15-441/641: Computer Networks

Intradomain Routing, cont'd

15-441 Spring 2019

Profs Peter Steenkiste & **Justine Sherry**

Carnegie Mellon University

billions of packets
@justinesherry

"Packets have a very short lifetime. It's really kind of a miserable life."

#RealTalkWithProfSteenkiste

3:10 PM · Aug 29, 2019 · Buffer

Jeff Mogul
@JeffreyMogul

Replying to @justinesherry

It’s worse than that for IP packets; they know their remaining time to live, so there’s no room for hope.

3:13 PM · Aug 29, 2019 · Twitter for iPhone

1 Retweet **19** Likes

Jitendra Padhye
@jitu_pd

Replying to @justinesherry

Unlike people, packets know their destination and proceed to it single mindedly. Their life may be short, but it is full of purpose.

4:09 PM · Aug 29, 2019 · Twitter for iPhone

3 Retweets **19** Likes

Jeff Dean
@JeffDean

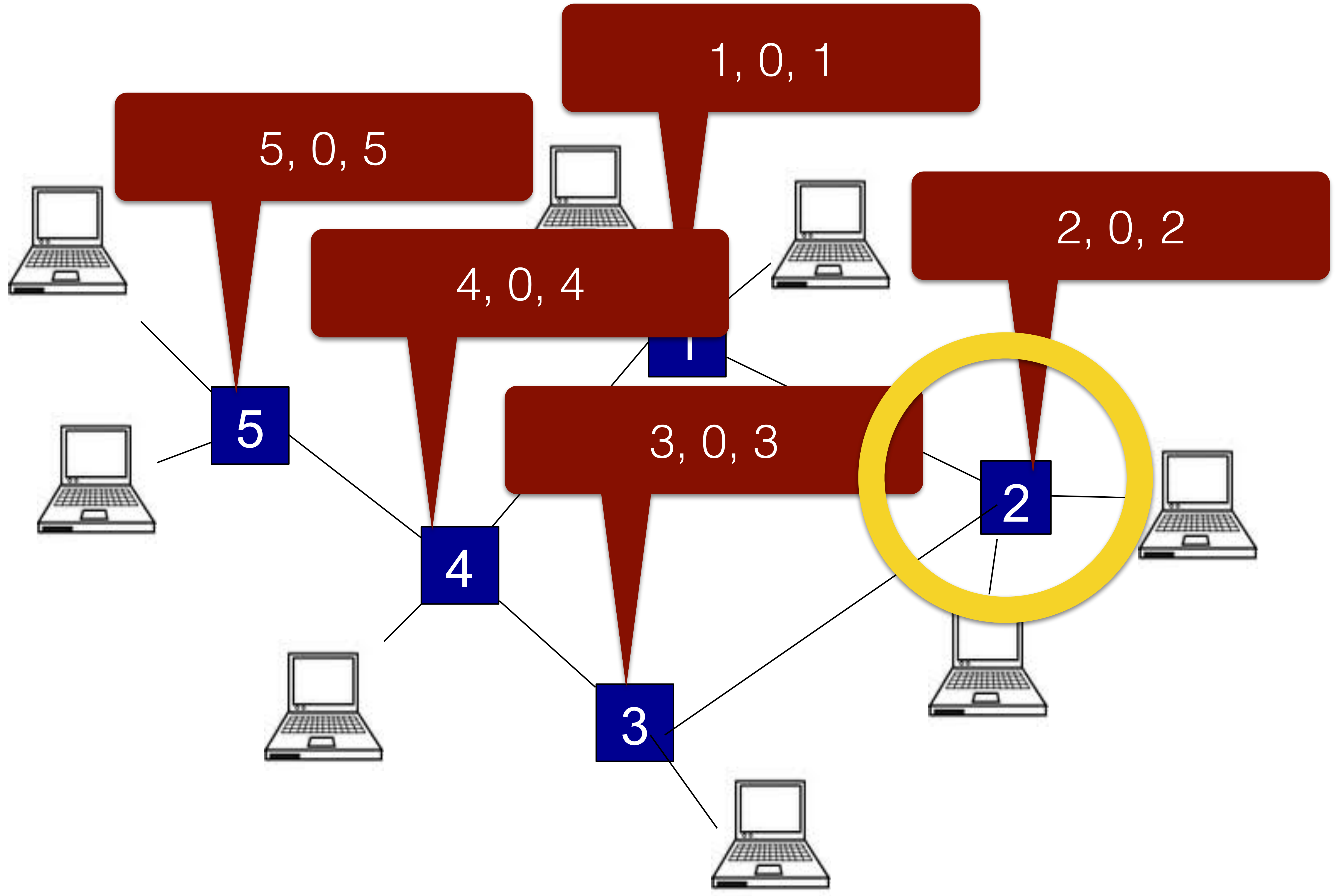
Replying to @justinesherry

I feel like that's true now for our terrestrial packets, but eventually we'll want packets that live longer than disks. Otherwise, how will we communicate to everyone?

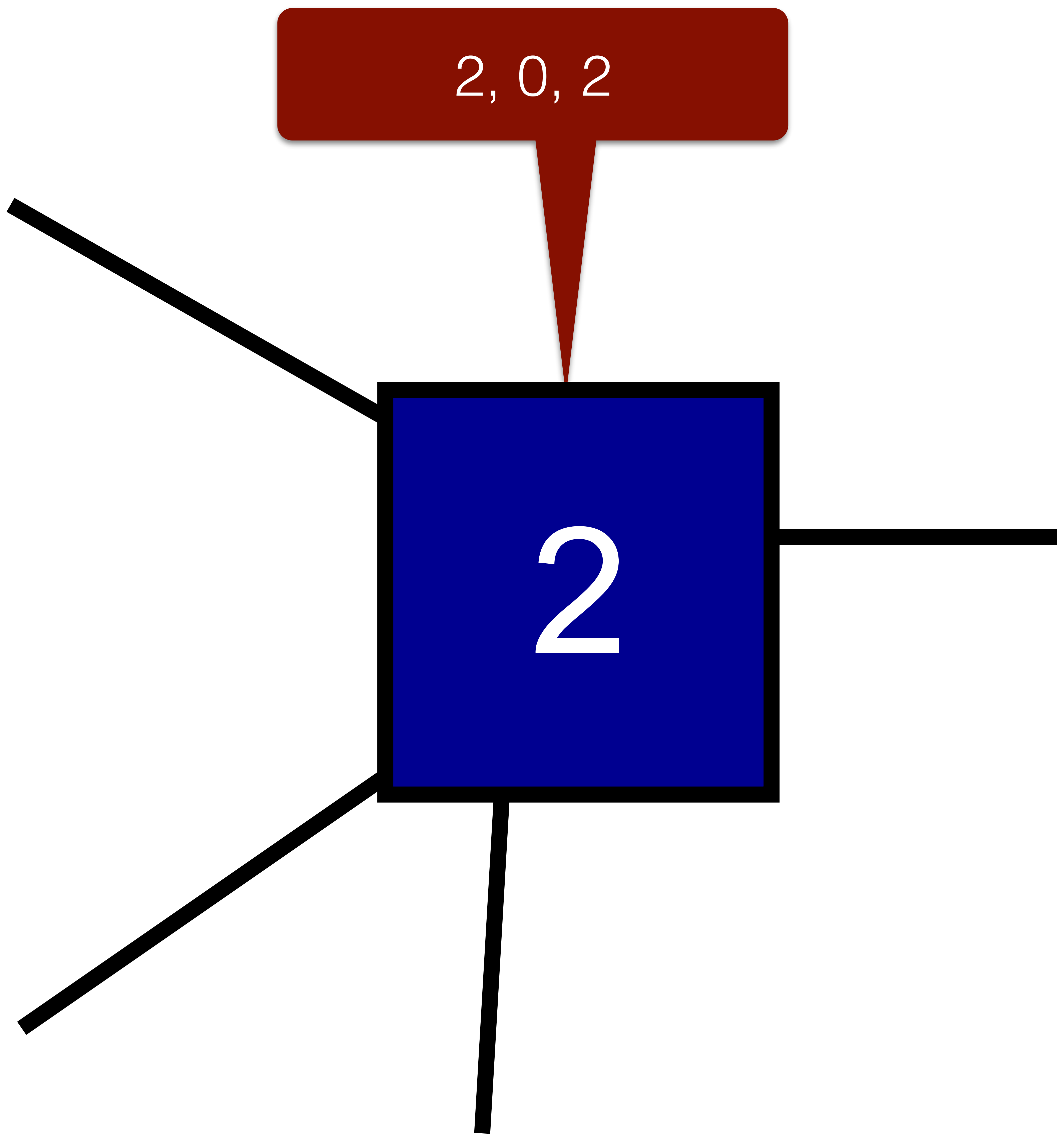
10:52 PM · Aug 29, 2019 · Twitter for Android

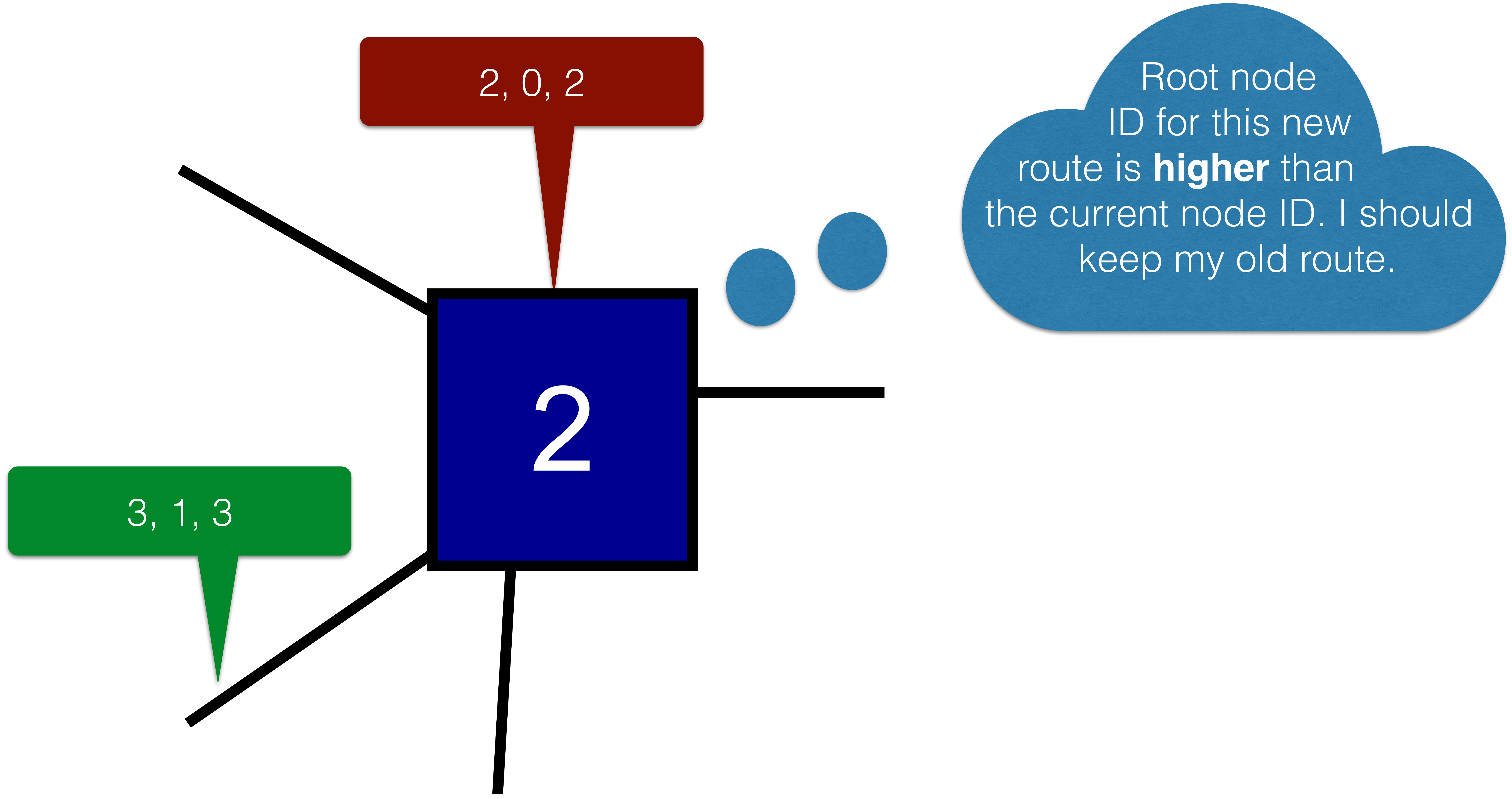
3 Retweets **24** Likes

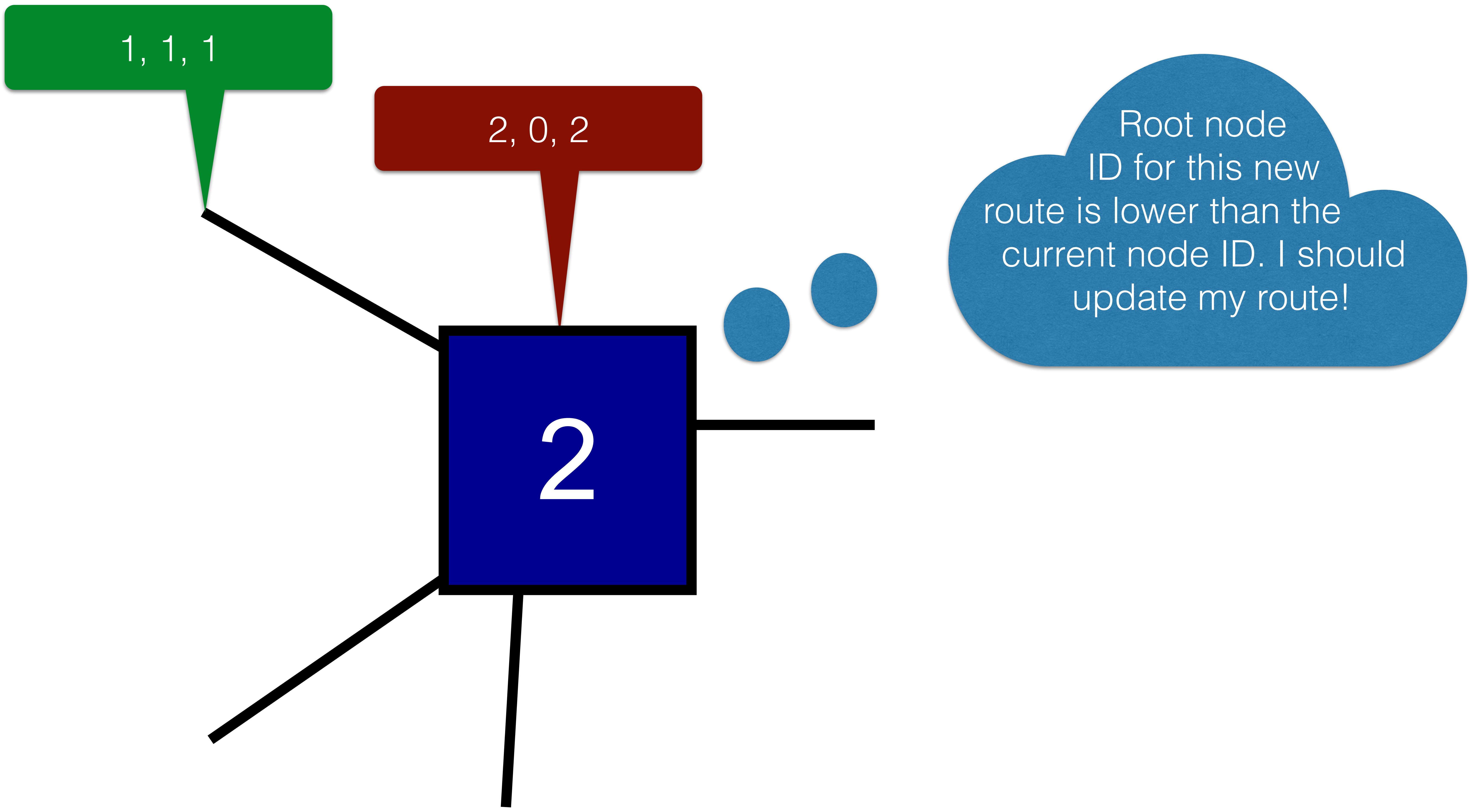
Refresher…

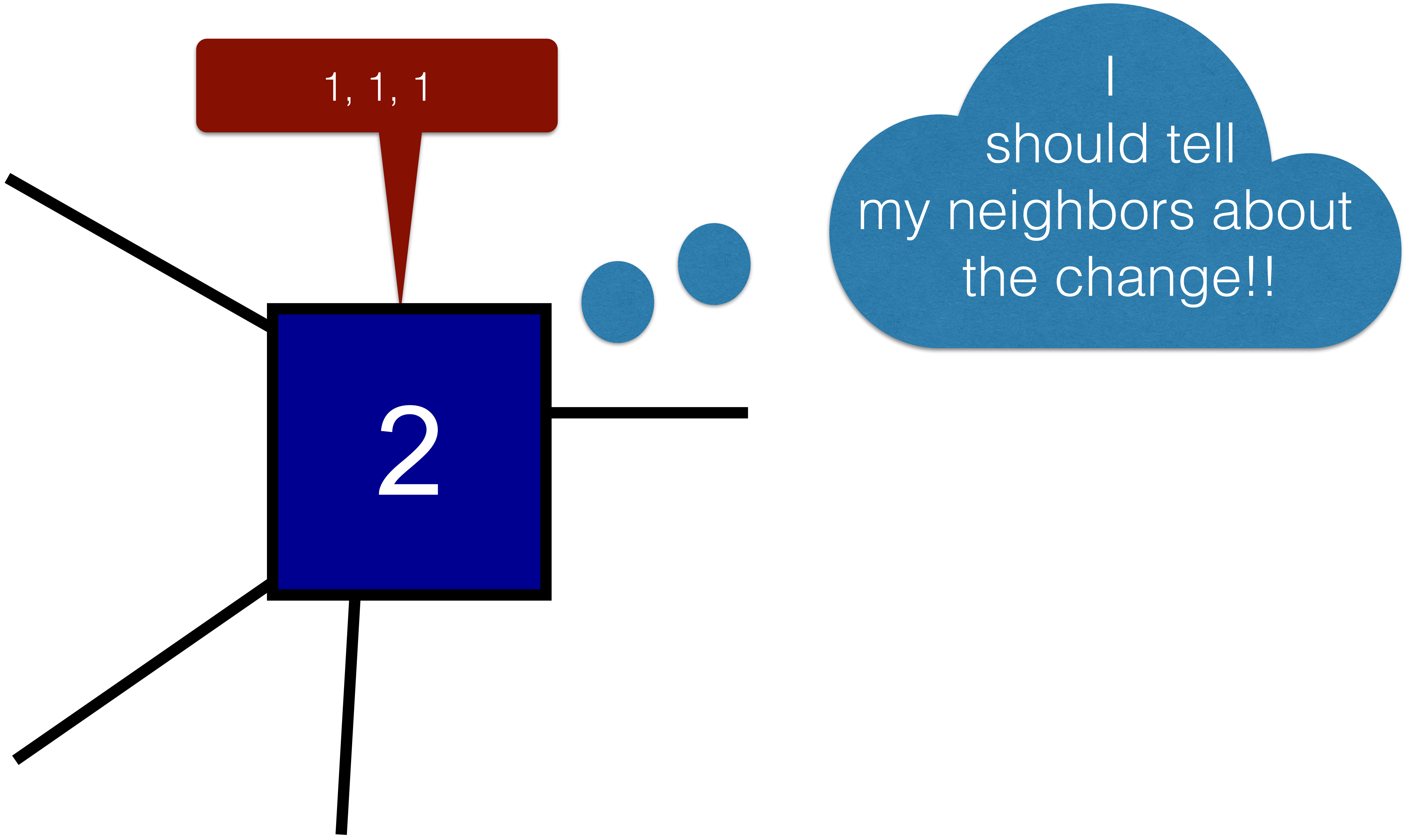


(root, path lenth, next hop)

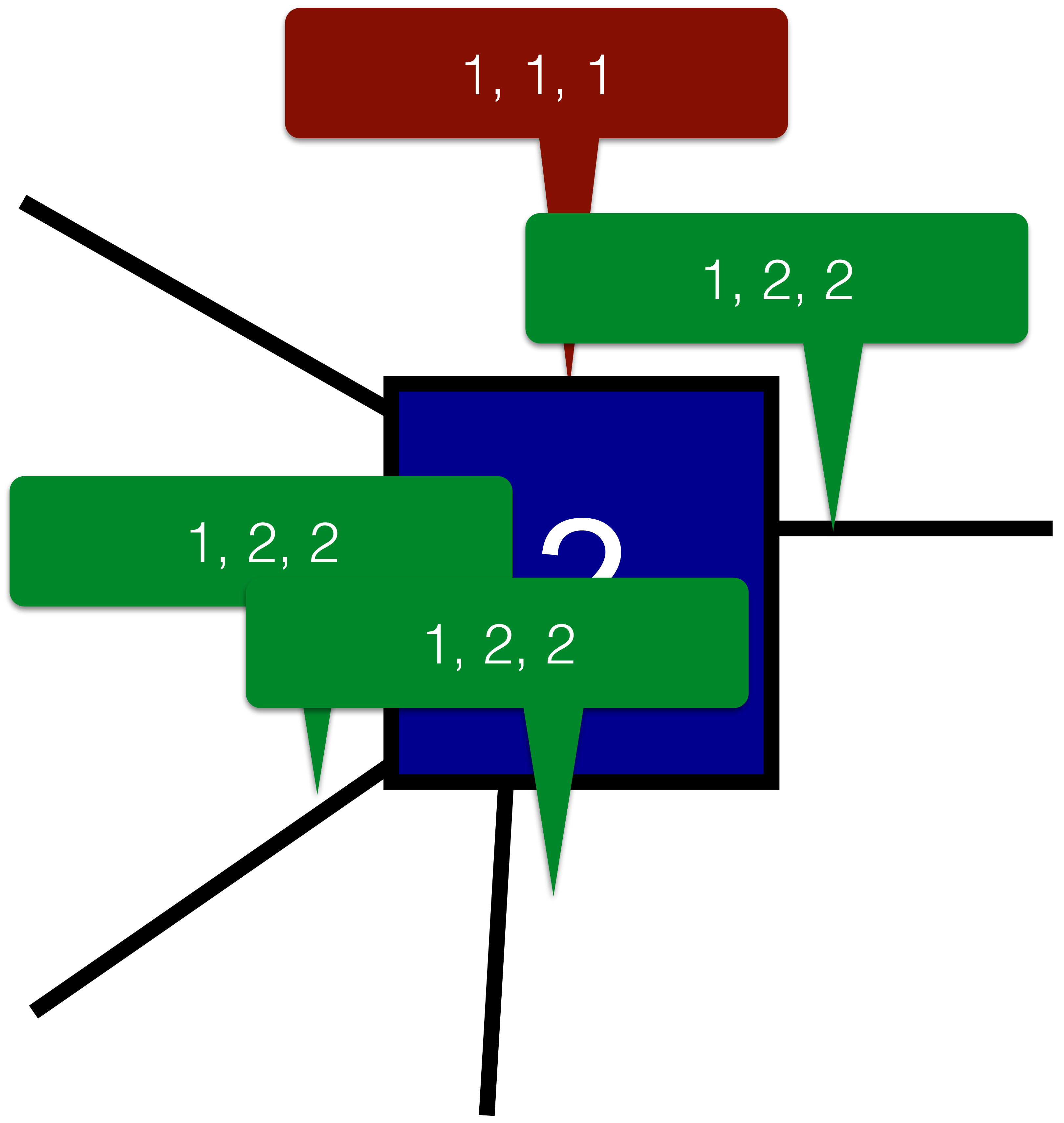

2, 0, 2
2


2, 0, 2
2
3, 1, 3
Root node ID for this new route is **higher** than the current node ID. I should keep my old route.


1, 1, 1
2, 0, 2
2
Root node
ID for this new
route is lower than the
current node ID. I should
update my route!



1, 1, 1
2
I
should tell
my neighbors about
the change!!


1, 1, 1
1, 2, 2
1, 2, 2
1, 2, 2

Refresh: Try it Out!

Trade-Offs

	Broadcast Network w/ Learning Switches	Broadcast Network w/ Learning Switches and Spanning Tree
Resilience	If there is a route, the packet will reach dest!	Need to recompute spanning tree if failure

Resilience: the ability to provide and maintain an acceptable level of service in the face of faults and challenges to normal operation

Trade-Offs

	Broadcast Network w/ Learning Switches	Broadcast Network w/ Learning Switches and Spanning Tree
Resilience	If there is a route, the packet will reach dest!	Need to recompute spanning tree if failure
Fully Distributed	Yes	Yes

Fully Distributed: does not assume the previous existence of a central coordinator.

Trade-Offs

	Broadcast Network w/ Learning Switches	Broadcast Network w/ Learning Switches and Spanning Tree
Resilience	If there is a route, the packet will reach dest!	Need to recompute spanning tree if failure
Fully Distributed	Yes	Yes
State per Node	Learning Switch: O(#nodes)	Learning Switch: O(#nodes) + Path to Root: O(constant)

State: The amount of memory each node uses

Trade-Offs

	Broadcast Network w/ Learning Switches	Broadcast Network w/ Learning Switches and Spanning Tree
Resilience	If there is a route, the packet will reach dest!	Need to recompute spanning tree if failure
Fully Distributed	Yes	Yes
State per Node	Learning Switch: O(#nodes)	Learning Switch: O(#nodes) + Path to Root: O(constant)
Convergence	No setup time at all!	Need to run spanning tree protocol before routing

Convergence: the process of routers/switches agreeing on optimal routes for forwarding packets and thereby completing the updating of their routing table

Trade-Offs

	Broadcast Network w/ Learning Switches	Broadcast Network w/ Learning Switches and Spanning Tree
Resilience	If there is a route, the packet will reach dest!	Need to recompute spanning tree if failure
Fully Distributed	Yes	Yes
State per Node	Learning Switch: O(#nodes)	Learning Switch: O(#nodes) + Path to Root: O(constant)
Convergence	No setup time at all!	Need to run spanning tree protocol before routing
Routing Efficiency	Broadcast Storms	Still sends new connections everywhere.

Do the packets go where they need to get efficiently — without wasting resources at switches?

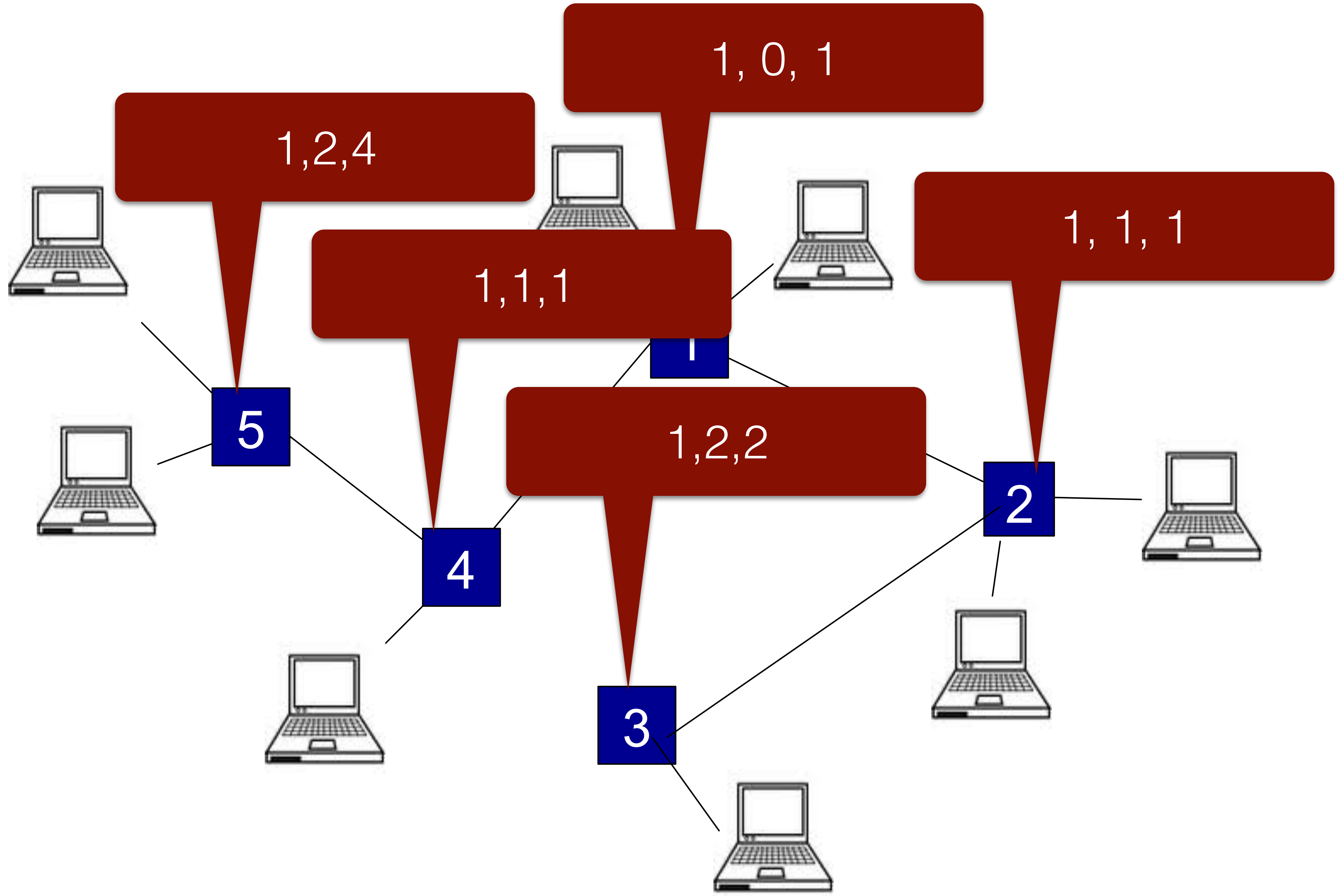
Trade-Offs

	Broadcast Network w/ Learning Switches	Broadcast Network w/ Learning Switches and Spanning Tree
Resilience	If there is a route, the packet will reach dest!	Need to recompute spanning tree if failure
Fully Distributed	Yes	Yes
State per Node	Learning Switch: O(#nodes)	Learning Switch: O(#nodes) + Path to Root: O(constant)
Convergence	No setup time at all!	Need to run spanning tree protocol before routing
Routing Efficiency	Broadcast Storms	Still sends new connections everywhere.
Shortest Path?	Not Necessarily…	Not Necessarily…

We know packets will reach their destination… but do they take the shortest path to get there?

What if 3 wants to communicate with 4?

What if 5 wants to communicate with 3?



Real World

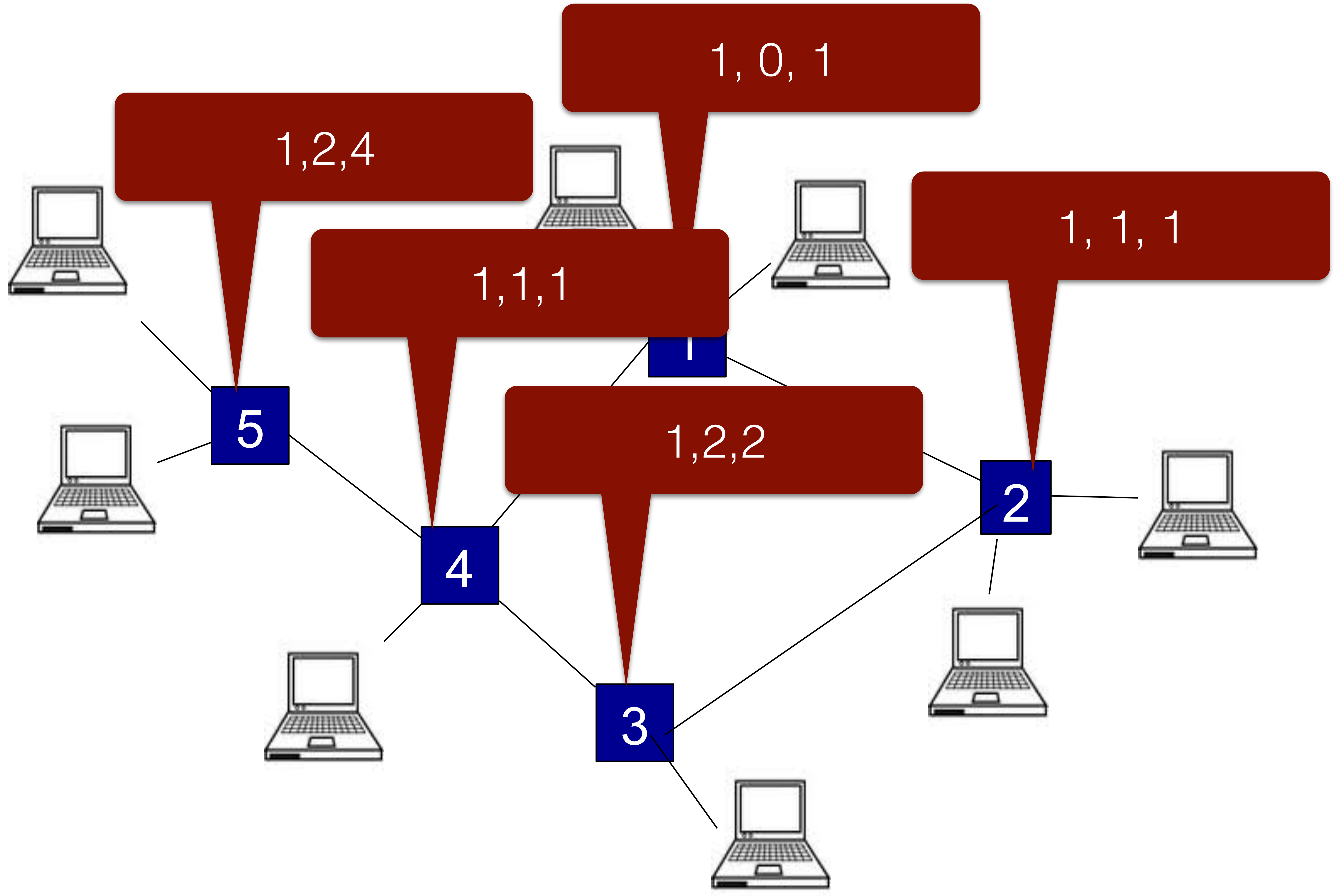
- We only use broadcast routing in very small networks.
  - One rack in the machine room.
  - A wing of one floor in GHC.
- To orchestrate the bigger network — across campus — we use other algorithms.
  - Why do you think that is?

Yet Another Algorithm…

	Broadcast Network w/ Learning Switches	Broadcast Network w/ Learning Switches and Spanning Tree	Distance Vector e.g RIP
Resilience	If there is a route, the packet will reach dest!	Need to recompute spanning tree if failure	
Fully Distributed	Yes	Yes	
State per Node	Learning Switch: O(#nodes)	Learning Switch: O(#nodes) + Path to Root: O(constant)	
Convergence	No setup time at all!	Need to run spanning tree protocol before routing	
Routing Efficiency	Broadcast Storms	Still sends new connections everywhere.	Packets sent directly to their destination.
Shortest Path?	Not Necessarily…	Not Necessarily…	Yes

Recall: Spanning Tree

There is exactly one node that every *does* have the shortest path to.



How Distance-Vector (DV) Works

Each router maintains its shortest distance to every destination v f its neighbors

Neighbor (next-hop)

A's Route Table

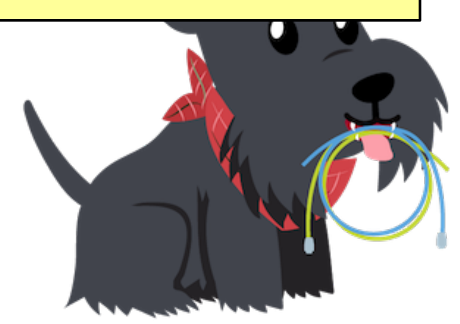
A

	via B	viaC
to B		
to C		
to D		

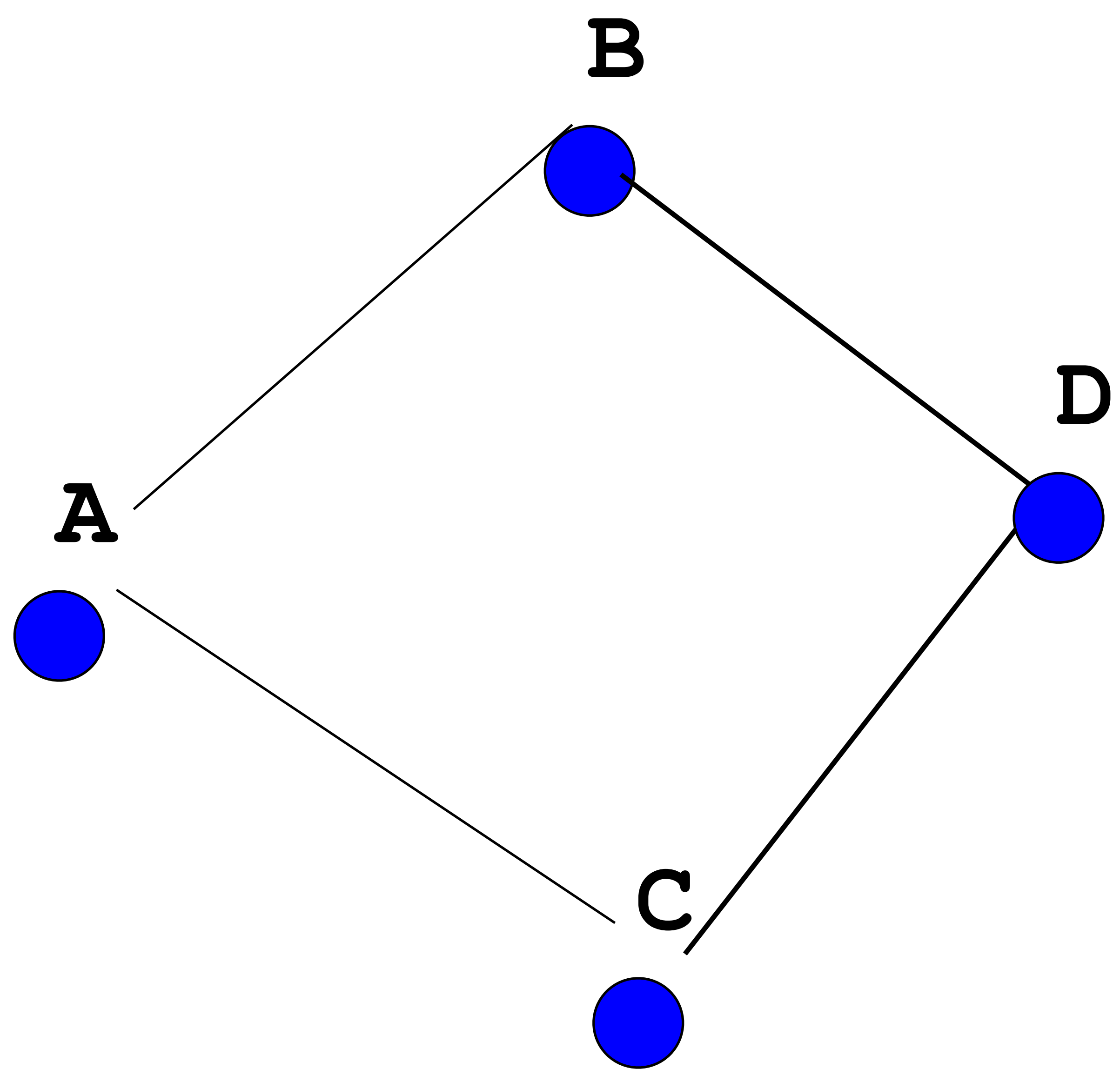
Destinations

dist$_C$(A, D): shortest distance from A to D via C

Each router computes its shortest distance to every destination via <u>any</u> of its neighbors



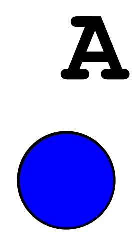
How Distance-Vector (DV) Works



How Distance-Vector (DV) Works

Each router maintains its shortest distance to every destination via each of its neighbors

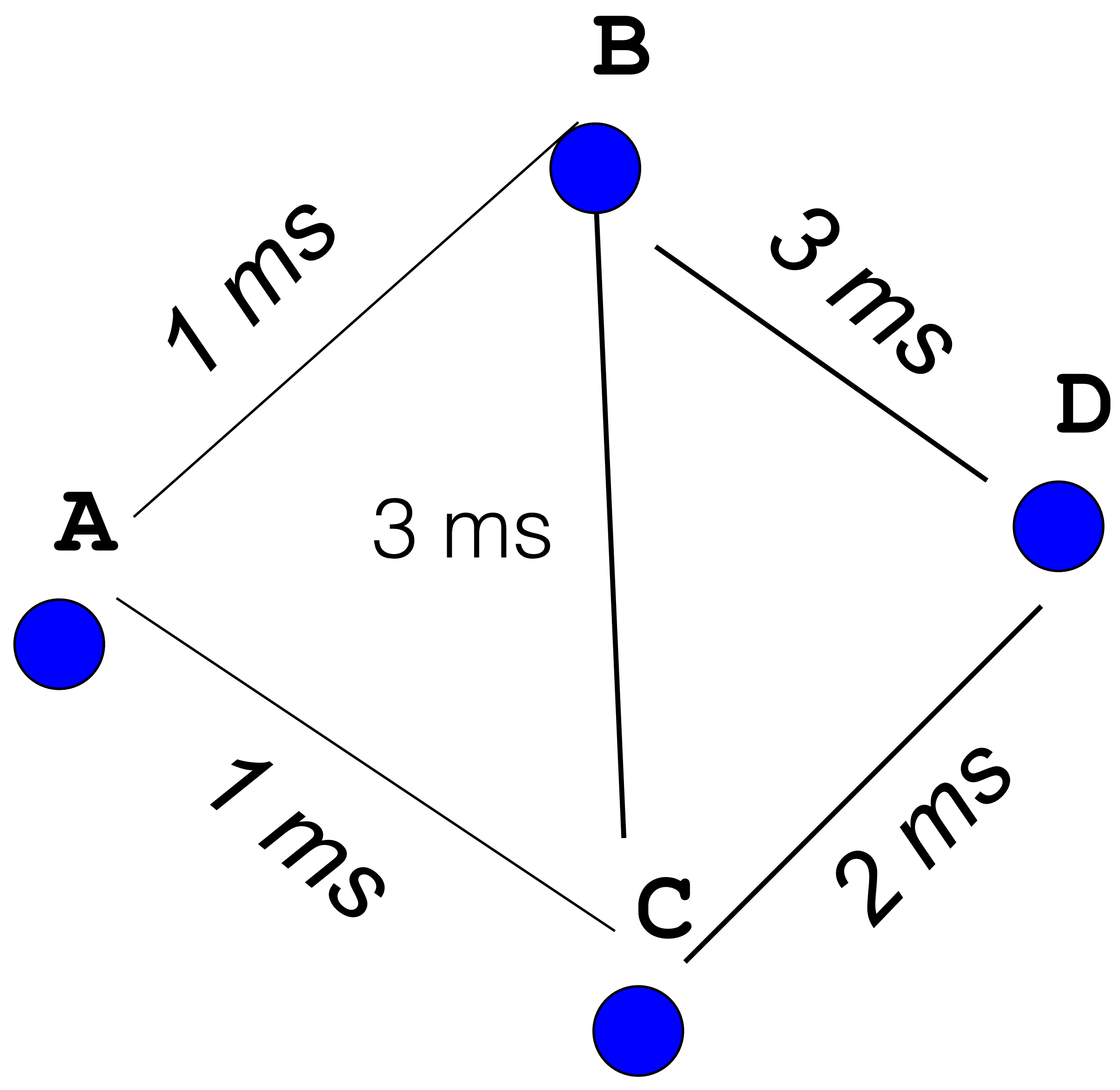
A's Route Table



	via B	viaC
to B	1	
to C		1
to D		

Each router computes its shortest distance to every destination via <u>any</u> of its neighbors

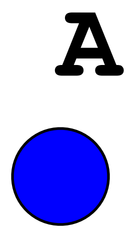
How Distance-Vector (DV) Works



Link distance doesn't have to be 1! Could be some other value — e.g., latency of the link

How Distance-Vector (DV) Works

Each router maintains its shortest distance to every destination via each of its neighbors



A's Route Table

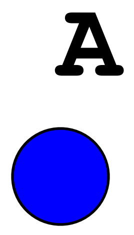
	via B	viaC
to B	1	4
to C	4	1
to D	4	3

Each router computes its shortest distance to every destination via <u>any</u> of its neighbors

How Distance-Vector (DV) Works

Routers send a summary of their tables to their neighbors.

This summary is called a “distance vector”



A’s Route Table

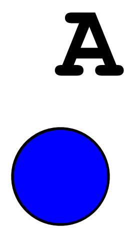
	via B	viaC
to B	1	4
to C	2	1
to D	4	3

A’s distance vector (DV)

	min dist
to A	
to B	
to C	
to D	

Each router computes its shortest distance to every destination via any of its neighbors

How Distance-Vector (DV) Works



A's Route Table

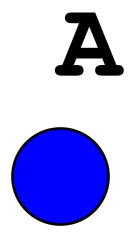
	via B	viaC
to B	1	4
to C	2	1
to D	4	3

A's distance vector (DV)

	min dist
to A	0
to B	
to C	
to D	

Update route to min(all of my B routes)

How Distance-Vector (DV) Works



A's Route Table

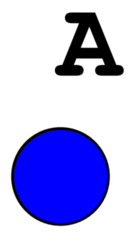
	via B	viaC
to B	1	4
to C	2	1
to D	4	3

A's distance vector (DV)

	min dist
to A	0
to B	1
to C	1
to D	

Update route to min(all of my C routes)

How Distance-Vector (DV) Works



A's Route Table

	via B	viaC
to B	1	4
to C	2	1
to D	4	3

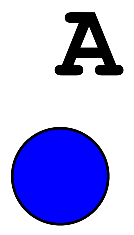
A's distance vector (DV)

	min dist
to A	0
to B	1
to C	1
to D	3

Update route to min(all of my D routes)

How Distance-Vector (DV) Works

B's DV

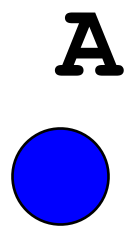


A's Route Table

	via B	viaC
to B	1	∞
to C	∞	1
to D	∞	∞

But, when we start the table is mostly empty…
We have to learn by receiving DV's from others.

How Distance-Vector (DV) Works



A's Route Table

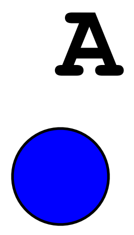
	via B	viaC
to B	1	∞
to C	∞	1
to D	∞	∞

B's DV

	mindist
to A	1
to C	3
to D	2

But, when we start the table is mostly empty…
We have to learn by receiving DV's from others.

How Distance-Vector (DV) Works



A's Route Table

	via B	viaC
to B	1	∞
to C	4	1
to D	3	∞

B's DV

	mindist
to A	1
to C	3
to D	2

But, when we start the table is mostly empty…
We have to learn by receiving DV's from others.

Distance Vector Routing: Summary

- Each router knows the links to its neighbors
- Each router has provisional "shortest path" to every other router -- its distance vector (DV)
- Routers exchange this DV with their neighbors
- Routers look over the set of options offered by their neighbors and select the best one
- Iterative process converges to set of shortest paths

Tricky Question

- Let's assume our DV algorithm runs in "rounds"
  - In lock-step, all routers send out a DV to their neighbors
  - Then they update their tables — all at the same time! — with the new information they have received.
  - Then, in lock-step, they all send out a DV at the same time. (Repeat)
- **Q: How many "rounds" will it take for the DV algorithm to *converge*?**

Intuition

- Initial state: best one-hop paths
- One simultaneous round: best two-hop paths
- Two simultaneous rounds: best three-hop paths
- …
- Kth simultaneous round: best (k+1) hop paths
- Must eventually converge
  - <u>as soon as it reaches longest best path</u>

	Broadcast Network w/ Learning Switches	Broadcast Network w/ Learning Switches and Spanning Tree	Distance Vector e.g RIP
Resilience	If there is a route, the packet will reach dest!	Need to recompute spanning tree if failure	
Fully Distributed	Yes	Yes	
State per Node	Learning Switch: O(#nodes)	Learning Switch: O(#nodes) + Path to Root: O(constant)	
Convergence	No setup time at all!	Need to run spanning tree protocol before routing	
Routing Efficiency	Broadcast Storms	Still sends new connections everywhere.	Packets sent directly to their destination.
Shortest Path?	Not Necessarily…	Not Necessarily…	Yes

	Broadcast Network w/ Learning Switches	Broadcast Network w/ Learning Switches and Spanning Tree	Distance Vector e.g RIP
Resilience	If there is a route, the packet will reach dest!	Need to recompute spanning tree if failure	
Fully Distributed	Yes	Yes	
State per Node	Learning Switch: O(#nodes)	Learning Switch: O(#nodes) + Path to Root: O(constant)	
Convergence	No setup time at all!	Need to run spanning tree protocol before routing	Need to run DV before routing — takes length of longest best path time.
Routing Efficiency	Broadcast Storms	Still sends new connections everywhere.	Packets sent directly to their destination.
Shortest Path?	Not Necessarily…	Not Necessarily…	Yes

	Broadcast Network w/ Learning Switches	Broadcast Network w/ Learning Switches and Spanning Tree	Distance Vector e.g RIP
Resilience	If there is a route, the packet will reach dest!	Need to recompute spanning tree if failure	
Fully Distributed	Yes	Yes	
State per Node	Learning Switch: O(#nodes)	Learning Switch: O(#nodes) + Path to Root: O(constant)	O(# switches * max node degree)
Convergence	No setup time at all!	Need to run spanning tree protocol before routing	Need to run DV before routing — takes length of longest best path time.
Routing Efficiency	Broadcast Storms	Still sends new connections everywhere.	Packets sent directly to their destination.
Shortest Path?	Not Necessarily…	Not Necessarily…	Yes

	Broadcast Network w/ Learning Switches	Broadcast Network w/ Learning Switches and Spanning Tree	Distance Vector e.g RIP
Resilience	If there is a route, the packet will reach dest!	Need to recompute spanning tree if failure	
Fully Distributed	Yes	Yes	Yes
State per Node	Learning Switch: O(#nodes)	Learning Switch: O(#nodes) + Path to Root: O(constant)	O(# switches * max node degree)
Convergence	No setup time at all!	Need to run spanning tree protocol before routing	Need to run DV before routing — takes length of longest best path time.
Routing Efficiency	Broadcast Storms	Still sends new connections everywhere.	Packets sent directly to their destination.
Shortest Path?	Not Necessarily…	Not Necessarily…	Yes

	Broadcast Network w/ Learning Switches	Broadcast Network w/ Learning Switches and Spanning Tree	Distance Vector e.g RIP
Resilience	If there is a route, the packet will reach dest!	Need to recompute spanning tree if failure	**I have some bad news.**
Fully Distributed	Yes	Yes	Yes
State per Node	Learning Switch: O(#nodes)	Learning Switch: O(#nodes) + Path to Root: O(constant)	O(# switches * max node degree)
Convergence	No setup time at all!	Need to run spanning tree protocol before routing	Need to run DV before routing — takes length of longest best path time.
Routing Efficiency	Broadcast Storms	Still sends new connections everywhere.	Packets sent directly to their destination.
Shortest Path?	Not Necessarily…	Not Necessarily…	Yes

Before the bad news… try it out.

A's Route Table

	via B	viaC
to B	1	2
to C	4	3
to D	2	6

A's DV

	mindist
to B	?
to C	?
to D	?

A

What values does A announce in it's Distance Vector?

C

C's Route Table

	via B	via D
to A	1	5
to B	1	7
to D	2	3

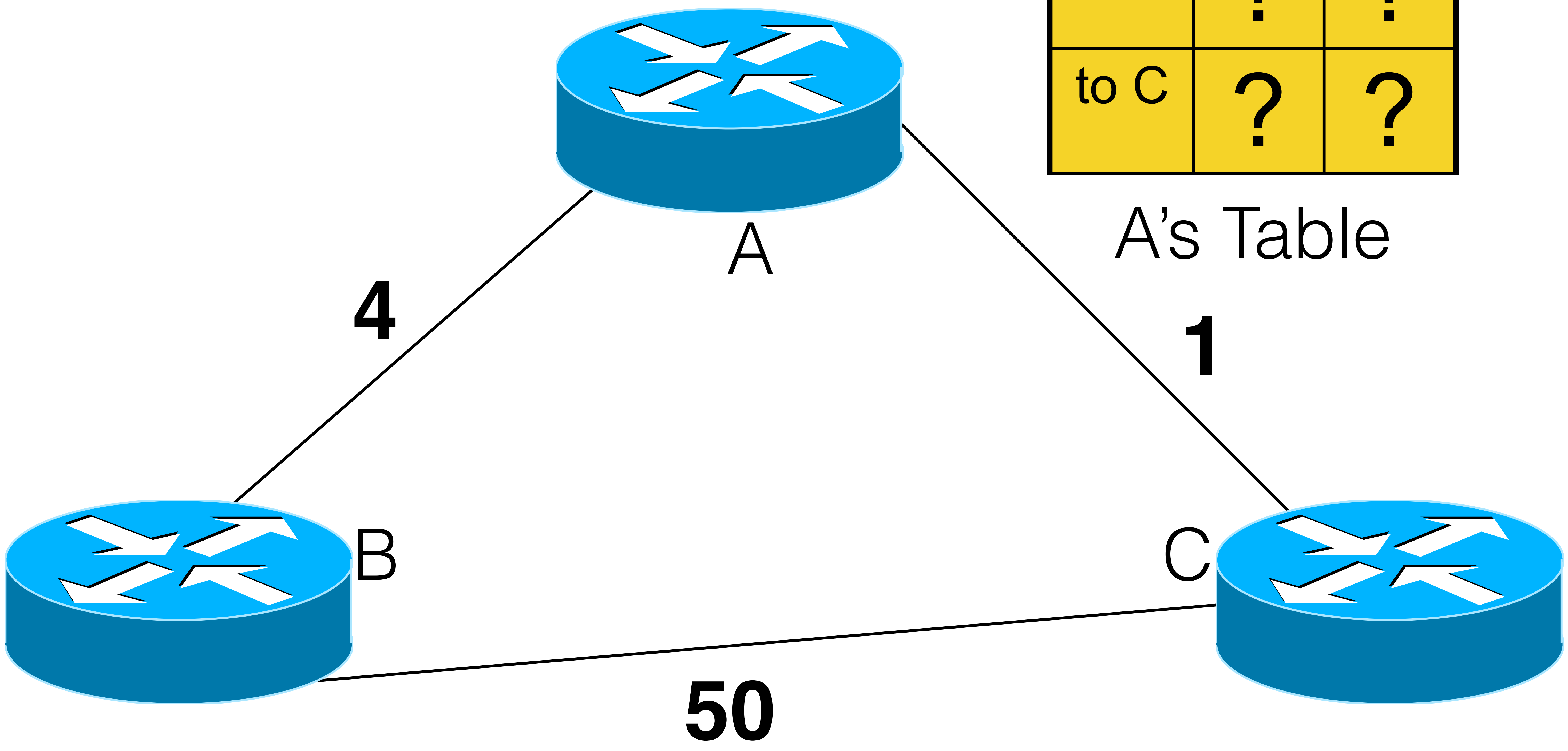
	mindist
to A	4
to B	2
to D	0

C receives the above DV from its neighbor D. How does it change its routing table?
(Assume the link weight from C to D is 3)

	Broadcast Network w/ Learning Switches	Broadcast Network w/ Learning Switches and Spanning Tree	Distance Vector e.g RIP
Resilience	If there is a route, the packet will reach dest!	Need to recompute spanning tree if failure	**I have some bad news.**
Fully Distributed	Yes	Yes	Yes
State per Node	Learning Switch: O(#nodes)	Learning Switch: O(#nodes) + Path to Root: O(constant)	O(# switches * max node degree) + O(#nodes)
Convergence	No setup time at all!	Need to run spanning tree protocol before routing	Need to run DV before routing — takes length of longest best path time.
Routing Efficiency	Broadcast Storms	Still sends new connections everywhere.	Packets sent directly to their destination.
Shortest Path?	Not Necessarily…	Not Necessarily…	Yes

Distance Vector Algorithms suffer long convergence times when link weights *increase* or when *links go down.*

Running into trouble…



	via B	viaC
to B	?	?
to C	?	?

A’s Table

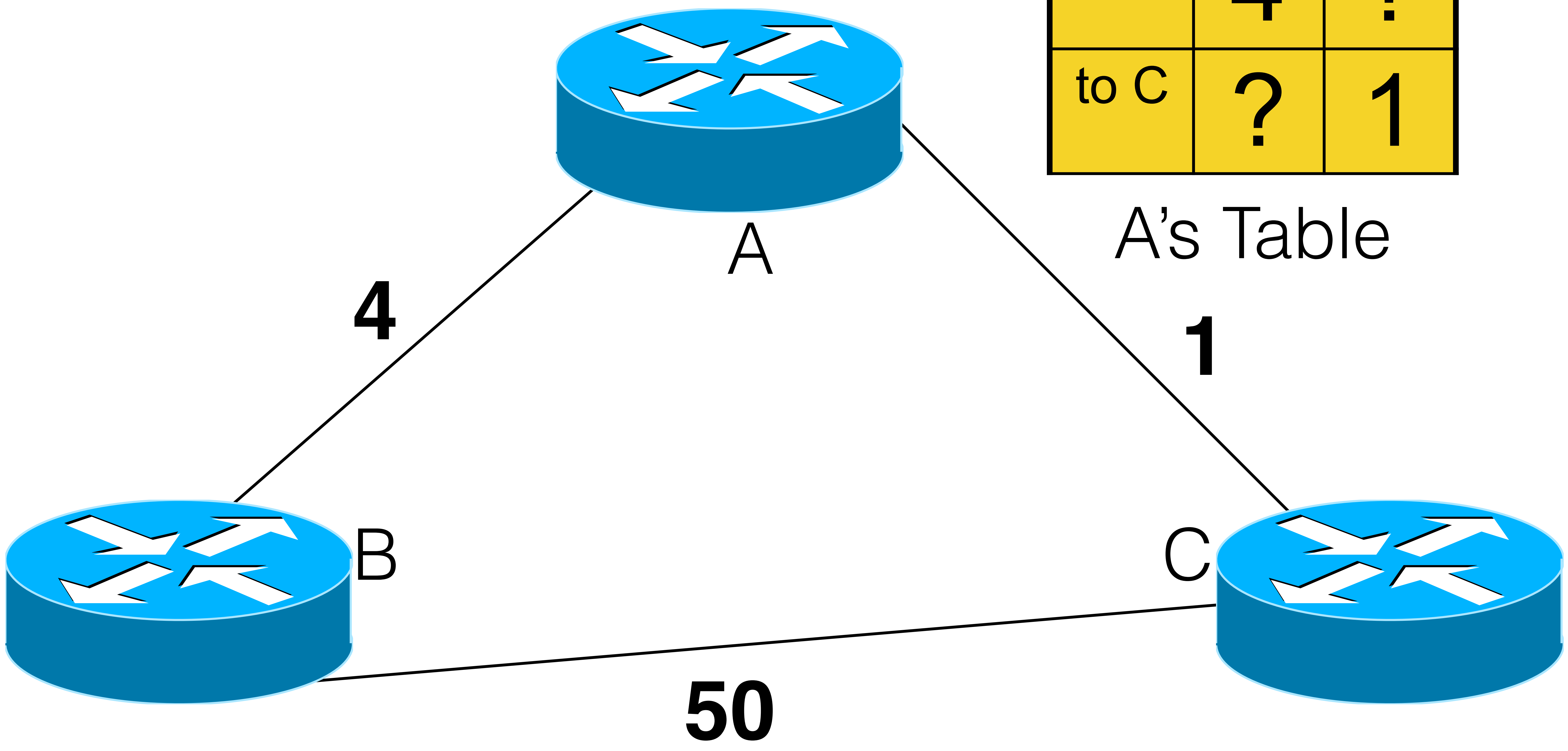
	via A	viaC
to A	?	?
to C	?	?

B’s Table

	via A	via B
to A	?	?
to B	?	?

C’s Table

Running into trouble…



	via B	viaC
to B	4	?
to C	?	1

A's Table

	via A	via B
to A	1	?
to B	?	50

C's Table

	via A	viaC
to A	4	?
to C	?	50

B's Table

Running … trouble…

DV Update!

To B: 4
To C: 1

	via B	viaC
to B	4	?
to C	?	1

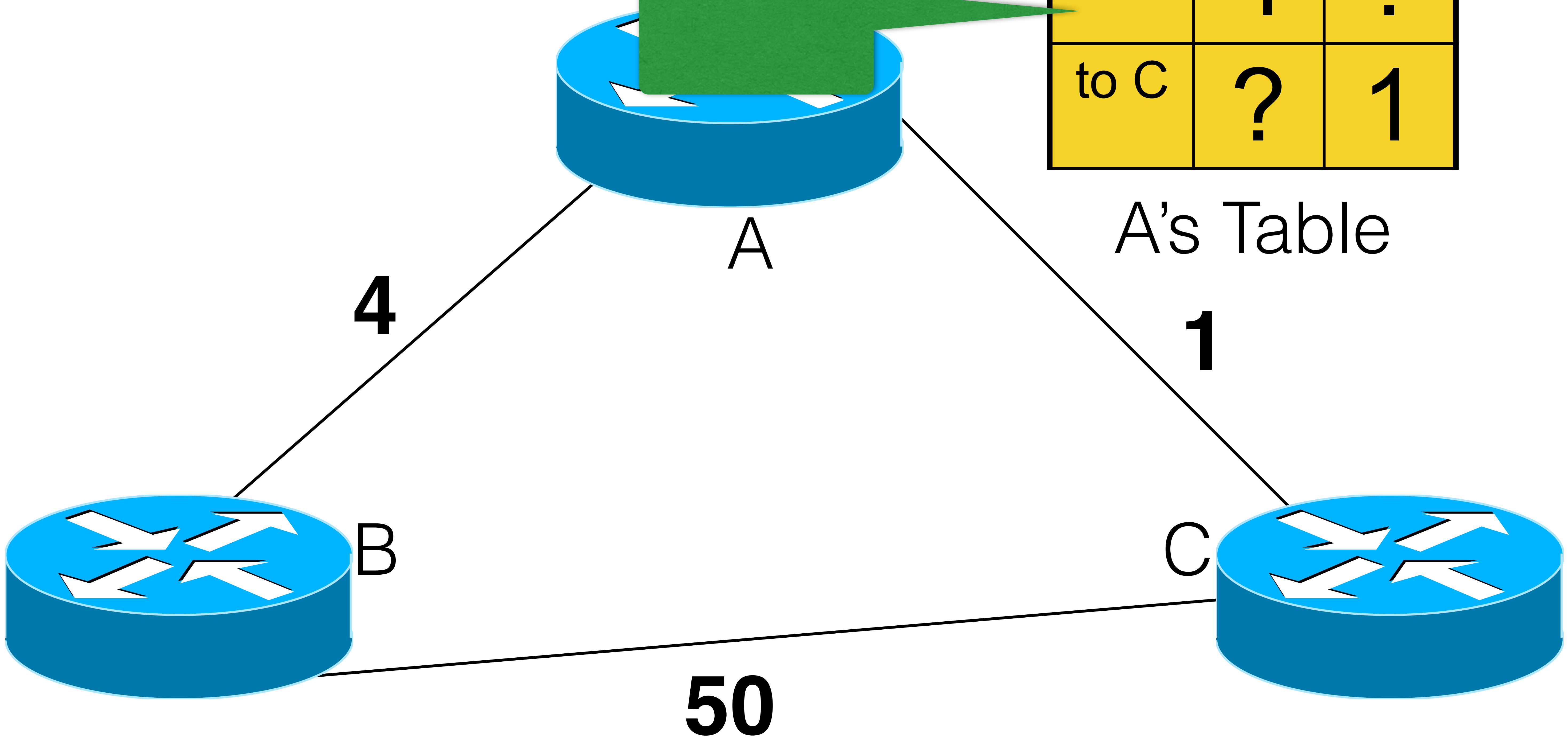
A's Table

DV Update!

To A: 4
To C: 50

	via A	viaC
to A	4	?
to C	?	50

B's Table



DV Update!

To A: 1
To B: 50

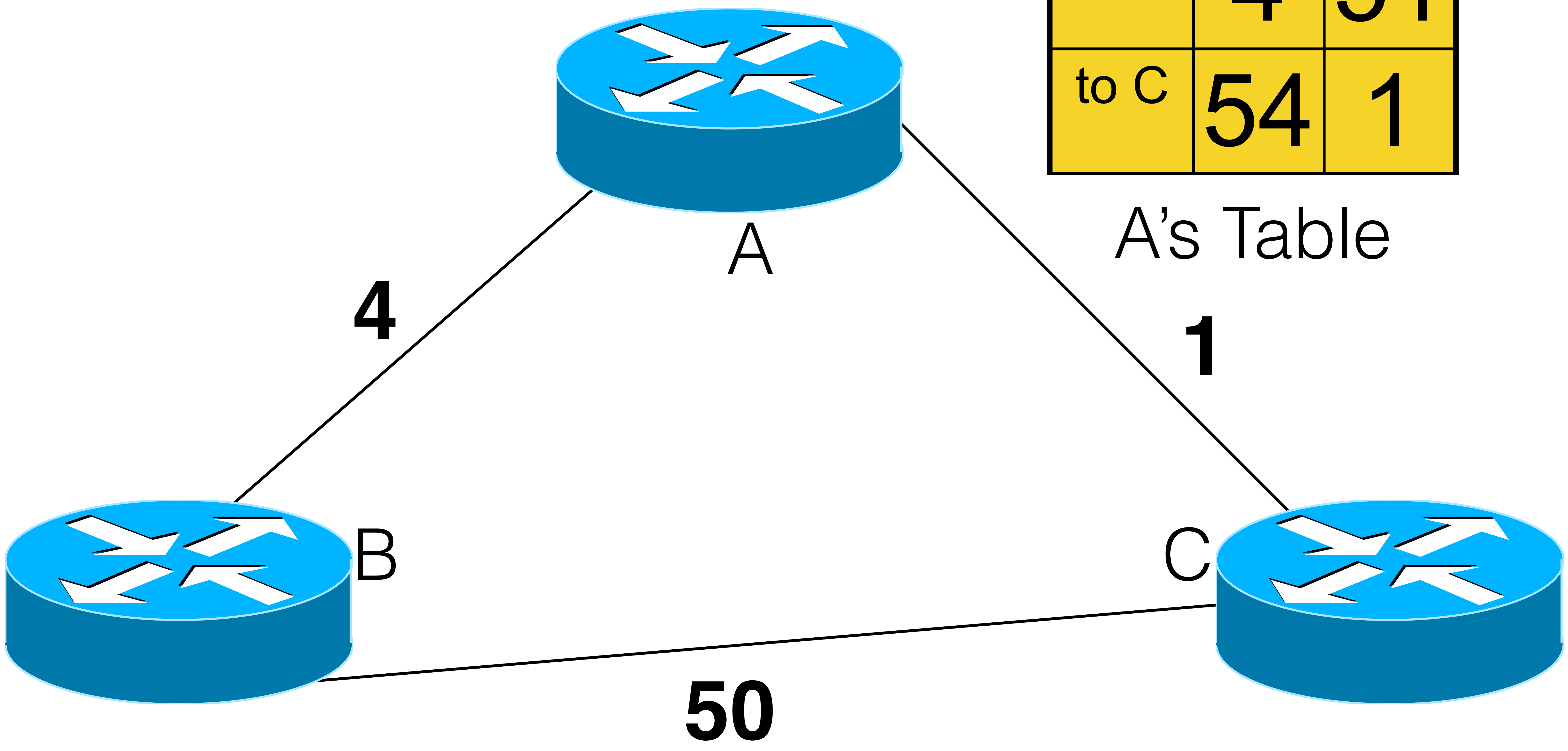
	via A	via B
to A	1	?
to B	?	50

C's Table

Running into trouble…

	via B	viaC
to B	4	51
to C	54	1

A's Table



	via A	viaC
to A	4	51
to C	5	50

B's Table

	via A	via B
to A	1	54
to B	5	50

C's Table

Running trouble…

DV Update!

To A: 4
To C: 5

	via A	viaC
to A	4	51
to C	5	50

B's Table

DV Update!

To B: 4
To C: 1

	via B	viaC
to B	4	51
to C	54	1

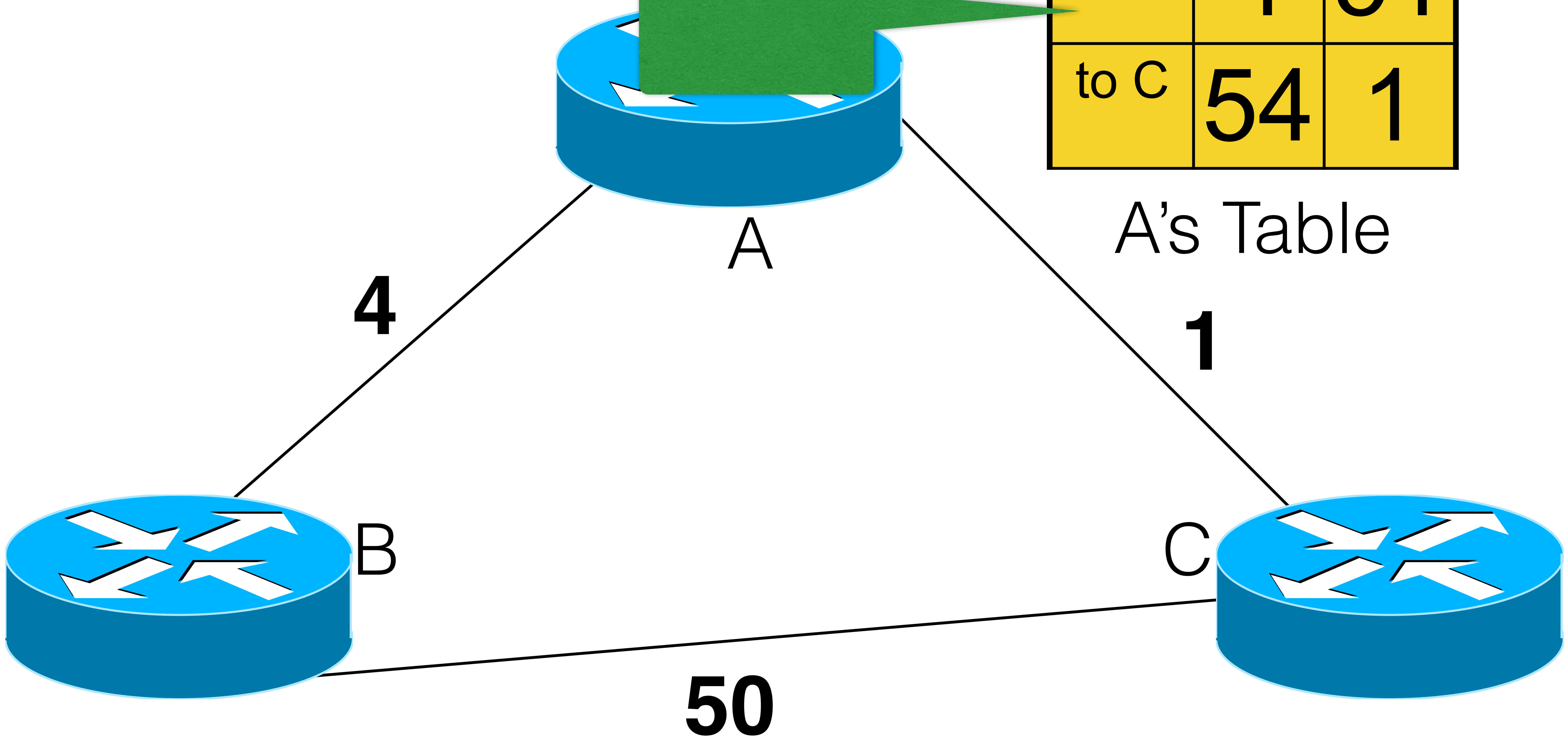
A's Table

DV Update!

To A: 1
To B: 5

	via A	via B
to A	1	54
to B	5	50

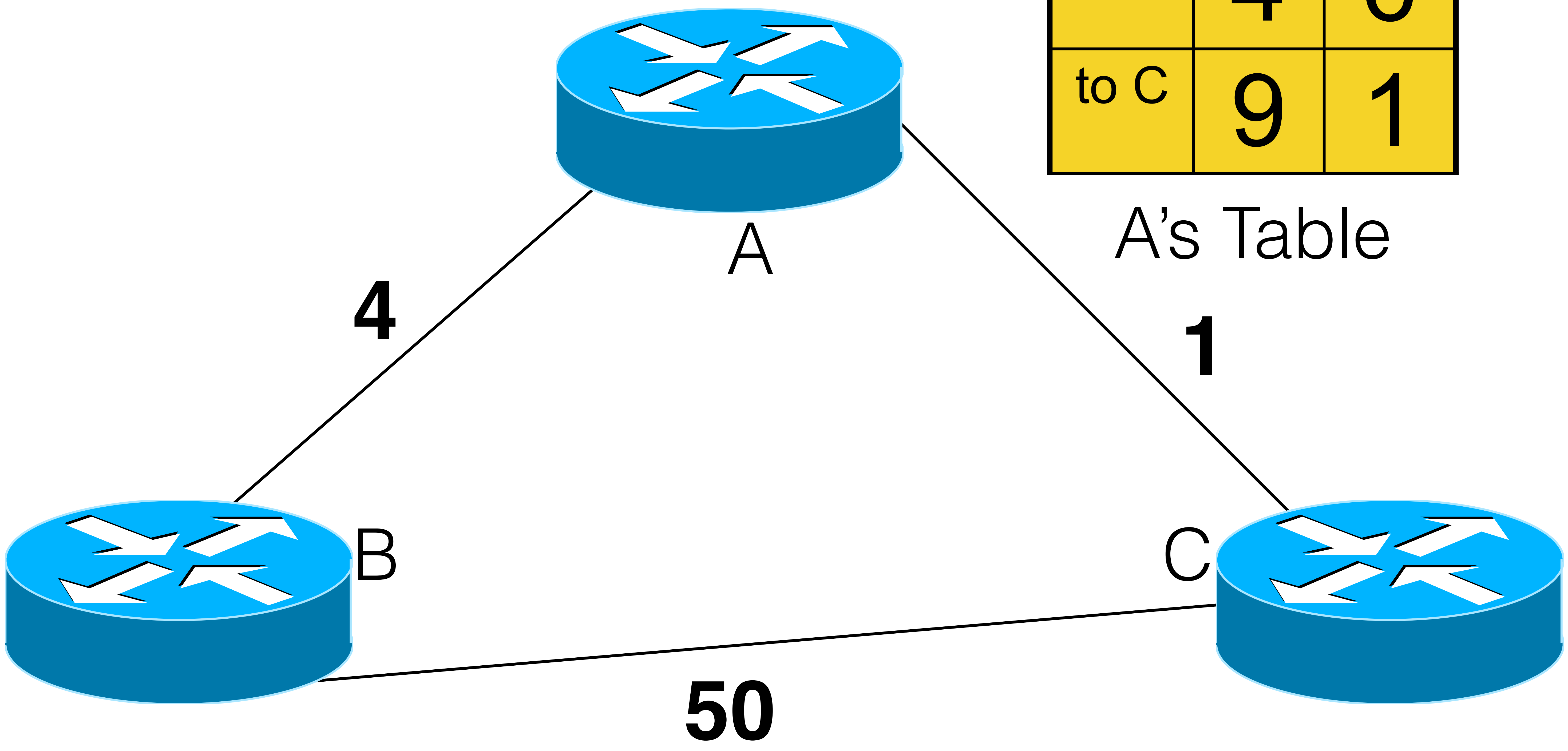
C's Table



Running into trouble…

	via B	viaC
to B	4	6
to C	9	1

A's Table



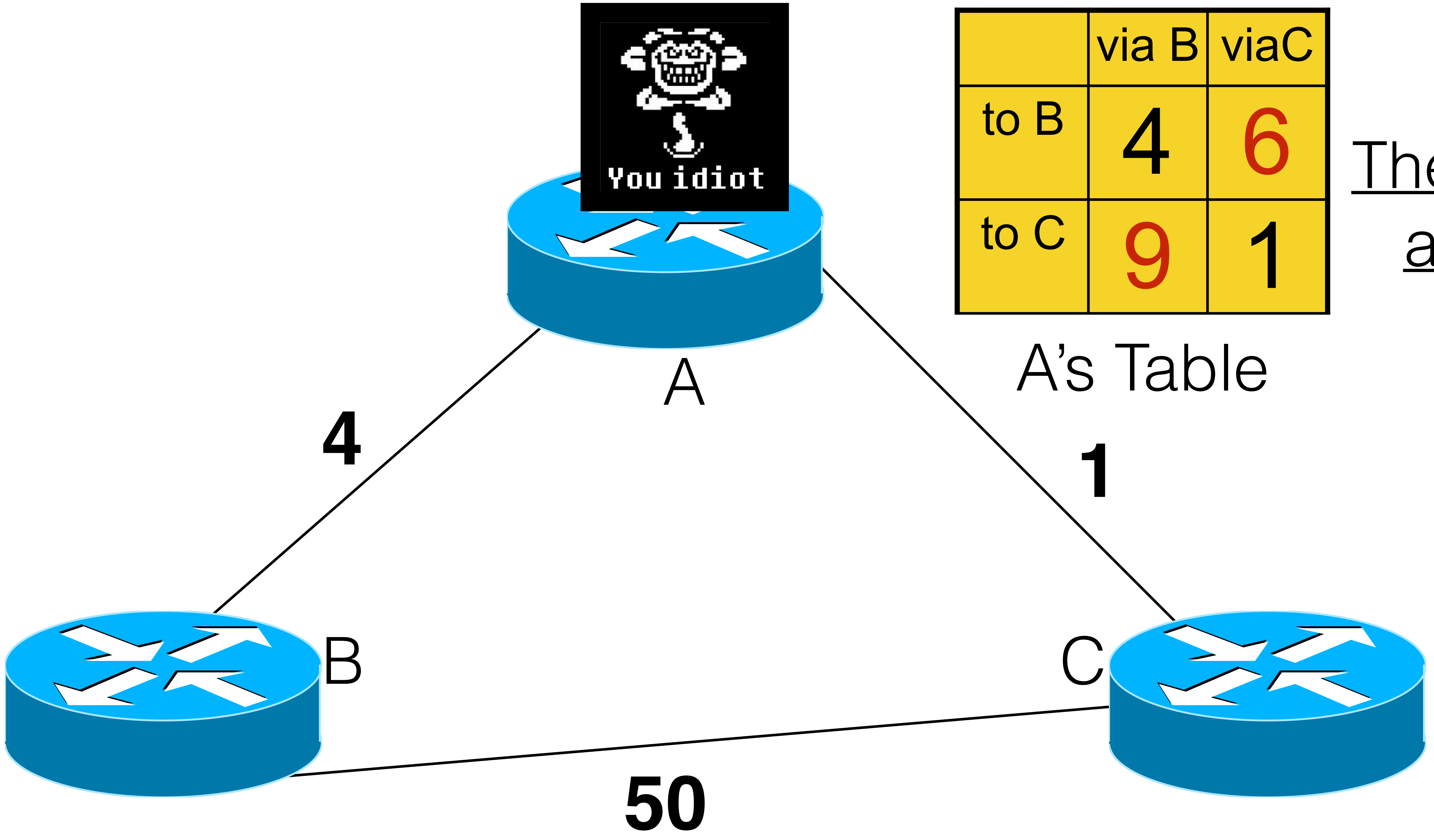
	via A	via B
to A	1	54
to B	5	50

C's Table

	via A	viaC
to A	4	51
to C	5	50

B's Table

Running into trouble…



	via B	viaC
to B	4	6
to C	9	1

A's Table

These routes have a loop in them.

	via A	viaC
to A	4	51
to C	5	50

B's Table

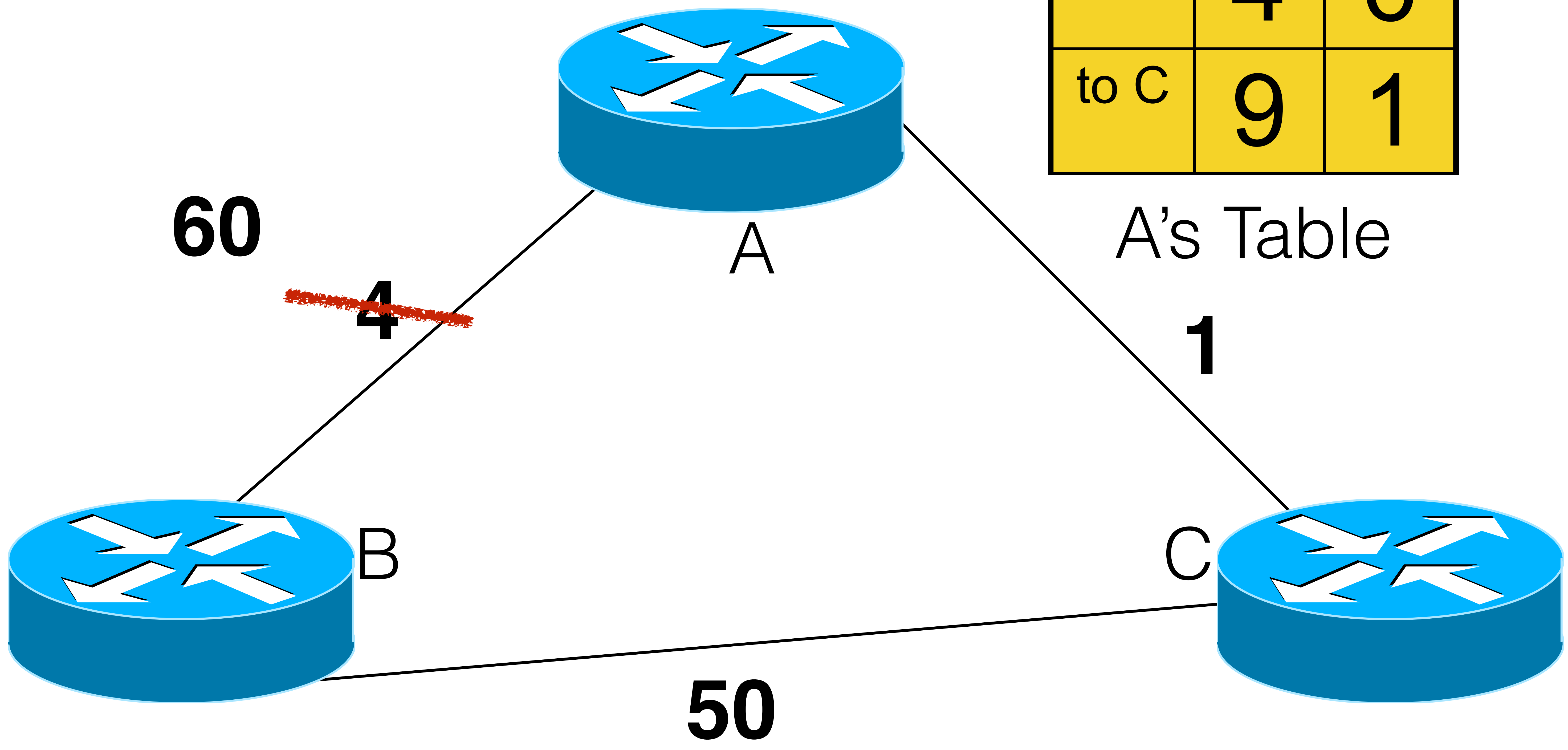
	via A	via B
to A	1	54
to B	5	50

C's Table

Loopy Routes

- These route are *fine* under normal operations — because they don't get used.
    - Why take the loopy route when you can take the direct path?
- But when link *updates* happen, bad things can happen.
    - If a link becomes more expensive.
    - If a link fails.

Running into trouble…



	via B	viaC
to B	4	6
to C	9	1

A’s Table

	via A	via B
to A	1	54
to B	5	50

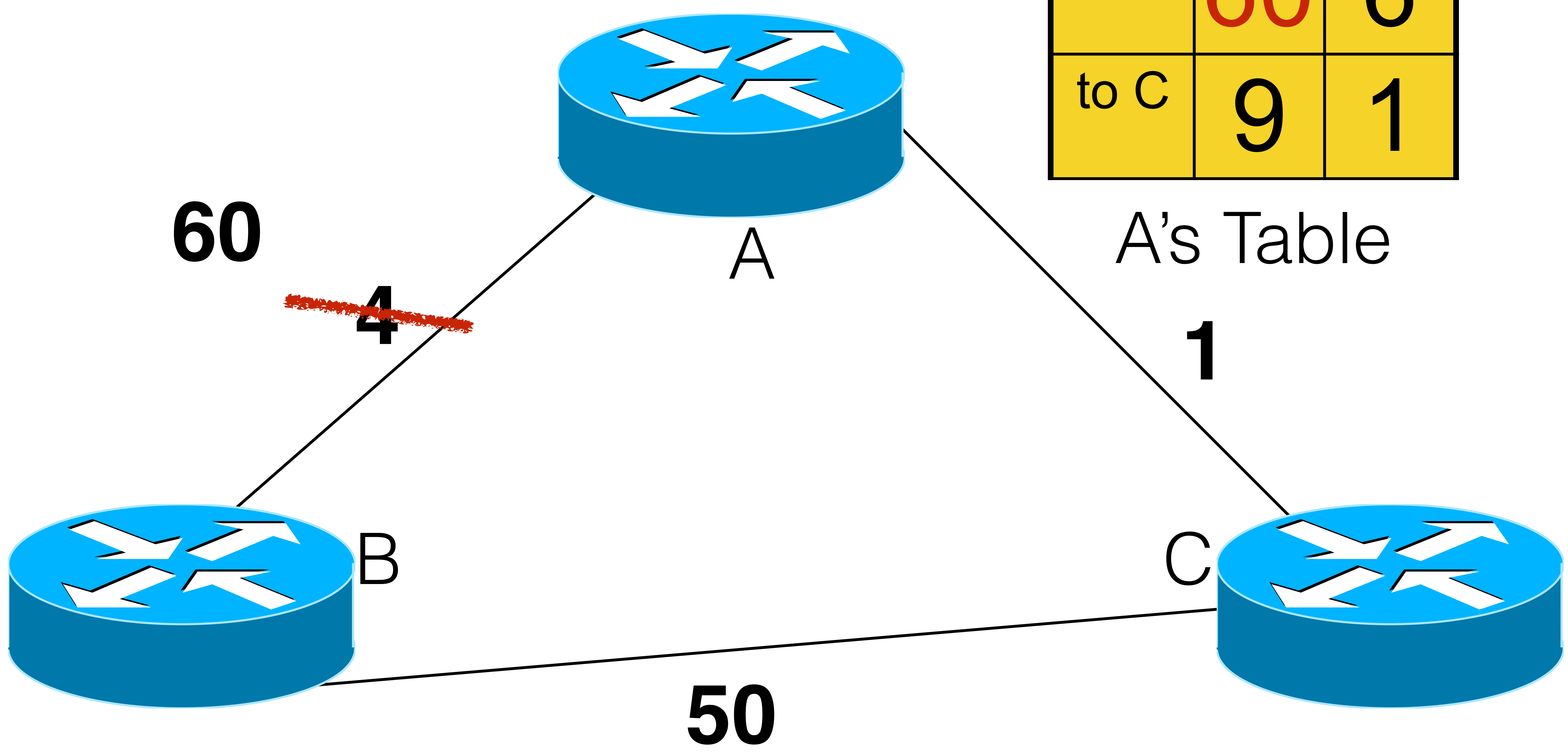
C’s Table

	via A	viaC
to A	4	51
to C	5	50

B’s Table

DO IT YOURSELF

Running into trouble…



	via B	viaC
to B	60	6
to C	9	1

A's Table

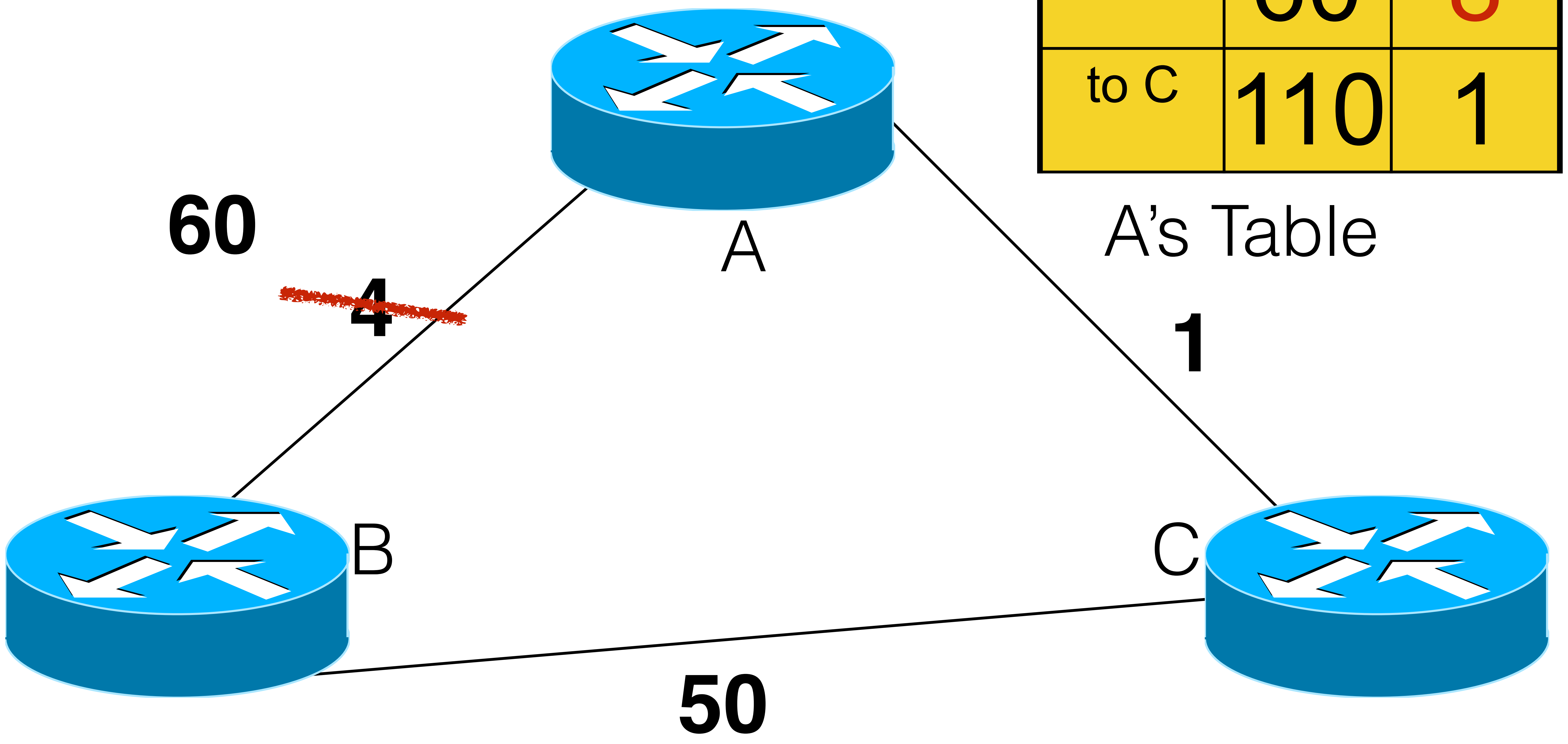
	via A	via B
to A	1	54
to B	5	50

C's Table

	via A	viaC
to A	60	51
to C	5	50

B's Table

Running into trouble…



	via B	viaC
to B	60	8
to C	110	1

A's Table

	via A	via B
to A	1	101
to B	7	50

C's Table

	via A	viaC
to A	60	51
to C	61	50

B's Table

Running trouble…

DV Update!

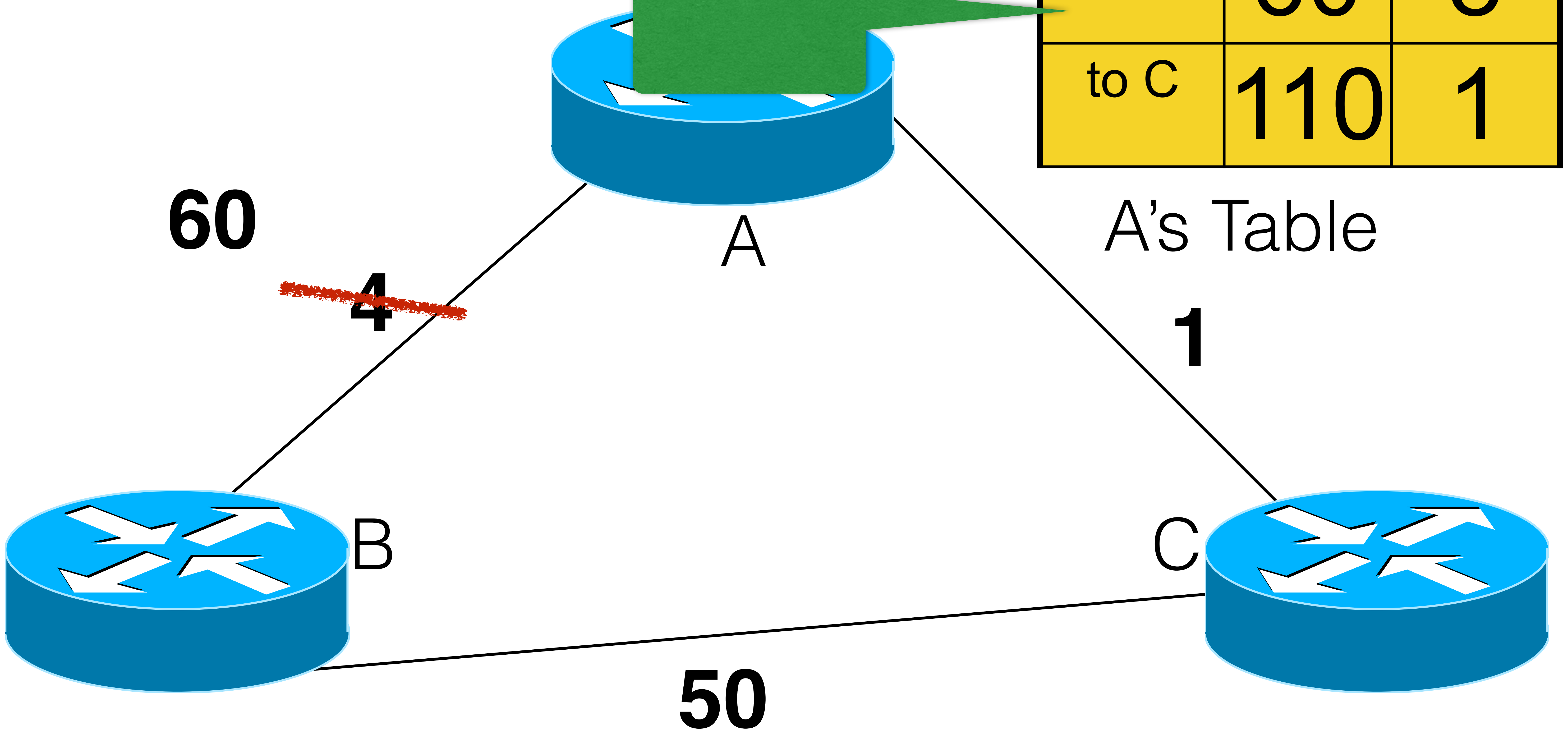
To B: 8
To C: 1

	via B	viaC
to B	60	8
to C	110	1

A's Table

DV Update!

To A: 51
To C: 50



	via A	viaC
to A	60	51
to C	61	50

B's Table

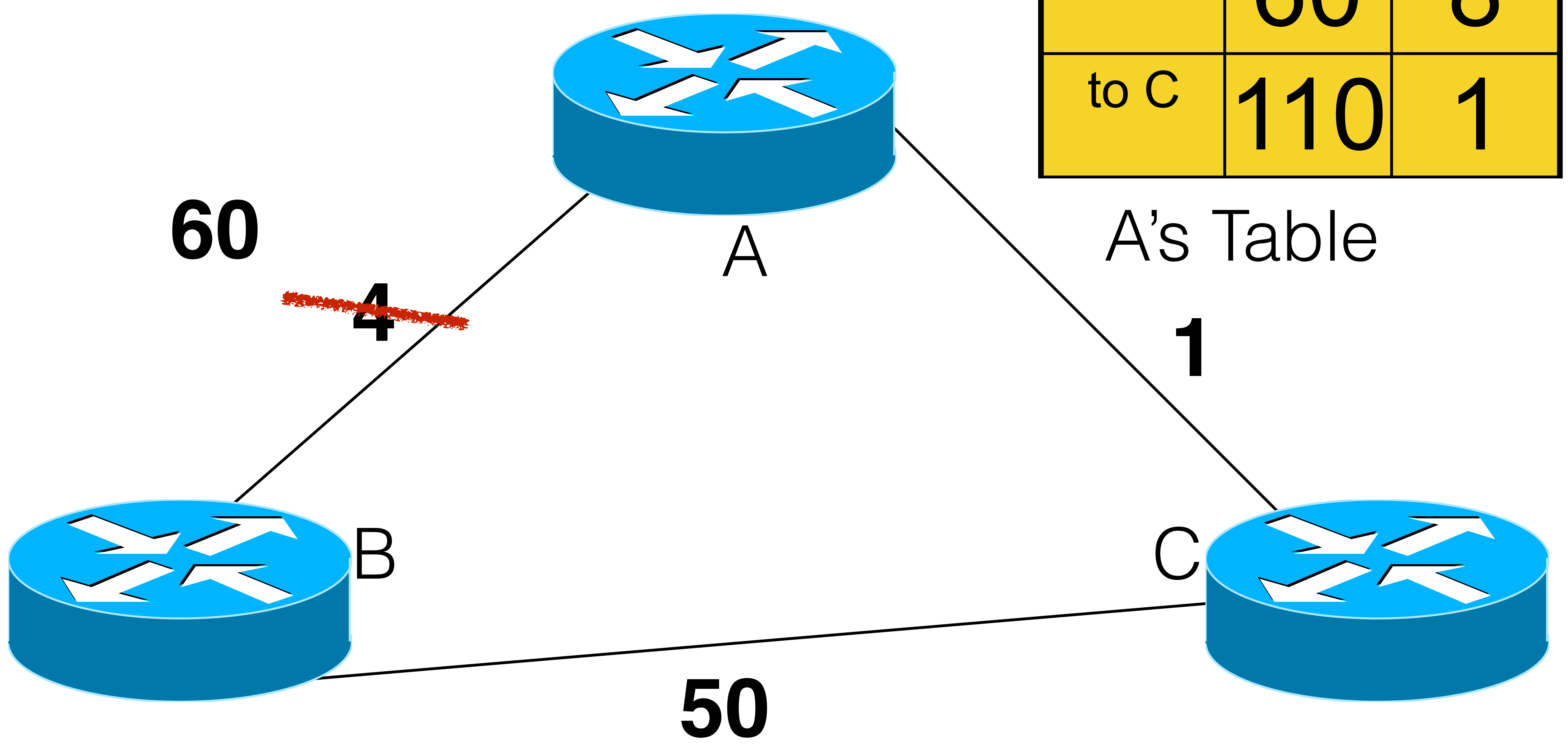
DV Update!

To A: 1
To B: 7

	via A	via B
to A	1	101
to B	7	50

C's Table

Running into trouble…



	via B	viaC
to B	60	8
to C	110	1

A's Table

	via A	via B
to A	1	101
to B	9	50

C's Table

	via A	viaC
to A	60	51
to C	61	50

B's Table

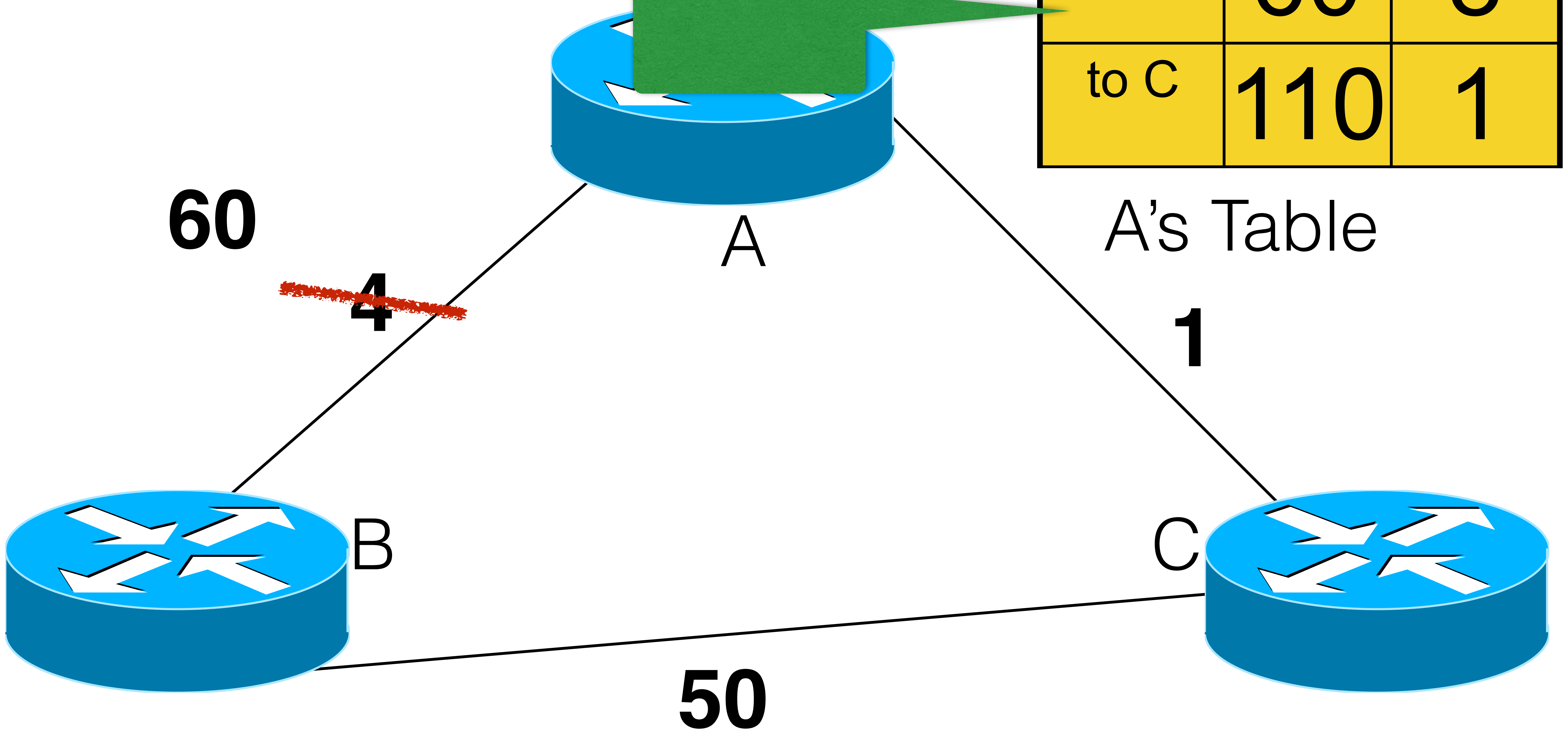
Running … trouble…

DV Update!

To B: 8
To C: 1

	via B	viaC
to B	60	8
to C	110	1

A's Table



DV Update!

To A: 51
To C: 50

	via A	viaC
to A	60	51
to C	61	50

B's Table

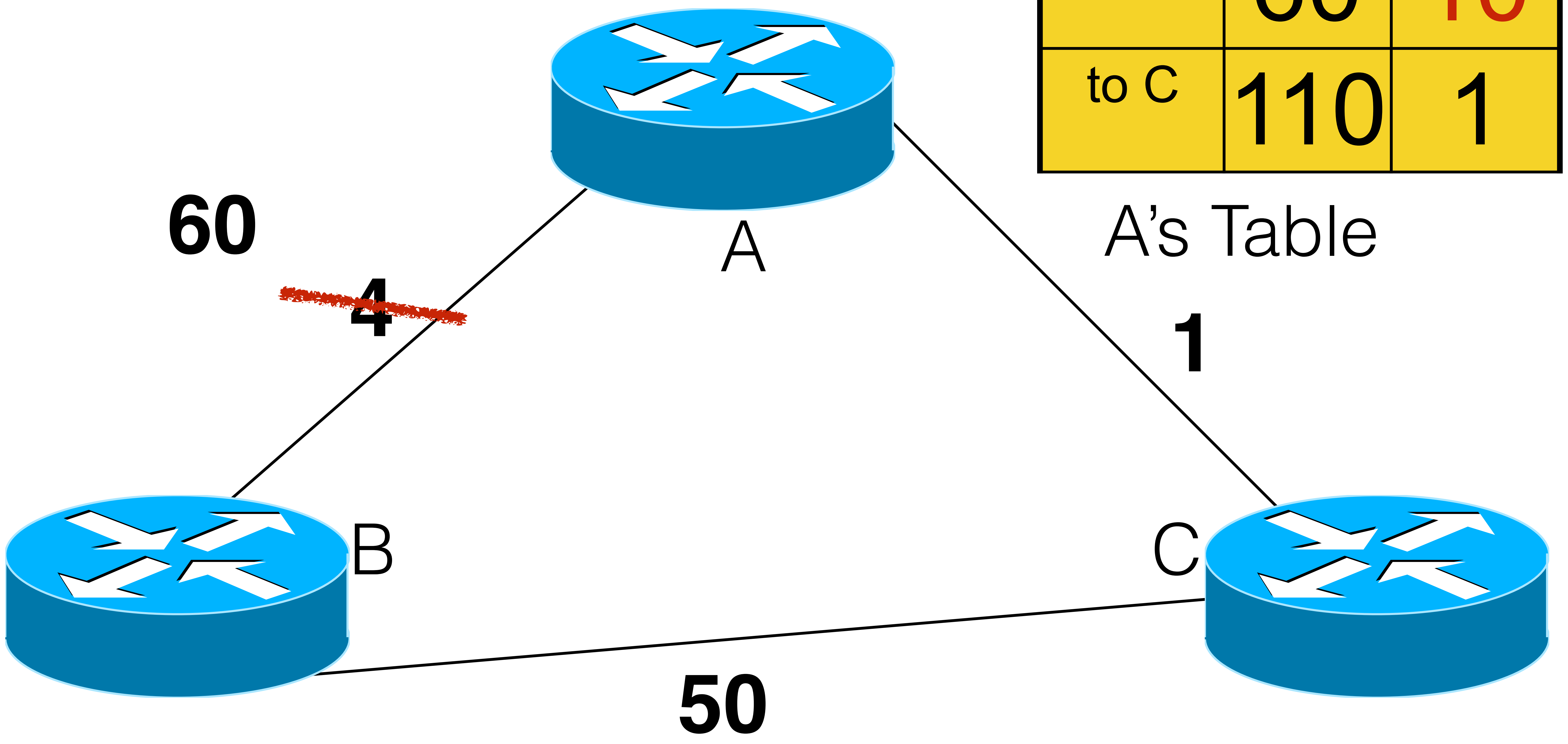
DV Update!

To A: 1
To B: 9

	via A	via B
to A	1	101
to B	9	50

C's Table

Running into trouble…



	via B	viaC
to B	60	10
to C	110	1

A's Table

	via A	via B
to A	1	101
to B	9	50

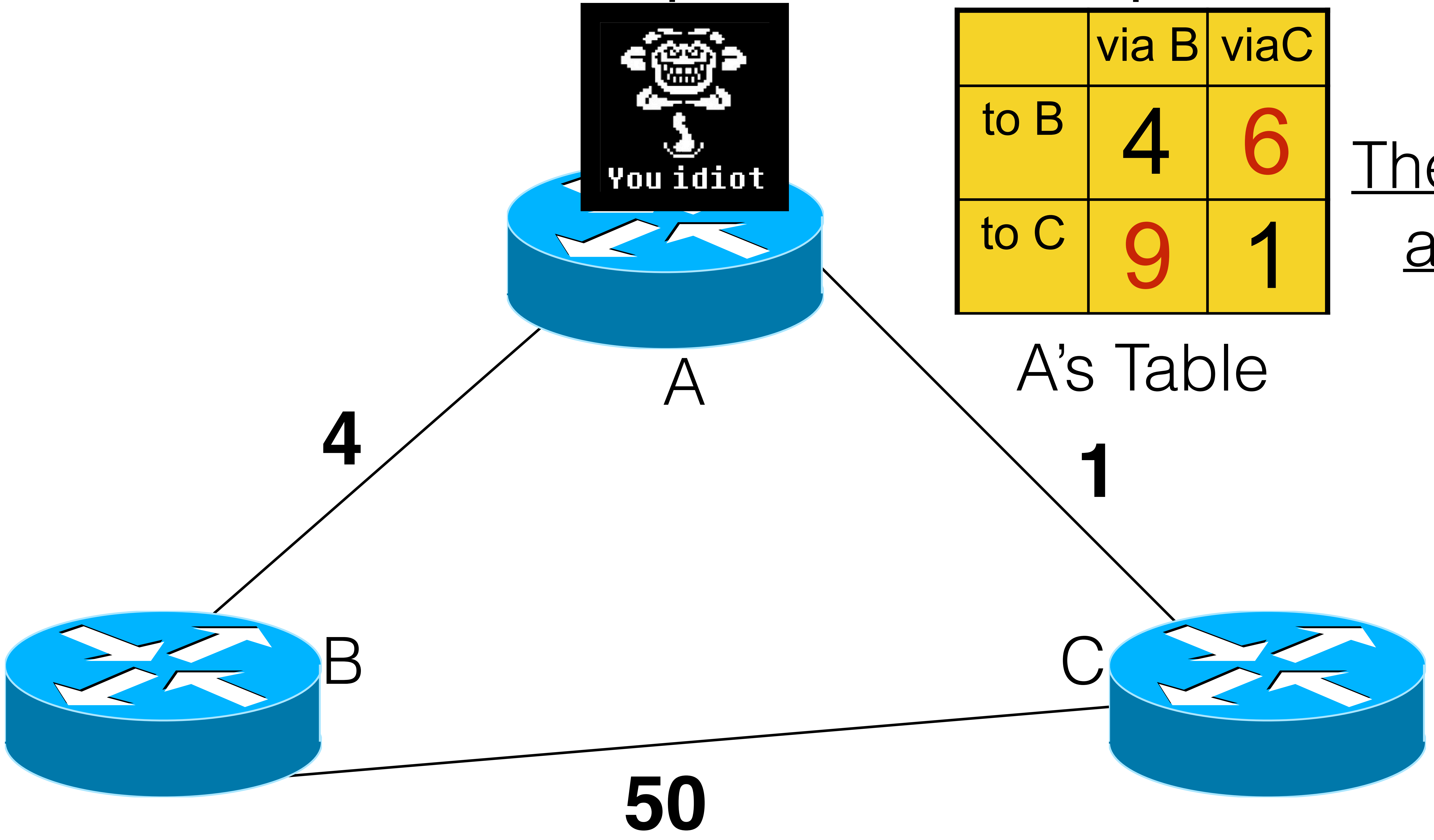
C's Table

	via A	viaC
to A	60	51
to C	61	50

B's Table

This is called the “count to infinity” problem.

Root of the Problem: DV algorithm has no way to detect and prevent loops.



	via B	viaC
to B	4	6
to C	9	1

A’s Table

These routes have a loop in them.

	via A	via B
to A	1	54
to B	5	50

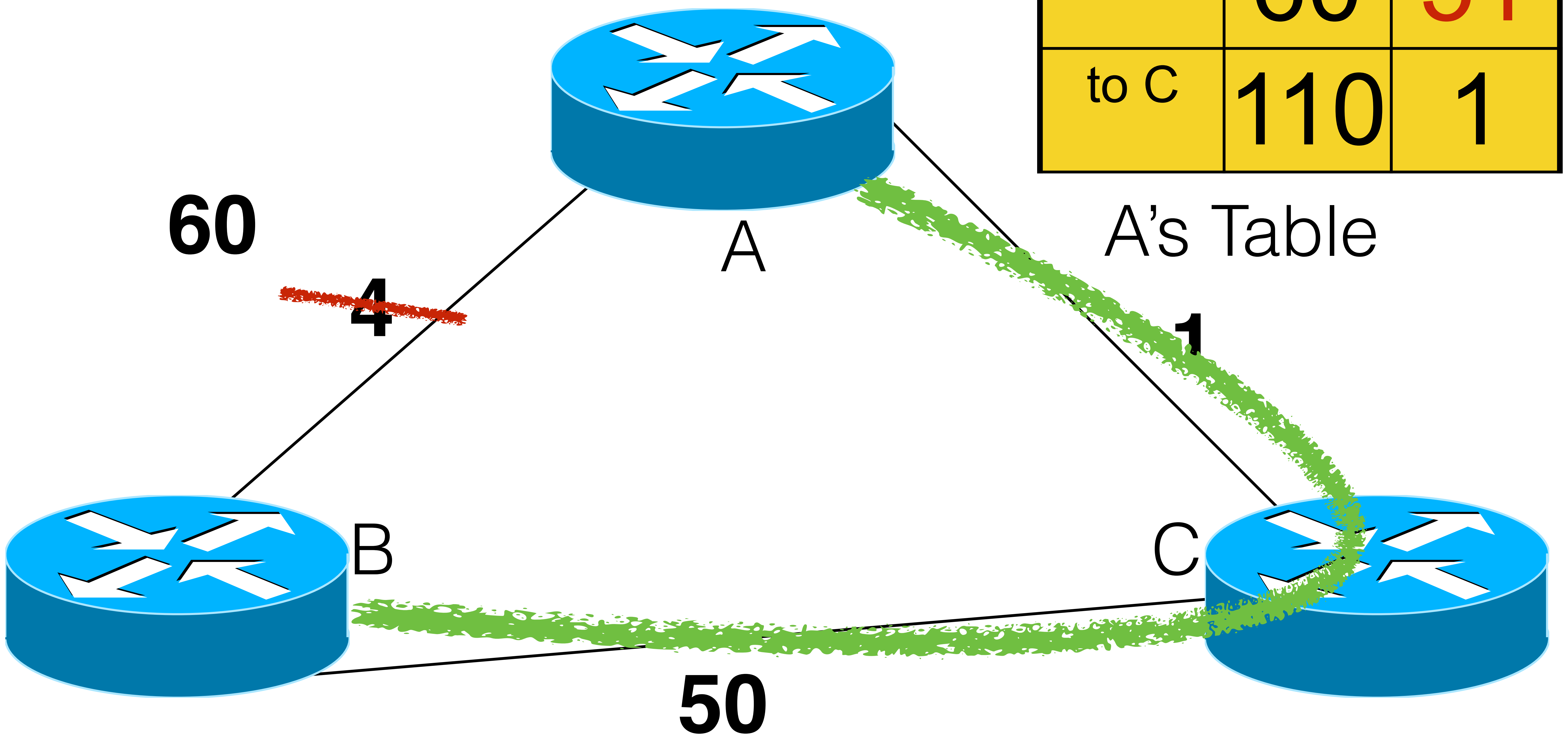
C’s Table

	via A	viaC
to A	4	51
to C	5	50

B’s Table

When will this slow-counting finally end?

Count until we equal the “real” shortest path.



	via B	viaC
to B	60	51
to C	110	1

A’s Table

	via A	via B
to A	1	101
to B	52	50

C’s Table

	via A	viaC
to A	60	51
to C	61	50

B’s Table

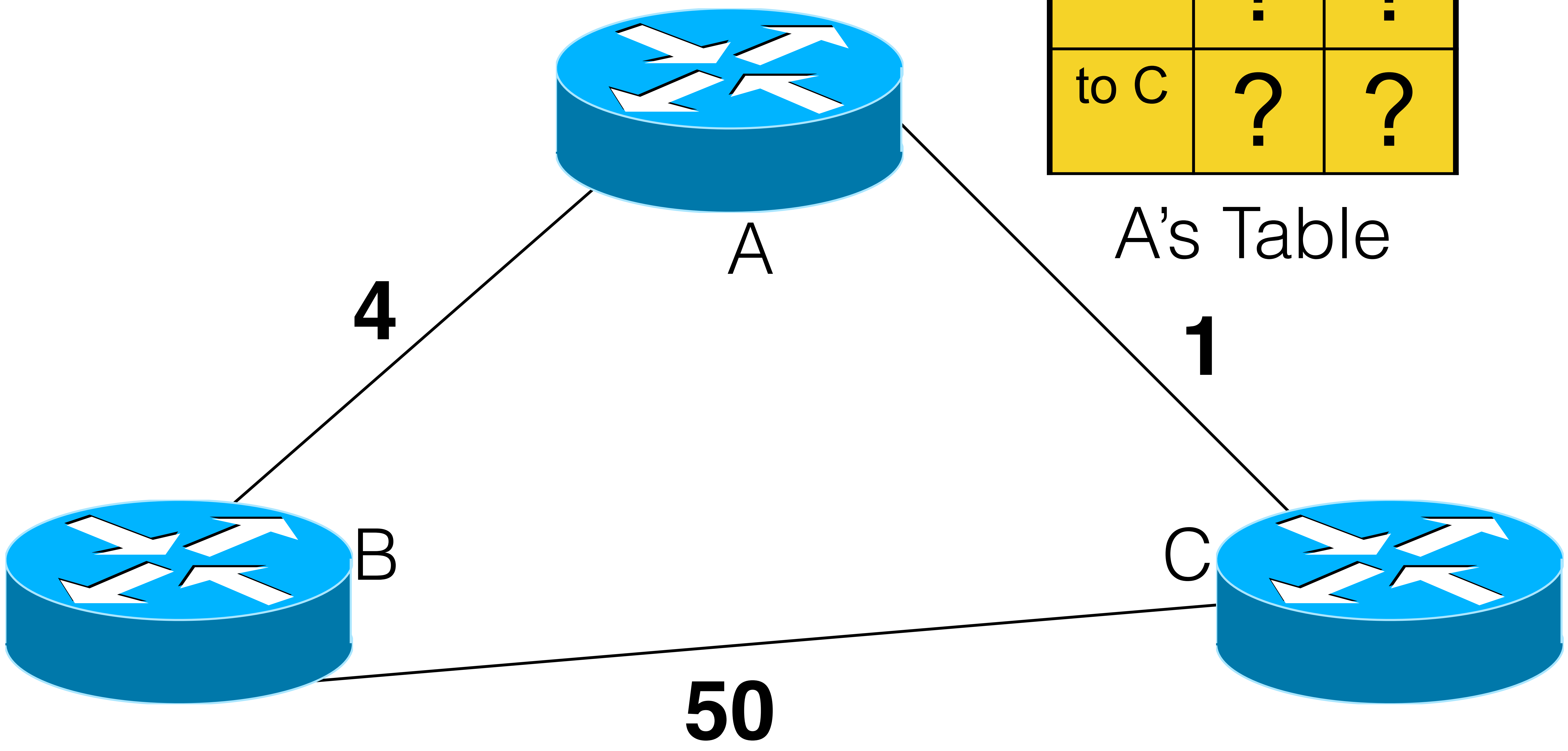
Three Techniques for Mitigating Count to Infinity

- Split Horizon/Poison Reverse
- Maximum Path Lengths
- Pushdown Timers

Three Techniques for Mitigating Count to Infinity

- Split Horizon/Poison Reverse
  - If I select a route I received from you in my distance vector, instead I will report a path length of INFINITY back to you.
- Maximum Path Lengths
- Pushdown Timers

Flashback: What Does That Look Like?



	via B	viaC
to B	?	?
to C	?	?

A's Table

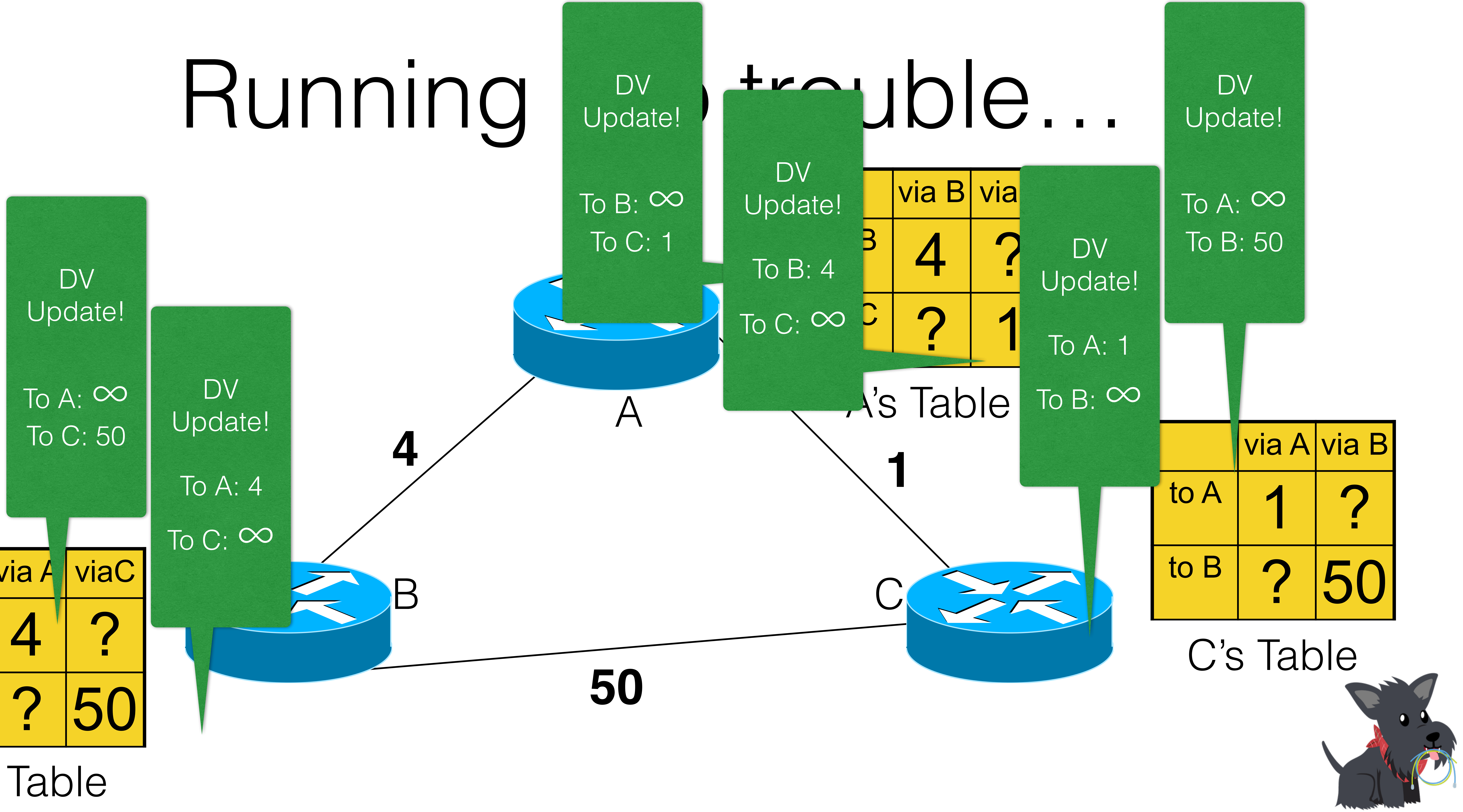
	via A	via B
to A	?	?
to B	?	?

C's Table

	via A	viaC
to A	?	?
to C	?	?

B's Table

Running … trouble…



	via B	via
B	4	?
C	?	1

A's Table

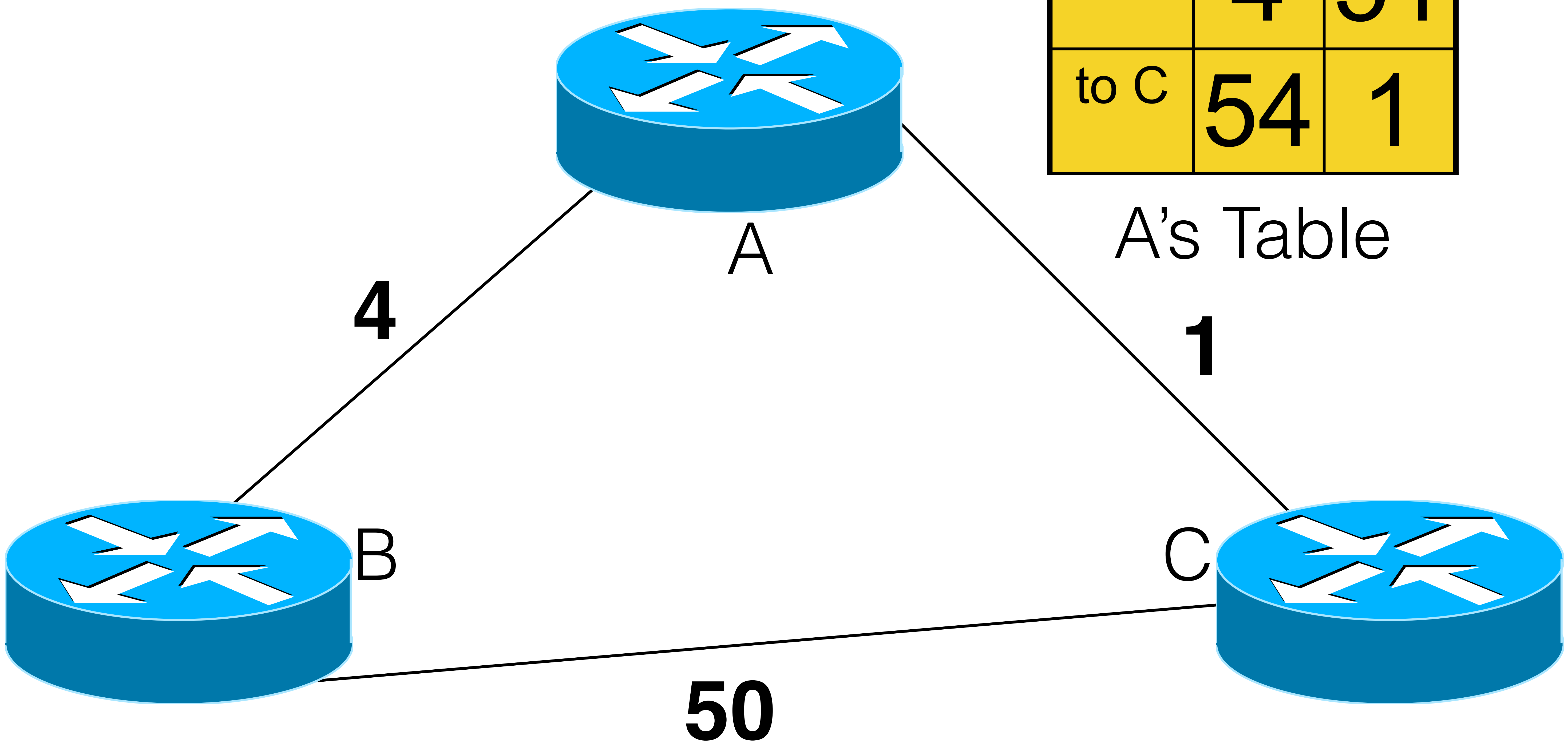
	via A	viaC
to A	4	?
to C	?	50

B's Table

	via A	via B
to A	1	?
to B	?	50

C's Table

Running into trouble…



	via B	viaC
to B	4	51
to C	54	1

A's Table

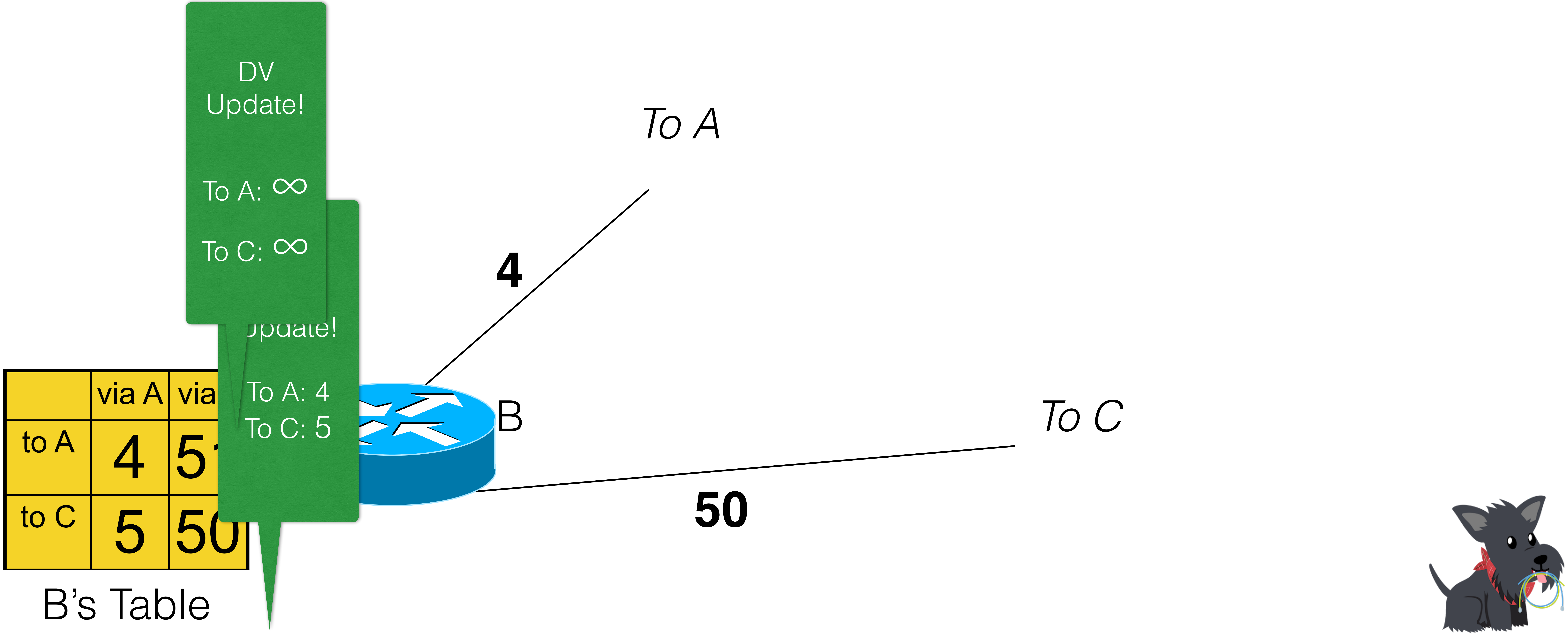
	via A	via B
to A	1	54
to B	5	50

C's Table

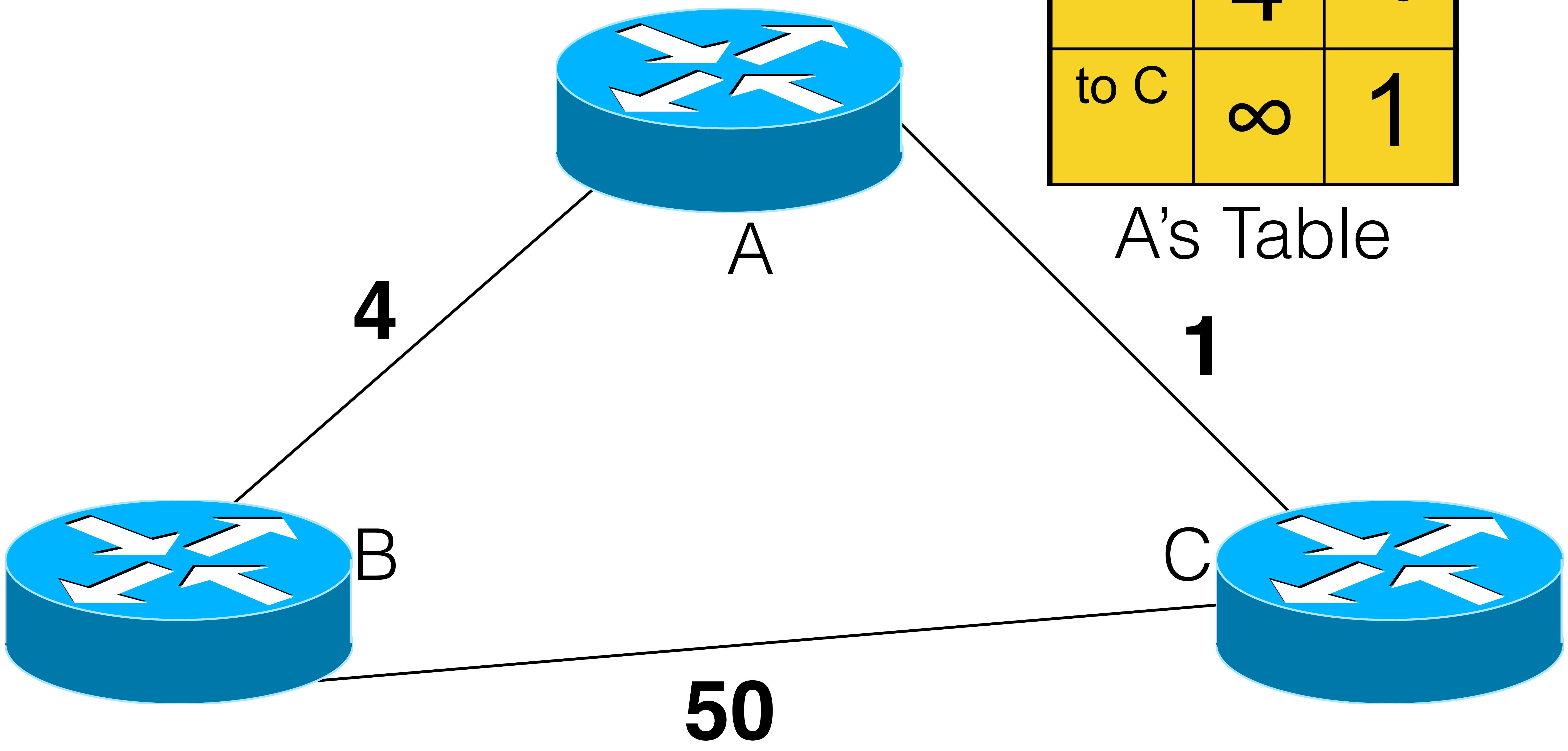
	via A	viaC
to A	4	51
to C	5	50

B's Table

Zoom In


DV
Update!
To A: ∞
To C: ∞
To A: 4
To C: 5
To A
To C
4
50
B
via A
via
to A
to C
4
5
50
B's Table

No Bad Loopy Routes!



	via B	viaC
to B	4	∞
to C	∞	1

A’s Table

	via A	via B
to A	1	54
to B	5	50

C’s Table

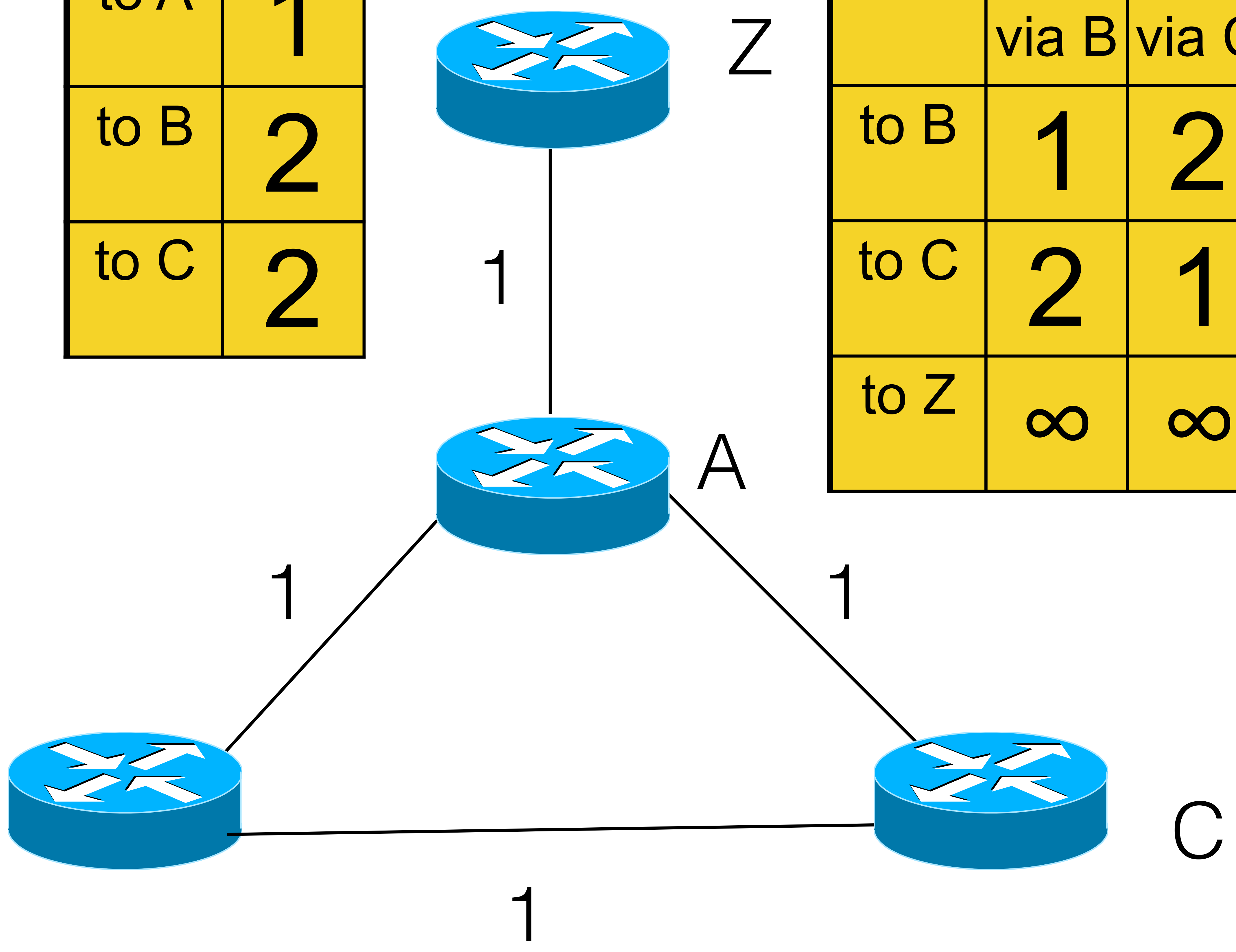
	via A	viaC
to A	4	51
to C	5	50

B’s Table

…for that graph.

A completely sad graph

	via A
to A	1
to B	2
to C	2

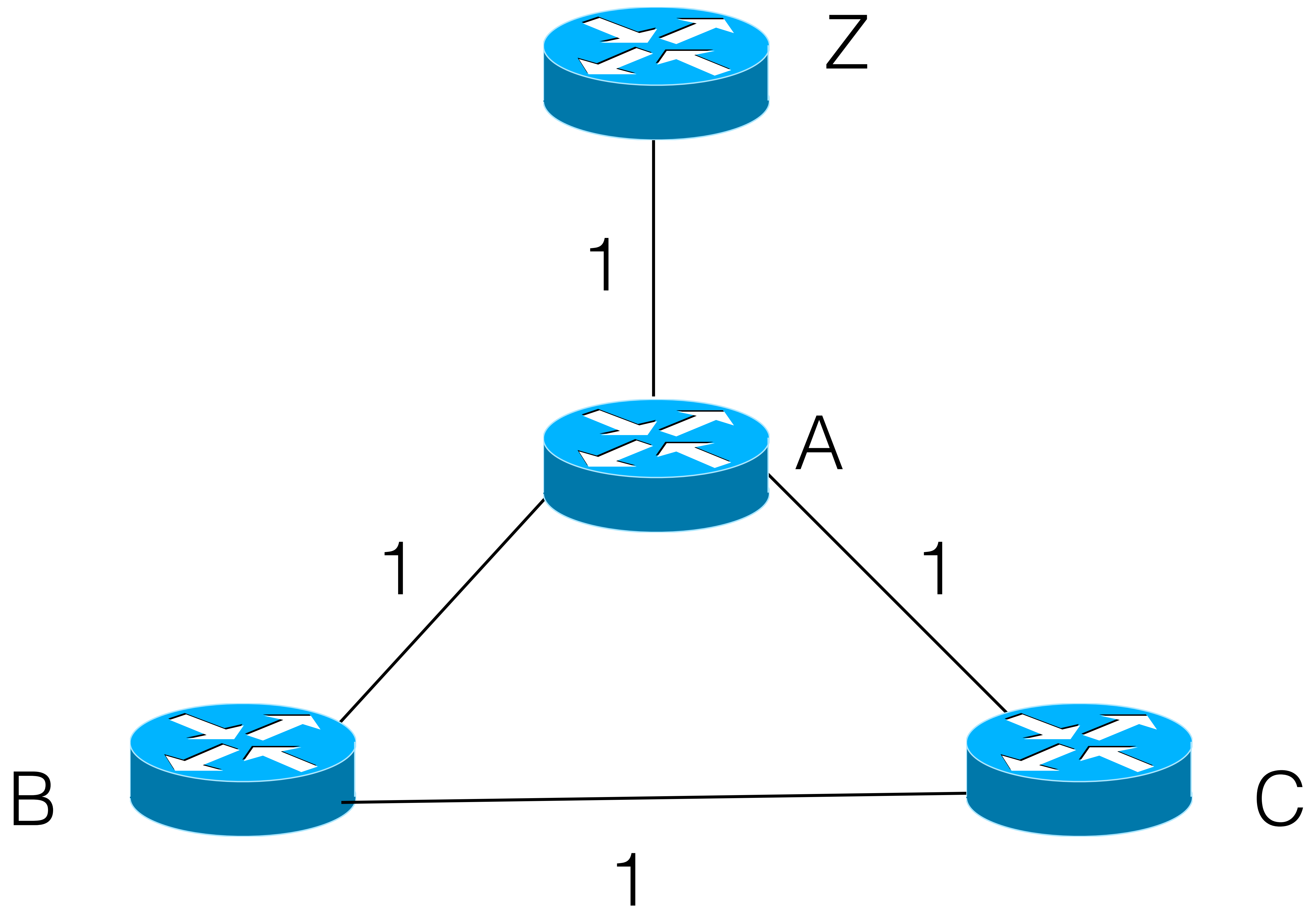


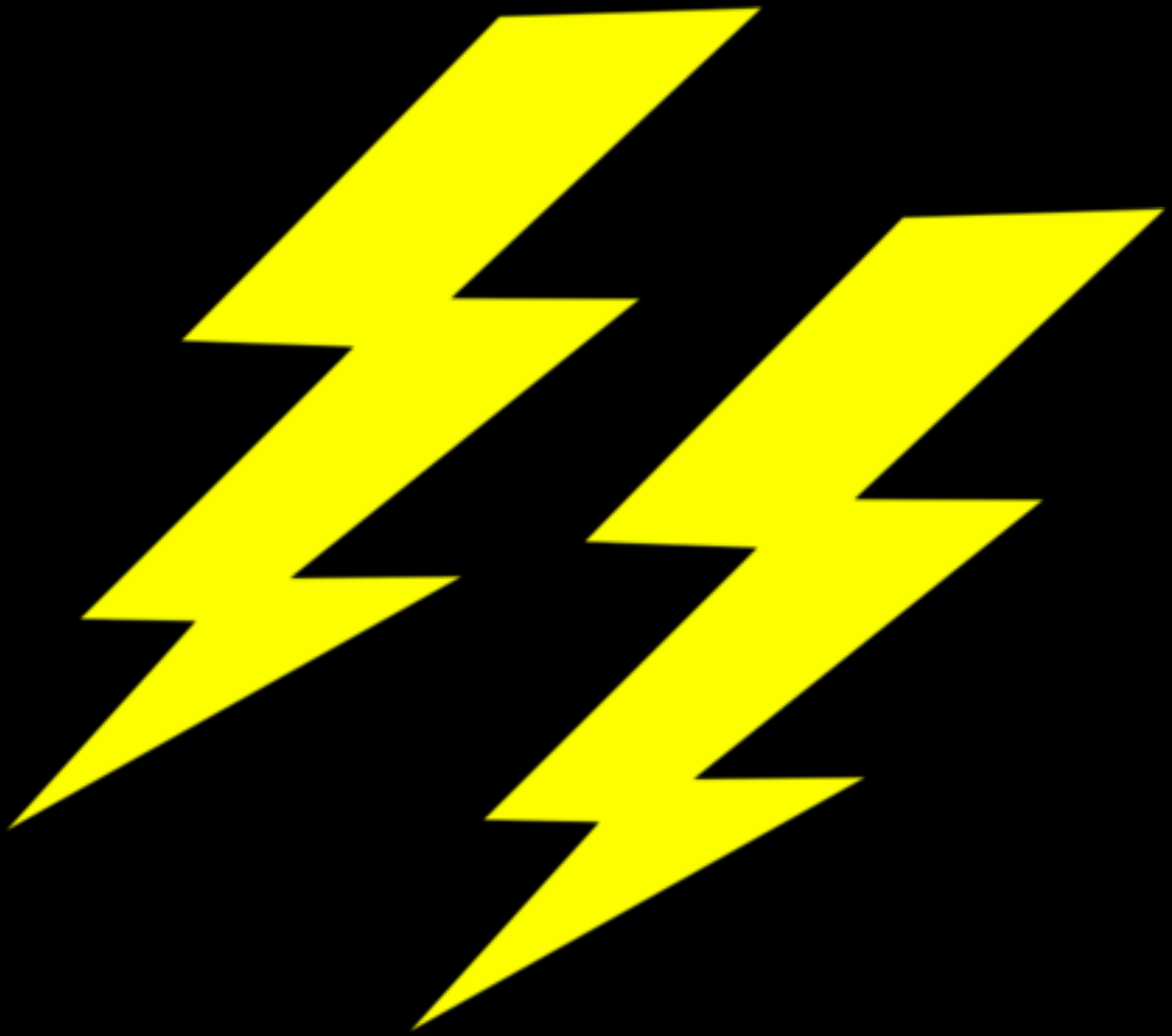
	via B	via C	via Z
to B	1	2	∞
to C	2	1	∞
to Z	∞	∞	1

	via A	viaC
to A	1	2
to C	2	1
to Z	2	3

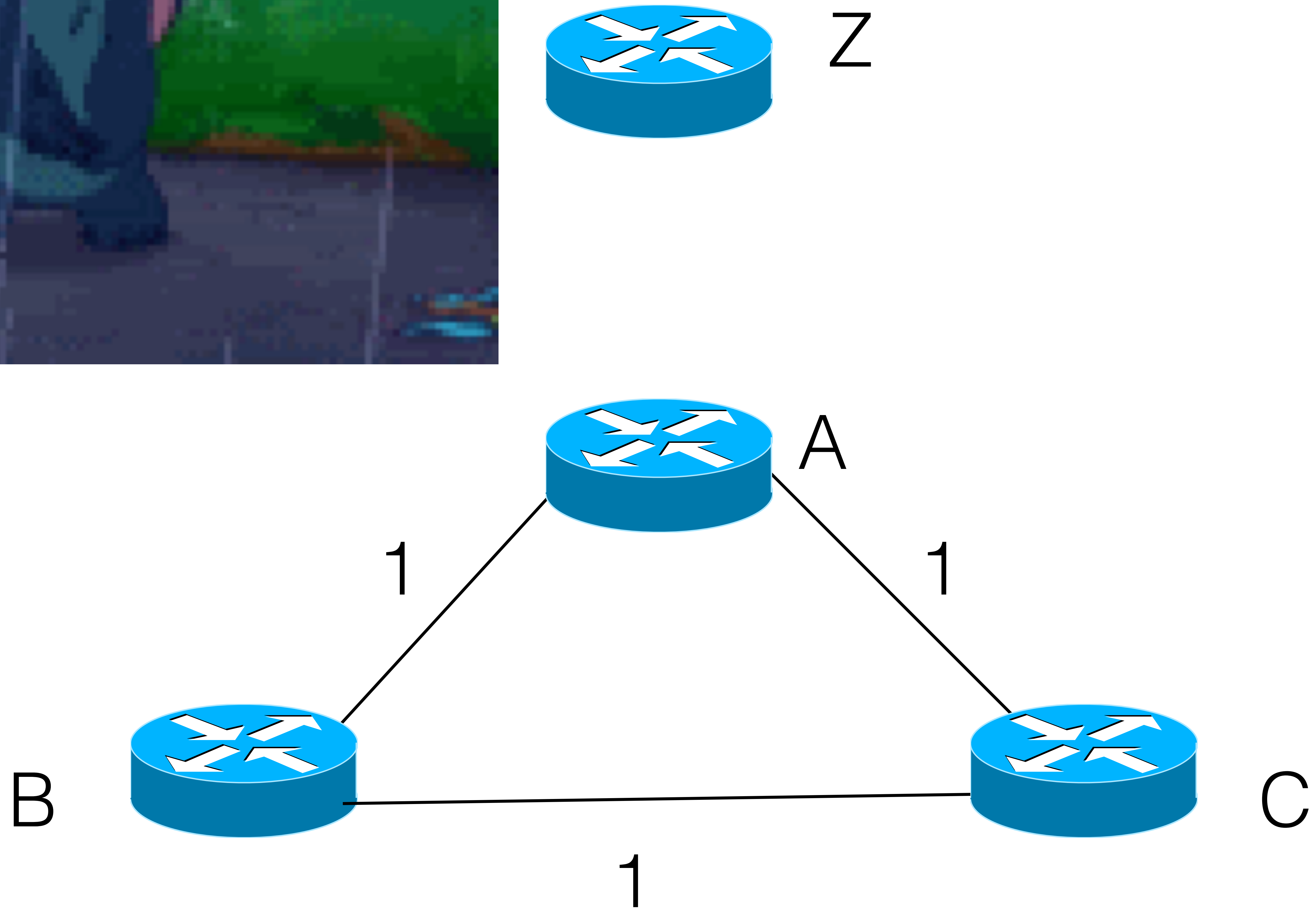
	via A	via B
to A	1	2
to B	2	1
to Z	2	3

A completely sad graph





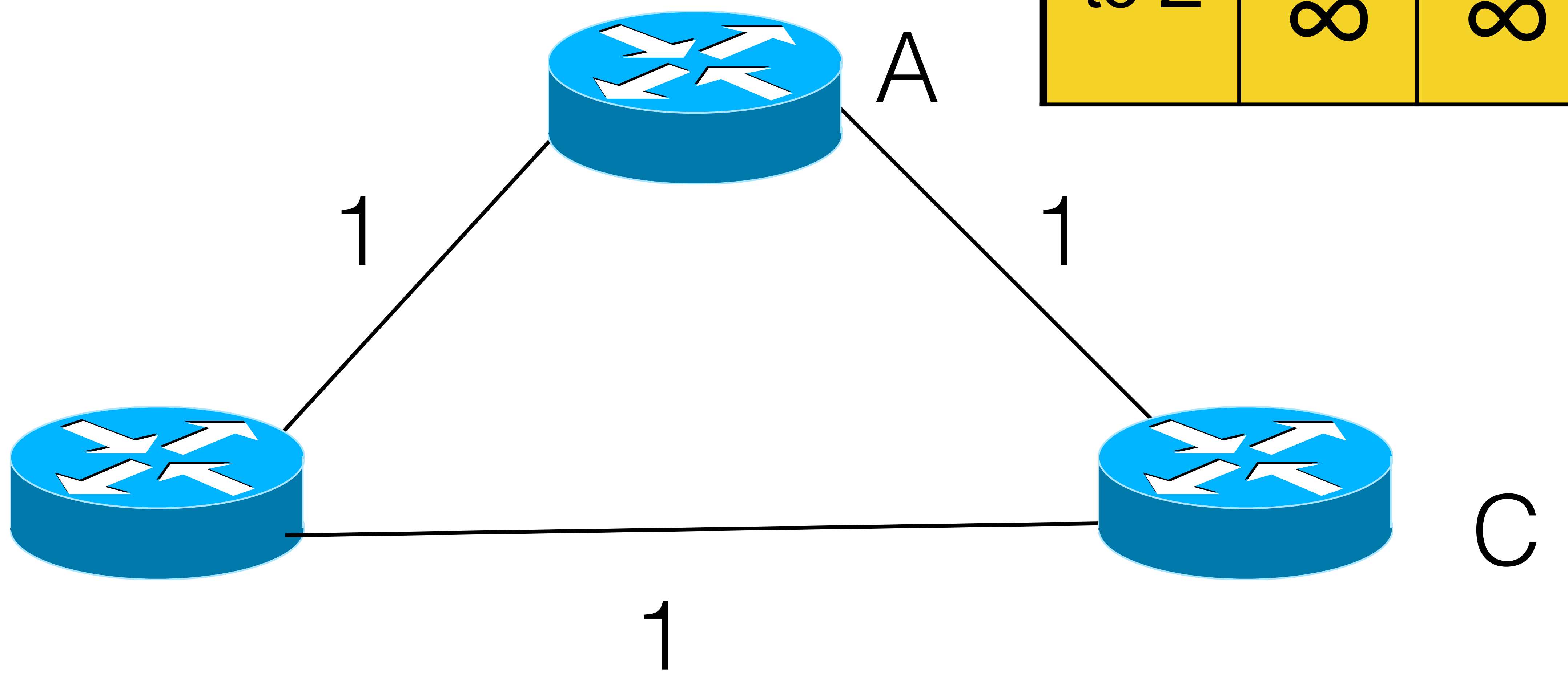
letely sad graph



A completely sad graph

	via B	via C	via Z
to B	1	2	∞
to C	2	1	∞
to Z	∞	∞	∞

	via A	viaC
to A	1	2
to C	2	1
to Z	2	3



	via A	via B
to A	1	2
to B	2	1
to Z	2	3

A comple y sad graph

DV Update!

To A: 1
To C: 1
To Z: 2

	via A	viaC
to A	1	2
to C	2	1
to Z	2	3

DV Update!

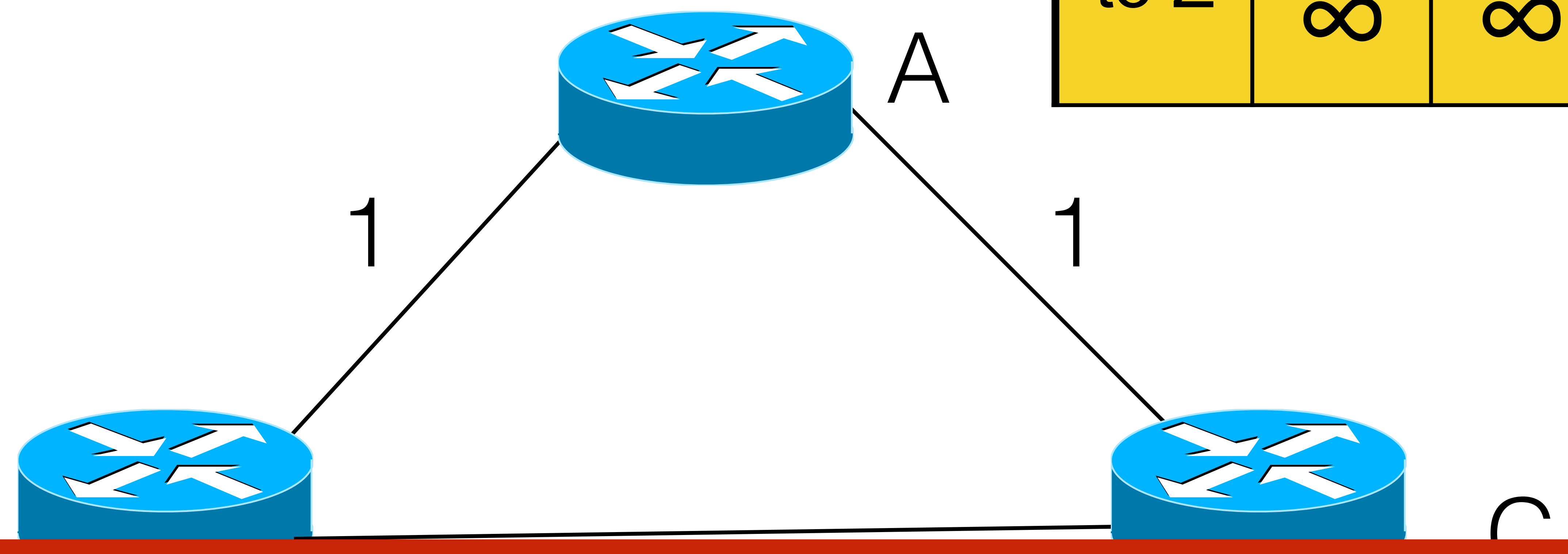
To B: 1
To C: 1
To Z: ∞

	via B	via C	via Z
to B	1	2	∞
to C	2	1	∞
to Z	∞	∞	∞

DV Update!

To A: 1
To B: 1
To Z: 2

	via A	via B
to A	1	2
to B	2	1
to Z	2	3

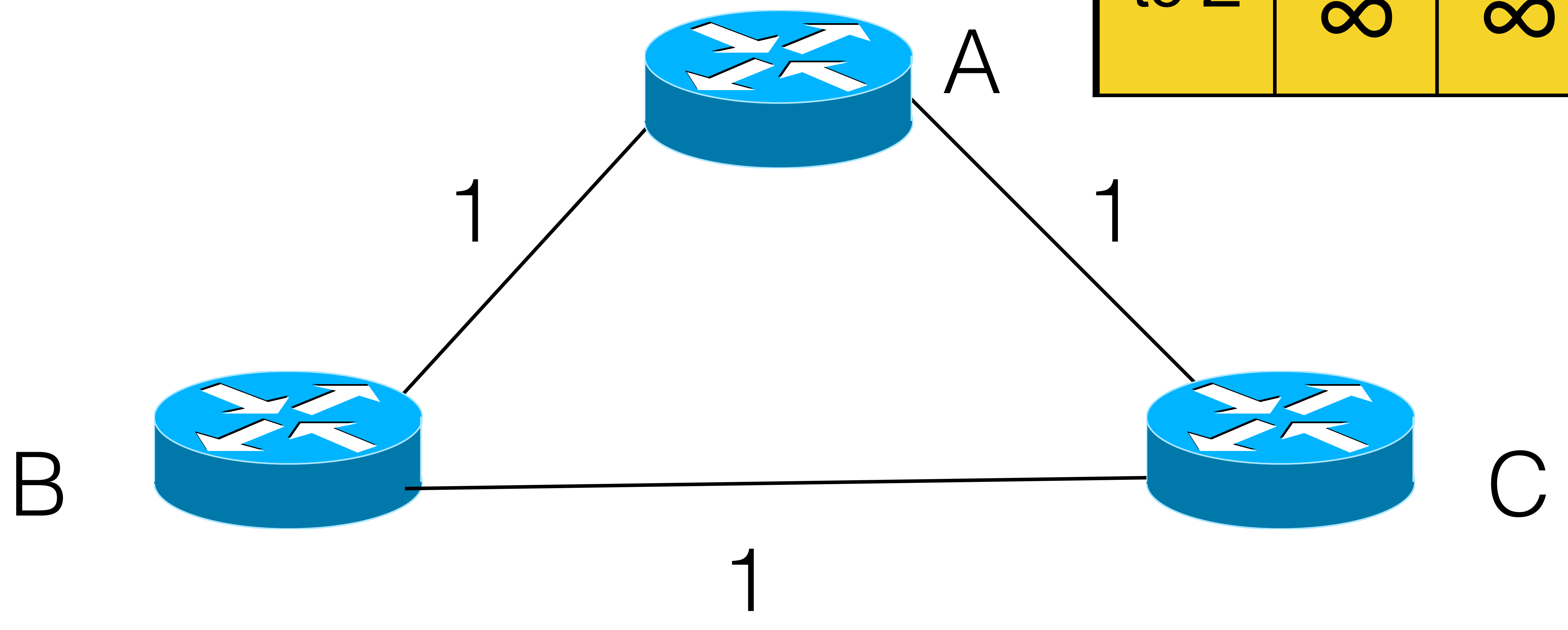


B and C won't advertise the route for Z to A
But they will advertise it to each other…

A completely sad graph

	via B	via C	via Z
to B	1	2	∞
to C	2	1	∞
to Z	∞	∞	∞

	via A	viaC
to A	1	2
to C	2	1
to Z	∞	3



	via A	via B
to A	1	2
to B	2	1
to Z	∞	3

A comple y sad graph

DV
Update!

To B: 1
To C: 1

To Z: ∞

	via B	via C	via Z
to B	1	2	∞
to C	2	1	∞
to Z	∞	∞	∞

DV
Update!

To A: 1
To C: 1
To Z: 3

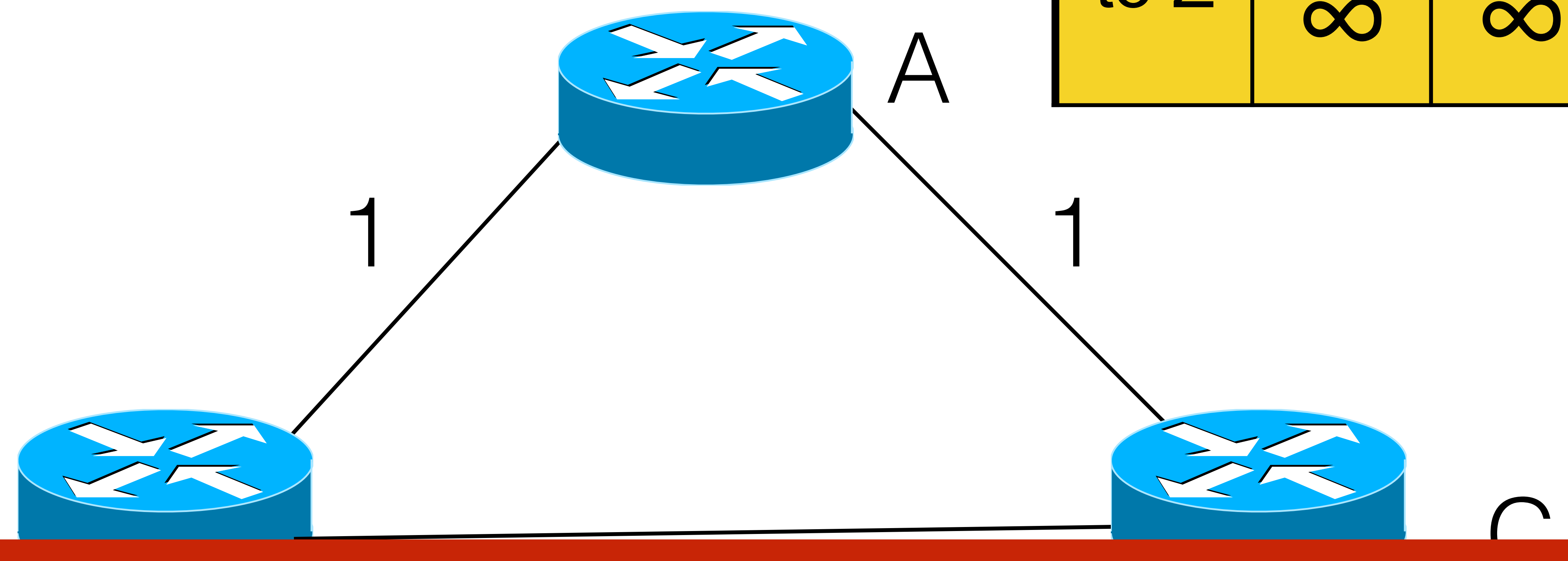
	via A	viaC
to A	1	2
to C	2	1
to Z	∞	3

DV
Update!

To A: 1
To B: 1

To Z: 3

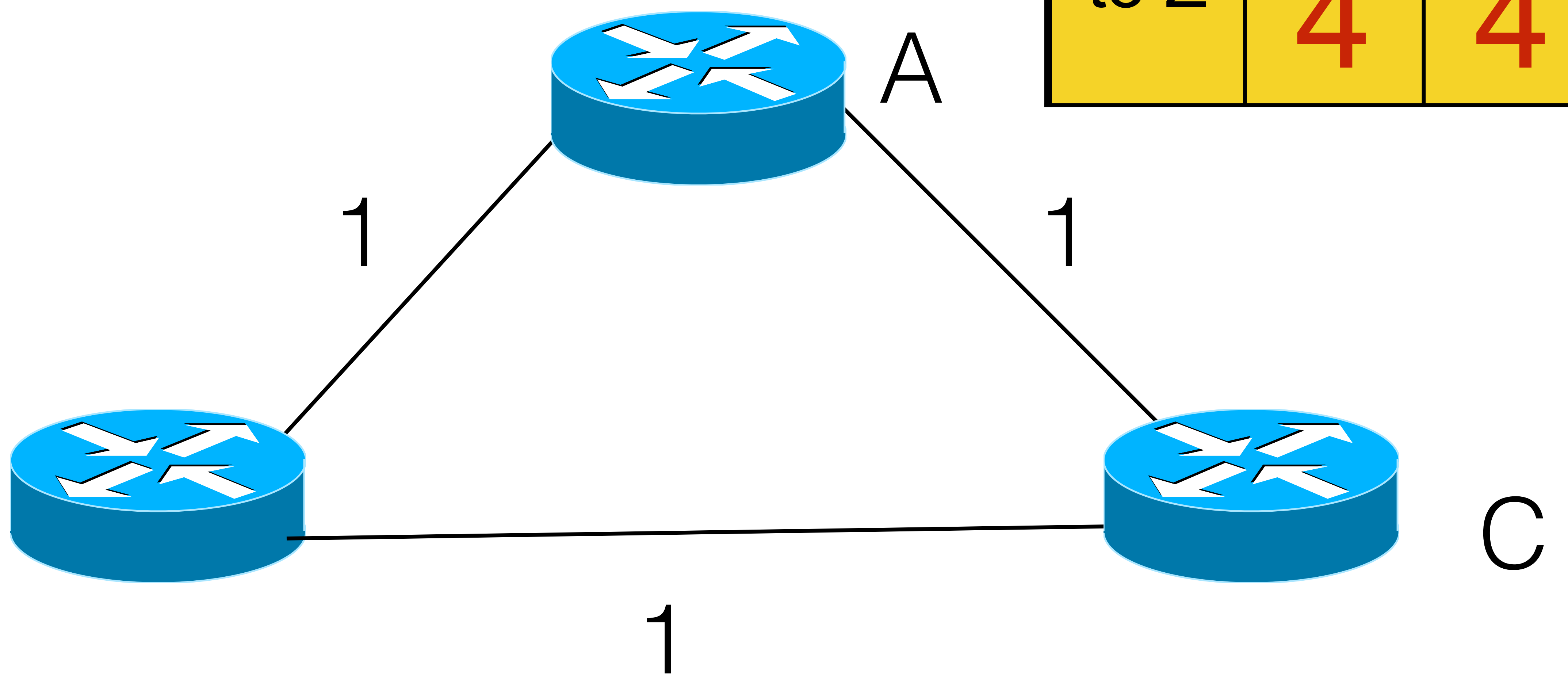
	via A	via B
to A	1	2
to B	2	1
to Z	∞	3



And they will advertise the routes they learned from each other back to A.

A completely sad graph

	via B	via C	via Z
to B	1	2	∞
to C	2	1	∞
to Z	4	4	∞



	via A	viaC
to A	1	2
to C	2	1
to Z	∞	3

	via A	via B
to A	1	2
to B	2	1
to Z	∞	3

Here we go again…

In this case, we will count how high?

Three Techniques for Mitigating Count to Infinity

- Split Horizon/Poison Reverse
  - If I select a route I received from you in my distance vector, instead I will report a path length of INFINITY back to you.
- Maximum Path Lengths
  - Need to stop counting forever — set a path length limit to stop things from counting forever.
- Pushdown Timers

Three Techniques for Mitigating Count to Infinity

- Split Horizon/Poison Reverse
  - If I select a route I received from you in r… path length of INFINITY back to you.
- Maximum Path Lengths
  - Need to stop counting forever — set a path length limit to stop things from counting

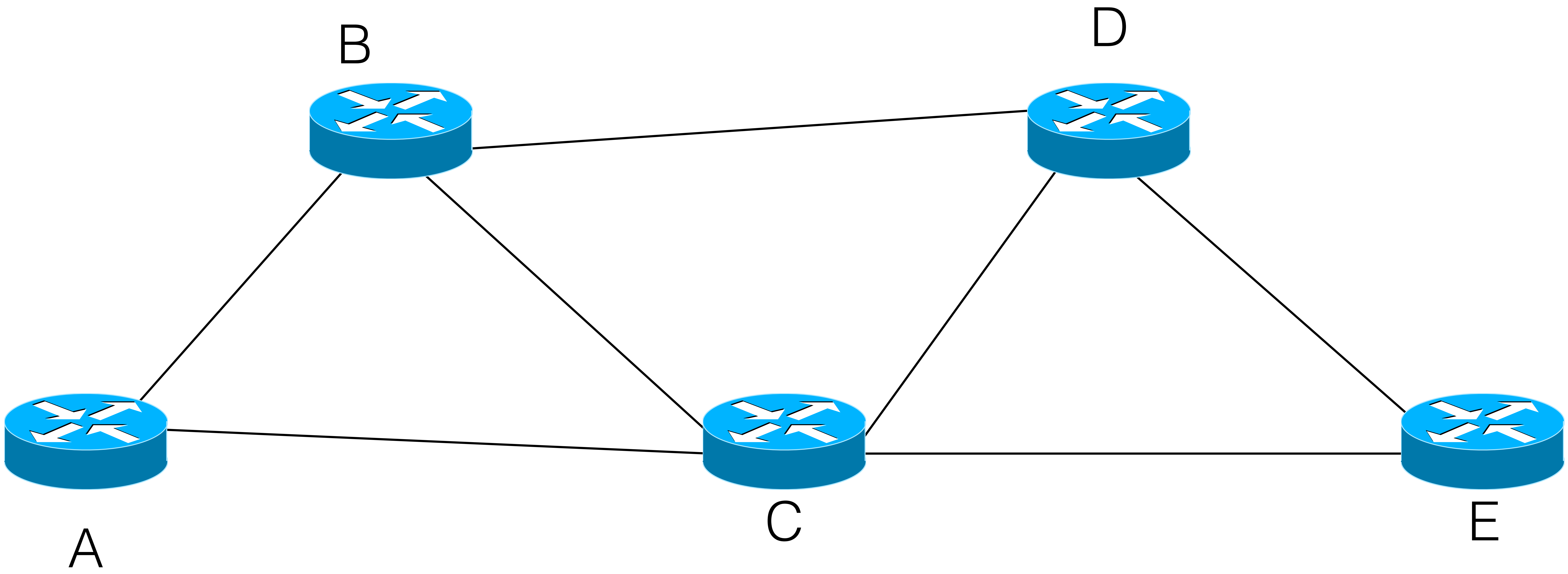
Sorry not sorry

I'm here to exercise your brain and make you mentally BUFF.

- Pushdown Timers
  - I'm not going to talk about these in lecture, instead I'm going to give you a tricky question about them, either on the homework or the midterm.

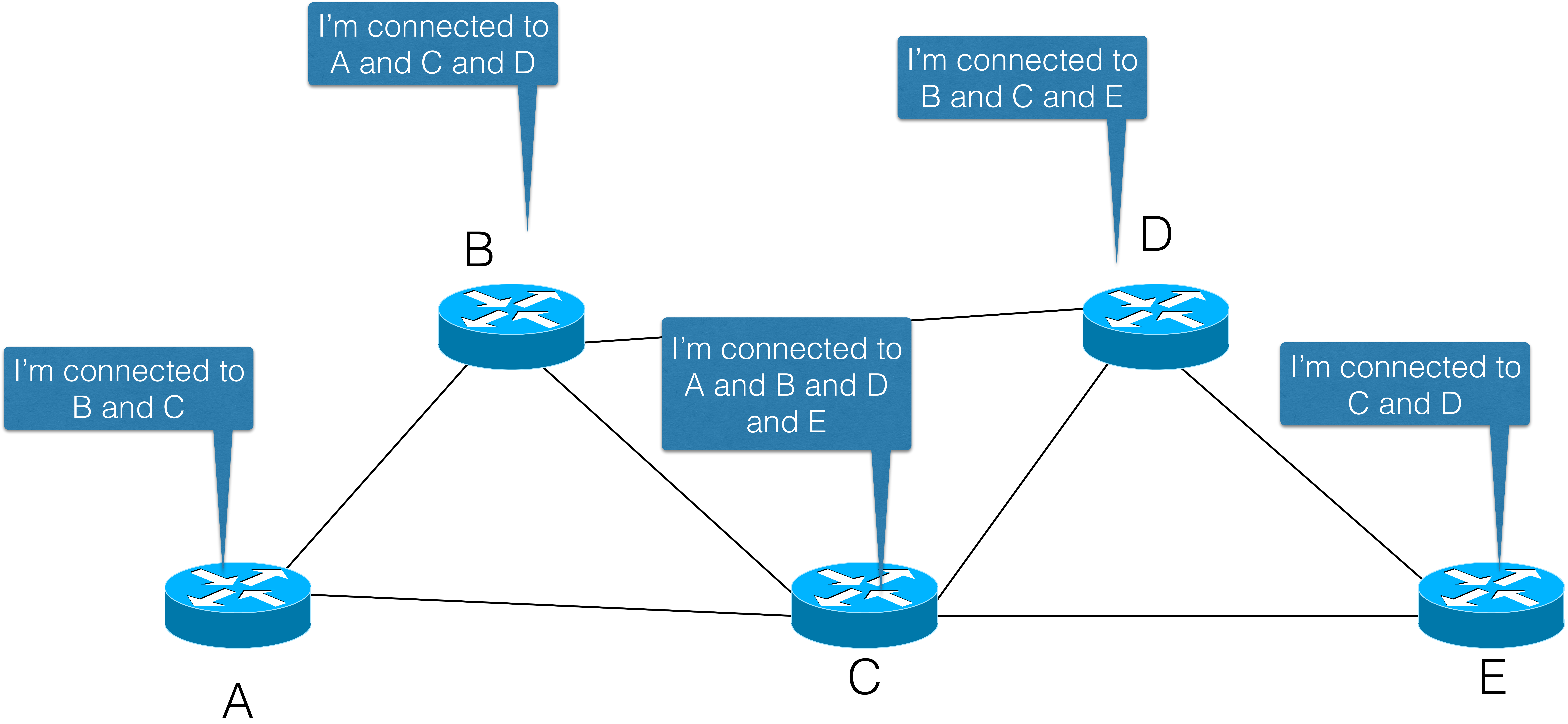
	Broadcast Network w/ Learning Switches	Broadcast Network w/ Learning Switches and Spanning Tree	Distance Vector e.g RIP
Resilience	If there is a route, the packet will reach dest!	Need to recompute spanning tree if failure	Very slow recovery due to count to infinity.
Fully Distributed	Yes	Yes	Yes
State per Node	Learning Switch: O(#nodes)	Learning Switch: O(#nodes) + Path to Root: O(constant)	O(# switches * max node degree) O(#nodes)
Convergence	No setup time at all!	Need to run spanning tree protocol before routing	Need to run DV before routing — takes length of longest best path time.
Routing Efficiency	Broadcast Storms	Still sends new connections everywhere.	Packets sent directly to their destination.
Shortest Path?	Not Necessarily…	Not Necessarily…	Yes

Our Newest Friend:
Link State Algorithm
(e.g. OSPF)

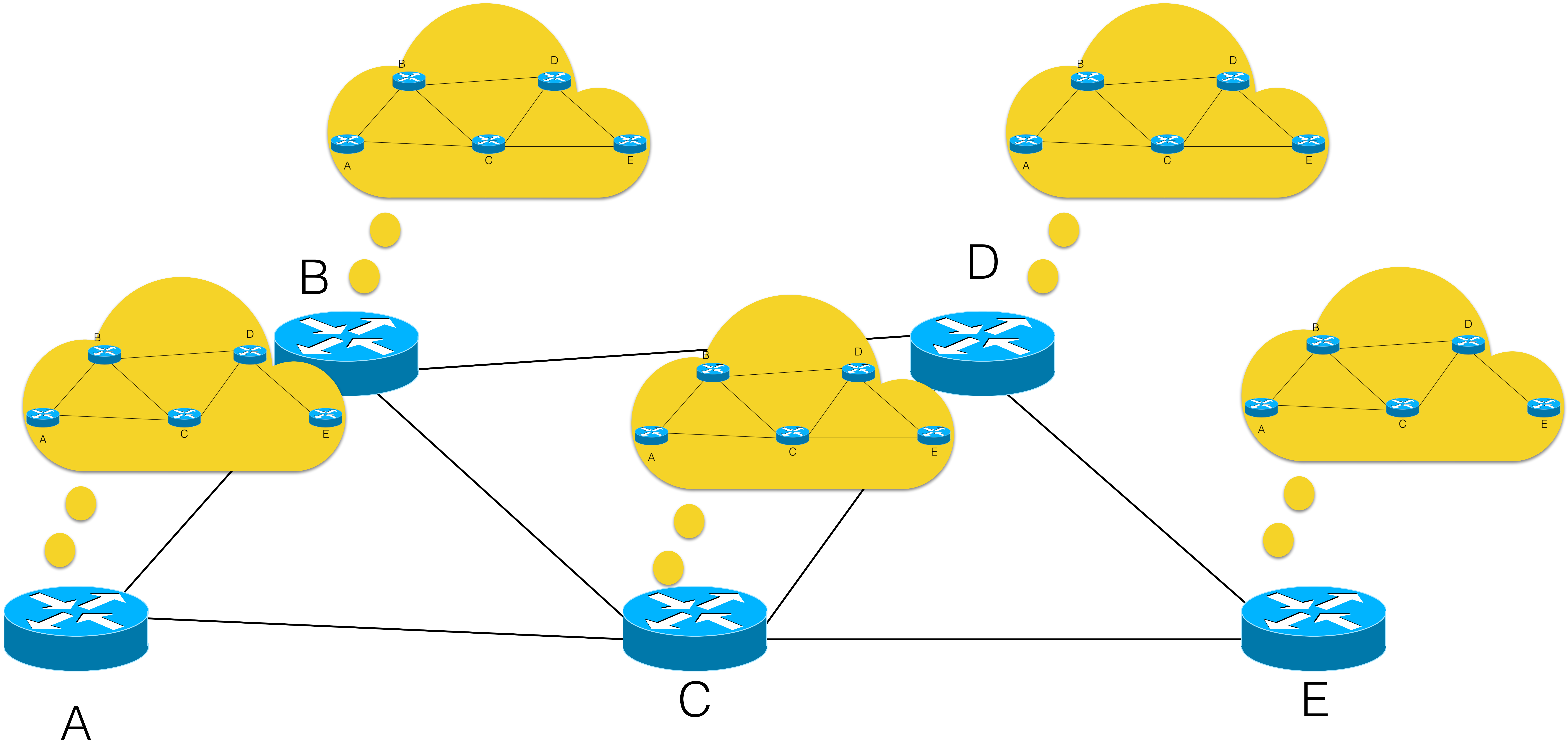

B
D
A
C
E

Link State, In a Nutshell

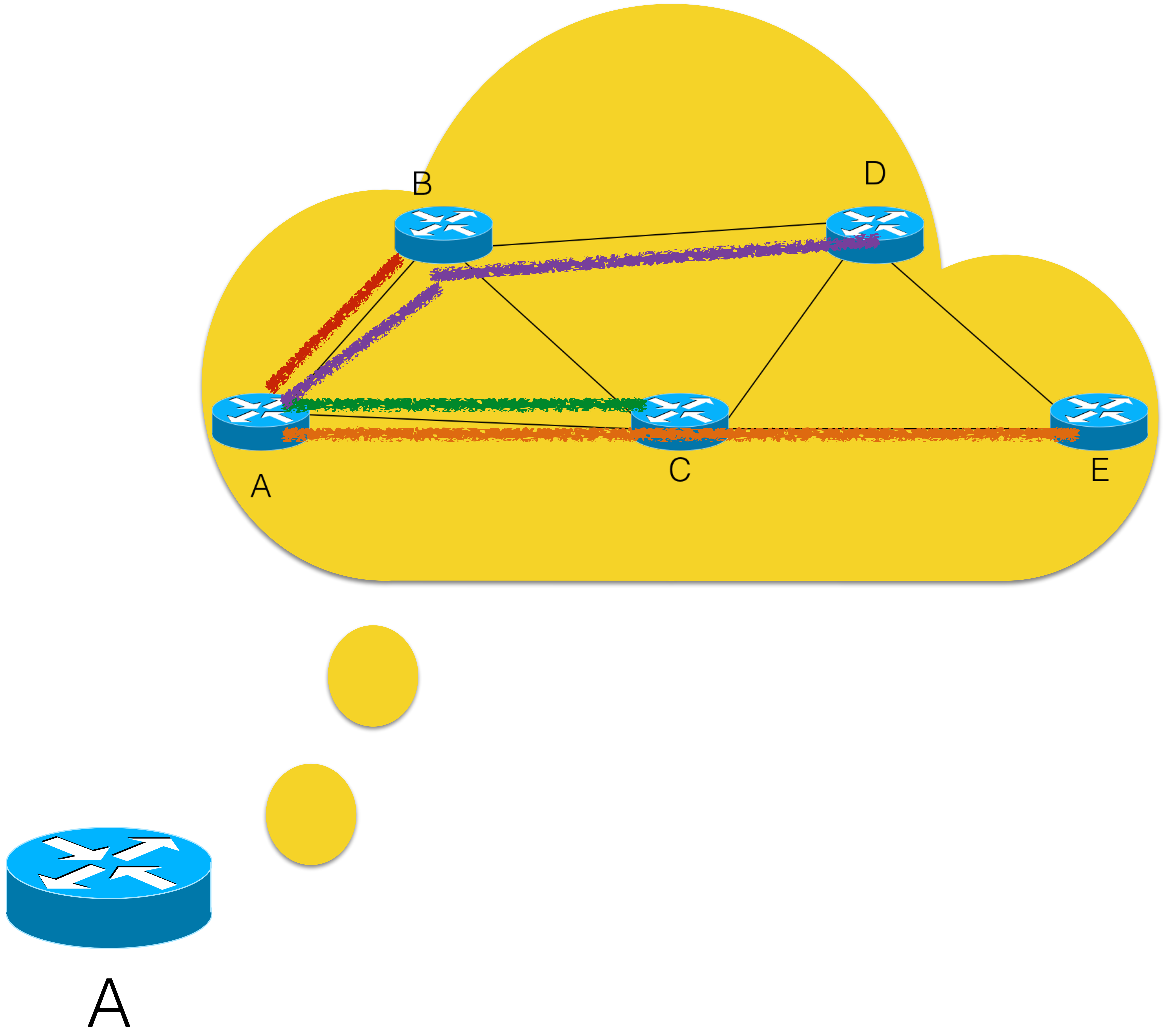
- Everyone knows who they are connected to directly.
- Every node *broadcasts* a list of who they are connected to (and with what link weight) to every other node in the network.
- Every node then can — locally — figure out what the entire network graph is.
- Each node then uses a shortest-path algorithm to find its shortest path to every possible destination.
- Each node then builds its own routing table.



I think animating all of these messages being broadcast will give me carpal tunnel syndrome. Imagine they are being broadcast.



Each node, through receiving these broadcast messages, can then figure out the entire network structure.



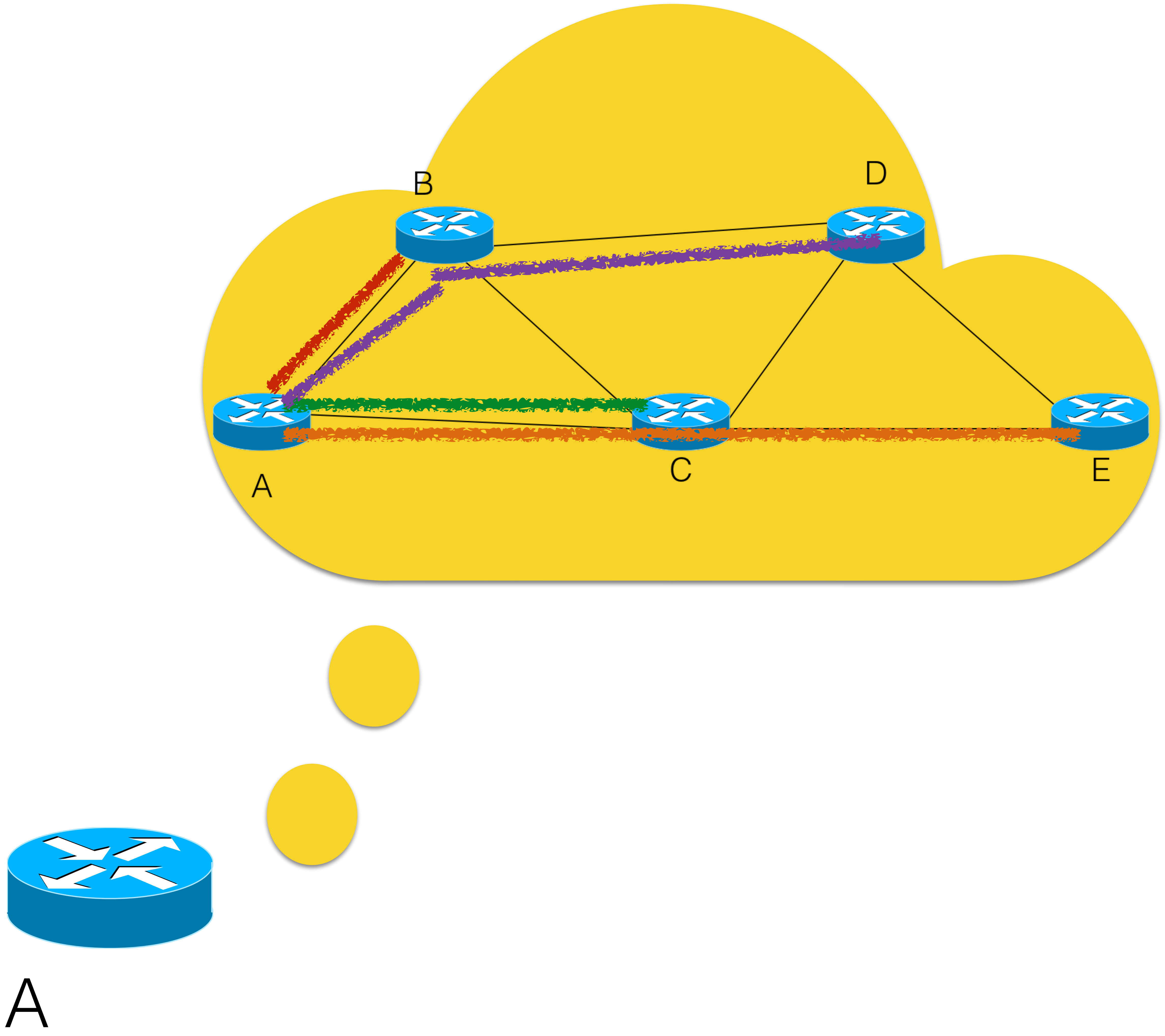
Once a router knows the entire graph structure, it can easily compute its shortest path to every other node in the network.

How do we compute A's shortest path to all other nodes?

Dijkstra's Algorithm

Edsgar Dijkstra
Turing Award Winner
(11 May 1930 – 6 August 2002)

- Finds the shortest path from a source to all other destinations
- Runs in O(|E| + |V| log |V|) time
  - for E edges and V vertices
- You should have learned this 15-210, 15-251, or literally a billion other courses.
- *If you haven't learned it yet, come to OH and I'll teach it to you.*
  - *It's also on Wikipedia.*
  - *It's very cool — greedy algo!*



To B? Forward to B

To C? Forward to C

To D? Forward to B

To E? Forward to C

Once A knows its shortest path to every destination, it creates a routing table — what is the next hop for each destination?

That’s like… it.

	Broadcast Network w/ Learning Switches	Broadcast Network w/ Learning Switches and Spanning Tree	Distance Vector e.g RIP
Resilience	If there is a route, the packet will reach dest!	Need to recompute spanning tree if failure	Very slow recovery due to count to infinity.
Fully Distributed	Yes	Yes	Yes
State per Node	Learning Switch: O(#nodes)	Learning Switch: O(#nodes) + Path to Root: O(constant)	O(# switches * max node degree) + O(#nodes)
Convergence	No setup time at all!	Need to run spanning tree protocol before routing	Need to run DV before routing — takes length of longest best path time.
Routing Efficiency	Broadcast Storms	Still sends new connections everywhere.	Packets sent directly to their destination.
Shortest Path?	Not Necessarily…	Not Necessarily…	Yes

	Distance Vector e.g RIP
Resilience	Very slow recovery due to count to infinity.
Fully Distributed	Yes
State per Node (+ Routing Table)	O(# switches * max node degree)
Convergence	Need to run DV before routing — takes length of longest best path time.
Routing Efficiency	Packets sent directly to their destination.
Shortest Path?	Yes

	Distance Vector e.g RIP	Link State e.g OSPF
Resilience	Very slow recovery due to count to infinity.	
Fully Distributed	Yes	
State per Node (+ Routing Table)	O(# switches * max node degree)	
Convergence	Need to run DV before routing — takes length of longest best path time.	
Routing Efficiency	Packets sent directly to their destination.	Packets sent directly to their destination.
Shortest Path?	Yes	Yes

Okay, I did the first two for you — fill in the rest of this table w/ your neighbor.

	Distance Vector e.g RIP	Link State e.g OSPF
Resilience	Very slow recovery due to count to infinity.	
Fully Distributed	Yes	
State per Node (+ Routing Table)	O(# switches * max node degree)	
Convergence	Need to run DV before routing — takes length of longest best path time.	Flood network w/ updates and then run Dijkstra: O(|E| + |V| log |V|)
Routing Efficiency	Packets sent directly to their destination.	Packets sent directly to their destination.
Shortest Path?	Yes	Yes

	Distance Vector e.g RIP	Link State e.g OSPF
Resilience	Very slow recovery due to count to infinity.	
Fully Distributed	Yes	
State per Node (+ Routing Table)	O(# switches * max node degree)	O(# edges)
Convergence	Need to run DV before routing — takes length of longest best path time.	Flood network w/ updates and then run Dijkstra: $O(\|E\| + \|V\| \log \|V\|)$
Routing Efficiency	Packets sent directly to their destination.	Packets sent directly to their destination.
Shortest Path?	Yes	Yes

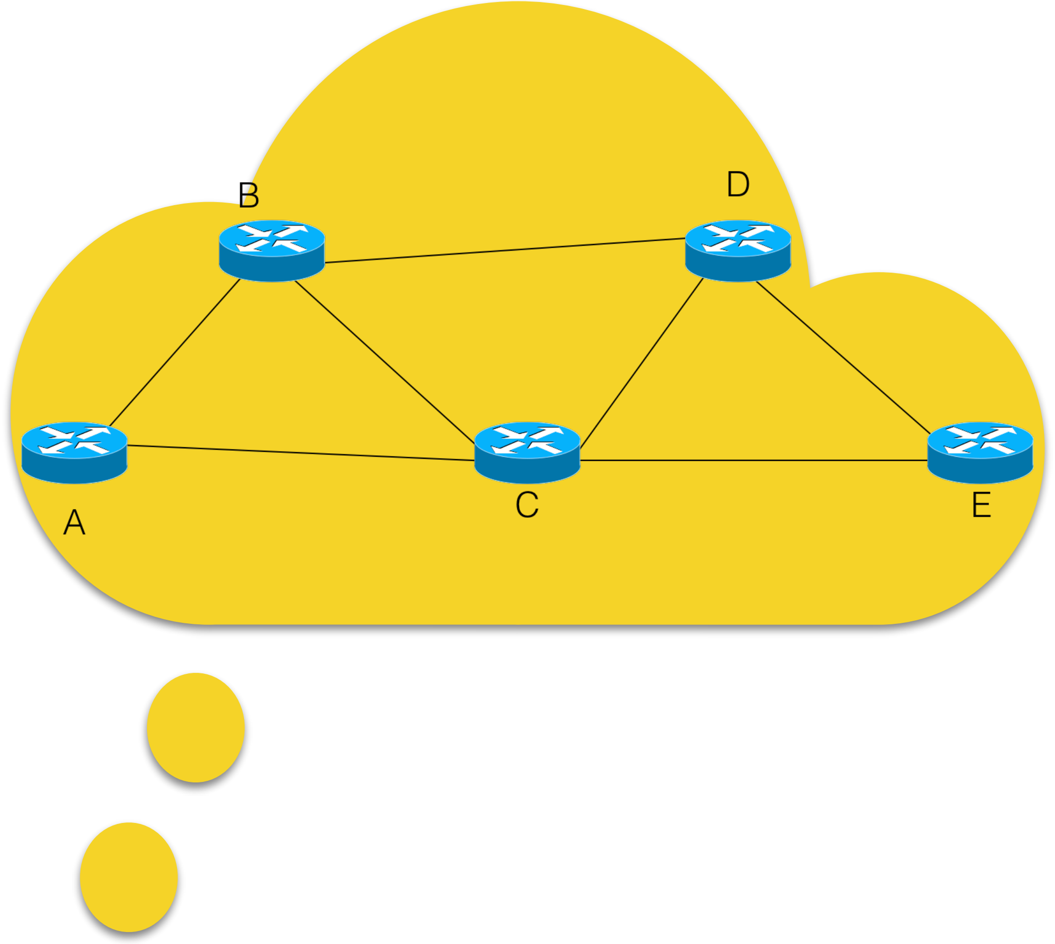
	Distance Vector e.g RIP	Link State e.g OSPF
Resilience	Very slow recovery due to count to infinity.	
Fully Distributed	Yes	Yes
State per Node (+ Routing Table)	O(# switches * max node degree)	O(# edges)
Convergence	Need to run DV before routing — takes length of longest best path time.	Flood network w/ updates and then run Dijkstra: O(|E| + |V| log |V|)
Routing Efficiency	Packets sent directly to their destination.	Packets sent directly to their destination.
Shortest Path?	Yes	Yes

	Distance Vector e.g RIP	Link State e.g OSPF
Resilience	Very slow recovery due to count to infinity.	Re-Run Dijkstra and you’re good to go.
Fully Distributed	Yes	Yes
State per Node (+ Routing Table)	O(# switches * max node degree)	O(# edges)
Convergence	Need to run DV before routing — takes length of longest best path time.	Flood network w/ updates and then run Dijkstra: O(|E| + |V| log |V|)
Routing Efficiency	Packets sent directly to their destination.	Packets sent directly to their destination.
Shortest Path?	Yes	Yes

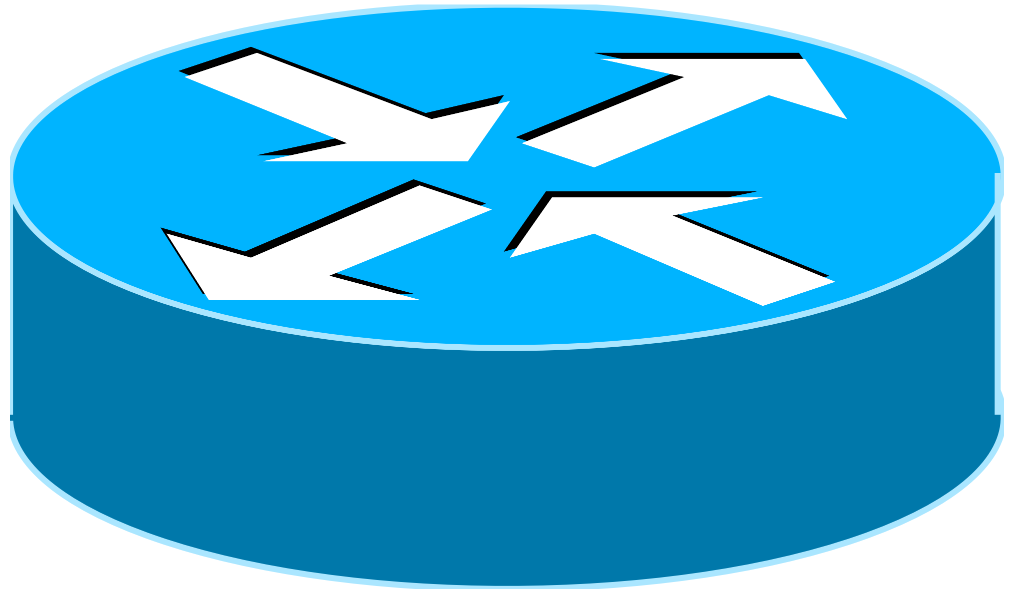
One more point of comparison: Switch Control Plane Complexity

A switch has two components

- “Data Plane”
  - When a packet comes in, the data plane reads from a routing table and decides where to send the packet. Then it sends the packet out the correct port.
- “Control Plane”
  - This is the “brains” of the switch — the part that decides what to put into the routing table.



Send and receive updates
Decide what my new routes should be
WRITE to routing table



Control Plane

Data Plane

ROUTING TABLE
To B? Forward to B

To C? Forward to C

To D? Forward to B

To E? Forward to C

Receive packets.
READ from routing table to learn where I am supposed to send them
Send packet out of the correct port.

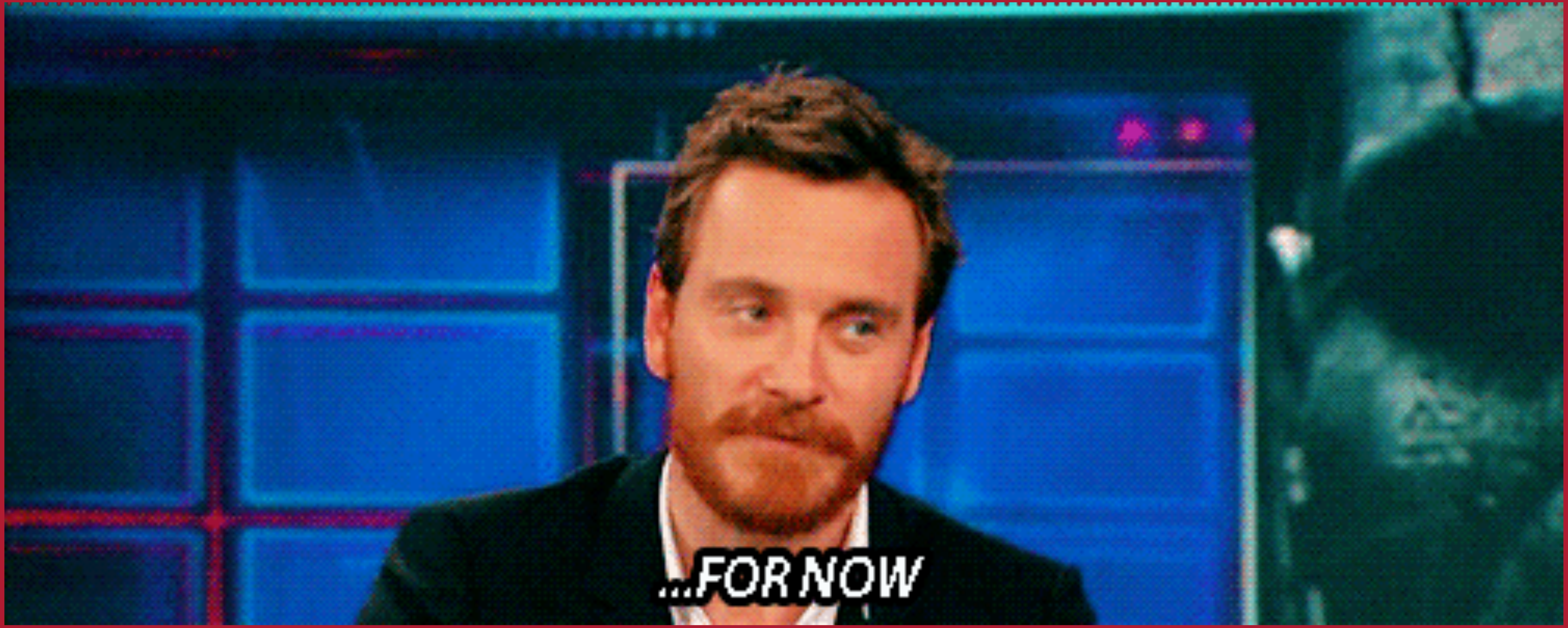
	Distance Vector e.g RIP	Link State e.g OSPF
Resilience	Very slow recovery due to count to infinity.	Re-Run Dijkstra and you’re good to go.
Fully Distributed	Yes	Yes
State per Node (+ Routing Table)	O(# switches * max node degree)	O(# edges)
Convergence	Need to run DV before routing — takes length of longest best path time.	Flood network w/ updates and then run Dijkstra: $O(\|E\| + \|V\| \log \|V\|)$
Control Plane Complexity		

	Distance Vector e.g RIP	Link State e.g OSPF
Resilience	Very slow recovery due to count to infinity.	Re-Run Dijkstra and you’re good to go.
Fully Distributed	Yes	Yes
State per Node (+ Routing Table)	O(# switches * max node degree)	O(# edges)
Convergence	Need to run DV before routing — takes length of longest best path time.	Flood network w/ updates and then run Dijkstra: $O(\|E\| + \|V\| \log \|V\|)$
Control Plane Complexity	Just select the “min” of all the updates I have heard from. (Dumb-ish Switch)	

	Distance Vector e.g RIP	Link State e.g OSPF
Resilience	Very slow recovery due to count to infinity.	Re-Run Dijkstra and you're good to go.
Fully Distributed	Yes	Yes
State per Node (+ Routing Table)	O(# switches * max node degree)	O(# edges)
Convergence	Need to run DV before routing — takes length of longest best path time.	Flood network w/ updates and then run Dijkstra: O(|E| + |V| log |V|)
Control Plane Complexity	Just select the "min" of all the updates I have heard from. (Dumb-ish Switch)	Rebuild network topology, run Dijkstra's algorithm over it.

£$€

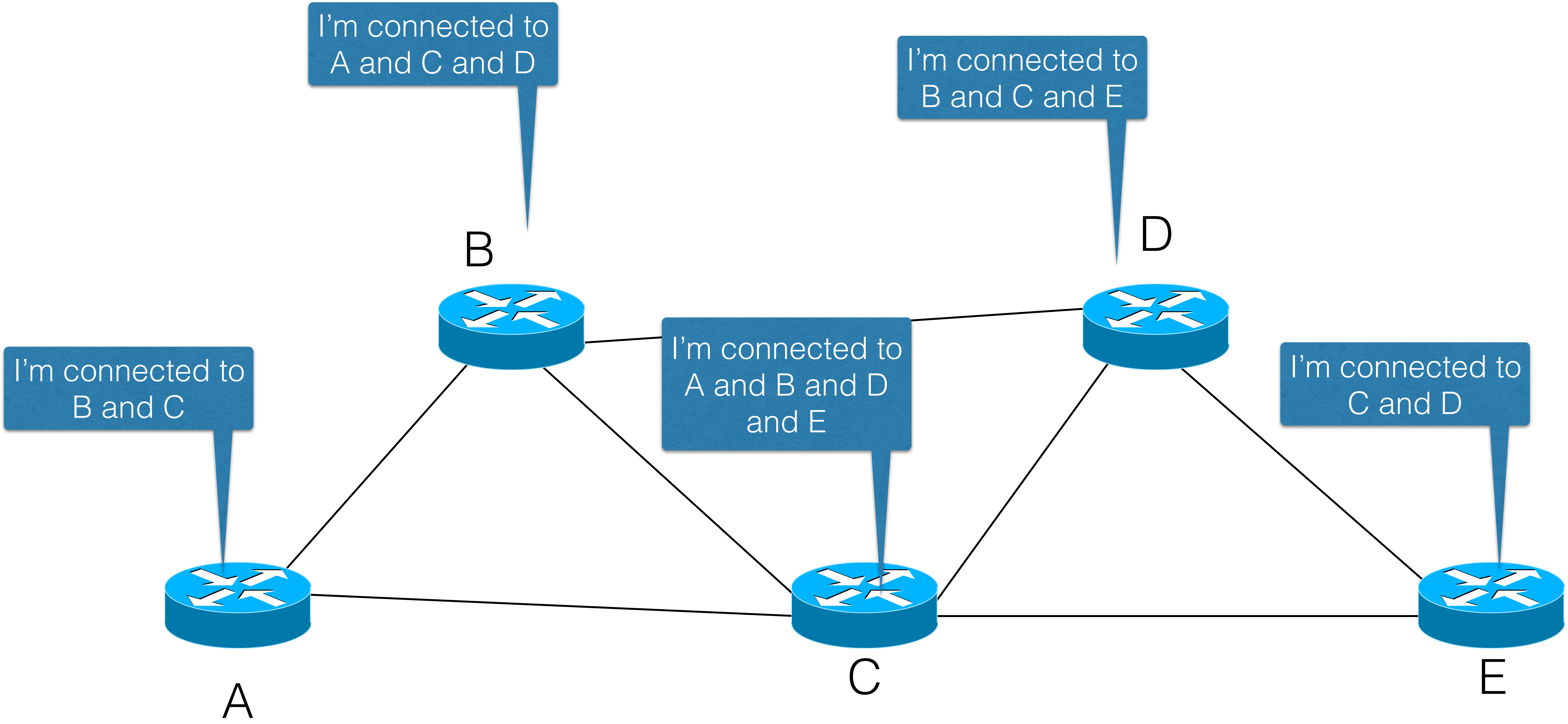
Last Routing Algorithm…

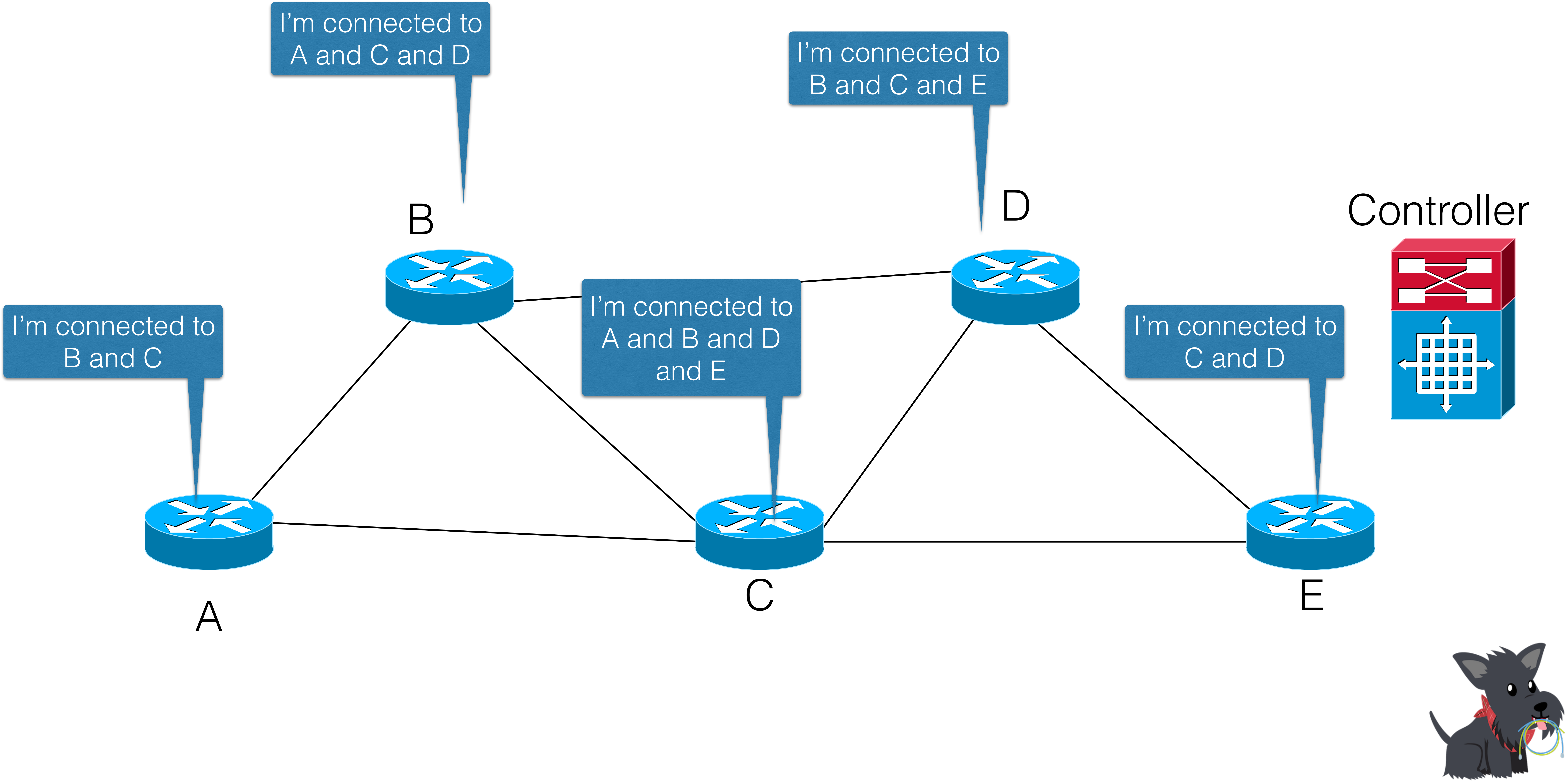

...FOR NOW

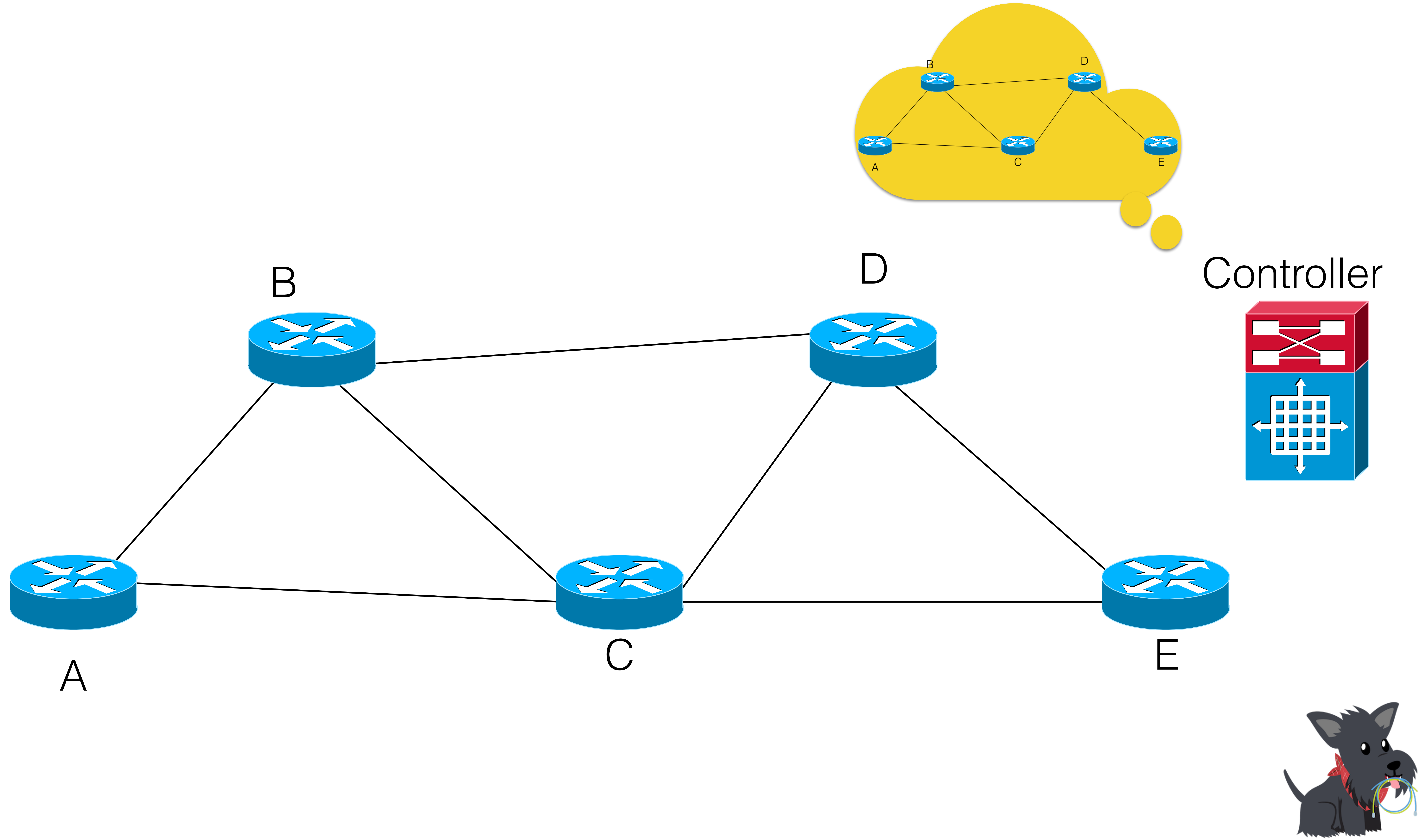
Centralized Routing
(aka “Software-Defined Networking”)

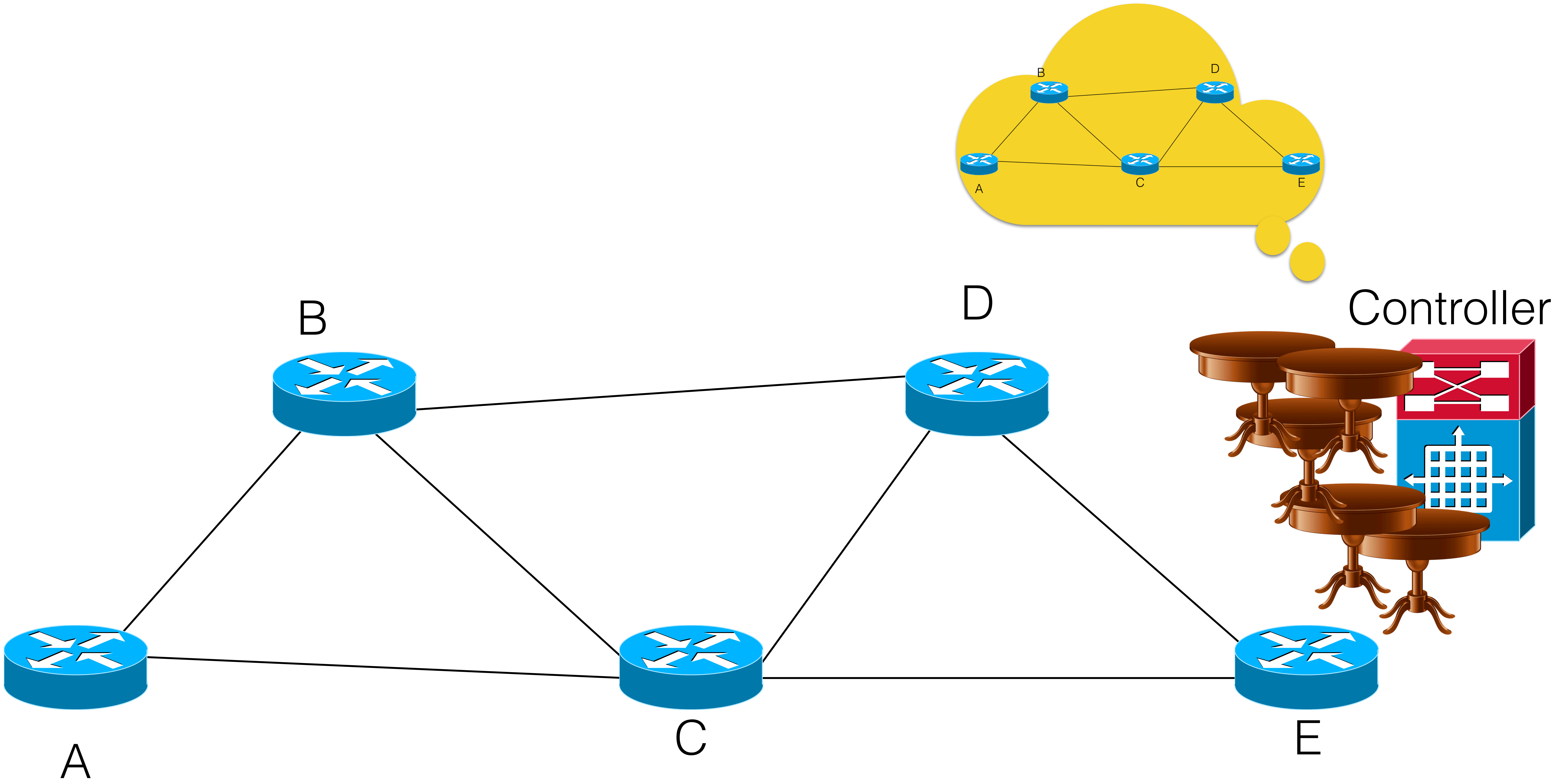
Centralized Routing, In a Nutshell

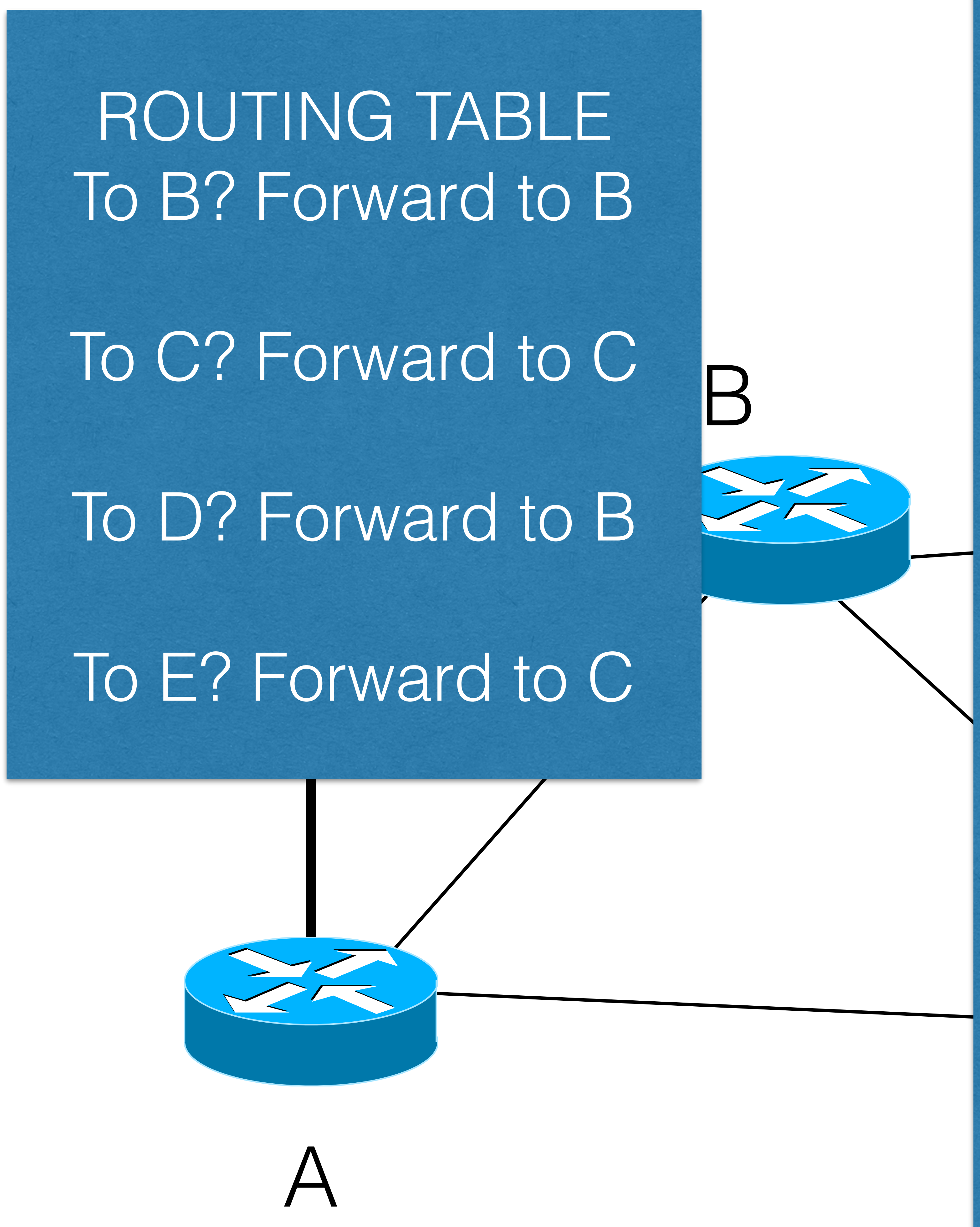
- Like Link State, every node knows who it is connected to.
- Instead of broadcasting to every other node, all nodes tell a special *controller* node who they are connected to.
- The controller computes the best routes for everyone.
- The controller then tells every node what entries to put in their routing tables.


I'm connected to
A and C and D
I'm connected to
B and C and E
B
D
I'm connected to
B and C
I'm connected to
A and B and D
and E
I'm connected to
C and D
A
C
E


I'm connected to
A and C and D
I'm connected to
B and C and E
Controller
B
D
I'm connected to
B and C
I'm connected to
A and B and D
and E
I'm connected to
C and D
A
C
E


B
D
A
C
E
Controller
B
D
A
C
E


B
D
A
C
E
Controller
B
D
A
C
E



Now each switch remembers the new routing table.

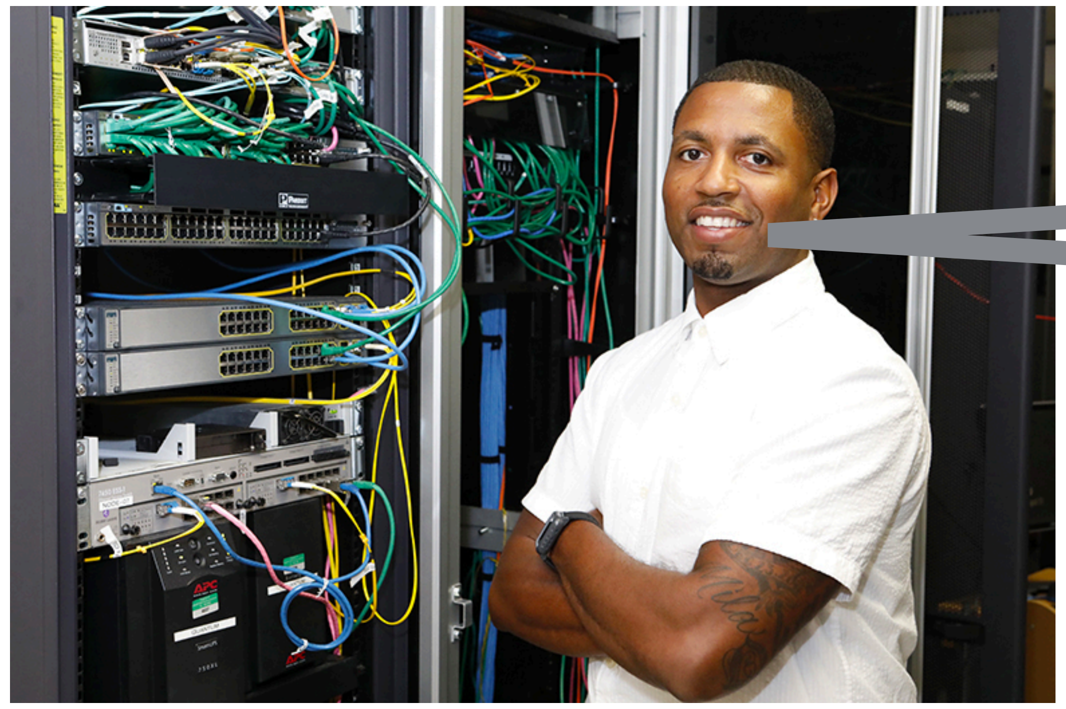
If a link or node fails, the switches notify the controller. The controller re-computes each node's route and sends the new routes out.

	Distance Vector e.g RIP	Link State e.g OSPF	Centralized e.g SDN
Resilience	Very slow recovery due to count to infinity.	Re-Run Dijkstra and you're good to go.	Doesn't recover at all if controller(s) fail.
Fully Distributed	Yes	Yes	No
State per Node (+ Routing Table)	O(# switches * max node degree)	O(# edges)	O(C) (Besides Routing Table)
Convergence	Need to run DV before routing — takes length of longest best path time.	Flood network w/ updates and then run Dijkstra: $O(\|E\| + \|V\| \log \|V\|)$	No distributed convergence
Control Plane Complexity	Just select the "min" of all the updates I have heard from. (Dumb-ish Switch)	Rebuild network topology, run Dijkstra's algorithm over it.	EXTREMELY SIMPLE

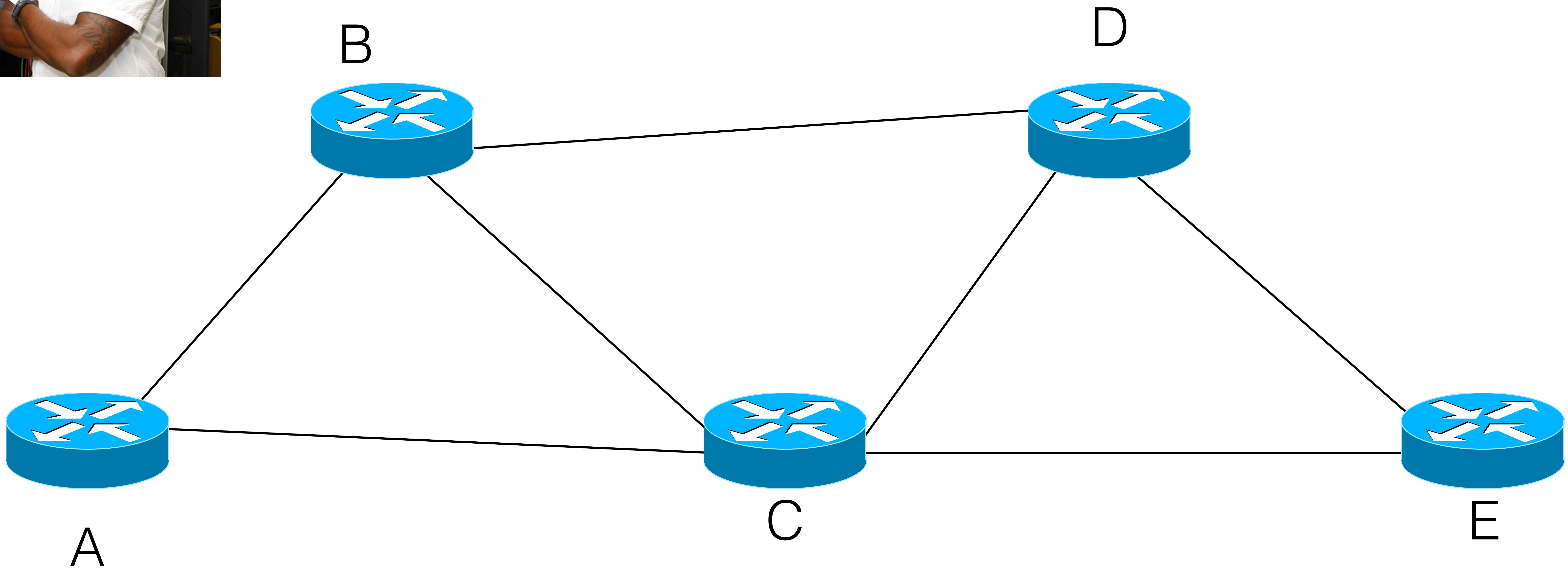
	Distance Vector e.g RIP	Link State e.g OSPF	Centralized e.g SDN
Resilience	Very slow recovery due to count to infinity.	Re-Run Dijkstra and you're good to go.	Doesn't recover at all if controller(s) fail.
Fully Distributed	Yes	Yes	No
State per Node (+ Routing Table)	O(# switches * max node degree)	O(# edges)	O(C) (Besides Routing Table)
Convergence	Need to run DV before routing — takes length of longest best path time.	Flood network w/ updates and then run Dijkstra: O(|E| + |V| log |V|)	No distributed convergence
Control Plane Complexity	Just select the "min" of all the updates I have heard from. (Dumb-ish Switch)	Rebuild network topology, run Dijkstra's algorithm over it.	EXTREMELY SIMPLE


How could you think that was a good idea?

Fun Fact: Centralized Routing is considered “state of the art” — why in the world would people choose this over other designs that are fundamentally more resilient??

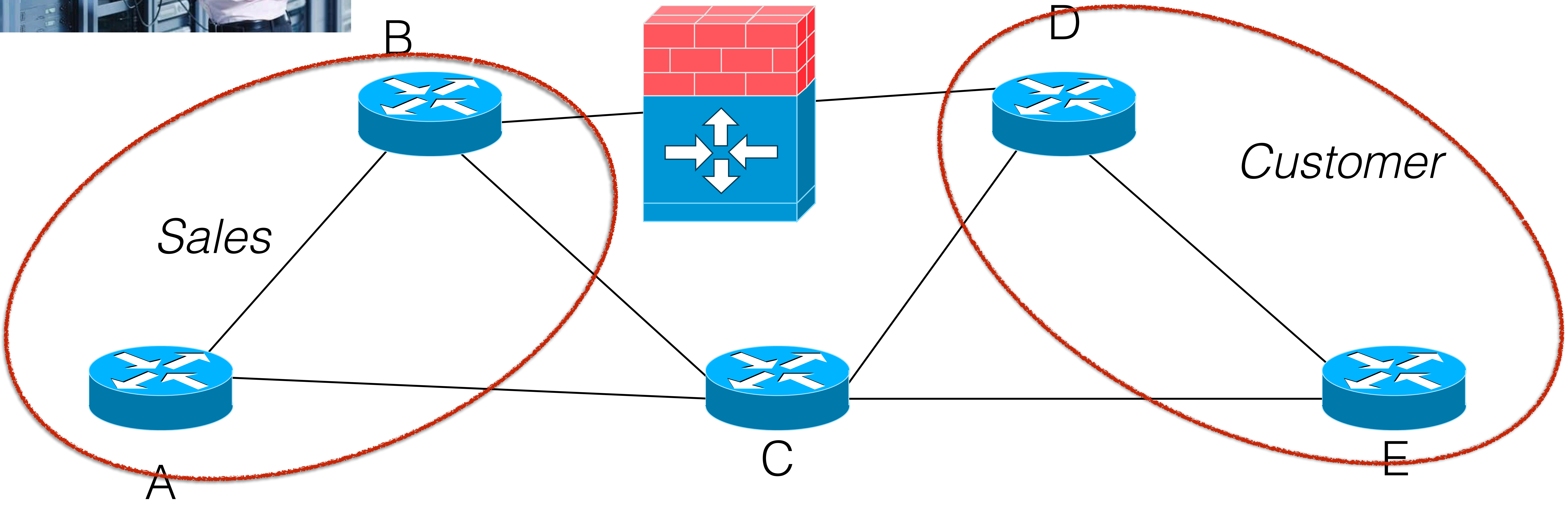


I want my network to work normally, EXCEPT B should NOT be allowed to communicate with E.



Traditional routing algorithms are designed to achieve global reachability — but can't enforce *policy requirements.*

I need to make sure all traffic going from my sales network to my customer network goes through a firewall.



Traditional routing algorithms are designed to achieve global reachability — but can't enforce *policy requirements.*

Network Policy

- You want to tell the network an “exception” or a “special case”
  - Something to do other than “Let everyone talk to each other!”
- With fully distributed algorithms, you have to distribute the policy
  - And different nodes have to behave differently! You might even need to configure each node specially, depending on the policies.
- With a centralized controller, you configure the controller with your policy. The controller makes the decisions, and the switches don’t have to be configured specially to apply the policy.
  - They just receive their routing tables from the controller.

	Distance Vector e.g RIP	Link State e.g OSPF	Centralized e.g SDN
Resilience	Very slow recovery due to count to infinity.	Re-Run Dijkstra and you're good to go.	Doesn't recover at all if controller(s) fail.
Fully Distributed	Yes	Yes	No
State per Node (+ Routing Table)	O(# switches * max node degree)	O(# edges)	O(C)
Convergence	Need to run DV before routing — takes length of longest best path time.	Flood network w/ updates and then run Dijkstra: O(|E| + |V| log |V|)	No distributed convergence
Control Plane Complexity	Just select the "min" of all the updates I have heard from. (Dumb-ish Switch)	Rebuild network topology, run Dijkstra's algorithm over it.	EXTREMELY SIMPLE
Network Policy Support	Hard	Hard	Easy

So what algorithm is best?

Well, it depends…

- Resilience
- Fully Distributed
- Convergence
- Network Policy Support
- State per Node (+ Routing Table)
- Routing Efficiency
- Shortest Path?
- Control Plane Complexity

Do I want cheap, dumb switches, or smart ones?

Is my network big or small?

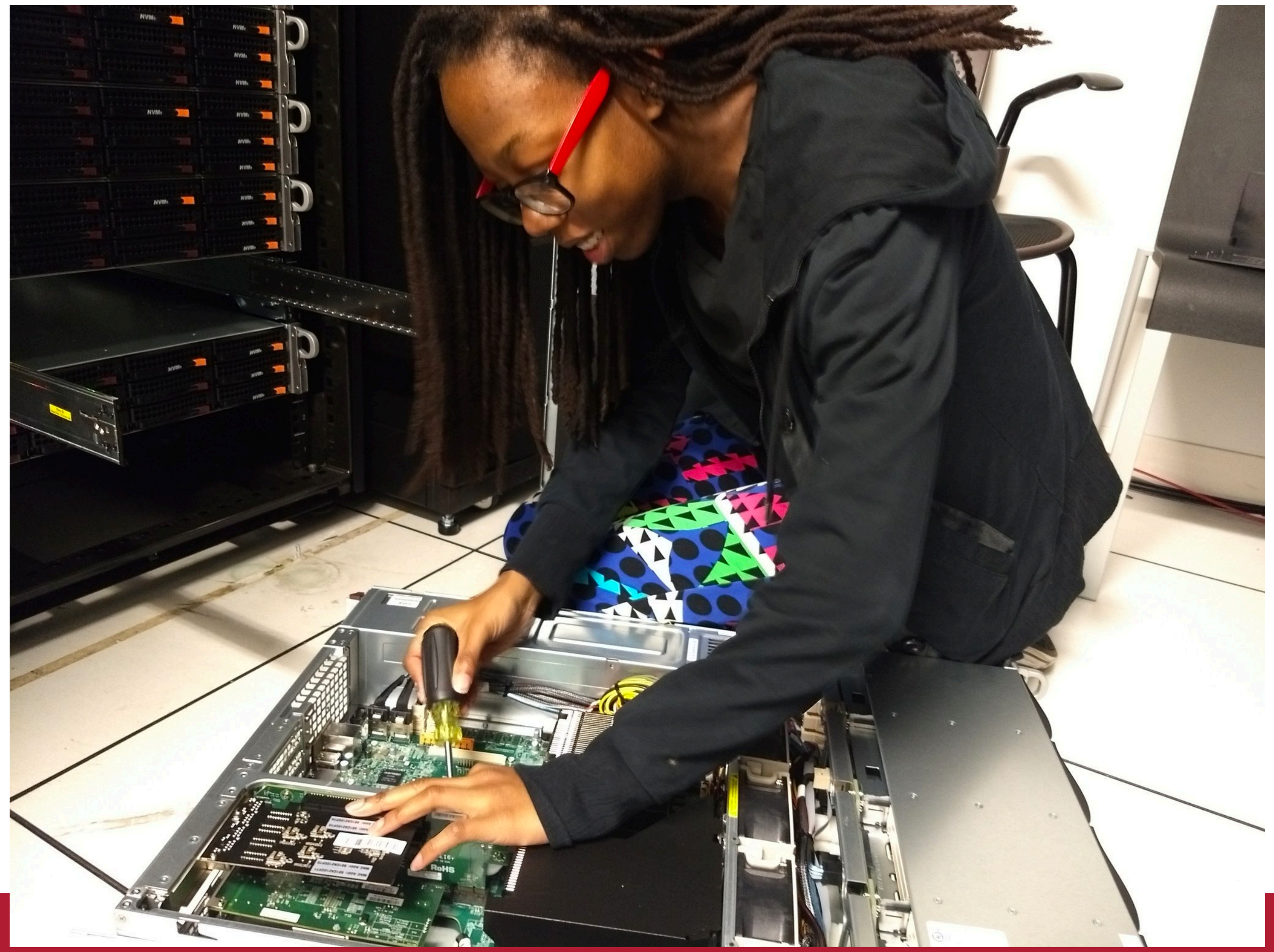
Do I have network policies to enforce?

Do I need my network to survive a hurricane? earthquake? natural disaster?

In Practice

- My rack of servers in my research lab just uses broadcast routing!
- There are only six machines, connected by one switch.
- Most small setups like this just use broadcast routing.

My student Ray installing new network cards in the machine room.

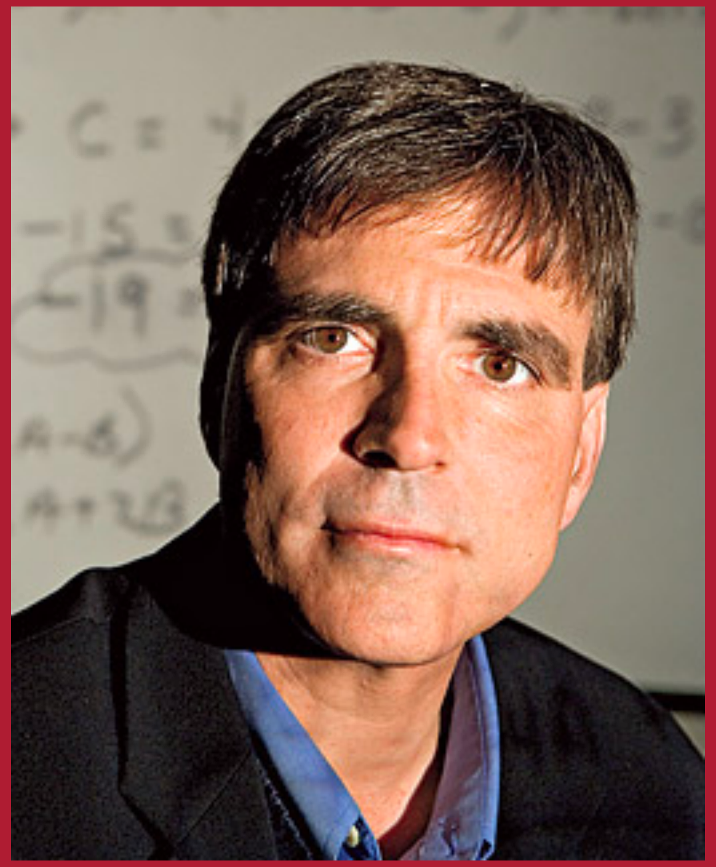


In Practice

- Google’s network spans across the whole world
  - This is called a “WAN” — a “wide area network”
  - Still administered by one organization — so it’s one network (not the INTERnet). But it’s very big.
- This network is called B4 and it uses a COMBINATION of Link State Routing (OSPF) and Centralized Routing (SDN)
- Just for fun: you can read about this network here: https://dl.acm.org/citation.cfm?id=2486019

Systems Engineering Wisdom

"Engineering isn't about perfect solutions. It's about doing the best you can with limited resources."



— Randy Pausch
CMU Professor, ACM Fellow
(October 23, 1960 – July 25, 2008)

You have survived basic routing!

- We will learn a few more routing algorithms later in this class.
    - But now you are read to move on to learn about…
        - THE INTERNET!!!!
            - Any questions before we move on?

Questions? Comments? Concerns? Feedback?

Fill out an anonymous notecard on your way out.