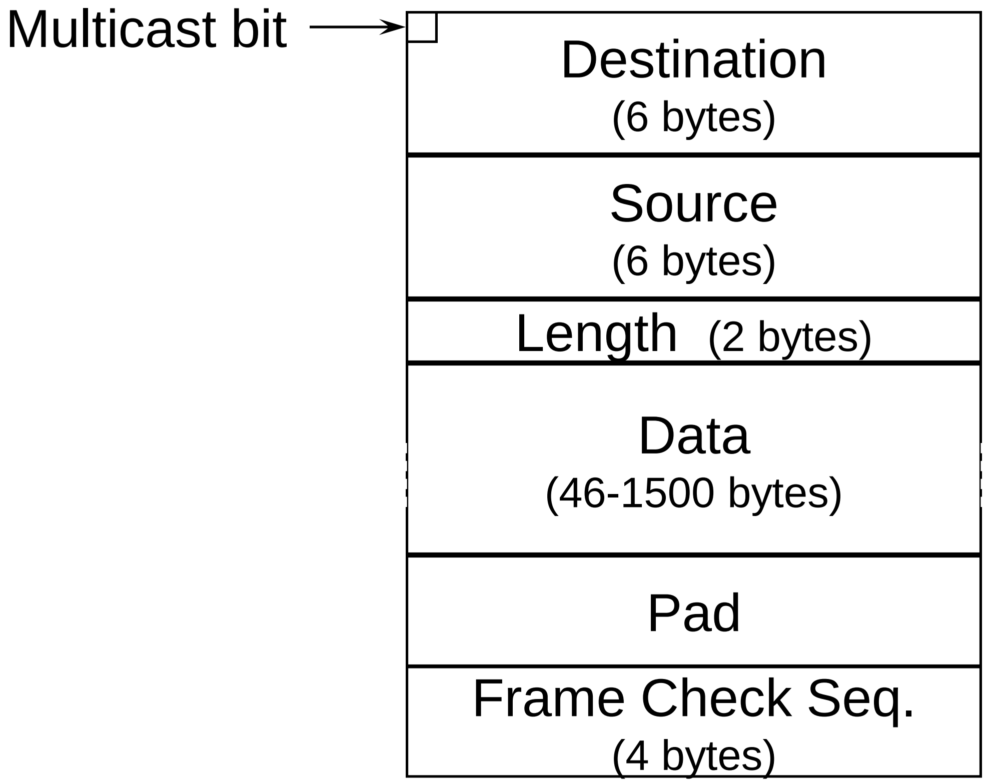
MAC Layer Ethernet Frame Format



Ethernet MAC Frame Address Field

- Destination and Source Addresses:
  - 6 bytes each
- Two types of destination addresses
  - Physical address: Unique for each user
  - Multicast address: Group of users
  - First bit of address determines which type of address is being used

    0 = physical address

    1 = multicast address

Ethernet MAC Frame Other Fields

- Length Field
  - 2 bytes in length
  - determines length of data payload
- Data Field: between 0 and 1500 bytes
- Pad: Filled when Length < 46
- Frame Check Sequence Field
  - 4 bytes
  - Cyclic Redundancy Check (CRC-32)

CSMA/CD

- Recall:
  - CSMA is a “carrier sense” protocol.
    - If channel is idle, transmit immediately
    - If busy, wait until the channel becomes idle
  - CSMA/CD can detect collisions.
    - Abort transmission immediately if there is a collision
    - Try again later according to a backoff algorithm

Ethernet Backoff Algorithm: Binary Exponential Backoff

- If collision,
  - Choose one slot randomly from 2^k slots, where k is the number of collisions the frame has suffered.
  - One contention slot length = 2 x end-to-end propagation delay

This algorithm can adapt to changes in network load.

Binary Exponential Backoff *(cont'd)*

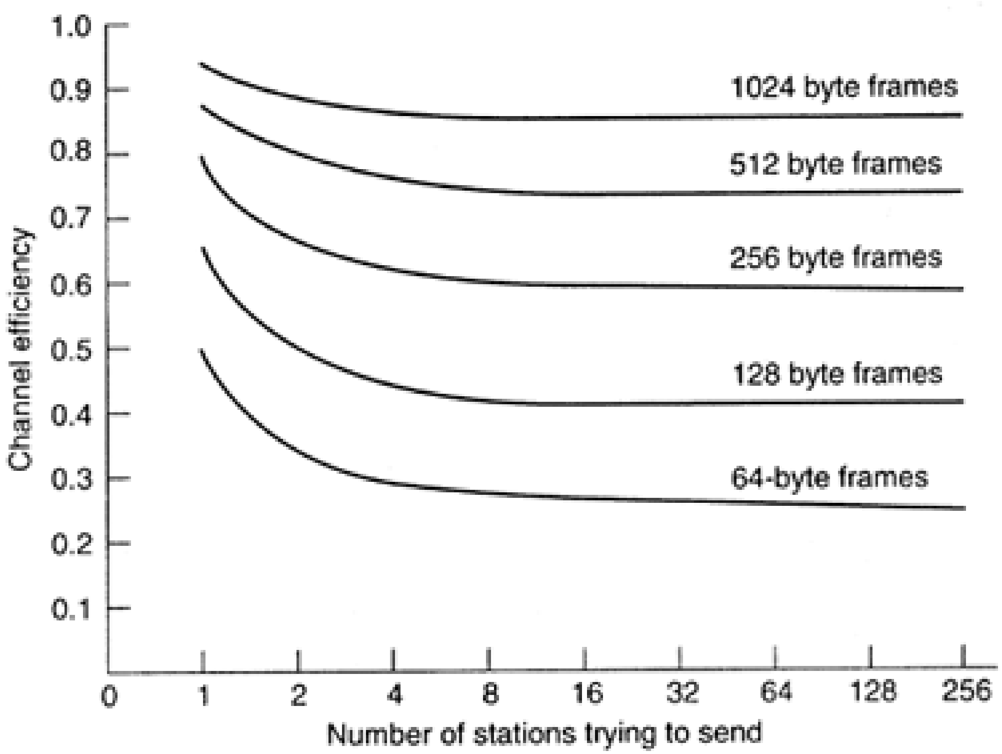
slot length = 2 x end-to-end delay = 50 μs

t=0μs:	Assume A and B collide ($k_A = k_B = 1$)
	A, B choose randomly from 2^1 slots: [0,1]
	Assume A chooses 1, B chooses 1
t=100μs:	A and B collide ($k_A = k_B = 2$)
	A, B choose randomly from 2^2 slots: [0,3]
	Assume A chooses 2, B chooses 0
t=150μs:	B transmits successfully
t=250μs:	A transmits successfully

Binary Exponential Backoff *(cont'd)*

- In Ethernet,
  - Binary exponential backoff will allow a maximum of 15 retransmission attempts
  - If 16 backoffs occur, the transmission of the frame is considered a failure.

Ethernet Performance



Ethernet Features and Advantages

1. Passive interface: No active element
2. Broadcast: All users can listen
3. Distributed control: Each user makes own decision

Simple

Reliable

Easy to reconfigure

Ethernet Disadvantages

- Lack of priority levels
- Security issues

Hubs, Switches, Routers

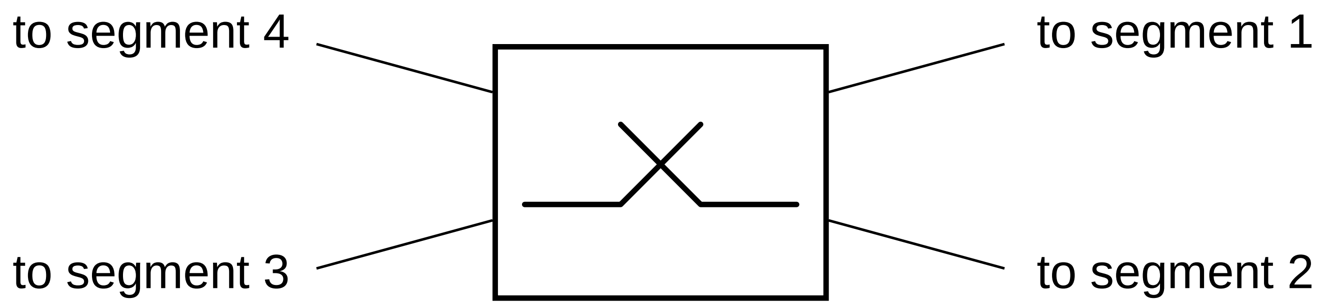
- Hub:
  - Behaves like Ethernet
- Switch:
  - Supports multiple collision domains
  - A collision domain is a segment
- Router: operates on level-3 packets

Why Ethernet Switching?

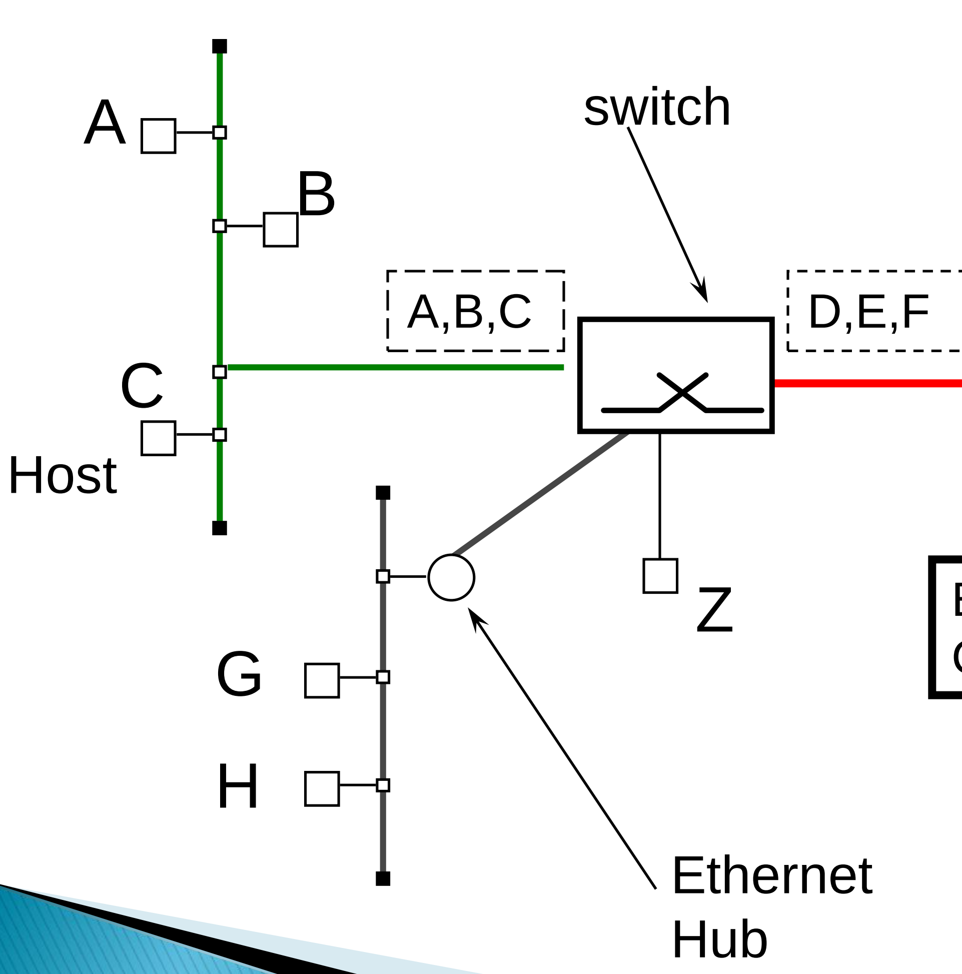
- LANs may grow very large
  - The switch has a very fast backplane
  - It can forward frames very quickly to the appropriate subnet
- Cheaper than upgrading all host interfaces to use a faster network

Ethernet Switching

- Connect many Ethernet through an “Ethernet switch”
- Each Ethernet is a “segment”
- Make one large, logical segment

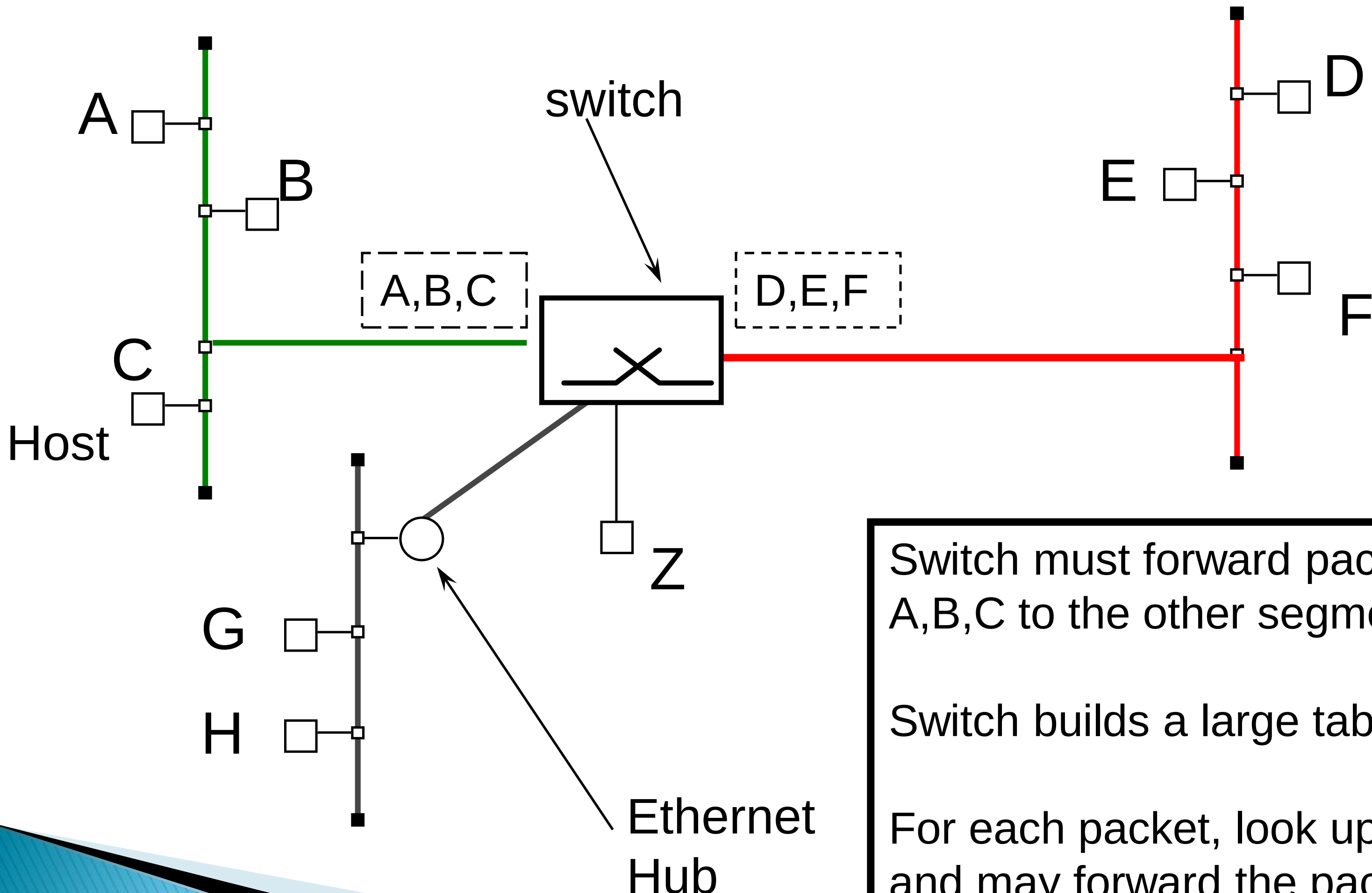


Collision Domains



Each segment runs a standard CSMA protocol

Layer-2 routing tables

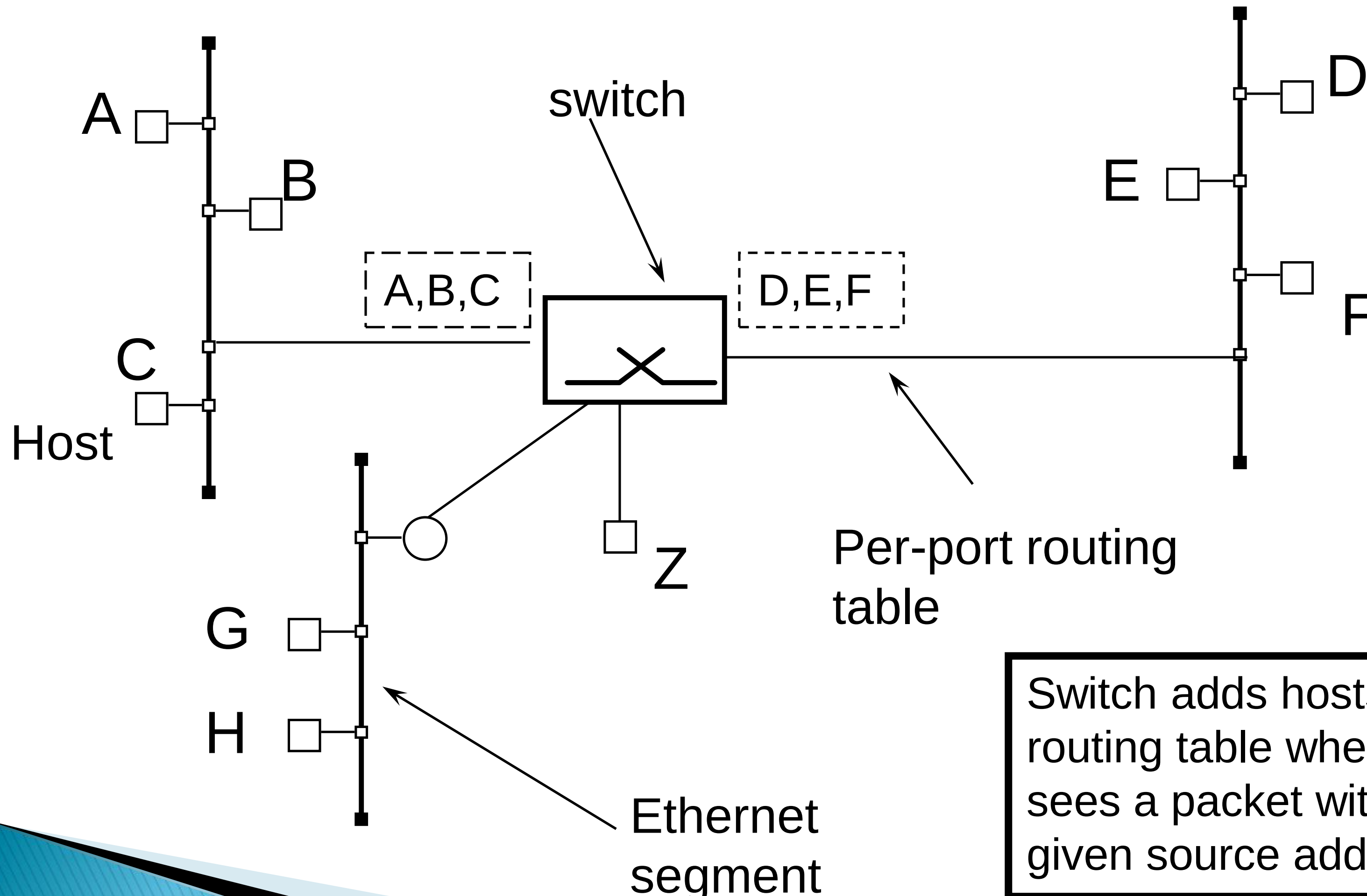


Switch must forward packets from A,B,C to the other segment

Switch builds a large table

For each packet, look up in table and may forward the packet

Learning MAC addresses

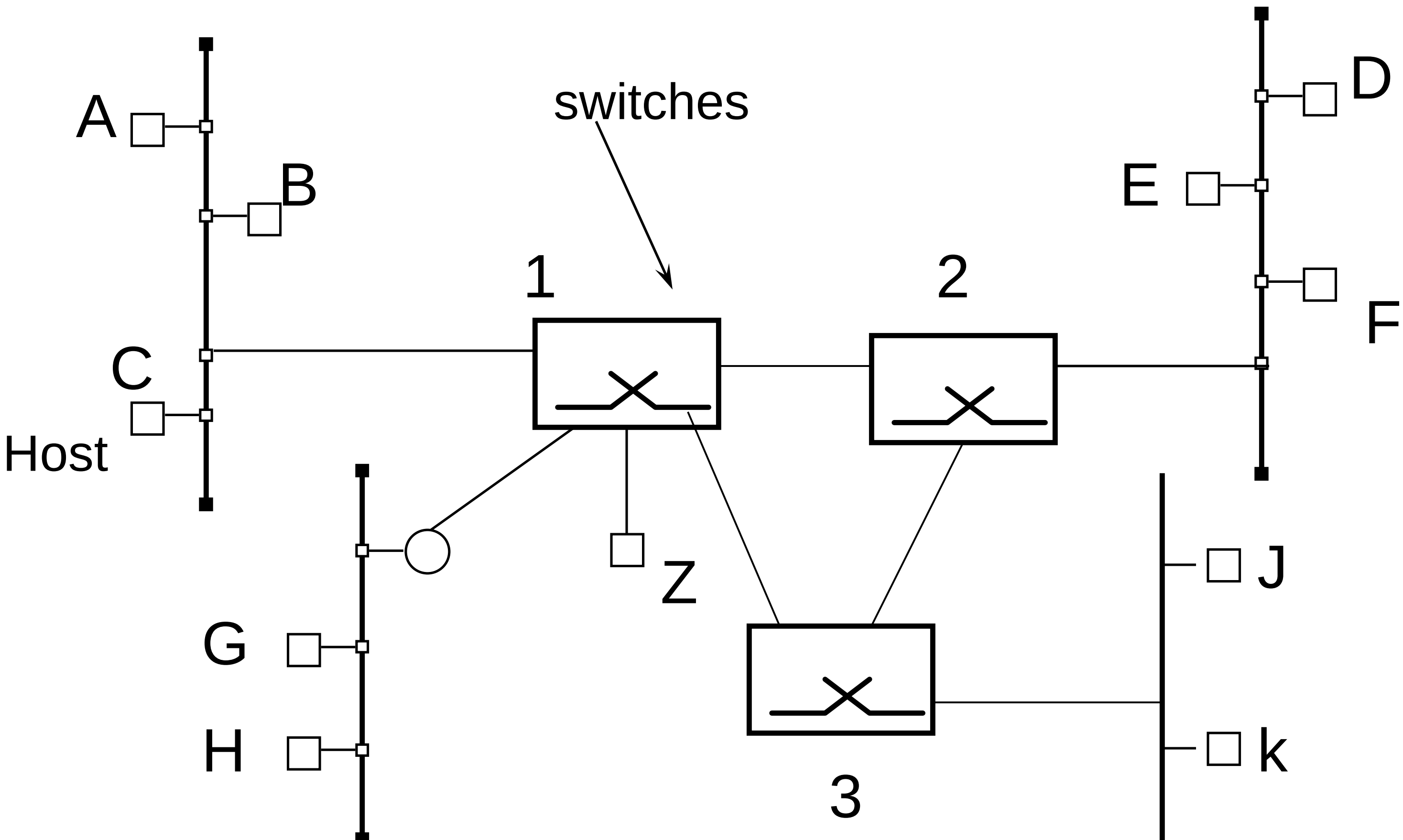


Switch adds hosts to routing table when it sees a packet with a given source address

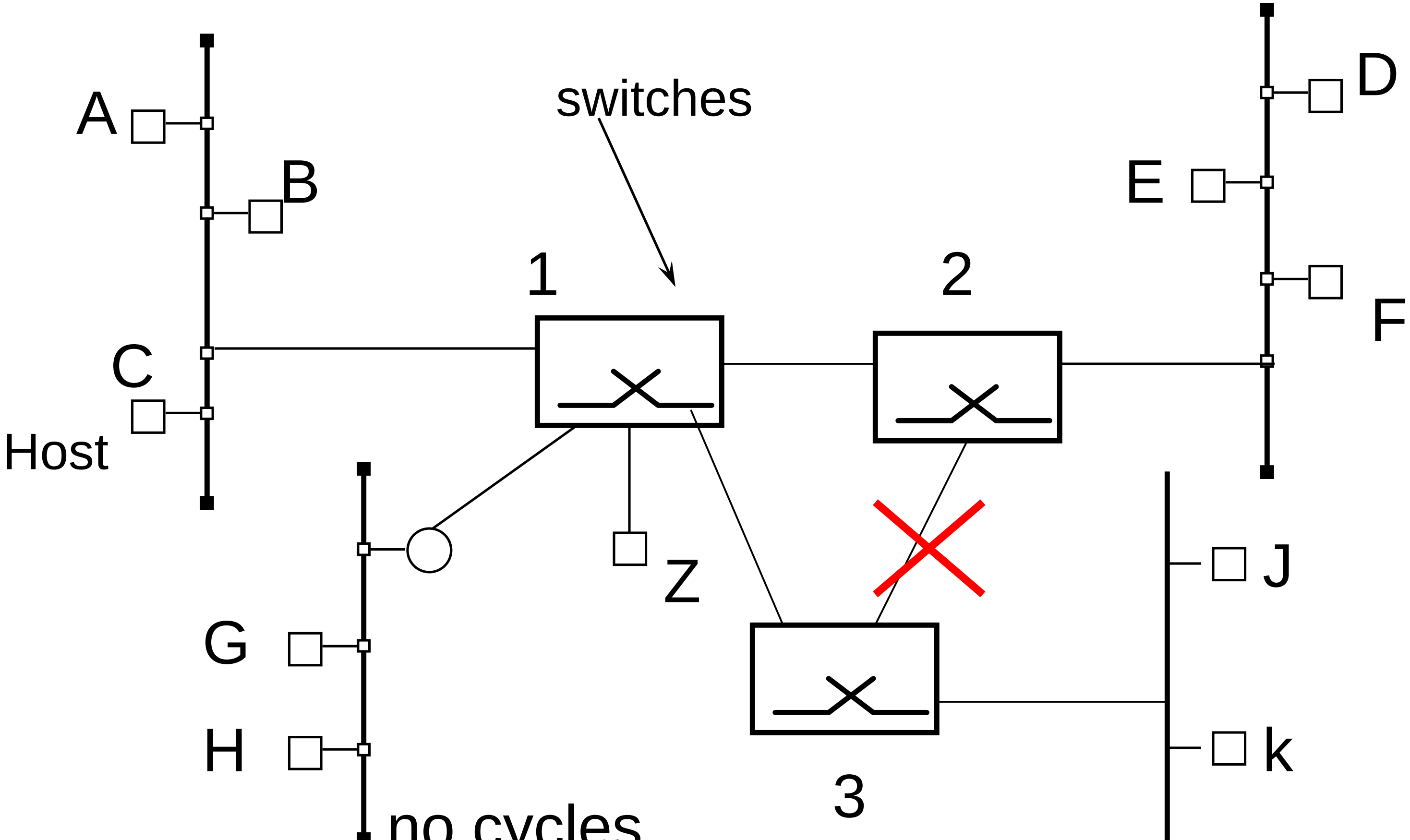
Spanning Trees

- Want to allow multiple switches to connect together
- What If there is a cycle in the graph of switches connected together?
  - Can’t have packets circulate forever!
  - Must break the cycle by restricting routes

Spanning Trees



Spanning Trees



no cycles
in the graph of switches

Spanning Tree Protocol

1. Each switch periodically sends a configuration message out of every port. A message contains: (ID of sender, ID of root, distance from sender to root).
2. Initially, every switch claims to be root and sends a distance field of 0.
3. A switch keeps sending the same message (periodically) until it hears a “better” message.
4. “Better” means:
   - A root with a smaller ID
   - A root with equal ID, but with shorter distance
   - The root ID and distance are the same as we already have, but the sending bridge has a smaller ID.
5. When a switch hears a better configuration message, it stops generating its own messages, and just forwards ones that it receives (adding 1 to the distance).
6. If the switch realizes that it is not the designated bridge for a segment, it stops sending configuration messages to that segment.

Eventually:

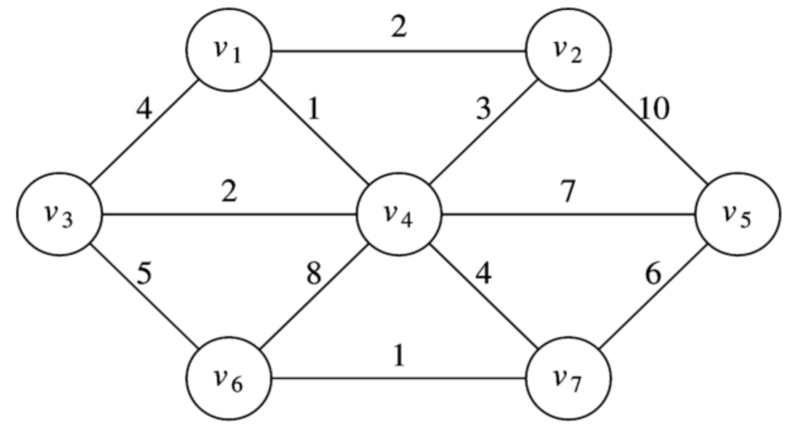
- Only the root switch generates configuration messages,
- Other switches send configuration messages to segments for which they are the designated switch

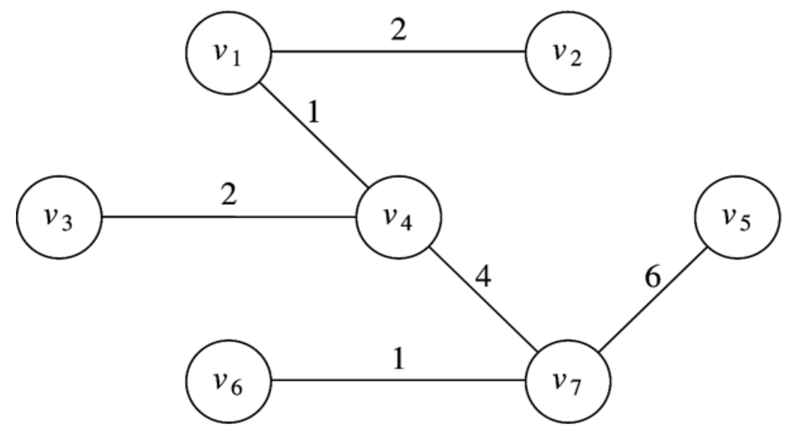
Minimum Spanning Tree

- Focus on minimum spanning tree in an undirected graph.
  - Finding minimum spanning tree in a directed graph is harder.
- Definition: A Tree formed from graph edges that connects all the vertices of G at lowest total cost.
- A minimum spanning tree exists if and only if the graph is connected.

Minimum Spanning Tree (Example)

- The number of edges in the minimum spanning tree is V-1. (Why?)
- Minimum spanning tree:
  - A tree: acyclic
  - Spanning: covers every vertex
  - Minimum: lowest total cost





Prim's Algorithm (Minimum Spanning Tree)

- Grow the tree in successive stages:
  - At each stage, one node is picked as the root;
  - add an edge and an associated vertex with the lowest cost to the tree.
    - Rule to add: a new vertex v to add to the tree by choosing the edge (u, v) such that the cost of (u, v) is the smallest among all the edges where u is in the tree and v is not.
    - Rule to update: for each unknown vertex w adjacent to v, update $d_w = \min(d_w, c_{wv})$.
- Very similar to Dijkstra's algorithm,
  - But use on undirected graphs.
  - Different node as the root at different stages.

Prim's Algorithm (Example)

1. Initial configuration
2. After v1 is declared known

v	*known*	d_v	p_v
v_1	F	0	0
v_2	F	∞	0
v_3	F	∞	0
v_4	F	∞	0
v_5	F	∞	0
v_6	F	∞	0
v_7	F	∞	0

v	*known*	d_v	p_v
v_1	T	0	0
v_2	F	2	v_1
v_3	F	4	v_1
v_4	F	1	v_1
v_5	F	∞	0
v_6	F	∞	0
v_7	F	∞	0

Prim's Algorithm (Example Cont.)

3. After v4 is declared known
4. After v2 and then v3 are declared known

v	*known*	d_v	p_v
v_1	T	0	0
v_2	F	2	v_1
v_3	F	2	v_4
v_4	T	1	v_1
v_5	F	7	v_4
v_6	F	8	v_4
v_7	F	4	v_4

v	*known*	d_v	p_v
v_1	T	0	0
v_2	T	2	v_1
v_3	T	2	v_4
v_4	T	1	v_1
v_5	F	7	v_4
v_6	F	5	v_3
v_7	F	4	v_4

Prim's Algorithm (Example Cont.)

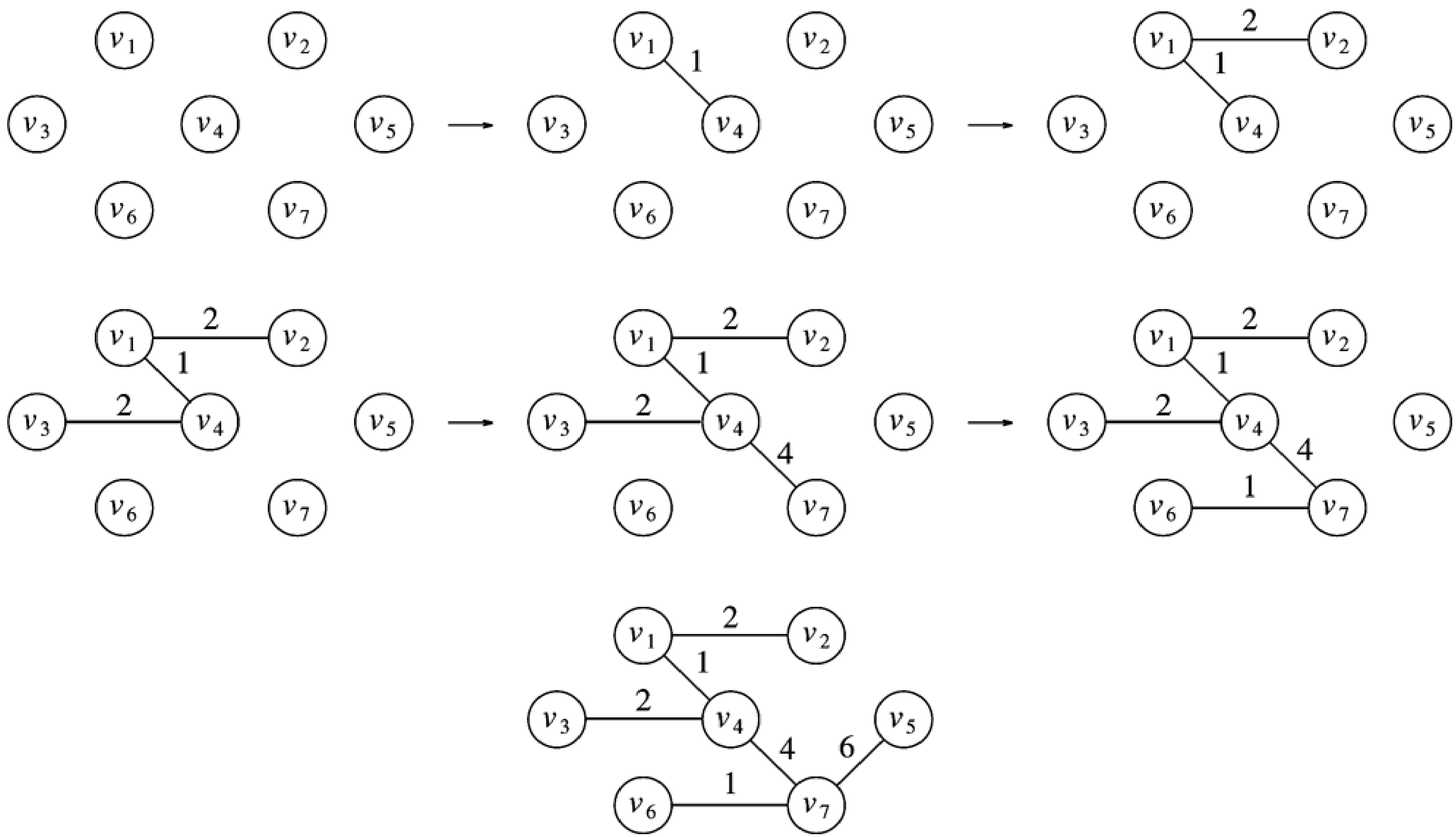
5. After v7 is declared known
6. After v6 and then v5 are declared known

v	*known*	d_v	p_v
v_1	T	0	0
v_2	T	2	v_1
v_3	T	2	v_4
v_4	T	1	v_1
v_5	F	6	v_7
v_6	F	1	v_7
v_7	T	4	v_4

v	*known*	d_v	p_v
v_1	T	0	0
v_2	T	2	v_1
v_3	T	2	v_4
v_4	T	1	v_1
v_5	T	6	v_7
v_6	T	1	v_7
v_7	T	4	v_4

Prim's Algorithm (Example Cont.)

Summary:



Kruskal’s Algorithm (Minimum Spanning Tree)

- Maintains a forest – a collection of trees
  - Initially, there are V single-node trees.
  - Adding an edge merges two trees into one.
  - When there is only one tree, the algorithm terminates -> Minimum Spanning Tree.
- Rules to select edges (simple and efficient)
  - The smallest from the remaining edges
  - Accept an edge only if it does not cause a cycle.

Kruskal's Algorithm (Example)

Edge	Weight	Action
(v_1, v_4)	1	Accepted
(v_6, v_7)	1	Accepted
(v_1, v_2)	2	Accepted
(v_3, v_4)	2	Accepted
(v_2, v_4)	3	Rejected
(v_1, v_3)	4	Rejected
(v_4, v_7)	4	Accepted
(v_3, v_6)	5	Rejected
(v_5, v_7)	6	Accepted

Kruskal's Algorithm (Example)