

Routing in the Internet

CPSC 433/533, Spring 2021

Anurag Khandelwal

Administrivia

- Project I (Routing) out today — woohoo!
 - **One suggestion:** Start early... seriously
 - **Lead:** Josh Beasley; **Other TFs/ULAs:** Jonathan, Ramlan
 - Reach out to folks in that order; if none of them can help, talk to me
 - **Beware:** I will address conceptual issues only. Code is your responsibility.
 - A project overview recording will be posted on Canvas soon

Recap: Link State Routing

Recap: Link State Routing

- Obtain **global** picture of the network at each router
 - By flooding link-state information (periodic, on change of state)

Recap: Link State Routing

- Obtain **global** picture of the network at each router
 - By flooding link-state information (periodic, on change of state)
 - Run Dijkstra's Algorithm on each router

Recap: Link State Routing

- Obtain **global** picture of the network at each router
 - By flooding link-state information (periodic, on change of state)
- Run Dijkstra's Algorithm on each router
- **Convergence delay** causes routing loops, dead-ends, reordered packets

Recap: Distance Vector Routing

Recap: Distance Vector Routing

- **Distributed** algorithm!

Recap: Distance Vector Routing

- **Distributed** algorithm!
- Each router maintains distance vectors (DVs) for **itself** and **its neighbors**
 - **Initially:** based on local link costs

Recap: Distance Vector Routing

- **Distributed** algorithm!
- Each router maintains distance vectors (DVs) for **itself** and **its neighbors**
 - **Initially:** based on local link costs
- Sends out its initial DV to immediate neighbors

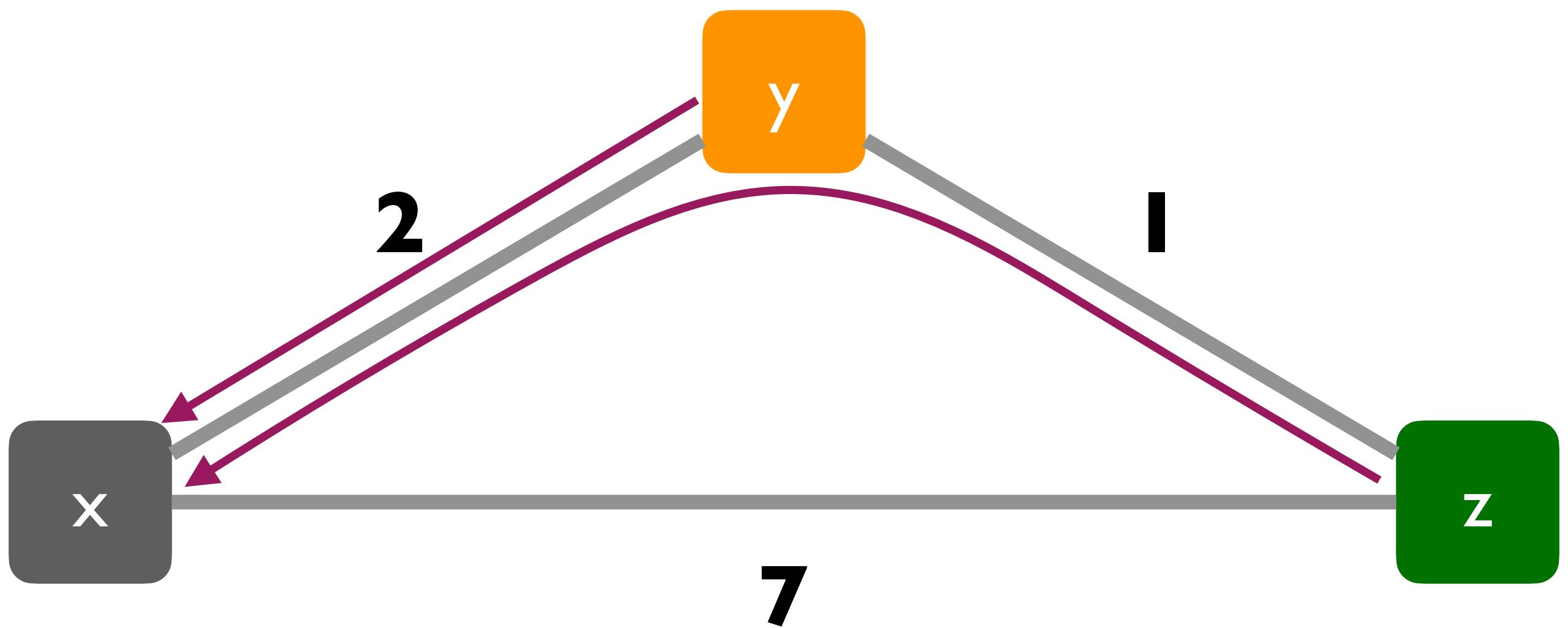
Recap: Distance Vector Routing

- **Distributed** algorithm!
- Each router maintains distance vectors (DVs) for **itself** and **its neighbors**
 - **Initially:** based on local link costs
- Sends out its initial DV to immediate neighbors
- If the received DVs indicate shorter path to other routers (Bellman-Ford Equation):
 - Update own DV and advertise it to neighbors if changed

Recap: Distance Vector Routing

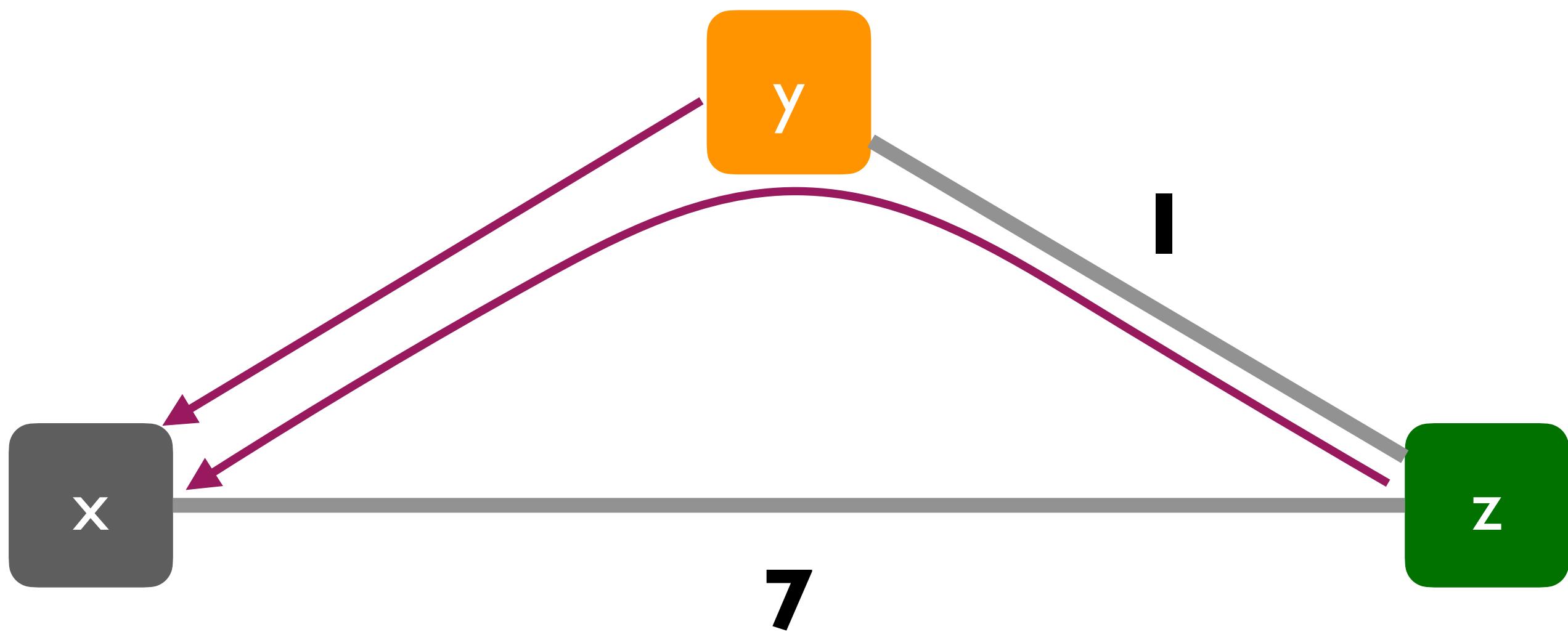
- **Distributed** algorithm!
- Each router maintains distance vectors (DVs) for **itself** and **its neighbors**
 - **Initially:** based on local link costs
- Sends out its initial DV to immediate neighbors
- If the received DVs indicate shorter path to other routers (Bellman-Ford Equation):
 - Update own DV and advertise it to neighbors if changed
- Convergence delay is **worse** for DV
 - Count-to-infinity scenario!

	x	y	z
y	2	0	1
z	3	1	0



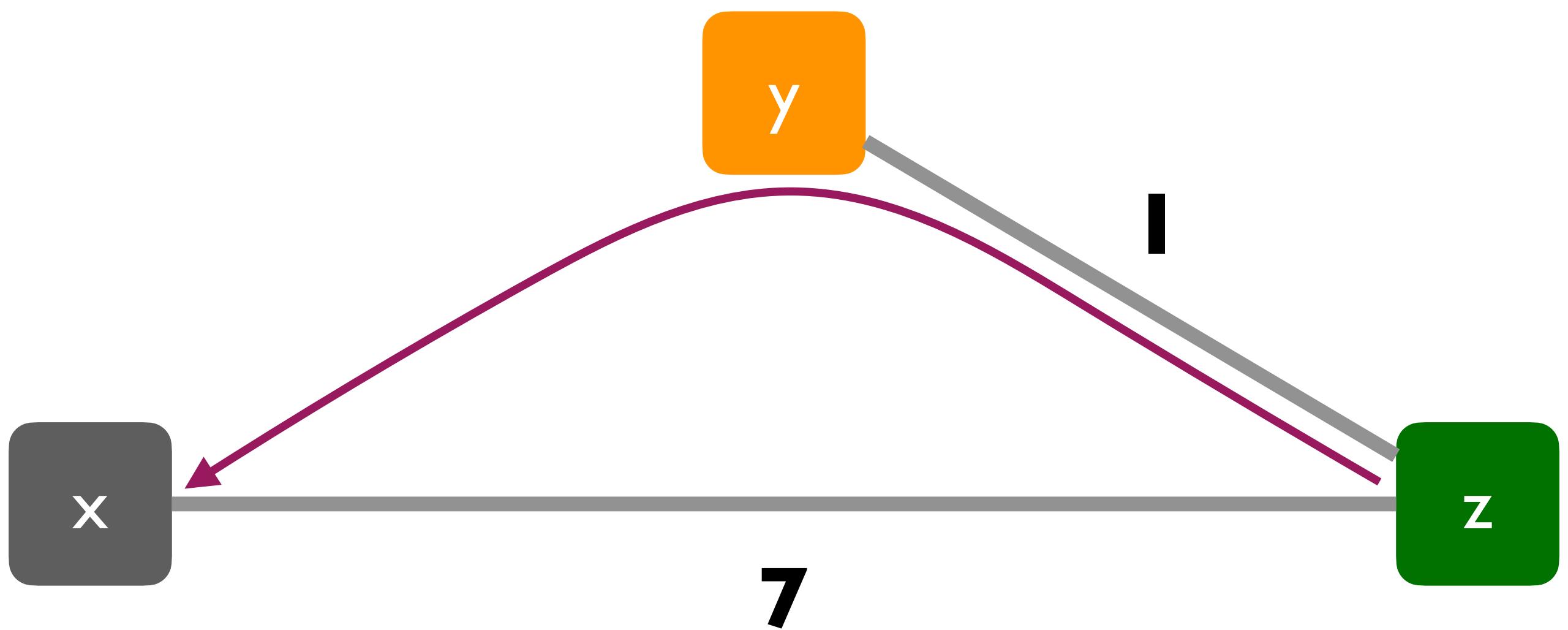
	x	y	z
y	2	0	1
z	3	1	0

	x	y	z
y	2	0	1
z	3	1	0



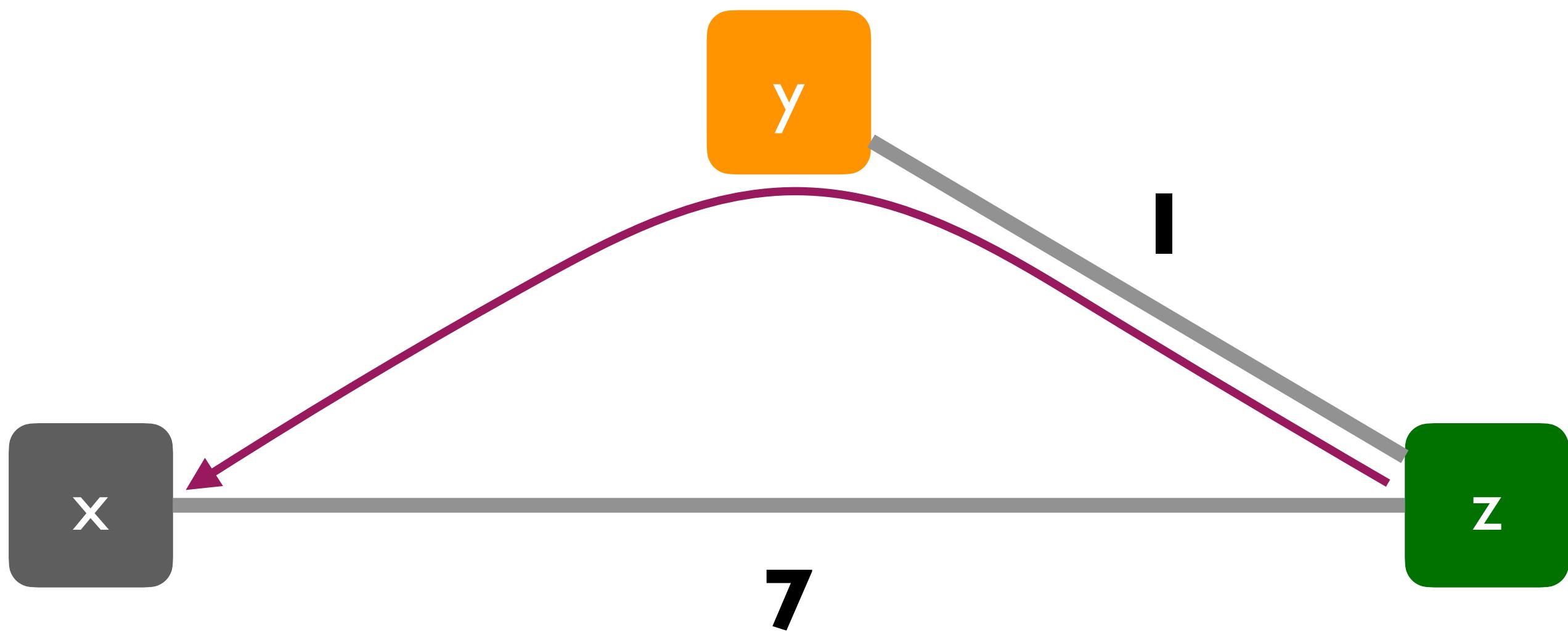
	x	y	z
y	2	0	1
z	3	1	0

	x	y	z
x			
y		0	1
z	3	1	0



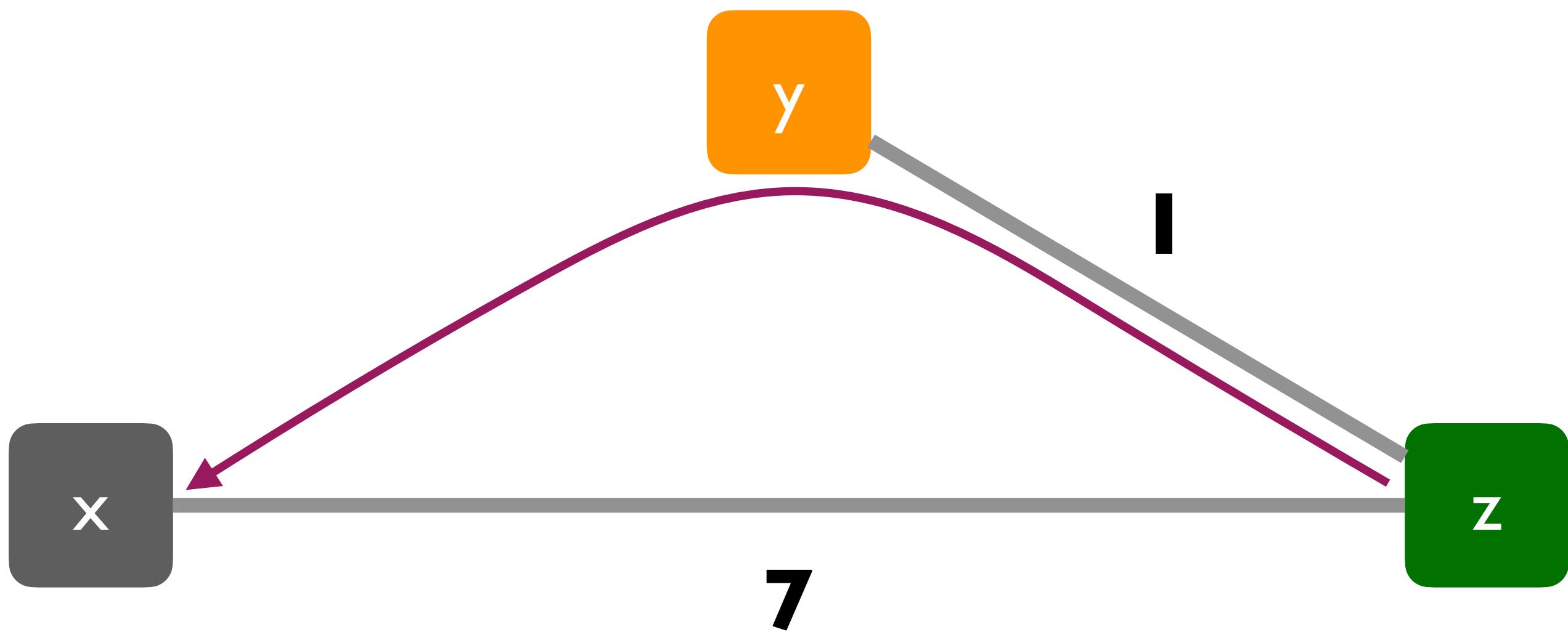
	x	y	z
x			
y	2	0	1
z	3	1	0

	x	y	z
y			1
z	3	1	0



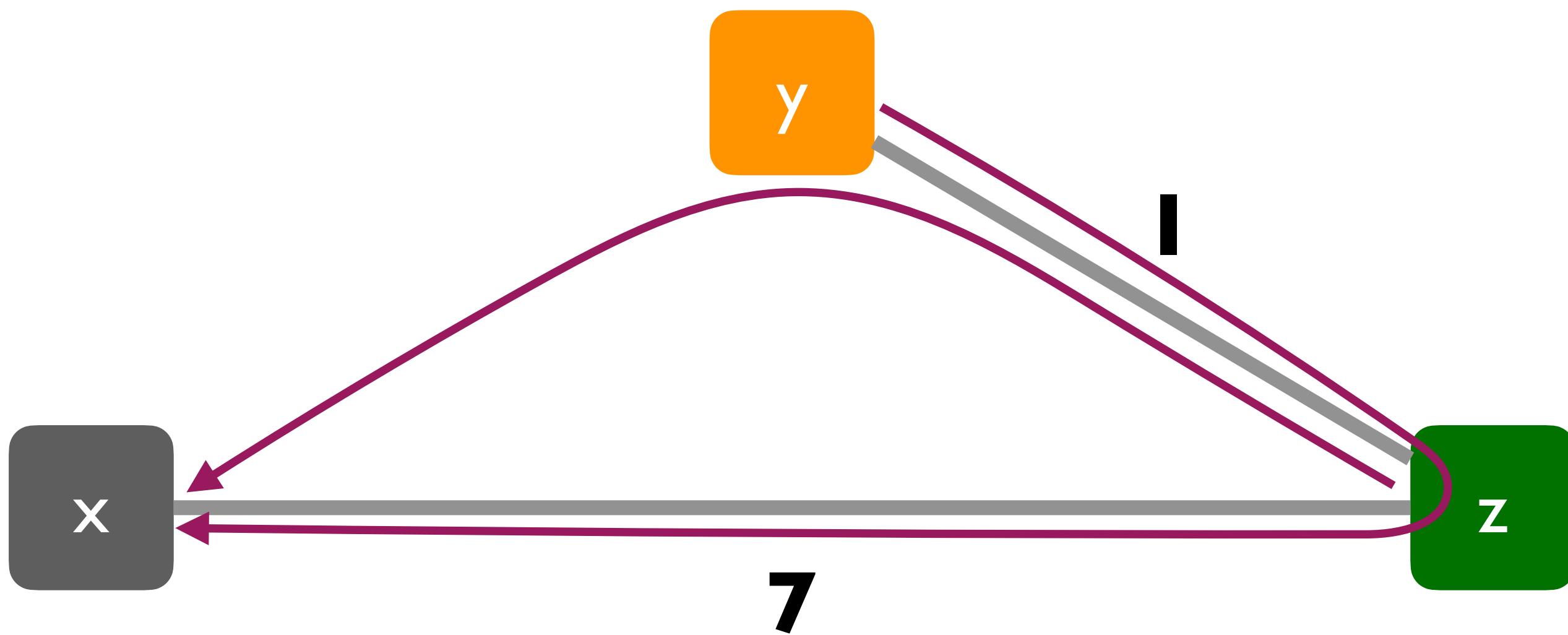
	x	y	z
y	2	0	1
z	3	1	0

	x	y	z
y	4	0	1
z	3	1	0



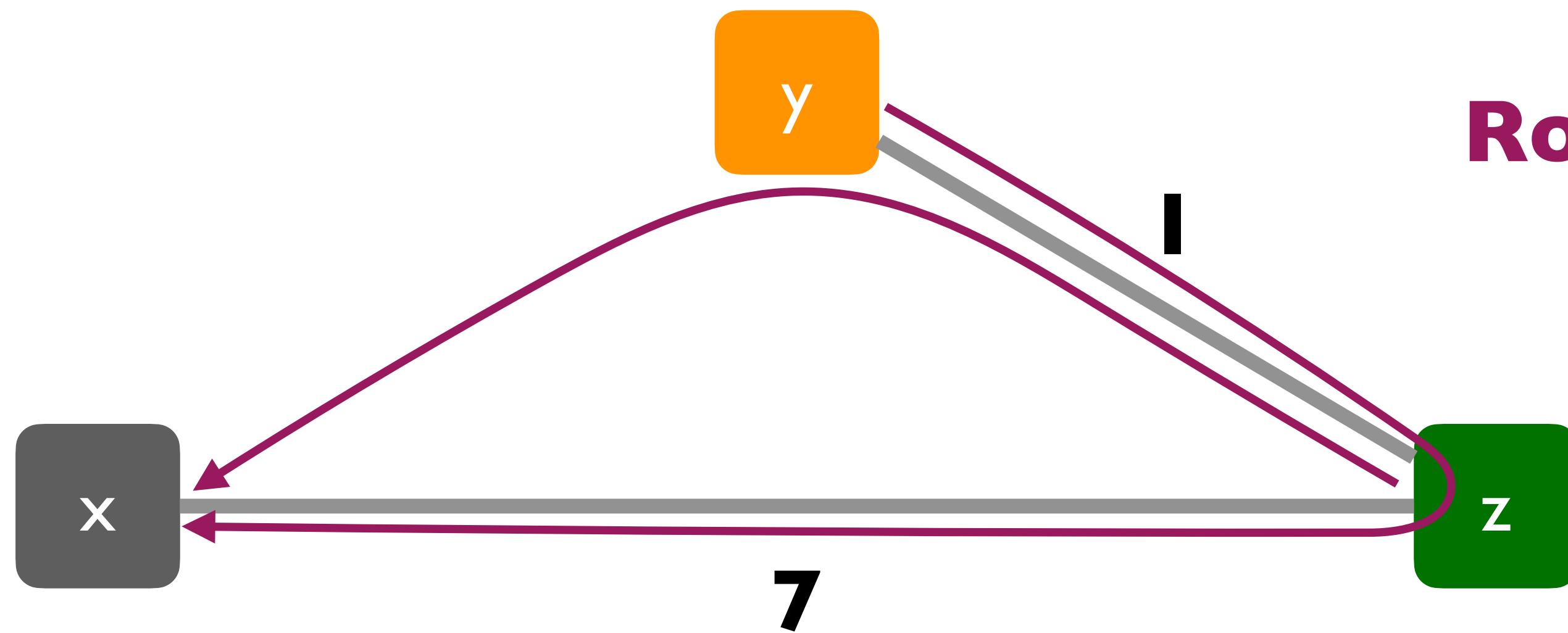
	x	y	z
y	2	0	1
z	3	1	0

	x	y	z
y	4	0	1
z	3	1	0



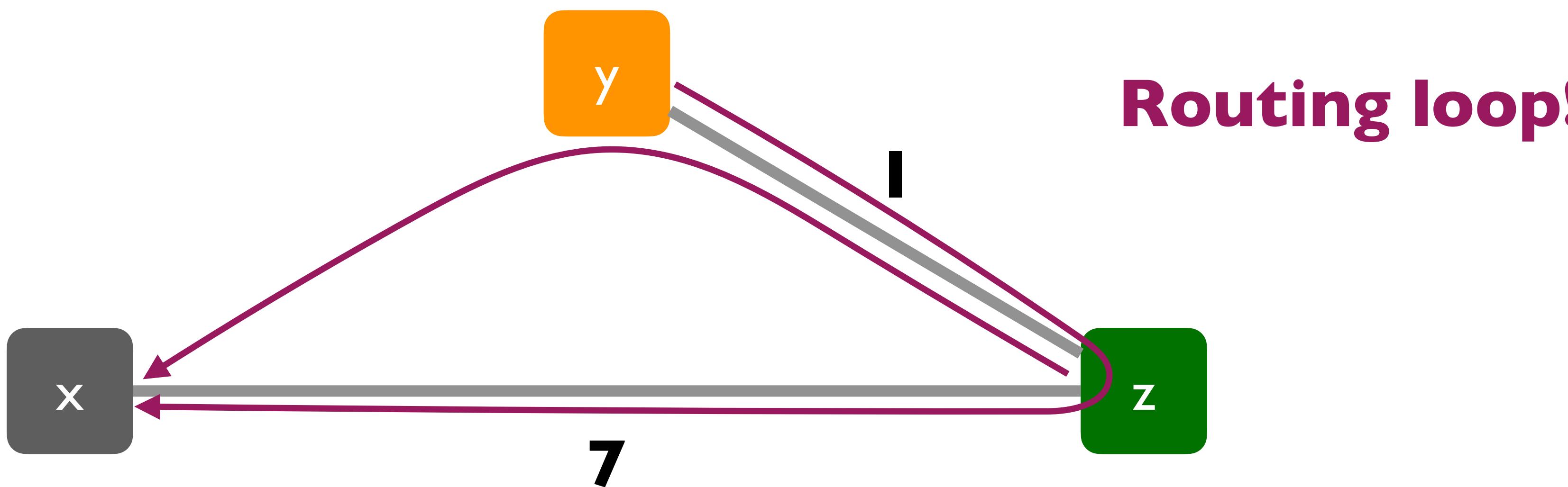
	x	y	z
y	2	0	1
z	3	1	0

	x	y	z
y	4	0	1
z	3	1	0



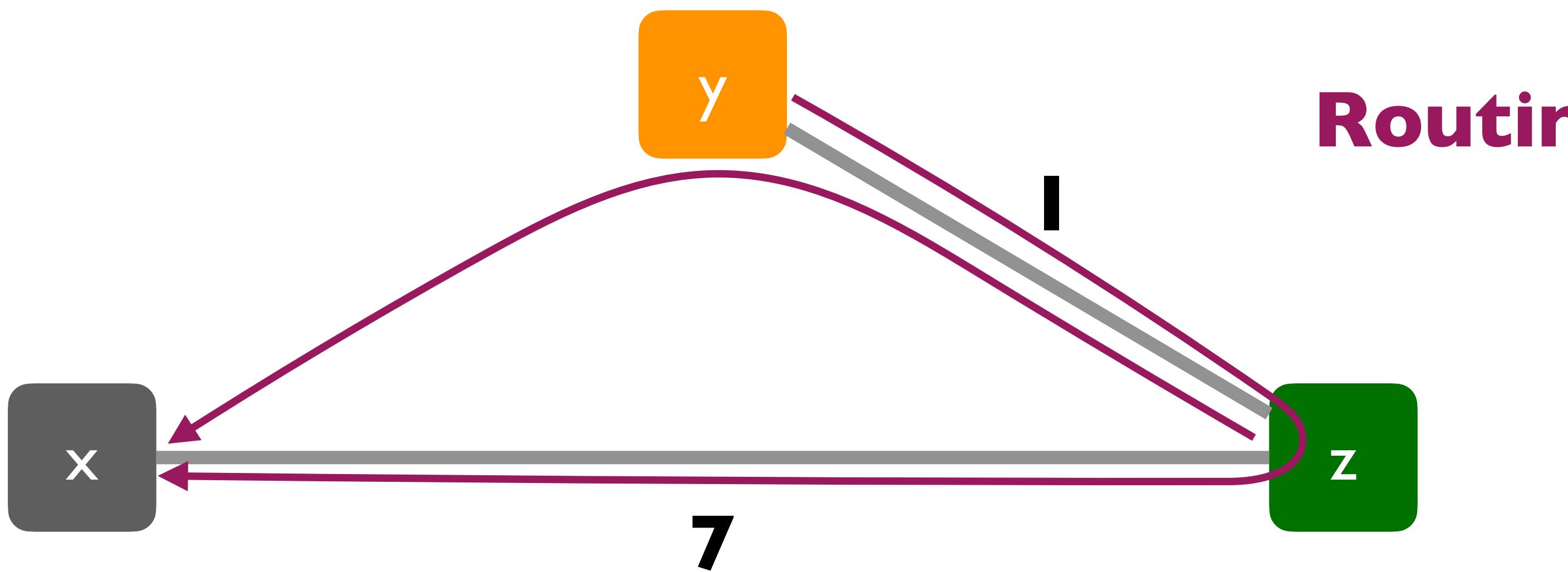
	x	y	z
y	2	0	1
z	3	1	0

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



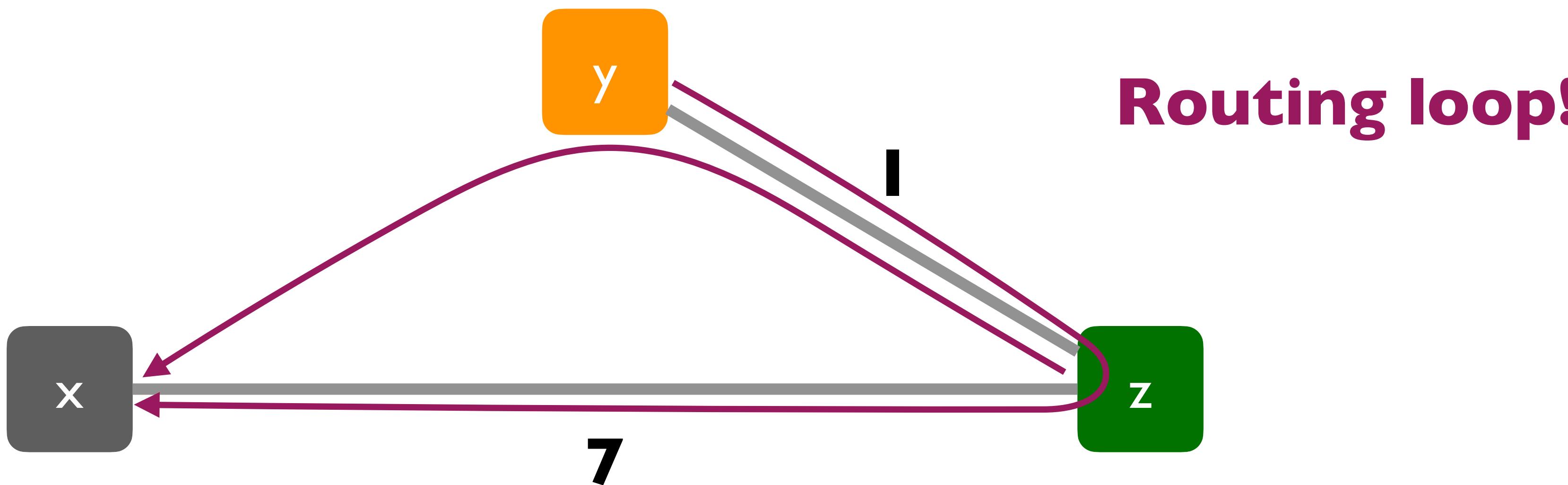
	x	y	z
x			
y	4	0	1
z	3	1	0

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



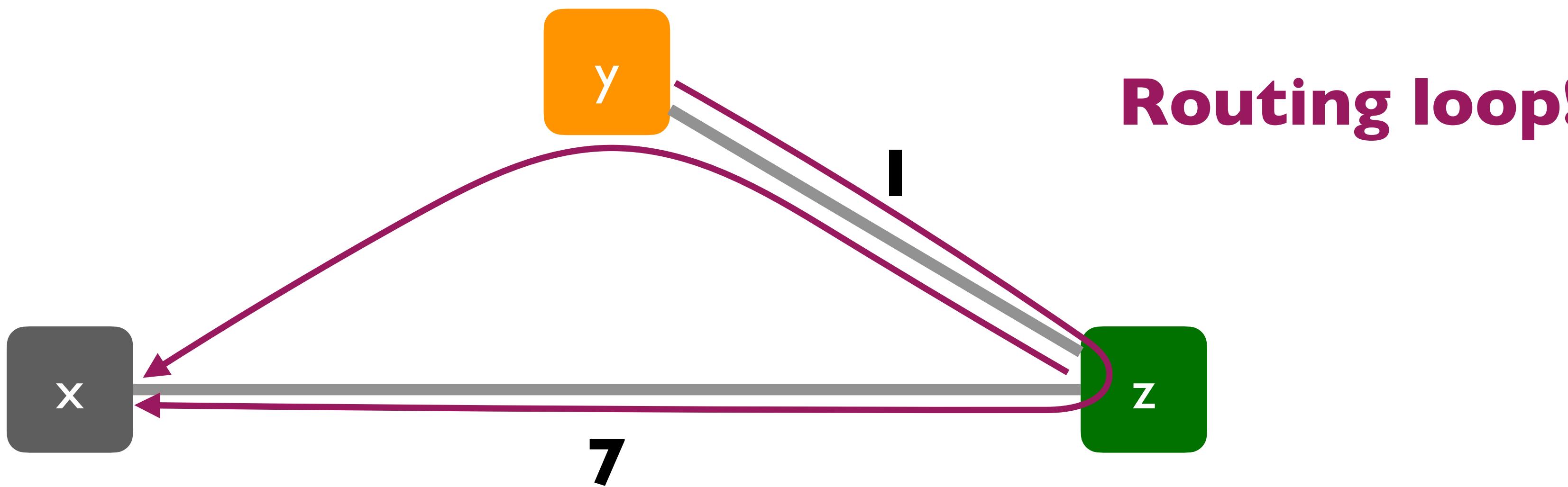
	x	y	z
x			
y	4	0	1
z	3	1	0

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



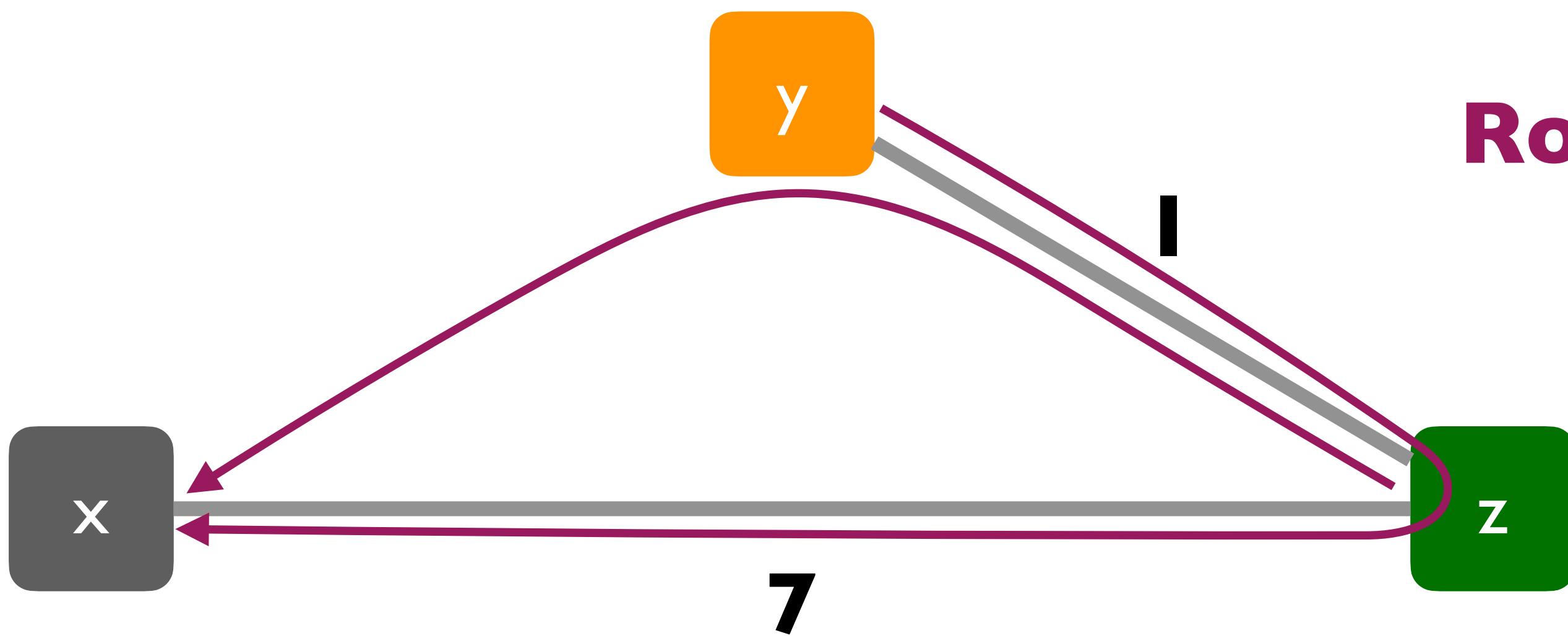
	x	y	z
x			
y	4	0	1
z	5	1	0

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



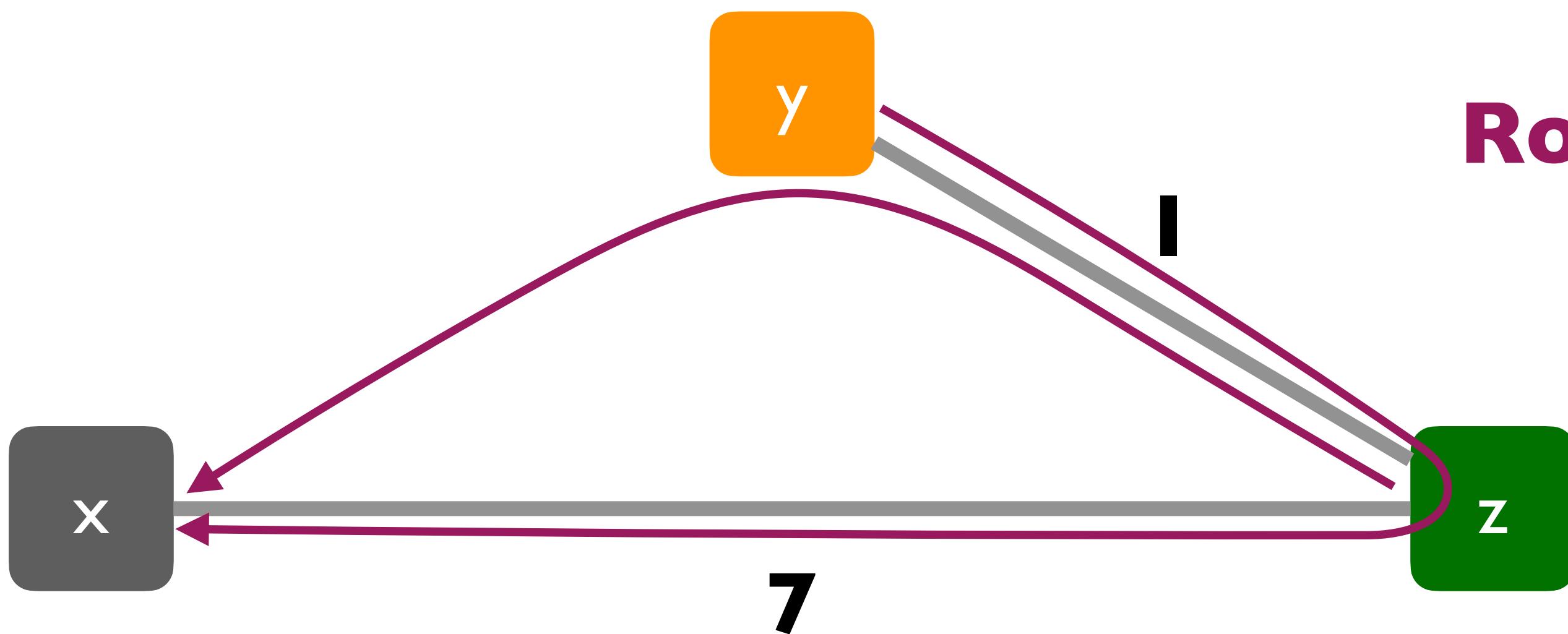
	x	y	z
x			
y	4	0	1
z	5	1	0

	x	y	z
y	4	0	1
z	5	1	0



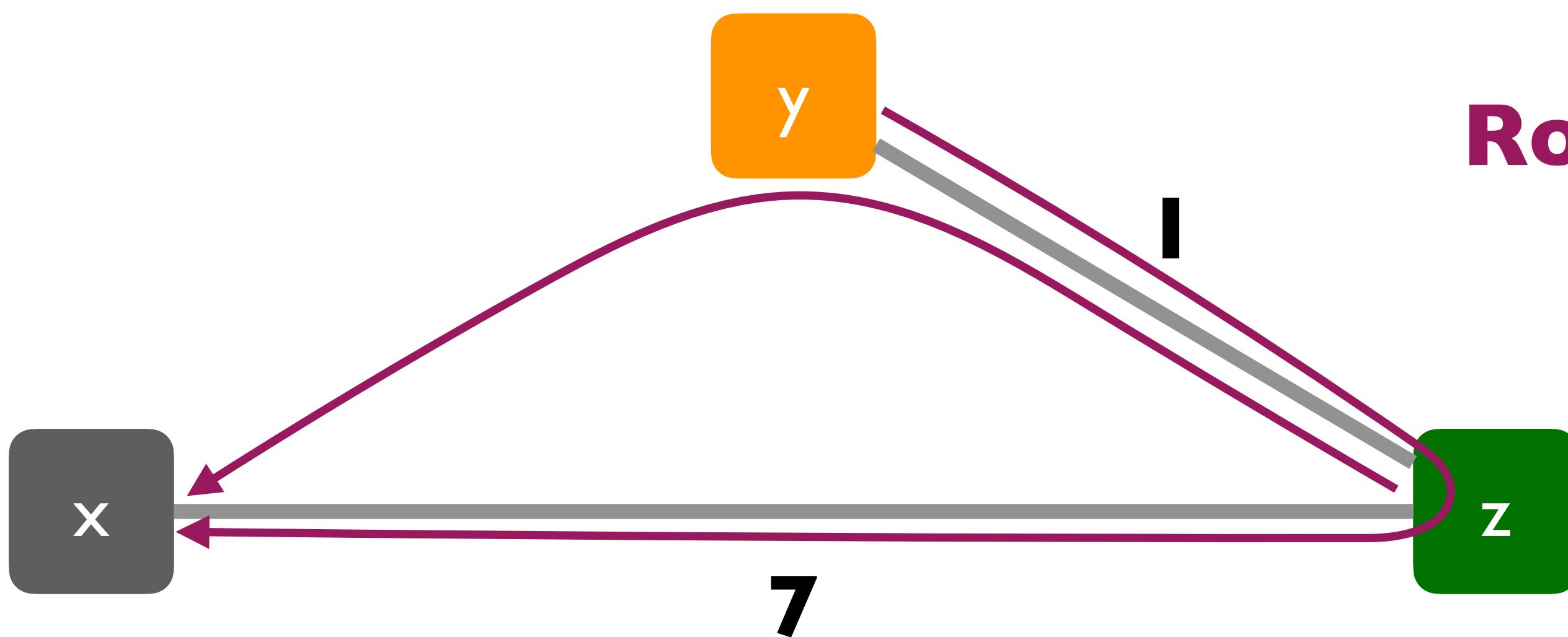
	x	y	z
y	4	0	1
z	5	1	0

	x	y	z
y	6	0	1
z	5	1	0



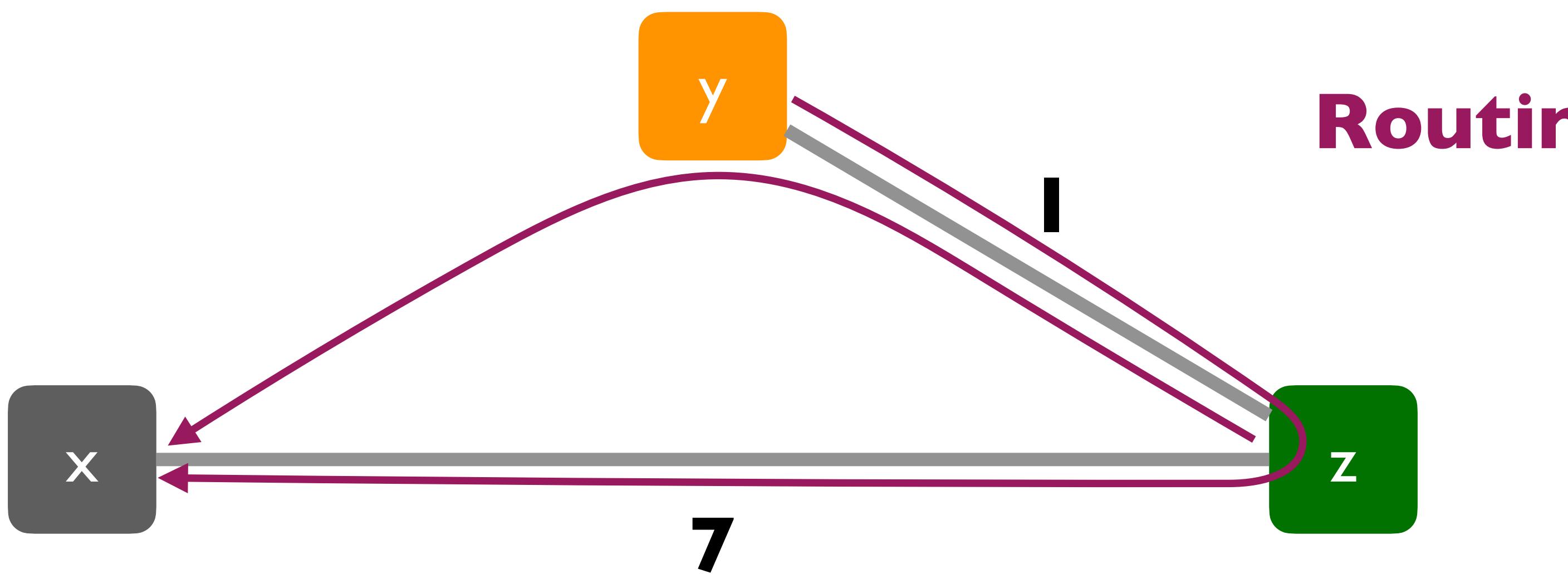
	x	y	z
y	4	0	1
z	5	1	0

	x	y	z
y	6	0	1
z	5	1	0



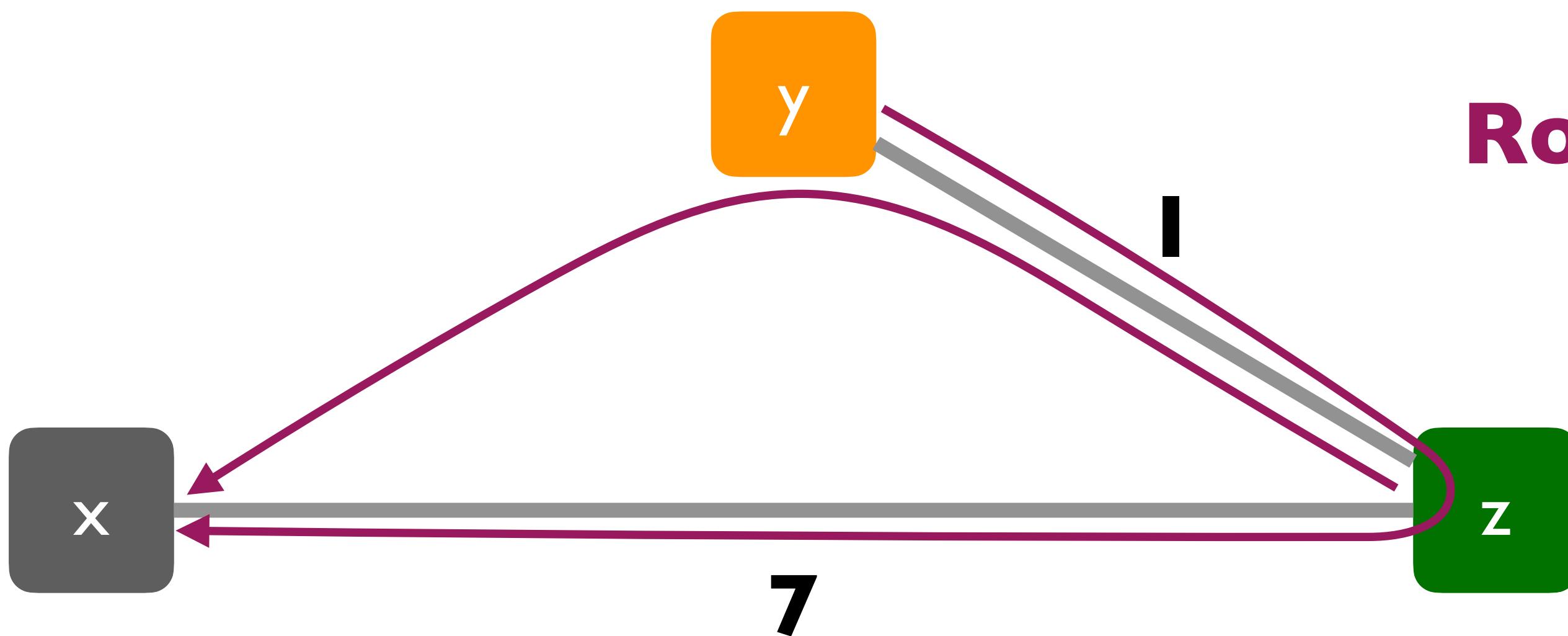
	x	y	z
y	6	0	1
z	5	1	0

	x	y	z
y	6	0	1
z	5	1	0



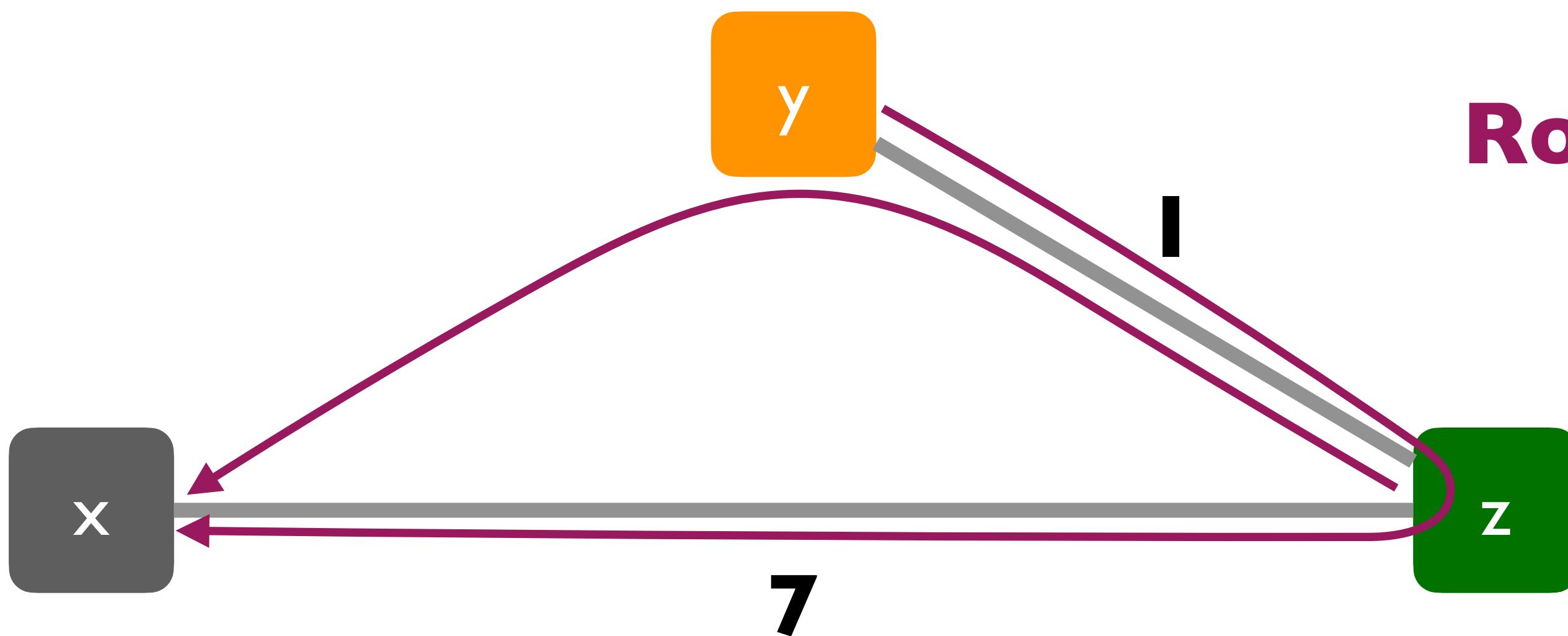
	x	y	z
y	6	0	1
z	5	1	0

	x	y	z
y	6	0	1
z	5	1	0



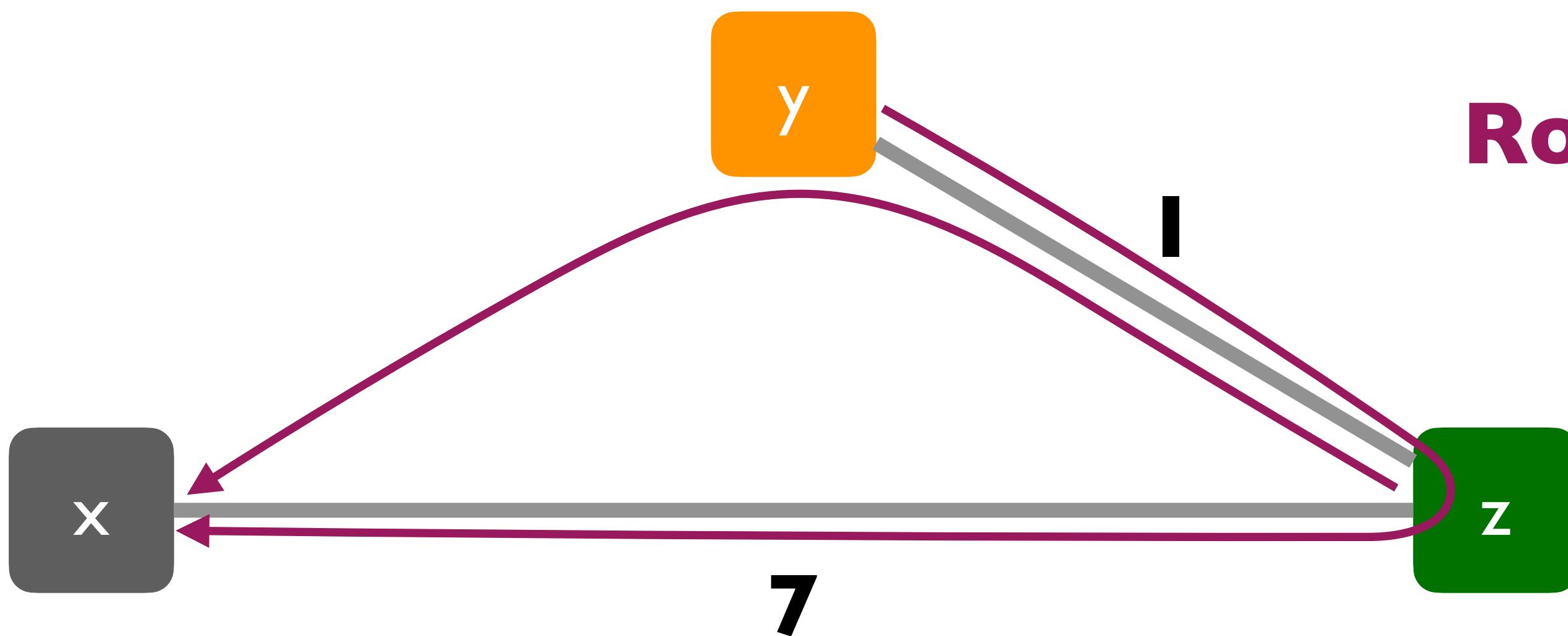
	x	y	z
y	6	0	1
z	7	1	0

	x	y	z
y	6	0	1
z	7	1	0



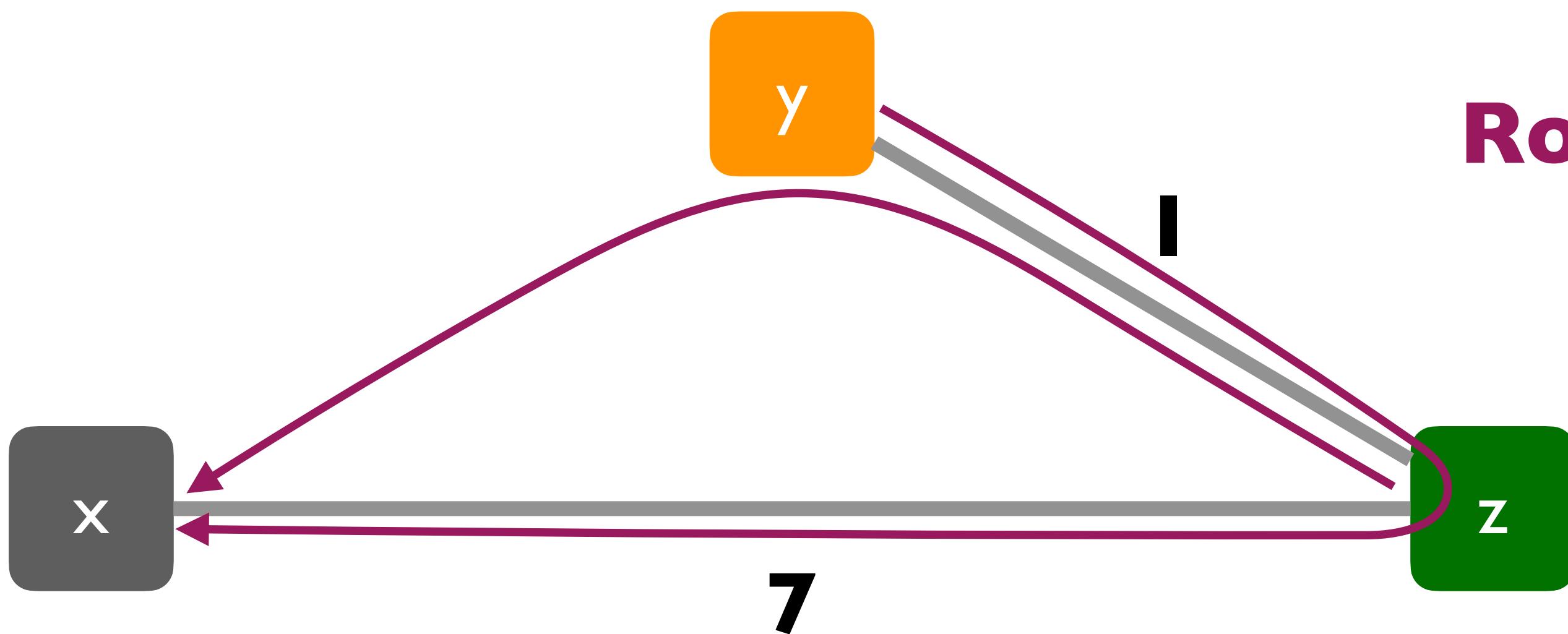
	x	y	z
y	6	0	1
z	7	1	0

	x	y	z
y	6	0	1
z	7	1	0



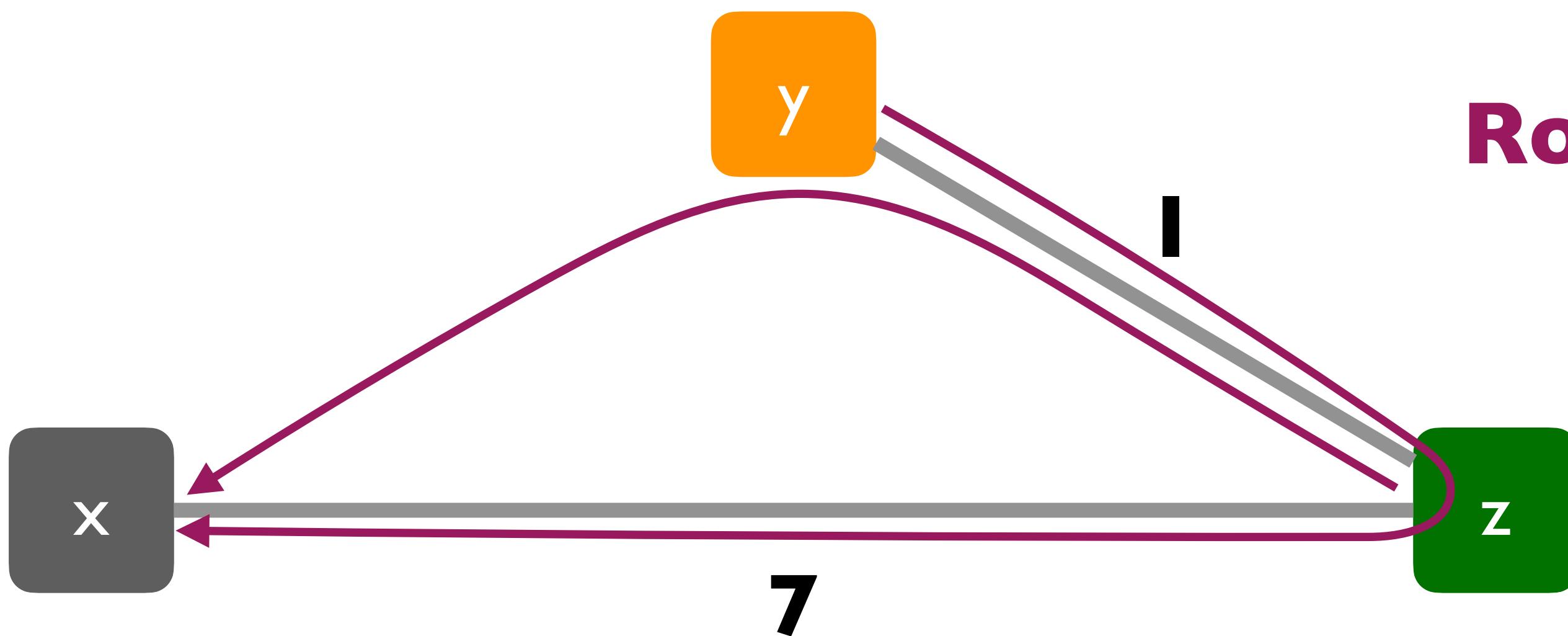
	x	y	z
y	6	0	1
z	7	1	0

	x	y	z
y	8	0	1
z	7	1	0



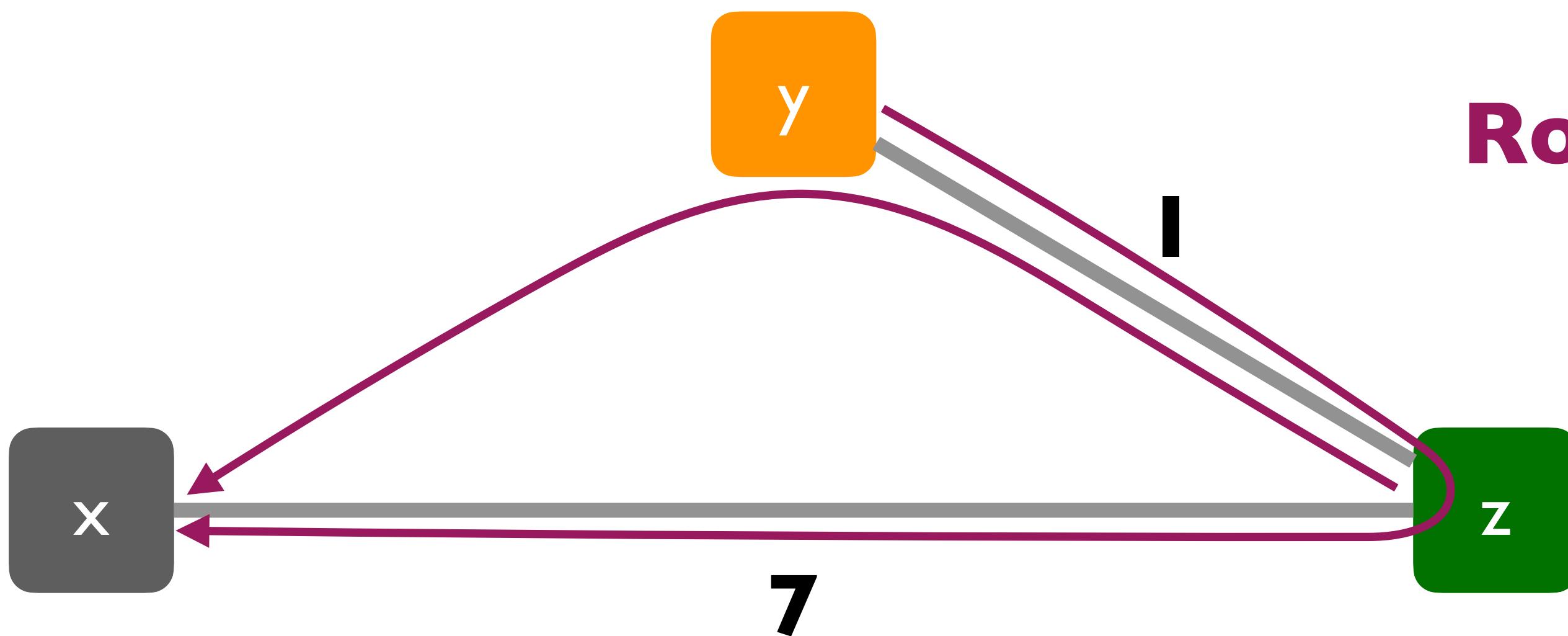
	x	y	z
y	6	0	1
z	7	1	0

	x	y	z
y	8	0	1
z	7	1	0



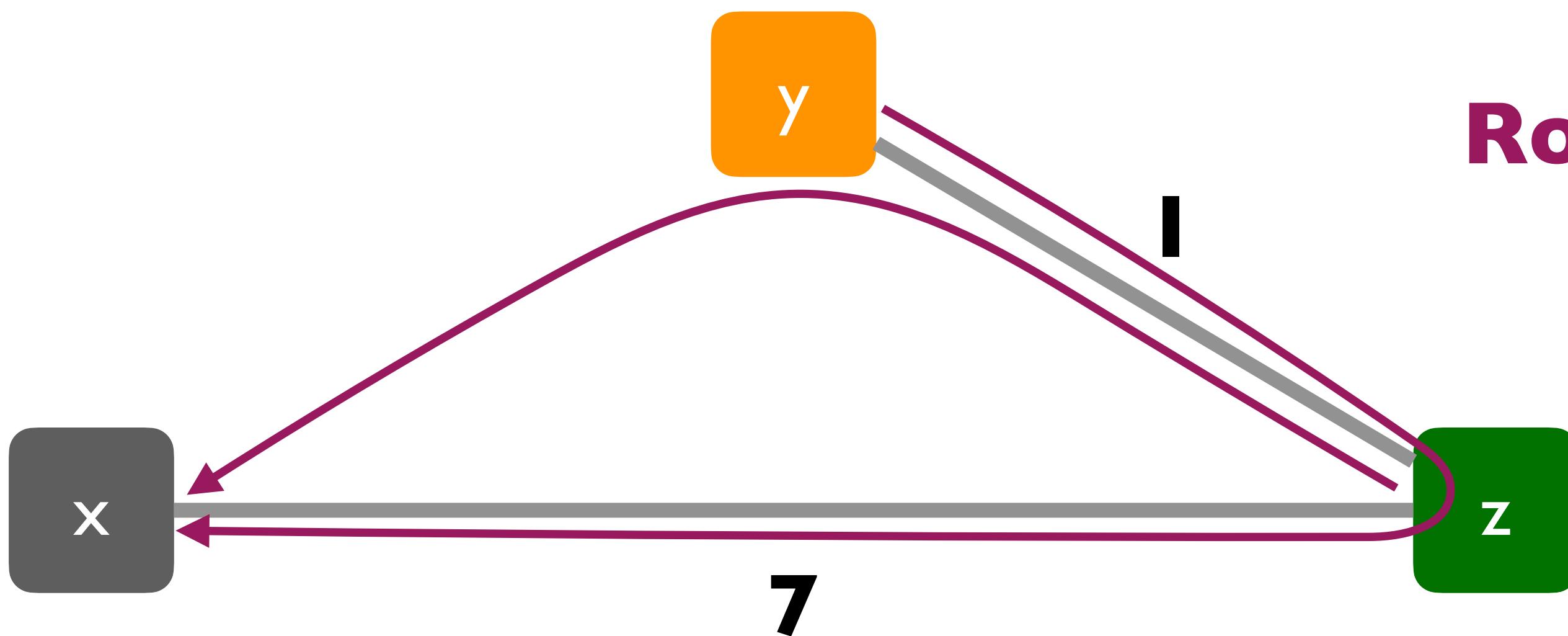
	x	y	z
y	8	0	1
z	7	1	0

	x	y	z
y	8	0	1
z	7	1	0



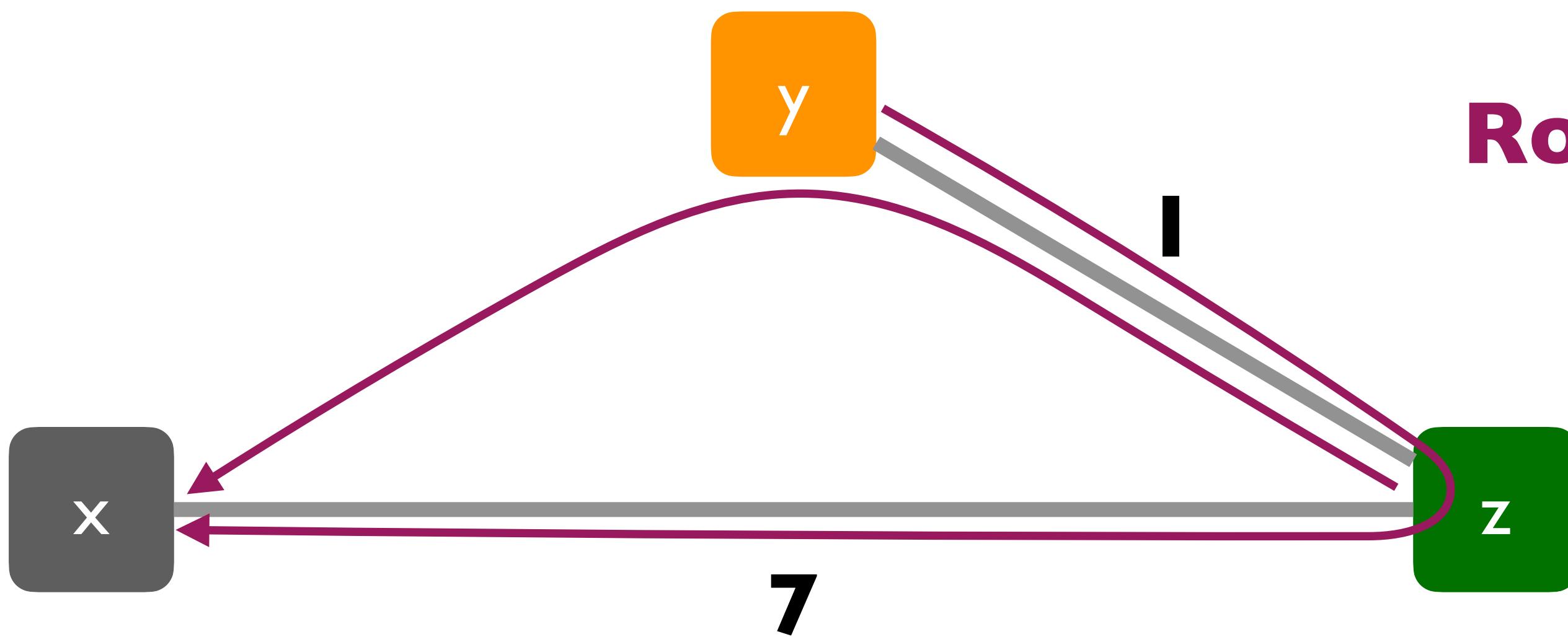
	x	y	z
y	8	0	1
z	7	1	0

	x	y	z
y	8	0	1
z	7	1	0



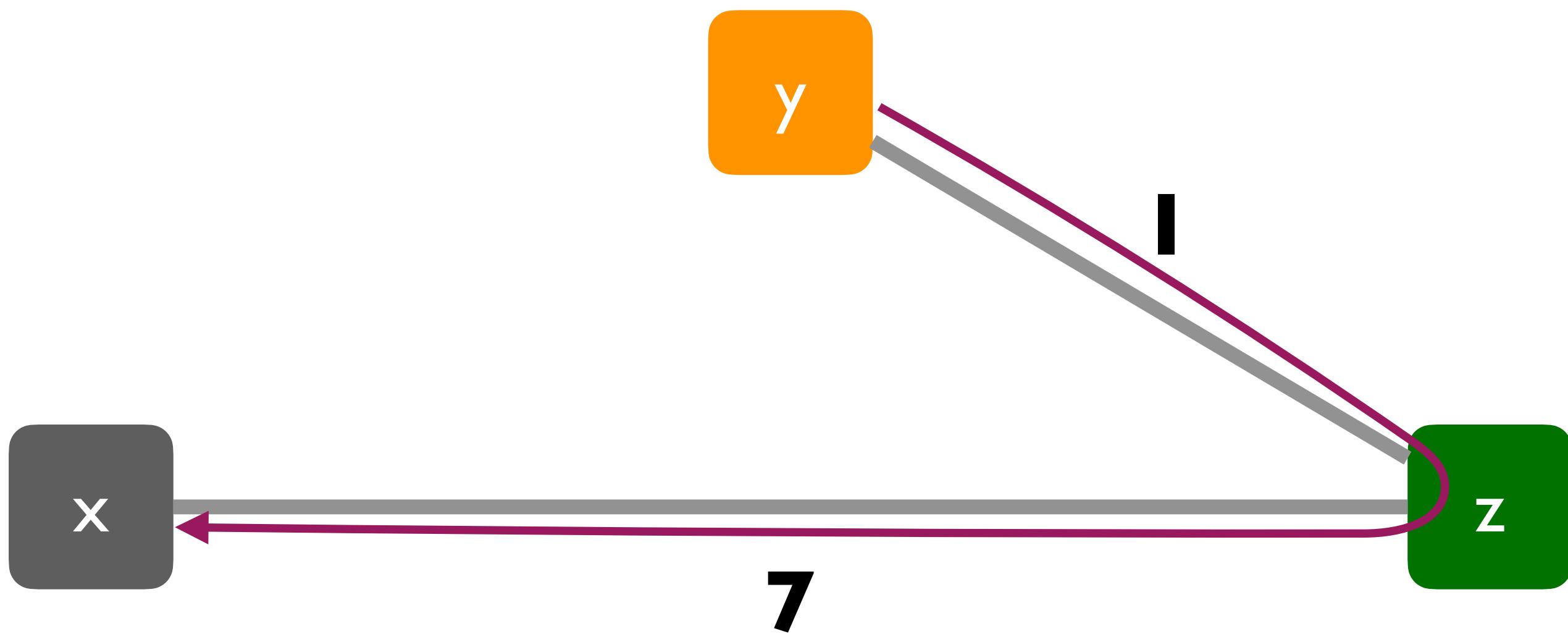
	x	y	z
y	8	0	1
z	7	1	0

	x	y	z
y	8	0	1
z	7	1	0



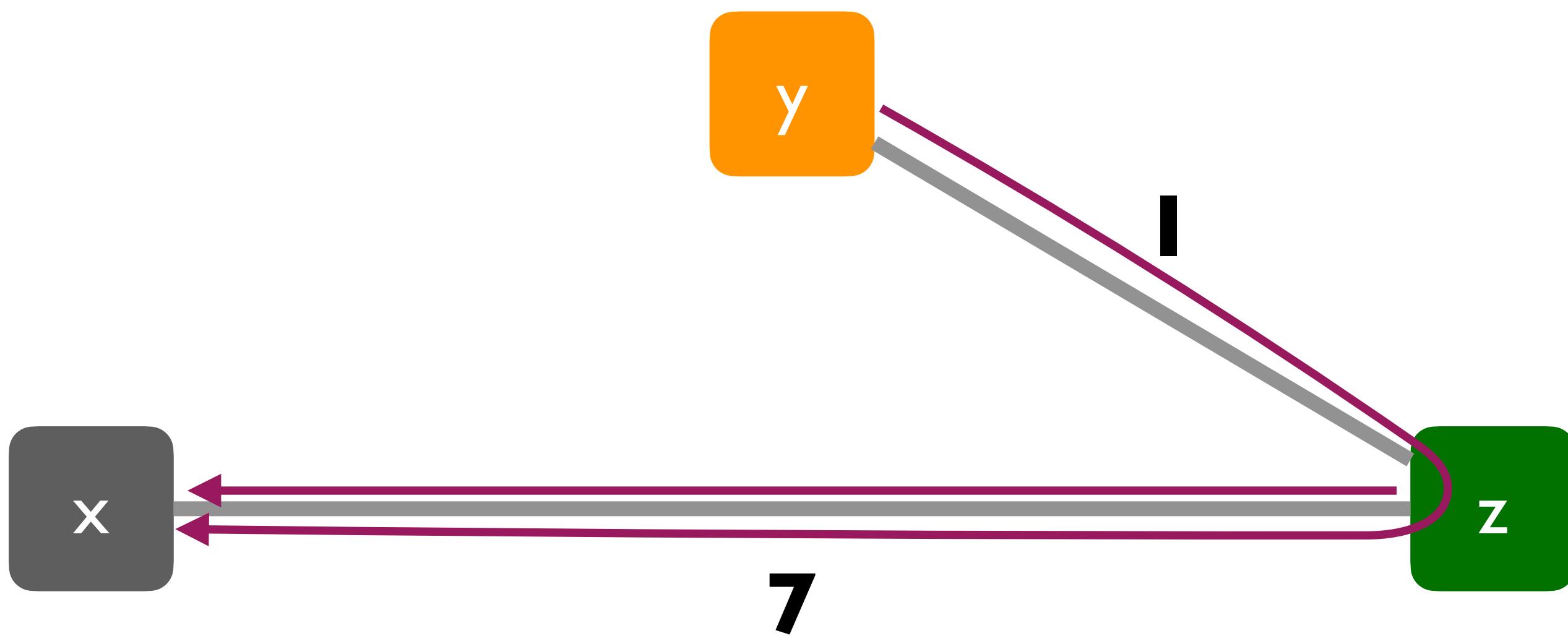
	x	y	z
y	8	0	1
z	7	1	0

	x	y	z
y	8	0	1
z	7	1	0



	x	y	z
y	8	0	1
z	7	1	0

	x	y	z
y	8	0	1
z	7	1	0



	x	y	z
y	8	0	1
z	7	1	0

What went wrong there?

What went wrong there?

- What do you think?

What went wrong there?

- What do you think?
- Why would you advertise a path back to the router who advertised it to you?!

What went wrong there?

- What do you think?
- Why would you advertise a path back to the router who advertised it to you?!
- Telling them about your entry going through them:

What went wrong there?

- What do you think?
- Why would you advertise a path back to the router who advertised it to you?!
- Telling them about your entry going through them:
 - Doesn't tell them anything new

What went wrong there?

- What do you think?
- Why would you advertise a path back to the router who advertised it to you?!
- Telling them about your entry going through them:
 - Doesn't tell them anything new
 - Can mislead them into thinking you have a different path!

So what's the solution then?

So what's the solution then?

- **Solution: If you are using a next-hop's path for some destination...**

So what's the solution then?

- **Solution: If you are using a next-hop's path for some destination...**
 - ...don't advertise that destination to them!

So what's the solution then?

- **Solution: If you are using a next-hop's path for some destination...**
 - ...don't advertise that destination to them!
 - Called **split horizon**

So what's the solution then?

- **Solution: If you are using a next-hop's path for some destination...**
 - ...don't advertise that destination to them!
 - Called **split horizon**
- **Further improvement: Split-horizon with poisoned reverse!**

So what's the solution then?

- **Solution: If you are using a next-hop's path for some destination...**
 - ...don't advertise that destination to them!
 - Called **split horizon**
- **Further improvement: Split-horizon with poisoned reverse!**
 - If you are using a next-hop's path for some destination...

So what's the solution then?

- **Solution: If you are using a next-hop's path for some destination...**
 - ...don't advertise that destination to them!
 - Called **split horizon**
- **Further improvement: Split-horizon with poisoned reverse!**
 - If you are using a next-hop's path for some destination...
 - ...explicitly advertise a path of infinity to them for that destination

So what's the solution then?

- **Solution: If you are using a next-hop's path for some destination...**
 - ...don't advertise that destination to them!
 - Called **split horizon**
- **Further improvement: Split-horizon with poisoned reverse!**
 - If you are using a next-hop's path for some destination...
 - ...explicitly advertise a path of infinity to them for that destination
 - Prevents them from trying to route back through you if failures happen

So what's the solution then?

- **Solution: If you are using a next-hop's path for some destination...**
 - ...don't advertise that destination to them!
 - Called **split horizon**
- **Further improvement: Split-horizon with poisoned reverse!**
 - If you are using a next-hop's path for some destination...
 - ...explicitly advertise a path of infinity to them for that destination
 - Prevents them from trying to route back through you if failures happen
- **More concretely:**

So what's the solution then?

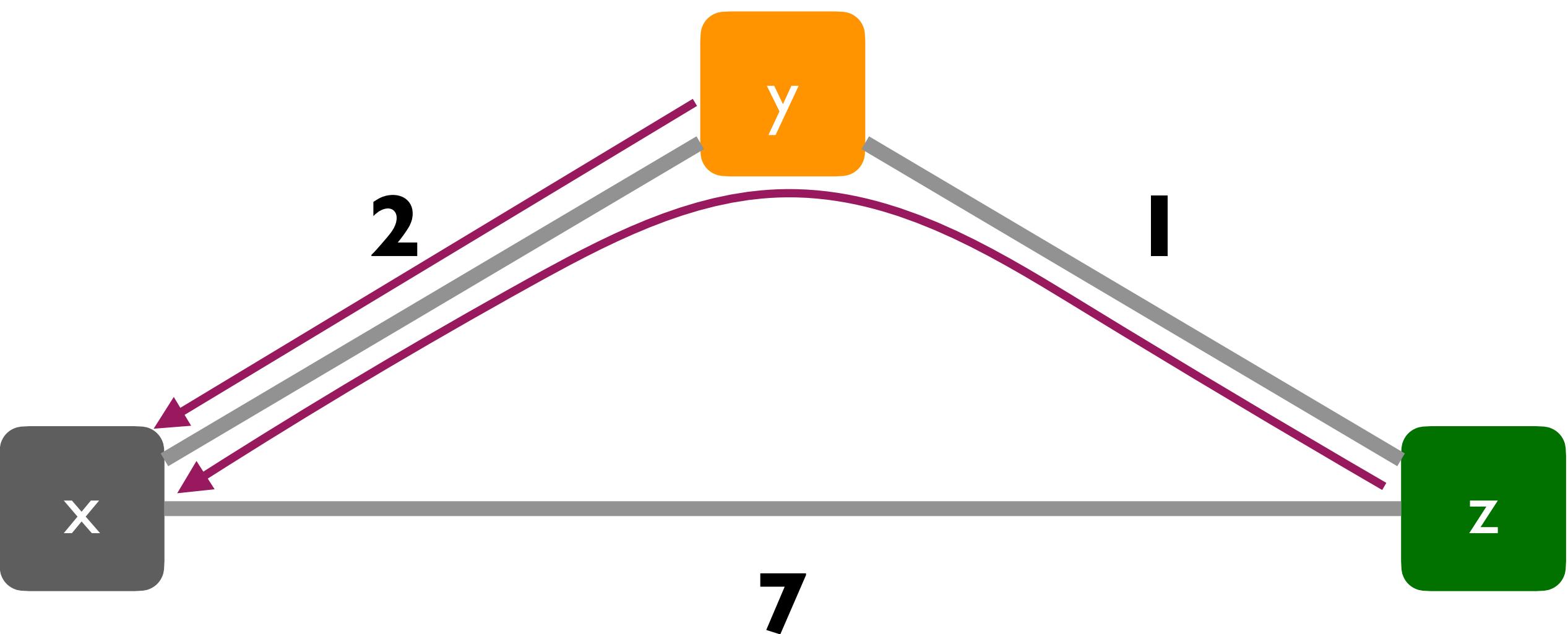
- **Solution: If you are using a next-hop's path for some destination...**
 - ...don't advertise that destination to them!
 - Called **split horizon**
- **Further improvement: Split-horizon with poisoned reverse!**
 - If you are using a next-hop's path for some destination...
 - ...explicitly advertise a path of infinity to them for that destination
 - Prevents them from trying to route back through you if failures happen
- **More concretely:**
 - If z routes to x through y , z advertises to y that its cost to x is infinite

So what's the solution then?

- **Solution: If you are using a next-hop's path for some destination...**
 - ...don't advertise that destination to them!
 - Called **split horizon**
- **Further improvement: Split-horizon with poisoned reverse!**
 - If you are using a next-hop's path for some destination...
 - ...explicitly advertise a path of infinity to them for that destination
 - Prevents them from trying to route back through you if failures happen
- **More concretely:**
 - 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

Poisoned reverse

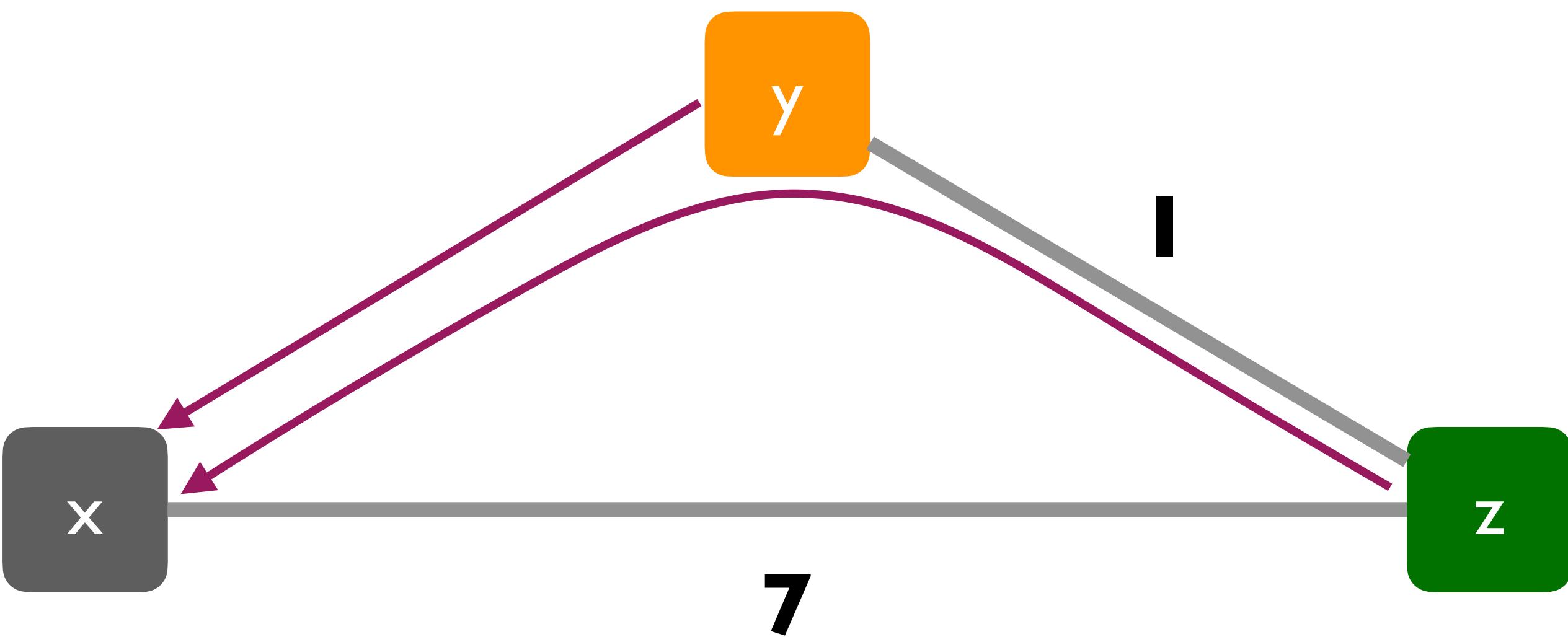
	x	y	z
y	2	0	1
z	∞	1	0



	x	y	z
y	2	0	1
z	3	1	0

Poisoned reverse

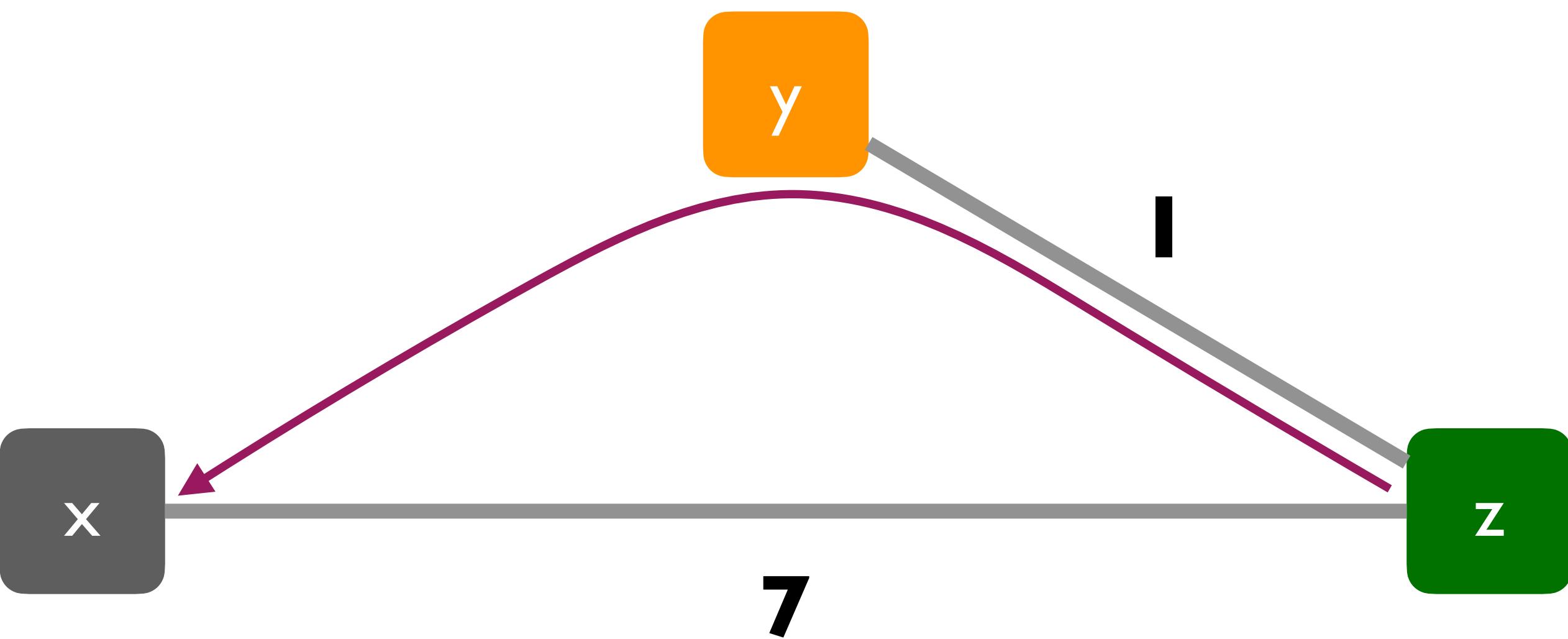
	x	y	z
y	2	0	1
z	∞	1	0



	x	y	z
y	2	0	1
z	3	1	0

Poisoned reverse

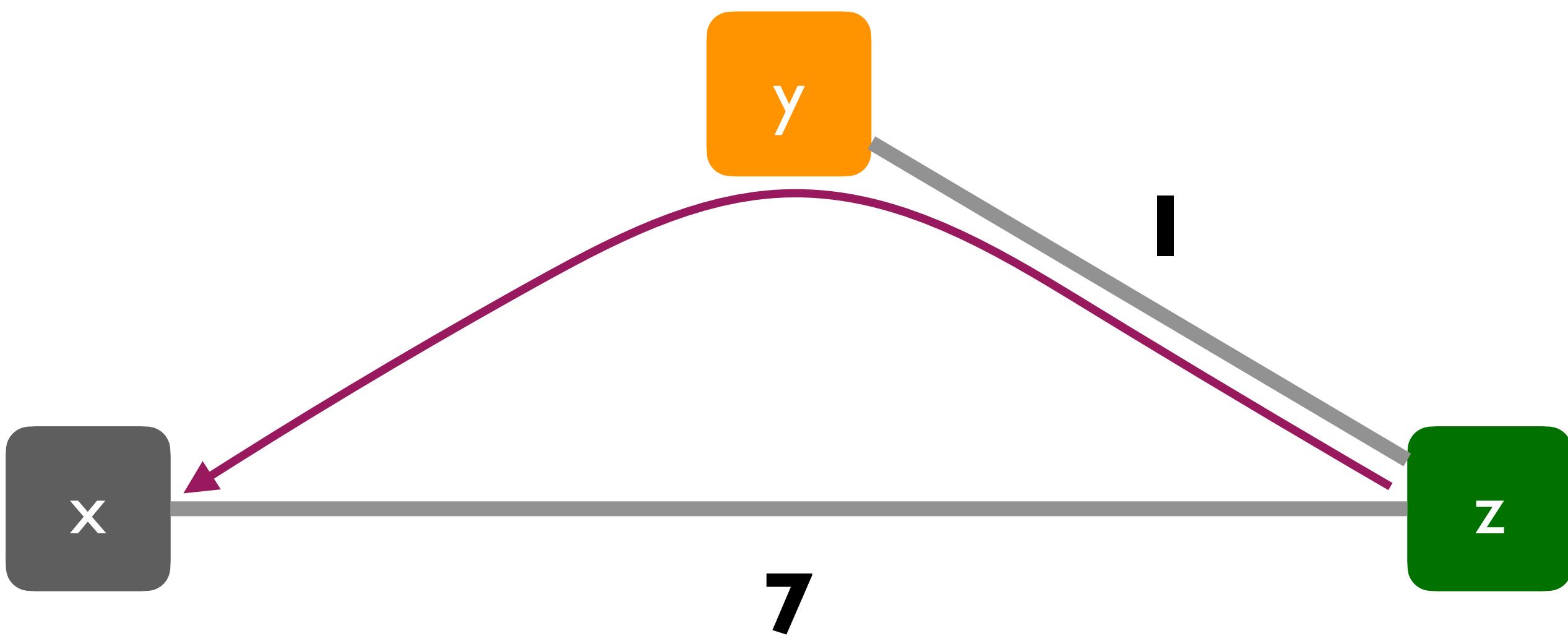
	x	y	z
x			
y	∞	0	1
z	∞	1	0



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

Poisoned reverse

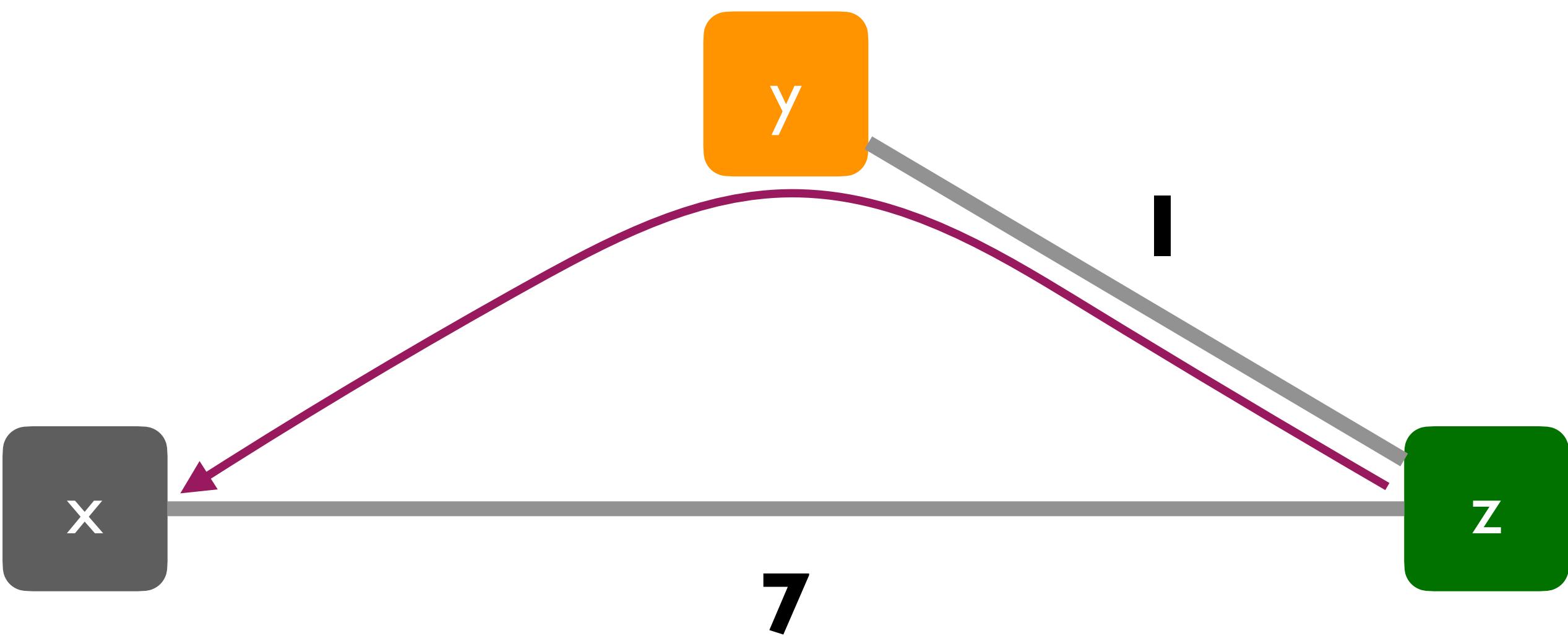
	x	y	z
x			
y	∞	0	1
z	∞	1	0



	x	y	z
x			
y	∞	0	1
z	3	1	0

Poisoned reverse

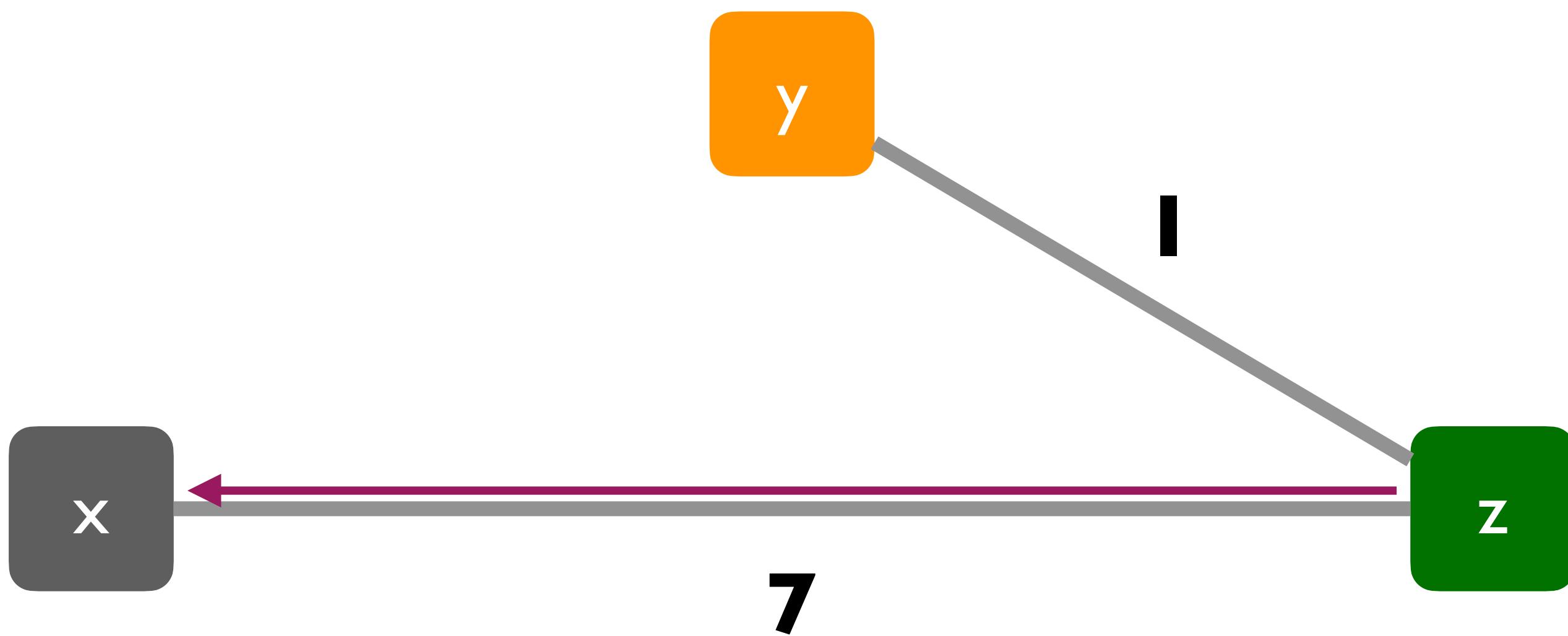
	x	y	z
x			
y	∞	0	1
z	∞	1	0



	x	y	z
x			
y	∞	0	1
z	7	1	0

Poisoned reverse

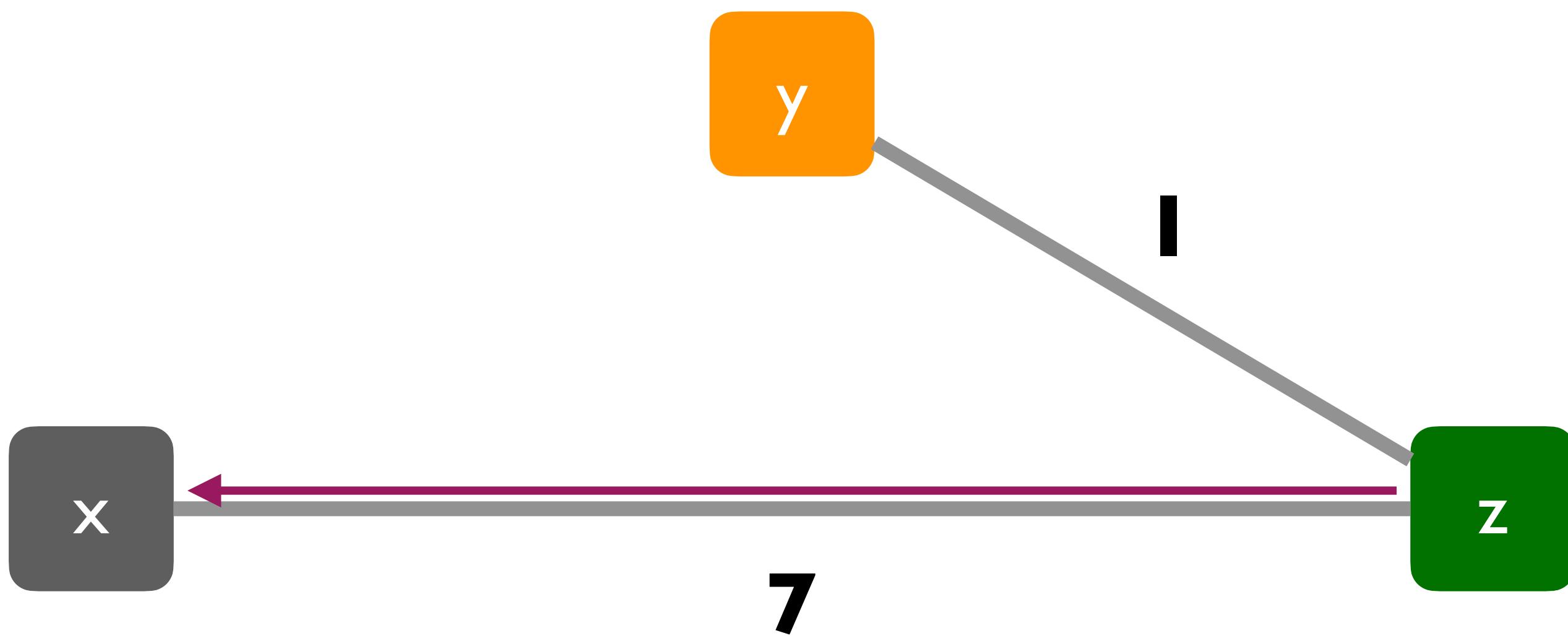
	x	y	z
x			
y	∞	0	1
z	∞	1	0



	x	y	z
x			
y	∞	0	1
z	7	1	0

Poisoned reverse

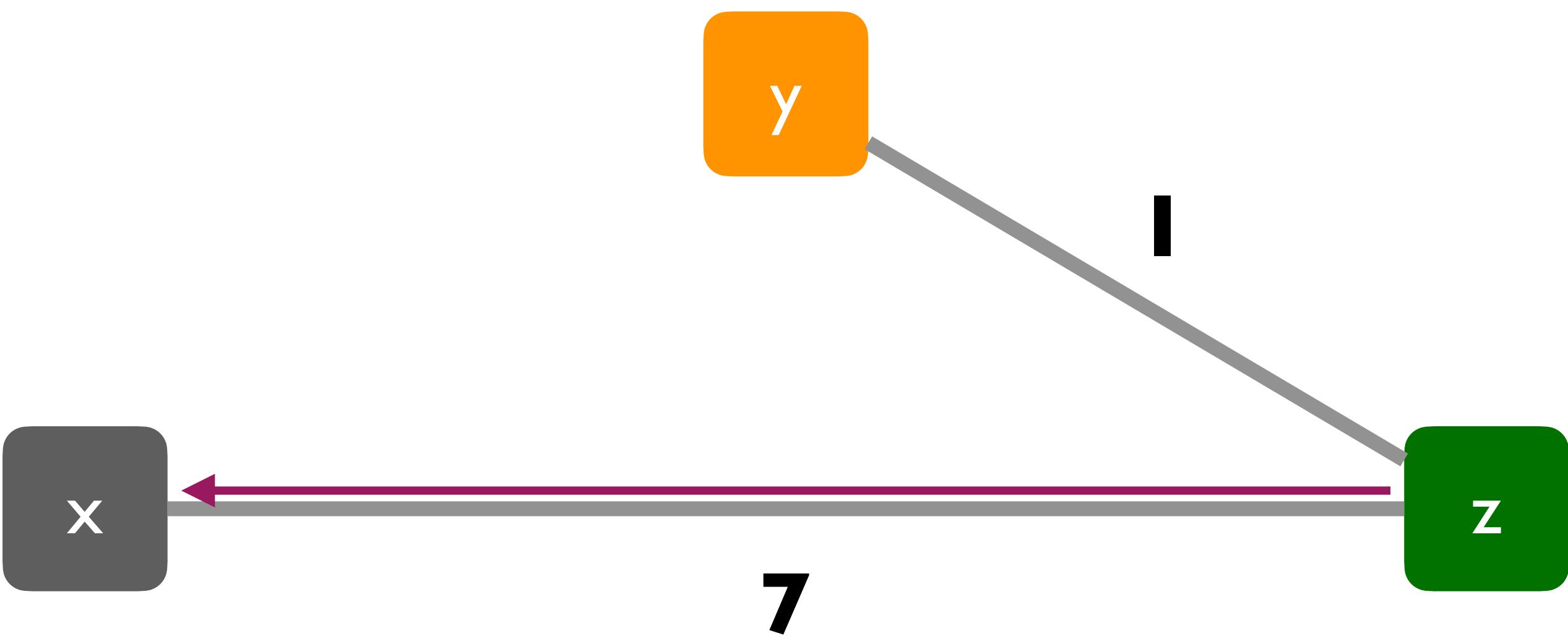
	x	y	z
x			
y	∞	0	1
z	7	1	0



	x	y	z
x			
y	∞	0	1
z	7	1	0

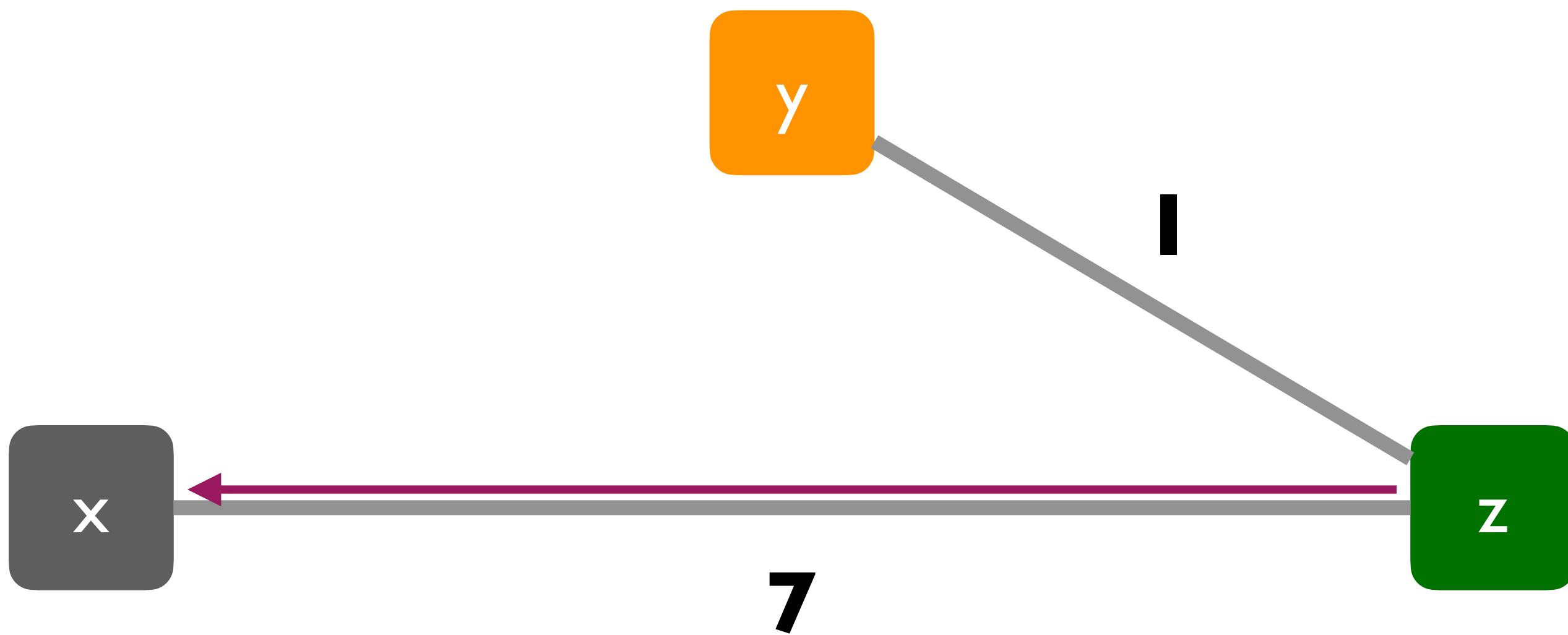
Poisoned reverse

	x	y	z
y	∞	0	1
z	7	1	0



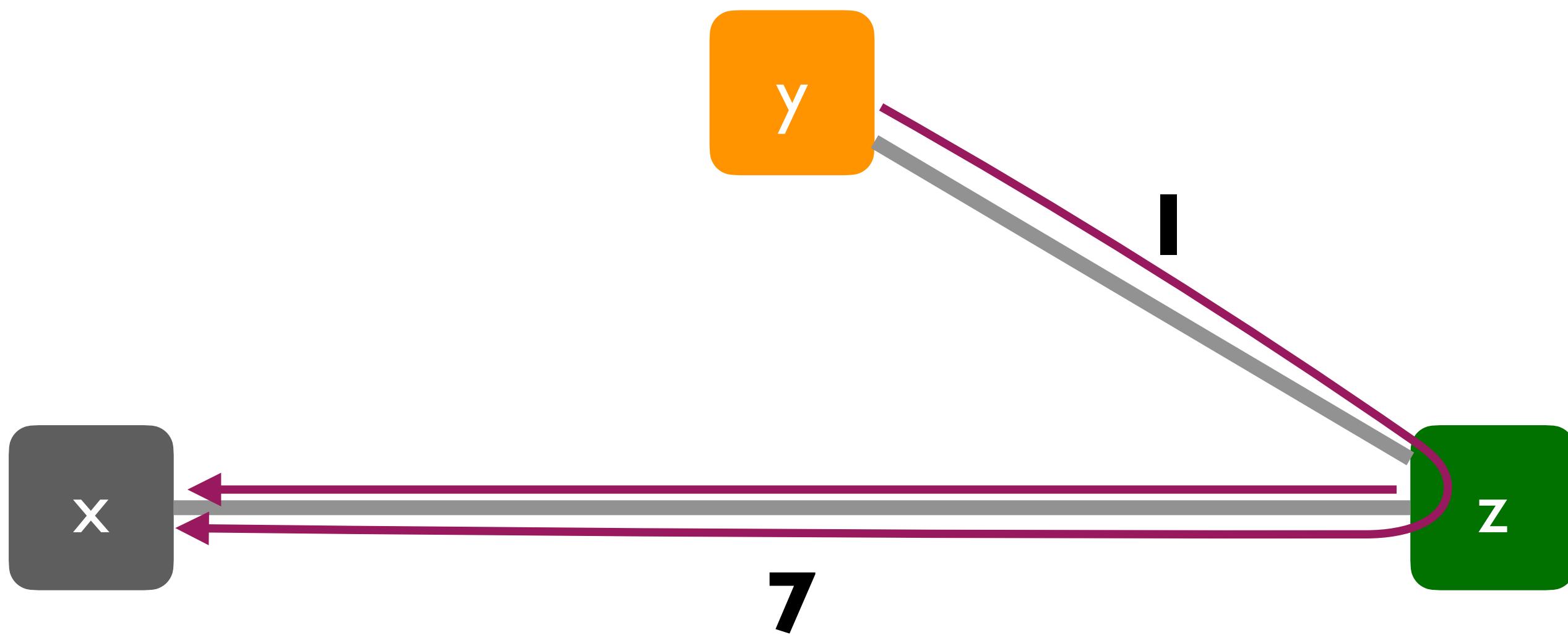
	x	y	z
y	∞	0	1
z	7	1	0

	x	y	z
y	8	0	1
z	7	1	0



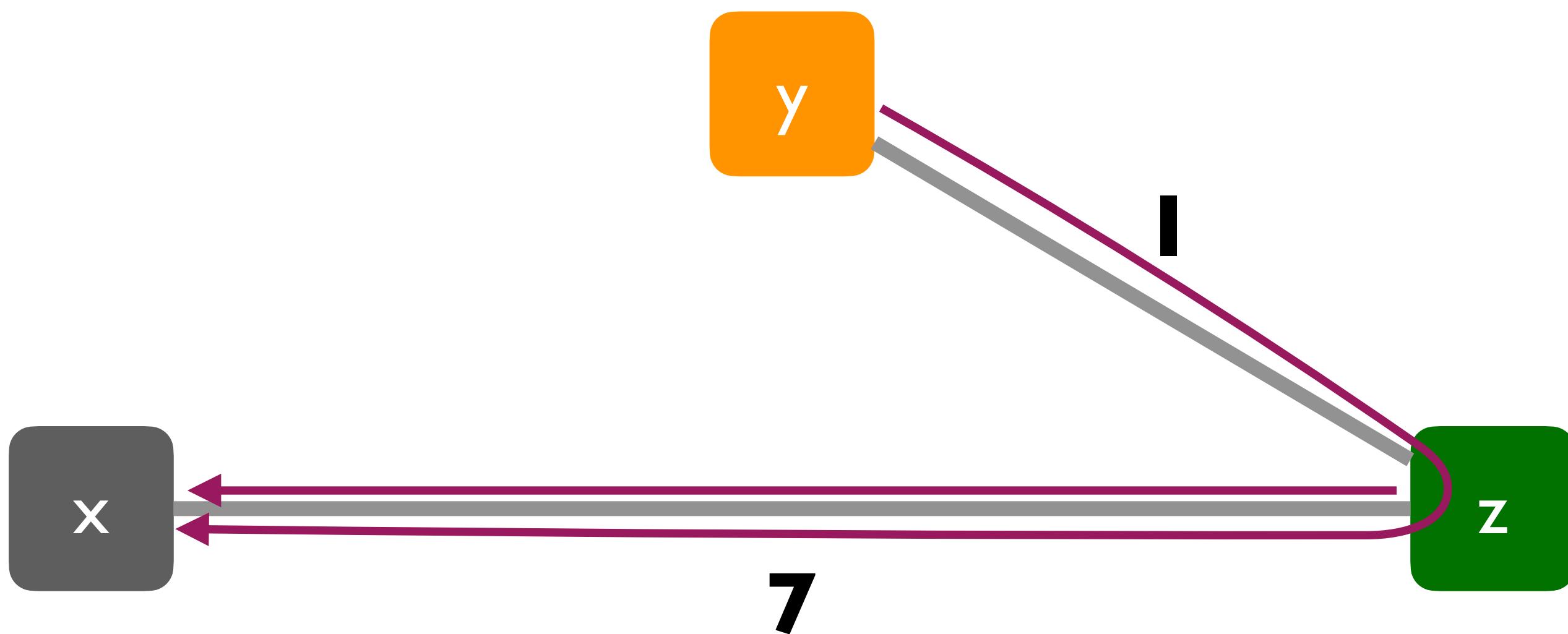
	x	y	z
y	∞	0	1
z	7	1	0

	x	y	z
y	8	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

Does Split-Horizon w/ Poison Reverse Completely Solve the Count-to-Infinity Problem?

Does Split-Horizon w/ Poison Reverse Completely Solve the Count-to-Infinity Problem?

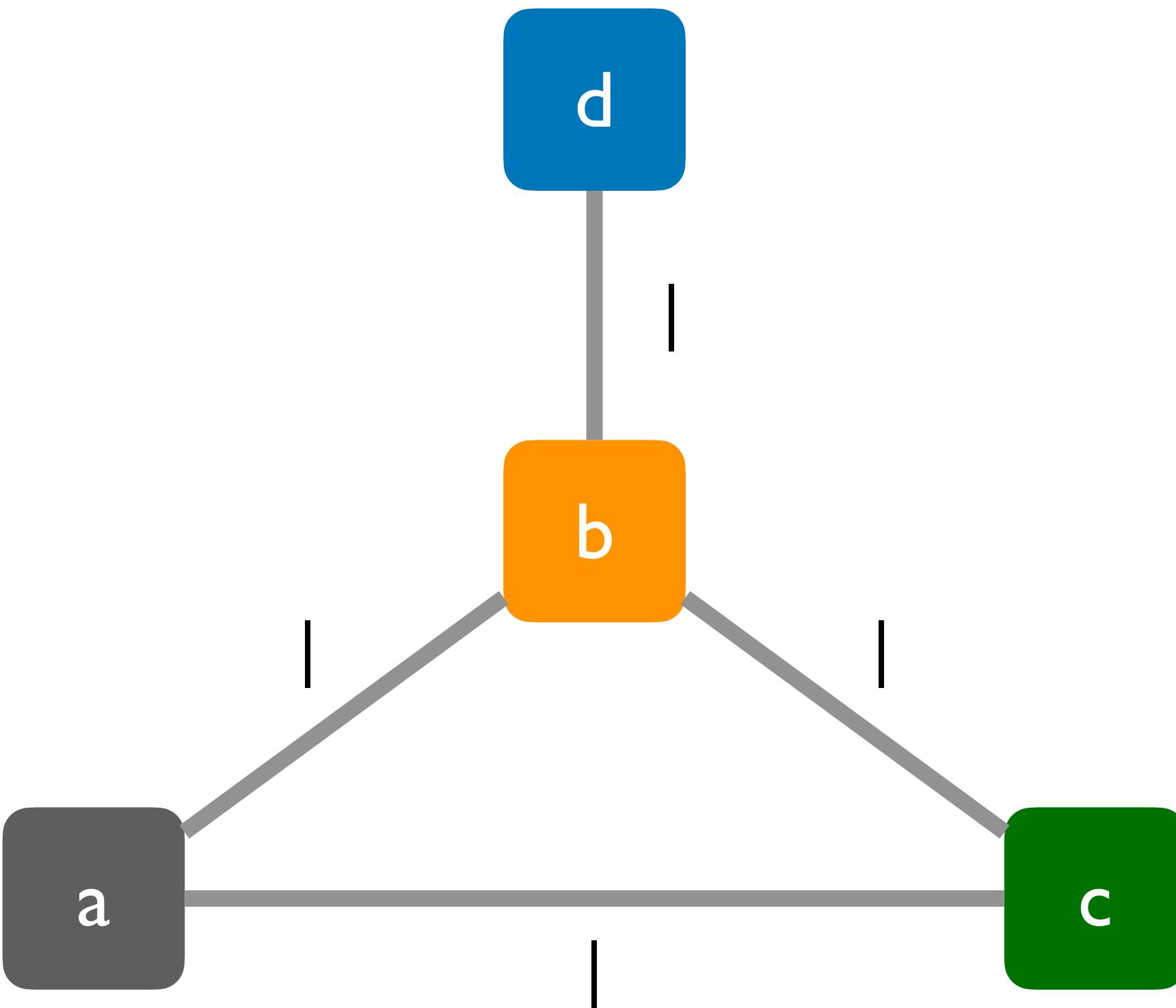
- Not really...

Does Split-Horizon w/ Poison Reverse Completely Solve the Count-to-Infinity Problem?

- Not really...
- Can you think about an example?

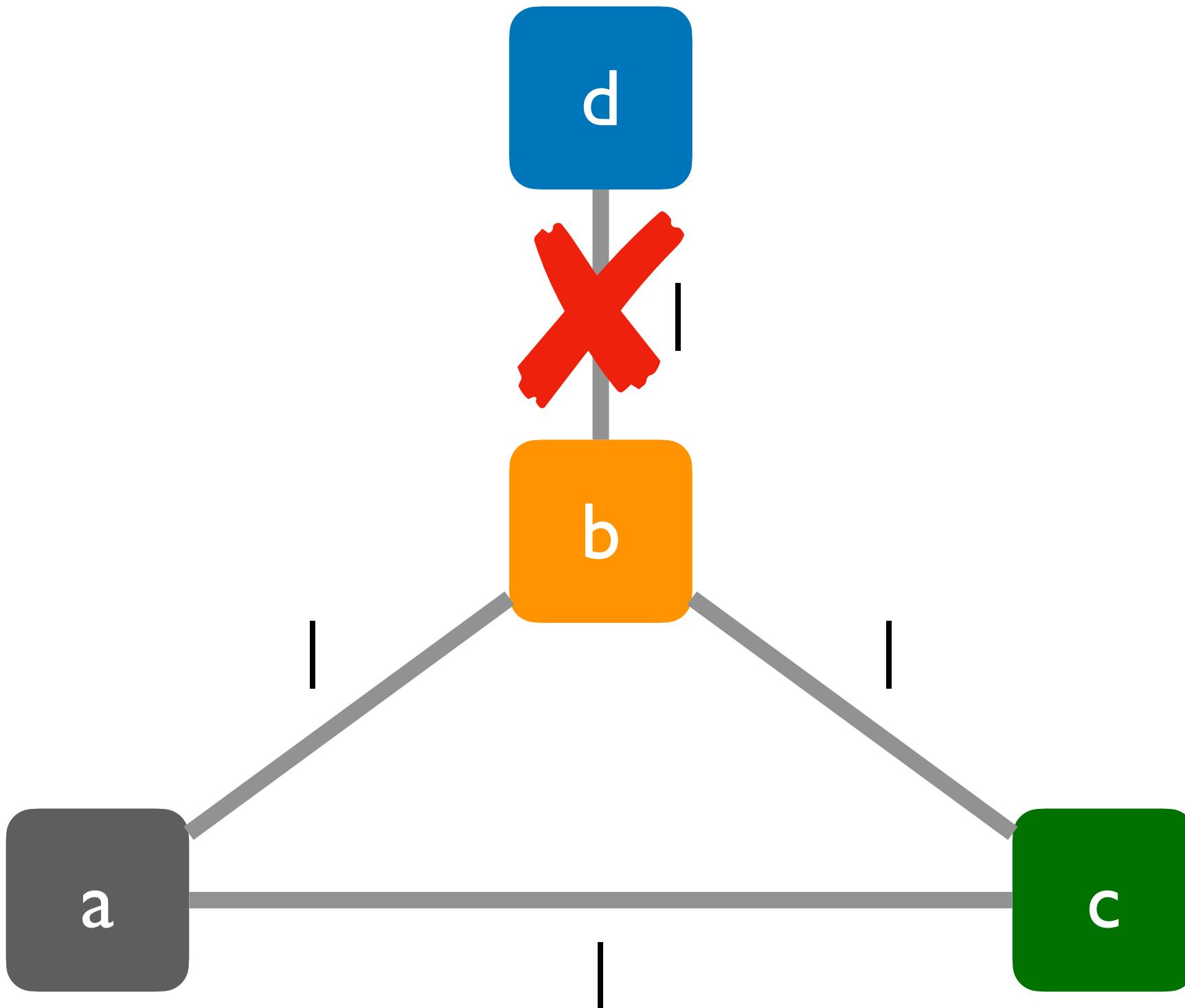
Does Split-Horizon w/ Poison Reverse Completely Solve the Count-to-Infinity Problem?

- Not really...
- Can you think about an example?
 - Hint...



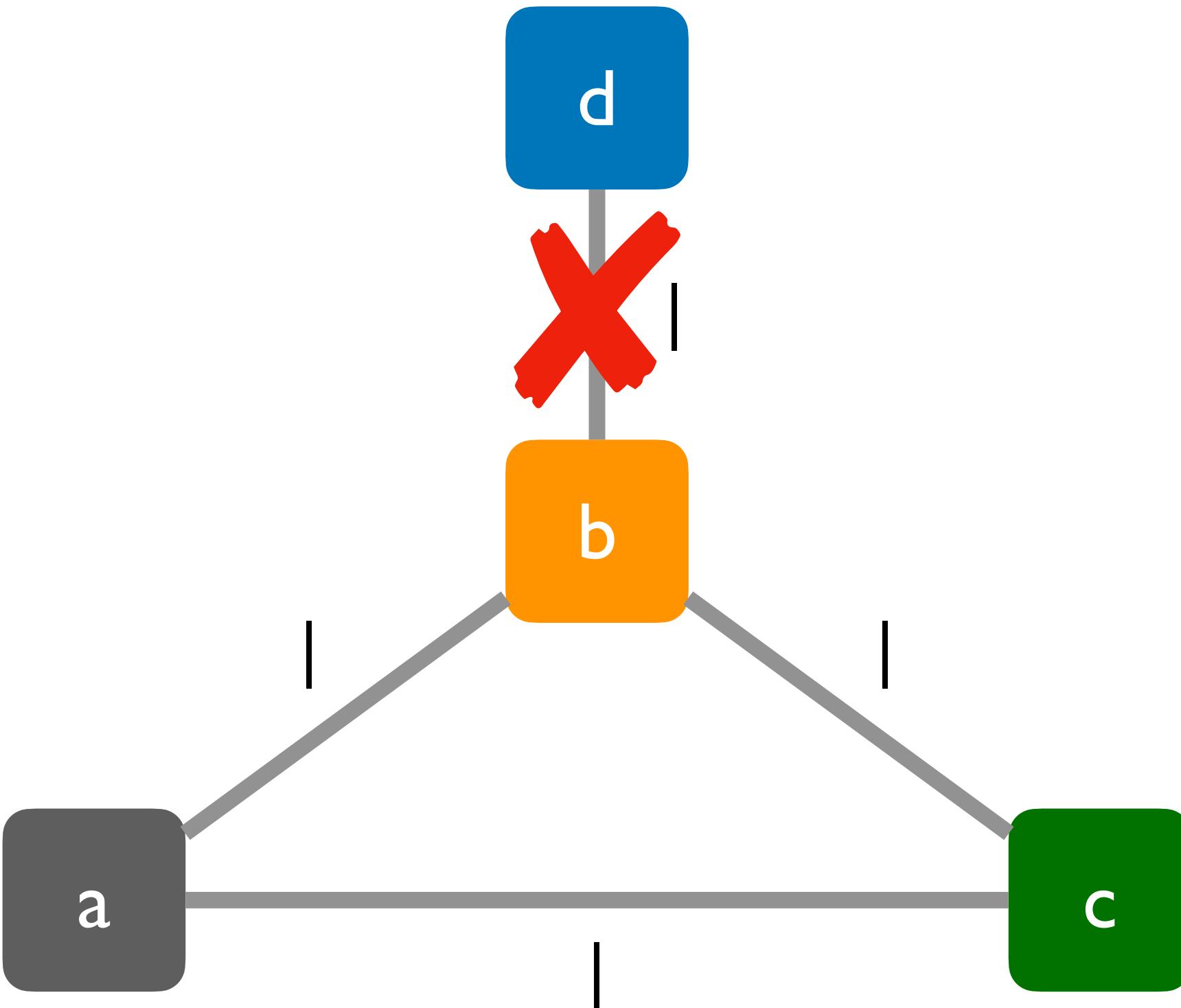
Does Split-Horizon w/ Poison Reverse Completely Solve the Count-to-Infinity Problem?

- Not really...
- Can you think about an example?
 - Hint...



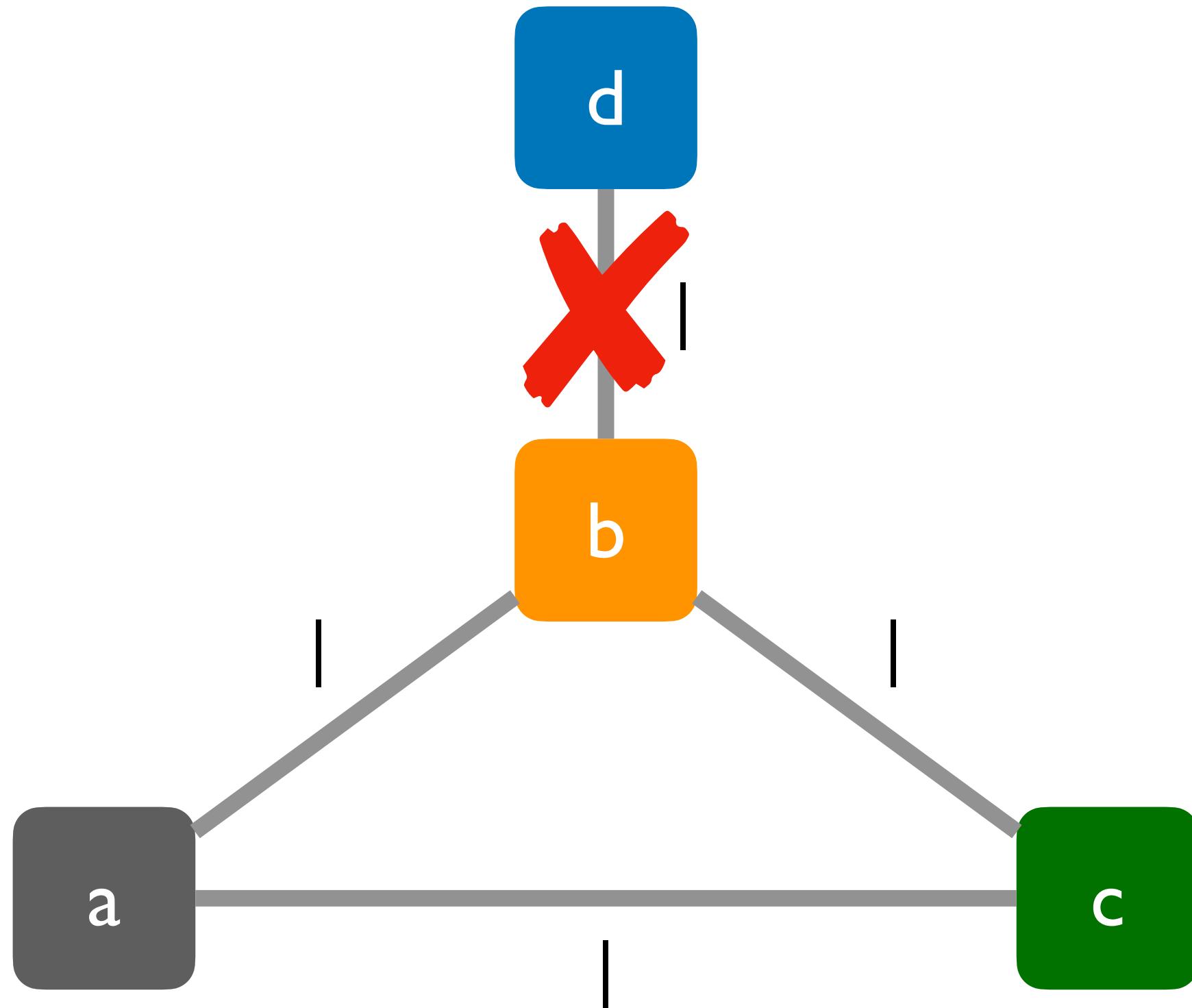
Does Split-Horizon w/ Poison Reverse Completely Solve the Count-to-Infinity Problem?

- Not really...
- Can you think about an example?
 - Hint...
- Work through it!



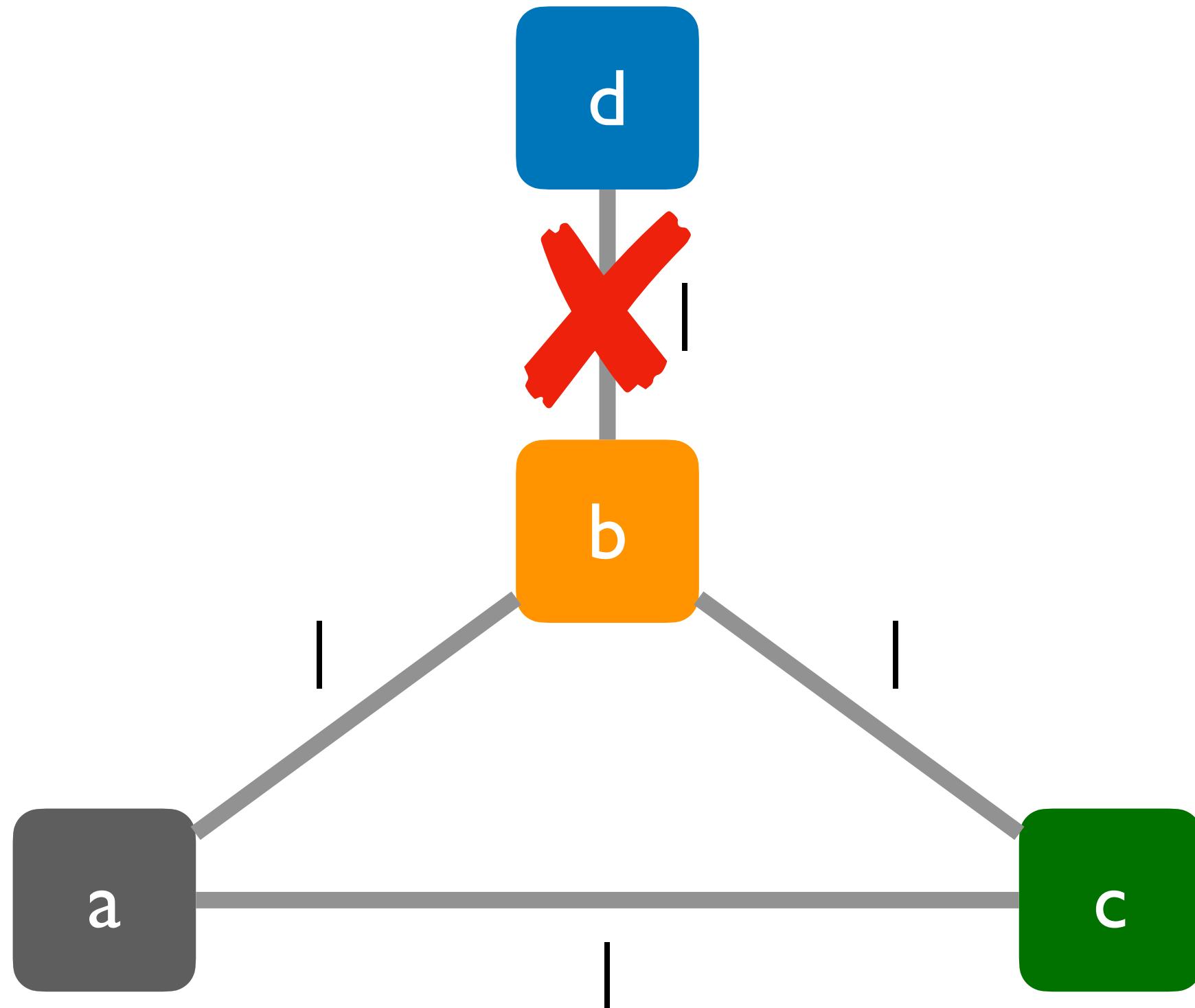
Does Split-Horizon w/ Poison Reverse Completely Solve the Count-to-Infinity Problem?

- Not really...
- Can you think about an example?
 - Hint...
- Work through it!
 - Post on Piazza if you can crack it



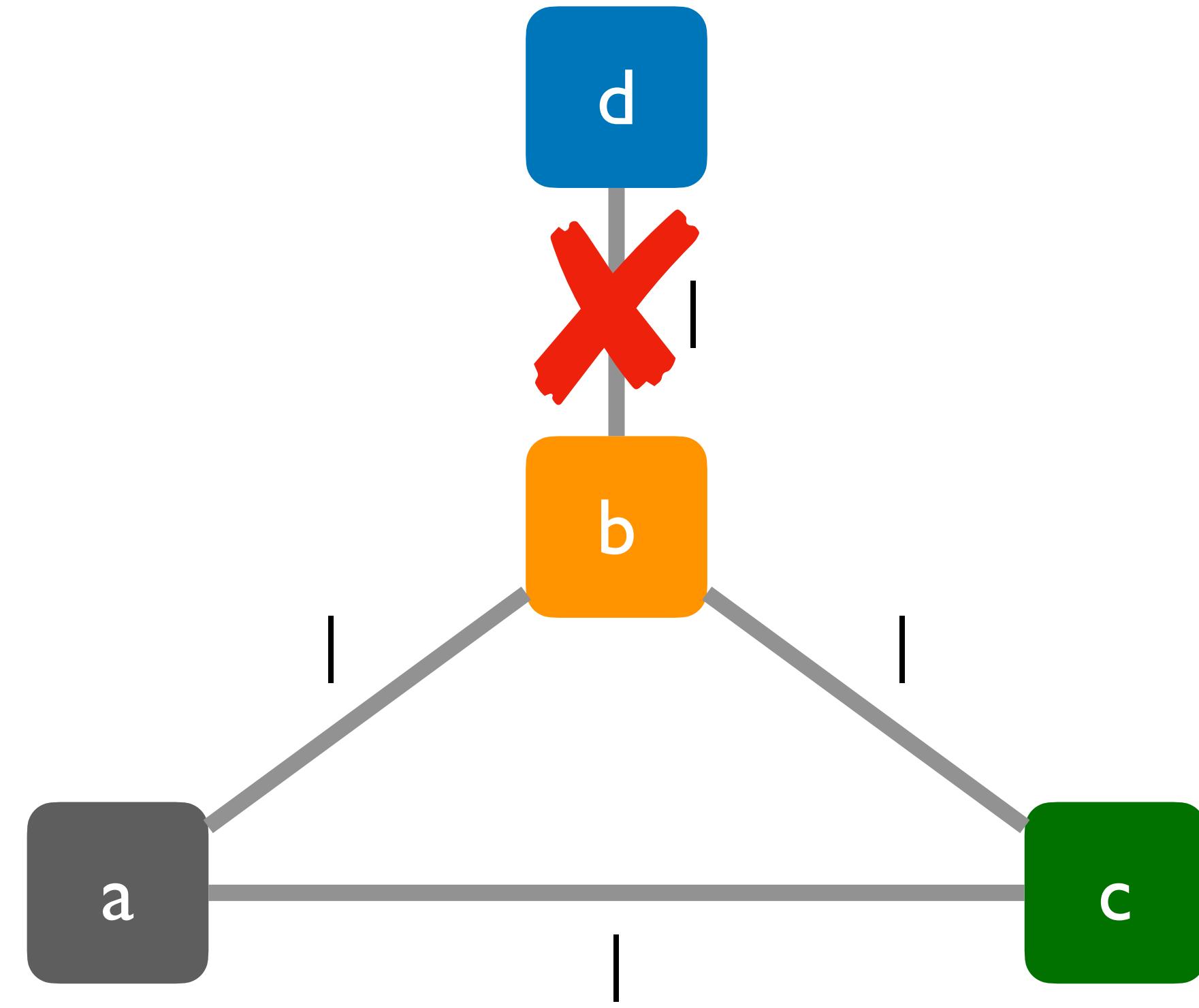
Does Split-Horizon w/ Poison Reverse Completely Solve the Count-to-Infinity Problem?

- Not really...
- Can you think about an example?
 - Hint...
- Work through it!
 - Post on Piazza if you can crack it
 - Win the adoration (or scorn) of your classmates



Does Split-Horizon w/ Poison Reverse Completely Solve the Count-to-Infinity Problem?

- Not really...
- Can you think about an example?
 - Hint...
- Work through it!
 - Post on Piazza if you can crack it
 - Win the adoration (or scorn) of your classmates
 - Maybe earn some participation points ;)



Distance Vector Routing

Distance Vector Routing

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

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
 - $O(\text{network diameter})$ convergence time
 - $O(N)$ entries in forwarding table

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
 - $O(\text{network diameter})$ convergence time
 - $O(N)$ entries in forwarding table
- RIP is a protocol that implements DV (IETF RFC 2080)

Link-State & Distance-Vector

- Read text to make sure you really understand them!
 - Understand the mapping between Dijkstra's <-> LS
 - Understand the mapping between Bellman Ford <-> DV
- Figure out when poison reverse fails!

Questions?

Routing in the Internet

Routing in the Internet

- The routing protocols we have discussed so far (LS, DV):

Routing in the Internet

- The routing protocols we have discussed so far (LS, DV):
 - How routing happens within a **domain**

Routing in the Internet

- The routing protocols we have discussed so far (LS, DV):
 - How routing happens within a **domain**
 - *What is a domain?*

Routing in the Internet

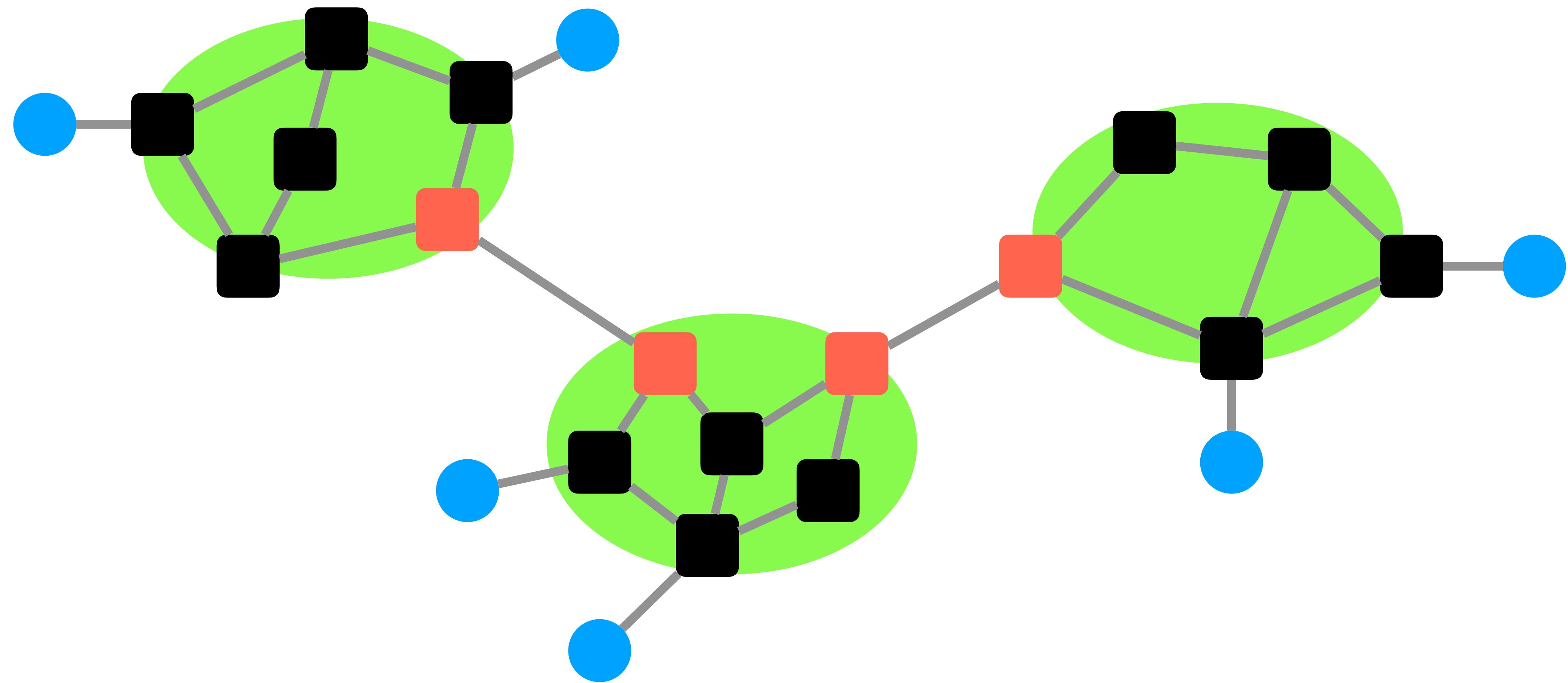
- The routing protocols we have discussed so far (LS, DV):
 - How routing happens within a **domain**
 - *What is a domain?*
- **Domain:** Network under single administrative authority

Routing in the Internet

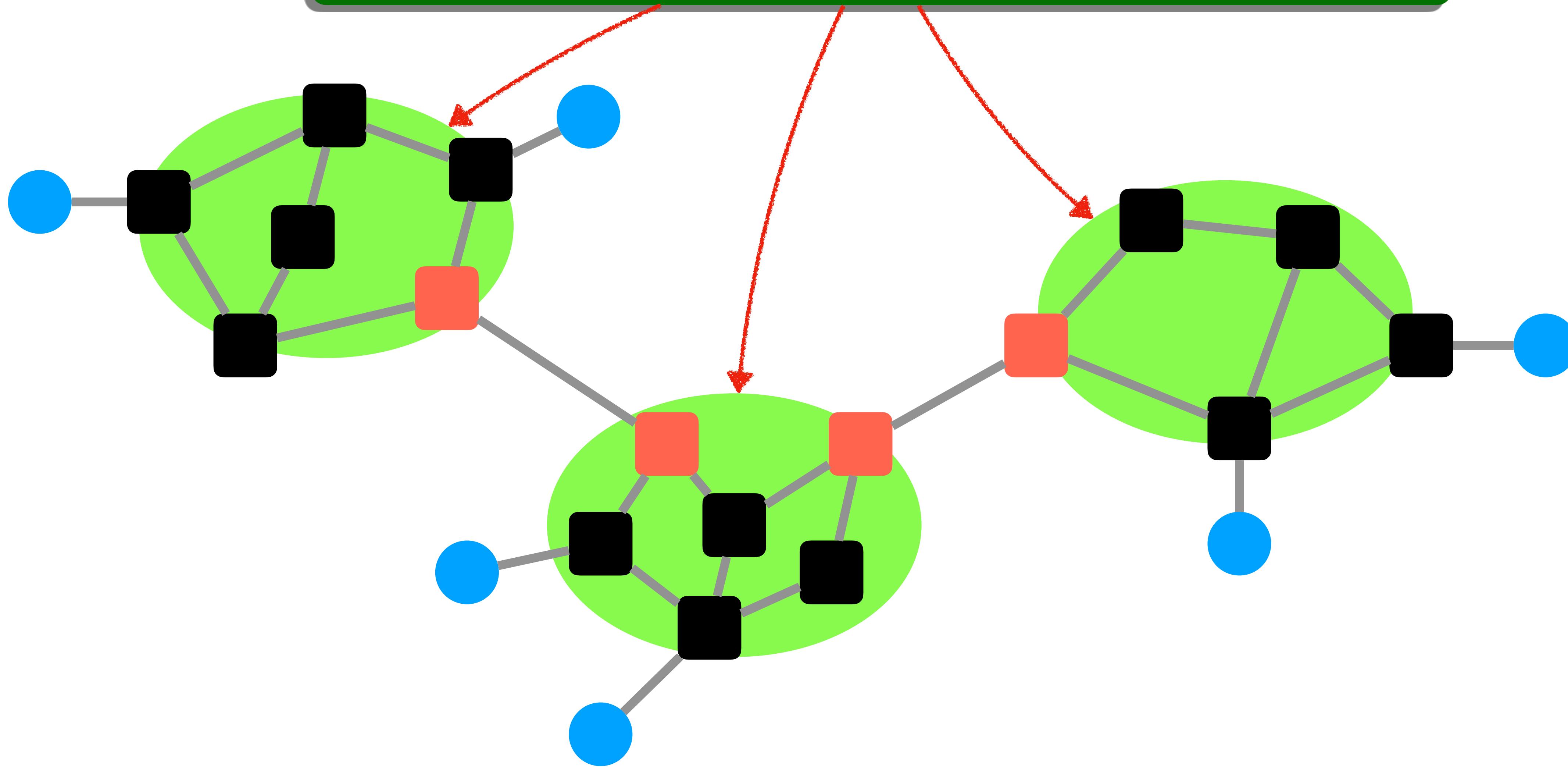
- The routing protocols we have discussed so far (LS, DV):
 - How routing happens within a **domain**
 - *What is a domain?*
- **Domain:** Network under single administrative authority
- Many issues can be ignored in this setting because there is central administrative control over routers

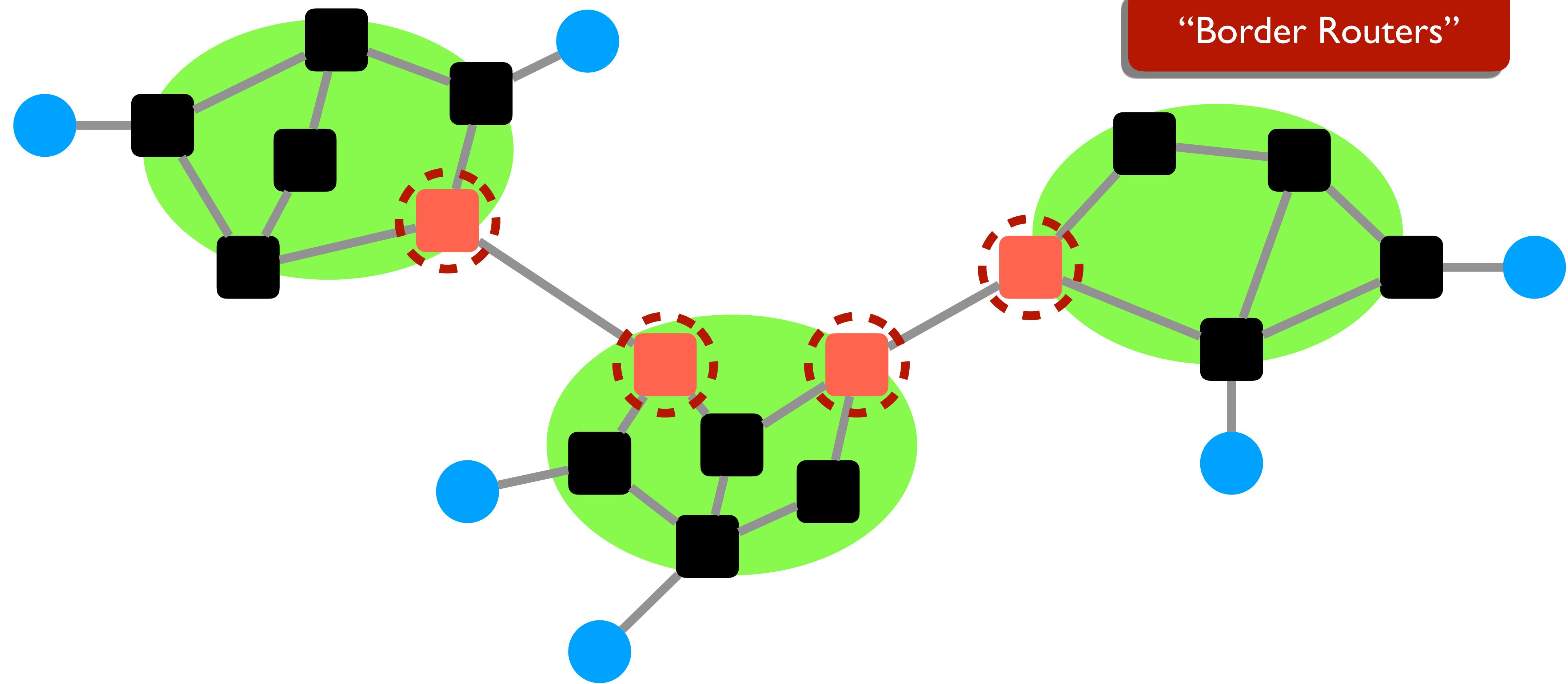
Routing in the Internet

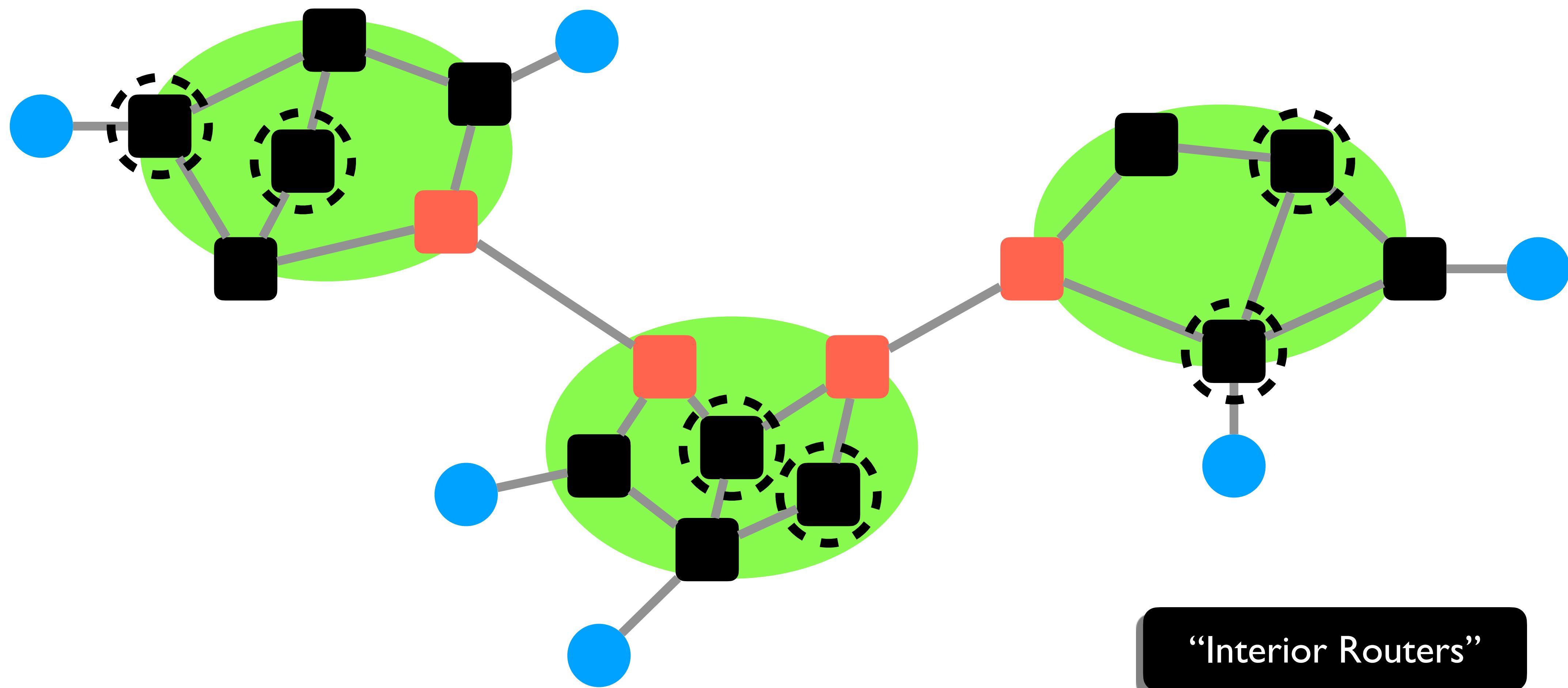
- The routing protocols we have discussed so far (LS, DV):
 - How routing happens within a **domain**
 - *What is a domain?*
- **Domain:** *Network under single administrative authority*
- Many issues can be ignored in this setting because there is central administrative control over routers
 - Issues such as *autonomy, privacy, policy*

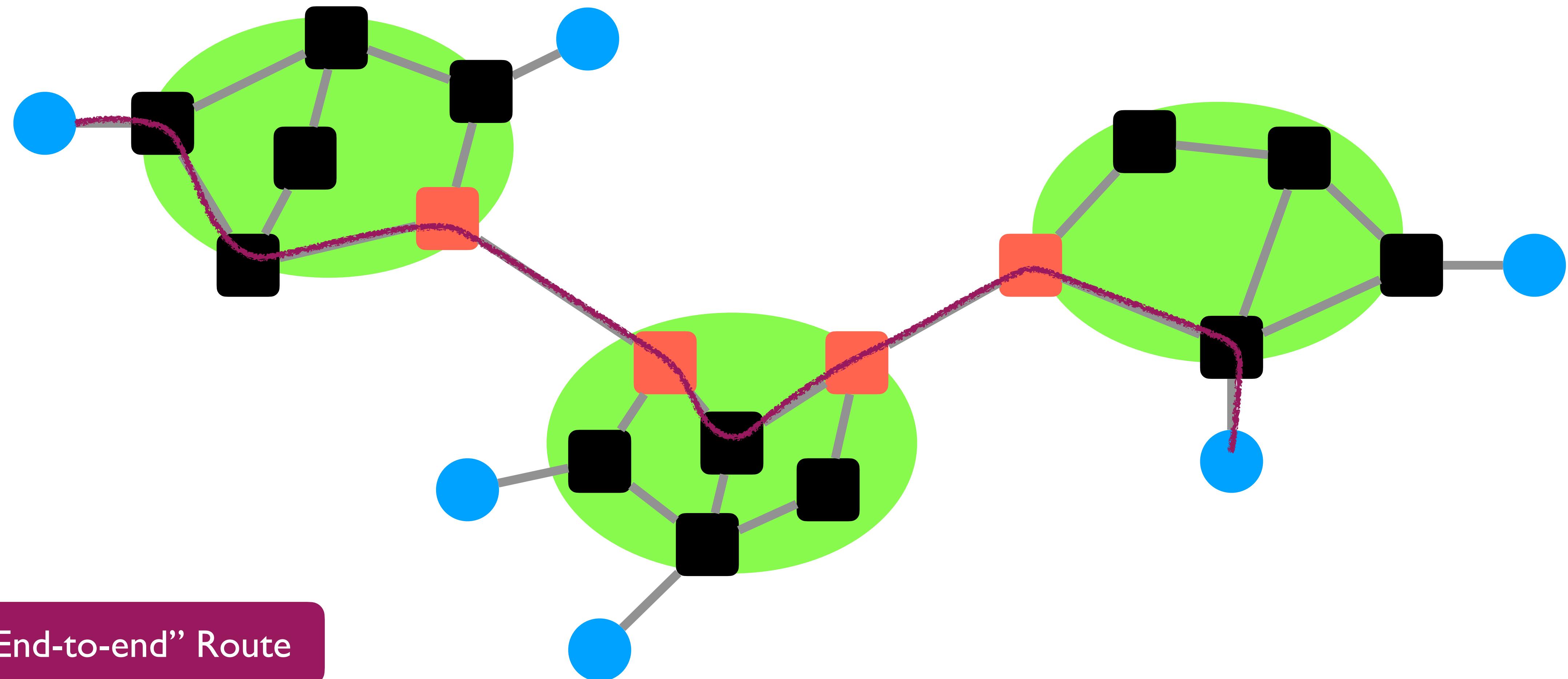


“Autonomous System (AS)” or “Domain”
Region of a network under a single administrative authority









Autonomous Systems (AS)

Autonomous Systems (AS)

- AS is a **network under a single administrative control**
 - Currently over 70,000 ASes
 - Think AT&T, France Telecom, Yale, IBM, etc.

Autonomous Systems (AS)

- AS is a **network under a single administrative control**
 - Currently over 70,000 ASes
 - Think AT&T, France Telecom, Yale, IBM, etc.
- ASes are sometimes called “domains”

Autonomous Systems (AS)

- AS is a **network under a single administrative control**
 - Currently over 70,000 ASes
 - Think AT&T, France Telecom, Yale, IBM, etc.
- ASes are sometimes called “domains”
- Each AS is assigned **a unique identifier**
 - 16 bit **AS Number (ASN)**
 - E.g., ASN 29 is Yale

“Intradomain” Routing: Within an AS

“Intradomain” Routing: Within an AS

- Link-State (**OSPF**) and Distance Vector (**RIP, IGRP**)

“Intradomain” Routing: Within an AS

- Link-State (**OSPF**) and Distance Vector (**RIP, IGRP**)
- **Focus**
 - “Least-cost” paths
 - Convergence

“Interdomain” Routing: Between ASes

“Interdomain” Routing: Between ASes

- **Two key challenges:**

“Interdomain” Routing: Between ASes

- **Two key challenges:**
 - Scaling

“Interdomain” Routing: Between ASes

- **Two key challenges:**
 - Scaling
 - Administrative structure:

“Interdomain” Routing: Between ASes

- **Two key challenges:**
 - Scaling
 - Administrative structure:
 - Issues of autonomy, policy, privacy

“Interdomain” Routing: Between ASes

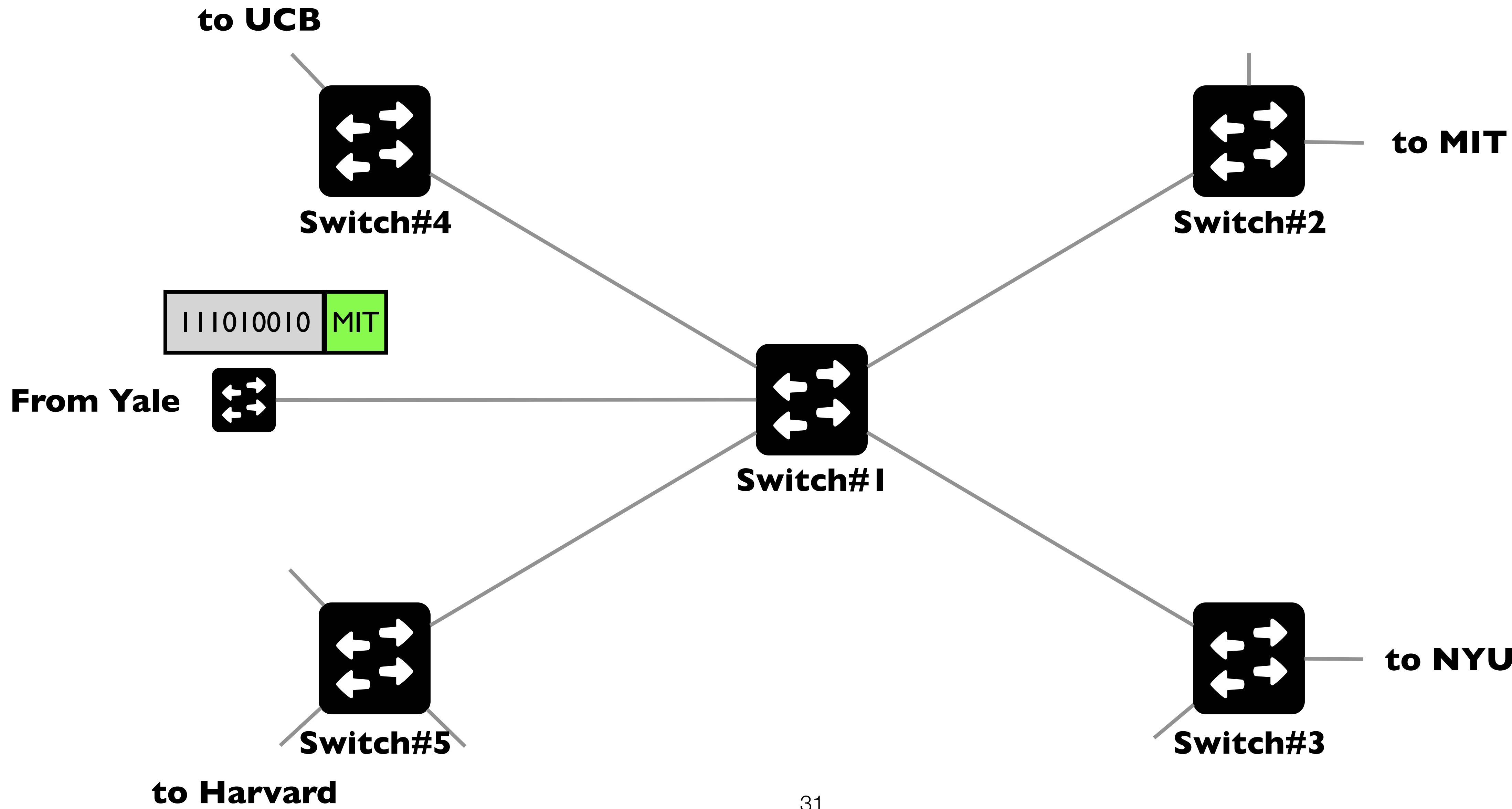
- **Two key challenges:**
 - Scaling
 - Administrative structure:
 - Issues of autonomy, policy, privacy

Recall from Lecture 4

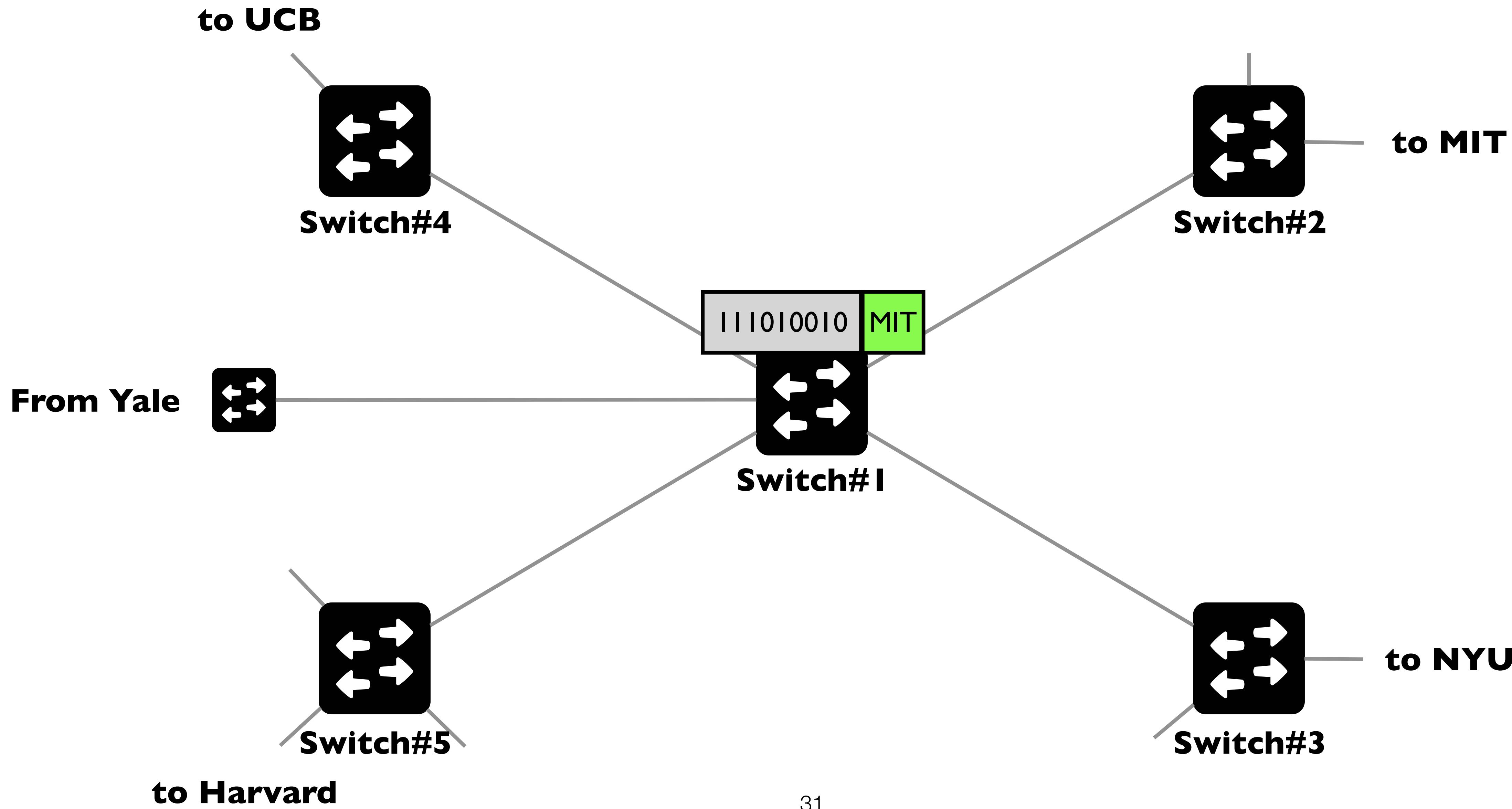
Recall from Lecture 4

- **Assume each host has a unique ID**
- **No particular structure to those IDs**

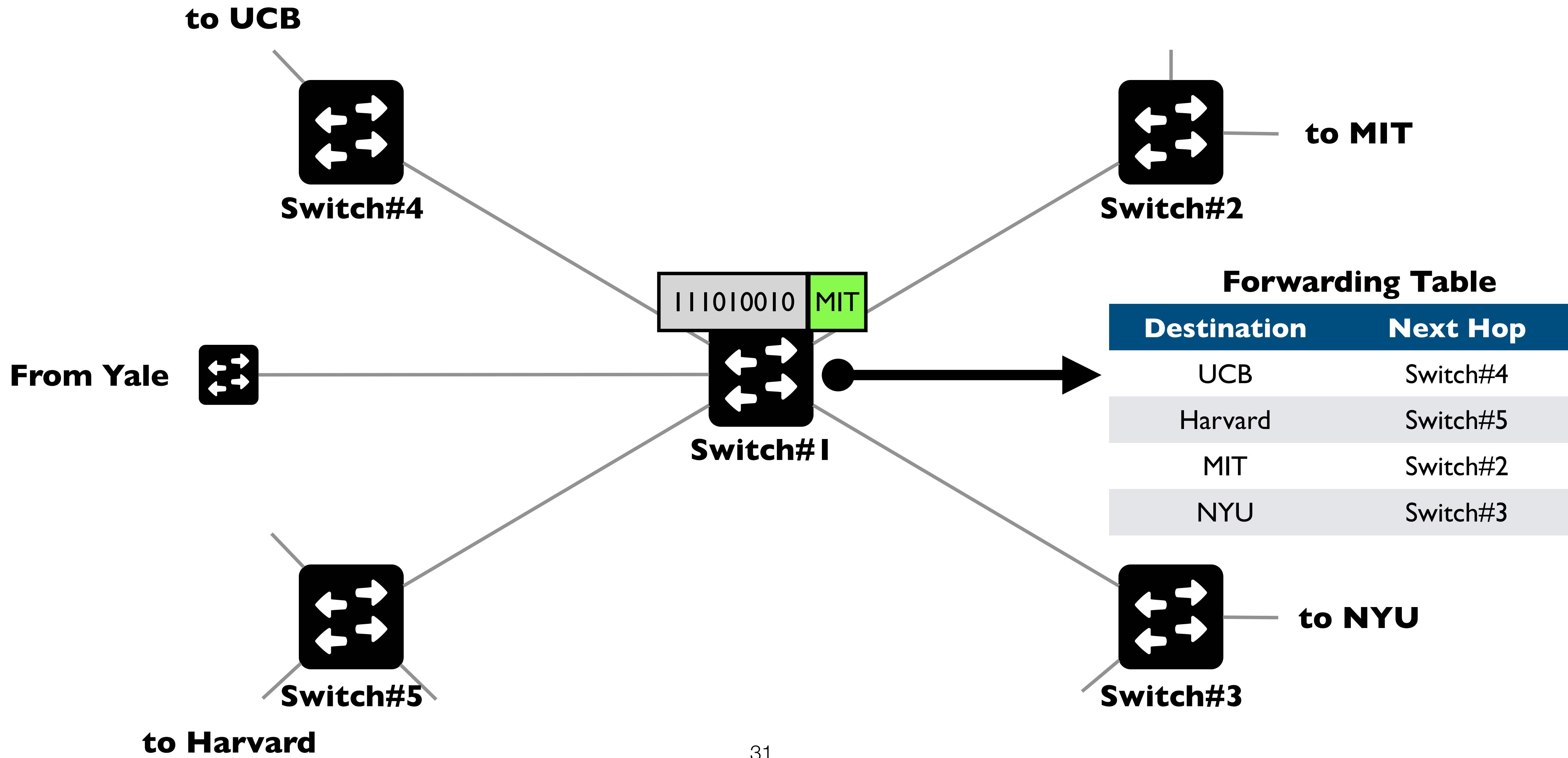
Recall Also...



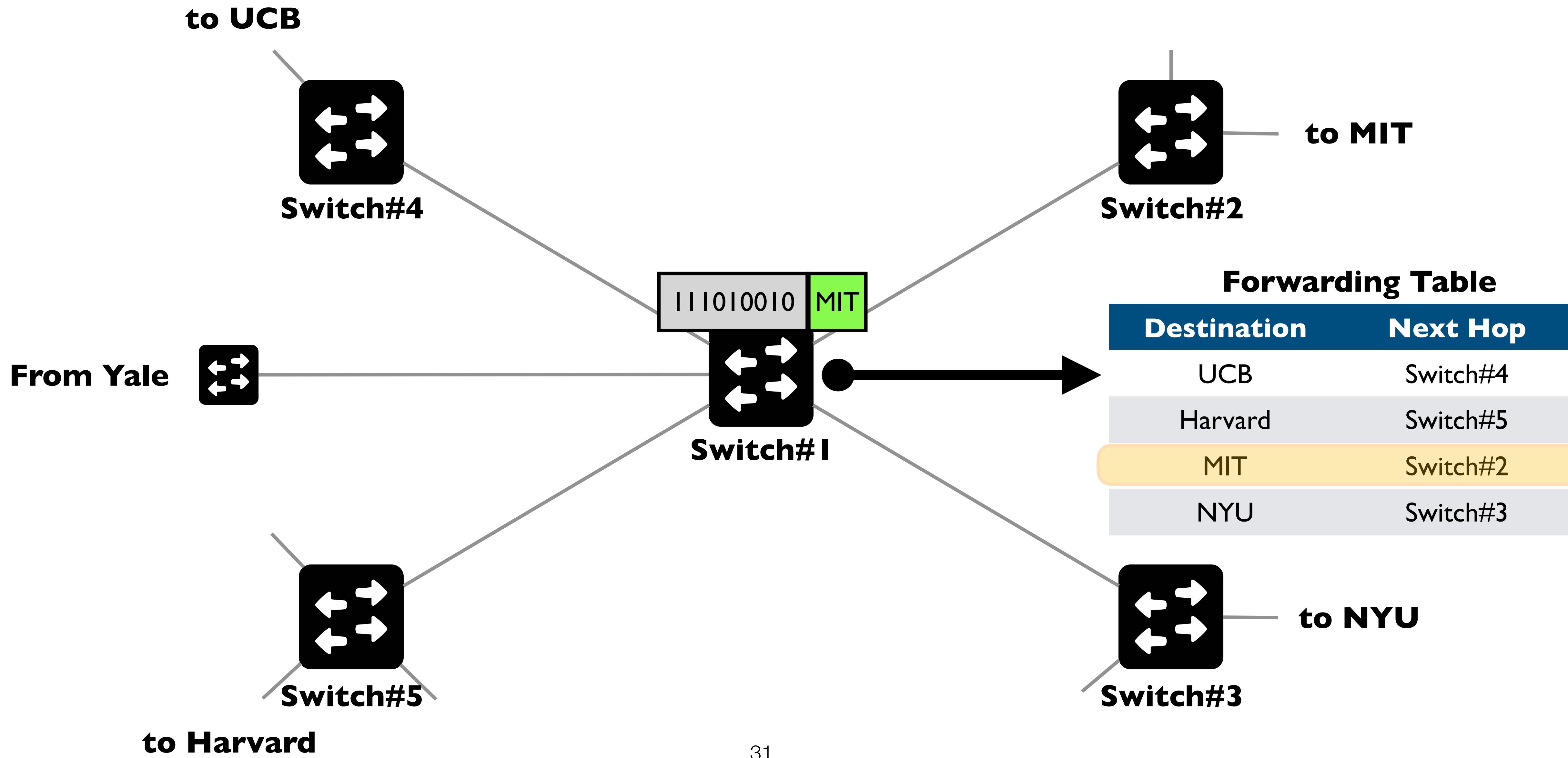
Recall Also...



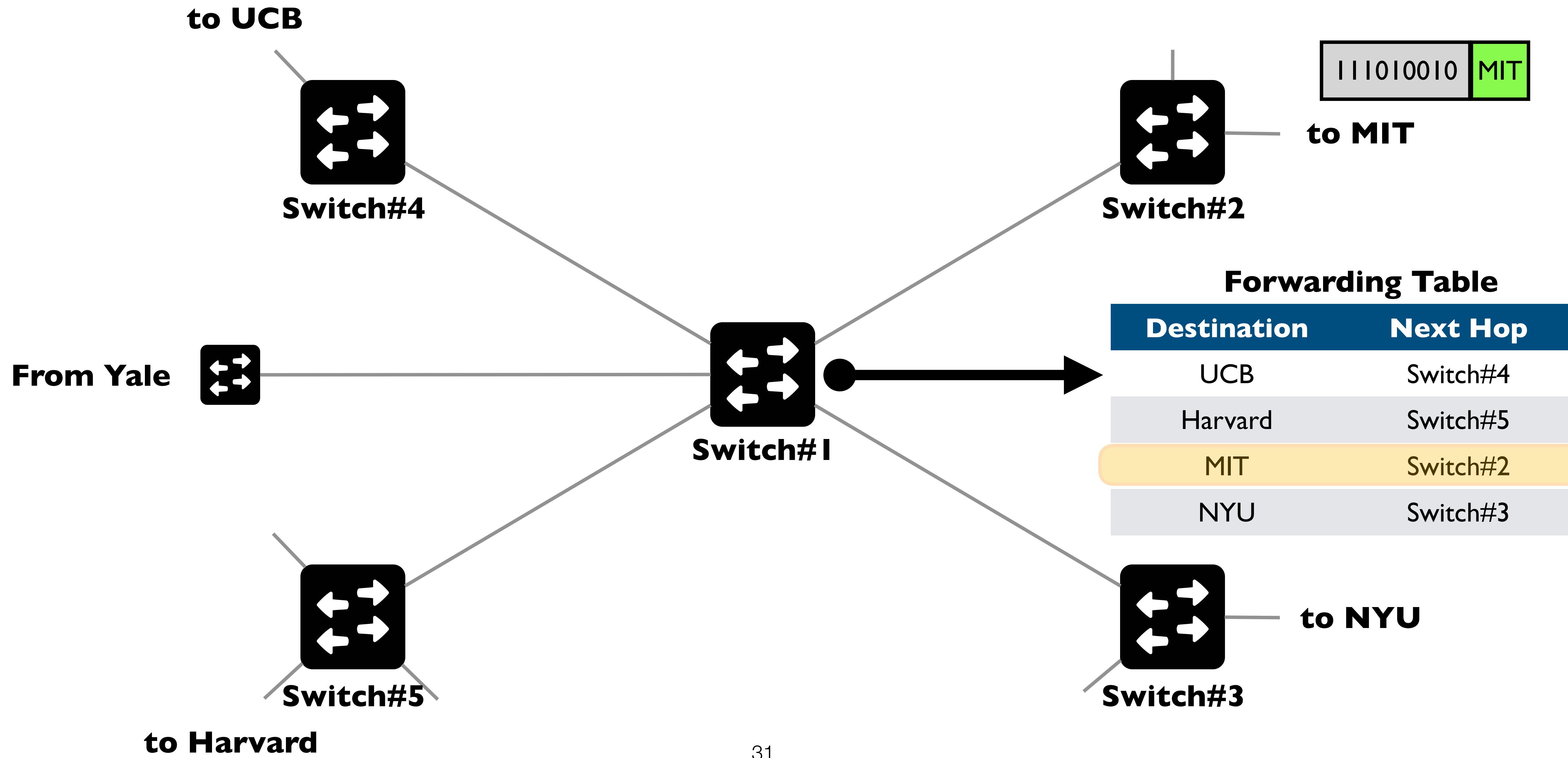
Recall Also...



Recall Also...



Recall Also...



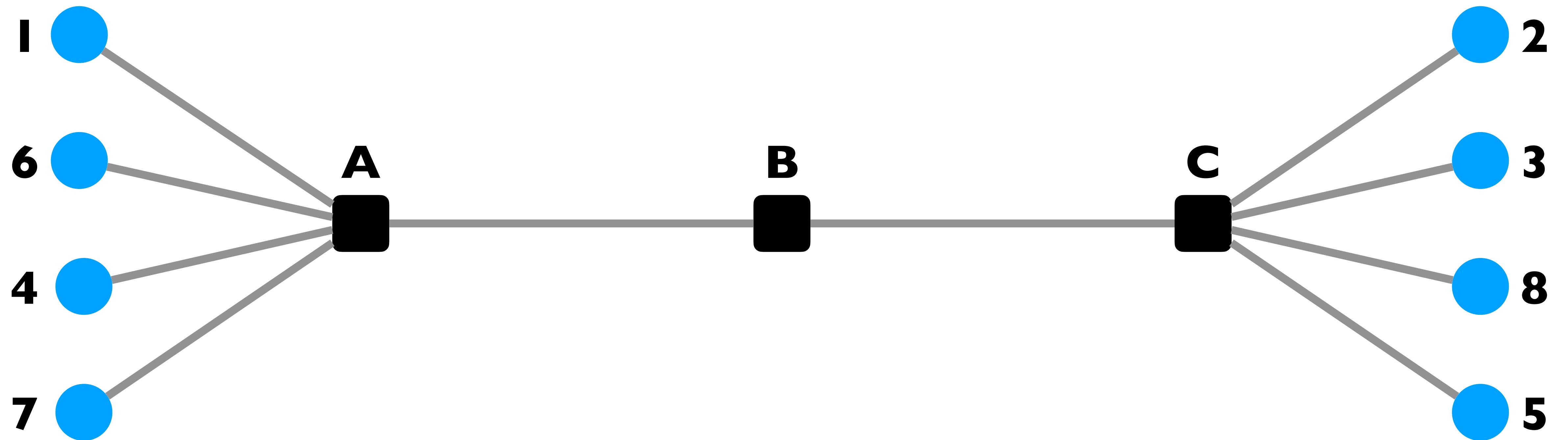
Scaling

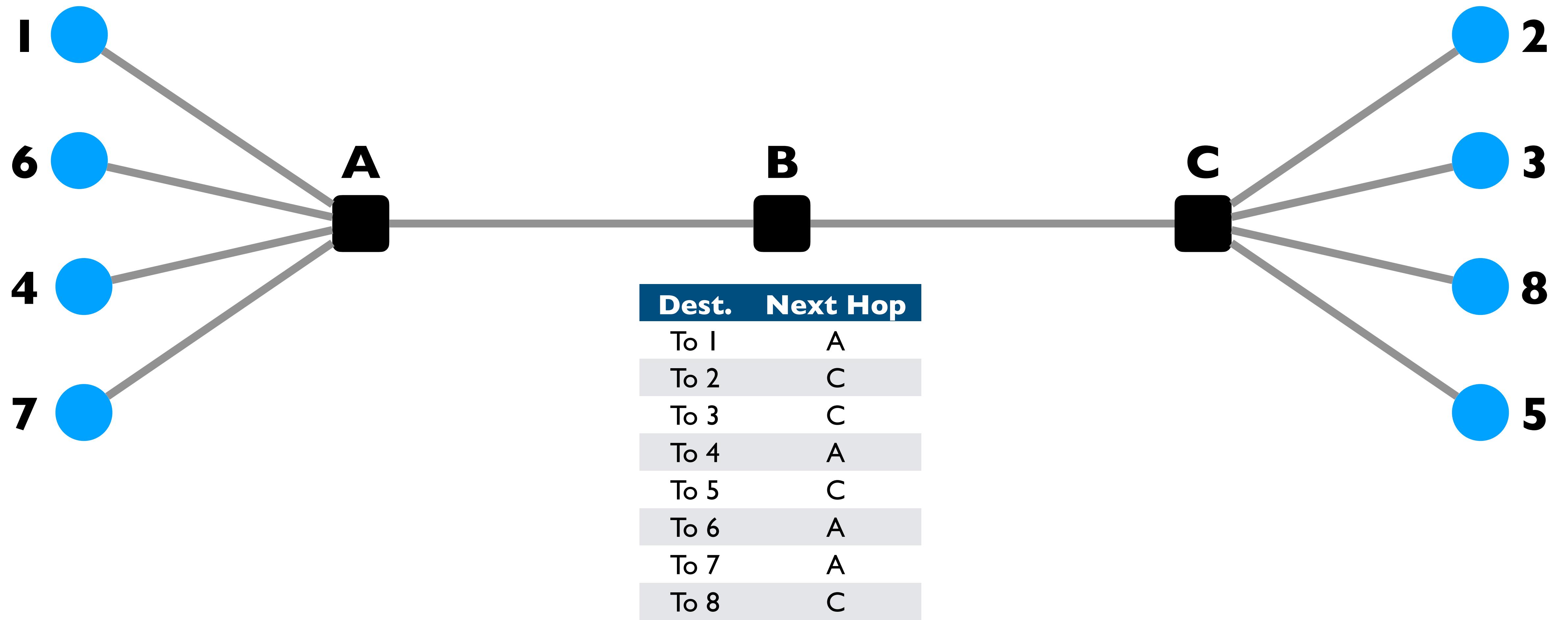
Scaling

- A router must be able to reach *any* destination
 - Given packet's destination address, lookup “next hop”

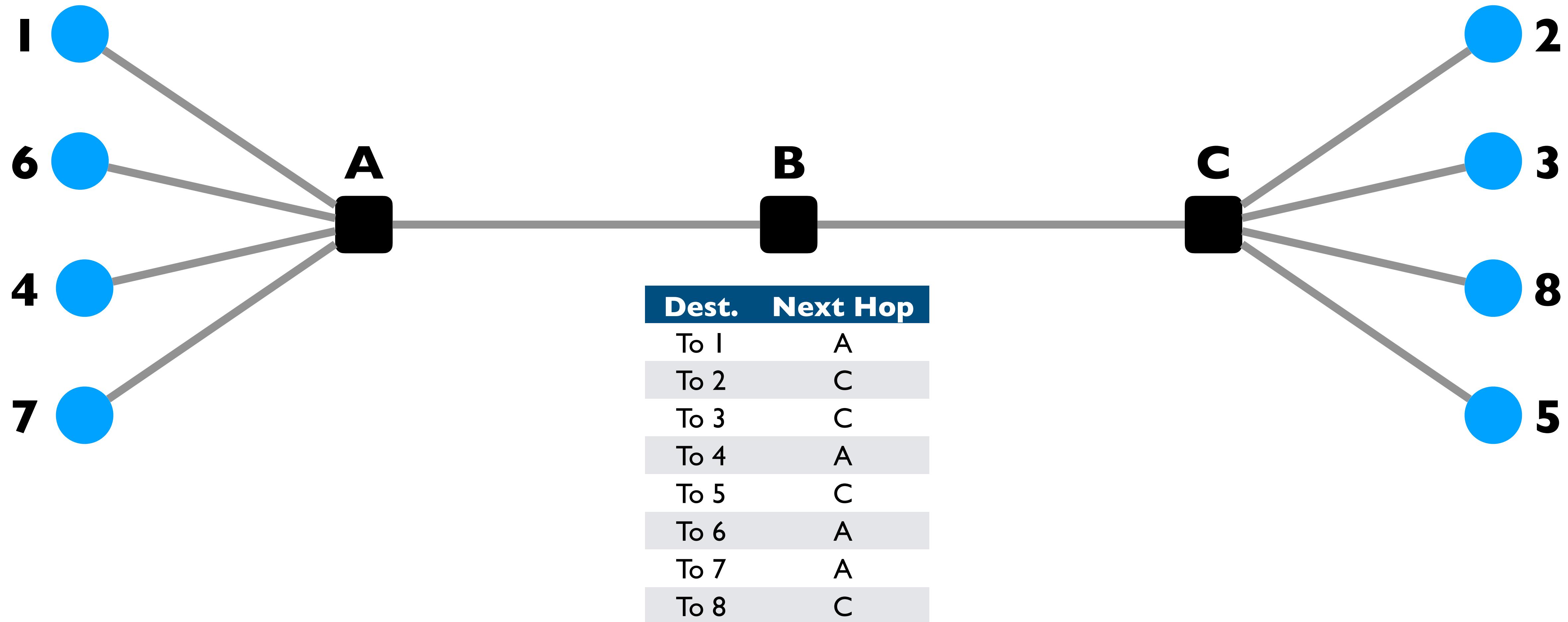
Scaling

- A router must be able to reach *any* destination
 - Given packet's destination address, lookup “next hop”
 - **Naive:** Have an entry for each destination
 - There would be **billions of entries!**
 - **And routing updates per destination!**

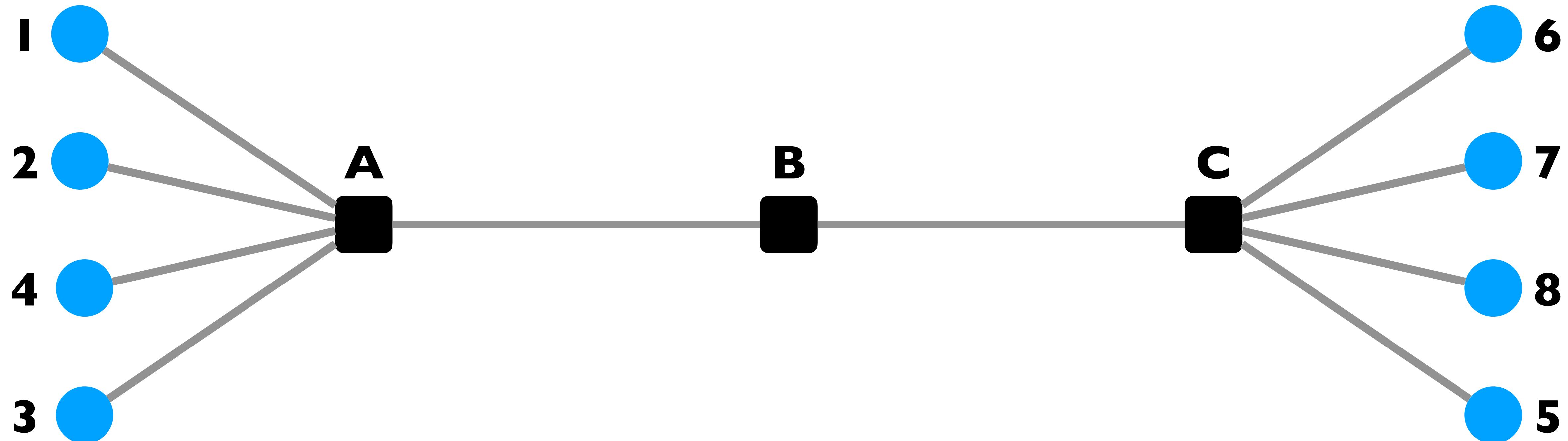




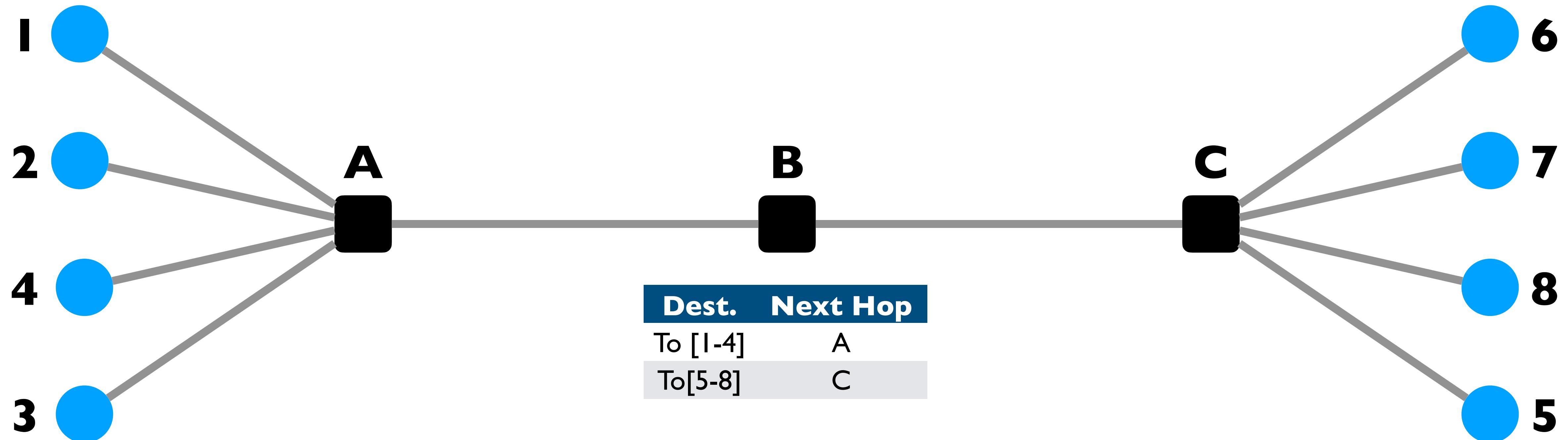
A smaller table at node B?



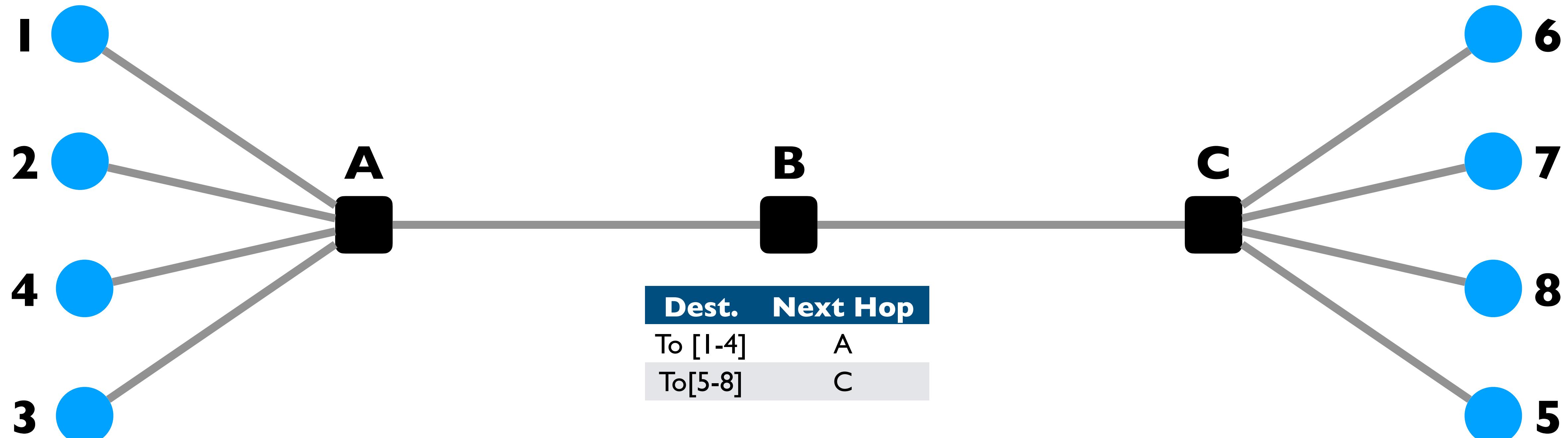
Re-number the end-systems?



Re-number the end-systems?

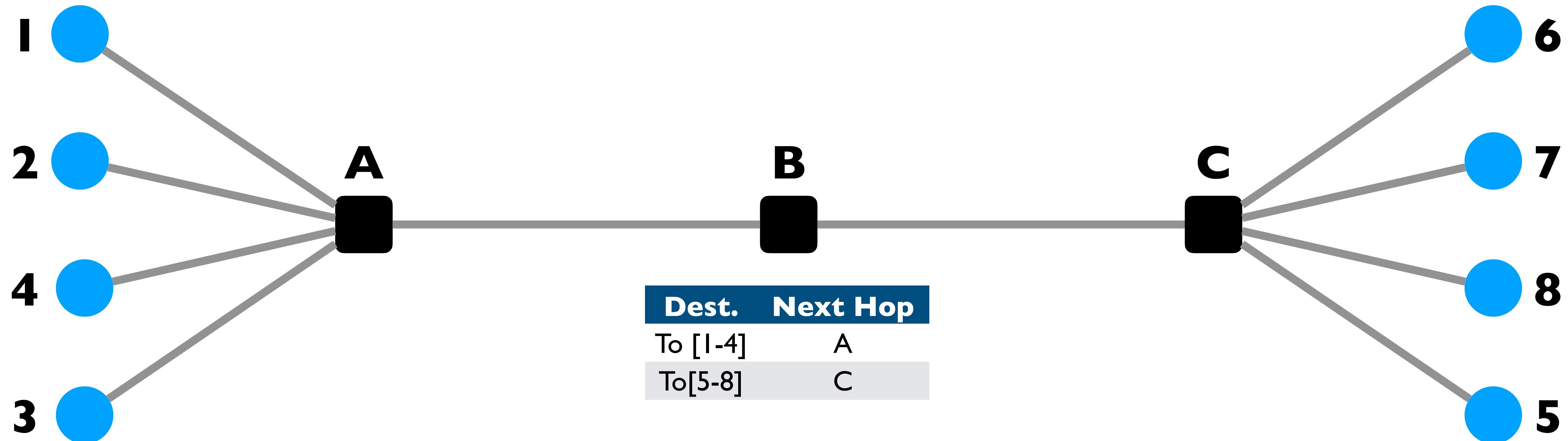


Re-number the end-systems?



- Careful address assignment → can aggregate
Multiple addresses into one range → scalability!

Re-number the end-systems?



- Careful address assignment → can aggregate Multiple addresses into one range → scalability!
- Akin to reducing the number of destinations

Scaling

- A router must be able to reach *any* destination
- **Naive:** Have an entry for each destination
- **Better:** Have an entry for a range of addresses
 - But can't do this if addresses are assigned randomly!

Scaling

- A router must be able to reach *any* destination
- **Naive:** Have an entry for each destination
- **Better:** Have an entry for a range of addresses
 - But can't do this if addresses are assigned randomly!
 - How addresses are allocated will matter!!

Scaling

- A router must be able to reach *any* destination
- **Naive:** Have an entry for each destination
- **Better:** Have an entry for a range of addresses
 - But can't do this if addresses are assigned randomly!
 - How addresses are allocated will matter!!

Host addressing is key to scaling

Two Key Challenges

- Scaling
- Administrative structure:
 - Issues of autonomy, policy, privacy

Administrative Structure Shapes Interdomain Routing

Administrative Structure Shapes Interdomain Routing

- ASes want freedom to pick routes based on **policy**
 - “*My traffic cannot be carried over my competitor’s network*”
 - “*I don’t want to carry A’s traffic through my network*”
 - Not expressible as Internet-wide “least-cost”!

Administrative Structure Shapes Interdomain Routing

- ASes want freedom to pick routes based on **policy**
 - “*My traffic cannot be carried over my competitor’s network*”
 - “*I don’t want to carry A’s traffic through my network*”
 - Not expressible as Internet-wide “least-cost”!
- ASes want **autonomy**
 - Want to choose their own internal routing protocol
 - Want to choose their own policy

Administrative Structure Shapes Interdomain Routing

- ASes want freedom to pick routes based on **policy**
 - “My traffic cannot be carried over my competitor’s network”
 - “I don’t want to carry A’s traffic through my network”
 - Not expressible as Internet-wide “least-cost”!
- ASes want **autonomy**
 - Want to choose their own internal routing protocol
 - Want to choose their own policy
- ASes want **privacy**
 - Choice of network topology, routing policies, etc.

Choice of Routing Algorithm

Choice of Routing Algorithm

- Link State (LS) vs. Distance Vector (DV)?
 - ***LS offers no privacy*** - broadcasts all network information
 - ***LS limits autonomy*** - need agreement on metric, algorithm

Choice of Routing Algorithm

- Link State (LS) vs. Distance Vector (DV)?
 - ***LS offers no privacy*** - broadcasts all network information
 - ***LS limits autonomy*** - need agreement on metric, algorithm
- DV is a decent starting point
 - Per-destination updates by intermediate nodes give us a hook
 - But wasn't designed to implement policy
 - And is vulnerable to loops if shortest paths not taken

Choice of Routing Algorithm

- Link State (LS) vs. Distance Vector (DV)?
 - ***LS offers no privacy*** - broadcasts all network information
 - ***LS limits autonomy*** - need agreement on metric, algorithm
- DV is a decent starting point
 - Per-destination updates by intermediate nodes give us a hook
 - But wasn't designed to implement policy
 - And is vulnerable to loops if shortest paths not taken

The “Border Gateway Protocol” (BGP) extends distance-vector ideas to accommodate policy

Questions?

Outline

- Addressing
- BGP
 - **Today:** context & basic ideas
 - **Next lecture:** details and issues

Outline

- Addressing
- BGP
 - **Today:** context & basic ideas
 - **Next lecture:** details and issues

Addressing Goal: Scalable Routing

Addressing Goal: Scalable Routing

- **State:** Small forwarding tables at routers
 - Much less than the number of hosts

Addressing Goal: Scalable Routing

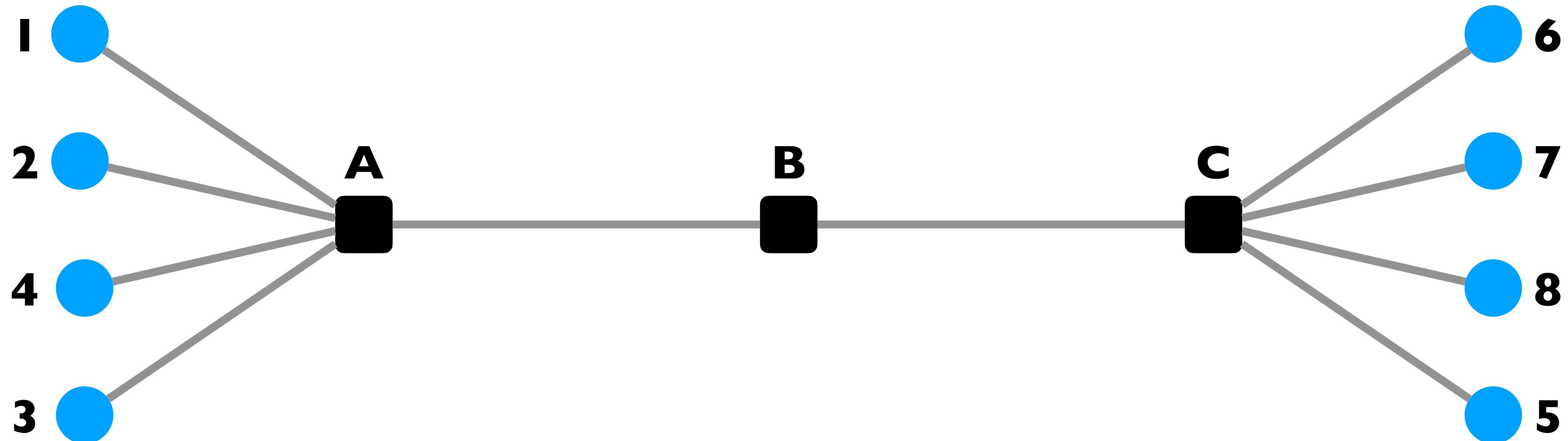
- **State:** Small forwarding tables at routers
 - Much less than the number of hosts
- **Churn:** Limited rate of change in routing tables

Addressing Goal: Scalable Routing

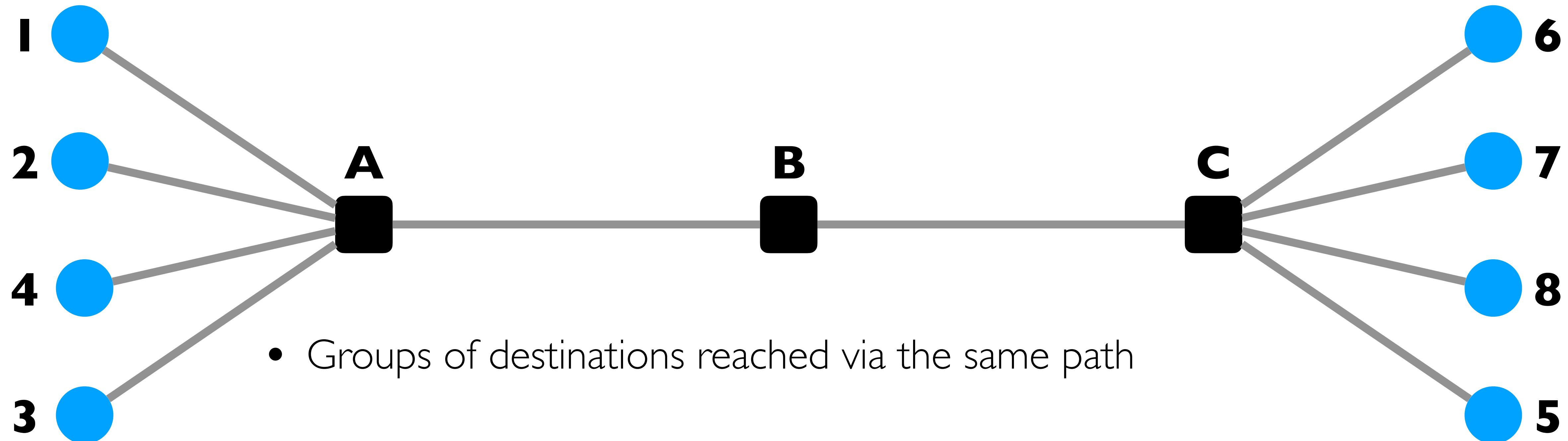
- **State:** Small forwarding tables at routers
 - Much less than the number of hosts
- **Churn:** Limited rate of change in routing tables

Ability to aggregate addresses is crucial for both
(One entry to summarize many addresses)

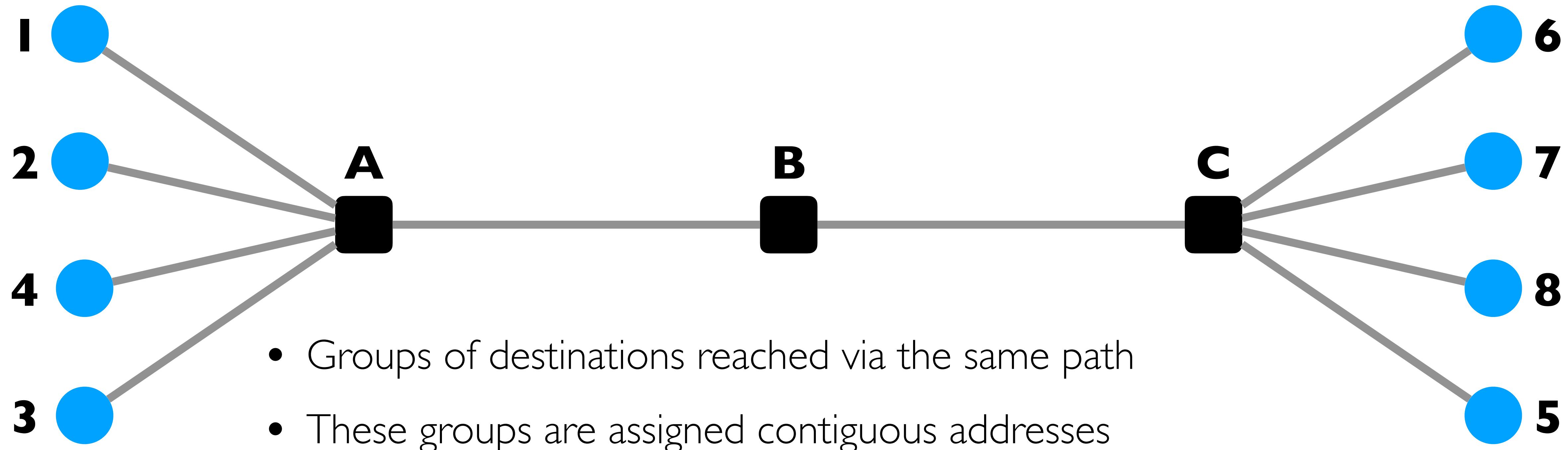
Aggregation only works if...



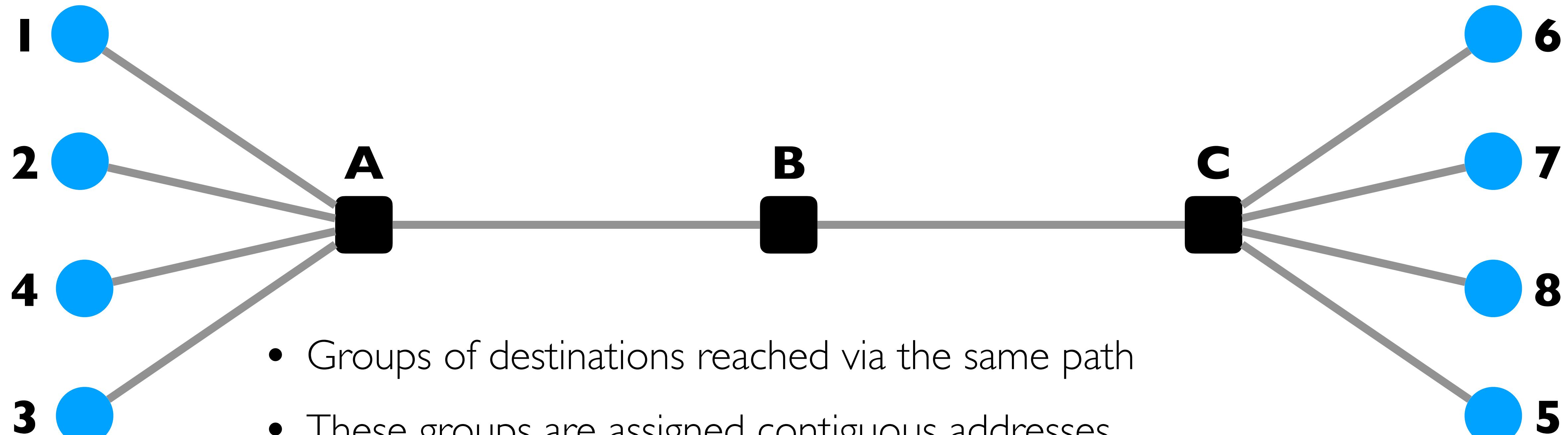
Aggregation only works if...



Aggregation only works if...

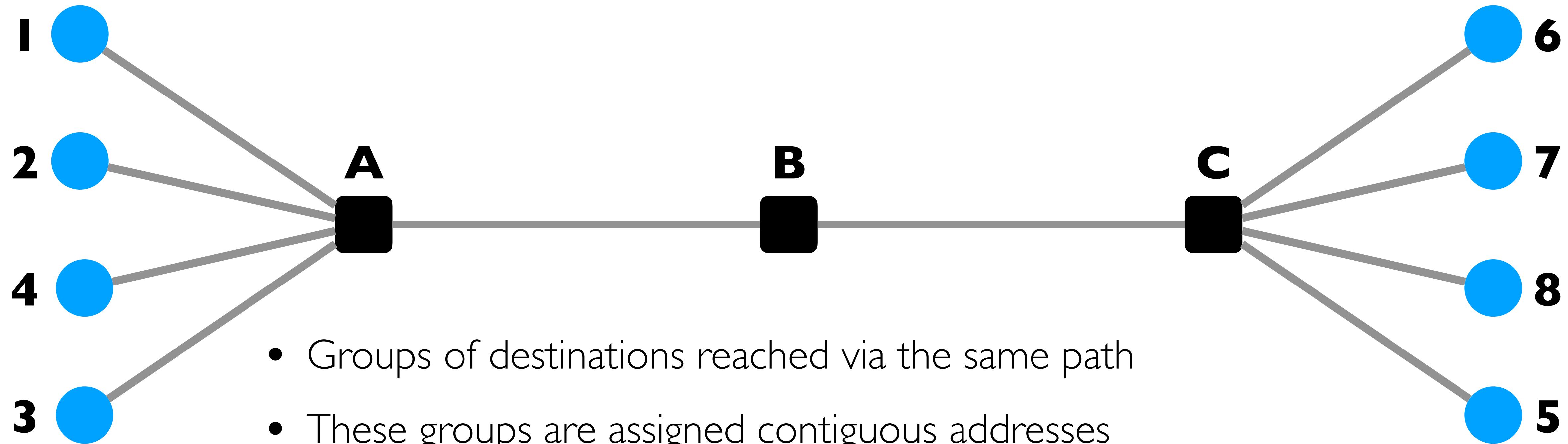


Aggregation only works if...



- Groups of destinations reached via the same path
- These groups are assigned contiguous addresses
- These groups are relatively stable

Aggregation only works if...



- Groups of destinations reached via the same path
- These groups are assigned contiguous addresses
- These groups are relatively stable
- Few enough groups to make forwarding easy

Hence, IP Addressing: Hierarchical

Hence, IP Addressing: Hierarchical

- **Hierarchical address structure**

Hence, IP Addressing: Hierarchical

- **Hierarchical address structure**
- **Hierarchical address allocation**

Hence, IP Addressing: Hierarchical

- **Hierarchical address structure**
- **Hierarchical address allocation**
- **Hierarchical addresses and routing scalability**

IP Addresses (IPv4)

IP Addresses (IPv4)

- Unique 32-bit number associated with a host

IP Addresses (IPv4)

- Unique 32-bit number associated with a host

0000|100 00|000|0 100|1|10 0000|0|

IP Addresses (IPv4)

- Unique 32-bit number associated with a host

0000|100 00|000|0 100|1|10 0000|0|

- Represented with the **“dotted quad”** notation, e.g., 12.34.158.5

IP Addresses (IPv4)

- Unique 32-bit number associated with a host

0000|100 00|000|0 100|1|10 0000|0|

- Represented with the **“dotted quad”** notation, e.g., 12.34.158.5

0000 100	00 000 0	100 1 10	00000 0
-----------------	-----------------	-----------------	-----------------

IP Addresses (IPv4)

- Unique 32-bit number associated with a host

0000 | 100 00 | 000 | 00 | 100 | 1 | 1 | 0 0000 | 0 |

- Represented with the **“dotted quad”** notation, e.g., 12.34.158.5



IP Addresses (IPv4)

- 32 bits are partitioned into a prefix and suffix components

IP Addresses (IPv4)

- 32 bits are partitioned into a prefix and suffix components
- Prefix is the **network component**; suffix is the **host component**

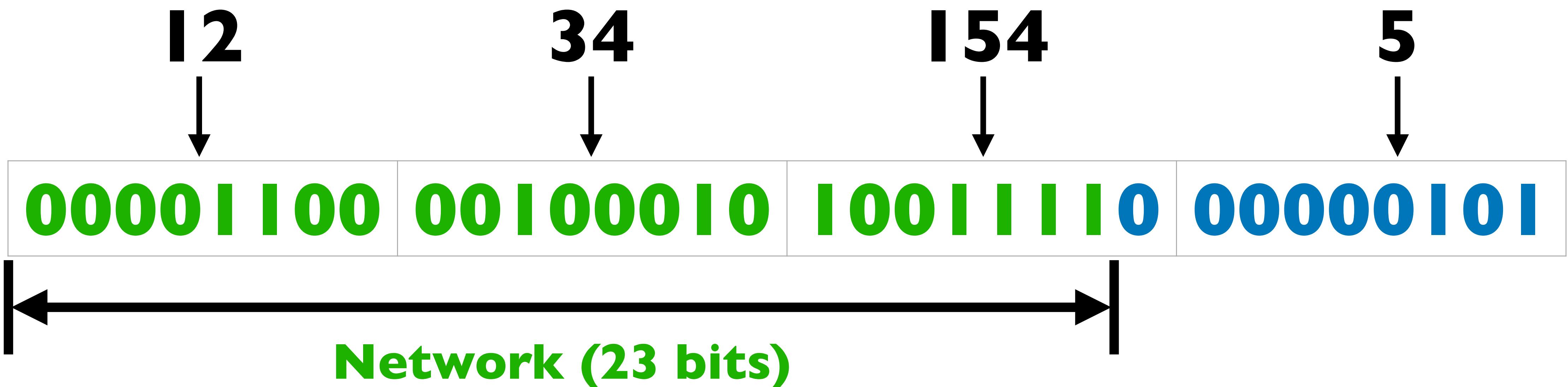
IP Addresses (IPv4)

- 32 bits are partitioned into a prefix and suffix components
- Prefix is the **network component**; suffix is the **host component**



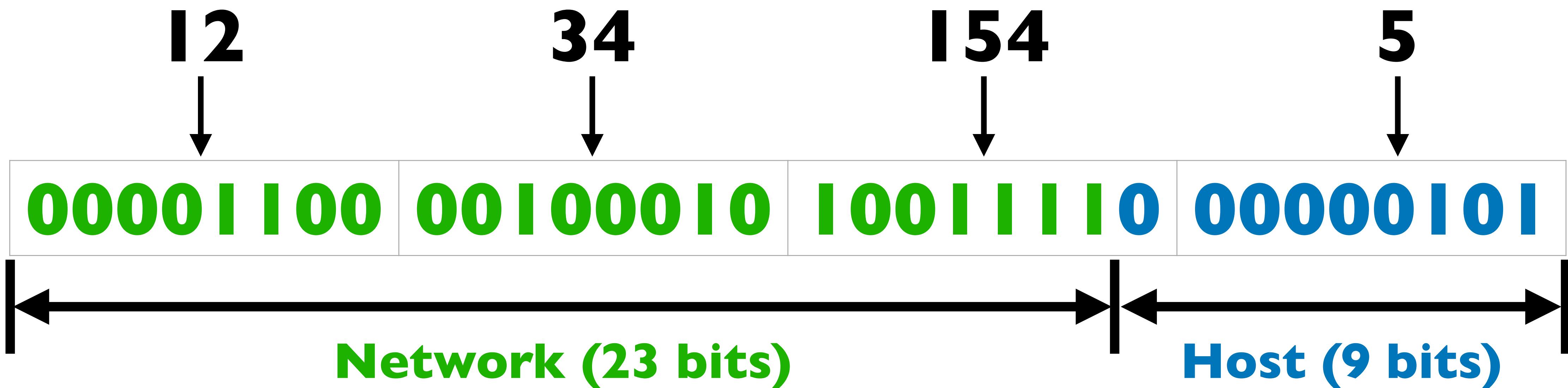
IP Addresses (IPv4)

- 32 bits are partitioned into a prefix and suffix components
- Prefix is the **network component**; suffix is the **host component**



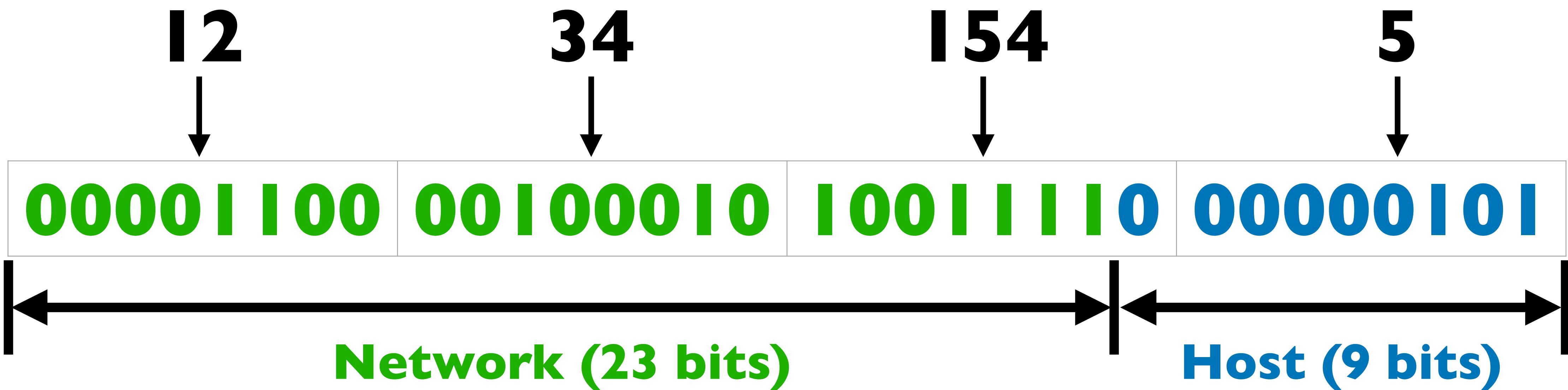
IP Addresses (IPv4)

- 32 bits are partitioned into a prefix and suffix components
- Prefix is the **network component**; suffix is the **host component**



IP Addresses (IPv4)

- 32 bits are partitioned into a prefix and suffix components
- Prefix is the **network component**; suffix is the **host component**



- Interdomain routing operates on the **network prefix**

History of Internet Addressing

History of Internet Addressing

- Always dotted quad notation

History of Internet Addressing

- Always dotted quad notation
- Always network/host split

History of Internet Addressing

- Always dotted quad notation
- Always network/host split
- But nature of that split has changed over time

Original Internet Addresses

Original Internet Addresses

- **First eight bits:** network component

Original Internet Addresses

- **First eight bits:** network component
- **Last 24 bits:** host component

Original Internet Addresses

- **First eight bits:** network component
- **Last 24 bits:** host component
- *Assumed 256 networks were more than enough!*

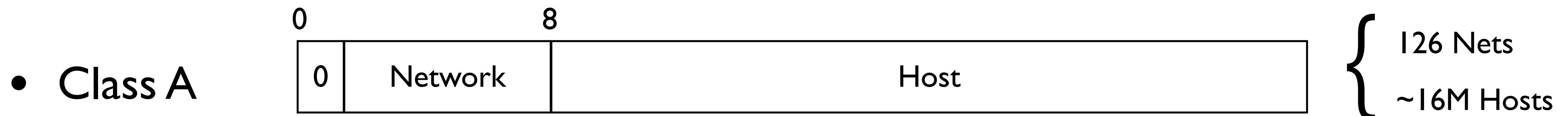
Next Design: “Classful” Addressing

Next Design: “Classful” Addressing

- **Three main classes**

Next Design: “Classful” Addressing

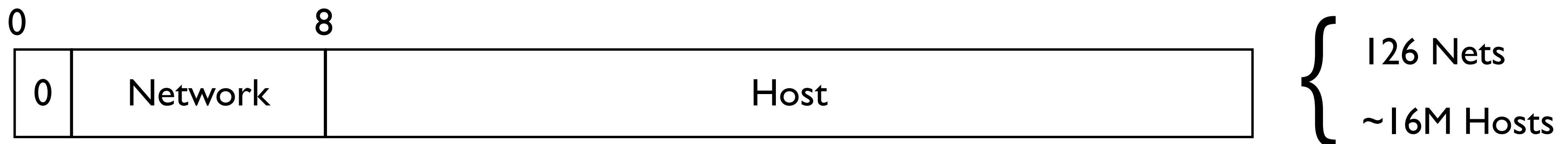
- **Three main classes**



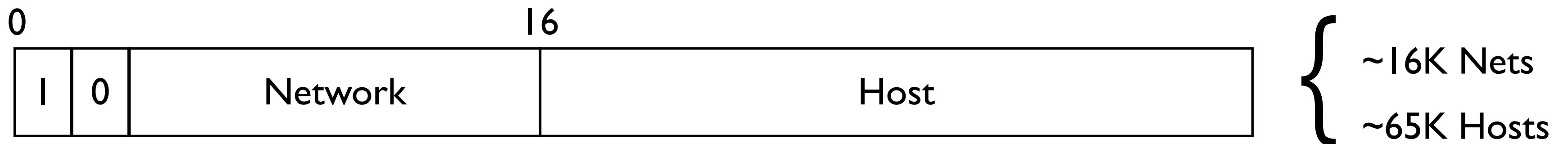
Next Design: “Classful” Addressing

- **Three main classes**

- Class A

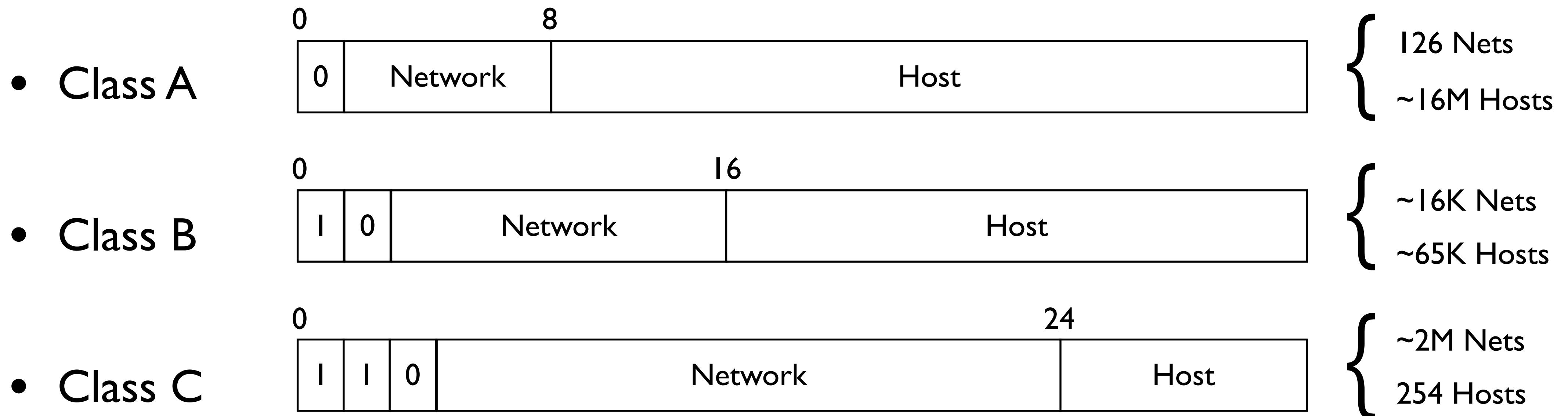


- Class B



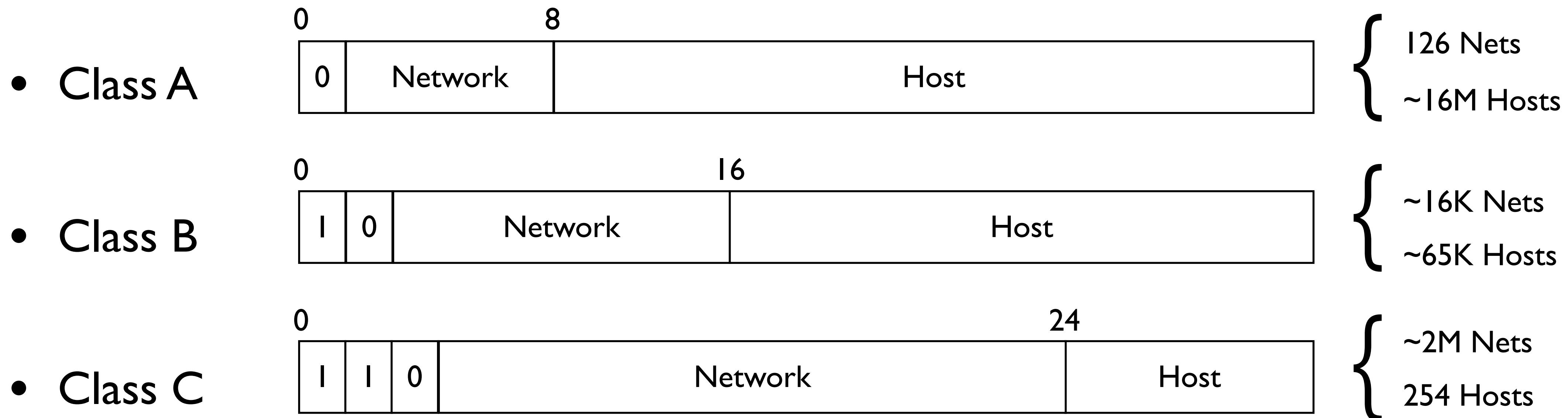
Next Design: “Classful” Addressing

- **Three main classes**



Next Design: “Classful” Addressing

- Three main classes



Problem: Networks only come in three sizes!

Today's Addressing: CIDR

Today's Addressing: CIDR

- **CIDR** = Classless InterDomain Routing

Today's Addressing: CIDR

- **CIDR** = Classless InterDomain Routing
- **Idea:** Flexible division between network and host addresses

Today's Addressing: CIDR

- **CIDR** = Classless InterDomain Routing
- **Idea:** Flexible division between network and host addresses
- **Motivation:** Offer a better tradeoff between size of the routing table and efficient use of the IP address space

CIDR Example

CIDR Example

- Suppose a network has 50 computers

CIDR Example

- Suppose a network has 50 computers
 - Allocate 6 bits for host addresses (since $2^5 < 50 < 2^6$)

CIDR Example

- Suppose a network has 50 computers
 - Allocate 6 bits for host addresses (since $2^5 < 50 < 2^6$)
 - Remaining $32 - 6 = 26$ bits as a network prefix

CIDR Example

- Suppose a network has 50 computers
 - Allocate 6 bits for host addresses (since $2^5 < 50 < 2^6$)
 - Remaining $32 - 6 = 26$ bits as a network prefix
- Flexible boundary means the boundary must be explicitly specified with the network address!

CIDR Example

- Suppose a network has 50 computers
 - Allocate 6 bits for host addresses (since $2^5 < 50 < 2^6$)
 - Remaining $32 - 6 = 26$ bits as a network prefix
- Flexible boundary means the boundary must be explicitly specified with the network address!
 - Informally, “slash 26” → 128.23.9/26

CIDR Example

- Suppose a network has 50 computers
 - Allocate 6 bits for host addresses (since $2^5 < 50 < 2^6$)
 - Remaining $32 - 6 = 26$ bits as a network prefix
- Flexible boundary means the boundary must be explicitly specified with the network address!
 - Informally, “slash 26” → 128.23.9/26
 - Formally, prefix represented with a 32 bit mask: 255.255.255.192 where all network prefix bits set to “1” and host suffix bits to 0

Classful vs. Classless addresses

Classful vs. Classless addresses

- Example: An organization needs 500 addresses

Classful vs. Classless addresses

- Example: An organization needs 500 addresses
 - A single class C address not enough (254 hosts)

Classful vs. Classless addresses

- Example: An organization needs 500 addresses
 - A single class C address not enough (254 hosts)
 - Instead a class B address is allocated (~65k hosts)

Classful vs. Classless addresses

- Example: An organization needs 500 addresses
 - A single class C address not enough (254 hosts)
 - Instead a class B address is allocated (~65k hosts)
 - That's overkill, a huge waste!

Classful vs. Classless addresses

- Example: An organization needs 500 addresses
 - A single class C address not enough (254 hosts)
 - Instead a class B address is allocated (~65k hosts)
 - That's overkill, a huge waste!
- CIDR allows an arbitrary prefix-suffix boundary

Classful vs. Classless addresses

- Example: An organization needs 500 addresses
 - A single class C address not enough (254 hosts)
 - Instead a class B address is allocated (~65k hosts)
 - That's overkill, a huge waste!
- CIDR allows an arbitrary prefix-suffix boundary
 - Hence, organization allocated a single /23 address (equivalent of two class C's)

Classful vs. Classless addresses

- **Example:** An organization needs 500 addresses
 - A single class C address not enough (254 hosts)
 - Instead a class B address is allocated (~65k hosts)
 - That's overkill, a huge waste!
- **CIDR allows an arbitrary prefix-suffix boundary**
 - Hence, organization allocated a single /23 address (equivalent of two class C's)
 - Waste < 50% always!

Hence, IP Addressing: Hierarchical

- **Hierarchical address structure**
- **Hierarchical address allocation**
- **Hierarchical addresses and routing scalability**

Allocation Done Hierarchically

Allocation Done Hierarchically

- Internet Corporation for Assigned Names and Numbers (ICANN) gives large blocks to...

Allocation Done Hierarchically

- Internet Corporation for Assigned Names and Numbers (ICANN) gives large blocks to...
- Regional Internet Registries, such as the American Registry for Internet Names (ARIN), which gives blocks to...

Allocation Done Hierarchically

- Internet Corporation for Assigned Names and Numbers (ICANN) gives large blocks to...
- Regional Internet Registries, such as the American Registry for Internet Names (ARIN), which gives blocks to...
- Large institutions (ISPs), which gives addresses to...

Allocation Done Hierarchically

- Internet Corporation for Assigned Names and Numbers (ICANN) gives large blocks to...
- Regional Internet Registries, such as the American Registry for Internet Names (ARIN), which gives blocks to...
- Large institutions (ISPs), which gives addresses to...
- Individuals and smaller institutions

Allocation Done Hierarchically

- Internet Corporation for Assigned Names and Numbers (ICANN) gives large blocks to...
- Regional Internet Registries, such as the American Registry for Internet Names (ARIN), which gives blocks to...
- Large institutions (ISPs), which gives addresses to...
- Individuals and smaller institutions
- Fake example:
 - ICANN → ARIN → AT&T → Yale → CS

CIDR: Addresses allocated in contiguous prefix chunks

CIDR: Addresses allocated in contiguous prefix chunks

Recursively break down chunks as you get closer to host

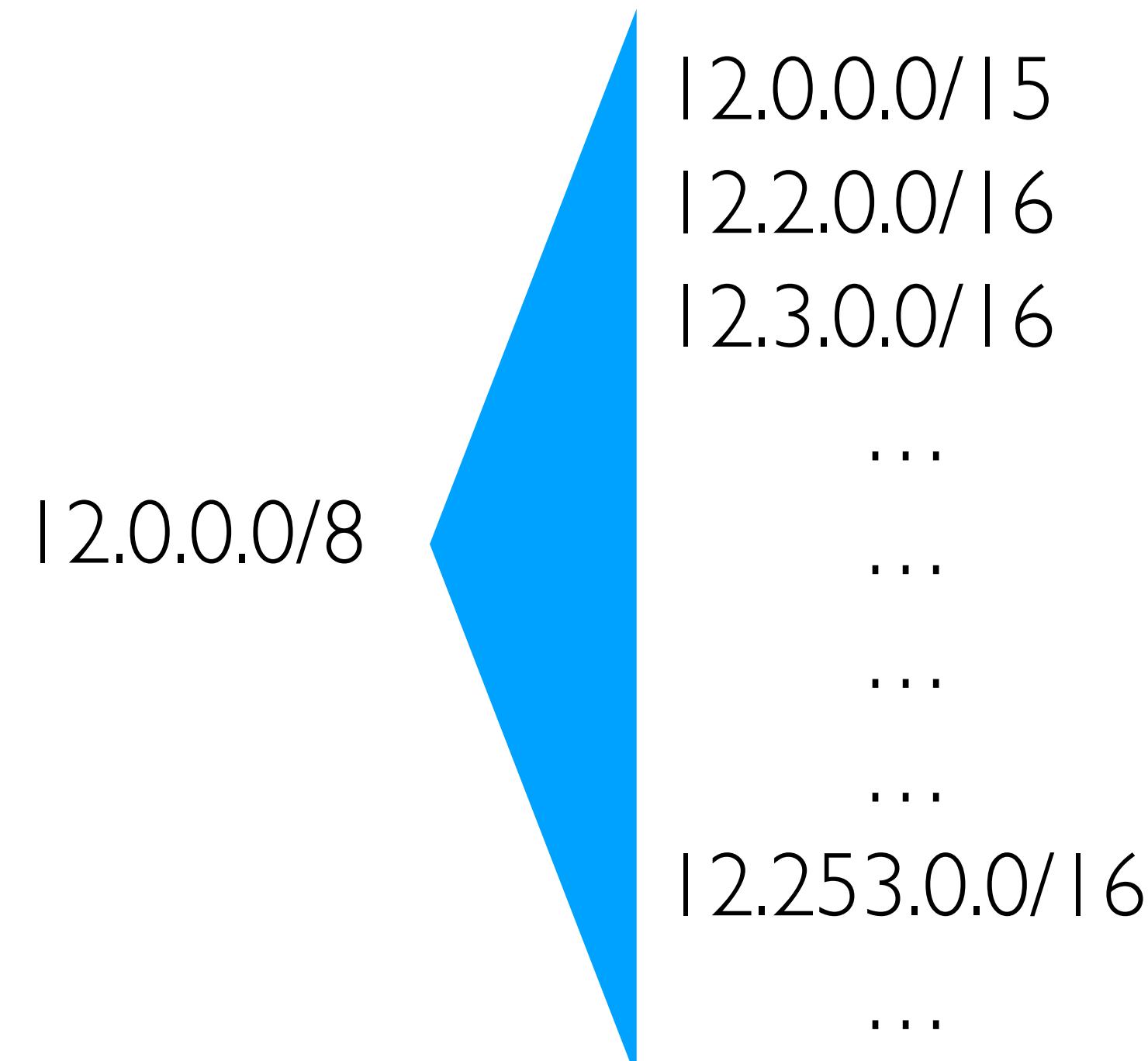
CIDR: Addresses allocated in contiguous prefix chunks

Recursively break down chunks as you get closer to host

12.0.0.0/8

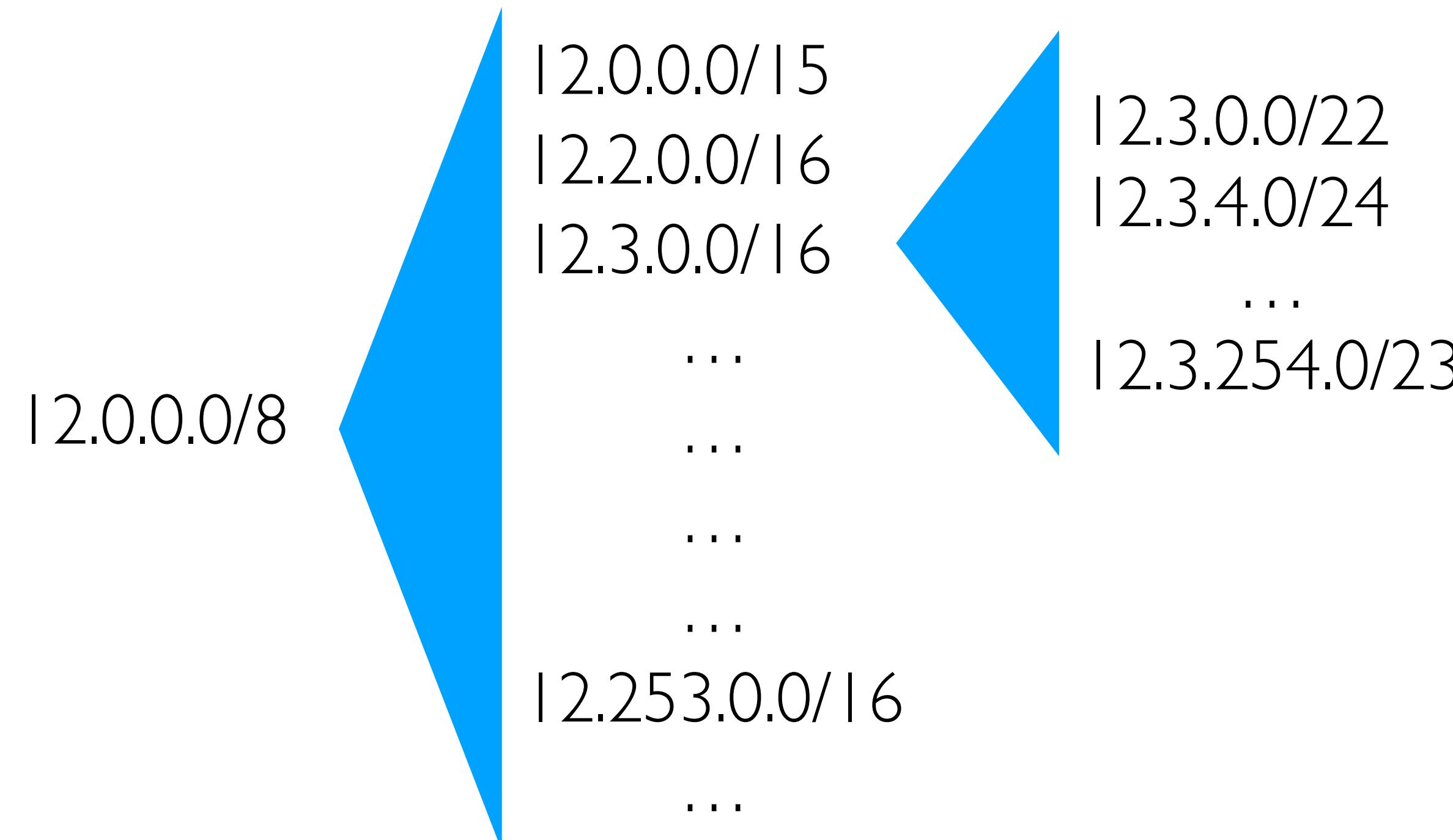
CIDR: Addresses allocated in contiguous prefix chunks

Recursively break down chunks as you get closer to host



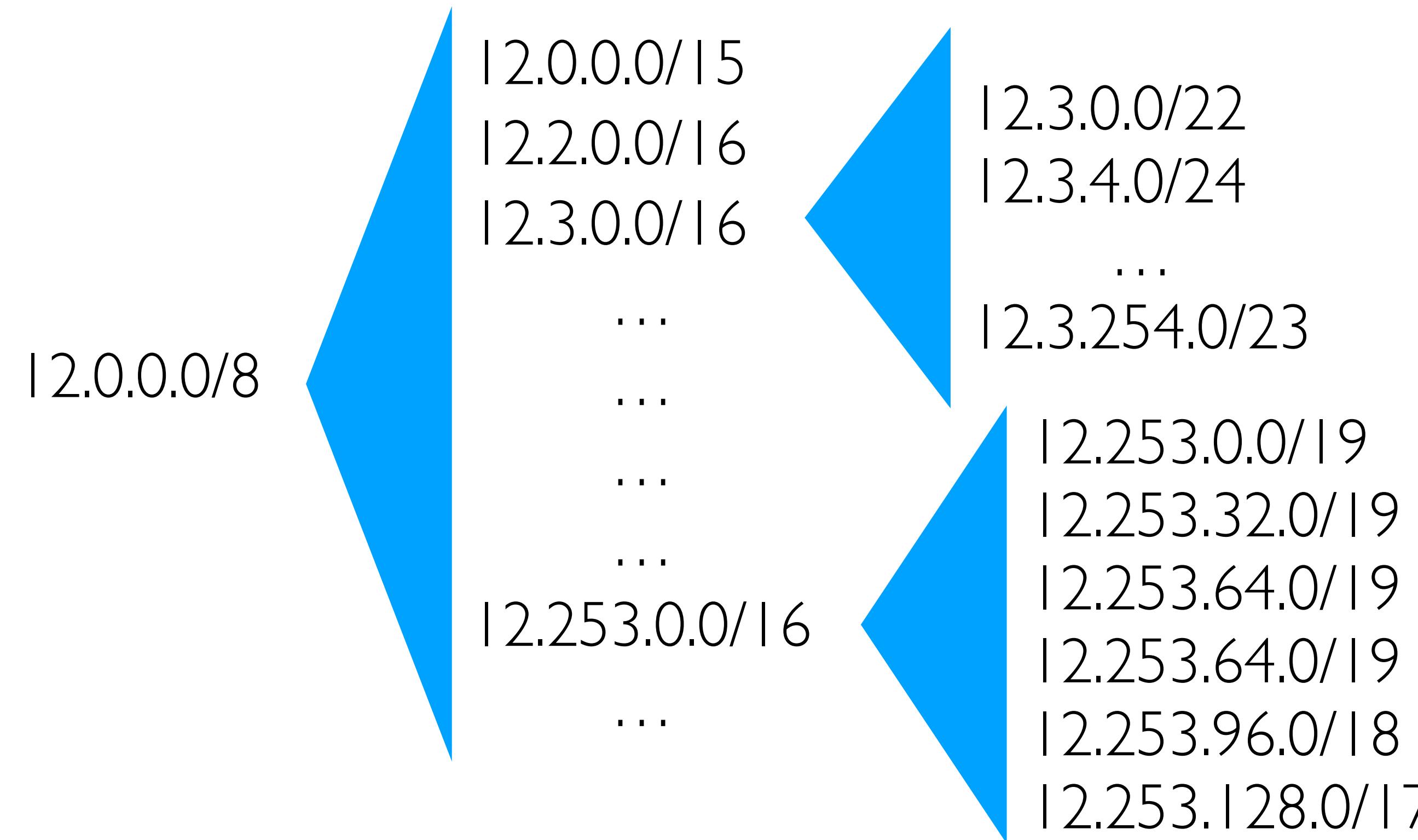
CIDR: Addresses allocated in contiguous prefix chunks

Recursively break down chunks as you get closer to host



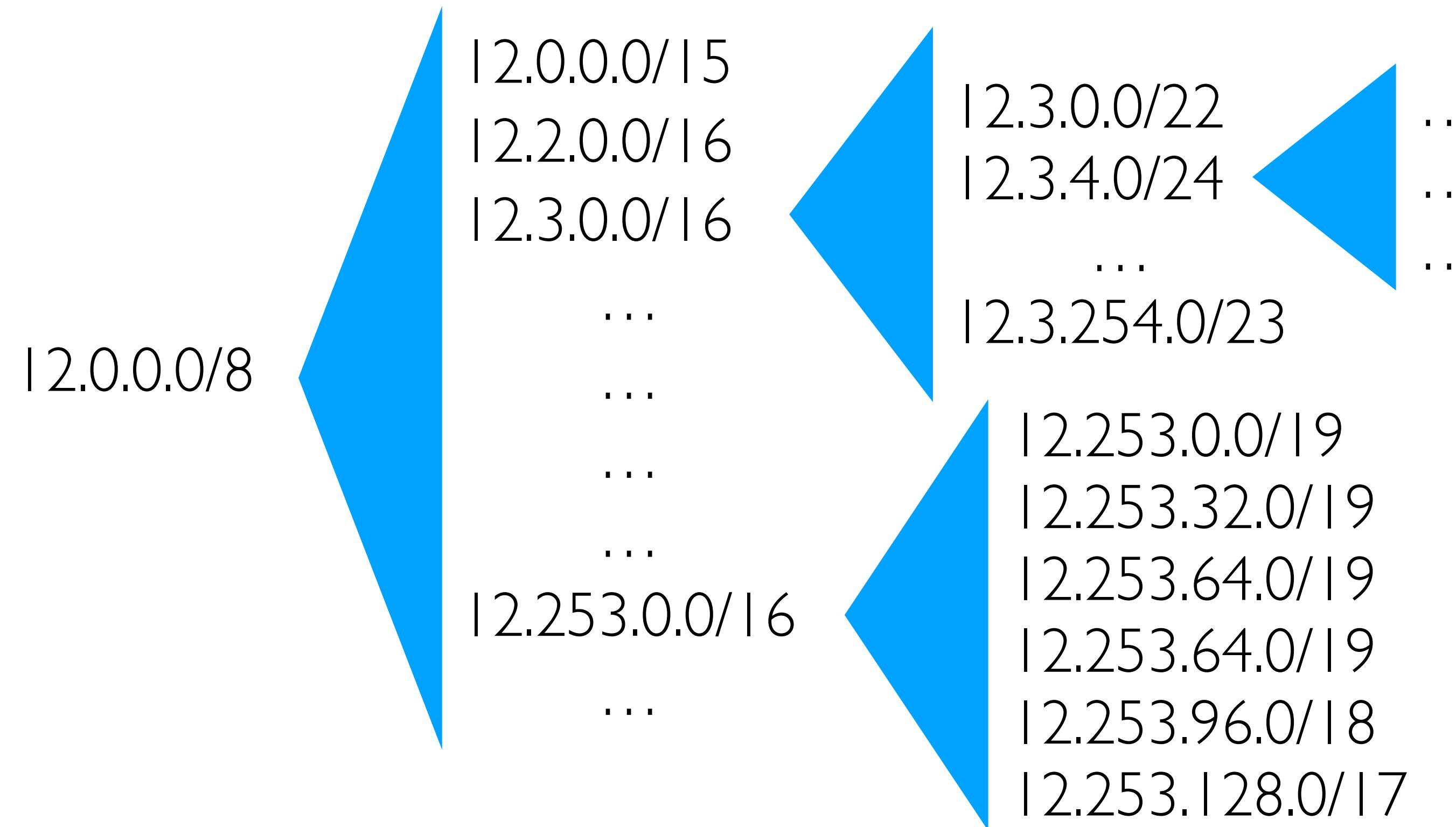
CIDR: Addresses allocated in contiguous prefix chunks

Recursively break down chunks as you get closer to host



CIDR: Addresses allocated in contiguous prefix chunks

Recursively break down chunks as you get closer to host



Fake Example in More Detail...

Fake Example in More Detail...

- ICANN gives ARIN several /8s

Fake Example in More Detail...

- ICANN gives ARIN several /8s
- ARIN gives AT&T one /8, **12.0.0.0/8**
 - Network prefix: **00001 100**

Fake Example in More Detail...

- ICANN gives ARIN several /8s
- ARIN gives AT&T one /8, **12.0.0.0/8**
 - Network prefix: **00001100**
- AT&T gives Yale a /16, **12.197.0.0/16**
 - Network prefix: **0000110011000101**

Fake Example in More Detail...

- ICANN gives ARIN several /8s
- ARIN gives AT&T one /8, **12.0.0.0/8**
 - Network prefix: **00001100**
- AT&T gives Yale a /16, **12.197.0.0/16**
 - Network prefix: **0000110011000101**
- Yale gives CS a /24, **12.197.45.0/24**
 - Network prefix: **000011001100010100101101**

Fake Example in More Detail...

- ICANN gives ARIN several /8s
- ARIN gives AT&T one /8, **12.0.0.0/8**
 - Network prefix: **00001100**
- AT&T gives Yale a /16, **12.197.0.0/16**
 - Network prefix: **0000110011000101**
- Yale gives CS a /24, **12.197.45.0/24**
 - Network prefix: **000011001100010100101101**
- CS gives my laptop a specific address, **12.197.45.23**
 - Address: **00001100110001010010110100010111**

Hence, IP Addressing: Hierarchical

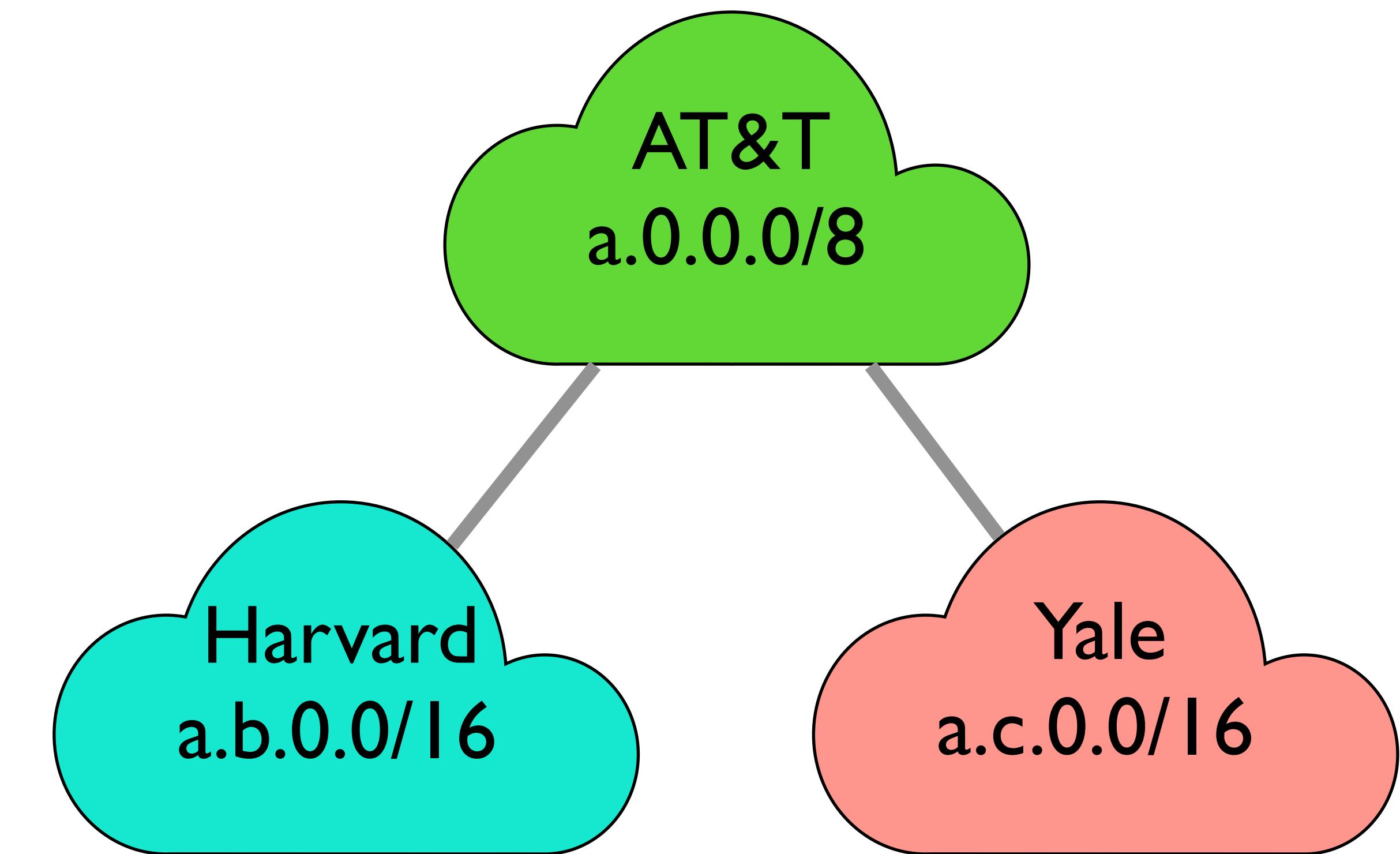
- **Hierarchical address structure**
- **Hierarchical address allocation**
- **Hierarchical addresses and routing scalability**

IP Addressing → Scalable Routing?

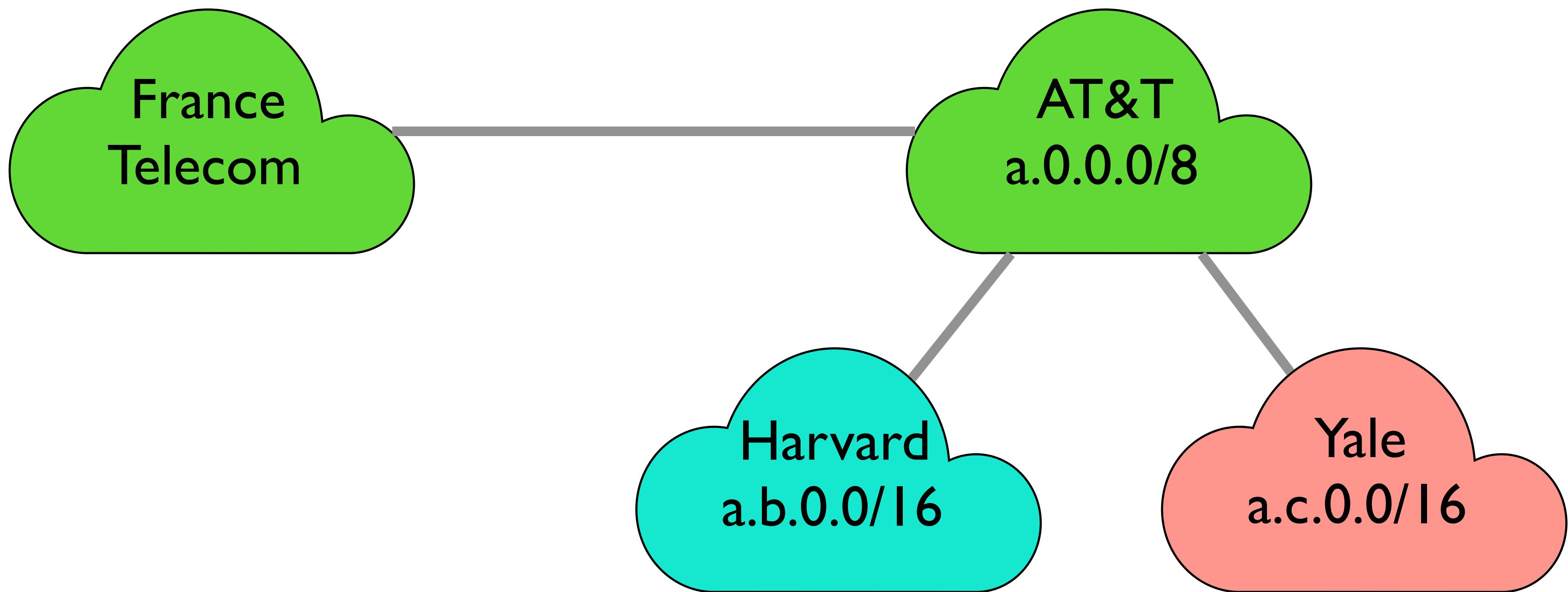
IP Addressing → Scalable Routing?

Hierarchical address allocation only helps routing scalability if allocation matches topological hierarchy

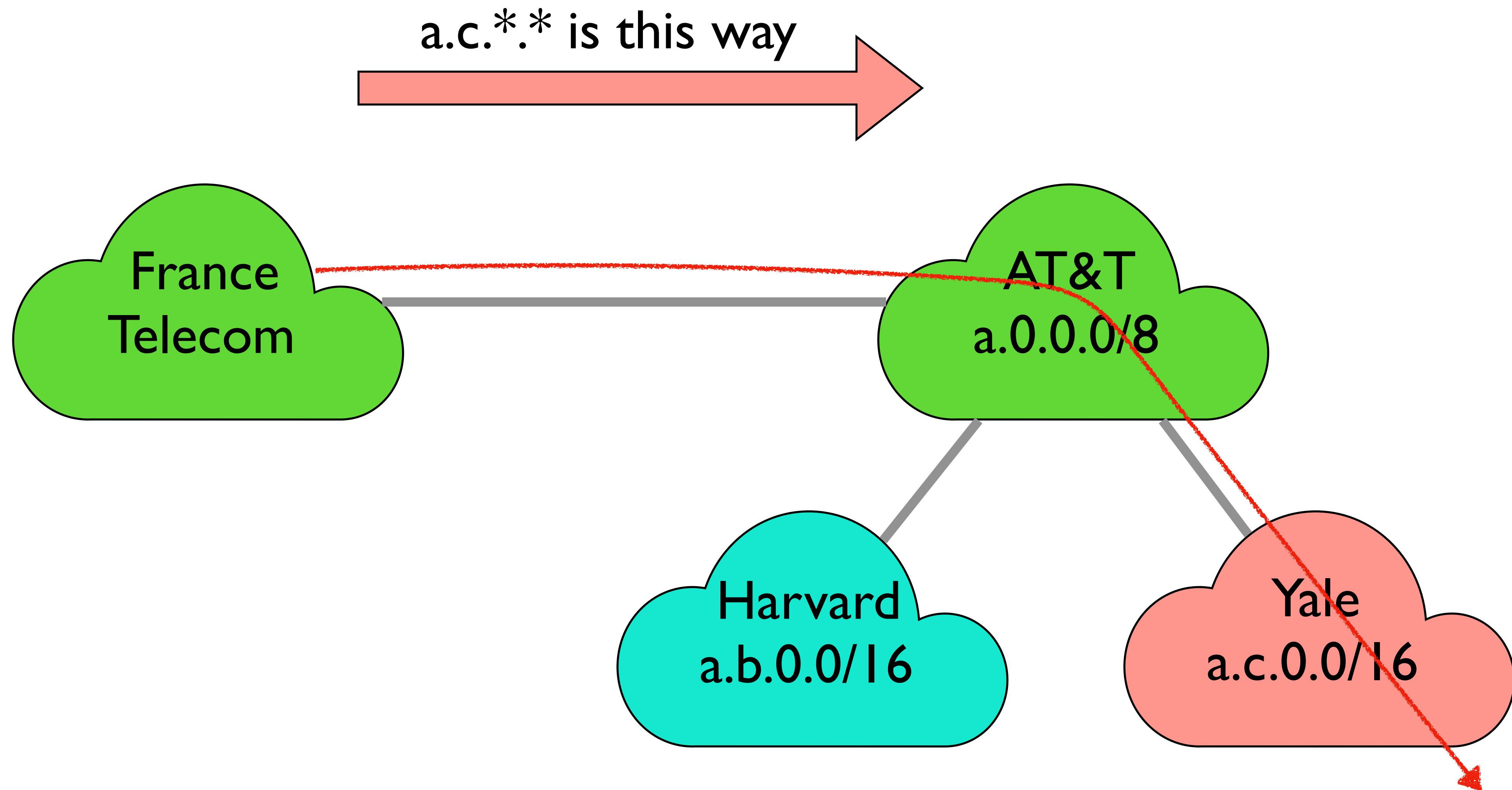
IP Addressing → Scalable Routing?



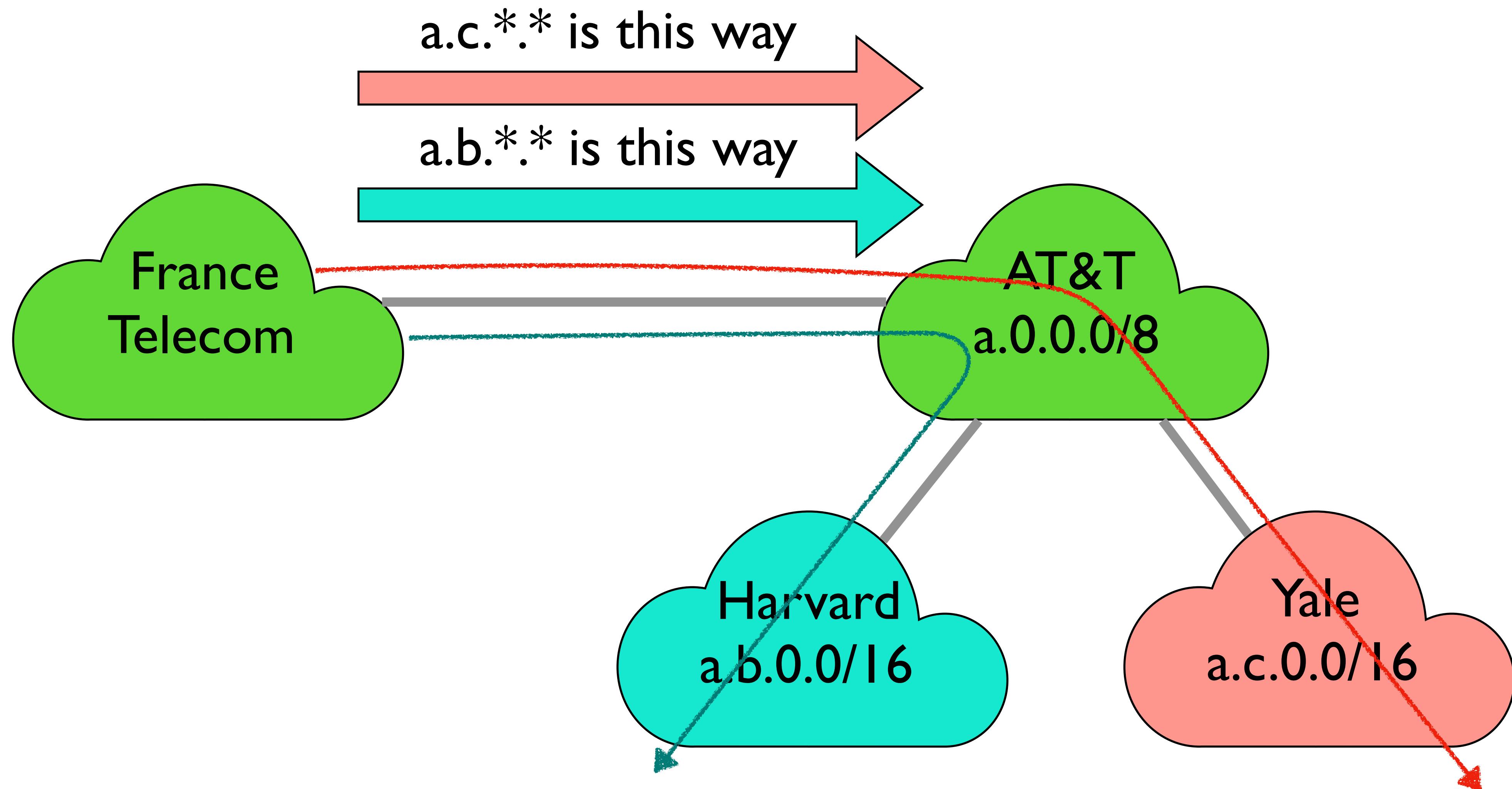
IP Addressing → Scalable Routing?



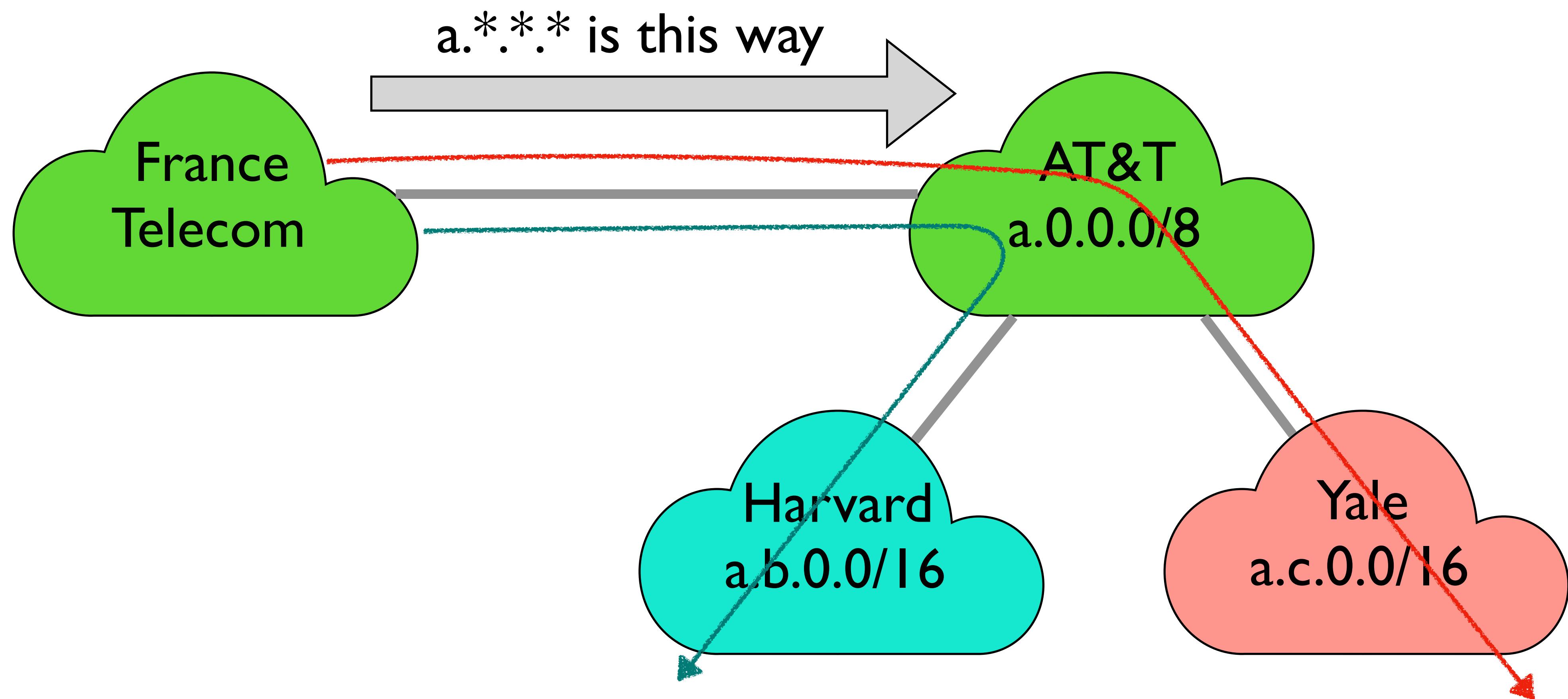
IP Addressing → Scalable Routing?



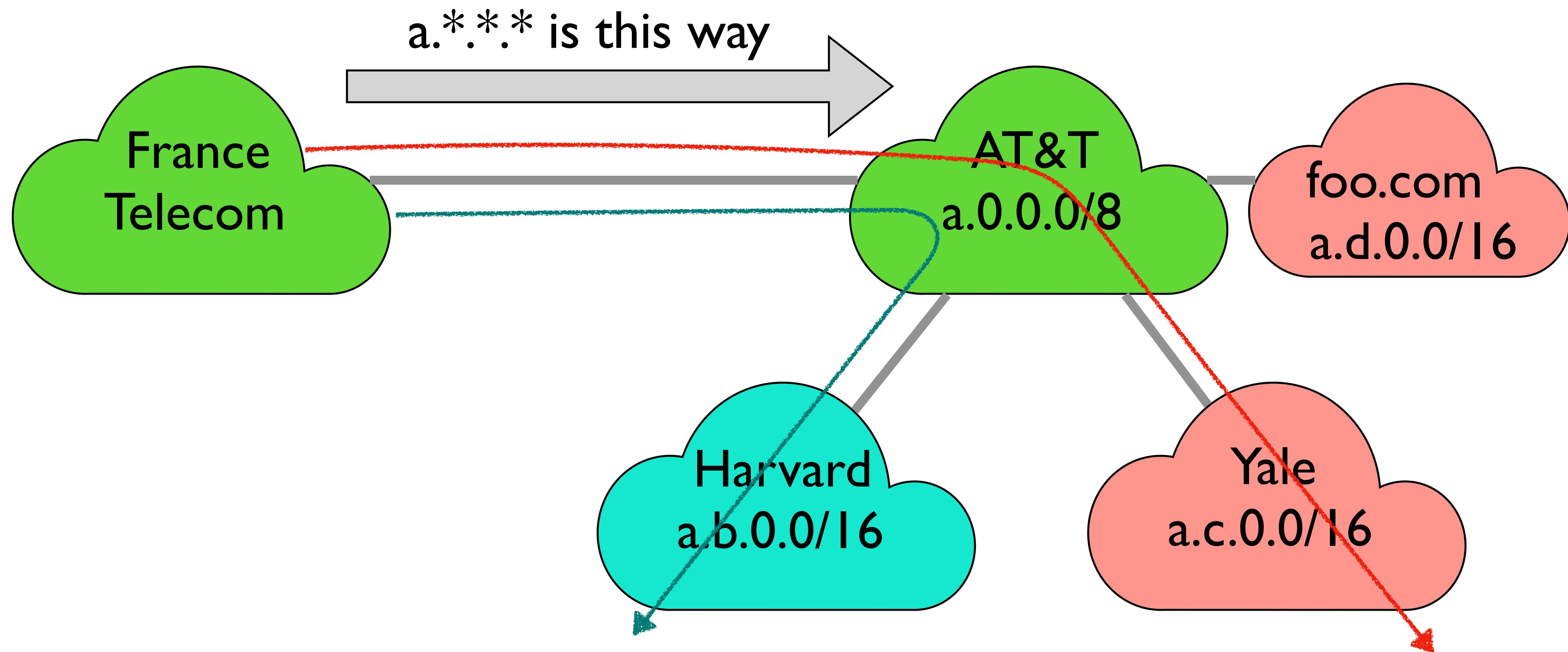
IP Addressing → Scalable Routing?



IP Addressing → Scalable Routing?

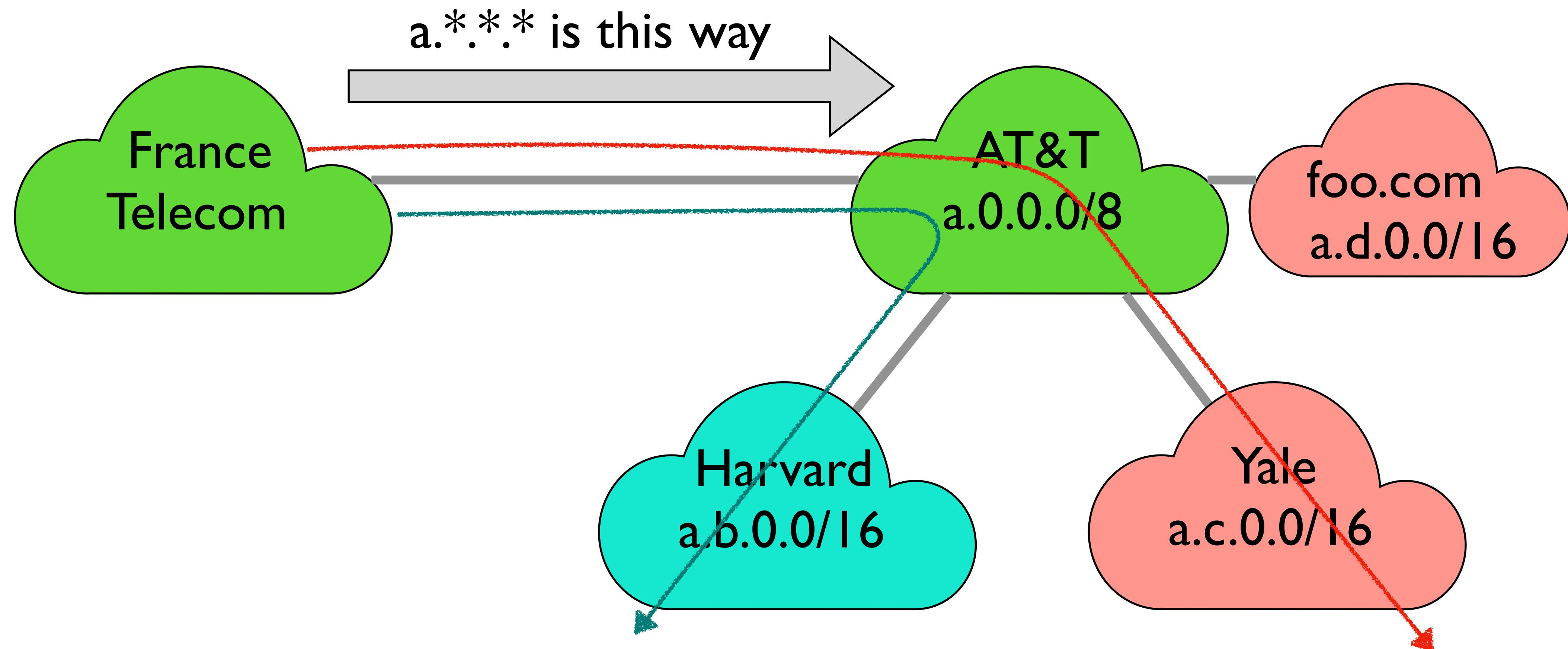


IP Addressing → Scalable Routing?

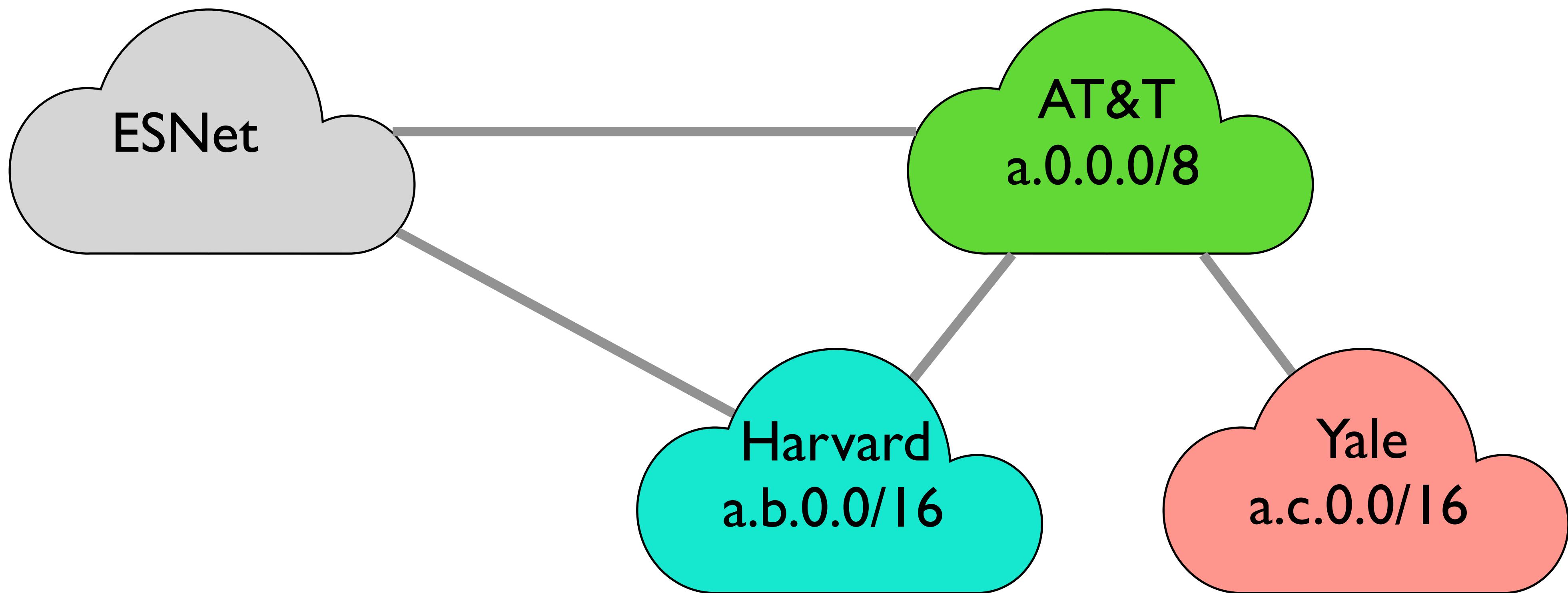


IP Addressing → Scalable Routing?

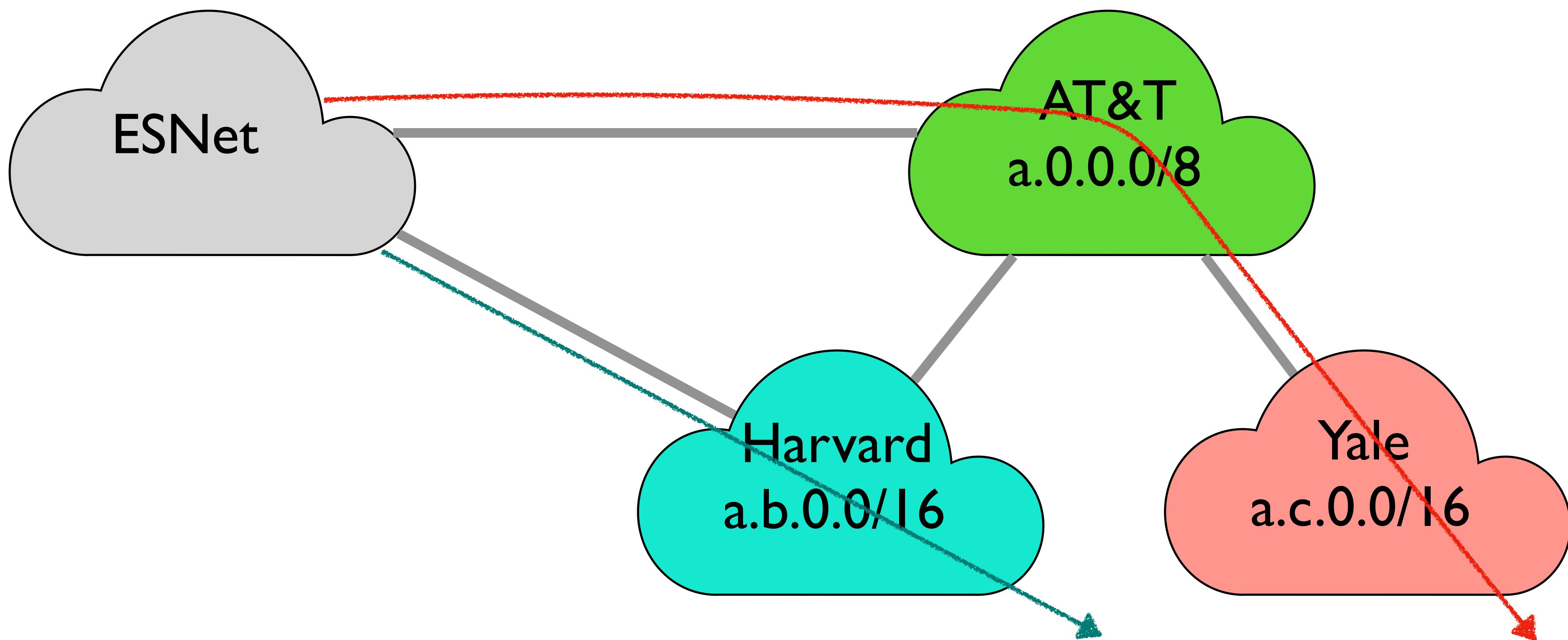
Can add new hosts/networks without updating the routing entries at France Telecom



IP Addressing → Scalable Routing?

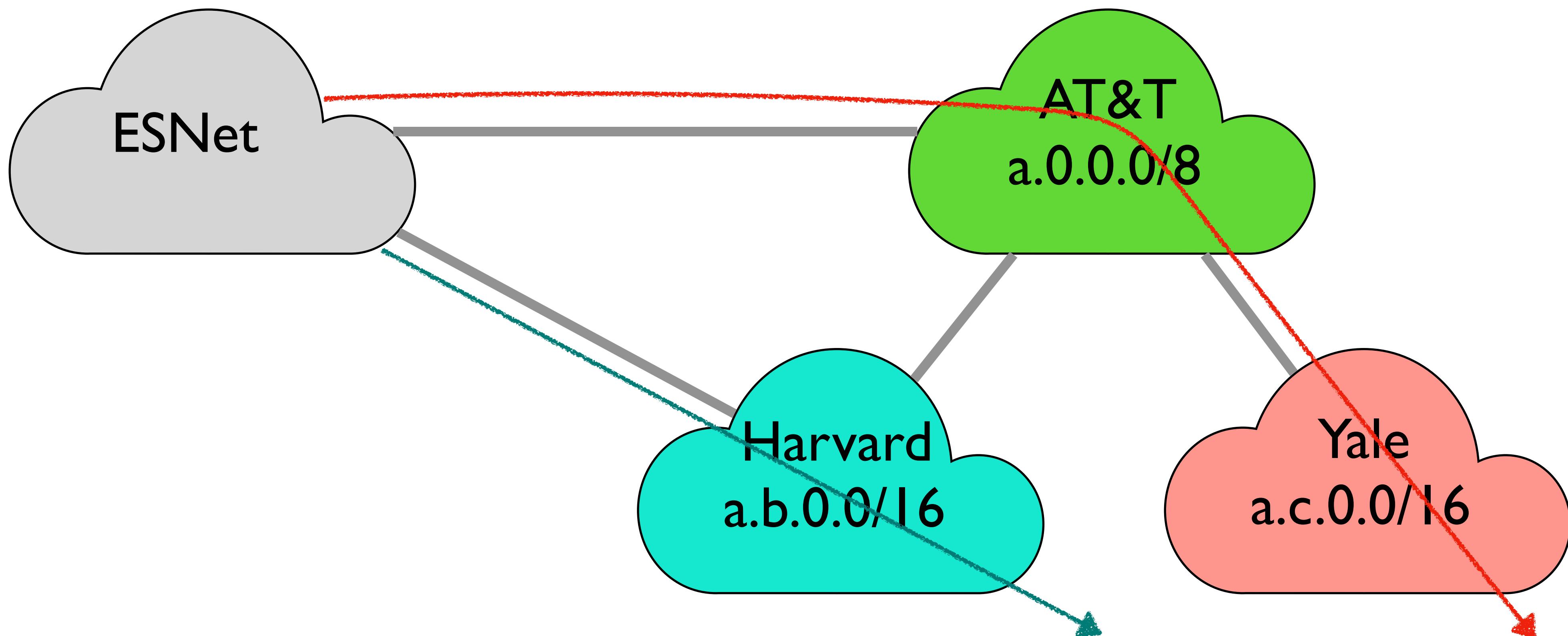


IP Addressing → Scalable Routing?



IP Addressing → Scalable Routing?

ESNet must maintain routing entries for $a.*.*$ and $a.b.*.*$



IP Addressing → Scalable Routing?

Hierarchical address allocation only helps routing scalability if allocation matches topological hierarchy

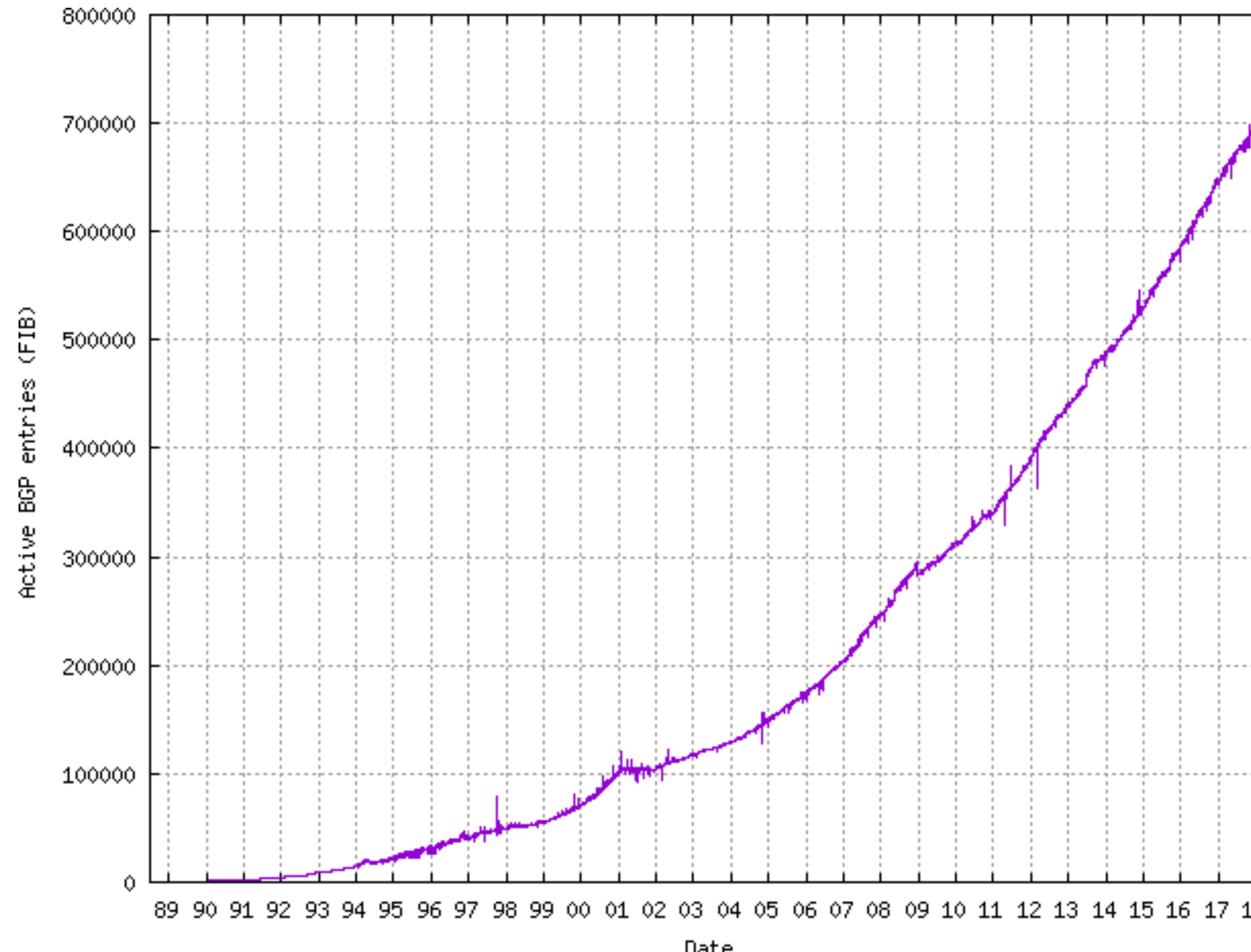
- **Problem:** May not be able to aggregate addresses for “multi-homed” networks

IP Addressing → Scalable Routing?

Hierarchical address allocation only helps routing scalability if allocation matches topological hierarchy

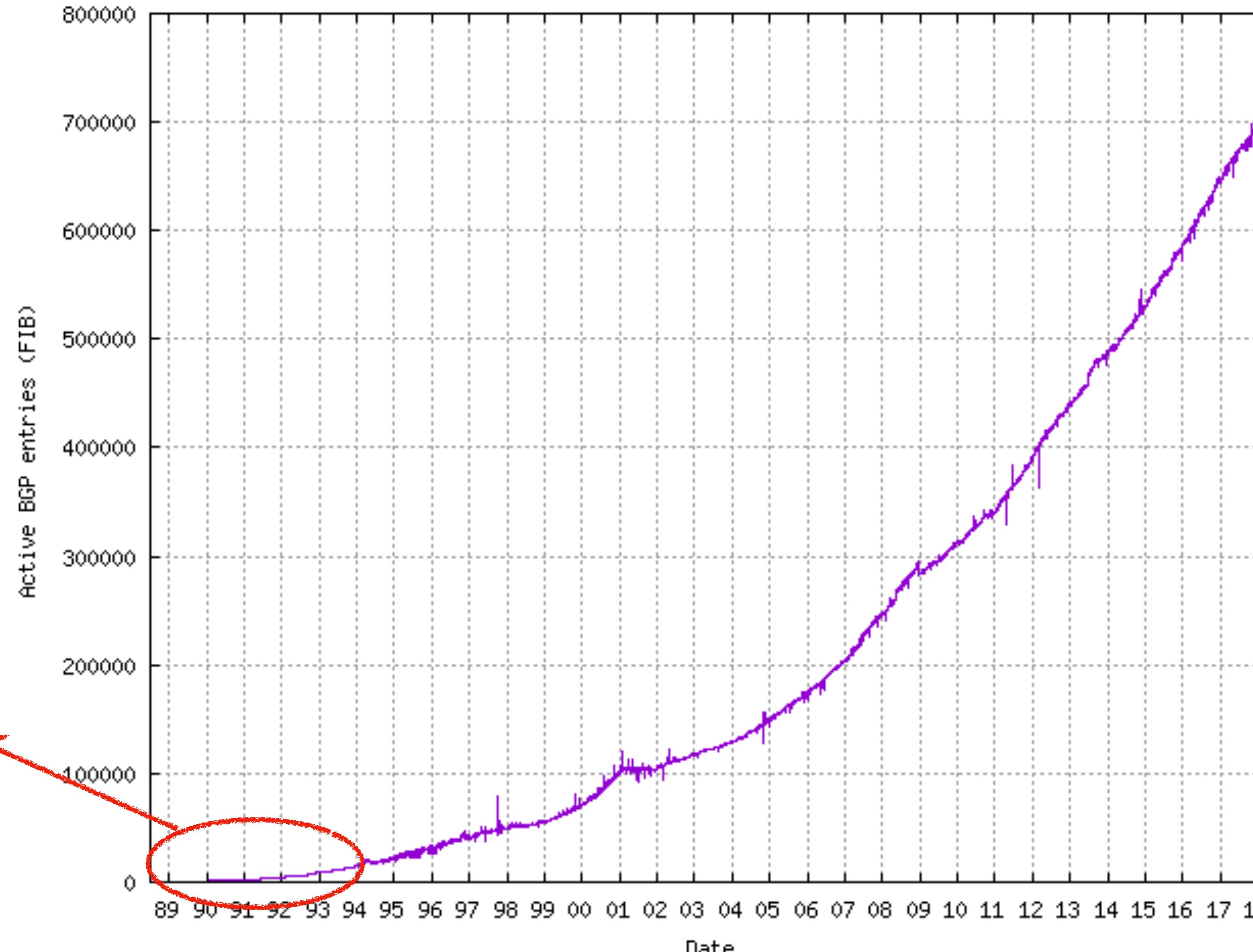
- **Problem:** May not be able to aggregate addresses for “multi-homed” networks
- Two competing forces in scalable routing
 - Aggregation reduces the number of routing entries
 - Multi-homing increases number of entries

Growth in Routed Prefixes



Growth in Routed Prefixes

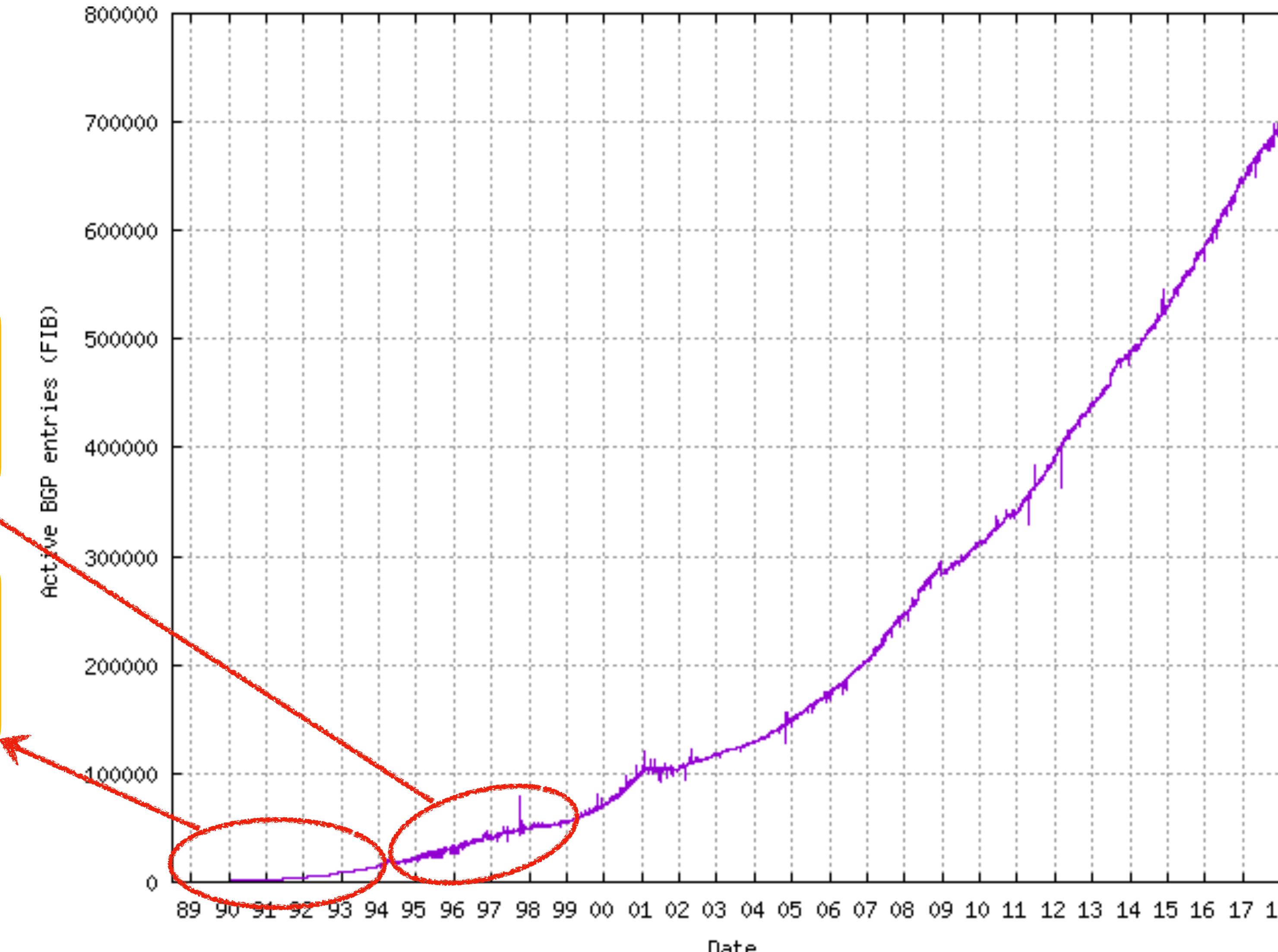
Initial growth
super-linear; no
aggregation



Growth in Routed Prefixes

Advent of CIDR
allows aggregation:
linear growth

Initial growth
super-linear; no
aggregation

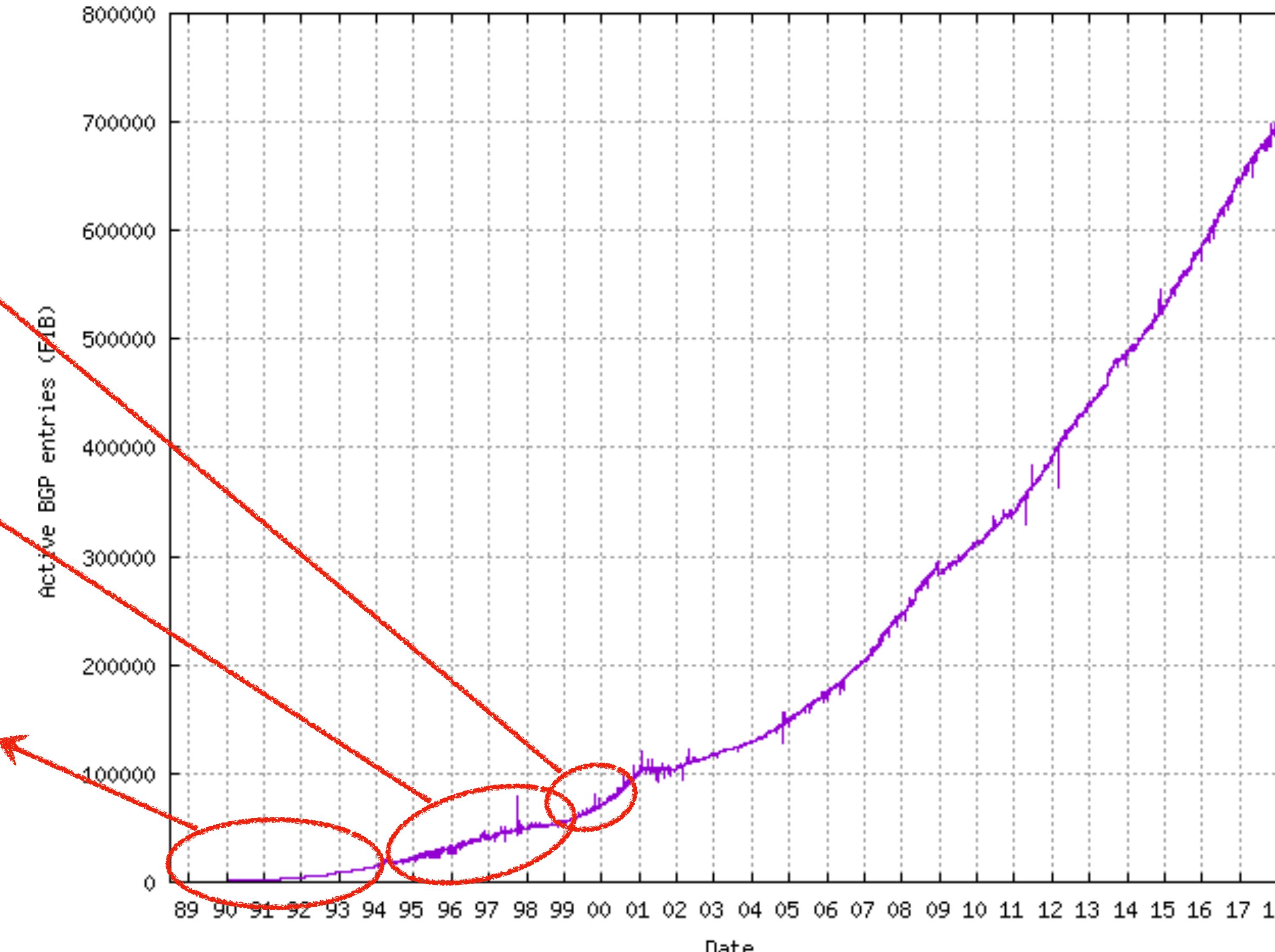


Growth in Routed Prefixes

Internet boom:
multihoming drives
superlinear growth

Advent of CIDR
allows aggregation:
linear growth

Initial growth
super-linear; no
aggregation



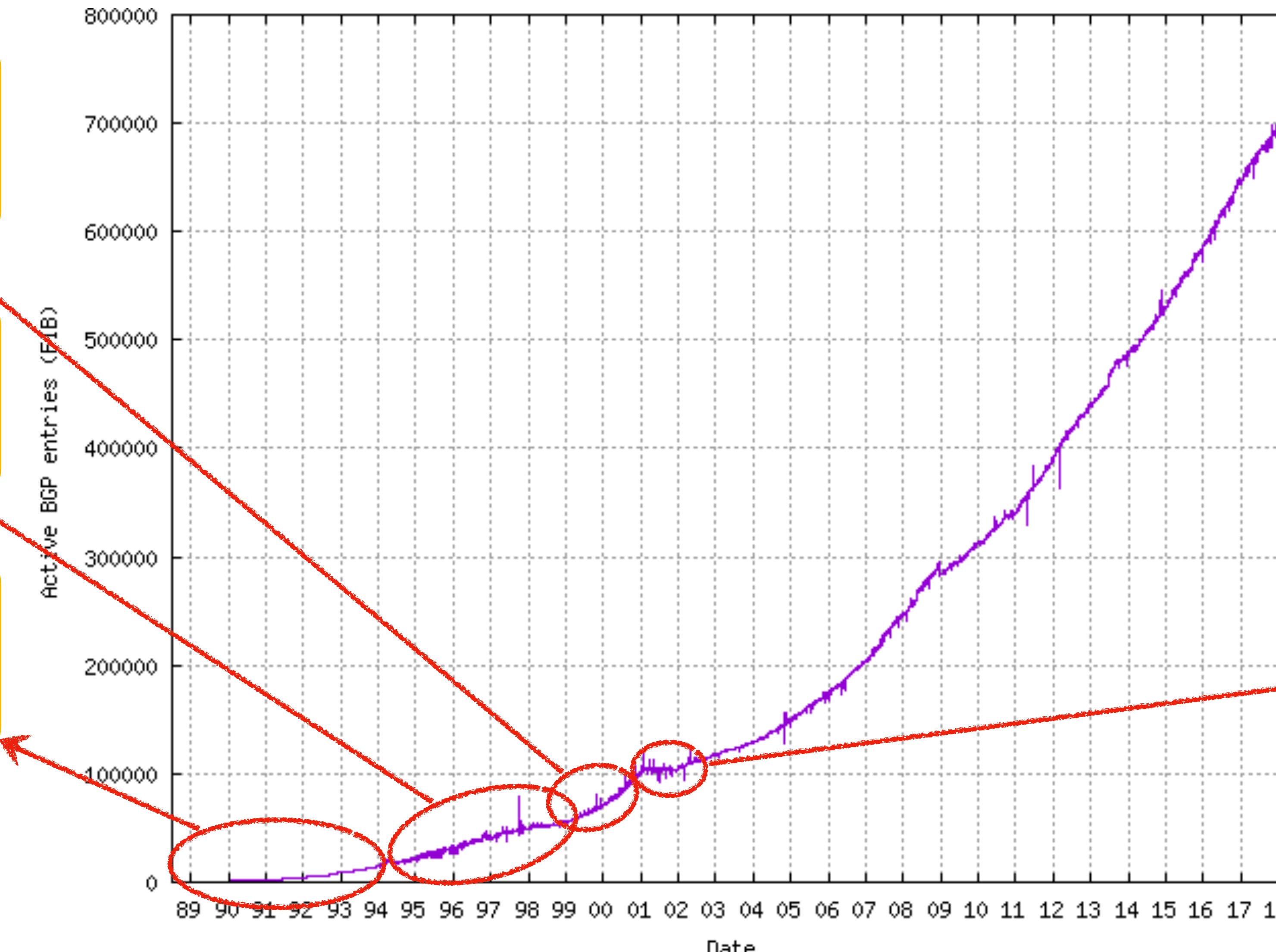
Growth in Routed Prefixes

Internet boom:
multihoming drives
superlinear growth

Advent of CIDR
allows aggregation:
linear growth

Initial growth
super-linear; no
aggregation

Dot-com
implosion: Internet
bubble bursts



Growth in Routed Prefixes

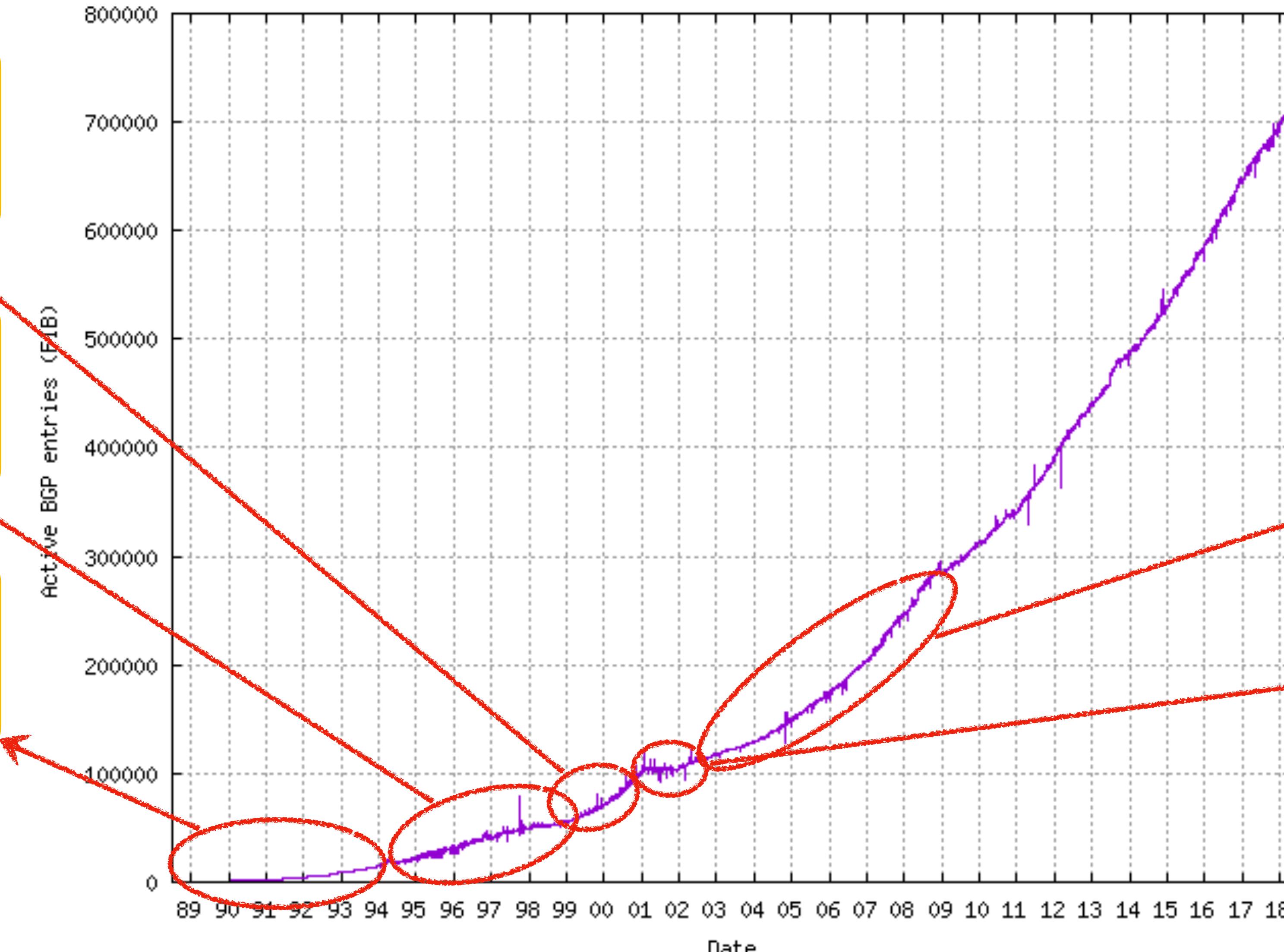
Internet boom:
multihoming drives
superlinear growth

Advent of CIDR
allows aggregation:
linear growth

Initial growth
super-linear; no
aggregation

Back in business

Dot-com
implosion: Internet
bubble bursts



Growth in Routed Prefixes

Internet boom:
multihoming drives
superlinear growth

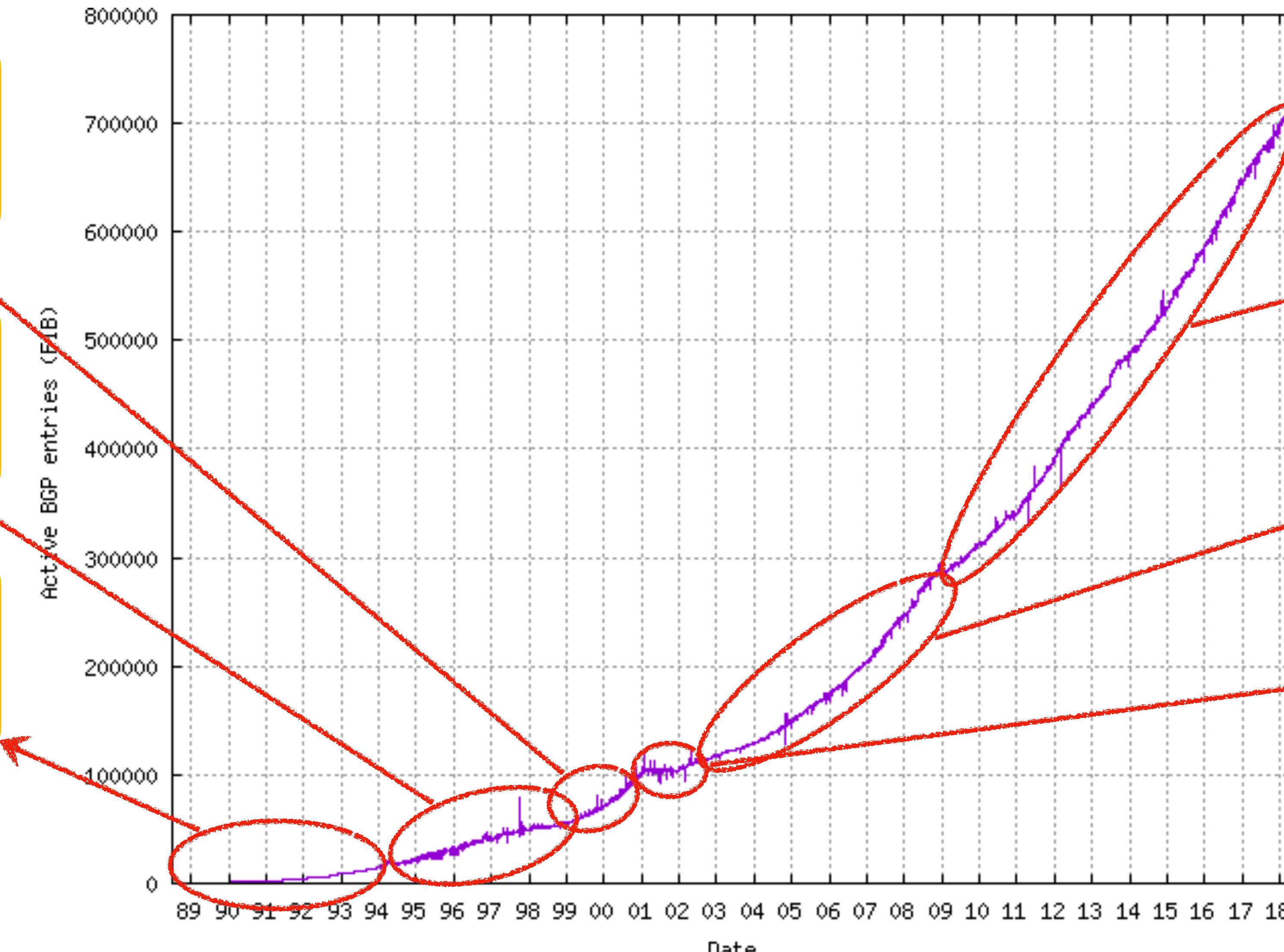
Advent of CIDR
allows aggregation:
linear growth

Initial growth
super-linear; no
aggregation

Stable linear
growth

Back in business

Dot-com
implosion: Internet
bubble bursts



Summary of Addressing

Summary of Addressing

- **Hierarchical** addressing
 - Critical for **scalable** system
 - Don't require everyone to know everyone else
 - Reduces amount of updating when something changes

Summary of Addressing

- **Hierarchical** addressing
 - Critical for **scalable** system
 - Don't require everyone to know everyone else
 - Reduces amount of updating when something changes
- **Non-uniform** hierarchy
 - Useful for heterogenous networks of different sizes
 - Class-based addressing was far too course
 - Classless InterDomain Routing (CIDR) more flexible

Summary of Addressing

- **Hierarchical** addressing
 - Critical for **scalable** system
 - Don't require everyone to know everyone else
 - Reduces amount of updating when something changes
- **Non-uniform** hierarchy
 - Useful for heterogenous networks of different sizes
 - Class-based addressing was far too course
 - Classless InterDomain Routing (CIDR) more flexible
- A later lecture: impact of CIDR on router designs

Questions?

Outline

- Addressing
- BGP
 - **Today:** context & basic ideas
 - **Next lecture:** details and issues

BGP (Today)

BGP (Today)

- **The role of policy**
 - What we mean by it
 - Why we need it

BGP (Today)

- **The role of policy**
 - What we mean by it
 - Why we need it
- **Overall Approach**
 - Four non-trivial changes to DV
 - How policy is implemented (detail-free version)

Administrative Structure Shapes Interdomain Routing

Administrative Structure Shapes Interdomain Routing

- ASes want freedom to pick routes based on **policy**

Administrative Structure Shapes Interdomain Routing

- ASes want freedom to pick routes based on **policy**
- ASes want **autonomy**

Administrative Structure Shapes Interdomain Routing

- ASes want freedom to pick routes based on **policy**
- ASes want **autonomy**
- ASes want **privacy**

Topology and Policy is shaped by business relationships between ASes

Topology and Policy is shaped by business relationships between ASes

- **Three basic kinds of relationships between ASes**

Topology and Policy is shaped by business relationships between ASes

- **Three basic kinds of relationships between ASes**
 - AS A can be AS B's customer

Topology and Policy is shaped by business relationships between ASes

- **Three basic kinds of relationships between ASes**
 - AS A can be AS B's *customer*
 - AS A can be AS B's *provider*

Topology and Policy is shaped by business relationships between ASes

- **Three basic kinds of relationships between ASes**
 - AS A can be AS B's *customer*
 - AS A can be AS B's *provider*
 - AS A can be AS B's *peer*

Topology and Policy is shaped by business relationships between ASes

- **Three basic kinds of relationships between ASes**
 - AS A can be AS B's customer
 - AS A can be AS B's provider
 - AS A can be AS B's peer
- **Business implications**

Topology and Policy is shaped by business relationships between ASes

- **Three basic kinds of relationships between ASes**
 - AS A can be AS B's customer
 - AS A can be AS B's provider
 - AS A can be AS B's peer
- **Business implications**
 - Customer pays provider

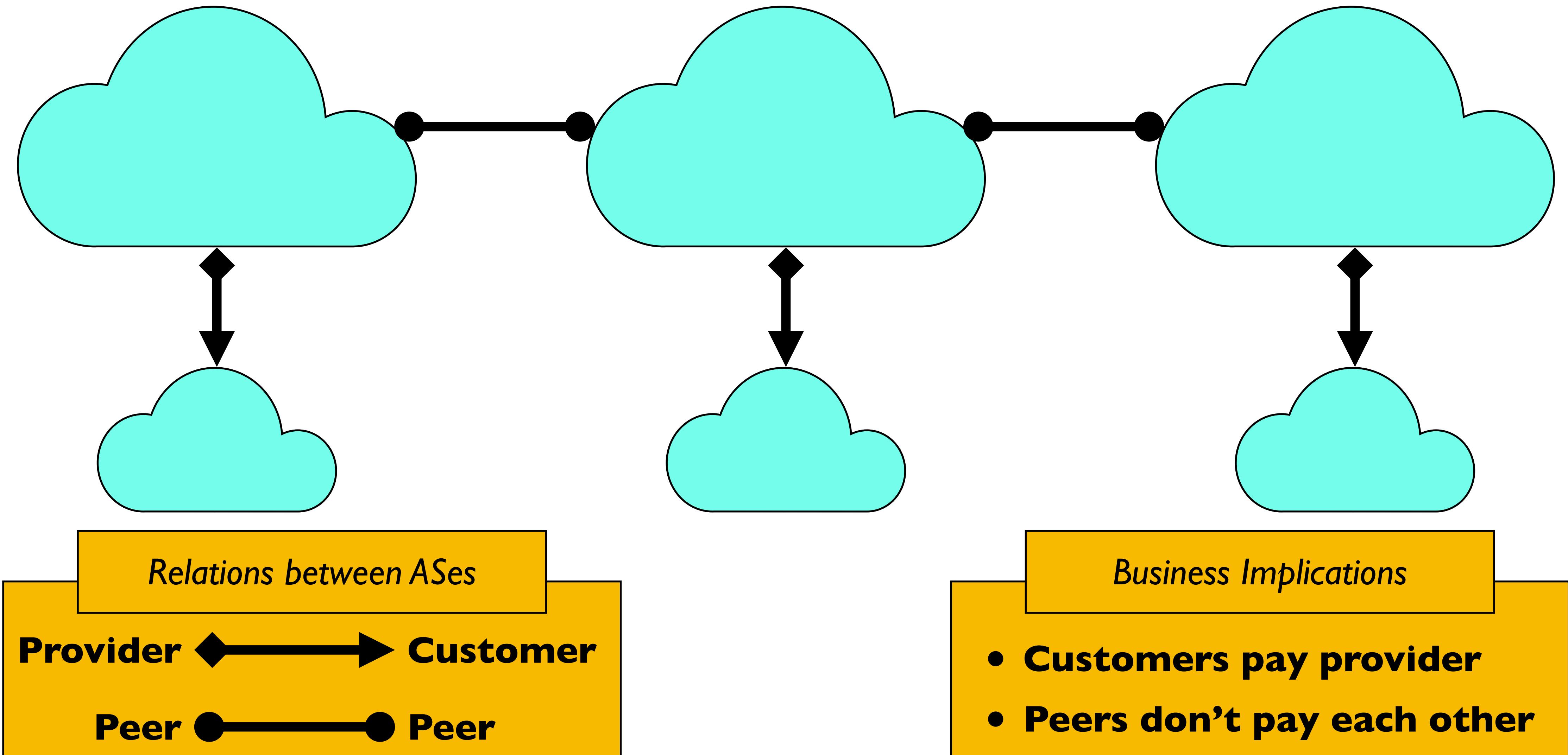
Topology and Policy is shaped by business relationships between ASes

- **Three basic kinds of relationships between ASes**
 - AS A can be AS B's customer
 - AS A can be AS B's provider
 - AS A can be AS B's peer
- **Business implications**
 - Customer pays provider
 - Peers don't pay each other

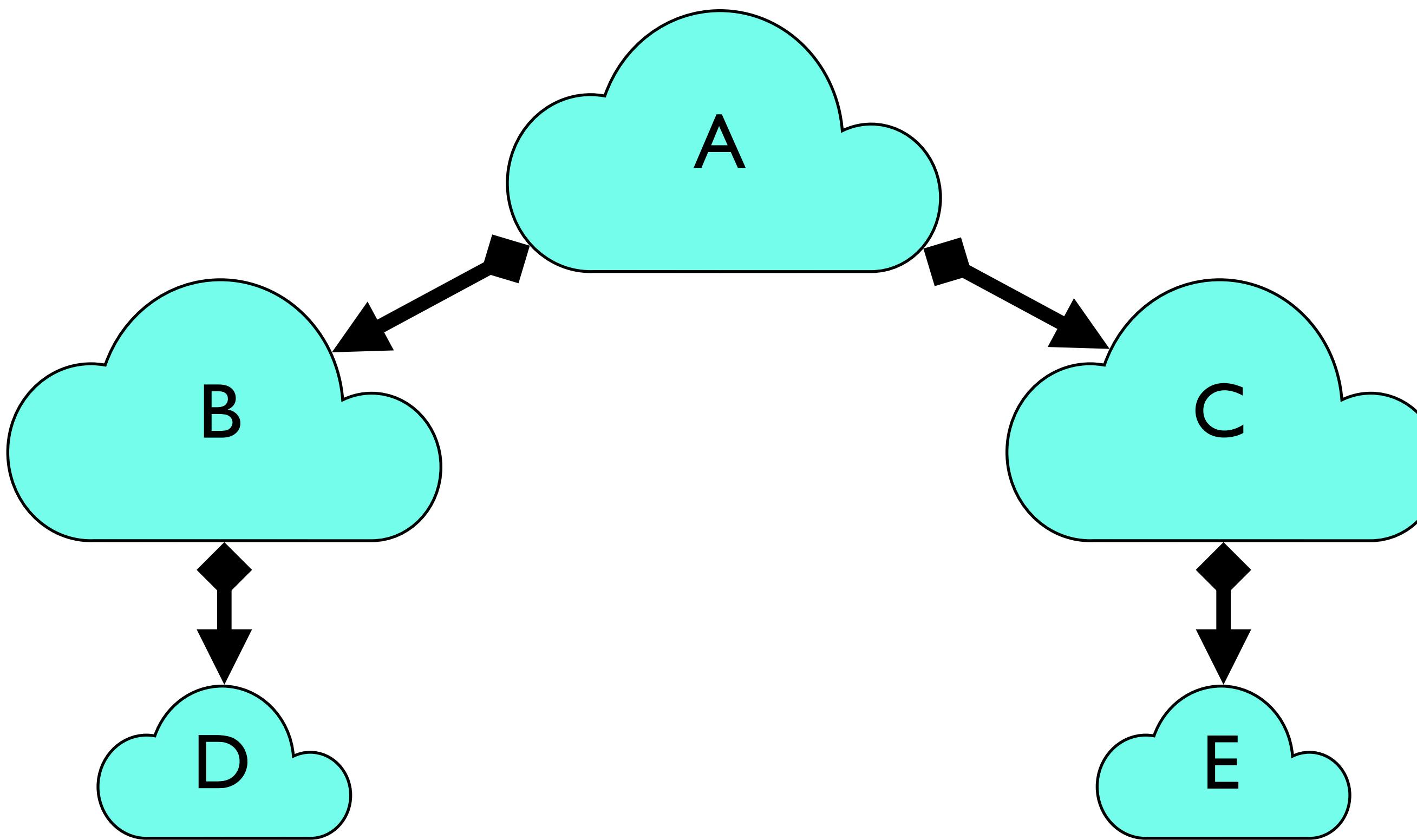
Topology and Policy is shaped by business relationships between ASes

- **Three basic kinds of relationships between ASes**
 - AS A can be AS B's customer
 - AS A can be AS B's provider
 - AS A can be AS B's peer
- **Business implications**
 - Customer pays provider
 - Peers don't pay each other
 - Exchange roughly equal traffic

Business Relationships



Why Peer?



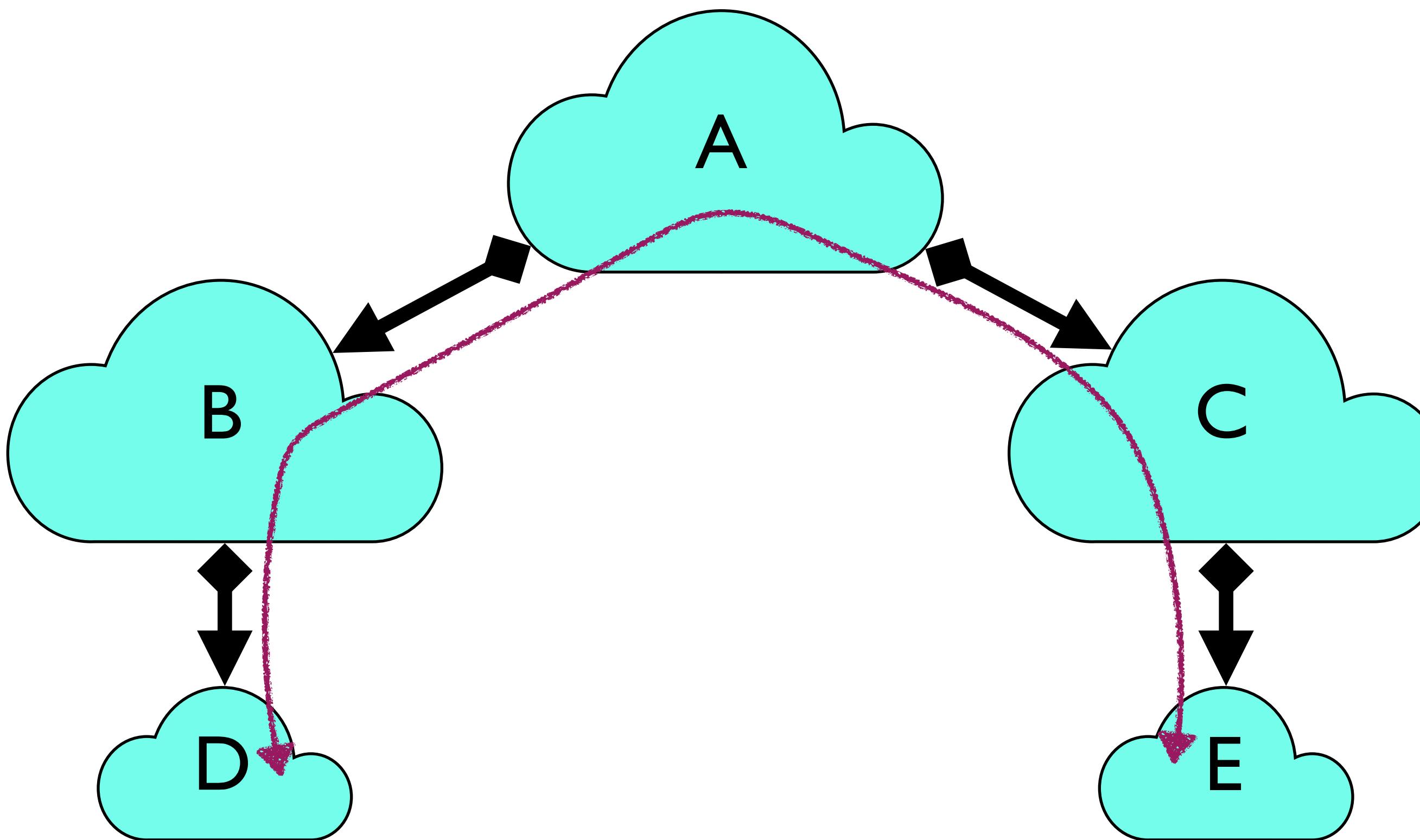
Relations between ASes

Provider ←→ **Customer**
Peer —●— Peer

Business Implications

- **Customers pay provider**
- **Peers don't pay each other**

Why Peer?



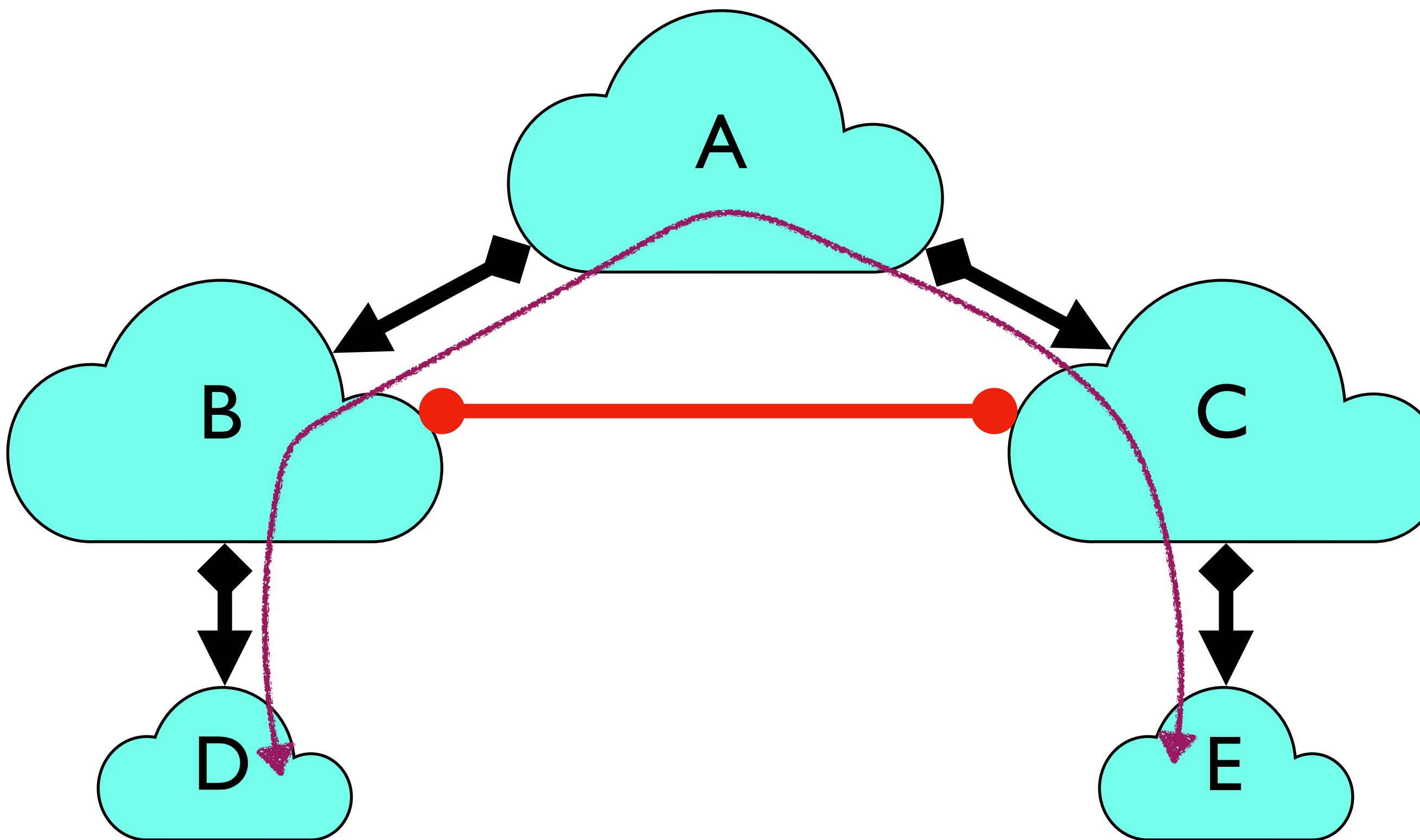
Relations between ASes

Provider ←→ **Customer**
Peer —————●— Peer

Business Implications

- **Customers pay provider**
- **Peers don't pay each other**

Why Peer?



Relations between ASes

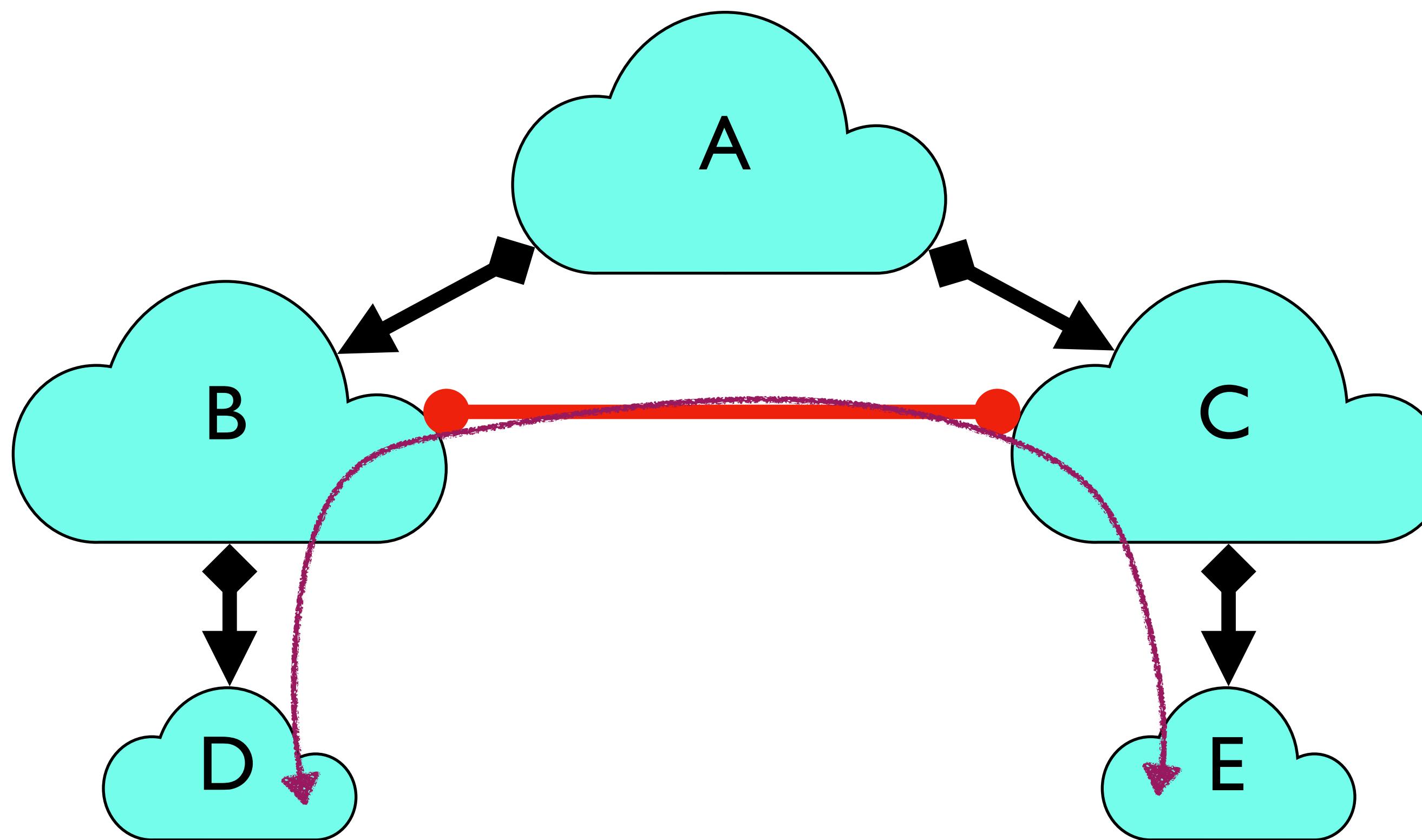
Provider ←→ **Customer**
Peer —————●— Peer

Business Implications

- **Customers pay provider**
- **Peers don't pay each other**

E.g. D & E talk a lot

Why Peer?



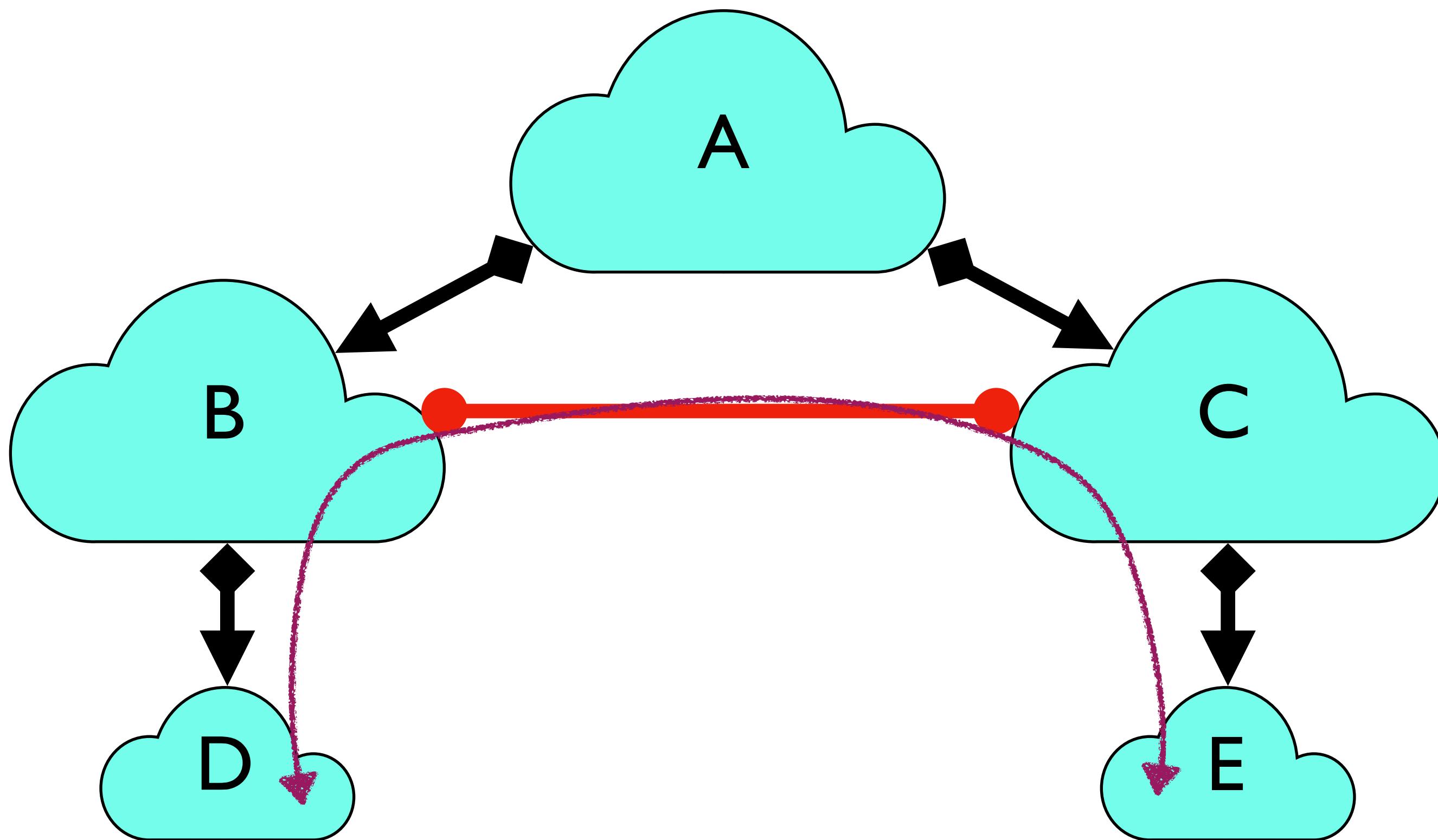
Relations between ASes

Provider ←→ **Customer**
Peer —●— Peer

Business Implications

- **Customers pay provider**
- **Peers don't pay each other**

Why Peer?



Relations between ASes

Provider ←→ **Customer**
Peer —●— Peer

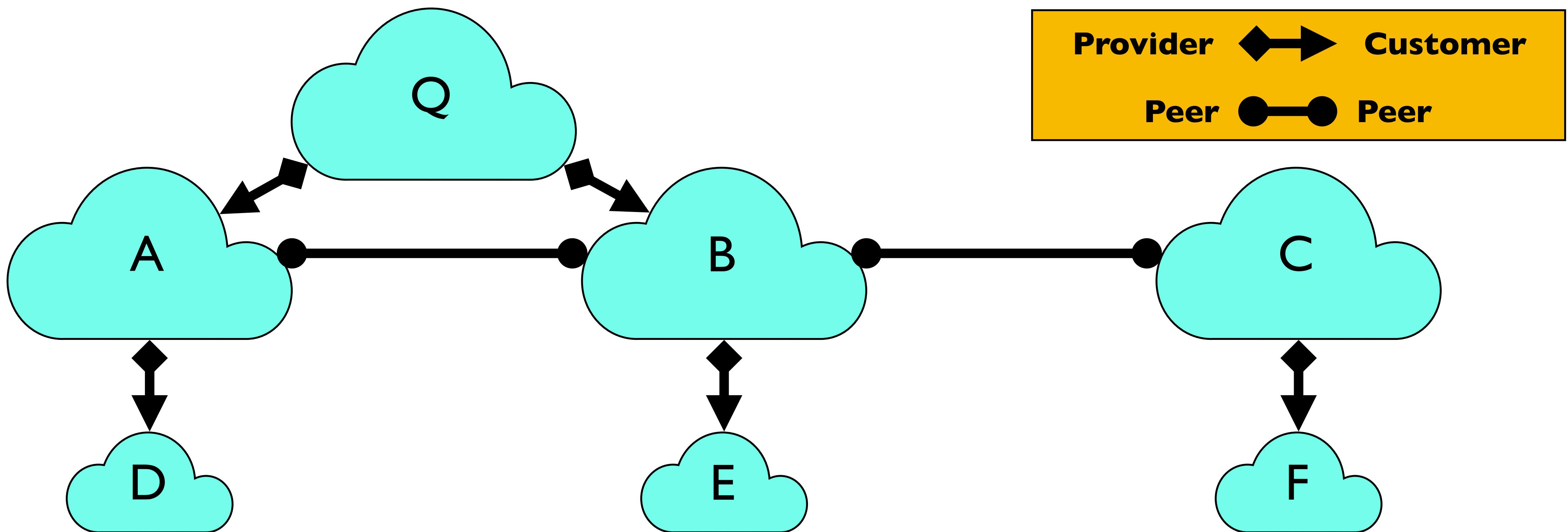
E.g. D & E talk a lot

Peering saves B and C money!

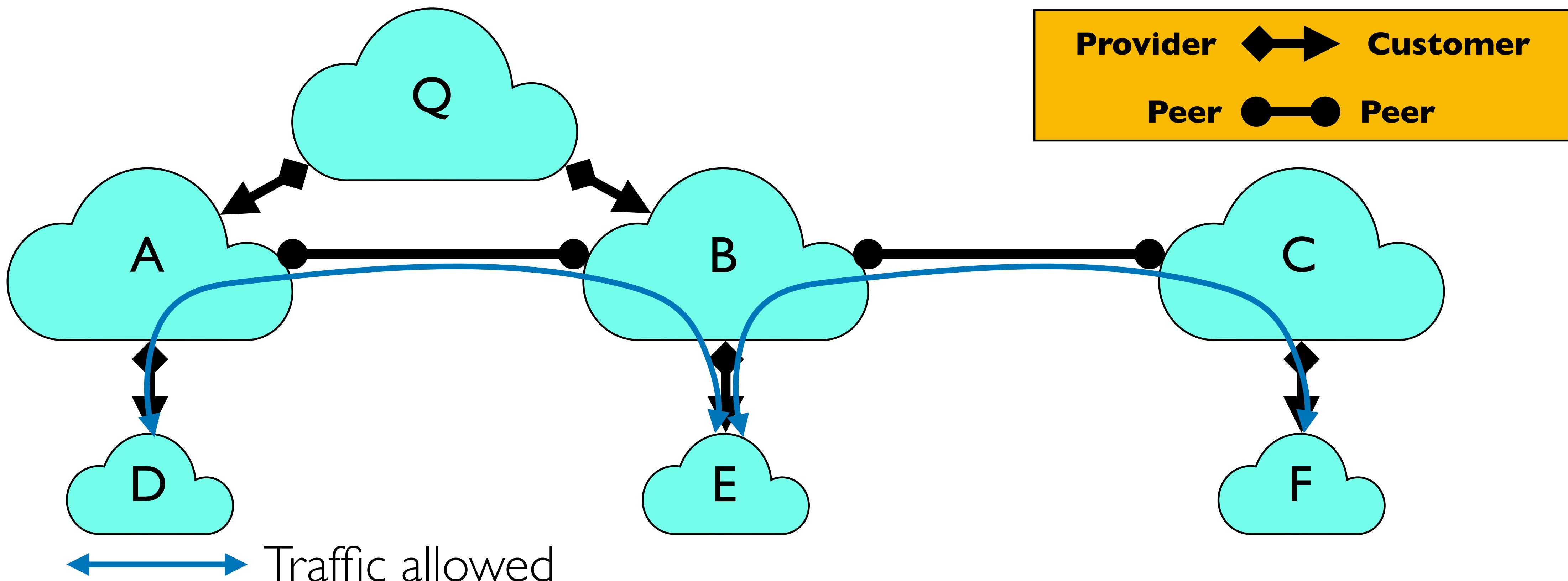
Business Implications

- **Customers pay provider**
- **Peers don't pay each other**

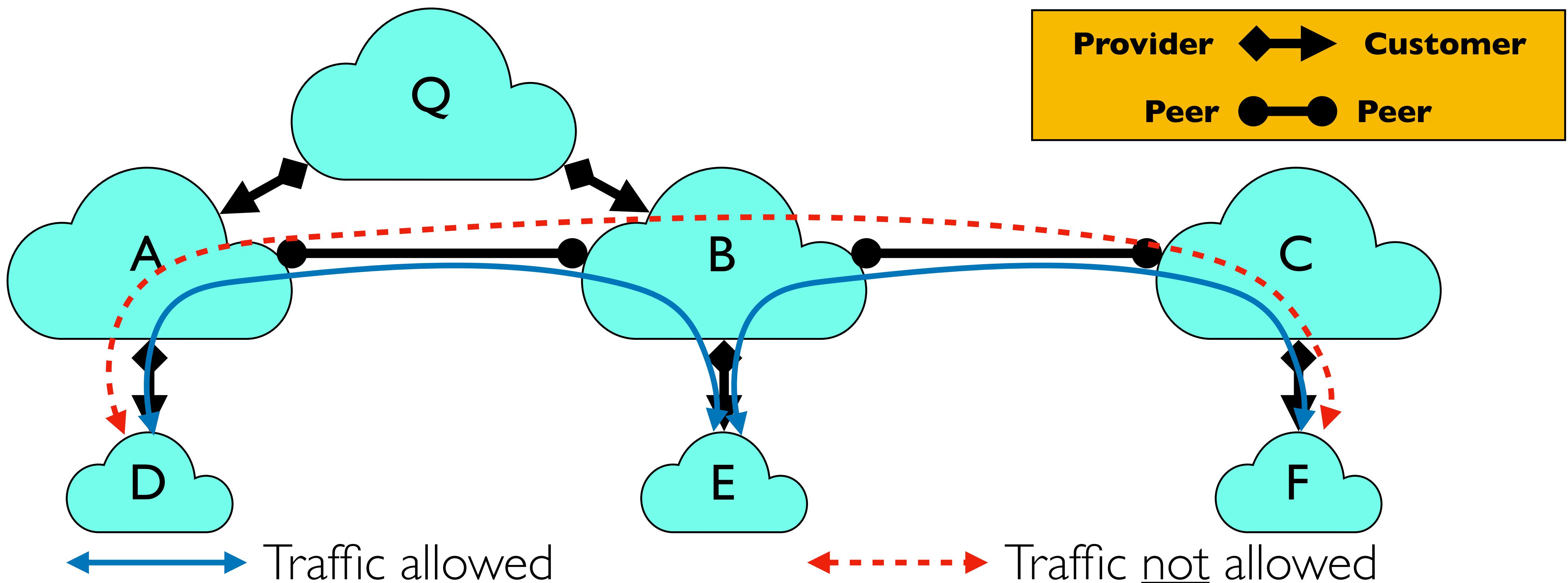
Routing Follows the Money!



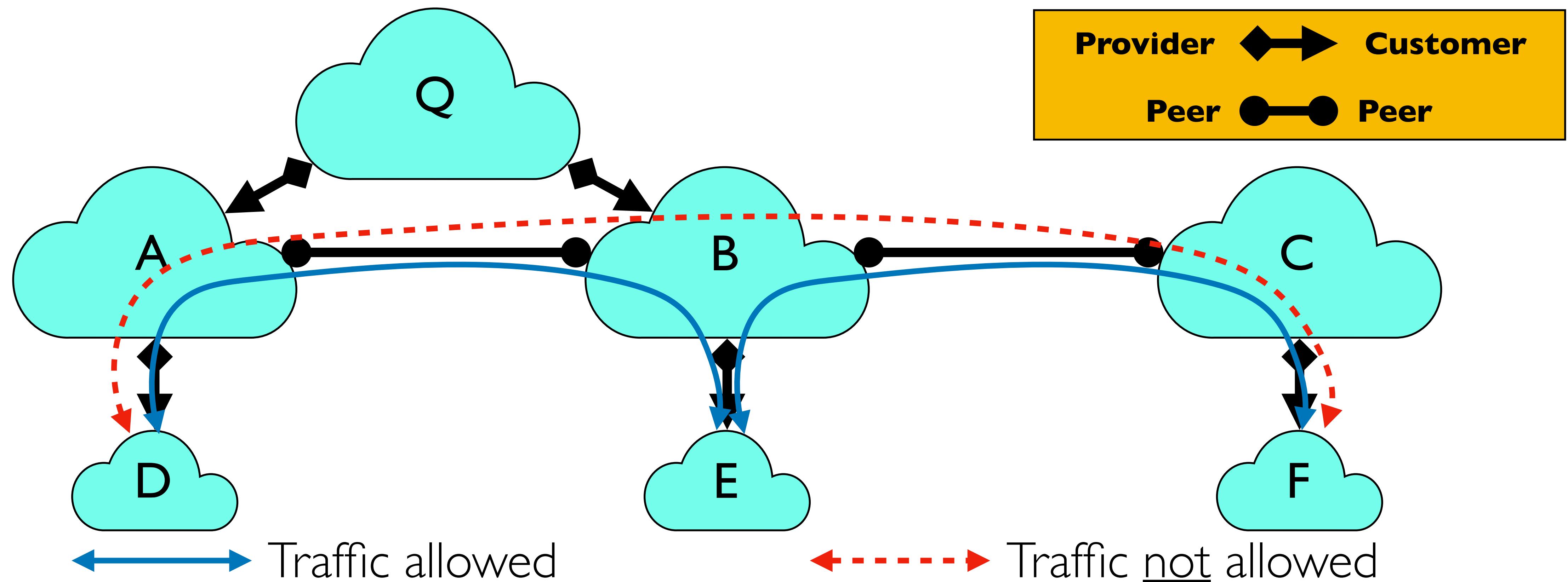
Routing Follows the Money!



Routing Follows the Money!

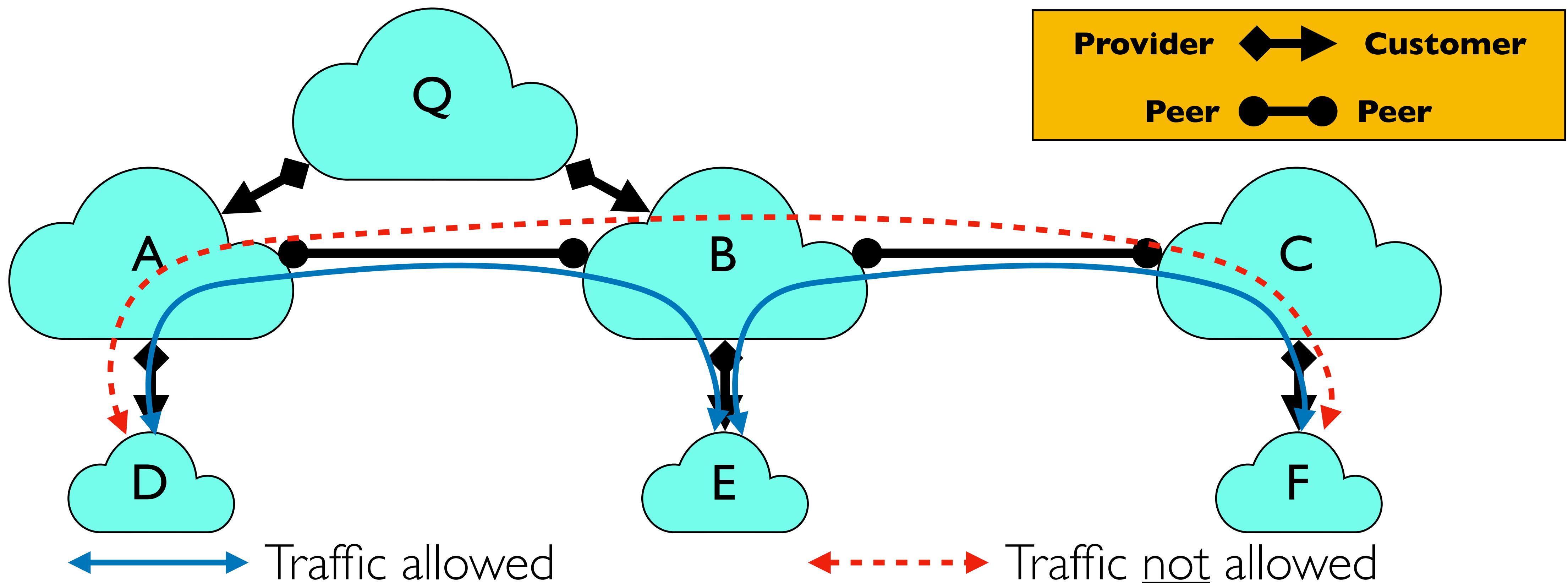


Routing Follows the Money!



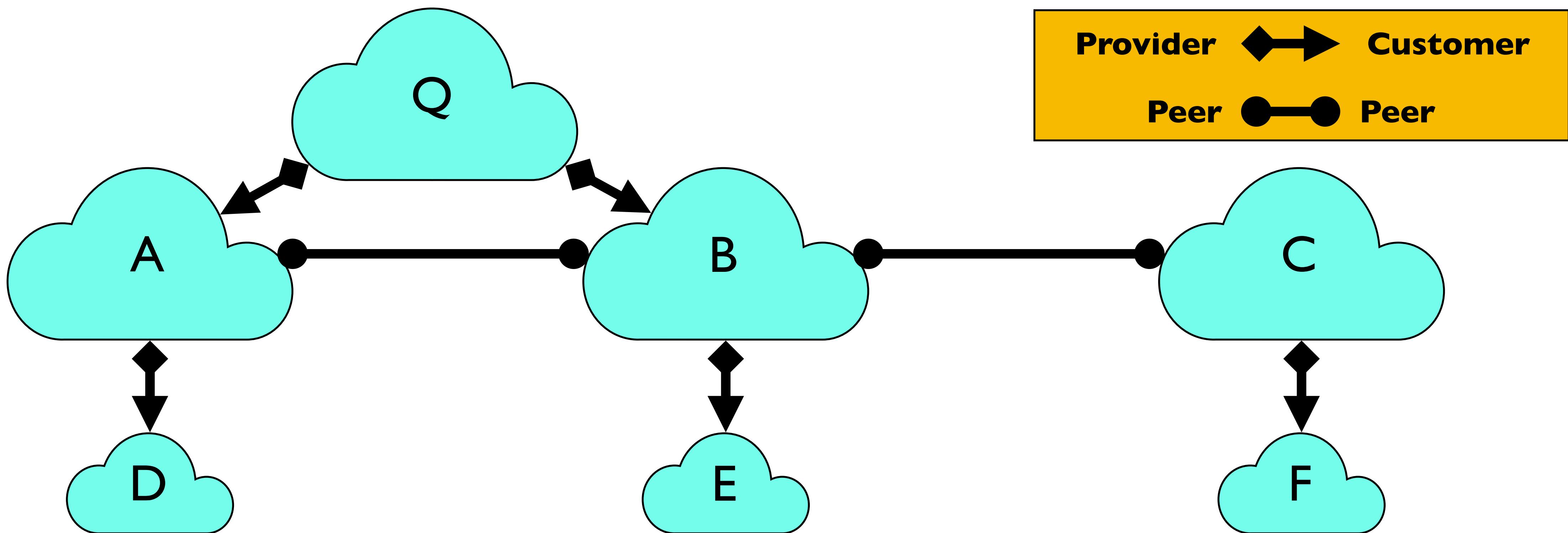
- ASes provide “transit” between their customers

Routing Follows the Money!

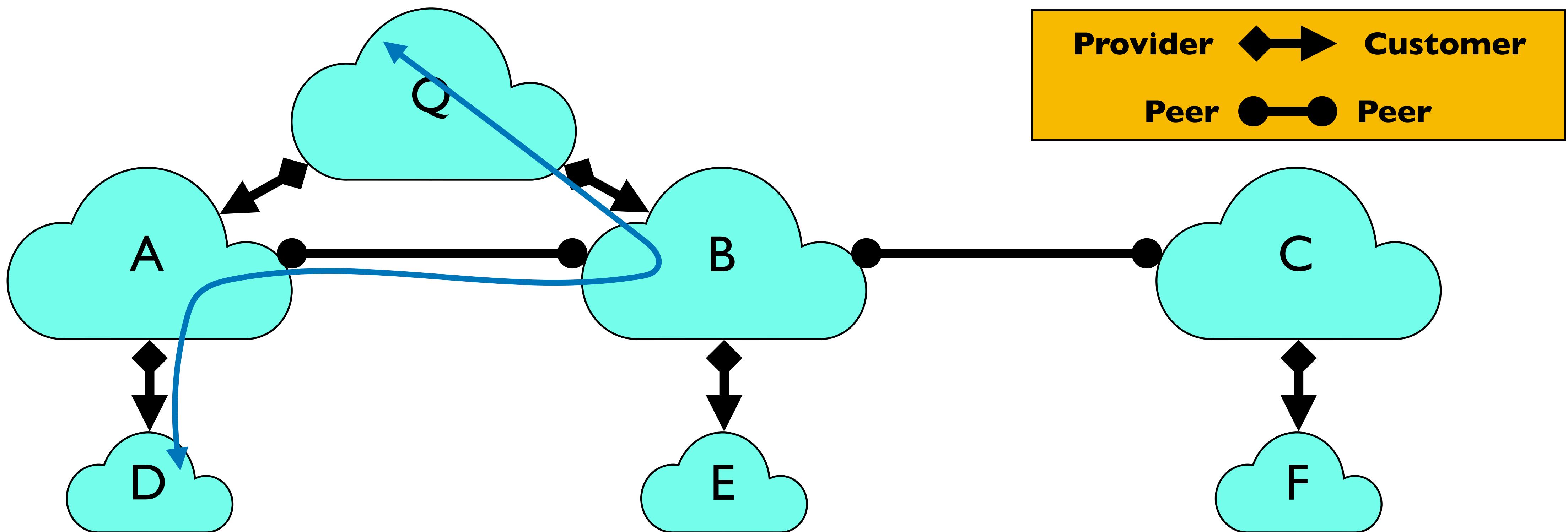


- ASes provide “transit” between their customers
- Peers do not provide transit between other peers

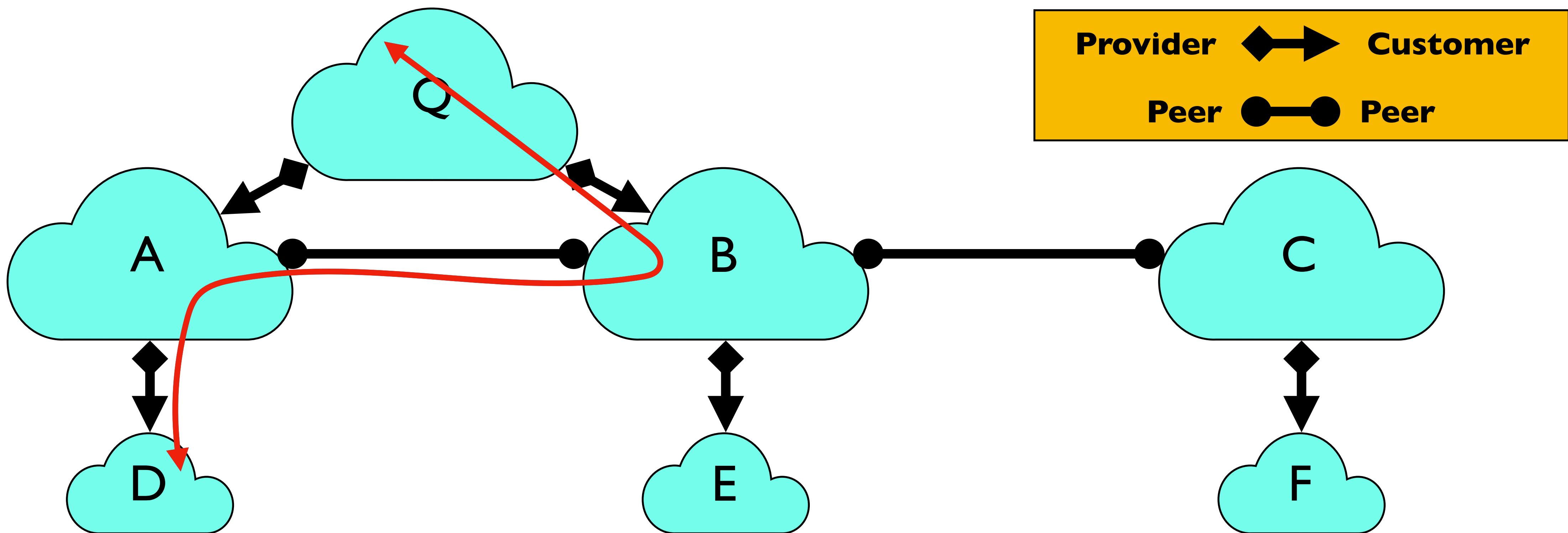
Routing Follows the Money!



Routing Follows the Money!

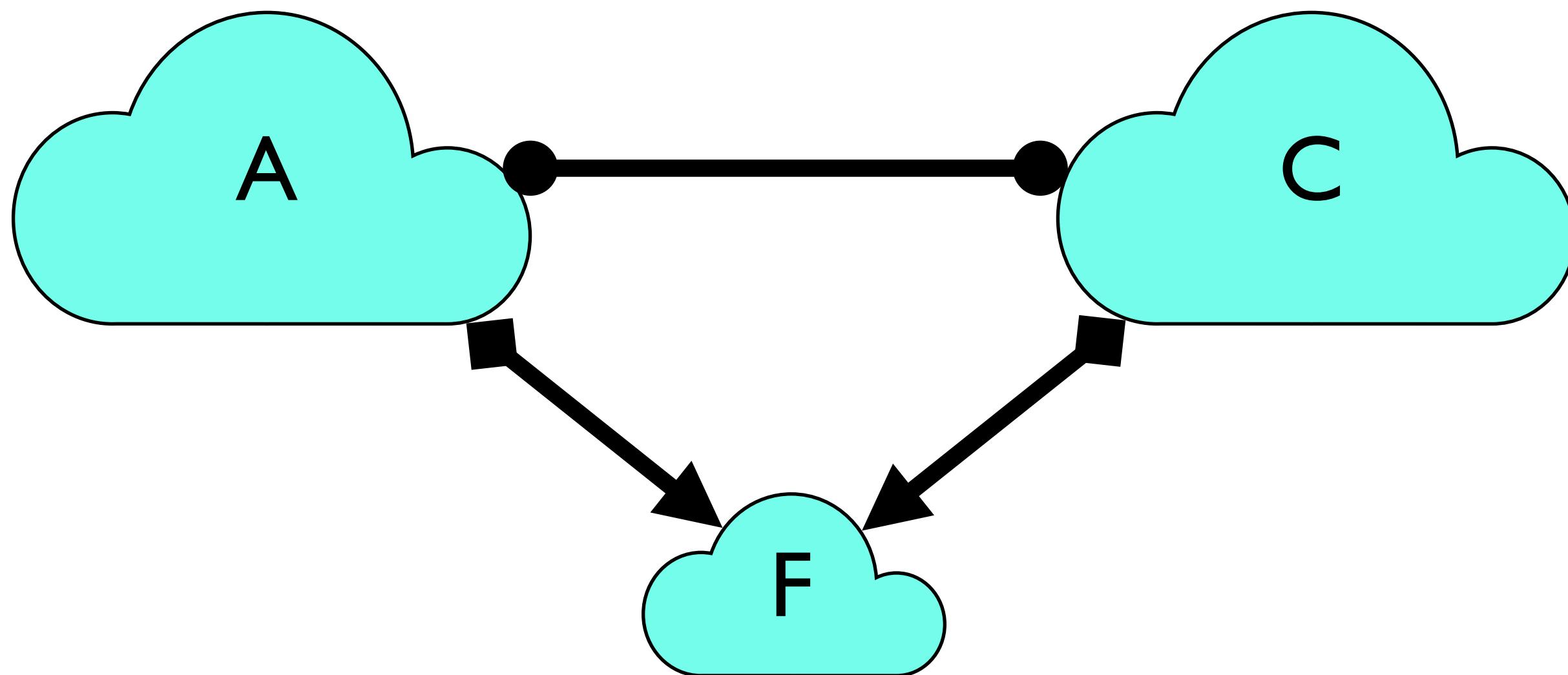
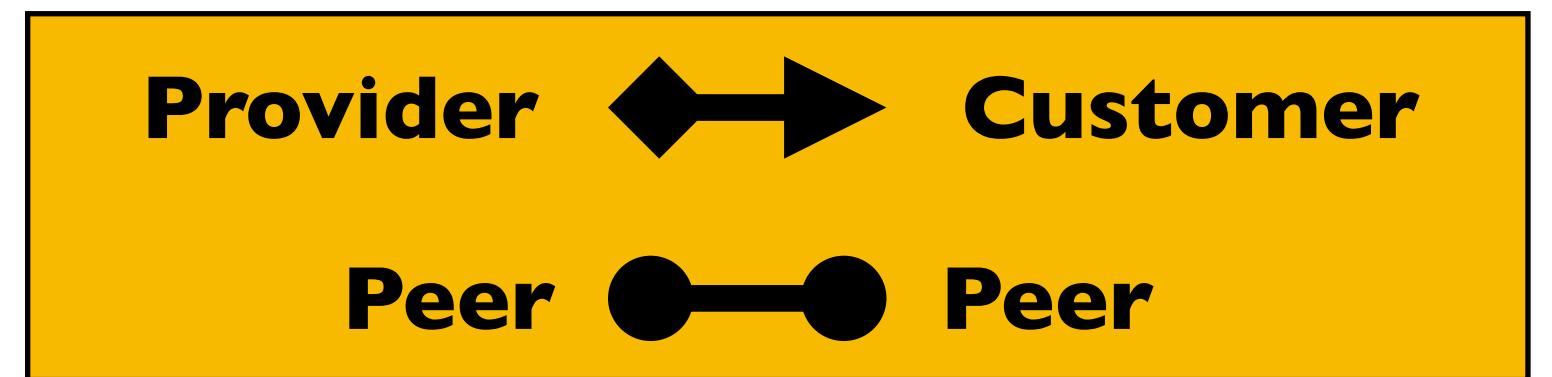


Routing Follows the Money!

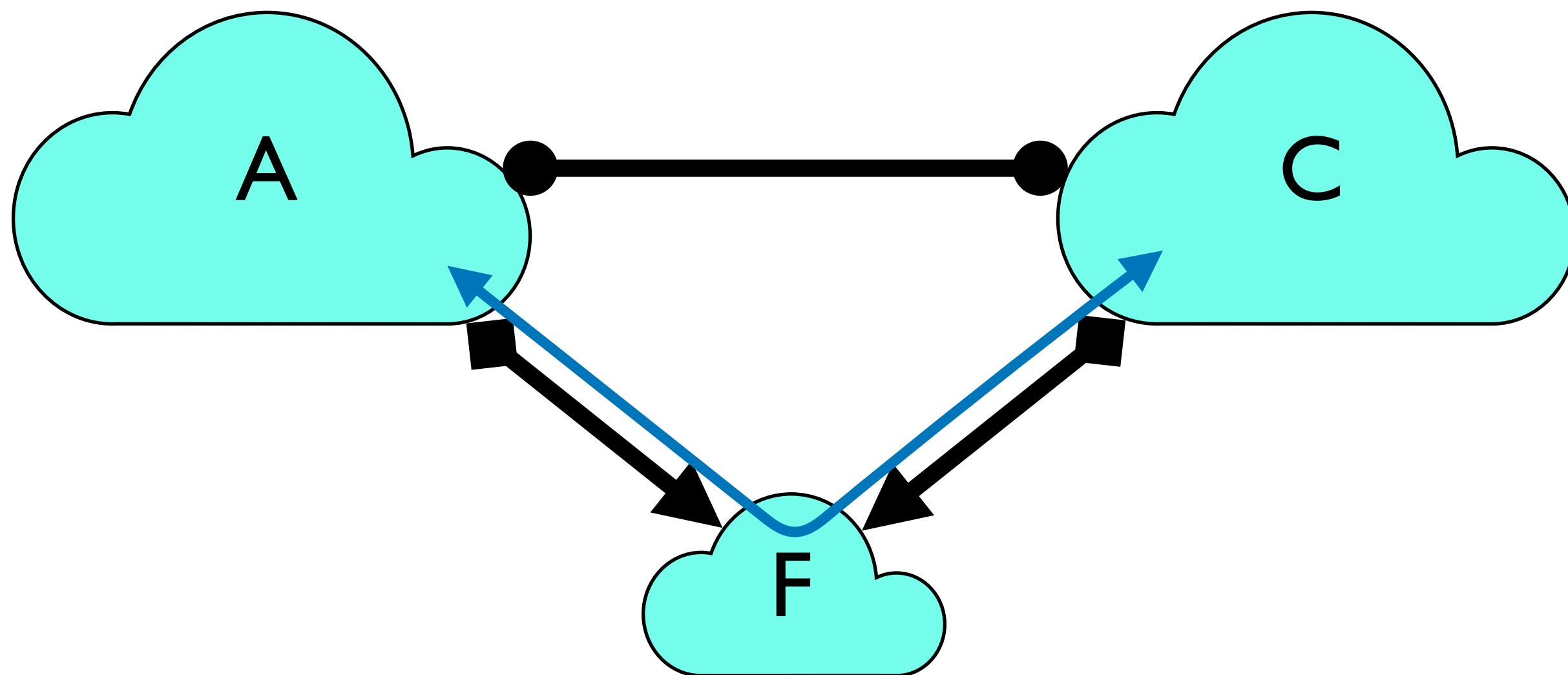
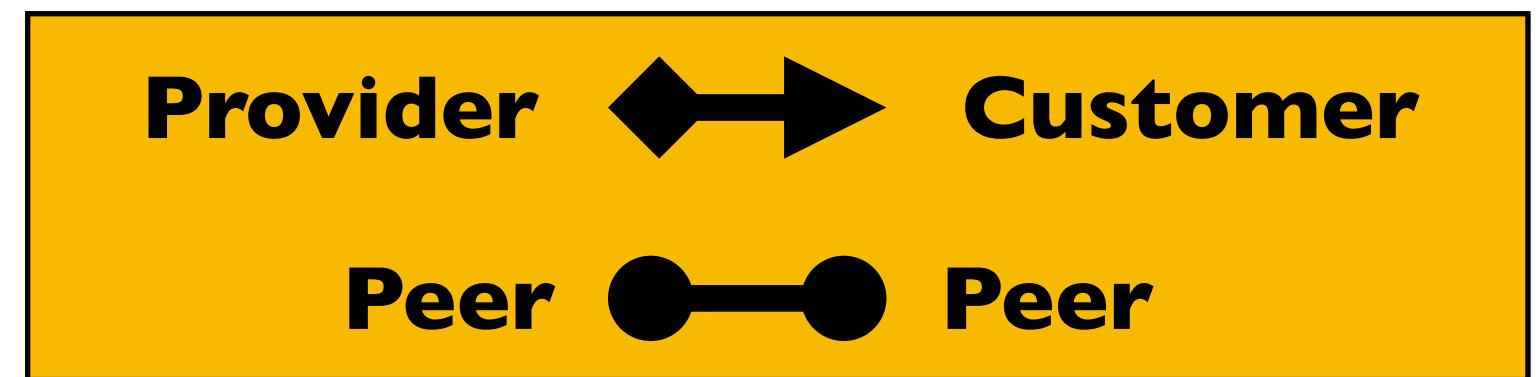


- An AS only carries traffic to/from its own customers over a peering link

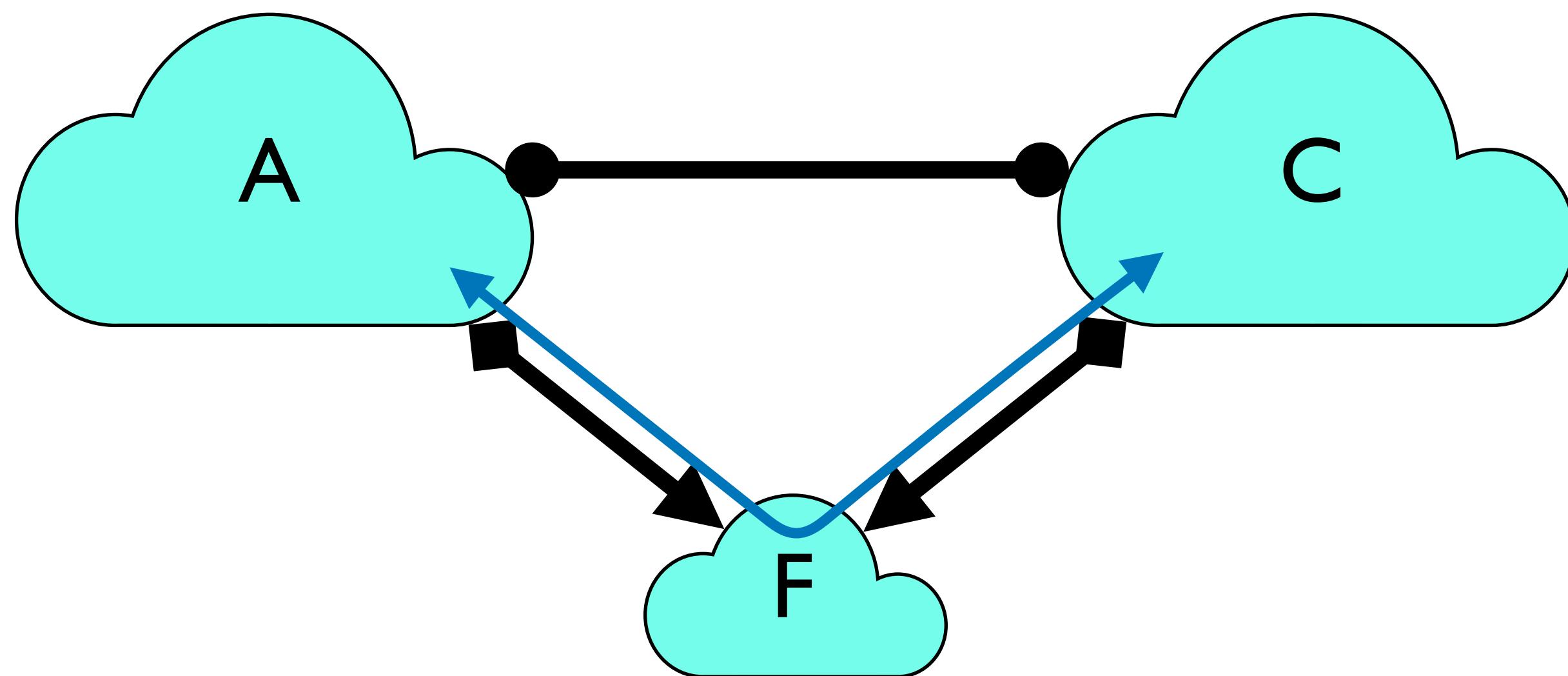
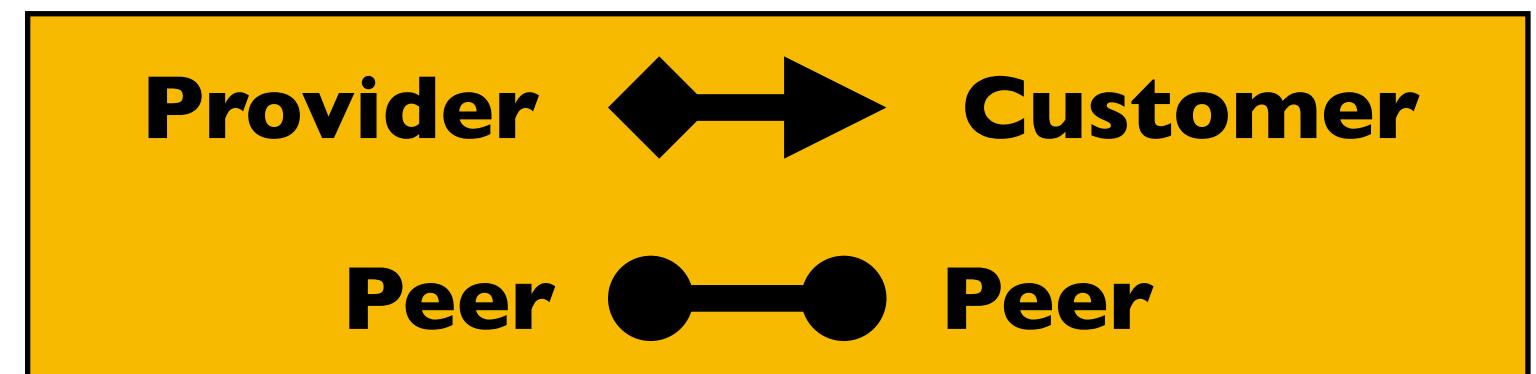
Routing Follows the Money!



Routing Follows the Money!

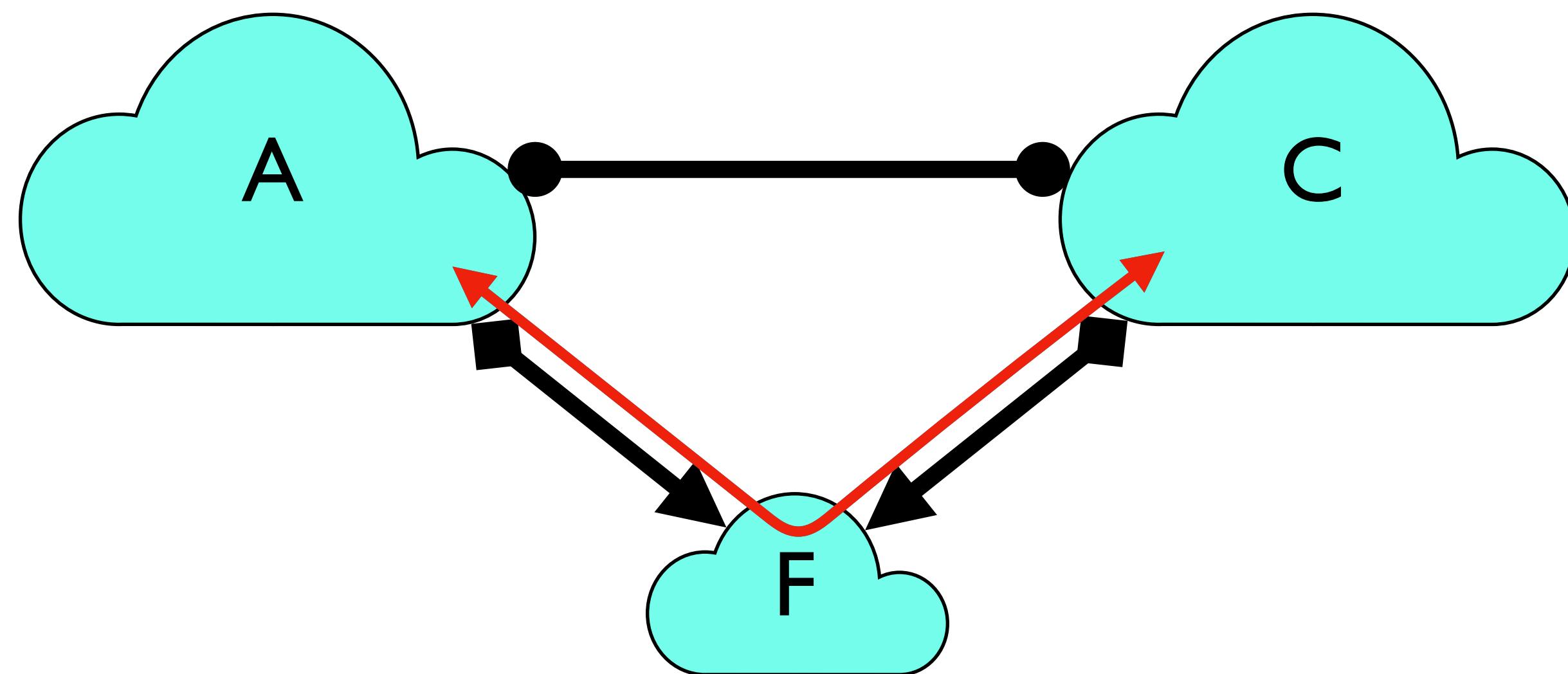
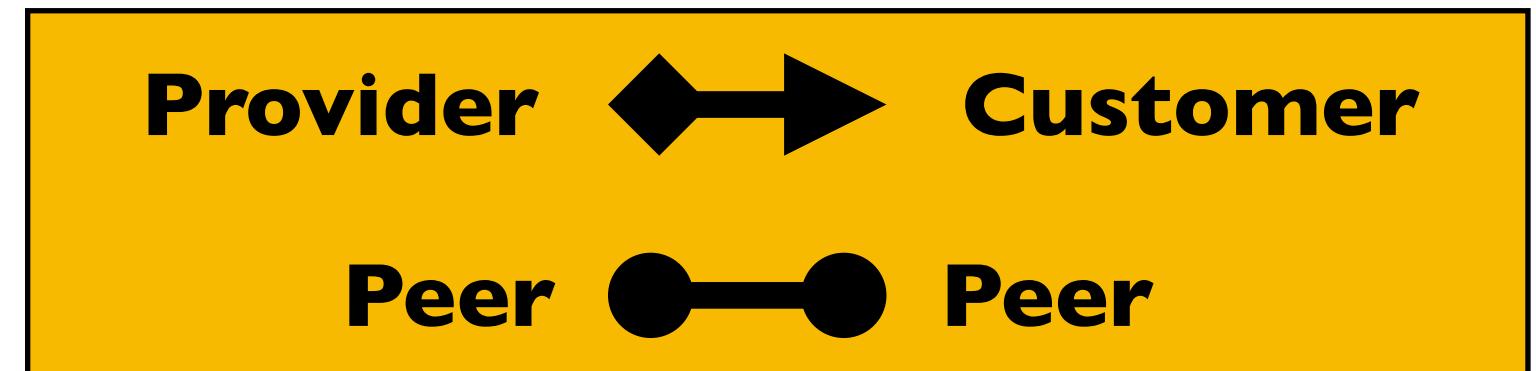


Routing Follows the Money!



- Routes are “valley free” (will return to this later)

Routing Follows the Money!



- Routes are “valley free” (will return to this later)

In Short

In Short

- AS topology reflects business relationships between ASes

In Short

- AS topology reflects business relationships between ASes
- Business relationships between ASes impact which routes are acceptable

In Short

- AS topology reflects business relationships between ASes
- Business relationships between ASes impact which routes are acceptable
- **BGP Policy:** Protocol design that allows ASes to control which routes are used

In Short

- AS topology reflects business relationships between ASes
- Business relationships between ASes impact which routes are acceptable
- **BGP Policy:** Protocol design that allows ASes to control which routes are used
- **Next lecture:** More formal analysis of the impact of policy and route stability

Questions?

BGP (Today)

- **The role of policy**
 - What we mean by it
 - Why we need it
- **Overall Approach**
 - Four non-trivial changes to DV
 - How policy is implemented (detail-free version)

Interdomain Routing: Setup

Interdomain Routing: Setup

- Destinations are IP prefixes ($12.0.0.0/8$)

Interdomain Routing: Setup

- Destinations are IP prefixes ($12.0.0.0/8$)
- Nodes are Autonomous Systems (ASes)

Interdomain Routing: Setup

- Destinations are IP prefixes ($12.0.0.0/8$)
- Nodes are Autonomous Systems (ASes)
 - Internals of each AS are hidden

Interdomain Routing: Setup

- Destinations are IP prefixes ($12.0.0.0/8$)
- Nodes are Autonomous Systems (ASes)
 - Internals of each AS are hidden
- Links represent both physical links and business relationships

Interdomain Routing: Setup

- Destinations are IP prefixes ($12.0.0.0/8$)
- Nodes are Autonomous Systems (ASes)
 - Internals of each AS are hidden
- Links represent both physical links and business relationships
- Border Gateway Protocol (BGP) is the Interdomain routing protocol

Interdomain Routing: Setup

- Destinations are IP prefixes ($12.0.0.0/8$)
- Nodes are Autonomous Systems (ASes)
 - Internals of each AS are hidden
- Links represent both physical links and business relationships
- Border Gateway Protocol (BGP) is the Interdomain routing protocol
 - Implemented by AS border routers

BGP: Basic Idea

BGP: Basic Idea

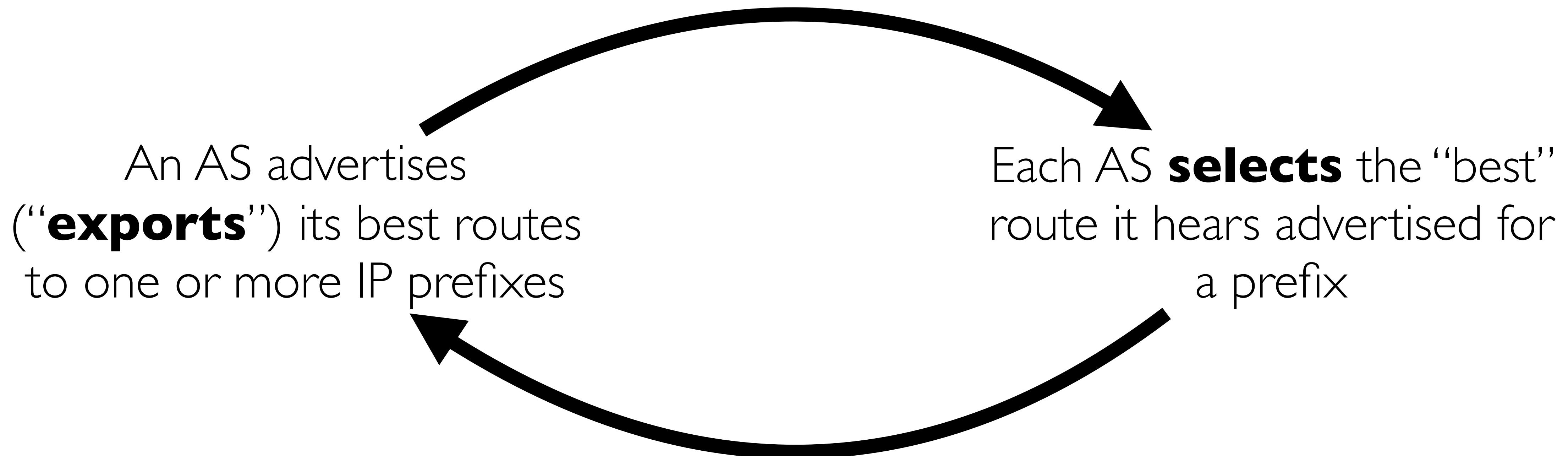
An AS advertises
("“**exports**”") its best routes
to one or more IP prefixes

BGP: Basic Idea

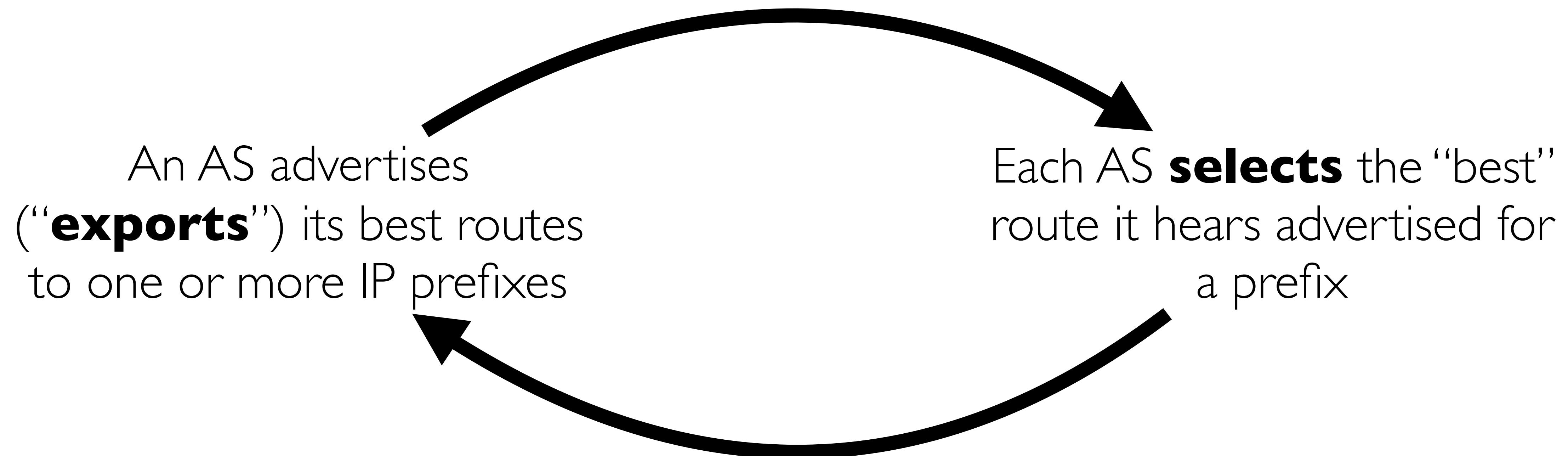
An AS advertises (“**exports**”) its best routes to one or more IP prefixes

Each AS **selects** the “best” route it hears advertised for a prefix

BGP: Basic Idea



BGP: Basic Idea



You’ve heard this story before!

BGP inspired by Distance Vector

BGP inspired by Distance Vector

- Per-destination route advertisements

BGP inspired by Distance Vector

- Per-destination route advertisements
- No global sharing of network topology information

BGP inspired by Distance Vector

- Per-destination route advertisements
- No global sharing of network topology information
- Iterative and distributed convergence on paths

BGP inspired by Distance Vector

- Per-destination route advertisements
- No global sharing of network topology information
- Iterative and distributed convergence on paths
- *With four crucial differences*

Differences between BGP and DV

(I) not picking the shortest path routes

Differences between BGP and DV

(I) not picking the shortest path routes

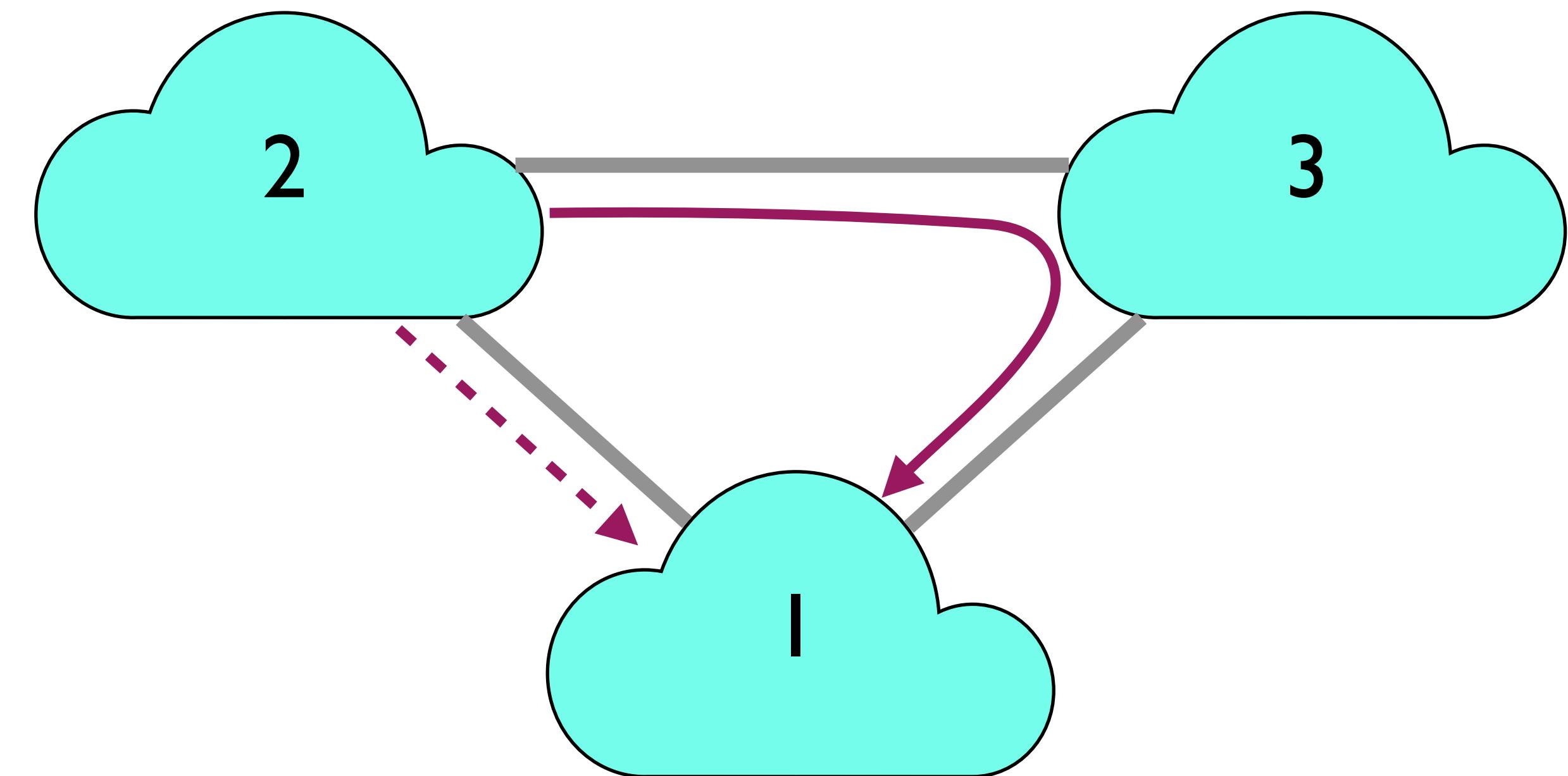
- BGP selects the east route based on policy, not shortest distance (least cost)

Differences between BGP and DV

(I) not picking the shortest path routes

- BGP selects the east route based on policy, not shortest distance (least cost)

**Node 2 may prefer
“2, 3, 1” over “2, 1”**

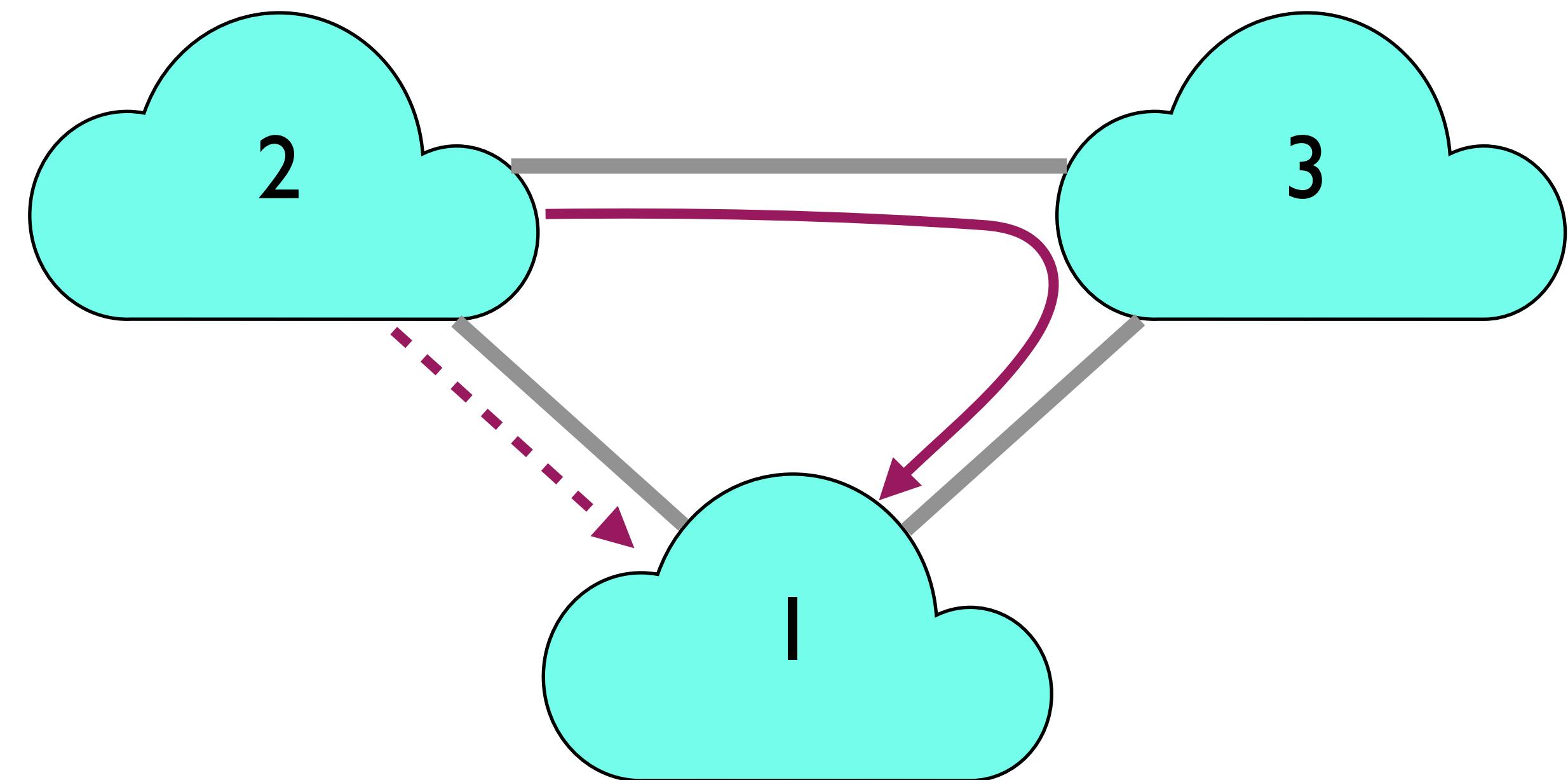


Differences between BGP and DV

(I) not picking the shortest path routes

- BGP selects the east route based on policy, not shortest distance (least cost)

**Node 2 may prefer
“2, 3, 1” over “2, 1”**



- How do we avoid loops?

Differences between BGP and DV

(2) path-vector routing

Differences between BGP and DV

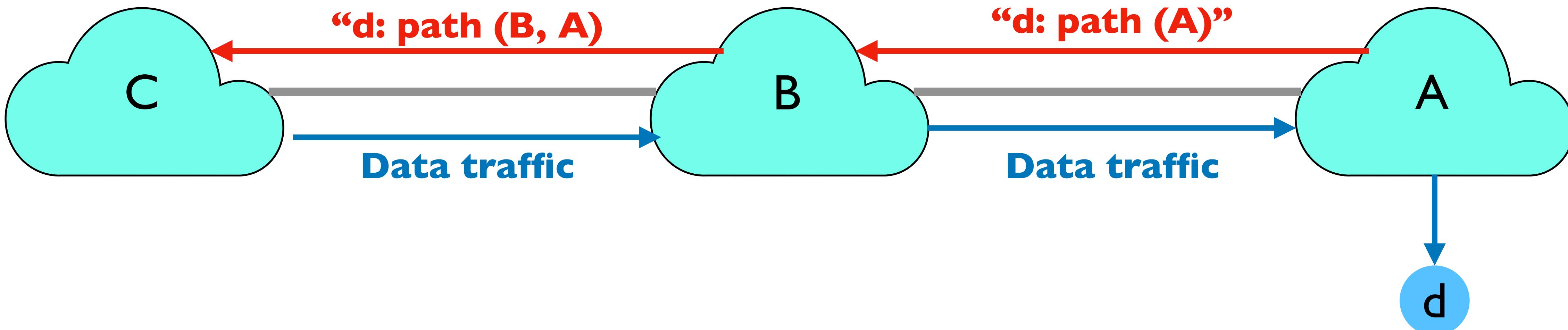
(2) path-vector routing

- **Key idea:** advertise the entire path
 - Distance vector: send distance metric per destination d
 - Path vector: send the entire path for each destination d

Differences between BGP and DV

(2) path-vector routing

- **Key idea:** advertise the entire path
 - Distance vector: send distance metric per destination d
 - Path vector: send the entire path for each destination d



Differences between BGP and DV

(2) path-vector routing

- **Key idea:** advertise the entire path
 - Distance vector: send distance metric per destination d
 - Path vector: send the entire path for each destination d
- **Benefits**
 - Loop avoidance is easy

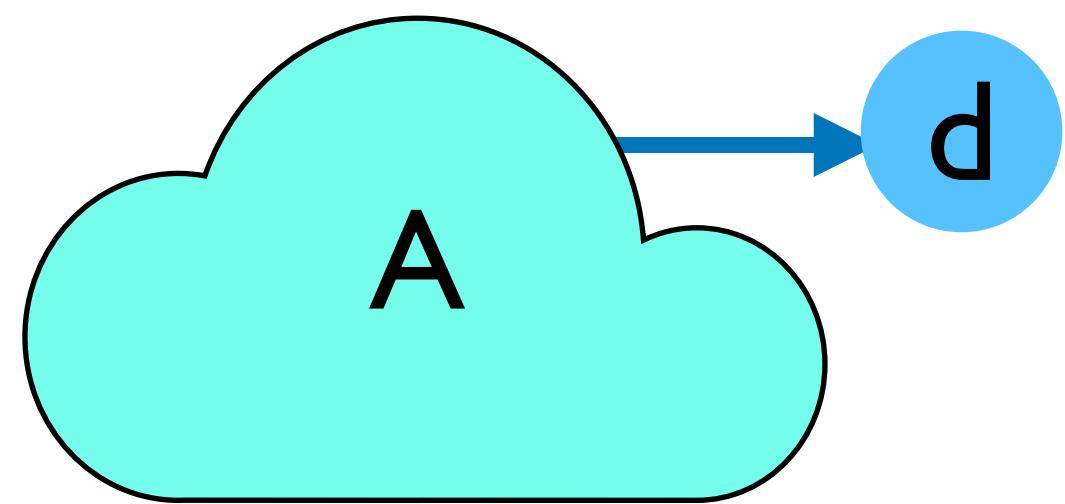
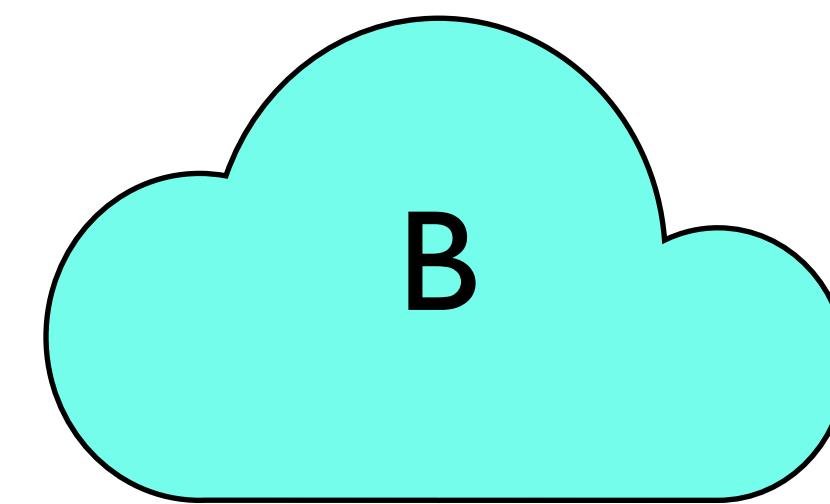
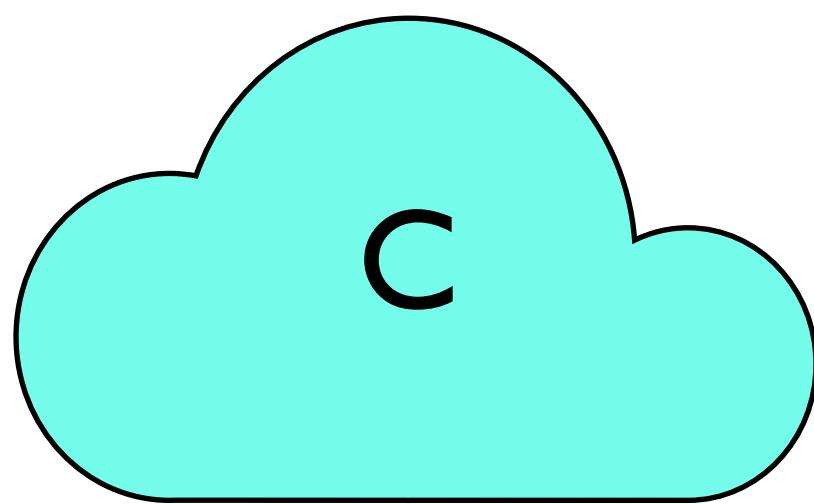
Loop Detection with Path-Vector

Loop Detection with Path-Vector

- Node can easily detect a loop
 - Look for its own node identifier in the path

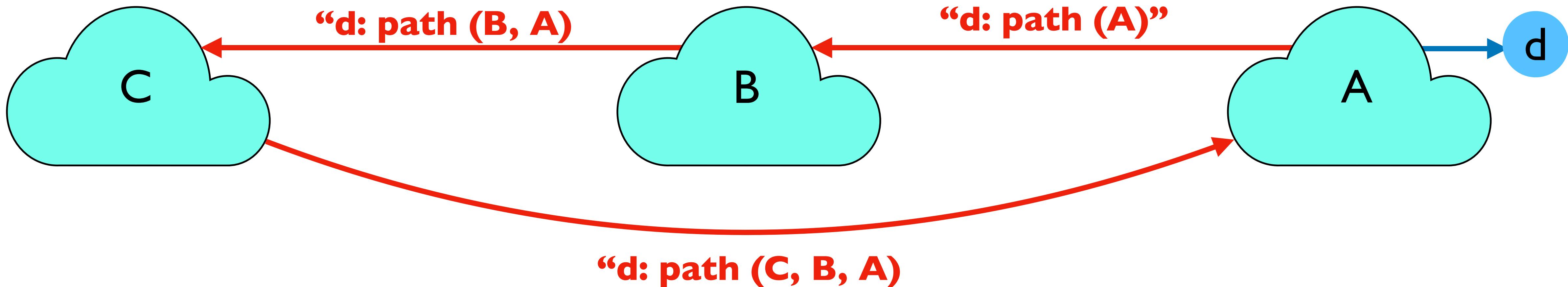
Loop Detection with Path-Vector

- Node can easily detect a loop
 - Look for its own node identifier in the path



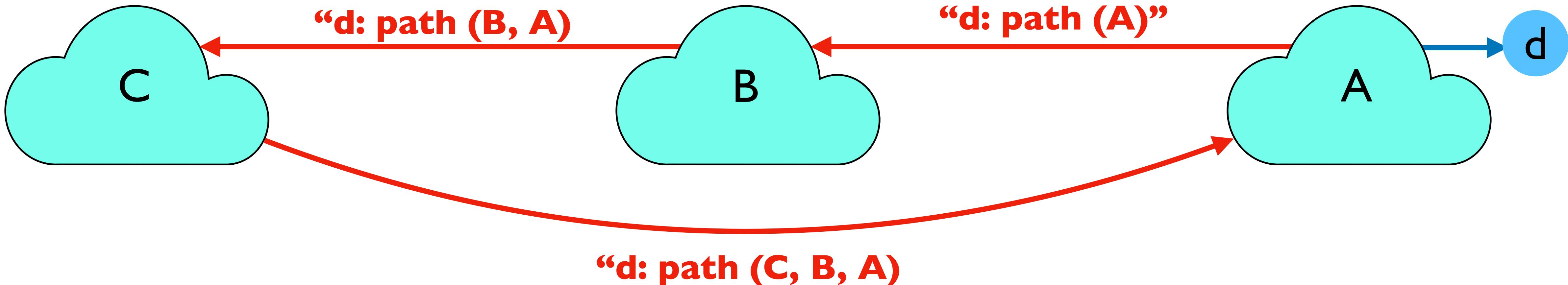
Loop Detection with Path-Vector

- Node can easily detect a loop
 - Look for its own node identifier in the path



Loop Detection with Path-Vector

- Node can easily detect a loop
 - Look for its own node identifier in the path
 - Node can simply discard paths with loops
 - E.g., node A sees itself in path “C, B, A” & discards the advertisement



Differences between BGP and DV

(2) path-vector routing

- **Key idea:** advertise the entire path
 - Distance vector: send distance metric per destination d
 - Path vector: send the entire path for each destination d
- **Benefits**
 - Loop avoidance is easy
 - Flexible policies based on entire path

Differences between BGP and DV

(3) selective route advertisement

Differences between BGP and DV

(3) selective route advertisement

- For policy reasons, an AS may choose not to advertise a route to a destination

Differences between BGP and DV

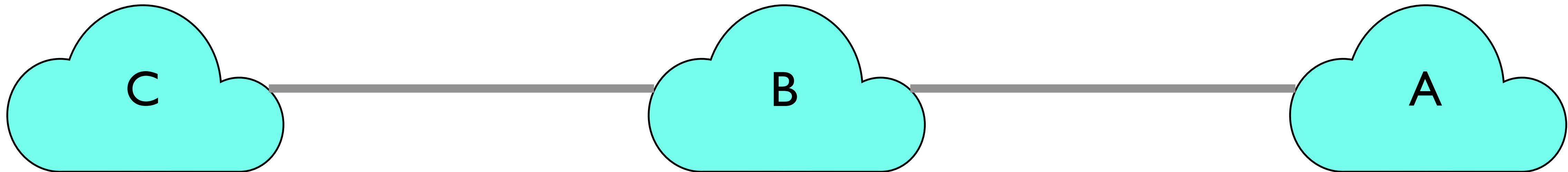
(3) selective route advertisement

- For policy reasons, an AS may choose not to advertise a route to a destination
- Hence reachability is not guaranteed even if the graph is connected

Differences between BGP and DV

(3) selective route advertisement

- For policy reasons, an AS may choose not to advertise a route to a destination
- Hence reachability is not guaranteed even if the graph is connected



Example: AS B does not want to carry traffic between AS A and AS C

Differences between BGP and DV

(4) BGP may *aggregate routes*

Differences between BGP and DV

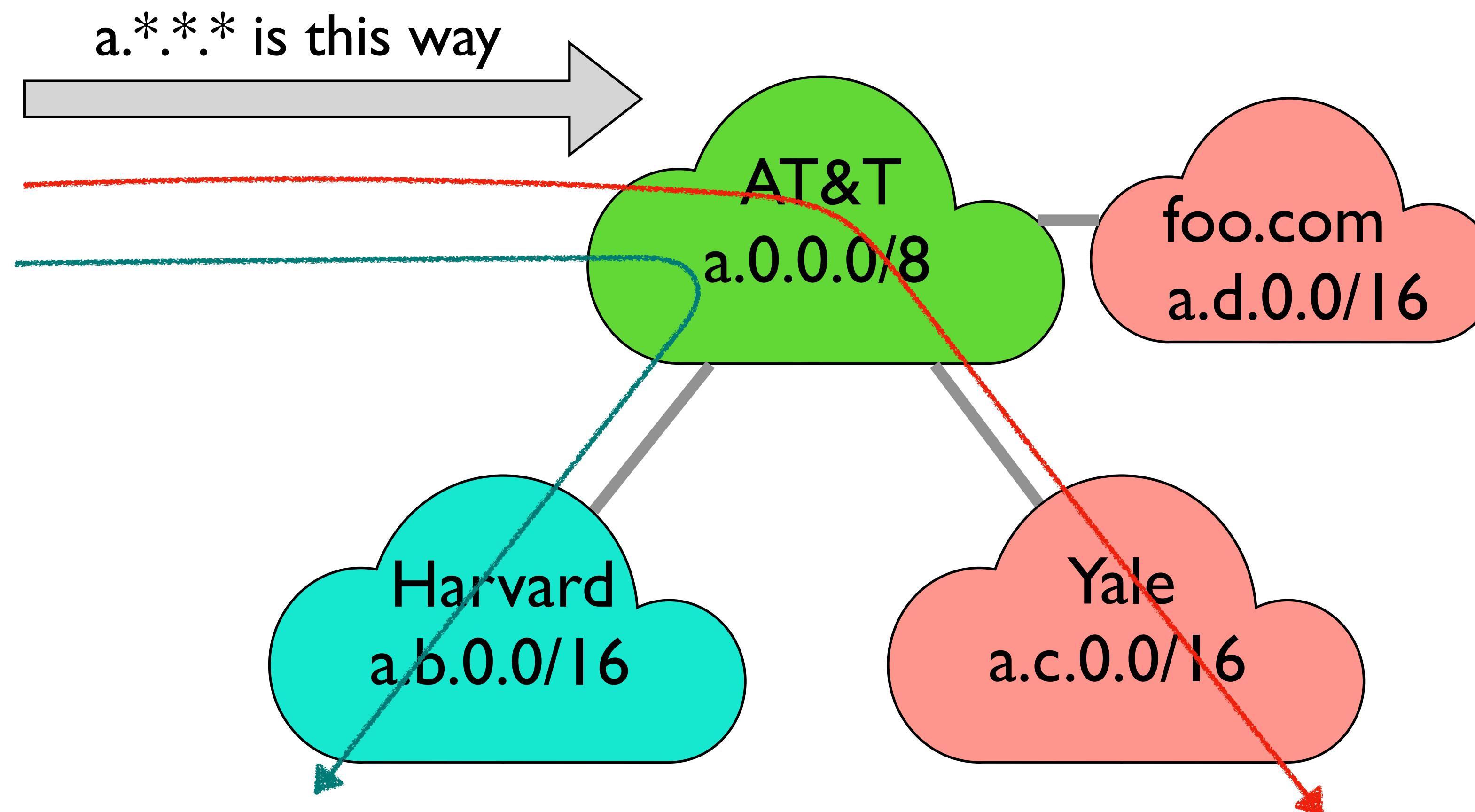
(4) BGP may *aggregate* routes

- For scalability, BGP may aggregate routes for different prefixes

Differences between BGP and DV

(4) BGP may *aggregate routes*

- For scalability, BGP may aggregate routes for different prefixes



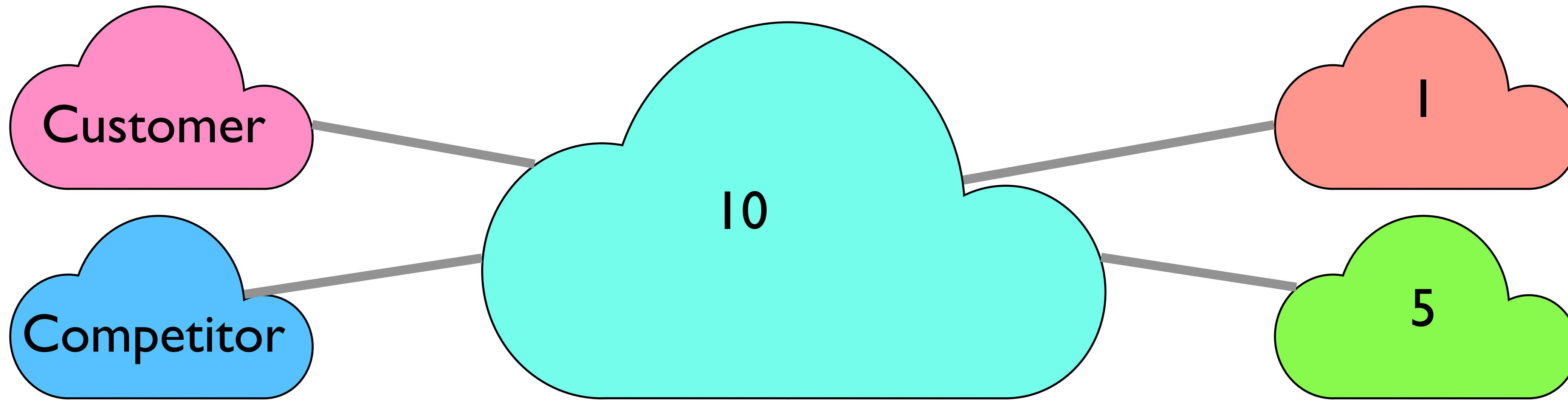
Questions?

BGP (Today)

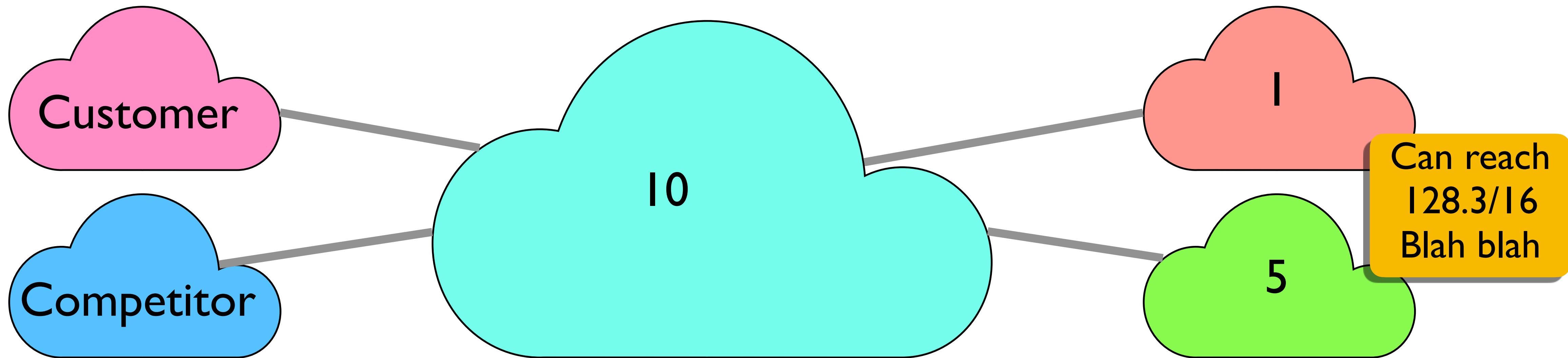
- **The role of policy**
 - What we mean by it
 - Why we need it
- **Overall Approach**
 - Four non-trivial changes to DV
 - How policy is implemented (detail-free version)

Policy imposed in how routes are selected and exported

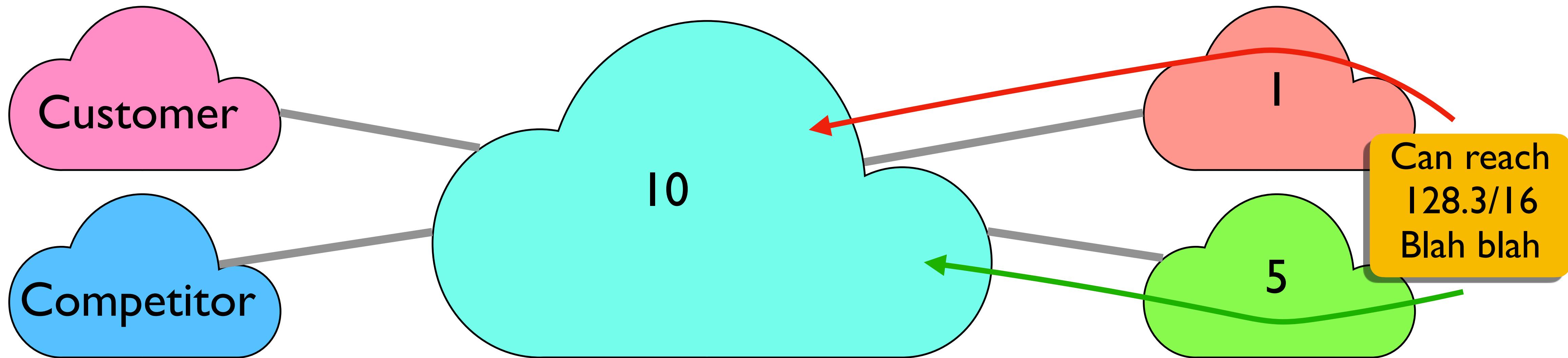
Policy imposed in how routes are selected and exported



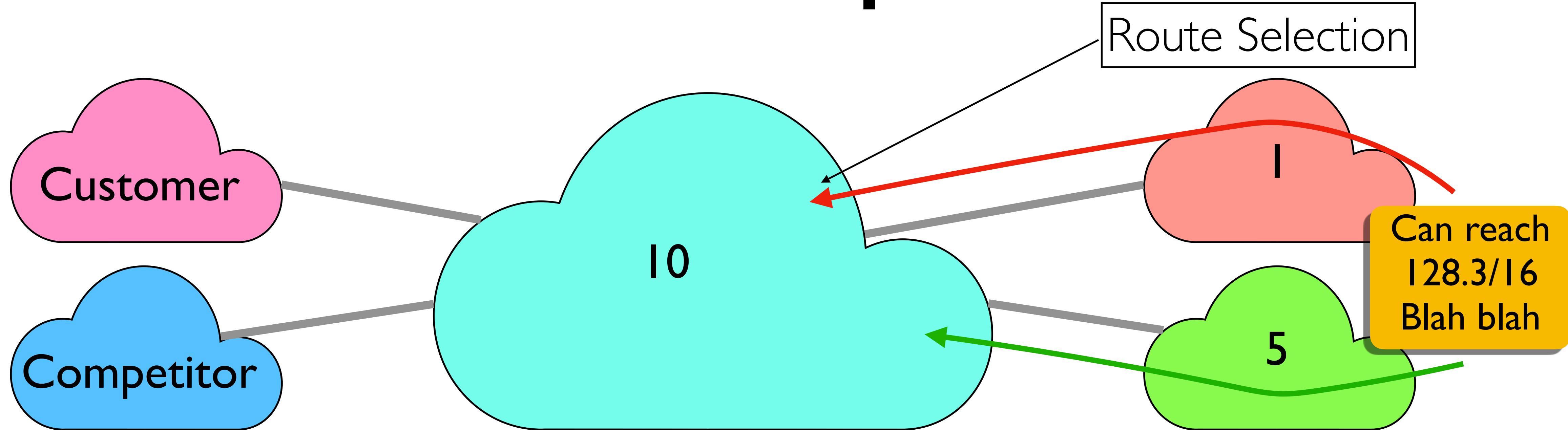
Policy imposed in how routes are selected and exported



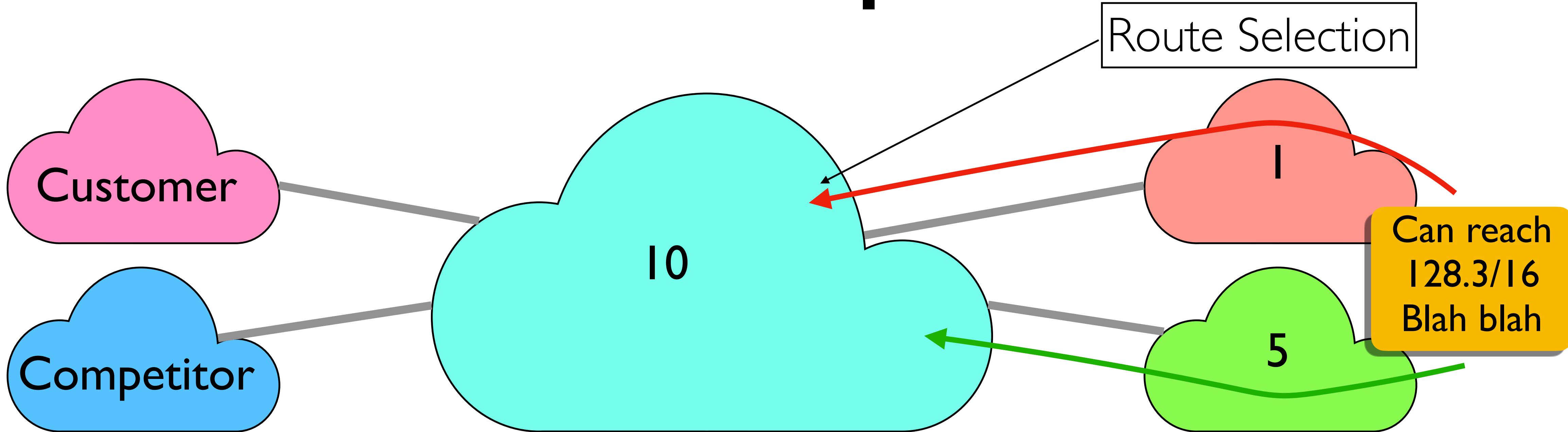
Policy imposed in how routes are selected and exported



Policy imposed in how routes are selected and exported

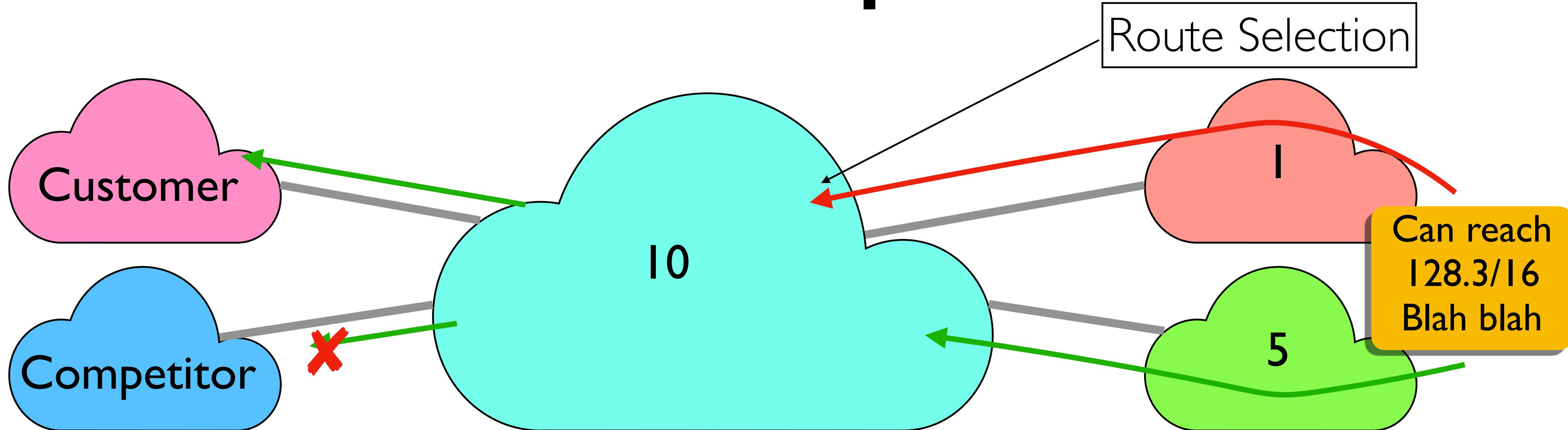


Policy imposed in how routes are selected and exported



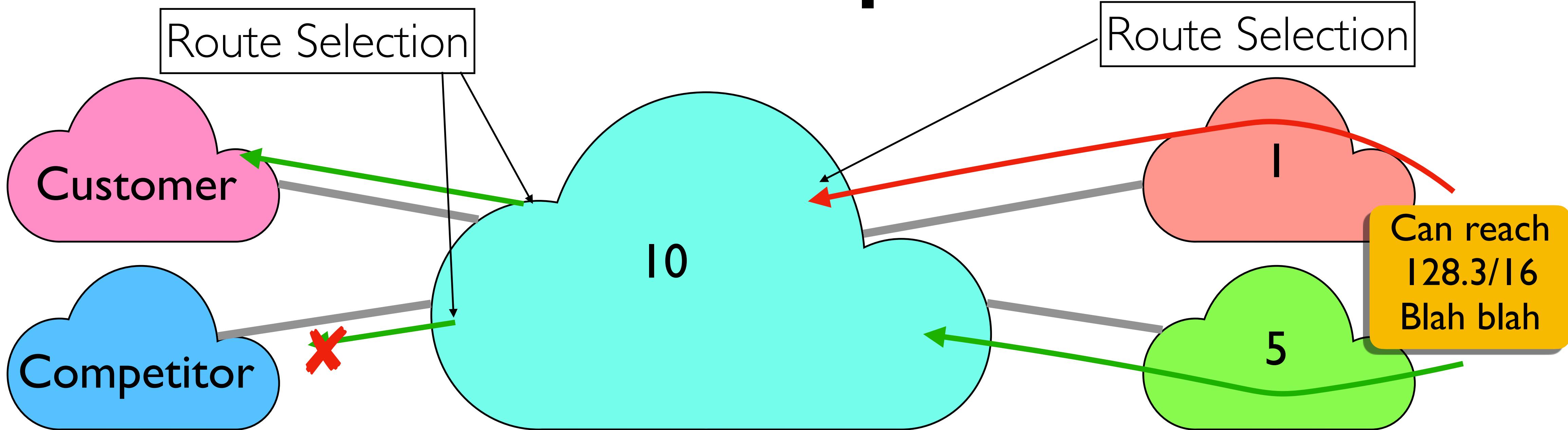
- **Selection:** Which path to use?
 - Controls whether/how traffic **leaves** the network

Policy imposed in how routes are selected and exported



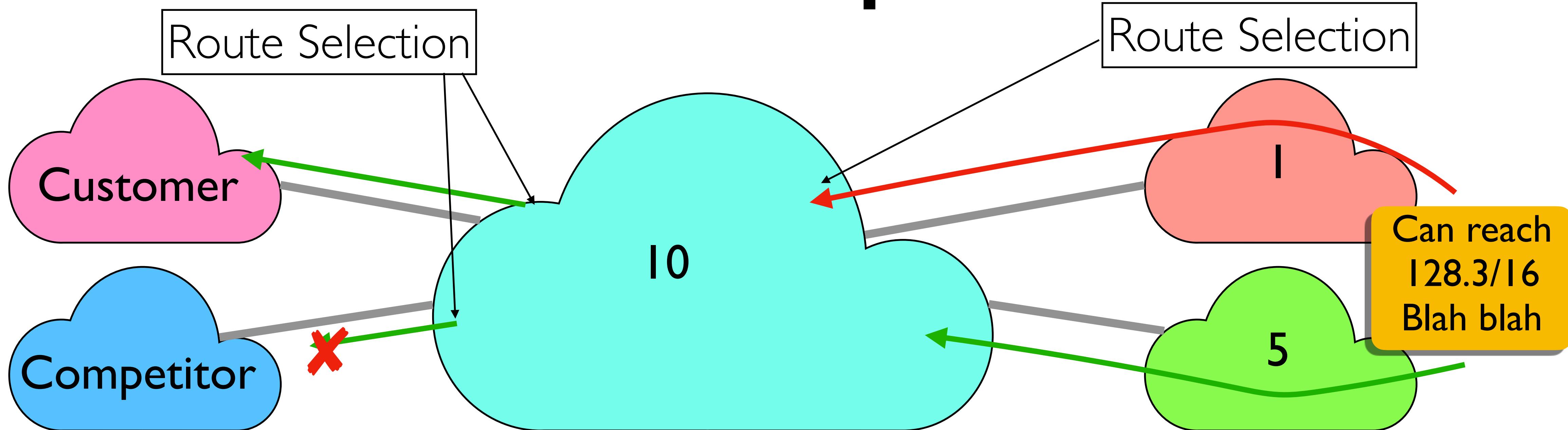
- **Selection:** Which path to use?
 - Controls whether/how traffic **leaves** the network

Policy imposed in how routes are selected and exported



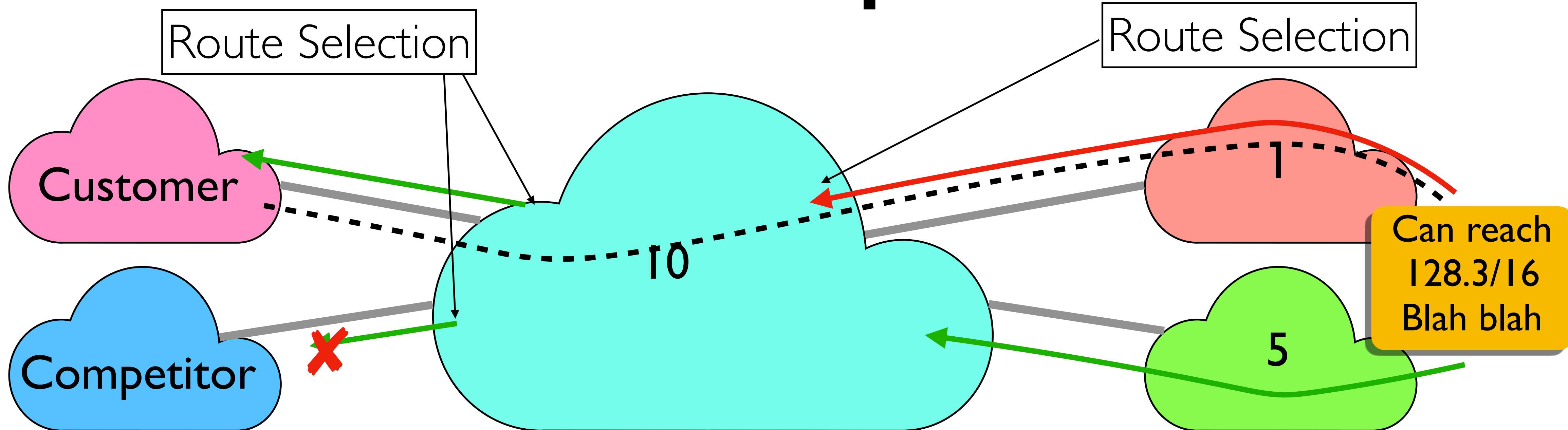
- **Selection:** Which path to use?
 - Controls whether/how traffic **leaves** the network

Policy imposed in how routes are selected and exported



- **Selection:** Which path to use?
 - Controls whether/how traffic **leaves** the network
- **Export:** Which paths to advertise?
 - Controls whether/how traffic **enters** the network

Policy imposed in how routes are selected and exported



- **Selection:** Which path to use?
 - Controls whether/how traffic **leaves** the network
- **Export:** Which paths to advertise?
 - Controls whether/how traffic **enters** the network

Typical Selection Policy

Typical Selection Policy

- In decreasing order of priority
 - Make/save money (send to customer > peer > provider)
 - Maximize performance (small AS path length)
 - Minimize use of my network bandwidth (“hot potato”)
 - ...

Typical Selection Policy

- In decreasing order of priority
 - Make/save money (send to customer > peer > provider)
 - Maximize performance (small AS path length)
 - Minimize use of my network bandwidth (“hot potato”)
 - ...
- BGP uses something called route “attributes” to implement the above (next lecture)

Typical Export: Peer-Peer Case

- Peer exchange traffic between their customers



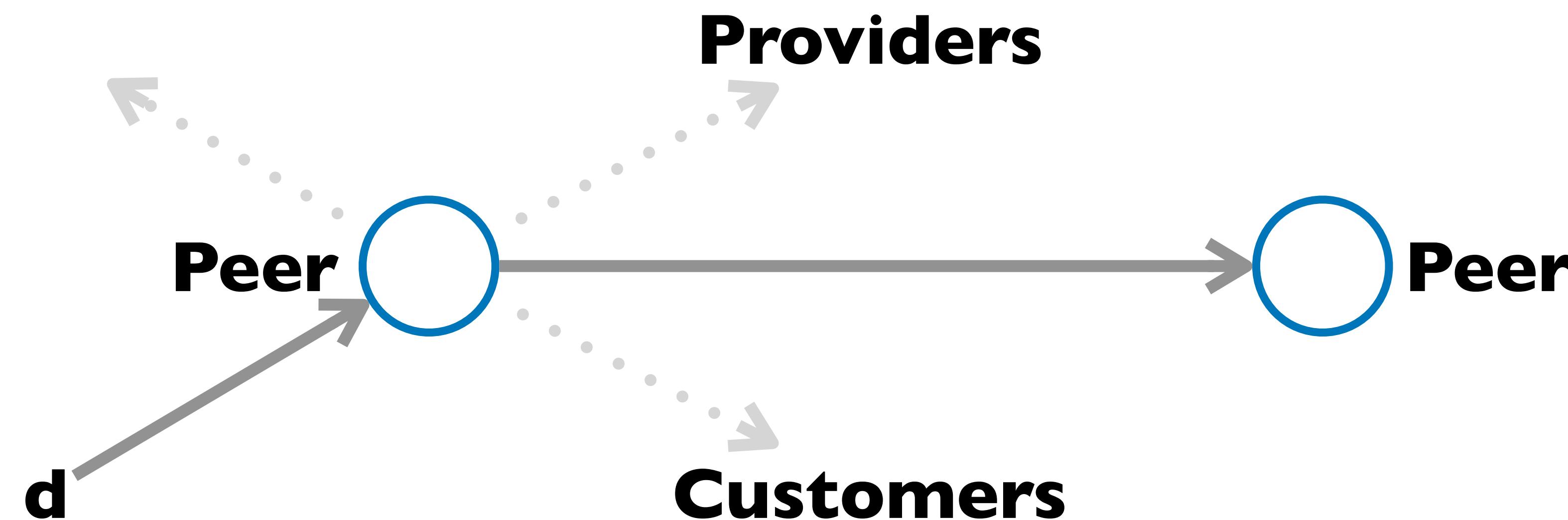
Typical Export: Peer-Peer Case

- Peer exchange traffic between their customers
 - AS exports **only customer routes to a peer**



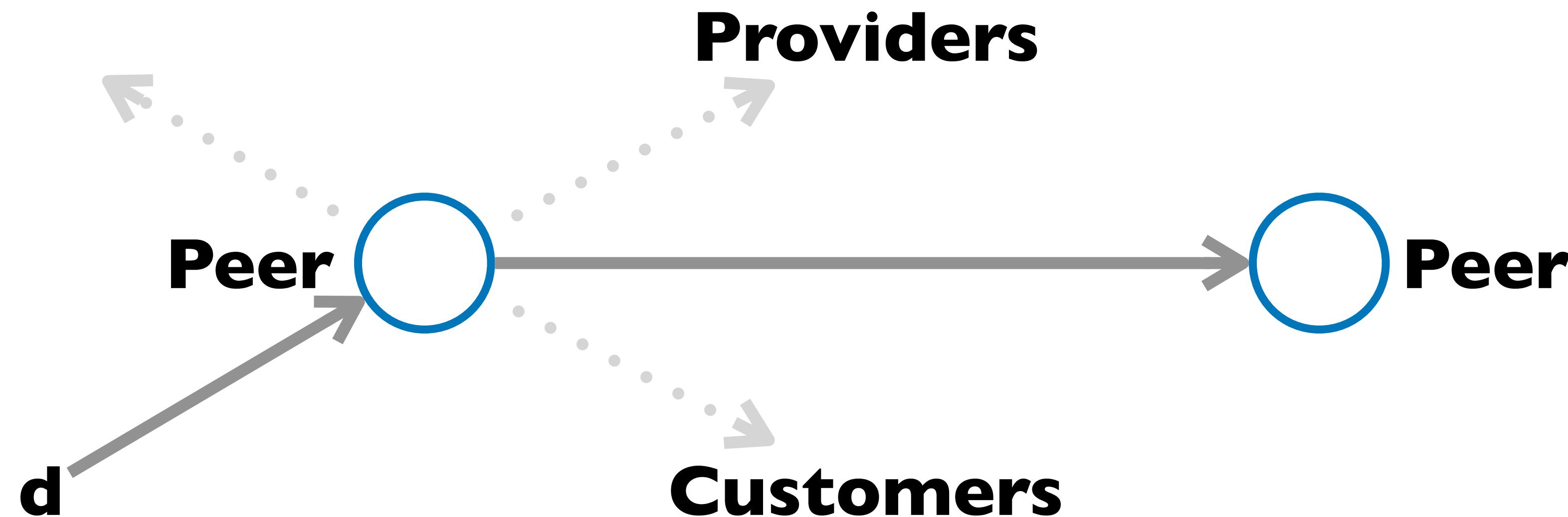
Typical Export: Peer-Peer Case

- Peer exchange traffic between their customers
 - AS exports **only customer routes to a peer**



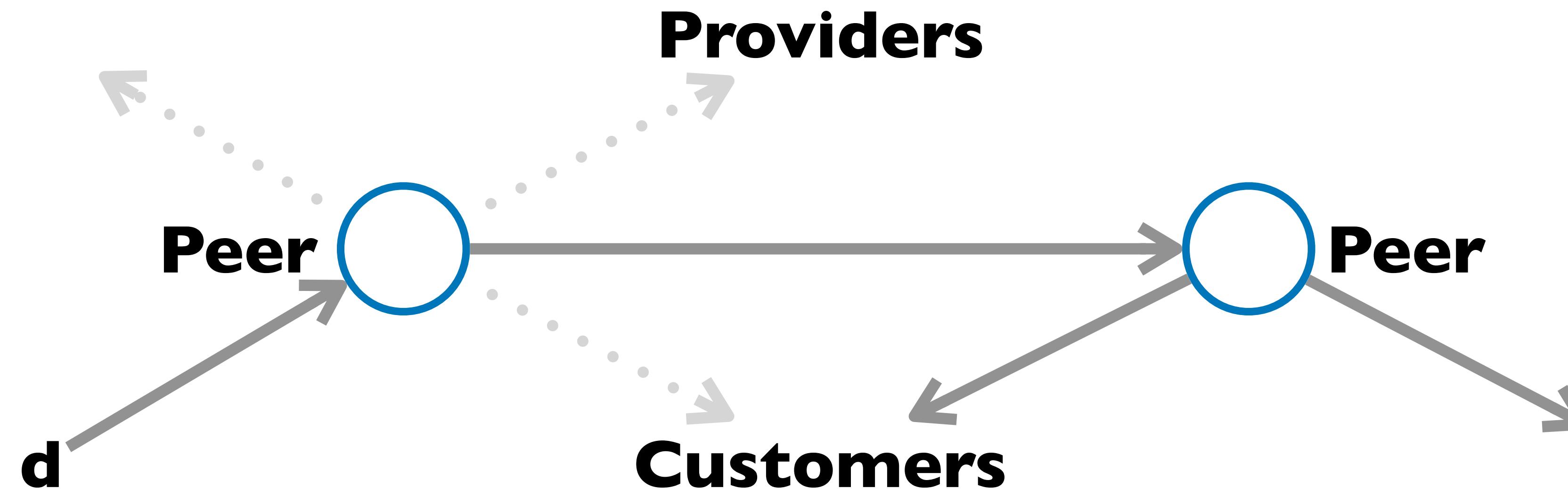
Typical Export: Peer-Peer Case

- Peer exchange traffic between their customers
 - AS exports **only customer routes to a peer**
 - AS exports **a peer's routes only to its customers**



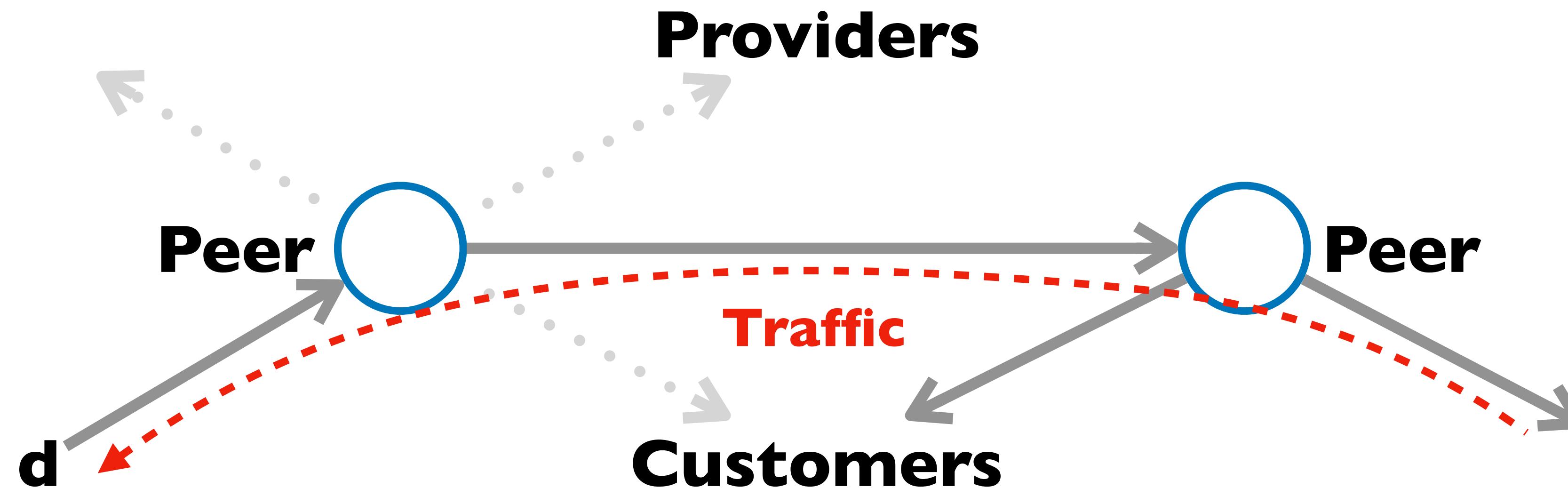
Typical Export: Peer-Peer Case

- Peer exchange traffic between their customers
 - AS exports **only customer routes to a peer**
 - AS exports **a peer's routes only to its customers**



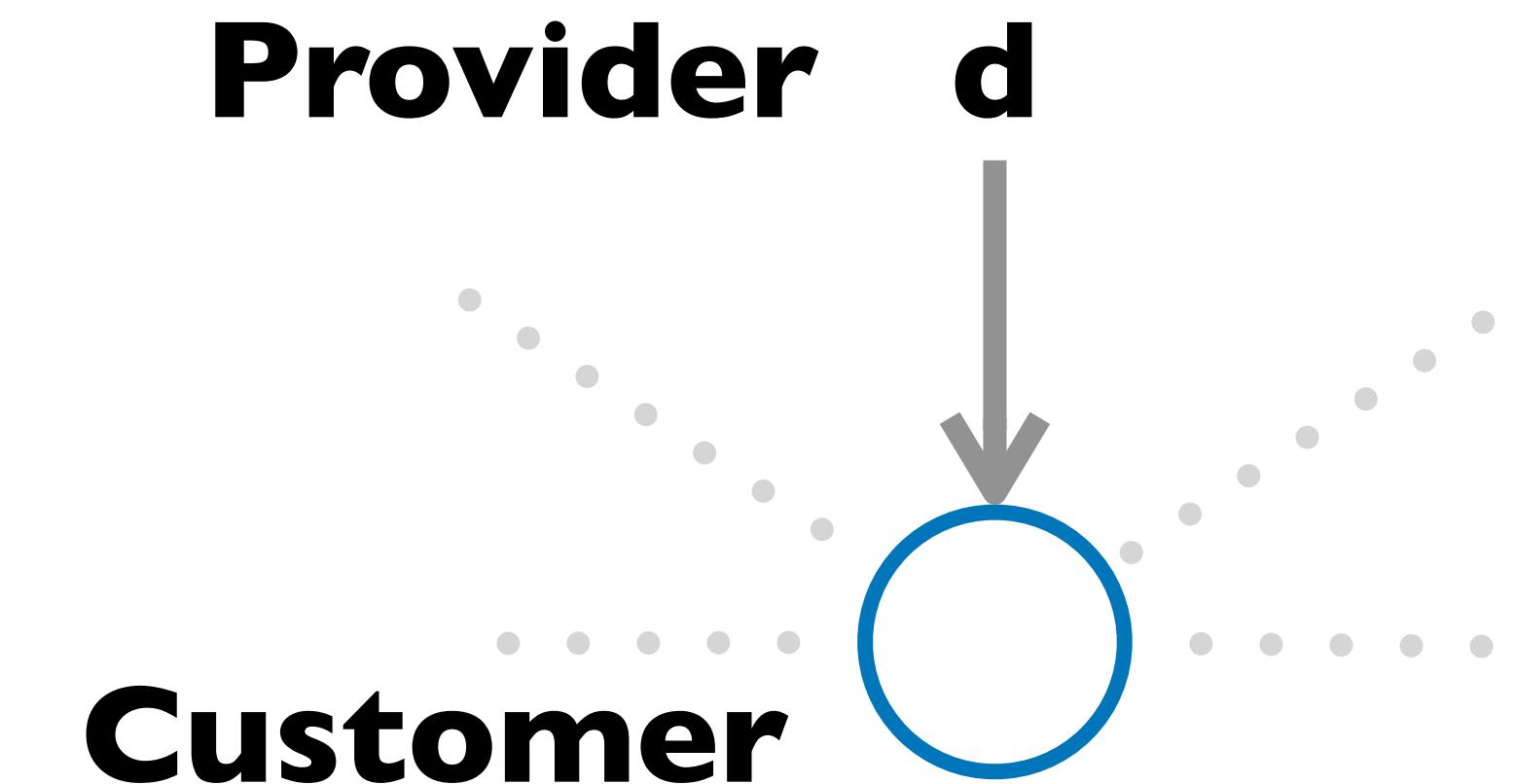
Typical Export: Peer-Peer Case

- Peer exchange traffic between their customers
 - AS exports **only customer routes to a peer**
 - AS exports **a peer's routes only to its customers**



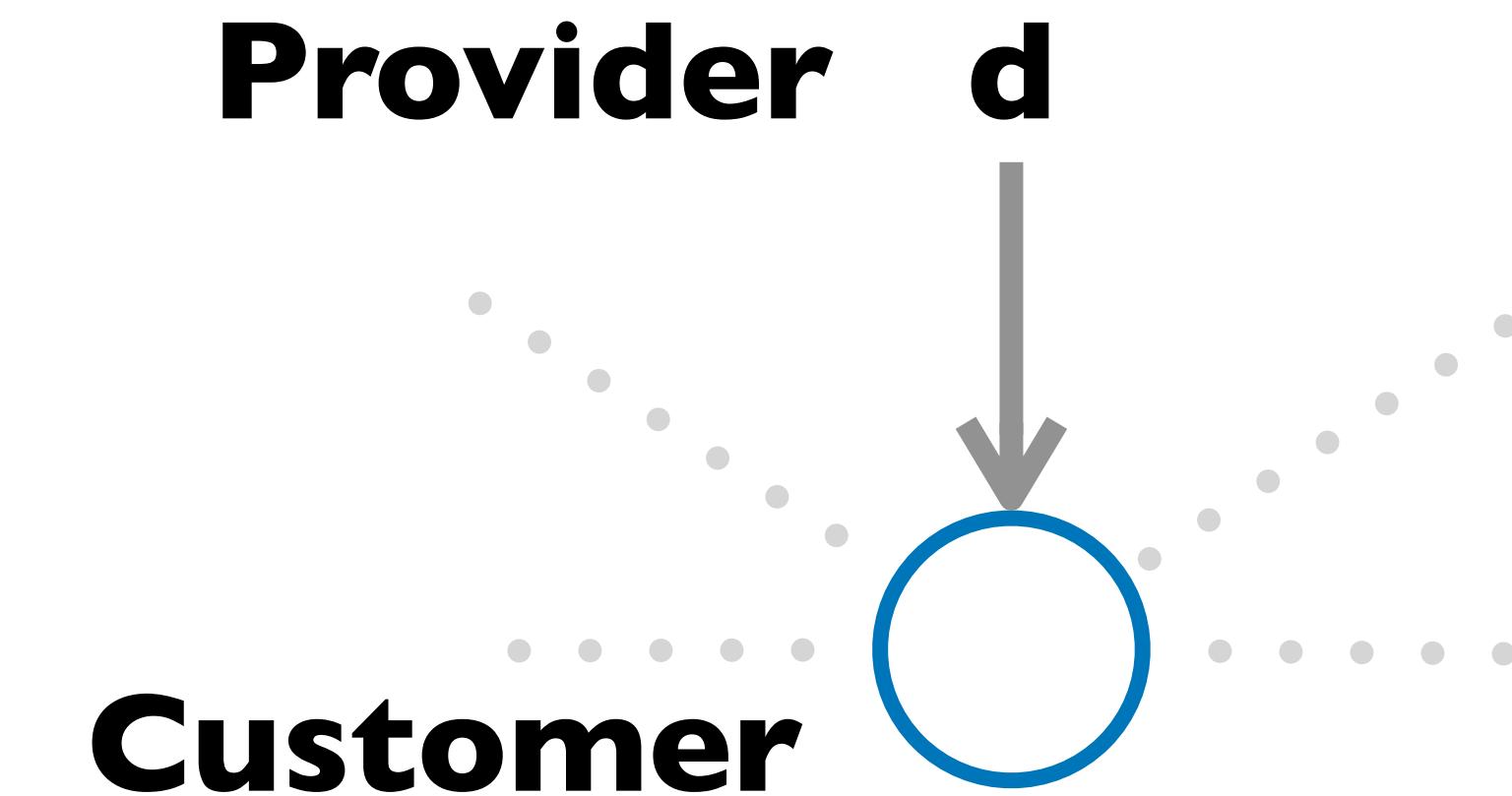
Typical Export: Customer-Provider Case

- Customer pays provider for access to Internet



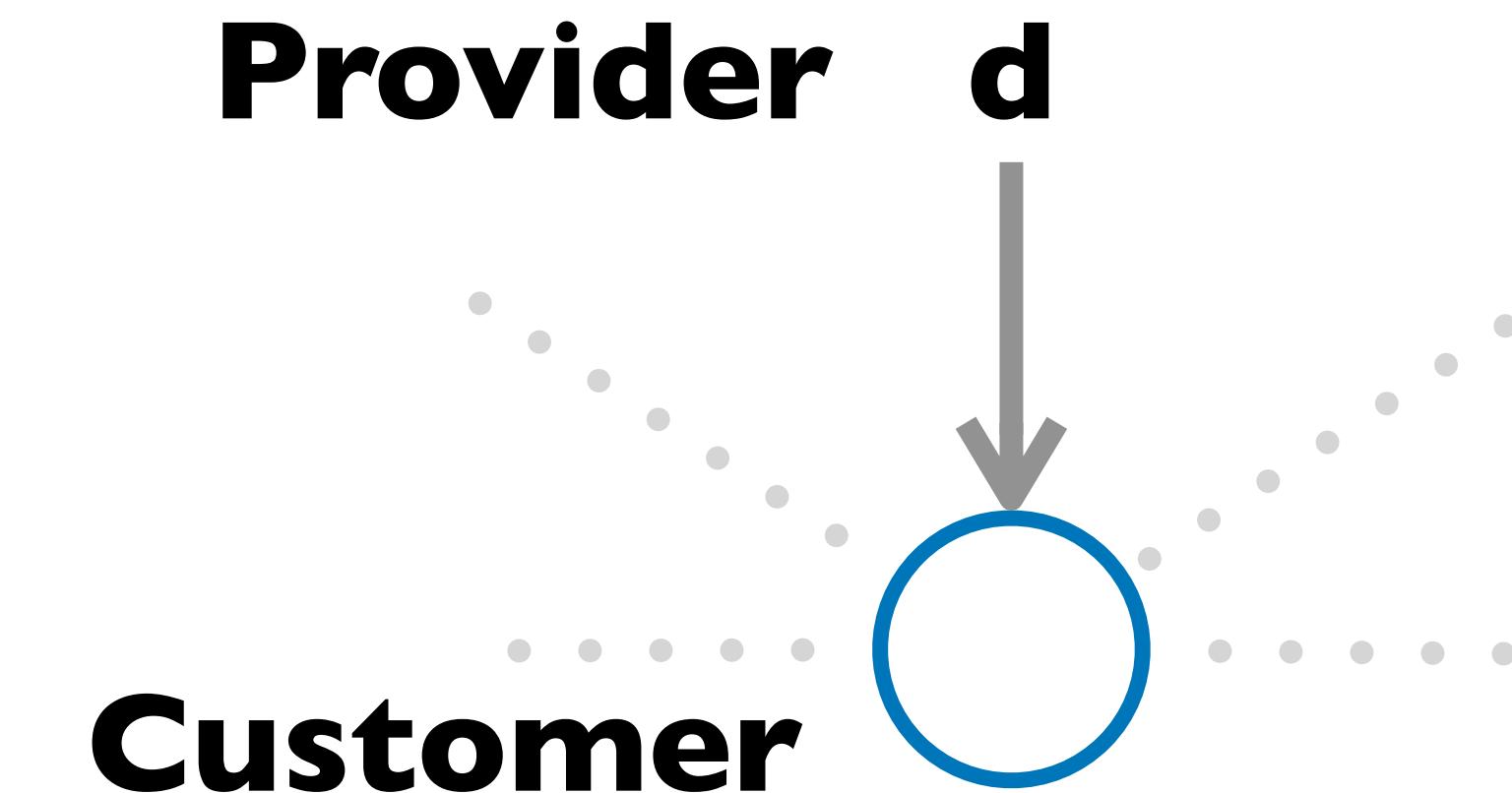
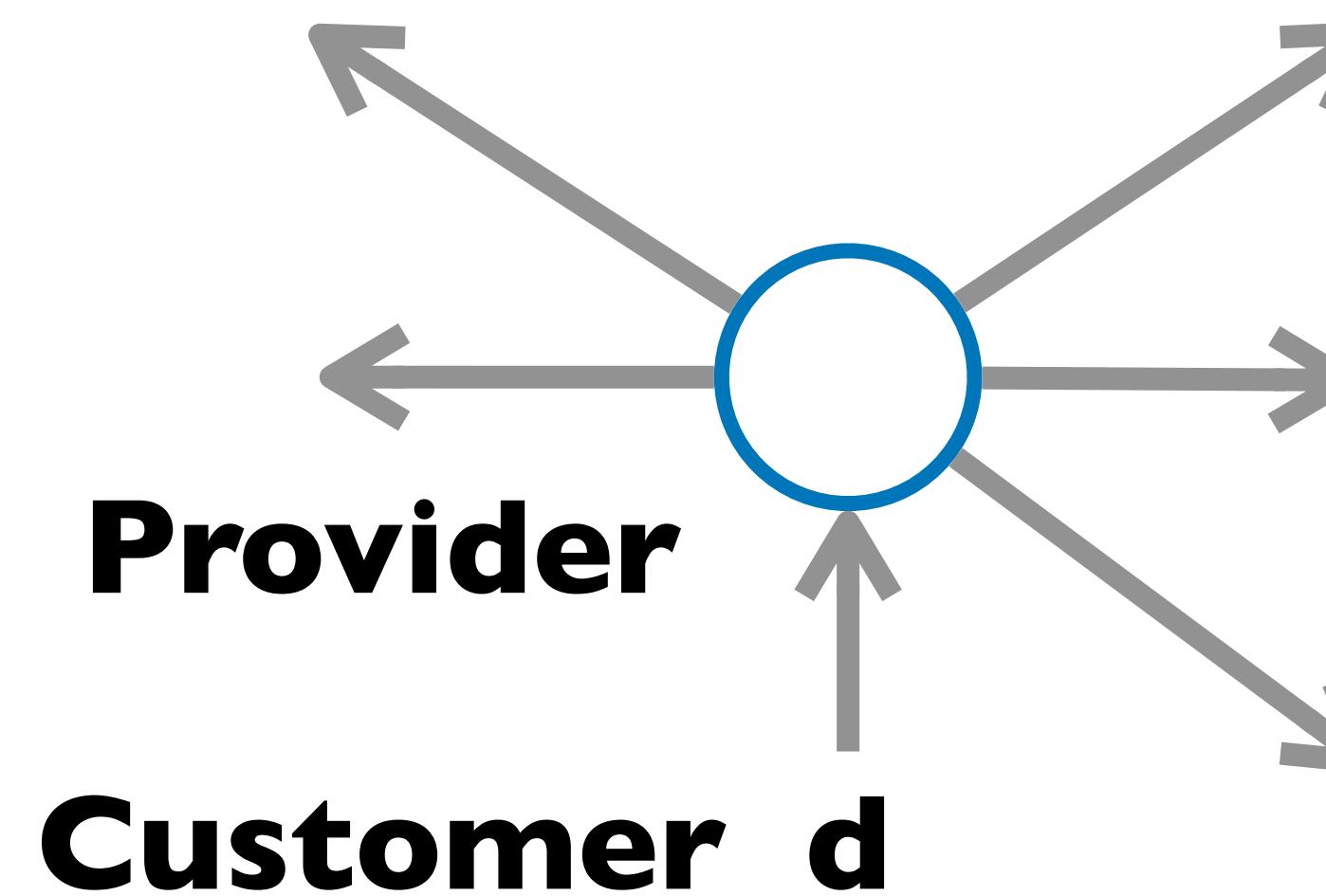
Typical Export: Customer-Provider Case

- Customer pays provider for access to Internet
 - Provider exports **its customer routes to everybody**



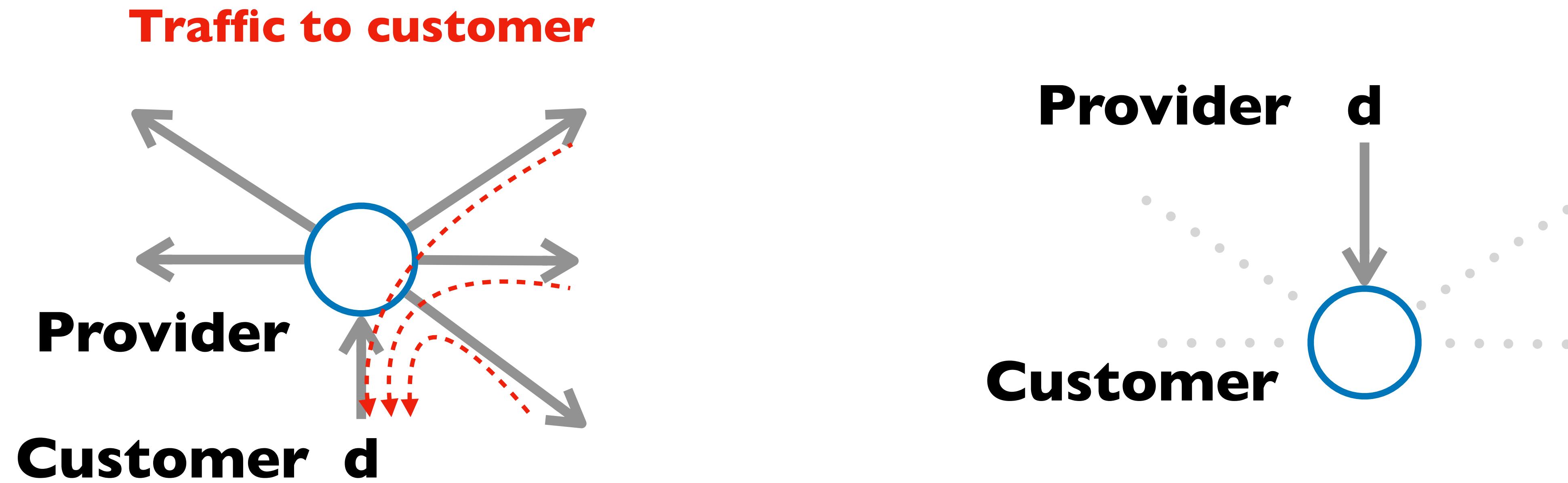
Typical Export: Customer-Provider Case

- Customer pays provider for access to Internet
 - Provider exports **its customer routes to everybody**



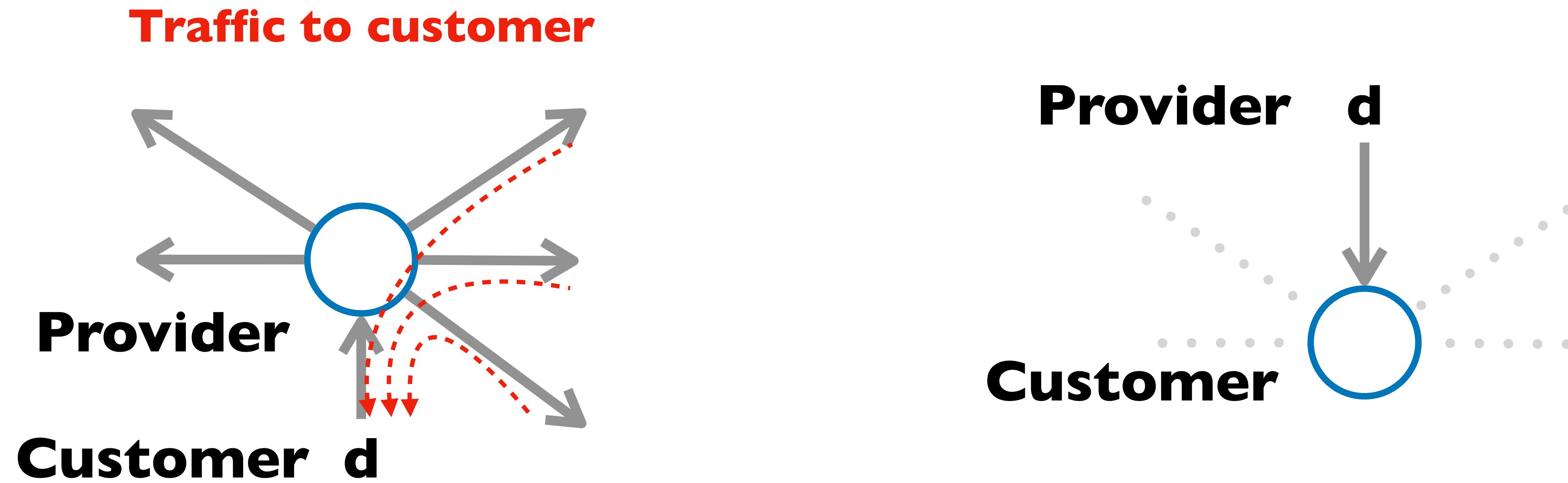
Typical Export: Customer-Provider Case

- Customer pays provider for access to Internet
 - Provider exports **its customer routes to everybody**



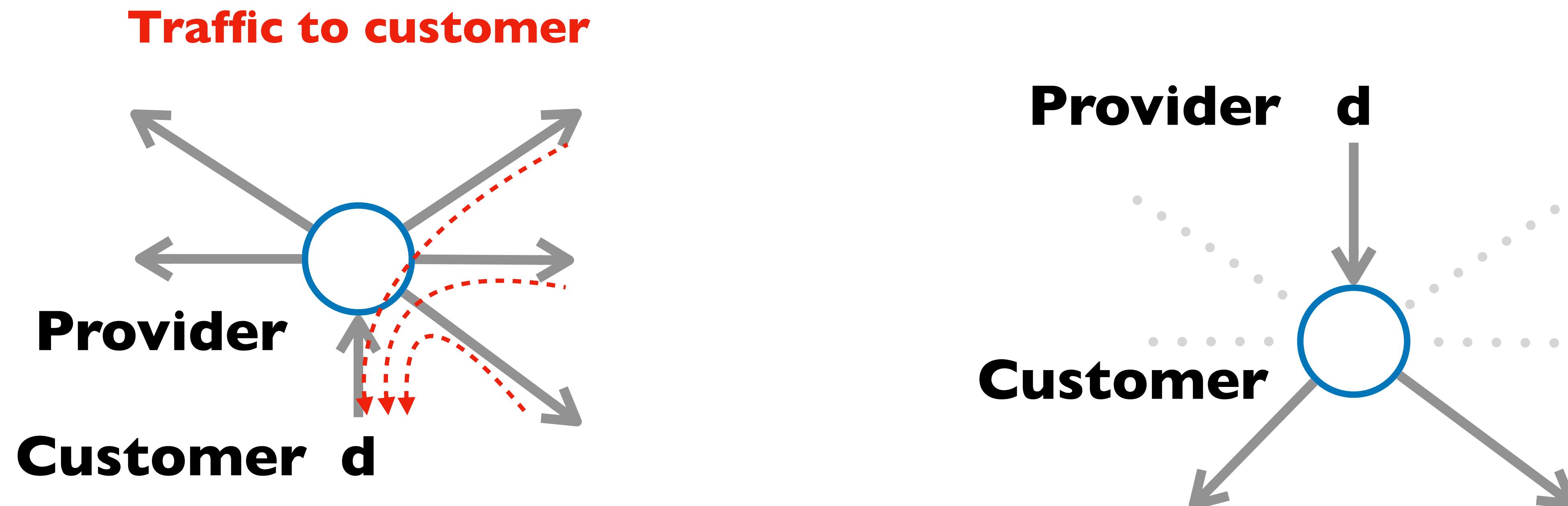
Typical Export: Customer-Provider Case

- Customer pays provider for access to Internet
 - Provider exports **its customer routes to everybody**
 - Customer exports **a provider routes only to its customers**



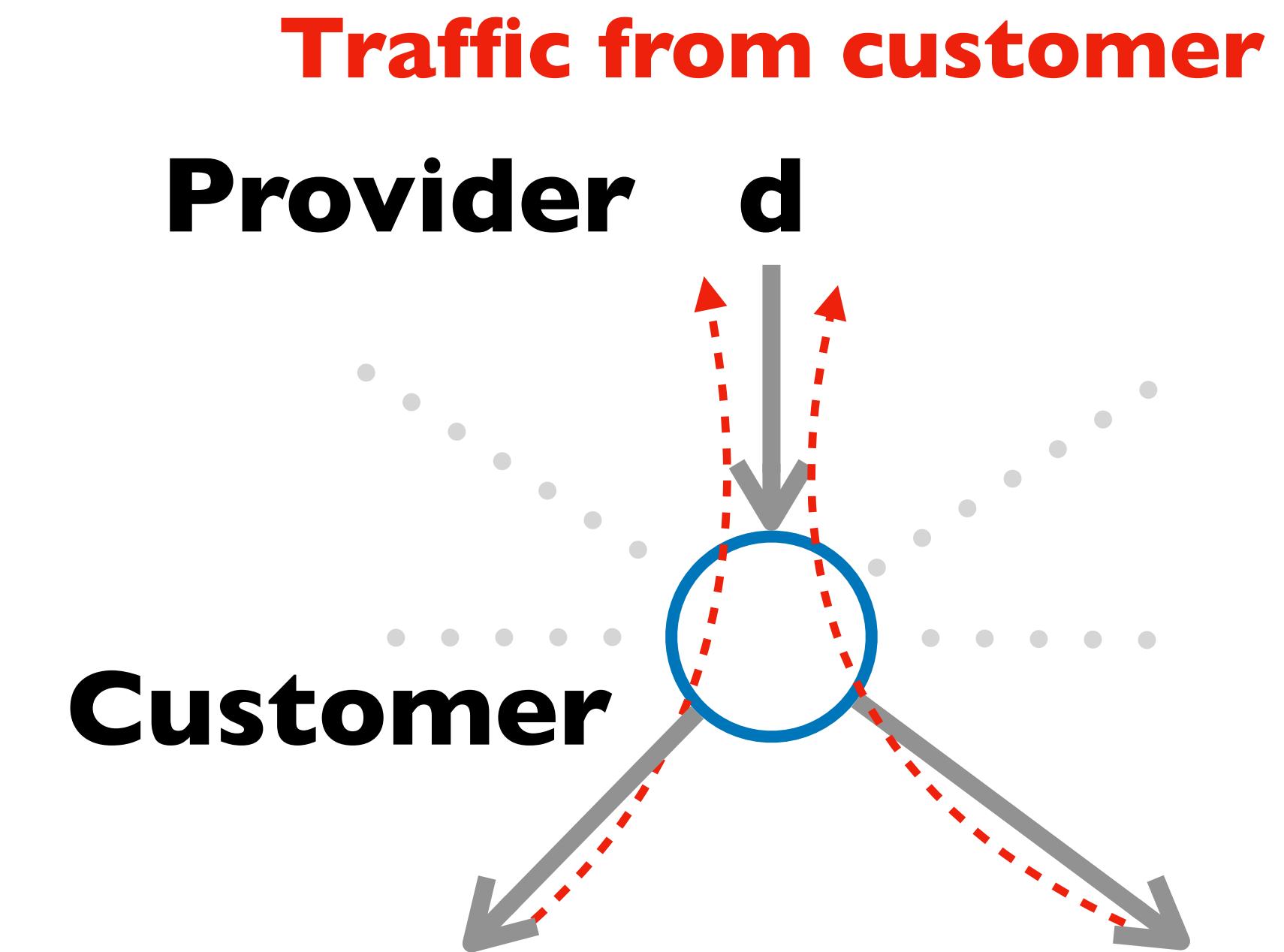
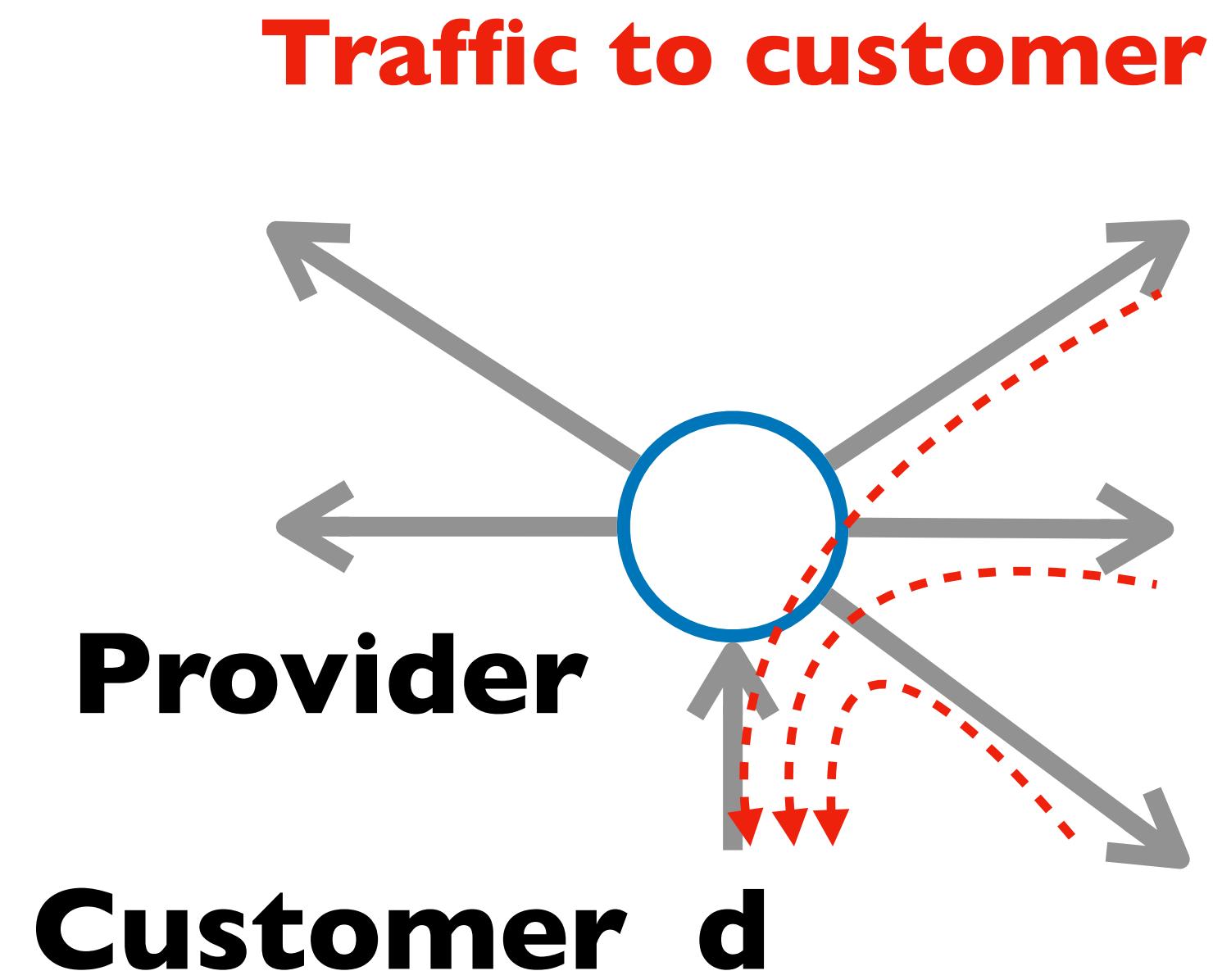
Typical Export: Customer-Provider Case

- Customer pays provider for access to Internet
 - Provider exports **its customer routes to everybody**
 - Customer exports **a provider routes only to its customers**



Typical Export: Customer-Provider Case

- Customer pays provider for access to Internet
 - Provider exports **its customer routes to everybody**
 - Customer exports **a provider routes only to its customers**



Next Time

- **Wrap up BGP**

- Protocol details
- Pitfalls

Questions?