Computer Networks
@CS.NCTU

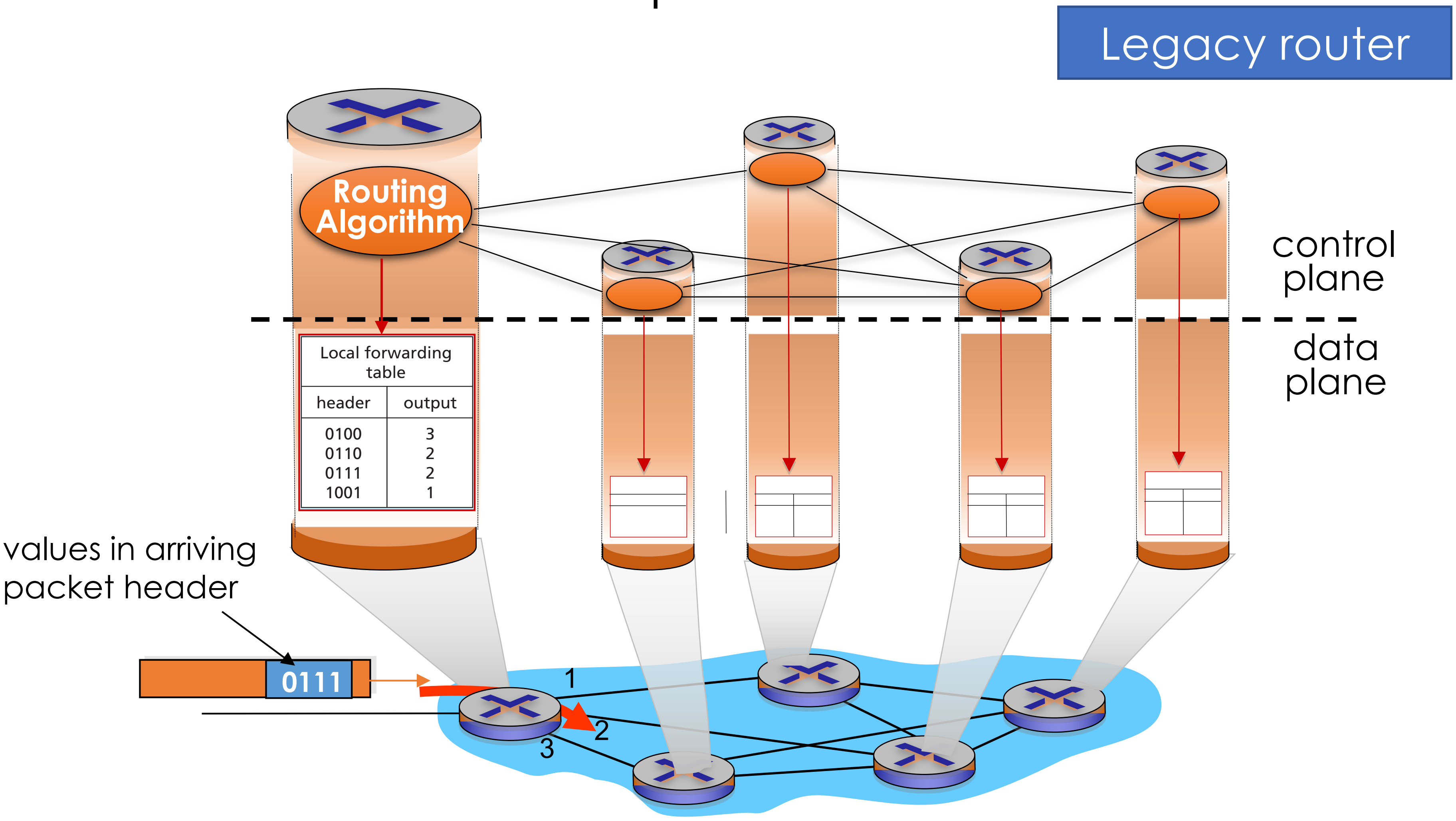
Lecture 5: Network Layer: Control Plane

Instructor: Kate Ching-Ju Lin (林靖茹)

Slides modified from
"Computer Networking: A Top-Down Approach" 7th Edition

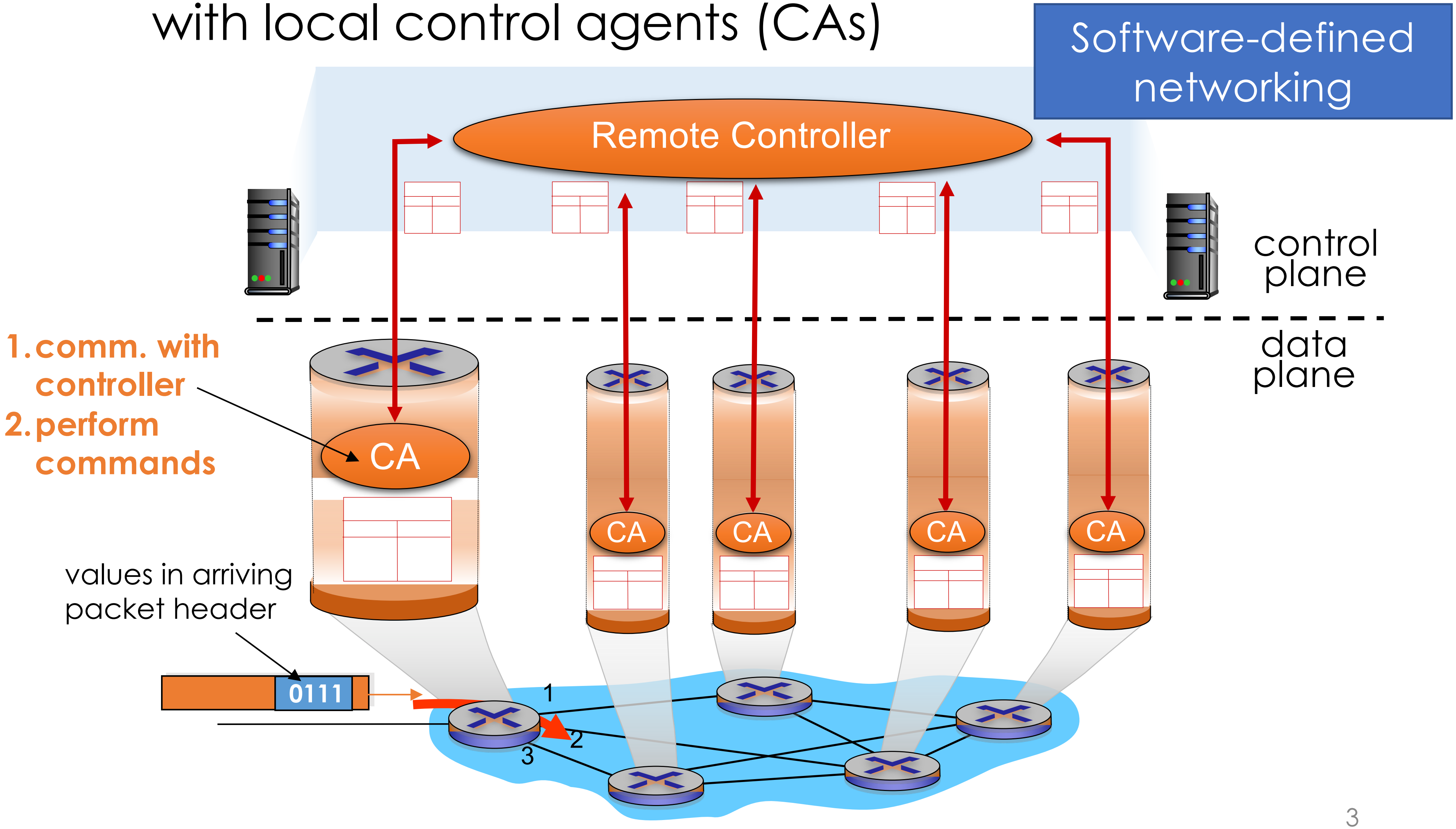
Per-Router Control Plane

- Individual forwarding components in every router interact in the control plane



Centralized Control Plane

- A distinct (typically remote) controller interacts with local control agents (CAs)



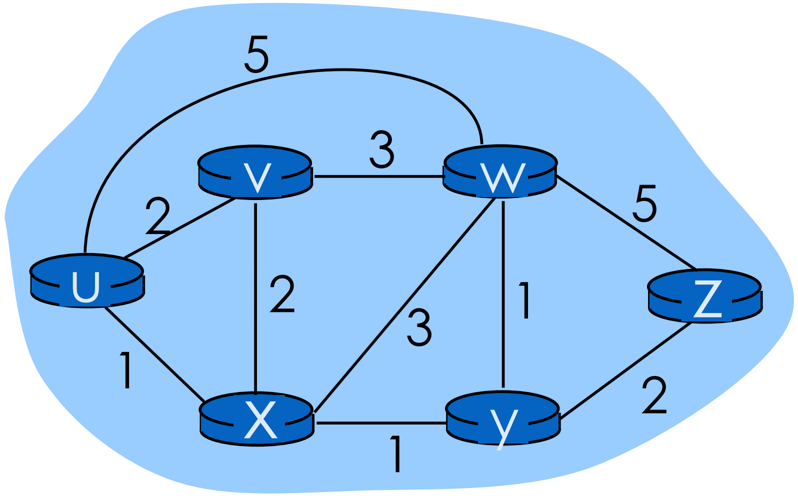
Outline

- **Routing**
  - Link-State Algorithm
  - Distance-Vector Algorithm
- Intra-AS Routing
- Inter-AP Routing
- SDN Control Plane
- ICMP
- SNMP

What is Routing?

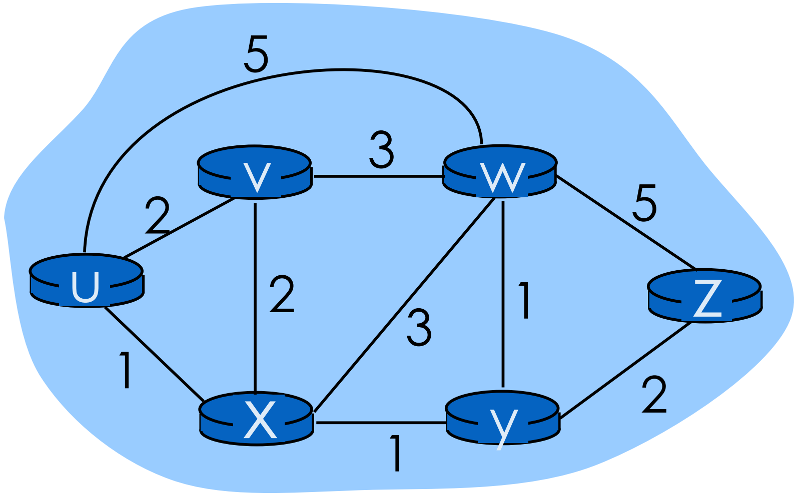
- Find a good **path** from a sender to a receiver, through the network of routers
  - Sequence of routers connecting a sender to a receiver
- How to evaluate the quality of a path?
  - Has the **least cost** → **cost can be:**
    - Number of hops
    - Latency
    - Link geometric distance
    - Bandwidth
    - Prices
  - Usually the "cost" can be elastically set, not defined in the standard

Graph Abstraction



- Graph: G(V,E)
- V: set of vertices (routers)
  - {u, v, w, x, y, z}
- E: set of edges (links)
  - {(u,v), (u,w), (u,x), (v,w), (v,x), (w,x), (w,y), (w,z), (x,y), (y,z)}
  - Each link (x,x') has a cost c(x,x')
  - Edges are usually undirectional

Graph Abstraction: Path



If so, how can it be difficult?

- Large graph
- Many flows
- Partial information
- Dynamic demands
- ...

- Path: a sequence of vertices
  - $(x_1, x_2, x_3, \ldots, x_p)$
- Cost of a path
  - $c(x_1,x_2) + c(x_2,x_3) + \ldots + c(x_{p-1},x_p)$
- Routing
  - Find a **loop-free path** with the least cost
    → find a shortest path! (e.g., Dijkstra's algorithm)
  - Might have many feasible least-cost paths

Routing Algorithm Classification

Centralized or Decentralized?

- **Link-State (LS) algorithms**
  - Centralized, based on global knowledge of the graph
- **Distance-vector (DV) algorithms**
  - Decentralized, each router makes decisions based only on local information (e.g., link costs to neighbors)
  - Iteratively converge to optimal routing

Static or Dynamic

- **Static**
  - Routes change slowly, less overhead
- **Dynamic**
  - React to changes in link costs, higher overhead

Outline

- Routing
  - **Link-State Algorithm**
  - Distance-Vector Algorithm
- Intra-AS Routing
- Inter-AP Routing
- SDN Control Plane
- ICMP
- SNMP

Link-State Routing

- All link costs are known
  - Each router broadcasts link-state information to all others
  - All the routers have an identical global view

- Find the shortest path to all possible destinations
  - Use Dijkstra or other shortest-path algorithms

- Update the forwarding table of each router for a particular source

LS Routing: Example for u

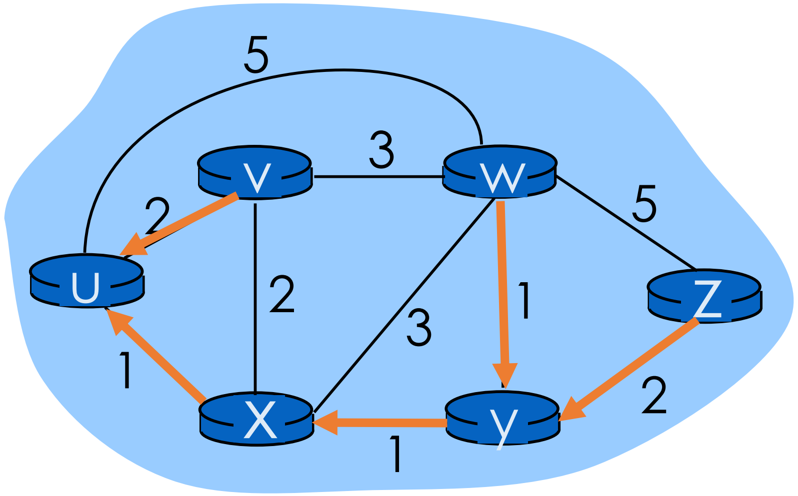
Step	N'	D(v) p(v)	D(w) p(w)	D(x) p(x)	D(y) p(y)	D(z) p(z)
0	u	2,u	5,u	1,u	∞	∞
1	ux	2,u	4,x		2,x	∞
2	uxy	2,u	3,y			4,y
3	uxyv		3,y			4,y
4	uxyvw					4,y
5	uxyvwz					

notes:

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

algorithm:

1. Find the one, n, with **min cost**
2. Update the cost of each neighbor m to **min(D(m), D(n) + c(n,m))**

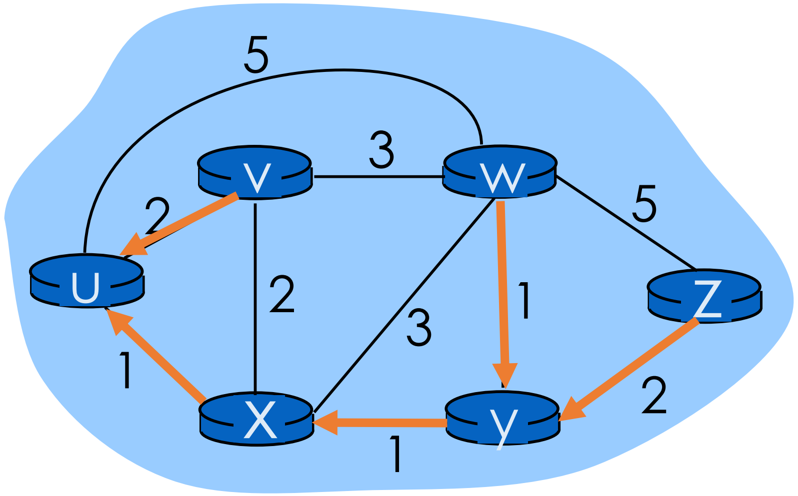


LS Routing: Example for u

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					

Forwarding table for router u

Destination	Link
v	(u, v)
w	(u, x)
x	(u, x)
y	(u, x)
z	(u, x)



Dijkstra's Algorithm

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

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

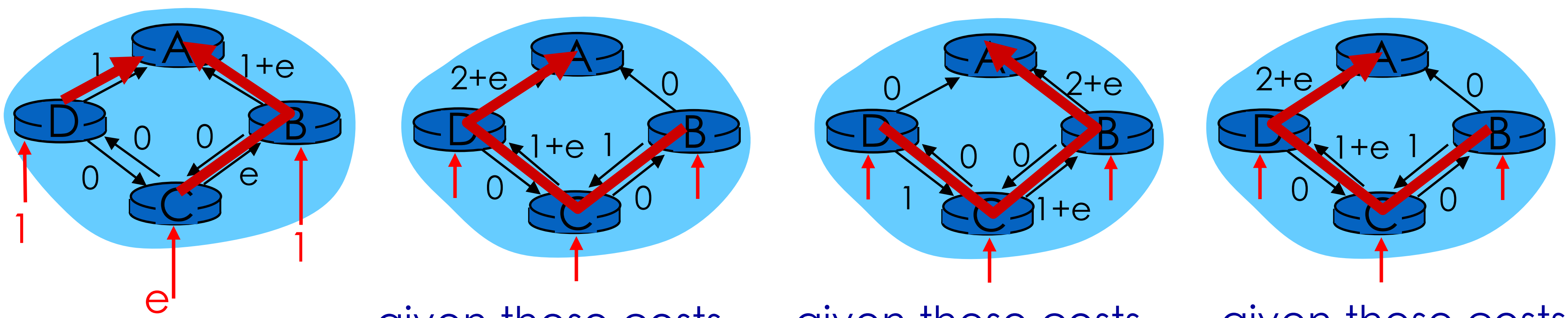
Complexity: (n nodes)

- n(n+1)/2 comparisons: $O(n^2)$
- more efficient implementations possible: $O(n \log n)$

Oscillation problem

- For example, if we set link cost equal to the amount of carried traffic
  → frequent link cost change could lead to oscillation

D→A: 1 B→A: 1 C→A: e



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

Possible Solutions to Oscillation

1. Avoid using the amount of traffic load as link costs
   - Not that practical, since we usually want to find a path with a lighter load → load balancing

2. Not all routers run the LS algorithm at the same time
   - More practical
   - How: each router has a randomized interval to broadcast link advertisement

Outline

- Routing
  - Link-State Algorithm
  - **Distance-Vector Algorithm**
- Intra-AS Routing
- Inter-AP Routing
- SDN Control Plane
- ICMP
- SNMP

Distance-Vector Routing

- **Distributed**
  - Each router only receives information from directly attached neighbors
  - Calculate its forwarding option locally
- **Iterative**
  - Continue updating its forwarding rules until convergence
  - Convergence = local information stabilized
- **Asynchronous**
  - Does not require all nodes to operate in lockstep with each other
  - That is, nodes can update their forwarding rules either simultaneously or not

Bellman-Ford Algorithm

- A shortest path algorithm based on dynamic programming
- Intuition: (for a node x)
  1. Get the least cost from a neighbor v to destination
  2. Check whether the cost can be reduced if traffic goes through v
  3. Identify the neighbor that minimizes the e2e cost
- **Bellman-Ford equation**

Only need to know the cost, not sub-path

$$d_x(y) = \mathit{min}_v \{c(x,v) + d_v(y)\}$$

→ min cost from x through v* to y

→ $v^* = \arg \mathit{min}_v \{c(x,v) + d_v(y)\}$ is the next hop

Idea of DV Routing

iterative, asynchronous:

each local iteration caused by:

- local link cost changes
- DV update message from neighbors

distributed:

- each node notifies neighbors only when its DV changes
- broadcast DV to 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

DV Algorithm

For a router x,

1. For each neighbor v, get the cost c(x,v)
2. Maintain a distance vector:
   - each element is the estimated cost from x to any destination y

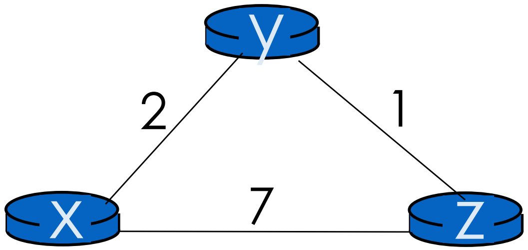
   $\mathbf{D}_x = [D_x(y)\text{: } y \text{ in } N\ ]$

3. Periodically request for the distance vector of its neighbor v, i.e., $\mathbf{D}_v$
4. Update its DV, $\mathbf{D}_x$, based on BF equation

   $D_x(y) = \mathit{min}_v\ \{c(x,v) + D_v(y)\ \}$, for each y in N

5. Skip if the local info., $\mathbf{D}_x$ and c(x,v), is unchanged

Simple Example (Step 1.1)



For x:

1. $D_y = \{2, 0, 1\}$
2. $D_z = \{7, 1, 0\}$

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

x's table

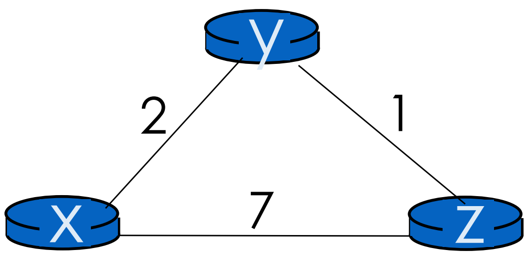
from \ cost to	x	y	z
x	∞	∞	∞
y	2	0	1
z	∞	∞	∞

y's table

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

z's table

Simple Example (Step 1.2)



For x:

1. $D_y = \{2, 0, 1\}$
2. $D_z = \{7, 1, 0\}$
3. $D_x = \{0, 2, 7\}$
4. $D_{x\text{ via }y} = D_y + c(x, y) = \{4, 2, 3\}$
5. $D_{x\text{ via }z} = D_z + c(x, z) = \{14, 8, 7\}$

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

x's table

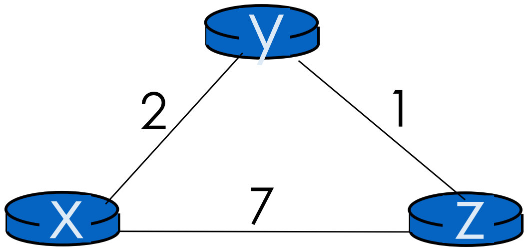
from \ cost to	x	y	z
x	0	2	7
y	2	0	1
z	7	1	0

y's table

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

z's table

Simple Example (Step 2.3)



For x:

1. $D_y = \{2, 0, 1\}$
2. $D_z = \{7, 1, 0\}$
3. $D_x = \{0, 2, 7\}$
4. $D_{x\ via\ y} = D_y + c(x, y) = \{4, 2, 3\}$
5. $D_{x\ via\ z} = D_z + c(x, z) = \{14, 8, 7\}$

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

x's table

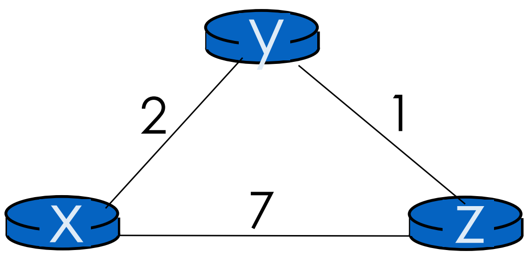
from \ cost to	x	y	z
x	0	2	7
y	2	0	1
z	7	1	0

y's table

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

z's table

Simple Example (Step 3.1)



For x:

1. $D_y = \{2, 0, 1\}$
2. $D_z = \{3, 1, 0\}$

x's table (from \ cost to)

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

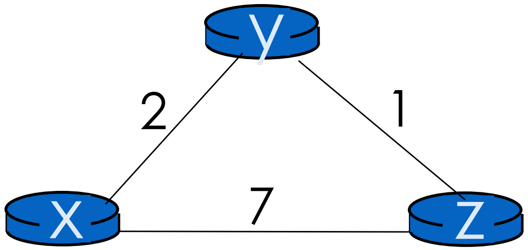
y's table (from \ cost to)

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

z's table (from \ cost to)

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

Simple Example (Step 3.2)



x and z broadcast the updated distance vector to neighbors

x's table

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

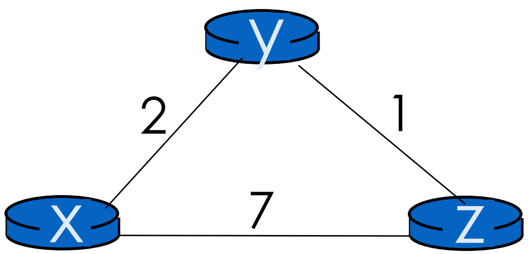
y's table

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

z's table

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

Simple Example (Step 3.3)



For x:

1. $D_y = \{2, 0, 1\}$
2. $D_z = \{3, 1, 0\}$
3. $D_x = \{0, 2, 3\}$
4. $D_{x \text{ via } y} = D_y + c(x, y) = \{4, 2, 3\}$
5. $D_{x \text{ via } z} = D_z + c(x, z) = \{10, 8, 7\}$

all min → unchanged!

x's table (cost to)

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

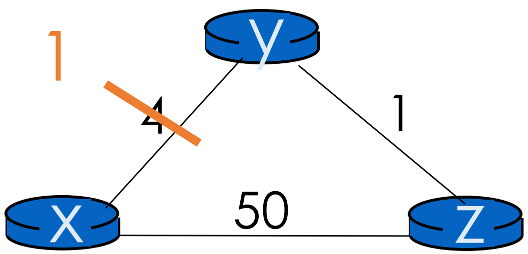
y's table (cost to)

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

z's table (cost to)

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

Link-Cost Changes



For x:

1. D_x = {0, 1, 50}
2. $D_{x\ via\ y} = D_y + c(x, y) =$ {5, 1, 2}
3. $D_{x\ via\ z} = D_z + c(x, z) =$ {55, 51, 50}
4. Broadcast D_x = {0, 1, 2}

x's table

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

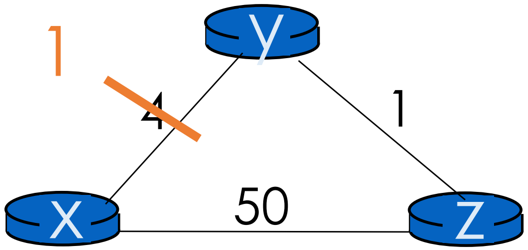
y's table

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

z's table

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

Link-Cost Changes



For y:

1. $D_y = \{1, 0, 1\}$
2. $D_{y \text{ via } x} = D_x + c(y, x) = \{1, 2, 3\}$

from \ cost to	x	y	z
x	0	1	2
y	4	0	1
z	5	1	0

x's table

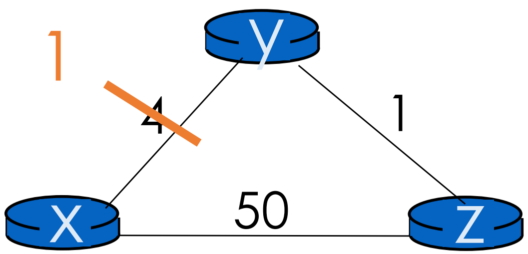
from \ cost to	x	y	z
x	0	1	2
y	4	0	1
z	5	1	0

y's table

from \ cost to	x	y	z
x	0	1	2
y	4	0	1
z	5	1	0

z's table

Link-Cost Changes



For y:

1. $D_y \qquad\qquad\qquad\qquad = \{4, 0, 1\}$
2. $D_{y \text{ via } x} = D_x + c(y, x) = \{1, 2, 3\}$

x's table

from \ cost to	x	y	z
x	0	1	2
y	1	0	1
z	5	1	0

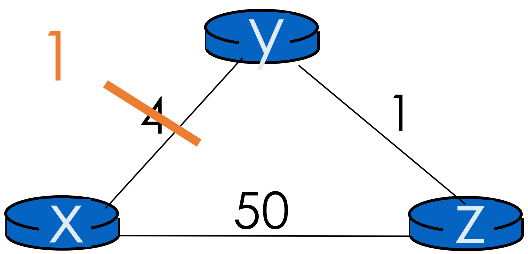
y's table

from \ cost to	x	y	z
x	0	1	2
y	1	0	1
z	5	1	0

z's table

from \ cost to	x	y	z
x	0	1	2
y	1	0	1
z	5	1	0

Link-Cost Changes



For z:

1. $D_z = \{5, 1, 0\}$
2. $D_{z \text{ via } x} = D_x + c(z, x) = \{50, 51, 52\}$
3. $D_{z \text{ via } y} = D_y + c(z, y) = \{2, 1, 2\}$

from \ cost to	x	y	z
x	0	1	2
y	1	0	1
z	5	1	0

x's table

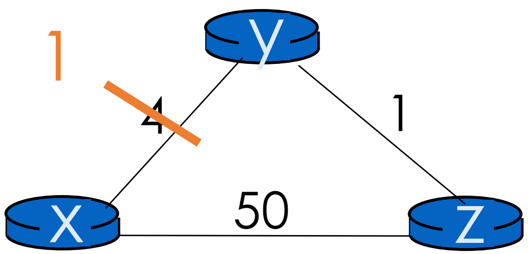
from \ cost to	x	y	z
x	0	1	2
y	1	0	1
z	5	1	0

y's table

from \ cost to	x	y	z
x	0	1	2
y	1	0	1
z	5	1	0

z's table

Link-Cost Changes



For z:

1. $D_z = \{5, 1, 0\}$
2. $D_{z \text{ via } x} = D_x + c(z, x) = \{50, 51, 52\}$
3. $D_{z \text{ via } y} = D_y + c(z, y) = \{2, 1, 2\}$

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

x's table

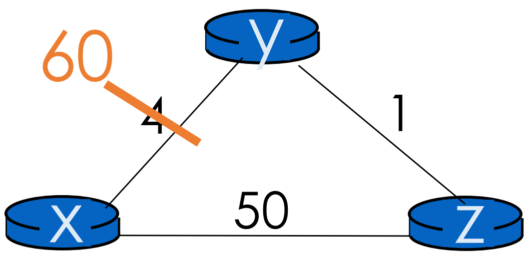
from \ cost to	x	y	z
x	0	1	2
y	1	0	1
z	2	1	0

y's table

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

z's table

"Count to Infinity" Problem



✓

For y:

1. Detect c(y,x) = 60
2. D_y = {60, 0, 1}
3. $D_{y \text{ via } x} = D_x + c(y, x) =$ {60, 64, 65}
4. $D_{y \text{ via } z} = D_z + c(y, z) =$ {6, 2, 1}

Issue 1: The cost via z is wrong!!

Issue 2: to get to x, y→z and z→y → loop!

x's table

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

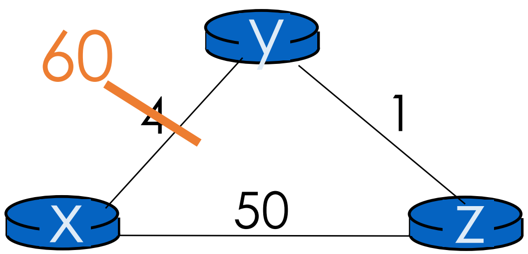
y's table

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

z's table

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

"Count to Infinity" Problem



For z:

1. $D_z = \{50, 1, 0\}$
2. $D_{z\ \text{via}\ x} = D_x + c(z, x) = \{50, 54, 55\}$
3. ✓ $D_{z\ \text{via}\ y} = D_y + c(z, y) = \{7, 1, 2\}$

Issue 1: The cost via z is wrong!!

Issue 2: to get to x, y→z and z→y → loop!

from \ cost to	x	y	z
x	0	4	5
y	6	0	1
z	5	1	0

x's table

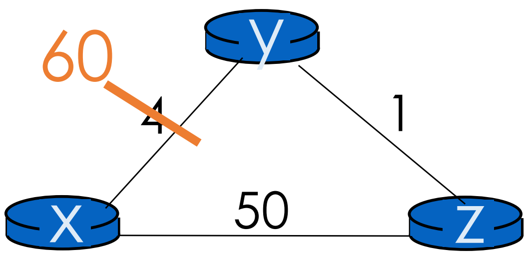
from \ cost to	x	y	z
x	0	4	5
y	6	0	1
z	5	1	0

y's table

from \ cost to	x	y	z
x	0	4	5
y	6	0	1
z	5	1	0

z's table

"Count to Infinity" Problem



$D_y(x) = 6,\ D_z(x) = 7$

$D_y(x) = 8,\ D_z(x) = 9$

⋮

$D_y(x) = 48,\ D_z(x) = 49$

x's table (cost to)

from	x	y	z
x	0	4	5
y	6	0	1
z	7	1	0

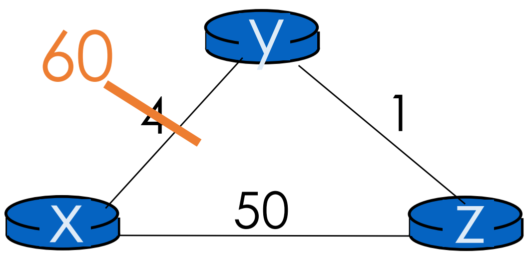
y's table (cost to)

from	x	y	z
x	0	4	5
y	6	0	1
z	7	1	0

z's table (cost to)

from	x	y	z
x	0	4	5
y	6	0	1
z	7	1	0

"Count to Infinity" Problem



For y:

1. $D_y = \{60, 0, 1\}$
2. $D_{y \text{ via } x} = D_x + c(y, x) = \{60, 64, 65\}$
3. $D_{y \text{ via } z} = D_z + c(y, z) = \{50, 2, 1\}$

x's table

from \ cost to	x	y	z
x	0	4	5
y	48	0	1
z	49	1	0

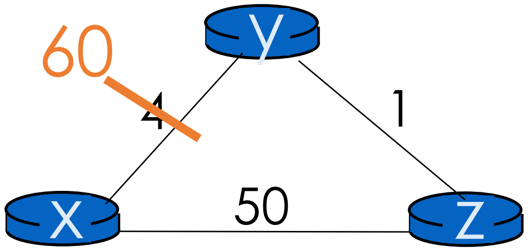
y's table

from \ cost to	x	y	z
x	0	4	5
y	48	0	1
z	49	1	0

z's table

from \ cost to	x	y	z
x	0	4	5
y	48	0	1
z	49	1	0

"Count to Infinity" Problem



For z:

1. $D_z = \{50, 1, 0\}$
2. $D_{z\ via\ x} = D_x + c(z, x) = \{50, 54, 55\}$
3. $D_{z\ via\ y} = D_y + c(z, y) = \{51, 1, 2\}$

from \ cost to	x	y	z
x	0	4	5
y	50	0	1
z	49	1	0

x's table

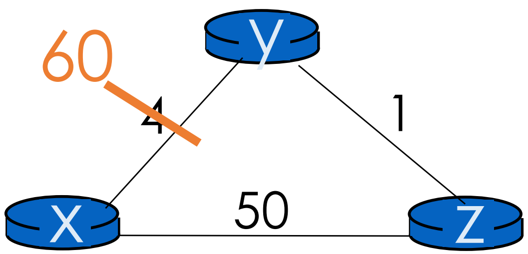
from \ cost to	x	y	z
x	0	4	5
y	50	0	1
z	49	1	0

y's table

from \ cost to	x	y	z
x	0	4	5
y	50	0	1
z	49	1	0

z's table

"Count to Infinity" Problem



For z:

1. $D_z = \{50, 1, 0\}$
2. $D_{z\ via\ x} = D_x + c(z, x) = \{50, 54, 55\}$
3. $D_{z\ via\ y} = D_y + c(z, y) = \{51, 1, 2\}$

to get to x, y→z and z→x → Loop free!

Take 44 iterations!!

from \ cost to	x	y	z
x	0	4	5
y	50	0	1
z	50	1	0

x's table

from \ cost to	x	y	z
x	0	4	5
y	50	0	1
z	50	1	0

y's table

from \ cost to	x	y	z
x	0	4	5
y	50	0	1
z	50	1	0

z's table

Comparison of LS and DV

message complexity

- LS: with n nodes, E links, O(nE) msgs sent
- DV: exchange between neighbors only

speed of convergence

- LS: O(n^2) algorithm requires O(nE) msgs
  - may have oscillations
- DV: convergence time varies
  - may have 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:
  - node can advertise incorrect path cost
  - each node's table used by others
  - error propagate thru network

Outline

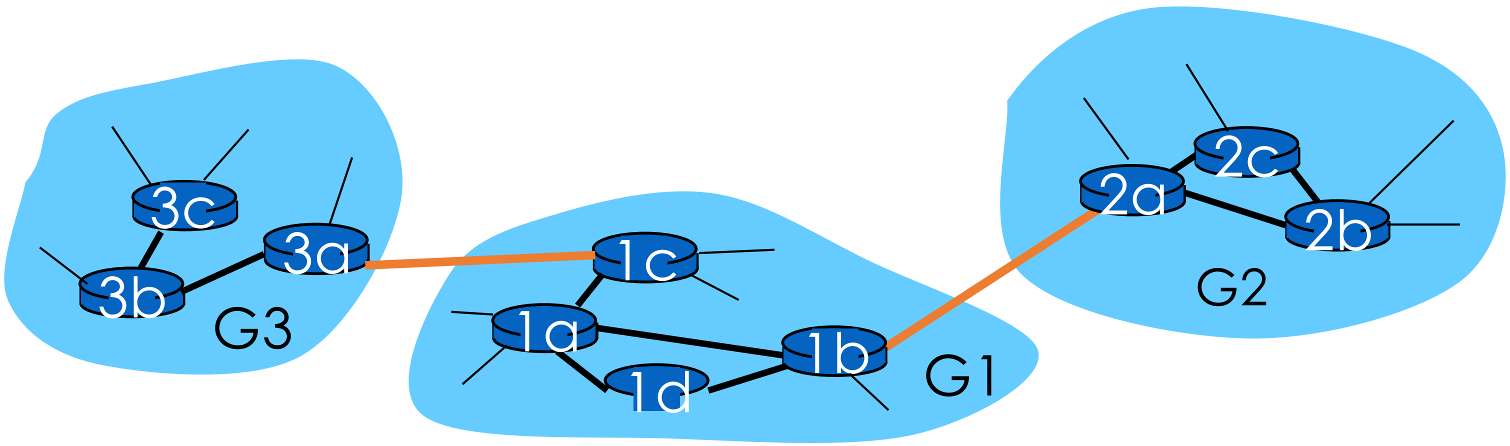
- Routing
  - Link-State Algorithm
  - Distance-Vector Algorithm
- **Intra-AS Routing**
- Inter-AP Routing
- SDN Control Plane
- ICMP
- SNMP

Scalability?

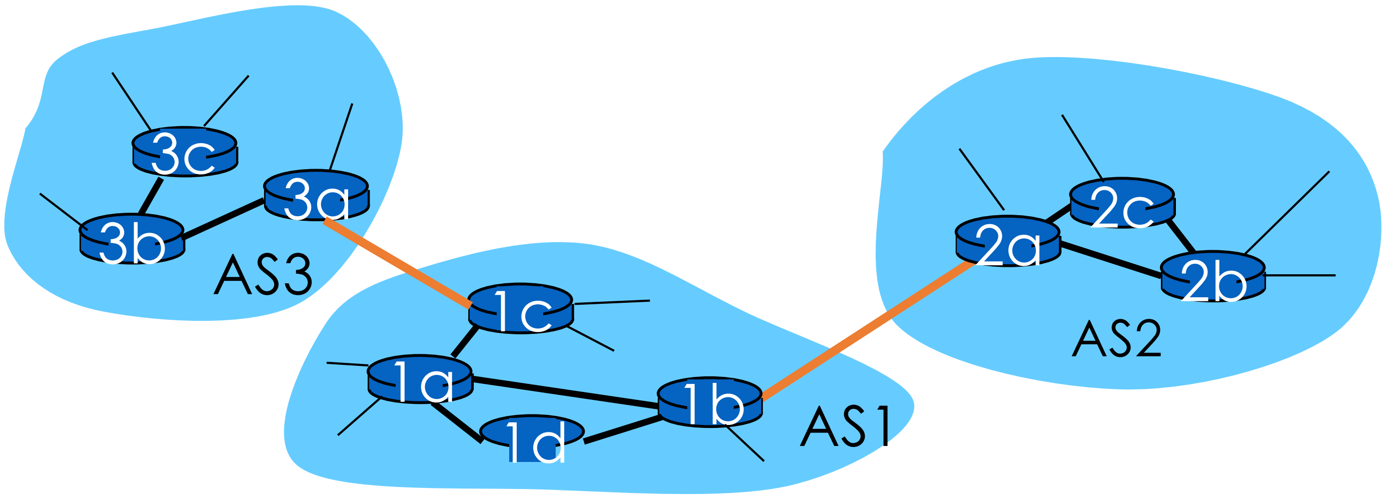
- So far, we consider simple examples with only a few nodes
- In reality,
  - **Scale:** Hundreds of millions of routers in today's Internet
    - LS: huge overhead
    - DV: may never converge
  - **Heterogeneous:** Routers might not all function in the same way (not homogeneous)
  - **Administrative autonomy:** an ISP may hide internal organization from outside

Hierarchical Routing!

- Group a subset of nodes as a group
- Each group as one (or multiple) gateways
- Inter-group: each gateway knows who is the next "group"
  - e.g., G3 → G1 → G2
- Intra-group: each node only needs to know how to reach any gateway
  - e.g., 1c → 1a → 1d → 1b in G1

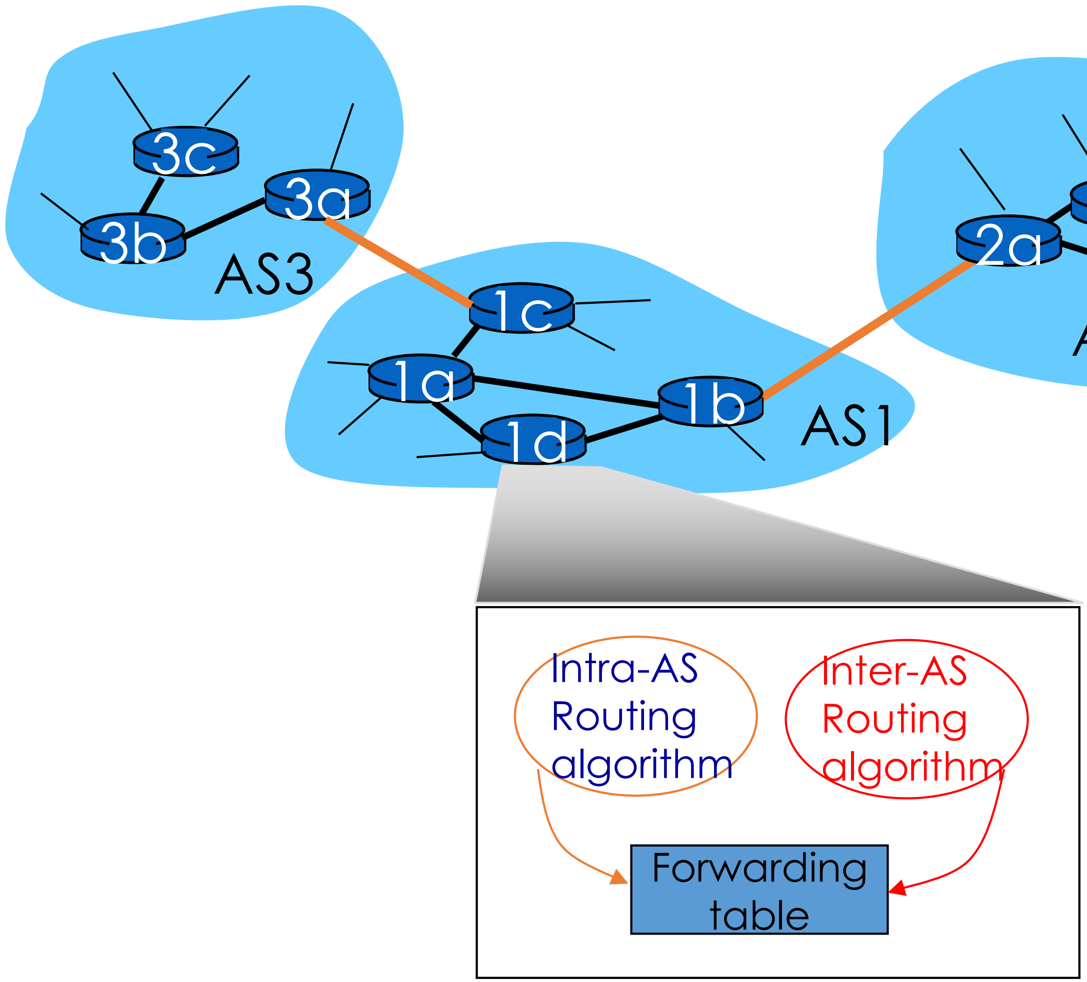


Autonomous System (AS)



- A group of routers that are under the same administrative control
- Usually the routers in an ISP and interconnected via physical links
- Each AS has a unique ID (again assigned by ICANN)

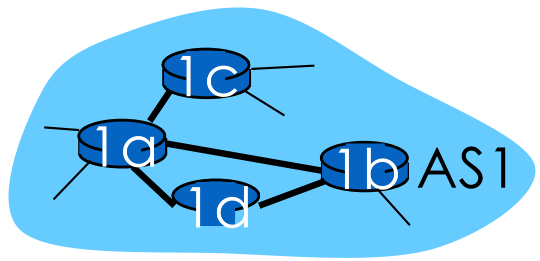
Intra- and Inter-AS Routing



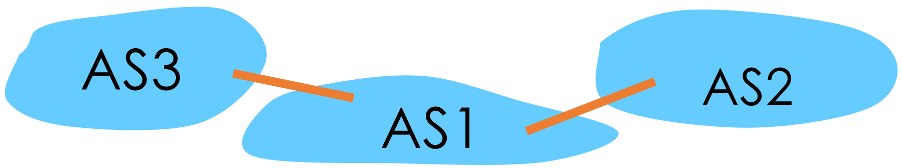
- forwarding table configured by both intra- and inter-AS routing
  - intra-AS routing determine entries for destinations within AS
  - inter-AS & intra-AS determine entries for external destinations

Intra- and Inter-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)



- Most common inter-AS routing protocol:
  - **BGP**: Border Gateway Protocol



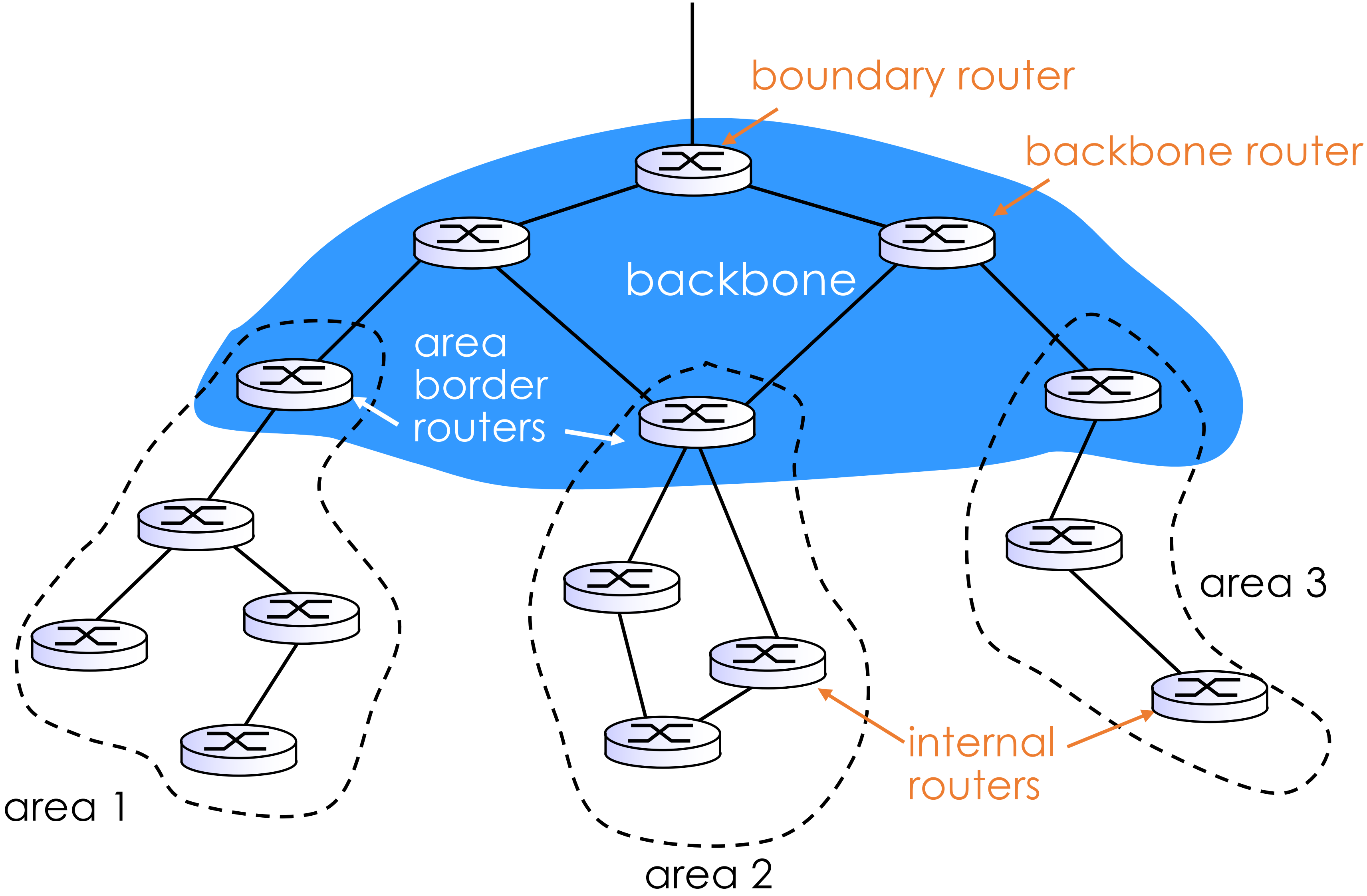
OSPF (Open Shortest Path First)

- "open": publicly available
- Based on link-state algorithm
  - link state packet dissemination
  - topology map at each node
  - route computation using Dijkstra's algorithm
  - no policy about how link weights are set
- Router floods OSPF link-state advertisements to all other routers in entire AS
  - carried in OSPF messages directly over IP (protocol 89), rather than TCP or UDP
  - link state: for each attached link
- IS-IS routing protocol: nearly identical to OSPF

OSPF (Open Shortest Path First)

- **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 (type of service)
- Integrated uni- and multi-cast support:
  - Multicast OSPF (MOSPF) uses same topology data base as OSPF
- Hierarchical OSPF in large domains

Hierarchical OSPF



Hierarchical OSPF

- **Two-level hierarchy**: local area, backbone
  - Link-state advertisements only in area
  - Each node 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'es

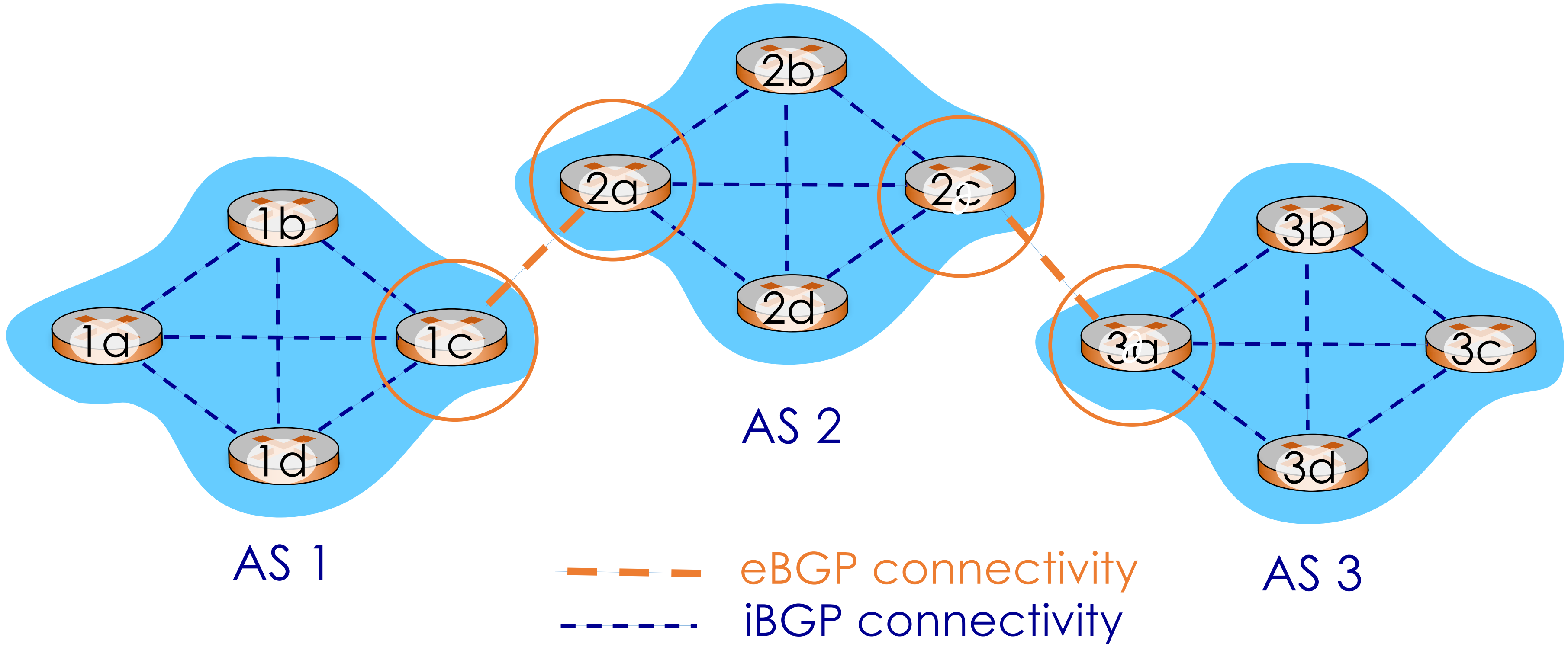
Outline

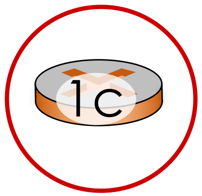
- Routing
  - Link-State Algorithm
  - Distance-Vector Algorithm
- Intra-AS Routing
- **Inter-AP Routing**
- SDN Control Plane
- ICMP
- SNMP

Internet inter-AS routing: BGP

- **BGP (Border Gateway Protocol)**
  - the de facto inter-domain routing protocol
  - glue that holds the Internet together
  - decentralized and asynchronous
- Enable each BGP to
  - **eBGP**: obtain subnet reachability information from neighboring ASes (inter-AS)
  - **iBGP**: reachability information to all AS-internal routers (intra-AS)
  - determine "good" routes to other networks based on reachability and policy
- Allows a subnet to advertise its existence to rest of Internet

eBGP/iBGP Connections

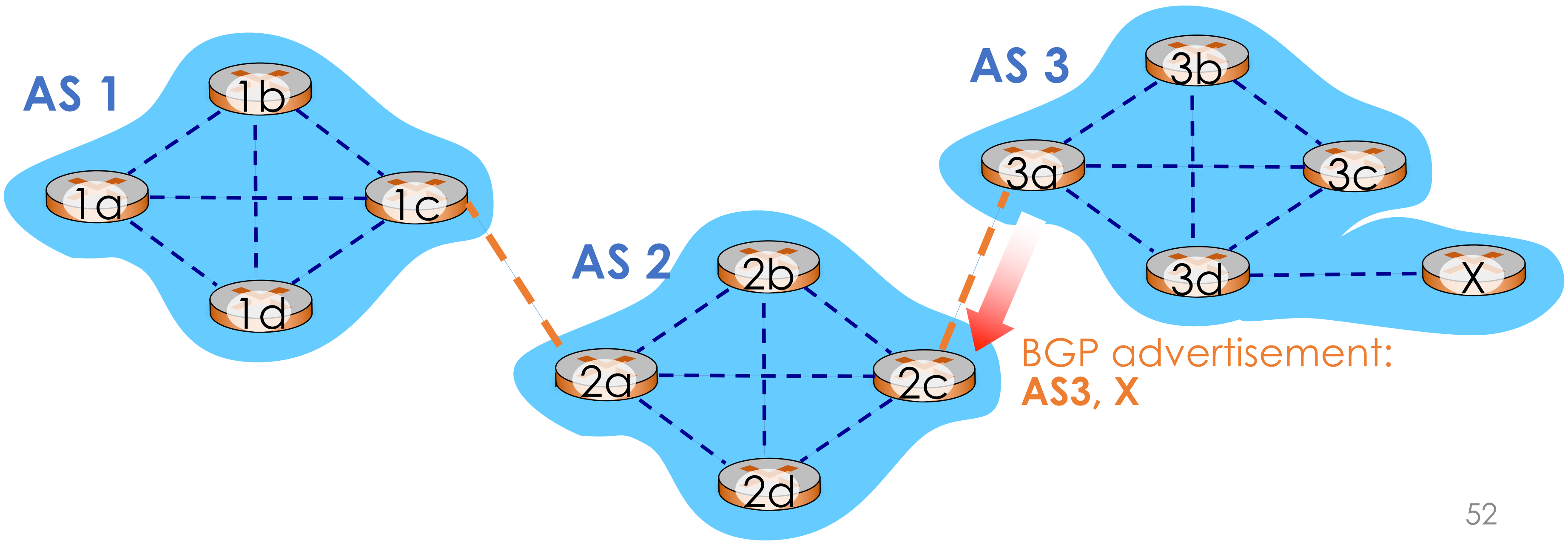




gateway routers run both eBGP and iBGP protocols

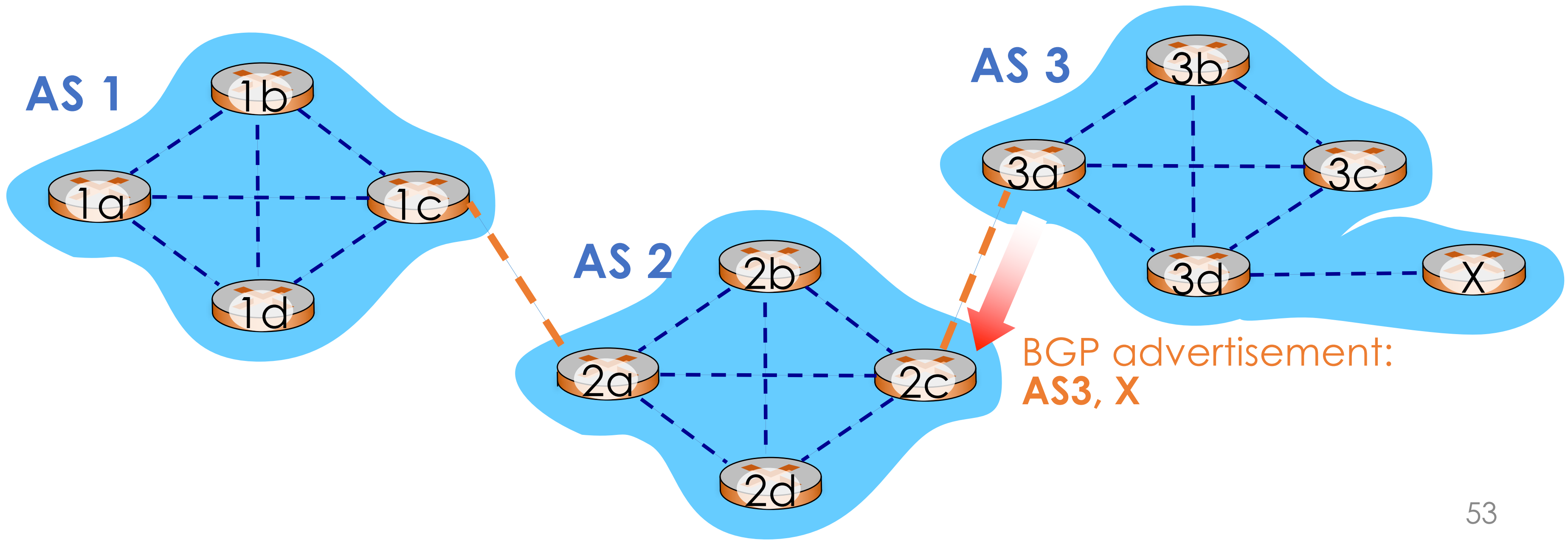
How eBGP Works?

- Two BGP routers ("peers") exchange BGP messages over semi-permanent TCP connection
  - advertise paths to different destination network prefixes
- Route for prefix x
  - AS3 gateway 3a advertises path **AS3,X** to AS2 gateway 2c
  - AS2 gateway 2a advertises path **AS2,AS3,X** to AS1 gateway 1c
  - datagrams forwarded through AS1 → AS2 → AS3



How iBGP Works?

- Given an eBPG path, router use iBGP to send the path to all routers in the same AS
  - Use intra-AS routing
- In reality:
  - an AS might have different paths to a given destination
  - each path is a sequence of ASs (may through different gateways) (e.g., AS1→AS2→AS3→x)



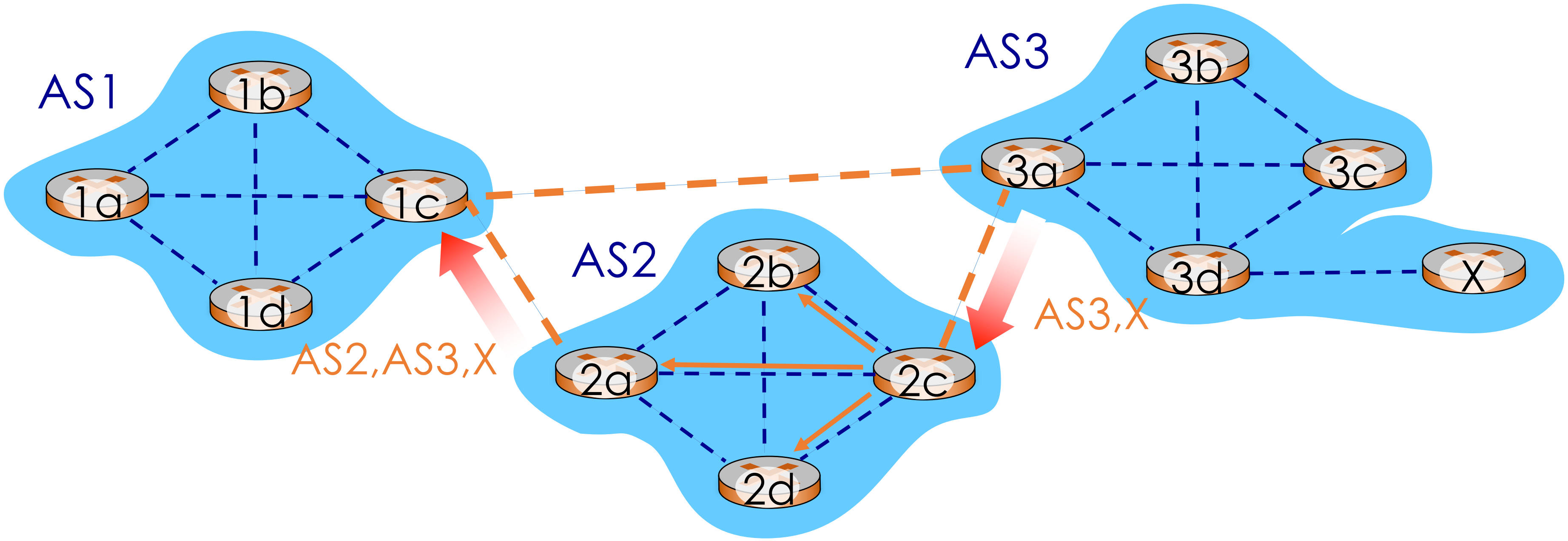
BGP Advertisement

- router advertise a route across BGP connection
  - **Route = “prefix + BGP attributes”**

- **BGP attributes:**
  - AS-PATH: list of ASs of a route, e.g., ”AS3 AS2”
    - Prevent looping
  - NEXT-HOP: IP addr. of the router interface to next AS

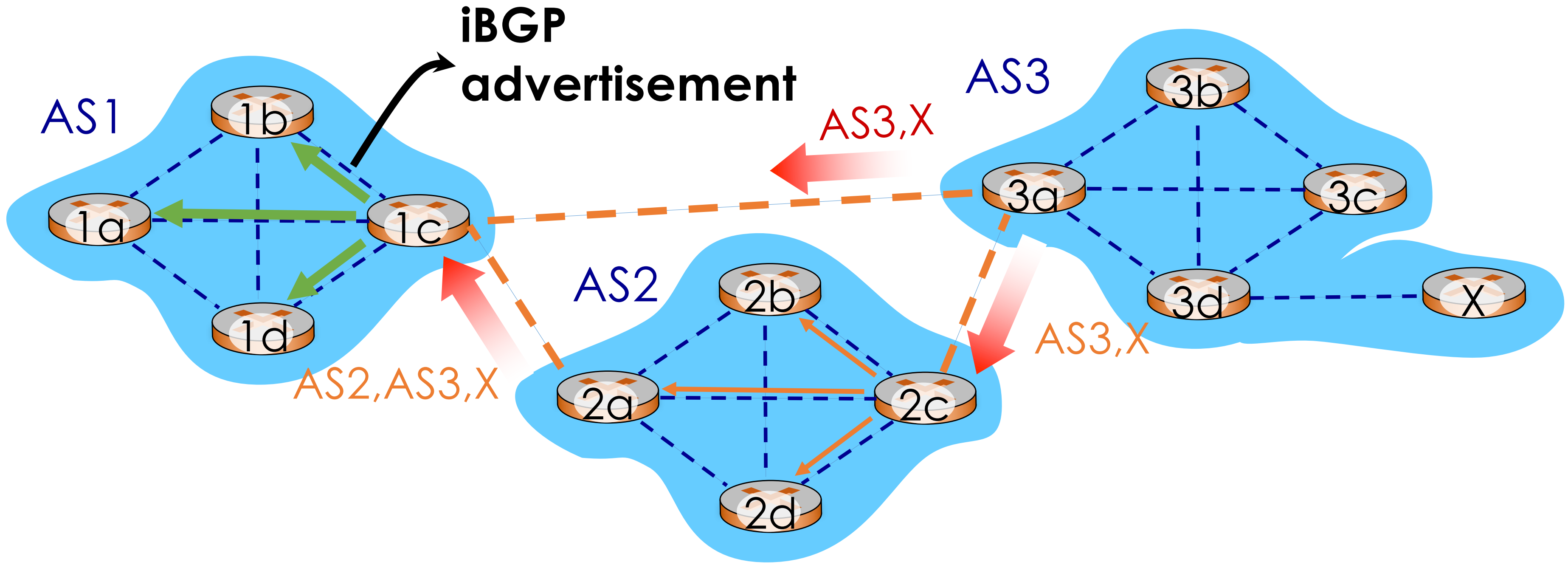
- **Policy-based routing**
  - use policies to accept or decline a path
  - only forward accepted paths to neighboring ASs

Advertisement: Route 1



1. AS2 router 2c receives path advertisement **AS3,X** (via eBGP) from AS3 router 3a
2. Based on AS2 policy, AS2 router 2c accepts path **AS3,X**, propagates (via iBGP) to all AS2 routers
3. Based on AS2 policy, AS2 router 2a advertises (via eBGP) path **AS2, AS3, X** to AS1 router 1c

Advertisement: Route 2

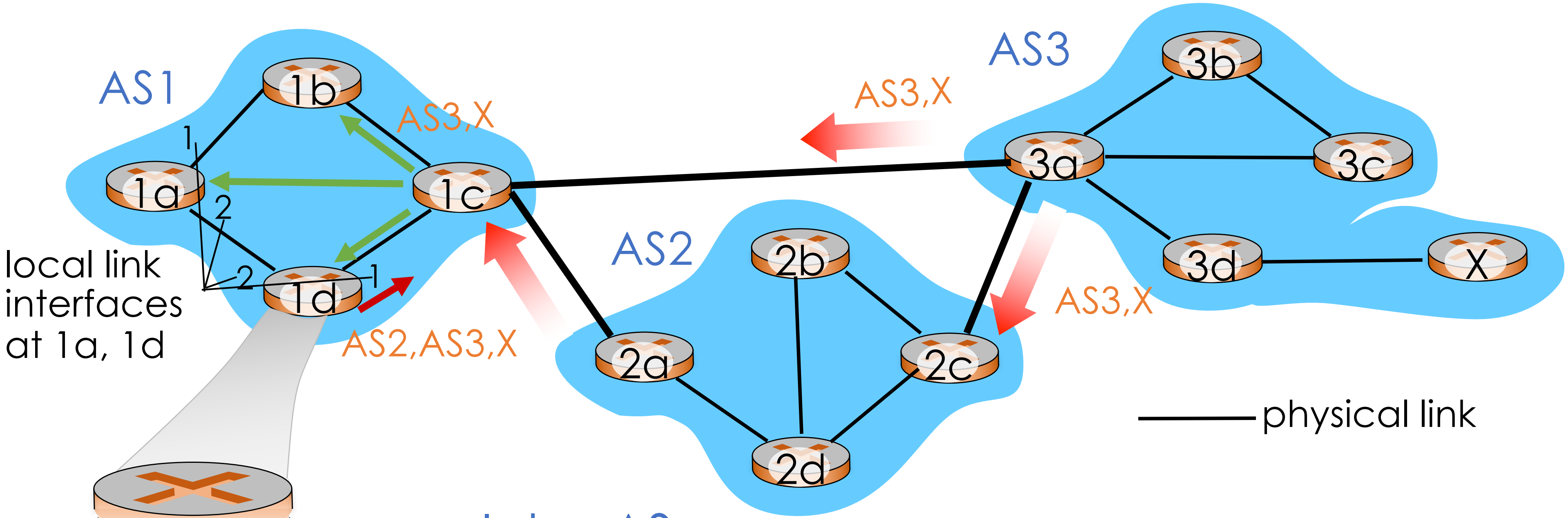


- Gateway router may learn about multiple paths to destination:
  - IP of left interface of 3a: AS3; x   ✔ 1c chooses shorter one
  - IP of bottom interface of 3a: AS3 AS2; x

NEXT-HOP   AS-PATH   Subnet prefix

BGP + OSPF

Q: how does router set flowtable entry to distant prefix?



dest	interface
…	…
X	1
…	…

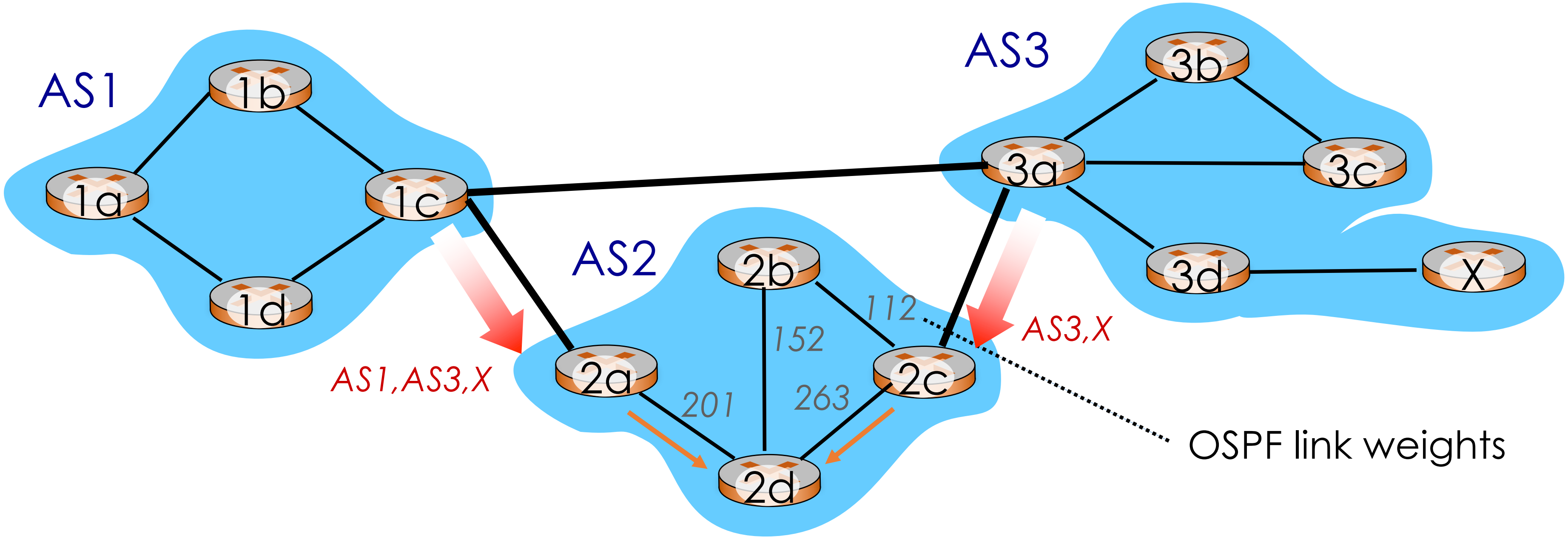
- Intra-AS:
  - 1a, 1b, 1d learn about dest X via iBGP from 1c
  - Use OSPF intra-AS routing to reach 1c e.g., 1d forwards over interface 1
- Inter-AS:
  - Gateway router 1c forwards datagrams to 3a via eBGP

BGP Route Selection

- Select one from the multiple received advertised paths
  - Local preference (policy-based)
  - Shortest AS-PATH
  - Closest NEXT-HOP router (**hot potato routing**)
  - Additional criteria
  - …

Hot Potato Routing

- Choose local gateway that has least intra-domain cost



- 2d can forward to either 2a or 2c
- c(2d,2a) < c(2d, 2c) ➔ forward to 2a
  - selfish protocol, might need a longer end-to-end path

Multi-Rule Routing

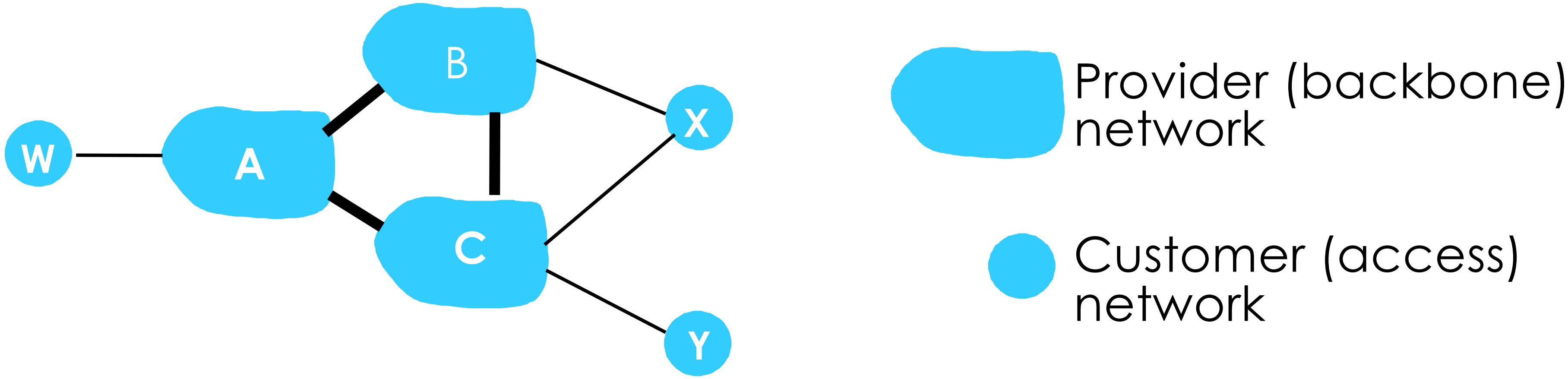
- Sequentially eliminate rules until only one route remains

1. Only keep any route matching local preference (policies)
2. Only keep shortest AS-PATH (e.g., DV but distance = AS hops)
3. Hop potato: only keep the those with the closest NEXT-HOP
4. Keep the one the smallest BGP identifier

Routing Policy

- **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

Routing Policy



- **For access ISP**: X is a multi-homed access ISP → How to prevent X from forwarding traffic between B and C?
  - X always advertises that it is a destination AS
  - e.g., XCY will never be advertised
- **For backbone ISP**: may not advertise due to selfish
  - e.g., B receiving the advertisement AW, but not advertising BAW to C → don't want to help C
  - No standard governing how to advertise → need **peering agreement** (<u>often confidential</u>)