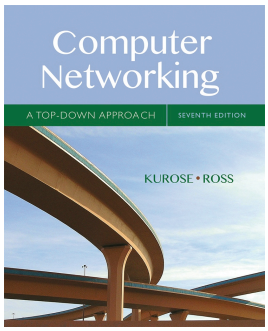
Chapter 5: Network Layer: The Control Plane

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



Computer Networking: A Top Down Approach
7th edition
Jim Kurose, Keith Ross
Pearson/Addison-Wesley
April 2016

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

Network-layer functions

Recall: two network-layer functions:

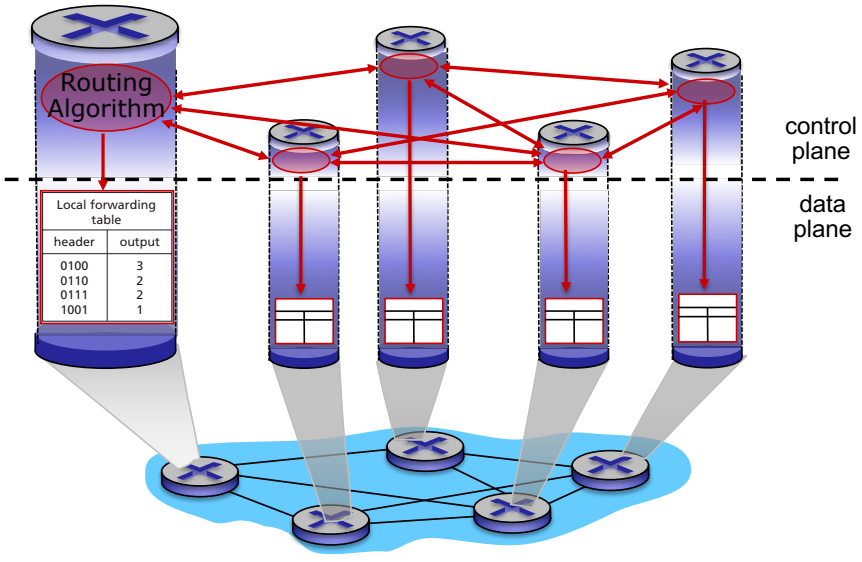
- *forwarding:* move packets from router's input to appropriate router output — *data plane*
- *routing:* determine route taken by packets from source to destination — *control plane*

Two approaches to structuring network control plane:

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

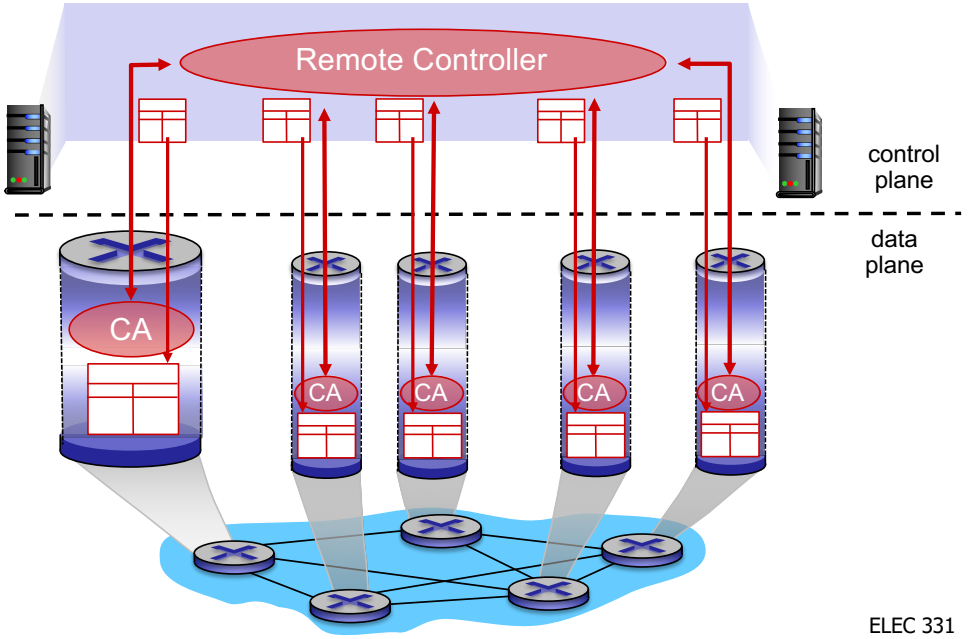
Per-router control plane

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

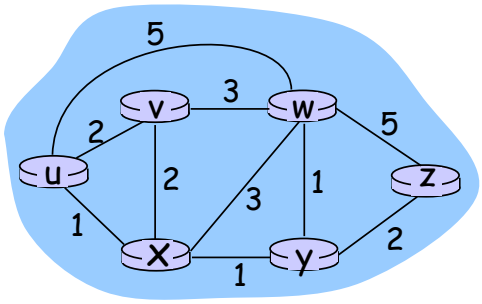
- 5.1 Introduction
- 5.2 Routing protocols
  - link state
  - distance vector
- 5.3 Intra-AS routing in the Internet: OSPF
- 5.4 Routing among the ISPs: BGP
- 5.5 The SDN control plane
- 5.6 ICMP: The Internet Control Message Protocol
- 5.7 Network management and SNMP

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 a network



Graph: $G = (N, E)$

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

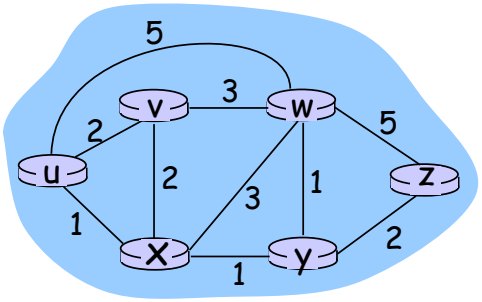
E = set of links ={ (u,v), (u,x), (u,w), (v,x), (v,w), (x,w), (x,y), (w,y), (w,z), (y,z) }

Neighbors of node u = {v, w, x}

Remark: Graph abstraction is useful in other network contexts

Example: 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 related to congestion

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

Question: What's the least-cost path between u and z ?

Routing algorithm: Algorithm that finds least-cost path

Routing Algorithm Classification

Global or decentralized info?

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

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

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

Notations:

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

Dijsktra's Algorithm for Source Node u

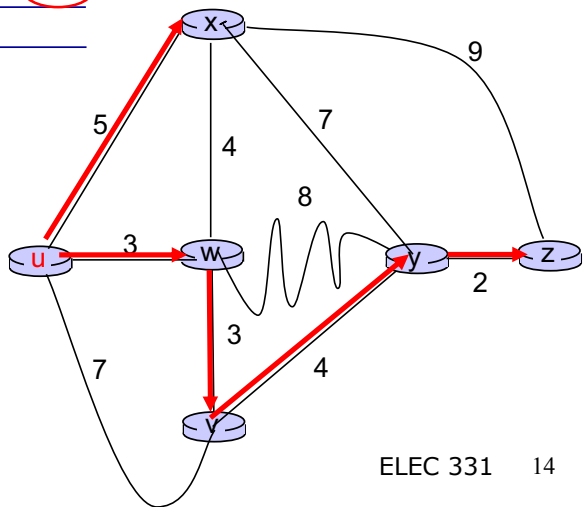
```
1  Initialization:
2     N' = {u}
3     for all nodes v
4        if v is a neighbor of u
5           then D(v) = c(u,v)
6        else D(v) = ∞
7
8  Loop
9        find w not in N' such that D(w) is a minimum
10       add w to N'
11       update D(v) for each neighbor v of w and not in N' :
12    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

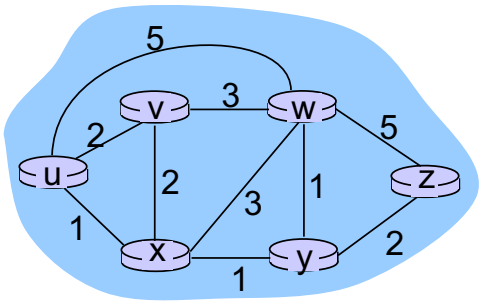
notes:

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



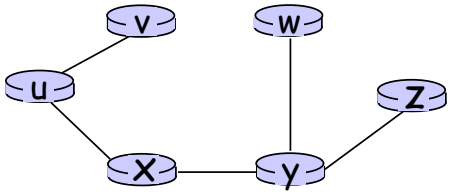
Dijkstra's algorithm: Another example

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



Dijkstra's algorithm: Example (cont.)

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)

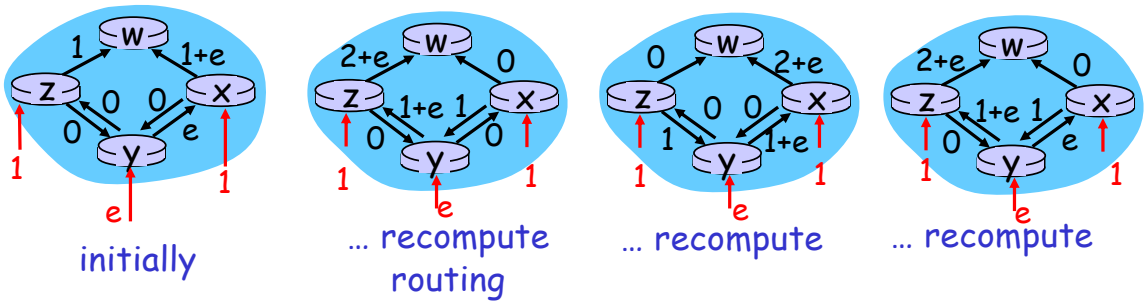
Dijkstra's algorithm, discussion

Algorithm complexity: n nodes (not counting the source)

- each iteration: need to check all nodes, w, not in N'
- n(n+1)/2 comparisons: $O(n^2)$

Oscillations possible:

- e.g., link cost = amount of carried traffic
- link costs are not symmetric



- Solution: Link advertisement be sent at random time.

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)

Define

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

Then

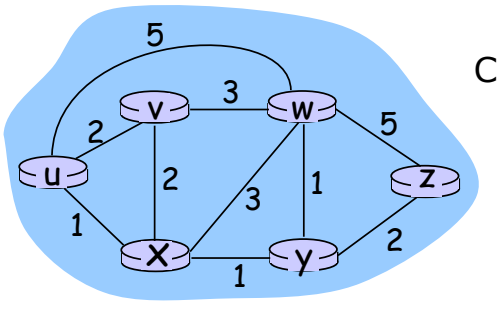
$$d_x(y) = \min_v \{c(x,v) + d_v(y)\}$$

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

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

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

Bellman-Ford example



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

Bellman-Ford equation says:

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

$$= \min \{ 2 + 5, \; 1 + 3, \; 5 + 3 \} = 4$$

The neighboring node that achieves minimum is the next hop in shortest path ➜ forwarding table

Distance Vector Algorithm

- Let N denote the set of nodes in the network
- Each node x maintains the following info

1. Local topology: Node x knows the cost to each neighbor v: c(x,v)
2. Node x maintains its own distance vector

$$\mathbf{D}_x = [D_x(y): y \in N]$$

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

3. Node x also maintains its neighbors' distance vectors
   - For each neighbor v of x, node x maintains

$$\mathbf{D}_v = [D_v(y): y \in N]$$

Distance Vector Algorithm

Basic idea:

- From time to time, each node sends its own distance vector estimate to neighbors
- When a node x receives new DV estimate from its neighbor, node x updates its own DV using Bellman-Ford equation:

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

- If node x's distance vector has changed after the update, it will send its updated distance vector $\mathbf{D}_x$ to all of its neighbors.
- Under minor, natural conditions, the estimate $D_x(y)$ *converge to the actual least cost* $d_x(y)$

Distance Vector Algorithm

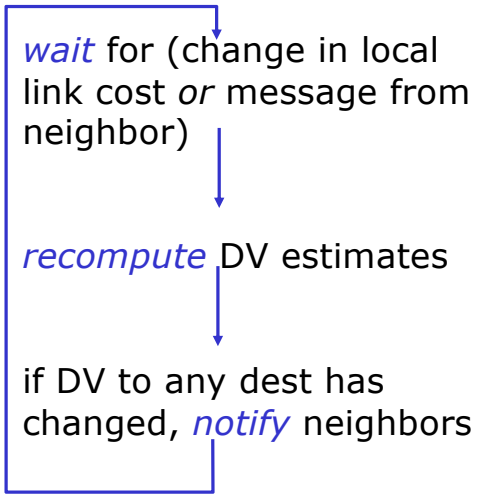
Iterative, asynchronous: each local iteration caused by:

- local link cost change
- DV update message from neighbor

Distributed:

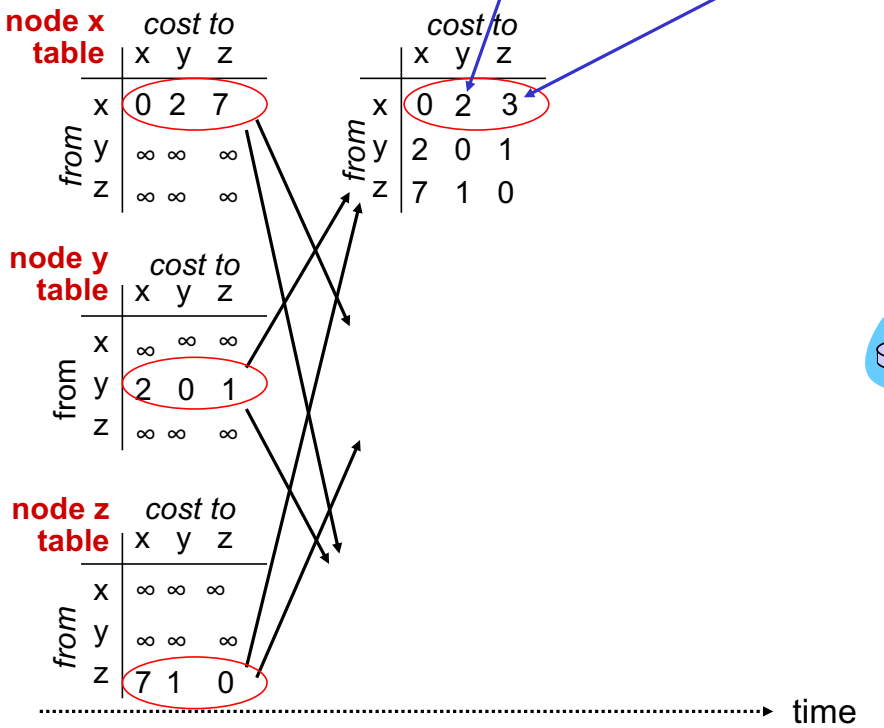
- each node notifies neighbors *only* when its DV changes
  - neighbors then notify their neighbors if necessary

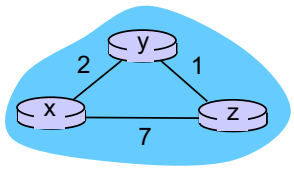
Each node:

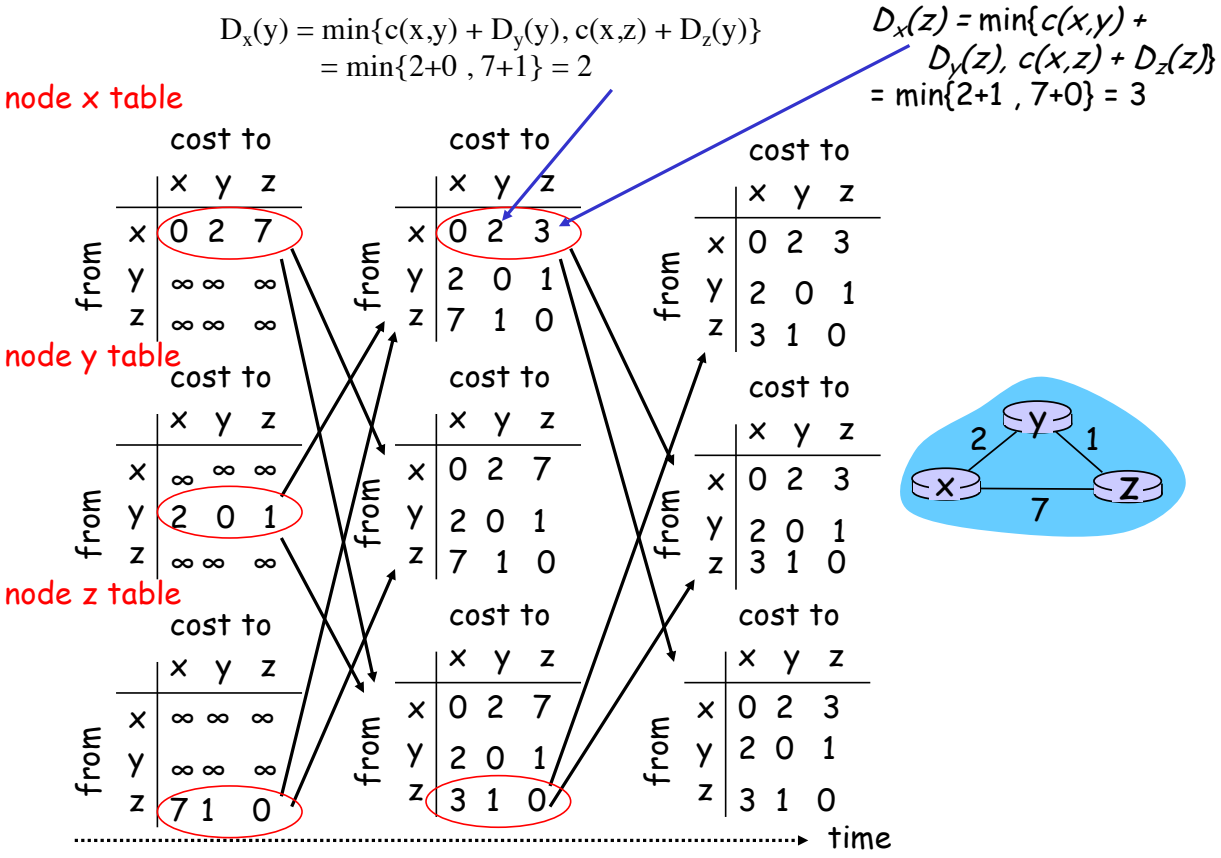


$D_x(y) = \min\{c(x,y) + D_y(y), c(x,z) + D_z(y)\}$
$= \min\{2+0, 7+1\} = 2$

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



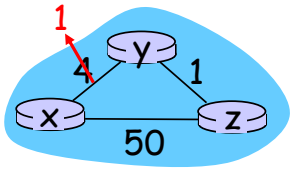




Distance Vector: link cost changes

Link cost changes:

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



"good news travels fast"

At time t_0, y detects the link-cost change, updates its DV, and informs its neighbors.

At time t_1, z receives the update from y and updates its table. It computes a new least cost to x and sends its neighbors its DV.

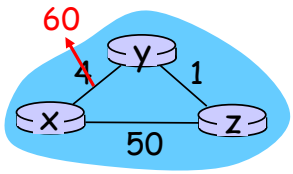
At time t_2, y receives z's update and updates its distance table. y's least costs do not change and hence y does *not* send any message to z.

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

Distance Vector: link cost changes

Link cost changes:

- bad news travels slow: routing loop
- "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

Message complexity

- LS: with n nodes, E links, O(nE) messages sent
- DV: exchange between neighbors only
  - convergence time varies

Speed of Convergence

- LS: $O(n^2)$ algorithm requires O(nE) messages
  - 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 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 - idealization

- all routers identical
- network “flat”

… *not* true in practice

Scale: with over 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")

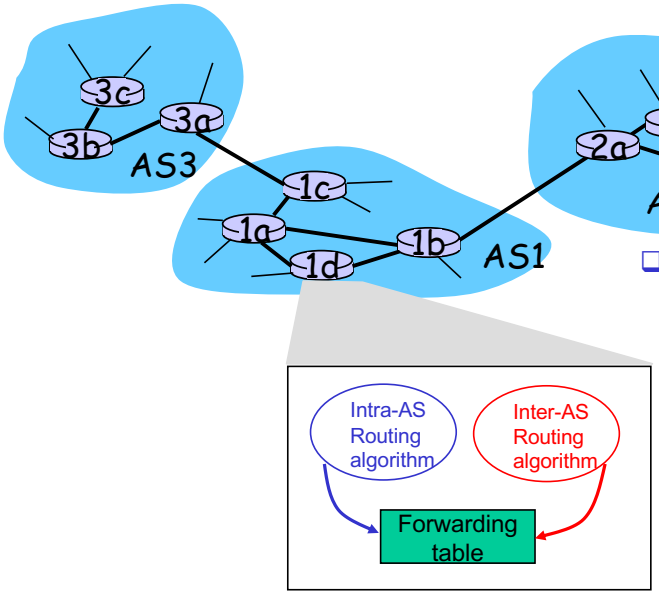
Intra-AS routing

- 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 AS'es

Inter-AS routing

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

Interconnected ASes



- Forwarding table is configured by both intra- and inter-AS routing algorithm
  - Intra-AS sets entries for internal destinations
  - Inter-AS & Intra-AS sets entries for external destinations

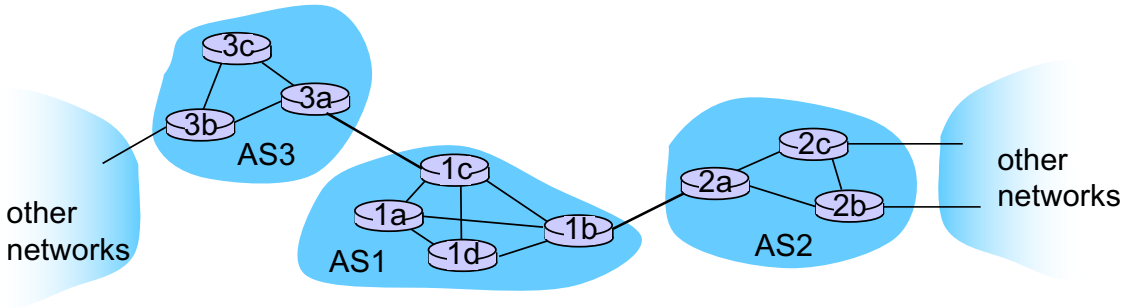
Inter-AS tasks

- Suppose router in AS1 receives datagram for which destination is outside of AS1
  - Router should forward packet towards one of the gateway routers, but which one?

AS1 must

1. learn which destinations are reachable through AS2 and 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
  - EIGRP: Enhanced Interior Gateway Routing Protocol (Cisco proprietary for decades, until 2016)

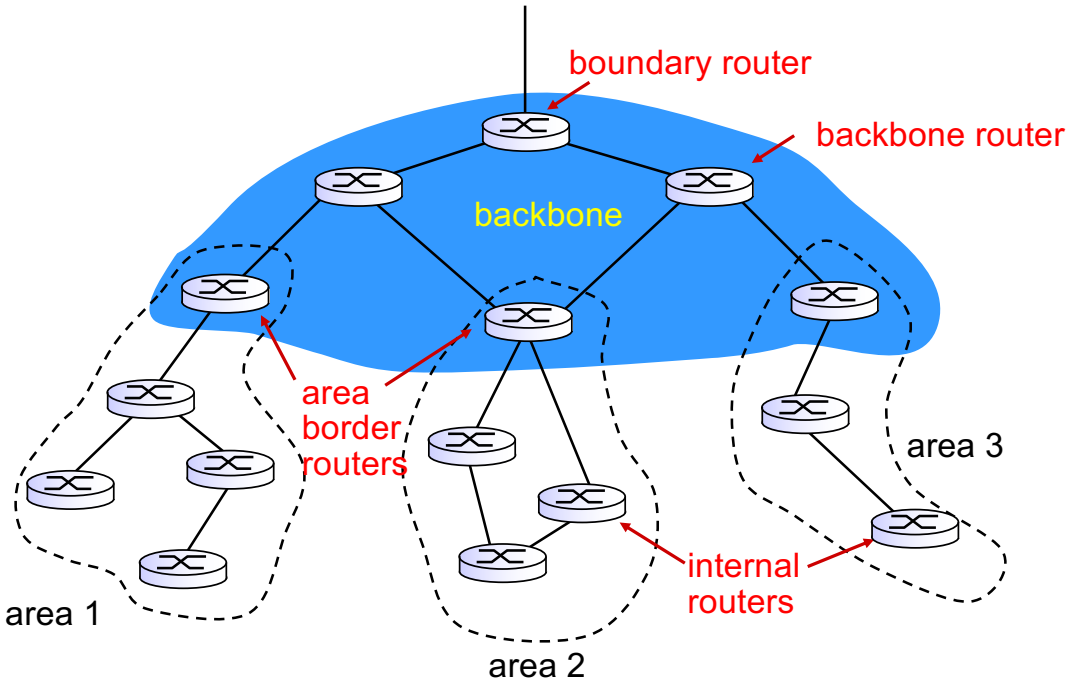
OSPF (Open Shortest Path First)

- "open": publicly available (RFC 1058, version 2 in RFC 2328, version 3 in RFC 5340 for IPv6)
- Uses Link State (LS) algorithm
  - LS packet dissemination
  - topology map of AS 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

OSPF "advanced" features

- Security: all OSPF messages authenticated (to prevent malicious intrusion) via shared secret keys
- Multiple same-cost paths allowed
- 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 uni- 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
- Each area runs its own OSFP link-state routing algorithm
  - Link-state advertisements only in an area
- Area border routers: "summarize" distances to subnets in own area, advertise to other area border routers
- Backbone routers: run OSPF routing limited to backbone
- Boundary routers: connect to other AS's

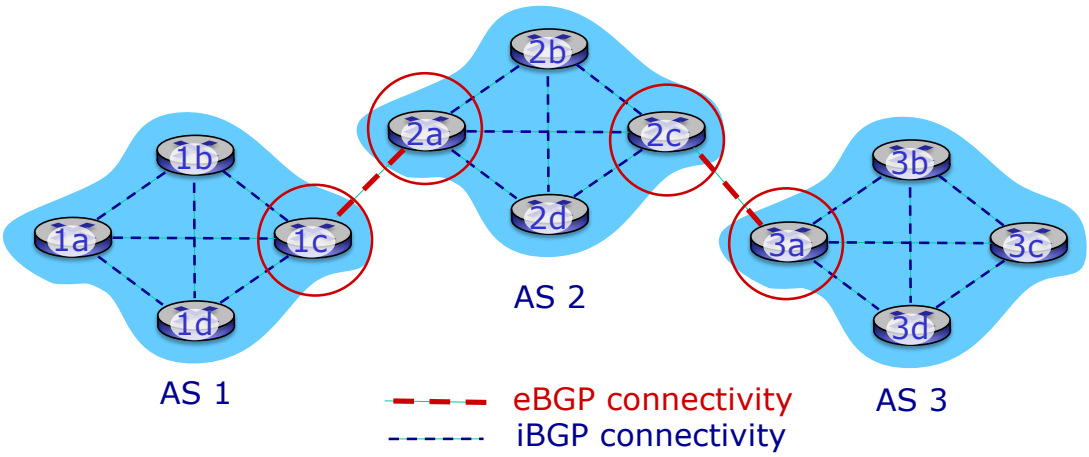
Chapter 5: Outline

- 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

Internet inter-AS routing: BGP (RFC 4271)

- BGP (Border Gateway Protocol): *the* de facto standard inter-domain routing protocol
- BGP provides each AS a means to:
  1. eBGP: Obtain subnet reachability information from neighboring ASes
  2. iBGP: Propagate the reachability information to all AS-internal routers
  3. Determine "good" routes to subnets based on reachability information and on AS policy
- Allows a subnet to advertise its existence to rest of the Internet: *"I am here"*

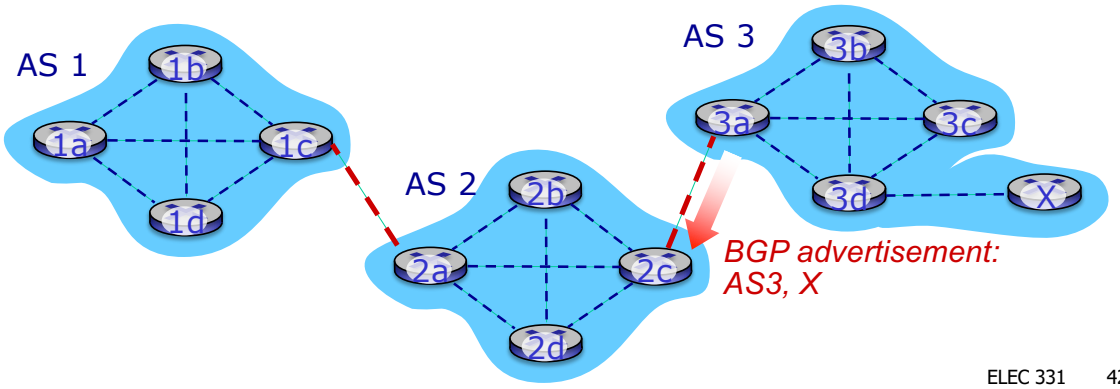
eBGP, iBGP connections



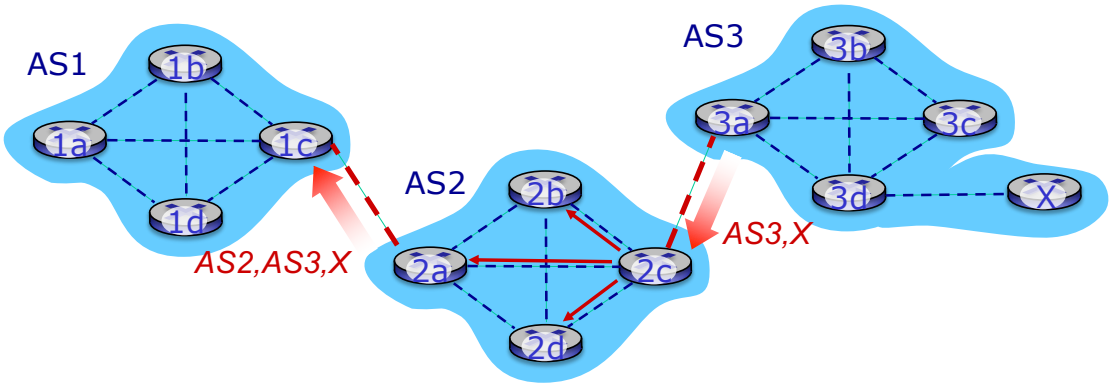
gateway routers run both eBGP and iBGP protocols

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



BGP path advertisement

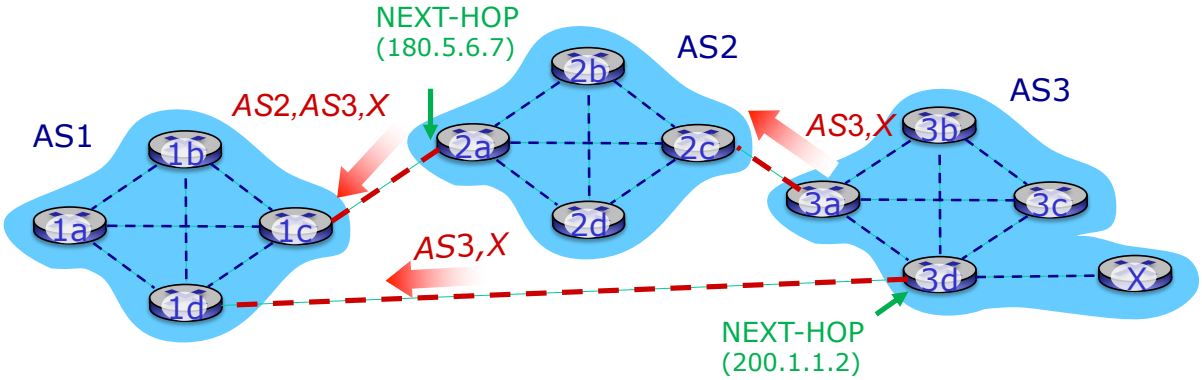


- AS2 router 2c receives path advertisement AS3,X (via eBGP) from AS3 router 3a
- Based on AS2 policy, AS2 router 2c accepts path AS3,X, propagates (via iBGP) to all AS2 routers
- Based on AS2 policy, AS2 router 2a advertises (via eBGP) path AS2, AS3, X to AS1 router 1c

Path attributes and BGP routes

- advertised prefix includes BGP attributes
  - prefix + attributes = “route”
- Two important attributes:
  - AS-PATH: contains the ASs through which the advertisement for the prefix passed: AS 67 AS 17
  - NEXT-HOP: indicates IP address of router interface that begins the AS-PATH. (There may be multiple links from current AS 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 other neighboring ASes

BGP path advertisement

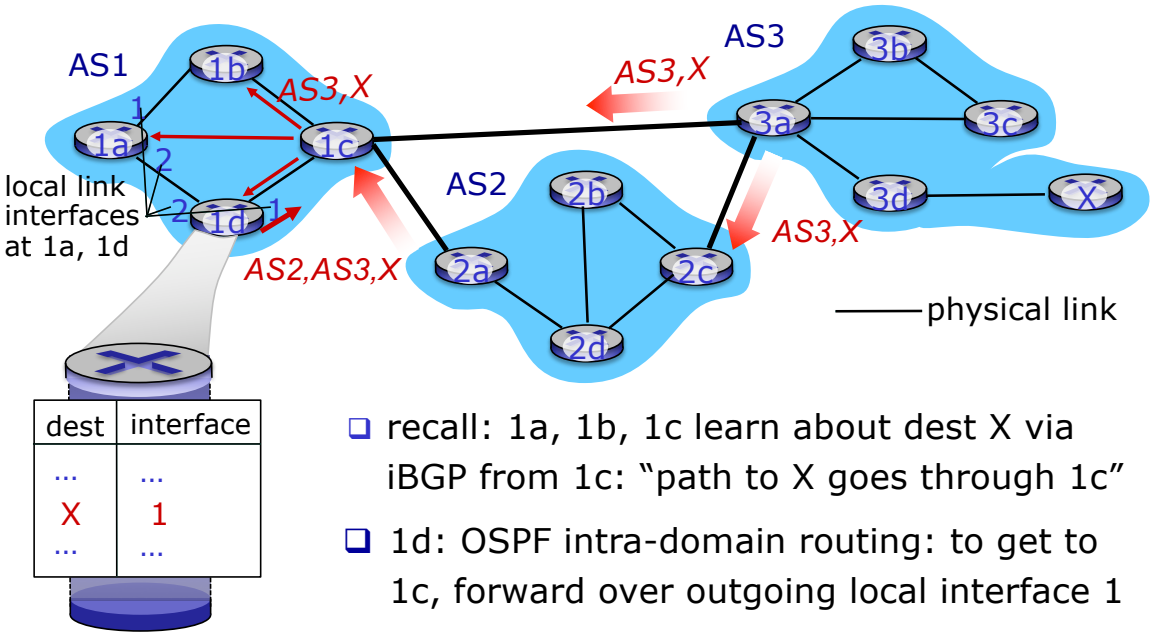


AS 1 may learn about multiple routes to destination prefix:

- AS1 gateway router 1c learns route 180.5.6.7; *AS2, AS3; X* from router 2a
- AS1 gateway router 1d learns route 200.1.1.2; *AS3; X* from router 3d

BGP, OSPF, forwarding table entries

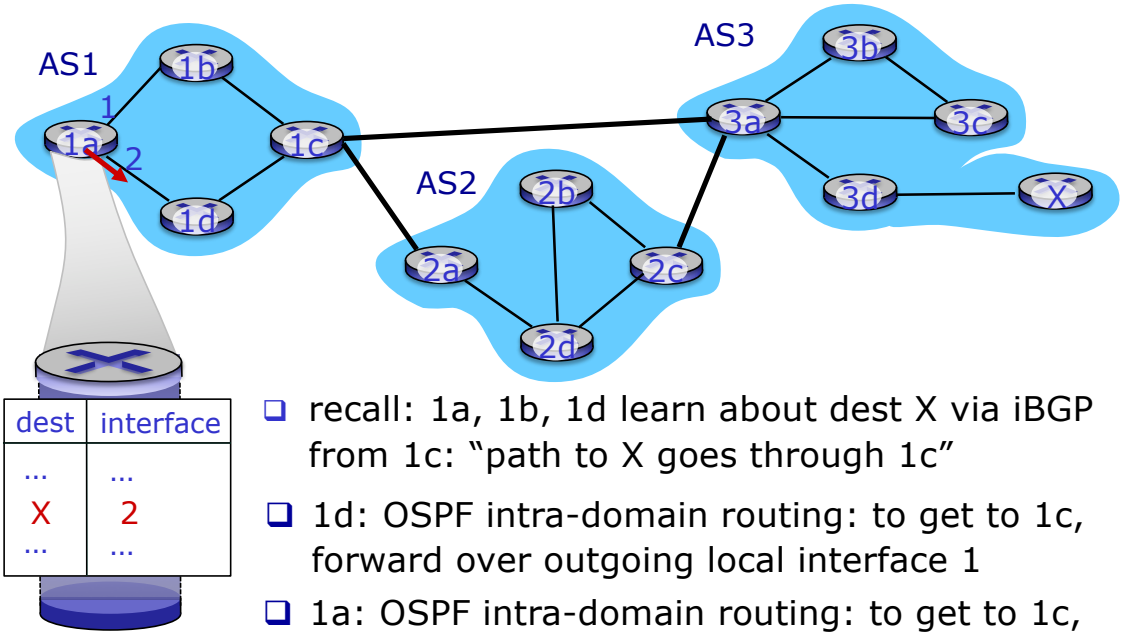
Q: how does router set forwarding table entry to distant prefix?



- 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

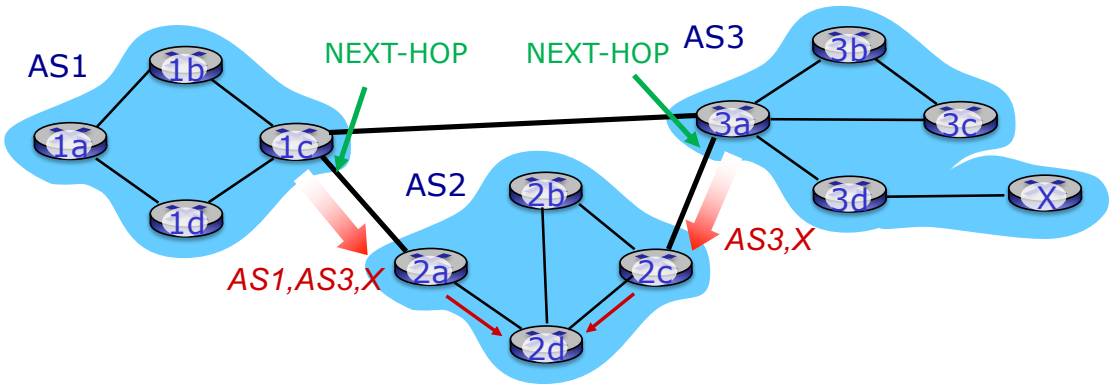
BGP, OSPF, forwarding table entries

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



- recall: 1a, 1b, 1d 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

Hot Potato Routing

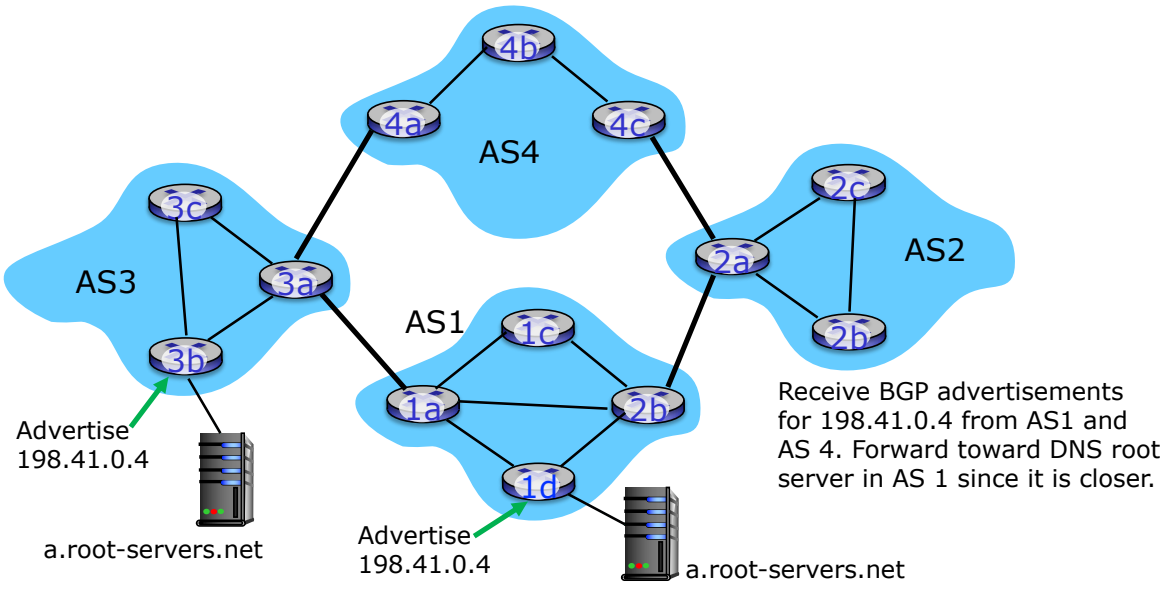


- 2d learns (via iBGP) it can route to X via 2a or 2c
- 2d will determine the least cost-path to NEXT-HOP router 1c and the least cost-path to NEXT-HOP router 3a, and choose the one with the minimum value
- *hot potato routing:* choose local gateway that has least cost

BGP route selection

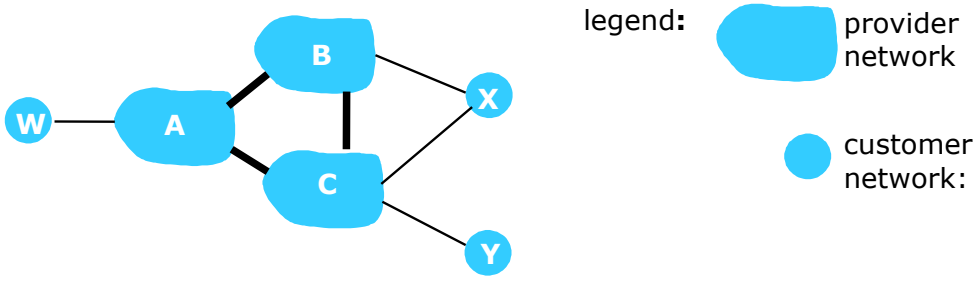
- Router may learn about more than one route to some prefix. Router must select one of the possible routes.
- Elimination rules:
  1. Local preference value attribute: policy decision and is set by AS's network administrator. Routes with the highest local preference are selected
  2. From the remaining routes (with same highest local preference value), select the route with shortest AS-PATH
  3. From the remaining routes (with same highest local preference value and same AS-PATH length), select the route with closest NEXT-HOP route using hot potato routing
  4. Additional criteria

IP-Anycast (IETF RFC 1546, 7094)



- IP-anycast is used by the DNS system to direct DNS queries to the closest root DNS server.

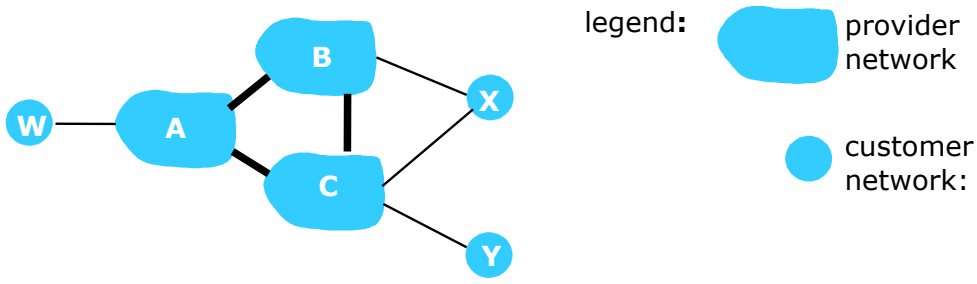
BGP: achieving policy via advertisements



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

BGP: achieving policy via advertisements



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

- A advertises path AW to B and to C
- B *chooses not to advertise* BAW to C:
  - B gets no "revenue" for routing CBAW, since none of C, A, W are B's customers
  - C does not learn about CBAW path
- C will route CAW (not using B) to get to W

Why different Intra- and 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, reduced update traffic

Performance:

- Intra-AS: can focus on performance
- Inter-AS: policy may dominate over performance

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

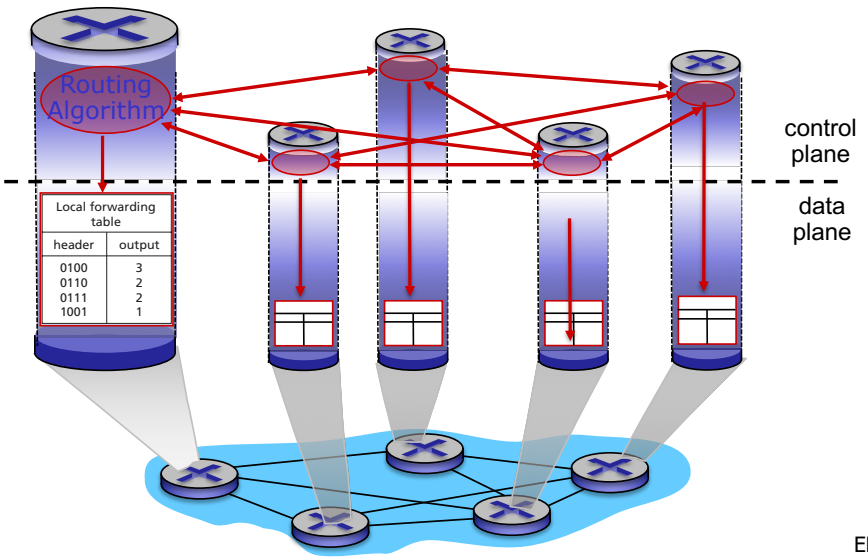
Software-Defined Networking (SDN)

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

- ~2005: renewed interest in rethinking network control plane

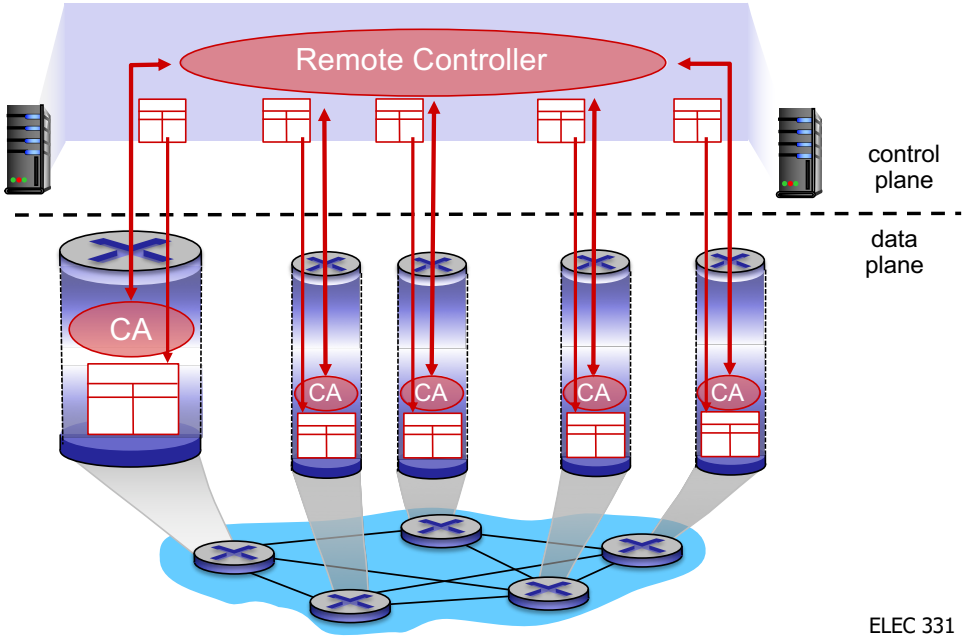
Recall: per-router control plane

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



Recall: logically centralized control plane

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

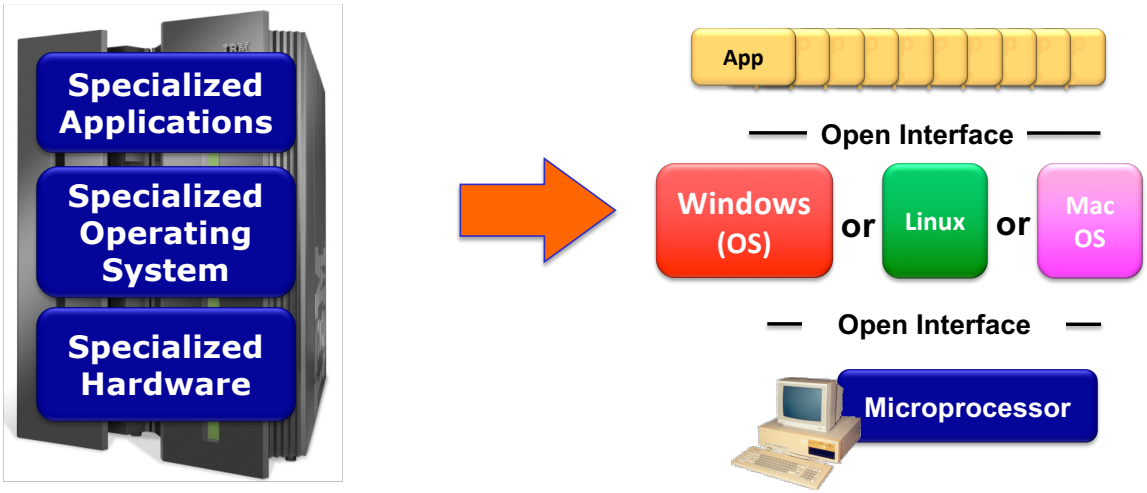


Software-Defined Networking (SDN)

Why a *logically centralized* control plane?

- 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

Analogy: mainframe to PC evolution*

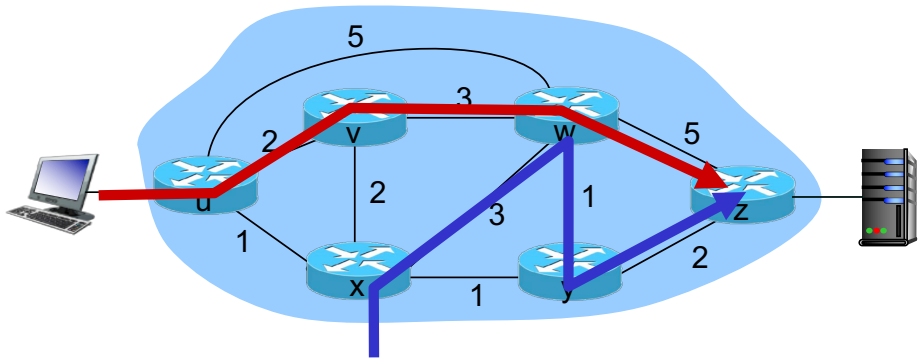


Vertically integrated
Closed, proprietary
Slow innovation
Small industry

Horizontal
Open interfaces
Rapid innovation
Huge industry

* Slide courtesy: N. McKeown

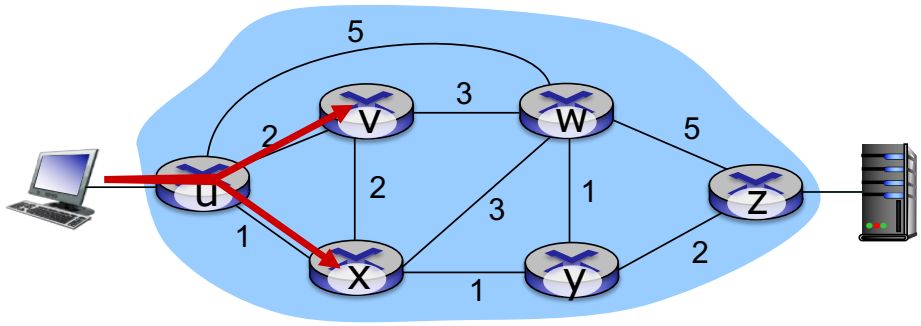
Traffic engineering: difficult



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

A: can't do it (with destination based forwarding, and link-state, distance-vector routing)

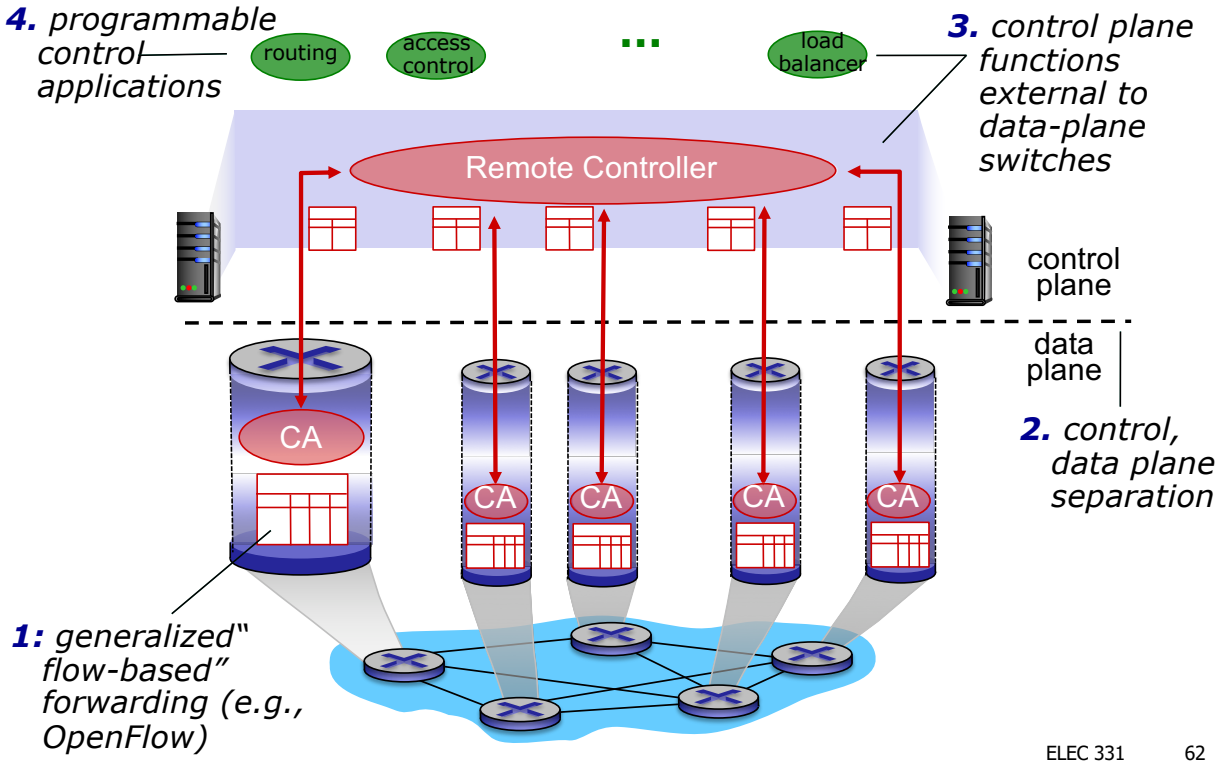
Traffic engineering: difficult



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)

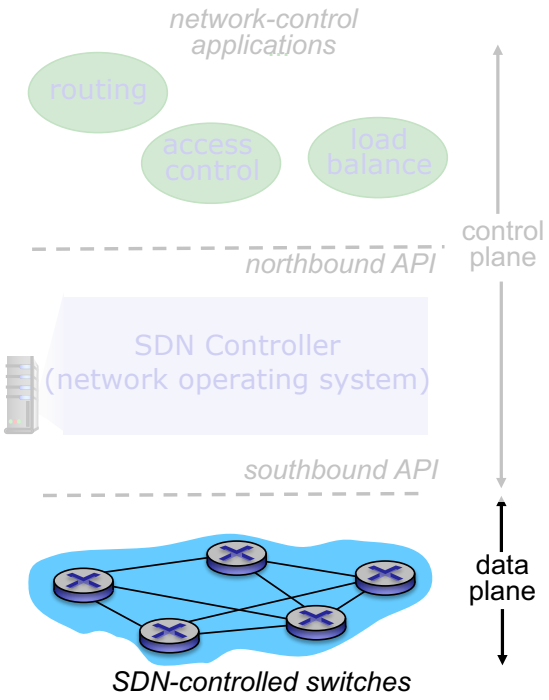
Software-Defined Networking (SDN)



SDN perspective: data plane switches

Data plane switches

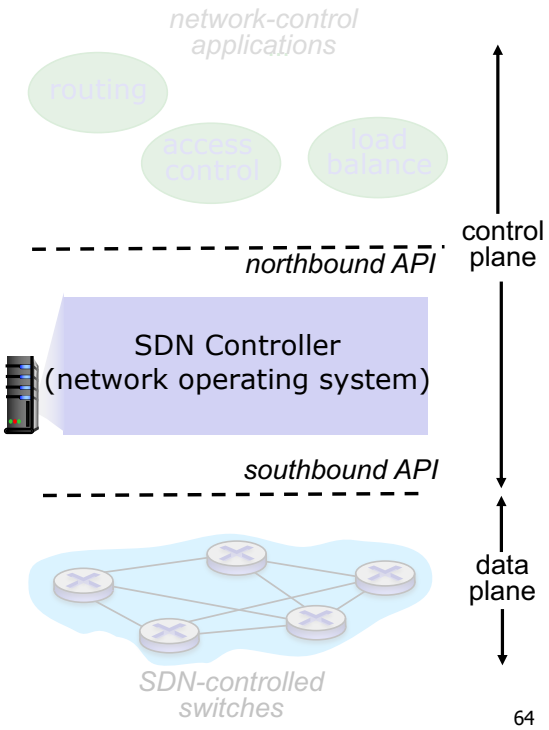
- fast, simple, commodity switches implementing generalized data-plane forwarding (Section 4.4) in hardware
- switch flow table computed, installed by controller
- API for table-based switch control (e.g., OpenFlow)
  - defines what is controllable and what is not
- protocol for communicating with controller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



SDN perspective: SDN controller

SDN controller (network OS):

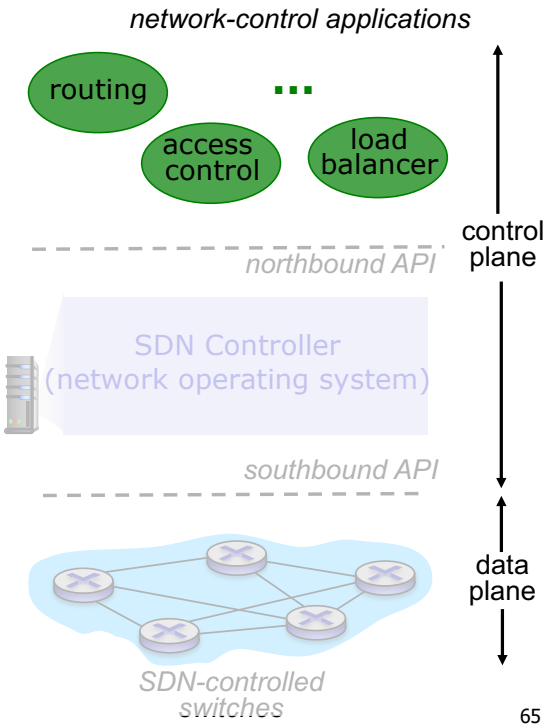
- maintain network state information
- interacts with network control applications "above" via northbound API
- interacts with network switches "below" via southbound API
- implemented as distributed system for performance, scalability, fault-tolerance, robustness



SDN perspective: control applications

network-control apps:

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

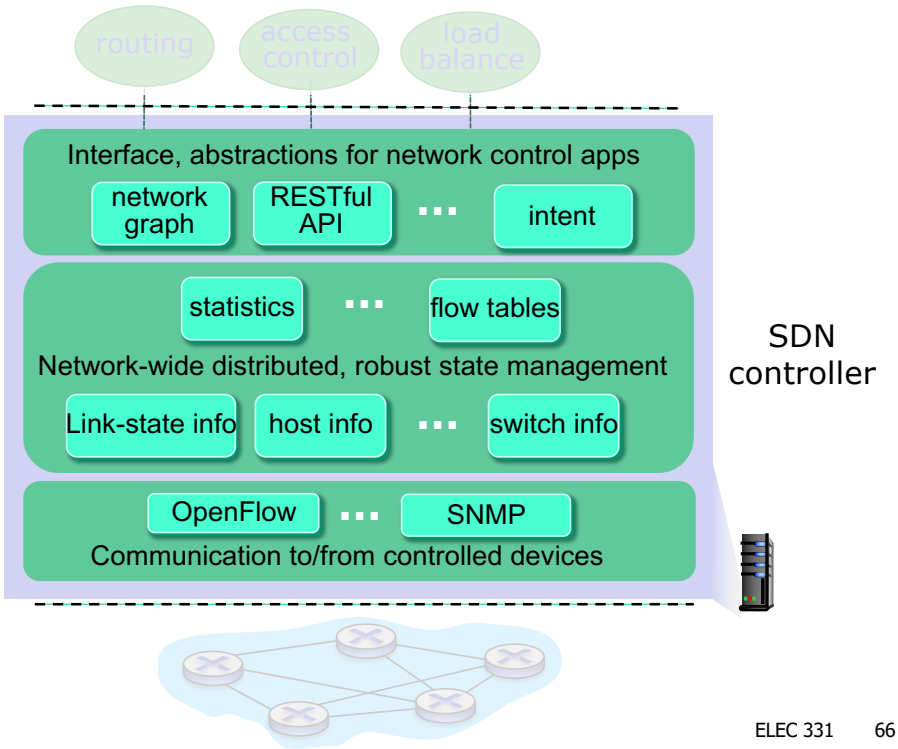


Components of SDN controller

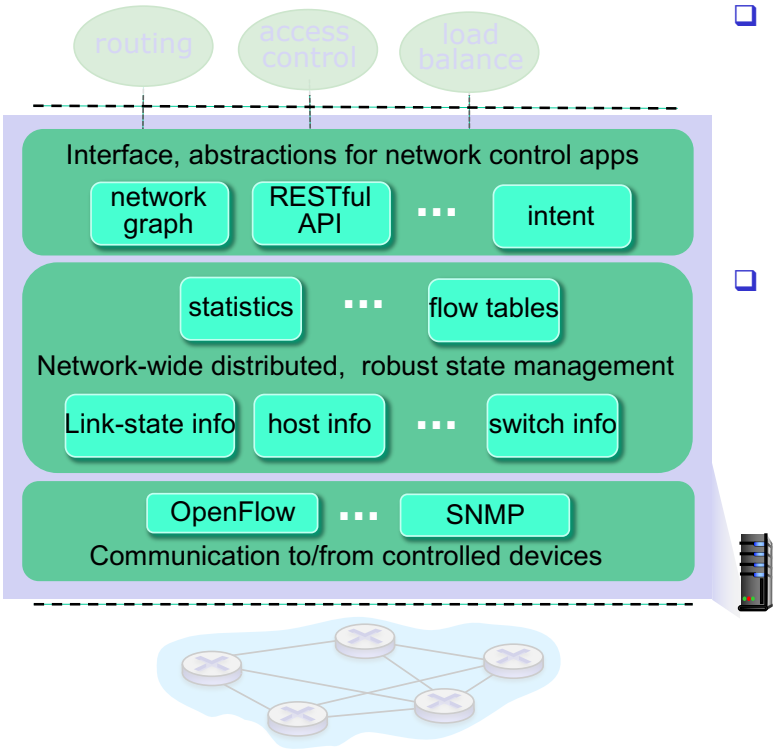
Interface layer to network control apps: abstractions API

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

Communication layer: communicate between SDN controller and controlled switches

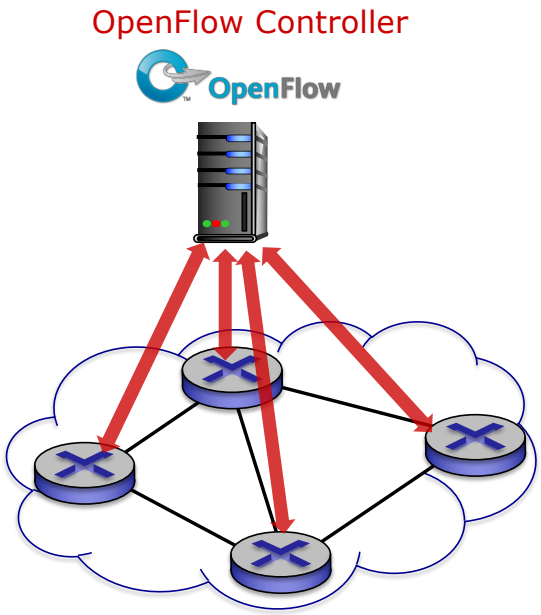


Components of SDN controller



- *RESTful API:* Representational State Transfer (REST) request-response interface
- *intent:* abstracts network objects and capabilities, expresses user's intent on what to do declaratively, but not how to do
  - **block** an HTTP flow
  - **adjust** bandwidth (data rate) of a connection to 1 Gb/s

OpenFlow protocol

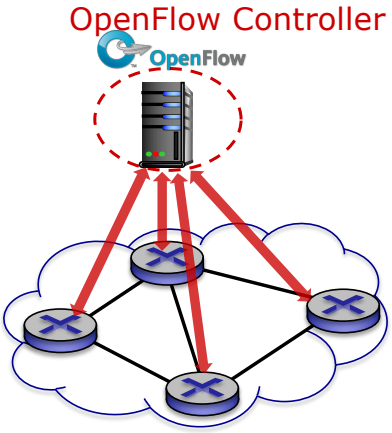


- operates between an SDN controller and an SDN-controlled switch
- TCP (port #6653) used to exchange messages
  - optional encryption
- three classes of OpenFlow messages:
  - controller-to-switch
  - asynchronous (switch to controller)
  - symmetric (misc)

OpenFlow: Controller-to-Switch Messages

Key controller-to-switch messages

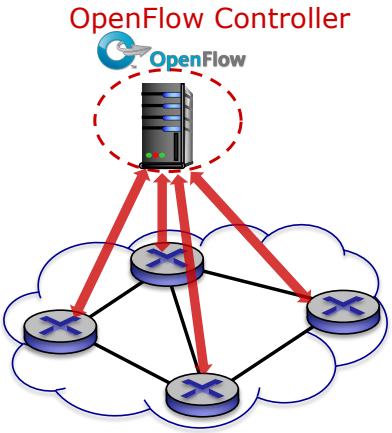
- *configuration:* controller to query and set a switch's configuration parameters
- *modify-state:* add, delete, modify flow table entries, set switch port properties
- *read-state:* controller collects statistics and counter values from switch's flow table and ports
- *send-packet:* controller can send a packet out of specific switch port at the controlled switch. Message contains packet to be sent in its payload



OpenFlow: Switch-to-Controller Messages

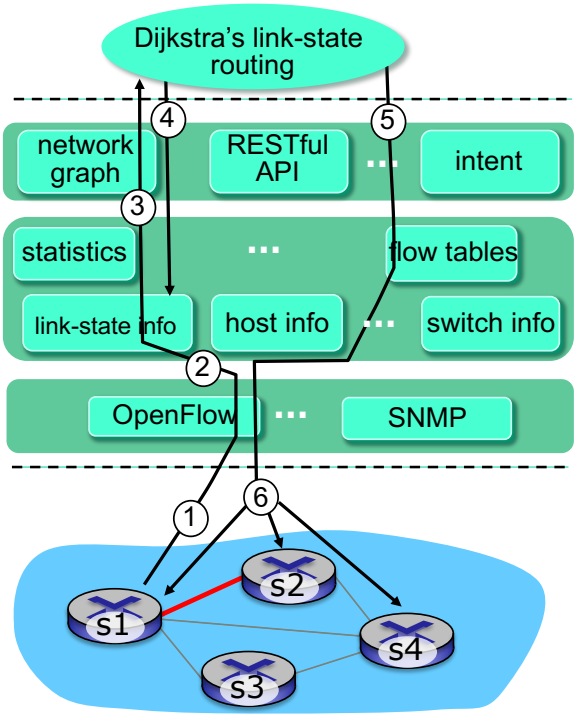
Key switch-to-controller messages

- *flow-removed:* informs the controller that flow table entry deleted at switch
- *port-status:* inform the controller of a change in port status
- *packet-in:* transfer packet to controller for additional processing. See send-packet message from controller



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

SDN: Control/Data Plane Interaction Example



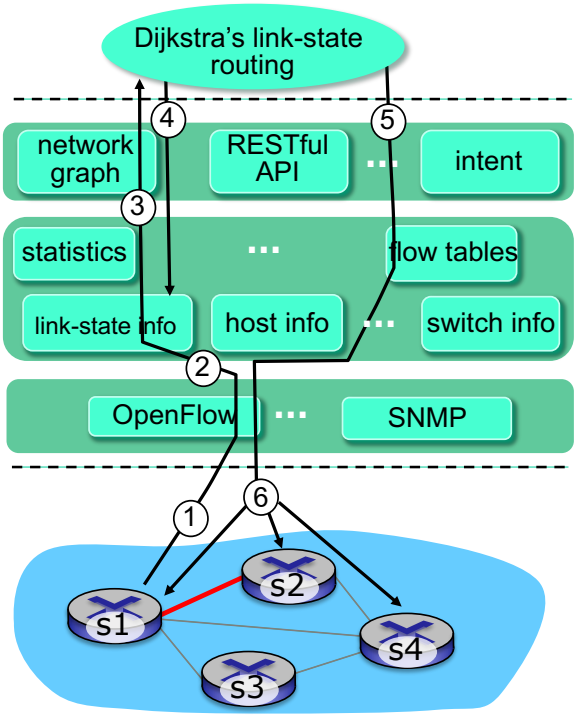
① Switch s1, experiencing link failure, uses OpenFlow *port-status* message to notify controller

② SDN controller receives OpenFlow message, notifies link-state manager, which updates a link-state database

③ Network-control application "Dijkstra's routing algorithm" has previously registered to be called whenever link status changes. That app receives notification of link-state change

④ The app "Dijkstra's routing algorithm" accesses network graph info, link state info in controller, computes new routes

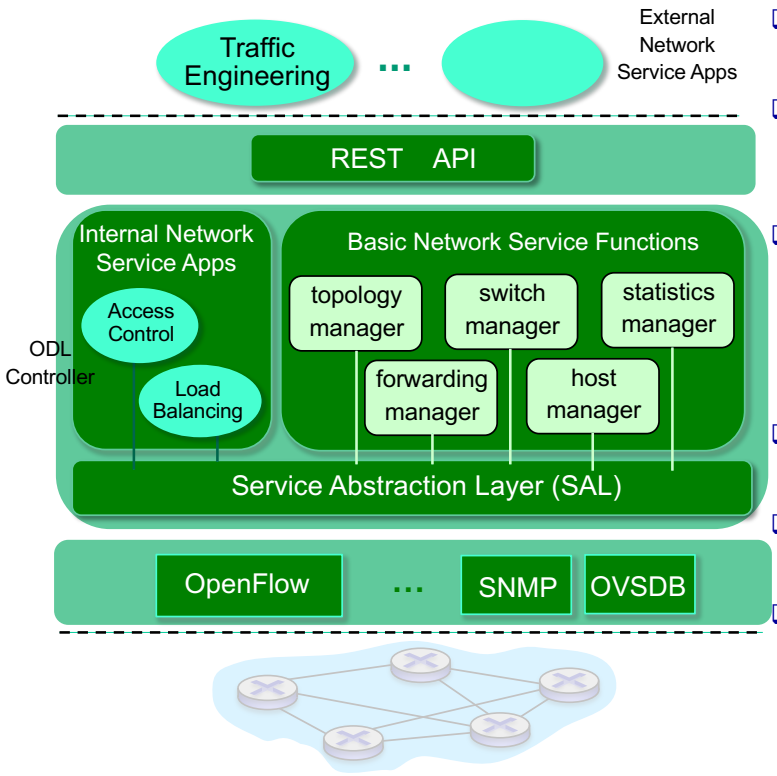
SDN: Control/Data Plane Interaction Example



⑤ link state routing app interacts with flow table manager in SDN controller, which determines flow tables to be updated

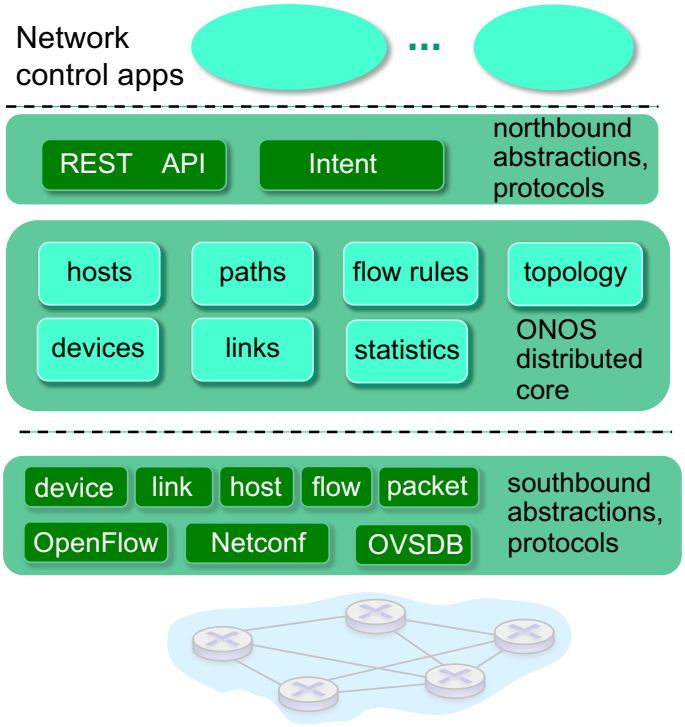
⑥ flow table manager uses OpenFlow protocol to update flow table entries at affected switches

OpenDayLight (ODL) Controller



- ODL Lithium controller platform
- network apps may be contained within, or be external to SDN controller
- external apps communicate with controller modules using a REST request-response API running over HTTP
- internal apps communicate with each other via SAL
- SNMP: network management protocol
- OVSDB: a protocol used to manage data center switching

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

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 to each other
  - error reporting: unreachable host, network, port, protocol
  - echo request/reply (used by ping)
- network-layer "above" IP:
  - ICMP messages carried in IP datagrams

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

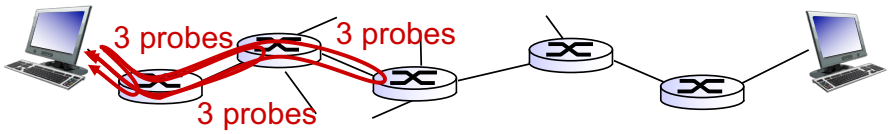
- ICMP message: type, code field, plus header and first 8 bytes of IP datagram causing error

Traceroute and ICMP

- Source sends series of UDP segments to destination
  - First segment has TTL =1
  - Second has TTL = 2, etc
  - Unlikely dest port number
- When n^{th} datagram arrives at n^{th} router (TTL has expired):
  - Router discards datagram and sends to source an ICMP message (type 11, code 0)
  - Message includes name of router & its IP address
- When ICMP message arrives, source calculates RTT
- Traceroute does this 3 times

Stopping criterion

- UDP segment eventually arrives at destination host
- Destination returns ICMP "dest port unreachable" packet (type 3, code 3)
- When source gets this ICMP, it stops.



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

next stop: link layer!