# Module 2
# Transparent Switches

# Transparent Switches

- A switch in which the **stations are completely unaware of the switch's existence**.
- If a switch is **added or deleted** from the system, *reconfiguration of the stations is unnecessary*.
- According to the **IEEE 802.1d specification**, a system equipped with transparent switches must **meet three criteria**:
  - Frames must be **forwarded** from one station to another.
  - The **forwarding table is automatically made** by learning frame movements in the network.
  - **Loops** in the system **must be prevented**.

# Transparent Switches

- Forwarding
  - A transparent switch must **correctly forward the frames.**

- Learning
  - The ***earliest switches*** had ***switching tables*** that ***were static***.
    - The ***system administrator*** would ***manually*** enter ***each table entry*** during switch setup.
    - Although the process was simple, it was not practical.
  - If a station was added or deleted, the table had to be modified manually.
  - The same was true **if a station's MAC address changed**, which is not a rare event.
    - For example, putting in a new network card means a new MAC address.

# Transparent Switches

- **Learning Switches**
  - A better solution to the static table is a **dynamic table** that **maps addresses to ports (interfaces) automatically.**
  - To make a table dynamic, **we need a switch** that **gradually learns from the frames' movements**.
  - To do this, the **switch inspects both the destination and the source addresses** in **each frame that passes** through the switch.
    - The **destination address** is used for the **forwarding decision** (table lookup);
    - The **source address** is used for **adding entries to the table** and **for updating purposes.**

# Transparent Switches

- Learning Switches

**Gradual building of table**

| Address | Port |
| --- | --- |

a. Original

| Address | Port |
| --- | --- |
| 71:2B:13:45:61:41 | 1 |

b. After A sends a frame to D

| Address | Port |
| --- | --- |
| 71:2B:13:45:61:41 | 1 |
| 64:2B:13:45:61:13 | 4 |

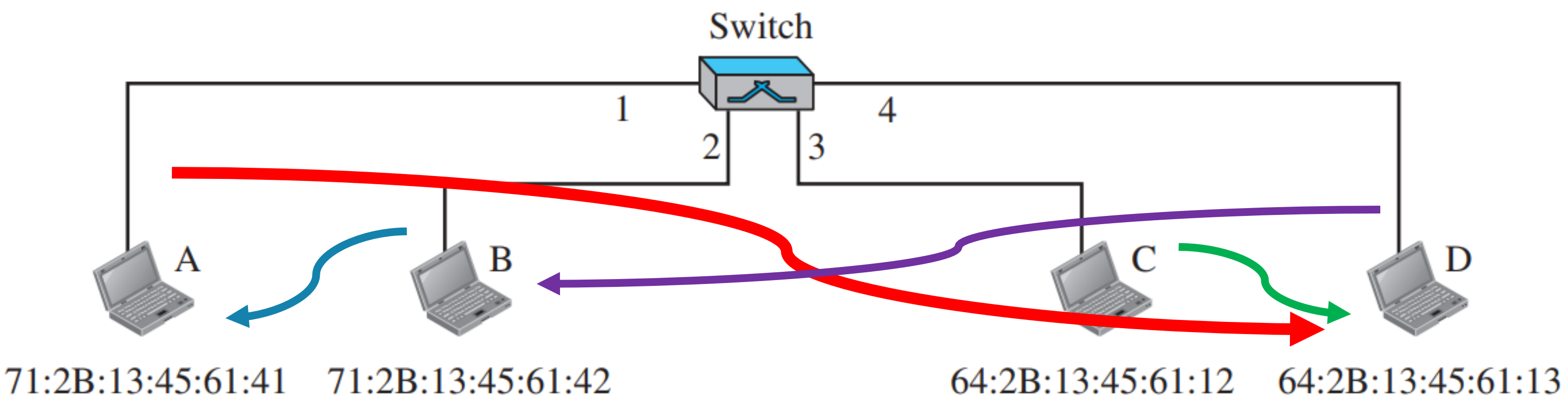
c. After D sends a frame to B

| Address | Port |
| --- | --- |
| 71:2B:13:45:61:41 | 1 |
| 64:2B:13:45:61:13 | 4 |
| 71:2B:13:45:61:42 | 2 |

d. After B sends a frame to A

| Address | Port |
| --- | --- |
| 71:2B:13:45:61:41 | 1 |
| 64:2B:13:45:61:13 | 4 |
| 71:2B:13:45:61:42 | 2 |
| 64:2B:13:45:61:12 | 3 |

e. After C sends a frame to D



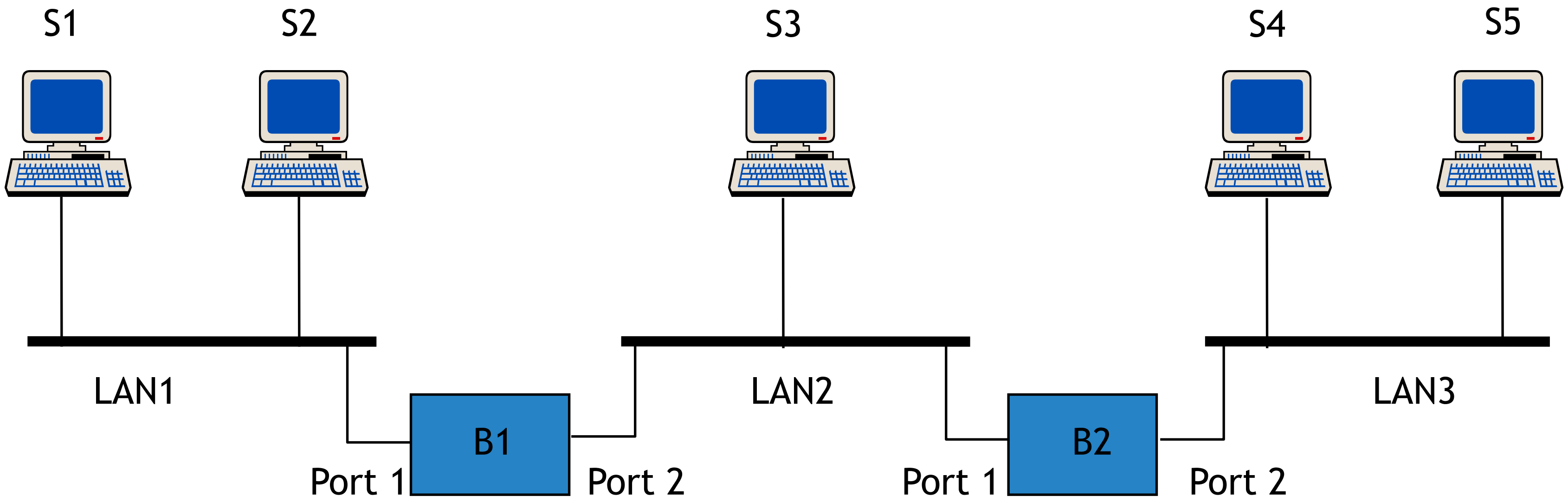
# Transparent Switches

- Learning Switches
- Step 1
  - When station A sends a frame to station D, the switch does not have an entry for either D or A.
  - The frame goes out from all three ports; the frame floods the network.
  - However, by looking at the source address, the **switch learns** that **station A** must be **connected to port 1**.
  - This means that frames destined for A, in the future, must be sent out through port 1.
  - The **switch adds this entry to its table**.
  - The table has its first entry now.
- Step 2
  - When station D sends a frame to station B, the switch has no entry for B, so it **floods the network again**.
  - However, it adds **one more entry** to the table **related to station D**.

# Transparent Switches

- **Step 3**
  - The learning process continues until the table has information about every port.
  - However, note that the **learning process may take a long time**.
  - For example, if a ***station does not send out a frame*** (a rare situation), the ***station will never have an entry in the table***.

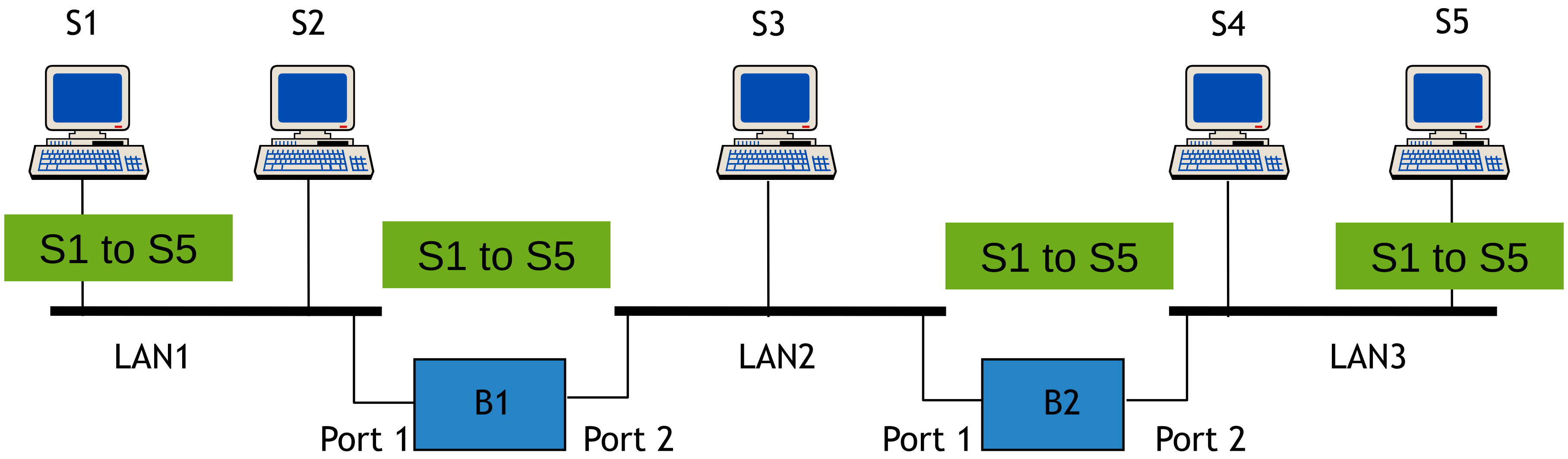
## *A learning bridge and the process of learning*




| Address | Port |
|---|---|
| | |
| | |
| | |
| | |
| | |

| Address | Port |
|---|---|
| | |
| | |
| | |
| | |
| | |

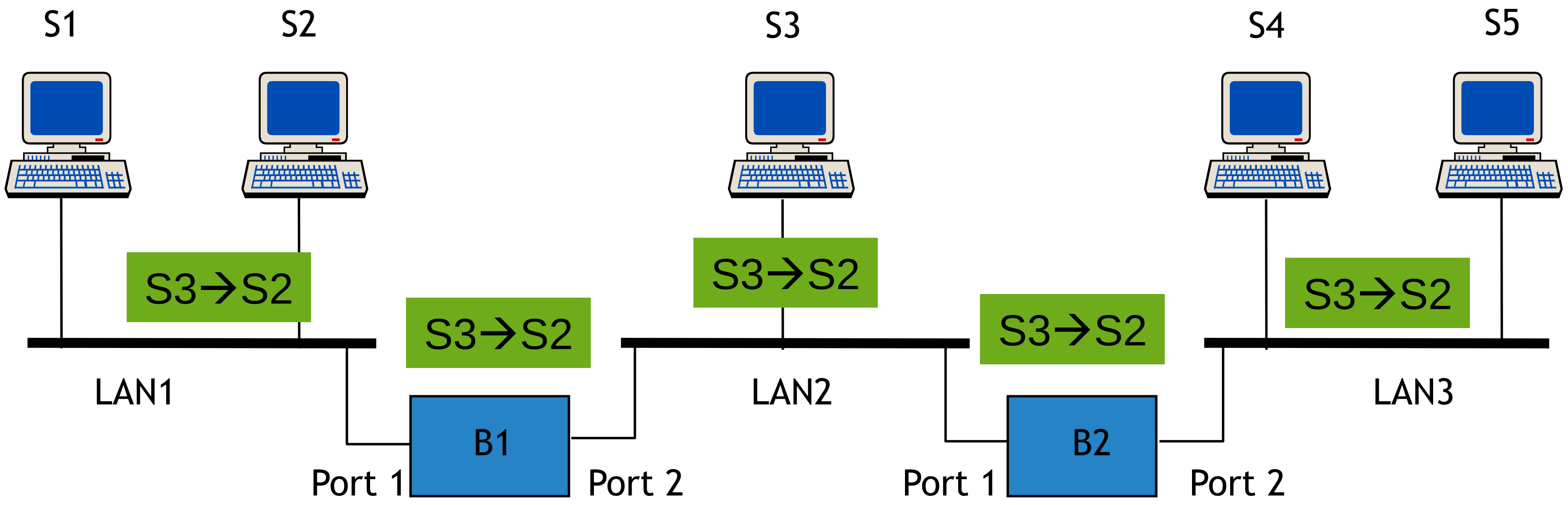
# S1→S5




| Address | Port |
|---|---|
| S1 | 1 |
| | |
| | |
| | |
| | |

| Address | Port |
|---|---|
| S1 | 1 |
| | |
| | |
| | |
| | |

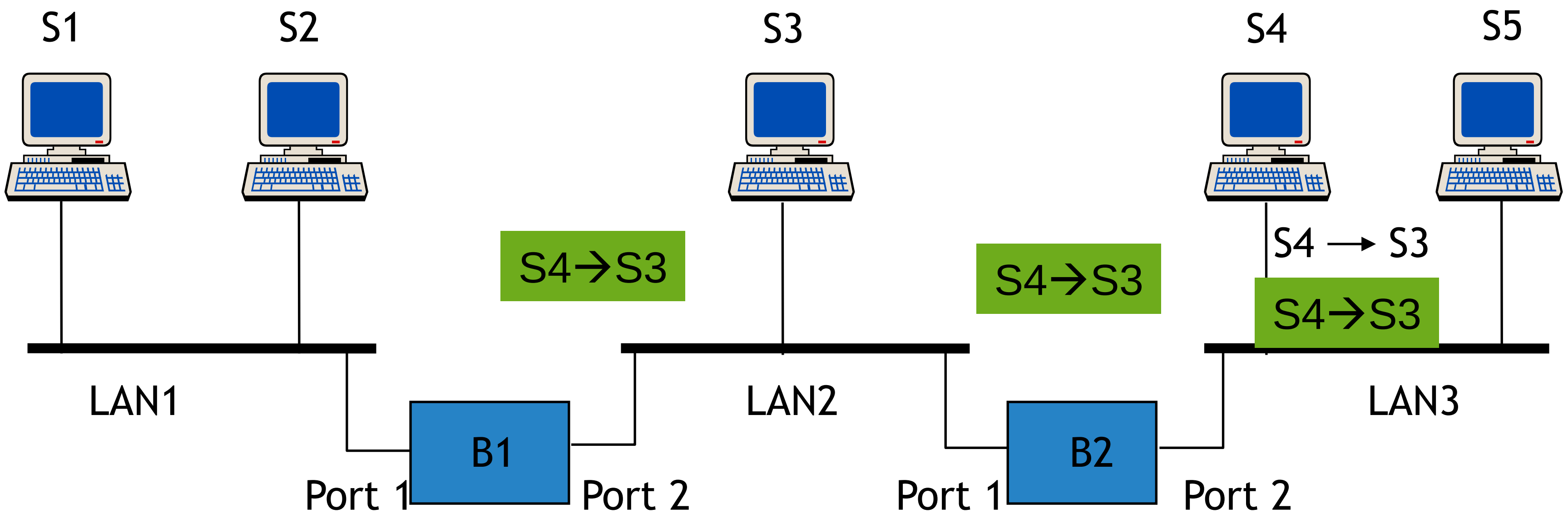
# S3→S2




| Address | Port |
|---|---|
| S1 | 1 |
| S3 | 2 |
| | |
| | |
| | |

| Address | Port |
|---|---|
| S1 | 1 |
| S3 | 1 |
| | |
| | |
| | |

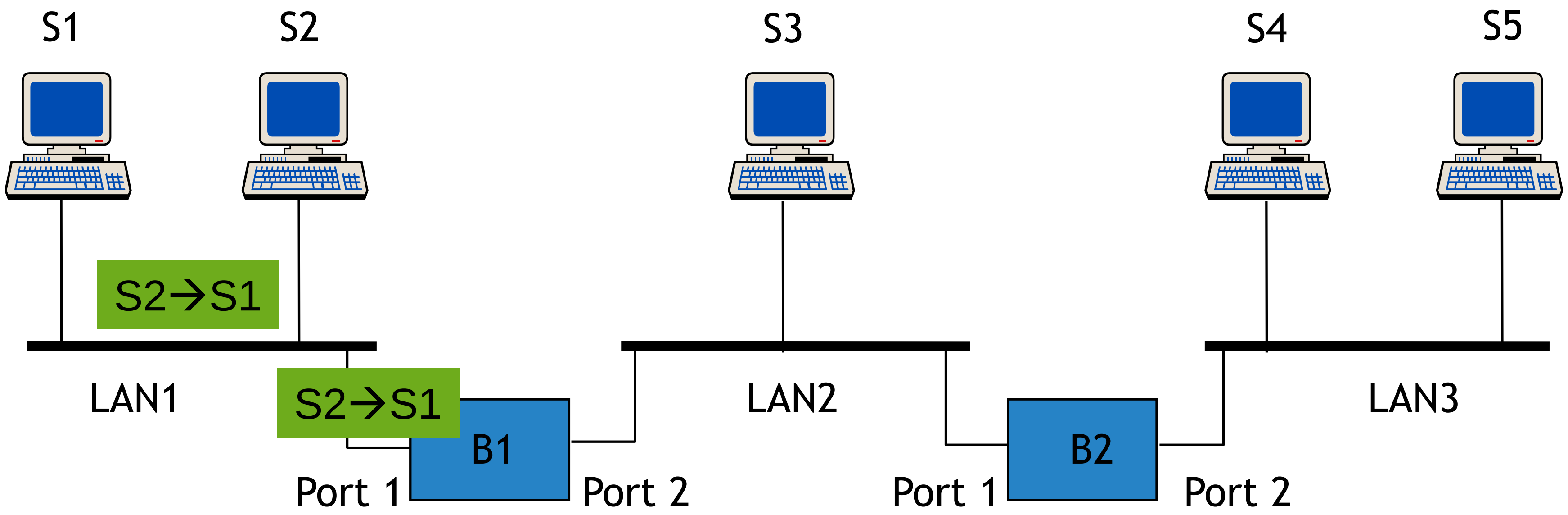
# S4→S3




| Address | Port |
|---|---|
| S1 | 1 |
| S3 | 2 |
| S4 | 2 |
| | |
| | |

| Address | Port |
|---|---|
| S1 | 1 |
| S3 | 1 |
| S4 | 2 |
| | |
| | |

# S2→S1




| Address | Port |
|---|---|
| S1 | 1 |
| S3 | 2 |
| S4 | 2 |
| S2 | 1 |
| | |

| Address | Port |
|---|---|
| S1 | 1 |
| S3 | 1 |
| S4 | 2 |
| | |
| | |

# Transparent Switches

## *Loop Problem in Learning Switches*

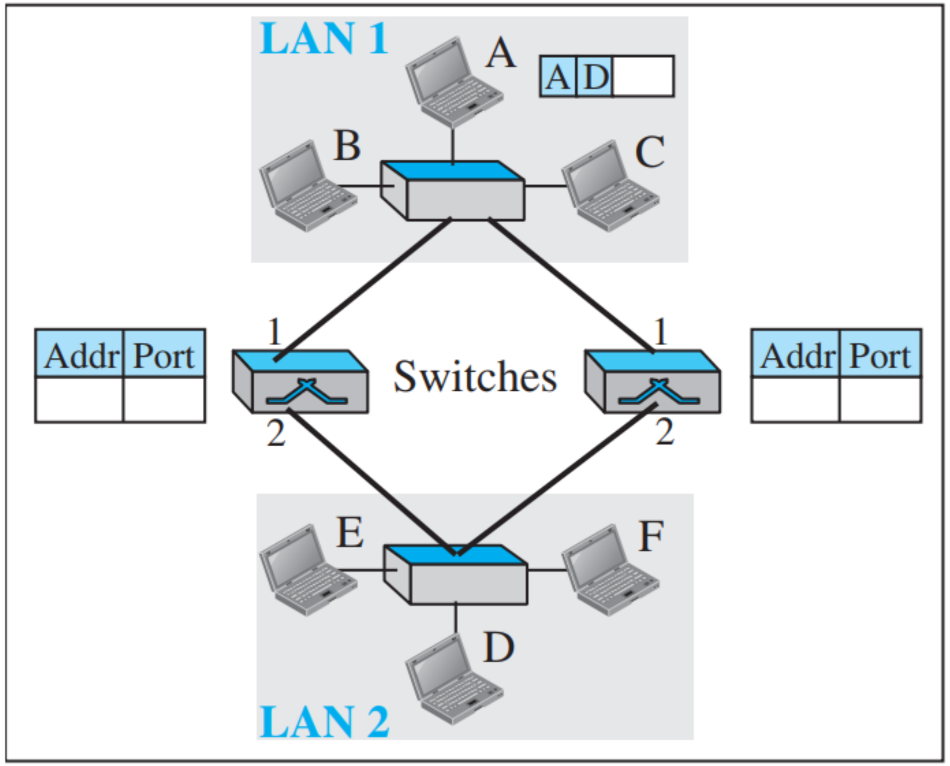
- Transparent switches work fine as long as there are no redundant switches in the system.
- Systems administrators, however, like to have redundant switches (more than one switch between a pair of LANs) to make the system more reliable.
- If a switch fails, **another switch takes over** until the failed one is repaired or replaced.
- Redundancy can **create loops** in the system, which is **very undesirable**.
- Loops can be created only when **two or more broadcasting LANs** (those using hubs, for example) are **connected by more than one switch.**

# Transparent Switches

## *Loop Problem in Learning Switches*

- **Station A** sends a frame to **station D.**
- The **tables** of **both switches** are empty.
- **Both forward the frame** and **update their tables** based on the **source address A.**

a. Station A sends a frame to station D

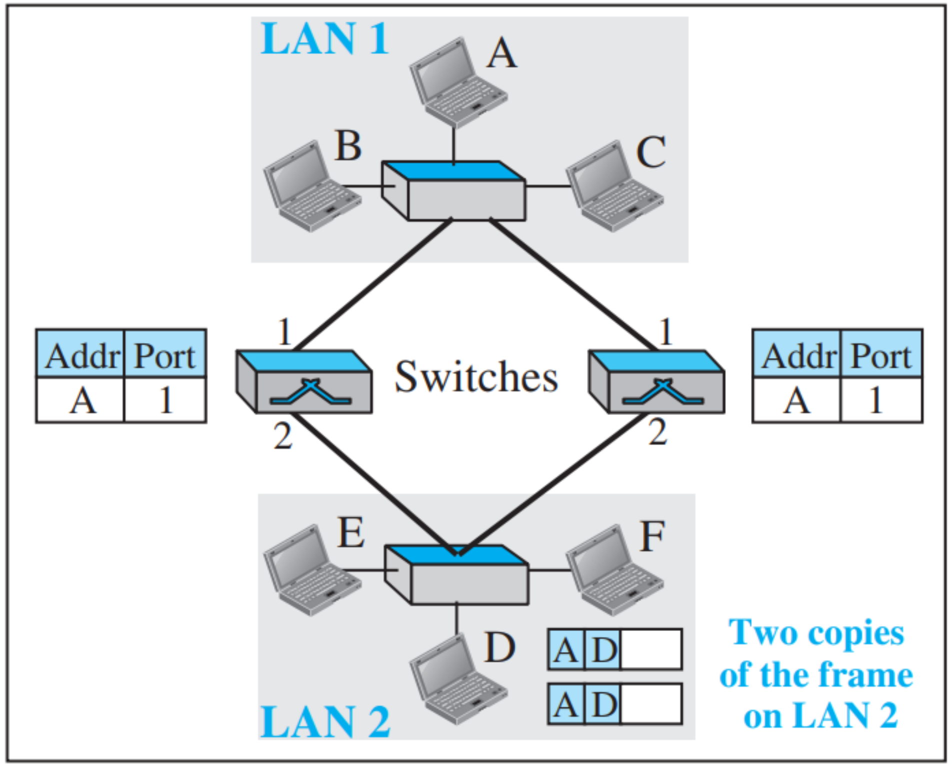


# Transparent Switches

## *Loop Problem in Learning Switches*

- Now **there are** two copies of the **frame on** LAN 2.
- The **copy sent out by the left switch** is **received by the right switch**, **which does not have** any **information** about the **destination address D**; it **forwards the frame**.
- The **copy sent out by the right switch is received by the left switch** and is **sent out for lack of information about D**.
- Note that **each frame** is **handled separately** because switches, as **two nodes** on a **broadcast network sharing the medium**.
- The tables of both switches are updated, but still there is no information for destination D.

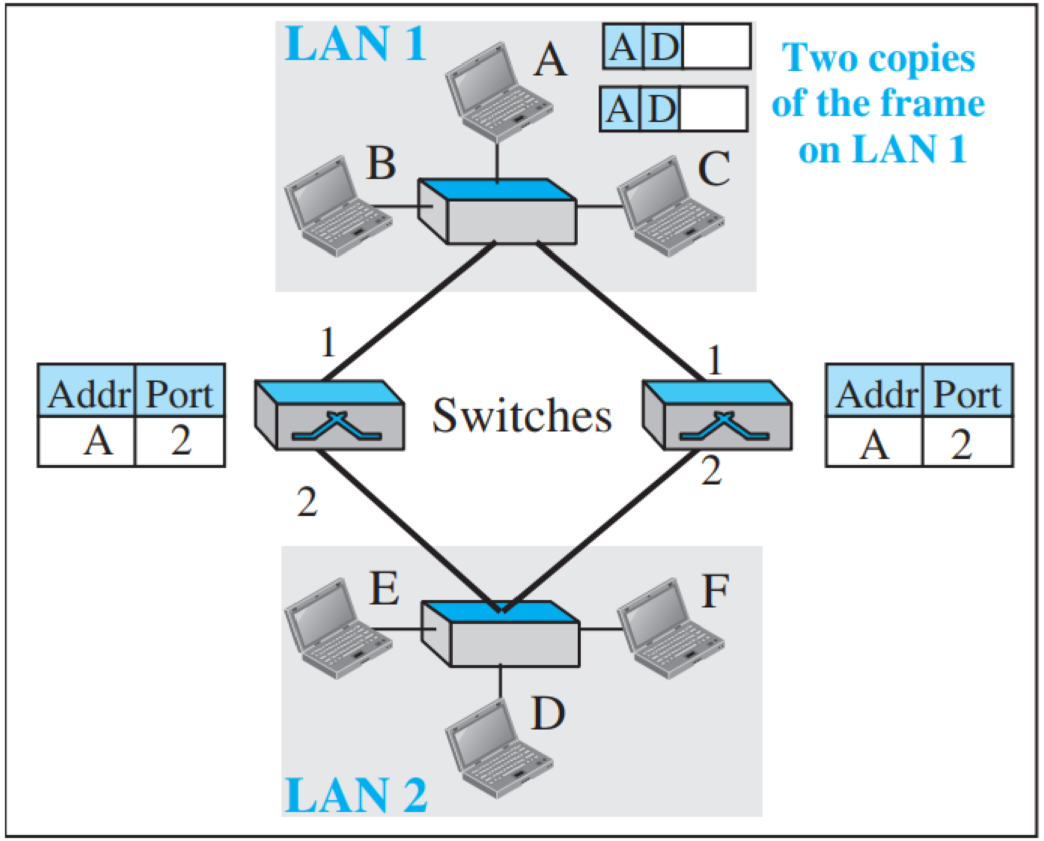
b. Both switches forward the frame



# Transparent Switches

## Loop Problem in Learning Switches

- Now there are **two copies** of the **frame** on **LAN 1**.
- Step 2 is repeated, and **both copies** are **sent** to **LAN2**.

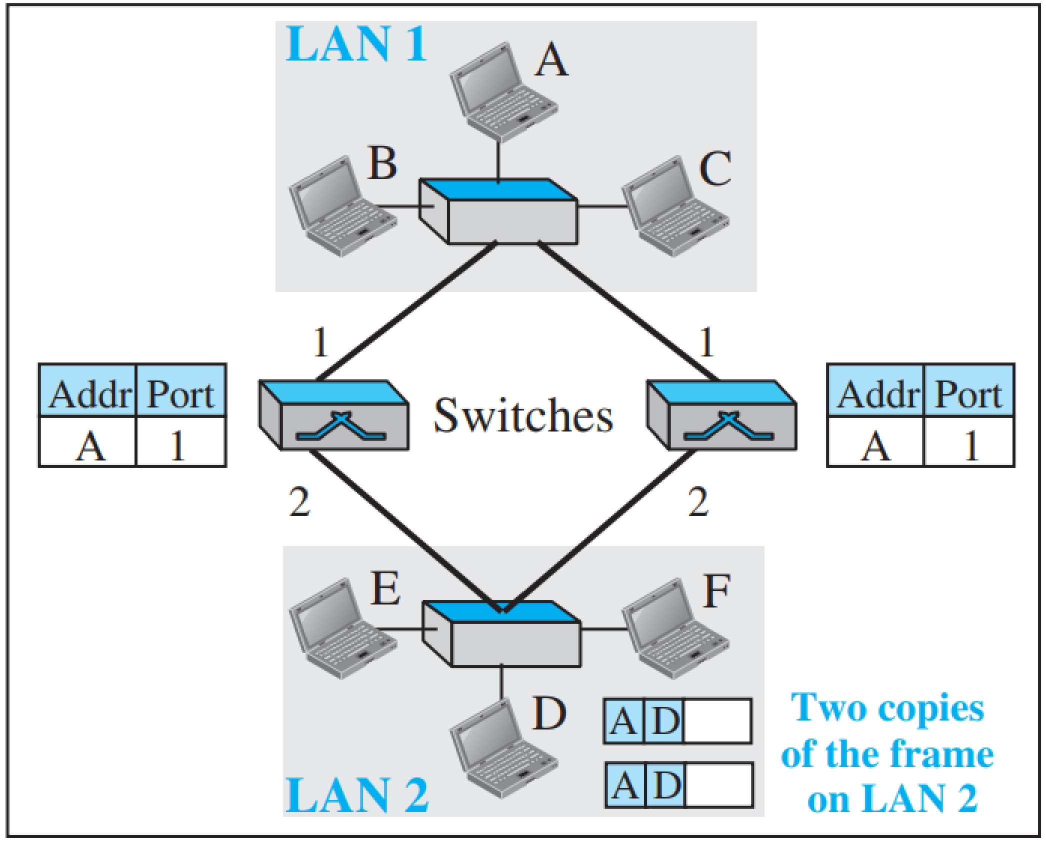



c. Both switches forward the frame

# Transparent Switches

## *Loop Problem in Learning Switches*

- The **process continues on and on**.
- Note that switches are also repeaters and regenerate frames.
- So in **each iteration**, **there are newly generated fresh copies of the frames**.




c. Both switches forward the frame

## Module 2

# Spanning Tree Algorithm

**BECE401L**

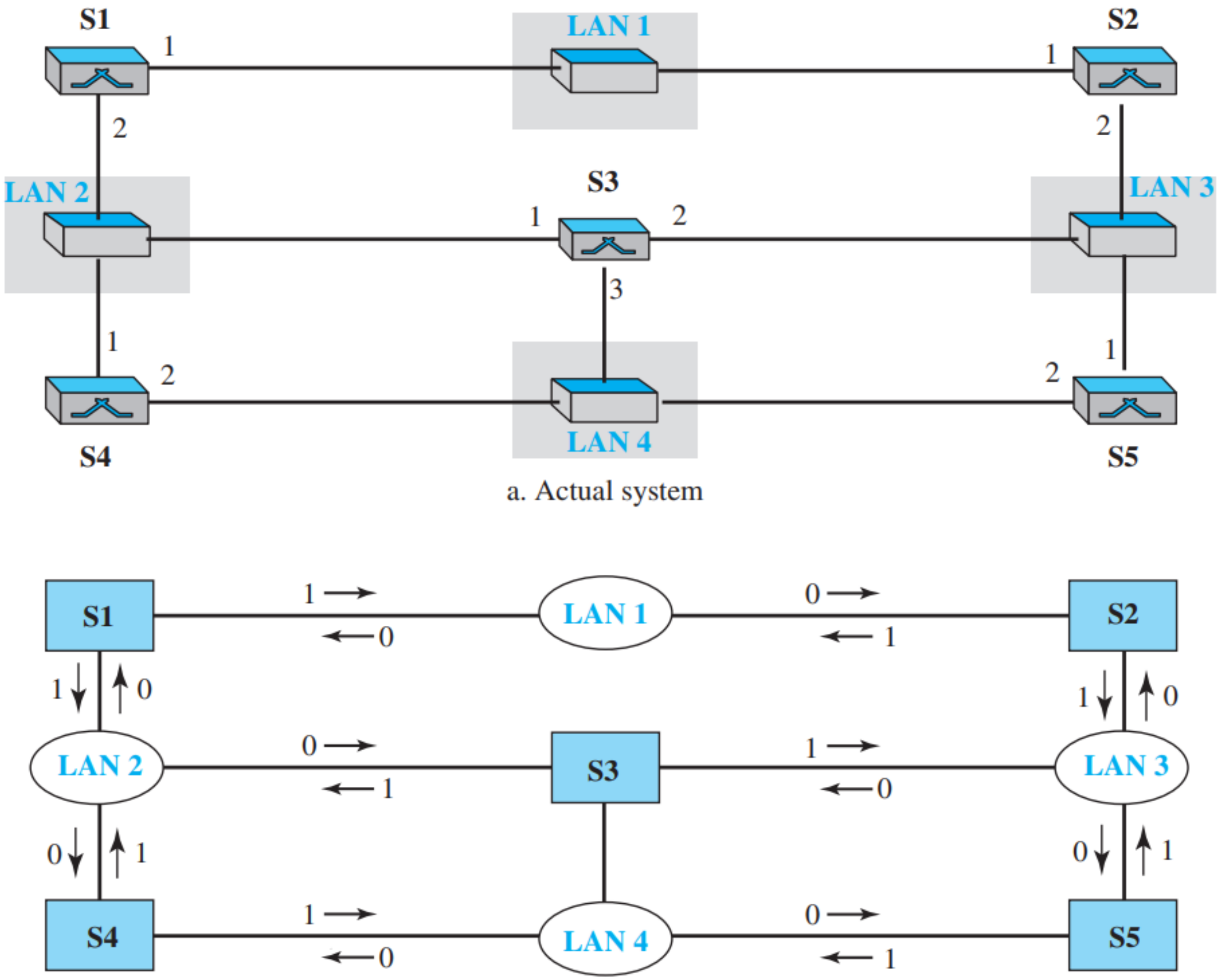
Dr. Nitish Katal

# Spanning Tree Algorithm

- To solve the looping problem, the **IEEE specification** requires that switches use the spanning tree algorithm to create a loopless topology.
- In graph theory, a spanning tree is a graph in which there is no loop.
- In a switched LAN, this means ***creating a topology*** in which ***each LAN*** can be ***reached*** from ***any other LAN through one path only*** (no loop).
- We cannot change the physical topology of the system because of physical connections between cables and switches, but we can create a logical topology that overlays the physical one.

# Spanning Tree Algorithm

- Figure shows a system with four LANs and five switches.
- The **connecting arcs** show the **connection of a LAN to a switch and vice versa.**
- To find the **spanning tree**, we need to **assign a cost (metric) to each arc**.
- The interpretation of the cost is left up to the systems administrator.
- We have chosen ***the minimum hops***.

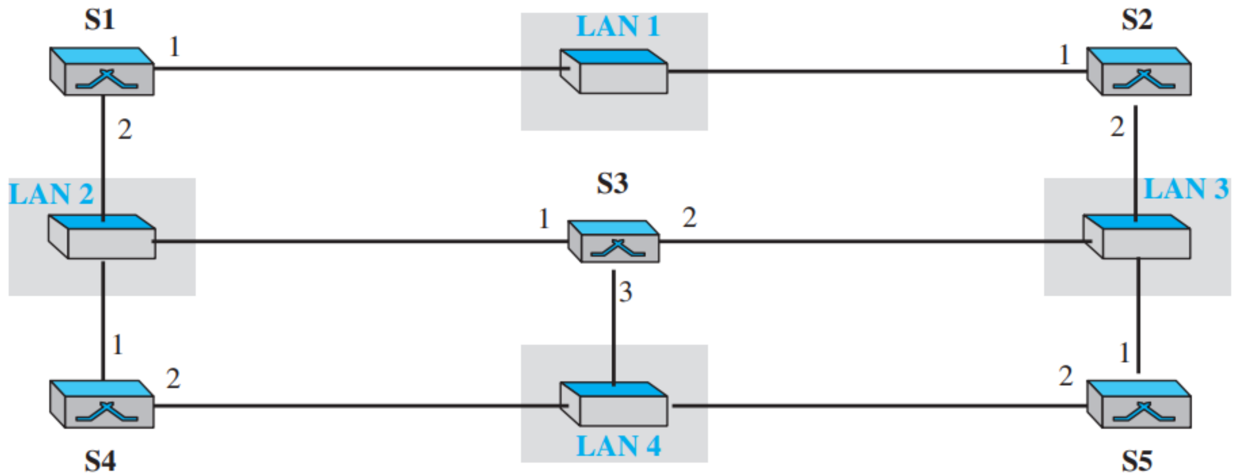


a. Actual system

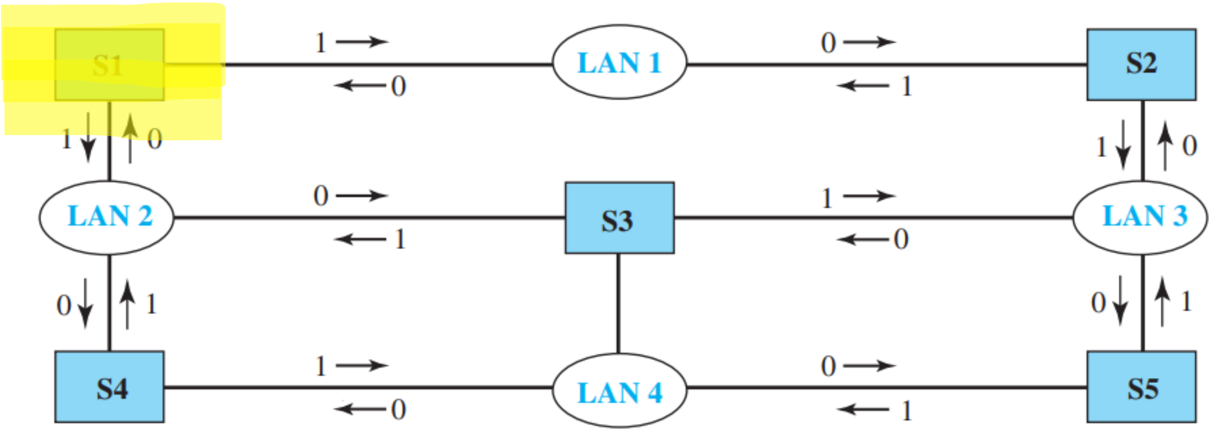
b. Graph representation with cost assigned to each arc

# Spanning Tree Algorithm

- Executed in 3 Steps:
- Step 1:
- Every switch has a *built-in ID* (normally the *serial number*, which is unique).
- **Each switch** broadcasts this **ID** so that **all switches** know which one has the smallest ID.
- The switch with the **smallest ID** is selected as the root switch (root of the tree).
- Assume that *switch S1* has the *smallest ID*.
  - *It is, therefore, selected as the root switch.*



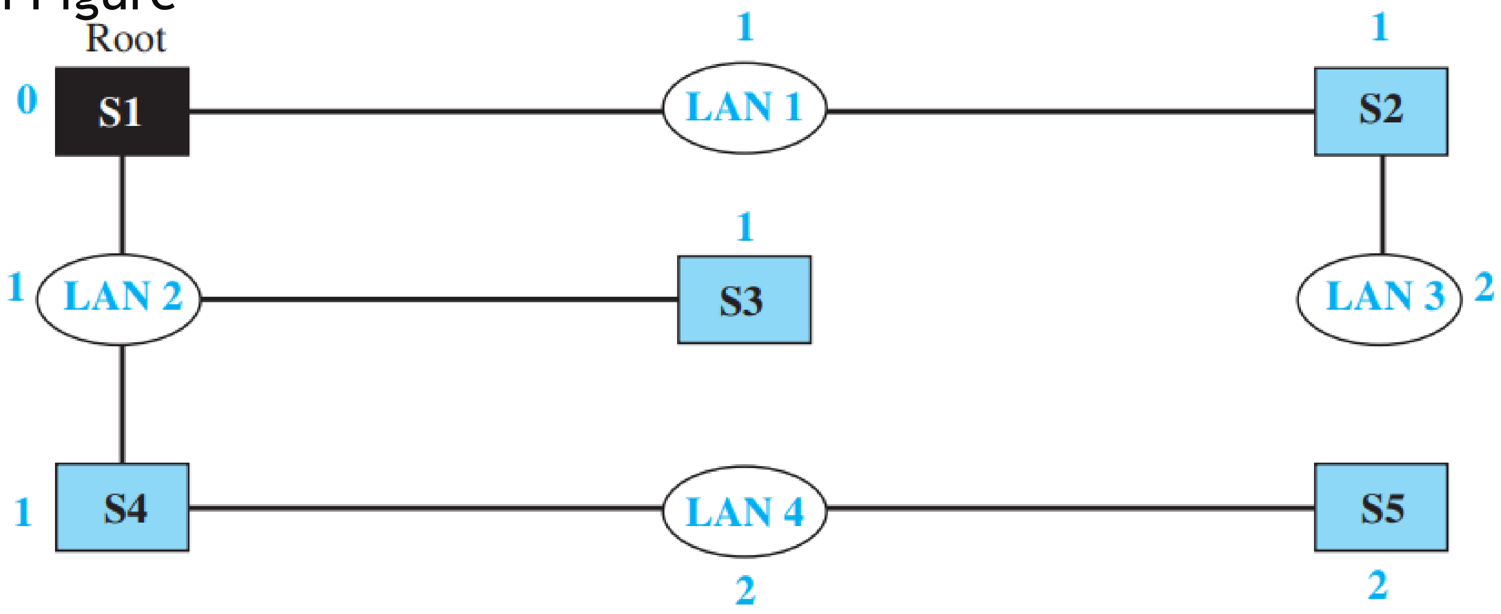
a. Actual system



b. Graph representation with cost assigned to each arc

# Spanning Tree Algorithm

- **Step 2:**
- The *algorithm tries* to find the *shortest path* (a path with *the shortest cost*) from the *root switch* to *every other switch* or *LAN*.
- The ***shortest path*** can be found by ***examining the total cost*** from the ***root switch to the destination.***
- The **combination of the shortest paths** creates the ***shortest tree***, which is also shown in Figure



# Spanning Tree Algorithm

- **Step 3:**
- Based on the spanning tree, we **mark the ports that are part of it**, the **forwarding ports**, which forward a frame that the switch receives.
- We also ***mark those ports that are not part of the spanning*** tree, the **blocking ports**, which block the frames received by the switch.

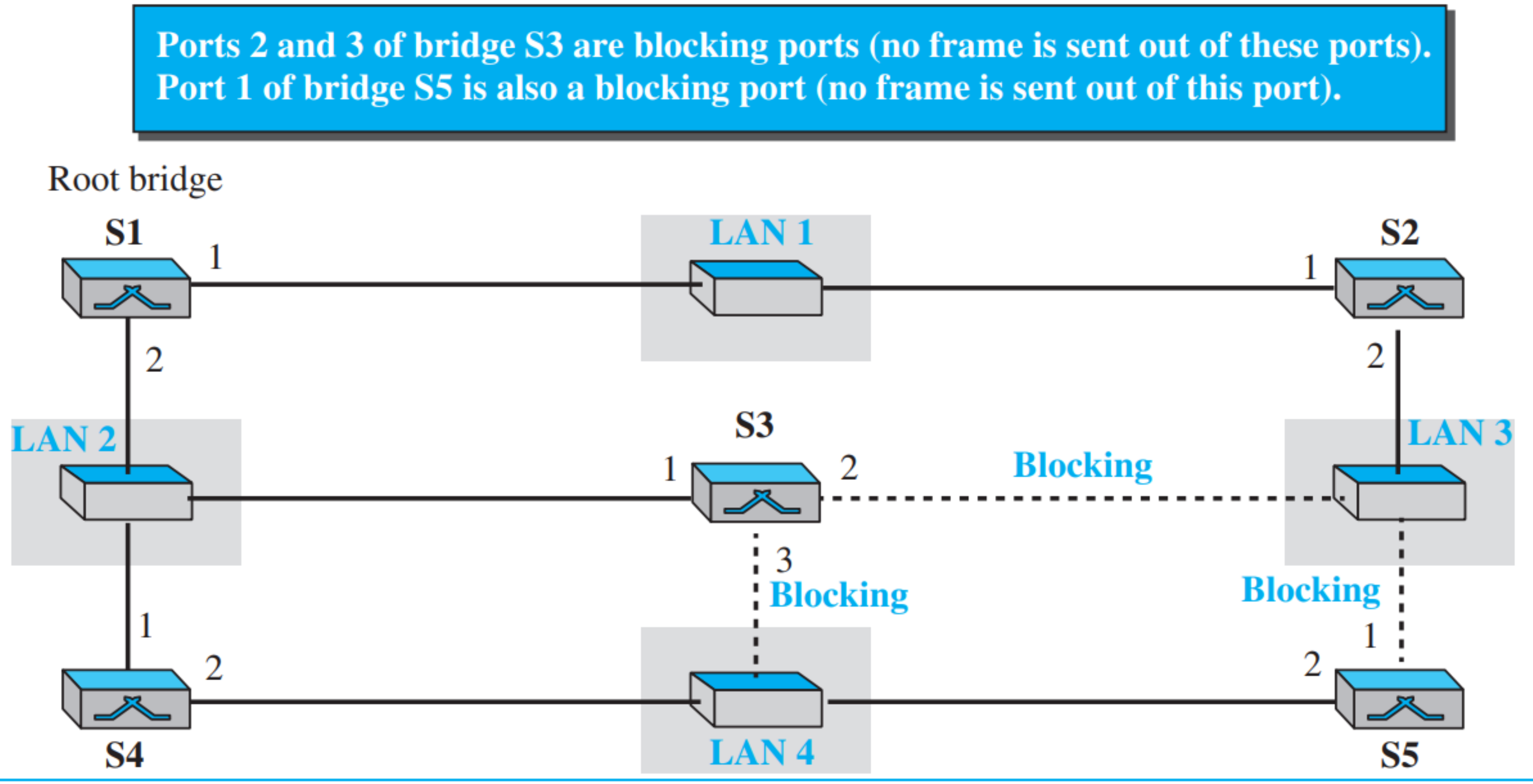
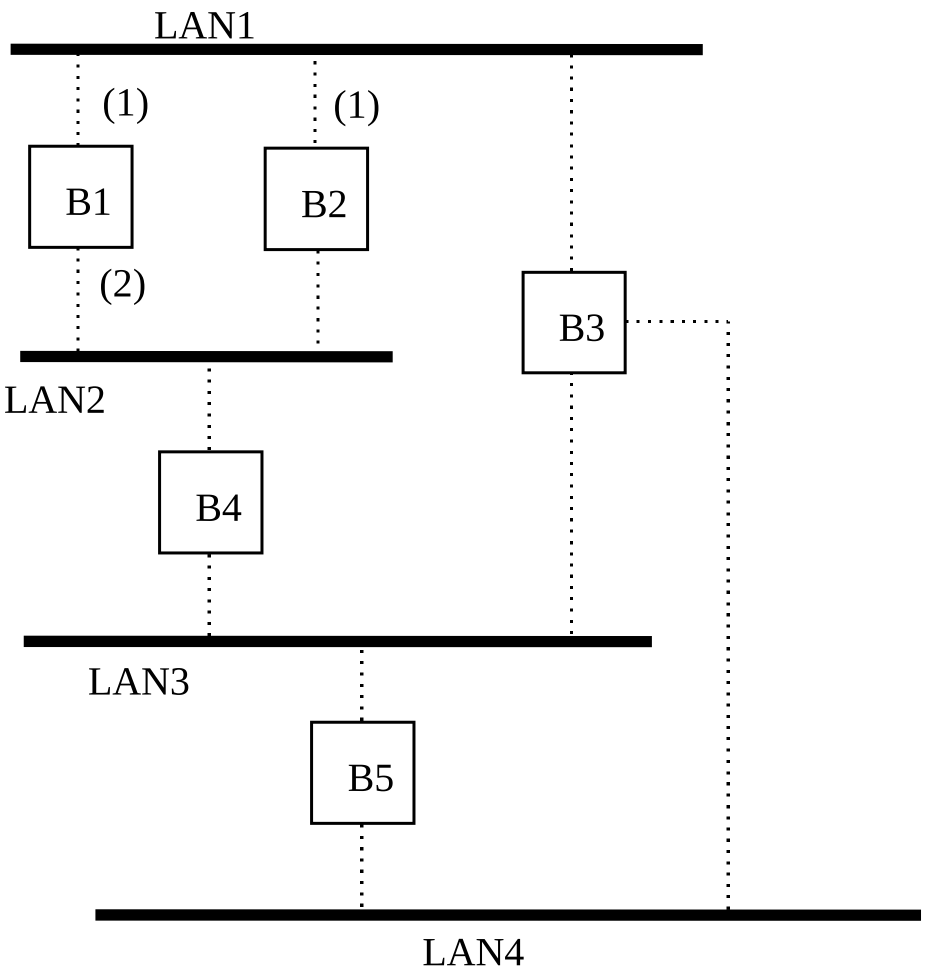
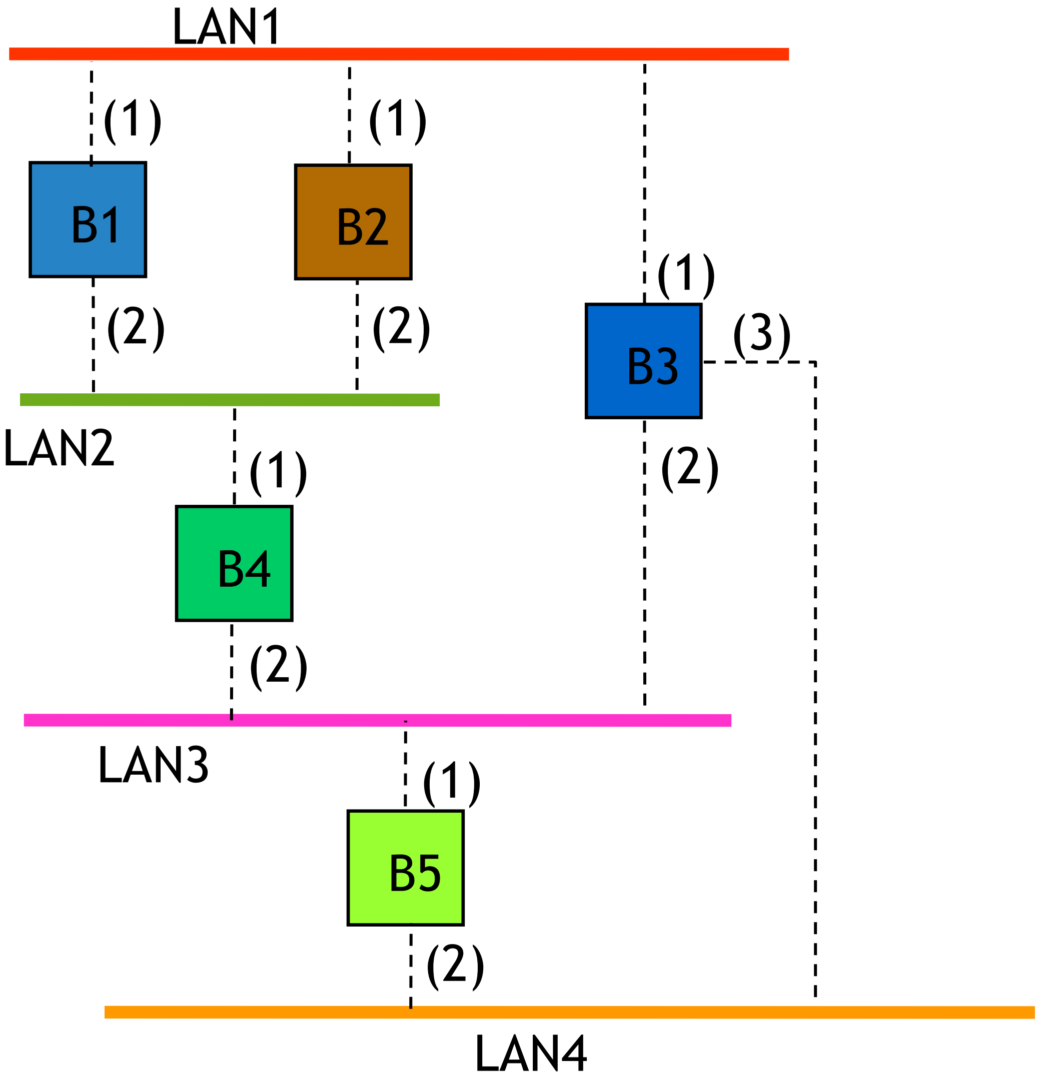


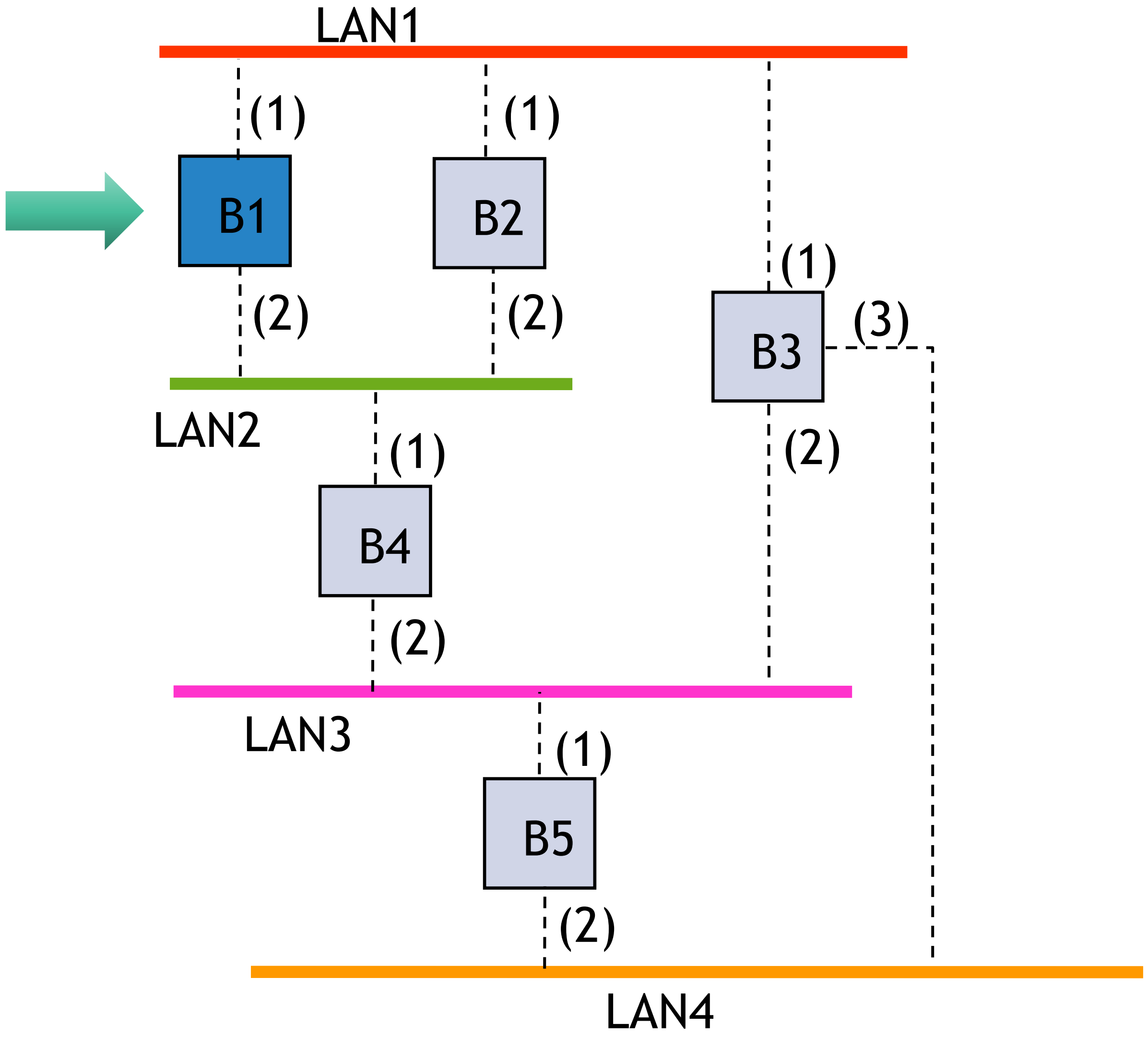

Figure shows the logical systems of LANs with forwarding ports (solid lines) and blocking ports (broken lines)

# Example 2 : Spanning Tree Algorithm



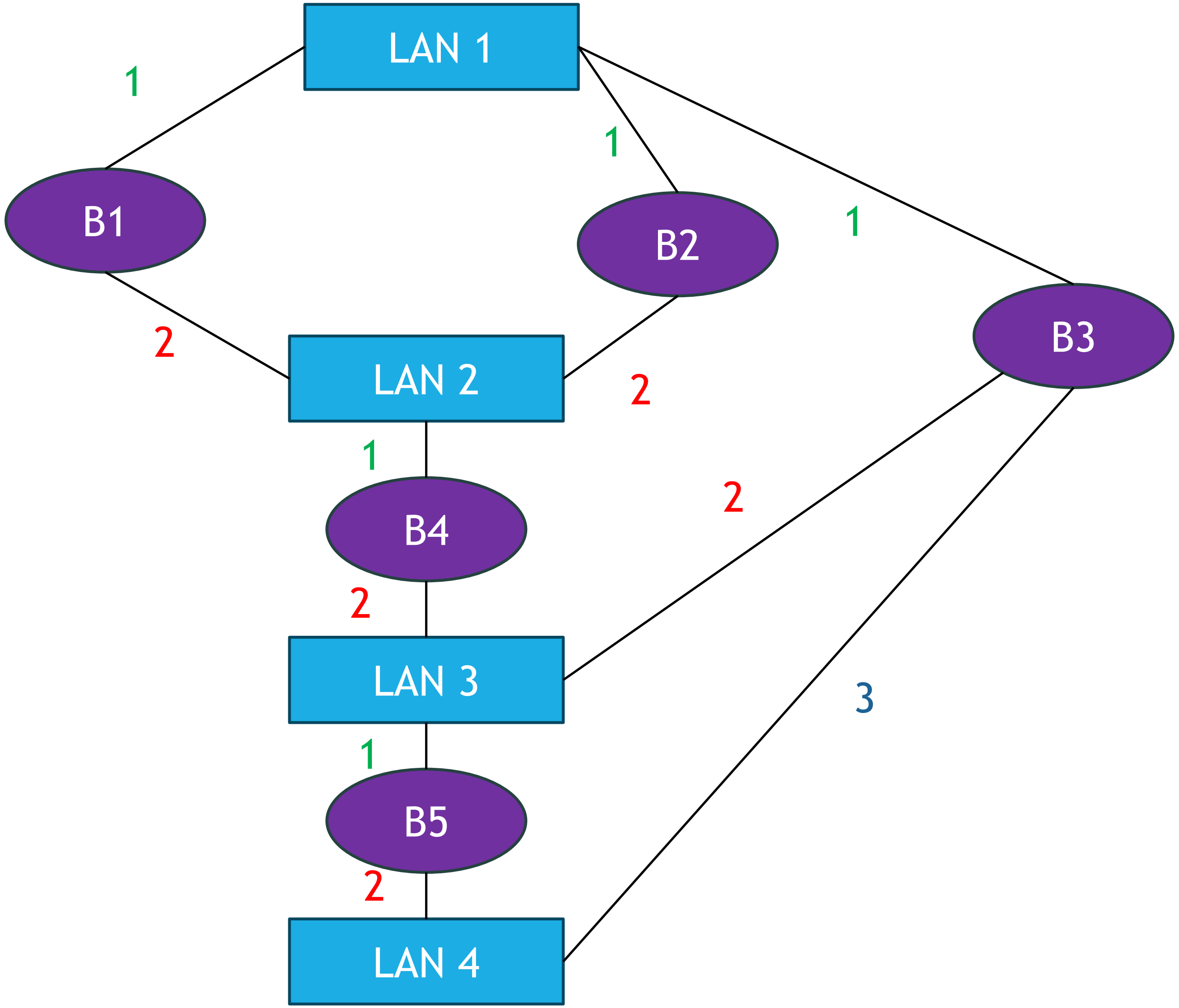

LAN1
(1)
(1)
B1
B2
(1)
(2)
(2)
(3)
B3
LAN2
(1)
(2)
B4
(2)
LAN3
(1)
B5
(2)
LAN4

1. Select a *root bridge* among all the bridges.
   - root bridge = the ***lowest bridge ID***.



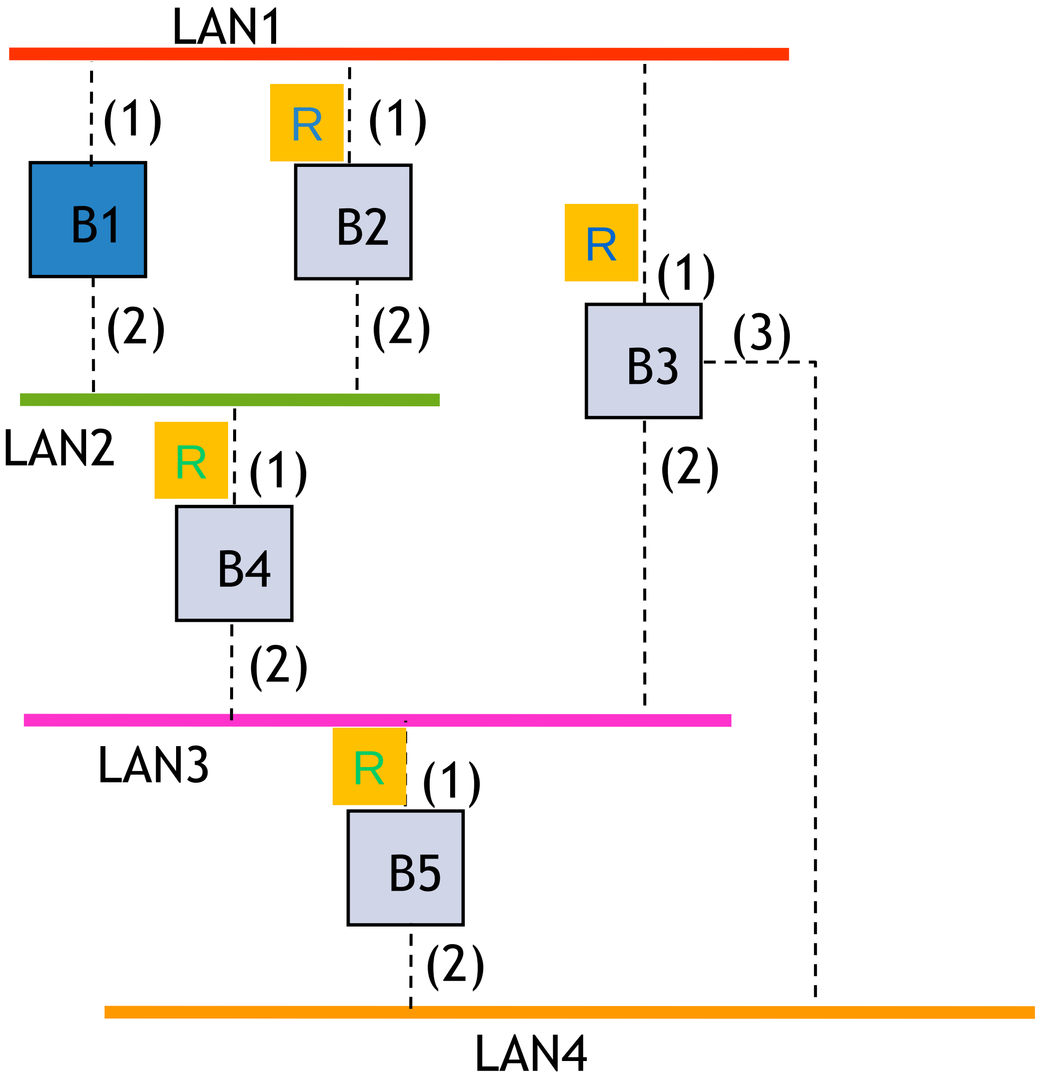

Bridge 1 selected as root bridge

## Distance Calculation:



## 2.Determine the *root port* for each *bridge except the root bridge*

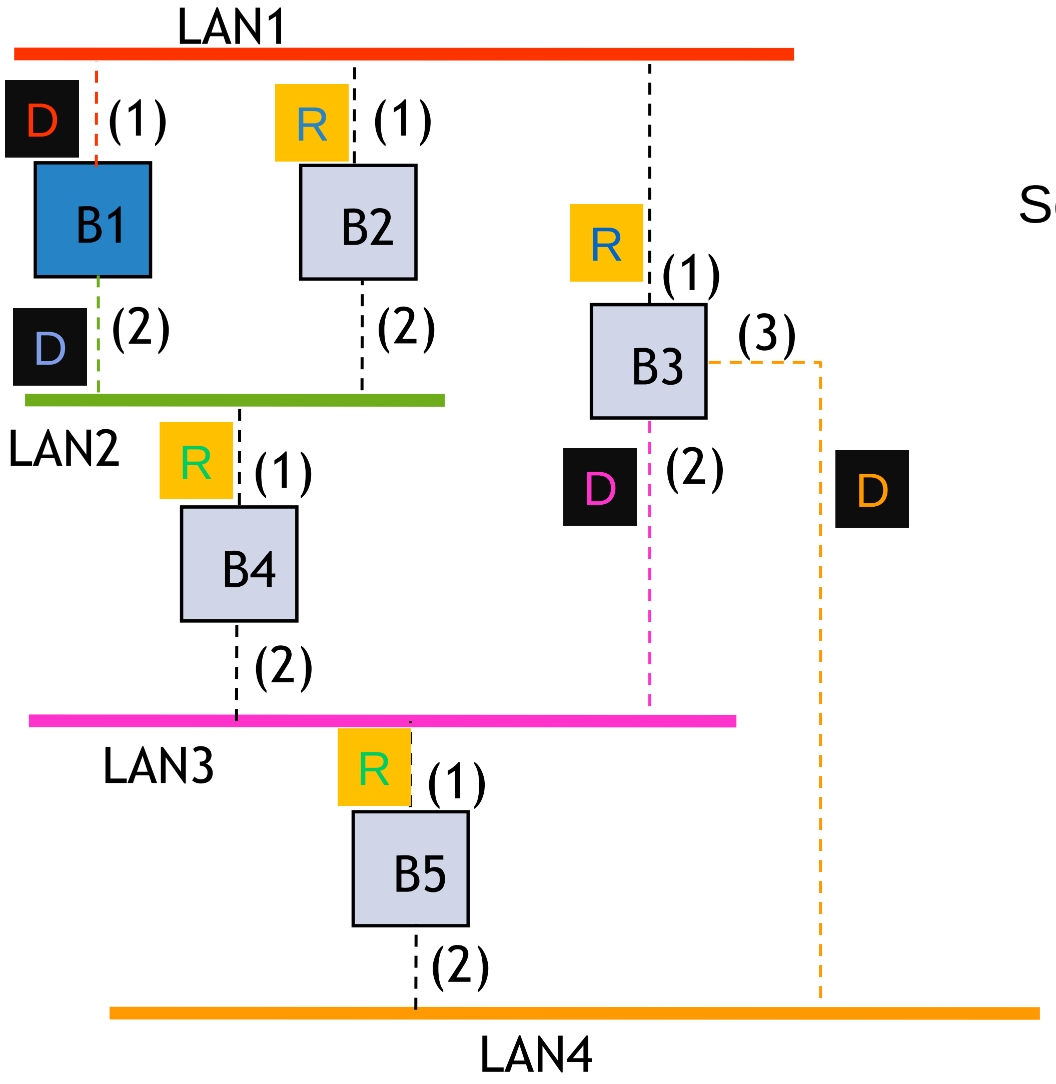
- root port = port with the least-cost path to the root bridge



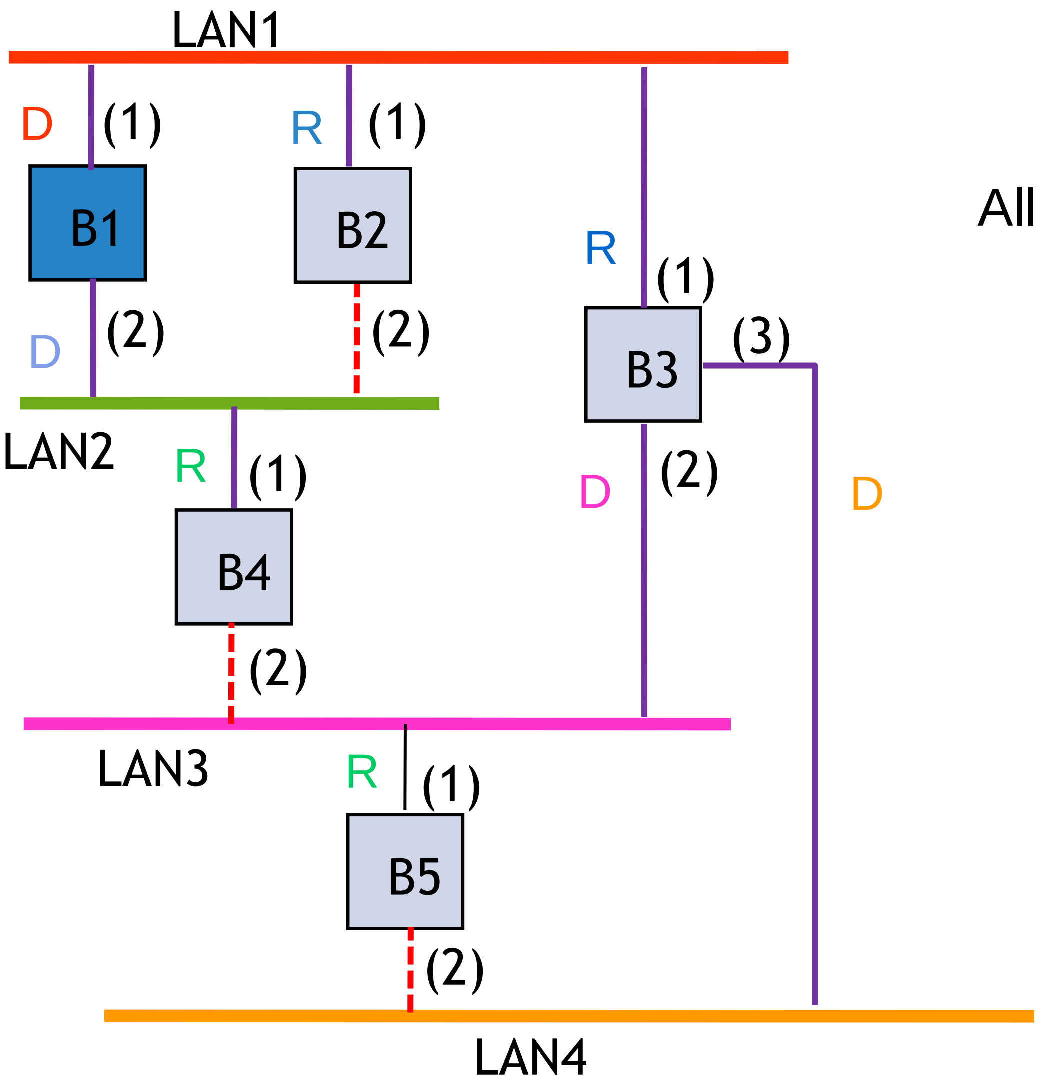

Root port selected for every bridge except root bridge

3.Select a *designated bridge* for each LAN

- designated bridge = bridge has least-cost path from the LAN to the root bridge.
- *designated port* connects the LAN and the designated bridge




Select designated bridge for each LAN




All root ports & designated ports put in forwarding state

# Advantages & Disadvantages of Bridges

- *Advantages of using a bridge*
  - Extend physical network
  - Reduce network traffic with minor segmentation
  - Creates separate collision domains
  - Reduce collisions
  - Connect different architecture
- *Disadvantages of using bridges*
  - Slower that repeaters due to filtering
  - Do not filter broadcasts
  - More expensive than repeaters

# Gateway

- Connect ***two networks above the network layer*** of OSI model.
- Are ***capable of converting data frames*** and ***network protocols into the format needed by another network***.
- Provide for ***translation services*** between ***different computer protocols***.
- **Transport gateways** make a connection between two networks at the **transport layer**.
- **Application gateways** connect two parts of an application in the **application layer**, e.g., sending email between two machines using different mail formats
- Example: Broadband-modem-router