

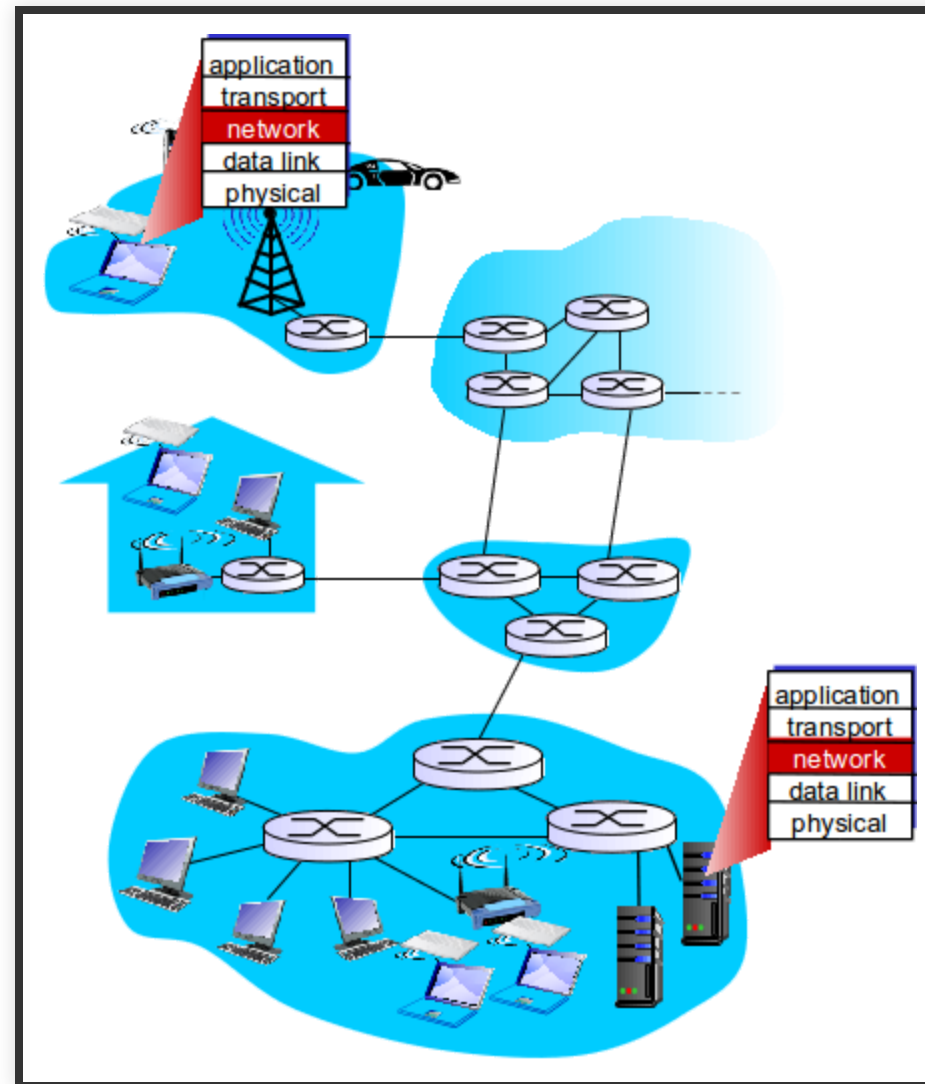
A background pattern of a network diagram consisting of interconnected nodes and lines. The nodes are represented by circles of varying sizes and shades of gray, connected by thin gray lines. The overall appearance is that of a complex, interconnected network structure.

NETWORK LAYER: CONTROL PLANE

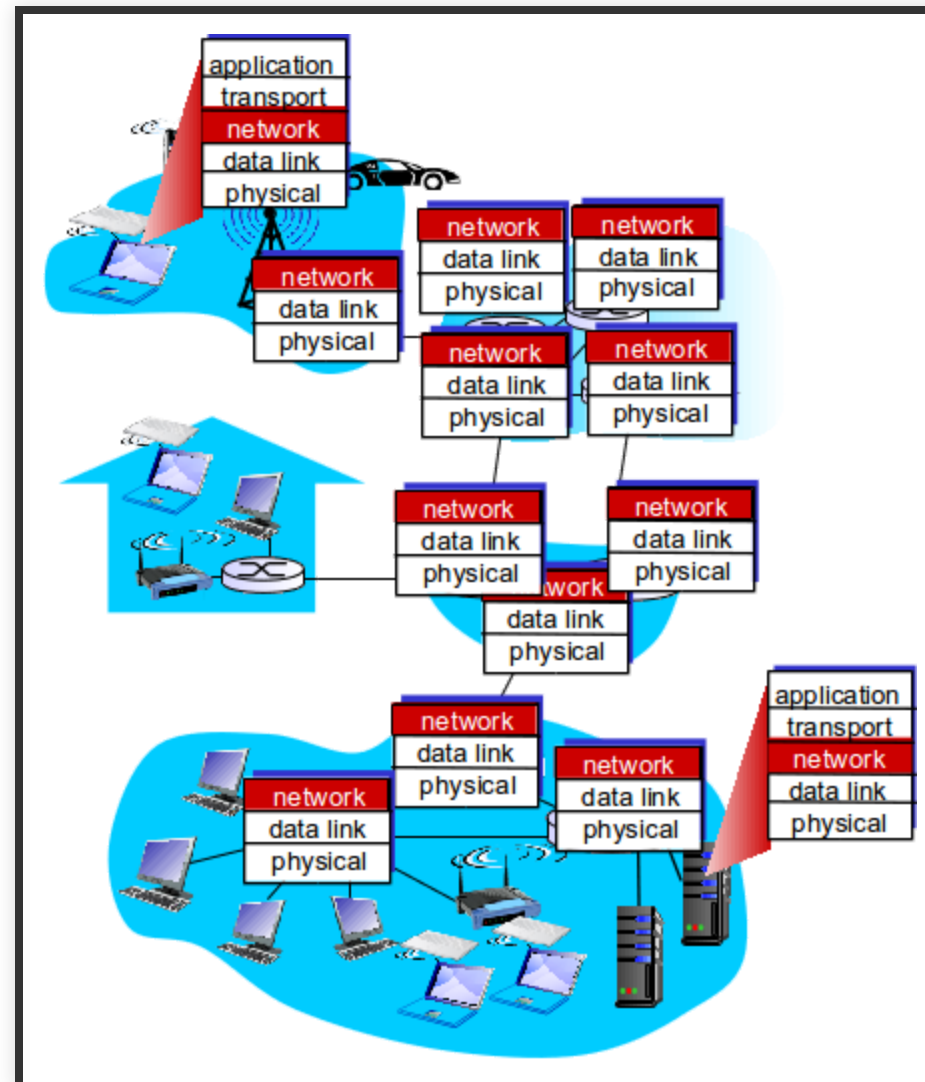
GOALS

- 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

INTRODUCTION



INTRODUCTION



NETWORK LAYER FUNCTIONS

Data plane

Forwarding: move packets from router's input to appropriate router output

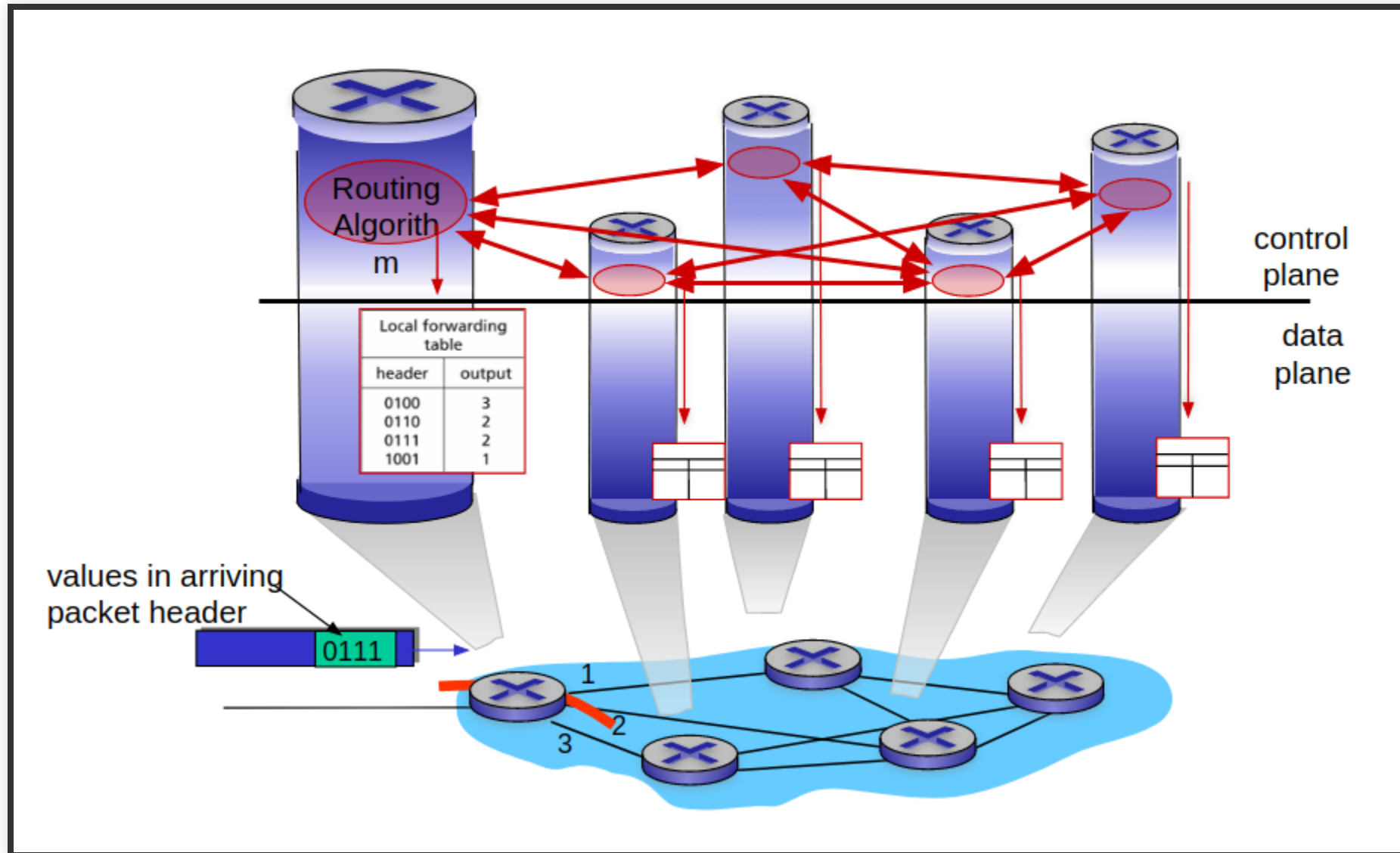
Control plane

Routing: determine route taken by packets from source to dest.

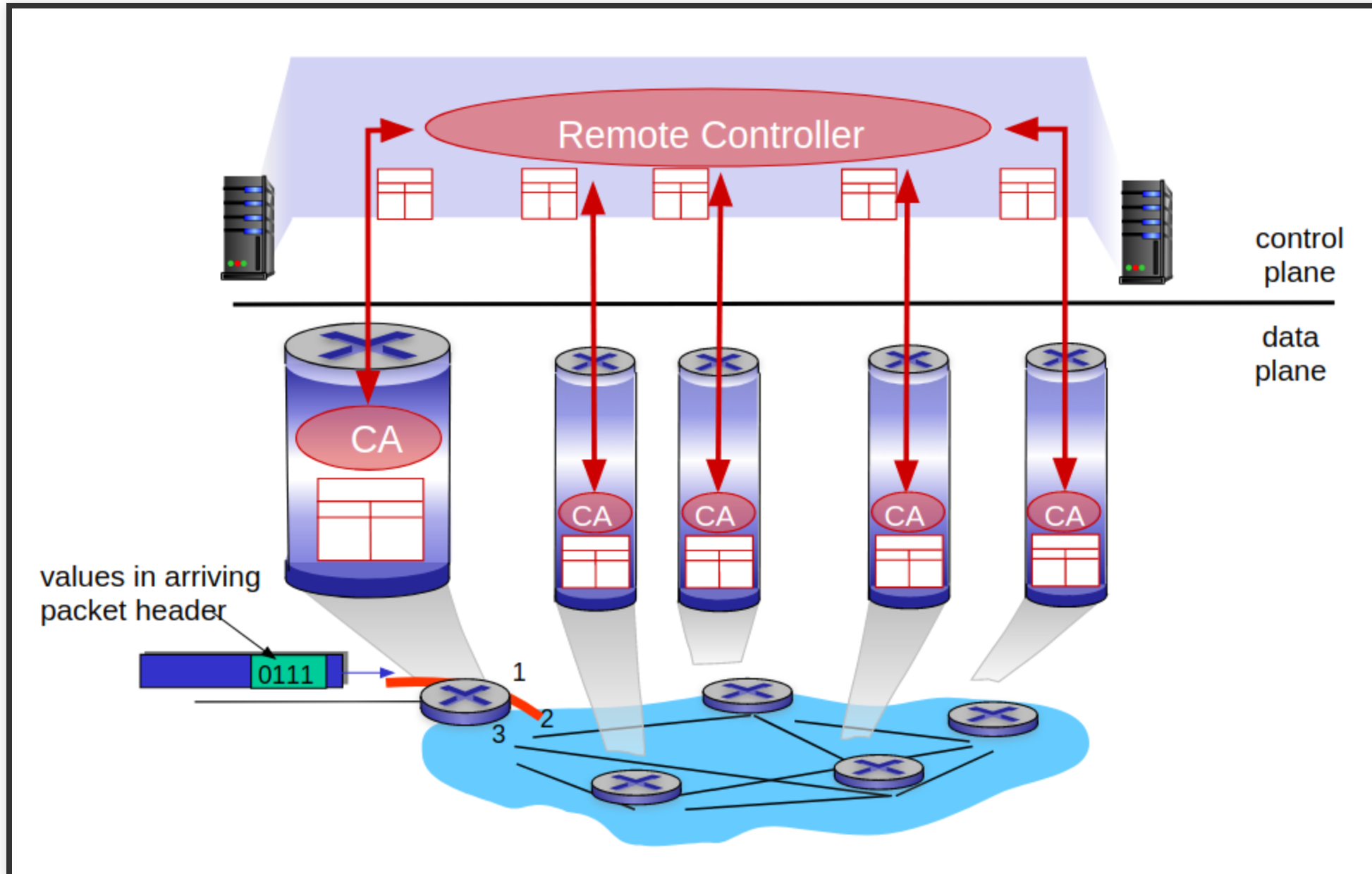
Two approaches to structuring network control plane:

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

PER-ROUTER CONTROL PLANE

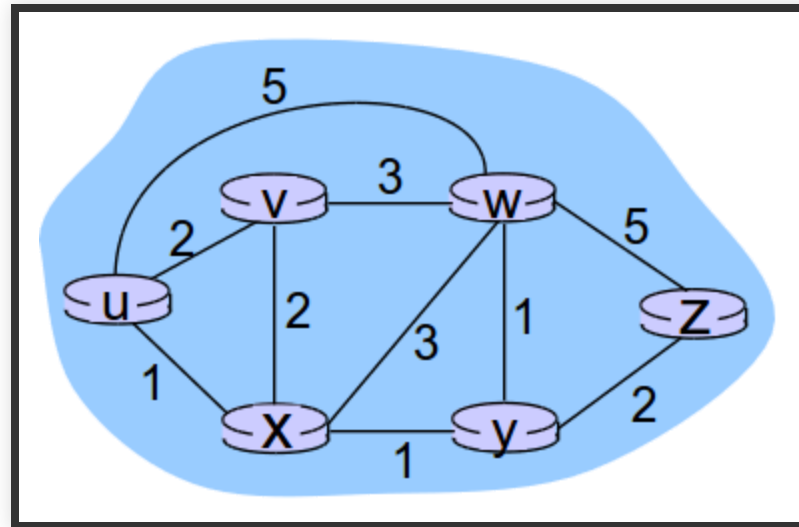


LOGICALLY CENTRALIZED CONTROL PLANE (SDN)



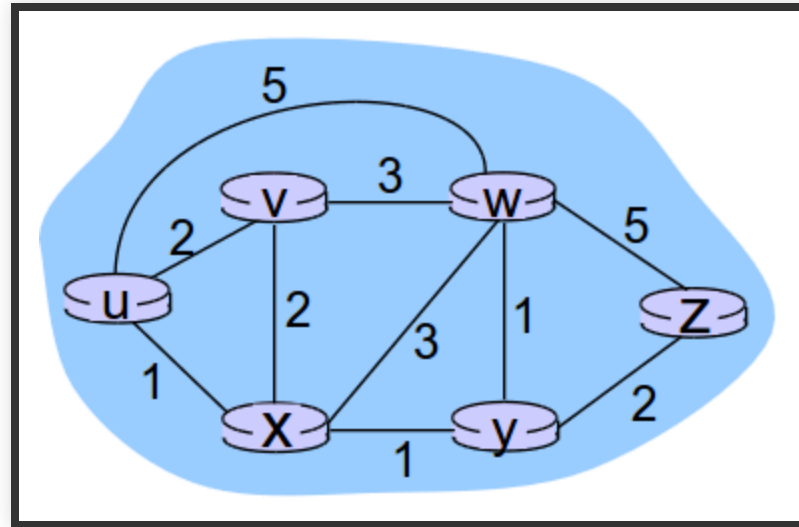
ROUTING ALGORITHMS

GRAPH ABSTRACTION



- graph: $G = (N,E)$
- $N = \text{set of routers} = \{ u, v, w, x, y, z \}$
- $E = \text{set of links} = \{ (u,v), (u,x), (v,x), (v,w), (x,w), (x,y), (w,y), (w,z), (y,z) \}$
- 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

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

GRAPH ABSTRACTION: COSTS

- ❗ **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?

ROUTING ALGORITHM CLASSIFICATION

Q: Static or dynamic?

- Static:
 - routes change slowly over time
- Dynamic:
 - routes change more quickly
 - periodic update
 - in response to link cost changes

A LINK-STATE ROUTING ALGORITHM

! Dijkstra's algorithm

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

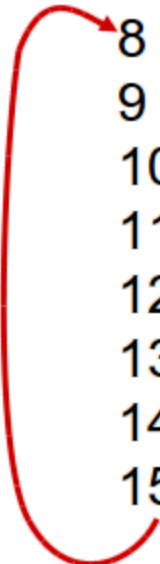
A LINK-STATE ROUTING ALGORITHM

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

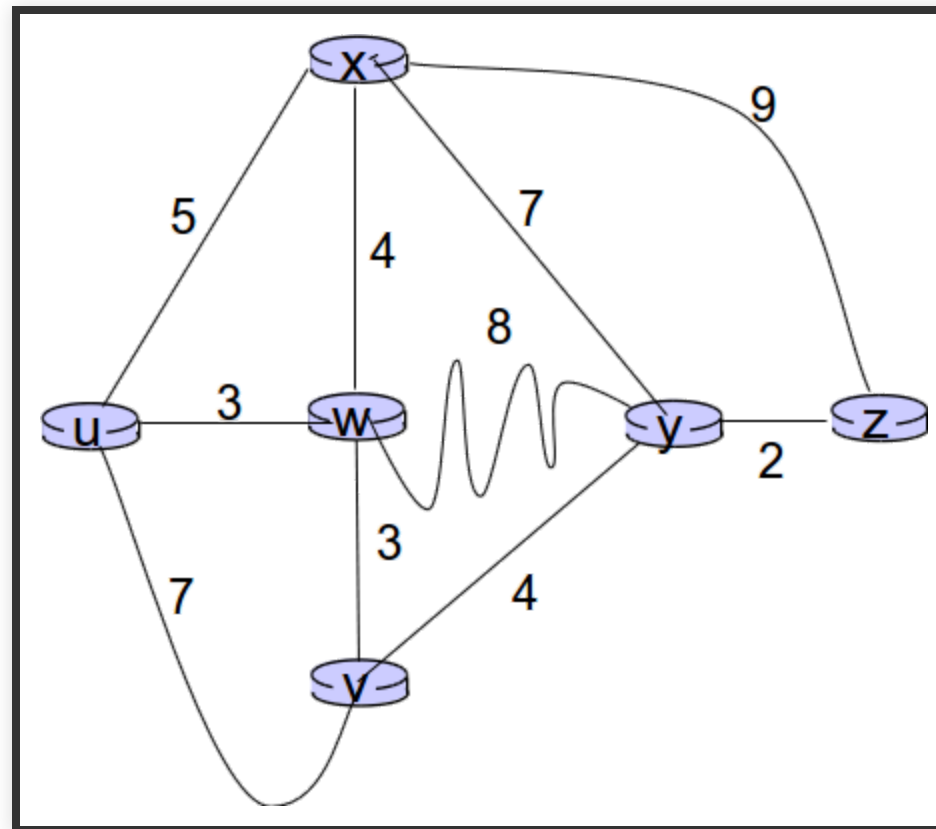
DIJKSTRA'S ALGORITHM

```
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  $N'$ 
```

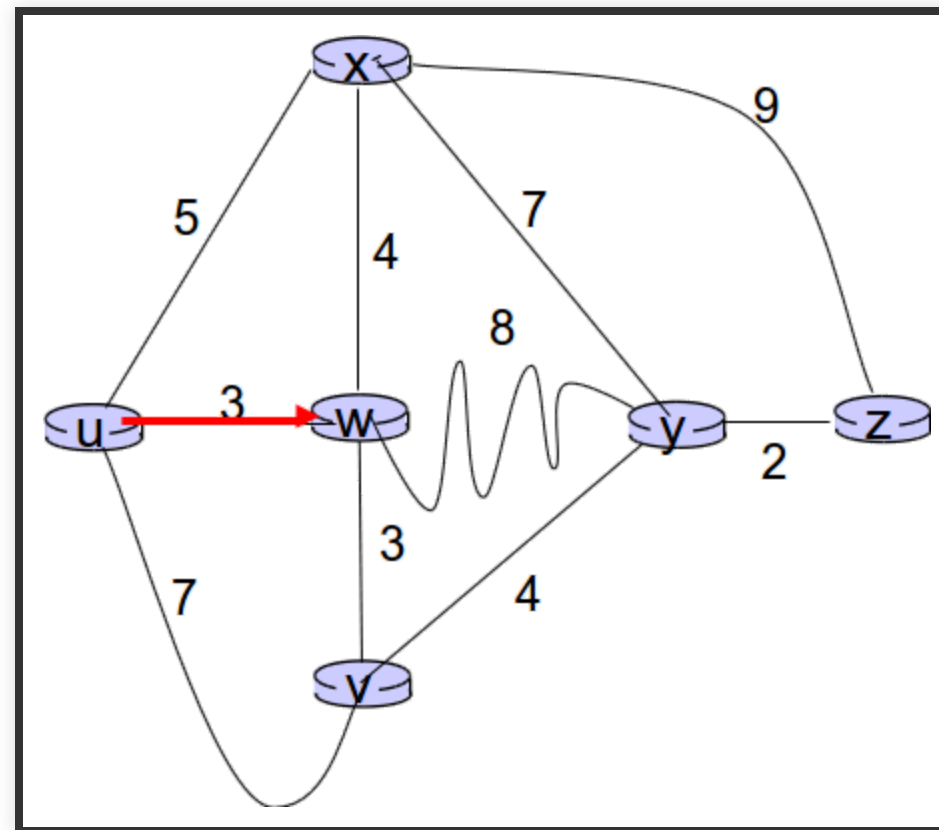


DIJKSTRA'S ALGORITHM: EXAMPLE

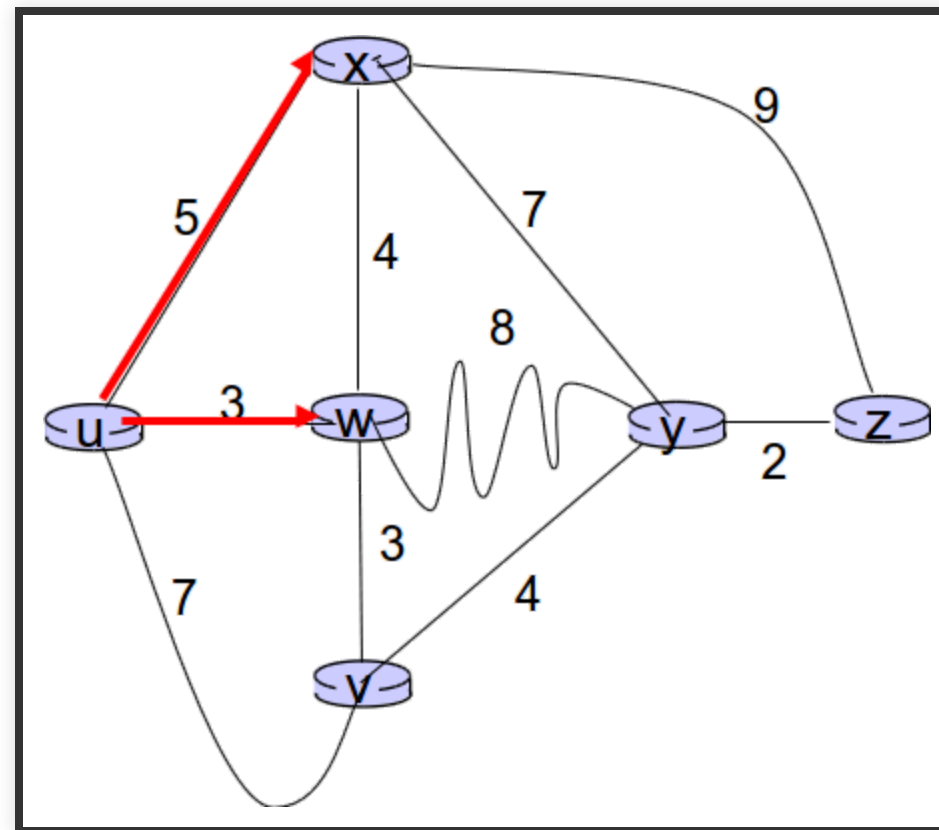
Shortest paths from u



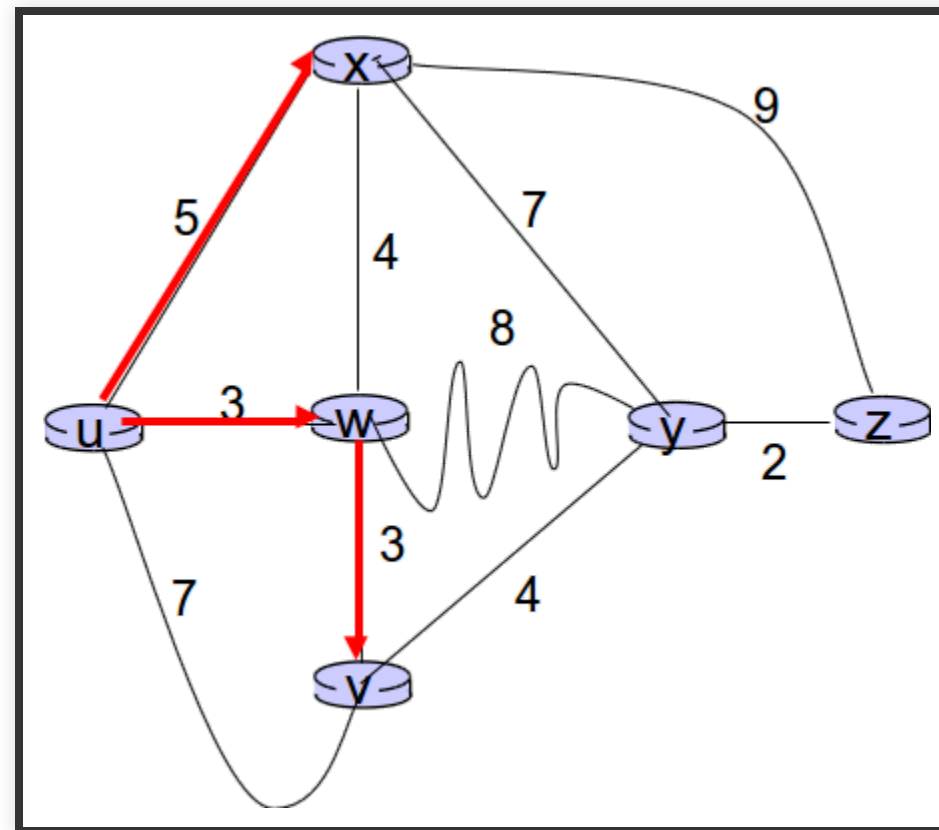
DIJKSTRA'S ALGORITHM: EXAMPLE



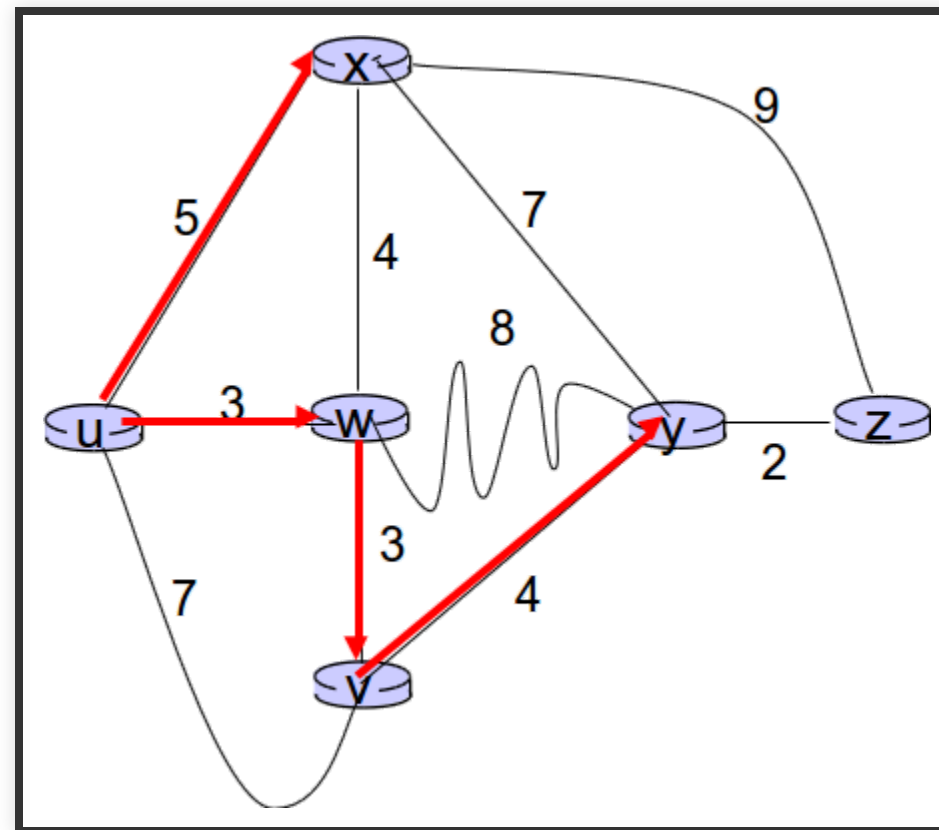
DIJKSTRA'S ALGORITHM: EXAMPLE



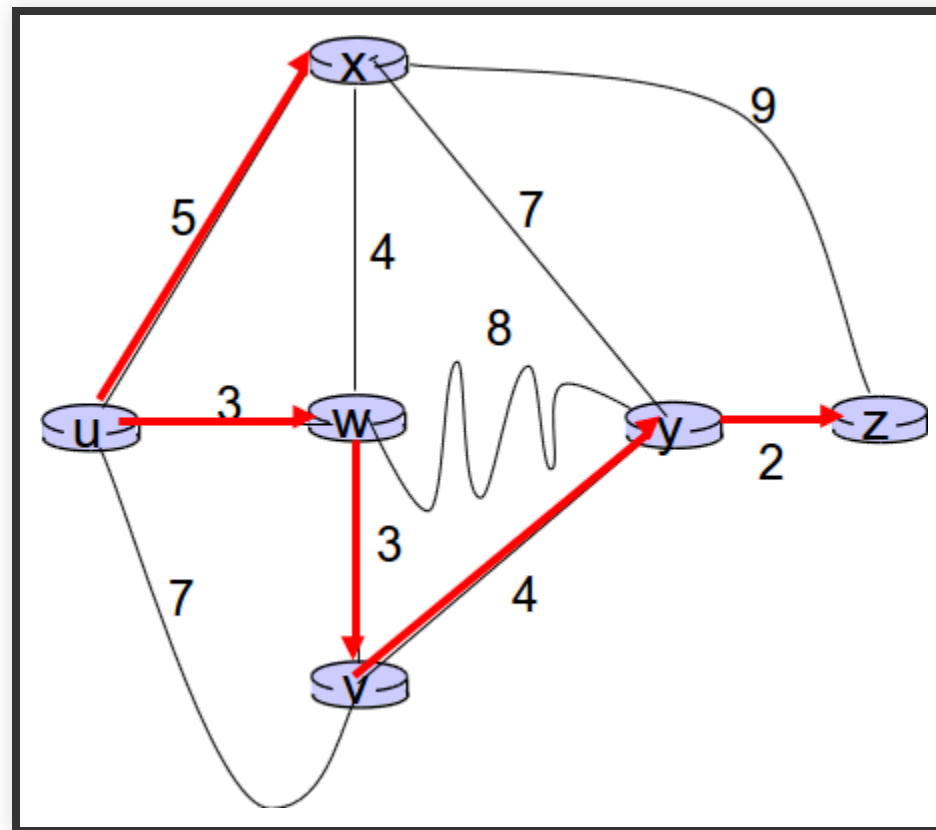
DIJKSTRA'S ALGORITHM: EXAMPLE



DIJKSTRA'S ALGORITHM: EXAMPLE



DIJKSTRA'S ALGORITHM: EXAMPLE

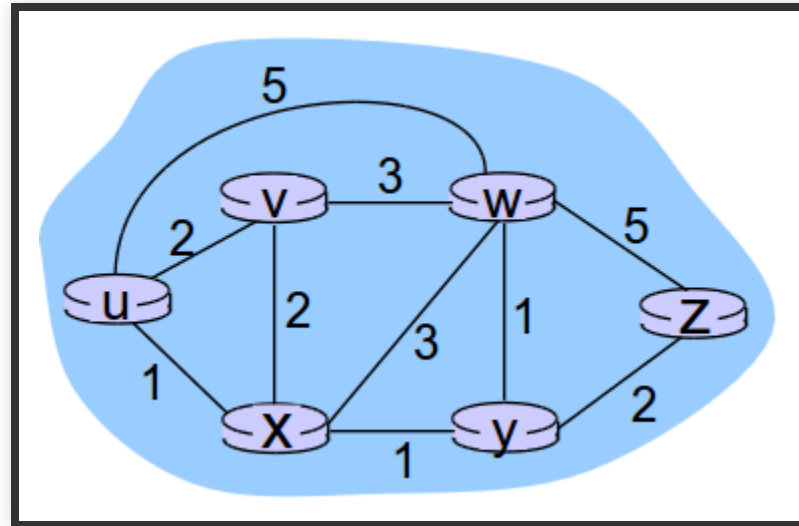


DIJKSTRA'S ALGORITHM: EXAMPLE

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

Step	N'	D(v)	D(w)	D(x)	D(y)	D(z)
		p(v)	p(w)	p(x)	p(y)	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					

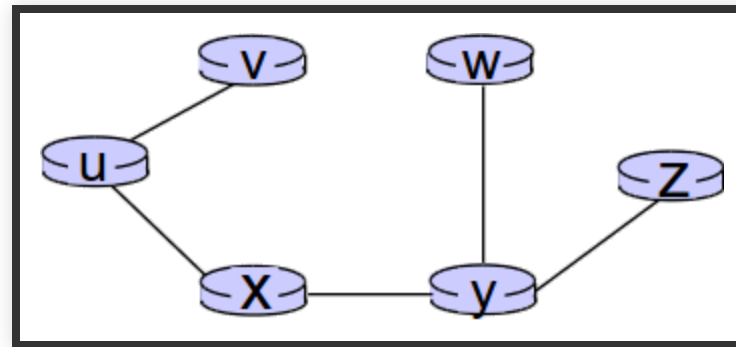
DIJKSTRA'S ALGORITHM: ANOTHER EXAMPLE



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					

DIJKSTRA'S ALGORITHM: ANOTHER EXAMPLE

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)

DIJKSTRA'S ALGORITHM, DISCUSSION

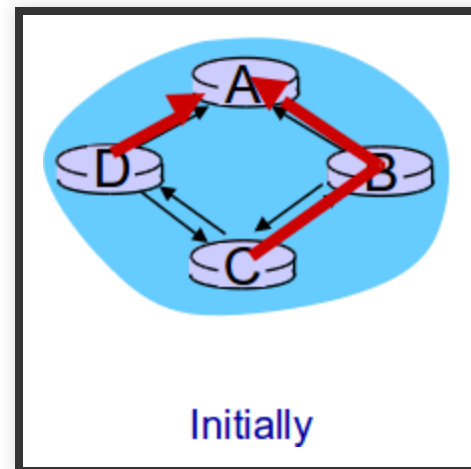
❗ Algorithm complexity: n nodes

- each iteration: need to check all nodes, w , not in N
- $n(n+1)/2$ comparisons: $O(n^2)$
- more efficient implementations possible: $O(n \log(n))$

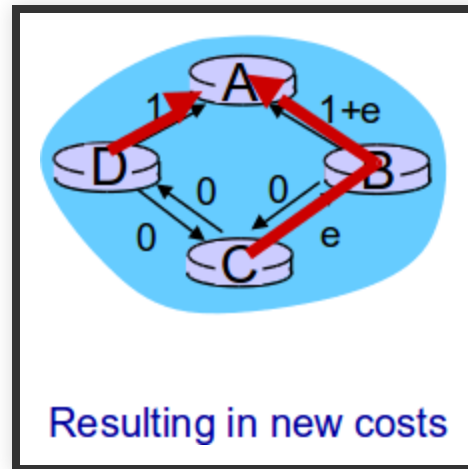
DIJKSTRA'S ALGORITHM, DISCUSSION

- ❗ Oscillations possible: e.g., support link cost equals amount of carried traffic

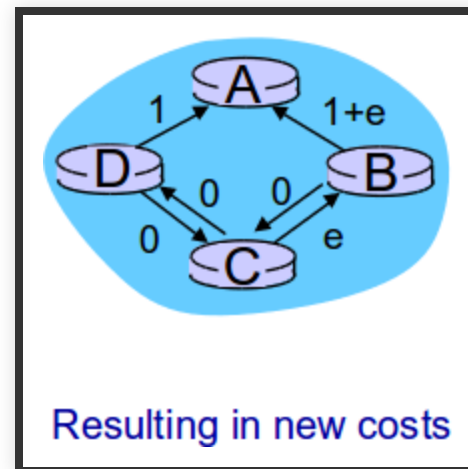
DIJKSTRA'S ALGORITHM, DISCUSSION



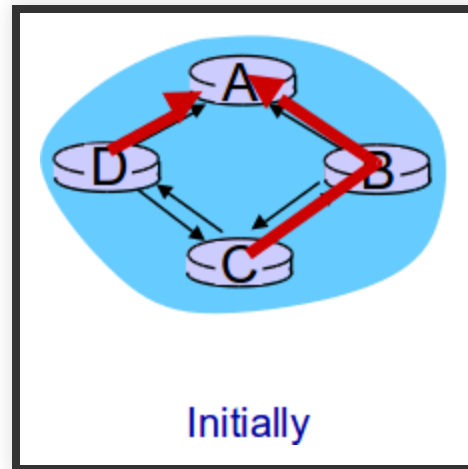
DIJKSTRA'S ALGORITHM, DISCUSSION



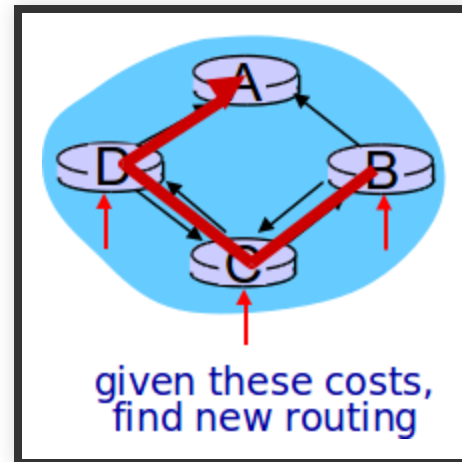
DIJKSTRA'S ALGORITHM, DISCUSSION



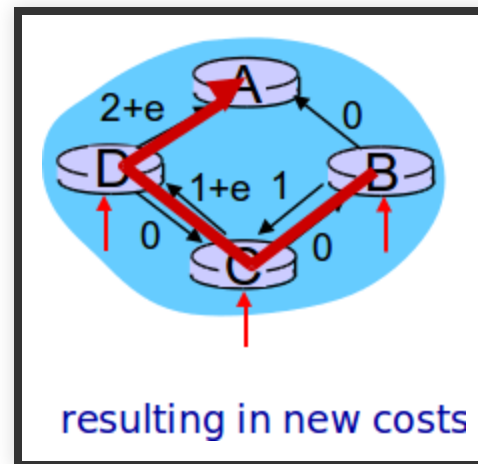
DIJKSTRA'S ALGORITHM, DISCUSSION



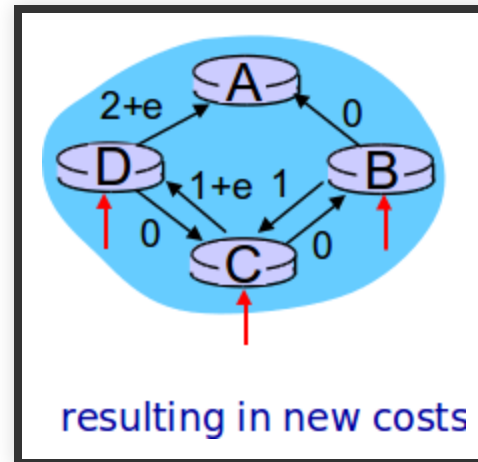
DIJKSTRA'S ALGORITHM, DISCUSSION



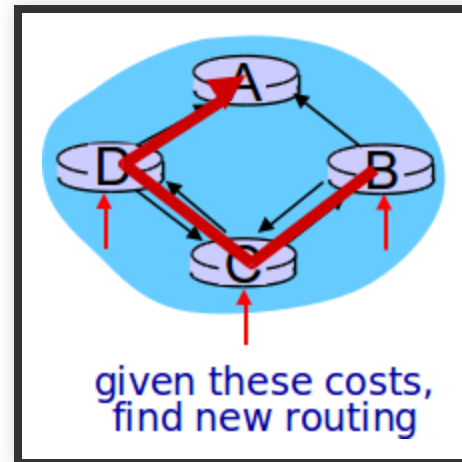
DIJKSTRA'S ALGORITHM, DISCUSSION



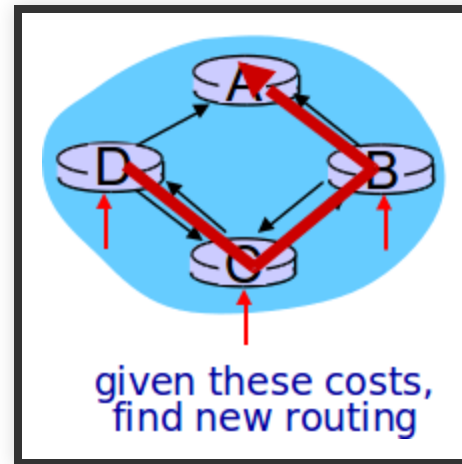
DIJKSTRA'S ALGORITHM, DISCUSSION



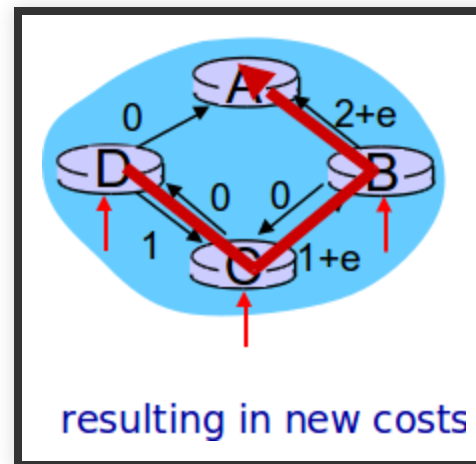
DIJKSTRA'S ALGORITHM, DISCUSSION



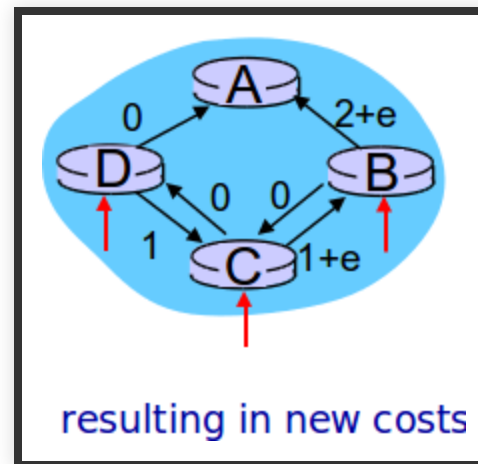
DIJKSTRA'S ALGORITHM, DISCUSSION



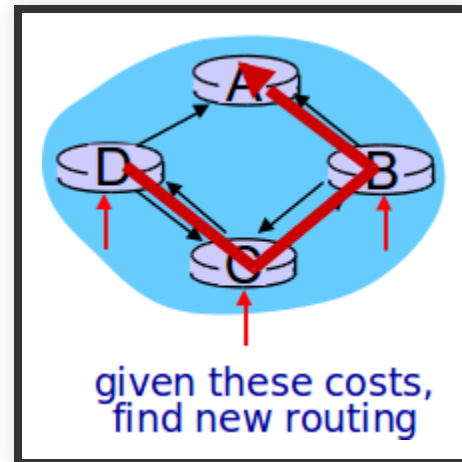
DIJKSTRA'S ALGORITHM, DISCUSSION



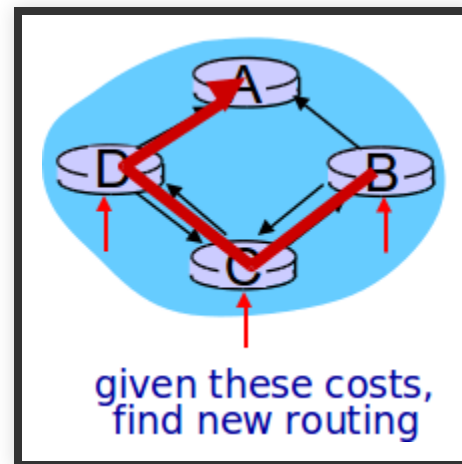
DIJKSTRA'S ALGORITHM, DISCUSSION



DIJKSTRA'S ALGORITHM, DISCUSSION



DIJKSTRA'S ALGORITHM, DISCUSSION

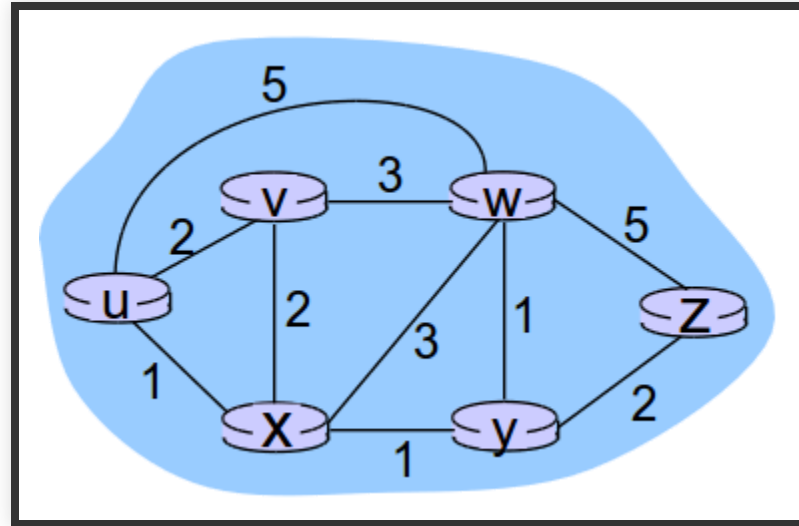


DISTANCE VECTOR ALGORITHM

❗ Bellman-Ford equation (dynamic programming)

- **let:** $d_x(y) :=$ cost of least-cost path from x to y
- **then:** $d_x(y) = \min_v \{ c(x,v) + d_v(y) \}$
- **Where:**
 - \min_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



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

B-F equation says:

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

BELLMAN-FORD EXAMPLE

- 💡 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 $D_x = [D_x(y): y \text{ in } N]$
- node x :
 - knows cost to each neighbor v : $c(x,v)$
 - maintains its neighbors' distance vectors. For each neighbor v , x maintains $D_v = [D_v(y): y \text{ in } N]$

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:

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

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

DISTANCE VECTOR ALGORITHM

Iterative, asynchronous:

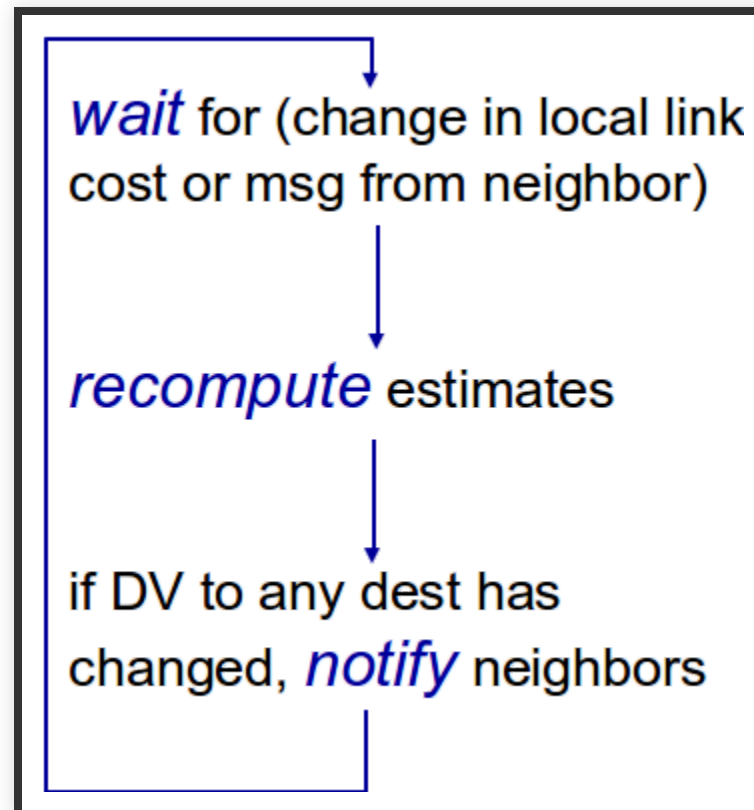
- Each local iteration caused by:
 - local link cost change
 - DV update message from neighbor

Distributed:

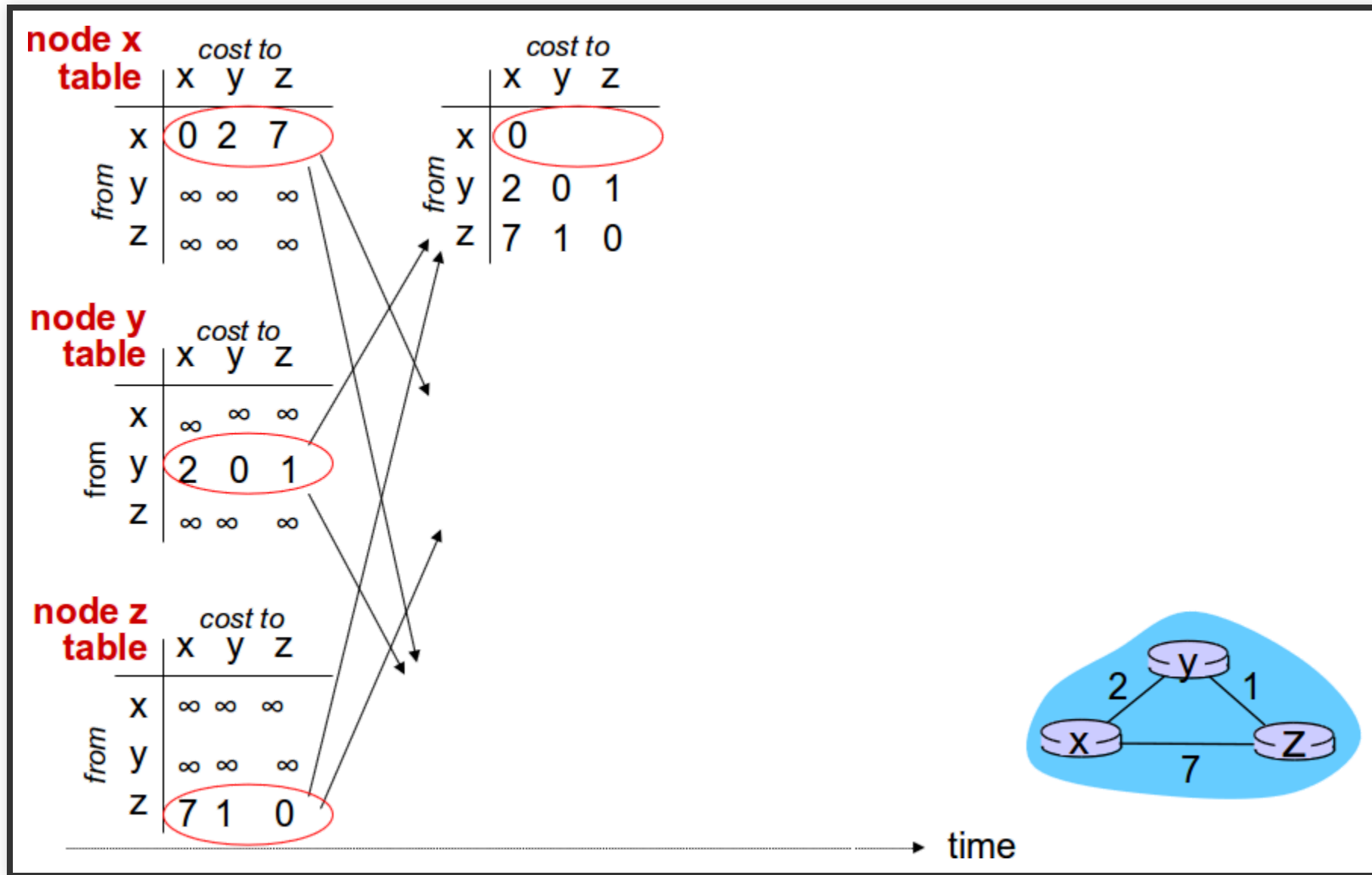
- Each node notifies neighbors only when its DV changes
 - neighbors then notify their neighbors if necessary

DISTANCE VECTOR ALGORITHM

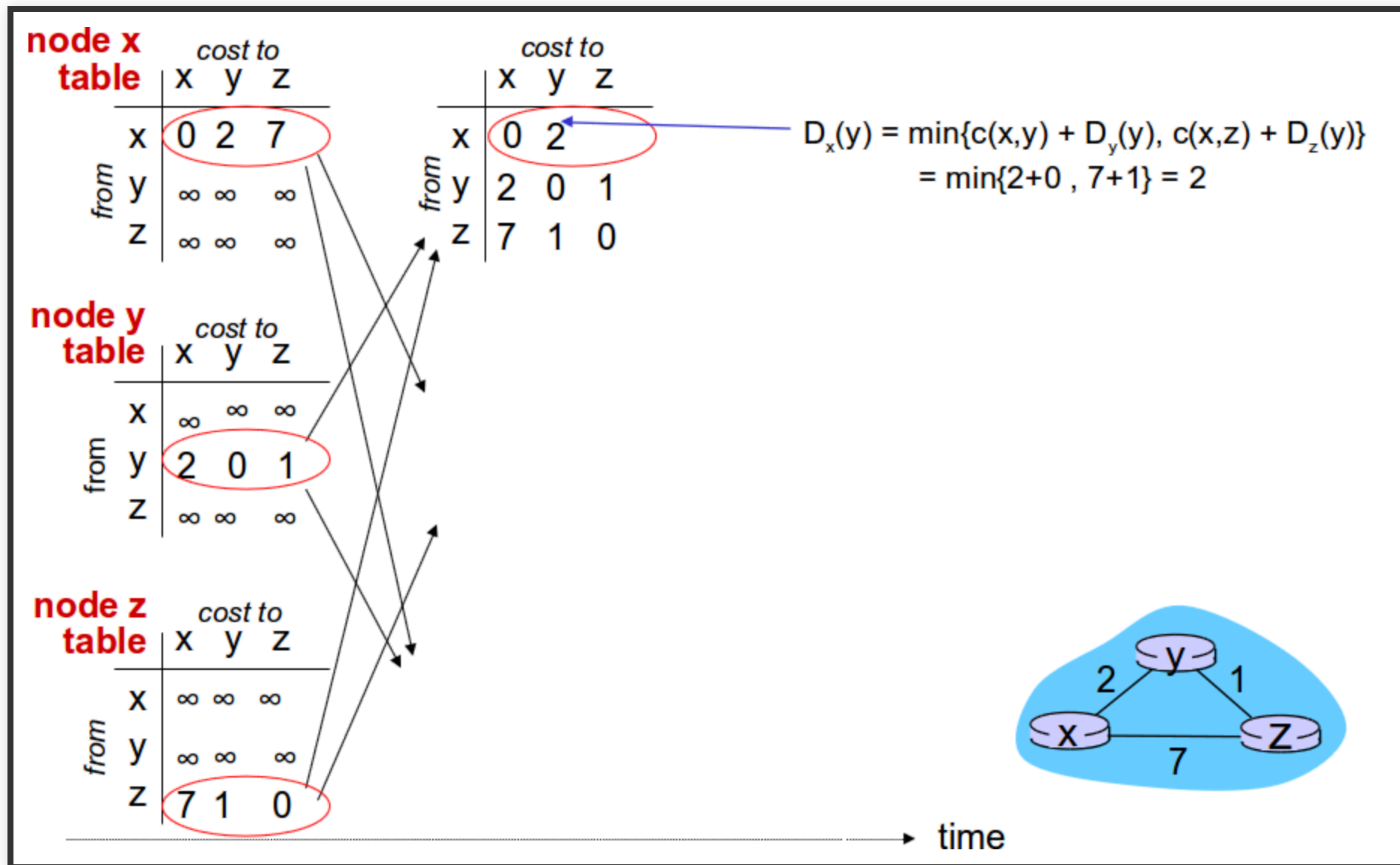
Each node:



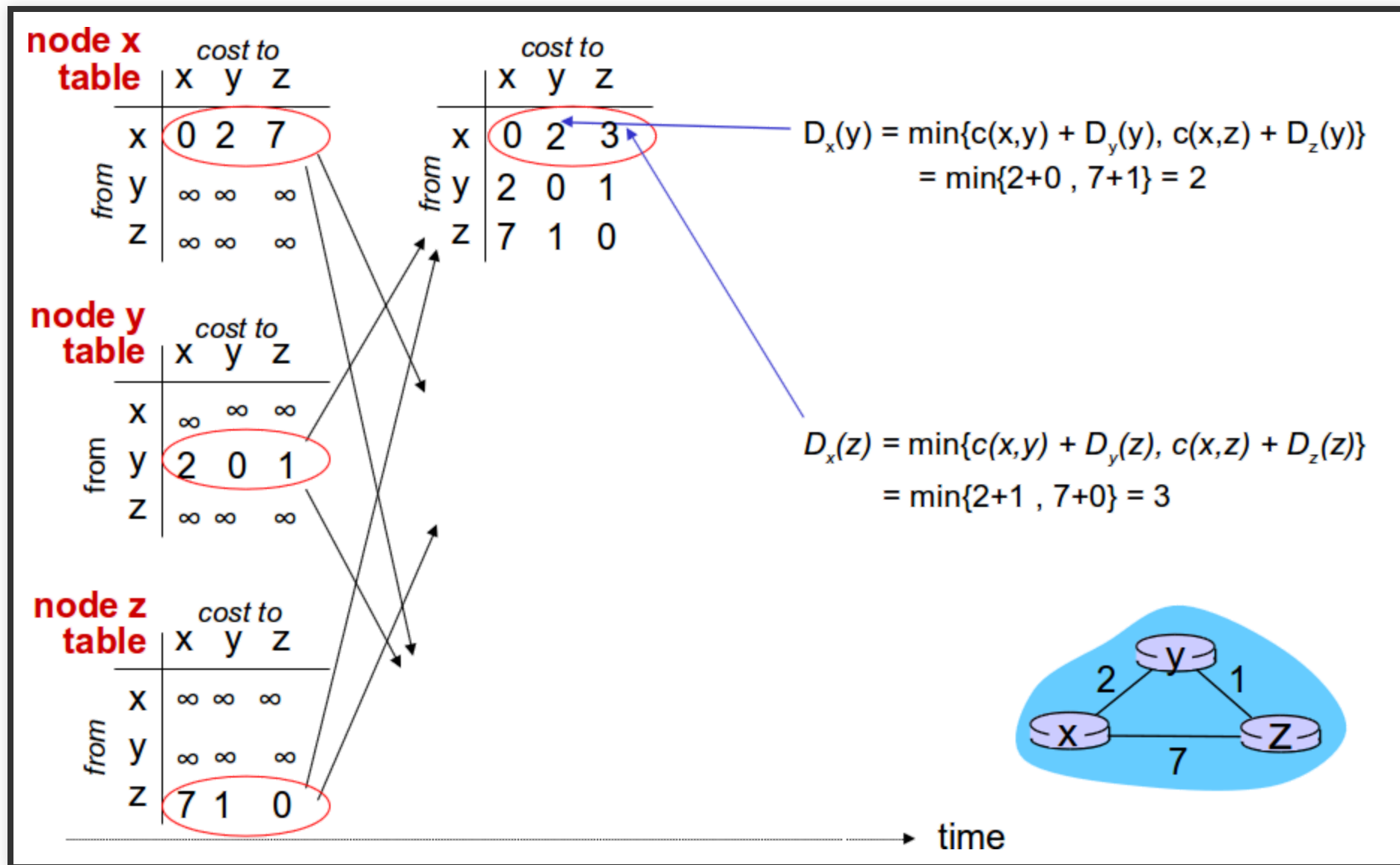
DISTANCE VECTOR ALGORITHM - EXAMPLE



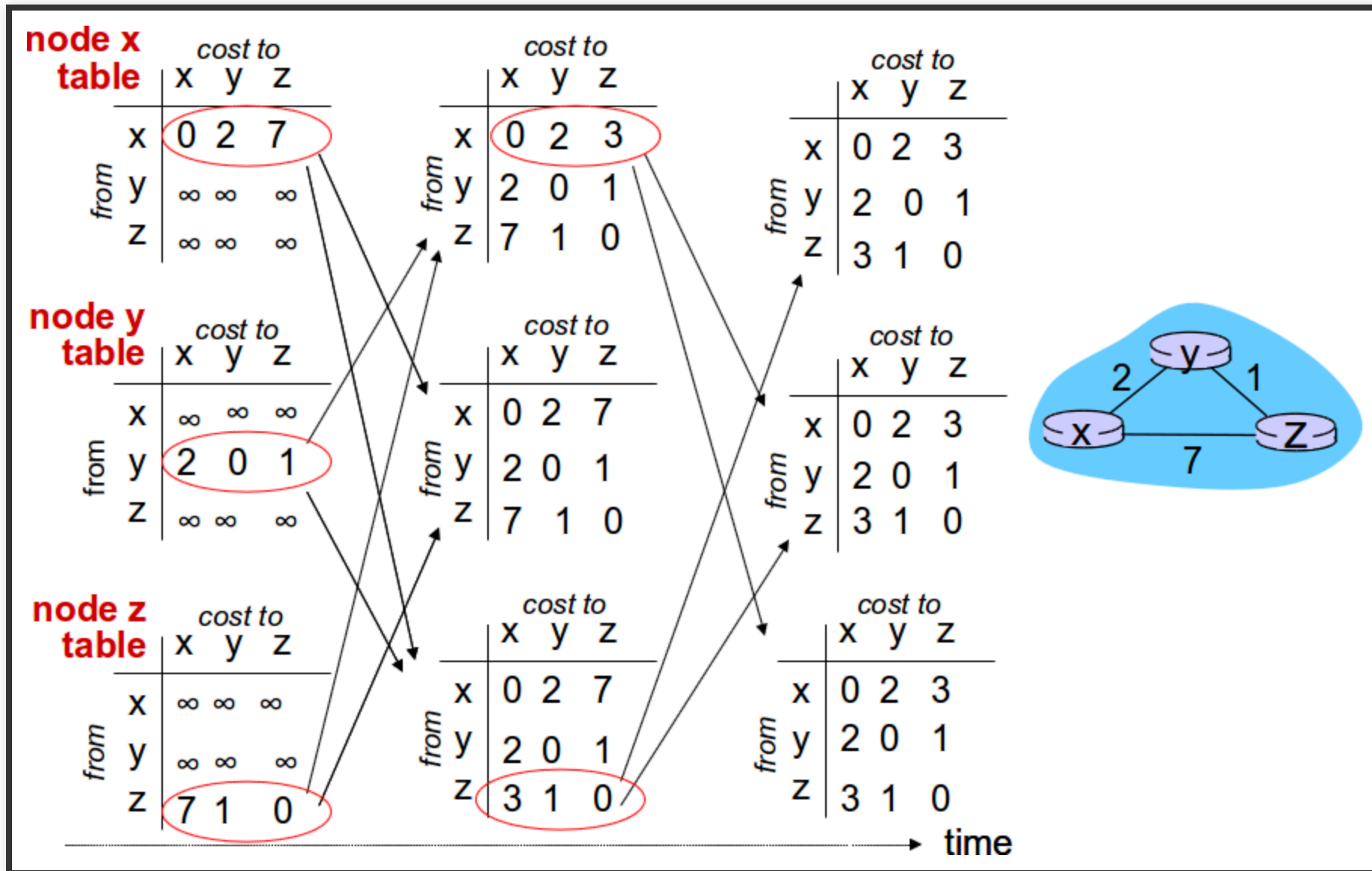
DISTANCE VECTOR ALGORITHM - EXAMPLE



DISTANCE VECTOR ALGORITHM - EXAMPLE



DISTANCE VECTOR ALGORITHM - EXAMPLE

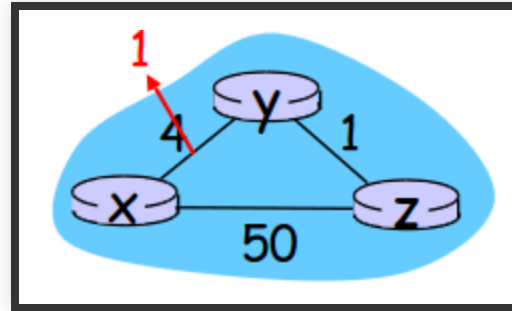


DISTANCE VECTOR: LINK COST CHANGES

Link cost changes:

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

DISTANCE VECTOR: LINK COST CHANGES

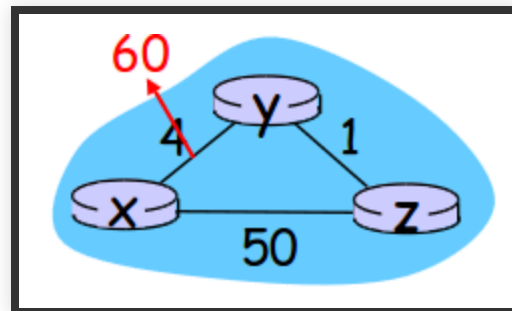


"good news travels fast"

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 textbook



DISTANCE VECTOR: 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

COMPARISON OF LS AND DV ALGORITHMS

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

COMPARISON OF LS AND DV ALGORITHMS

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 through network

ROUTING IN THE INTERNET

ROUTING SO FAR

- our routing study thus far - idealization
- all routers identical
- network “flat”
- ... not true in practice

HIERARCHICAL ROUTING

Scale: with 600 million destinations:

- can't store all dest's 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, “autonomous systems” (AS)

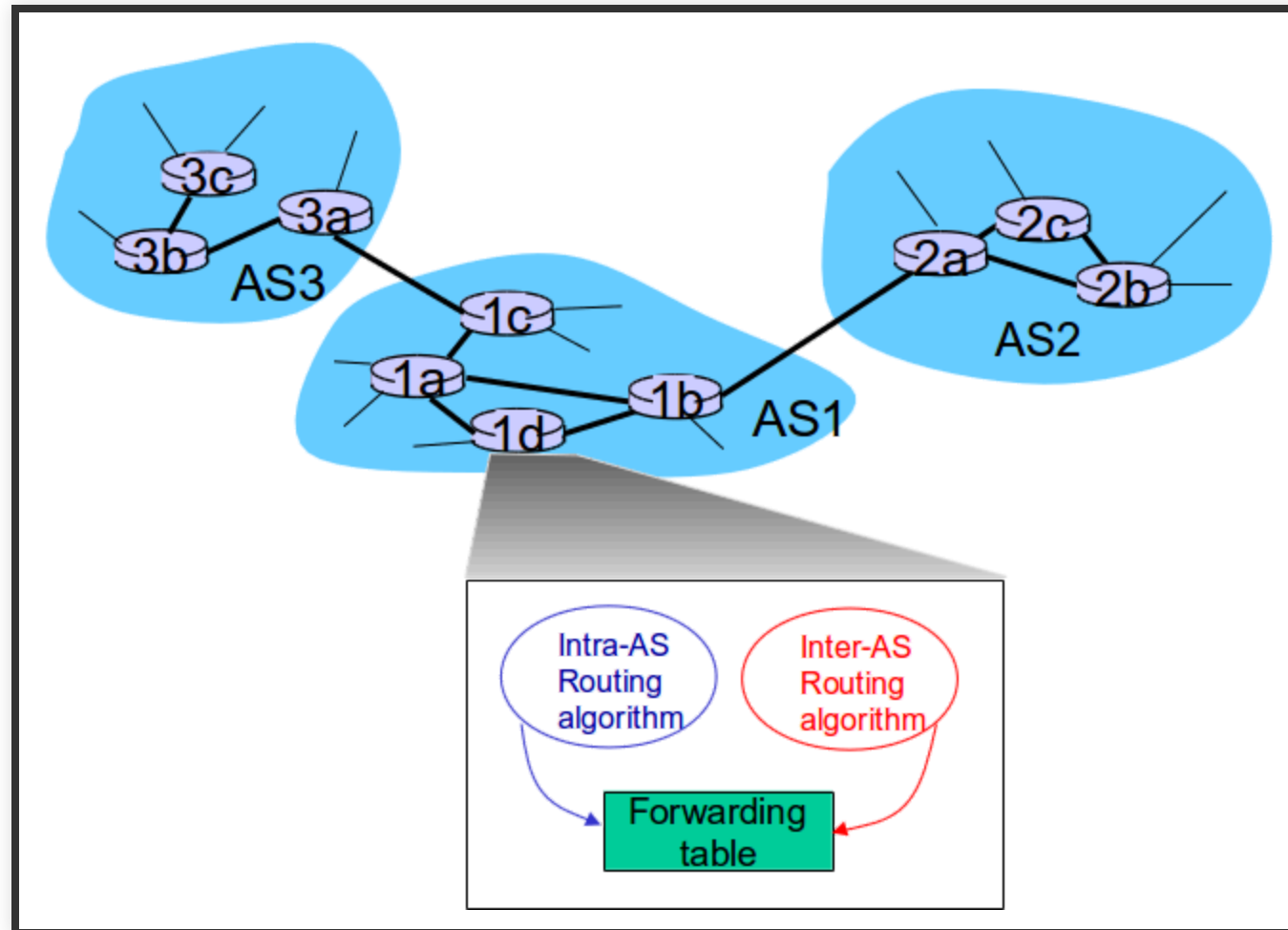
INTRA-AS ROUTING

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

INTER-AS ROUTING

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

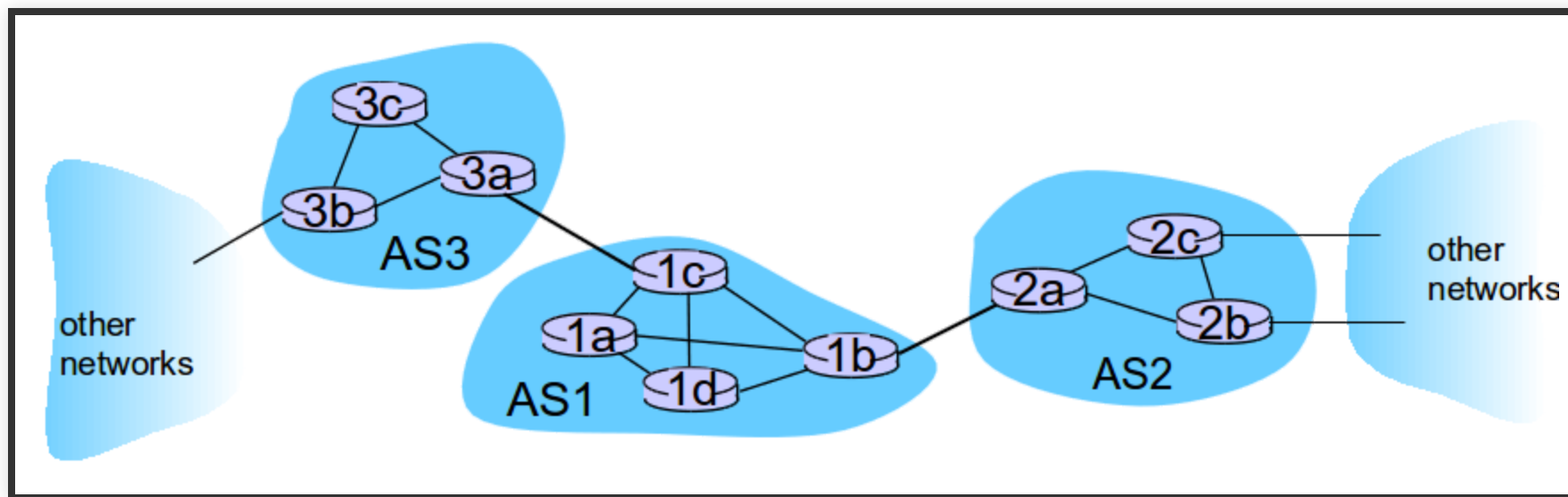
INTERCONNECTED AS'ES



INTERCONNECTED AS'ES

- Forwarding table configured by both intra- and inter-AS routing algorithm
 - intra-AS sets entries for internal dests
 - inter-AS and intra-AS sets entries for external dests

INTER-AS TASKS



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:

- learn which dests are reachable through AS2, which through AS3
- propagate this reachability info to all routers in AS1

Job of inter-AS routing!

INTRA-AS ROUTING

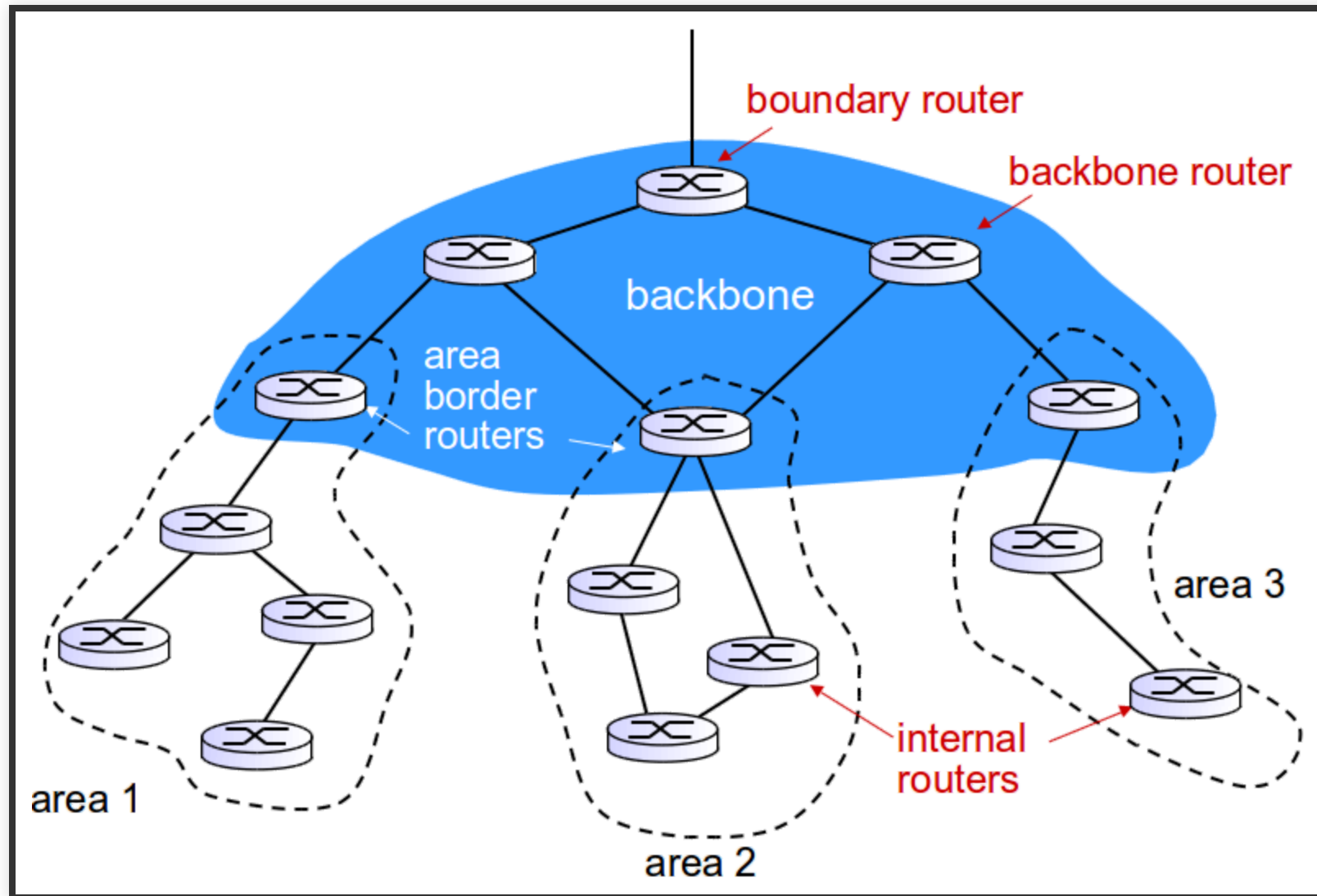
- also known as interior gateway protocols (IGP)
- most common intra-AS routing protocols:
 - **RIP:** Routing Information Protocol
 - **OSPF:** Open Shortest Path First
 - **IGRP:** Interior Gateway Routing Protocol (Cisco proprietary)

OSPF (OPEN SHORTEST PATH FIRST)

OSPF “ADVANCED” FEATURES

- **Security:** all OSPF messages authenticated (to prevent malicious intrusion)
- **Multiple** same-cost paths allowed (only one path in RIP)
- Integrated uni- and multicast support:
 - Multicast OSPF (MOSPF) uses same topology data base as OSPF
- hierarchical OSPF in large domains.

HIERARCHICAL OSPF



HIERARCHICAL OSPF

- **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's.

ROUTING AMONG THE ISPS: BGP

INTERNET INTER-AS ROUTING: BGP

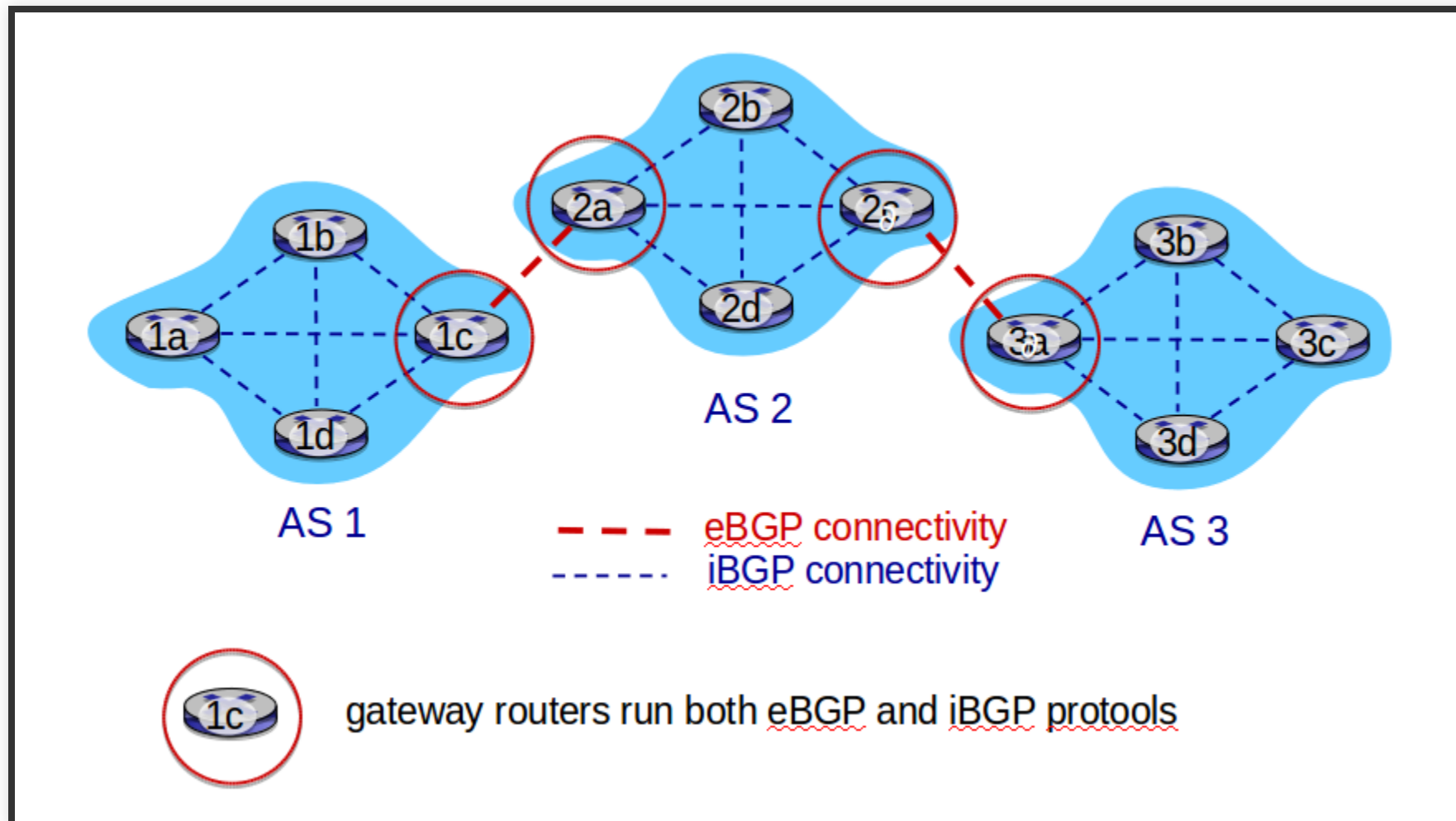
- **BGP (Border Gateway Protocol):** the de facto inter-domain routing protocol

 “glue that holds the Internet together”

INTERNET INTER-AS ROUTING: BGP

- BGP provides each AS a means to:
 - **eBGP**: obtain subnet reachability information from neighboring ASs.
 - **iBGP**: propagate reachability information to all AS-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**”

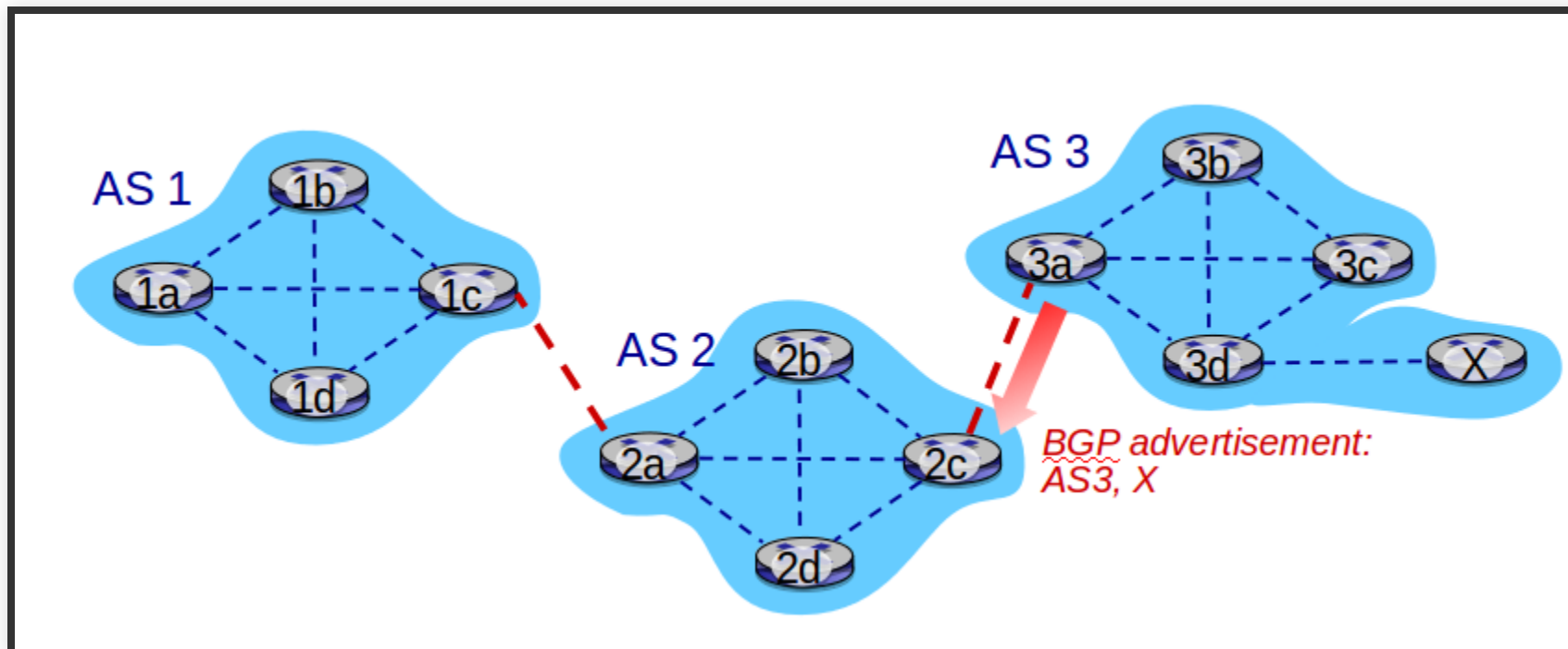
EBGP, IBGP CONNECTIONS



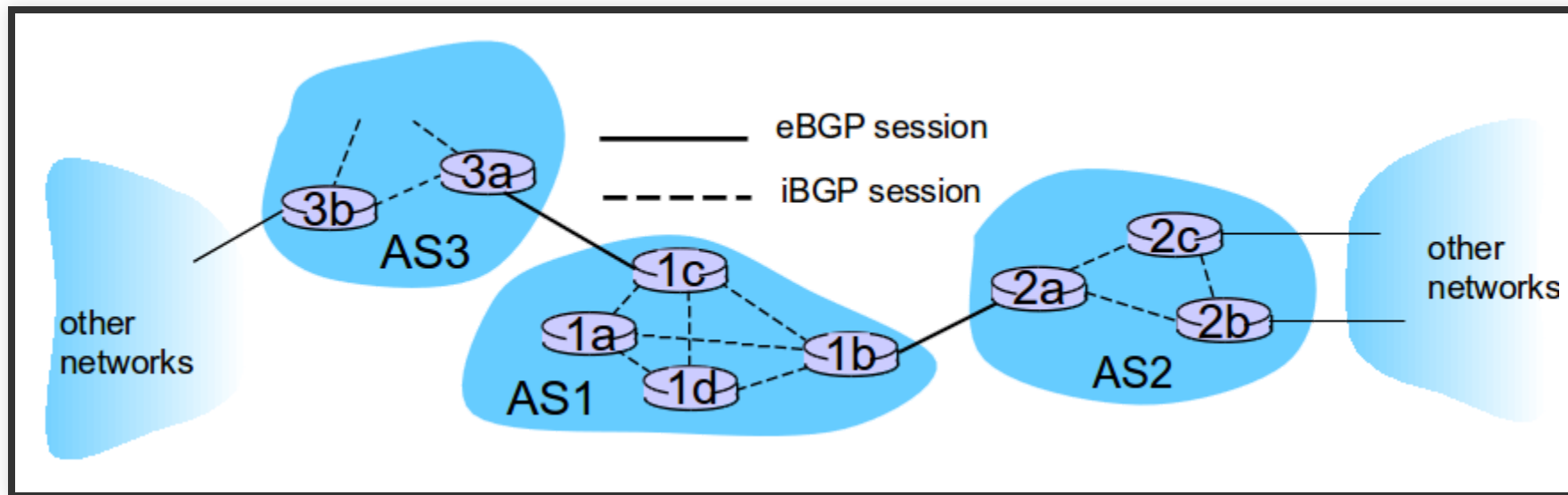
BGP BASICS

- **BGP session:** two BGP routers (“peers”) exchange BGP messages:
 - advertising **paths** to different destination network prefixes (“path vector” protocol)
 - exchanged over semi-permanent TCP connections
- when AS3 advertises a prefix to AS1:
 - AS3 **promises** it will forward datagrams towards that prefix
 - AS3 can aggregate prefixes in its advertisement

BGP BASICS



DISTRIBUTING PATH INFORMATION



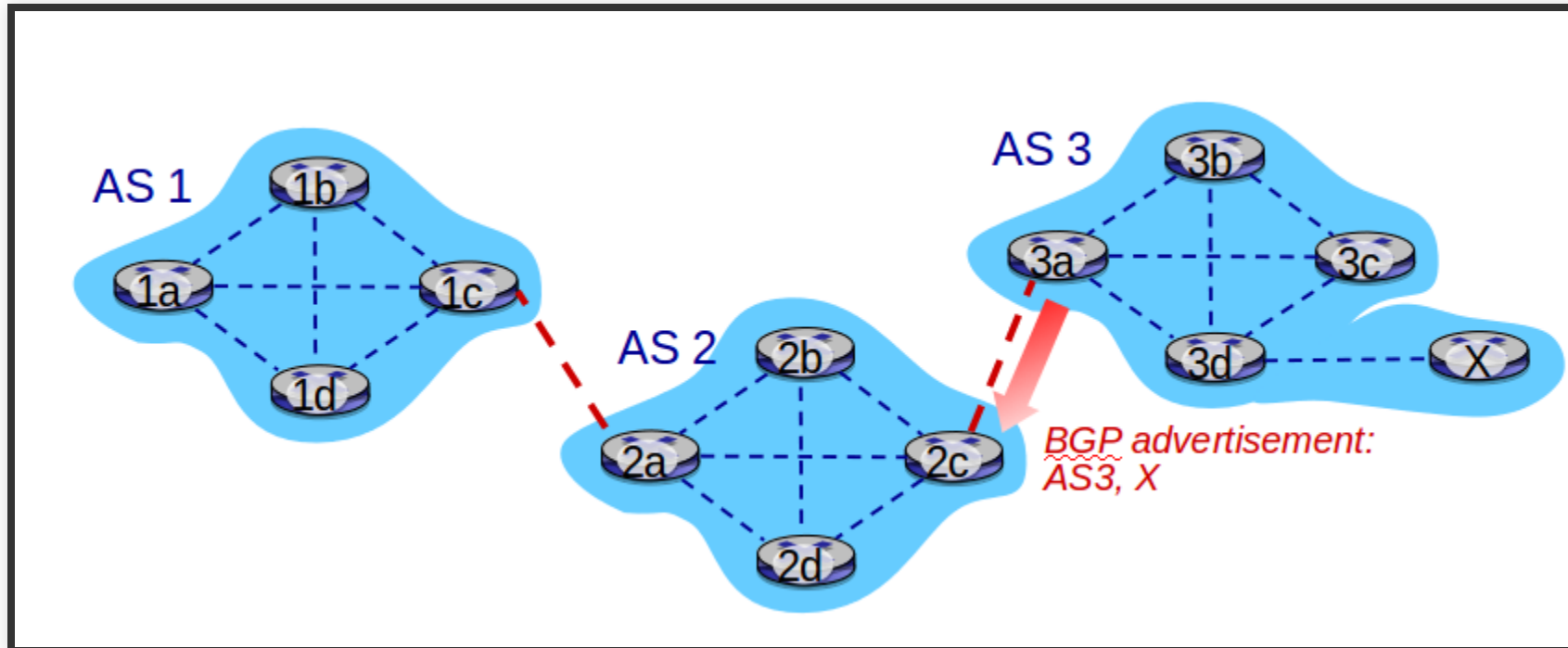
- advertised prefix includes BGP attributes
 - prefix + attributes = “route”

BGP ATTRIBUTES

Important attributes:

- **AS-PATH:** contains ASs through which prefix advertisement has passed: e.g., AS 67, AS 17
- **NEXT-HOP:** indicates specific internal-AS router to next-hop AS - IP address. (may be multiple links from current AS to next-hop-AS)
- **LOCAL_PREF:** indicates policy by administrator. 100 is default, higher priority number wins.

BGP ATTRIBUTES



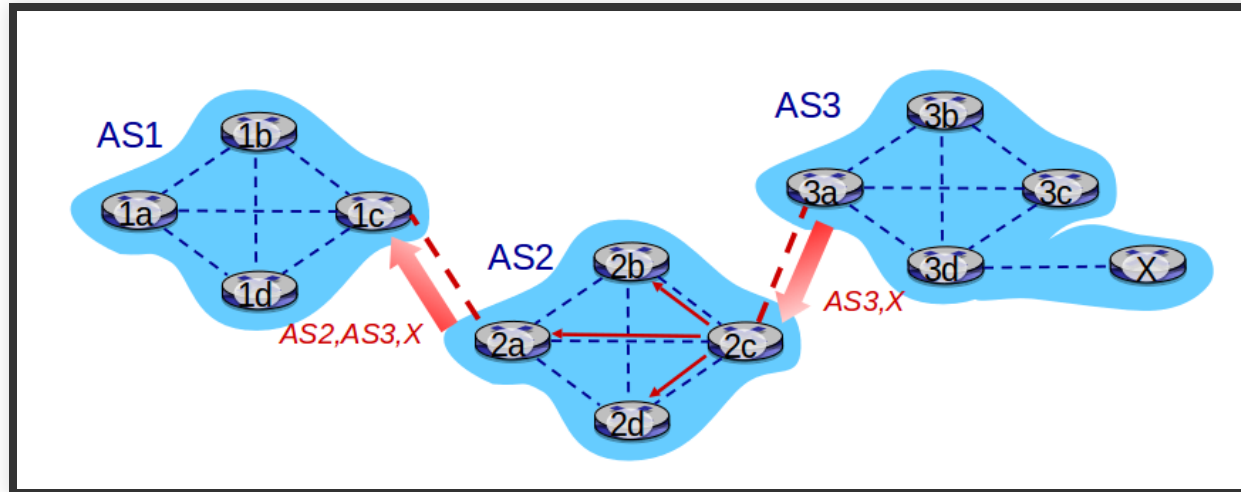
Internal routers in AS2 would get this info:

```
AS-PATH: AS3, AS2, x  
NEXT-HOP: 2c ;  
LOCAL_PREF: 100
```

PATH ATTRIBUTES AND BGP ROUTES

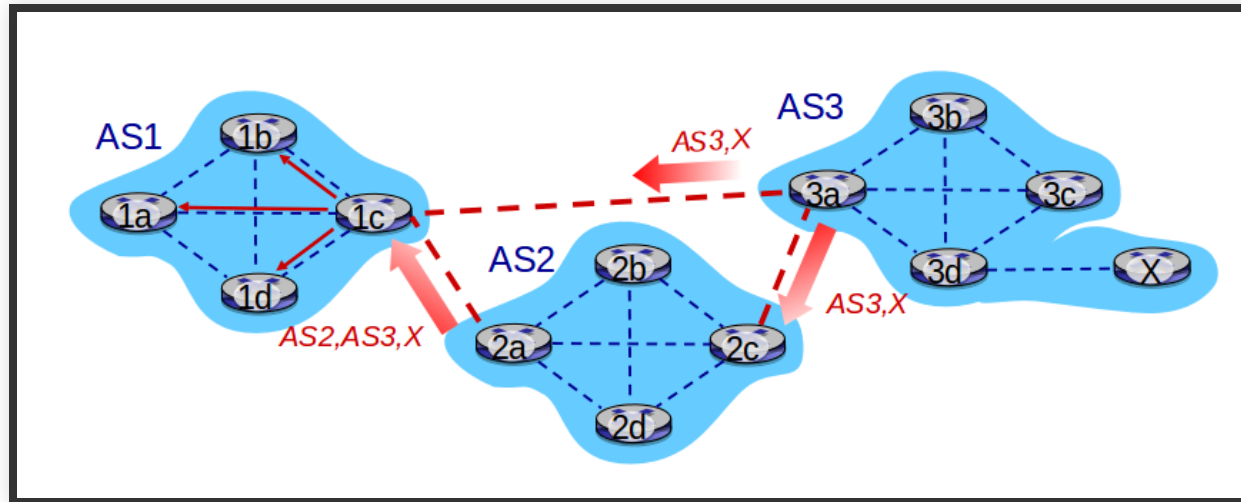
- gateway router receiving route advertisement uses import policy to accept/decline
 - e.g., never route through AS x
 - **policy-based routing**

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



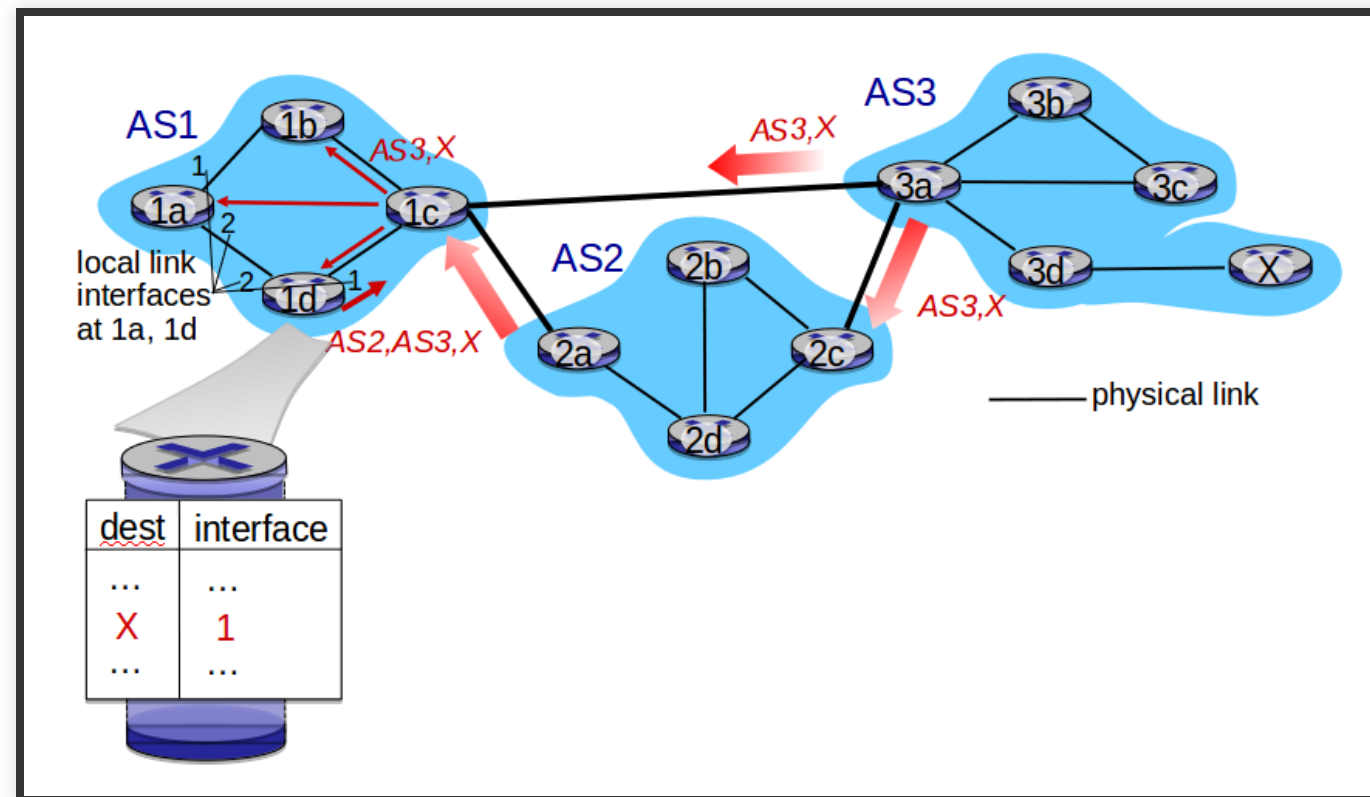
- Gateway router may learn about multiple paths to destination:
 - AS1 gateway router 1c learns path AS2,AS3,X from 2a
 - AS1 gateway router 1c learns path AS3,X from 3a
 - Based on policy, AS1 gateway router 1c chooses path AS3,X, and advertises path within AS1 via iBGP

BGP MESSAGES

- BGP messages exchanged between peers over TCP connection
- BGP messages:
 - **[OPEN:]** opens TCP connection to peer and authenticates sender
 - **[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?



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 ROUTE SELECTION ALGORITHM

Router 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

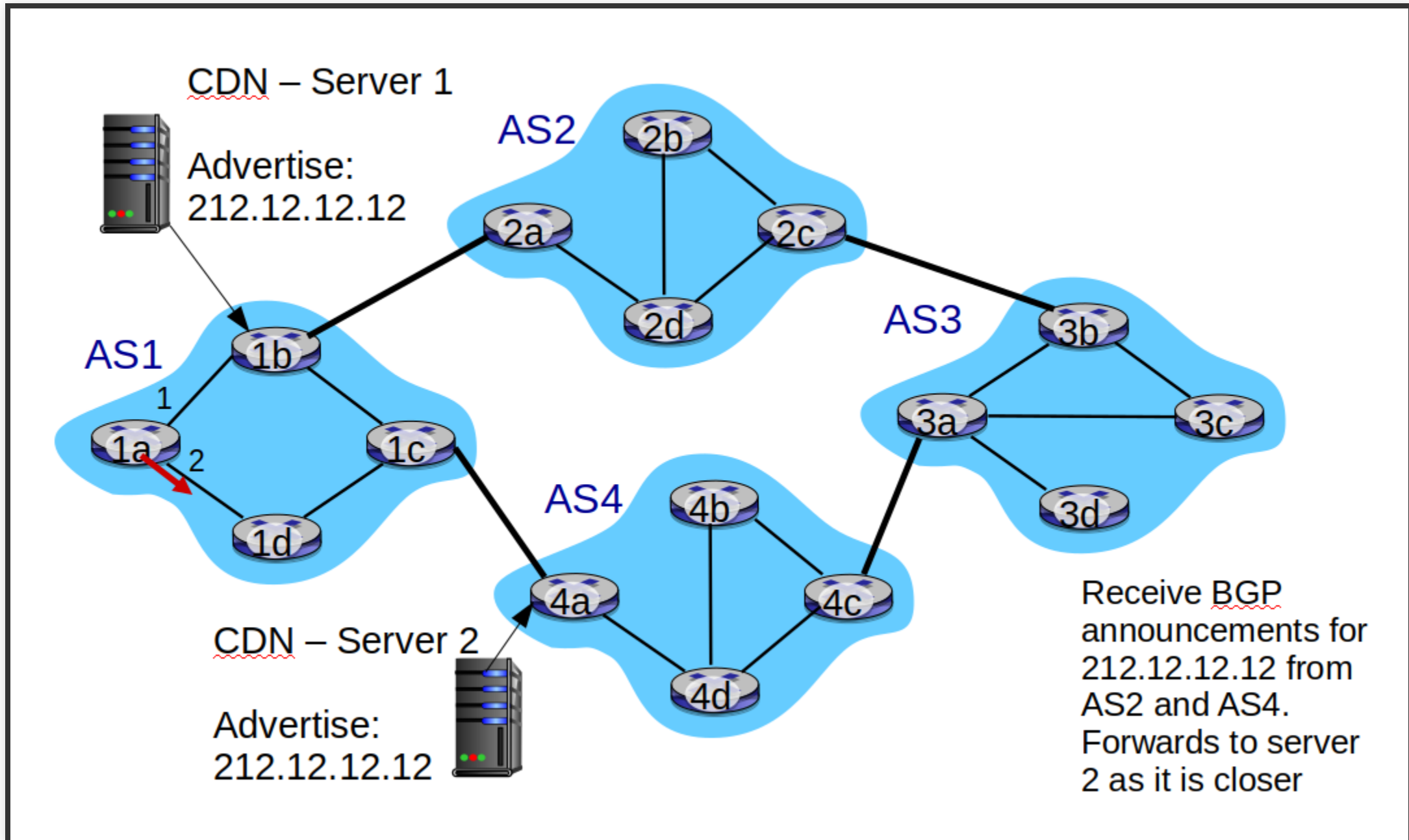
IP-ANYCAST

Consider

1. Applications having replicated content on different servers in different geographical locations
2. Have each user access the closest server

IP-Anycast service → Often used in DNS → BGP's route selection-algorithm

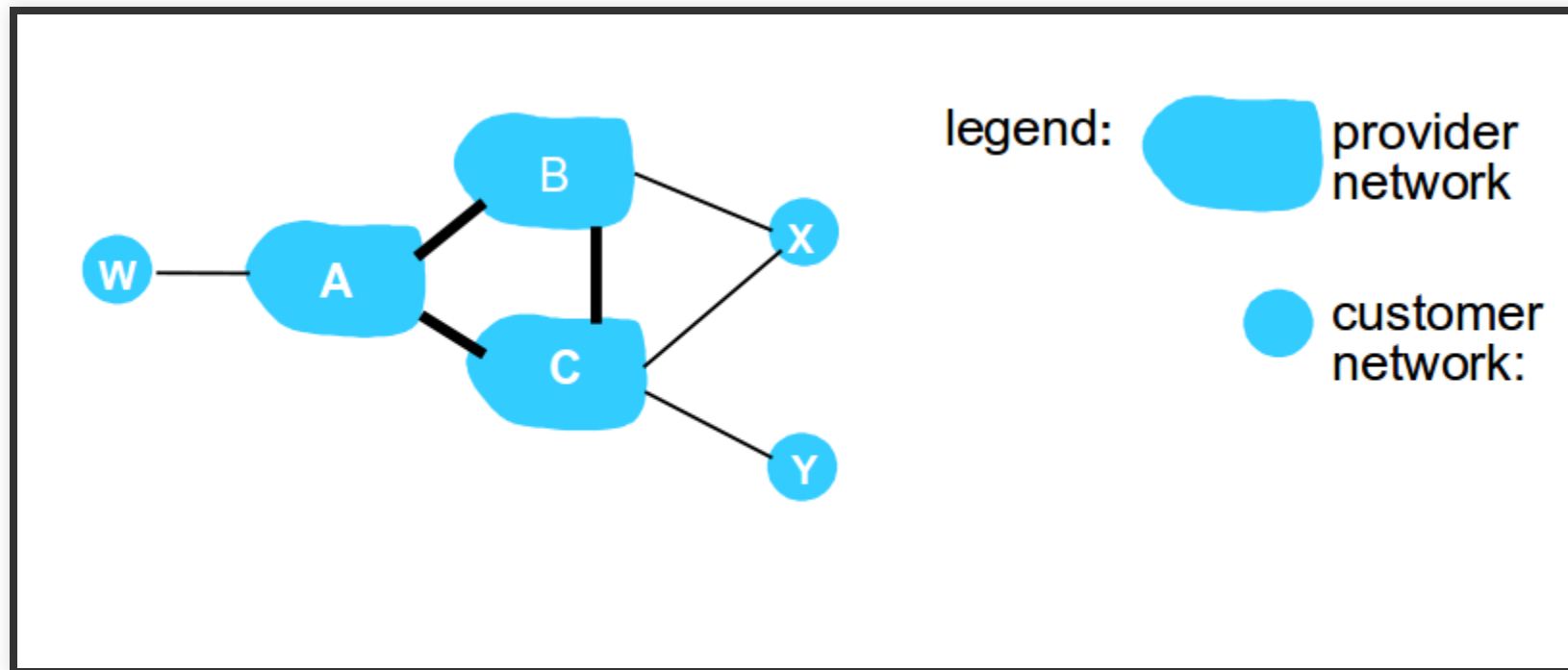
IP-ANYCAST



IP-ANYCAST - CAVEAT

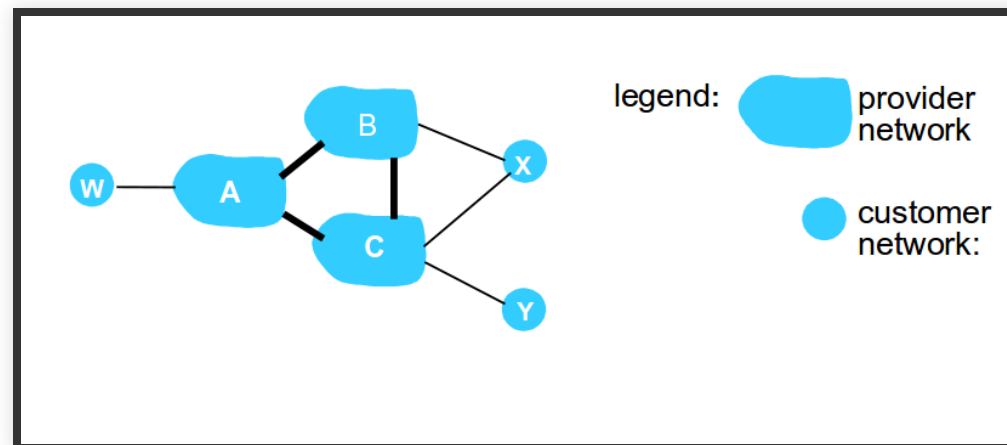
- If used by CDN: different datagrams might end up at different locations.
- Extensively used in DNS to locate root DNS server
 - Recall: 13 different IP addresses for Root DNS servers → each with scattered set of servers
 - IP-Anycast is used to route to the closest

BGP ROUTING POLICY



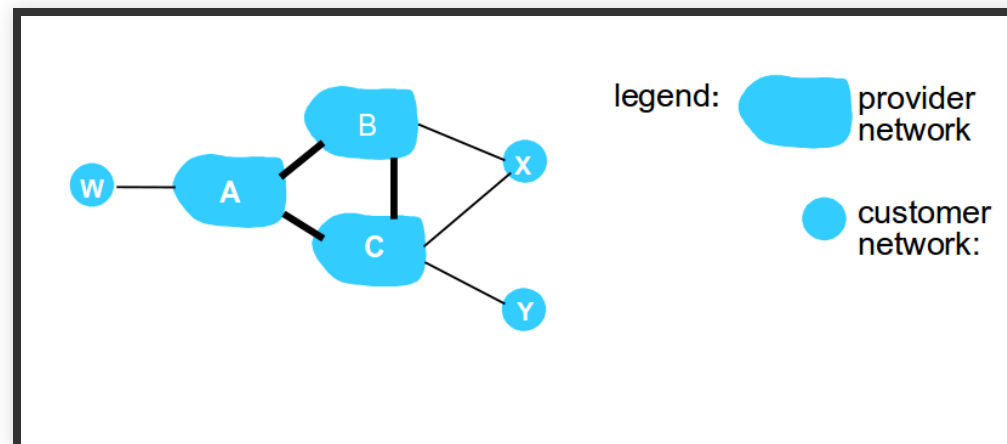
BGP ROUTING POLICY

- 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 ROUTING POLICY

- A advertises path AW to B and to C
- **B Chooses not to advertise** path 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



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

WHY DIFFERENT INTRA-, INTER-AS ROUTING?

! Scale

- hierarchical routing saves table size, reduced update traffic

WHY DIFFERENT INTRA-, INTER-AS ROUTING?

! Performance

- intra-AS: can focus on performance
- inter-AS: policy may dominate over performance

OBTAINING INTERNET PRESENCE

THE PROBLEM

💡 You have made a new company - and wish to have an internet presence! How is this obtained?

- Website with product presentation
- Email for your employees
- DNS server for

This includes not just BGP, but also IP Addressing, DNS and more

BRINGING PIECES TOGETHER

Internet connectivity:

- Contact local ISP
- Buy internet via DSL, cable, fiber etc.
- Also receive an IP range fx. a /24 address range (256 addresses)

Assign IP address to:

- Webserver (or loadbalancer and webserver)
- Mail server
- DNS server

BRINGING PIECES TOGETHER

Domain name:

- Contact an internet registrar to obtain a domain name for your company: *"Nerdy-stuff-are-us.com"*
- Provide the registrar with the IP address for your DNS server (authoritative)
- Now registered in the top level domain server

BRINGING PIECES TOGETHER

When a customer now knows your IP address, how will the internet know where to locate your server?

I.e. how to deliver the TCP-SYN datagram?

BRINGING PIECES TOGETHER

Router will look in forwarding table for an output direction your companys /24 address range

Each router should know where it is located!

💡 BGP - your local ISP will use BGP to announce the presence of your subnet to the ISPs it connects to, who in turn will use BGP to propagate this information.

THE SDN CONTROL PLANE

SOFTWARE DEFINED NETWORKING (SDN)

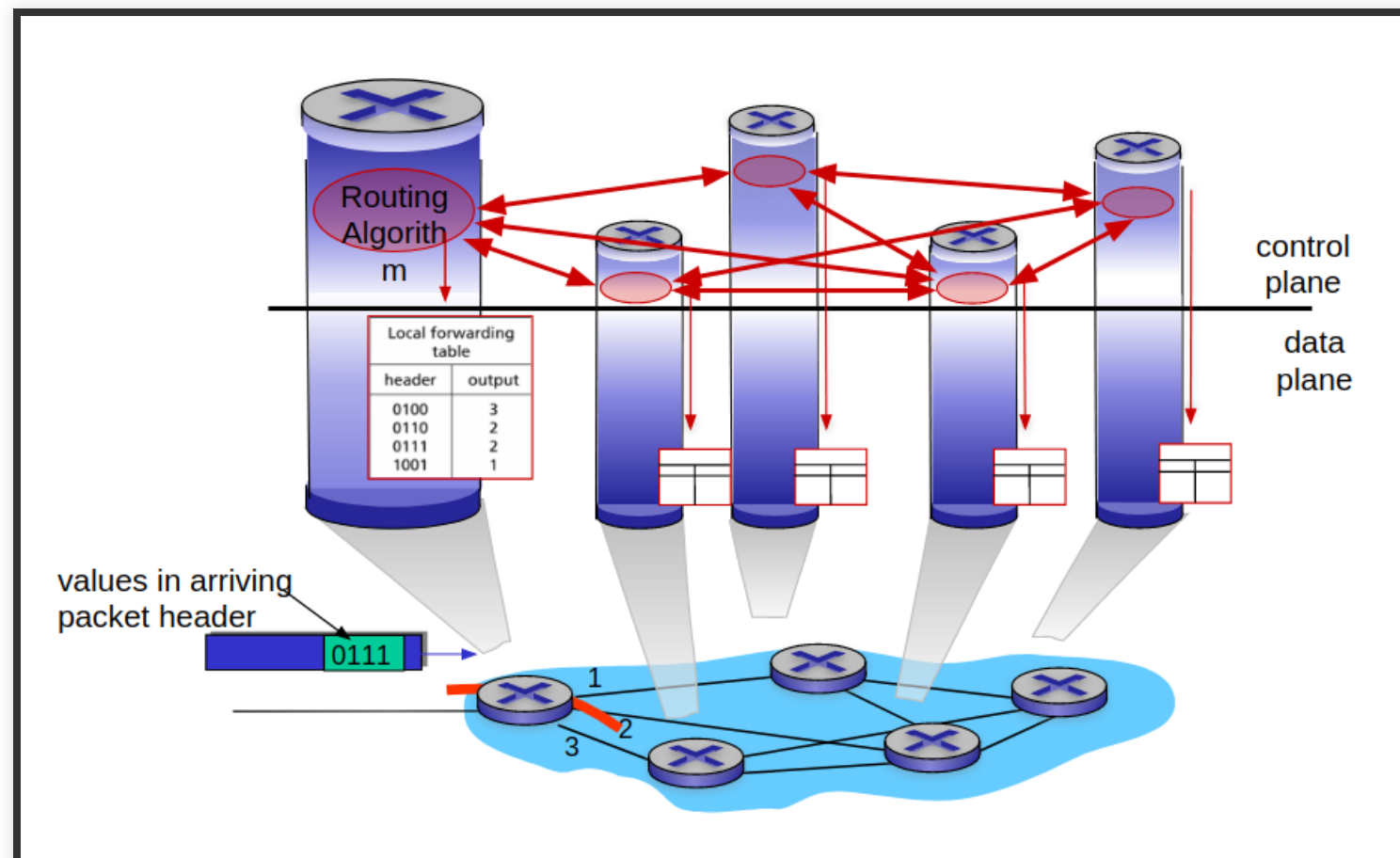
Internet network layer: historically has been implemented via distributed, per-router approach

- 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

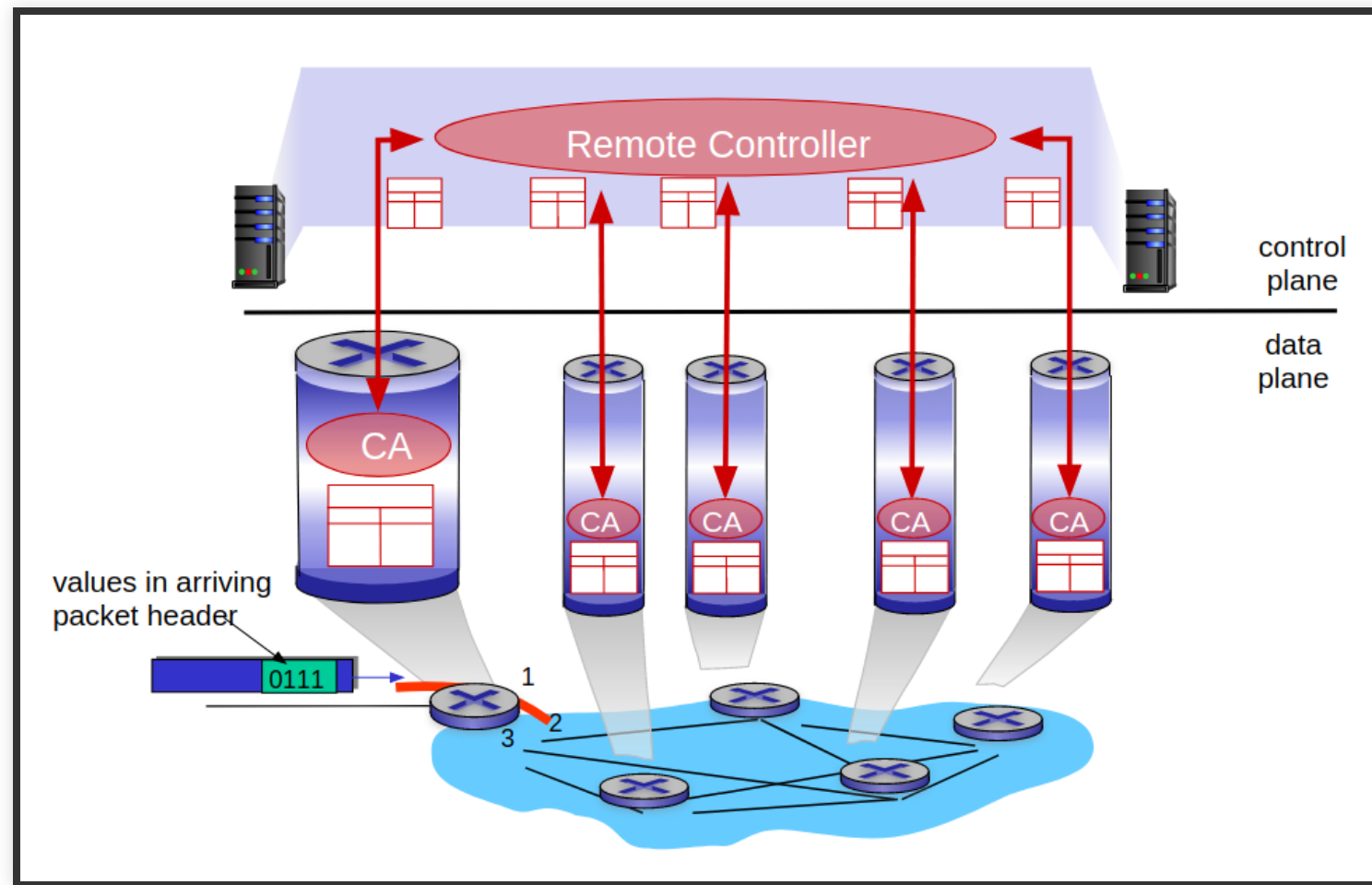
RECALL: PER-ROUTER CONTROL PLANE

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

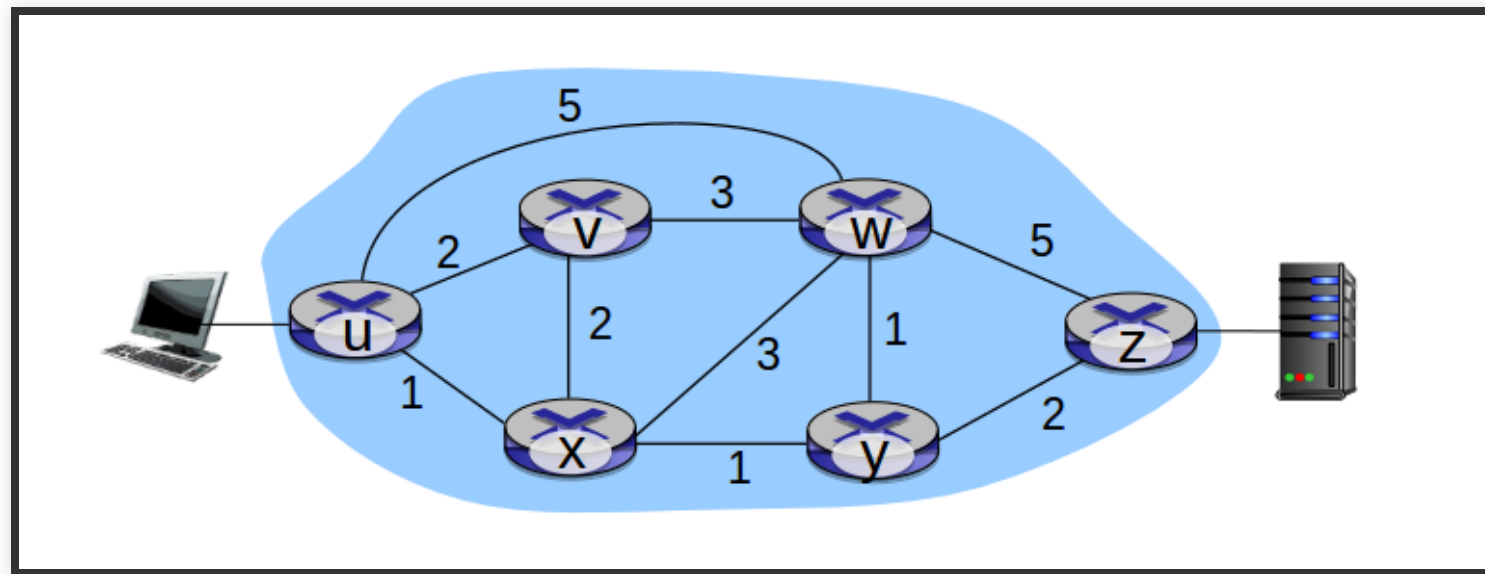
A distinct (typically remote) controller interacts with local control agents (CAs) in routers to compute forwarding tables



SOFTWARE DEFINED NETWORKING (SDN)

Why a logically centralized control plane?

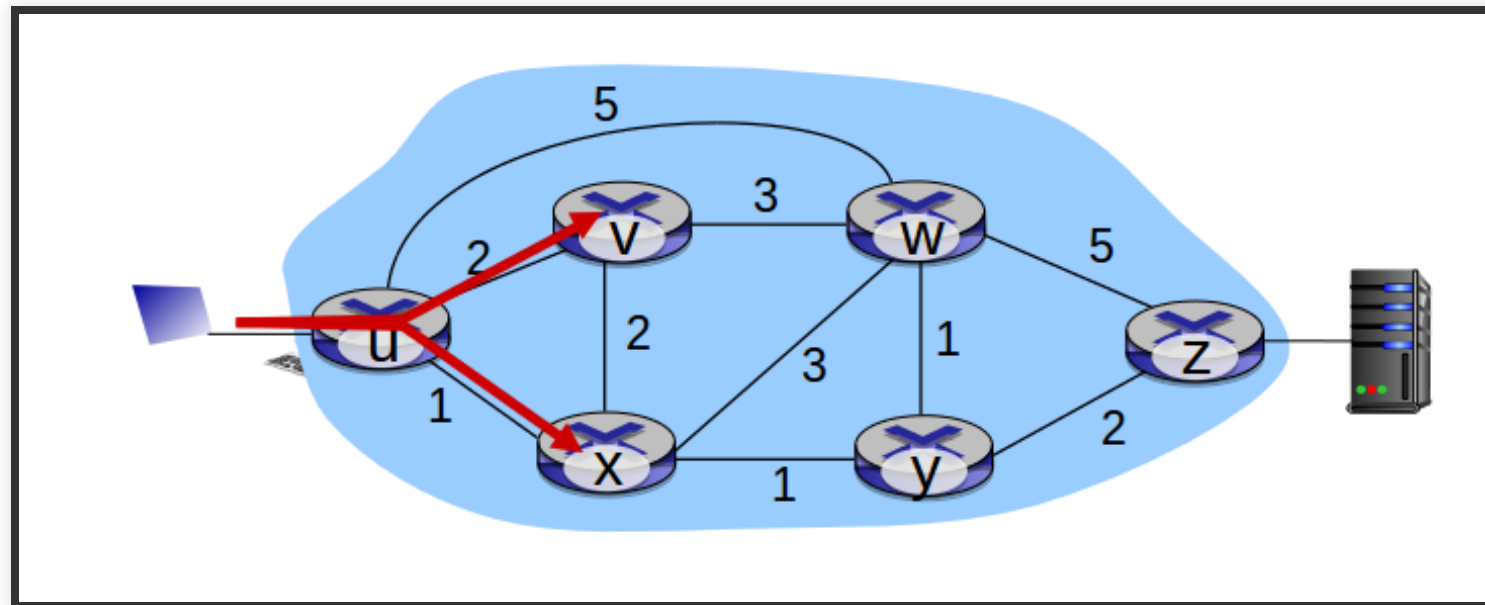
TRAFFIC ENGINEERING: DIFFICULT TRADITIONAL ROUTING



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)!

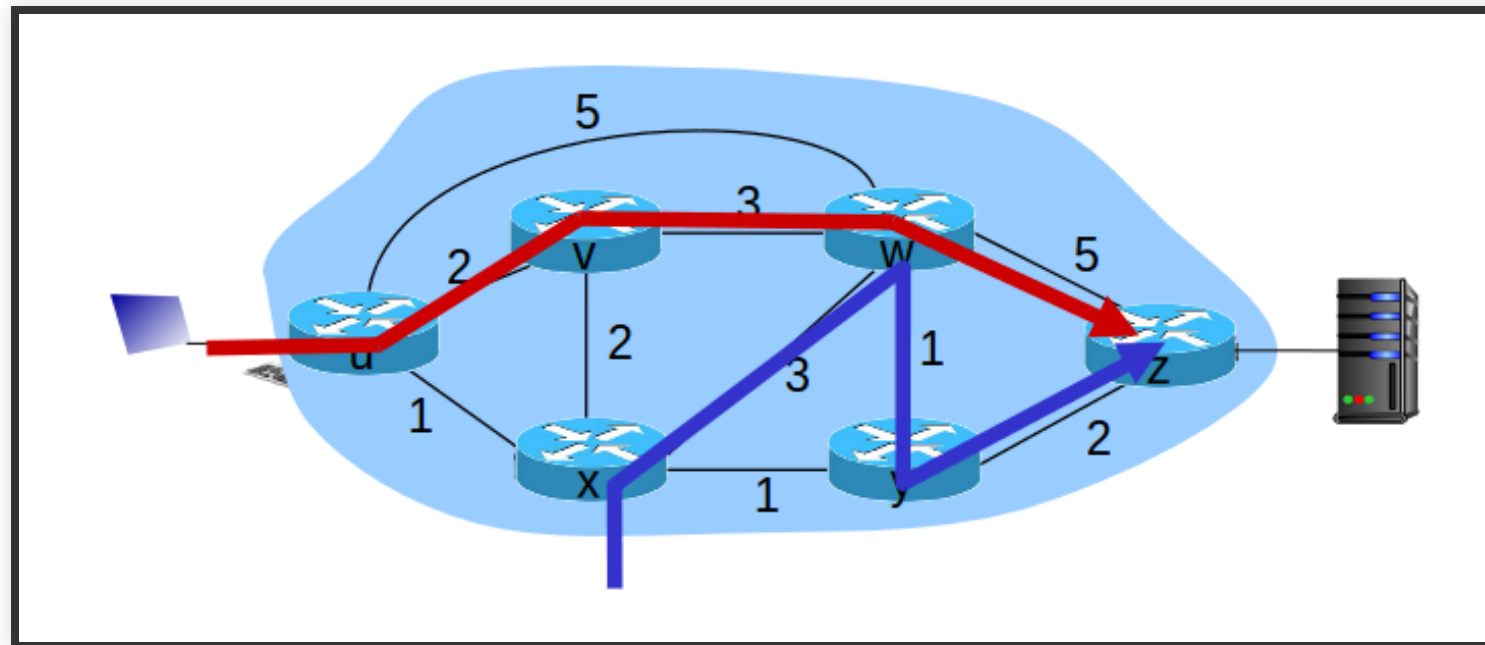
TRAFFIC ENGINEERING: DIFFICULT TRADITIONAL ROUTING



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)

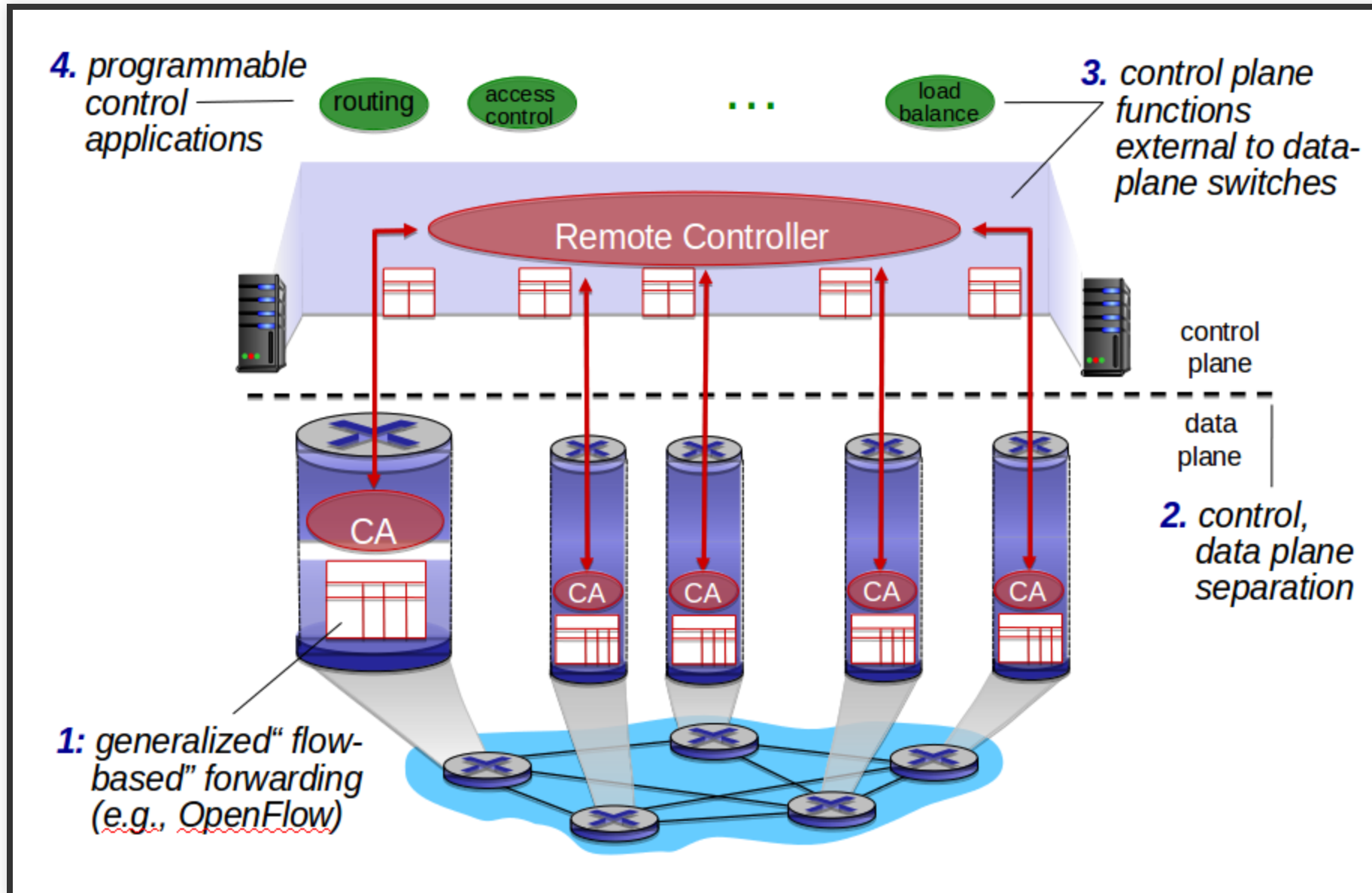
TRAFFIC ENGINEERING: DIFFICULT TRADITIONAL ROUTING



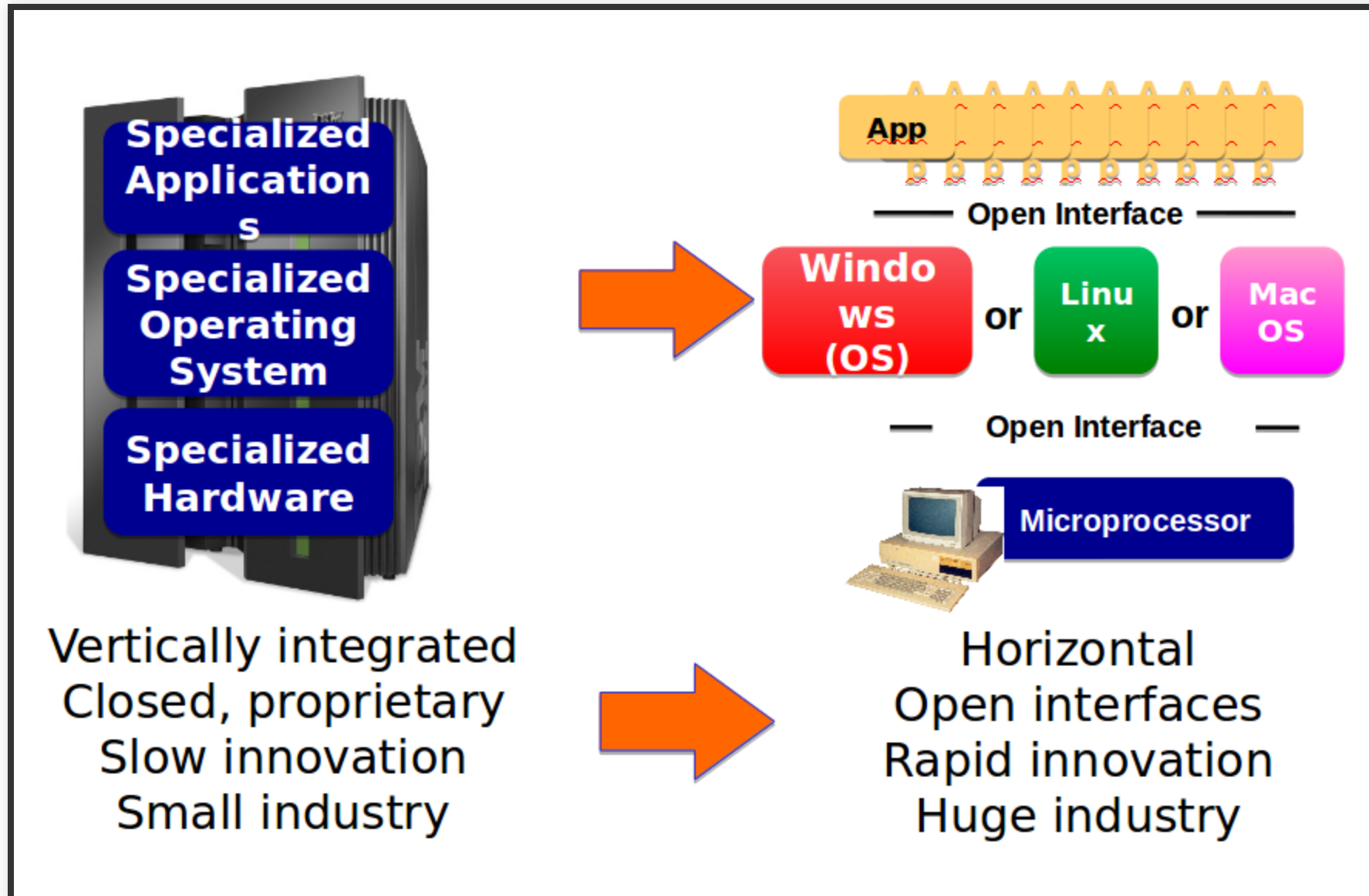
Q: what if w wants to route blue and red traffic differently?

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

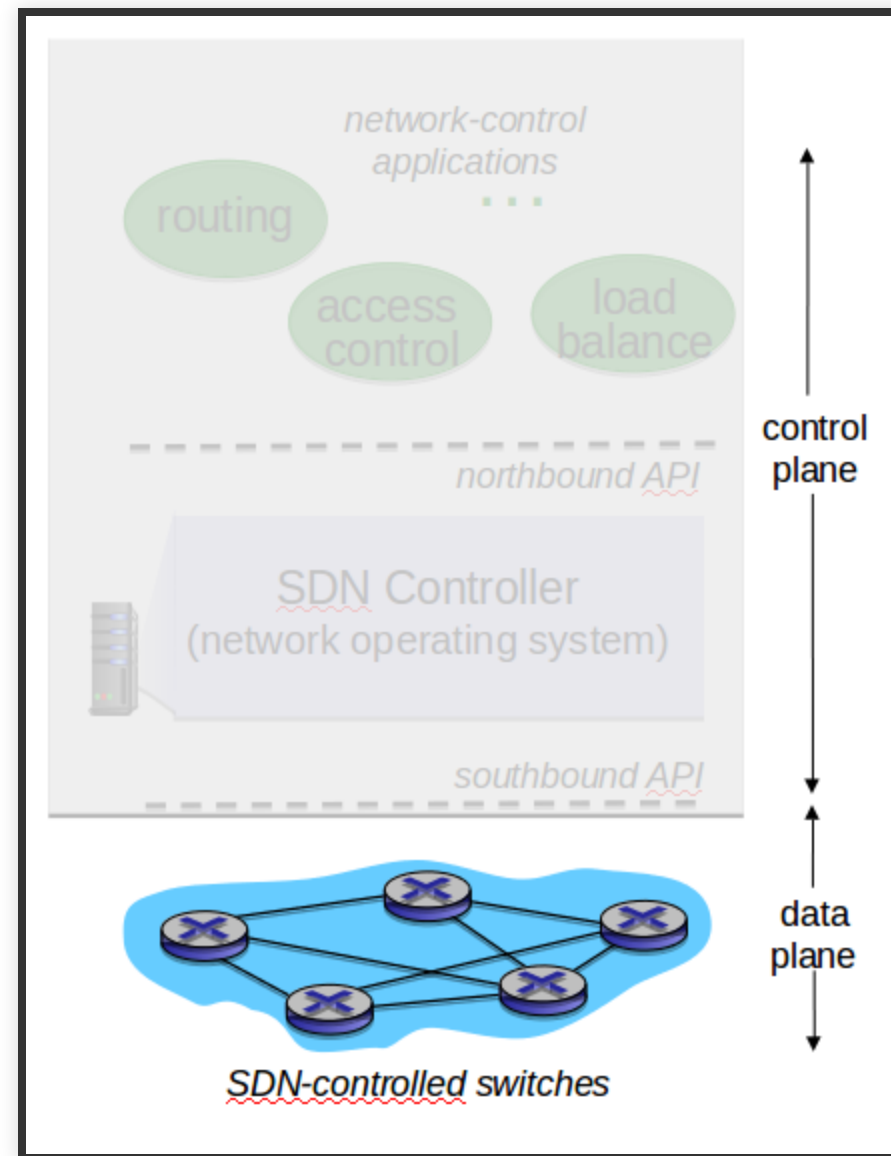
SOFTWARE DEFINED NETWORKING (SDN)



ANALOGY: MAINFRAME TO PC EVOLUTION



SDN PERSPECTIVE: DATA PLANE SWITCHES

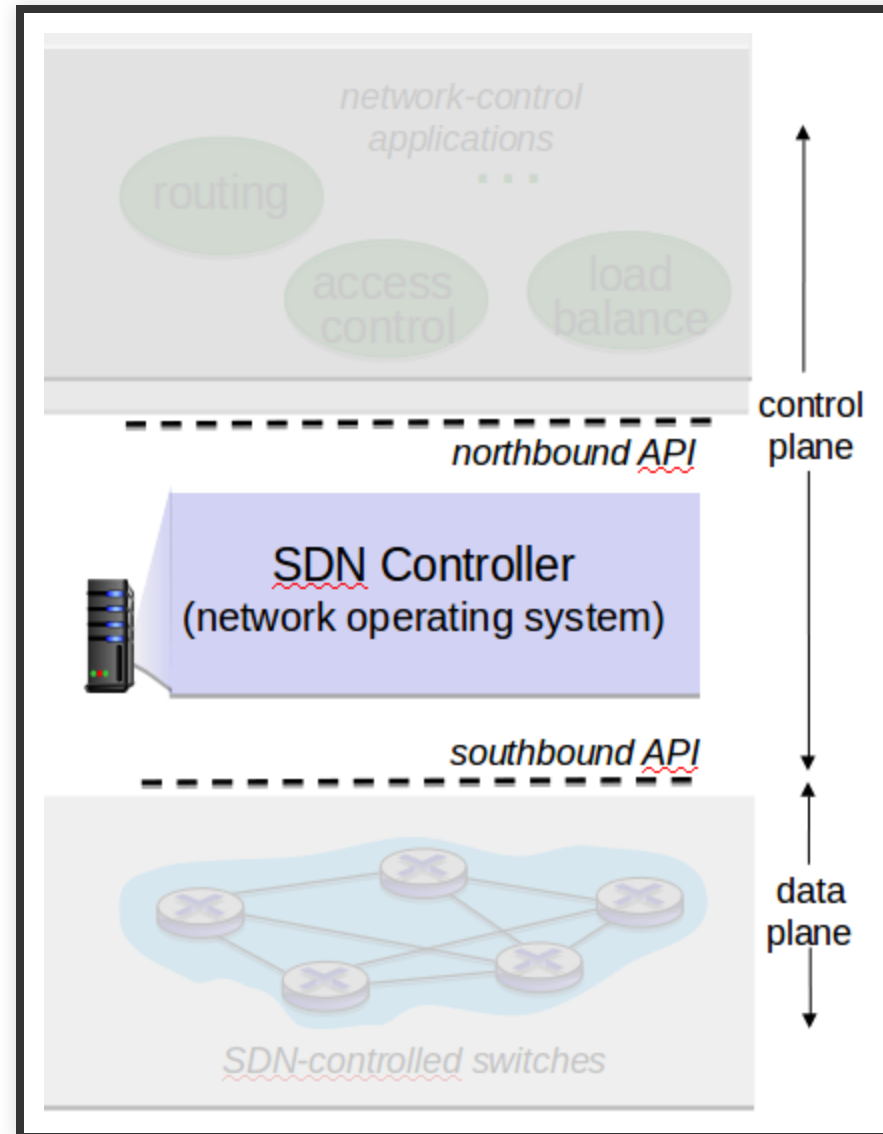


SDN PERSPECTIVE: DATA PLANE SWITCHES

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)

SDN PERSPECTIVE: SDN CONTROLLER

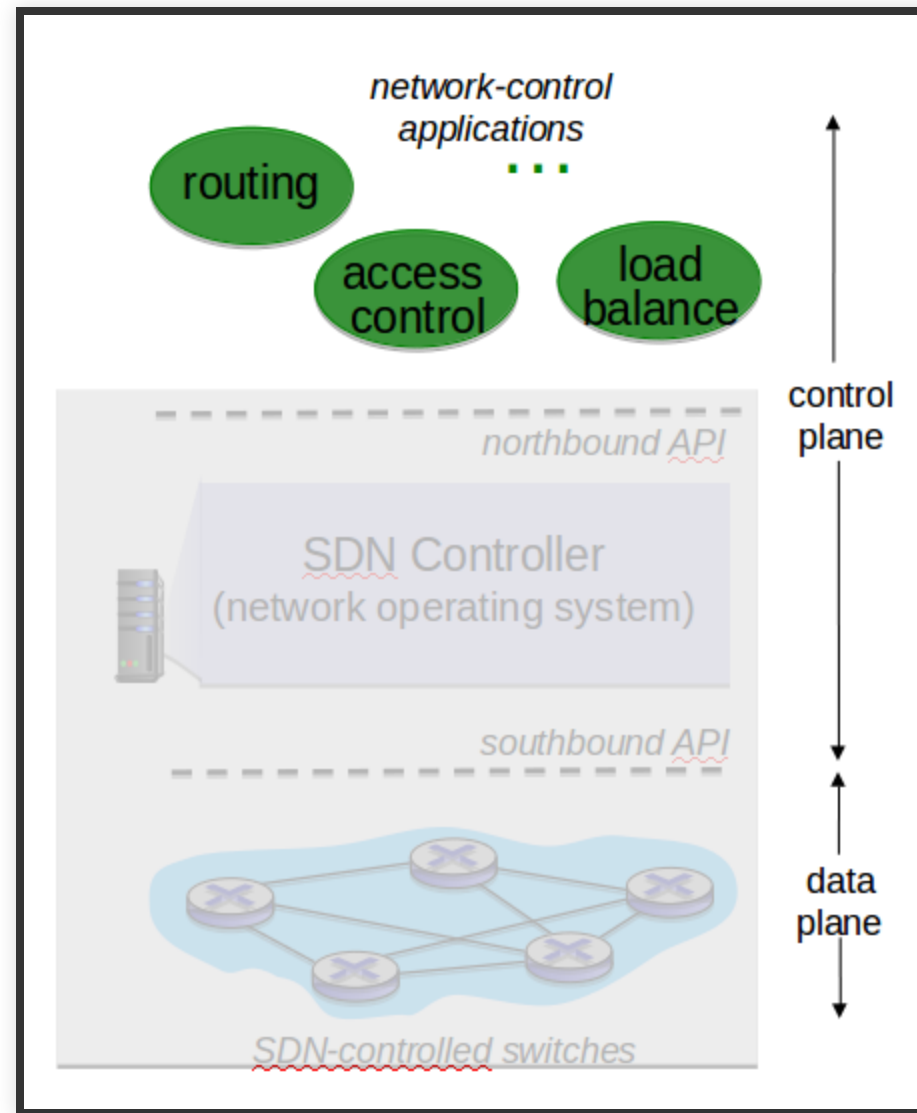


SDN PERSPECTIVE: SDN CONTROLLER

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

SDN PERSPECTIVE: CONTROL APPLICATIONS

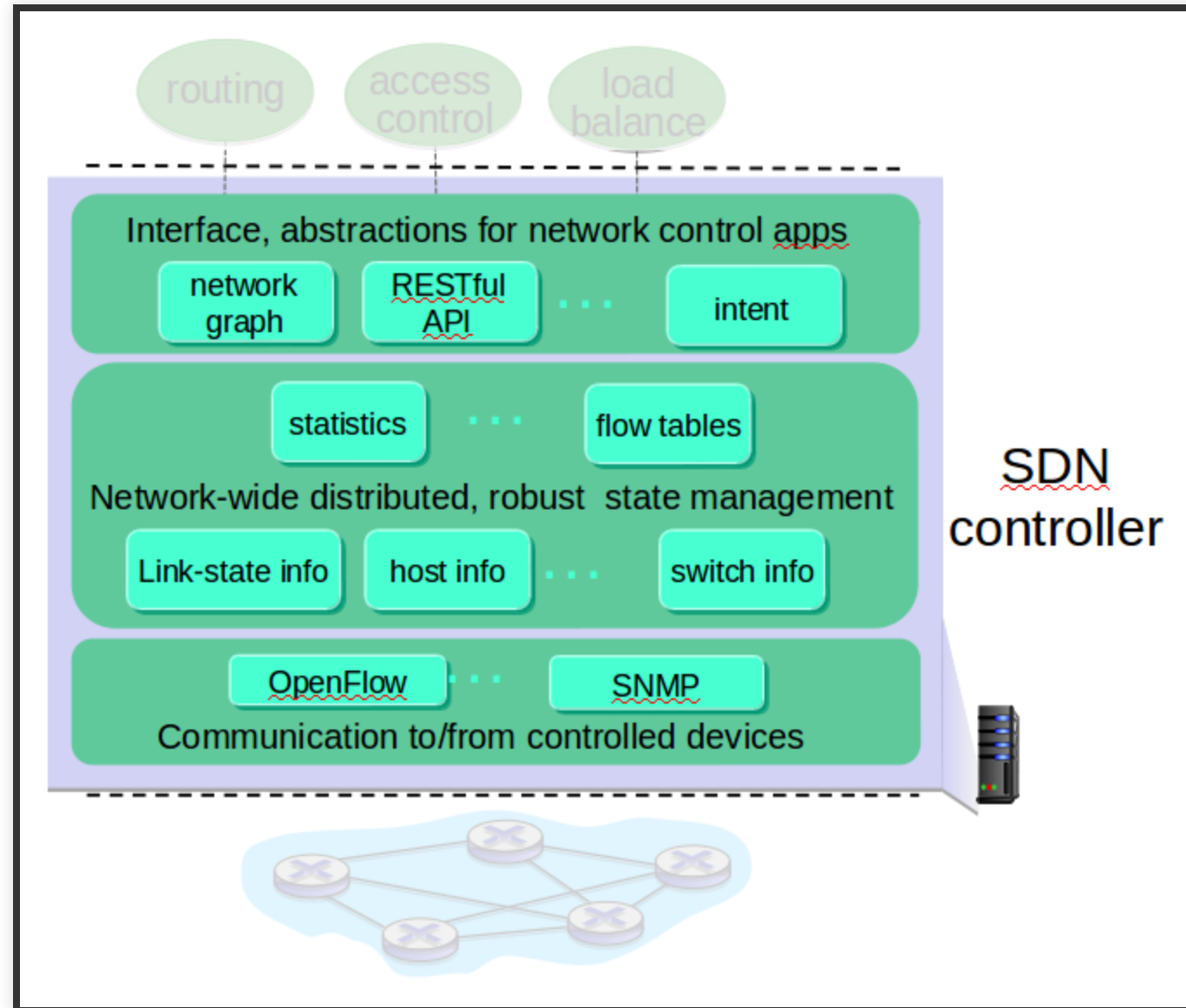


SDN PERSPECTIVE: CONTROL APPLICATIONS

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

COMPONENTS OF SDN CONTROLLER



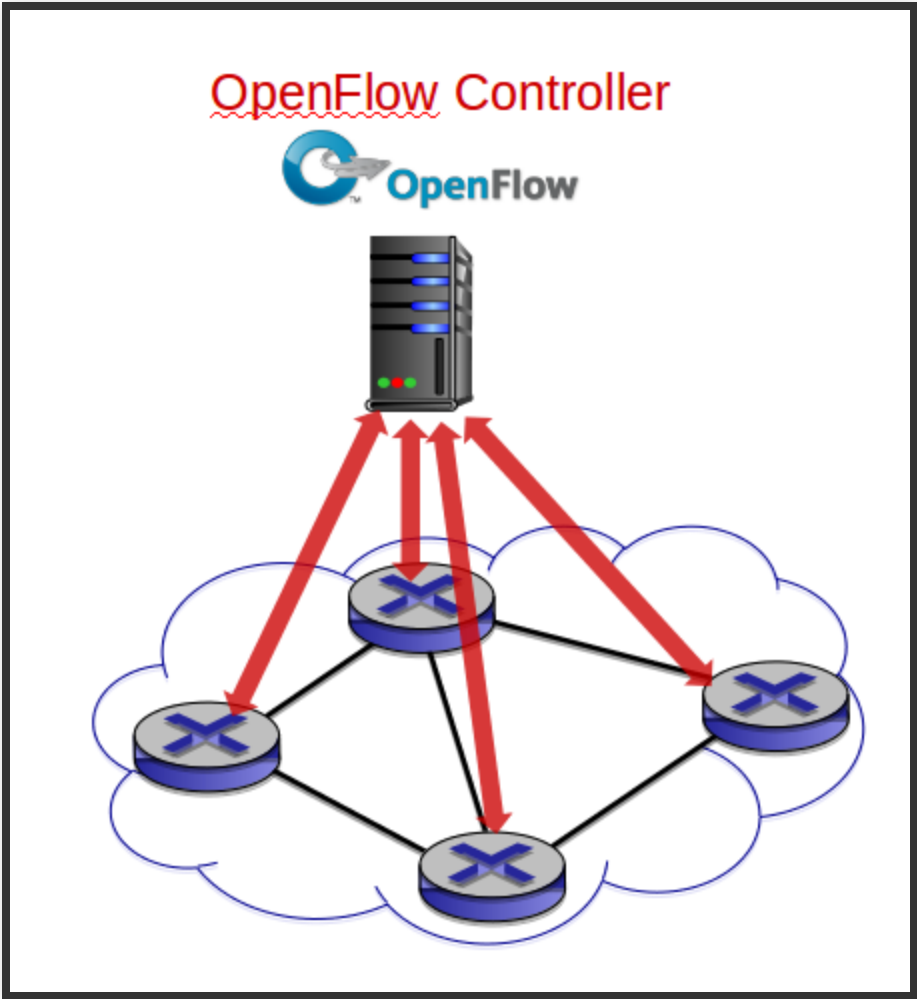
COMPONENTS OF SDN CONTROLLER

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

OPENFLOW PROTOCOL



OPENFLOW PROTOCOL

- Operates between controller, switch
- TCP used to exchange messages (Port 6653)
 - optional encryption
- 3 classes of OpenFlow messages:
 - controller-to-switch
 - asynchronous (switch to controller)
 - symmetric (misc)

OPENFLOW: CONTROLLER-TO-SWITCH MESSAGES

Key controller-to-switch messages

- **Configuration:** controller queries/sets switch configuration parameters
- **Modify-State:** add, delete, modify flow entries in the OpenFlow tables
- **Read-State:** controller queries switch features, switch replies (for statistics, counter values)
- **Send-Packet:** controller can send this packet out of specific switch port (Message includes package)

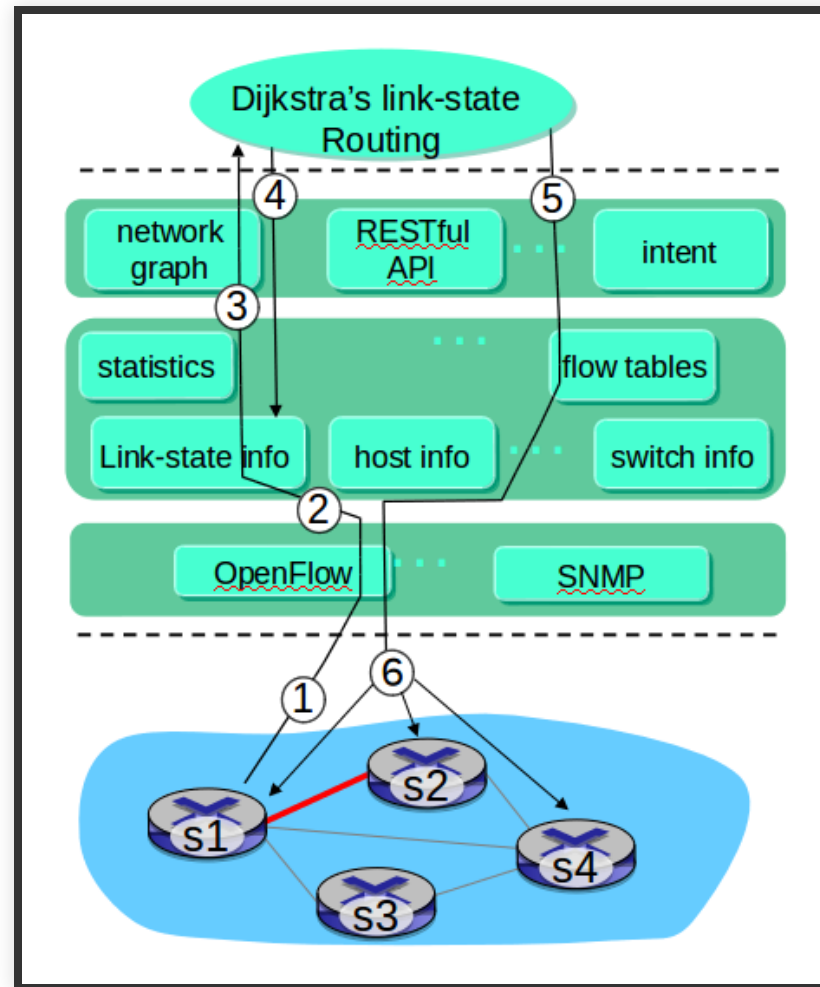
OPENFLOW: SWITCH-TO-CONTROLLER MESSAGES

Key switch-to-controller messages

- **Flow-Removed:** flow table entry deleted at switch (timeout etc)
- **Port status:** inform controller of a change on a port.
- **Packet-in:** transfer packet (and its control) to controller. (FX if not matching any rules)

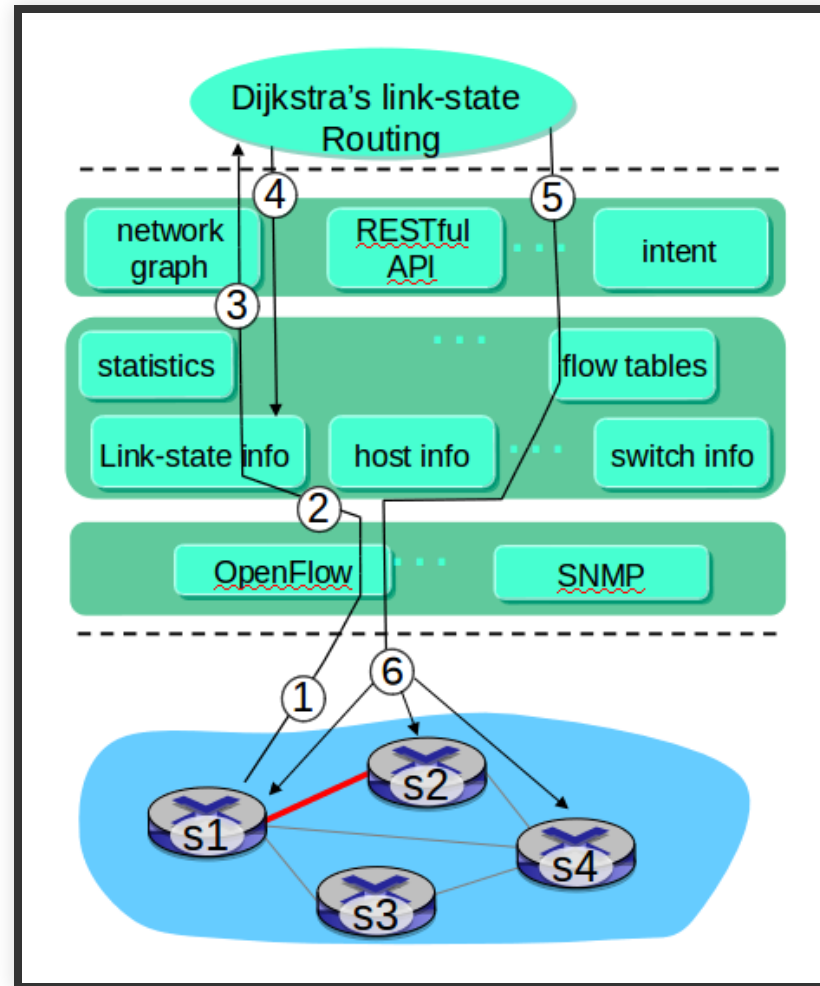
💡 Fortunately, network operators don't "program" switches by creating/sending OpenFlow messages directly. Instead use higher-level abstraction at controller

SDN: CONTROL/DATA PLANE INTERACTION EXAMPLE



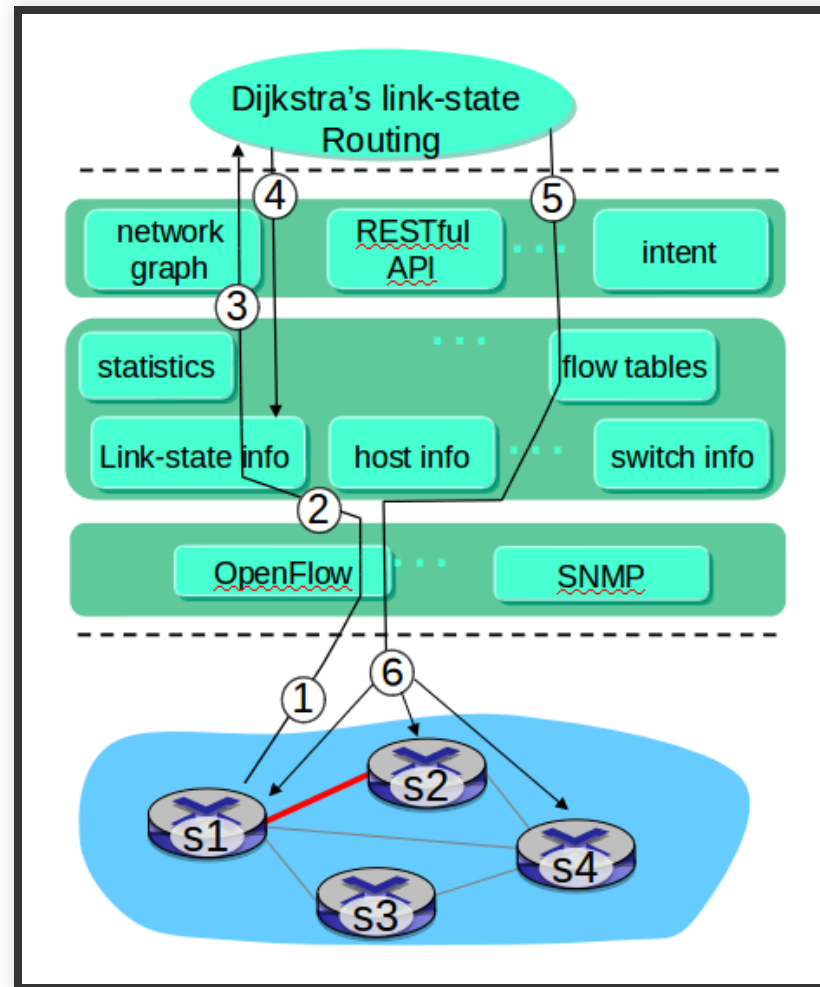
S1, experiencing link failure using OpenFlow port status message to notify controller

SDN: CONTROL/DATA PLANE INTERACTION EXAMPLE



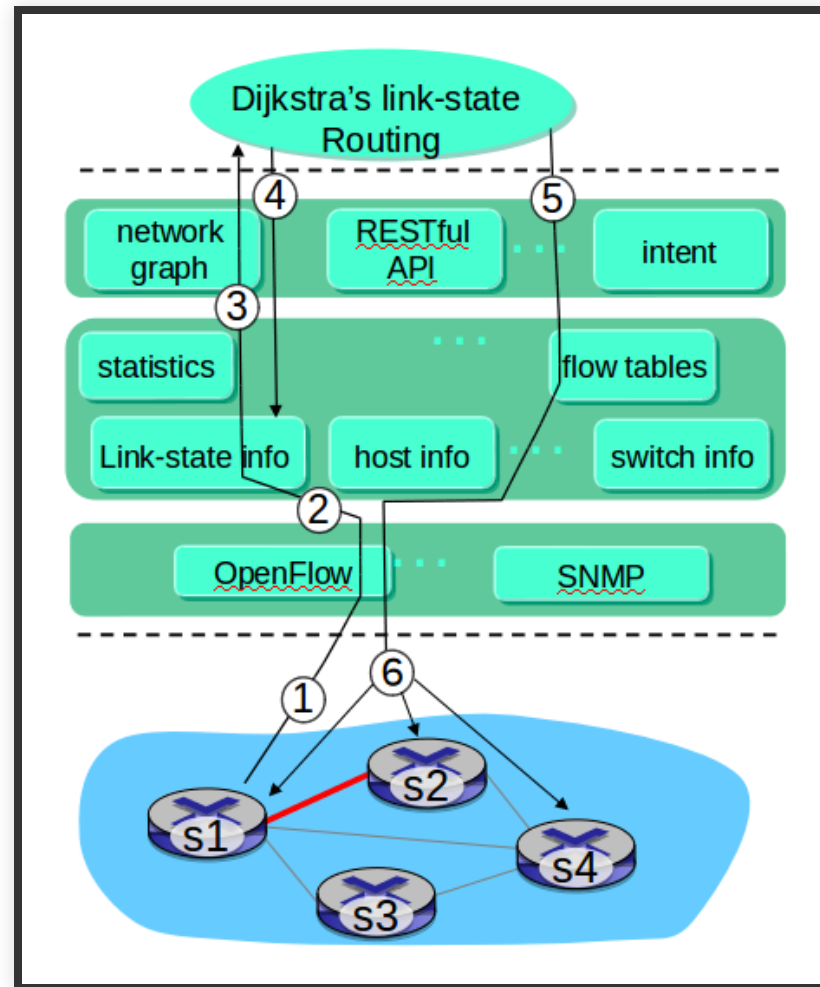
SDN controller receives OpenFlow message, updates link status info

SDN: CONTROL/DATA PLANE INTERACTION EXAMPLE



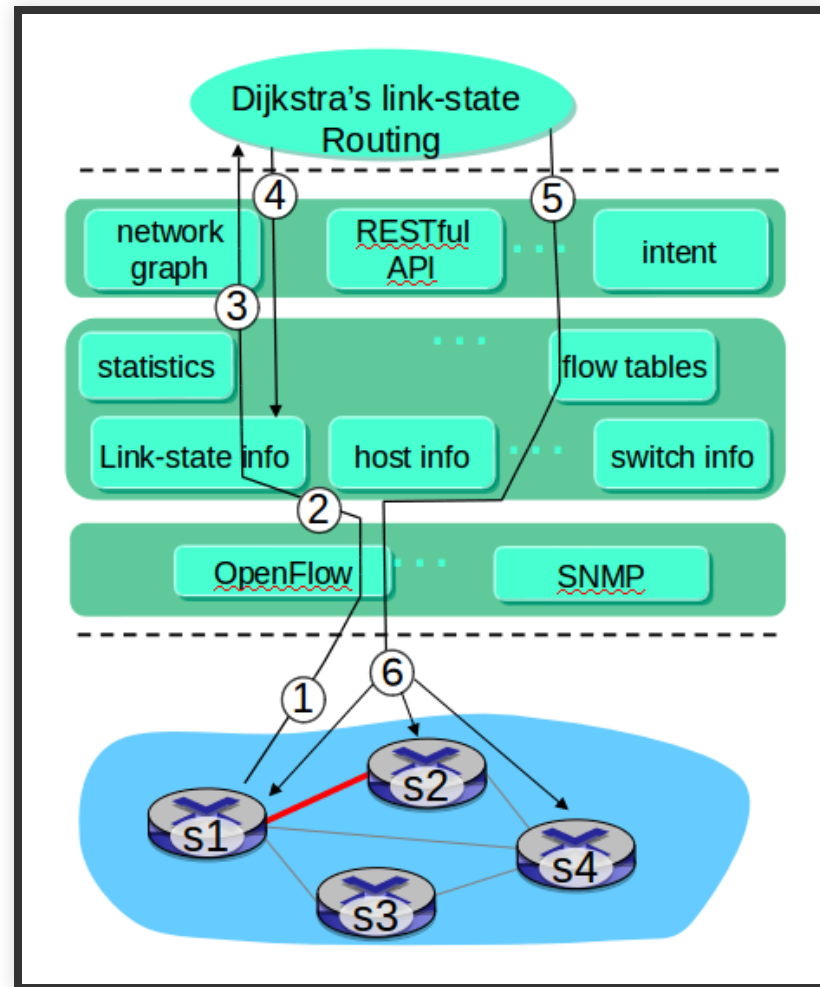
Dijkstra's routing algorithm application has previously registered to be called when ever link status changes. It is called.

SDN: CONTROL/DATA PLANE INTERACTION EXAMPLE



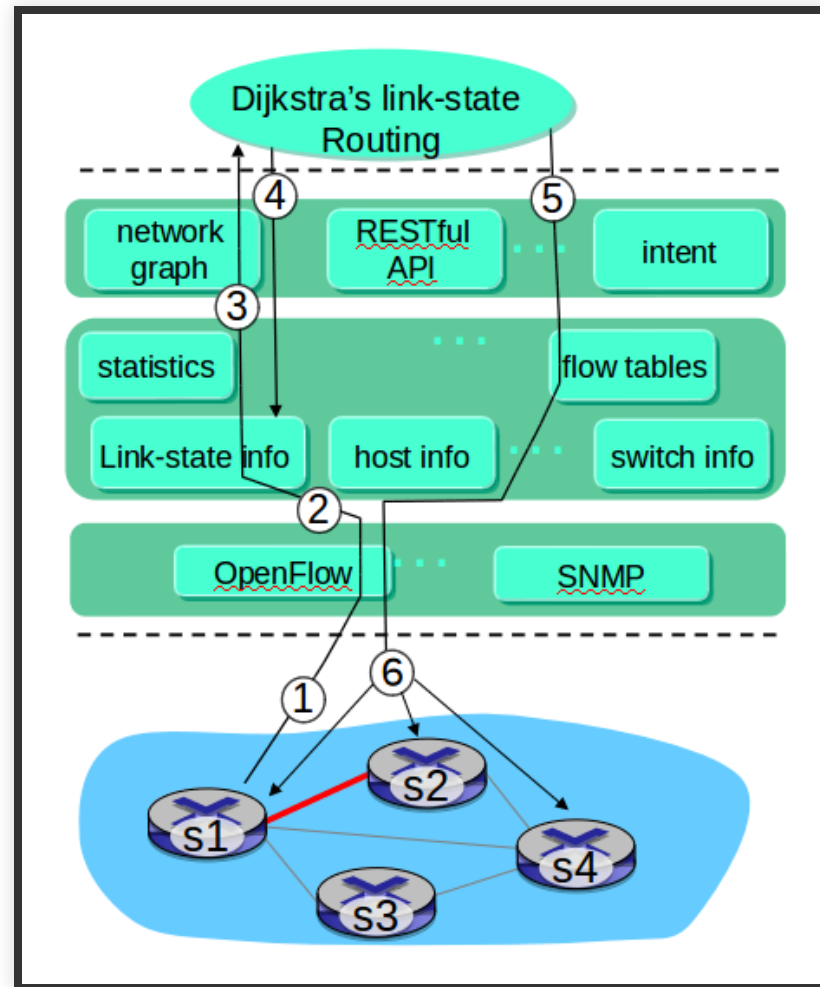
Dijkstra's routing algorithm access network graph info, link state info in controller, computes new routes

SDN: CONTROL/DATA PLANE INTERACTION EXAMPLE



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

SDN: CONTROL/DATA PLANE INTERACTION EXAMPLE



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

ICMP

ICMP

Internet control message protocol

- used by hosts and 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: TYPES

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

PING AND ICMP

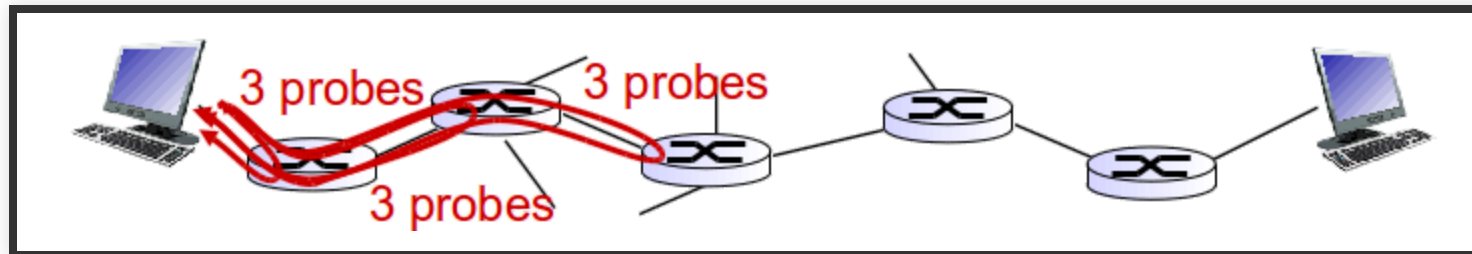
ICMP Type 8 Code 0 Message to the specified host

TRACEROUTE AND ICMP

- source sends series of UDP segments to dest
 - first set has TTL =1
 - second set has TTL=2, etc.
 - unlikely port number
- when nth set of datagrams arrives to nth router:
 - router discards datagrams
 - and sends source ICMP messages (type 11, code 0) ICMP messages includes name of router & IP address
- when ICMP messages arrives, source records RTTs

TRACEROUTE AND ICMP

- stopping criteria:
 - UDP segment eventually arrives at destination host
 - destination returns ICMP “port unreachable” message (type 3, code 3)
 - source stops



NETWORK MANAGEMENT AND SNMP

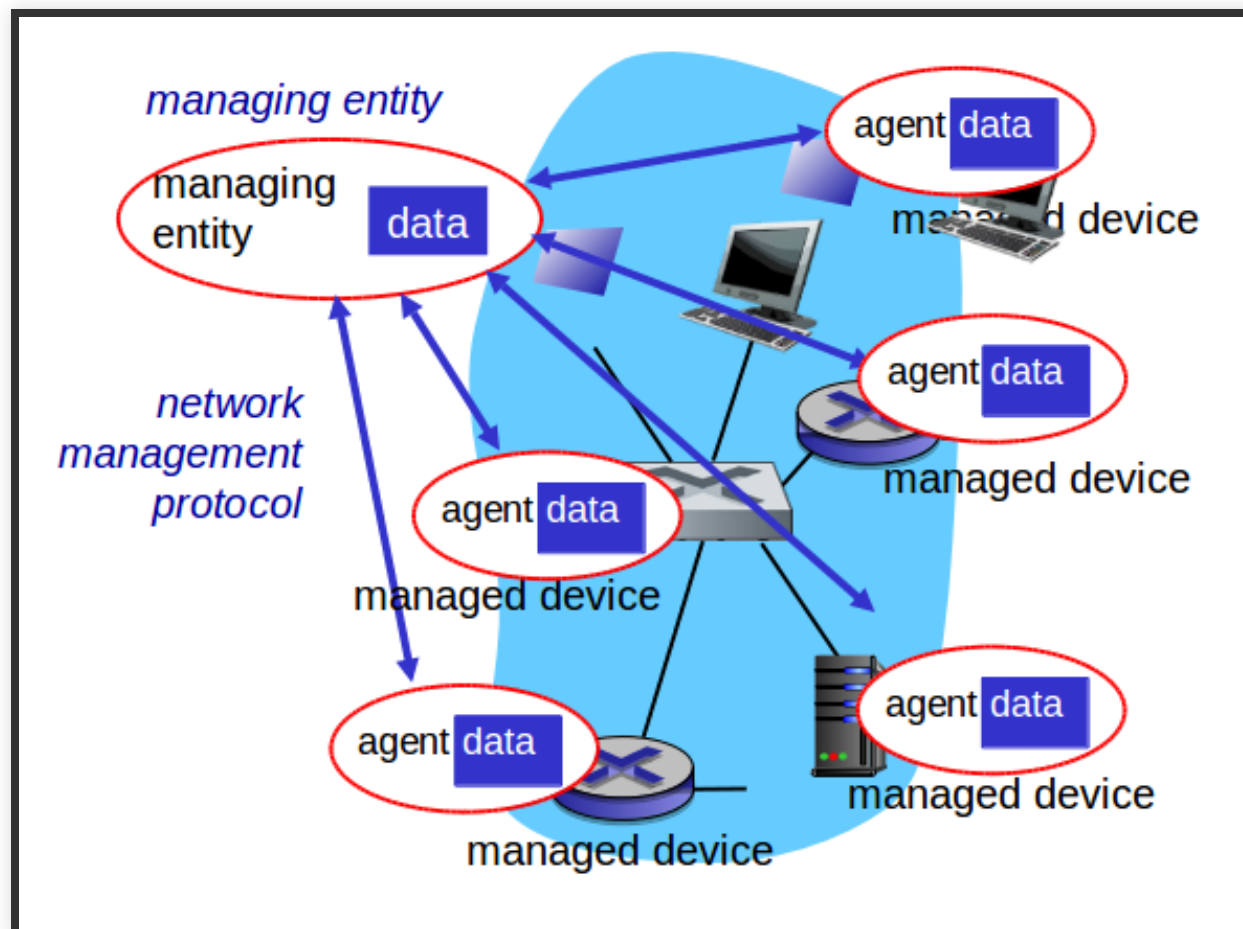
WHAT IS NETWORK MANAGEMENT?

autonomous systems (aka “network”): 1000s of interacting hardware/software components

💡 "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

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

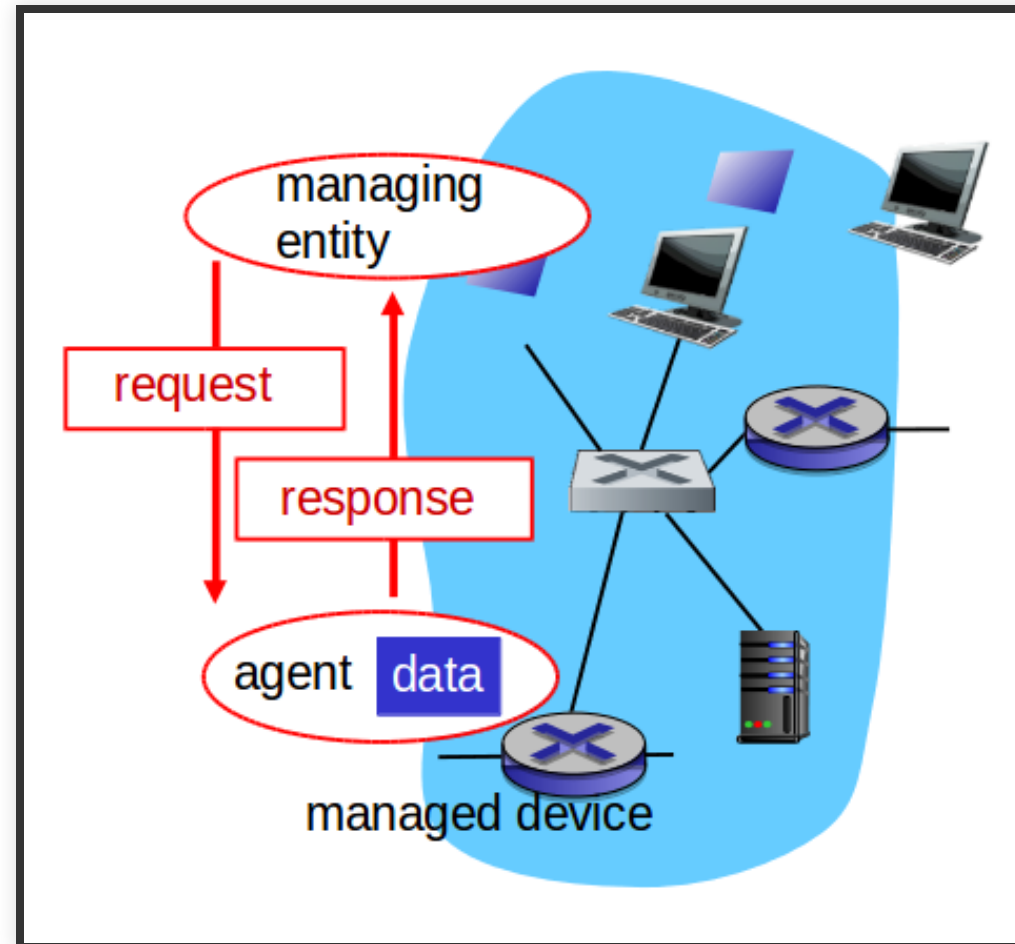


SNMP PROTOCOL

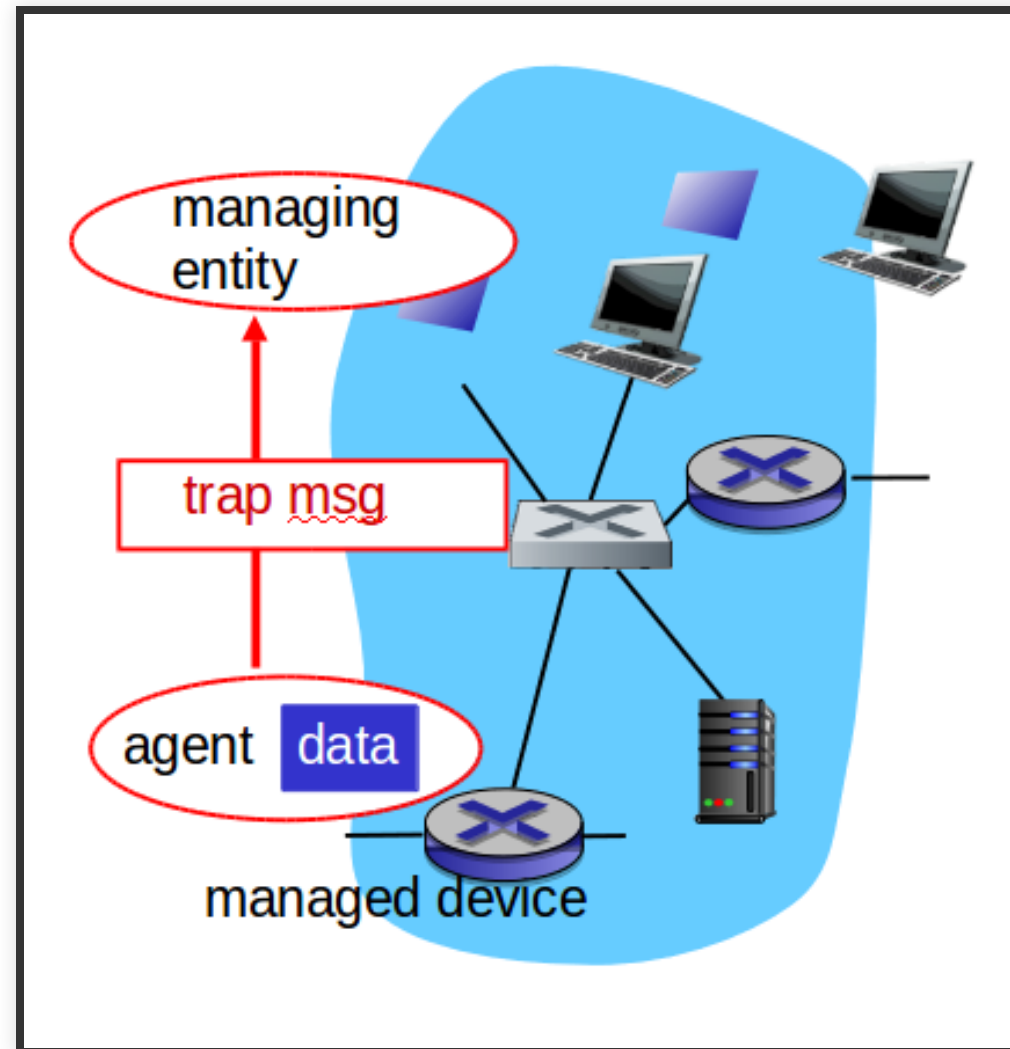
Two ways to convey MIB info, commands:

- request/response mode
- trap mode

SNMP PROTOCOL: REQUEST/RESPONSE MODE



SNMP PROTOCOL: 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

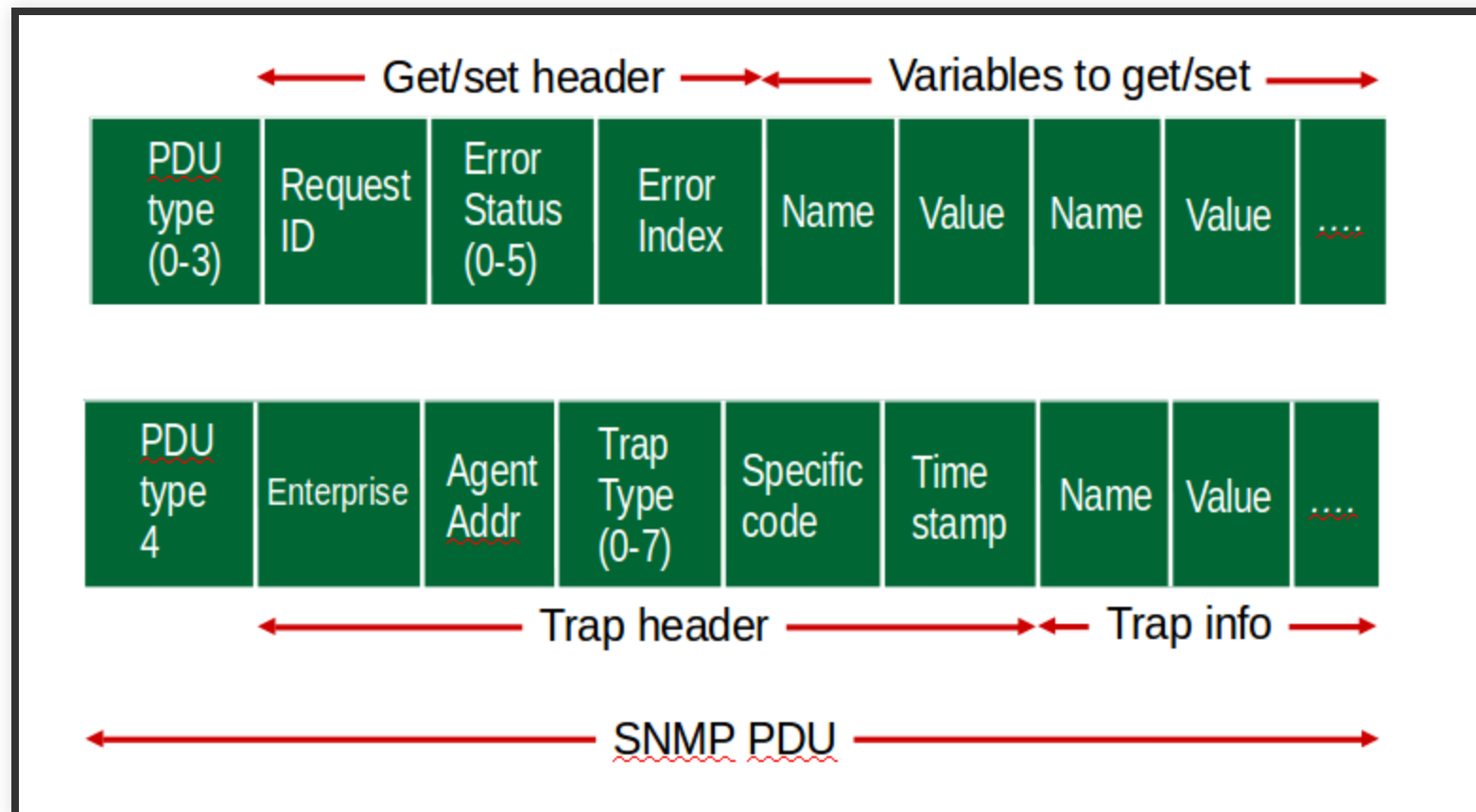
Message type

Function

Trap

Agent-to-manager: inform manager of exceptional event

SNMP PROTOCOL: MESSAGE FORMATS



SUMMARY

- 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
- Internet Control Message Protocol
- Network management (SNMP)