Networks

Control Plane - Introduction

Adopted from material in “Computer Networking: A Top Down Approach” by Kurose and Ross and slides developed by William Conner

Control Plane - Introduction

Section 5.1

Review: Network Layer Functions

- Forwarding (data plane): move packets from router's input ports to appropriate output ports

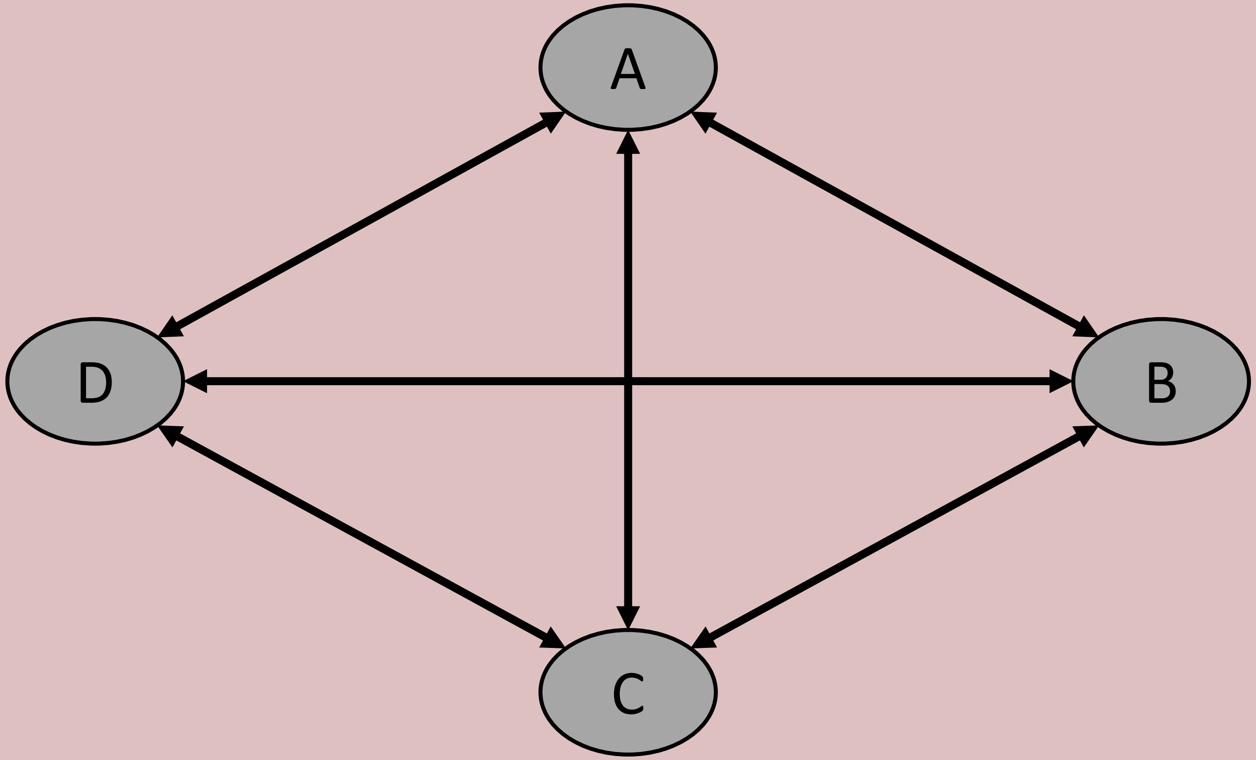
- Routing (control plane): determine route taken by packets from source to destination

Control Plane Approaches

- Distributed per-router control (traditional routing)
  - Link state (global view)
  - Distance vector (decentralized view)

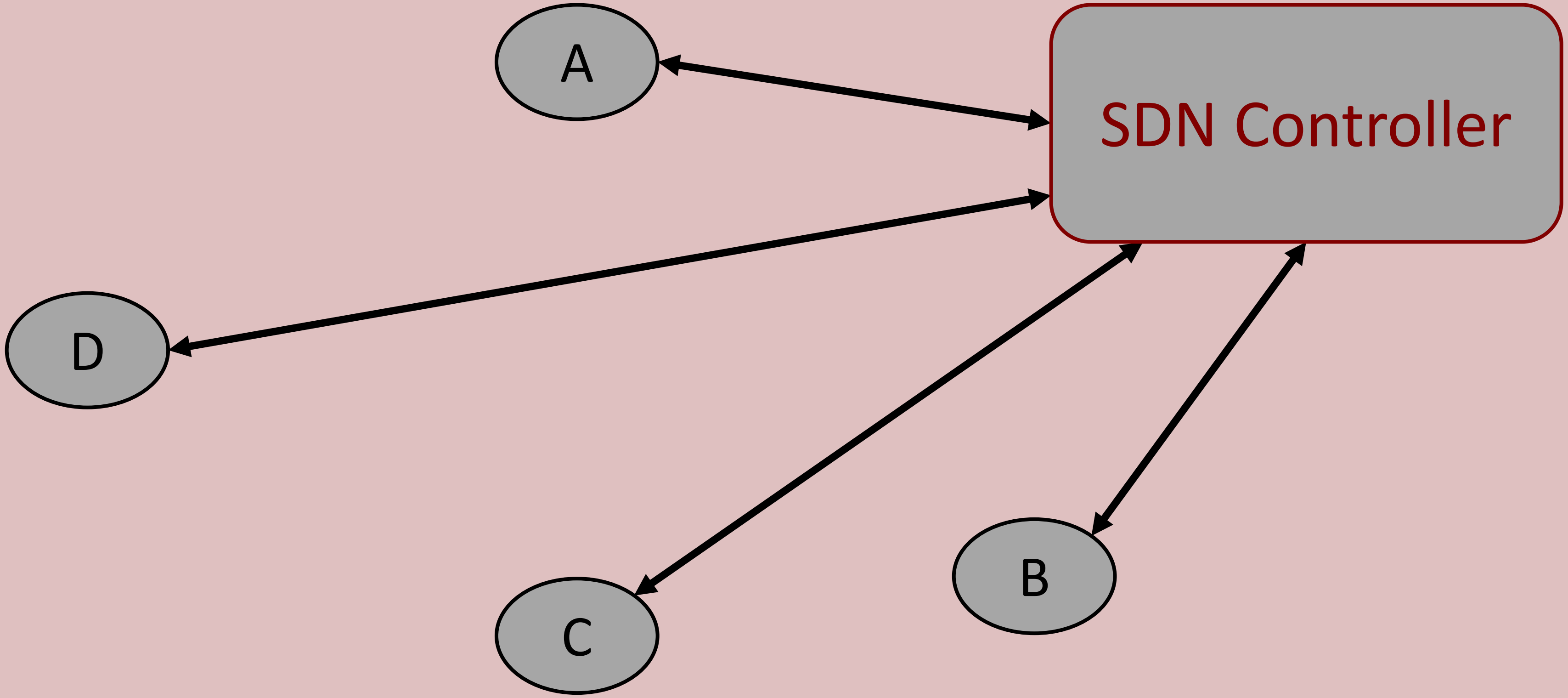
- Logically centralized control (software defined networking)

Per-Router Control Plane



Each router participates in traditional *routing algorithm* to compute its forwarding table

Logically Centralized Control Plane



Remote SDN controller computes and distributes forwarding tables.
Routers communicate to controller via controller agents.

Vacation Analogy

- Group of friends collectively figure out daily sightseeing schedule during breakfast (traditional routing algorithm)

- Travel agent emails daily sightseeing schedule to group each morning (software-defined networking)

Thank You!

Networks

Routing Algorithms

Adopted from material in “Computer Networking: A Top Down Approach” by Kurose and Ross and slides developed by William Conner

Routing Algorithms

Section 5.2

Routing Protocols

- Objective: determine “good” paths (i.e., routes) from source to destination

  - Path: sequence of routers that packets will traverse from source host to destination host

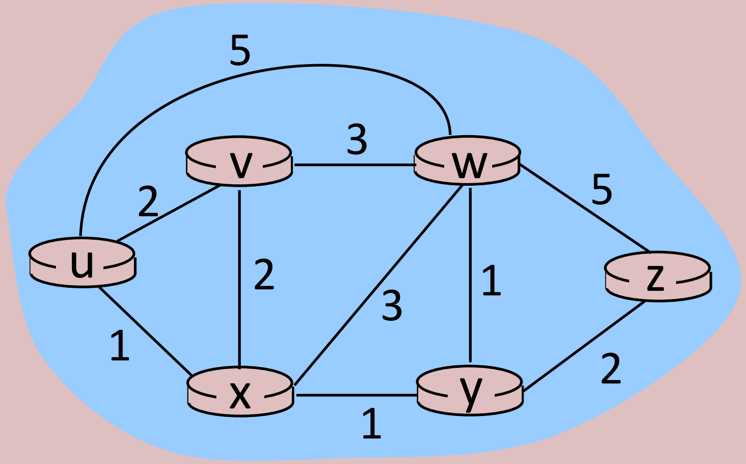
  - “Good”: determined by some metric (e.g., least cost, fastest, least congested)

Network Graph Abstraction

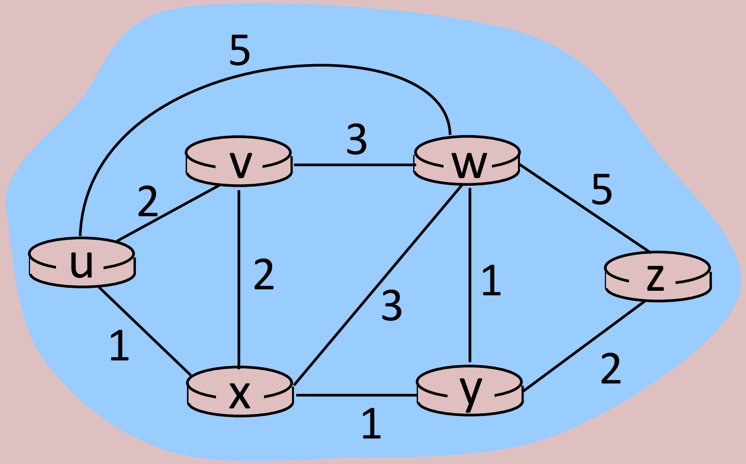
Graph G = (N, E)

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

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



Network Graph Abstraction



$c(x, x')$ = cost of link (x, x')

$c(w, z) = 5$

Cost could always be 1, or inversely related to bandwidth, or 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 the least cost path

Routing Algorithm Classification

Global View	• All routers have complete topology and link cost information • Exchange information via broadcast and/or flooding • *Link state* algorithms
Decentralized View	• Routers only know immediate neighbors and link cost to neighbors • Exchange information with neighbors • Iterative computation • *Distance vector* algorithms

Static	• Routes change slowly over time • Might involve manual configuration
Dynamic	• Routes change relatively fast • Triggered by periodic updates • Triggered by link cost changes

Dijkstra's Algorithm

- Link state routing algorithm
- Network topology and link costs known by all nodes via link state broadcasts
- Compute least cost paths from single source node to all other nodes
- After *k* iterations, know least cost path to *k* destinations

Dijkstra’s Algorithm

c(x,y)	Link cost from node x to y; set to ∞ if not direct neighbors
D(v)	current value of cost of path from single source to destination v
p(v)	predecessor node along path from source to v
N’	set of nodes whose least cost path is definitively known

Dijkstra's Algorithm - Intialiazation

N' = {u}

for all nodes v $\in$ N

    if v adjacent to u

        D(v) = c(u, v) // Link cost between u and v

    else

        D(v) = ∞

Dijkstra's Algorithm – Main Loop

While some nodes are not in N':

Find w not in N' such that D(w) is a minimum

Add w to N'

For all v adjacent to w and not in N'

// Update cost if known shortest path to w plus cost from w to v is shorter than current cost

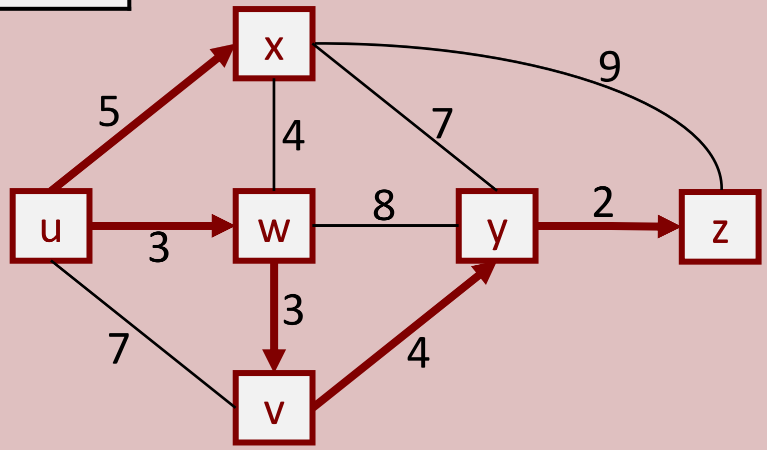
D(v) = min(D(v), D(w) + c(w, v))

Also, if new path, p(v) = w

Example: Dijkstra's Algorithm

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, v	14, x
4	uwxvy					12, y
5	uwxvyz					

Shortest path tree from source to destinations can be constructed by tracing predecessor nodes



Dijkstra’s Algorithm Complexity

N = set of nodes

n = |N|, or the number of nodes in N

Checks all nodes, w, not in N’ on each iteration

n(n+1)/2 comparisons: $O(n^2)$

More efficient implementations are possible: O(n log n)

Bellman-Ford Algorithm

- Distance vector routing algorithm
- Each router knows distance to neighbors, but *has no global view of network topology*
- Neighbors exchange distance vectors containing their best known distances to each destination

Bellman-Ford Equation

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

cost from neighbor v to destination y

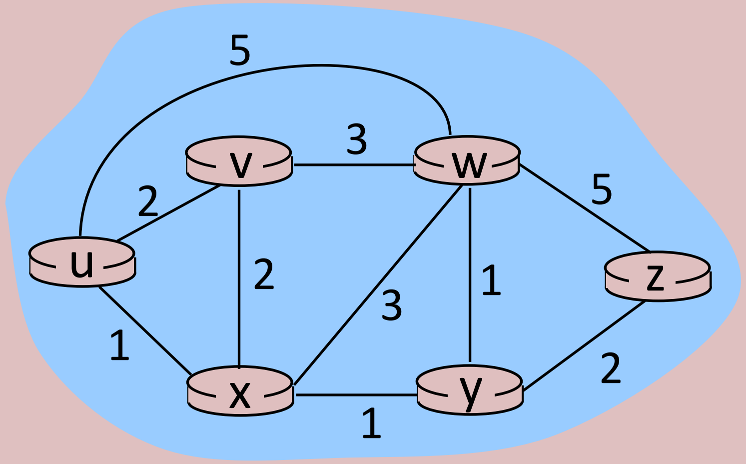
cost from x to neighbor v

minimum taken over all neighbors v of x

cost of least-cost path from x to y

Example: Bellman-Ford

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



Bellman-Ford equation says:

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

What's is u's next hop to z?

The node achieving the minimum is the next hop in the shortest path, used in the forwarding table

Distance Vector Algorithm

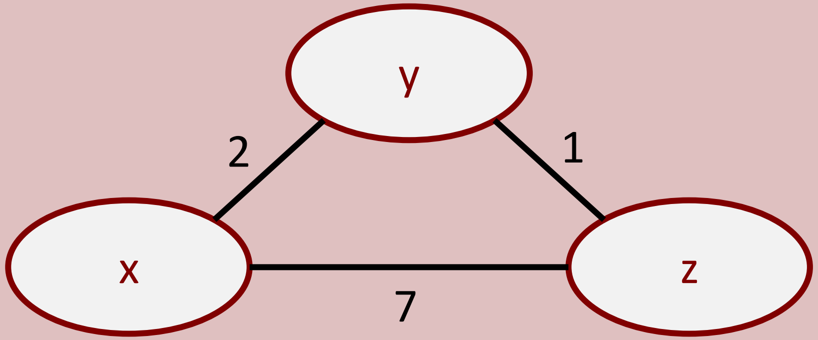
Wait for local link cost change or distance vector update from neighbor

Recompute local distance vector using Bellman-Ford equation

Notify neighbors of updated distance vector if any entries change

Repeat

Example: Distance Vector



t=0 (x only knows local links)

$DV_x = (0, 2, 7)$

$DV_y = (\infty, \infty, \infty)$

$DV_z = (\infty, \infty, \infty)$

t=1 (x exchanges with y)

$DV_x = (0, 2, 3)$

$DV_y = (2, 0, 1)$

$DV_z = (\infty, \infty, \infty)$

Better distance found to z, so notify neighbors of latest DV_x

t=2 (x exchanges with z)

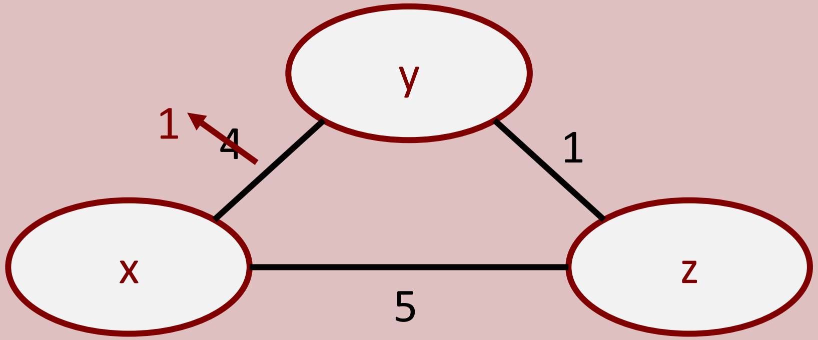
$DV_x = (0, 2, 3)$

$Dv_y = (2, 0, 1)$

$DV_z = (7, 1, 0)$

No change, so do not send any updates

Link Cost Improvements



t=0

$DV_x = (0, 4, 5)$

$DV_y = (4, 0, 1)$

$DV_z = (5, 1, 0)$

t=1 (link cost improvement)

$DV_x = (0, 1, 2)$

$DV_y = (1, 0, 1)$

$DV_z = (5, 1, 0)$

x and y update distance vectors, then notify neighbors

t=2 (exchange with z)

$DV_x = (0, 1, 2)$

$Dv_y = (1, 0, 1)$

$DV_z = (2, 1, 0)$

z updates distance vector and notifies neighbors, but no further updates

Count-to-Infinity Problem

t=0

$d_y(x)$ = min(4, 1+5) = 4

$d_z(x)$ = min(50, 1+4) = 5

t=1 (link cost increase)

$d_y(x)$ = min(60, 1+5) = 6

$d_z(x)$ = min(50, 1+6) = 7

t=2

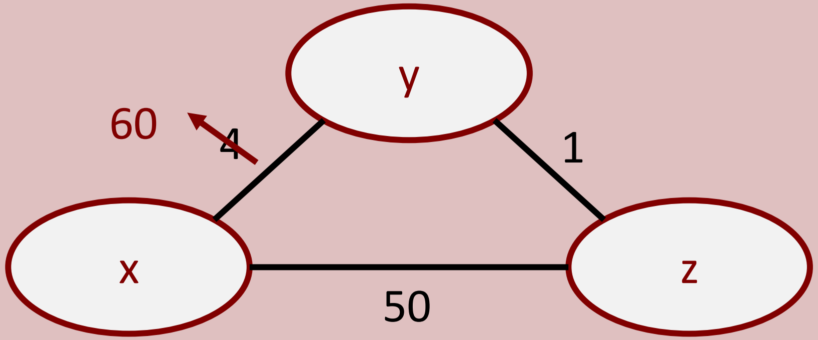
$d_y(x)$ = min(60, 1+7) = 8

$d_z(x)$ = min(50, 1+8) = 9

t=3

$d_y(x)$ = min(60, 1+9) = 10

$d_z(x)$ = min(50, 1+10) = 11



$d_y(x)$ = min{ c(y,x), c(y,z) + $d_z(x)$ }
$d_z(x)$ = min{ c(z,x), c(z,y) + $d_y(x)$ }

When will the distance vectors finally converge to stable values?

Poison Reverse

- Count-to-infinity is fundamentally caused by circular dependency between z and y

- Breaking this cycle should solve the issue

- If some node u depends on node v to reach some destination w, then u advertises *infinite distance* to w to v

Poison Reverse

t=0

$d_y(x) = \min(4, 1+\infty) = 4$

$d_z(x) = \min(50, 1+4) = 5$

t=1 (link cost increase)

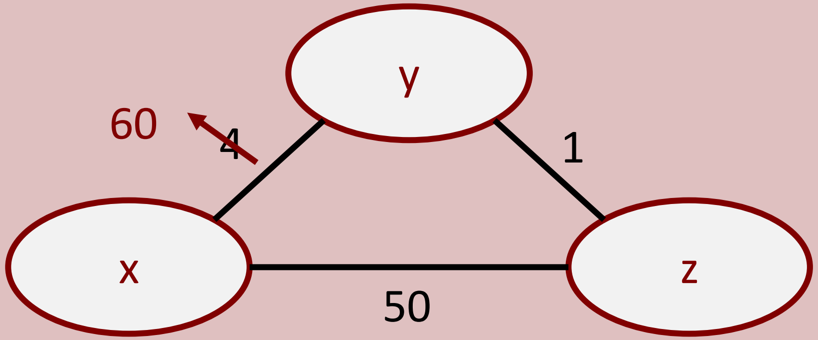
$d_y(x) = \min(60, 1+\infty) = 60$

$d_z(x) = \min(50, 1+60) = 50$

t=2

$d_y(x) = \min(60, 1+50) = 51$

$d_z(x) = \min(50, 1+\infty) = 50$



$d_y(x) = \min\{ c(y,x), c(y,z) + d_z(x) \}$
$d_z(x) = \min\{ c(z,x), c(z,y) + d_y(x) \}$

Thank You!

Networks

Intra-AS Routing

Adopted from material in “Computer Networking: A Top Down Approach” by Kurose and Ross and slides developed by William Conner

Intra-AS Routing

Section 5.3

Autonomous Systems

- Internet = network of networks
- Network admins want to control routing in their own networks
- Aggregate routers into autonomous systems (ASes) under single administrative domains

Intra-AS Routing

- Routing amongst hosts and routers within the same AS
- All routers in the same AS must run the same intra-domain protocol
- Routers in different ASes can run different intra-domain protocols
- Gateway router: “edge” of its own AS with links to routers in other ASes
- Also known as interior gateway protocols (IGP)

Intra-AS Routing

- Most common intra-AS routing protocols
  - RIP: Routing Information Protocol
  - OSPF: Open Shortest Path First
  - IGRP: Interior Gateway Routing Protocol (Cisco proprietary protocol until 2016)

Open Shortest Path First (OSPF)

- “Open”: publicly available (RFC 2328)
- Link-state algorithm
- Messages sent directly over IP
- Routers flood link-state advertisements for each attached link to all other routers in the entire AS

OSPF Features

- All messages are authenticated
- Multiple same-cost paths are allowed (only one path in RIP)
- Integrated multicast support by simply reusing shortest path tree
- Hierarchical OSPF in large domains

Hierarchical OSPF

- Two-level hierarchy: local areas and backbone
- Link-state advertisements only in local area
- Each node has detailed local area topology; only know direction (shortest path) to other areas

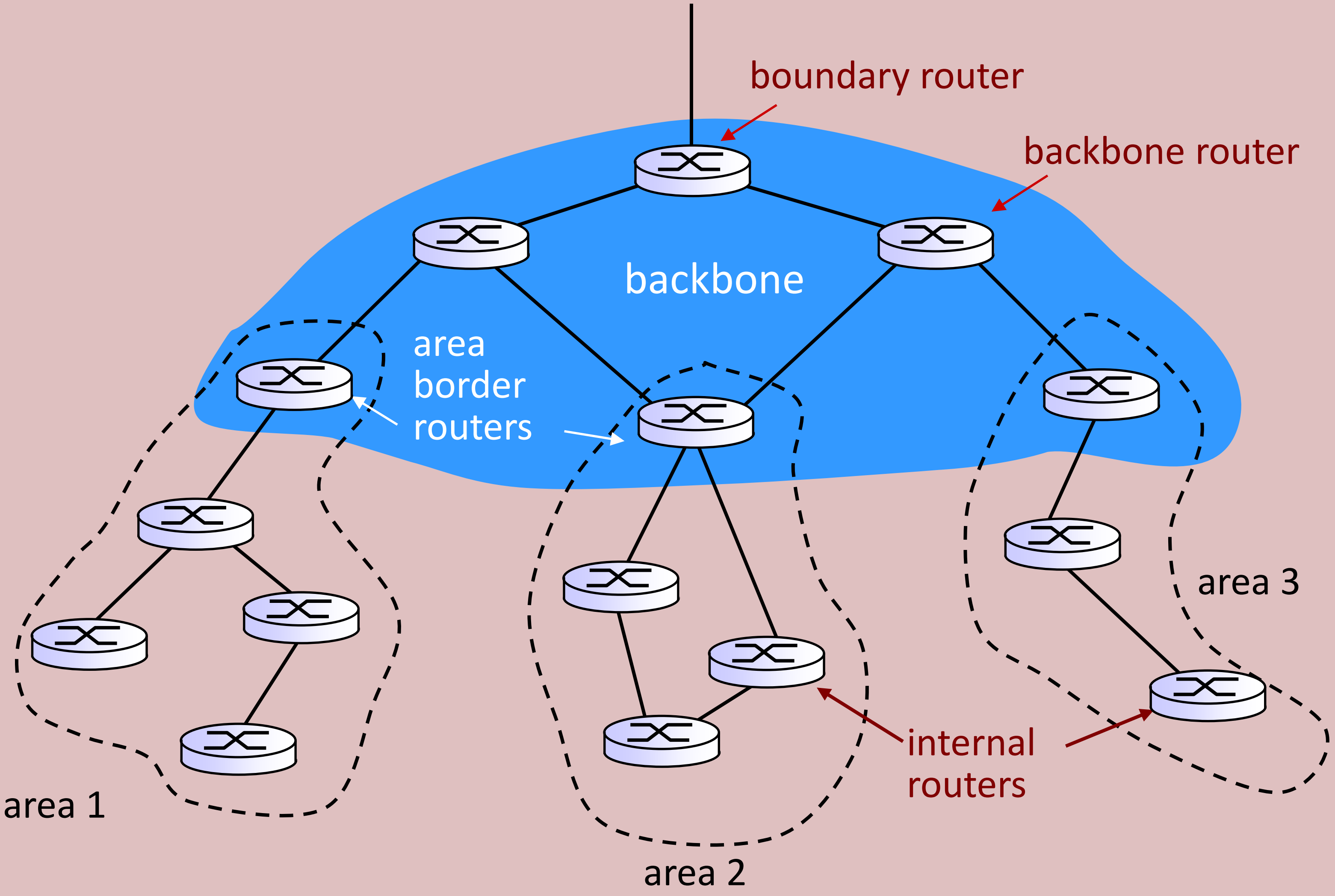
Hierarchical OSPF

- Area border routers: “summarize” distances to networks in own area, advertise to other areas

- Backbone routers: run OSPF routing limited to backbone

- Boundary routers: connect to other ASes

Hierarchical OSPF



Routing Information Protocol (RIP)

- One of the first intra-AS routing protocols in TCP/IP (first defined in RFC 1058)
- Distance vector algorithm
- Uses fewer resources than OSPF (no flooding)
- Three versions: RIP-1 and RIP-2 for IPv4 and RIPng for IPv6

RIP Updates

- Routers can pull routing information (e.g., immediately after booting up)
- Routers can push routing information
- Update timer typically fires every 30 seconds to trigger push
- Routes have expiration times (updates refresh the expiration timer)

RIP Issues

- Slow convergence since topology changes might take minutes to converge
- Routing loops can occur in rare instances
- Counting to infinity
- Newer OSPF is supposed to replace RIP

Thank You!

Networks

Inter-AS Routing

Adopted from material in “Computer Networking: A Top Down Approach” by Kurose and Ross and slides developed by William Conner

Inter-AS Routing

Section 5.4

Inter-AS Routing

- Routing among ASes
- Gateway routers must run inter-domain protocol in addition to intra-domain protocol
- Inter-domain protocol must be the same across different domains

Border Gateway Protocol (BGP)

- The de-facto inter-domain routing protocol
- *Glue* that binds the various ASes into the global internet
- Allows subnet to *advertise* its existence to the rest of the Internet
- Determine “good” routes to other networks based on reachability information *and policy*

Internal and External BGP

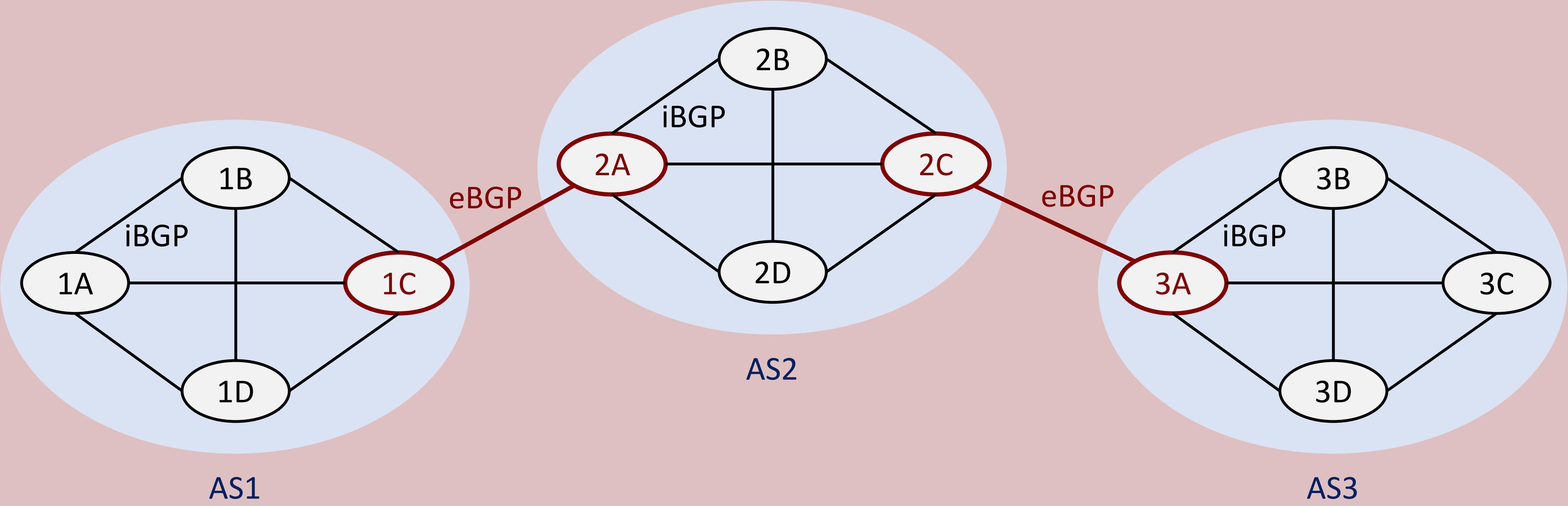
- External BGP (eBGP): allows each AS to obtain subnet reachability information from neighboring ASes
- Internal BGP (iBGP): allows each AS to propagate reachability information to all AS-internal routers

BGP Router

- BGP-capable routers are called speakers
- BGP neighbors are called peers
- Peers within same AS speak iBGP
- Peers across ASes speak eBGP

eBGP and iBGP

- Gateway routers run eBGP and iBGP
- Internal routers only run iBGP



BGP Session

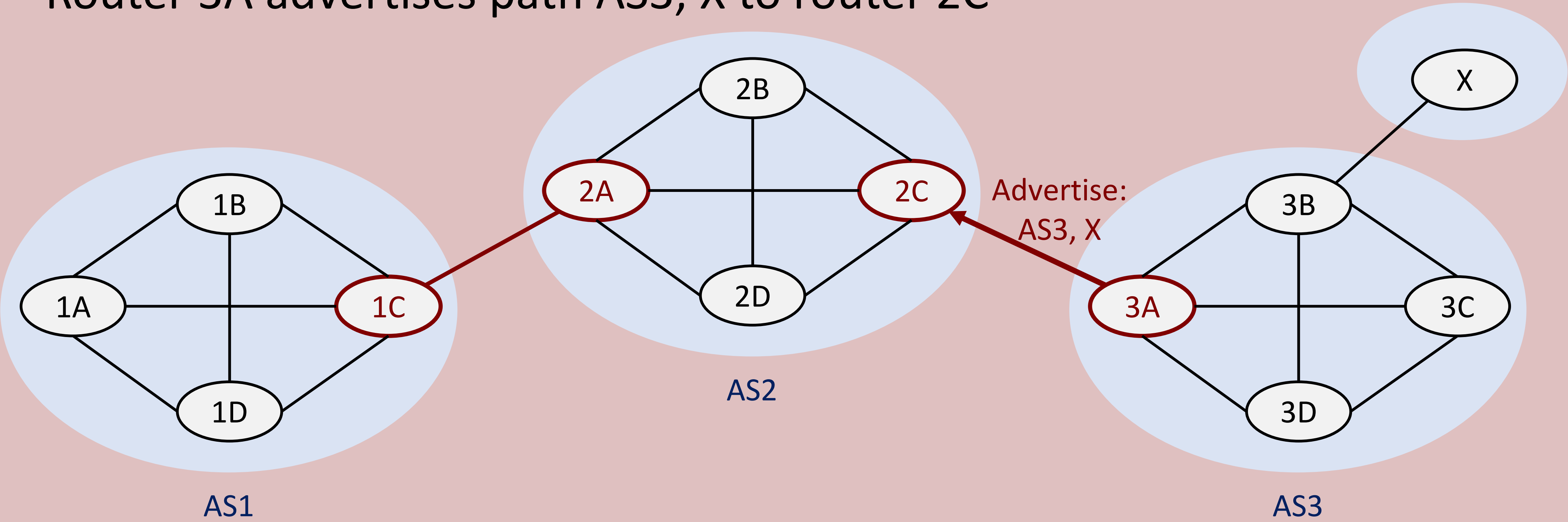
- Peers establish session over semi-permanent TCP connections
- Peers exchange path information to network prefixes via advertisements
- ASes indicate willingness to forward packets along advertised paths

BGP Message Types

- `OPEN`: opens TCP connection to remote BGP peer and authenticates
- `UPDATE`: advertises new path or withdraws old path
- `KEEPALIVE`: keeps connection alive in absence of UPDATEs; also ACKs OPEN request
- `NOTIFICATION`: reports errors; also used to close connection

Example: BGP Session

- AS3 is willing to route traffic from AS2 to X
- Router 3A advertises path AS3, X to router 2C



BGP Path Attributes

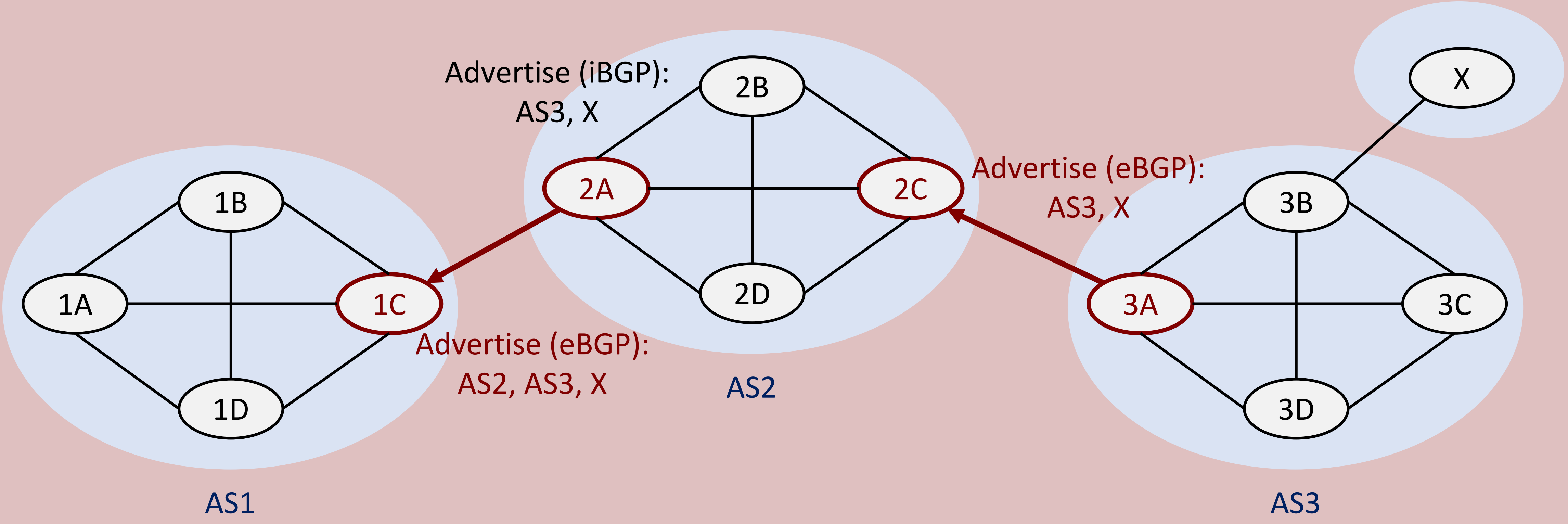
- BGP considers path attributes, not just distance (unlike OSPF and RIP)
- `AS-PATH:` list of ASes through which prefix advertisement has passed
- `NEXT-HOP:` IP address router interface that begins AS-PATH (why?)

BGP Policy-Based Routing

- Import policy: determines whether or not to accept path in advertisements received via gateway router (e.g., never route through AS4)

- Export policy: determines whether or not to advertise paths to other neighboring ASes (e.g., pretend that AS5 is not reachable even though it is)

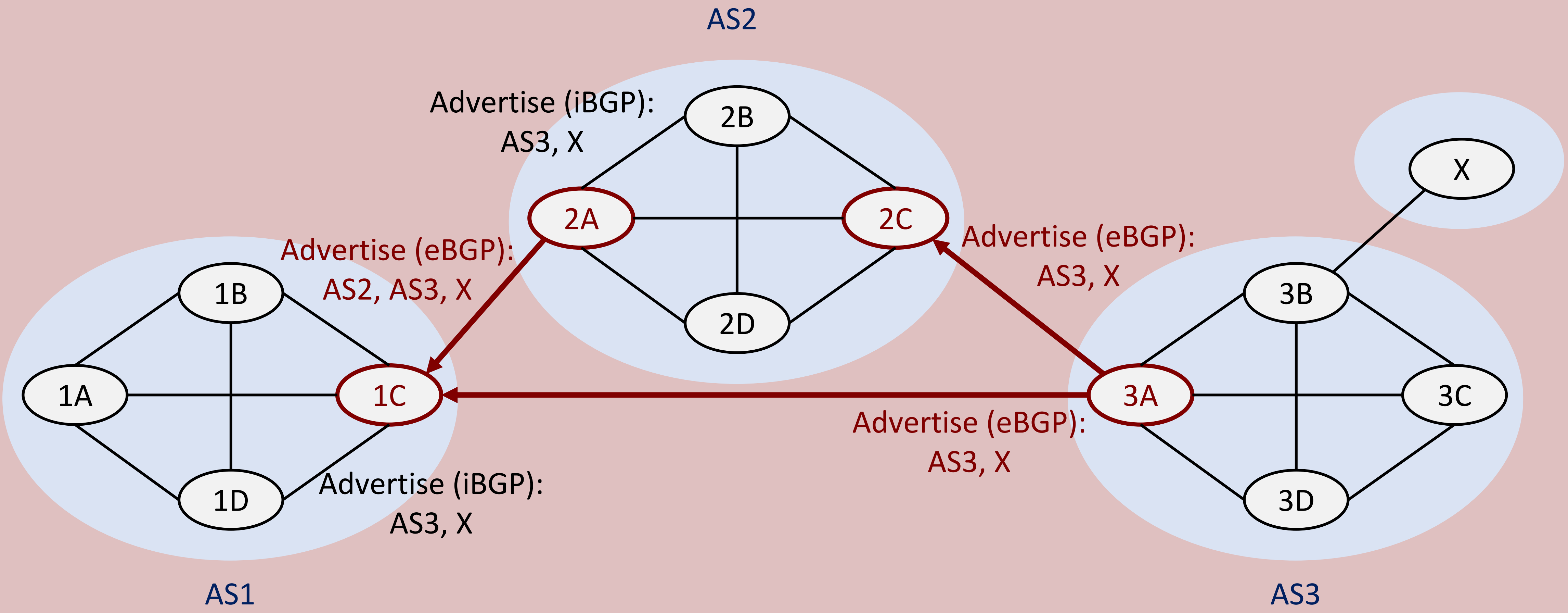
BGP Path Advertising



AS2 policy accepts path AS3, X

AS2 policy advertises path AS2, AS3, X

Multiple BGP Path Advertisements



AS1 policy prefers path AS3, X over AS2, AS3, X

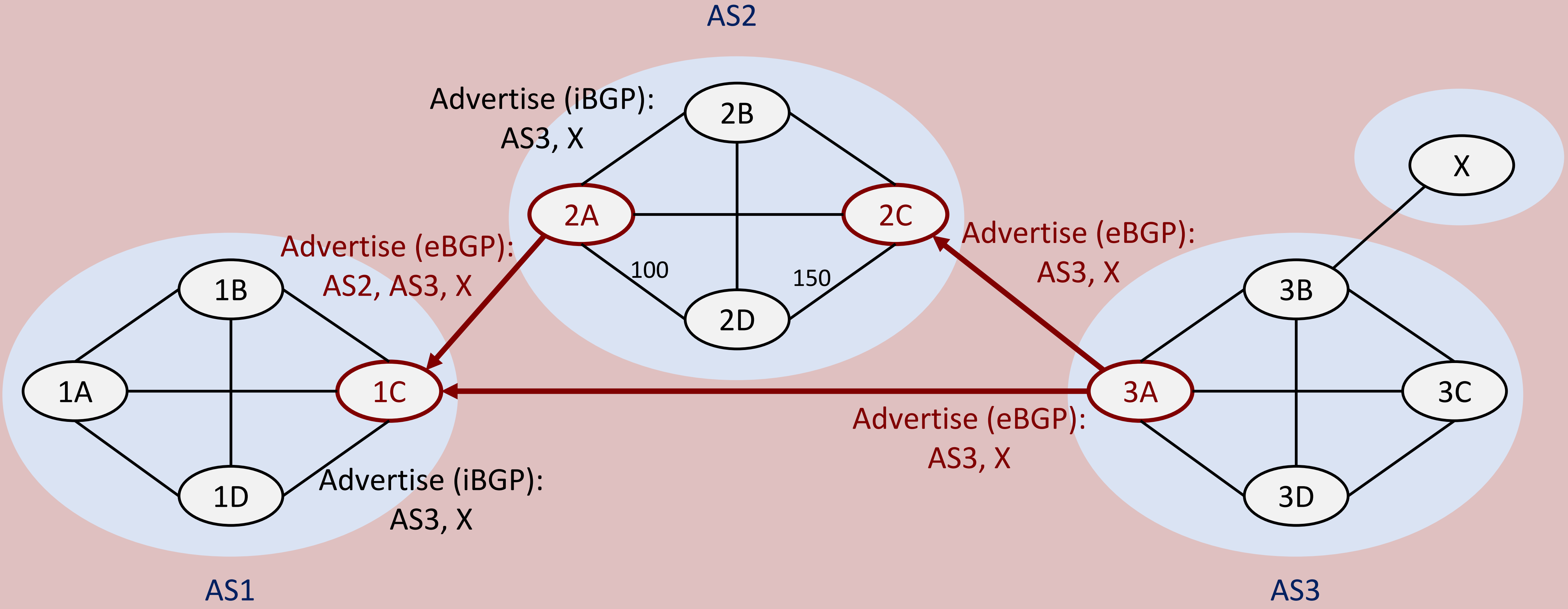
BGP Route Selection Criteria

1. Local preference value attribute (policy decision)
2. Shortest `AS-PATH`
3. Closest `NEXT-HOP` router (hot potato routing)
4. Additional criteria

Hot Potato Routing

- Optimize to achieve lowest intra-domain routing cost on your AS
- Do not worry about inter-domain routing cost for other ASes
- Can take precedence over other criteria (e.g., shortest `AS-PATH`)

Hot Potato Routing



Router 2D chooses route to 2A over 2C to reach X (i.e., longer `AS-PATH` for lower intra-domain cost)

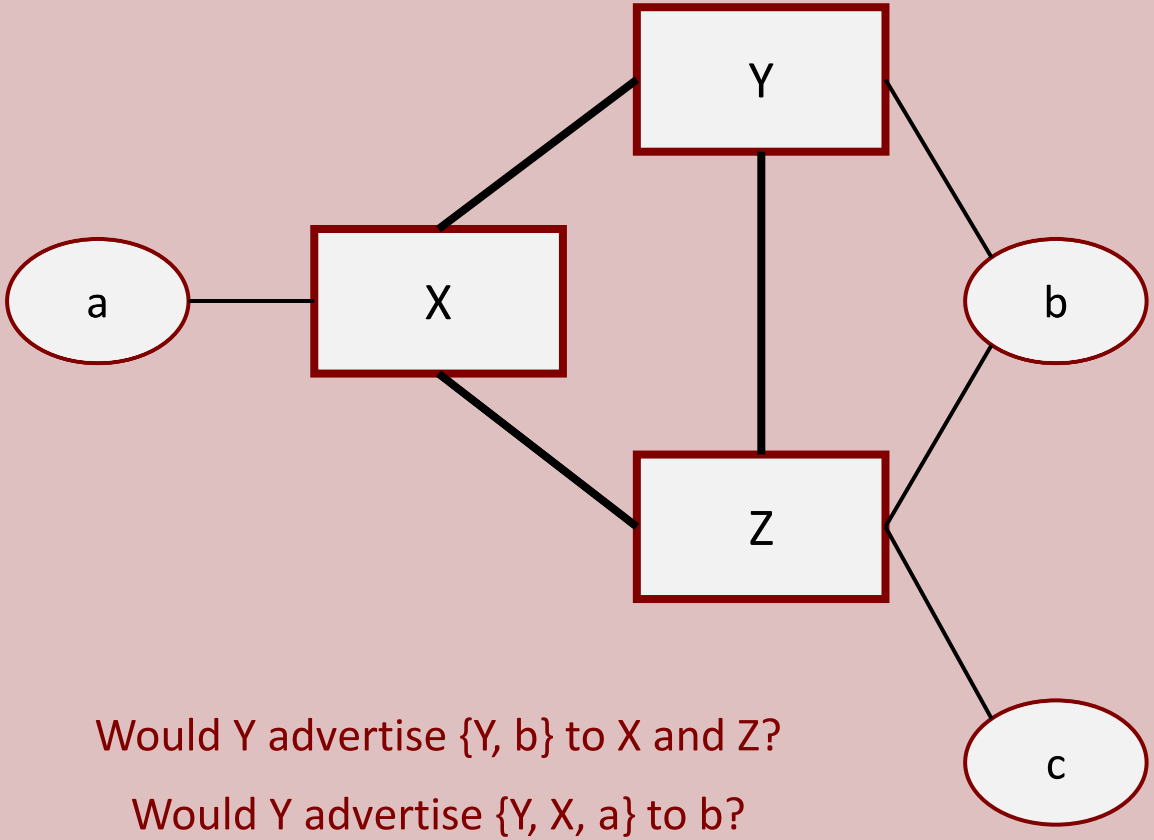
BGP Policy-Based Advertisements

Goal: ISP Y only routes traffic to/from its customers

X advertises {X, a} to Y and Z

Y does not advertise {Y, X, a} to Z because it does not earn any revenue from X, Z, or a

Z routes traffic via {Z, X, a}



Would Y advertise {Y, b} to X and Z?

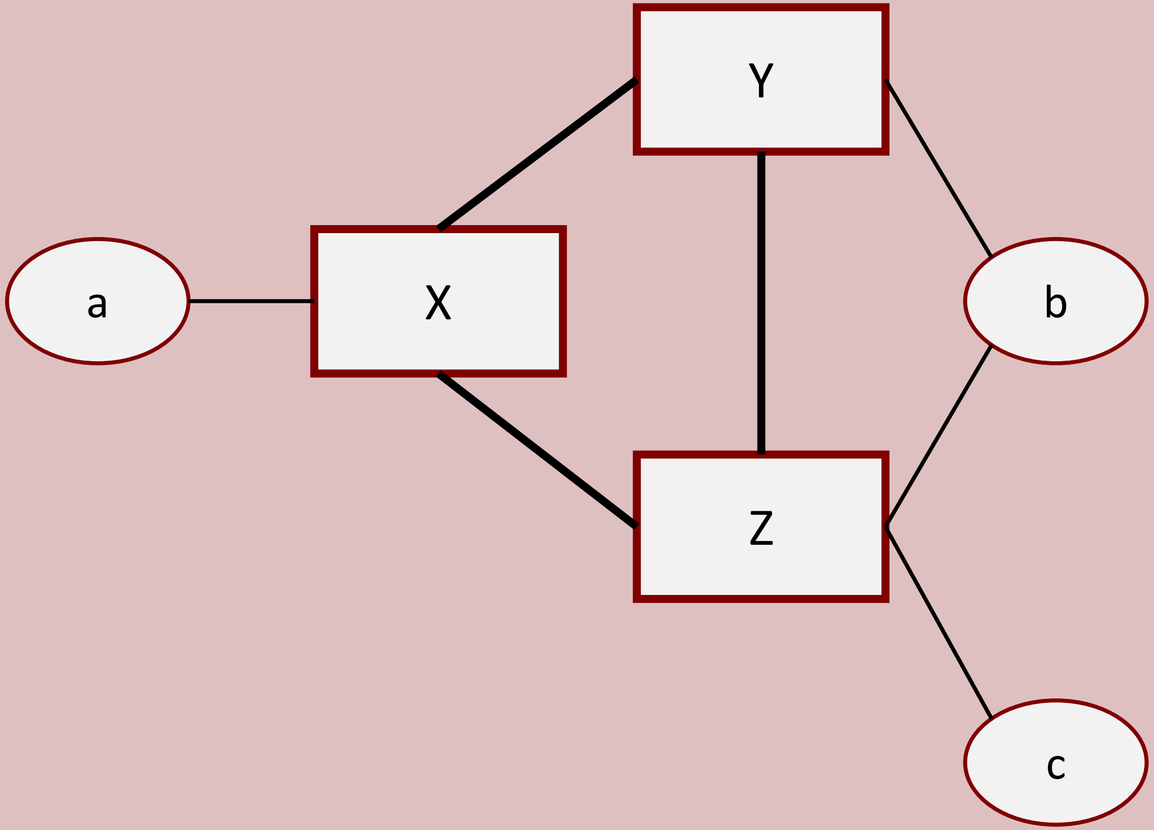
Would Y advertise {Y, X, a} to b?

BGP Policy-Based Advertisements

Goal: Multihomed customer b does not want to transit traffic between ISPs

b suppresses its reachability to Y from Z

b suppresses its reachability to Z from Y



Thank You!

Networks

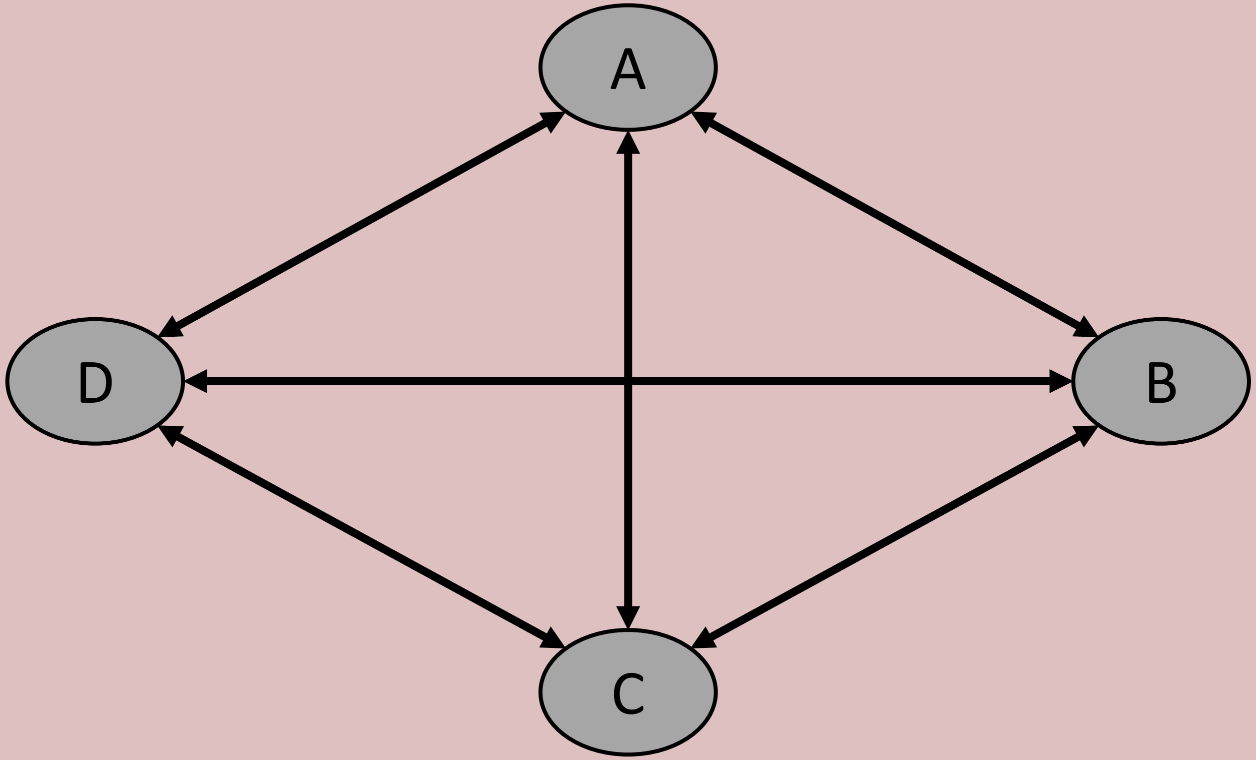
SDN Control Plane

Adopted from material in “Computer Networking: A Top Down Approach” by Kurose and Ross and slides developed by William Conner

SDN Control Plane

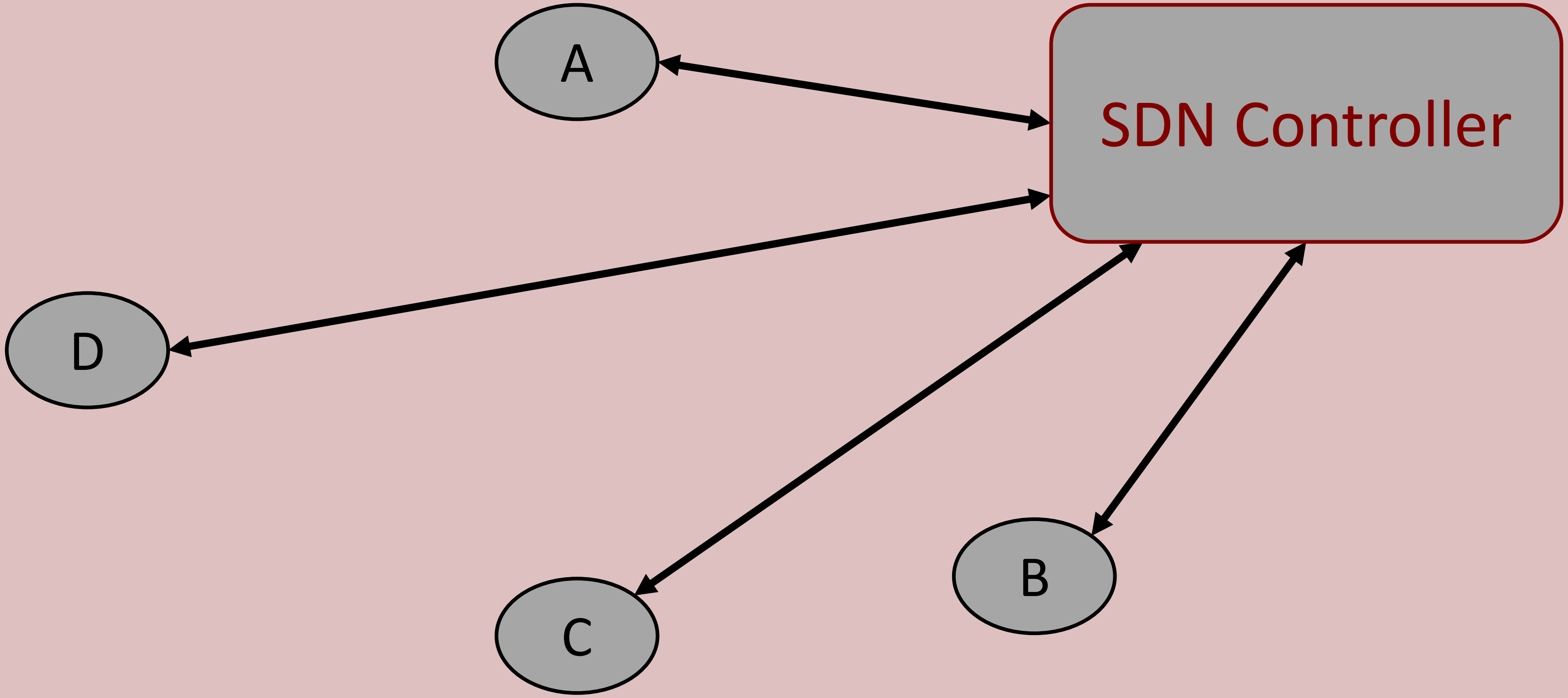
Section 5.5

Review: Per-Router Control Plane



Each router participates in traditional *routing algorithm* to compute its forwarding table

Review: Software-Defined Networking



Remote SDN controller computes and distributes forwarding tables.
Routers communicate to controller via controller agents.

Per-Router Control Plane

- Traditional Routing

- Monolithic routers running proprietary protocols on proprietary OSes (e.g., Cisco IOS)

- Separate hardware (middleboxes) for network functions: firewalls, load balancers, and NATs

SDN Motivation

- Easier network management
  - Avoid router misconfigurations
  - Greater flexibility of traffic flows
- Open (non-proprietary) implementation of control plane
  - Mix-and-match generalized forwarding hardware from different vendors
- Table-based forwarding allows “programming” routers
  - Centralized “programming” is easier: compute tables centrally and distribute
  - Distributed “programming” is more difficult: compute tables as result of distributed algorithm (protocol) implemented in each and every router

Analogy: Mainframe to PC Evolution

App

Open Interface

Windows (OS) or Linux or Mac OS

Open Interface

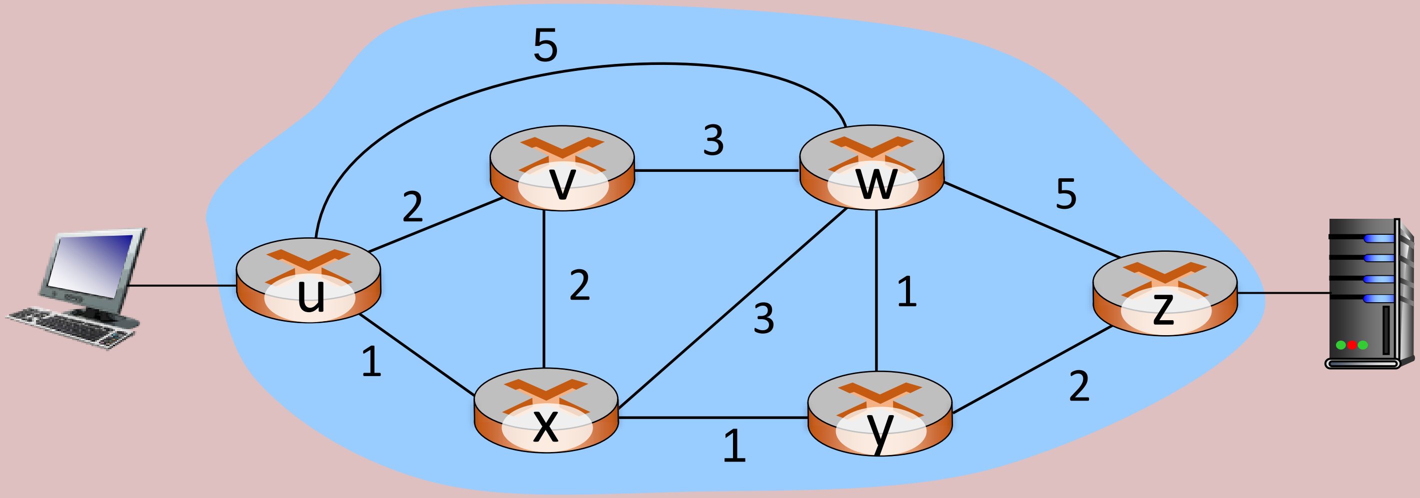
Microprocessor

Vertically integrated
Closed, proprietary
Slow innovation
Small industry

Horizontal
Open interfaces
Rapid innovation
Huge industry

* Slide courtesy: N. McKeown

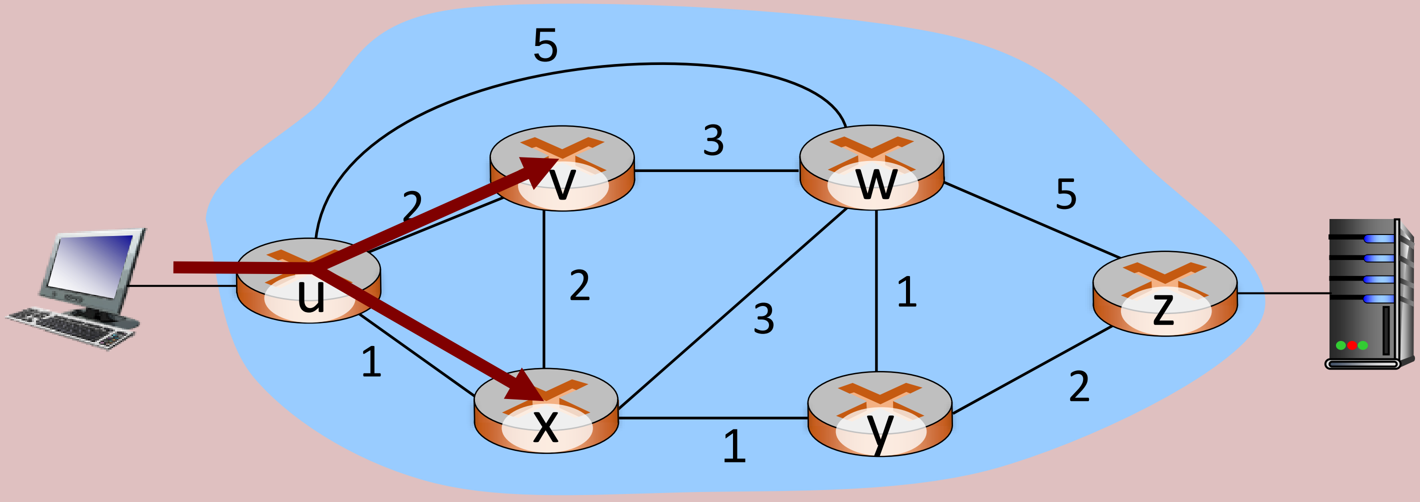
Traffic Engineering



Q: What if network operator wants u-to-z traffic to flow along uvwz, and x-to-z traffic to flow xwyz?

A: Need to define link weights so traffic routing algorithm computes routes accordingly (or need a new routing algorithm)!

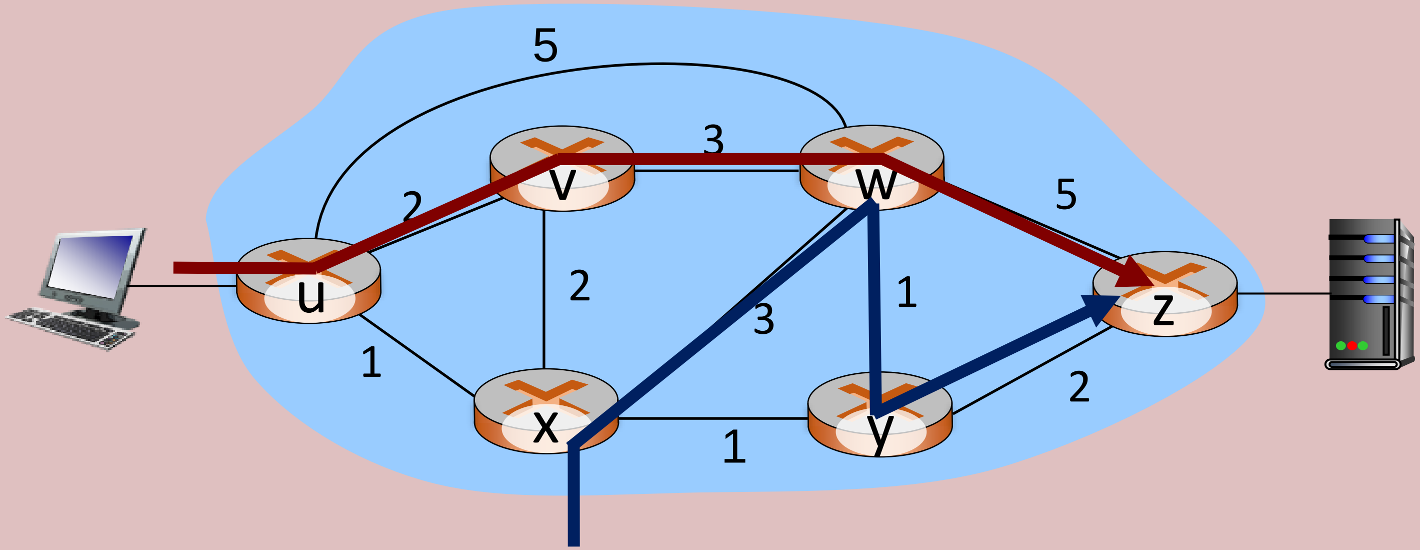
Traffic Engineering



Q: What if network operator wants u-to-z traffic to flow along uvwz and uxyz (load balancing)?

A: Can’t do it (or need a new routing algorithm)

Traffic Engineering



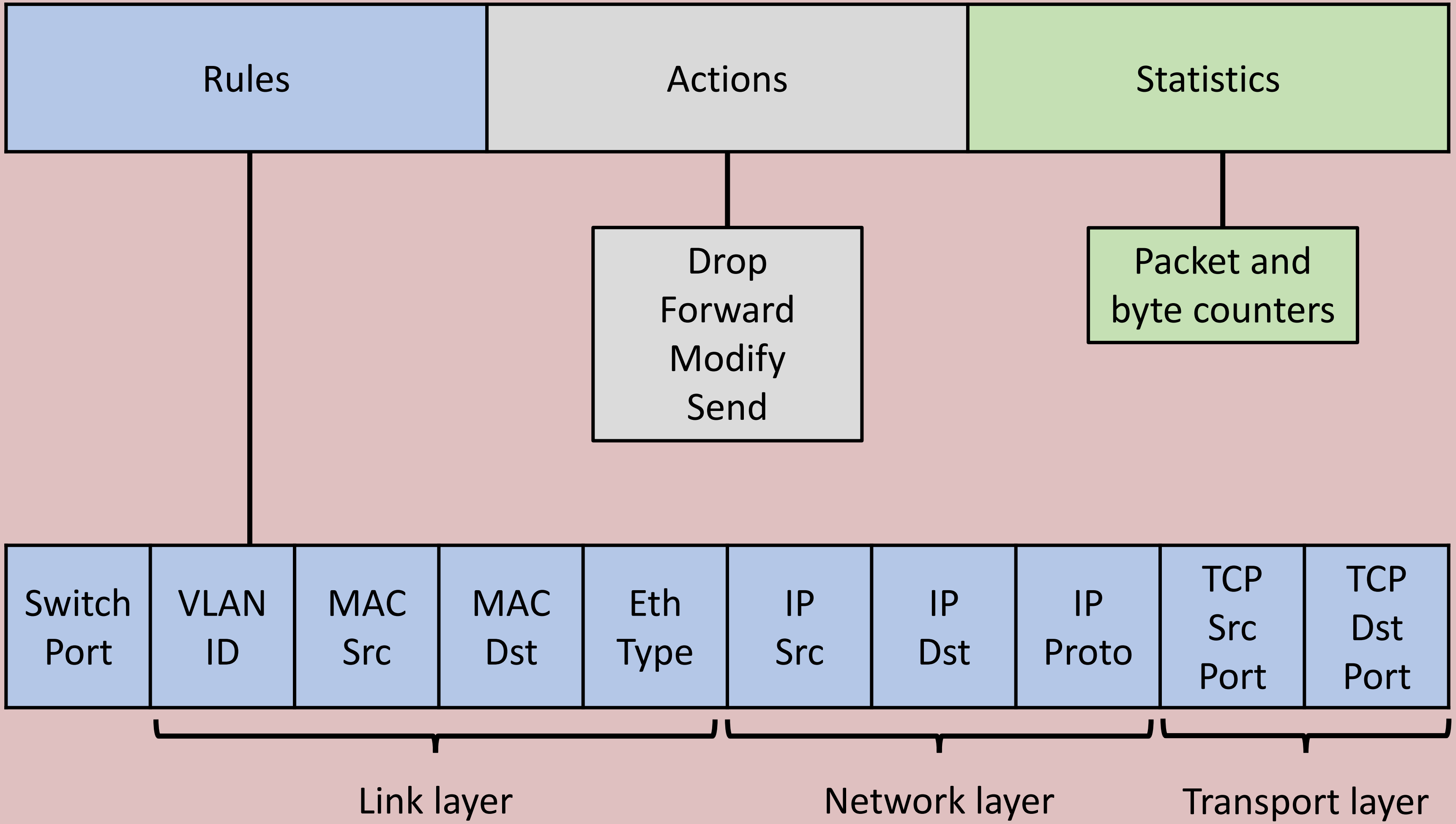
Q: What if w wants route blue and red traffic differently?

A: Can't do it (with destination based forwarding, and LS, DV routing)

Review: Generalized Forwarding

- Logically centralized controller computes and distributes flow tables
- More than just forwarding packet to destination (router)
- Following actions are also supported
  - Drop packet (firewall)
  - Modify packet (NAT)
  - Send packet to SDN controller

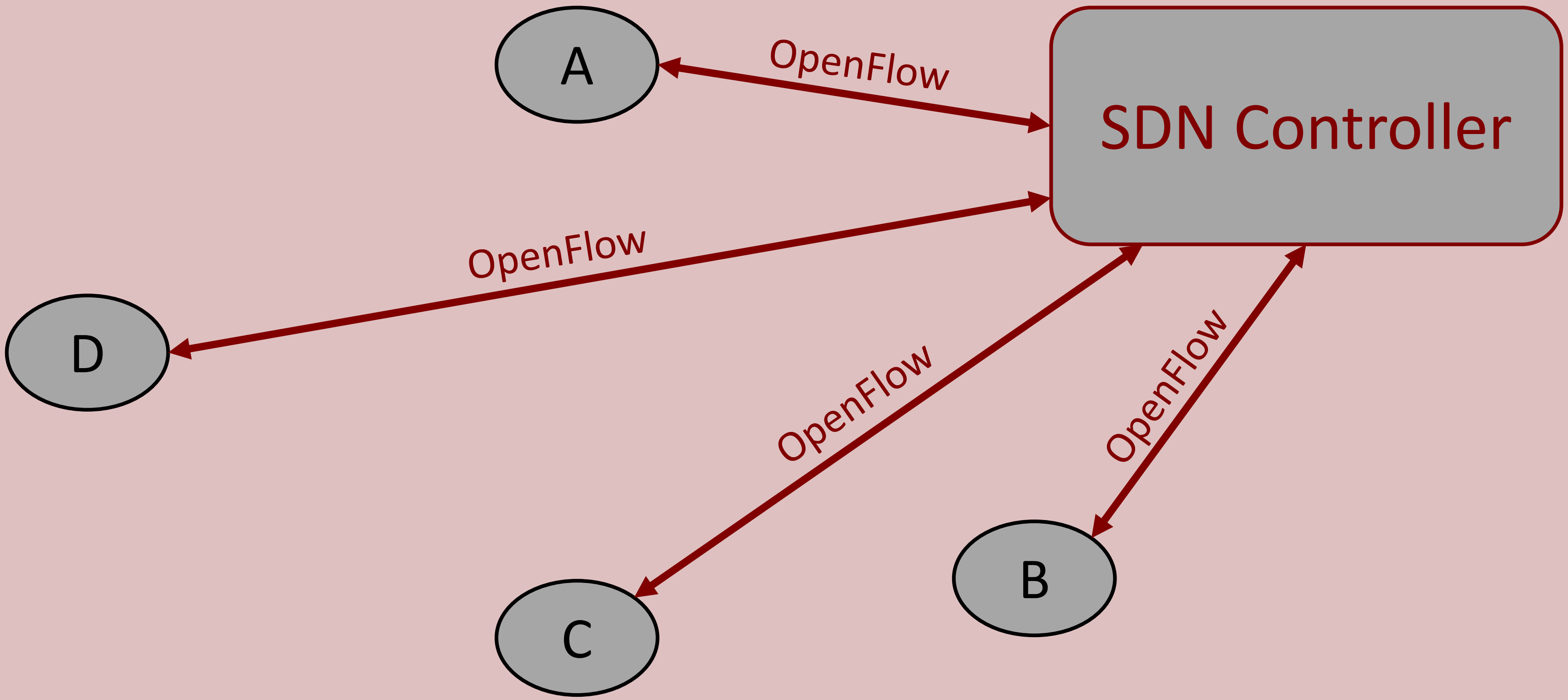
Review: Flow Table



Software-Defined Networking (SDN)

- Logically centralized SDN controller
- OpenFlow: protocol between SDN controller and routers; flow table format (mix and match hardware vendors)
- Routers/switches (same hardware) perform generalized forwarding based on flow tables from controller

SDN



Routers: report statistics and topology updates

SDN Controller: computes and distributes flow tables

Thank You!