Chapter 5
Network Layer: Control Plane

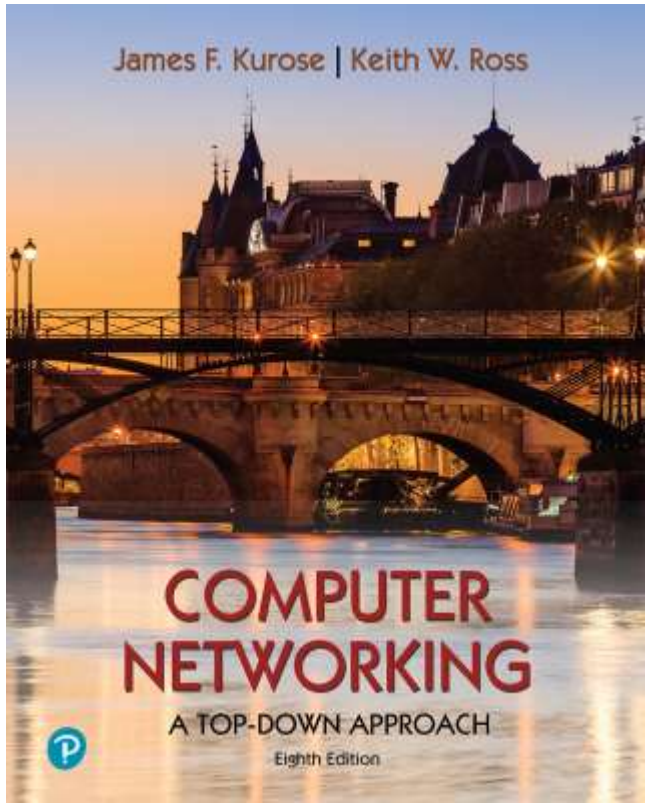
A note on the use of these PowerPoint slides:

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

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

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

Thanks and enjoy! JFK/KWR



Computer Networking: A Top-Down Approach

8th edition
Jim Kurose, Keith Ross
Pearson, 2020

Network layer control plane: our goals

- understand principles behind network control plane:
  - traditional routing algorithms
  - SDN controllers
  - network management, configuration
- instantiation, implementation in the Internet:
  - OSPF, BGP
  - OpenFlow, ODL and ONOS controllers
  - SNMP, YANG/NETCONF (NOT COVERED)

Network layer: "control plane" roadmap

- introduction
- routing protocols
  - link state
  - distance vector
- intra-ISP routing: OSPF
- routing among ISPs: BGP
- SDN control plane
- Internet Control Message Protocol (ICMP)

- network management, configuration
  - SNMP
  - NETCONF/YANG (NOT COVERED)

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)

- Recalling the next figures for *forwarding table* and the *flow table* were the principal elements that linked the network layer's data plane and control plane.

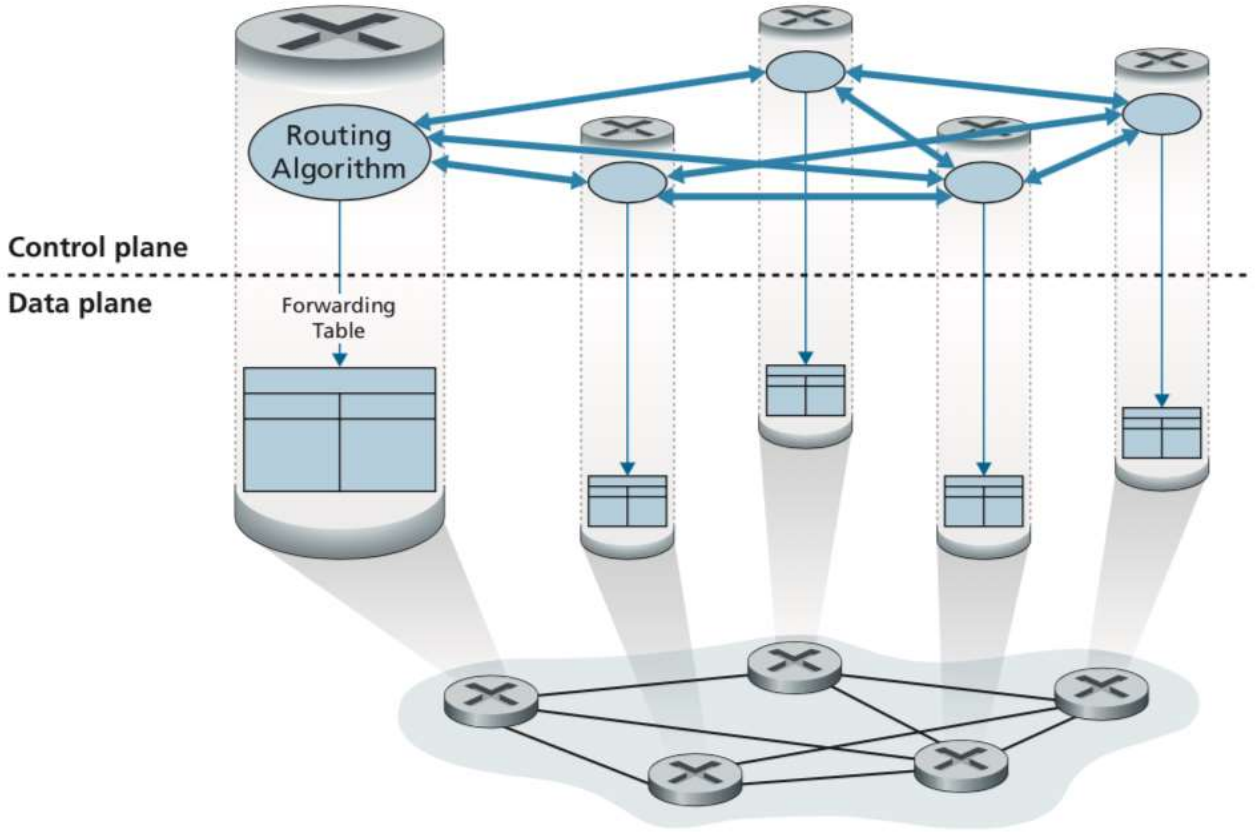
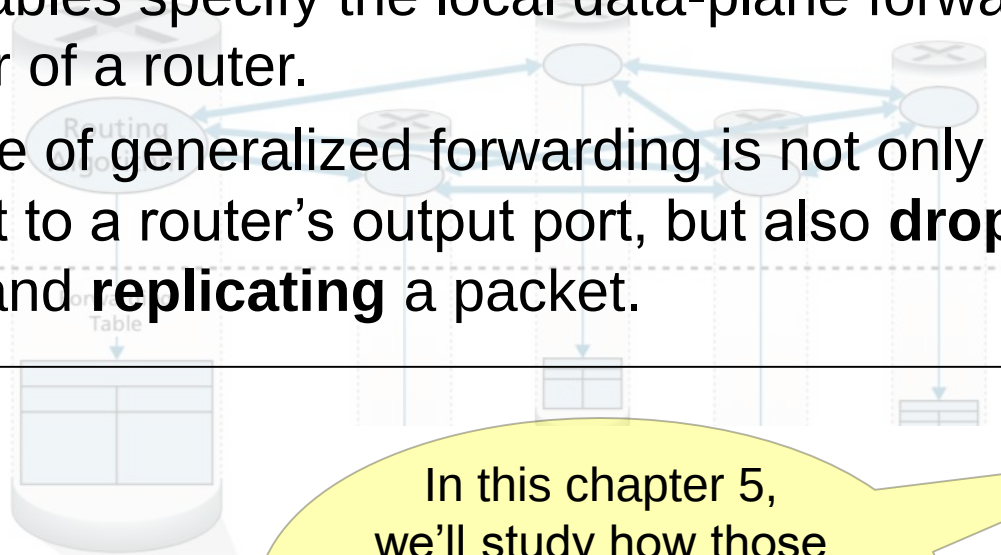


Figure:
Per-router control: Individual routing algorithm components interact in the control plane

Kurose, J.F. and Ross, K.W. (2021). Computer Networking: A Top-Down Approach (8th Edition). Pearson Education Limited, England. (pp. 378)

- Recalling the next figures for *forwarding table* and the *flow table* were the principal elements that linked the network layer's data plane and control plane.

- These tables specify the local data-plane forwarding behavior of a router.
- The case of generalized forwarding is not only **forwarding** a packet to a router's output port, but also **dropping** a packet and **replicating** a packet.



In this chapter 5, we'll study how those forwarding and flow tables are **computed**, **maintained** and **installed**.

Kurose, J.F. and Ross, K.W. (2021). Computer Networking: A Top-Down Approach (8th Edition). Pearson Education Limited, England. (pp. 378)

Two Approaches

(a) Per-router control:

The OSPF and BGP protocols

Section 5.3

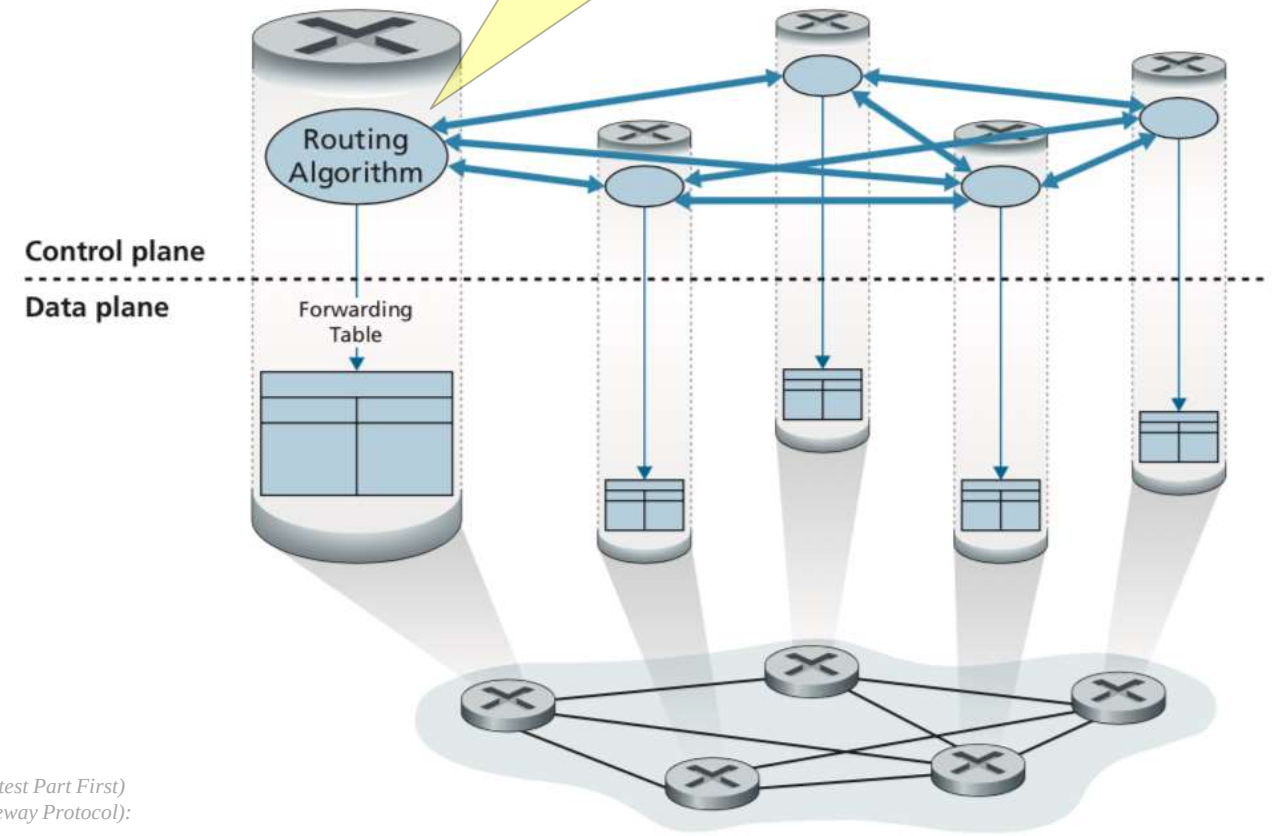


Figure:
Per-router control: Individual routing algorithm components interact in the control plane

OSPF (Open Shortest Part First)
BGP (Border Gateway Protocol):

Kurose, J.F. and Ross, K.W. (2021). Computer Networking: A Top-Down Approach (8th Edition). Pearson Education Limited, England. (pp. 378)

Two Approaches

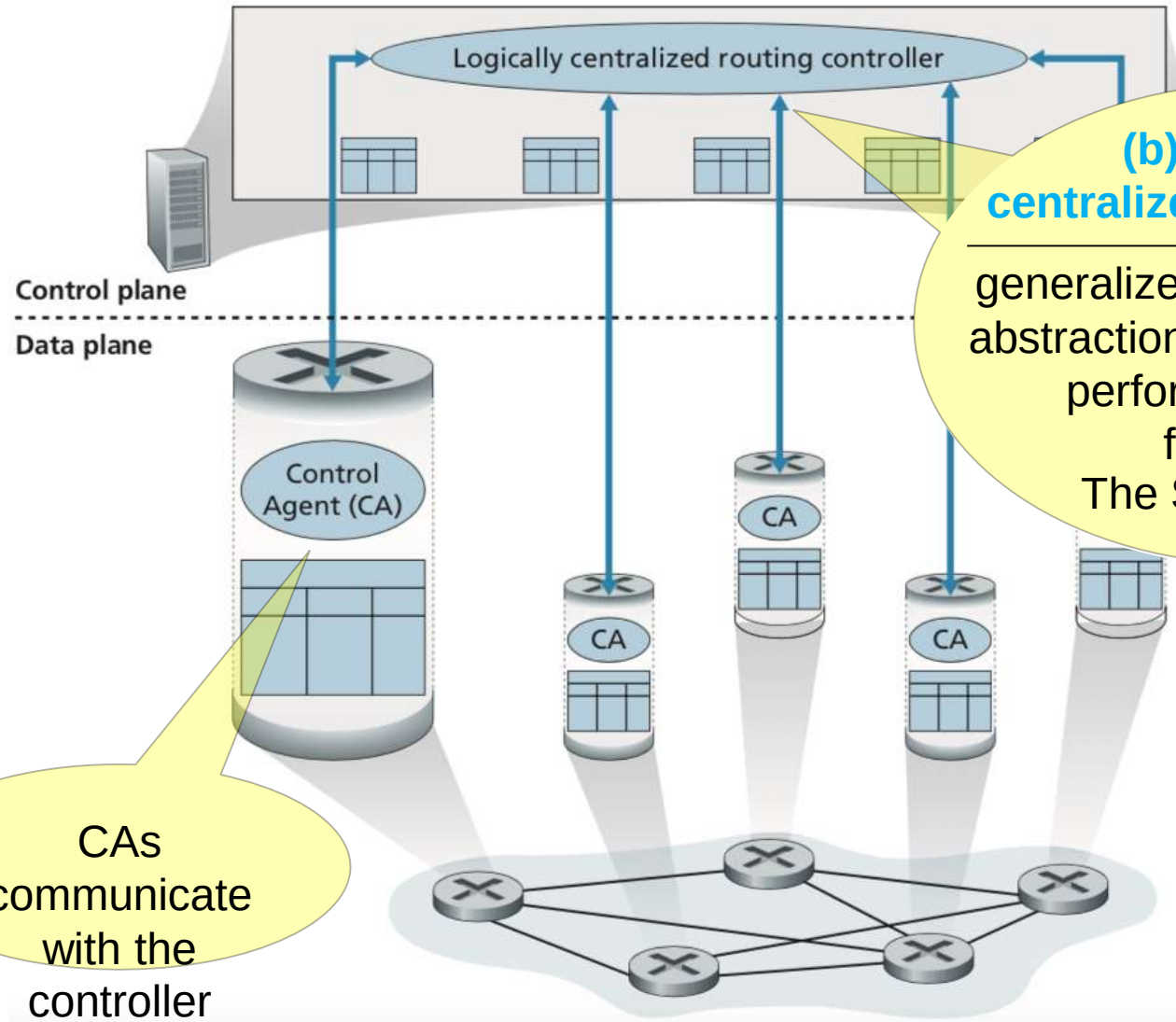


Figure:
Logically centralized control: A distinct, typically remote, controller interacts with local Control Agents (CAs)

Kurose, J.F. and Ross, K.W. (2021). Computer Networking: A Top-Down Approach (8th Edition). Pearson Education Limited, England. (pp. 379)

Network layer: "control plane" roadmap

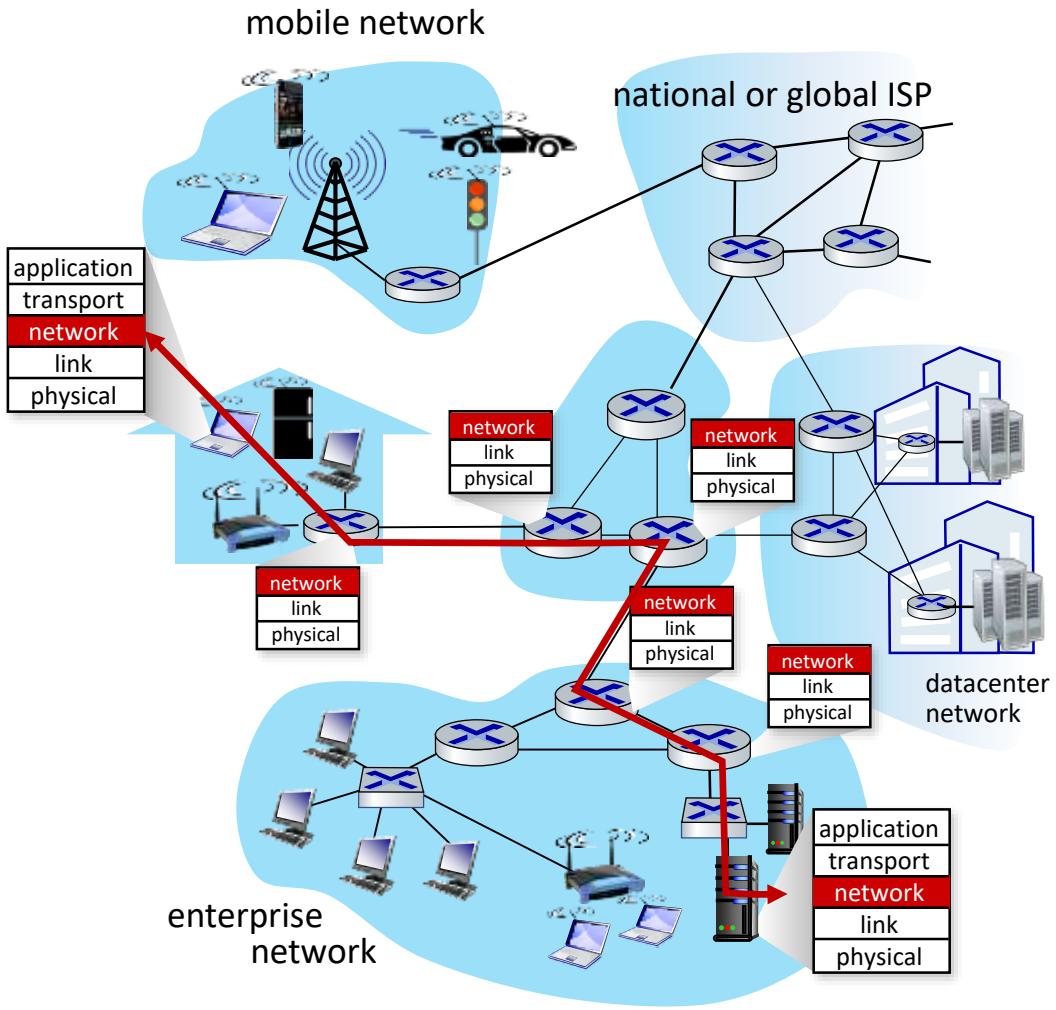
- introduction
- routing protocols
  - link state
  - distance vector
- intra-ISP routing: OSPF
- routing among ISPs: BGP
- SDN control plane
- Internet Control Message Protocol (ICMP)

- network management, configuration
  - SNMP
  - NETCONF/YANG (NOT COVERED)

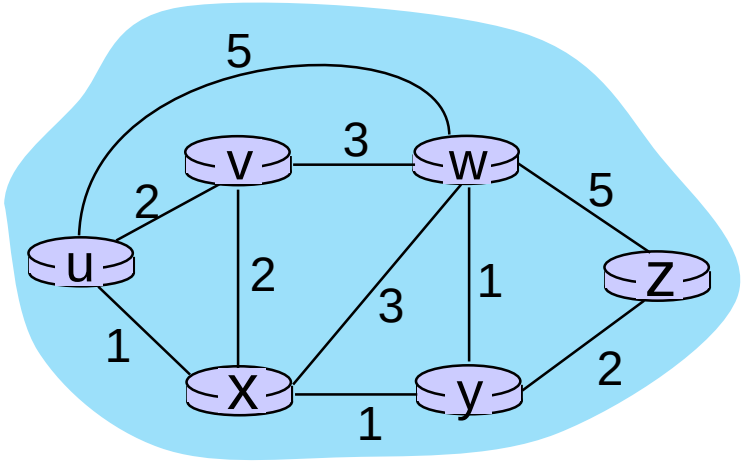
Routing protocols

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

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



Graph abstraction: link costs



$c_{a,b}$: cost of *direct* link connecting *a* and *b*

e.g., $c_{w,z} = 5$, $c_{u,z} = \infty$

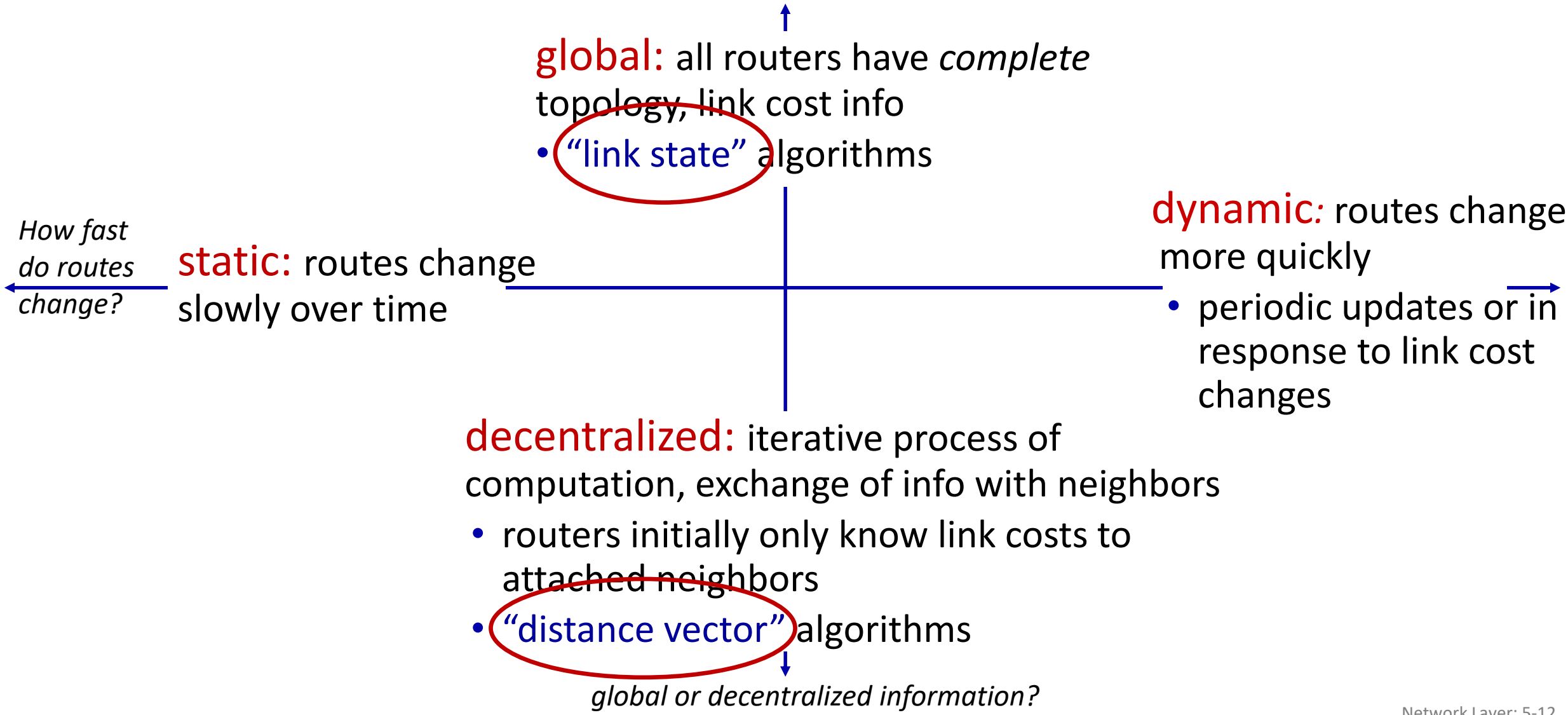
cost defined by network operator: could always be 1, or inversely related to bandwidth, or inversely related to congestion

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)* }

Routing algorithm classification



Network layer: "control plane" roadmap

- introduction
- routing protocols
  - link state
  - distance vector
- intra-ISP routing: OSPF
- routing among ISPs: BGP
- SDN control plane
- Internet Control Message Protocol (ICMP)

- network management, configuration
  - SNMP
  - NETCONF/YANG (NOT COVERED)

Dijkstra's link-state routing algorithm

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

notation

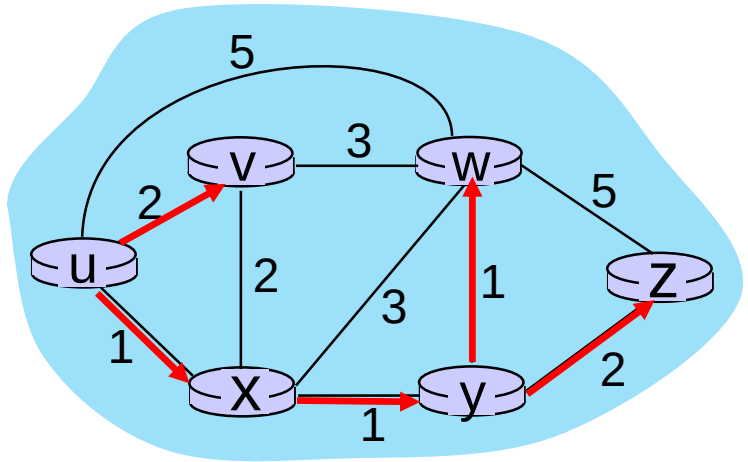
- $c_{x,y}$: direct link cost from node *x* to *y*; = ∞ if not direct neighbors
- *D(v): current* estimate of cost of least-cost-path from source to destination *v*
- *p(v):* predecessor node along path from source to *v*
- *N':* set of nodes whose least-cost-path *definitively* known

Dijkstra's link-state routing algorithm

```
1  Initialization:
2    N' = {u}                    /* compute least cost path from u to all other nodes */
3    for all nodes v
4      if v adjacent to u        /* u initially knows direct-path-cost only to  direct neighbors  */
5        then D(v) = c_u,v       /* but may not be minimum cost!                                  */
6      else D(v) = ∞
7
8  Loop
9      find a not in N' such that D(a) is a minimum
10     add a to N'
11     update D(b) for all b adjacent to a and not in N' :
12         D(b) = min ( D(b),  D(a) + c_a,b )
13     /* new least-path-cost to b is either old least-cost-path to b or known
14     least-cost-path to b plus direct-cost from b to a */
15 until all nodes in N'
```

Dijkstra's algorithm: an 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					

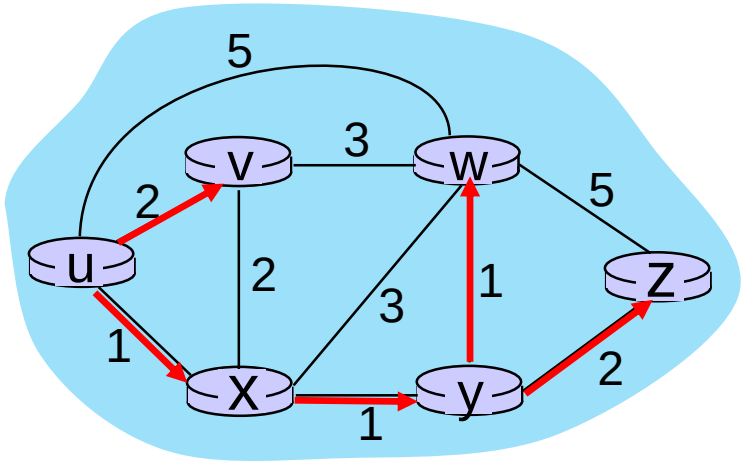


Initialization (step 0): For all a: if *a* adjacent to then $D(a) = c_{u,a}$

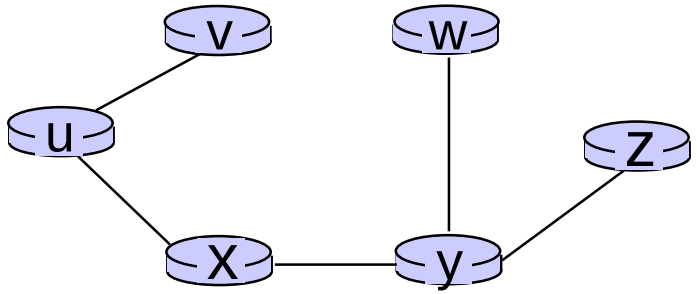
find *a* not in *N'* such that *D(a)* is a minimum
add *a* to *N'*
update *D(b)* for all *b* adjacent to *a* and not in *N'* :

$$D(b) = \min (D(b), D(a) + c_{a,b})$$

Dijkstra's algorithm: an example



resulting least-cost-path tree from u:



resulting forwarding table in u:

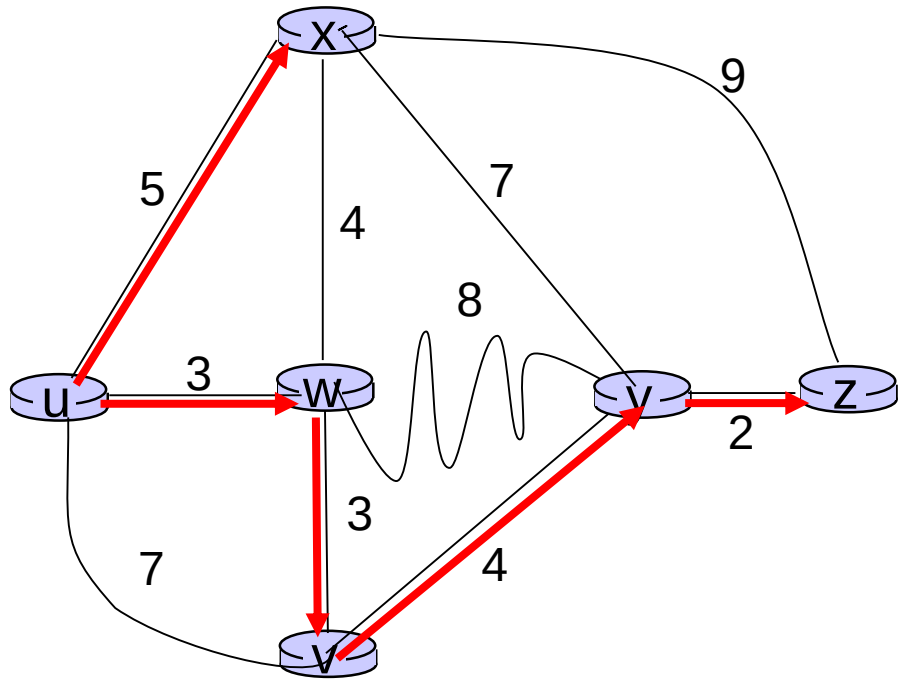
destination	outgoing link
v	(u,v)
x	(u,x)
y	(u,x)
w	(u,x)
z	(u,x)

route from *u* to *v* directly

route from u to all other destinations via *x*

Dijkstra's algorithm: another example

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

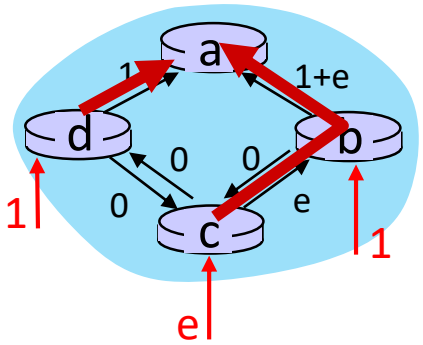


notes:

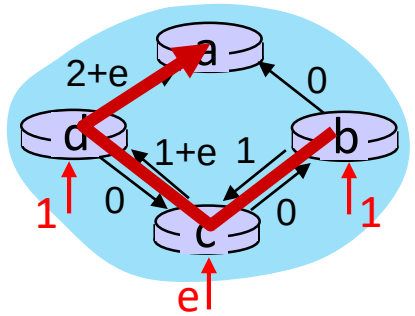
- construct least-cost-path tree by tracing predecessor nodes
- ties can exist (can be broken arbitrarily)

Dijkstra's algorithm: oscillations possible

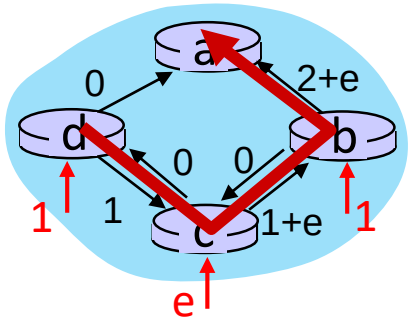
- when link costs depend on traffic volume, route oscillations possible
- sample scenario:
  - routing to destination a, traffic entering at d, c, e with rates 1, e (<1), 1
  - link costs are directional, and volume-dependent



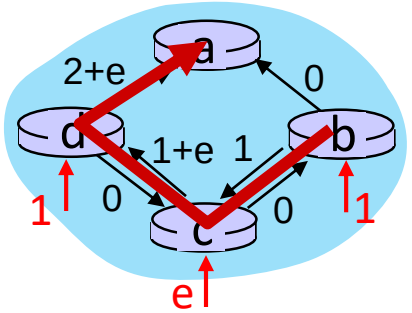
initially



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

Exercise 5.0

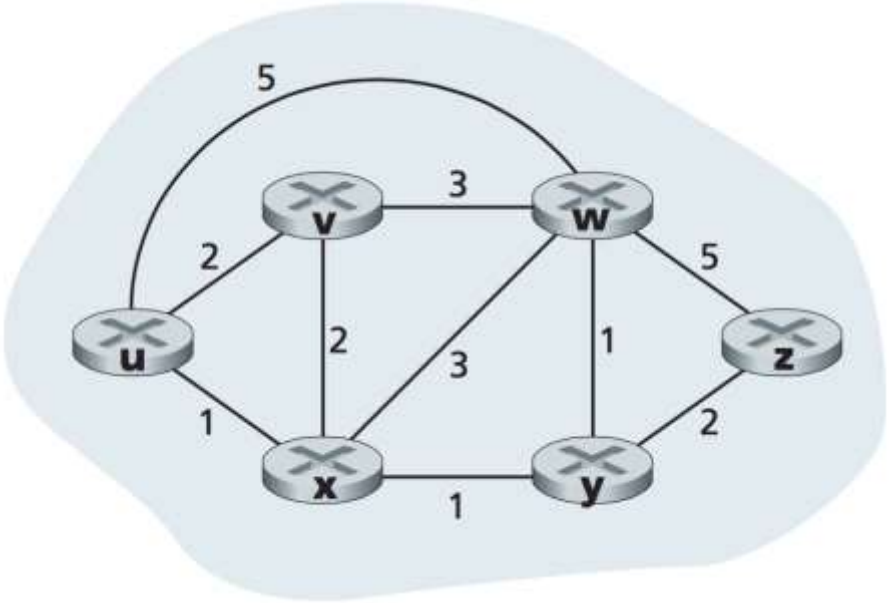
Dijkstra's Algorithm

Routers in nodes u, v, w, x, y and z have been assigned with link cost value as stated in the diagram.

(a) Construct table least cost paths from node z to node u
- construct shortest path tree by tracing predecessor nodes

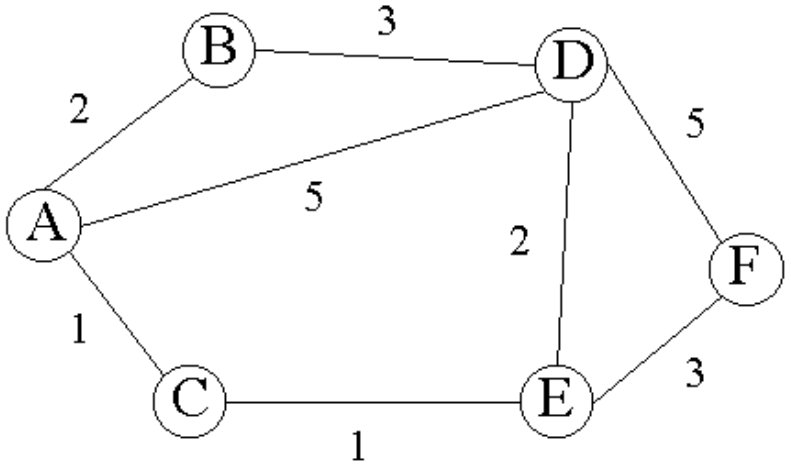
(b) Computes least cost paths from node z to node u

(c) Produce forwarding table for node z



Exercise 5.1

Routers in nodes A, B, C, D, E and F have been assigned with link cost value as stated in the diagram.



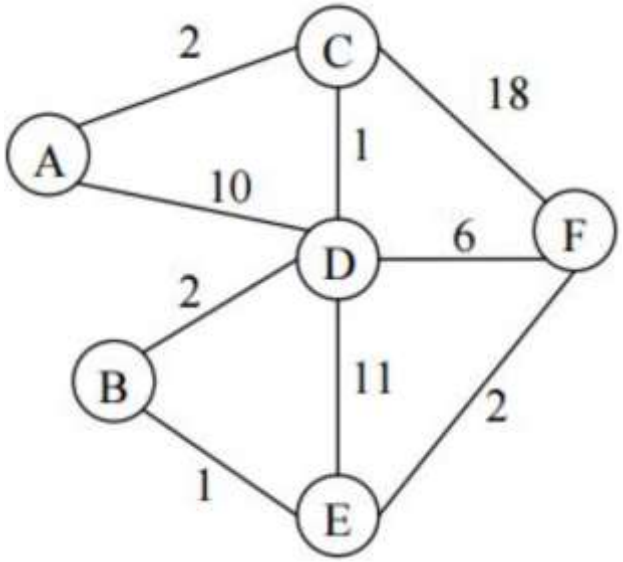
(a) Construct table least cost paths from node A to node F using Dijkstra's algorithm
(b) Draw least cost paths from node A to node F
(c) Produce forwarding table for node A

http://www4.ncsu.edu/~chou/course/Diagrams/distance-vector.gif

Exercise 5.2

Routers in nodes A, B, C, D, E, and F have been assigned with link cost value as stated in the diagram.

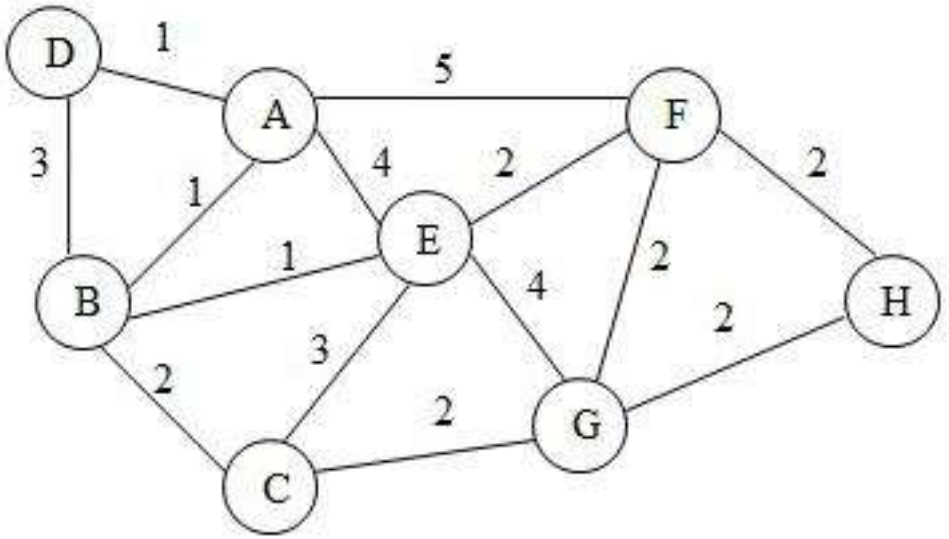
(a) Construct table least cost paths from node D to node H using Dijkstra's algorithm
(b) Draw least cost paths from node D to node H
(c) Produce forwarding table for node D



https://i.stack.imgur.com/bcz2a.jpg

Exercise 5.3

Routers in nodes A, B, C, D, E, F, G and H have been assigned with link cost value as stated in the diagram.



(a) Construct table least cost paths from node D to node H using Dijkstra's algorithm

(b) Draw least cost paths from node D to node H

(c) Produce forwarding table for node D

http://s3.amazonaws.com/answer-board-image/6bfef7b5-05b9-4210-afb5-712510e57a37.jpeg

Network layer: "control plane" roadmap

- introduction
- routing protocols
  - link state
  - distance vector
- intra-ISP routing: OSPF
- routing among ISPs: BGP
- SDN control plane
- Internet Control Message Protocol (ICMP)

- network management, configuration
  - SNMP
  - NETCONF/YANG (NOT COVERED)

Distance vector algorithm

Based on *Bellman-Ford* (BF) equation (dynamic programming):

Bellman-Ford equation

Let $D_x(y)$: cost of least-cost path from x to y.
Then:

$$D_x(y) = \min_v \{ c_{x,v} + D_v(y) \}$$

min taken over all neighbors v of x

direct cost of link from x to v

v's estimated least-cost-path cost to y

Distance vector algorithm

key idea:

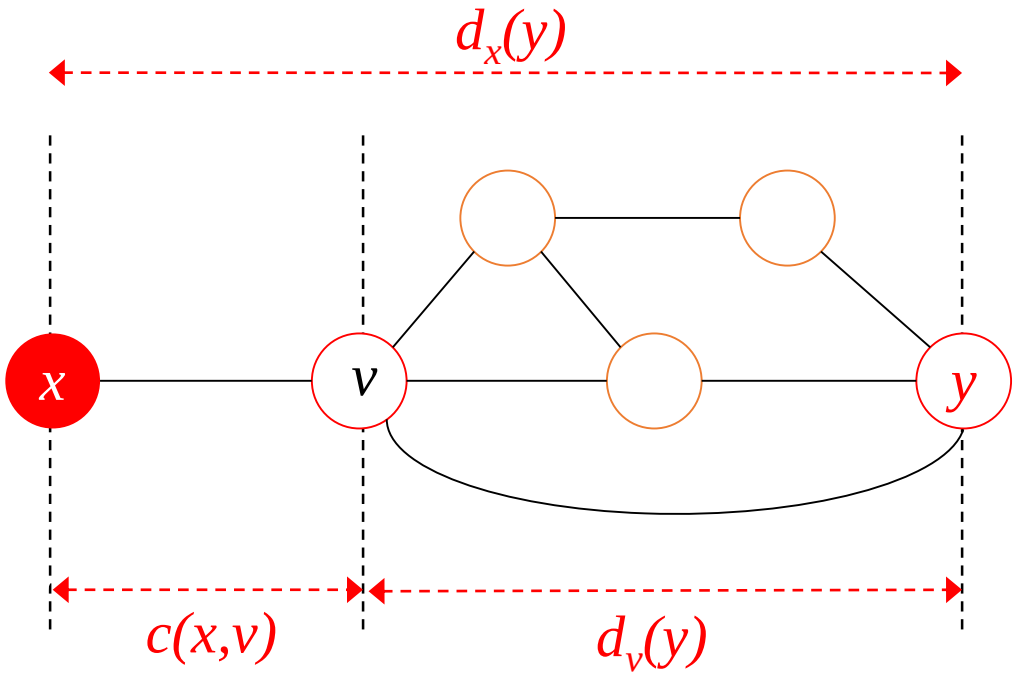
- from time-to-time, each node sends its own distance vector estimate to neighbors
- when *x* receives new DV estimate from any neighbor, it updates its own DV using B-F equation:

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

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

The Distance Vector (DV)

Bellman-Ford



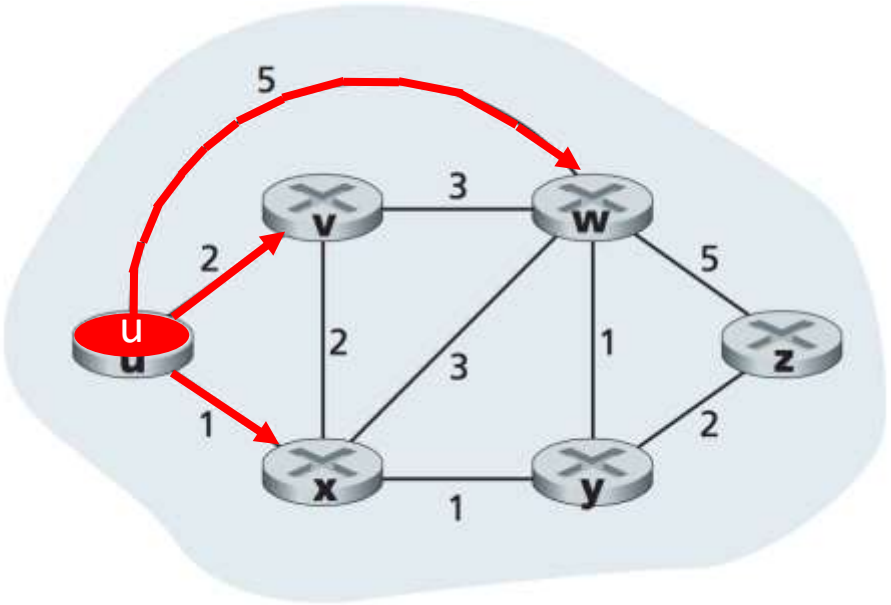
$$d_x(y) = \min_v \{c(x,v) + d_v(y)\}$$

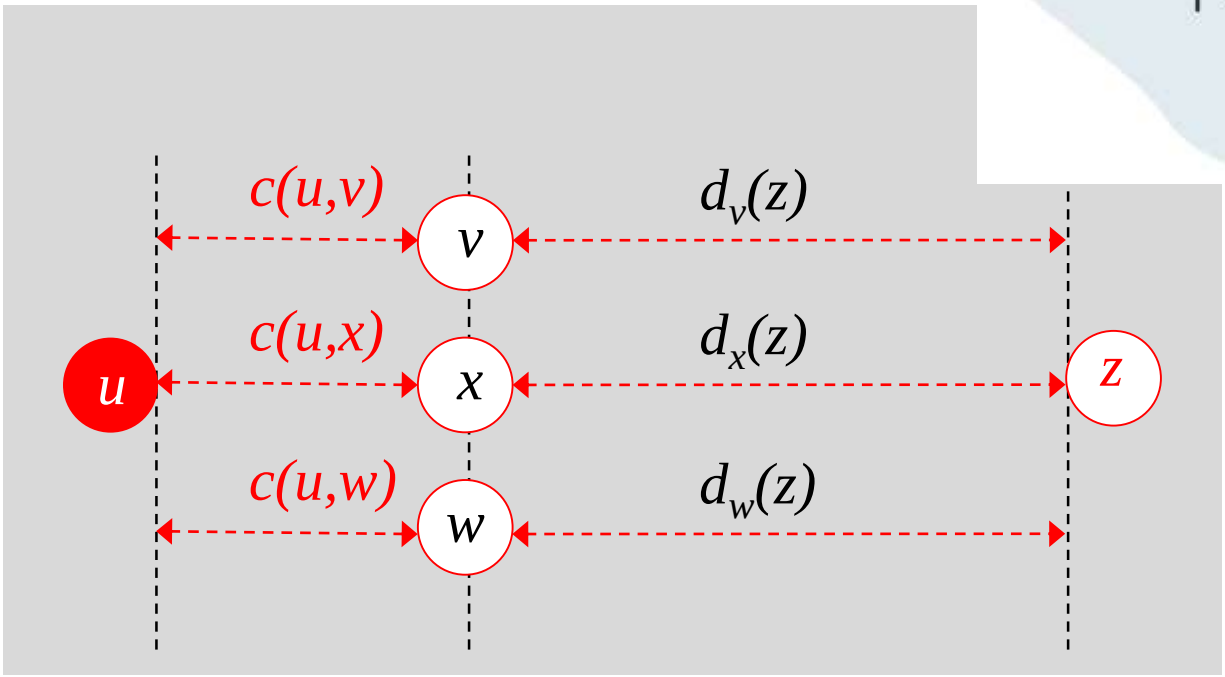
- under minor, natural conditions, the estimate $d_x(y)$ *converge to the actual least cost* $d_x(y)$

The Distance Vector (DV)

Bellman-Ford

Example:
Neighbor to u are v, x, and w;

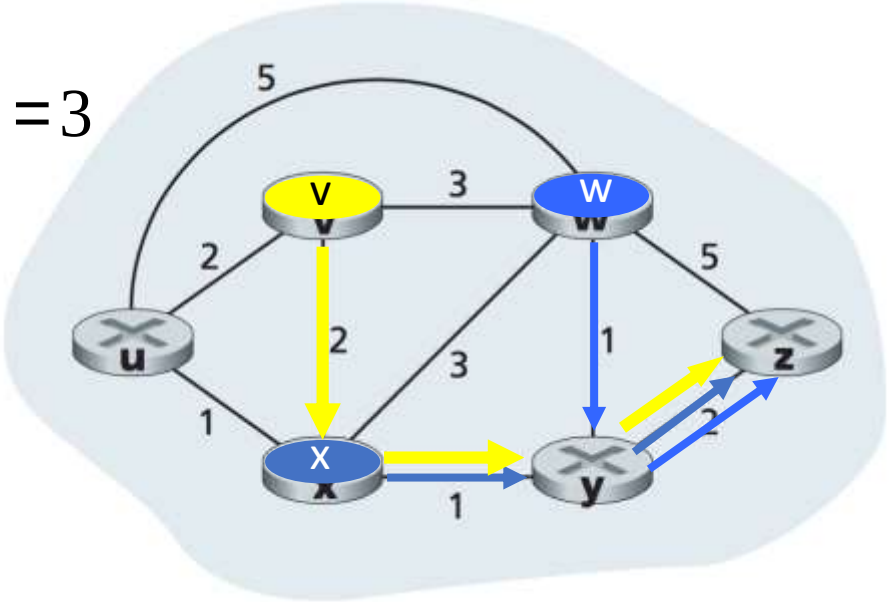


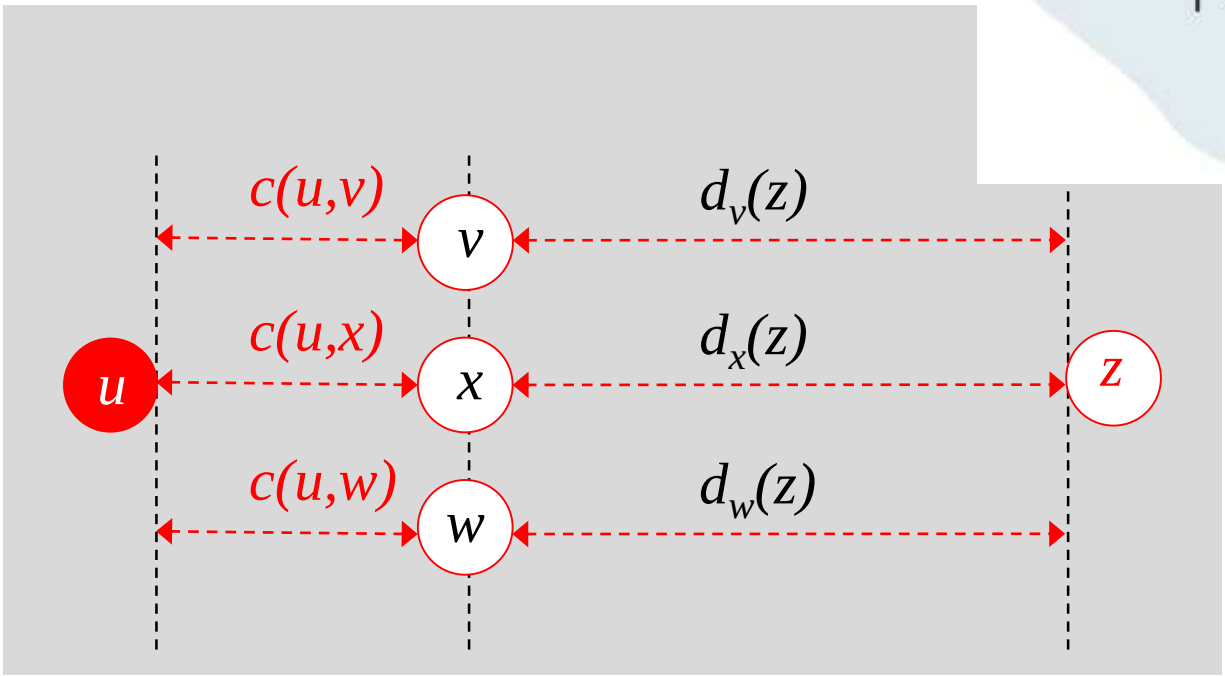


The Distance Vector (DV)

Bellman-Ford

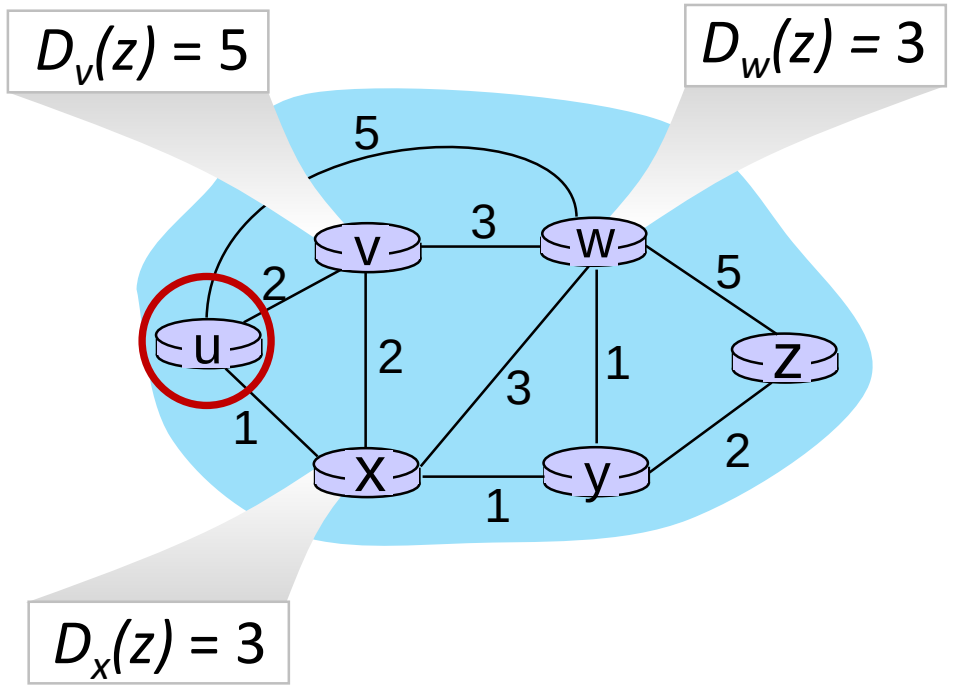
Clearly, $d_v(z) = 5$, $d_x(z) = 3$, $d_w(z) = 3$





Bellman-Ford Example

Suppose that *u*'s neighboring nodes, *x,v,w,* know that for destination *z*:



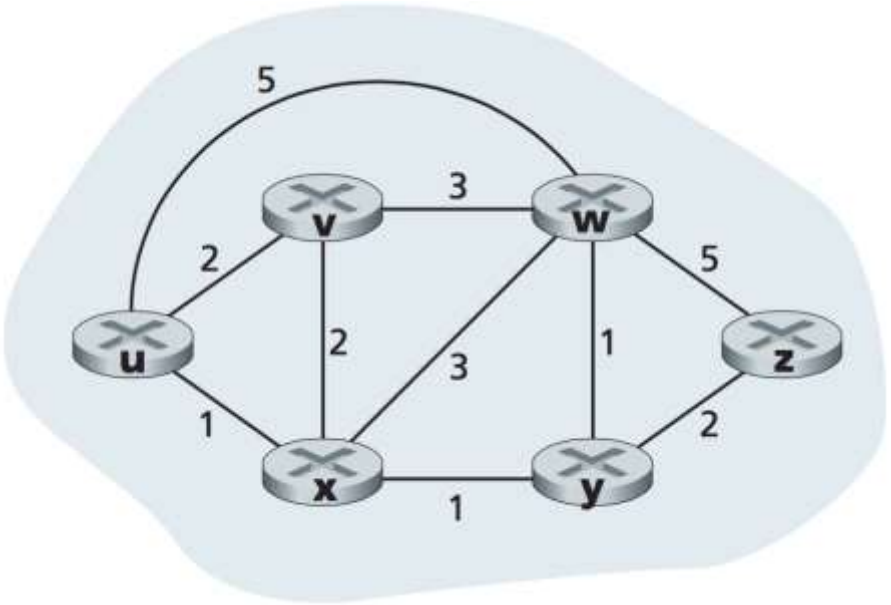
Bellman-Ford equation says:

$$D_u(z) = \min \{ c_{u,v} + D_v(z), \; c_{u,x} + D_x(z), \; c_{u,w} + D_w(z) \}$$
$$= \min \{2 + 5, \; 1 + 3, \; 5 + 3\} = 4$$

node achieving minimum (x) is next hop on estimated least-cost path to destination (z)

Exercise 5.4

Calculate the cost from z to u by using the Bellman-Ford algorithm.



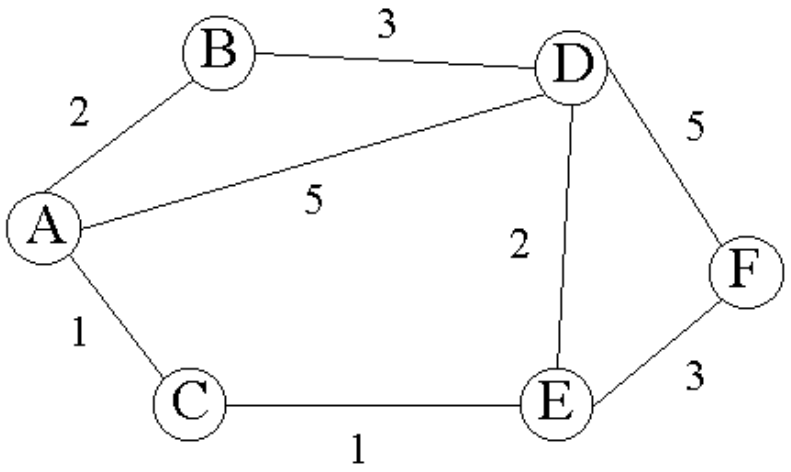
Exercise 5.6

Using the Bellman-Ford algorithm to calculate the cost from:

(a) A to F

(b) B to E

(c) F to B



http://www4.ncsu.edu/~chou/course/Diagrams/distance-vector.gif

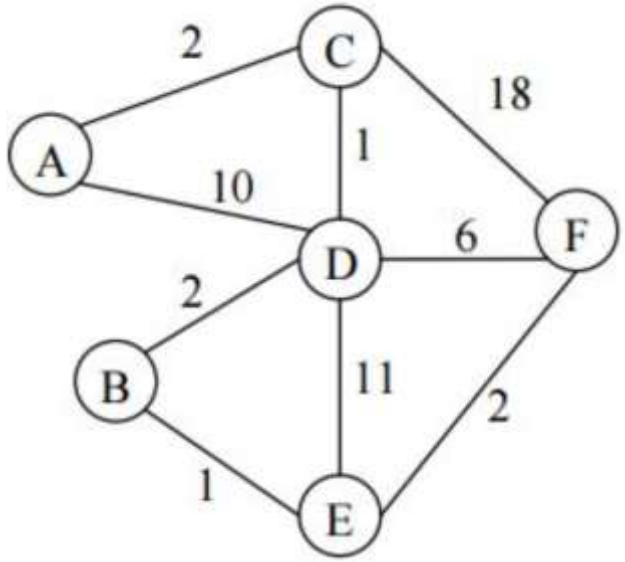
Exercise 5.7

Using the Bellman-Ford algorithm to calculate the cost from:

(a) A to F

(b) B to C

(c) F to B



https://i.stack.imgur.com/bcz2a.jpg

Exercise 5.8

Using the Bellman-Ford algorithm to calculate the cost from:

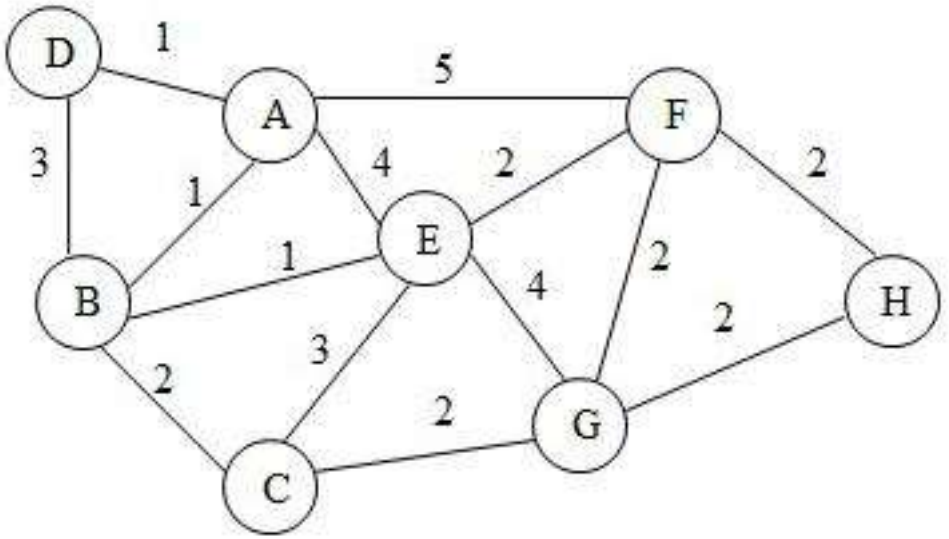
(a) D to E

(b) D to H

(c) H to B

(d) F to D

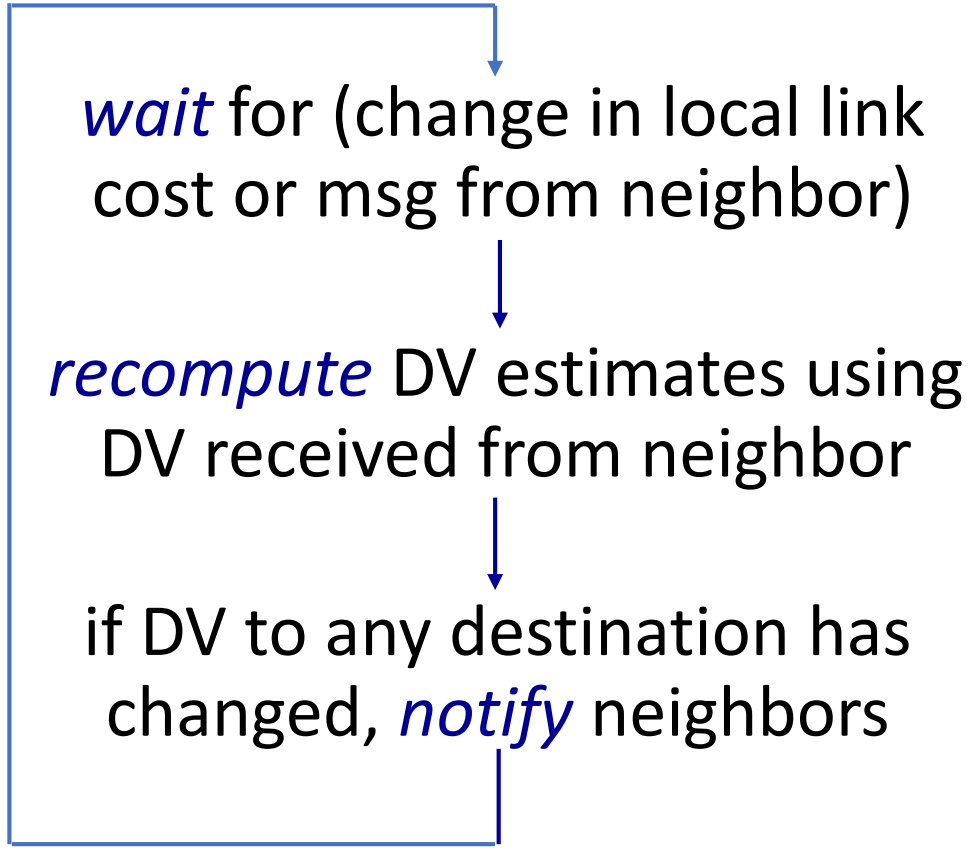
(e) A to E



http://s3.amazonaws.com/answer-board-image/6bfef7b5-05b9-4210-afb5-712510e57a37.jpeg

Distance vector algorithm:

each node:



iterative, asynchronous: each local iteration caused by:

- local link cost change
- DV update message from neighbor

distributed, self-stopping: each node notifies neighbors *only* when its DV changes

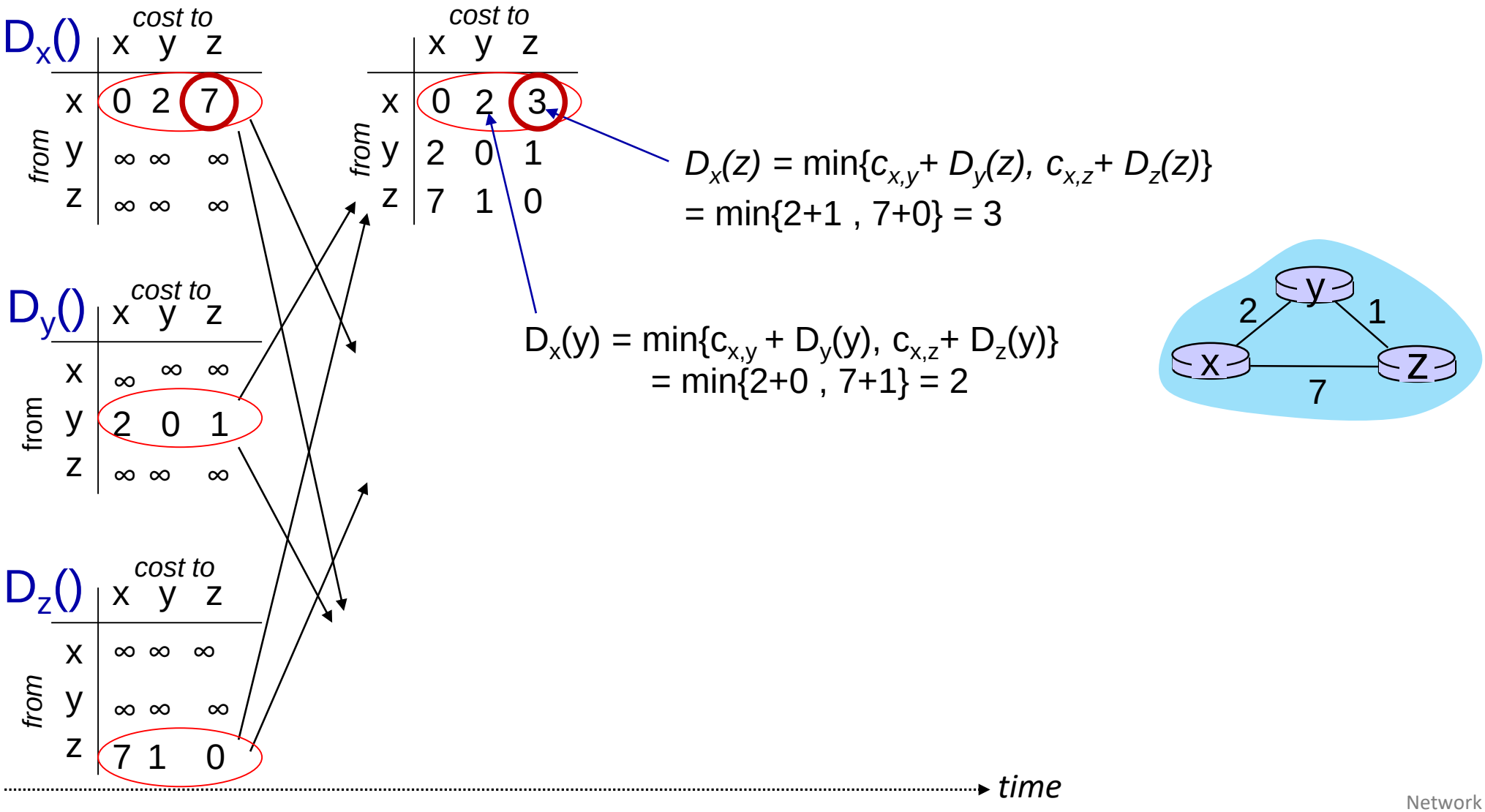
- neighbors then notify their neighbors – *only if necessary*
- no notification received, no actions taken!

The Distance Vector (DV)

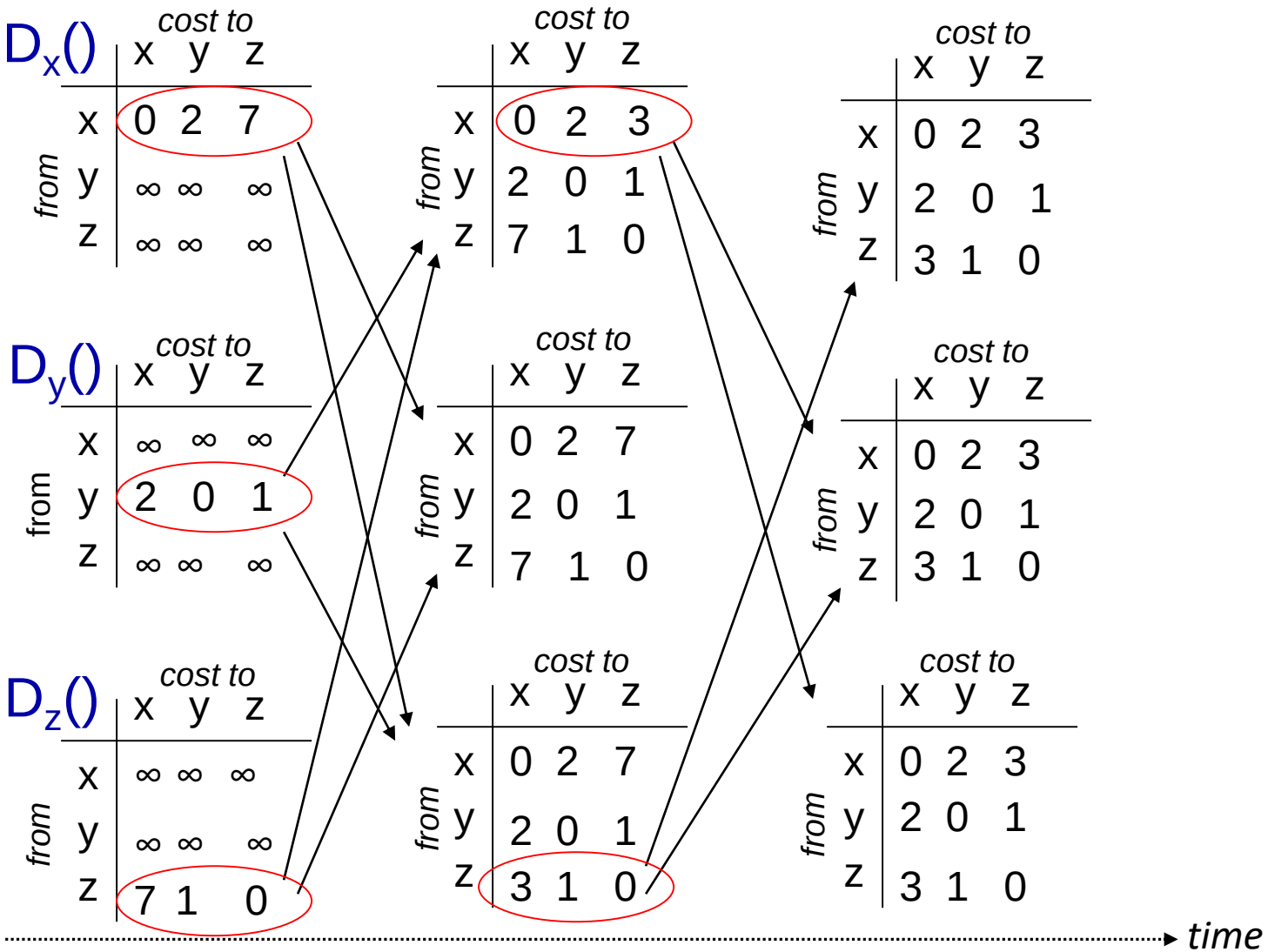
```
1  Initialization:
2     for all destinations y in N:
3        D_x(y) = c(x,y)    /* if y is not a neighbor then c(x,y) = ∞ */
4     for each neighbor w
5        D_w(y) = ? for all destinations y in N
6     for each neighbor w
7        send distance vector D_x = [D_x(y): y in N] to w
8
9  loop
10    wait (until I see a link cost change to some neighbor w or
11          until I receive a distance vector from some neighbor w)
12
13    for each y in N:
14       D_x(y) = min_v{c(x,v) + D_v(y)}
15
16    if D_x(y) changed for any destination y
17       send distance vector D_x = [D_x(y): y in N] to all neighbors
18
19 forever
```

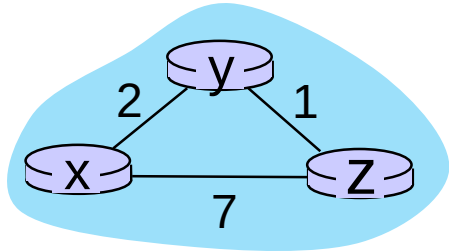
Figure: DV algorithm at each node, x

Distance vector: another example



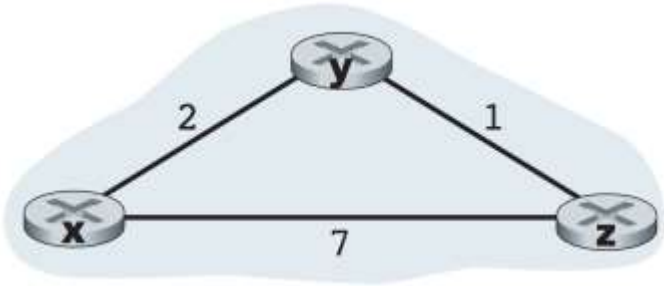
Distance vector: another example





Distance Vector (DV)

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



Cost from x to y:

$$D_x(y) = \min\{c(x,y) + D_y(y), \; c(x,z) + D_z(y)\}$$
$$= \min\{2+0, 7+1\}$$
$$= \min\{2,8\} = 2$$

Cost from z to x:

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

Cost from y to z:

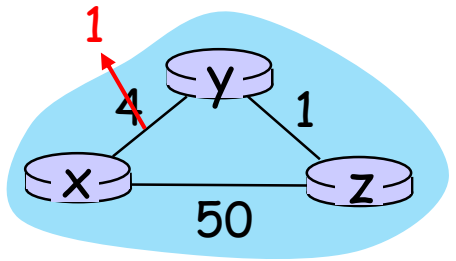
$$D_y(z) = \min\{c(y,z) + D_z(z), \; c(y,x) + D_x(z)\}$$
$$= \min\{1+0, 2+7\}$$
$$= \min\{1,9\} = 1$$

Self-Test

Distance vector: link cost changes

link cost changes:

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



"good news travels fast"

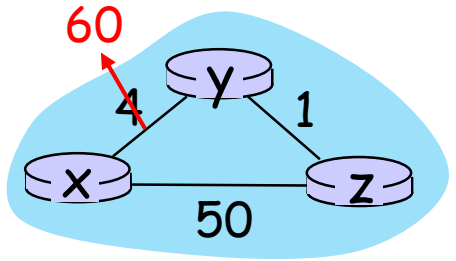
t_0 : *y* detects link-cost change, updates its DV, informs its neighbors.

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:
  - *y* sees direct link to *x* has new cost 60, but z has said it has a path at cost of 5. So *y* computes "my new cost to x will be 6, via z); notifies *z* of new cost of 6 to *x.*
  - *z* learns that path to *x* via *y* has new cost 6, so *z* computes "my new cost to *x* will be 7 via y), notifies *y* of new cost of 7 to *x.*
  - *y* learns that path to *x* via *z* has new cost 7, so *y* computes "my new cost to *x* will be 8 via y), notifies *z* of new cost of 8 to *x.*
  - *z* learns that path to *x* via *y* has new cost 8, so *z* computes "my new cost to *x* will be 9 via y), notifies *y* of new cost of 9 to *x.*

    …
- see text for solutions. *Distributed algorithms are tricky!*

Comparison of LS and DV algorithms

message complexity

LS: n routers, $O(n^2)$ messages sent

DV: exchange between neighbors; convergence time varies

speed of convergence

LS: $O(n^2)$ algorithm, $O(n^2)$ messages

- may have oscillations

DV: convergence time varies

- may have routing loops
- count-to-infinity problem

robustness: what happens if router malfunctions, or is compromised?

LS:

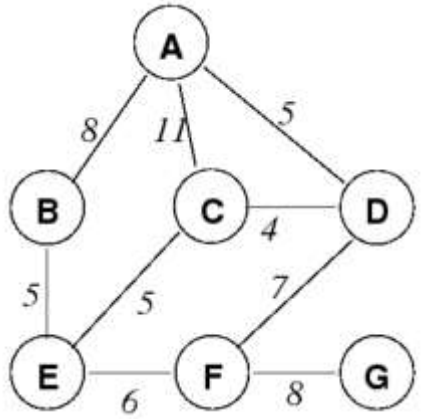
- router can advertise incorrect *link* cost
- each router computes only its *own* table

DV:

- DV router can advertise incorrect *path* cost ("I have a *really* low cost path to everywhere"): black-holing
- each router's table used by others: error propagate thru network

Exercise 5.9

Answer the questions based on the figure and the table with some values that constructed by Bellman-Ford algorithm.



(a) Calculate the cost at *w*, *x* and *y*.

(b) Proof the cost in the blue shaded cells.

(c) Calculate the cost at *z*. Justify your answer.

cost to

from	A	B	C	D	E	F	G
A					*w*		
B				*x*			
C	*y*						
D		13					
E					9		
F			11				
G	*z*						

http://homepages.herts.ac.uk/~comqrgd/docs/network-notes/network-notes-img31.png

Network layer: "control plane" roadmap

- introduction
- routing protocols
- intra-ISP routing: OSPF
- routing among ISPs: BGP
- SDN control plane
- Internet Control Message Protocol (ICMP)

- network management, configuration
  - SNMP
  - NETCONF/YANG (NOT COVERED)

Making routing scalable

our routing study thus far - idealized

- all routers identical
- network “flat”

… not true in practice

scale: billions of destinations:

- can’t store all destinations in routing tables!
- routing table exchange would swamp links!

administrative autonomy:

- Internet: a 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")

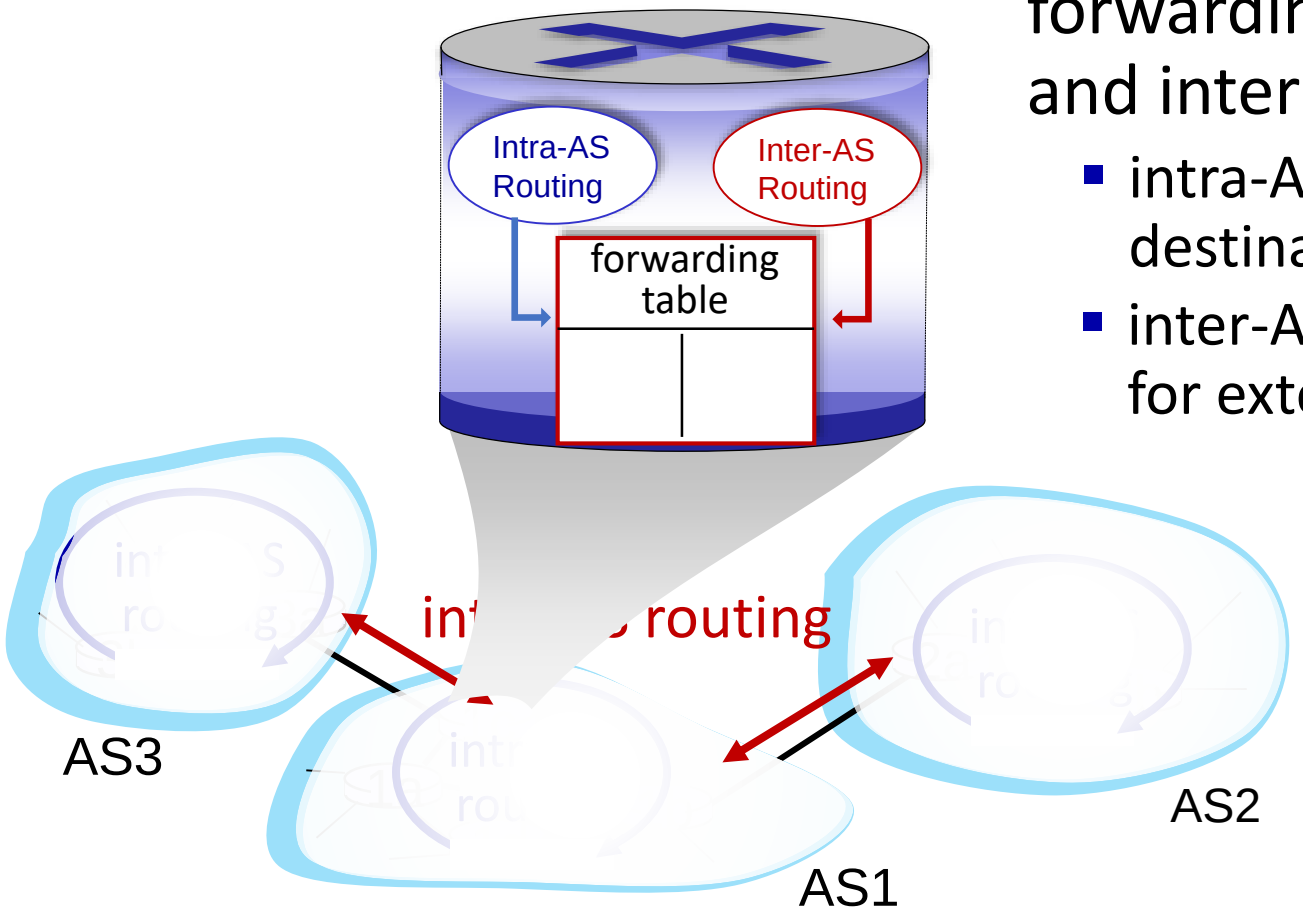
intra-AS (aka "intra-domain"): routing among *within same AS ("network")*

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

inter-AS (aka "inter-domain"): routing *among* AS'es

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

Interconnected ASes



forwarding table configured by intra- and inter-AS routing algorithms

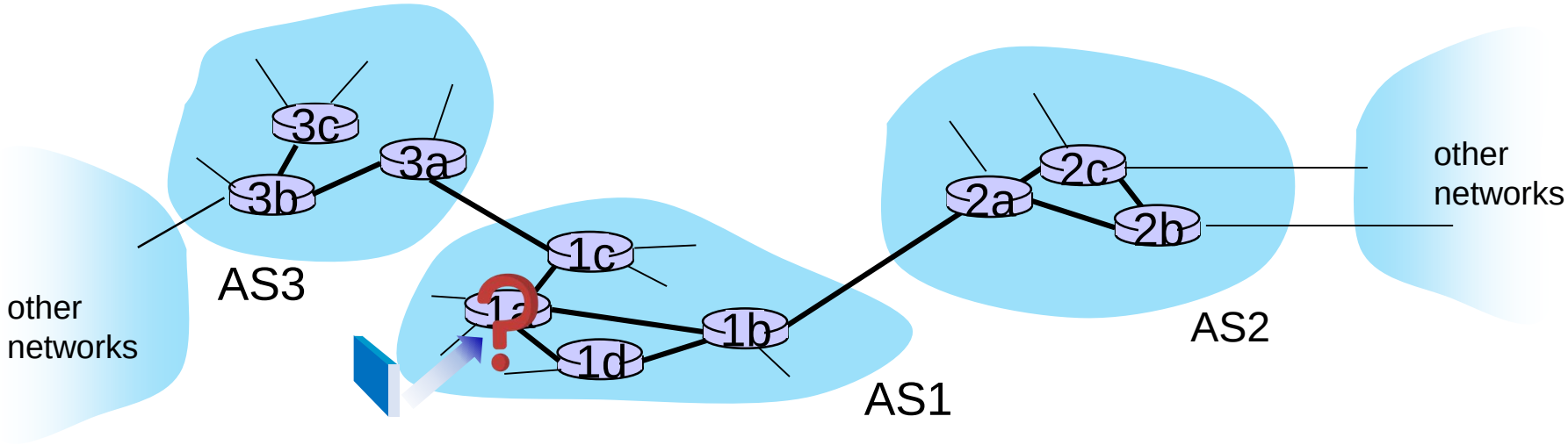
- intra-AS routing determine entries for destinations within AS
- inter-AS & intra-AS determine entries for external destinations

Inter-AS routing: a role in intradomain forwarding

- suppose router in AS1 receives datagram destined outside of AS1:
  - router should forward packet to gateway router in AS1, but which one?

AS1 inter-domain routing must:

1. learn which destinations reachable through AS2, which through AS3
2. propagate this reachability info to all routers in AS1



Intra-AS routing: routing within an AS

also known as interior gateway protocols (IGP)

most common intra-AS routing protocols:

- RIP: Routing Information Protocol [RFC 1723]
  - classic DV: DVs exchanged every 30 secs
  - no longer widely used
- IGRP/EIGRP: Enhanced Interior Gateway Routing Protocol
  - DV based
  - formerly Cisco-proprietary for decades (became open in 2013 [RFC 7868])
- OSPF: Open Shortest Path First [RFC 2328]
  - link-state routing
  - IS-IS protocol (ISO standard, not RFC standard) essentially same as OSPF

OSPF (Open Shortest Path First) routing

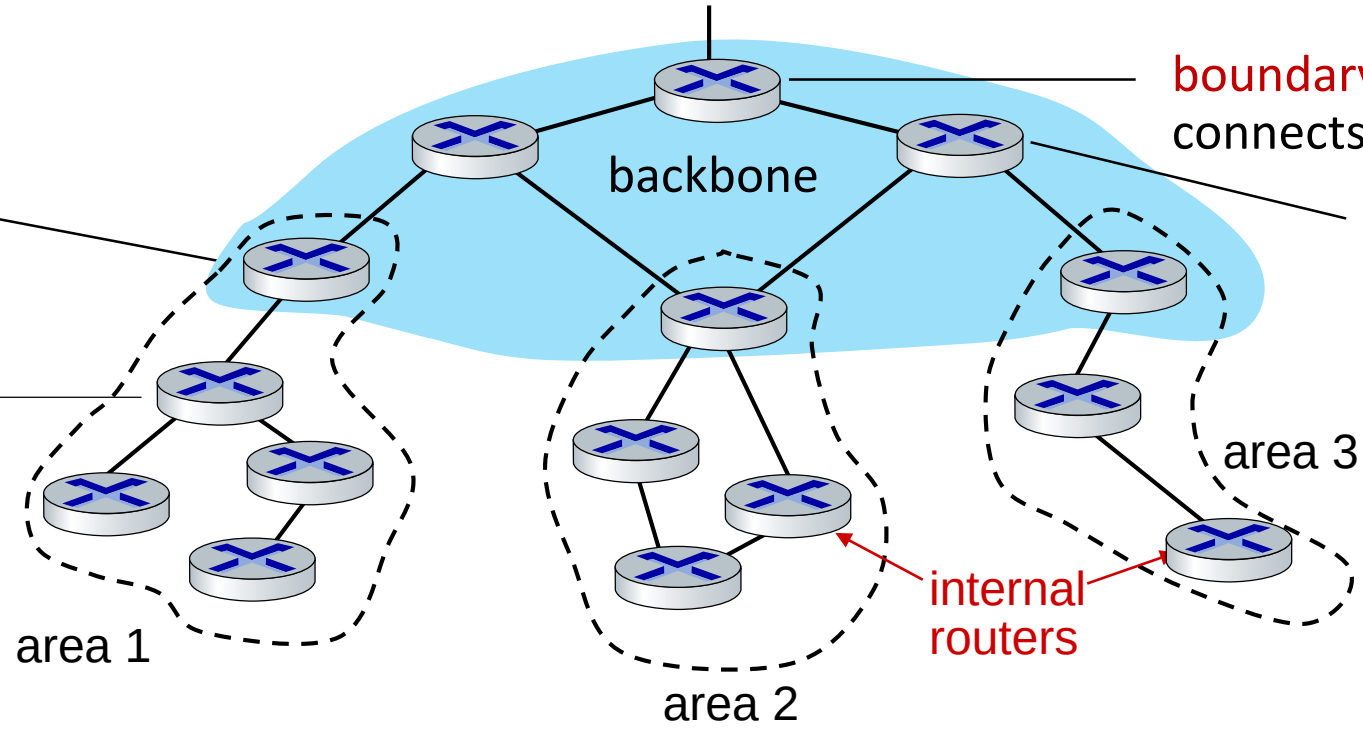
- "open": publicly available
- classic link-state
  - each router floods OSPF link-state advertisements (directly over IP rather than using TCP/UDP) to all other routers in entire AS
  - multiple link costs metrics possible: bandwidth, delay
  - each router has full topology, uses Dijkstra's algorithm to compute forwarding table
- *security:* all OSPF messages authenticated (to prevent malicious intrusion)

Hierarchical OSPF

- two-level hierarchy: local area, backbone.
  - link-state advertisements flooded only in area, or backbone
  - each node has detailed area topology; only knows direction to reach other destinations

area border routers:
"summarize" distances to destinations in own area, advertise in backbone

local routers:
- flood LS in area only
- compute routing within area
- forward packets to outside via area border router



boundary router:
connects to other ASes

backbone router:
runs OSPF limited to backbone

Network layer: "control plane" roadmap

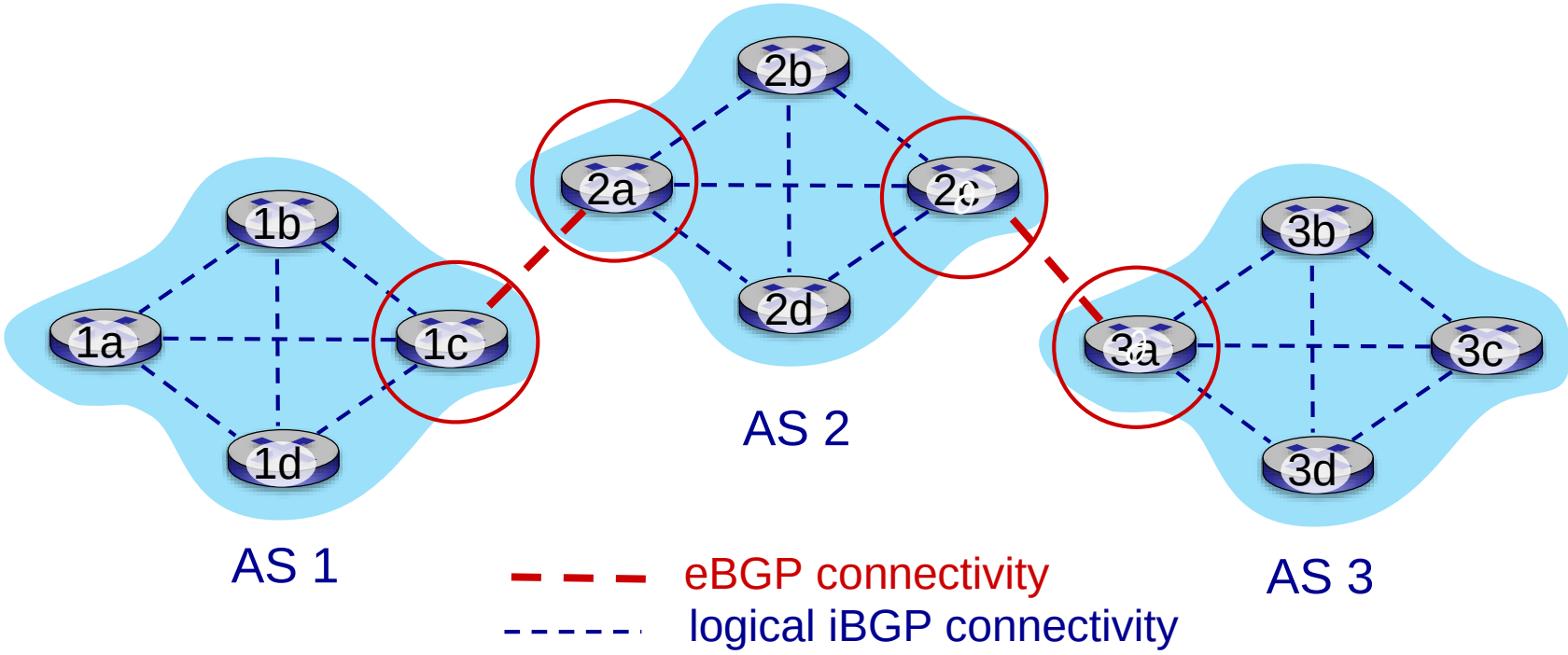
- introduction
- routing protocols
- intra-ISP routing: OSPF
- routing among ISPs: BGP
- SDN control plane
- Internet Control Message Protocol (ICMP)

- network management, configuration
  - SNMP
  - NETCONF/YANG (NOT COVERED)

Internet inter-AS routing: BGP

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

eBGP, iBGP connections



gateway routers run both eBGP and iBGP protocols

Why different Intra-, Inter-AS routing ?

policy:

- inter-AS: admin wants control over how its traffic routed, who routes through its network
- intra-AS: single admin, so policy less of an issue

scale:

- hierarchical routing saves table size, reduced update traffic

performance:

- intra-AS: can focus on performance
- inter-AS: policy dominates over performance

Network layer: "control plane" roadmap

- introduction
- routing protocols
- intra-ISP routing: OSPF
- routing among ISPs: BGP
- SDN control plane
- Internet Control Message Protocol (ICMP)

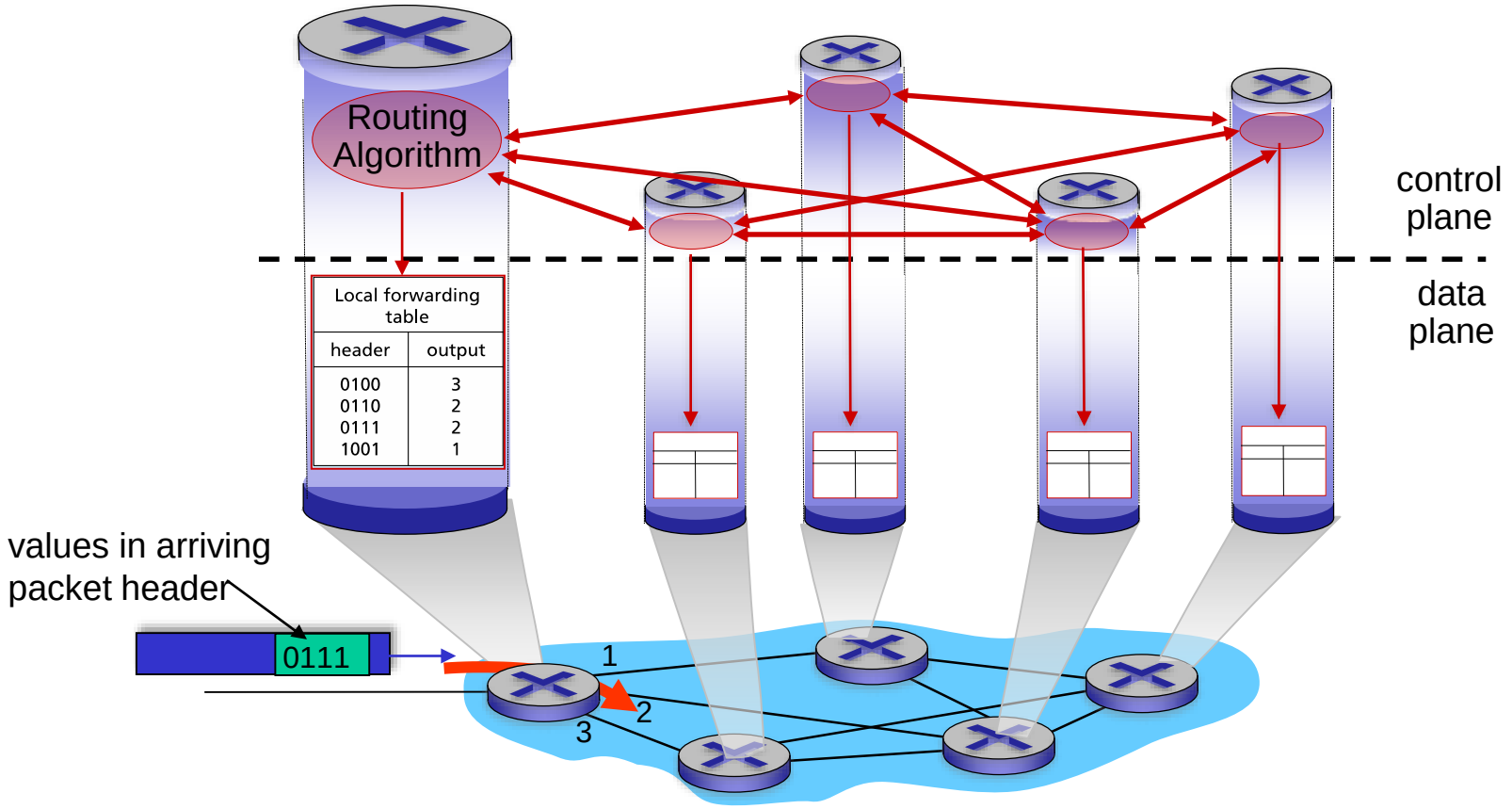
- network management, configuration
  - SNMP
  - NETCONF/YANG (NOT COVERED)

Software defined networking (SDN)

- Internet network layer: historically implemented via distributed, per-router control 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

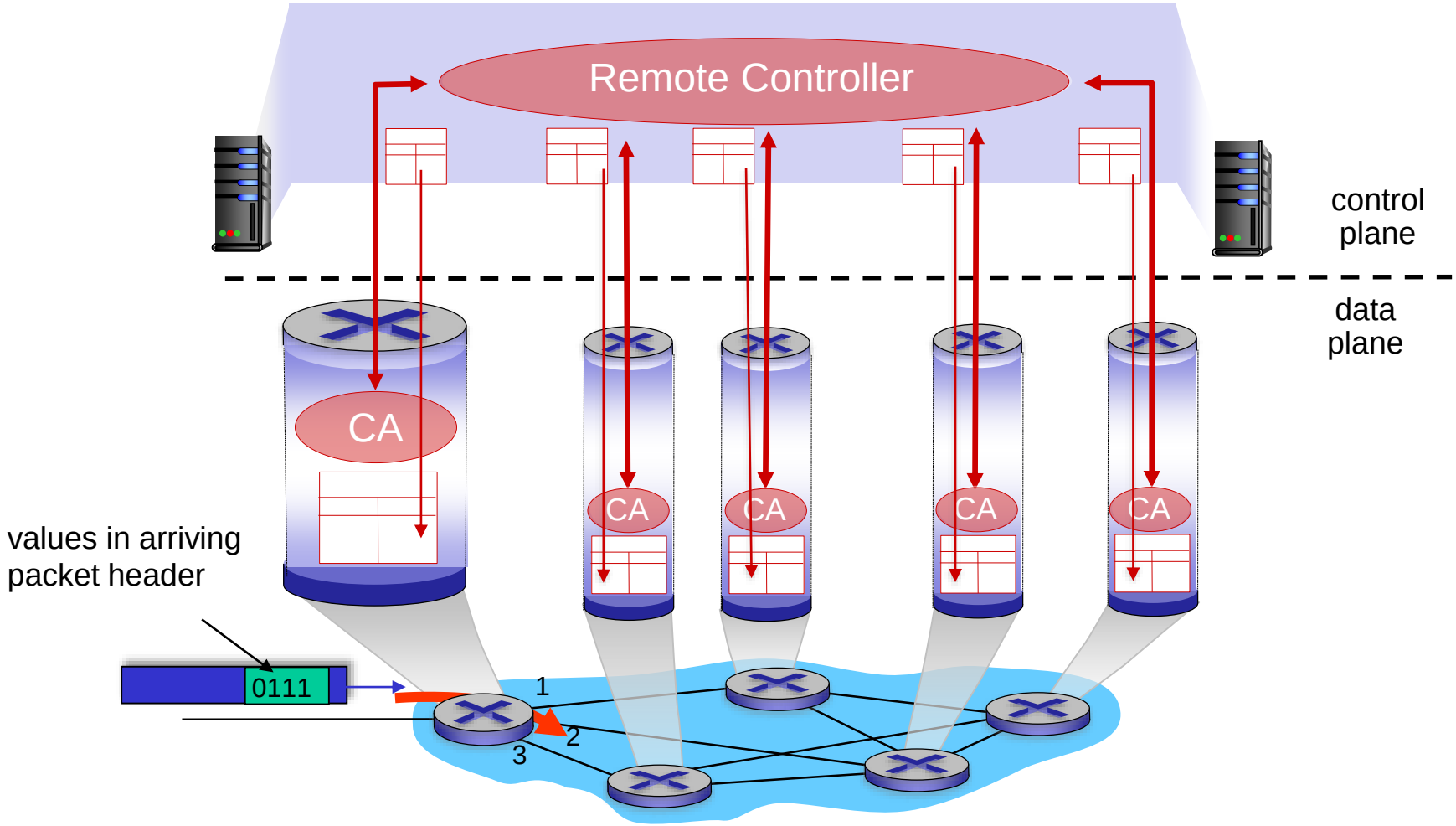
Per-router control plane

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



Software-Defined Networking (SDN) control plane

Remote controller computes, installs forwarding tables in routers

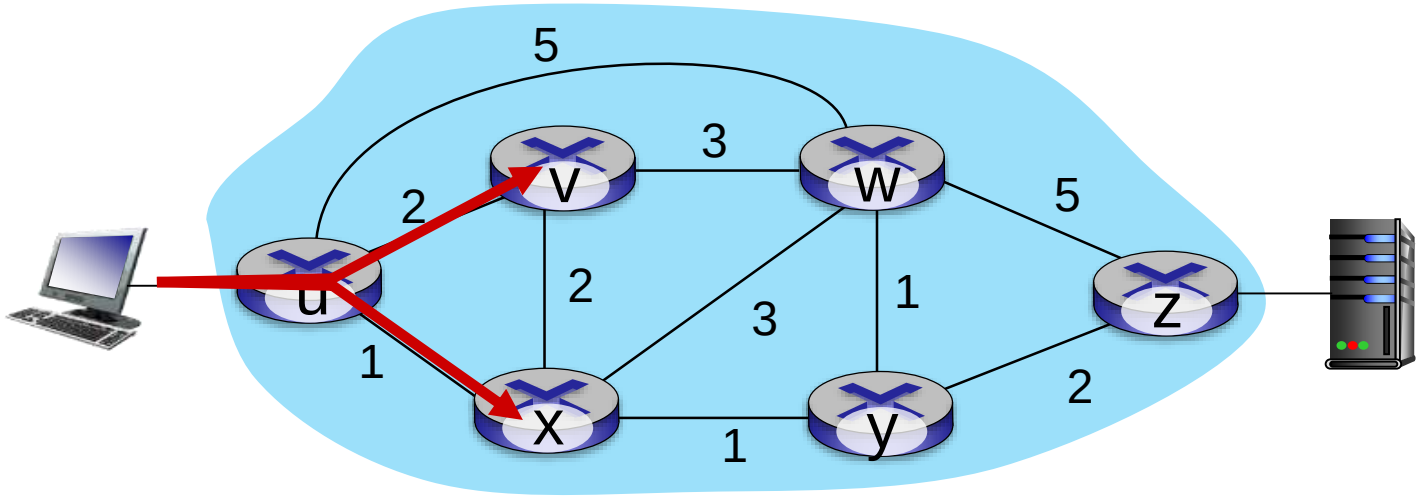


Software defined networking (SDN)

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
  - foster innovation: let 1000 flowers bloom

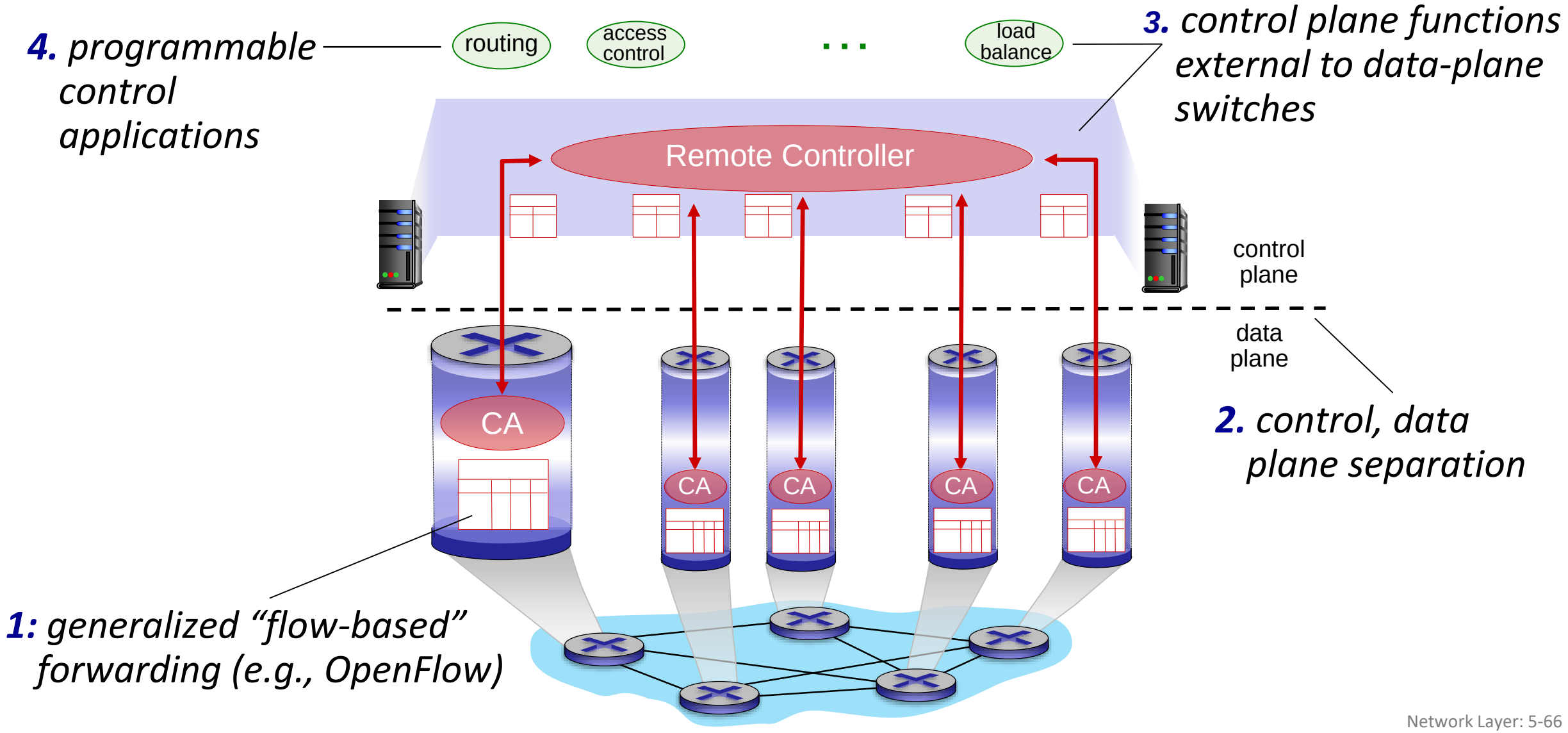
Traffic engineering: difficult with traditional routing



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)

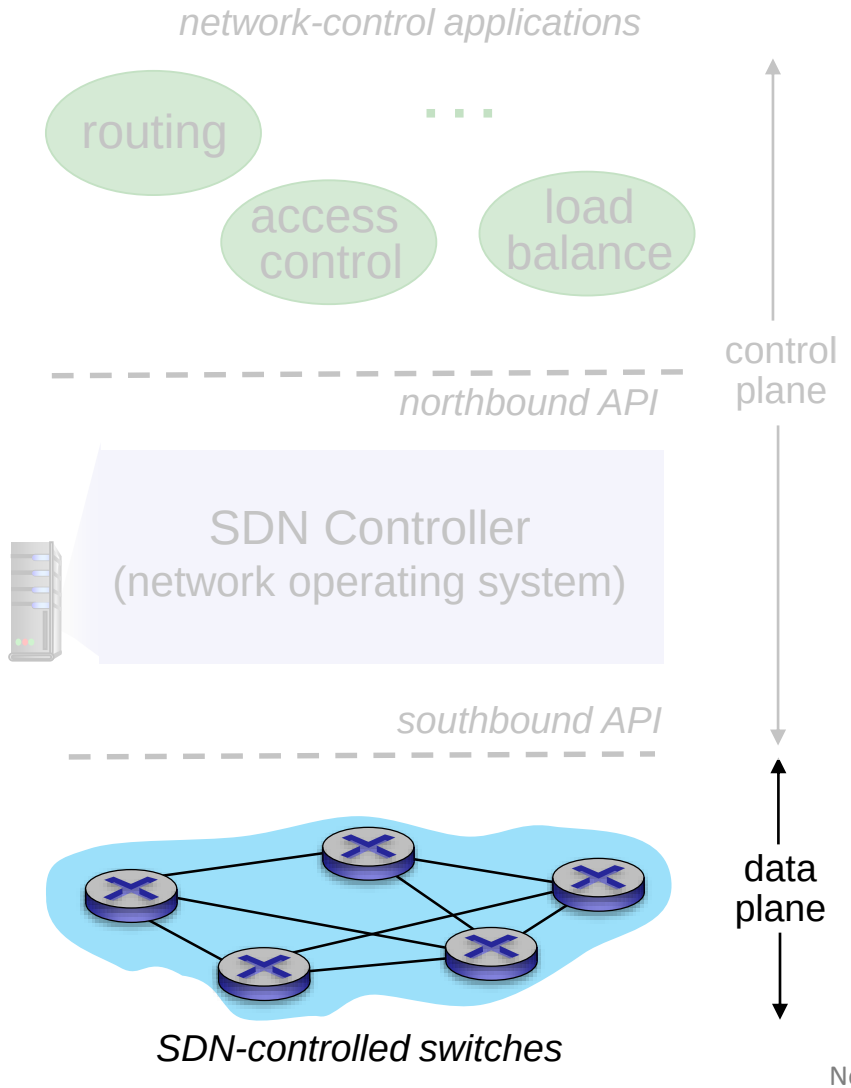
Software defined networking (SDN)



Software defined networking (SDN)

Data-plane switches:

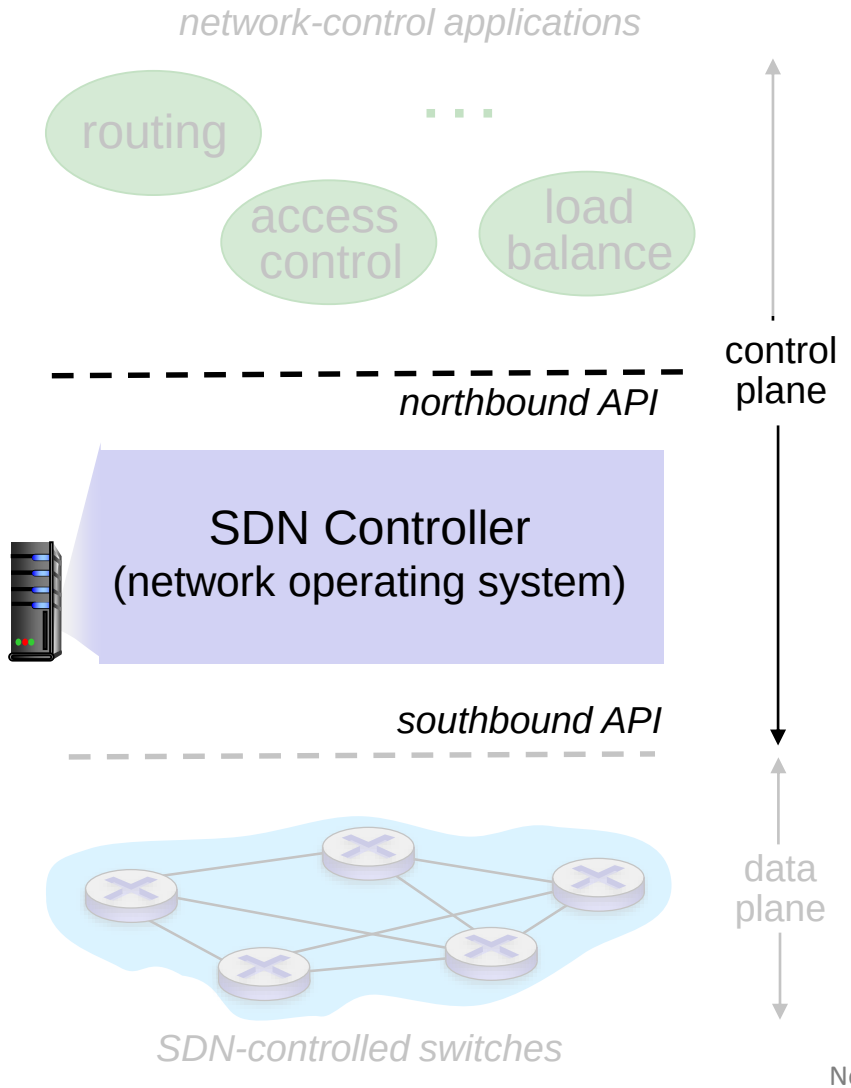
- fast, simple, commodity switches implementing generalized data-plane forwarding in hardware
- flow (forwarding) table computed, installed under controller supervision
- API for table-based switch control (e.g., OpenFlow)
  - defines what is controllable, what is not
- protocol for communicating with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



Software defined networking (SDN)

SDN controller (network OS):

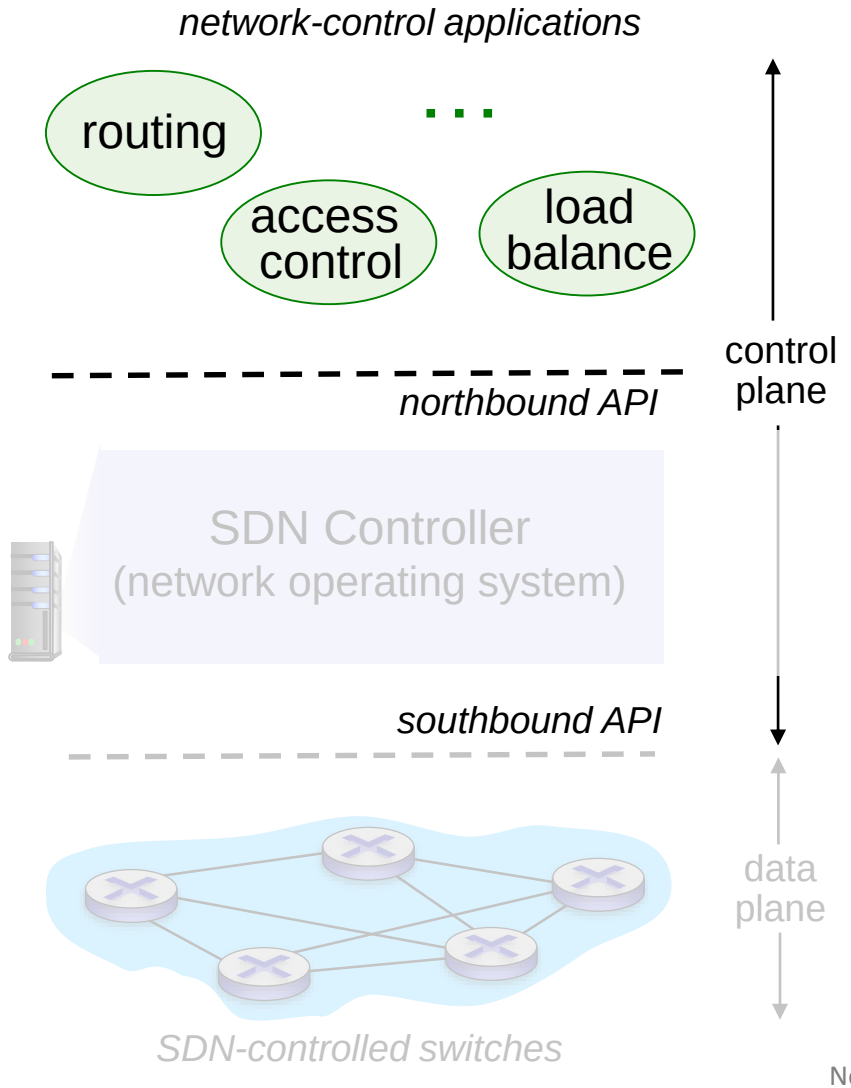
- maintain network state information
- interacts with network control applications “above” via northbound API
- interacts with network switches “below” via southbound API
- implemented as distributed system for performance, scalability, fault-tolerance, robustness



Software defined networking (SDN)

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

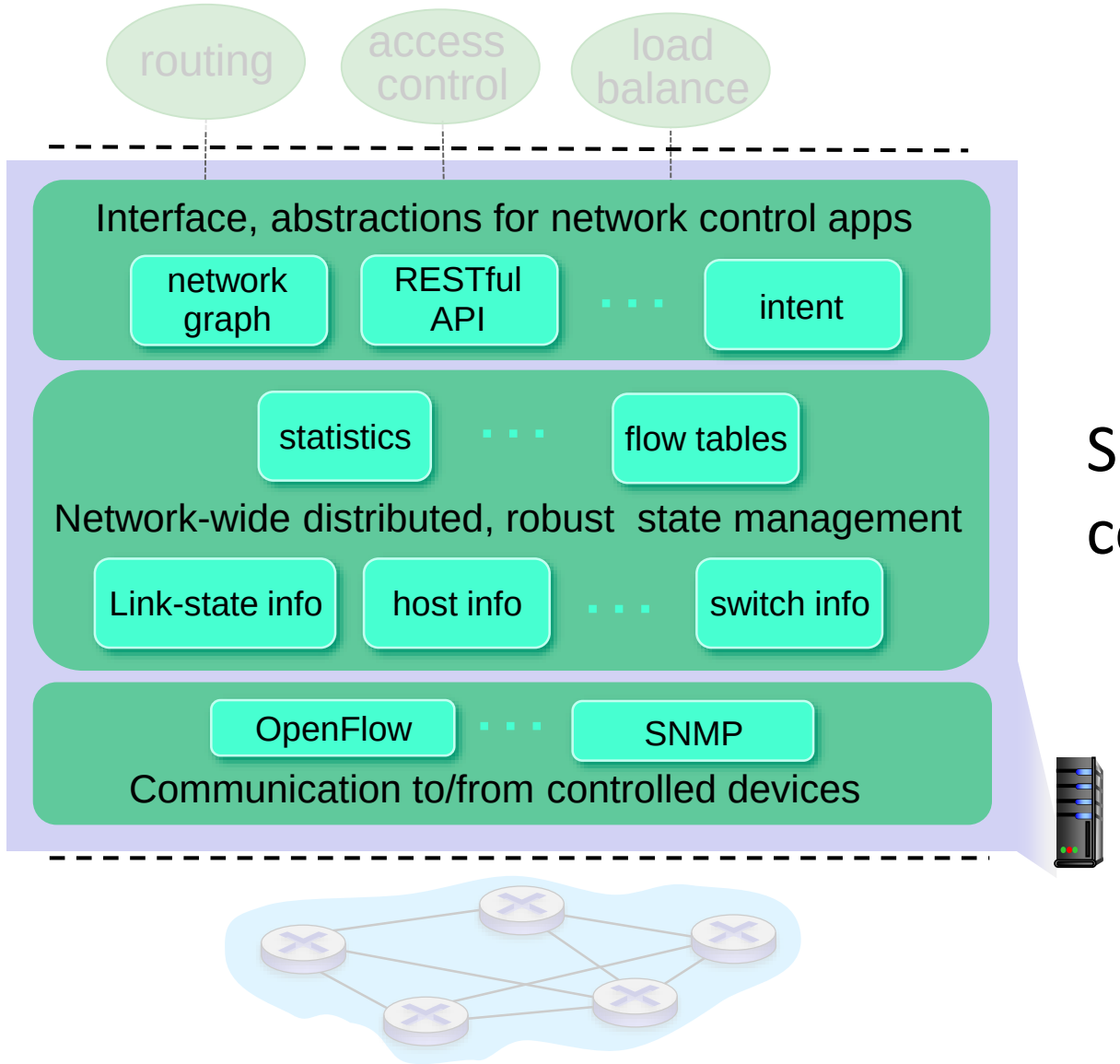


Components of SDN controller

interface layer to network control apps: abstractions API

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

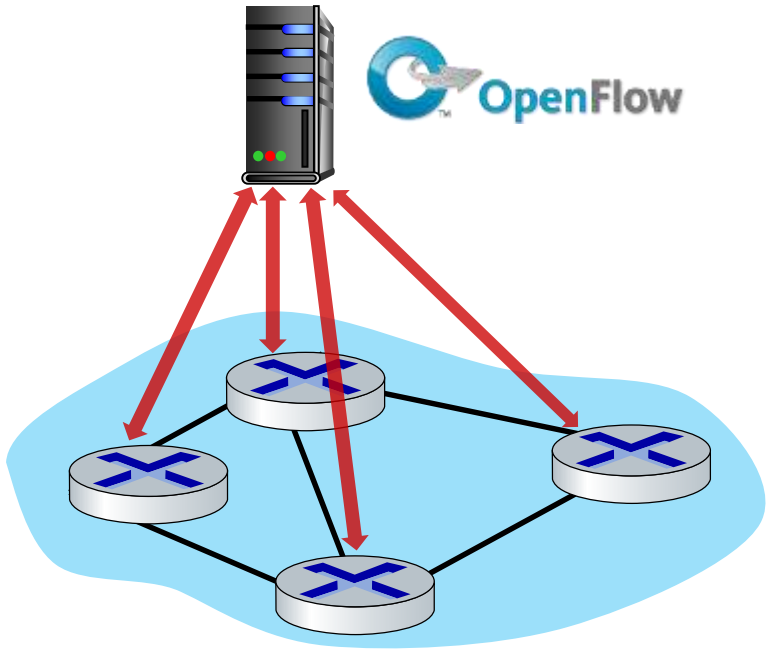
communication: communicate between SDN controller and controlled switches



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.)
- distinct from OpenFlow API
  - API used to specify generalized forwarding actions

OpenFlow Controller

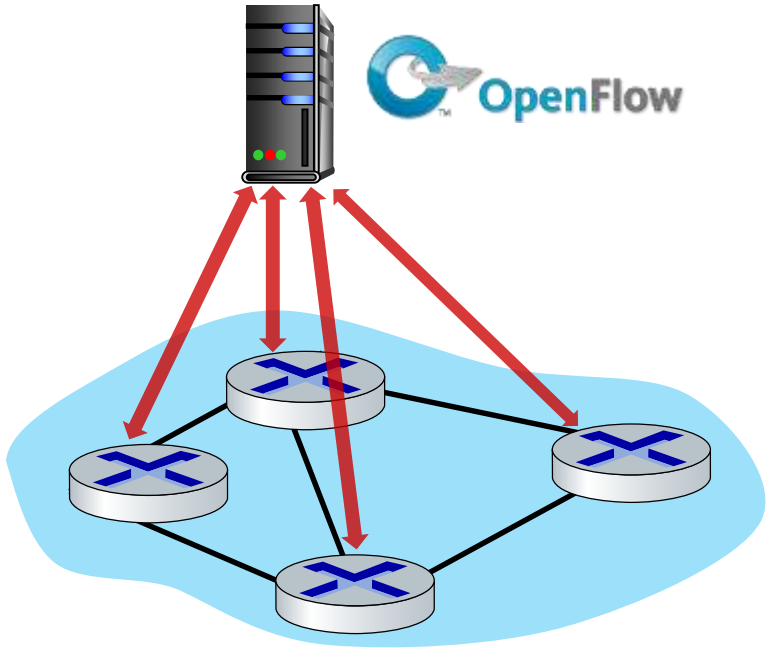


OpenFlow: controller-to-switch messages

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

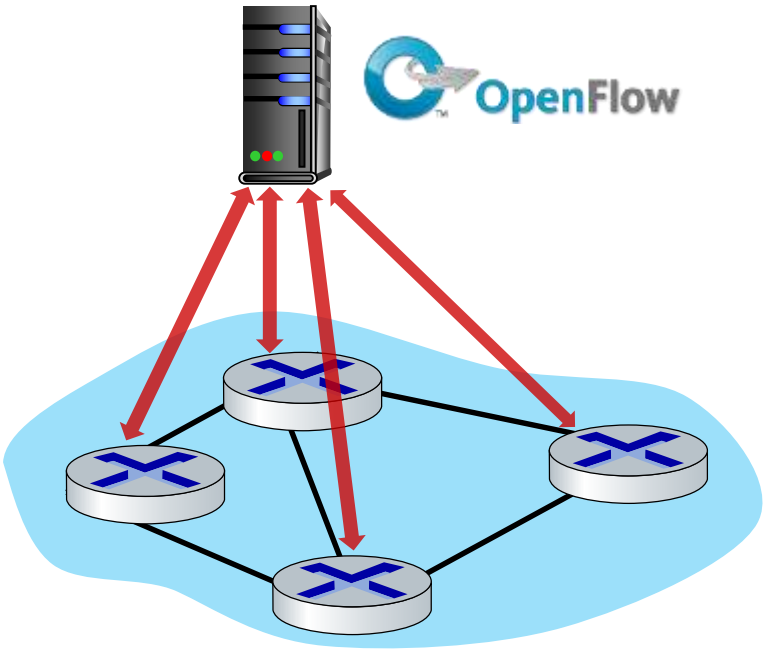


OpenFlow: switch-to-controller messages

Key switch-to-controller messages

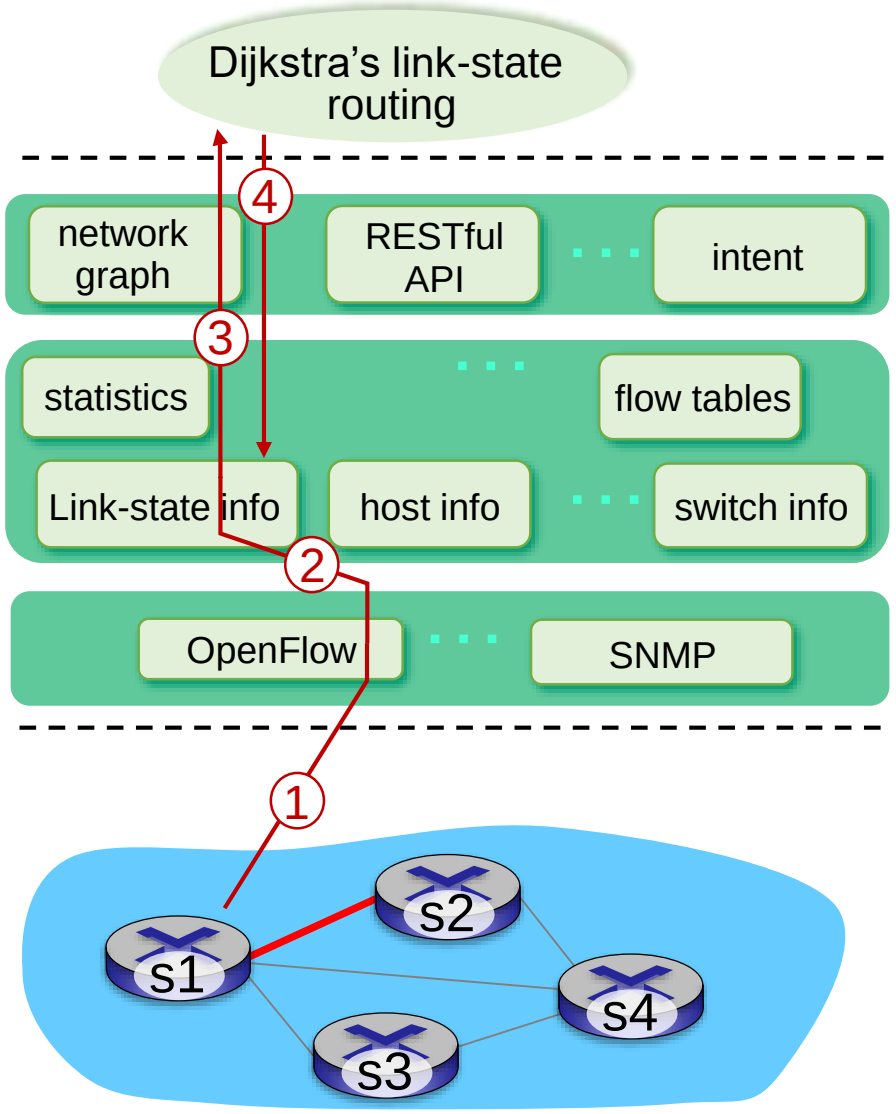
- *packet-in:* transfer packet (and its control) to controller. See packet-out message from controller
- *flow-removed:* flow table entry deleted at switch
- *port status:* inform controller of a change on a port.

OpenFlow Controller



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



① S1, experiencing link failure uses 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: 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: beyond a single AS
- SDN critical in 5G cellular networks

SDN and the future of traditional network protocols

- SDN-computed versus router-computer forwarding tables:
  - just one example of logically-centralized-computed versus protocol computed
- one could imagine SDN-computed congestion control:
  - controller sets sender rates based on router-reported (to controller) congestion levels

How will implementation of network functionality (SDN versus protocols) evolve?

Network layer: "control plane" roadmap

- introduction
- routing protocols
- intra-ISP routing: OSPF
- routing among ISPs: BGP
- SDN control plane
- Internet Control Message Protocol (ICMP)

- network management, configuration
  - SNMP
  - NETCONF/YANG (NOT COVERED)

ICMP: internet control message protocol

- used by hosts & routers to communicate network-level information
  - error reporting: unreachable host, network, port, protocol
  - echo request/reply (used by ping)
- network-layer "above" IP:
  - ICMP msgs carried in IP datagrams
- ICMP message: type, code plus first 8 bytes of IP datagram causing error

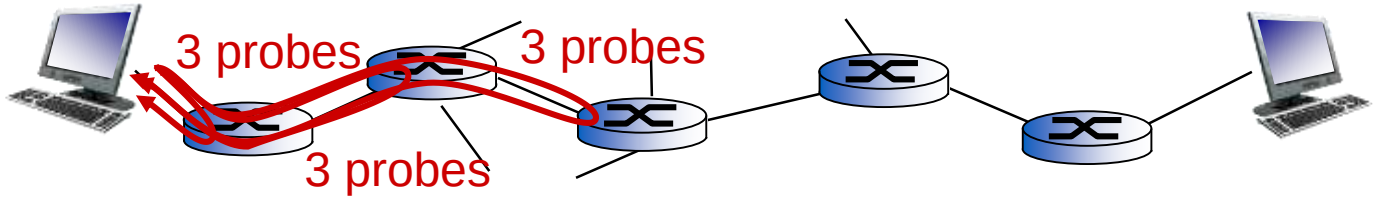
Type	Code	description
0	0	echo reply (ping)
3	0	dest. network unreachable
3	1	dest host unreachable
3	2	dest protocol unreachable
3	3	dest port unreachable
3	6	dest network unknown
3	7	dest host unknown
4	0	source quench (congestion control - not used)
8	0	echo request (ping)
9	0	route advertisement
10	0	router discovery
11	0	TTL expired
12	0	bad IP header

Traceroute and ICMP

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

stopping criteria:

- UDP segment eventually arrives at destination host
- destination returns ICMP "port unreachable" message (type 3, code 3)
- source stops



ICMPv6 (RFC 4443)

- A new version of ICMP has been defined by IPv6 in RFC4443.
- In addition to reorganizing the existing ICMP type and code definitions, ICMPv6 also added new types and codes required by the new IPv6 functionality.
- These included the "Packet Too Big" type and an "unrecognized IPv6 options" error code.
- The frame format has 3 fields (Type – 8bits, Code – 8 bits, Checksum – 16 bits)

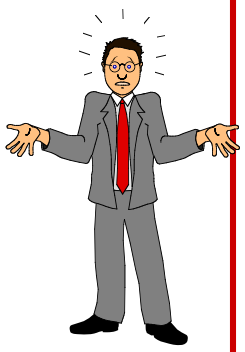
Network layer: "control plane" roadmap

- introduction
- routing protocols
- intra-ISP routing: OSPF
- routing among ISPs: BGP
- SDN control plane
- Internet Control Message Protocol (ICMP)

- network management, configuration
  - SNMP
  - NETCONF/YANG (NOT COVERED)

What is network management?

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

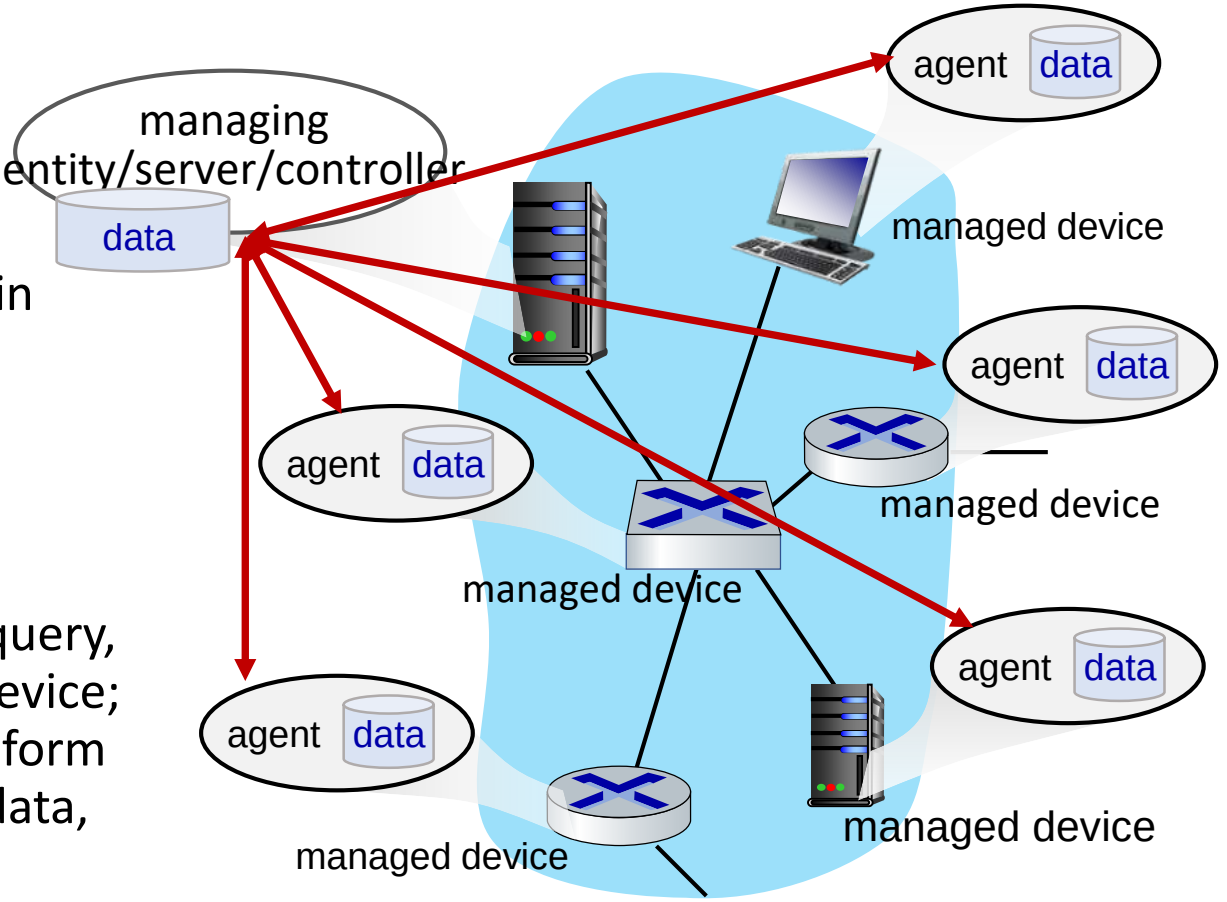


"Network management includes the deployment, integration and coordination of the hardware, software, and human elements to monitor, test, poll, configure, analyze, evaluate, and control the network and element resources to meet the real-time, operational performance, and Quality of Service requirements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

Components of network management

Managing entity/server: application, typically with network managers (humans) in the loop

Network management protocol: used by managing server to query, configure, manage device; used by devices to inform managing server of data, events.



Managed device:

- equipment with manageable, configurable hardware, software components
- managed devices contain managed objects whose data is gathered into a Management Information Base (MIB)

Data: device "state" configuration data, operational data, device statistics

Network operator approaches to management

CLI (Command Line Interface)

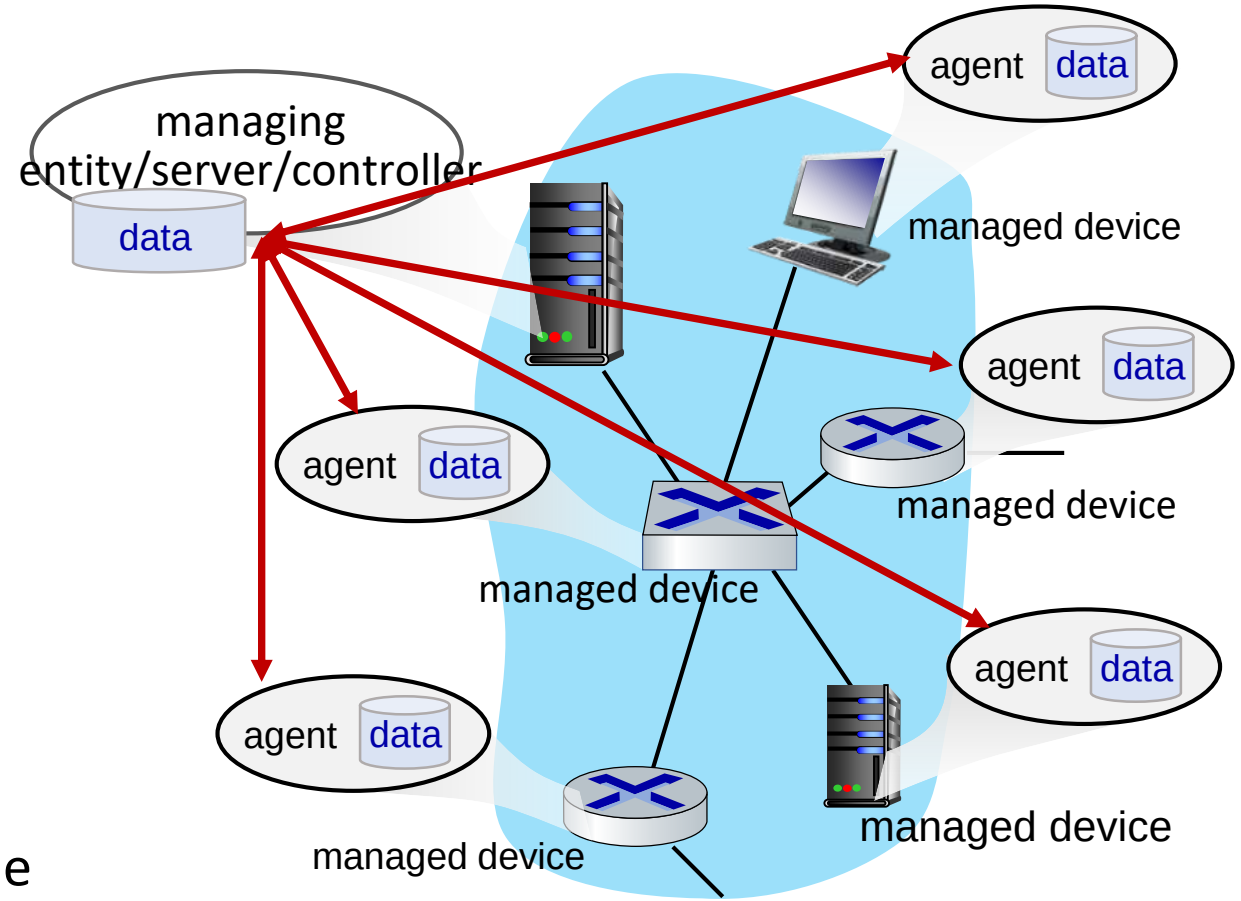
- operator issues (types, scripts) direct to individual devices (e.g., vis ssh)

SNMP/MIB

- operator queries/sets devices data (MIB) using Simple Network Management Protocol (SNMP)

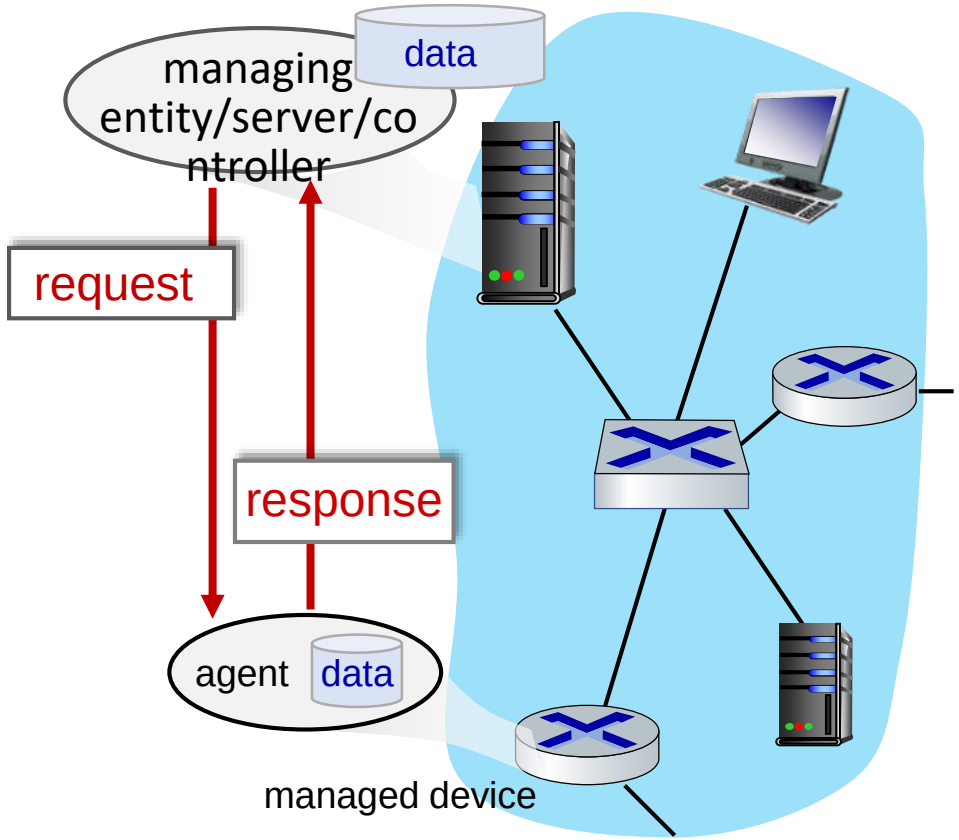
NETCONF/YANG

- more abstract, network-wide, holistic
- emphasis on multi-device configuration management.
- YANG: data modeling language
- NETCONF: communicate YANG-compatible actions/data to/from/among remote devices

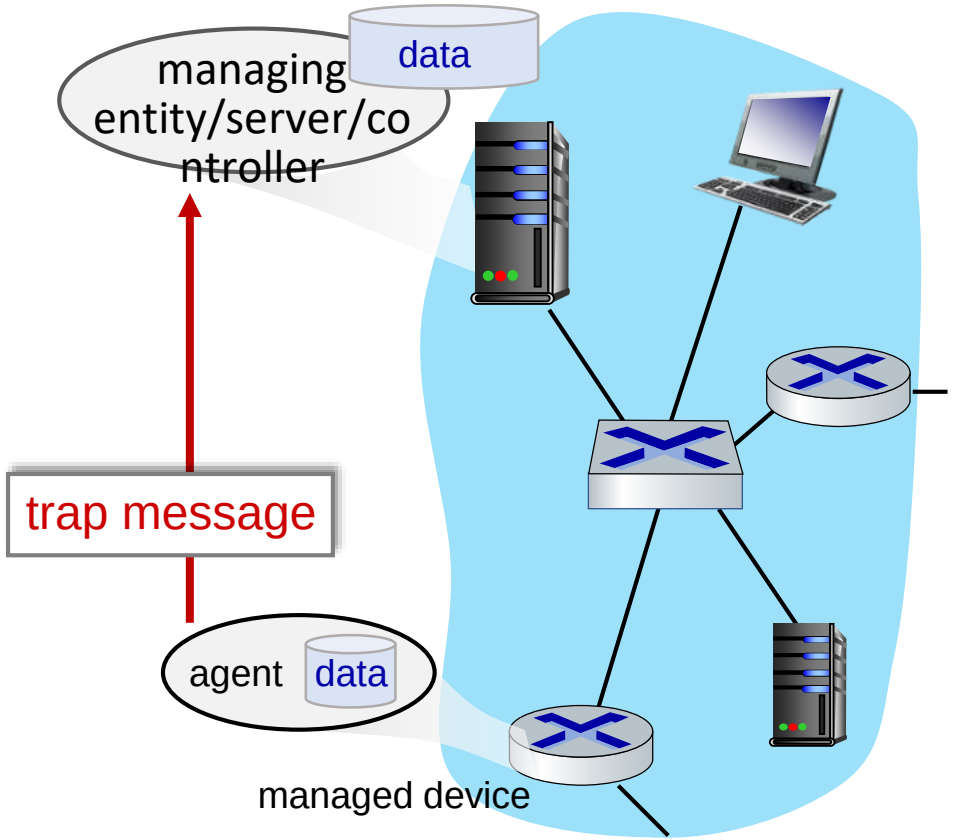


SNMP protocol

Two ways to convey MIB info, commands:



request/response mode



trap mode

SNMP Protocol (Message Types)

Table: SNMPv3 PDU types

	Message	Sender-Receiver	Function

1
2
3
	`GetRequest`
	`GetNextRequest`
	`GetBulkRequest`
		Manager-to-Agent	"*Get me data*"
			(data instance, next data in list, block of data)

4	`InformRequest`	Manager-to-Manager	Here's MIB value
5	`SetRequest`	Manager-to-Agent	Set MIB value
6	`Response`	Agent-to-Manager	Value, response to `Request`.
		Manager-to-Manager	
7	`SNMPv2-Trap`	Agent-to-Manager	Inform manager of exceptional event

Protocol Data Unit (PDU)

Kurose, J.F. and Ross, K.W. (2021). Computer Networking: A Top-Down Approach (8th Edition). Pearson Education Limited, England. (pp. 429)

SNMP Protocol (Message Formats)

message types 0-3

message type 4

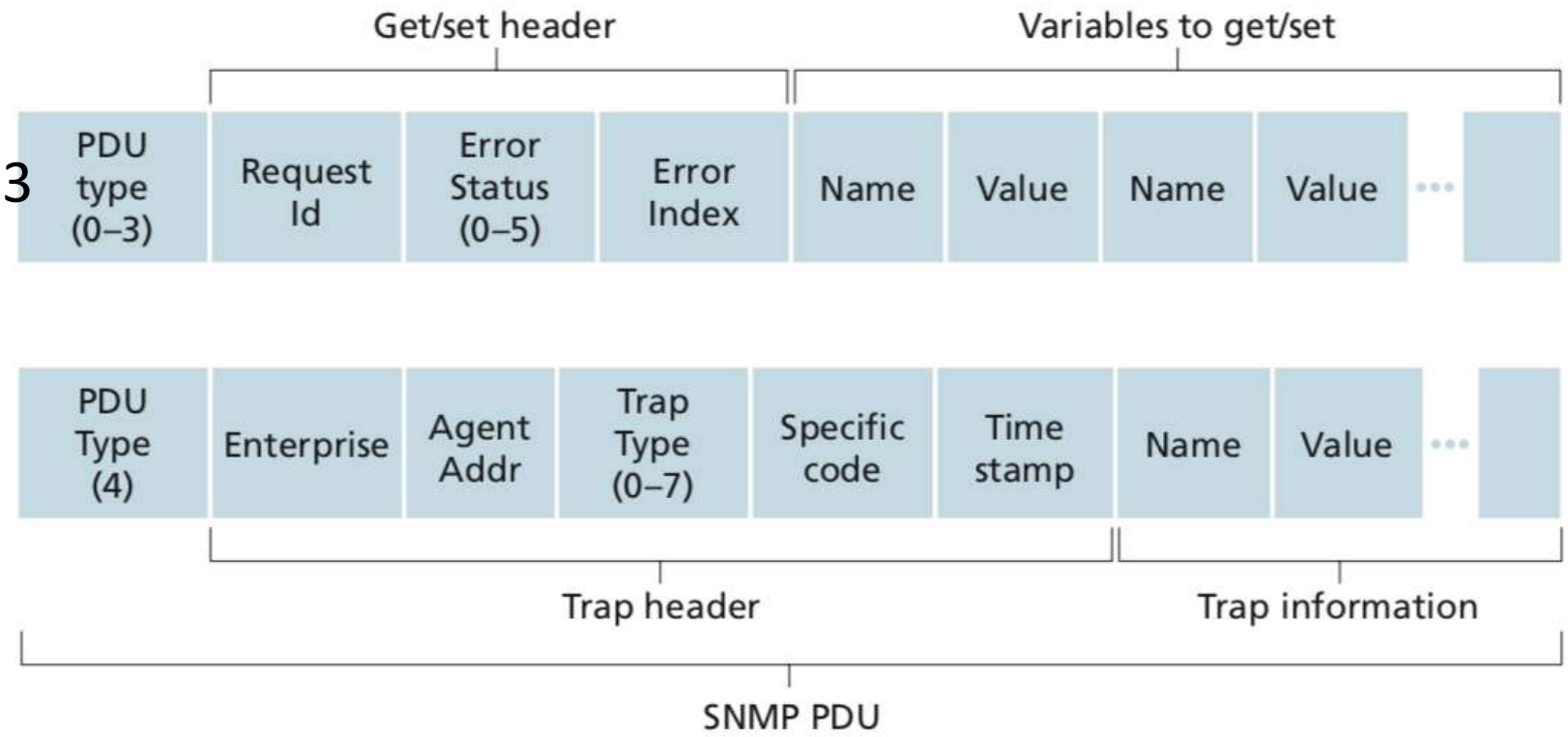


Figure: SNMP PDU format

Protocol Data Unit (PDU)

Kurose, J.F. and Ross, K.W. (2021). Computer Networking: A Top-Down Approach (8th Edition). Pearson Education Limited, England. (pp. 430)

SNMP: Management Information Base (MIB)

- managed device's operational (and some configuration) data

- gathered into device MIB module
  - 400 MIB modules defined in RFC's; many more vendor-specific MIBs
- Structure of Management Information (SMI): data definition language
- example MIB variables for UDP protocol:

Object ID	Name	Type	Comments
1.3.6.1.2.1.7.1	UDPInDatagrams	32-bit counter	total # datagrams delivered
1.3.6.1.2.1.7.2	UDPNoPorts	32-bit counter	# undeliverable datagrams (no application at port)
1.3.6.1.2.1.7.3	UDInErrors	32-bit counter	# undeliverable datagrams (all other reasons)
1.3.6.1.2.1.7.4	UDPOutDatagrams	32-bit counter	total # datagrams sent
1.3.6.1.2.1.7.5	udpTable	SEQUENCE	one entry for each port currently in use

- The Simple Network Management Protocol (SNMP) was designed to facilitate the management and monitoring of network devices, such as routers, switches, servers, and more. The decision to use UDP (User Datagram Protocol) instead of TCP (Transmission Control Protocol) as the transport protocol for SNMP was made based on several considerations:
  - Lower Overhead: UDP is a connectionless protocol with less overhead compared to TCP. It doesn't have the inherent reliability mechanisms such as acknowledgments, retransmissions, and flow control, which makes it lightweight. SNMP, being a protocol that primarily deals with small, non-critical messages (like status updates, queries, and notifications), can benefit from the reduced complexity and overhead offered by UDP.
  - Real-Time Monitoring: SNMP often deals with real-time monitoring and data collection tasks. UDP's lack of handshaking and acknowledgment mechanisms means it operates faster and suits scenarios where immediate data retrieval is crucial. While UDP doesn't guarantee delivery, for SNMP, the periodic nature of data collection often mitigates the need for guaranteed delivery of every message.
  - Stateless Communication: UDP is stateless, meaning each datagram is treated as an independent entity. This aligns well with SNMP's model, where each request-response cycle is independent. With SNMP, a manager sends a request (Get, Set, etc.), and the agent responds with the requested information. There is no need for maintaining a connection state between manager and agent, making UDP's stateless nature fitting for this purpose.
  - Efficiency in Certain Network Conditions: In environments where occasional packet loss or less critical information transfer is acceptable, UDP can be more efficient. For example, in networks with low latency and low error rates, UDP can perform well without the overhead of TCP's reliability mechanisms.
- However, it's essential to acknowledge that while UDP offers advantages in terms of simplicity and speed, it lacks the reliability features provided by TCP. SNMP versions, such as SNMPv3, have added security features like authentication and encryption, compensating for UDP's lack of inherent security.