

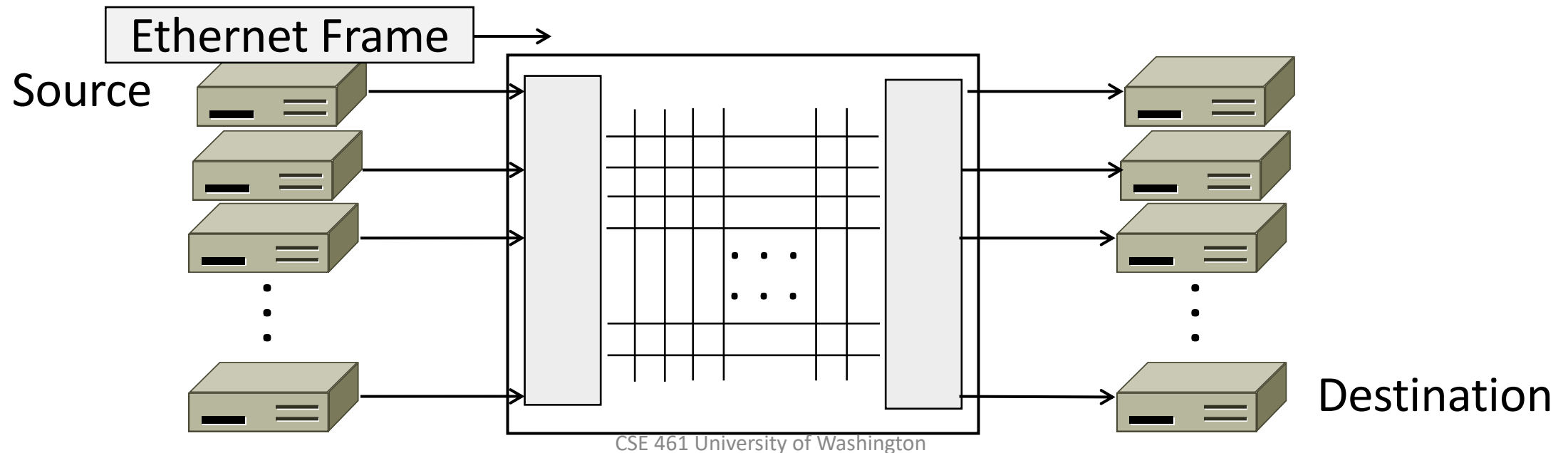
# Switch forwarding

CSE 461

Ratul Mahajan

# Switch Forwarding

- Switch needs to find the right output port for the destination address in the Ethernet frame. How?
  - Link-level, don't look at IP

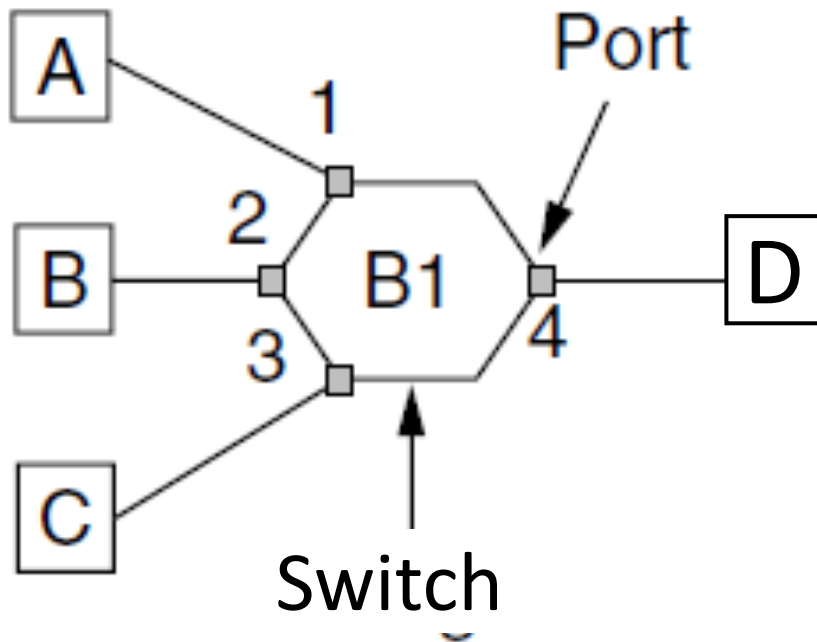


# Backward Learning

- 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)

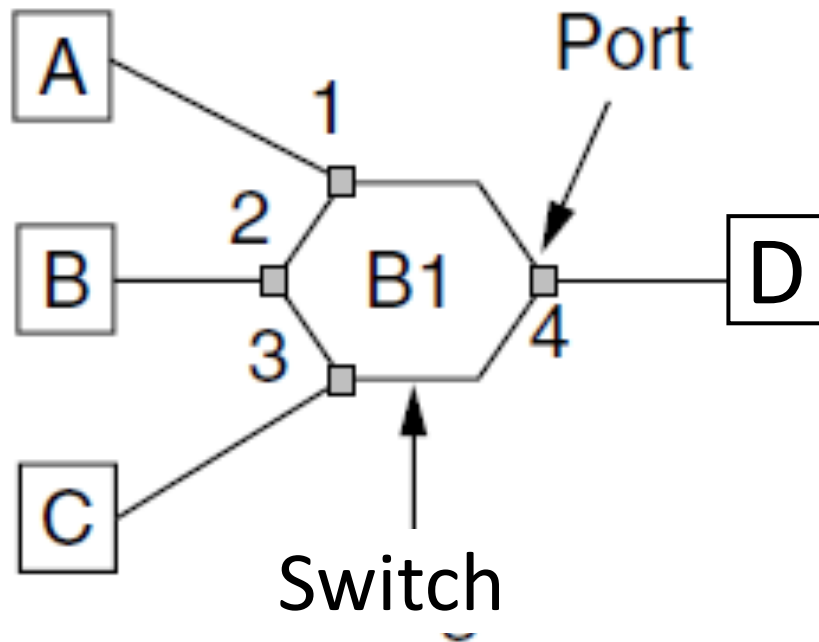
- 1: A sends to D



Address	Port
A	
B	
C	
D	

# Backward Learning (3)

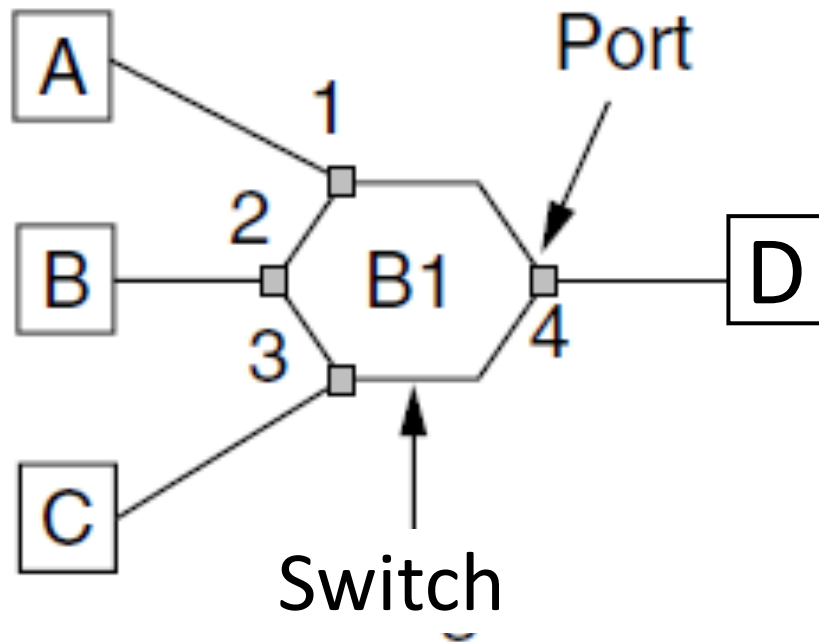
- 2: D sends to A



Address	Port
A	1
B	
C	
D	

# Backward Learning (4)

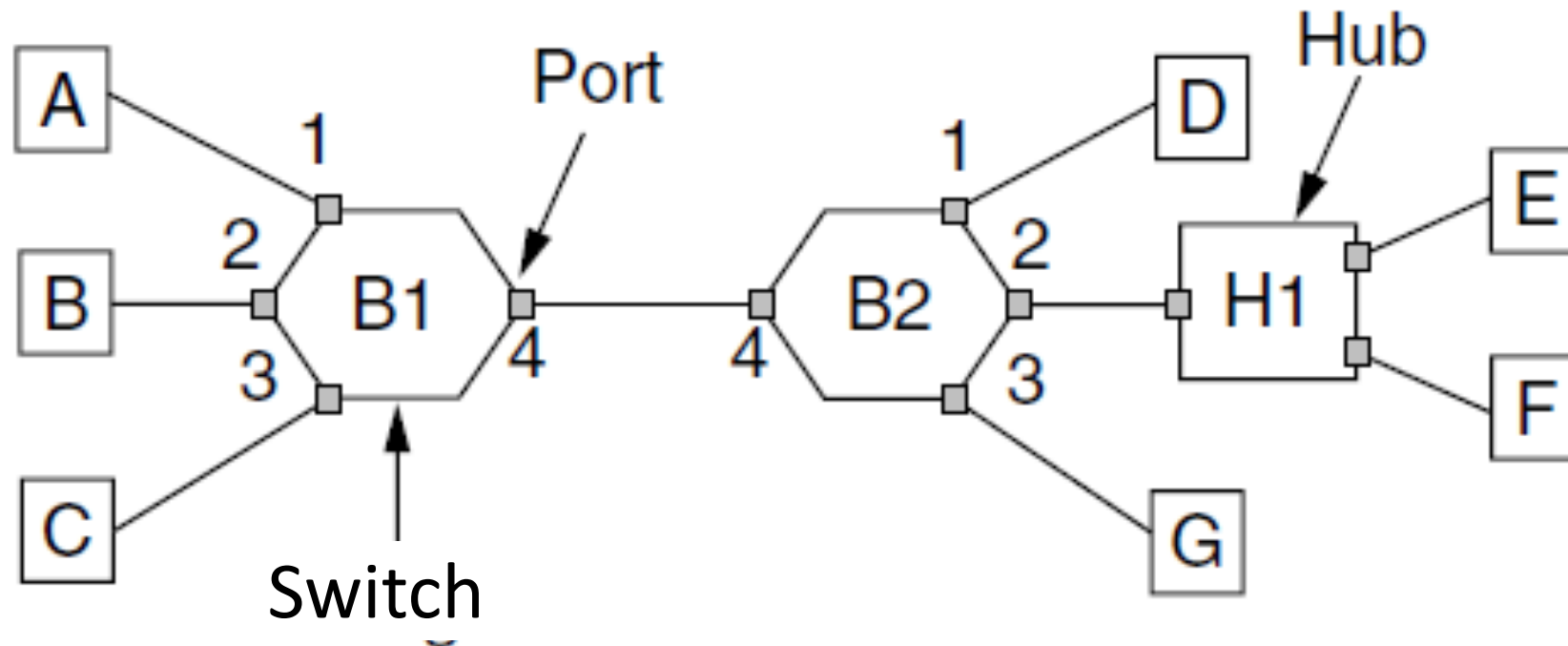
- 3: A sends to D



Address	Port
A	1
B	
C	
D	4

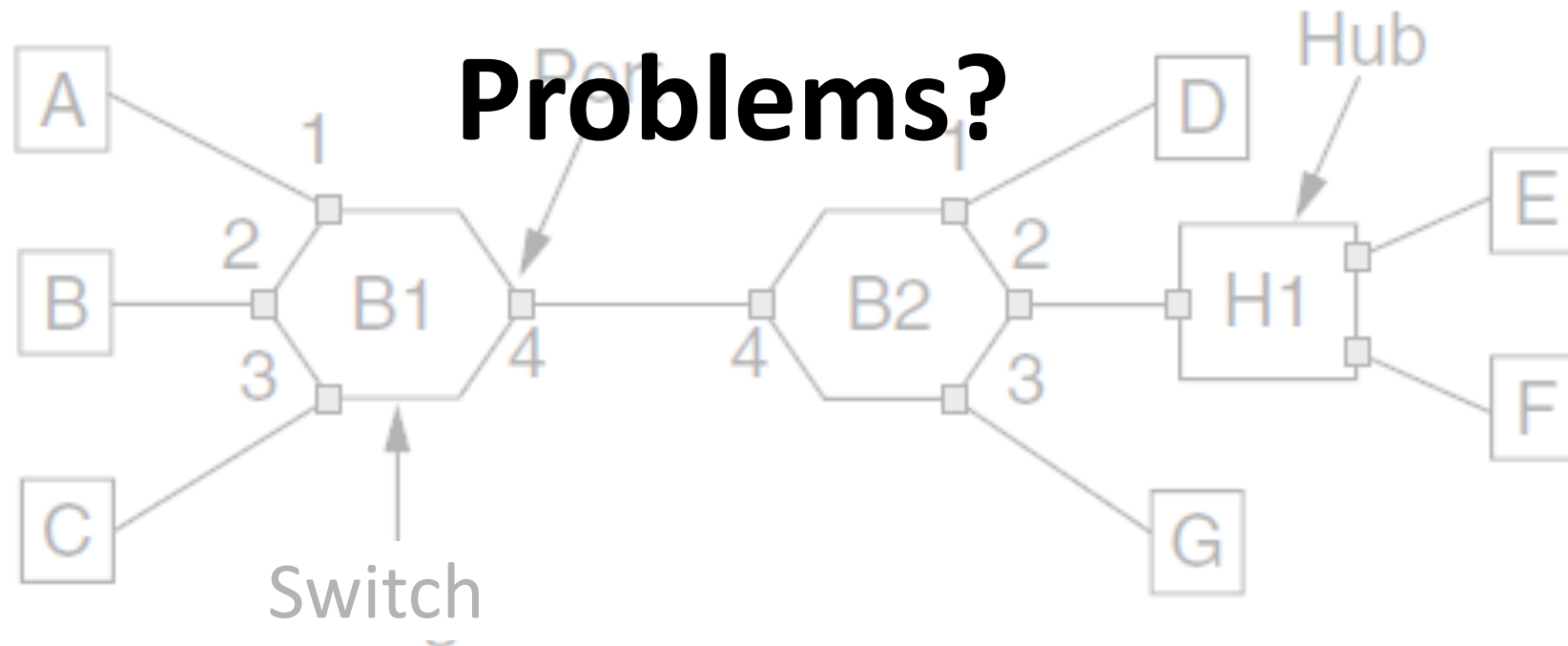
# Learning with Multiple Switches

- Just works with multiple switches and a mix of hubs, e.g., A -> D then D -> A



# Learning with Multiple Switches

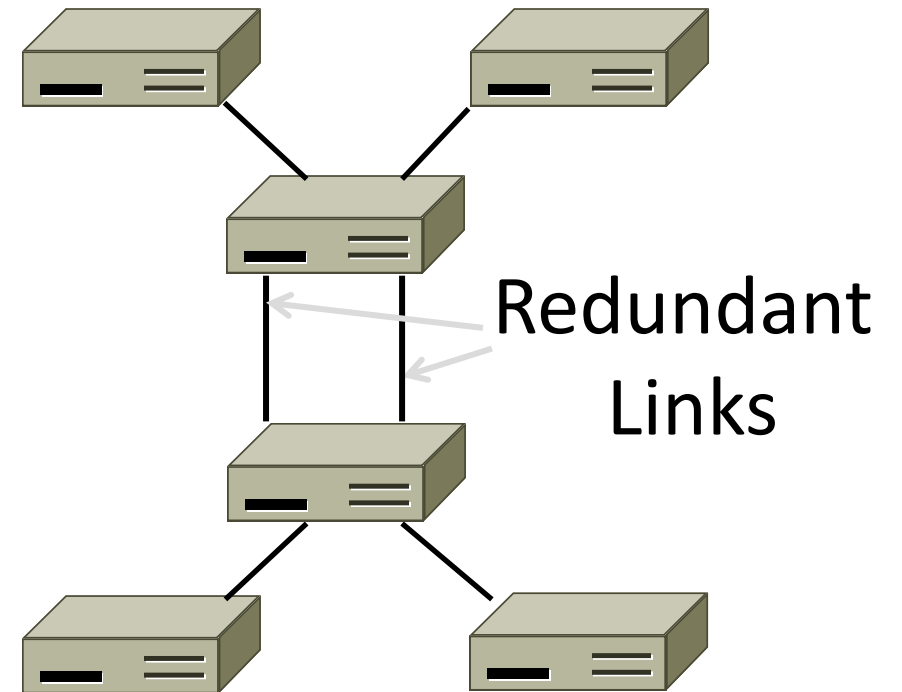
- Just works with multiple switches and a mix of hubs, e.g., A -> D then D -> A





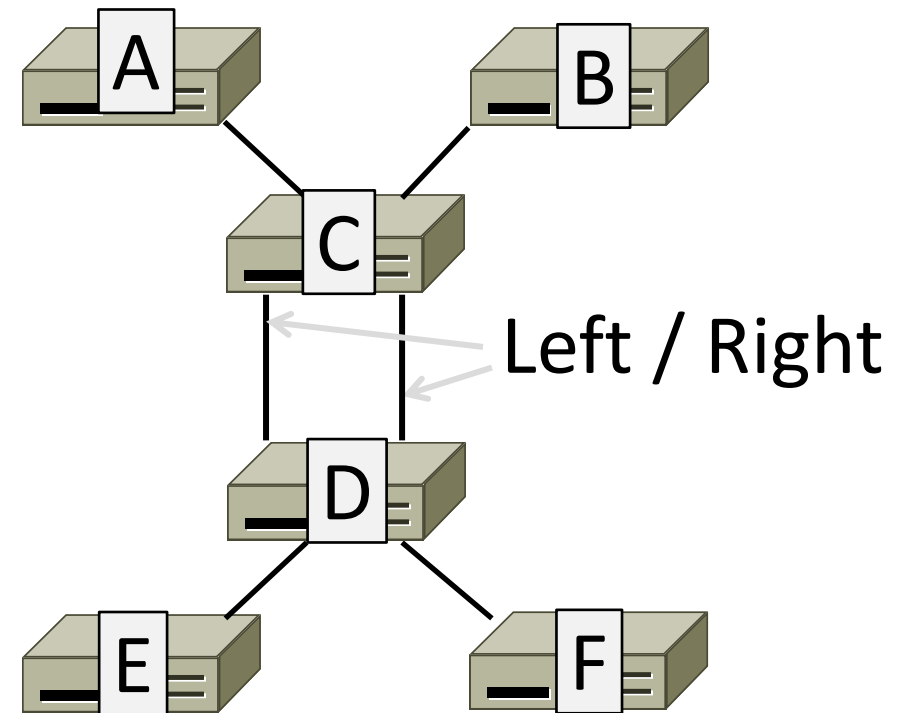
# Problem – Forwarding Loops

- 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 ...



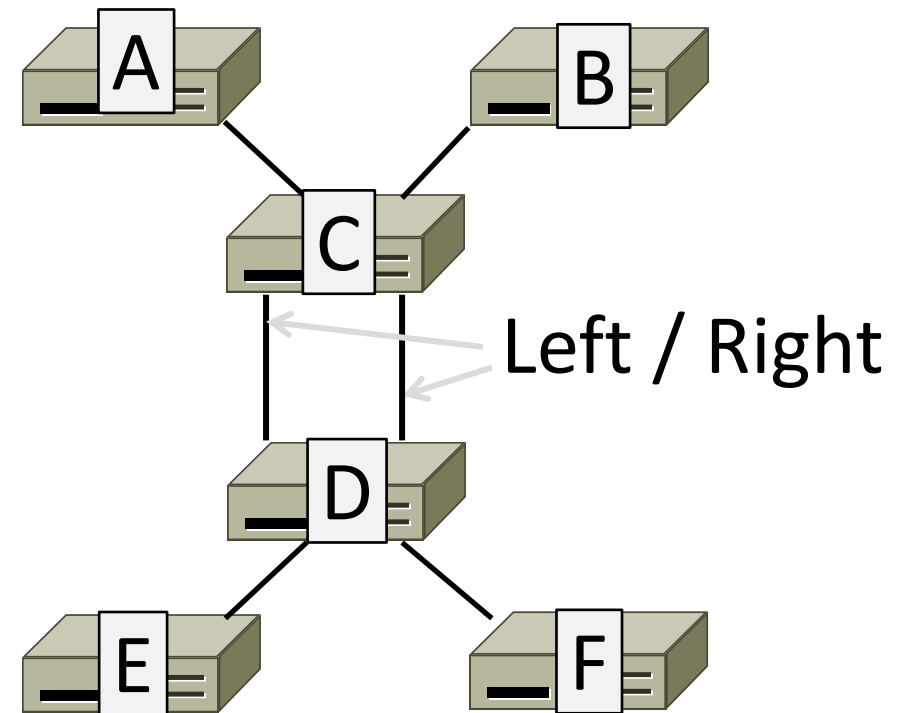
## Forwarding Loops (2)

- 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 → ...

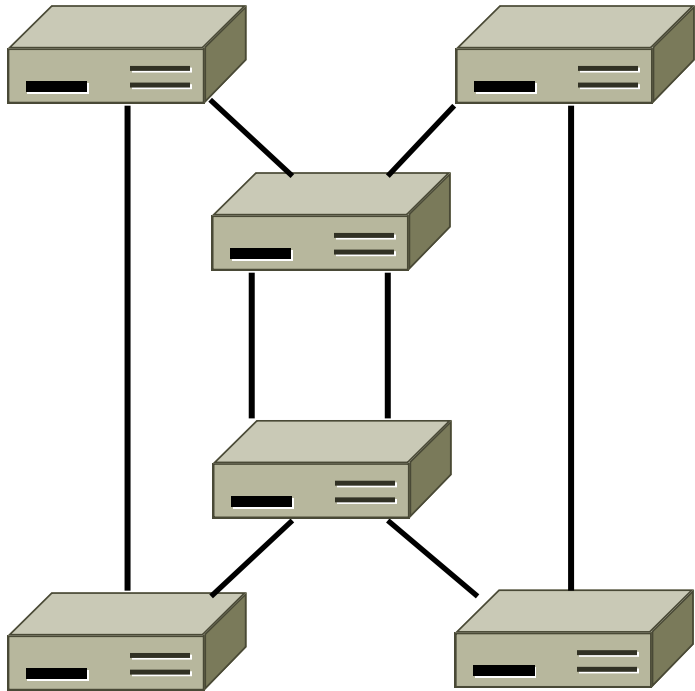


# Spanning Tree Solution

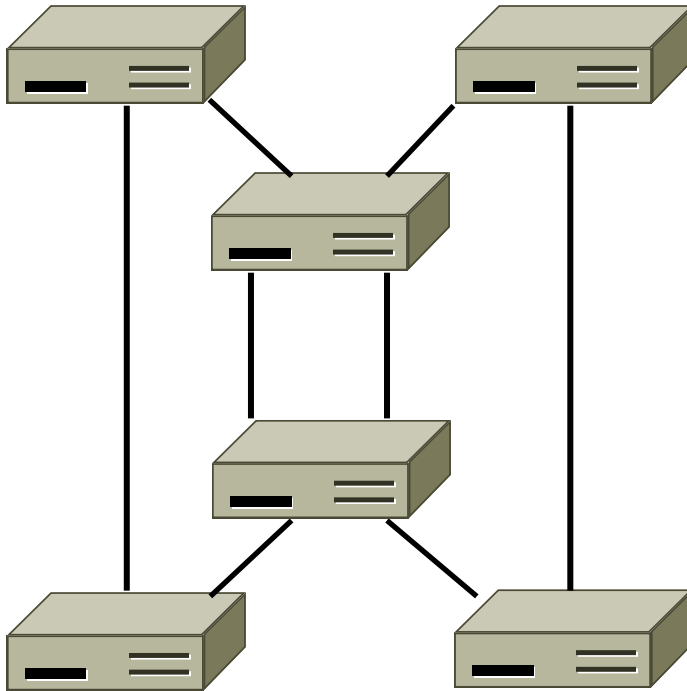
- Switches collectively find a spanning tree for the topology
  - A subset of links that is a tree (no loops) and reaches all switches
  - They switches forward as normal on the spanning tree
  - Broadcasts will go up to the root of the tree and down all the branches

# Spanning Tree (2)

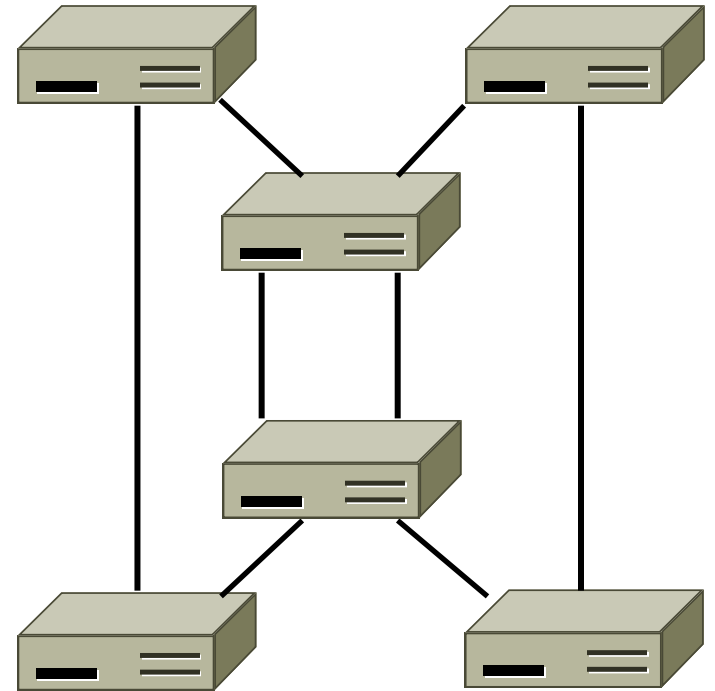
Topology



One ST

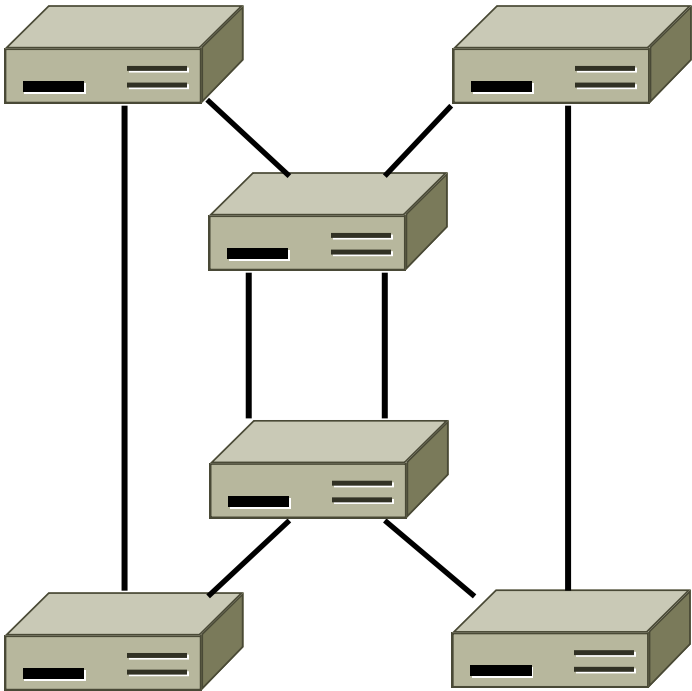


Another ST

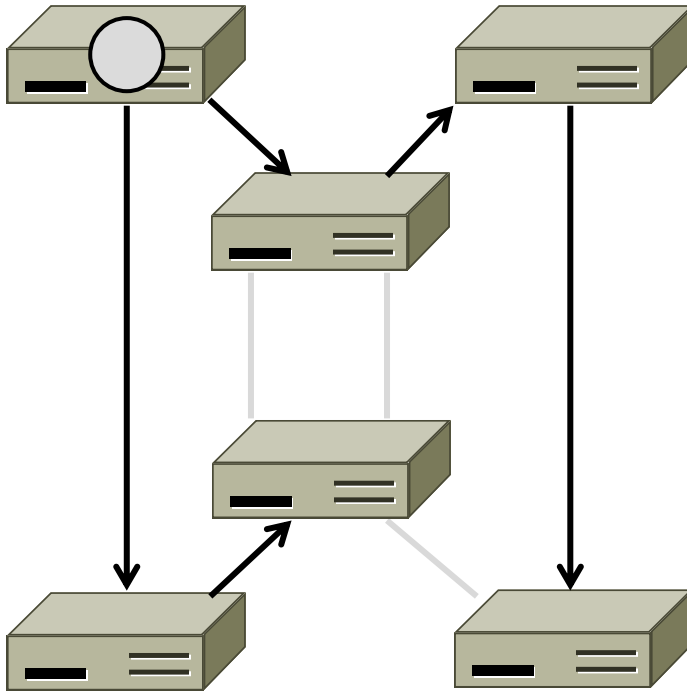


# Spanning Tree (3)

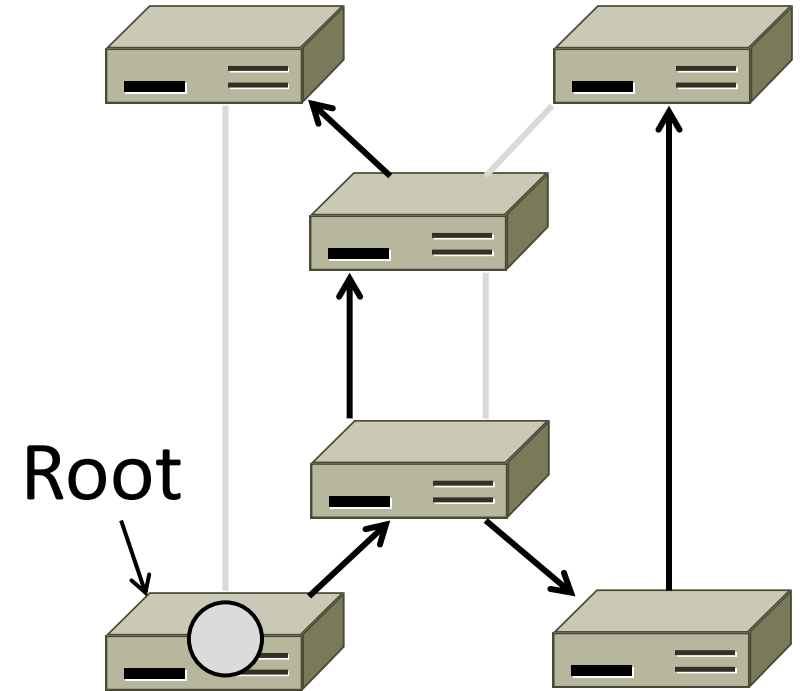
Topology



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)
- 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:

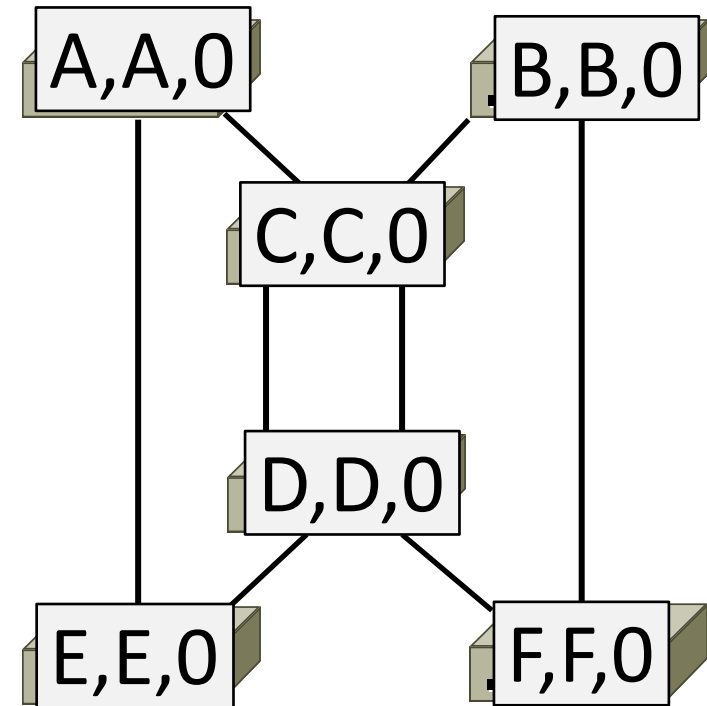
- Each switch initially believes it is the root of the tree
- Each switch sends periodic updates to neighbors with:
  - Its address, address of the root, and distance (in hops) to root
  - Short-circuit when topology changes
- Switches favors ports with shorter distances to lowest root
  - Uses lowest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

Hi, I'm C, the root is A, it's 2 hops away or (C, A, 2)



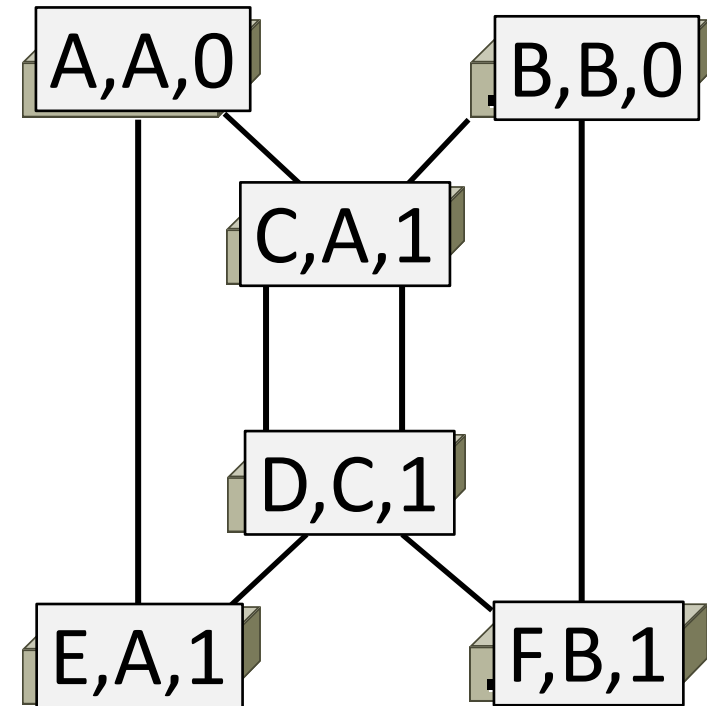
# Spanning Tree Example

- 1<sup>st</sup> round, sending:
  - A sends (A, A, 0) to say it is root
  - B, C, D, E, and F do likewise
- 1<sup>st</sup> 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)



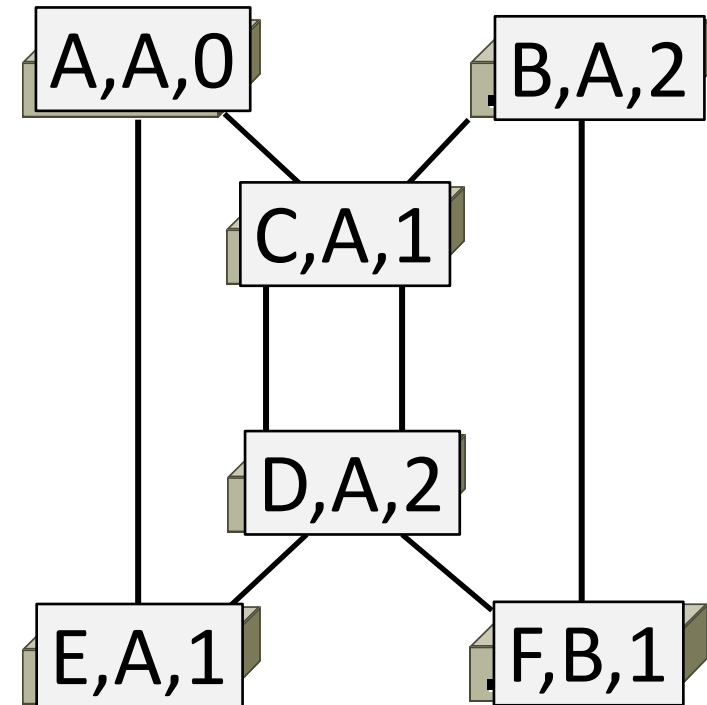
# Spanning Tree Example (2)

- 2<sup>nd</sup> round, sending
  - Nodes send their updated state
- 2<sup>nd</sup> 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)



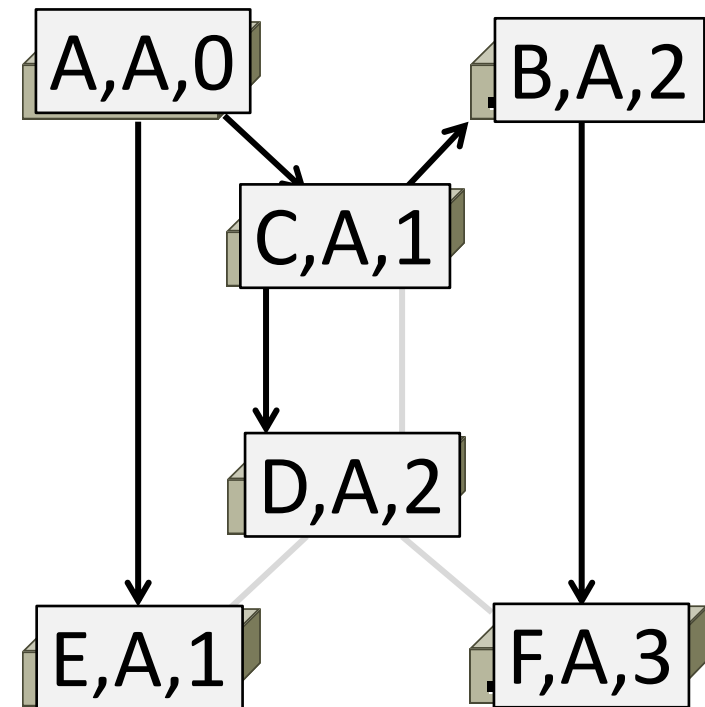
# Spanning Tree Example (3)

- 3<sup>rd</sup> round, sending
  - Nodes send their updated state
- 3<sup>rd</sup> 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



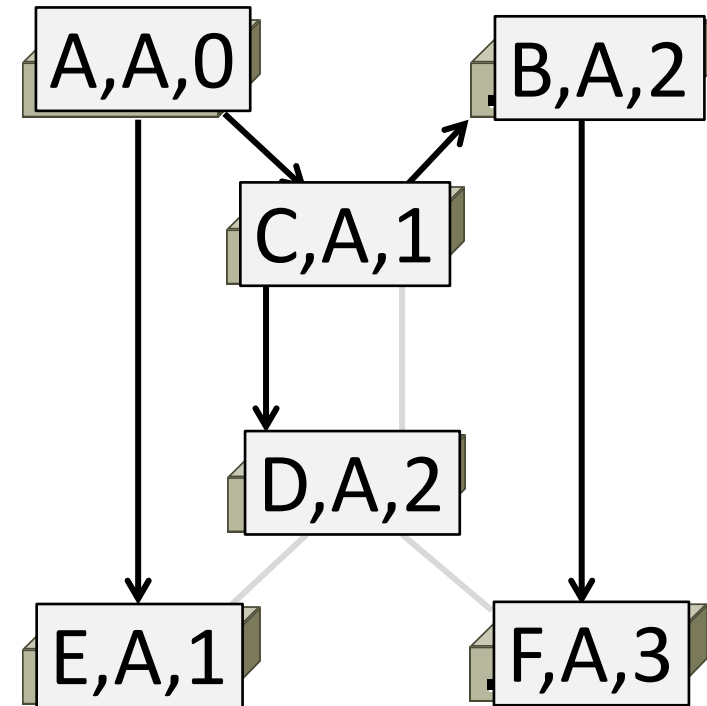
# Spanning Tree Example (4)

- 4<sup>th</sup> 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



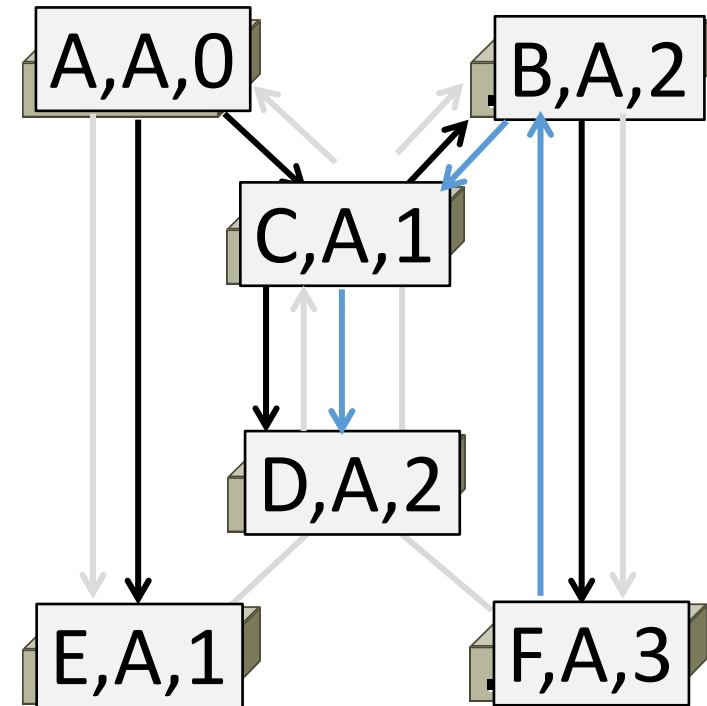
# Spanning Tree Example (5)

- 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:
  - $D \rightarrow C$ -left
  - $C \rightarrow A, B$
  - $A \rightarrow E$
  - $B \rightarrow F$
- And F sends back to D:
  - $F \rightarrow B$
  - $B \rightarrow C$
  - $C \rightarrow D$

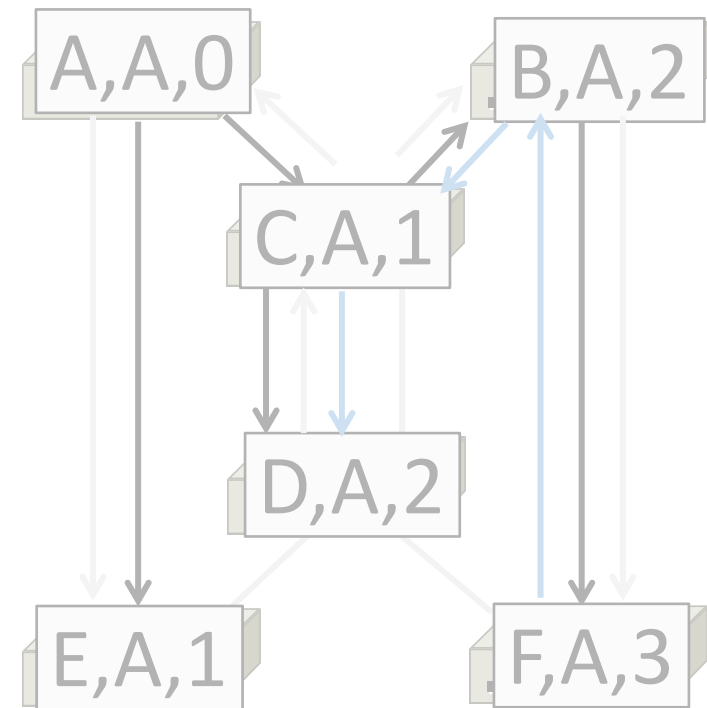




# Spanning Tree Example (6)

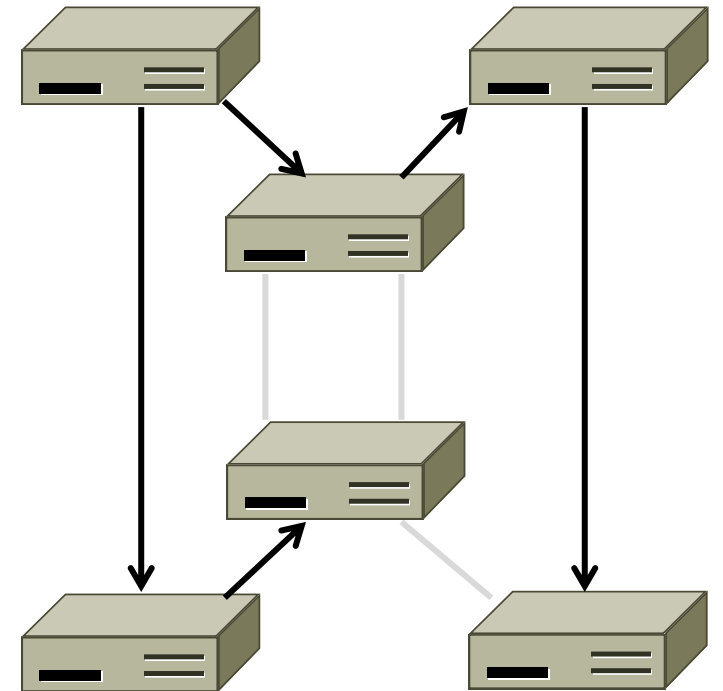
- Forwarding proceeds as usual on the ST
- Initially D sends to F:
  - $D \rightarrow C$ -left
  - $C \rightarrow A, B$
  - $A \rightarrow E$
  - $B \rightarrow F$
- And F sends back to D:
  - $F \rightarrow B$
  - $B \rightarrow C$
  - $C \rightarrow D$

**Problems?**



# Challenges with Switching

- Long paths
- Wasted capacity
- Lack of redundancy



# Switching vs Routing

- Switches are easier to set up – plug and play
  - Routing requires configuration
- Routing scales better
  - Hierarchy, aggregation, subnetting
- Routing uses network resources better

# Algorhyme by Radia Perlman

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.