Lecture 8:

The Network Layer

Katerina Argyraki, EPFL

Outline

- Network-layer functions
  - forwarding
  - routing
- Network-layer types
  - virtual-circuit networks
  - datagram networks
- IP forwarding
- IP routing

Forwarding

- Local process that takes place at a router and determines output link for each packet
- How: read value from packet's network-layer header, search forwarding table for output link

Routing

- Network-wide process that populates forwarding tables
- How: routing algorithm run on a logically centralised network controller or the routers themselves

Outline

- Network-layer functions
  - forwarding
  - routing
- Network-layer types
  - virtual-circuit networks
  - datagram networks
- IP forwarding
- IP routing

Outline

- Network-layer functions
  - forwarding
  - routing
- Network-layer types
  - virtual-circuit networks
  - datagram networks
- IP forwarding
- IP routing

IP forwarding

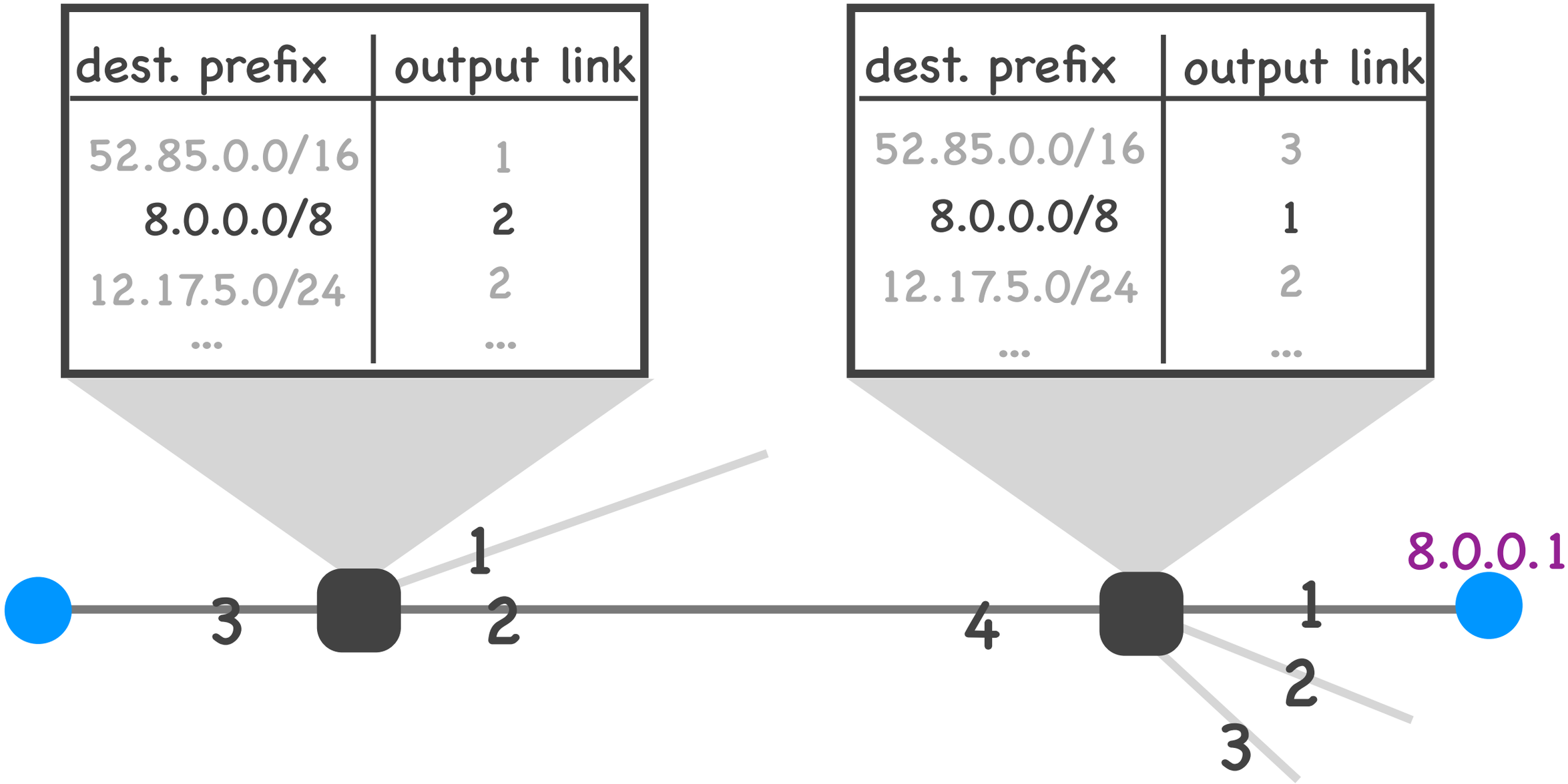
- Router's forwarding table maps IP prefixes to output links
- Reads destination IP address from packet's network-layer header
- Performs longest prefix matching
  - finds, in its forwarding table, the IP prefix that matches the dst IP address the best

Outline

- Network-layer functions
  - forwarding
  - routing
- Network-layer types
  - virtual-circuit networks
  - datagram networks
- IP forwarding
- **IP routing**

dest. prefix	output link
52.85.0.0/16	1
8.0.0.0/8	2
12.17.5.0/24	2
…	…

dest. prefix	output link
52.85.0.0/16	3
8.0.0.0/8	1
12.17.5.0/24	2
…	…



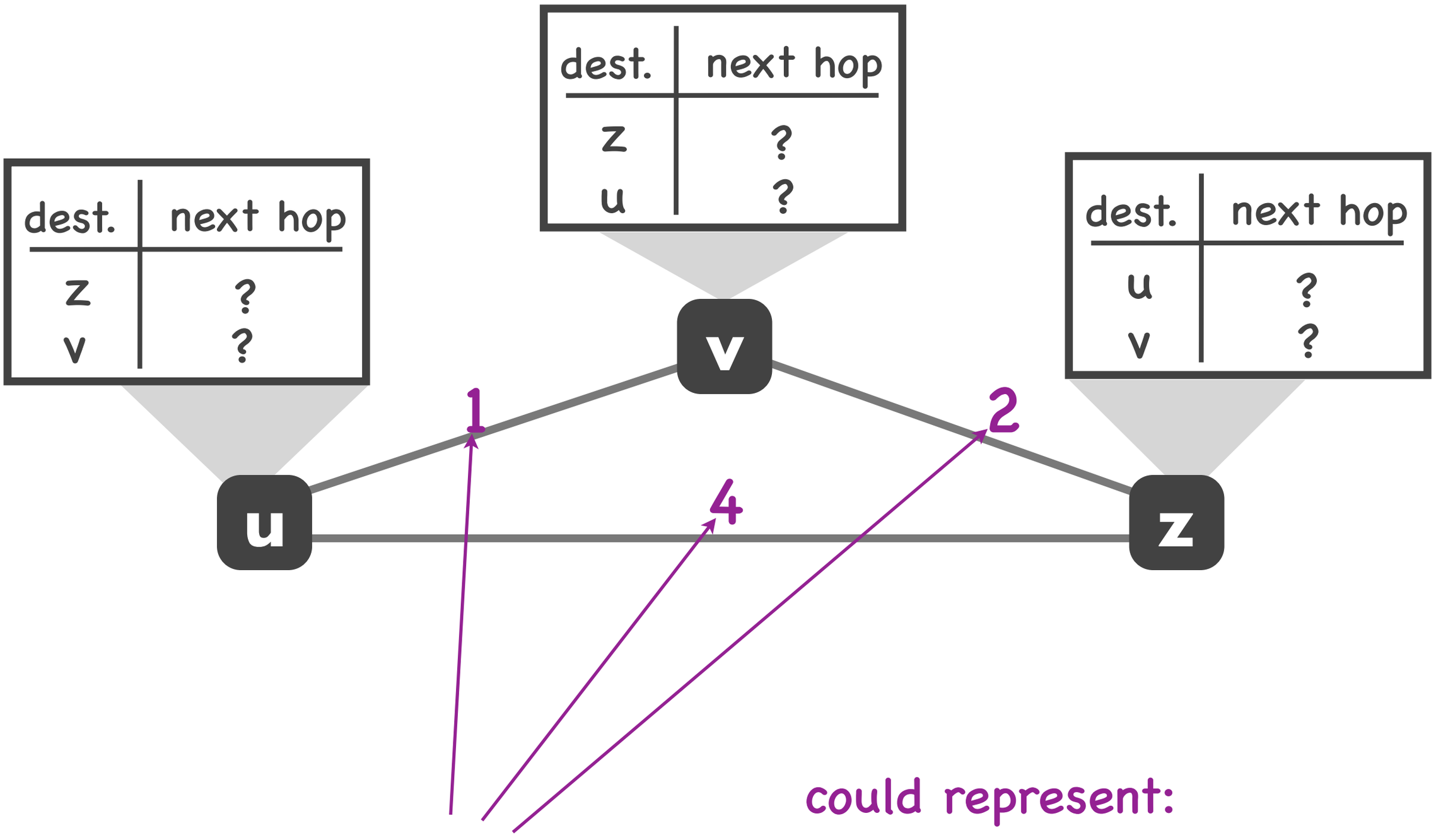
dest.	out. link
5.0.0.0/8	3
8.0.0.0/8	2

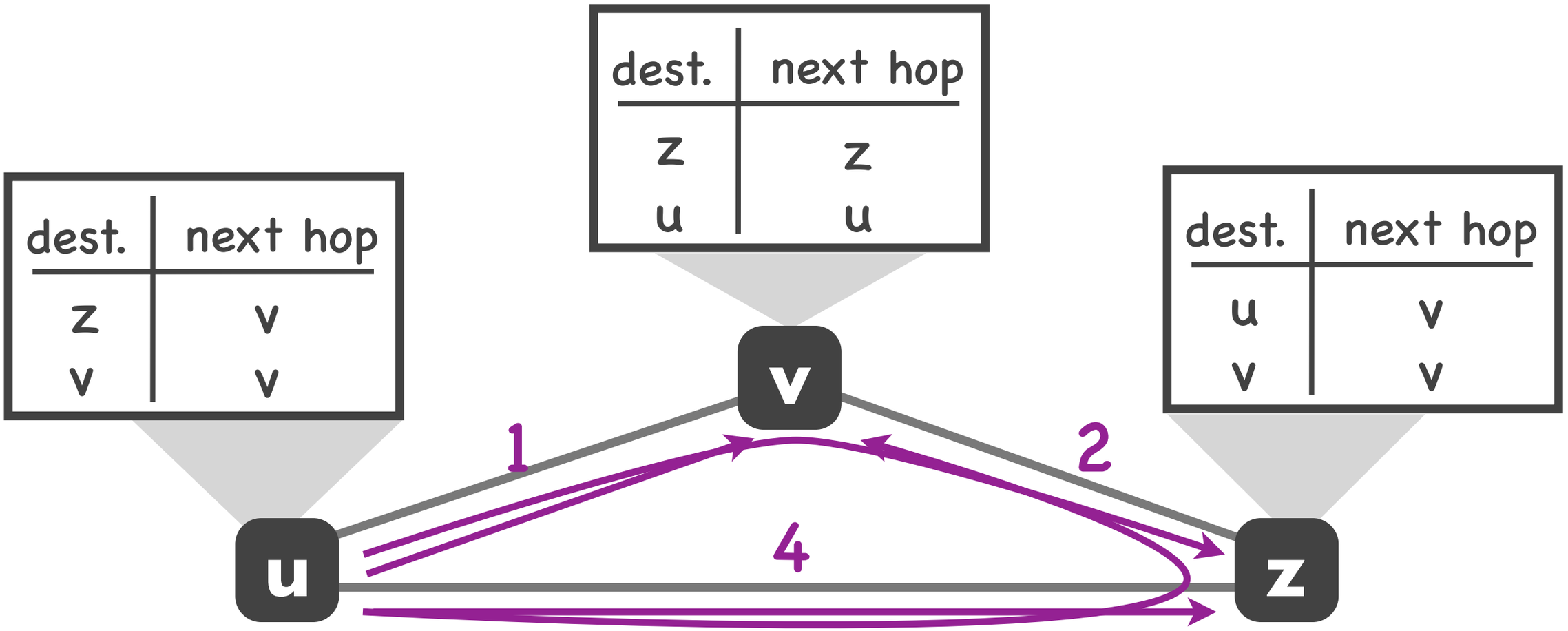
dest.	out. link
8.0.0.0/8	1
5.0.0.0/8	2

first-hop router for IP subnet 5.0.0.0/8

first-hop router for IP subnet 8.0.0.0/8


u
z


dest. | next hop
z | ?
u | ?
dest. | next hop
z | ?
v | ?
dest. | next hop
u | ?
v | ?
v
u
z
1
2
4
link costs
could represent:
propagation delay
usage fee
…



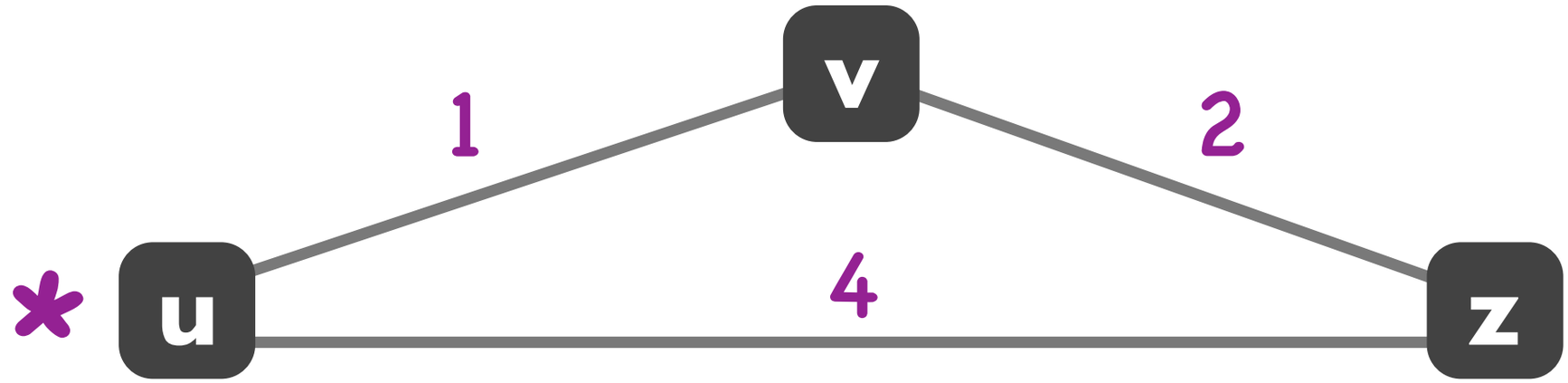
least-cost path from u to z: u v z

least-cost path from u to v: u v

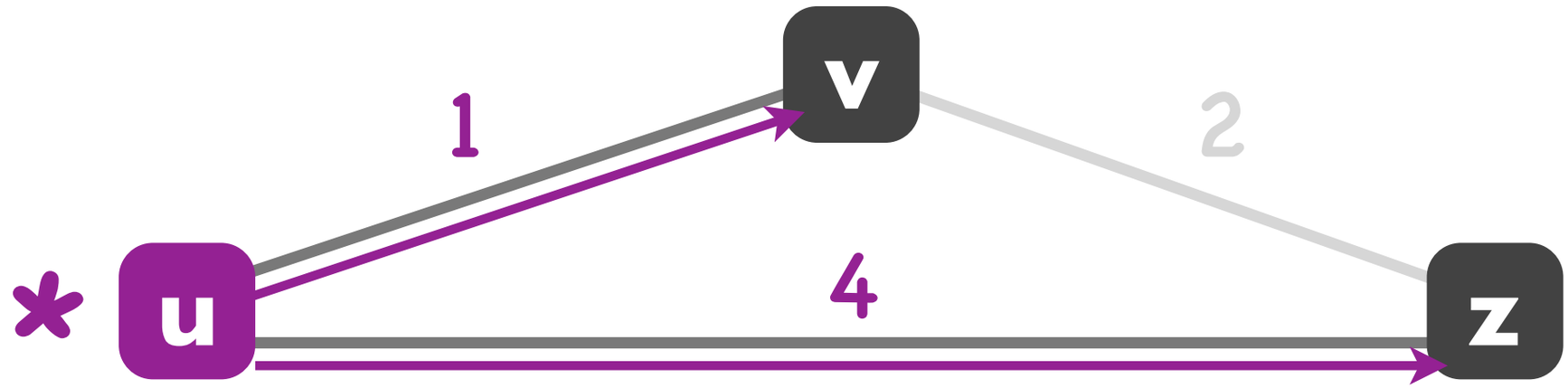
Least-cost path routing

- Goal: find least-cost path
  from each source router to each
  destination router

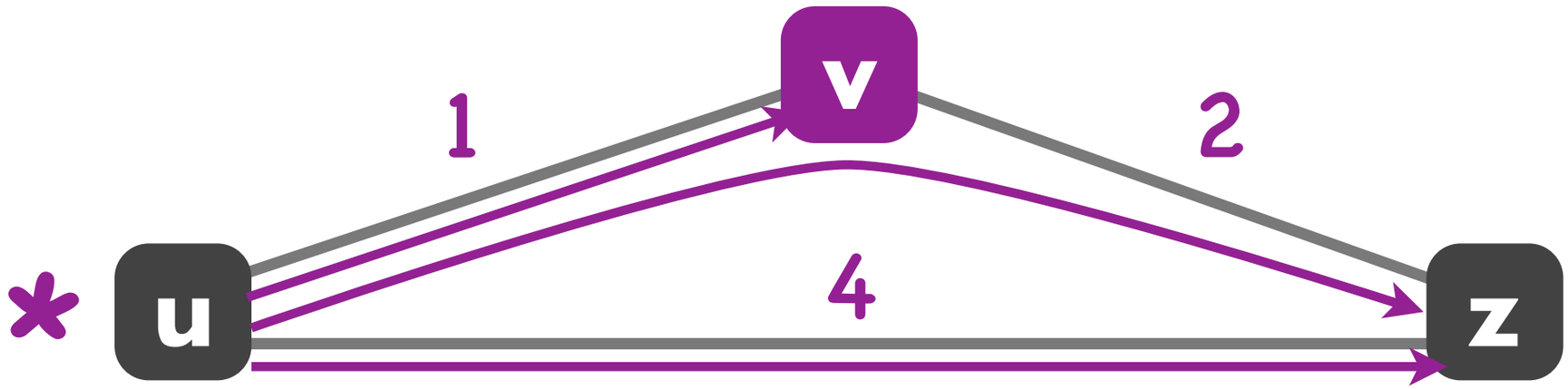
dest.	next hop	cost
z		
v		



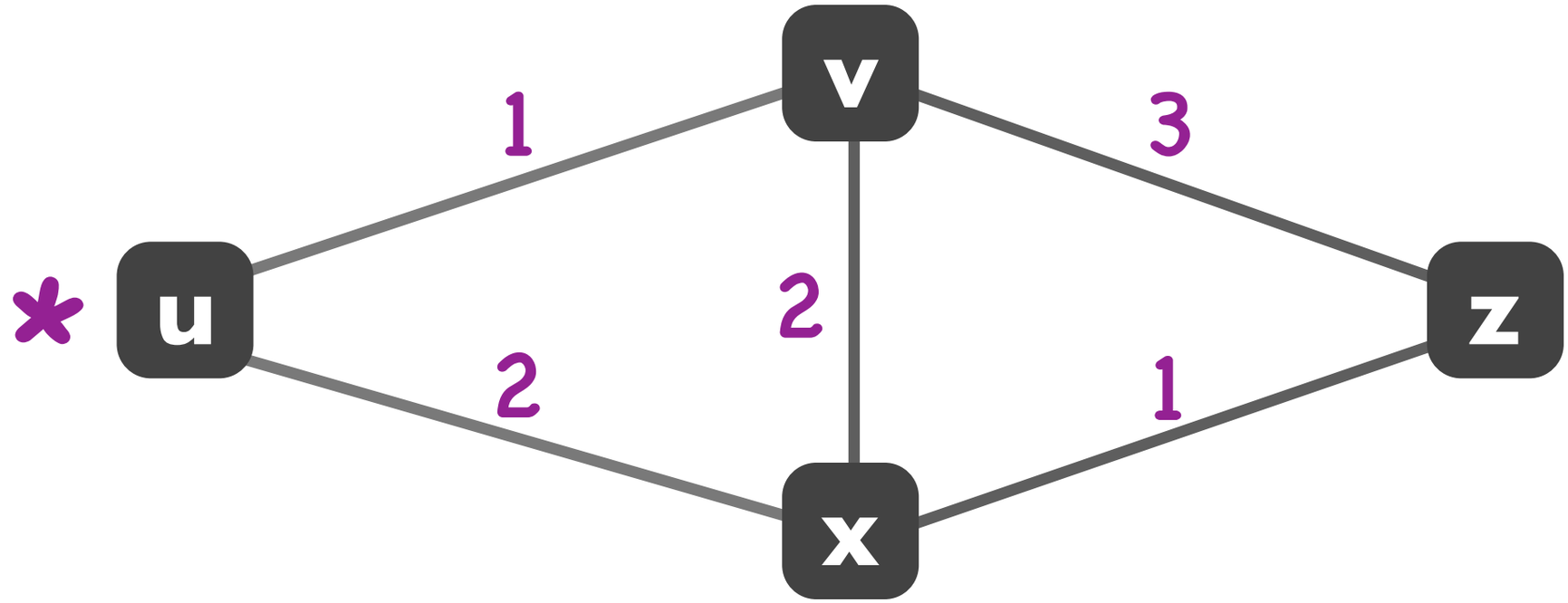
dest.	next hop	cost
z	z	4
v	v	1



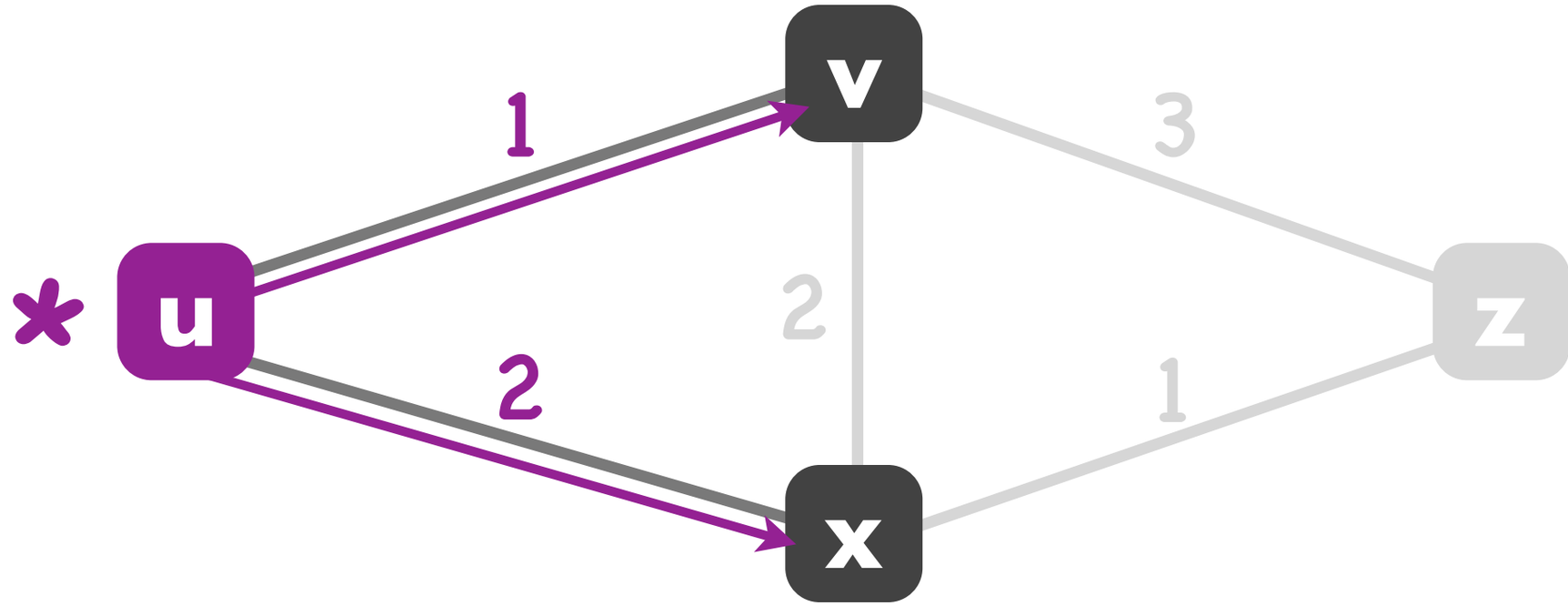
dest.	next hop	cost
z	~~z~~ v	~~4~~ 3
v	v	1



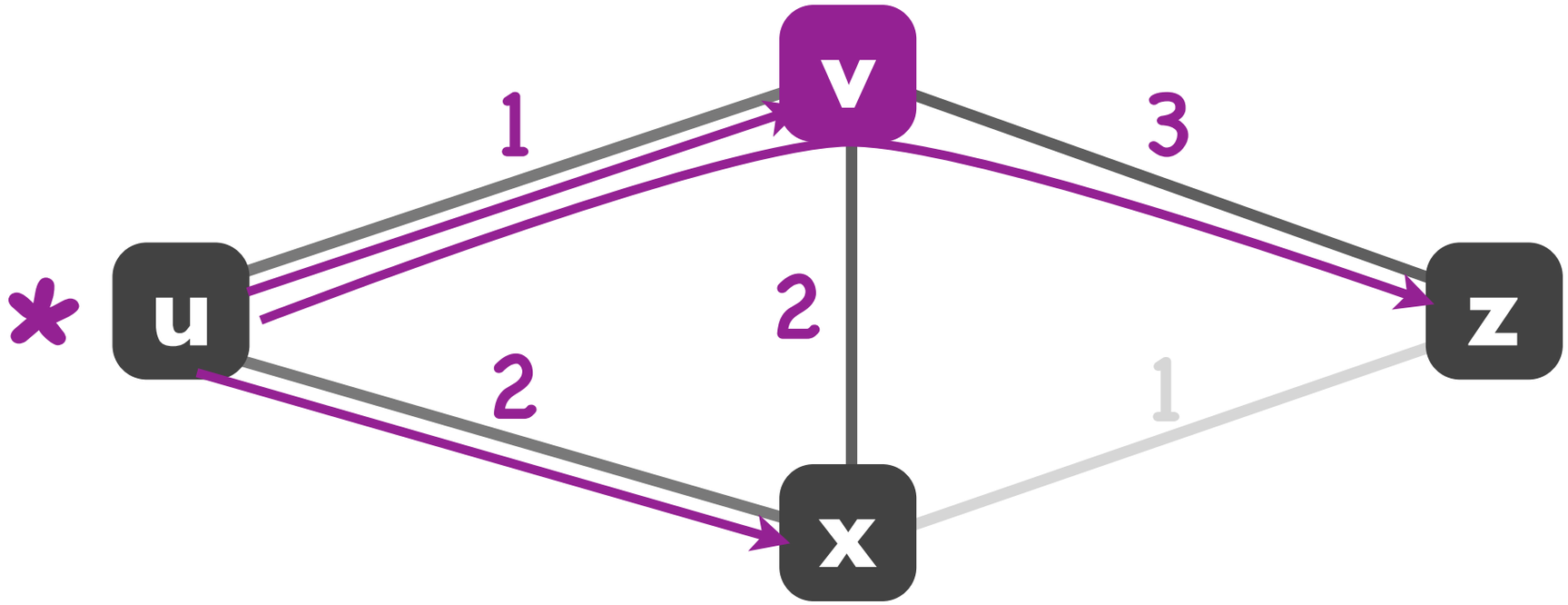
dest.	next hop	cost
z		
v		
x		



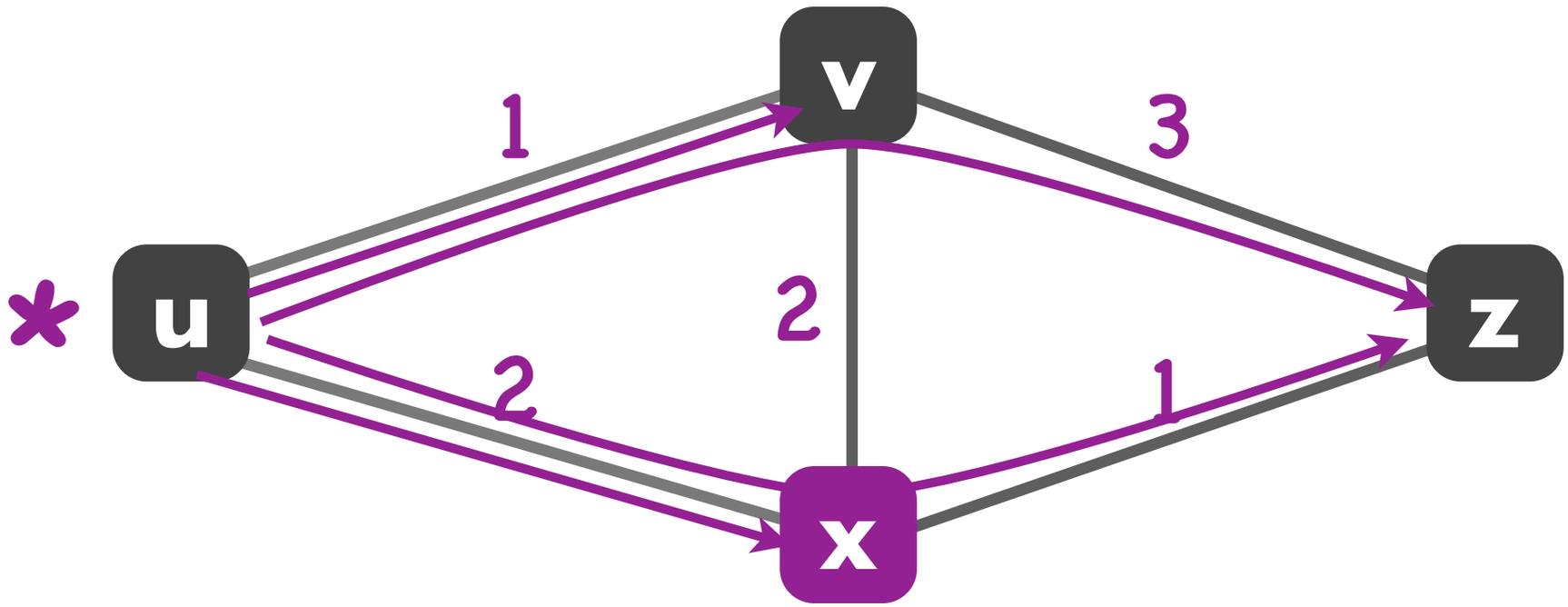
dest.	next hop	cost
z	-	-
v	v	1
x	x	2



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



Link-state routing algorithm for source u

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

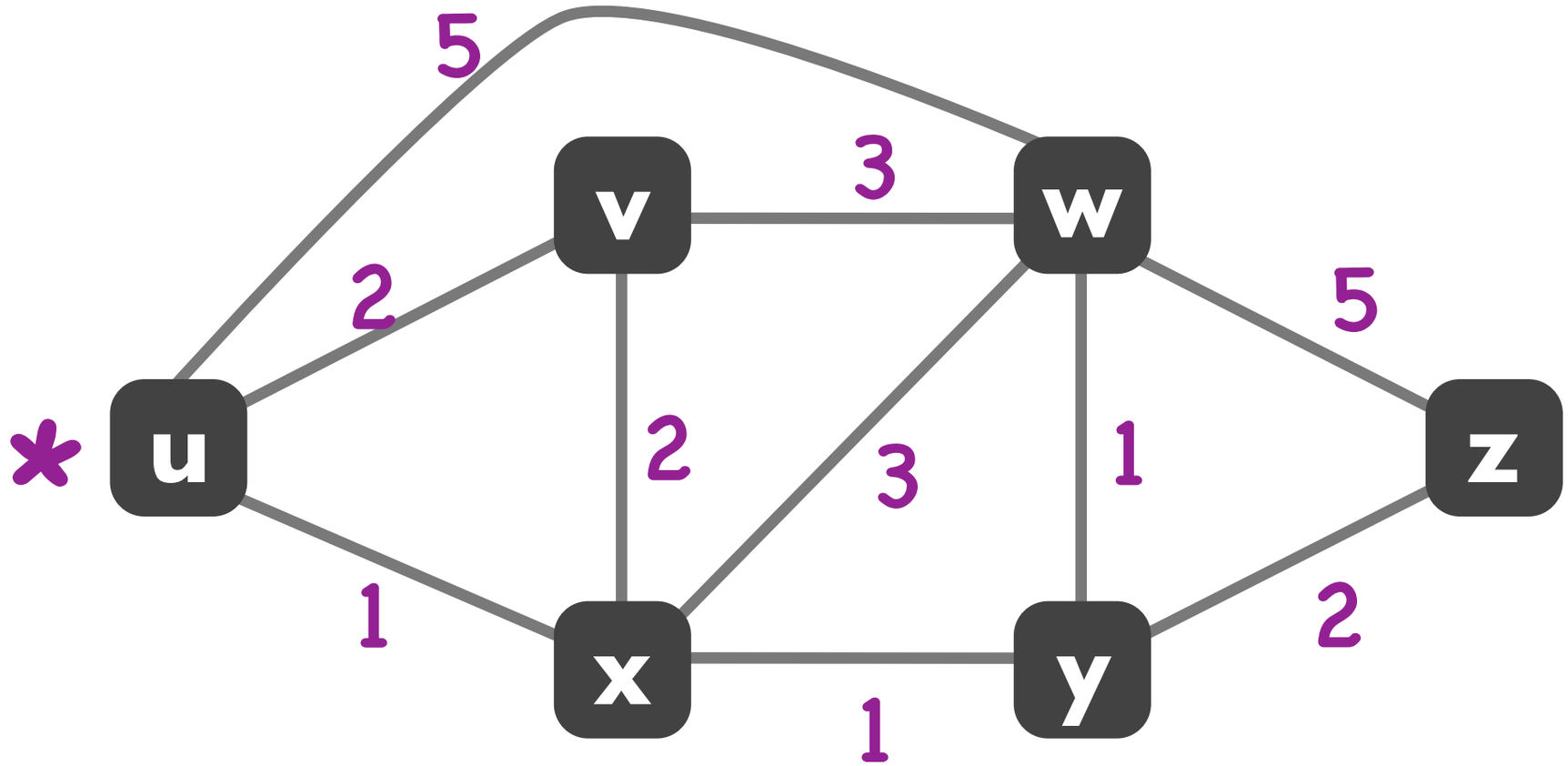
Link-state routing algorithm for source u

- "Centralized" algorithm: runs on a single entity
- Option #1: Router u runs the algorithm
- Option #2: Separate computer ("network controller") runs the algorithm for all the routers

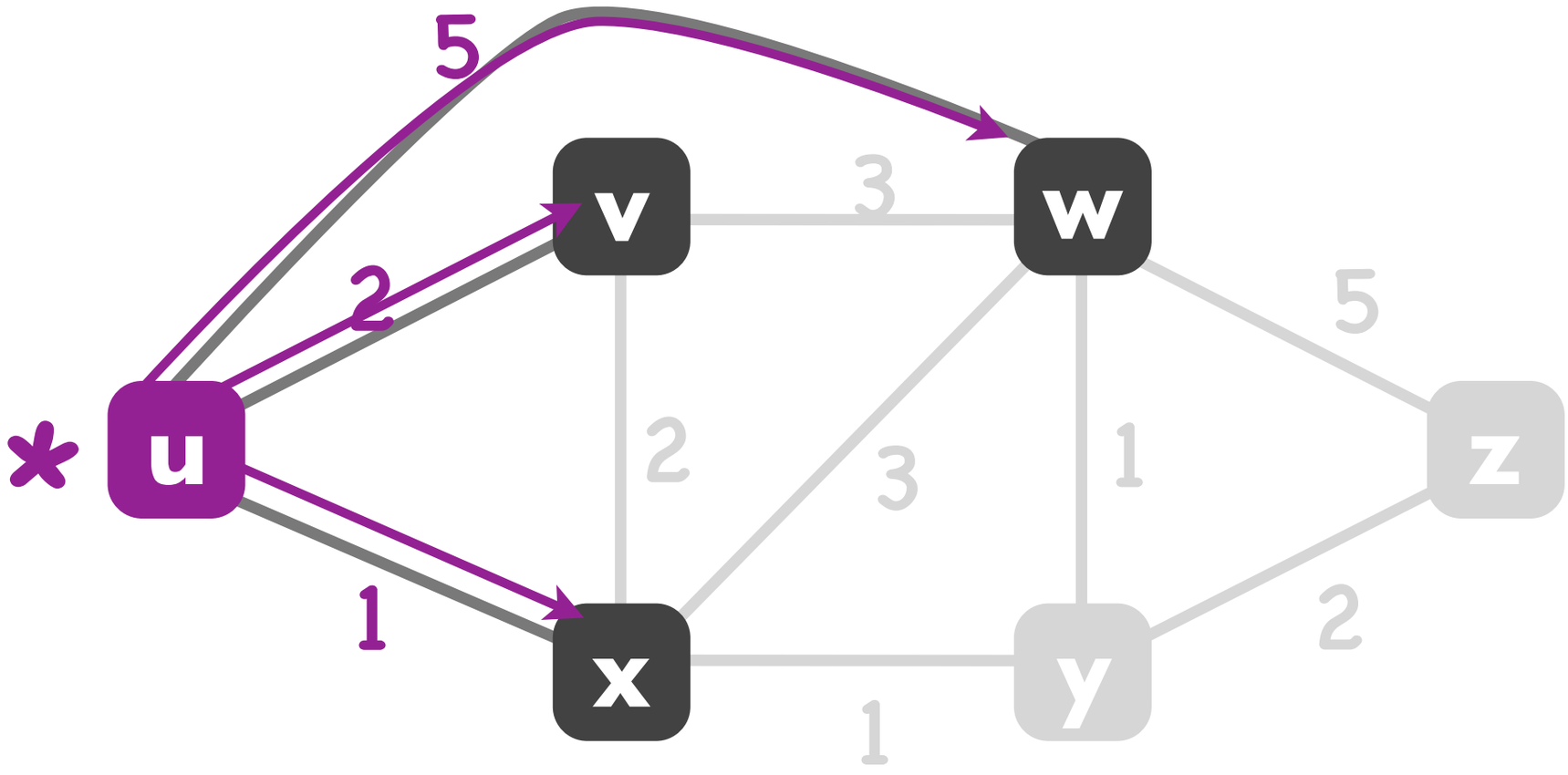
Dijkstra's algorithm

- At each step, consider a new router
  - starting from "closest" neighbor
- Check whether current paths can be improved
  - by using that router as an intermediate point
- End when no improvement is possible

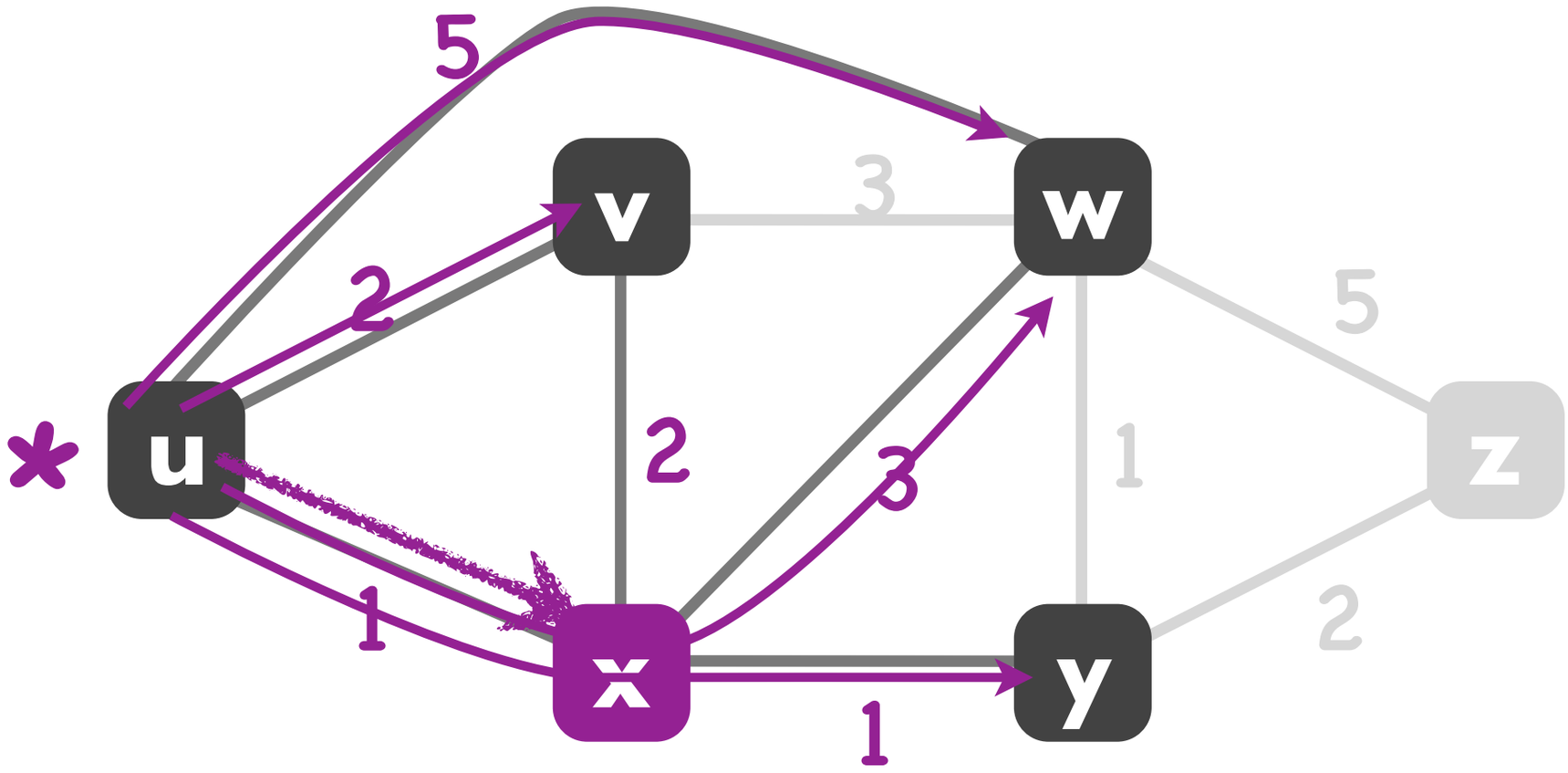
dest.	next hop	cost
z		
w		
y		
v		
x		



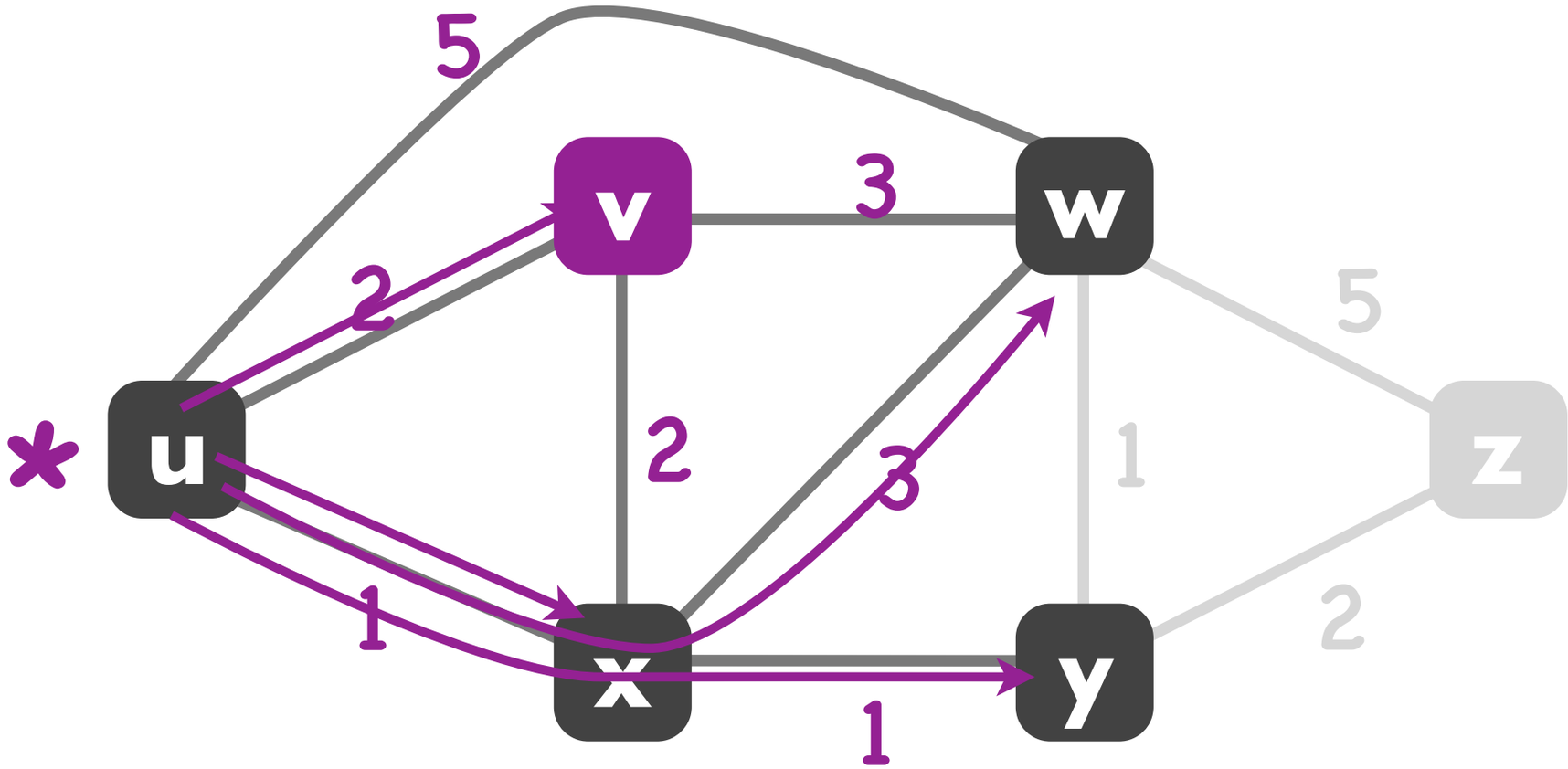
dest.	next hop	cost
z	-	-
w	w	5
y	-	-
v	v	2
x	x	1



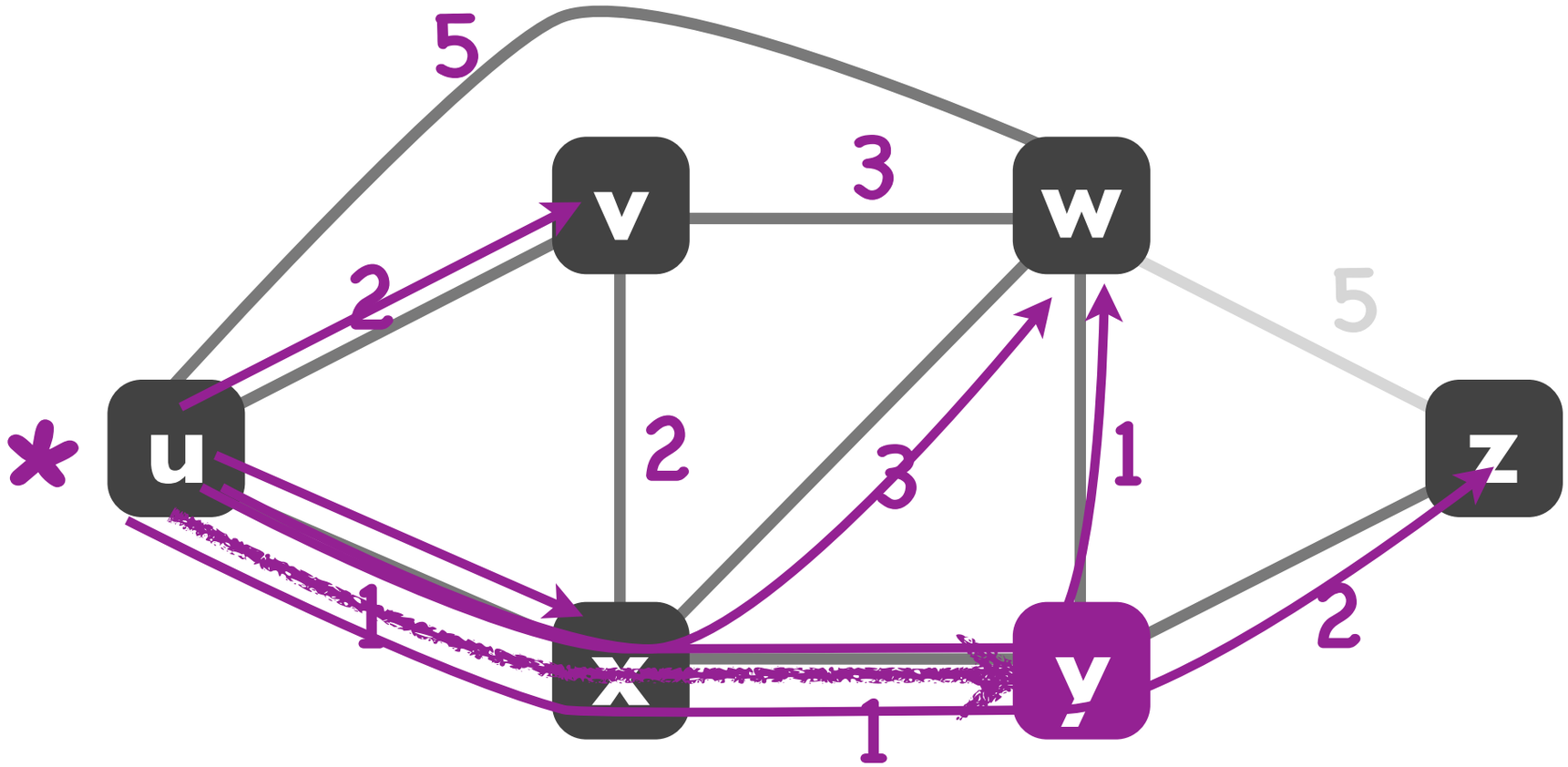
dest.	next hop	cost
z	-	-
w	~~w~~ x	~~5~~ 4
y	~~-~~ x	~~-~~ 2
v	v	2
x	x	1



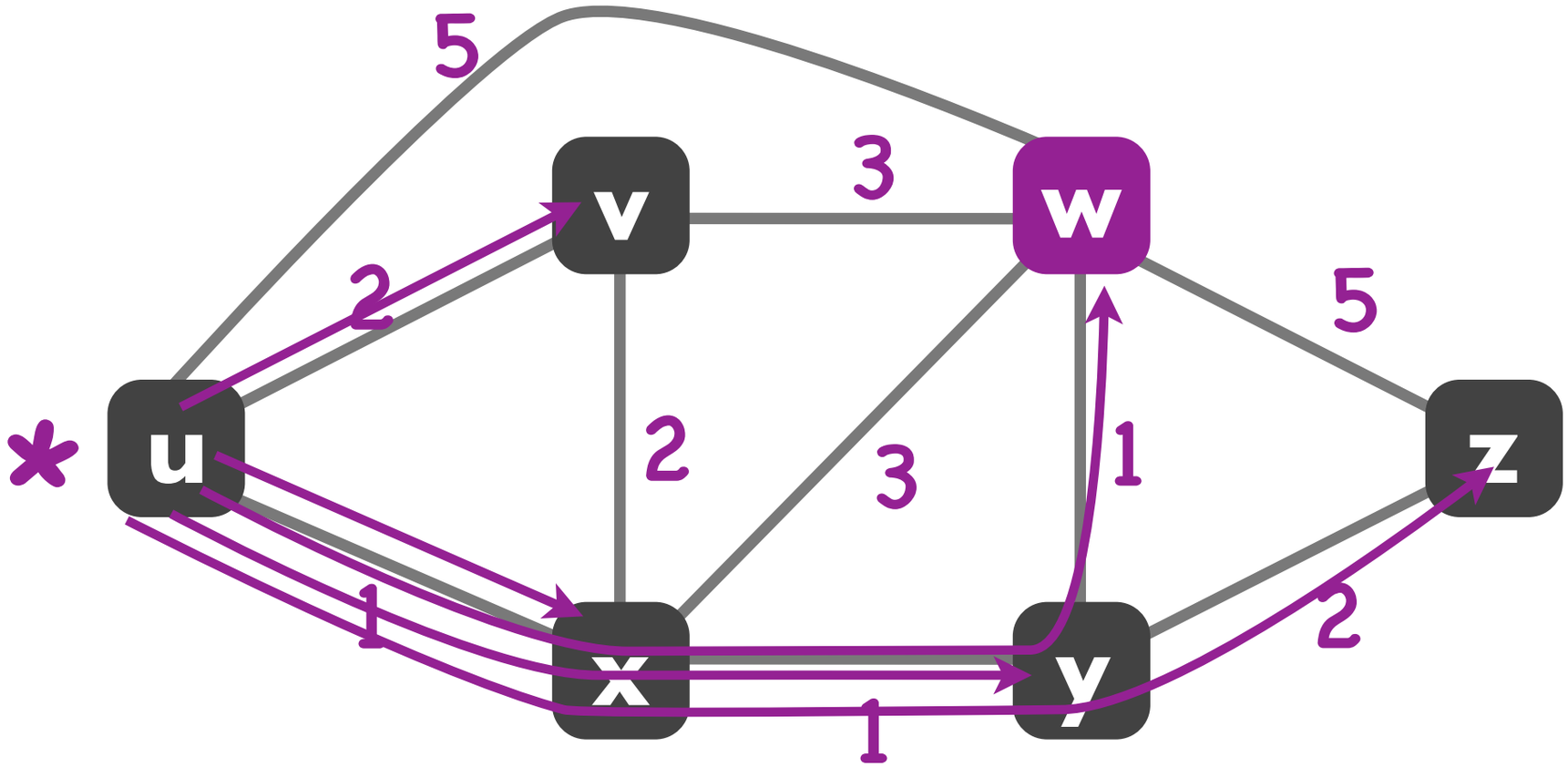
dest.	next hop	cost
z	-	-
w	~~w~~ x	~~5~~ 4
y	~~-~~ x	~~-~~ 2
v	v	2
x	x	1



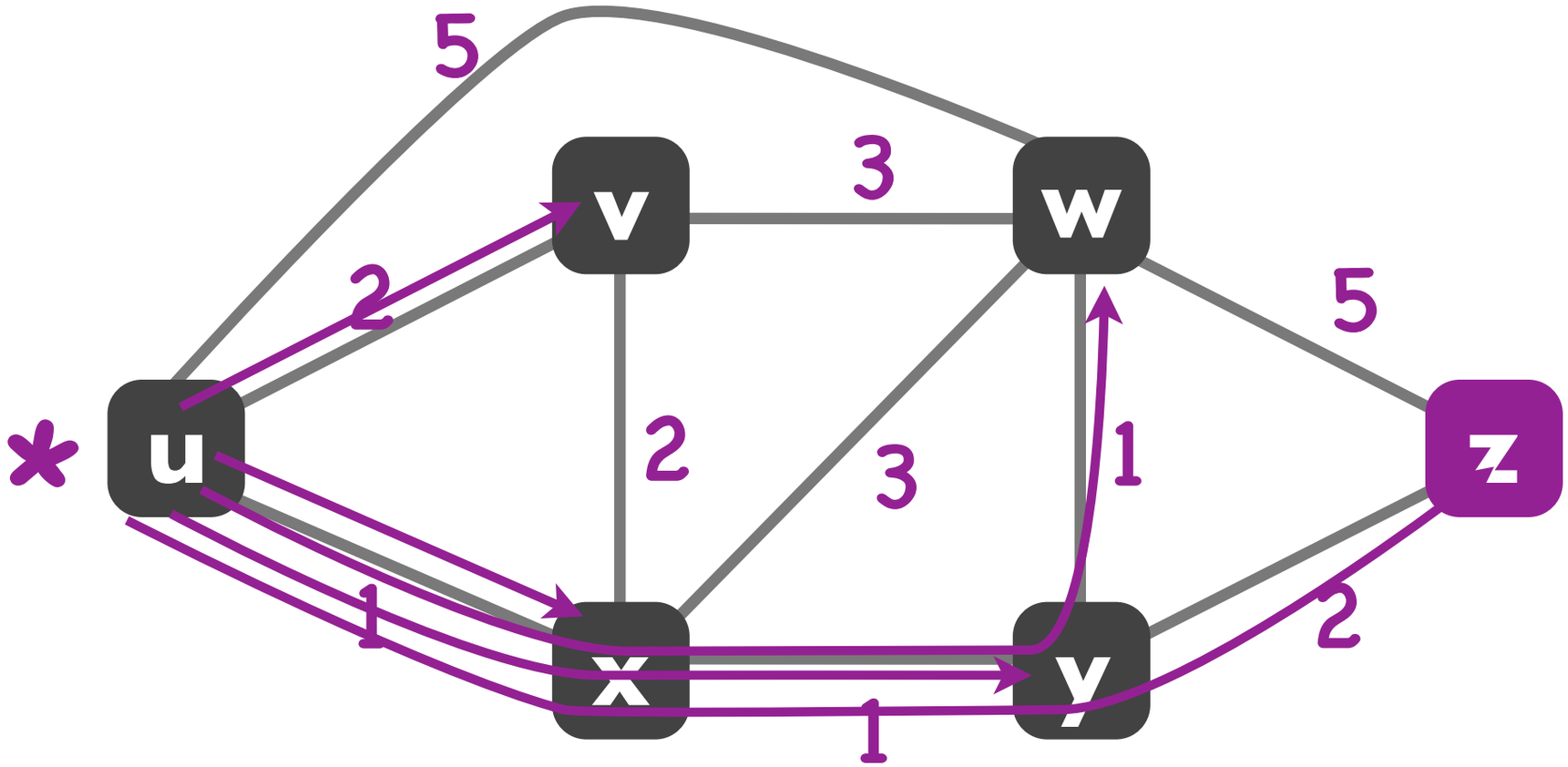
dest.	next hop	cost
z	~~-~~ x	~~-~~ 4
w	~~w~~ x	~~5~~ ~~4~~ 3
y	~~-~~ x	~~-~~ 2
v	v	2
x	x	1



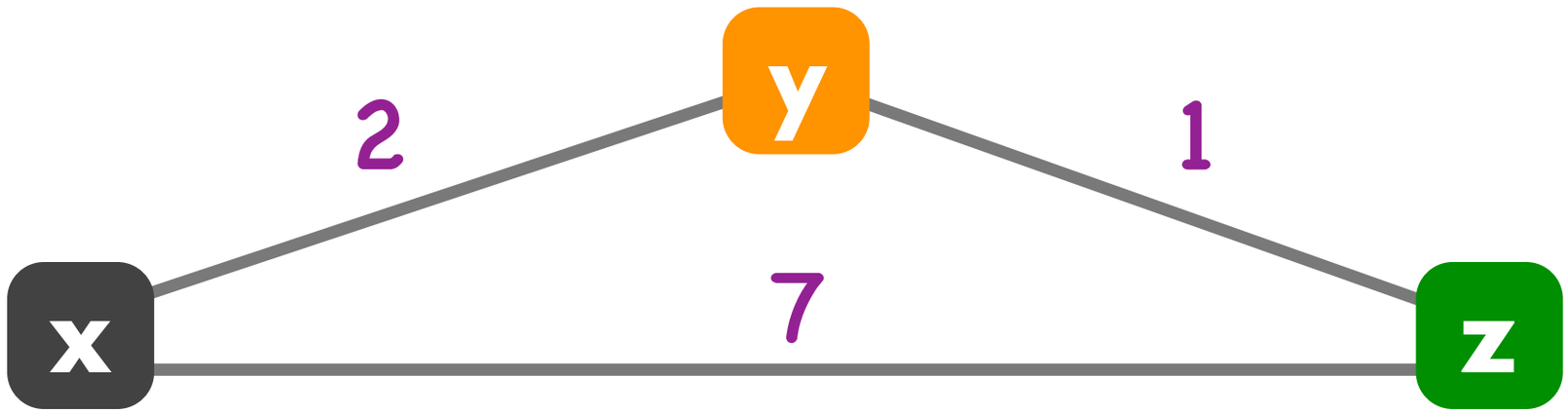
dest.	next hop	cost
z	~~-~~ x	~~-~~ 4
w	~~w~~ x	~~5~~ ~~4~~ 3
y	~~-~~ x	~~-~~ 2
v	v	2
x	x	1



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



	x	y	z
x	–	–	–
y	2	0	1
z	–	–	–

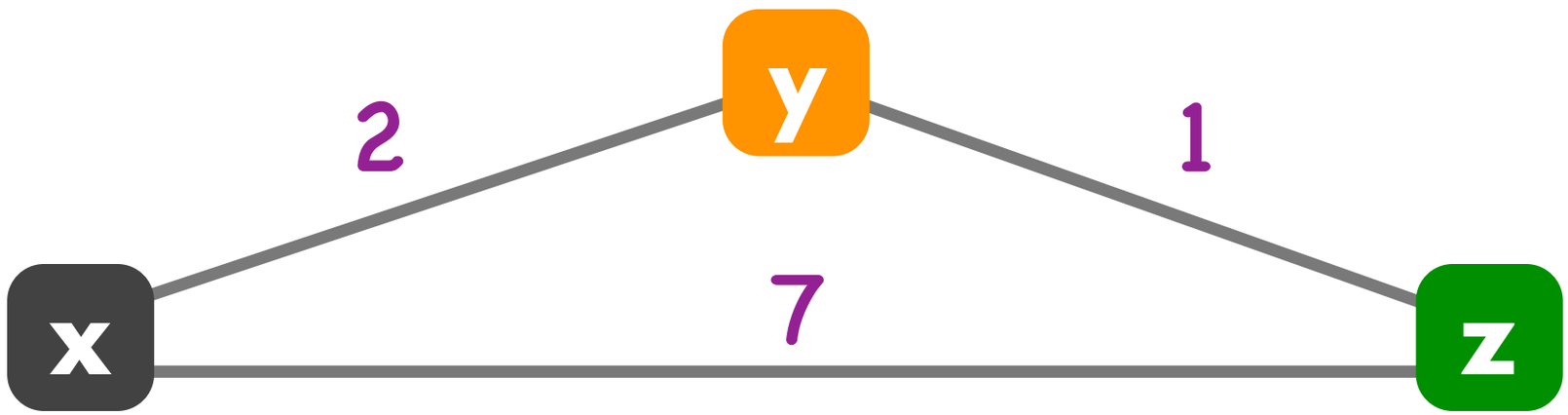


to

from	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



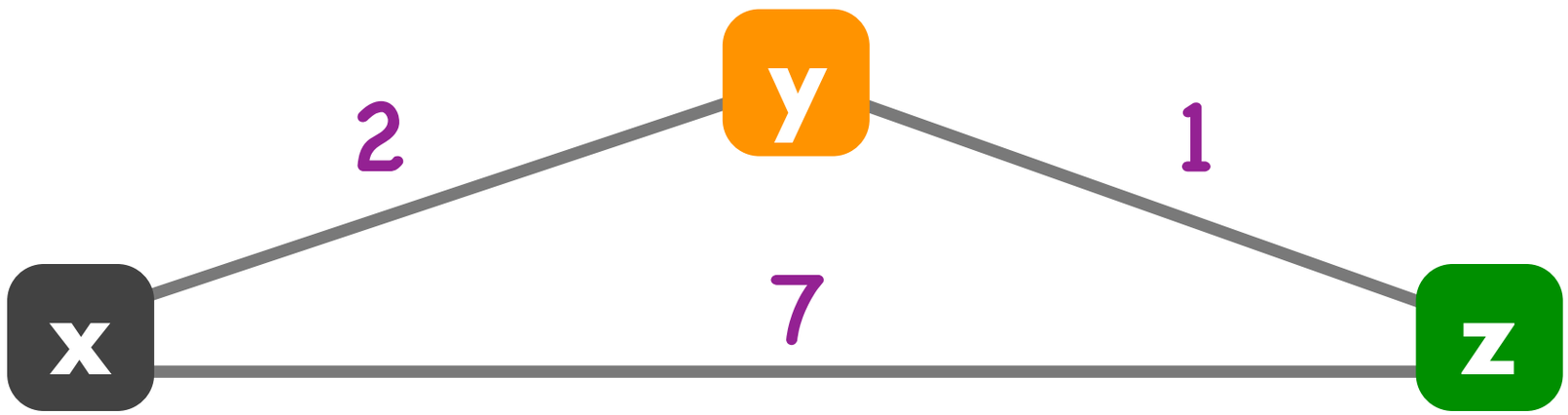
to

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



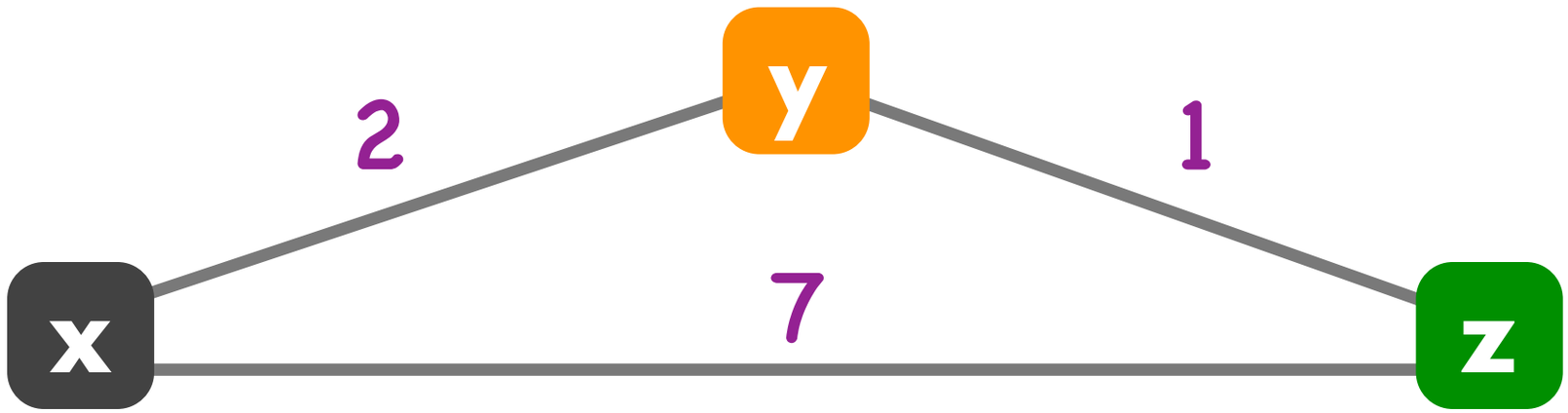
to

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



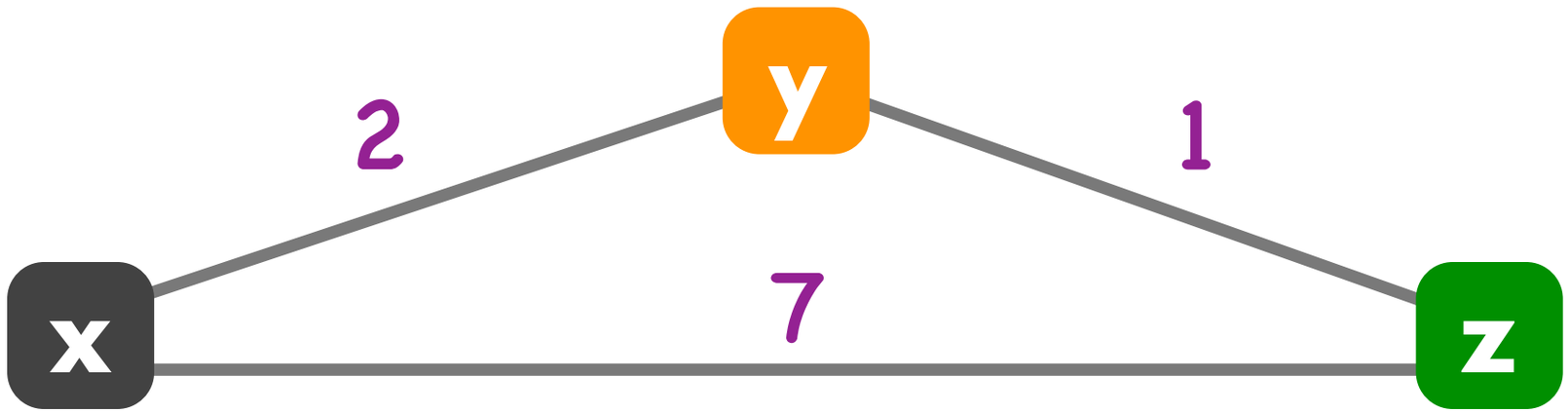
to

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

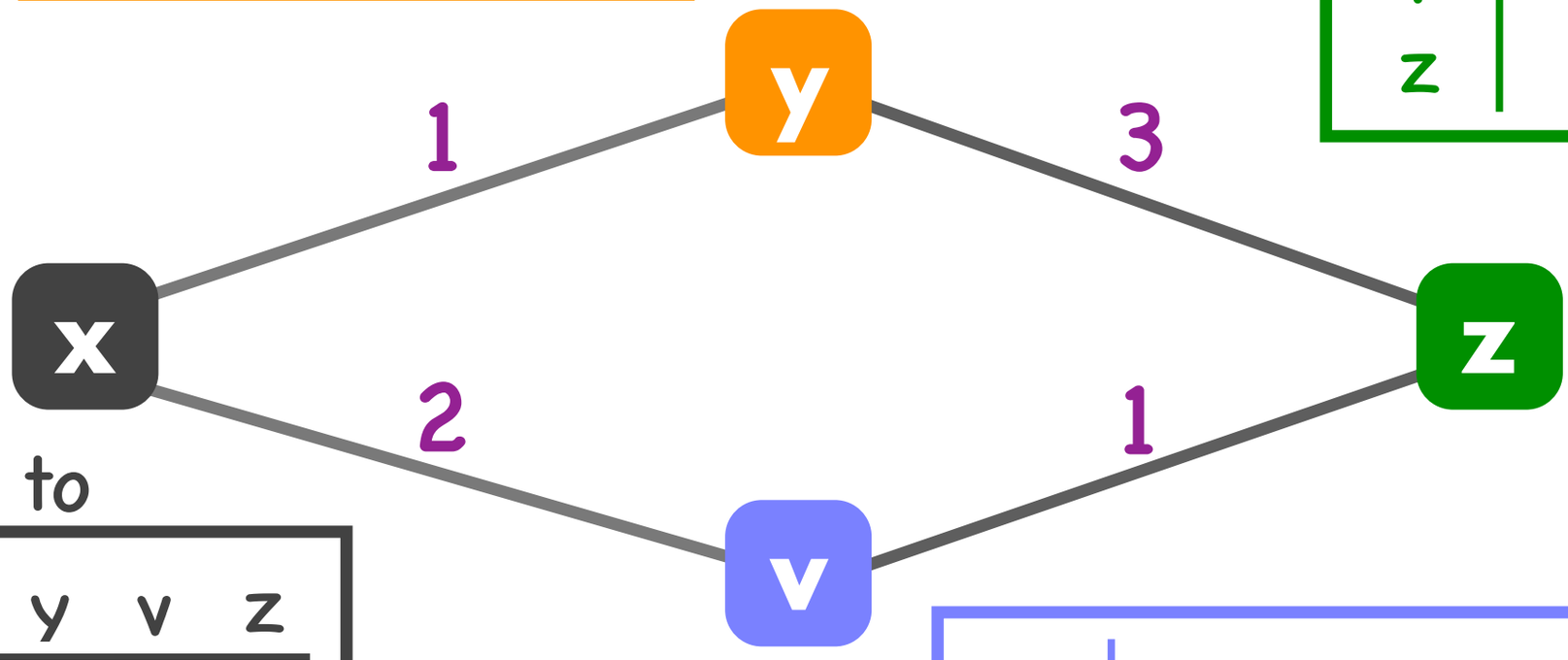
from

	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

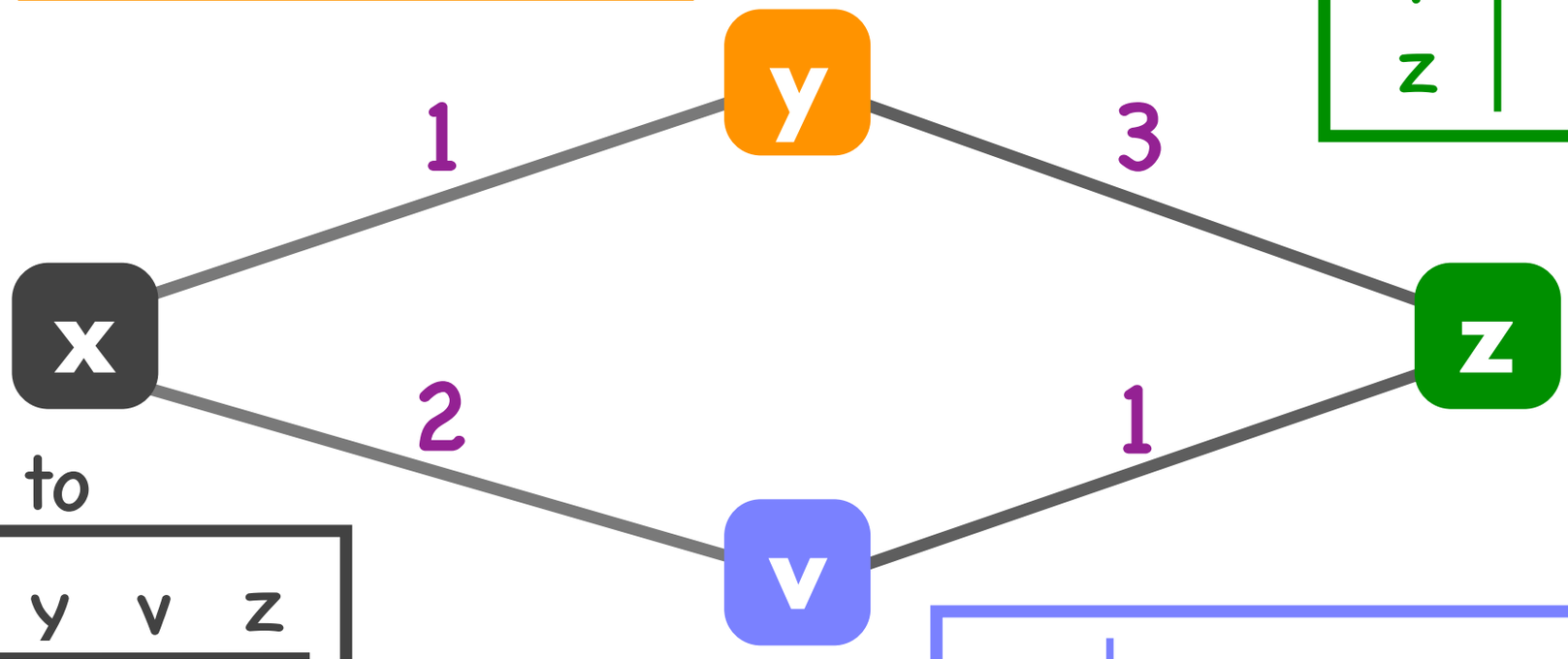


from \ to

	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

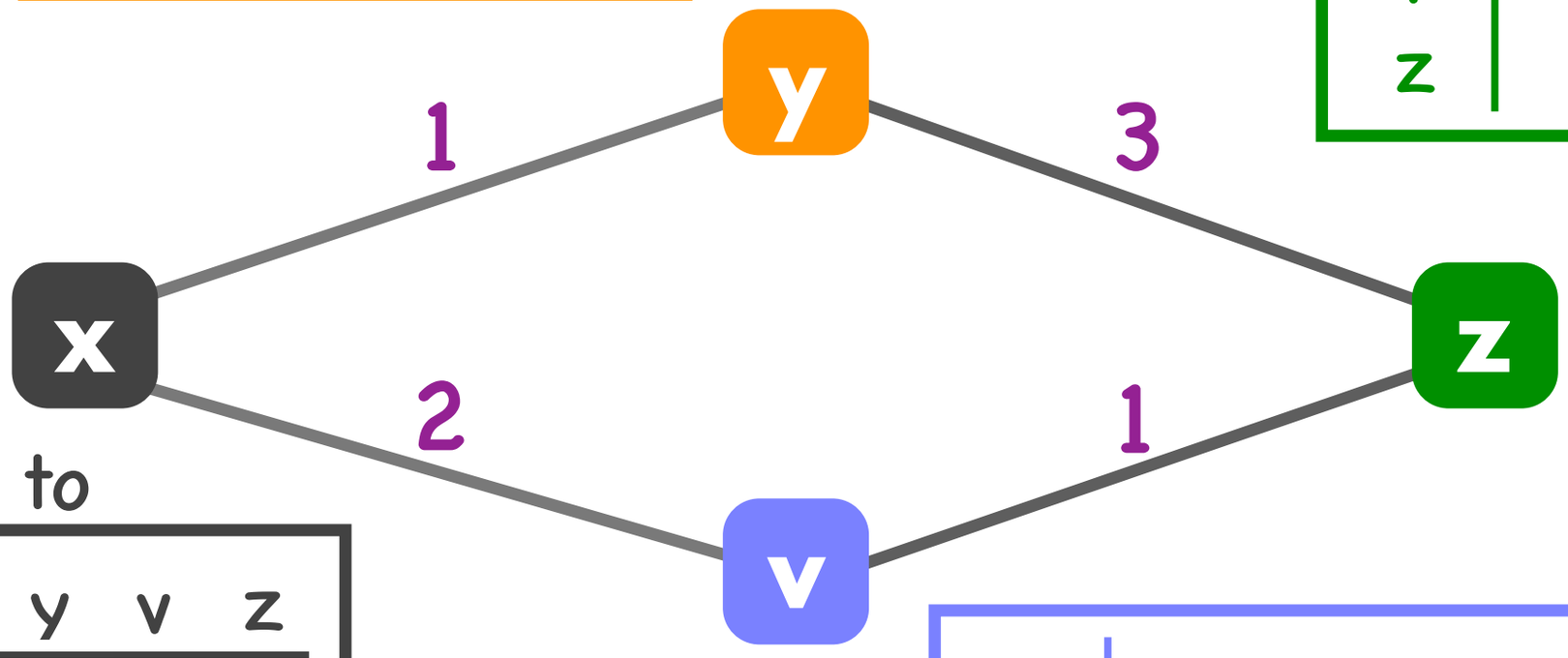
from

	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



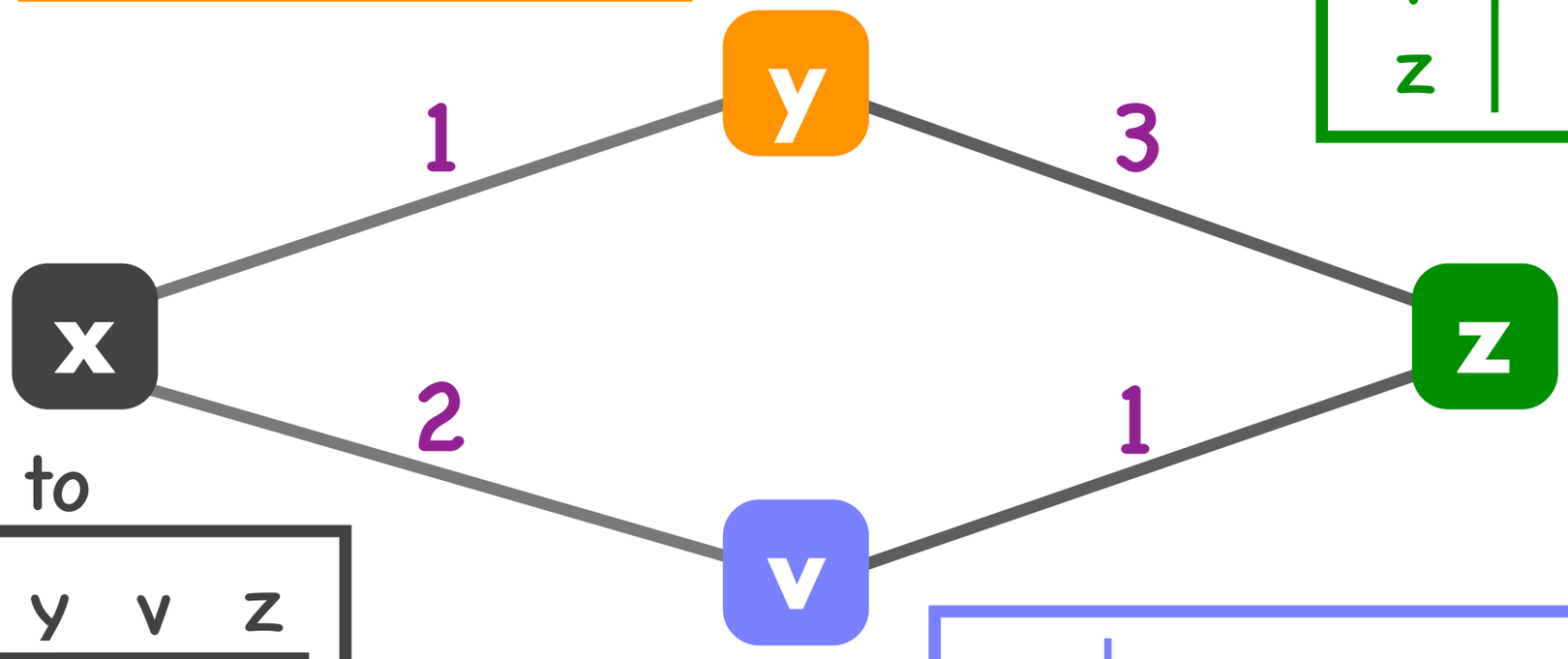
from \ to

	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

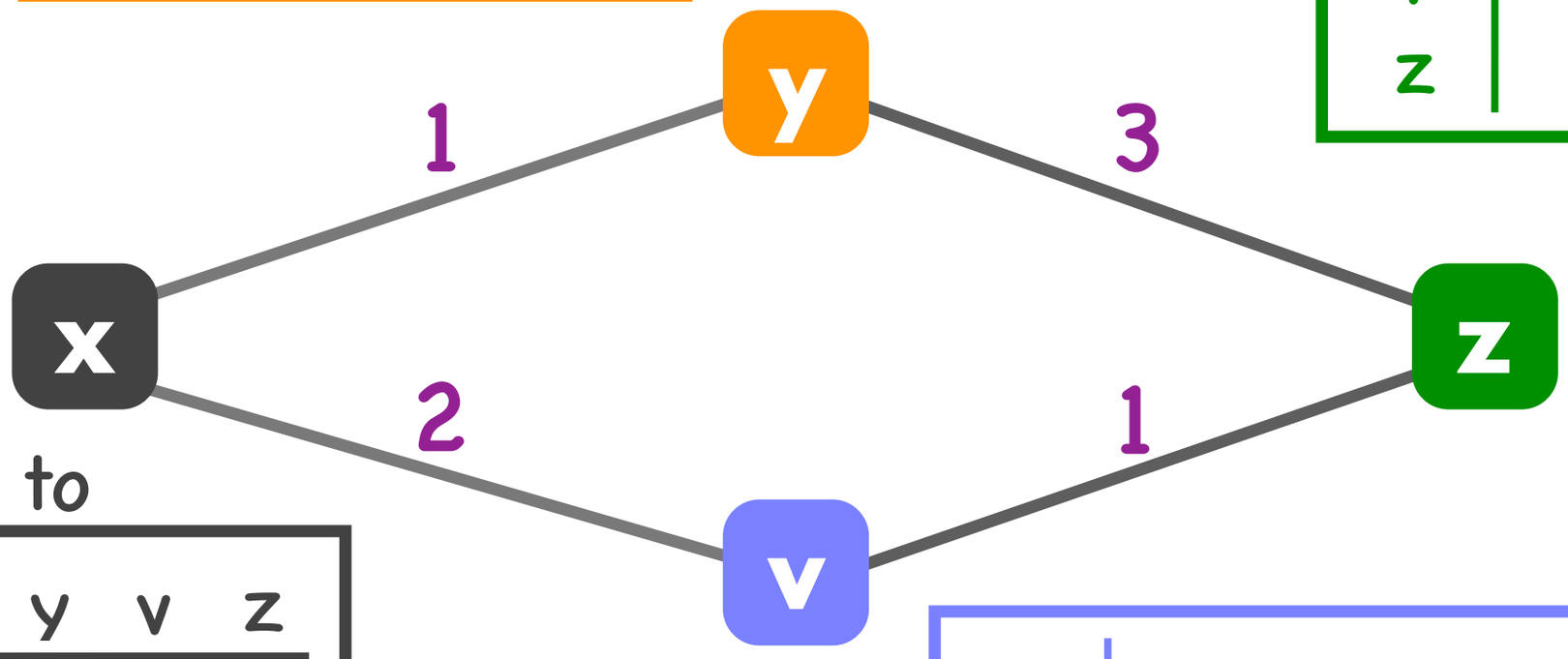
from

	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



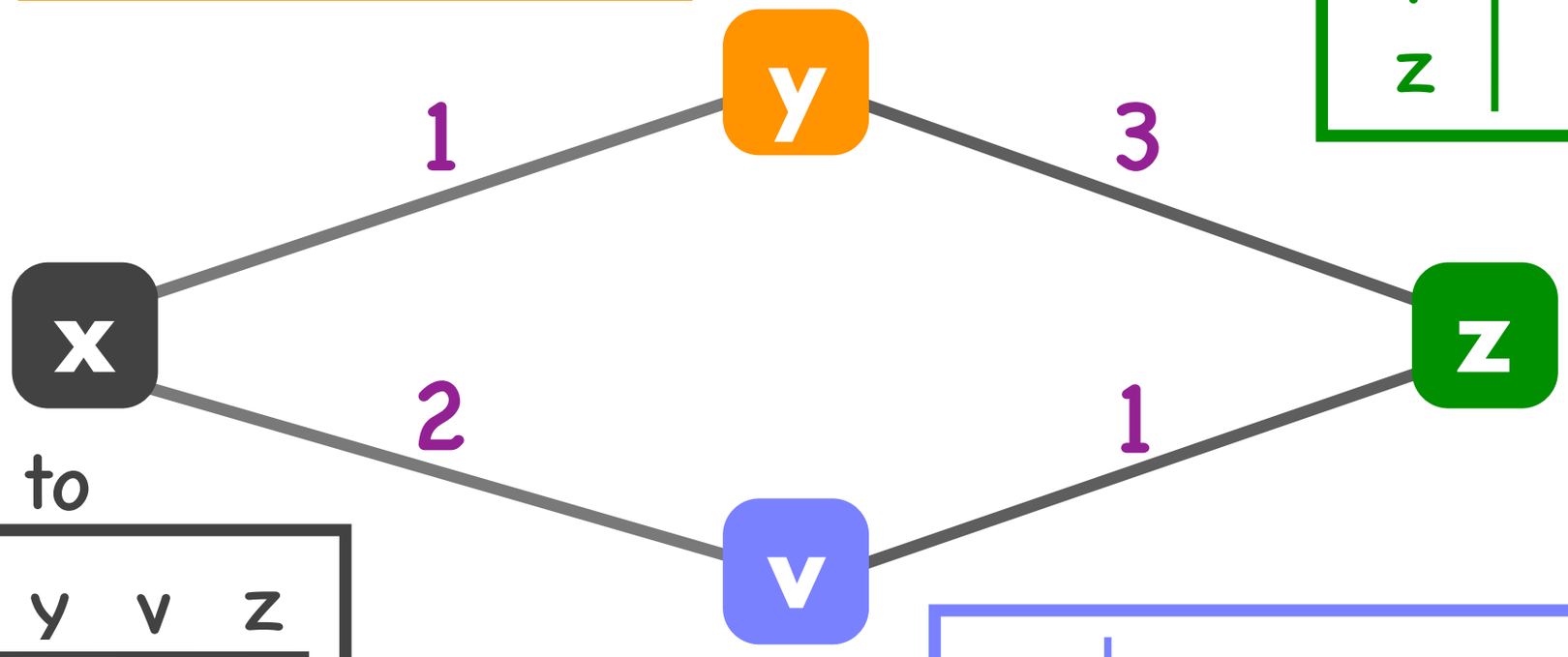
from \ to

	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



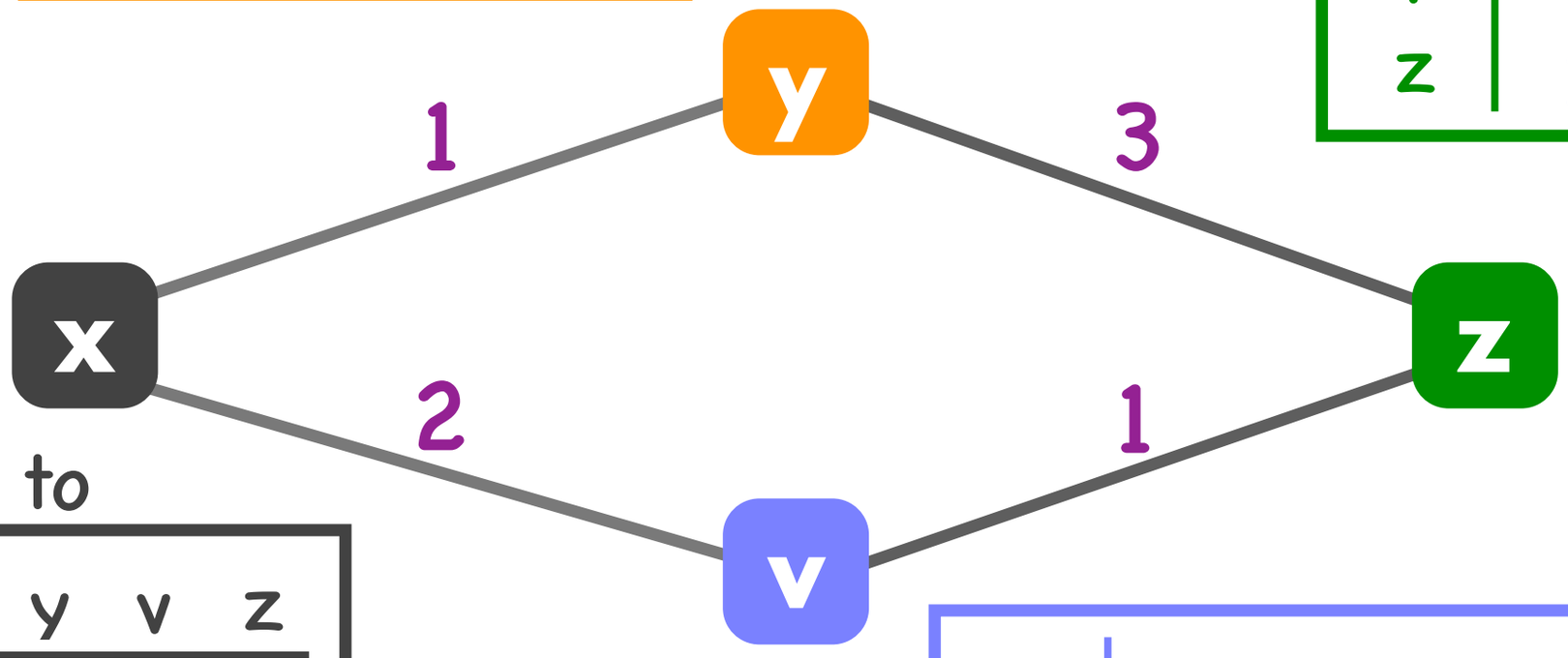
from \ to

	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



from \ to

	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 algorithm

- Input to each router: local link costs & neighbor messages
- Output of each router: least-cost path to every other router

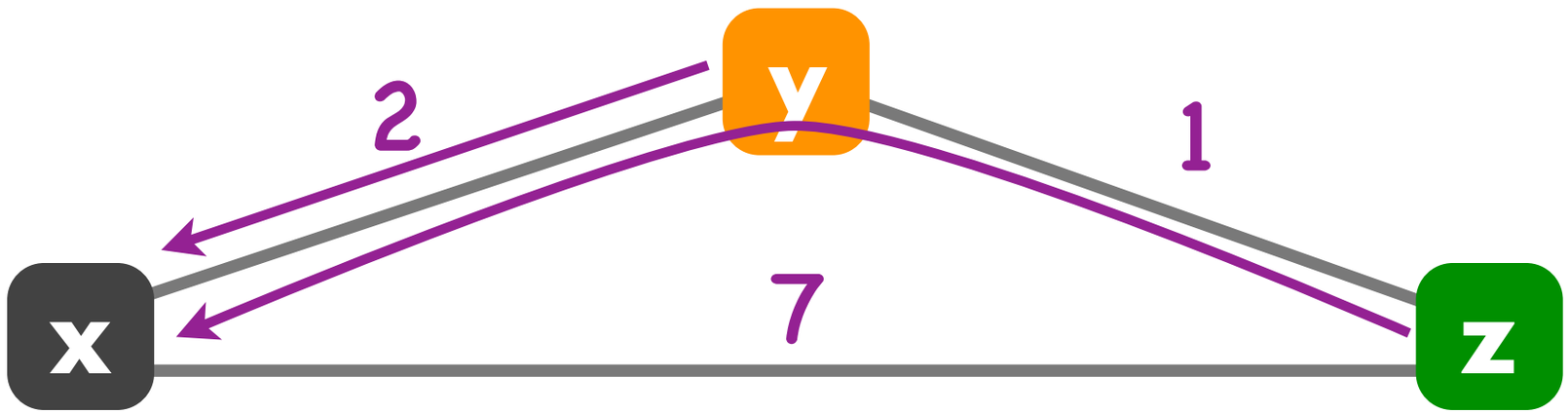
Distance-vector routing algorithm

- "Distributed" algorithm
- All routers run it "together": 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
y	2	0	1
z	3	1	0

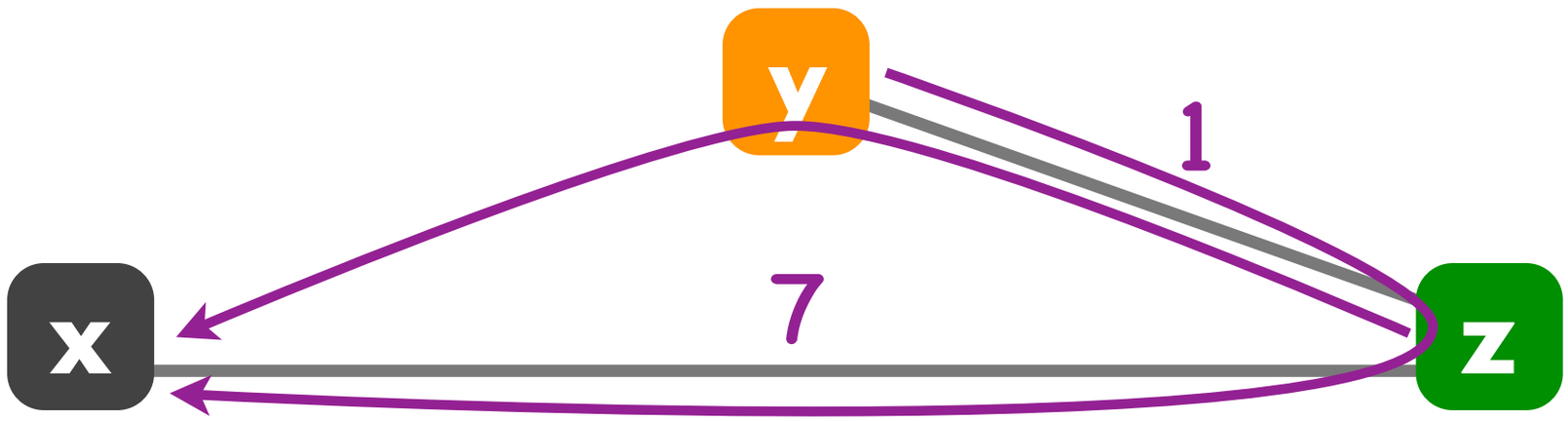


from \ to	x	y	z
y	2	0	1
z	3	1	0

to

from	x	y	z
y	4	0	1
z	3	1	0

routing loop!

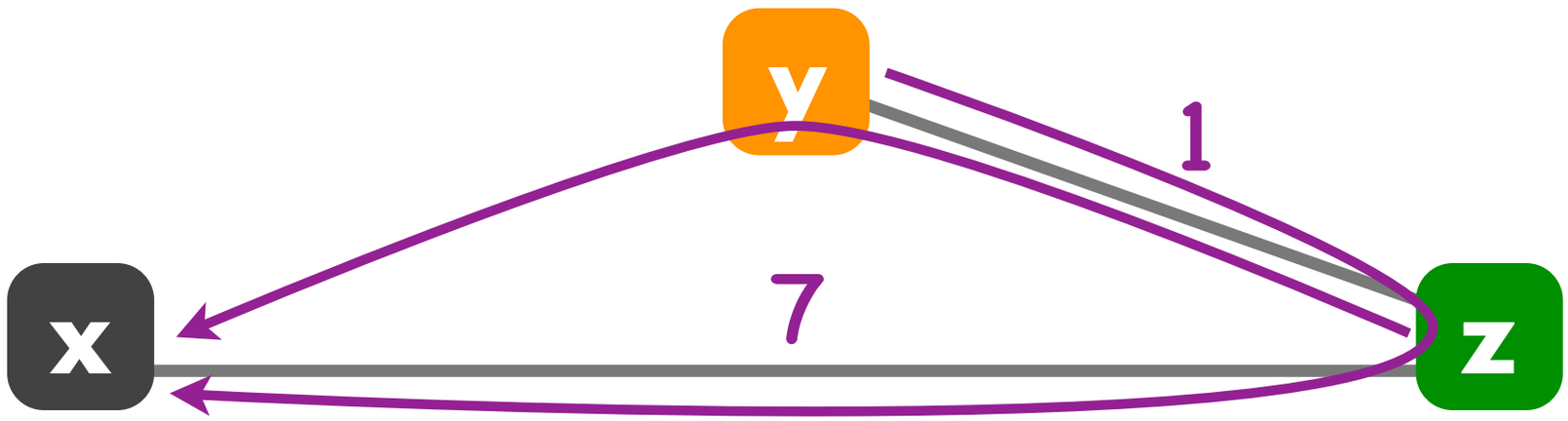


to

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

from \ to	x	y	z
y	4	0	1
z	3	1	0

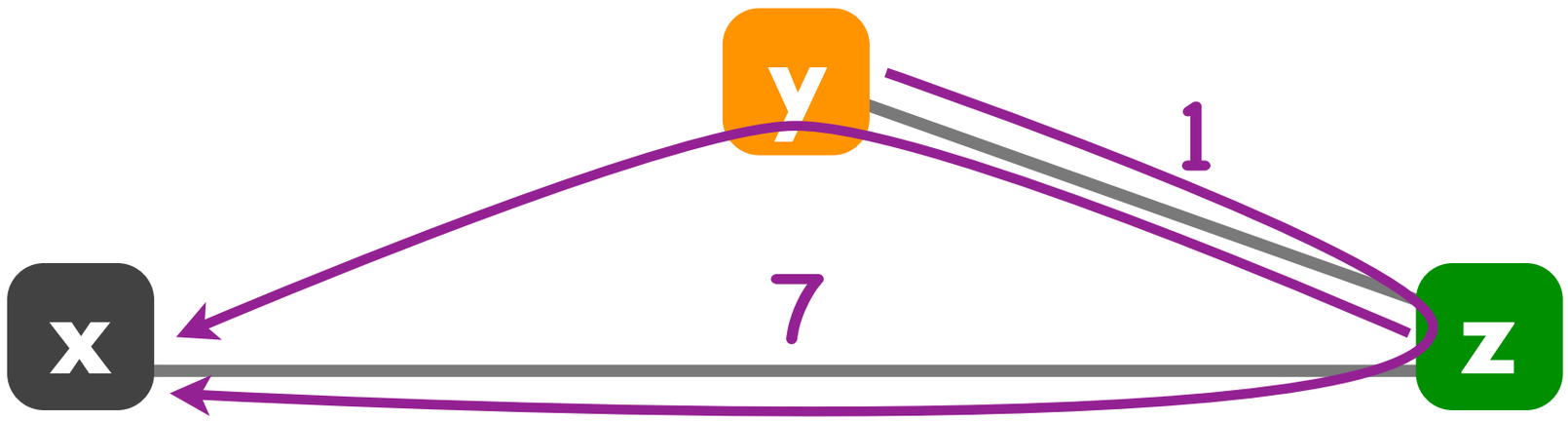
routing loop!



from \ to	x	y	z
y	4	0	1
z	3	1	0

from \ to	x	y	z
y	4	0	1
z	3	1	0

routing loop!

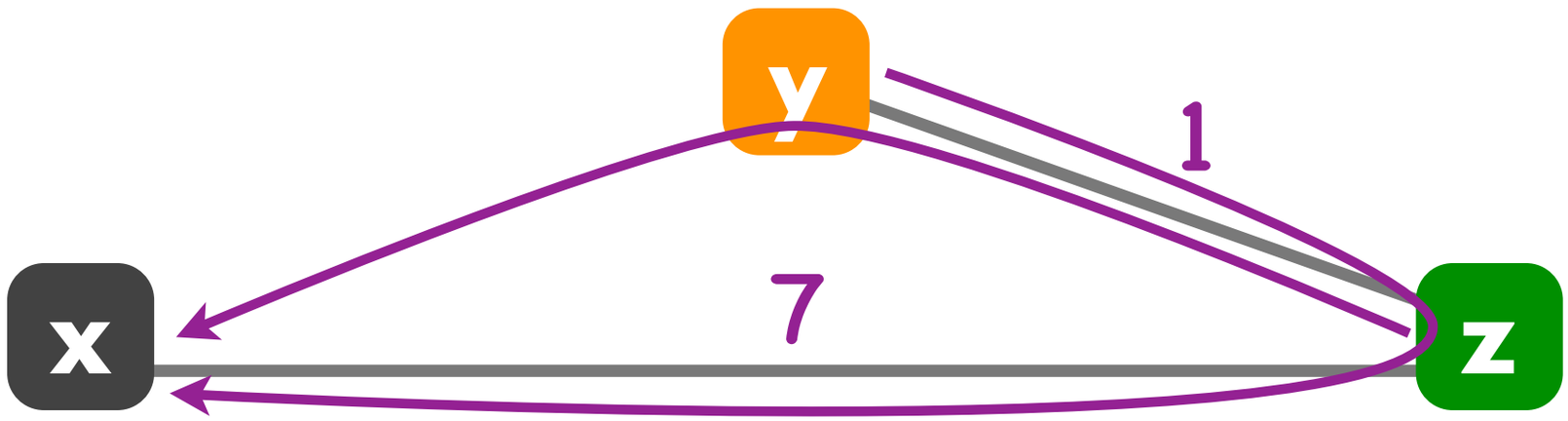


from \ to	x	y	z
y	4	0	1
z	5	1	0

to

from	x	y	z
y	4	0	1
z	5	1	0

routing loop!

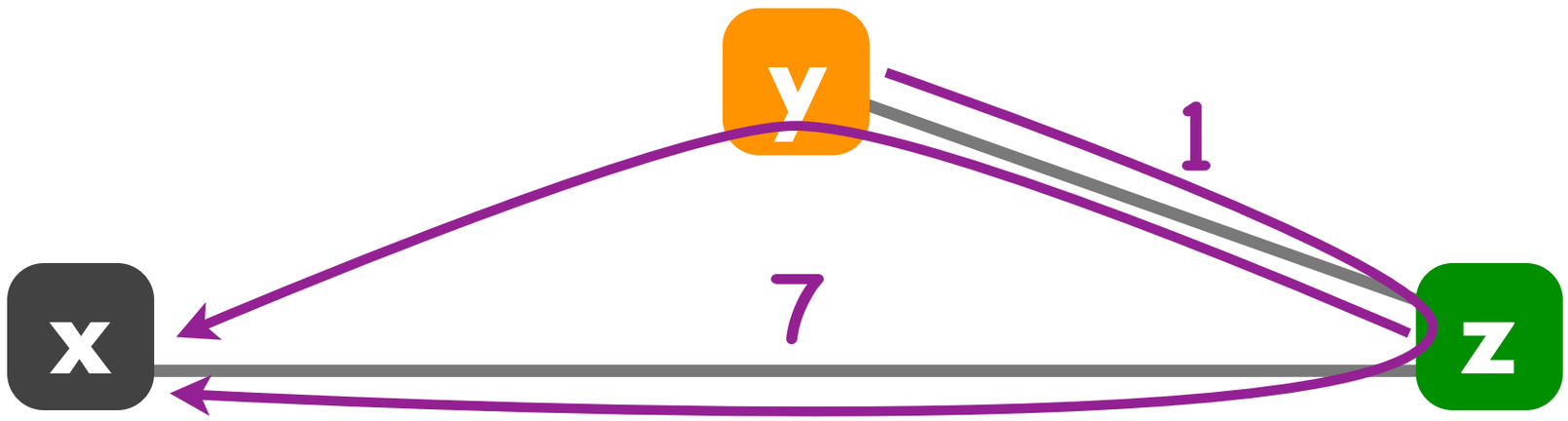


to

from	x	y	z
y	4	0	1
z	5	1	0

from \ to	x	y	z
y	6	0	1
z	5	1	0

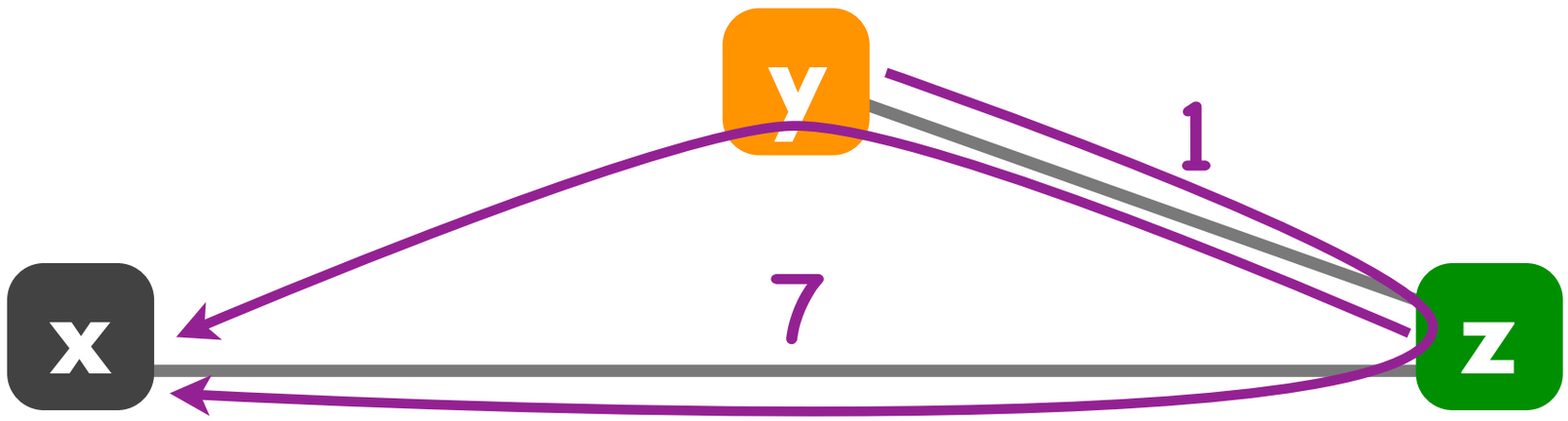
routing loop!



from \ to	x	y	z
y	4	0	1
z	5	1	0

from \ to	x	y	z
y	6	0	1
z	5	1	0

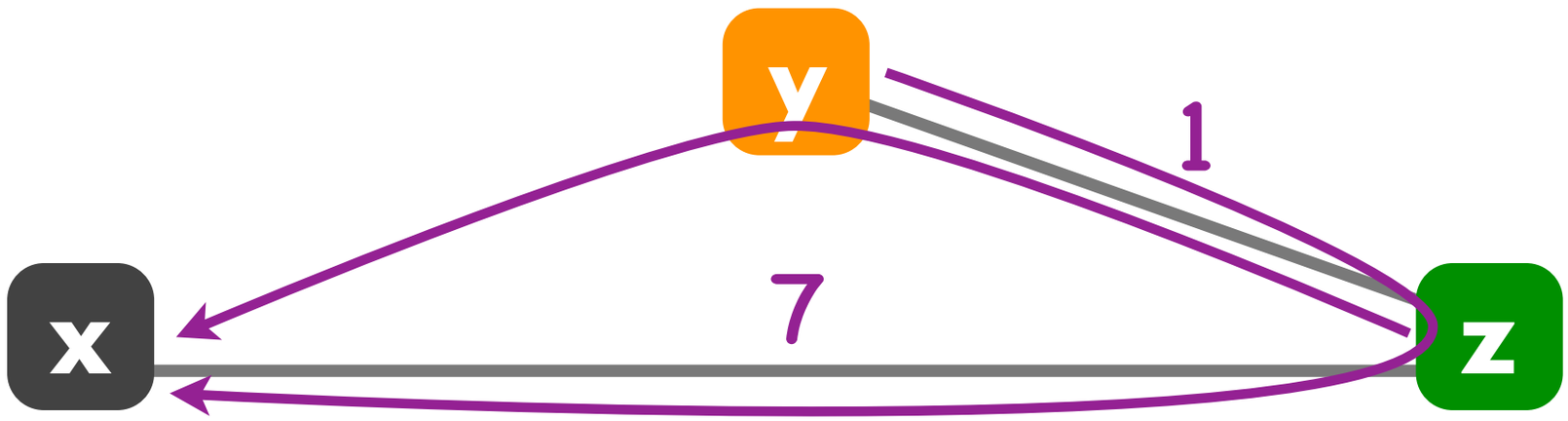
routing loop!



from \ to	x	y	z
y	6	0	1
z	5	1	0

from \ to	x	y	z
y	6	0	1
z	5	1	0

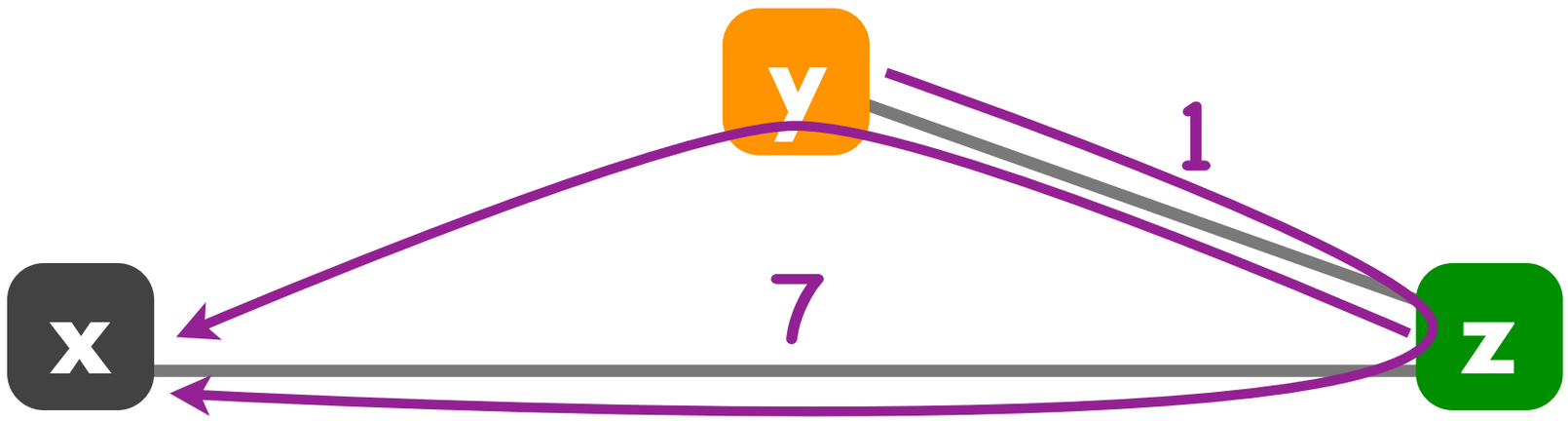
routing loop!



from \ to	x	y	z
y	6	0	1
z	7	1	0

from \ to	x	y	z
y	6	0	1
z	7	1	0

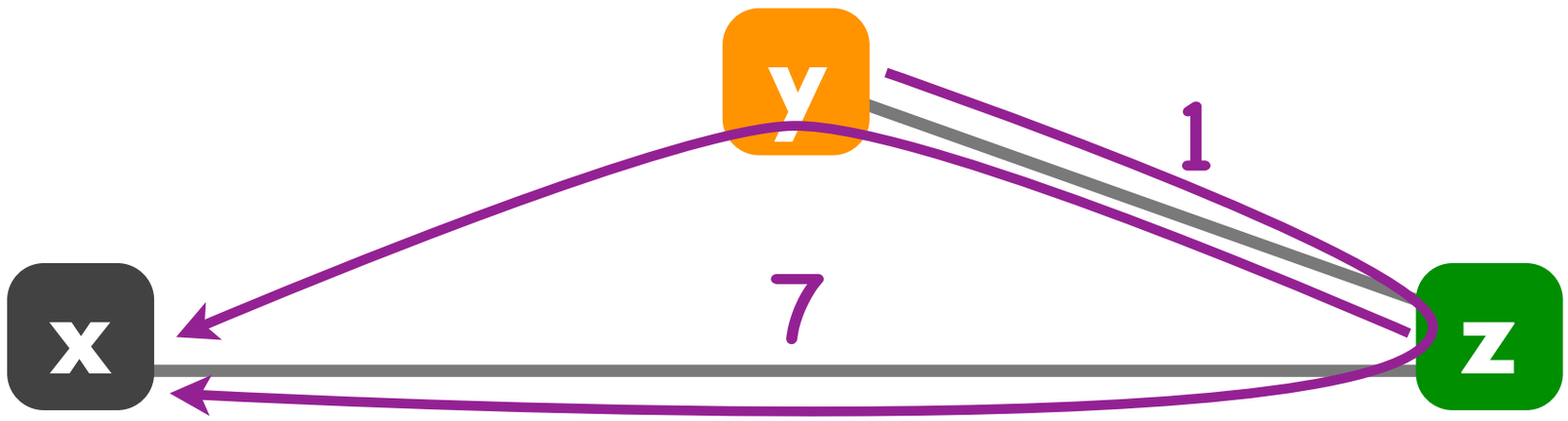
routing loop!



from \ to	x	y	z
y	6	0	1
z	7	1	0

from \ to	x	y	z
y	8	0	1
z	7	1	0

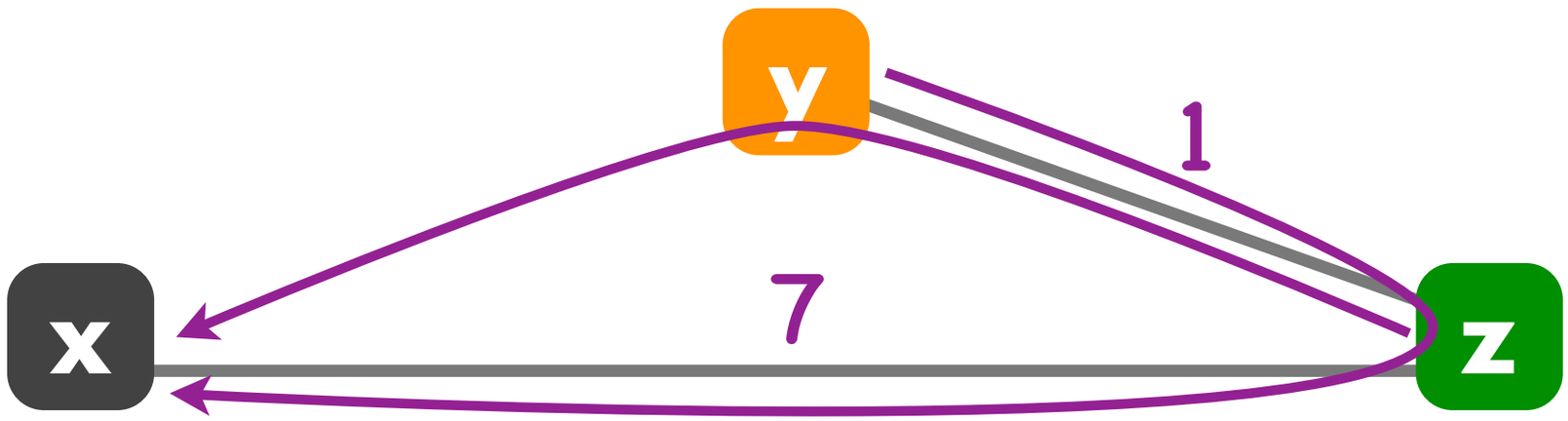
routing loop!



from \ to	x	y	z
y	6	0	1
z	7	1	0

from \ to	x	y	z
y	8	0	1
z	7	1	0

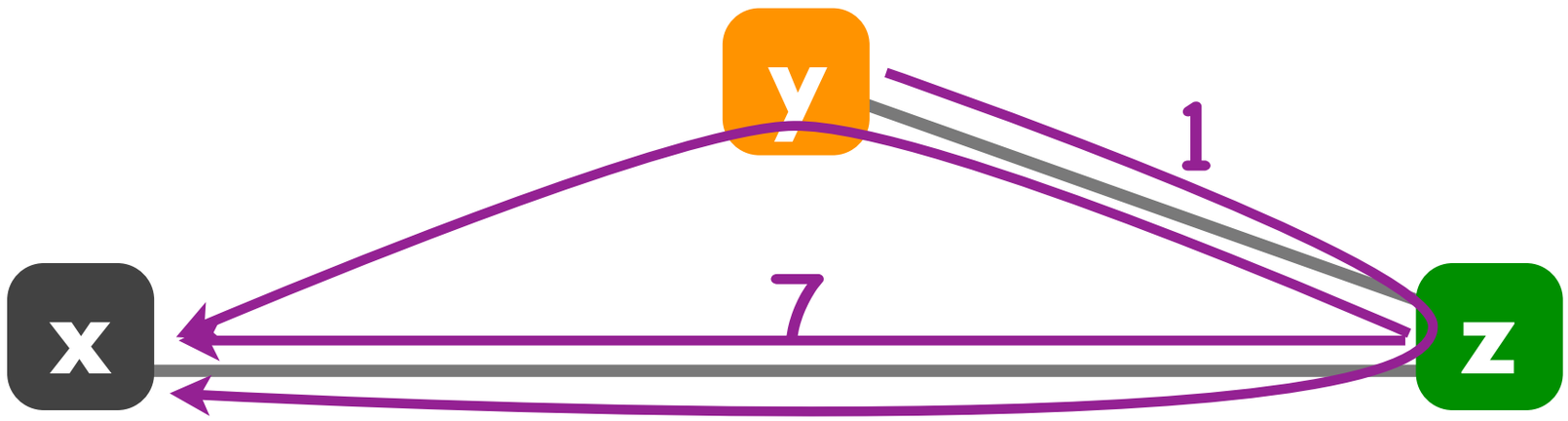
routing loop!



from \ to	x	y	z
y	8	0	1
z	7	1	0

from \ to	x	y	z
y	8	0	1
z	7	1	0

routing loop!



count-to-infinity scenario

from \ to	x	y	z
y	8	0	1
z	7	1	0

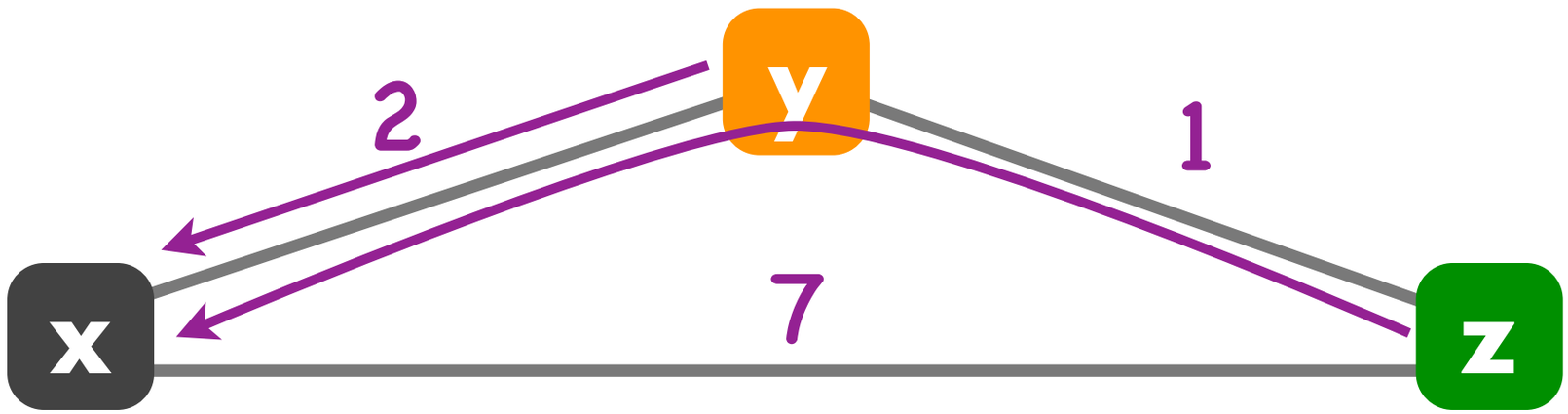
Problem with Bellman-Ford

- Routing loop
  - z routes to x through y
  - y loses connectivity to x
  - y decides to route to x through z

- Can take very long to resolve
  - count-to-infinity scenario

poisoned reverse

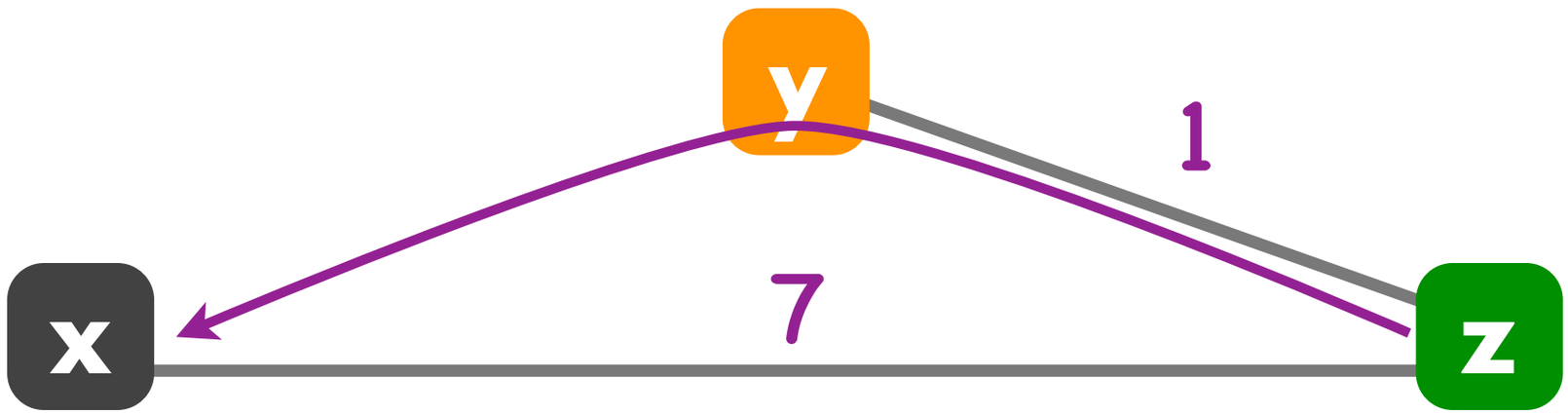
from \ to	x	y	z
y	2	0	1
z	∞	1	0



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

poisoned reverse

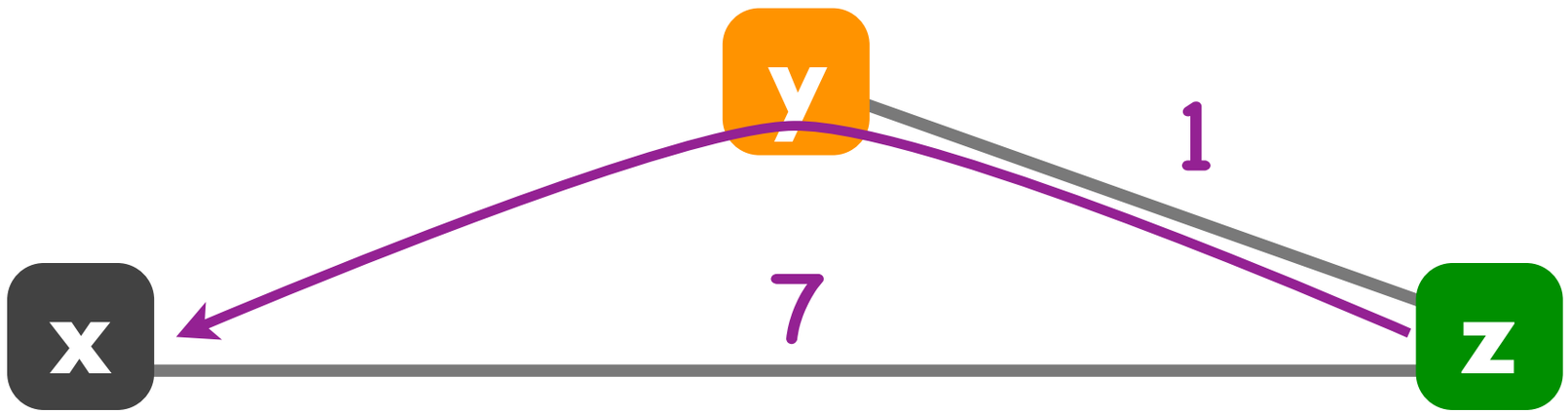
from \ to	x	y	z
y	∞	0	1
z	∞	1	0



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

poisoned reverse

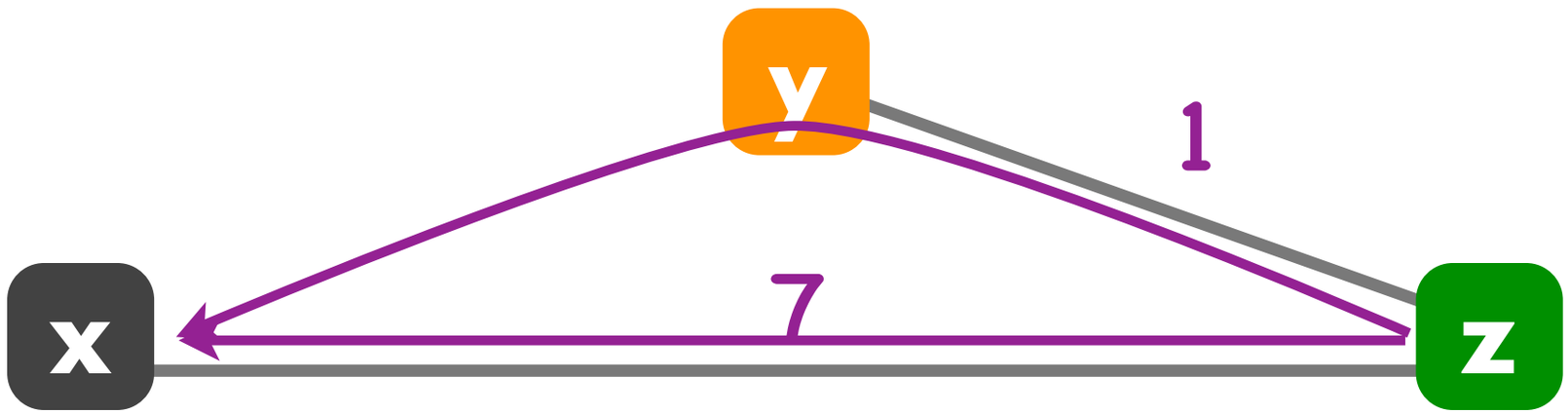
from \ to	x	y	z
y	∞	0	1
z	∞	1	0



from \ to	x	y	z
y	∞	0	1
z	3	1	0

poisoned reverse

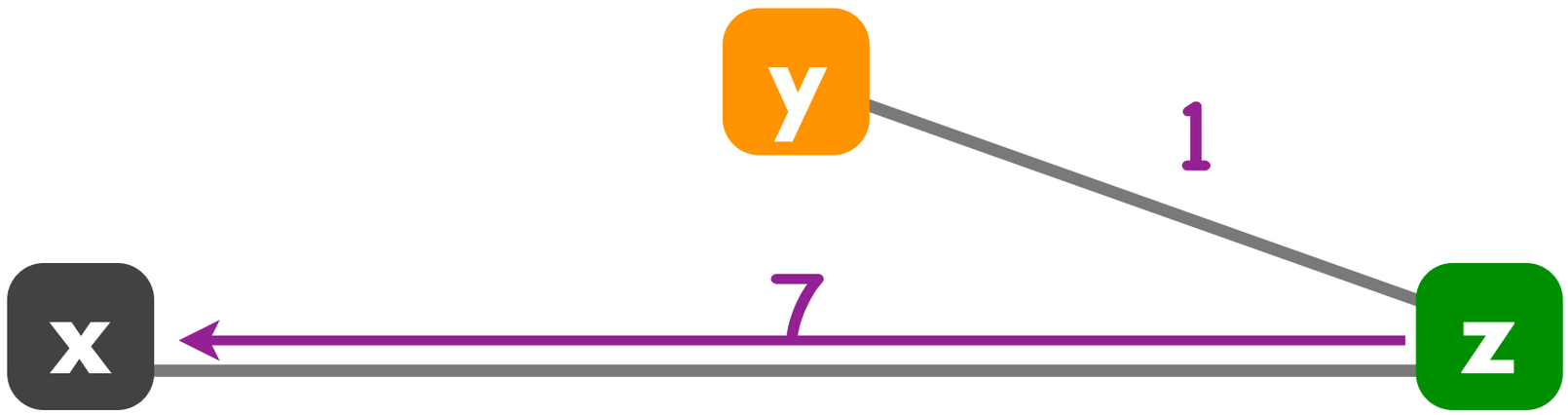
from \ to	x	y	z
y	∞	0	1
z	∞	1	0



from \ to	x	y	z
y	∞	0	1
z	7	1	0

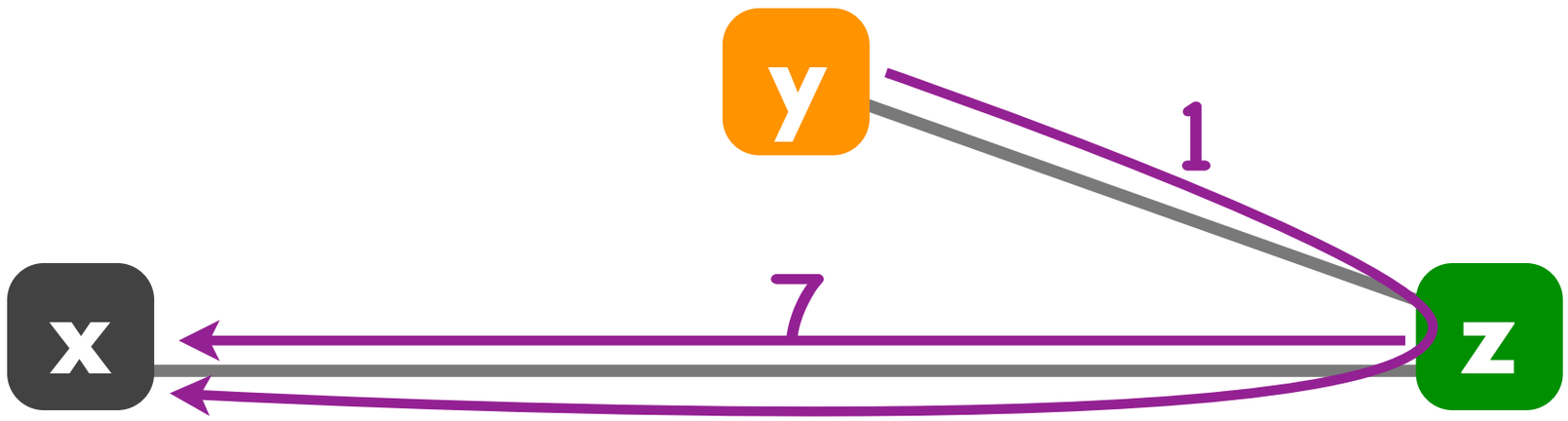
poisoned reverse

from \ to	x	y	z
y	∞	0	1
z	7	1	0



from \ to	x	y	z
y	∞	0	1
z	7	1	0

from \ to	x	y	z
y	8	0	1
z	7	1	0



poisoned reverse

from \ to	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 infinite
    - y never decides to route to x through z

- Algorithm re-converges quickly
  - avoids count-to-infinity scenario

Link-state + distance-vector

- They solve the same problem:
  compute the least-cost path
  from each source router
  to each destination router

Link-state vs. distance-vector

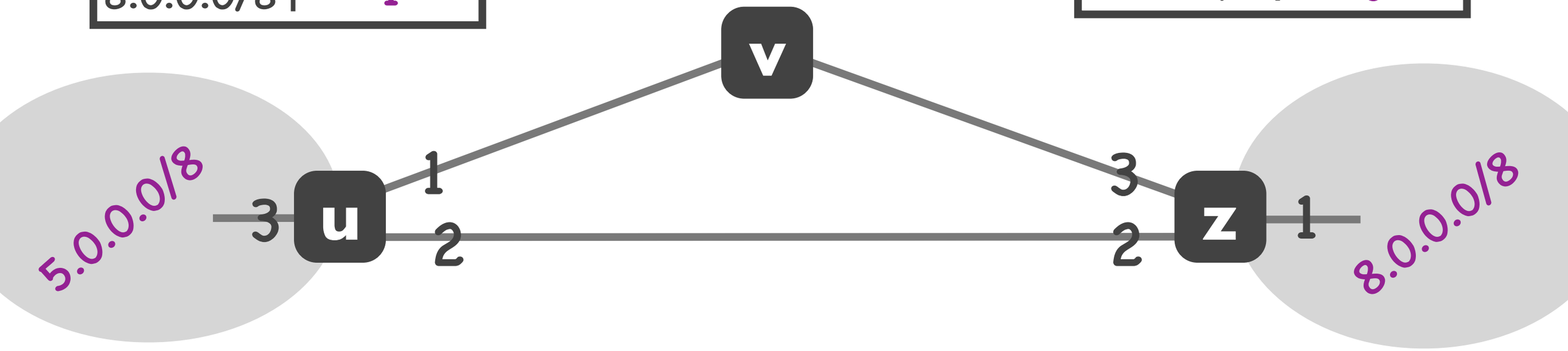
- Link state: each entity first obtains complete view of the network, then computes the least-cost paths
- Distance vector: each entity obtains incrementally new information about the network at every round

Link-state vs. distance-vector

- Link-state converges faster
  - each router starts with full picture of the network
- Distance-vector uses less bandwidth
  - each router only talks to its neighbors

dest.	out. link
5.0.0.0/8	3
8.0.0.0/8	1

dest.	out. link
8.0.0.0/8	1
5.0.0.0/8	3



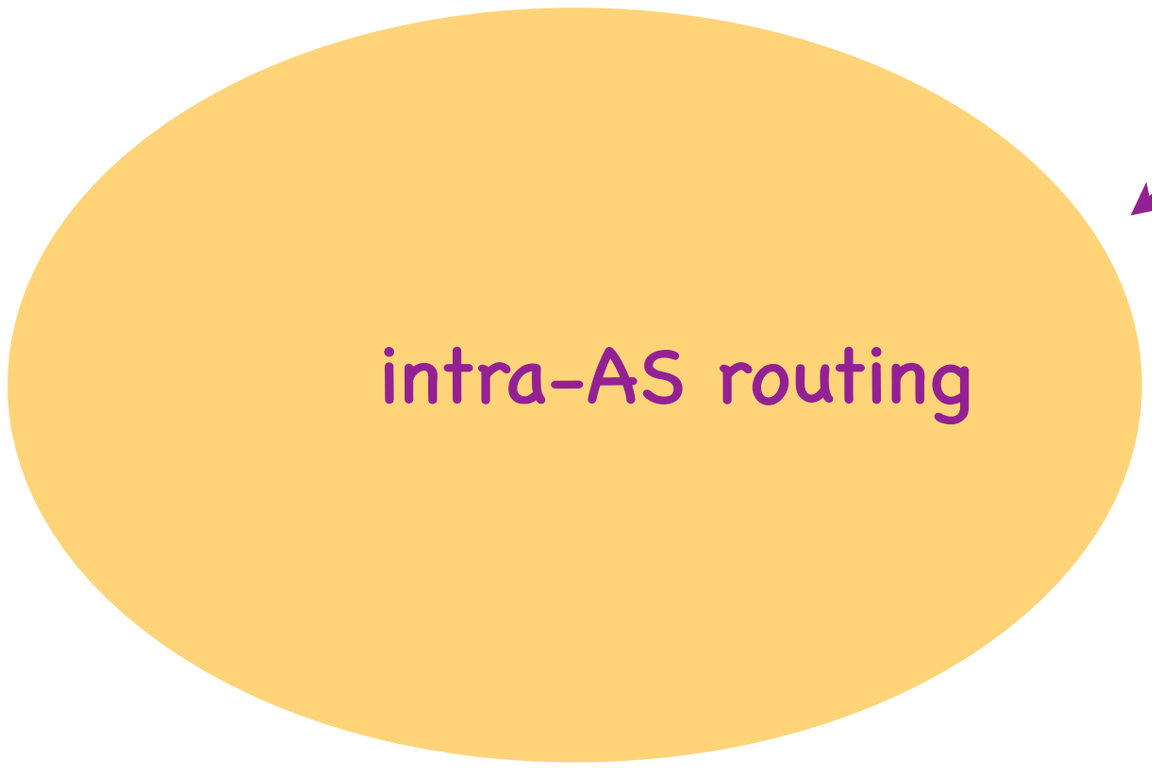
dest.	IP prefixes	next hop
z	8.0.0.0/8	v

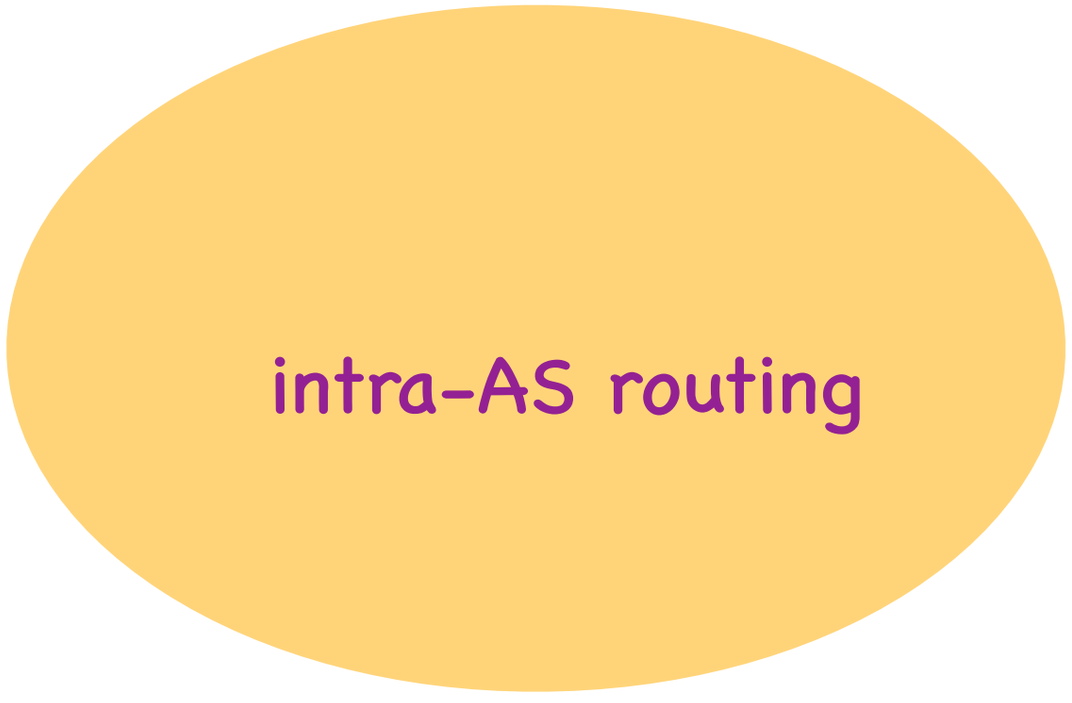
dest.	IP prefixes	next hop
u	5.0.0.0/8	v

Internet routing challenges

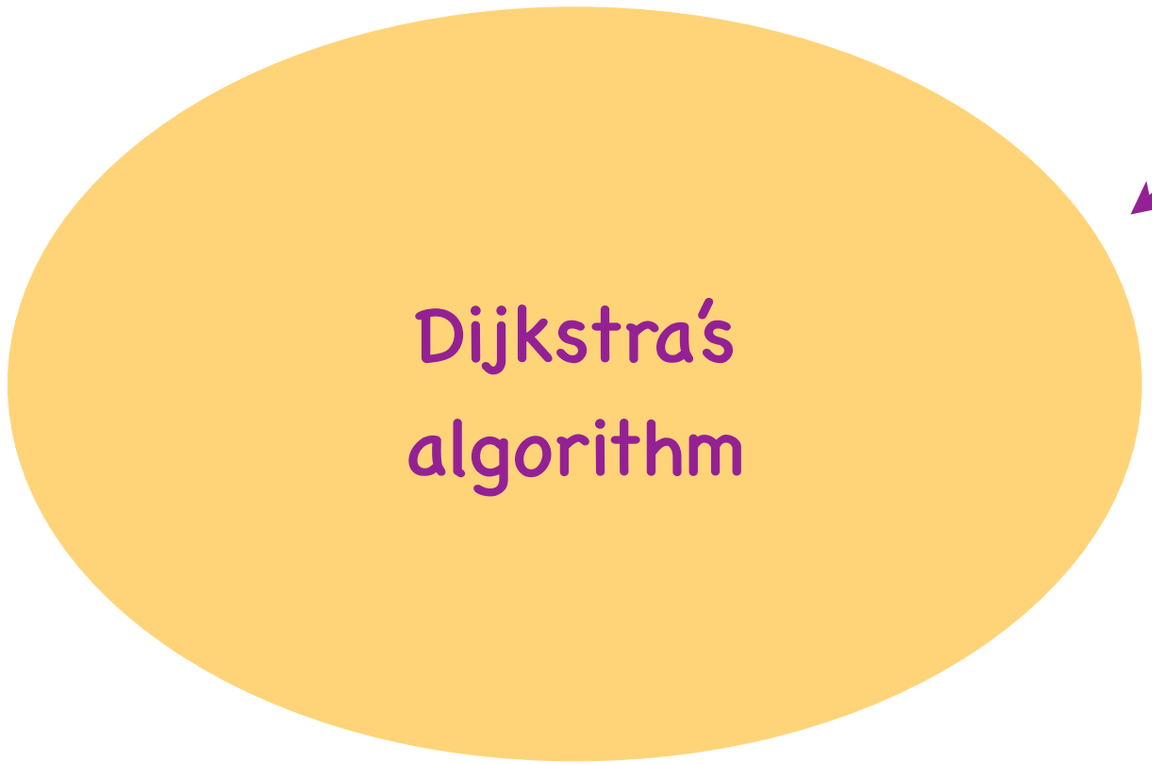
- Scale
  - link-state would cause flooding
  - distance-vector would not converge
- Administrative autonomy
  - an ISP may not want to do least-cost routing
  - may want to hide its link costs from the world

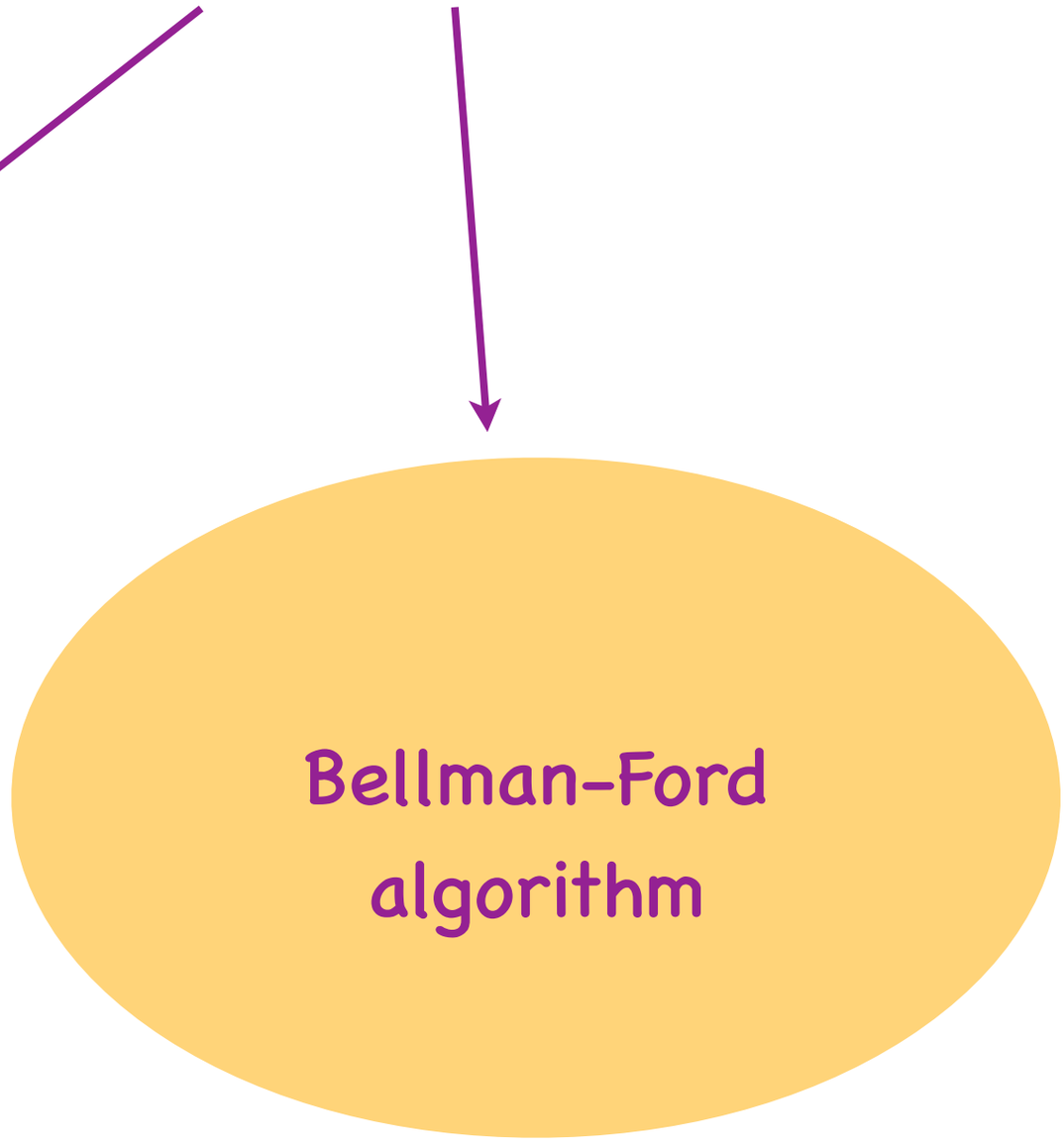
Autonomous Systems

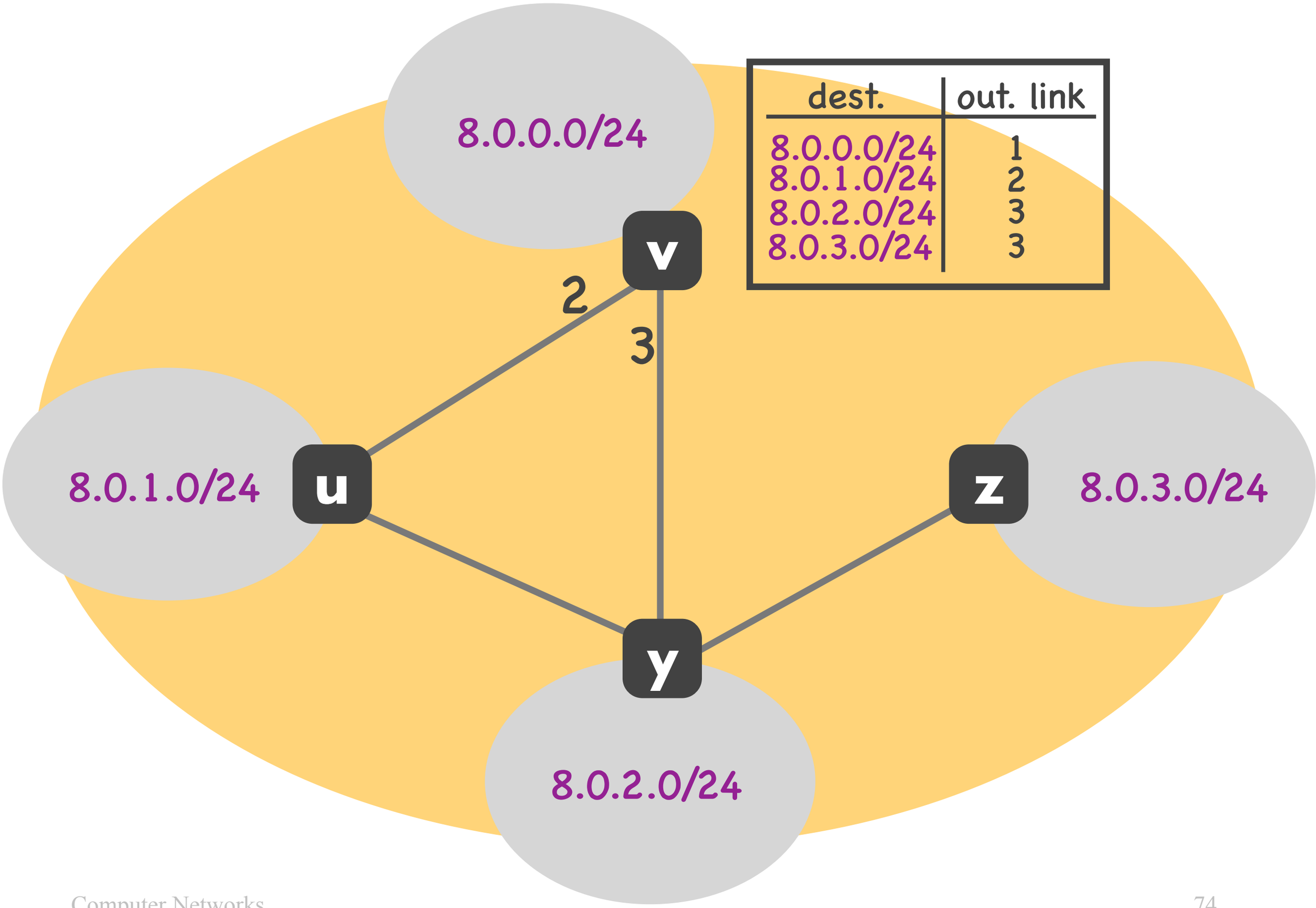




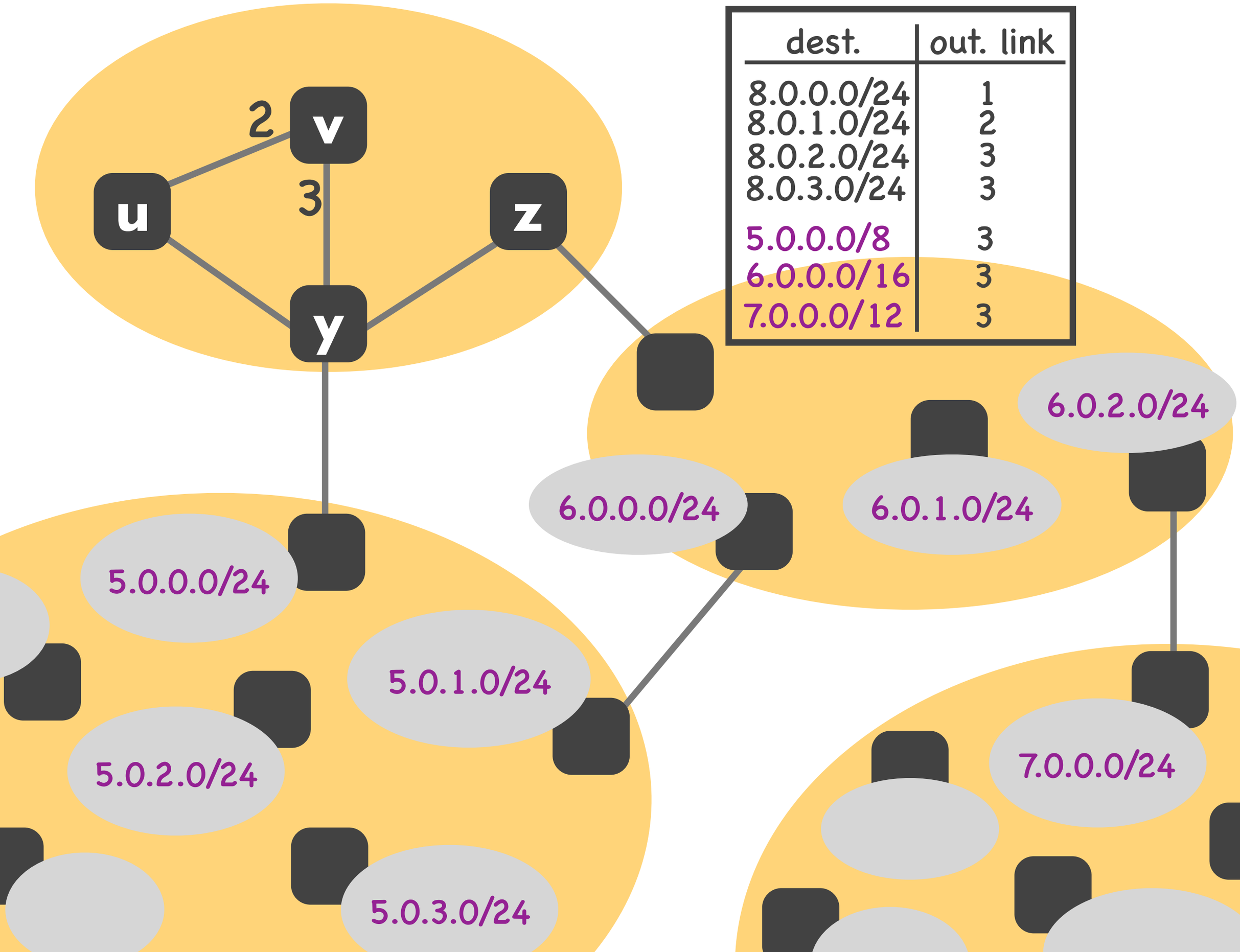
Autonomous Systems

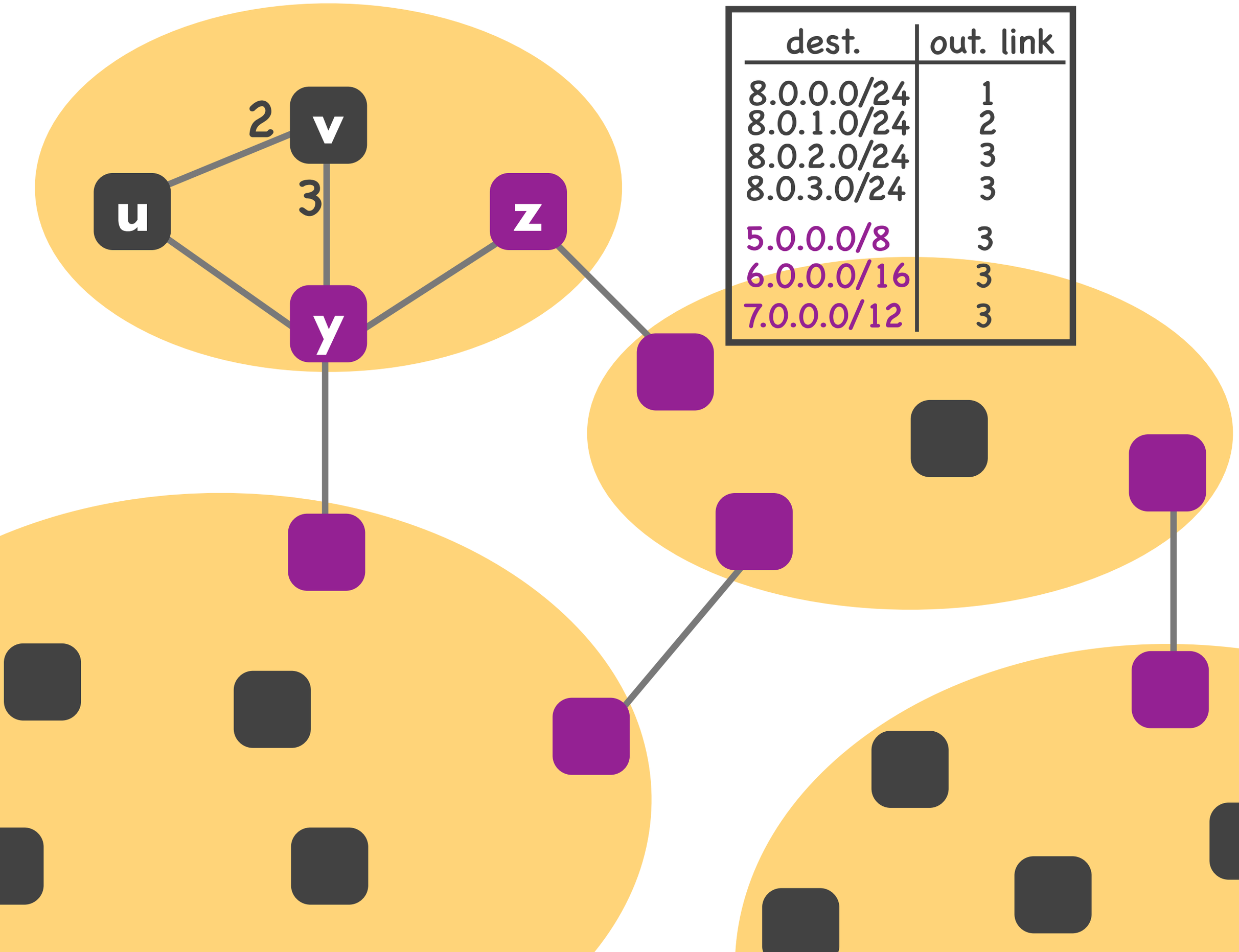






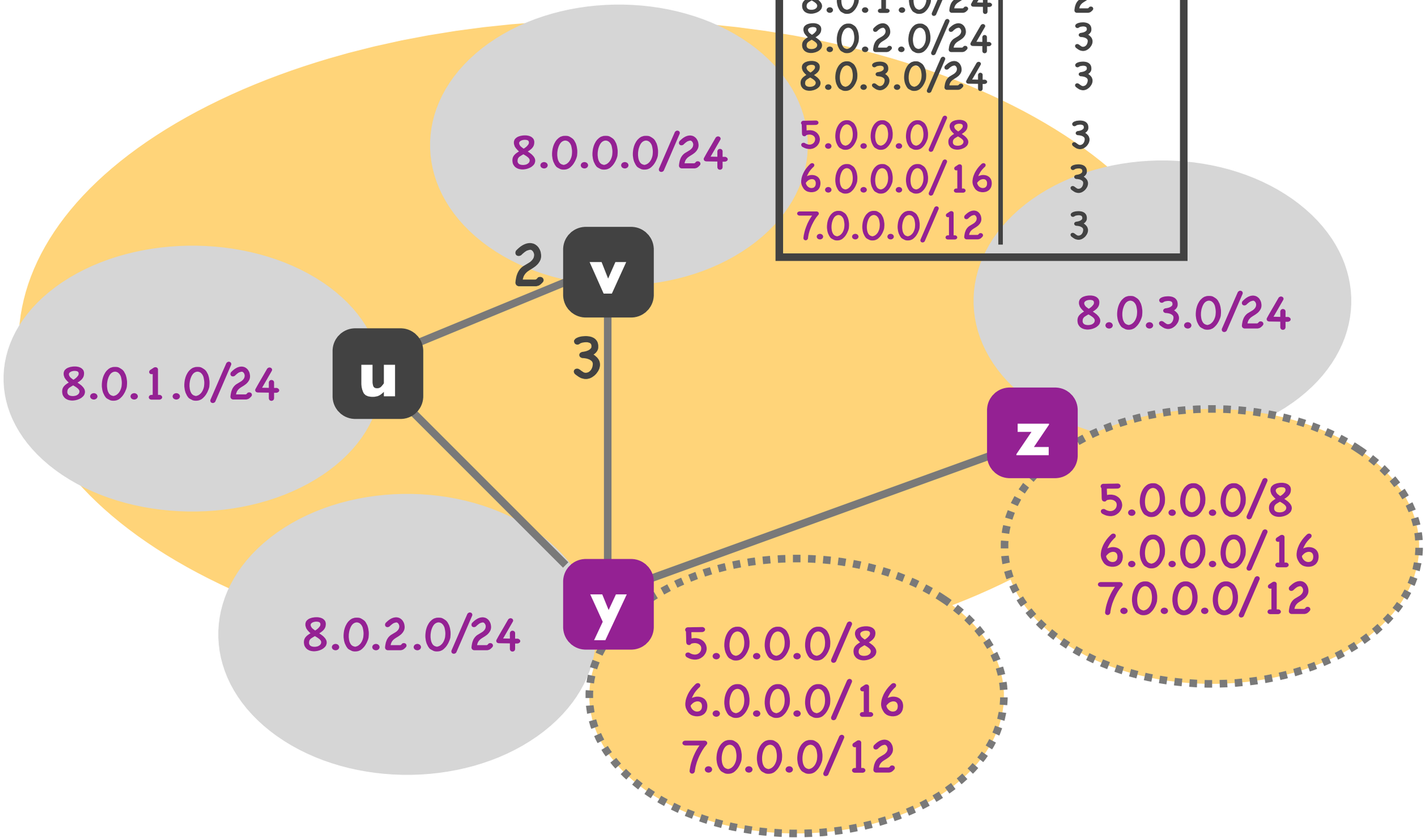
dest.	out. link
8.0.0.0/24	1
8.0.1.0/24	2
8.0.2.0/24	3
8.0.3.0/24	3


u
v
y
z
2
3
dest.
out. link
8.0.0.0/24
1
8.0.1.0/24
2
8.0.2.0/24
3
8.0.3.0/24
3
5.0.0.0/8
3
6.0.0.0/16
3
7.0.0.0/12
3
6.0.2.0/24
6.0.0.0/24
6.0.1.0/24
5.0.0.0/24
5.0.1.0/24
5.0.2.0/24
5.0.3.0/24
7.0.0.0/24



dest.	out. link
8.0.0.0/24	1
8.0.1.0/24	2
8.0.2.0/24	3
8.0.3.0/24	3
5.0.0.0/8	3
6.0.0.0/16	3
7.0.0.0/12	3

dest.	out. link
8.0.0.0/24	1
8.0.1.0/24	2
8.0.2.0/24	3
8.0.3.0/24	3
5.0.0.0/8	3
6.0.0.0/16	3
7.0.0.0/12	3



Internet routing

- Each router learns one route to each IP subnet in local AS
- Each router learns one or a few routes to each foreign AS
  - but not one route to each IP subnet of each foreign AS

Intra-AS routing

- Run by all routers in the same AS
- Goal: propagate routes within local AS
  - each router advertizes routes to its local IP subnets
  - and potentially routes to other ASes that it has learned through BGP
- OSPF, RIP, …

Inter-AS routing

- Run by all border routers between ASes
- Goal: propagate routes outside local AS
  - each border router talks to external neighbors (eBGP)
  - and to the other border routers of the local AS (iBGP)

- BGP = Border Gateway Protocol

Internet routing challenges

- Scale
  - link-state would cause flooding
  - distance-vector would not converge
- Administrative autonomy
  - an ISP may not want to do least-cost routing
  - may want to hide its link costs from the world

Solution: hierarchy

- Scale: state not per IP subnet
  - each router needs forwarding entries
    per local IP subnets and foreign IP prefixes
  - each router may communicate with
    all other local routers and external neighbors

- Administrative autonomy:
  each AS chooses its own intra-AS
  routing protocol