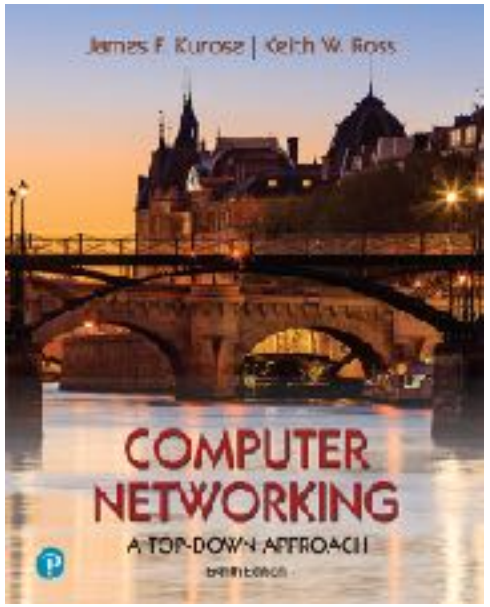
Chapter 5
Network Layer Control Plane



Computer Networking: A Top-Down Approach
8th edition
Jim Kurose, Keith Ross
Pearson, 2020

A note on the use of these PowerPoint slides:
We're making these slides freely available to all (faculty, students, readers). They're in PowerPoint form so you see the animations; and can add, modify, and delete slides (including this one) and slide content to suit your needs. They obviously represent a *lot* of work on our part. In return for use, we only ask the following:

- If you use these slides (e.g., in a class) that you mention their source (after all, we'd like people to use our book!)
- If you post any slides on a www site, that you note that they are adapted from (or perhaps identical to) our slides, and note our copyright of this material.

For a revision history, see the slide note for this page.

Thanks and enjoy! JFK/KWR

Day 19: Routing Protocols

CSEE 4119
Computer Networks
Ethan Katz-Bassett

COLUMBIA UNIVERSITY
IN THE CITY OF NEW YORK

Slides adapted from (and often identical to) slides from Kurose and Ross.

Unacceptable behavior in office hours and on Ed

- Multiple reports of inappropriate conduct during office hours
- Rude, condescending, or passive-aggressive comments during office hours or on Ed

- Feedback on course/HW is welcome, but make it constructive
- We want a positive atmosphere where everyone can enjoy learning and where instructors can best support you towards that goal
- TAs and classmates must be treated respectfully
  - Minority of students are hurting environment for majority who behave appropriately
  - TAs try their best & often go beyond what's expected to help all students

- Zero tolerance
  - If it happens again, you will not be allowed to attend TA office hours
  - If you experience or witness it, please speak up (to the misbehaving person or to instructors)

11/19 Admin

- Masks strongly recommended, over nose and mouth
- Attendance & participation not required
- If you are not feeling well or were exposed to COVID, please stay home
- Project 1 due a few days ago
- Project 2 released soon
  - Preliminary stage: very easy and fast, short timeline
  - Should finalize Hackathon details next week
- HW4 released soon
- Today is out of order compared to book
    - Skip generalized forwarding and routing algorithms, coming back to them later

Chapter 5: network layer control plane

chapter goals: understand principles behind network control plane

- traditional routing algorithms
- SDN controllers
- Internet Control Message Protocol
- network management

and their instantiation, implementation in the Internet:

- OSPF, BGP, OpenFlow, ODL and ONOS controllers, ICMP, SNMP

Chapter 5: outline

5.1 introduction

5.2 routing protocols

- link state
- distance vector

5.3 intra-AS routing in the Internet: OSPF

5.4 routing among the ISPs: BGP

5.5 The SDN control plane

5.6 ICMP: The Internet Control Message Protocol

5.7 Network management and SNMP

Chapter 5: outline

5.1 introduction

5.2 routing protocols

- link state
- distance vector

5.3 intra-AS routing in the Internet: OSPF

5.4 routing among the ISPs: BGP

5.5 The SDN control plane

5.6 ICMP: The Internet Control Message Protocol

5.7 Network management and SNMP

Chapter 5: outline

5.1 introduction

5.2 routing protocols

- link state
- distance vector

5.3 intra-AS routing in the Internet: OSPF

5.4 routing among the ISPs: BGP

5.5 The SDN control plane

5.6 ICMP: The Internet Control Message Protocol

5.7 Network management and SNMP

Chapter 5: outline

5.1 introduction

5.2 routing protocols

- link state
- distance vector

5.3 intra-AS routing in the Internet: OSPF

5.4 routing among the ISPs: BGP

5.5 The SDN control plane

5.6 ICMP: The Internet Control Message Protocol

5.7 Network management and SNMP

Chapter 5: outline

5.1 introduction

5.2 routing protocols

- link state
- distance vector

5.3 intra-AS routing in the Internet: OSPF

5.4 routing among the ISPs: BGP

5.5 The SDN control plane

5.6 ICMP: The Internet Control Message Protocol

5.7 Network management and SNMP

Chapter 5: outline

5.1 introduction

5.2 routing protocols

- link state
- distance vector

5.3 intra-AS routing in the Internet: OSPF

5.4 routing among the ISPs: BGP

5.5 The SDN control plane

5.6 ICMP: The Internet Control Message Protocol

5.7 Network management and SNMP

Chapter 5: outline

Today's class

5.1 introduction

5.2 routing protocols

- link state
- distance vector

5.3 intra-AS routing in the Internet: OSPF

5.4 routing among the ISPs: BGP

5.5 The SDN control plane

5.6 ICMP: The Internet Control Message Protocol

5.7 Network management and SNMP

Chapter 5: outline

5.1 introduction

5.2 routing protocols

- link state
- distance vector

5.3 intra-AS routing in the Internet: OSPF

5.4 routing among the ISPs: BGP

5.5 The SDN control plane

5.6 ICMP: The Internet Control Message Protocol

5.7 Network management and SNMP

(refresher) Network-layer functions

Recall: two network-layer functions:

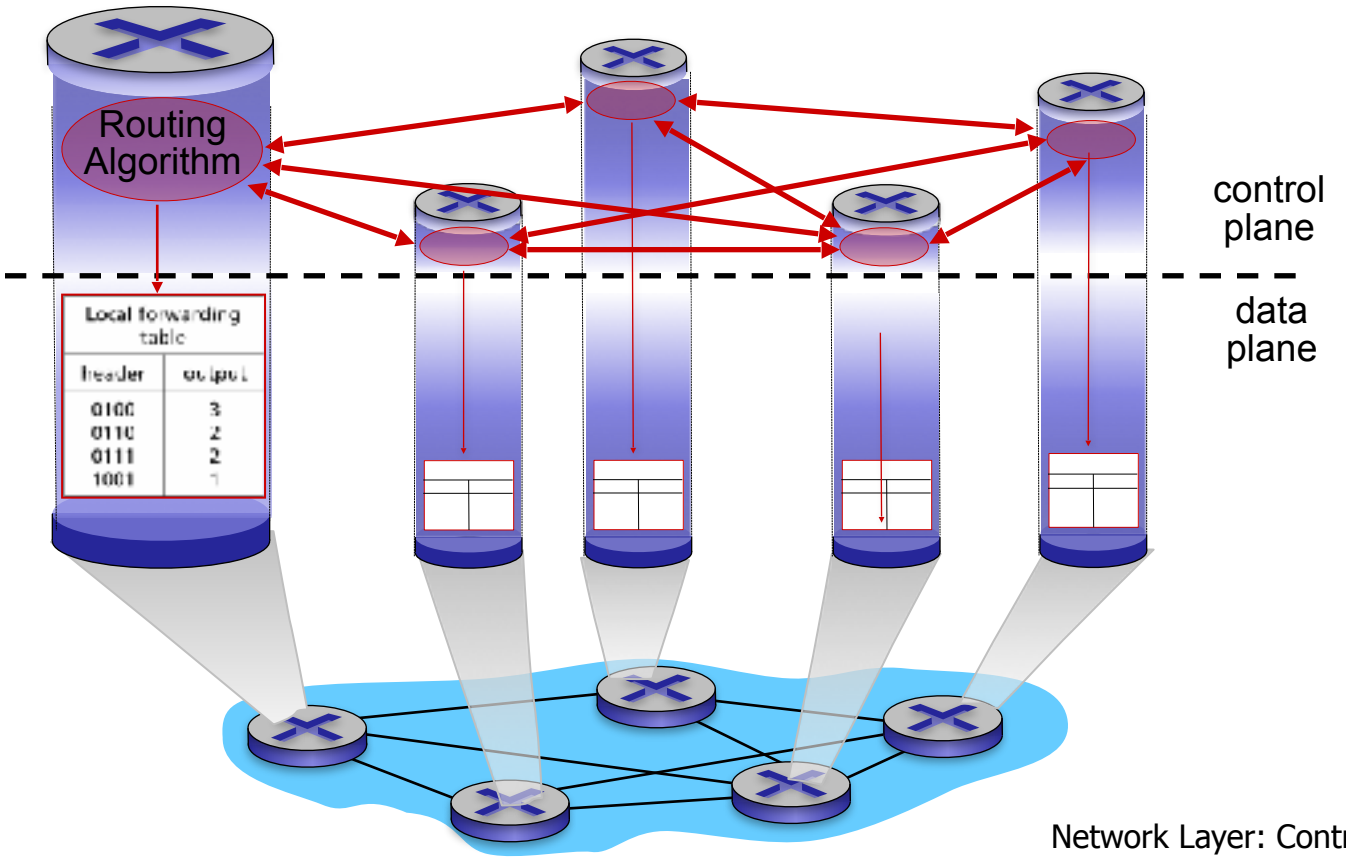
- *forwarding:* move packets from router's input to appropriate router output — *data plane*
- *routing:* determine route taken by packets from source to destination — *control plane*

Two approaches to structuring network control plane:

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

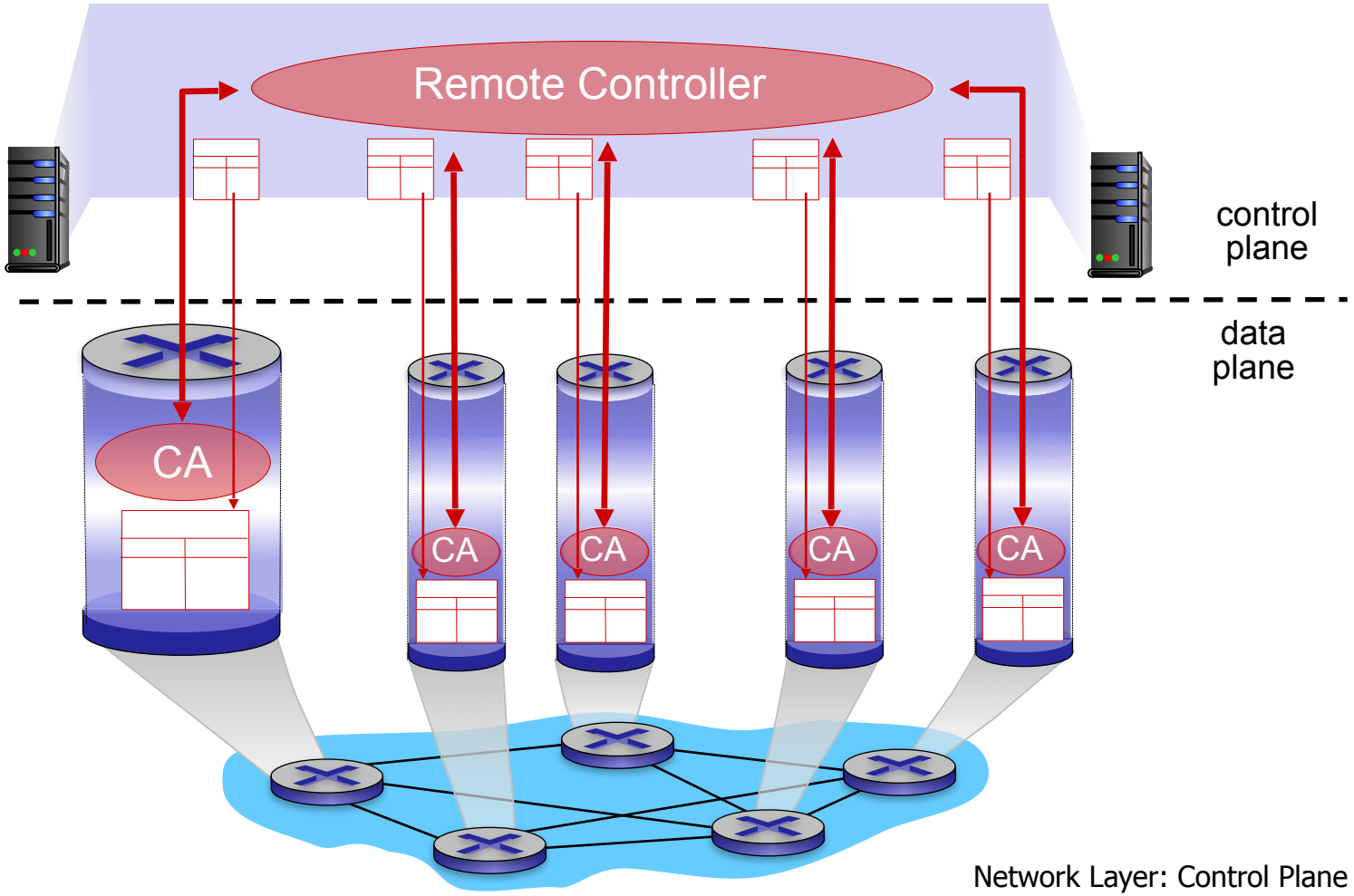
(refresher) Per-router control plane

Individual routing algorithm components *in each and every router* interact with each other in control plane to compute forwarding tables



(refresher) Software-Defined Networking (SDN) Logically centralized control plane

A distinct (typically remote) controller interacts with local control agents (CAs) in routers to compute forwarding tables



Chapter 5: outline

5.1 introduction

5.2 routing protocols

- link state
- distance vector

5.3 intra-AS routing in the Internet: OSPF

5.4 routing among the ISPs: BGP

5.5 The SDN control plane

5.6 ICMP: The Internet Control Message Protocol

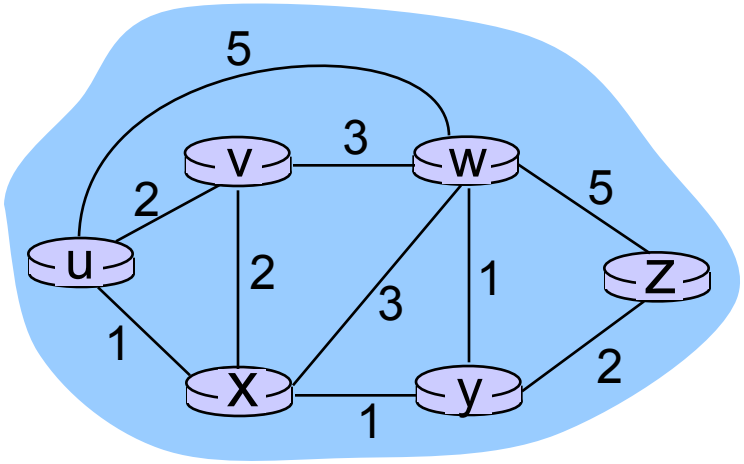
5.7 Network management and SNMP

Routing protocols

Routing protocol goal: determine "good" paths (equivalently, routes), from sending hosts to receiving host, through network of routers

- path: sequence of routers packets will traverse in going from given initial source host to given final destination host
- "good": least "cost", "fastest", "least congested", "best meets our goals/policies"
- routing: a "top-10" networking challenge!

Graph abstraction of the network



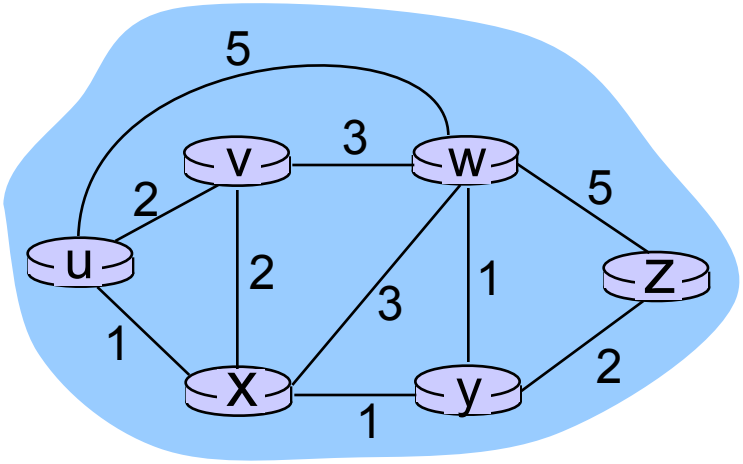
graph: G = (N,E)

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) }

aside: graph abstraction is useful in other network contexts, e.g., P2P, where *N* is set of peers and *E* is set of TCP connections

Graph abstraction: costs



c(x,x') = cost of link (x,x')
e.g., c(w,z) = 5

cost could always be 1,
based on physical length, cost,
inversely related to bandwidth,
inversely 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
- OSPF
- ***not*** necessarily centralized

decentralized:

- routers start knowing physically-connected neighbors, link costs to neighbors
- exchange info with neighbors of what each can reach, update based on this
- iterative process of computation
- "distance vector" algorithms
- BGP

Q: static or dynamic?

static:

- routes change slowly over time

dynamic:

- routes change more quickly
  - periodic update
  - in response to link cost changes

Internet Routing: OSPF & BGP

CSEE 4119
Computer Networks
Ethan Katz-Bassett

Chapter 5: outline

5.1 introduction

5.2 routing protocols

- link state
- distance vector

5.3 intra-AS routing in the Internet: OSPF

5.4 routing among the ISPs: BGP

5.5 The SDN control plane

5.6 ICMP: The Internet Control Message Protocol

5.7 Network management and SNMP

Making routing scalable

our routing discussion 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" (ASes) (a.k.a. "domains")

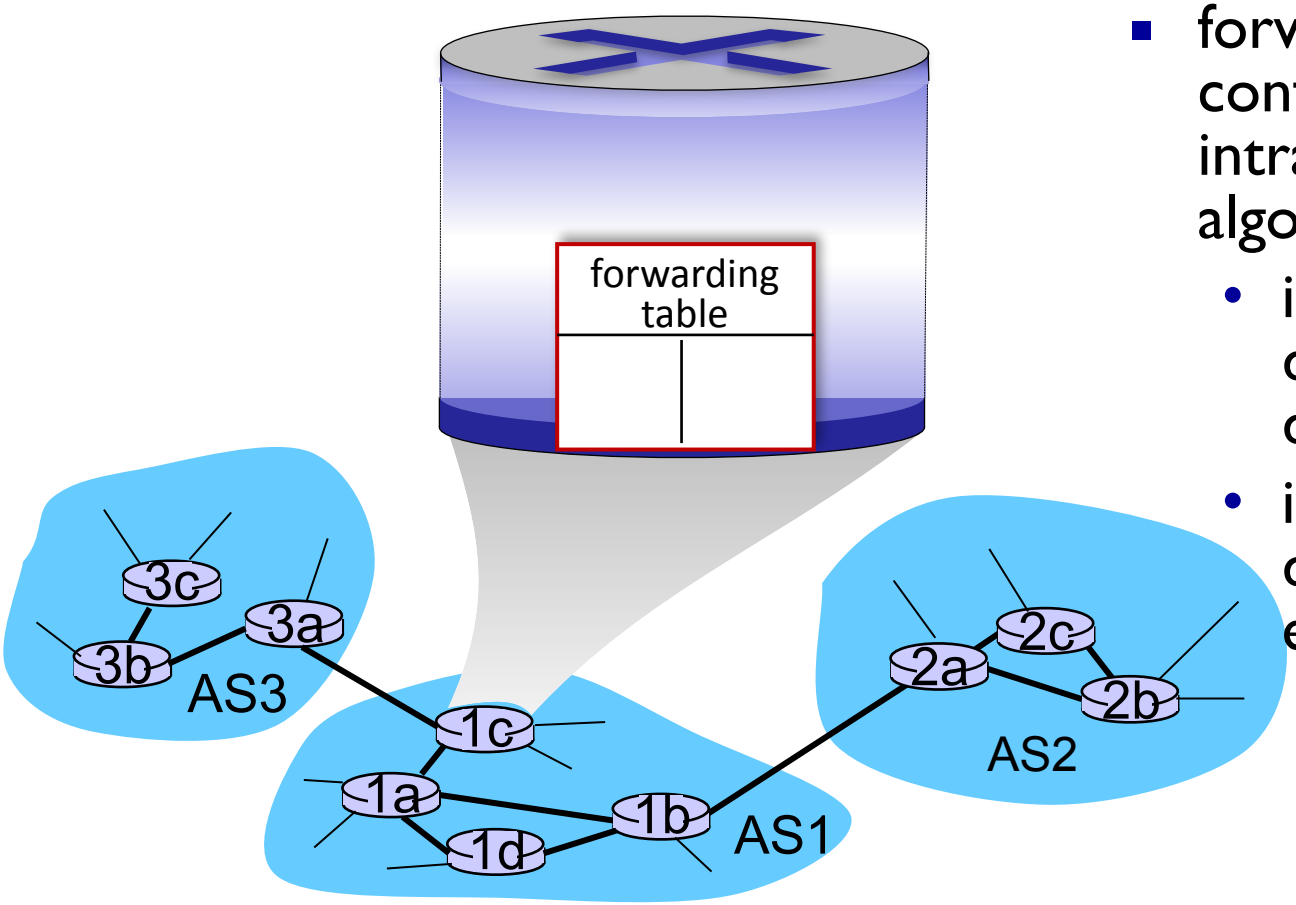
intra-AS routing

- routing among hosts, routers in same AS (network)
- all routers in AS must run *same* intra-domain protocol
- routers in *different* ASes can run *different* intra-domain protocols
- gateway router: at "edge" of an own AS, has link(s) to router(s) in other ASes

inter-AS routing

- routing among ASes
- gateways perform inter-domain routing (as well as intra-domain routing)

Interconnected ASes



- 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

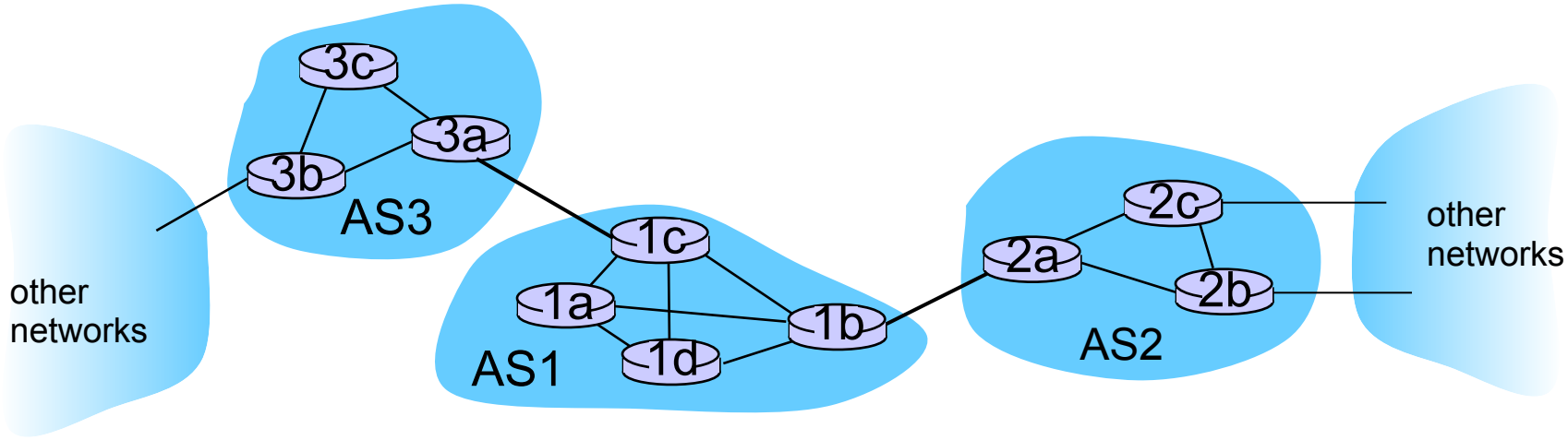
Inter-AS tasks

- 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!



Intra-AS Routing

- also known as *interior gateway protocols (IGP)*
- 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

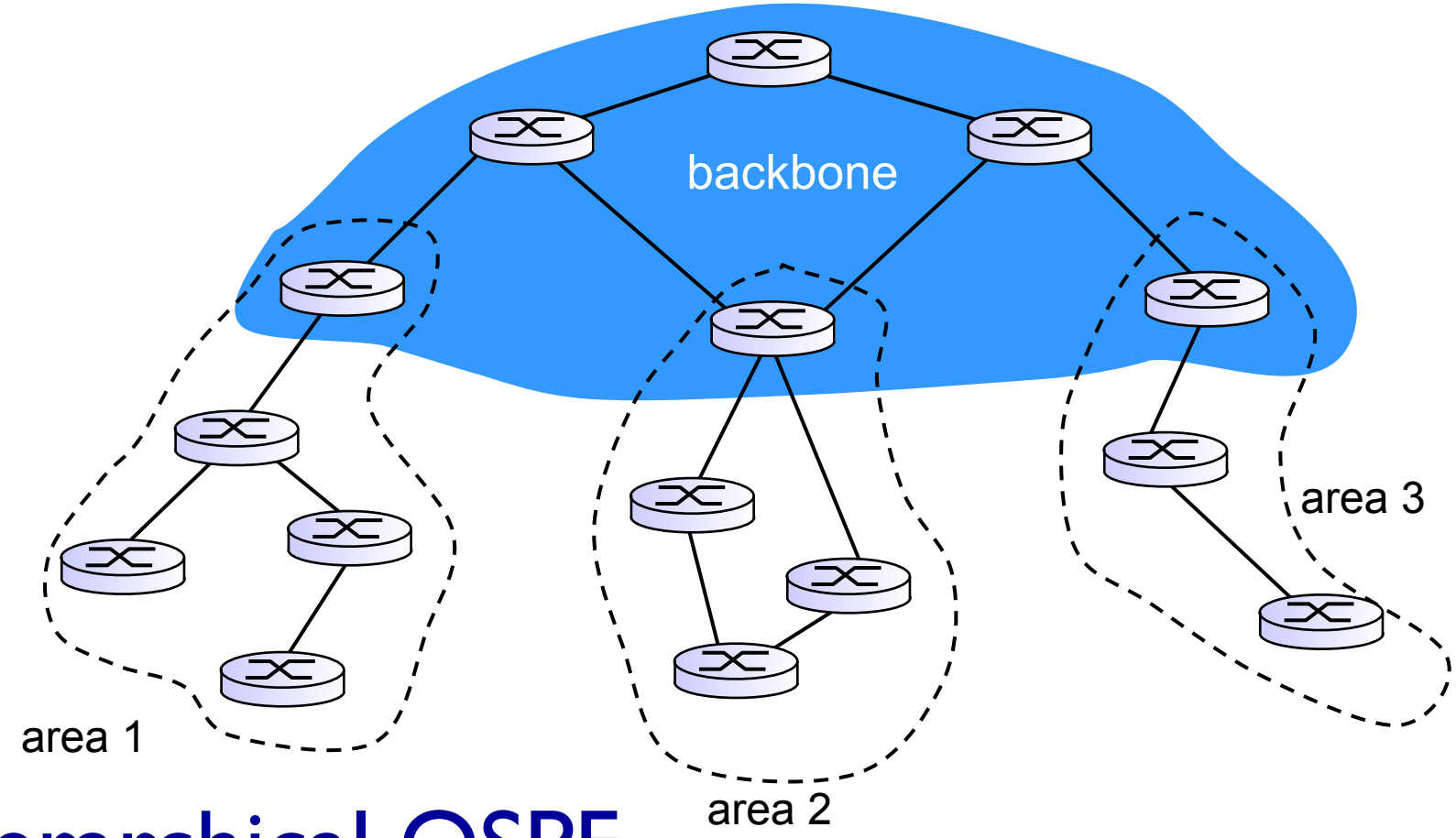
OSPF (Open Shortest Path First)

- "open": publicly available
- OSPF messages directly over IP (no TCP/UDP)
- uses link-state algorithm
  - each router floods OSPF link-state advertisements to all other routers in *entire* AS
    - link state: each connected links (and subnet) and its cost
    - gives each router complete topology, with link costs
  - each router computes least cost path to all destinations,
    - builds its forwarding table
    - route computation using Dijkstra's algorithm

OSPF "advanced" features

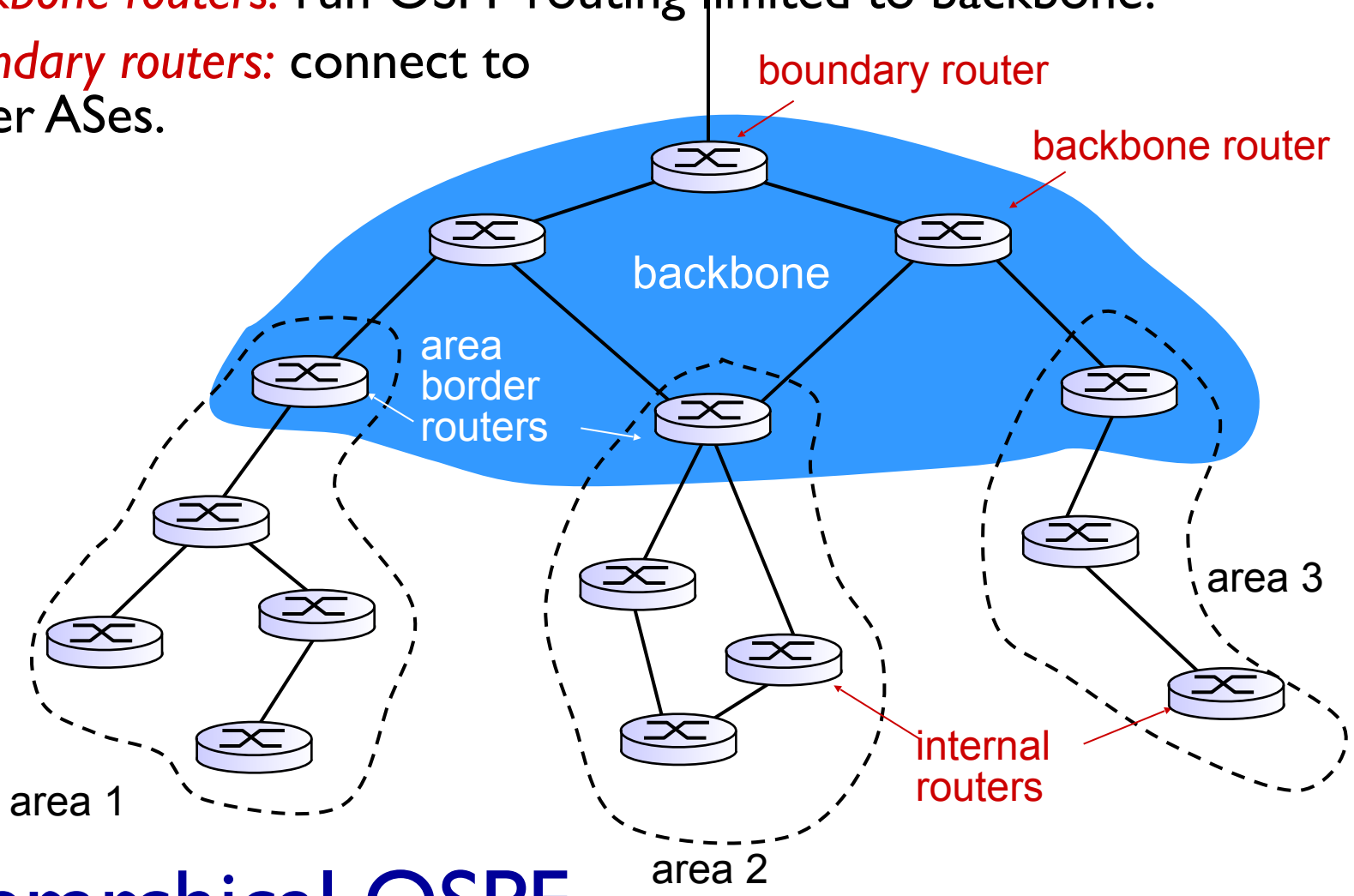
- *security:* all OSPF messages authenticated (to prevent malicious intrusion)
- multiple same-cost paths allowed (RIP only allows one path)
- for each link, multiple cost metrics for different TOS (e.g., satellite link cost set low for best effort ToS; high for real-time ToS)
- integrated uni- and multi-cast support:
  - Multicast OSPF (MOSPF) uses same topology data base as OSPF
- hierarchical OSPF in large domains.

- *two-level hierarchy:* local area, backbone.
  - link-state advertisements only in area
  - each nodes has detailed area topology; only know direction (shortest path) to nets in other areas.



Hierarchical OSPF

- *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 ASes.



Hierarchical OSPF

Chapter 5: outline

5.1 introduction

5.2 routing protocols

- link state
- distance vector

5.3 intra-AS routing in the Internet: OSPF

5.4 routing among the ISPs: BGP

5.5 The SDN control plane

5.6 ICMP: The Internet Control Message Protocol

5.7 Network management and SNMP

Internet inter-AS routing: BGP

- BGP (Border Gateway Protocol): ***the*** inter-domain routing protocol
  - glue that holds the Internet together
  - allows subnet to advertise its existence to rest of Internet: "I am here"
  - propagate these advertisements to build routes
- BGP provides each AS a means to:
  - determine "good" routes to other networks based on reachability information and *policy*

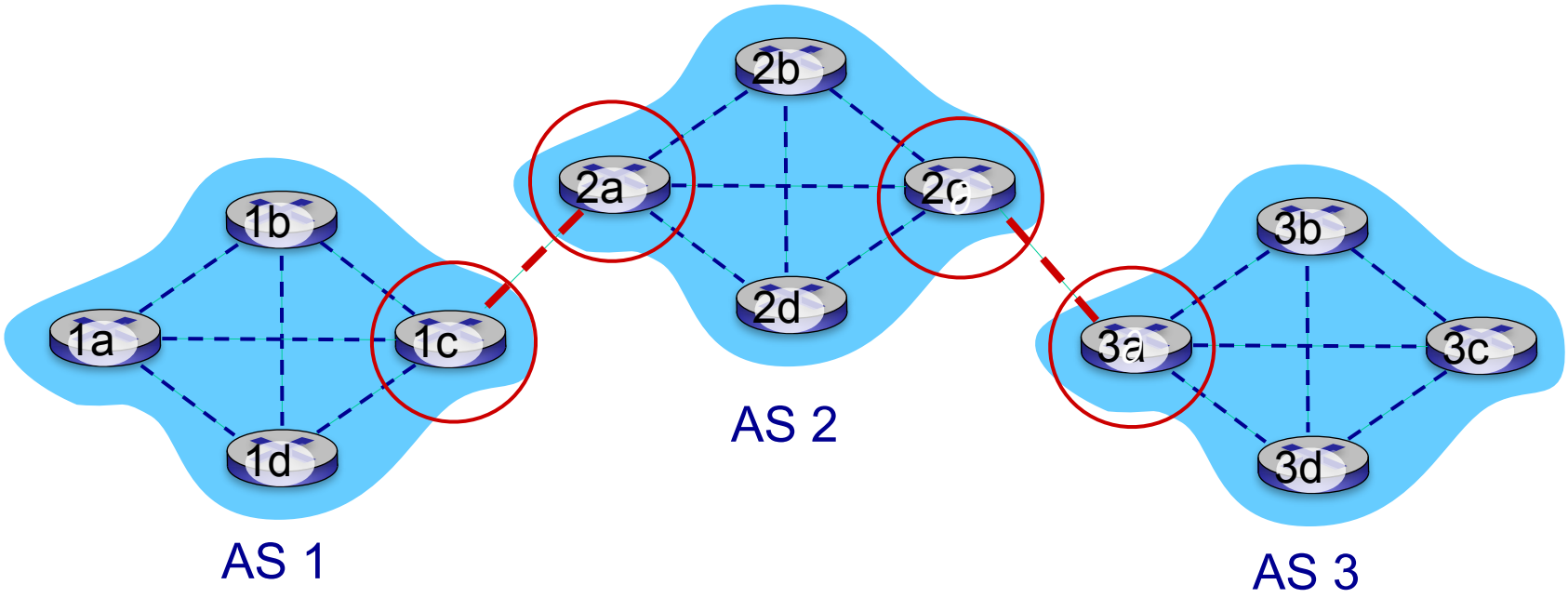
BGP is policy-based

- *Policy-based routing:*
  - gateway receiving route advertisement uses *import policy* to accept/decline path (e.g., never route through AS Y).
  - *route selection policy* to pick best path when multiple exist
    - NOT NECESSARILY shortest path
  - *export policy* determines which (if any) neighboring ASes to advertise best path to
    - NOT NECESSARILY all paths to all neighbors
    - CAN be different to different neighbors

Internet inter-AS routing: BGP

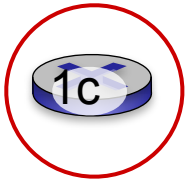
- BGP (Border Gateway Protocol): ***the*** inter-domain routing protocol
  - glue that holds the Internet together
  - allows subnet to advertise its existence to rest of Internet: "I am here"
  - propagate these advertisements to build routes
- BGP provides each AS a means to:
  - determine "good" routes to other networks based on reachability information and *policy*
  - eBGP: obtain subnet reachability information from neighboring ASes
  - iBGP: propagate reachability information to all AS-internal routers.

eBGP, iBGP connections



eBGP connectivity, only between connected gateway routers

iBGP connectivity, full mesh within each AS



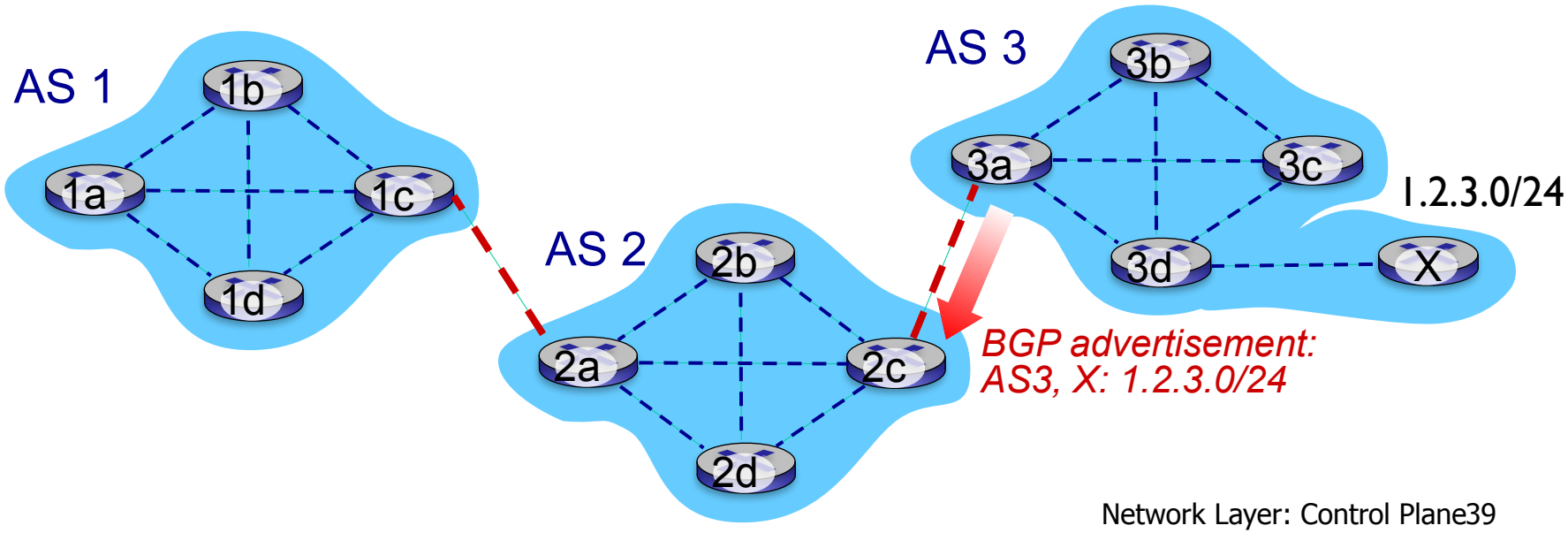
gateway routers run both eBGP and iBGP protools

BGP messages

- BGP messages exchanged between peers over TCP connection
- BGP messages:
  - OPEN: opens TCP connection to remote BGP peer and authenticates sending BGP peer
  - UPDATE: advertises new path (or withdraws old)
  - KEEPALIVE: keeps connection alive in absence of UPDATES; also ACKs OPEN request
  - NOTIFICATION: reports errors in previous msg; also used to close connection

BGP basics

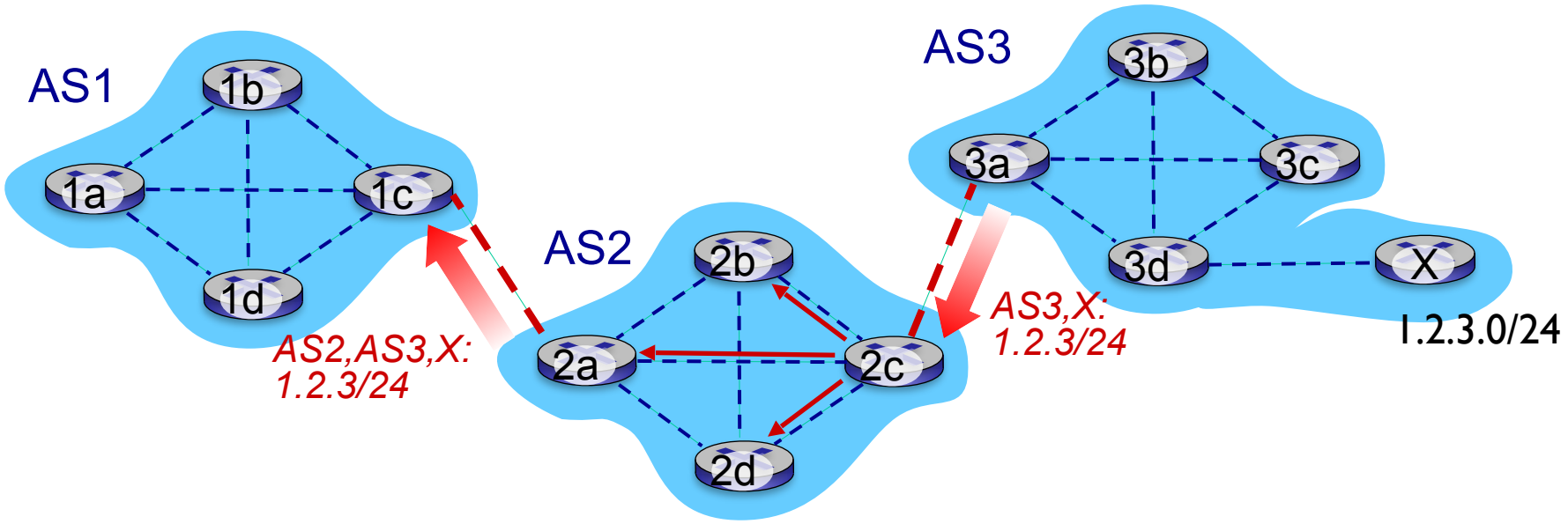
- BGP session: two BGP routers ("peers") exchange BGP messages over semi-permanent TCP connection:
  - advertising *paths* to different destination network prefixes (BGP is a "path vector" protocol)
- when AS3 gateway router 3a advertises path to 1.2.3.0/24 to AS2 gateway router 2c, AS3 *promises* to AS2 it will forward datagrams towards 1.2.3.0/24



Path attributes and BGP routes

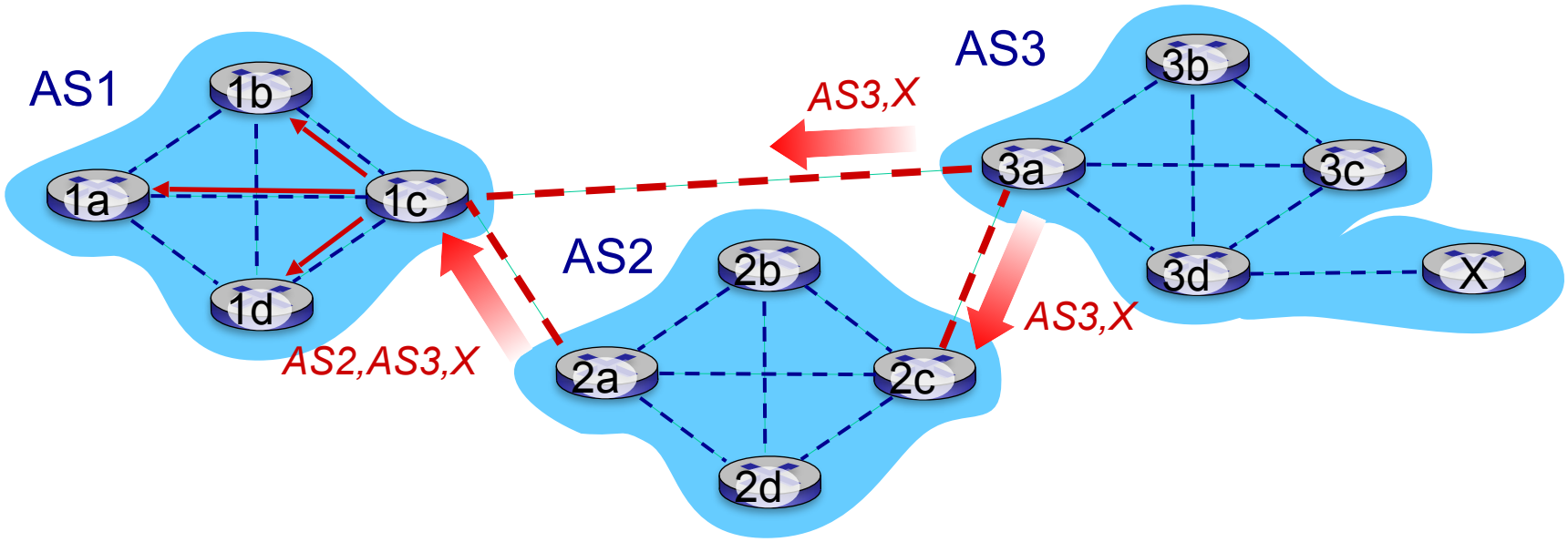
- advertised prefix includes BGP attributes
  - prefix + attributes = "route"
- two important attributes:
  - AS-PATH: list of ASes through which advertisement passed
    - enables loop detection
  - NEXT-HOP:...<coming in a few slides>

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



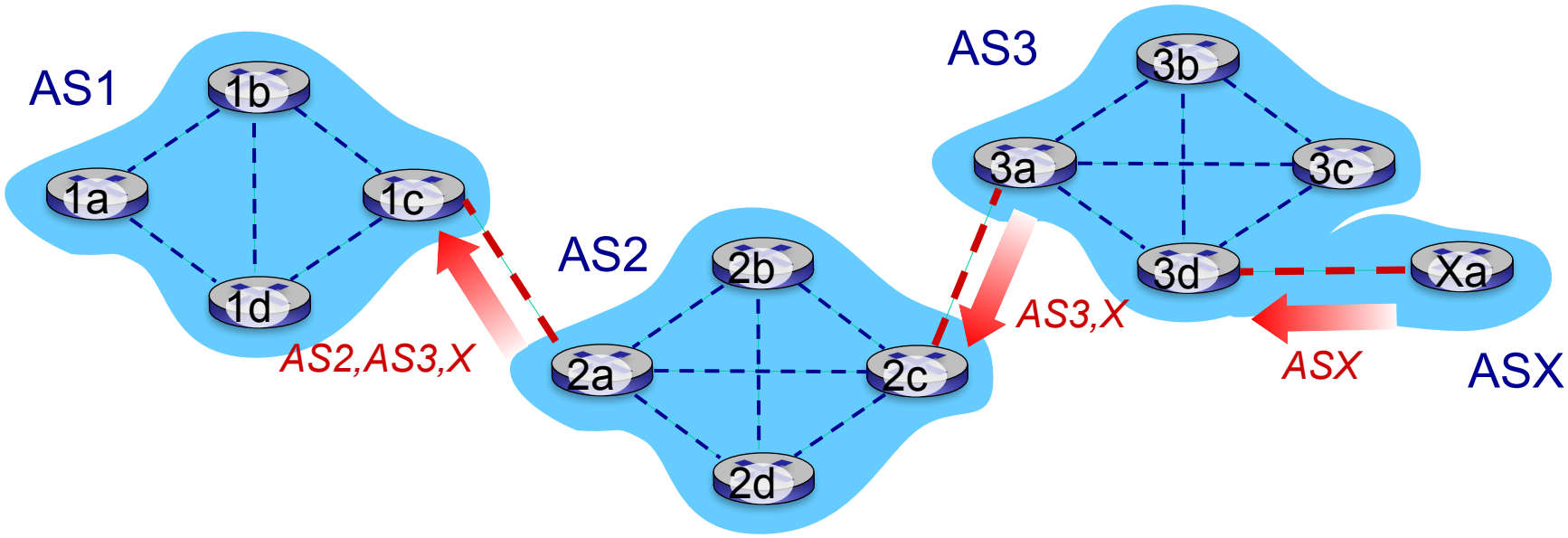
gateway router may learn about multiple paths to destination:

- 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*

Path attributes and BGP routes

- advertised prefix includes BGP attributes
  - prefix + attributes = "route"
- two important attributes:
  - AS-PATH: list of ASes through which advertisement passed
    - enables loop detection
  - NEXT-HOP: indicates IP address to forward traffic to in order to achieve AS path
    - usually first (border) router in next AS or last (border) router in this AS
    - **must** be reachable without using BGP, via IGP (e.g., OSPF) or because directly connected
    - stitches together the ASes in the path

BGP NEXT-HOP (v1)



What is NEXT-HOP at each router?

- 3d? Xa
- 3c? 3d
- 3a? 3d
- 2c? 3a
- 2a? 2c
- 1c? 2a
- 1b? 1c
- 1a? 1c

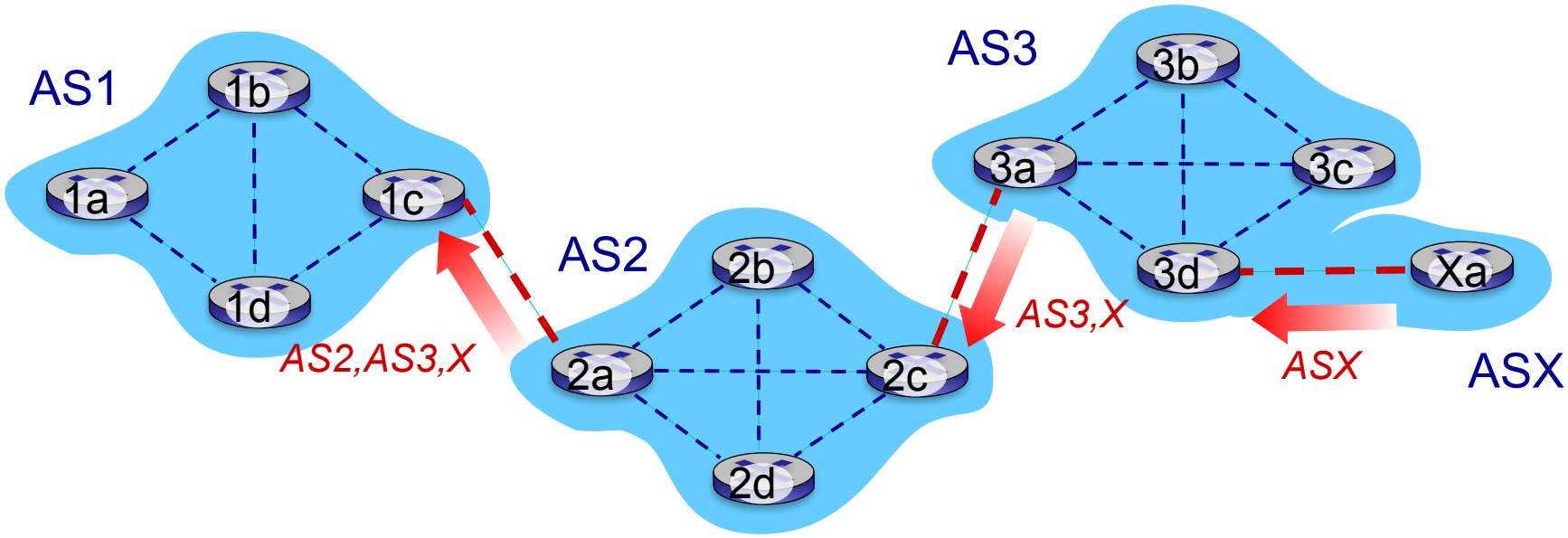
Day 20: BGP Routing Policy

CSEE 4119
Computer Networks
Ethan Katz-Bassett

COLUMBIA UNIVERSITY
IN THE CITY OF NEW YORK

Slides adapted from (and often identical to) slides from Kurose and Ross.

Recap: BGP NEXT-HOP (v1)

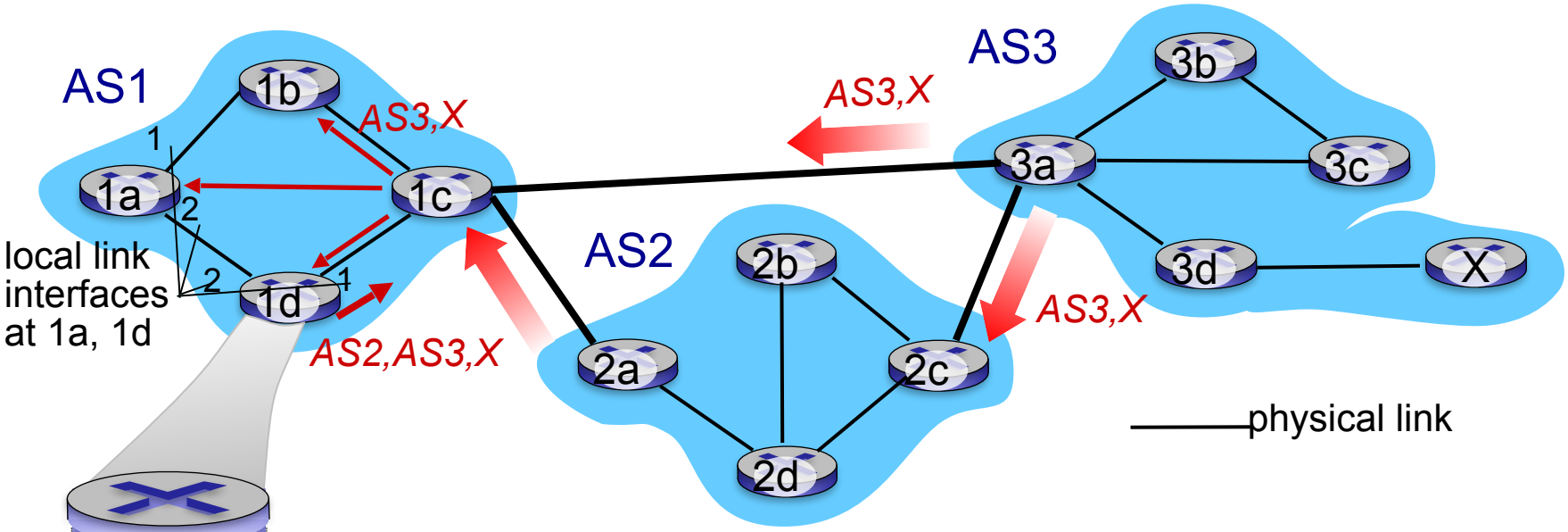


What is NEXT-HOP at each router?

- 3d? Xa
- 3c? 3d
- 3a? 3d
- 2c? 3a
- 2a? 2c
- 1c? 2a
- 1b? 1c
- 1a? 1c

BGP, OSPF, forwarding table entries

Q: how does router set forwarding table entry to distant prefix?

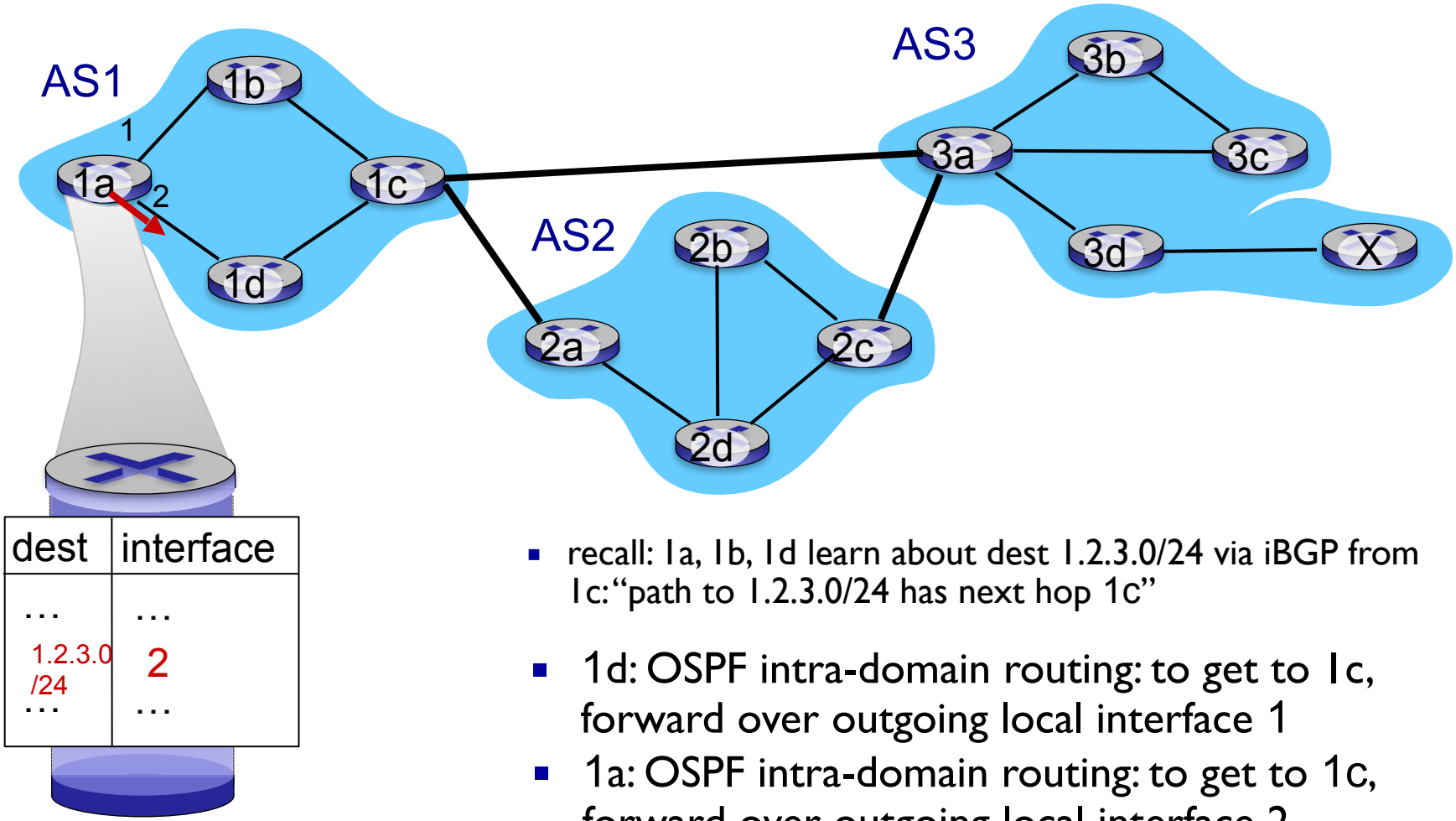


dest	interface
…	…
1.2.3.0 /24	1
…	…

- recall: 1a, 1b, 1d learn about dest 1.2.3.0/24 via iBGP from 1c: "path to 1.2.3.0/24 has next hop 1c"
- 1d: OSPF intra-domain routing: to get to 1c, forward over outgoing local interface 1

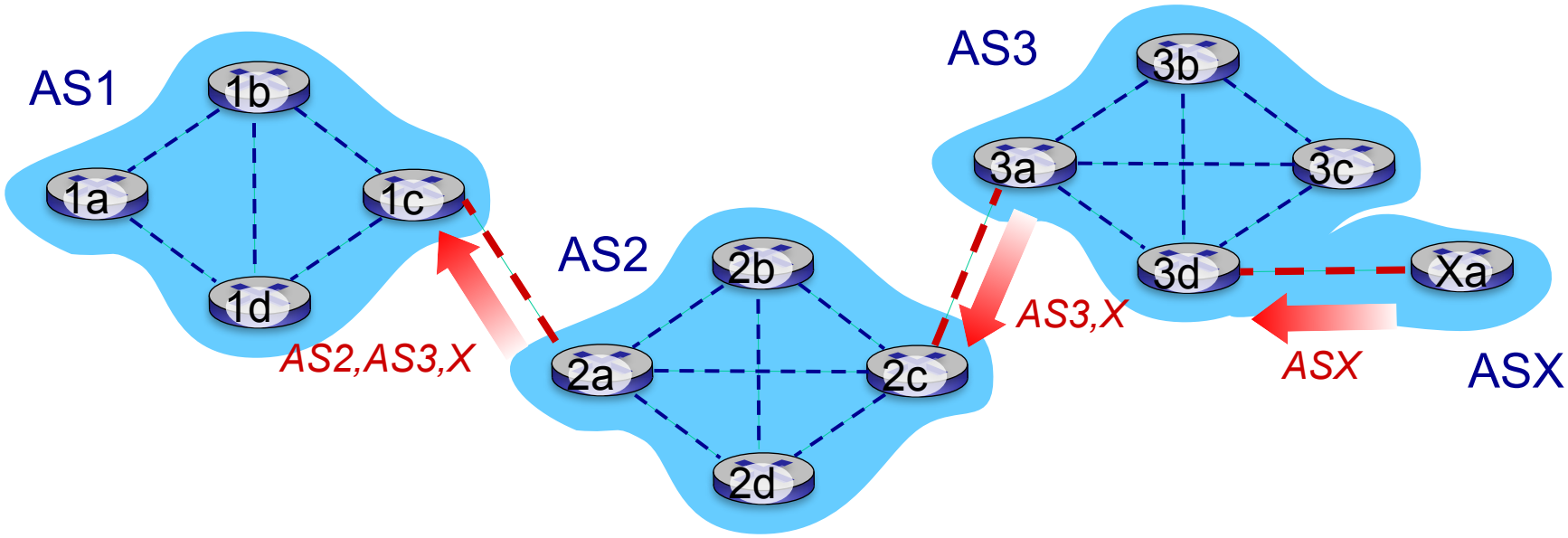
BGP, OSPF, forwarding table entries

Q: how does router set forwarding table entry to distant prefix?



- recall: 1a, 1b, 1d learn about dest 1.2.3.0/24 via iBGP from 1c: "path to 1.2.3.0/24 has next hop 1c"
- 1d: OSPF intra-domain routing: to get to 1c, forward over outgoing local interface 1
- 1a: OSPF intra-domain routing: to get to 1c, forward over outgoing local interface 2

BGP NEXT-HOP (v2)



What is NEXT-HOP at each router?

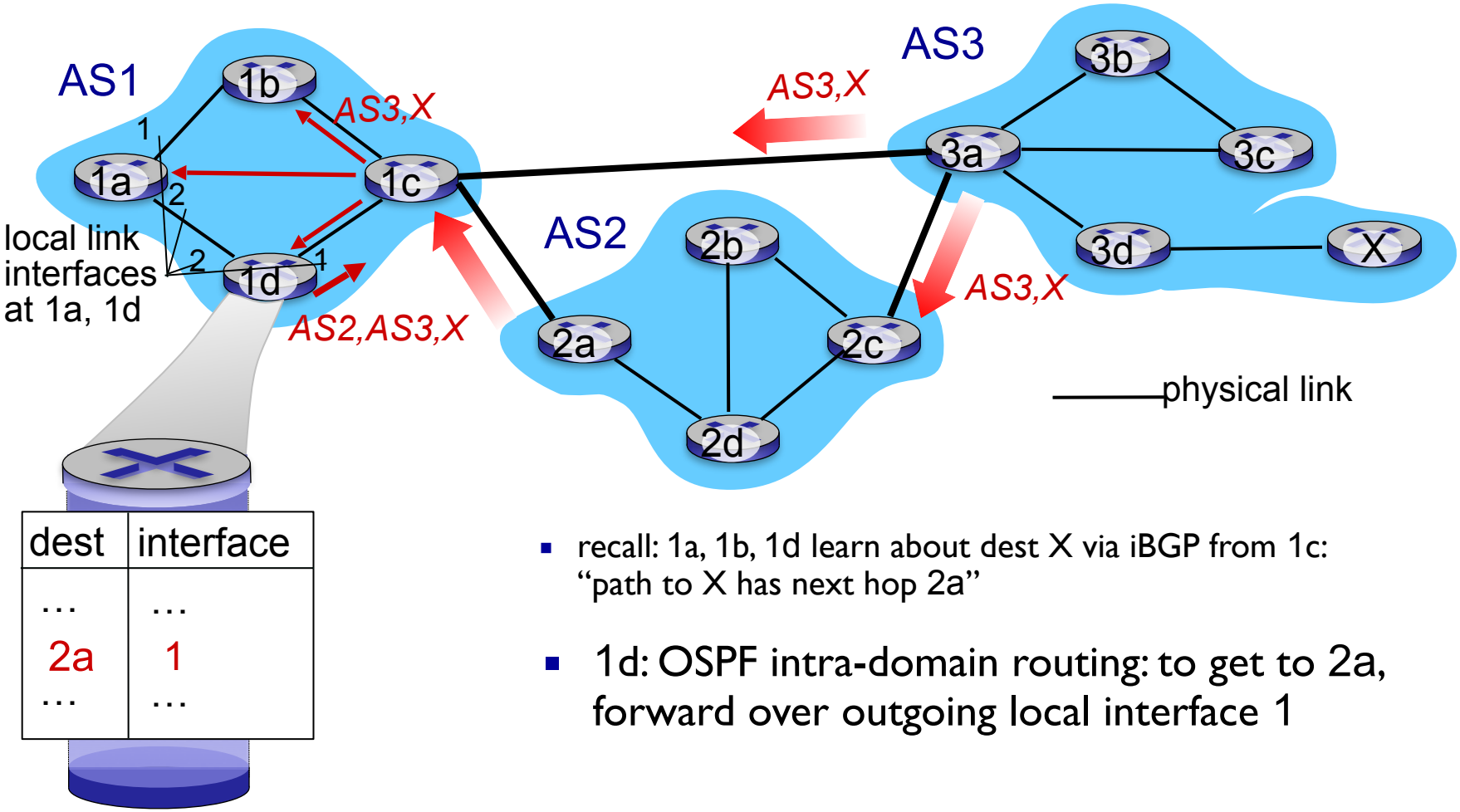
- 3d? Xa
- 3c? Xa
- 3a? Xa
- 2c? 3a
- 2a? 3a
- 1c? 2a
- 1b? 2a
- 1a? 2a

How does 1a know how to reach NEXT-HOP 2a?

- Not directly connected
- Not in its AS

BGP, OSPF, forwarding table entries

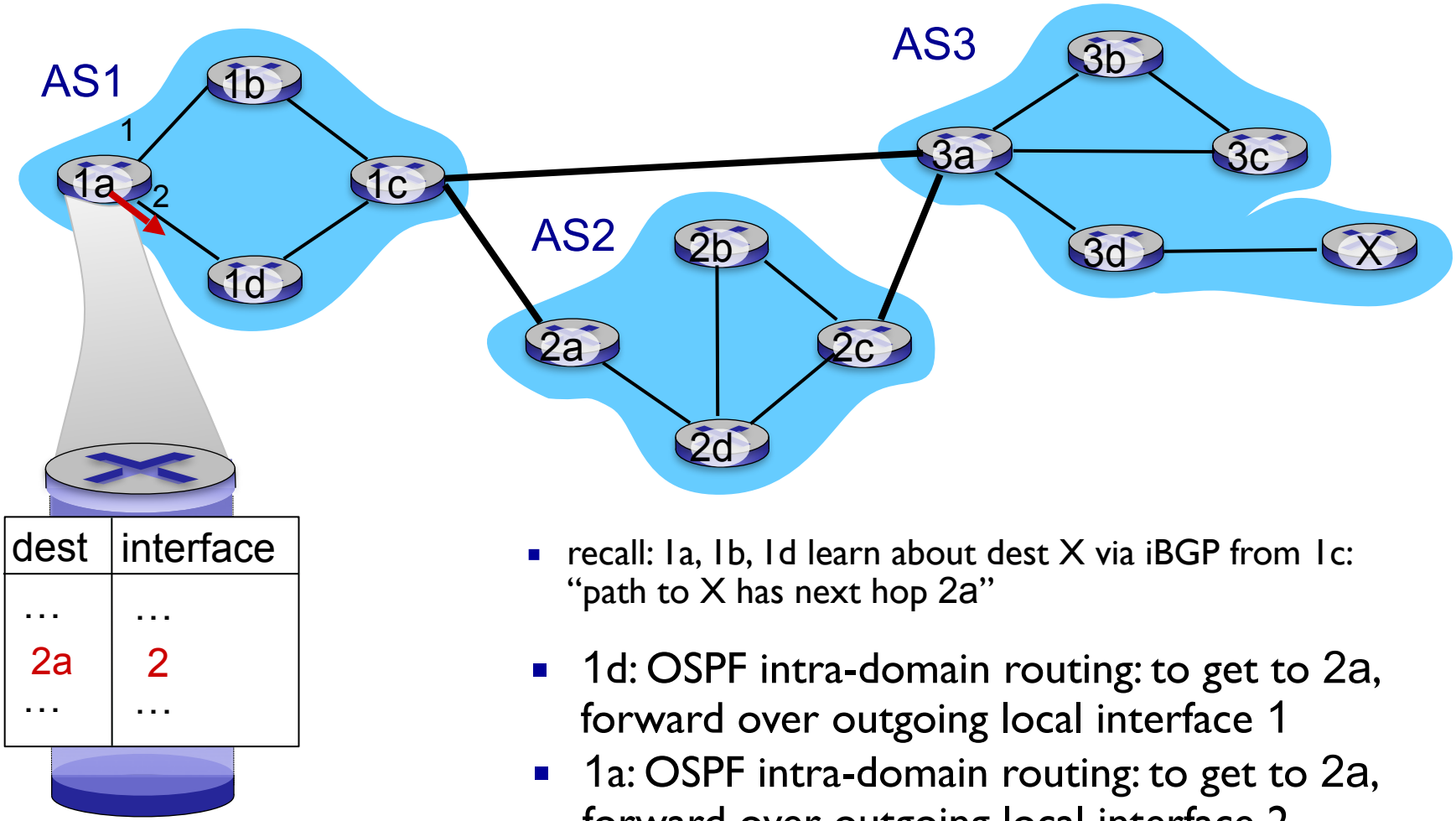
Q: how does router set forwarding table entry to distant prefix?



- recall: 1a, 1b, 1d learn about dest X via iBGP from 1c: "path to X has next hop 2a"
- 1d: OSPF intra-domain routing: to get to 2a, forward over outgoing local interface 1

BGP, OSPF, forwarding table entries

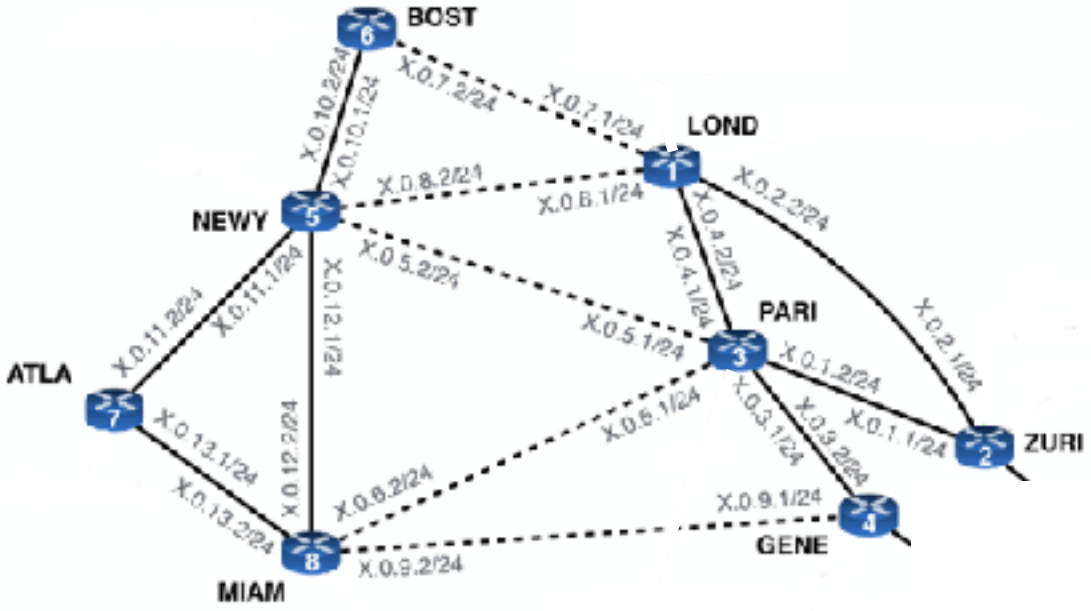
Q: how does router set forwarding table entry to distant prefix?



- recall: 1a, 1b, 1d learn about dest X via iBGP from 1c: "path to X has next hop 2a"
- 1d: OSPF intra-domain routing: to get to 2a, forward over outgoing local interface 1
- 1a: OSPF intra-domain routing: to get to 2a, forward over outgoing local interface 2

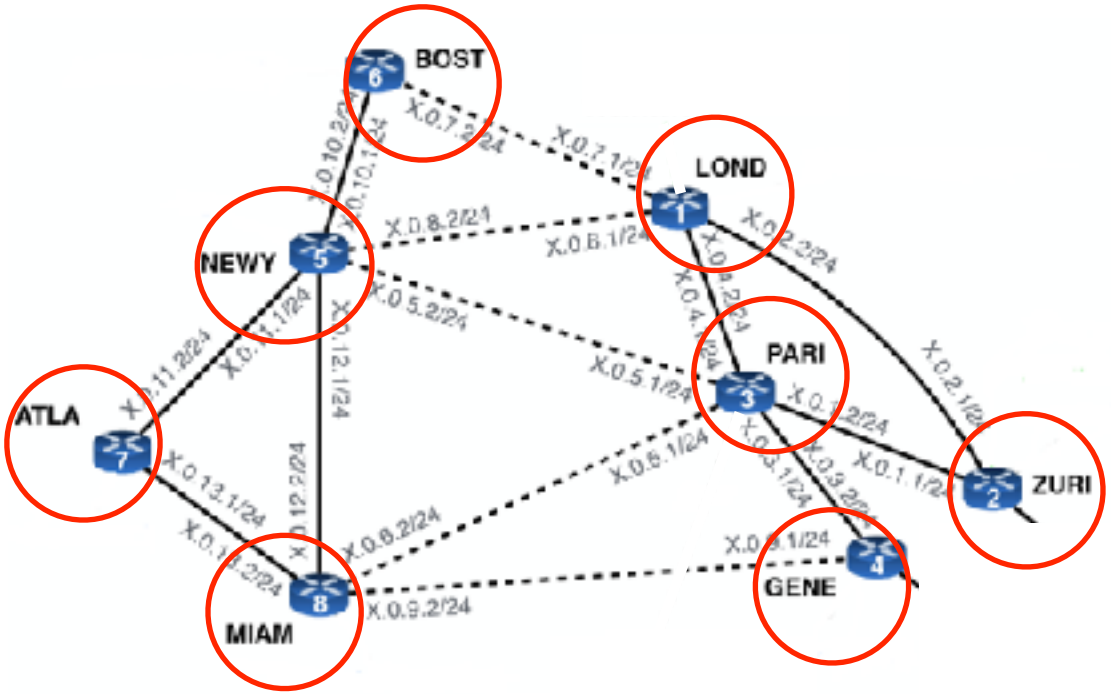
Project 2 Stage A Overview

- I like the project because it makes material real
- Each student runs a network



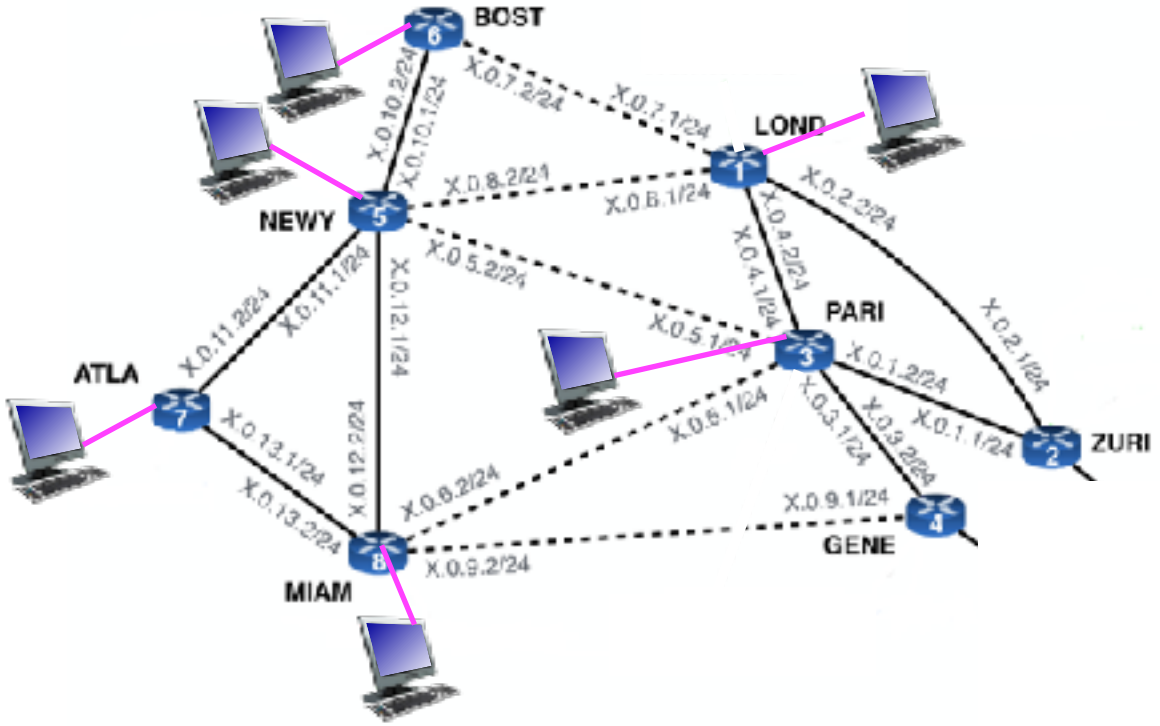
Project 2 Stage A Overview

- I like the project because it makes material real
- Each student runs a network
  - network is inside a VM, consists of multiple hosts/routers



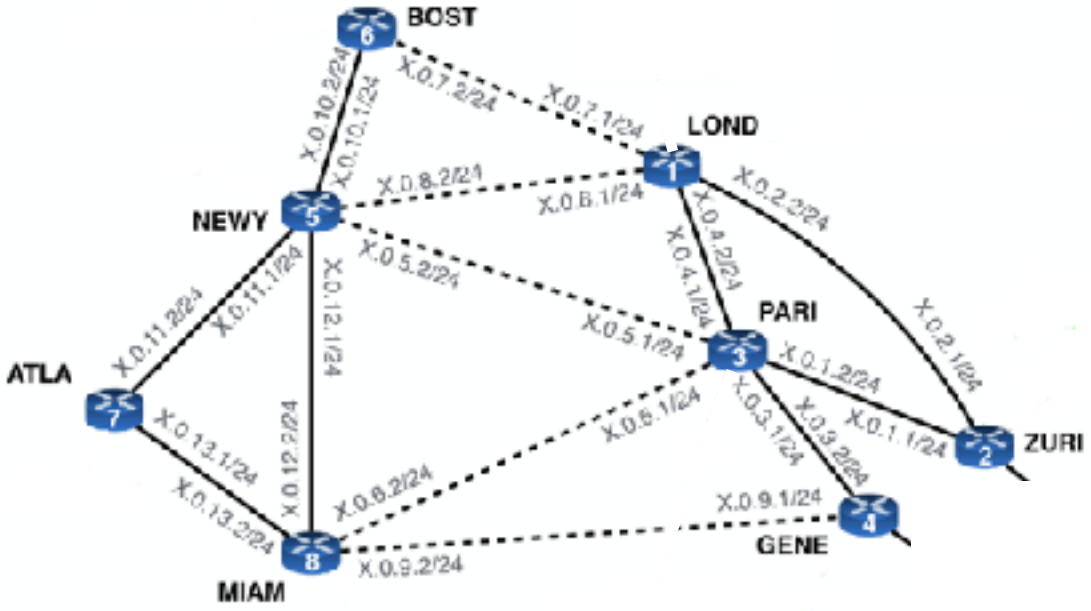
Project 2 Stage A Overview

- I like the project because it makes material real
- Each student runs a network
  - network is inside a VM, consists of multiple hosts/routers



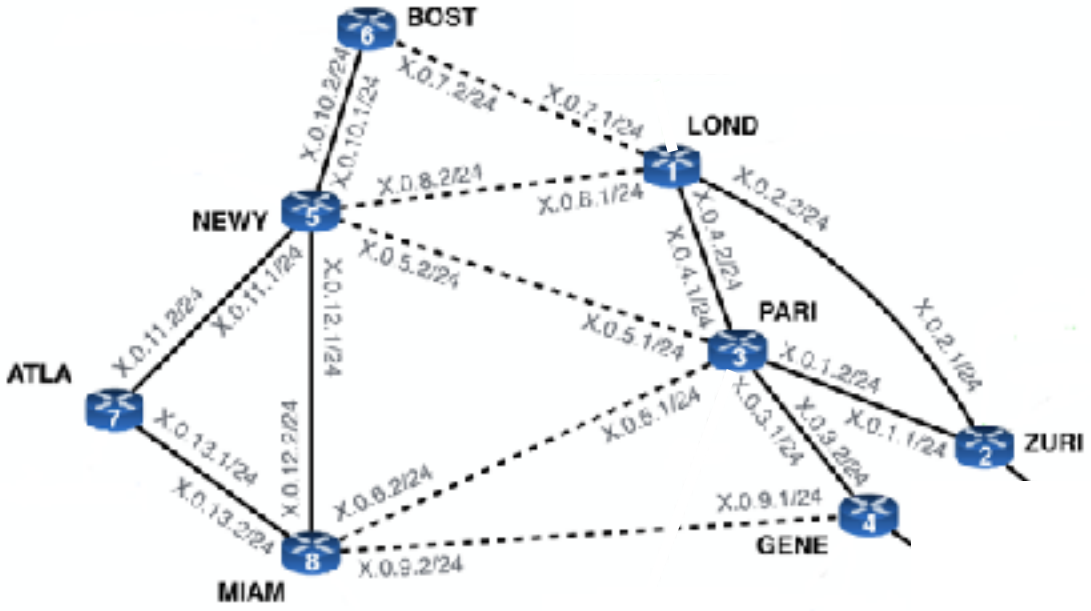
Project 2 Stage A Overview

- I like the project because it makes material real
- Each student runs a network
  - network is inside a VM, consists of multiple hosts/routers
    - `goto.sh` is how to "`ssh`" to particular host/router



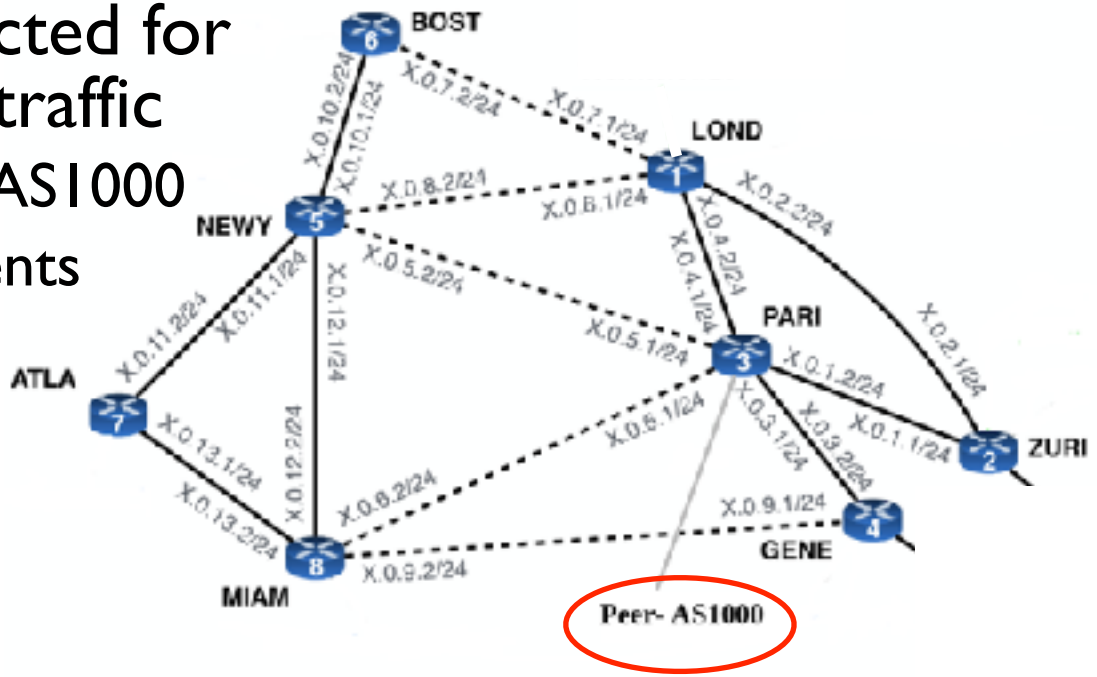
Project 2 Stage A Overview

- I like the project because it makes material real
- Each student runs a network
  - network is inside a VM, consists of multiple hosts/routers
    - `goto.sh` is how to "`ssh`" to particular host/router
  - If you are network `X`, you are AS`X` and control `X.0.0.0/8`
    - e.g., if we assign you AS`2`, you control `2.0.0.0/8`



Project 2 Stage A Overview

- I like the project because it makes material real
- Each student runs a network
  - network is inside a VM, consists of multiple hosts/routers
    - `goto.sh` is how to "`ssh`" to particular host/router
  - If you are network `X`, you are AS`X` and control `X.0.0.0/8`
    - e.g., if we assign you AS`2`, you control `2.0.0.0/8`
- VMs are connected for eBGP/internet traffic
  - Initially, to TA AS1000
  - Later, to students



- where are OSPF, eBGP, and iBGP used?

Project advice

- I like the project because it makes material real
- Each student runs a network
- VMs are connected for eBGP/internet traffic
- Advice
  - make sure you understand the task at hand, then learn the configuration to accomplish that task (we shared a *partial* tutorial)
  - you won't know how to do everything up front, so you'll have to try things out to see what works
    - engineering/computer science is about problem solving
    - save config outside VM regularly
    - before posting question on Piazza, try to solve (and describe what you tried), search for similar problems on Piazza, search online
    - help each other out on Piazza (with basics of navigating the system and FRR CLI, not with detailed answers to policy configuration)

Day 21: BGP Routing Policy

CSEE 4119
Computer Networks
Ethan Katz-Bassett

COLUMBIA UNIVERSITY
IN THE CITY OF NEW YORK

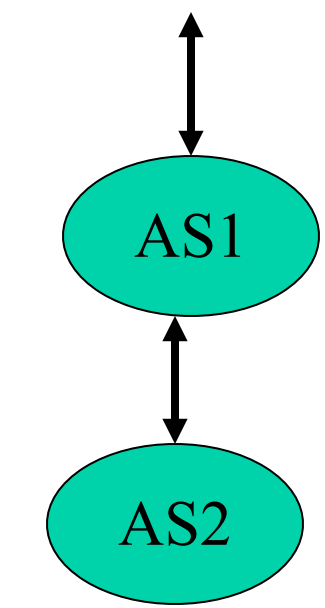
Slides adapted from (and often identical to) slides from Kurose and Ross.

Recap: BGP is policy-based

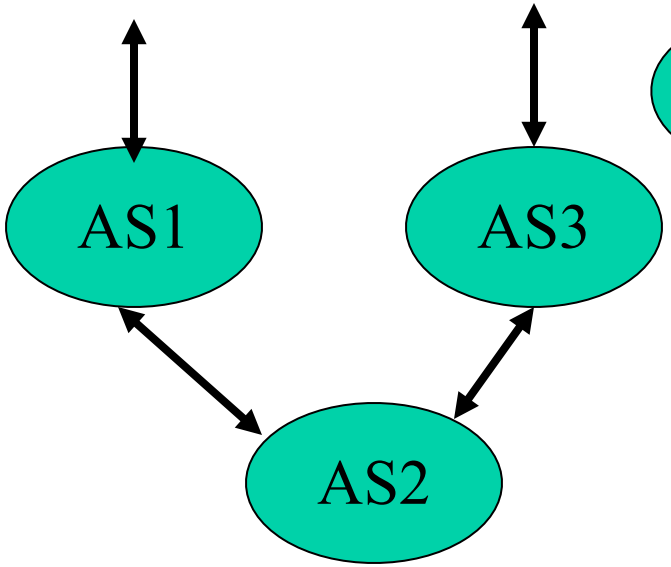
- *Policy-based routing:*
  - gateway receiving route advertisement uses *import policy* to accept/decline path (e.g., never route through AS Y).
  - *route selection policy* to pick best path when multiple exist
    - NOT NECESSARILY shortest path
  - *export policy* determines which (if any) neighboring ASes to advertise best path to
    - NOT NECESSARILY all paths to all neighbors
    - CAN be different to different neighbors

Why policy?

AS Types

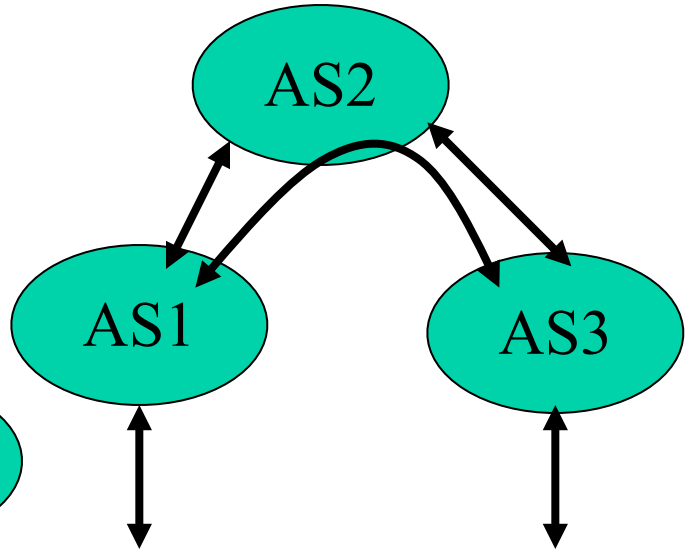


AS 2 is *Stub*



AS 2 is *Multi-homed*

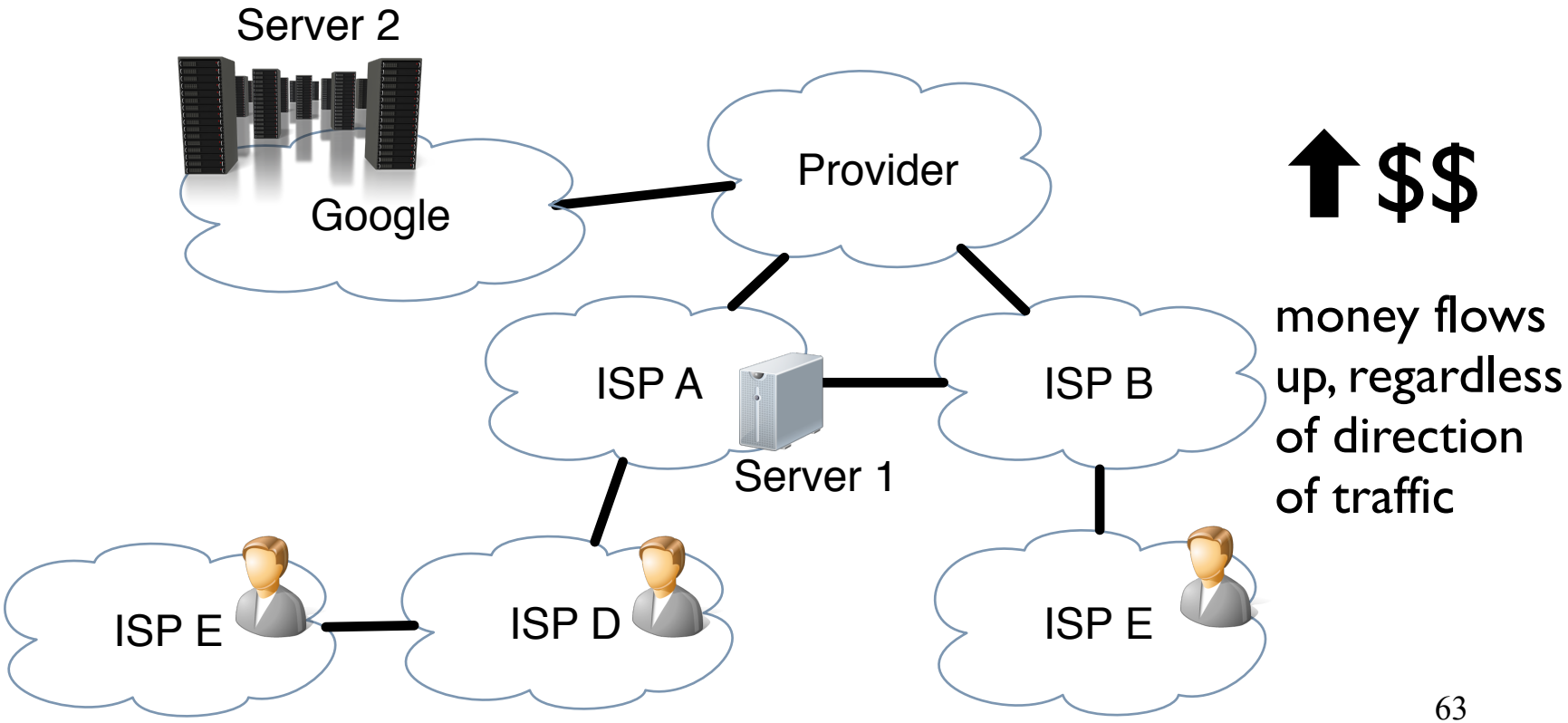
but there are other kinds of multihoming, too



AS 2 is *Transit AS*

AS relationships and policy

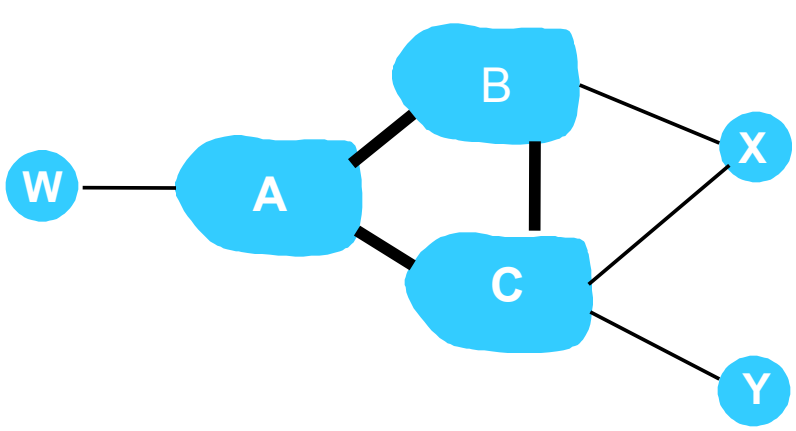
- Most common relationships
  - Customer-provider: customer pays provider to access Internet
  - Settlement-free peer: freely exchange traffic
  - Often depicted as provider above customer, peers side-by-side

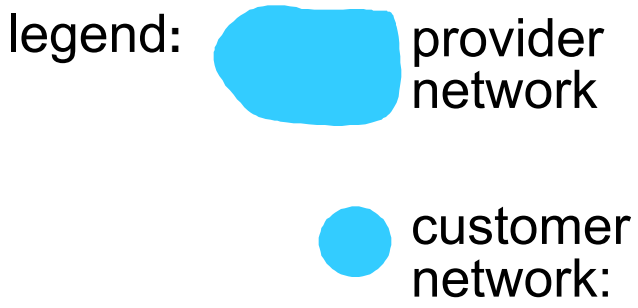


BGP is policy-based

- *Policy-based routing:*
  - gateway receiving route advertisement uses *import policy* to accept/decline path (e.g., never route through AS Y).
  - *route selection policy* to pick best path when multiple exist
  - ***export policy*** **determines which (if any) neighboring ASes to advertise best path to**
    - NOT NECESSARILY all paths to all neighbors
    - CAN be different to different neighbors

BGP: achieving policy via advertisements

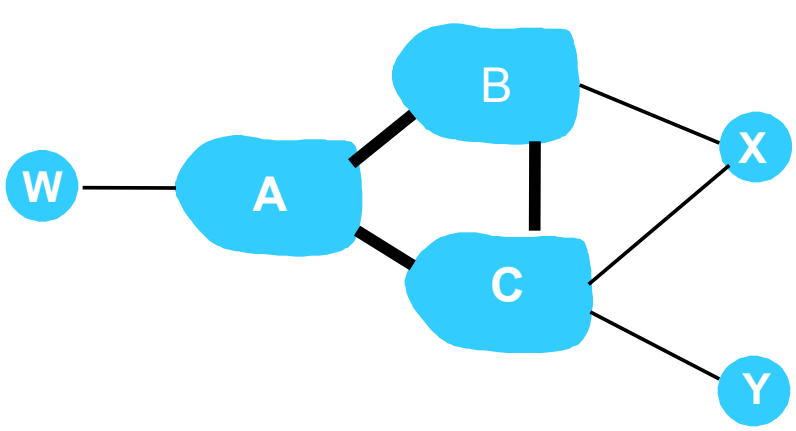


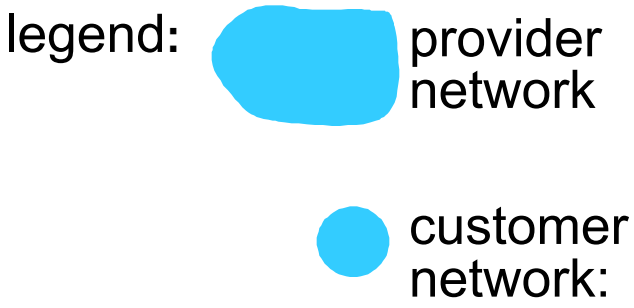


Suppose an ISP only wants to route traffic to/from its customer networks (does not want to carry transit traffic between other ISPs)

- A advertises path Aw to B and to C
- B *chooses not to advertise* BAw to C:
  - B gets no "revenue" for routing CBAw, since none of C, A, w are B's customers
  - C does not learn about CBAw path
- C will route CAw (not using B) to get to w

BGP: achieving policy via advertisements



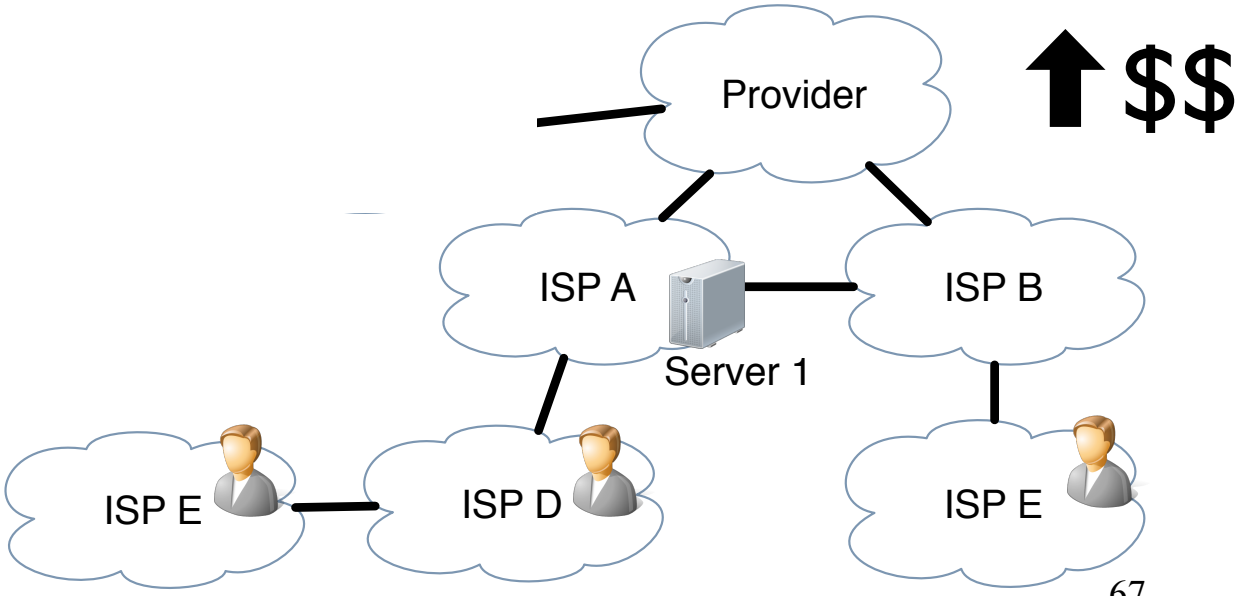


Suppose an ISP only wants to route traffic to/from its customer networks (does not want to carry transit traffic between other ISPs)

- A,B,C are *provider networks*
- X,W,Y are customer (of provider networks)
- X is *dual-homed:* attached to two networks
- *policy to enforce:* X does not want to route from B to C via X
  - .. so X will not advertise to B a route to C

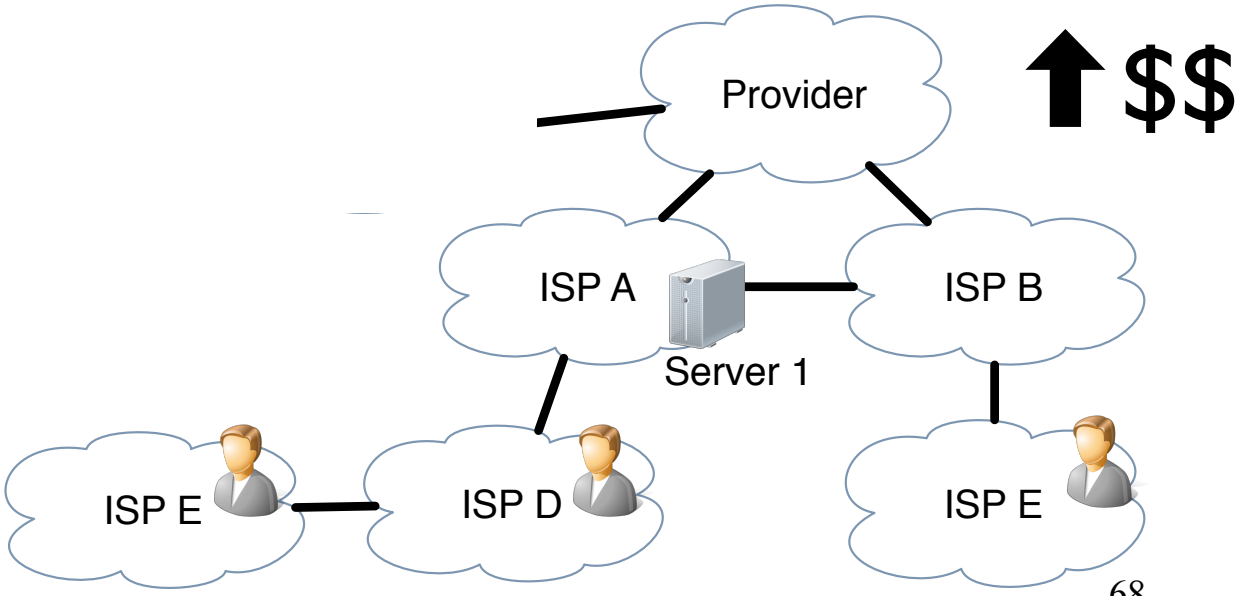
No Valley (common BGP policy)

- Customer-provider: customer pays provider to access Internet
- Settlement-free peer: freely exchange traffic
- Often depicted as provider above customer, peers side-by-side
- Policy: "No valley"
  - What: Don't transit traffic for provider/peer to another provider/peer
  - Why:



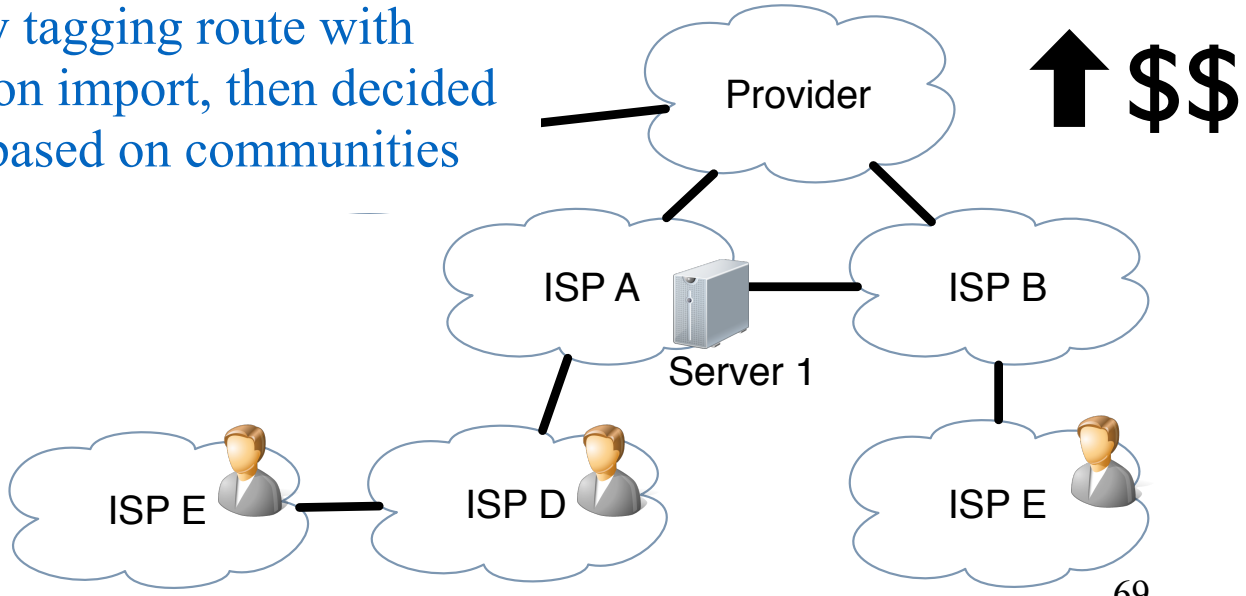
No Valley (common BGP policy)

- Customer-provider: customer pays provider to access Internet
- Settlement-free peer: freely exchange traffic
- Often depicted as provider above customer, peers side-by-side
- Policy: "No valley"
  - What: Don't transit traffic for provider/peer to another provider/peer
  - Why: you don't get paid or incur benefit, but have to haul the traffic
  - How to implement:



No Valley (common BGP policy)

- Customer-provider: customer pays provider to access Internet
- Settlement-free peer: freely exchange traffic
- Often depicted as provider above customer, peers side-by-side
- Policy: "No valley"
  - What: Don't transit traffic for provider/peer to another provider/peer
  - Why: you don't get paid or incur benefit, but have to haul the traffic
  - How to implement: Don't announce (export) route learned from peer/provider to another peer/provider
    Often achieved by tagging route with BGP community on import, then decided who to export to based on communities



BGP route selection

- *Policy-based routing:*
  - gateway receiving route advertisement uses *import policy* to accept/decline path (e.g., never route through AS Y).
  - *route selection policy* to pick best path when multiple exist
  - *export policy* determines which (if any) neighboring ASes to advertise best path to

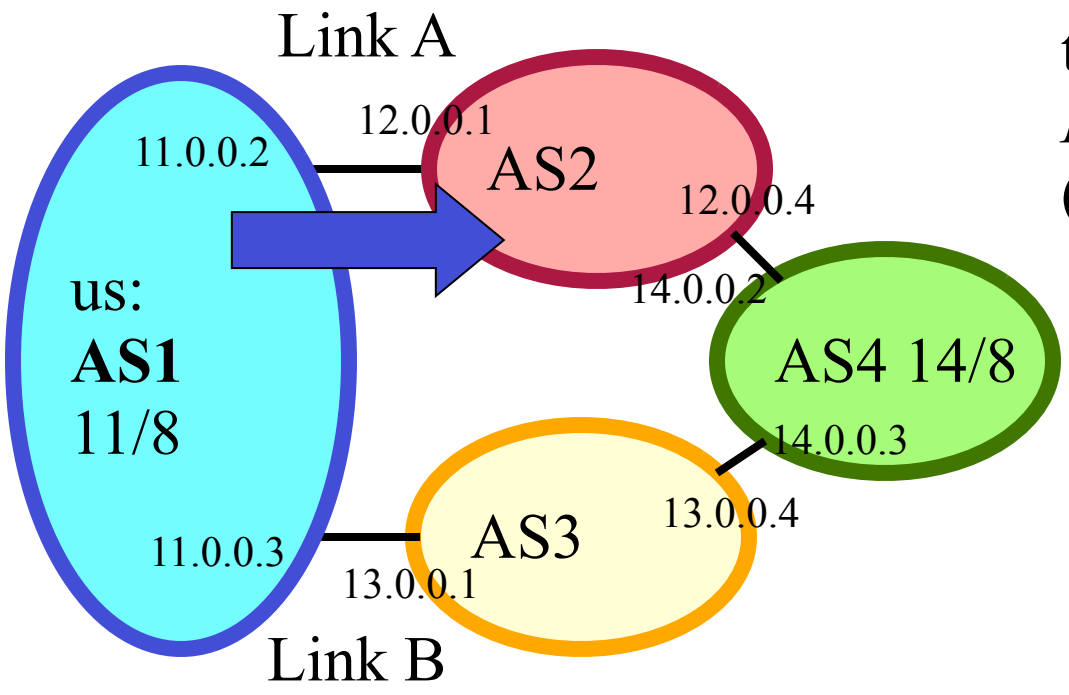
BGP route selection

- *Policy-based routing:*
  - gateway receiving route advertisement uses *import policy* to accept/decline path (e.g., never route through AS Y).
  - *route selection policy* to pick best path when multiple exist
  - *export policy* determines which (if any) neighboring ASes to advertise best path to
- selects route based on:
  1. **local preference value attribute: policy decision**
  2. shortest AS-PATH
  3. (lowest multi-exit discriminator [MED])
  4. closest NEXT-HOP router: hot potato routing
  5. additional criteria

Policy 1: LOCAL-PREF

- From local configuration
  - affects *your* AS only
  - (does not propagate to others)
  - can influence any prefixes
- Pick which path to prefer for a prefix for **outgoing** traffic
- Rule: *BGP prefers paths with higher LOCAL-PREF*

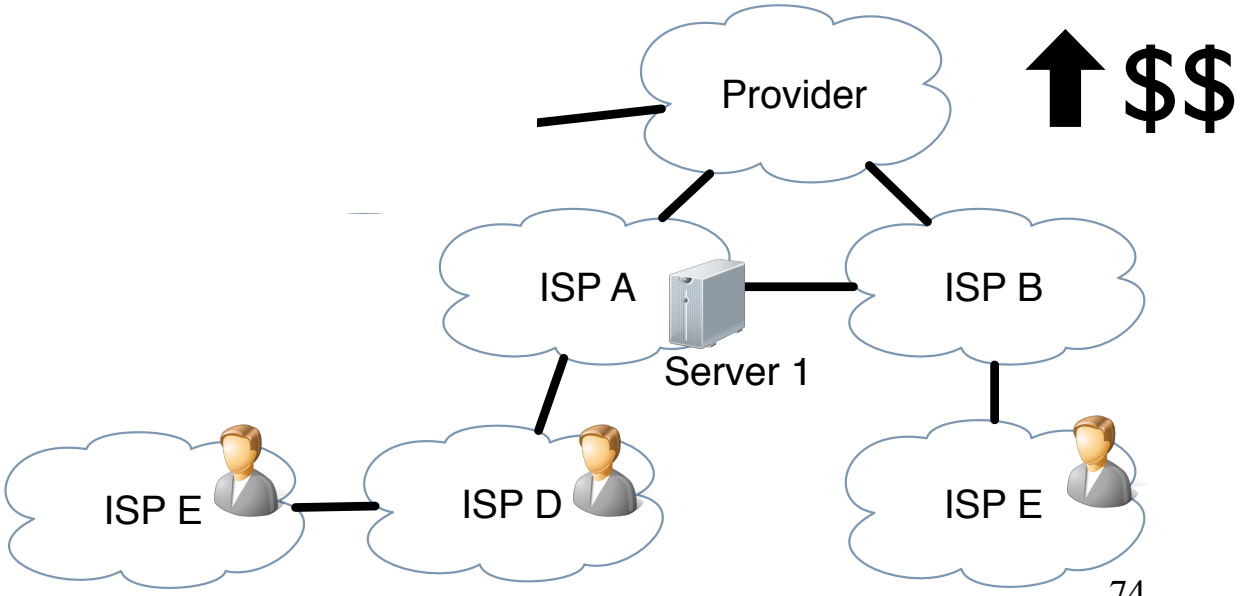
LOCAL-PREF Example 1



You are AS1;
two links A&B to AS 2/3
to reach AS4 and prefix 14/8.
Policy goal: use only link A
(unless it's down)

Prefer customer (common BGP policy)

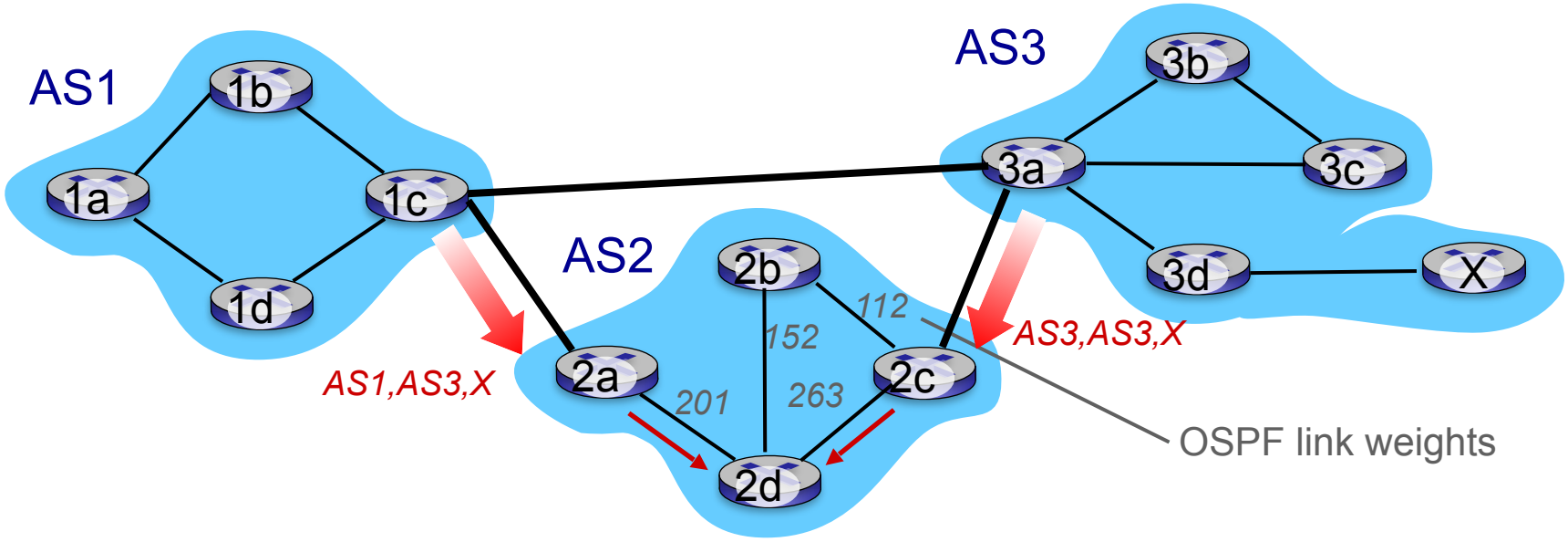
- Customer-provider: customer pays provider to access Internet
- Settlement-free peer: freely exchange traffic
- Often depicted as provider above customer, peers side-by-side
- Policy: "Prefer customer"
  - What: Prefer customer route over peer route, peer route over provider
  - Why:



BGP route selection

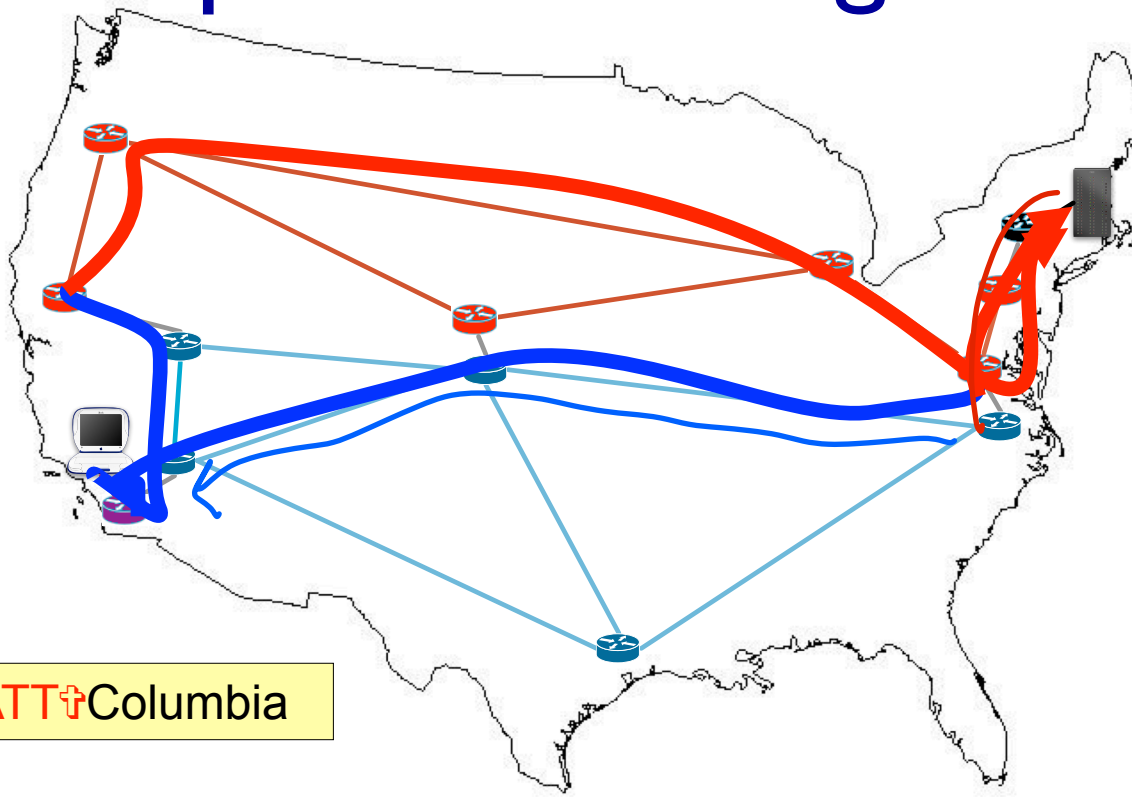
- *Policy-based routing:*
  - gateway receiving route advertisement uses *import policy* to accept/decline path (e.g., never route through AS Y).
  - *route selection policy* to pick best path when multiple exist
  - AS *export policy* also determines whether to advertise best path to other other neighboring ASes
- selects route based on:
  1. local preference value attribute: policy decision
  2. shortest AS-PATH
  3. (lowest multi-exit discriminator [MED])
  4. **closest NEXT-HOP router: hot potato routing**
  5. additional criteria

Hot Potato Routing



- 2d learns (via iBGP) it can route to X via 2a or 2c
- *hot potato (aka early exit) routing:* 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!
  - this example shows 2d choosing between different AS paths; common variant: AS2 connects to AS1 in multiple locations, 2d chooses which exit to use

Ex 2: hot potato routing



USC †L3†ATT†Columbia

- Each network decides how to route across its domain and where to hand traffic to next network
- End-to-end depends on inter-network & intra-network
  - Performance and availability stem from these decisions

BGP communities

- an optional BGP attribute
  - think of as labels that can be attached to any path to pass information between routers (same or different AS)
  - Can be used to inform localpref, whether to export path, etc
- 32 bit number with no innate meaning (with some exceptions)
  - convention is that first 16 bits indicate which AS defines the meaning, to pass info to/from that AS. Examples from NTT (AS2914):
    - 2914:4429 "tell NTT to not advertise to peers in Asia"
    - 2914:1009 "NTT informs you it learned the route in NYC"
  - operators configure routers to add and act on communities
    - We will add FRR details to tutorial doc for Stage C

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

Uploading course materials to sites such as CourseHero, Chegg or Github is academic misconduct at Columbia (see pg 10 of Columbia guide).

Day 23: Shortest Path Routing

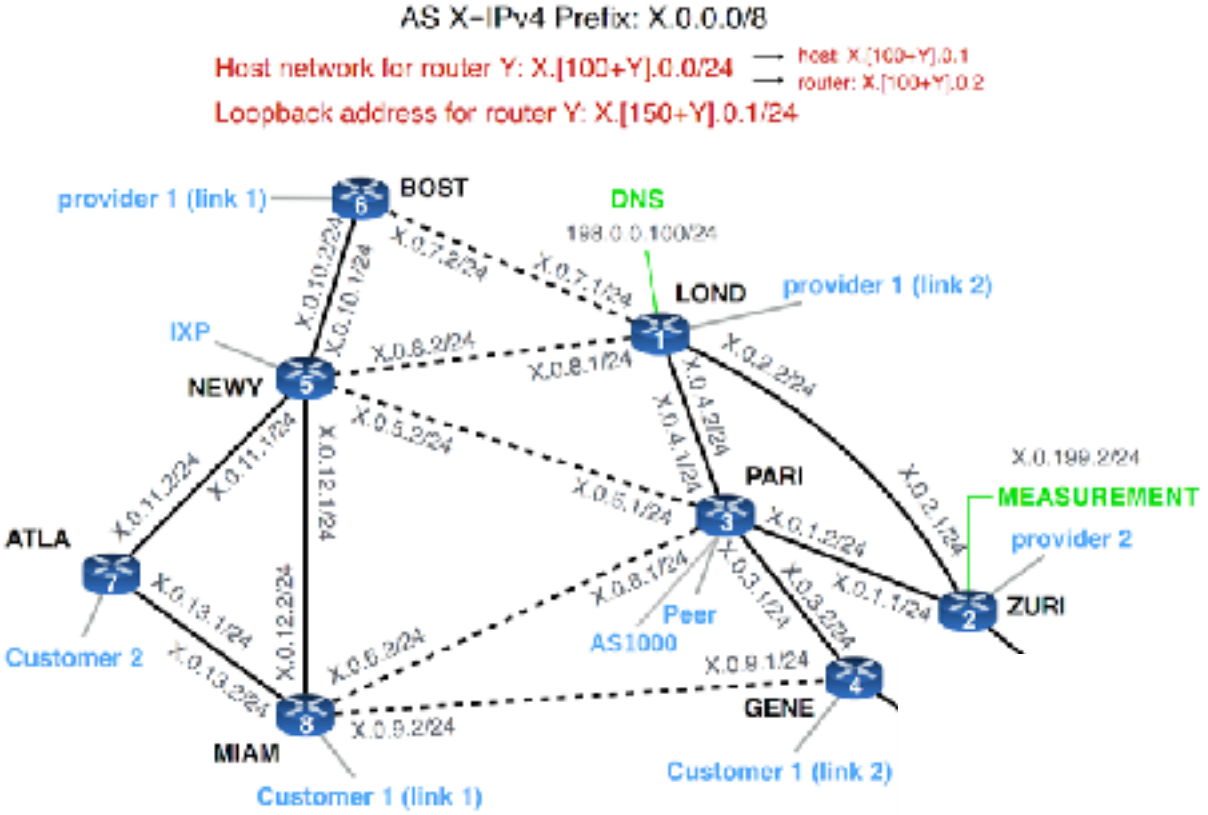
CSEE 4119
Computer Networks
Ethan Katz-Bassett

COLUMBIA UNIVERSITY
IN THE CITY OF NEW YORK

Slides adapted from (and often identical to) slides from Kurose and Ross.

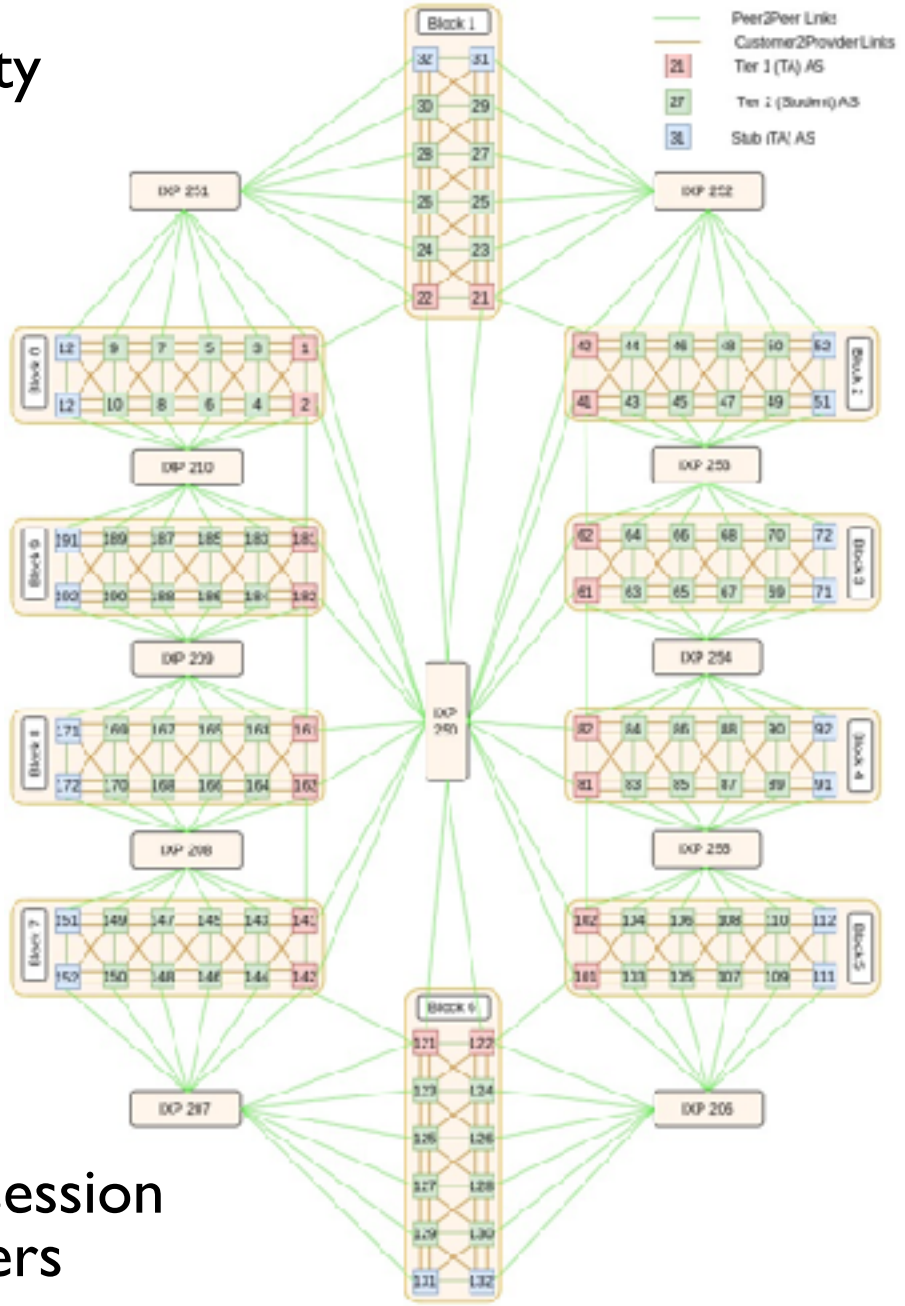
Project 2 Stage B Overview

- Each router connects to another AS, run either by another student or by the TA
  - PARI connects to AS1000 ***and*** another AS
- Goal of Stage B: global connectivity



Project 2 Stage B Overview

- Goal of Stage B: global connectivity
- Enabled via pairwise eBGP sessions like the one from Preliminary Stage
- Due Dec. 4, but should not require you to learn anything new, just coordinate with others and replicate configuration equivalent to Preliminary Stage
- Each node in this figure is an AS like one on previous slide. Link between two ASes means eBGP session between 1 or more pairs of routers



Chapter 5: outline

5.1 introduction

5.2 routing protocols

- link state
- distance vector

5.3 intra-AS routing in the Internet: OSPF

5.4 routing among the ISPs: BGP

5.5 The SDN control plane

5.6 ICMP: The Internet Control Message Protocol

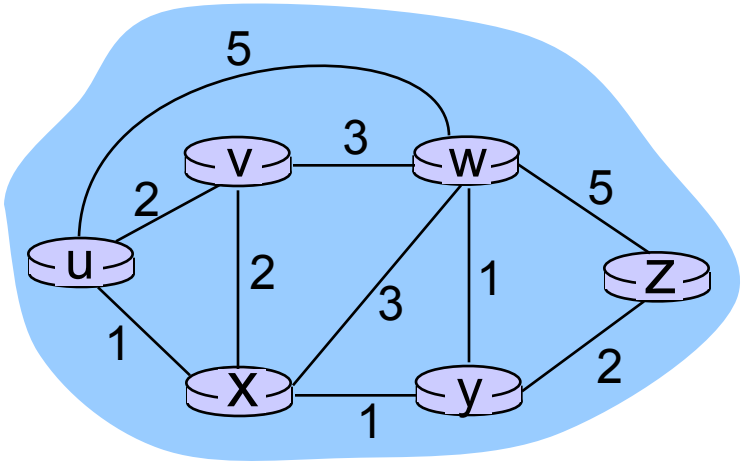
5.7 Network management and SNMP

Recap:Routing protocols

Routing protocol goal: determine "good" paths (equivalently, routes), from sending hosts to receiving host, through network of routers

- path: sequence of routers packets will traverse in going from given initial source host to given final destination host
- "good": least "cost", "fastest", "least congested"
- routing: a "top-10" networking challenge!

Recap: Graph abstraction of the network



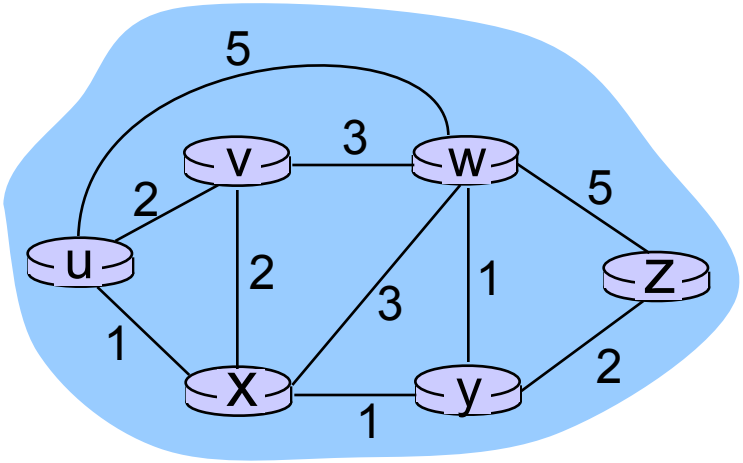
graph: G = (N,E)

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) }

aside: graph abstraction is useful in other network contexts, e.g., P2P, where *N* is set of peers and *E* is set of TCP connections

Recap: Graph abstraction: costs



c(x,x') = cost of link (x,x')
e.g., c(w,z) = 5

cost could always be 1,
based on physical length, cost,
inversely related to bandwidth,
inversely 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

Chapter 5: outline

5.1 introduction

5.2 routing protocols

- link state
- distance vector

5.3 intra-AS routing in the Internet: OSPF

5.4 routing among the ISPs: BGP

5.5 The SDN control plane

5.6 ICMP: The Internet Control Message Protocol

5.7 Network management and SNMP

A link-state routing algorithm

Dijkstra's algorithm

- complete 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* destinations (specifically the *k* with the smallest costs)

notation:

- c(x,y): link cost from x to y; ∞ if not direct neighbors
- D(v): current value of cost of path from source to destination v
- p(v): predecessor neighbor along path from source to v
- N': set of nodes whose least cost path definitively known

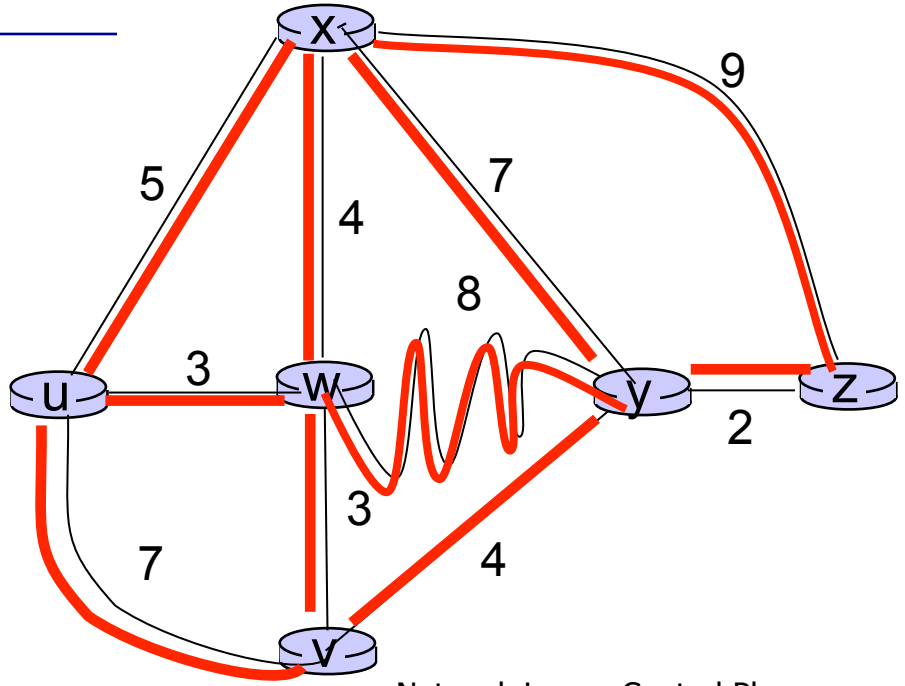
Dijsktra's algorithm

```
1  Initialization:
2    N' = {u} /* u is source node */
3    for all nodes v
4      if v adjacent to u
5          then D(v) = c(u,v)
6      else D(v) = ∞
7
8  Loop
9     find w not in N' such that D(w) is a minimum
10    add w to N'
11    update D(v) for all v adjacent to w and not in N' :
12       D(v) = min( D(v), D(w) + c(w,v) )
13    /* new cost to v is either old cost to v or known
14     shortest path cost to w plus cost from w to v */
15 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,v	14,x
4	uwxvy					12,y
5	uwxvyz					

D(n) = c(u,n)
D(n) = min(D(n), 3 + c(w,n))
D(n) = min(D(n), 5 + c(x,n))
D(n) = min(D(n), 6 + c(v,n))
D(n) = min(D(n), 10 + c(y,n))

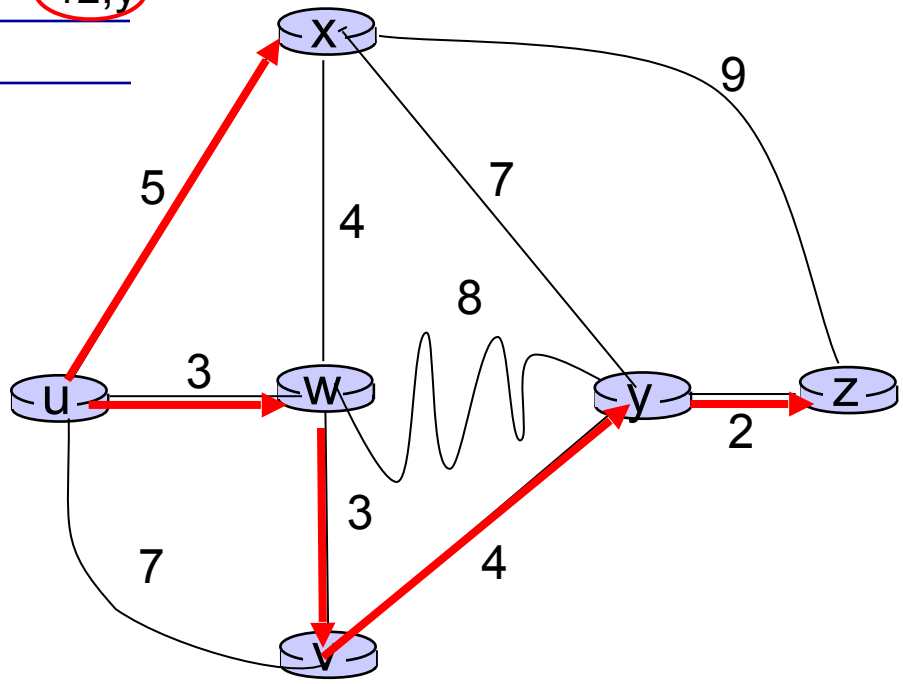


Dijkstra's algorithm: example

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

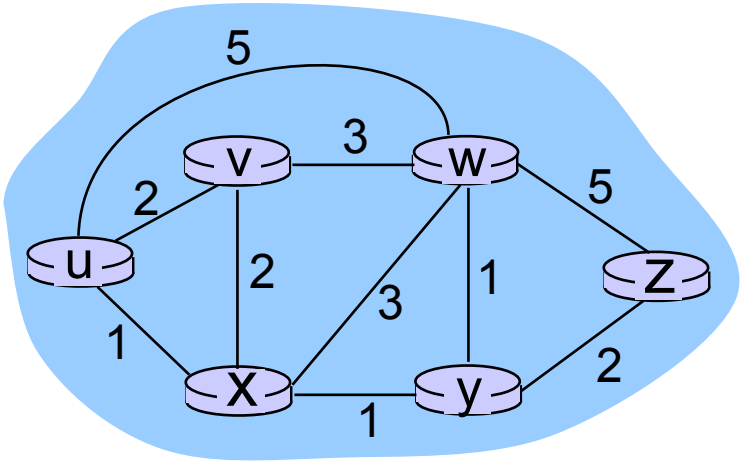
notes:

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



Dijkstra's algorithm: another example

Step	N'	D(v),p(v)	D(w),p(w)	D(x),p(x)	D(y),p(y)	D(z),p(z)
0	u	2,u	5,u	1,u	∞	∞
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					

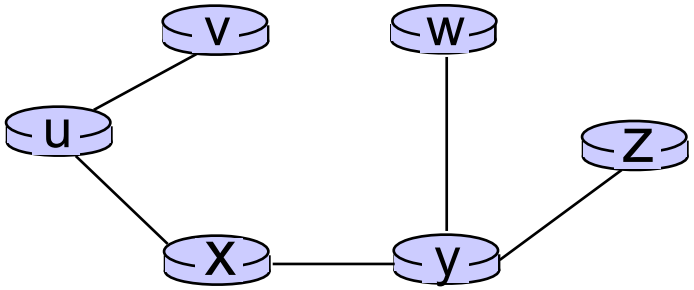


* Check out the online interactive exercises for more examples: http://gaia.cs.umass.edu/kurose_ross/interactive/

Dijkstra's algorithm: another example (2)

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					

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)

* Check out the online interactive exercises for more examples: http://gaia.cs.umass.edu/kurose_ross/interactive/

Dijkstra's algorithm: complexity

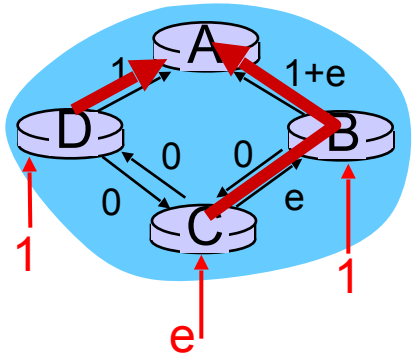
algorithm complexity: n nodes

- each iteration: need to check all nodes not in N to find one with minimum cost
  - first iteration: n
  - second iteration: n-1
  - …
- $n(n+1)/2$ comparisons: $O(n^2)$
- more efficient implementations possible: O(nlogn)
  - using a heap to find the minimum cost node

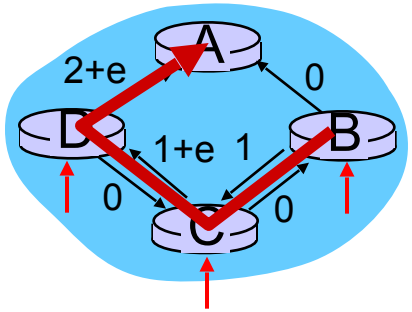
Dijkstra's algorithm: pathological case

oscillations possible:

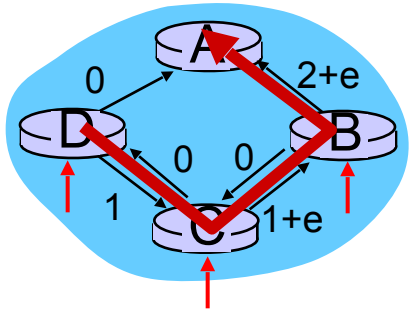
- Suppose link cost equals amount of carried traffic
  - So cost of a link may differ in the two directions
- Scenario
  - D is sending one unit of traffic to A
  - B is sending one unit of traffic to A
  - C is sending e units of traffic to A
- Possible solutions
  - Not allow link costs that depend on amount of traffic (unacceptable?)
  - Make sure nodes don't all run algorithm at the same time



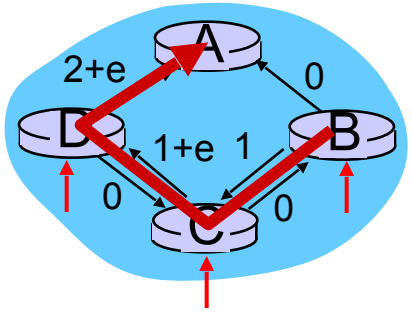
initially



given these costs, find new routing…. resulting in new costs



given these costs, find new routing…. resulting in new costs



given these costs, find new routing…. resulting in new costs

Chapter 5: outline

5.1 introduction

5.2 routing protocols

- link state
- distance vector

5.3 intra-AS routing in the Internet: OSPF

5.4 routing among the ISPs: BGP

5.5 The SDN control plane

5.6 ICMP: The Internet Control Message Protocol

5.7 Network management and SNMP

Distance vector algorithm

Approach is distributed, iterative, asynchronous

Bellman-Ford equation (dynamic programming)

let

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

then

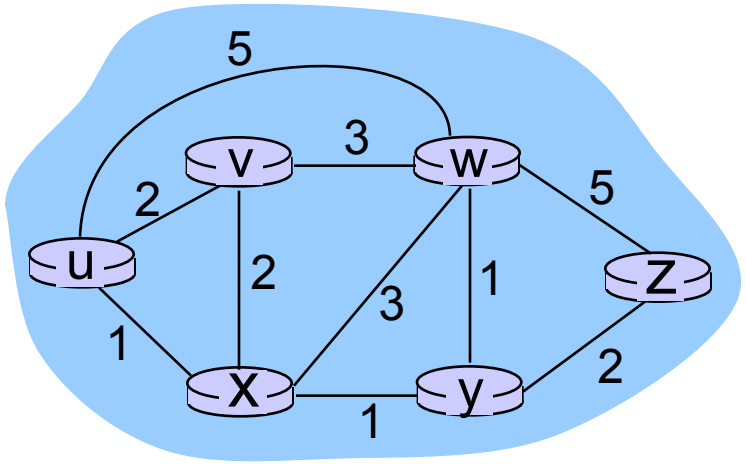
$$d_x(y) = \min_v \{c(x,v) + d_v(y)\}$$

v: *min* taken over all neighbors v of x

$c(x,v)$: cost to neighbor v

$d_v(y)$: cost from neighbor v to destination y

Bellman-Ford example



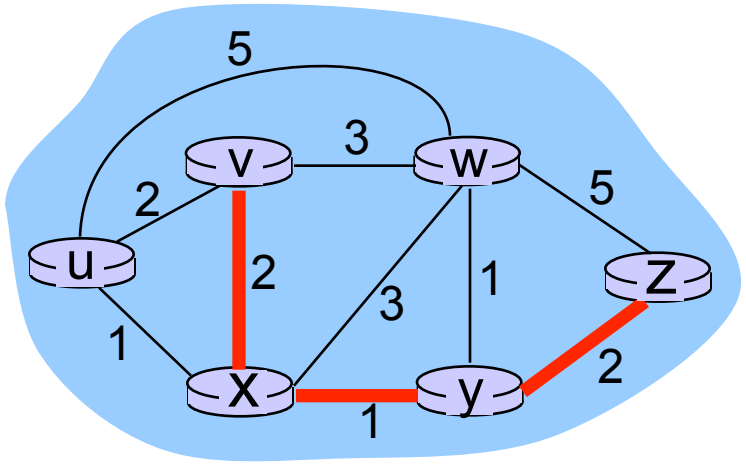
$d_v(z) = 5,\ d_x(z) = 3,\ d_w(z) = 3$

B-F equation says:

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

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

Bellman-Ford example



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

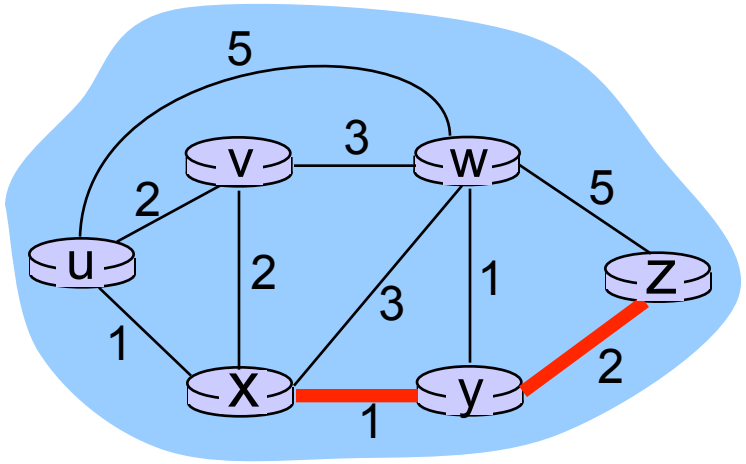
B-F equation says:

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

$$= \min \{2 + 5,\ 1 + 3,\ 5 + 3\} = 4$$

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

Bellman-Ford example



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

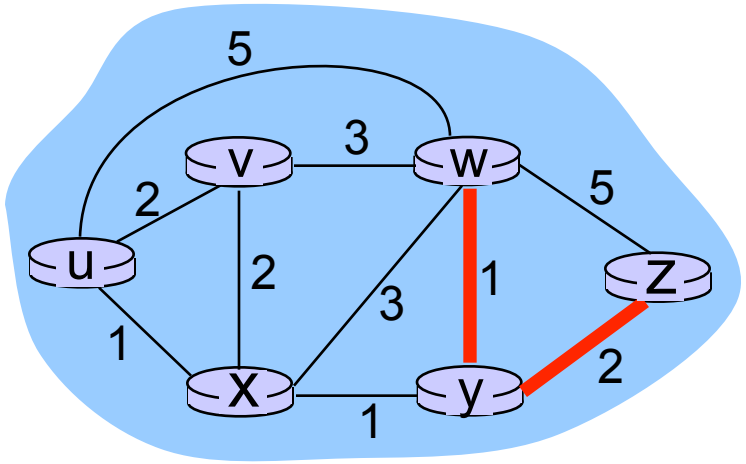
B-F equation says:

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

$$= \min \{2 + 5,\ 1 + 3,\ 5 + 3\} = 4$$

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

Bellman-Ford example



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

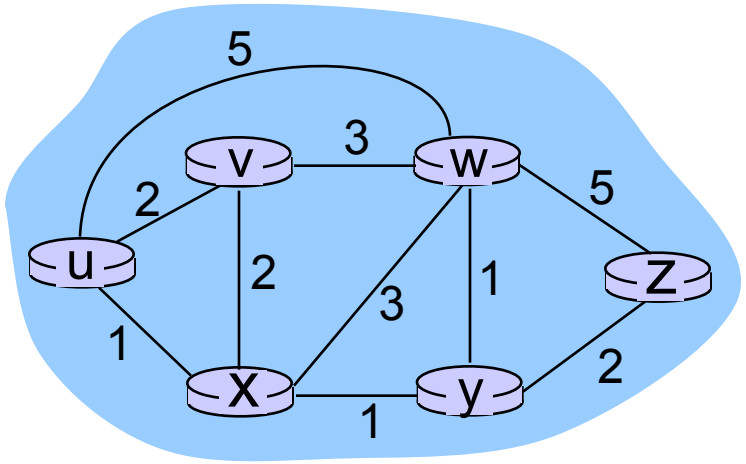
B-F equation says:

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

$$= \min \{2 + 5,\ 1 + 3,\ 5 + 3\} = 4$$

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

Bellman-Ford example



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

B-F equation says:

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

$$= \min \{2 + 5,\ 1 + 3,\ 5 + 3\} = 4$$

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

Distance vector algorithm (1)

- $D_x(y)$ = 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 (2)

key idea:

- from time-to-time, each node sends its own distance vector estimate to neighbors
- when x receives new DV estimate from neighbor, it updates its own DV using B-F equation:

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

- under minor natural assumptions, $D_x(y)$ *converges to the actual least cost* $d_x(y)$

Distance vector algorithm (3)

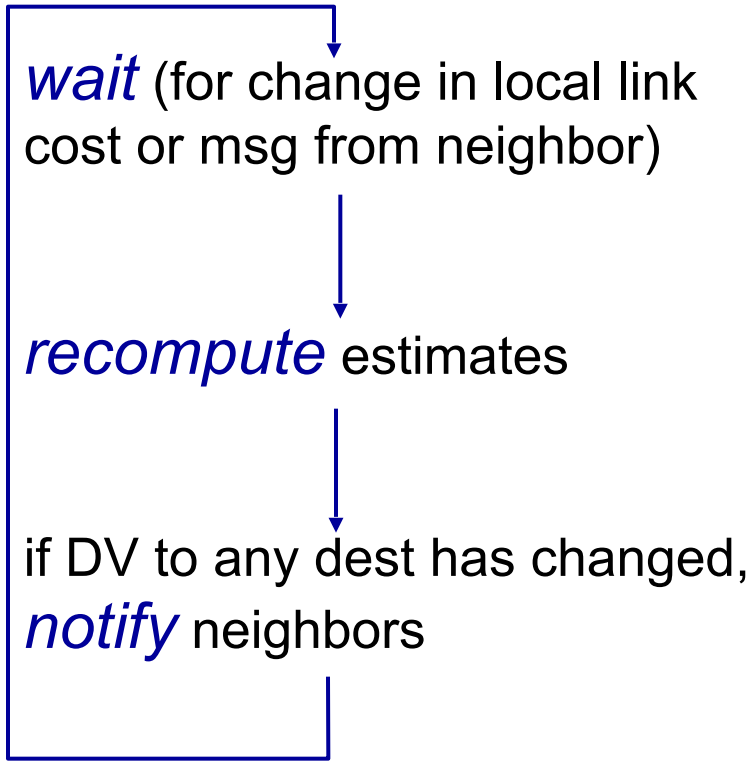
iterative, asynchronous: local iteration triggered by:

- local link cost change
- DV update message from neighbor

distributed:

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

each node:



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

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

node x table

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

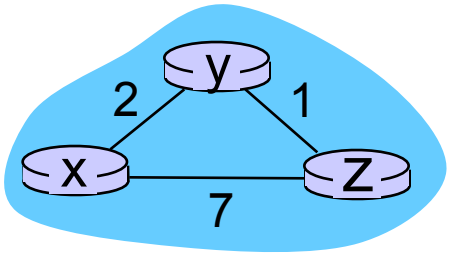
from \ cost to	x	y	z
x	0	2	3
y	2	0	1
z	7	1	0

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

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

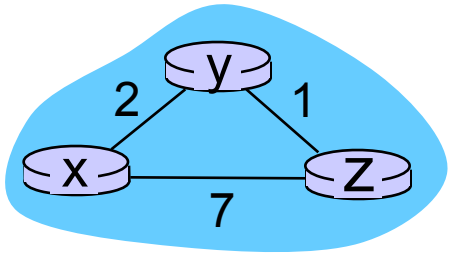
node y table

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

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

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

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

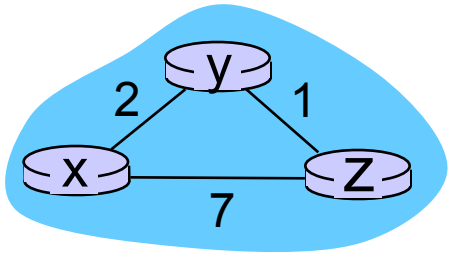
from \ cost to	x	y	z
x	∞	∞	∞
y	**2**	**0**	**1**
z	∞	∞	∞

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

node z table

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

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

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

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

node y table

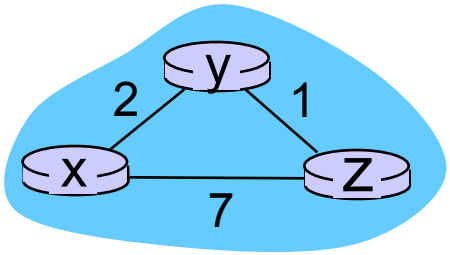
from \ cost to	x	y	z
x	∞	∞	∞
y	**2**	**0**	**1**
z	∞	∞	∞

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

node z table

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

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

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

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

node y table

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

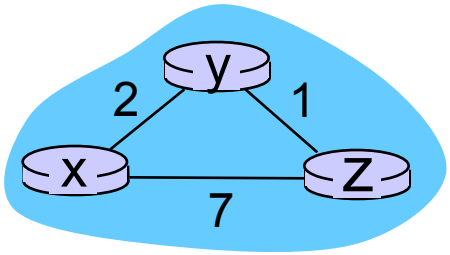
from \ cost to	x	y	z
x	0	2	7
y	2	0	1
z	7	1	0

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

node z table

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

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

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

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

node y table

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

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

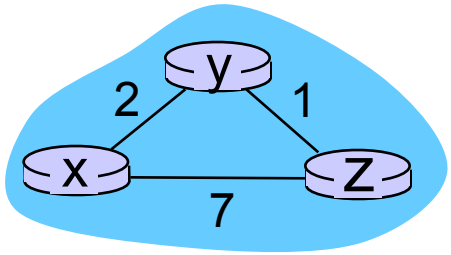
from \ cost to	x	y	z
x	0	2	3
y	2	0	1
z	3	1	0

node z table

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

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

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

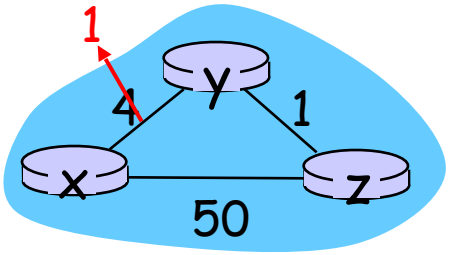


time

Distance vector: link cost decreases

link cost changes:

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



"good news travels fast"

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

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

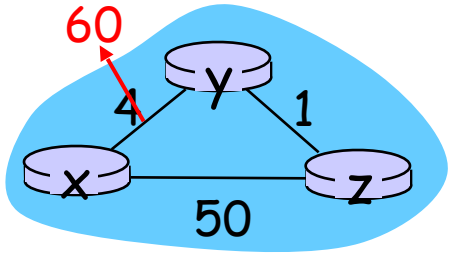
t_2 : *y* receives *z*'s update, updates its distance table. *y*'s least costs do *not* change, so *y* does *not* send a message to *z*.

* Check out the online interactive exercises for more examples: http://gaia.cs.umass.edu/kurose_ross/interactive/

Distance vector: link cost increases

link cost changes:

- node detects local link cost change
- *bad news travels slowly* - "count to infinity" problem!



Distance vector: count-to-infinity

node x table

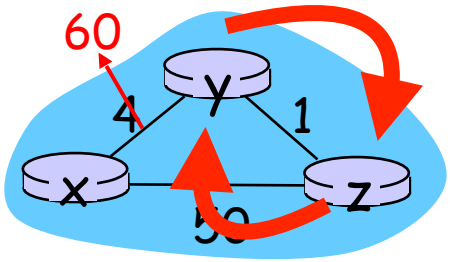
from \ cost to	x	y	z
x	0	4	5
y	4	0	1
z	5	1	0

$D_y(x) = \min\{c(y,x) + D_x(x), c(y,z) + D_z(x)\}$
$= \min\{60+0 , 1+5\} = 6$

node y table

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

from	x
x	0
y	**6**
z	5



node z table

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

time

Distance vector: count-to-infinity

node x table

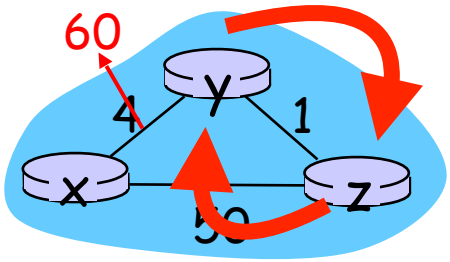
from \ cost to	x	y	z
x	0	4	5
y	4	0	1
z	5	1	0

$$D_z(x) = \min\{c(z,y) + D_y(x),\ c(z,x) + D_x(x)\}$$
$$= \min\{1+6\ ,\ 50+0\} = 7$$

node y table

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

from	x
x	0
y	**6**
z	5



node z table

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

from	x
x	0
y	**6**
z	

time

Distance vector: count-to-infinity

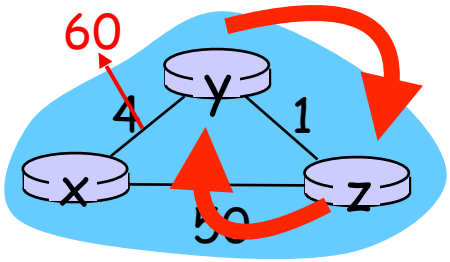
node x table

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

node y table

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

		via x
from	x	0
	y	**6**
	z	5



node z table

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

		via x
from	x	0
	y	**6**
	z	**7**

time

Distance vector: count-to-infinity

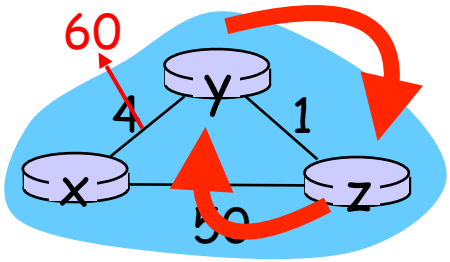
node x table

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

node y table

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

from	x
x	0
y	**6**
z	5 **7**



node z table

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

from	x
x	0
y	**6**
z	**7**

time

Distance vector: count-to-infinity

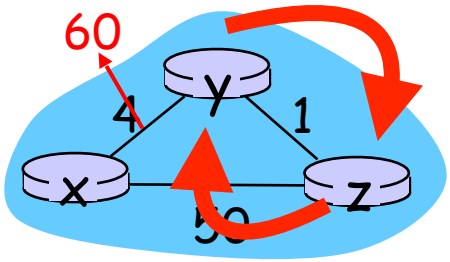
node x table

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

node y table

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

from	x
x	0
y	**6** **8**
z	5 **7**



node z table

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

from	x
x	0
y	**6**
z	**7**

time

Distance vector: count-to-infinity

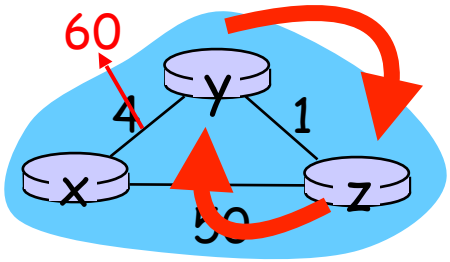
node x table

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

node y table

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

from	x
x	0
y	**6** **8**
z	5 **7**



node z table

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

from	x
x	0
y	**6** **8**
z	**7**

time

Distance vector: count-to-infinity

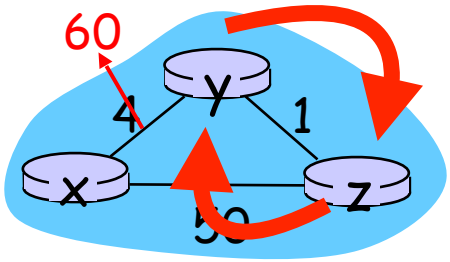
node x table

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

node y table

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

from	x
x	0
y	**6** **8**
z	5 **7**



node z table

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

from	x
x	0
y	**6** **8**
z	**7** **9**

time

Distance vector: count-to-infinity

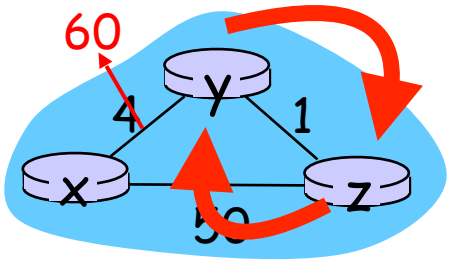
node x table

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

node y table

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

from	x
x	0
y	**6** **8**
z	5 **7** **9**



node z table

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

from	x
x	0
y	**6** **8**
z	**7** **9**

time

Distance vector: count-to-infinity

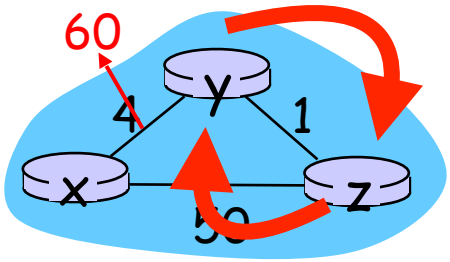
node x table

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

node y table

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

from	x
x	0
y	**6** **8** **10**
z	5 **7** **9**



node z table

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

from	x
x	0
y	**6** **8**
z	**7** **9**

time

Distance vector: count-to-infinity

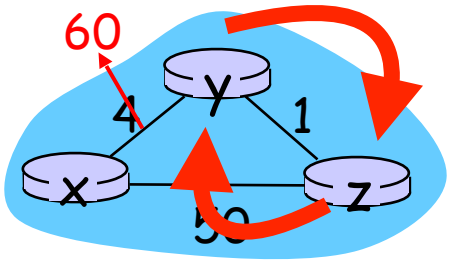
node x table

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

node y table

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

from	x
x	0
y	**6** **8** **10**
z	5 **7** **9**



node z table

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

from	x
x	0
y	**6** **8** **10**
z	**7** **9**

time

Distance vector: count-to-infinity

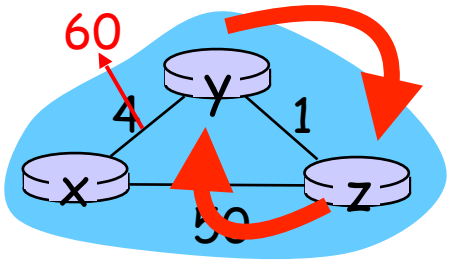
node x table

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

node y table

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

from	x
x	0
y	**6** **8** **10**
z	5 **7** **9**



node z table

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

from	x
x	0
y	**6** **8** **10**
z	**7** **9** **11**

time

Distance vector: count-to-infinity

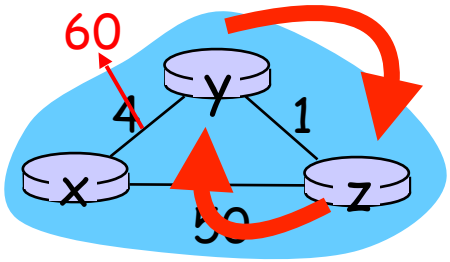
node x table

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

node y table

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

from	x
x	0
y	**6 8 10…**
z	5 **7 9…**



node z table

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

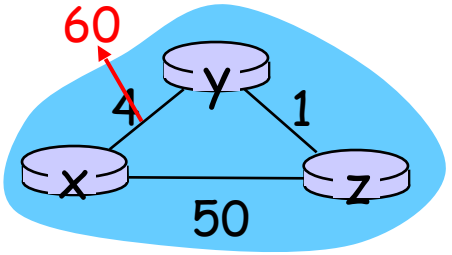
from	x
x	0
y	**6 8 10…**
z	**7 9 11…**

time

New: Distance vector: link cost increases

link cost changes:

- node detects local link cost change
- *bad news travels slowly* - "count to infinity" problem!
- 44 iterations before algorithm stabilizes: see text



poisoned reverse:

- If Z routes through Y to get to X :
  - Z tells Y its (Z's) distance to X is infinite (so Y won't route to X via Z)
- will this completely solve count to infinity problem?

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
  - 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:

- DV node can advertise incorrect *path* cost
- each node's table used by others
  - error propagate thru network

Uploading course materials to sites such as CourseHero, Chegg or Github is academic misconduct at Columbia (see pg 10 of Columbia guide).

Day 24: Software-Defined Networking (SDN)

CSEE 4119
Computer Networks
Ethan Katz-Bassett

COLUMBIA UNIVERSITY
IN THE CITY OF NEW YORK

Chapter 4: outline

4.1 Overview of Network layer
- data plane
- control plane

4.2 What's inside a router

4.3 IP: Internet Protocol
- datagram format
- fragmentation
- IPv4 addressing
- network address translation
- IPv6

4.4 Generalized forwarding and the SDN data plane
- match
- action
- OpenFlow examples of match-plus-action in action

Chapter 5: outline

5.1 introduction

5.2 routing protocols

- link state
- distance vector

5.3 intra-AS routing in the Internet: OSPF

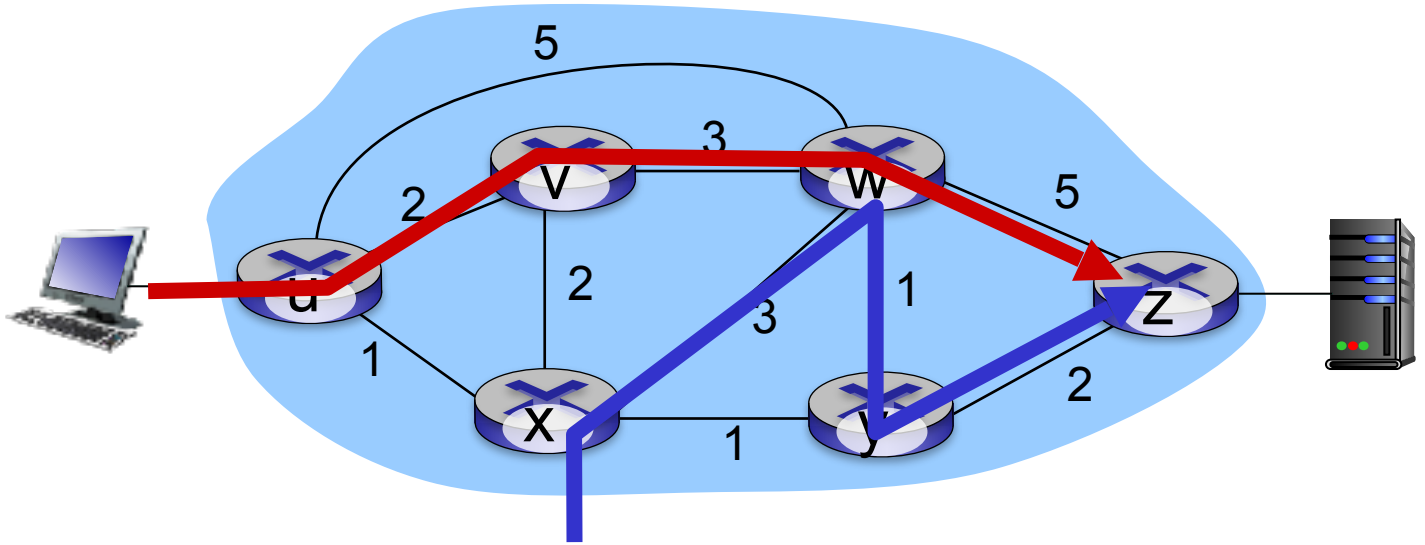
5.4 routing among the ISPs: BGP

5.5 The SDN control plane

5.6 ICMP: The Internet Control Message Protocol

5.7 Network management and SNMP

Traffic engineering: difficult traditional routing

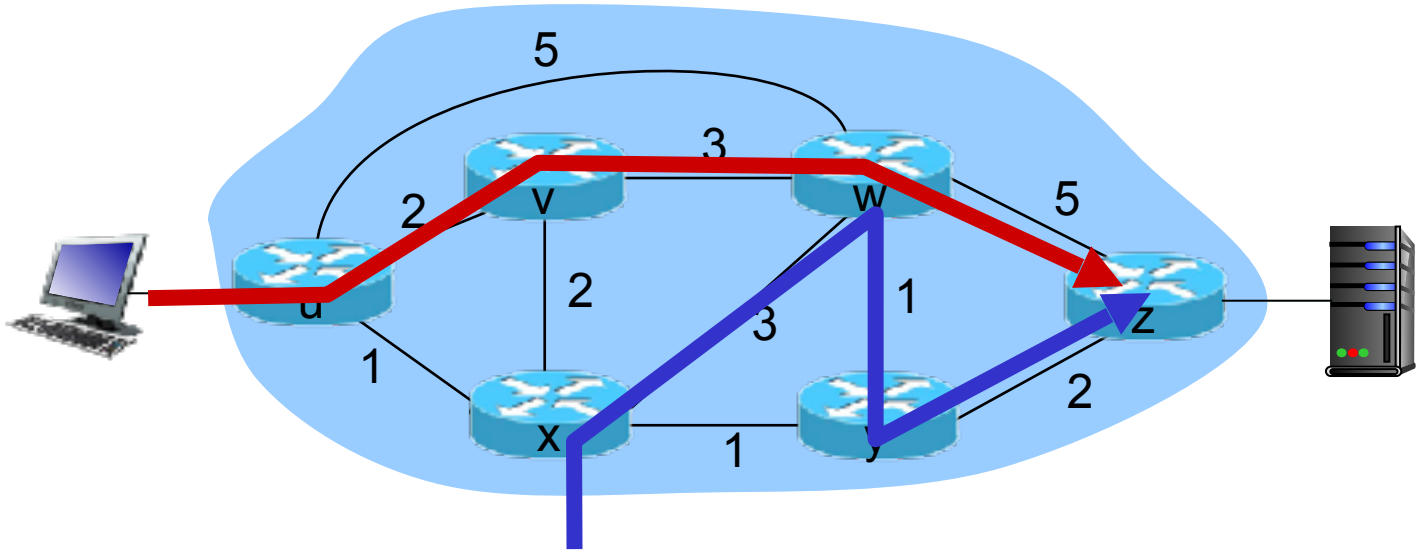


Q: what if network operator wants u-to-z traffic to flow along *uvwz*, x-to-z traffic to flow *xwyz*?

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

Link weights are only control "knobs": wrong!

Traffic engineering: difficult



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

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

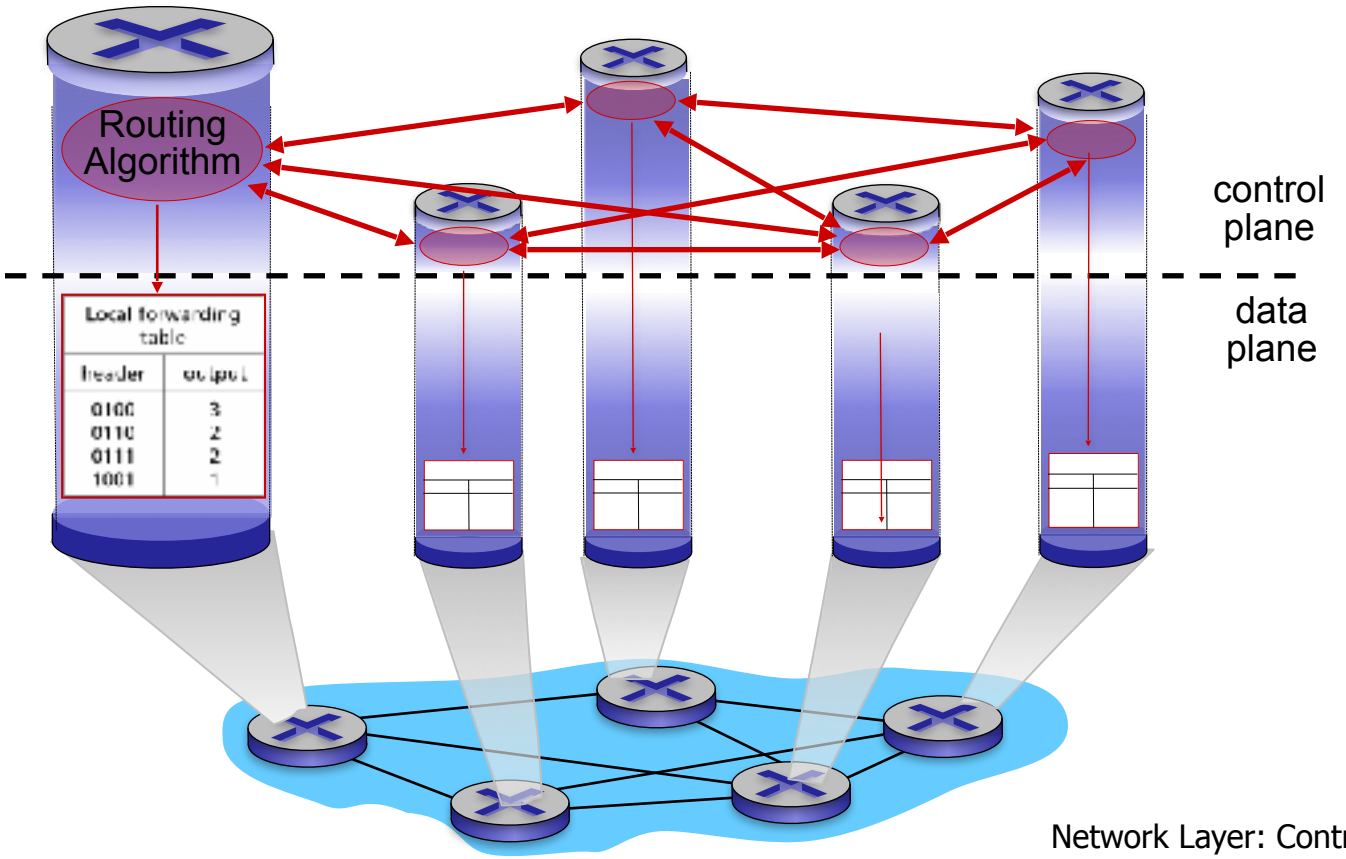
Rethinking network control

- Internet network layer: historically has been implemented via distributed, per-router approach
  - *monolithic* router contains switching hardware, runs proprietary implementation of Internet standard protocols (IP, RIP, IS-IS, OSPF, BGP) in proprietary router OS (e.g., Cisco IOS)
  - different "middleboxes" for different network layer functions: firewalls, load balancers, NAT boxes, ..

- ~2005: renewed interest in rethinking network control plane

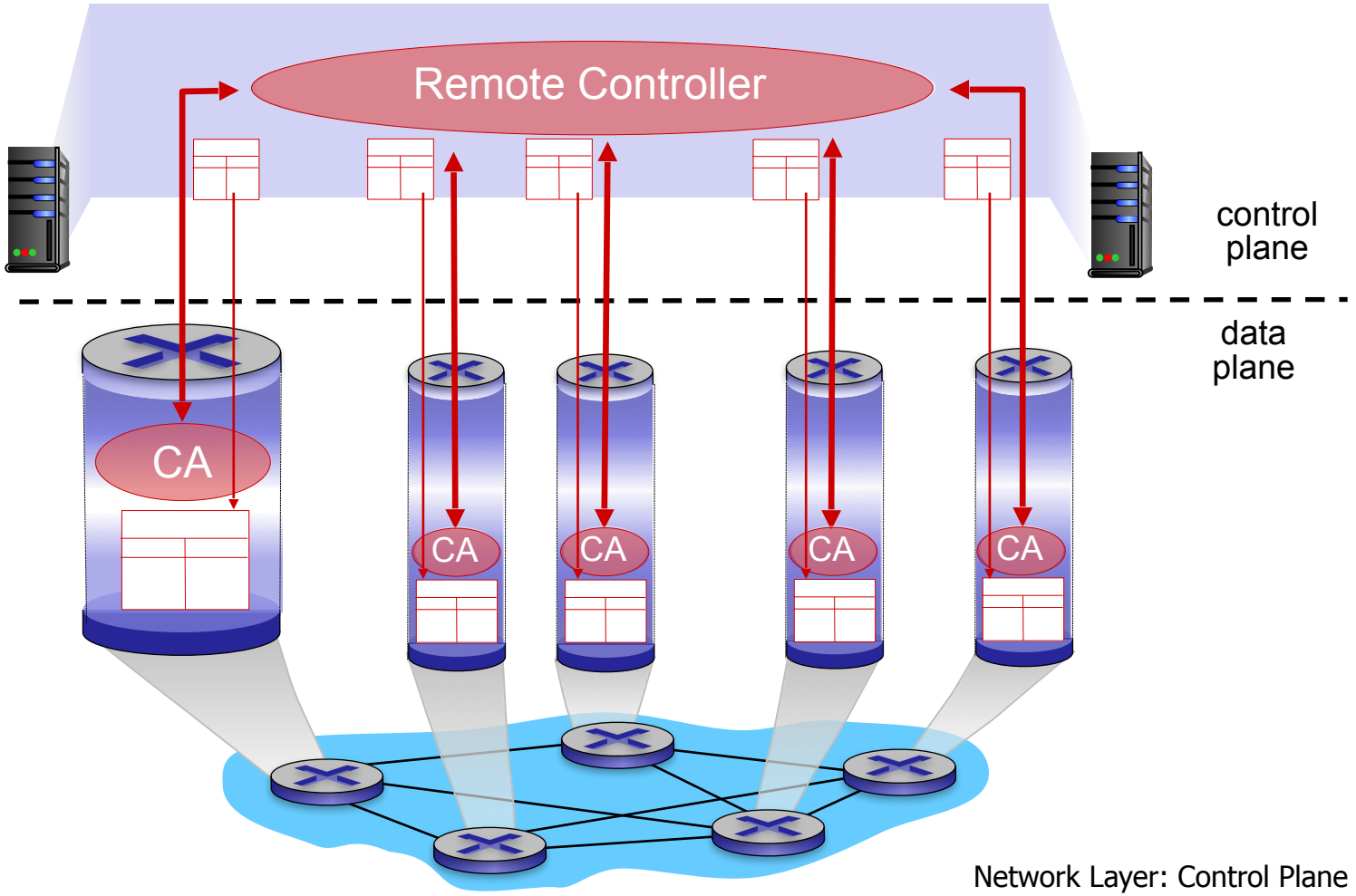
Recall: per-router control plane

Individual routing algorithm components *in each and every router* interact with each other in control plane to compute forwarding tables



Recall: logically centralized control plane

A distinct (typically remote) controller interacts with local control agents (CAs) in routers to compute forwarding tables

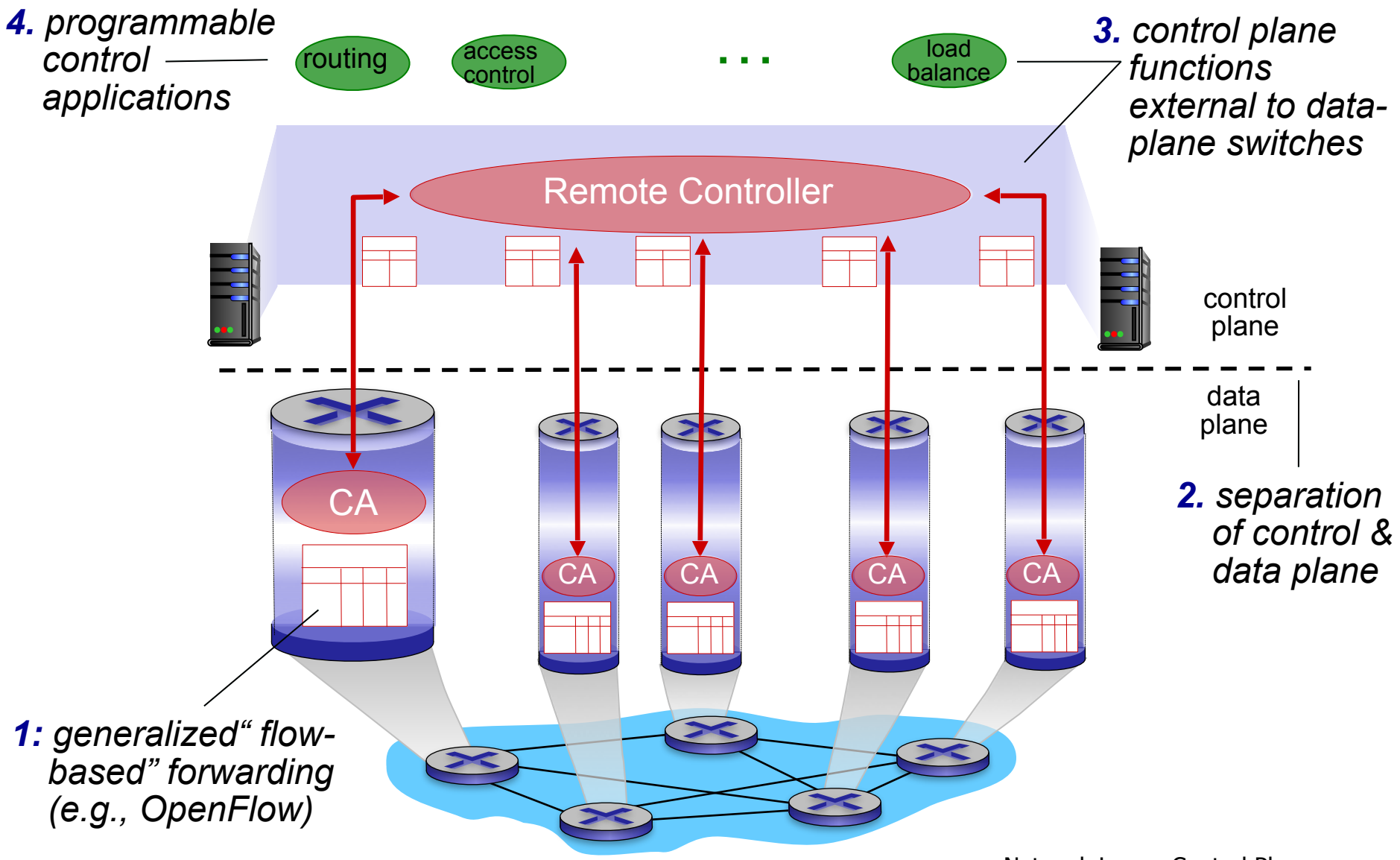


Software defined networking (SDN)

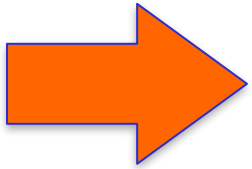
Why a *logically centralized* control plane?

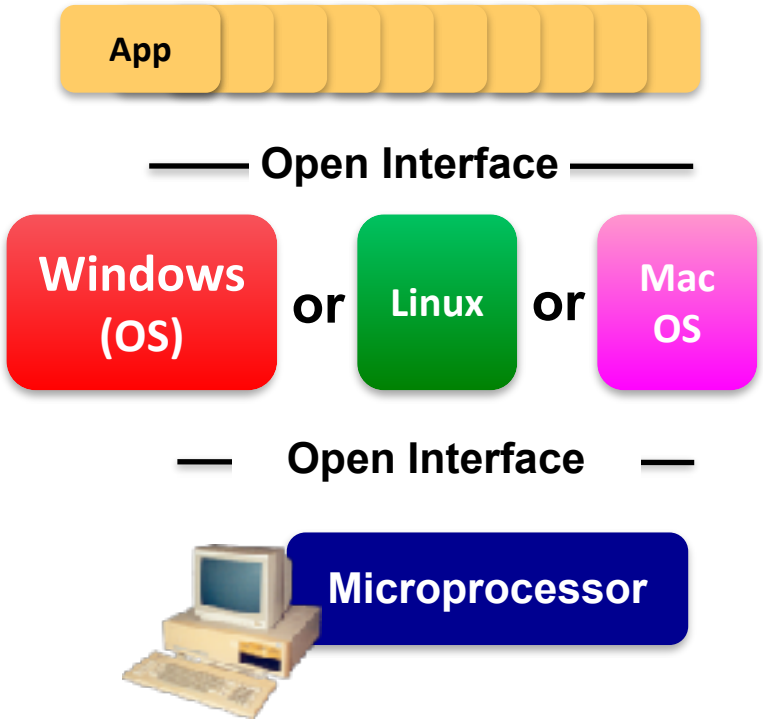
- easier network management: avoid router misconfigurations, greater flexibility of traffic flows
- table-based forwarding (e.g., OpenFlow API) allows "programming" routers
  - centralized "programming" easier: compute tables centrally and distribute
  - distributed "programming: more difficult: compute tables as result of distributed algorithm (protocol) implemented in each and every router
- open (non-proprietary) implementation of control plane
  - foster innovation: let 1000 flowers bloom
- evolve as software: fast, using established software engineering practices

Software defined networking (SDN)

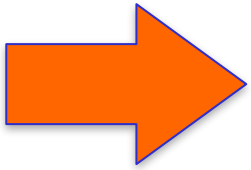


Analogy: mainframe to PC evolution*





Vertically integrated
Closed, proprietary
Slow innovation
Small industry



Horizontal
Open interfaces
Rapid innovation
Huge industry

* Slide courtesy: N. McKeown

Chapter 4: outline

4.1 Overview of Network layer
- data plane
- control plane

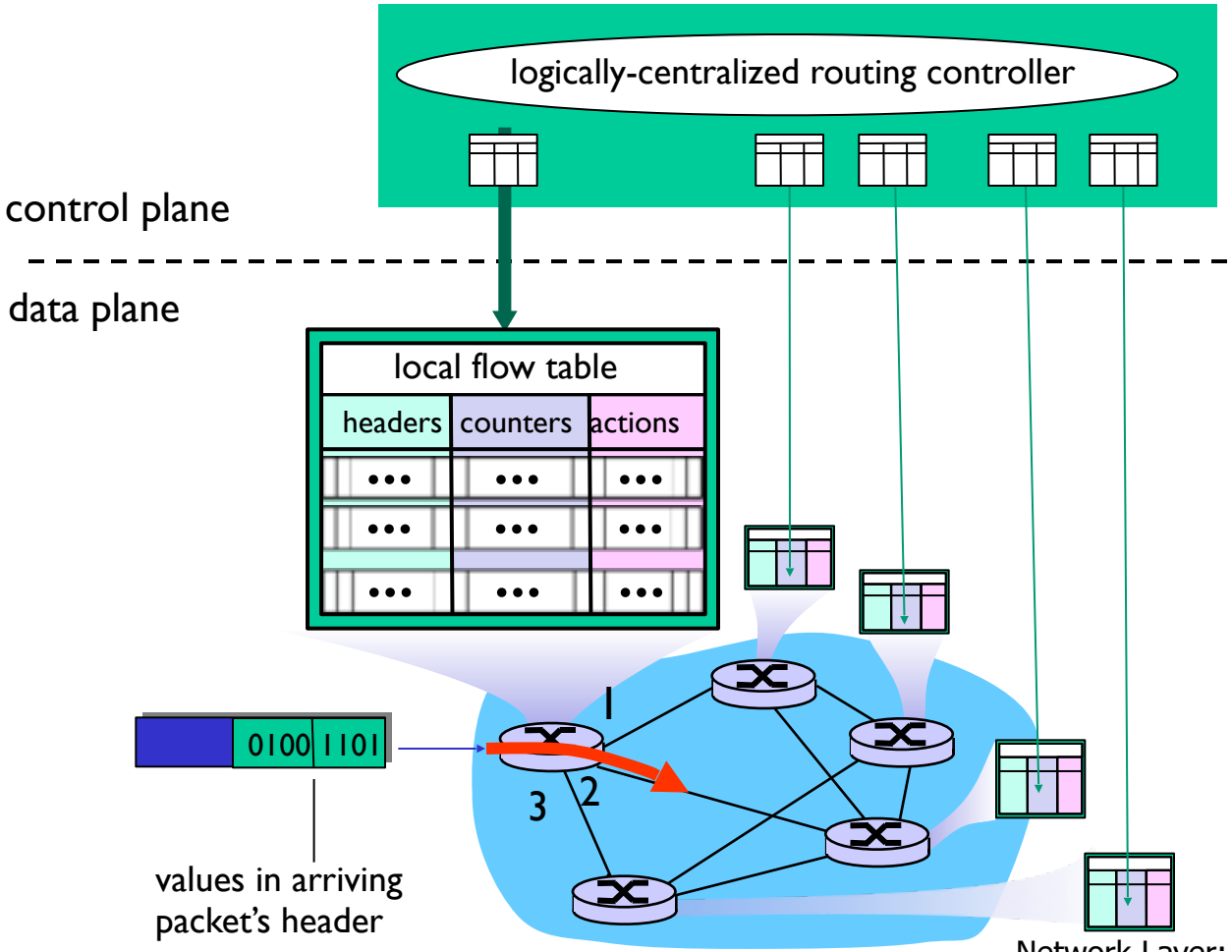
4.2 What's inside a router

4.3 IP: Internet Protocol
- datagram format
- fragmentation
- IPv4 addressing
- network address translation
- IPv6

4.4 Generalized Forward and SDN
- match -> action
- OpenFlow examples of match-plus-action in action

Generalized Forwarding and SDN

Each router contains a *flow table* that is computed and distributed by a *logically centralized* routing controller



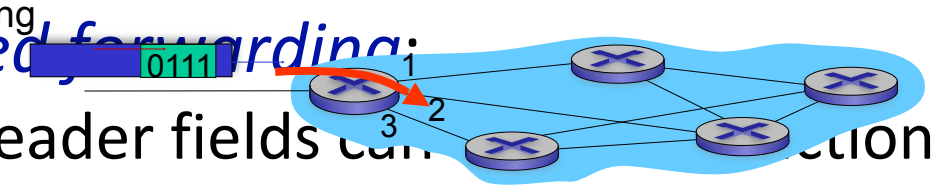
Generalized forwarding: match plus action

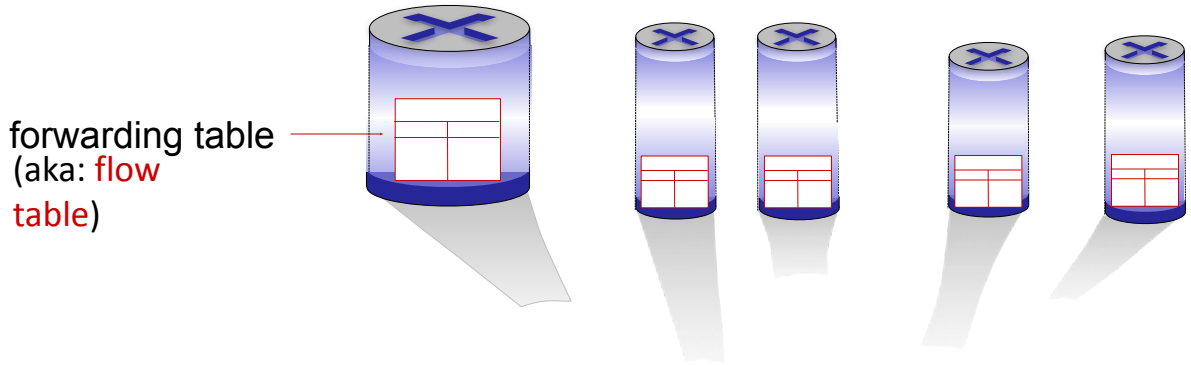
Each router contains a forwarding table (aka flow table)

- "match plus action" abstraction:

  match bits in arriving packet, take action

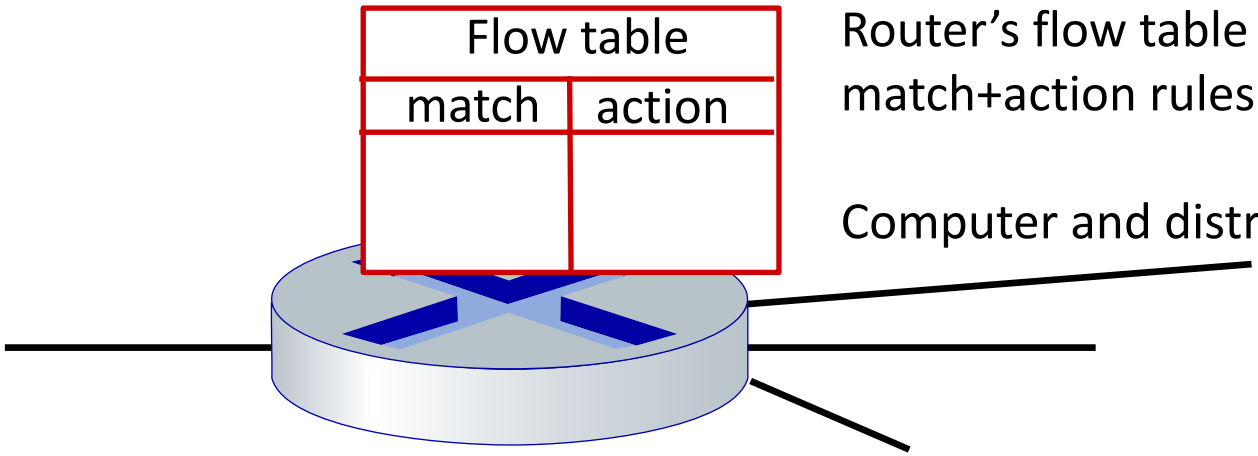
  - *destination-based forwarding:* forward based on dest. IP address
  - *generalized forwarding:*
    - many header fields can determine action
    - many action possible: drop/copy/modify/log packet





OpenFlow data plane abstraction

- *flow*: defined by header fields
- generalized forwarding: simple packet-handling rules
  - *Match:* pattern 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

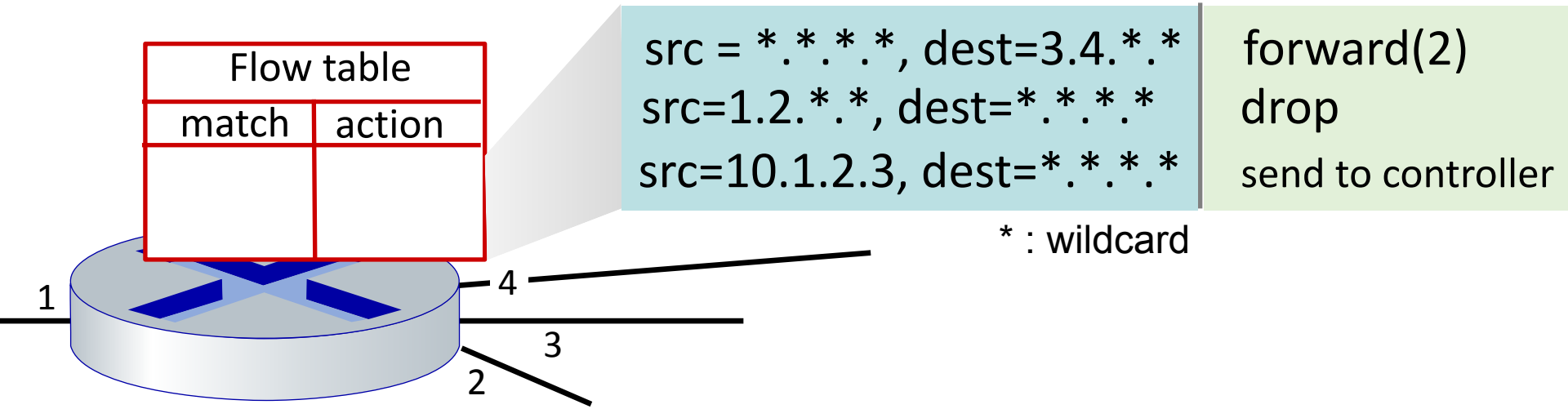


Router's flow table define router's match+action rules.

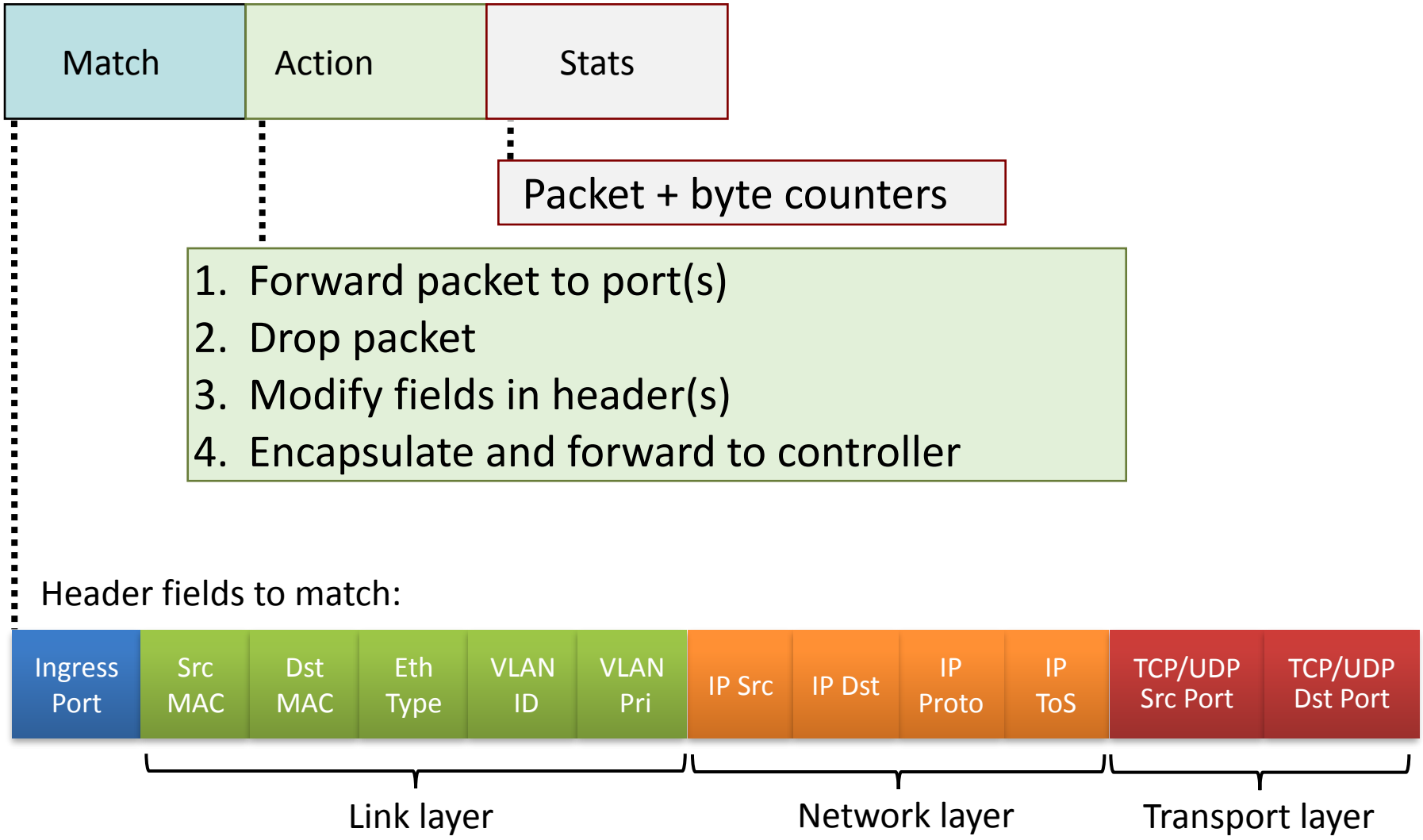
Computer and distributed by controller.

OpenFlow data plane abstraction

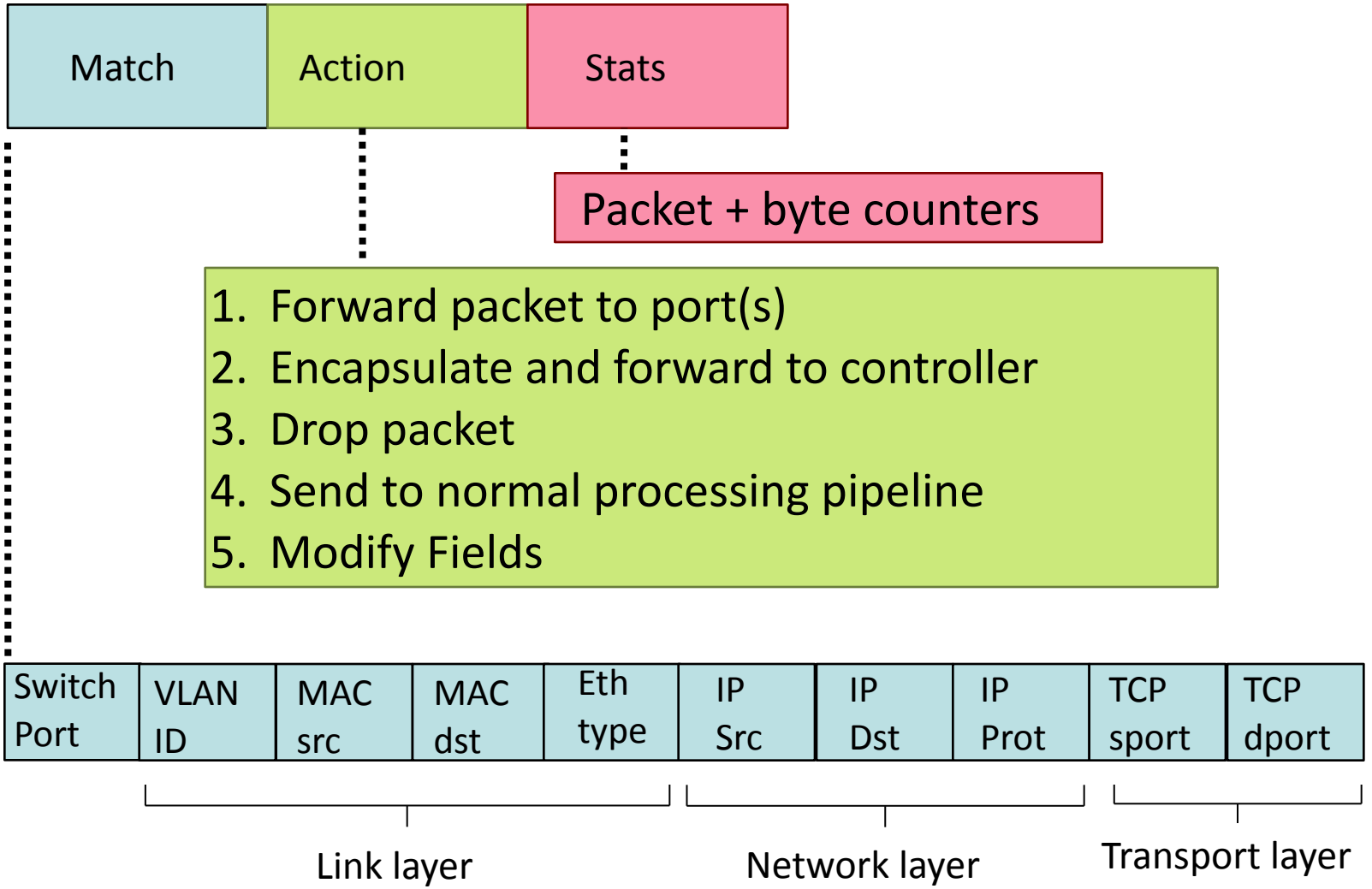
- *flow*: defined by header fields
- generalized forwarding: simple packet-handling rules
  - *Match:* pattern values in packet header fields
  - *Actions (for matched packet):* drop, forward, modify, matched packet; or send matched packet to controller
  - *Priority*: disambiguate overlapping patterns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



OpenFlow: Flow Table Entries



OpenFlow: Flow Table Entries



Examples

Destination-based forwarding:

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	Action
*	*	*	*	*	*	51.6.0.8	*	*	*	port6

IP datagrams destined to IP address 51.6.0.8 should be forwarded to router output port 6

Firewall:

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	Forward
*	*	*	*	*	*	*	*	*	22	drop

do not forward (block) all datagrams destined to TCP port 22

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	Forward
*	*	*	*	*	128.119.1.1	*	*	*	*	drop

do not forward (block) all datagrams sent by host 128.119.1.1

Examples

Destination-based layer 2 (switch) forwarding:

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	Action
*	22:A7:23: 11:E1:02	*	*	*	*	*	*	*	*	port3

layer 2 frames from MAC address 22:A7:23:11:E1:02 should be forwarded to output port 6

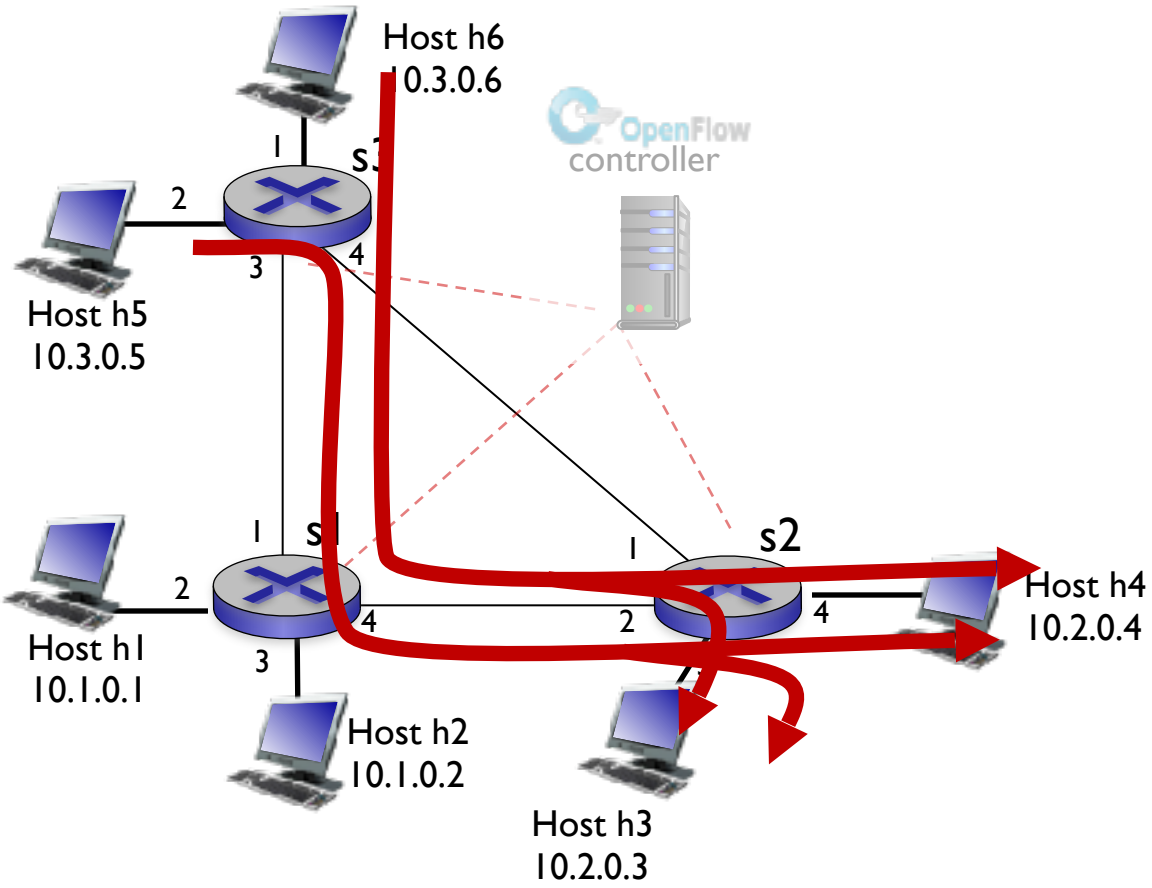
OpenFlow abstraction

- *match+action:* unifies different kinds of devices

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

OpenFlow example

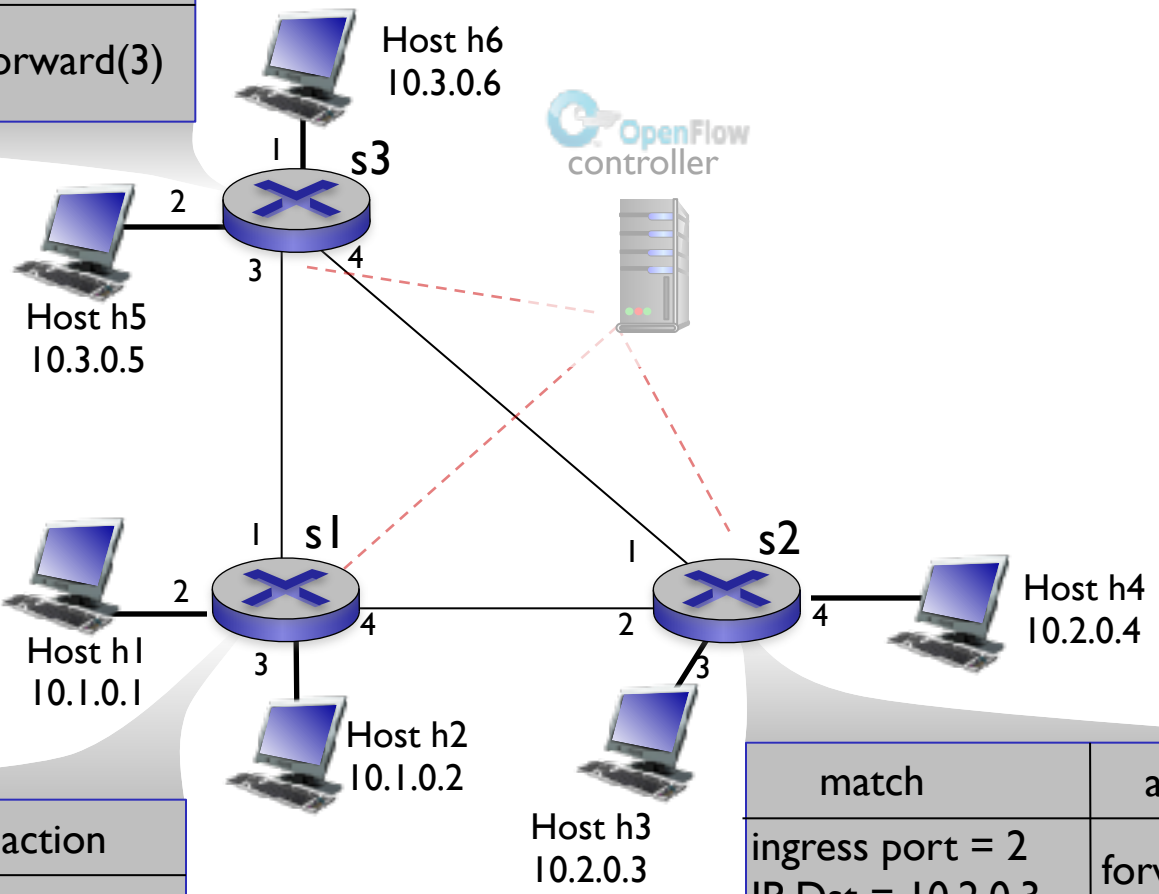
Example: for datagrams from hosts h5/h6 to h3/h4, send via s1 and then s2



OpenFlow example

Example: for datagrams from hosts h5/h6 to h3/h4, send via s1 and then s2

match	action
IP Src = 10.3.*.* IP Dst = 10.2.*.*	forward(3)



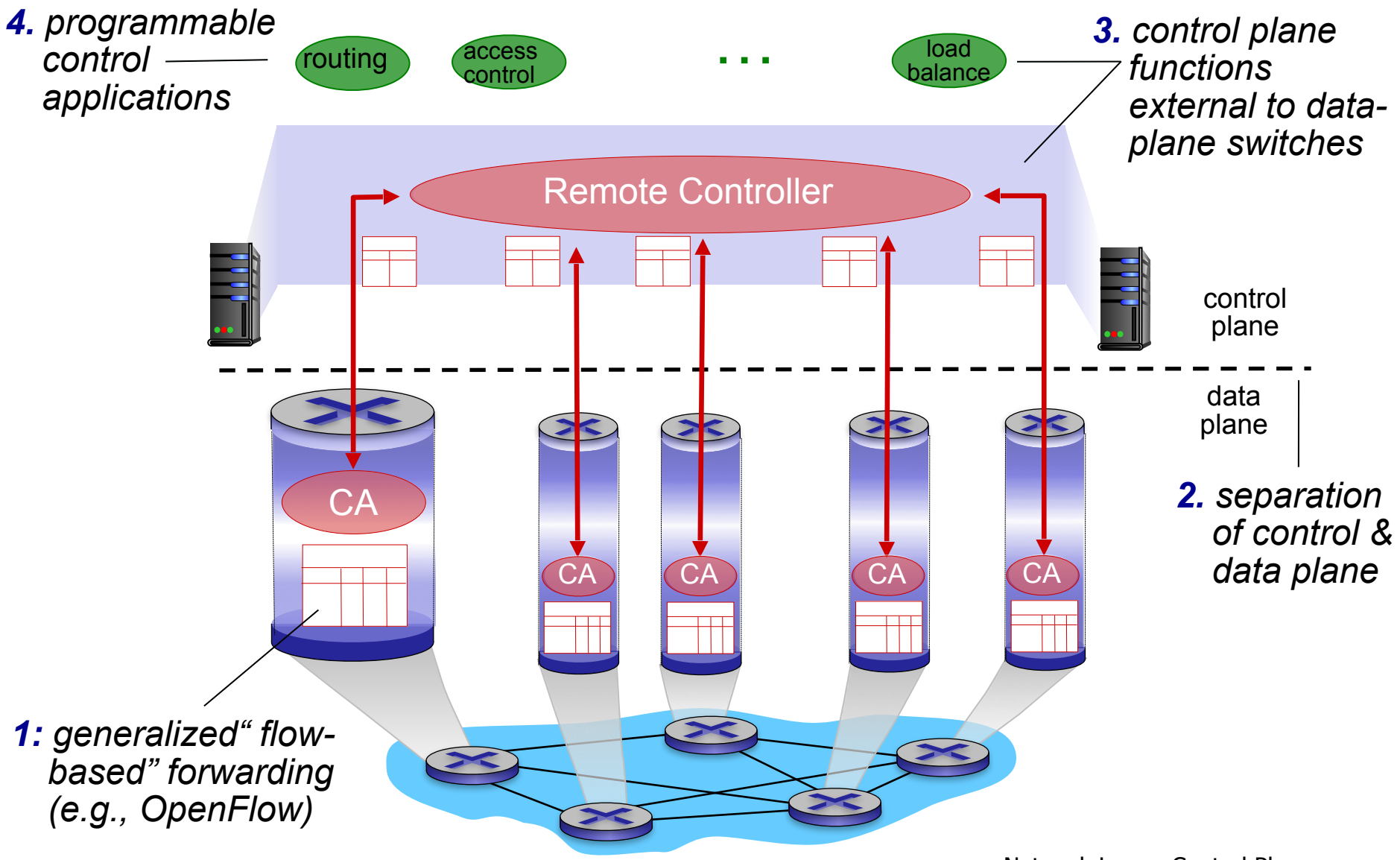
match	action
ingress port = 1 IP Src = 10.3.*.* IP Dst = 10.2.*.*	forward(4)

match	action
ingress port = 2 IP Dst = 10.2.0.3	forward(3)
ingress port = 2 IP Dst = 10.2.0.4	forward(4)

Generalized forwarding: summary

- "match plus action" abstraction: match bits in arriving packet header(s) in any layers, take action
  - matching over many fields (link-, network-, transport-layer)
  - local actions: drop, forward, modify, or send matched packet to controller
  - "program" n*etwork-wide* behaviors
- simple form of "network programmability"
  - programmable, per-packet "processing"
  - *historical roots:* active networking
  - *today:* more generalized programming: P4 (see p4.org).

Software defined networking (SDN)

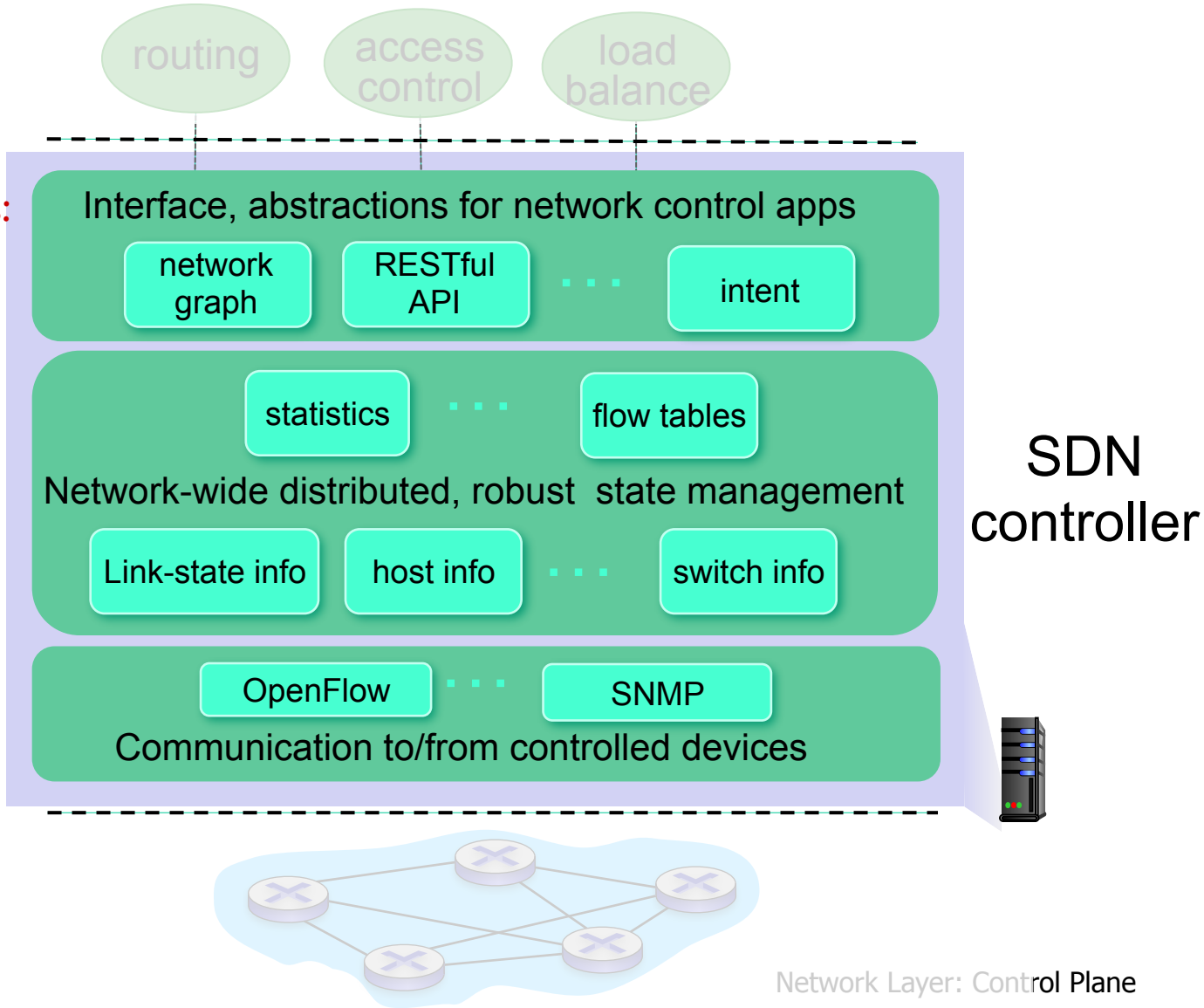


Components of SDN controller

Interface layer to network control apps: abstractions API

Network-wide state management layer: state of networks links, switches, services: a *distributed database*

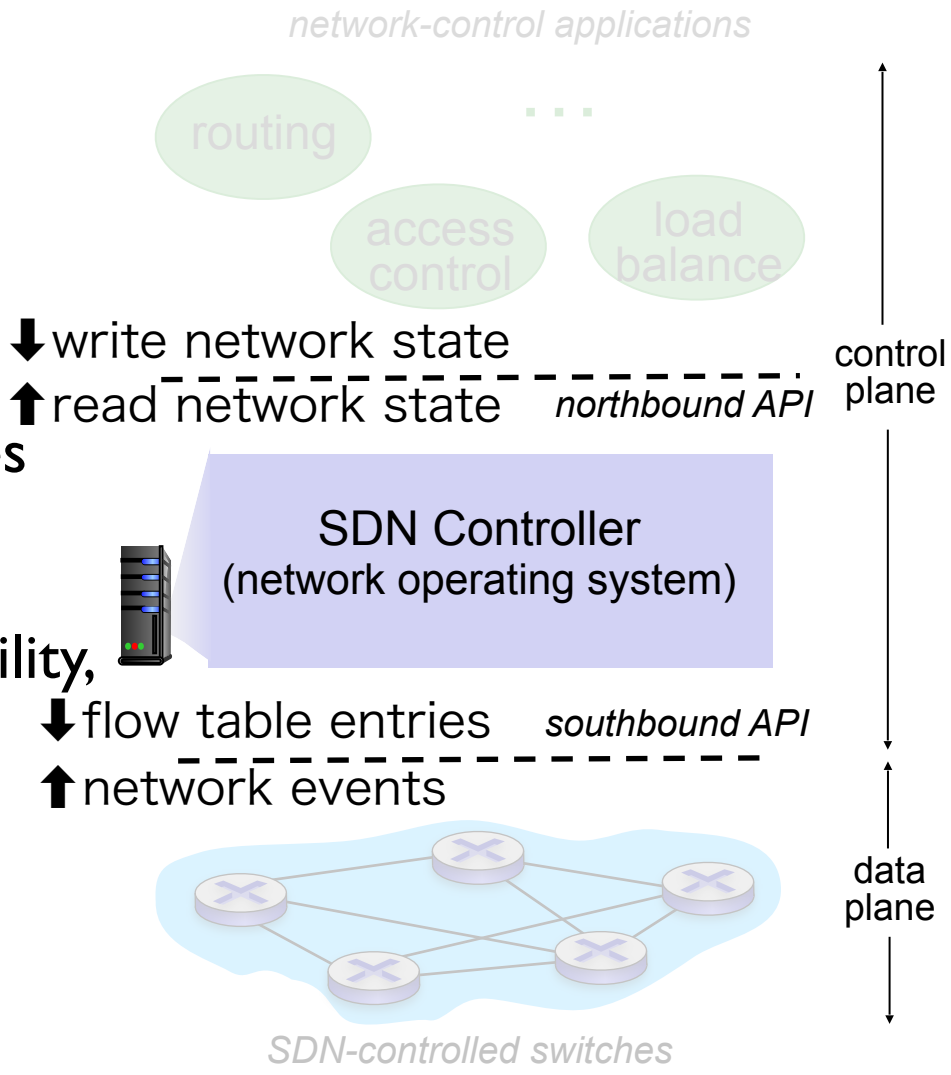
communication layer: communicate between SDN controller and controlled switches



SDN perspective: SDN controller

SDN controller (network OS):

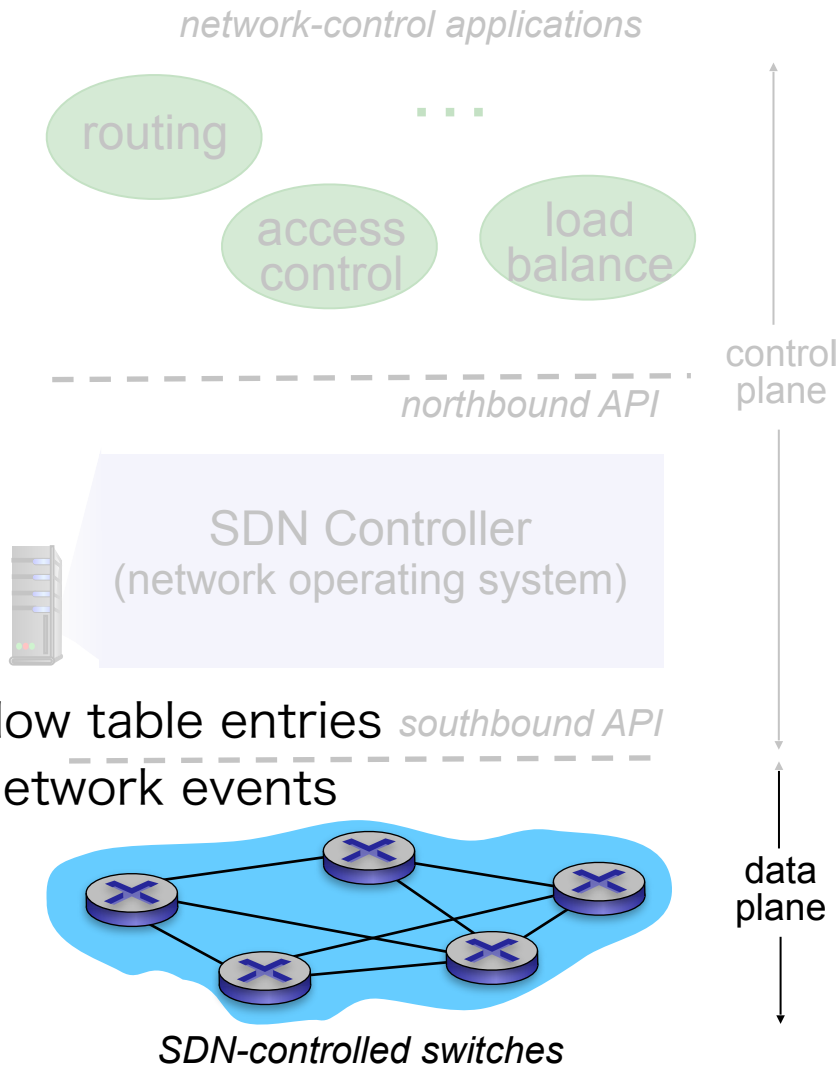
- maintain network state
- interacts with network control applications "above" via northbound API
- interacts with network switches "below" via southbound API
- implemented as distributed system for performance, scalability, fault-tolerance, robustness



SDN perspective: data plane switches

Data plane switches

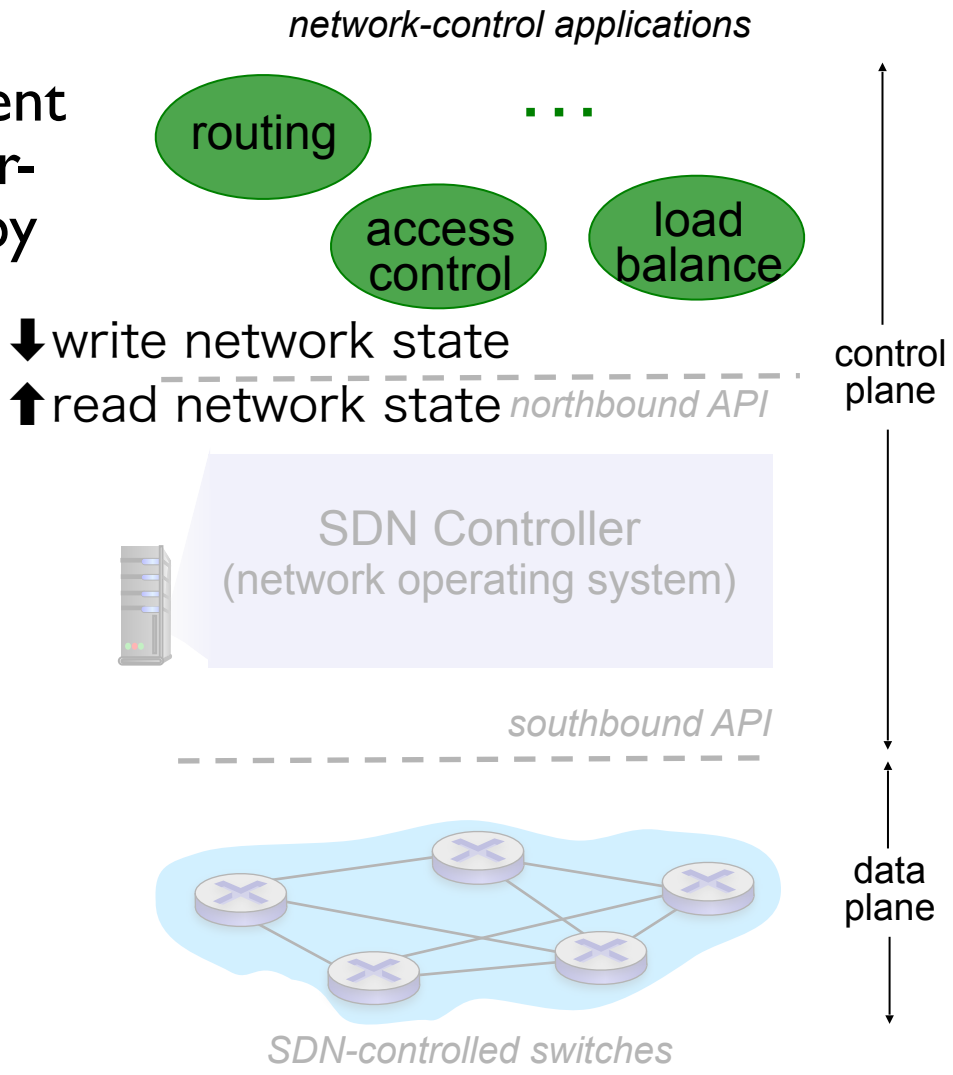
- fast, simple, commodity switches implementing generalized data-plane forwarding in hardware
- switch flow table computed, installed by controller
- API for table-based switch control (e.g., OpenFlow)
  - defines what is controllable and what is not
- protocol for communicating with controller (e.g., OpenFlow)



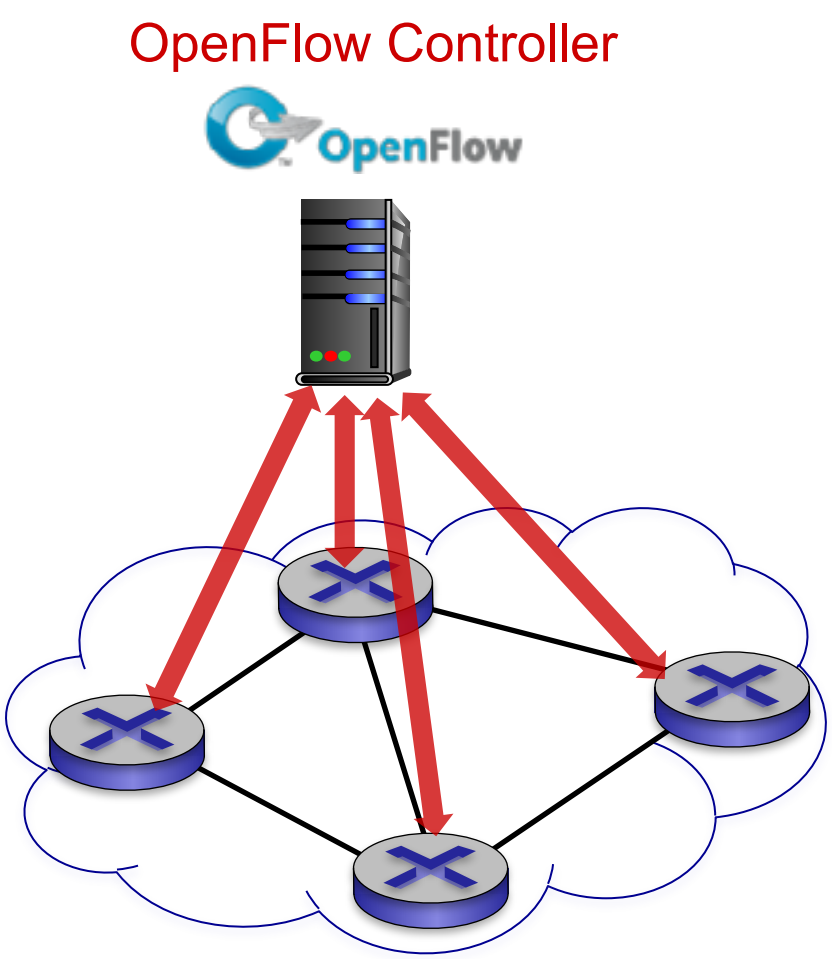
SDN perspective: control applications

network-control apps:

- "brains" of control: implement control functions using lower-level services, API provided by SDN controller
- *unbundled:* can be provided by 3rd party: distinct from routing vendor or SDN controller author



OpenFlow protocol

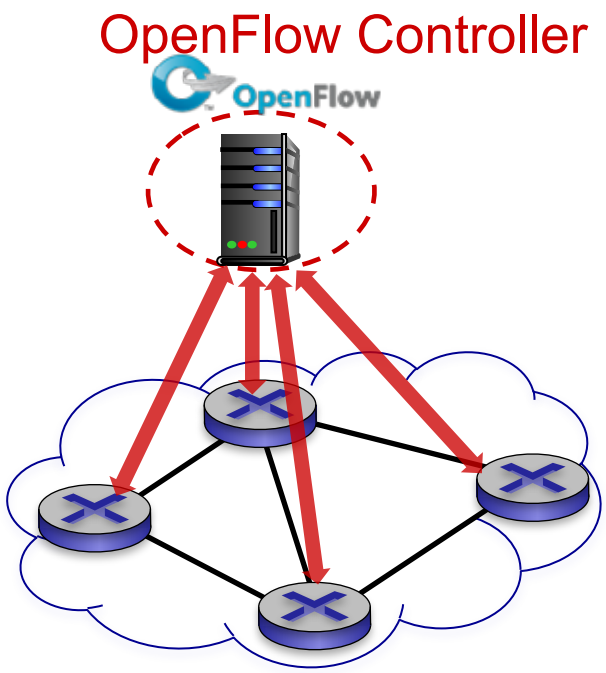


- operates between controller, switch
- TCP used to exchange messages
  - optional encryption

OpenFlow: controller-to-switch messages

Key controller-to-switch messages

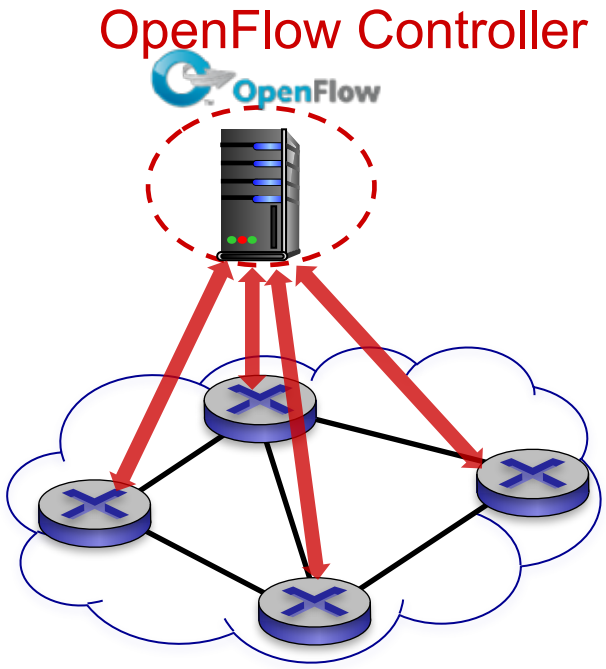
- *features:* controller queries switch features, switch replies
- *configure:* controller queries/sets switch configuration parameters
- *modify-state:* add, delete, modify entries in the OpenFlow tables
- *packet-out:* controller instructs switch to send packet (included in message) out of specific switch port



OpenFlow: switch-to-controller messages

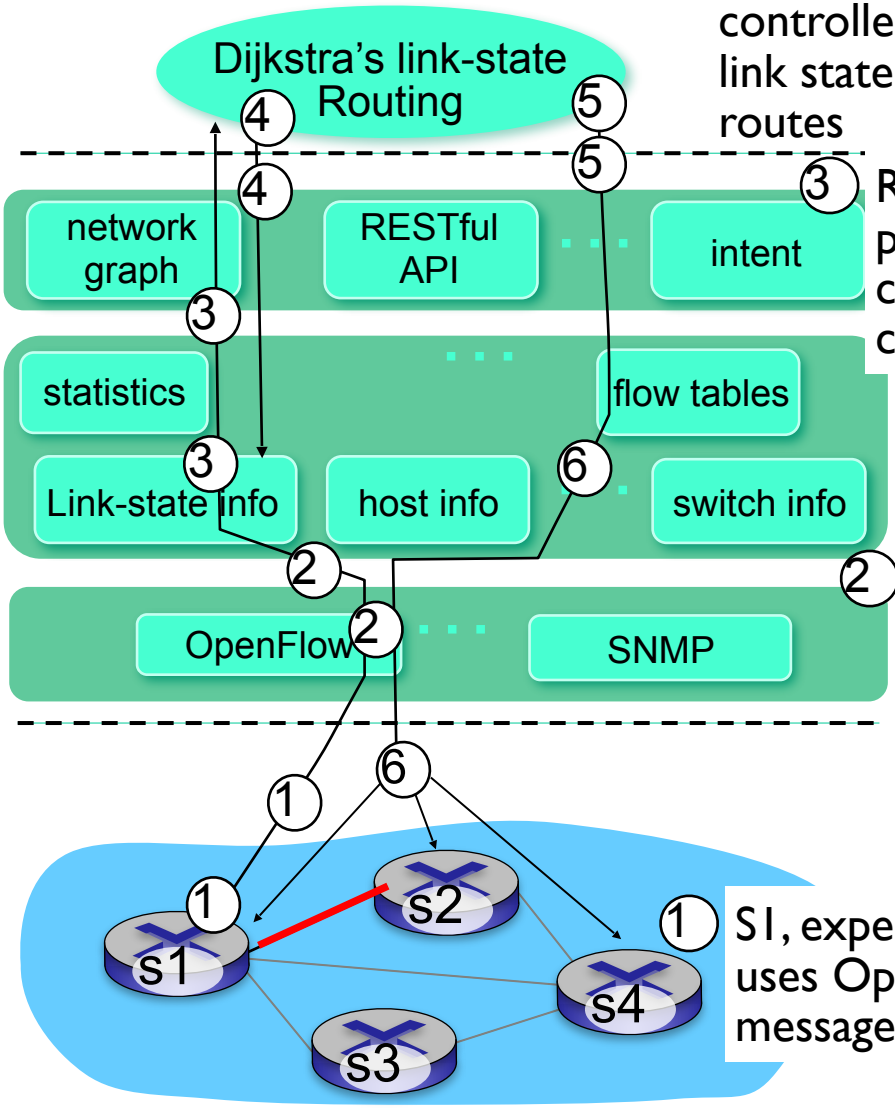
Key switch-to-controller messages

- *packet-in:* transfer packet (and its control) to controller
- *flow-removed:* flow table entry deleted at switch
- *port-status:* inform controller of a change on a port.



Fortunately, network operators don't "program" switches by creating/sending OpenFlow messages directly. Instead use higher-level abstraction at controller

SDN: control/data plane interaction example



4 Routing application asks controller for network graph & link state info, computes new routes

3 Routing application has previously registered to be called whenever link status changes. Controller calls it.

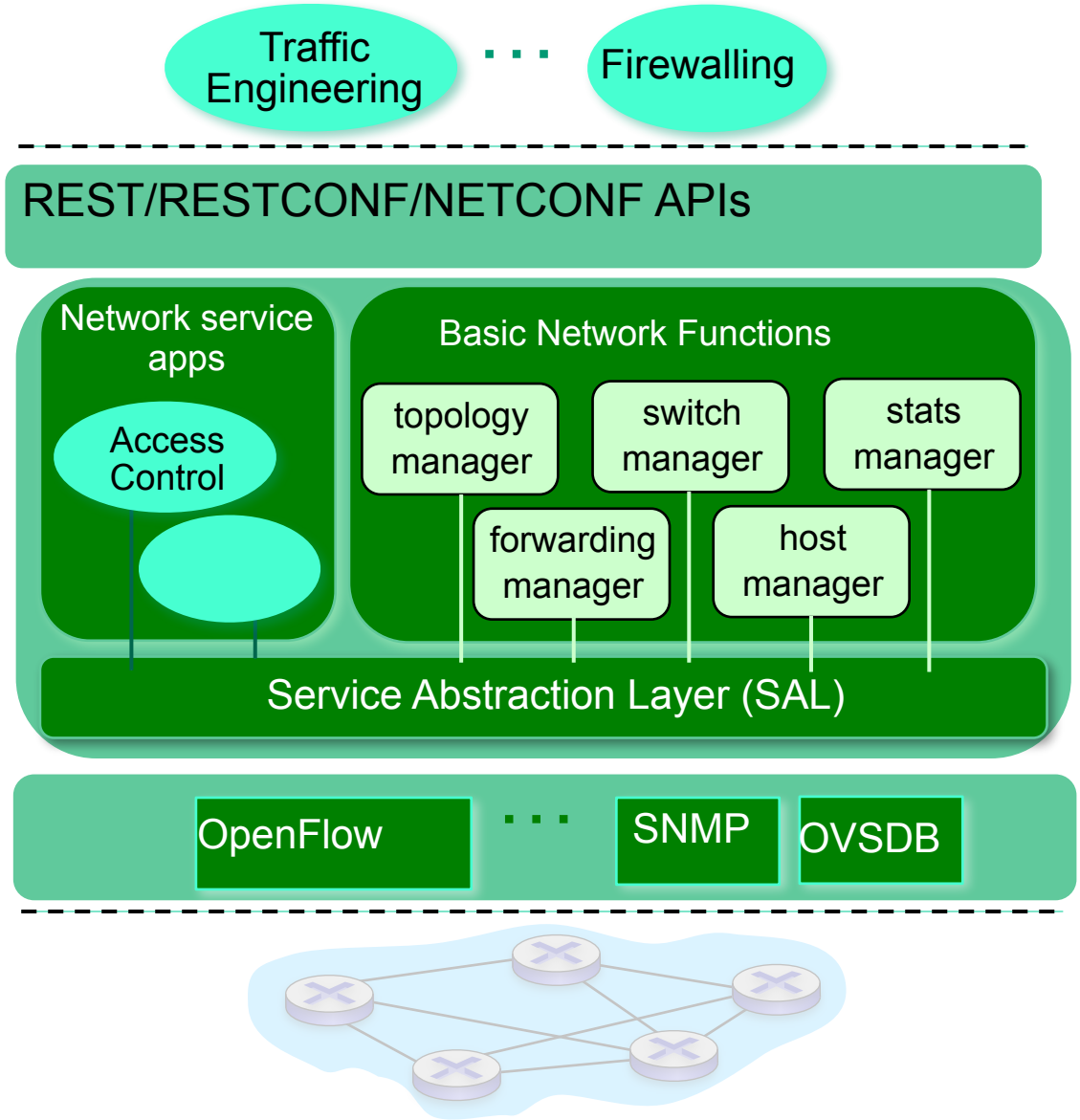
5 Routing application instructs controller to update flow tables

2 Controller receives message, updates link status info

6 Controller uses OpenFlow to install new tables in switches that need updating

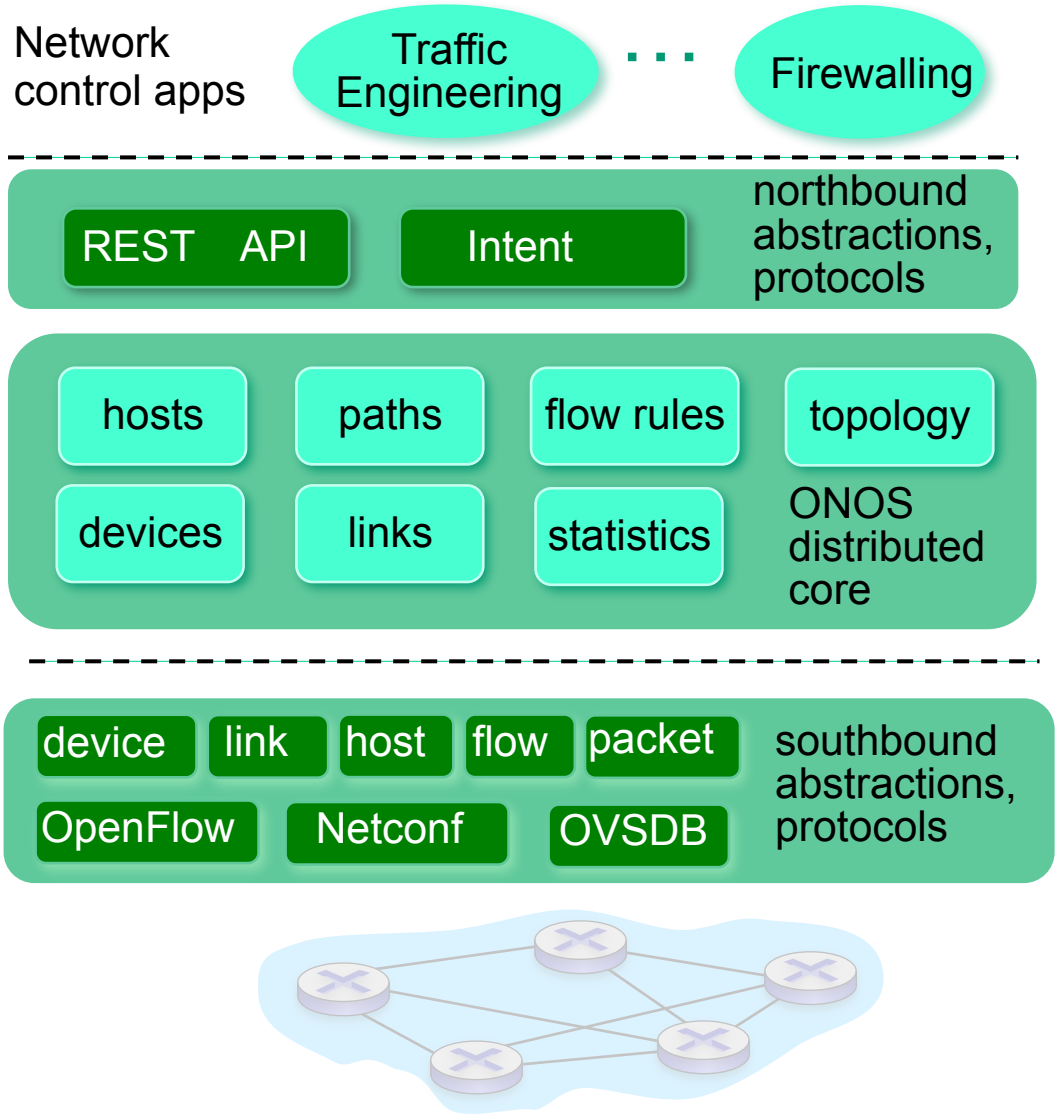
1 S1, experiencing link failure uses OpenFlow port status message to notify controller

OpenDaylight (ODL) controller



- network apps may be contained within, or be external to SDN controller
- Service Abstraction Layer: interconnects internal, external applications and services

ONOS controller



- control apps separate from controller
- intent framework: high-level specification of service: *what to achieve* rather than *how to achieve it*
- considerable emphasis on distributed core: service reliability, replication performance scaling

SDN: selected challenges

- hardening the control plane: dependable, reliable, performance-scalable, secure distributed system
  - robustness to failures: leverage strong theory of reliable distributed system for control plane
  - dependability, security: "baked in" from day one?
  - verifying correctness of control programs
- networks, protocols meeting mission-specific requirements
  - e.g., real-time, ultra-reliable, ultra-secure
- Internet-scaling and inter-domain control
- SDN critical in 5G cellular networks

Applying SDN to BGP

Edge Fabric:
Delivering Oceans of Content to the World

Brandon Schlinker[1,2]
Hyojeong Kim[1], Timothy Cui[1], Ethan Katz-Bassett[2,3], Harsha V. Madhyastha[4], Italo Cunha[5]
James Quinn[1], Saif Hasan[1], Petr Lapukhov[1], James Hongyi Zeng[1]

[1] Facebook, [2] University of Southern California, [3] Columbia University,
[4] University of Michigan, [5] Universidade Federal de Minas Gerais

SIGCOMM, August 2017

Facebook's Global Network

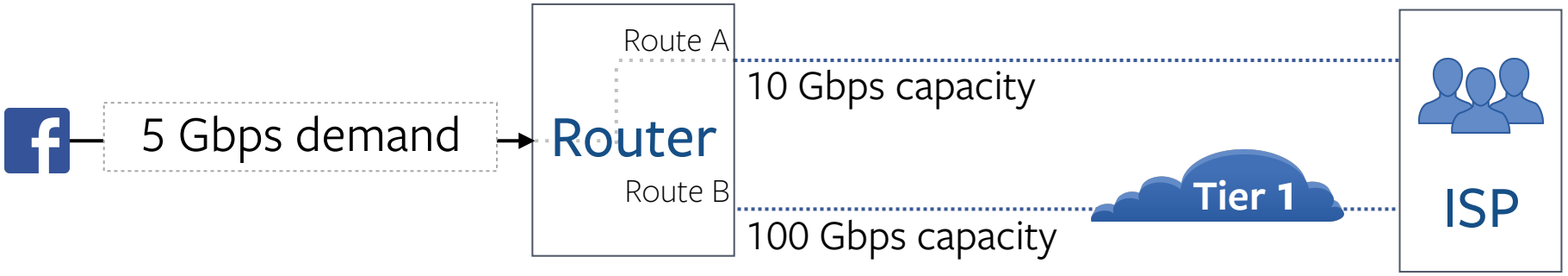
points of presence
around the world

interconnect with
thousands of networks

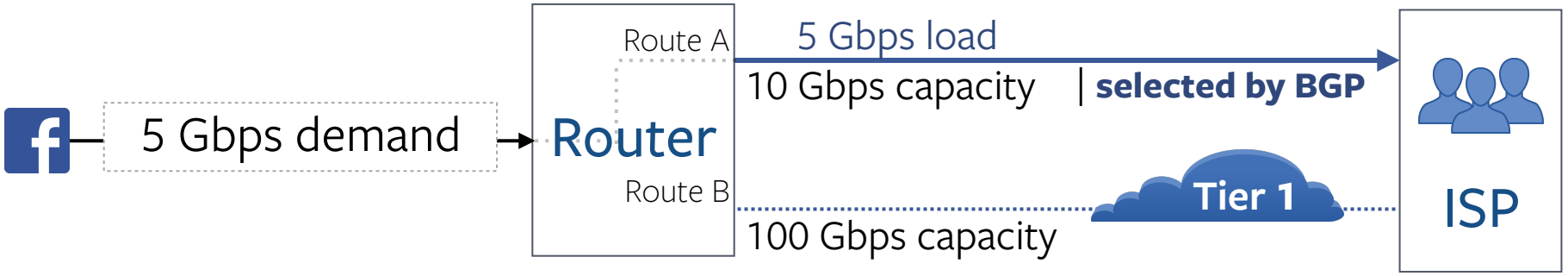
Challenges to Using Our Connectivity

objective	deliver traffic with the best performance possible
challenge	BGP does not consider demand, capacity or performance

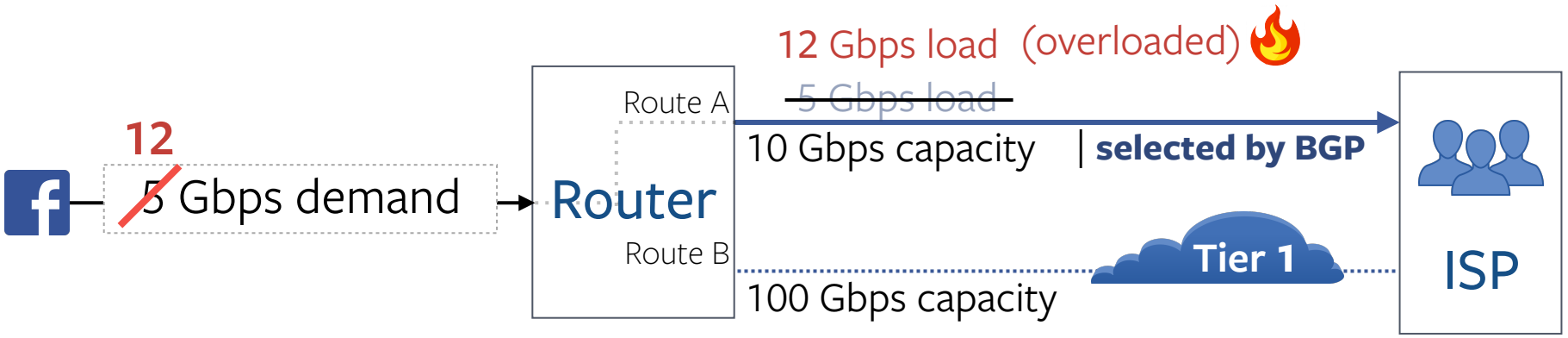
BGP Does Not Consider Demand and Capacity



BGP Does Not Consider Demand and Capacity



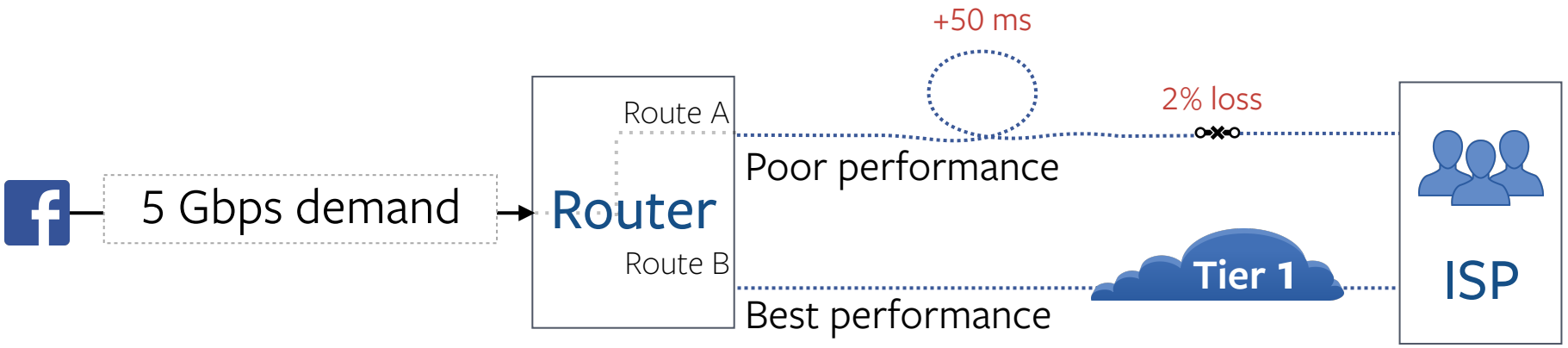
BGP Does Not Consider Demand and Capacity



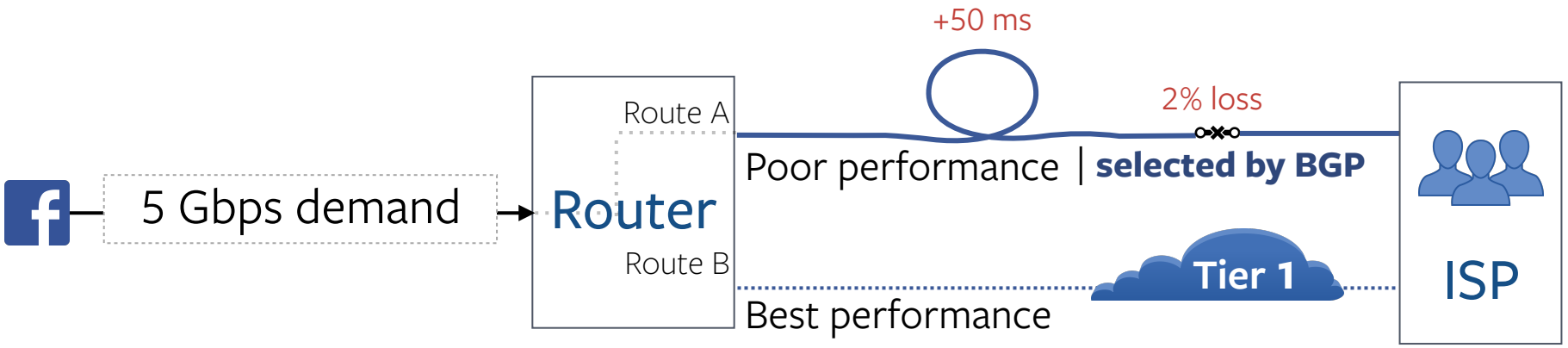
Cannot configure BGP to adapt to demand/capacity in real time

Not possible to express with BGP policy terms

BGP Does Not Consider Performance



BGP Does Not Consider Performance



Cannot configure BGP to adapt to performance in real time

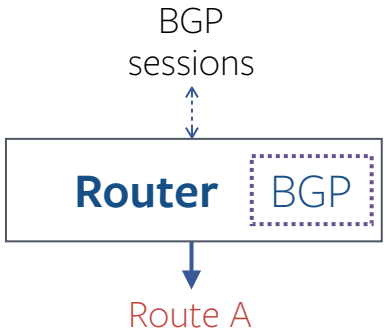
Not possible to express with BGP policy terms

Sidestepping BGP's Limitations

objective	deliver traffic with the best performance possible
challenge	BGP does not consider demand, capacity or performance
approach	shift control from BGP at routers to a software controller

Edge Fabric's Approach to Control

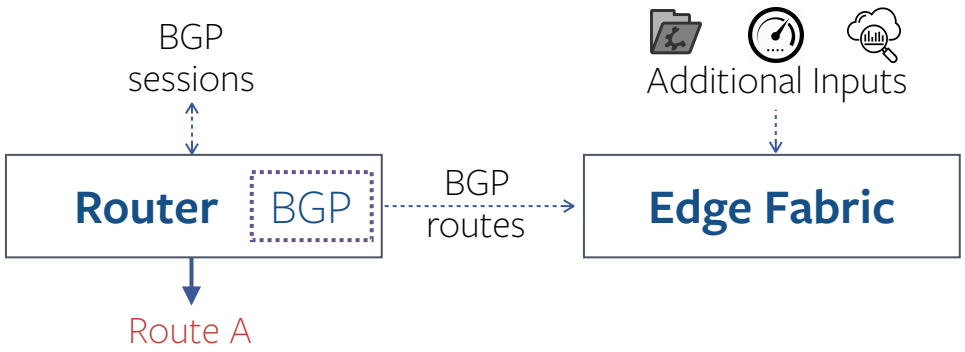
1 Router selects routes using BGP



Edge Fabric's Approach to Control

1 Router selects routes using BGP

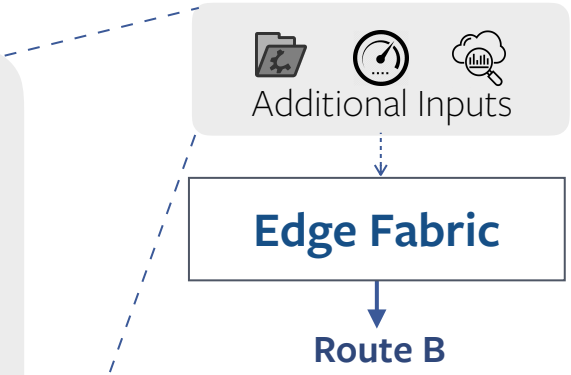
2 **Edge Fabric** selects ideal routes using BGP routes + other inputs



Edge Fabric's Approach to Control

Inputs to Edge Fabric

- BGP routes (from router)
- Advanced policy
- 1 Gbps Prefix traffic rates
- 40 Gbps Circuit capacities
- Route performance measurements

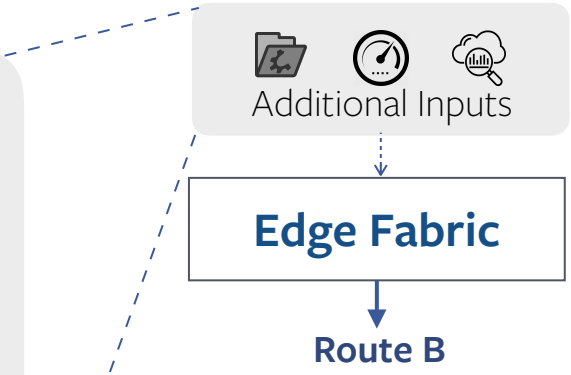


51

Edge Fabric's Approach to Control

Inputs to Edge Fabric

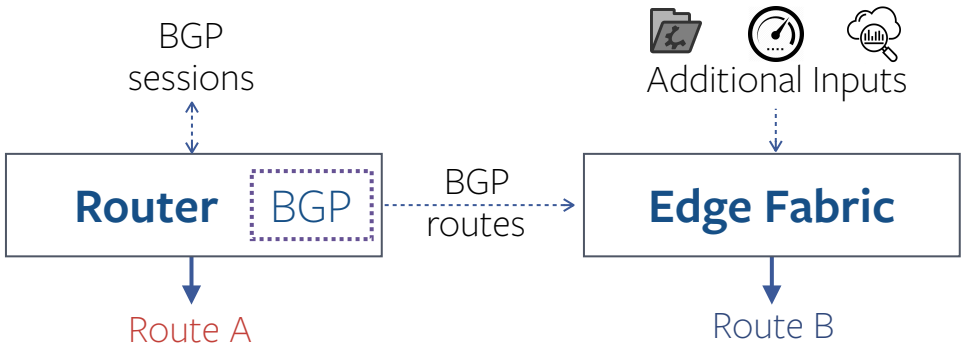
- BGP routes (from router)
- Advanced policy
- 1 Gbps Prefix traffic rates
- 40 Gbps Circuit capacities
- Route performance measurements



51

Edge Fabric's Approach to Control

1. Router selects routes using BGP
2. **Edge Fabric** selects ideal routes using BGP routes + other inputs

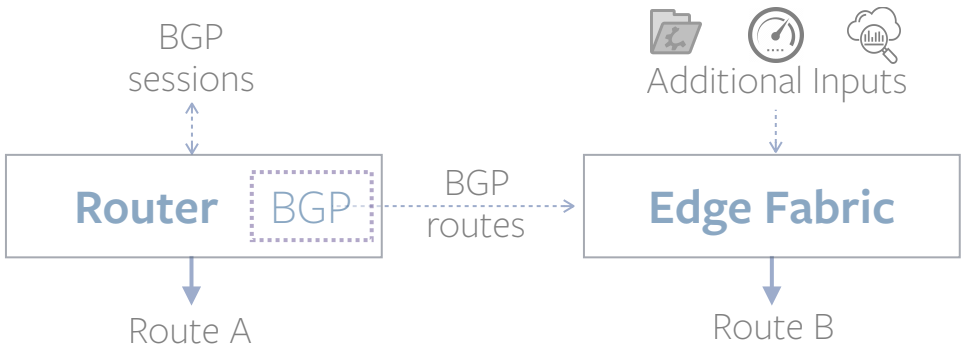


52

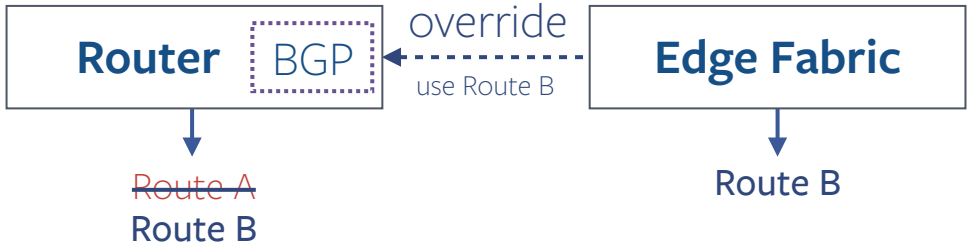
Edge Fabric's Approach to Control

1 Router selects routes using BGP

2 **Edge Fabric** selects ideal routes using BGP routes + other inputs



3 If router and Edge Fabric choose different routes, **override router**



53

Types of Edge Fabric Overrides

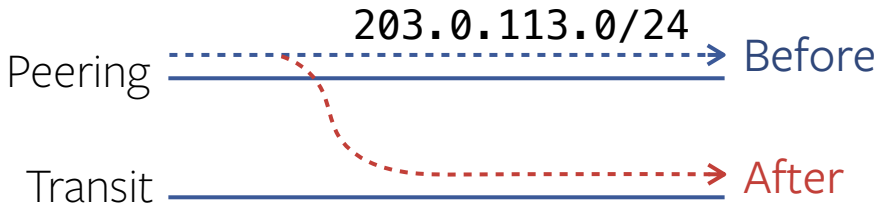
Edge Fabric can override BGP's decision in order to...

Types of Edge Fabric Overrides

Edge Fabric can override BGP's decision in order to...

Move traffic for set of end-users

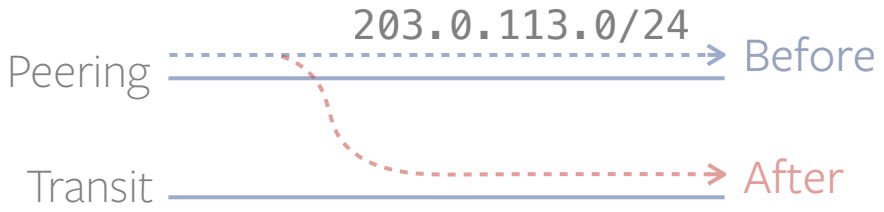
override per <destination>



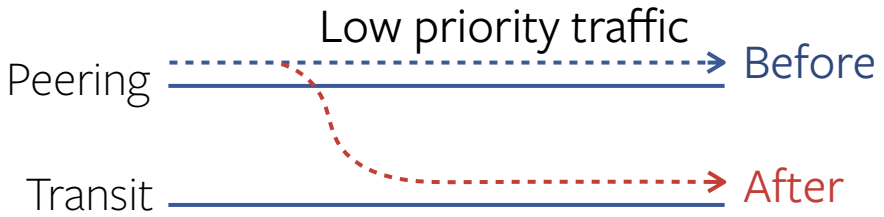
Types of Edge Fabric Overrides

Edge Fabric can override BGP's decision in order to...

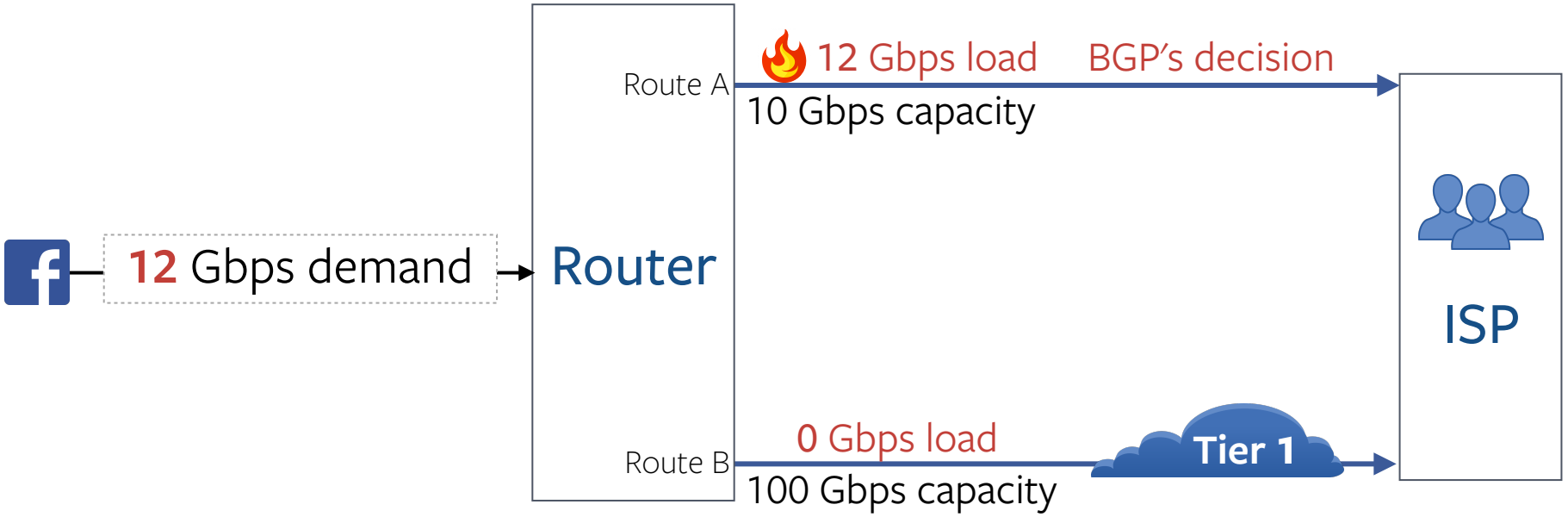
Move traffic for set of end-users
override per <destination>



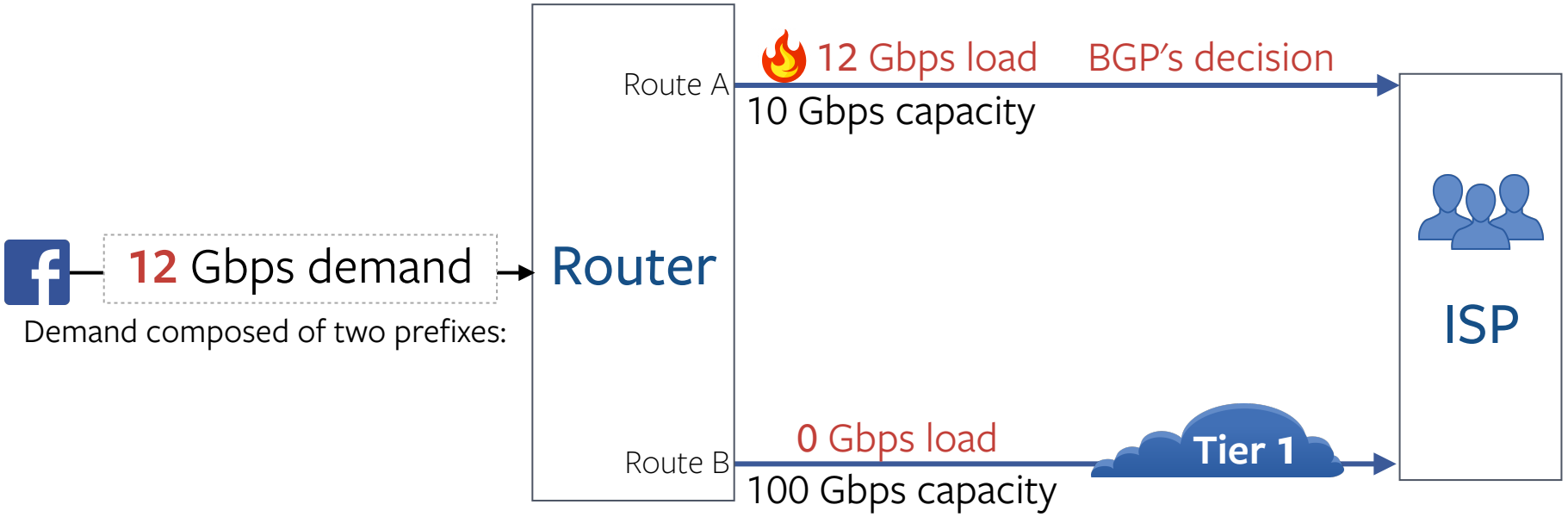
Move class of end-user traffic
override per <destination, traffic class>
(see paper for details)



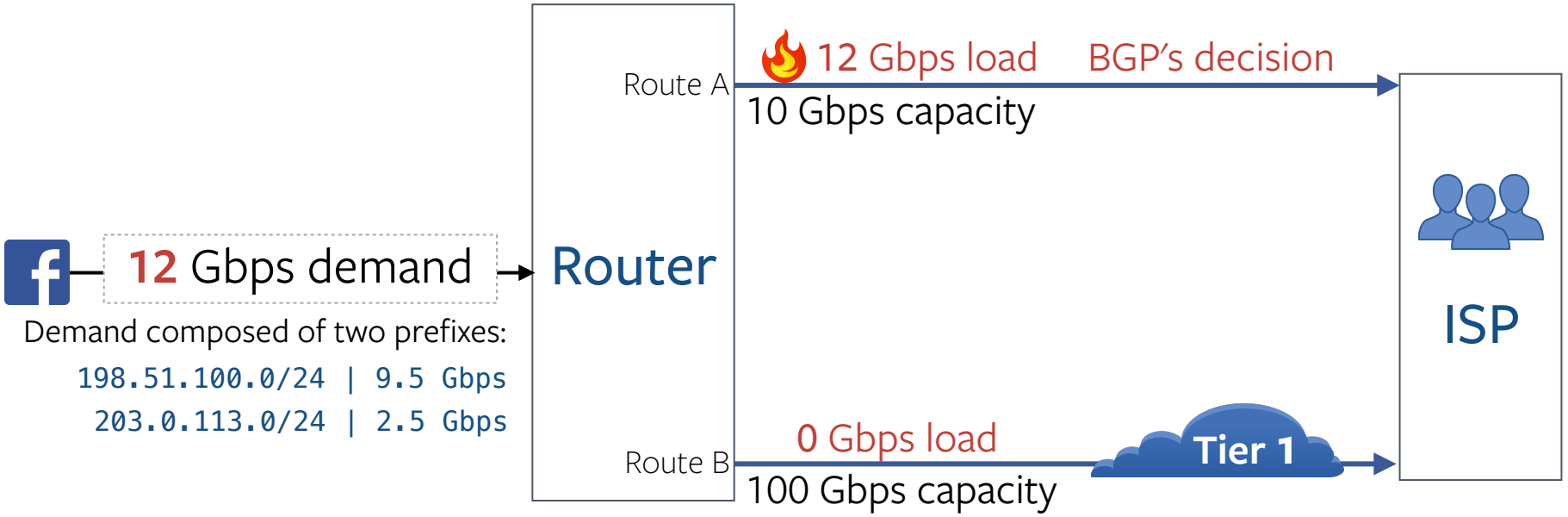
Example Override: Preventing Congestion



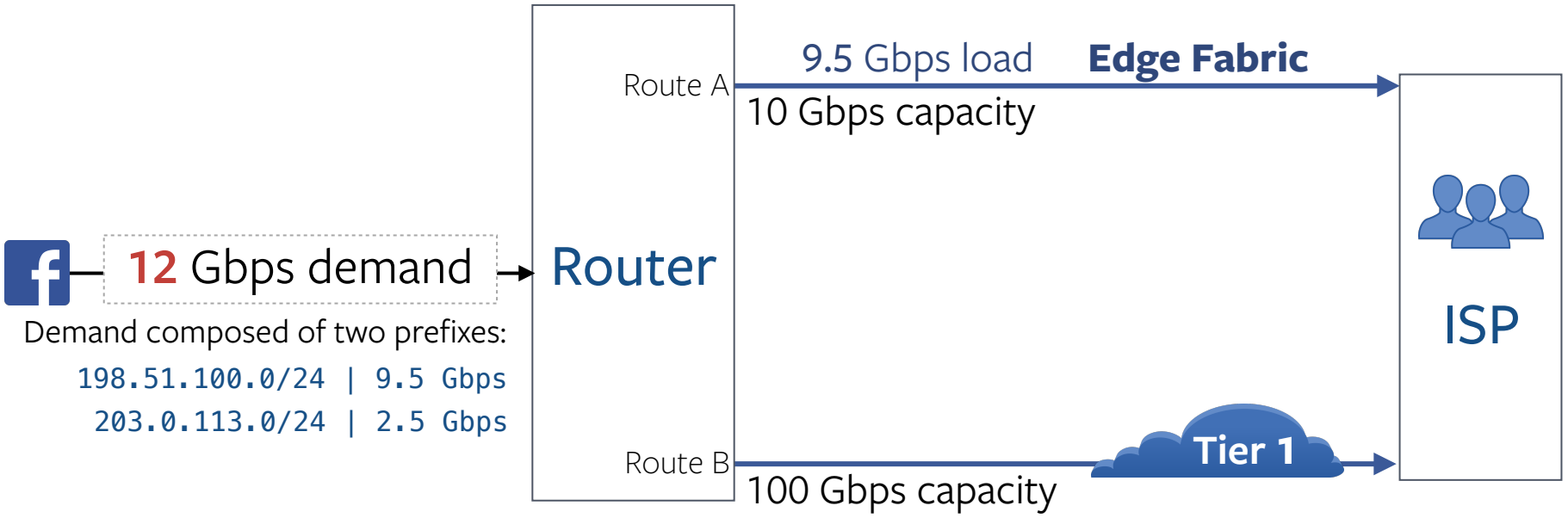
Example Override: Preventing Congestion



Example Override: Preventing Congestion

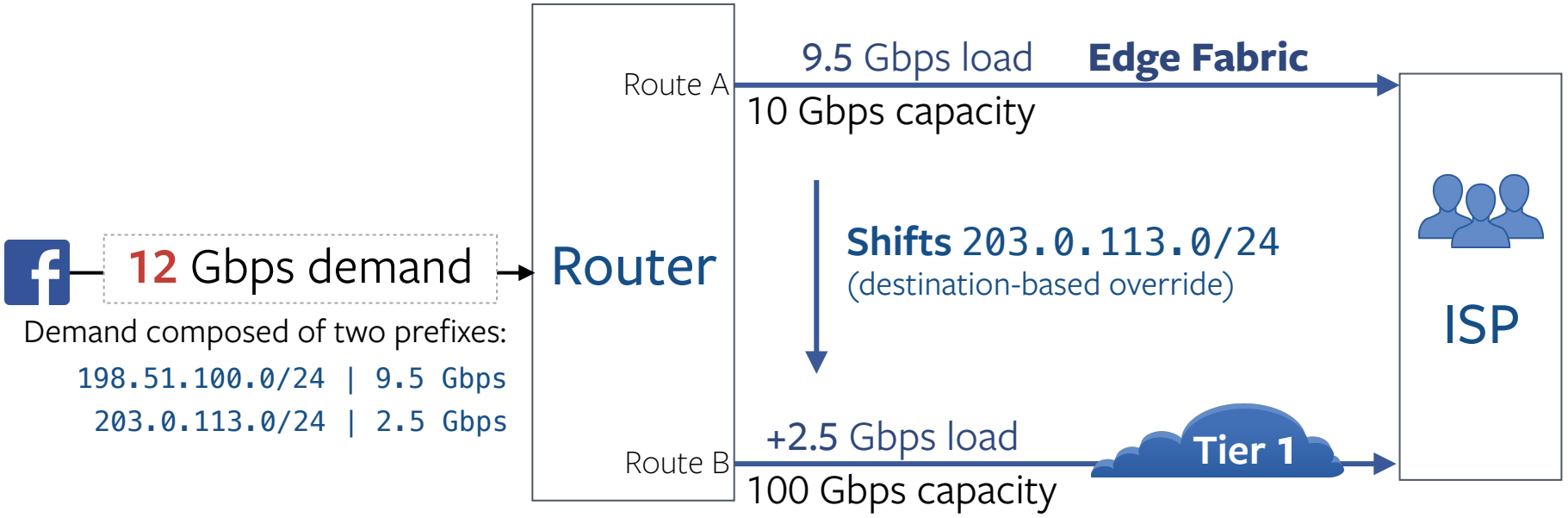


Example Override: Preventing Congestion



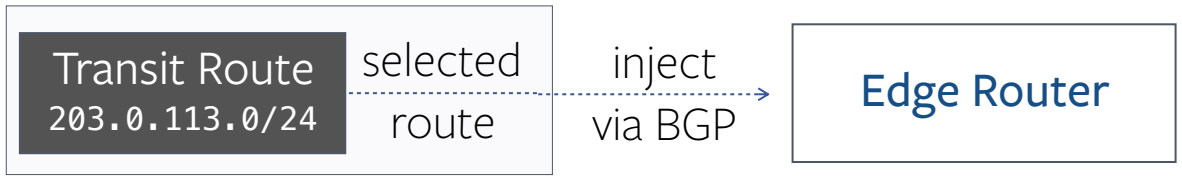
Edge Fabric shifts a prefix's traffic to an alternate link

Example Override: Preventing Congestion



Edge Fabric shifts a prefix's traffic to an alternate link

Enacting Overrides at Routers



1 **Edge Fabric** injects override route via BGP

Enacting Overrides at Routers

1 **Edge Fabric** injects override route via BGP

2 BGP at routers prefers routes from **Edge Fabric**

Enacting Overrides at Routers

Edge Fabric monitors BGP's decisions and overrides them as needed

We gain centralized control over the distributed BGP process without removing BGP from our routers

Chapter 5: outline

5.1 introduction

5.2 routing protocols

- link state
- distance vector

5.3 intra-AS routing in the Internet: OSPF

5.4 routing among the ISPs: BGP

5.5 The SDN control plane

5.6 ICMP: The Internet Control Message Protocol

5.7 Network management and SNMP

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)
- network-layer "above" IP:
  - ICMP msgs carried in IP datagrams
- ICMP message: type, code plus first 8 bytes of IP datagram causing error

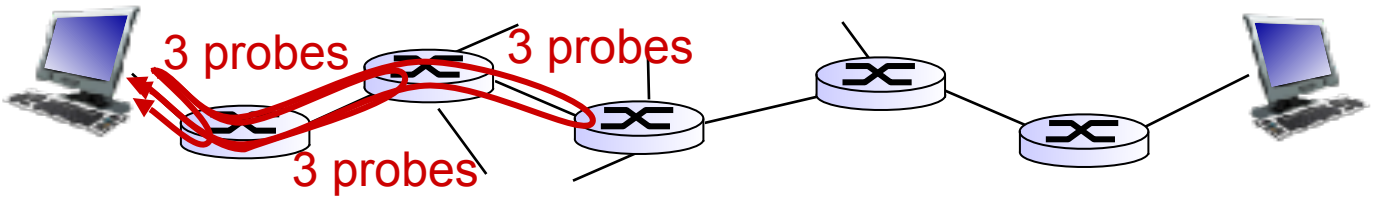
Type	Code	description
0	0	echo reply (ping)
3	0	dest. network unreachable
3	1	dest host unreachable
3	2	dest protocol unreachable
3	3	dest port unreachable
3	6	dest network unknown
3	7	dest host unknown
4	0	source quench (congestion control - not used)
8	0	echo request (ping)
9	0	route advertisement
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.
  - port number unlikely to be in use
- when datagram in *n*th set arrives to nth 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



Chapter 5: outline

5.1 introduction

5.2 routing protocols

- link state
- distance vector

5.3 intra-AS routing in the Internet: OSPF

5.4 routing among the ISPs: BGP

5.5 The SDN control plane

5.6 ICMP: The Internet Control Message Protocol

5.7 Network management and SNMP

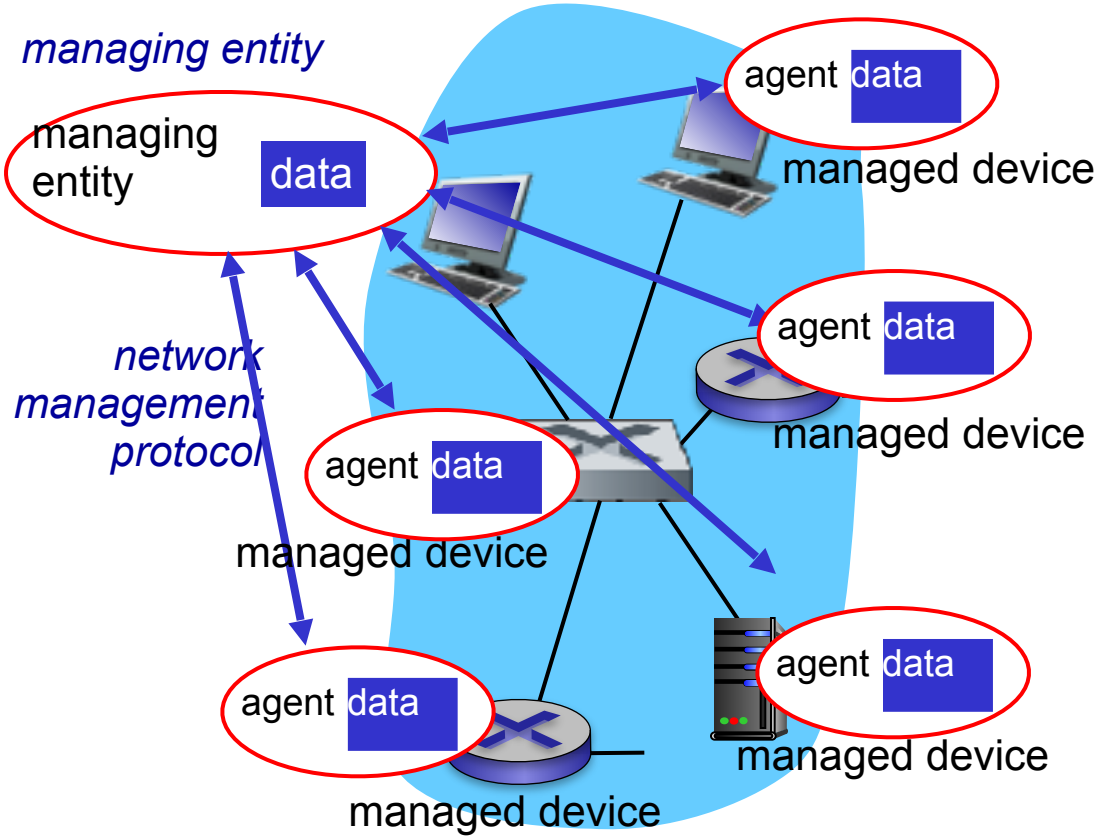
What is network management?

- autonomous systems (aka "network"): 1000s of interacting hardware/software components
- other complex systems requiring monitoring, control:
  - jet airplane
  - nuclear power plant
  - others?

"Network management includes the deployment, integration and coordination of the hardware, software, and human elements to monitor, test, poll, configure, analyze, evaluate, and control the network and element resources to meet the real-time, operational performance, and Quality of Service requirements at a reasonable cost."

Infrastructure for network management

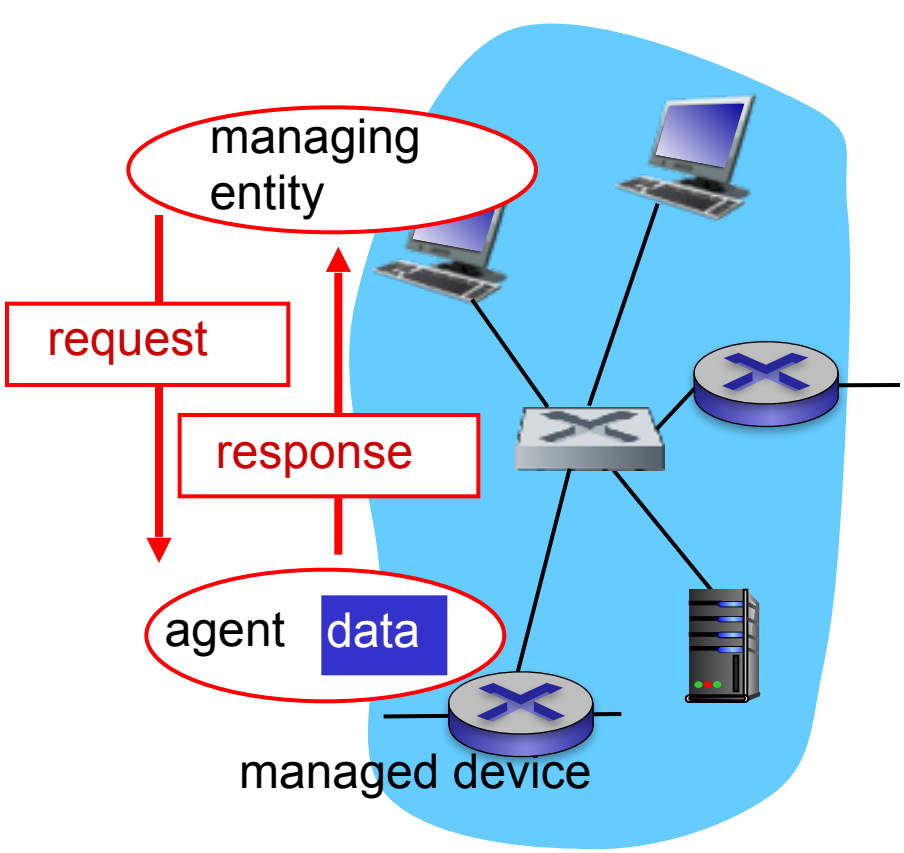
definitions:



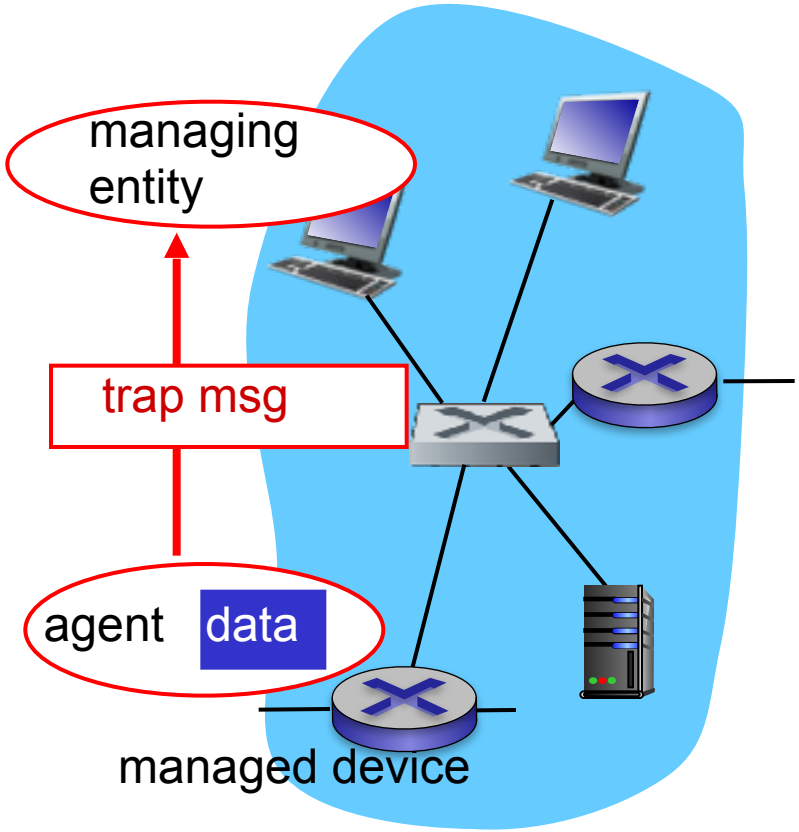
managed devices contain *managed objects* whose data is gathered into a *Management Information Base (MIB)*

SNMP protocol

Two ways to convey MIB info, commands:



request/response mode

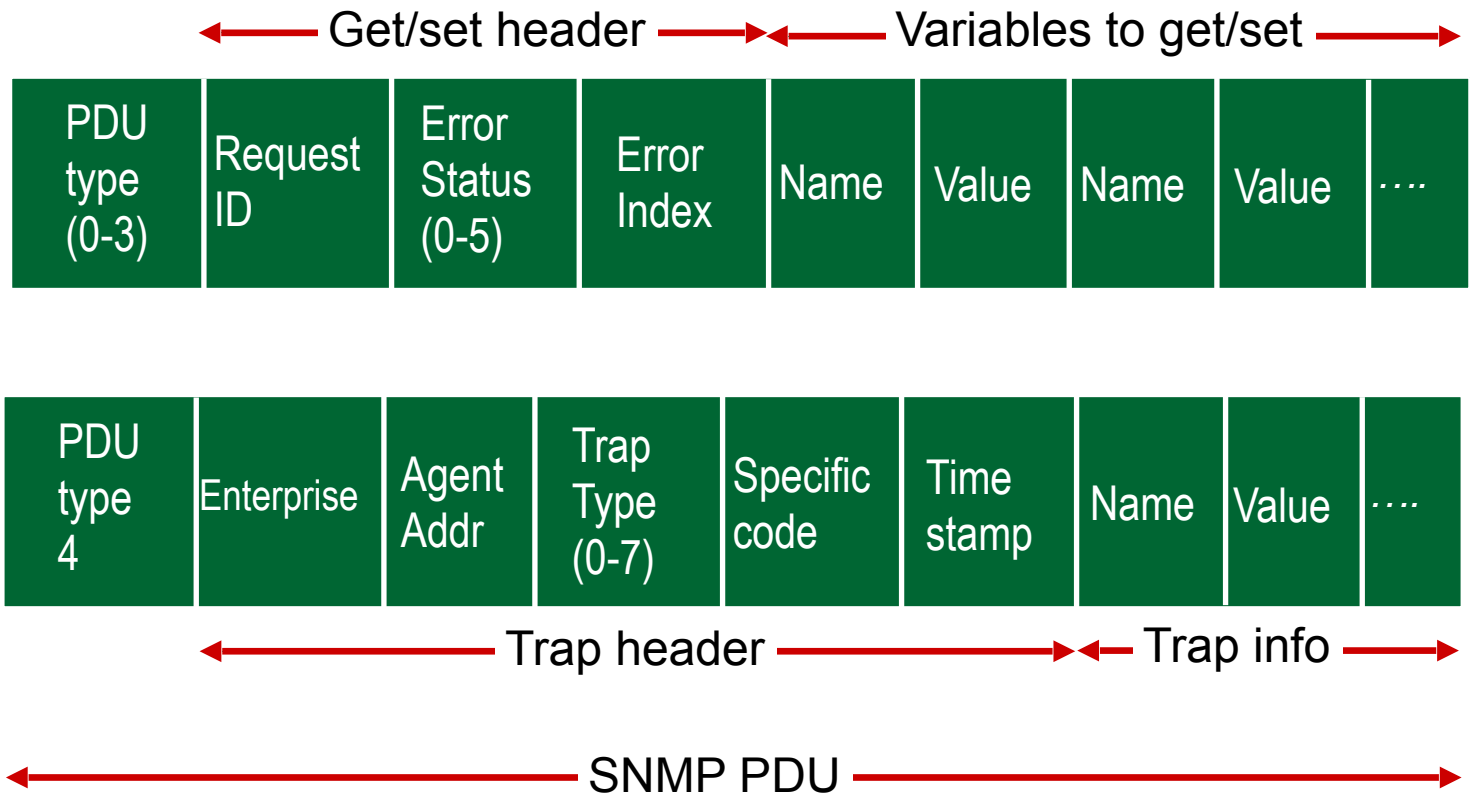


trap mode

SNMP protocol: message types

Message type	Function
GetRequest GetNextRequest GetBulkRequest	manager-to-agent: “get me data” (data instance, next data in list, block of data)
InformRequest	manager-to-manager: here’s MIB value
SetRequest	manager-to-agent: set MIB value
Response	Agent-to-manager: value, response to Request
Trap	Agent-to-manager: inform manager of exceptional event

SNMP protocol: message formats



More on network management: see earlier editions of text!

Chapter 5: summary

we've learned a lot!

- approaches to network control plane
  - per-router control (traditional)
  - logically centralized control (software defined networking)
- traditional routing algorithms
  - implementation in Internet: OSPF, BGP
- SDN controllers
  - implementation in practice: ODL, ONOS
- Internet Control Message Protocol
- network management

next stop: link layer!

Uploading course materials to sites such as CourseHero, Chegg or Github is academic misconduct at Columbia (see pg 10 of Columbia guide).