CS144
An Introduction to Computer Networks

Routing – Lecture 1

Nick McKeown
Professor of Electrical Engineering
and Computer Science, Stanford University

Videos and Lectures this week

Lectures (Wed and Fri): Mostly the “why” we do it this way

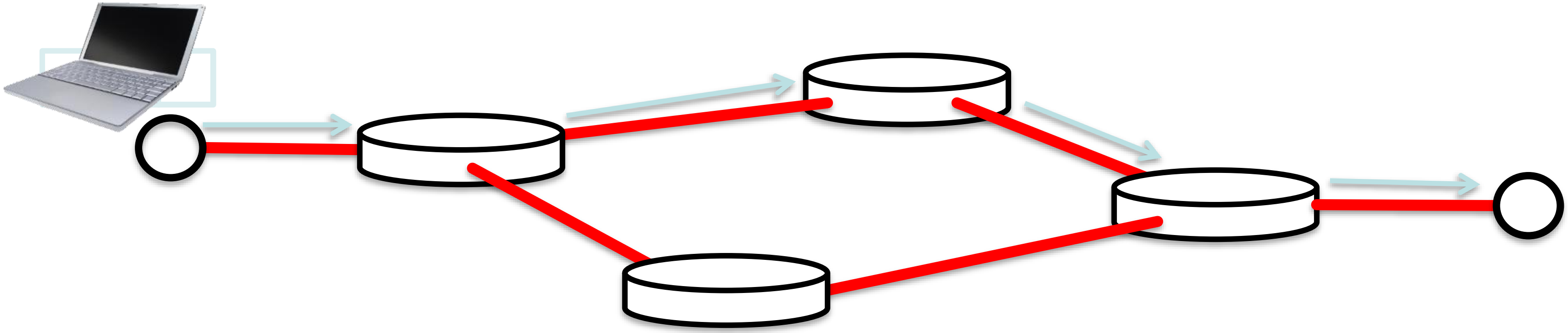
Videos: Mostly the “what” and the “how”

Today’s lecture:

1. Different approaches to routing
2. The Bellman Ford ”distance vector” algorithm

Routers forward packets **one at a time**.

Routers look at IP addresses,
then send packets to a router closer to the destination.



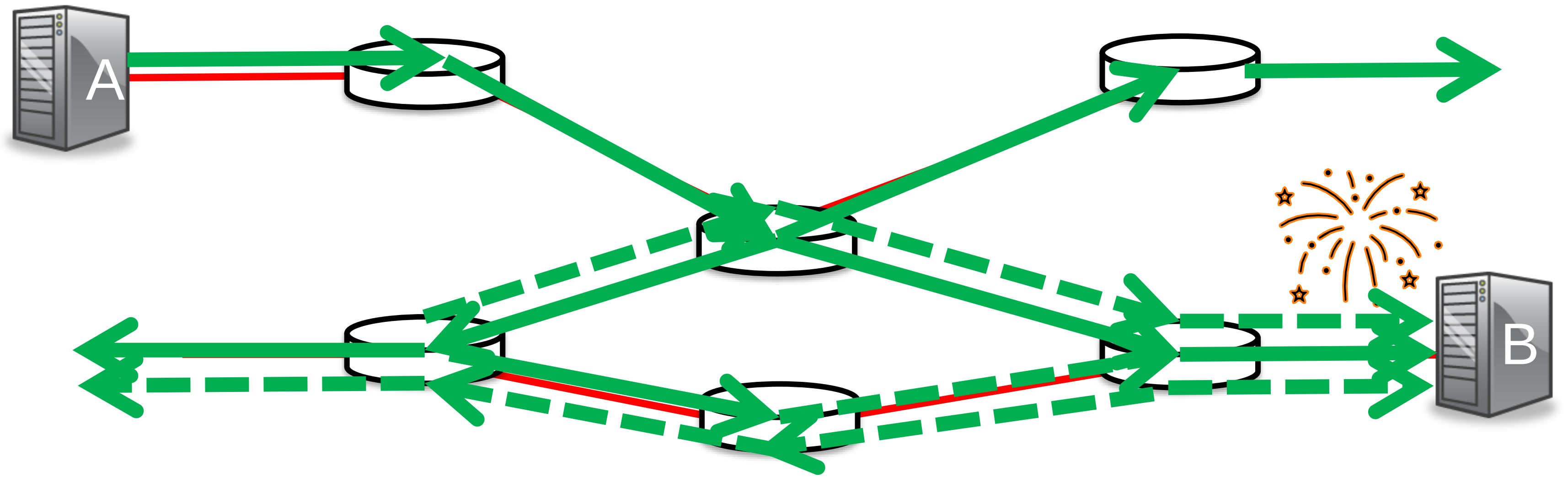
How does a router know
where to send a packet next?

Here are three ways

1. **Flooding**: Every router sends an arriving packet to every neighbor
2. **Source Routing**: End host lists the routers to visit along the way (in each packet)
3. **Distributed Algorithm**: Routers talk to each other and construct forwarding tables using a clever algorithm

1. Flooding

Routers forward an arriving packet to every interface, except the one through which it arrived



Pros

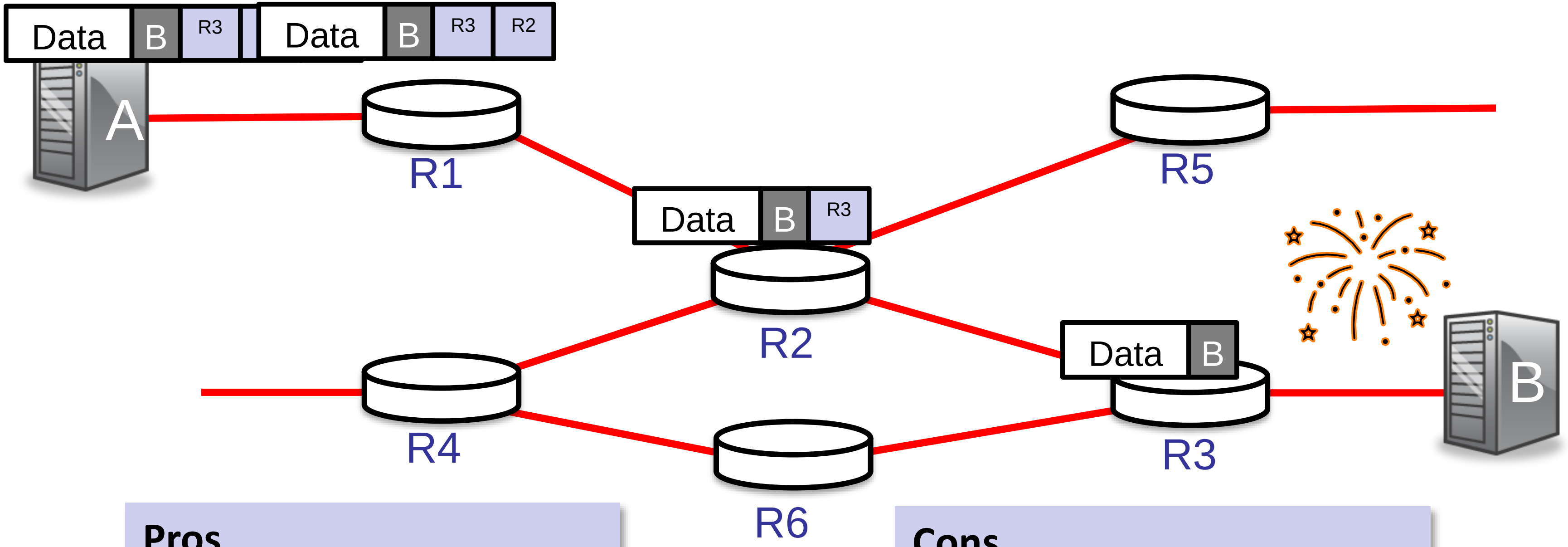
- Packet reaches destination along shortest path
- Works when we don't know the topology

Cons

- Packets can loop forever (need TTL!)
- Inefficient use of the links
- Packets are delivered to everyone

2. Source Routing

Source includes a list of the routers along the path



Pros
- Source picks the path
- No loops
- No need for tables in routers

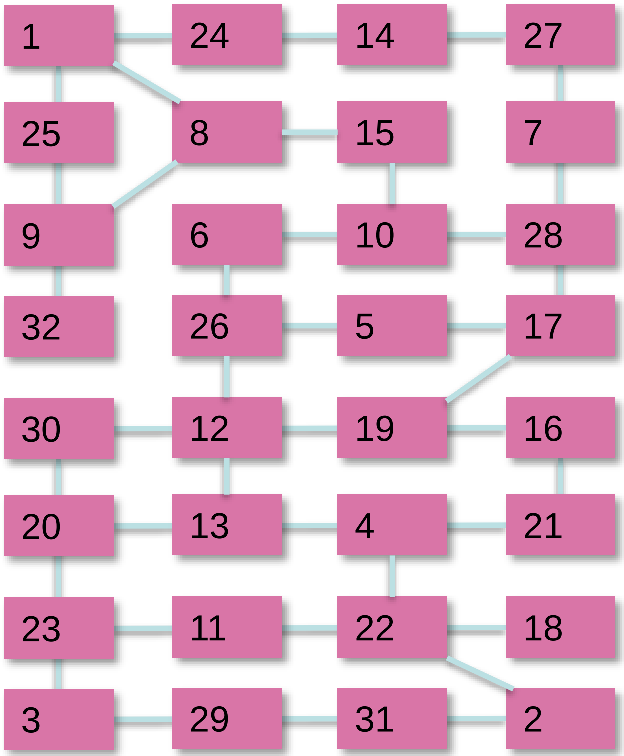
Cons
- Source needs to know topology
- Potentially large headers

Here are three ways

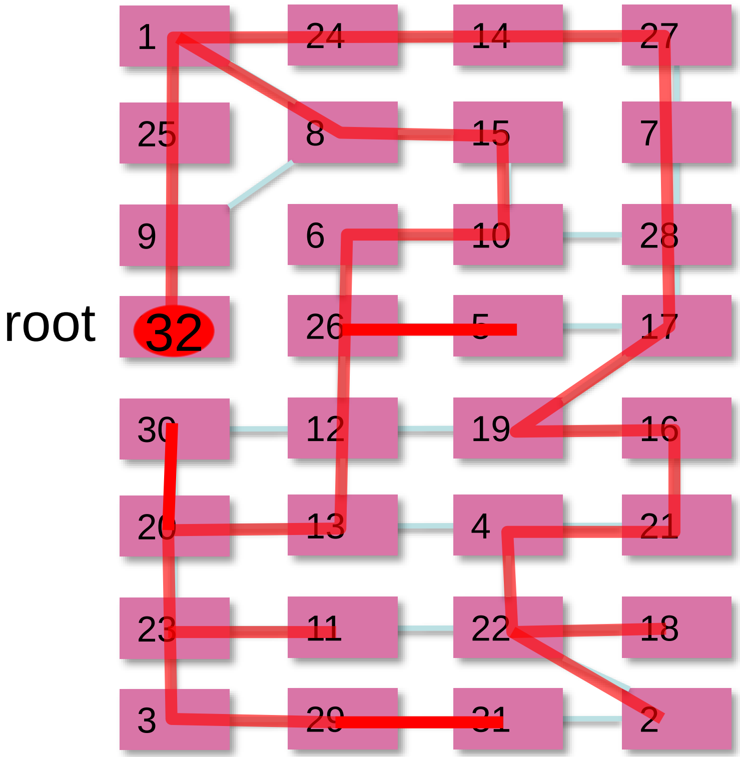
1. **Flooding**: Every router sends an arriving packet to every neighbor
2. **Source Routing**: End host lists the routers to visit along the way (in each packet)
3. **Distributed Algorithm**: Routers talk to each other and construct forwarding tables using a clever algorithm

The rest of today's lecture....

In this network of 32 routers, how can the routers forward packets, based only on the destination address, so each packet is delivered to the correct router, exactly once?

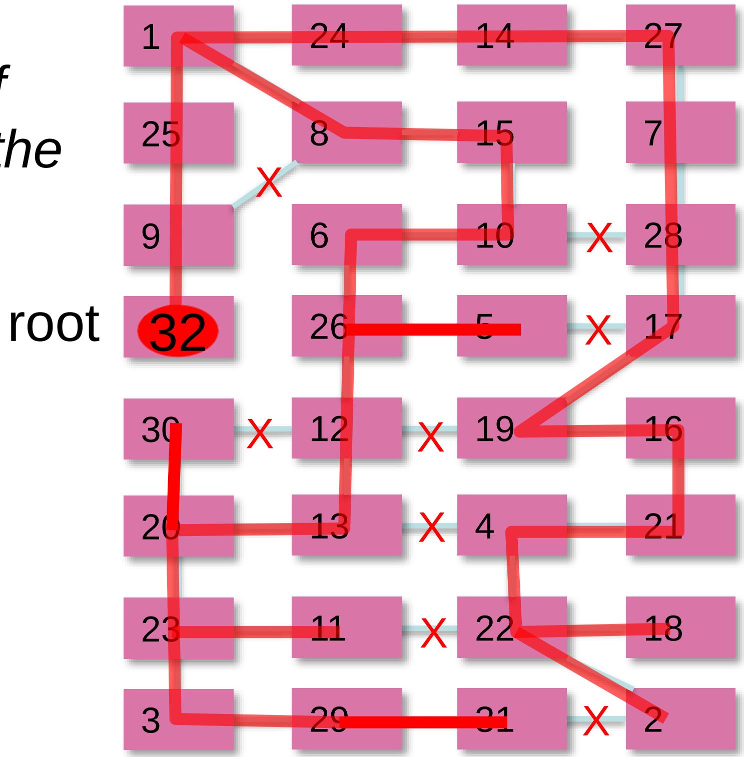


In this network of 32 routers, how can the routers forward packets, based only on the destination address, so each packet is delivered to the correct router, exactly once?



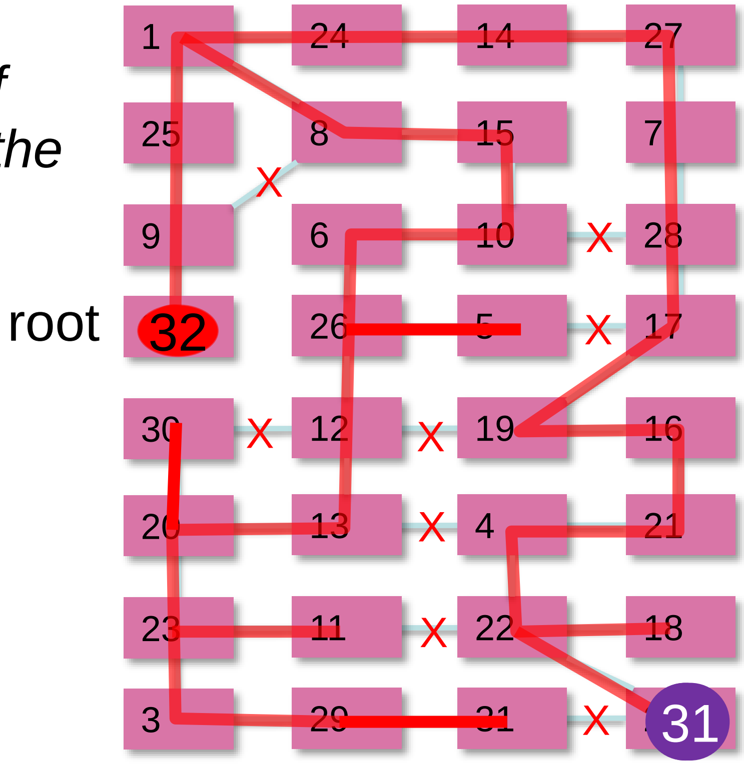
In this network of 32 routers, how can the routers forward packets, based only on the destination address, so each packet is delivered to the correct router, exactly once?

We could switch off all the links not on the spanning tree…



In this network of 32 routers, how can the routers forward packets, based only on the destination address, so each packet is delivered to the correct router, exactly once?

We could switch off all the links not on the spanning tree…



But…

1. *Paths can be crazy long*
2. *Some links are unused*
3. *Need to remember to switch unused links back on if needed*

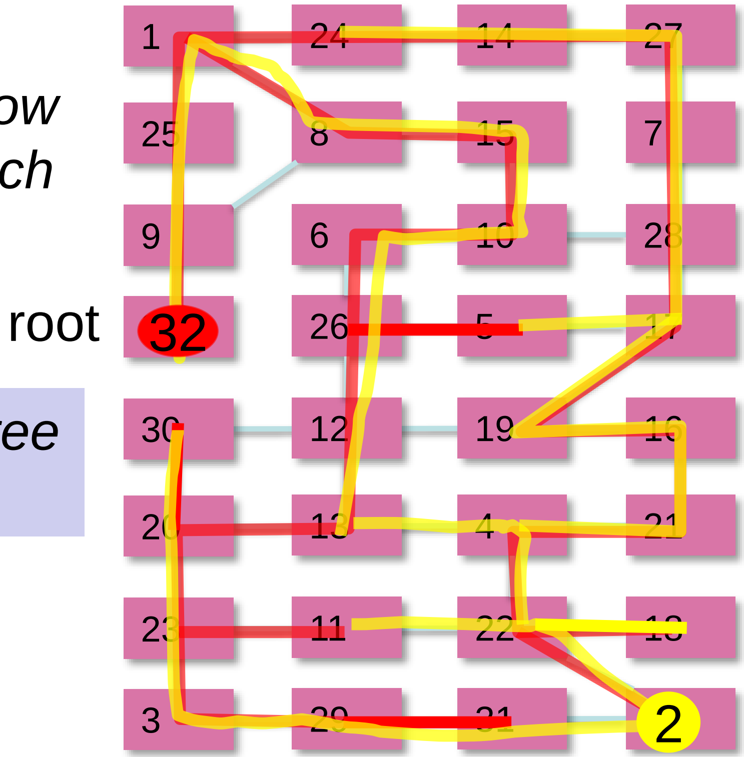
Observations

- Ethernet switches build a single spanning tree between them. Some links are switched off. Packets follow the spanning tree. In Video 5-7 you will learn about the “Spanning Tree Protocol” that Ethernet switches use.

- Routers instead work together, to build a separate spanning tree ***rooted at each destination***.

In this network of 32 routers, how can the routers forward packets, based only on the destination address, so each packet is delivered to the correct router, exactly once?

Packets could follow the red tree to reach router 32…



…and the yellow tree to reach router 2.

The problem becomes:

For each destination, a router needs to put an entry in its forwarding table to forward packets along the spanning tree rooted at that destination.

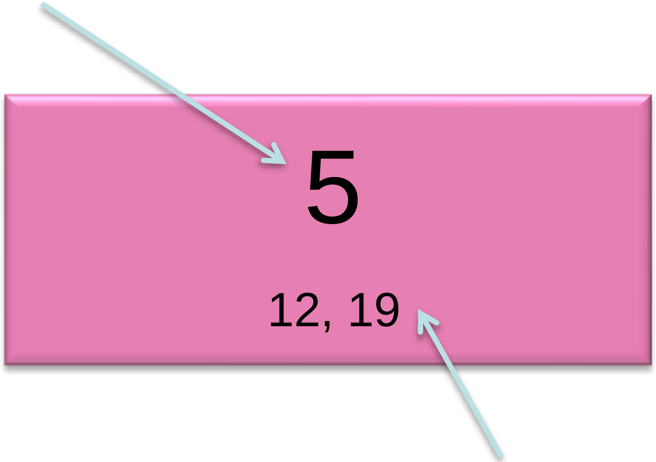
How does it know what entry to add?

The non-pandemic edition

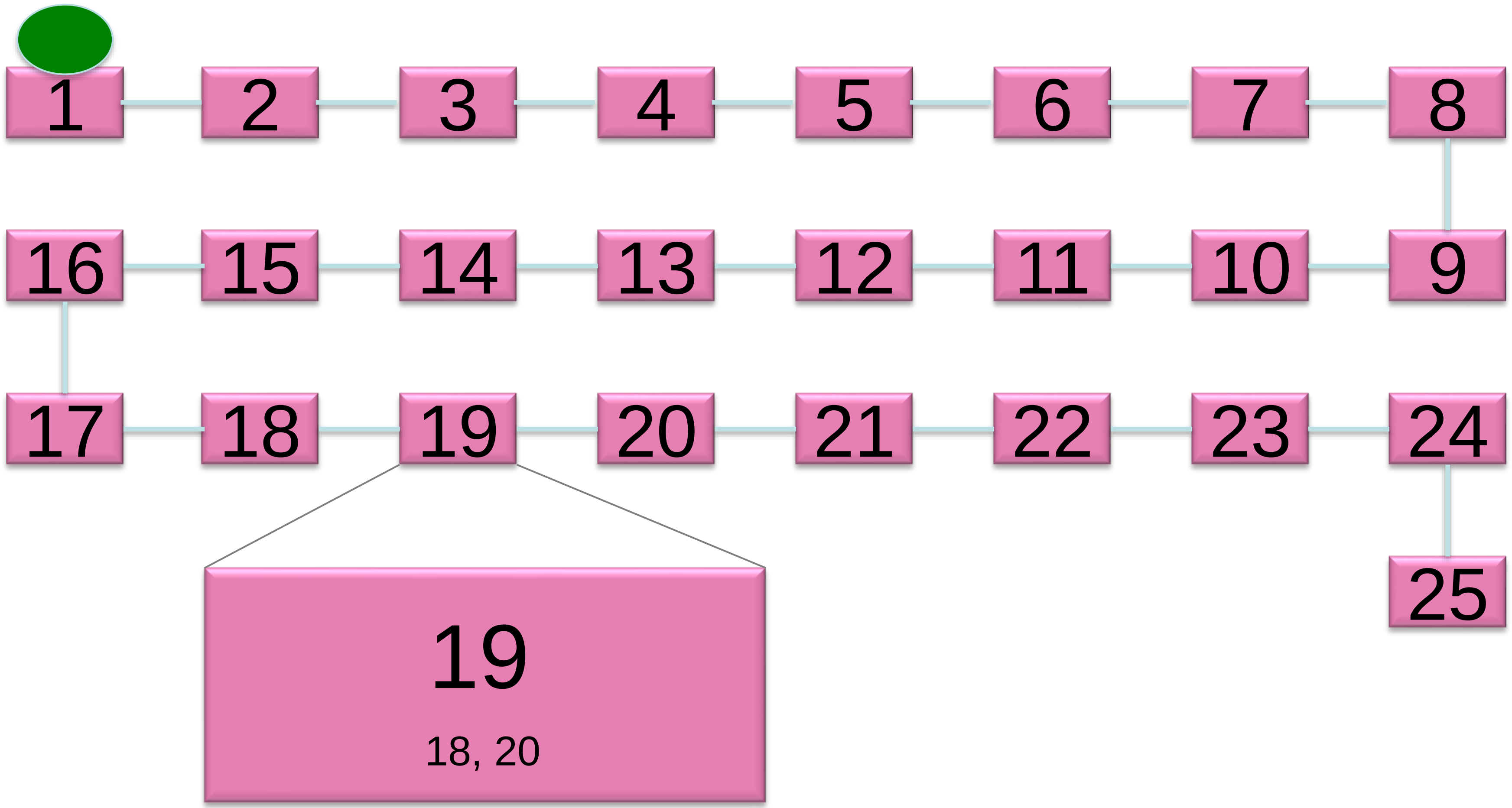
Game: Routing Competition

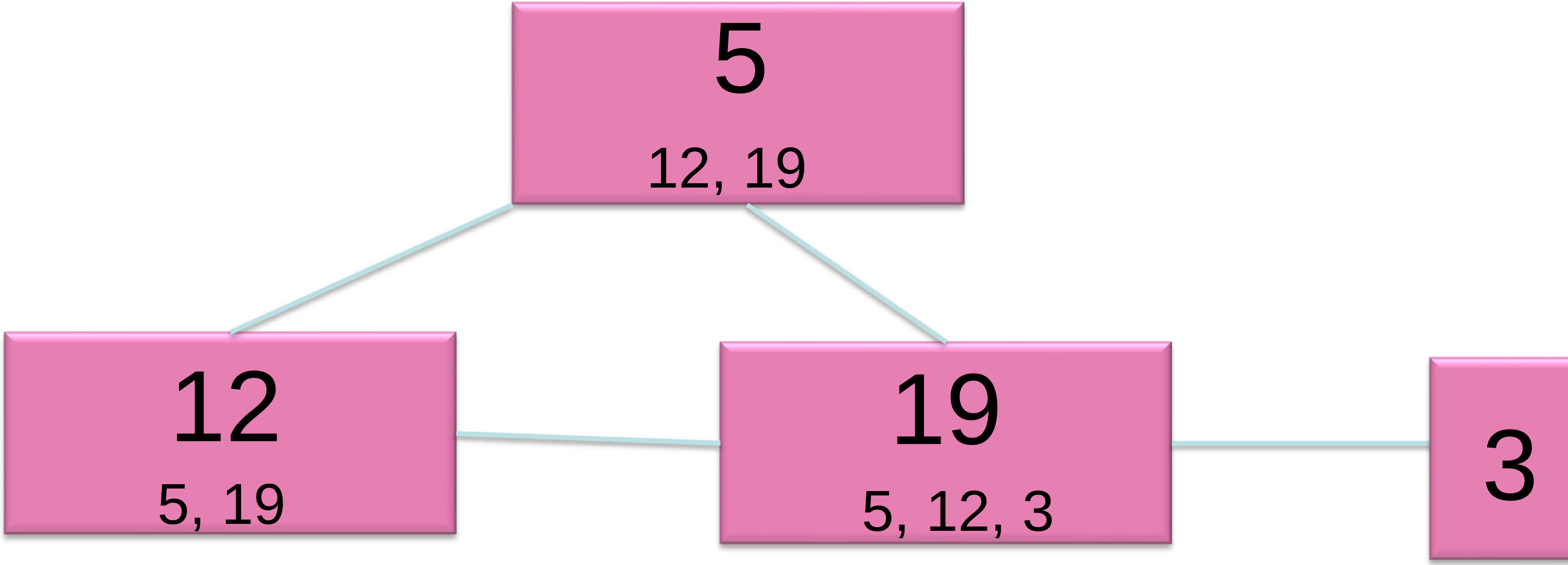
Each team member has a card

Your router ID



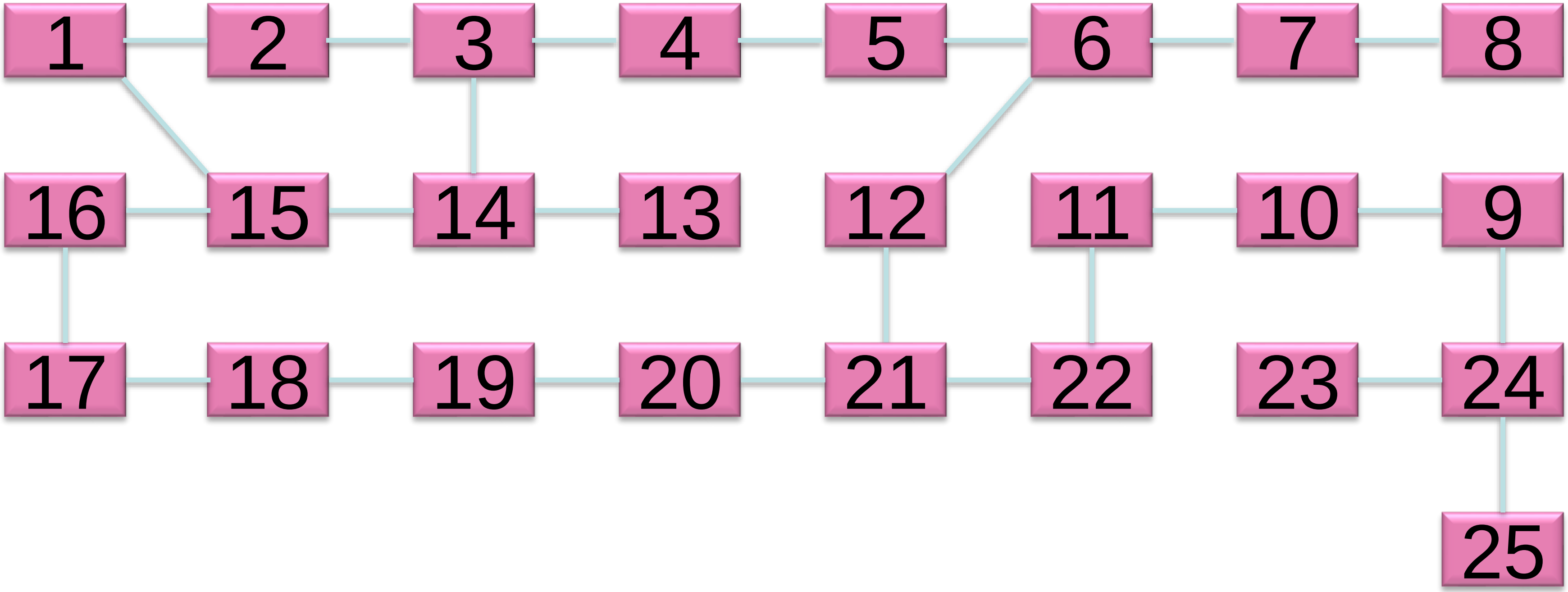
The IDs of your neighbors


1
2
3
4
5
6
7
8
16
15
14
13
12
11
10
9
17
18
19
20
21
22
23
24
25
19
18, 20



5
12, 19
12
5, 19
19
5, 12, 3
3

Find the shortest path



In a real network, the routers don’t know what the network looks like.

This time, *I won’t show you the network*.

The non-pandemic edition

Rules

You may not

- Pass your card to anyone else
- Leave your seat
- Write anything down

You may

- Ask nearby friends (in your group) for advice
- Shout to other participants (anything you want!!!)
- Say bad things about Nick

Task

Find the shortest path from
Node 1 to Node 40.

When you are done, you must be able
to repeat it correctly.

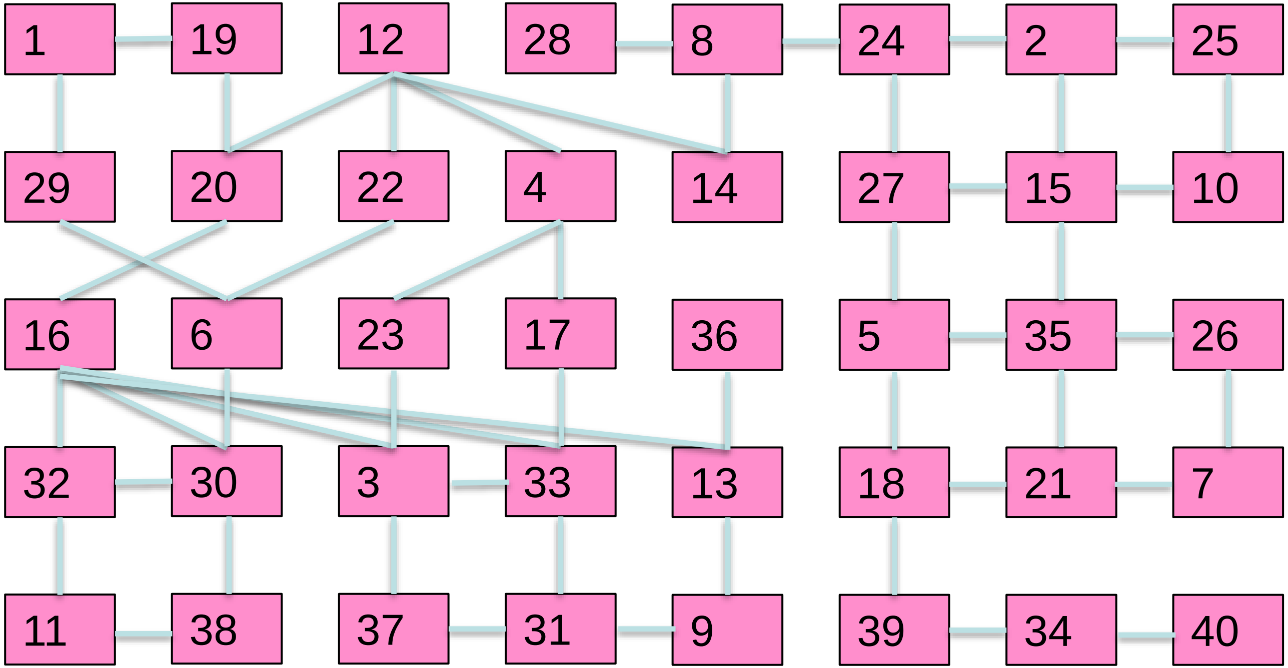
The first group to finish is the champion!!

Pink Group

1	…	…	…	…	…	2	…
…	…	…	…	…	…	…	…
…	…	…	…	…	…	…	…
…	…	3	…	…	…	…	…
…	…	…	…	…	…	…	40

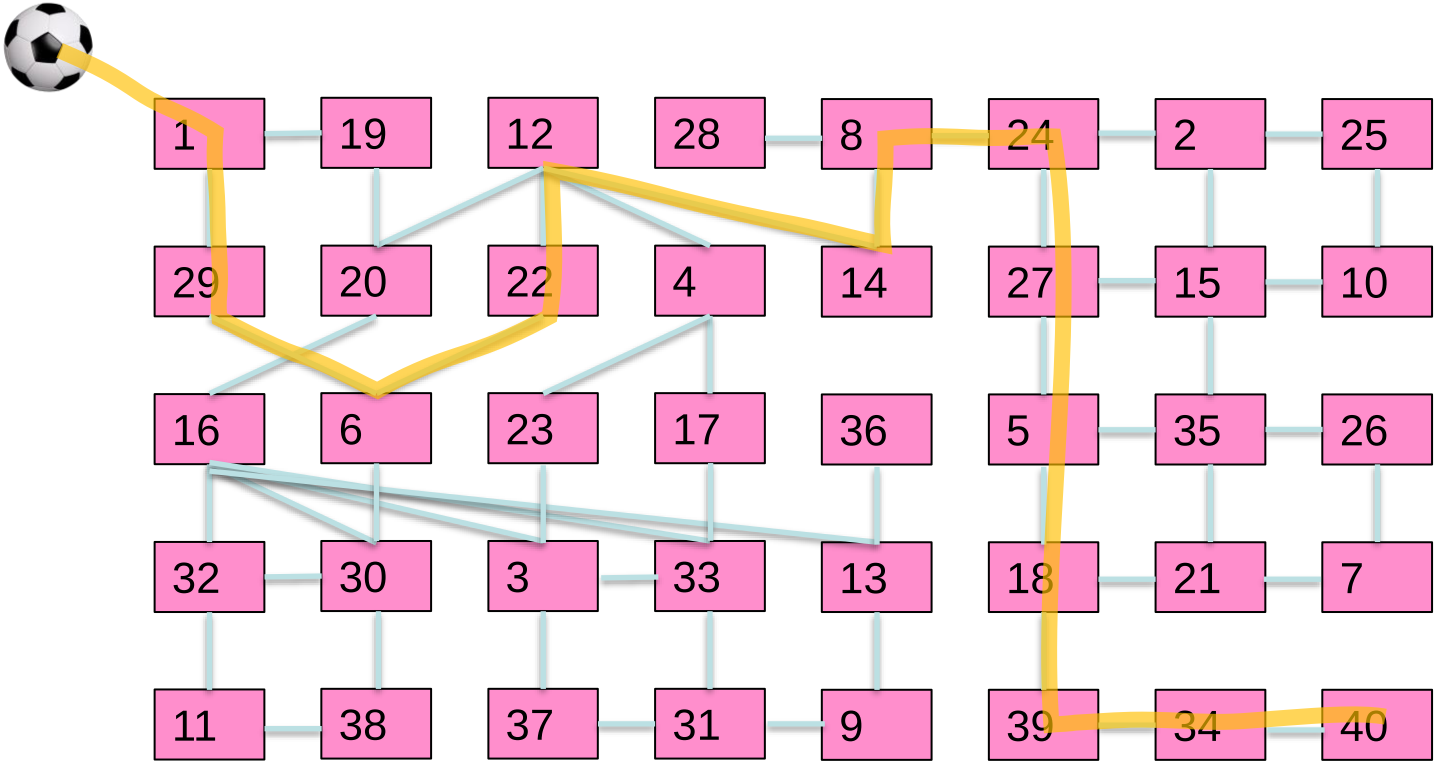
Solution

Pink Group



1
19
12
28
8
24
2
25
29
20
22
4
14
27
15
10
16
6
23
17
36
5
35
26
32
30
3
33
13
18
21
7
11
38
37
31
9
39
34
40

Pink Group



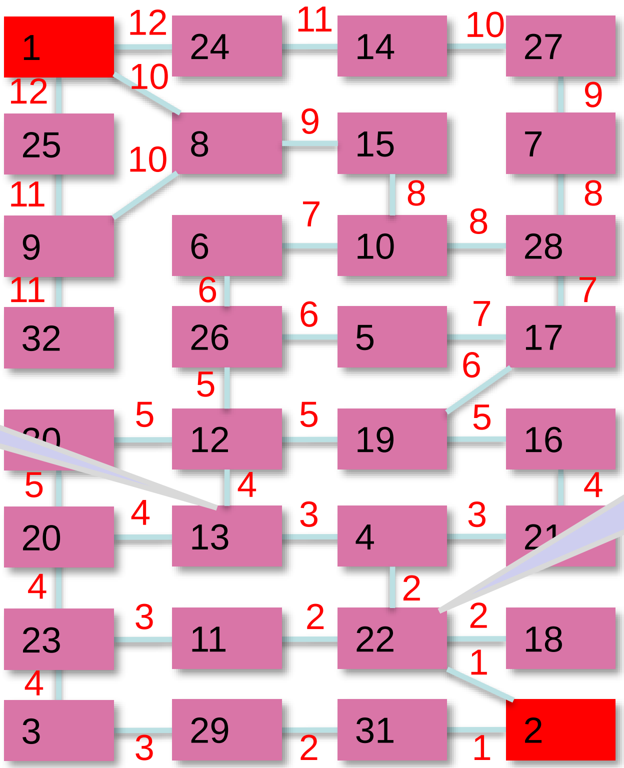
1
19
12
28
8
24
2
25
29
20
22
4
14
27
15
10
16
6
23
17
36
5
35
26
32
30
3
33
13
18
21
7
11
38
37
31
9
39
34
40

How would your team solve it?

An algorithm to find the shortest path spanning tree

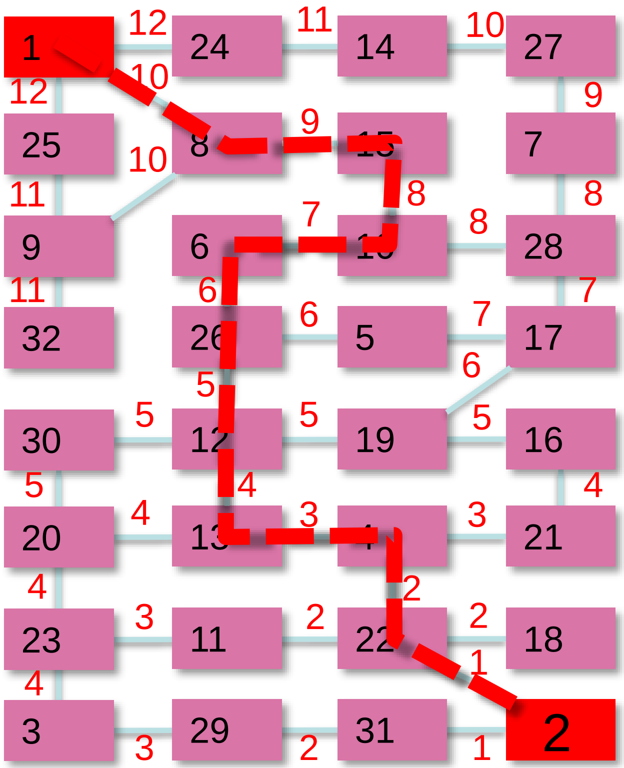
Find the shortest path spanning tree rooted at router 2

IP Address	Next-hop
Node 2	Node 4

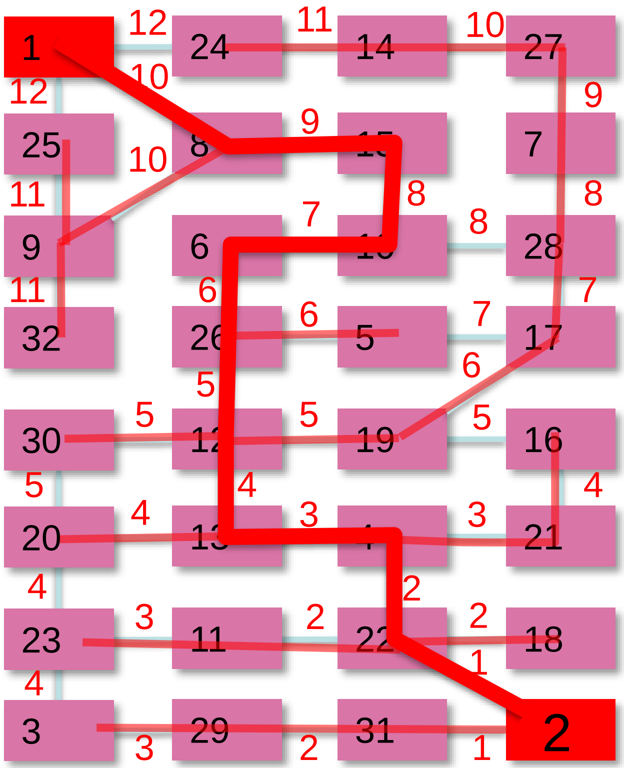


"You can reach node 2 in 1 hop from node 22"

This is the shortest path from 1 to 2



The shortest path spanning tree



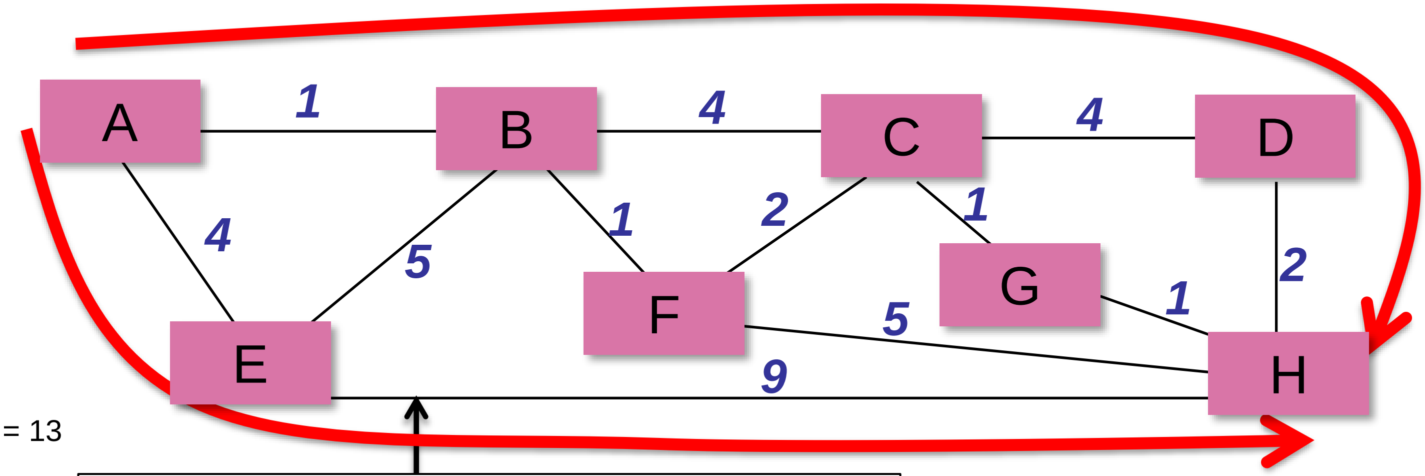
Distributed Algorithm

Questions:

1. What is the maximum run time of the algorithm?
2. Will the algorithm always converge?
3. What happens when routers/links fail?

What if each link has a “cost”?

Cost = 1+4+4+2 = 11

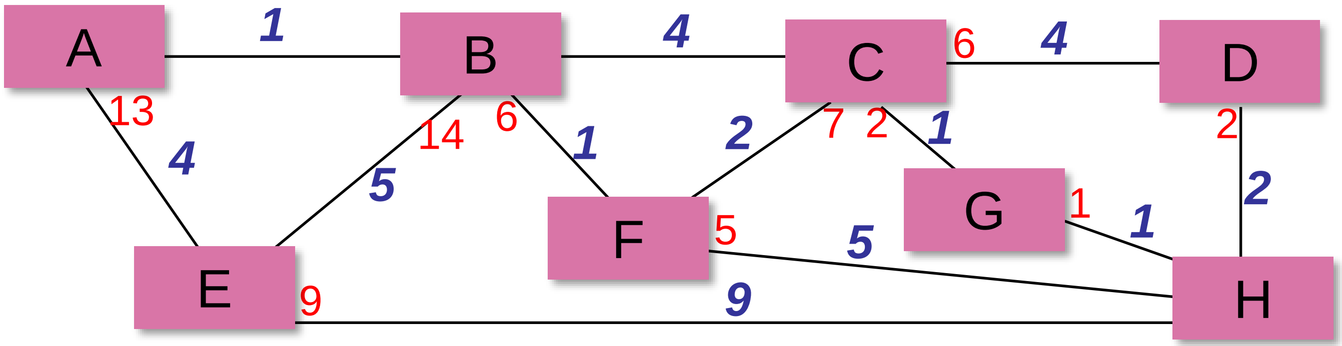


Cost = 4+9 = 13

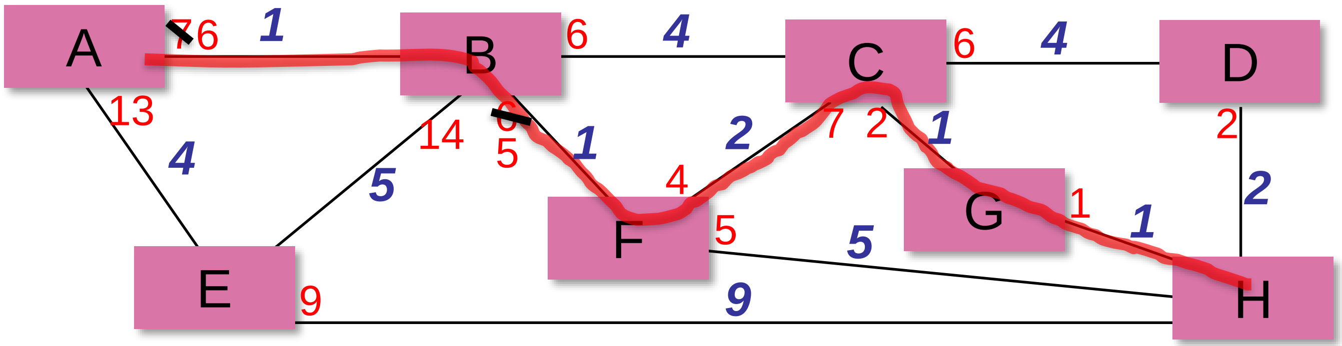
“Expensive link”:

It might be very long. e.g. a link from Europe to USA.
Or it might be very busy. e.g. it connects to Google or CNN.
Or it may be very slow. e.g. 1Mb/s instead of 100Mb/s.

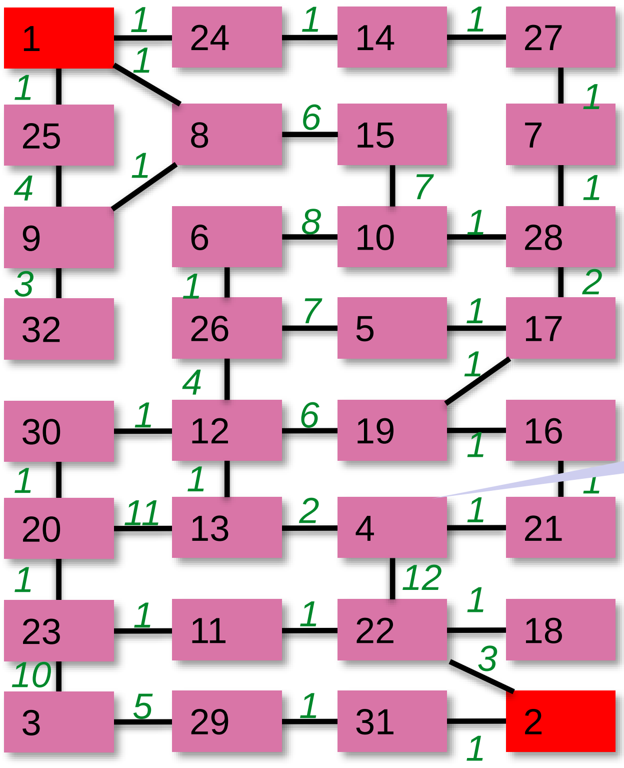
Find lowest cost path to H



Find lowest cost path to H



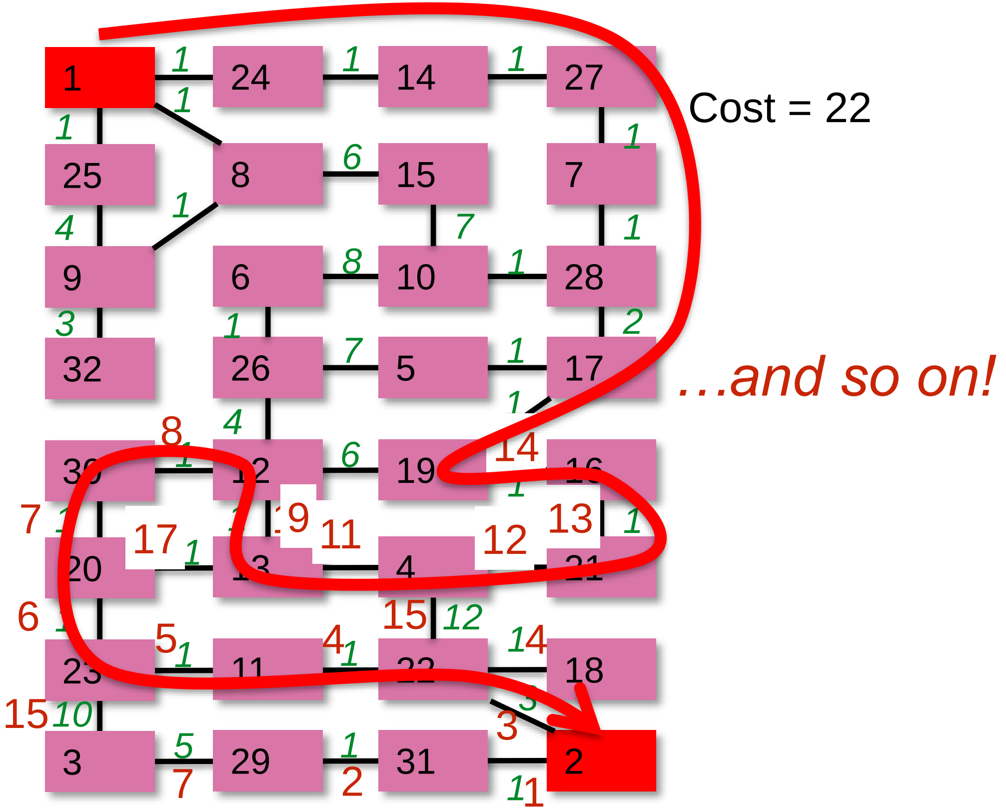
Find the lowest cost path



Router 4 tells its neighbors:

“I can reach 2 with a cost of 15”

Solution



The Distributed Bellman-Ford Algorithm

Example: Find min-cost spanning tree to router **R**

- Assume routers know cost of link to each neighbor.
- Router R_i maintains value of cost C_i to reach **R**, and the next hop**.**
- Vector **C**=(C_1, C_2,…) is the *distance vector* to **R**.
- Initially, set **C** = (∞, ∞, … ∞)
  1. After **T** seconds, R_i sends C_i to its neighbors.
  2. If R_i learns of a lower cost path, update C_i. Remember next hop.
  3. Repeat.

The Distributed Bellman-Ford Algorithm

Questions:

1. What is the maximum run time of the algorithm?
2. Will the algorithm always converge?
3. What happens when routers/links fail?