The network layer: routing

CS168, Fall 2014

Sylvia Ratnasamy

http://inst.eecs.berkeley.edu/~cs168/fa14/

Last Time

- Forwarding: “data plane”
  - *Directing one data packet*
  - *Each router using local routing state*

- Routing: “control plane”
  - *Computing the forwarding tables that guide packets*
  - *Jointly computed by routers using a distributed ptotocol*

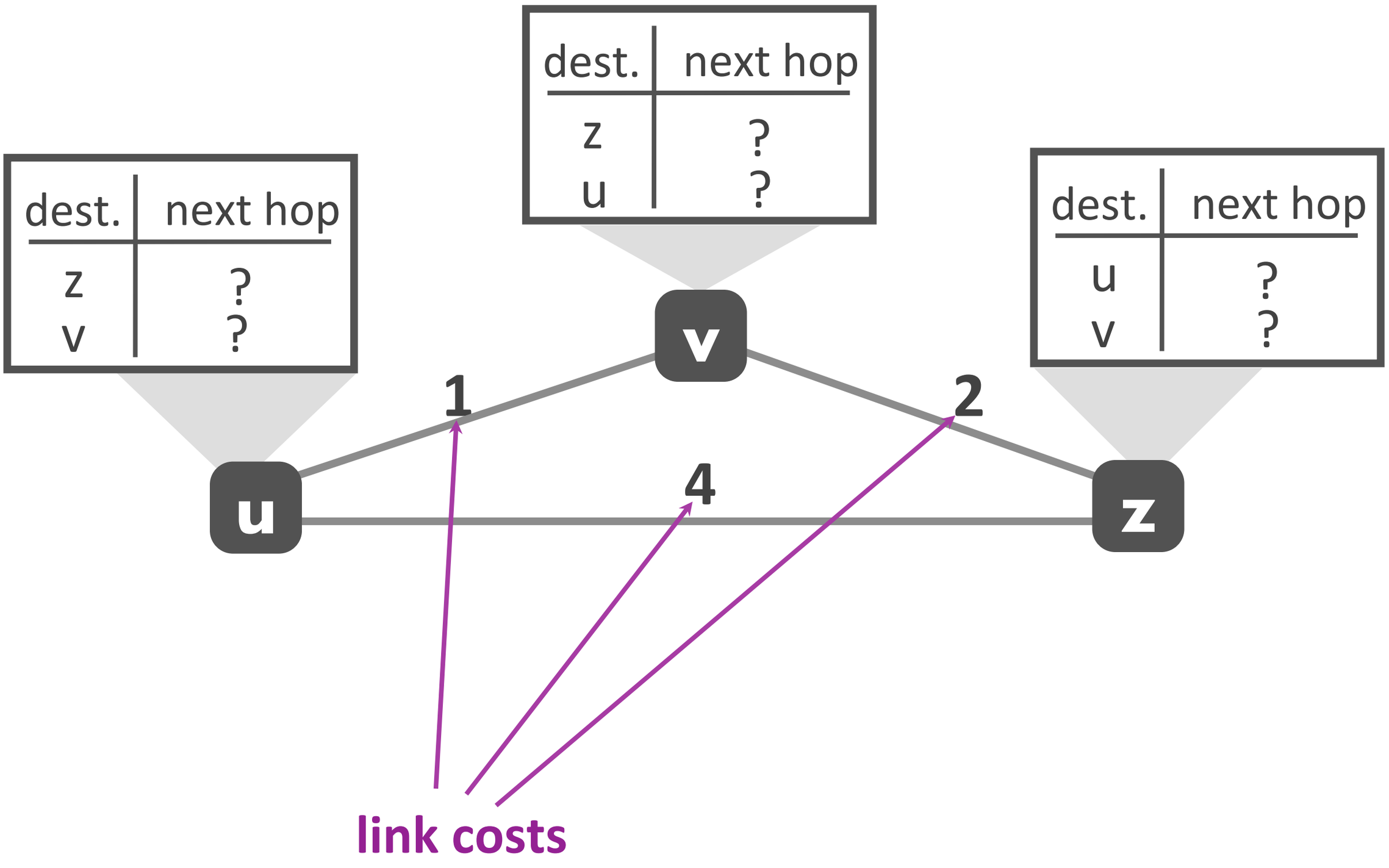
- Routing will be our focus for the next ~2 lectures!

Last Time

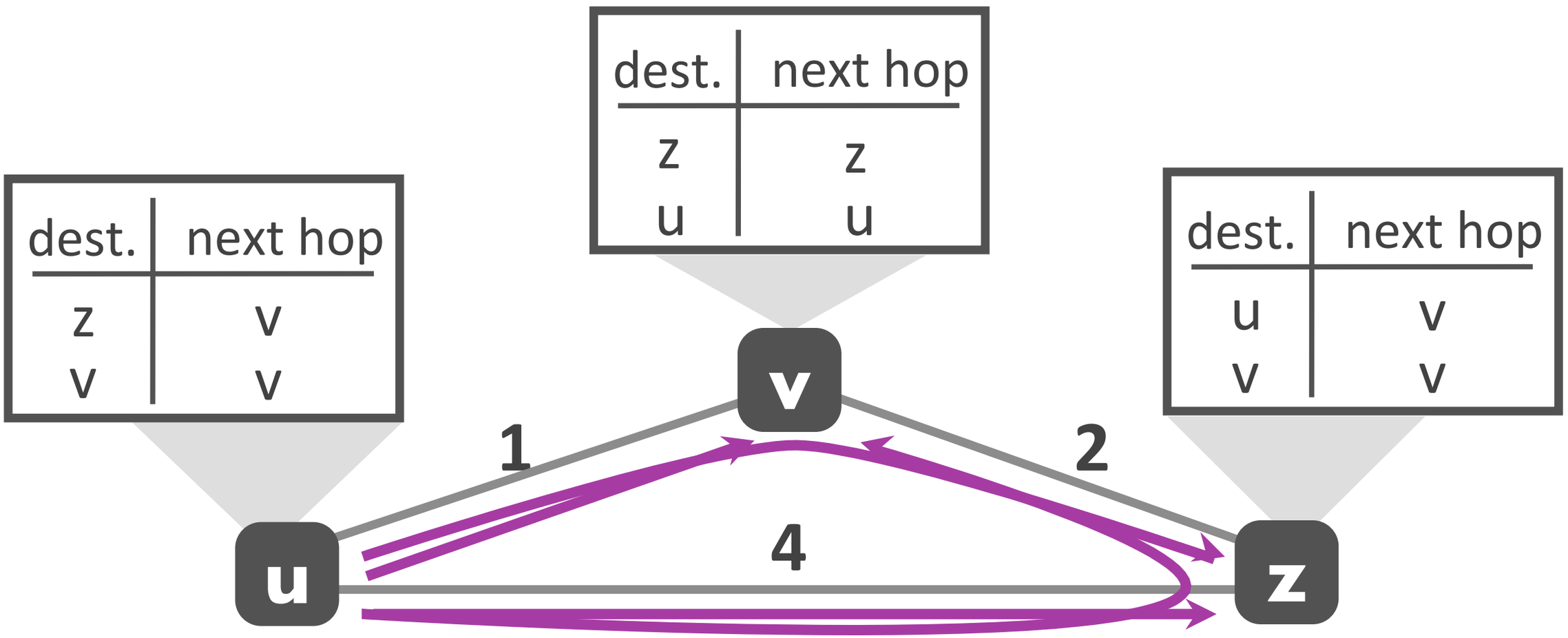
- Two correctness conditions for routing state:
  - *no deadends*
  - *no loops*

Outline

- Least-cost path routing
- Approach 1: link-state routing
- Approach 2: distance-vector routing
- Routing in the Internet (next lecture)



could represent: propagation delay



least-cost path from u to z: u v z

least cost path from u to v: u v

Least-cost path routing

- Given: router graph & link costs
- Goal: find least-cost path
  from each source router
  to each destination router

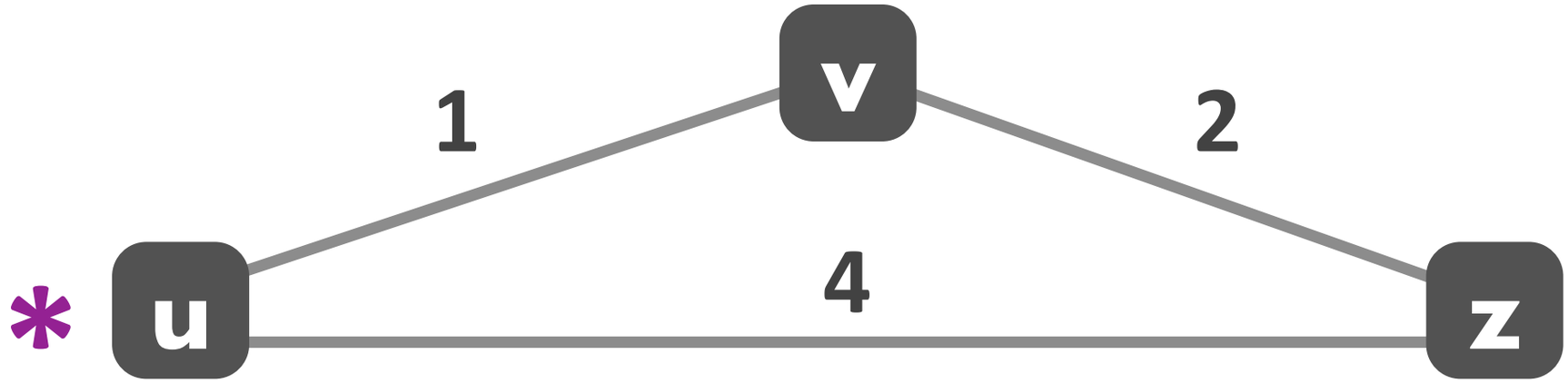
“Least Cost” Routes

- “Least cost” routes an easy way to avoid loops
  - *No sensible cost metric is minimized by traversing a loop*
- Least cost routes are also “destination-based”
  - *i.e., do not depend on the source*
  - *Why is this?*
- Least-cost paths form a spanning tree

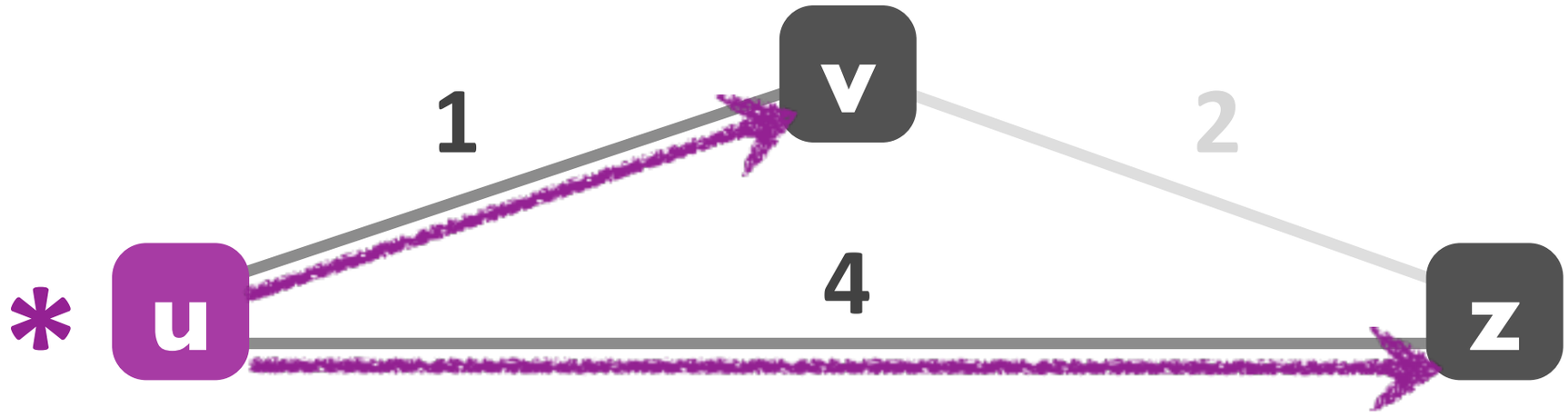
Outline

- Least-cost path routing
- Approach 1: link-state routing
- Approach 2: distance-vector routing
- Routing in the Internet

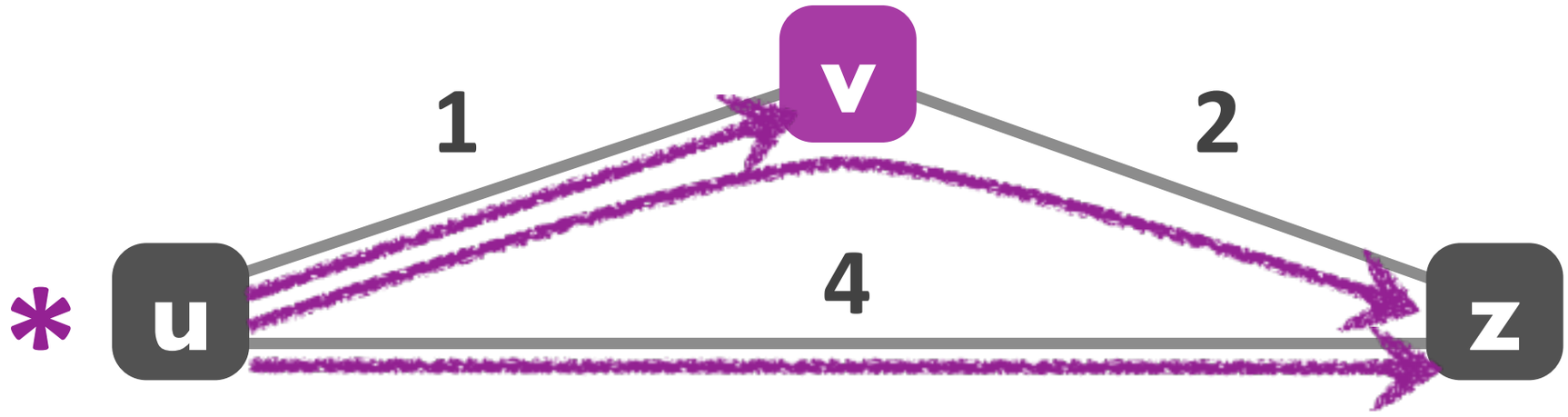
dest.	next hop	cost
z		
v		



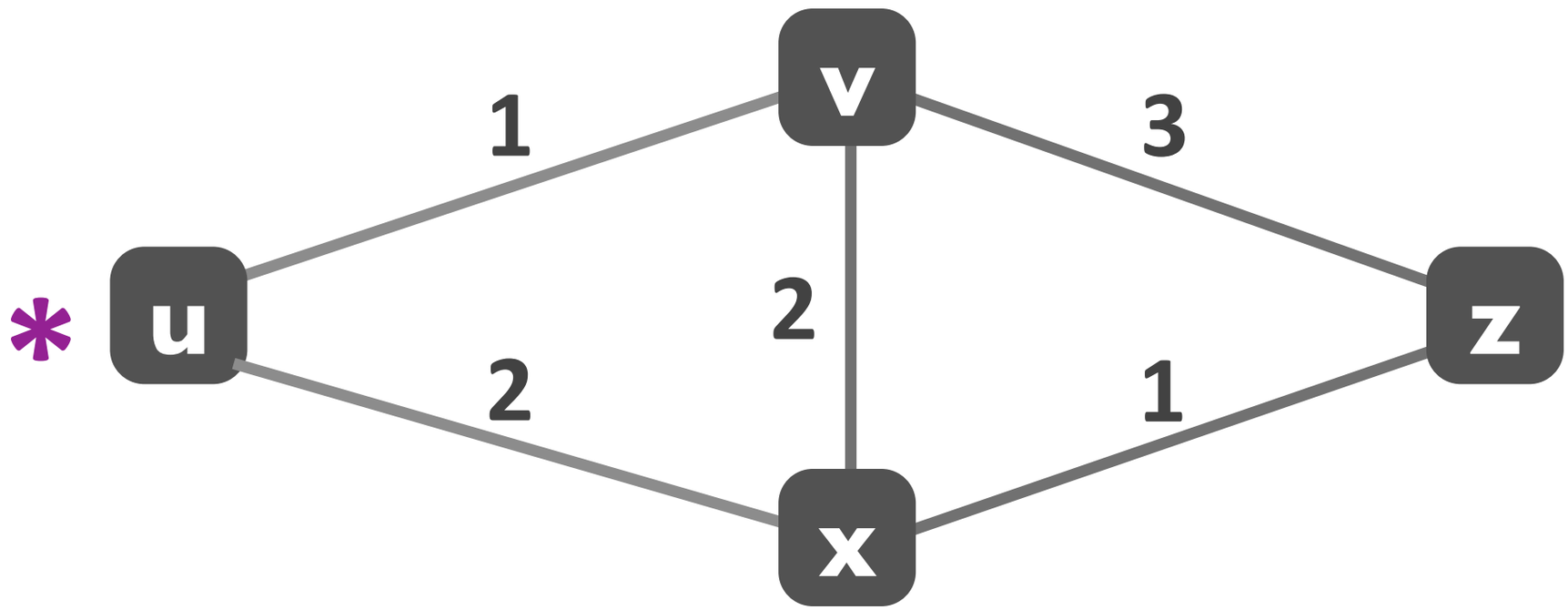
dest.	next hop	cost
z	z	4
v	v	1



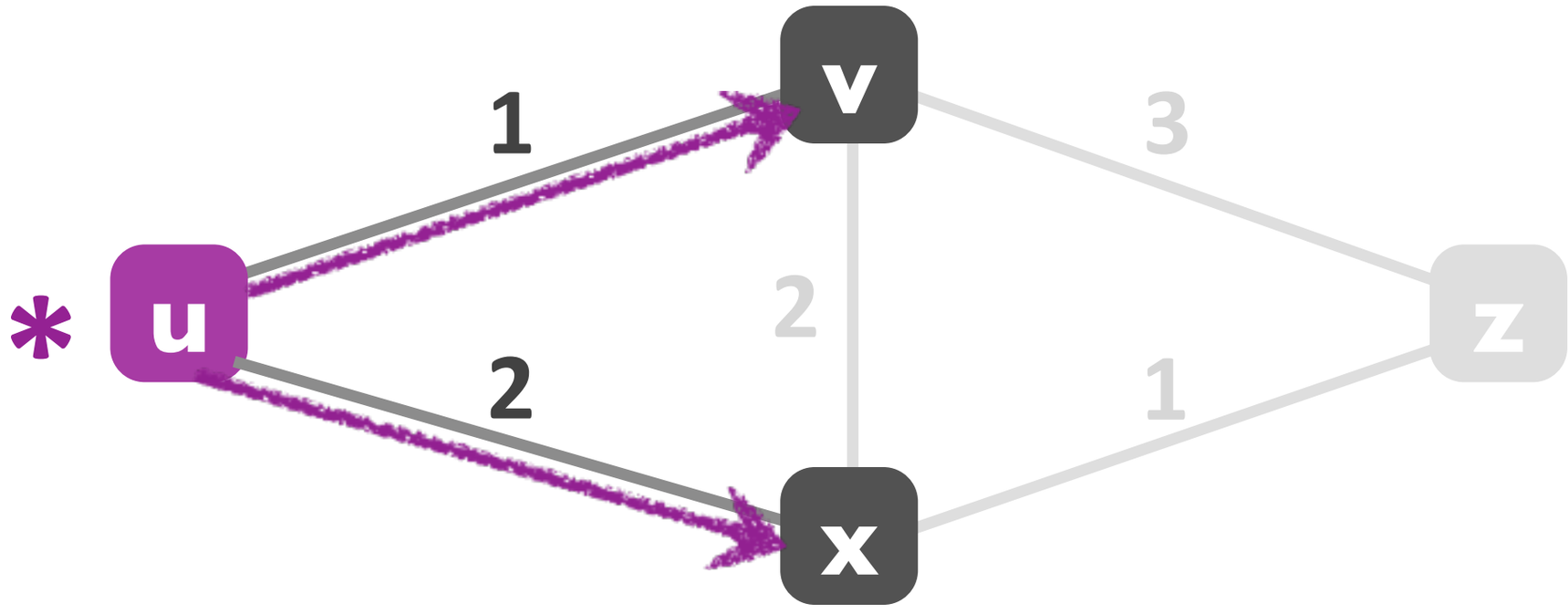
dest.	next hop	cost
z	~~z~~ v	~~4~~ 3
v	v	1



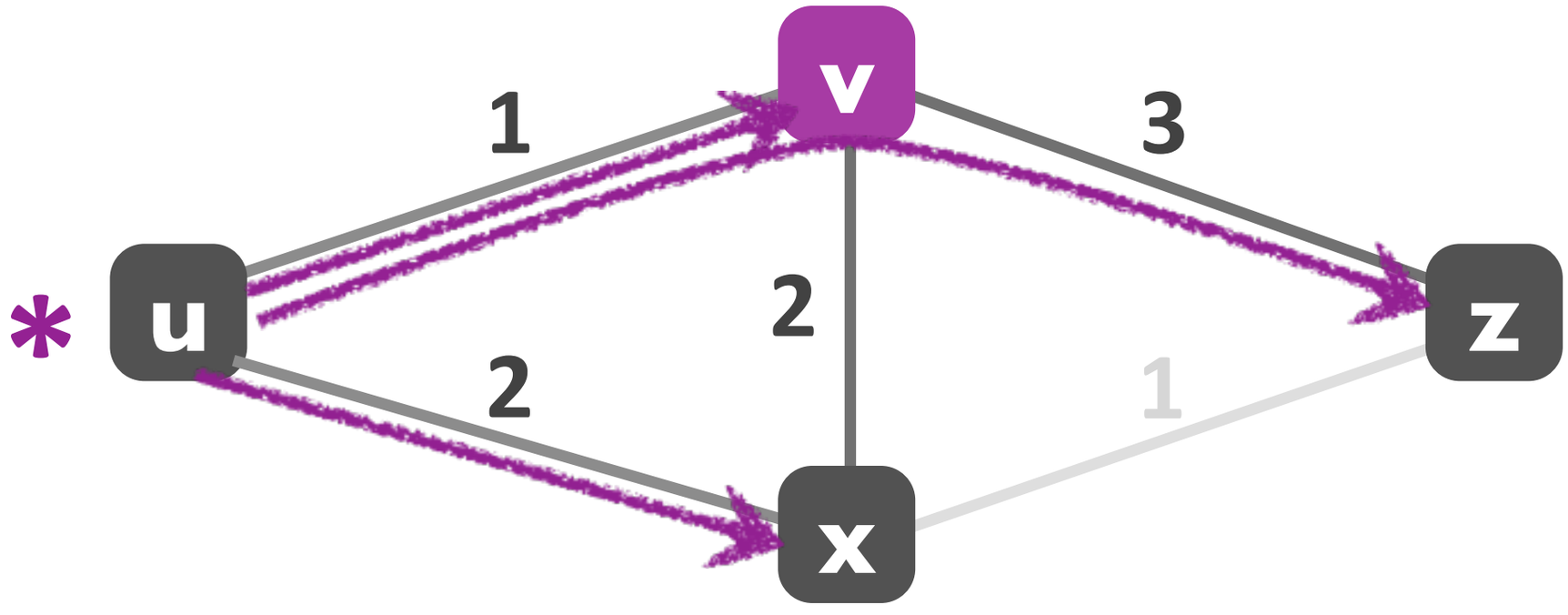
dest.	next hop	cost
z		
v		
x		



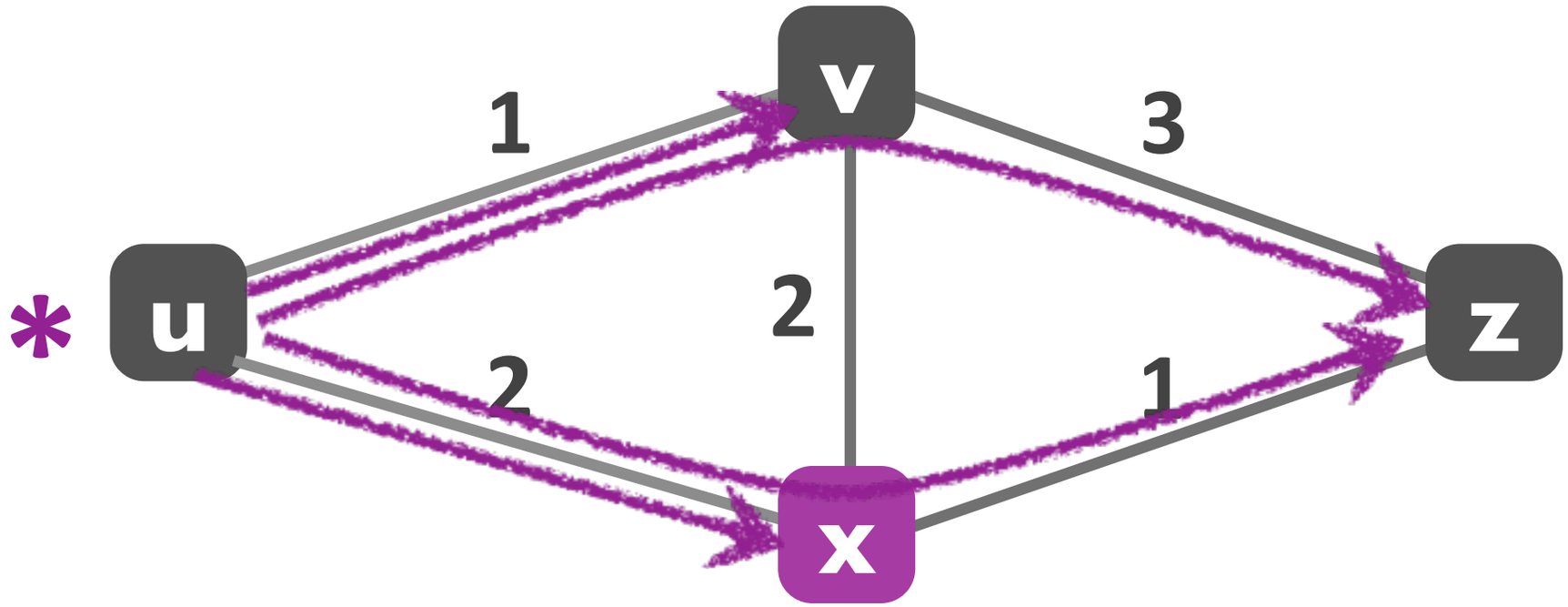
dest.	next hop	cost
z	-	-
v	v	1
x	x	2



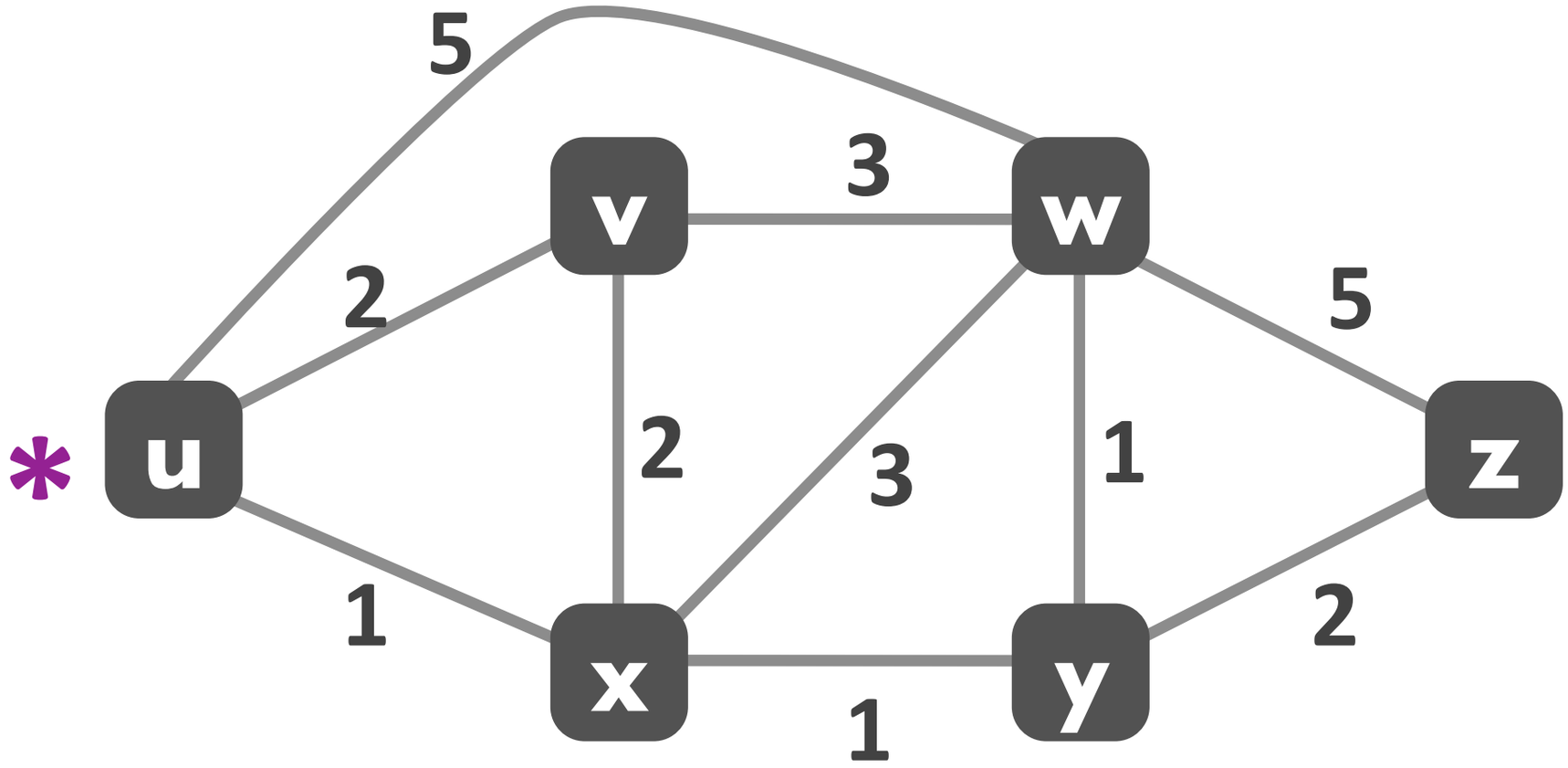
dest.	next hop	cost
z	~~-~~ v	~~-~~ 4
v	v	1
x	x	2



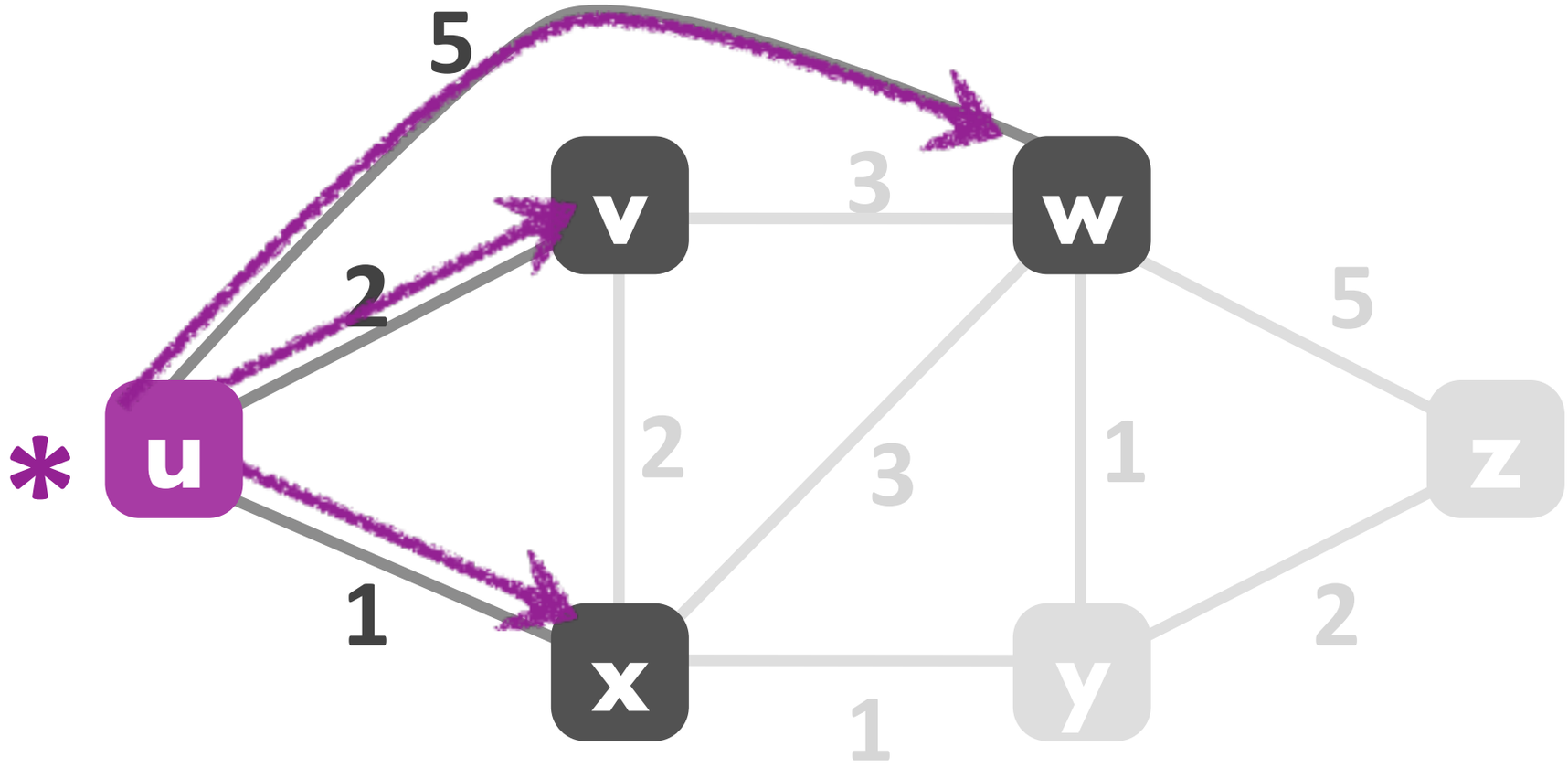
dest.	next hop	cost
z	~~-~~ ~~v~~ x	~~-~~ ~~4~~ 3
v	v	1
x	x	2



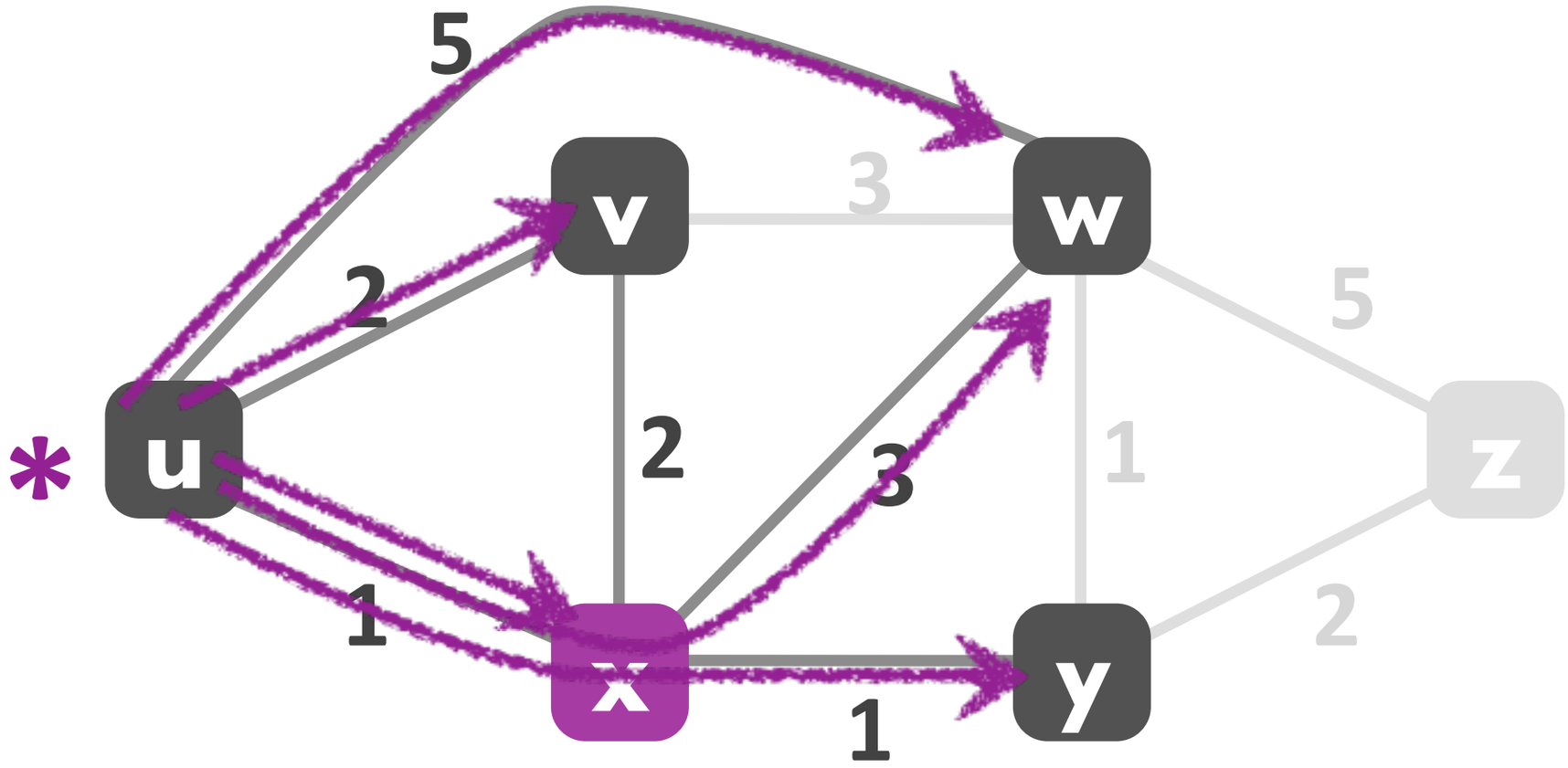
dest.	next hop	cost
z		
w		
y		
v		
x		



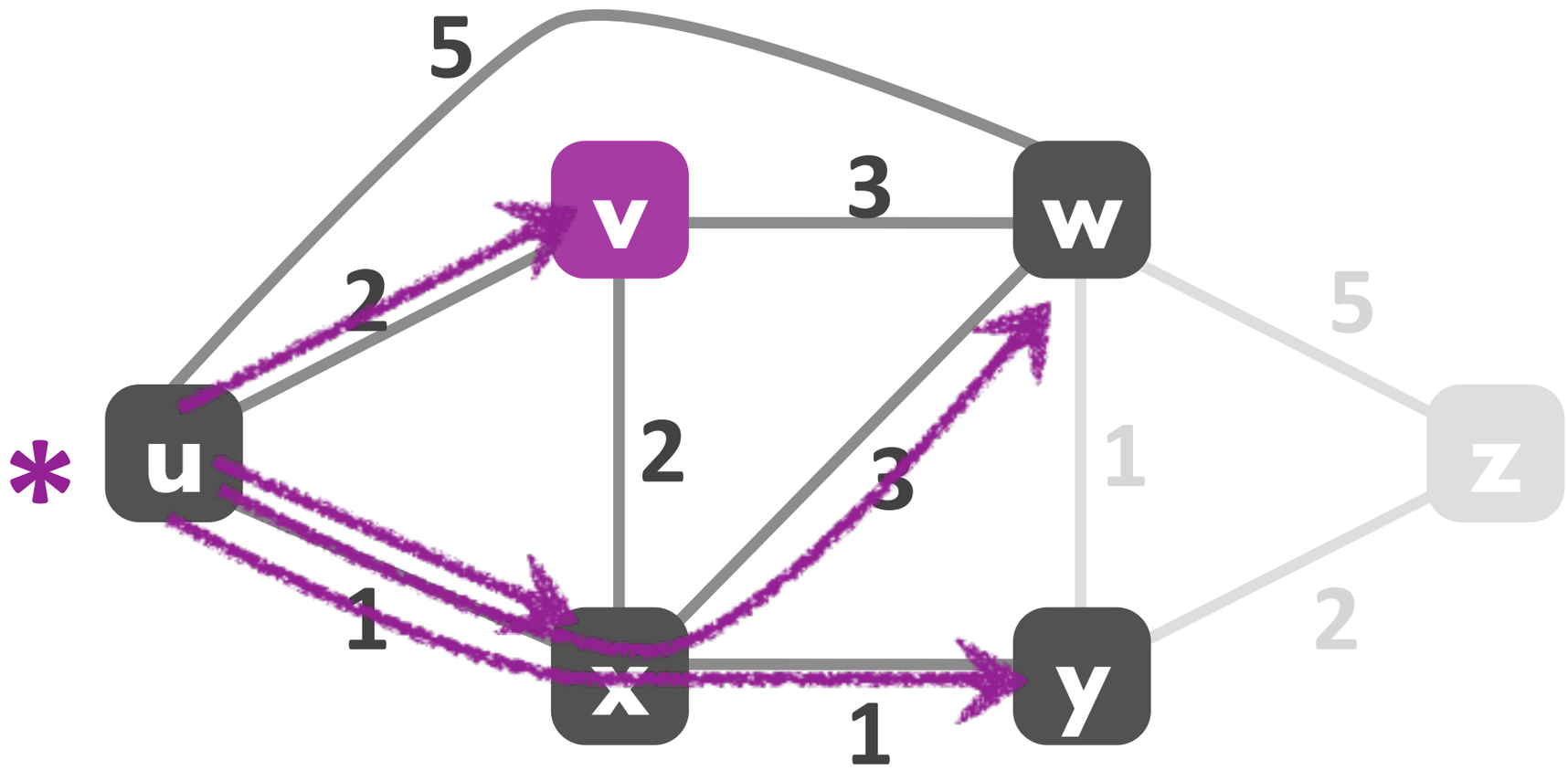
dest.	next hop	cost
z	-	-
w	w	5
y	-	-
v	v	2
x	x	1



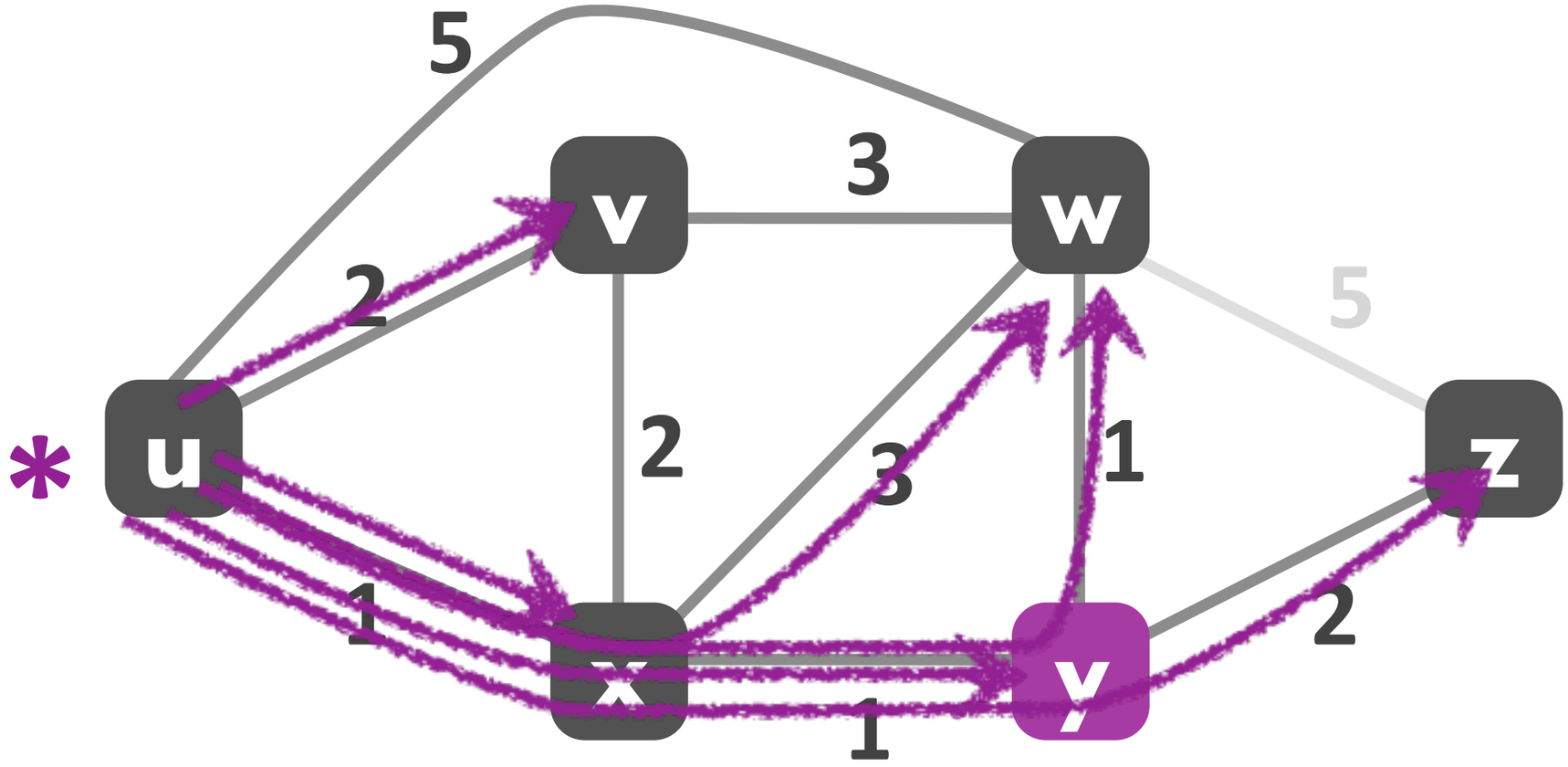
dest.	next hop	cost
z	-	-
w	~~w~~ x	~~5~~ 4
y	~~-~~ x	~~-~~ 2
v	v	2
x	x	1



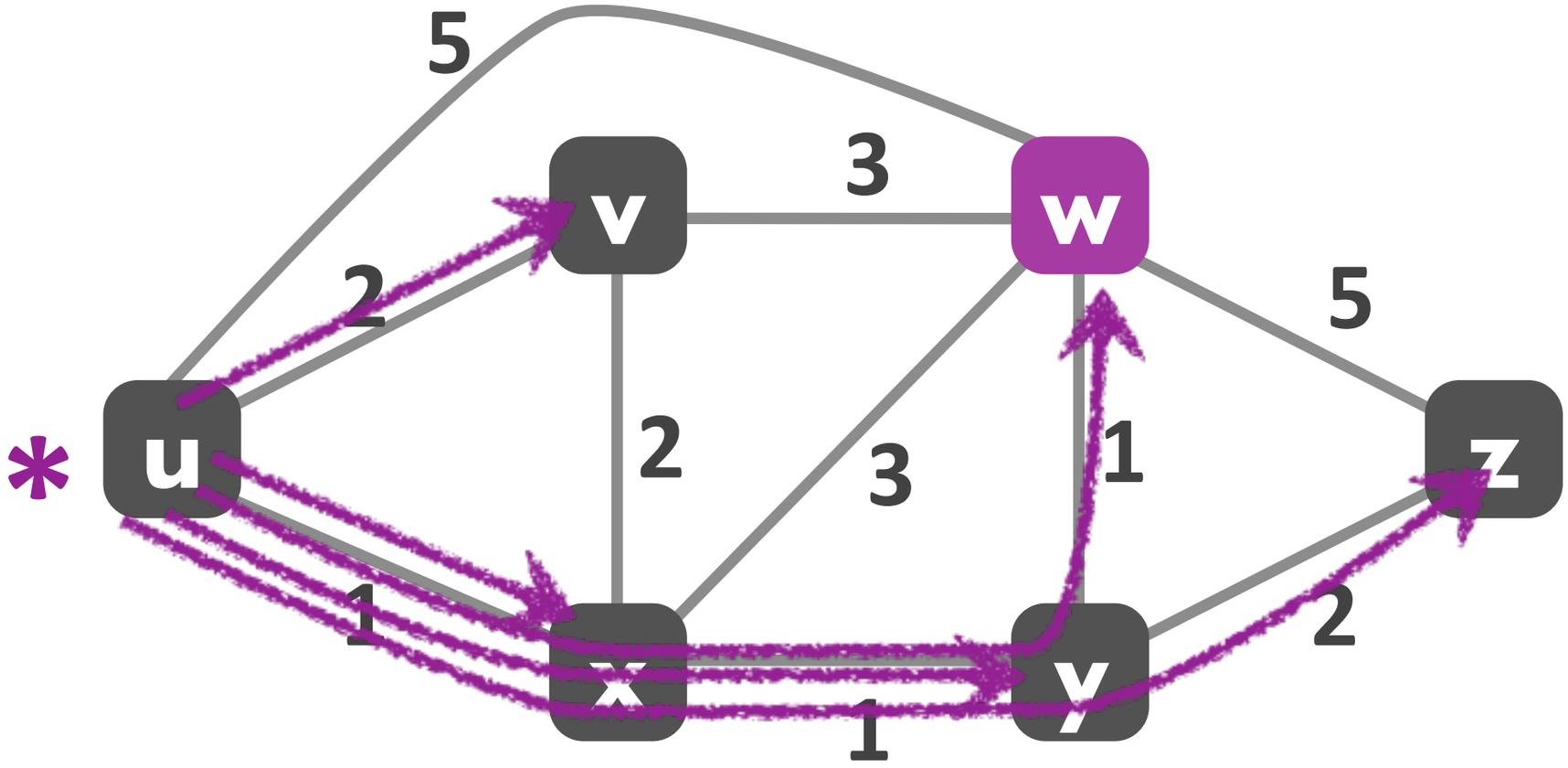
dest.	next hop	cost
z	-	-
w	~~w~~ x	~~5~~ 4
y	~~-~~ x	~~-~~ 2
v	v	2
x	x	1



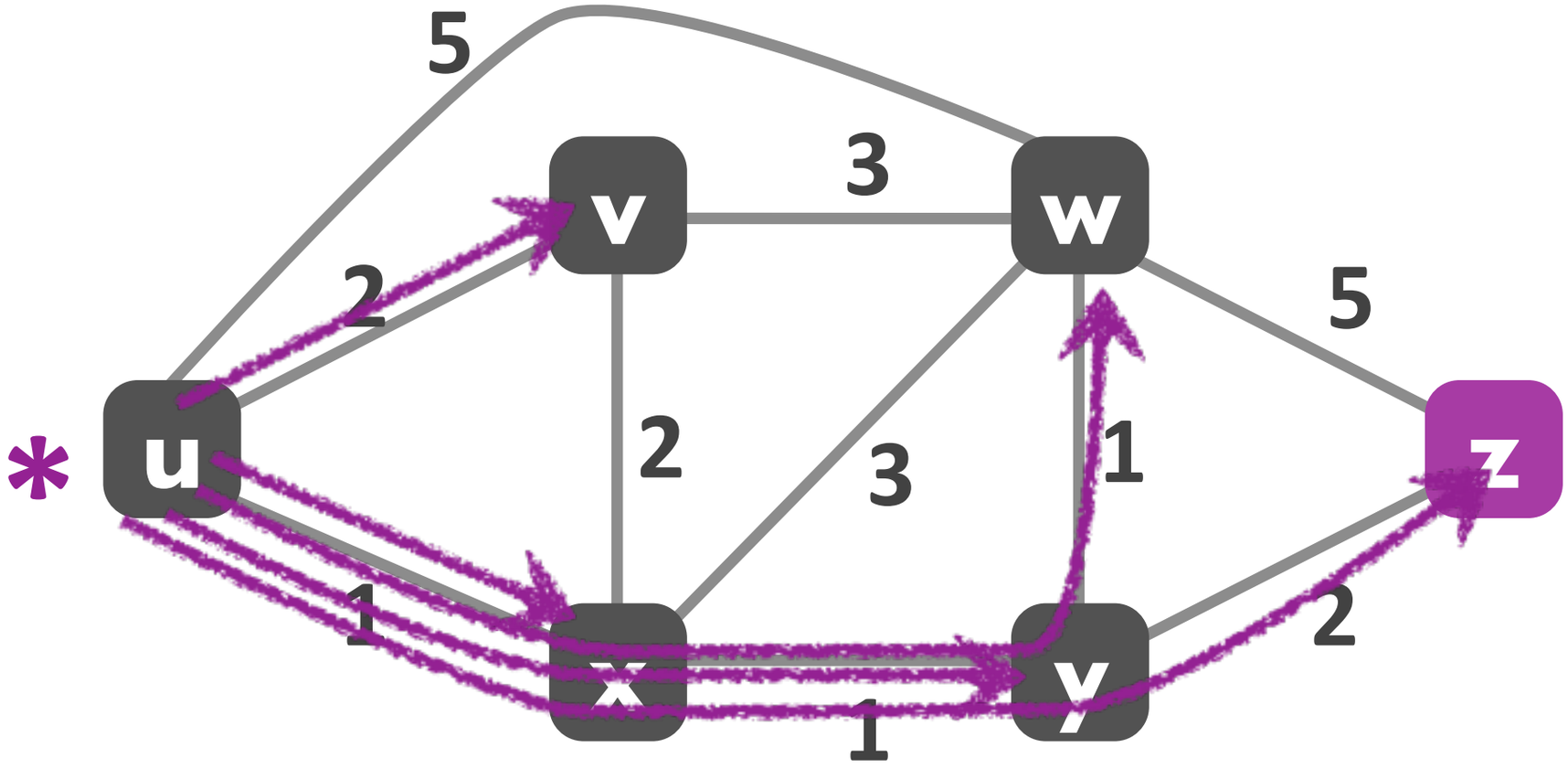
dest.	next hop	cost
z	~~-~~ x	~~-~~ 4
w	~~w~~ x	~~5~~ ~~4~~ 3
y	~~-~~ x	~~-~~ 2
v	v	2
x	x	1



dest.	next hop	cost
z	~~-~~ x	~~-~~ 4
w	~~w~~ x	~~5~~ ~~4~~ 3
y	~~-~~ x	~~-~~ 2
v	v	2
x	x	1



dest.	next hop	cost
z	~~-~~ x	~~-~~ 4
w	~~w~~ x	~~5~~ ~~4~~ 3
y	~~-~~ x	~~-~~ 2
v	v	2
x	x	1



Routing algorithm for source u

- Input: router graph & link costs
- Output: least-cost path
  from source router u
  to every other router

Dijkstra's algorithm

- Source router considers every other router
  - *starting from next "closest" neighbor*
- Checks whether it can improve current paths
  - *by using that router as an intermediate point*
- Ends when all intermediaries have been considered
- See text / 61C notes for exact algorithm

From routing algorithm to protocol

- Note: Dijkstra's is a local computation!
  - *computed by one node given complete network graph*

- Possibilities:
  - *Option#1: a separate machine runs the algorithm*
  - *Option#2: every router runs the algorithm*

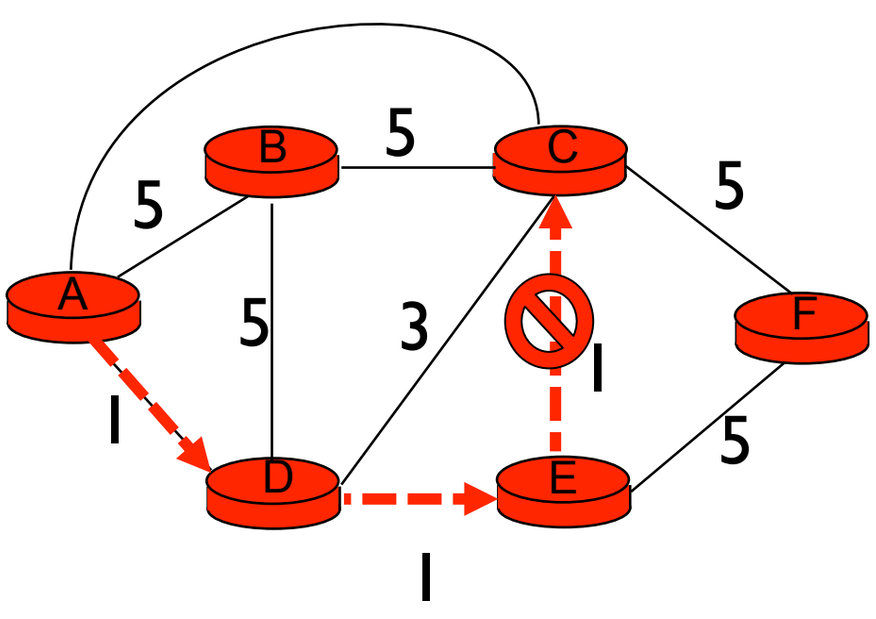
- The Internet currently uses Option#2

Link State Routing

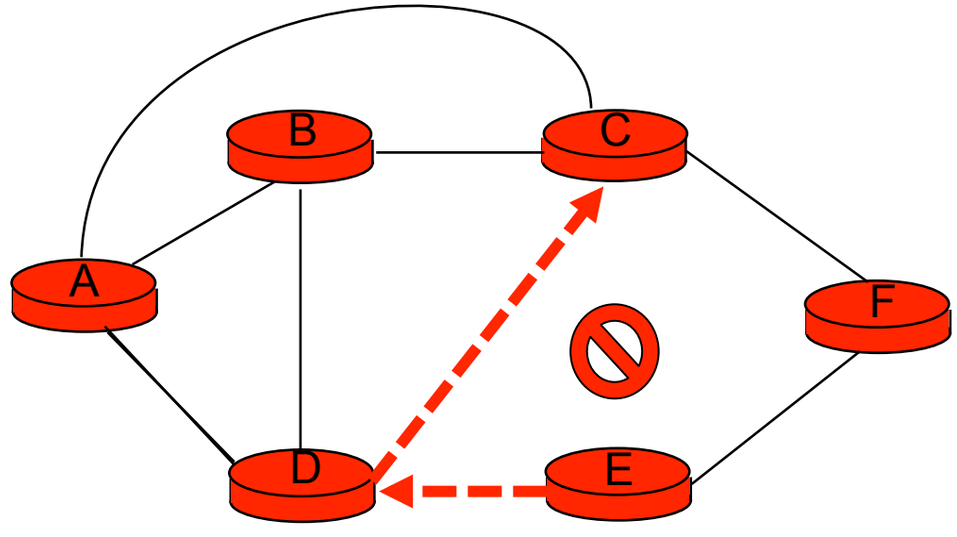
- Every router knows its local "link state"
  - *router u: "(u,v) with cost=2; (u,x) with cost=1"*
- A router floods its link state to all other routers
  - *does so periodically or when its link state changes*
- Hence, every router learns the entire network graph
  - *runs Dijkstra's locally to compute its forwarding table*
- OSPF is a link-state protocol (IETF RFC 2328 or 5340)
  - *Berkeley runs OSPF internally!*

Link State Routing

- Are loops possible?



A and D think that this is the path to C



E thinks that this is the path to C

Convergence

- All routers have consistent routing information
  - *E.g., all nodes having the same link-state database*

- Forwarding is consistent after convergence
  - *All nodes have the same link-state database*
  - *All nodes forward packets on same paths*

Convergence Delay

- Time to achieve convergence
- Sources of convergence delay?
  - *time to detect failure*
  - *time to flood link-state information*
  - *time to re-compute forwarding tables*
- Performance during convergence period?
  - *lost packets due to blackholes*
  - *looping packets*
  - *out-of-order packets reaching the destination*

Link State Routing

- Are loops possible?
  - *yes, until convergence*
- Scalability?
  - *O(NE) messages*
  - *$O(N^2)$ computation time*
  - *O(network diameter) convergence time*
  - *O(N) entries in forwarding table*
- Do we have to use Dijkstra's?

Outline

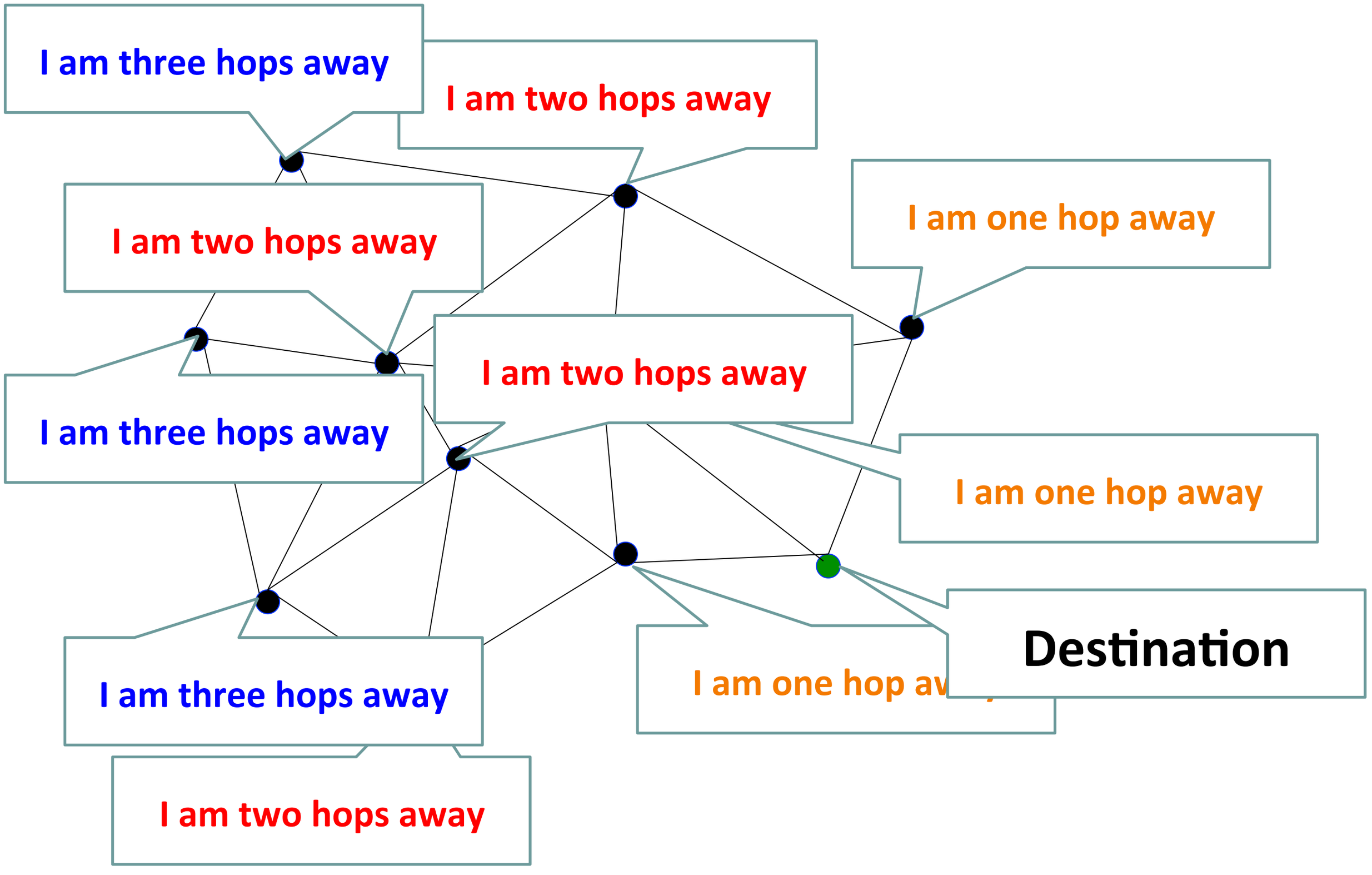
- Least-cost path routing
- Approach 1: link-state routing
- Approach 2: distance-vector routing
- Routing in the Internet

Experiment

- Your job: find the youngest person in the room
- Ground Rules
  - *You may not leave your seat, nor shout loudly across the class*
  - *You may talk with your immediate neighbors (hint: "exchange updates" with them)*
- At the end of 5 minutes, I will pick a victim and ask:
  - *who is the youngest person in the room? (name , date)*
  - *which one of your neighbors first told you this information?*

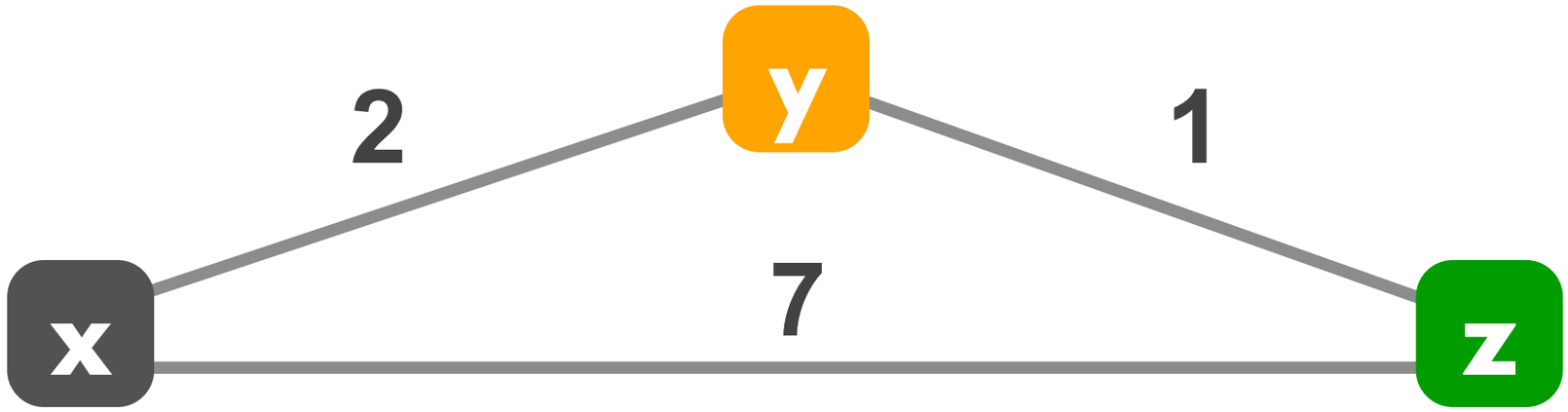
Go!

Example of Distributed Computation



This is y's "distance vector"

	x	y	z
x	-	-	-
y	2	0	1
z	-	-	-

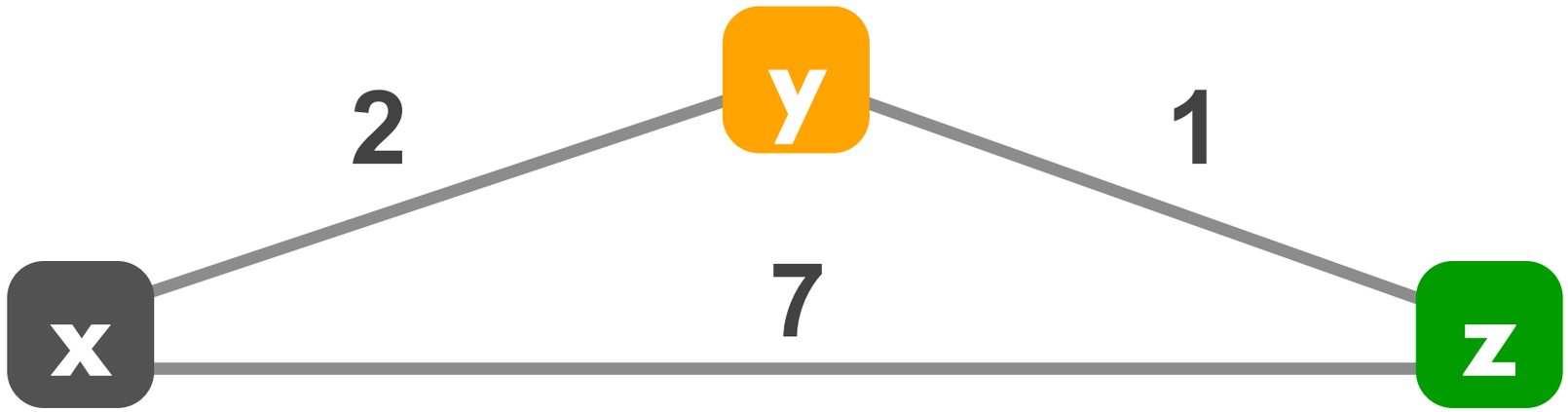


from \ to

	x	y	z
x	0	2	7
y	-	-	-
z	-	-	-

	x	y	z
x	-	-	-
y	-	-	-
z	7	1	0

	x	y	z
x	0	2	7
y	~~2~~	~~0~~	1
z	7	1	0



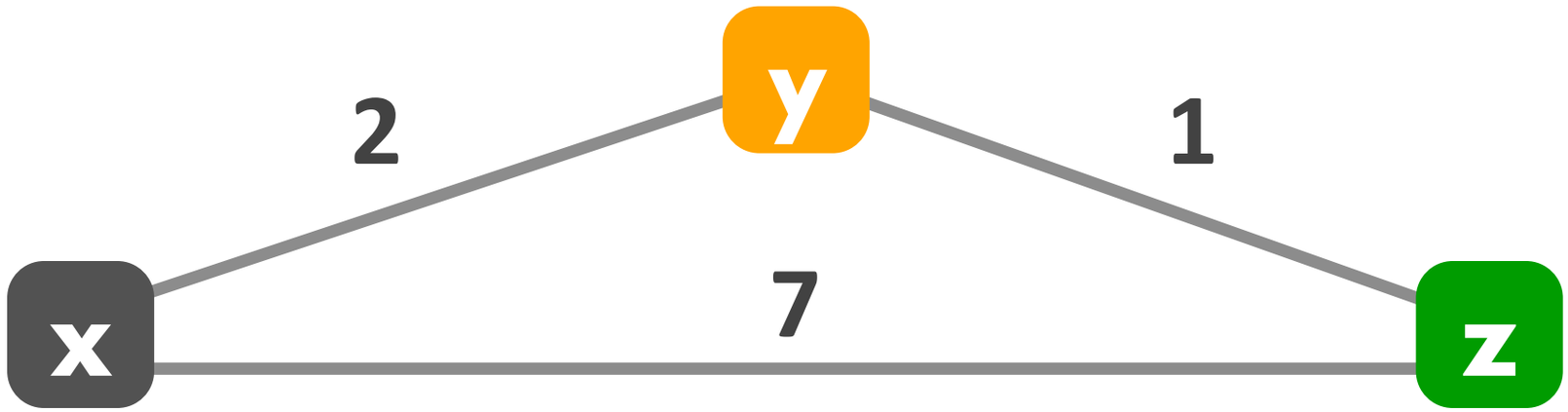
to

from

	x	y	z
x	0	~~2~~	~~7~~
y			
z			

	x	y	z
x	0	2	7
y	2	0	1
z	~~7~~	~~1~~	~~0~~

	x	y	z
x	0	2	7
y	2	0	1
z	7	1	0



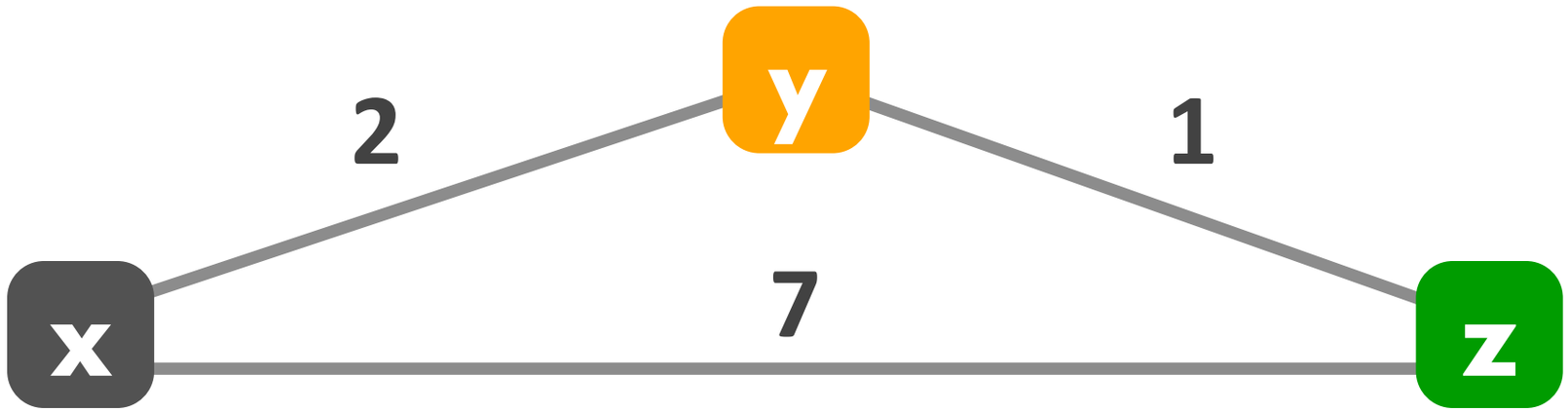
to

from

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

	x	y	z
x	0	2	7
y	2	0	1
z	7	1	0

	x	y	z
x	0	2	7
y	2	0	1
z	7	1	0



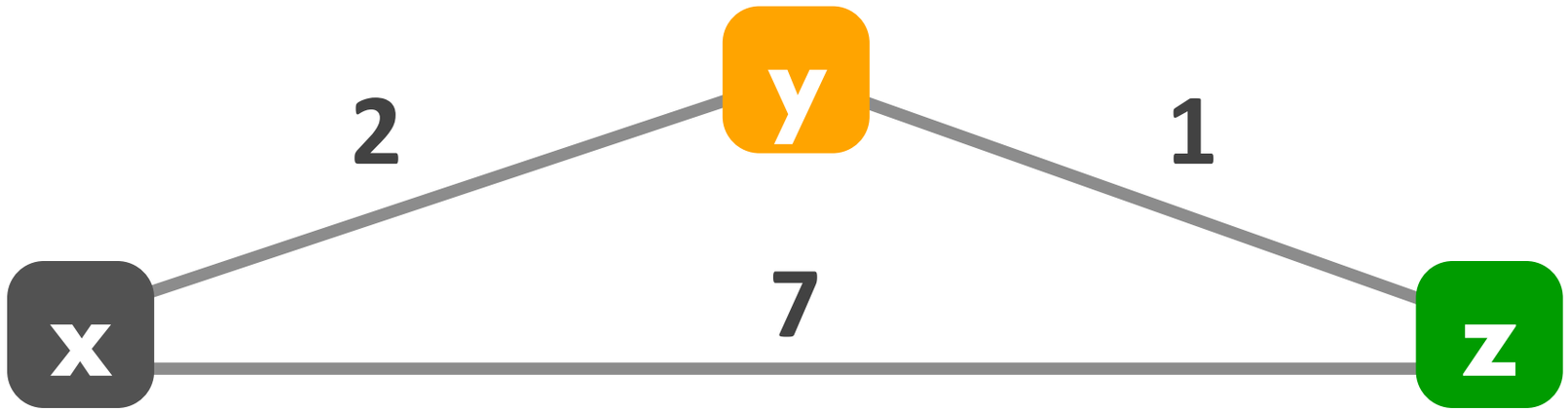
to

from

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

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

	x	y	z
x	0	2	3
y	2	0	1
z	3	1	0



to

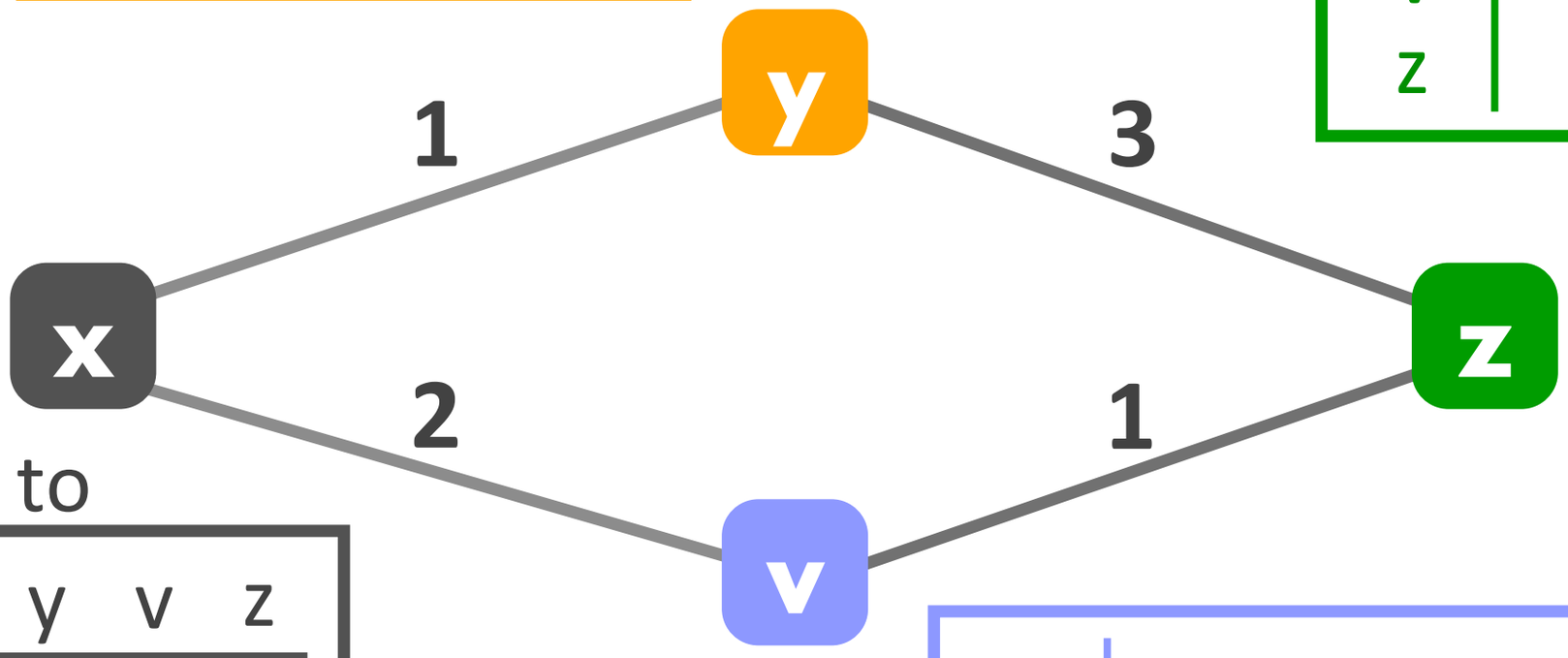
from

	x	y	z
x	0	2	3
y	2	0	1
z	3	1	0

	x	y	z
x	0	2	3
y	2	0	1
z	3	1	0

	x	y	v	z
x	-	-	-	-
y	1	0	-	3
z	-	-	-	-

	x	y	v	z
y	-	-	-	-
v	-	-	-	-
z	-	3	1	0



to

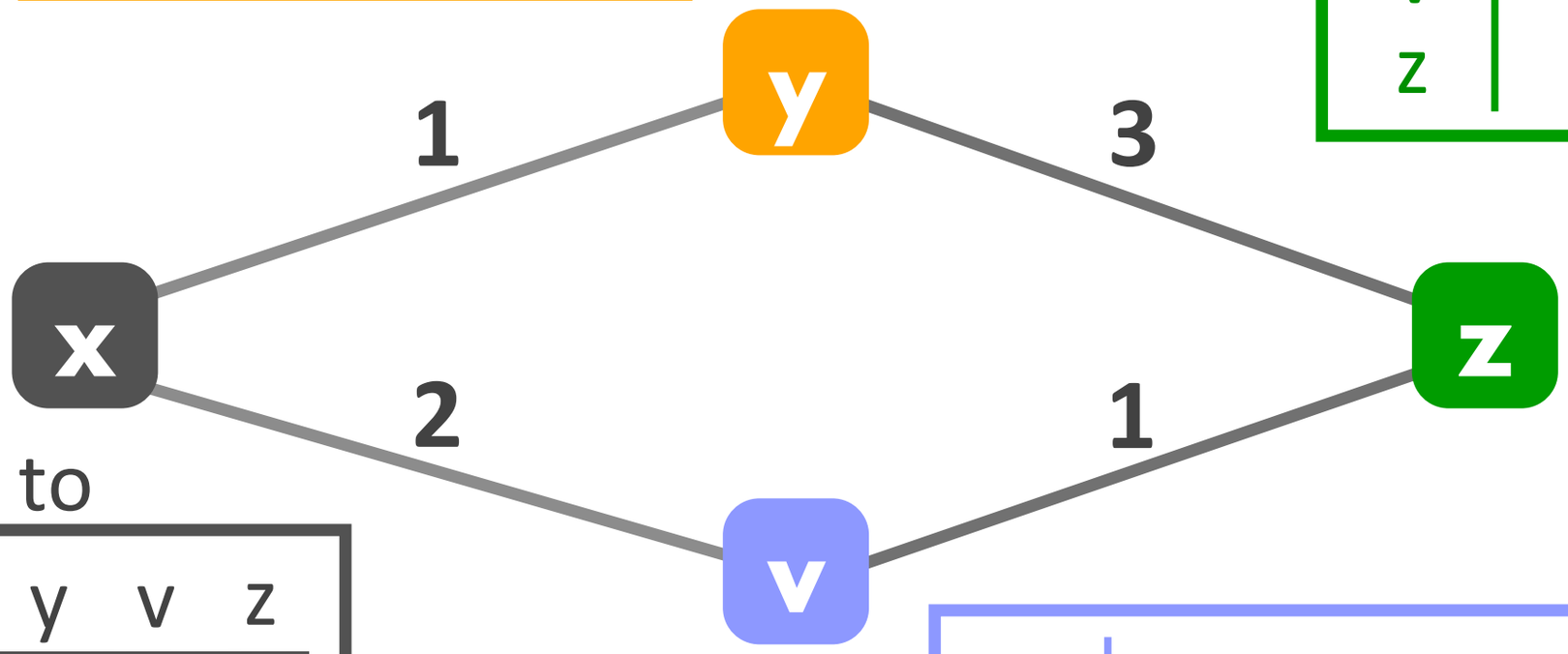
from

	x	y	v	z
x	0	1	2	-
y	-	-	-	-
v	-	-	-	-

	x	y	v	z
x	-	-	-	-
v	2	-	0	1
z	-	-	-	-

	x	y	v	z
x	0	1	2	-
y	1	0	-	3
z	-	3	1	0

	x	y	v	z
y	1	0	-	3
v	2	-	0	1
z	-	3	1	0



to

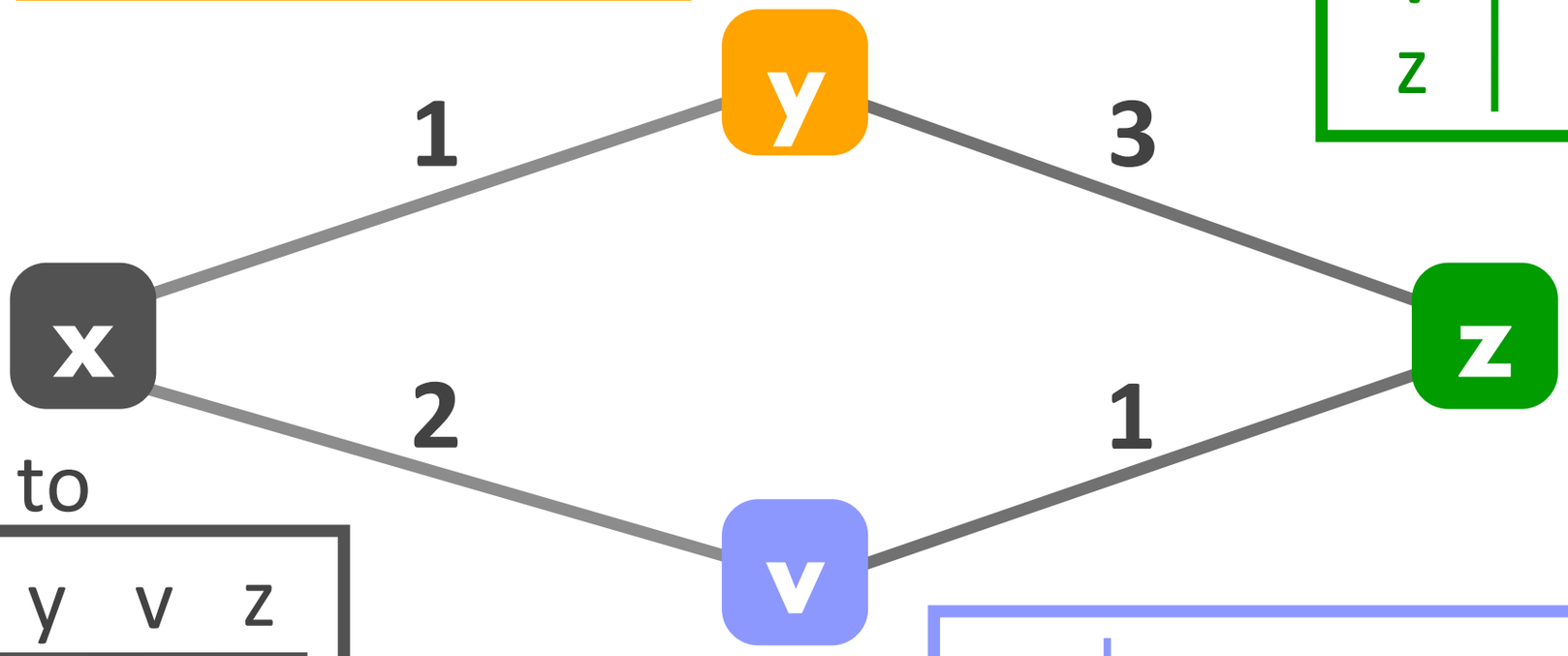
from

	x	y	v	z
x	0	1	2	-
y	1	0	-	3
v	2	-	0	1

	x	y	v	z
x	0	1	2	-
v	2	-	0	1
z	-	3	1	0

	x	y	v	z
x	0	1	2	-
y	1	0	-	3
z	-	3	1	0

	x	y	v	z
y	1	0	-	3
v	2	-	0	1
z	-	3	1	0



to

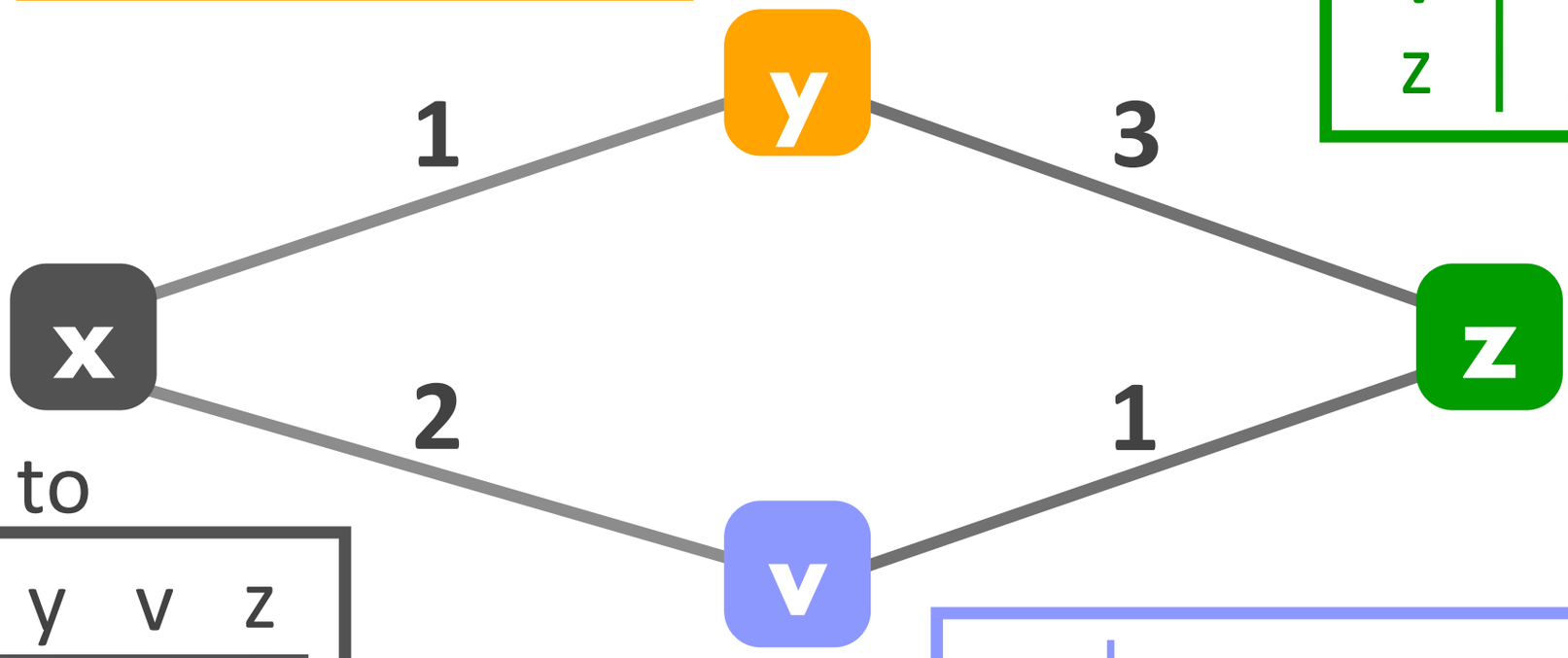
from

	x	y	v	z
x	0	1	2	3
y	1	0	-	3
v	2	-	0	1

	x	y	v	z
x	0	1	2	-
v	2	-	0	1
z	-	3	1	0

	x	y	v	z
x	0	1	2	-
y	1	0	-	3
z	-	3	1	0

	x	y	v	z
y	1	0	-	3
v	2	-	0	1
z	3	3	1	0



to

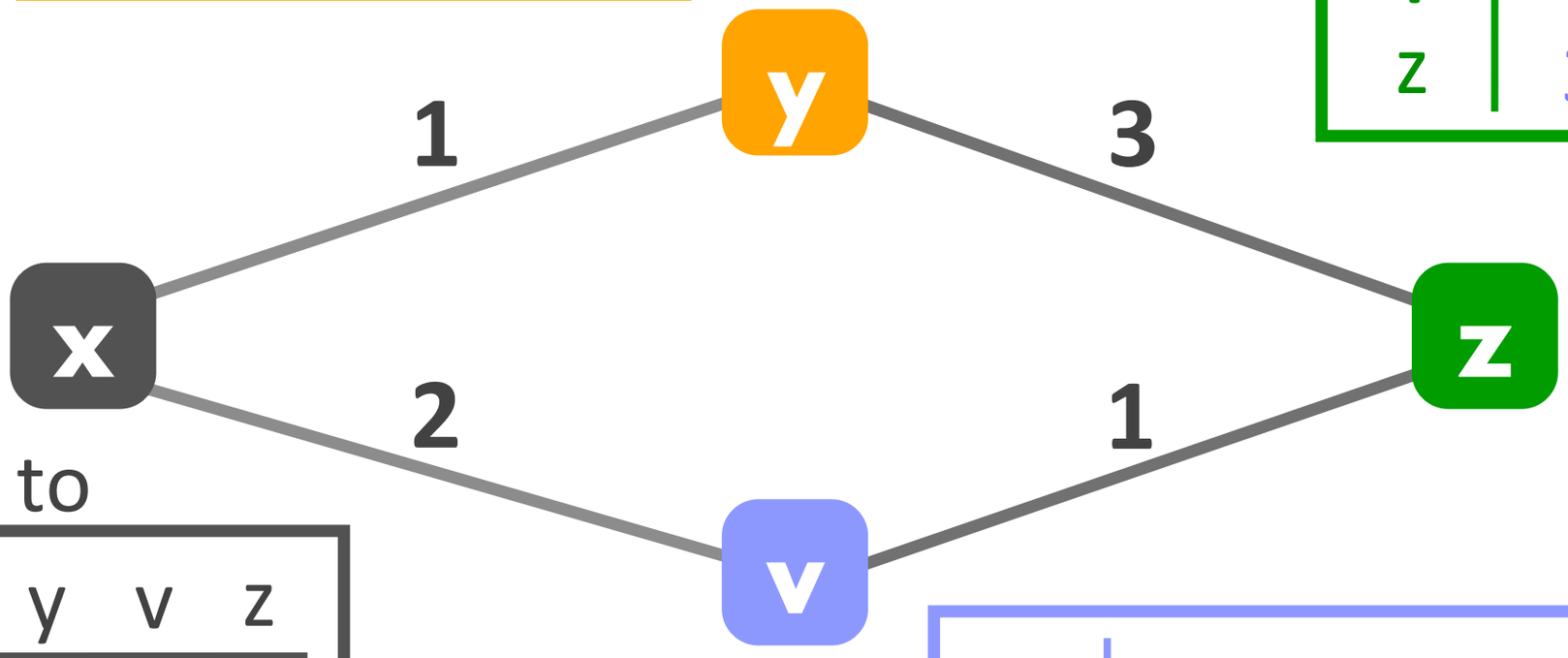
from

	x	y	v	z
x	0	1	2	3
y	1	0	-	3
v	2	-	0	1

	x	y	v	z
x	0	1	2	-
v	2	-	0	1
z	-	3	1	0

	x	y	v	z
x	0	1	2	-
y	1	0	3	3
z	-	3	1	0

	x	y	v	z
y	1	0	-	3
v	2	-	0	1
z	3	3	1	0



to

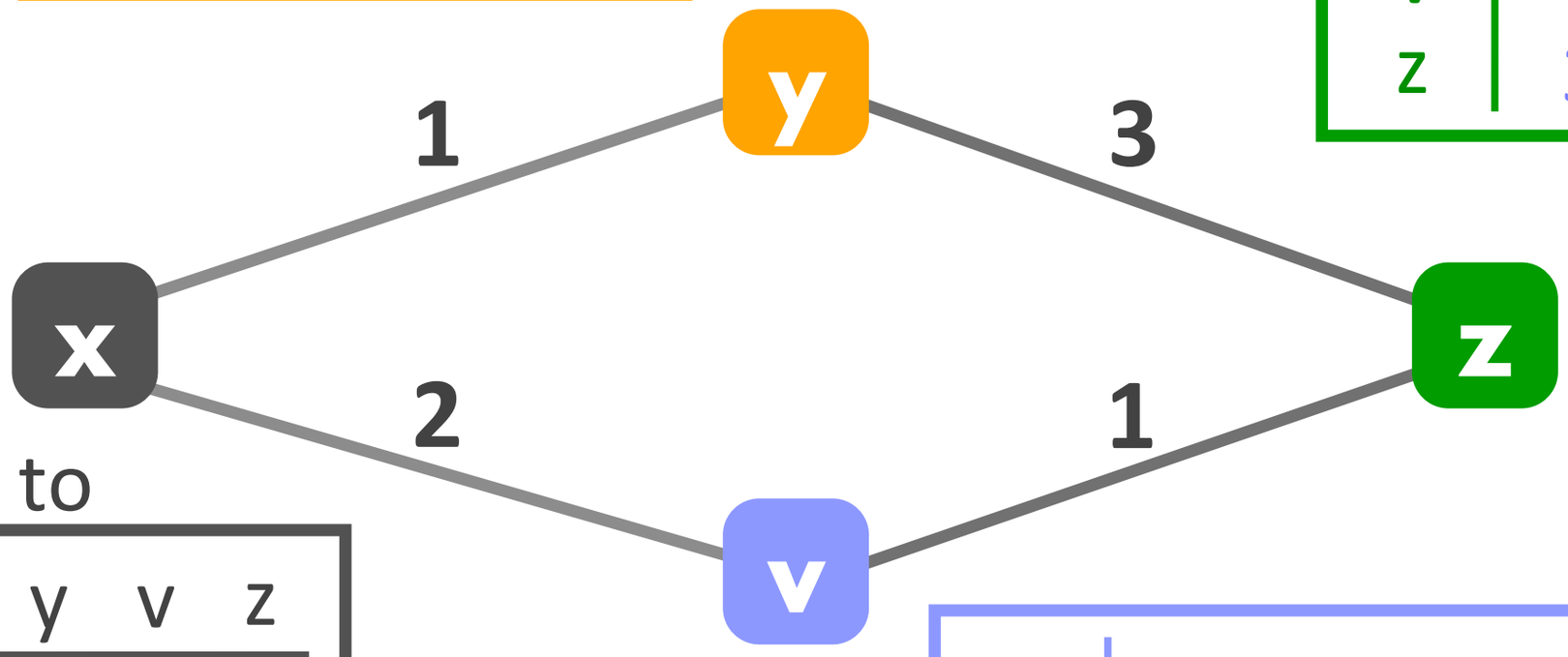
from

	x	y	v	z
x	0	1	2	3
y	1	0	-	3
v	2	-	0	1

	x	y	v	z
x	0	1	2	-
v	2	-	0	1
z	-	3	1	0

	x	y	v	z
x	0	1	2	-
y	1	0	3	3
z	-	3	1	0

	x	y	v	z
y	1	0	-	3
v	2	-	0	1
z	3	3	1	0



to

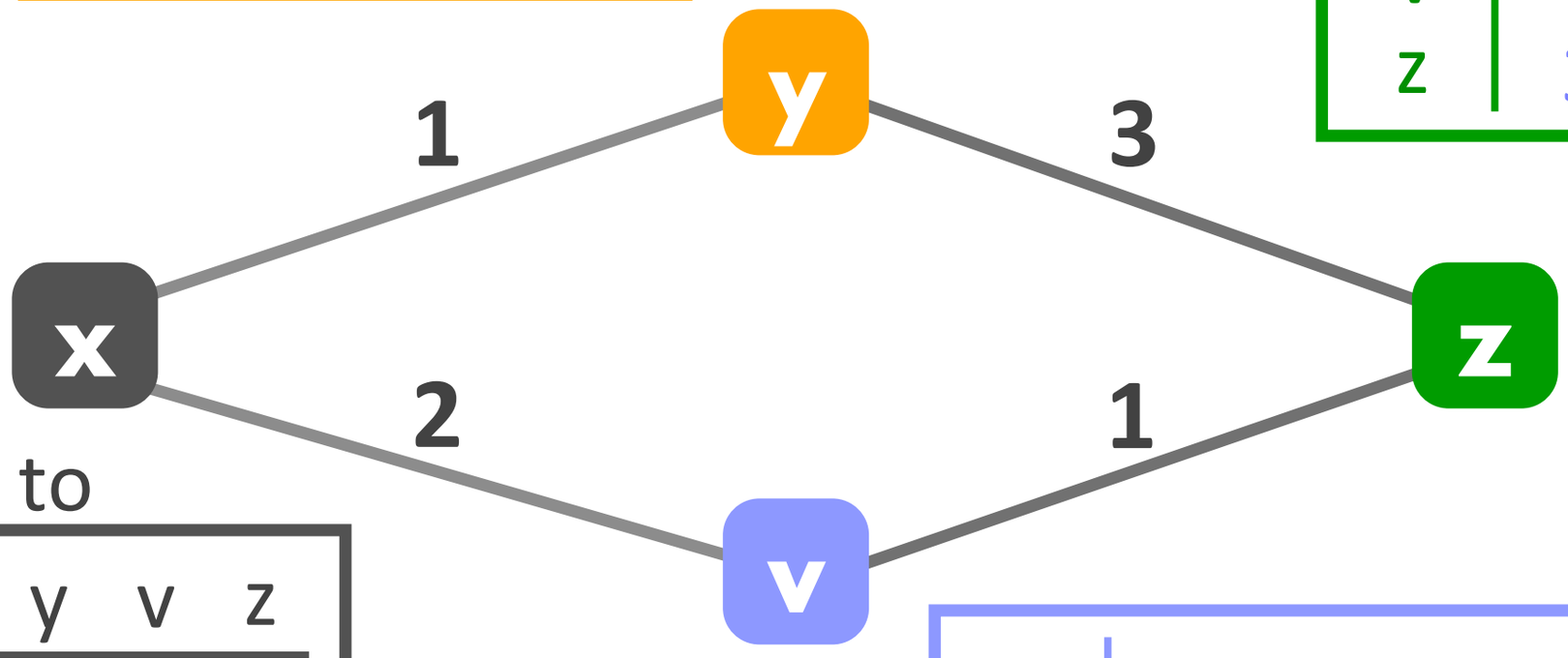
from

	x	y	v	z
x	0	1	2	3
y	1	0	-	3
v	2	-	0	1

	x	y	v	z
x	0	1	2	-
v	2	3	0	1
z	-	3	1	0

	x	y	v	z
x	0	1	2	3
y	1	0	3	3
z	3	3	1	0

	x	y	v	z
y	1	0	3	3
v	2	3	0	1
z	3	3	1	0



to

from

	x	y	v	z
x	0	1	2	3
y	1	0	3	3
v	2	3	0	1

	x	y	v	z
x	0	1	2	3
v	2	3	0	1
z	3	3	1	0

Distance-vector routing

- Input to each router:

  local link costs & neighbor messages

- Output of each router:

  least-cost path to every other router

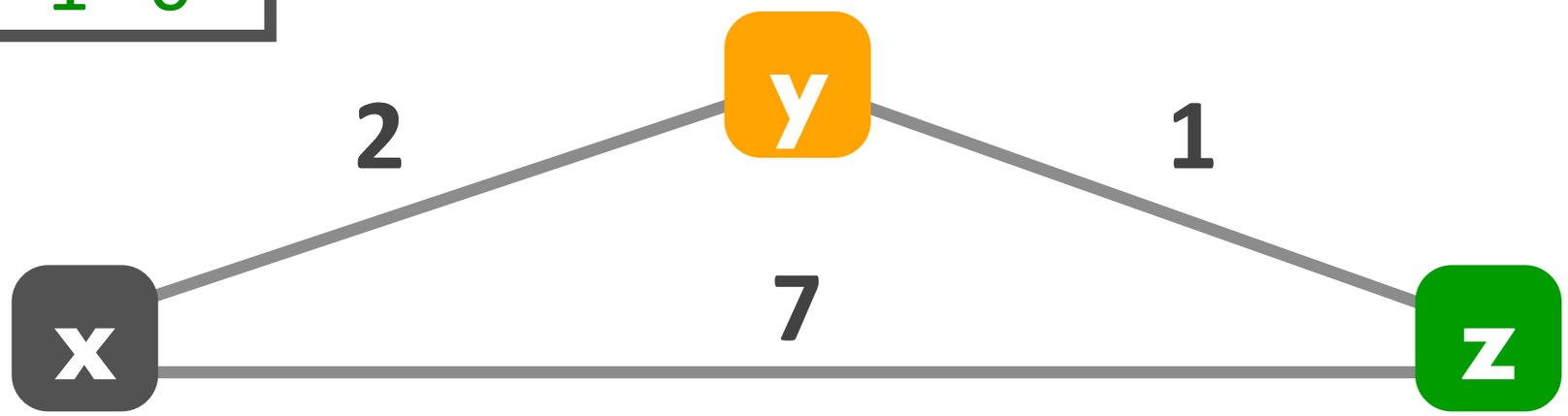
Distance-vector routing

- Distributed algorithm
- All routers run it “together”
  - *each router runs its own instance*
  - *neighbors exchange and react to each other’s messages*

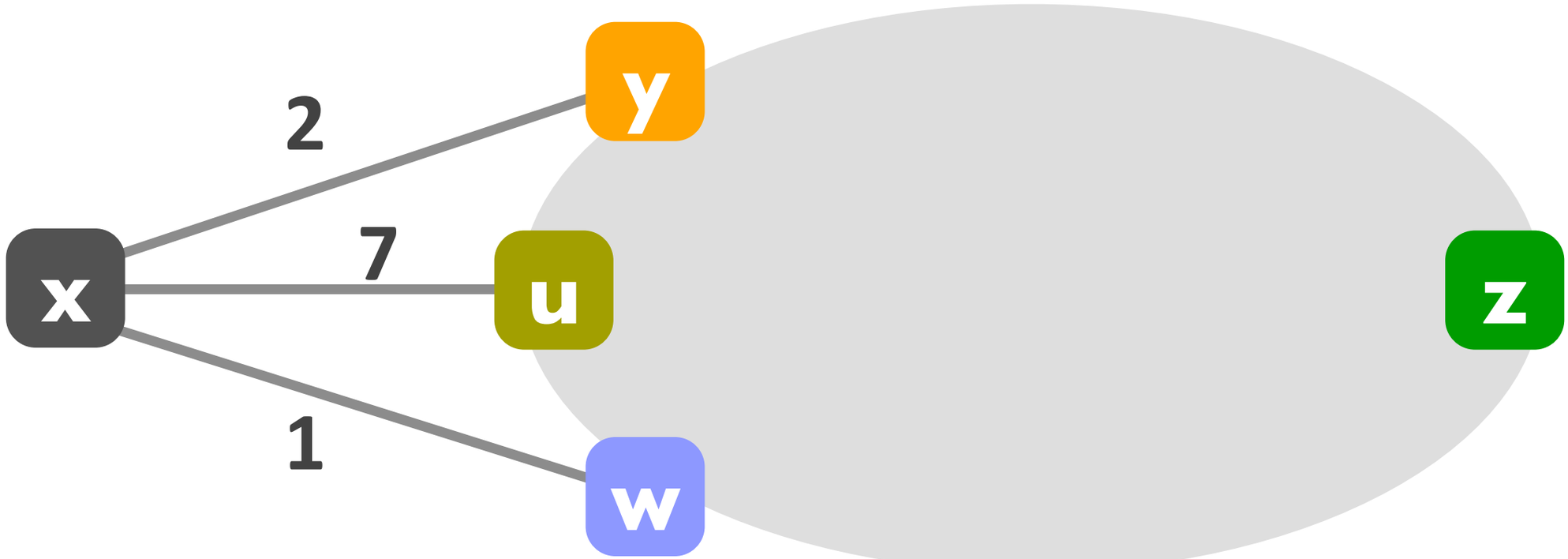
Bellman-Ford algorithm

- All neighbors exchange information
  - *Each router checks whether it can improve current paths by leveraging the new information*
- Ends when no improvement is possible

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



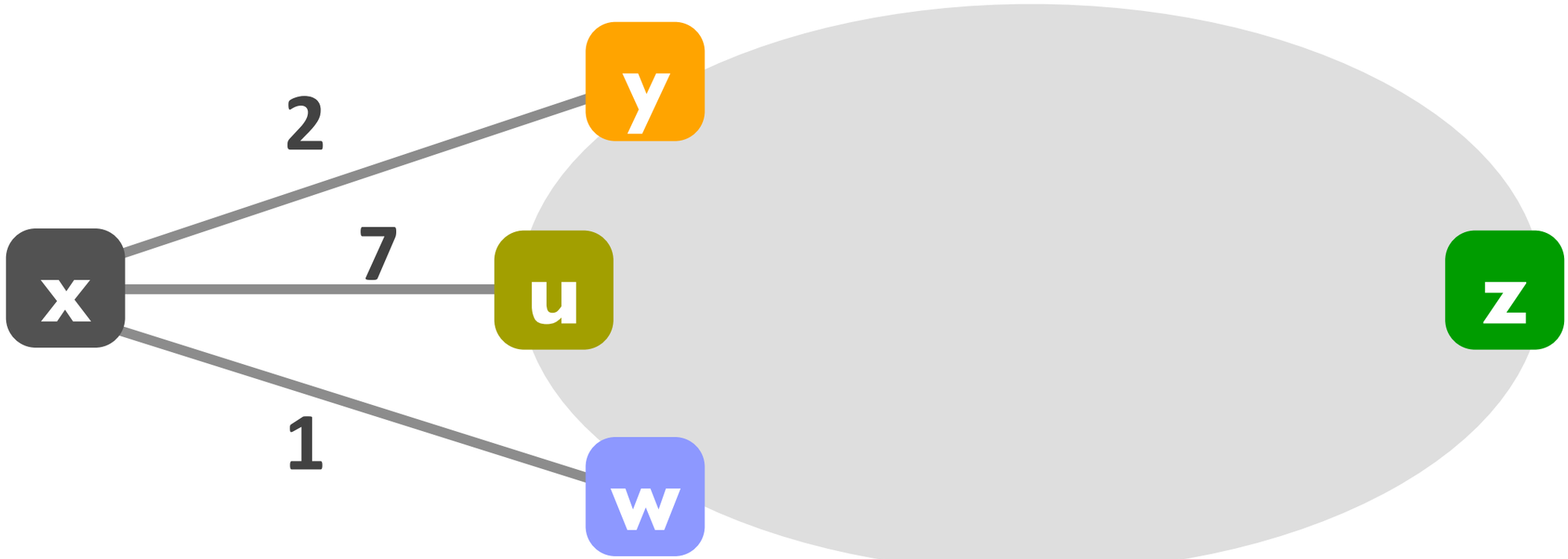
$d_x(z) = \min\{ \; cost(x,z), \; cost(x,y) + d_y(z) \; \}$



$$d_x(z) = \min\{\ cost(x,y) + d_y(z),$$

$$cost(x,u) + d_u(z),$$

$$cost(x,w) + d_w(z)\ \}$$



$$d_x(z) = \min_n\{\ cost(x,n) + d_n(z)\ \}$$

for all neighbors n

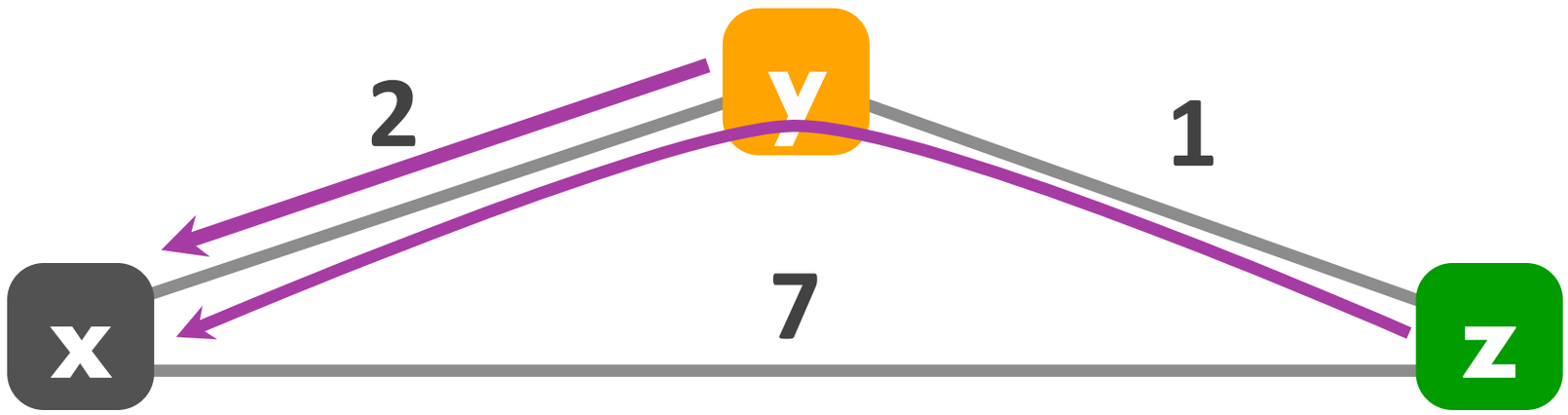
Bellman-Ford equation

Bellman-Ford equation

- Formalizes the following decision:

  - *pick as the next hop for destination z*
    the neighbor that results
    in the least-cost path to z.

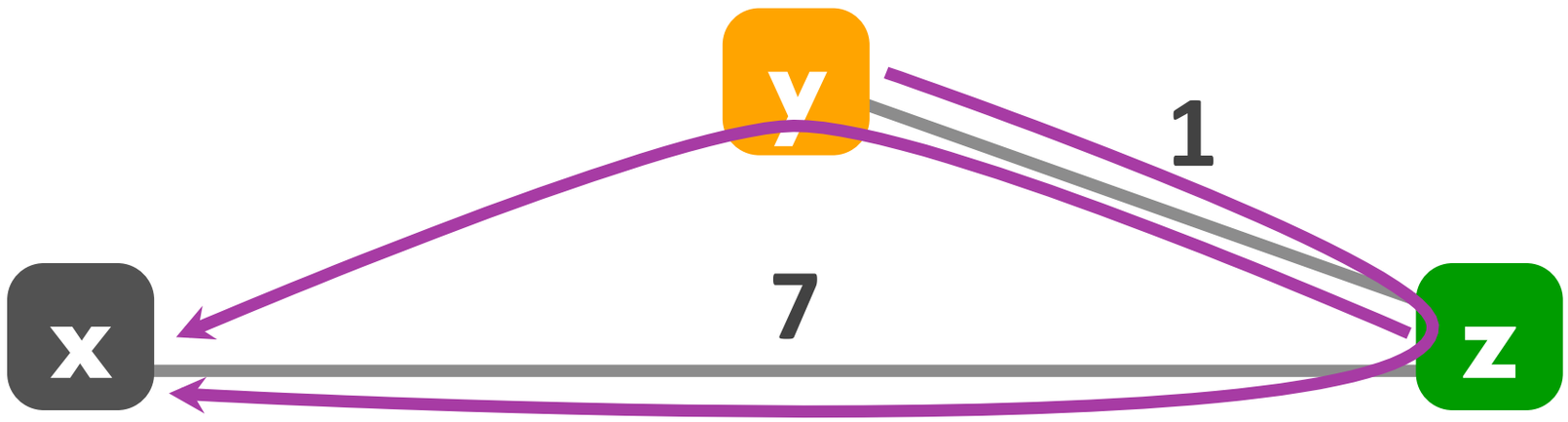
	x	y	z
y	2	0	1
z	3	1	0



	x	y	z
y	2	0	1
z	3	1	0

	x	y	z
y	4	0	1
z	3	1	0

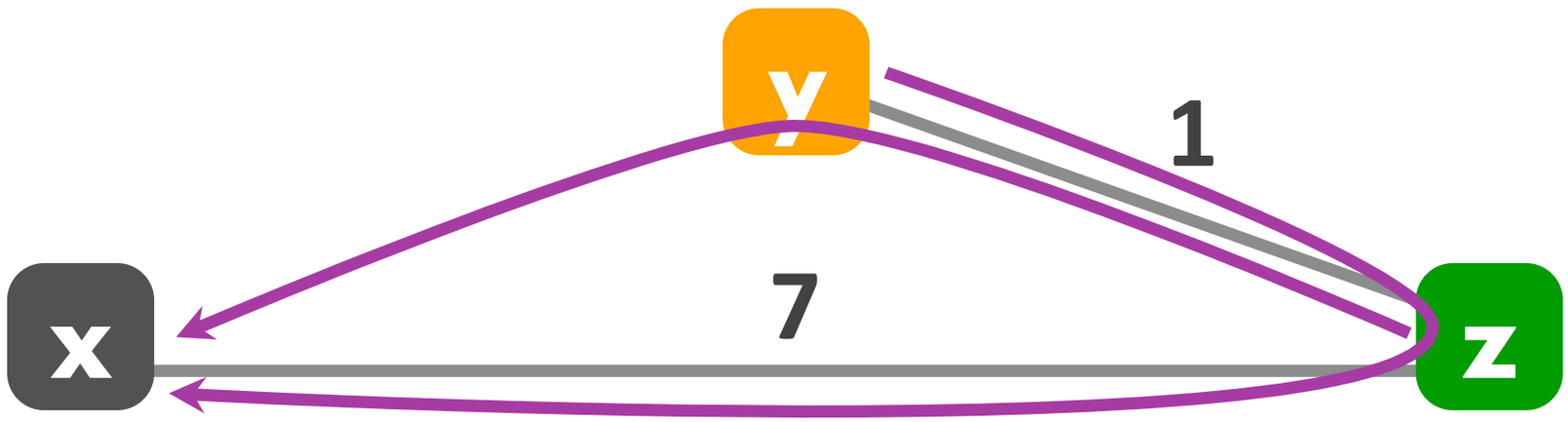
routing loop!



	x	y	z
y	2	0	1
z	3	1	0

	x	y	z
y	4	0	1
z	3	1	0

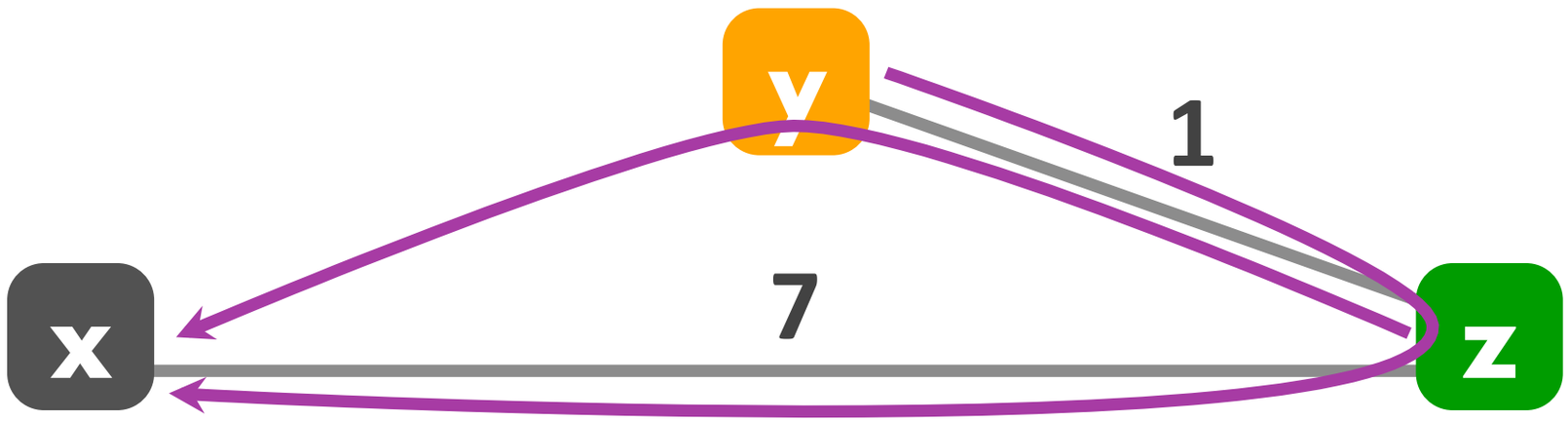
routing loop!



	x	y	z
y	4	0	1
z	3	1	0

	x	y	z
y	4	0	1
z	3	1	0

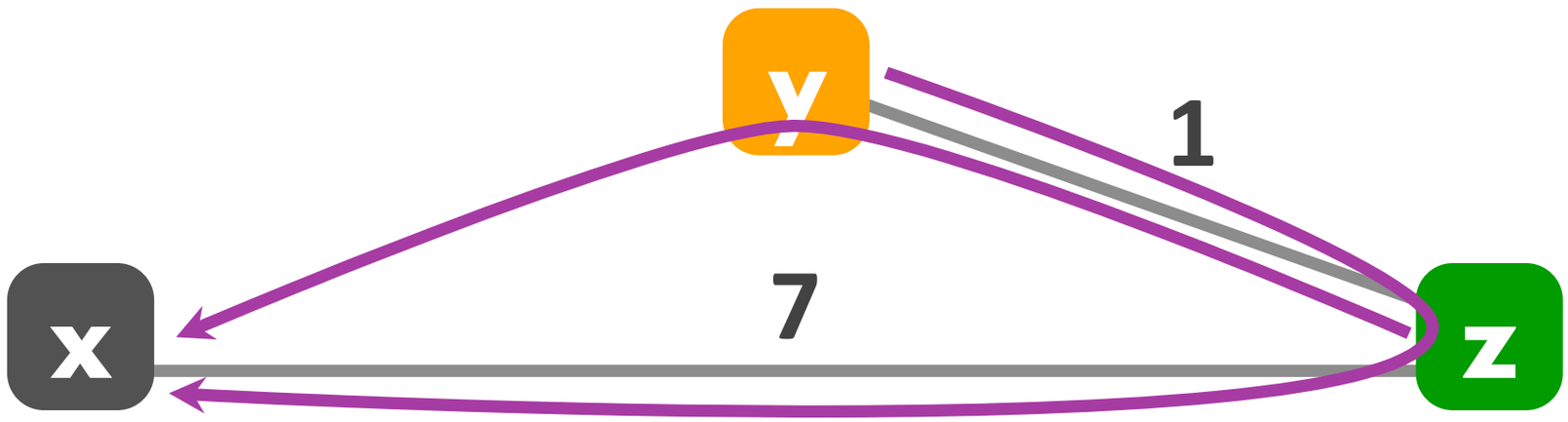
routing loop!



	x	y	z
y	4	0	1
z	5	1	0

	x	y	z
y	4	0	1
z	5	1	0

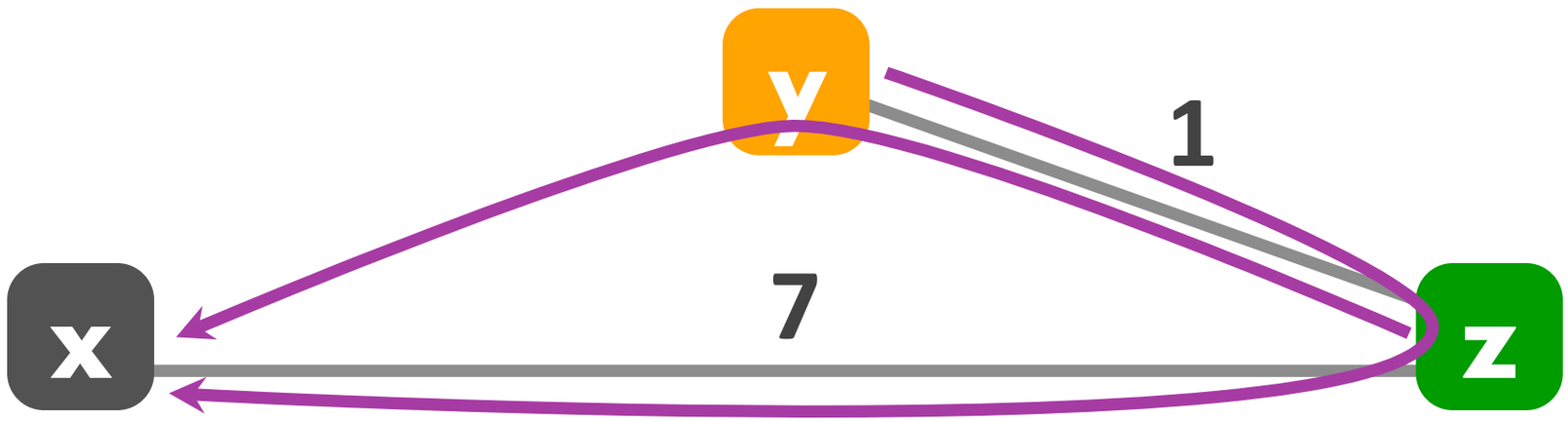
routing loop!



	x	y	z
y	4	0	1
z	5	1	0

	x	y	z
y	6	0	1
z	5	1	0

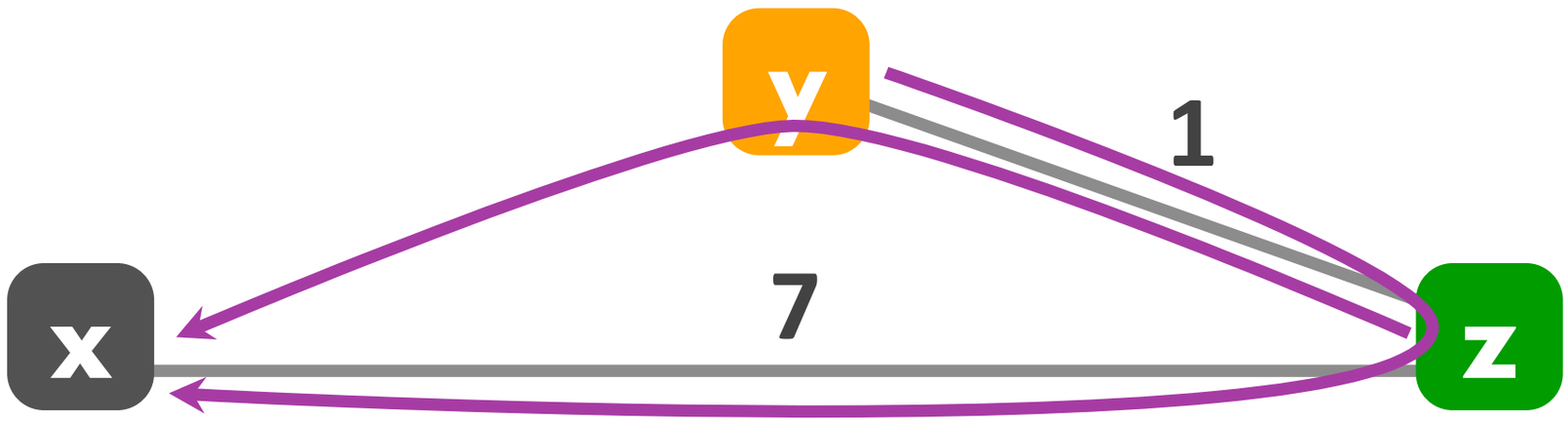
routing loop!



	x	y	z
y	4	0	1
z	5	1	0

	x	y	z
y	6	0	1
z	5	1	0

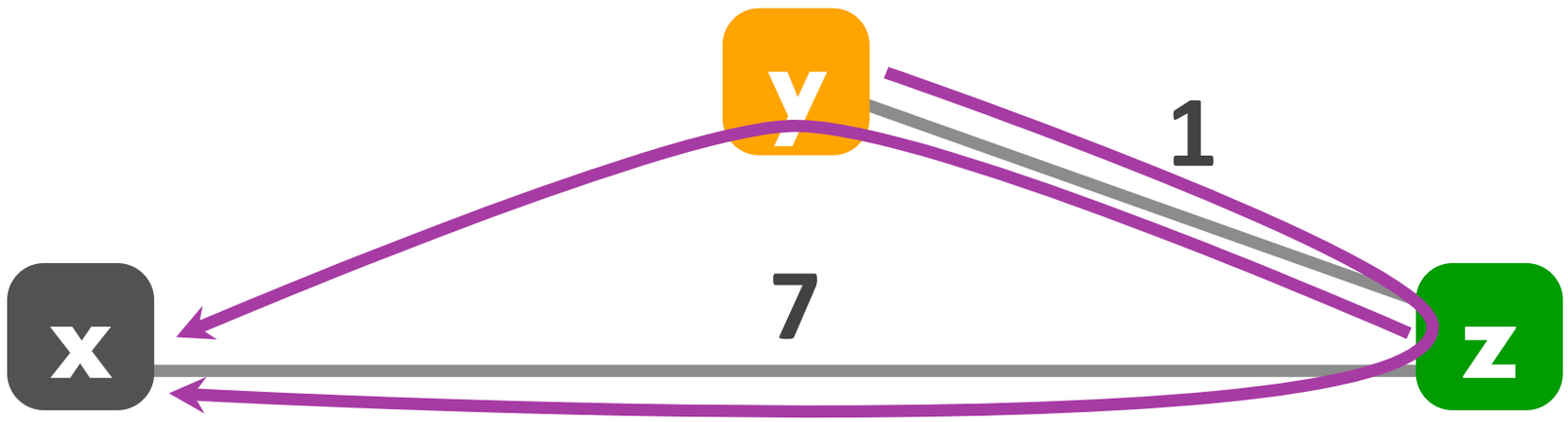
routing loop!



	x	y	z
y	6	0	1
z	5	1	0

	x	y	z
y	6	0	1
z	5	1	0

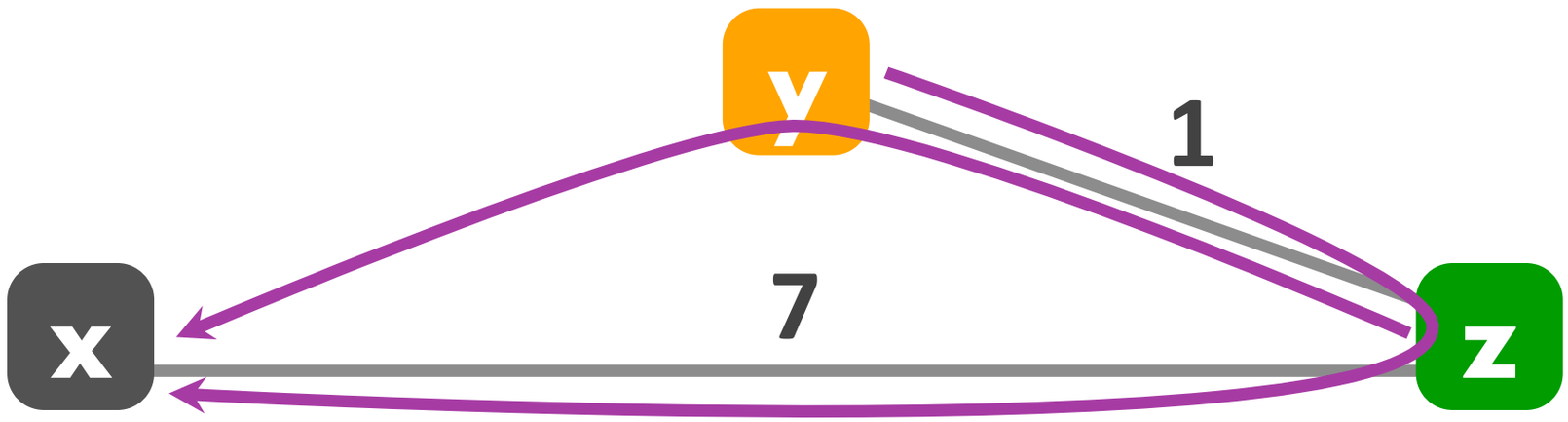
routing loop!



	x	y	z
y	6	0	1
z	7	1	0

	x	y	z
y	6	0	1
z	7	1	0

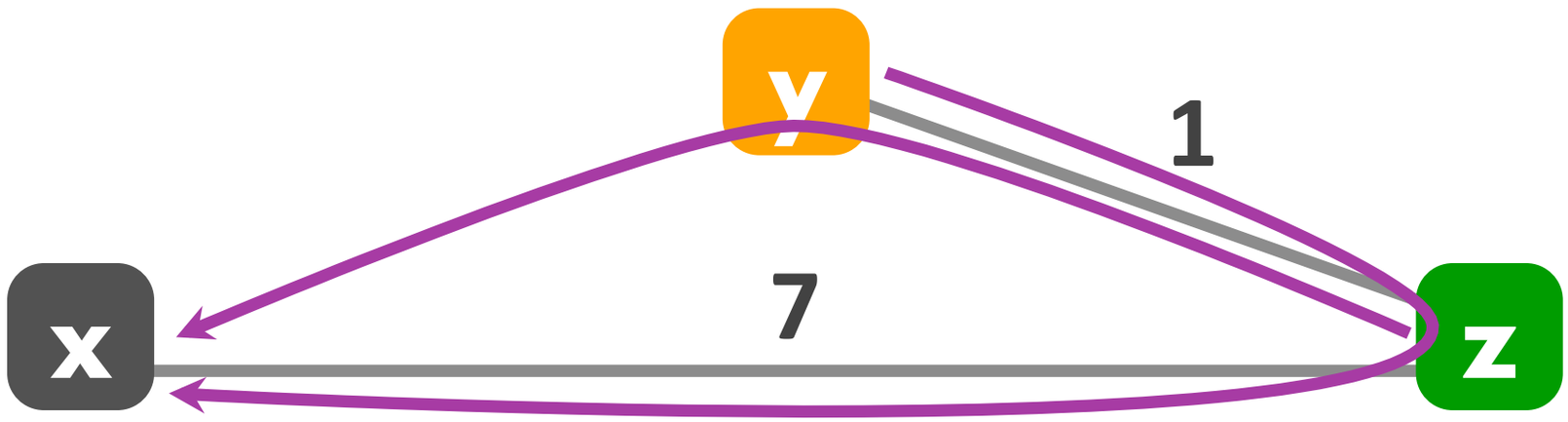
routing loop!



	x	y	z
y	6	0	1
z	7	1	0

	x	y	z
y	8	0	1
z	7	1	0

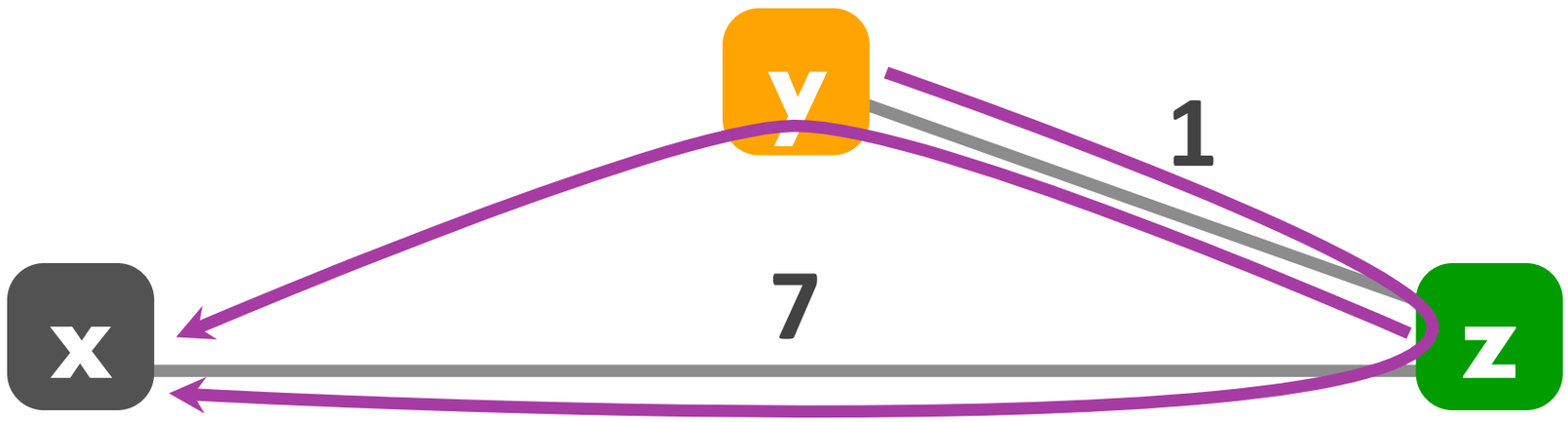
routing loop!



	x	y	z
y	6	0	1
z	7	1	0

	x	y	z
y	8	0	1
z	7	1	0

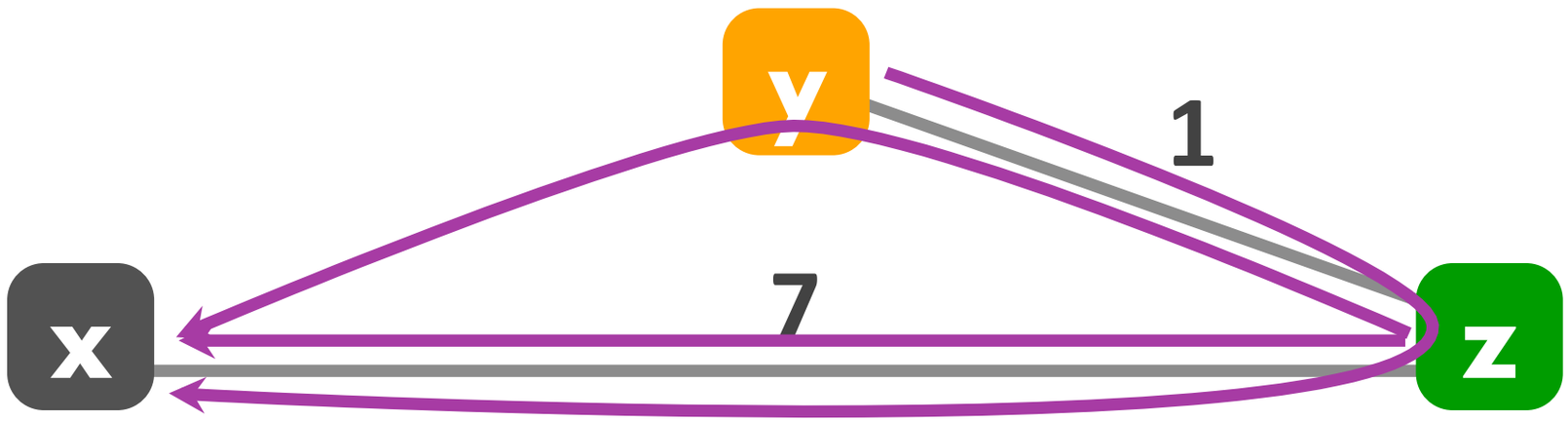
routing loop!



	x	y	z
y	8	0	1
z	7	1	0

	x	y	z
y	8	0	1
z	7	1	0

routing loop!



count-to-infinity scenario

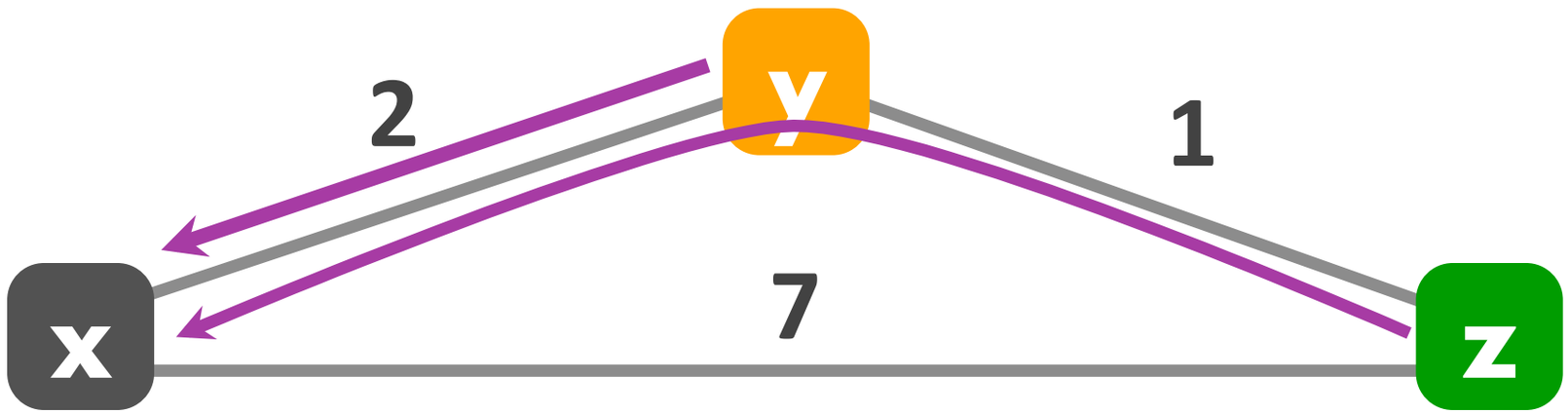
	x	y	z
y	8	0	1
z	7	1	0

Problems with Bellman-Ford

- Routing loops
  - *z routes through y, y routes through x*
  - *y loses connectivity to x*
  - *y decides to route through z*
- Can take a very long time to resolve
  - *Count-to-infinity scenario*

poisoned reverse

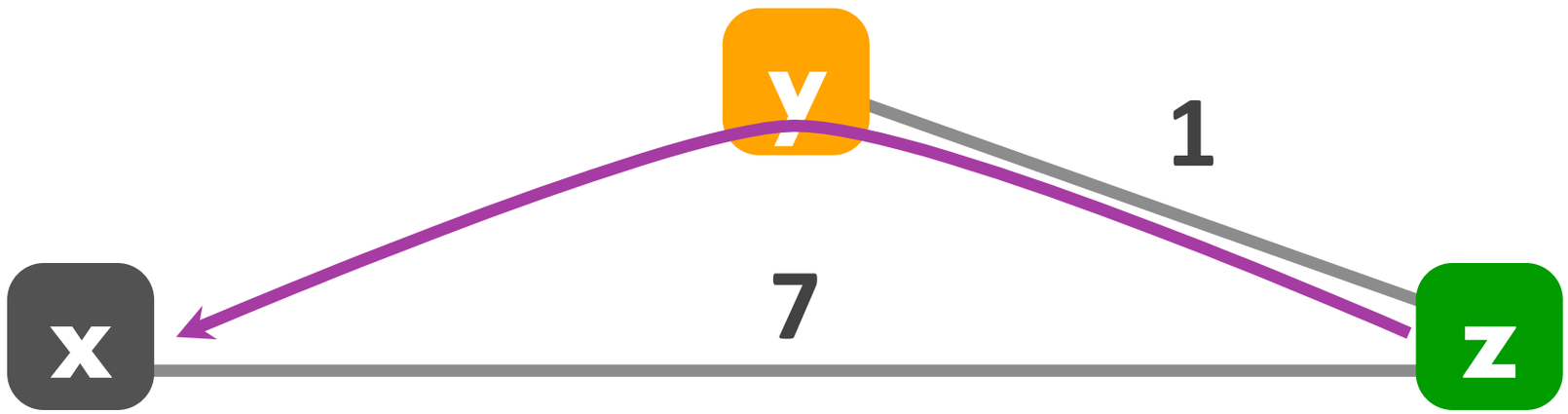
	x	y	z
y	2	0	1
z	∞	1	0



	x	y	z
y	2	0	1
z	3	1	0

poisoned reverse

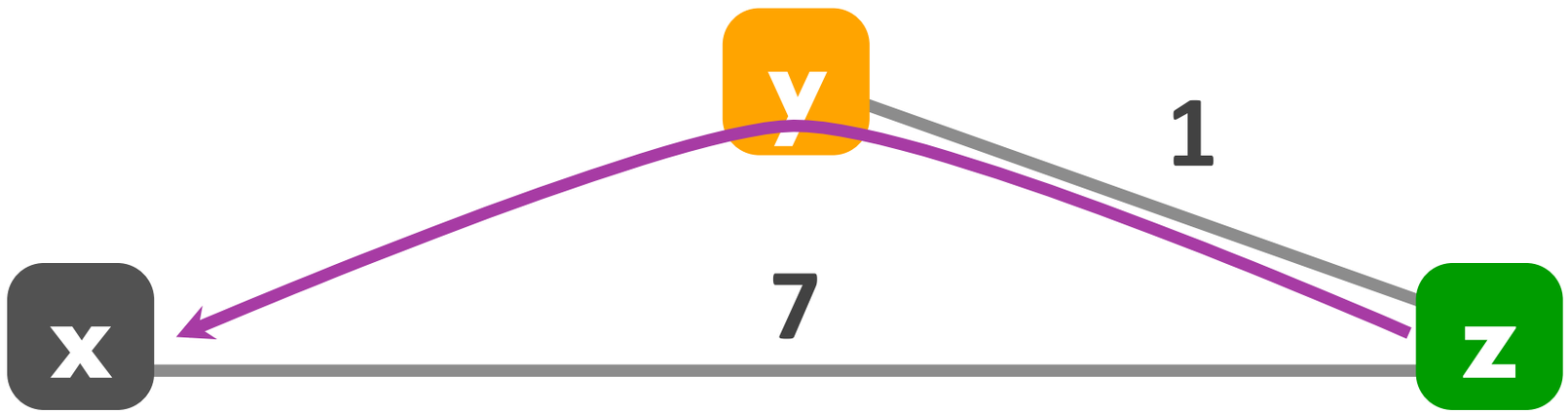
	x	y	z
y	∞	0	1
z	∞	1	0



	x	y	z
y	2	0	1
z	3	1	0

poisoned reverse

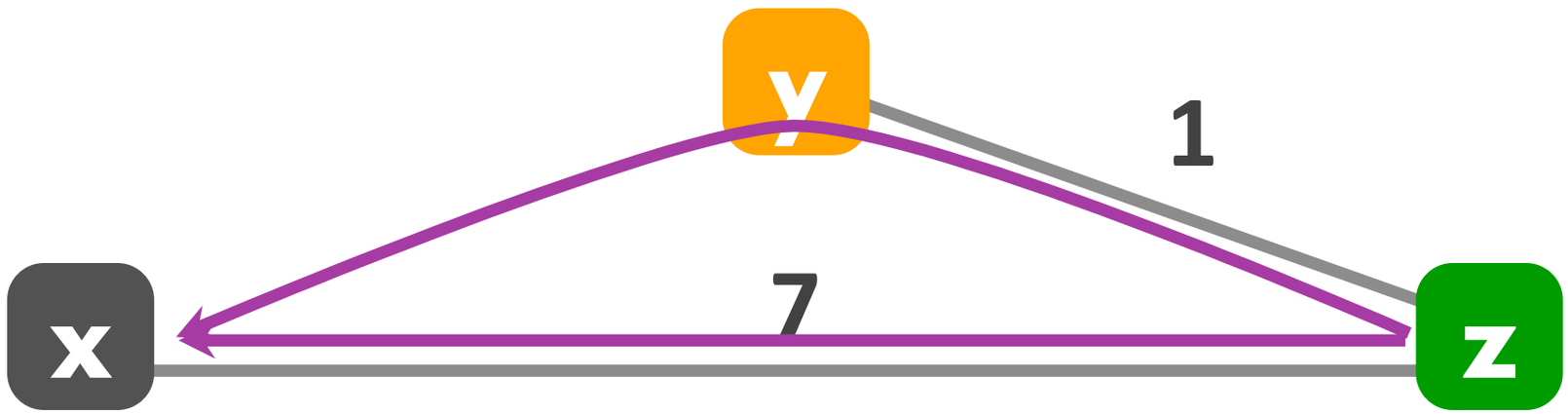
	x	y	z
y	∞	0	1
z	∞	1	0



	x	y	z
y	∞	0	1
z	3	1	0

poisoned reverse

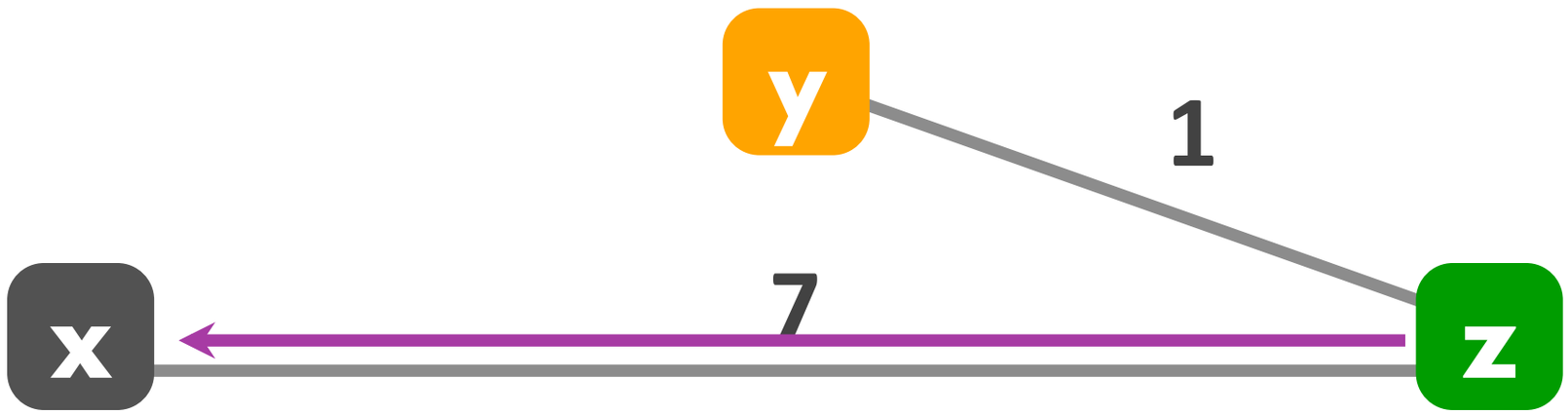
	x	y	z
y	∞	0	1
z	∞	1	0



	x	y	z
y	∞	0	1
z	7	1	0

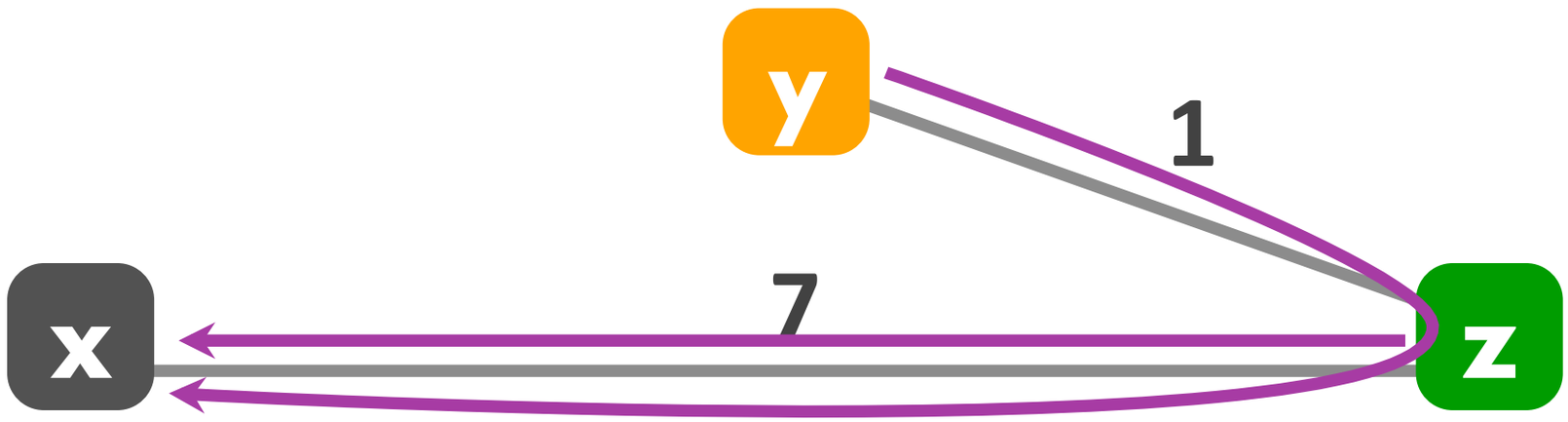
poisoned reverse

	x	y	z
y	∞	0	1
z	7	1	0



	x	y	z
y	∞	0	1
z	7	1	0

	x	y	z
y	8	0	1
z	7	1	0



poisoned reverse

	x	y	z
y	∞	0	1
z	7	1	0

Solution

- Poisoned reverse
  - *If z routes to x through y, z advertises to y that its cost to x is infinite*
  - *y never decides to route to x through z*
- Often avoids the count-to-infinity problem

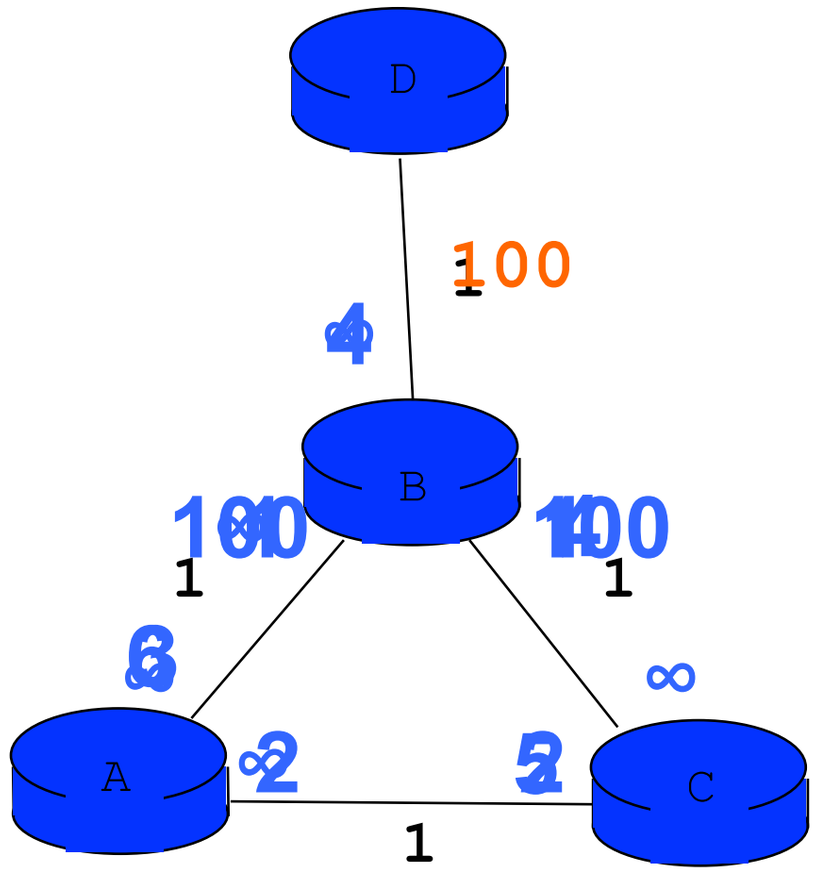
Distance Vector Routing

- Are loops possible?
  - *yes, until convergence*
  - *convergence slower than in Link-State*

- Scalability?
  - *requires fewer messages than Link-State*
  - *O(N) update time on arrival of a new DV from neighbor*
  - *O(network diameter) convergence time*
  - *O(N) entries in forwarding table*

- RIP is a protocol that implements DV (IETF RFC 2080)

Does Poison-Reverse Completely Solve the Count-to-Infinity Problem?



Outline

- Least-cost path routing
- Approach 1: link-state routing
- Approach 2: distance-vector routing
- Routing in the Internet (next time)