Switching and Bridging

COS 460 - Fall 2019

More Card Games

In this exercise we will be emulating an interconnected, switched network to deliver datagrams

- Break into **5 groups of ~5 people each**

- Choose one of your group to be **the switch**
  On the white **team sheet** place the green **switch cards** in ports (numbered spaces). Assign players to the other ports. Choose one of the green spaces as your **default port**

- Choose one of your group to be **the runner** they should have ready access to the other teams

- The remaining members take an orange lettered **host card** and **3 random datagrams** (small white cards)

Go!

Hosts:

1. If there is a datagram in the port assigned to you by the switch, take it and keep it until the end of the game
2. If you have datagrams to send, choose one and place it in your port and wait for it to disappear. You can wait random amounts of time to make it more interesting.

Switches: When a new datagram arrives in a port, move it to the correct port to be delivered or to your default port if you don't know what to do with it. *There can be only one message in a port at a time. You may have to wait for the runner or hosts*

Runners: Watch for new datagram in the **green ports** take and deliver (one at a time) to the switch that is assigned to that port and drop it in the right port. *Don't bring any back.*

Problems?

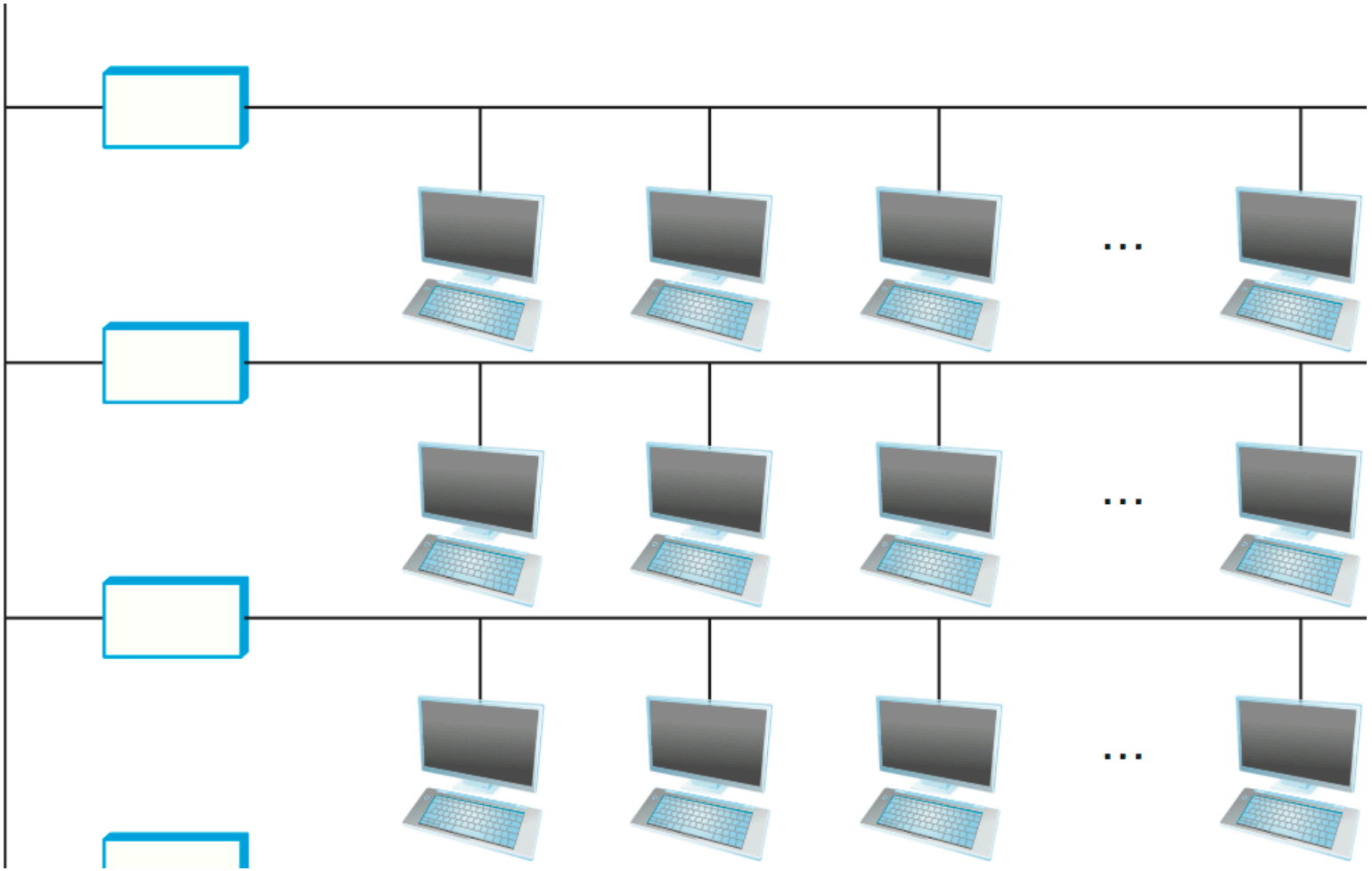
What happened?

Learning Switches

Hosts: Write your address (name) on the outgoing datagrams

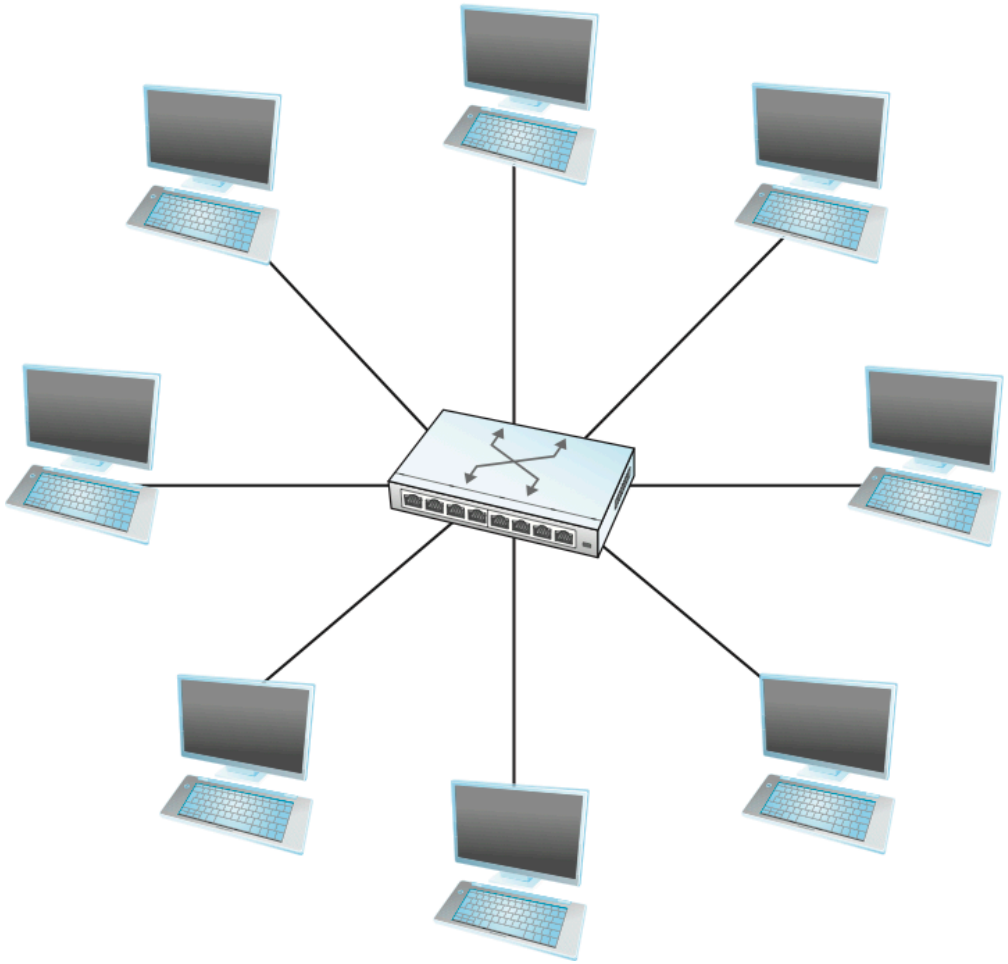
Switches: When moving datagrams, write down the source address in the space it came from

Runners: No changes for you



Ethernet as we know it

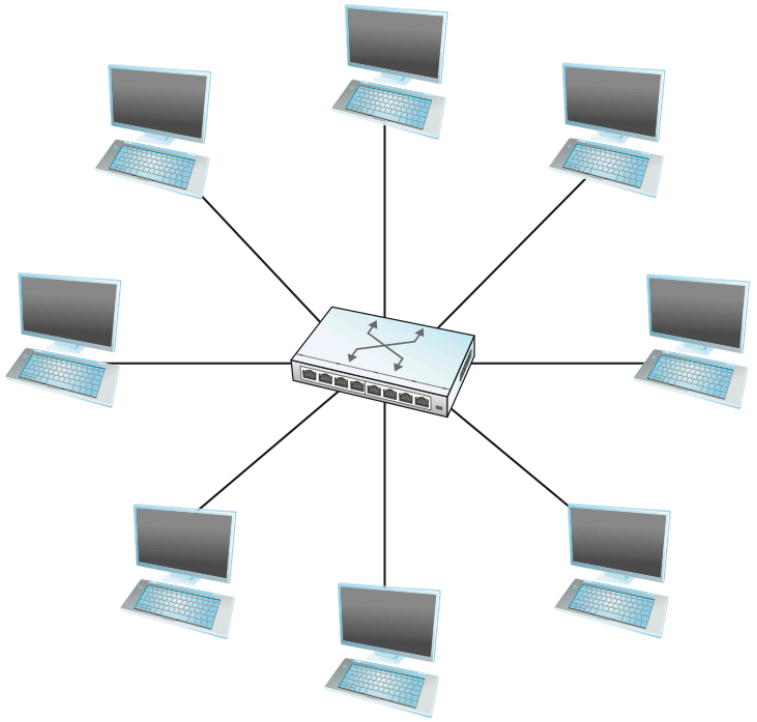
All hosts connected to same “wire”
All hosts see the same signals



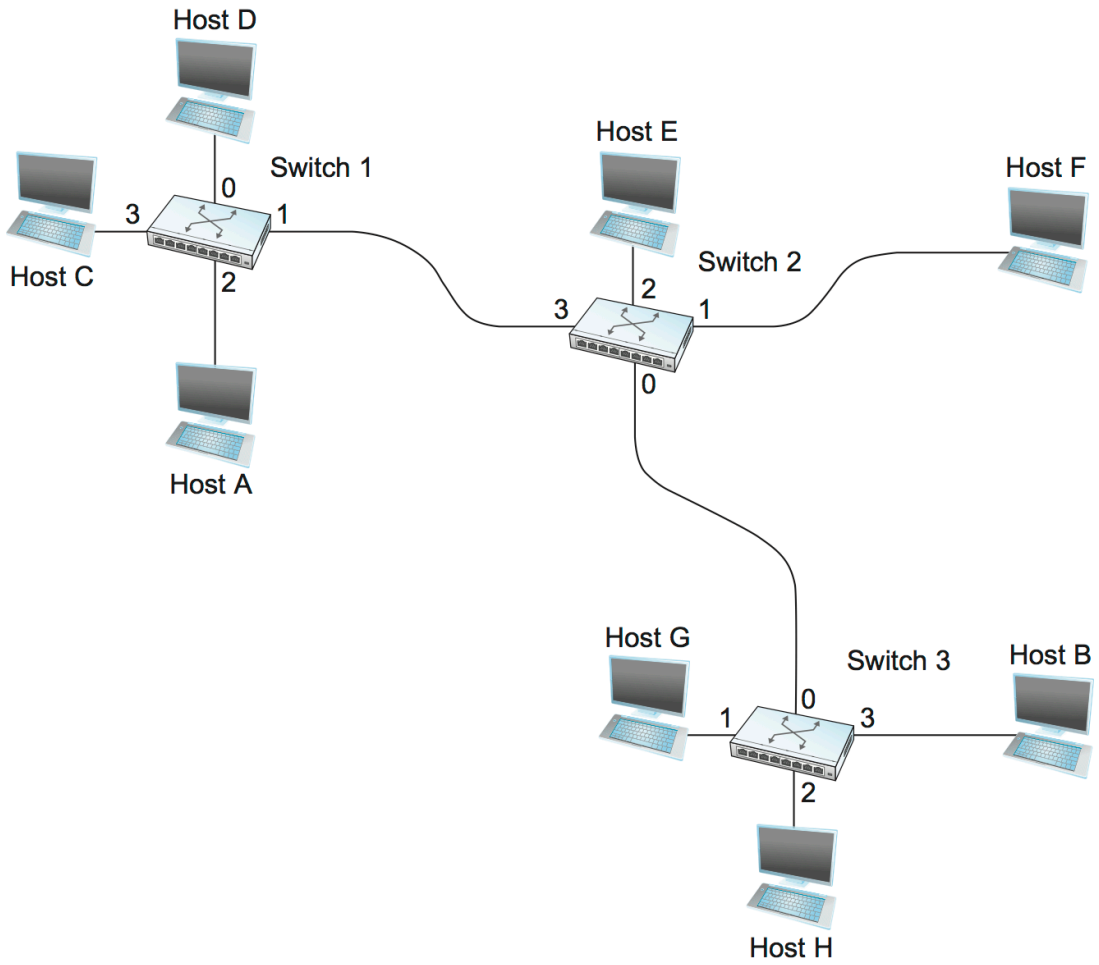
The Star Topology

Hosts connected to central intelligent “switch”

The Star Topology



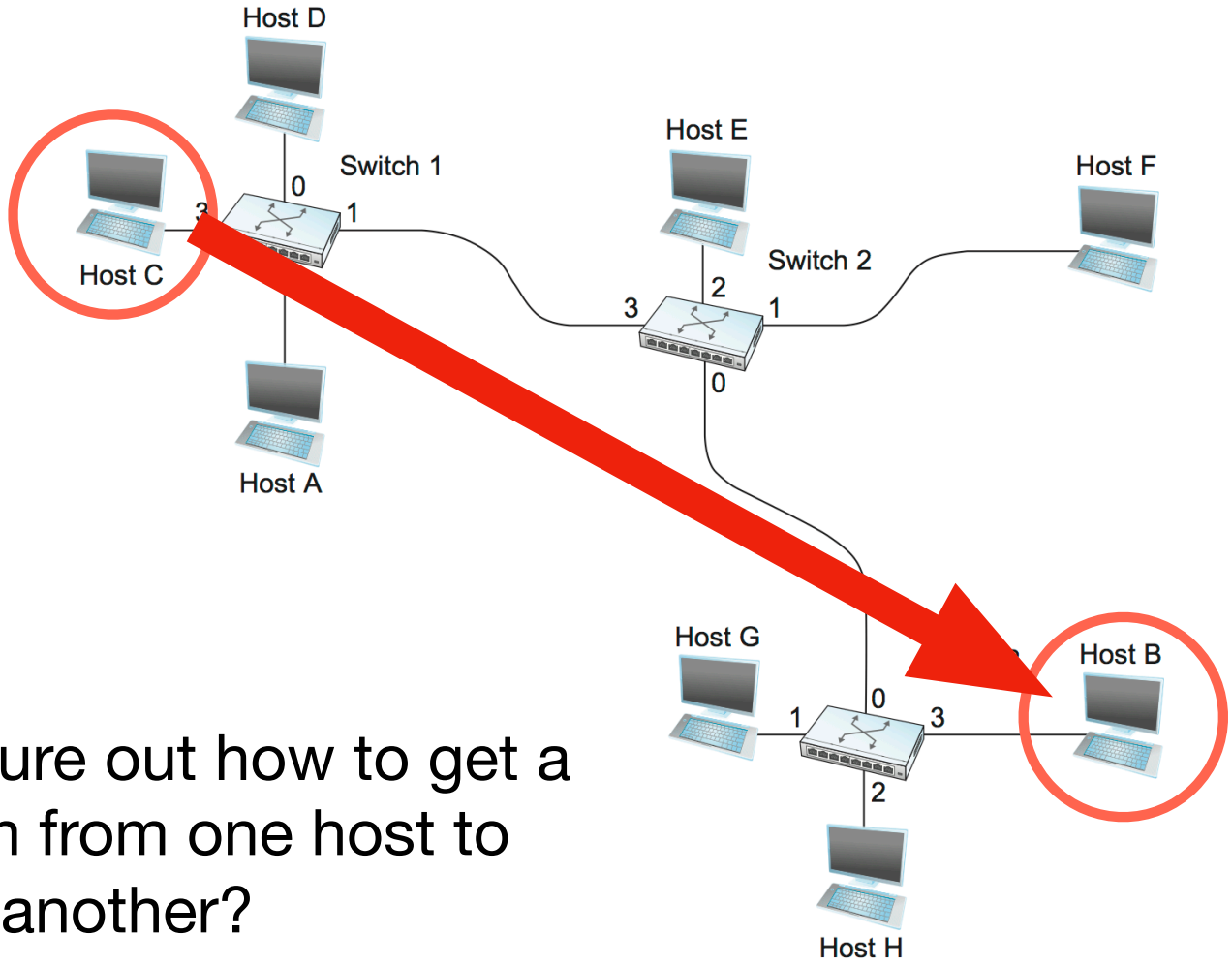
- Fixed number of ***ports***
- Interconnect **hosts** or **switches**
  - …to form larger networks
- No reduction in performance of network*



The Star Topology

interconnecting networks

Interconnected Stars



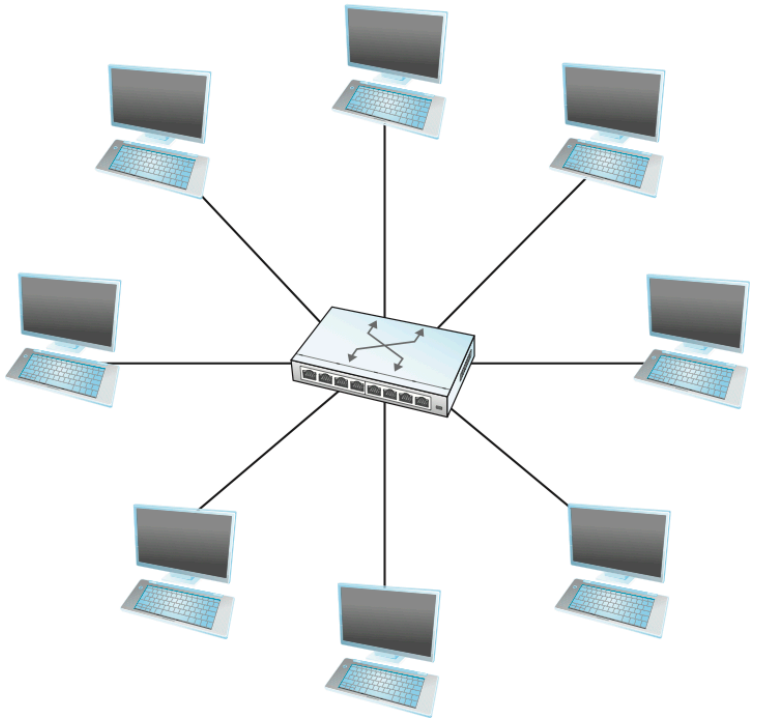
How do figure out how to get a datagram from one host to another?

let's just worry about any path right now. later we will talk about routing.

Some Switching Options

1. Datagram Switching
2. Virtual Circuit Switching
3. Source Routing

Datagram Switching

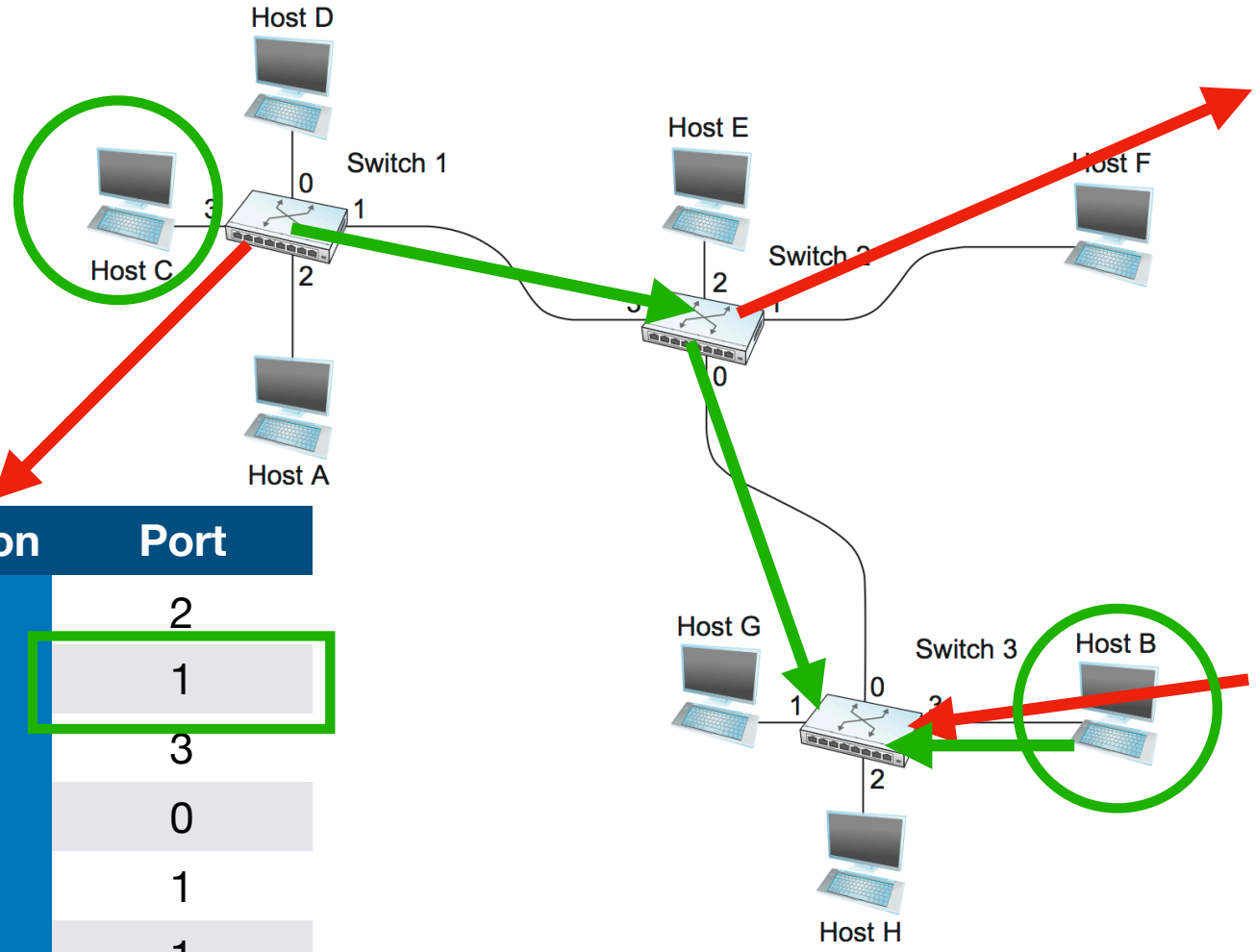


- Run **Ethernet protocol** on each link
- **Broadcast** when needed

Datagram Switching

- A host can send a packet **anywhere** at **any time** (*connectionless*)
- When sending, the host does not know if the network can deliver it or not (*unreliable*)
- Each packet is **delivered independently** of all other packets
- Switches and links fail, alternate paths route around problems

Datagram Forwarding

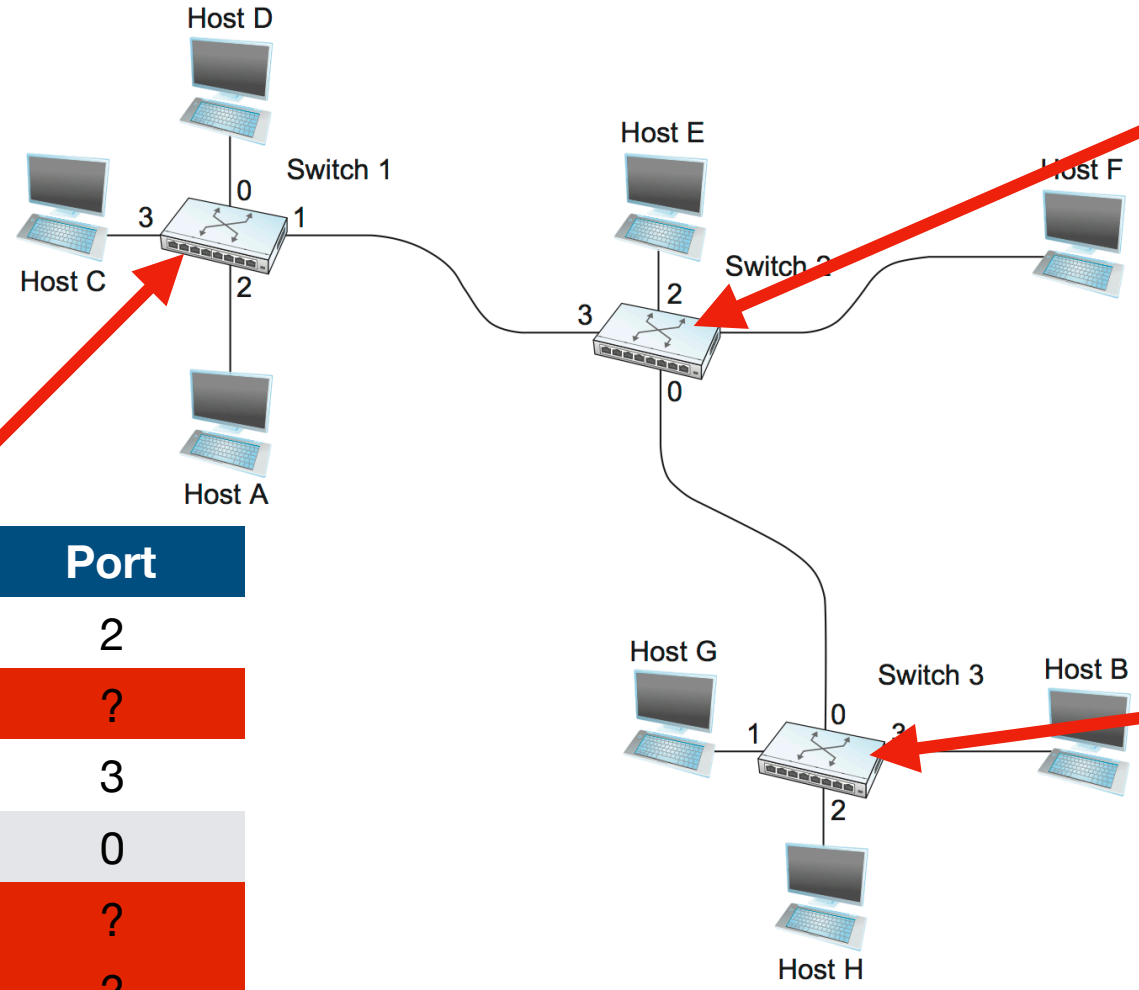


Destination	Port
A	2
B	1
C	3
D	0
E	1
F	1
G	1
H	1
...	...

Destination	Port
A	3
B	0
C	3
D	3
E	2
F	1
...	...

Destination	Port
A	0
B	3
C	0
D	0
E	0
F	0

Forwarding Tables

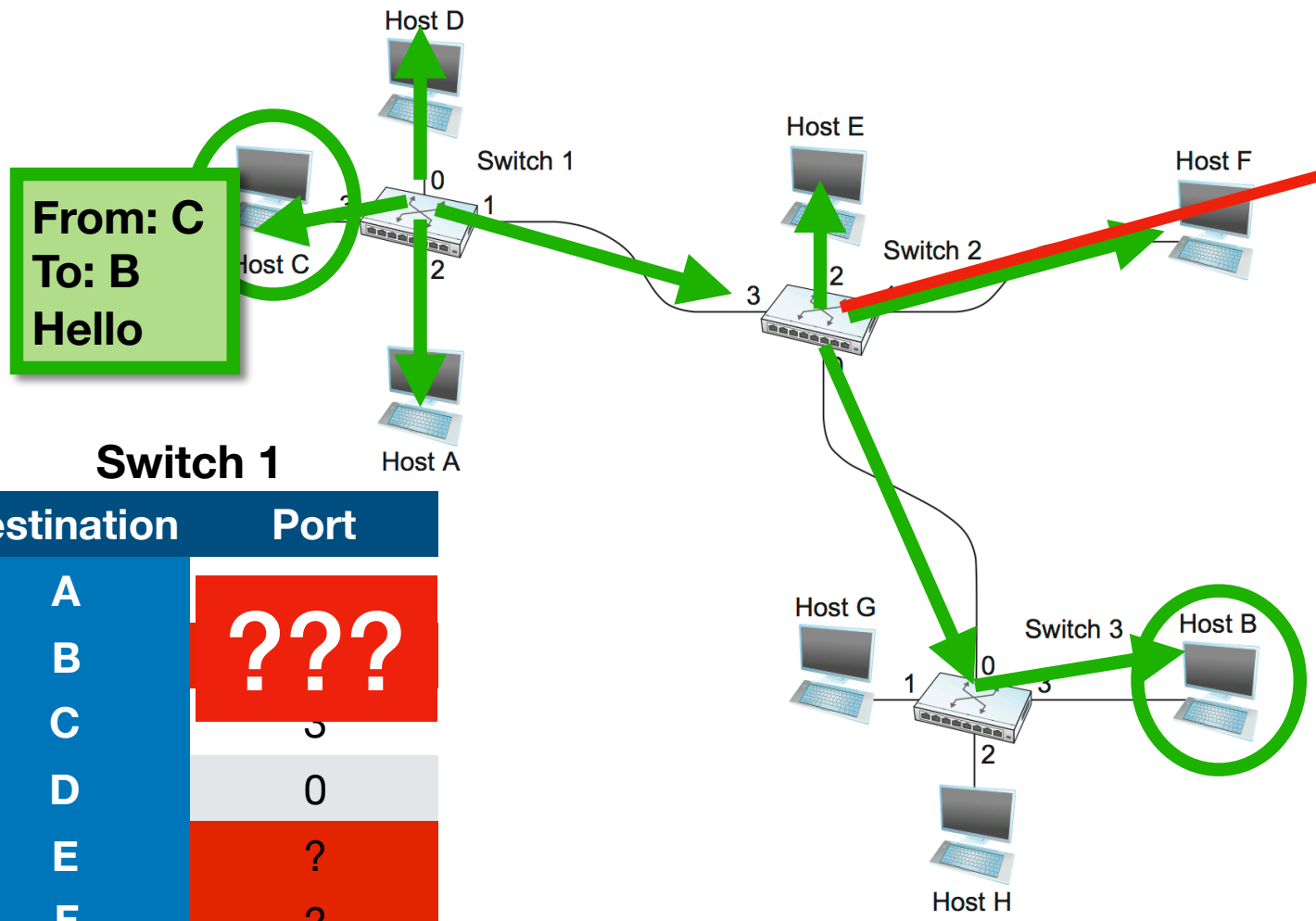


Destination	Port
A	2
B	?
C	3
D	0
E	?
F	?
G	?
H	?
...	...

Destination	Port
A	?
B	?
C	?
D	?
E	2
F	1
...	...

Destination	Port
A	?
B	3
C	?
D	?
E	?
F	?

Forwarding Tables



Switch 1

Destination	Port
A	???
B	
C	3
D	0
E	?
F	?
G	?
H	?
...	...

Switch 2

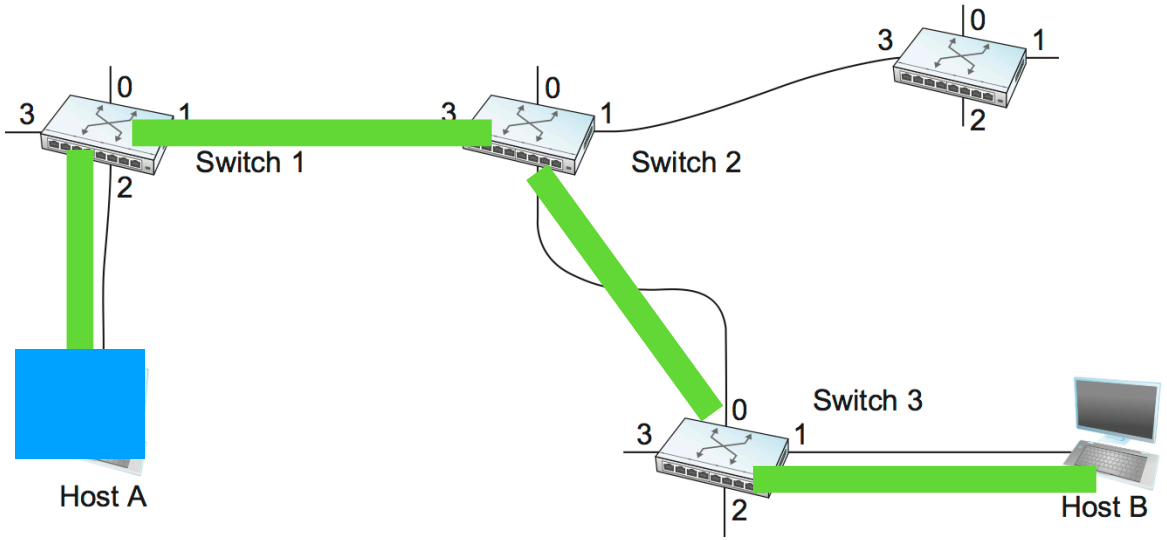
Destination	Port
A	3
B	?
C	?
D	?
E	2
F	1
...	...

Switch 3

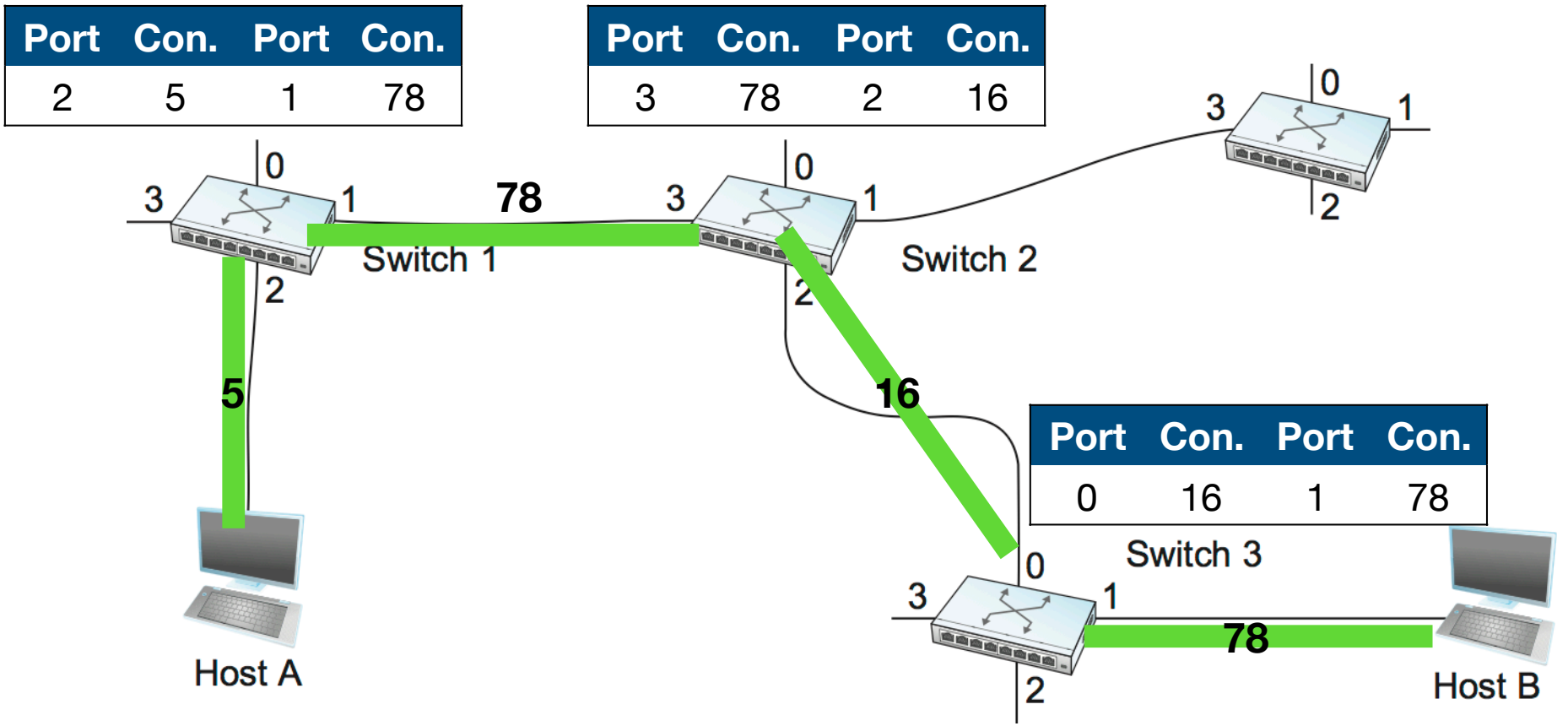
Destination	Port
A	3
B	3
C	?
D	?
E	?
F	?

Virtual Circuit Switching

- Set up connection between hosts through network
- Datagrams then go through connection
- Only need addresses to set up connection



Virtual Circuit Switching



How do we set up the connection?

Virtual Circuit Setup

- Signaling protocol, out of band data between switches
- Host embeds global unique address of destination
- Broadcast like through switches to find destination
- Circuit set up on return path confirmations

Virtual Circuit Switching

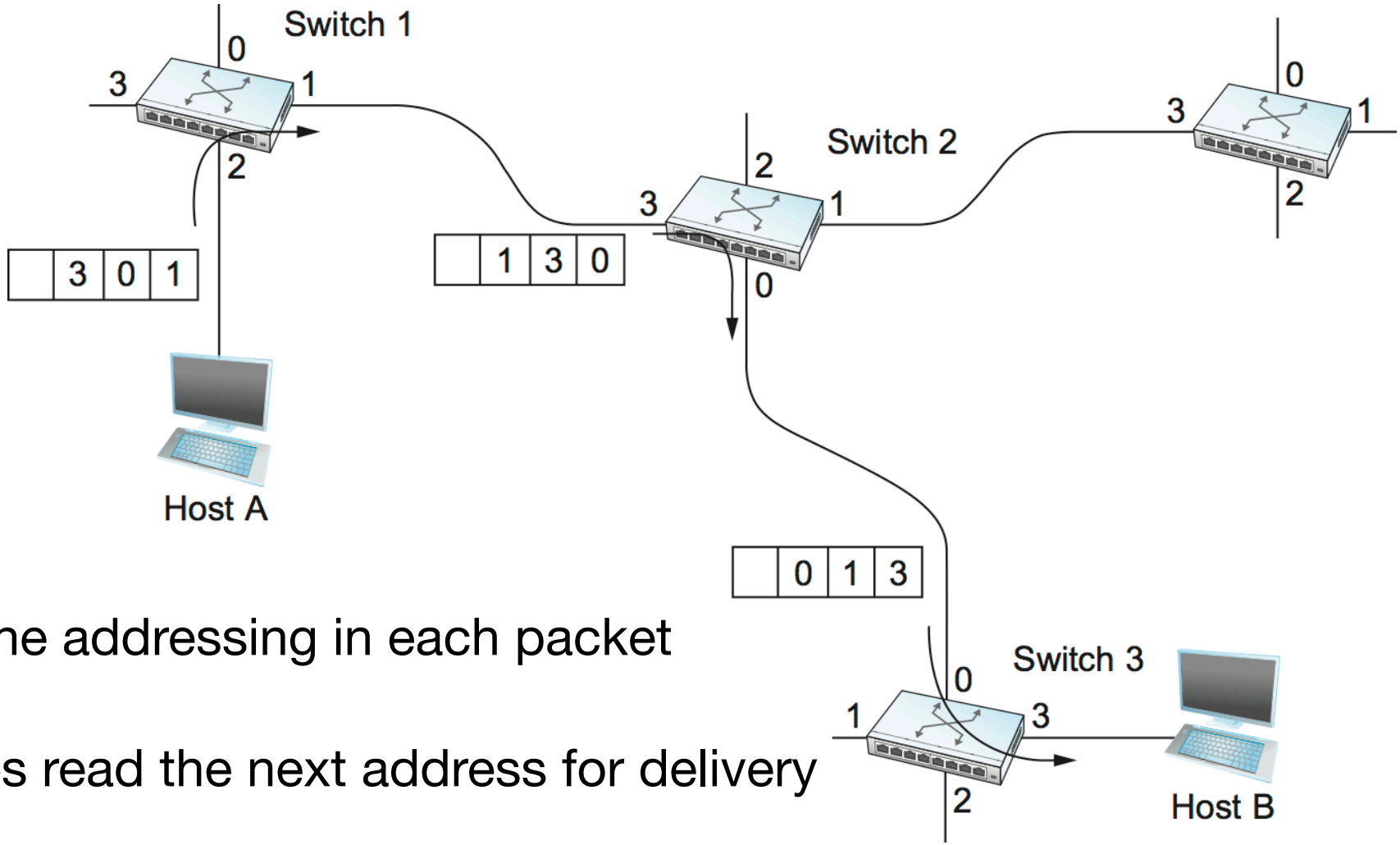
Pros:

- Less overhead on individual packets (less addressing)
- Less variability in delivery time across network
- Lots of knowledge about the network after setup (times, buffers, etc.)
- Quality of Service (QoS) easier* to implement

Cons:

- At least 1 x RTT to setup connection
- If switch or link fails, need to make new connection
- Convoluted out-of-band setup and signaling needed.

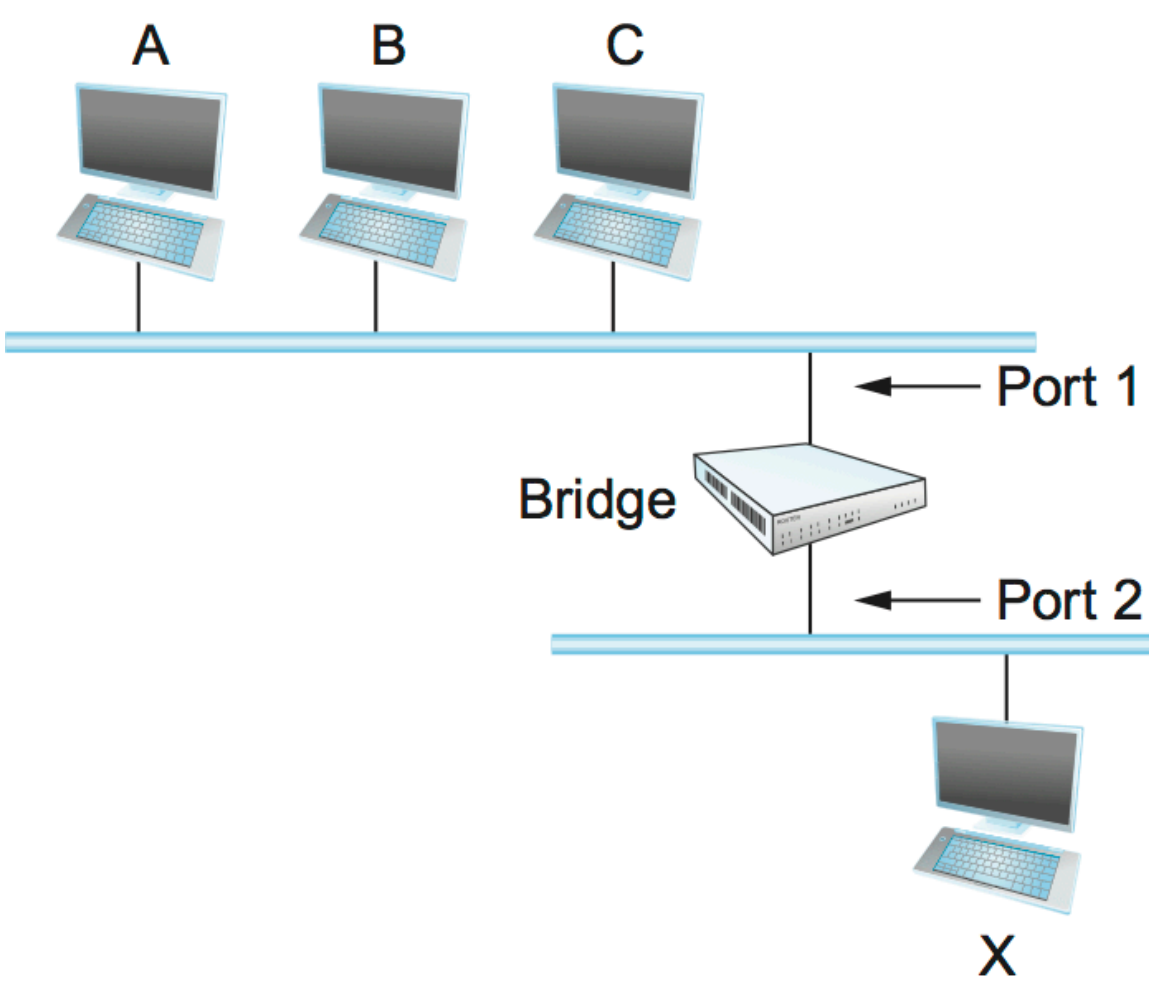
Source Routing



- Put all the addressing in each packet
- Switches read the next address for delivery
- “Rotate” the address field to create return path

Bridges and L2 Switches

- Learning Bridges
- Spanning Tree Algorithm
- Broadcast and Multicast
- Limitations

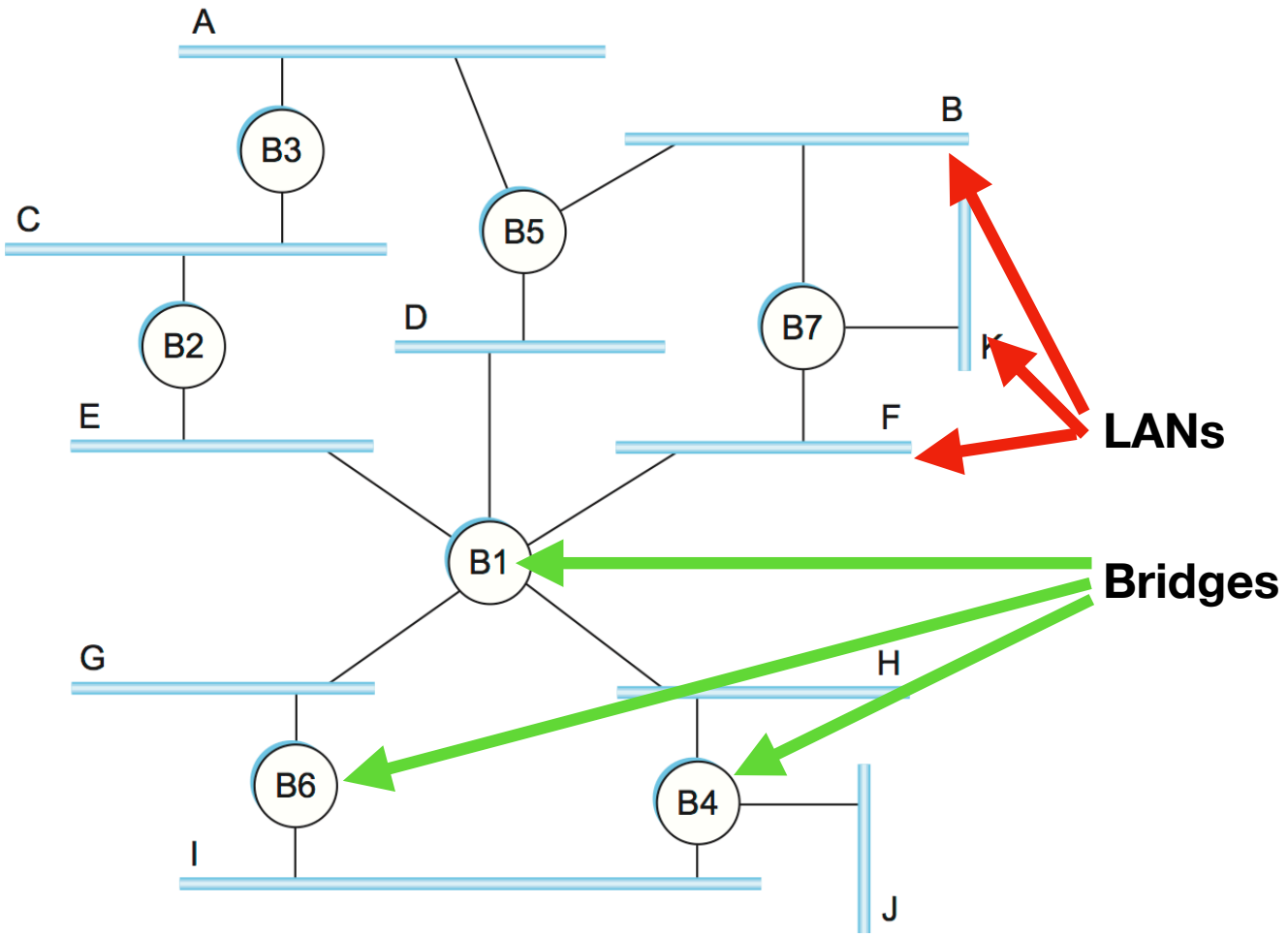


Forwarding Table

Host	Port
A	1
B	1
C	1
…	…
X	2
Y	2
Z	2

Learning Bridge

Same as we saw earlier, listen to the source addresses on each port (*promiscuously*)



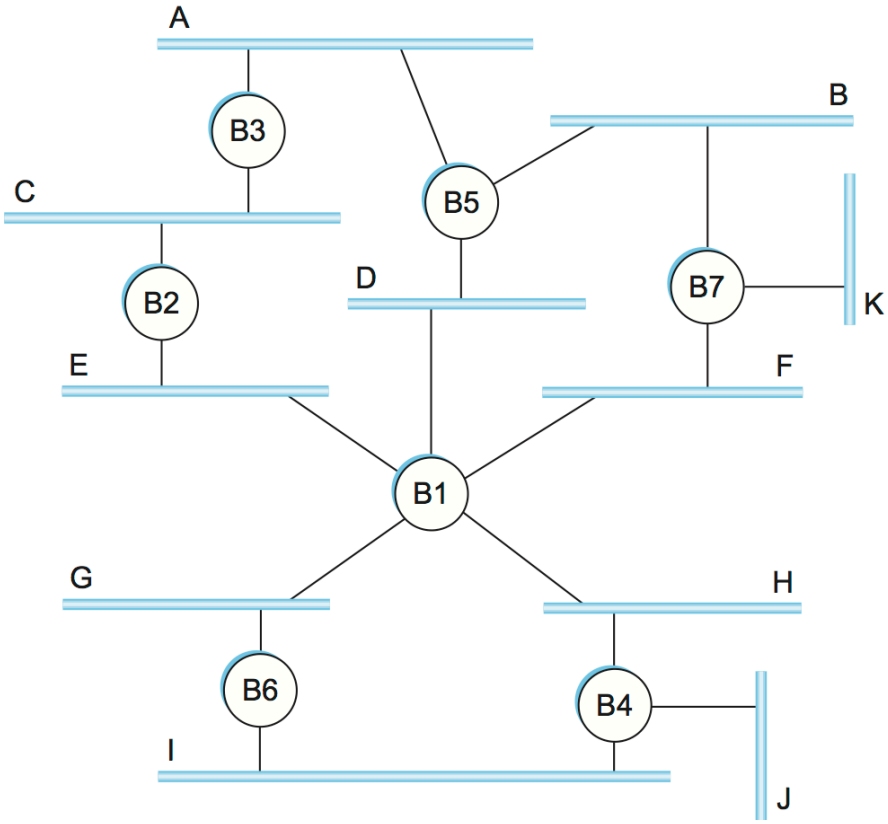
More Complex Connections

When the extended LAN has a **loop** in it, our previous strategies may have delivery problems

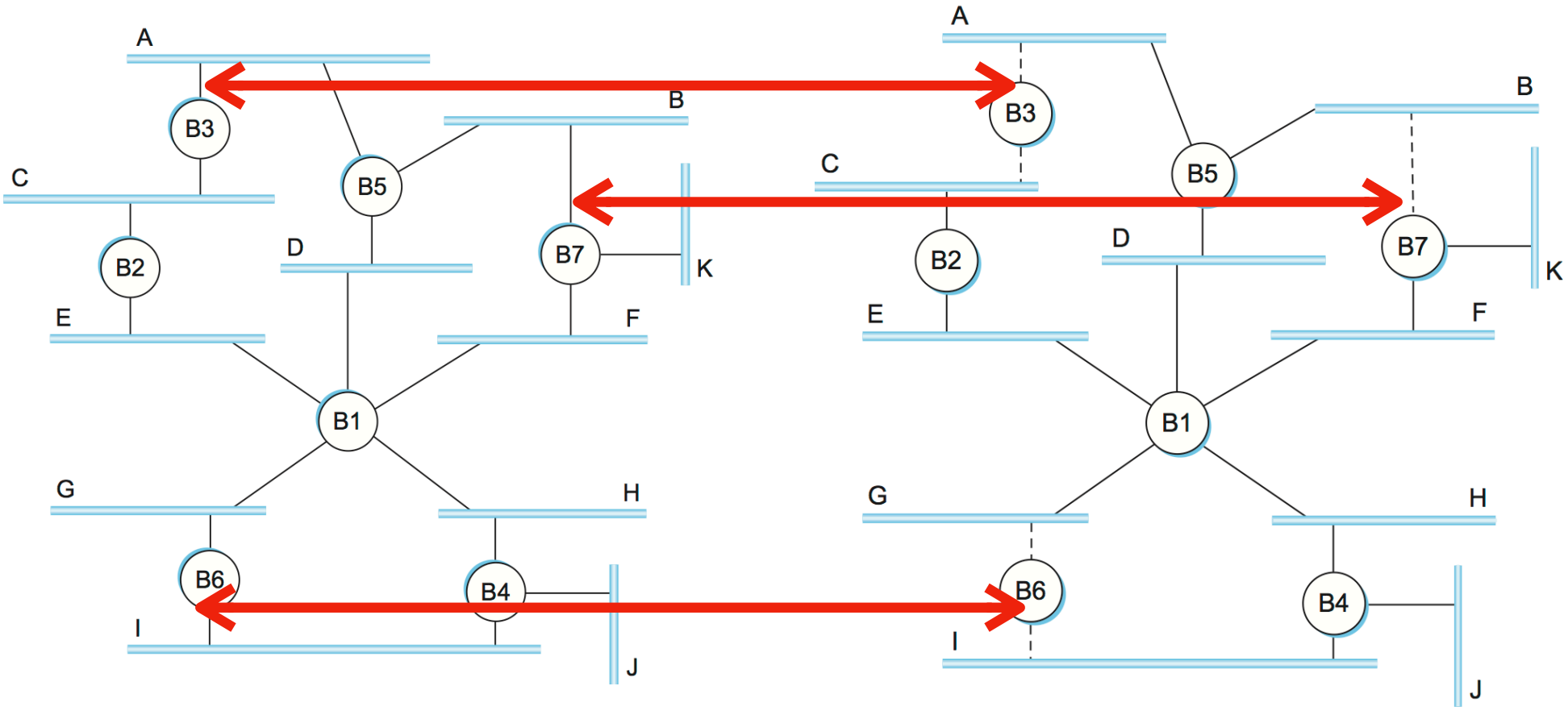
Spanning Tree

The Big Idea

- Bridges **select ports** they will forward packets
- These ports will cover the network **without loops**
- Some switches will **disable** ports to prevent loops
- **Distributed Algorithm**, all bridges run it independently



Spanning Tree Result



Spanning Tree Algorithm

- Exchange Configuration Messages on all ports <BridgeID, RootID, Distance>
  - My Bridge Identifier
  - Bridge I think is the root bridge
  - Distance (in hops) from me to root
- Pretend I'm the root bridge and send out configuration <me, me, 0>
- Record best configuration messages on each port
  - Root ID is smaller than the what I think is the Root ID
  - Root ID is the same but shorter distance
  - Root ID and distance are the same but sending bridge ID is smaller
- If new configuration is better than old one
  - Discard old one
  - Stop generating own configuration messages
  - Send out new configuration adding one (1) to distance field

Another Card Game?

of course!