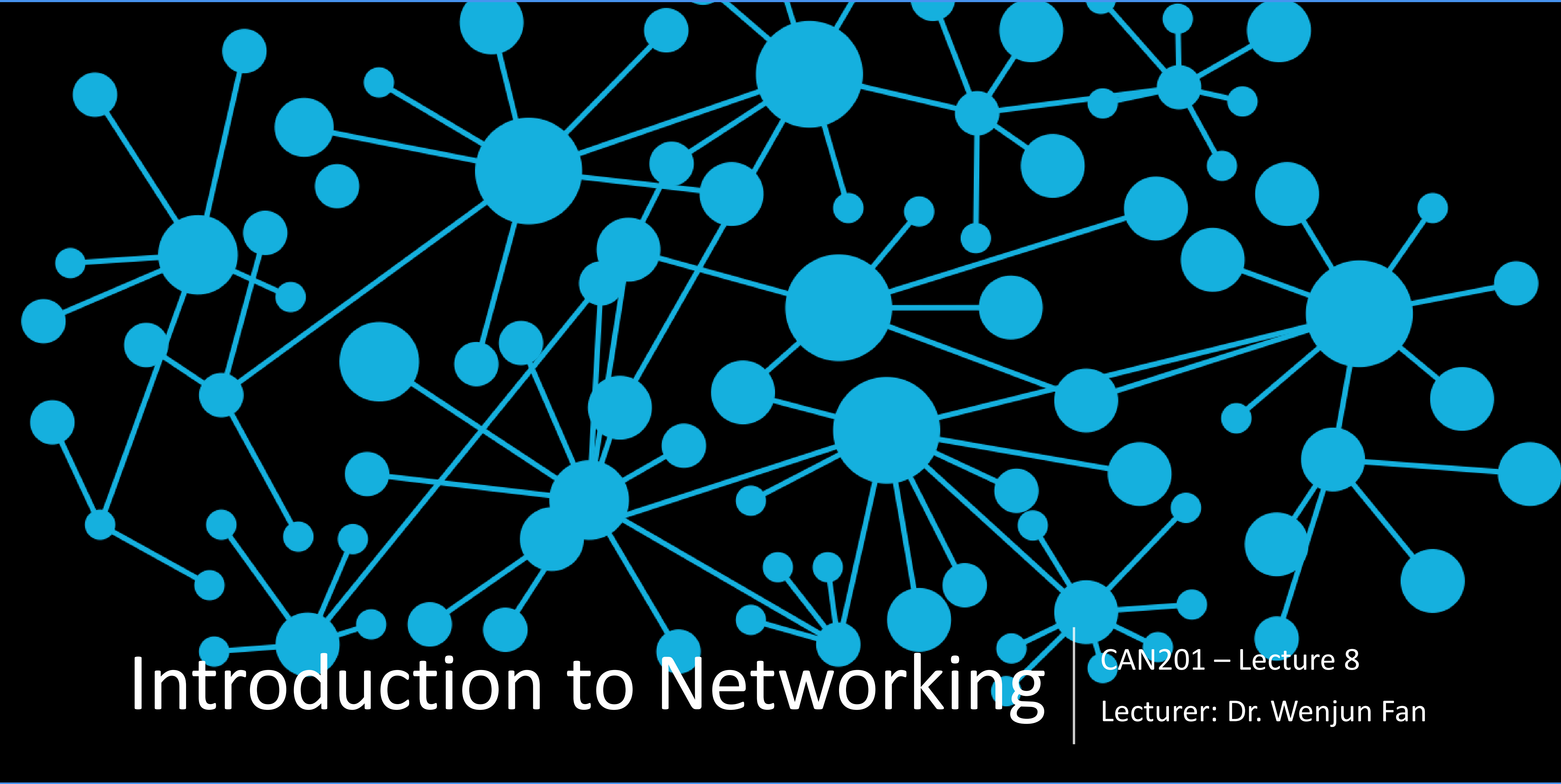


Introduction to Networking

CAN201 – Lecture 8

Lecturer: Dr. Wenjun Fan

Lecture 8 – Network Layer Control Plane (2)

- **Roadmap**
  1. Routing (2) - Distance vector algorithm
  2. Intra-AS routing in the Internet: OSPF
  3. Routing among the ISPs: BGP
  4. The SDN control plane
  5. ICMP
  6. SNMP

Distance vector algorithm

- Bellman-Ford equation (dynamic programming)

let

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

then

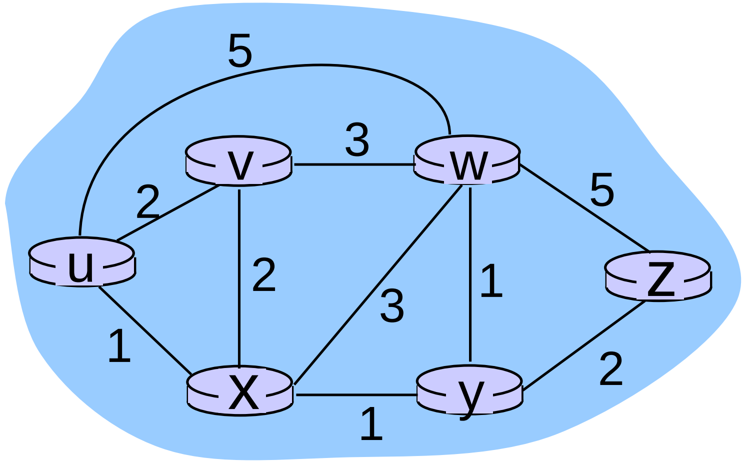
$$d_x(y) = \min_v\{\, c(x,v) + d_v(y) \,\}$$

cost from neighbor v to destination y

cost to neighbor v

min_v is taken over all neighbors v of x

Bellman-Ford example



B-F equation:

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

From u's perspective estimated:

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

Node achieving minimum is the 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 estimate of 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
      $\mathbf{D_v} = [D_v(y): y \in N]$

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

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:**

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

- Under minor, natural conditions, the estimate $D_x(y)$ *converge to the actual least cost* $d_x(y)$

Distance vector algorithm

Iterative, asynchronous:

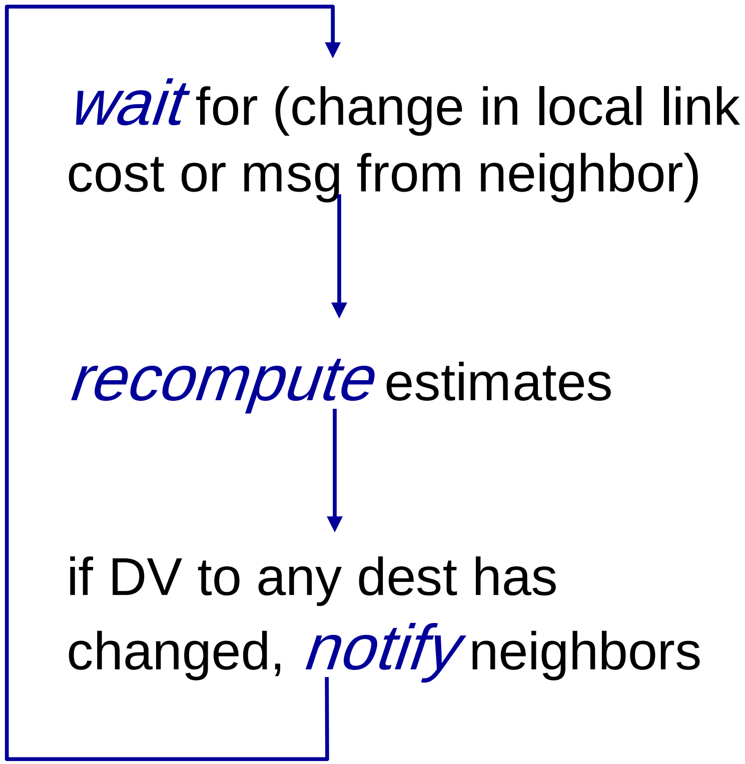
each local iteration caused by:

- **Local link cost change**
- **DV update message from neighbor**

Distributed:

- **Each node notifies neighbors *only* when its DV changes**
  - Its neighbors then further notify their neighbors if necessary

Each node:



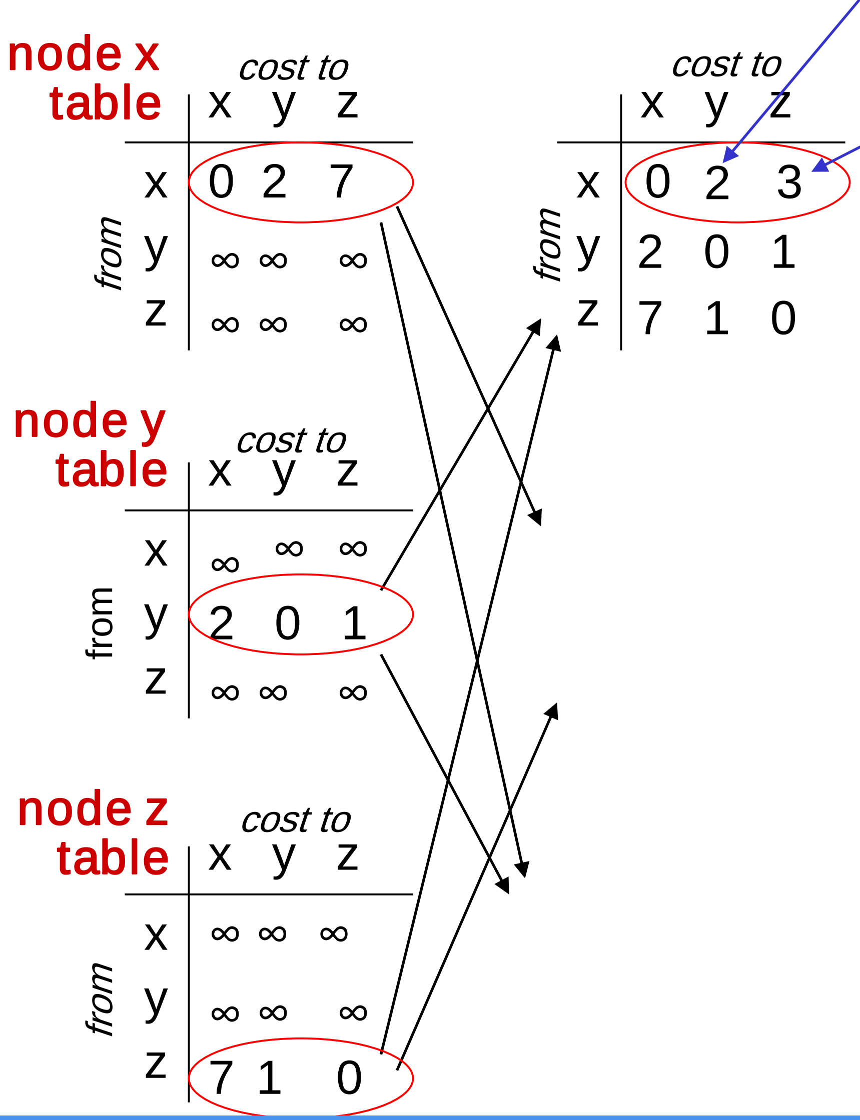
wait for (change in local link cost or msg from neighbor)

recompute estimates

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

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

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



node x table

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

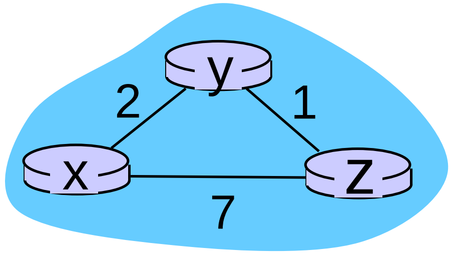
		cost to		
		x	y	z
from	x	0	2	3
	y	2	0	1
	z	7	1	0

node y table

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

node z table

		cost to		
		x	y	z
from	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

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

cost to

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

node y table

cost to

from	x	y	z
x	∞	∞	∞
y	2	0	1
z	∞	∞	∞

cost to

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

cost to

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

node z table

cost to

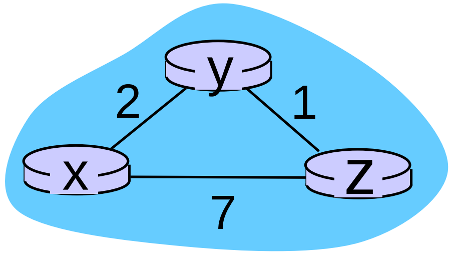
from	x	y	z
x	∞	∞	∞
y	∞	∞	∞
z	7	1	0

cost to

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

cost to

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

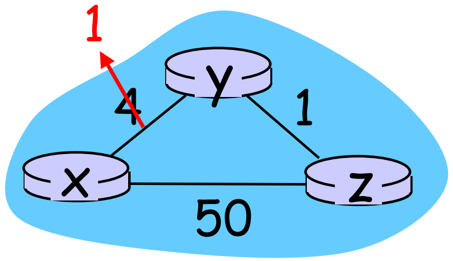


time

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.

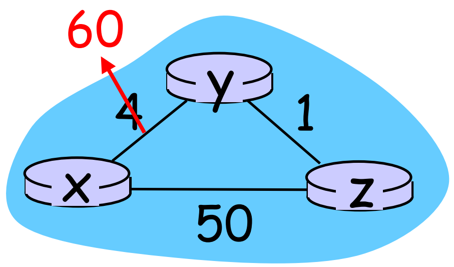
t_1 : *z* receives update from *y*, updates its table, computes new least cost to *x* , sends its neighbors its DV.

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*.

Distance vector: link cost changes

Link cost changes:

- Node detects local link cost change
- *Bad news travels slow* – "count to infinity" problem!
- How many iterations before algorithm stabilizes?
    - 44 iterations before algorithm stabilizes



Poisoned reverse: 反向中毒

- 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?

Lecture 8 – Network Layer Control Plane (2)

- **Roadmap**
  1. Routing (2) - Distance vector algorithm
  2. Intra-AS routing in the Internet: OSPF
  3. Routing among the ISPs: BGP
  4. The SDN control plane
  5. ICMP
  6. SNMP

Making routing scalable 使路由可拓展

Our routing study thus far - idealized

- All routers identical
- Network "flat"

… *not* true in practice because of the two reasons below

Scale: **with billions of destinations:**

- Can't store all destinations in routing tables!
- Routing table exchange would swamp links!

Administrative autonomy 行政自主

- Internet = network of networks/ISPs
- Each network/ISP admin may want to control routing in its own network

Internet approach to scalable routing

Aggregate routers into regions known as "autonomous 自制系统 systems" (AS) (a.k.a. "domains")

内部

Intra-AS routing

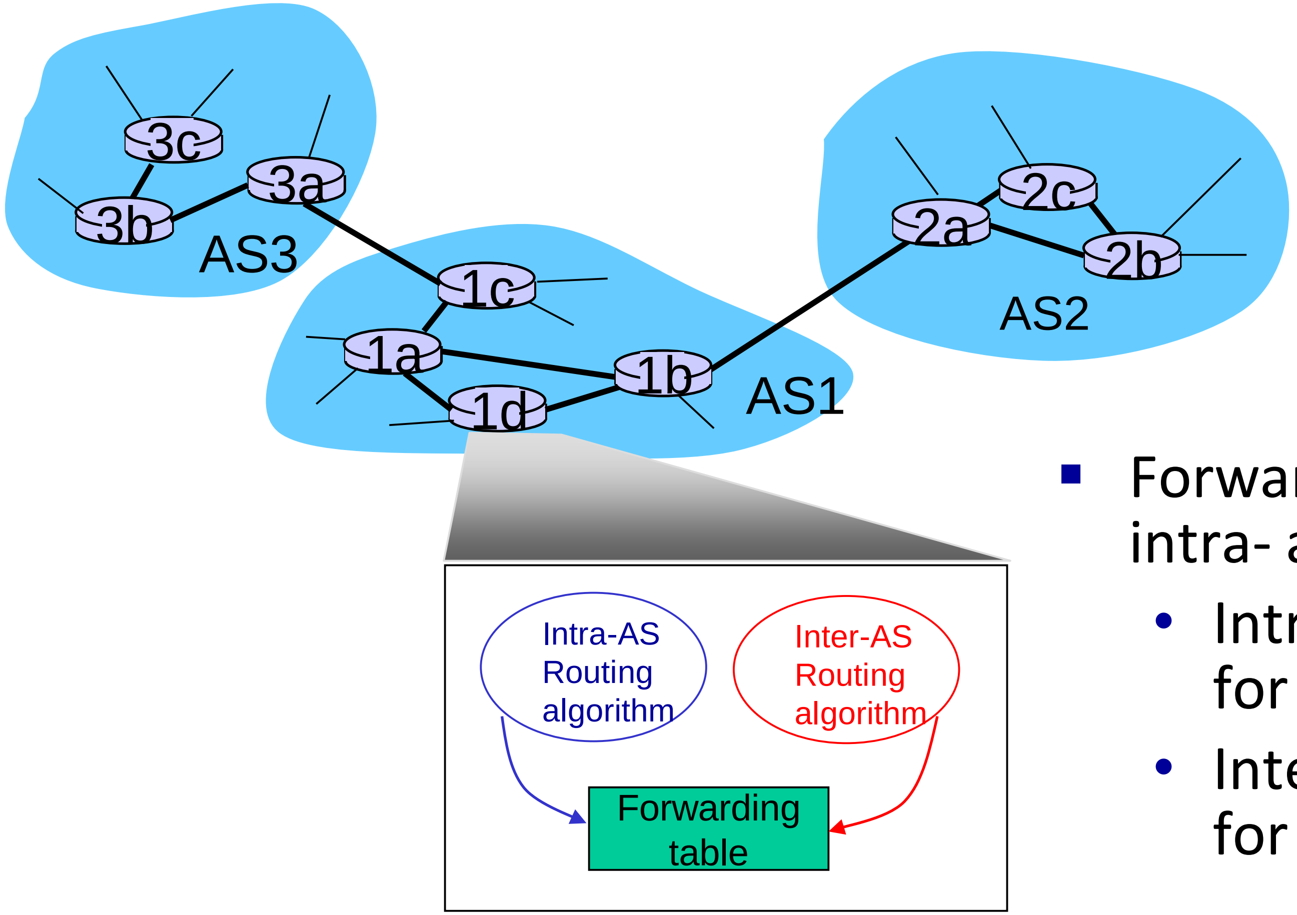
- Routing among hosts, routers in same AS ("network")
- All routers in AS must run *same* intra-domain protocol
- Routers in *different* AS can run *different* intra-domain routing protocol
- Gateway router: at "edge" of its own AS, has link(s) to router(s) in other AS'es

互相

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

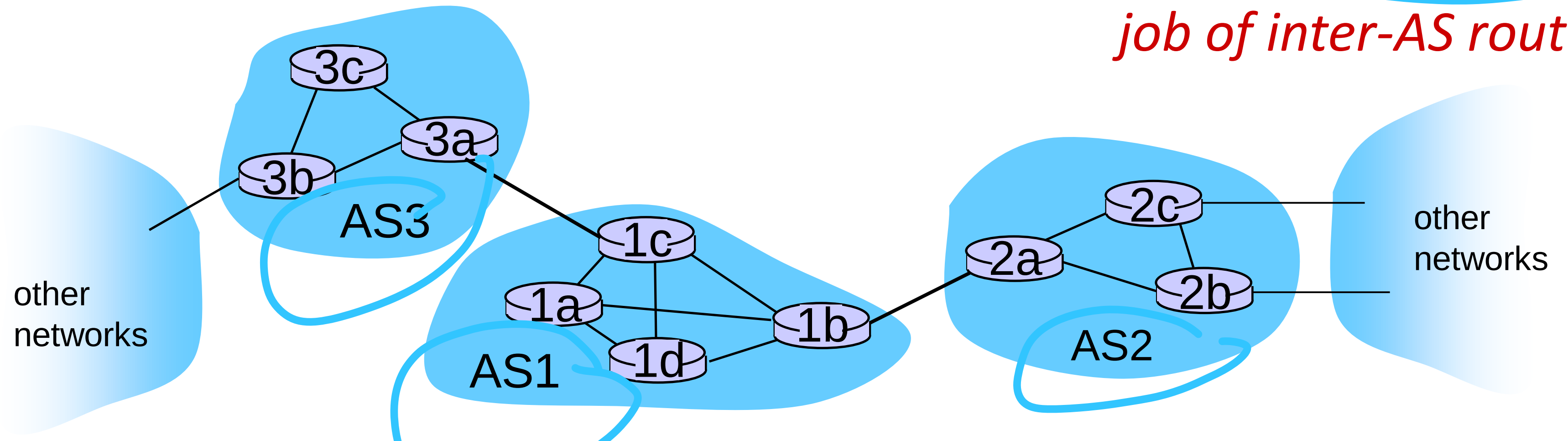
Inter-AS tasks

- **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)

*ISIS = Intermediate System-to-Intermediate System

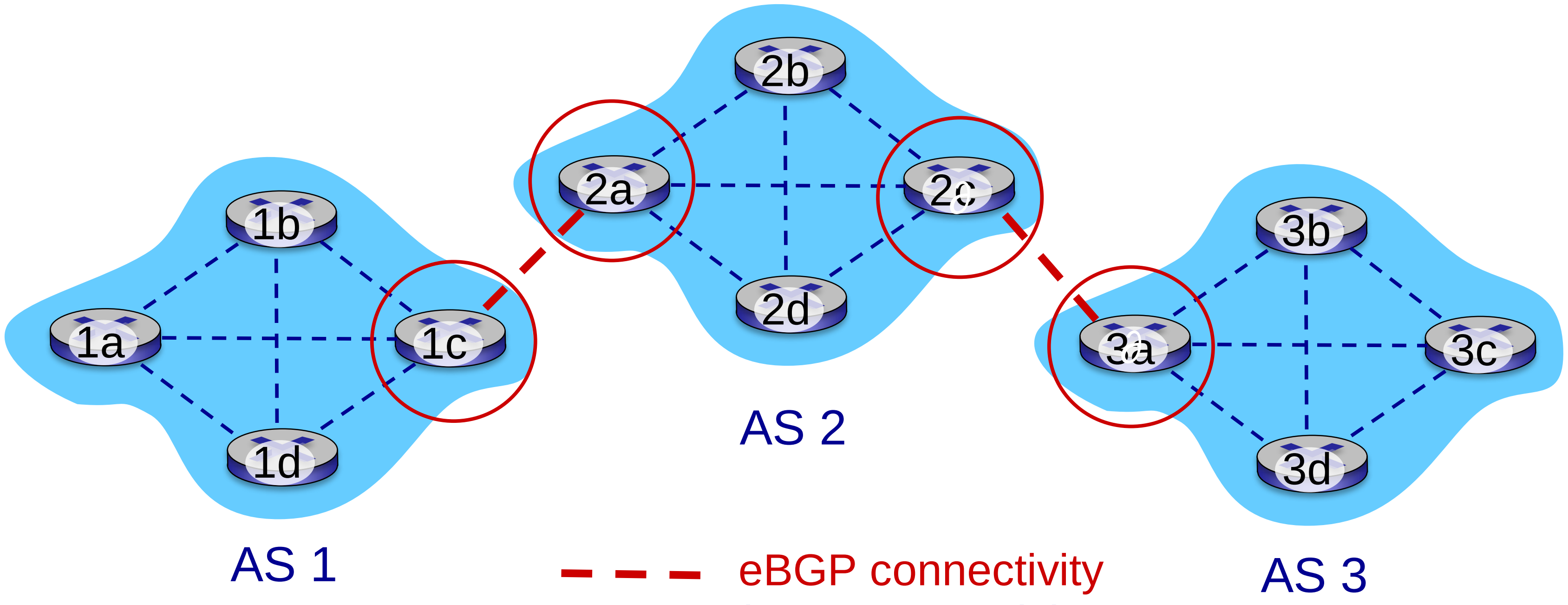
Lecture 8 – Network Layer Control Plane (2)

- **Roadmap**
  1. Routing (2) - Distance vector algorithm
  2. Intra-AS routing in the Internet: OSPF
  3. Routing among the ISPs: BGP
  4. The SDN control plane
  5. ICMP
  6. SNMP

Internet inter-AS routing: BGP

- **BGP (Border Gateway Protocol): the *de facto* inter-domain routing protocol** 边界网关协议
  - “glue that holds the Internet together”
- **BGP provides each AS router a means to:**
  - 外部 eBGP: obtain subnet prefix reachability information from neighboring ASes. Allows subnet to advertise its existence to rest of Internet: *“I am here.”*
  - 内部 iBGP: propagate reachability information to all AS-internal routers, and determine “best” routes to other networks based on reachability information and *policy*

eBGP, iBGP connections



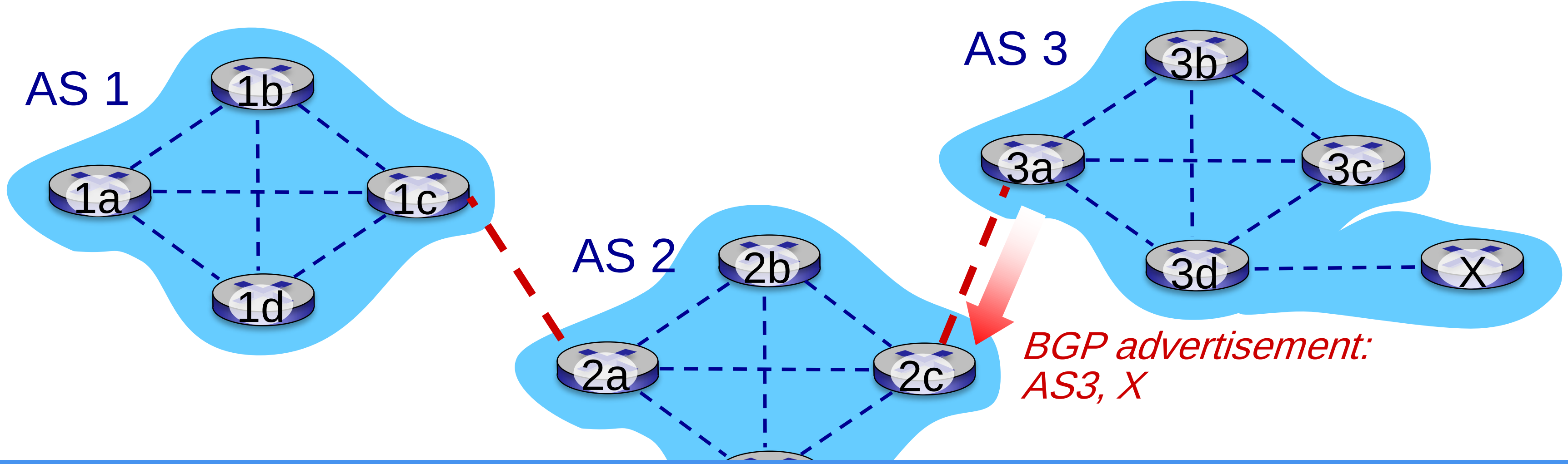
▬ ▬ ▬ eBGP connectivity

- - - - - - iBGP connectivity

gateway routers run both eBGP and iBGP protocols

BGP basics

- **BGP session: two BGP routers ("peers") exchange BGP messages over semi-permanent TCP connection (using port 179):**
  - 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

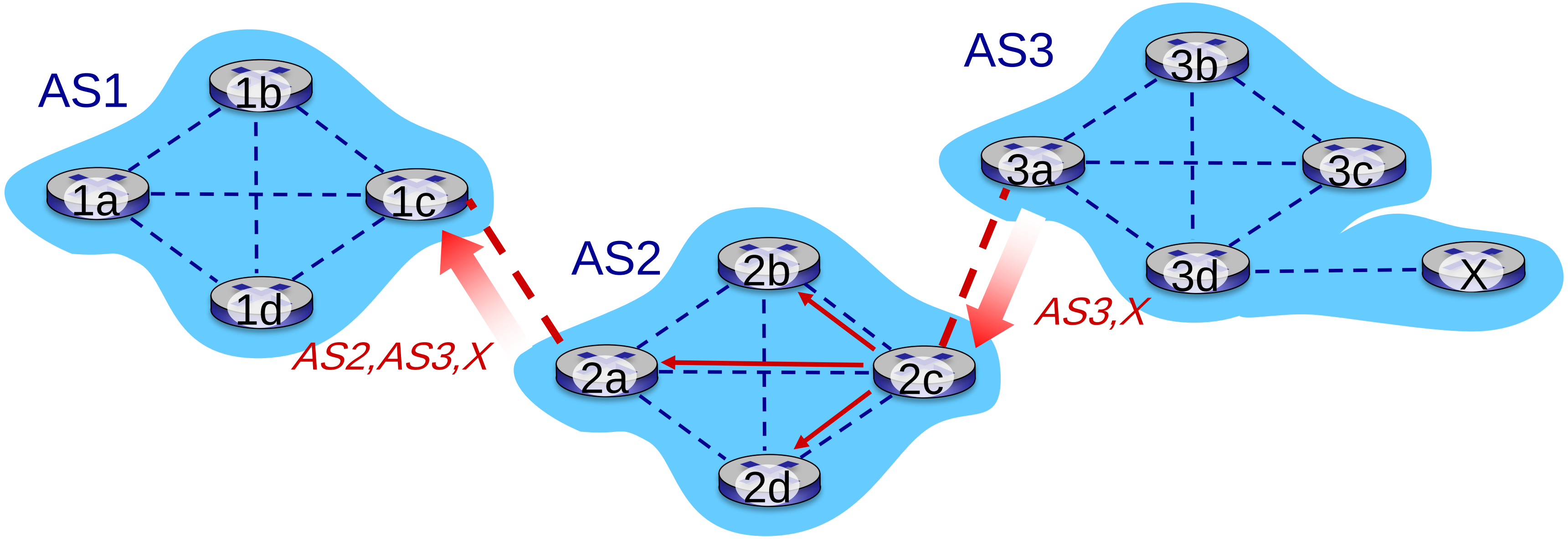


BGP advertisement:
AS3, X

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 the IP address of the router interface that begins the AS-PATH.
- ***Policy-based routing:***
  - Gateway receiving route advertisement uses *import policy* to accept/decline path (e.g., never route through AS Y).
  - AS policy also determines whether to *advertise* path to other neighboring ASes.

BGP path advertisement

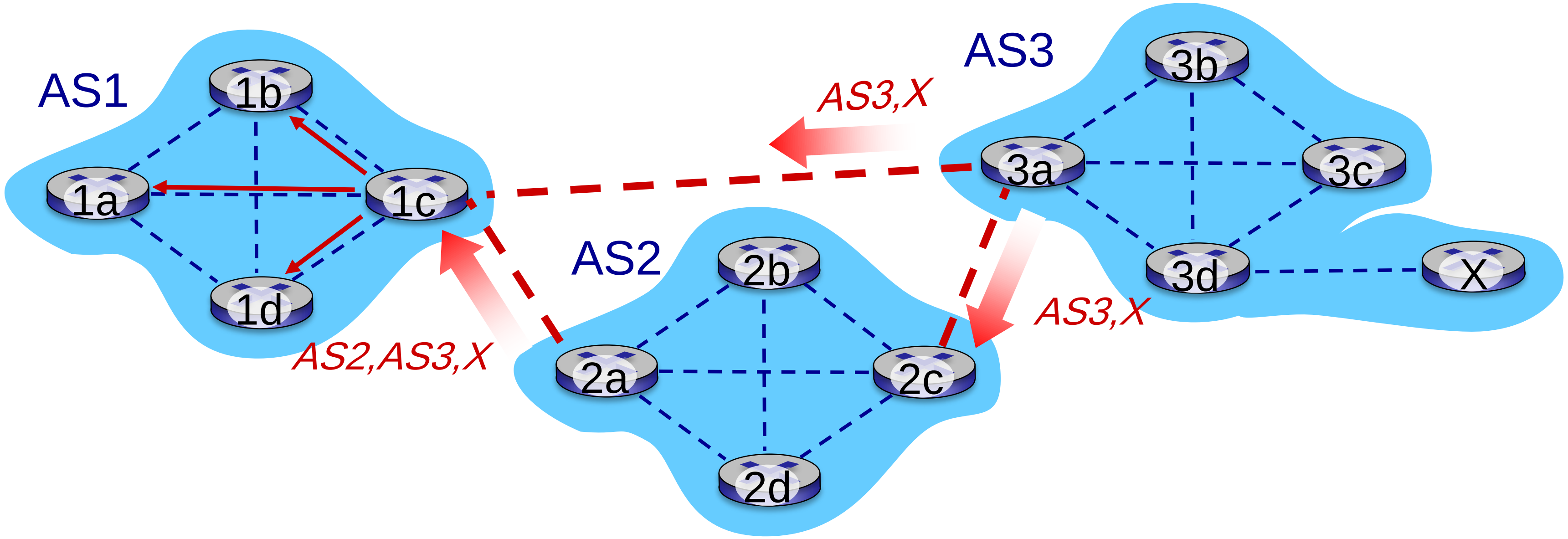


- AS2 router 2c receives path advertisement AS3,X (via eBGP) from AS3 router 3a

- Based on AS2 policy, AS2 router 2c accepts path AS3,X, propagates (via iBGP) to all AS2 routers

- Based on AS2 policy, AS2 router 2a advertises (via eBGP) path AS2, AS3, X to AS1 router 1c

BGP path advertisement



Gateway router may learn about multiple paths to destination:

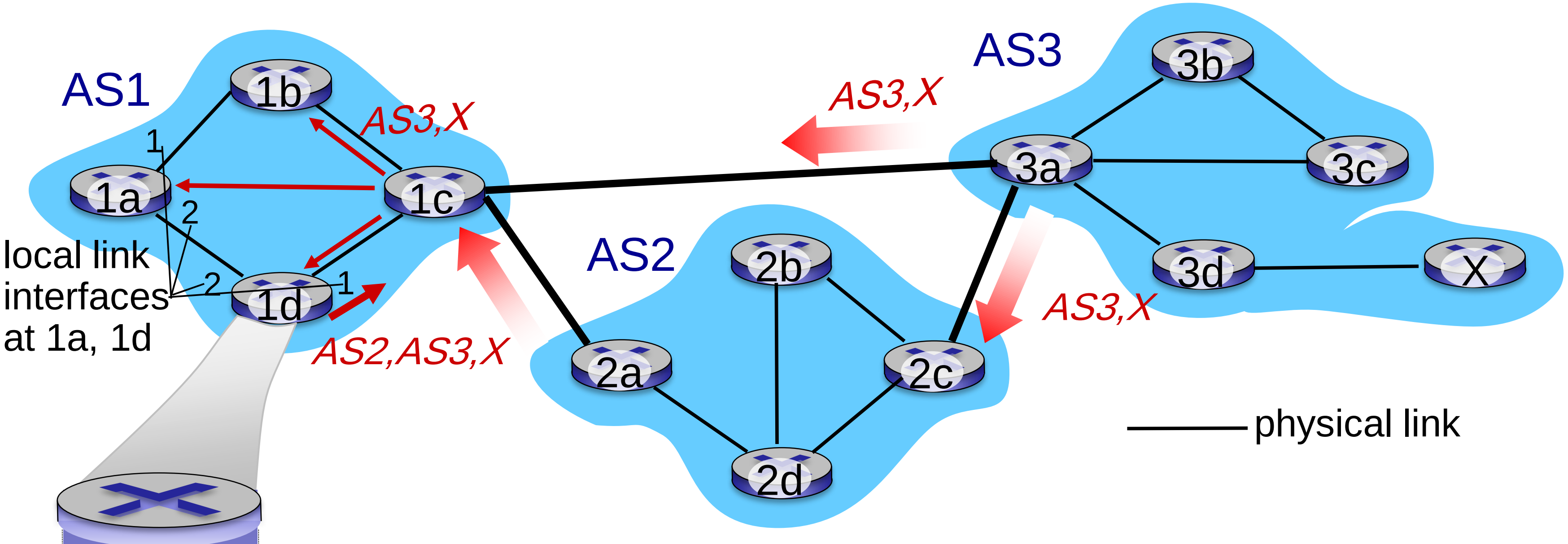
- AS1 gateway router 1c learns path *AS2,AS3,X* from 2a
- AS1 gateway router 1c learns path *AS3,X* from 3a
- Based on policy, AS1 gateway router 1c chooses path *AS3,X, and advertises path within AS1 via iBGP*

BGP 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, OSPF, forwarding table entries

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

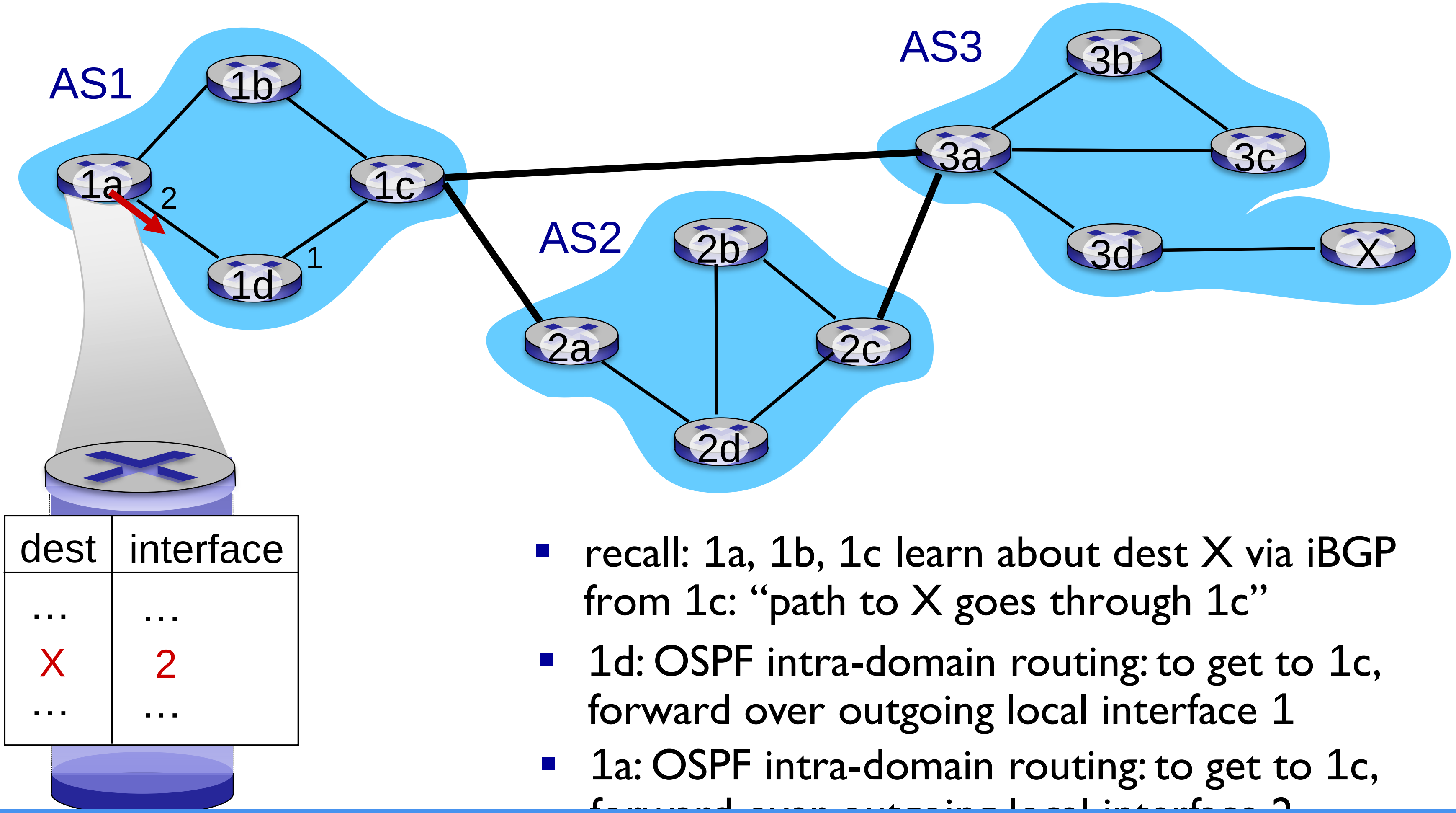


dest	interface
…	…
X	1
…	…

- 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

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



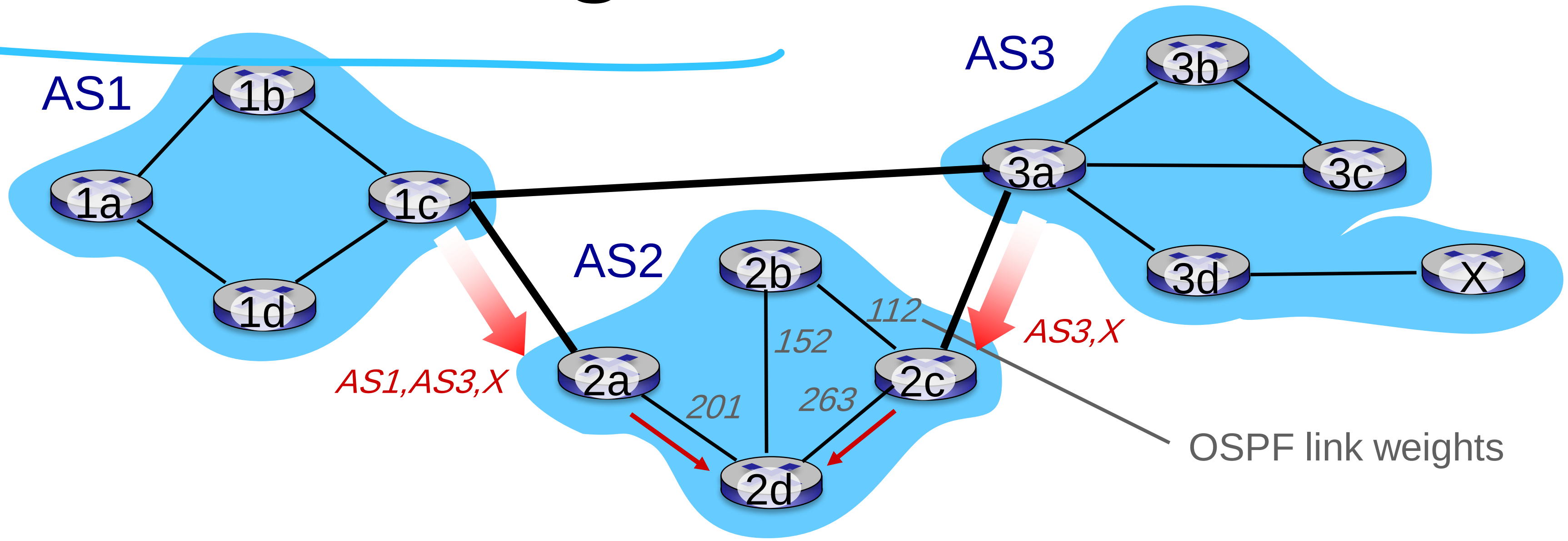
dest	interface
…	…
X	2
…	…

- 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

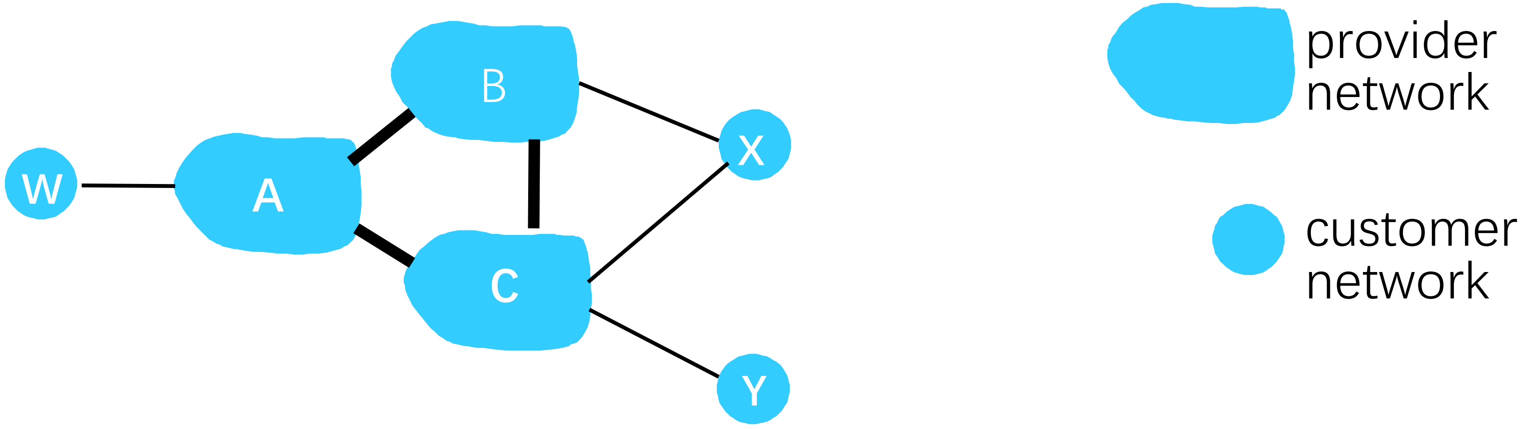
- Router may learn about more than one route to destination AS, selects route based on:
  1. local preference value attribute: policy decision
  2. shortest AS-PATH
  3. closest NEXT-HOP router: hot potato routing
  4. additional criteria

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!

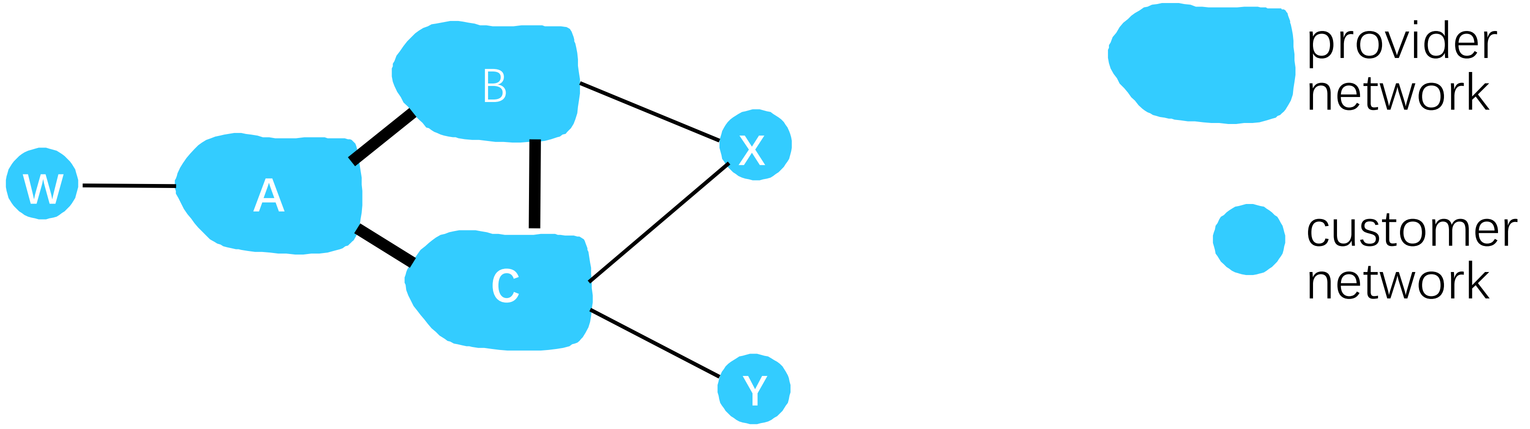
BGP: achieving policy via advertisements



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

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 through CAw (not using B) to get to w

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

Lecture 8 – Network Layer Control Plane (2)

- **Roadmap**
  1. Routing (2) - Distance vector algorithm
  2. Intra-AS routing in the Internet: OSPF
  3. Routing among the ISPs: BGP
  4. The SDN control plane
  5. ICMP
  6. SNMP

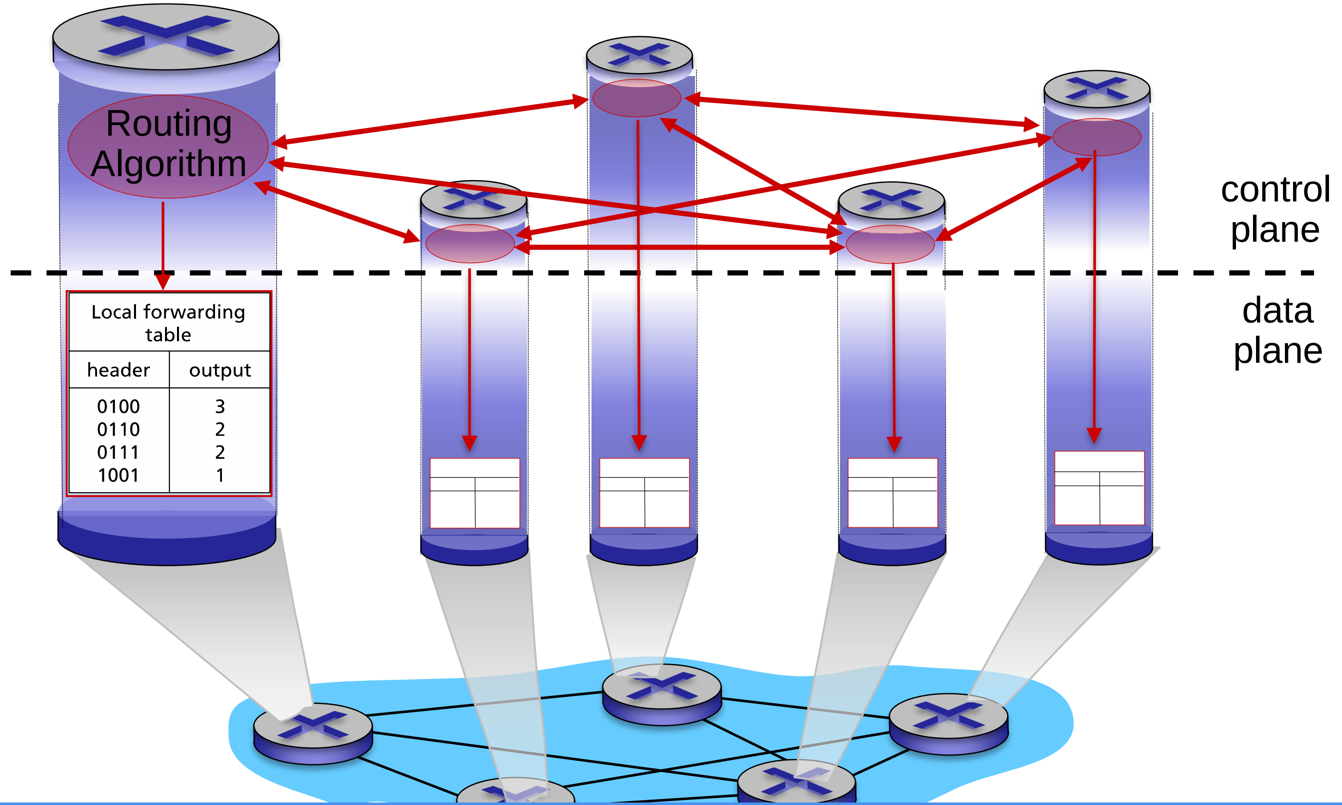
Software defined networking (SDN)

- **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**

Per-router control plane

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

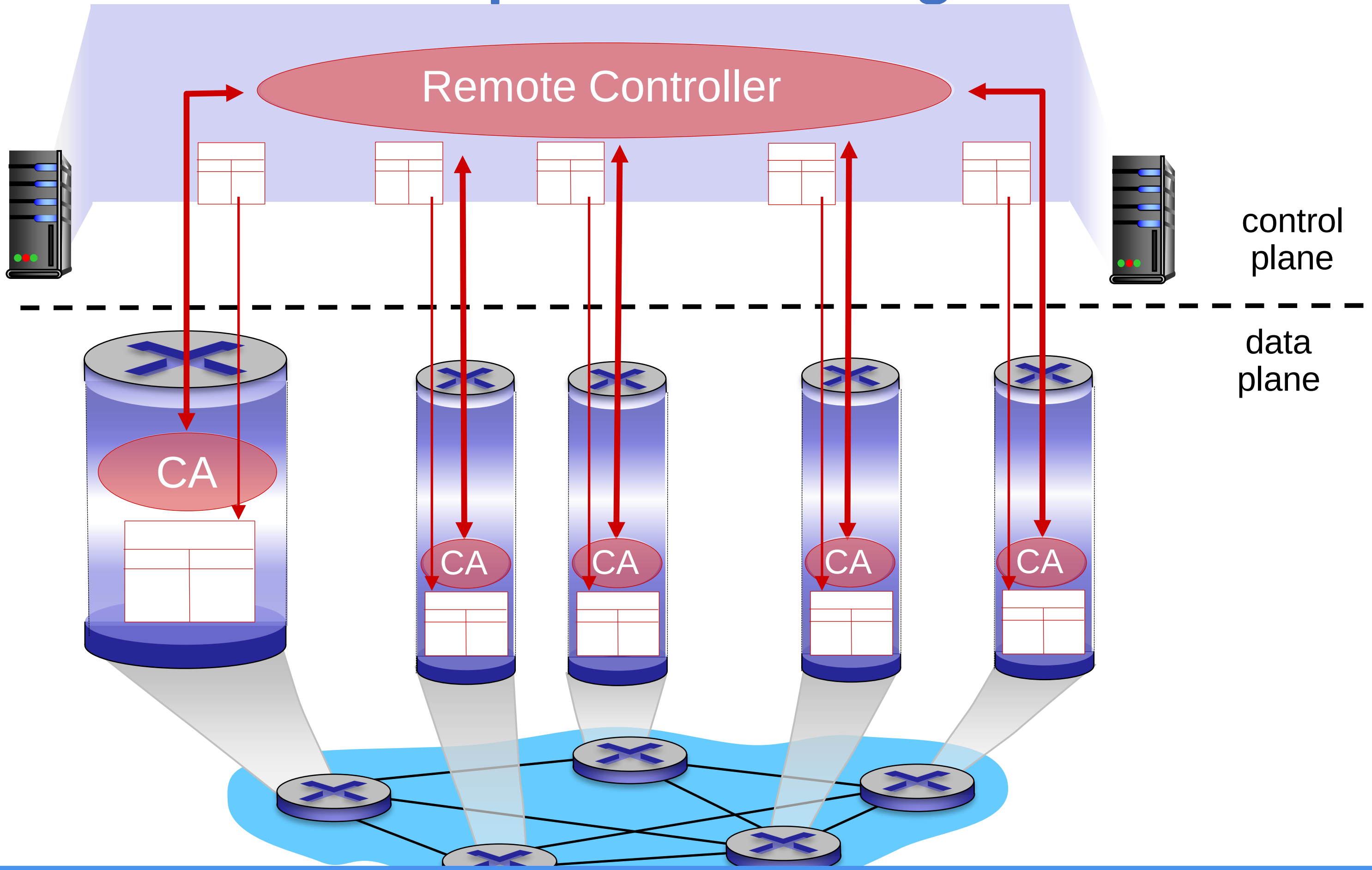


Local forwarding table

header	output
0100	3
0110	2
0111	2
1001	1

Logically centralized control plane

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

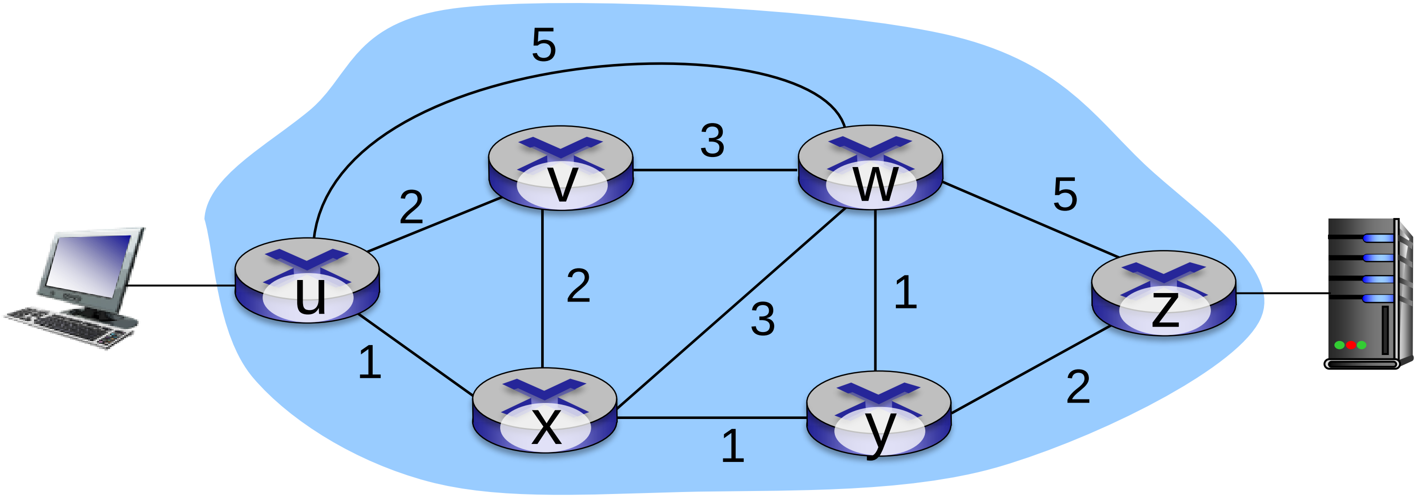


Software defined networking (SDN)

Why **a *logically centralized* control plane?**

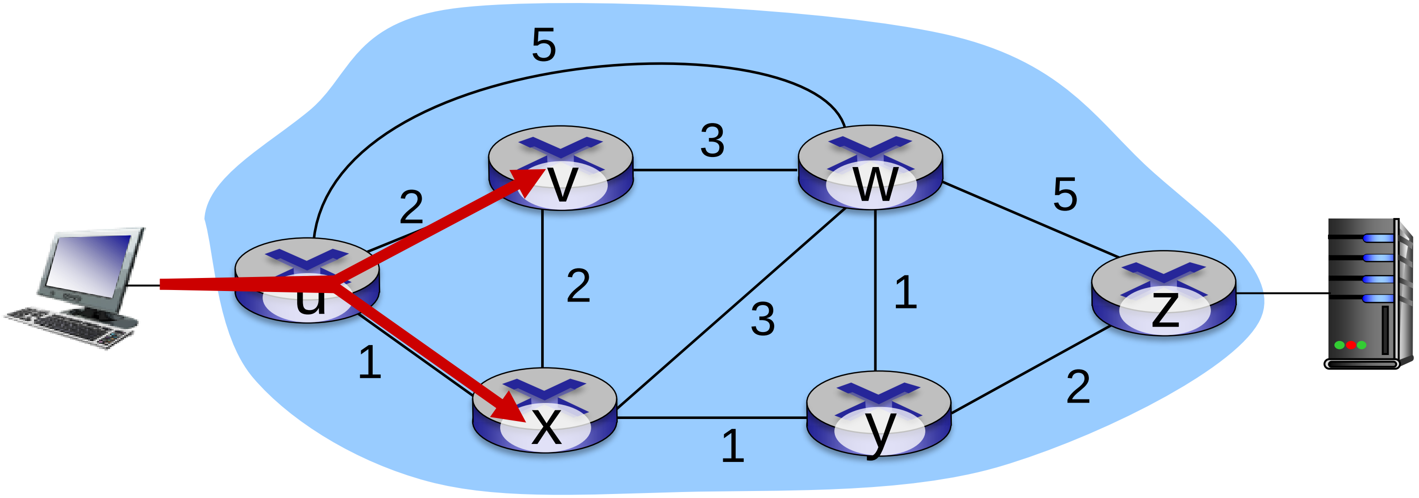
- **Easier network management: avoid router misconfigurations, greater flexibility of traffic flows**
- **Table-based forwarding (recall OpenFlow API) allows "programming" routers**
  - Centralized "programming" easier: compute tables centrally and distribute
  - Distributed "programming more difficult: compute tables as result of distributed algorithm (protocol) implemented in each and every router
- **Open (non-proprietary) implementation of control plane**

Traffic engineering: difficult traditional routing



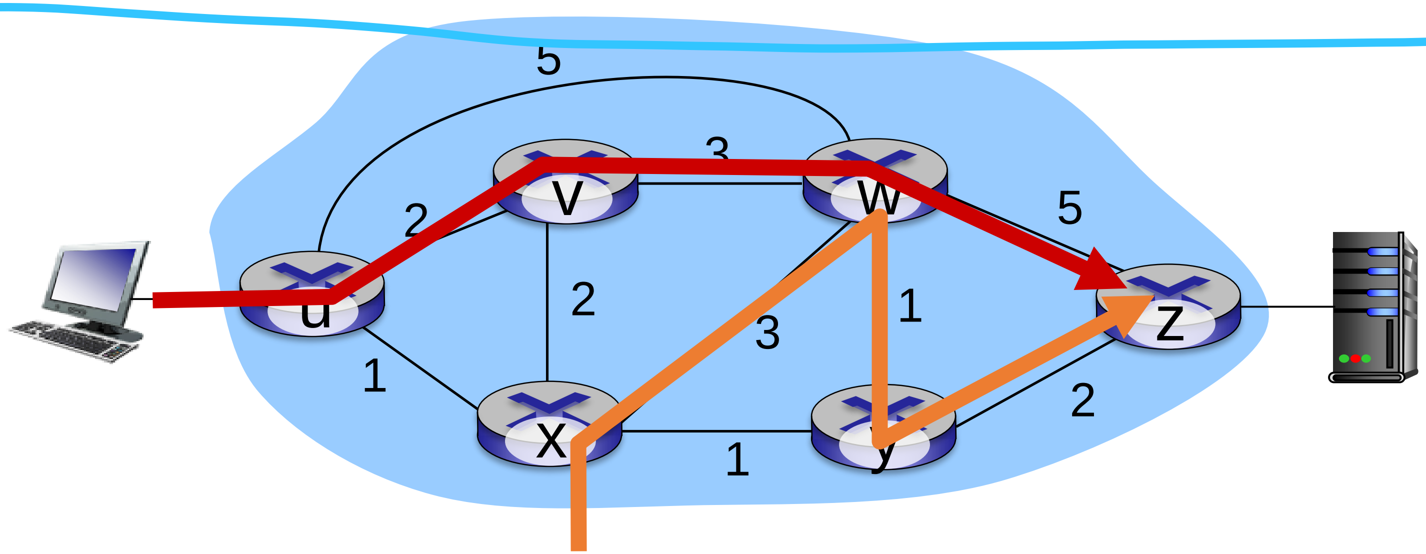
- ***Q:*** what if network operator wants u-to-z traffic to flow along *uvwz*, x-to-z traffic to flow *xwyz*?

Traffic engineering: difficult traditional routing



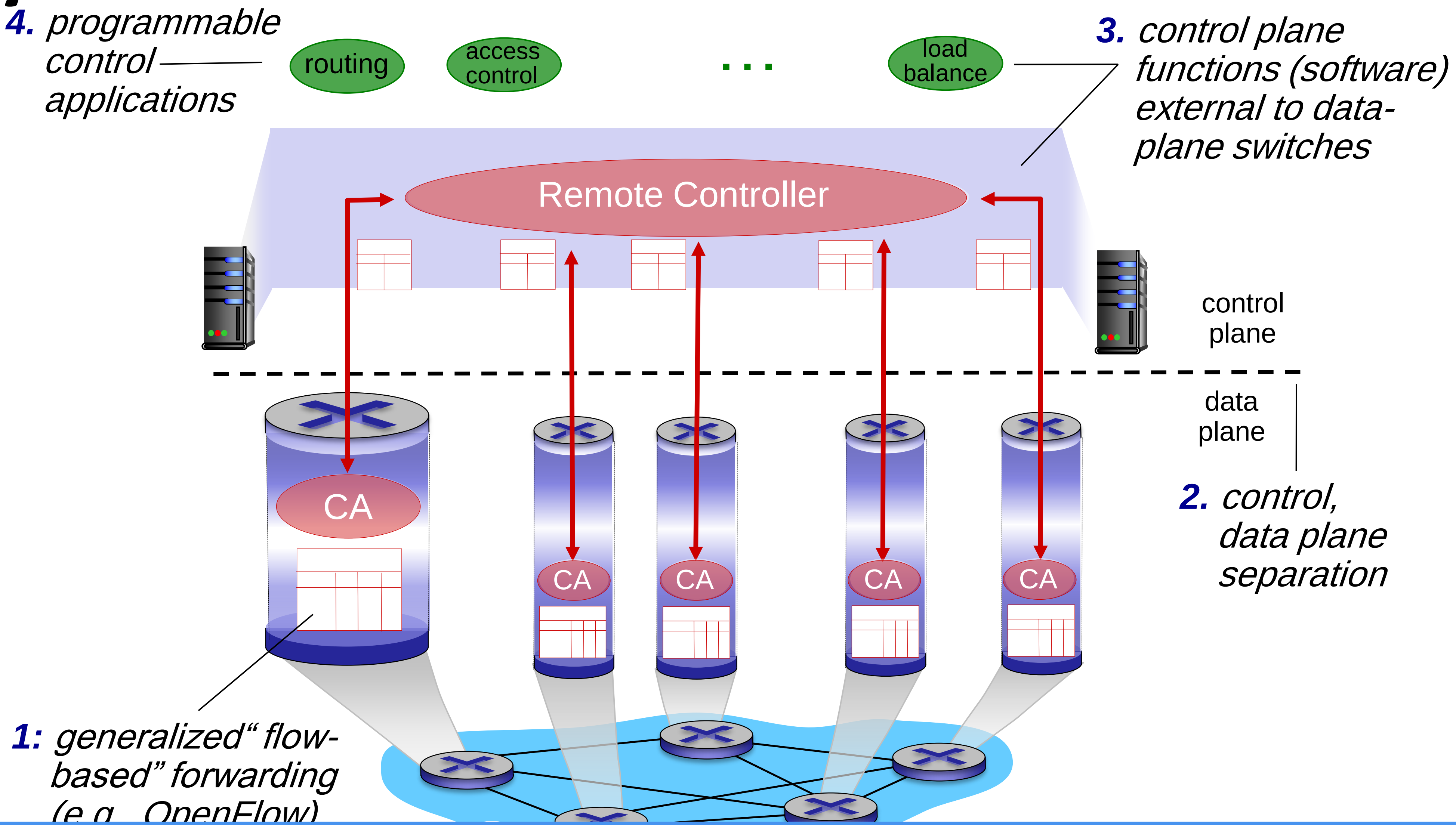
- ***Q:*** what if network operator wants to split u-to-z traffic along uvwz ***and*** uxyz (load balancing)?

Traffic engineering: difficult traditional routing



- ***Q:*** what if w wants to route red and orange traffic differently?

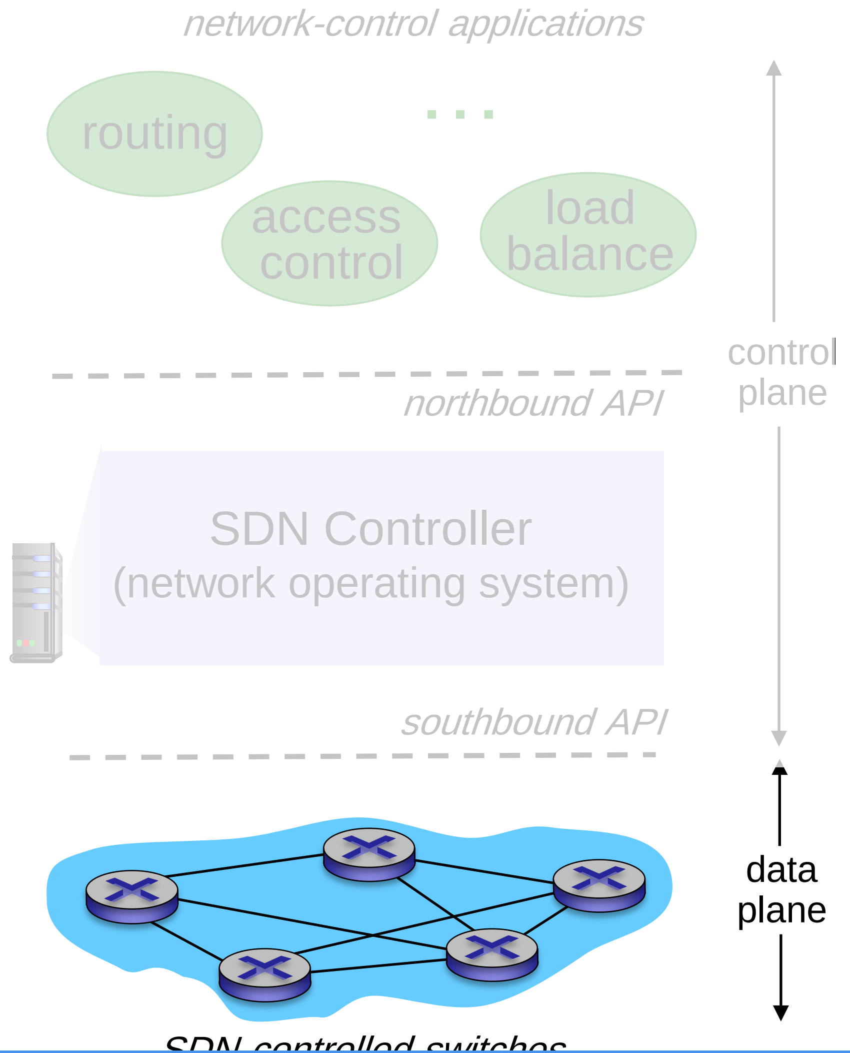
Key Characteristics of SDN Architecture


4. programmable control applications
routing
access control
. . .
load balance
3. control plane functions (software) 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

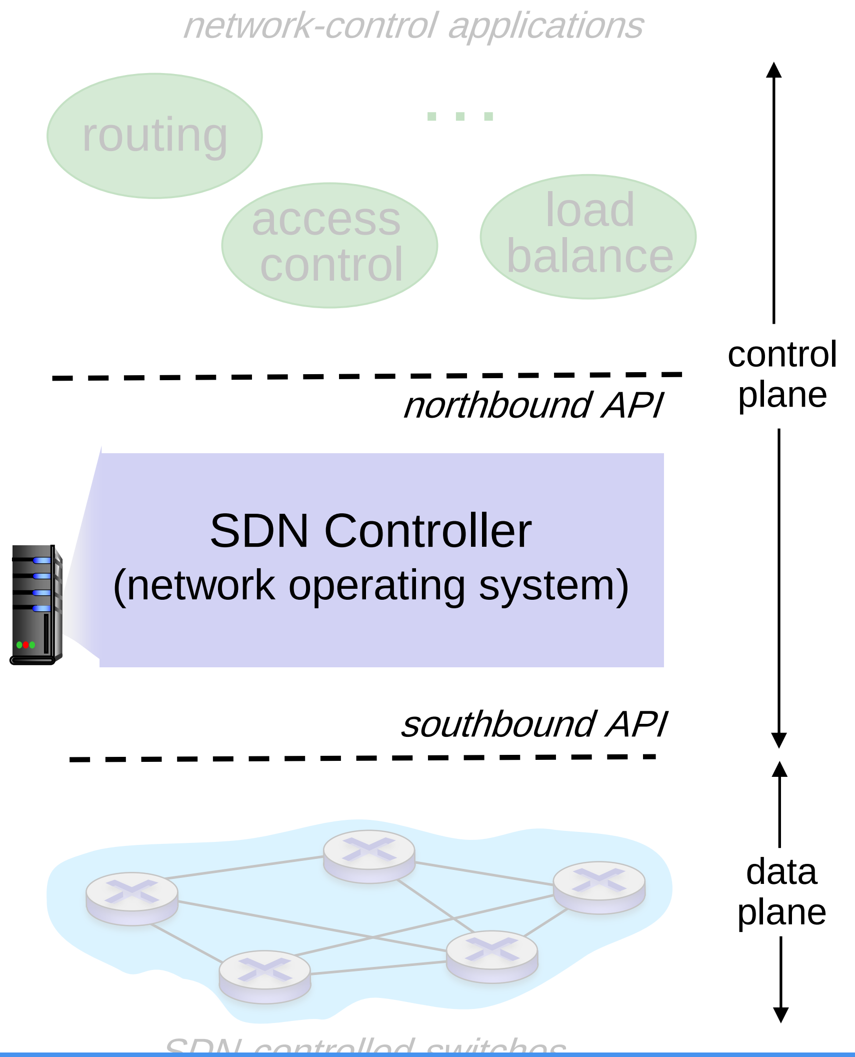
- **Fast, simple, commodity switches implementing generalized data-plane forwarding in hardware**
- **Switch flow table computed, installed by controller**
- **API for table-based switch control (e.g., OpenFlow)**
  - defines what is controllable and what is not
- **Protocol for communicating with controller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



SDN perspective: SDN controller

SDN controller (network OS):

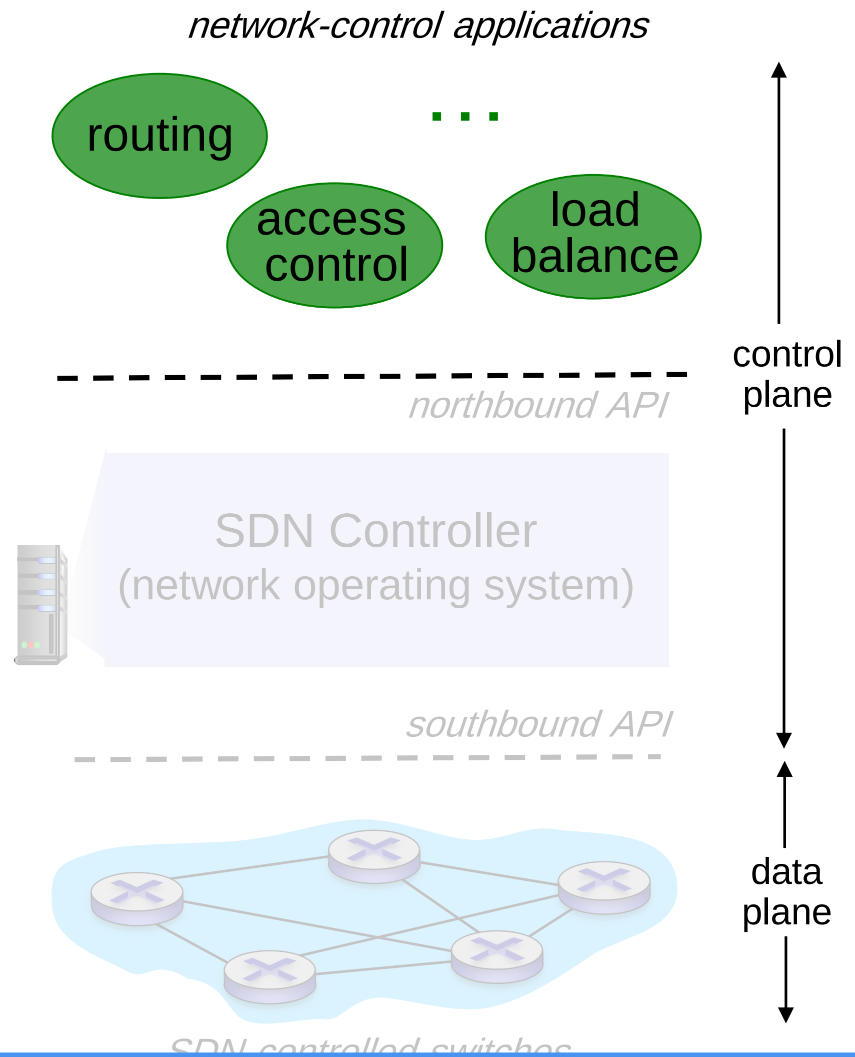
- Maintains network state information
- Interacts with network control applications "above" via northbound API
- Interacts with network switches "below" via southbound API
- Implemented as distributed system for performance, scalability, fault-tolerance, robustness



SDN perspective: control applications

Network-control apps:

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

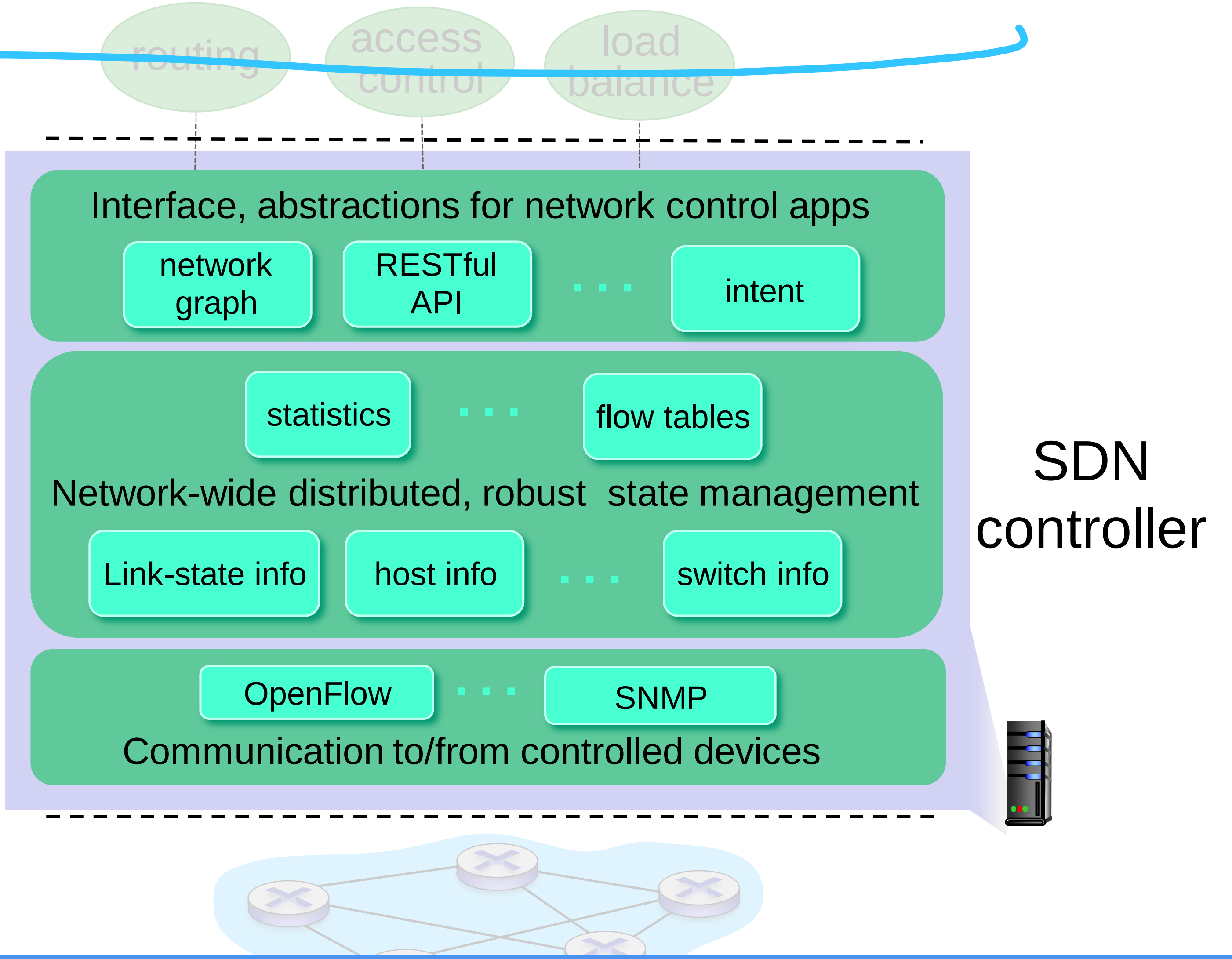


Components of SDN controller

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

. . .

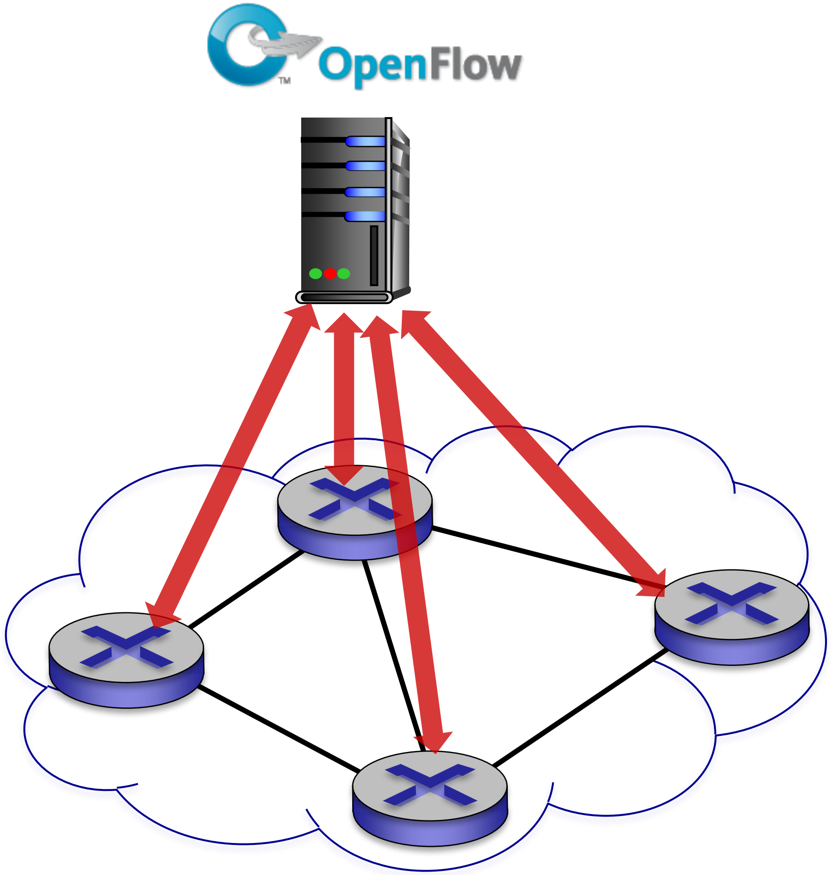
SNMP

Communication to/from controlled devices

SDN controller

OpenFlow protocol

OpenFlow Controller



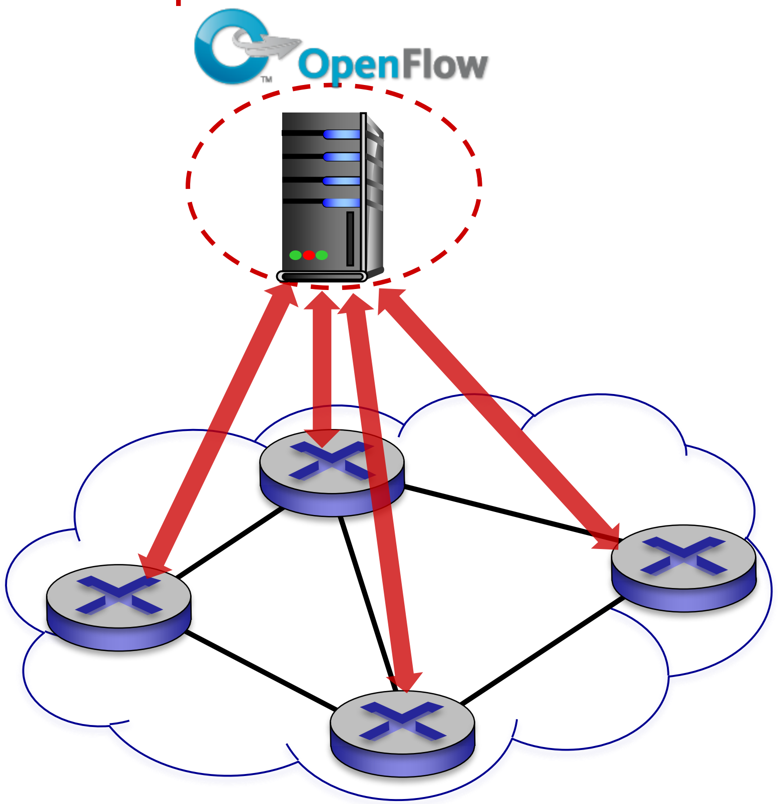
- Operates between controller, switch
- TCP used to exchange messages
  - optional encryption
- Three classes of OpenFlow messages:
  - controller-to-switch
  - asynchronous (switch to controller)
  - symmetric (misc)

OpenFlow: controller-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

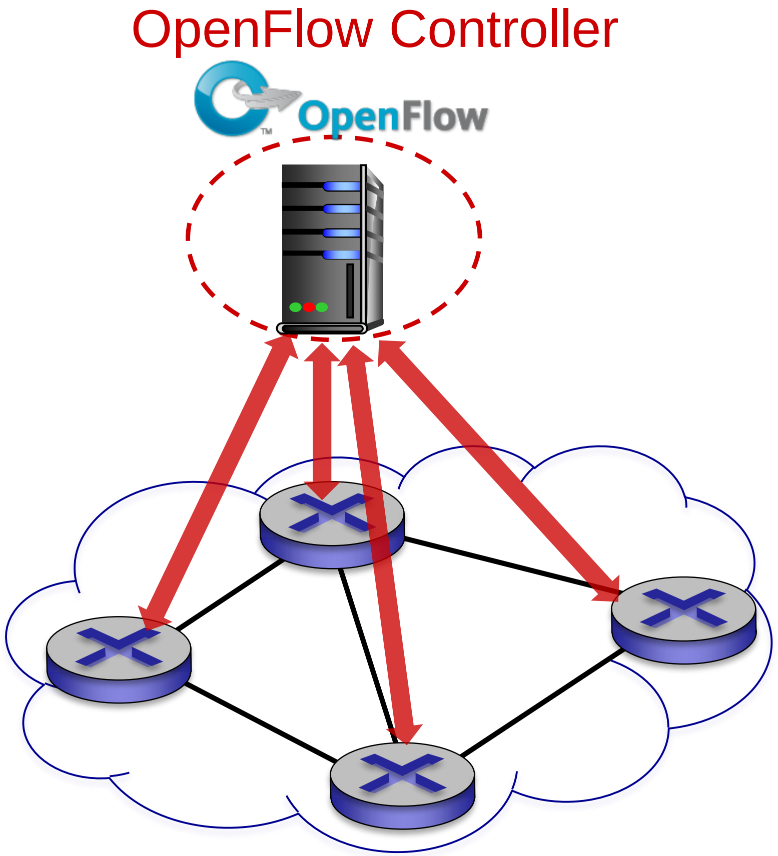
OpenFlow Controller



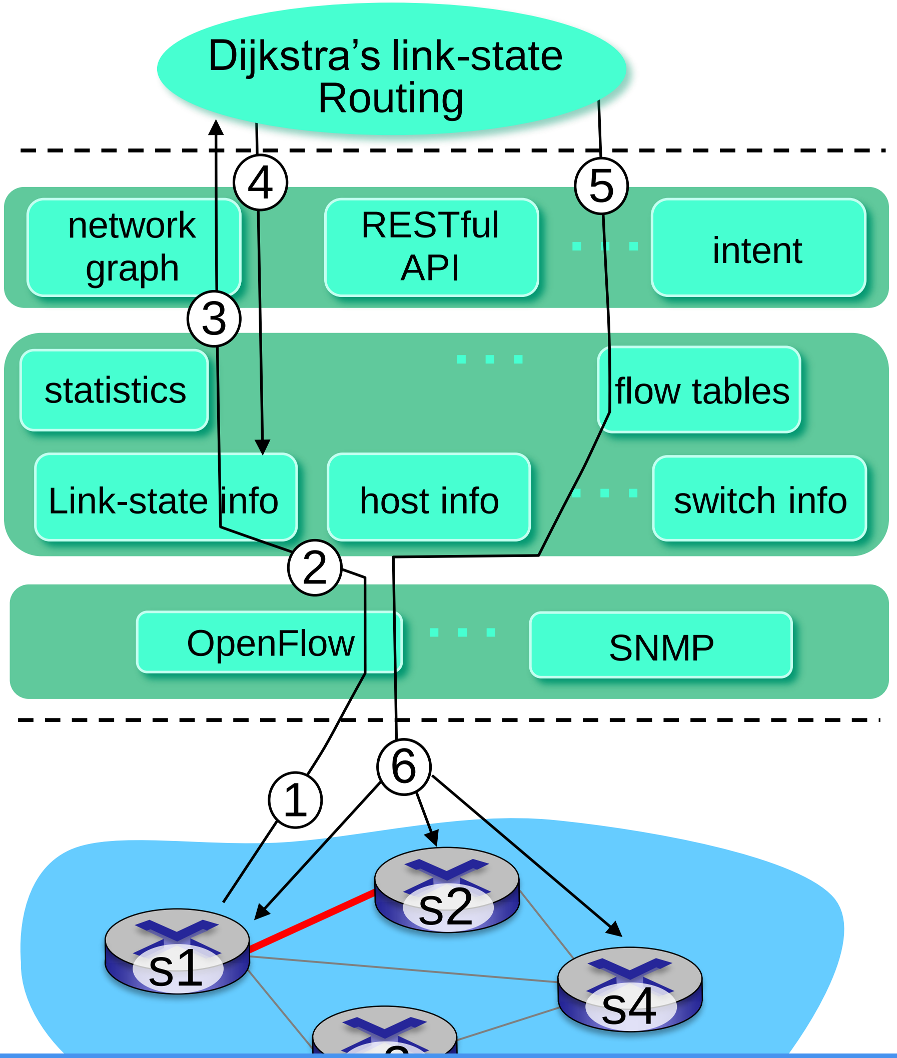
OpenFlow: switch-to-controller messages

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.



SDN: control/data plane interaction example



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

② SDN controller receives OpenFlow message, updates link status info

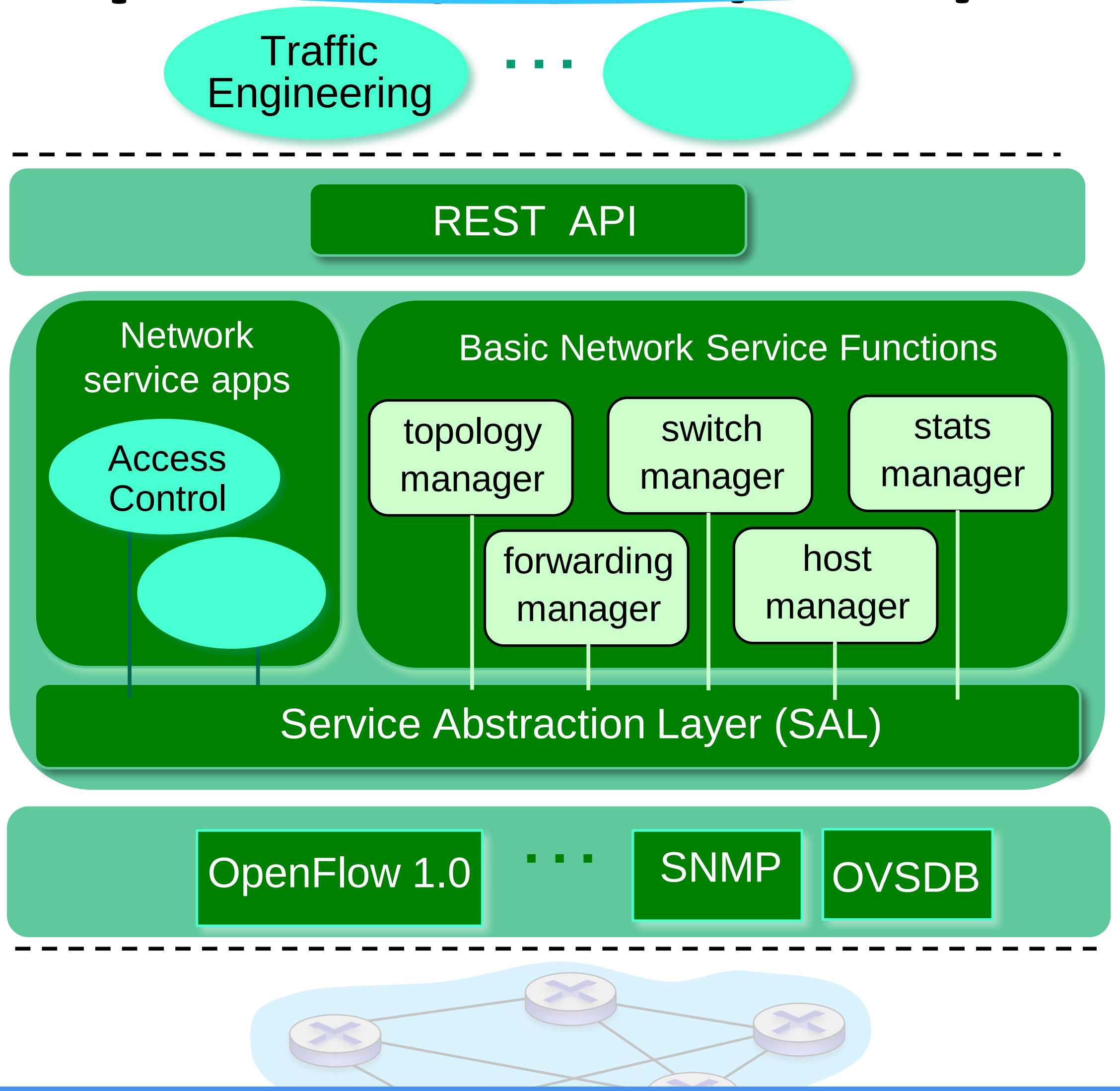
③ Dijkstra's routing algorithm application has previously registered to be called when ever link status changes. It is called.

④ Dijkstra's routing algorithm access network graph info, link state info in controller, computes new routes

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

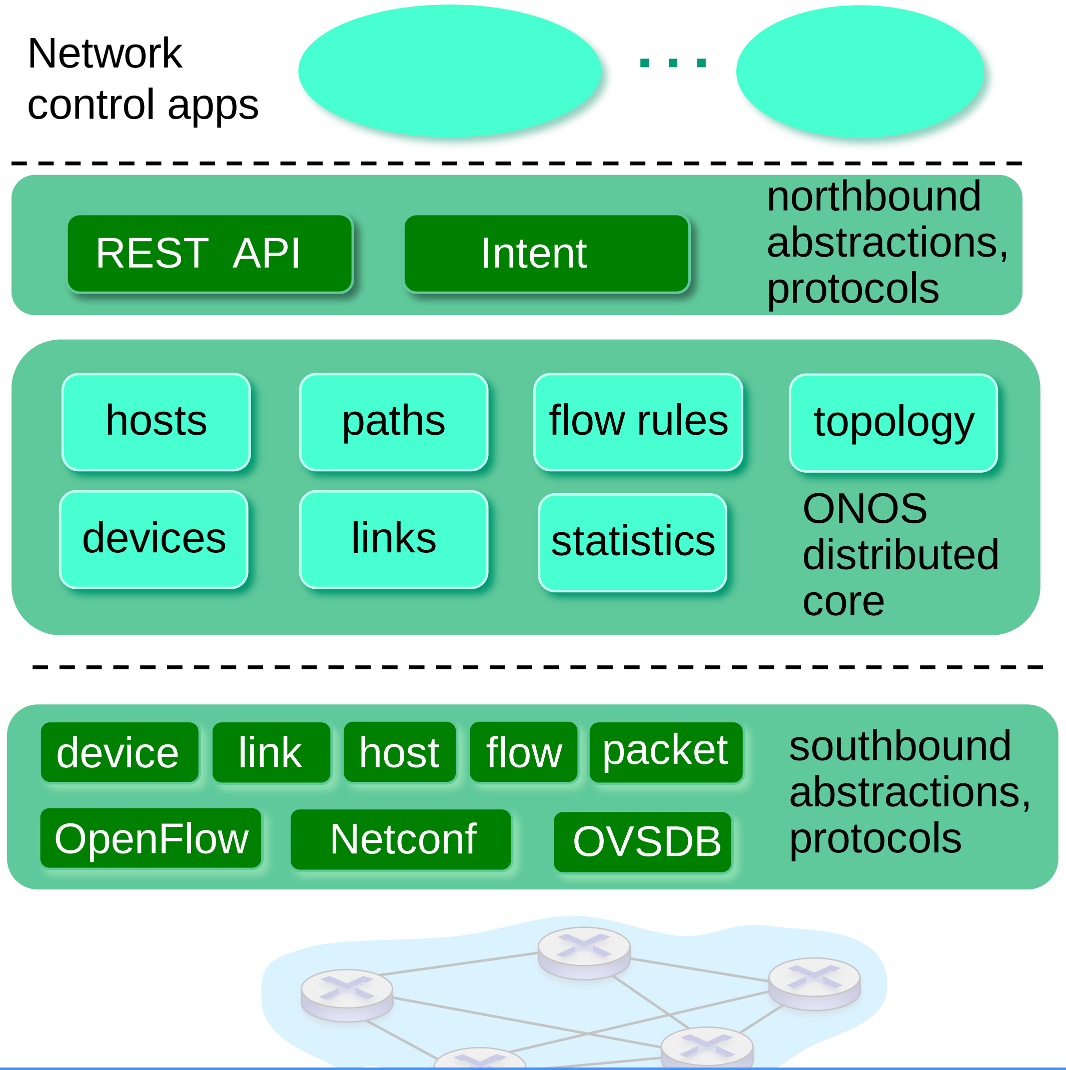
⑥ Controller uses OpenFlow to install new tables in switches that need updating

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



Network control apps

…

REST API

Intent

northbound abstractions, protocols

hosts

paths

flow rules

topology

devices

links

statistics

ONOS distributed core

device

link

host

flow

packet

OpenFlow

Netconf

OVSDB

southbound abstractions, protocols

- 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

Lecture 8 – Network Layer Control Plane (2)

- **Roadmap**

1. Routing (2) - Distance vector algorithm
2. Intra-AS routing in the Internet: OSPF
3. Routing among the ISPs: BGP
4. The SDN control plane
5. ICMP: The Internet Control Message Protocol
6. 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' field and 'code' field plus the first 8 bytes of IP datagram that caused the 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

TTL=n 每过一个路由-1 ⇒ TTL=0 抛弃这个包.

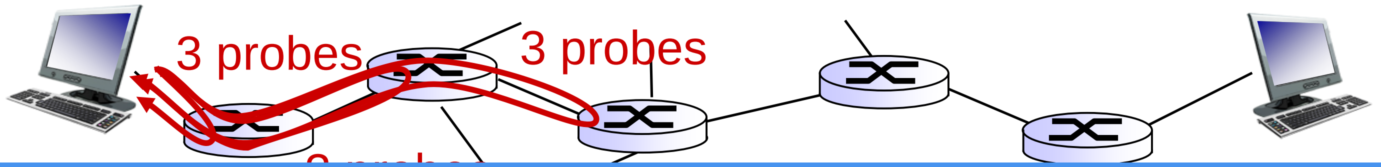
- Source sends series of UDP segments to destination
  - first set has TTL =1
  - second set has TTL=2, etc.
  - with an unlikely dest. port number
- When datagram in *n*th set arrives to nth router:
  - router discards datagram and sends source ICMP "TTL expired" message (type 11, code 0)
  - ICMP message include name of router & IP address

发送回去

- 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



Lecture 8 – Network Layer Control Plane (2)

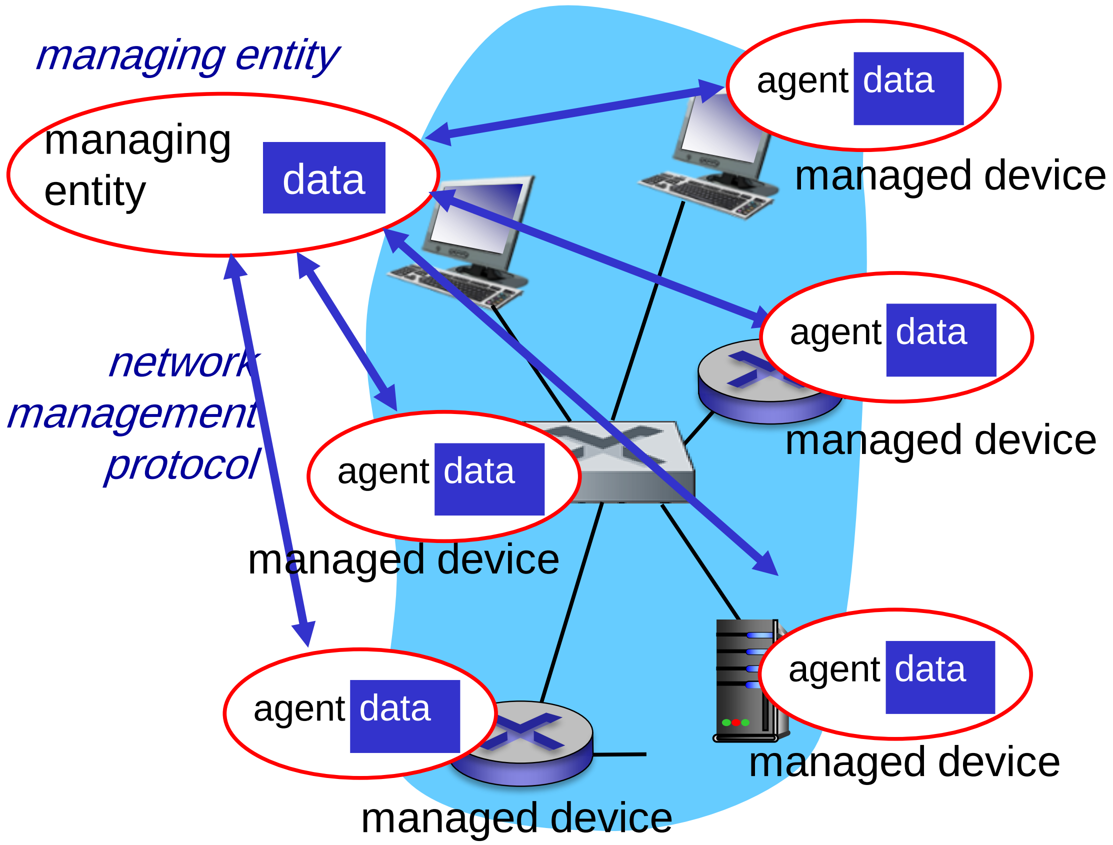
- **Roadmap**
  1. Routing (2) - Distance vector algorithm
  2. Intra-AS routing in the Internet: OSPF
  3. Routing among the ISPs: BGP
  4. The SDN control plane
  5. ICMP
  6. SNMP: Network Management

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." -- Saydam 1996

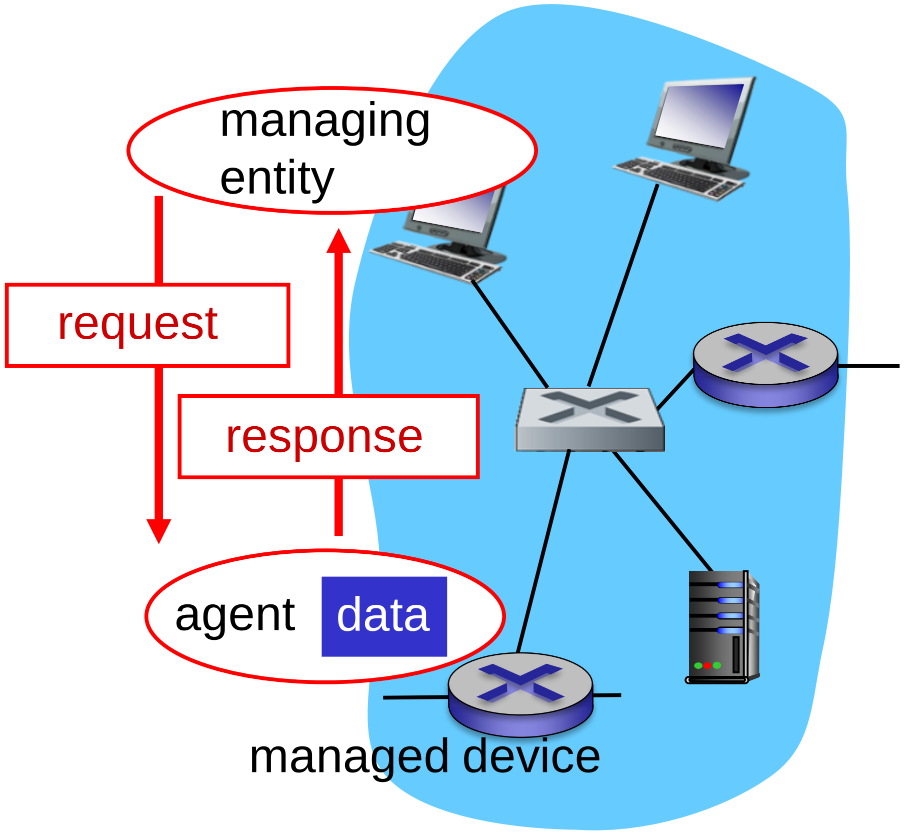
Infrastructure for network management



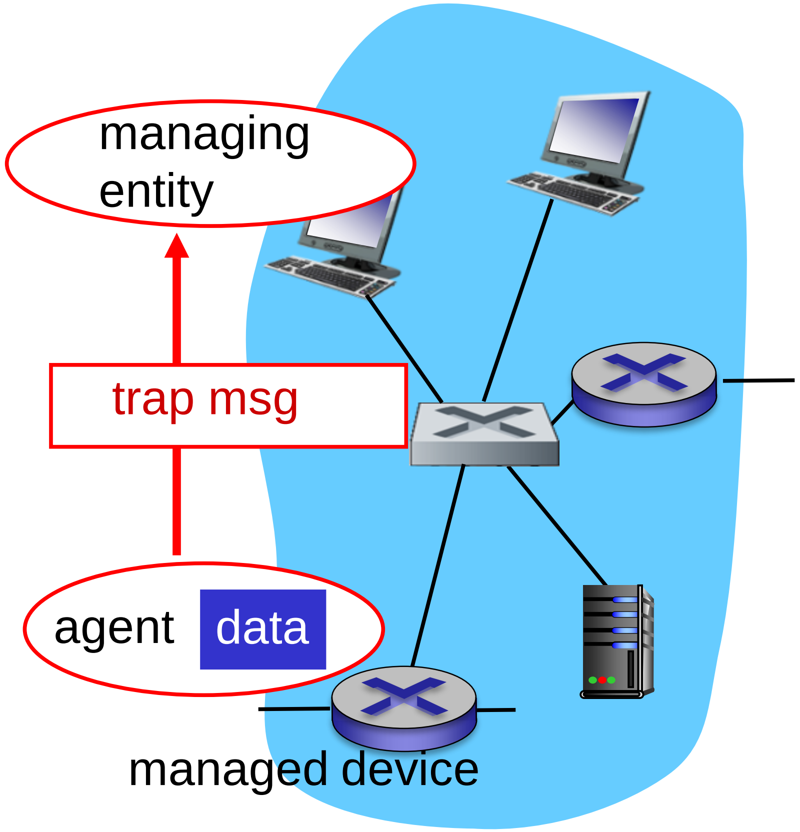
Managed devices contain *managed objects* whose data is gathered into a *Management Information Base (MIB)*

SNMP protocol

Two ways to convey MIB info, commands:



request/response mode

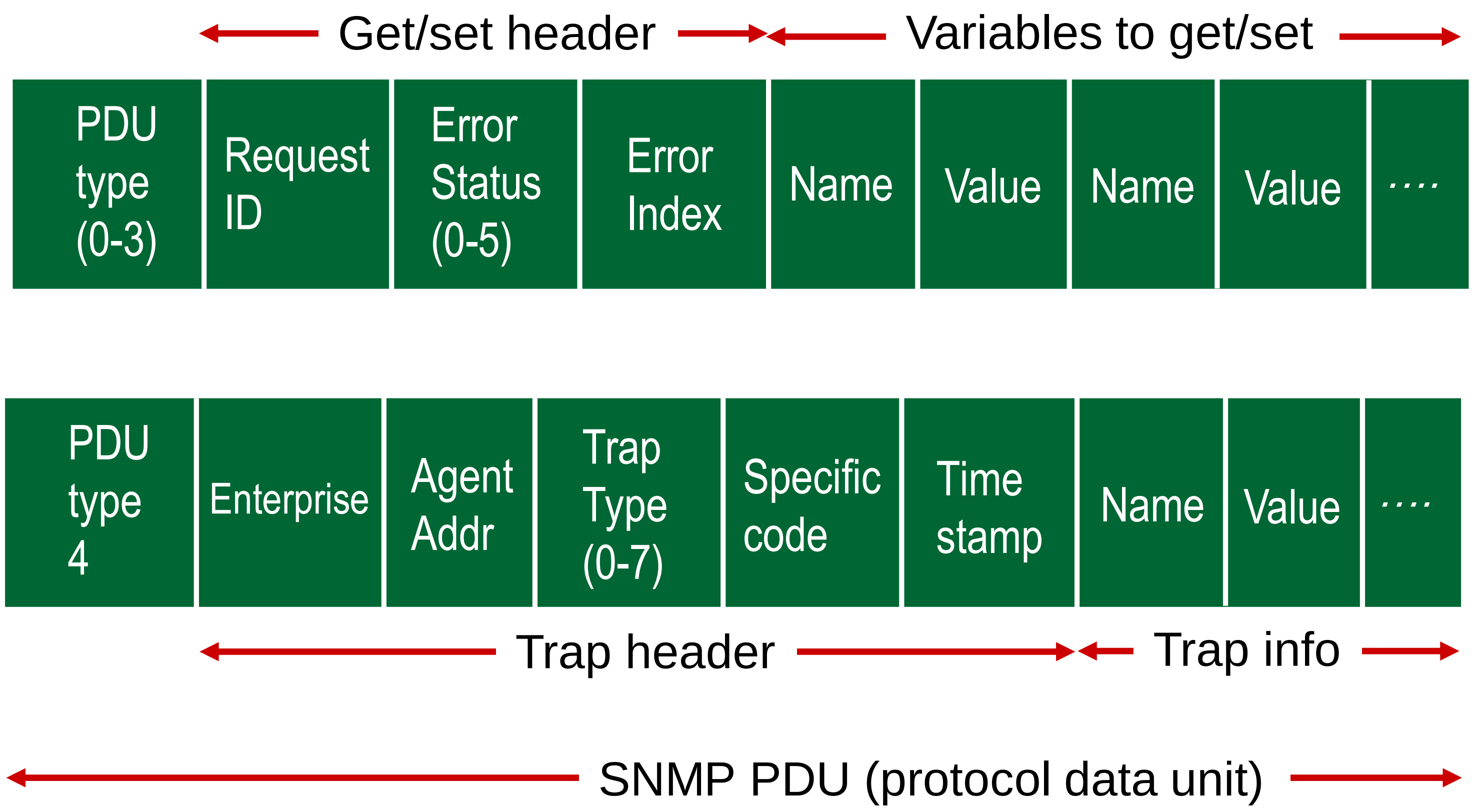


trap mode

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



Get/set header | Variables to get/set

PDU type (0-3)	Request ID	Error Status (0-5)	Error Index	Name	Value	Name	Value	….

PDU type 4	Enterprise	Agent Addr	Trap Type (0-7)	Specific code	Time stamp	Name	Value	….

Trap header | Trap info

SNMP PDU (protocol data unit)

More on network management: see earlier editions of text!

Thanks.

- **Addresses**
  - Email: Wenjun.Fan@xjtlu.edu.cn
  - Office: EE 214
- **Office hours**
  - Monday: 12:00 – 13:00
  - Tuesday: 12:00 – 13:00