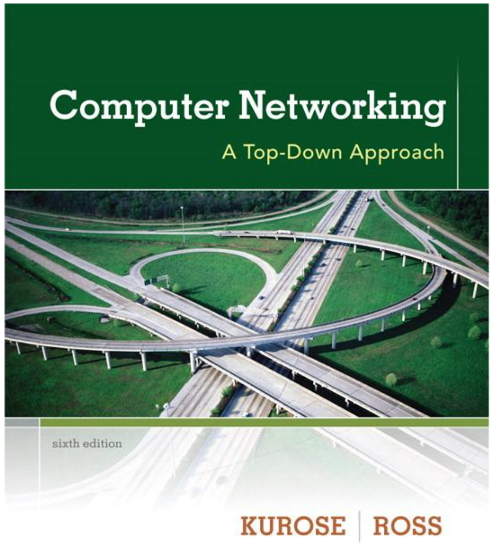
# Chapter 5 Network Layer (Routing)



*Computer Networking: A Top Down Approach*
6th edition
Jim Kurose, Keith Ross
Addison-Wesley
March 2012

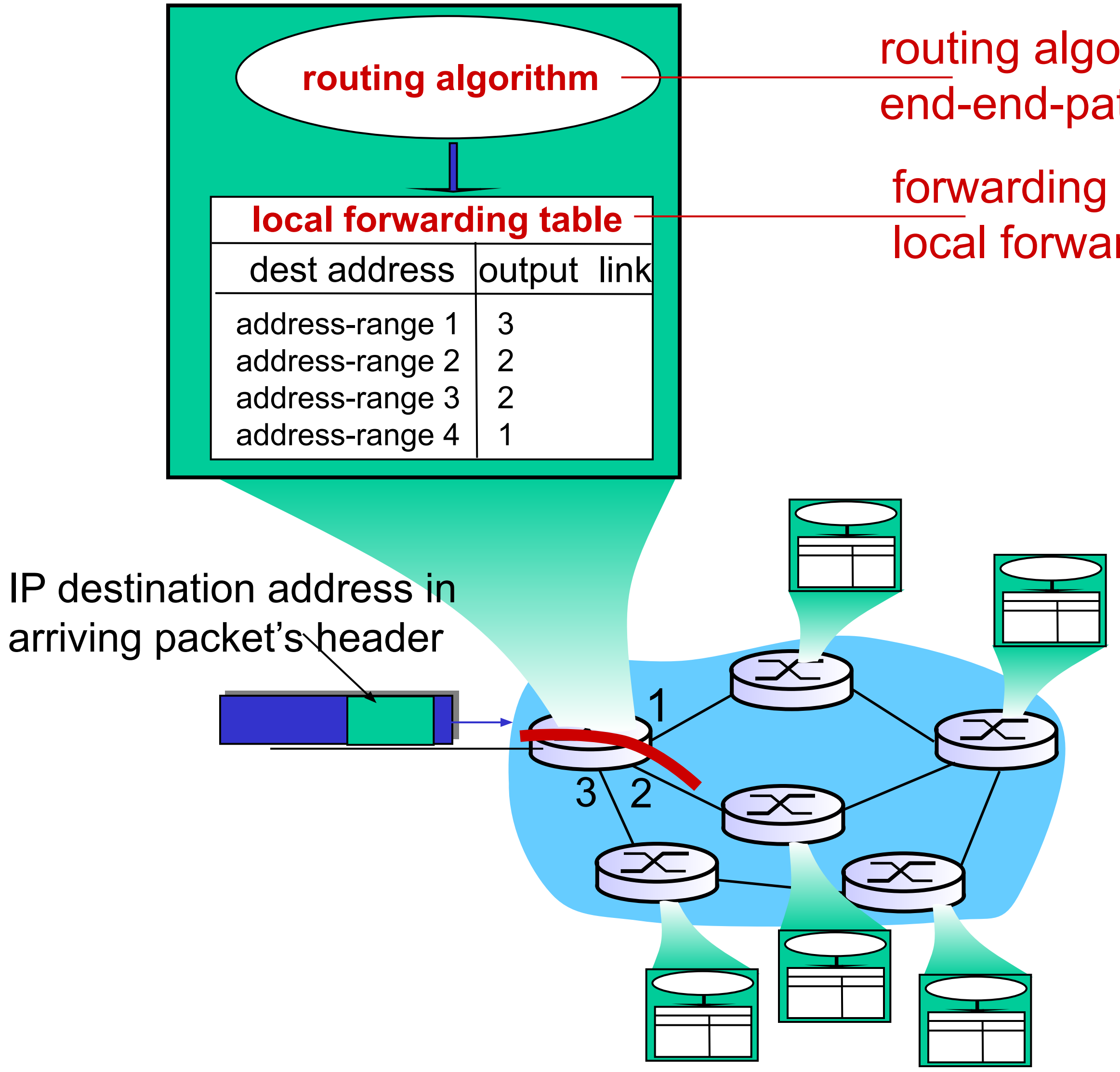
# Chapter 5: outline

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

5.6 routing in the Internet
- RIP
- OSPF
- BGP

5.7 broadcast and multicast routing

# Interplay between routing, forwarding

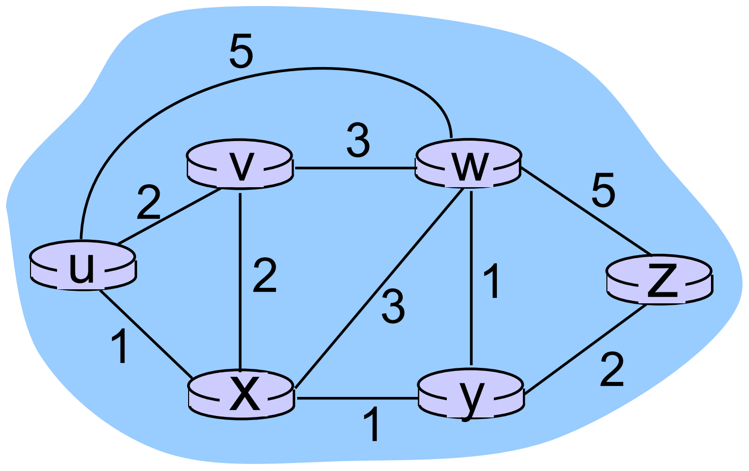


| local forwarding table | |
|---|---|
| dest address | output link |
| address-range 1 | 3 |
| address-range 2 | 2 |
| address-range 3 | 2 |
| address-range 4 | 1 |

routing algorithm determines end-end-path through network

forwarding table determines local forwarding at this router

# Graph abstraction

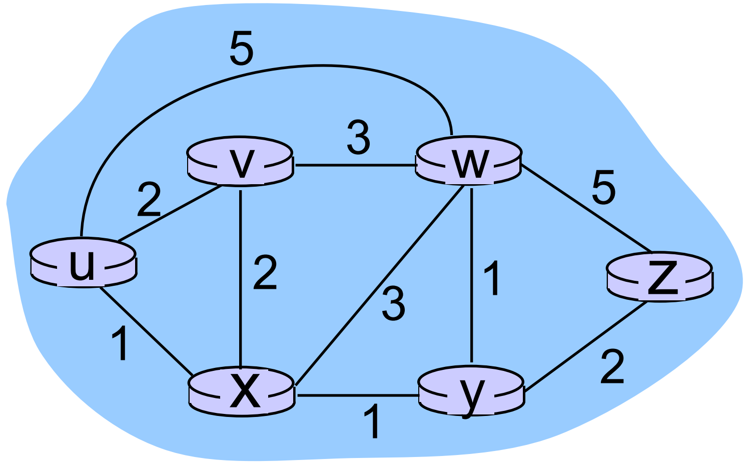


graph: G = (N,E)

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 directly 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

# 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

```
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

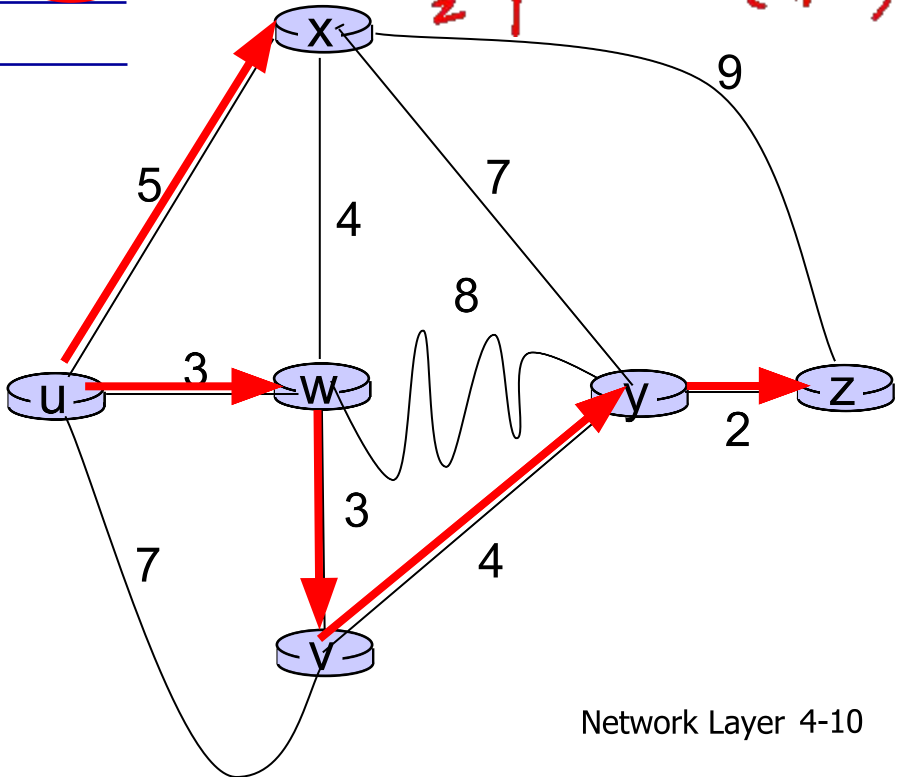
| Step | N' | D(v) p(v) | D(w) p(w) | D(x) p(x) | D(y) p(y) | D(z) p(z) |
|---|---|---|---|---|---|---|
| 0 | u | 7,u | 3,u | 5,u | ∞ | ∞ |
| 1 | uw | 6,w | | 5,u | 11,w | ∞ |
| 2 | uwx | 6,w | | | 11,w | 14,x |
| 3 | uwxv | | | | 10,y | 14,x |
| 4 | uwxvy | | | | | 12,y |
| 5 | uwxvyz | | | | | |

Forwarding table for 'u'

| Dst. | Link |
|---|---|
| w | (u,w) |
| x | (u,x) |
| v | (u,w) |
| y | (u,w) |
| z | (u,w) |

*notes:*

- construct shortest path tree by tracing predecessor nodes
- ties can exist (can be broken arbitrarily)
- How will the forwarding table look like?



# Chapter 4: outline

## Distance vector algorithm

*Bellman-Ford equation (dynamic programming)*

let

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

then

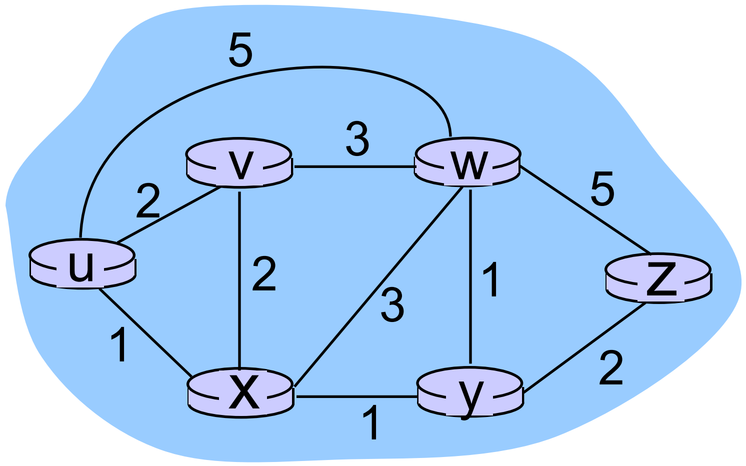
$$d_x(y) = \min_v \{c(x,v) + d_v(y)\}$$

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:

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

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:

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

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

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

**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 |

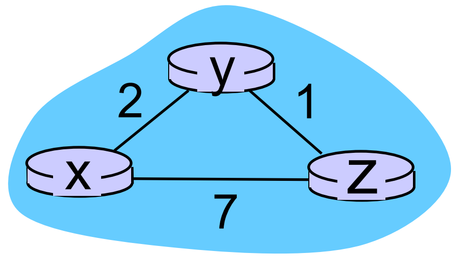
**node y table**

| from \ *cost to* | x | y | z |
|---|---|---|---|
| x | ∞ | ∞ | ∞ |
| y | 2 | 0 | 1 |
| z | ∞ | ∞ | ∞ |

**node z table**

| *from* \ *cost to* | 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**

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 |

cost to

| from | x | y | z |
|---|---|---|---|
| x | 0 | 2 | 3 |
| y | 2 | 0 | 1 |
| z | 3 | 1 | 0 |

**node y table**

cost to

| from | x | y | z |
|---|---|---|---|
| x | ∞ | ∞ | ∞ |
| y | 2 | 0 | 1 |
| z | ∞ | ∞ | ∞ |

cost to

| from | x | y | z |
|---|---|---|---|
| x | 0 | 2 | 7 |
| y | 2 | 0 | 1 |
| z | 7 | 1 | 0 |

cost to

| from | x | y | z |
|---|---|---|---|
| x | 0 | 2 | 3 |
| y | 2 | 0 | 1 |
| z | 3 | 1 | 0 |

**node z table**

cost to

| from | x | y | z |
|---|---|---|---|
| x | ∞ | ∞ | ∞ |
| y | ∞ | ∞ | ∞ |
| z | 7 | 1 | 0 |

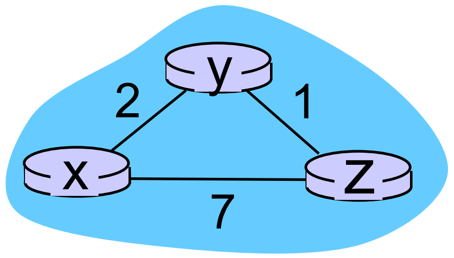
cost to

| from | x | y | z |
|---|---|---|---|
| x | 0 | 2 | 7 |
| y | 2 | 0 | 1 |
| z | 3 | 1 | 0 |

cost to

| from | x | y | z |
|---|---|---|---|
| x | 0 | 2 | 3 |
| y | 2 | 0 | 1 |
| z | 3 | 1 | 0 |

time



# Distance vector algorithm

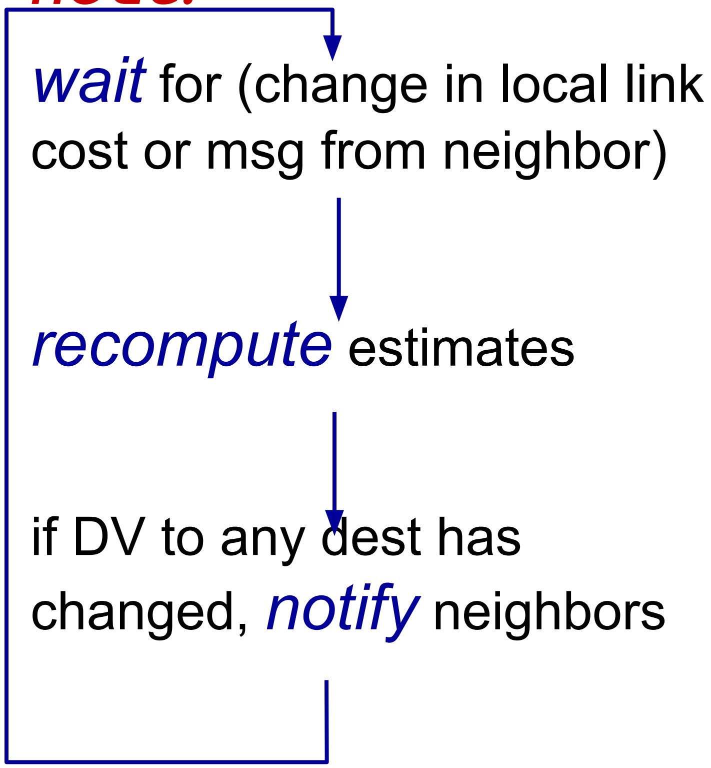
*iterative, asynchronous:*
each local iteration caused by:

- ❖ 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)

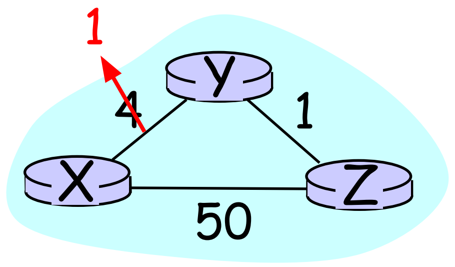
*recompute* estimates

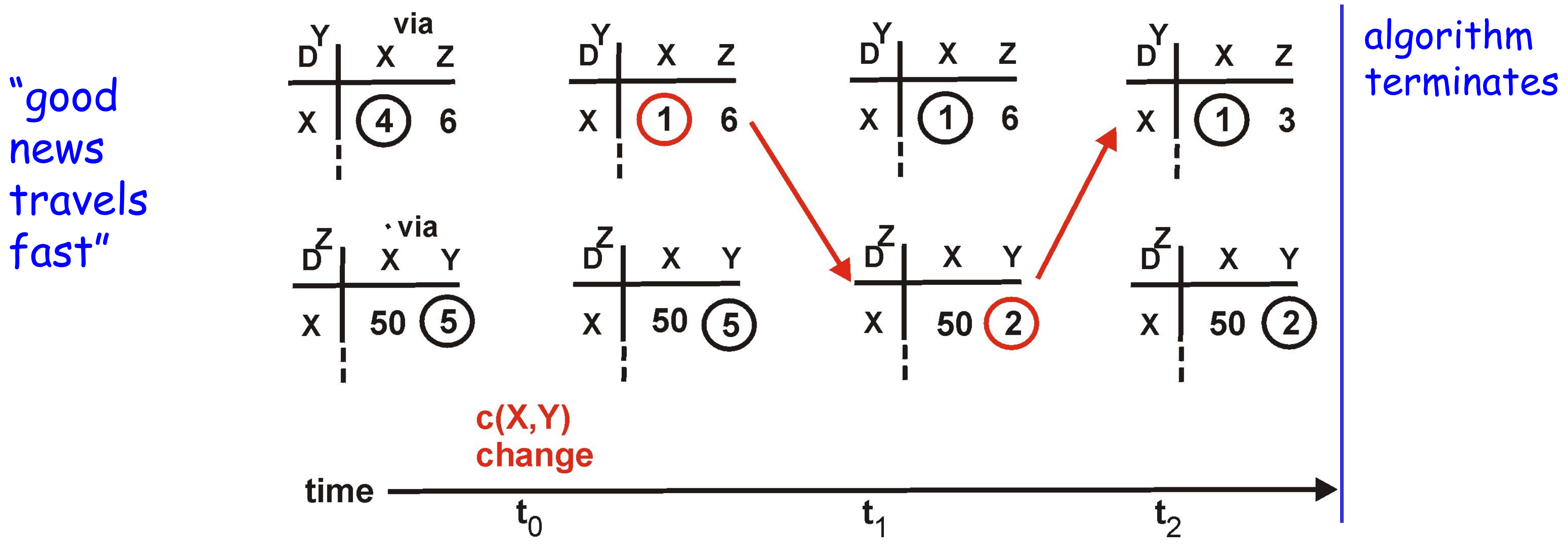
if DV to any dest has changed, *notify* neighbors

# Distance Vector Algorithm: Link Cost Changes

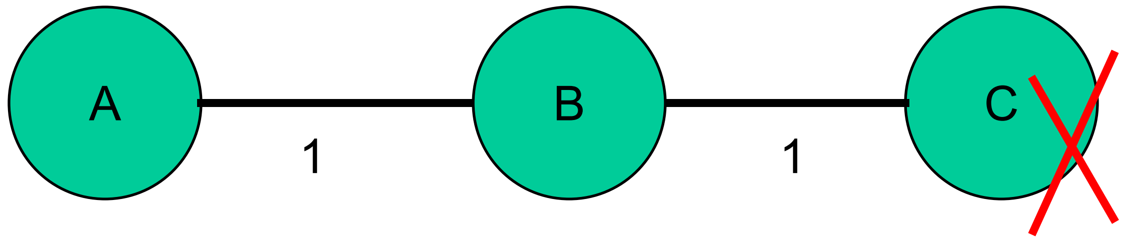
## Link cost changes:

- node detects local link cost change
- updates distance table (line 15)
- if cost change in least cost path, notify neighbors (lines 23,24)





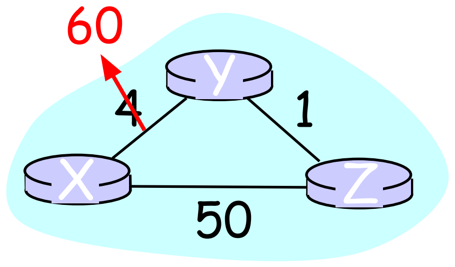
# Count to infinity: Another Example

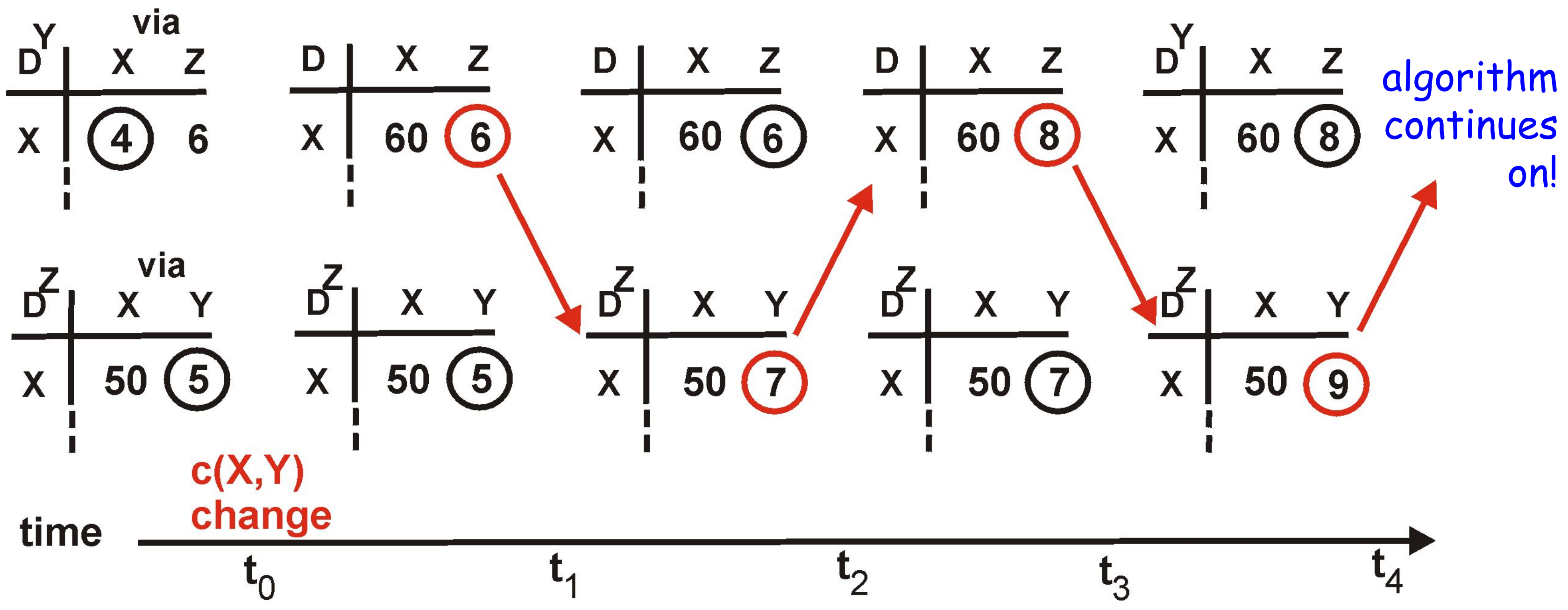


## Distance Vector: Link Cost Changes

Link cost changes:

- good news travels fast
- bad news travels slow - "count to infinity" problem!





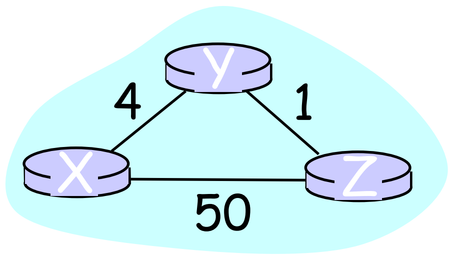
# When list cost is reduced

**node y table**

| from \ *cost to* | x | y | z |
|---|---|---|---|
| y | 4 | 0 | 1 |
| z | 5 | 1 | 0 |

**node z table**

| from \ *cost to* | x | y | z |
|---|---|---|---|
| y | 4 | 0 | 1 |
| z | 5 | 1 | 0 |



**node y table**

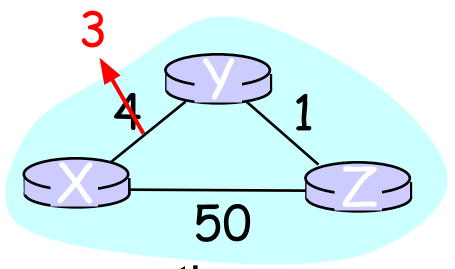
| from \ *cost to* | x | y | z |
|---|---|---|---|
| y | 3 | 0 | 1 |
| z | 5 | 1 | 0 |

| from \ *cost to* | x | y | z |
|---|---|---|---|
| y | 3 | 0 | 1 |
| z | 41 | 0 | |

**node z table**

| from \ *cost to* | x | y | z |
|---|---|---|---|
| y | 3 | 0 | 1 |
| z | 41 | 0 | |

**Algorithm terminates here as no change now**



time

## When list cost is increased

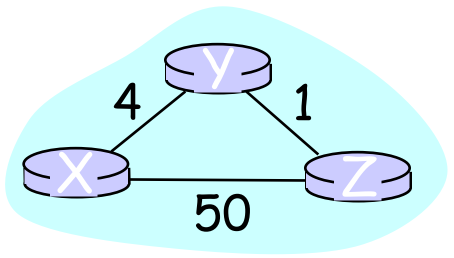
**And so on… Continues until the path cost becomes >50**

**node y table**

| from \ cost to | x | y | z |
|---|---|---|---|
| y | 4 | 0 | 1 |
| z | 5 | 1 | 0 |

**node z table**

| from \ cost to | x | y | z |
|---|---|---|---|
| y | 4 | 0 | 1 |
| z | 5 | 1 | 0 |



**node y table**

| from \ cost to | x | y | z |
|---|---|---|---|
| y | 6 | 0 | 1 |
| z | 5 | 1 | 0 |

| from \ cost to | x | y | z |
|---|---|---|---|
| y | 8 | 0 | 1 |
| z | 7 | 1 | 0 |

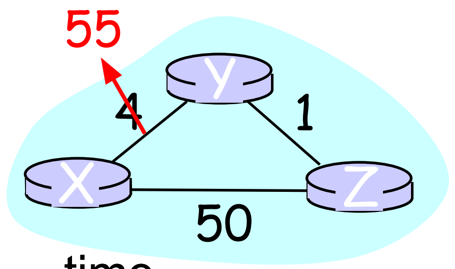
| from \ cost to | x | y | z |
|---|---|---|---|
| y | 10 | 0 | 1 |
| z | 9 | 1 | 0 |

**node z table**

| from \ cost to | x | y | z |
|---|---|---|---|
| y | 6 | 0 | 1 |
| z | 7 | 1 | 0 |

| from \ cost to | x | y | z |
|---|---|---|---|
| y | 8 | 0 | 1 |
| z | 9 | 1 | 0 |

| from \ cost to | x | y | z |
|---|---|---|---|
| y | 10 | 0 | 1 |
| z | 11 | 1 | 0 |

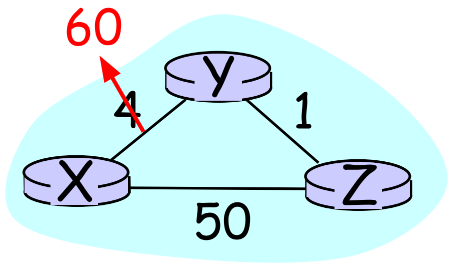


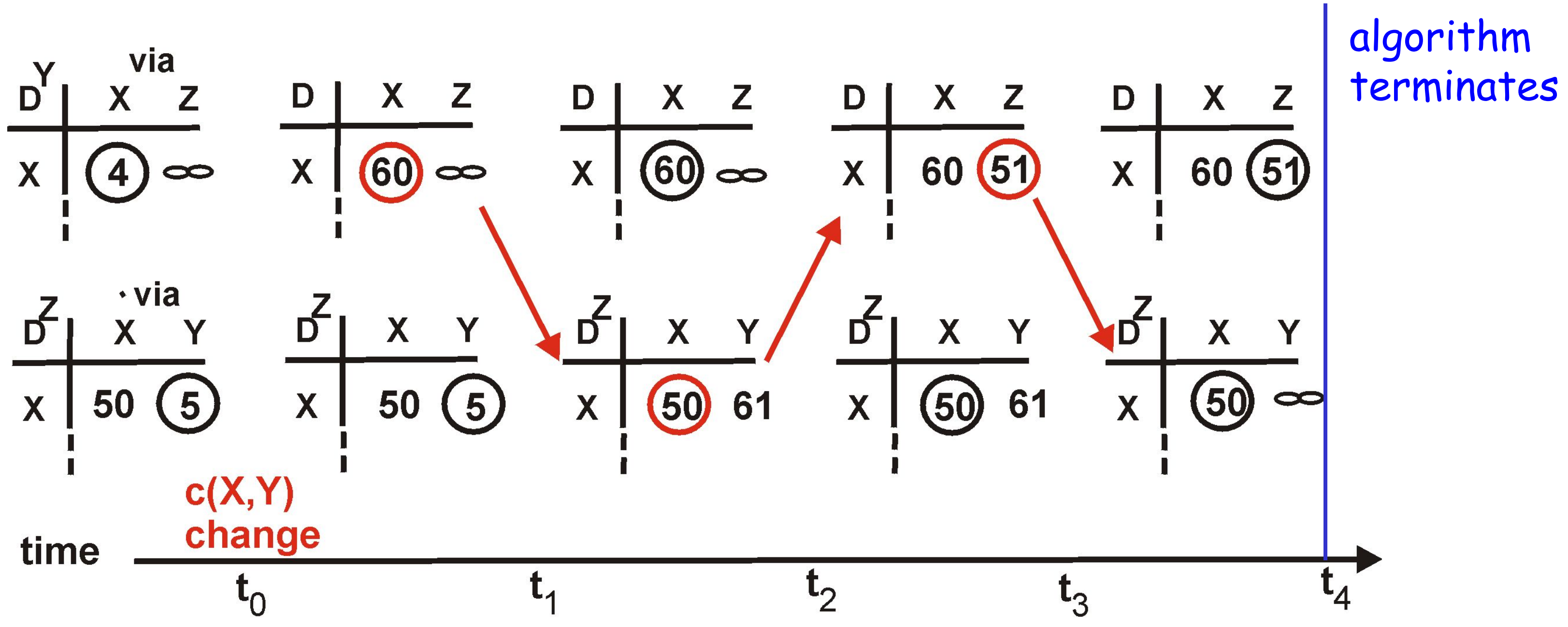
time

# 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?





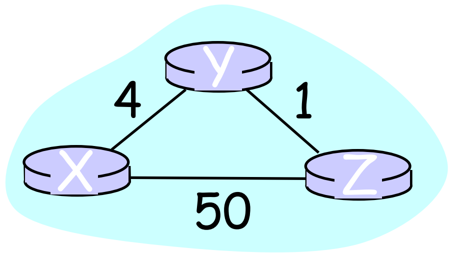
# Poisoned reverse example

**node y table**

| from \ cost to | x | y | z |
|---|---|---|---|
| y | 4 | 0 | 1 |
| z | ∞ | 1 | 0 |

**node z table**

| from \ cost to | x | y | z |
|---|---|---|---|
| y | 4 | 0 | 1 |
| z | 5 | 1 | 0 |



**node y table**

| from \ cost to | x | y | z |
|---|---|---|---|
| y | 55 | 0 | 1 |
| z | ∞ | 1 | 0 |

| from \ cost to | x | y | z |
|---|---|---|---|
| y | 51 | 0 | 1 |
| z | 7 | 1 | 0 |

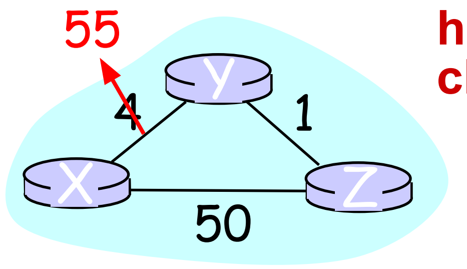
∞ 0 1

**node z table**

| from \ cost to | x | y | z |
|---|---|---|---|
| y | 55 | 0 | 1 |
| z | 50 | 1 | 0 |

| from \ cost to | x | y | z |
|---|---|---|---|
| y | ∞ | 0 | 1 |
| z | 50 | 1 | 0 |

**Algorithm terminates here as no change now**



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
- ***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

## 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

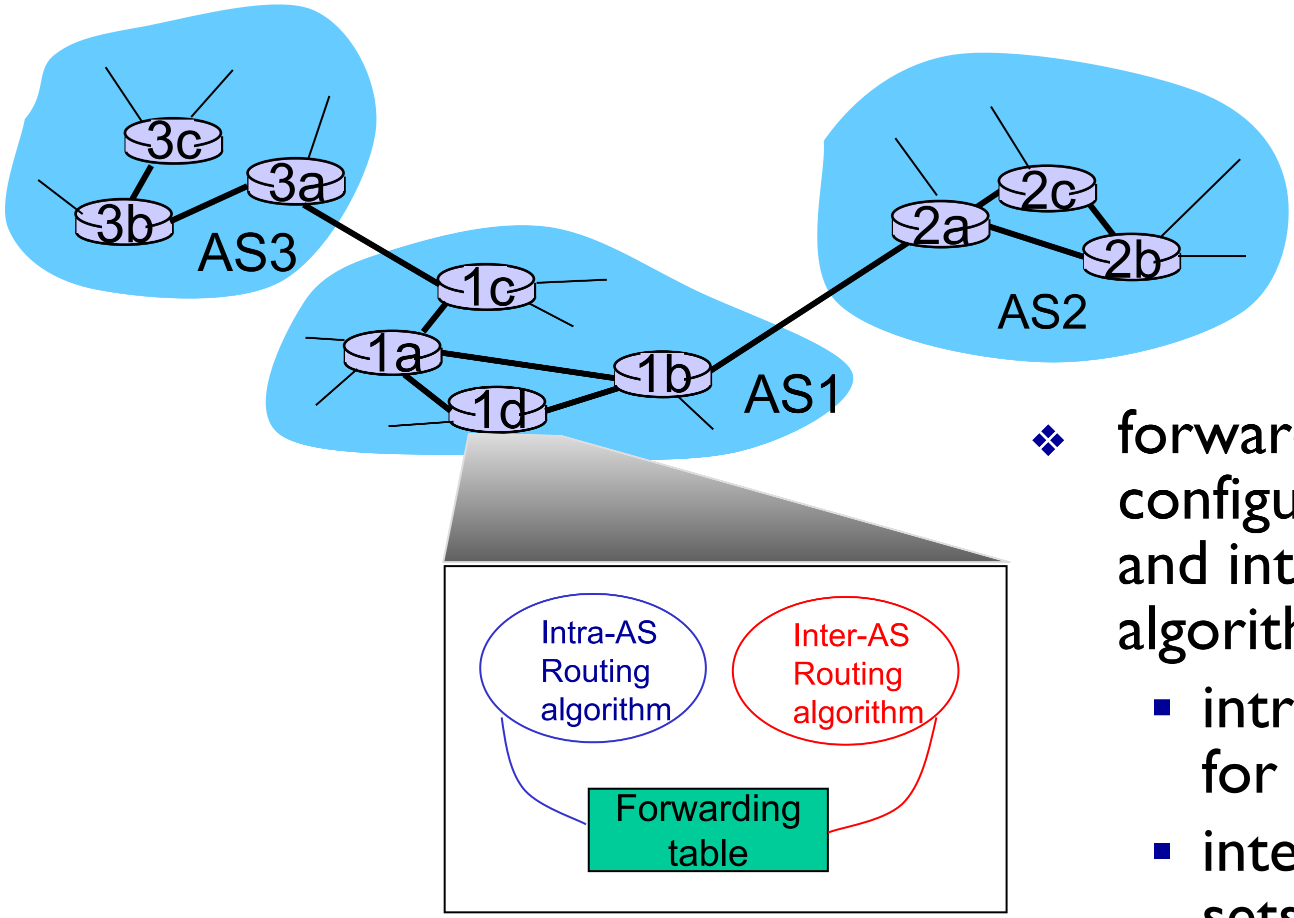
## Hierarchical routing

- 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

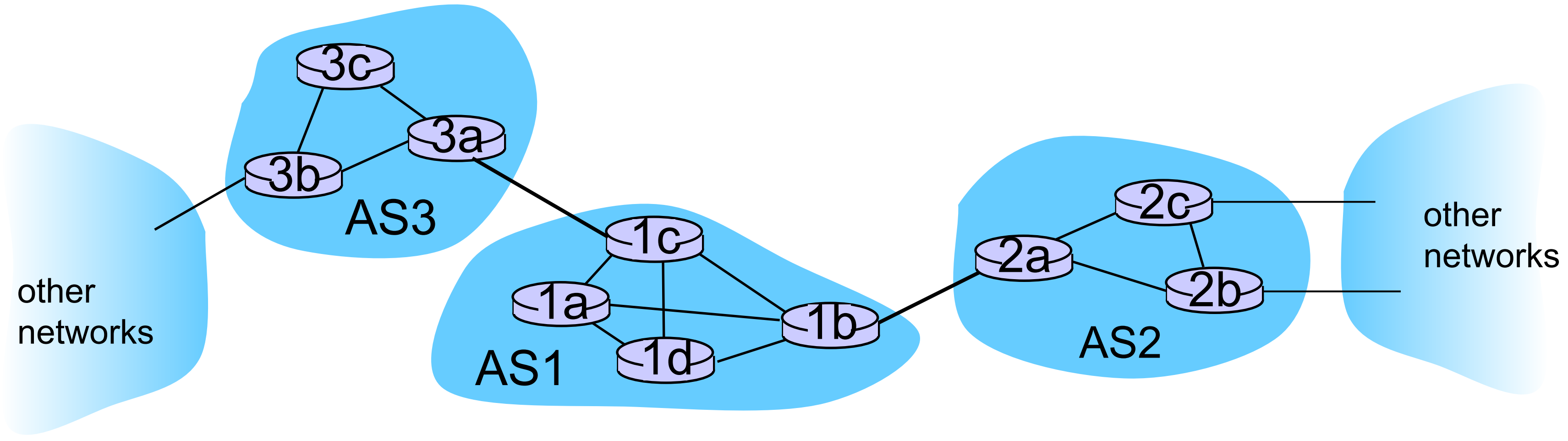
# Inter-AS tasks

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

*AS1 must:*

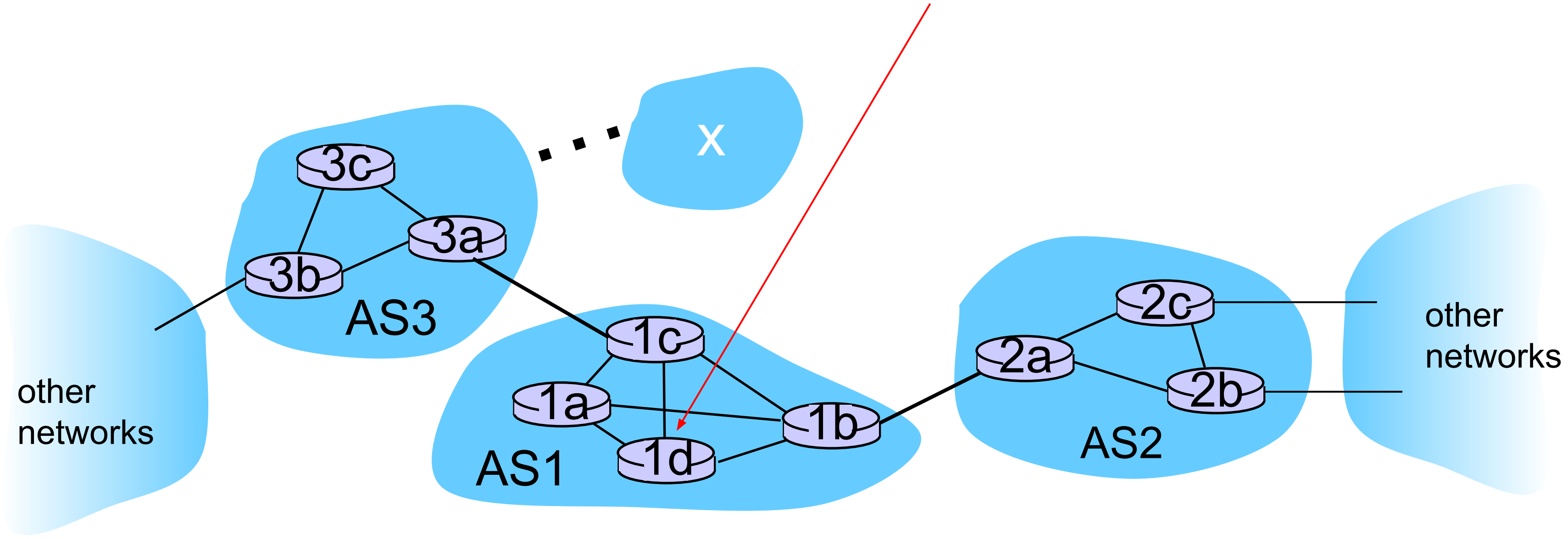
1. learn which dests are reachable through AS2, which through AS3
2. propagate this reachability info to all routers in AS1

*job of inter-AS routing!*



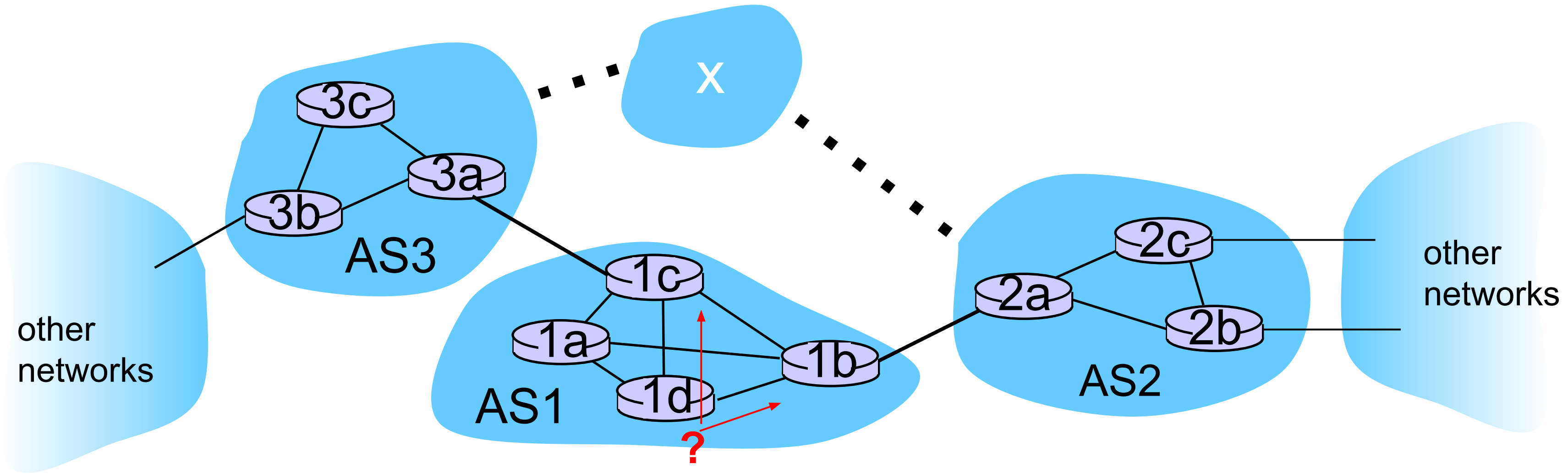
## Example: setting forwarding table in router 1d

- suppose AS1 learns (via inter-AS protocol) that subnet *x* reachable via AS3 (gateway 1c)
  - 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)*



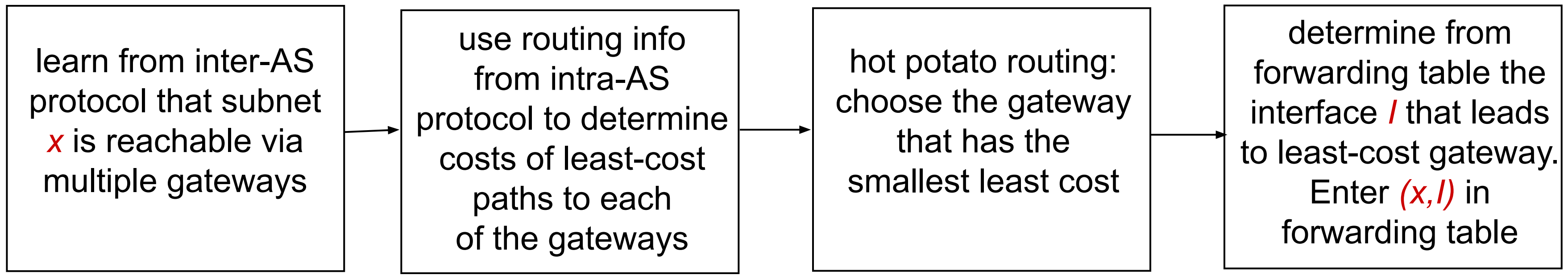
# Example: choosing among multiple ASes

- 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



# Example: choosing among multiple ASes

- *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

## 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)

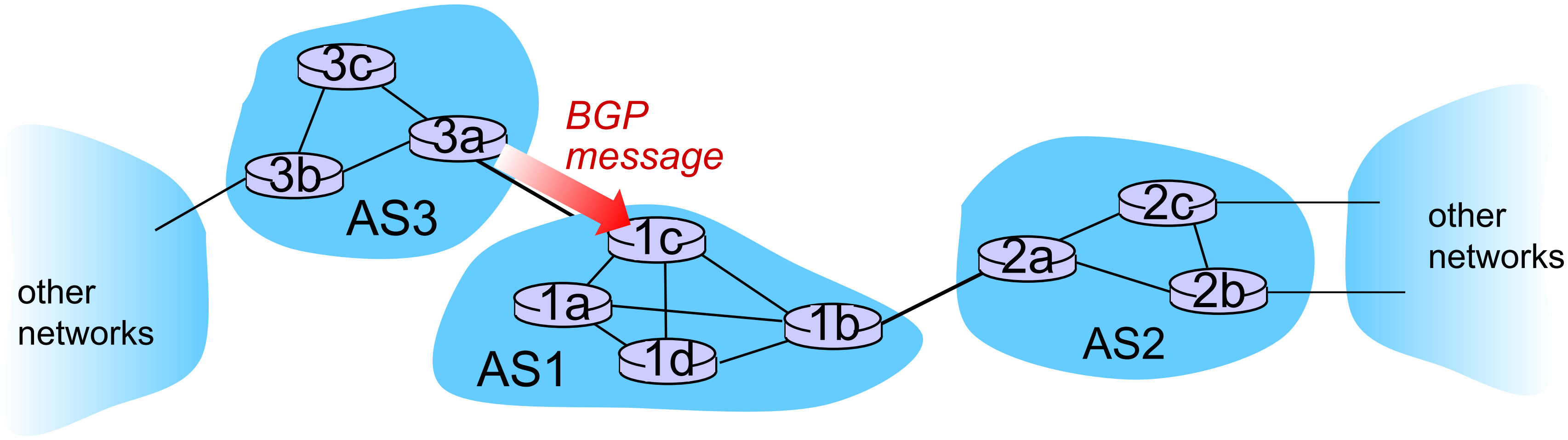
## Internet inter-AS routing: BGP

- 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

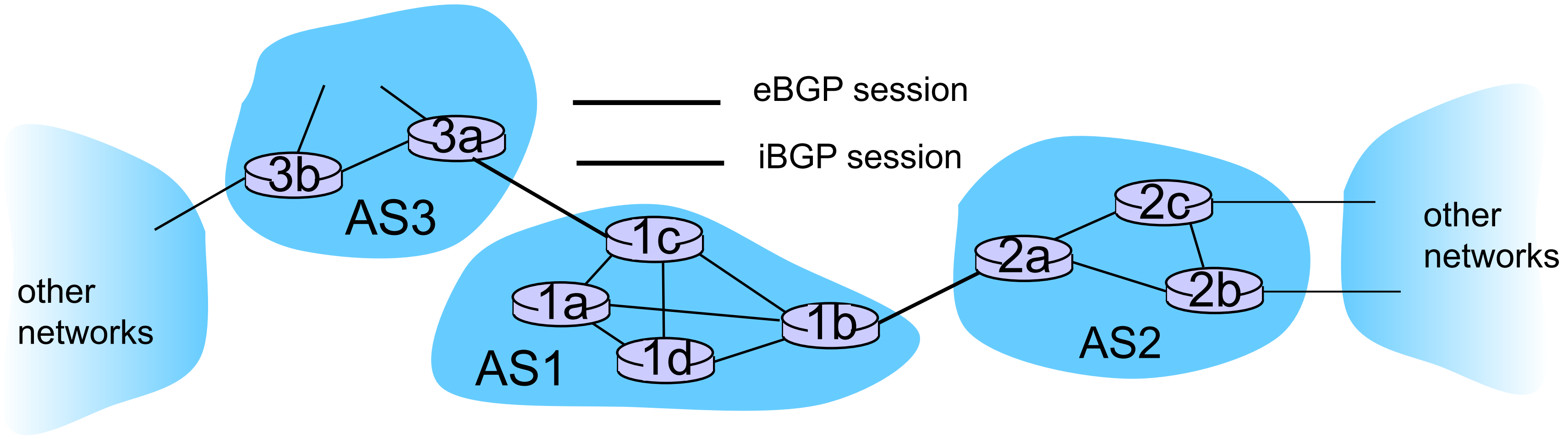
- BGP session: two BGP routers ("peers") exchange BGP messages:
  - advertising *paths* to different destination network prefixes ("path vector" protocol)
  - exchanged over semi-permanent TCP connections

- 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.



## 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)

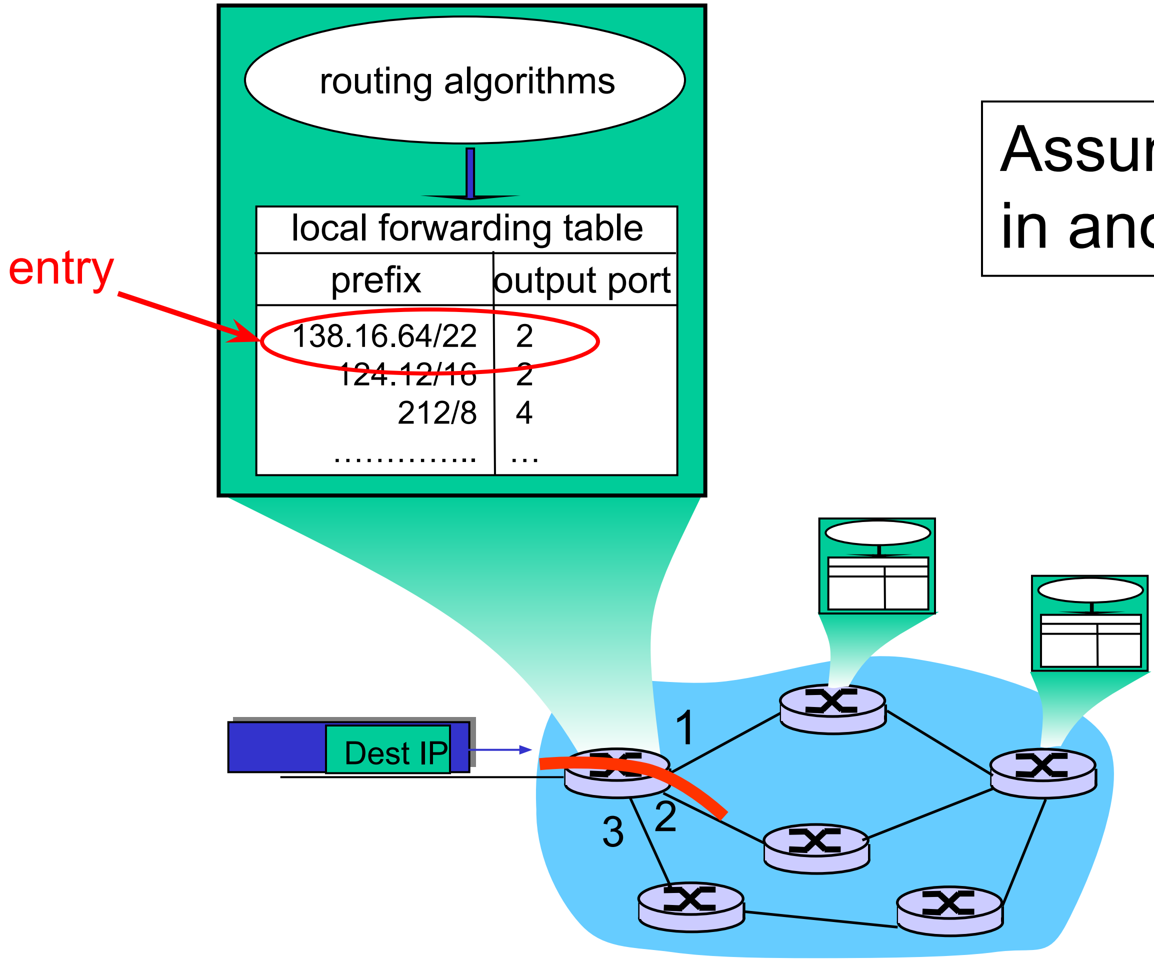
## BGP route selection

- router may learn about more than one route to destination AS, selects route based on:
  1. local preference value attribute: policy decision
  2. shortest AS-PATH
  3. closest NEXT-HOP router: hot potato routing
  4. additional criteria

## Putting it Altogether: *How Does an Entry Get Into a Router's Forwarding Table?*

- Answer is complicated!
- Ties together hierarchical routing (Section 4.5.3) with BGP (4.6.3) and OSPF (4.6.2).
- Provides nice overview of BGP!

# How does entry get in forwarding table?



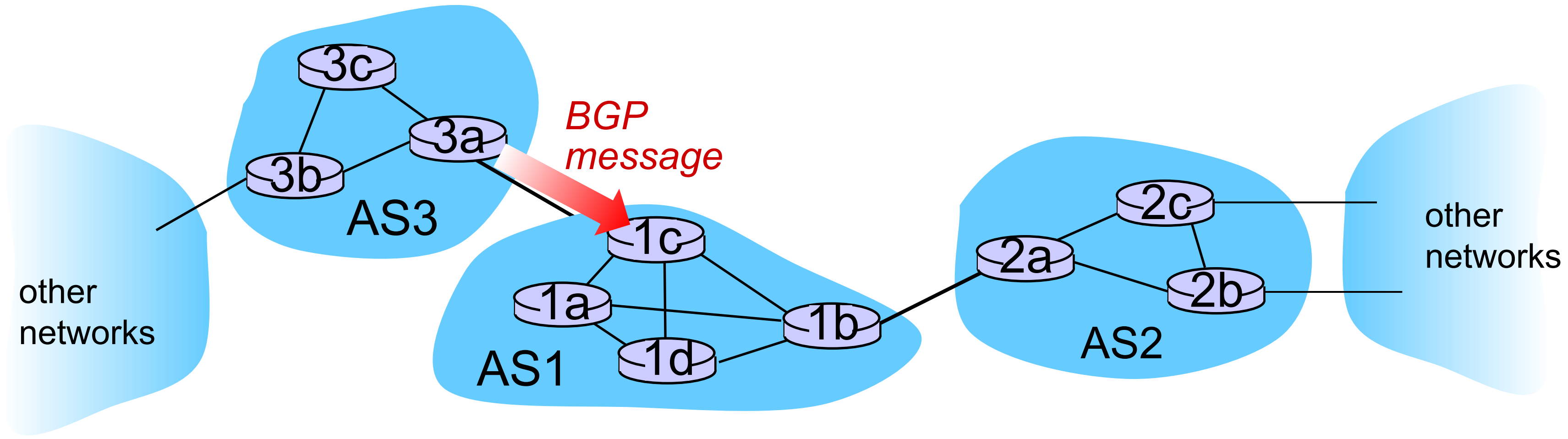
Assume prefix is in another AS.

# How does entry get in forwarding table?

High-level overview

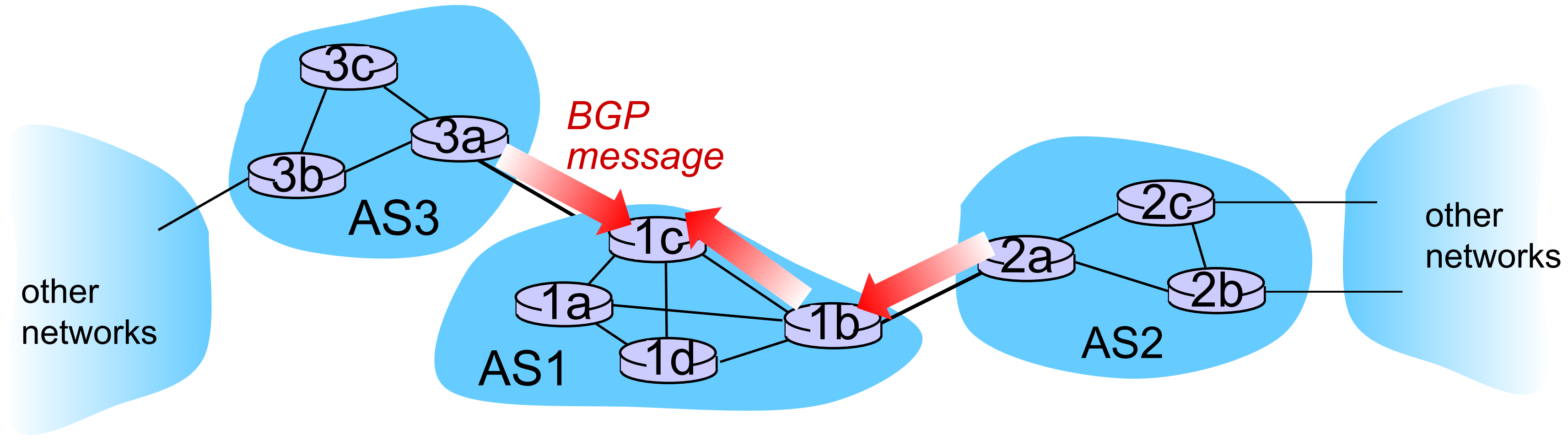
1. Router becomes aware of prefix
2. Router determines output port for prefix
3. Router enters prefix-port in forwarding table

# Router becomes aware of prefix



- BGP message contains "routes"
- "route" is a prefix and attributes: AS-PATH, NEXT-HOP,…
- Example: route:
  - Prefix:138.16.64/22 ; AS-PATH: AS3 AS131 ; NEXT-HOP: 201.44.13.125

# Router may receive multiple routes



- Router may receive multiple routes for <u>same</u> prefix
- Has to select one route

# Select best BGP route to prefix

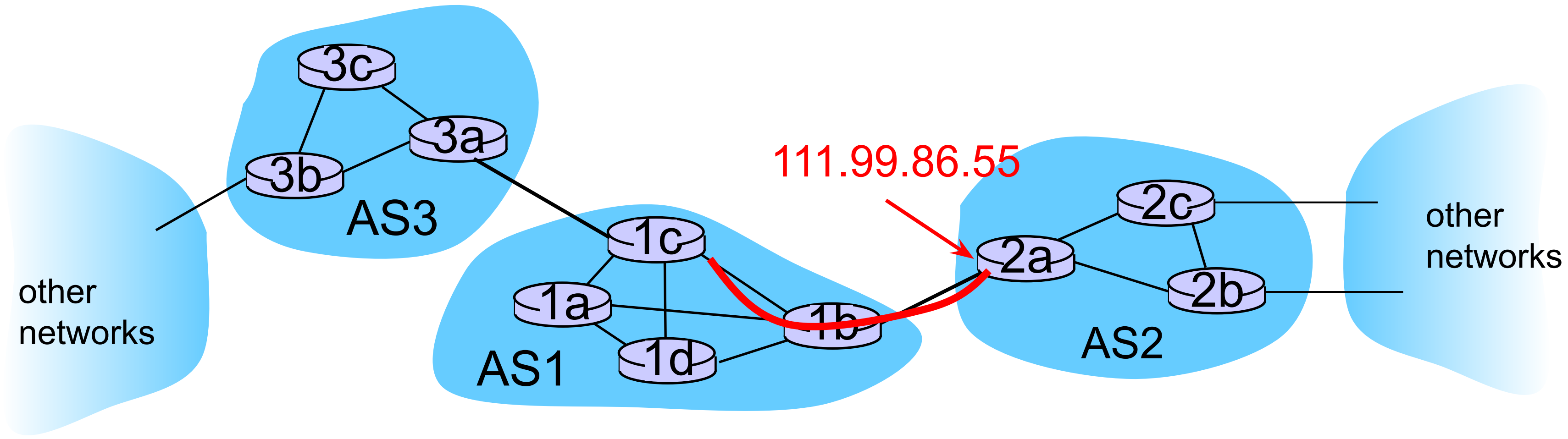
- Router selects route based on shortest AS-PATH

- Example:
  - AS2 AS17  to 138.16.64/22
  - AS3 AS131 AS201 to 138.16.64/22

select

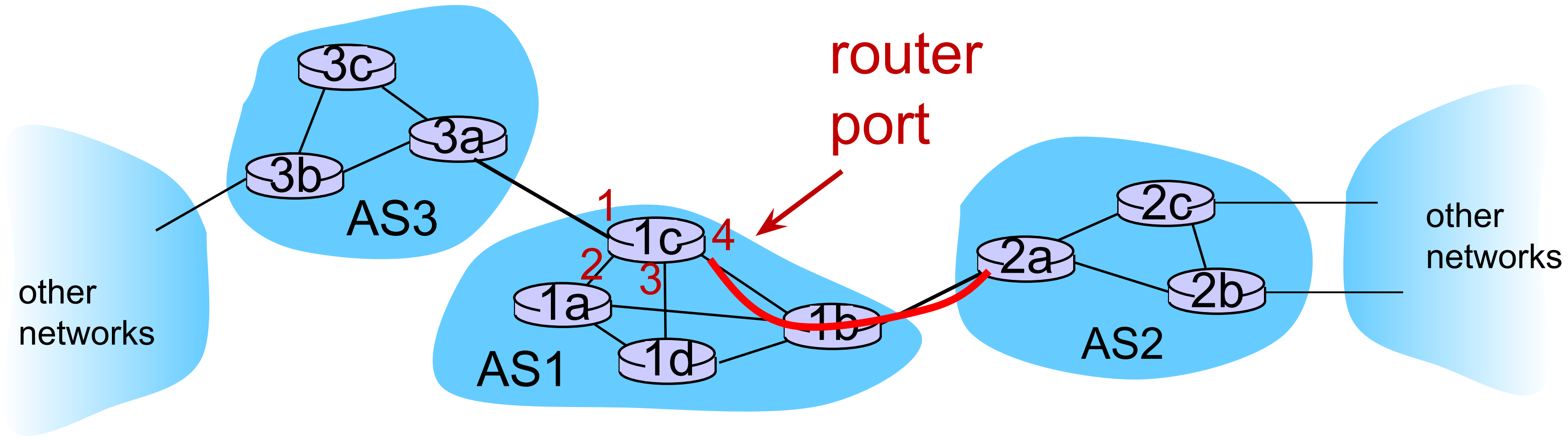
# Find best intra-route to BGP route

- Use selected route's NEXT-HOP attribute
  - Route's NEXT-HOP attribute is the IP address of the router interface that begins the AS PATH.
- Example:
  - AS-PATH: AS2 AS17 ; NEXT-HOP: 111.99.86.55
- Router uses OSPF to find shortest path from 1c to 111.99.86.55



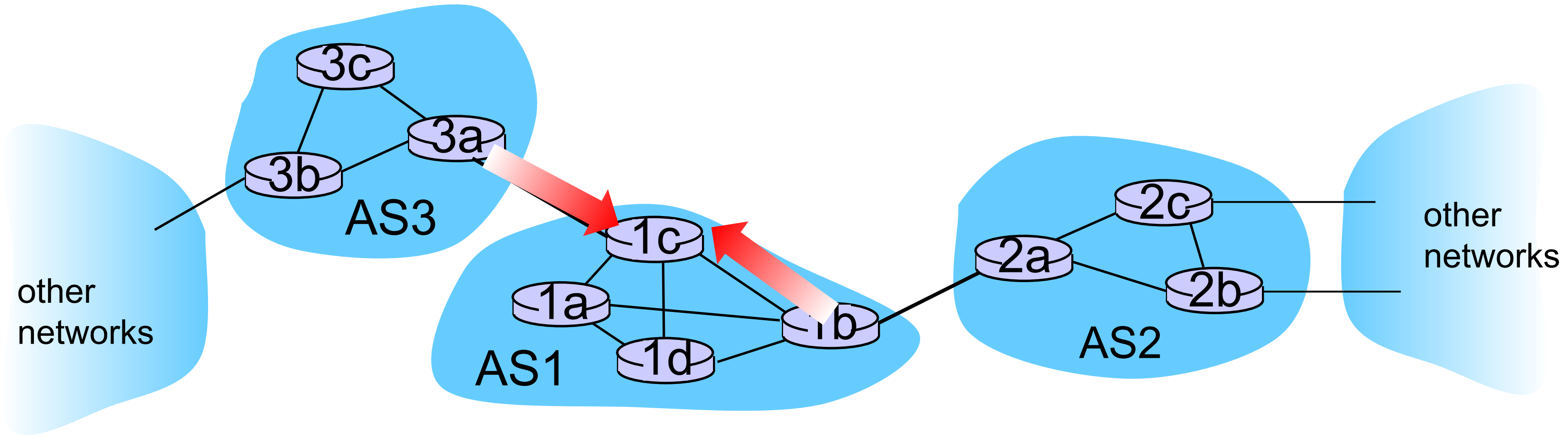
# Router identifies port for route

- Identifies port along the OSPF shortest path
- Adds prefix-port entry to its forwarding table:
  - (138.16.64/22 , port 4)



# Hot Potato Routing

- Suppose there two or more best inter-routes.
- Then choose route with closest NEXT-HOP
  - Use OSPF to determine which gateway is closest
  - Q: From 1c, chose AS3 AS131 or AS2 AS17?
  - A: route AS3 AS131 since it is closer



# How does entry get in forwarding table?

<u>Summary</u>

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

## Why different Intra-, Inter-AS routing ?

*Reading Assignment*