CS4450

Computer Networks: Architecture and Protocols

Lecture 8
Switched Ethernet
Spanning Tree Protocol

Rachit Agarwal

Goals for Today's Lecture

- **Bring us back into our love for computer networks (and me?) …**
- **Experience (the beauty of) Spanning Tree Protocol**
- **Why** do we need network layer?
    - **Why** not just use switched Ethernet across the Internet?

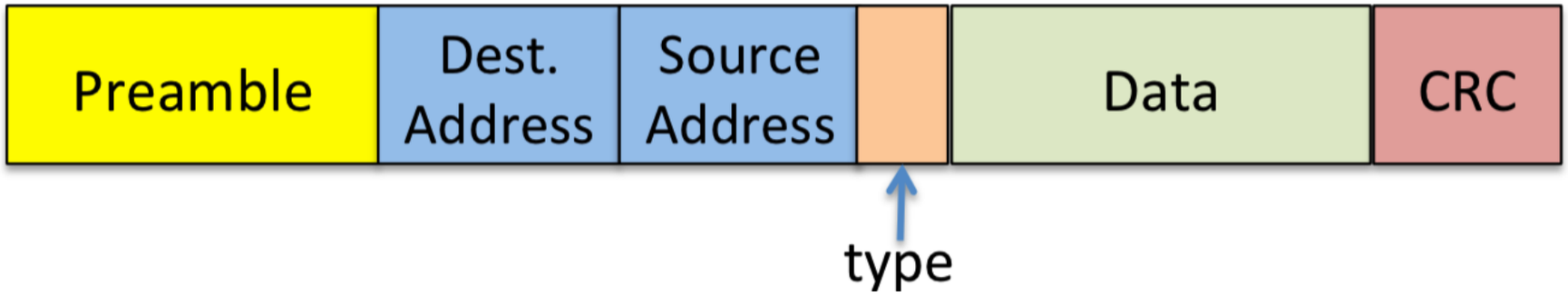
Recap: CSMA/CD

- **Carrier Sense: continuously listen to the channel**
    - If idle: start transmitting
    - If busy: wait until idle

- **Collision Detection: listen while transmitting**
    - No collision: transmission complete
    - Collision: abort transmission

- **When to retransmit?: exponential back off**
    - After collision, transmit after “waiting time”
    - After k collisions, choose “waiting time” from {0, …, 2^k-1)
    - Exponentially increasing waiting times
    - But also, exponentially larger success probability

Recap: Ethernet “Frames”



- **Preamble:**
  - 7 bytes for clock synchronization
  - 1 byte to indicate start of the frame
- **Names:** 6 + 6 bytes (MAC names/addresses)
- **Protocol type:** 2 bytes, indicating higher layer protocol (e.g., IP)
- **Data payload:** max 1500 bytes, minimum 46 bytes
- **CRC:** 4 bytes for error detection

Recap: Why Frames?

- Physical layer sends/receives bits on a link, and forwards to link layer

- Data link layer **interfaces** with **physical layer** using **frames**
    - Implemented by the network adaptor

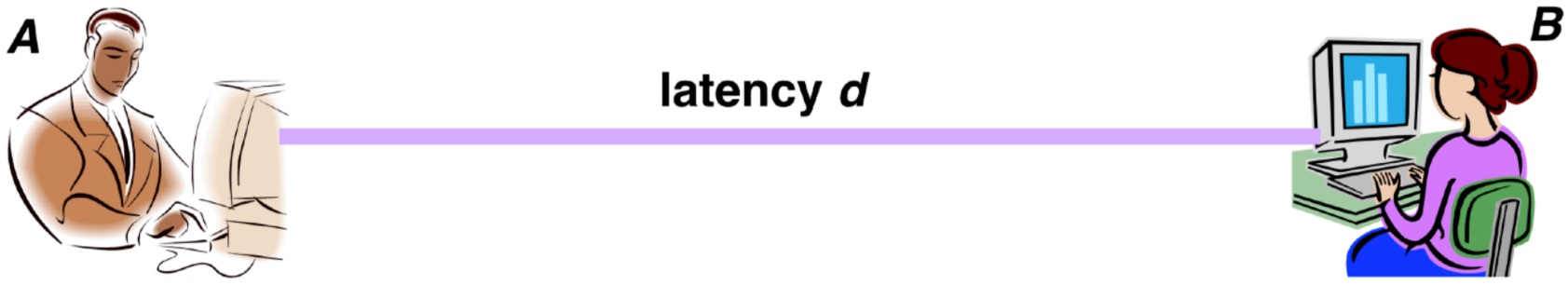
- **Frames are reorganization of packets from the network layer to:**
    - Incorporate sentinel bits for identifying frame start/end
    - Incorporate link layer source and destination names
    - Incorporate CRC for checking correctness of received frames

Recap: Traditional Ethernet

- At the Source:
    - Link layer receives packets from the network layer (more later)
    - Link layer converts packets into frames
    - Link layer passes the frame to physical layer

- At **EACH** destination:
    - Physical layer collects the bits
    - And passes them to the link layer
    - **Link layer regenerates the frame…**
        - Sends the packet to the network layer …. **If and only if:**
            - destination name matches the receiver's MAC name
            - Or, the destination name is the broadcast address (FF:FF:FF:FF:FF:FF)

Recap: Why Switched Ethernet?



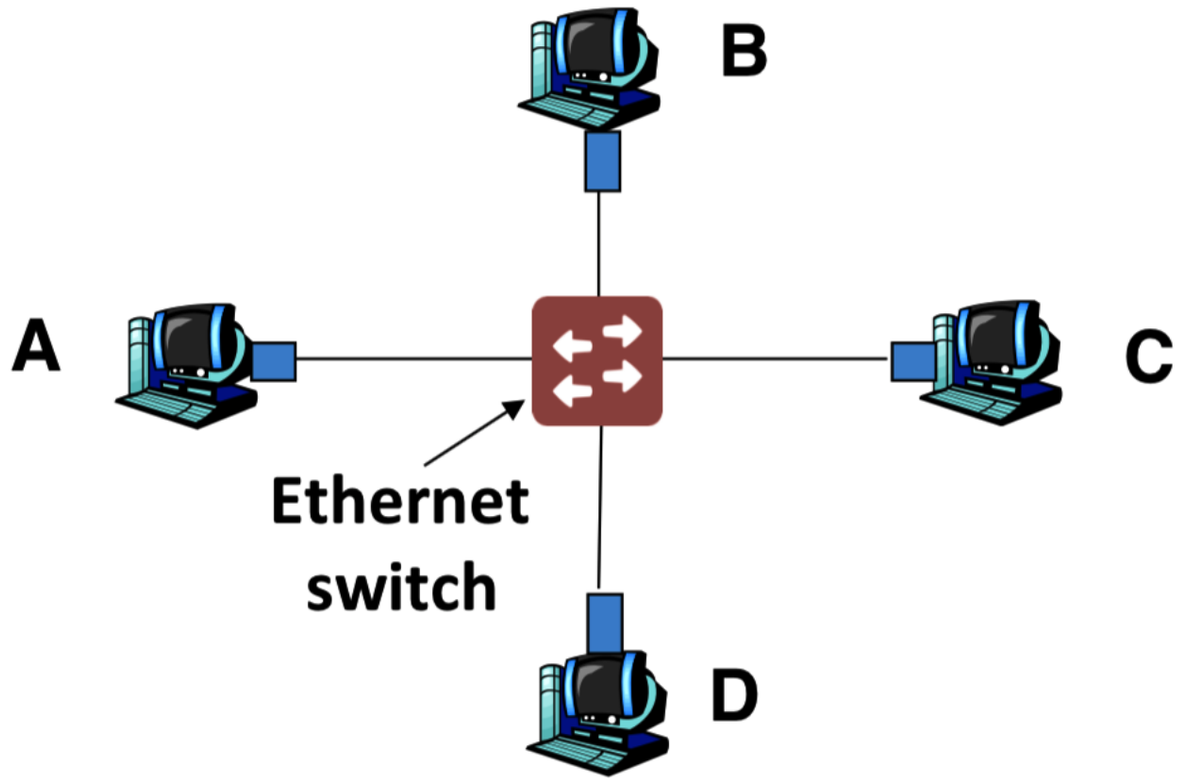
- **Transmission time > 2 * propagation delay**

- Requires either very large frames (underutilization) or small scale

- Cannot scale beyond ~76.8m for 1Gbps and beyond ~7.68m for 10Gbps

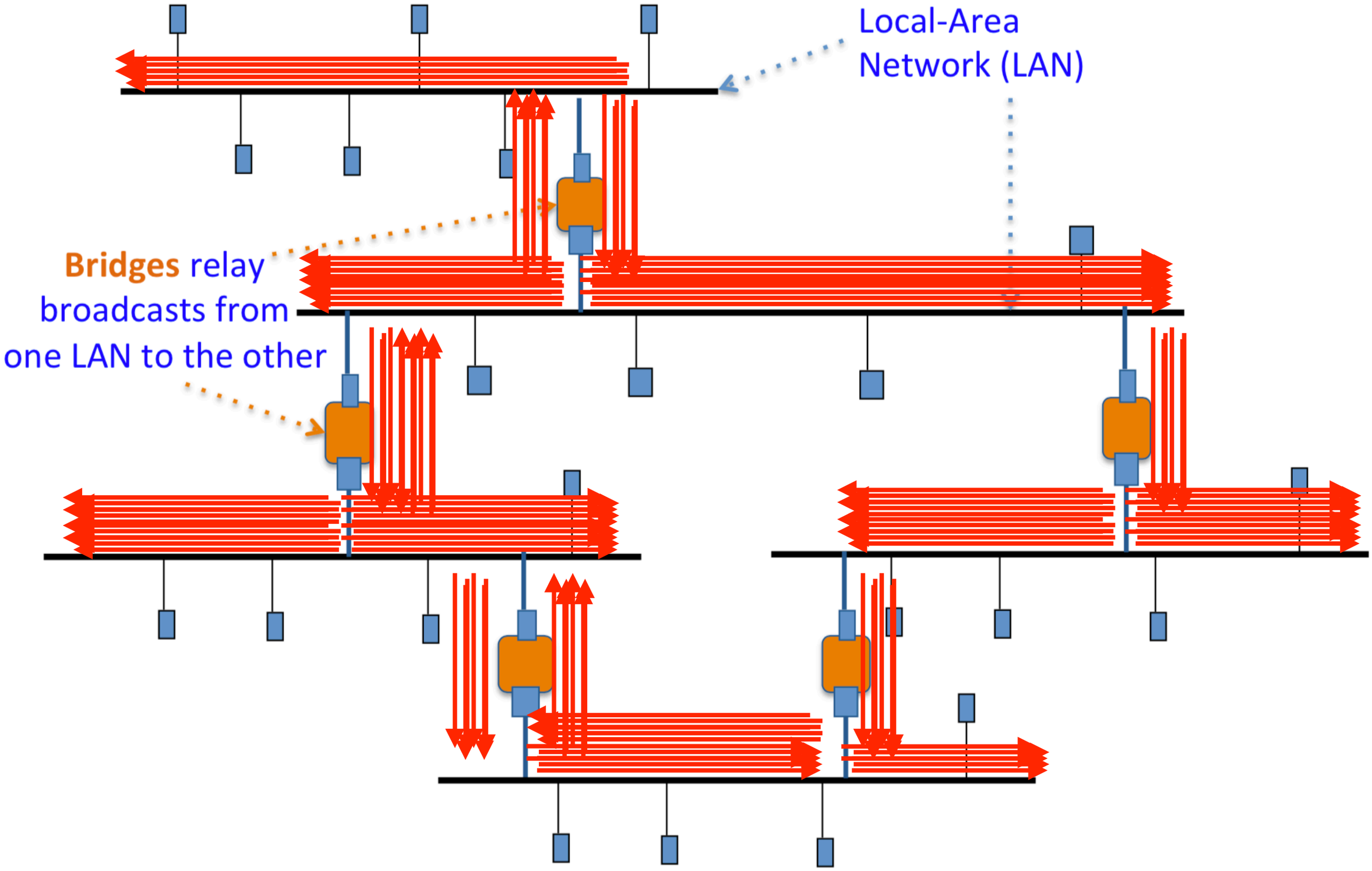
- This is WHY modern Ethernet networks are “switched”

Recap: Switched Ethernet



- Enables concurrent communication
    - Host A can talk to C, while B talks to D
    - No collisions -> no need for CSMA, CD
    - No constraints on link lengths or frame size

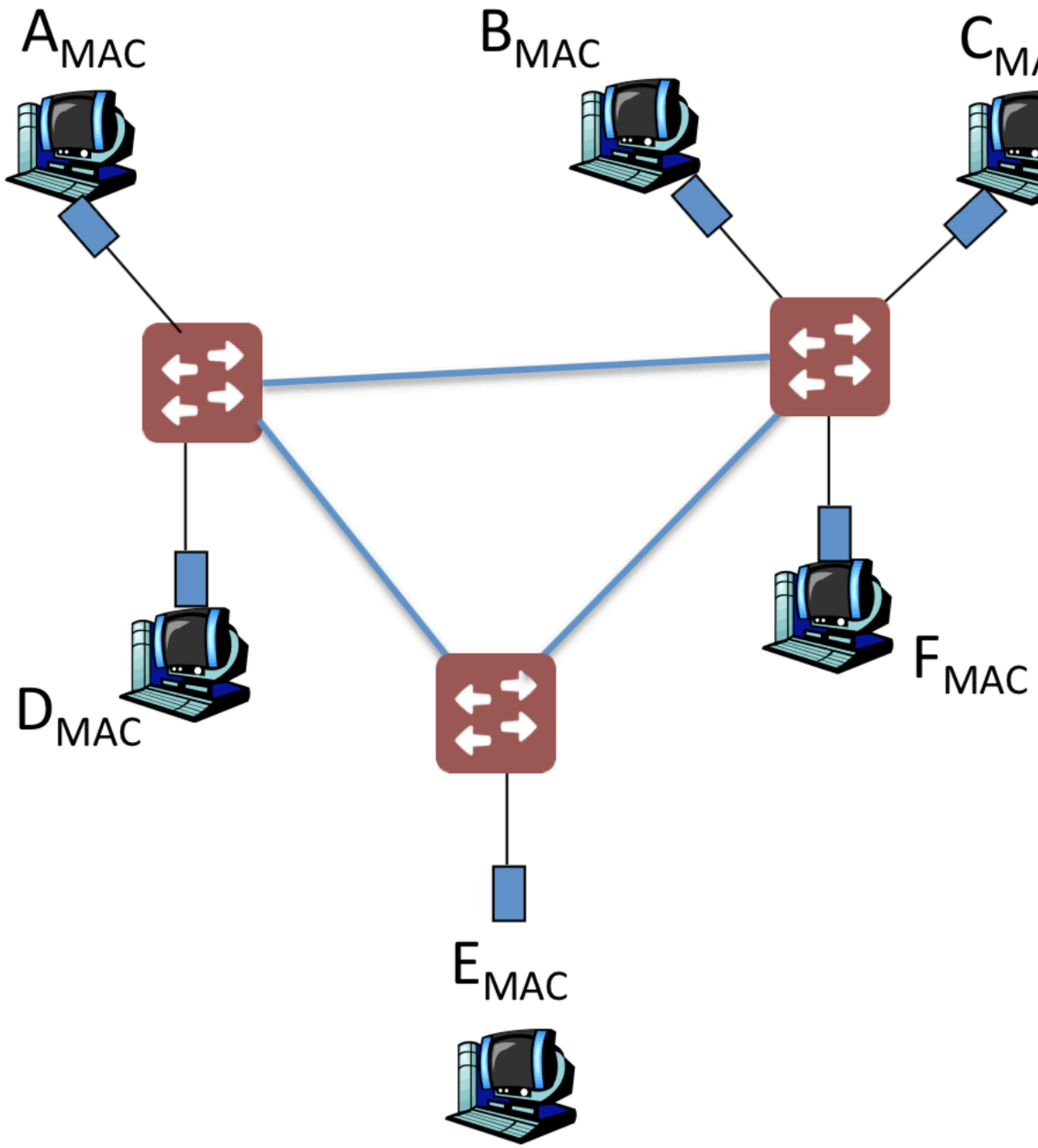
Recap: Broadcast storm



How to avoid the Broadcast Storm Problem?

Get rid of the loops!

Lets get back to the graph representation!



Easiest Way to Avoid Loops

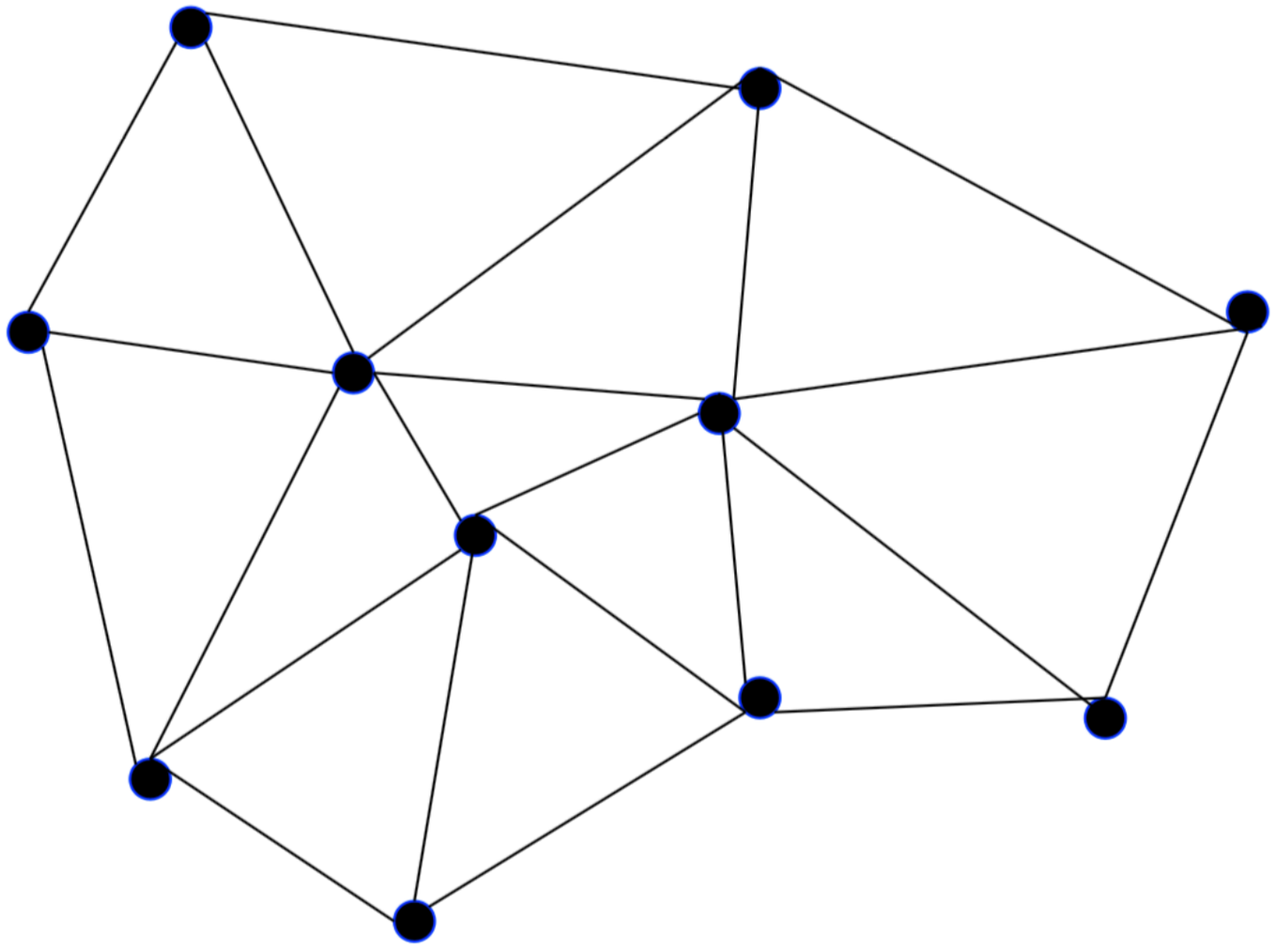
- Use a network topology (graph) where loop is impossible!

- Take arbitrary topology (graph)

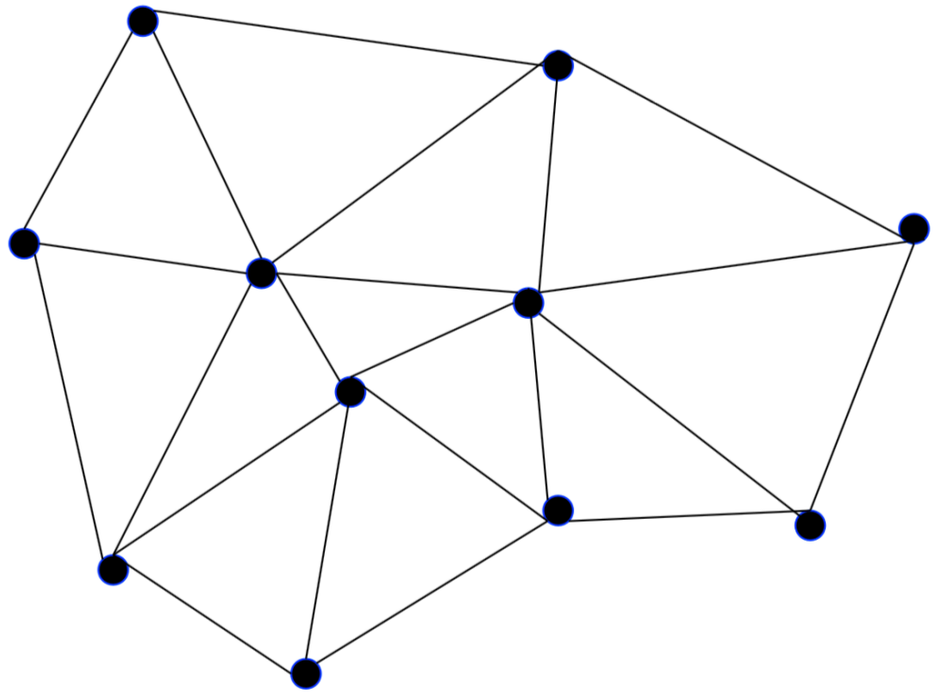
- **Build spanning tree**
    - **Subgraph that includes all vertices but contains no cycles**
    - Links not in the spanning tree are not used in forwarding frames

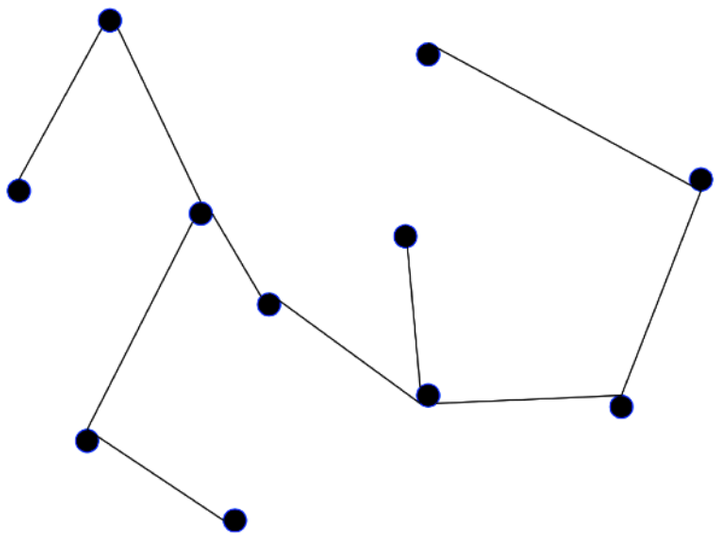
- Only one path to destinations on spanning trees
    - So don't have to worry about loops!

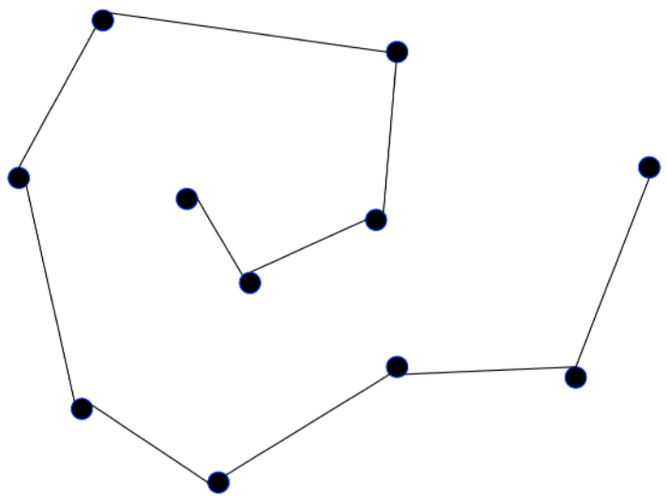
Consider Graph

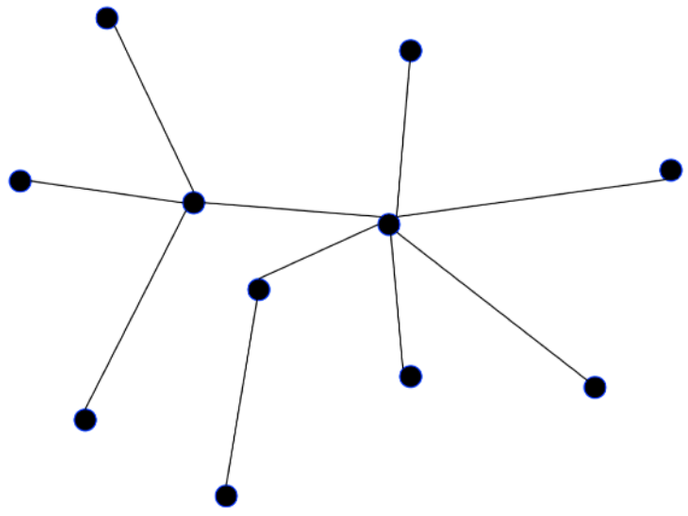


Multiple Spanning Trees









Questions?

Spanning Tree Approach

- Take arbitrary topology
- Pick subset of links that form a spanning tree
- Only forward packets on the spanning tree
    - => No loops
    - => No broadcast storm

Spanning Tree Protocol

- Protocol by which bridges construct a spanning tree
- Nice properties
    - Zero configuration (by operators or users)
    - Self healing
- Still used today
- Constraints for backwards compatibility
    - No changes to end-hosts
    - Maintain plug-n-play aspect
- Earlier Ethernet achieved plug-n-play by leveraging a broadcast medium
    - Can we do the same for a switched topology?

Algorithm has Two Aspects…

- Pick a root:
    - Destination to which the shortest paths go
    - Pick the one with the smallest identifier (MAC name/address)

- Compute the shortest paths to the root
    - No shortest path can have a cycle
    - Only keep the links on the shortest path
        - Break ties in some way
            - so we only keep one shortest path from each node

- Ethernet's spanning tree construction does both with a single algorithm

Breaking Ties

- When there are multiple shortest paths to the root:
    - Choose the path via neighbor switch with the smallest identifier

- **One could use any tie breaking system**
    - This is just an easy one to remember and implement

Constructing a Spanning Tree

- **Messages (Y,d,X)**
  - Proposing Y as the root
  - From node X
  - And advertising a distance d between X and Y

- Switches elect the node with smallest identifier (MAC address) as root
  - **Y** in messages

- Each switch determines if a link is on its shortest path to the root
  - If not, excludes it from the tree
  - **d** to **Y** in the message is used to determine this

Steps in Spanning Tree Protocol

- **Messages (Y,d,X)**
    - Proposing root Y; from node X; advertising a distance d to Y

- Initially each switch proposes itself as the root
    - that is, switch X announces (X,0,X) to its neighbors

- Switches update their view
    - Upon receiving message (Y,d,Y) from Z, check Y's id
    - If Y's id < current root: set root = Y
    - Set next-hop = Z

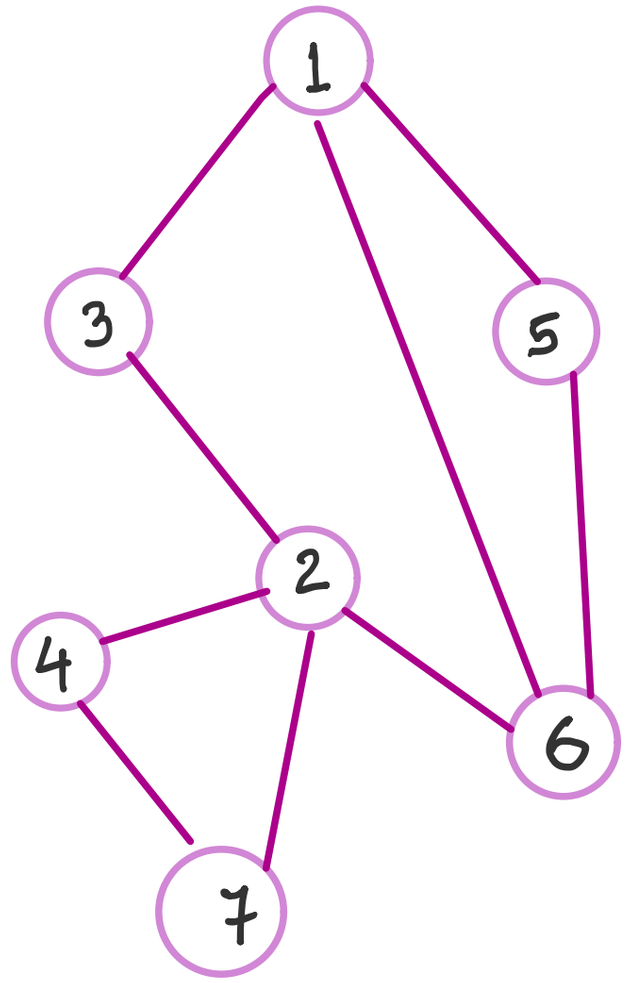
- Switches compute their distance from the root
    - Add 1 to the shortest distance received from a neighbor

- If **root changed OR shortest distance to the root changed:**
    - send neighbors updated message (Y,d+1,X)

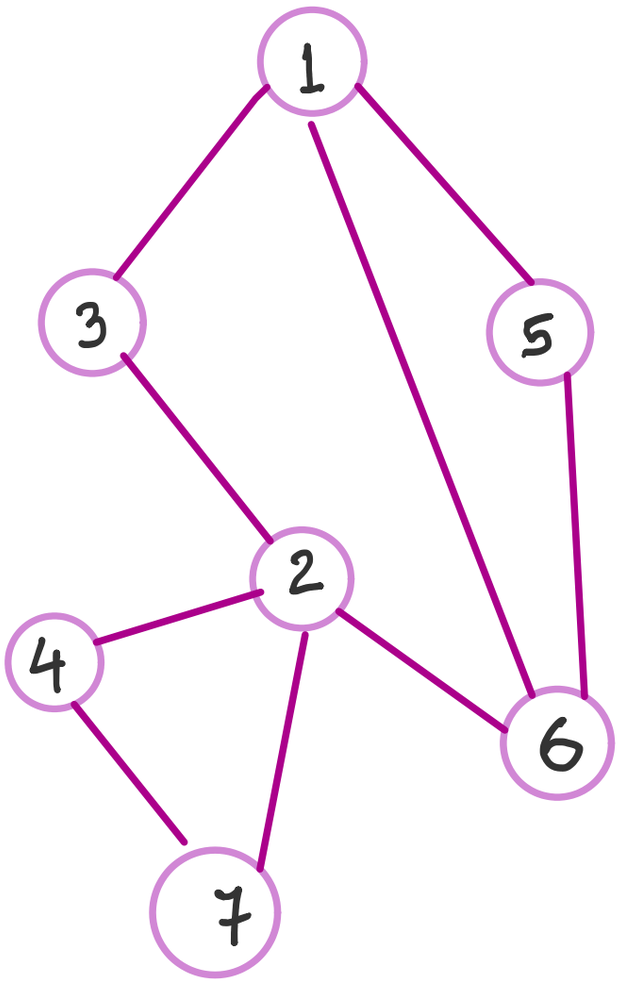
Group Exercise:

Lets run the Spanning Tree Protocol on this example

(assume all links have “distance” 1)

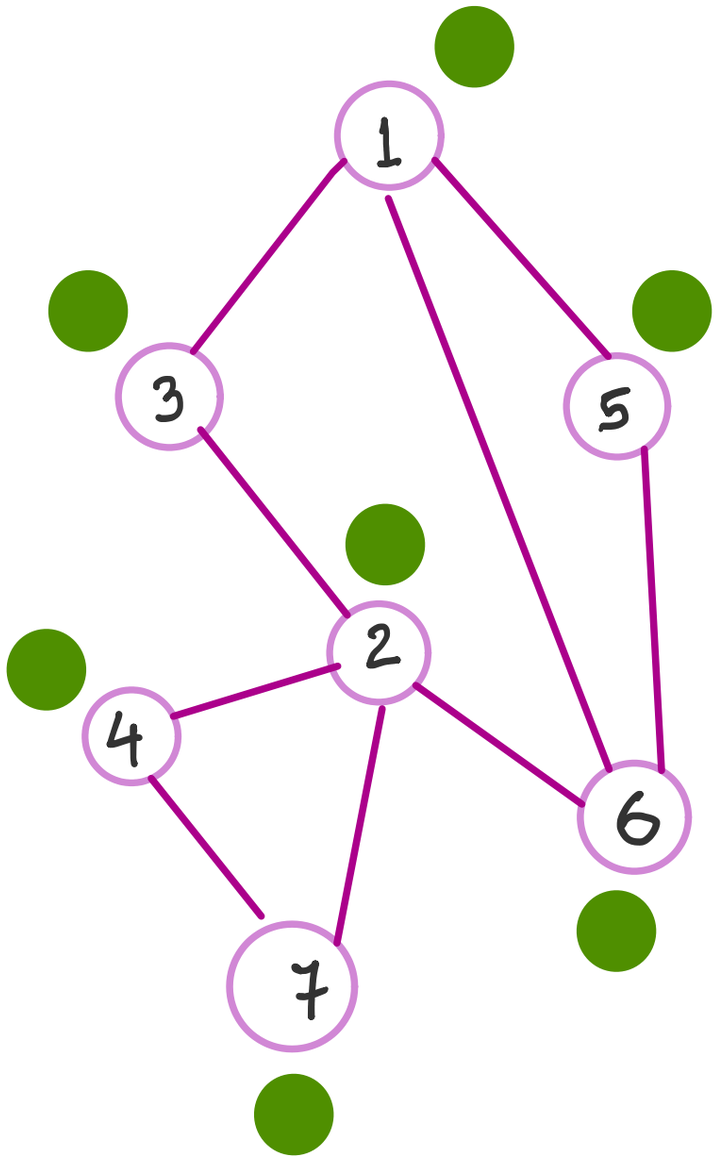


Round 1



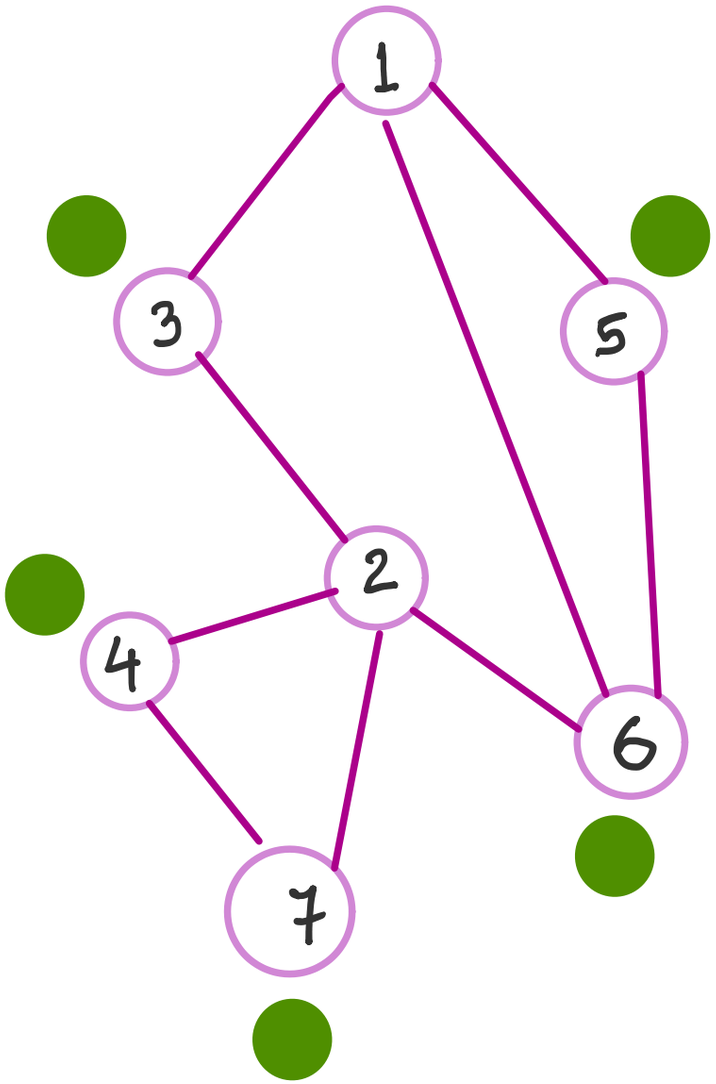
	Receive	Send	Next-hop
1		(1, 0, 1)	1
2		(2, 0, 2)	2
3		(3, 0, 3)	3
4		(4, 0, 4)	4
5		(5, 0, 5)	5
6		(6, 0, 6)	6
7		(7, 0, 7)	7

Round 2



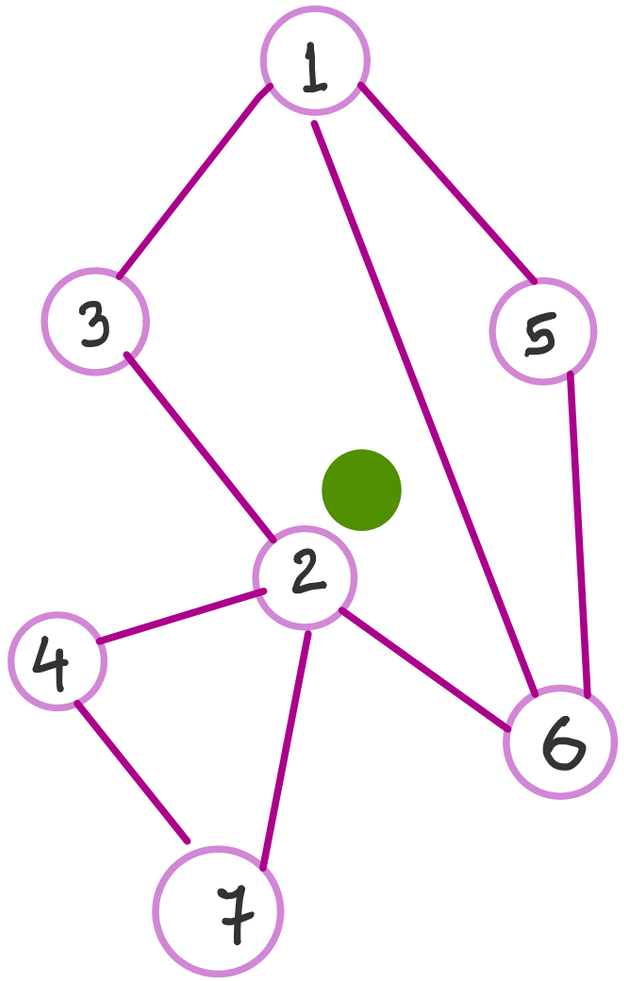
	Receive	Send	Next hop
1 (1, 0, 1)	(3, 0, 3), (5, 0, 5), (6, 0, 6)		1
2 (2, 0, 2)	(3, 0, 3), (4, 0, 4), (6, 0, 6), (7, 0, 7)		2
3 (3, 0, 3)	(1, 0, 1), (2, 0, 2)	**(1, 1, 3)**	**1**
4 (4, 0, 4)	(2, 0, 2), (7, 0, 7)	**(2, 1, 4)**	**2**
5 (5, 0, 5)	(1, 0, 1), (6, 0, 6)	**(1, 1, 5)**	**1**
6 (6, 0, 6)	(1, 0, 1), (2, 0, 2), (5, 0, 5)	**(1, 1, 6)**	**1**
7 (7, 0, 7)	(2, 0, 2), (4, 0, 4)	**(2, 1, 7)**	**2**

Round 3



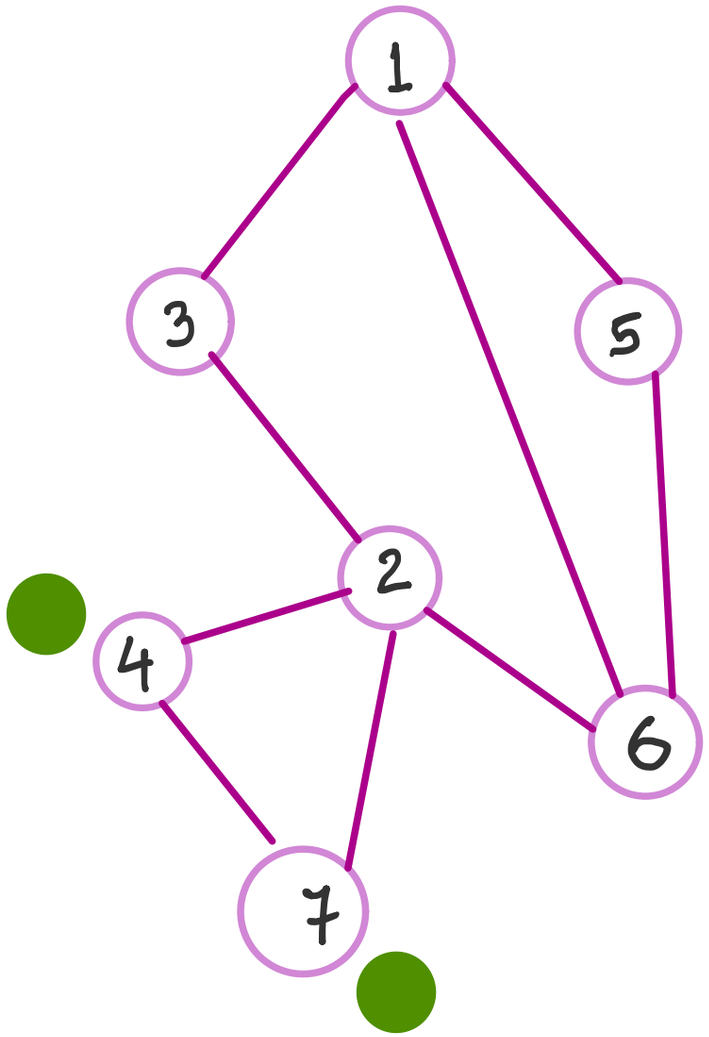
	Receive	Send	Next hop
1 (1, 0, 1)	(1, 1, 3), (1, 1, 5), (1, 1, 6)		1
2 (2, 0, 2)	(1, 1, 3), (2, 1, 4), (1, 1, 6), (2, 1, 7)	**(1, 2, 2)**	**3 (or 6)**
3 (1, 1, 3)			1
4 (2, 1, 4)	(2, 1, 7)		2
5 (1, 1, 5)	(1, 1, 6)		1
6 (1, 1, 6)	(1, 1, 5)		1
7 (2, 1, 7)	(2,1, 4)		2

Round 4



	Receive	Send	Next hop
1 (1, 0, 1)			1
2 (1, 2, 2)			3
3 (1, 1, 3)	(1, 2, 2)		1
4 (2, 1, 4)	(1, 2, 2)	**(1, 3, 4)**	**2**
5 (1, 1, 5)			1
6 (1, 1, 6)	(1, 2, 2)		1
7 (2, 1, 7)	(1, 2, 2)	**(1, 3, 7)**	**2**

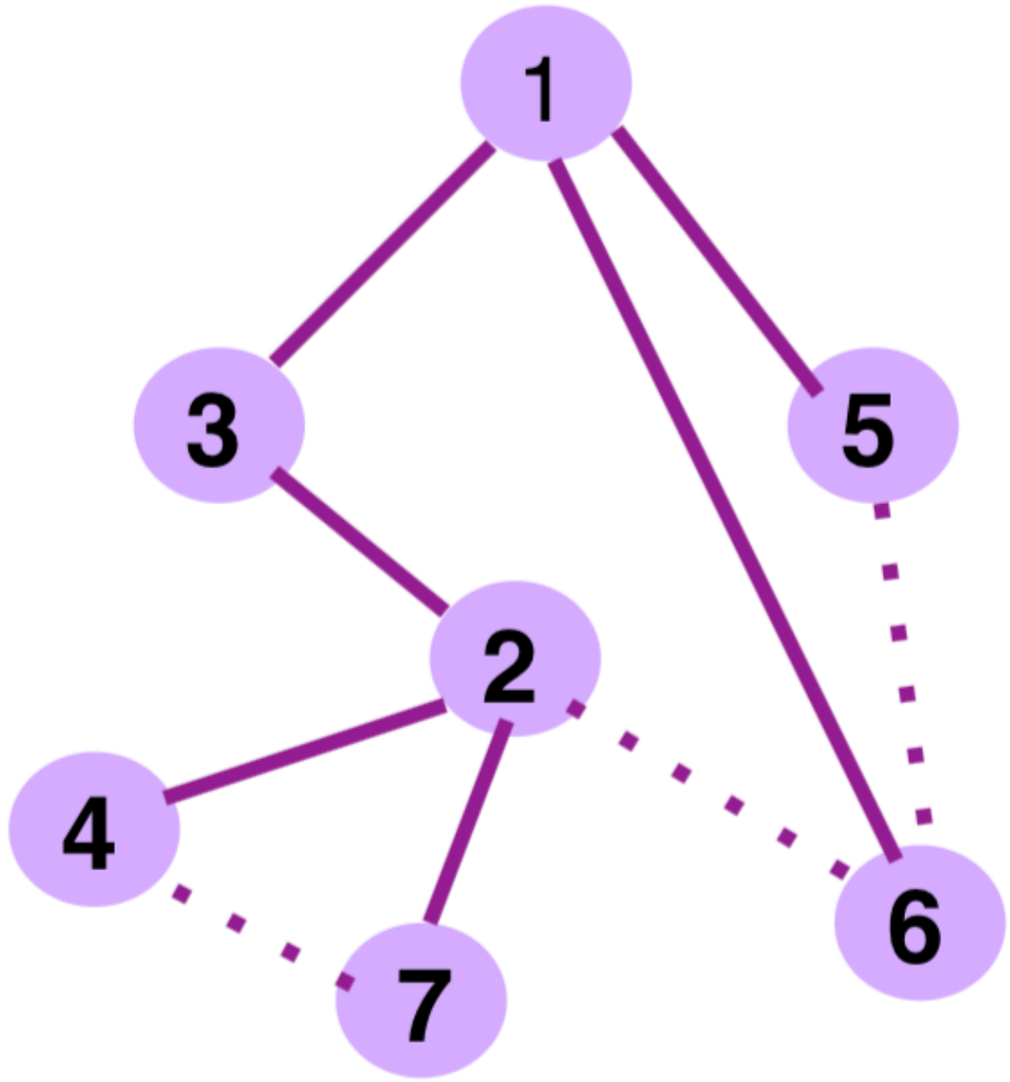
Round 5



	Receive	Send	Next hop
1 (1, 0, 1)			1
2 (1, 2, 2)	(1, 3, 4), (1, 3, 7)		3
3 (1, 1, 3)			1
4 (1, 3, 4)	(1, 3, 7)		2
5 (1, 1, 5)			1
6 (1, 1, 6)			1
7 (1, 3, 7)	(1, 3, 4)		2

After Round 5: We have our Spanning Tree

- 3-1
- 5-1
- 6-1
- 2-3
- 4-2
- 7-2



Questions?

Spanning Tree Protocol ++ (incorporating failures)

- Protocol must react to failures
    - Failure of the root node
    - Failure of switches and links

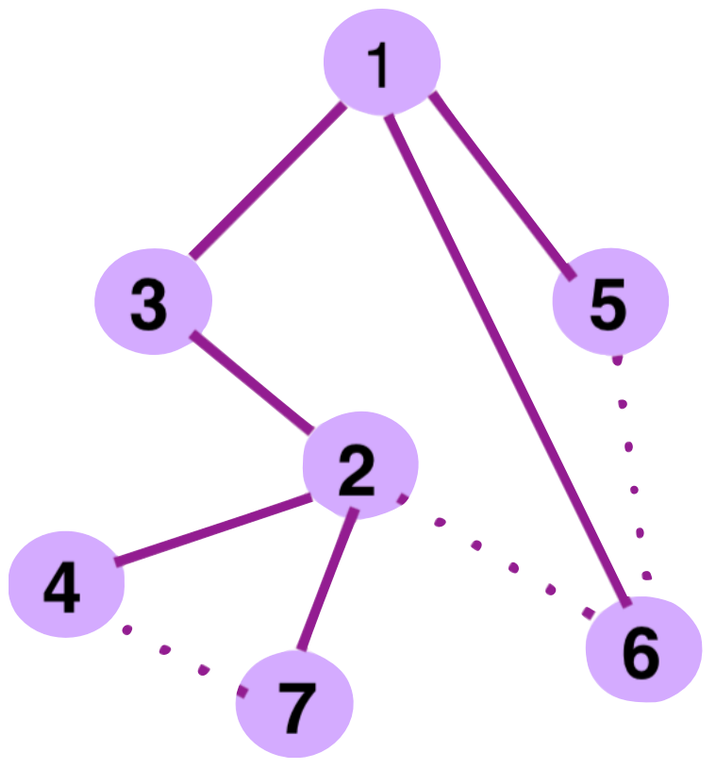
- **Root node sends periodic announcement messages**
    - Few possible implementations, but this is simple to understand
    - Other switches continue forwarding messages

- Detecting failures through timeout (soft state)
    - If no word from root, time out and send a (Y, 0, Y) message to all neighbors (in the graph)!

- **If multiple messages with a new root received, send message (Y, d, X) to the neighbor sending the message**

Suppose link 2-4 fails

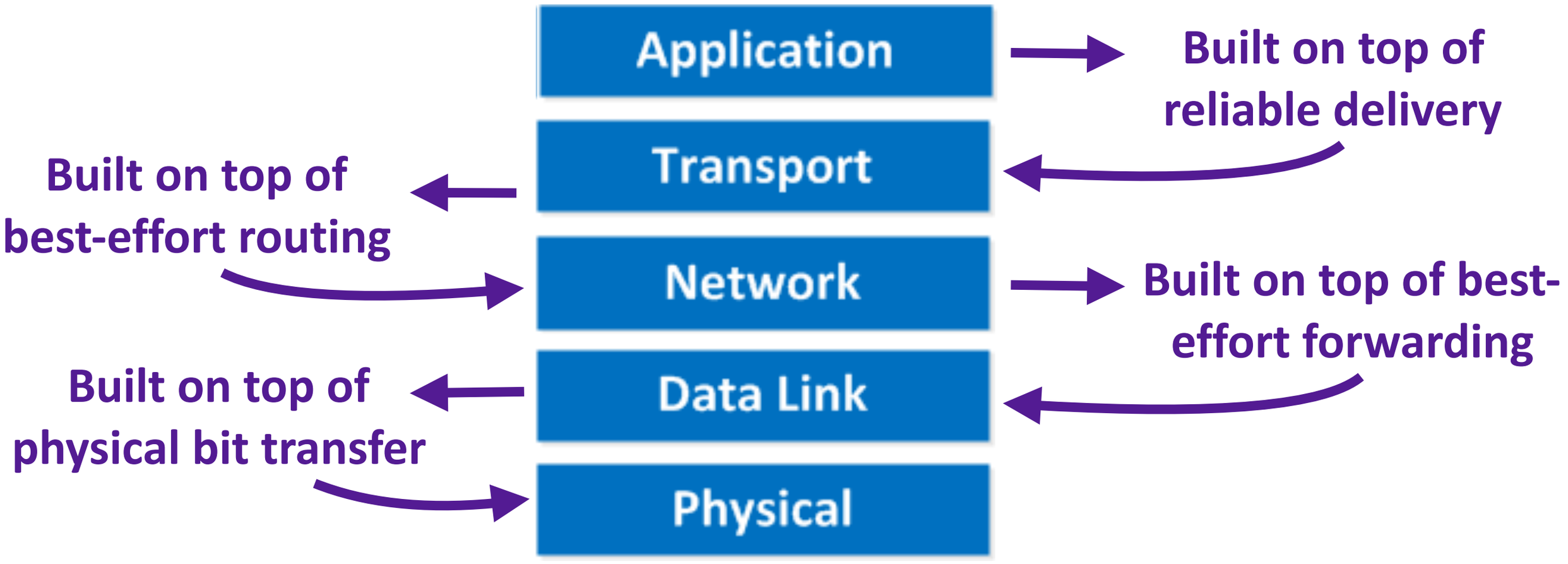
- 4 will send (4, 0, 4) to all its neighbors
    - 4 will stop receiving announcement messages from the root
    - Why?
- At some point, 7 will respond with (1, 3, 7)
- 4 will now update to (1, 4, 4) and send update message
- New spanning tree!



Questions?

The end of Link Layer ….

And the beginning of network layer :-D



Why do we need a network layer?

- There's only one path from source to destination
- How do you find that path? Ideas?
- Easy to design routing algorithms for trees
    - **Nodes can "flood" packet to all other nodes**

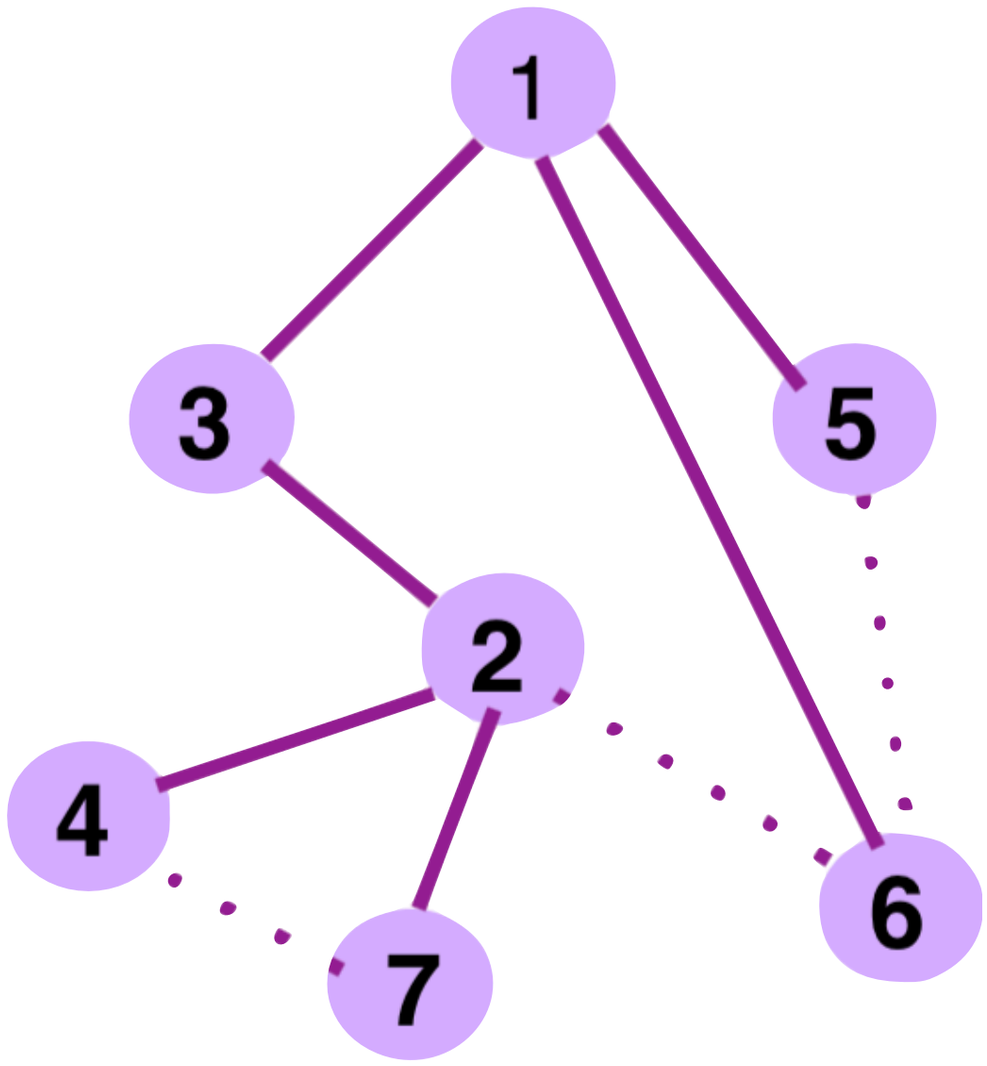
Flooding on a Spanning Tree

- Sends packet to ***every*** node in the network

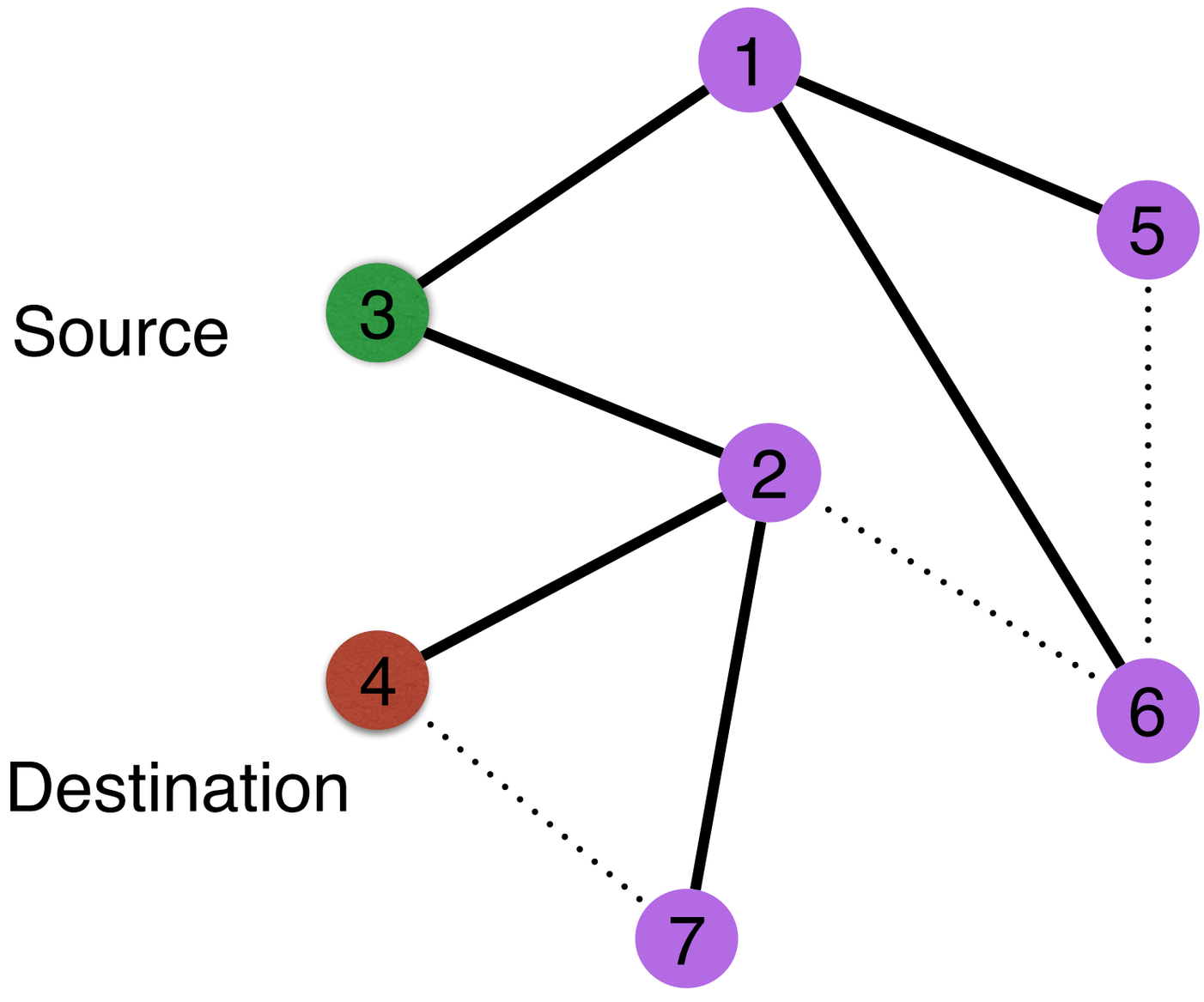
- **Step 1**: Ignore the links not belonging to the Spanning Tree

- **Step 2**: Originating node sends “flood” packet out every link (on spanning tree)

- **Step 3**: Send incoming packet out to all links **other than the one that sent the packet**

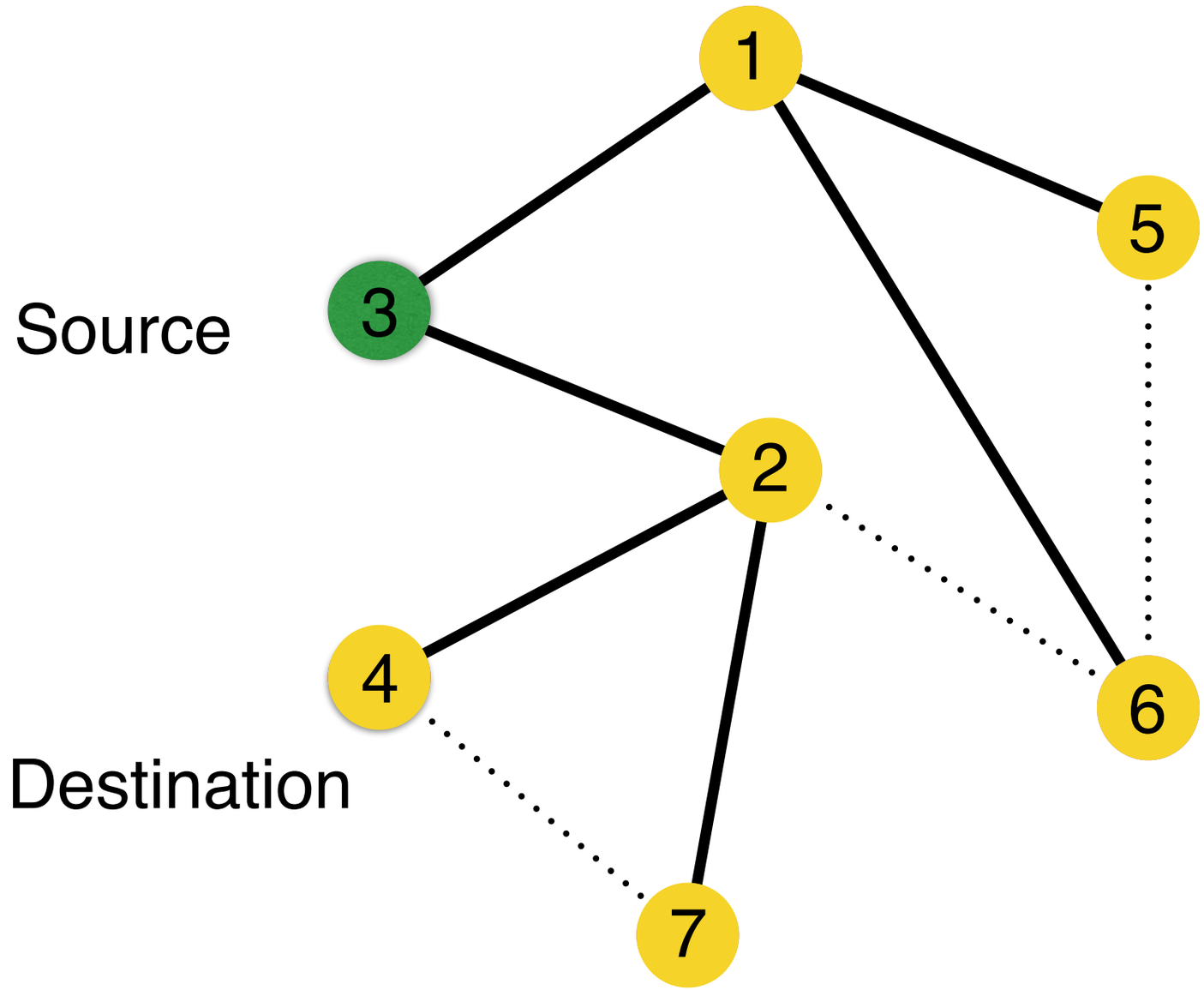


Flooding Example



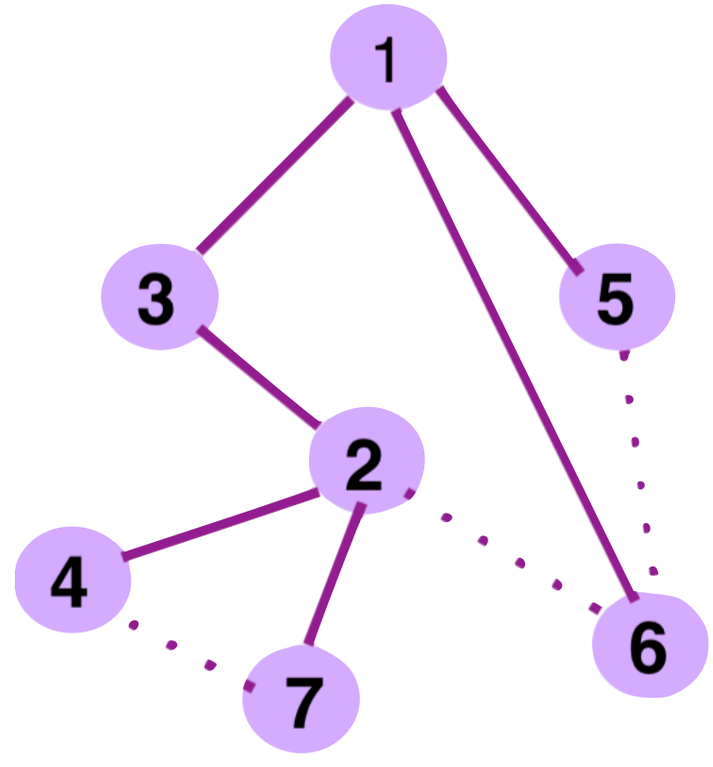
Flooding Example

Eventually all nodes are covered



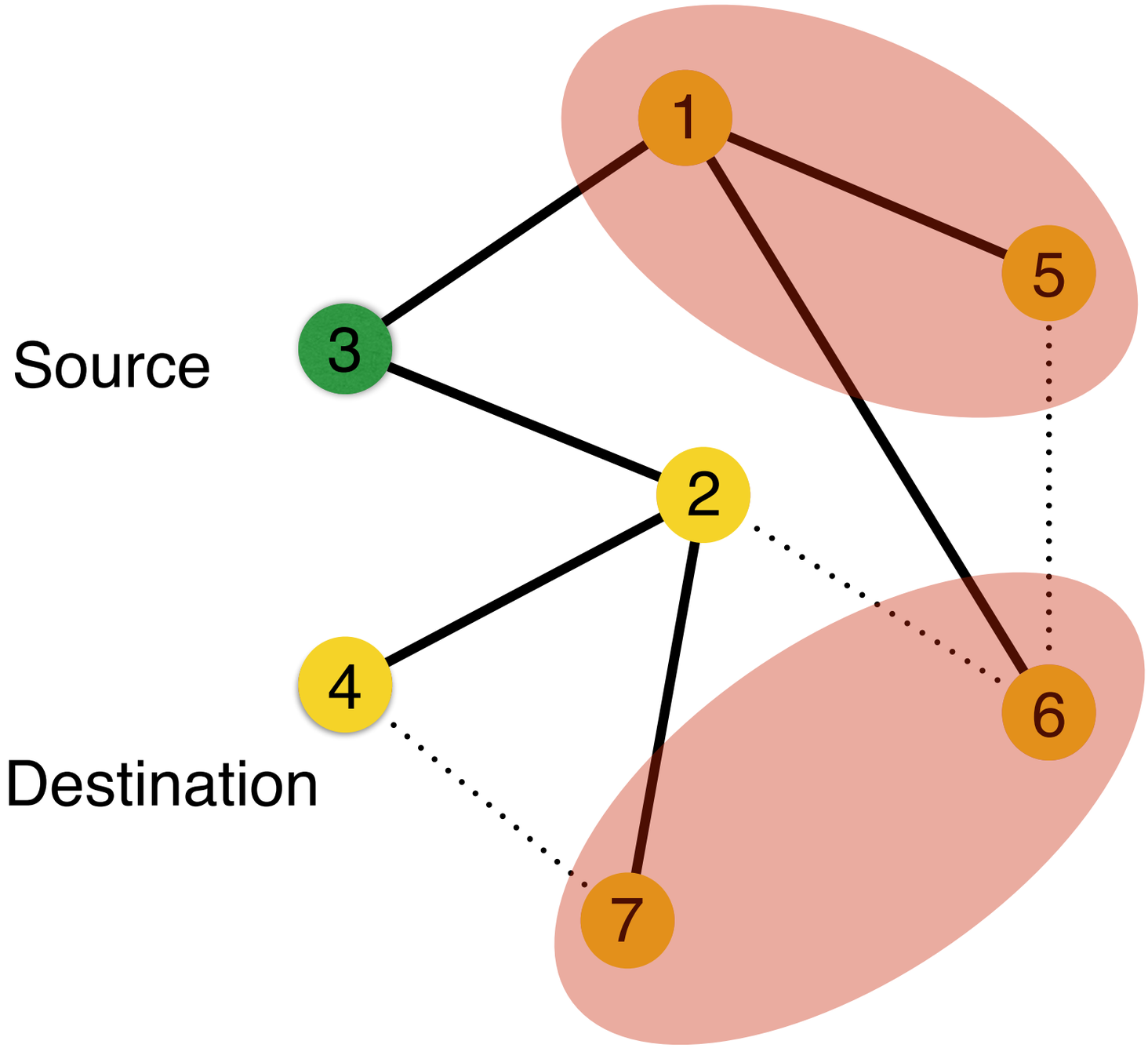
One copy of packet delivered to destination

Routing via Flooding on Spanning Tree …



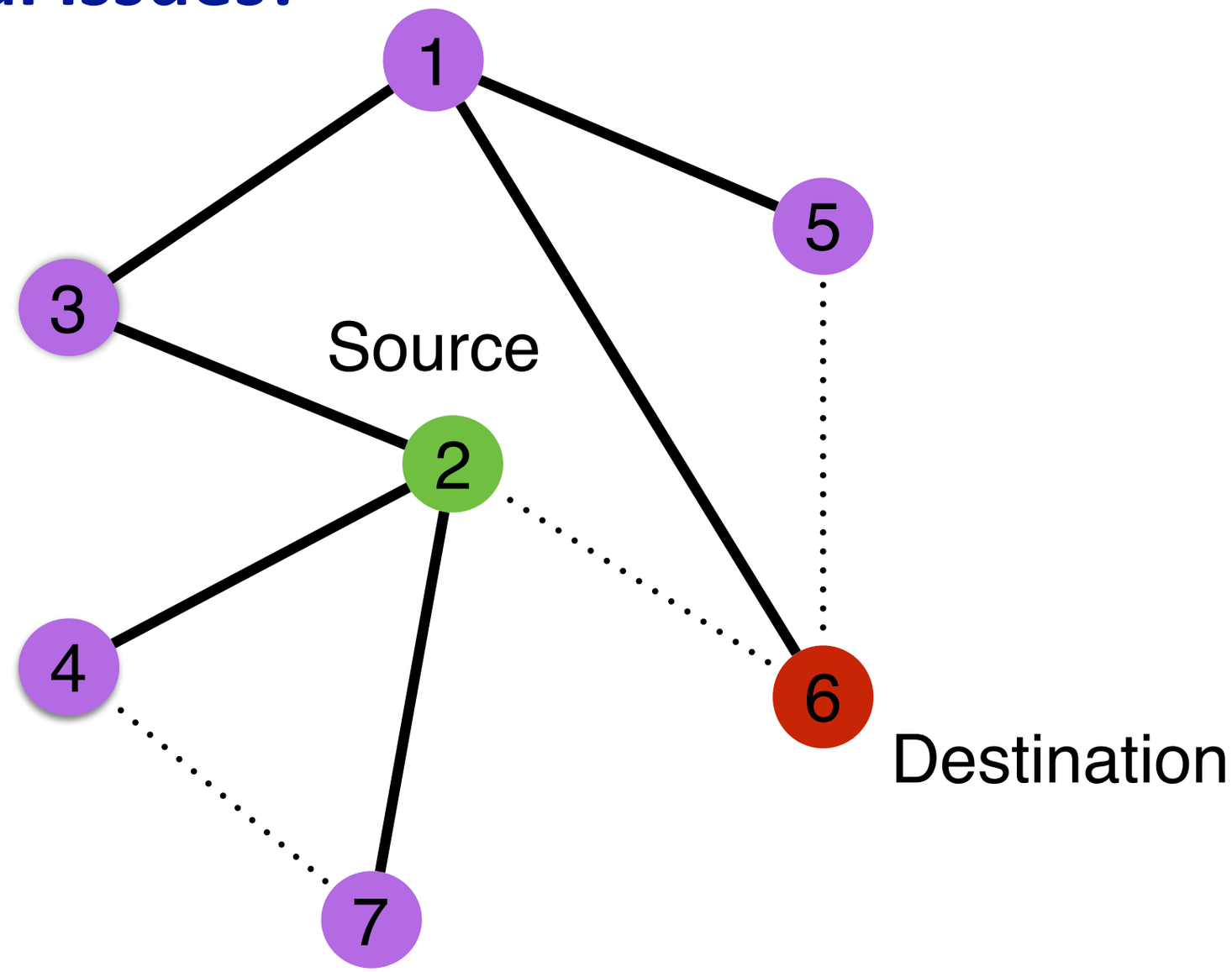
- There's only one path from source to destination
- How do you find that path? Ideas?
- Easy to design routing algorithms for trees
    - **Nodes can "flood" packet to all other nodes**
- Amazing properties:
    - No routing tables needed!
    - No packets will ever loop.
    - At least (and exactly) one packet must reach the destination
        - Assuming no failures

Three fundamental issues!



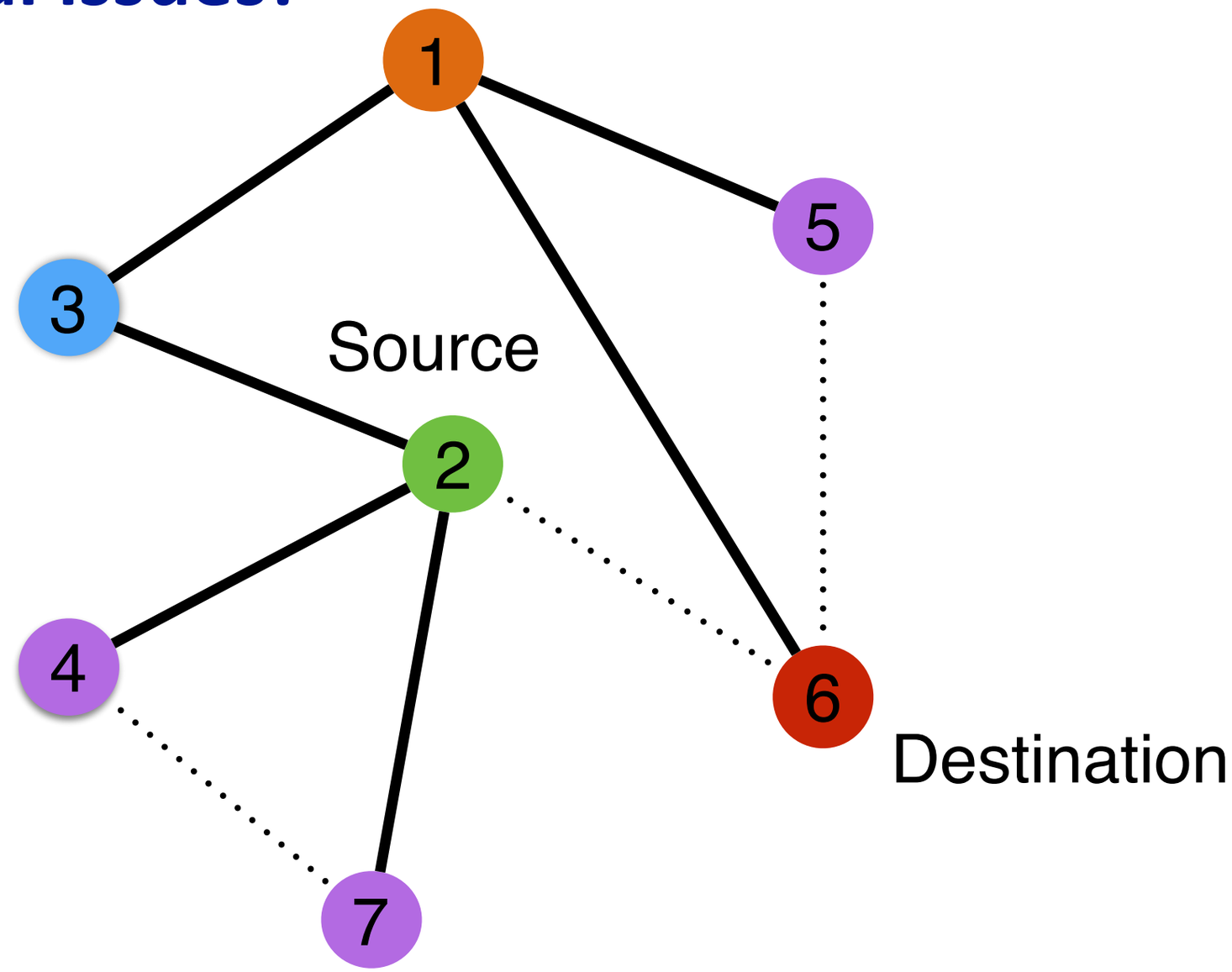
Issue 1: Each host has to do unnecessary packet processing!
(to decide whether the packet is destined to the host)

Three fundamental issues!



Issue 2: Higher latency!
(The packets unnecessarily traverse much longer paths)

Three fundamental issues!



Issue 2: Lower bandwidth availability!
(2-6 and 3-1 packets unnecessarily have to share bandwidth)

Questions?

Why do we need a network layer?

- Network layer performs “routing” of packets to alleviate these issues
- Uses routing tables
- Lets understand routing tables first
    - **We will see routing tables are nothing but …**
    - **Guess?**
- **A collection of (carefully constructed) spanning trees**
    - **One per destination**