

# **Lecture 7**

## The Network Layer Control Plane and ICMP



# Subjects of today:

- The Core of the Control Plane
- Intra Autonomous System Routing
- Inter Autonomous System Routing
- Software Defined Networking
- Internet Control Message Protocol

# 8.1 The Core of the Control Plane

# Network-layer functions

*Recall: two network-layer functions:*

- **Forwarding:** move packets from router's input to appropriate router output  Data plane
- **Routing:** determine route taken by packets from source to destination  Control plane

*Two approaches to structuring network control plane:*

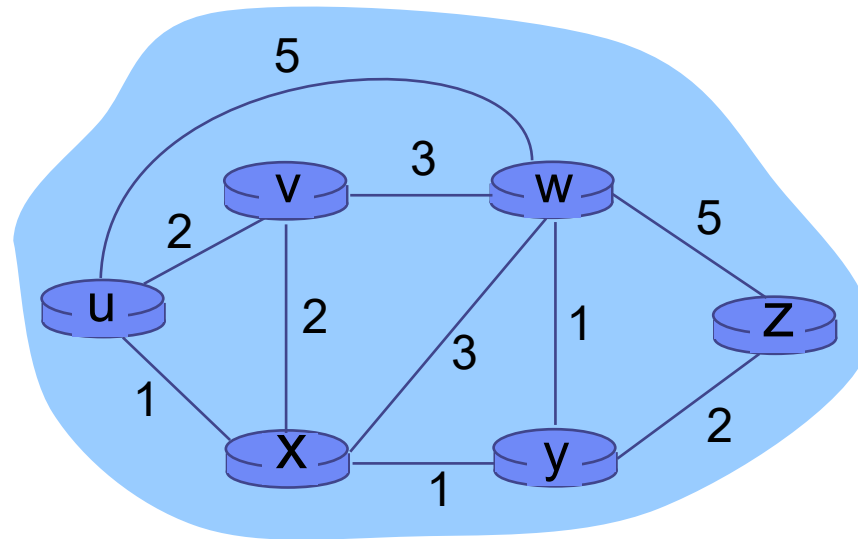
1. per-router control (traditional)
2. logically centralized control (software defined networking)

# Routing protocols

**Goal:** Determine *good paths* from sending hosts to receiving host, through network of routers

- *Path*: Sequence of routers - that packets will traverse in going from given initial source host to given final destination host
- "*Good*": Least *cost*,
- Cost could always be 1, or inversely related to bandwidth, or related to congestion. But, also fees, delay, quality and politics.

# Graph Abstraction of the Network



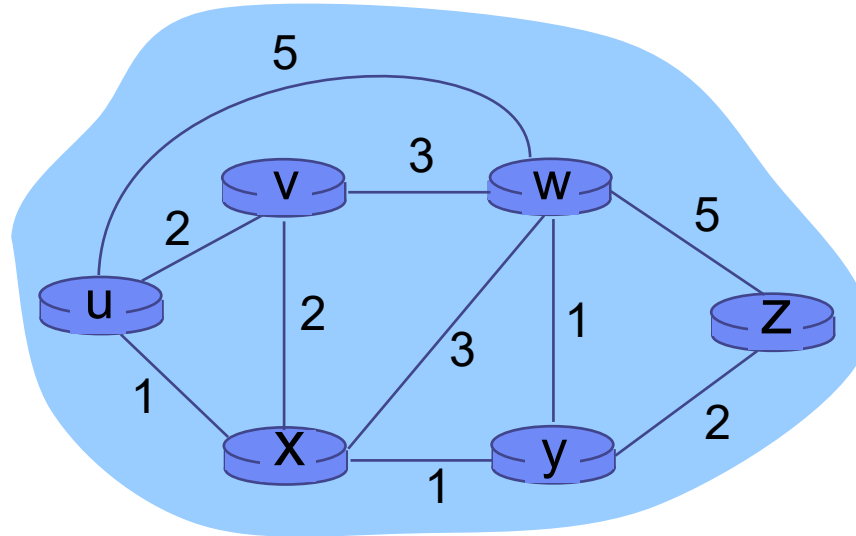
Graph:  $G = (N, E)$

$N$  = set of routers =  $\{ u, v, w, x, y, z \}$

$E$  = set of links =  $\{ (u,v), (u,x), (v,x), (v,w), (x,w), (x,y), (w,y), (w,z), (y,z) \}$

# Graph Abstraction: Costs

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



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*

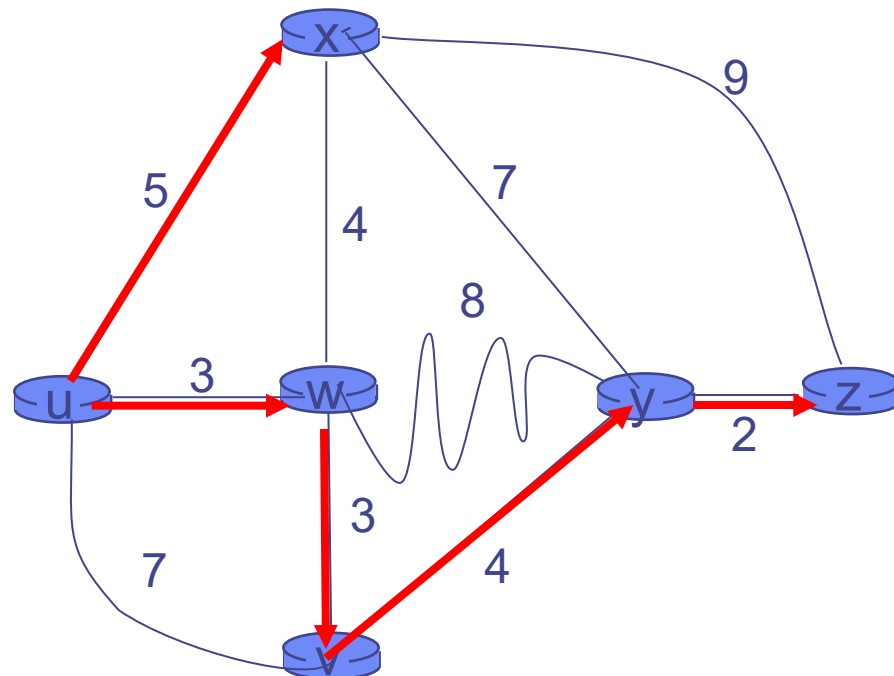
- net topology, link costs known to all nodes
  - accomplished via "link state broadcast"
  - all nodes have same info
- computes least cost paths from one node ("source") to all other nodes
  - gives *forwarding table* for that node
- iterative: after k iterations, know least cost path to k dest.'s

## *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

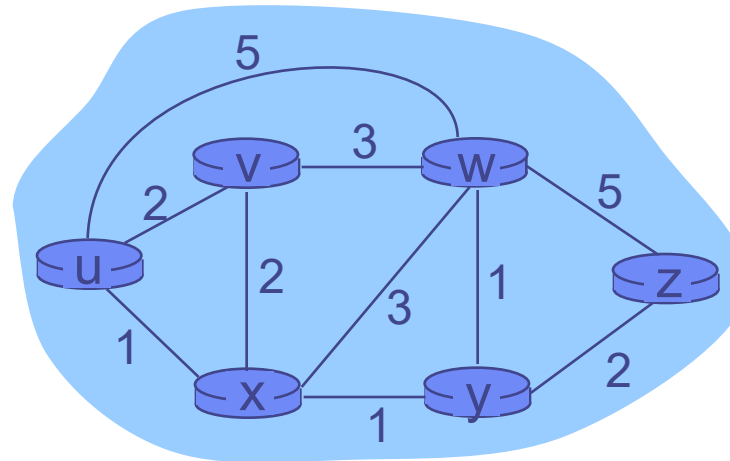
# Dijkstra's algorithm: Example

Step	N'	D( <b>v</b> ) p(v)	D( <b>w</b> ) p(w)	D( <b>x</b> ) p(x)	D( <b>y</b> ) p(y)	D( <b>z</b> ) 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					



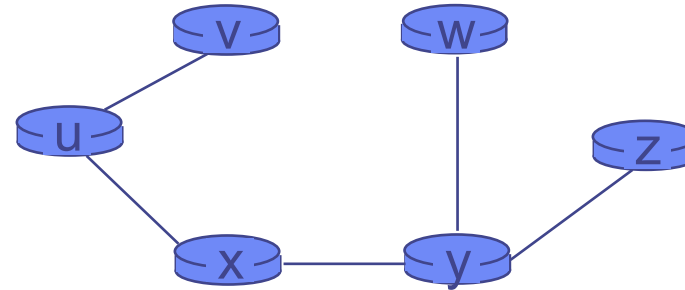
# Dijkstra's algorithm: another example

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)

# A Distance Vector Routing Algorithm

Bellman-Ford equation (dynamic programming)

let

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

then

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

min taken over all neighbors  $v$  of  $x$

cost to neighbor  $v$

cost from neighbor  $v$  to destination  $y$

# Bellman-Ford Example

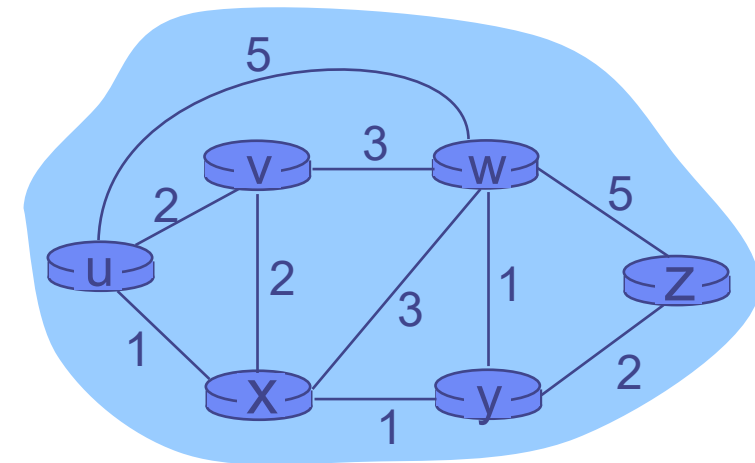
$$d_v(z) = 5, d_x(z) = 3, d_w(z) = 3$$

B-F equation says:

$$d_u(z) = \min \{ c(u,v) + d_v(z), \\ c(u,x) + d_x(z), \\ c(u,w) + d_w(z) \}$$

$$= \min \{ 2 + 5, 1 + 3, 5 + 3 \}$$

$$= 4$$



# Distance Vector Algorithm Strategy

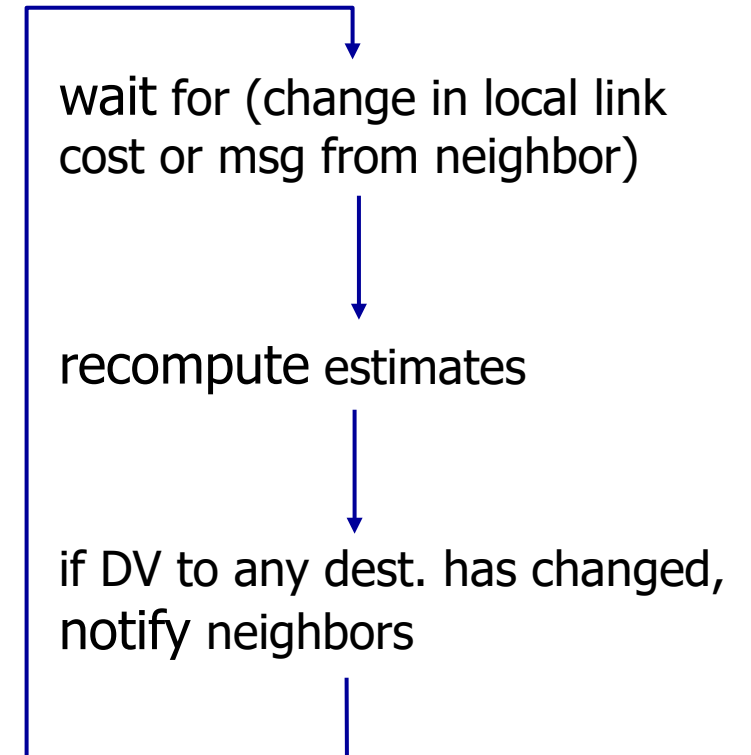
**Iterative, asynchronous:** each local iteration caused by:

- local link cost change
- DV update message from neighbor

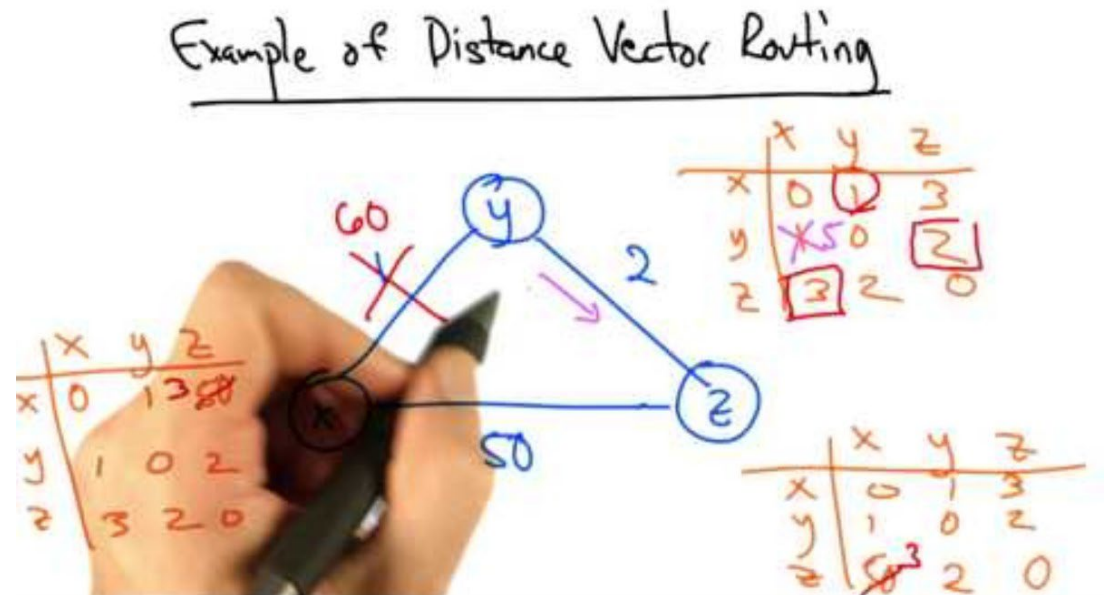
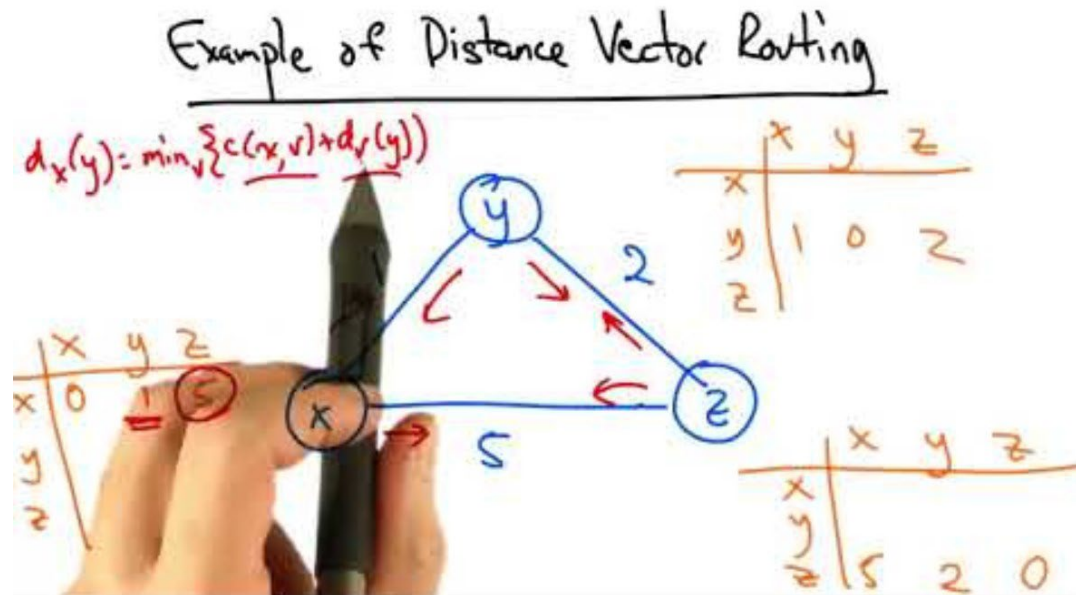
**Distributed:**

- each node notifies neighbors only when its DV changes
  - neighbors then notify their neighbors ONLY if necessary

For each node:

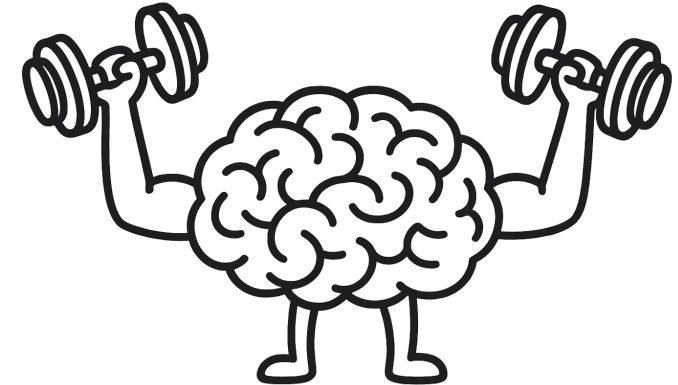


# Distance Vector Algorithm Example 2



# Routing algorithm: Exercises

1. Go to: [http://gaia.cs.umass.edu/kurose\\_ross/interactive/](http://gaia.cs.umass.edu/kurose_ross/interactive/)
2. Complete the exercise called - Dijkstra's Link State Algorithm
3. Complete the exercise called - Bellman Ford Distance Vector Algorithm



# 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 as 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 (black-holing)
- each node's table used by others
  - error propagate thru network

## 8.2 Intra Autonomous System Routing

# Making Routing Scalable

Our routing study thus far - idealized

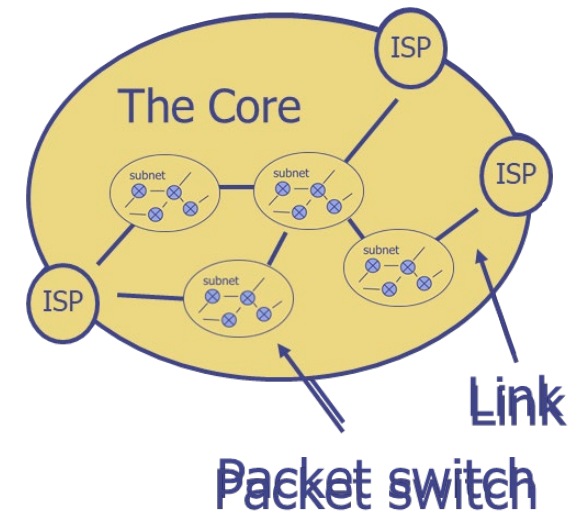
- all routers identical
- network “flat”  
... *not* true in practice

**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

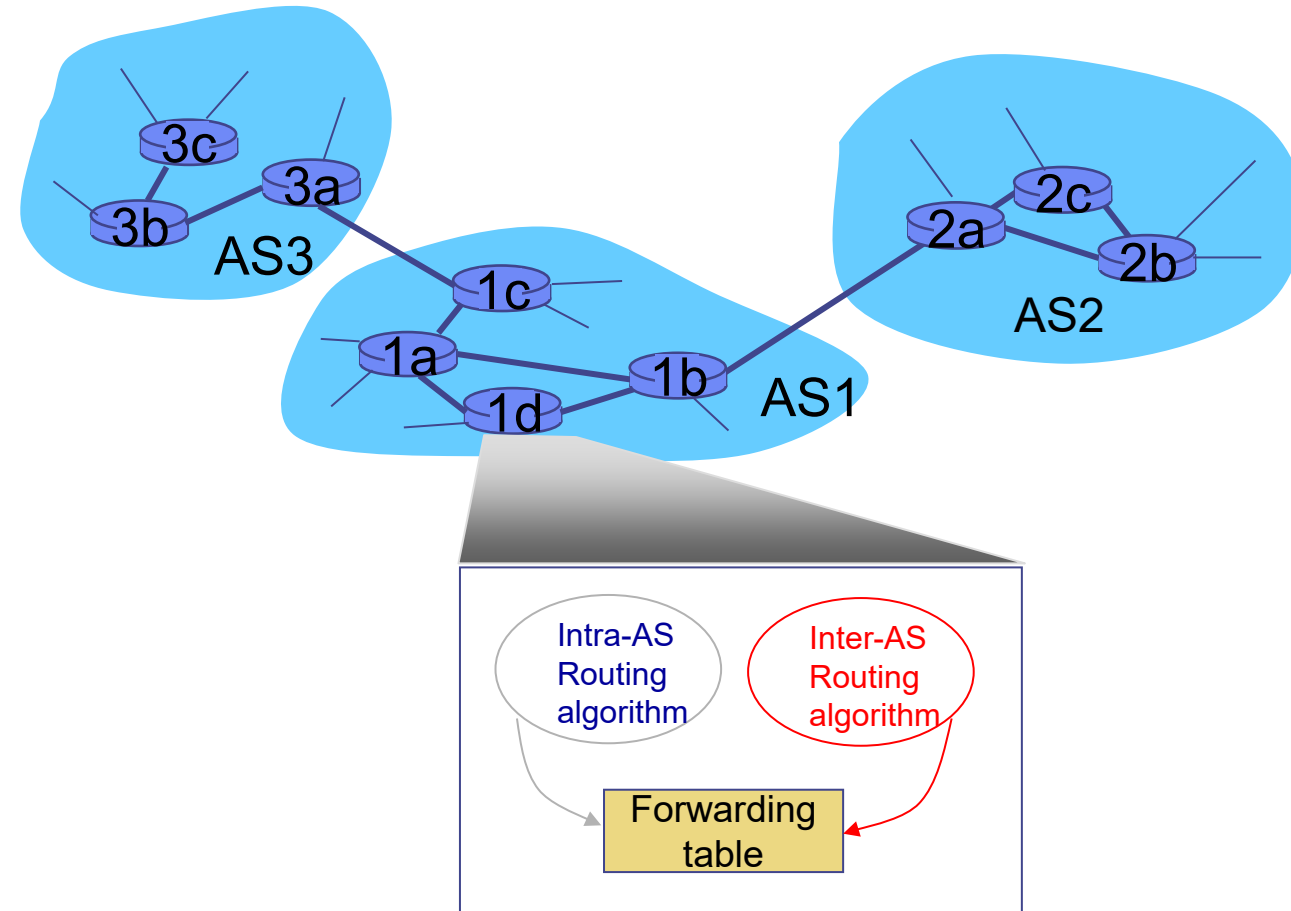
- Routing among hosts, routers in same AS (“network”)
- All routers in AS must run *same* intra-domain protocol
- Routers in *different* AS can run *different* intra-domain routing protocol
- Gateway router: at “edge” of its own AS, has link(s) to router(s) in other AS'es

## Inter-AS routing

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

# Interconnected AS's

- 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

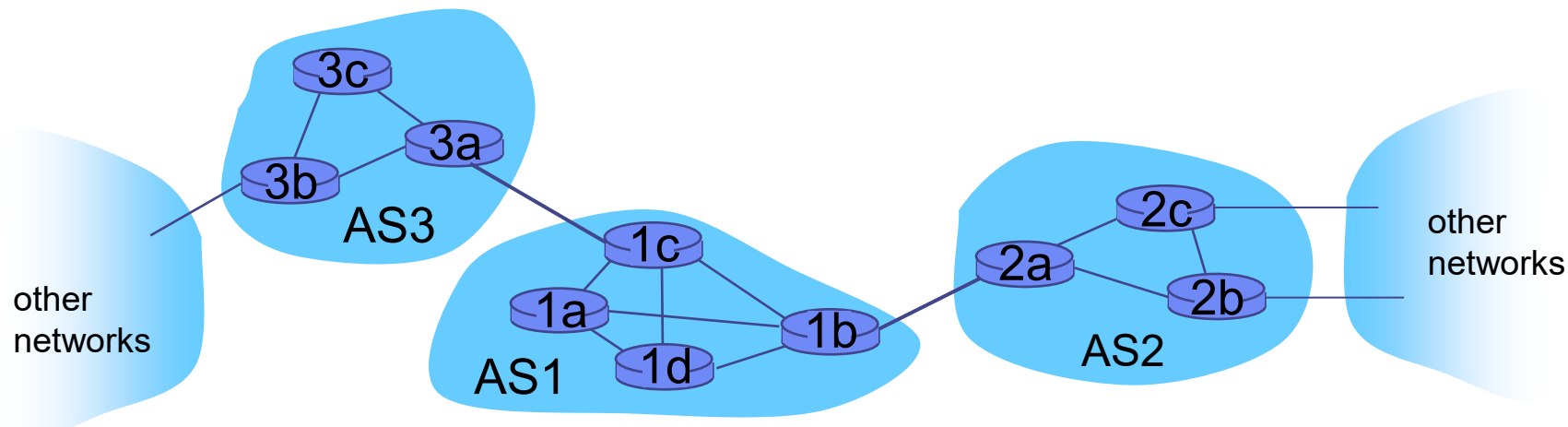


# Inter-AS tasks

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

## AS1 must:

1. Learn which destinations are reachable through AS2, which through AS3
2. Propagate this reachability info to all routers in AS1



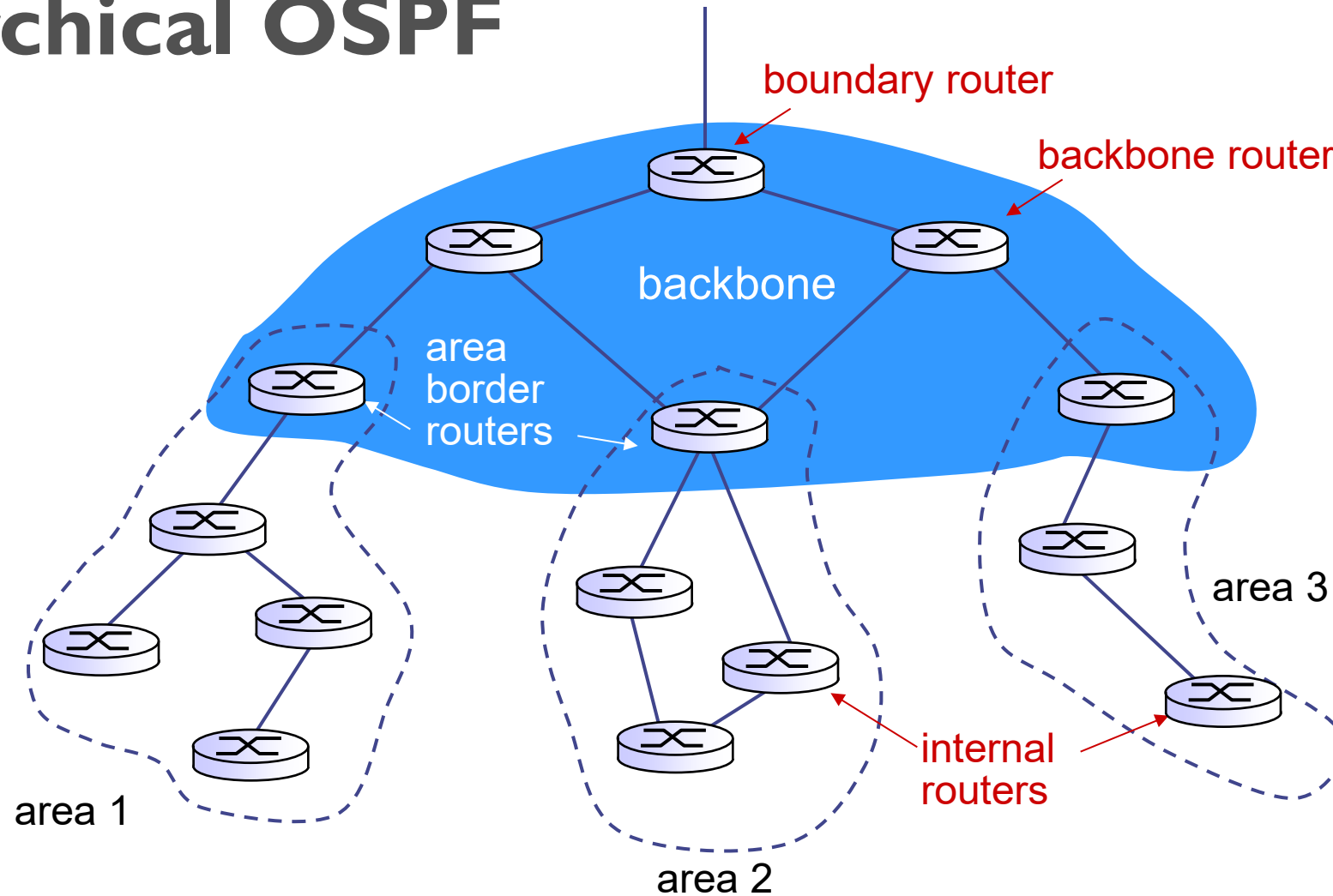
# Intra-AS Routing

- Also known as *interior gateway protocols (IGP)*
- Most common intra-AS routing protocols:
  - RIP: Routing Information Protocol [RFC 1723]
    - classic DV: DVs exchanged every 30 secs
    - no longer widely used
  - EIGRP: Enhanced Interior Gateway Routing Protocol
    - DV based
    - formerly Cisco-proprietary for decades (became open in 2013 [RFC 7868])
  - OSPF: Open Shortest Path First [RFC 2328]
    - link-state routing
    - IS-IS protocol (ISO standard, not RFC standard) essentially same as OSPF

# OSPF (Open Shortest Path First)

- Open: publicly available
- Uses link-state algorithm
  - link state packet dissemination
  - topology map at each node
  - route computation using Dijkstra's algorithm
- Router floods OSPF link-state advertisements to all other routers in *entire* AS
  - carried in OSPF messages directly over IP (rather than TCP or UDP)
  - link state: for each attached link

# Hierarchical OSPF



## 8.3 Inter Autonomous System Routing

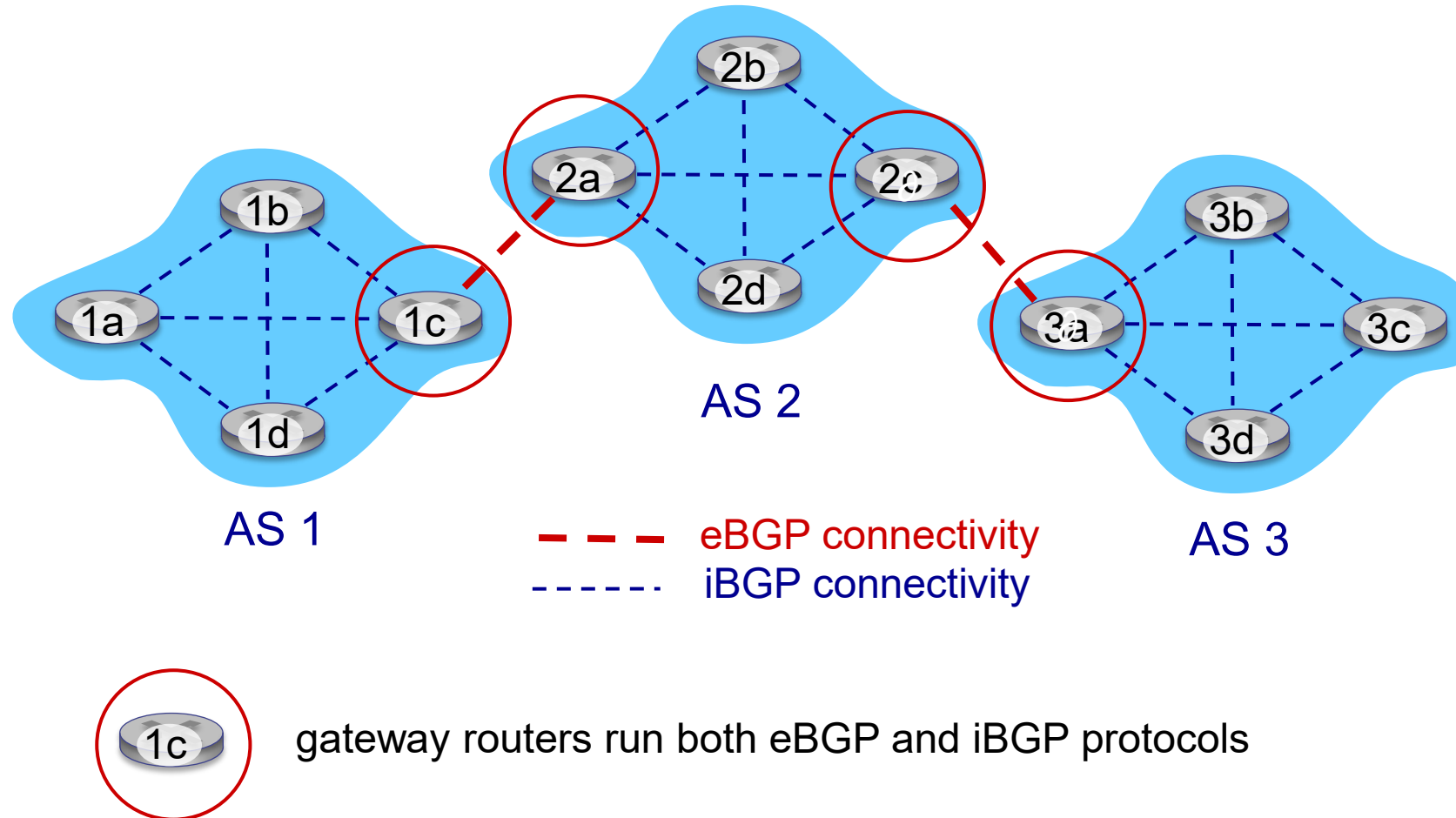
# Internet inter-AS routing: BGP

## **BGP (Border Gateway Protocol):**

The de facto inter-domain routing protocol “glue that holds the Internet together”

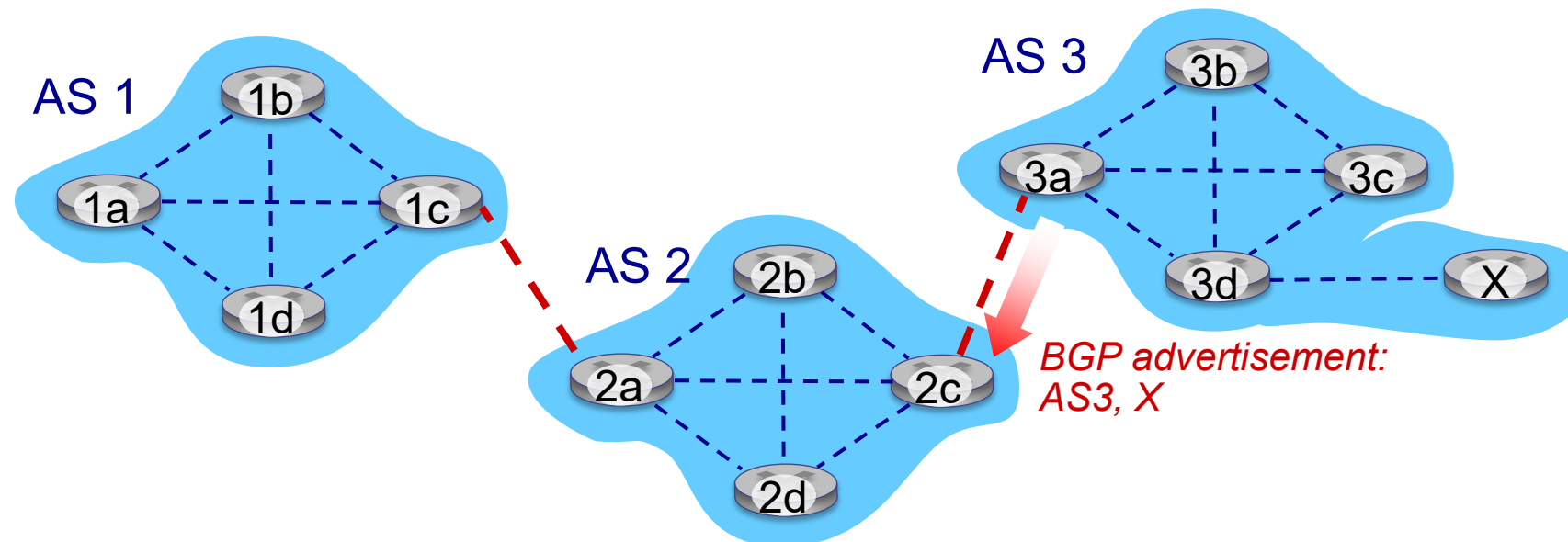
- BGP provides each AS a means to:
  - Obtain subnet reachability information from neighboring As'es (**eBGP**)
  - Determine routes to other networks based on reachability information and **policy**
  - Propagate reachability information to all AS-internal routers (**iBGP**)
- Allows subnet to advertise its existence to rest of Internet: **“I am here”**
  - If it wants to!

# eBGP, iBGP connections



# BGP basics

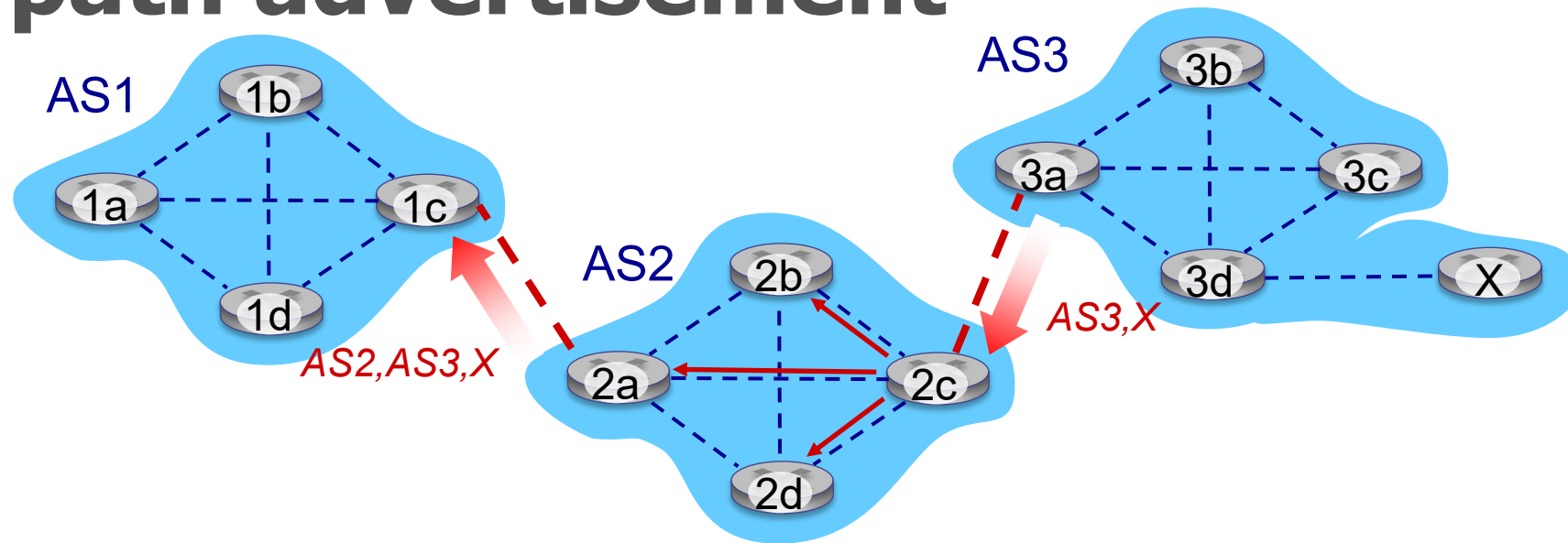
- **BGP session:** two BGP routers (“peers”) exchange BGP messages over TCP:
  - advertising *paths* to different destination network prefixes (BGP is a “path vector” protocol)
- When AS3 gateway router 3a advertises path AS3,X to AS2 gateway router 2c:
  - AS3 **promises** to AS2 it will forward datagrams towards X



# Path Attributes and BGP routes

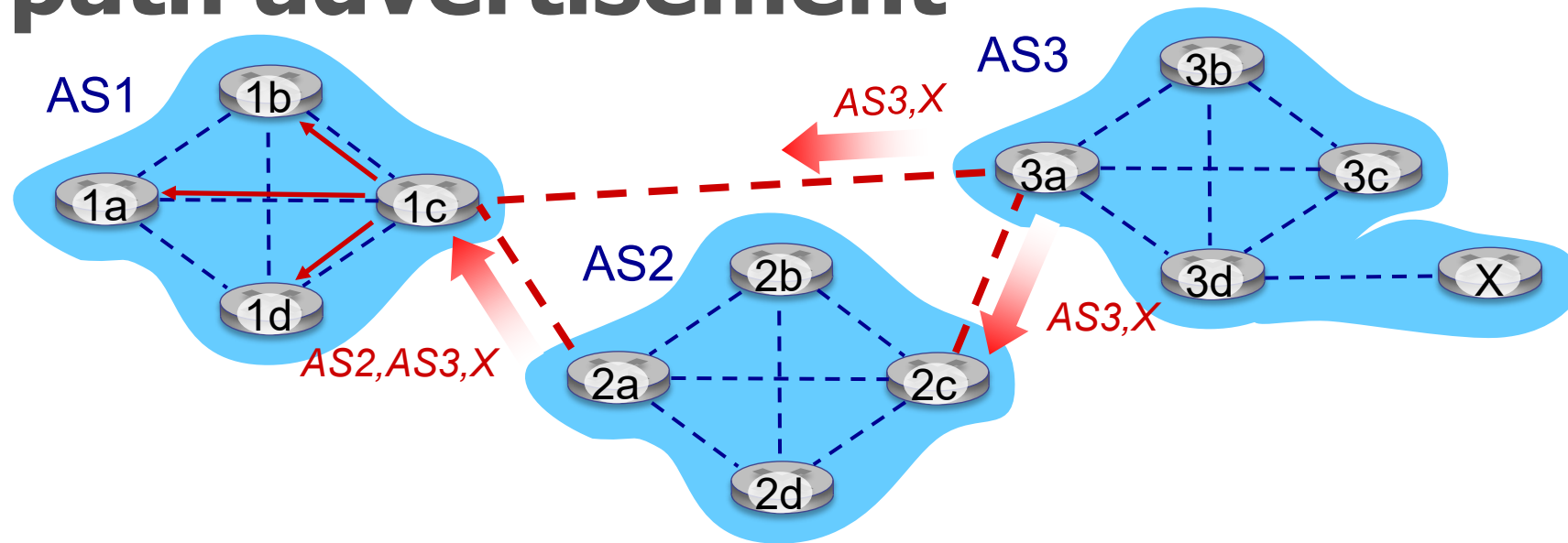
- BGP advertised attributes: prefix + attributes = "route"
  - Prefix: destination
  - Two important attributes:
    - **AS-PATH:** list of AS's through which prefix advertisement has passed
    - **NEXT-HOP:** indicates specific internal-AS router to next-hop AS
- **Policy-based routing:**
  - Gateway receiving route advertisement uses **import policy** to accept/decline path (e.g., never route through AS Y).
  - AS policy also determines whether to **advertise** path to other neighboring AS'es

# BGP path advertisement



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

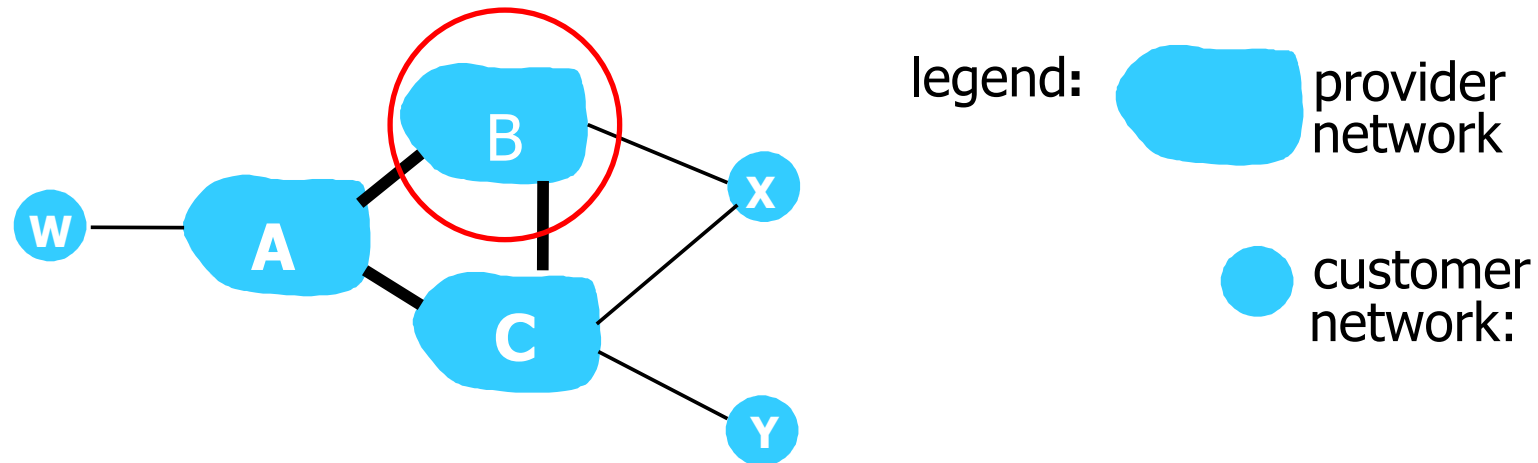
# BGP path advertisement



gateway router may learn about **multiple** paths to destination:

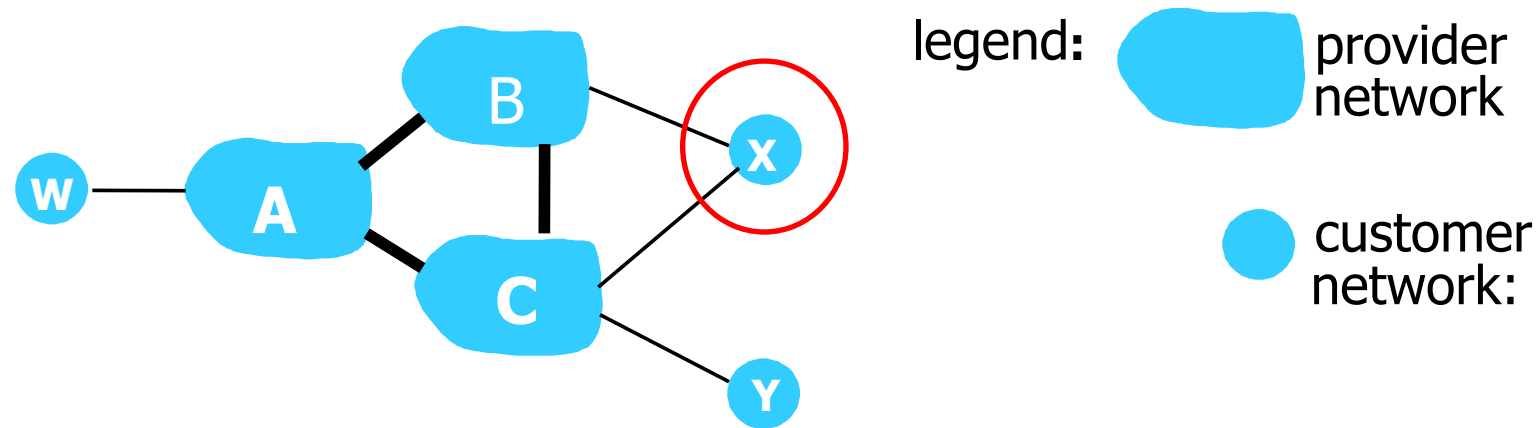
- AS1 gateway router 1c learns path **AS2,AS3,X** from 2a
- AS1 gateway router 1c learns path **AS3,X** from 3a
- Based on policy, AS1 gateway router 1c chooses path **AS3,X**, and *advertises path within AS1 via iBGP*

# BGP: achieving policy via advertisements



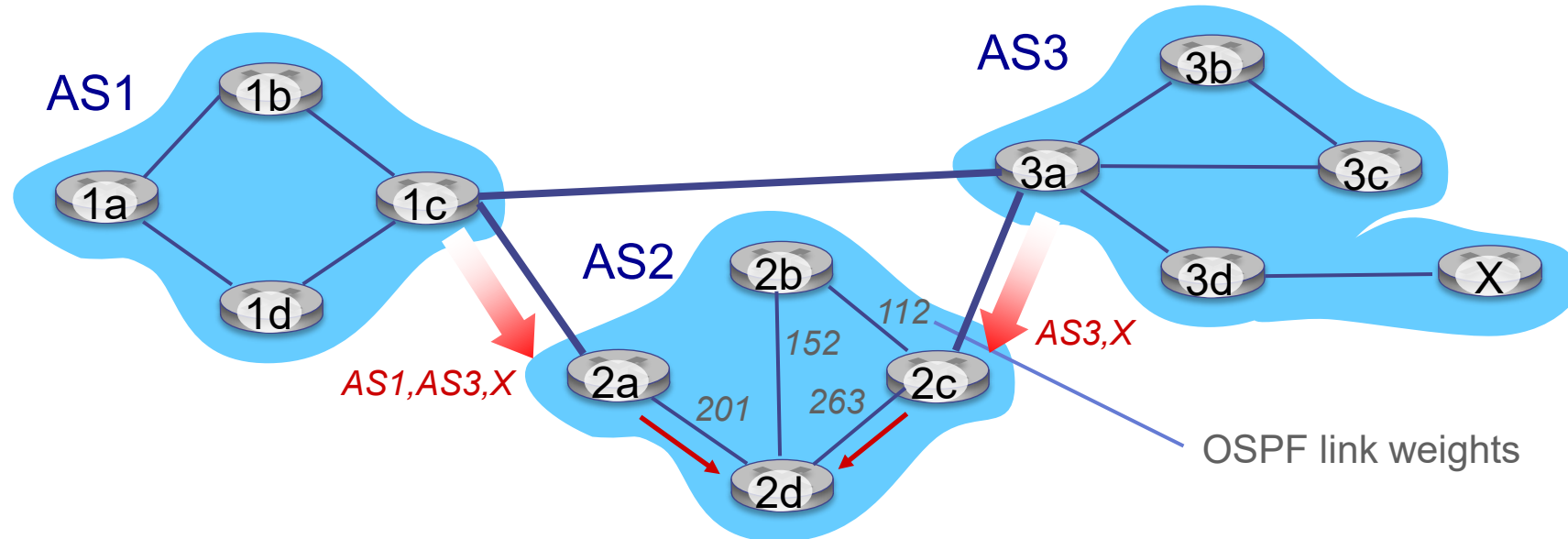
An ISP (B) may only want to route traffic to/from its customer networks (does not want to carry transit traffic between other ISPs)

# BGP: achieving policy via advertisements



Customer network x is dual-homed  
A customer network may not want to route traffic between ISP

# Hot Potato Routing



- 2d learns (via iBGP) it can route to X via 2a or 2c
- Choose local gateway that has least intra-domain cost
  - (e.g., 2d chooses 2a, even though more AS hops to X): don't worry about inter-domain cost!

# 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

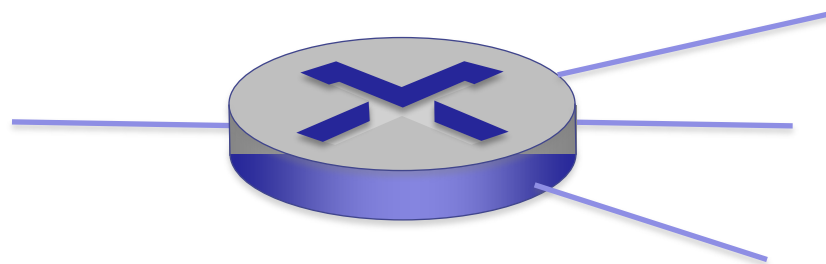
## **Performance:**

- Intra-AS: can focus on performance
- Inter-AS: policy may dominate over performance

## 8.4 Software Defined Networking

# OpenFlow data plane abstraction

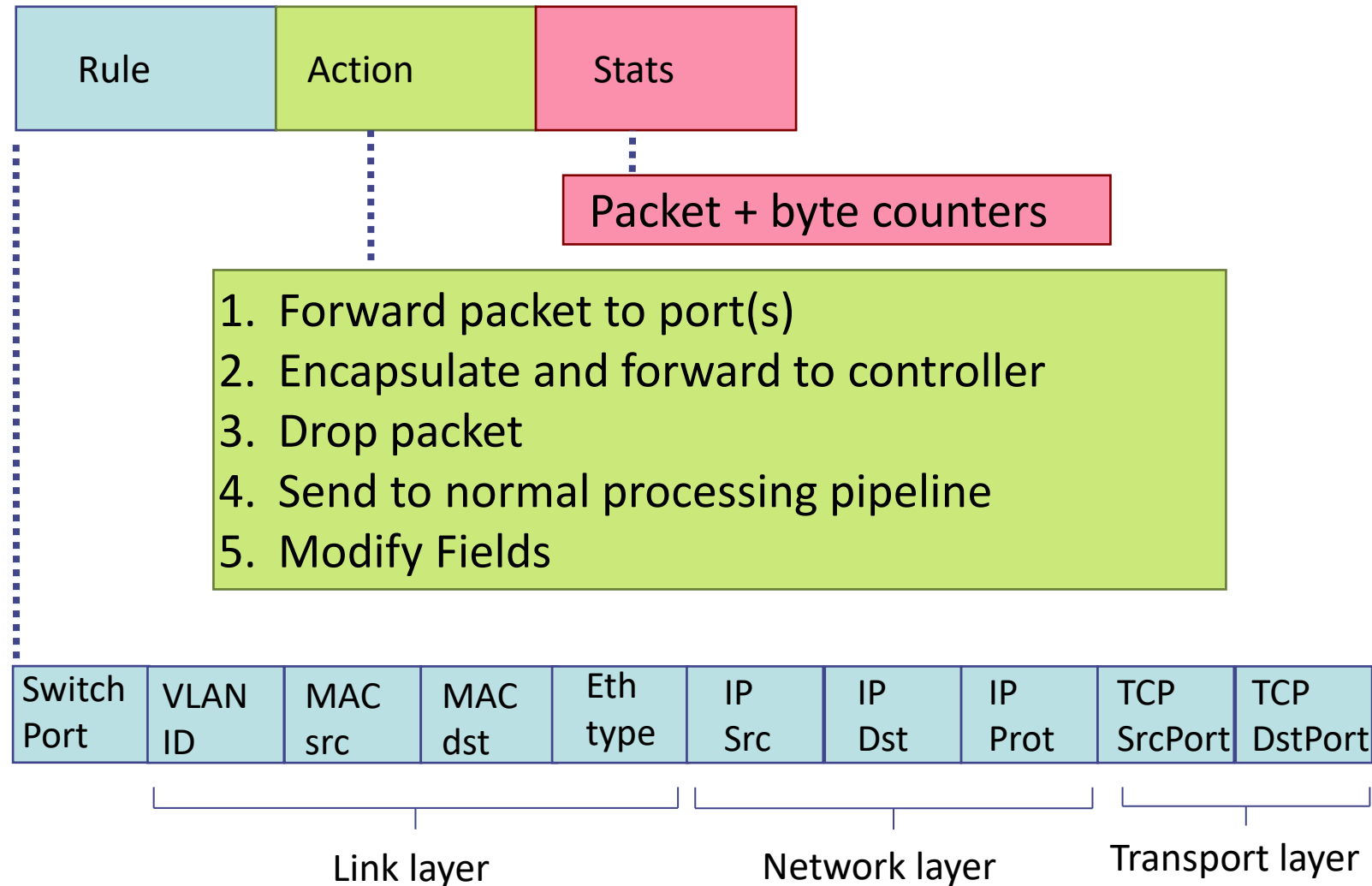
- *flow*: defined by header fields
- generalized forwarding: simple packet-handling rules
  - *Pattern*: match values in packet header fields
  - *Actions: for matched packet*: drop, forward, modify, matched packet or send matched packet to controller
  - *Priority*: disambiguate overlapping patterns
  - *Counters*: #bytes and #packets



\* : wildcard

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

# OpenFlow: Flow Table Entries



# Software defined networking (SDN)

4. programmable control applications

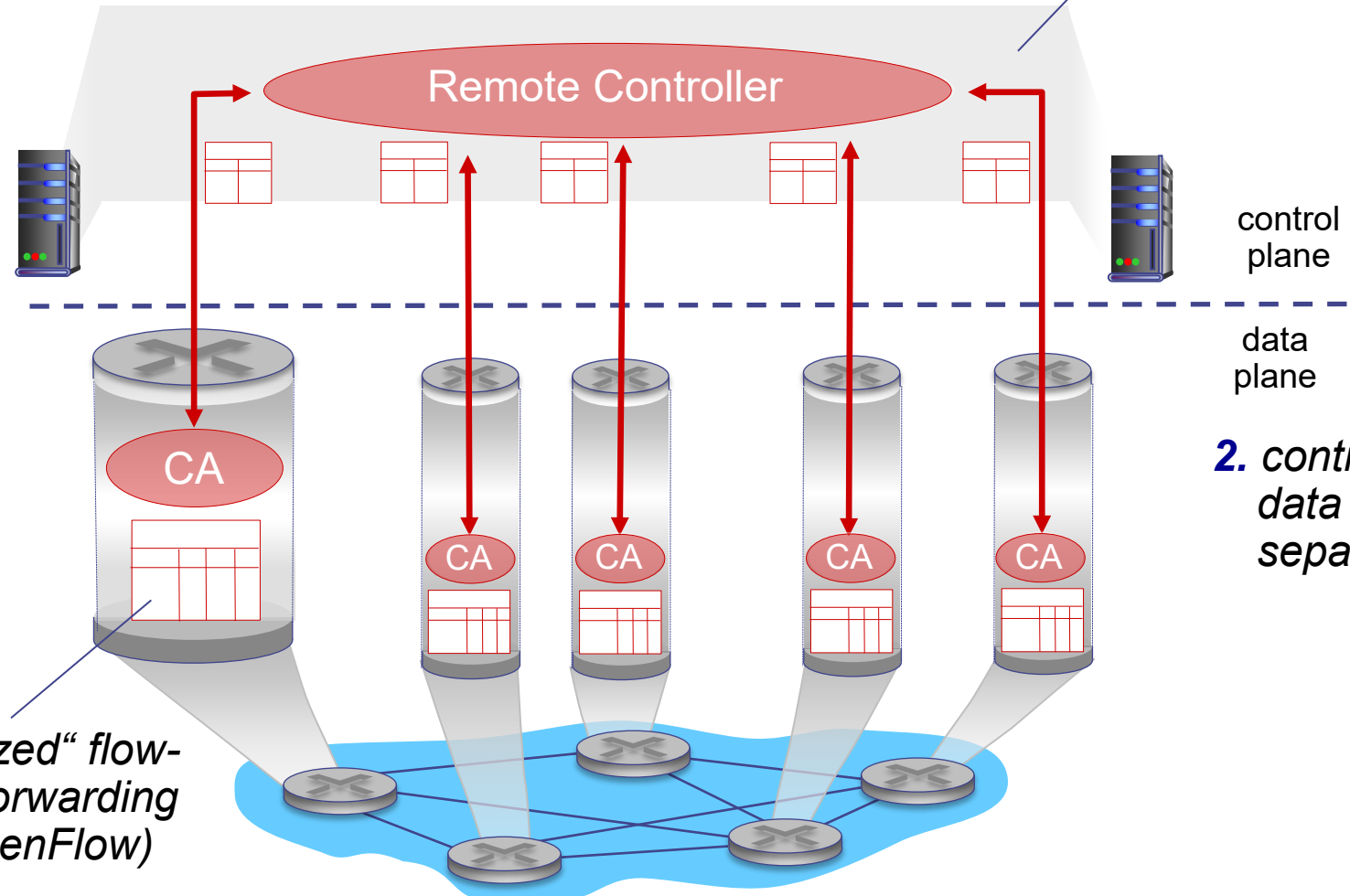
routing

access control

...

load balance

3. control plane functions external to data-plane switches

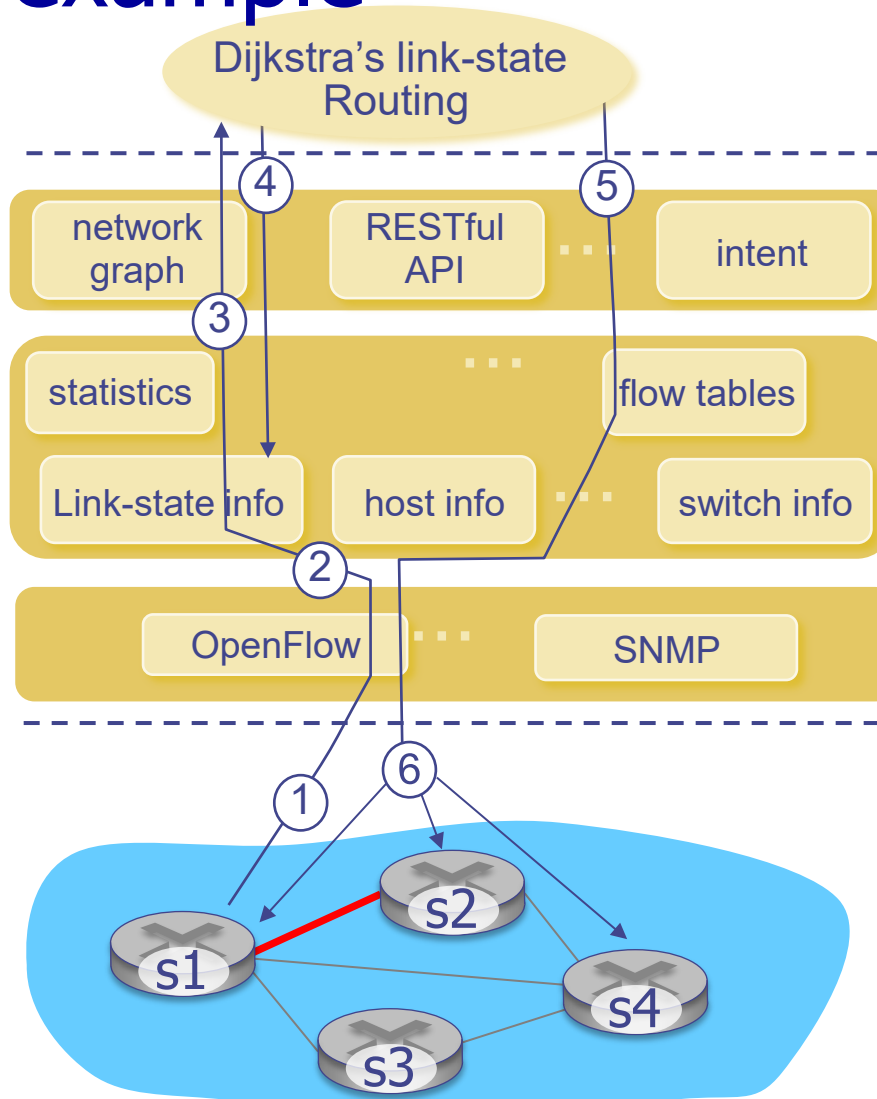


2. control, data plane separation

1: generalized "flow-based" forwarding (e.g., OpenFlow)

# SDN: control/data plane interaction

## example



- ① S1, experiencing link failure using OpenFlow port status message to notify controller
- ② SDN controller receives OpenFlow message, updates link status info
- ③ Dijkstra's routing algorithm application has previously registered to be called when ever link status changes. It is called.
- ④ Dijkstra's routing algorithm access network graph info, link state info in controller, computes new routes
- ⑤ link state routing app interacts with flow-table-computation component in SDN controller, which computes new flow tables needed
- ⑥ Controller uses OpenFlow to install new tables in switches that need updating

# 8.5 Internet Control Message Protocol

# ICMP: Internet Control Message Protocol

- Used by hosts & routers to communicate network-level information
  - Error reporting: unreachable host, network, port, protocol
  - Echo request/reply (used by ping)
- ICMP msgs carried in IP datagrams
  - Protocol number 1
- ICMP message:
  - type, code, checksum
  - For error messages it also includes some of the problem causing IP datagram (header + 8 bytes of payload)

<u>Type</u>	<u>Code</u>	<u>description</u>
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

# Traceroute and ICMP

- Source sends series of UDP segments to destination
  - first set has TTL = 1
  - second set has TTL=2, etc.
  - unlikely port number
- When datagram in  $n^{\text{th}}$  set arrives to  $n^{\text{th}}$  router:
  - router discards datagram and sends source ICMP message (type 11, code 0)
  - ICMP message include name of router & IP address

- When ICMP message 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

