Computer Networking: A Top Down Approach
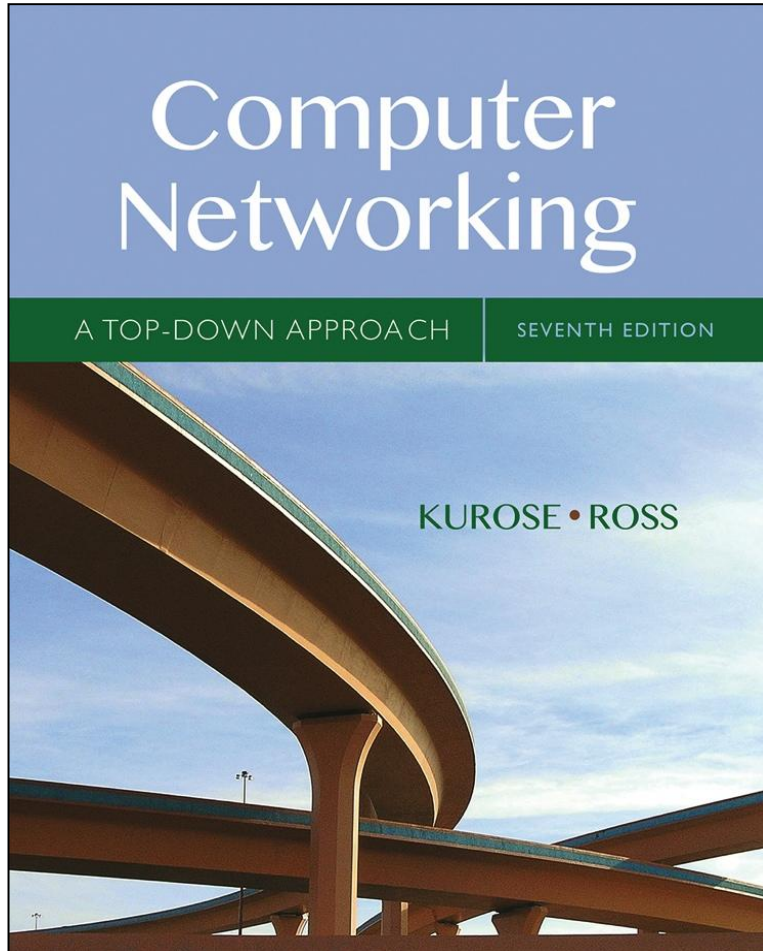
Seventh Edition

Chapter 5

The Network Layer:
Control Plane

Chapter 5: Network Layer Control Plane

chapter goals: understand principles behind network control plane

- traditional routing algorithms

- SDN controlllers

- Internet Control Message Protocol

- network management

and their instantiation, implementation in the Internet:

- OSPF, BGP, OpenFlow, ODL and ONOS controllers, ICMP, S NMP

Learning Objectives (1 of 9)

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

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)

Per-Router Control Plane

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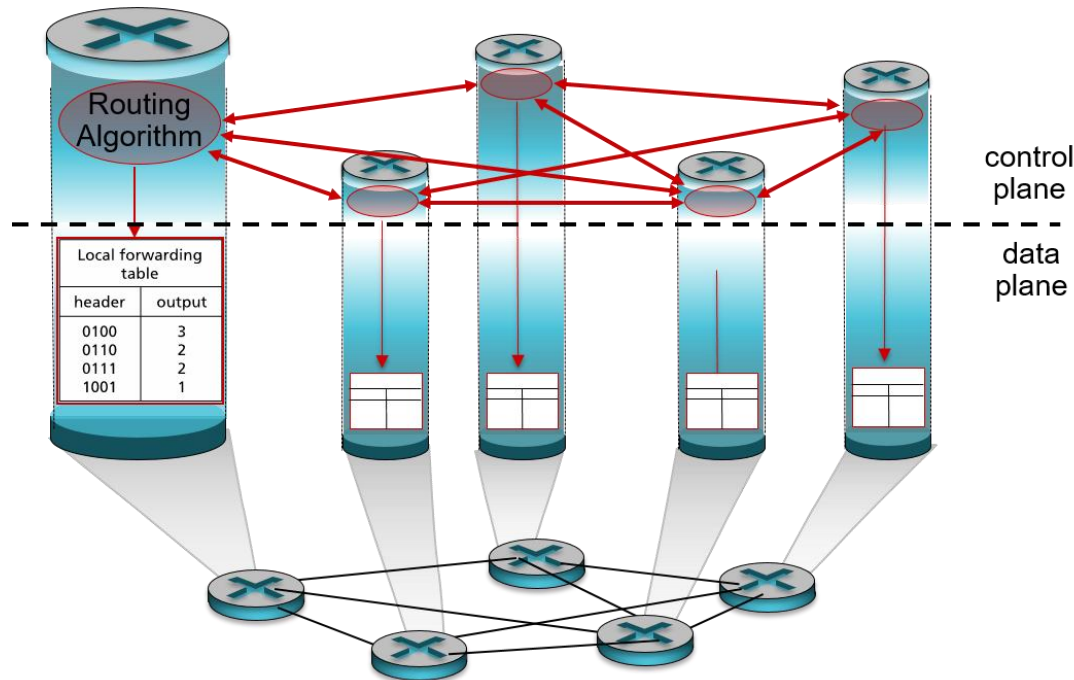
Logically Centralized Control Plane

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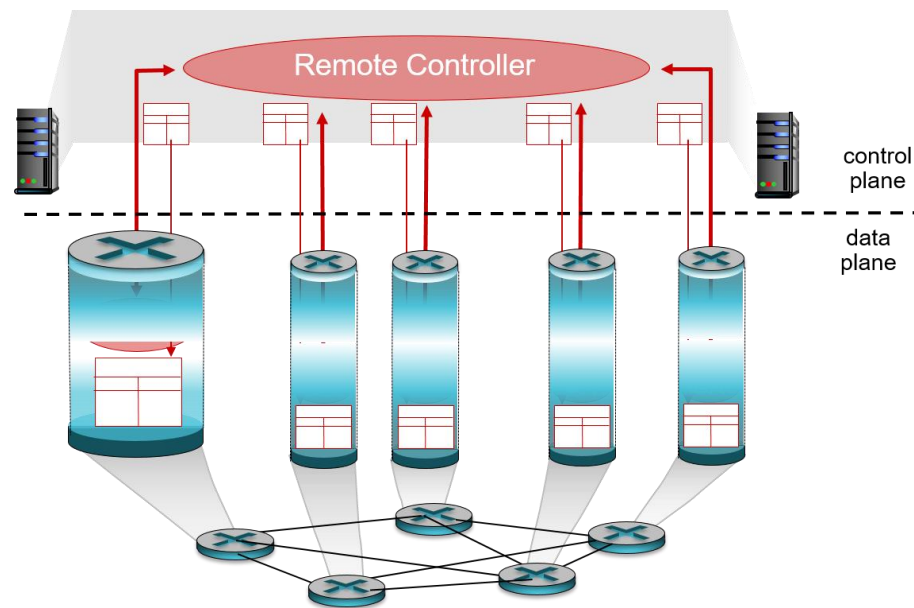
Learning Objectives

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"

- routing: a "top-10" networking challenge!
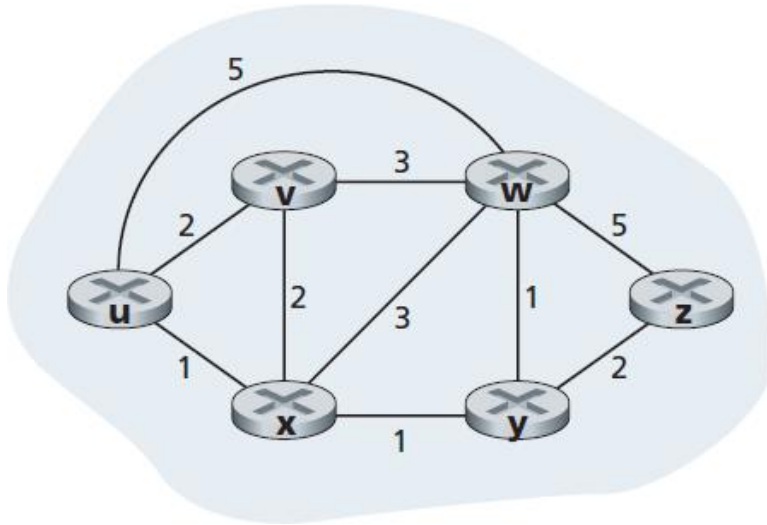
Graph Abstraction of the Network
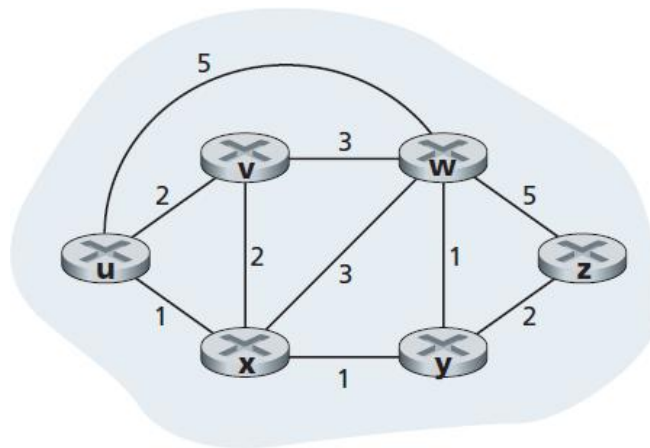
graph: G = (N, E)

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, or inversely related to bandwidth, or inversely related to congestion

$$\text{cost of path } (x_1, x_2, x_3, \ldots, x_p) = c(x_1, x_2) + c(x_2, x_3) + \ldots + c(x_{p-1}, x_p)$$

key question: what is the least-cost path between u and z?

routing algorithm: algorithm that finds that least cost path

Routing Algorithm Classification

Q: global or decentralized information?

global:

- all routers have complete topology, link cost info
- **"link state" algorithms**

decentralized:

- router knows physically-connected neighbors, link costs to neighbors
- iterative process of computation, exchange of info with neighbors
- **"distance vector" algorithms**

Q: static or dynamic?

static:

- routes change slowly over time

dynamic:

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

Learning Objectives (3 of 9)

Pearson

A Link-State Routing Algorithm

Dijkstra's algorithm

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

notation:

- **c(x,y):** link cost from node

  x to y; = ∞ if not direct neighbors

- **D(v):** current value of cost of path from source to dest. v

- **p(v):** predecessor node along path from source to v

- **N':** set of nodes whose least cost path definitively known

Dijsktra's Algorithm

1 **Initialization:**
2   $N' = \{u\}$
3   for all nodes v
4   if v adjacent to u
5   then $D(v) = c(u,v)$
6   else $D(v) = \infty$
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
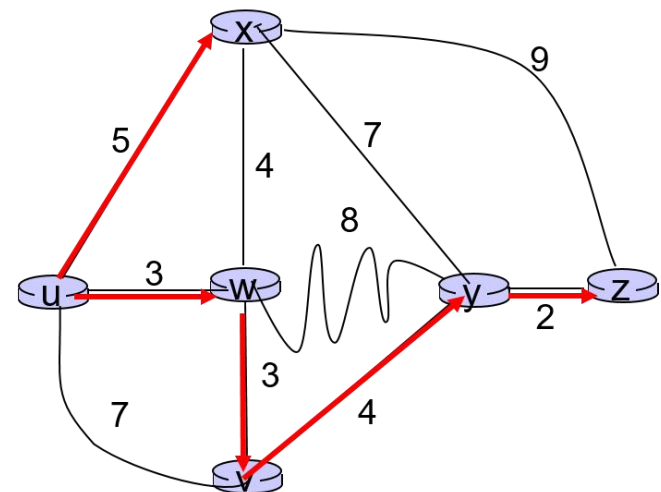14   shortest path cost to w plus cost from w to v */
15 **until all nodes in N' `**

Pearson

Dijkstra's Algorithm: Example

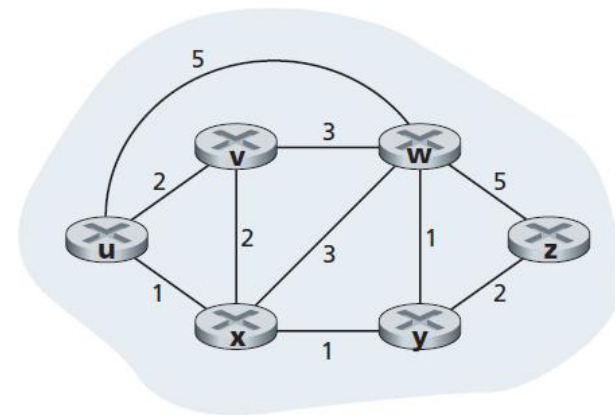| Step	N'	D(v) p(v)	D(**w**) p(w)	D(**x**) p(x)	D(**y**) p(y)	D(**z**) p(z)
0	u	7,u	(3,u)	5,u	∞	∞
1	uw	6,w		(5,u)	11,w	∞
2	uwx	(6,w)			11,w	14,x
3	uwxv				(10,v)	14,x
4	uwxvy					12,y
5	uwxvyz					

notes:

- construct shortest path tree by tracing predecessor nodes

- ties can exist (can be broken arbitrarily)

Dijkstra's Algorithm: Another Example

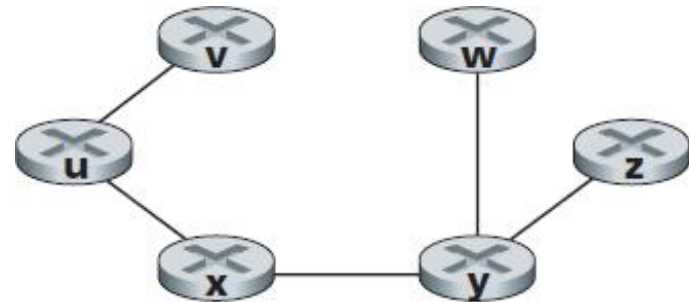| Step	N'	D(v),p(v)	D(w),p(w)	D(x),p(x)	D(y),p(y)	D(z),p(z)
0	u	2,u	5,u	1,u	∞	∞
1	ux	2,u	4,x		2,x	∞
2	uxy	2,u	3,y			4,y
3	uxyv		3,y			4,y
4	uxyvw			3,y		4,y
5	uxyvwz					

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

Dijkstra's Algorithm: Example

- resulting forwarding table in u:

resulting shortest-path tree from u:

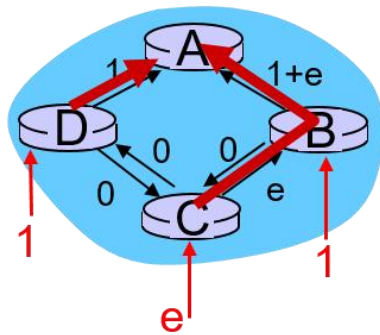| destination	Link
V	(u,v)
X	(u,x)
Y	(u,x)
W	(u,x)
Z	(u,x)

Dijkstra's Algorithm, Discussion

algorithm complexity: n nodes

- each iteration: need to check all nodes, w, not in N

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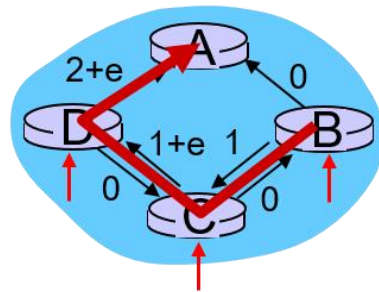
- more efficient implementations possible: $O(n\log n)$

oscillations possible:

- e.g., support link cost equals amount of carried traffic:
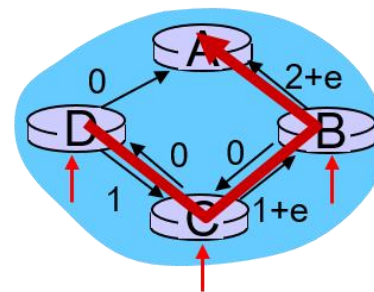
Dijkstra's Algorithm, Discussion

initially

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

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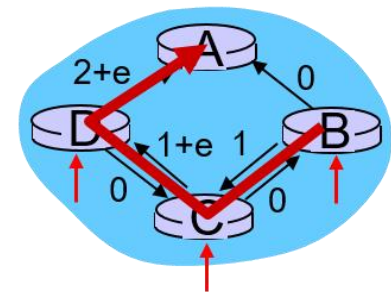
given these costs,
find new routing….
resulting in new costs

Learning Objectives

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

Bellman-Ford equation (dynamic programming)

let

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

then

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

cost from neighbor v to destination y

cost to neighbor v

min taken over all neighbors v of x

Bellman-Ford Example
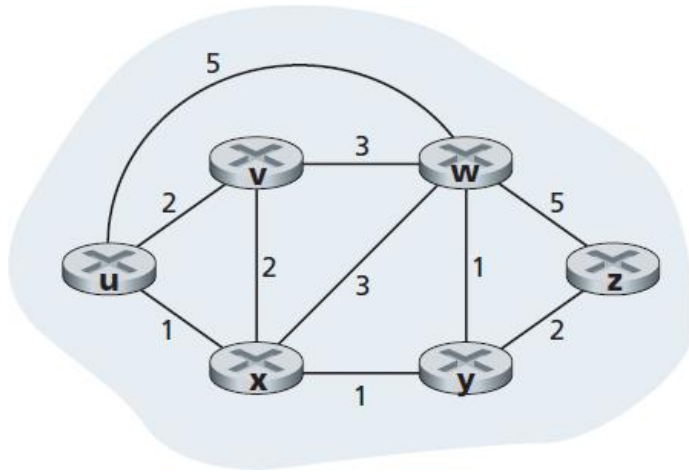
clearly, $d_v(z)=5$, $d_x(z)=3$, $d_w(z)=3$

B-F equation says:

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

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

Distance Vector Algorithm

- $D_x(y) = $ estimate of least cost from x to y
  - x maintains distance vector $D_x = \begin{bmatrix} D_x(y) : y \in N \end{bmatrix}$

- node x:
  - knows cost to each neighbor v: **c(x,v)**
  - maintains its neighbors' distance vectors. For each neighbor v, x maintains $D_v = \begin{bmatrix} D_v(y) : y \in N \end{bmatrix}$

Distance Vector Algorithm

key idea:

- from time-to-time, each node sends its own distance vector estimate to neighbors

- when x receives new DV estimate from neighbor, it updates its own DV using B-F equation:

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

- under minor, natural conditions, the estimate $D_x(y)$ converge to the actal least cost $d_x(y)$

Distance Vector Algorithm

iterative, asynchronous:
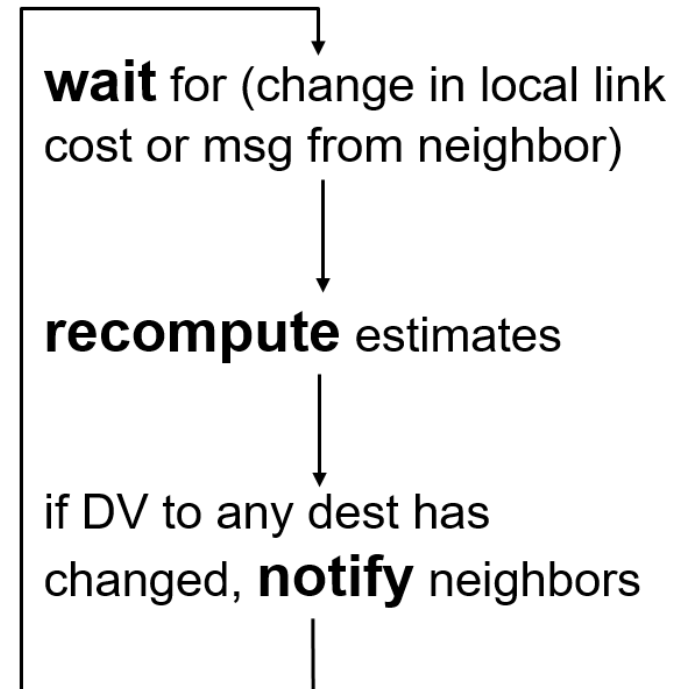each local iteration caused by:

- local link cost change
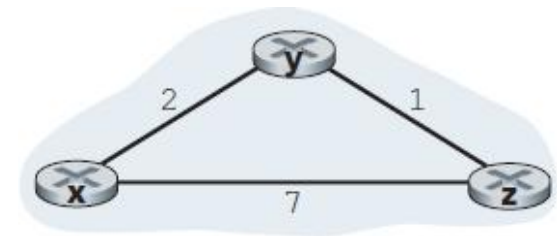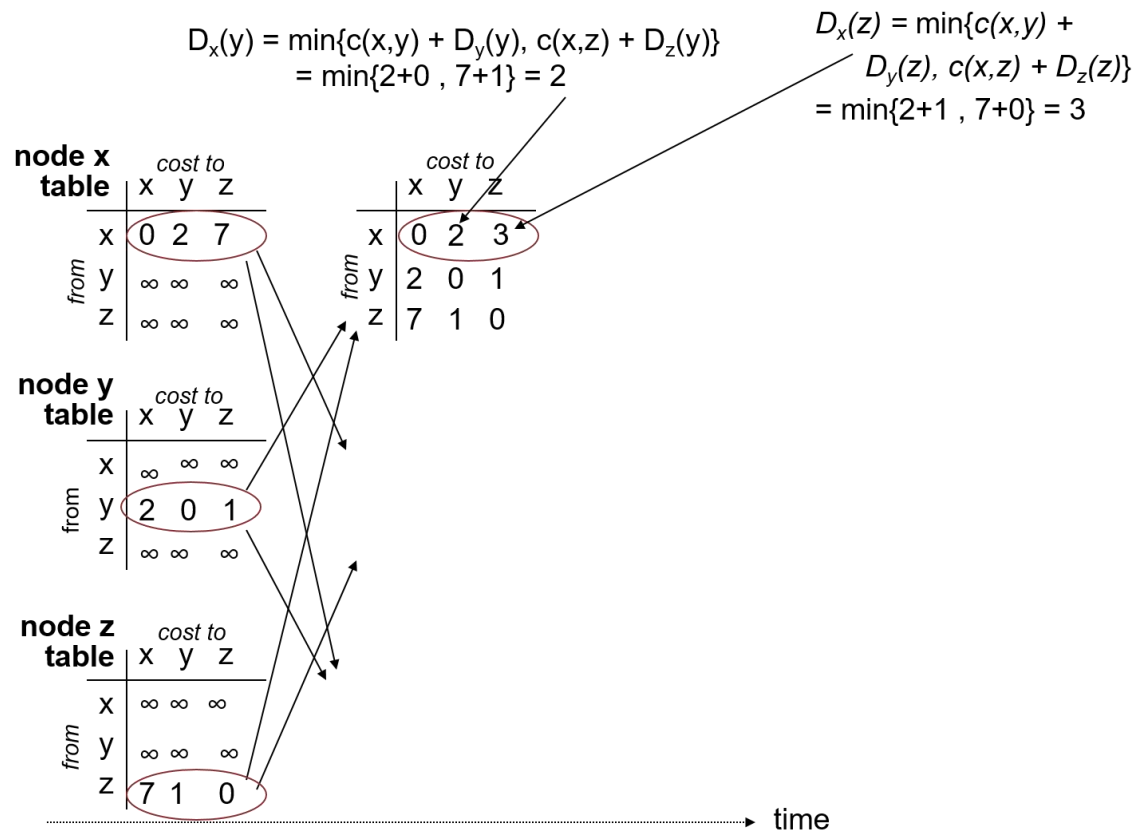
- DV update message from neighbor

distributed:

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

each node:

wait for (change in local link cost or msg from neighbor)

↓

recompute estimates

↓

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

Distance Vector Algorithm

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

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

node x table

cost to

from	x	y	z
x	0	2	7
y	∞	∞	∞
z	∞	∞	∞

cost to

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

node y table

cost to

from	x	y	z
x	∞	∞	∞
y	2	0	1
z	∞	∞	∞

node z table

cost to

from	x	y	z
x	∞	∞	∞
y	∞	∞	∞
z	7	1	0

time

Distance Vector Algorithm

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

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

node x table

cost to
	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

from

node y table

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

from

node z table

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

from

time

Distance Vector: Link Cost Changes

link cost changes:

- node detects local link cost change

- updates routing info, recalculates distance vector

- if DV changes, notify neighbors
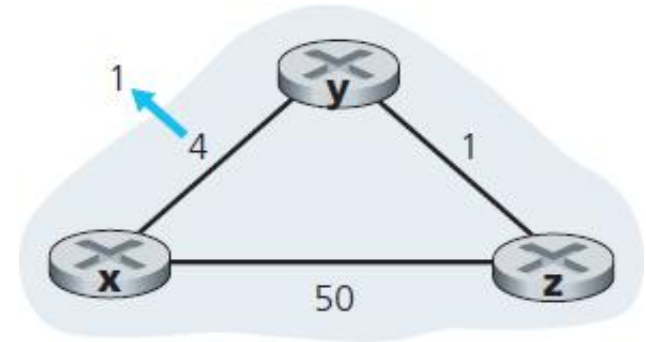
"good news travels fast"

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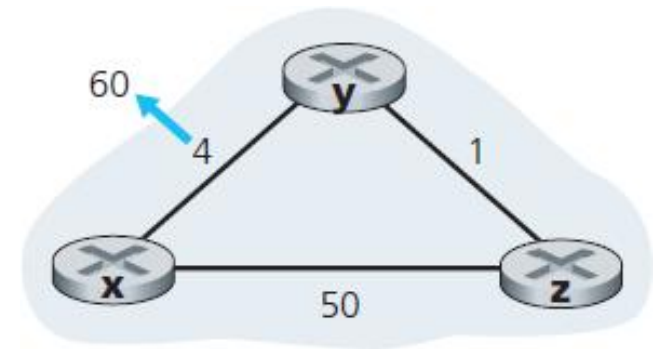
* Check out the online interactive exercises for more examples: http://gaia.cs.umass.edu/kurose_ross/interactive/

Distance Vector: Link Cost Changes

link cost changes:

- node detects local link cost change

- **bad news travels slow** – "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 requires O(nE) msgs
  - may have oscillations
- **DV:** convergence time varies
  - may be routing loops
  - count-to-infinity problem

robustness: what happens if router malfunctions?

LS:

- node can advertise incorrect **link** cost
- each node computes only its **own** table

DV:

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

Learning Objectives (5 of 9)

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 study thus far - idealized

- all routers identical

- network "flat"

- **…** **not** true in practice

scale: with billions of destinations:

- can't store all destinations in routing tables!

- routing table exchange would swamp links!

administrative autonomy

- internet = network of networks

- each network admin may want to control routing in its own network

Internet Approach to Scalable Routing

aggregate routers into regions known as **"autonomous systems" (AS) (a.k.a. "domains")**

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

inter-AS routing

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

Interconnected ASes

- forwarding table configured by both intra- and inter-AS routing algorithm

  - intra-AS routing determine entries for destinations within AS

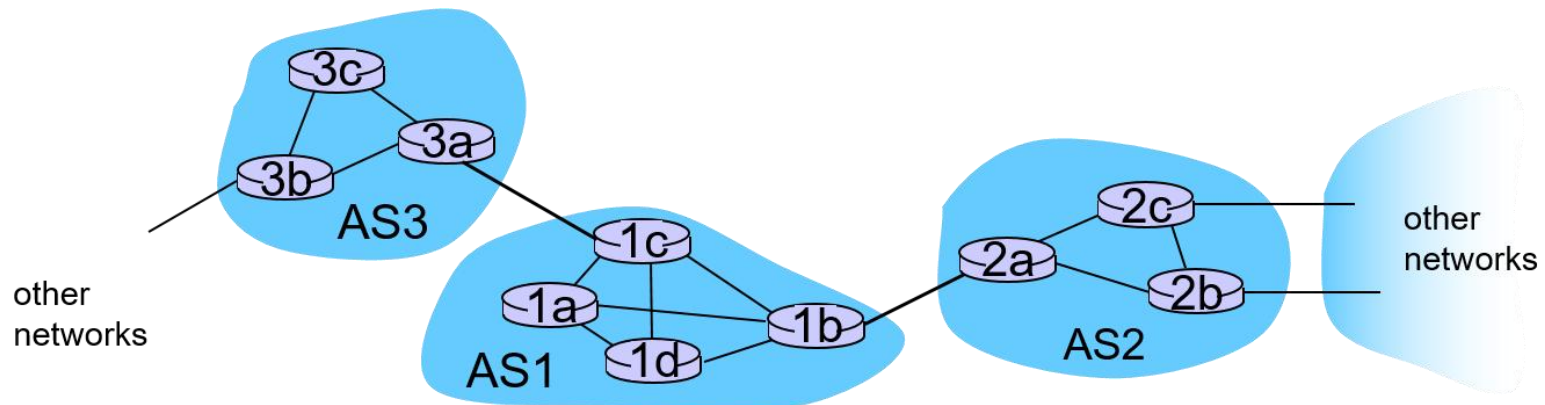 - inter-AS & intra-AS determine entries for external destinations
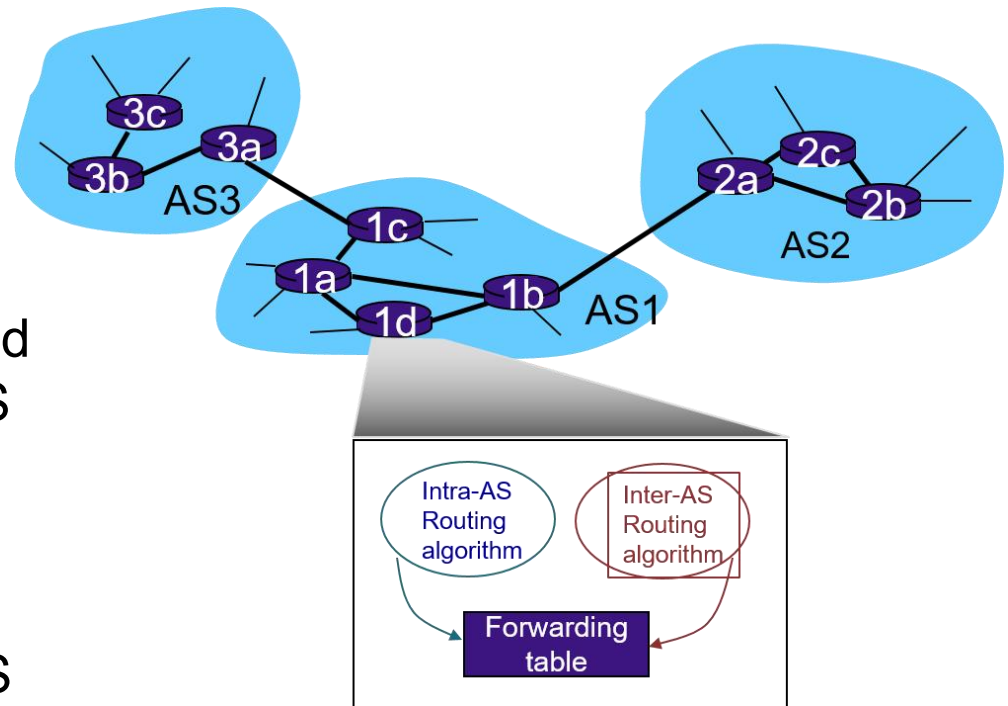
Inter-AS Tasks

- 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)**

- most common intra-AS routing protocols:
  - RIP: Routing Information Protocol
  - OSPF: Open Shortest Path First (IS-IS protocol essentially same as OSPF)
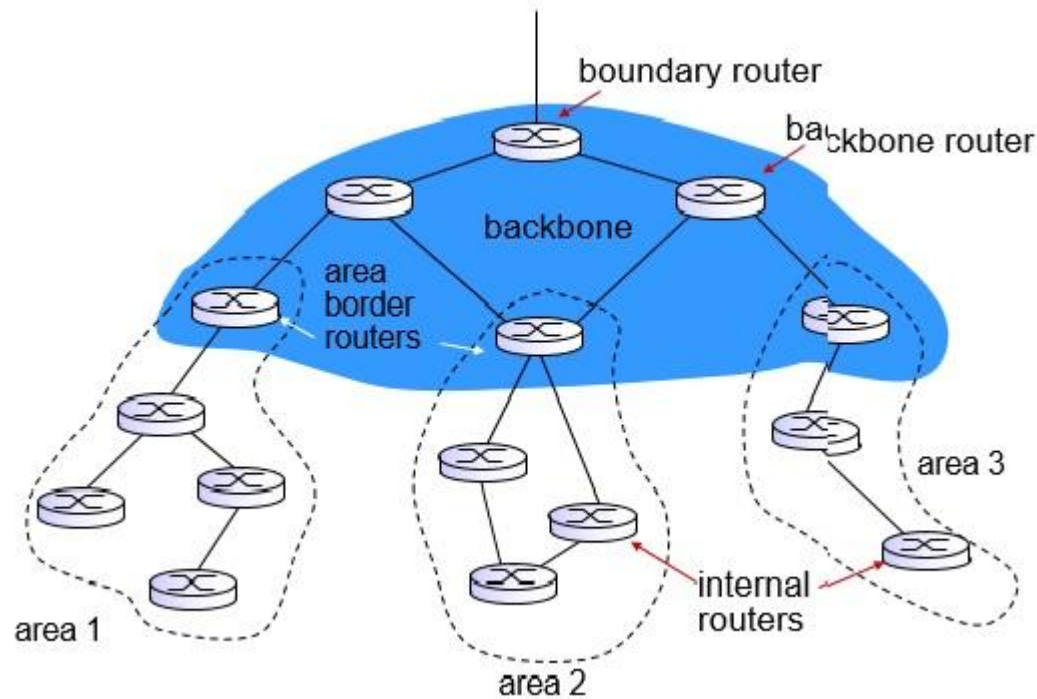 - IGRP: Interior Gateway Routing Protocol (Cisco proprietary for decades, until 2016)

OSPF (Open Shortest Path First)

- "open": publicly available

- uses link-state algorithm
  - link state packet dissemination
  - topology map at each node
  - route computation using Dijkstra's algorithm

- router floods OSPF link-state advertisements to all other routers in **entire** AS
  - carried in OSPF messages directly over IP (rather than TCP or UDP
  - link state: for each attached link

- **IS - IS routing** protocol: nearly identical to OSPF

OSPF "Advanced" Features

- **security***: all OSPF messages authenticated (to prevent malicious intrusion)

- **multiple** same-cost **paths** allowed (only one path in RIP)

- for each link, multiple cost metrics for different **TOS** (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.

P Pearson

Hierarchical OSPF (1 of 2)

- **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.

- **area border routers:** "summarize" distances to nets in own area, advertise to other Area Border routers.

- **backbone routers:** run OSPF routing limited to backbone.

- **boundary routers:** connect to other AS'es.

Learning Objective

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 de facto inter-domain routing protocol
  - "glue that holds the Internet together"

- B G P provides each A S a means to:
  - **eBGP:** obtain subnet reachability information from neighboring ASes
  - **iBGP:** propagate reachability information to all A S-internal routers.
  - determine "good" routes to other networks based on reachability information and **policy**

- allows subnet to advertise its existence to rest of Internet: **"I am here"**

eBGP, iBGP Connections
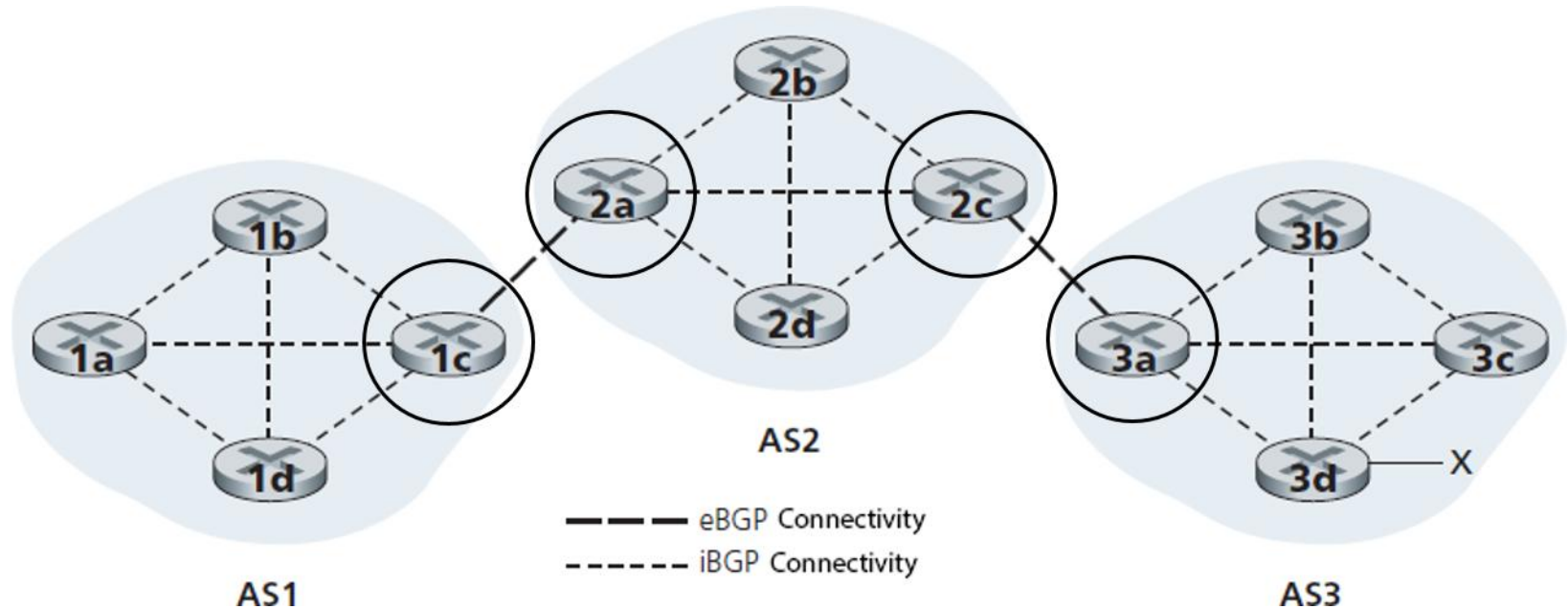
gateway routers run both eBGP and iBGP protocols

BGP Basics

- **BGP session:** two BGP routers ("peers") exchange B G P 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 **AS3 , X** to AS2 gateway router 2c:
  - AS3 **promises** to AS2 it will forward datagrams towards X

BGP advertisement:
AS3, X
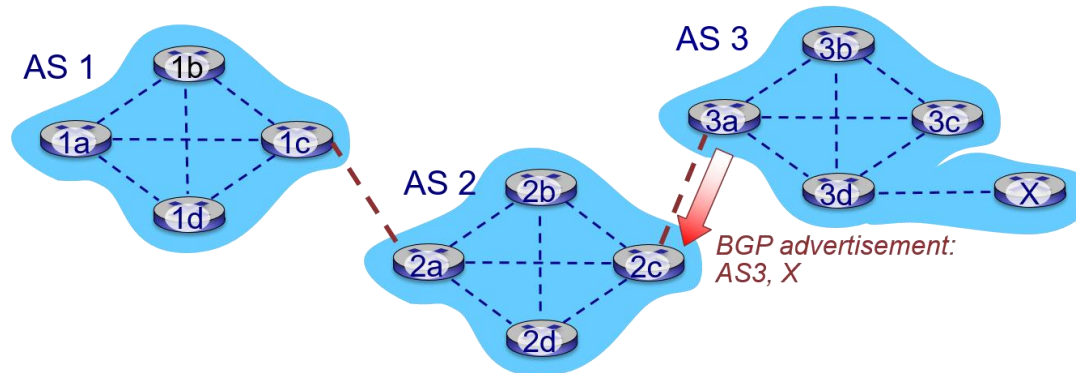
Path Attributes and BGP Routes

- advertised prefix includes BGP attributes
  - prefix + attributes = "route"

- two important attributes:
  - **AS-PATH:** list of ASes through which prefix advertisement has passed
  - **NEXT-HOP:** indicates specific internal-AS router to next-hop AS

- **Policy-based routing:**
  - gateway receiving route advertisement uses **import policy** to accept/decline path (e.g., never route through ASY).
  - AS policy also determines whether to **advertise** path to other other neighboring ASes

BGP Path Advertisement (1 of 2)
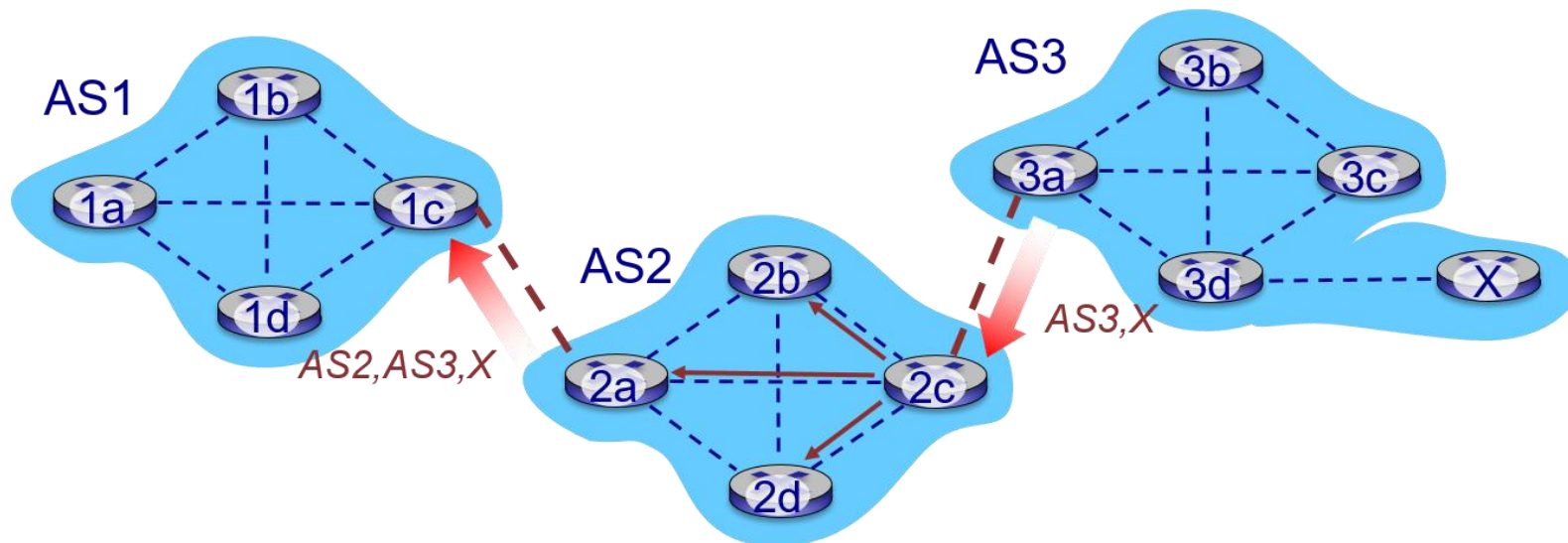
- AS2 router 2c receives path advertisement **AS3,X** (via eBGP) from AS3 router 3a

- 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 (2 of 2)
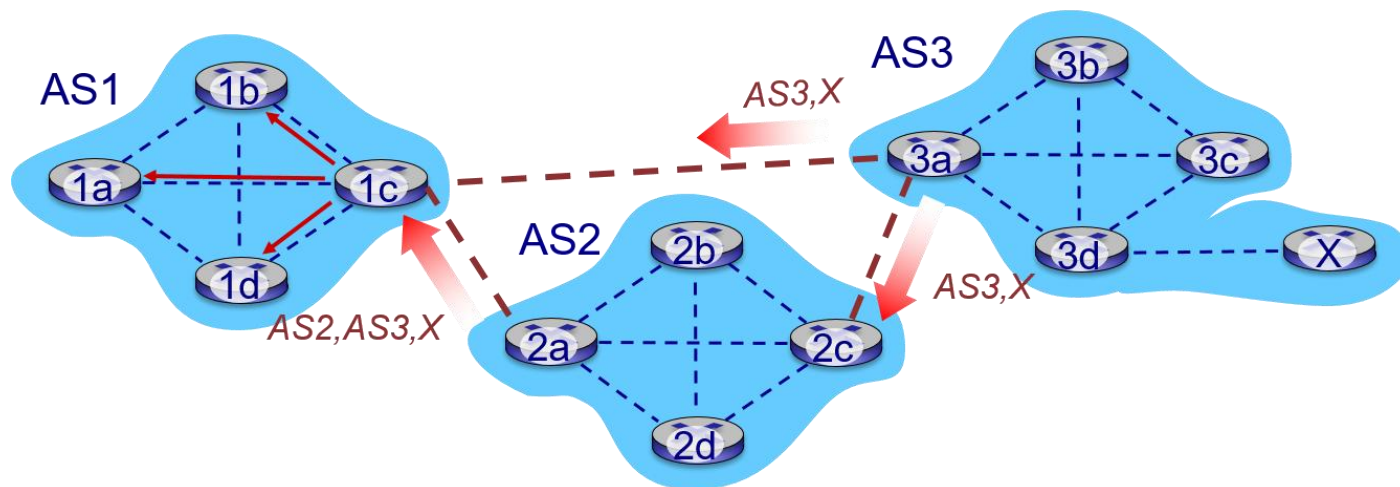
gateway router may learn about multiple paths to destination:

- AS1 gateway router 1c learns path **AS2,AS3,X** from 2a

- AS1 gateway router 1c learns path **AS3,X** from 3a

- Based on policy, AS1 gateway router 1c chooses path **AS3 , X, and advertises path within AS1 via iBGP**

BGP 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 message; also used to close connection

BGP, OSPF, Forwarding Table Entries (1 of 2)

**Q: how does router set forwarding table entry to distant prefix?**
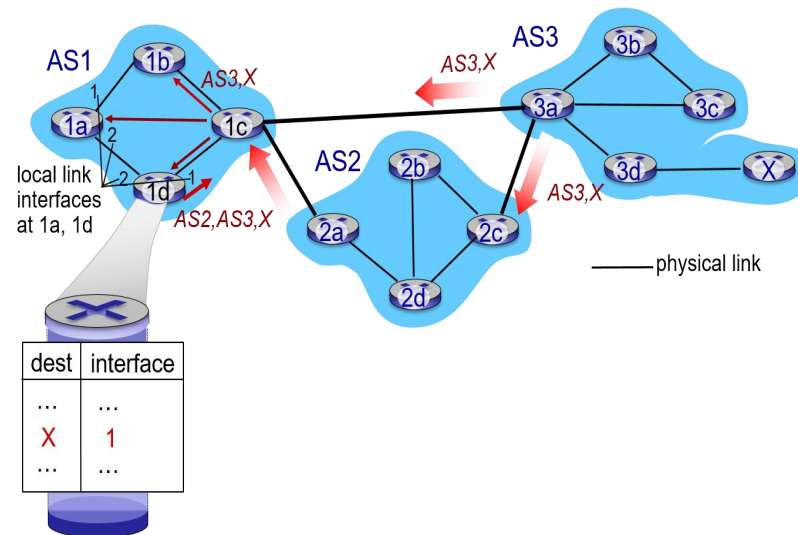
- recall: 1a, 1b, 1c learn about dest X via iBGP from 1c: "path to X goes through 1c"
- 1d: OSPF intra-domain routing: to get to 1c, forward over outgoing local interface 1

BGP, OSPF, Forwarding Table Entries (2 of 2)

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

- recall: 1a, 1b, 1c learn about dest X via iBGP from 1c: "path to X goes through 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
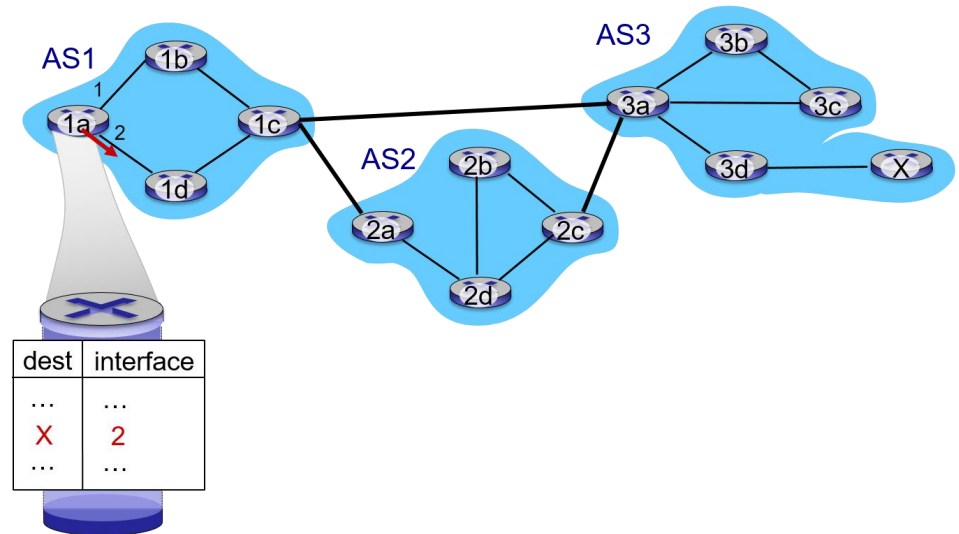
BGP Route Selection

- router may learn about more than one route to destination AS, selects route based on:

  1. local preference value attribute: policy decision
  2. shortest AS-PATH
  3. closest NEXT-HOP router: hot potato routing
  4. additional criteria

Hot Potato Routing
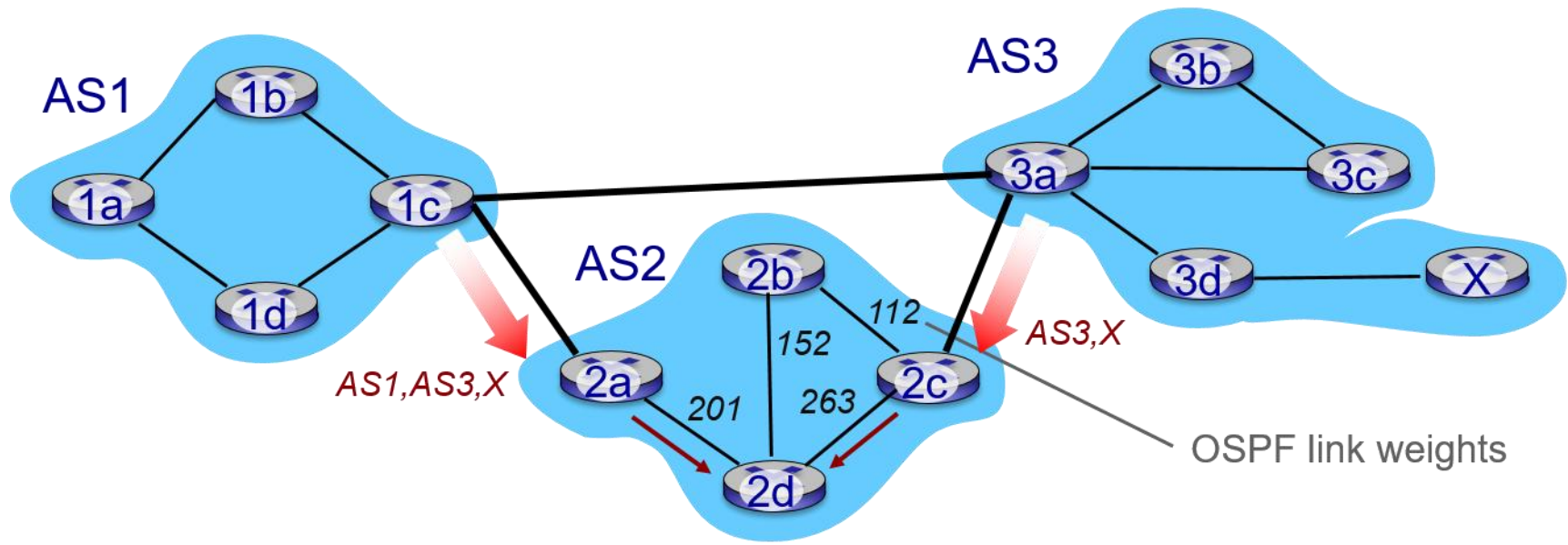
- 2d learns (via iBGP) it can route to X via 2a or 2c

- **hot potato 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!
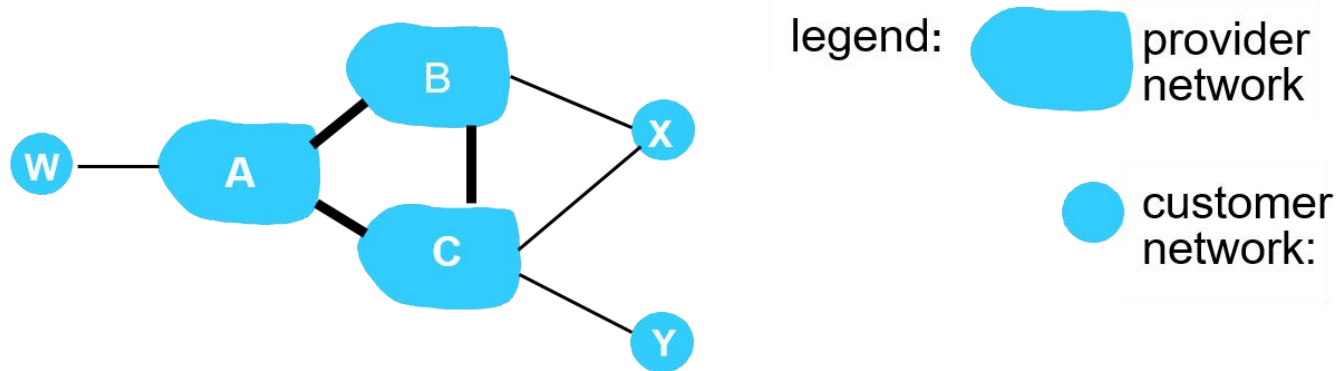
BGP: Achieving Policy Via Advertisements (1 of 2)
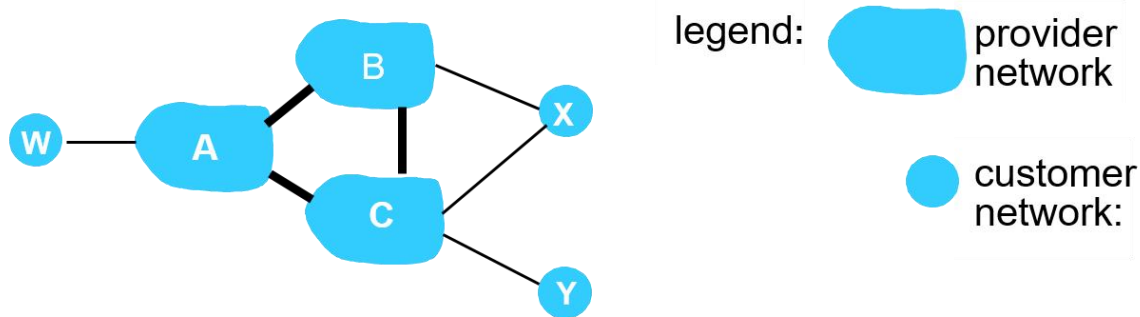
Suppose an ISP only wants to route traffic to/from its customer networks (does not want to carry transit traffic between other ISPs)

- 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 (2 of 2)

legend:
provider network

customer network:

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

P Pearson

Why Different Intra-, Inter-As Routing?

policy*:*

- inter - A S: admin wants control over how its traffic routed, who routes through its net.

- intra - A S: single admin, so no policy decisions needed

scale:

- hierarchical routing saves table size, reduced update traffic

performance:

- intra - A S: can focus on performance

- inter - A S: policy may dominate over performance

Learning Objectives

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

Software Defined Networking (SDN) (1 of 3)

- 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
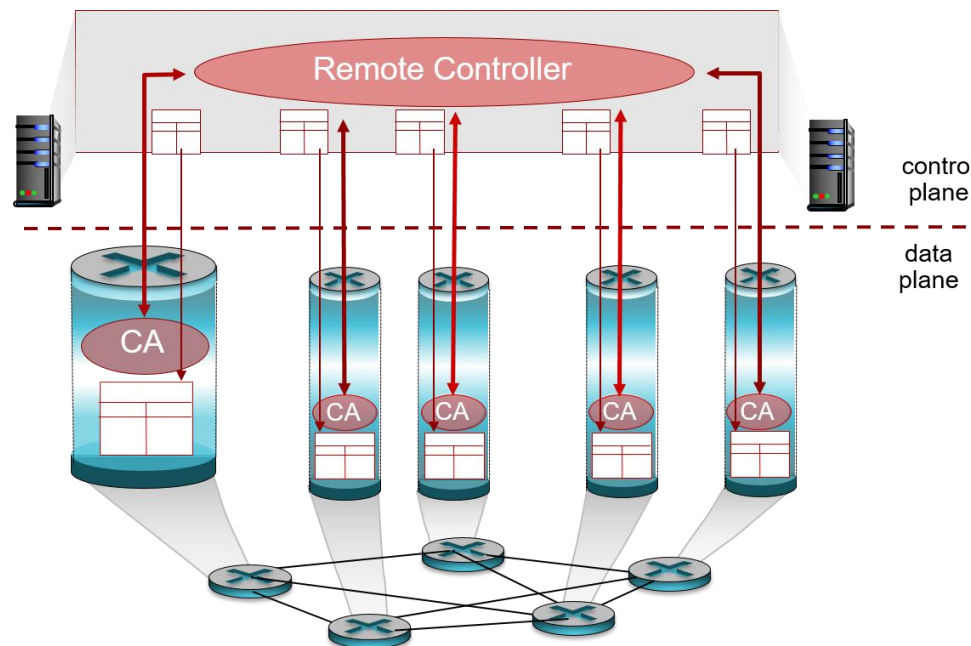
Recall: Per-Router Control Plane

Individual routing algorithm components **in each and every router** interact with each other in control plane to compute forwarding tables
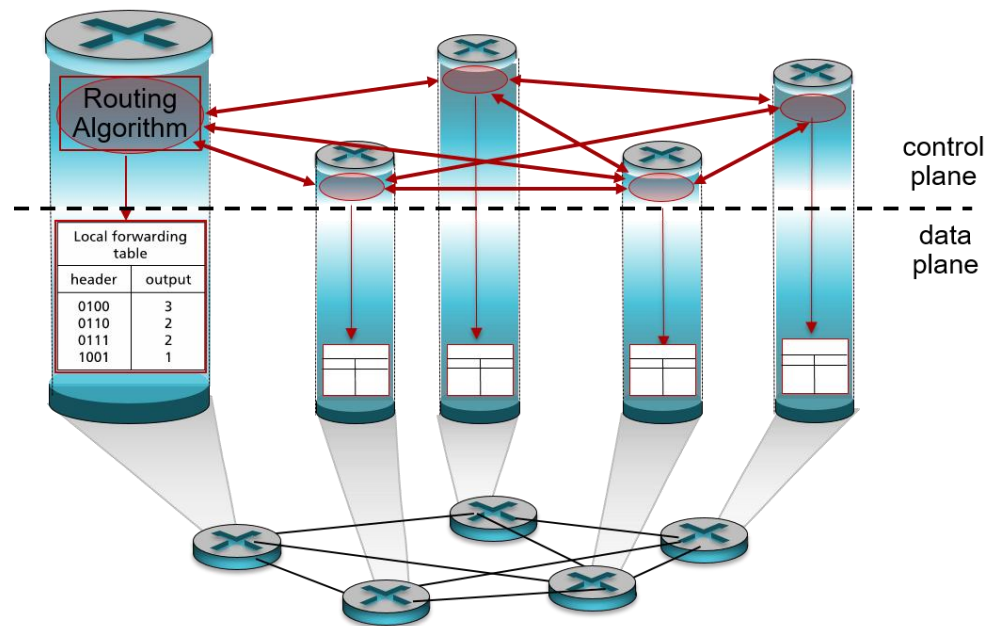
Recall: Logically Centralized Control Plane

A distinct (typically remote) controller interacts with local control agents (CAs) in routers to compute forwarding tables
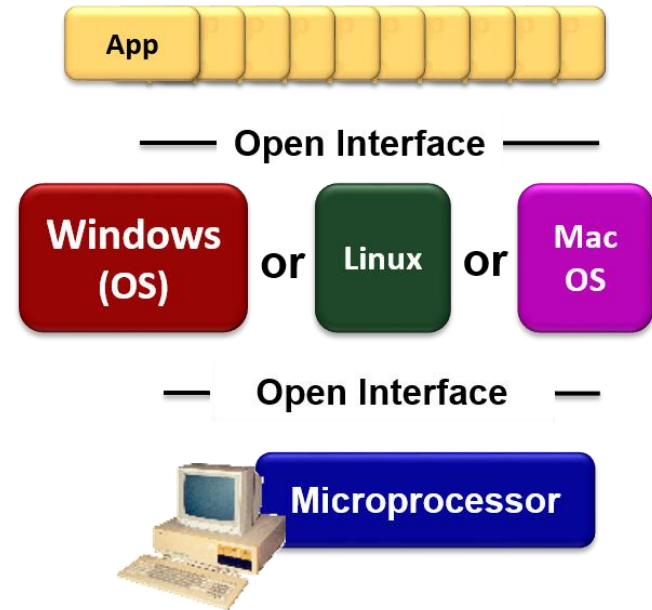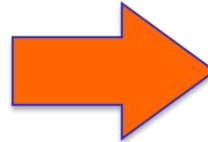
Software Defined Networking (SDN) (2 of 3)

Why a **logically centralized** control plane?

- easier network management: avoid router misconfigurations, greater flexibility of traffic flows

- table-based forwarding (recall 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

Pearson

Analogy: Mainframe to PC Evolution*

Specialized Applications

Specialized Operating System

Specialized Hardware

App

Open Interface

Windows (OS) or **Linux** or **Mac OS**

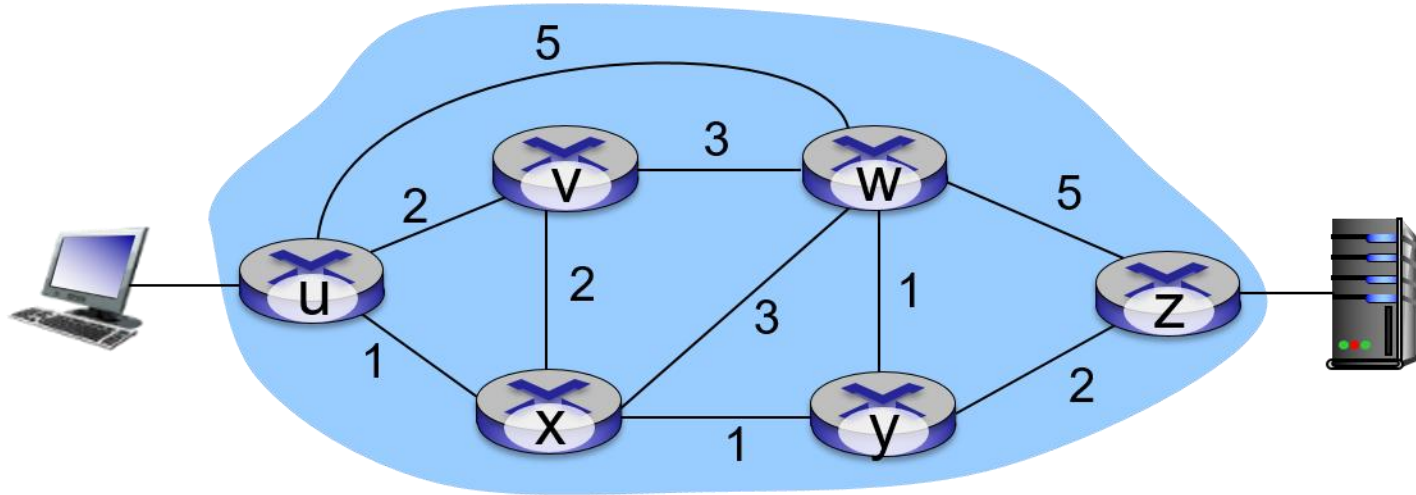Open Interface

Microprocessor

Vertically integrated
Closed, proprietary
Slow innovation
Small industry

Horizontal
Open interfaces
Rapid innovation
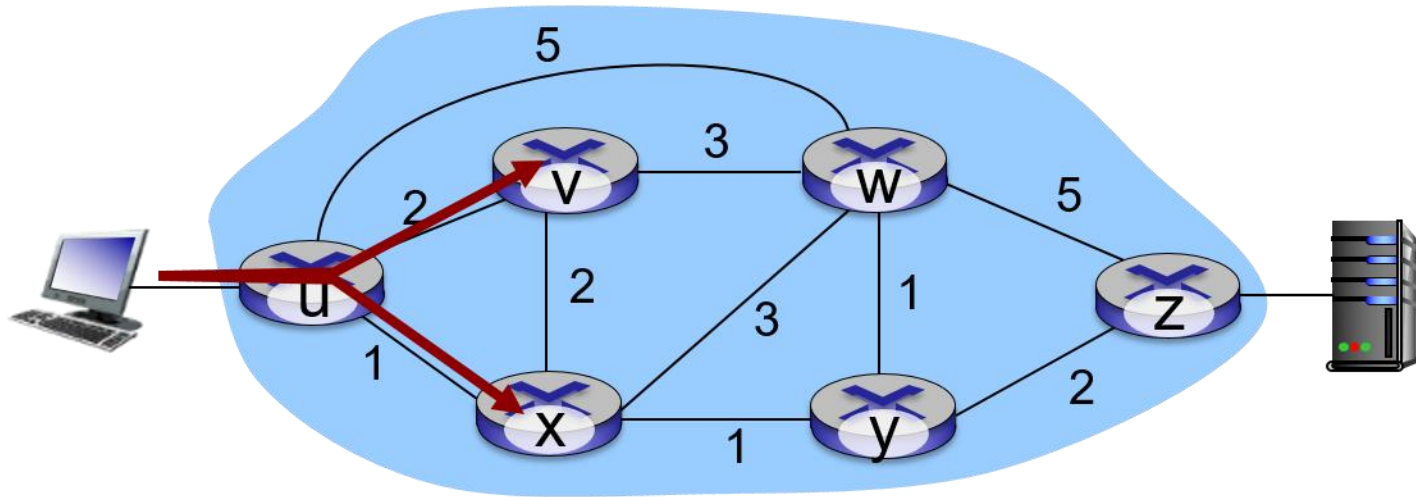Huge industry
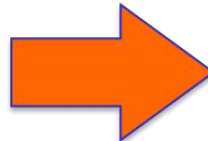
Traffic Engineering: Difficult Traditional Routing

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)!

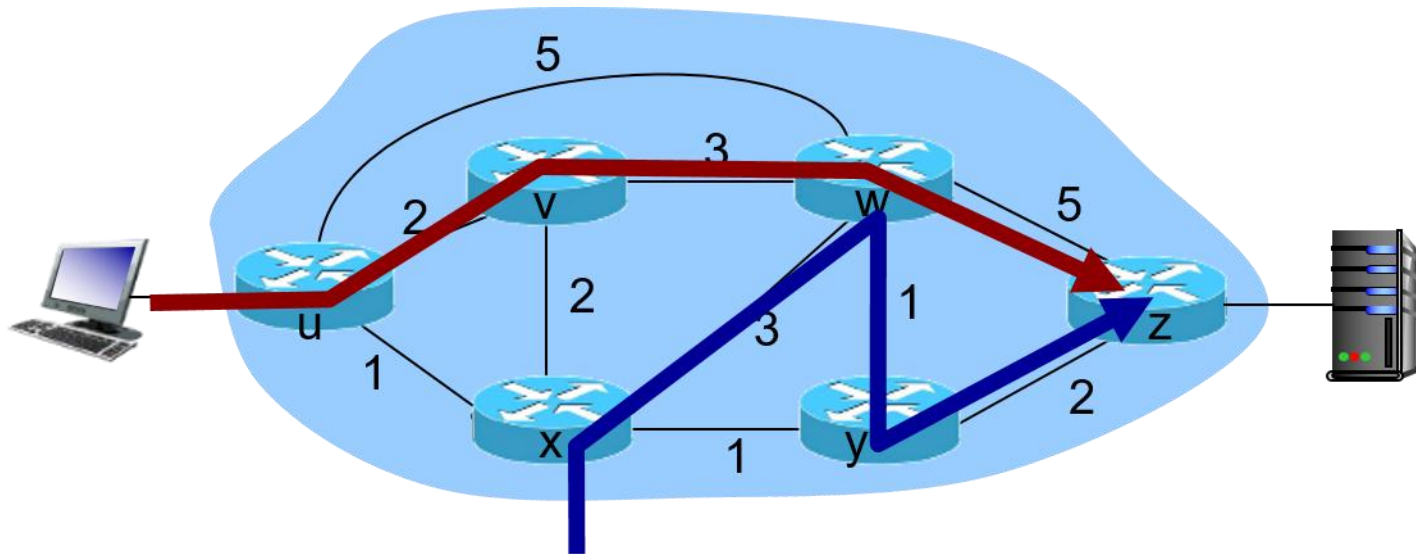**Link weights are only control "knobs": wrong!**

Pearson

Traffic Engineering: Difficult

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

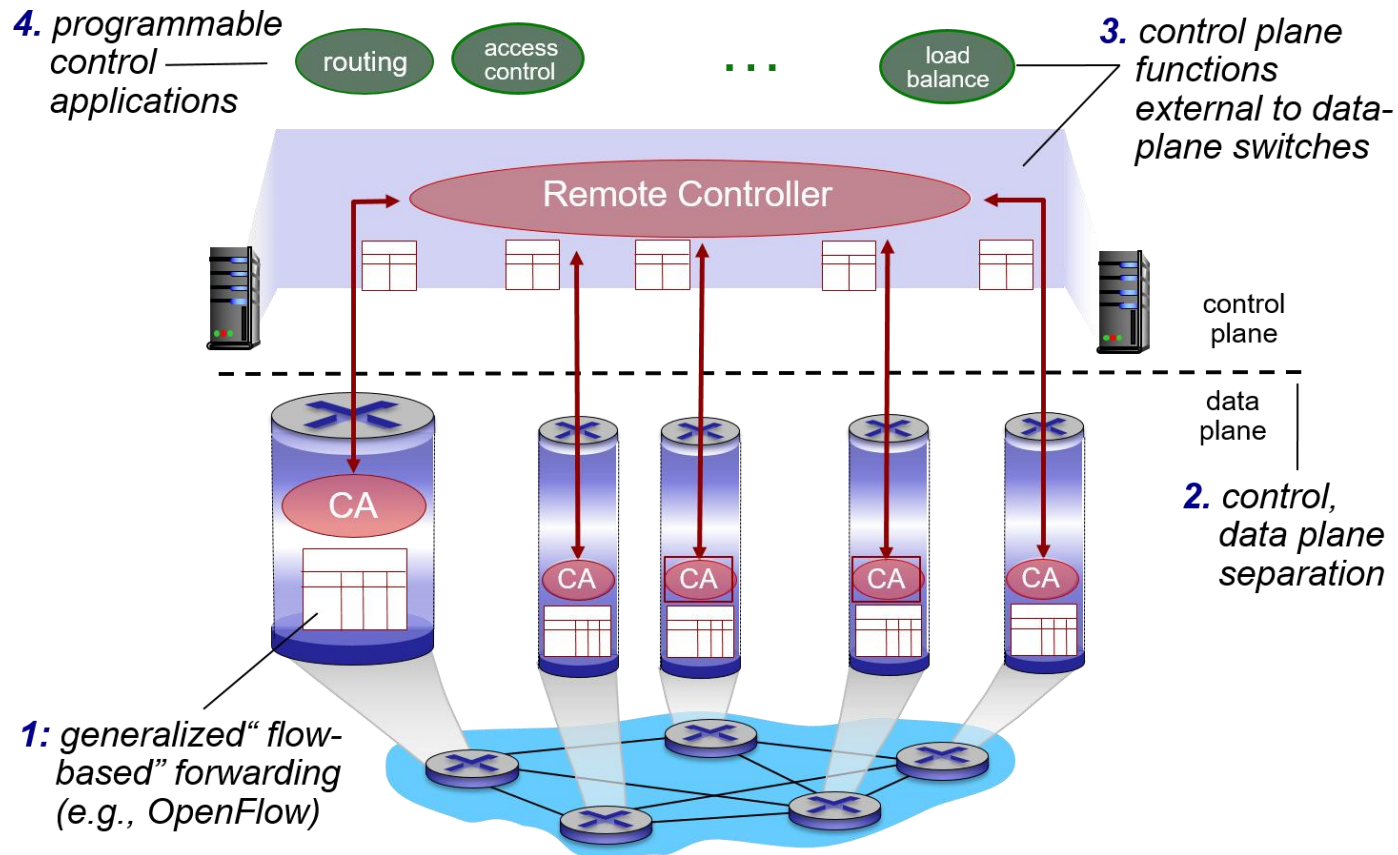**A:** can't do it (or need a new routing algorithm)

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)

Software Defined Networking (SDN) (3 of 3)

4. *programmable control applications*

routing

access control

. . .

load balance

3. *control plane functions external to data-plane switches*

Remote Controller

control plane

data plane

2. *control, data plane separation*

CA

CA

CA

CA

CA

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

SDN Perspective: Data Plane Switches
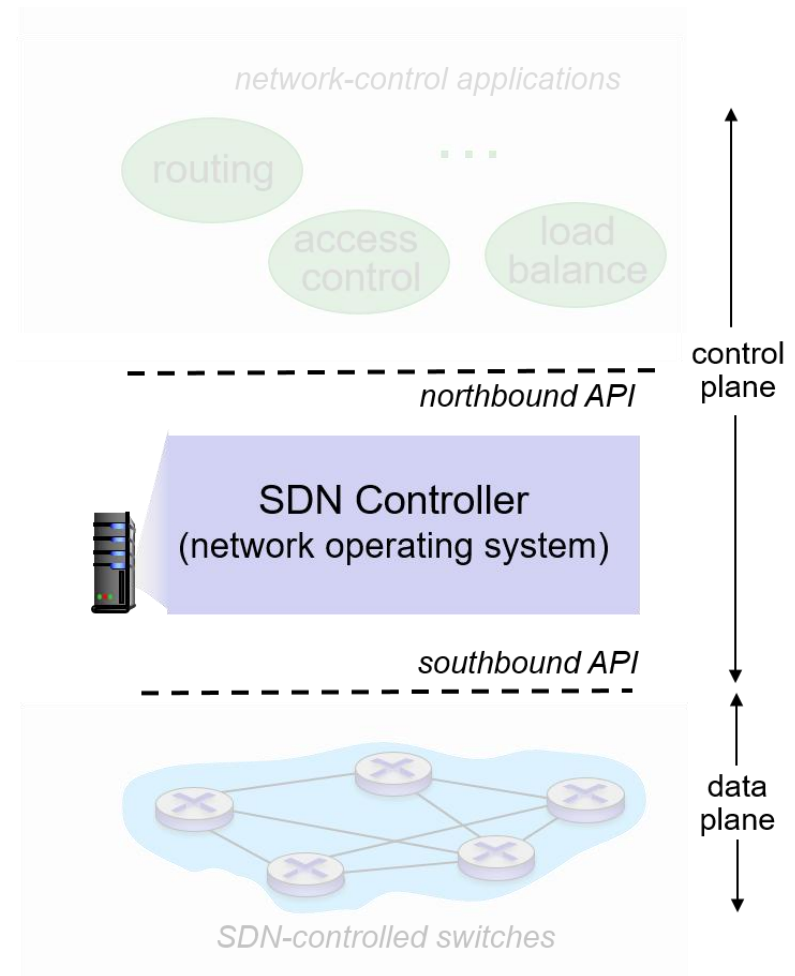
Data plane switches

- fast, simple, commodity switches implementing generalized data-plane forwarding (Section 4.4) 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)

network-control applications

routing

access control

load balance

northbound API

control plane

SDN Controller
(network operating system)

southbound API

data plane

SDN-controlled switches

PEARSON
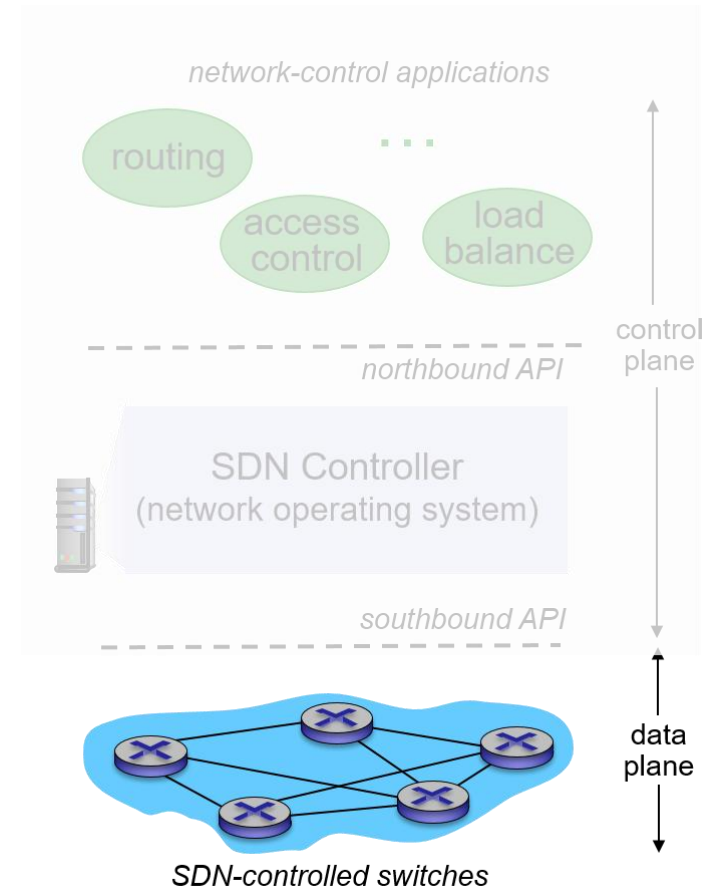
SDN Perspective: SDN Controller
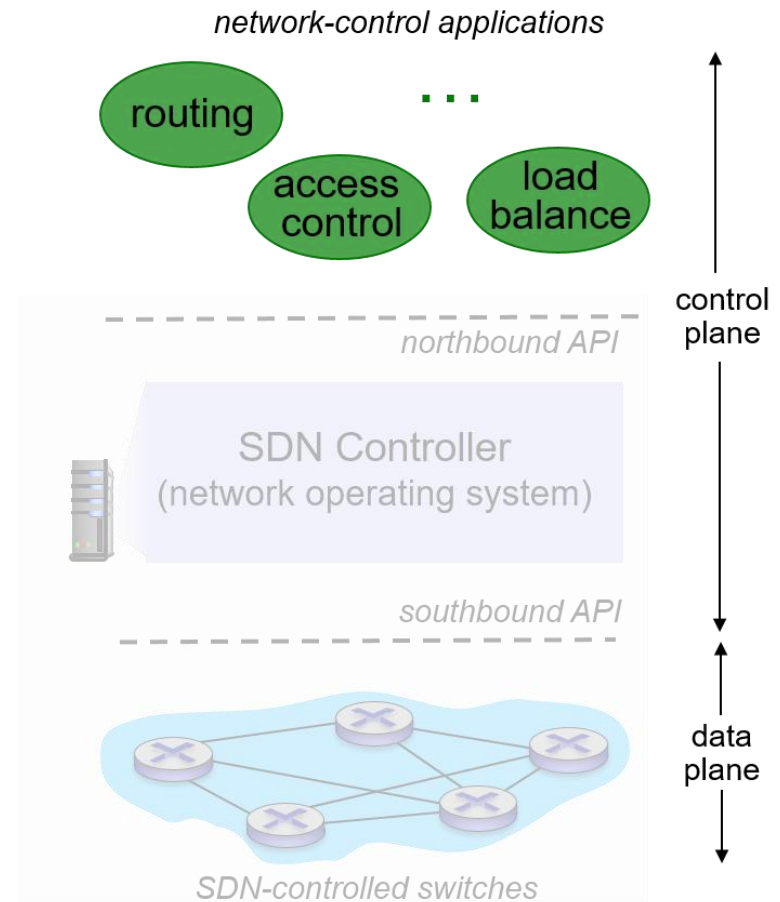
SDN controller (network OS):

- maintain network state information

- 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

network-control applications

routing

access control

load balance

northbound API

control plane

SDN Controller
(network operating system)

southbound API

data plane

SDN-controlled switches

SDN Perspective: Control Applications

network-control apps:

- "brains" of control: implement control functions using lower-level services, API provided by SND controller

- **unbundled:** can be provided by 3rd party: distinct from routing vendor, or SDN controller
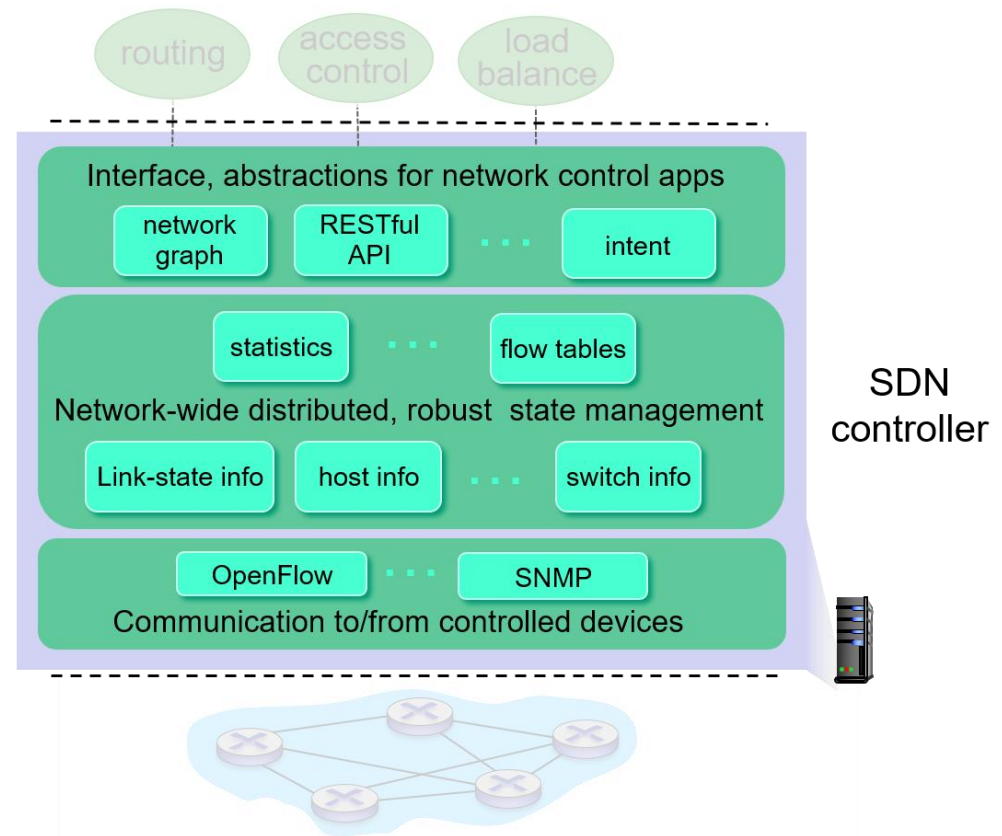

network-control applications

routing   ...

access control   load balance

control plane

northbound API

SDN Controller
(network operating system)

southbound API

data plane

SDN-controlled switches

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

routing access control load balance

Interface, abstractions for network control apps

network graph | RESTful API | ⋯ | intent

statistics ⋯ flow tables

Network-wide distributed, robust state management

Link-state info | host info | ⋯ | switch info

OpenFlow ⋯ SNMP

Communication to/from controlled devices

SDN controller

OpenFlow Protocol

- operates between controller, switch

- TCP used to exchange messages
  - optional encryption

- three classes of OpenFlow messages:
  - controller-to-switch
  - asynchronous (switch to controller)
  - symmetric (misc)

OpenFlow Controller
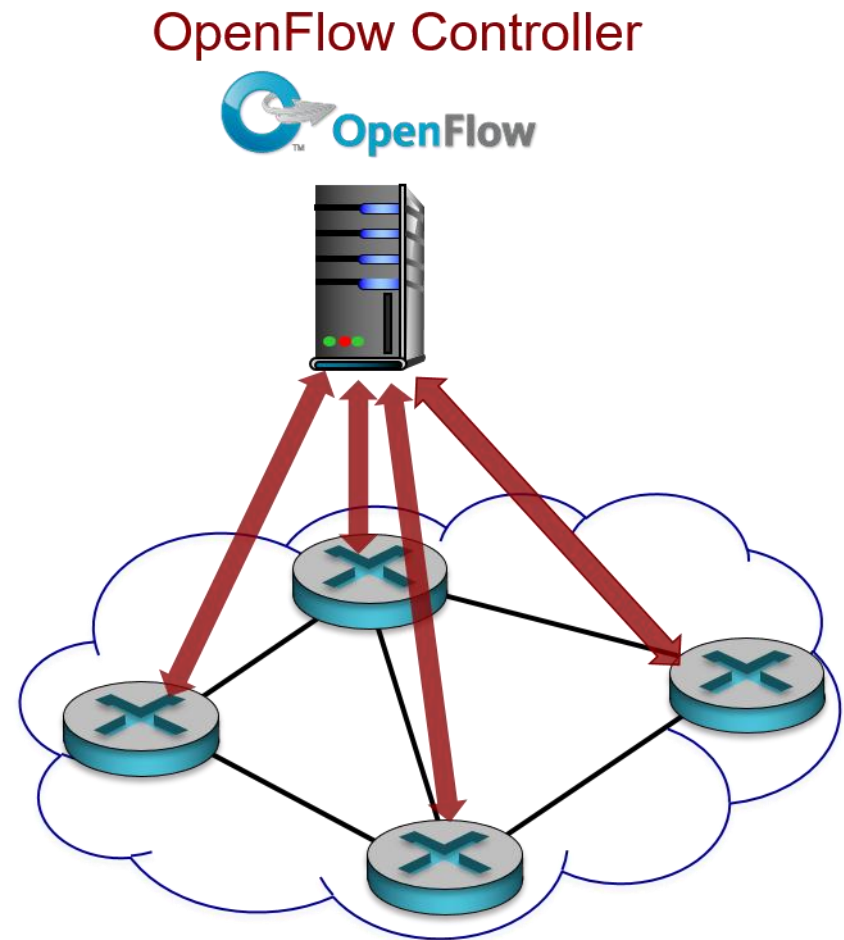
OpenFlow: Controller-to-Switch Messages

Key controller-to-switch messages

- **features:** controller queries switch features, switch replies

- **configure:** controller queries/sets switch configuration parameters

- **modify-state:** add, delete, modify flow entries in the OpenFlow tables

- **packet-out:** controller can send this packet out of specific switch port

OpenFlow Controller
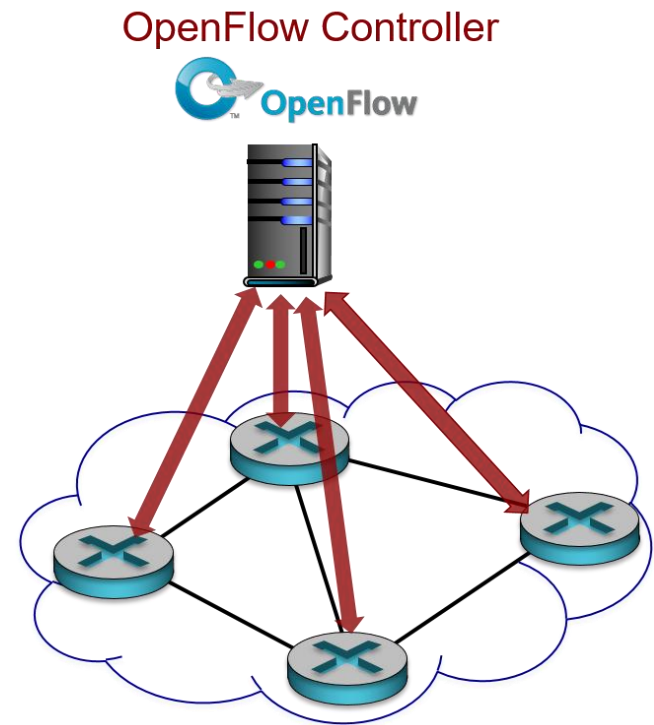
OpenFlow

OpenFlow: Switch-to-Controller Messages

Key switch-to-controller messages

- **packet-in:** transfer packet (and its control) to controller. See packet-out message from controller

- **flow-removed:** flow table entry deleted at switch

- **port status:** inform controller of a change on a port.

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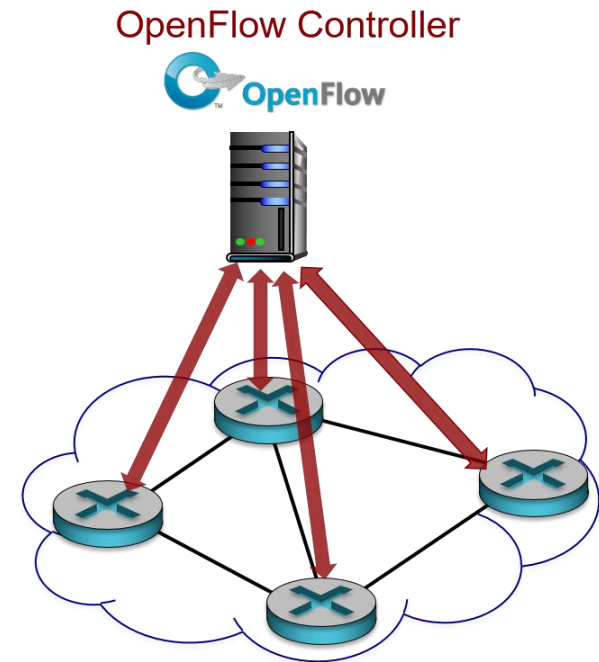
Fortunately, network operators don't "program" switches by creating/sending OpenFlow messages directly. Instead use higher-level abstraction at controller

PEARSON

# SDN: Control/Data Plane Interaction Example (1 of 2)

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

2. SDN controller receives OpenFlow message, updates link status info

3. Dijkstra's routing algorithm application has previously registered to be called when ever link status changes. It is called.

4. Dijkstra's routing algorithm access network graph info, link state info in controller, computes new routes
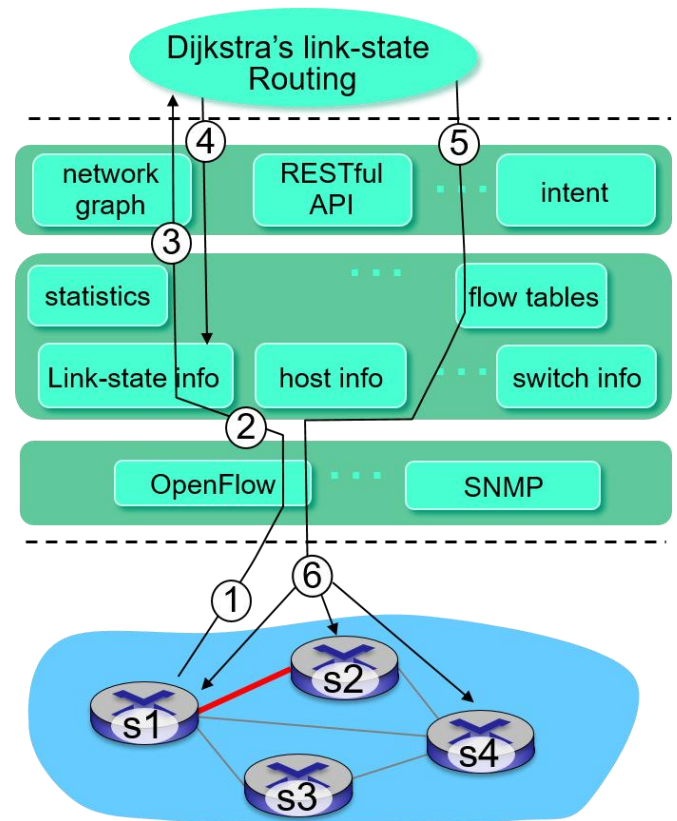
SDN: Control/Data Plane Interaction Example (2 of 2)

5. link state routing app interacts with flow-table-computation component in SDN controller, which computes new flow tables needed

6. Controller uses OpenFlow to install new tables in switches that need updating
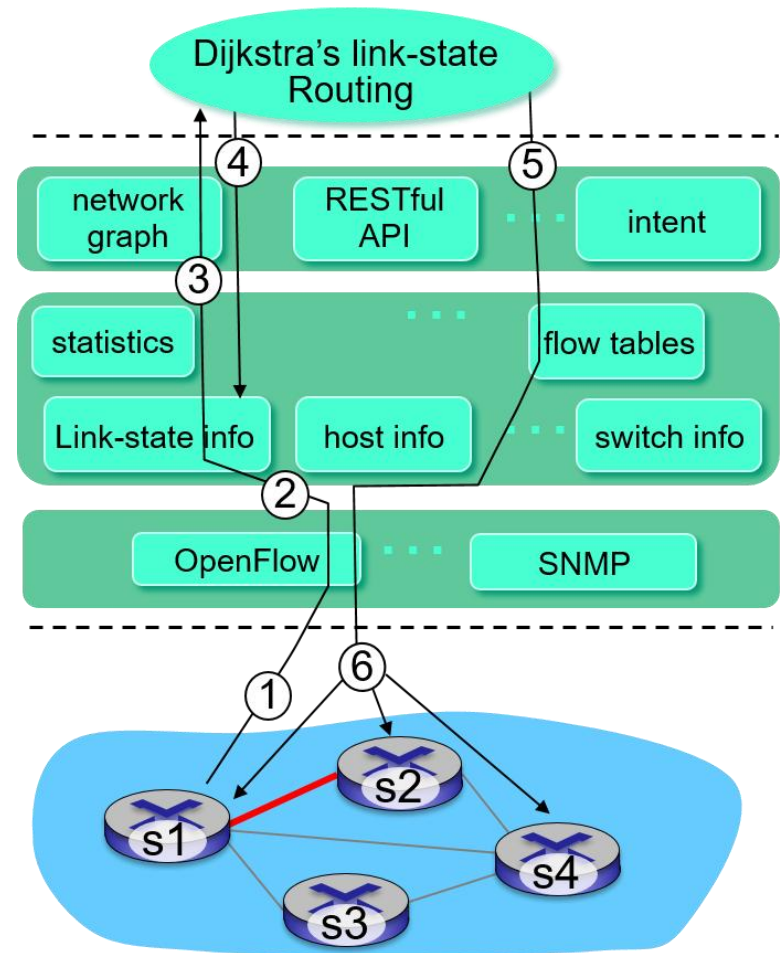
OpenDaylight (ODL) Controller

- ODL Lithium controller

- network apps may be contained within, or be external to SDN controller

- Service Abstraction Layer: interconnects internal, external applications and services
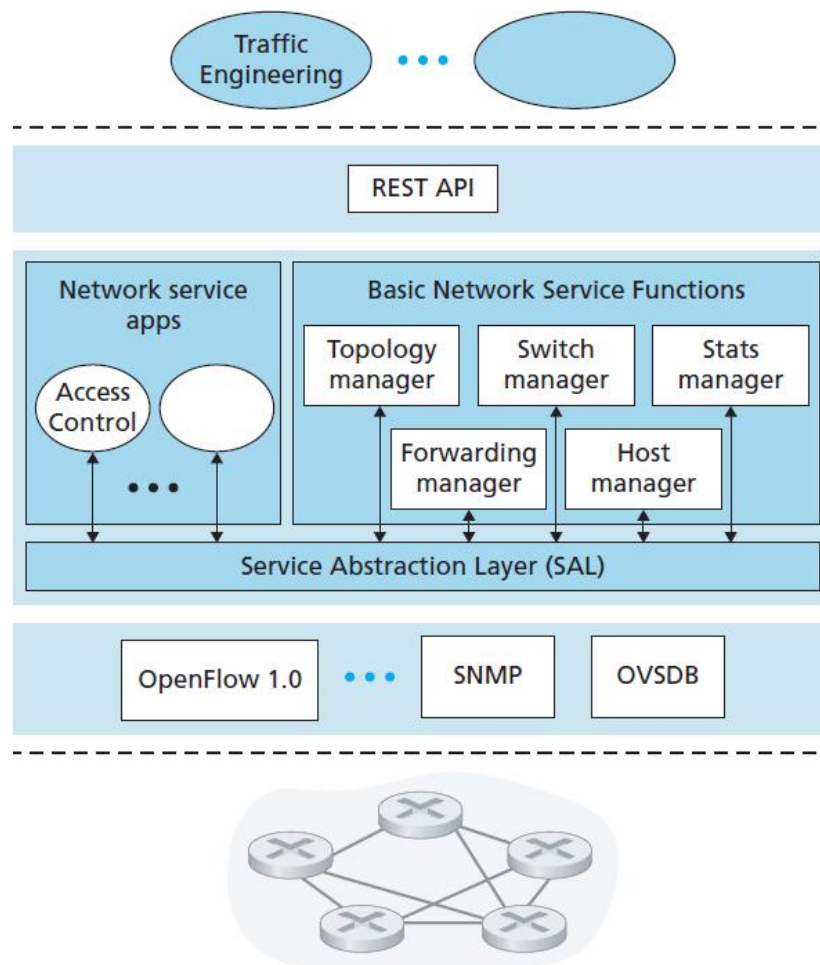
ONOS Controller

- control apps separate from controller

- intent framework: high-level specification of service: what rather than how

- considerable emphasis on distributed core: service reliability, replication performance scaling
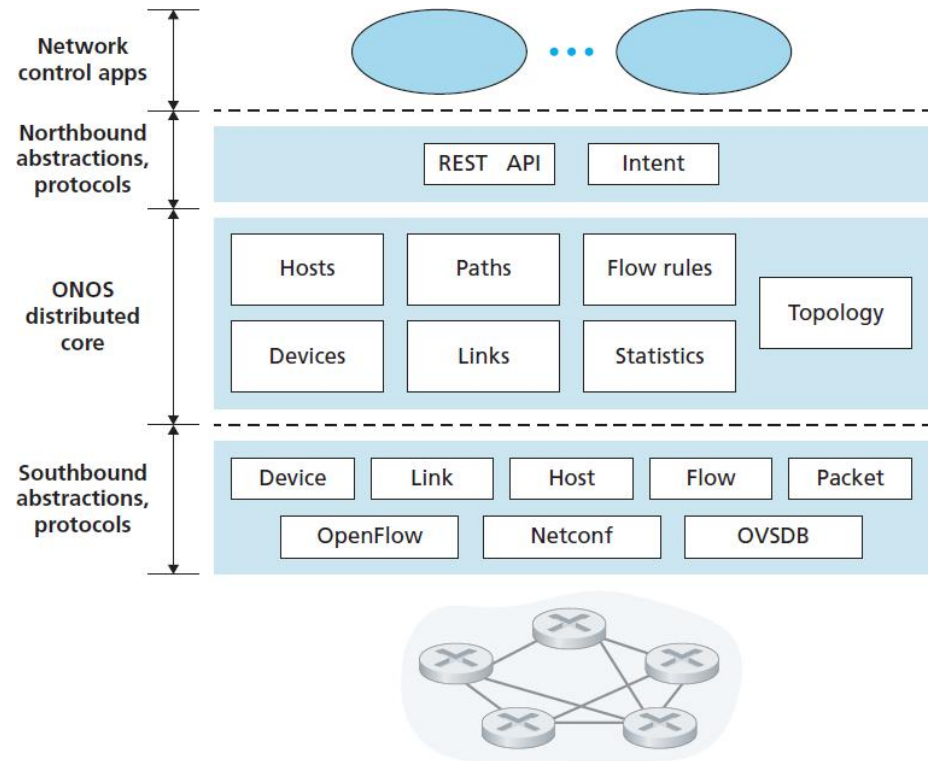
SDN: Selected Challenges

- 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?

- networks, protocols meeting mission-specific requirements
  - e.g., real-time, ultra-reliable, ultra-secure

- Internet-scaling

Learning Objectives

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 (1 of 2)

- 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

ICMP: Internet Control Message Protocol (2 of 2)

| ICMP Type	Code	Description
0	0	echo reply (to ping)
3	0	destination network unreachable
3	1	destination host unreachable
3	2	destination protocol unreachable
3	3	destination port unreachable
3	6	destination network unknown
3	7	destination host unknown
4	0	source quench (congestion control)
8	0	echo request
9	0	router advertisement
10	0	router discovery
11	0	TTL expired
12	0	IP header bad

Traceroute and ICMP (1 of 2)

- source sends series of UDP segments to destination
  - first set has TTL =1
  - second set has TTL = 2, etc.
  - unlikely port number

- when datagram in *n*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

Pearson

Copyright © 2017, 2013, 2010 Pearson Education, Inc. All Rights Reserved

Traceroute and ICMP (2 of 2)

stopping criteria:

- UDP segment eventually arrives at destination host

- destination returns ICMP "port unreachable" message (type 3, code 3)

- source stops
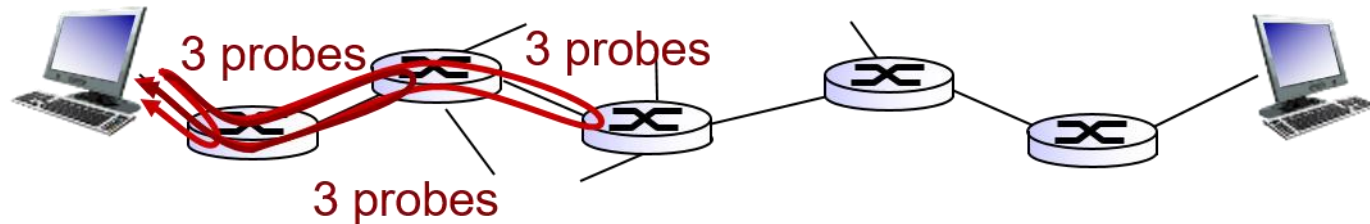
Pearson Copyright © 2017, 2013, 2010 Pearson Education, Inc. All Rights Reserved

Learning Objectives

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."
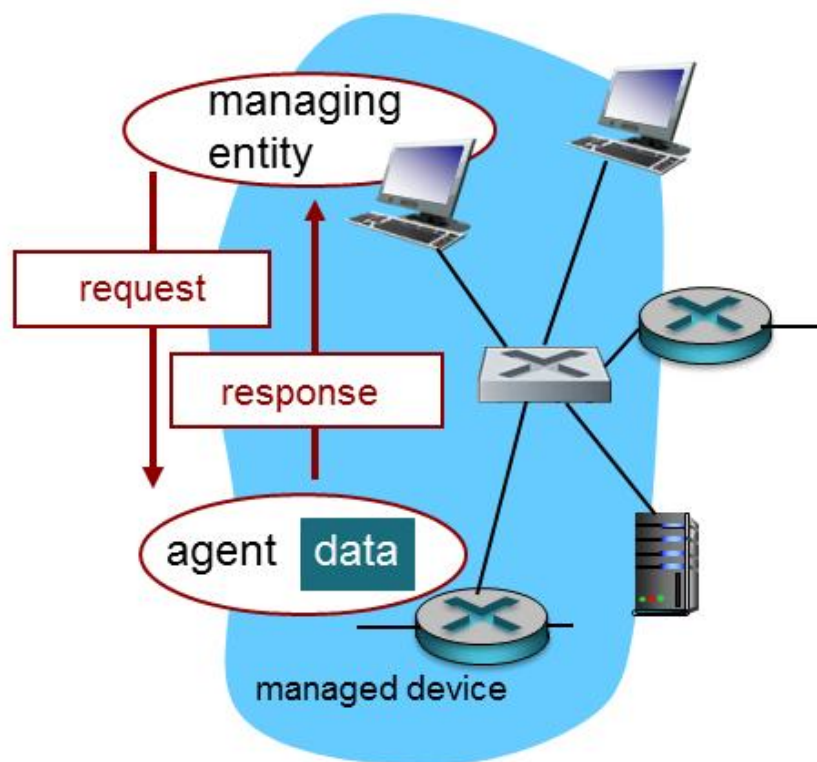
Infrastructure for Network Management

definitions:

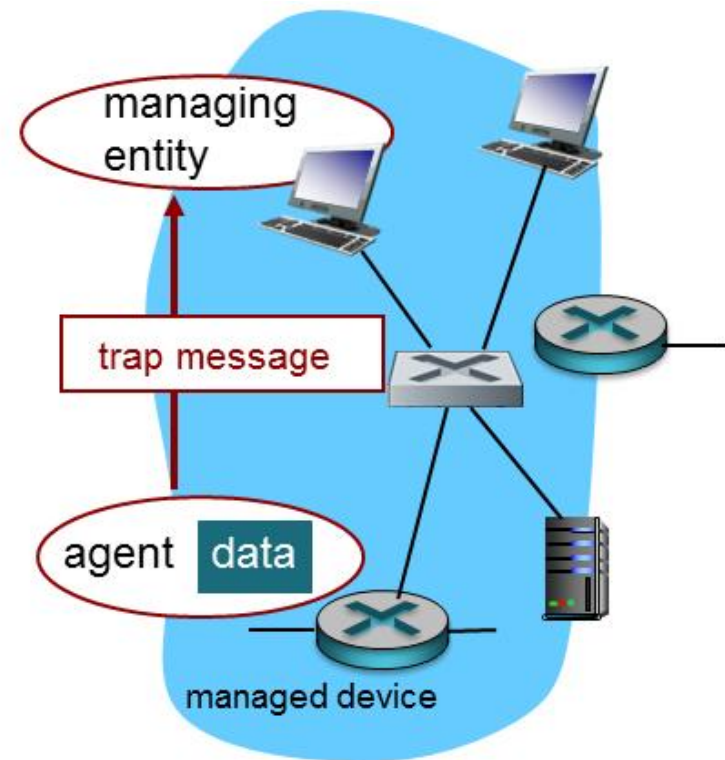**managed devices** contain **managed objects** whose data is gathered into a **Management Information Base (MIB)**
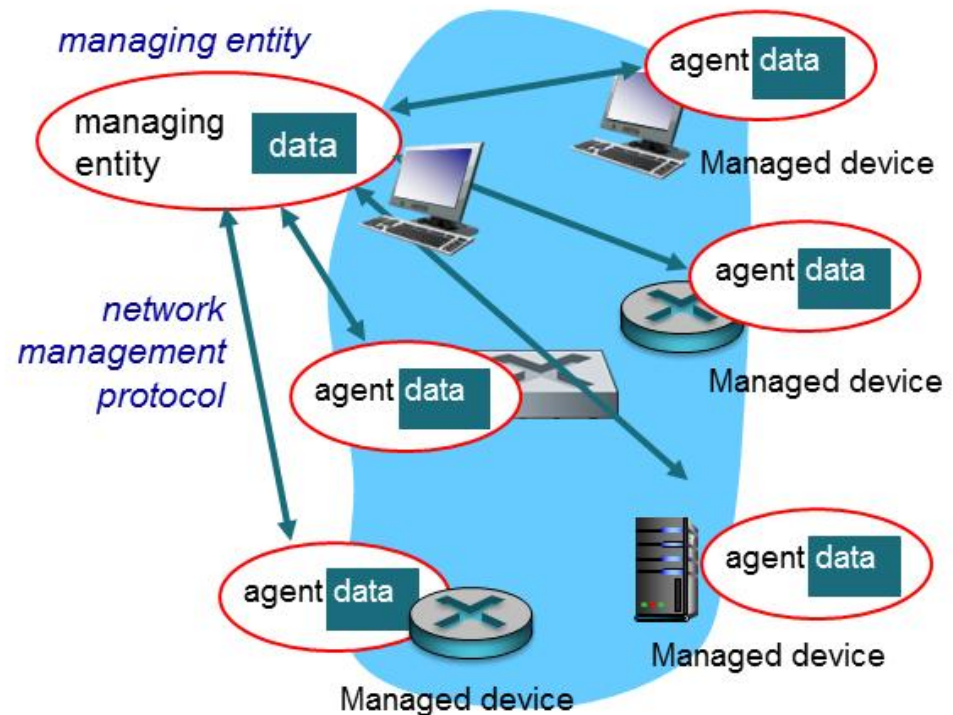
SNMP Protocol

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
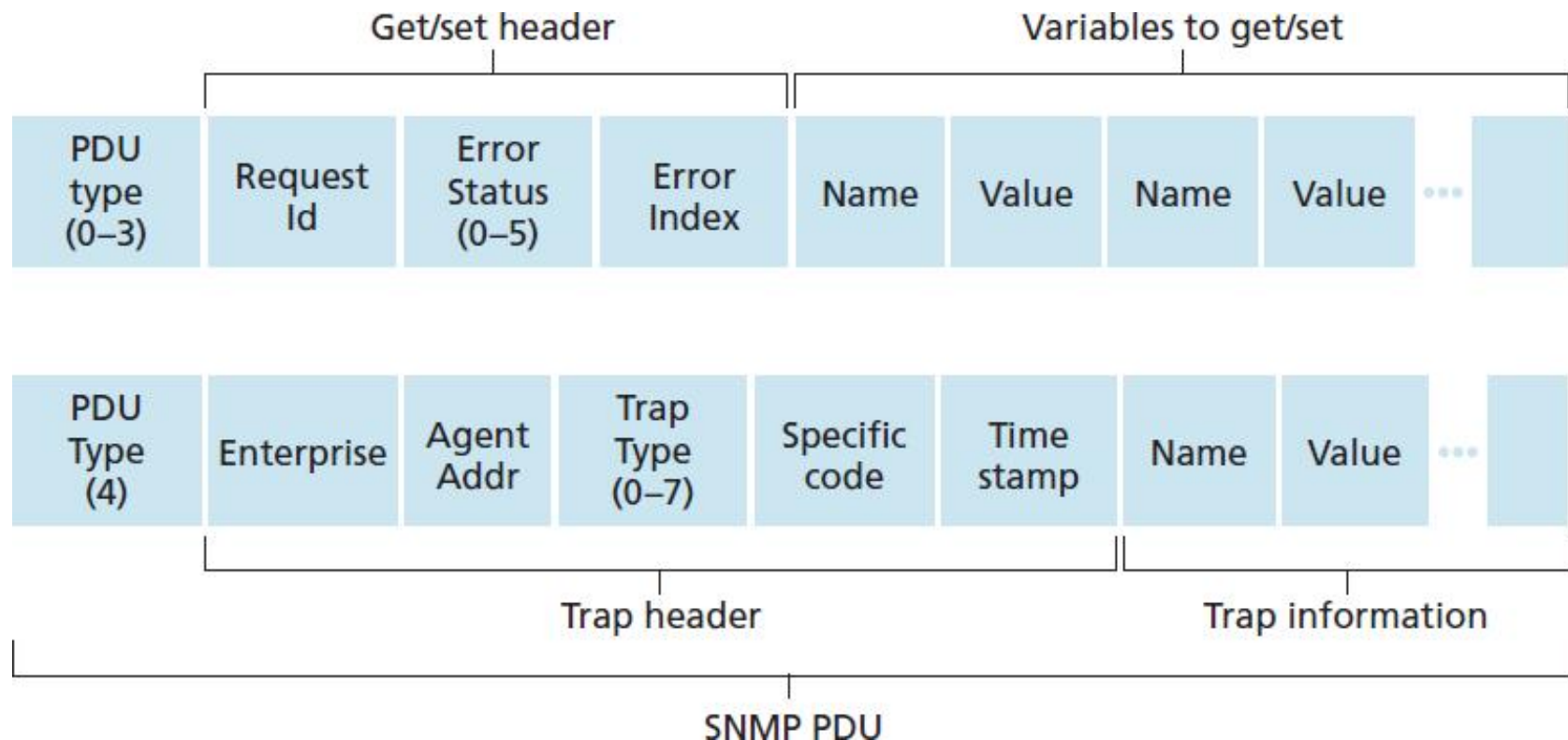
Chapter 5: Summary

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!

Copyright

This work is protected by United States copyright laws and is provided solely for the use of instructors in teaching their courses and assessing student learning. Dissemination or sale of any part of this work (including on the World Wide Web) will destroy the integrity of the work and is not permitted. The work and materials from it should never be made available to students except by instructors using the accompanying text in their classes. All recipients of this work are expected to abide by these restrictions and to honor the intended pedagogical purposes and the needs of other instructors who rely on these materials.