Introduction to Computer Networks

CS640

L2 Switching

https://pages.cs.wisc.edu/~mgliu/CS640/S26/index.html

Ming Liu
mgliu@cs.wisc.edu

Outline

- Last
  - Framing and Error Handling
- Today
  - L2 Switching
- Announcements
  - Quiz1 this Thursday (Feb. 12) in-class

Recap

- Key Questions
  - How do we identify a frame from a bit stream?
  - How do we handle frame errors?

- Terminology
  - Byte Stuffing, Byte Counting, and Bit Stuffing
  - Parity Bit, 2-D Parity, Internet Checksum, CRC

Ethernet

- The most popular wired computer network technology
  - Co-invented by Robert Metcalfe (Turing Award 2023)
  - First developed at Xerox PARC for Alto computers to communicate
  - IEEE 802.3 standards
  - Widely used in local area network (LAN) and wide area network (WAN)

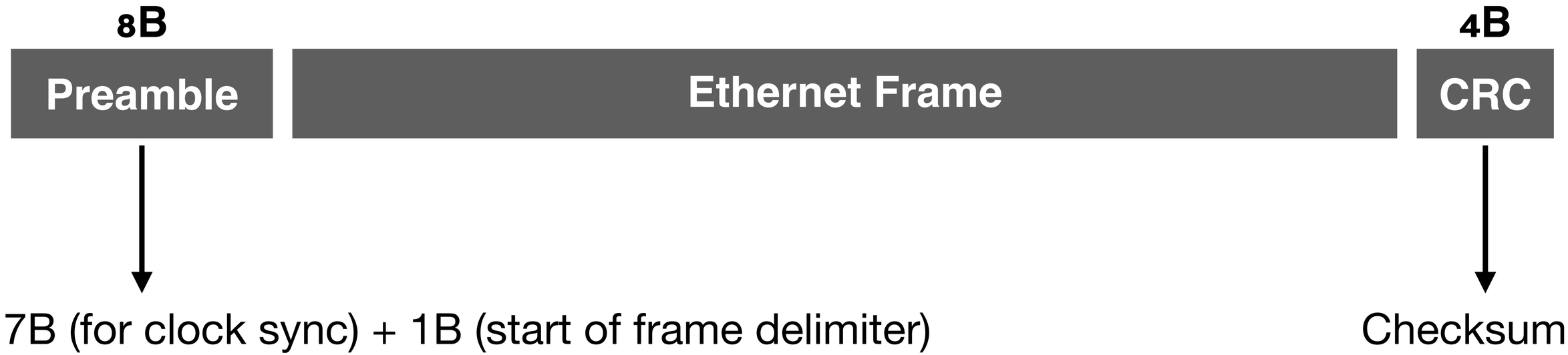
Ethernet

- The most popular wired computer network technology
  - Co-invented by Robert Metcalfe (Turing Award 2023)
  - First developed at Xerox PARC for Alto computers to communicate
  - IEEE 802.3 standards
  - Widely used in local area network (LAN) and wide area network (WAN)

We will discuss L2 switching based on Ethernet.

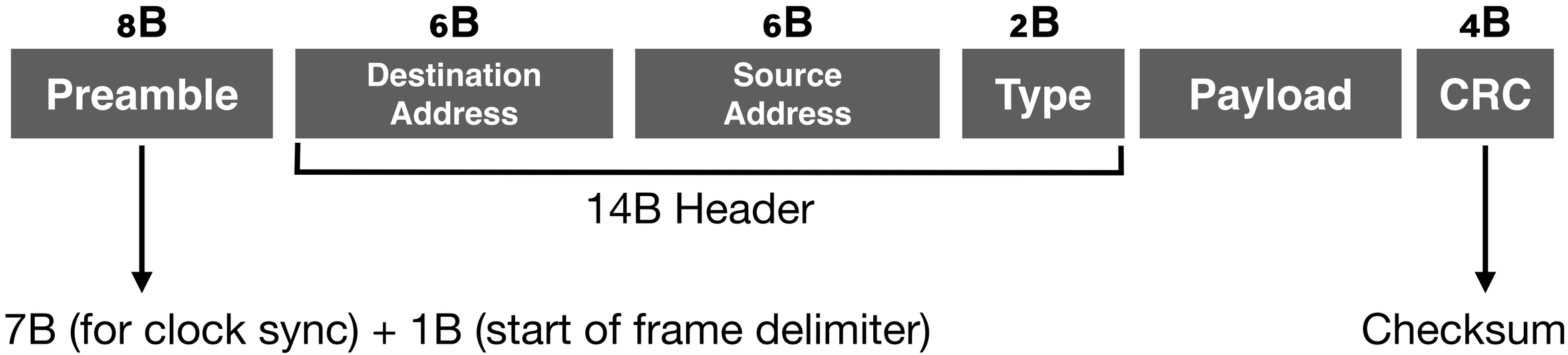
Ethernet Framing

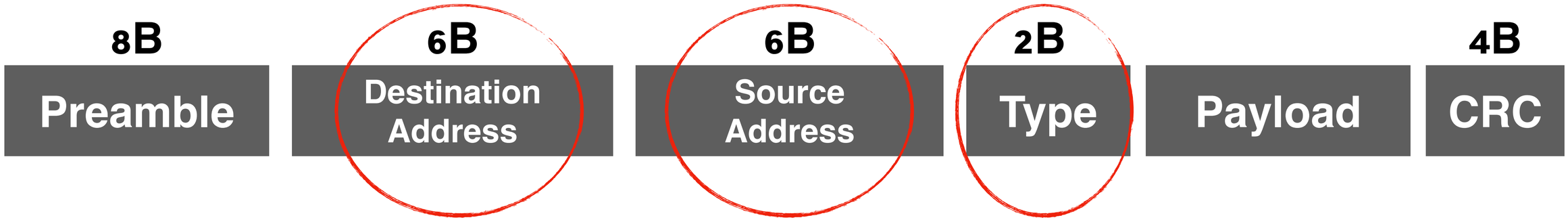
- Ethernet uses the bit stuffing technique
- Ethernet uses the CRC checksum for error handling



Ethernet Framing

- Ethernet uses the bit stuffing technique
- Ethernet uses the CRC checksum for error handling





What is an address?
What is a type?

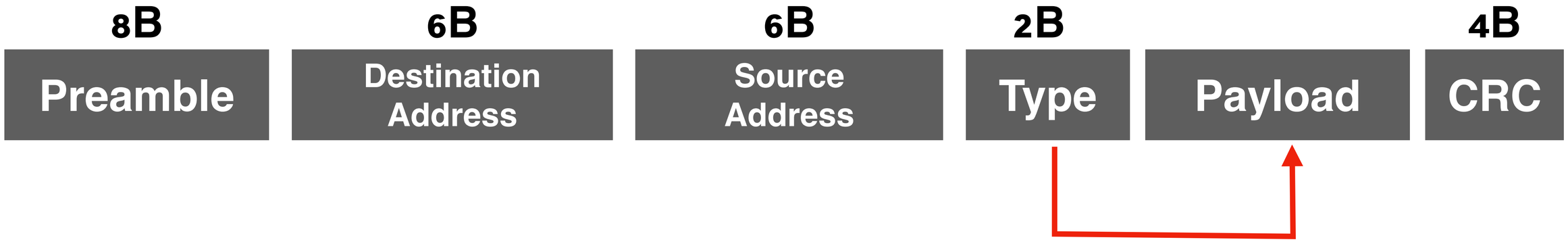
MAC Address

- MAC = Media Access Control Address
  - A unique identifier for a switch (and NIC) port
  - Originally from the Xerox System Ethernet Addressing scheme
  - 48-bit, e.g., bc:97:e1:13:82:d4

- Assigned by the hardware vendor
  - The first three bytes identify the organization
  - Also known as the organizational unique identifier (OUI)
  - https://www.wireshark.org/tools/oui-lookup.html

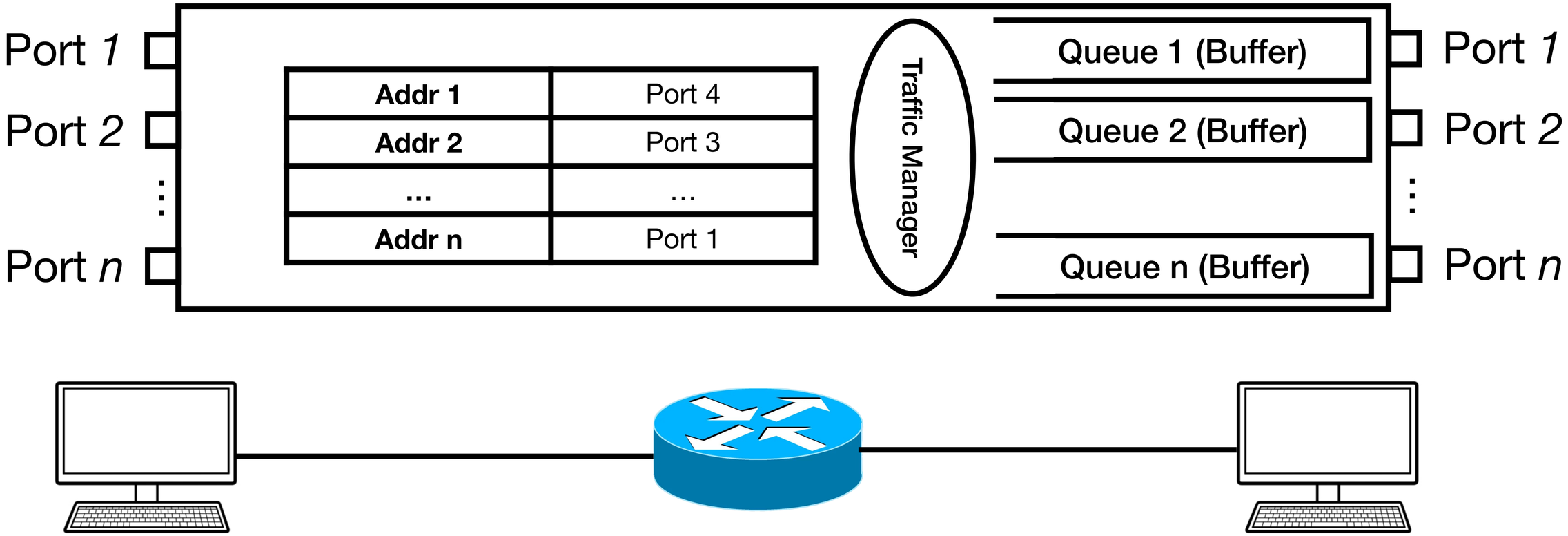
Ethernet Frame Type

- Indicate which protocol encapsulates the payload
  - 0x0800 —> IPv4 (Internet Protocol Version 4)
  - 0x0806 —> ARP (Address Resolution Protocol)
  - 0x08DD —> IPv6 (Internet Protocol Version 6)

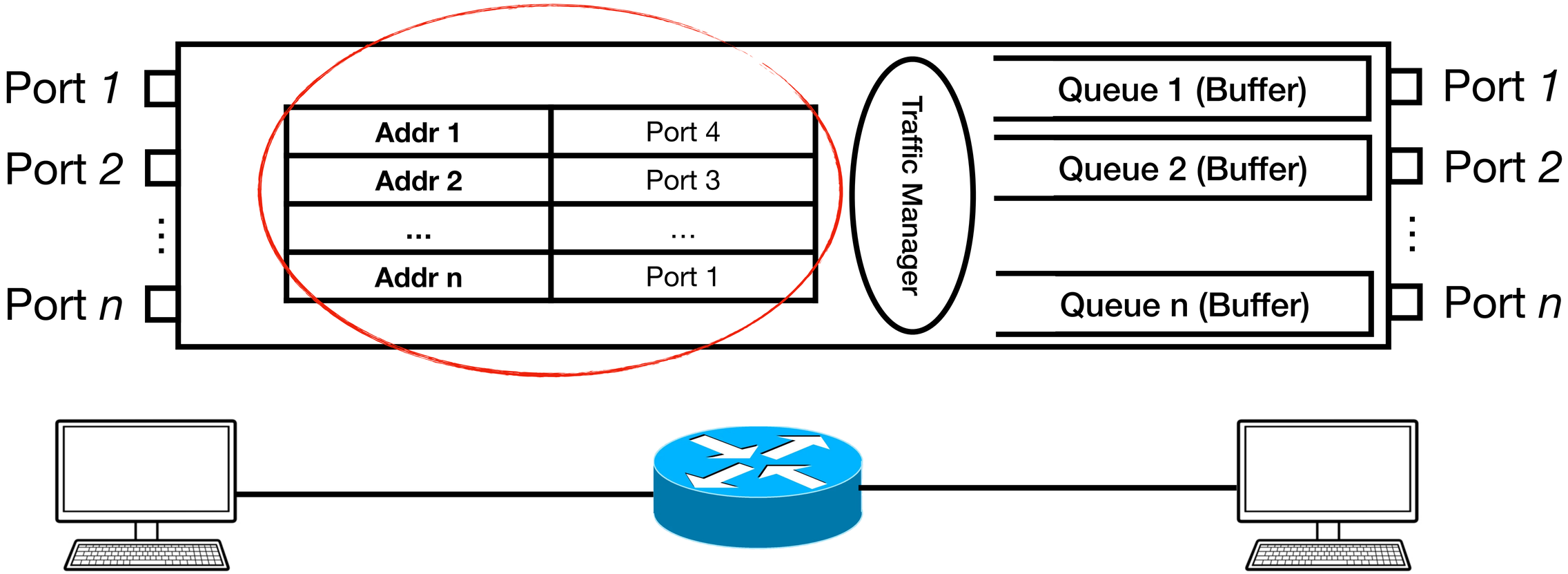


After identifying an Ethernet frame, how does the switch/router forward it?

Recap: Packet Switching



Recap: Packet Switching



Forwarding Table

- Each switch/router maintains a forwarding table:
  - <MAC address, port, age>
  - Mac address: the destination MAC address
  - Port: the forwarding port number of the switch
  - Age: the valid period of the entry

MAC address	Port	Age (s)
11:22:33:44:55:66	1	2
77:88:99:aa:bb:cc	3	4
dd:ee:ff:11:22:33	2	6

How does the frame forwarding work?

- For an incoming frame,
  - The switch looks up the forwarding table and performs an exact match
  - If there is a hit, send the frame to the matched port
  - If there is a miss, do broadcast, i.e., **send the frame to all ports except the incoming port**

How does the frame forwarding work?

- For an incoming frame,
  - The switch looks up the forwarding table and performs an exact match
  - If there is a hit, send the frame to the matched port
  - If there is a miss, do broadcast, i.e., **send the frame to all ports except the incoming port**

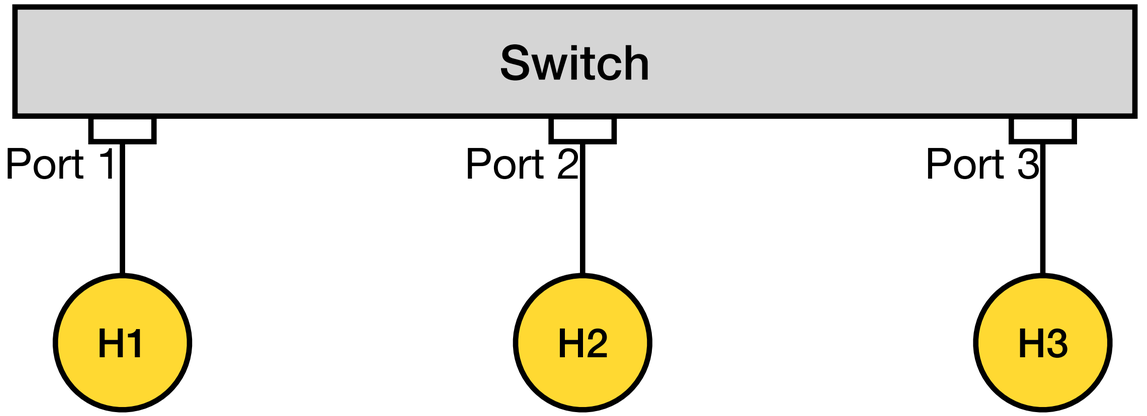
Ethernet Frame

77:88:99:aa:bb:cc	**11:22:33:44:55:66**	**0x08 00**	**Payload**

Forwarding Table

MAC address	**Port**	**Age (s)**
11:22:33:44:55:66	1	2
77:88:99:aa:bb:cc	3	4
dd:ee:ff:11:22:33	2	6

An Example



Host1 (11:22:33:44:55:66) **Host2** (dd:ee:ff:11:22:33) **Host3** (77:88:99:aa:bb:cc)

MAC address	Port	Age (s)
11:22:33:44:55:66	1	2
77:88:99:aa:bb:cc	3	4
dd:ee:ff:11:22:33	2	6

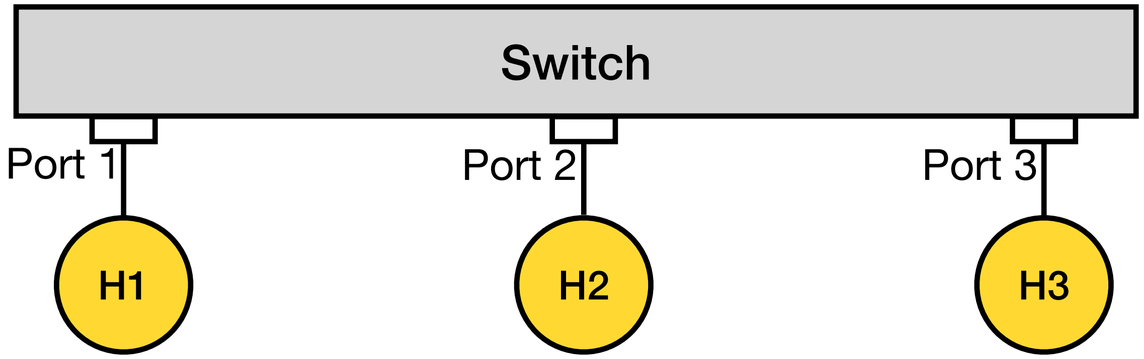
How is the forwarding table filled up?

MAC Learning

- An automatic MAC (forwarding) table filling technique
  - Keep track of the source address of a frame and the arriving port

MAC Learning

- An automatic MAC (forwarding) table filling technique
  - Keep track of the source address of a frame and the arriving port



Host1 (11:22:33:44:55:66)

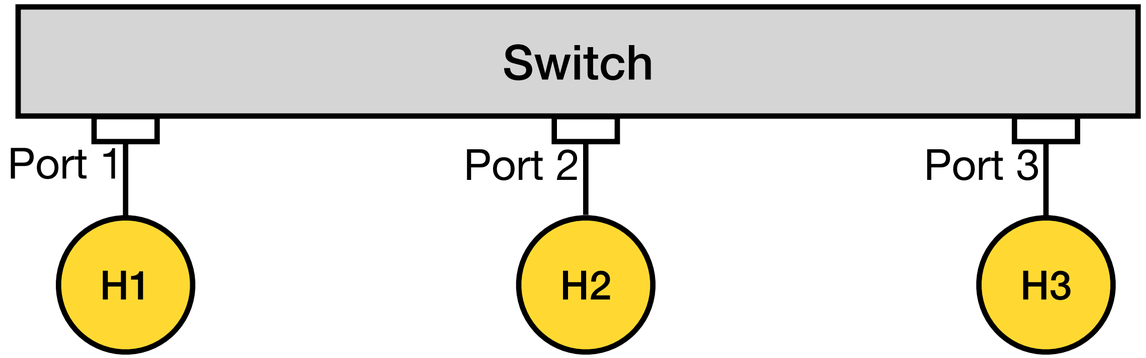
Host2 (dd:ee:ff:11:22:33)

Host3 (77:88:99:aa:bb:cc)

MAC address	Port	Age (s)

MAC Learning

- An automatic MAC (forwarding) table filling technique
  - Keep track of the source address of a frame and the arriving port



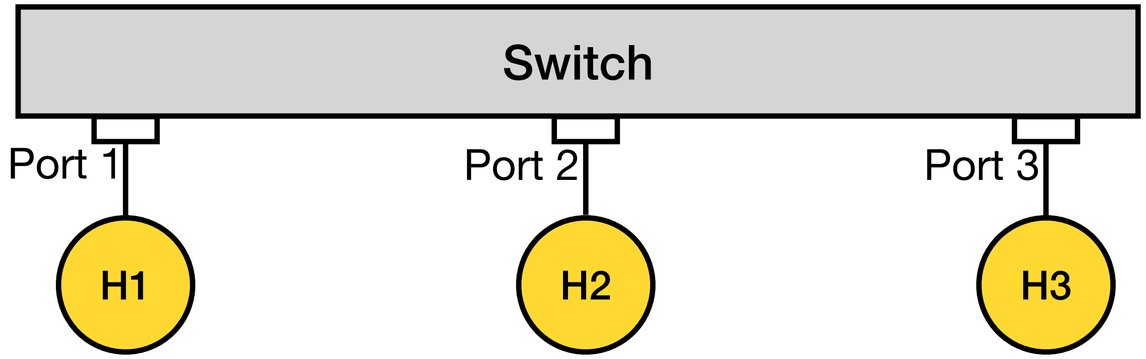
Host1 (11:22:33:44:55:66) **Host2** (dd:ee:ff:11:22:33) **Host3** (77:88:99:aa:bb:cc)

Host1 -> Host3

MAC address	Port	Age (s)

MAC Learning

- An automatic MAC (forwarding) table filling technique
  - Keep track of the source address of a frame and the arriving port



Host1 (11:22:33:44:55:66) **Host2** (dd:ee:ff:11:22:33) **Host3** (77:88:99:aa:bb:cc)

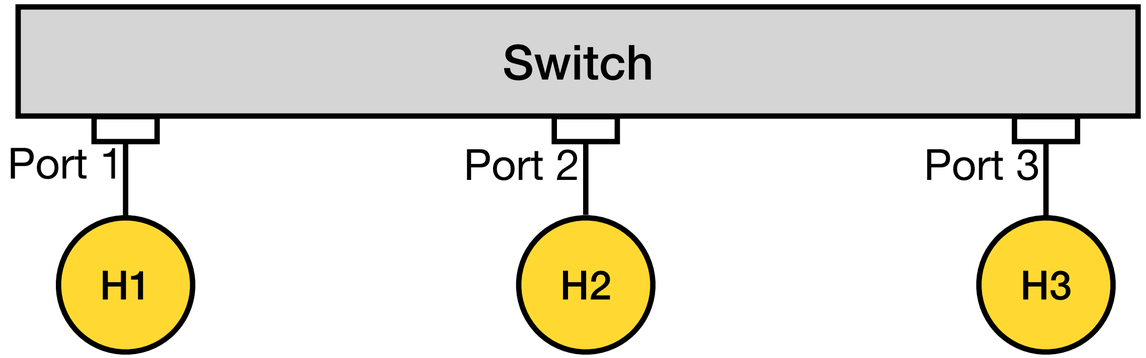
Host1 -> Host3

- The frame comes to port 1 and is broadcast
- The default aging time is 10s

MAC address	Port	Age (s)
11:22:33:44:55:66	1	10

MAC Learning

- An automatic MAC (forwarding) table filling technique
  - Keep track of the source address of a frame and the arriving port



Host1 (11:22:33:44:55:66)

Host2 (dd:ee:ff:11:22:33)

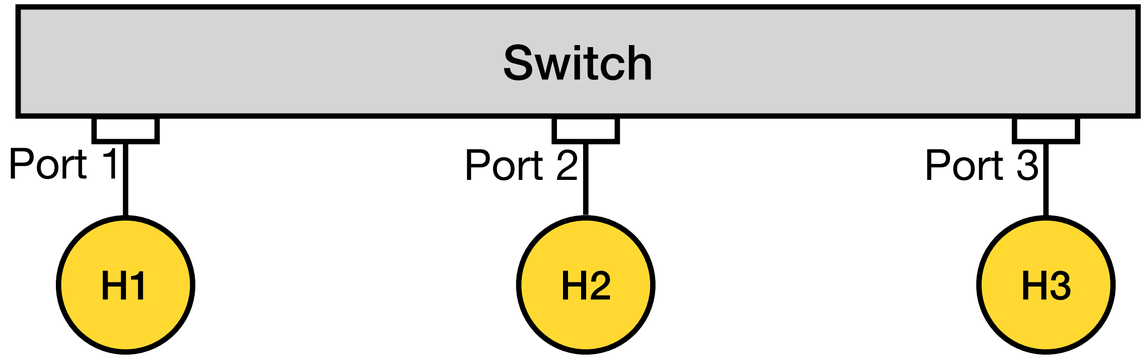
Host3 (77:88:99:aa:bb:cc)

Host3 -> Host2

MAC address	Port	Age (s)
11:22:33:44:55:66	1	10

MAC Learning

- An automatic MAC (forwarding) table filling technique
  - Keep track of the source address of a frame and the arriving port



Host1 (11:22:33:44:55:66) **Host2** (dd:ee:ff:11:22:33) **Host3** (77:88:99:aa:bb:cc)

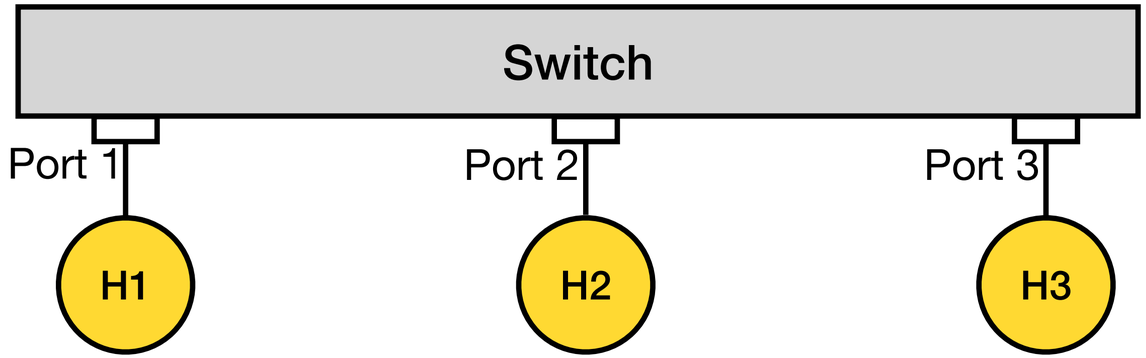
Host3 -> Host2

- The frame comes to port 3 and is broadcast
- The default aging time is 10s

MAC address	Port	Age (s)
11:22:33:44:55:66	1	8
77:88:99:aa:bb:cc	3	10

MAC Learning

- An automatic MAC (forwarding) table filling technique
  - Keep track of the source address of a frame and the arriving port



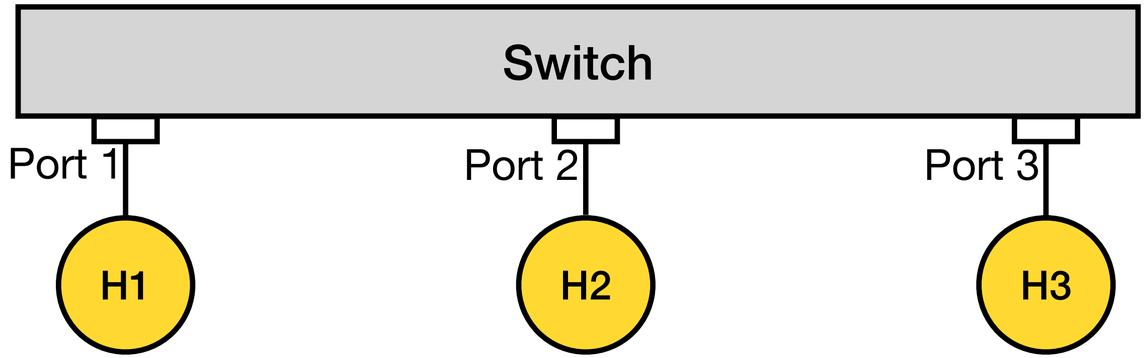
Host1 (11:22:33:44:55:66) **Host2** (dd:ee:ff:11:22:33) **Host3** (77:88:99:aa:bb:cc)

Host3 -> Host1

MAC address	Port	Age (s)
11:22:33:44:55:66	1	8
77:88:99:aa:bb:cc	3	10

MAC Learning

- An automatic MAC (forwarding) table filling technique
  - Keep track of the source address of a frame and the arriving port



Host1 (11:22:33:44:55:66) **Host2** (dd:ee:ff:11:22:33) **Host3** (77:88:99:aa:bb:cc)

Host3 -> Host1

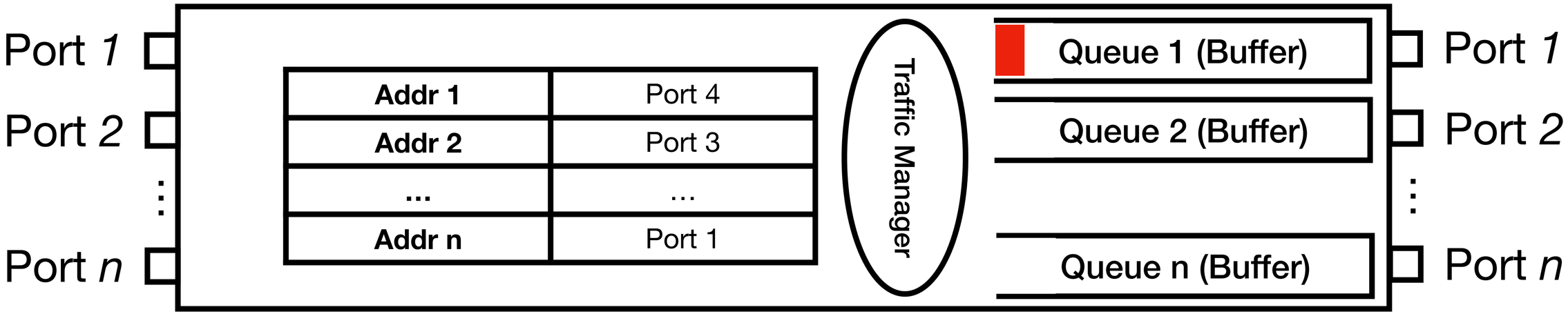
- The frame comes to port 3 and is forwarded

MAC address	Port	Age (s)
11:22:33:44:55:66	1	8
77:88:99:aa:bb:cc	3	10

Recap: Store-and-Forward Transmission

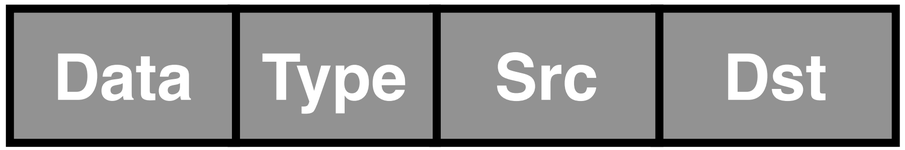
- The hardware can only forward after receiving the entire packet
  - Packets need to be buffered!

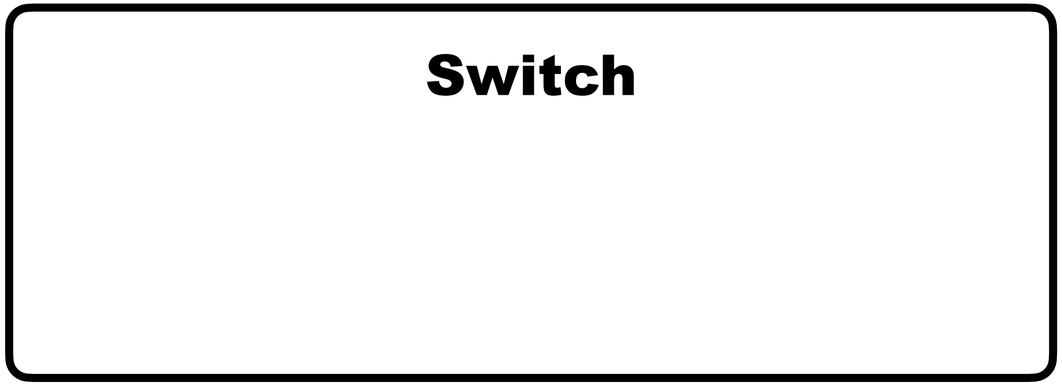
- Suppose a packet has *L* bits, and a switch transmits at *R* bits/sec
  - The switch takes *L/R* time to transmit the packet at the outbound port



Store-and-Forward under Ethernet

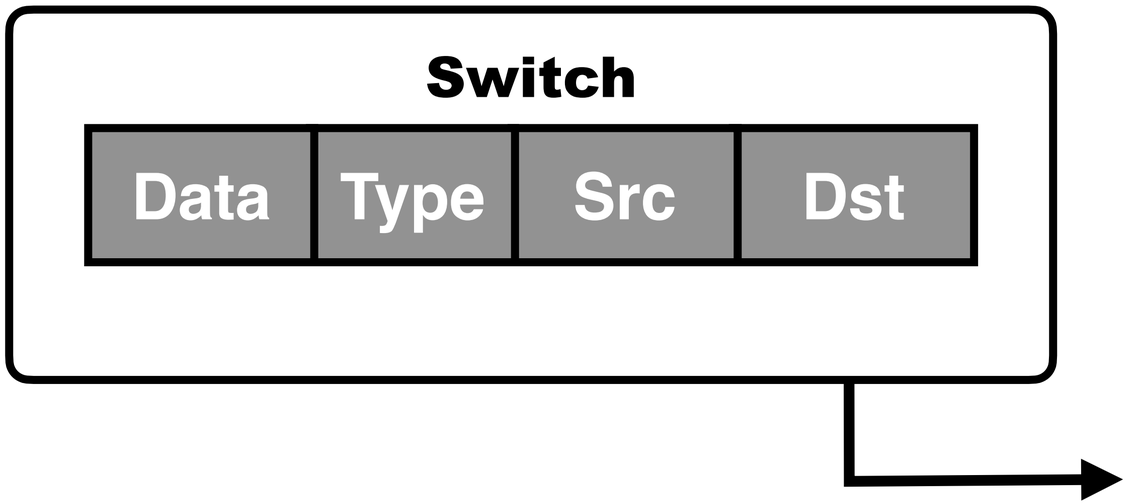
- The switch should wait to forward the frame until receiving an entire frame





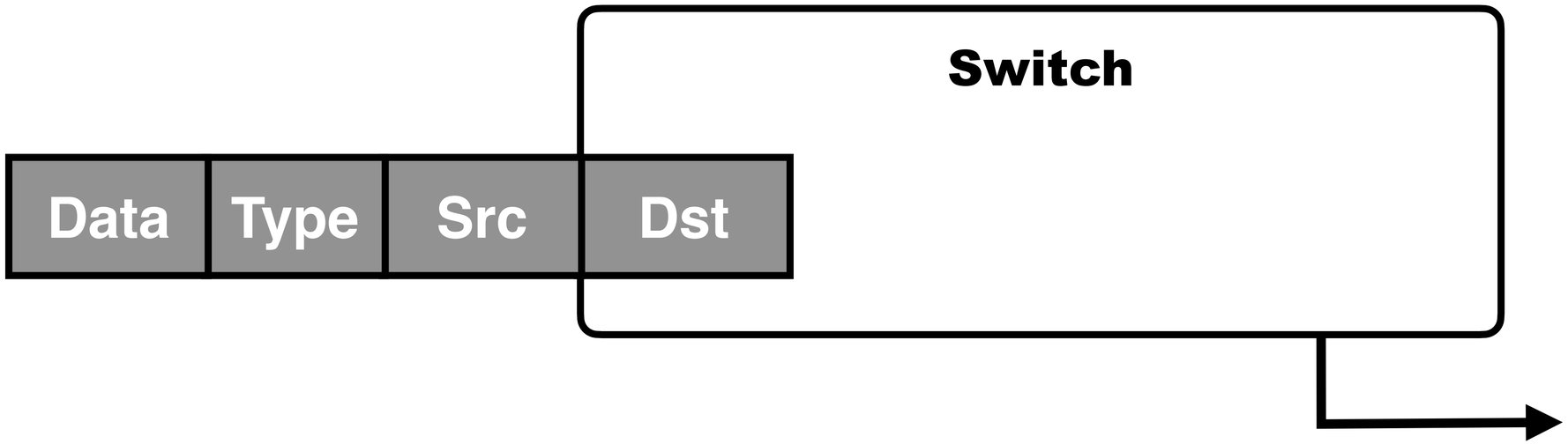
Store-and-Forward under Ethernet

- The switch should wait to forward the frame until receiving an entire frame



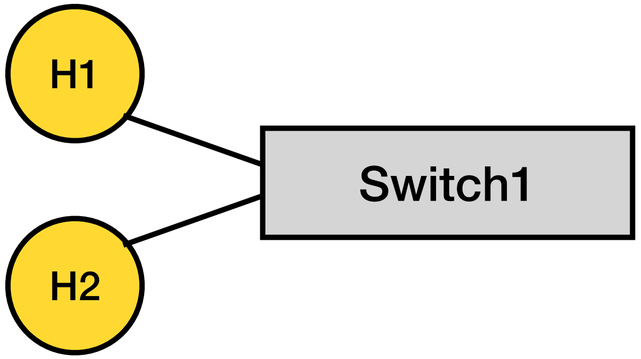
Cut-Through Switching

- The switch can forward the frame before the entire frame is fully received

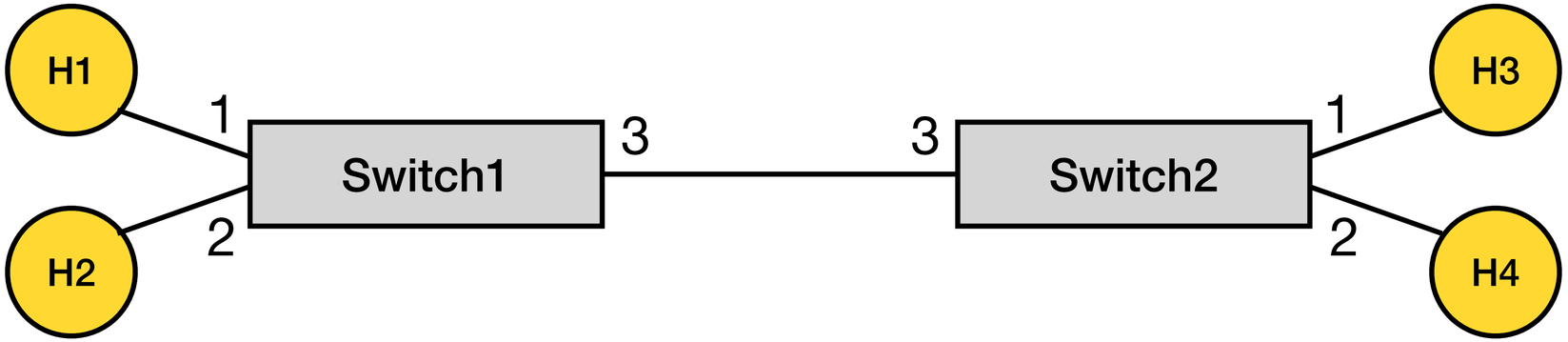


L2 Switching Enable Scaled Connectivity

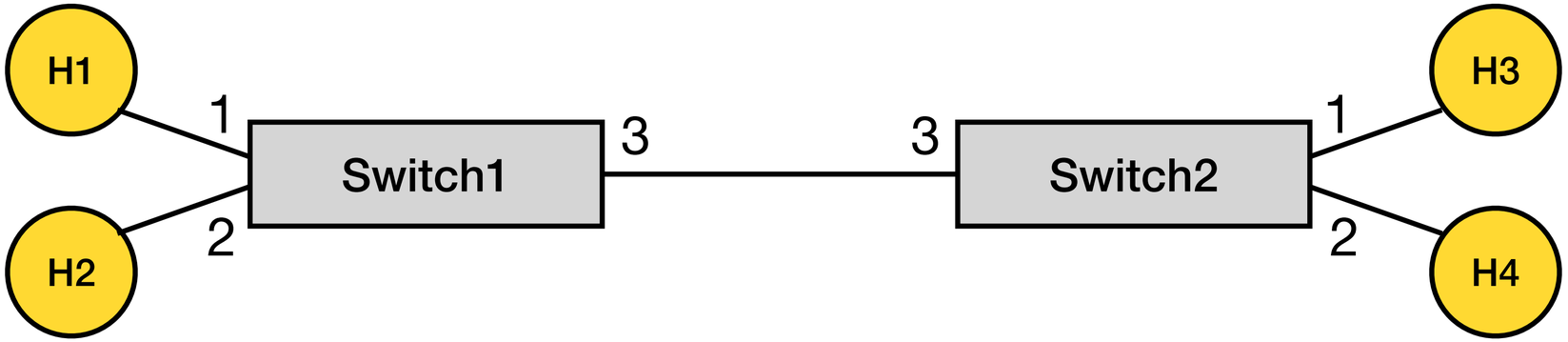
L2 Switching Enable Scaled Connectivity



L2 Switching Enable Scaled Connectivity



L2 Switching Enable Scaled Connectivity



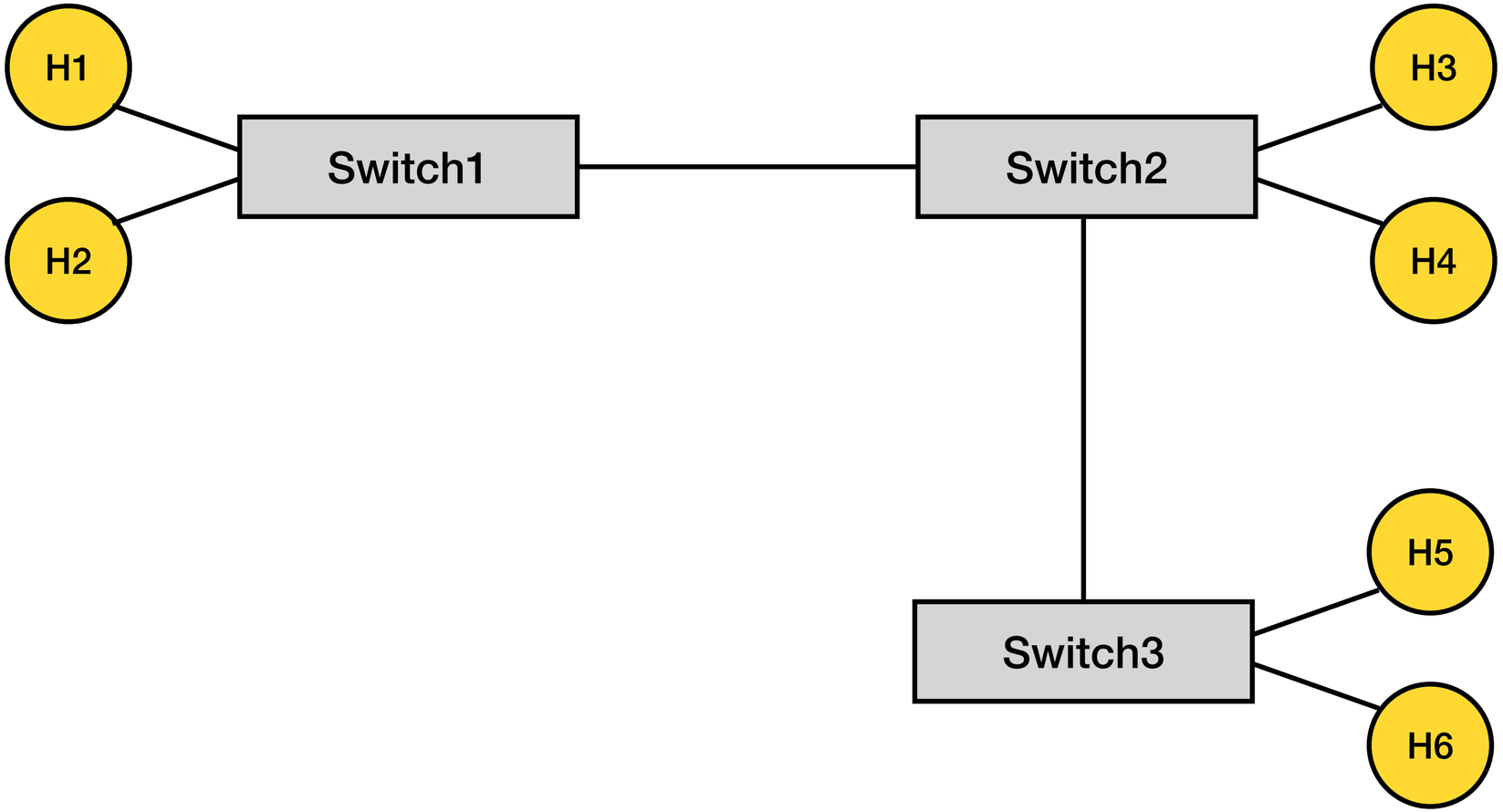
Switch1
Forwarding Table

MAC address	Port	Age (s)
H1 dst addr	1	8
H2 dst addr	2	6
H3 dst addr	3	8
H4 dst addr	3	6

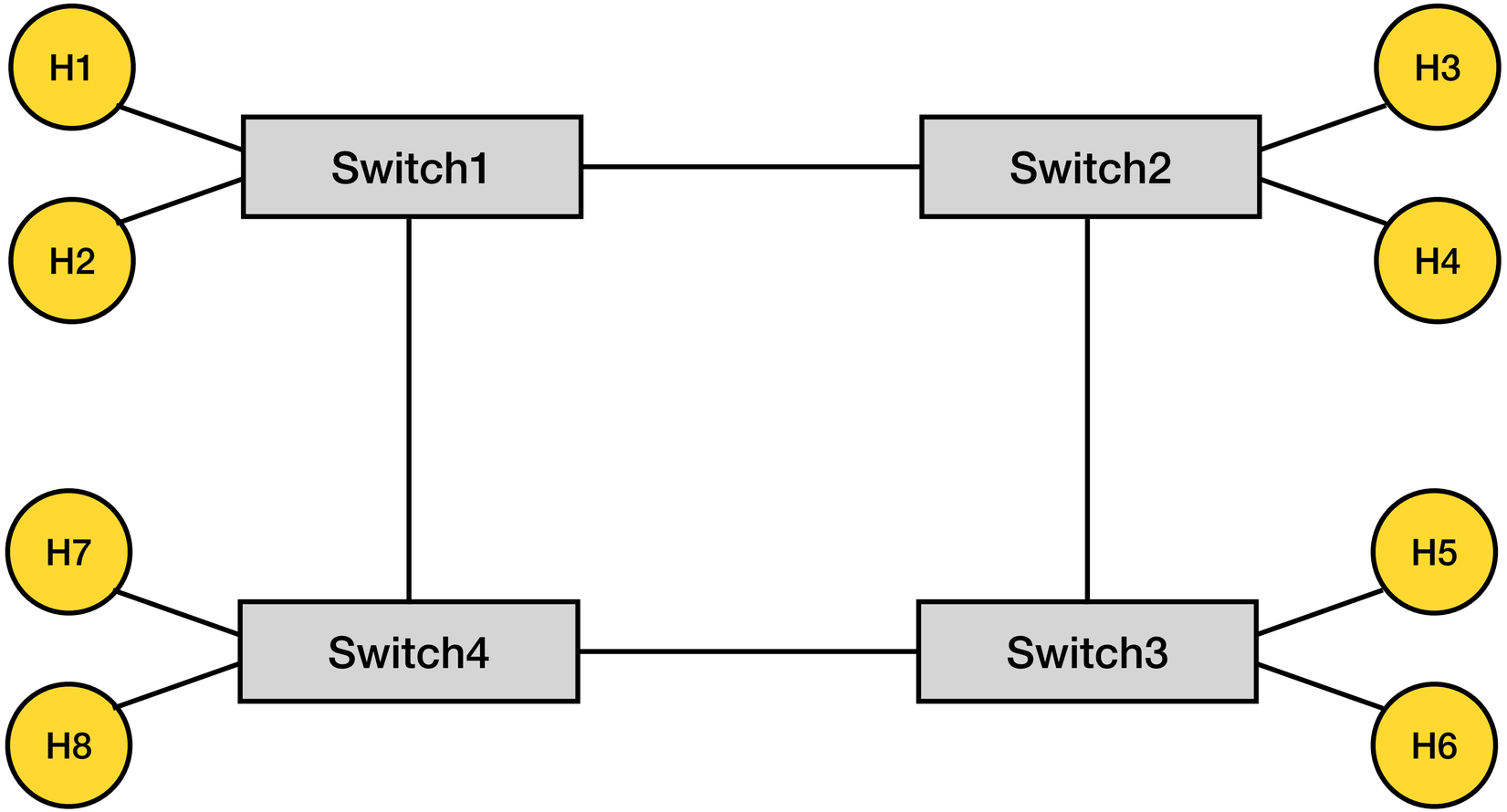
Switch2
Forwarding Table

MAC address	Port	Age (s)
H3 dst addr	1	8
H4 dst addr	2	6
H1 dst addr	3	8
H2 dst addr	3	6

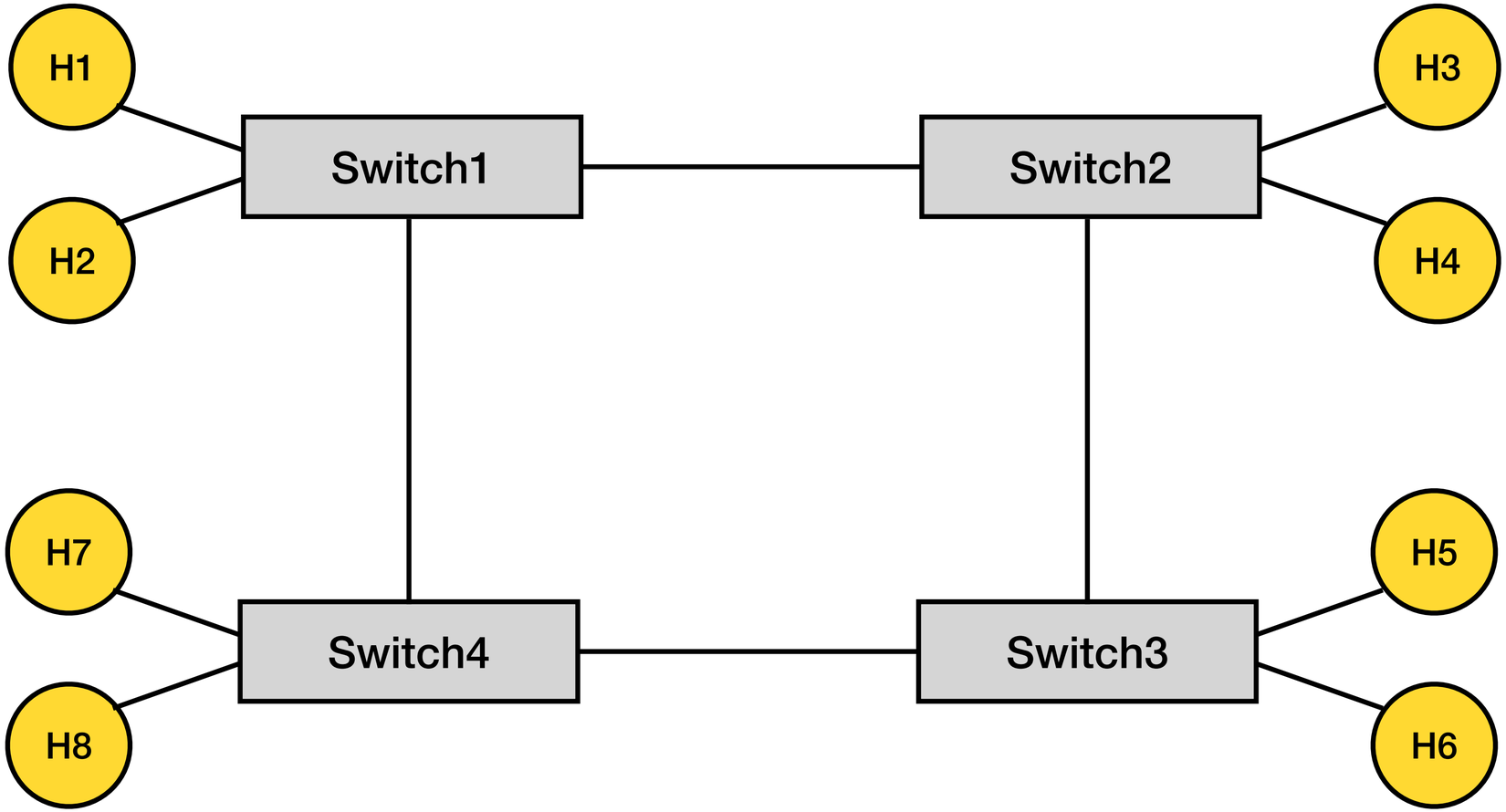
L2 Switching Enable Scaled Connectivity



L2 Switching Enable Scaled Connectivity



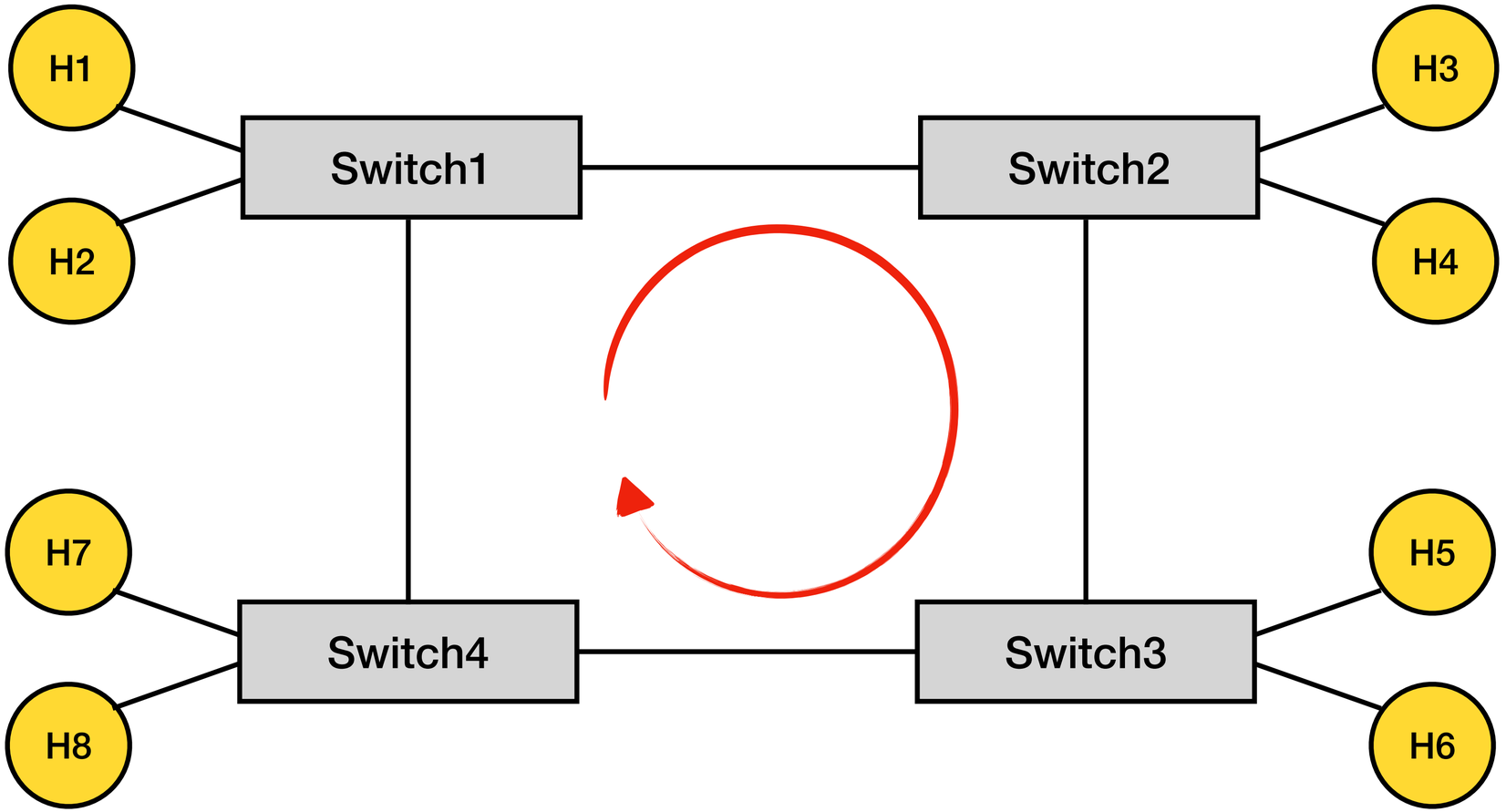
L2 Switching Enable Scaled Connectivity



Any problems?

Forwarding Loop

- A topological circle that keeps a frame forwarded
  - Waste switch/router bandwidth!



How can we avoid forwarding loops?

Spanning Tree Protocol (STP)

- A layer two protocol that detects and breaks loops
  - Invented by Radia Perlman from the Digital Equipment Corporation
  - Standardized as IEEE 802.1D

Spanning Tree Protocol (STP)

- A layer two protocol that detects and breaks loops
  - Invented by Radia Perlman from the Digital Equipment Corporation
  - Standardized as IEEE 802.1D

A protocol defines the format and the order of messages exchanged between two or more communication entities, as well as the actions taken on the transmission and/or receipt of a message or other event.

Spanning Tree Protocol (STP)

- A layer two protocol that detects and breaks loops
  - Invented by Radia Perlman from the Digital Equipment Corporation
  - Standardized as IEEE 802.1D

- Regarding the STP,
  - What are the protocol messages?
  - What are the actions associated with the message (protocol logic)?

Spanning Tree Protocol (STP)

- A layer two protocol that detects and breaks loops
  - Invented by Radia Perlman from the Digital Equipment Corporation
  - Standardized as IEEE 802.1D

- Regarding the STP,
  - What are the protocol messages?
  - What are the actions associated with the message (protocol logic)?

- **Key principle: minimal states**
  - Anytime and anywhere connectivity —> highly scalable systems
  - Not only for STP, but also for other protocols

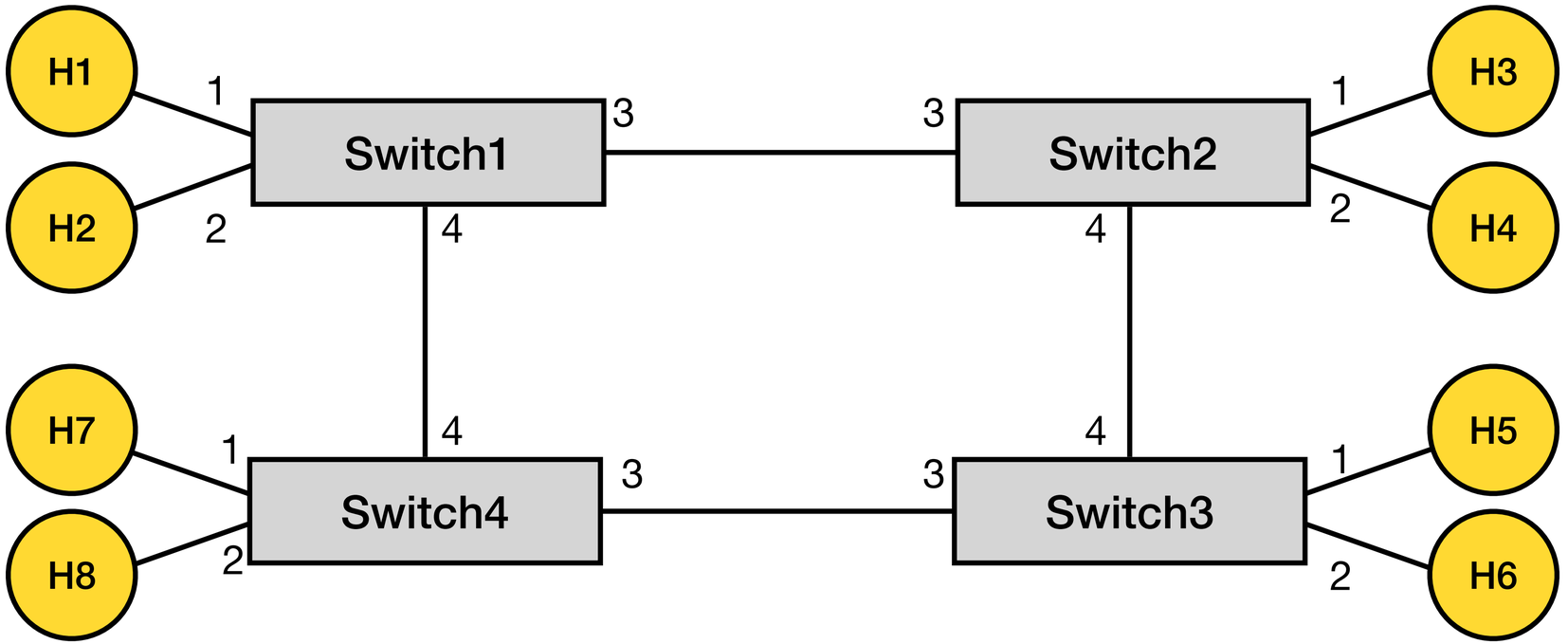
STP #1: States Maintained at the Switch

- Four states
  - Local switch ID —> Assigned by the network operator

STP #1: States Maintained at the Switch

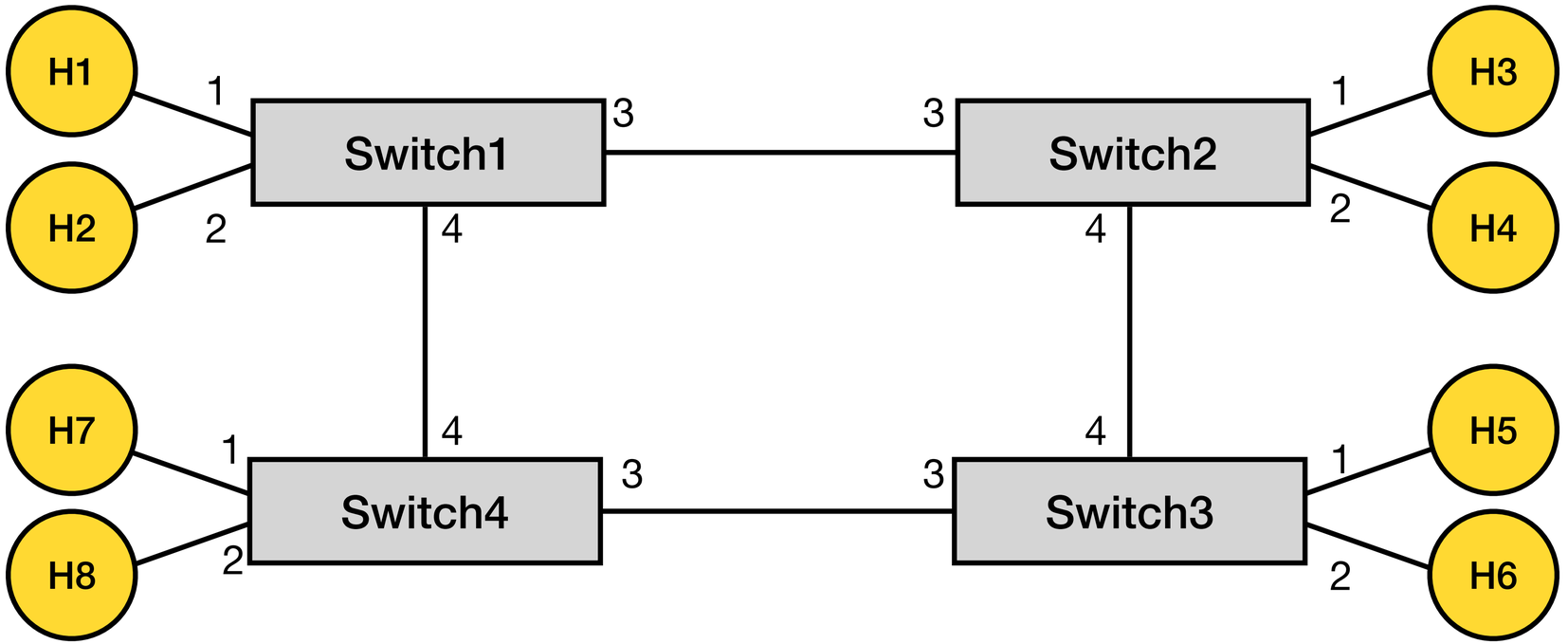
- Four states
  - Local switch ID —> Assigned by the network operator
  - **The switch ID of the root**
  - **The distance and port (i.e., the number of hops) to the root**
  - **Per-port action table**

} STP protocol



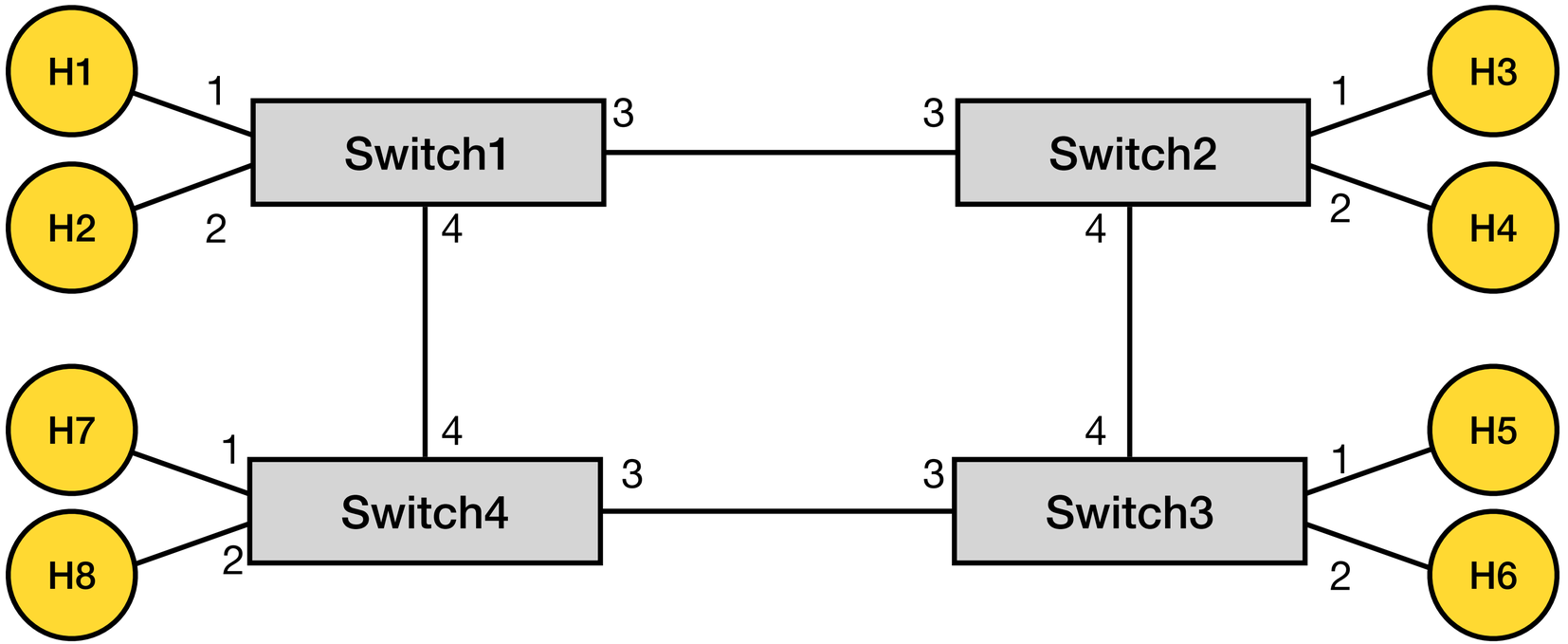
STP #1: States Maintained at the Switch

	Local Switch ID	Root Switch ID	<Hop#, port to root>	Port 1	Port 2	Port 3	Port4
Switch 1							
Switch 2							
Switch 3							
Switch 4							



STP #1: States Maintained at the Switch

	Local Switch ID	Root Switch ID	<Hop#, port to root>	Port 1	Port 2	Port 3	Port4
Switch 1	1						
Switch 2	2						
Switch 3	3						
Switch 4	4						

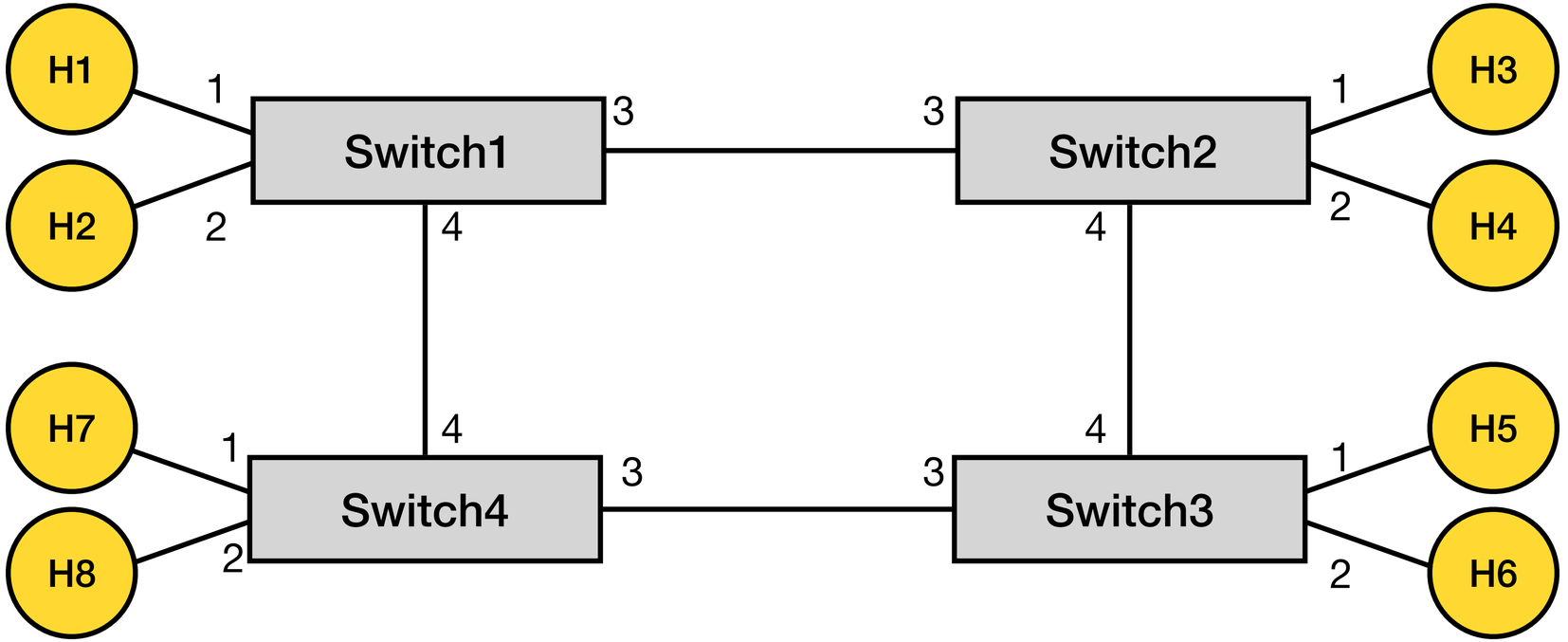


STP #2: Configuration Message

- Three tuples <Y, d, X>
  - Y: the root switch ID in my view
  - d: the distance to the root
  - X: my local switch ID
- The configuration message is issued periodically

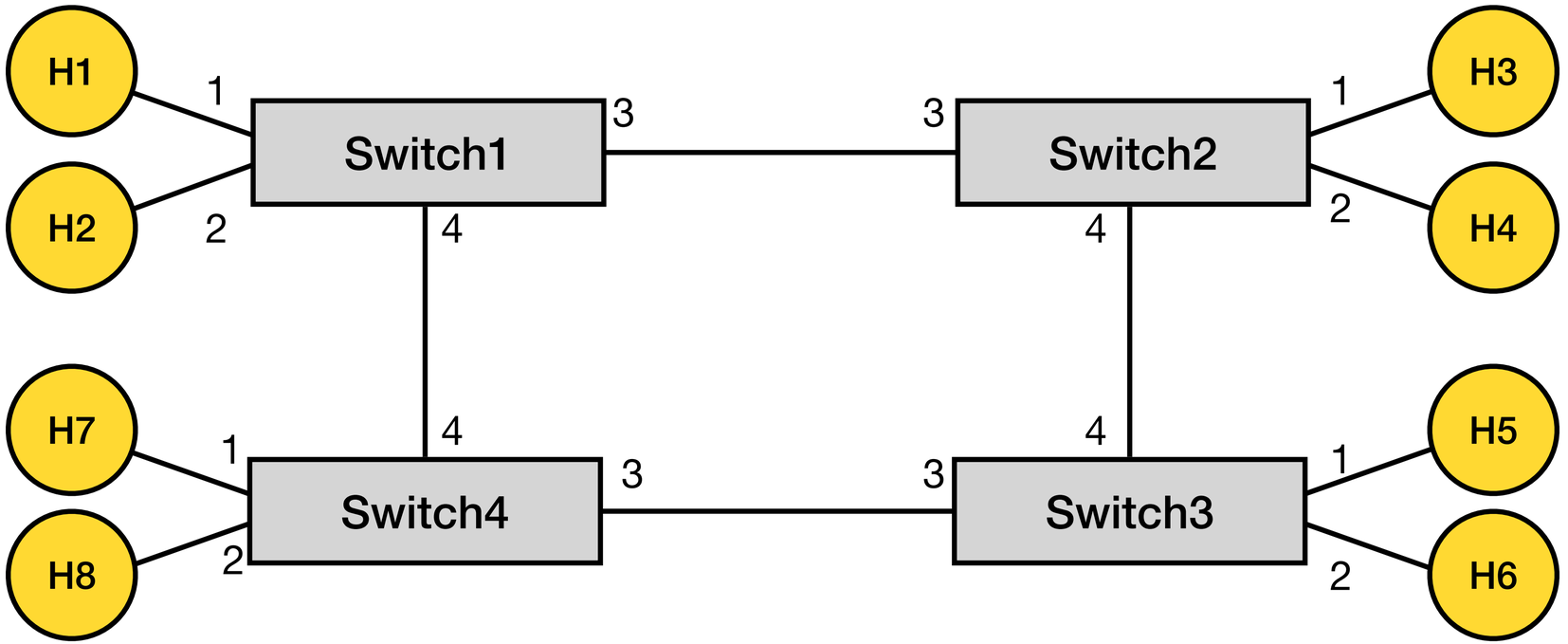
STP #2: Configuration Message

- Three tuples <Y, d, X>
  - Y: the root switch ID in my view
  - d: the distance to the root
  - X: my local switch ID
- The configuration message is issued periodically



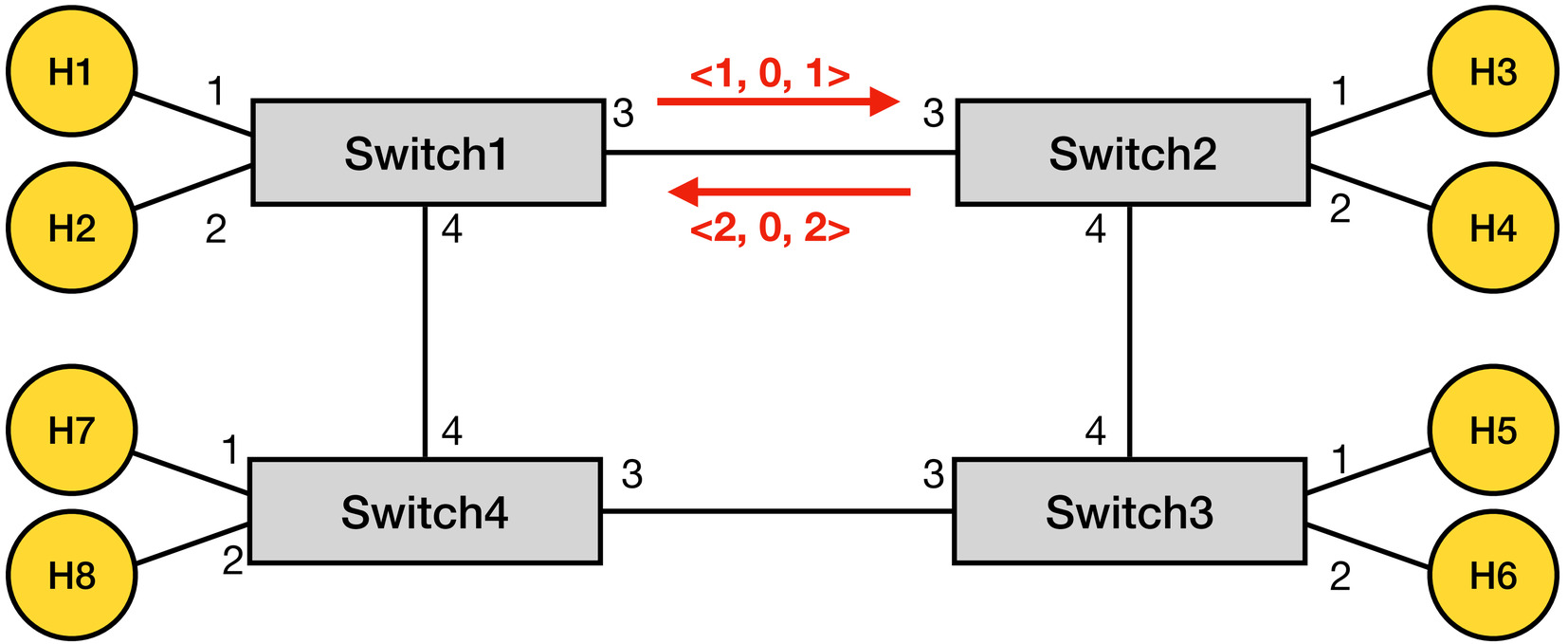
STP #2: Configuration Message

	Local Switch ID	Root Switch ID	<Hop#, port to root>	Port 1	Port 2	Port 3	Port4
Switch 1	1						
Switch 2	2						
Switch 3	3						
Switch 4	4						



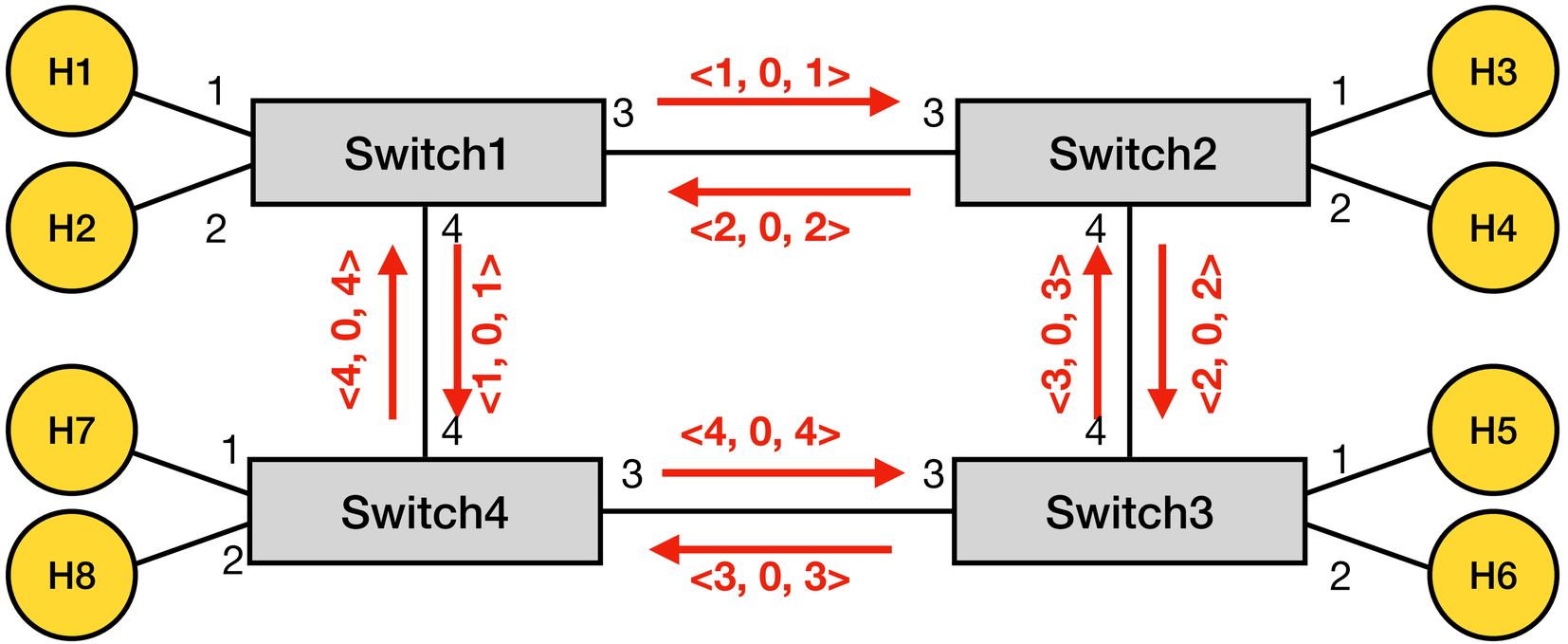
STP #2: Configuration Message

	Local Switch ID	Root Switch ID	<Hop#, port to root>	Port 1	Port 2	Port 3	Port4
Switch 1	1	1	<0, N/A>				
Switch 2	2	2	<0, N/A>				
Switch 3	3	3	<0, N/A>				
Switch 4	4	4	<0, N/A>				



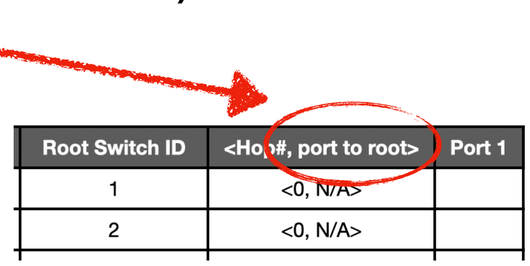
STP #2: Configuration Message

	Local Switch ID	Root Switch ID	<Hop#, port to root>	Port 1	Port 2	Port 3	Port4
Switch 1	1	1	<0, N/A>				
Switch 2	2	2	<0, N/A>				
Switch 3	3	3	<0, N/A>				
Switch 4	4	4	<0, N/A>				



STP #3: Protocol Actions

- Action #1: Root determination
  - If the root switch ID of the configuration message (<Y, d, X>) is **smaller than** the root switch ID of my local states, the switch should accept the new root switch and perform the following four operations:
    - Change my root switch ID to **Y**
    - Update the hop# (d_cur) to **d_cur = d+1**
    - Mark the switch port that receives the configuration message as **"Broadcast_YES (BC_YES)"**
    - Mark the prior saved switch port (if it existed) as **"Broadcast_NO (BC_NO)"**
  - Otherwise, go to Action #2



STP #3: Protocol Actions

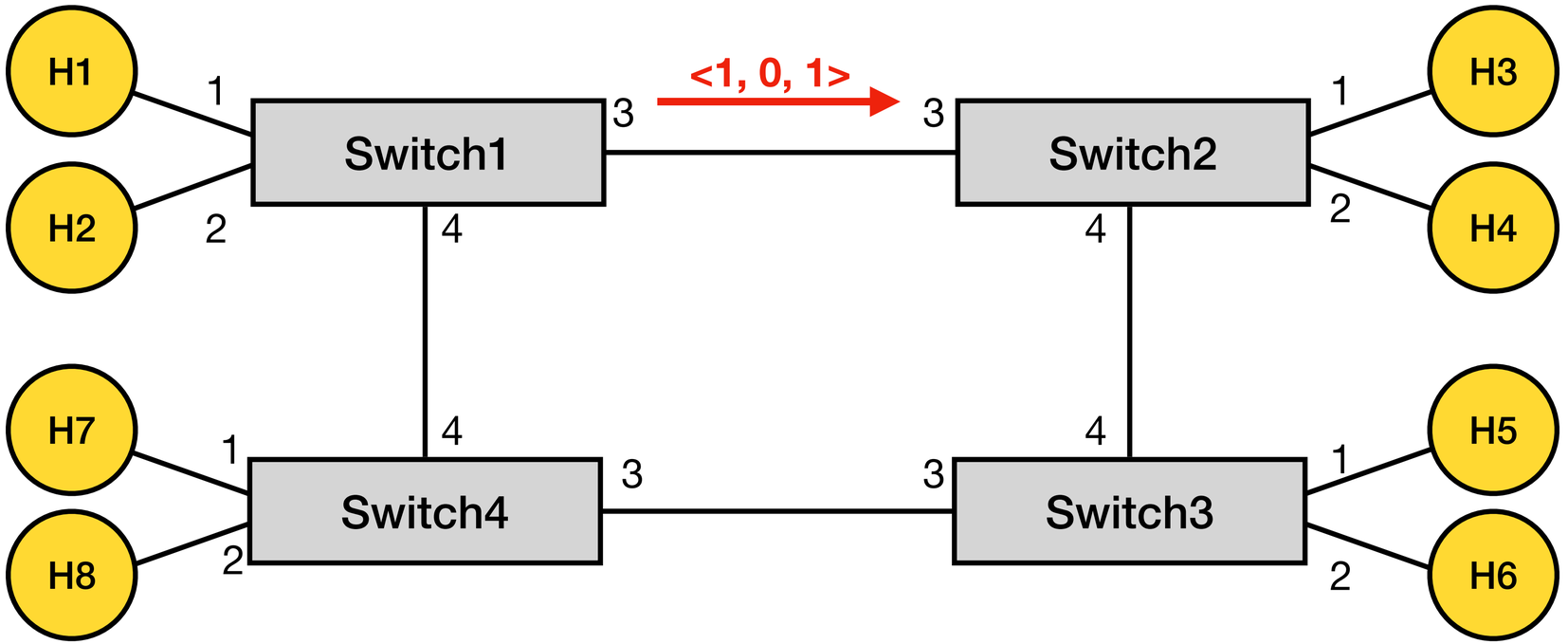
- Action #2: Path determination
  - If the root switch ID of the configuration message (<Y, d, X>) is **the same as** the root switch ID of my local states, the switch should examine the hop# to figure out the shortest path and perform the following operations:
    - If **d+1 < d_cur**, the switch should (a) update the hop# (d_cur) to **d_cur = d+1**; (b) mark the switch port that receives the configuration message as **"Broadcast_YES (BC_YES)"**; (c) mark the prior saved switch port (if it existed) as **"Broadcast_NO (BC_NO)"**;
    - If **d+1 >= d_cur**, the switch should (a) discard the message; (b) mark the switch port that receives the configuration message as **"Broadcast_NO (BC_NO)"**. The root switch skips (b) and marks the port as **"Broadcast_YES (BC_YES)**;
  - Otherwise, go to Action #3

STP #3: Protocol Actions

- Action #3: Discard and block
  - The switch should (a) discard the message: (b) mark the switch port that receives the configuration message as **“Broadcast_NO (BC_NO)”**. If this is the root switch, (b) is skipped and the switch marks the port as **“Broadcast_YES (BC_YES)**;

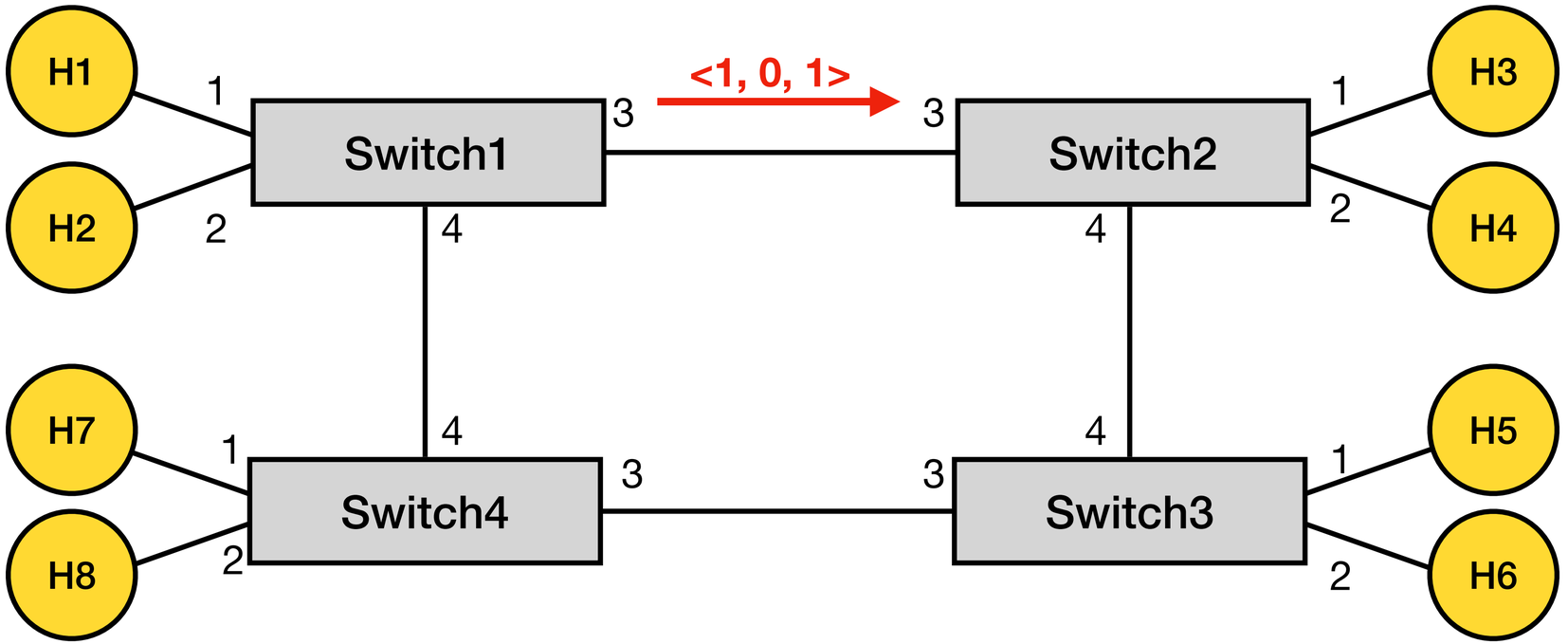
A Running Example

	Local Switch ID	Root Switch ID	<Hop#, port to root>	Port 1	Port 2	Port 3	Port4
Switch 1	1	1	<0, N/A>				
Switch 2	2	2	<0, N/A>				
Switch 3	3	3	<0, N/A>				
Switch 4	4	4	<0, N/A>				



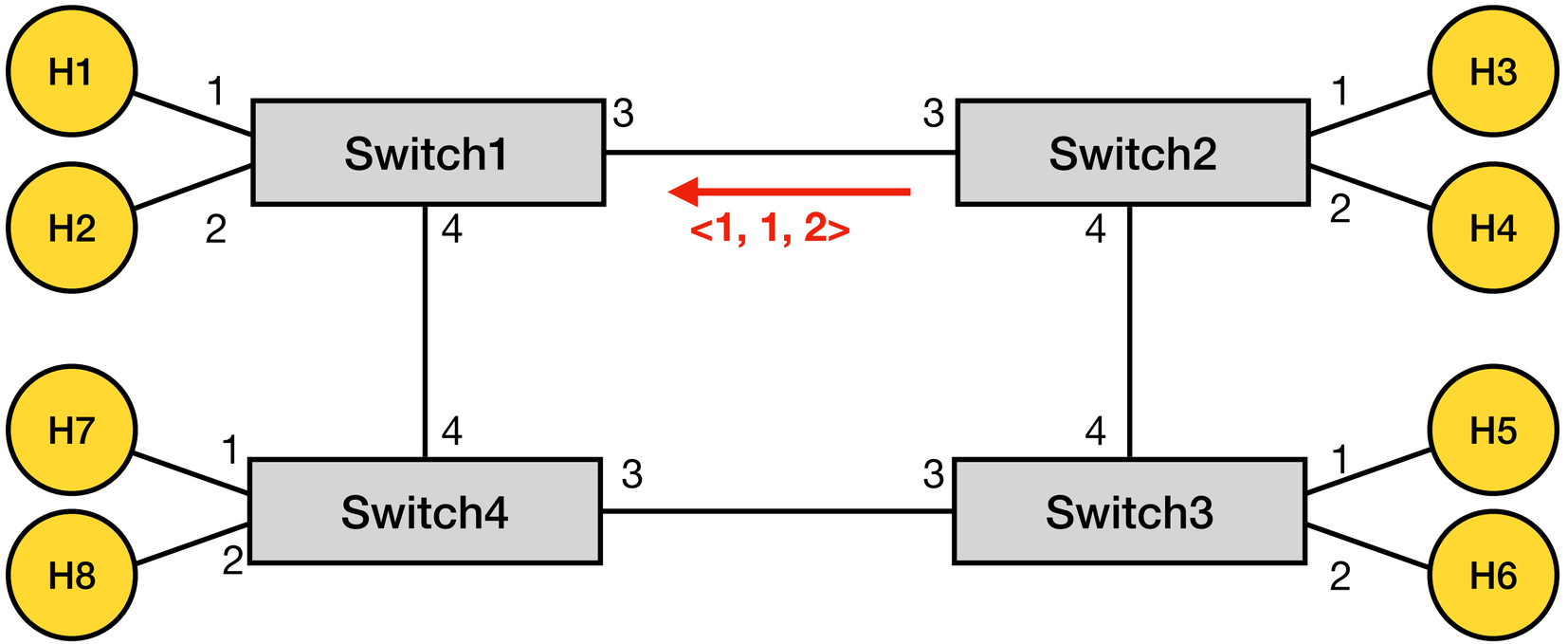
A Running Example

	Local Switch ID	Root Switch ID	<Hop#, port to root>	Port 1	Port 2	Port 3	Port4
Switch 1	1	1	<0, N/A>				
Switch 2	2	1	<1, 3>			BC_YES	
Switch 3	3	3	<0, N/A>				
Switch 4	4	4	<0, N/A>				



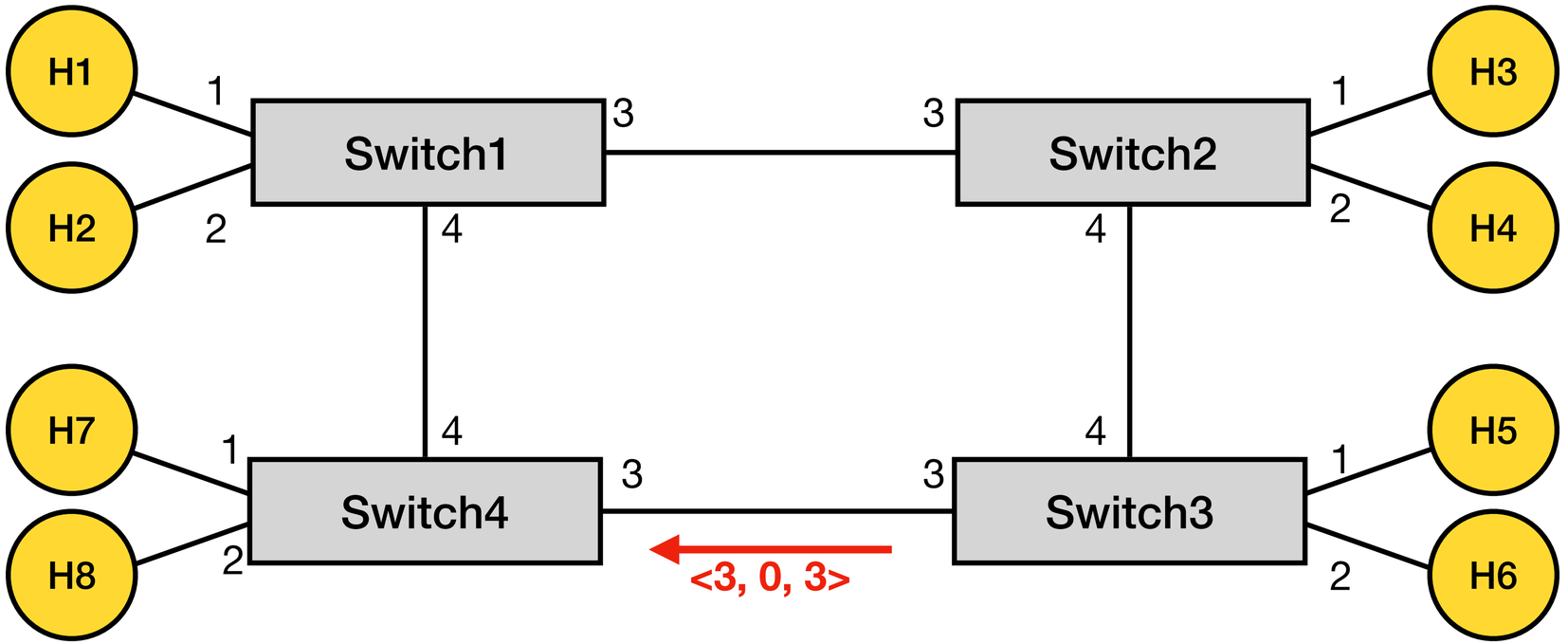
A Running Example

	Local Switch ID	Root Switch ID	<Hop#, port to root>	Port 1	Port 2	Port 3	Port4
Switch 1	1	1	<0, N/A>			BC_YES	
Switch 2	2	1	<1, 3>			BC_YES	
Switch 3	3	3	<0, N/A>				
Switch 4	4	4	<0, N/A>				



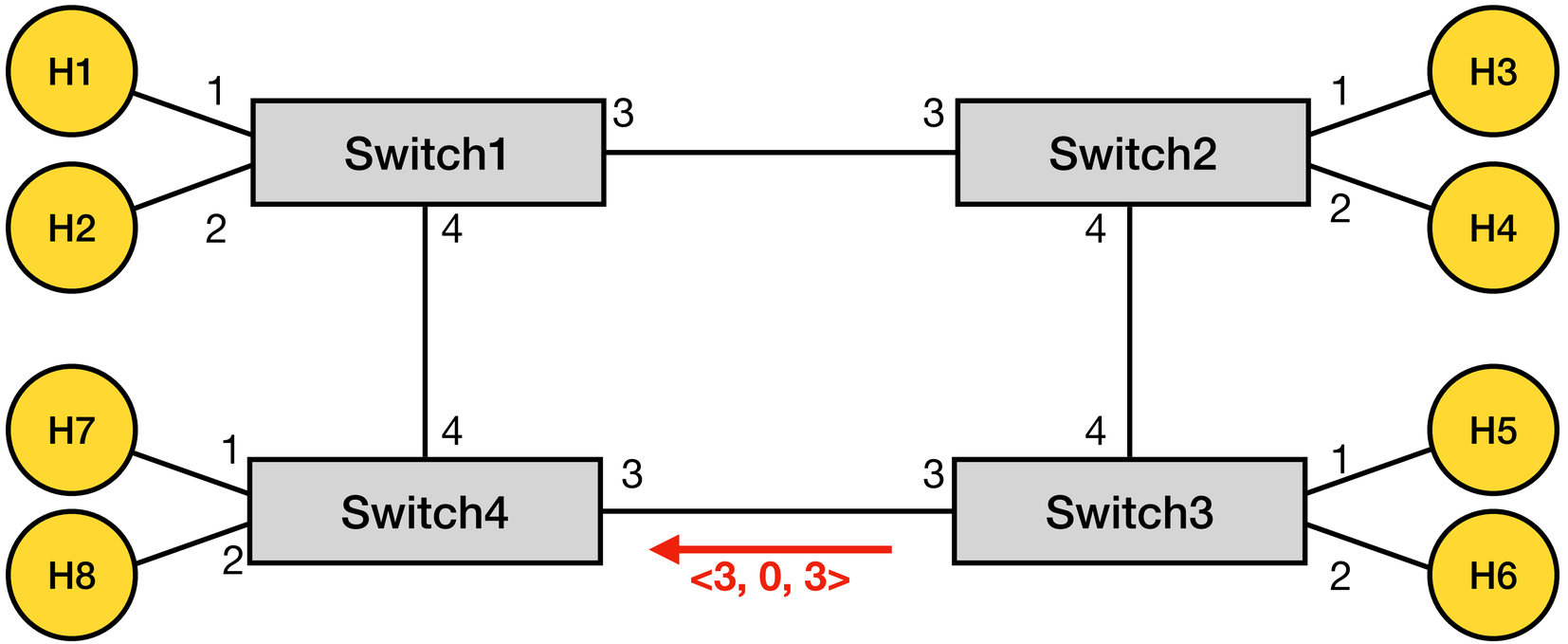
A Running Example

	Local Switch ID	Root Switch ID	<Hop#, port to root>	Port 1	Port 2	Port 3	Port4
Switch 1	1	1	<0, N/A>			BC_YES	
Switch 2	2	1	<1, 3>			BC_YES	
Switch 3	3	3	<0, N/A>				
Switch 4	4	4	<0, N/A>				



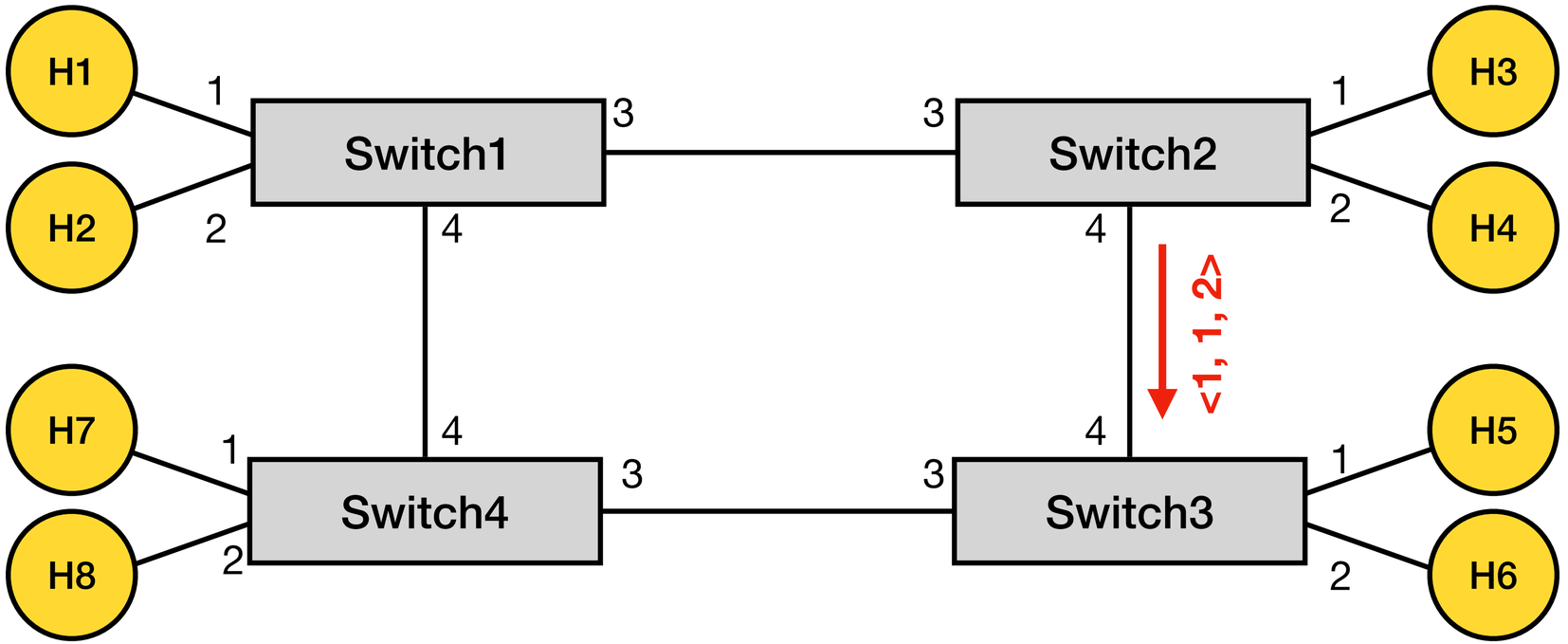
A Running Example

	Local Switch ID	Root Switch ID	<Hop#, port to root>	Port 1	Port 2	Port 3	Port4
Switch 1	1	1	<0, N/A>			BC_YES	
Switch 2	2	1	<1, 3>			BC_YES	
Switch 3	3	3	<0, N/A>				
Switch 4	4	3	<1, 3>			BC_YES	



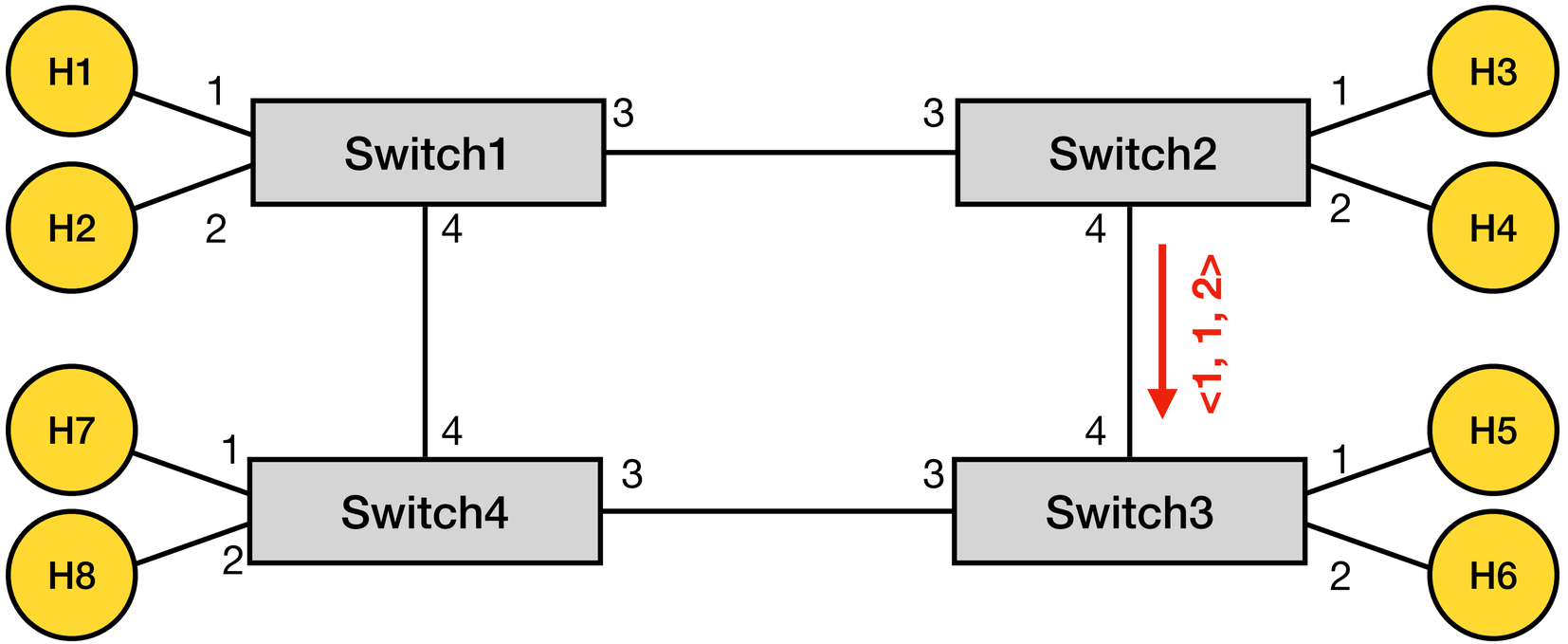
A Running Example

	Local Switch ID	Root Switch ID	<Hop#, port to root>	Port 1	Port 2	Port 3	Port4
Switch 1	1	1	<0, N/A>			BC_YES	
Switch 2	2	1	<1, 3>			BC_YES	
Switch 3	3	3	<0, N/A>				
Switch 4	4	3	<1, 3>			BC_YES	



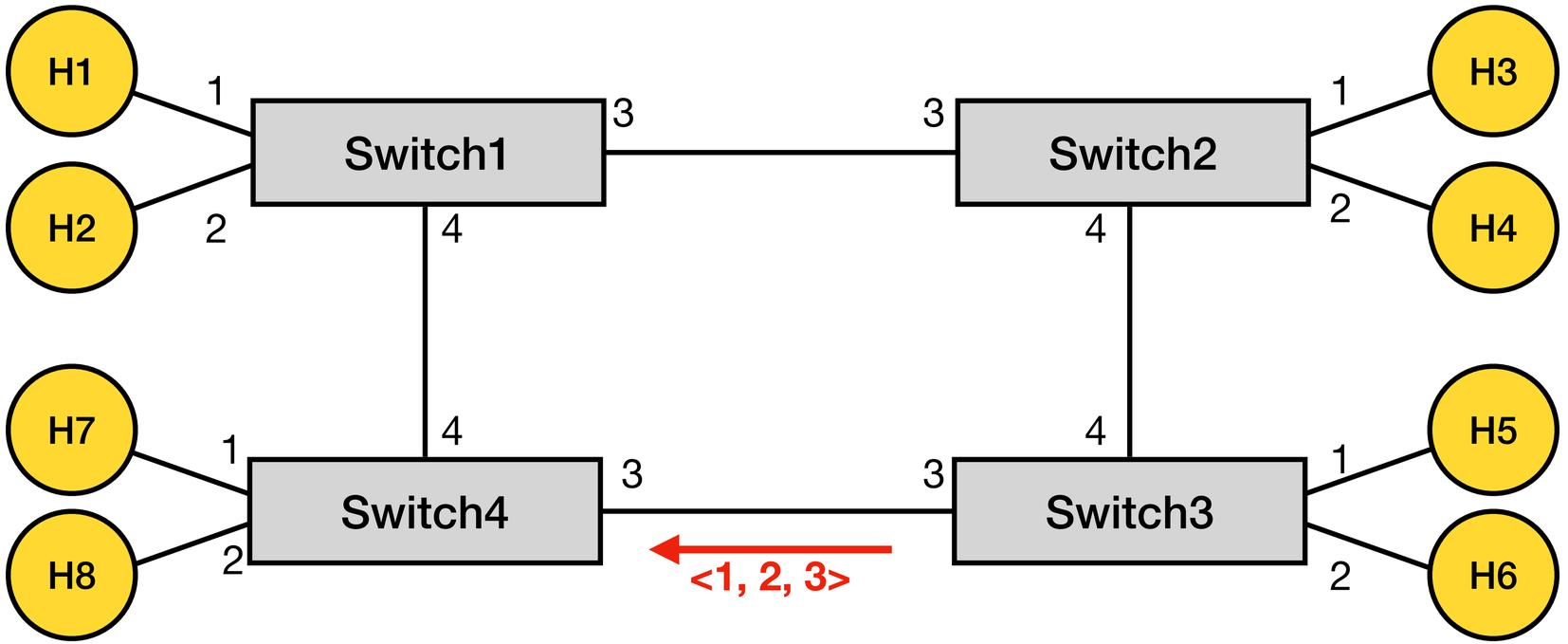
A Running Example

	Local Switch ID	Root Switch ID	<Hop#, port to root>	Port 1	Port 2	Port 3	Port4
Switch 1	1	1	<0, N/A>			BC_YES	
Switch 2	2	1	<1, 3>			BC_YES	
Switch 3	3	1	<2, 4>				BC_YES
Switch 4	4	3	<1, 3>			BC_YES	



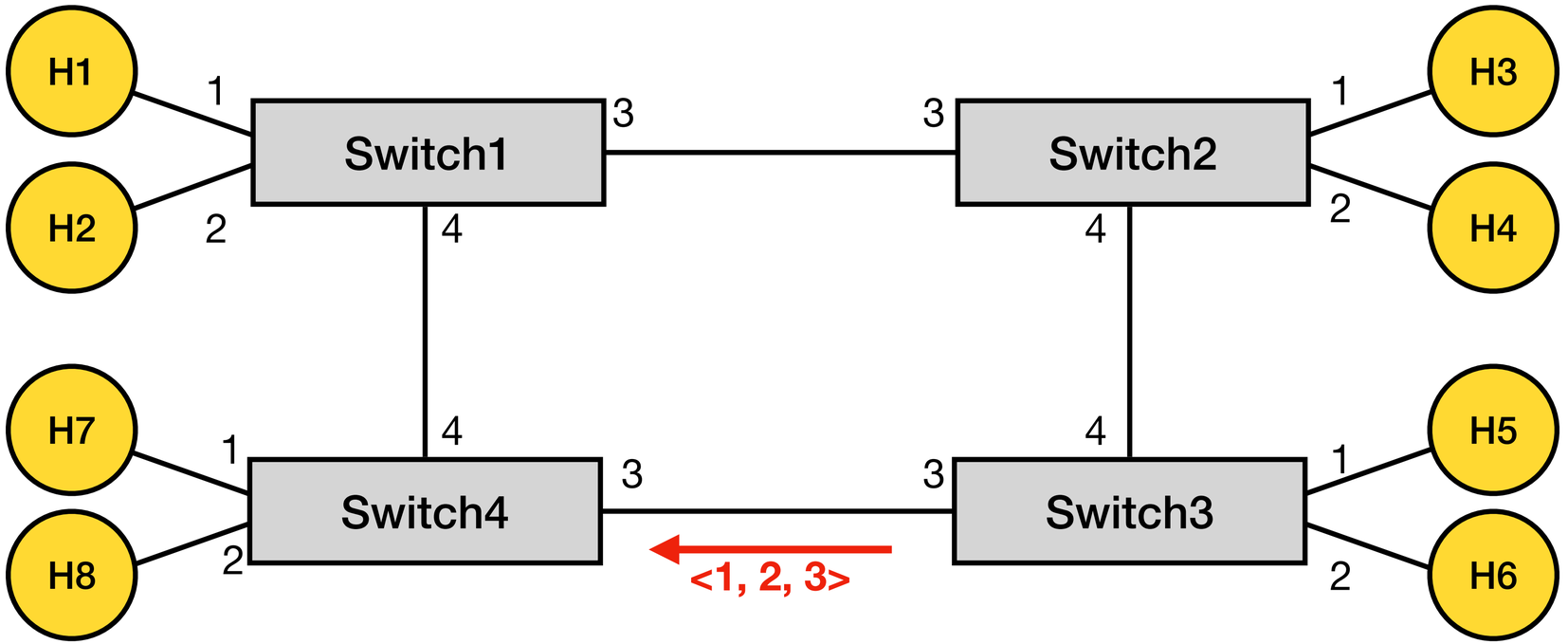
A Running Example

	Local Switch ID	Root Switch ID	<Hop#, port to root>	Port 1	Port 2	Port 3	Port4
Switch 1	1	1	<0, N/A>			BC_YES	
Switch 2	2	1	<1, 3>			BC_YES	
Switch 3	3	1	<2, 4>				BC_YES
Switch 4	4	3	<1, 3>			BC_YES	



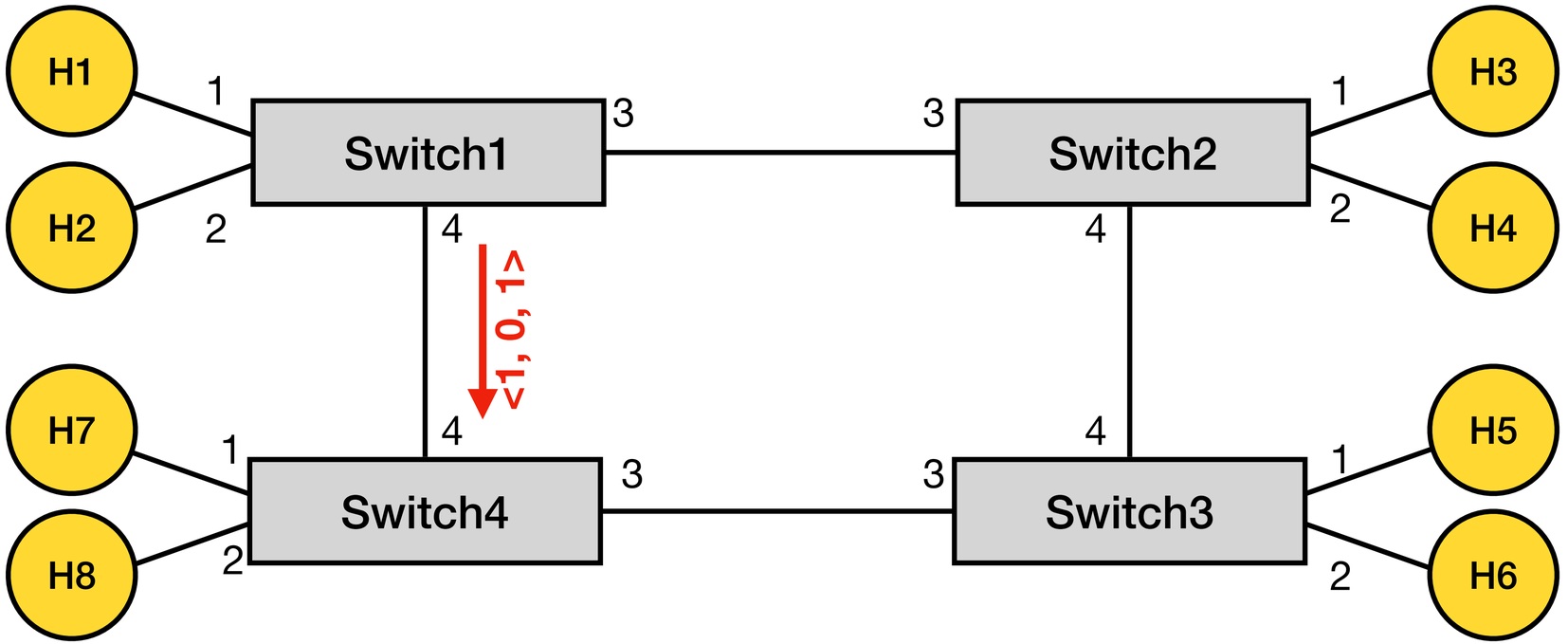
A Running Example

	Local Switch ID	Root Switch ID	<Hop#, port to root>	Port 1	Port 2	Port 3	Port4
Switch 1	1	1	<0, N/A>			BC_YES	
Switch 2	2	1	<1, 3>			BC_YES	
Switch 3	3	1	<2, 4>				BC_YES
Switch 4	4	1	<3, 3>			BC_YES	



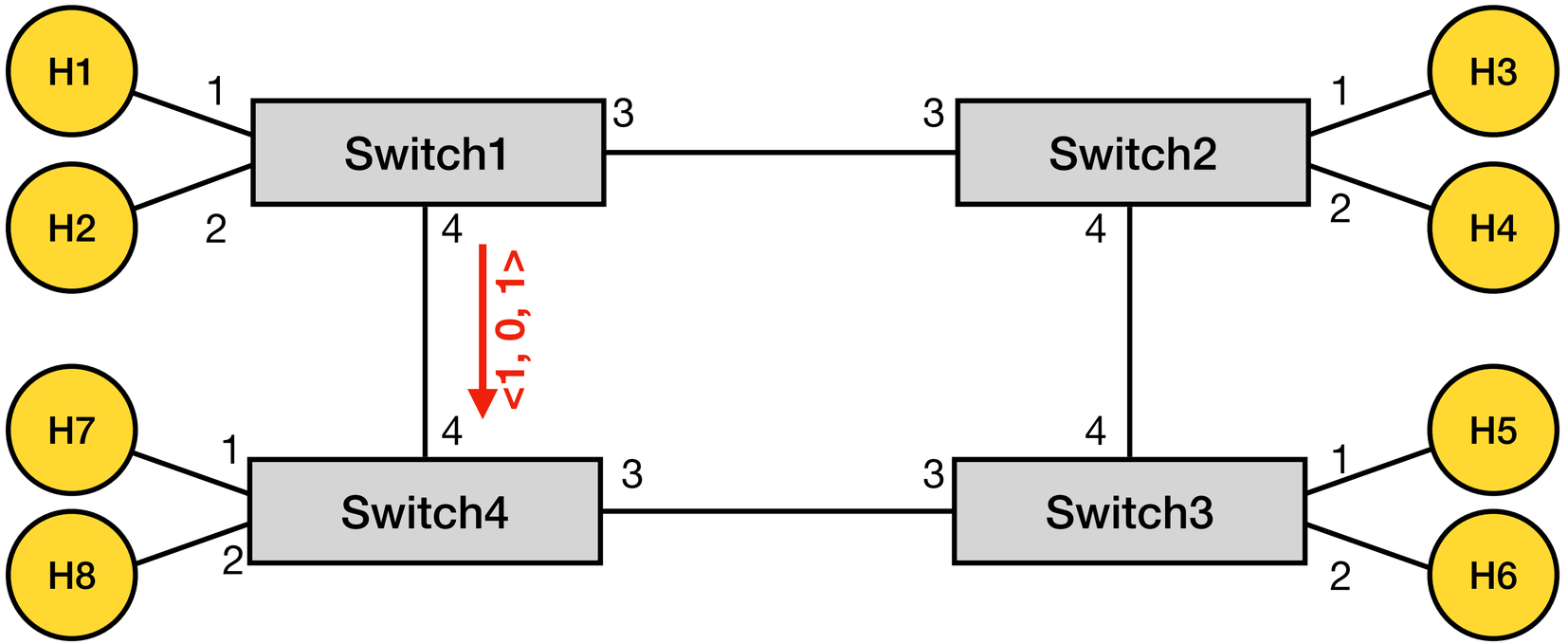
A Running Example

	Local Switch ID	Root Switch ID	<Hop#, port to root>	Port 1	Port 2	Port 3	Port4
Switch 1	1	1	<0, N/A>			BC_YES	
Switch 2	2	1	<1, 3>			BC_YES	
Switch 3	3	1	<2, 4>				BC_YES
Switch 4	4	1	<3, 3>			BC_YES	



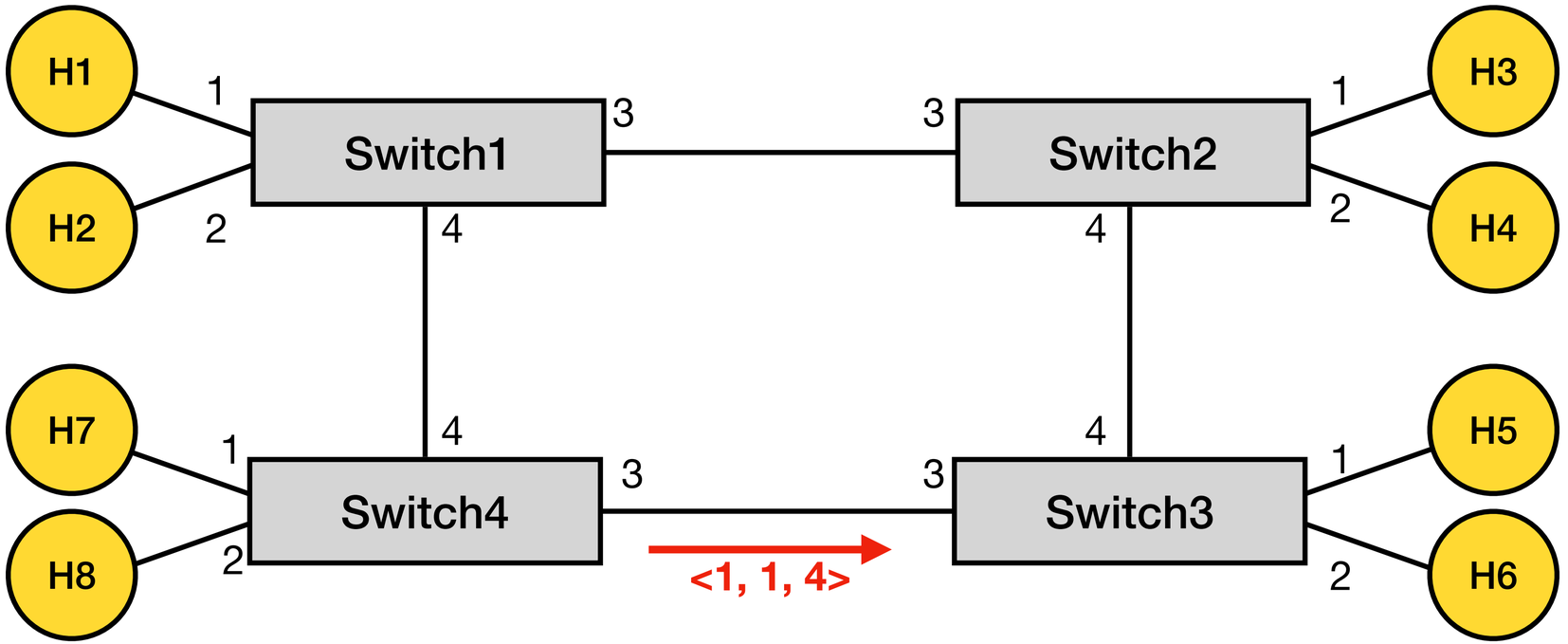
A Running Example

	Local Switch ID	Root Switch ID	<Hop#, port to root>	Port 1	Port 2	Port 3	Port4
Switch 1	1	1	<0, N/A>			BC_YES	
Switch 2	2	1	<1, 3>			BC_YES	
Switch 3	3	1	<2, 4>				BC_YES
Switch 4	4	1	<1, 4>			BC_NO	BC_YES



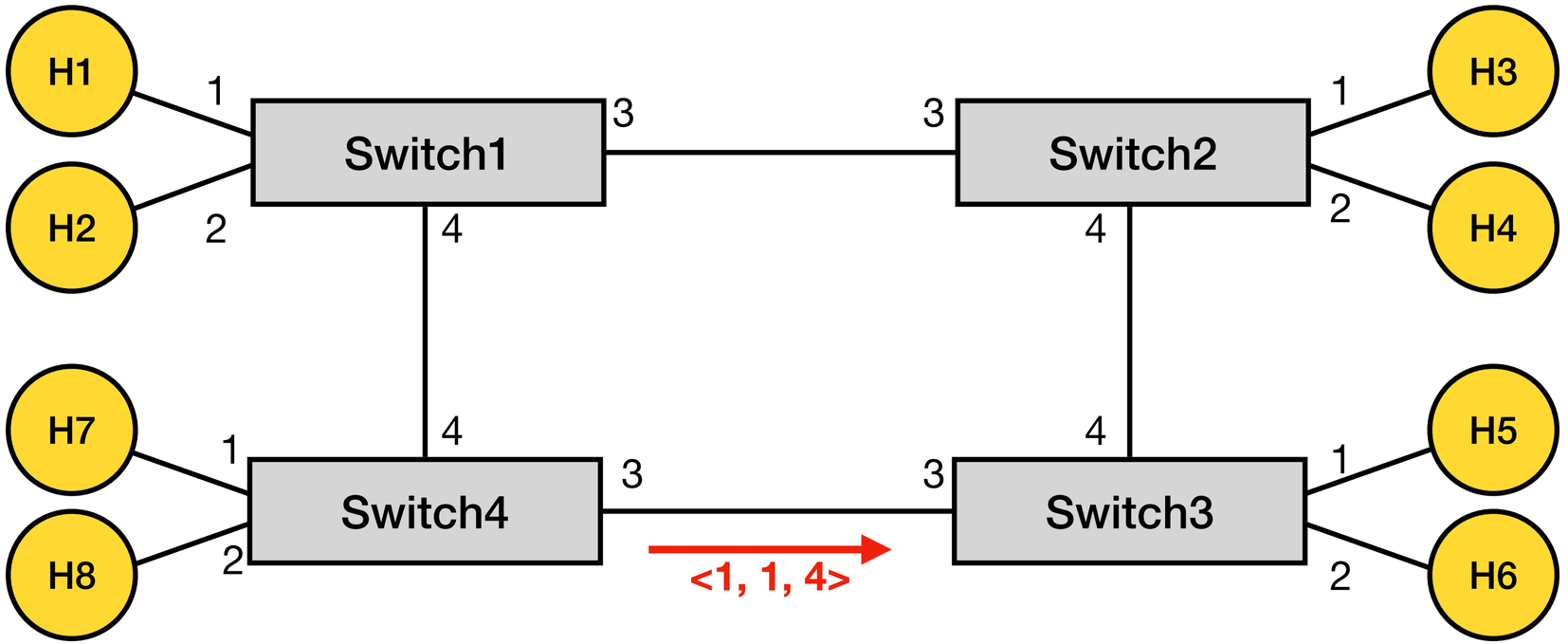
A Running Example

	Local Switch ID	Root Switch ID	<Hop#, port to root>	Port 1	Port 2	Port 3	Port4
Switch 1	1	1	<0, N/A>			BC_YES	
Switch 2	2	1	<1, 3>			BC_YES	
Switch 3	3	1	<2, 4>				BC_YES
Switch 4	4	1	<1, 4>			BC_NO	BC_YES



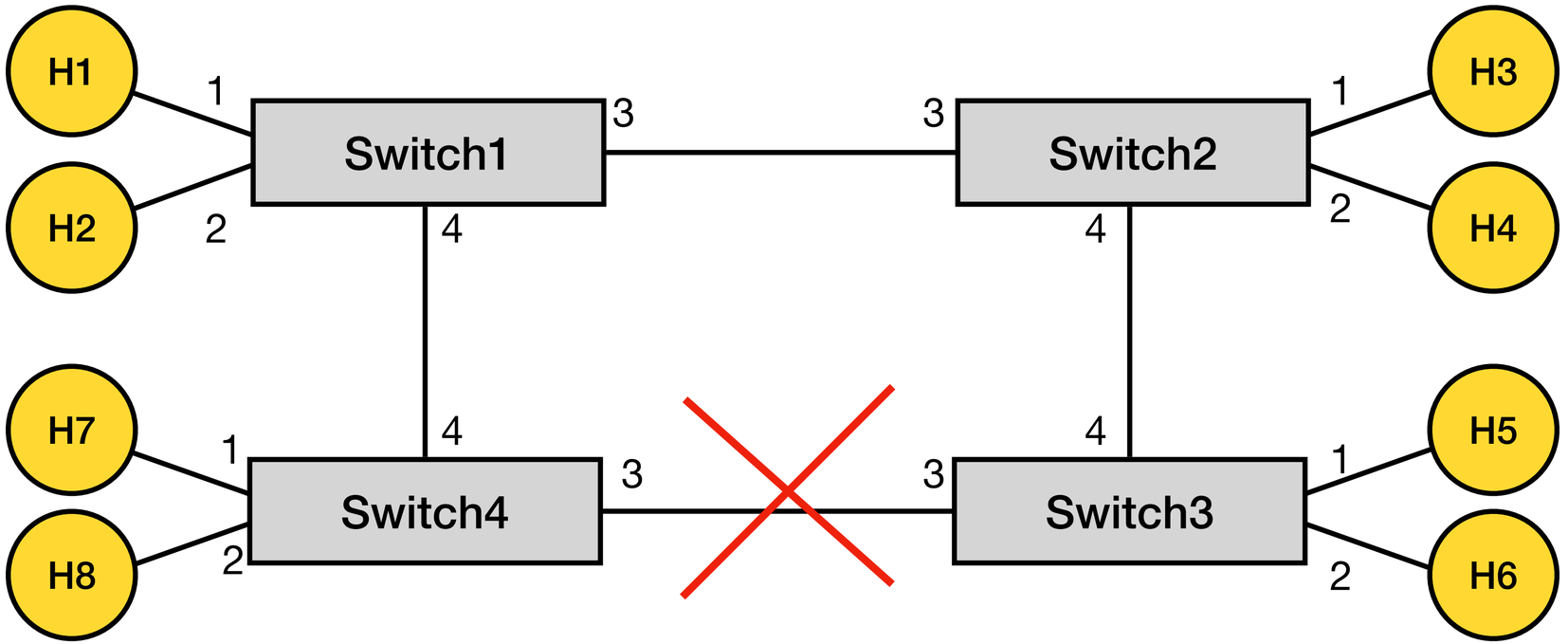
A Running Example

	Local Switch ID	Root Switch ID	<Hop#, port to root>	Port 1	Port 2	Port 3	Port4
Switch 1	1	1	<0, N/A>			BC_YES	
Switch 2	2	1	<1, 3>			BC_YES	
Switch 3	3	1	<2, 4>			BC_NO	BC_YES
Switch 4	4	1	<1, 4>			BC_NO	BC_YES



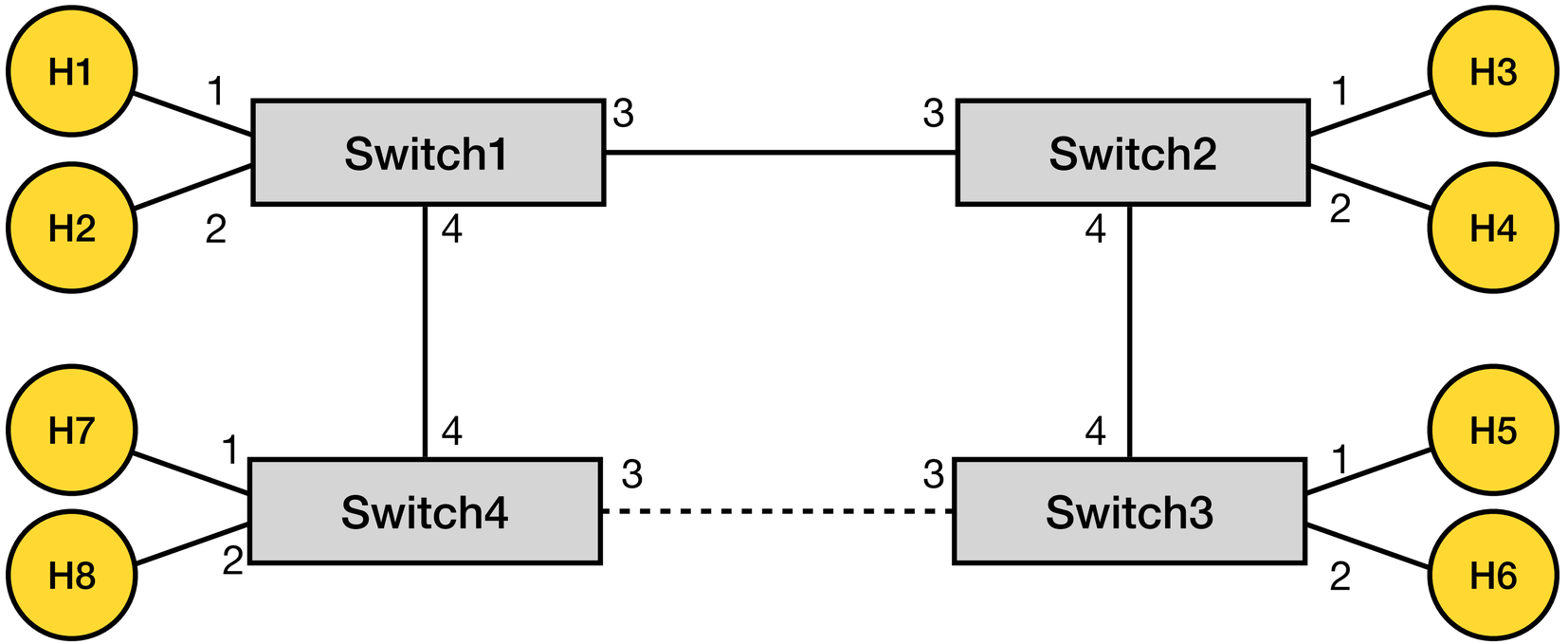
A Running Example

	Local Switch ID	Root Switch ID	<Hop#, port to root>	Port 1	Port 2	Port 3	Port4
Switch 1	1	1	<0, N/A>			BC_YES	
Switch 2	2	1	<1, 3>			BC_YES	
Switch 3	3	1	<2, 4>			BC_NO	BC_YES
Switch 4	4	1	<1, 4>			BC_NO	BC_YES



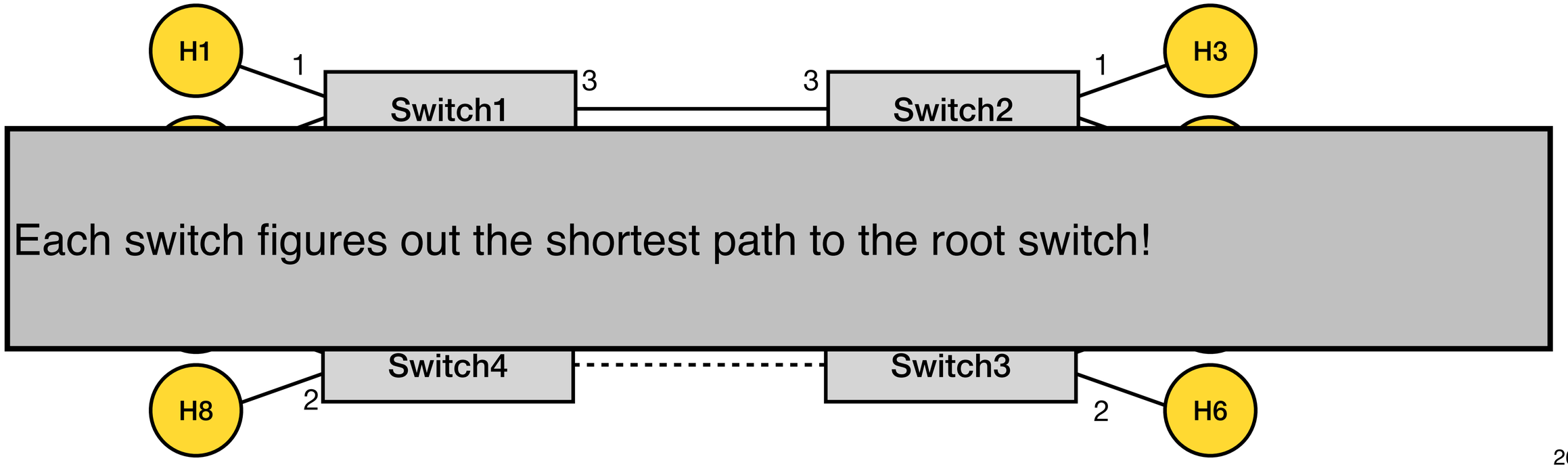
A Running Example

	Local Switch ID	Root Switch ID	<Hop#, port to root>	Port 1	Port 2	Port 3	Port4
Switch 1	1	1	<0, N/A>			BC_YES	BC_YES
Switch 2	2	1	<1, 3>			BC_YES	BC_YES
Switch 3	3	1	<2, 4>			BC_NO	BC_YES
Switch 4	4	1	<1, 4>			BC_NO	BC_YES



A Running Example

	Local Switch ID	Root Switch ID	<Hop#, port to root>	Port 1	Port 2	Port 3	Port4
Switch 1	1	1	<0, N/A>			BC_YES	BC_YES
Switch 2	2	1	<1, 3>			BC_YES	BC_YES
Switch 3	3	1	<2, 4>			BC_NO	BC_YES
Switch 4	4	1	<1, 4>			BC_NO	BC_YES



STP Discussion

- STP changes a topological graph to a tree
  - Blocking a switching port on broadcasting

- STP runs periodically
  - When a switch recovers from a failure, it restarts from the scratch
  - The states of a switch keep updating when the tree structure changes

Summary

- Today
  - L2 Forwarding
  - L2 mac learning
  - STP

- Next lecture
  - Ethernet
  - Quiz