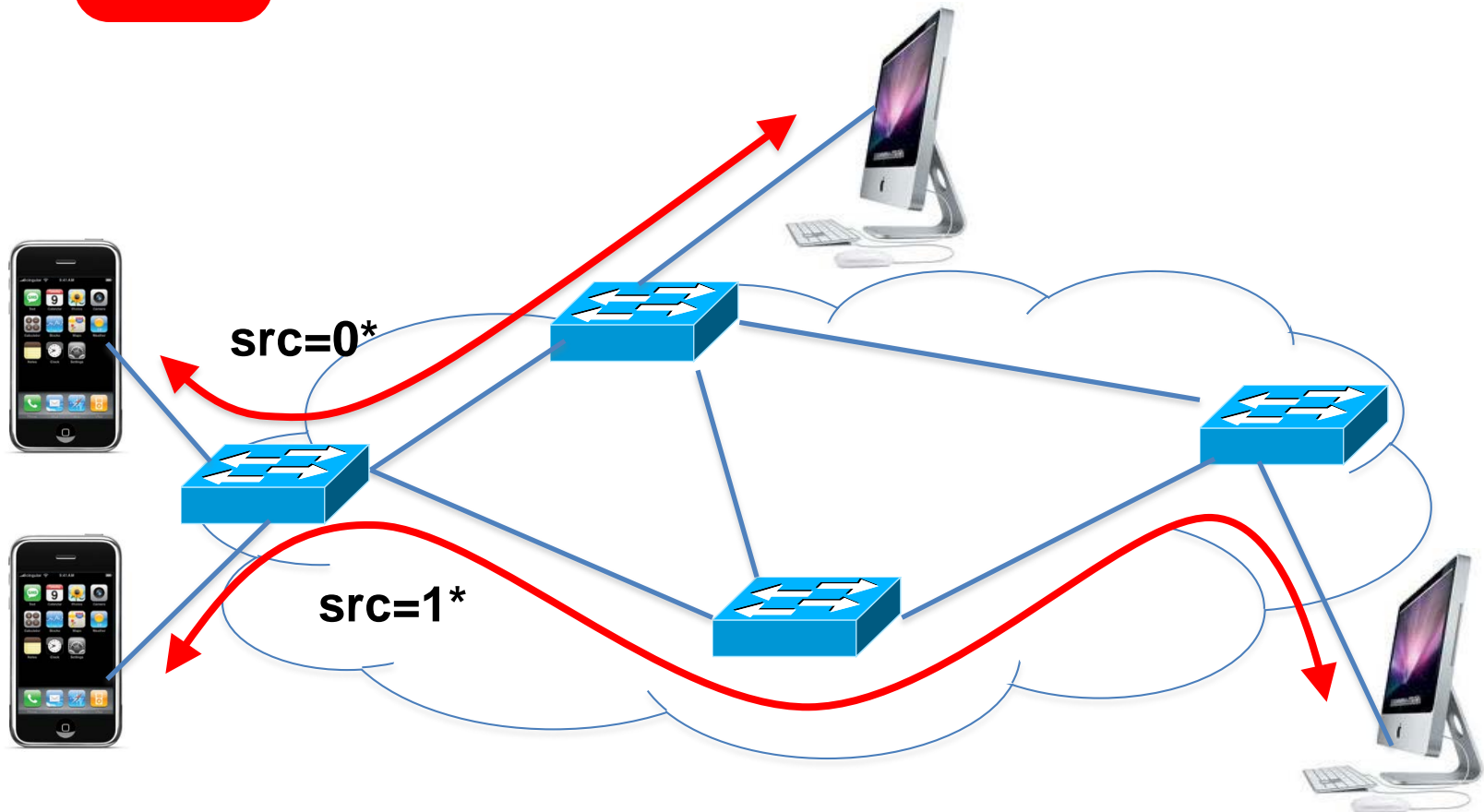
- Inspect first packet of a connection
- Consult the access control policy
- Install rules to block or route traffic



# E.g.: Server Load Balancing



- Pre-install load-balancing policy
- Split traffic based on source IP



# 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

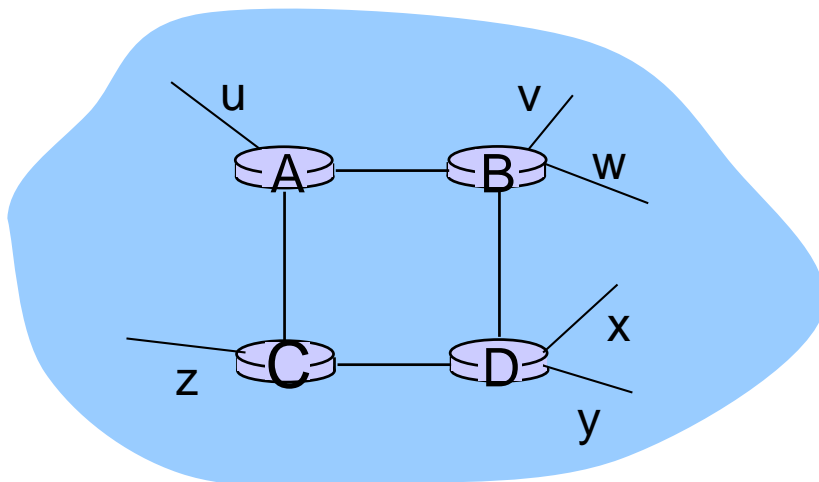
4.7 broadcast and multicast routing

# 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)

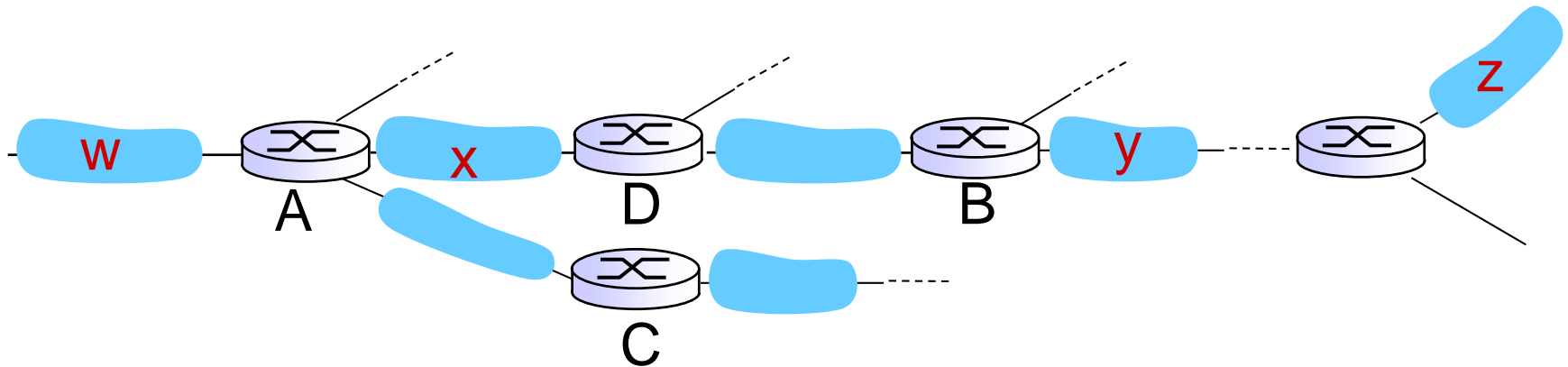
- ❖ included in BSD-UNIX distribution in 1982
- ❖ distance vector algorithm
  - distance metric: # hops (max = 15 hops), each link has cost 1
  - DVs exchanged with neighbors every 30 sec in response message (aka **advertisement**)
  - each advertisement: list of up to 25 destination **subnets** (in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



from router A to destination **subnets**:

<u>subnet</u>	<u>hops</u>
u	1
v	2
w	2
x	3
y	3
z	2

# RIP: example



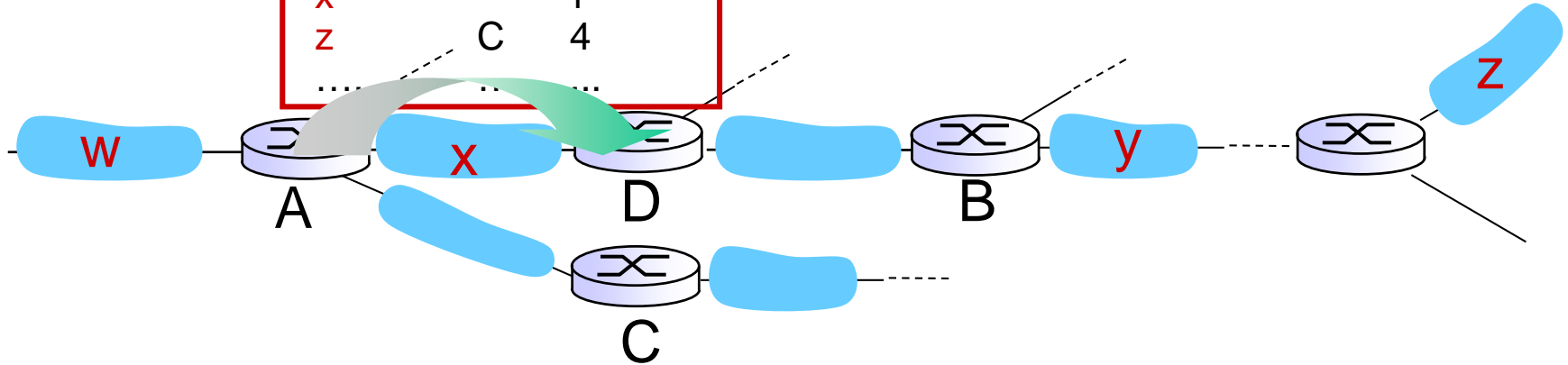
routing table in router D

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	<del>B</del> → A	<del>7</del> → 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



# RIP table processing

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

