SOFTENG 364: Computer Networks

Network Layer: The Control Plane

Kurose and Ross, chapter 5

ENGINEERING

Learning Outcomes

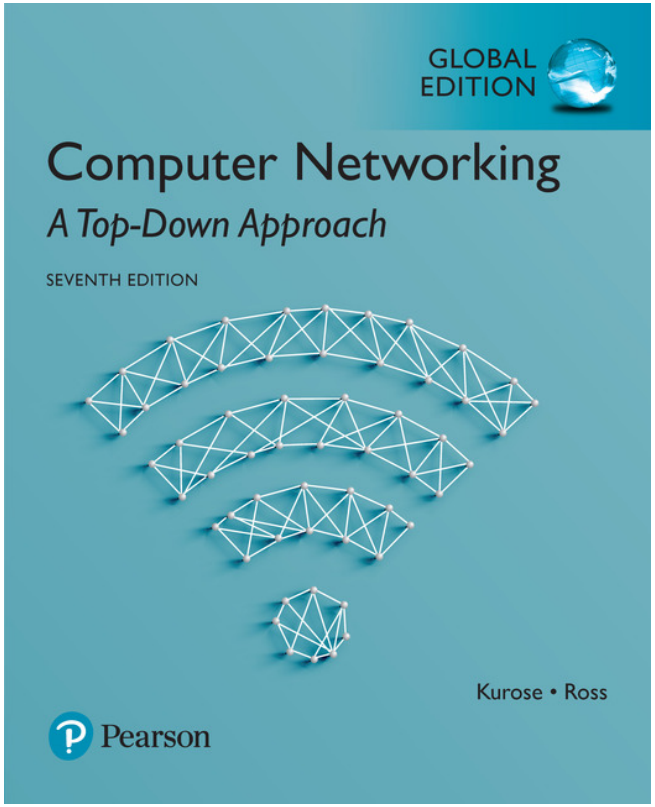
By the end of this module you should be able to:

- Describe the functions of the control plane in the network layer
- Explain how Dijsktra's algorithm works
- Explain how the Distance Vector algorithm works
- Contrast how the Dijsktra's and Distance Vector algorithm work
- Explain how routing works within and between Autonomous Systems
- Describe the key features of how the OSPF and BGP protocols
- Illustrate the differences between the OSPF and BGP protocols
- Describe the key features of Software Defined Networks (SDN)
- Contrast the key differences between per-router control and SDN control
- Demonstrate how the ICMP and SNMP protocols work

References

Computer Networking: A Top Down Approach, 7th edition (2016). *By J. Kurose & K. Ross.*

- Chapter 5



THE UNIVERSITY OF AUCKLAND
Te Whare Wananga o Tamaki Makaurau
NEW ZEALAND
ENGINEERING

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:

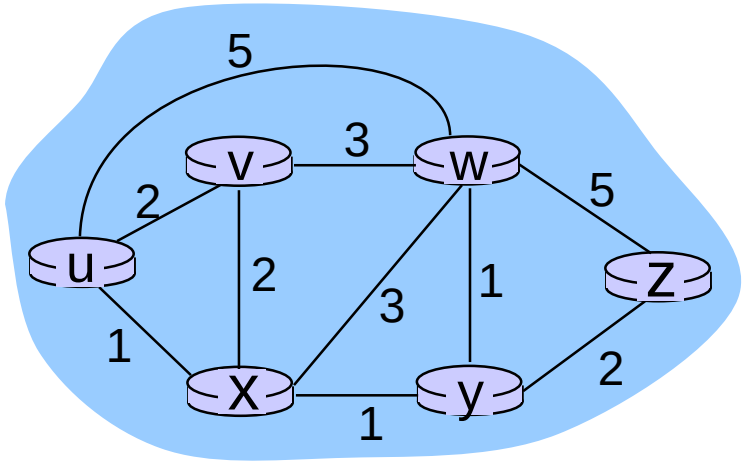
- per-router control (traditional)
- logically centralized control (software defined networking)

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 network



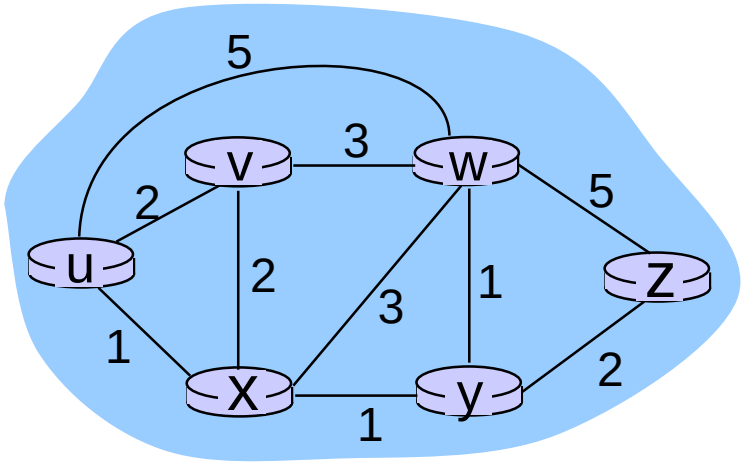
graph: G = (N,E)

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

E = set of links = { (u,v), (u,x), (v,x), (v,w), (x,w), (x,y), (w,y), (w,z), (y,z) }

aside: graph abstraction is useful in other 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

A link-state routing algorithm

THE UNIVERSITY OF AUCKLAND
Te Whare Wananga o Tamaki Makaurau
NEW ZEALAND

ENGINEERING

Dijkstra's algorithm

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

Notation:

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

Dijsktra's algorithm

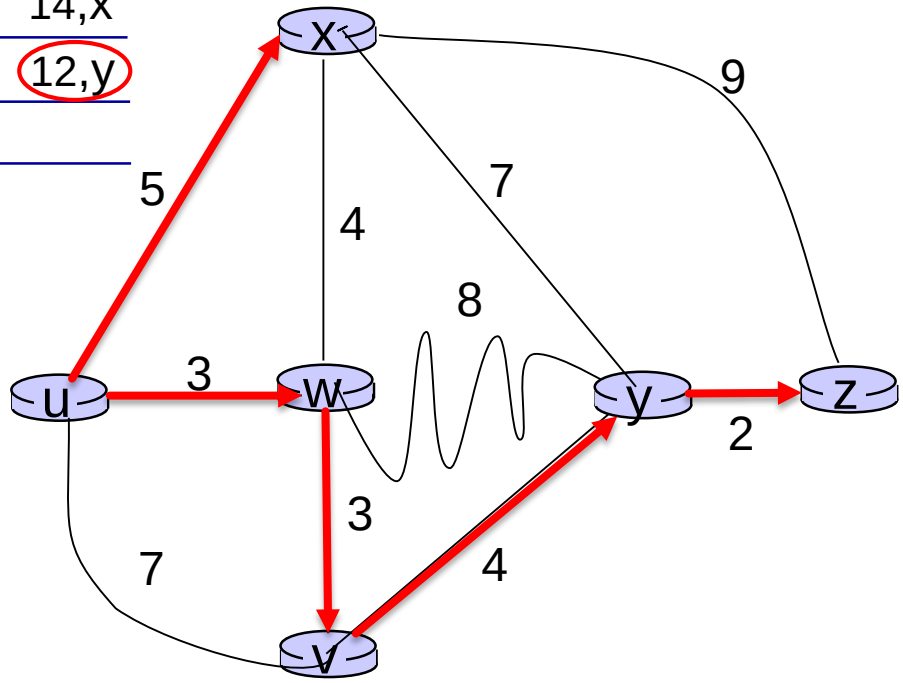
```
1  Initialization:
2    N' = {u}
3    for all nodes v
4      if v adjacent to u
5          then D(v) = c(u,v)
6      else D(v) = ∞
7
8  Loop
9     find w not in N' such that D(w) is a minimum
10    add w to N'
11    update D(v) for all v adjacent to w and not in N' :
12       D(v) = min( D(v), D(w) + c(w,v) )
13    /* new cost to v is either old cost to v or known
14     shortest path cost to w plus cost from w to v */
15 until all nodes in N'
```

Dijkstra's algorithm: example

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

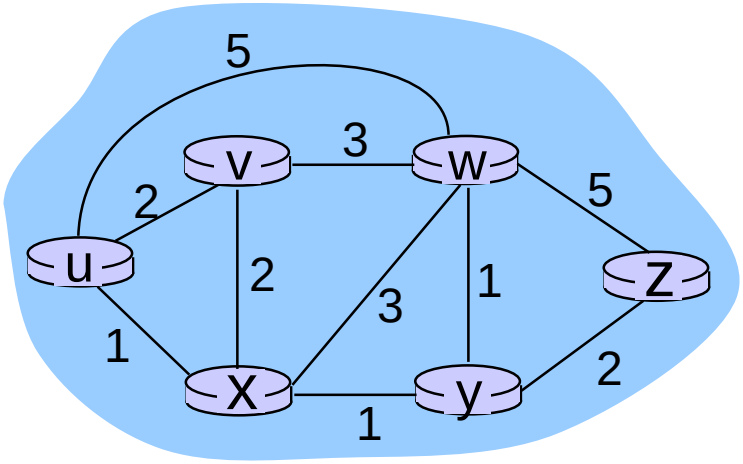
Notes:

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



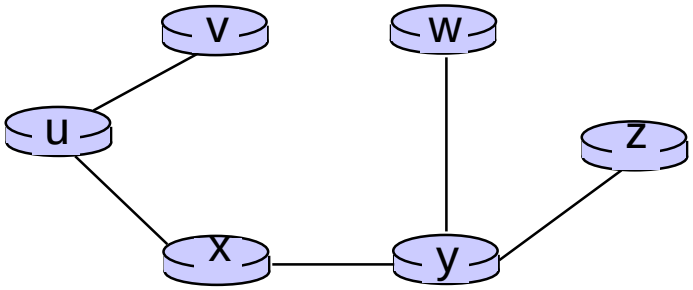
Dijkstra's algorithm: example (2)

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



Dijkstra's algorithm: example (2)

Resulting shortest-path tree from u:



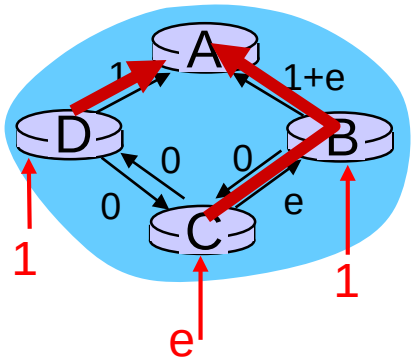
Resulting forwarding table in u:

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

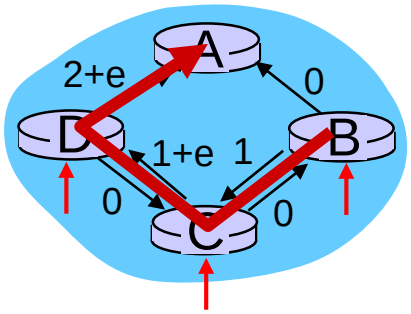
Dijkstra's algorithm: discussion

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

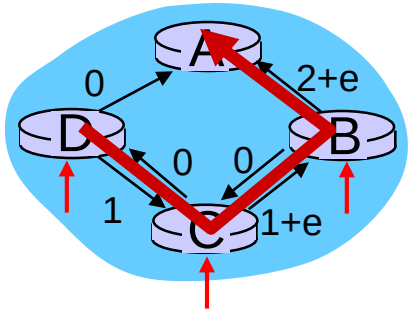
Oscillations possible:
- e.g., support link cost equals amount of carried traffic:



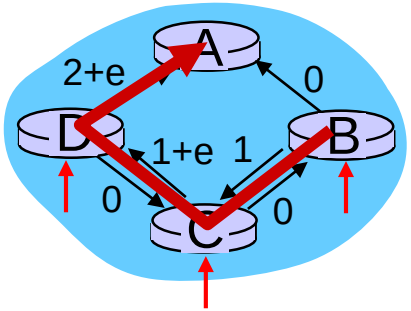
initially



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

Distance vector algorithm

THE UNIVERSITY OF AUCKLAND
Te Whare Wananga o Tamaki Makaurau
NEW ZEALAND
ENGINEERING

Bellman-Ford equation (dynamic programming)

let

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

then

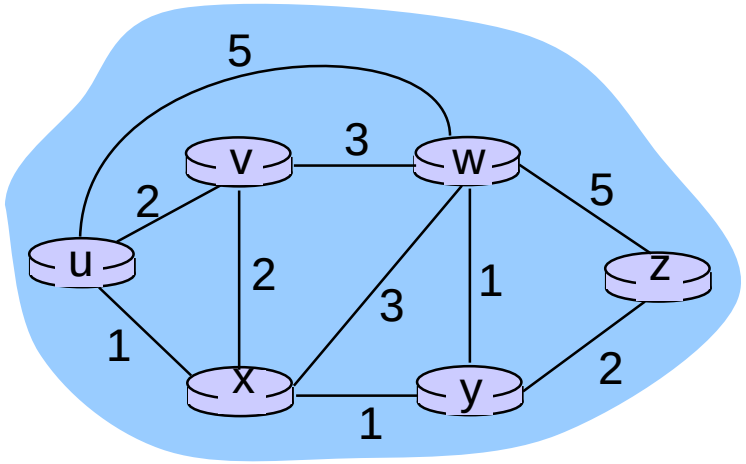
$$d_x(y) = \min_v \{c(x,v) + d_v(y)\}$$

$d_v(y)$: cost from neighbor v to destination y

$c(x,v)$: cost to neighbor v

$\min_v$: *min* taken over all neighbors v of x

Bellman-Ford example



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

B-F equation says:

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

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

Distance vector algorithm

$D_x(y)$ = estimate of least cost from x to y

- x maintains distance vector $\mathbf{D}_x = [D_x(y): y \in N]$

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

THE UNIVERSITY OF AUCKLAND
Te Whare Wananga o Tamaki Makaurau
NEW ZEALAND
ENGINEERING

Key idea:

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

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

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

Distance vector algorithm

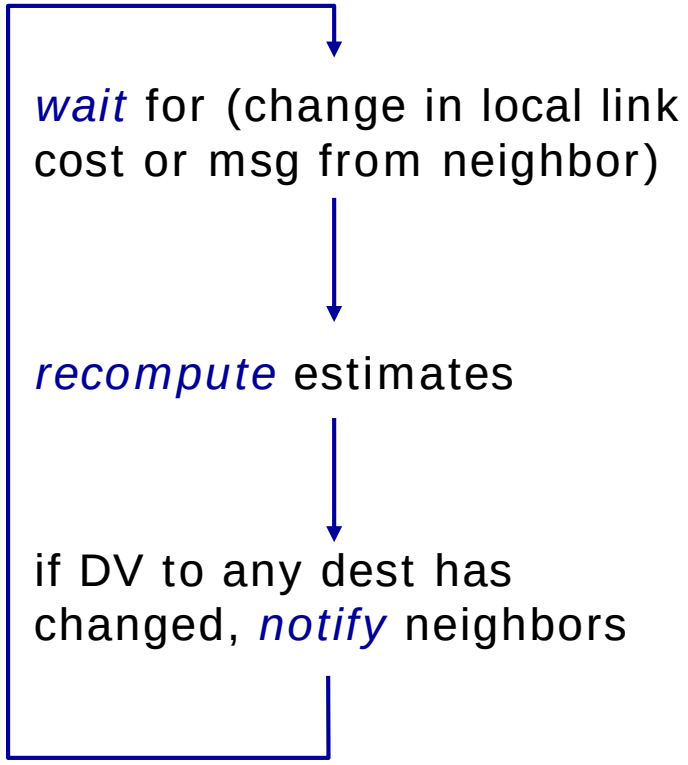
Iterative, asynchronous: each local iteration caused by:

- Local link cost change
- DV update message from neighbor

Distributed:

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

each node:



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

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

node x table

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

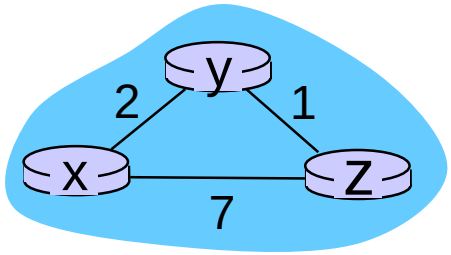
from \ cost to	x	y	z
x	0	2	3
y	2	0	1
z	7	1	0

node y table

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

node z table

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



time

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

node y table

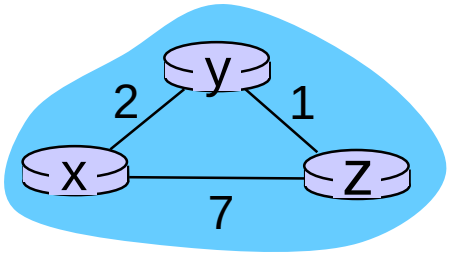
from	cost to x	y	z
x	∞	∞	∞
y	2	0	1
z	∞	∞	∞

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

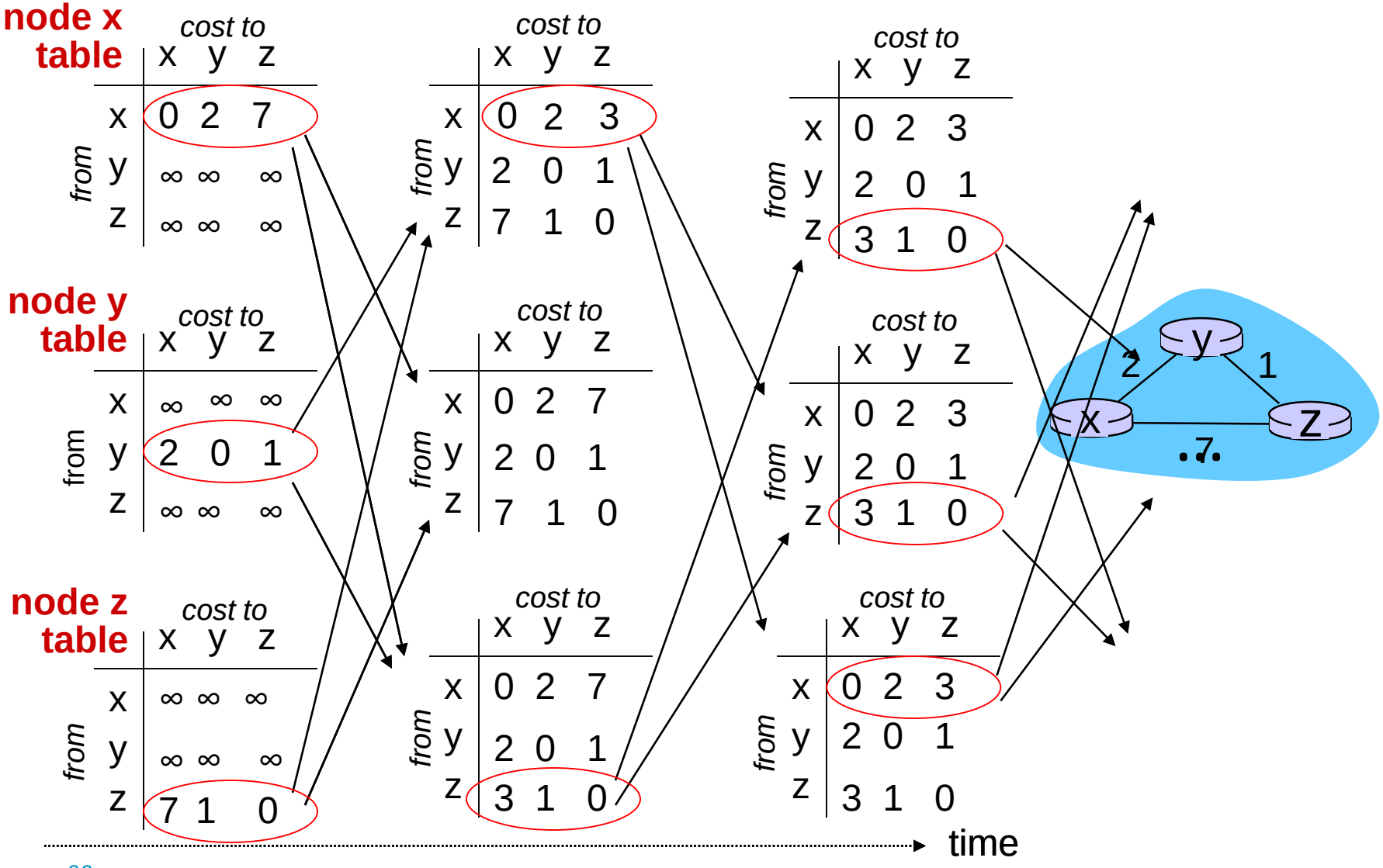
node z table

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



time

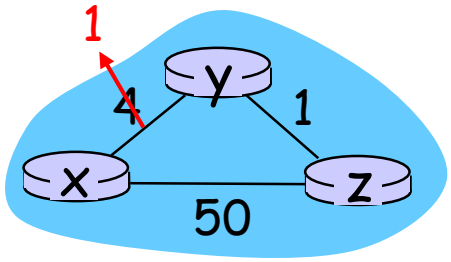


When do we stop?

Link cost changes

link cost changes:

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



"good news travels fast"

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

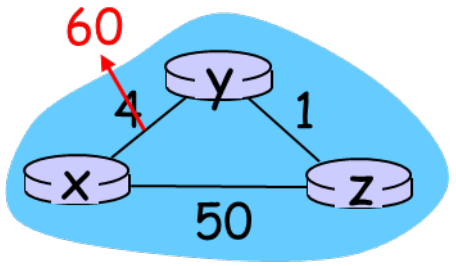
t_1 : *z* receives update from *y*, updates its table, computes new least cost to *x* , sends its neighbors its DV.

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

Link cost changes

Link cost changes:

- Node detects local link cost change
- *Bad news travels slow* - "count to infinity" problem!
- 44 iterations before algorithm stabilizes: see text



Poisoned reverse:

- If Z routes through Y to get to X :
  - Z tells Y its (Z's) distance to X is infinite (so Y won't route to X via Z)
- Will this completely solve count to infinity problem?

Comparison of LS and DV algorithms

THE UNIVERSITY OF AUCKLAND
Te Whare Wananga o Tamaki Makaurau
NEW ZEALAND
ENGINEERING

Message complexity

LS: with n nodes, E links, O(nE) msgs sent

DV: exchange between neighbors only

- convergence time varies

Speed of convergence

LS: $O(|N|^2)$ algorithm requires $O(|N|\ |E|)$ msgs

- may have oscillations

DV: convergence time varies

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

Robustness: what happens if router malfunctions?

LS:

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

DV:

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

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 routing

THE UNIVERSITY OF AUCKLAND
Te Whare Wananga o Tamaki Makaurau
NEW ZEALAND
ENGINEERING

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

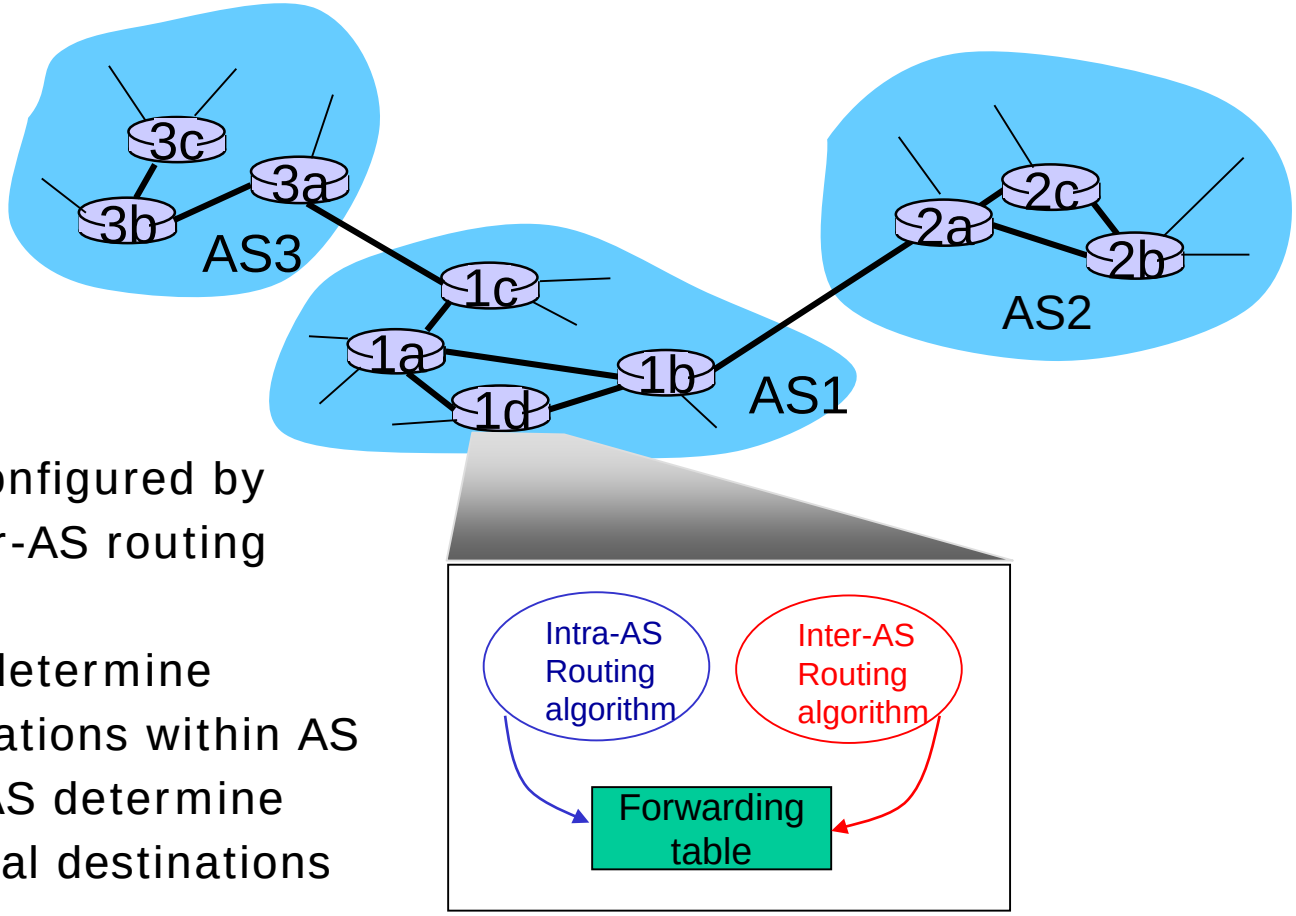
Intra-as routing

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

Inter-as routing

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

Interconnected ASes



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

- Intra-as routing determine entries for destinations within AS
- Inter-as & intra-AS determine entries for external destinations

Inter-AS tasks

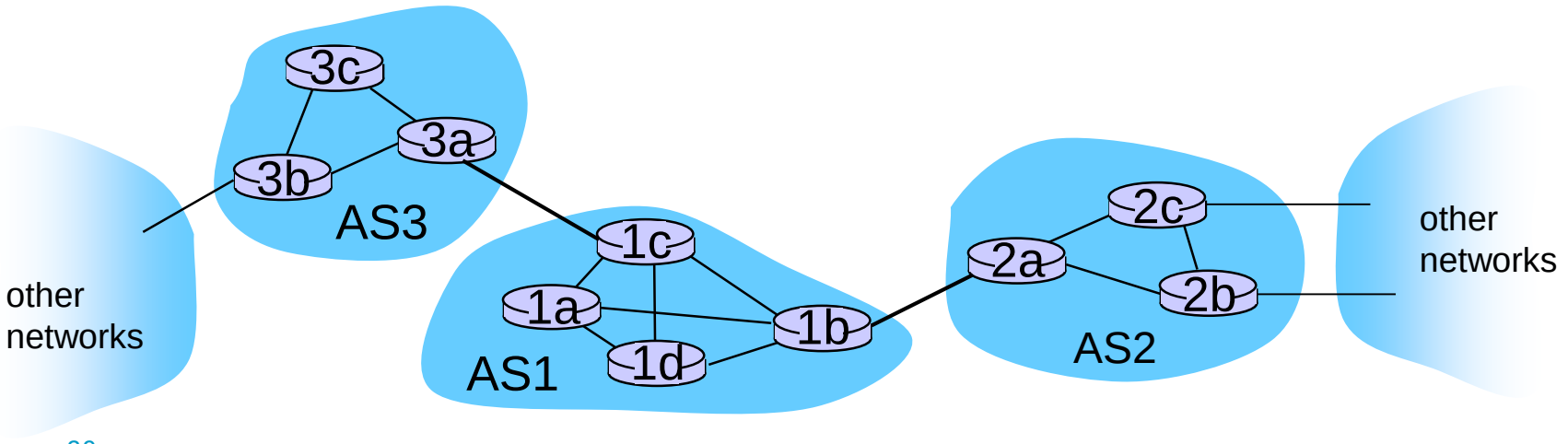
Suppose router in AS1 receives datagram destined outside of AS1:

- Router should forward packet to gateway router, but which one?

AS1 must:

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

Job of inter-AS routing!



Intra-AS Routing

Also known as *Interior Gateway Protocols (IGP)*

Most common intra-AS routing protocols:

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

Open Shortest Path First

Also known as OSPF

"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

Intermediate System - Intermediate System (IS-IS routing) protocol: nearly identical to OSPF

OSPF "advanced" features

Security: all OSPF messages authenticated (to prevent malicious intrusion)

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

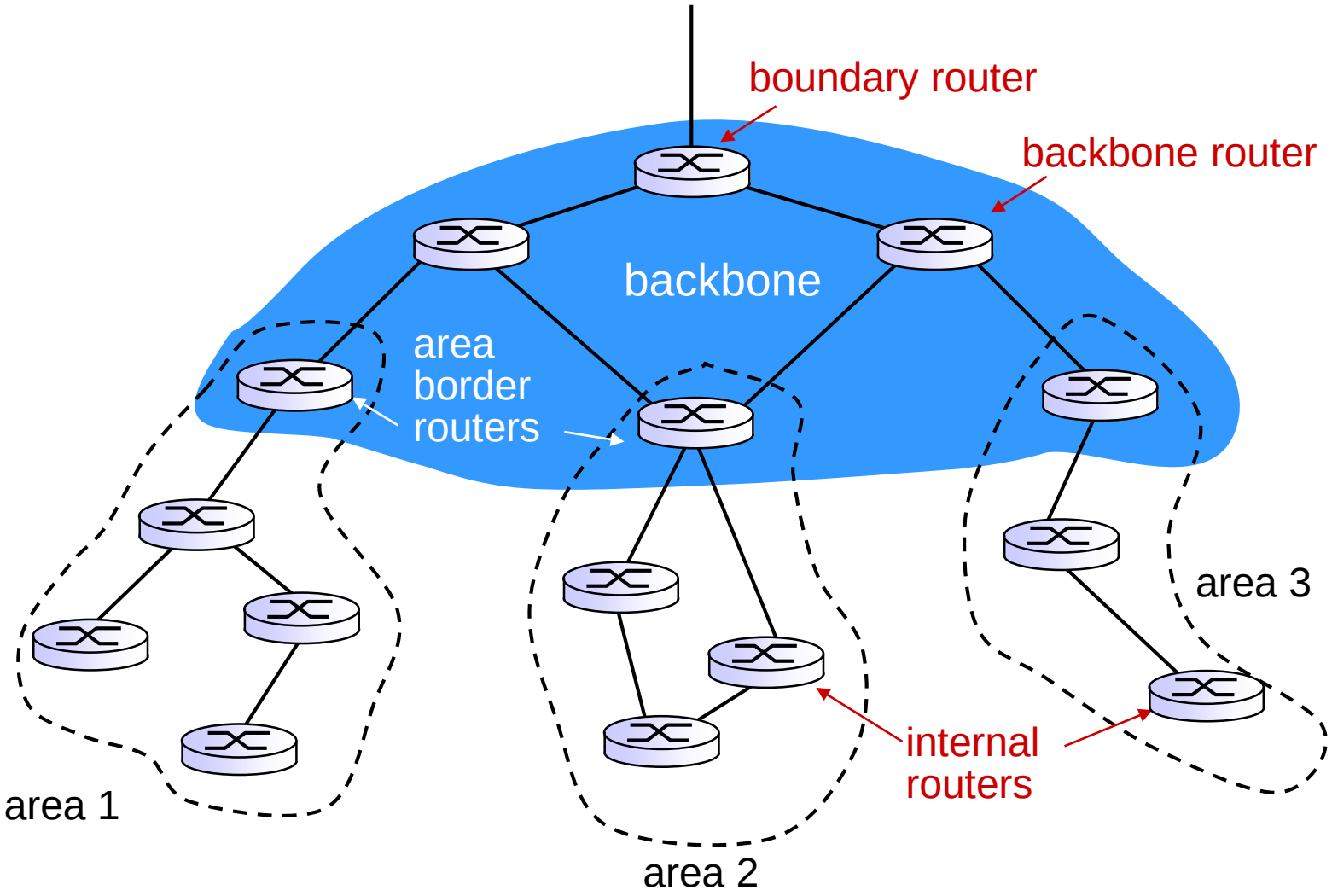
For each link, multiple cost metrics for different *TOS* (e.g., satellite link cost set low for best effort ToS; high for real-time ToS)

Integrated uni- and *multi-cast* support:

- Multicast OSPF (MOSPF) uses same topology data base as OSPF

Hierarchical OSPF in large domains.

Hierarchical OSPF



Hierarchical OSPF

Two-level hierarchy: local area, backbone.

- Link-state advertisements only in area
- Each nodes has detailed area topology; only know direction (shortest path) to nets in other areas.

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

Backbone routers: run OSPF routing limited to backbone.

Boundary routers: connect to other AS'es.

Internet inter-AS routing: BGP

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

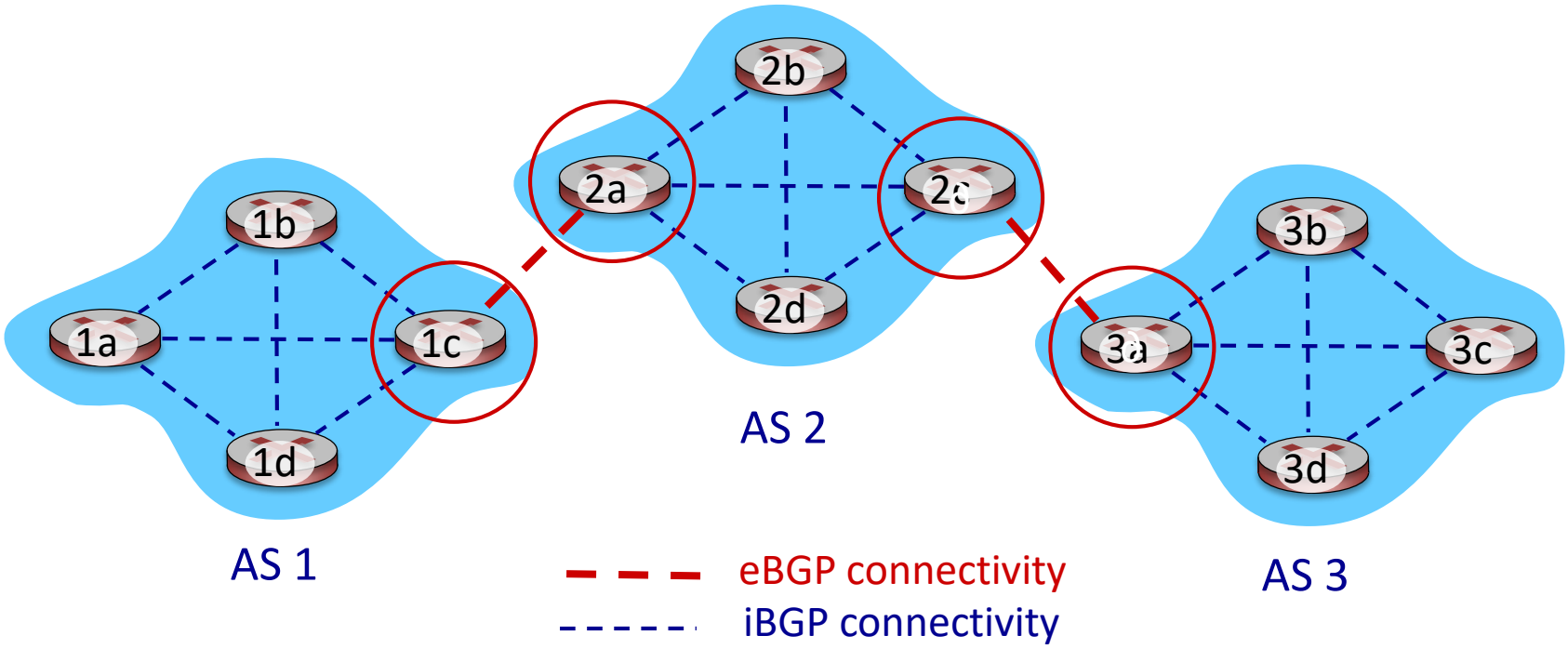
- "Glue that holds the Internet together"

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 rest of Internet: *"I am here"*

eBGP & iBGP connections



gateway routers run both eBGP and iBGP protools

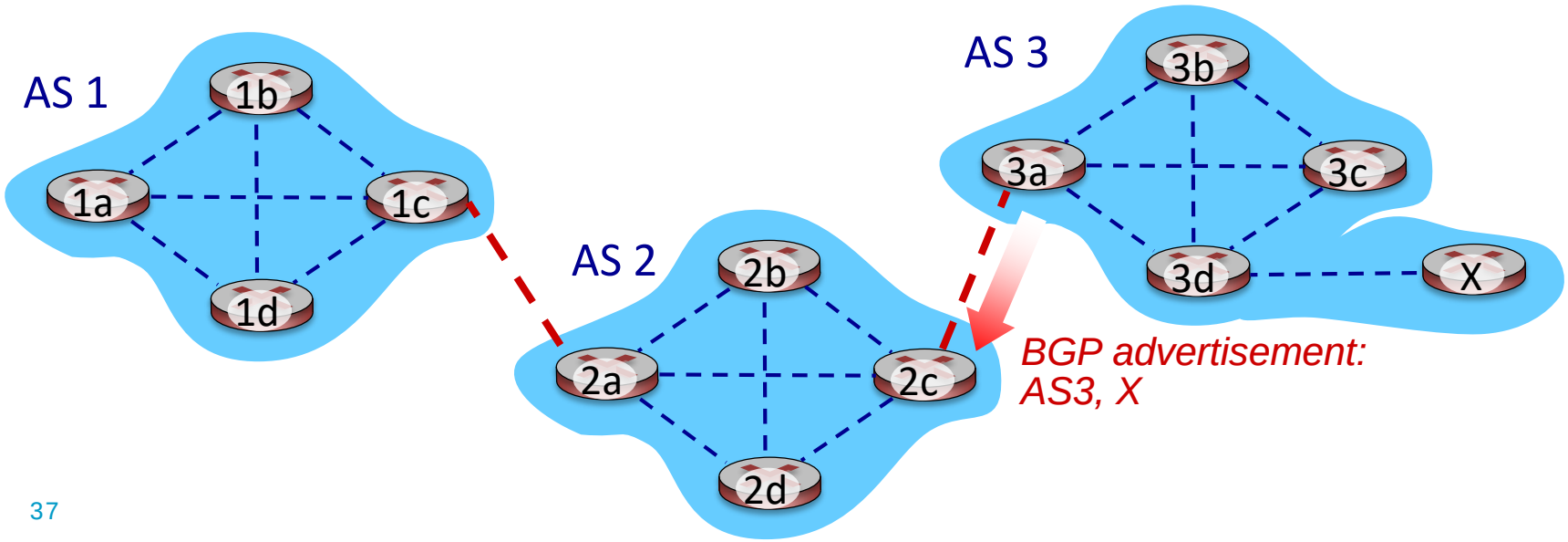
BGP basics

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



BGP routes and path attributes

THE UNIVERSITY OF AUCKLAND
Te Whare Wananga o Tamaki Makaurau
NEW ZEALAND
ENGINEERING

Advertised prefix includes BGP attributes

- Prefix + attributes = “route”

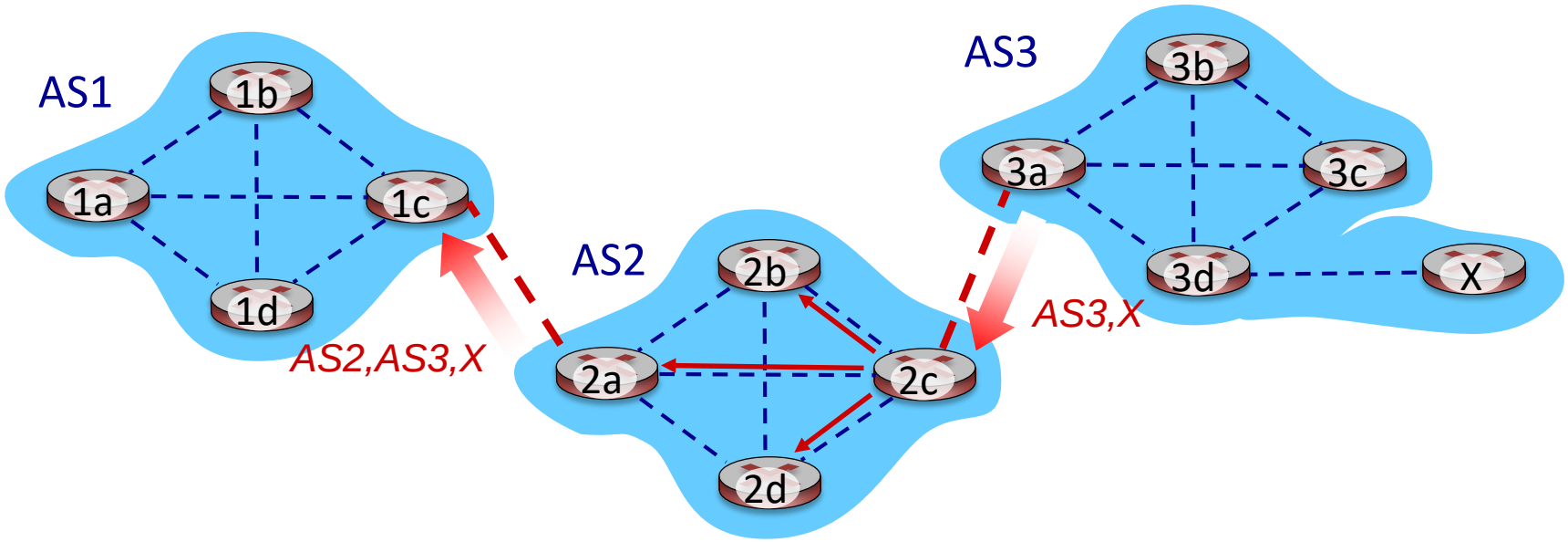
Two important attributes:

- 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 policies* to accept/decline path (e.g., never route through AS Y).
- AS policy also determines whether to *advertise* path to other neighboring ASes

BGP path advertisement

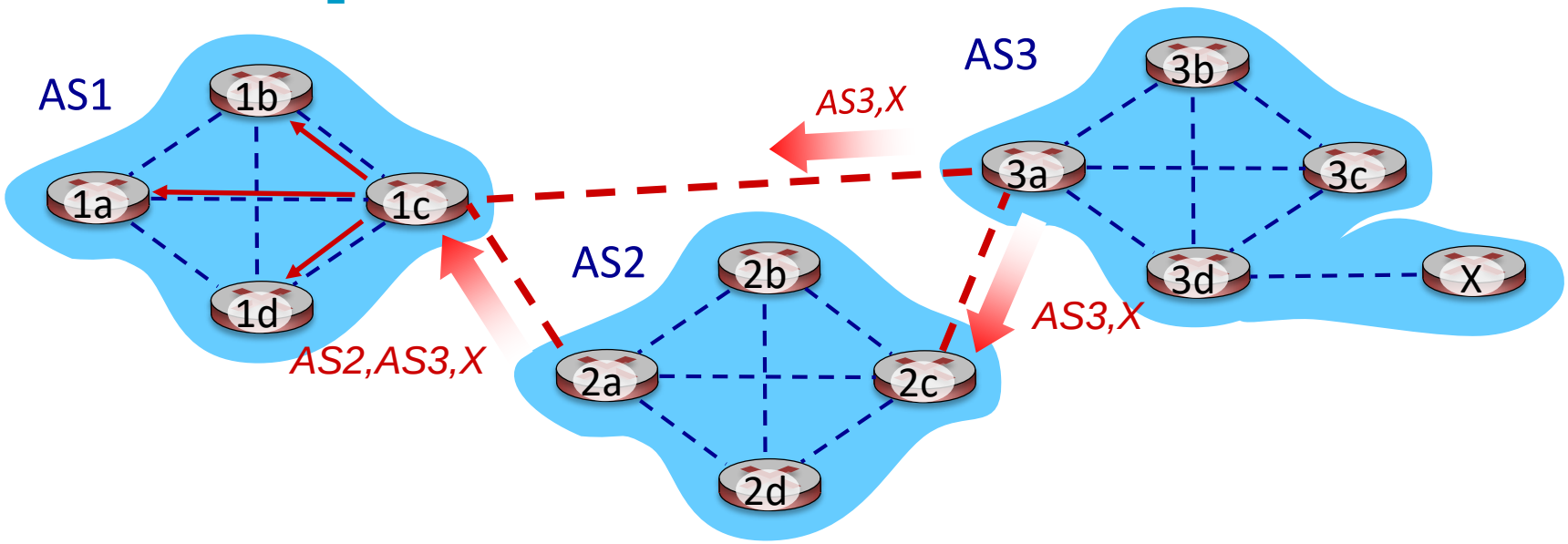


AS2 router 2c receives path advertisement AS3,X (via eBGP) from AS3 router 3a

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

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

BGP path advertisement



gateway router may learn about multiple paths to destination:

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

BGP messages

BGP messages exchanged between peers over TCP connection

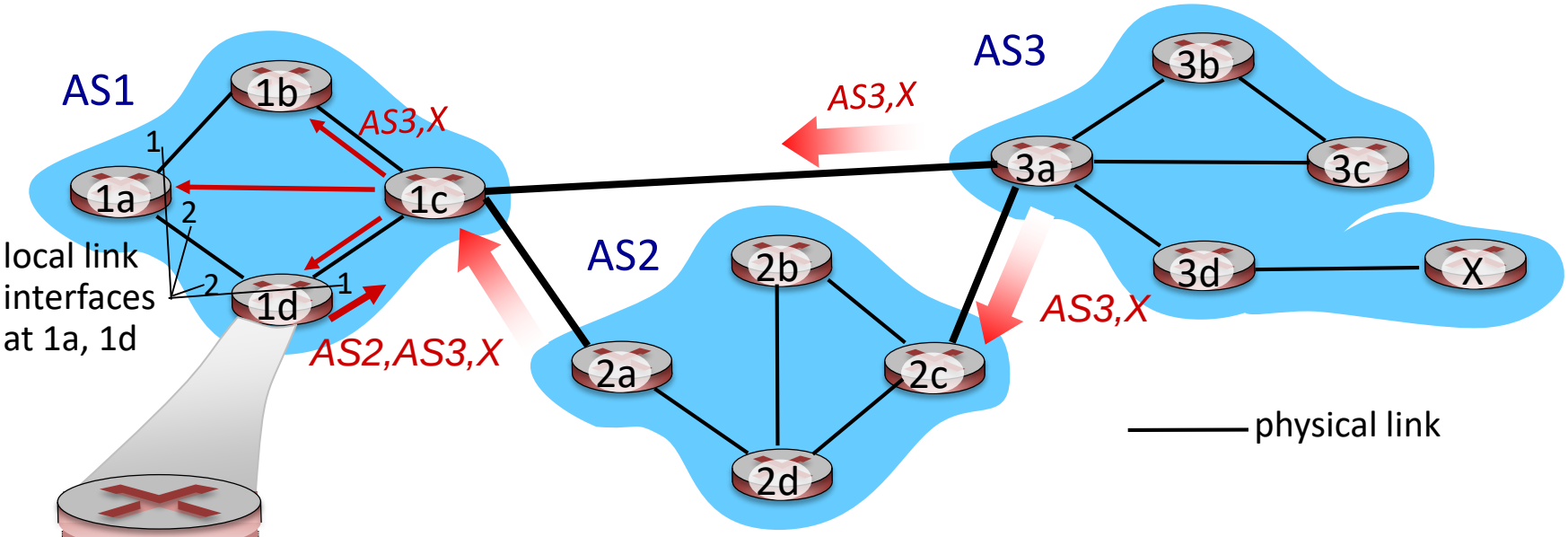
BGP messages:

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

BGP, OSPF, forwarding table entries

THE UNIVERSITY OF AUCKLAND
Te Whare Wananga o Tamaki Makaurau
NEW ZEALAND
ENGINEERING

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



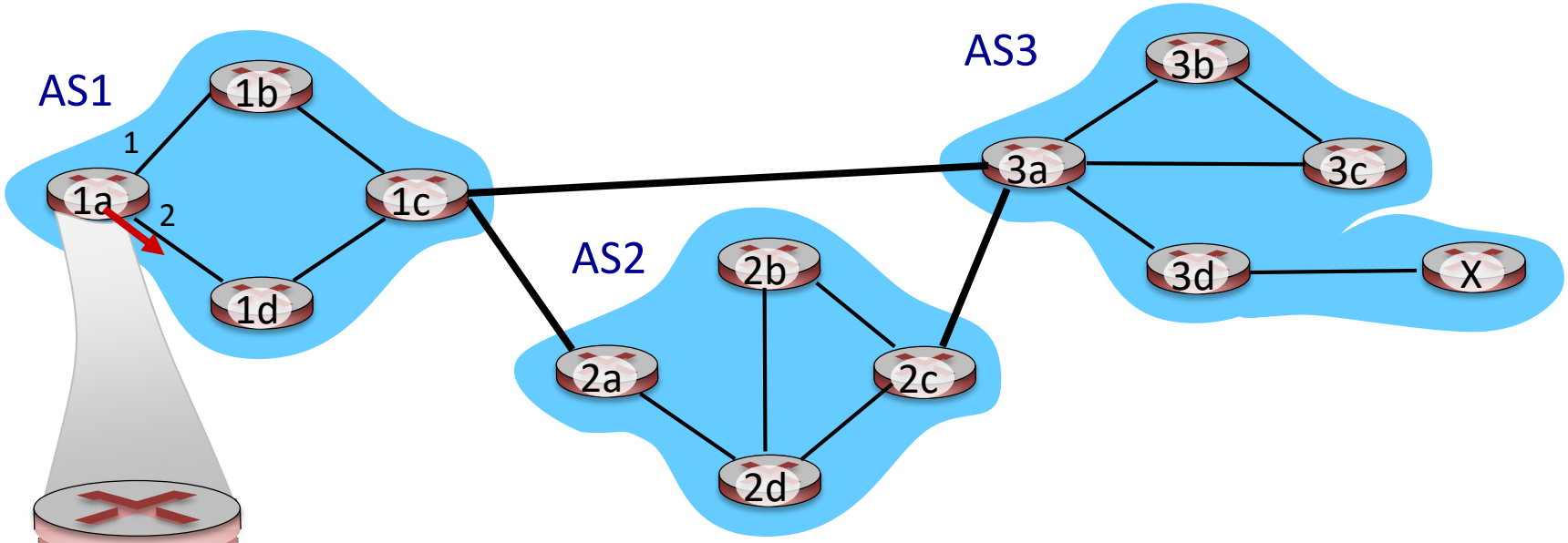
dest	interface
…	…
X	1
…	…

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

BGP, OSPF, forwarding table entries

THE UNIVERSITY OF AUCKLAND
Te Whare Wananga o Tamaki Makaurau
NEW ZEALAND
ENGINEERING

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



dest	interface
…	…
X	2
…	…

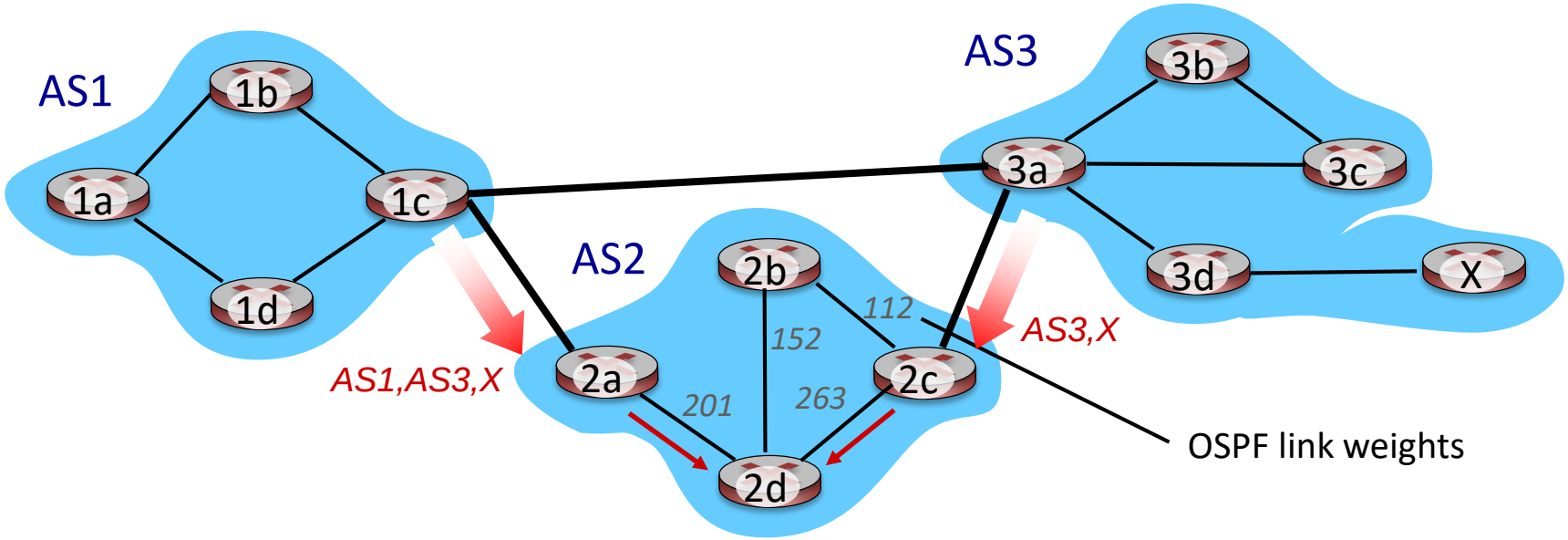
- recall: 1a, 1b, 1c learn about dest X via iBGP from 1c: “path to X goes through 1c”
- 1d: OSPF intra-domain routing: to get to 1c, forward over outgoing local interface 1

BGP route selection

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

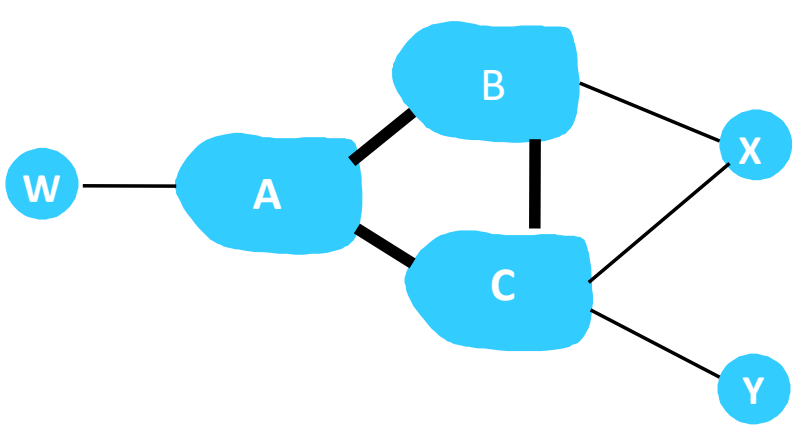
- Local preference value attribute: policy decision
- Shortest AS-PATH
- Closest NEXT-HOP router: hot potato routing
- Additional criteria

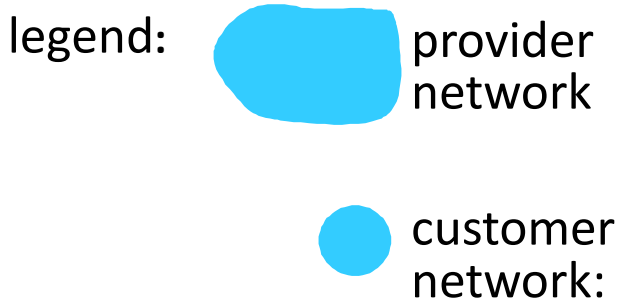
Hot Potato Routing



- 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)

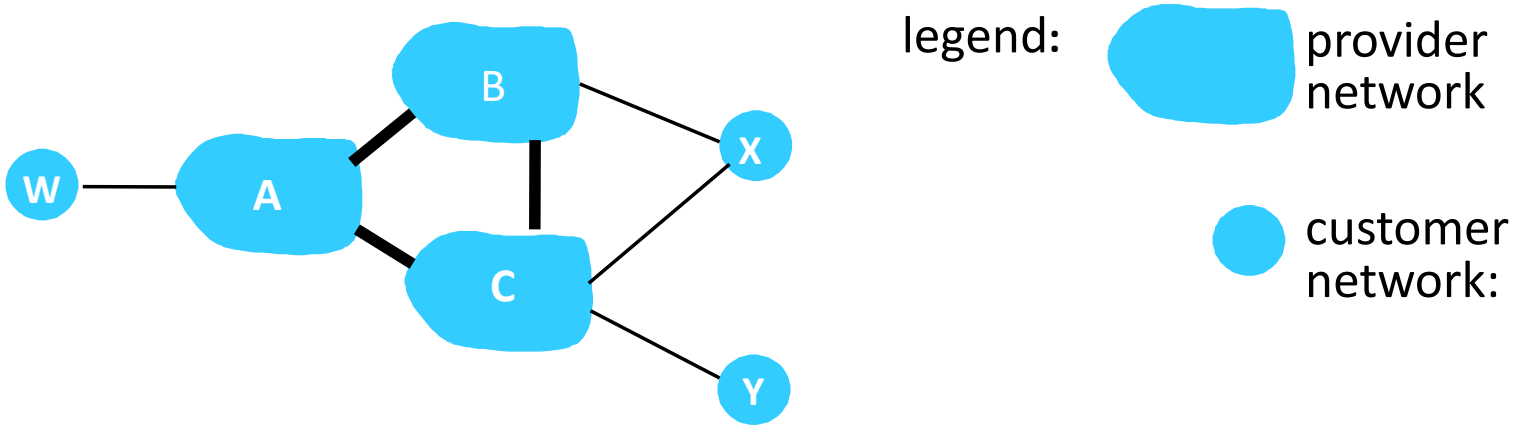
A advertises path Aw to B and to C

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

Why different Intra-, Inter-AS routing ?

Policy:

- inter-AS: admin wants control over how its traffic routed, who routes through its net.
- intra-AS: single admin, so no policy decisions needed

Scale:

- Hierarchical routing saves table size, reduced update traffic

Performance:

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

Software Defined Networking

THE UNIVERSITY OF AUCKLAND
Te Whare Wananga o Tamaki Makaurau
NEW ZEALAND
ENGINEERING

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

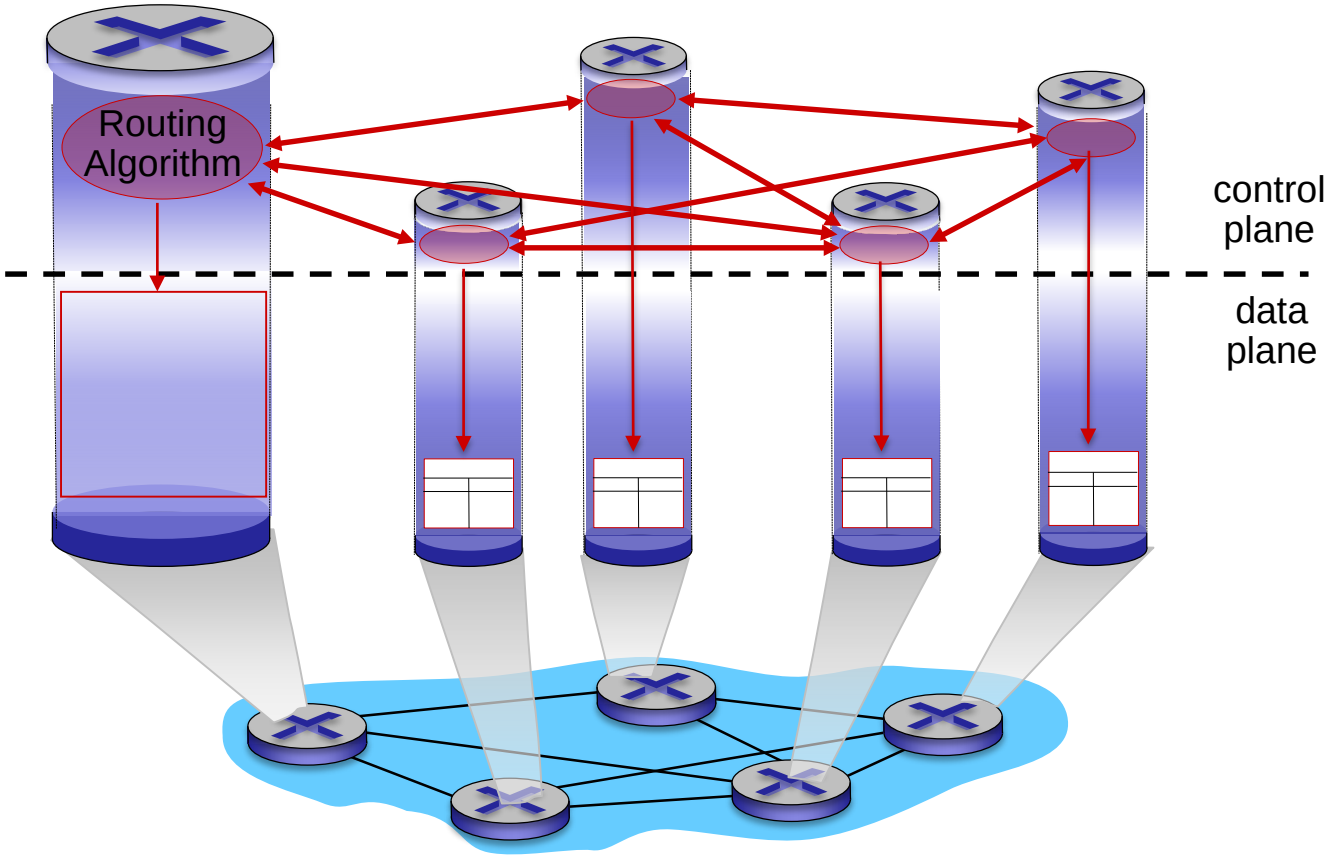
- *Monolithic* router contains switching hardware, runs proprietary implementation of Internet standard protocols (IP, RIP, IS-IS, OSPF, BGP) in proprietary router OS (e.g., Cisco IOS)
- Different "middleboxes" for different network layer functions: firewalls, load balancers, NAT boxes, ..

Interest in rethinking network control plane

- 2004: IETF standard *Forwarding and Control Element Separation* (ForCES)
- 2006: *Ethene* project at Stanford
- 2009: *OpenFlow* protocol version 1.0 released
- 2014: *OpenFlow* protocol version 1.5 released (current version is 1.5.1)

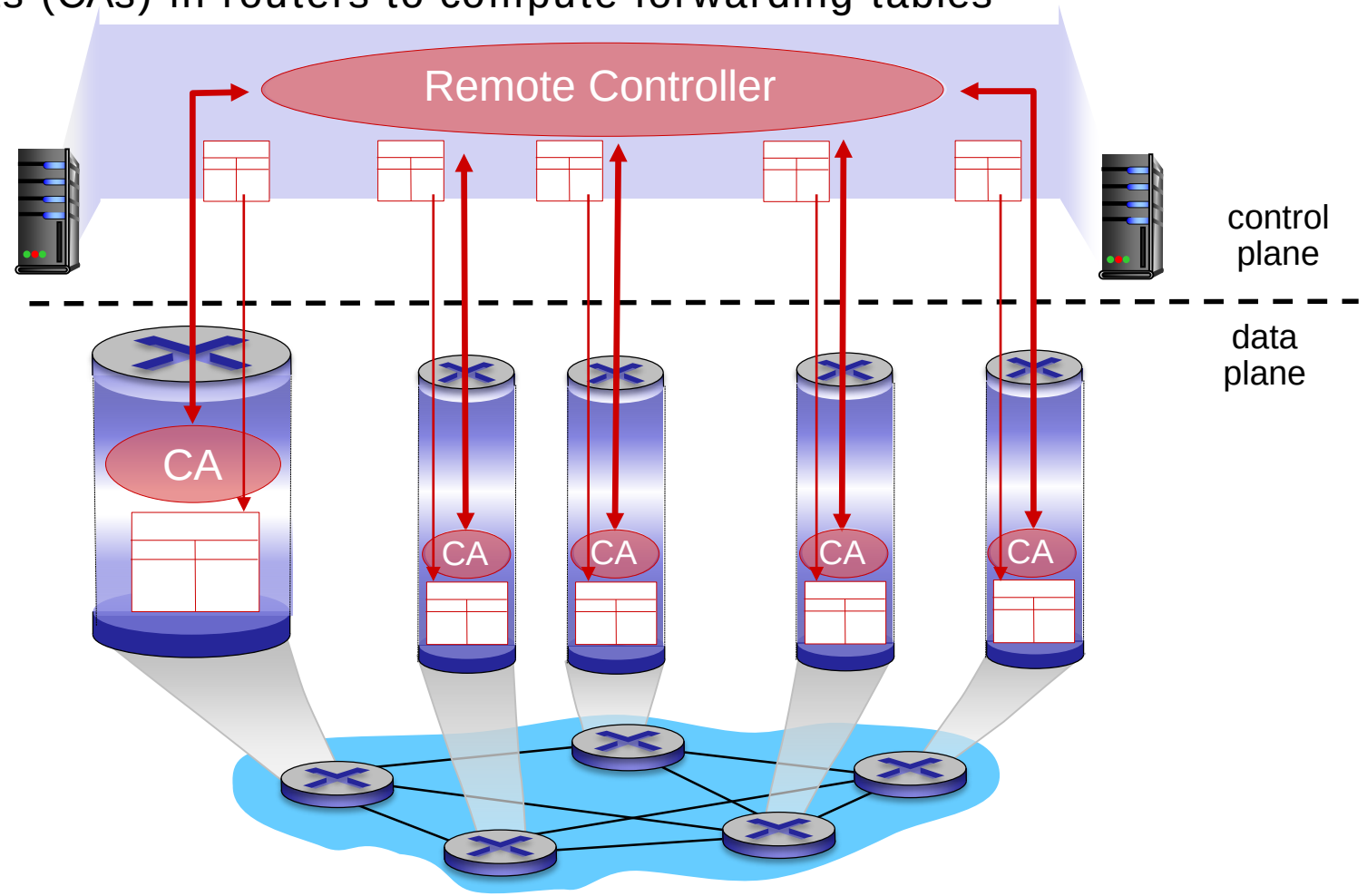
Per-router control plane

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



Software defined networking

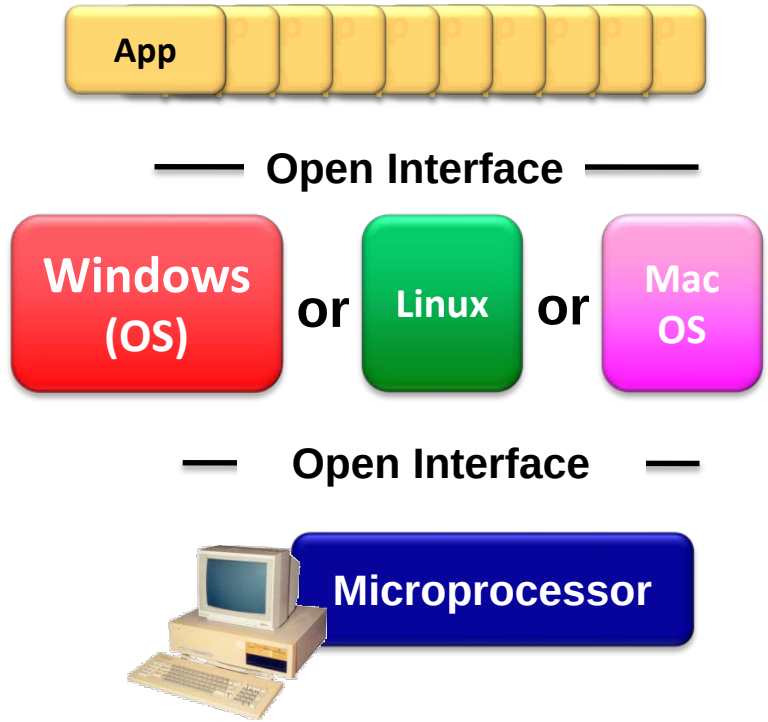
THE UNIVERSITY OF AUCKLAND
Te Whare Wananga o Tamaki Makaurau
NEW ZEALAND
ENGINEERING

Why a *logically centralized* control plane?

- Easier network management: avoid router misconfigurations, greater flexibility of traffic flows
- Table-based forwarding 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

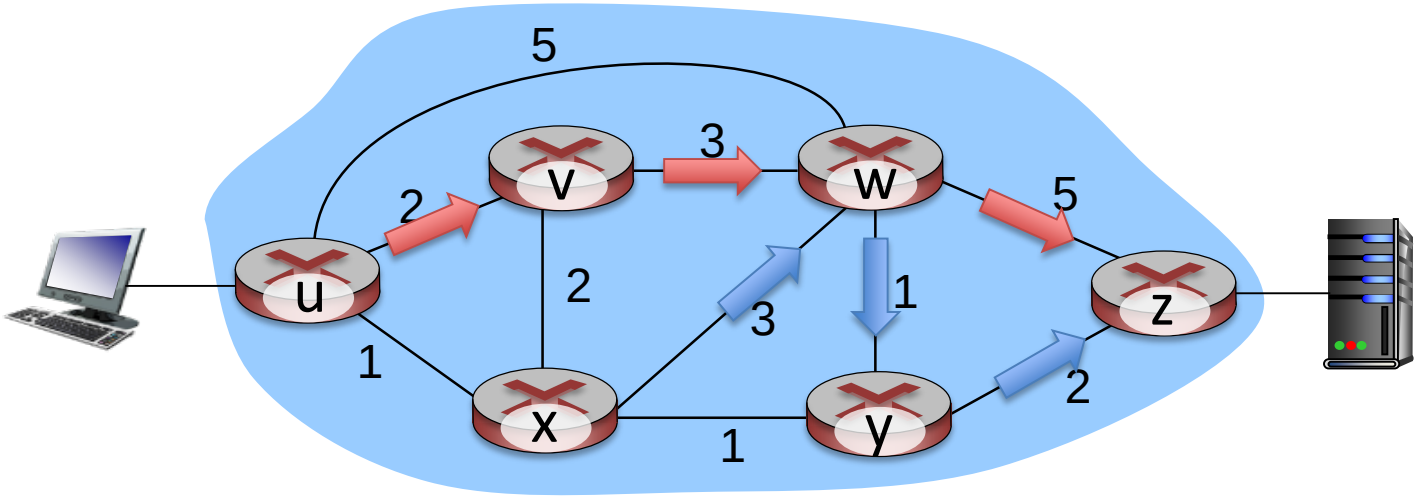
Analogy: mainframe to PC evolution

Vertically integrated
Closed, proprietary
Slow innovation
Small industry



Horizontal
Open interfaces
Rapid innovation
Huge industry

Traffic engineering: difficult routing

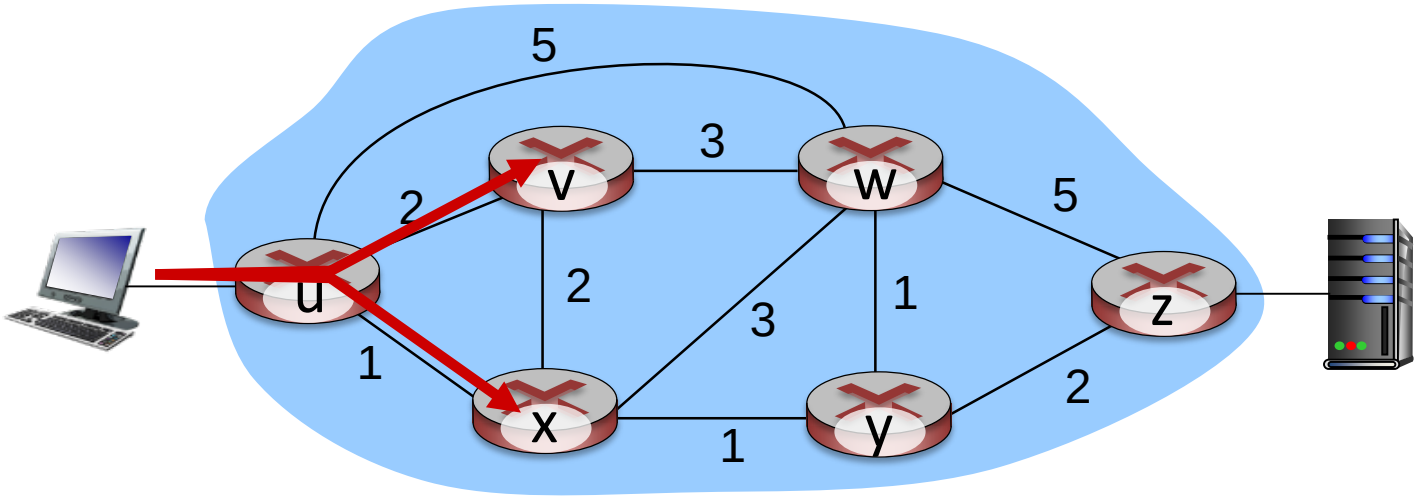


Q: what if network operator wants u-to-z traffic to flow along *uvwz*, x-to-z traffic to flow *xwyz*?

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!

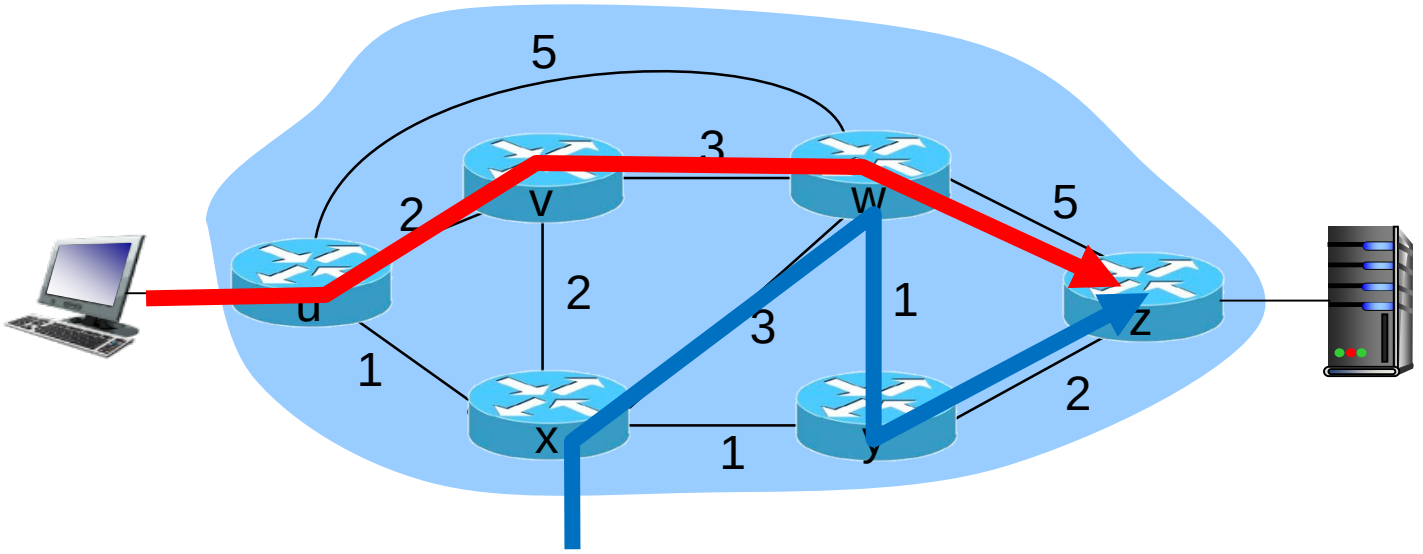
Traffic engineering: difficult task



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

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

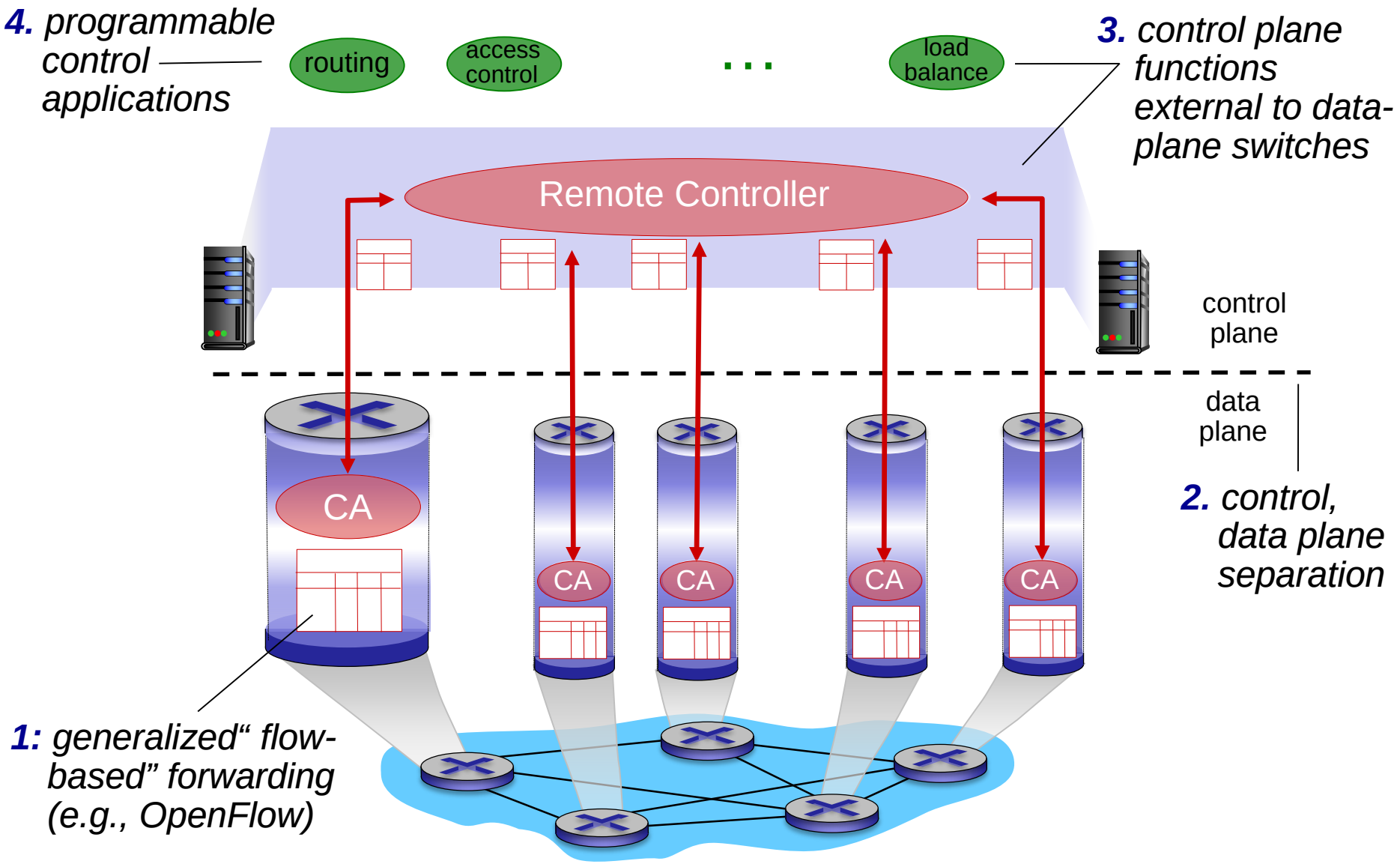
Traffic engineering: difficult task



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)

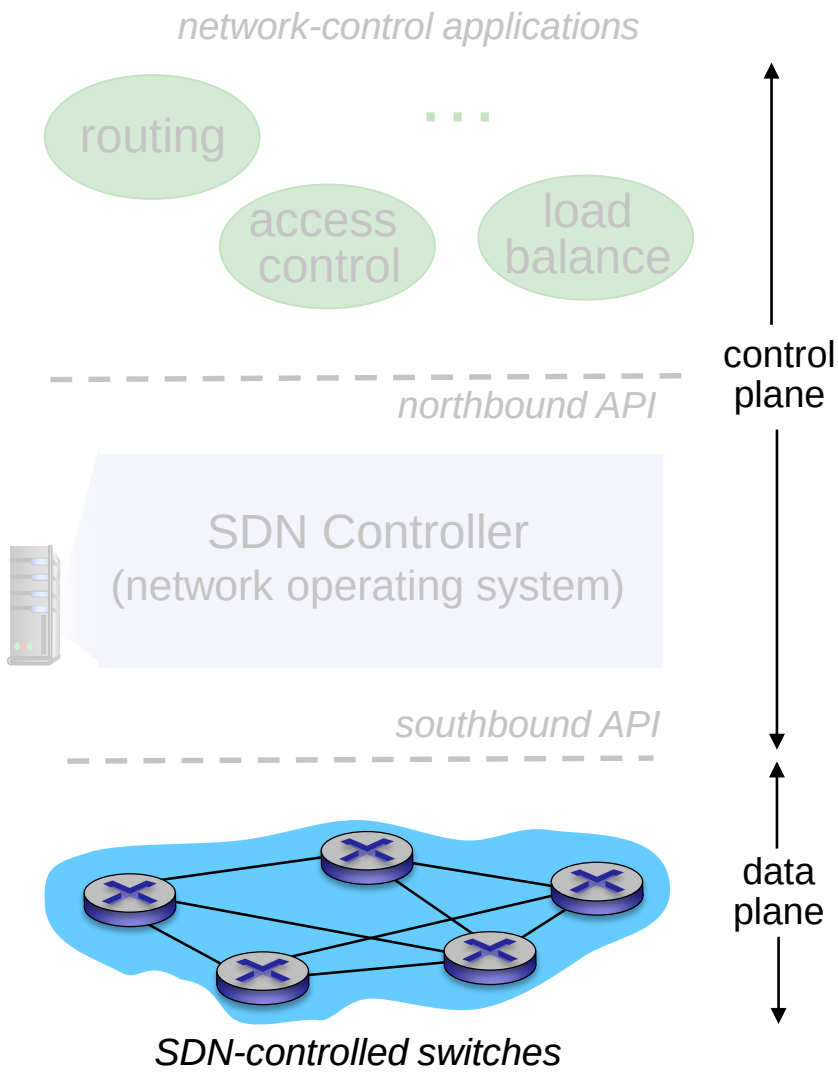
SDN Implementation



SDN perspective

Data plane switches

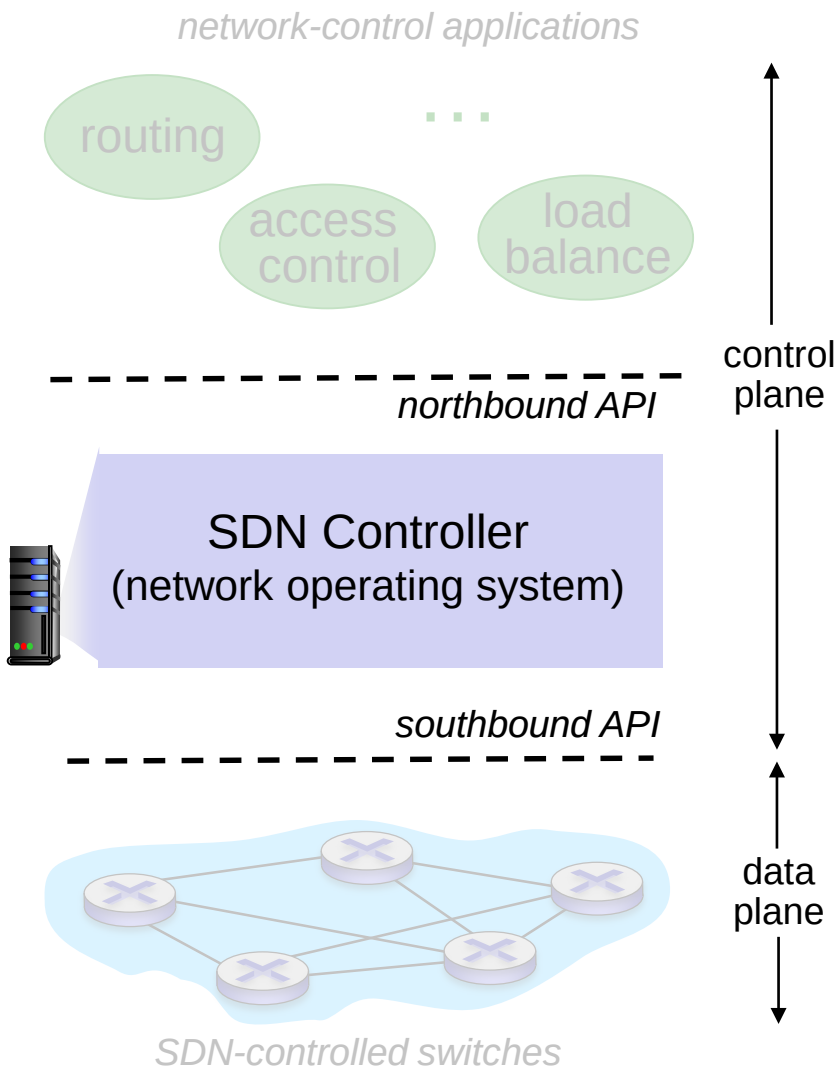
- Fast, simple, commodity switches implementing generalized data-plane forwarding (Section 4.4) in hardware
- Switch flow table computed, installed by controller
- API for table-based switch control (e.g., OpenFlow)
    - Defines what is controllable and what is not
- Protocol for communicating with controller (e.g., OpenFlow)



SDN perspective

SDN controller (network OS):

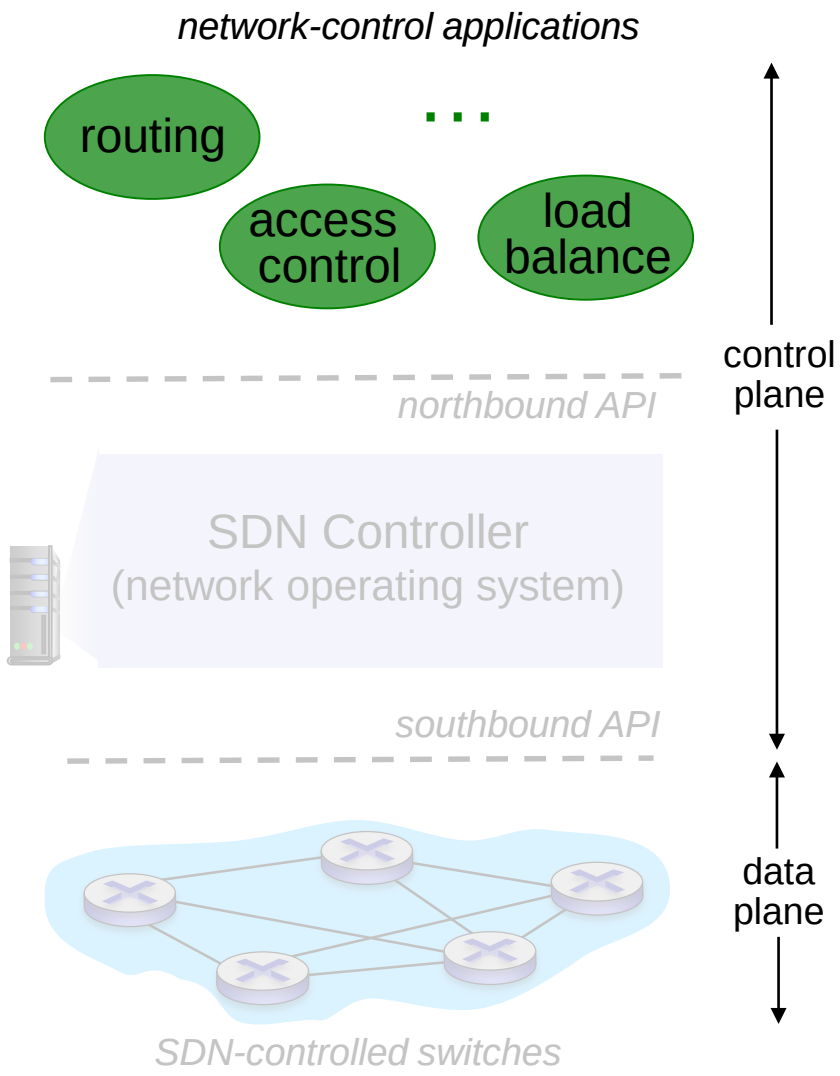
- Maintain network state information
- Interacts with network control applications “above” via northbound API
- Interacts with network switches “below” via southbound API
- Implemented as distributed system for performance, scalability, fault-tolerance, robustness



SDN perspective

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

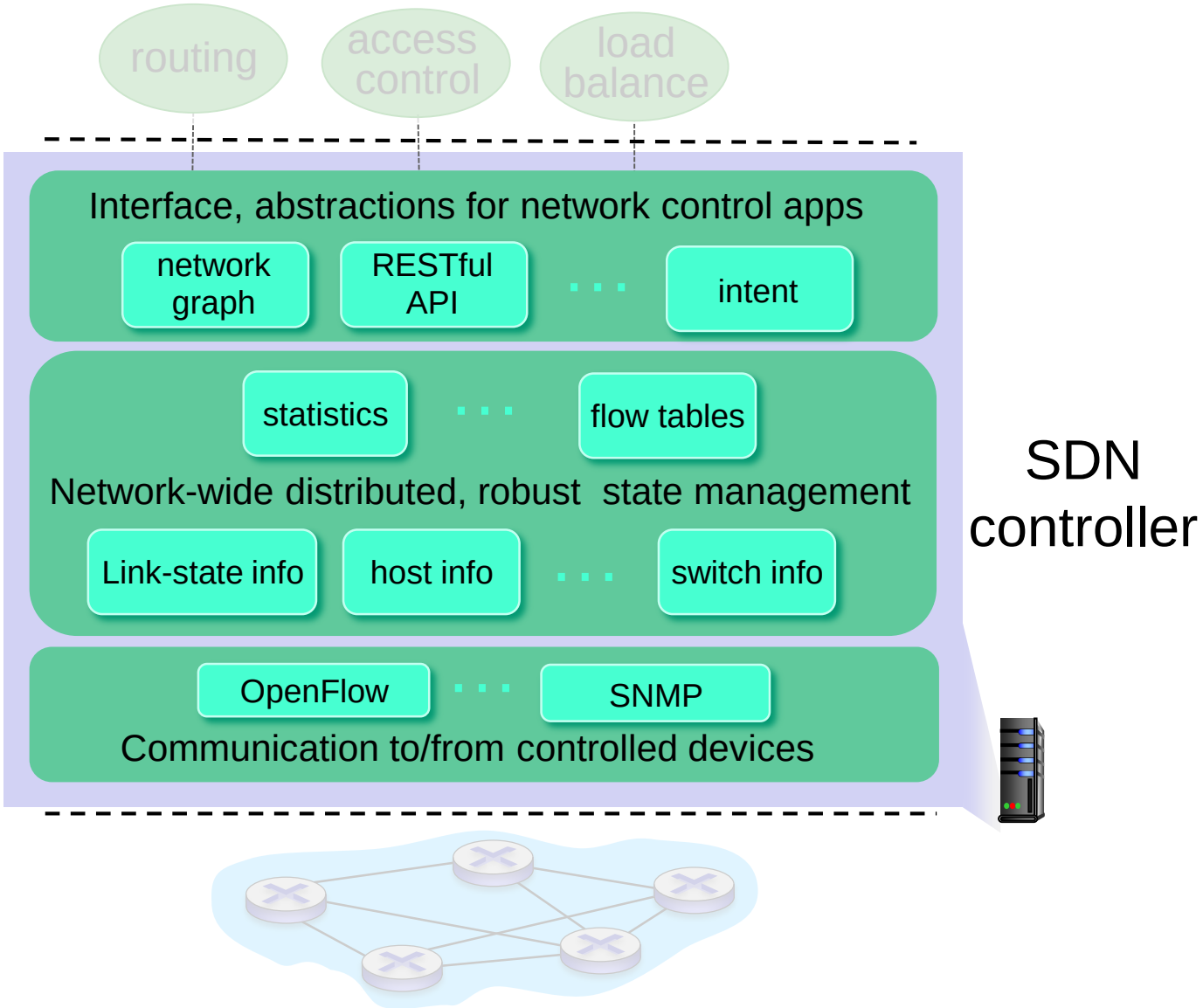


SDN controller

Interface layer to network control apps: abstractions API

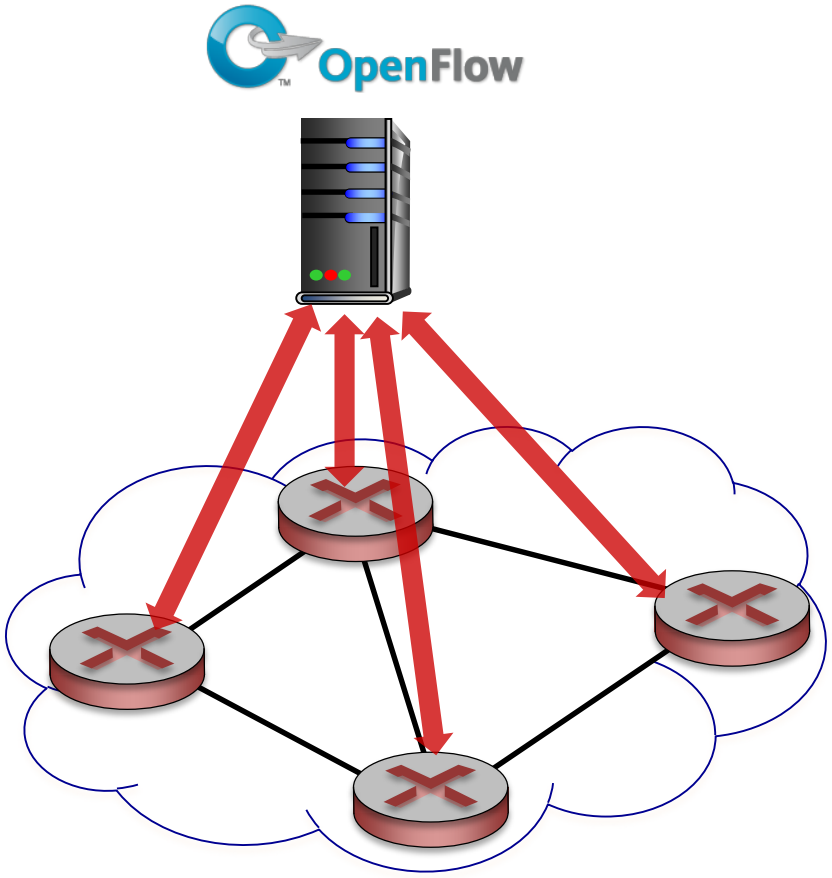
Network-wide state management layer: state of networks links, switches, services: a *distributed database*

communication layer: communicate between SDN controller and controlled switches



OpenFlow protocol

OpenFlow Controller



Operates between controller, switch

TCP used to exchange messages

- Optional encryption

Three classes of OpenFlow messages:

- Controller-to-switch
- Asynchronous (switch to controller)
- Symmetric (misc)

OpenFlow 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

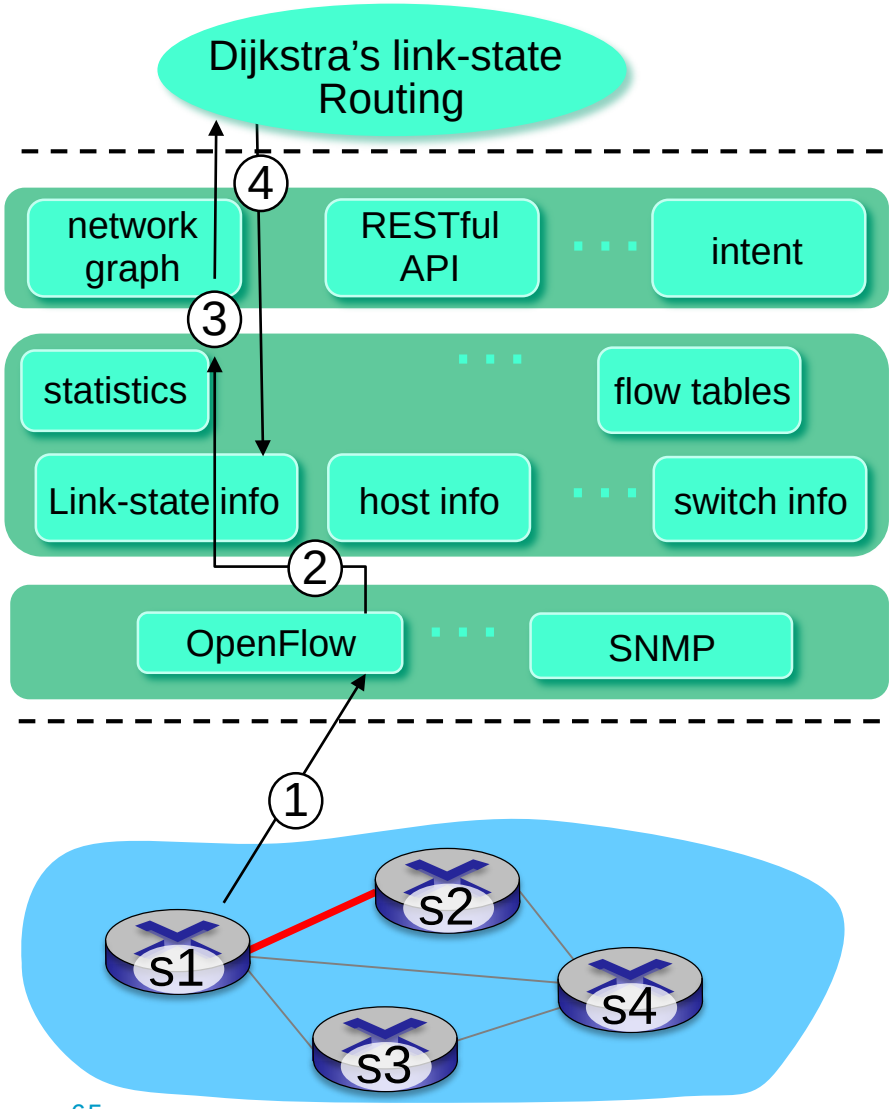
OpenFlow messages

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

Interaction Example



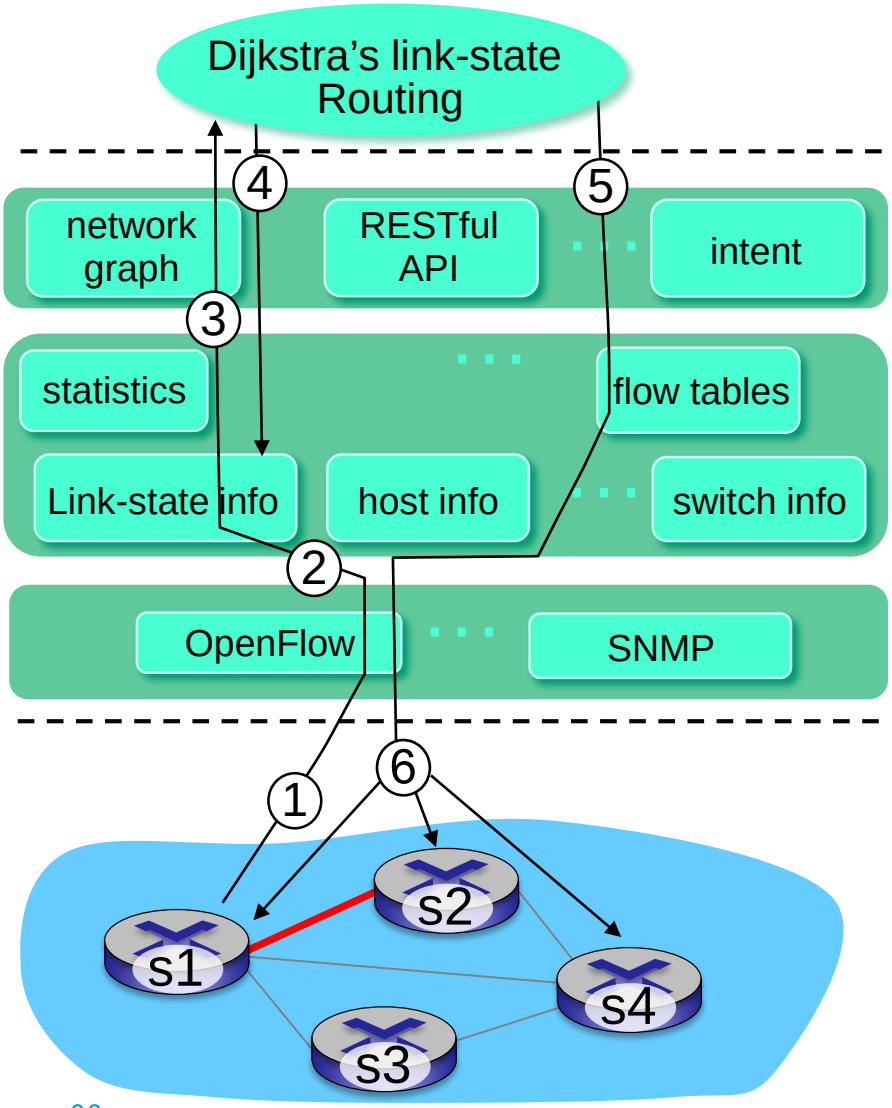
① S1, experiencing link failure uses OpenFlow port status message to notify controller.

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 accesses network graph info, link state info in controller, computes new routes.

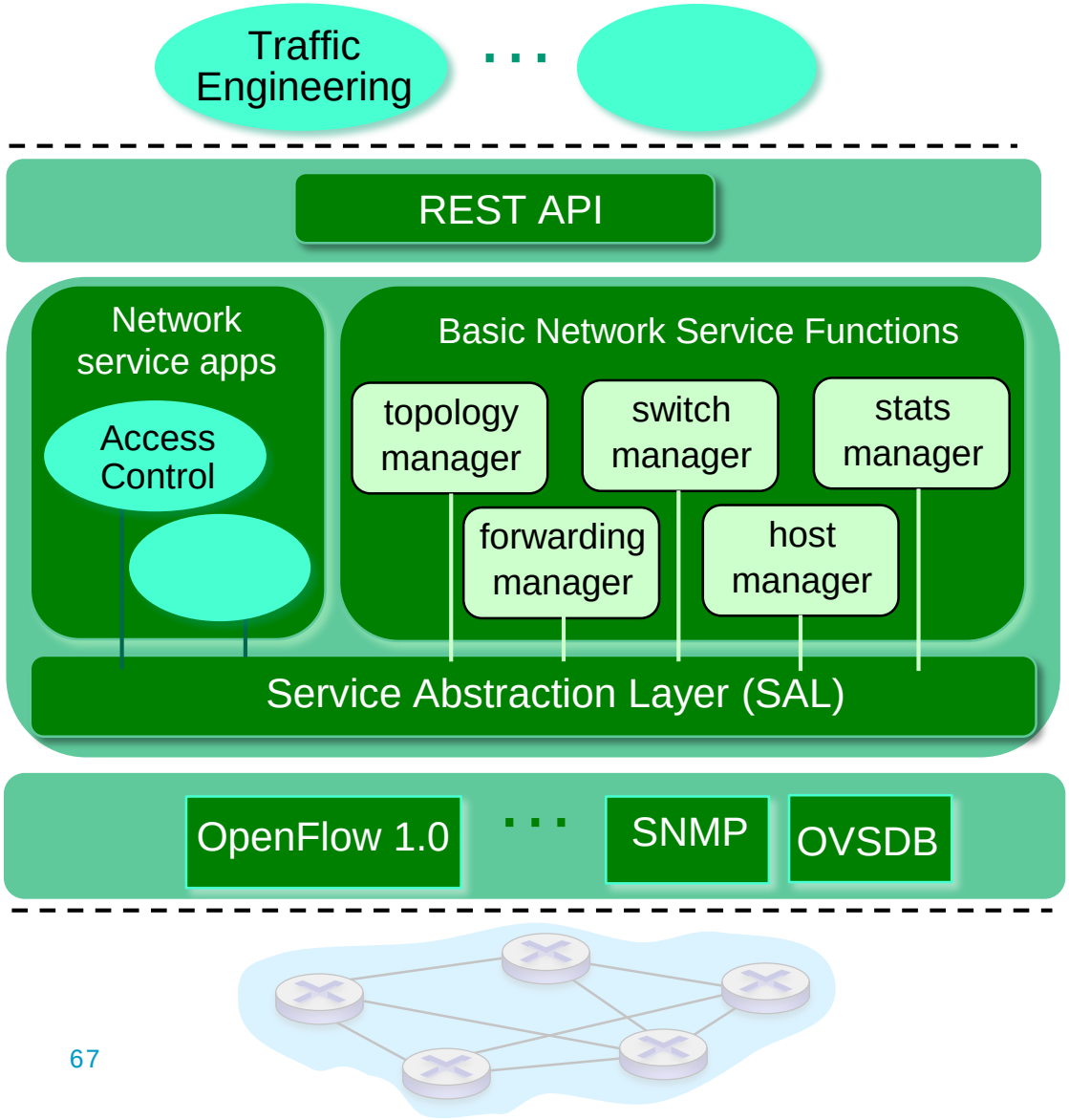
Interaction Example



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

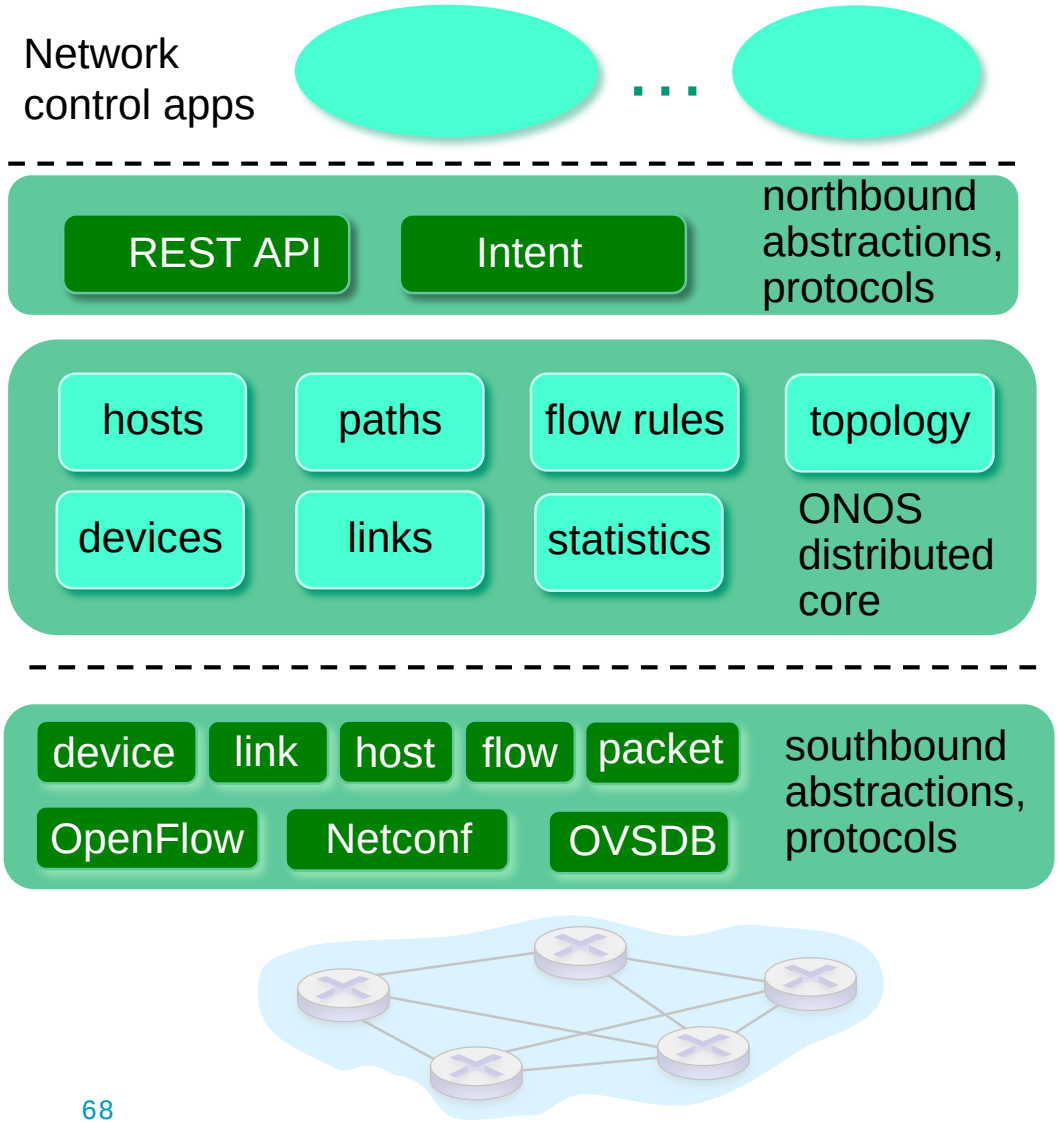
⑥ Controller uses OpenFlow to install new tables in switches that need updating.

OpenDaylight (ODL)



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

ONOS controller



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

SDN: selected challenges

Hardening the control plane: dependable, reliable, performance-scalable, secure distributed system

- Robustness to failures: leverage strong theory of reliable distributed system for control plane
- Dependability, security: “baked in” from day one?

Networks, protocols meeting mission-specific requirements

- e.g., real-time, ultra-reliable, ultra-secure

Internet-scaling

SDN in Practice

SDN at Google

- Started building in 2006!

Four pillars:

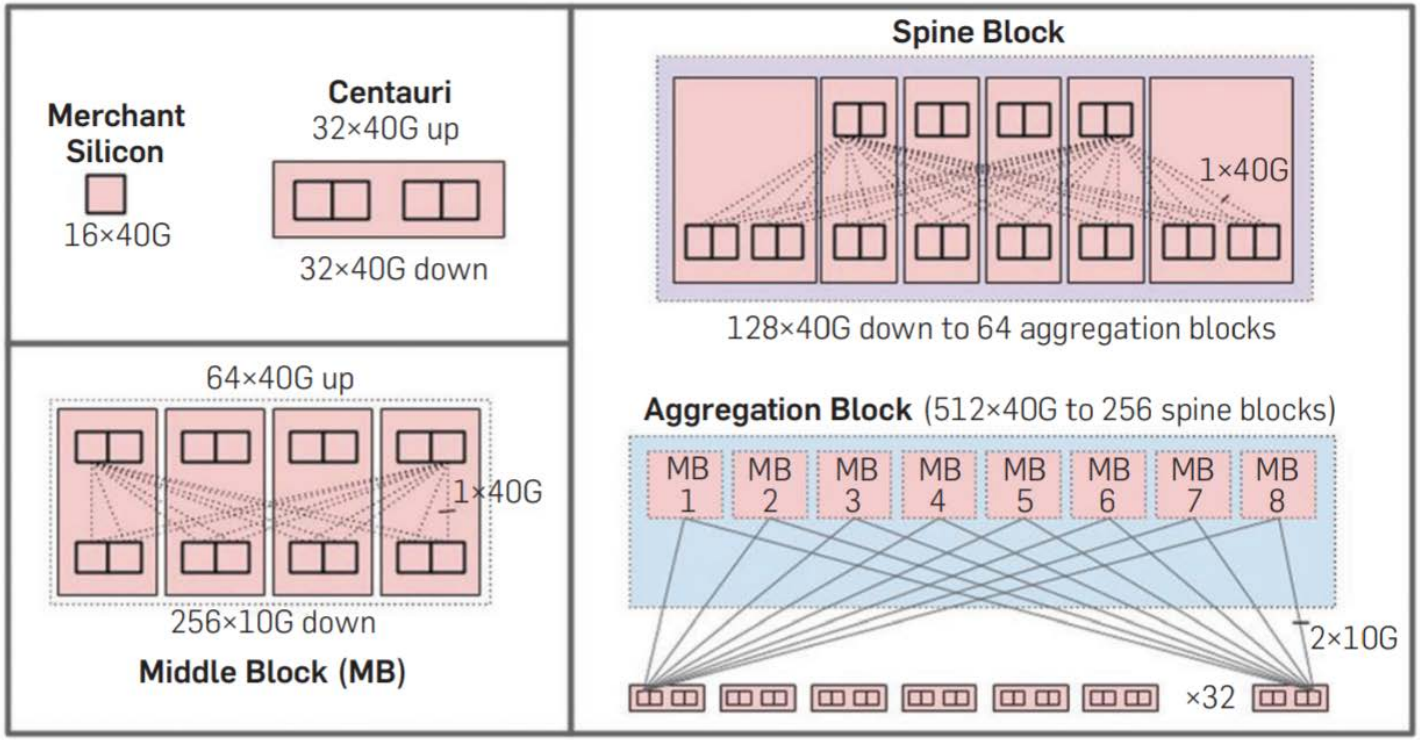
1. *Jupiter*: data centre interconnect
2. *B4*: connects data centres together (WAN)
3. *Andromeda*: network functions virtualization
4. *Expresso*: connects to public internet

SDN in Practise: Jupiter

Centralised control mechanism for a data centre

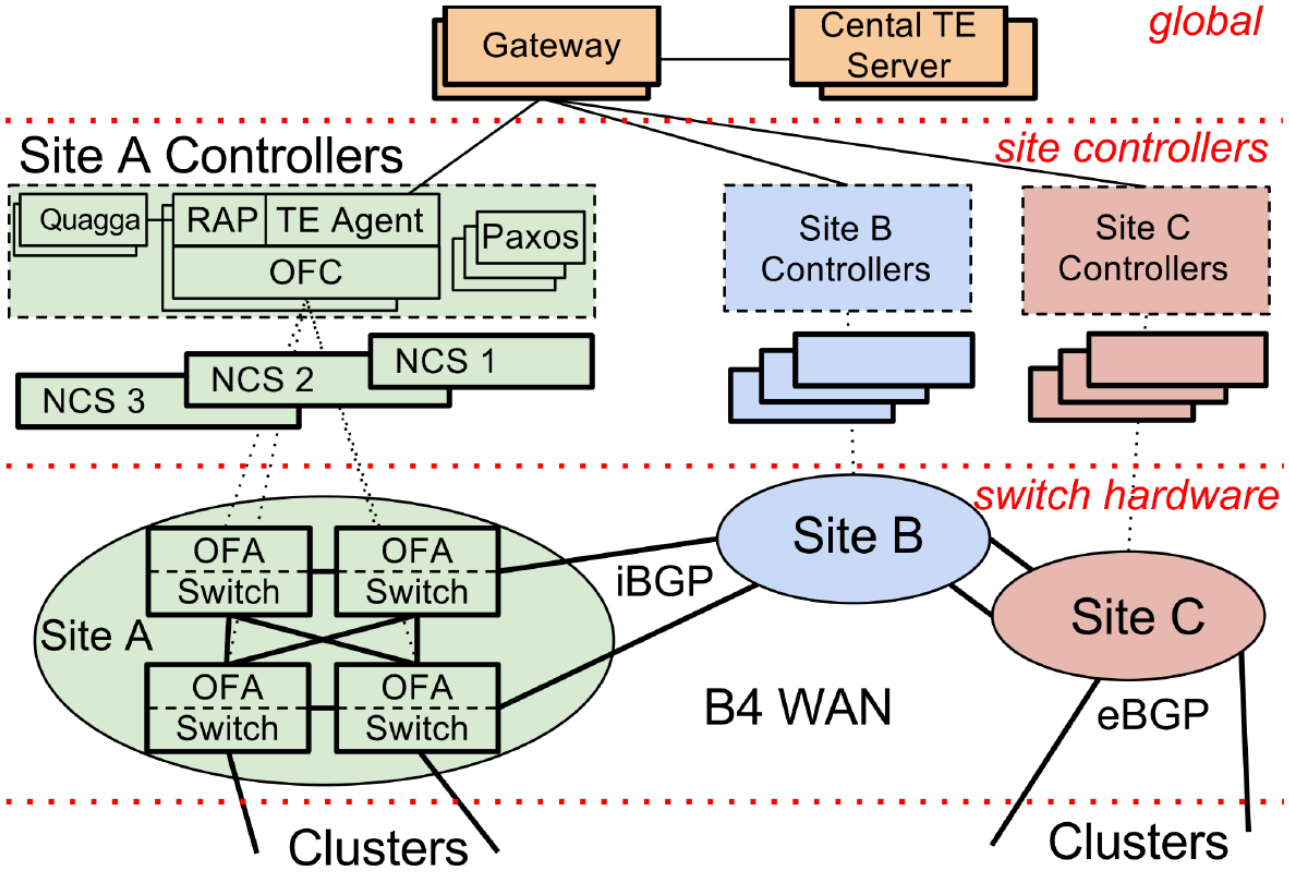
Controls all switches and routers within each data centre

- Allows more than 1 Pbps of bandwidth (125,000,000,000 KBps!)



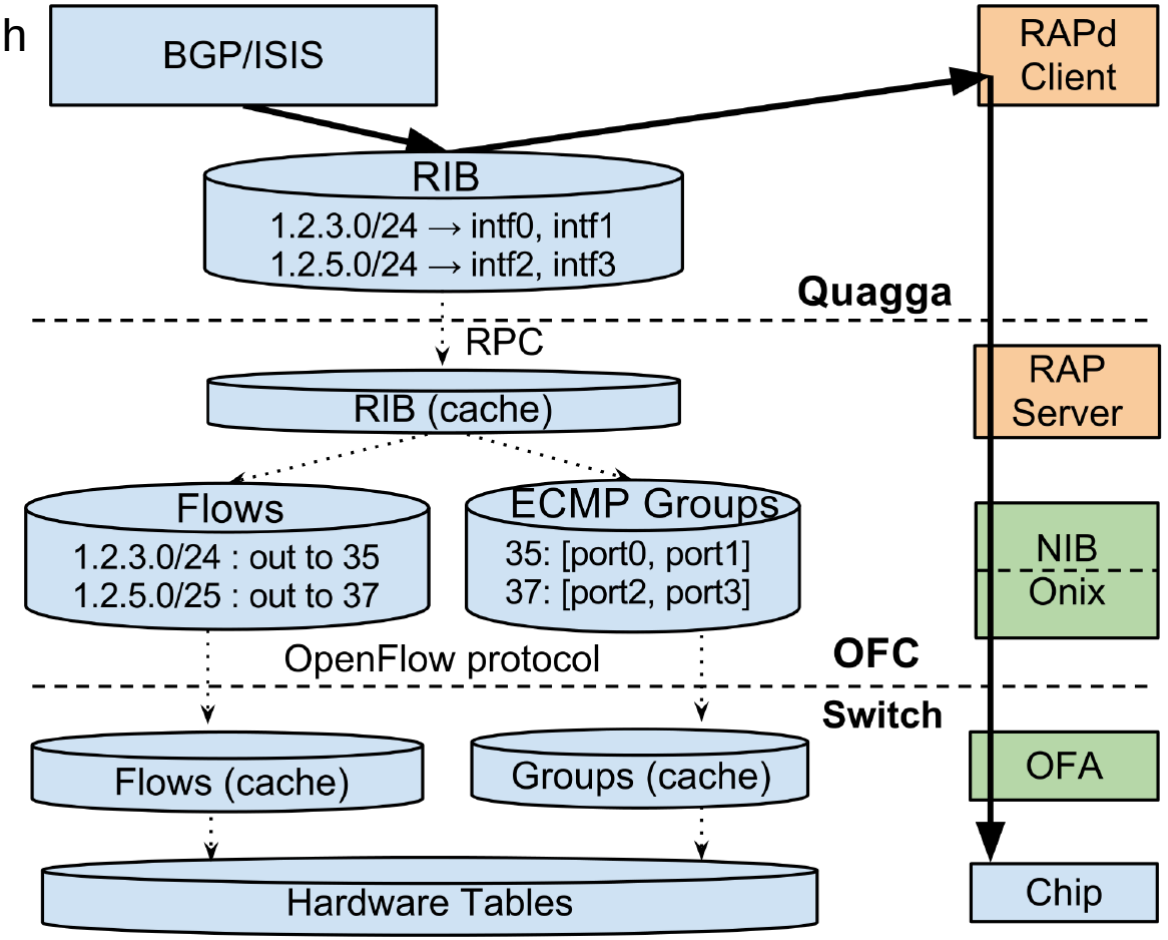
SDN in Practice: B4

Private WAN connecting Data Centres



SDN in Practice: B4

Integrated routing with OpenFlow



SDN in Practice: Andromeda

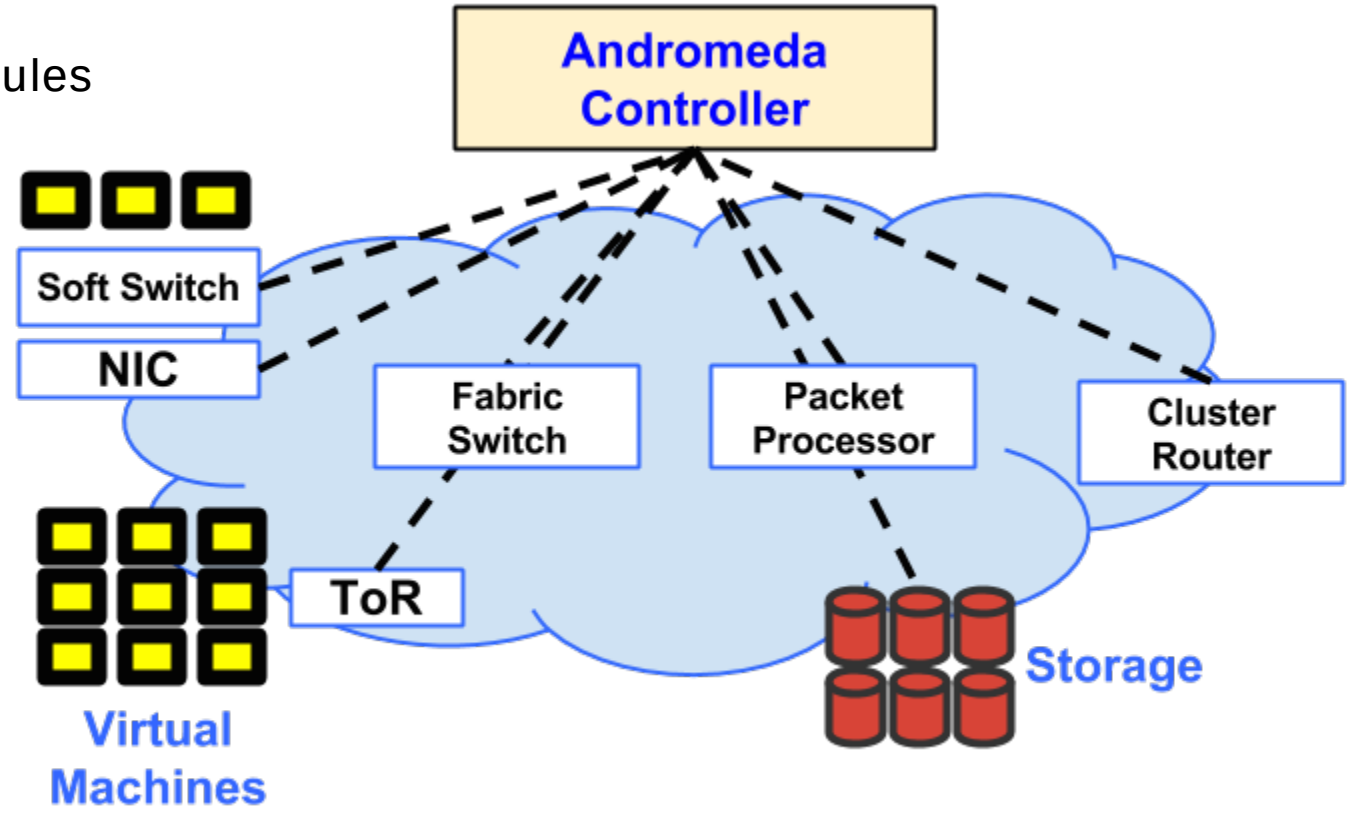
THE UNIVERSITY OF AUCKLAND
Te Whare Wananga o Tamaki Makaurau
NEW ZEALAND
ENGINEERING

Network virtualization software

Exposes raw performance of network (SDN-based)

Contains:

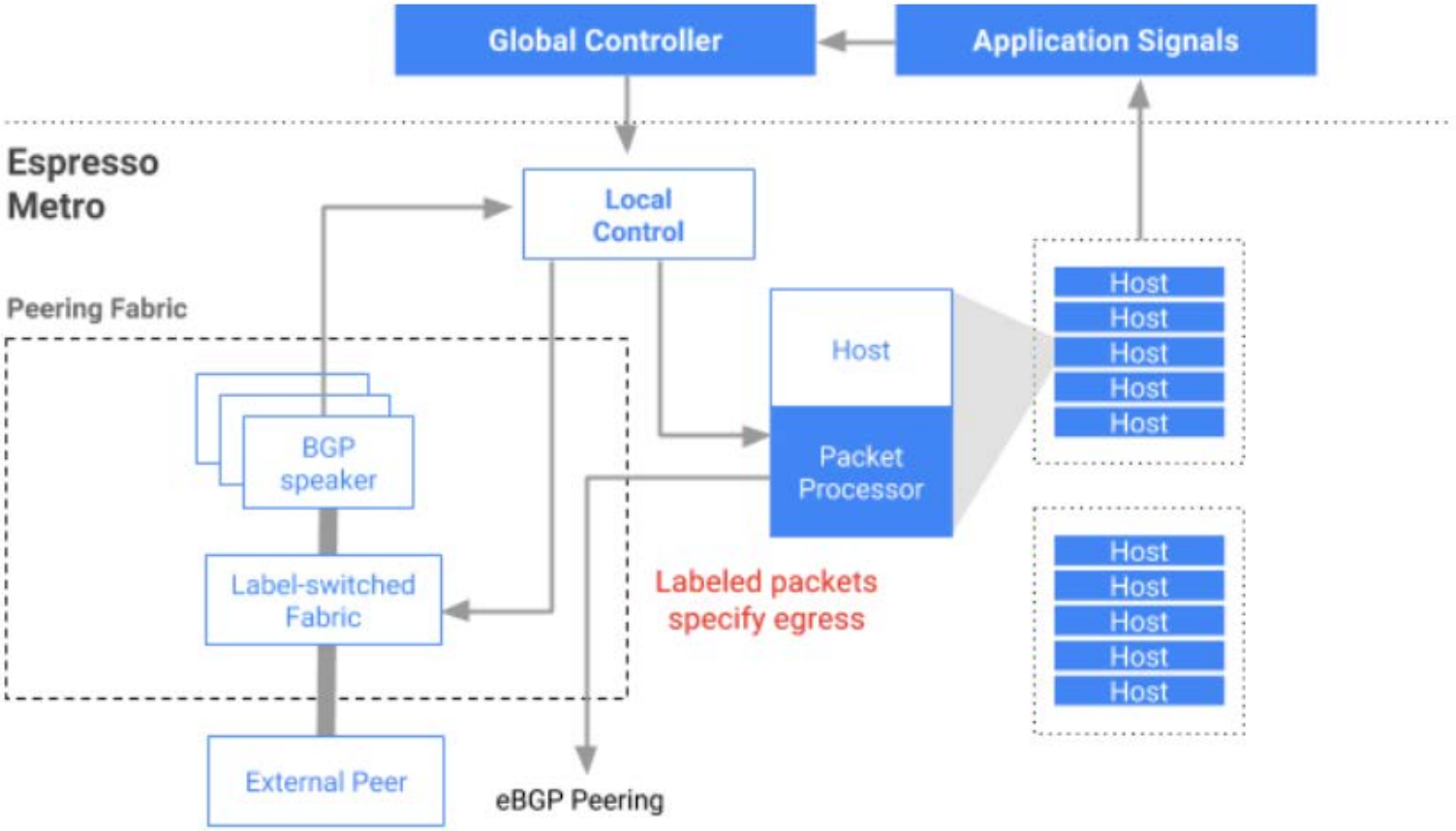
- Forwarding rules
- Firewalls
- Routing



SDN in Practice: Espresso

THE UNIVERSITY OF AUCKLAND
Te Whare Wananga o Tamaki Makaurau
NEW ZEALAND
ENGINEERING

Connection to public internet
Allows monitoring and control



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
- One of the main protocol in IP
    - Versions for IPv4 and IPv6

Internet Control Message Protocol

ICMP message: type, code, checksum plus ICMP data and original IP header

Identified as IP protocol 1

Can be used in denial-of-service attacks

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

Source sends series of UDP segments to destination

- First set has TTL =1
- Second set has TTL=2, etc.
- Unlikely port number

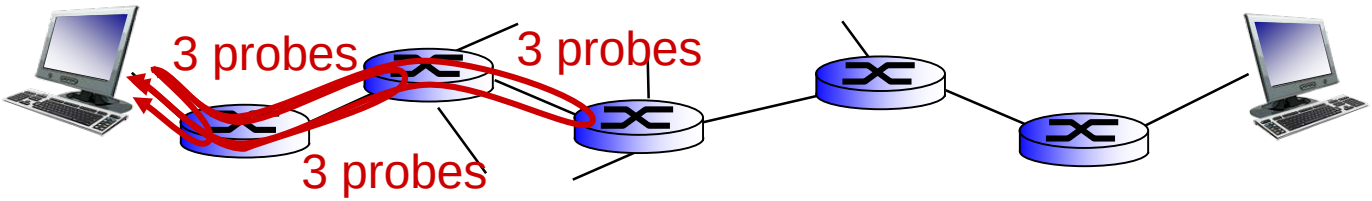
When datagram in nth set arrives to nth router:

- 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

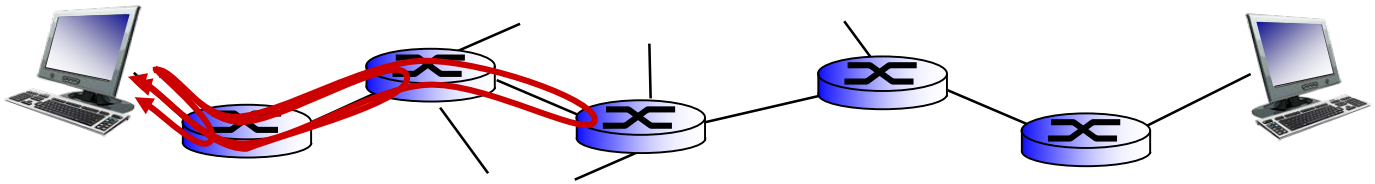


Traceroute

```
Tracing route to google.com [172.217.167.110]
over a maximum of 30 hops:

  1     1 ms     2 ms     1 ms  www.routerlogin.com [192.168.0.1]
  2     2 ms     3 ms     3 ms  lo-vprn-100.br1.mdr.vygr.net [114.23.3.255]
  3     6 ms     3 ms     3 ms  ae-0-621.cr1.mdr.vygr.net [114.23.3.230]
  4    29 ms    87 ms    63 ms  xe-0-1-7-cr1.sy4.vygr.net [114.23.11.250]
  5    29 ms    26 ms    27 ms  as15169.nsw.ix.asn.au [218.100.52.3]
  6    29 ms    29 ms    27 ms  108.170.247.65
  7    28 ms    27 ms    31 ms  209.85.253.181
  8    64 ms    27 ms    27 ms  syd09s17-in-f14.1e100.net [172.217.167.110]

Trace complete.
```

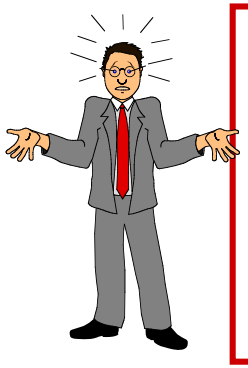


What is network management?

Autonomous systems (aka "*network*"): 1,000's of interacting hardware/software components

Other complex systems requiring monitoring, control:

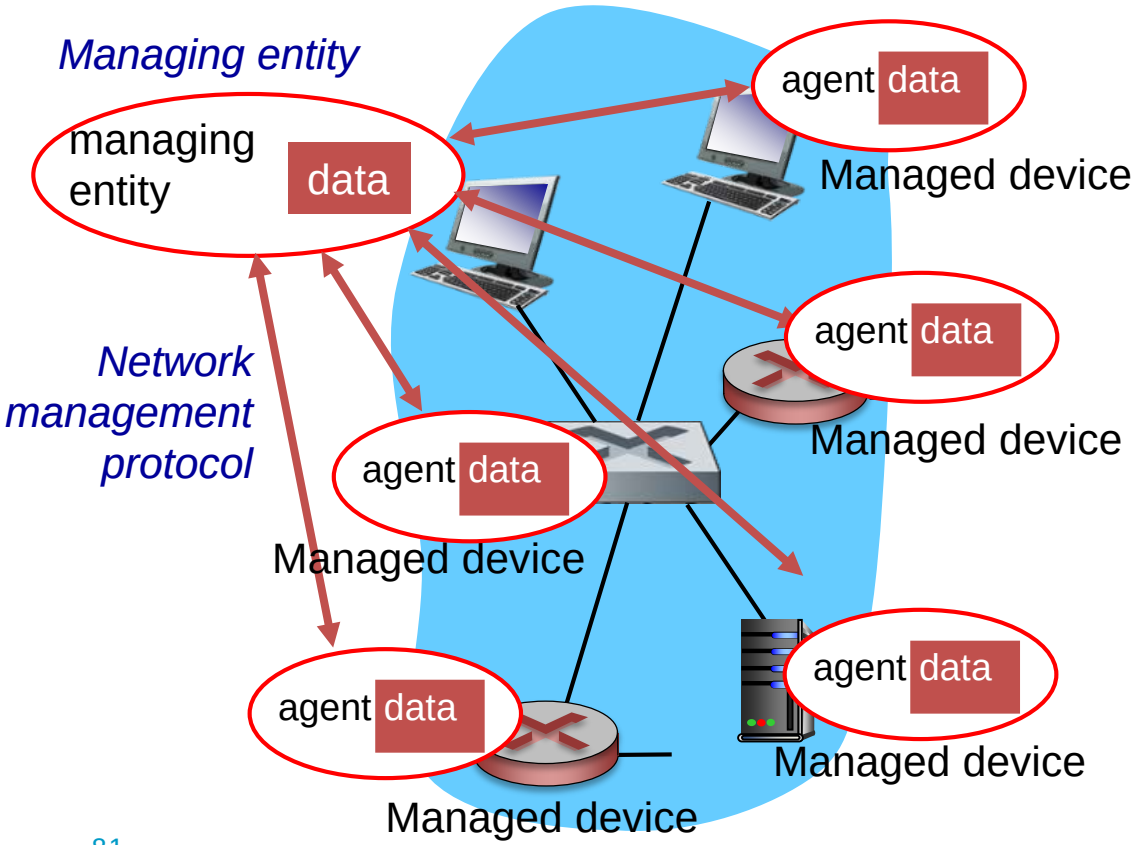
- Jet airplane
- Nuclear power plant
- Others?



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

Infrastructure for network management

Definitions:

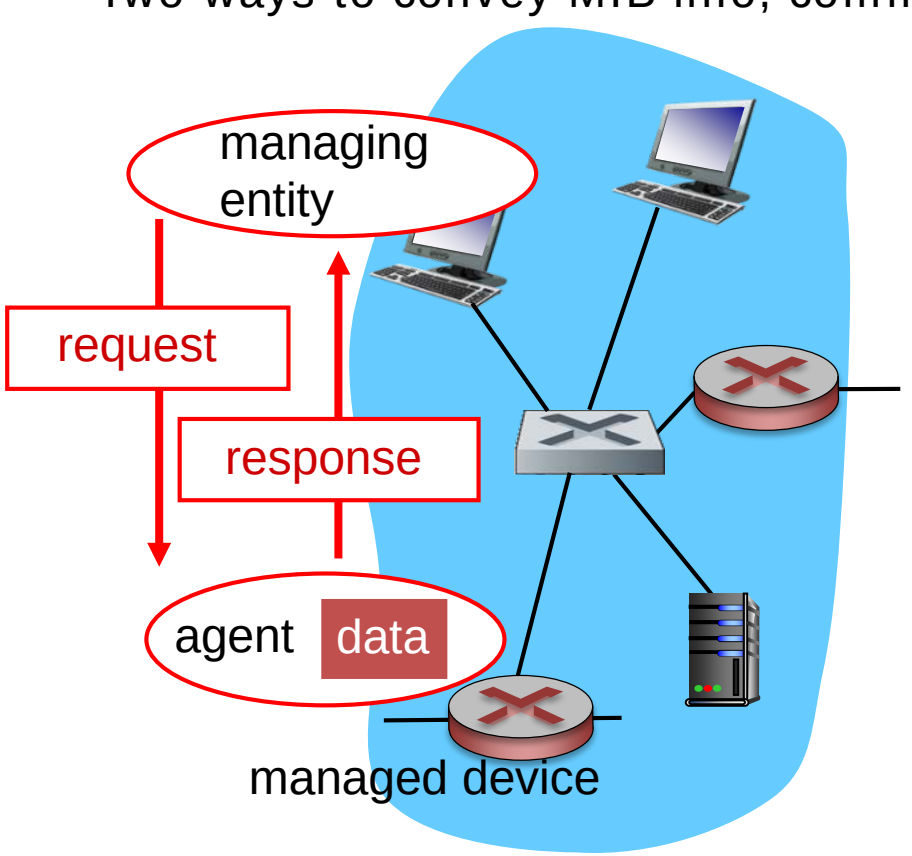


Managed devices contain *Managed objects* whose data is gathered into a *Management Information Base (MIB)*

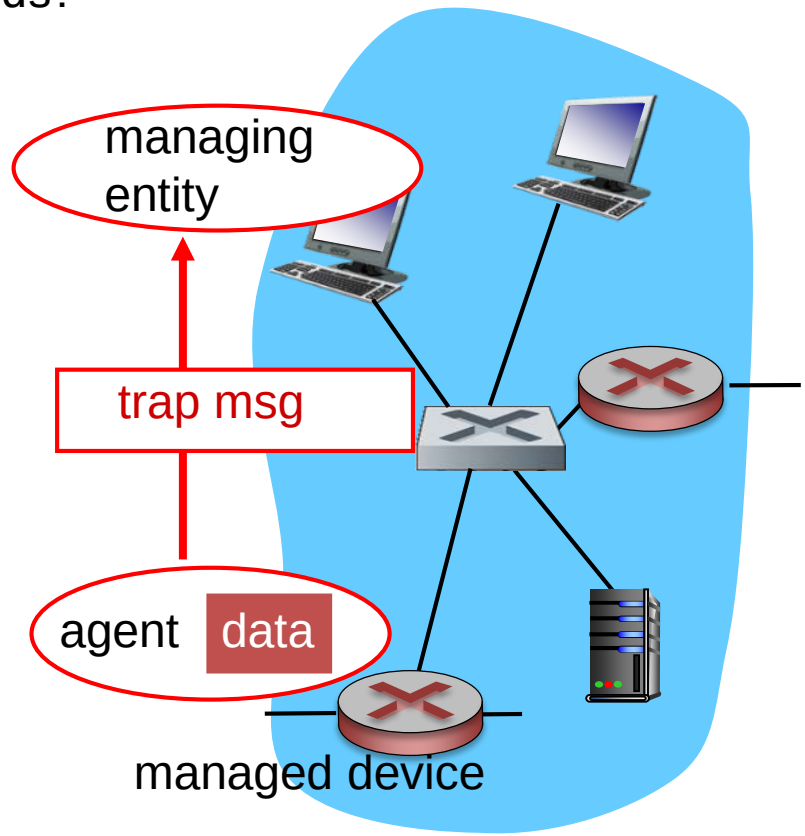
Simple Network Management Protocol

SNMP protocol

Two ways to convey MIB info, commands:



Request/Response mode

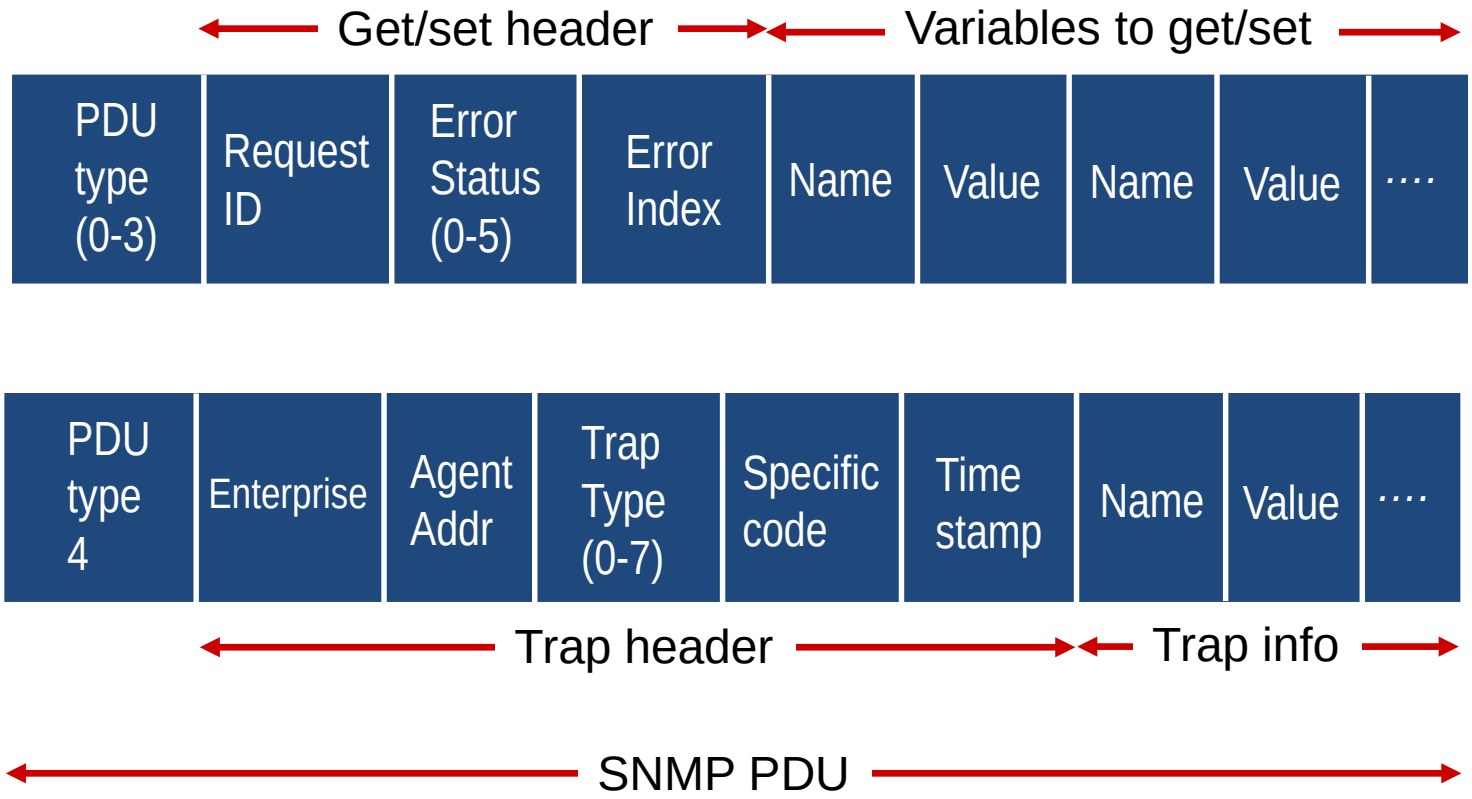


Trap mode

SNMP protocol: message types

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

SNMP protocol: message formats



Summary

Reviewed the control plane of the network layer

Identified two approaches for implementation (per-router and centralized control)

Introduced the terminology for graph abstraction

Described the link-state (Dijkstra's) algorithm

Described the distance vector algorithm

Compared the link-state and distance vector algorithms

Introduced OSPF and how it works

Introduced BGP and how it works

Described how OSPF and BGP work together and their differences

Software Defined Networking:

- Definition
- Challenges of per-router control
- SDN components and OpenFlow
- OpenDaylight and ONOS Controllers

Internet Control Message Protocol

Simple Network Management Protocol

ENGINEERING