CS144: An Introduction to Computer Networks

Routing: How do packets know the way?

Today: Different approaches (Part 1 of 3)

Nick McKeown

Videos and Lectures this week

Lectures: Mostly the “why” we do it this way

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

Today’s lecture and discussion:

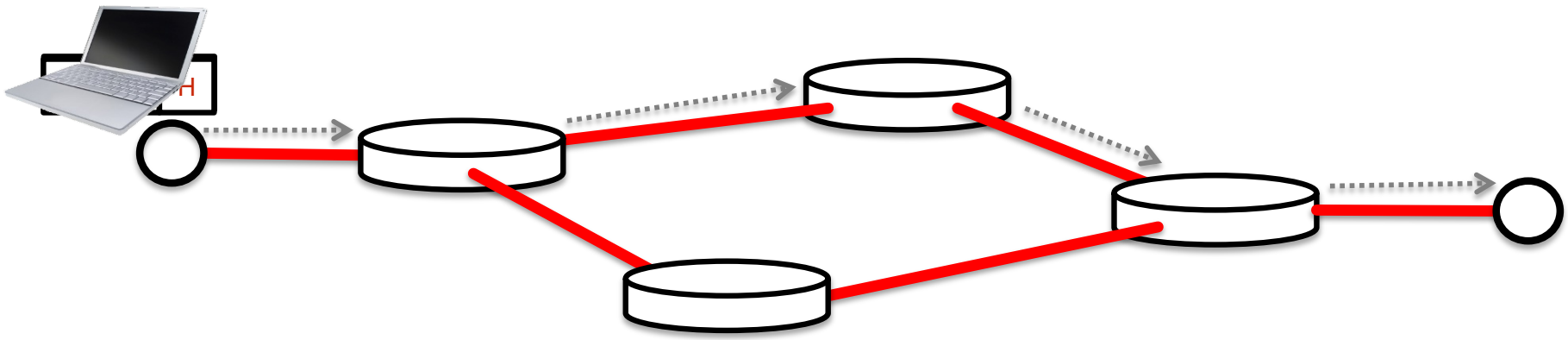
Different approaches to routing

Before Wednesday, watch three Videos:

Basics, Bellman Ford and Dijkstra

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

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



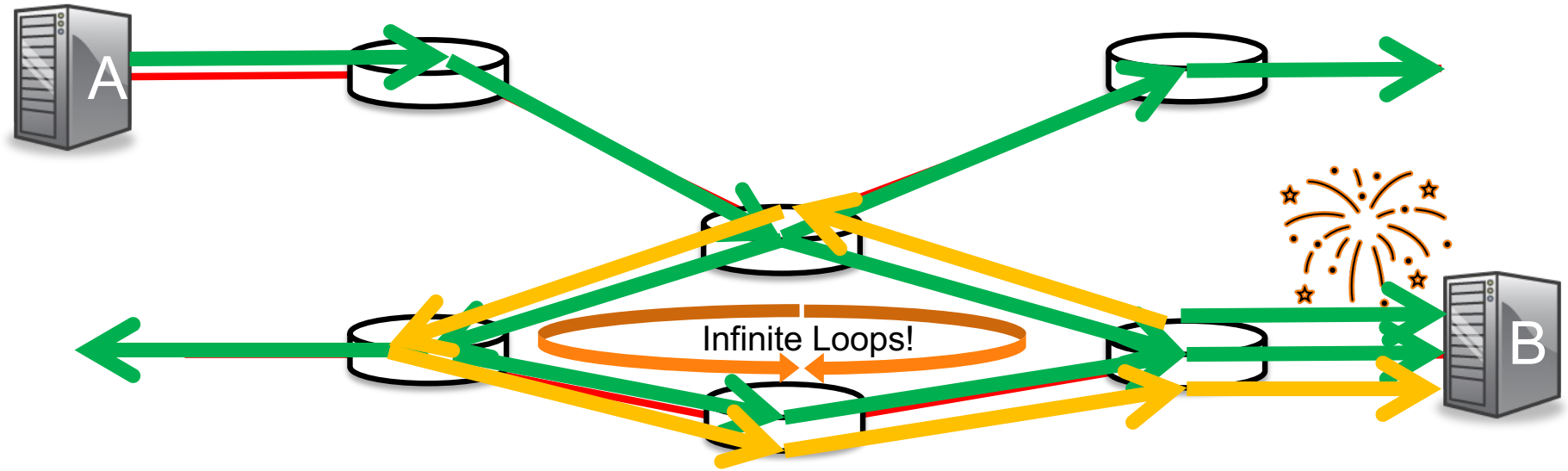
How does a router know
where to send a packet next?

Here are three ways

1. **Flooding**: Every router sends an arriving packet to every neighbor.
2. **Source Routing**: Source host adds, to every packet, a list of routers to visit along the way.
3. **Distributed Algorithm**: Routers talk to each other, then construct forwarding tables using a clever algorithm.

1. Flooding

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



Cons

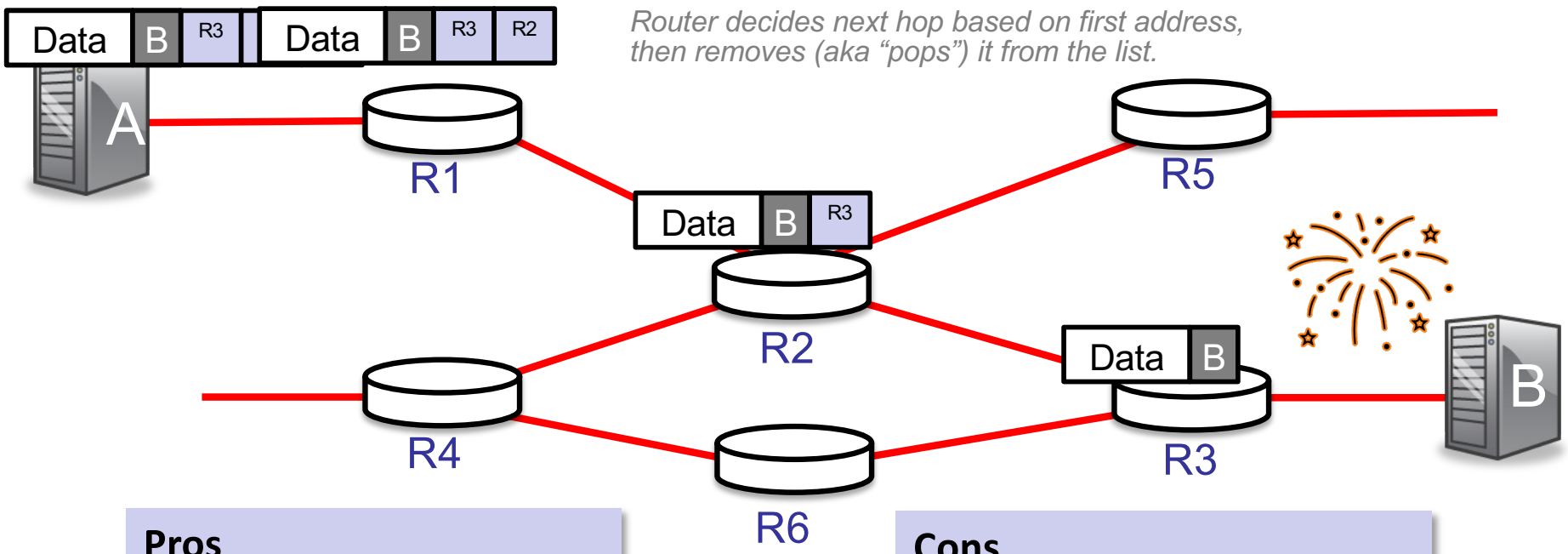
- Packets can loop forever (need TTL!)
- Inefficient use of the links
- Packets are delivered to everyone

Pros

- Packet reaches destination along shortest path
- Works when we don't know the topology (and gets used!)

2. Source Routing

Source adds a list of routers to every packet



Router decides next hop based on first address, then removes (aka "pops") it from the list.

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**: Source host adds, to every packet, a list of routers to visit along the way
3. **Distributed Algorithm**: Routers talk to each other, then construct forwarding tables using a clever algorithm.

The rest of today's class....

Basic operations of an Internet router

1. If the Ethernet DA of the arriving frame belongs to the router, **accept** the frame. Else **drop** it.
2. Check the IP version number and length of the datagram.
3. Decrement the TTL, update the IP header checksum.
4. Check to see if TTL == 0.
5. If the IP DA is in the forwarding table, forward to the correct egress port(s) for the next hop.
6. Find the Ethernet DA for the next hop router.
7. Create a new Ethernet frame and send it.

Routing tables

Example table:

Usually written as
123.66.44/16
i.e. a 16-bit prefix

Rule	Next hop IP address
IP DA = 127.43.57.99	56.99.32.16
IP DA = 123.66.44.X	22.45.21.126
IP DA = 76.9.X.X	56.99.32.16
…	…

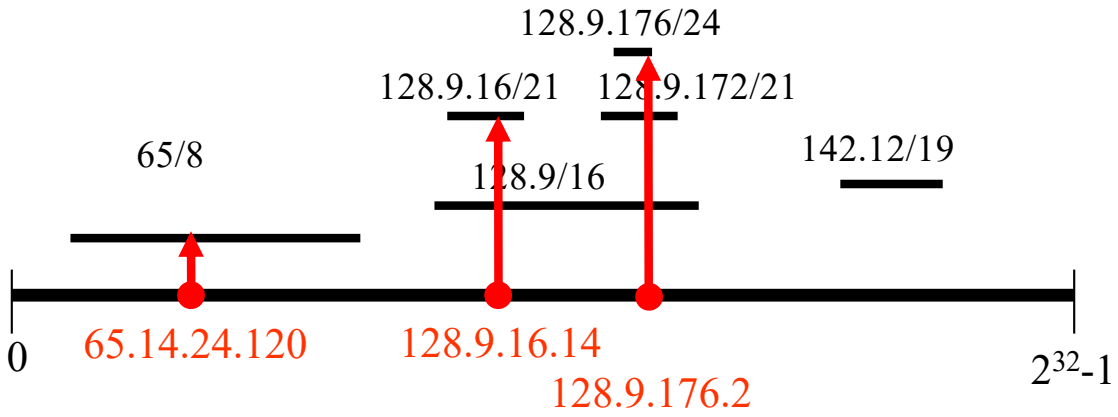
All IP addresses with the same prefix are forwarded to the same next-hop

The lookup matches on the longest matching prefix

Q: Why?

Longest prefix match

Example



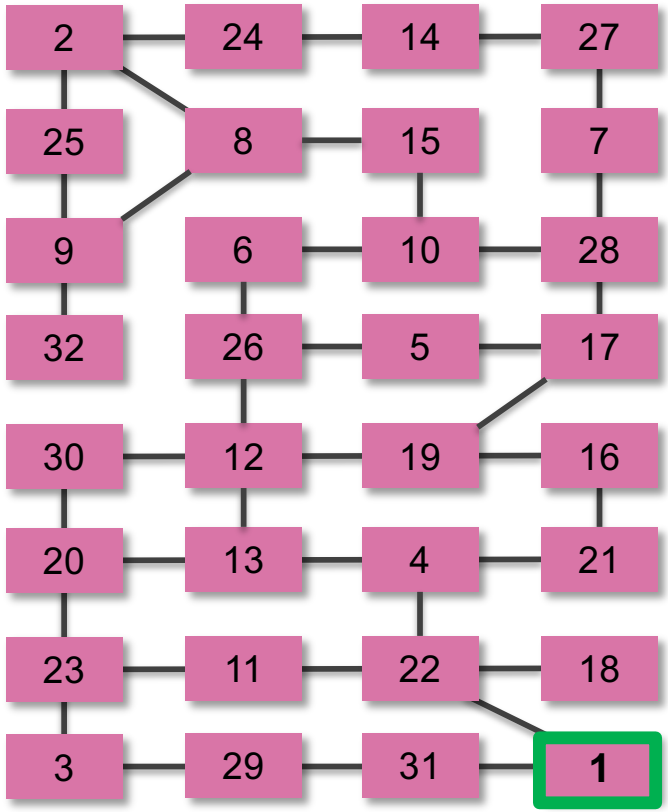
Routing lookup: Find the longest matching prefix (*aka* **the most specific route**) among all prefixes that match the destination address.

How does the table get constructed?

A thought experiment…

The network below consists of 32 routers, each with a unique ID.
Packets from any source router should be delivered to destination router 1, exactly once, along the shortest path.

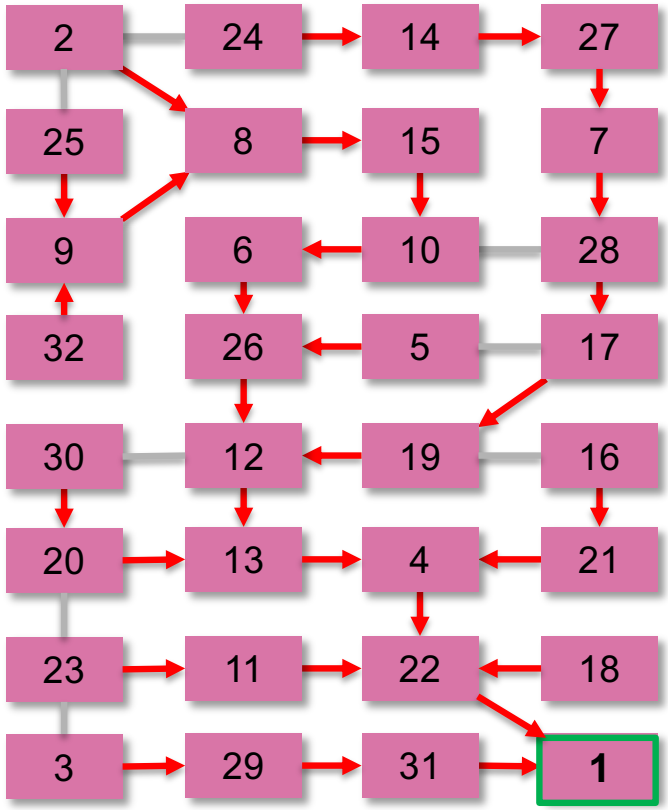
***Q**: Which hop should each router forward packets to next?*

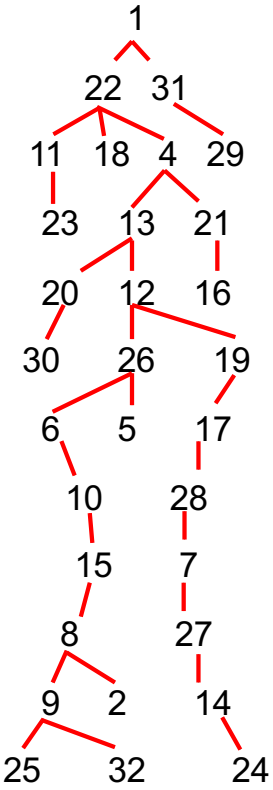


The network below consists of 32 routers, each with a unique ID.
Packets from any source router should be delivered to destination router 1, exactly once, along the shortest path.

***A**: Follow the shortest path* **spanning tree**, *rooted at router 1*

Why use a spanning tree?
1. Includes every node (spanning)
2. Loop-free (tree)





Shortest path spanning tree

Observations

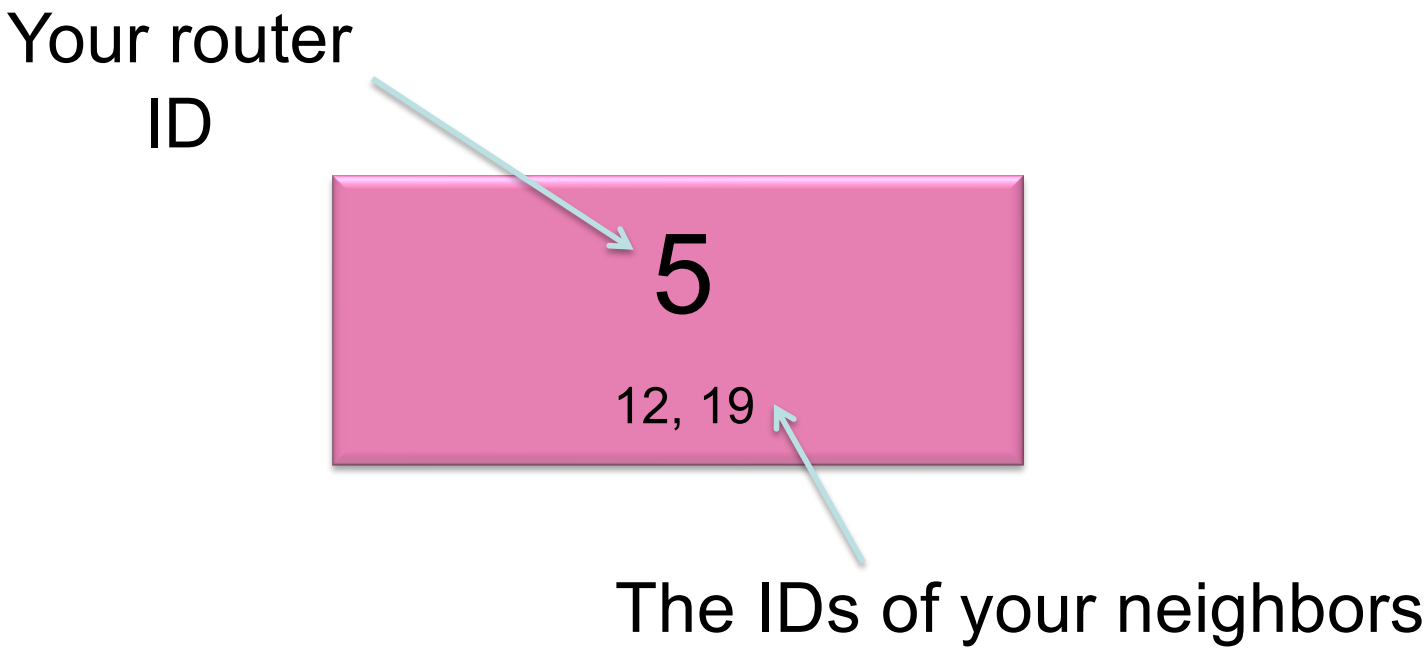
Routers work together, to build a spanning tree ***for each destination***.

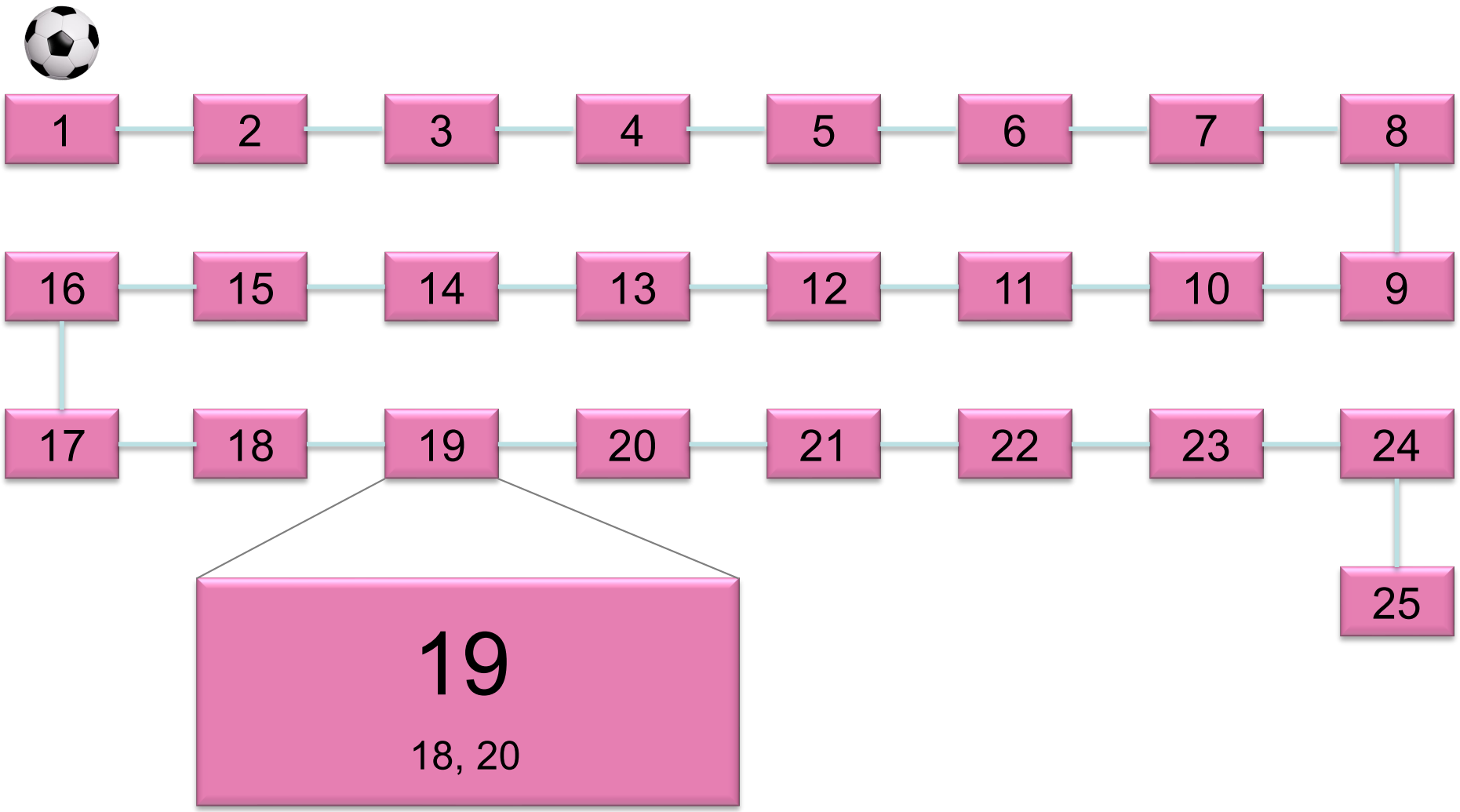
A routing table entry tells the router, for each destination, which hop to send the packet to next, so that the packet follows the spanning tree.

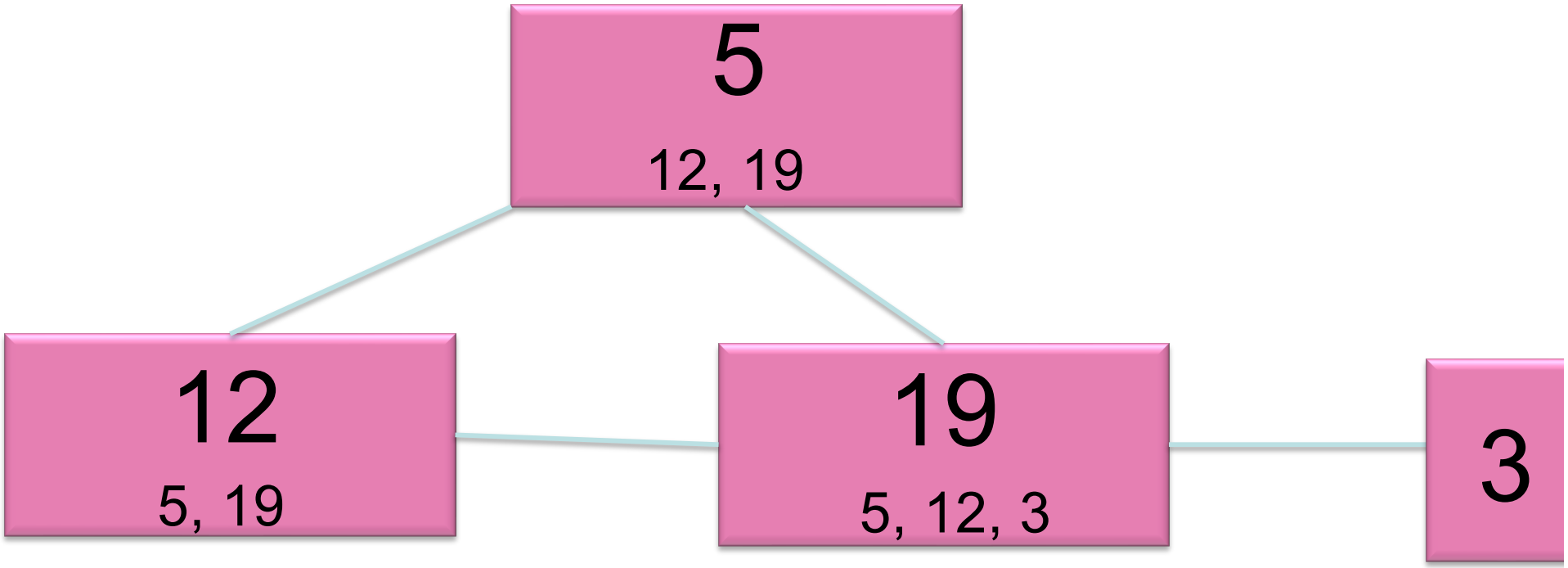
Q: *How does a router know what entries to add in its routing table?*

Game: Routing Competition

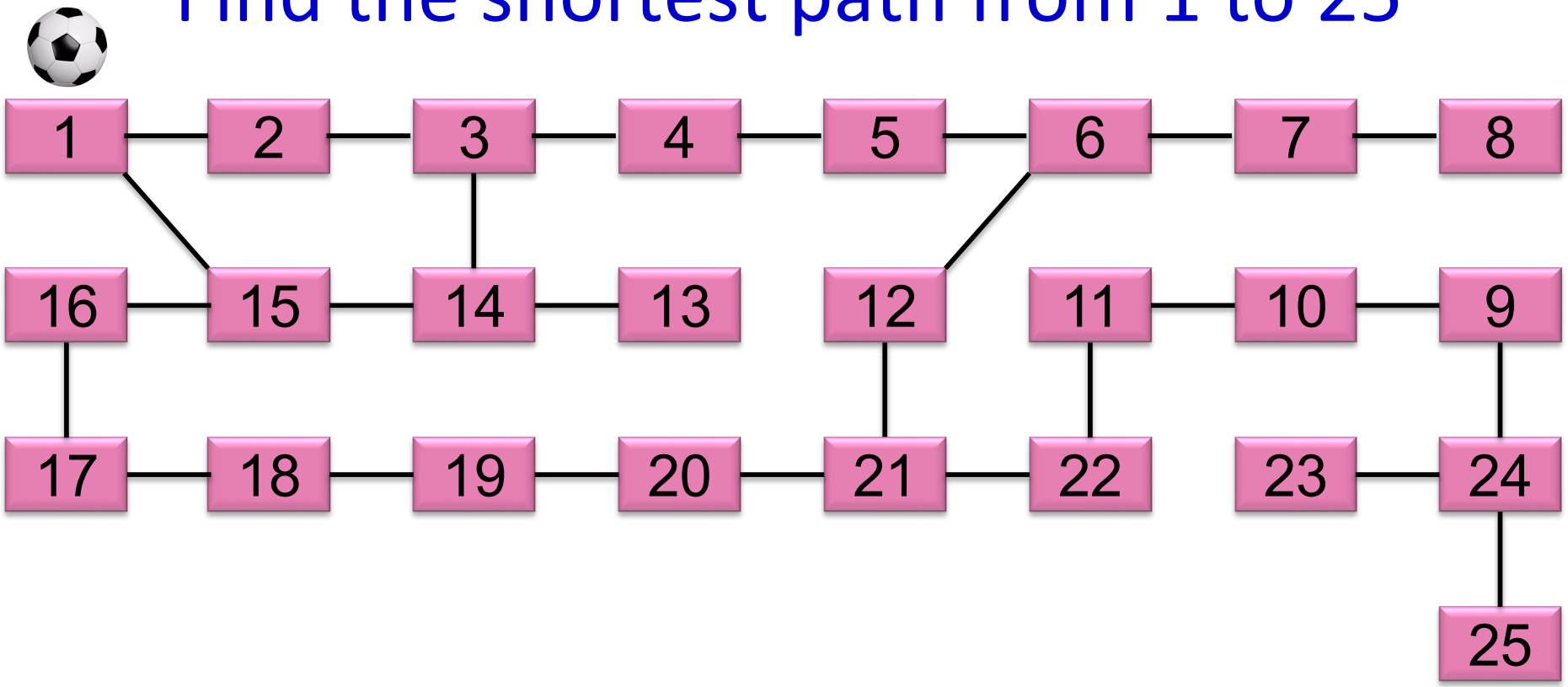
Each team member has a card




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 from 1 to 25



Observation

It’s straightforward when we know the topology

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

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

But you still need to find the shortest path.

Rules

You may <u>not</u>

1. Pass your card to anyone else
2. Leave your seat
3. Write anything down

You may

1. Ask nearby friends (in your team) for advice
2. Shout to other participants in your team (anything!)
3. Say bad things about Nick

Pink Group

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

Task

Find the shortest path from
Node 1 to Node 40.

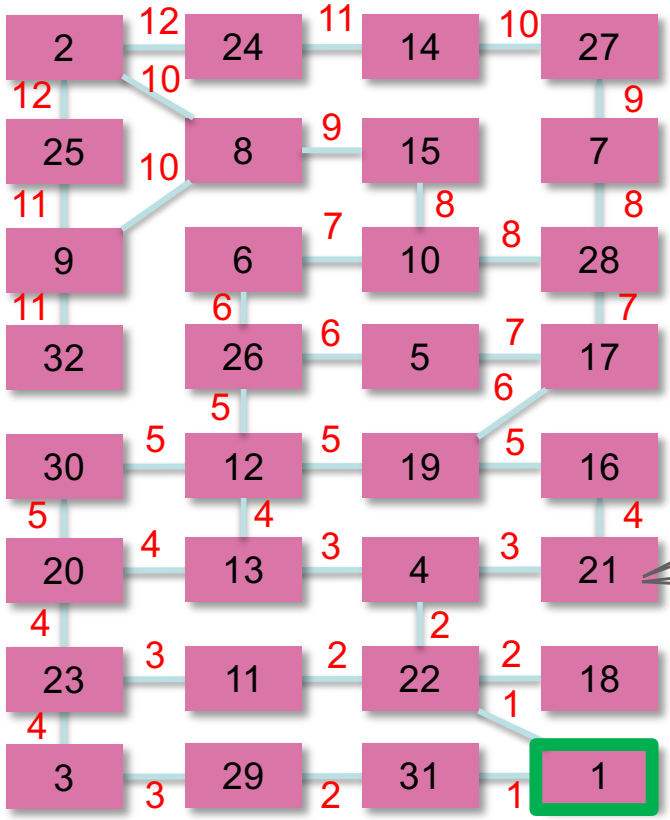
When you are done, you must be able
to repeat it correctly.

The first group to finish is the champion!!

How did your team solve it?

Bellman-Ford: A distributed algorithm to find the shortest path spanning tree when you don’t know the topology.

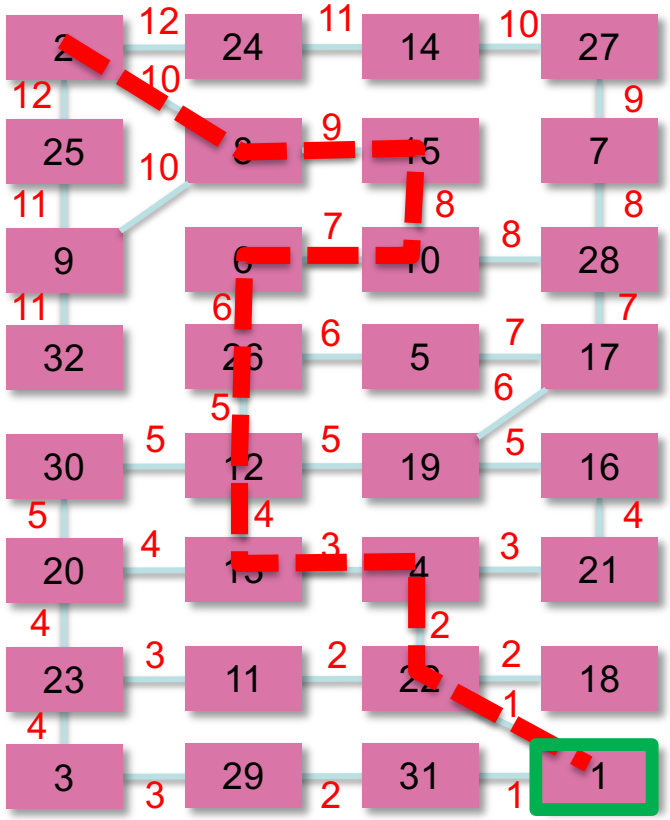
Find the shortest path spanning tree rooted at router 1



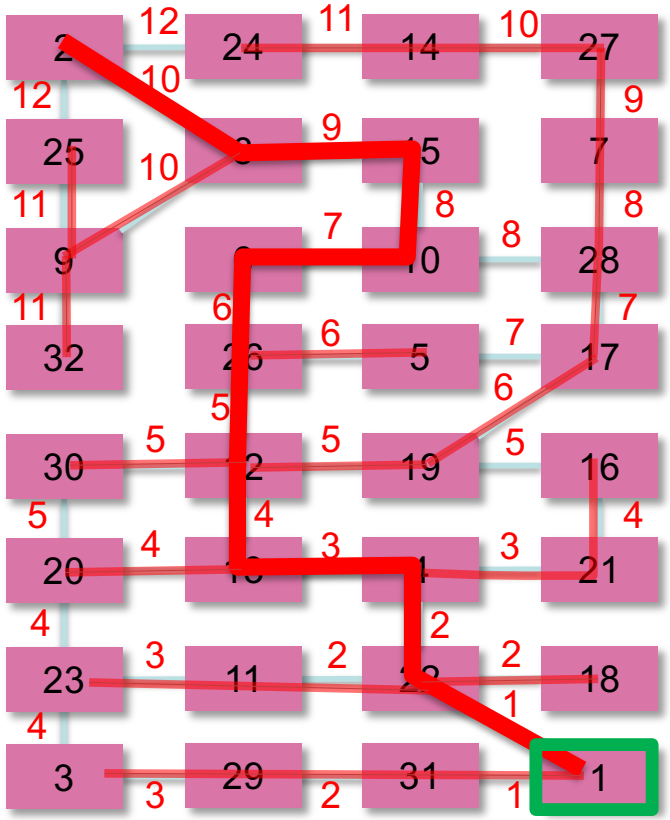
"You can reach router 1 in four hops via me"

Destination	Next-hop
Router 1	Router 4

This is the shortest path from router 2 to 1



The shortest path spanning tree rooted at router 1



Observation

When complete, every router has a table entry to reach router 1.

The algorithm can run simultaneously to build a spanning tree (and hence create routing table entries) to reach every router.

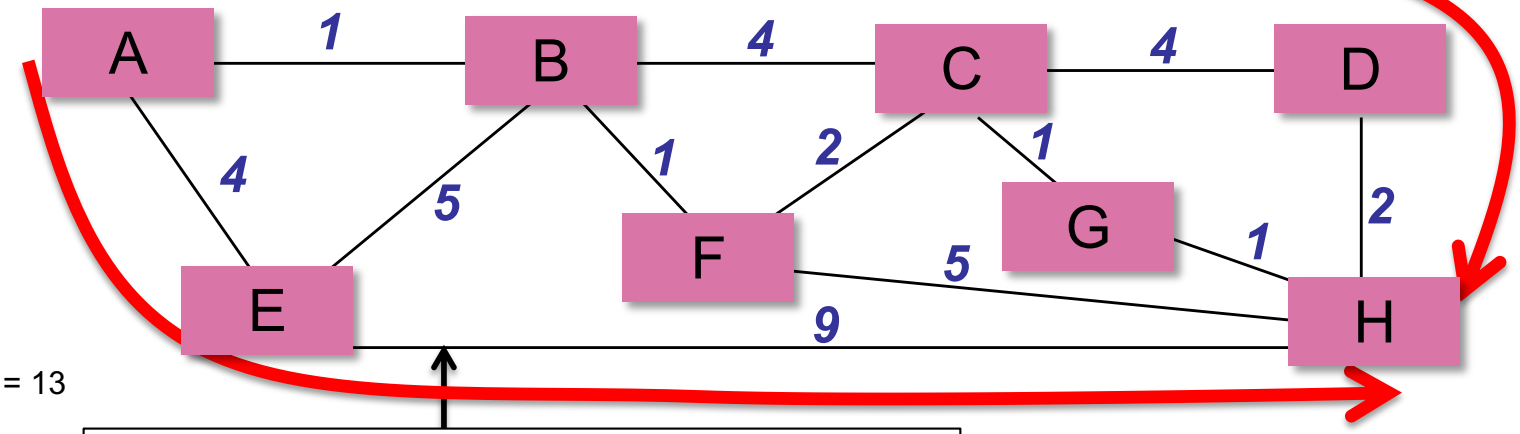
Bellman-Ford Algorithm

Questions:

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

What if each link has a “cost”?

Cost = 1+4+4+2 = 11

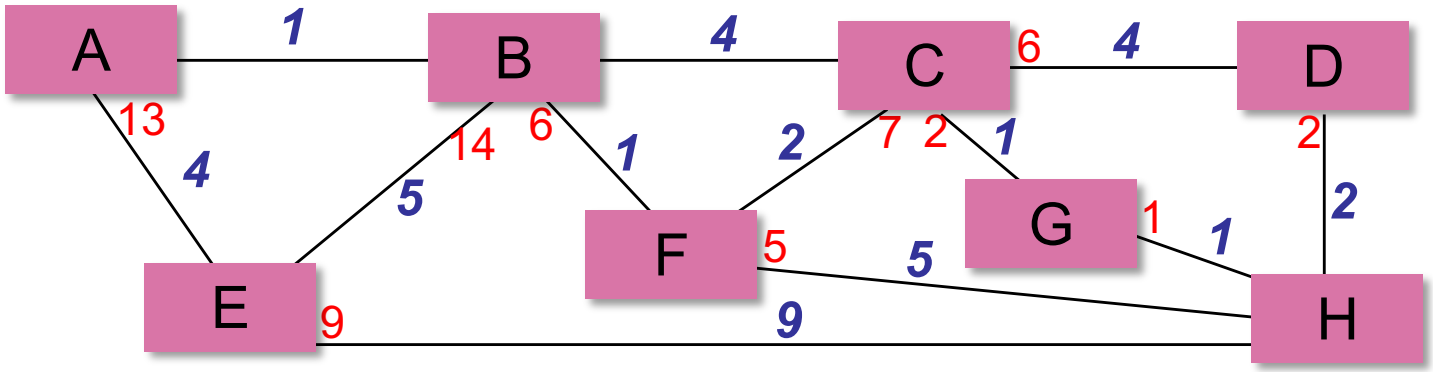


Cost = 4+9 = 13

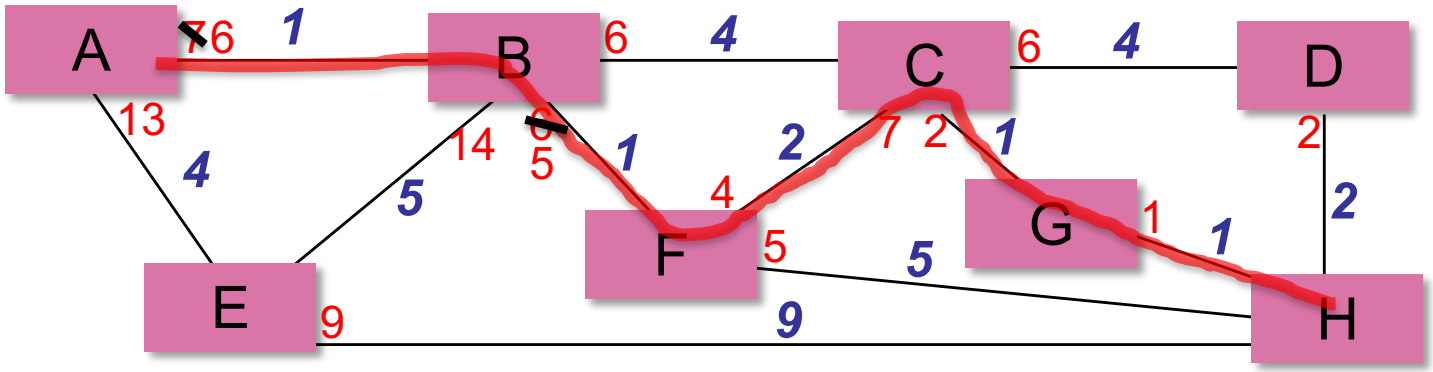
“Expensive link”:

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.

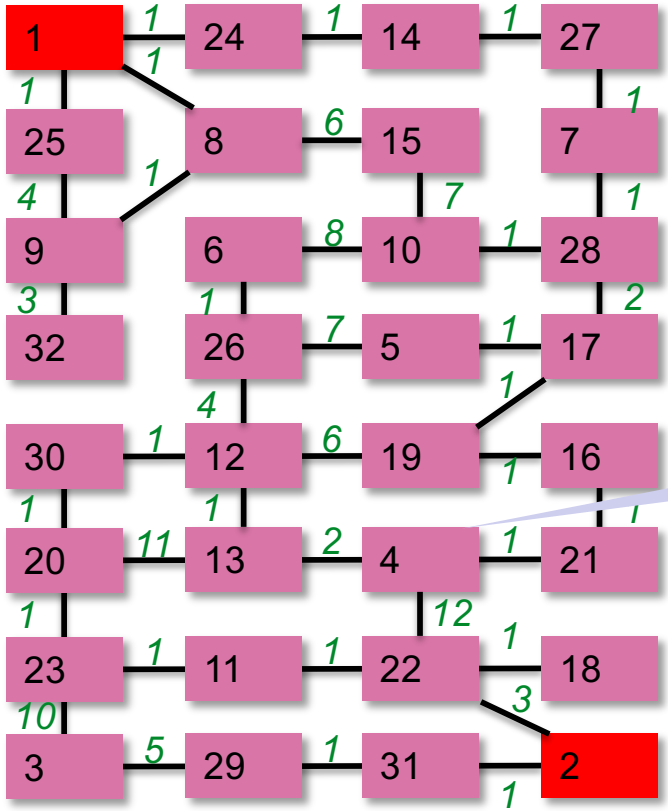
Find lowest cost path to H



Find lowest cost path to H

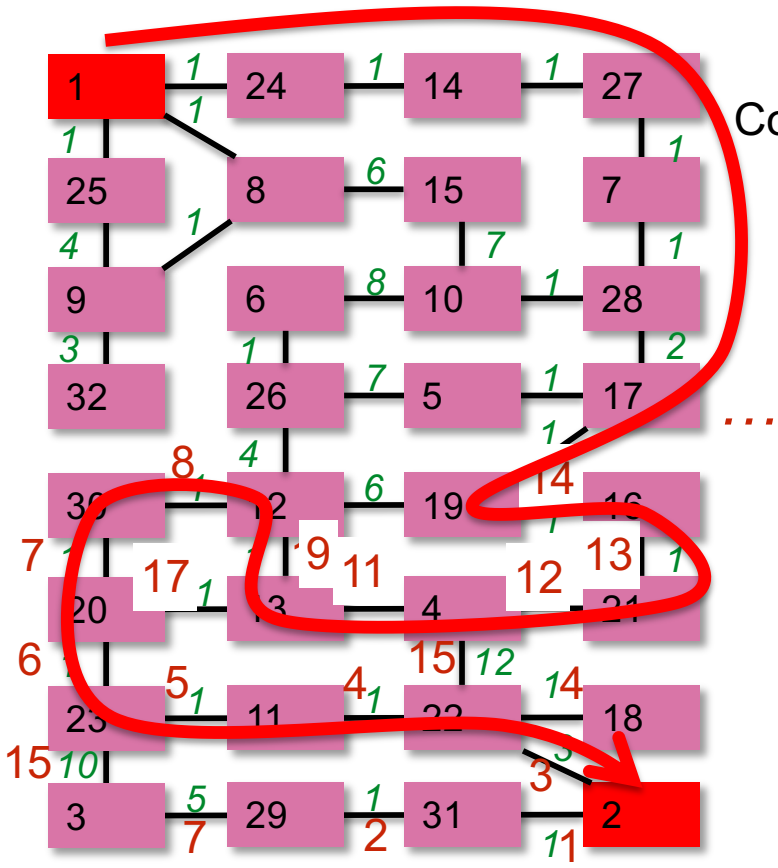


Find the lowest cost path



Router 4 tells its neighbors:
"I can reach 2 with a cost of 15"

Solution



Cost = 22

…and so on!

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 $\underline{\mathbf{C}}=(C_1, C_2,\ldots)$ is the *distance vector* to **R**.
- Initially, set $\underline{\mathbf{C}} = (\infty, \infty, \ldots \infty)$
  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.

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?