Università di Ferrara

Architettura di Reti

Chapter 5: Network Layer – Control Plane

Carlo Giannelli
carlo.giannelli@unife.it
http://www.unife.it/scienze/informatica/insegnamenti/architettura-reti/
http://docente.unife.it/carlo.giannelli

Computer Networking: A Top Down Approach
Jim Kurose, Keith Ross
Pearson, April 2016

Computer Networks
Andrew Tanenbaum
Prentice Hall

Slides adapted from "Computer Networking: A Top Down Approach", 7th Edition, Global Edition, Jim Kurose, Keith Ross, Pearson, April 2016

Chapter 5: network layer control plane

Chapter goals: understand principles behind network control plane…

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

…and their instantiation, implementation in the Internet:

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

Chapter 5: outline

Network-layer functions

Recall: two network-layer functions:

- *forwarding:* move packets from router's input to appropriate router output

data plane

- *routing:* determine route taken by packets from source to destination

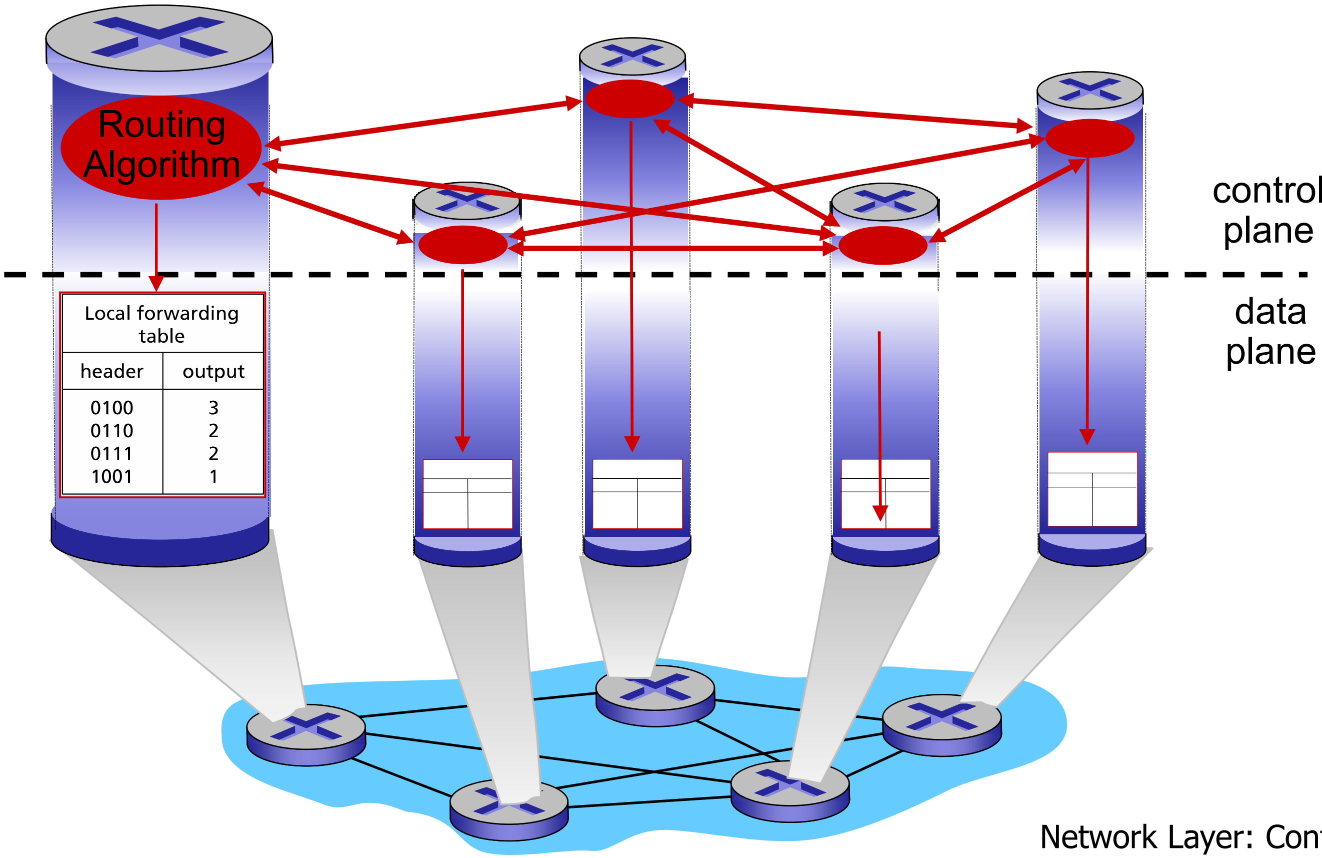
control plane

Two approaches to structuring network control plane:

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

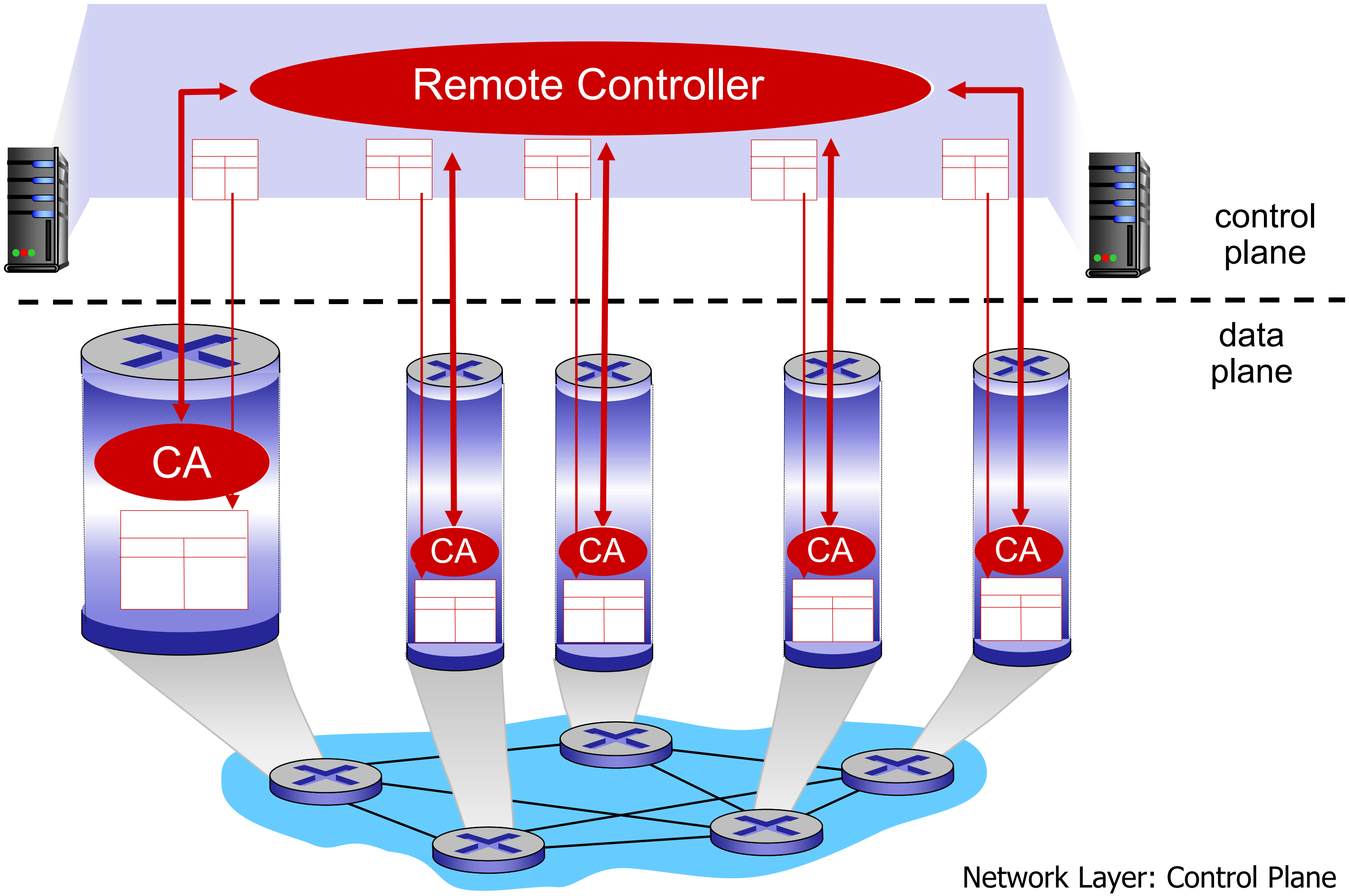
Per-router control plane

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



Logically centralized control plane

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



Chapter 5: outline

5.1 introduction

5.2 routing protocols

- link state
- distance vector

5.3 intra-AS routing in the Internet: OSPF

5.4 routing among the ISPs: BGP

5.5 The SDN control plane

5.6 ICMP: The Internet Control Message Protocol

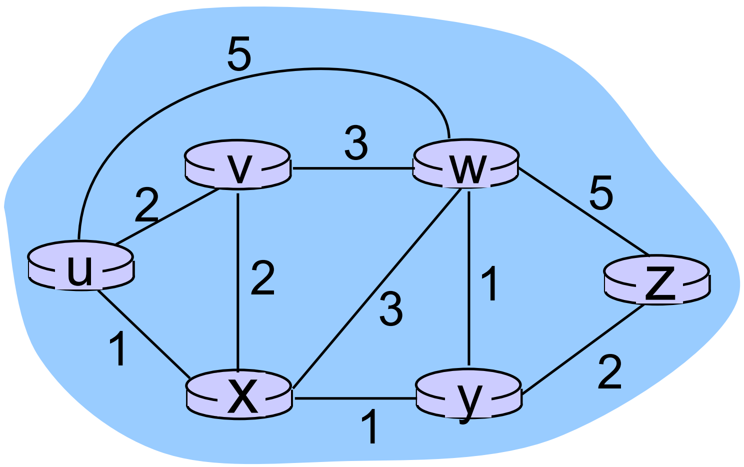
5.7 Network management and SNMP

Routing protocols

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

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

Graph abstraction of the network



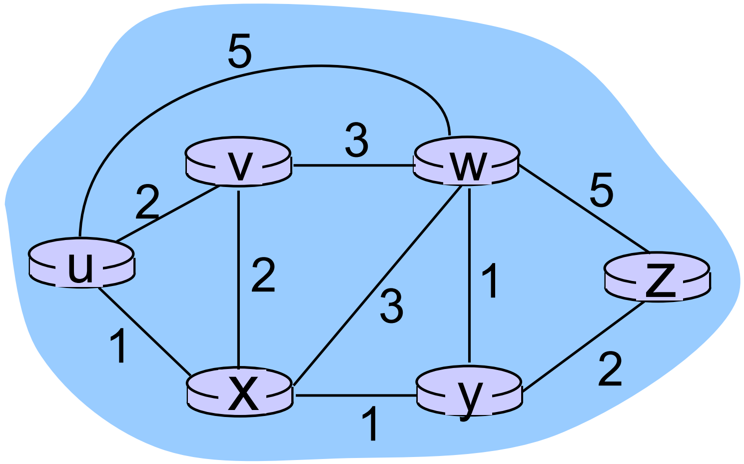
Graph: G = (N,E)

N = set of routers = { u, v, w, x, y, z }

E = set of links ={ (u,v), (u,x), (u,w), (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 $(x_1, x_2, x_3,\ldots, x_p) = c(x_1,x_2) + c(x_2,x_3) + \ldots + c(x_{p-1},x_p)$

Key question: what is the least-cost path between u and z?
Routing algorithm: algorithm that finds that least cost path

Routing algorithm classification

Q: global or decentralized information?

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

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

Q: static or dynamic?

static:
- routes change slowly over time

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

Chapter 5: outline

5.1 introduction

5.2 routing protocols

- link state
- distance vector

5.3 intra-AS routing in the Internet: OSPF

5.4 routing among the ISPs: BGP

5.5 The SDN control plane

5.6 ICMP: The Internet Control Message Protocol

5.7 Network management and SNMP

A link-state routing algorithm

Dijkstra's algorithm

- 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 destinations

Notation:

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

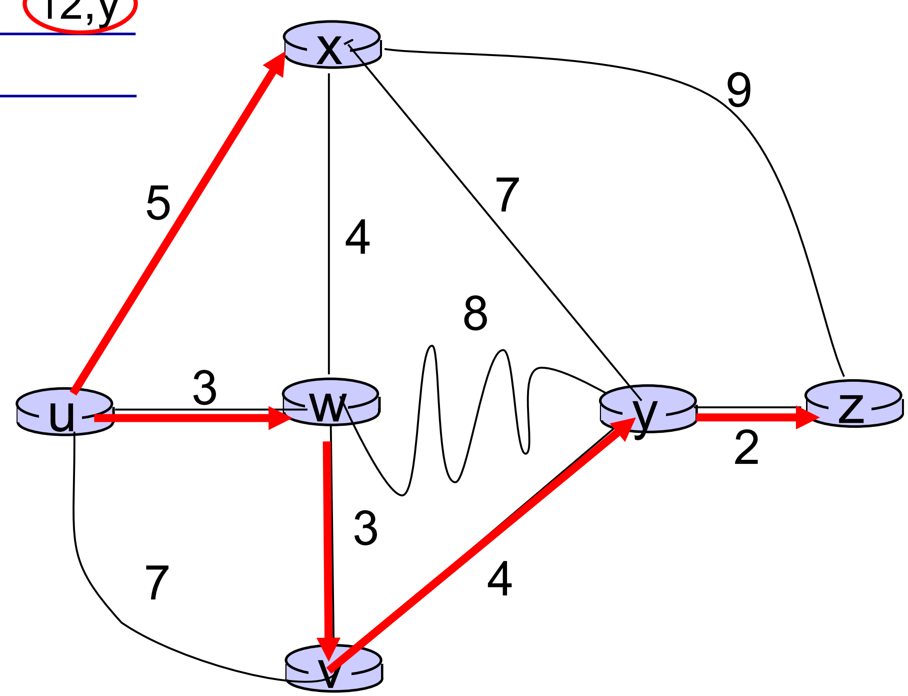
Dijsktra's algorithm

```
Initialization:
  N' = {u}
  for all nodes v
    if v adjacent to u
    then D(v) = c(u,v)
    else D(v) = ∞

 Loop
   find w not in N' such that D(w) is a minimum
   add w to N'
   update D(v) for all v adjacent to w and not in N' :
      D(v) = min( D(v), D(w) + c(w,v) )
   /* new cost to v is either old cost to v or known
    shortest path cost to w plus cost from w to v */
 until all nodes in N'
```

Dijkstra's algorithm: example

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

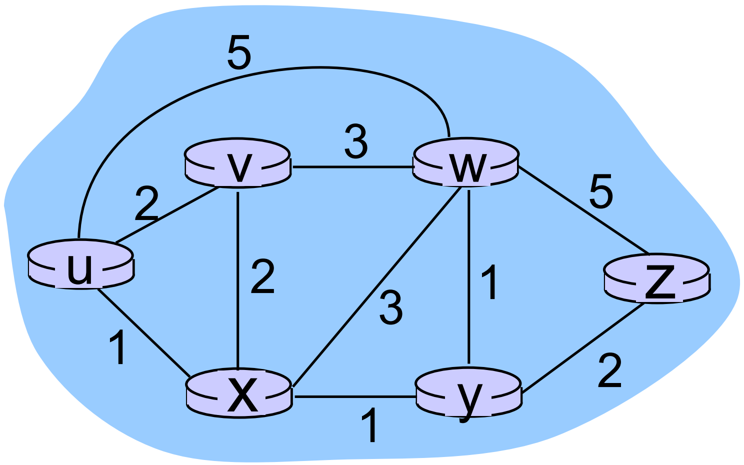


Notes:

- construct shortest path tree by tracing predecessor nodes
- at algorithm termination, for each node is known the least-cost path from the source node

Dijkstra's algorithm: another example

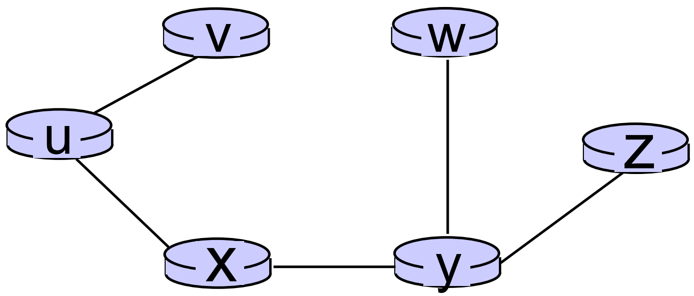
Step	N'	D(v),p(v)	D(w),p(w)	D(x),p(x)	D(y),p(y)	D(z),p(z)
0	u	2,u	5,u	1,u	∞	∞
1	ux	2,u	4,x		2,x	∞
2	uxy	2,u	3,y			4,y
3	uxyv		3,y			4,y
4	uxyvw					4,y
5	uxyvwz					



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

Dijkstra's algorithm: example (2)

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)

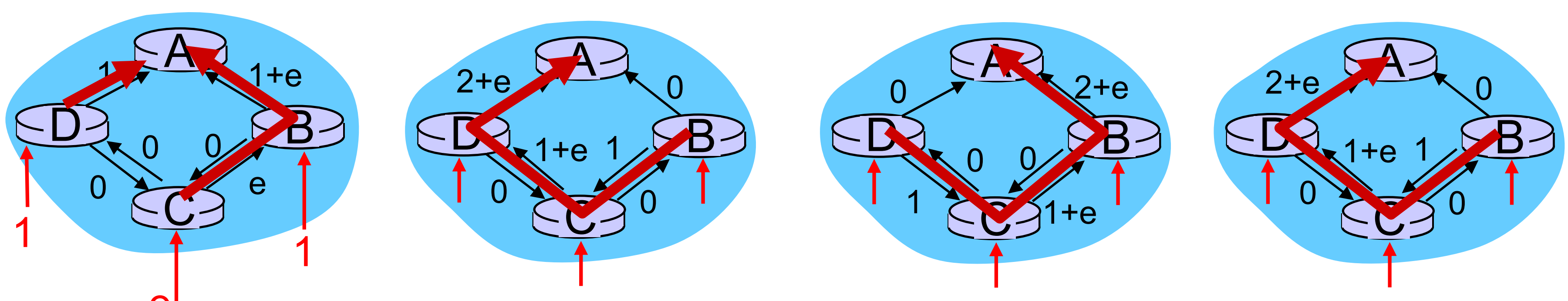
Dijkstra's algorithm, discussion

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(nlogn)

Oscillations possible:

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



initially

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

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

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

Chapter 5: outline

5.1 introduction

5.2 routing protocols

- link state
- distance vector

5.3 intra-AS routing in the Internet: OSPF

5.4 routing among the ISPs: BGP

5.5 The SDN control plane

5.6 ICMP: The Internet Control Message Protocol

5.7 Network management and SNMP

Distance vector algorithm

Bellman-Ford equation (dynamic programming)

let

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

then

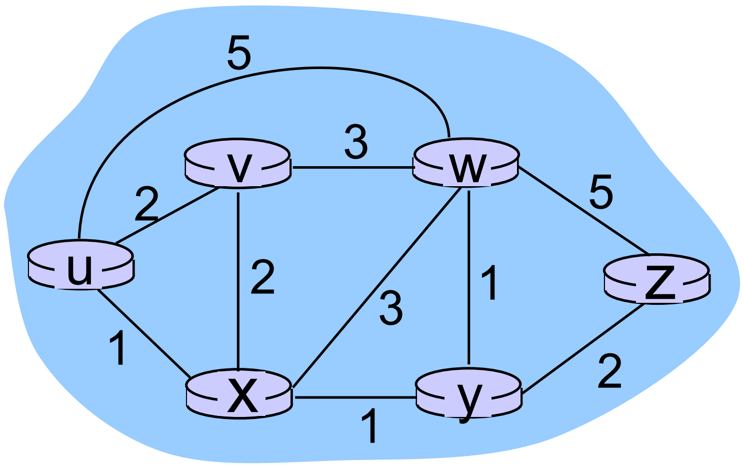
$$d_x(y) = min_v\{c(x,v) + d_v(y)\}$$

cost from neighbor v to destination y

cost to neighbor v

min taken over all neighbors v of x

Bellman-Ford example



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

B-F equation says:

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

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

Distance vector algorithm

Key idea:

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

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

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

Distance vector algorithm

Iterative, Asynchronous: each local iteration caused by:

- local link cost change
- DV update message from neighbor

Distributed:

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

Each node:

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

↓

recompute estimates

↓

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

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

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

node x table

cost to

from	x	y	z
x	0	2	7
y	∞	∞	∞
z	∞	∞	∞

cost to

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

node y table

cost to

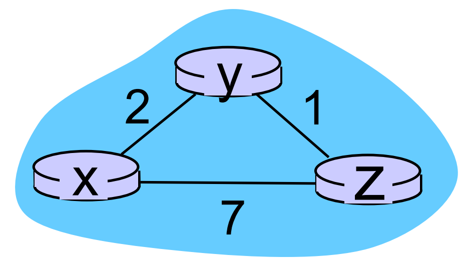
from	x	y	z
x	∞	∞	∞
y	2	0	1
z	∞	∞	∞

node z table

cost to

from	x	y	z
x	∞	∞	∞
y	∞	∞	∞
z	7	1	0

time



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

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

node x table

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

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

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

node y table

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

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

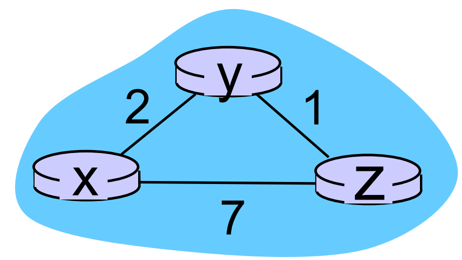
from \ cost to	x	y	z
x	0	2	3
y	2	0	1
z	3	1	0

node z table

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

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

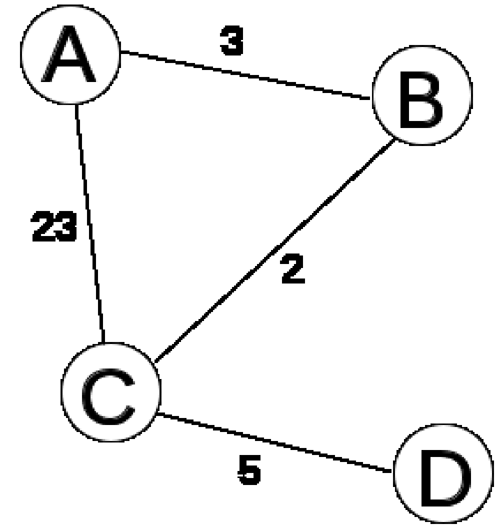
from \ cost to	x	y	z
x	0	2	3
y	2	0	1
z	3	1	0



time

Example from https://en.wikipedia.org/wiki/Distance-vector_routing_protocol

- green: shortest path (so far)
- yellow: a new shortest path
- grey: nodes not neighbors of the current node
- red: invalid entries since refer to distances from a node to itself, or via itself



T=1 Information about neighbors only

from A	via A	via B	via C	via D
to A				
to B		3		
to C			23	
to D				

from B	via A	via B	via C	via D
to A	3			
to B				
to C			2	
to D				

from C	via A	via B	via C	via D
to A	23			
to B		2		
to C				
to D				5

from D	via A	via B	via C	via D
to A				
to B				
to C			5	
to D				

T=2 1st update after DV dissemination among neighbors

from A	via A	via B	via C	via D
to A				
to B		3	25	
to C		5	23	
to D			28	

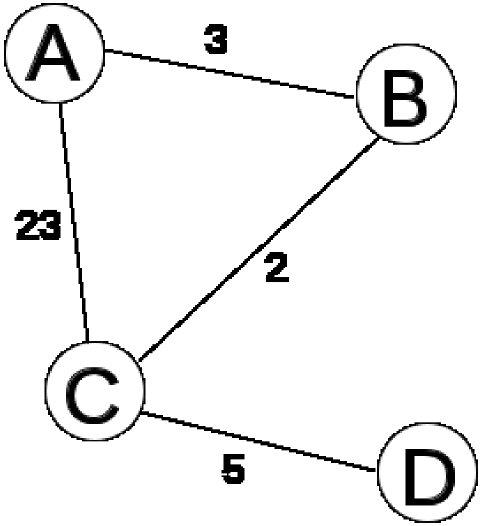
from B	via A	via B	via C	via D
to A	3		25	
to B				
to C	26		2	
to D			7	

from C	via A	via B	via C	via D
to A	23	5		
to B	26	2		
to C				
to D				5

from D	via A	via B	via C	via D
to A			28	
to B			7	
to C			5	
to D				

Example from https://en.wikipedia.org/wiki/Distance-vector_routing_protocol

- green: shortest path (so far)
- yellow: a new shortest path
- grey: nodes not neighbors of the current node
- red: invalid entries since refer to distances from a node to itself, or via itself



T=3 2nd update after DV dissemination among neighbors

from A	via A	via B	via C	via D
to A				
to B		3	25	
to C		5	23	
to D		10	28	

from B	via A	via B	via C	via D
to A	3		7	
to B				
to C	8		2	
to D	31		7	

from C	via A	via B	via C	via D
to A	23	5		33
to B	26	2		12
to C				
to D	51	9		5

from D	via A	via B	via C	via D
to A			10	
to B			7	
to C			5	
to D				

T=4 3rd update, but without shortest path modifications, thus the algorithm ends

from A	via A	via B	via C	via D
to A				
to B		3	25	
to C		5	23	
to D		10	28	

from B	via A	via B	via C	via D
to A	3		7	
to B				
to C	8		2	
to D	13		7	

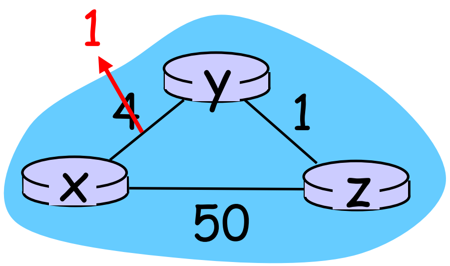
from C	via A	via B	via C	via D
to A	23	5		15
to B	26	2		12
to C				
to D	33	9		5

from D	via A	via B	via C	via D
to A			10	
to B			7	
to C			5	
to D				

Distance vector: link cost changes (1)

Link cost changes:

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



"good news travels fast"

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

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

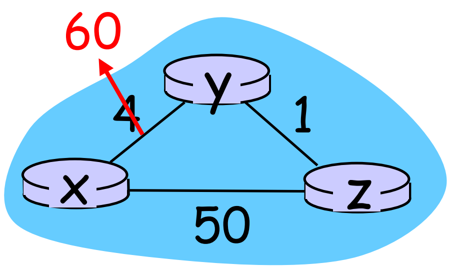
t_2 : *y* receives *z*'s update, updates its distance table. *y*'s least costs do *not* change, so *y* does *not* send a message to *z*.

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

Distance vector: link cost changes (2)

Link cost changes:

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

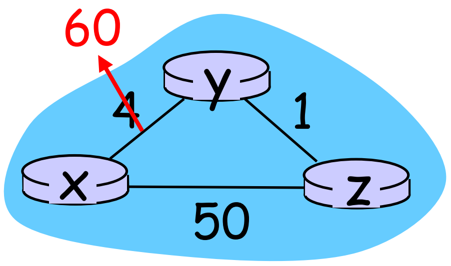


- 44 iterations before algorithm stabilizes
- y knows: from y to x the cost is 60, from z to x the cost is 5
- y (erroneously) computes: new shortest path to x is via z with cost 5+1=6 (y does not know this path is towards itself)
- y advertises to z: I have a shortest path towards x with cost 6
- z advertises shortest path towards x with cost 7 (via y)
- y advertises shortest path towards x with cost 8 (via z)
- …

Distance vector: link cost changes (3)

Link cost changes:

- node detects local link cost change
- *bad news travels slow* – "count to infinity" problem!
- 44 iterations before algorithm stabilizes



Poisoned reverse:

- If z routes through y to get to x:
  - at first, z tells y its (z's) distance to x is infinite (so y won't route to x via z)
  - then, y identifies the shortest path towards x via itself with cost 60
  - finally, y identifies the shortest path towards x via z with cost 51
- will this completely solve count to infinity problem?
  - no in case of loops involving three or more nodes (rather than simply two neighbor nodes)

Comparison of LS and DV algorithms

Message complexity

- ***LS:*** with 'N' nodes, 'E' links, O(NE) msgs sent
- ***DV:*** exchange between neighbors only

Speed of convergence

- ***LS:*** $O(N^2)$ algorithm requiring O(NE) msgs
  - fast, but may have oscillations with varying costs
- ***DV:*** convergence time varies, can be slow
  - may be routing loops
  - count-to-infinity problem

Robustness: what happens if a router malfunctions?

LS:

- node can advertise incorrect *link* cost
- each node computes only its *own* table
- calculations are separated on each node

DV:

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

Chapter 5: outline

5.1 introduction

5.2 routing protocols
- link state
- distance vector

5.3 intra-AS routing in the Internet: OSPF

5.4 routing among the ISPs: BGP

5.5 The SDN control plane

5.6 ICMP: The Internet Control Message Protocol

5.7 Network management and SNMP

Making routing scalable

Our routing study thus far - idealized

- all routers identical
- network "flat"

… *not* true in practice

Scale: with billions of destinations:

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

Administrative autonomy

- internet = network of networks
- each network admin may want to control routing in its own network

Internet approach to scalable routing

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

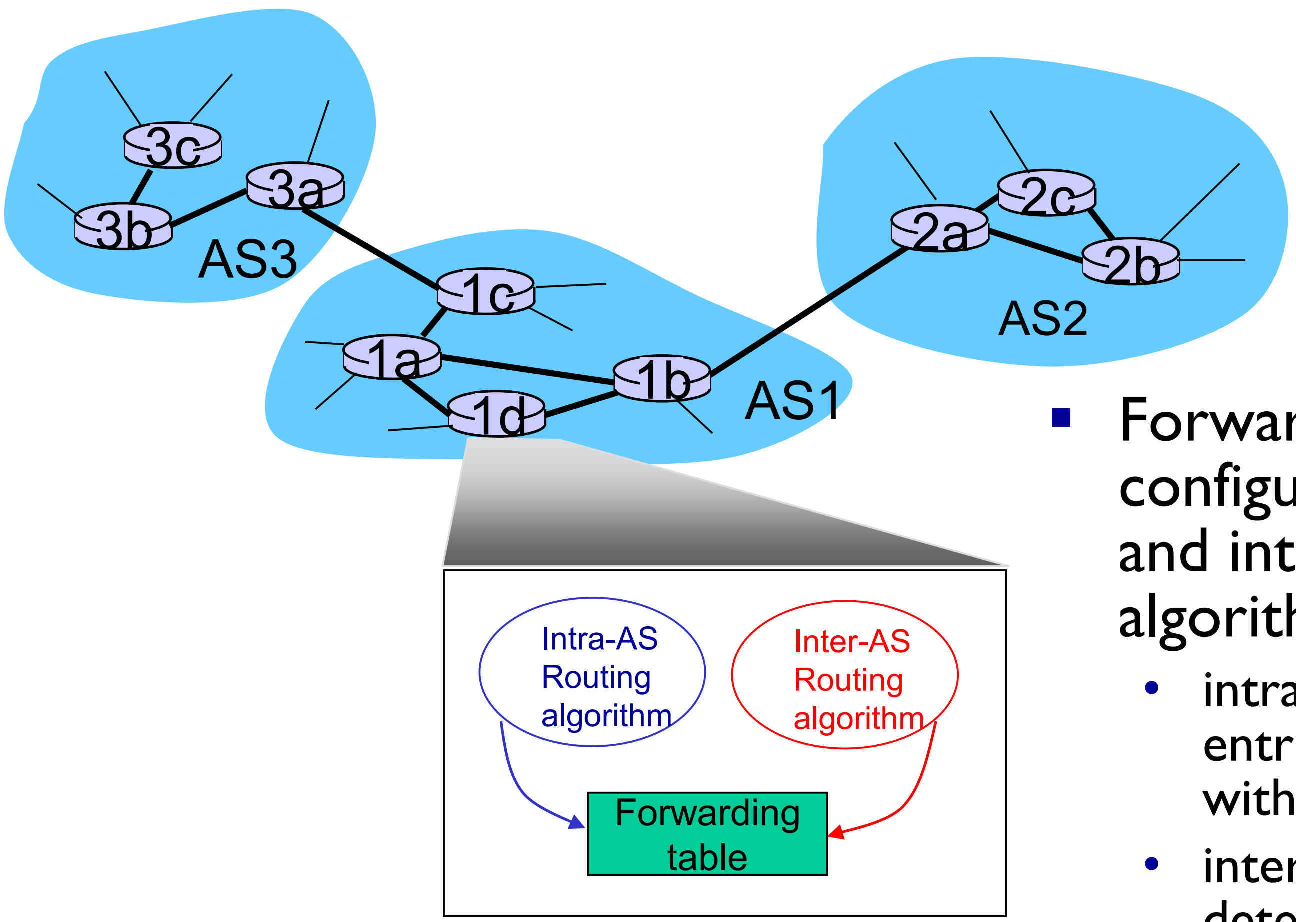
Intra-AS routing

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

Inter-AS routing

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

Interconnected ASes



- Forwarding table configured by both intra- and inter-AS routing algorithm
  - intra-AS routing determine entries for destinations within AS
  - inter-AS & intra-AS determine entries for external destinations

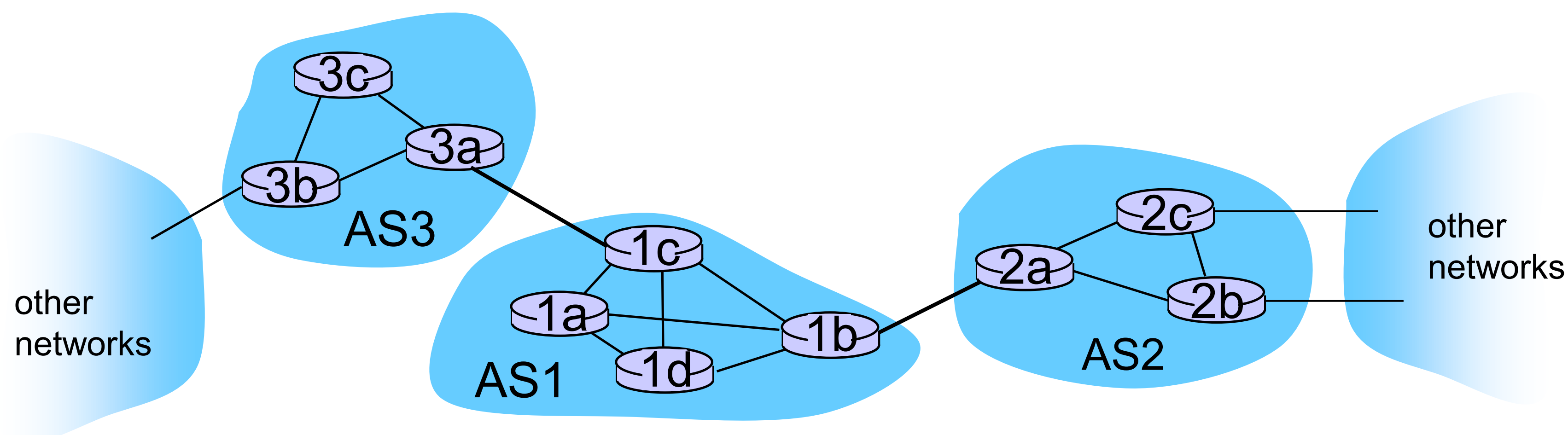
Inter-AS tasks

- Suppose router in AS1 receives datagram destined outside of AS1:
  - router should forward packet to gateway router wither 1c or 1b, but which one?

AS1 must:

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

job of inter-AS routing!



Intra-AS Routing

- Also known as *Interior Gateway Protocols (IGP)*
- Most common intra-AS routing protocols:
  - RIP: Routing Information Protocol
  - OSPF: Open Shortest Path First (Intermediate System to Intermediate System protocol, aka IS-IS, essentially same as OSPF)
  - IGRP: Interior Gateway Routing Protocol (Cisco proprietary for decades, until 2016)

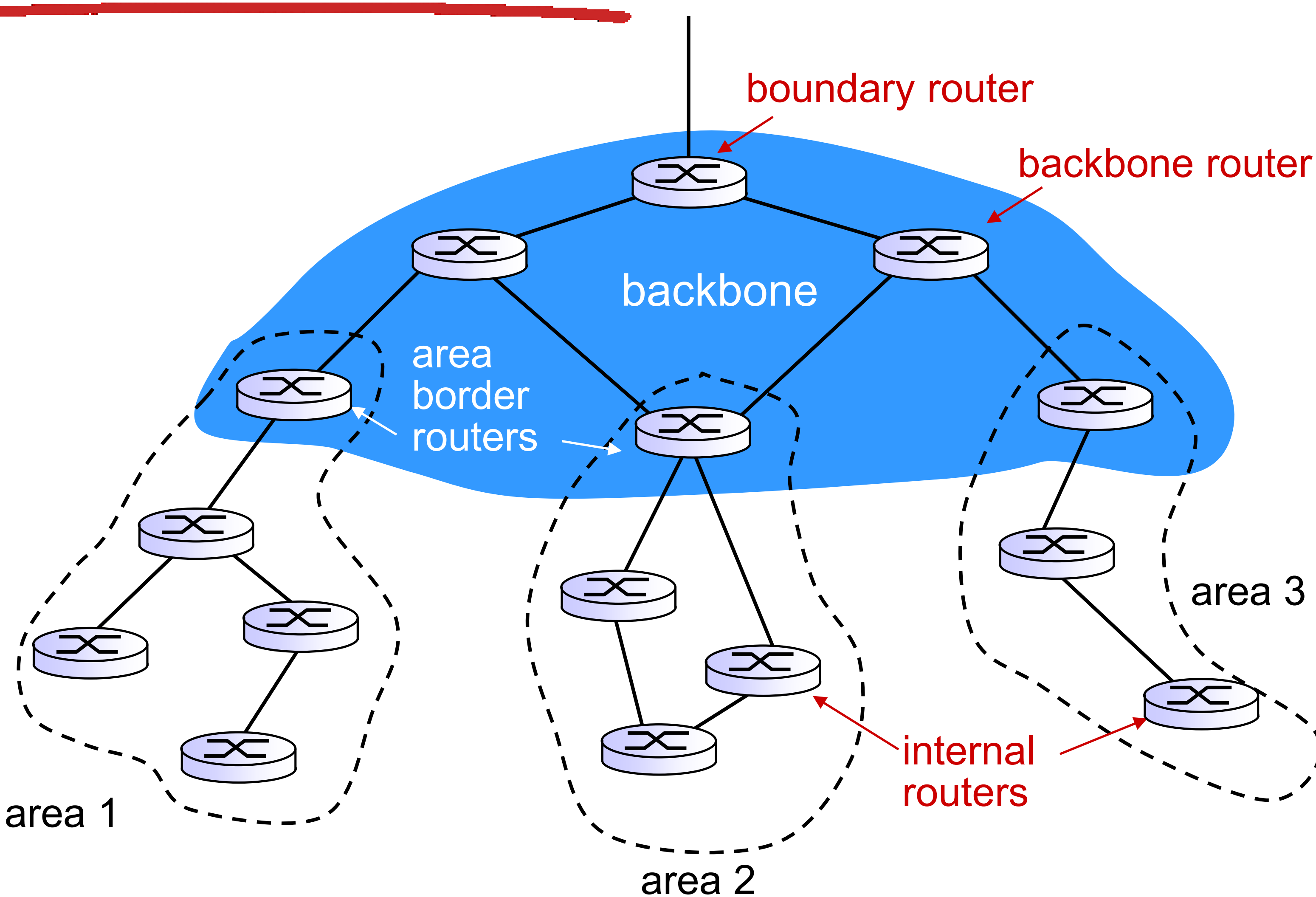
OSPF (Open Shortest Path First)

- "Open": publicly available
- Uses link-state algorithm
  - link state packet dissemination
  - topology map at each node
  - route computation using Dijkstra's algorithm
- Router floods OSPF link-state advertisements to all other routers in *entire* AS
  - carried in OSPF messages directly over IP (rather than TCP or UDP)
  - link state: for each attached link
- *IS-IS routing* protocol: nearly identical to OSPF

OSPF "advanced" features

- *Security:* all OSPF messages authenticated (to prevent malicious intrusion)
- Multiple same-cost paths allowed (only one path in RIP)
- For each link, multiple cost metrics for different Type of Service, ToS (e.g., satellite link cost set low for best effort ToS, high for real-time ToS)
- Integrated unicast and multicast support:
  - multicast OSPF (MOSPF) uses same topology database as OSPF
- Hierarchical OSPF in large domains

Hierarchical OSPF



Hierarchical OSPF

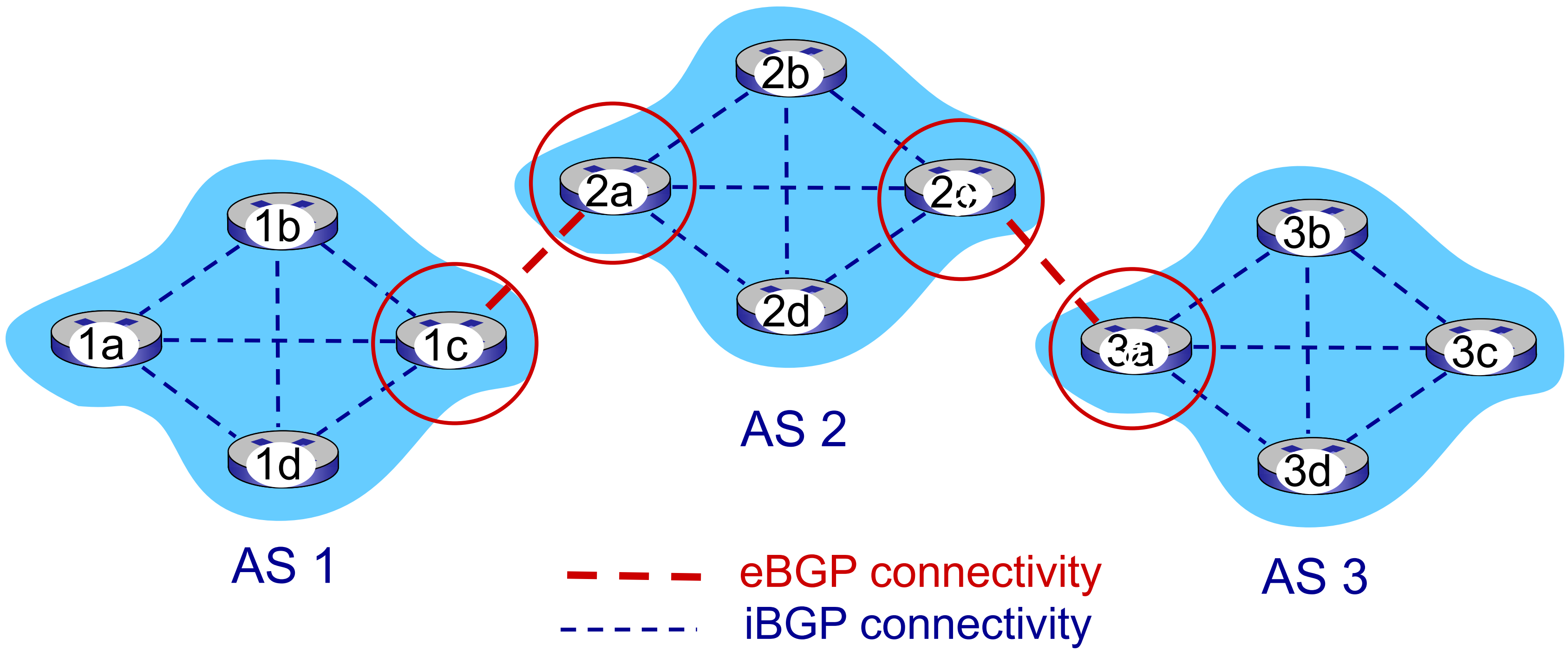
- *Two-level hierarchy:* local area, backbone
  - link-state advertisements only in area
  - each node has detailed area topology; only know direction (shortest path) to nets in other areas
- *Area Border routers:* "summarize" distances to nets in own area, advertise to other Area Border routers
- *Backbone routers:* run OSPF routing limited to backbone
- *Boundary routers:* connect to other ASes

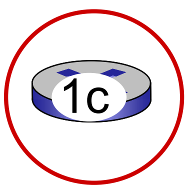
Chapter 5: outline

Internet inter-AS routing: BGP

- BGP (Border Gateway Protocol): *the* de facto inter-domain routing protocol
  - "glue that holds the Internet together"
- 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*
- Allows subnet to advertise its existence to the rest of the Internet: *"I am here"*

eBGP, iBGP connections

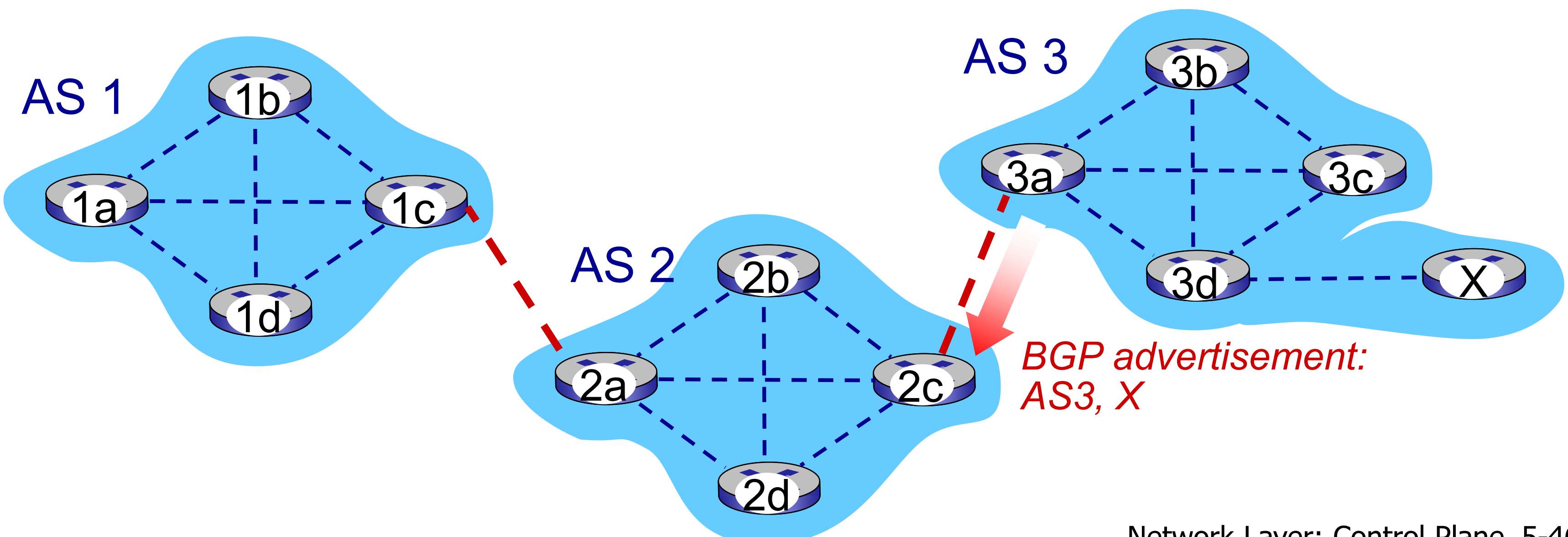




gateway routers run both eBGP and iBGP protocols

BGP basics

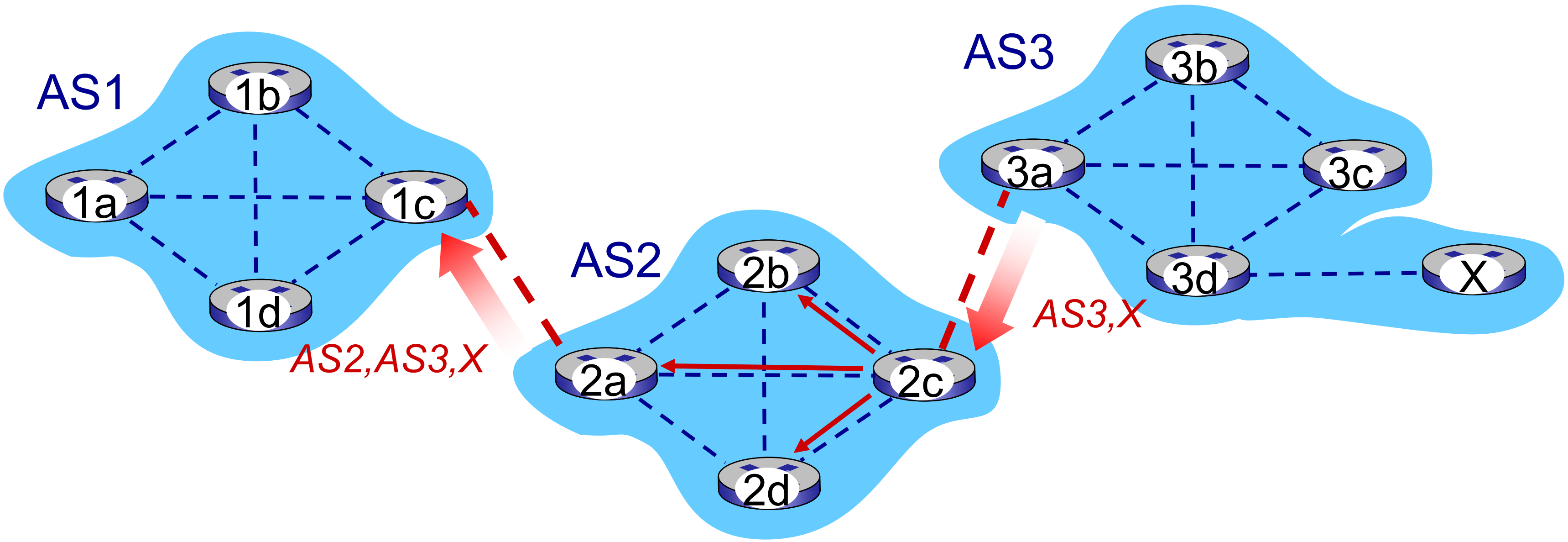
- BGP session: two BGP routers ("peers") exchange BGP messages over semi-permanent TCP connection:
  - advertising *paths* to different destination network prefixes (BGP is a "path vector" protocol)
- When AS3 gateway router 3a advertises path AS3,X to AS2 gateway router 2c:
  - AS3 *promises* to AS2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



Path attributes and BGP routes

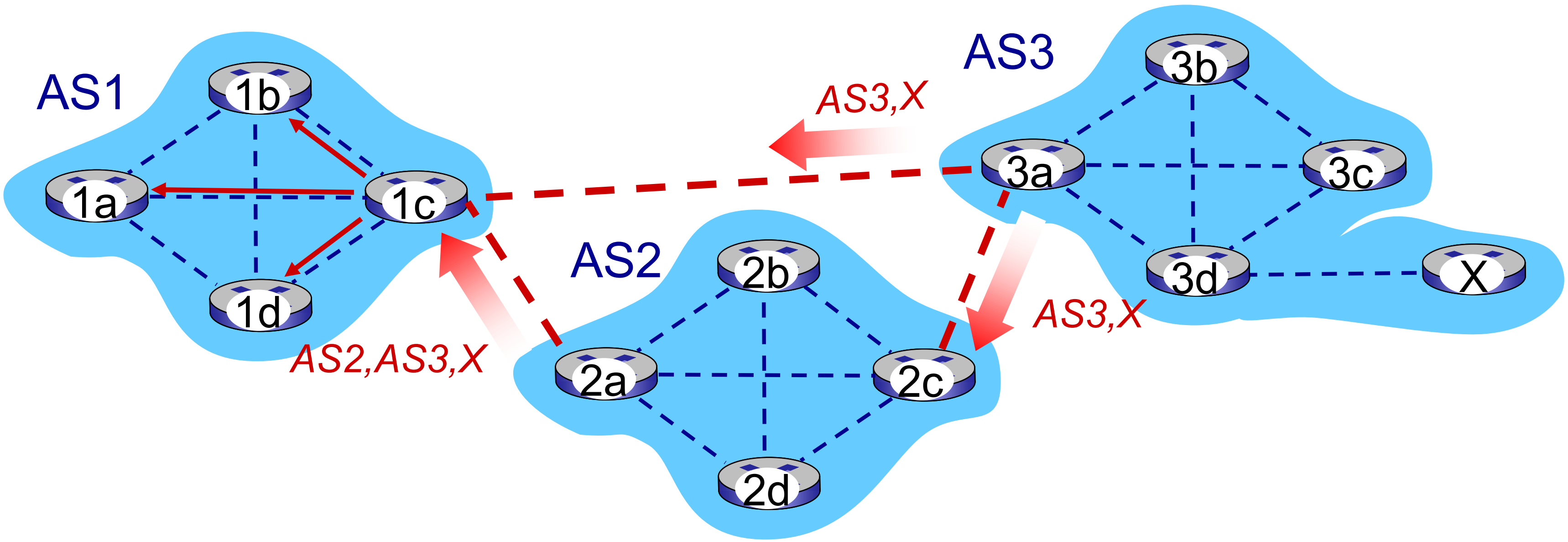
- Advertised prefix includes BGP attributes
  - prefix + attributes = "route"
- Two important attributes:
  - AS-PATH: list of ASes through which prefix advertisement has passed
  - NEXT-HOP: indicates specific internal-AS router to next-hop AS
- *Policy-based routing:*
  - gateway receiving route advertisement uses *import policy* to accept/decline path (e.g., never route through AS Y)
  - AS policy also determines whether to *advertise* path to other neighboring ASes

BGP path advertisement



- AS2 router 2c receives path advertisement AS3,X (**via eBGP**) from AS3 router 3a
- Based on AS2 policy, AS2 router 2c accepts path AS3,X and then propagates (**via iBGP**) the accepted path 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:

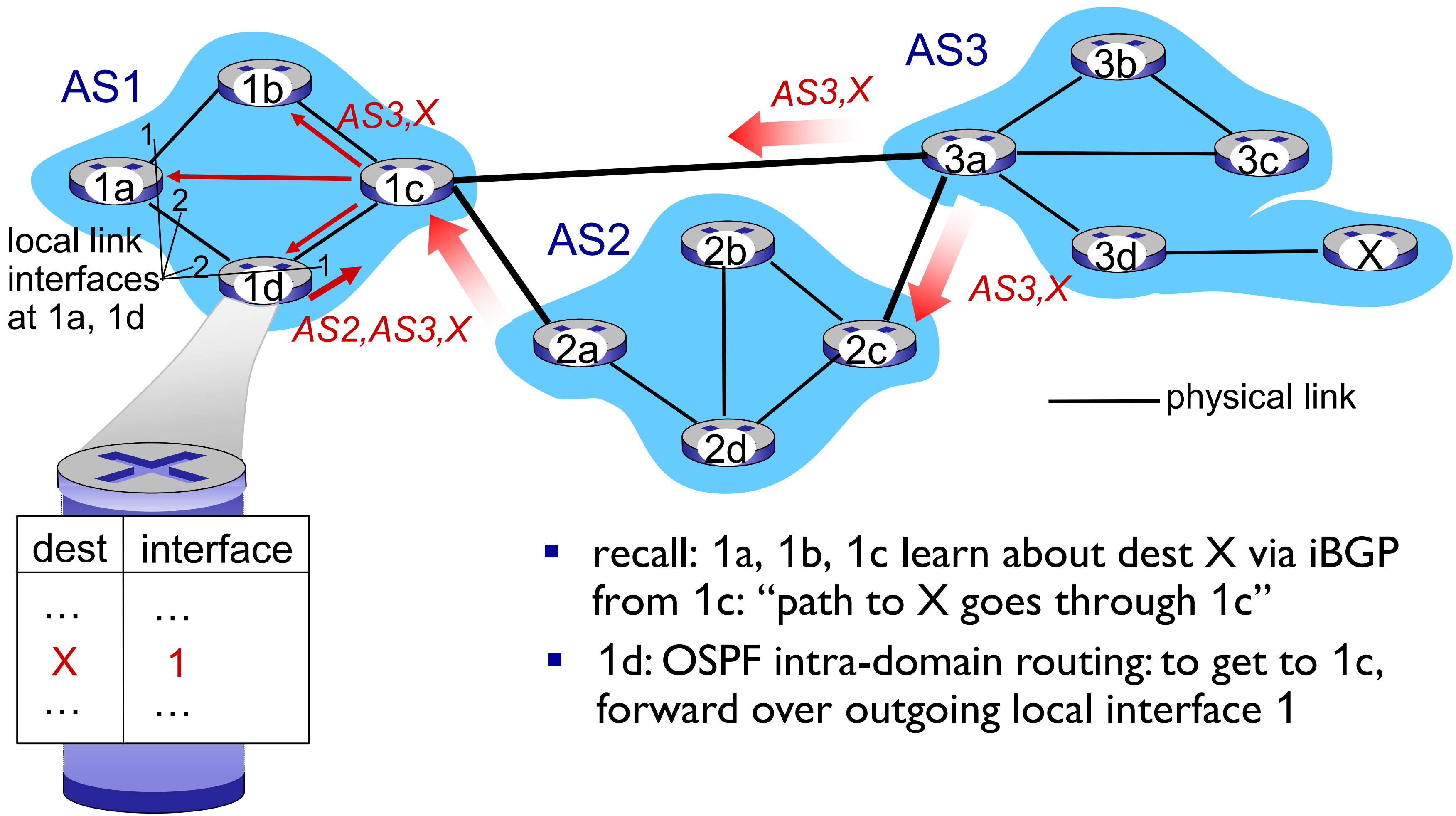
- AS1 gateway router 1c learns path *AS2,AS3,X* from 2a
- AS1 gateway router 1c learns path *AS3,X* from 3a
- Based on policy, (for example) AS1 gateway router 1c chooses path *AS3,X, and advertises path within AS1 via iBGP*

BGP messages

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

BGP, OSPF, forwarding table entries

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

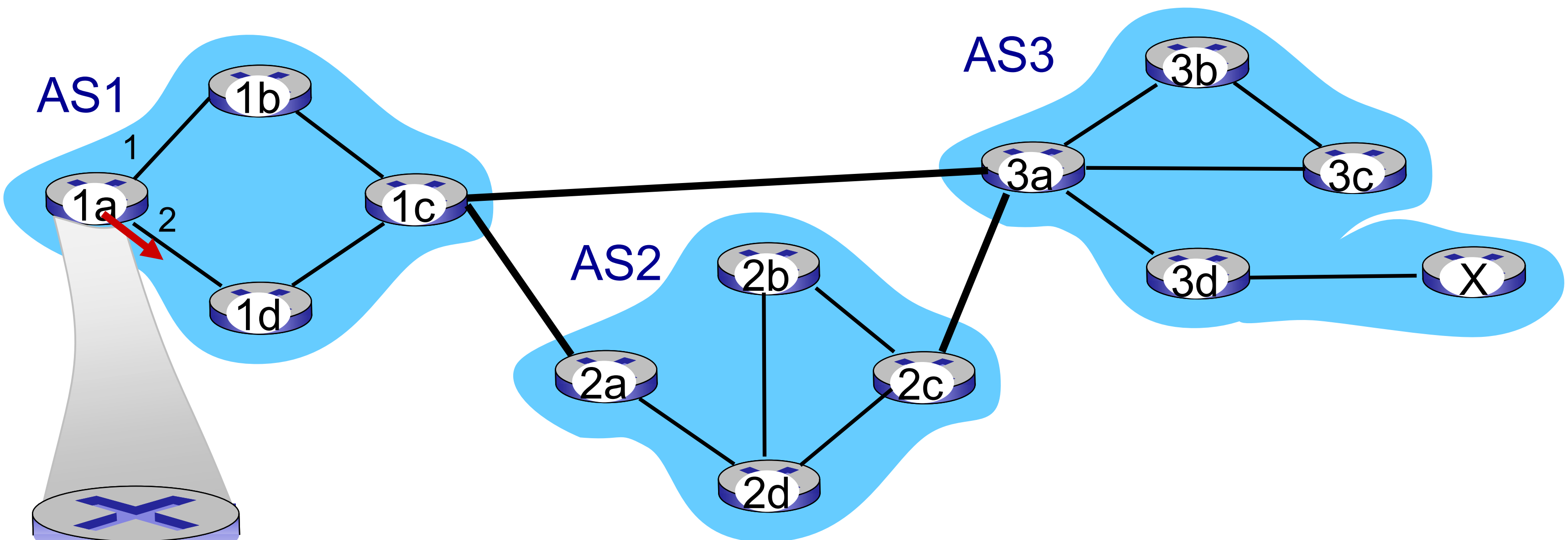


dest	interface
…	…
X	1
…	…

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

BGP, OSPF, forwarding table entries

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



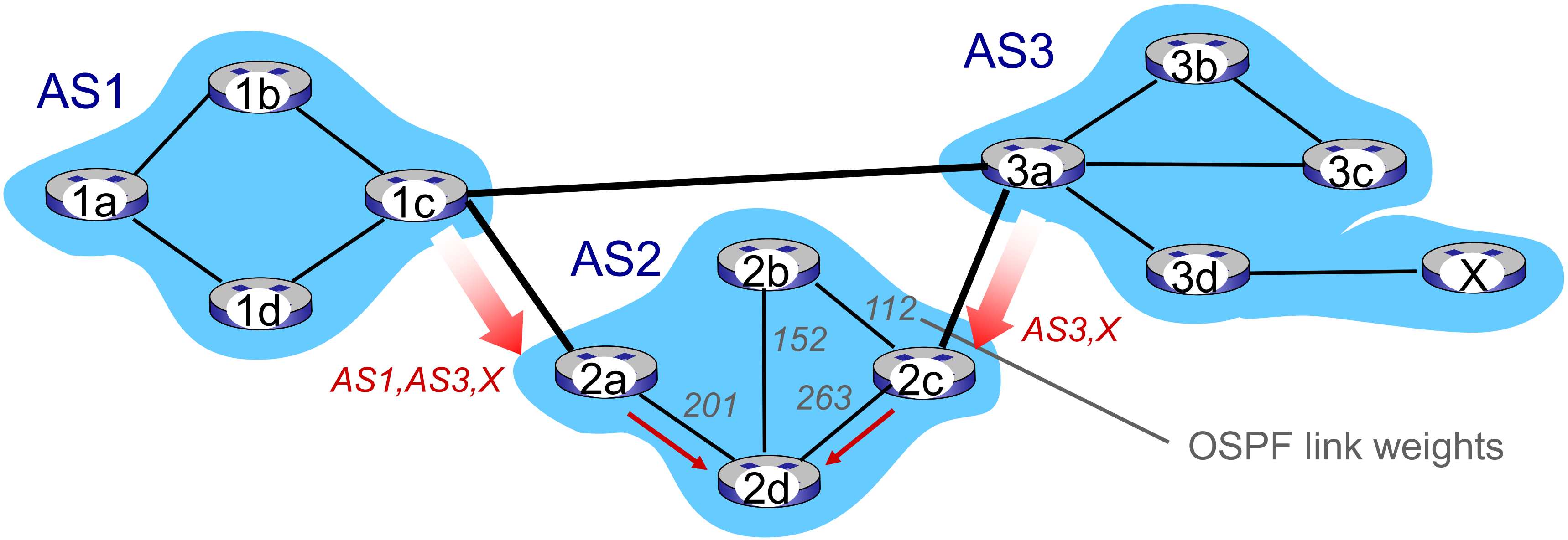
dest	interface
…	…
X	2
…	…

- recall: 1a, 1b, 1c learn about dest X via iBGP from 1c: “path to X goes through 1c”
- 1d: OSPF intra-domain routing: to get to 1c, forward over outgoing local interface 1
- 1a: OSPF intra-domain routing: to get to 1c, forward over outgoing local interface 2

BGP route selection

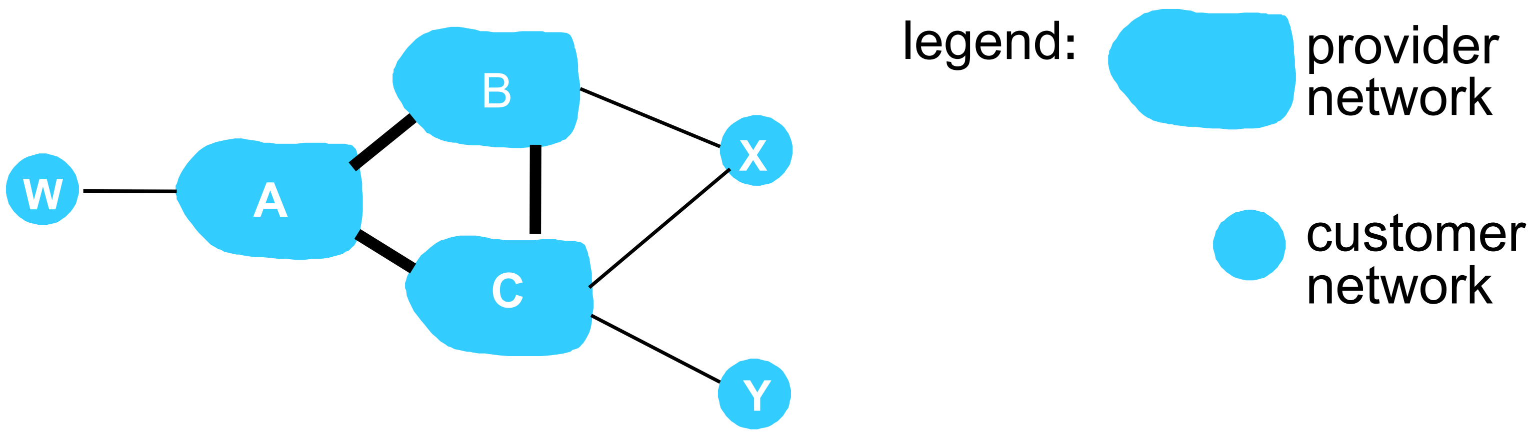
- Router may learn about more than one route to destination AS, selects route based on:
  1. local preference value attribute: policy decision
  2. shortest AS-PATH
  3. closest NEXT-HOP router: hot potato routing
  4. additional criteria

Hot Potato Routing



- 2d learns (via iBGP) it can route to X via 2a or 2c
- *hot potato routing:* choose local gateway that has least intra-domain cost (e.g., 2d chooses 2a, even though more AS hops to *X*): don't worry about inter-domain cost!

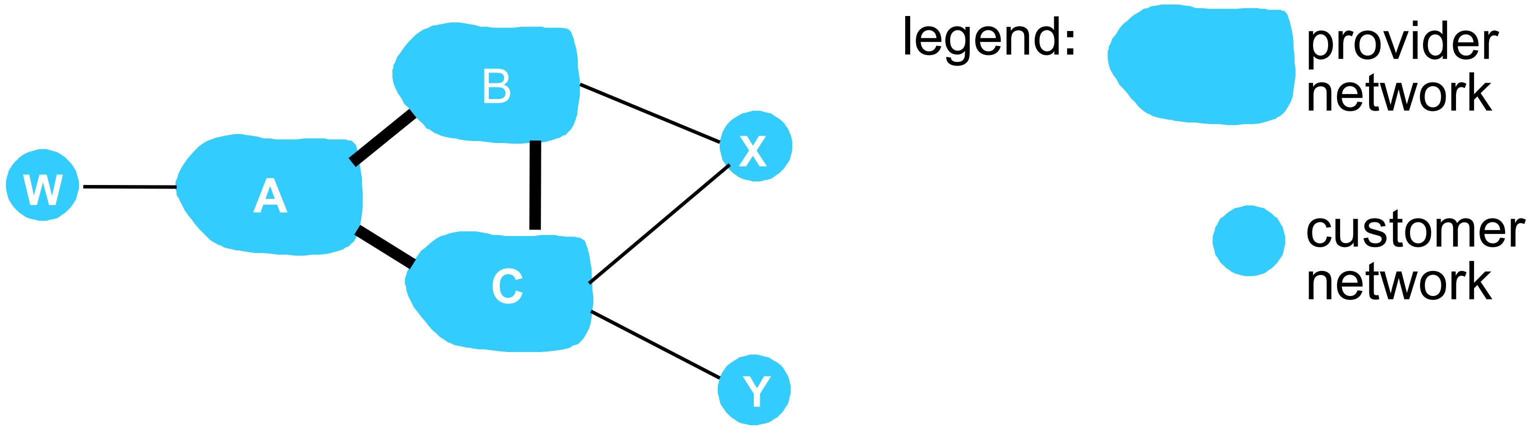
BGP: achieving policy via advertisements



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

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

BGP: achieving policy via advertisements



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

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

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 no policy decisions needed

Scale:

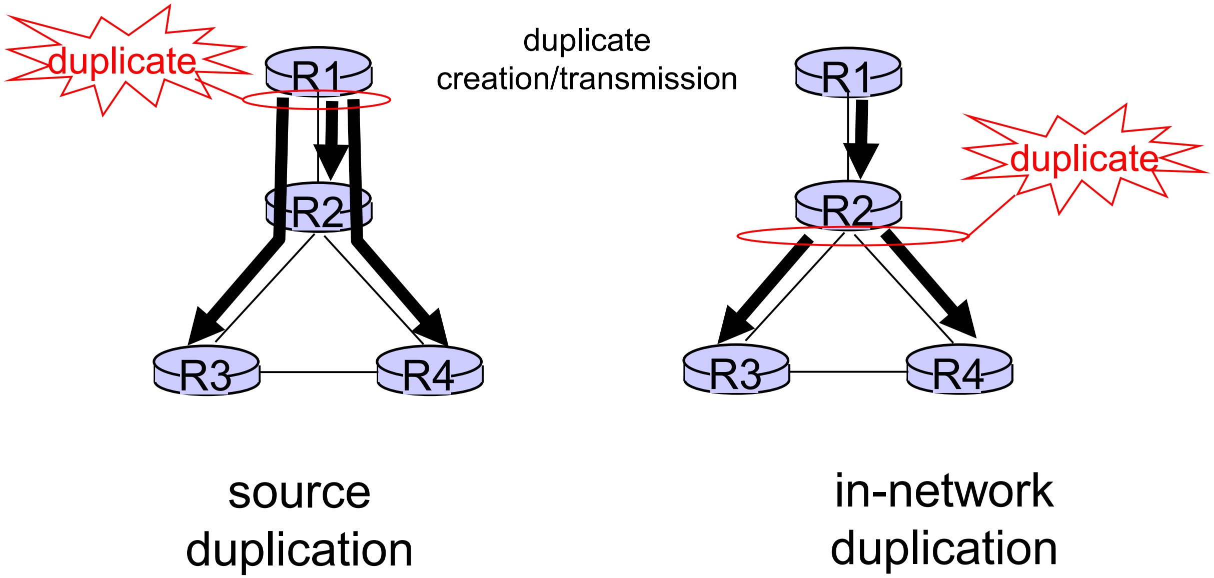
- hierarchical routing saves table size, reduces update traffic

Performance:

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

Broadcast routing

- Deliver packets from source to all other nodes
- Source duplication is inefficient:

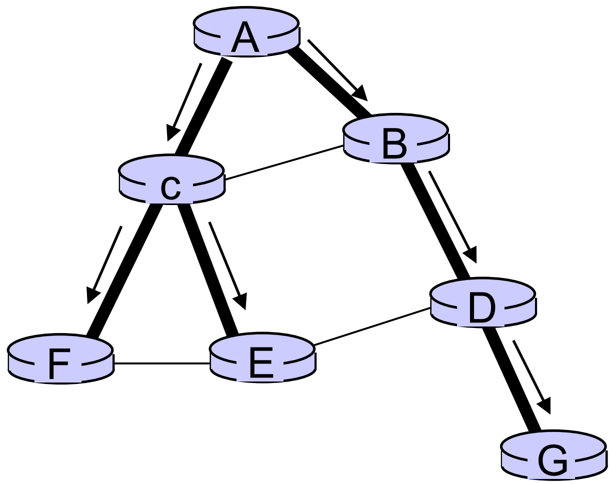


In-network duplication

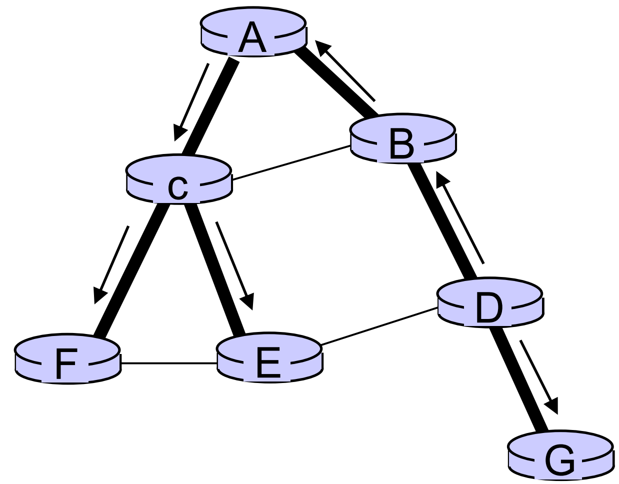
- *Flooding:* when node receives a broadcast packet, sends a copy to all neighbors
  - problems: cycles & broadcast storm
- *Controlled flooding:* node only broadcasts pkt if it hasn't broadcast same packet before
  - node keeps track of packet ids already broadcast
  - or reverse path forwarding (RPF): only forward packet if it arrived on shortest path between node and source
- *Spanning tree:*
  - no redundant packets received by any node

Spanning tree

- First construct a spanning tree
- Nodes then forward/make copies only along the spanning tree



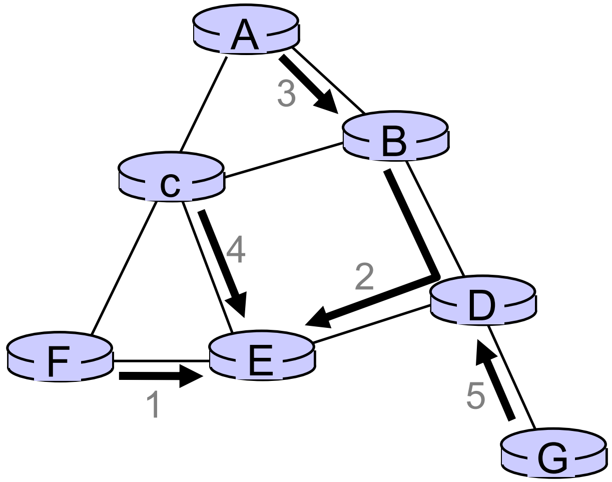
(a) broadcast initiated at A



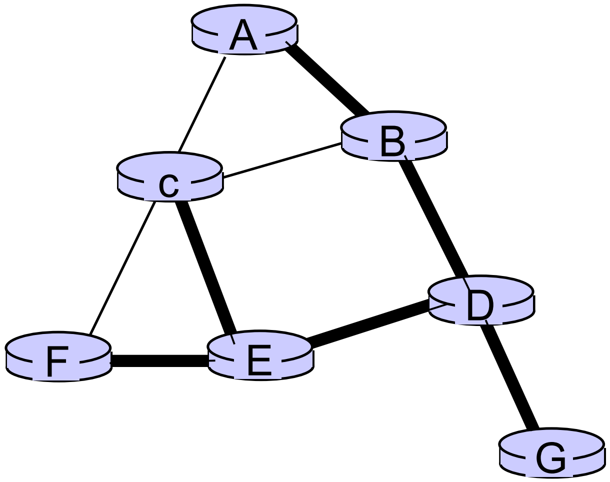
(b) broadcast initiated at D

Spanning tree: creation

- Center node
- Each node sends unicast join message to center node based on topology knowledge
  - message forwarded until it arrives at a node already belonging to the spanning tree



a) stepwise construction of spanning tree (center: E)



(b) constructed spanning tree

Additional Material

Chapter 5: outline

5.1 introduction

5.2 routing protocols

- link state
- distance vector

5.3 intra-AS routing in the Internet: OSPF

5.4 routing among the ISPs: BGP

5.5 The SDN control plane

5.6 ICMP: The Internet Control Message Protocol

5.7 Network management and SNMP

Additional Material

Software defined networking (SDN)

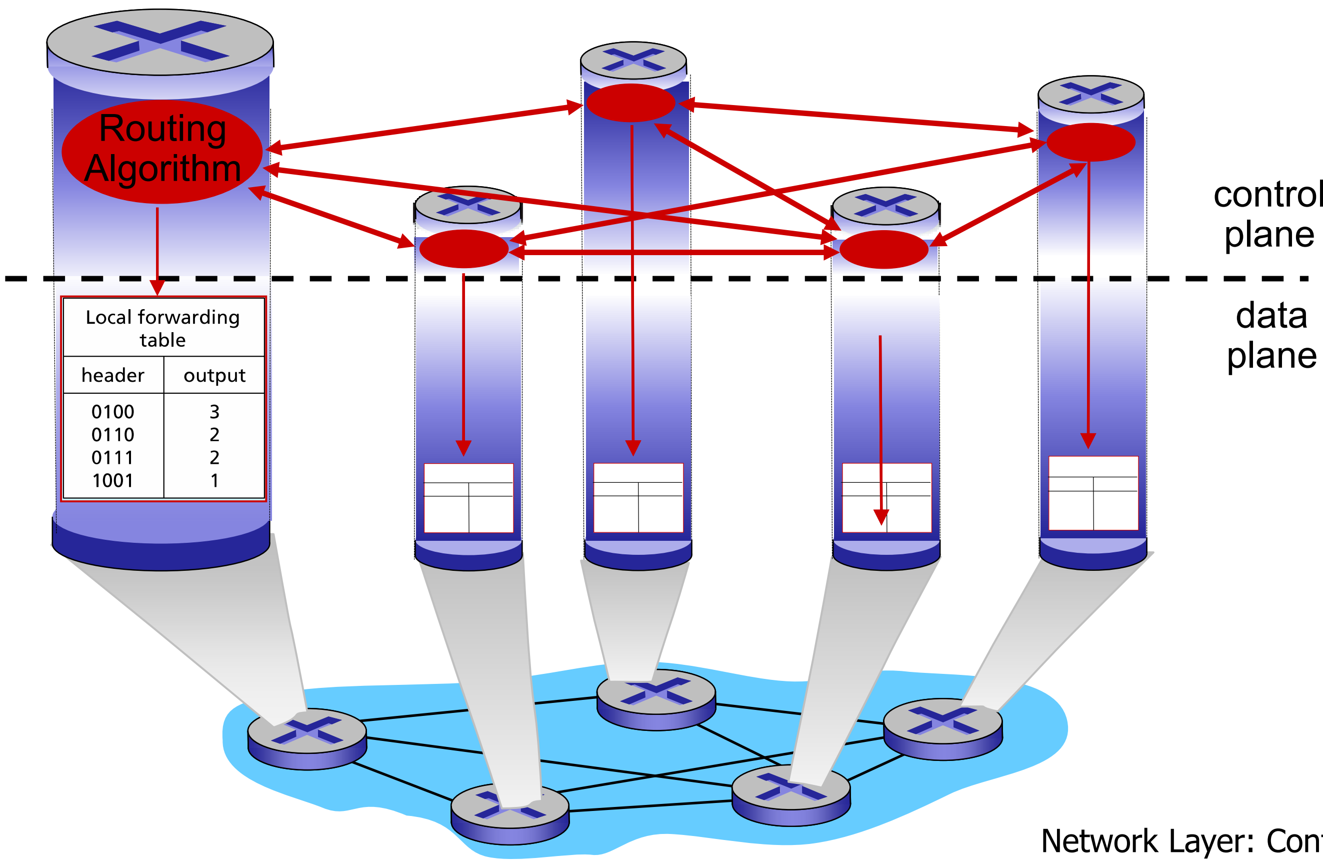
- Internet network layer: historically has been implemented via distributed, per-router approach
  - *monolithic* router contains switching hardware, runs proprietary implementation of Internet standard protocols (IP, RIP, IS-IS, OSPF, BGP) in proprietary router OS (e.g., Cisco IOS)
  - different "middleboxes"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

- ~2005: renewed interest in rethinking network control plane

Additional Material

Recall: per-router control plane

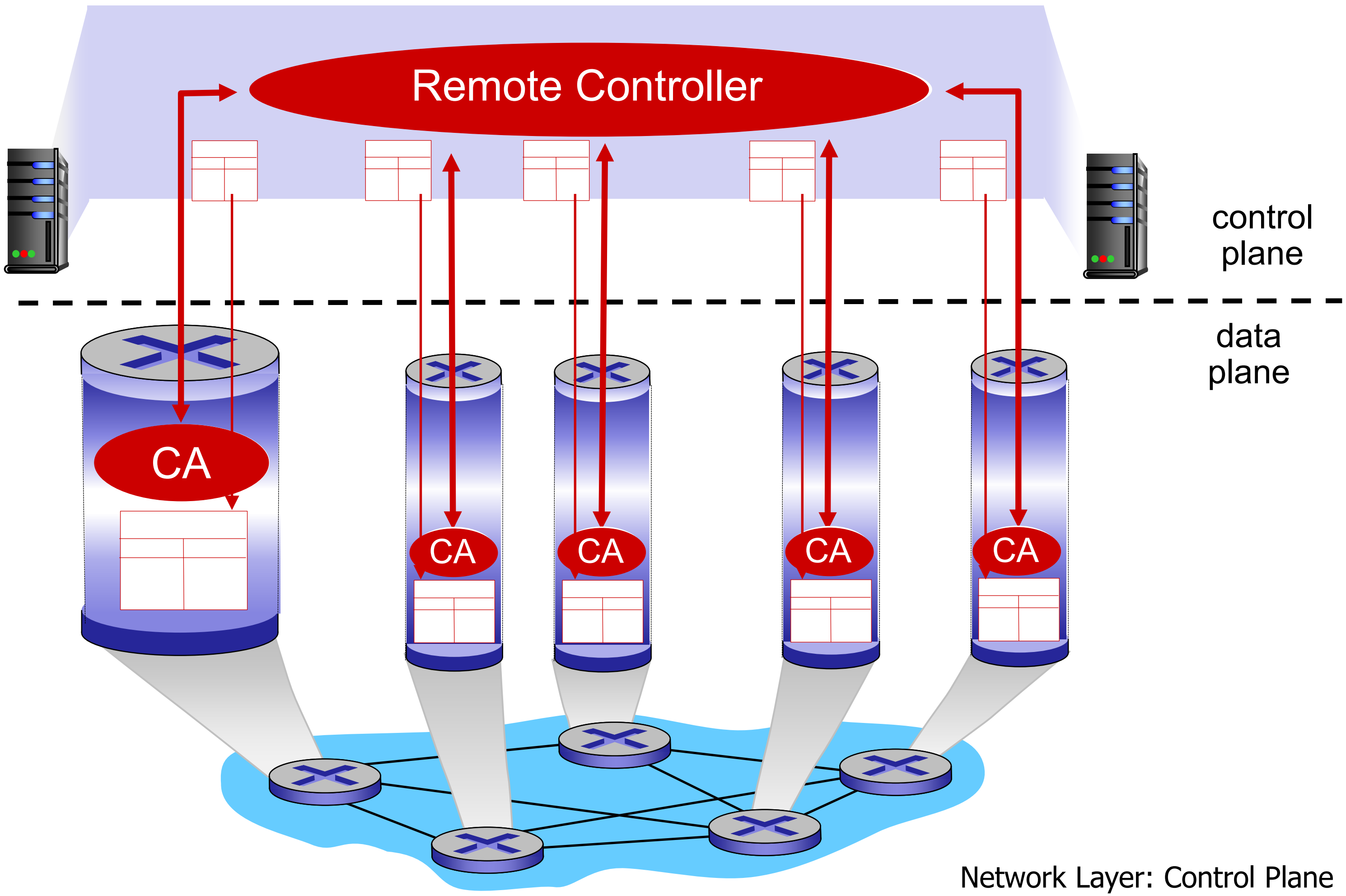
Individual routing algorithm components *in each and every router* interact with each other in control plane to compute forwarding tables



Additional Material

Recall: logically centralized control plane

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



Additional Material

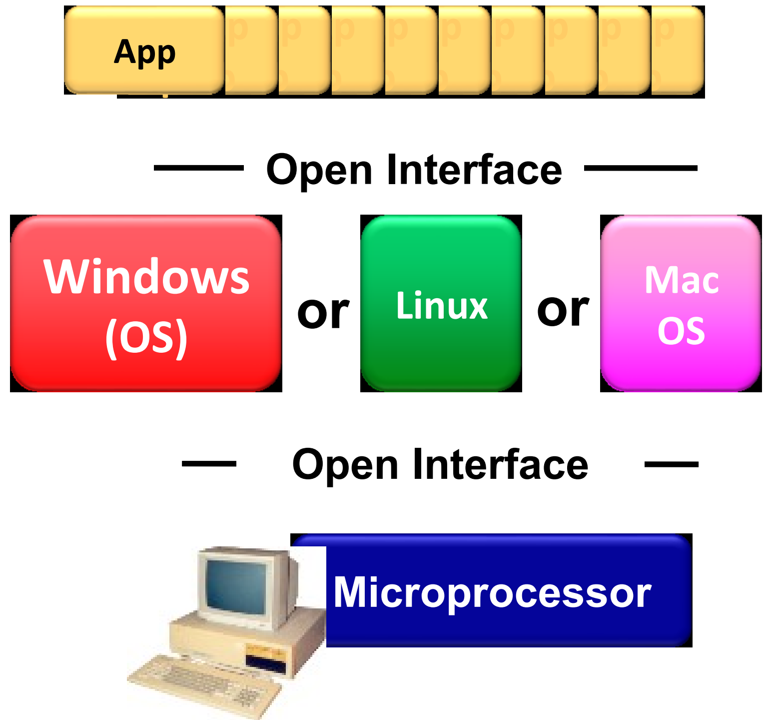
Software defined networking (SDN)

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

Additional Material

Analogy: mainframe to PC evolution*

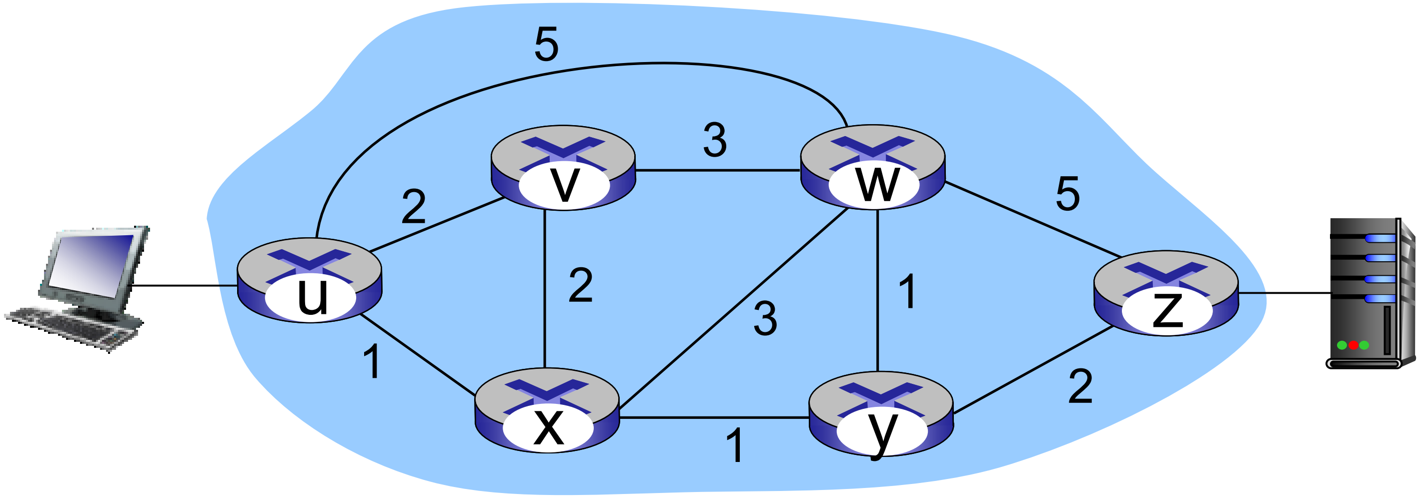


Vertically integrated
Closed, proprietary
Slow innovation
Small industry

Horizontal
Open interfaces
Rapid innovation
Huge industry

* Slide courtesy: N. McKeown

Traffic engineering: difficult traditional routing

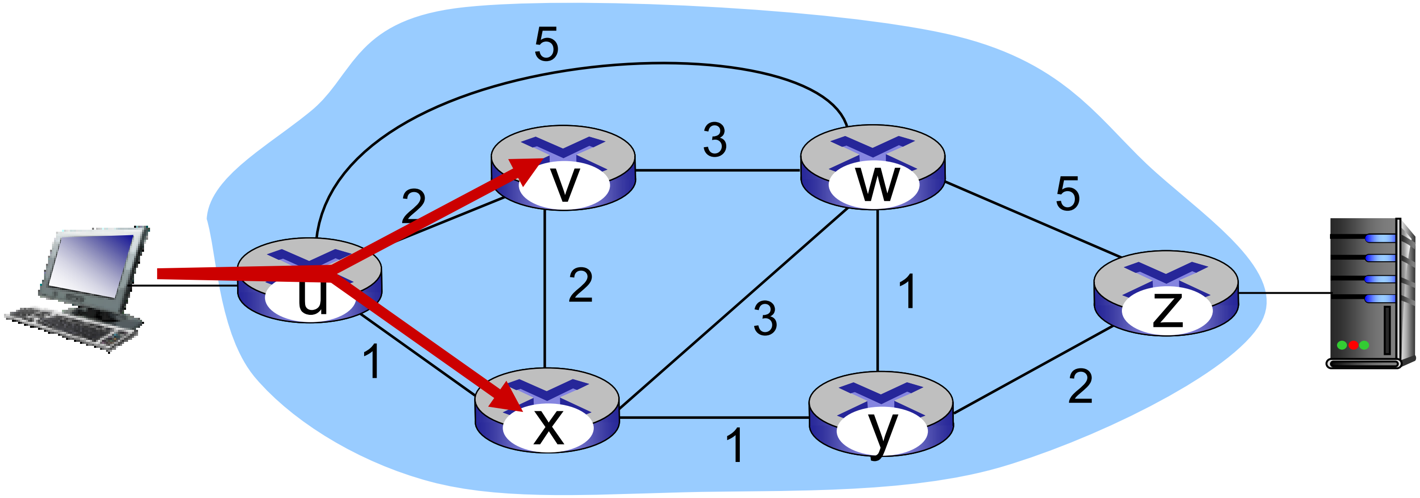


Q: what if network operator wants u-to-z traffic to flow along *uvwz*, x-to-z traffic to flow *xwyz*?

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

Link weights are only control "knobs": wrong!

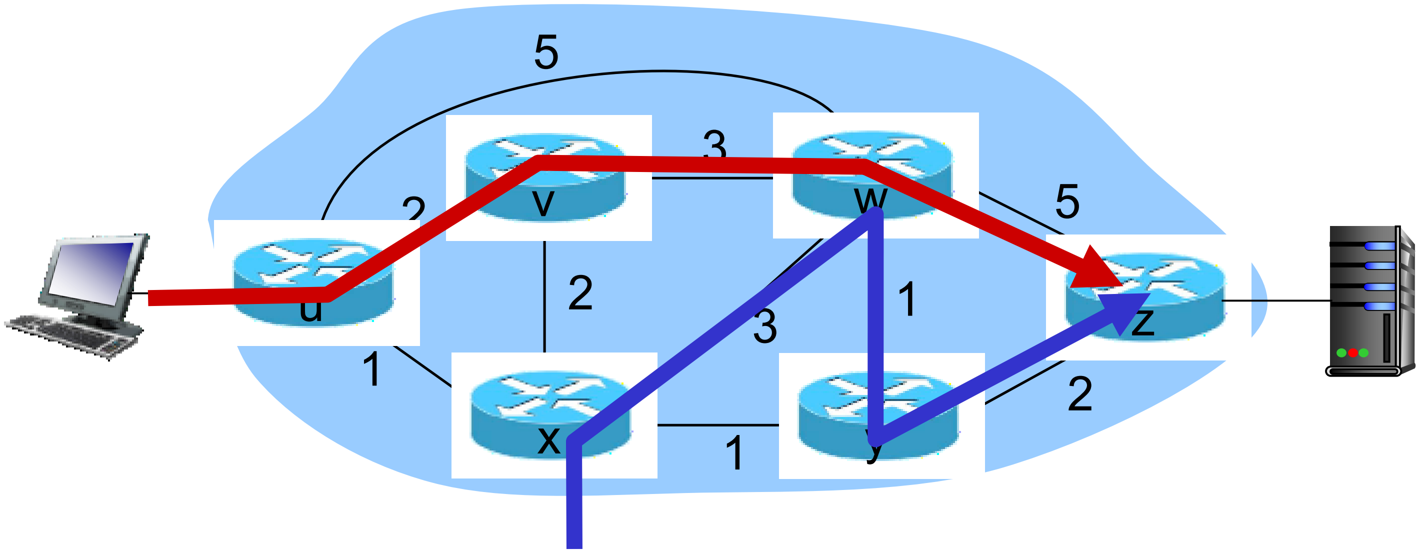
Traffic engineering: difficult



Q: what if 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)

Additional Material

Traffic engineering: difficult

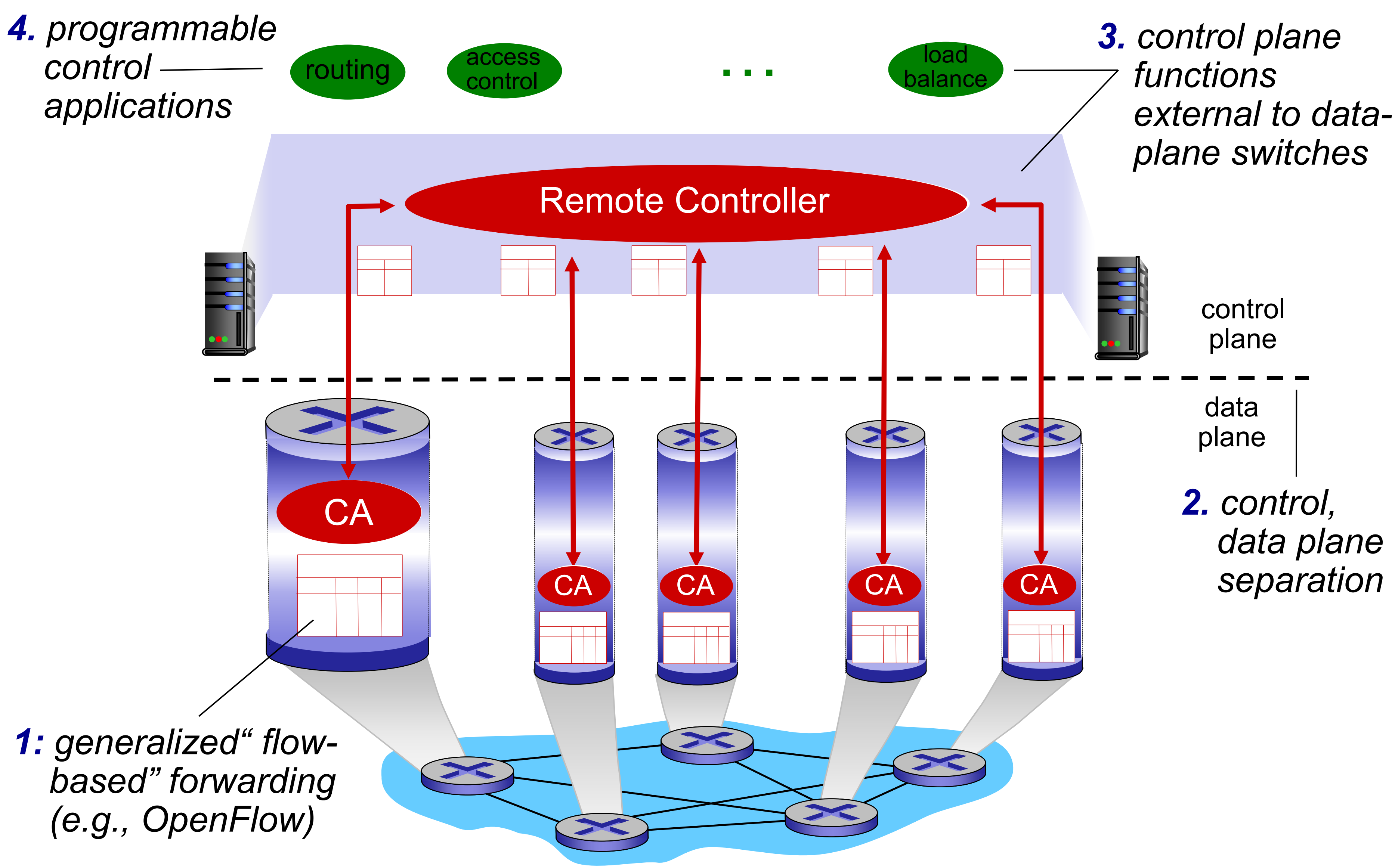


Q: what if w wants to route blue and red traffic differently?

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

Additional Material

Software defined networking (SDN)

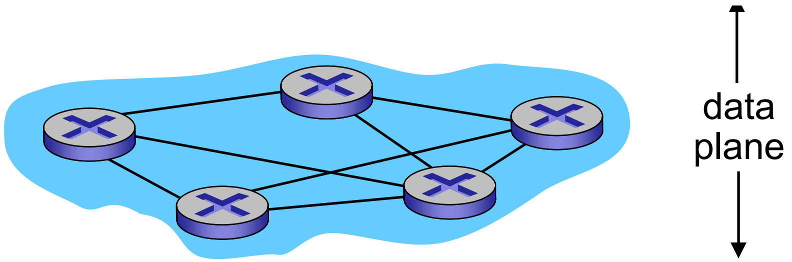


Additional Material

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)



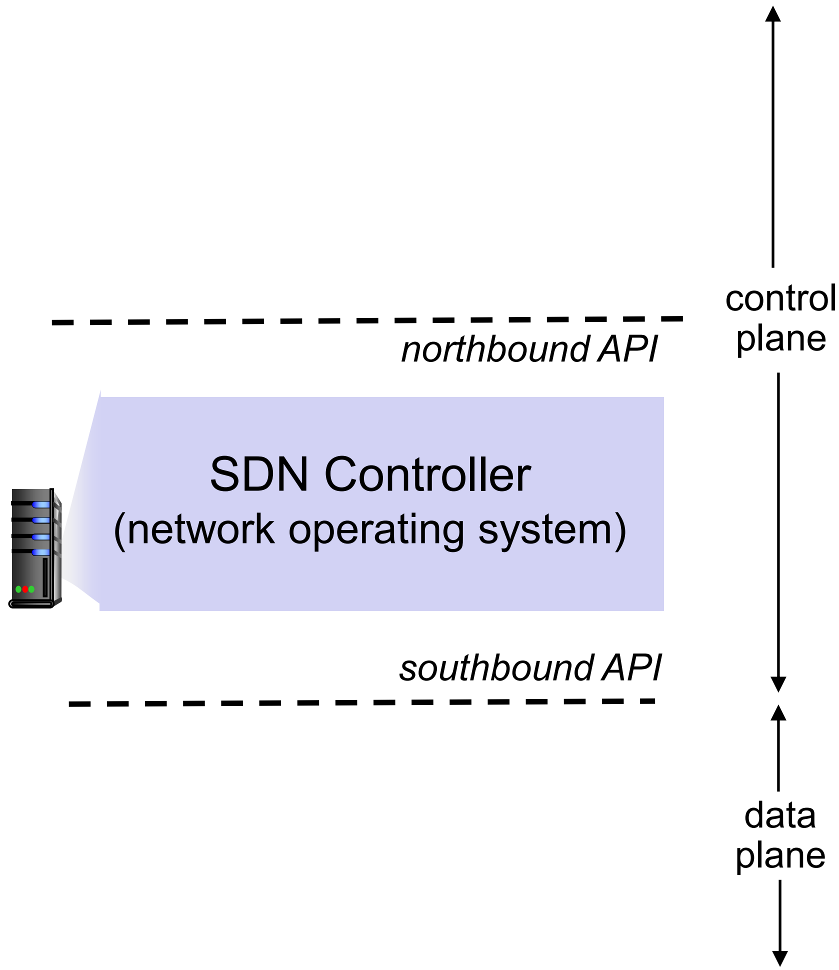
SDN-controlled switches

Additional Material

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

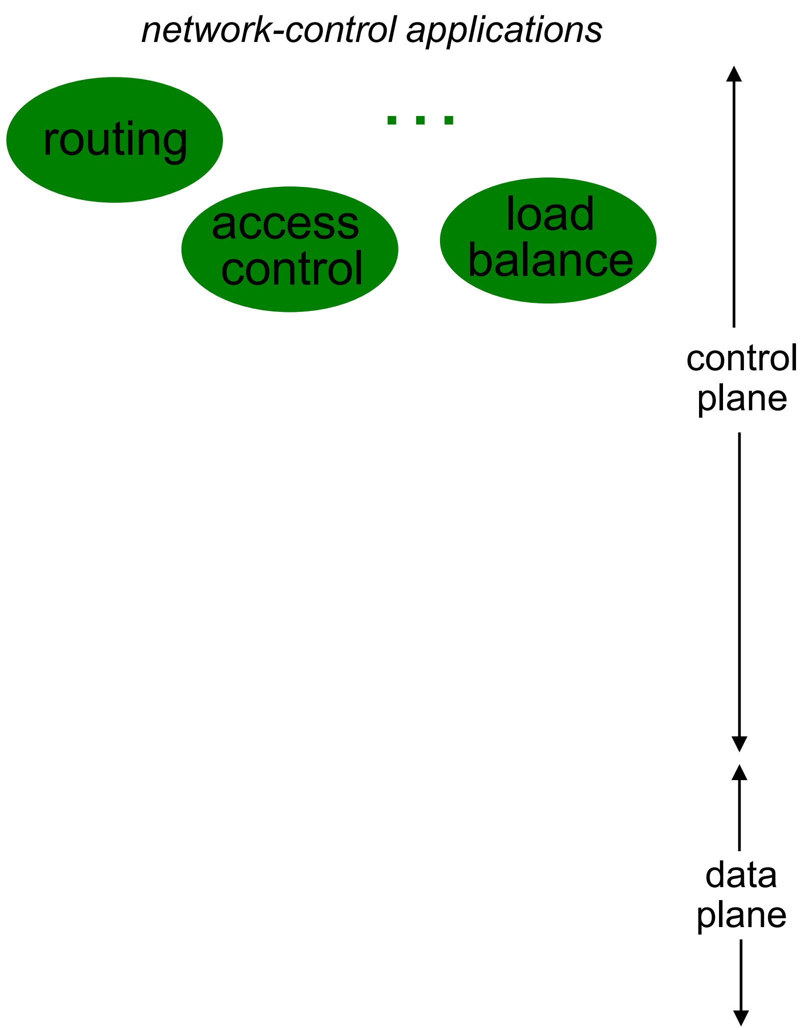


Additional Material

SDN perspective: control applications

network-control apps:

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



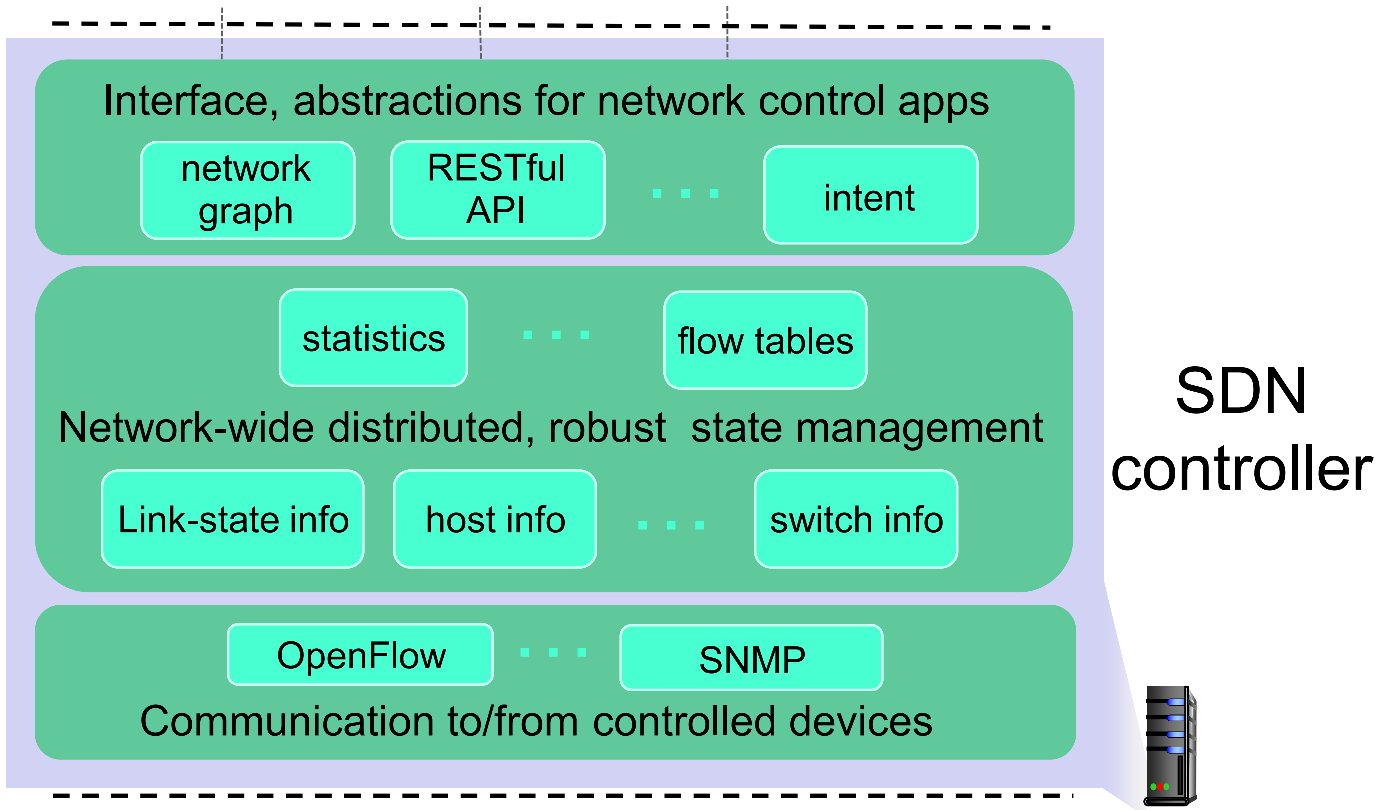
Additional Material

Components of SDN controller

Interface layer to network control apps: abstractions API

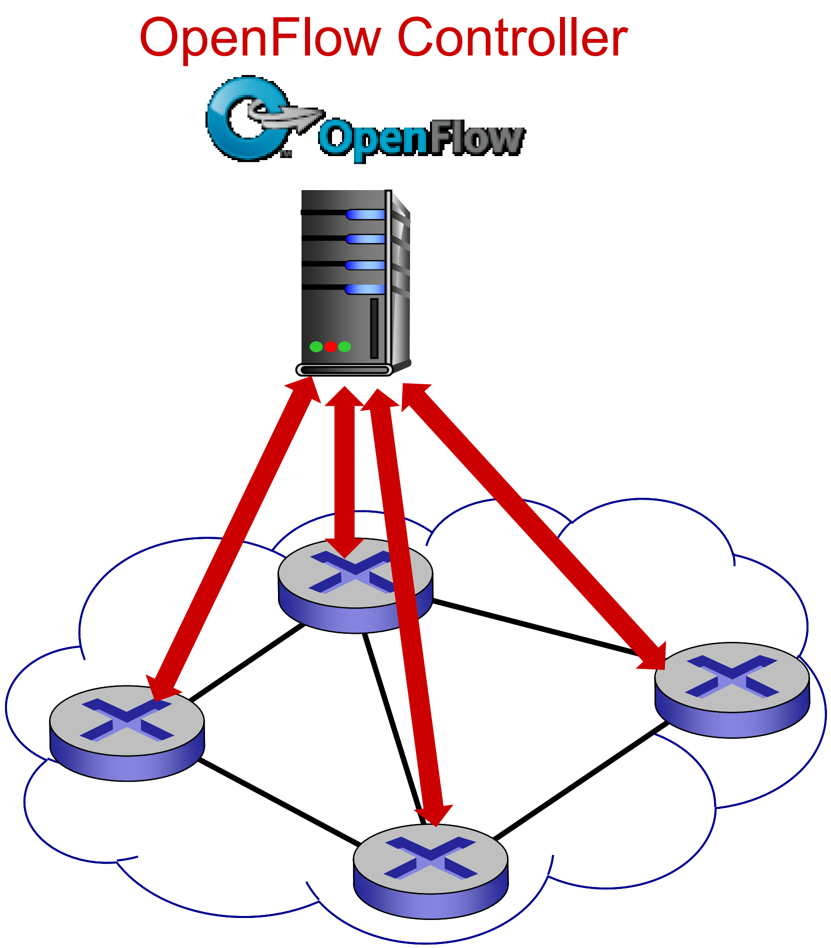
Network-wide state management layer: state of networks links, switches, services: a *distributed database*

***communication layer*:** communicate between SDN controller and controlled switches



Additional Material

OpenFlow protocol



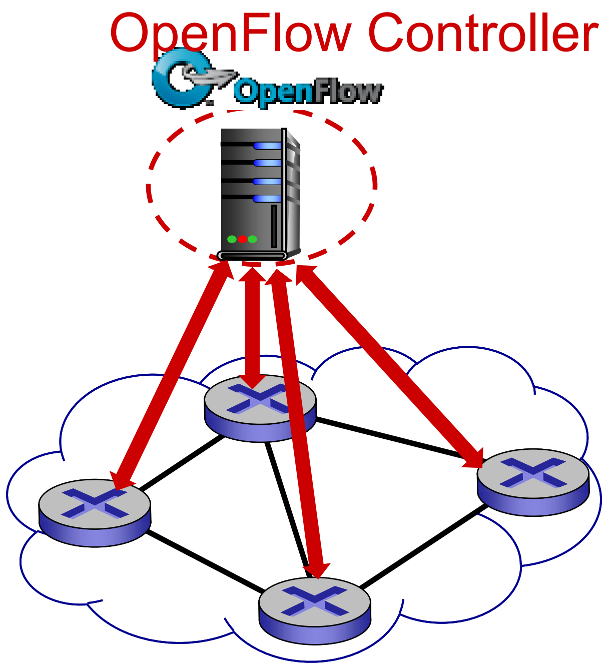
- operates between controller, switch
- TCP used to exchange messages
  - optional encryption
- three classes of OpenFlow messages:
  - controller-to-switch
  - asynchronous (switch to controller)
  - symmetric (misc)

Additional Material

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

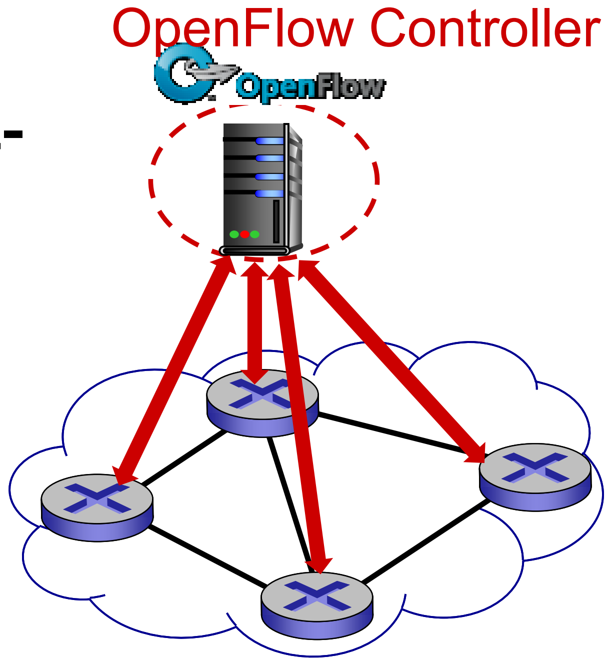


Additional Material

OpenFlow: switch-to-controller messages

Key switch-to-controller messages

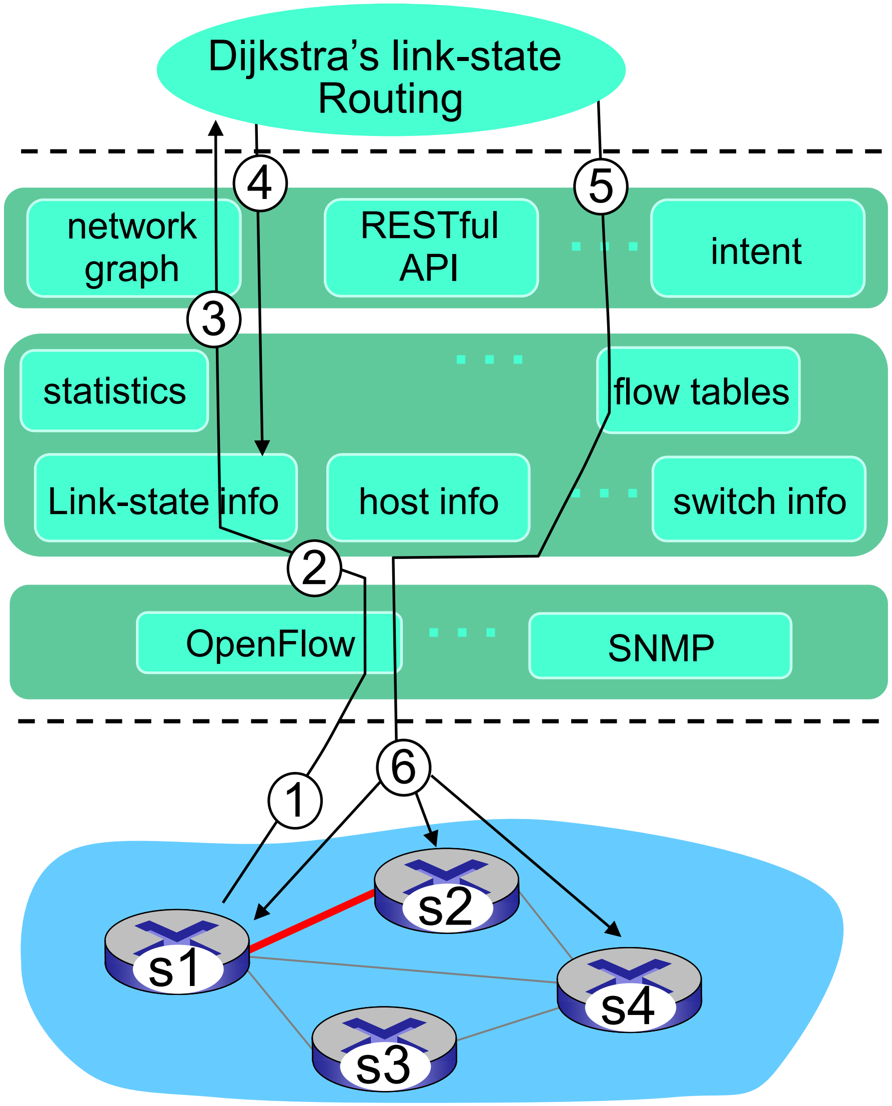
- *packet-in:* transfer packet (and its control) to controller. See packet-out message from controller
- *flow-removed:* flow table entry deleted at switch
- *port status:* inform controller of a change on a port.



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

Additional Material

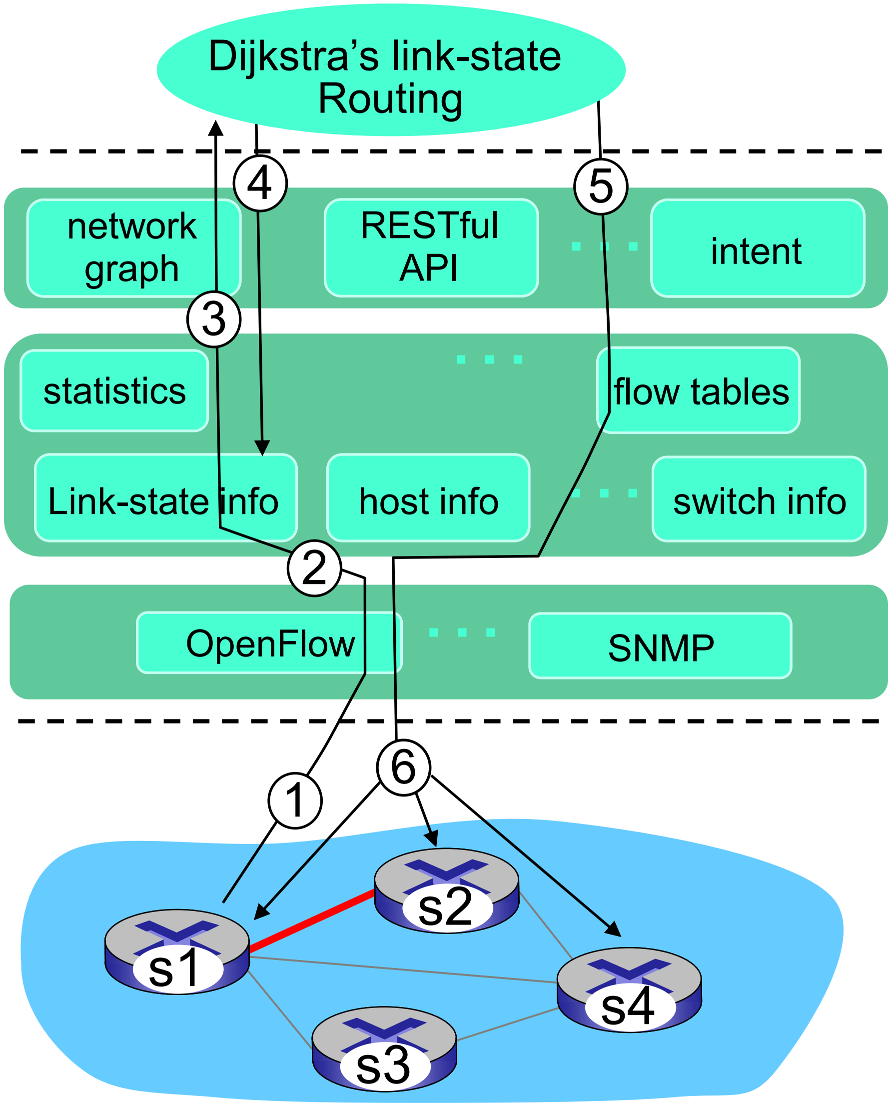
SDN: control/data plane interaction example



1. S1, experiencing link failure using OpenFlow port status message to notify controller
2. SDN controller receives OpenFlow message, updates link status info
3. Dijkstra's routing algorithm application has previously registered to be called when ever link status changes. It is called.
4. Dijkstra's routing algorithm access network graph info, link state info in controller, computes new routes

Additional Material

SDN: control/data plane interaction example

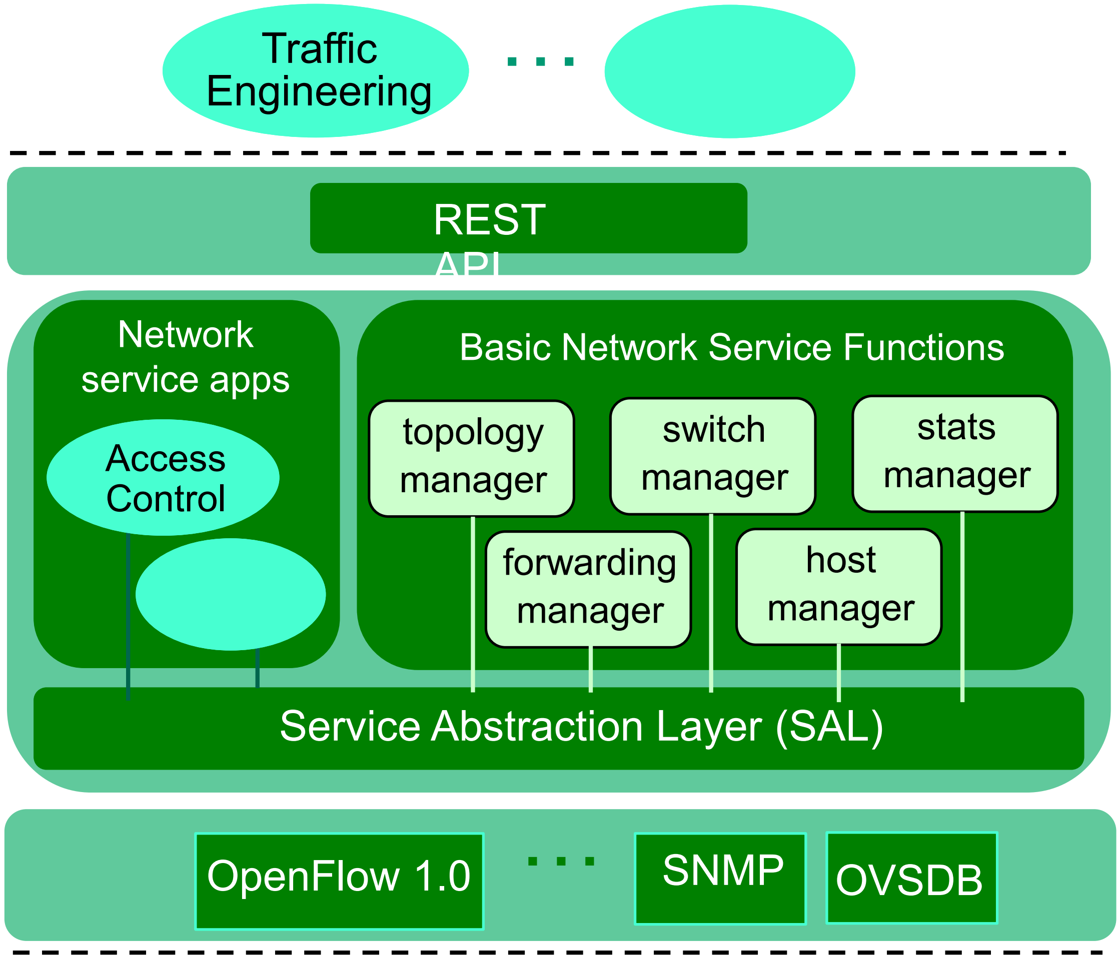


⑤ link state routing app interacts with flow-table-computation component in SDN controller, which computes new flow tables needed

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

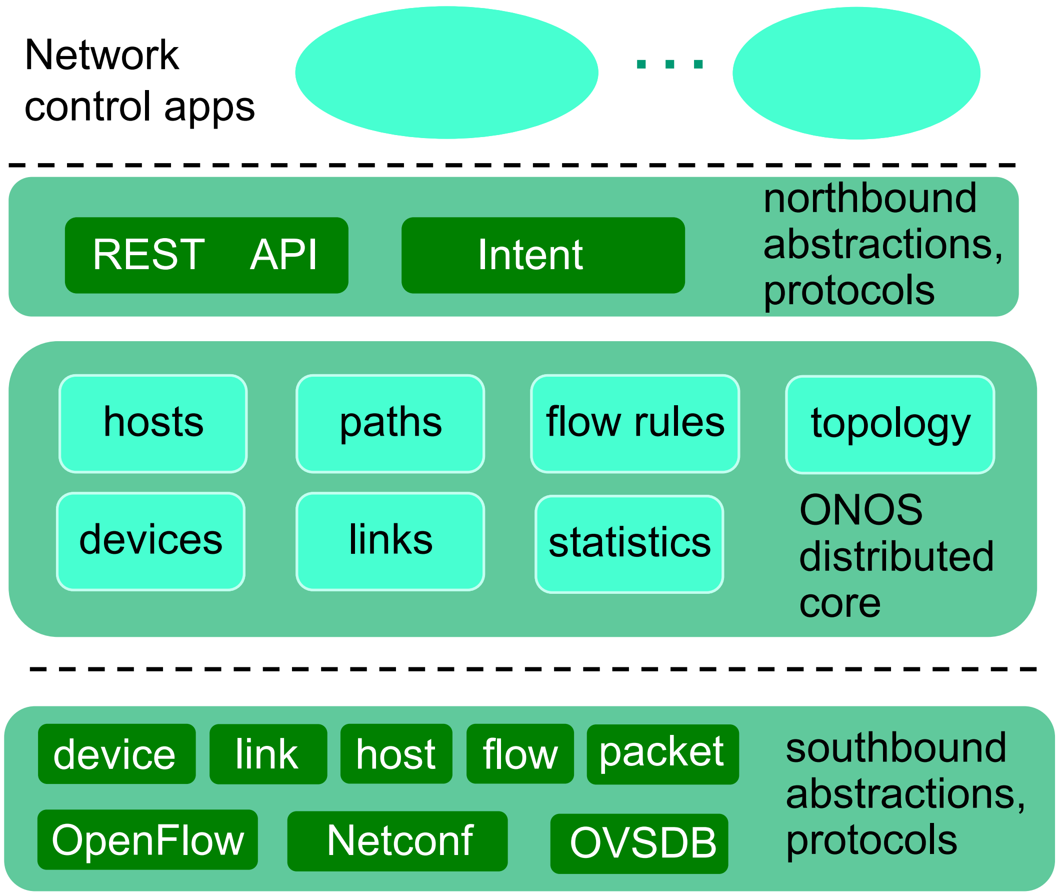
Additional Material

OpenDaylight (ODL) controller



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

ONOS controller



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

Additional Material

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

Chapter 5: outline

5.1 introduction

5.2 routing protocols

- link state
- distance vector

5.3 intra-AS routing in the Internet: OSPF

5.4 routing among the ISPs: BGP

5.5 The SDN control plane

5.6 ICMP: The Internet Control Message Protocol

5.7 Network management and SNMP

ICMP: Internet Control Message Protocol

- Used by hosts & routers to communicate network-level information
  - error reporting: unreachable host, network, port, protocol
  - echo request/reply (used by ping)
- Network-layer "above" IP:
  - ICMP msgs carried in IP datagrams
- ICMP message: type, code plus IP header and (at least) 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

ICMP packet

- ICMP packet encapsulated in IPv4, with IP protocol number 1 (one)
  - ICMP is above (and exploits) IP, but not transport layer!
- ICMP Header: 8 bytes
  - first 4 bytes with fixed format: type (1 byte), code (1 byte), checksum (2 bytes)
  - last 4 bytes depend on the type/code
- ICMP Data: variable size
  - the entire IP header and at least the first eight bytes of data from the IP packet that caused the error message
  - useful for the host to match the message to the appropriate process

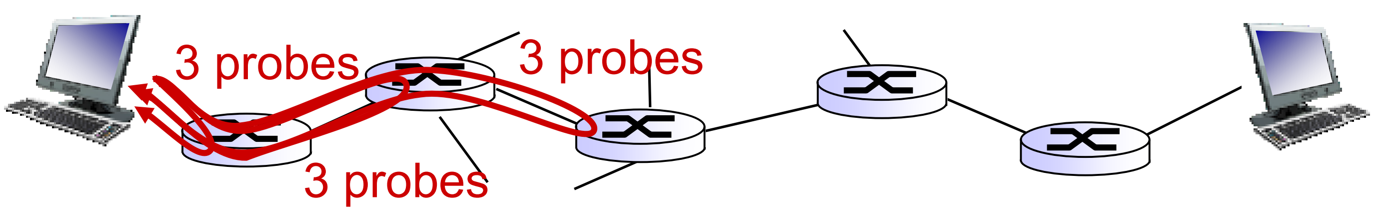
Traceroute and ICMP

- 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



Chapter 5: outline

What is network management?

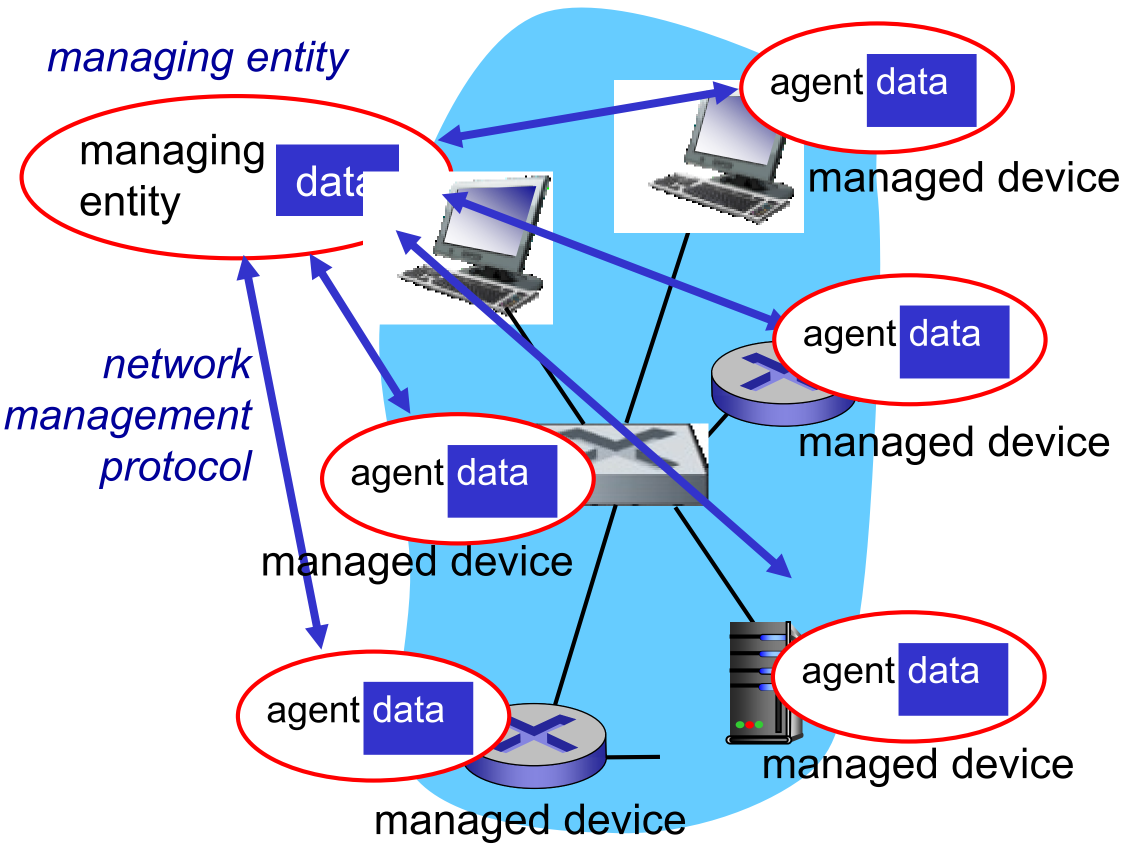
- Autonomous Systems (aka "network"): 1000s of interacting hardware/software components
- Other complex systems requiring monitoring, control:
  - jet airplane
  - nuclear power plant
  - others?

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

T. Saydam, T. Magedanz, "From Networks and Network Management into Service and Service Management", Journal of Networks and System Management, Vol. 4, No. 4, pp. 345-348, December 1996.

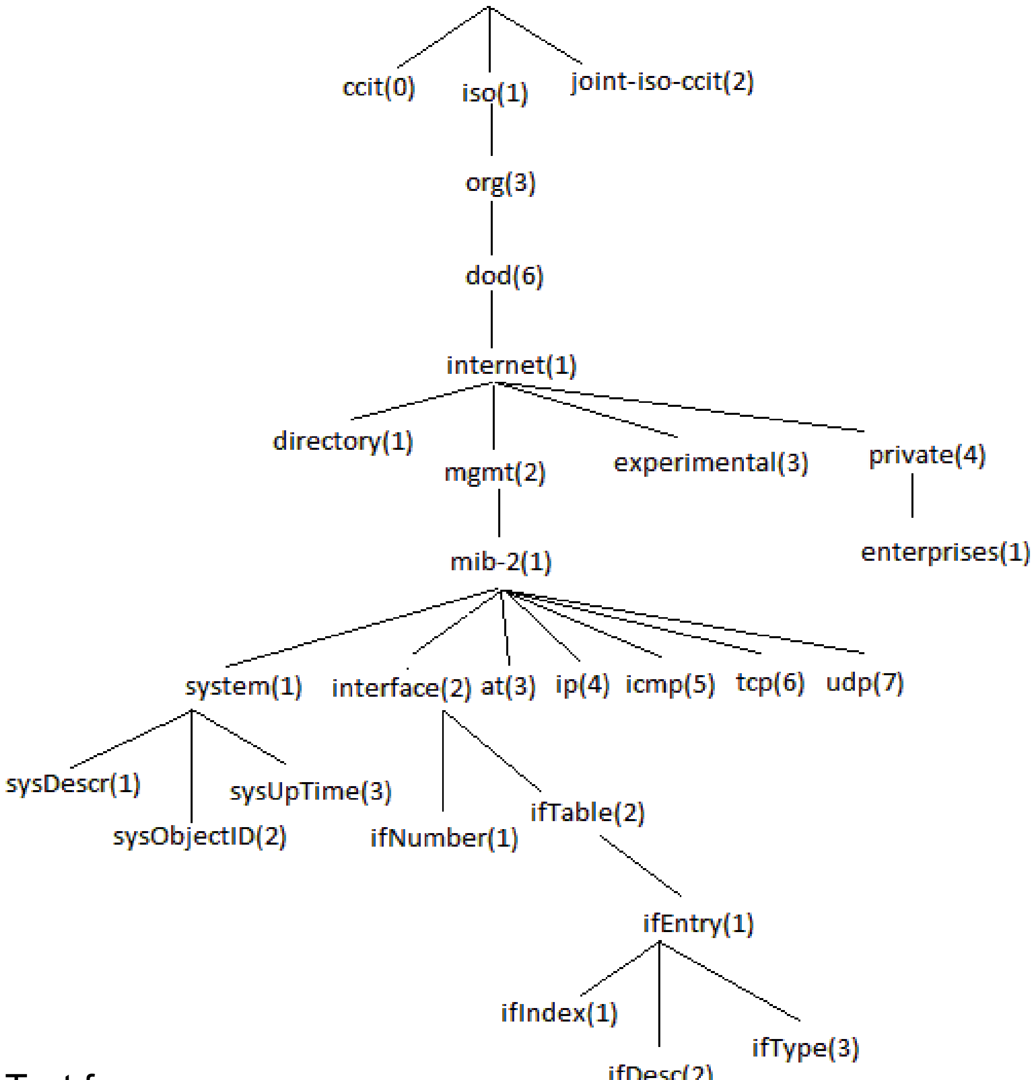
Infrastructure for network management

Definitions:



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

Management Information Base (MIB)

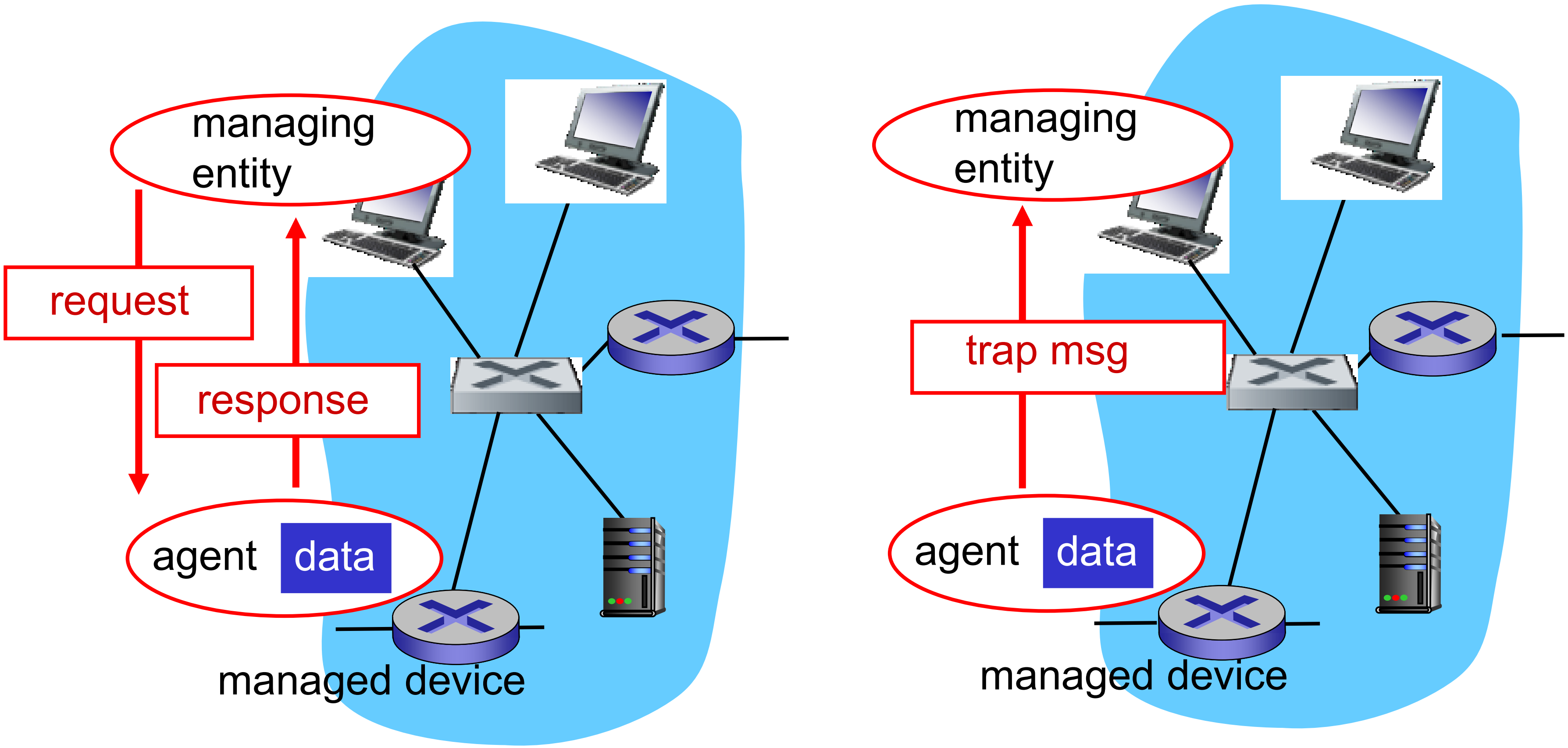


Text from
https://www.ibm.com/support/knowledgecenter/en/ssw_aix_72/com.ibm.aix.progcomc/mib.htm

- DB containing network management information organized as a tree
- Upper structure of the tree defined in RFC 1155/1213
- Internal nodes represent subdivisions by organization or function
- MIB variable values stored in leaves
- Children numbered sequentially from left to right, starting at 1
- Network management data for the Internet stored in the subtree reached by the path 1.3.6.1.2.1

SNMP: Simple Network Management Protocol

Two ways to convey MIB info, commands



request/response mode

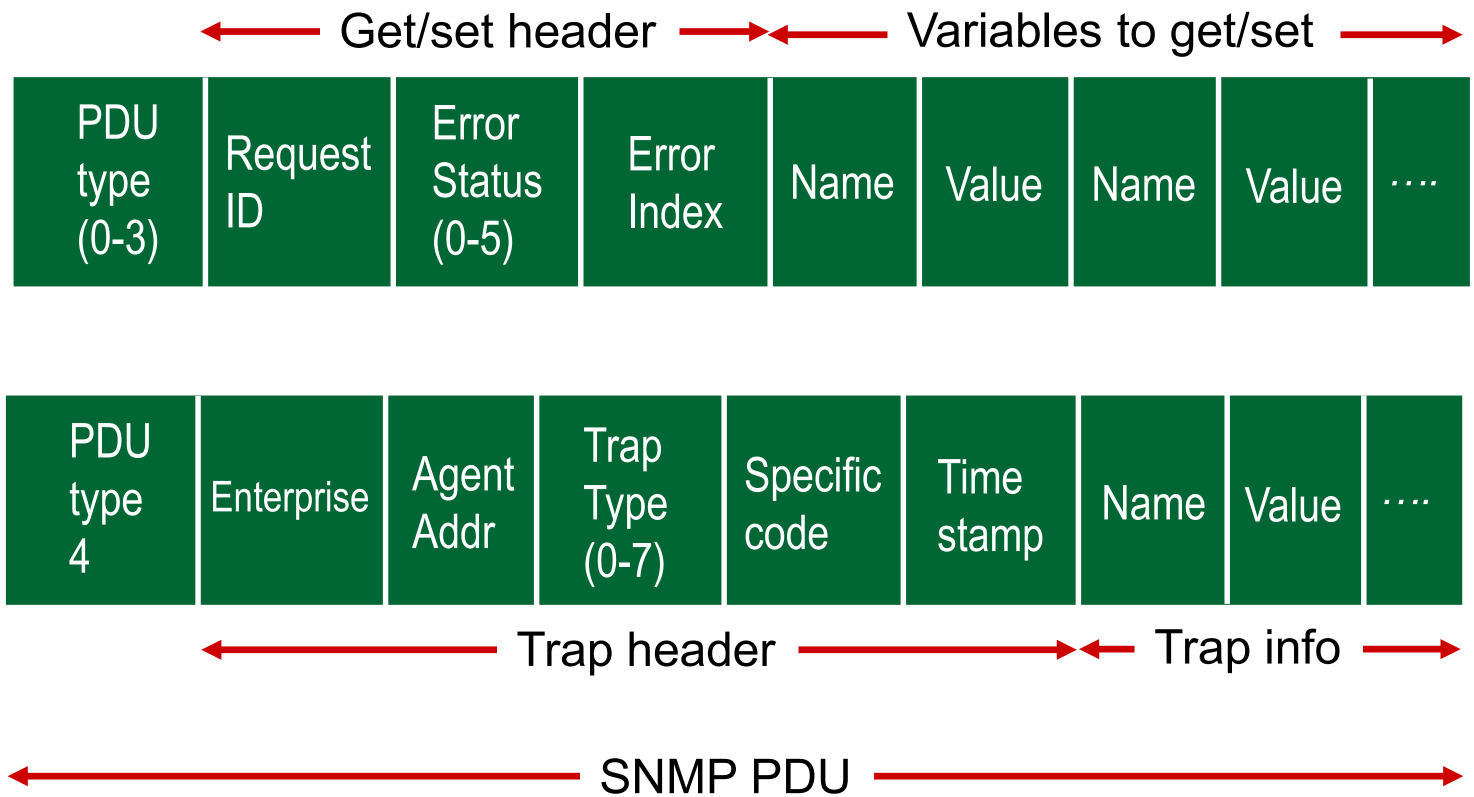
trap mode

SNMP protocol: message types

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

Additional Material

SNMP protocol: message formats



Chapter 5: summary

We've learned a lot!

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

next stop: link layer!