

Réseaux fixes

I.1 Segmentation par routage

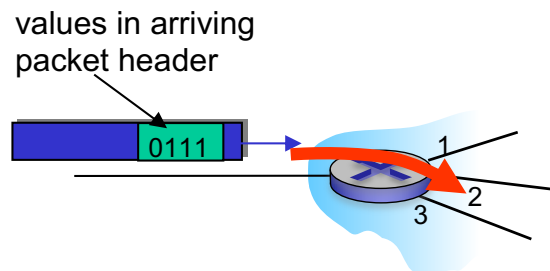
Luc Deneire

EII-5, Option Réseaux et Objets Connectés (ROC)

Network layer: data plane, control plane

Data plane:

- *local*, per-router function
- determines how datagram arriving on router input port is forwarded to router output port



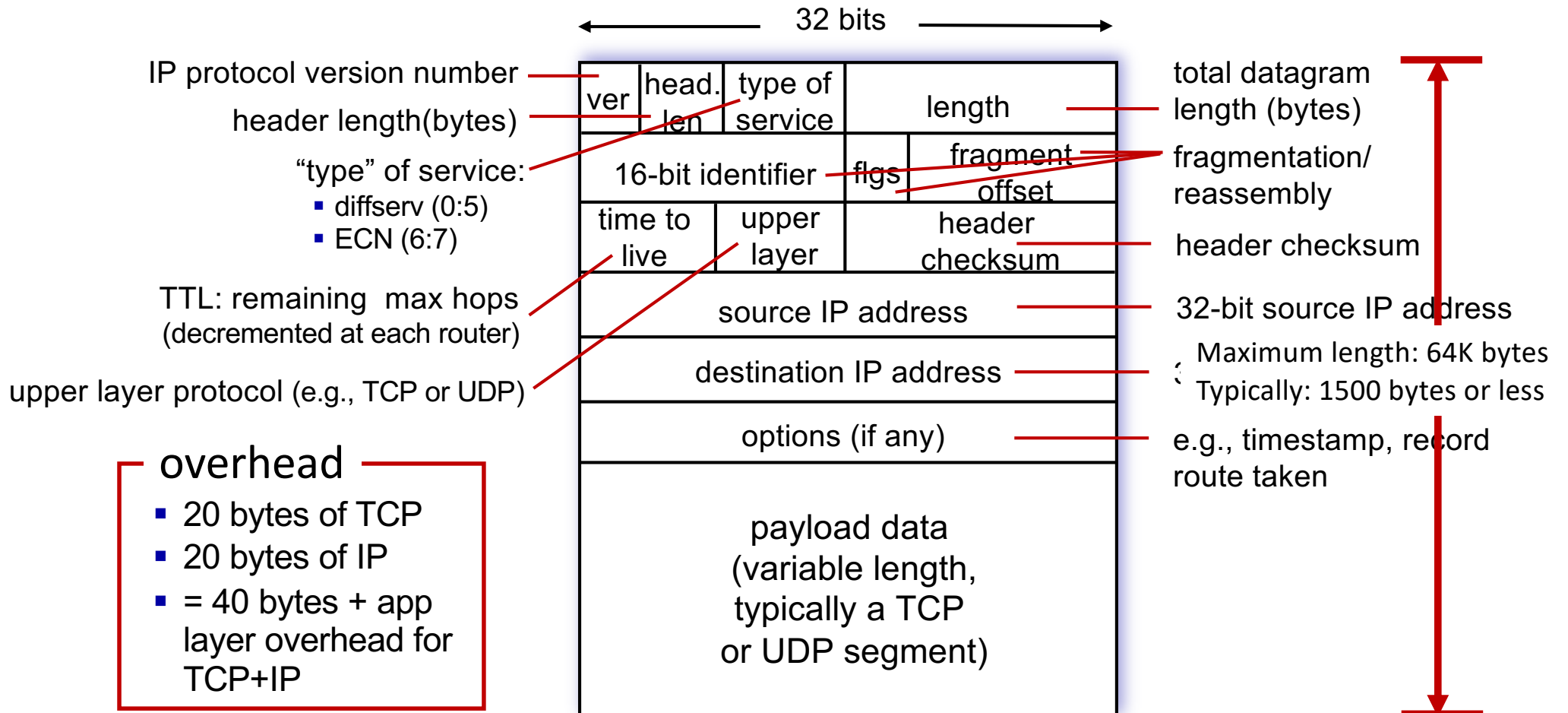
Control plane

network-wide logic

determines how datagram is routed among routers along end-end path from source host to destination host

- two control-plane approaches:
 - *traditional routing algorithms*: implemented in routers
 - *software-defined networking (SDN)*: implemented in (remote) servers

IP Datagram format

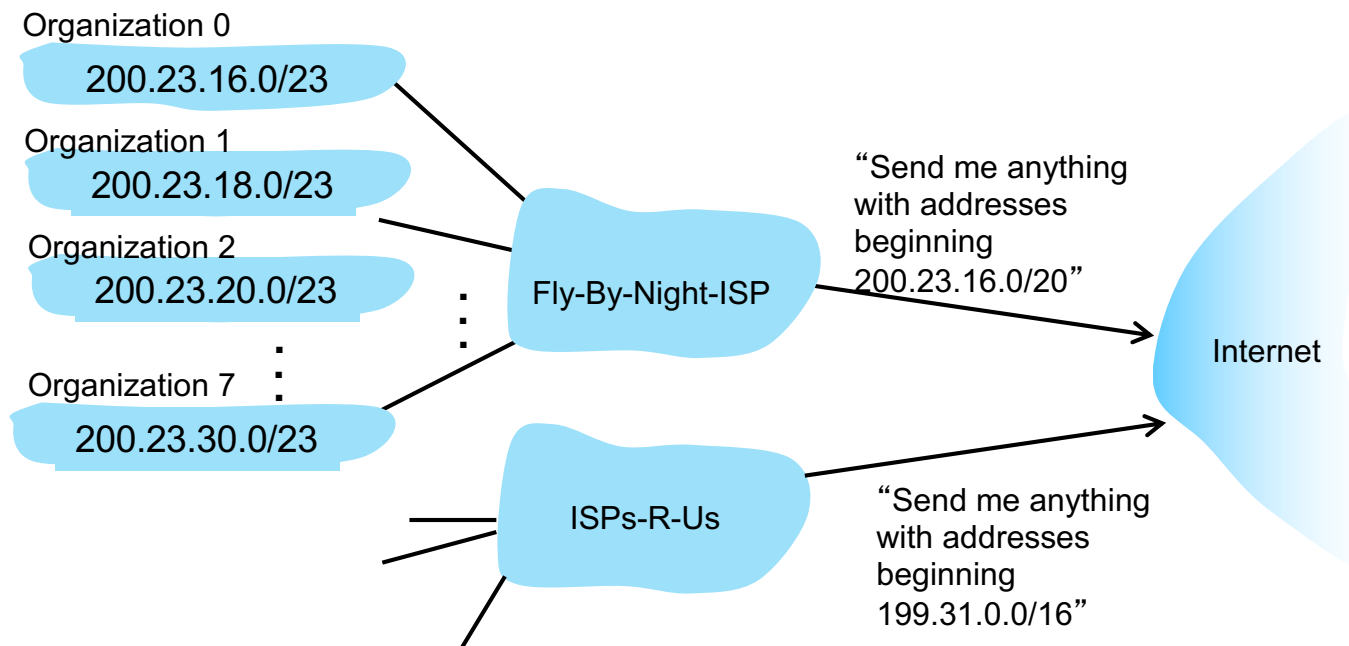


overhead

- 20 bytes of TCP
- 20 bytes of IP
- = 40 bytes + app layer overhead for TCP+IP

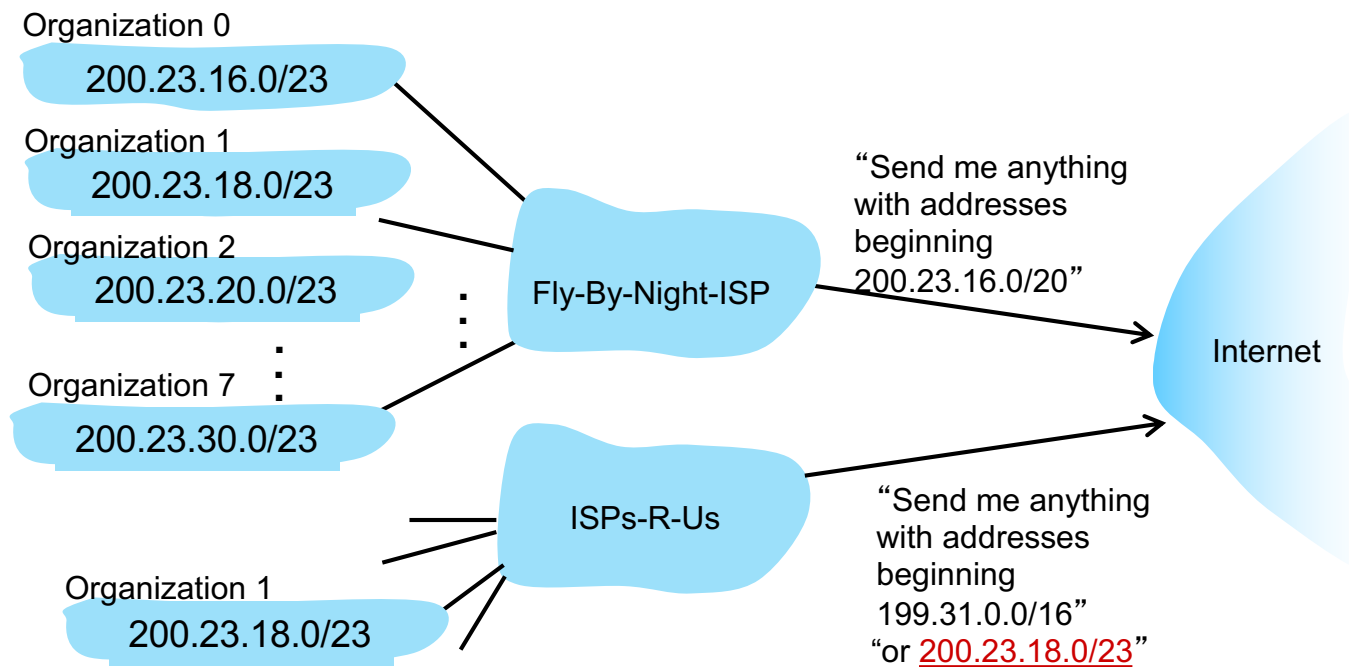
Hierarchical addressing: route aggregation

hierarchical addressing allows efficient advertisement of routing information:

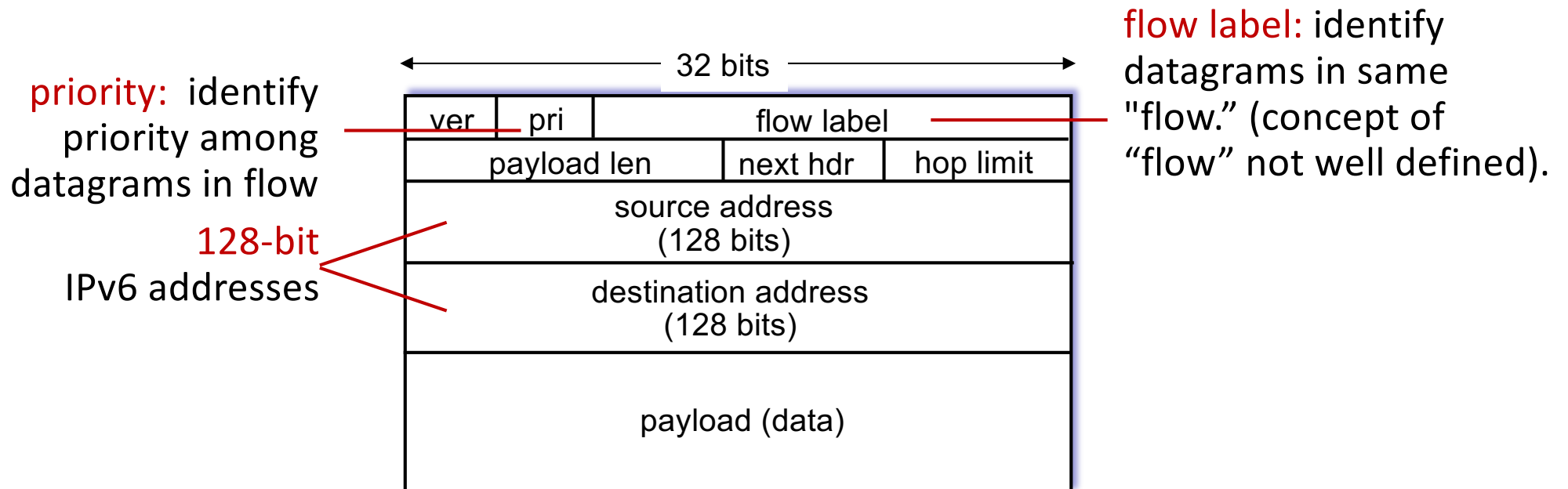


Hierarchical addressing: more specific routes

- Organization 1 moves from Fly-By-Night-ISP to ISPs-R-Us
- ISPs-R-Us now advertises a more specific route to Organization 1



IPv6 datagram format



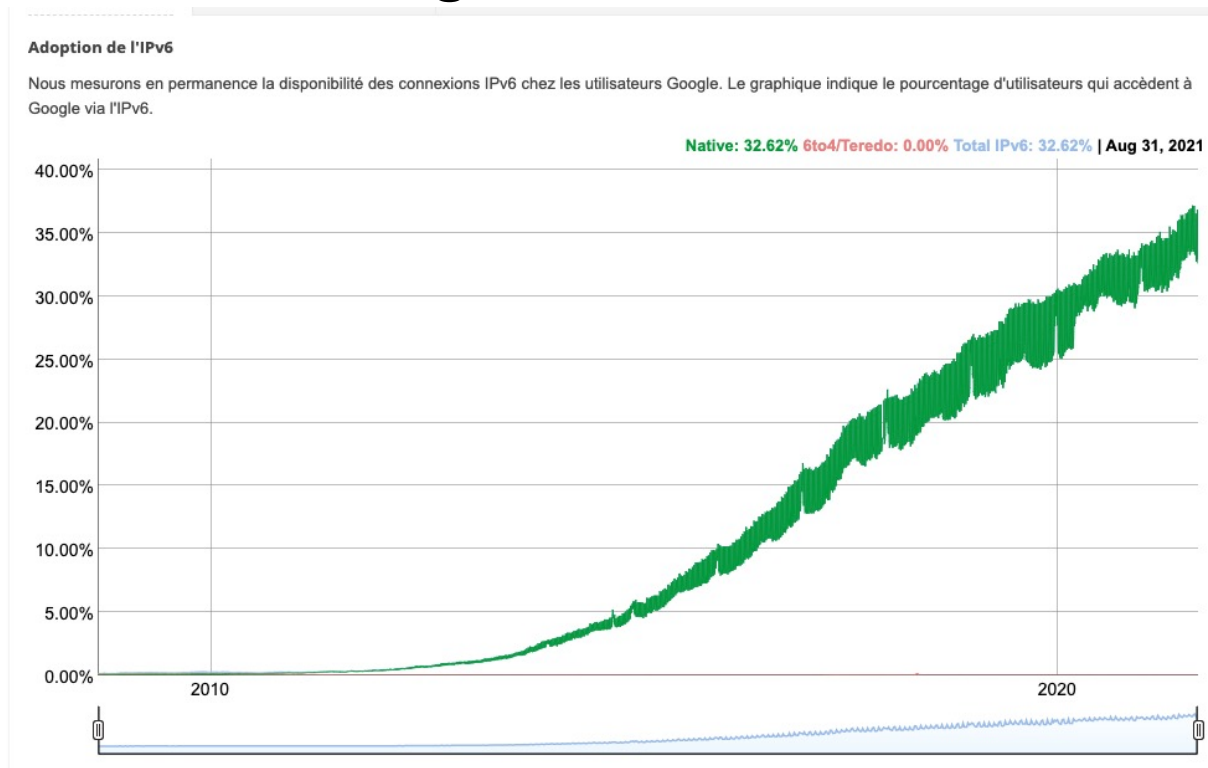
What's missing (compared with IPv4):

- no checksum (to speed processing at routers)
- no fragmentation/reassembly
- no options (available as upper-layer, next-header protocol at router)

IPv6: adoption

Google¹: ~ 33% of clients access services via IPv6

NIST: 1/3 of all US government domains are IPv6 capable



1

<https://www.google.com/intl/en/ipv6/statistics.html>

Network Layer: 4-9

Network layer: “control plane” roadmap

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



- network management, configuration
 - SNMP
 - NETCONF/YANG

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*

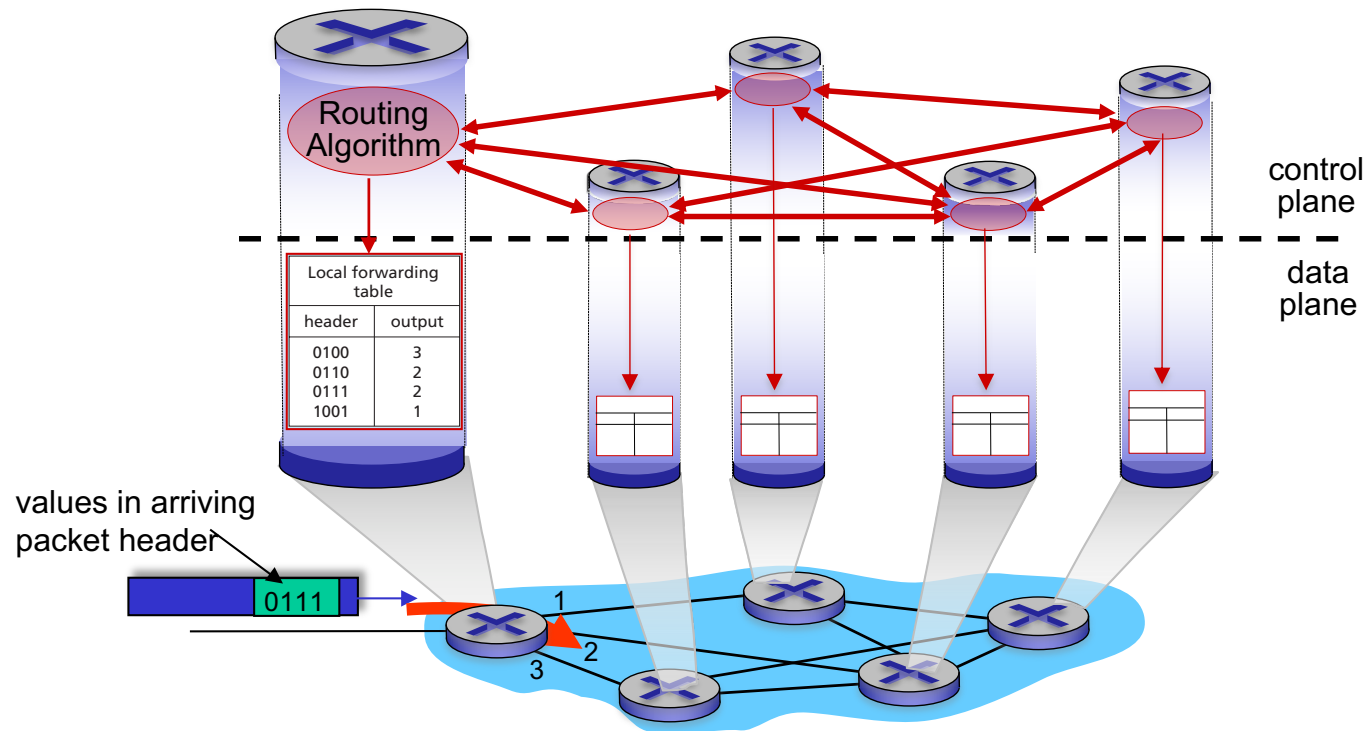
Two approaches to structuring network control plane:

per-router control (traditional)

logically centralized control (software defined networking)

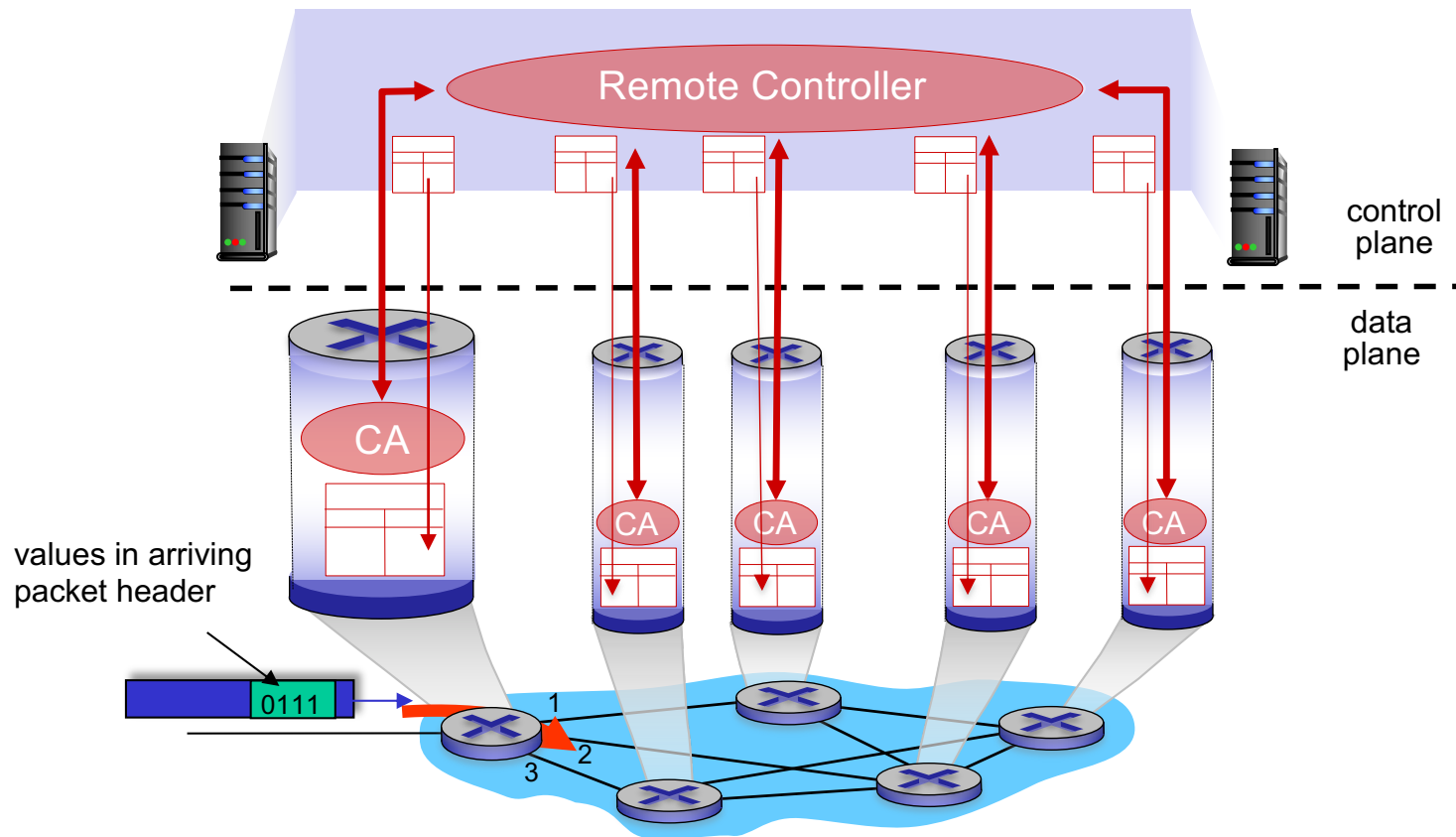
Per-router control plane

Individual routing algorithm components *in each and every router* interact in the control plane



Software-Defined Networking (SDN) control plane

Remote controller computes, installs forwarding tables in routers



Network layer: “control plane” roadmap

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



- network management, configuration
 - SNMP
 - NETCONF/YANG

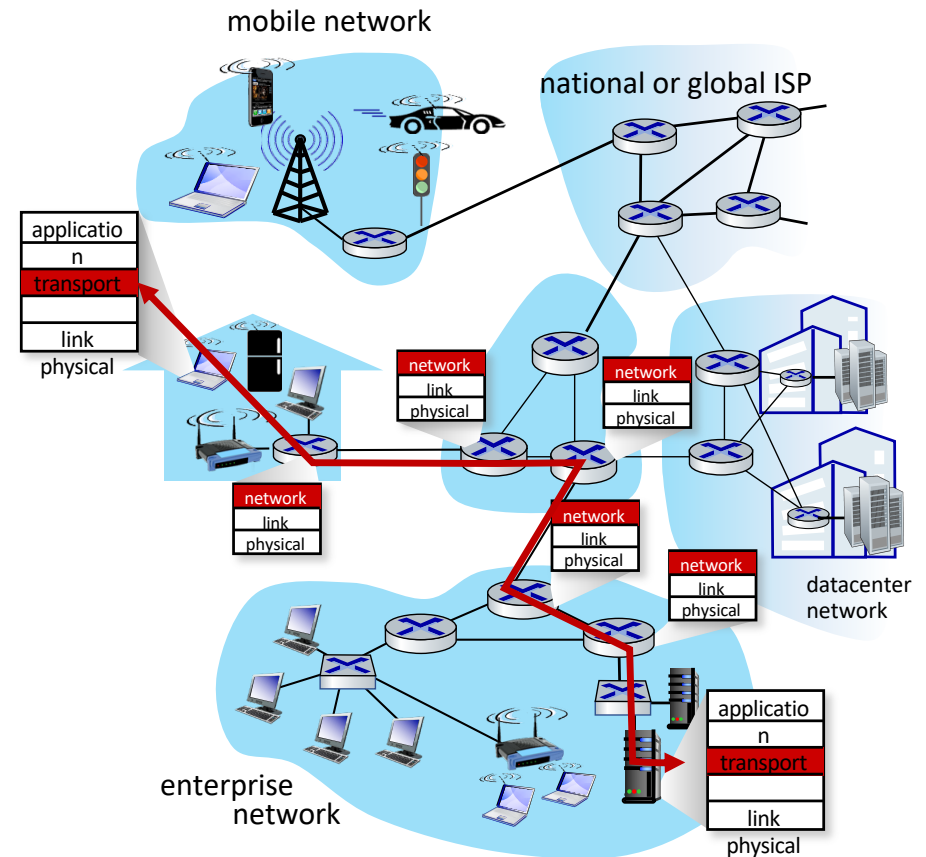
Routing protocols

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

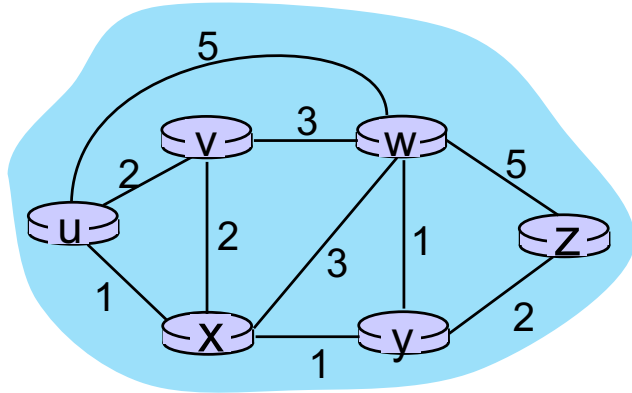
path: sequence of routers packets traverse from given initial source host to final destination host

“good”: least “cost”, “fastest”, “least congested”

routing: a “top-10” networking challenge!



Graph abstraction: link costs



$c_{a,b}$: cost of *direct* link connecting a and b
e.g., $c_{w,z} = 5$, $c_{u,z} = \infty$

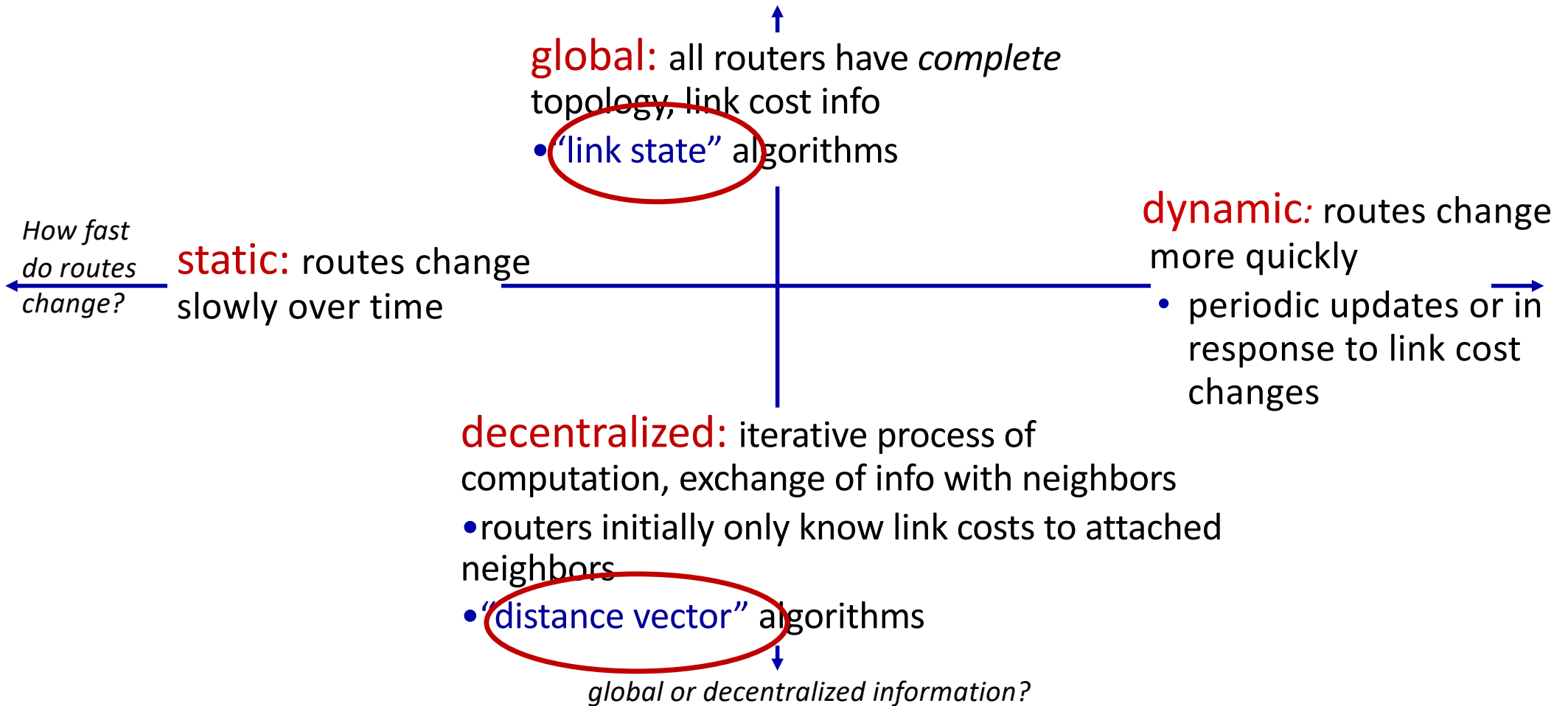
cost defined by network operator:
could always be 1, or inversely related
to bandwidth, or inversely related to
congestion

graph: $G = (N, E)$

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

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

Routing algorithm classification



Network layer: “control plane” roadmap

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



- network management, configuration
 - SNMP
 - NETCONF/YANG

Dijkstra's link-state routing algorithm

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

notation

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

Dijkstra's link-state routing algorithm

1 *Initialization:*

2 $N' = \{u\}$ */* compute least cost path from u to all other nodes */*

3 for all nodes v

4 if v adjacent to u */* u initially knows direct-path-cost only to direct neighbors */*

5 then $D(v) = c_{u,v}$ */* but may not be minimum cost! */*

6 else $D(v) = \infty$

7



8 *Loop*

9 find w not in N' such that $D(w)$ is a minimum

10 add w to N'

11 update $D(v)$ for all v adjacent to w and not in N' :

12 $D(v) = \min (D(v), D(w) + c_{w,v})$

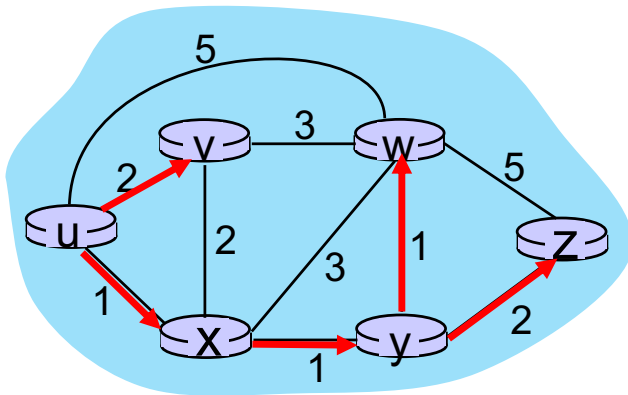
13 */* new least-path-cost to v is either old least-cost-path to v or known*

14 *least-cost-path to w plus direct-cost from w to v */*

15 *until all nodes in N'*

Dijkstra's algorithm: an example

Step	N'	$D(v), p(v)$	$D(w), p(w)$	$D(x), p(x)$	$D(y), p(y)$	$D(z), p(z)$
0	u	2, u	5, u	1, u	∞	∞
1	u, x	2, u	4, x		2, x	∞
2	u, x, y	2, u	3, y			4, y
3	u, x, y, v		3, y			4, y
4	u, x, y, w					4, y
5	u, x, y, w, z					



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

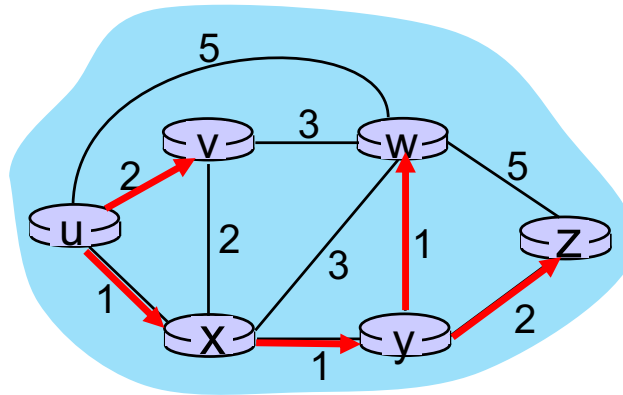
find a not in N' such that $D(a)$ is a minimum

add a to N'

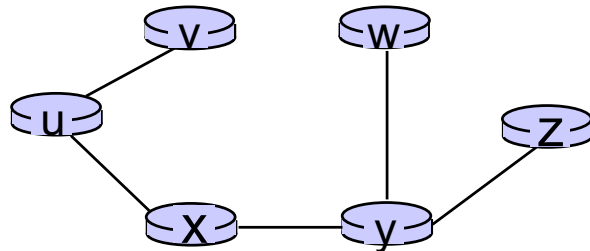
update $D(b)$ for all b adjacent to a and not in N' :

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

Dijkstra's algorithm: an example



resulting least-cost-path tree from u:



resulting forwarding table in u:

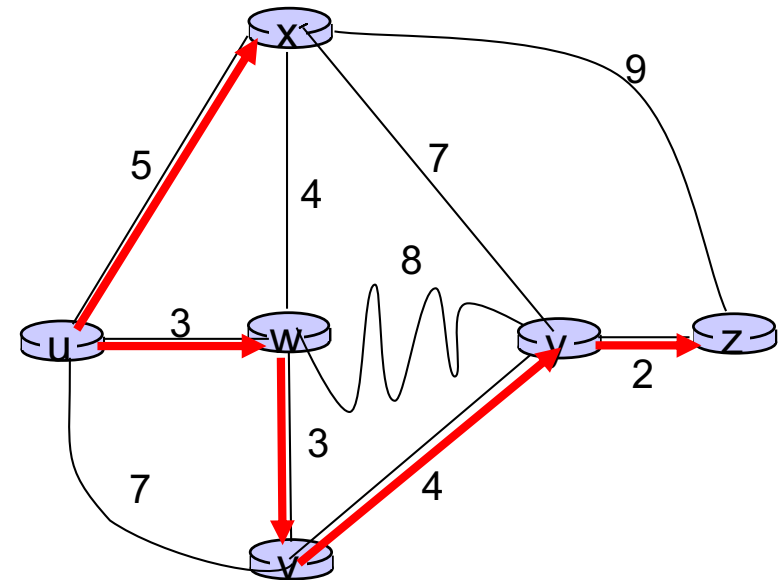
destination	outgoing link
v	(u,v)
x	(u,x)
y	(u,x)
w	(u,x)
x	(u,x)

route from u to v directly

route from u to all other destinations via x

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	7, u	3, u	5, u	∞	∞
1	uw	6, w		5, u	11, w	∞
2	uwx	6, w		u	11, w	14, x
3	uwx				10, v	14, x
4	uwxv					12, y
5	uwxvz					



notes:

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

Dijkstra's algorithm: discussion

algorithm complexity: n nodes

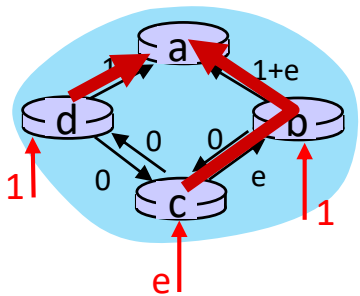
- each of n iteration: need to check all nodes, w , not in N
- $n(n+1)/2$ comparisons: $O(n^2)$ complexity
- more efficient implementations possible: $O(n \log n)$

message complexity:

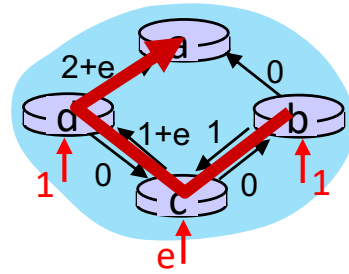
- each router must *broadcast* its link state information to other n routers
- efficient (and interesting!) broadcast algorithms: $O(n)$ link crossings to disseminate a broadcast message from one source
- each router's message crosses $O(n)$ links: overall message complexity: $O(n^2)$

Dijkstra's algorithm: oscillations possible

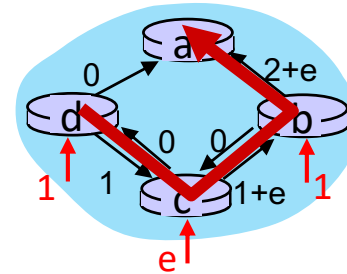
- when link costs depend on traffic volume, **route oscillations** possible
- sample scenario:
 - routing to destination a, traffic entering at d, c, e with rates 1, e (<1), 1
 - link costs are directional, and volume-dependent



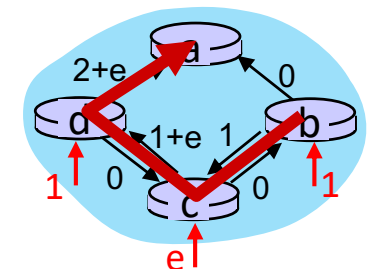
initially



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



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



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

Network layer: “control plane” roadmap

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



- network management, configuration
 - SNMP
 - NETCONF/YANG

Distance vector algorithm

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

Bellman-Ford equation

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

Then:

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

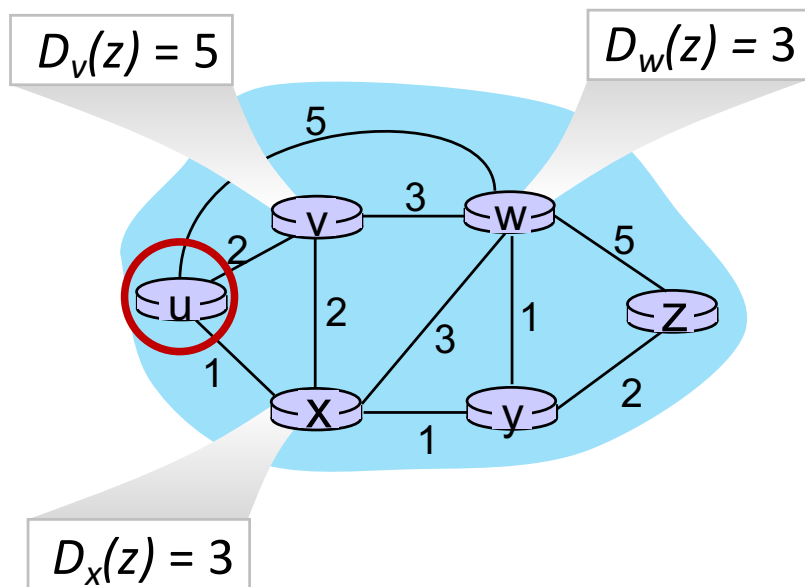
\min taken over all neighbors v of x

direct cost of link from x to v

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

Bellman-Ford Example

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



Bellman-Ford equation says:

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

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

Distance vector algorithm

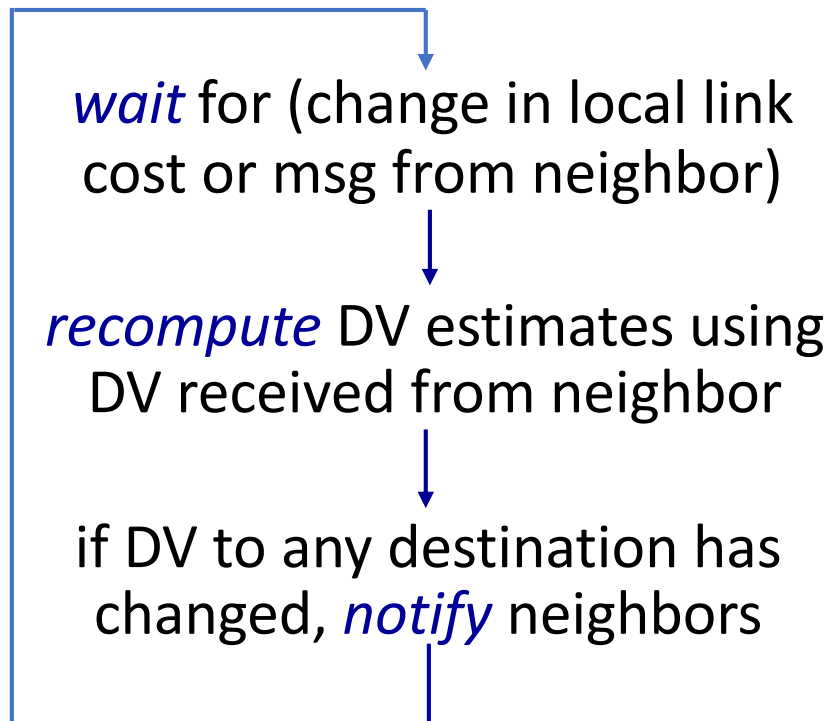
key idea:

- from time-to-time, each node sends its own distance vector estimate to neighbors
- under minor, natural conditions, the estimate $D_x(y)$ converge to the actual least cost $d_x(y)$
- when x receives new DV estimate from any neighbor, it updates its own DV using B-F equation:

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

Distance vector algorithm:

each node:



iterative, asynchronous: each local iteration caused by:

- local link cost change
- DV update message from neighbor

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

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

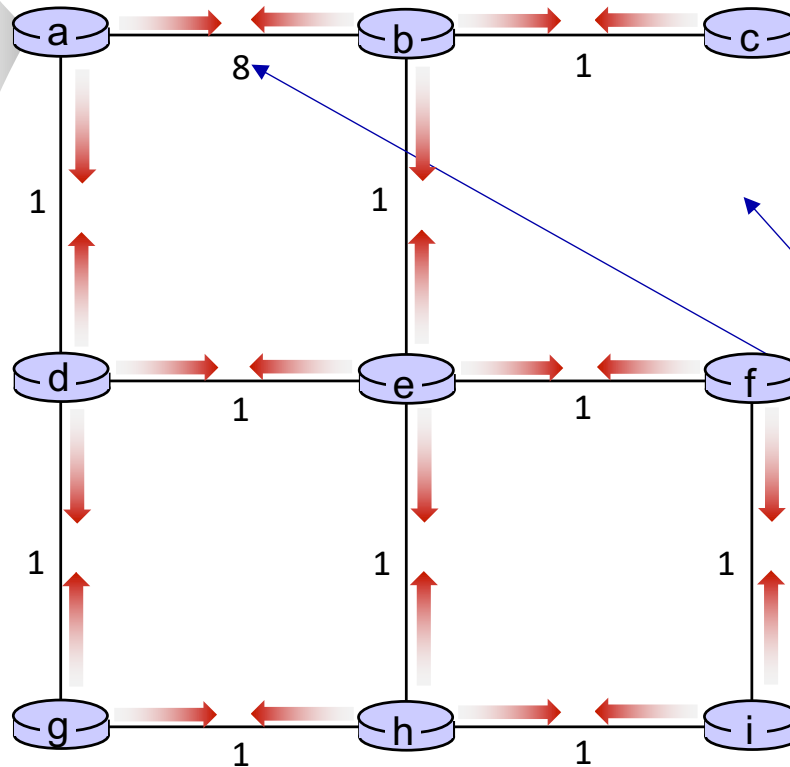
Distance vector: example



t=0

- All nodes have distance estimates to nearest neighbors (only)
- All nodes send their local distance vector to their neighbors

DV in a:
$D_a(a)=0$
$D_a(b)=8$
$D_a(c)=\infty$
$D_a(d)=1$
$D_a(e)=\infty$
$D_a(f)=\infty$
$D_a(g)=\infty$
$D_a(h)=\infty$
$D_a(i)=\infty$



- A few asymmetries:
- missing link
 - larger cost

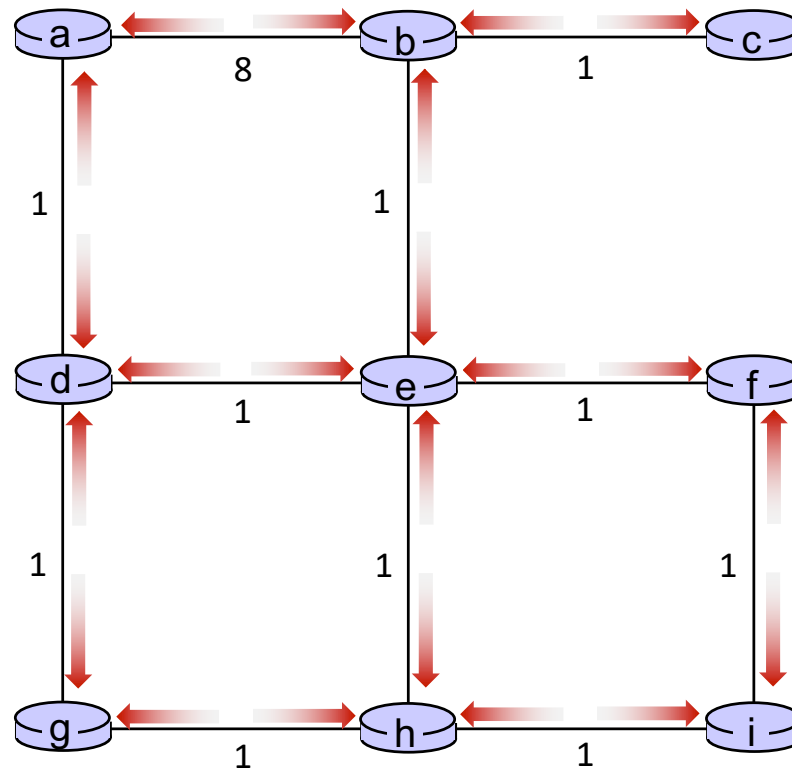
Distance vector example: iteration



t=1

All nodes:

- receive distance vectors from neighbors
- compute their new local distance vector
- send their new local distance vector to neighbors



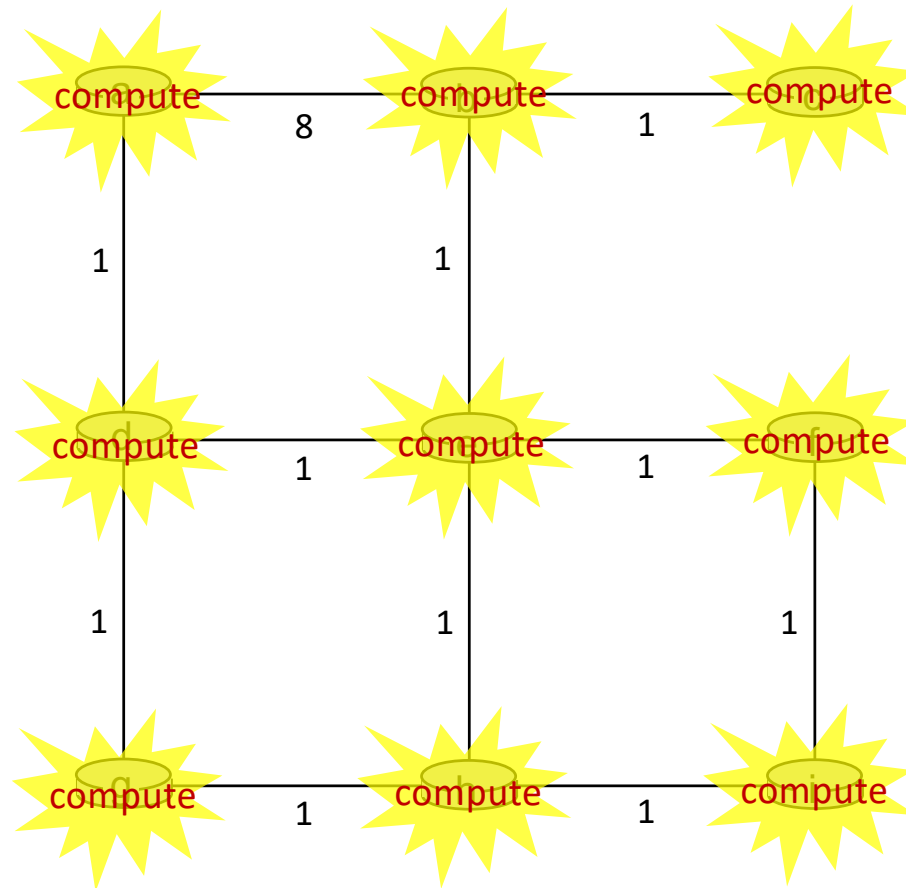
Distance vector example: iteration



t=1

All nodes:

- receive distance vectors from neighbors
- compute their new local distance vector
- send their new local distance vector to neighbors



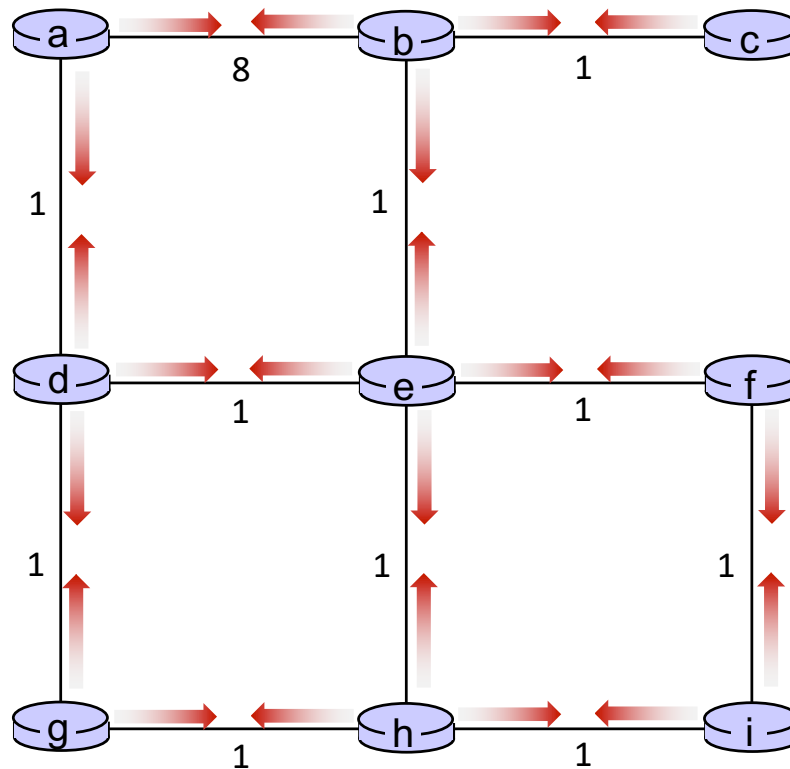
Distance vector example: iteration



t=1

All nodes:

- receive distance vectors from neighbors
- compute their new local distance vector
- send their new local distance vector to neighbors



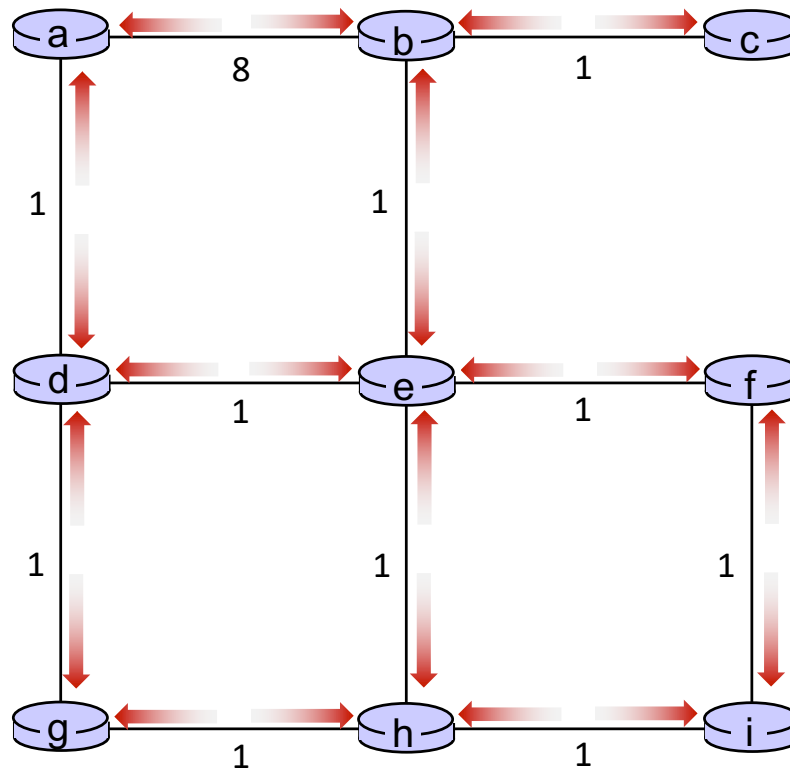
Distance vector example: iteration



t=2

All nodes:

- receive distance vectors from neighbors
- compute their new local distance vector
- send their new local distance vector to neighbors



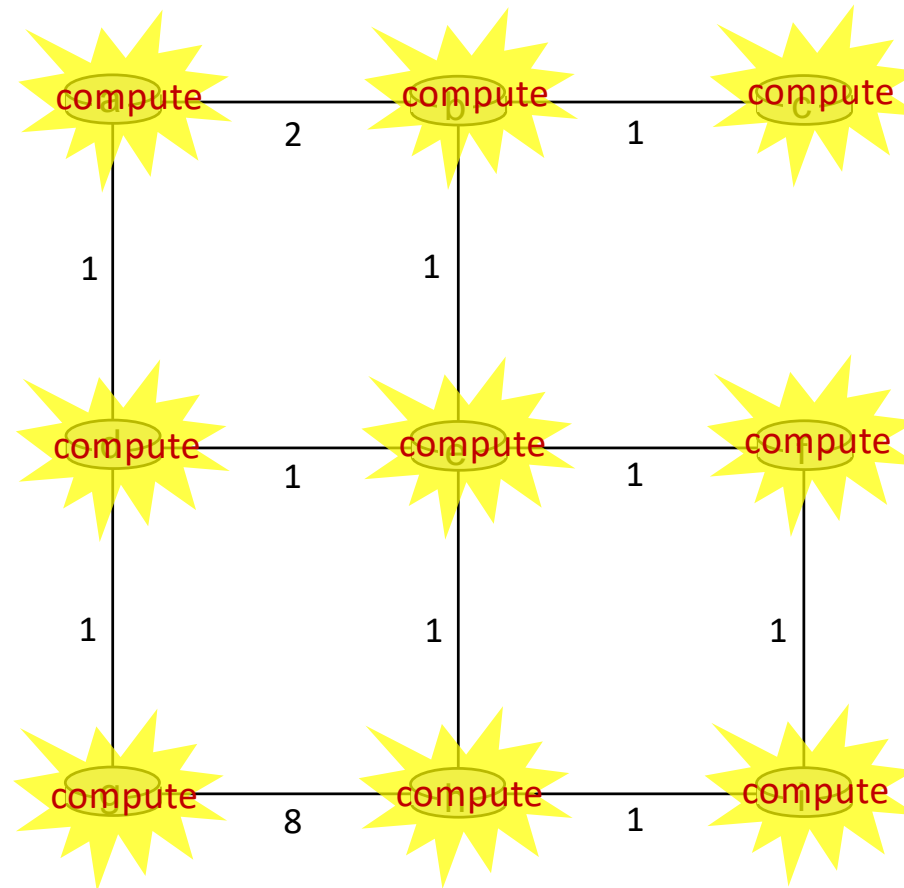
Distance vector example: iteration



t=2

All nodes:

- receive distance vectors from neighbors
- compute their new local distance vector
- send their new local distance vector to neighbors



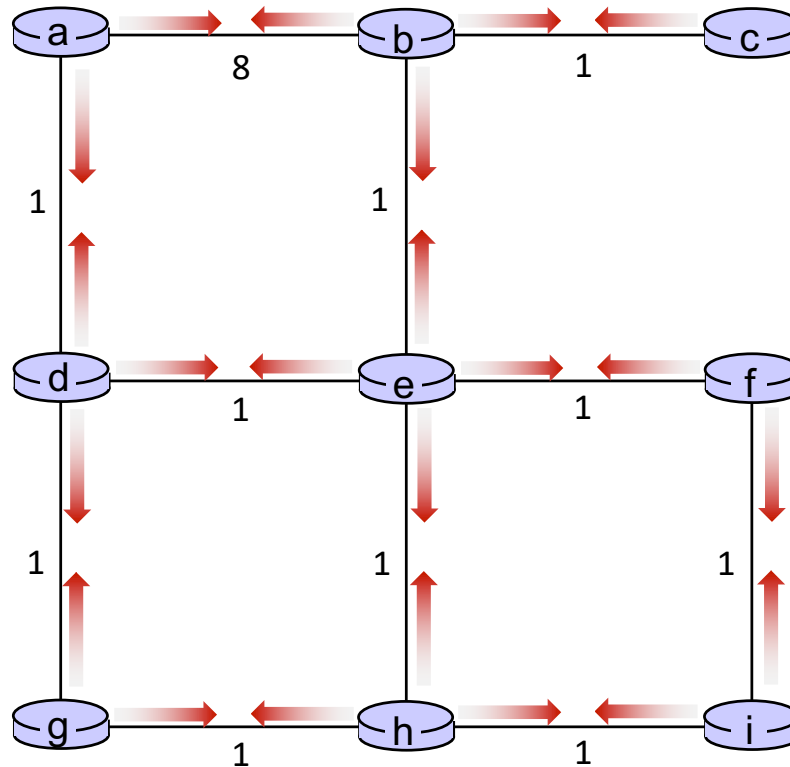
Distance vector example: iteration



t=2

All nodes:

- receive distance vectors from neighbors
- compute their new local distance vector
- send their new local distance vector to neighbors



Distance vector example: iteration

.... and so on

Let's next take a look at the iterative *computations* at nodes



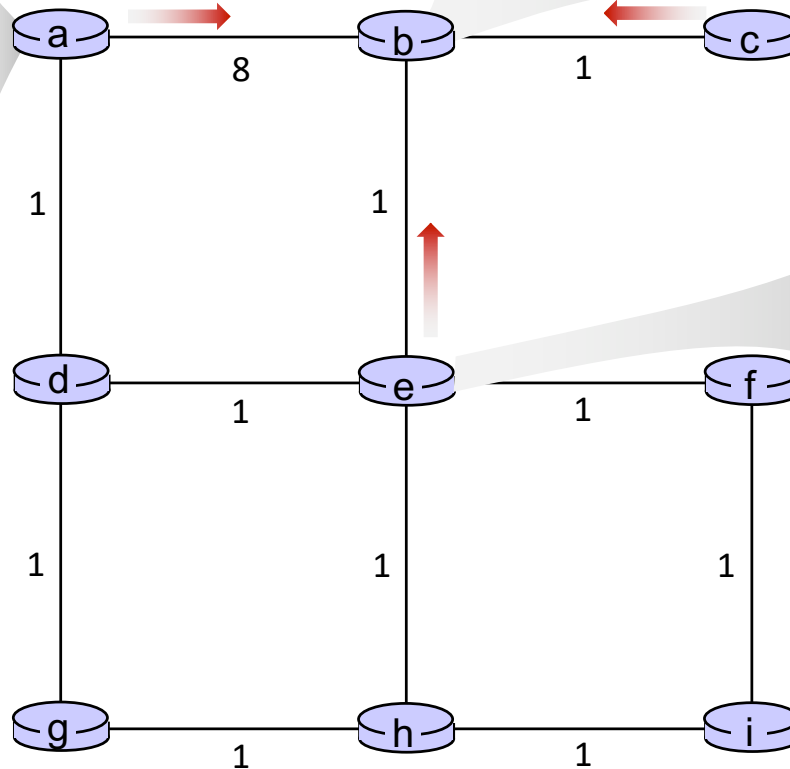
t=1

- b receives DVs from a, c, e

DV in a:
$D_a(a) = 0$
$D_a(b) = 8$
$D_a(c) = \infty$
$D_a(d) = 1$
$D_a(e) = \infty$
$D_a(f) = \infty$
$D_a(g) = \infty$
$D_a(h) = \infty$
$D_a(i) = \infty$

DV in b:	
$D_b(a) = 8$	$D_b(f) = \infty$
$D_b(c) = 1$	$D_b(g) = \infty$
$D_b(d) = \infty$	$D_b(h) = \infty$
$D_b(e) = 1$	$D_b(i) = \infty$

DV in c:
$D_c(a) = \infty$
$D_c(b) = 1$
$D_c(c) = 0$
$D_c(d) = \infty$
$D_c(e) = \infty$
$D_c(f) = \infty$
$D_c(g) = \infty$
$D_c(h) = \infty$
$D_c(i) = \infty$



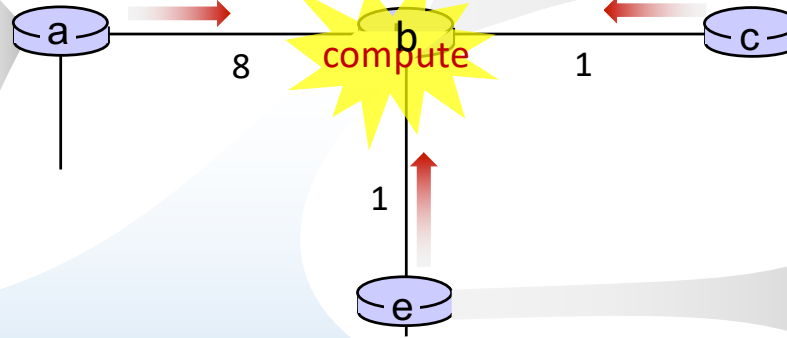
DV in e:
$D_e(a) = \infty$
$D_e(b) = 1$
$D_e(c) = \infty$
$D_e(d) = 1$
$D_e(e) = 0$
$D_e(f) = 1$
$D_e(g) = \infty$
$D_e(h) = 1$
$D_e(i) = \infty$



t=1

- b receives DVs from a, c, e, computes:

DV in a:	
$D_a(a) = 0$	
$D_a(b) = 8$	
$D_a(c) = \infty$	
$D_a(d) = 1$	
$D_a(e) = \infty$	
$D_a(f) = \infty$	
$D_a(g) = \infty$	
$D_a(h) = \infty$	
$D_a(i) = \infty$	



DV in b:	
$D_b(a) = 8$	$D_b(f) = \infty$
$D_b(c) = 1$	$D_b(g) = \infty$
$D_b(d) = \infty$	$D_b(h) = \infty$
$D_b(e) = 1$	$D_b(i) = \infty$

DV in c:	
$D_c(a) = \infty$	
$D_c(b) = 1$	
$D_c(c) = 0$	
$D_c(d) = \infty$	
$D_c(e) = \infty$	
$D_c(f) = \infty$	
$D_c(g) = \infty$	
$D_c(h) = \infty$	
$D_c(i) = \infty$	

DV in e:	
$D_e(a) = \infty$	
$D_e(b) = 1$	
$D_e(c) = \infty$	
$D_e(d) = 1$	
$D_e(e) = 0$	
$D_e(f) = 1$	
$D_e(g) = \infty$	
$D_e(h) = 1$	
$D_e(i) = \infty$	

$$D_b(a) = \min\{c_{b,a} + D_a(a), c_{b,c} + D_c(a), c_{b,e} + D_e(a)\} = \min\{8, \infty, \infty\} = 8$$

$$D_b(c) = \min\{c_{b,a} + D_a(c), c_{b,c} + D_c(c), c_{b,e} + D_e(c)\} = \min\{\infty, 1, \infty\} = 1$$

$$D_b(d) = \min\{c_{b,a} + D_a(d), c_{b,c} + D_c(d), c_{b,e} + D_e(d)\} = \min\{9, 2, \infty\} = 2$$

$$D_b(e) = \min\{c_{b,a} + D_a(e), c_{b,c} + D_c(e), c_{b,e} + D_e(e)\} = \min\{\infty, \infty, 1\} = 1$$

$$D_b(f) = \min\{c_{b,a} + D_a(f), c_{b,c} + D_c(f), c_{b,e} + D_e(f)\} = \min\{\infty, \infty, 2\} = 2$$

$$D_b(g) = \min\{c_{b,a} + D_a(g), c_{b,c} + D_c(g), c_{b,e} + D_e(g)\} = \min\{\infty, \infty, \infty\} = \infty$$

$$D_b(h) = \min\{c_{b,a} + D_a(h), c_{b,c} + D_c(h), c_{b,e} + D_e(h)\} = \min\{\infty, \infty, 2\} = 2$$

$$D_b(i) = \min\{c_{b,a} + D_a(i), c_{b,c} + D_c(i), c_{b,e} + D_e(i)\} = \min\{\infty, \infty, \infty\} = \infty$$

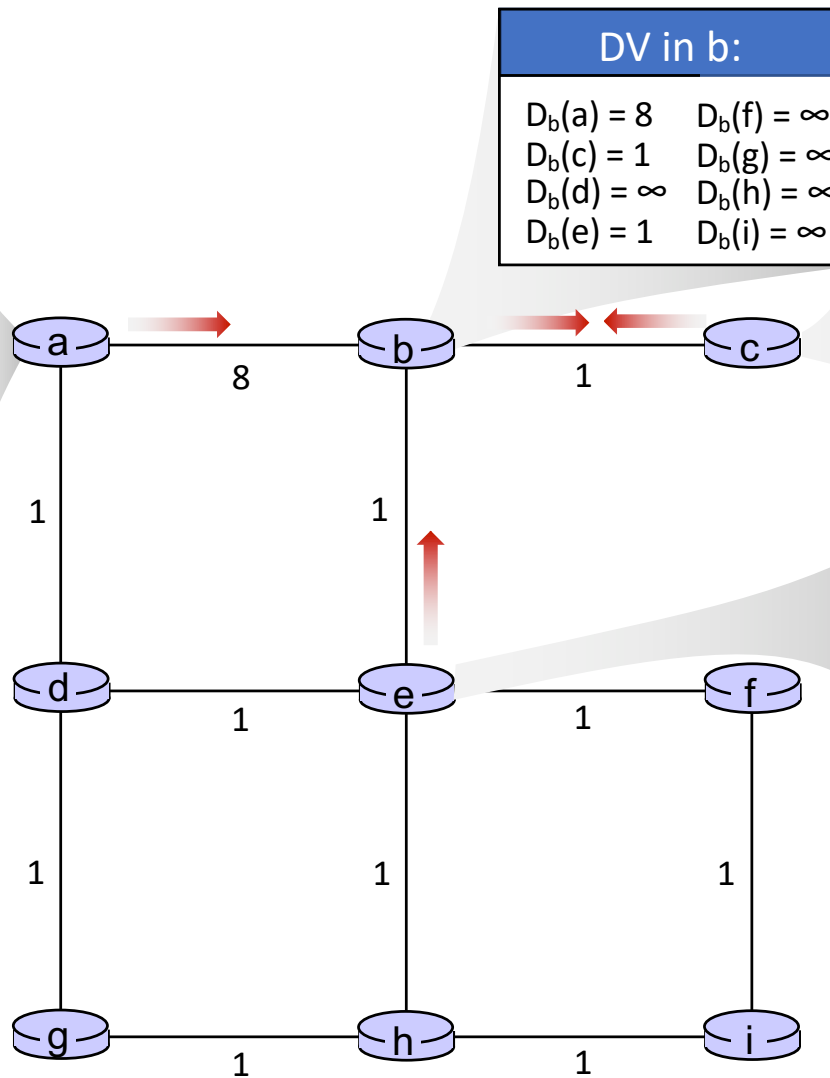
DV in b:	
$D_b(a) = 8$	$D_b(f) = 2$
$D_b(c) = 1$	$D_b(g) = \infty$
$D_b(d) = 2$	$D_b(h) = 2$
$D_b(e) = 1$	$D_b(i) = \infty$



t=1

- c receives DVs from b

DV in a:
$D_a(a) = 0$
$D_a(b) = 8$
$D_a(c) = \infty$
$D_a(d) = 1$
$D_a(e) = \infty$
$D_a(f) = \infty$
$D_a(g) = \infty$
$D_a(h) = \infty$
$D_a(i) = \infty$



DV in b:	
$D_b(a) = 8$	$D_b(f) = \infty$
$D_b(c) = 1$	$D_b(g) = \infty$
$D_b(d) = \infty$	$D_b(h) = \infty$
$D_b(e) = 1$	$D_b(i) = \infty$

DV in c:
$D_c(a) = \infty$
$D_c(b) = 1$
$D_c(c) = 0$
$D_c(d) = \infty$
$D_c(e) = \infty$
$D_c(f) = \infty$
$D_c(g) = \infty$
$D_c(h) = \infty$
$D_c(i) = \infty$

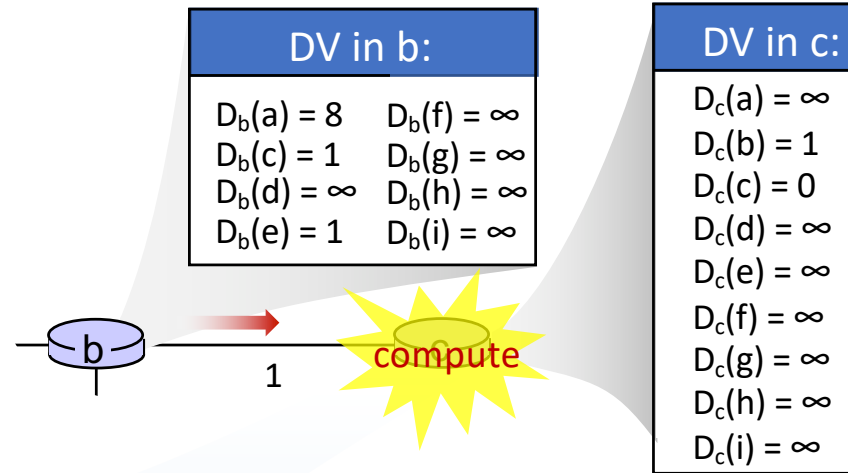
DV in e:
$D_e(a) = \infty$
$D_e(b) = 1$
$D_e(c) = \infty$
$D_e(d) = 1$
$D_e(e) = 0$
$D_e(f) = 1$
$D_e(g) = \infty$
$D_e(h) = 1$
$D_e(i) = \infty$



t=1

- c receives DVs from b computes:

$$\begin{aligned} D_c(a) &= \min\{c_{c,b} + D_b(a)\} = 1 + 8 = 9 \\ D_c(b) &= \min\{c_{c,b} + D_b(b)\} = 1 + 0 = 1 \\ D_c(d) &= \min\{c_{c,b} + D_b(d)\} = 1 + \infty = \infty \\ D_c(e) &= \min\{c_{c,b} + D_b(e)\} = 1 + 1 = 2 \\ D_c(f) &= \min\{c_{c,b} + D_b(f)\} = 1 + \infty = \infty \\ D_c(g) &= \min\{c_{c,b} + D_b(g)\} = 1 + \infty = \infty \\ D_c(h) &= \min\{c_{bc,b} + D_b(h)\} = 1 + \infty = \infty \\ D_c(i) &= \min\{c_{c,b} + D_b(i)\} = 1 + \infty = \infty \end{aligned}$$



DV in c:

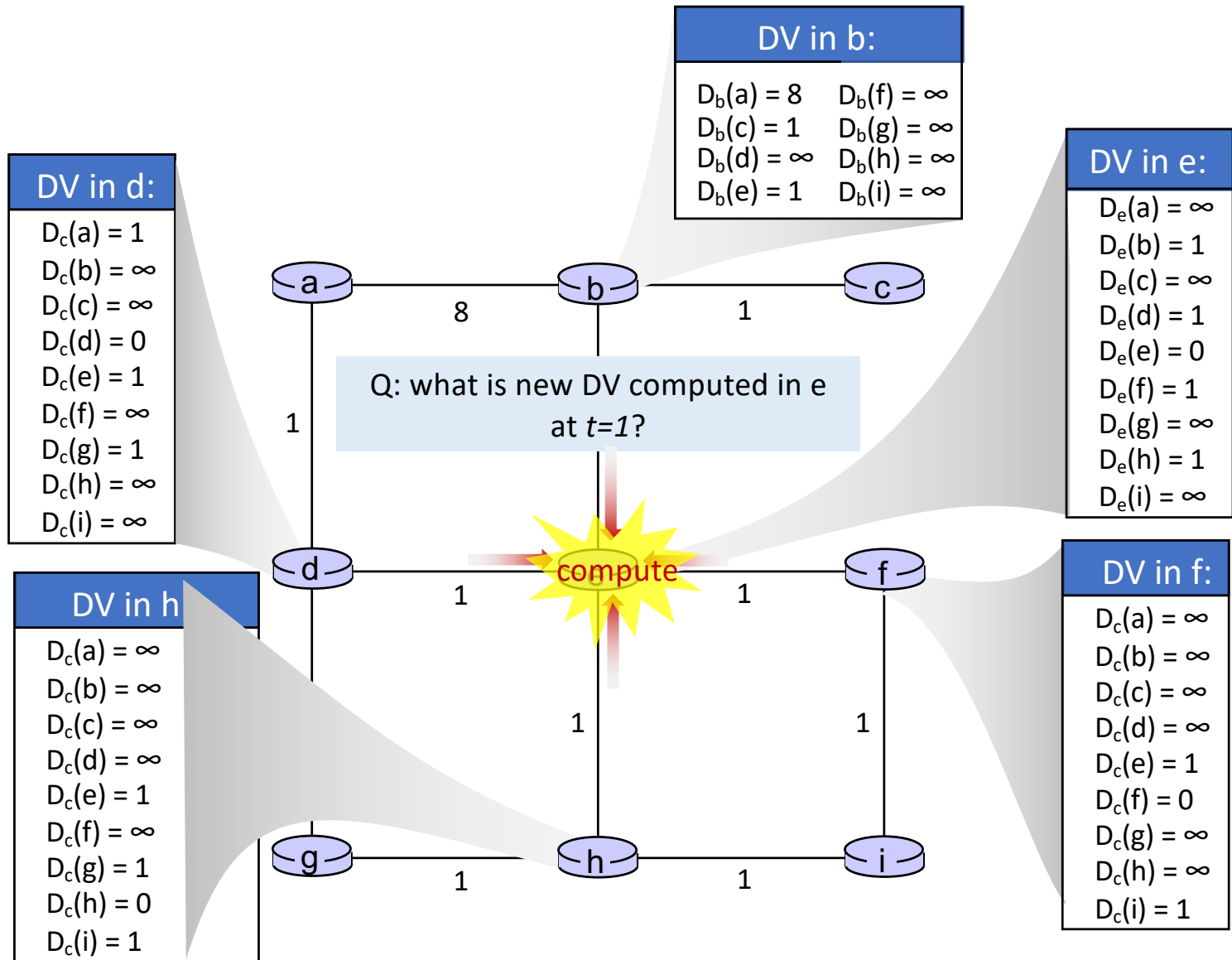
$D_c(a) = 9$
$D_c(b) = 1$
$D_c(c) = 0$
$D_c(d) = 2$
$D_c(e) = \infty$
$D_c(f) = \infty$
$D_c(g) = \infty$
$D_c(h) = \infty$
$D_c(i) = \infty$

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








t=1

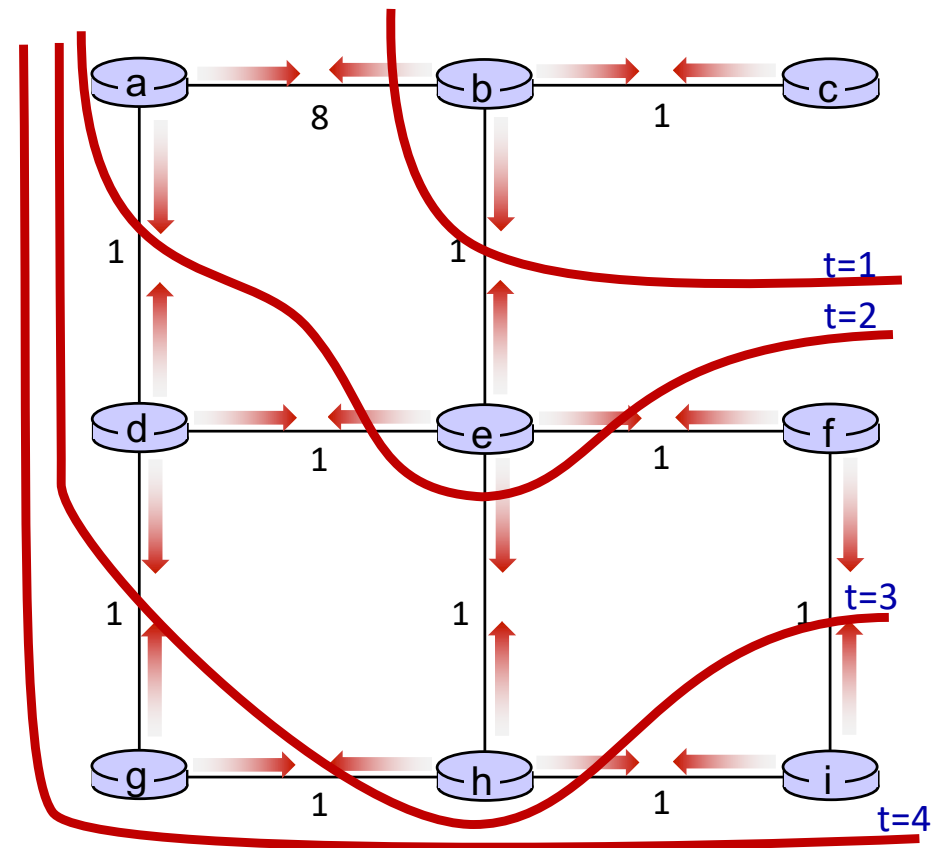
- e receives DVs from b, d, f, h



Distance vector: state information diffusion

Iterative communication, computation steps diffuses information through network:

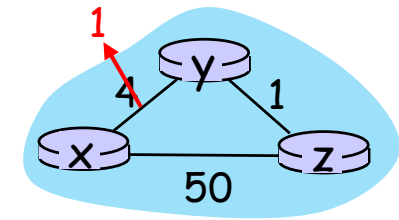
-  t=0 c's state at t=0 is at c only
-  t=1 c's state at t=0 has propagated to b, and may influence distance vector computations up to **1** hop away, i.e., at b
-  t=2 c's state at t=0 may now influence distance vector computations up to **2** hops away, i.e., at b and now at a, e as well
-  t=3 c's state at t=0 may influence distance vector computations up to **3** hops away, i.e., at b,a,e and now at c,f,h as well
-  t=4 c's state at t=0 may influence distance vector computations up to **4** hops away, i.e., at b,a,e, c, f, h and now at g,i as well



Distance vector: link cost changes

link cost changes:

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



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

“good news
travels fast”

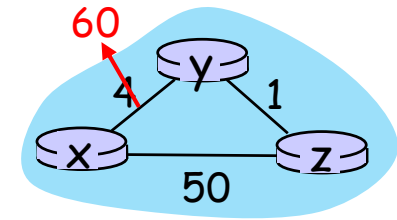
t_1 : z receives update from y, updates its table, computes new least cost to x, sends its neighbors its DV.

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

Distance vector: link cost changes

link cost changes:

- node detects local link cost change
- “bad news travels slow” – count-to-infinity problem:
 - y sees direct link to x has new cost 60, but z has said it has a path at cost of 5. So y computes “my new cost to x will be 6, via z); notifies z of new cost of 6 to x.
 - z learns that path to x via y has new cost 6, so z computes “my new cost to x will be 7 via y), notifies y of new cost of 7 to x.
 - y learns that path to x via z has new cost 7, so y computes “my new cost to x will be 8 via y), notifies z of new cost of 8 to x.
 - z learns that path to x via y has new cost 8, so z computes “my new cost to x will be 9 via y), notifies y of new cost of 9 to x.
 - ...



Comparison of LS and DV algorithms

message complexity

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

DV: exchange between neighbors;
convergence time varies

speed of convergence

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

- may have oscillations

DV: convergence time varies

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

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

LS:

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

DV:

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

Network layer: “control plane” roadmap

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



- network management, configuration
 - SNMP
 - NETCONF/YANG

Making routing scalable

our routing study thus far - idealized

all routers identical

network “flat”

... not true in practice

scale: billions of destinations:

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

administrative autonomy:

- Internet: a network of networks
- each network admin may want to control routing in its own network

Internet approach to scalable routing

aggregate routers into regions known as “autonomous systems” (AS) (a.k.a. “domains”)

intra-AS (aka “intra-domain”):

routing among *within same AS* (“network”)

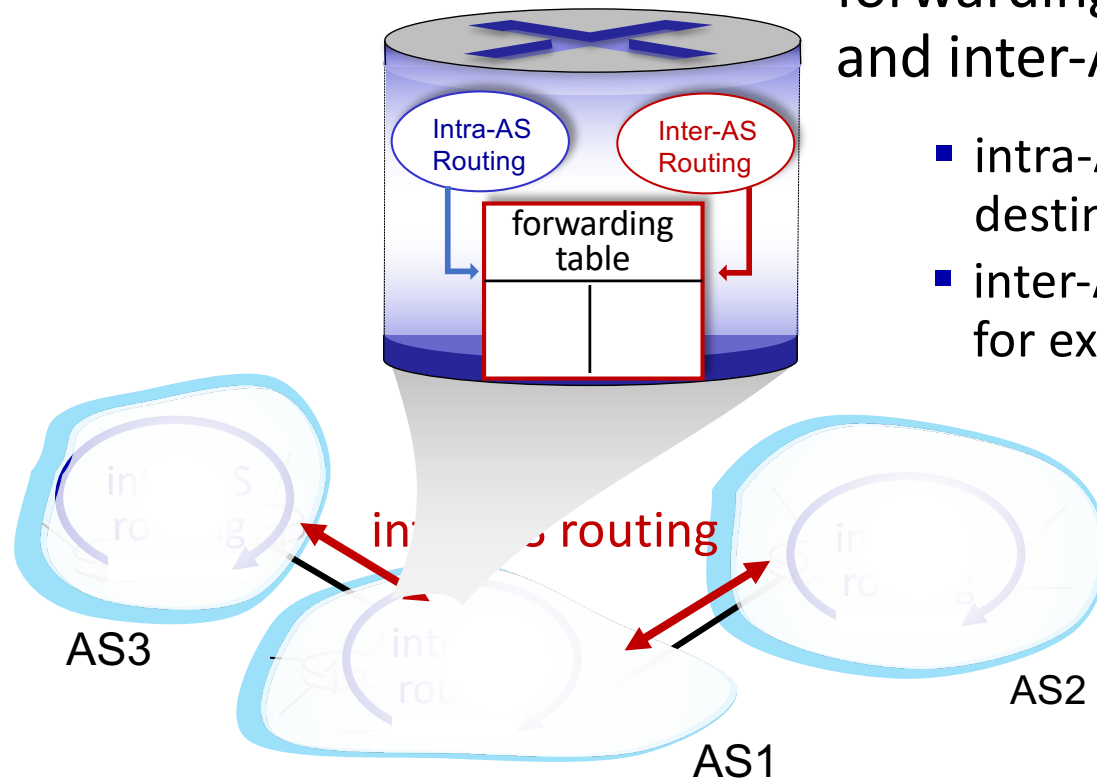
- all routers in AS must run same intra-domain protocol
- routers in different AS can run different intra-domain routing protocols
- **gateway router**: at “edge” of its own AS, has link(s) to router(s) in other AS'es

inter-AS (aka “inter-domain”):

routing *among* AS'es

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

Interconnected ASes



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

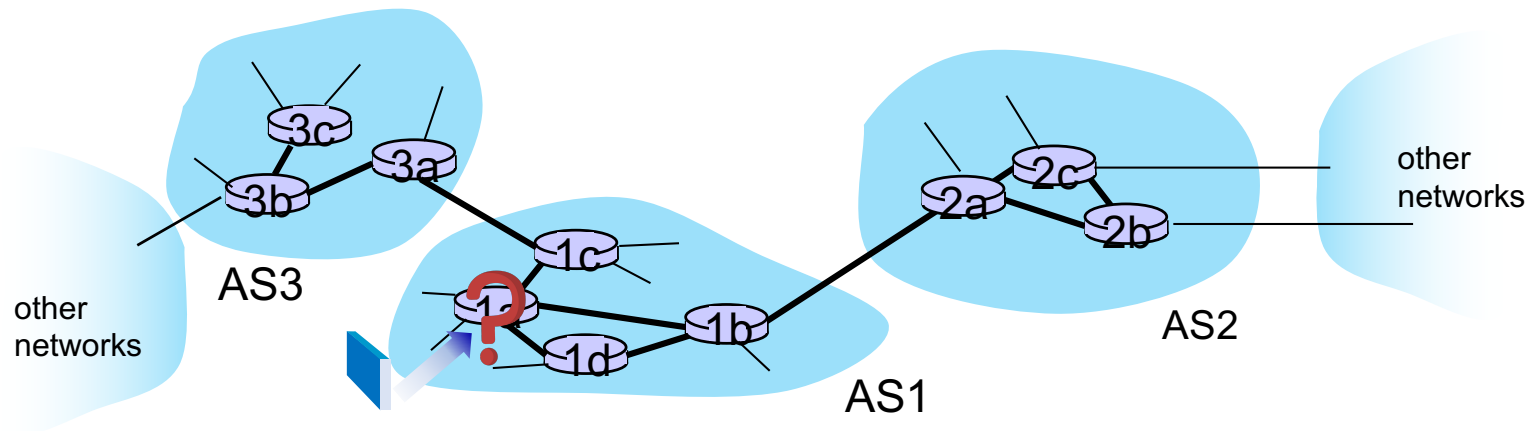
- intra-AS routing determine entries for destinations within AS
- inter-AS & intra-AS determine entries for external destinations

Inter-AS routing: a role in intradomain forwarding

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

AS1 inter-domain routing must:

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



Inter-AS routing: routing within an AS

most common intra-AS routing protocols:

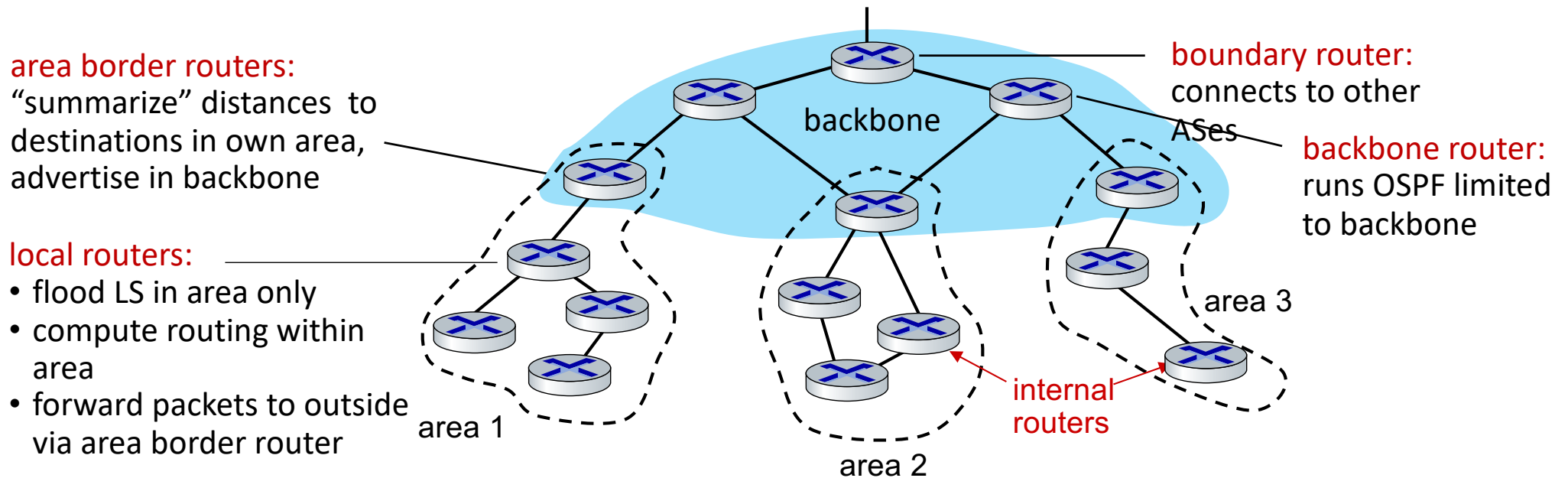
- **RIP: Routing Information Protocol** [RFC 1723]
 - classic DV: DVs exchanged every 30 secs
 - no longer widely used
- **EIGRP: Enhanced Interior Gateway Routing Protocol**
 - DV based
 - formerly Cisco-proprietary for decades (became open in 2013 [RFC 7868])
- **OSPF: Open Shortest Path First** [RFC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

OSPF (Open Shortest Path First) routing

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

Hierarchical OSPF

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



Network layer: “control plane” roadmap

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



- network management, configuration
 - SNMP
 - NETCONF/YANG

Internet inter-AS routing: BGP

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

Network layer: “control plane” roadmap

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



- network management, configuration
 - SNMP
 - NETCONF/YANG

Software defined networking (SDN)

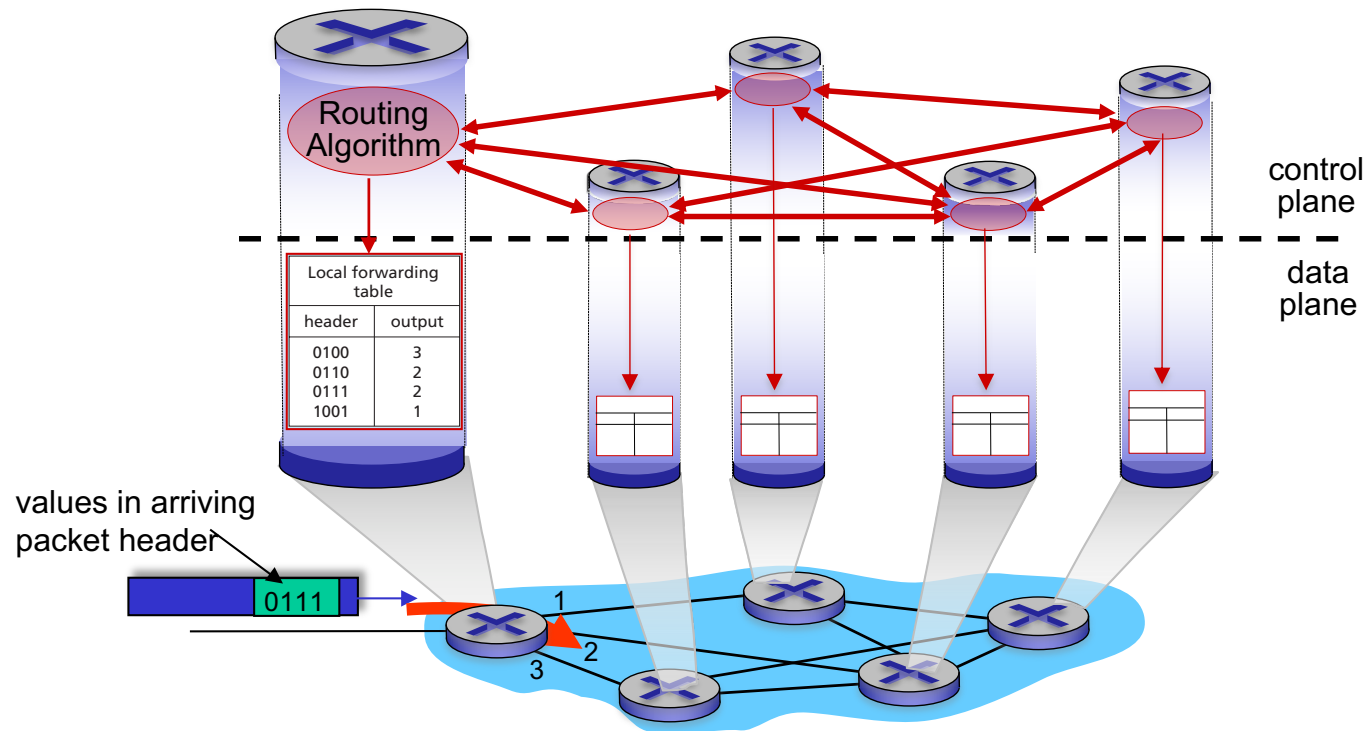
Internet network layer: historically implemented via distributed, per-router control approach:

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

~2005: renewed interest in rethinking network control plane

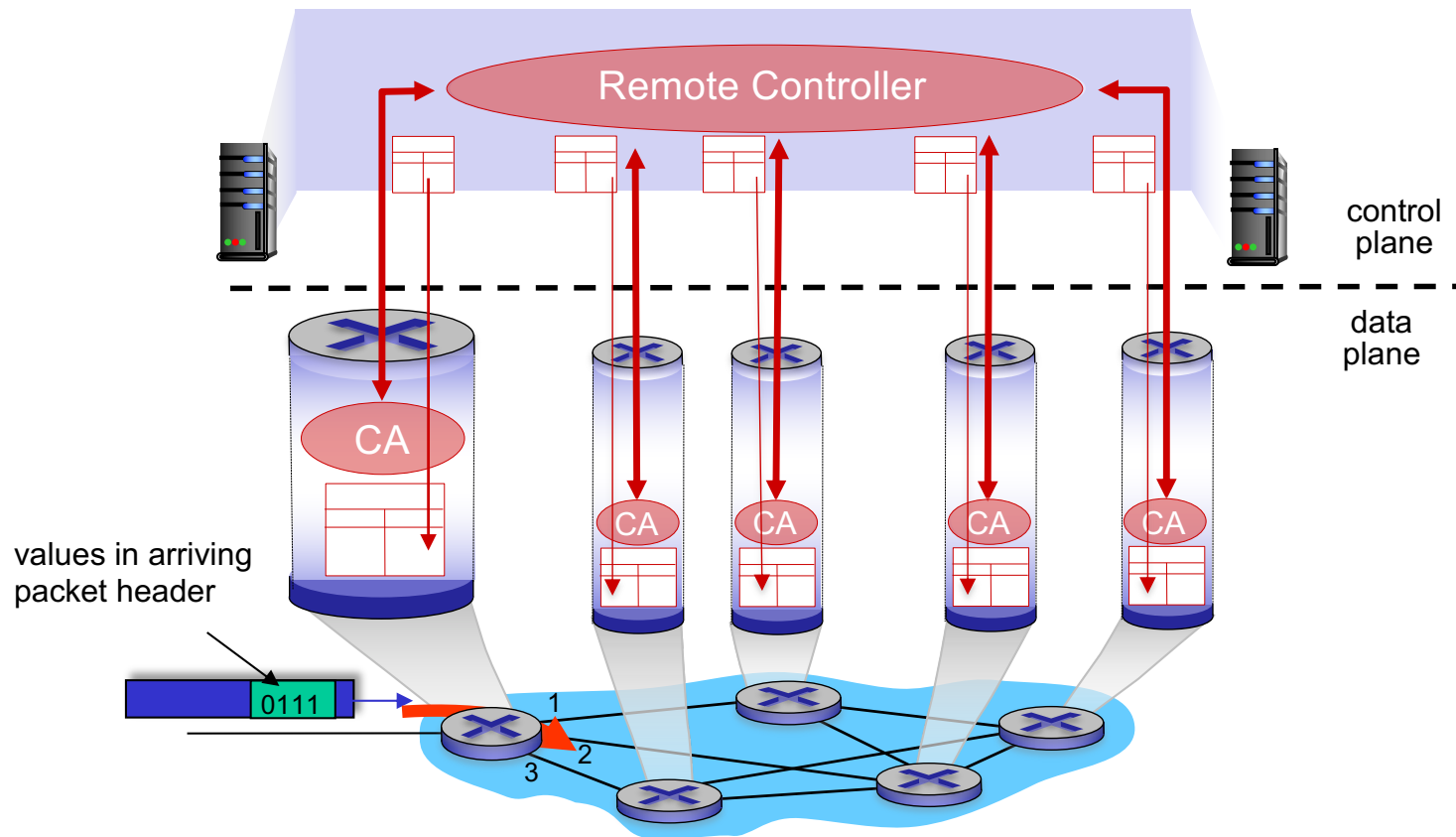
Per-router control plane

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



Software-Defined Networking (SDN) control plane

Remote controller computes, installs forwarding tables in routers



Software defined networking (SDN)

Why a *logically centralized* control plane?

easier network management: avoid router misconfigurations, greater flexibility of traffic flows

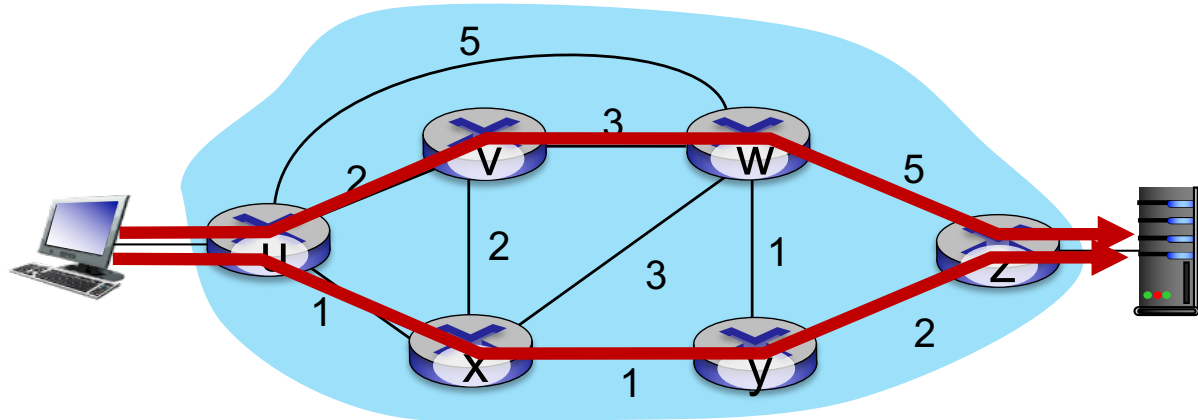
table-based forwarding (recall OpenFlow API) allows “programming” routers

- centralized “programming” easier: compute tables centrally and distribute
- distributed “programming” more difficult: compute tables as result of distributed algorithm (protocol) implemented in each-and-every router

open (non-proprietary) implementation of control plane

- foster innovation: let 1000 flowers bloom

Traffic engineering: difficult with traditional routing

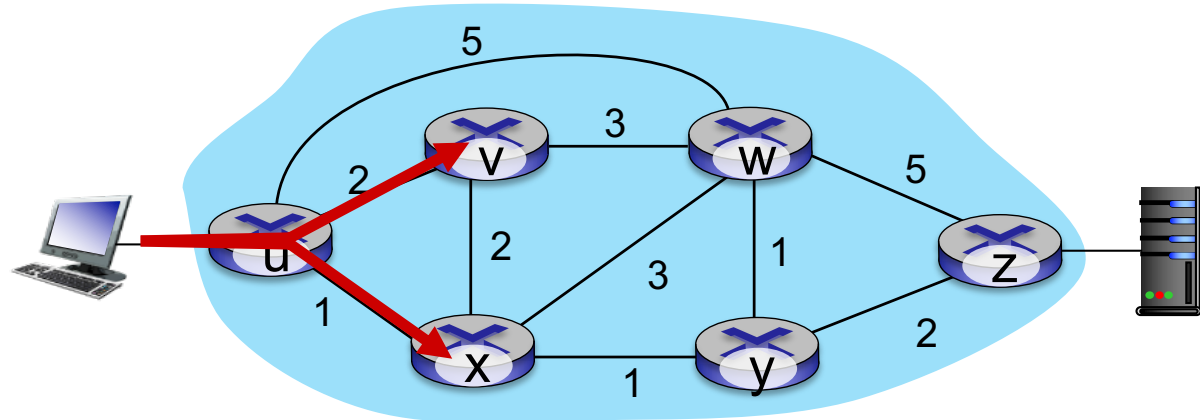


Q: what if network operator wants u-to-z traffic to flow along $uvwz$, rather than $uxyz$?

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

link weights are only control “knobs”: not much control!

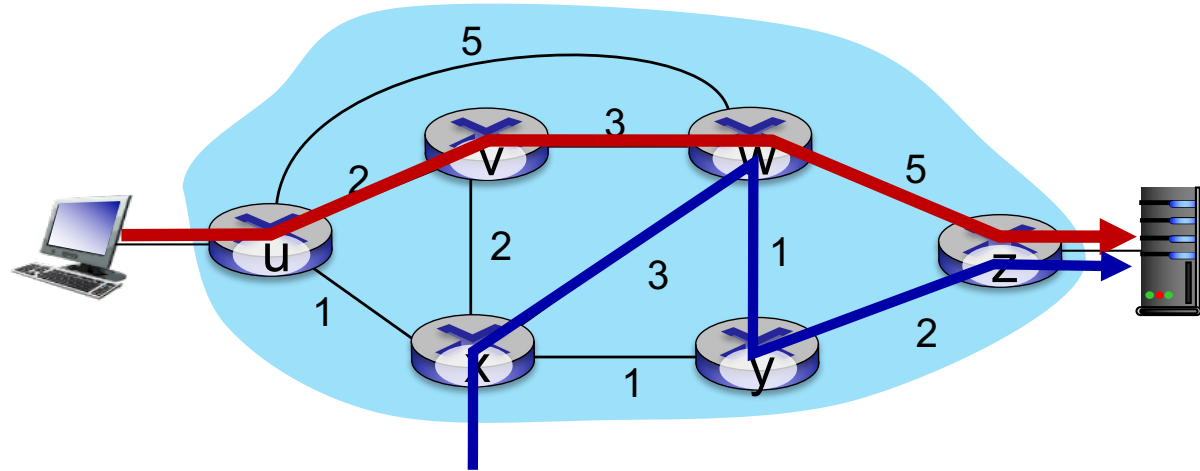
Traffic engineering: difficult with traditional routing



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

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

Traffic engineering: difficult with traditional routing

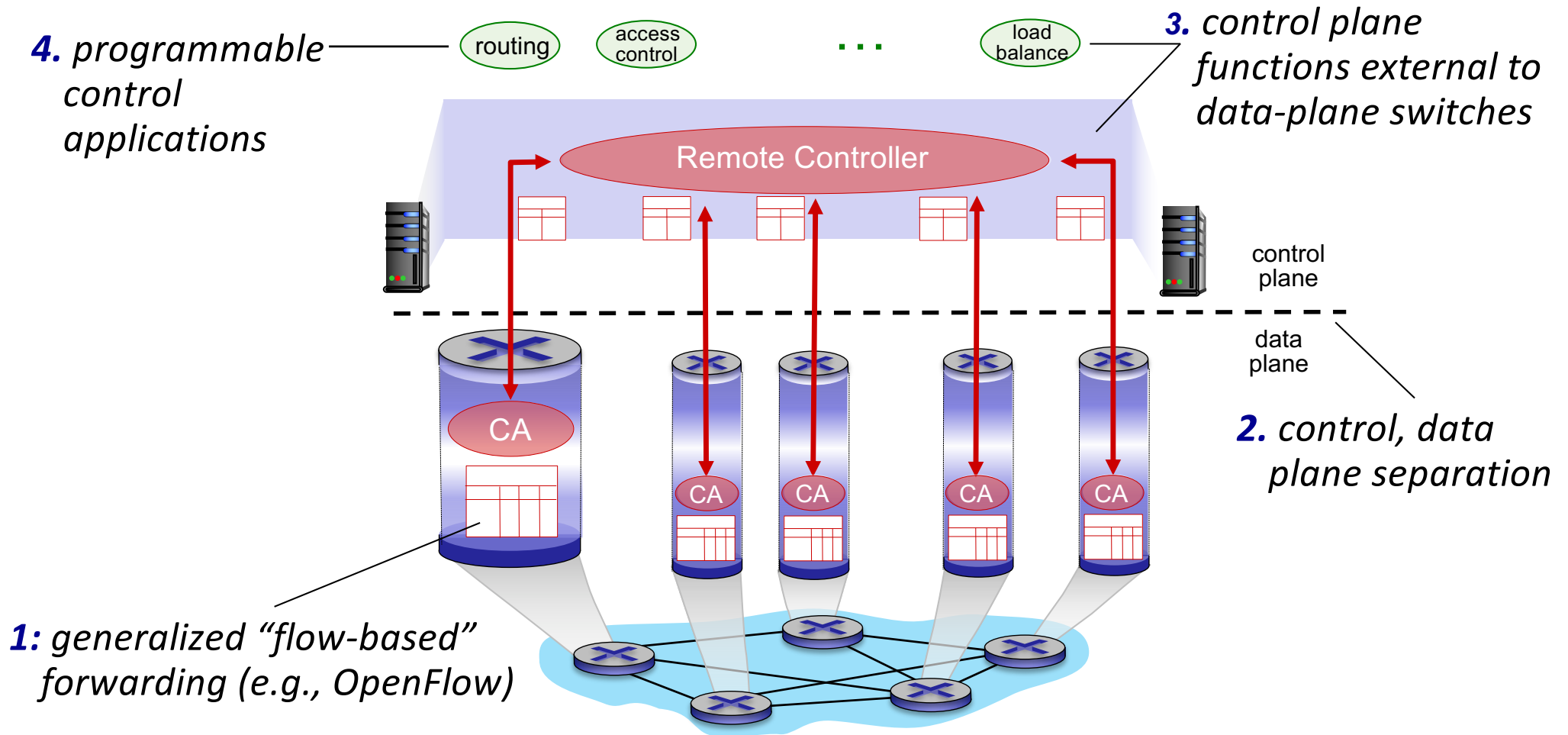


Q: what if w wants to route blue and red traffic differently from w to z?

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

We learned that generalized forwarding and SDN can be used to achieve *any* routing desired

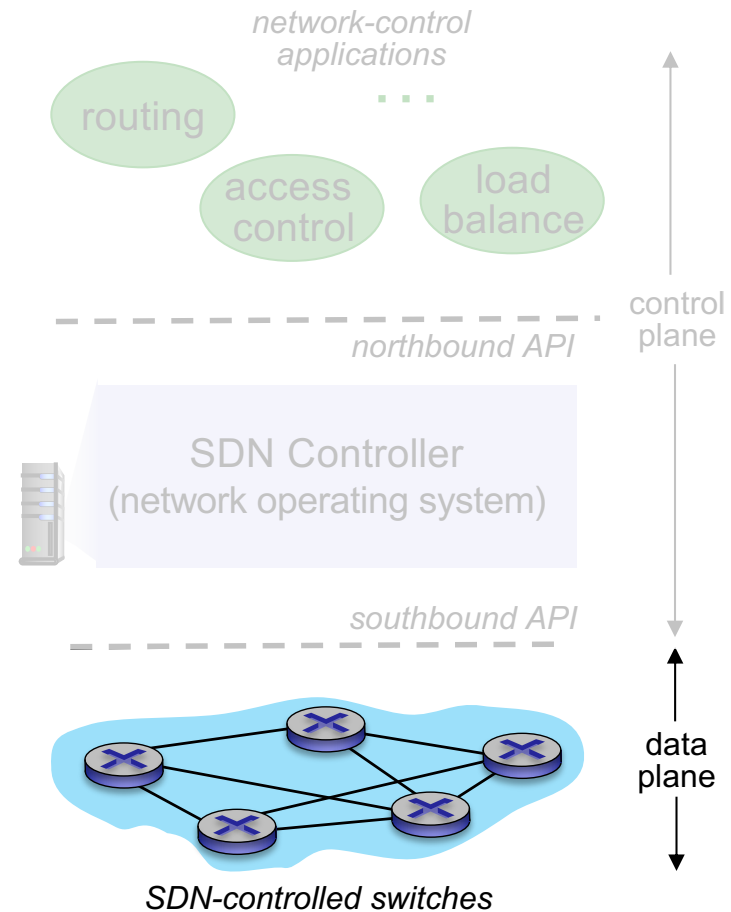
Software defined networking (SDN)



Software defined networking (SDN)

Data-plane switches:

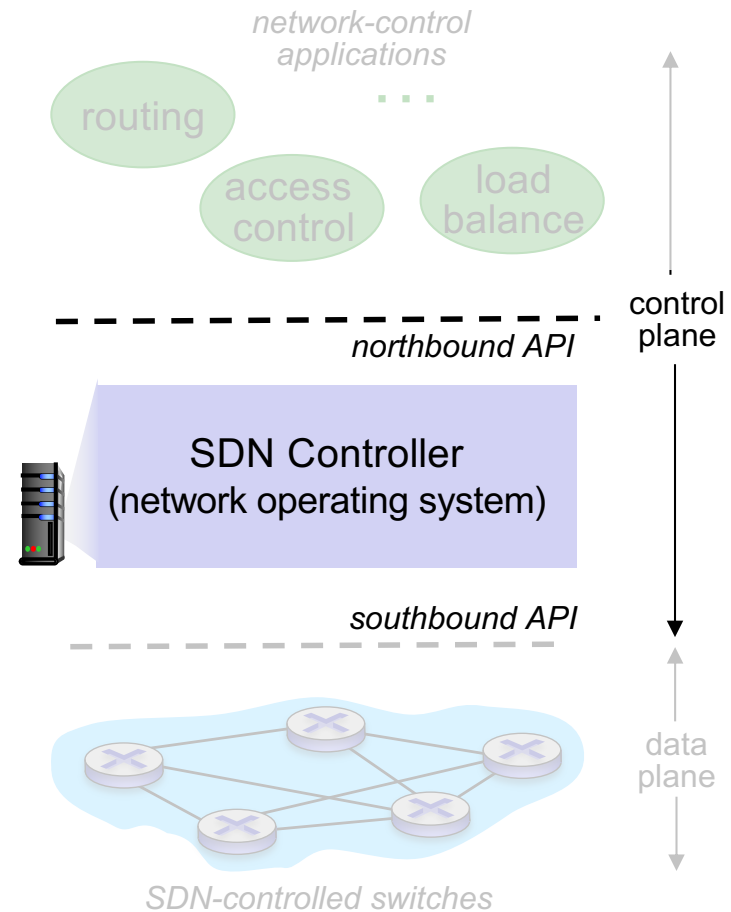
- fast, simple, commodity switches implementing generalized data-plane forwarding (Section 4.4) in hardware
- flow (forwarding) table computed, installed under controller supervision
- API for table-based switch control (e.g., OpenFlow)
 - defines what is controllable, what is not
- protocol for communicating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



Software defined networking (SDN)

SDN controller (network OS):

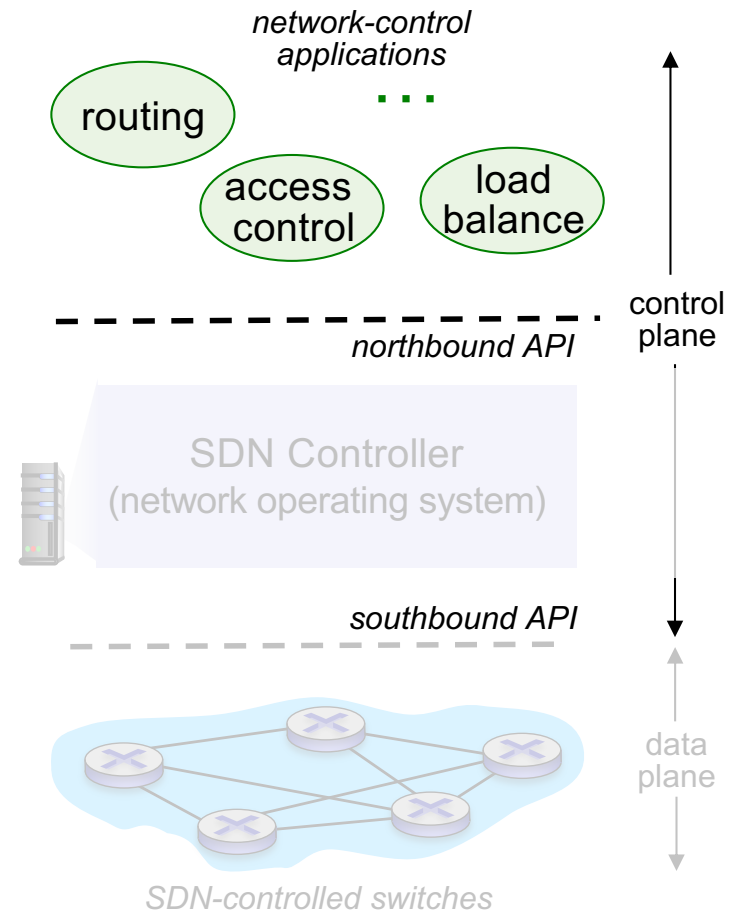
- maintain network state information
- interacts with network control applications “above” via northbound API
- interacts with network switches “below” via southbound API
- implemented as distributed system for performance, scalability, fault-tolerance, robustness

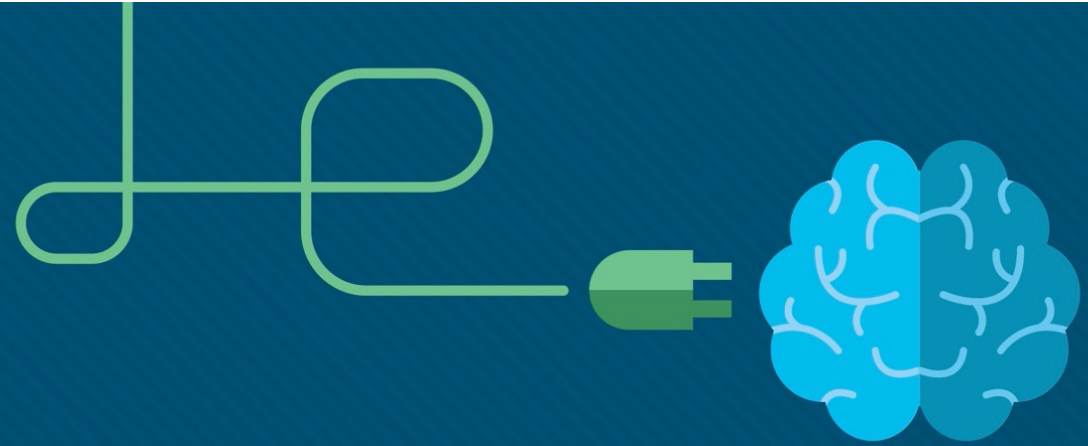


Software defined networking (SDN)

network-control apps:

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





Single-Area OSPFv2 Concepts

Enterprise Networking, Security, and Automation v7.0
(ENSA)



OSPF Features and Characteristics

Components of OSPF (Cont.)

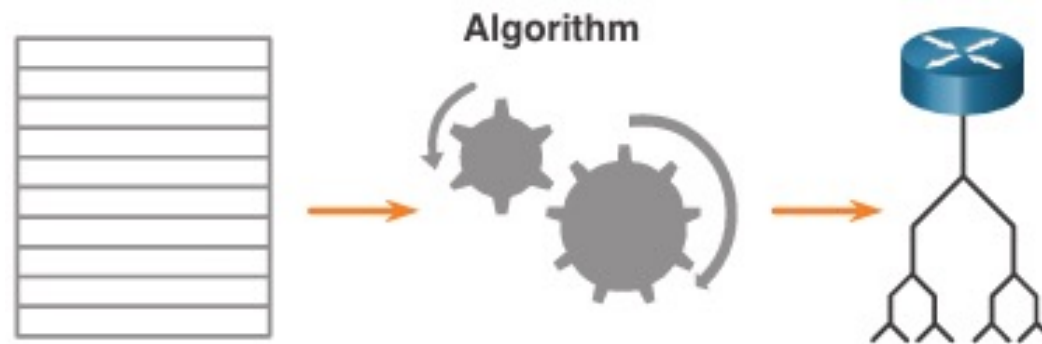
OSPF messages are used to create and maintain three OSPF databases, as follows:

Database	Table	Description
Adjacency Database	Neighbor Table	<ul style="list-style-type: none">•List of all neighbor routers to which a router has established bi-directional communication.•This table is unique for each router.•Can be viewed using the show ip ospf neighbor command.
Link-state Database (LSDB)	Topology Table	<ul style="list-style-type: none">•Lists information about all other routers in the network.•The database represents the network LSDB.•All routers within an area have identical LSDB.•Can be viewed using the show ip ospf database command.
Forwarding Database	Routing Table	<ul style="list-style-type: none">•List of routes generated when an algorithm is run on the link-state database.•Each router's routing table is unique and contains information on how and where to send packets to other routers.•Can be viewed using the show ip route command.

OSPF Features and Characteristics

Components of OSPF (Cont.)

- The router builds the topology table using results of calculations based on the Dijkstra shortest-path first (SPF) algorithm. The SPF algorithm is based on the cumulative cost to reach a destination.
- The SPF algorithm creates an SPF tree by placing each router at the root of the tree and calculating the shortest path to each node. The SPF tree is then used to calculate the best routes. OSPF places the best routes into the forwarding database, which is used to make the routing table.



OSPF Features and Characteristics

Link-State Operation

To maintain routing information, OSPF routers complete a generic link-state routing process to reach a state of convergence. The following are the link-state routing steps that are completed by a router:

1. Establish Neighbor Adjacencies
2. Exchange Link-State Advertisements
3. Build the Link State Database
4. Execute the SPF Algorithm
5. Choose the Best Route

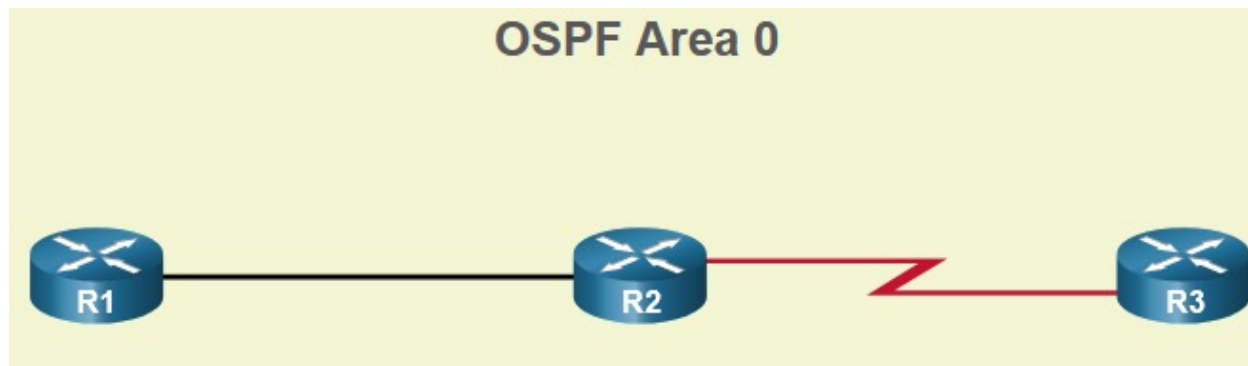
OSPF Features and Characteristics

Single-Area and Multiarea OSPF

To make OSPF more efficient and scalable, OSPF supports hierarchical routing using areas. An OSPF area is a group of routers that share the same link-state information in their LSDBs. OSPF can be implemented in one of two ways, as follows:

- **Single-Area OSPF** - All routers are in one area. Best practice is to use area 0.
- **Multiarea OSPF** - OSPF is implemented using multiple areas, in a hierarchical fashion. All areas must connect to the backbone area (area 0). Routers interconnecting the areas are referred to as Area Border Routers (ABRs).

The focus of this module is on single-area OSPFv2.



OSPF Features and Characteristics

OSPFv3

- OSPFv3 is the OSPFv2 equivalent for exchanging IPv6 prefixes. OSPFv3 exchanges routing information to populate the IPv6 routing table with remote prefixes.
- **Note:** With the OSPFv3 Address Families feature, OSPFv3 includes support for both IPv4 and IPv6. OSPF Address Families is beyond the scope of this curriculum.
- OSPFv3 has the same functionality as OSPFv2, but uses IPv6 as the network layer transport, communicating with OSPFv3 peers and advertising IPv6 routes. OSPFv3 also uses the SPF algorithm as the computation engine to determine the best paths throughout the routing domain.
- OSPFv3 has separate processes from its IPv4 counterpart. The processes and operations are basically the same as in the IPv4 routing protocol, but run independently.

OSPF Packets

Types of OSPF Packets

The table summarizes the five different types of Link State Packets (LSPs) used by OSPFv2. OSPFv3 has similar packet types.

Type	Packet Name	Description
1	Hello	Discovers neighbors and builds adjacencies between them
2	Database Description (DBD)	Checks for database synchronization between routers
3	Link-State Request (LSR)	Requests specific link-state records from router to router
4	Link-State Update (LSU)	Sends specifically requested link-state records
5	Link-State Acknowledgment (LSAck)	Acknowledges the other packet types

OSPF Packets

Link-State Updates

- LSUs are also used to forward OSPF routing updates. An LSU packet can contain 11 different types of OSPFv2 LSAs. OSPFv3 renamed several of these LSAs and also contains two additional LSAs.
- LSU and LSA are often used interchangeably, but the correct hierarchy is LSU packets contain LSA messages.

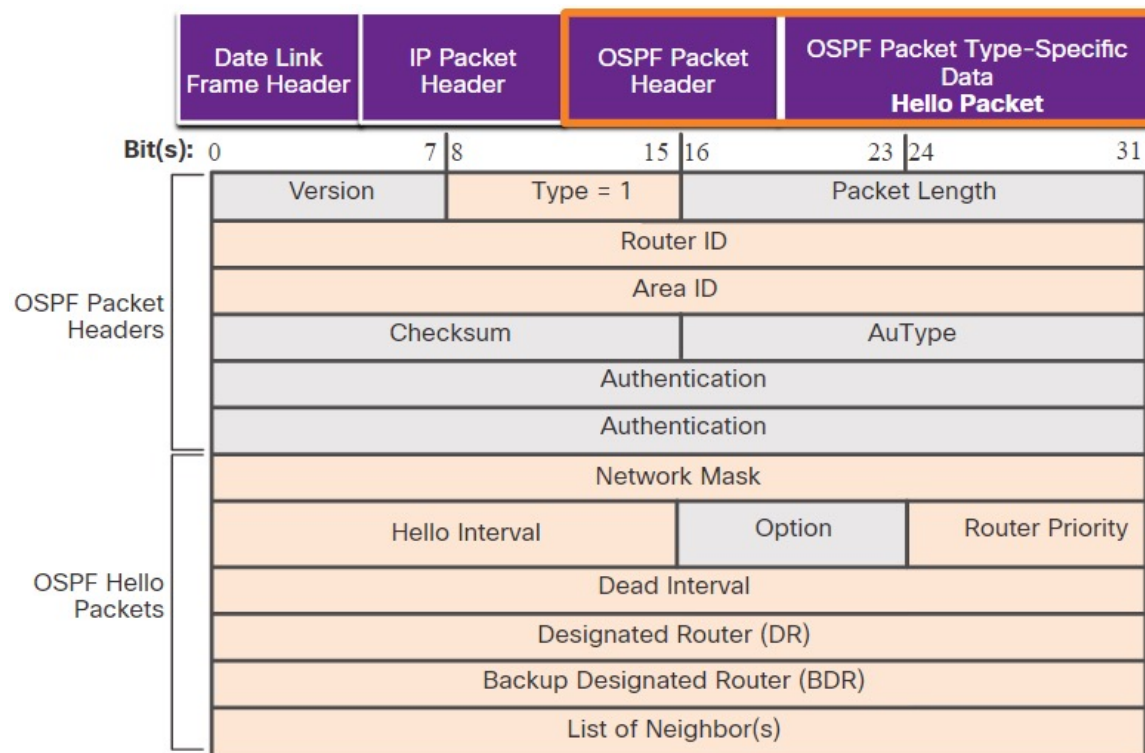
LSUs		
Type	Packet Name	Description
1	Hello	Discovers neighbors and builds adjacencies between them
2	DBD	Checks for database synchronization between routers
3	LSR	Requests specific link-state records from router to router
4	LSU	Sends specifically requested link-state records
5	LSAck	Acknowledges the other packet types

LSAs	
LSA Type	Description
1	Router LSAs
2	Checks for database synchronization between routers
3 or 4	Summary LSAs
5	Autonomous System External LSAs
6	Multicast OSPF LSAs
7	Defined for Not-So-Stubby Areas
8	External Attributes LSA for Border Gateway Patrol (BGPs)

OSPF Packets Hello Packet

The OSPF Type 1 packet is the Hello packet. Hello packets are used to do the following:

- Discover OSPF neighbors and establish neighbor adjacencies.
- Advertise parameters on which two routers must agree to become neighbors.
- Elect the Designated Router (DR) and Backup Designated Router (BDR) on multiaccess networks like Ethernet. Point-to-point links do not require DR or BDR.



OSPF Operation

OSPF Operational States

State	Description
Down State	<ul style="list-style-type: none">•No Hello packets received = Down.•Router sends Hello packets.•Transition to Init state.
Init State	<ul style="list-style-type: none">•Hello packets are received from the neighbor.•They contain the Router ID of the sending router.•Transition to Two-Way state.
Two-Way State	<ul style="list-style-type: none">•In this state, communication between the two routers is bidirectional.•On multiaccess links, the routers elect a DR and a BDR.•Transition to ExStart state.

OSPF Operation

OSPF Operational States (Cont.)

State	Description
ExStart State	On point-to-point networks, the two routers decide which router will initiate the DBD packet exchange and decide upon the initial DBD packet sequence number.
Exchange State	<ul style="list-style-type: none">•Routers exchange DBD packets.•If additional router information is required then transition to Loading; otherwise, transition to the Full state.
Loading State	<ul style="list-style-type: none">•LSRs and LSUs are used to gain additional route information.•Routes are processed using the SPF algorithm.•Transition to the Full state.
Full State	The link-state database of the router is fully synchronized.

OSPF Operation

Establish Neighbor Adjacencies

- To determine if there is an OSPF neighbor on the link, the router sends a Hello packet that contains its router ID out all OSPF-enabled interfaces. The Hello packet is sent to the reserved All OSPF Routers IPv4 multicast address 224.0.0.5. Only OSPFv2 routers will process these packets.
- The OSPF router ID is used by the OSPF process to uniquely identify each router in the OSPF area. A router ID is a 32-bit number formatted like an IPv4 address and assigned to uniquely identify a router among OSPF peers.
- When a neighboring OSPF-enabled router receives a Hello packet with a router ID that is not within its neighbor list, the receiving router attempts to establish an adjacency with the initiating router.

OSPF Operation

Establish Neighbor Adjacencies (Cont.)

The process routers use to establish adjacency on a multiaccess network:

1	Down to Init State	When OSPFv2 is enabled on the interface, R1 transitions from Down to Init and starts sending OSPFv2 Hellos out of the interface in an attempt to discover neighbors.
2	Init State	When a R2 receives a hello from the previously unknown router R1, it adds R1's router ID to the neighbor list and responds with a Hello packet containing its own router ID.
3	Two-Way State	R1 receives R2's hello and notices that the message contains the R1 router ID in the list of R2's neighbors. R1 adds R2's router ID to the neighbor list and transitions to the Two-Way State. If R1 and R2 are connected with a point-to-point link, they transition to ExStart If R1 and R2 are connected over a common Ethernet network, the DR/BDR election occurs.
4	Elect the DR & BDR	The DR and BDR election occurs, where the router with the highest router ID or highest priority is elected as the DR, and second highest is the BDR

OSPF Operation

Synchronizing OSPF Databases

After the Two-Way state, routers transition to database synchronization states. This is a three step process, as follows:

- Decide first router: The router with the highest router ID sends its DBD first.
- Exchange DBDs: As many as needed to convey the database. The other router must acknowledge each DBD with an LSAck packet.
- Send an LSR: Each router compares the DBD information with the local LSDB. If the DBD has more current link information, the router transitions to the loading state.

After all LSRs have been exchanged and satisfied, the routers are considered synchronized and in a full state. Updates (LSUs) are sent:

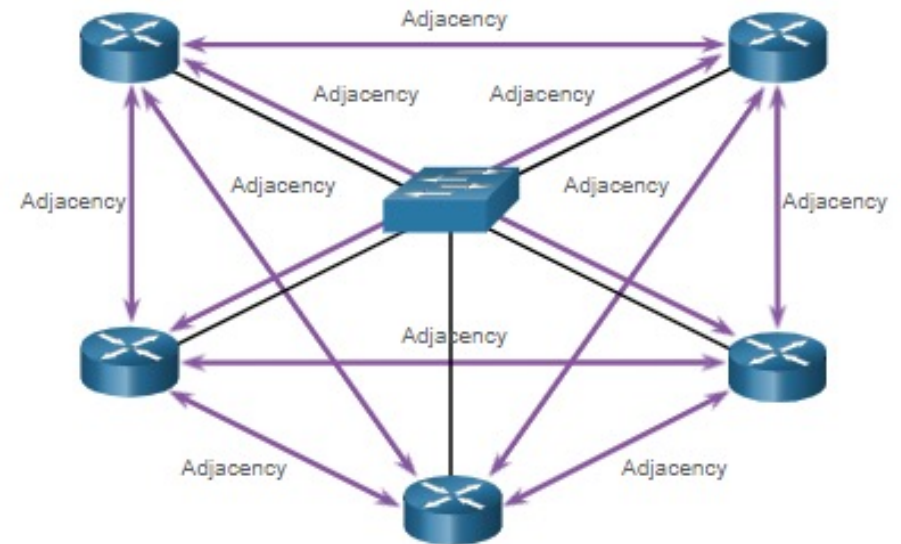
- When a change is perceived (incremental updates)
- Every 30 minutes

OSPF Operation

The Need for a DR

Multiaccess networks can create two challenges for OSPF regarding the flooding of LSAs, as follows:

- **Creation of multiple adjacencies** - Ethernet networks could potentially interconnect many OSPF routers over a common link. Creating adjacencies with every router would lead to an excessive number of LSAs exchanged between routers on the same network.
- **Extensive flooding of LSAs** - Link-state routers flood their LSAs any time OSPF is initialized, or when there is a change in the topology. This flooding can become excessive.



- Number of Adjacencies = $n(n - 1) / 2$
- n = number of routers
- Example: $5(5 - 1) / 2 = 10$ adjacencies

OSPF Operation

LSA Flooding with a DR

- An increase in the number of routers on a multiaccess network also increases the number of LSAs exchanged between the routers. This flooding of LSAs significantly impacts the operation of OSPF.
- If every router in a multiaccess network had to flood and acknowledge all received LSAs to all other routers on that same multiaccess network, the network traffic would become quite chaotic.
- On multiaccess networks, OSPF elects a DR to be the collection and distribution point for LSAs sent and received. A BDR is also elected in case the DR fails. All other routers become DROTHERs. A DROTHER is a router that is neither the DR nor the BDR.
- **Note:** The DR is only used for the dissemination of LSAs. The router will still use the best next-hop router indicated in the routing table for the forwarding of all other packets.

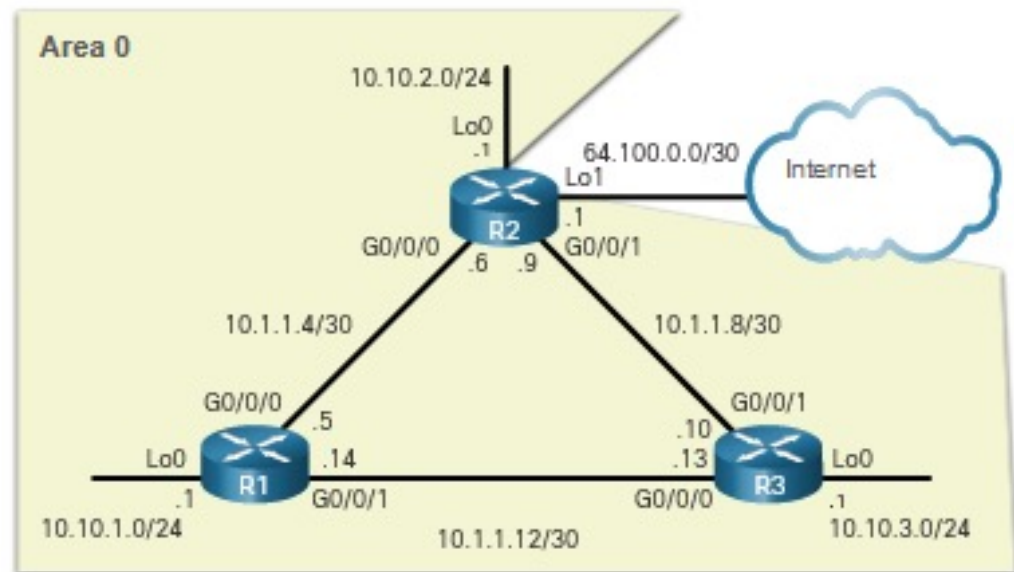
Implement single-area OSPFv2 in both point-to-point and broadcast multiaccess networks.

Topic Title	Topic Objective
OSPF Router ID	Configure an OSPFv2 router ID.
Point-to-Point OSPF Networks	Configure single-area OSPFv2 in a point-to-point network.
Multiaccess OSPF Networks	Configure the OSPF interface priority to influence the DR/BDR election in a multiaccess network.
Modify Single-Area OSPFv2	Implement modifications to change the operation of single-area OSPFv2.
Default Route Propagation	Configure OSPF to propagate a default route.
Verify Single-Area OSPFv2	Verify a single-area OSPFv2 implementation.

OSPF Router ID

OSPF Reference Topology

The figure shows the topology used for configuring OSPFv2 in this module. The routers in the topology have a starting configuration, including interface addresses. There is currently no static routing or dynamic routing configured on any of the routers. All interfaces on R1, R2, and R3 (except the loopback 1 on R2) are within the OSPF backbone area. The ISP router is used as the gateway to the internet of the routing domain.



OSPF Router ID

Router Configuration Mode for OSPF

OSPFv2 is enabled using the **router ospf process-id** global configuration mode command. The *process-id* value represents a number between 1 and 65,535 and is selected by the network administrator. The *process-id* value is locally significant. It is considered best practice to use the same *process-id* on all OSPF routers.

```
R1(config)# router ospf 10
R1(config-router)# ?
  area                OSPF area parameters
  auto-cost           Calculate OSPF interface cost according to
bandwidth
  default-information Control distribution of default information
  distance            Define an administrative distance
  exit                Exit from routing protocol configuration mode
  log-adjacency-changes Log changes in adjacency state
  neighbor            Specify a neighbor router
  network             Enable routing on an IP network
  no                  Negate a command or set its defaults
  passive-interface  Suppress routing updates on an interface
  redistribute        Redistribute information from another routing protocol
  router-id           router-id for this OSPF process
R1(config-router)#
```

OSPF Router ID

Router IDs

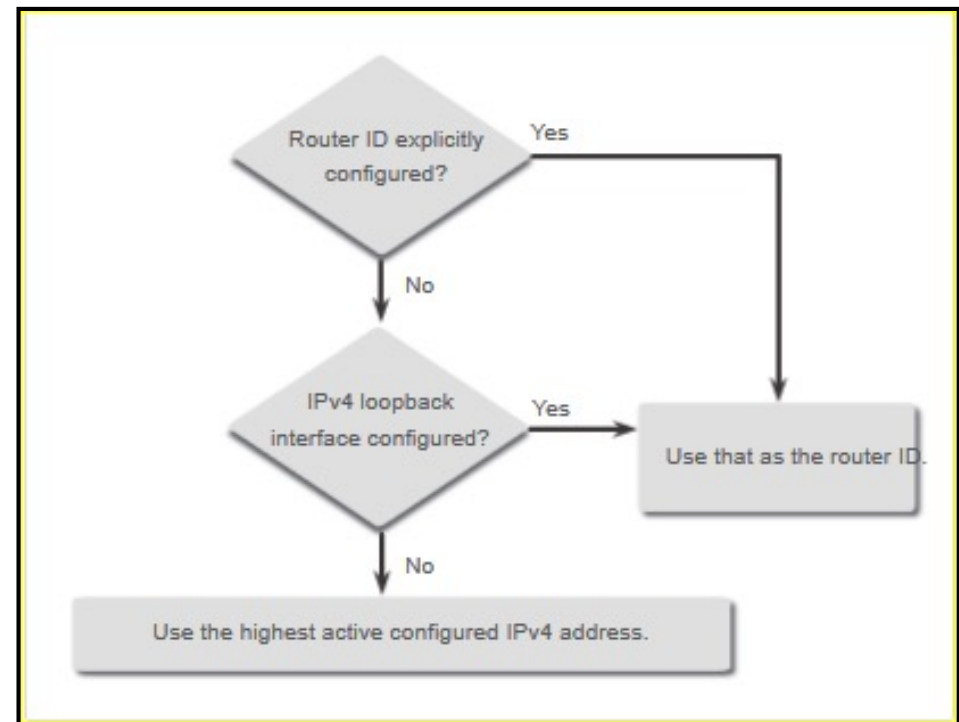
- An OSPF router ID is a 32-bit value, represented as an IPv4 address. It is used to uniquely identify an OSPF router, and all OSPF packets include the router ID of the originating router.
- Every router requires a router ID to participate in an OSPF domain. It can be defined by an administrator or automatically assigned by the router. The router ID is used by an OSPF-enabled router to do the following:
 - **Participate in the synchronization of OSPF databases** – During the Exchange State, the router with the highest router ID will send their database descriptor (DBD) packets first.
 - **Participate in the election of the designated router (DR)** - In a multiaccess LAN environment, the router with the highest router ID is elected the DR. The routing device with the second highest router ID is elected the backup designated router (BDR).

OSPF Router ID

Router ID Order of Precedence

Cisco routers derive the router ID based on one of three criteria, in the following preferential order:

1. The router ID is explicitly configured using the OSPF **router-id** *rid* router configuration mode command. This is the recommended method to assign a router ID.
2. The router chooses the highest IPv4 address of any of configured loopback interfaces.
3. The router chooses the highest active IPv4 address of any of its physical interfaces.



OSPF Router ID

Configure a Loopback Interface as the Router ID

Instead of relying on physical interface, the router ID can be assigned to a loopback interface. Typically, the IPv4 address for this type of loopback interface should be configured using a 32-bit subnet mask (255.255.255.255). This effectively creates a host route. A 32-bit host route would not get advertised as a route to other OSPF routers.

OSPF does not need to be enabled on an interface for that interface to be chosen as the router ID.

```
R1(config-if)# interface Loopback 1
R1(config-if)# ip address 1.1.1.1 255.255.255.255
R1(config-if)# end
R1# show ip protocols | include Router ID
  Router ID 1.1.1.1
R1#
```

OSPF Router ID

Explicitly Configure a Router ID

In our reference topology the router ID for each router is assigned as follows:

- R1 uses router ID 1.1.1.1
- R2 uses router ID 2.2.2.2
- R3 uses router ID 3.3.3.3

Use the **router-id** *rid* router configuration mode command to manually assign a router ID. In the example, the router ID 1.1.1.1 is assigned to R1. Use the **show ip protocols** command to verify the router ID.

```
R1(config)# router ospf 10
R1(config-router)# router-id 1.1.1.1
R1(config-router)# end
*May 23 19:33:42.689: %SYS-5-CONFIG_I: Configured from console by console
R1# show ip protocols | include Router ID
  Router ID 1.1.1.1
R1#
```

OSPF Router ID

Modify a Router ID

- After a router selects a router ID, an active OSPF router does not allow the router ID to be changed until the router is reloaded or the OSPF process is reset.
- Clearing the OSPF process is the preferred method to reset the router ID.

```
R1# show ip protocols | include Router ID
Router ID 10.10.1.1
R1# conf t
Enter configuration commands, one per line. End with CNTL/Z.
R1(config)# router ospf 10
R1(config-router)# router-id 1.1.1.1
% OSPF: Reload or use "clear ip ospf process" command, for this to take effect
R1(config-router)# end
R1# clear ip ospf process
Reset ALL OSPF processes? [no]: y
*Jun 6 01:09:46.975: %OSPF-5-ADJCHG: Process 10, Nbr 3.3.3.3 on GigabitEthernet0/0/1 from FULL to
DOWN, Neighbor Down: Interface down or detached
*Jun 6 01:09:46.981: %OSPF-5-ADJCHG: Process 10, Nbr 3.3.3.3 on GigabitEthernet0/0/1 from LOADING
to FULL, Loading Done *
R1# show ip protocols | include Router ID
Router ID 1.1.1.1
R1#
```

Point-to-Point OSPF Networks

The network Command Syntax

- You can specify the interfaces that belong to a point-to-point network by configuring the **network** command. You can also configure OSPF directly on the interface with the **ip ospf** command.
- The basic syntax for the **network** command is as follows:

```
Router(config-router)# network network-address wildcard-mask area area-id
```

- The *network-address wildcard-mask* syntax is used to enable OSPF on interfaces. Any interfaces on a router that match this part of the command are enabled to send and receive OSPF packets.
- The **area** *area-id* syntax refers to the OSPF area. When configuring single-area OSPFv2, the **network** command must be configured with the same *area-id* value on all routers. Although any area ID can be used, it is good practice to use an area ID of 0 with single-area OSPFv2. This convention makes it easier if the network is later altered to support multiarea OSPFv2.

Point-to-Point OSPF Networks

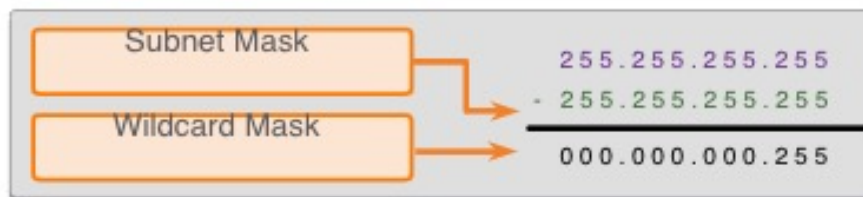
The Wildcard Mask

- The wildcard mask is typically the inverse of the subnet mask configured on that interface.
- The easiest method for calculating a wildcard mask is to subtract the network subnet mask from 255.255.255.255, as shown for /24 and /26 subnet masks in the figure.

Calculating a Wildcard Mask for /24



Calculating a Wildcard Mask for /26



Point-to-Point OSPF Networks

Configure OSPF Using the network Command

Within routing configuration mode, there are two ways to identify the interfaces that will participate in the OSPFv2 routing process.

- In the first example, the wildcard mask identifies the interface based on the network addresses. Any active interface that is configured with an IPv4 address belonging to that network will participate in the OSPFv2 routing process.
- **Note:** Some IOS versions allow the subnet mask to be entered instead of the wildcard mask. The IOS then converts the subnet mask to the wildcard mask format.

```
R1(config)# router ospf 10
R1(config-router)# network 10.10.1.0 0.0.0.255 area 0
R1(config-router)# network 10.1.1.4 0.0.0.3 area 0
R1(config-router)# network 10.1.1.12 0.0.0.3 area 0
R1(config-router)#
```

Point-to-Point OSPF Networks

Configure OSPF Using the network Command (Cont.)

- As an alternative, OSPFv2 can be enabled by specifying the exact interface IPv4 address using a quad zero wildcard mask. Entering **network 10.1.1.5 0.0.0.0 area 0** on R1 tells the router to enable interface Gigabit Ethernet 0/0/0 for the routing process.
- The advantage of specifying the interface is that the wildcard mask calculation is not necessary. Notice that in all cases, the **area** argument specifies area 0.

```
R1(config)# router ospf 10  
R1(config-router)# network 10.10.1.1 0.0.0.0 area 0  
R1(config-router)# network 10.1.1.5 0.0.0.0 area 0  
R1(config-router)# network 10.1.1.14 0.0.0.0 area 0  
R1(config-router)#
```

Point-to-Point OSPF Networks

Configure OSPF Using the `ip ospf` Command

To configure OSPF directly on the interface, use the **ip ospf** interface configuration mode command. The syntax is as follows:

```
Router(config-if)# ip ospf process-id area area-id
```

Remove the network commands using the **no** form of the command. Then go to each interface and configure the **ip ospf** command

```
R1(config)# router ospf 10
R1(config-router)# no network 10.10.1.1 0.0.0.0 area 0
R1(config-router)# no network 10.1.1.5 0.0.0.0 area 0
R1(config-router)# no network 10.1.1.14 0.0.0.0 area 0
R1(config-router)# interface GigabitEthernet 0/0/0
R1(config-if)# ip ospf 10 area 0
R1(config-if)# interface GigabitEthernet 0/0/1
R1(config-if)# ip ospf 10 area 0
R1(config-if)# interface Loopback 0
R1(config-if)# ip ospf 10 area 0
R1(config-if)#
```


Point-to-Point OSPF Networks

Passive Interface

By default, OSPF messages are forwarded out all OSPF-enabled interfaces. However, these messages only need to be sent out interfaces that are connecting to other OSPF-enabled routers.

Sending out unneeded messages on a LAN affects the network in three ways:

- **Inefficient Use of Bandwidth** - Available bandwidth is consumed transporting unnecessary messages.
- **Inefficient Use of Resources** - All devices on the LAN must process and eventually discard the message.
- **Increased Security Risk** - Without additional OSPF security configurations, OSPF messages can be intercepted with packet sniffing software. Routing updates can be modified and sent back to the router, corrupting the routing table with false metrics that misdirect traffic.

Point-to-Point OSPF Networks

Configure Passive Interfaces

- Use the **passive-interface** router configuration mode command to prevent the transmission of routing messages through a router interface, but still allow that network to be advertised to other routers.
- The **show ip protocols** command is then used to verify that the interface is listed as passive.

```
R1(config)# router ospf 10
R1(config-router)# passive-interface loopback 0
R1(config-router)# end
R1#
*May 23 20:24:39.309: %SYS-5-CONFIG_I: Configured from console by console
R1# show ip protocols
*** IP Routing is NSF aware ***
(output omitted)
Routing Protocol is "ospf 10"
  Outgoing update filter list for all interfaces is not set
  Incoming update filter list for all interfaces is not set
  Router ID 1.1.1.1
  Number of areas in this router is 1. 1 normal 0 stub 0 nssa
  Maximum path: 4
  Routing for Networks:
  Routing on Interfaces Configured Explicitly (Area 0):
    Loopback0
    GigabitEthernet0/0/1
    GigabitEthernet0/0/0
  Passive Interface(s):
    Loopback0
  Routing Information Sources:
    Gateway         Distance        Last Update
    3.3.3.3          110             01:01:48
    2.2.2.2          110             01:01:38
  Distance: (default is 110)
R1#
```

Point-to-Point OSPF Networks

OSPF Point-to-Point Networks

By default, Cisco routers elect a DR and BDR on Ethernet interfaces, even if there is only one other device on the link. You can verify this with the **show ip ospf interface** command. The DR/ BDR election process is unnecessary as there can only be two routers on the point-to-point network between R1 and R2. Notice in the output that the router has designated the network type as BROADCAST.

```
R1# show ip ospf interface GigabitEthernet 0/0/0
GigabitEthernet0/0/0 is up, line protocol is up
  Internet Address 10.1.1.5/30, Area 0, Attached via Interface Enable
  Process ID 10, Router ID 1.1.1.1, Network Type BROADCAST, Cost: 1
  Topology-MTID      Cost      Disabled      Shutdown      Topology Name
    0                1         no            no            Base
  Enabled by interface config, including secondary ip addresses
  Transmit Delay is 1 sec, State BDR, Priority 1
  Designated Router (ID) 2.2.2.2, Interface address 10.1.1.6
  Backup Designated router (ID) 1.1.1.1, Interface address 10.1.1.5
  Timer intervals configured, Hello 10, Dead 40, Wait 40, Retransmit 5
  oob-resync timeout 40
```

Point-to-Point OSPF Networks

OSPF Point-to-Point Networks (Cont.)

To change this to a point-to-point network, use the interface configuration command **ip ospf network point-to-point** on all interfaces where you want to disable the DR/BDR election process.

```
R1(config)# interface GigabitEthernet 0/0/0
R1(config-if)# ip ospf network point-to-point
*Jun 6 00:44:05.208: %OSPF-5-ADJCHG: Process 10, Nbr 2.2.2.2 on GigabitEthernet0/0/0 from
FULL to DOWN, Neighbor Down: Interface down or detached
*Jun 6 00:44:05.211: %OSPF-5-ADJCHG: Process 10, Nbr 2.2.2.2 on GigabitEthernet0/0/0 from
LOADING to FULL, Loading Done
R1(config-if)# end
R1# show ip ospf interface GigabitEthernet 0/0/0
GigabitEthernet0/0/0 is up, line protocol is up
  Internet Address 10.1.1.5/30, Area 0, Attached via Interface Enable
  Process ID 10, Router ID 1.1.1.1, Network Type POINT_TO_POINT, Cost: 1
  Topology-MTID      Cost      Disabled      Shutdown      Topology Name
```

Point-to-Point OSPF Networks

Loopbacks and Point-to-Point Networks

- Use loopbacks to provide additional interfaces for a variety of purposes. By default, loopback interfaces are advertised as /32 host routes.
- To simulate a real LAN, the loopback interface can be configured as a point-to-point network to advertise the full network.
- What R2 sees when R1 advertises the loopback interface as-is:

```
R2# show ip route | include 10.10.1
O          10.10.1.1/32 [110/2] via 10.1.1.5, 00:03:05, GigabitEthernet0/0/0
```

- Configuration change at R1:

```
R1(config-if)# interface Loopback 0
R1(config-if)# ip ospf network point-to-point
```

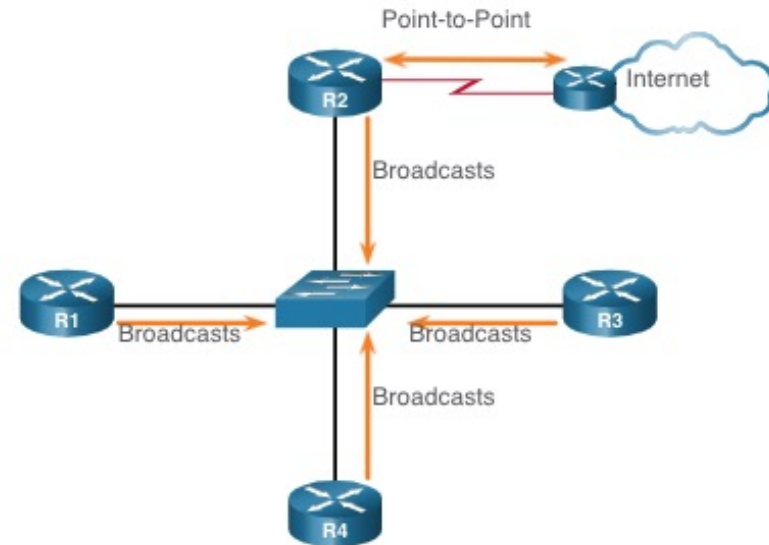
- Result at R2:

```
R2# show ip route | include 10.10.1
O          10.10.1.0/24 [110/2] via 10.1.1.5, 00:03:05, GigabitEthernet0/0/0
```

Multiaccess OSPF Networks OSPF Network Types

Another type of network that uses OSPF is the multiaccess OSPF network. Multiaccess OSPF networks are unique in that one router controls the distribution of LSAs.

The router that is elected for this role should be determined by the network administrator through proper configuration.



Multiaccess OSPF Networks

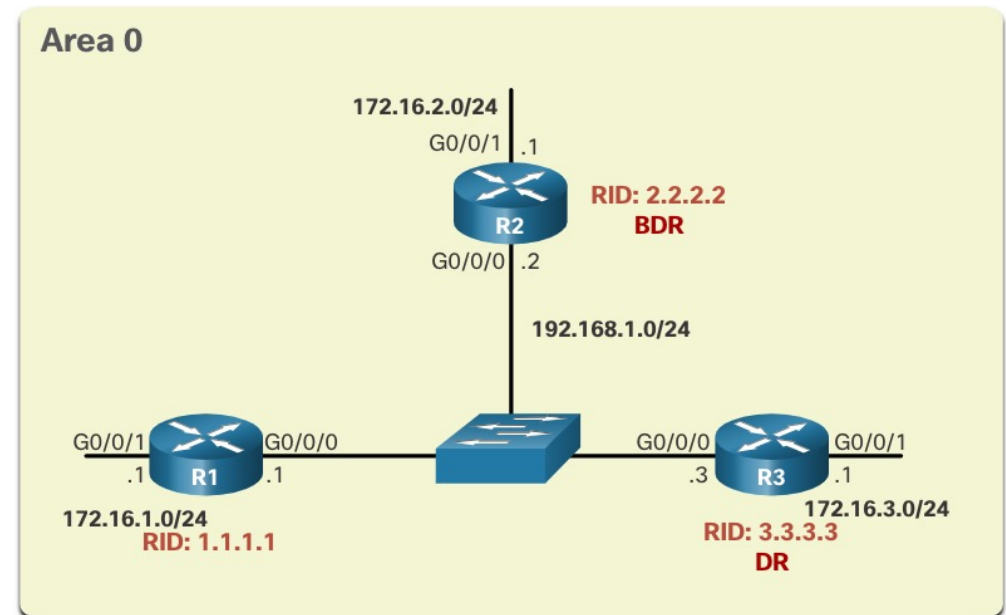
OSPF Designated Router

- In multiaccess networks, OSPF elects a DR and BDR. The DR is responsible for collecting and distributing LSAs sent and received. The DR uses the multicast IPv4 address 224.0.0.5 which is meant for all OSPF routers.
- A BDR is also elected in case the DR fails. The BDR listens passively and maintains a relationship with all the routers. If the DR stops producing Hello packets, the BDR promotes itself and assumes the role of DR.
- All other routers become a DROTHER (a router that is neither the DR nor the BDR). DROTHERs use the multiaccess address 224.0.0.6 (all designated routers) to send OSPF packets to the DR and BDR. Only the DR and BDR listen for 224.0.0.6.

Multiaccess OSPF Networks

OSPF Multiaccess Reference Topology

- In the multiaccess topology shown in the figure, there are three routers interconnected over a common Ethernet multiaccess network, 192.168.1.0/24.
- Because the routers are connected over a common multiaccess network, OSPF has automatically elected a DR and BDR. R3 has been elected as the DR because its router ID is 3.3.3.3, which is the highest in this network. R2 is the BDR because it has the second highest router ID in the network.



Multiaccess OSPF Networks

Verify OSPF Router Roles

To verify the roles of the OSPFv2 router, use the **show ip ospf interface** command.

The output generated by R1 confirms that the following:

- R1 is not the DR or BDR, but is a DROTHER with a default priority of 1. (Line 7)
- The DR is R3 with router ID 3.3.3.3 at IPv4 address 192.168.1.3, while the BDR is R2 with router ID 2.2.2.2 at IPv4 address 192.168.1.2. (Lines 8 and 9)
- R1 has two adjacencies: one with the BDR and one with the DR. (Lines 20-22)

```
R1# show ip ospf interface GigabitEthernet 0/0/0
GigabitEthernet0/0/0 is up, line protocol is up
  Internet Address 192.168.1.1/24, Area 0, Attached via Interface Enable
  Process ID 10, Router ID 1.1.1.1, Network Type BROADCAST, Cost: 1
  (output omitted)
  Transmit Delay is 1 sec, State DROTHER, Priority 1
  Designated Router (ID) 3.3.3.3, Interface address 192.168.1.3
  Backup Designated router (ID) 2.2.2.2, Interface address 192.168.1.2
  (output omitted)
  Neighbor Count is 2, Adjacent neighbor count is 2
  Adjacent with neighbor 2.2.2.2 (Backup Designated Router)
  Adjacent with neighbor 3.3.3.3 (Designated Router)
  Suppress hello for 0 neighbor(s)
R1#
```

Multiaccess OSPF Networks

Verify OSPF Router Roles (Cont.)

The output generated by R2 confirms that:

- R2 is the BDR with a default priority of 1. (Line 7)
- The DR is R3 with router ID 3.3.3.3 at IPv4 address 192.168.1.3, while the BDR is R2 with router ID 2.2.2.2 at IPv4 address 192.168.1.2. (Lines 8 and 9)
- R2 has two adjacencies; one with a neighbor with router ID 1.1.1.1 (R1) and the other with the DR. (Lines 20-22)

```
R2# show ip ospf interface GigabitEthernet 0/0/0
GigabitEthernet0/0/0 is up, line protocol is up
  Internet Address 192.168.1.2/24, Area 0, Attached via Interface Enable
  Process ID 10, Router ID 2.2.2.2, Network Type BROADCAST, Cost: 1
  (output omitted)
  Transmit Delay is 1 sec, State BDR, Priority 1
  Designated Router (ID) 3.3.3.3, Interface address 192.168.1.3
  Backup Designated Router (ID) 2.2.2.2, Interface address 192.168.1.2
  (output omitted)
  Neighbor Count is 2, Adjacent neighbor count is 2
  Adjacent with neighbor 1.1.1.1
  Adjacent with neighbor 3.3.3.3 (Designated Router)
  Suppress hello for 0 neighbor(s)
R2#
```

Multiaccess OSPF Networks

Verify OSPF Router Roles (Cont.)

The output generated by R3 confirms that:

- R3 is the DR with a default priority of 1. (Line 7)
- The DR is R3 with router ID 3.3.3.3 at IPv4 address 192.168.1.3, while the BDR is R2 with router ID 2.2.2.2 at IPv4 address 192.168.1.2. (Lines 8 and 9)
- R3 has two adjacencies: one with a neighbor with router ID 1.1.1.1 (R1) and the other with the BDR. (Lines 20-22)

```
R1# show ip ospf interface GigabitEthernet 0/0/0
GigabitEthernet0/0/0 is up, line protocol is up
  Internet Address 192.168.1.1/24, Area 0, Attached via Interface Enable
  Process ID 10, Router ID 1.1.1.1, Network Type BROADCAST, Cost: 1
  (output omitted)
  Transmit Delay is 1 sec, State DROTHER, Priority 1
  Designated Router (ID) 3.3.3.3, Interface address 192.168.1.3
  Backup Designated router (ID) 2.2.2.2, Interface address 192.168.1.2
  (output omitted)
  Neighbor Count is 2, Adjacent neighbor count is 2
  Adjacent with neighbor 2.2.2.2 (Backup Designated Router)
  Adjacent with neighbor 3.3.3.3 (Designated Router)
  Suppress hello for 0 neighbor(s)
R1#
```

Multiaccess OSPF Networks

Verify DR/BDR Adjacencies

To verify the OSPFv2 adjacencies, use the **show ip ospf neighbor** command. The state of neighbors in multiaccess networks can be as follows:

- **FULL/DROTHER** - This is a DR or BDR router that is fully adjacent with a non-DR or BDR router. These two neighbors can exchange Hello packets, updates, queries, replies, and acknowledgments.
- **FULL/DR** - The router is fully adjacent with the indicated DR neighbor. These two neighbors can exchange Hello packets, updates, queries, replies, and acknowledgments.
- **FULL/BDR** - The router is fully adjacent with the indicated BDR neighbor. These two neighbors can exchange Hello packets, updates, queries, replies, and acknowledgments.
- **2-WAY/DROTHER** - The non-DR or BDR router has a neighbor relationship with another non-DR or BDR router. These two neighbors exchange Hello packets.

The normal state for an OSPF router is usually FULL. If a router is stuck in another state, it is an indication that there are problems in forming adjacencies. The only exception to this is the 2-WAY state, which is normal in a multiaccess broadcast network.

Multiaccess OSPF Networks

Verify DR/BDR Adjacencies (Cont.)

The output generated by R2 confirms that R2 has adjacencies with the following routers:

- R1 with router ID 1.1.1.1 is in a Full state and R1 is neither the DR nor BDR.
- R3 with router ID 3.3.3.3 is in a Full state and the role of R3 is DR.

```
R2# show ip ospf neighbor
Neighbor ID      Pri      State           Dead Time      Address         Interface
1.1.1.1          1        FULL/DROTHER    00:00:31      192.168.1.1    GigabitEthernet0/0/0
3.3.3.3          1        FULL/DR         00:00:34      192.168.1.3    GigabitEthernet0/0/0
R2#
```

Multiaccess OSPF Networks

Default DR/BDR Election Process

The OSPF DR and BDR election is based on the following criteria, in sequential order:

1. The routers in the network elect the router with the highest interface priority as the DR. The router with the second highest interface priority is becomes the BDR.
 - The priority can be configured to be any number between 0 – 255.
 - If the interface priority value is set to 0, that interface cannot be elected as DR nor BDR.
 - The default priority of multiaccess broadcast interfaces is 1.
2. If the interface priorities are equal, then the router with the highest router ID is elected the DR. The router with the second highest router ID is the BDR.
 - The election process takes place when the first router with an OSPF-enabled interface is active on the network. If all of the routers on the network have not finished booting, it is possible that a router with a lower router ID becomes the DR.
 - The addition of a new router does not initiate a new election process.

Multiaccess OSPF Networks

DR Failure and Recovery

After the DR is elected, it remains the DR until one of the following events occurs:

- The DR fails.
- The OSPF process on the DR fails or is stopped.
- The multiaccess interface on the DR fails or is shutdown.

If the DR fails, the BDR is automatically promoted to DR. This is the case even if another DROTHER with a higher priority or router ID is added to the network after the initial DR/BDR election. However, after a BDR is promoted to DR, a new BDR election occurs and the DROTHER with the highest priority or router ID is elected as the new BDR.

Multiaccess OSPF Networks

The ip ospf priority Command

- If the interface priorities are equal on all routers, the router with the highest router ID is elected the DR.
- Instead of relying on the router ID, it is better to control the election by setting interface priorities. This also allows a router to be the DR in one network and a DROTHER in another.
- To set the priority of an interface, use the command **ip ospf priority *value***, where *value* is 0 to 255.
- A value of 0 does not become a DR or a BDR.
- A value of 1 to 255 on the interface makes it more likely that the router becomes the DR or the BDR.

Multiaccess OSPF Networks

Configure OSPF Priority

The example shows the commands being used to change the R1 G0/0/0 interface priority from 1 to 255 and then reset the OSPF process.

```
R1(config)# interface GigabitEthernet 0/0/0
R1(config-if)# ip ospf priority 255
R1(config-if)# end
R1# clear ip ospf process
Reset ALL OSPF processes? [no]: y
R1# *Jun 5 03:47:41.563: %OSPF-5-ADJCHG: Process 10, Nbr 2.2.2.2 on GigabitEthernet0/0/0
from FULL to DOWN, Neighbor Down: Interface down or detached
```

Modify Single-Area OSPFv2

Cisco OSPF Cost Metric

- Routing protocols use a metric to determine the best path of a packet across a network. OSPF uses cost as a metric. A lower cost indicates a better path.
- The Cisco cost of an interface is inversely proportional to the bandwidth of the interface. Therefore, a higher bandwidth indicates a lower cost. The formula used to calculate the OSPF cost is:

$$\text{Cost} = \text{reference bandwidth} / \text{interface bandwidth}$$

- The default reference bandwidth is 10^8 (100,000,000); therefore, the formula is:

$$\text{Cost} = 100,000,000 \text{ bps} / \text{interface bandwidth in bps}$$

- Because the OSPF cost value must be an integer, FastEthernet, Gigabit Ethernet, and 10 GigE interfaces share the same cost. To correct this situation, you can:
 - Adjust the reference bandwidth with the **auto-cost reference-bandwidth** command on each OSPF router.
 - Manually set the OSPF cost value with the **ip ospf cost** command on necessary interfaces.

Modify Single-Area OSPFv2 Cisco OSPF Cost Metric (Cont.)

Refer to the table for a breakdown of the cost calculation

Interface Type	Reference Bandwidth in bps		Default Bandwidth in bps	Cost
10 Gigabit Ethernet 10 Gbps	100,000,000	÷	10,000,000,000	0.01 = 1
Gigabit Ethernet 1 Gbps	100,000,000	÷	1,000,000,000	0.1 = 1
Fast Ethernet 100 Mbps	100,000,000	÷	100,000,000	1
Ethernet 10 Mbps	100,000,000	÷	10,000,000	1

Same Costs due to reference bandwidth

Modify Single-Area OSPFv2

Adjust the Reference Bandwidth

- The cost value must be an integer. If something less than an integer is calculated, OSPF rounds up to the nearest integer. Therefore, the OSPF cost assigned to a Gigabit Ethernet interface with the default reference bandwidth of 100,000,000 bps would equal 1, because the nearest integer for 0.1 is 1 instead of 0.

$$\text{Cost} = 100,000,000 \text{ bps} / 1,000,000,000 = 0.1 \rightarrow 1$$

- For this reason, all interfaces faster than Fast Ethernet will have the same cost value of 1 as a Fast Ethernet interface.
- To assist OSPF in making the correct path determination, the reference bandwidth must be changed to a higher value to accommodate networks with links faster than 100 Mbps.

Modify Single-Area OSPFv2

Adjust the Reference Bandwidth (Cont.)

- Changing the reference bandwidth does not actually affect the bandwidth capacity on the link; rather, it simply affects the calculation used to determine the metric.
- To adjust the reference bandwidth, use the **auto-cost reference-bandwidth** *Mbps* router configuration command.
- This command must be configured on every router in the OSPF domain.
- Notice in the command that the value is expressed in Mbps; therefore, to adjust the costs for Gigabit Ethernet, use the command **auto-cost reference-bandwidth 1000**. For 10 Gigabit Ethernet, use the command **auto-cost reference-bandwidth 10000**.
- To return to the default reference bandwidth, use the **auto-cost reference-bandwidth 100** command.
- Another option is to change the cost on one specific interface using the **ip ospf cost** *cost* command.

Modify Single-Area OSPFv2

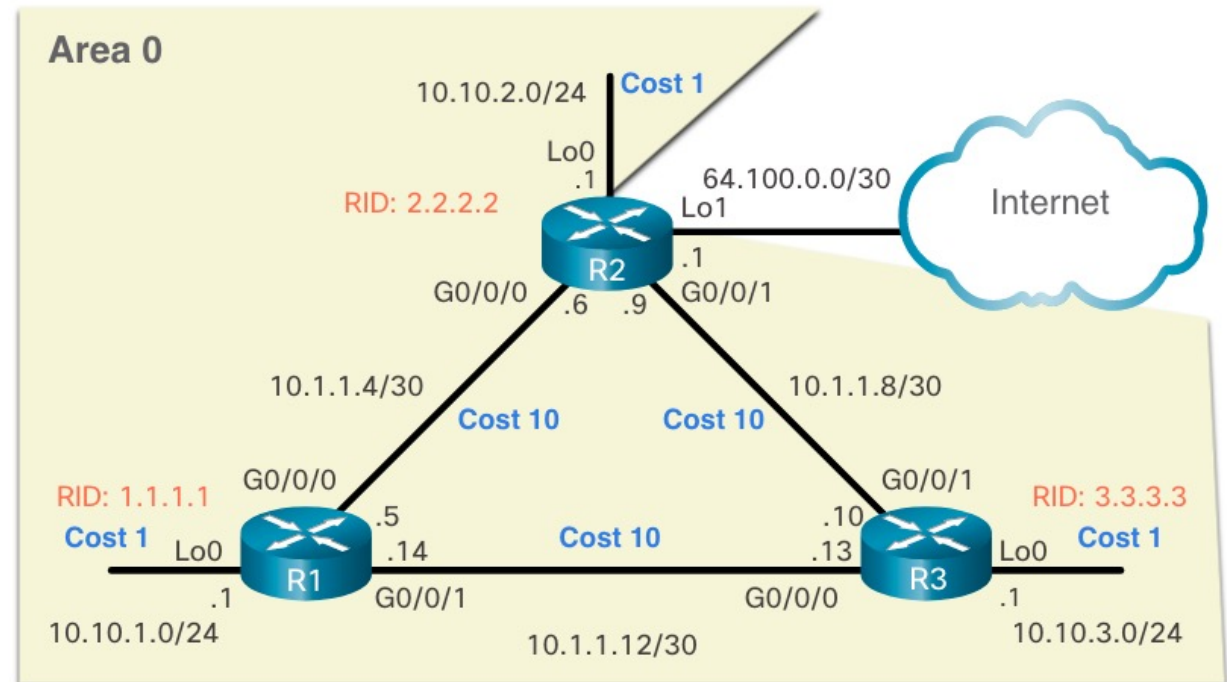
Adjust the Reference Bandwidth (Cont.)

- Whichever method is used, it is important to apply the configuration to all routers in the OSPF routing domain.
- The table shows the OSPF cost if the reference bandwidth is adjusted to accommodate 10 Gigabit Ethernet links. The reference bandwidth should be adjusted anytime there are links faster than FastEthernet (100 Mbps).
- Use the **show ip ospf interface** command to verify the current OSPFv2 cost assigned to the interface.

Interface Type	Reference Bandwidth in bps		Default Bandwidth in bps	Cost
10 Gigabit Ethernet 10 Gbps	10,000,000,000	÷	10,000,000,000	1
Gigabit Ethernet 1 Gbps	10,000,000,000	÷	1,000,000,000	10
Fast Ethernet 100 Mbps	10,000,000,000	÷	100,000,000	100
Ethernet 10 Mbps	10,000,000,000	÷	10,000,000	1000

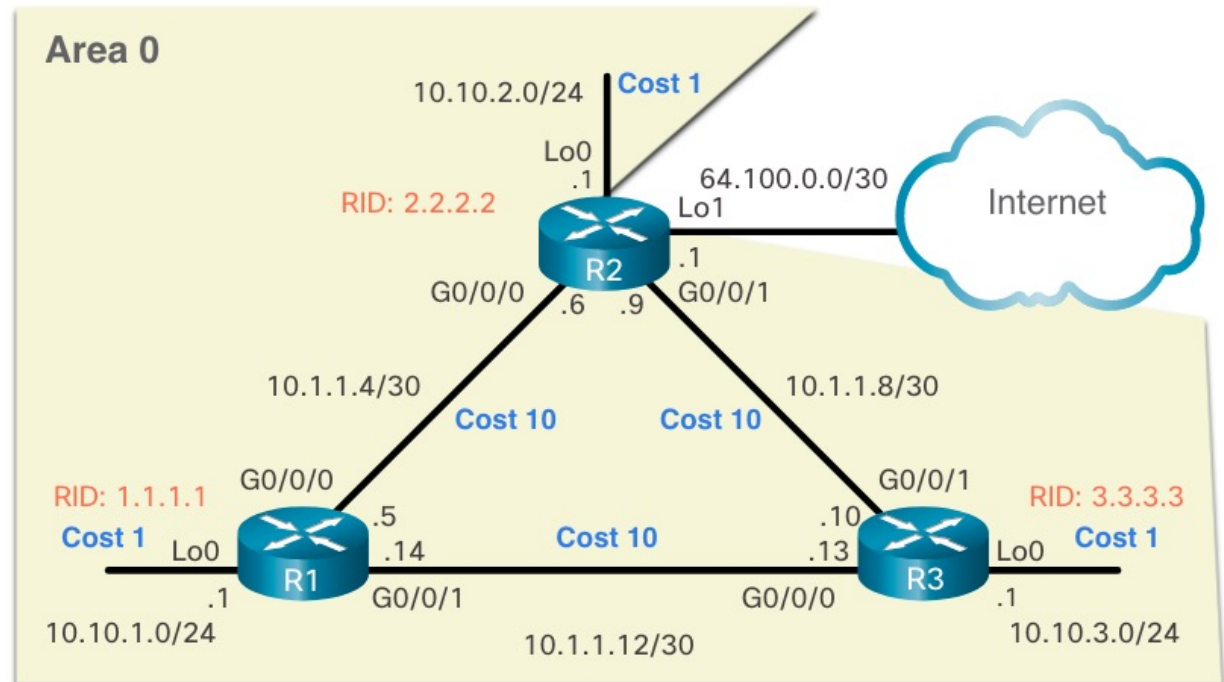
Modify Single-Area OSPFv2 OSPF Accumulates Cost

- The cost of an OSPF route is the accumulated value from one router to the destination network.
- Assuming the **auto-cost reference-bandwidth 10000** command has been configured on all three routers, the cost of the links between each router is now 10. The loopback interfaces have a default cost of 1.



Modify Single-Area OSPFv2 OSPF Accumulates Cost (Cont.)

- You can calculate the cost for each router to reach each network.
- For example, the total cost for R1 to reach the 10.10.2.0/24 network is 11. This is because the link to R2 cost = 10 and the loopback default cost = 1. $10 + 1 = 11$.
- You can verify this with the **show ip route** command.



Modify Single-Area OSPFv2 OSPF Accumulates Cost (Cont.)

Verifying the accumulated cost for the path to the 10.10.2.0/24 network:

```
R1# show ip route | include 10.10.2.0
O          10.10.2.0/24 [110/11] via 10.1.1.6, 01:05:02, GigabitEthernet0/0/0
R1# show ip route 10.10.2.0
Routing entry for 10.10.2.0/24
  Known via "ospf 10", distance 110, metric 11, type intra area
  Last update from 10.1.1.6 on GigabitEthernet0/0/0, 01:05:13 ago
  Routing Descriptor Blocks:
    * 10.1.1.6, from 2.2.2.2, 01:05:13 ago, via GigabitEthernet0/0/0
      Route metric is 11, traffic share count is 1
R1#
```

Modify Single-Area OSPFv2 Manually Set OSPF Cost Value

Reasons to manually set the cost value include:

- The Administrator may want to influence path selection within OSPF, causing different paths to be selected than what normally would be given default costs and cost accumulation.
- Connections to equipment from other vendors who use a different formula to calculate OSPF cost.

To change the cost of an interface configuration on routers, use the

```
R1(config)# interface g0/0/1 R1(config-if)# ip  
ospf cost 30 R1(config-if)# interface lo0  
R1(config-if)# ip ospf cost 10 R1(config-if)#  
end  
R1#
```

Modify Single-Area OSPFv2 Test Failover to Backup Route

What happens if the link between R1 and R2 goes down? You can simulate that by shutting down the Gigabit Ethernet 0/0/0 interface and verifying the routing table is updated to use R3 as the next-hop router. Notice that R1 can now reach the 10.1.1.4/30 network through R3 with a cost value of 50.

```
R1# show ip route ospf | begin 10
      10.0.0.0/8 is variably subnetted, 8 subnets, 3 masks
O       10.1.1.4/30 [110/50] via 10.1.1.13, 00:00:14, GigabitEthernet0/0/1
O       10.1.1.8/30 [110/40] via 10.1.1.13, 00:00:14, GigabitEthernet0/0/1
O       10.10.2.0/24 [110/50] via 10.1.1.13, 00:00:14, GigabitEthernet0/0/1
O       10.10.3.0/24 [110/40] via 10.1.1.13, 00:00:14, GigabitEthernet0/0/1
R1#
```

Modify Single-Area OSPFv2 Hello Packet Intervals

- OSPFv2 Hello packets are transmitted to multicast address 224.0.0.5 (all OSPF routers) every 10 seconds. This is the default timer value on multiaccess and point-to-point networks.

Note: Hello packets are not sent on interfaces set to passive by the **passive-interface** command.

- The Dead interval is the period that the router waits to receive a Hello packet before declaring the neighbor down. If the Dead interval expires before the routers receive a Hello packet, OSPF removes that neighbor from its link-state database (LSDB). The router floods the LSDB with information about the down neighbor out all OSPF-enabled interfaces. Cisco uses a default of 4 times the Hello interval. This is 40 seconds on multiaccess and point-to-point networks.

Modify Single-Area OSPFv2 Verify Hello and Dead Intervals

- The OSPF Hello and Dead intervals are configurable on a per-interface basis.
- The OSPF intervals must match or a neighbor adjacency does not occur.
- To verify the currently configured OSPFv2 interface intervals, use the **show ip ospf interface** command. The Gigabit Ethernet 0/0/0 Hello and Dead intervals are set to the default 10 seconds and 40 seconds respectively.

```
R1# show ip ospf interface g0/0/0
GigabitEthernet0/0/0 is up, line protocol is up
  Internet Address 10.1.1.5/30, Area 0, Attached via Interface Enable
  Process ID 10, Router ID 1.1.1.1, Network Type POINT_TO_POINT, Cost: 10
  Topology-MTID    Cost      Disabled      Shutdown      Topology Name
                0         10           no            no
Base
  Enabled by interface config, including secondary ip addresses
  Transmit Delay is 1 sec, State POINT_TO_POINT
  Timer intervals configured, Hello 10, Dead 40, Wait 40, Retransmit 5
    oob-resync timeout 40
(output omitted)
```

Modify Single-Area OSPFv2 Verify Hello and Dead Intervals (Cont.)

Use the **show ip ospf neighbor** command to see the Dead Time counting down from 40 seconds. By default, this value is refreshed every 10 seconds when R1 receives a Hello from the neighbor.

```
R1# show ip ospf neighbor
Neighbor ID      Pri      State      Dead Time      Address         Interface
3.3.3.3          0        FULL/ -    00:00:35      10.1.1.13      GigabitEthernet0/0/1
2.2.2.2          0        FULL/ -    00:00:31      10.1.1.6       GigabitEthernet0/0/0
R1#
```

Modify Single-Area OSPFv2

Modify OSPFv2 Intervals

- It may be desirable to change the OSPF timers so that routers detect network failures in less time. Doing this increases traffic, but sometimes the need for quick convergence is more important than the extra traffic it creates.

Note: The default Hello and Dead intervals are based on best practices and should only be altered in rare situations.

- OSPFv2 Hello and Dead intervals can be modified manually using the following interface configuration mode commands:

```
Router(config-if) # ip ospf hello-interval seconds  
Router(config-if) # ip ospf dead-interval seconds
```

- Use the **no ip ospf hello-interval** and **no ip ospf dead-interval** commands to reset the intervals to their default.

Modify Single-Area OSPFv2 Modify OSPFv2 Intervals (Cont.)

- In the example, the Hello interval for the link between R1 and R2 is changed to 5 seconds. The Cisco IOS automatically modifies the Dead interval to four times the Hello interval. However, you can document the new Dead interval in the configuration by manually setting it to 20 seconds, as shown.
- When the Dead Timer on R1 expires, R1 and R2 lose adjacency. R1 and R2 must be configured with the same Hello interval. Use the **show ip ospf neighbor** command on R1 to verify the neighbor adjacencies.

```
R1(config)# interface g0/0/0
R1(config-if)# ip ospf hello-interval 5
R1(config-if)# ip ospf dead-interval 20
R1(config-if)#
*Jun 7 04:56:07.571: %OSPF-5-ADJCHG: Process 10, Nbr 2.2.2.2 on GigabitEthernet0/0/0
from FULL to DOWN, Neighbor Down: Dead timer expired
R1(config-if)# end
R1# show ip ospf neighbor
Neighbor ID      Pri       State       Dead Time           Address             Interface
3.3.3.3          0         FULL/ -    00:00:37           10.1.1.13          GigabitEthernet0/0/1
R1#
```


Default Route Propagation

Propagate a Default Static Route in OSPFv2

To propagate a default route, the edge router must be configured with the following:

- A default static route using the **ip route 0.0.0.0 0.0.0.0** [*next-hop-address* | *exit-intf*] command.
- The **default-information originate** router configuration command. This instructs R2 to be the source of the default route information and propagate the default static route in OSPF updates.

In the example, R2 is configured with a loopback to simulate a connection to the internet. A default route is configured and propagated to all other OSPF routers in the routing domain.

Note: When configuring static routes, best practice is to use the next-hop IP address. However, when simulating a connection to the internet, there is no next-hop IP address. Therefore, we use the *exit-intf* argument.

```
R2(config)# interface lo1
R2(config-if)# ip address 64.100.0.1 255.255.255.252
R2(config-if)# exit
R2(config)# ip route 0.0.0.0 0.0.0.0 loopback 1
%Default route without gateway, if not a point-to-point interface, may impact performance
R2(config)# router ospf 10
R2(config-router)# default-information originate
R2(config-router)# end
R2#
```

Default Route Propagation

Verify the Propagated Default Route

- You can verify the default route settings on R2 using the **show ip route** command. You can also verify that R1 and R3 received a default route.
- Notice that the route source on R1 is **O*E2**, signifying that it was learned using OSPFv2. The asterisk identifies this as a good candidate for the default route. The E2 designation identifies that it is an external route. The meaning of E1 and E2 is beyond the scope of this module.

```
R2# show ip route | begin Gateway
Gateway of last resort is 0.0.0.0 to network 0.0.0.0
S*      0.0.0.0/0 is directly connected, Loopback1
        10.0.0.0/8 is variably subnetted, 9 subnets, 3 masks
(output omitted)
```

```
R1# show ip route | begin Gateway
Gateway of last resort is 10.1.1.6 to network 0.0.0.0
O*E2   0.0.0.0/0 [110/1] via 10.1.1.6, 00:11:08, GigabitEthernet0/0/0
        10.0.0.0/8 is variably subnetted, 9 subnets, 3 masks
(output omitted)
```

Verify Single-Area OSPFv2

Verify OSPF Neighbors

After configuring single-area OSPFv2, you will need to verify your configurations. The following two commands are particularly useful for verifying routing:

- **show ip interface brief** - This verifies that the desired interfaces are active with correct IP addressing.
- **show ip route**- This verifies that the routing table contains all the expected routes.

Additional commands for determining that OSPF is operating as expected include the following:

- **show ip ospf neighbor**
- **show ip protocols**
- **show ip ospf**
- **show ip ospf interface**

Verify Single-Area OSPFv2

Verify OSPF Neighbors (Cont.)

- Use the **show ip ospf neighbor** command to verify that the router has formed an adjacency with its neighboring routers. If the router ID of the neighboring router is not displayed, or if it does not show as being in a state of FULL, the two routers have not formed an OSPFv2 adjacency.

Note: A non-DR or BDR router that has a neighbor relationship with another non-DR or BDR router will display a two-way adjacency instead of full.

```
R1# show ip ospf neighbor
Neighbor ID      Pri   State   Dead Time   Address        Interface
3.3.3.3          0     FULL/ - 00:00:35   10.1.1.13     GigabitEthernet0/0/1
2.2.2.2          0     FULL/ - 00:00:31   10.1.1.6      GigabitEthernet0/0/0
R1#
```

Verify Single-Area OSPFv2

Verify OSPF Neighbors (Cont.)

Two routers may not form an OSPFv2 adjacency if the following occurs:

- The subnet masks do not match, causing the routers to be on separate networks.
- The OSPFv2 Hello or Dead Timers do not match.
- The OSPFv2 Network Types do not match.
- There is a missing or incorrect OSPFv2 network command.

Verify Single-Area OSPFv2

Verify OSPF Protocol Settings

The **show ip protocols** command is a quick way to verify vital OSPF configuration information, as shown in the command output. This includes the OSPFv2 process ID, the router ID, interfaces explicitly configured to advertise OSPF routes, the neighbors the router is receiving updates from, and the default administrative distance, which is 110 for OSPF.

```
R1# show ip protocols
*** IP Routing is NSF aware ***
(output omitted)
Routing Protocol is "ospf 10"
  Outgoing update filter list for all interfaces is not set
  Incoming update filter list for all interfaces is not set
  Router ID 1.1.1.1
  Number of areas in this router is 1. 1 normal 0 stub 0 nssa
  Maximum path: 4
  Routing for Networks:
  Routing on Interfaces Configured Explicitly (Area 0):
    Loopback0
    GigabitEthernet0/0/1
    GigabitEthernet0/0/0
  Routing Information Sources:
    Gateway Distance Last Update
    3.3.3.3 110 00:09:30
    2.2.2.2 110 00:09:58
  Distance: (default is 110)
R1#
```

Verify Single-Area OSPFv2

Verify OSPF Process Information

The **show ip ospf** command can also be used to examine the OSPFv2 process ID and router ID, as shown in the command output. This command displays the OSPFv2 area information and the last time the SPF algorithm was executed.

```
R1# show ip ospf
Routing Process "ospf 10" with ID 1.1.1.1
Start time: 00:01:47.390, Time elapsed: 00:12:32.320
(output omitted)
Cisco NSF helper support enabled
Reference bandwidth unit is 10000 mbps
Area BACKBONE(0)
    Number of interfaces in this area is 3
    Area has no authentication
    SPF algorithm last executed 00:11:31.231 ago
    SPF algorithm executed 4 times
    Area ranges are
    Number of LSA 3. Checksum Sum 0x00E77E
    Number of opaque link LSA 0. Checksum Sum
0x000000
    Number of DCbitless LSA 0 Number of
indication LSA 0
    Number of DoNotAge LSA 0 Flood list length 0
R1#
```

Verify Single-Area OSPFv2

Verify OSPF Interface Settings

The **show ip ospf interface** command provides a detailed list for every OSPFv2-enabled interface. Specify an interface to display the settings of just that interface. This command shows the process ID, the local router ID, the type of network, OSPF cost, DR and BDR information on multiaccess links (not shown), and adjacent neighbors.

```
R1# show ip ospf interface GigabitEthernet 0/0/0
GigabitEthernet0/0/0 is up, line protocol is up
  Internet Address 10.1.1.5/30, Area 0, Attached via Interface Enable
  Process ID 10, Router ID 1.1.1.1, Network Type POINT_TO_POINT, Cost: 10

<output omitted>

  Neighbor Count is 1, Adjacent neighbor count is 1
    Adjacent with neighbor 2.2.2.2
  Suppress hello for 0 neighbor(s)
R1#
```


Verify Single-Area OSPFv2

Verify OSPF Interface Settings (Cont.)

To get a quick summary of OSPFv2-enabled interfaces, use the **show ip ospf interface brief** command, as shown in the command output. This command is useful for seeing important information including:

- Interfaces are participating in OSPF
- Networks that are being advertised (IP Address/Mask)
- Cost of each link
- Network state
- Number of neighbors on each link

```
R1# show ip ospf interface brief
Interface          PID      Area      IP Address/Mask  Cost    State    Nbrs F/C
Lo0                10       10        0                10.10.1.1/24  10
Gi0/0/1           P2P      10        0                0/0
                  10       0         10.1.1.14/30    30
Gi0/0/0           P2P      10        0                10.1.1.5/30   10
                  P2P      10        0                1/1
R1#
```