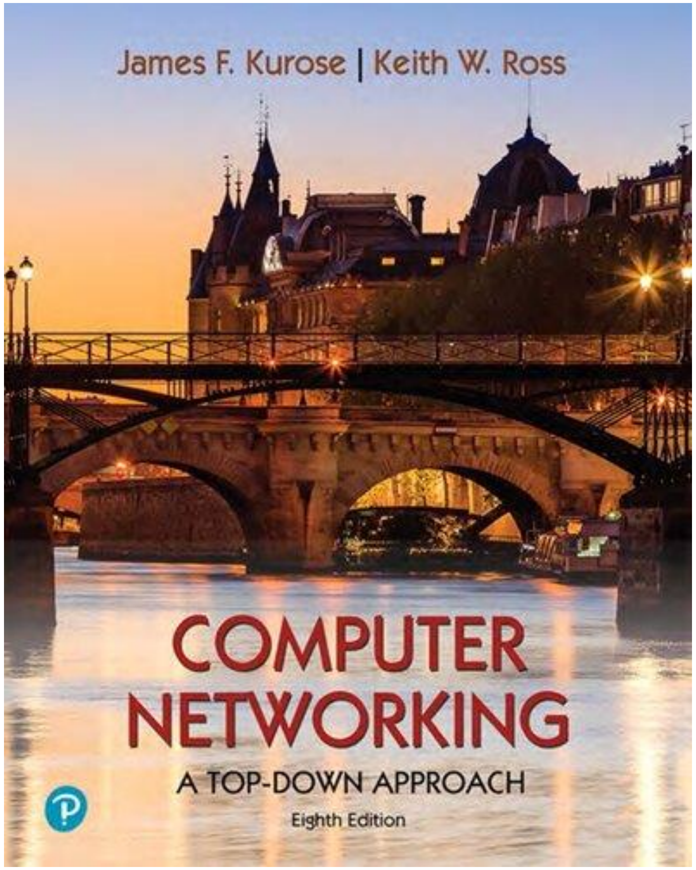
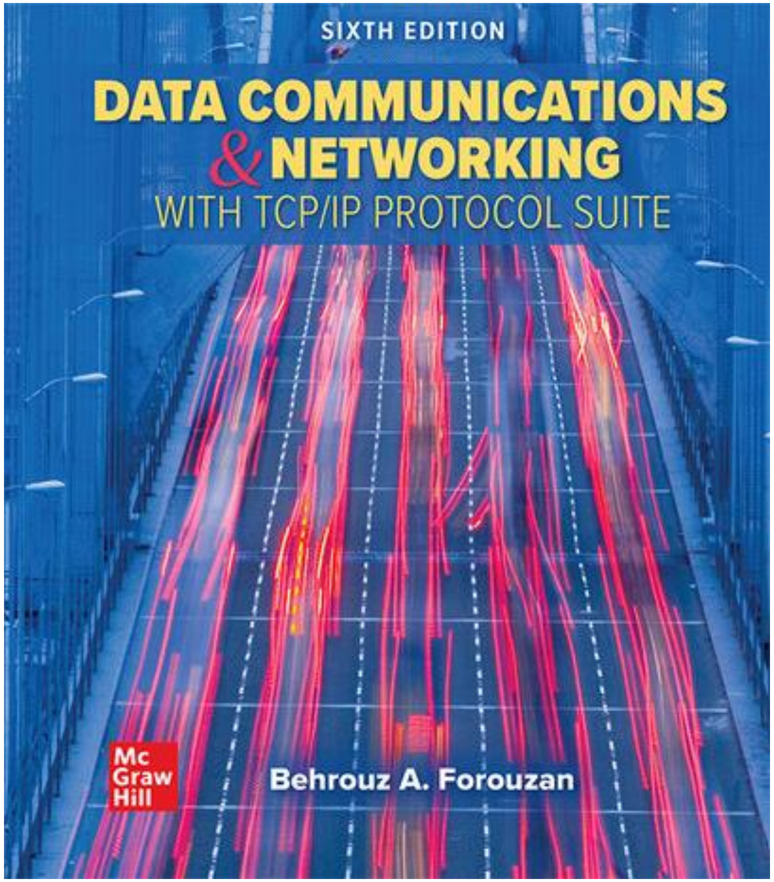
I3304
Network administration and security

Ahmad Fadlallah

Reference Textbooks





Outline

- Introduction
  - Introduction to the course
  - Recall Network Basics (I2208)
- Network Layer
  - Static Routing
  - Dynamic Routing
    - Dynamic Routing Algorithm
    - Dynamic Routing Protocols
  - NAT (Network Address Translation)
- Transport Layer
  - Function of the transport layer
  - UDP Protocol
  - TCP Protocol
    - Connection management
    - Flow control
    - Congestion control
- Application Layer
  - HTTP protocol
  - FTP protocol
  - Mail protocols
  - DNS
- Introduction to Security
  - Security services
  - Cryptography
  - Digital Signature
  - Principle of network security protocols

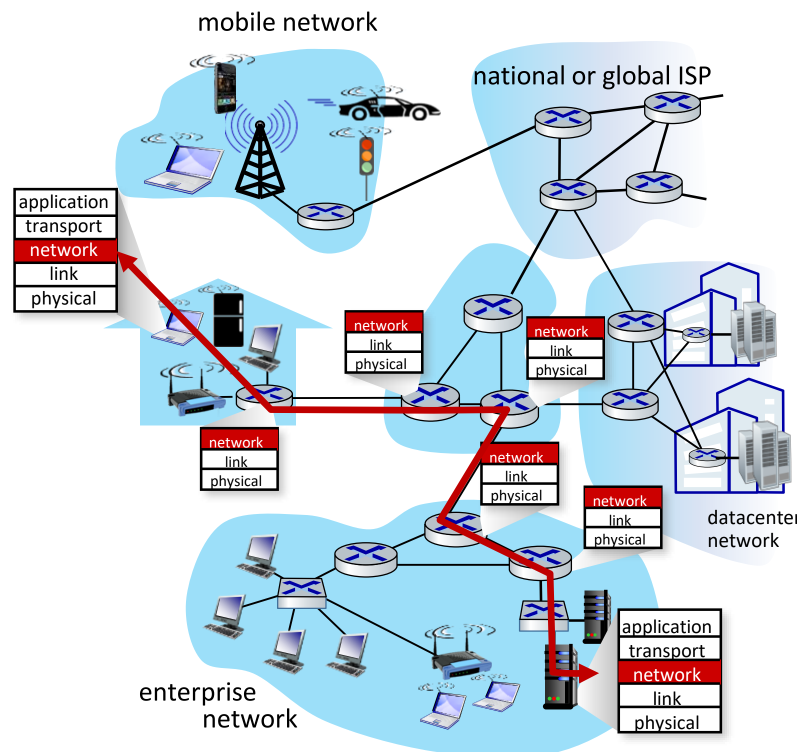
References

- The slides are based on the:
  - Cisco Networking Academy Program, Routing and Switching Essentials v6.0, Chapter 1: Routing Concepts
  - Jim Kurose, Keith Ross Slides for the Computer Networking: A Top-Down Approach, 8th edition, Pearson, 2020

Network Layer
Dynamic Routing

Routing protocols

- **<u>Routing protocol goal</u>**: 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", …



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

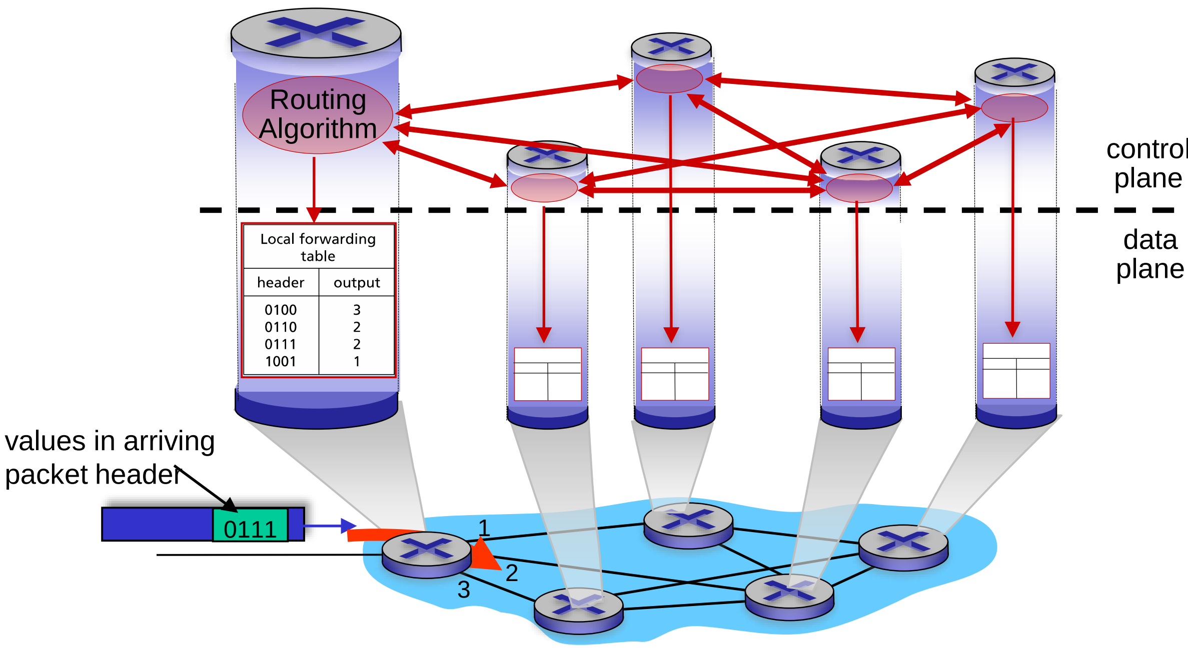
control plane

Two approaches to structuring network control plane:

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

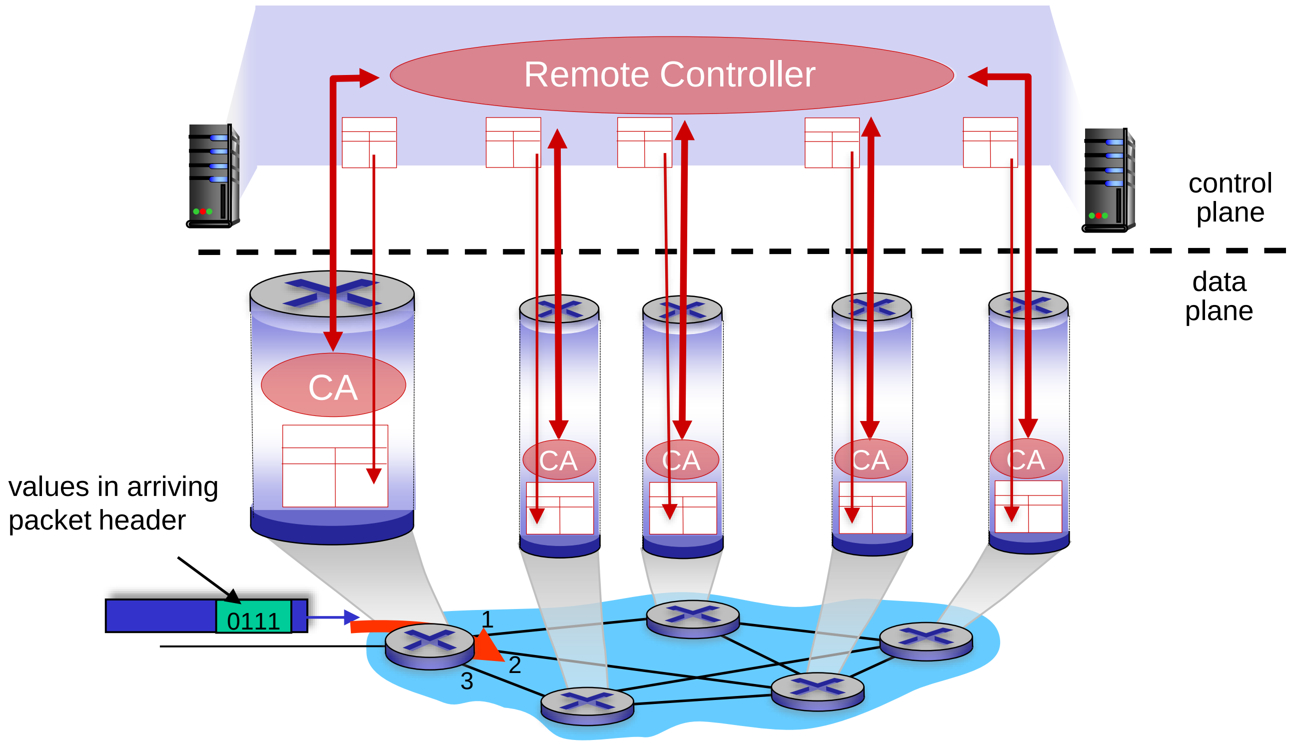
Per-router control plane

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



Software-Defined Networking (SDN)

Remote controller computes, installs forwarding tables in routers



Purpose of Dynamic Routing Protocols

- **Routing Protocols** are used to facilitate the exchange of routing information between routers.
- The purpose of dynamic routing protocols includes:
  - ⊙**Discovery of remote networks**
  - ⊙Maintaining **up-to-date routing information**
  - ⊙Choosing **the best path** to destination networks
  - ⊙Ability to **find a new best path** if the current path is no longer available

Dynamic verses Static Routing

Static Routing Scorecard

Static Routing Advantages and Disadvantages

Advantages	Disadvantages
Easy to implement in a small network.	Suitable only for simple topologies or for special purposes such as a default static route. Configuration complexity increases dramatically as network grows.
Very secure. No advertisements are sent as compared to dynamic routing protocols.	
Route to destination is always the same.	Manual intervention required to re-route traffic.
No routing algorithm or update mechanism required; therefore, extra resources (CPU or RAM) are not required.	

Dynamic Routing Scorecard

Dynamic Routing Advantages and Disadvantages

Advantages	Disadvantages
Suitable in all topologies where multiple routers are required.	Can be more complex to implement.
Generally independent of the network size.	Less secure. Additional configuration settings are required to secure.
Automatically adapts topology to reroute traffic if possible.	Route depends on the current topology.
	Requires additional CPU, RAM, and link bandwidth.

Dynamic Routing Protocols Components

Data structures

- Routing protocols typically use tables or databases for its operations.
- This information is kept in RAM.

Routing protocol messages

- Routing protocols use various types of messages to *discover neighboring routers*, *exchange routing information*, and other tasks to learn and maintain accurate information about the network.

Algorithm

- Routing protocols use algorithms for facilitating routing information for best path determination.

Dynamic Routing Protocol Operation

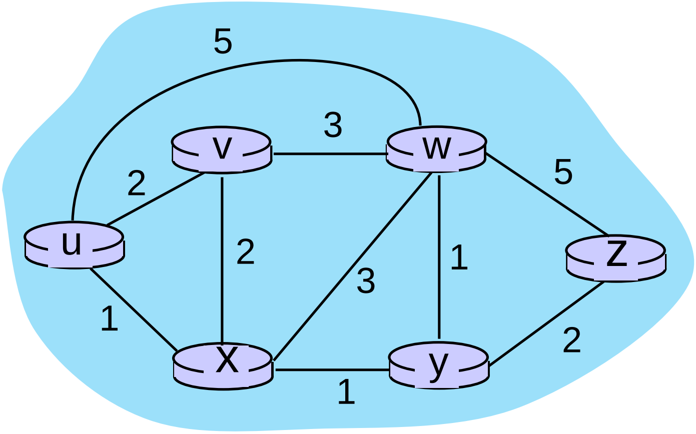
The router sends and receives routing messages on its interfaces.

The router shares routing messages and routing information with other routers that are using the same routing protocol.

Routers exchange routing information to learn about remote networks.

When a router detects a topology change the routing protocol can advertise this change to other routers.

Graph abstraction: link costs



graph: $G = (N,E)$

$c_{a,b}$: 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

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

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

How fast do routes change?

static: routes change slowly over time

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

decentralized: iterative process of computation, exchange of info with neighbors
- routers initially only know link costs to attached neighbors
- "distance vector" algorithms

global or decentralized information?

Dynamic Routing Algorithms
Link-State Algorithms

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

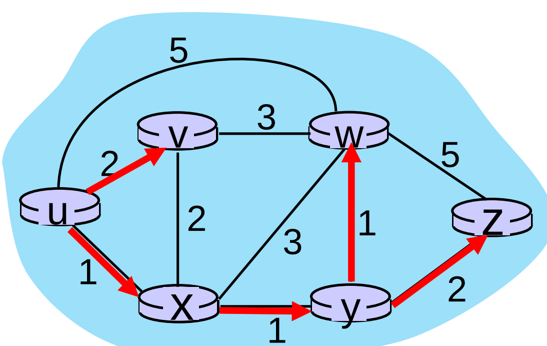
Dijkstra's link-state routing algorithm

Initialization:
  $N' = \{u\}$ /* compute least cost path from u to all other nodes */
  for all nodes *v*
    if *v* adjacent to *u* /* *u* initially knows direct-path-cost only to direct neighbors */
      then $D(v) = c_{u,v}$ /* but may not be *minimum* cost! */
    else $D(v) = \infty$

Loop
  find *w* not in N' such that $D(w)$ is a minimum
  add *w* to N'
  update $D(v)$ for all *v* adjacent to *w* and not in N' :
    $\mathbf{D(v) = min\ (\ D(v),\ \ D(w) + c_{w,v}\ )}$
  /* new least-path-cost to *v* is either old least-cost-path to *v* or known
  least-cost-path to *w* plus direct-cost from *w* to *v* */
until all nodes in N'

Dijkstra’s algorithm: an example

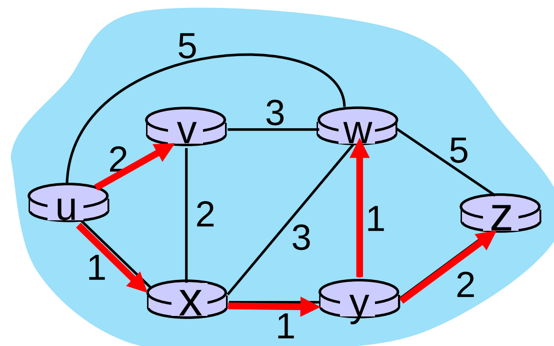
		v	w	x	y	z
Step	N'	D(v),p(v)	D(w),p(w)	D(x),p(x)	D(y),p(y)	D(z),p(z)
0	u	2,u	5,u	1,u	∞	∞
1	ux	2,u	4,x		2,x	∞
2	uxy	2,u	3,y			4,y
3	uxyv		3,y			4,y
4	uxyvw					4,y
5	uxyvwz					



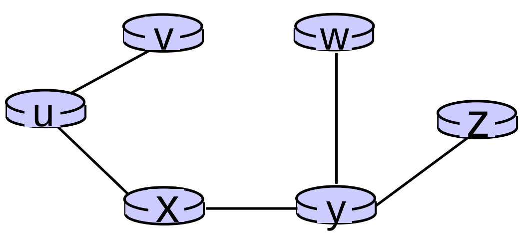
Initialization (step 0): For all a: if *a* adjacent to then $D(a) = c_{u,a}$

find *a* not in *N'* such that *D(a)* is a minimum
add *a* to *N'*
update *D(b)* for all *b* adjacent to *a* and not in *N'* :
$D(b) = \min (D(b), D(a) + c_{a,b})$

Dijkstra's algorithm: an example



resulting least-cost-path tree from u:



resulting forwarding table in u:

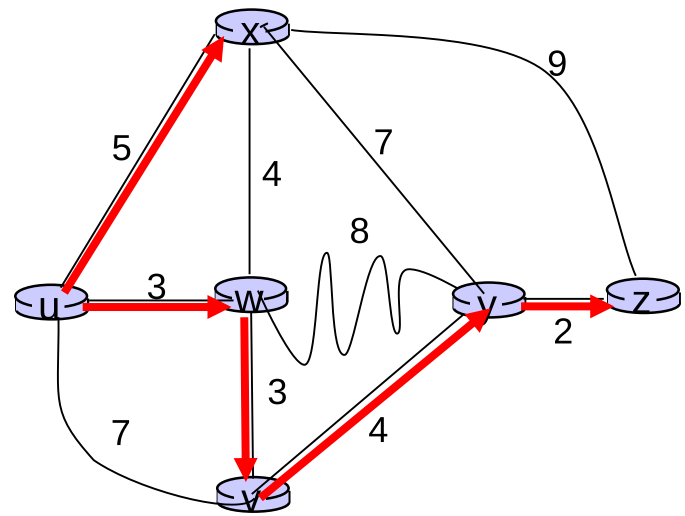
destination	outgoing link
v	(u,v)
x	(u,x)
y	(u,x)
w	(u,x)
x	(u,x)

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			11,w	14,x
3	uwxv				10,v	14,x
4	uwxvy					12,y
5	uwxvyz					

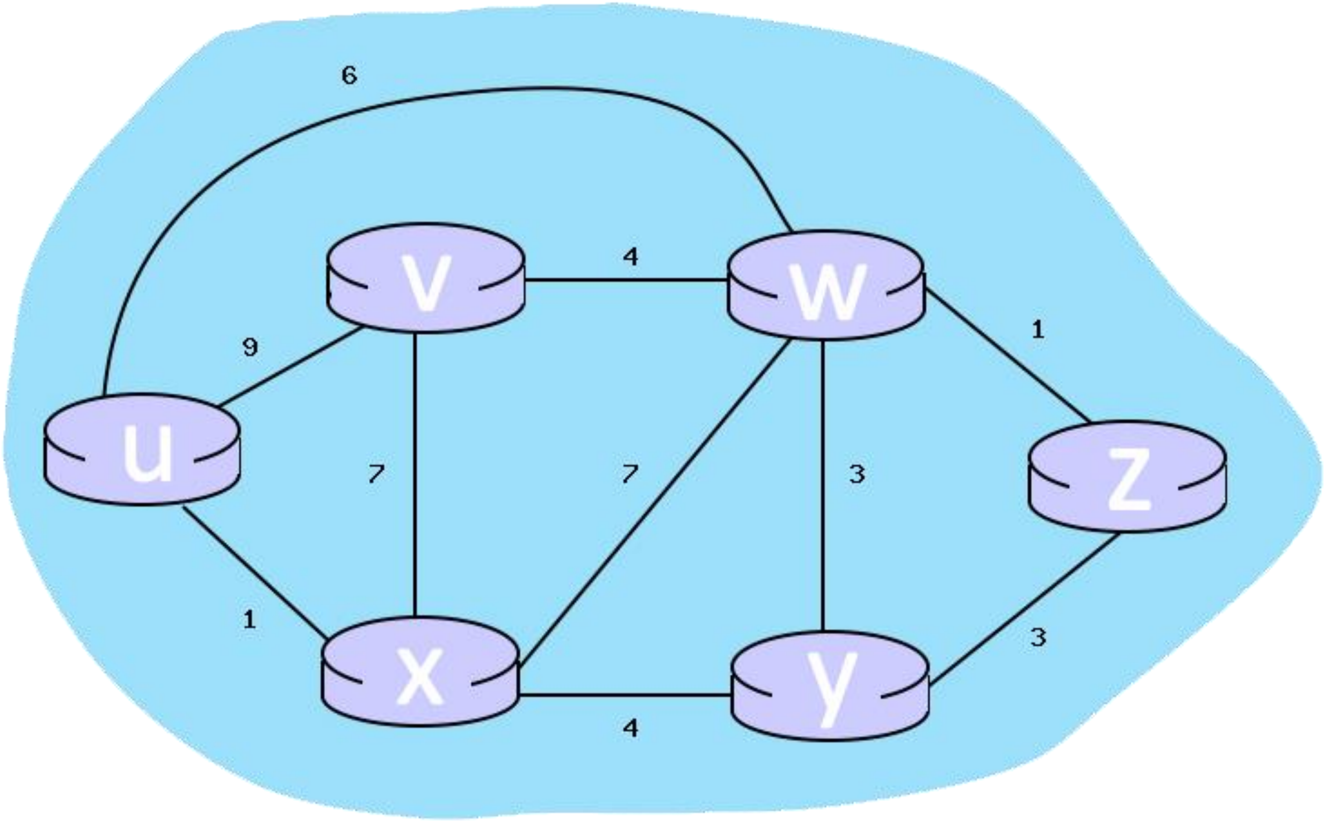


notes:

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

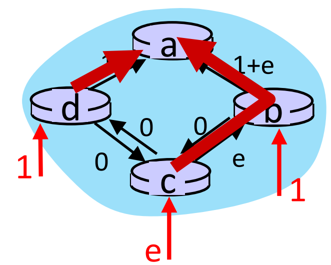
Exercise

- Using Dijkstra's algorithm, find the least cost path from source node U to all other destinations

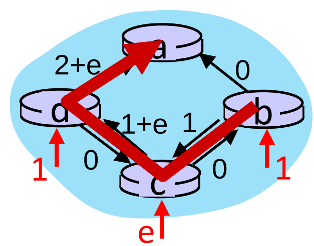


Dijkstra’s algorithm: oscillations possible

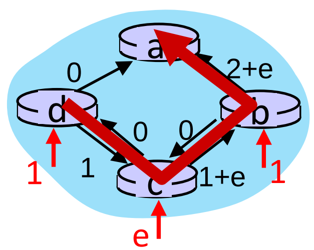
- When link costs depend on traffic volume, route oscillations possible
- Sample scenario:
  - Routing to destination a, traffic entering at d, c, e with rates 1, e (<1), 1
  - Link costs are directional, and volume-dependent



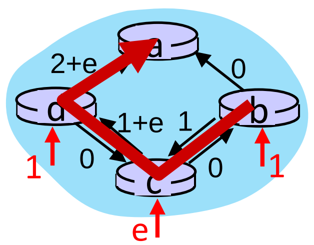
initially



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

Dynamic Routing Algorithms
Distance-Vector Algorithms

Distance vector algorithm

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

Bellman-Ford equation

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

Then:

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

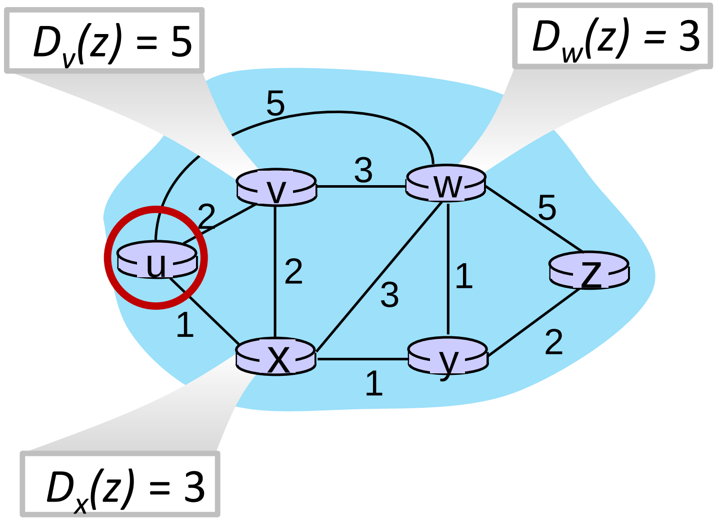
v's estimated least-cost-path cost to y

min taken over all neighbors v of x

direct cost of link from x to v

Bellman-Ford Example

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



Bellman-Ford equation says:

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

node achieving minimum (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
- When *x* receives new DV estimate from any neighbor, it updates its own DV using B-F equation:

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

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

Distance vector algorithm:

each node:

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

↓

recompute DV estimates using DV received from neighbor

↓

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

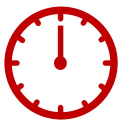
Iterative, Asynchronous: each local iteration caused by:

- 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

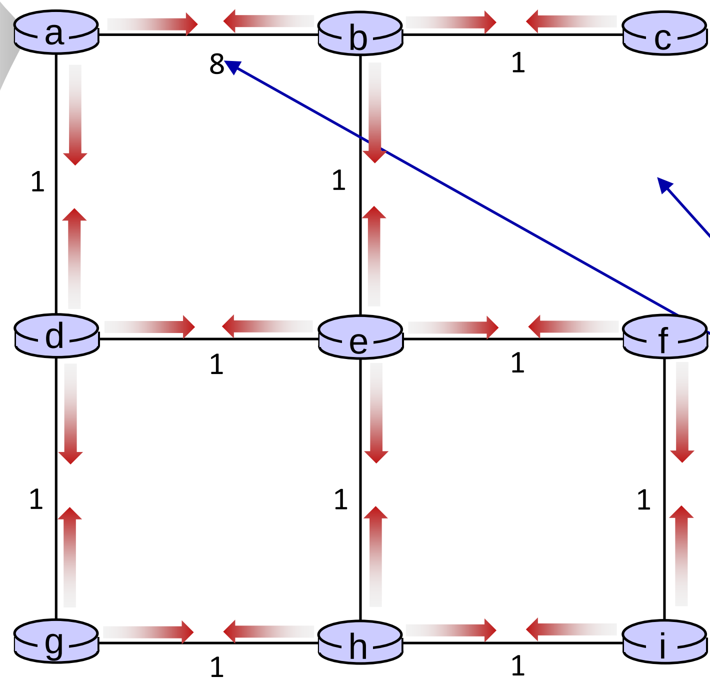


t=0

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

DV in a:

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



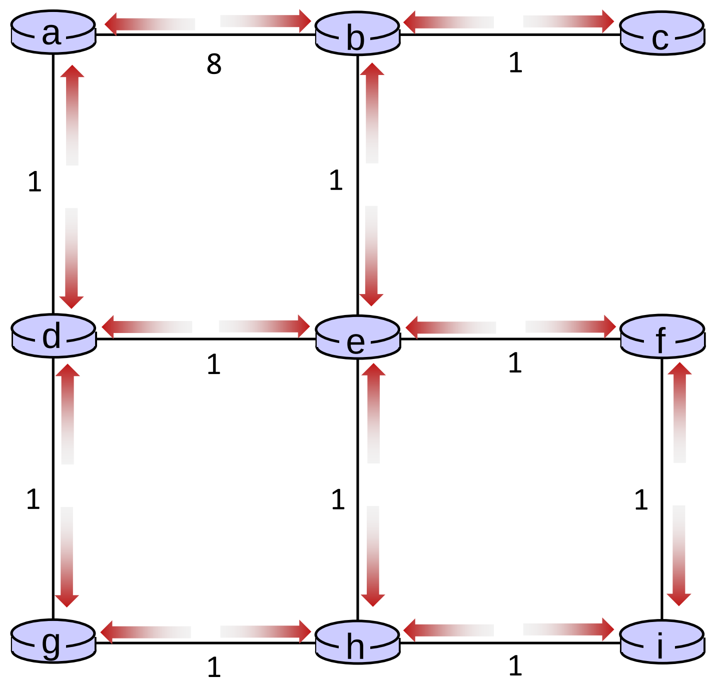
A few asymmetries:
- missing link
- larger cost

Distance vector example: iteration


t=1

All nodes:

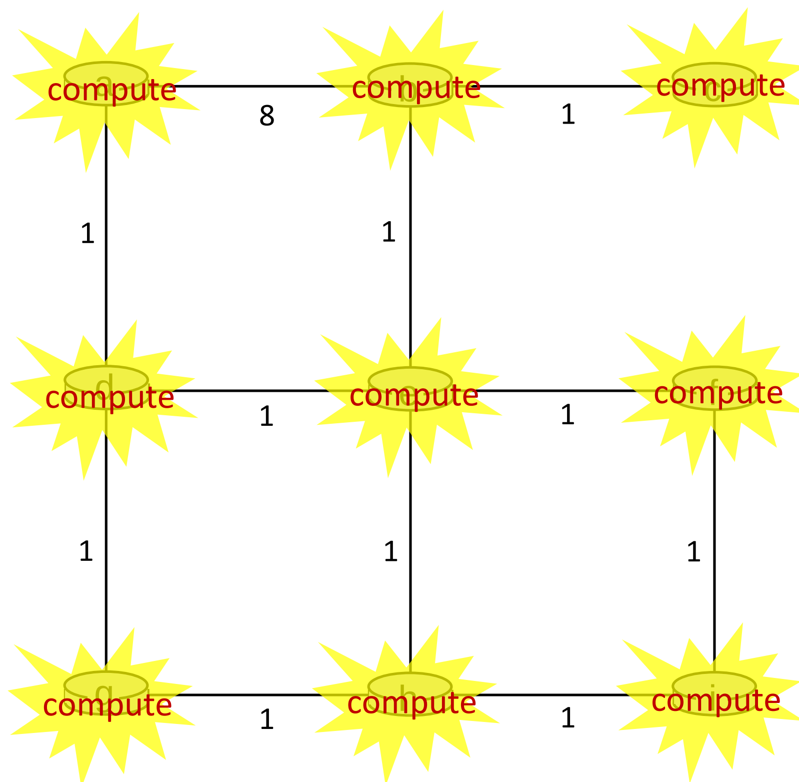
- receive distance vectors from neighbors
- compute their new local distance vector
- send their new local distance vector to neighbors



Distance vector example: iteration

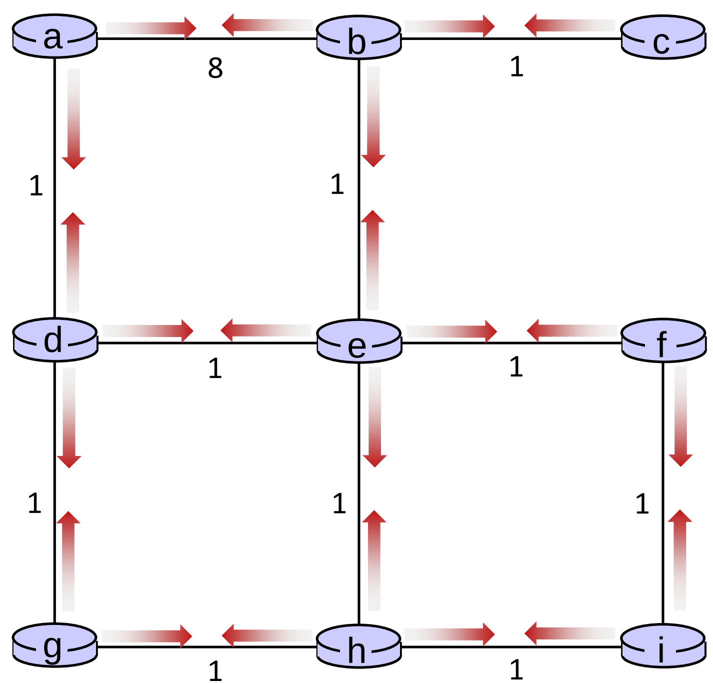
t=1

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

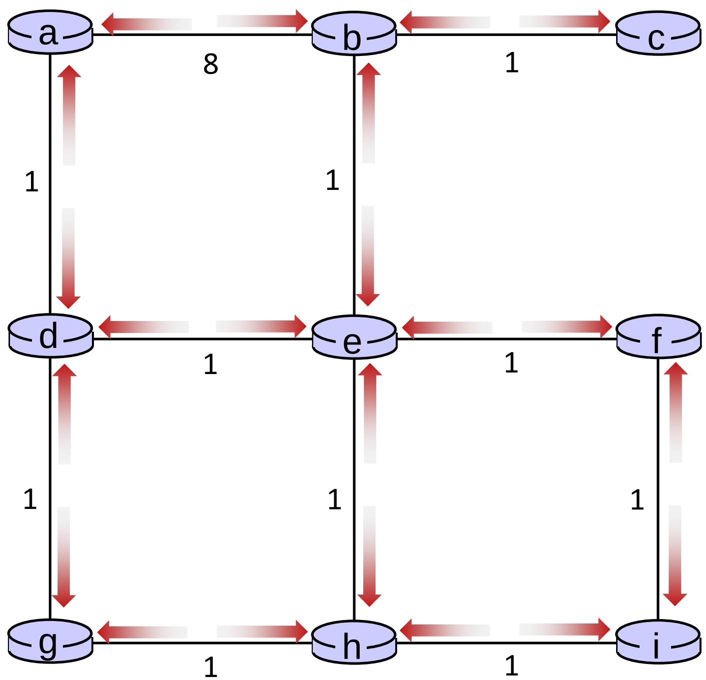
All nodes:
- receive distance vectors from neighbors
- compute their new local distance vector
- send their new local distance vector to neighbors



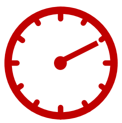
Distance vector example: iteration

t=2

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



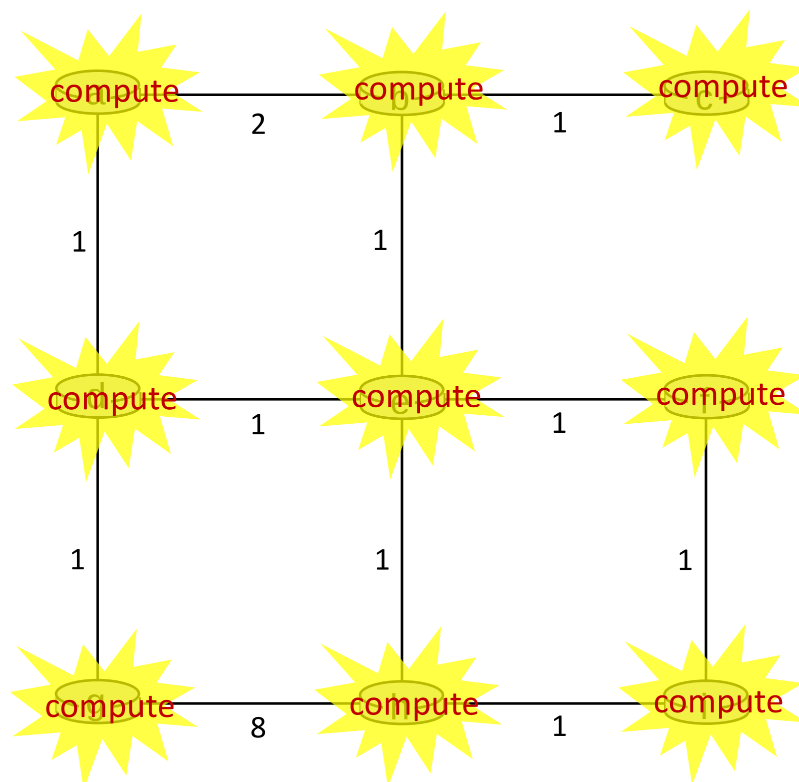
Distance vector example: iteration



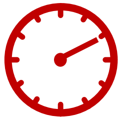
t=2

All nodes:

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



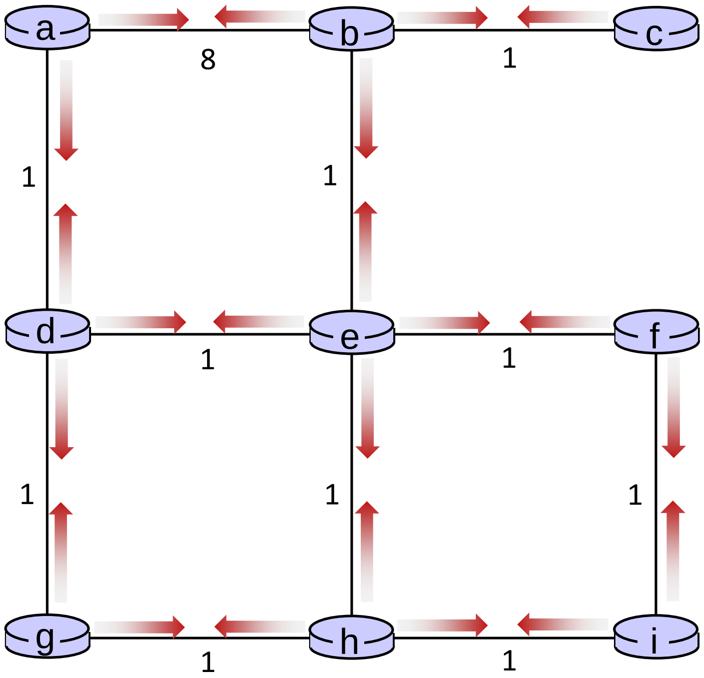
Distance vector example: iteration



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

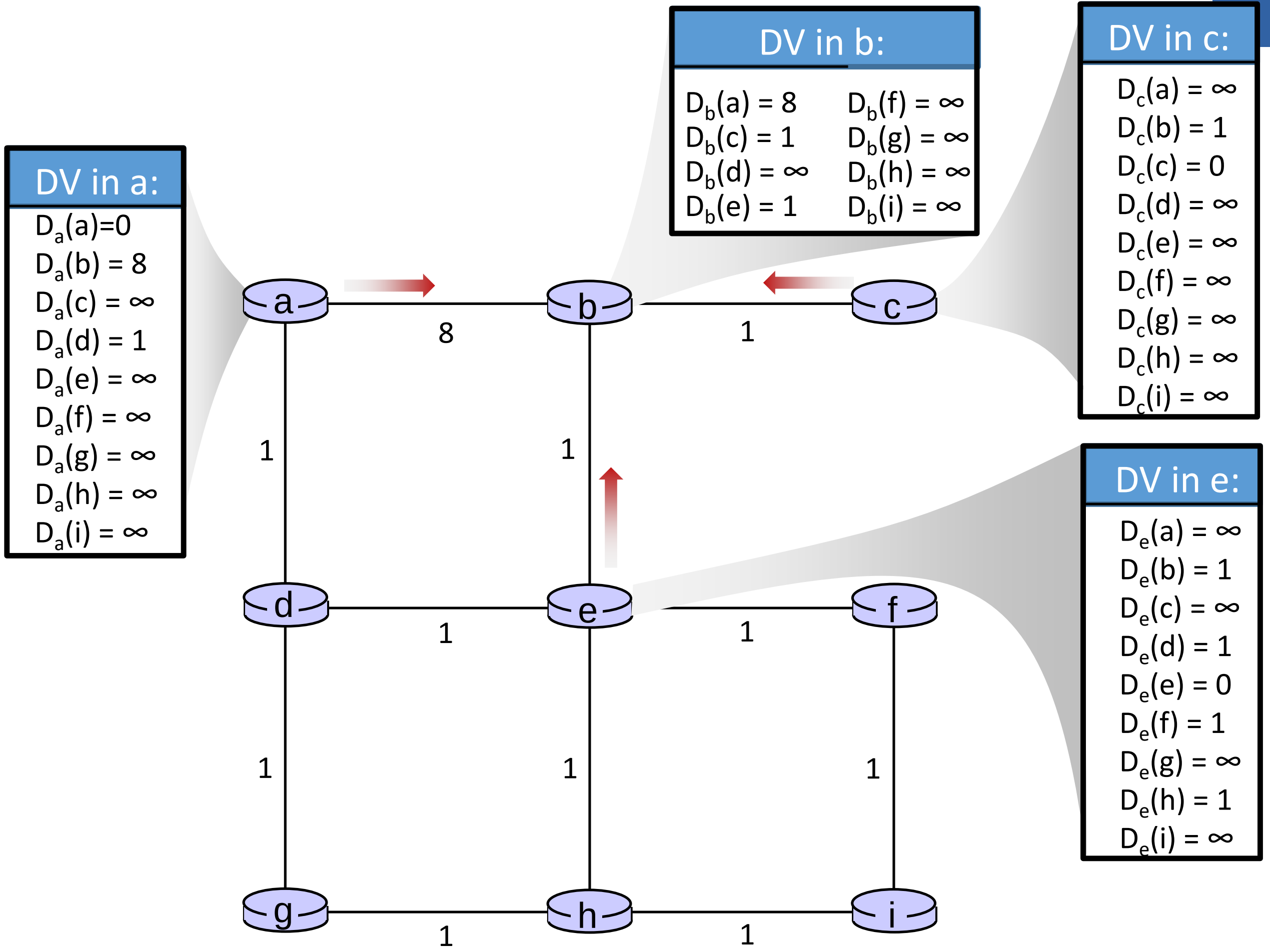
.... and so on

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

Distance vector example: computation

t=1

- b receives DVs from a, c, e



Distance vector example: computation

t=1

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

DV in a:

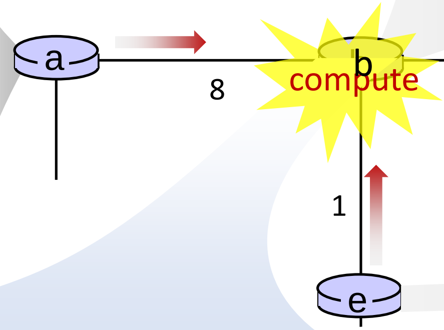
$D_a(a)=0$
$D_a(b) = 8$
$D_a(c) = \infty$
$D_a(d) = 1$
$D_a(e) = \infty$
$D_a(f) = \infty$
$D_a(g) = \infty$
$D_a(h) = \infty$
$D_a(i) = \infty$

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:

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

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

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

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

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

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

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

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

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

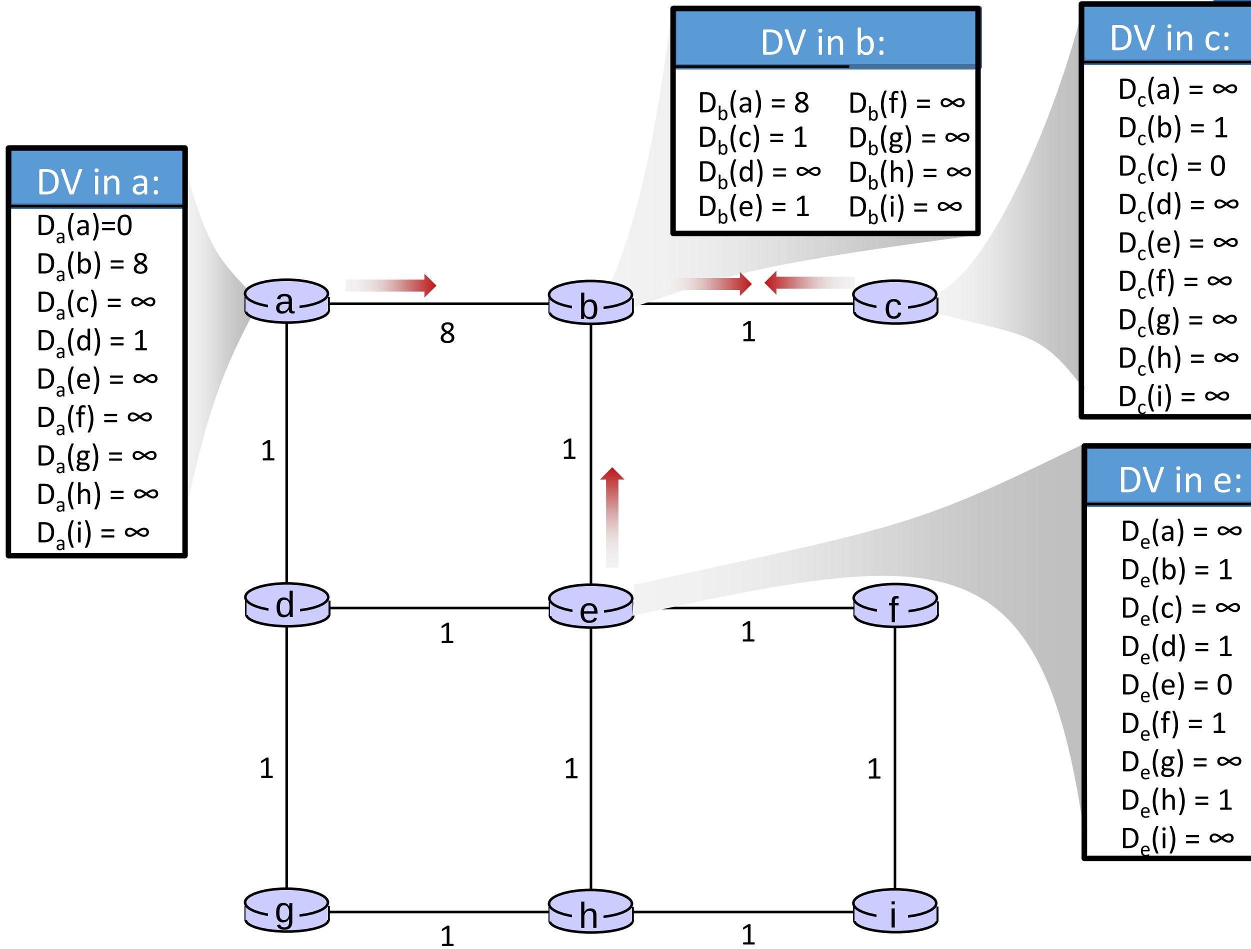
DV in b:

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

Distance vector example: computation

t=1

- c receives DVs from b



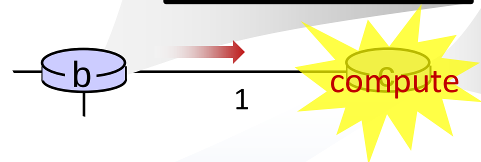
Distance vector example: computation

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:

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



t=1

- c receives DVs from b computes:

$D_c(a) = \min\{c_{c,b}+D_b(a)\} = 1 + 8 = 9$

$D_c(b) = \min\{c_{c,b}+D_b(b)\} = 1 + 0 = 1$

$D_c(d) = \min\{c_{c,b}+D_b(d)\} = 1+ \infty = \infty$

$D_c(e) = \min\{c_{c,b}+D_b(e)\} = 1 + 1 = 2$

$D_c(f) = \min\{c_{c,b}+D_b(f)\} = 1+ \infty = \infty$

$D_c(g) = \min\{c_{c,b}+D_b(g)\} = 1+ \infty = \infty$

$D_c(h) = \min\{c_{bc,b}+D_b(h)\} = 1+ \infty = \infty$

$D_c(i) = \min\{c_{c,b}+D_b(i)\} = 1+ \infty = \infty$

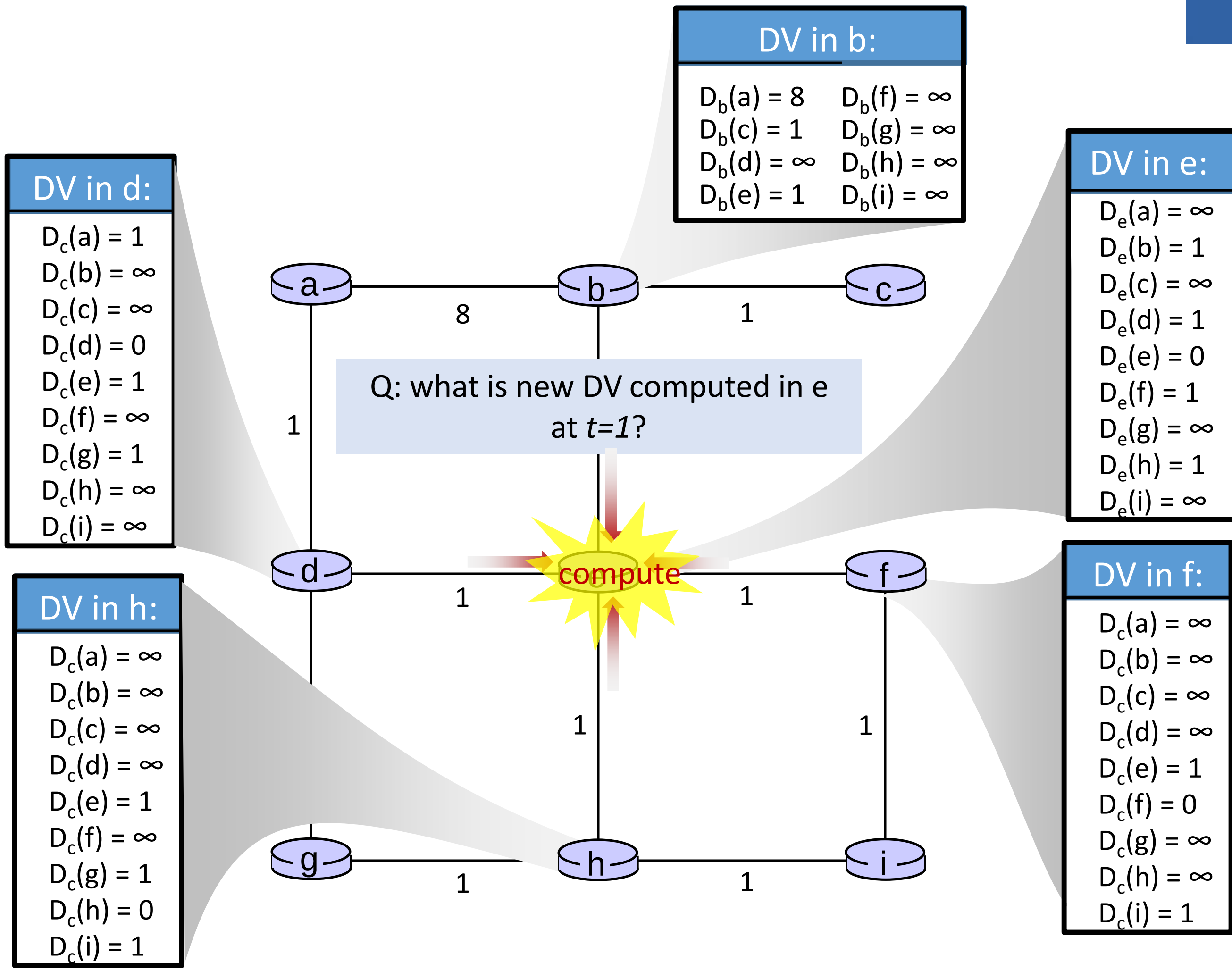
DV in c:

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

Distance vector example: computation

t=1

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



Distance vector: state information diffusion

Iterative communication, computation steps diffuses information through network:

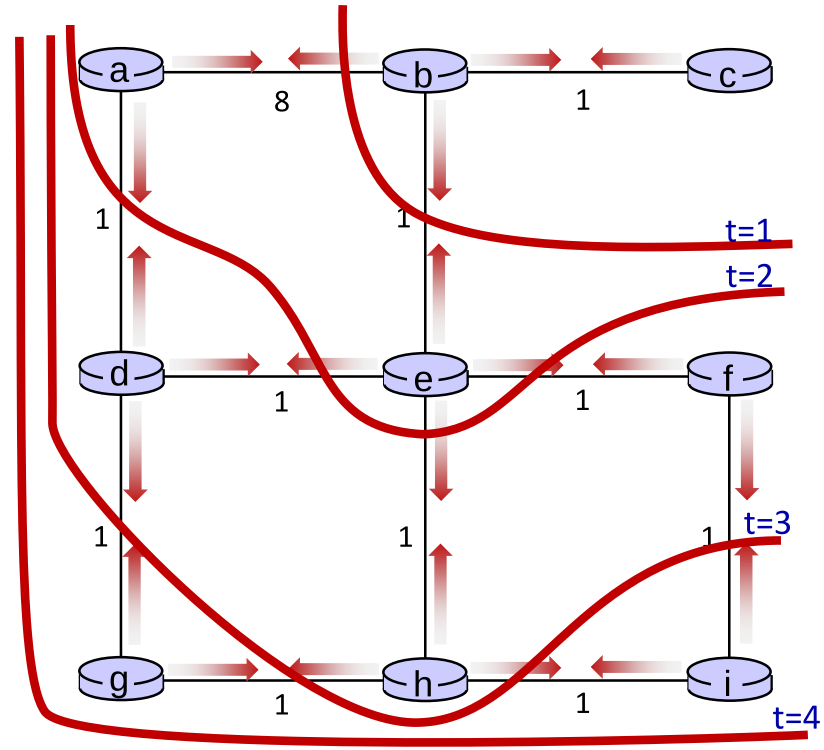
t=0 c's state at t=0 is at c only

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

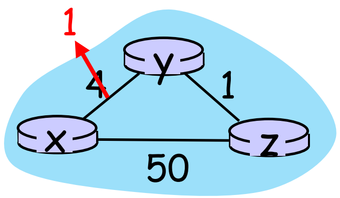
t=2 c's state at t=0 may now influence distance vector computations up to **2** hops away, i.e., at b and now at a, e as well

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

"good news travels fast"

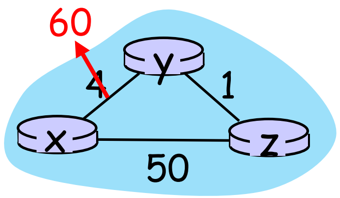
t_0 : *y* detects link-cost change, updates its DV, informs its neighbors.

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

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

Distance vector: link cost changes

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



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

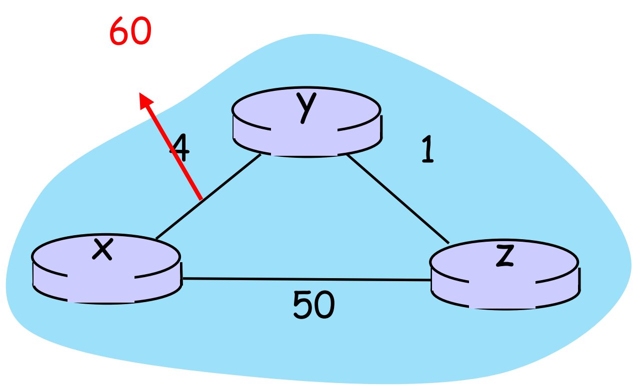
  ...

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

Distance-Vector: Adding Poisoned Reverse

- **Poisoned reverse** is a technique used to avoid some looping scenario in DV algorithms.
- How it works: if z routes through y to get to destination x, then z will advertise to y that its distance to x is infinity

  - z will advertise to y that $D_z(x) = \infty$ (even though z knows $D_z(x) = 5$ in truth).
  - z will continue telling this little "white lie" to y as long as it routes to x via y.
  - Since y believes that z has no path to x, y will never attempt to route to x via z, as long as z continues to route to x via y (and lies about doing so).



Distance vector: another example

$D_x()$

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

$D_y()$

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

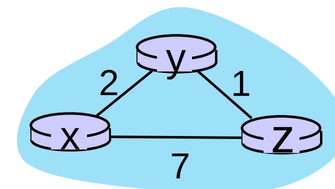
$D_z()$

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

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

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

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



time

Distance vector: another example

$D_x()$

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

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

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

$D_y()$

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

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

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

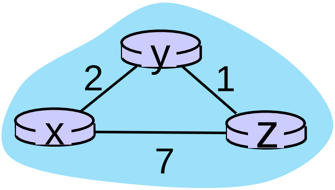
$D_z()$

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

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

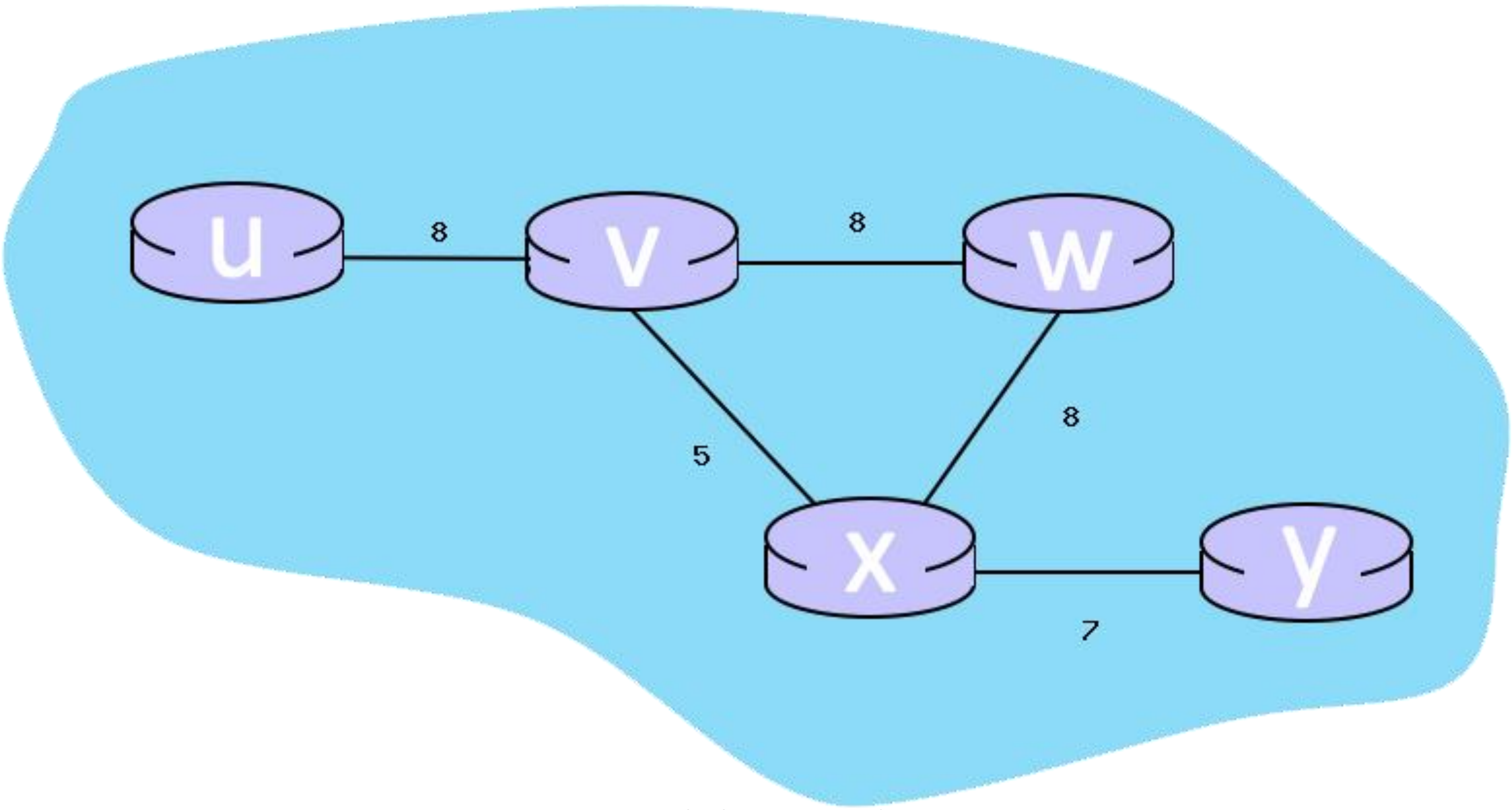
from \ cost to	x	y	z
x	0	2	3
y	2	0	1
z	3	1	0

time



Exercise

- When the algorithm converges, what are the distance vectors from router 'V' to all routers?



Dynamic Routing Protocols

Making routing scalable

Our routing study thus far - idealized

- All routers identical/ executing the same routing algorithm
- 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
  - Different Routing Algorithms
  - Hiding aspects of network's internal organization

Internet approach to scalable routing

- Aggregate routers into regions known as "Autonomous Systems" (AS) (a.k.a. "domains")
- Each AS consisting of a group of routers that are under the same administrative control.
- One ISP network ➔ one or more AS
- An autonomous system is identified by its globally unique Autonomous System Number (ASN) [RFC 1930].
- AS numbers, like IP addresses, are assigned by ICANN regional registries

Internet approach to scalable routing

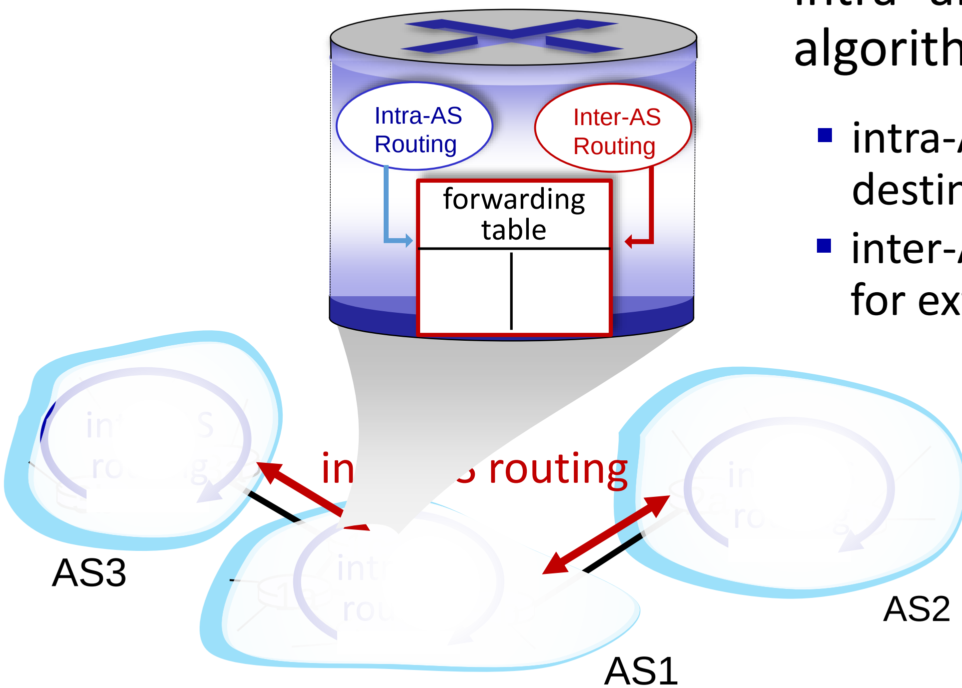
Intra-AS

- Routing *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

- 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

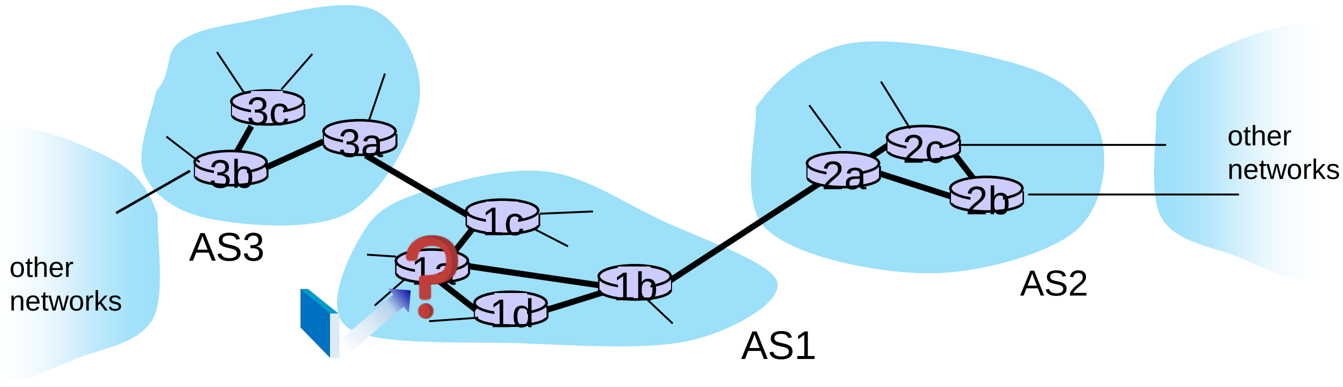
- intra-AS routing determine entries for destinations within AS
- inter-AS & intra-AS determine entries for external destinations

Intra-AS routing: a role in intra-domain forwarding

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

AS1 inter-domain routing must:

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

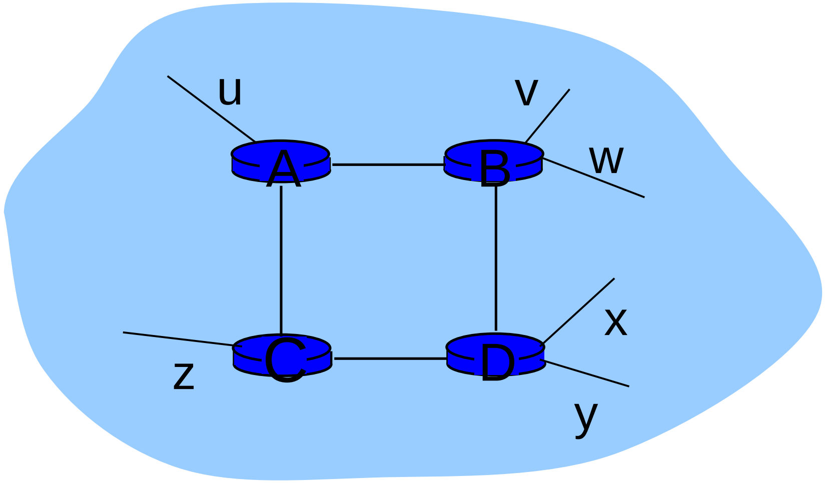


Intra-AS routing: routing within an AS

- 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

RIP (Routing Information Protocol)

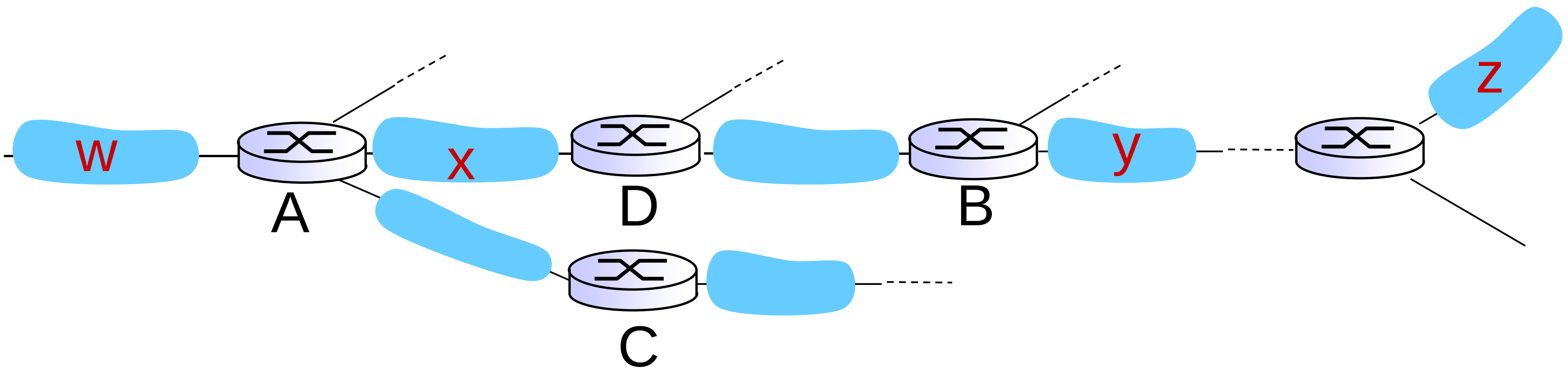
- Included in BSD-UNIX distribution in 1982
- Distance vector algorithm
  - distance metric: # hops (max = 15 hops), each link has cost 1
  - DVs exchanged with neighbors every 30 sec in response message (aka advertisement)
  - Each advertisement: list of up to 25 destination *subnets (in IP addressing sense)*



from router A to destination *subnets:*

subnet	hops
u	1
v	2
w	2
x	3
y	3
z	2

RIP: example



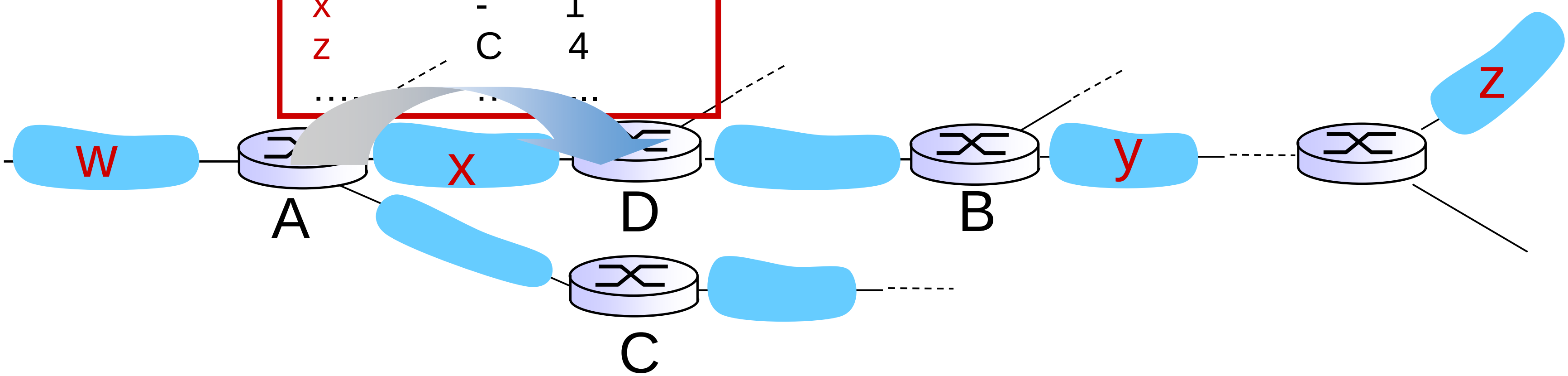
routing table in router D

destination subnet	next router	# hops to dest
w	A	2
y	B	2
z	B	7
x	--	1
….	….	….

RIP: example

A-to-D advertisement

dest	next	hops
w	-	1
x	-	1
z	C	4
….	…	..



routing table in router D

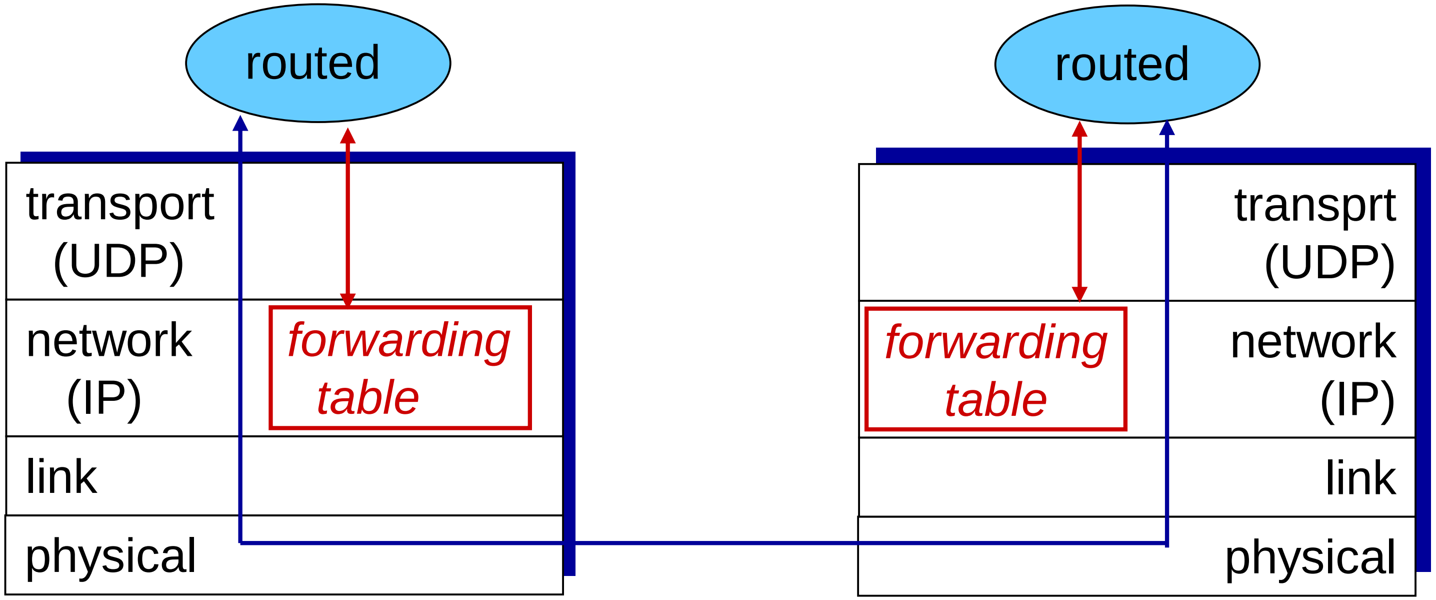
destination subnet	next router	# hops to dest
w	A	2
y	B	2
z	~~B~~ A	~~7~~ 5
x	--	1
….	….	….

RIP: link failure, recovery

- If no advertisement heard after 180 sec → neighbor/link declared dead
  - Routes via neighbor invalidated
  - New advertisements sent to neighbors
  - Neighbors in turn send out new advertisements (if tables changed)
  - Link failure info quickly (?) propagates to entire net
  - Poison reverse used to prevent ping-pong loops (infinite distance = 16 hops)

RIP table processing

- RIP routing tables managed by application-level process called *route-d* (daemon)
- Advertisements sent in UDP packets, periodically repeated

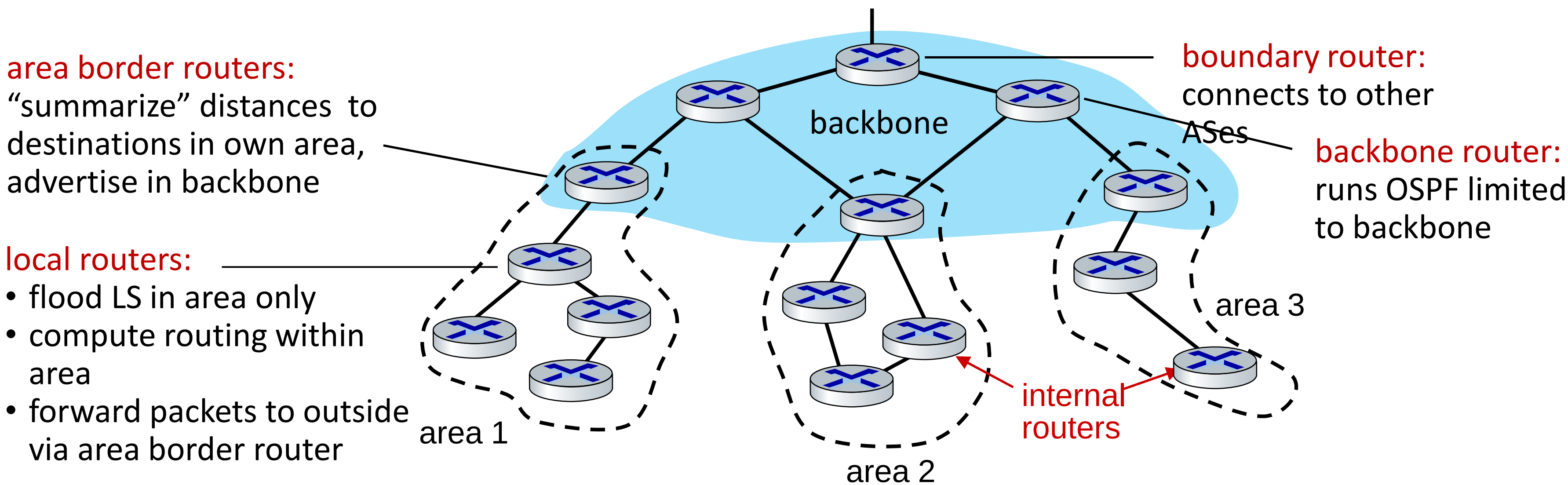


OSPF (Open Shortest Path First) routing

- "open": publicly available
  - OSPFv2, defined in RFC2328
- OSPF is a **link-state protocol** that uses flooding of link-state information and a Dijkstra's least-cost path algorithm.
  - 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



Dynamic Routing Protocols routing among ISPs: BGP

Introduction

- Building the forwarding table for a router (within an AS)
  - For destinations that are within the same AS, the entries in the router's forwarding table are determined by the AS's intra-AS routing protocol
  - What about destinations that are outside of the AS?

This is precisely where the Border Gateway Protocol (BGP) comes to the rescue.

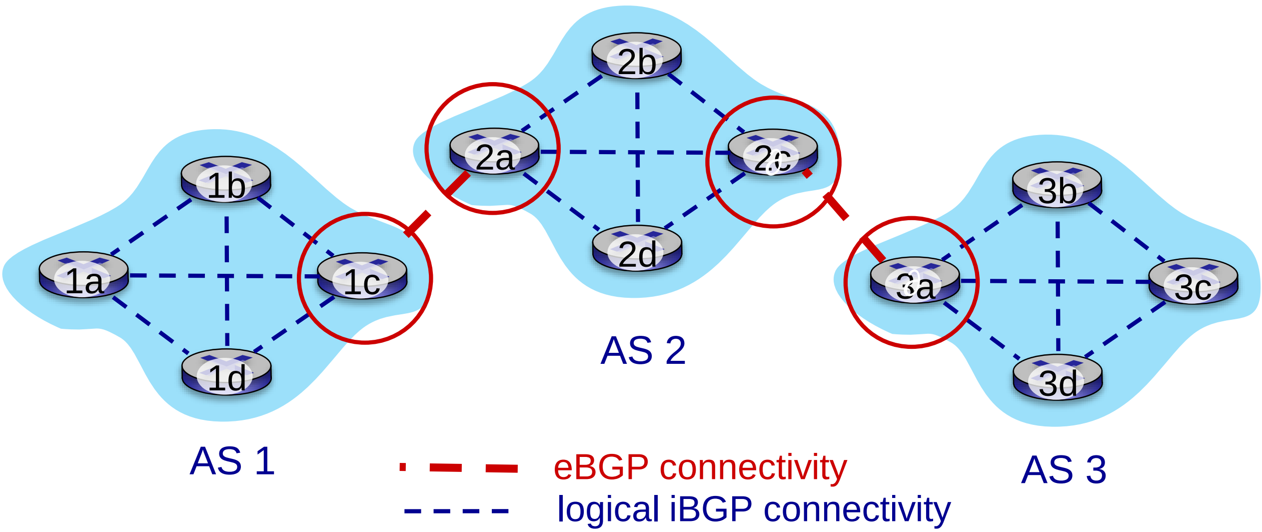
Internet inter-AS routing: BGP

- BGP (Border Gateway Protocol): the *de facto* inter-domain routing protocol
  - The most important of all Internet protocol (in contest with IP)
  - "**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"
- In BGP, packets are not routed to a specific destination address, but instead to **CIDRized prefixes**, with each prefix representing a subnet or a collection of subnets.
  - Example: a destination may take the form 138.16.68/22, which for this example includes 1,024 IP addresses.
  - A router's forwarding table will have entries of the form (*x*, *I*), where *x* is a prefix (such as 138.16.68/22) and *I* is an interface number for one of the router's interfaces.

Internet inter-AS routing: BGP

- BGP provides each AS a means to:
  - Obtain subnet reachability information from neighboring Ases
    - The role of **eBGP (external BGP)**
    - BGP allows each subnet to advertise its existence to the rest of the Internet
    - *A subnet screams, “I exist and I am here,” and BGP makes sure that all the routers in the Internet know about this subnet.*
  - Propagate reachability information to all AS-internal routers
    - The role of **iBGP (internal BGP)**
  - Determine the “best” routes to the prefixes.
    - The router will locally run a BGP route-selection procedure
    - determine “good” routes to other networks based on reachability information and policy

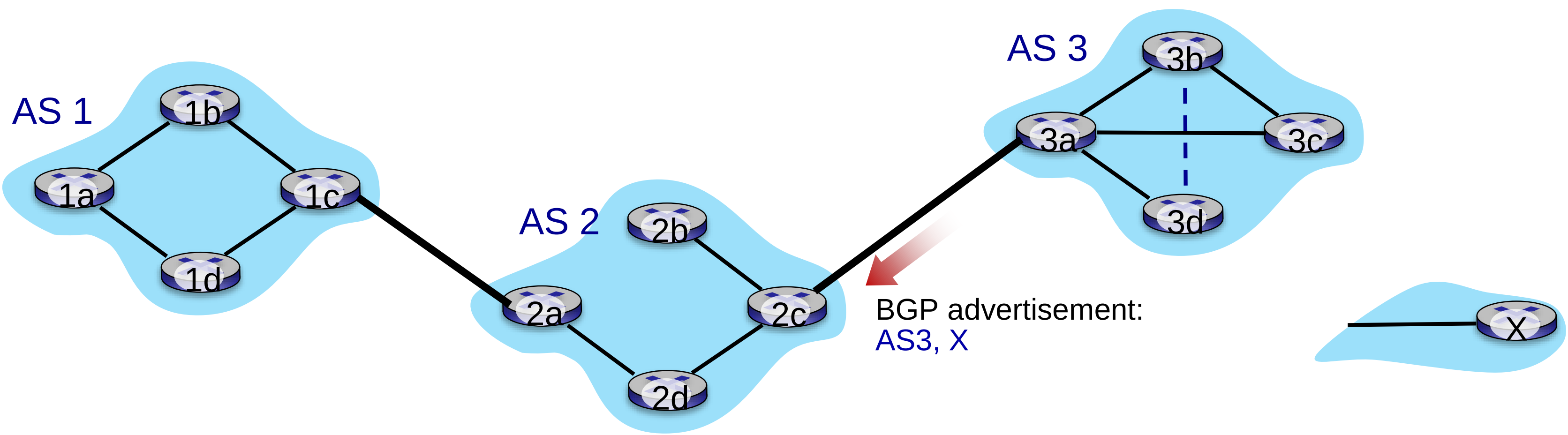
eBGP, iBGP connections



gateway routers run both eBGP and iBGP protocols

BGP basics

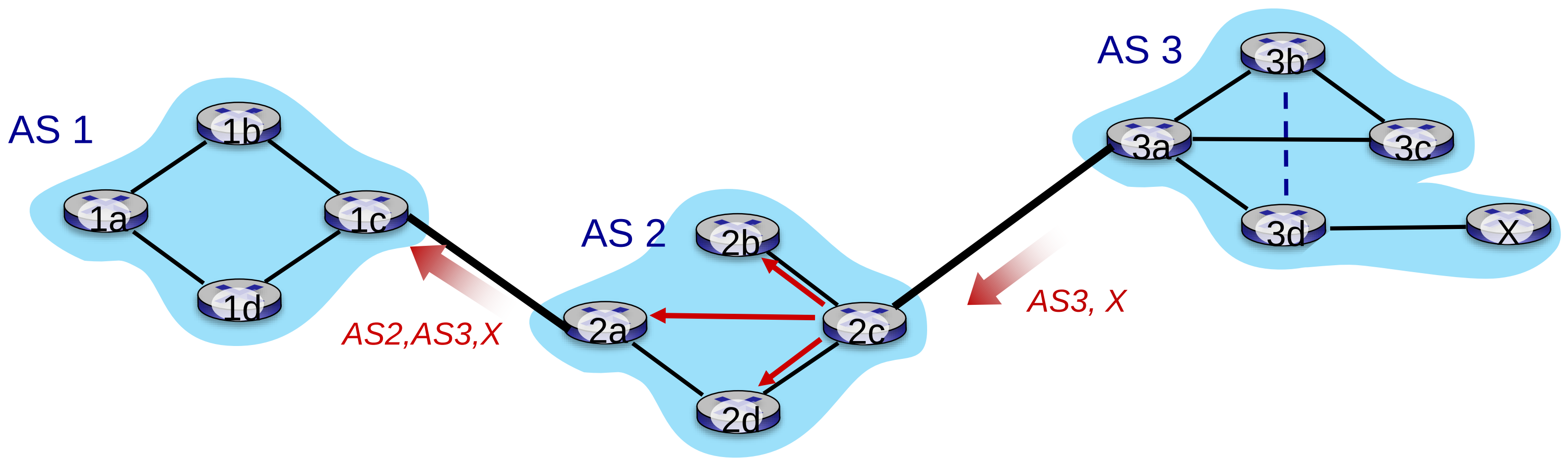
- BGP session: two BGP routers ("peers") exchange BGP messages over <u>semi-permanent</u> TCP connection (port 179):
  - ⊙Advertising paths to different destination network prefixes (BGP is a **"path vector"** protocol)
- When AS3 gateway 3a advertises path AS3,X to AS2 gateway 2c:
  - ⊙AS3 promises to AS2 it will forward datagrams towards X



Path attributes and BGP routes

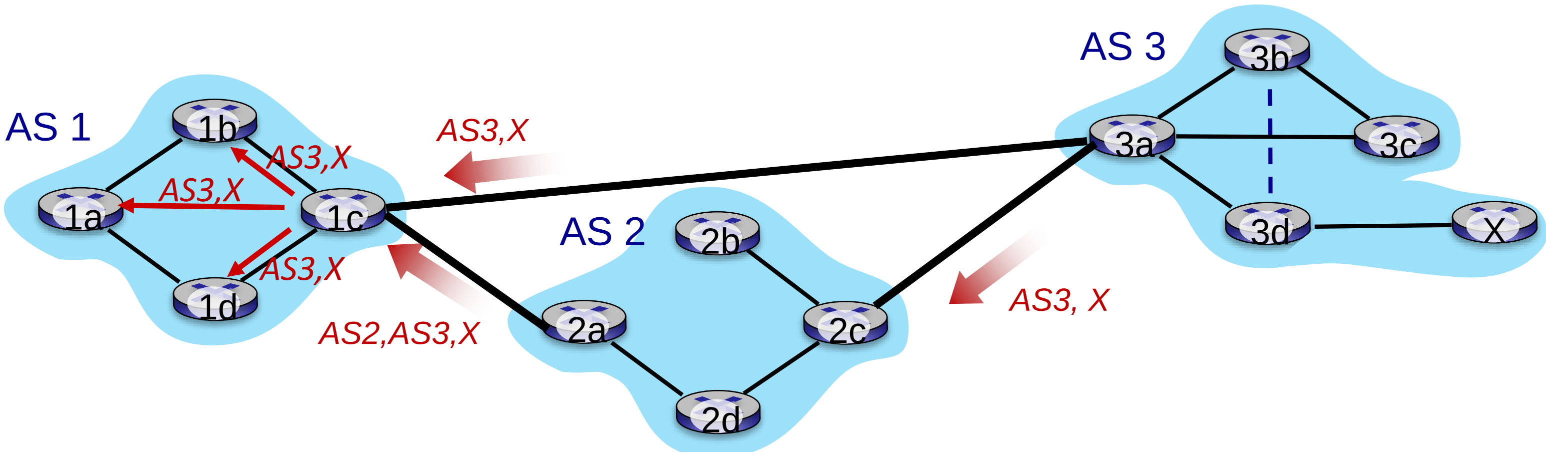
- BGP advertised route: **prefix** + **attributes**
  - **prefix**: destination being advertised
  - two important **attributes**:
    - AS-PATH: **list of ASes** through which prefix advertisement has passed
    - NEXT-HOP: indicates **specific internal-AS router to next-hop AS**
      - IP address of the router interface that begins the AS-PATH.
- **Policy-based routing:**
  - Gateway receiving route advertisement uses *import policy* to accept/decline path (e.g., never route through AS Y).
  - AS policy also determines whether to *advertise* path to other neighboring ASes

BGP path advertisement



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

BGP path advertisement (more)



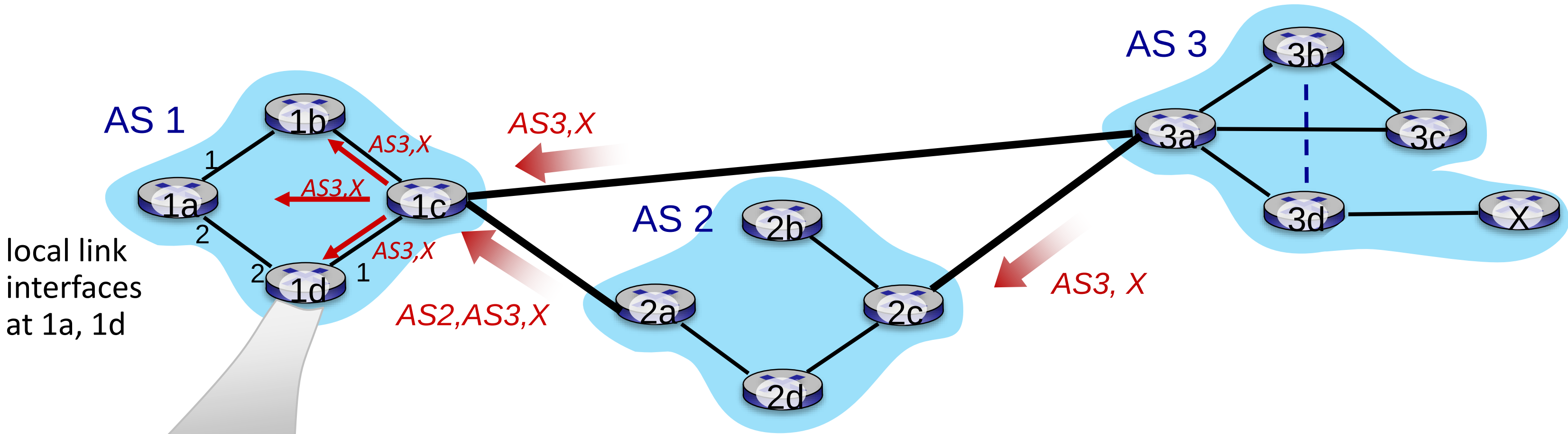
gateway router may learn about multiple paths to destination:

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

BGP messages

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

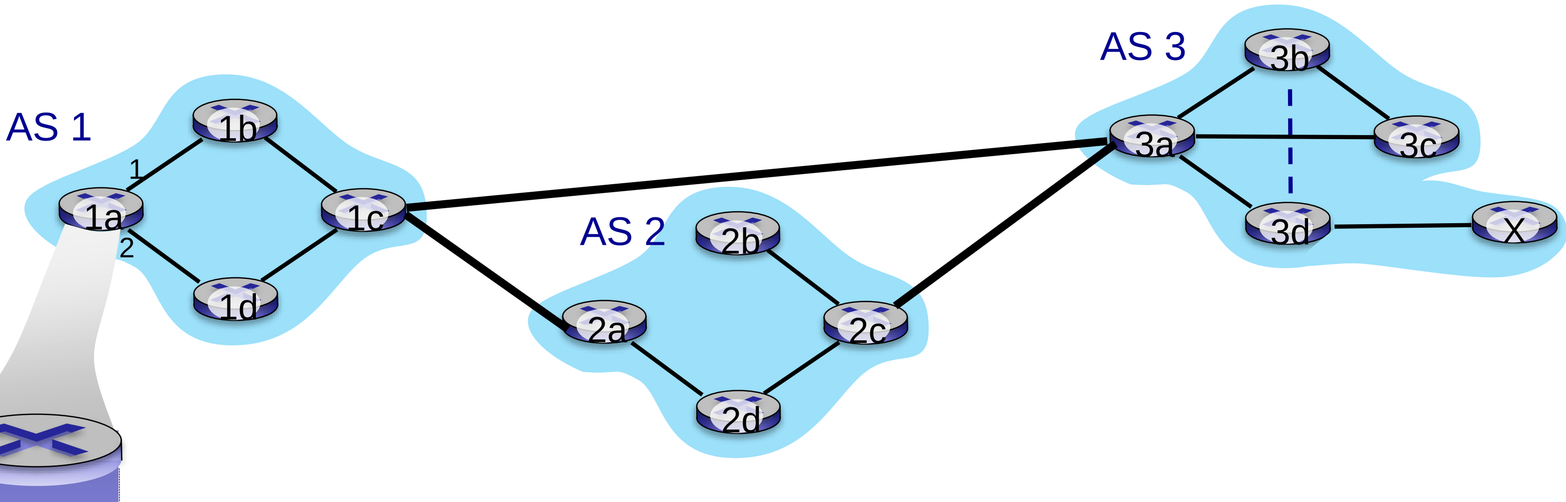
BGP path advertisement



dest	interface
…	…
1c	1
X	1
…	…

- recall: 1a, 1b, 1d learn via iBGP from 1c: “path to X goes through 1c”
- at 1d: OSPF intra-domain routing: to get to 1c, use interface 1
- at 1d: to get to X, use interface 1

BGP path advertisement



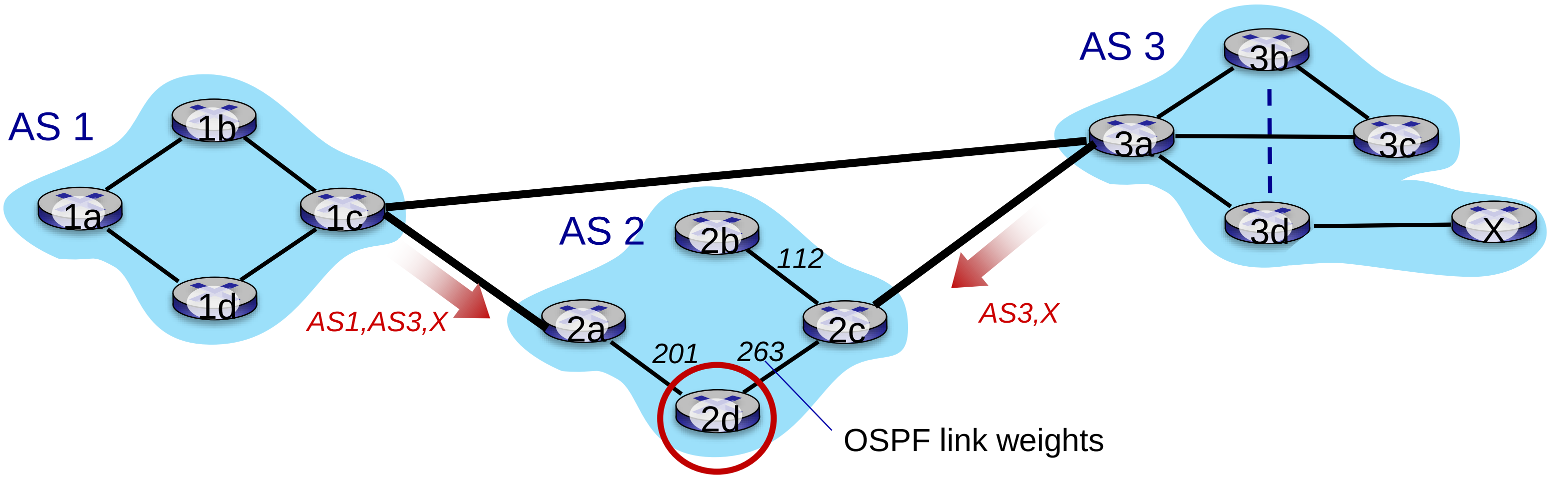
dest	interface
…	…
1c	2
X	2
…	…

- recall: 1a, 1b, 1d learn via iBGP from 1c: “path to X goes through 1c”
- at 1d: OSPF intra-domain routing: to get to 1c, use interface 1
- at 1d: to get to X, use interface 1
- at 1a: OSPF intra-domain routing: to get to 1c, use interface 2
- at 1a: to get to X, use interface 2

Why different Intra-, Inter-AS routing ?

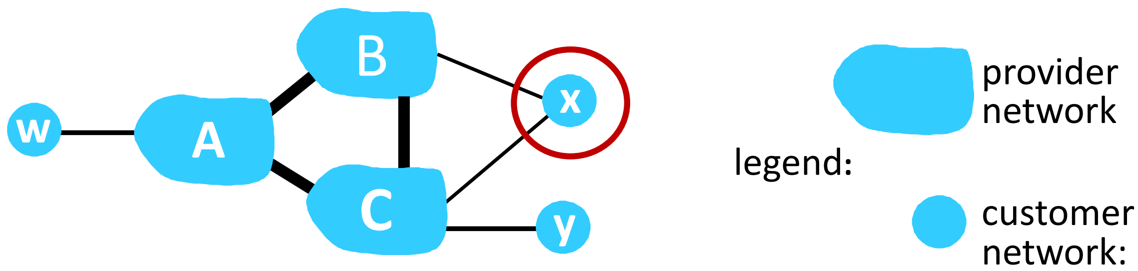
- **Policy**:
  - Inter-AS: admin wants control over how its traffic routed, who routes through its network
  - Intra-AS: single admin, so policy less of an issue
- **Scale**:
  - Hierarchical routing saves table size, reduced update traffic
- **Performance**:
  - Intra-AS: can focus on performance
  - Inter-AS: policy dominates over performance

Hot potato routing



- Router 2d learns (via iBGP) it can route to X via 2a or 2c
- Hot Potato Routing: choose local gateway that has least *intra-domain* cost (e.g., 2d chooses 2a, even though more AS hops to *X*): don't worry about inter-domain cost!
  - Route to closest exit point when there is more than one route to destination

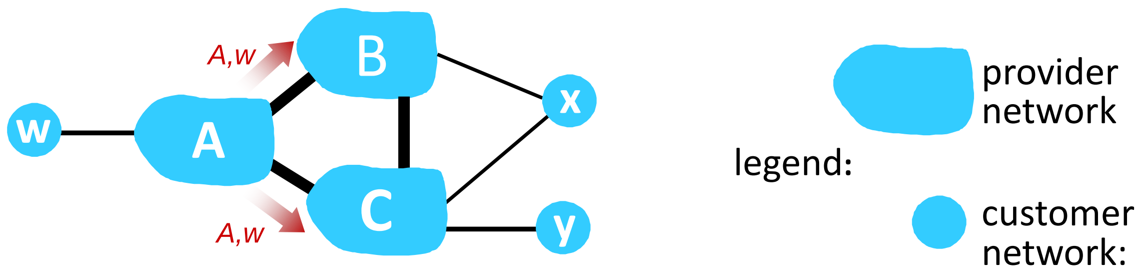
BGP: achieving policy via advertisements



ISP only wants to route traffic to/from its customer networks (does not want to carry transit traffic between other ISPs – a typical "real world" policy)

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

BGP: achieving policy via advertisements



ISP only wants to route traffic to/from its customer networks (does not want to carry transit traffic between other ISPs – a typical "real world" policy)

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

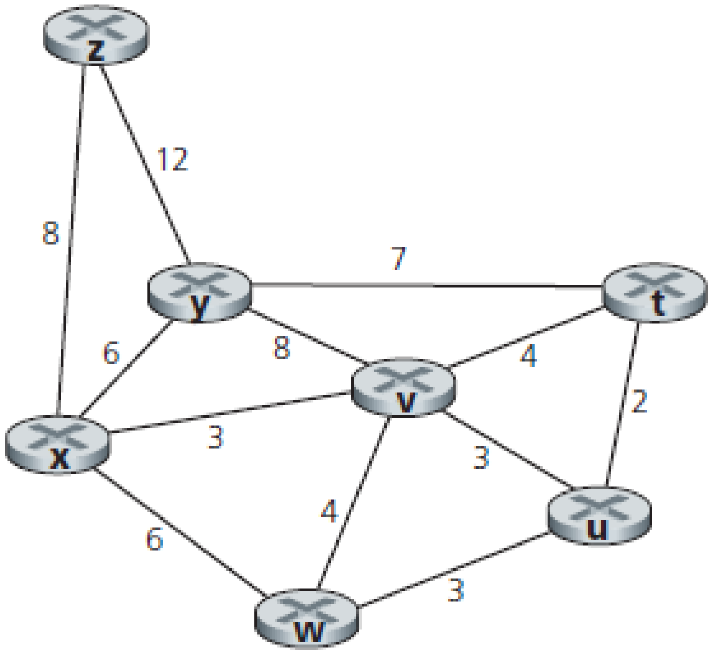
BGP route selection

- Router may learn about more than one route to destination AS, selects route based on:
  - Local preference value attribute: policy decision
  - Shortest AS-PATH
  - Closest NEXT-HOP router: hot potato routing
  - Additional criteria

Problems and Exercises

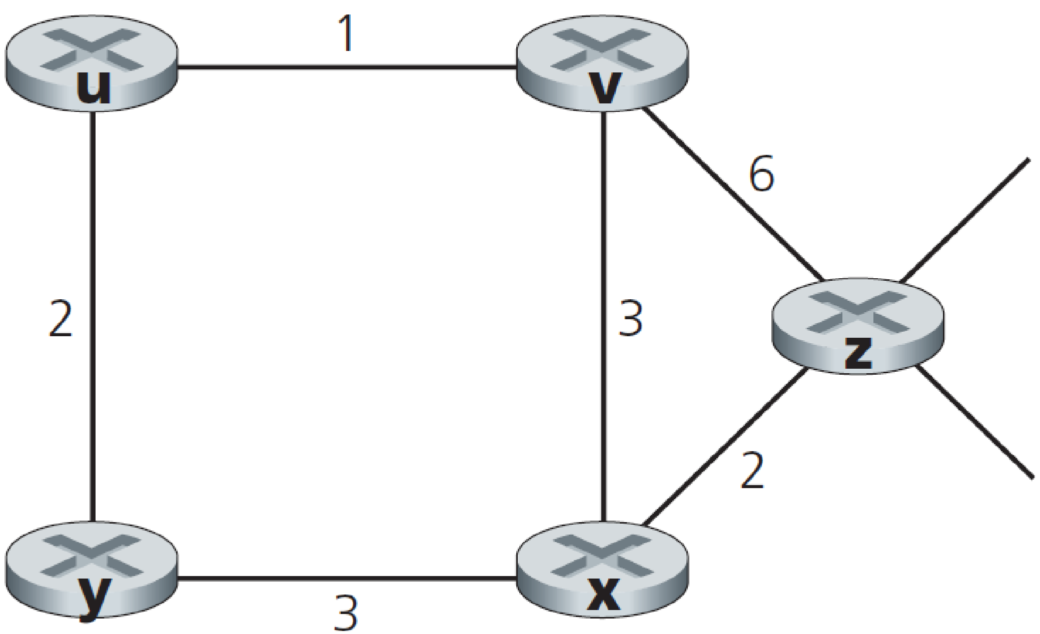
Problem – Dijkstra Algorithm

- Consider the following network. With the indicated link costs, use Dijkstra's shortest-path algorithm to compute the shortest path from x to all network nodes.



Problem DV-Algorithm

- Consider the network shown below, and assume that each node initially knows the costs to each of its neighbors. Consider the distance-vector algorithm and show the distance table entries at node *z*.



Problem - BGP

- Consider the network shown below. Suppose AS3 and AS2 are running OSPF for their intra-AS routing protocol. Suppose AS1 and AS4 are running RIP for their intra-AS routing protocol. Suppose eBGP and iBGP are used for the inter-AS routing protocol. Initially suppose there is *no* physical link between AS2 and AS4.
  a) Router 3c learns about prefix *x* from which routing protocol: OSPF, RIP, eBGP, or iBGP?
  b) Router 3a learns about *x* from which routing protocol?
  c) Router 1c learns about *x* from which routing protocol?
  d) Router 1d learns about *x* from which routing protocol?