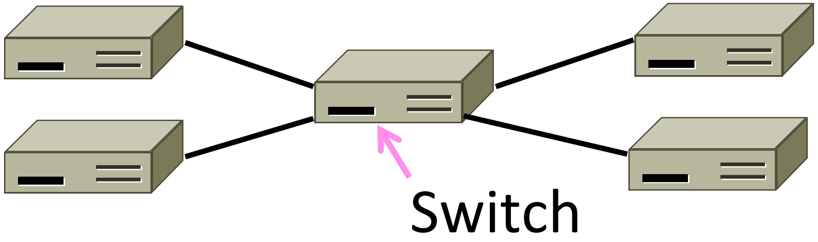
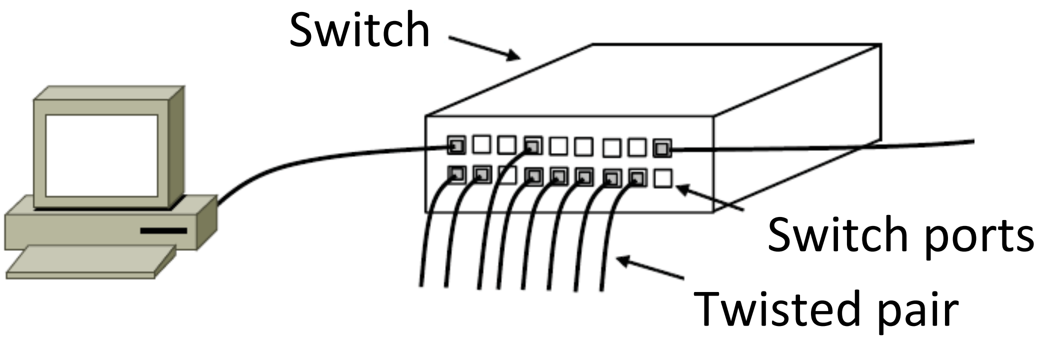
Topic

- How do we connect nodes with a switch instead of multiple access
  - Uses multiple links/wires
  - Basis of modern (switched) Ethernet



Switched Ethernet

- Hosts are wired to Ethernet switches with twisted pair
  - Switch serves to connect the hosts
  - Wires usually run to a closet



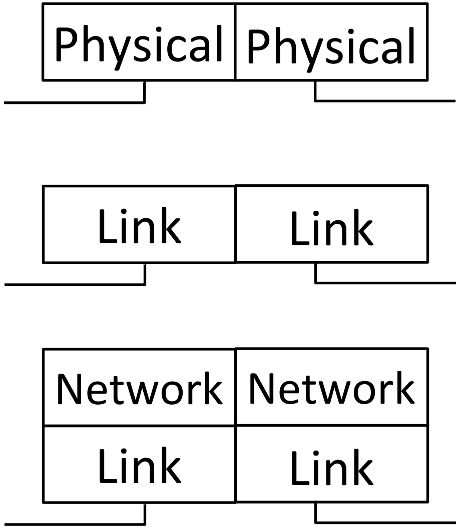
What's in the box?

- Remember from protocol layers:

Hub, or repeater

Switch

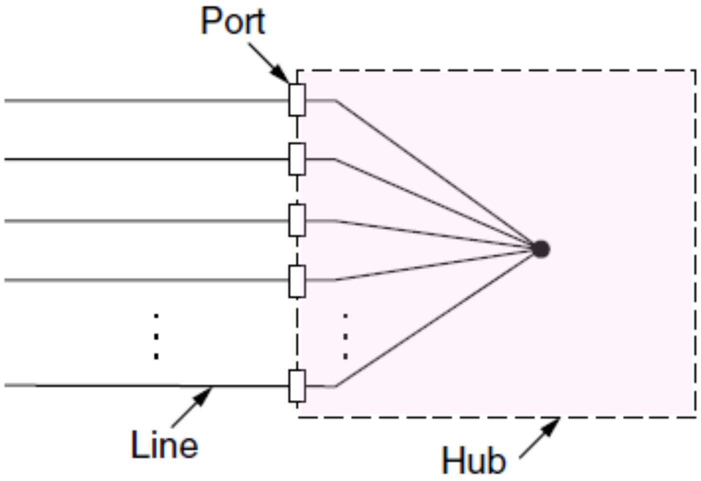
Router



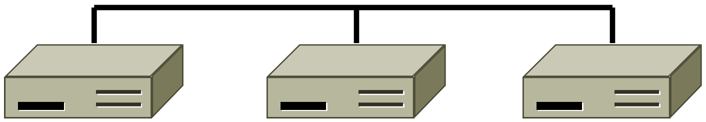
All look like this:

Inside a Hub

- All ports are wired together; more convenient and reliable than a single shared wire

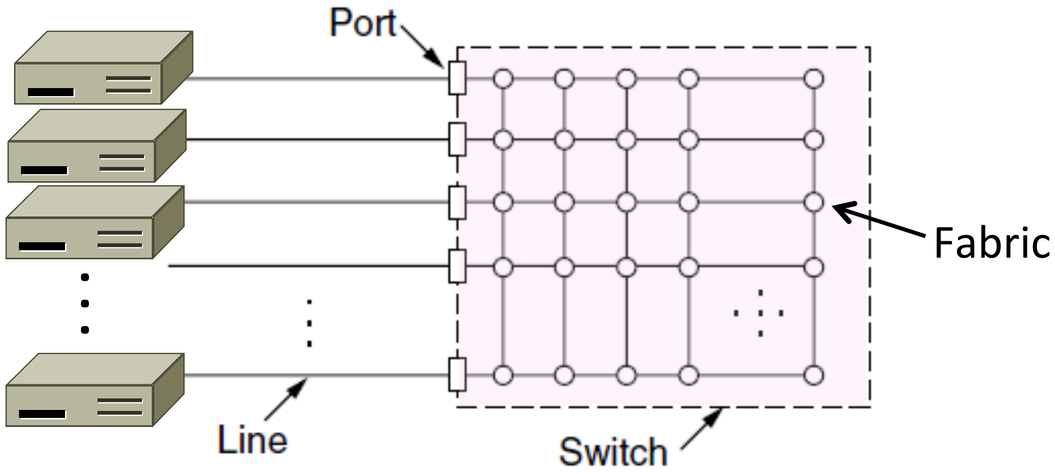


↔



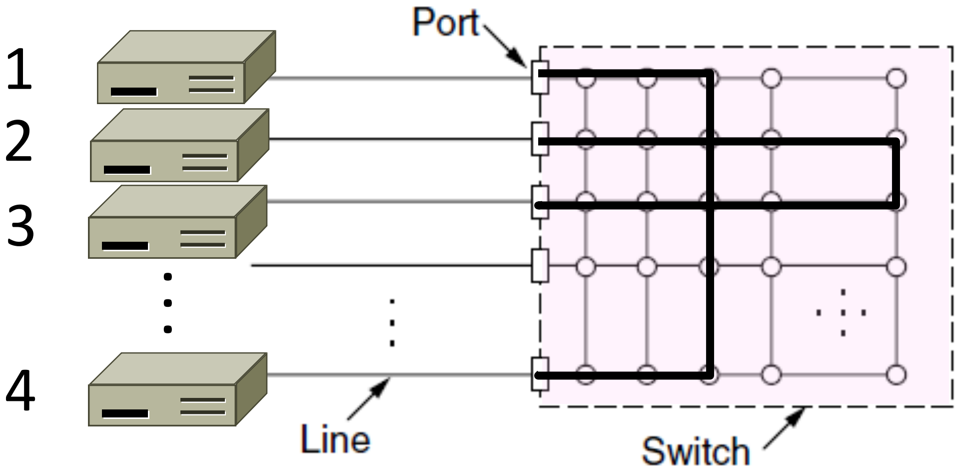
Inside a Switch

- Uses frame addresses to connect input port to the right output port; multiple frames may be switched in parallel



Inside a Switch (2)

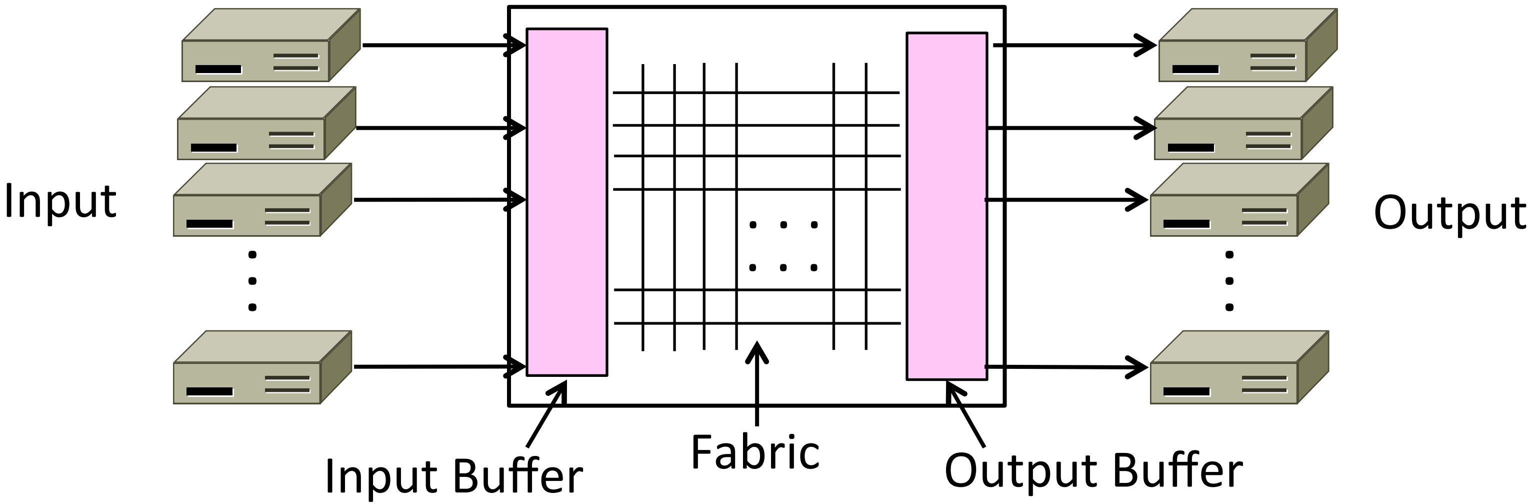
- Port may be used for both input and output (full-duplex)
  - Just send, no multiple access protocol



1 → 4
and
2 → 3

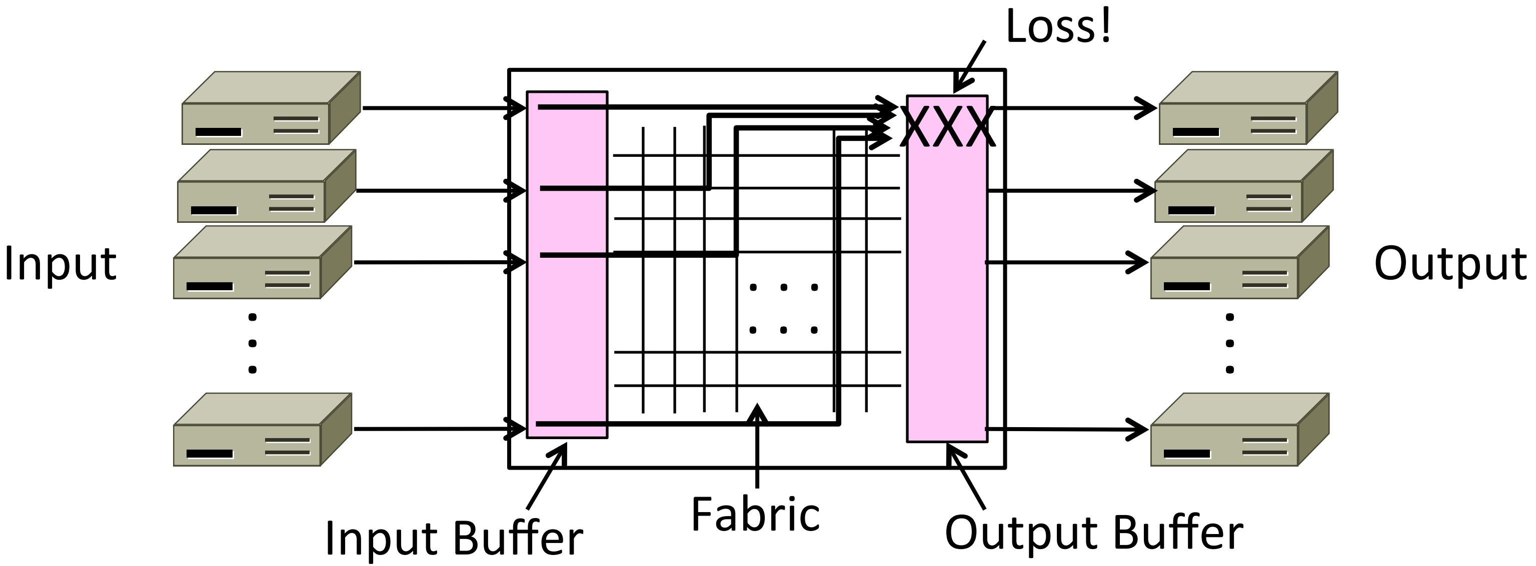
Inside a Switch (3)

- Need buffers for multiple inputs to send to one output



Inside a Switch (4)

- Sustained overload will fill buffer and lead to frame loss

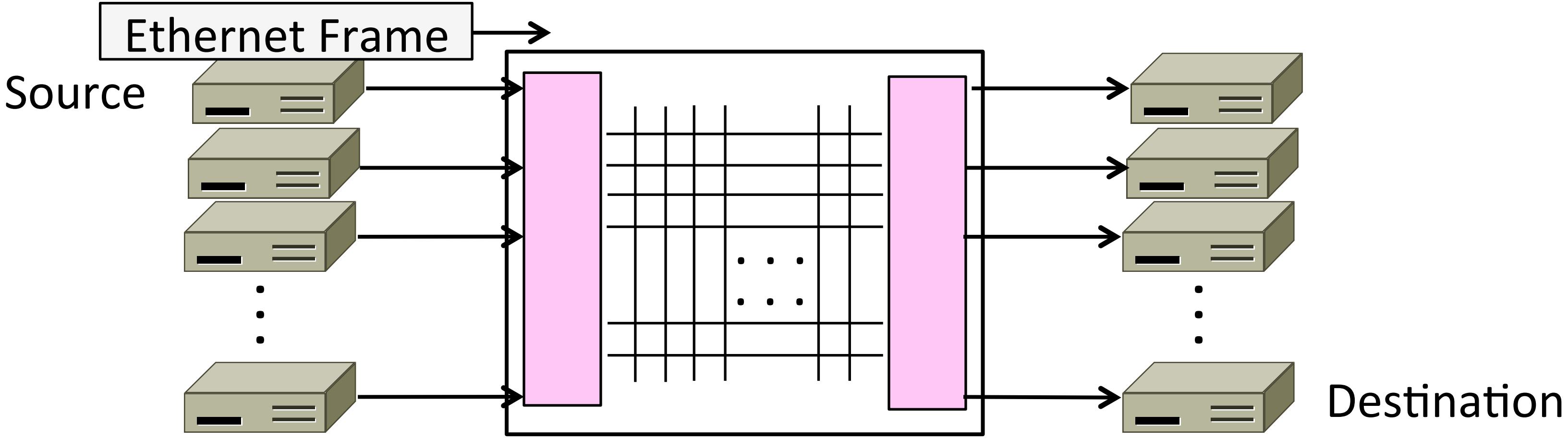


Advantages of Switches

- Switches and hubs have replaced the shared cable of classic Ethernet
  - Convenient to run wires to one location
  - More reliable; wire cut is not a single point of failure that is hard to find
- Switches offer scalable performance
  - E.g., 100 Mbps per port instead of 100 Mbps for all nodes of shared cable / hub

Switch Forwarding

- Switch needs to find the right output port for the destination address in the Ethernet frame. How?
  - Want to let hosts be moved around readily; don't look at IP

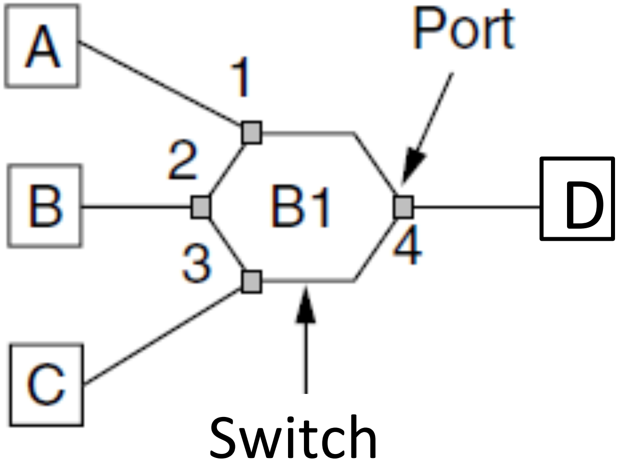


Backward Learning

- Switch forwards frames with a port/address table as follows:
  1. To fill the table, it looks at the source address of input frames
  2. To forward, it sends to the port, or else broadcasts to all ports

Backward Learning (2)

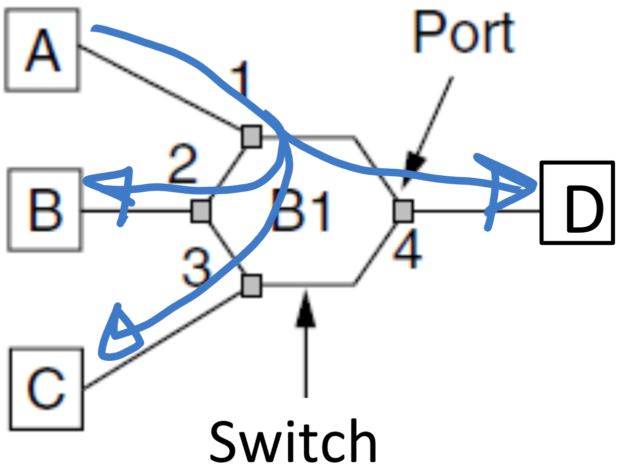
- 1: A sends to D



Address	Port
A	
B	
C	
D	

Backward Learning (3)

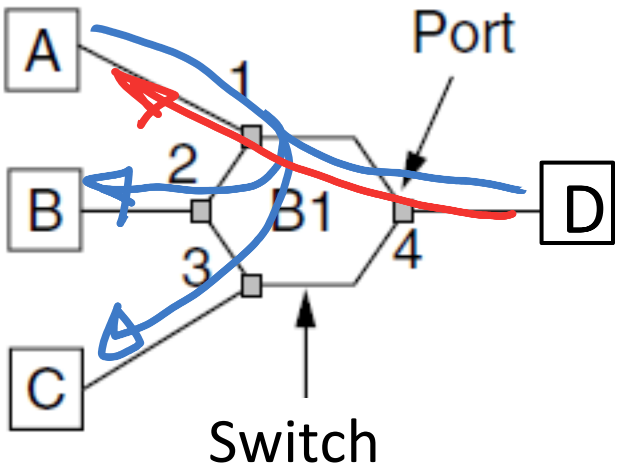
- 2: D sends to A



Address	Port
A	1
B	
C	
D	

Backward Learning (4)

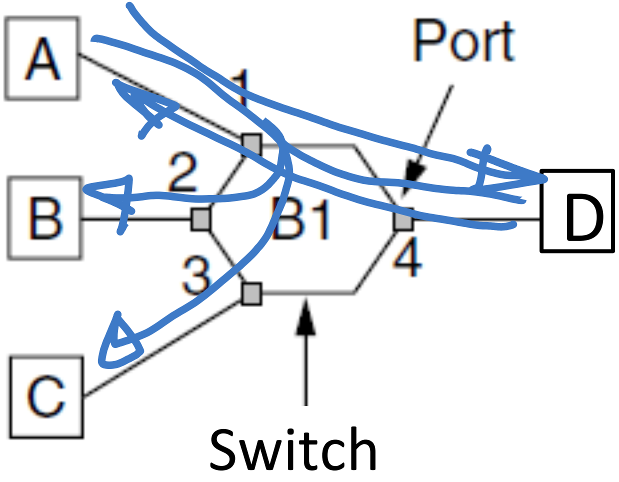
- 3: A sends to D



Address	Port
A	1
B	
C	
D	4

Backward Learning (5)

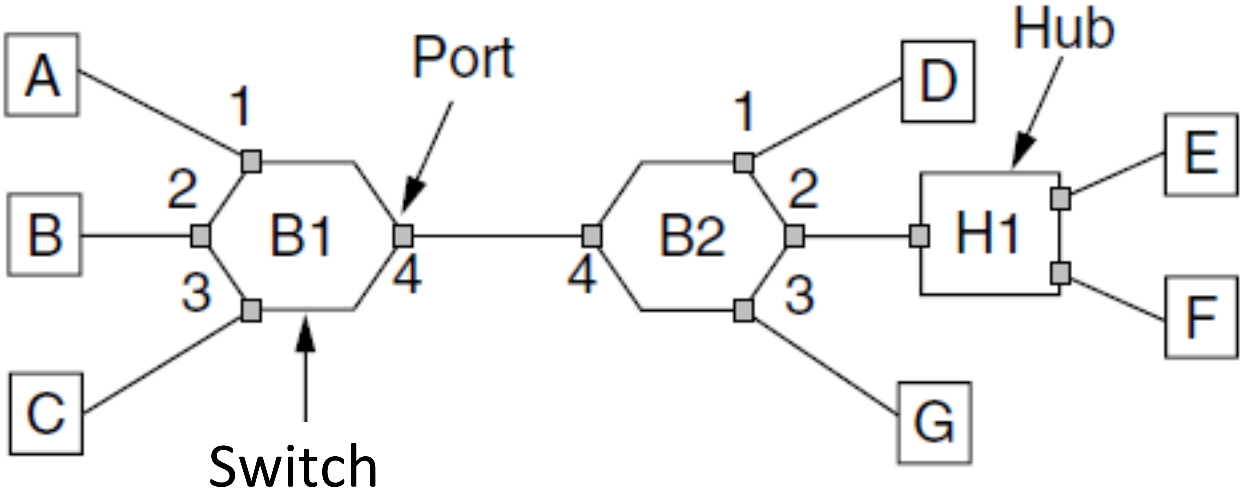
- 3: A sends to D



Address	Port
A	1
B	
C	
D	4

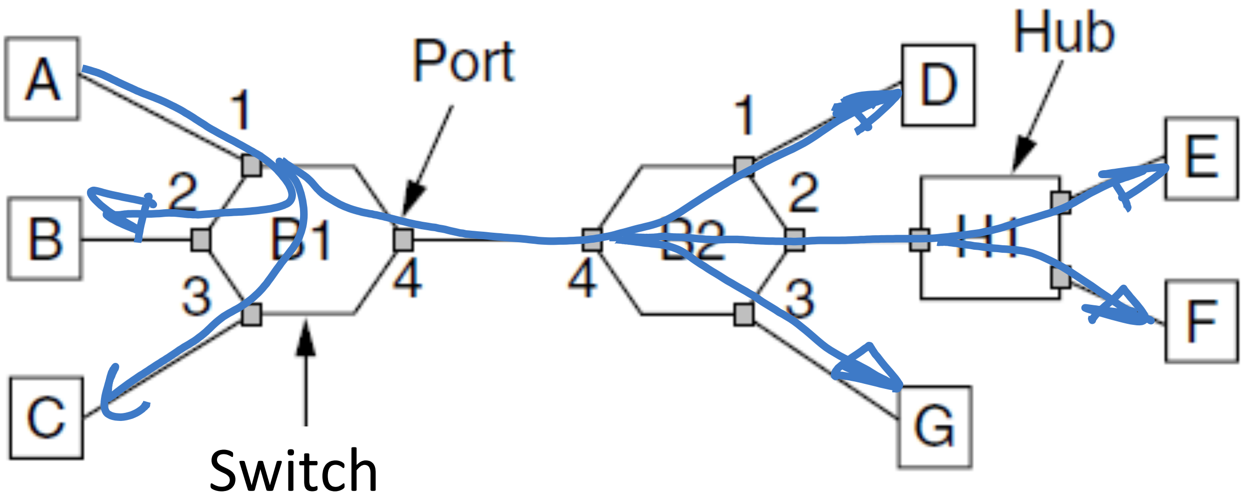
Learning with Multiple Switches

- Just works with multiple switches and a mix of hubs *assuming no loops*, e.g., A sends to D then D sends to A



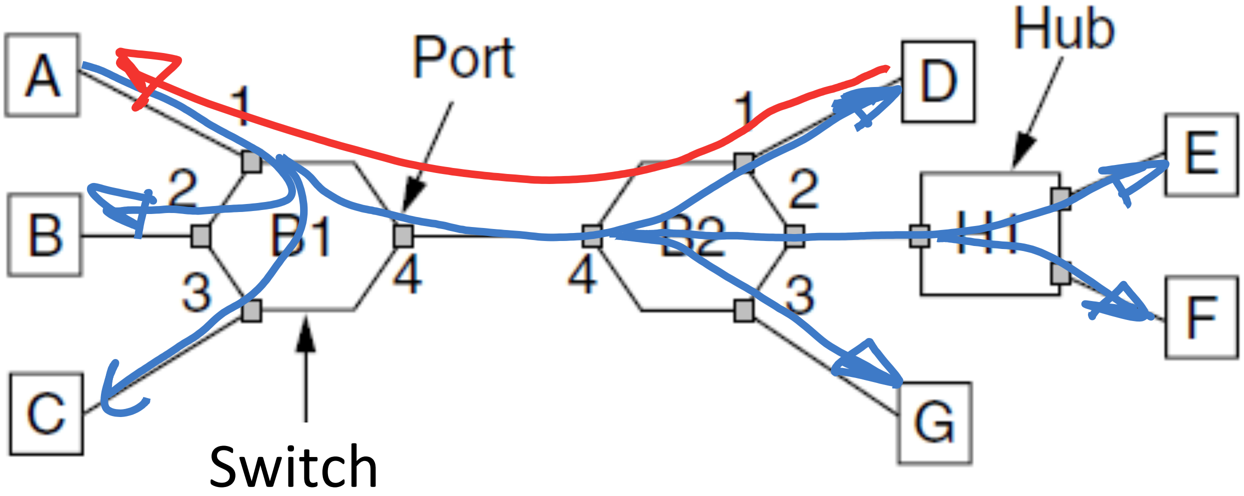
Learning with Multiple Switches (2)

- Just works with multiple switches and a mix of hubs *assuming no loops*, e.g., A sends to D then D sends to A



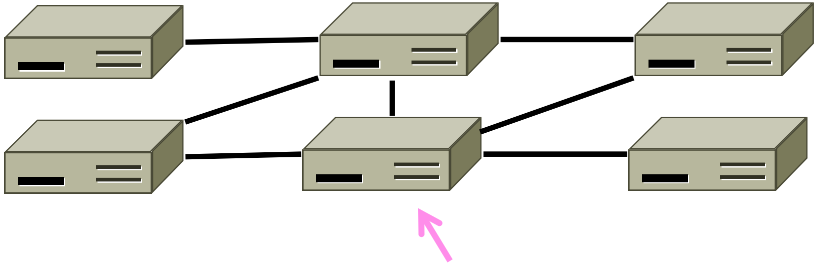
Learning with Multiple Switches (3)

- Just works with multiple switches and a mix of hubs *assuming no loops*, e.g., A sends to D then D sends to A



Topic

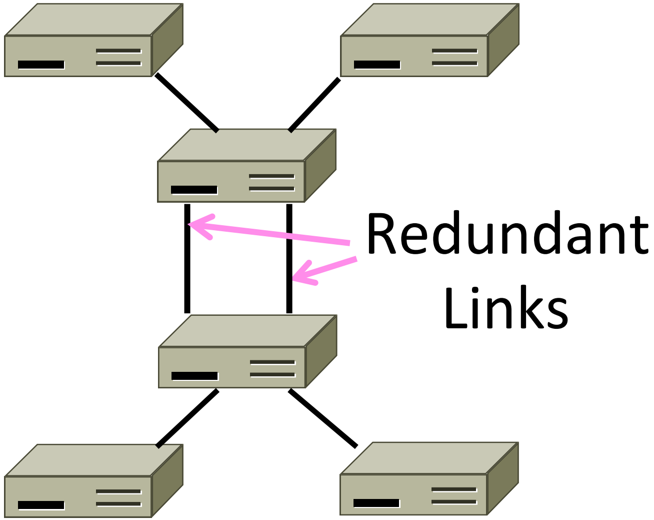
- How can we connect switches in any topology so they just work
  - This is part 2 of switched Ethernet



Loops – yikes!

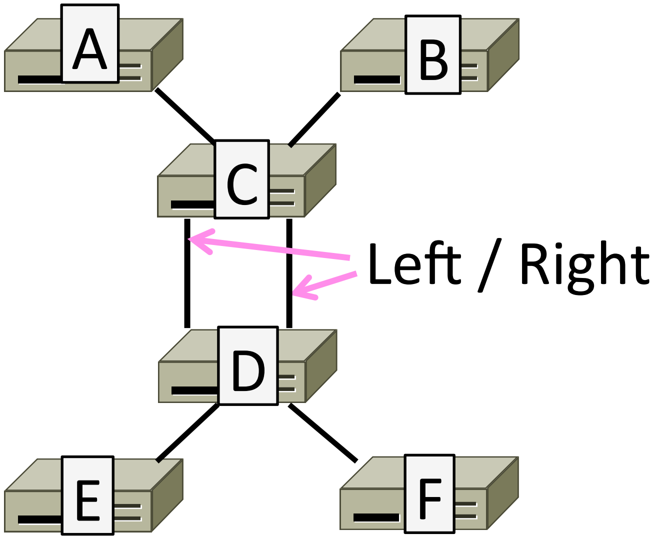
Problem – Forwarding Loops

- May have a loop in the topology
  - Redundancy in case of failures
  - Or a simple mistake
- Want LAN switches to "just work"
  - Plug-and-play, no changes to hosts
  - But loops cause a problem …



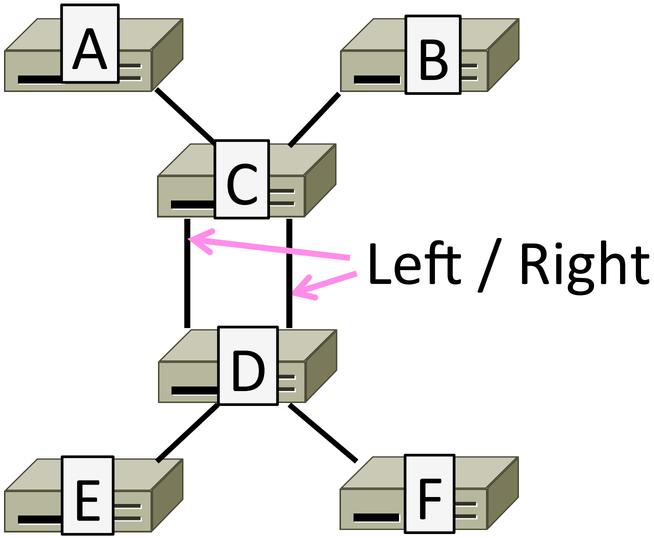
Forwarding Loops (2)

- Suppose the network is started and A sends to F. What happens?



Forwarding Loops (3)

- Suppose the network is started and A sends to F. What happens?
  - A → C → B, D-left, D-right
  - D-left → C-right, E, F
  - D-right → C-left, E, F
  - C-right → D-left, A, B
  - C-left → D-right, A, B
  - D-left → …
  - D-right → …

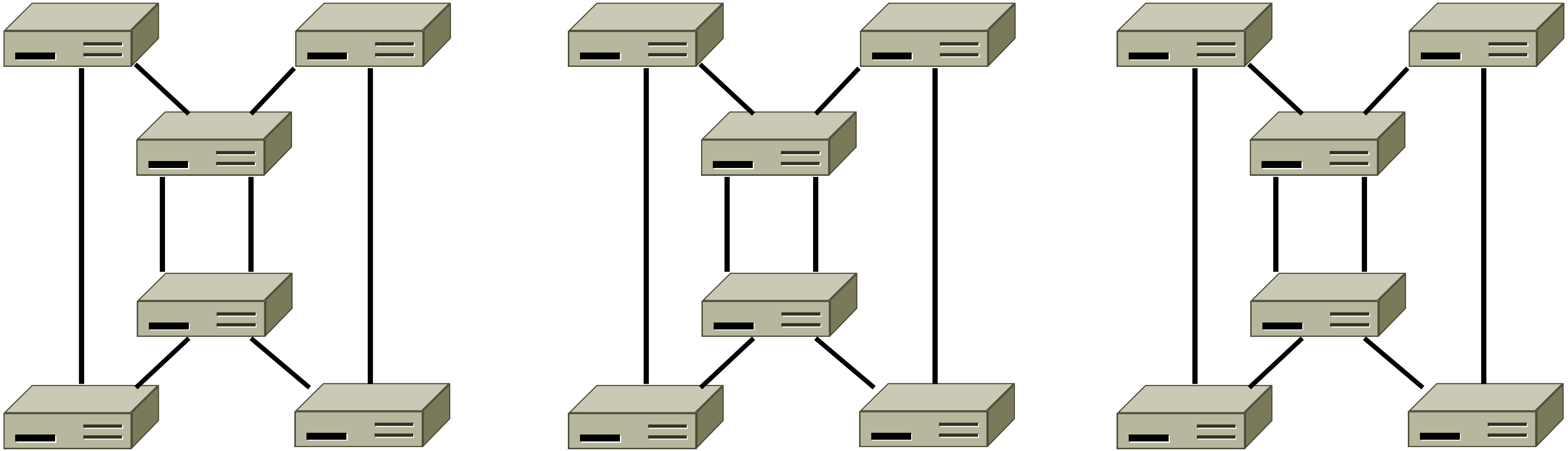


Spanning Tree Solution

- Switches collectively find a <u>spanning tree</u> for the topology
  - A subset of links that is a tree (no loops) and reaches all switches
  - They switches forward as normal on the spanning tree
  - Broadcasts will go up to the root of the tree and down all the branches

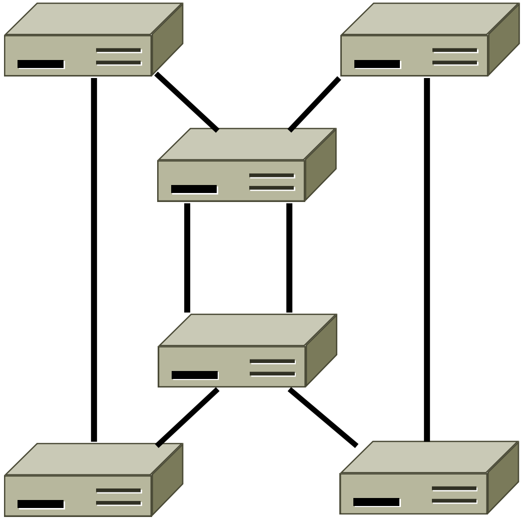
Spanning Tree (2)

Topology | One ST | Another ST

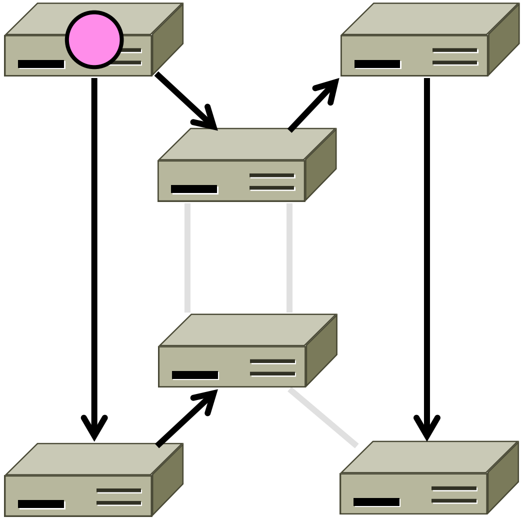


Spanning Tree (3)

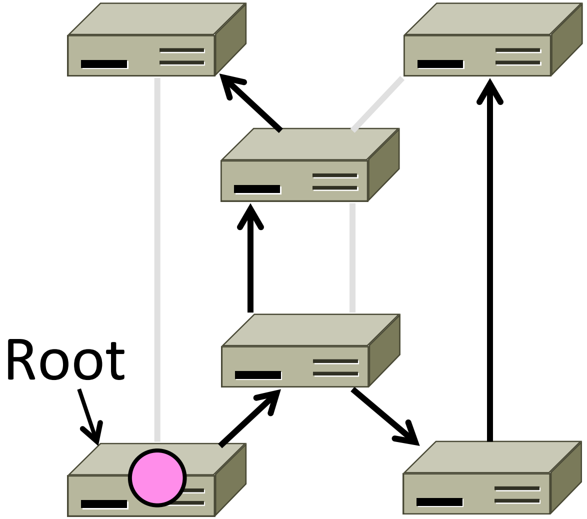
Topology



One ST



Another ST



Spanning Tree Algorithm

- Rules of the distributed game:
  - All switches run the same algorithm
  - They start with no information
  - Operate in parallel and send messages
  - Always search for the best solution

- Ensures a highly robust solution
  - Any topology, with no configuration
  - Adapts to link/switch failures, …

Radia Perlman (1952–)

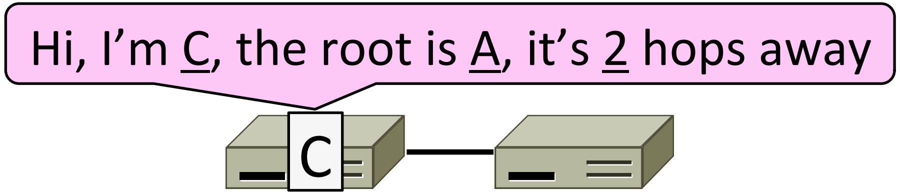
- Key early work on routing protocols
  - Routing in the ARPANET
  - Spanning Tree for switches (next)
  - Link-state routing (later)
- Now focused on network security

Spanning Tree Algorithm (2)

- Outline:
  1. Elect a root node of the tree (switch with the lowest address)
  2. Grow tree as shortest distances from the root (using lowest address to break distance ties)
  3. Turn off ports for forwarding if they aren’t on the spanning tree

Spanning Tree Algorithm (3)

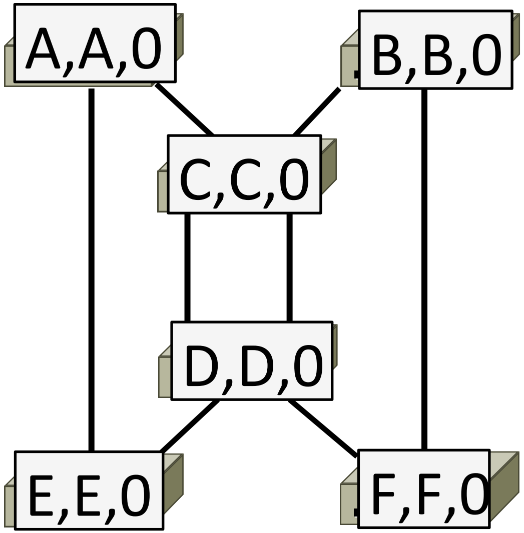
- Details:
  - Each switch initially believes it is the root of the tree
  - Each switch sends periodic updates to neighbors with:
    - Its address, address of the root, and distance (in hops) to root
  - Switches favors ports with shorter distances to lowest root
    - Uses lowest address as a tie for distances



or (C, A, 2)

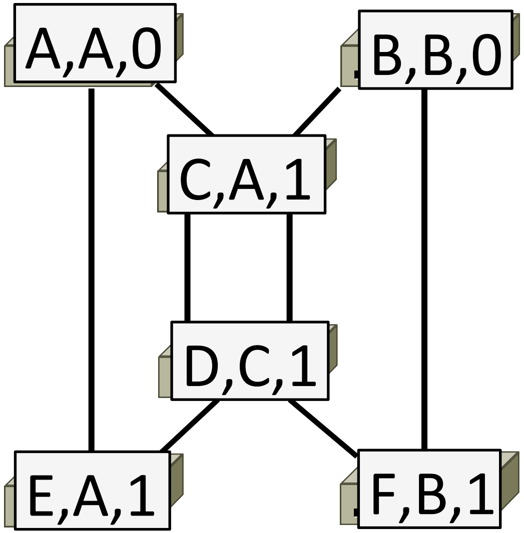
Spanning Tree Example

- 1st round, sending:
  - A sends (A, A, 0) to say it is root
  - B, C, D, E, and F do likewise
- 1st round, receiving:
  - A still thinks is it (A, A, 0)
  - B still thinks (B, B, 0)
  - C updates to (C, A, 1)
  - D updates to (D, C, 1)
  - E updates to (E, A, 1)
  - F updates to (F, B, 1)



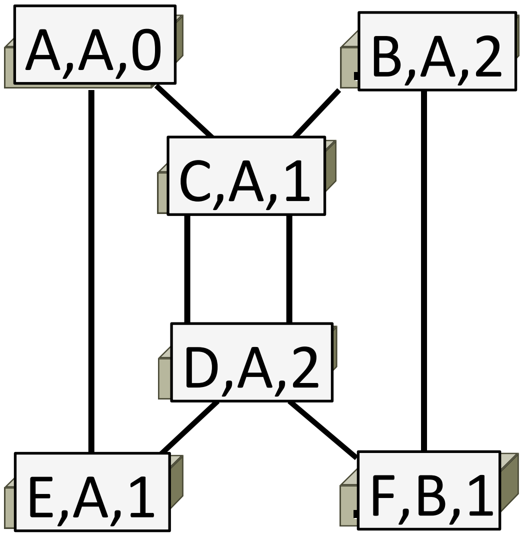
Spanning Tree Example (2)

- 2nd round, sending
  - Nodes send their updated state
- 2nd round receiving:
  - A remains (A, A, 0)
  - B updates to (B, A, 2) via C
  - C remains (C, A, 1)
  - D updates to (D, A, 2) via C
  - E remains (E, A, 1)
  - F remains (F, B, 1)



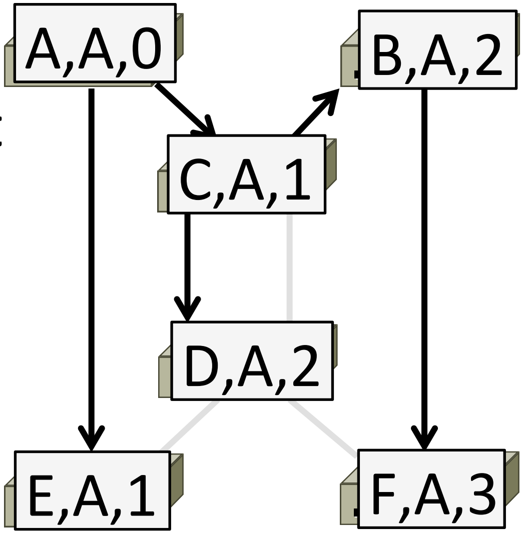
Spanning Tree Example (3)

- 3rd round, sending
  - Nodes send their updated state
- 3rd round receiving:
  - A remains (A, A, 0)
  - B remains (B, A, 2) via C
  - C remains (C, A, 1)
  - D remains (D, A, 2) via C-left
  - E remains (E, A, 1)
  - F updates to (F, A, 3) via B



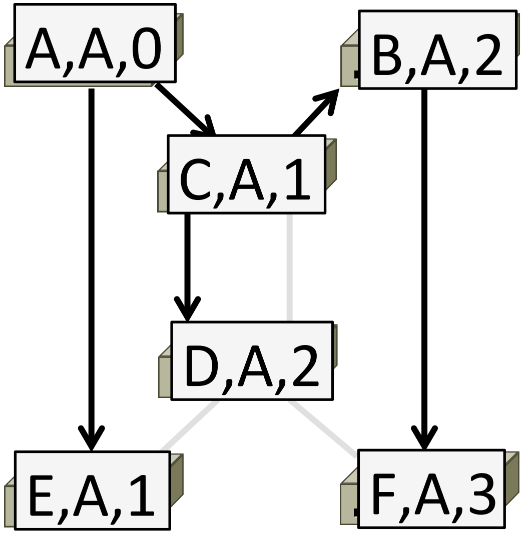
Spanning Tree Example (4)

- 4^{th} round
  - Steady-state has been reached
  - Nodes turn off forwarding that is not on the spanning tree
- Algorithm continues to run
  - Adapts by timing out information
  - E.g., if A fails, other nodes forget it, and B will become the new root



Spanning Tree Example (5)

- Forwarding proceeds as usual on the ST
- Initially D sends to F:

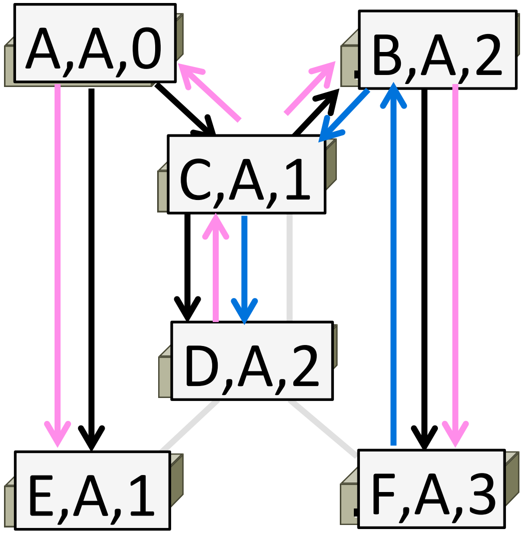


- And F sends back to D:

Spanning Tree Example (6)

- Forwarding proceeds as usual on the ST
- Initially D sends to F:
  - D → C-left
  - C → A, B
  - A → E
  - B → F
- And F sends back to D:
  - F → B
  - B → C
  - C → D

  (hm, not such a great route)



Algorhyme (Radia Perlman, 1985)

I think that I shall never see
A graph more lovely than a tree.
A tree whose crucial property
Is loop-free connectivity.
A tree that must be sure to span
So packets can reach every LAN.
First, the root must be selected.
By ID, it is elected.
Least-cost paths from root are traced.
In the tree, these paths are placed.
A mesh is made by folks like me,
Then bridges find a spanning tree.