Ethernet: Bridging

Based on Radia Perlman's
Interconnections

Administravia

- Project 1 deadline: pushed back to Friday, October 24 since the TCP server wasn’t running until yesterday
  - Don’t depend on last-minute extensions!

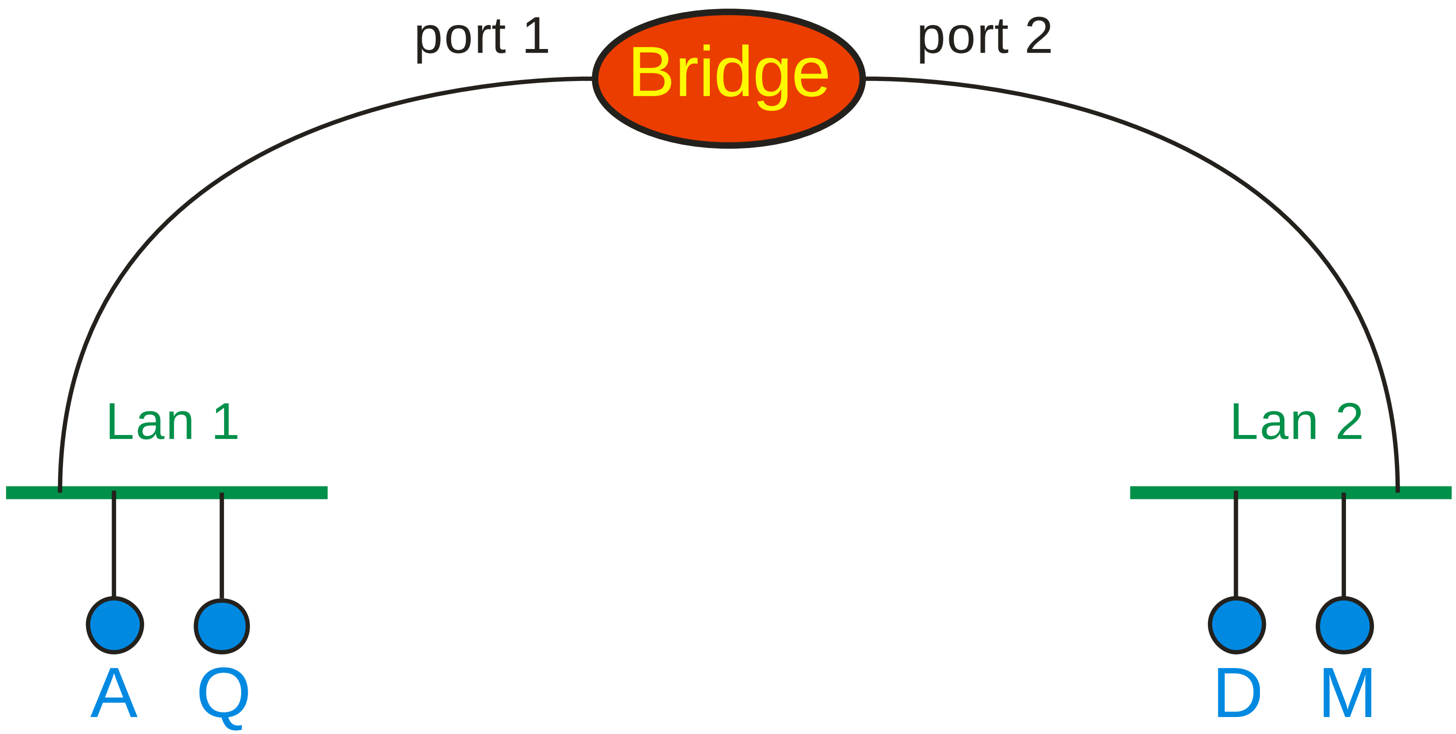
- Project 2 “stage 1” also due Friday, October 24

- P2S1™ is just to get “hello world” running on a router

When LANs Grow

- We learned one LAN can be limited in
  - Distance
    - Why? Do repeaters help?
  - Number of nodes
  - Performance
- How do we scale?
  - Interconnect LANs to each other
  - Store & forward eliminates most problems
- Can we build a transparent “bridge”

The No-Frills Bridge



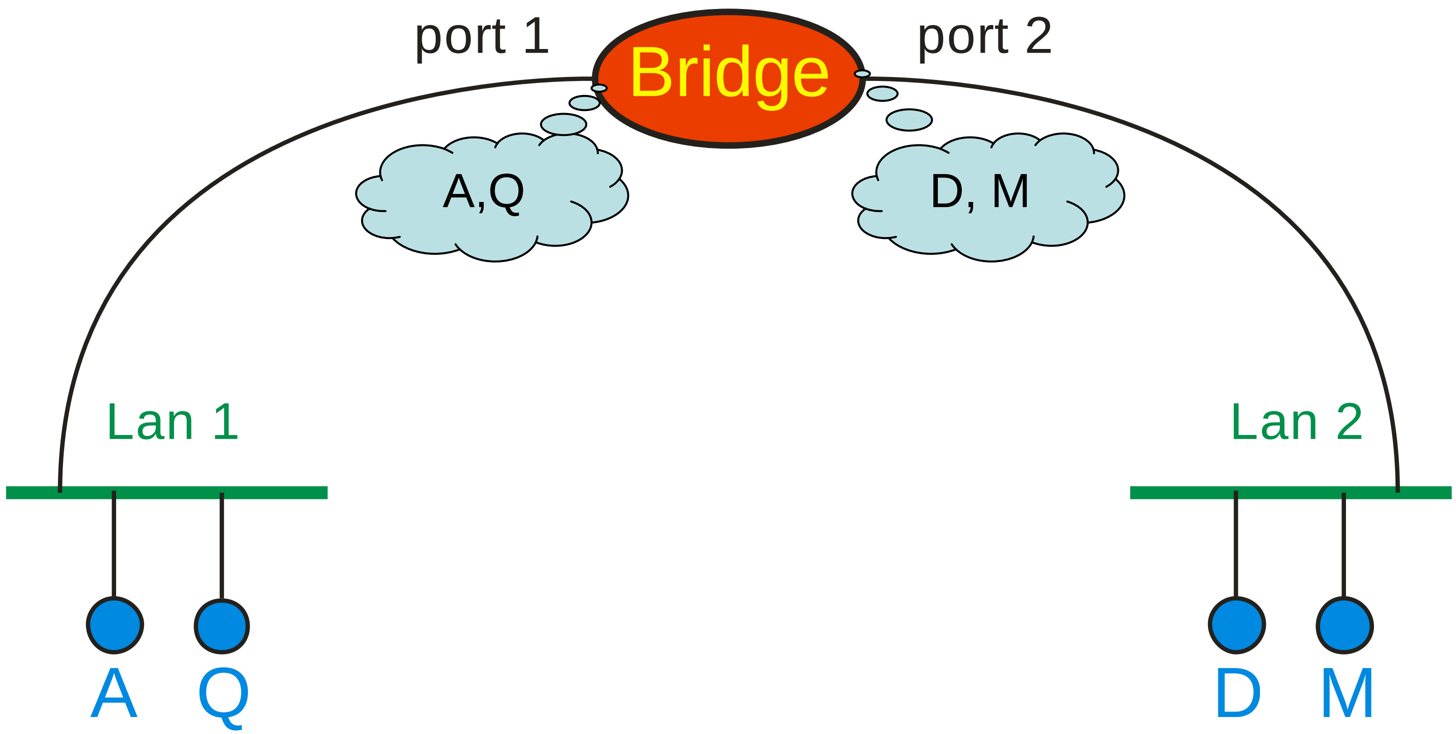
Bridge receives complete frames from one side, waits for the other side to be free, and re-transmits the exact same frame.

LAN is now two collision domains, it’s still one broadcast domain

Maximum speed is unchanged – why?

“Mostly” transparent (except for timing, loss, max packet lifetime…)

The Slightly Smarter Bridge



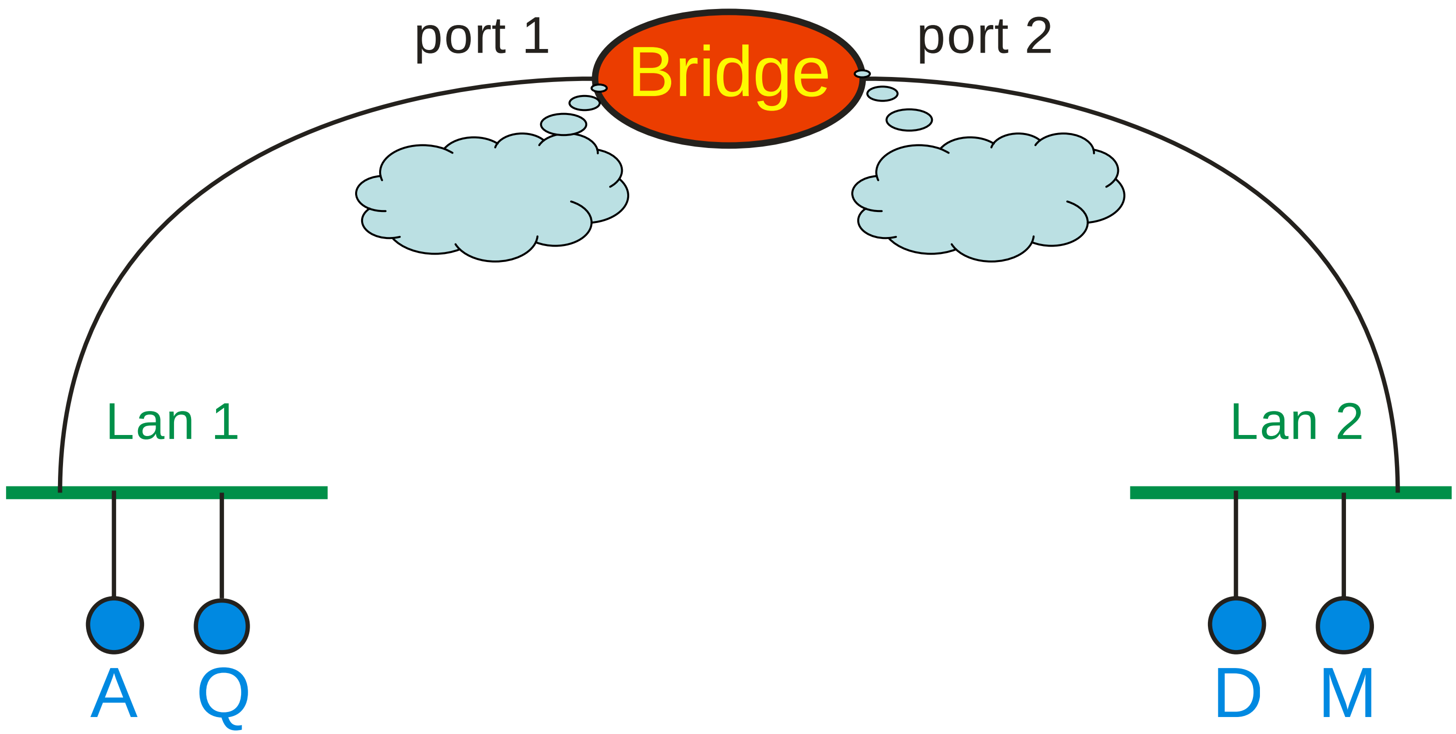
Can we improve this?

What if the bridge knew which nodes were on which side so that, for example, packets from A → Q are not transmitted to LAN 2?

Improves maximum data rate (why?)

One possibility is to manually configure a list of nodes on each side, but that's not very robust. Can it *learn?*

The Learning Bridge

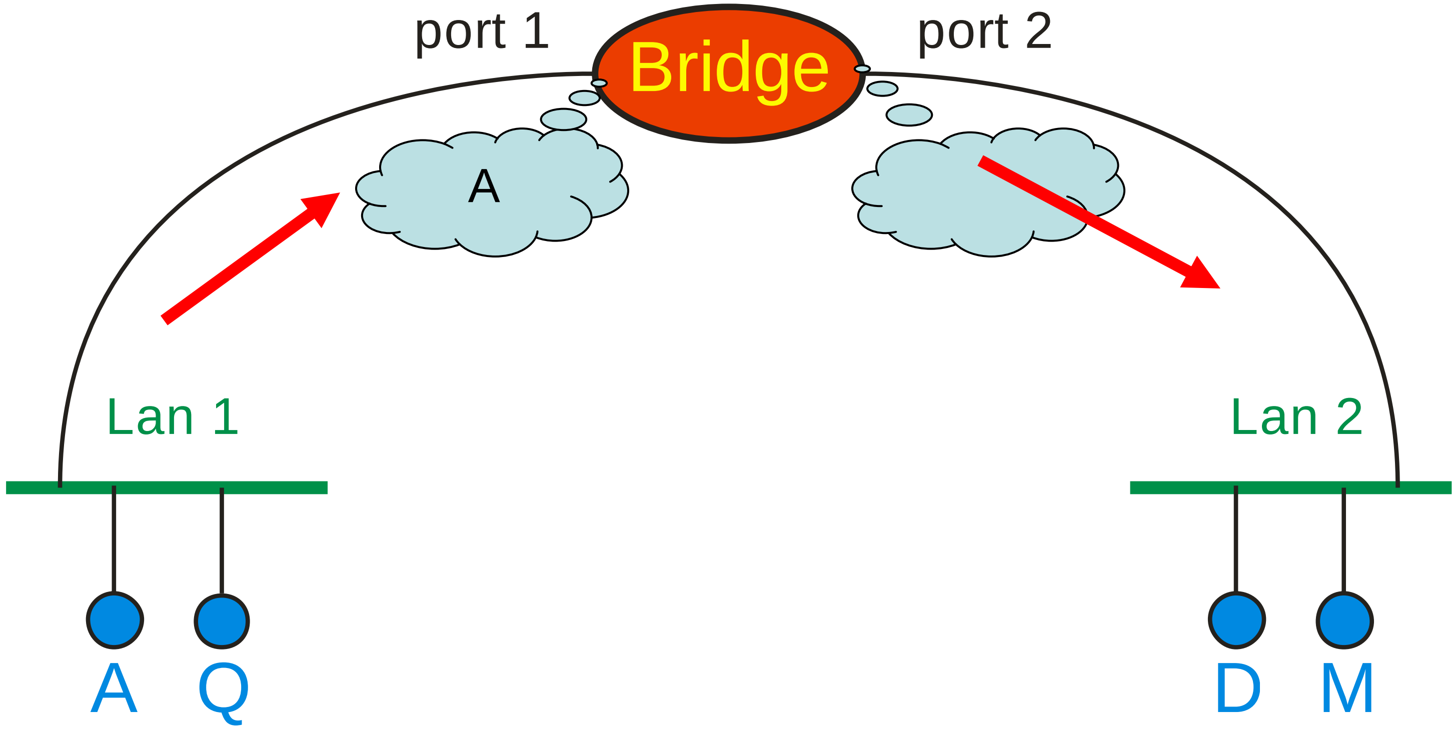


Station caches start out empty

Every time a packet is transmitted, add the source to that port's station cache

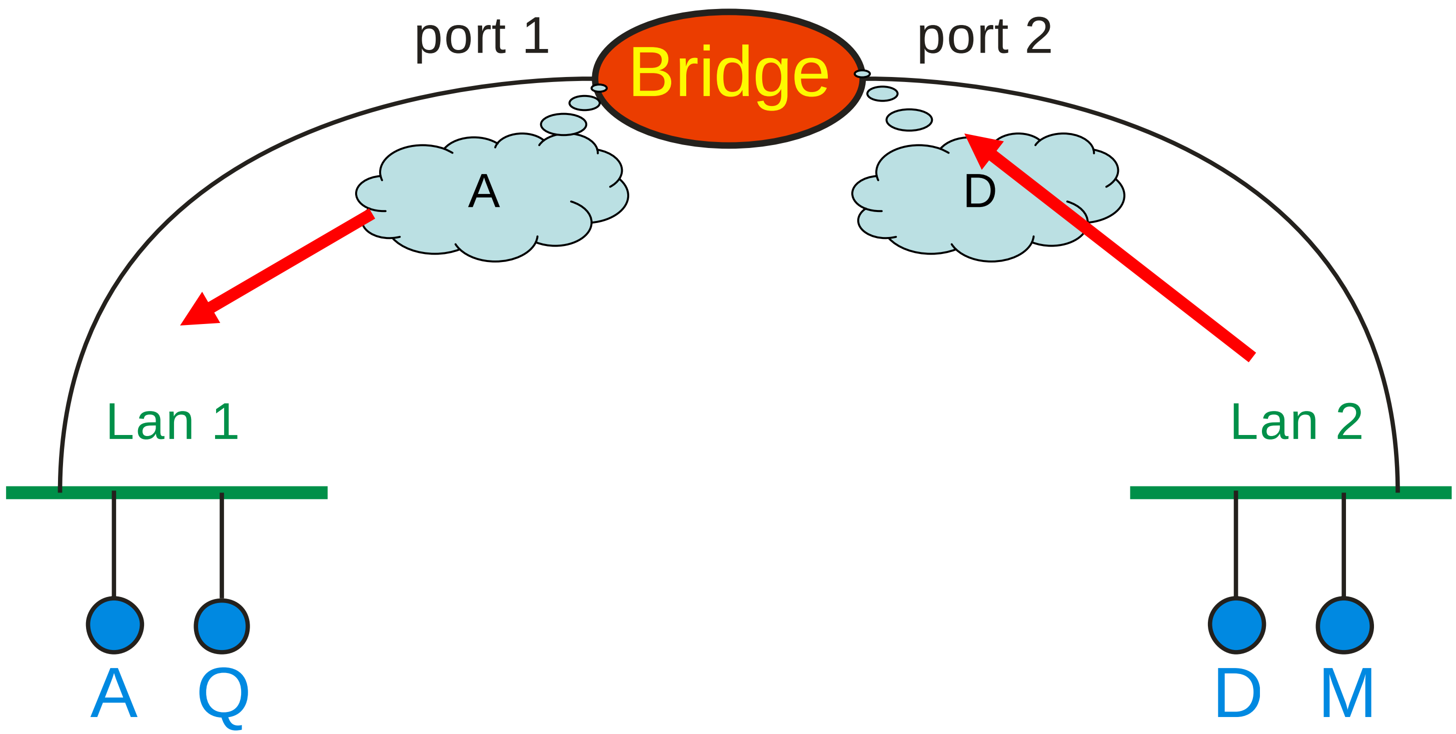
If the destination is in a station cache, transmit only to that port.
Otherwise, transmit to all ports other than the source port.

The Learning Bridge



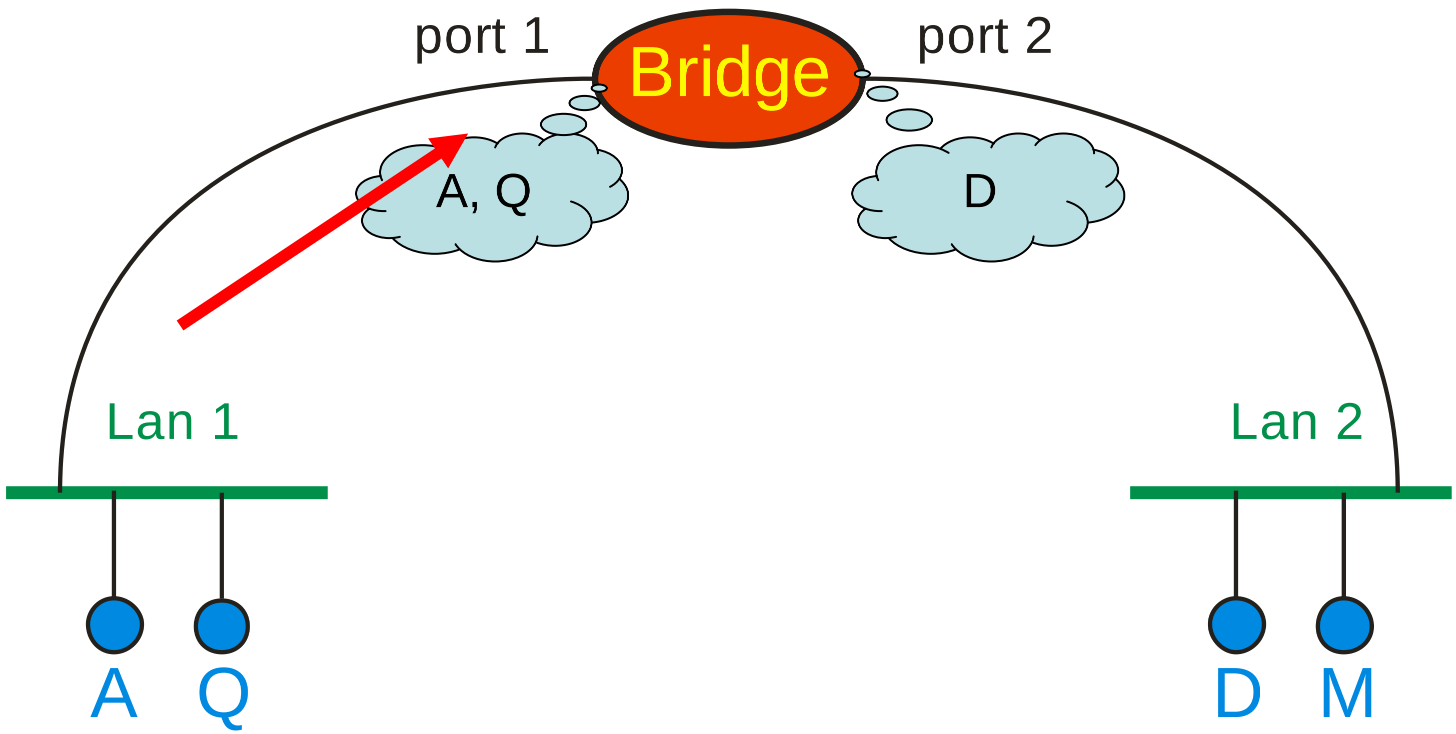
A transmits a frame to D

The Learning Bridge



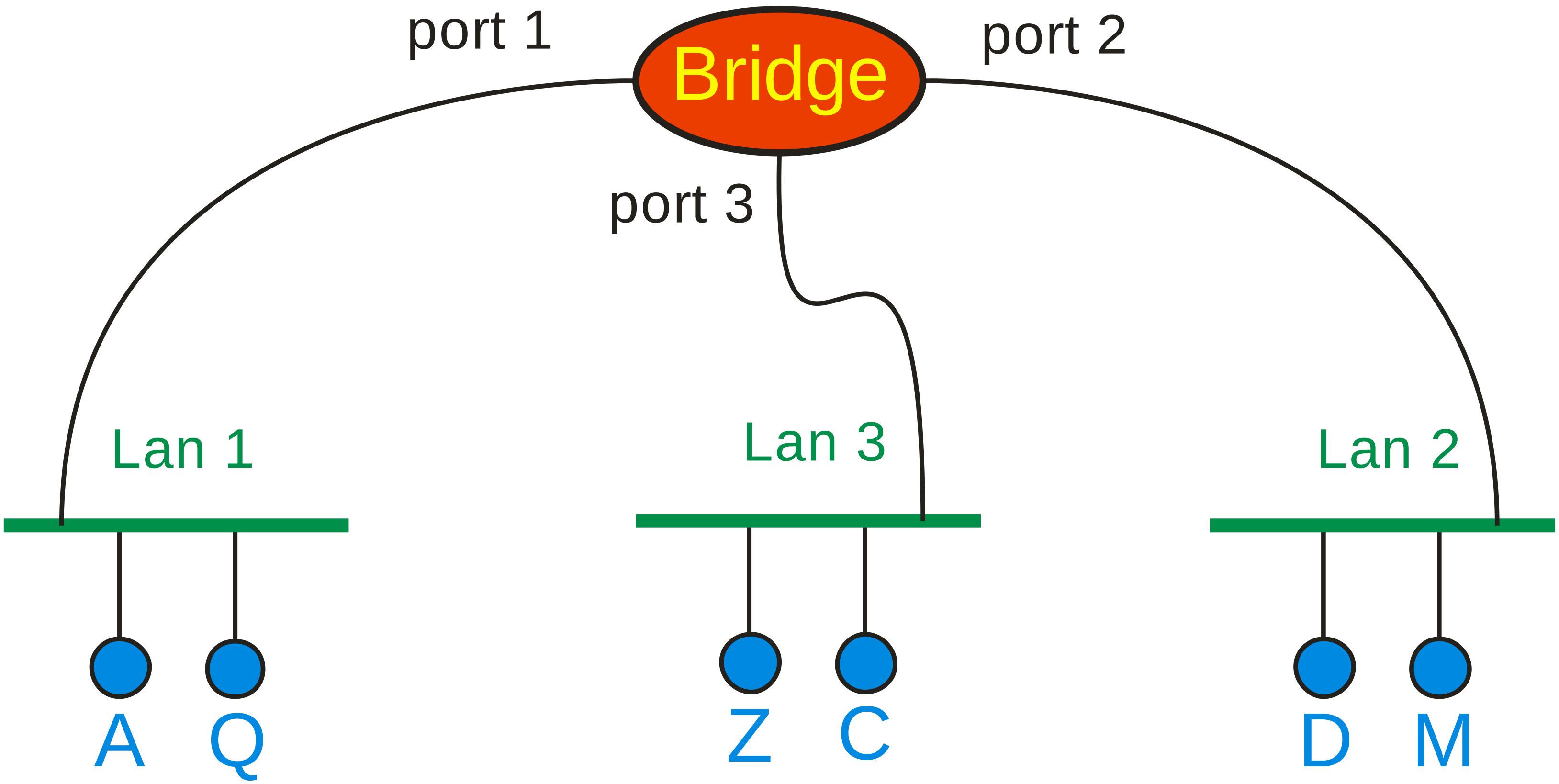
D transmits a frame to A

The Learning Bridge



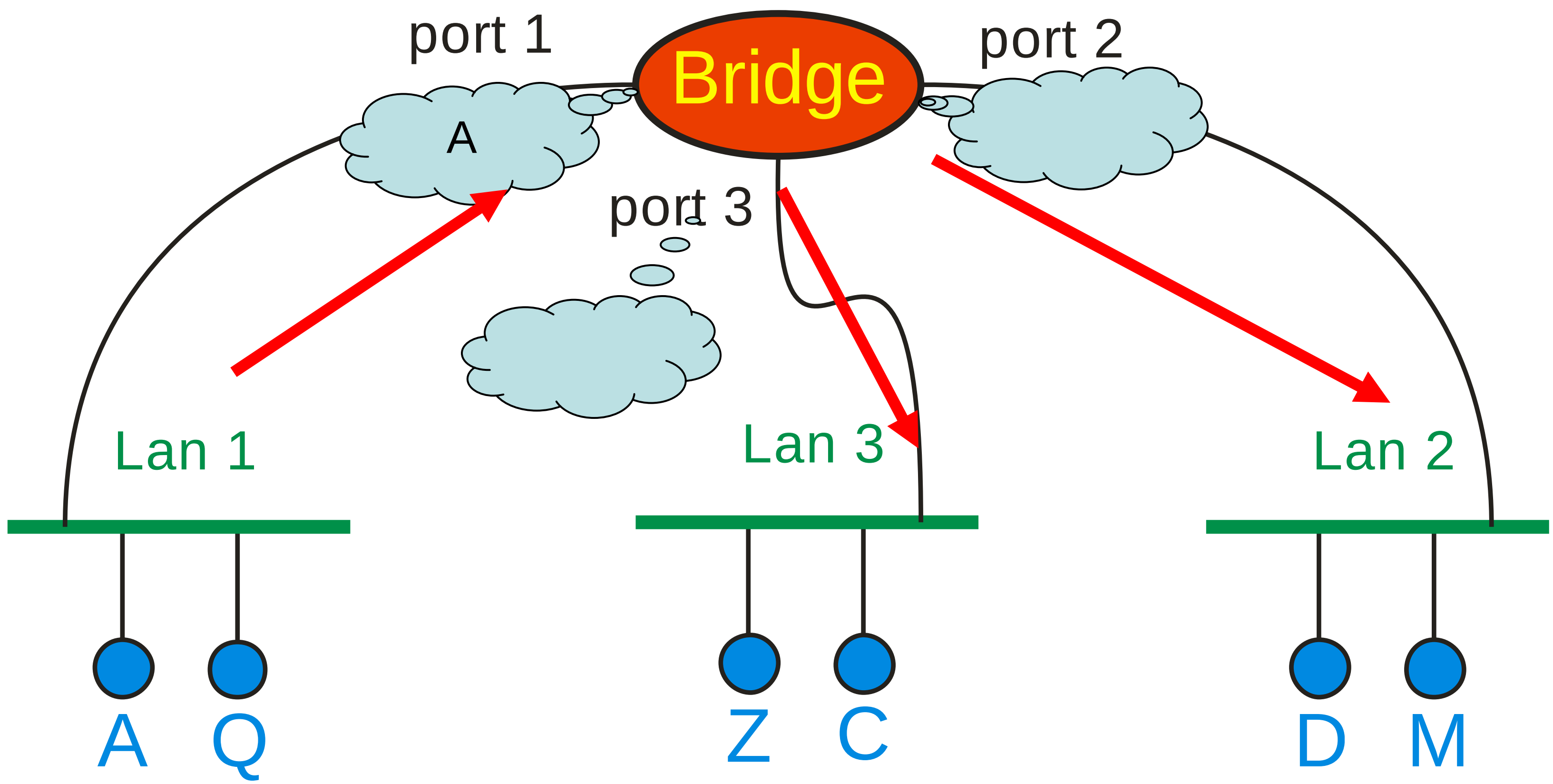
Q transmits a frame to A

Bridges work for *n* LANs, not just 2



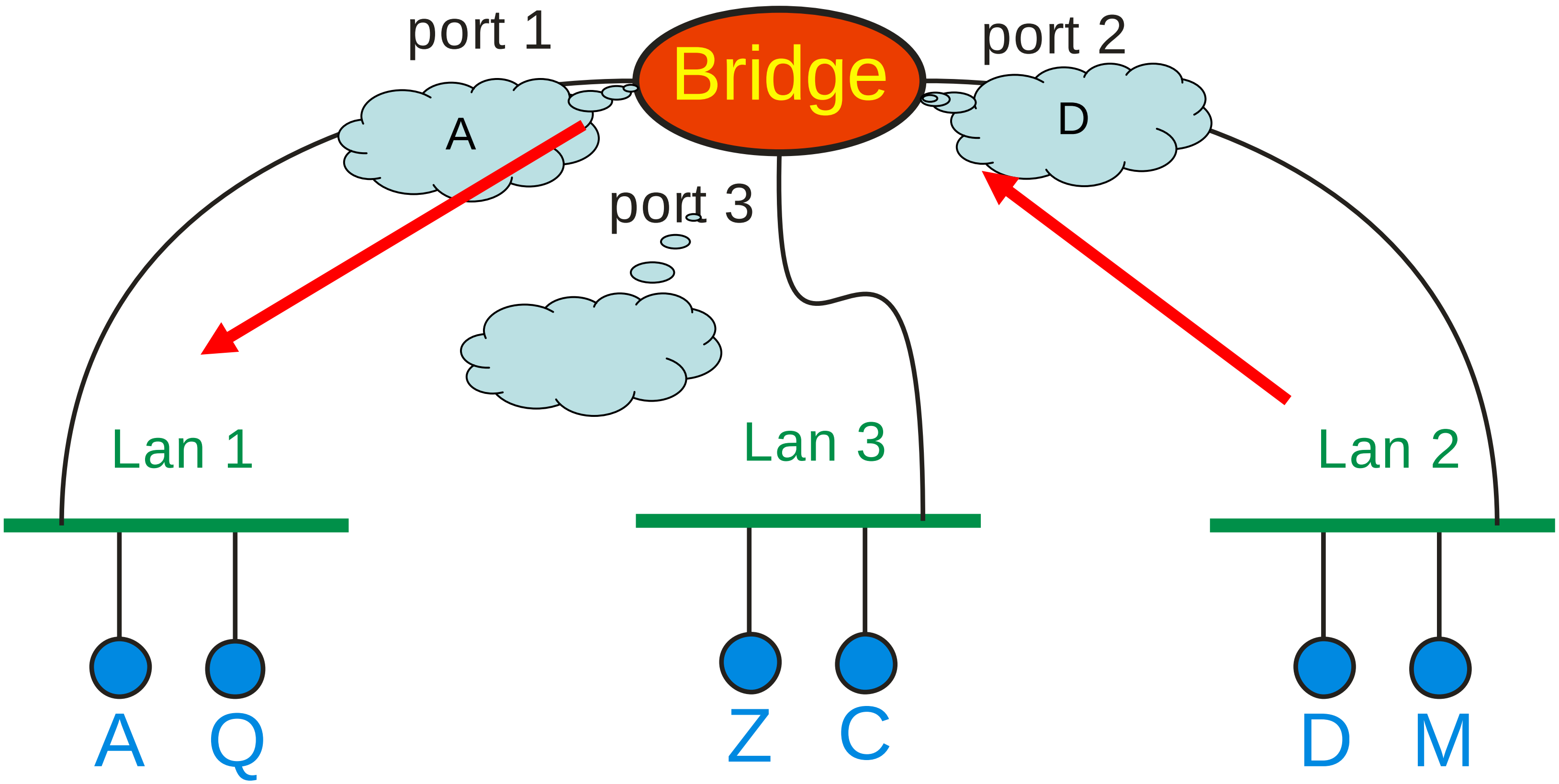
Frames to unknown destinations get forwarded to
all non-source ports

Bridges work for *n* LANs, not just 2



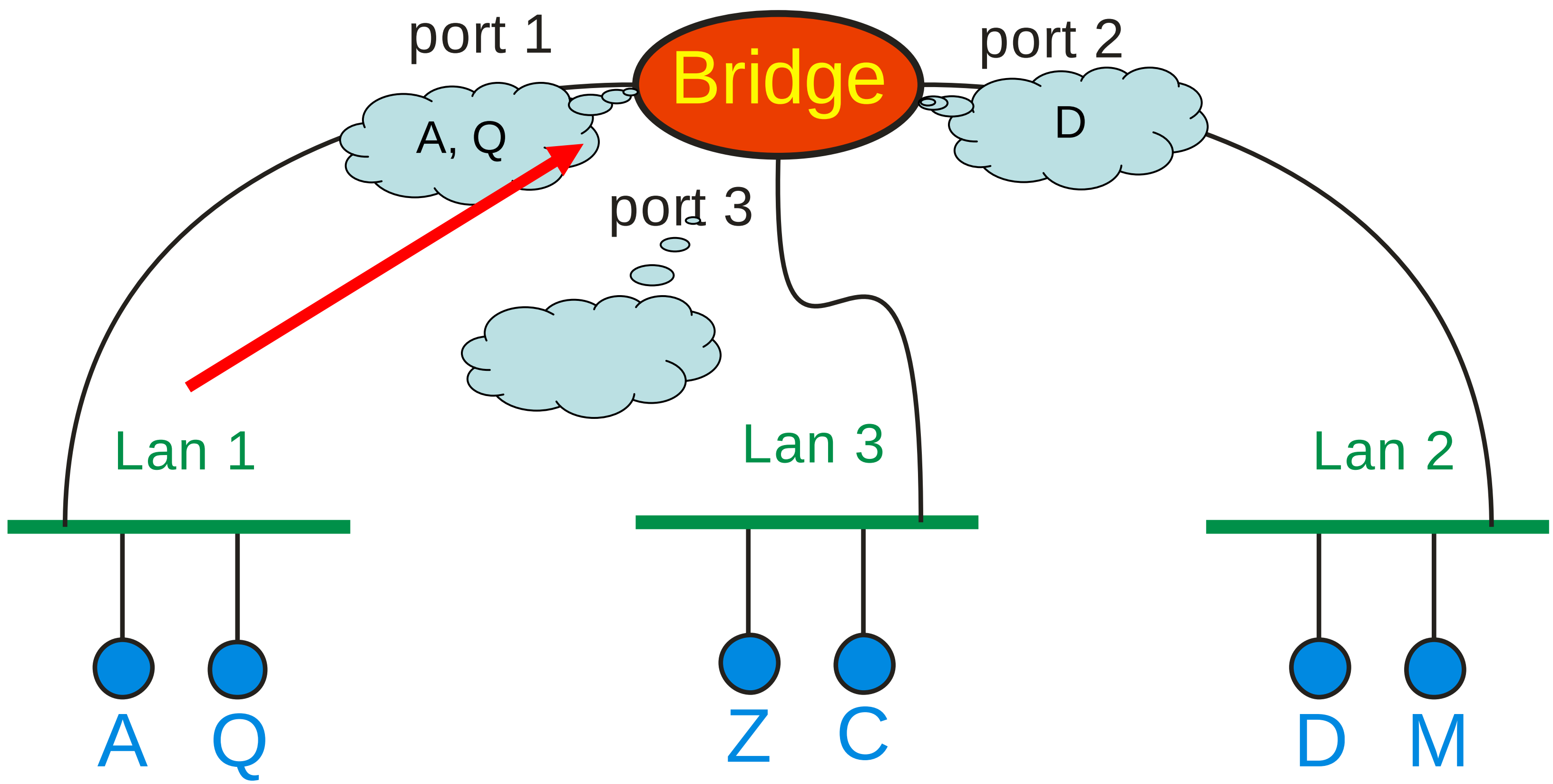
A transmits a frame to D

Bridges work for *n* LANs, not just 2



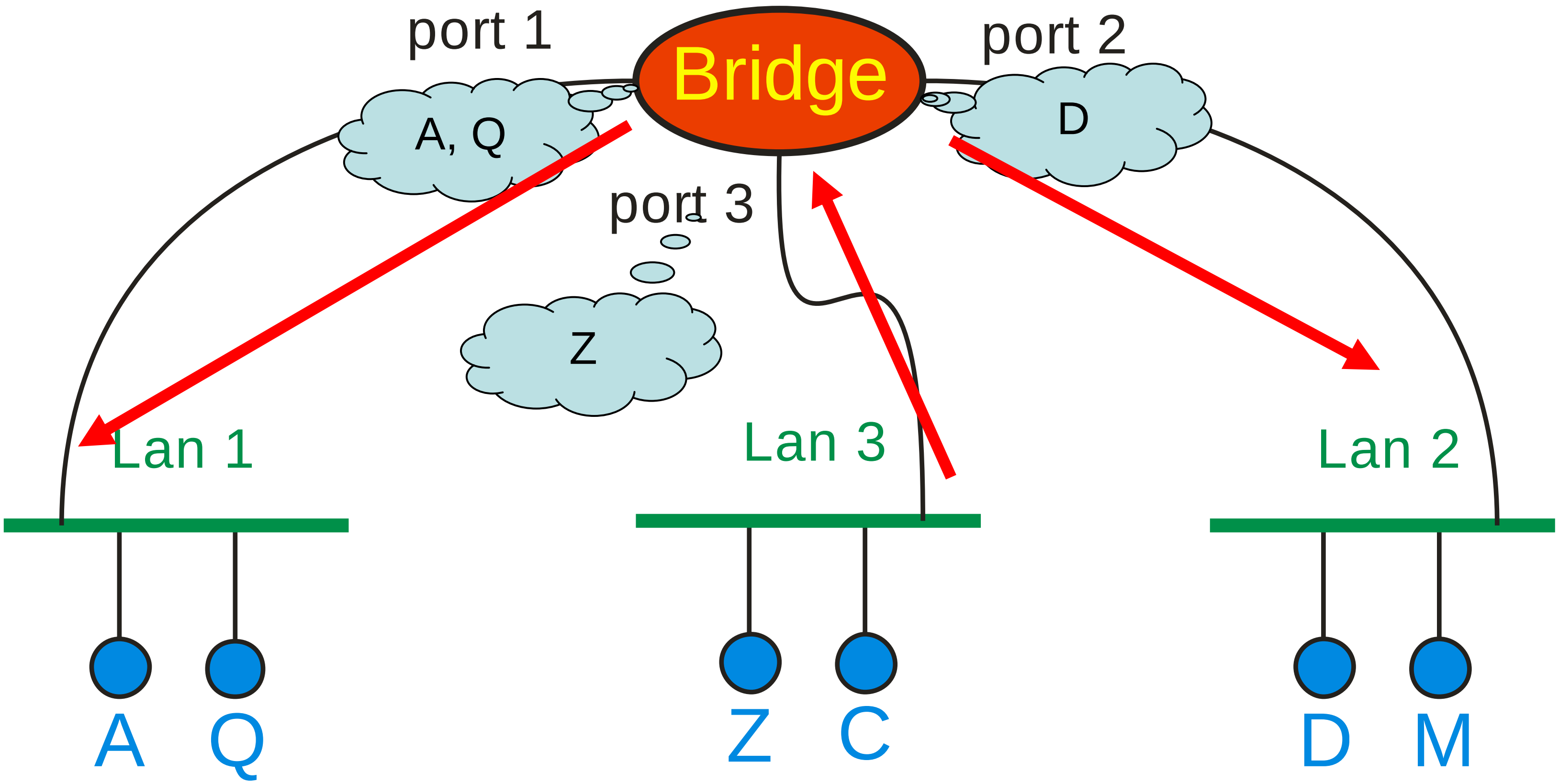
D transmits a frame to A

Bridges work for *n* LANs, not just 2



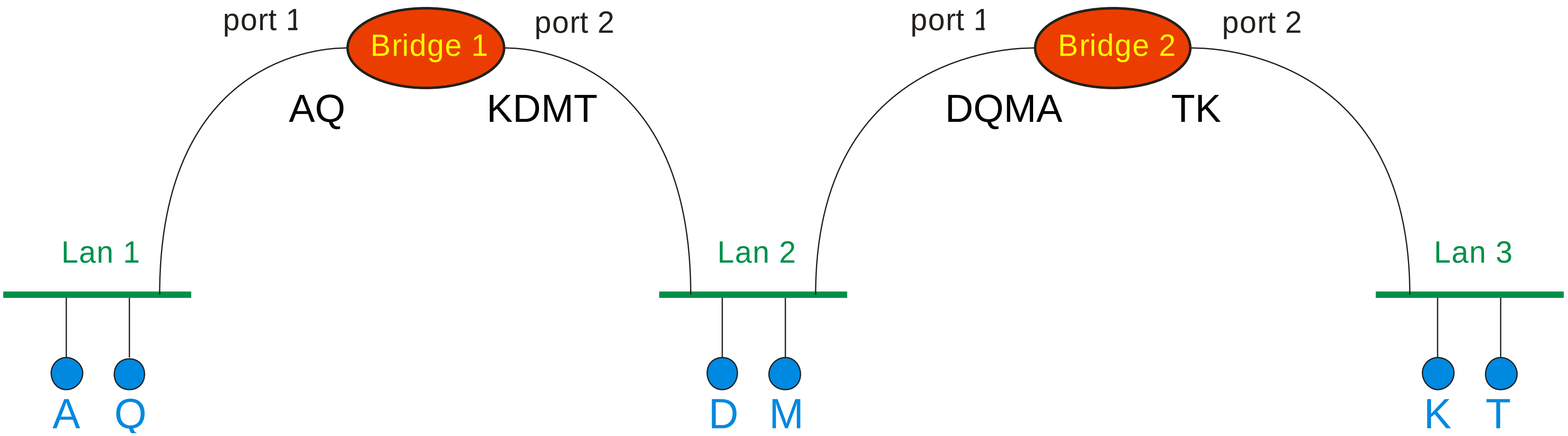
Q transmits a frame to A

Bridges work for *n* LANs, not just 2



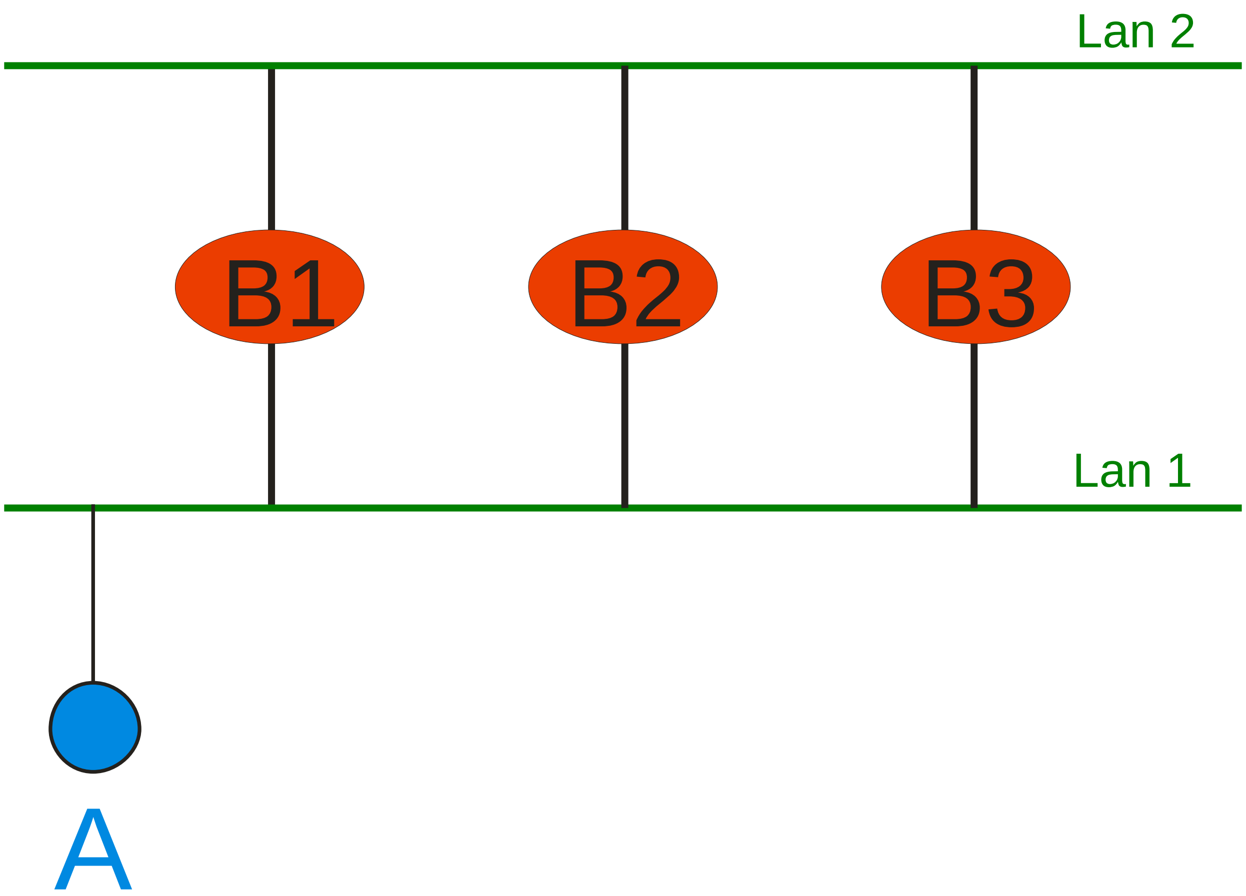
Z transmits a frame to C

Bridges can be chained together



Note Bridge 1 can not distinguish between LAN 2 and LAN 3.
Bridges are transparent – even to other bridges!

Is redundancy good or bad?

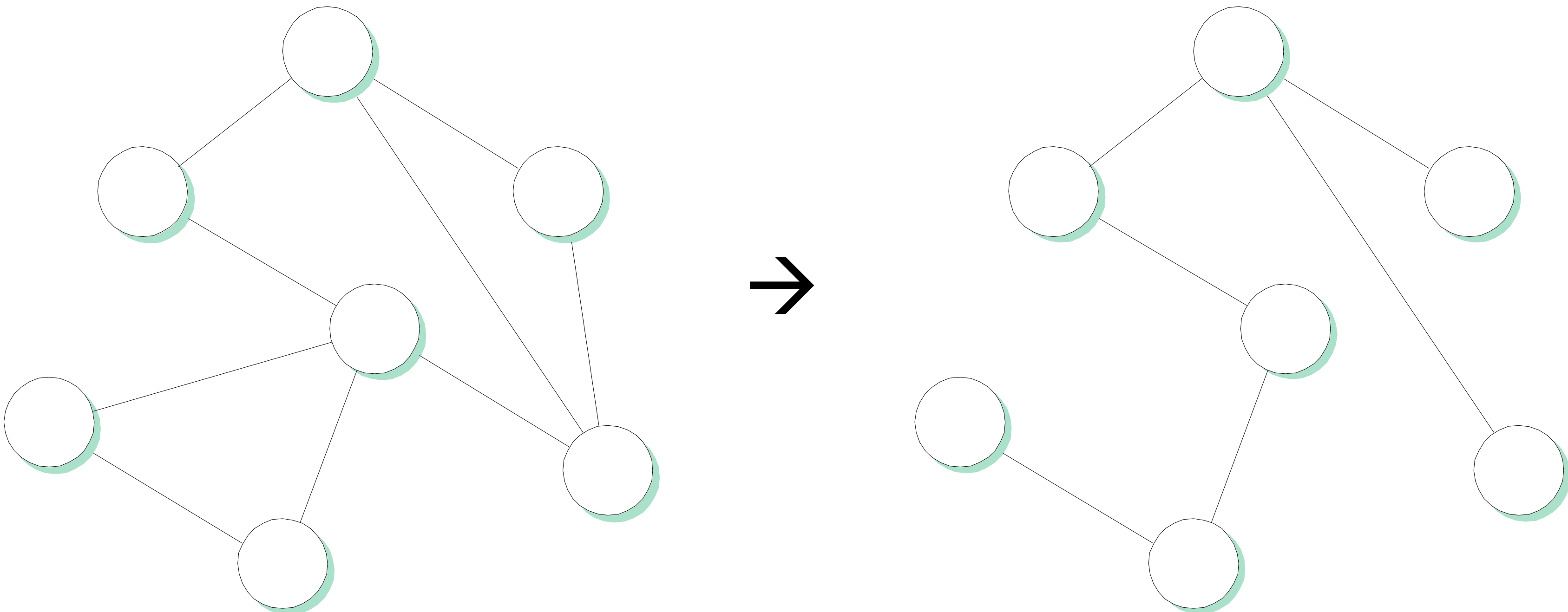


What happens when A transmits a frame – to anyone?
Packets don't just loop forever, they proliferate.
This is Bad™.

How do we handle loops?

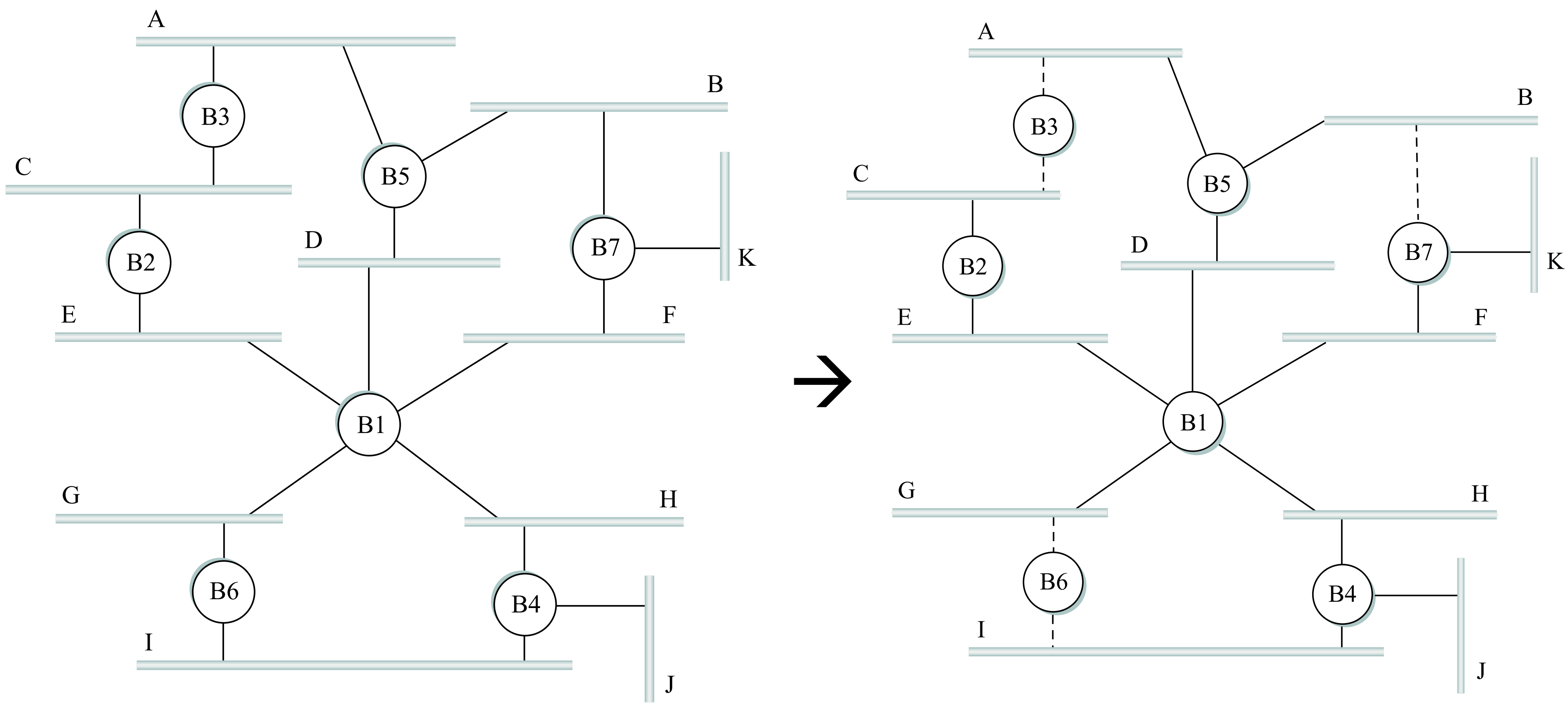
- Decide bridges were a stupid idea.
- Document the limitation and laugh when it is broken
- Detect misconfiguration and complain automatically
- Create a loop-free topology using a subset of the physical links. This is called a minimum spanning tree

What is a spanning tree?



Goal

- Maintain a spanning tree with one root.
- Elect a “designated bridge” on LANs that have more than one bridge.
  - The one “closest” to the root
- Whenever a bridge sees a frame on a port that is *part of the tree*, forward it to all other ports in the tree.
- Ignore frames from other ports.


A
B
B3
C
B5
D
B7
K
B2
E
F
B1
G
H
B6
B4
I
J
A
B3
B
C
B5
D
B7
K
B2
E
F
B1
G
H
B6
B4
I
J

General Design Principles

- All bridges to run the same algorithm
- All bridges start out with zero information
- Bridges send periodic messages about their own state
- Neighbor’s state that hasn’t been refreshed in a while is deleted (this is *soft state*)
- If we all have the same inputs and are running the same algorithm, we converge.

This is a very common design pattern. Learn it. Live it. Love it.

Algorithm

- On boot, assume you are the root
- Start sending messages containing:
  - Your bridge ID
  - Who you think the root is (initially, yourself)
  - Your cost to reach the root (initially, 0)
  - Your port number you’re sending on
- Keep track of the “best” root heard on each port
- Keep track of the “best” root heard from any port

What is “best”?

- B1 is better than B2 if the root ID is lower.
- If the IDs are equal (why?), B1 is better than B2 if its cost is lower
  - This is the only NON-arbitrary metric
- If IDs and costs are equal, B1 is better than B2 if the xmit bridge’s ID is lower (not the root ID, but the bridge in the middle)
- If all that is equal, pick the lower port number

Example. If we hear…

	Root	Cost	Transmitter
Port 1	12	93	51
Port 2	12	85	47
Port 3	81	0	81
Port 4	15	31	27

"Best" root is 12.
If our ID had been smaller than 12, we'd be the root.

We then determine our own cost to the bridge.
Add some cost (e.g., 1) to best root path:
12 is the root, our distance is 86.

Now what?

- Each bridge decides which ports in in the spanning tree.
- Which ports?
  - The port on which we received the “best” root message
  - Ports for which we ARE the best message
  - This is where our assumption comes into play: everyone’s hearing the same input and runs the same algorithm

Example -- if we're 92

	Root ID	Cost	Transmitter
Port 1	81	0	81
Port 2	41	19	125
Port 3	41	12	315
Port 4	41	12	111
Port 5	41	13	190

We assume:

Best root is 41
Best cost from us (bridge 92) is 12+1, or 13
Port 3 or 4 have same cost; 4 is picked with tiebreaker
Thus we transmit 41.13.92

This is a better message than what we heard on Port 1 or 2, so

Root Port is 4
We're the designated Bridge on 1 and 2
Ports 3 and 5 are turned "off"

Refinements

- Avoid temporary loops: listen before you blindly start forwarding

- Configure high-order bits of your ID and ports, to force better links to be the normal ones

- Transmit values like “hello period” with message, elected with root

Key Points

- Bridges are useful
- Auto-configuration is good
- Soft state is an important design pattern
- When do bridged LANs stop scaling? We'll see in the next lectures