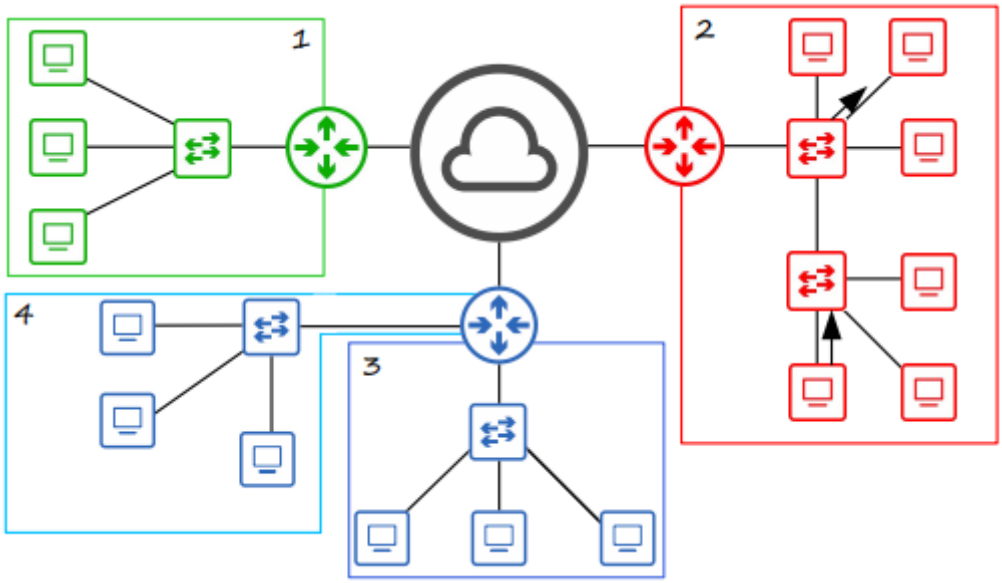
Ethernet LAN Switching (part 1)

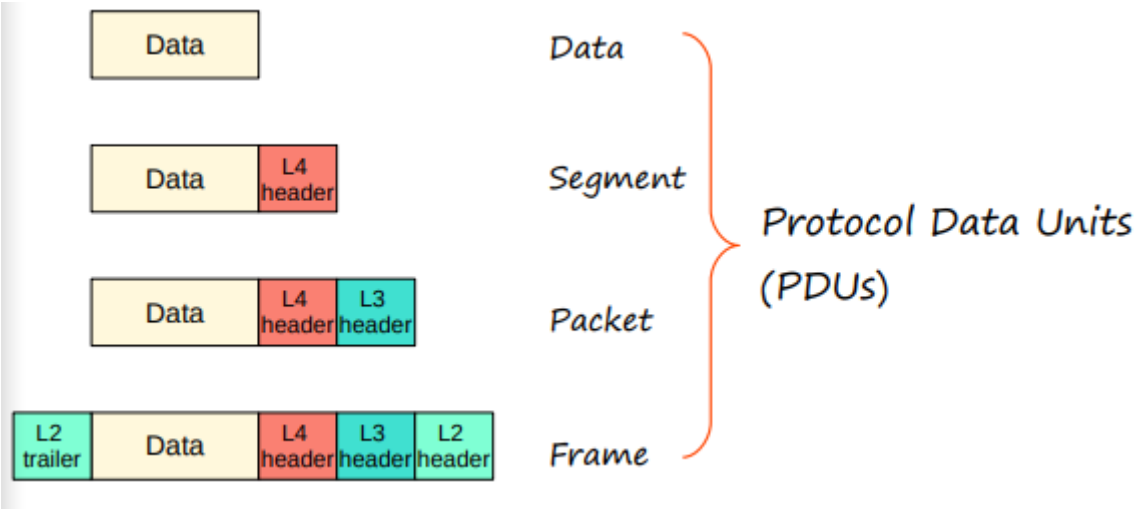
LAN Switching

In this session we will cover how data travels within a network. We will look closely at how data moves between switches, routers and end hosts.



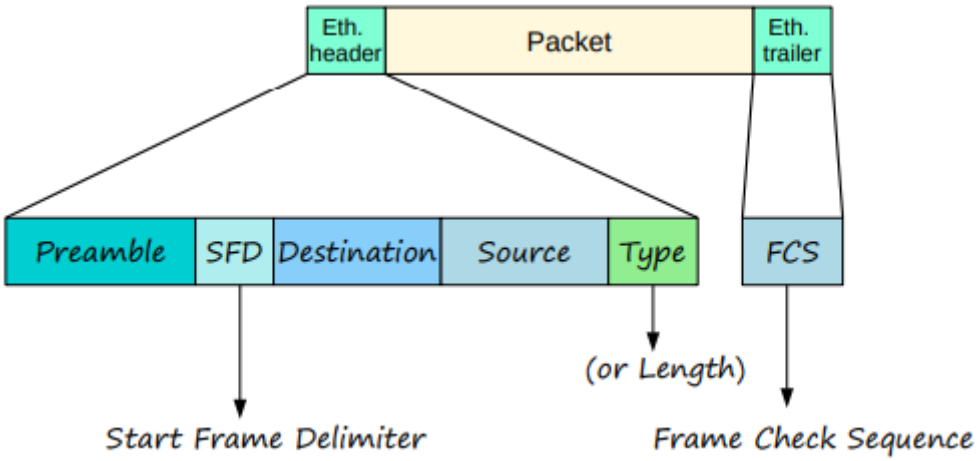
There are 4 LAN's in this network

We can see from this image that routers are use to connect to different LAN's. We will cover how traffic is sent and received within LAN's, and how the data is sent & received through the Layer 2 of the OSI Model. The data being sent/received is called a **frame**.



- `data` : **Layers 7 - 5**
- `segment` : **Layer 4**
- `packet` : **Layer 3**
- `frame` : **Layer 2**

We will look at what consists of an **Ethernet frame**. Ethernet frames are used in every LAN today!



The **frame header** is broken down into 5 fields - **Preamble, SFD, Destination, Source & Type**. The **frame trailer** has 1 field - **FCS**. We will go over the functionality of each field:

Preamble

- Length: 7 bytes (56 bits)
- Alternating 1's and 0's
- 10101010 * 7
- Allows devices to synchronize their receiver clocks

SFD

- 'Start Frame Delimiter'
- Length: 1 byte (8 bits)
- 10101011
- Marks the end of the preamble, and the beginning of the rest of the frame

Preamble & Start Frame Delimiter

Destination **Source**

- Indicate the devices sending and receiving the frame
- Consist of the destination and source 'MAC address'
- MAC = Media Access Control
- = 6 byte (48-bit) address of the physical device

Destination & Source

Type

OR

Length

- 2 byte (16-bit) field
- A value of **1500 or less** in this field indicates the LENGTH of the encapsulated packet (in bytes)
- A value of **1536 or greater** in this field indicates the TYPE of the encapsulated packet (usually IPv4 or IPv6), and the length is determined via other methods

IPv4 = 0x0800 (hexadecimal) (2048 in decimal)

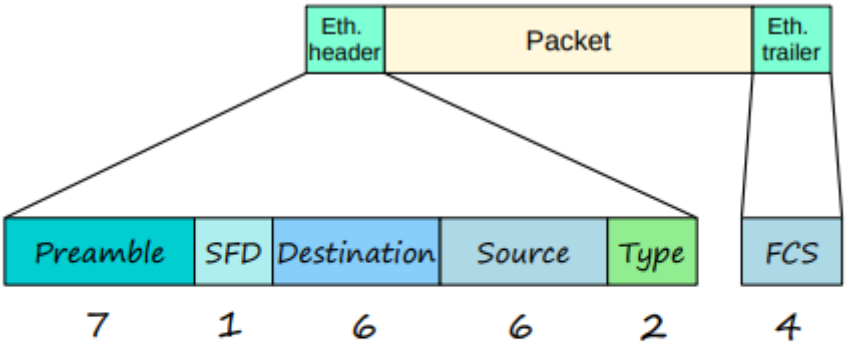
IPv6 = 0x86DD (hexadecimal) (34525 in decimal)

Type / Length

- 'Frame Check Sequence'
- 4 bytes (32 bits) in length
- Detects corrupted data by running a 'CRC' algorithm over the received data
- CRC = 'Cyclic Redundancy Check'

Frame Check Sequence (FCS)

You should memorise the total amount of bytes in a frame:



= 26 bytes (header + trailer)

MAC Address

A quick overview on what a MAC address is:

- It's the **physical address** assigned to the machine when it is made

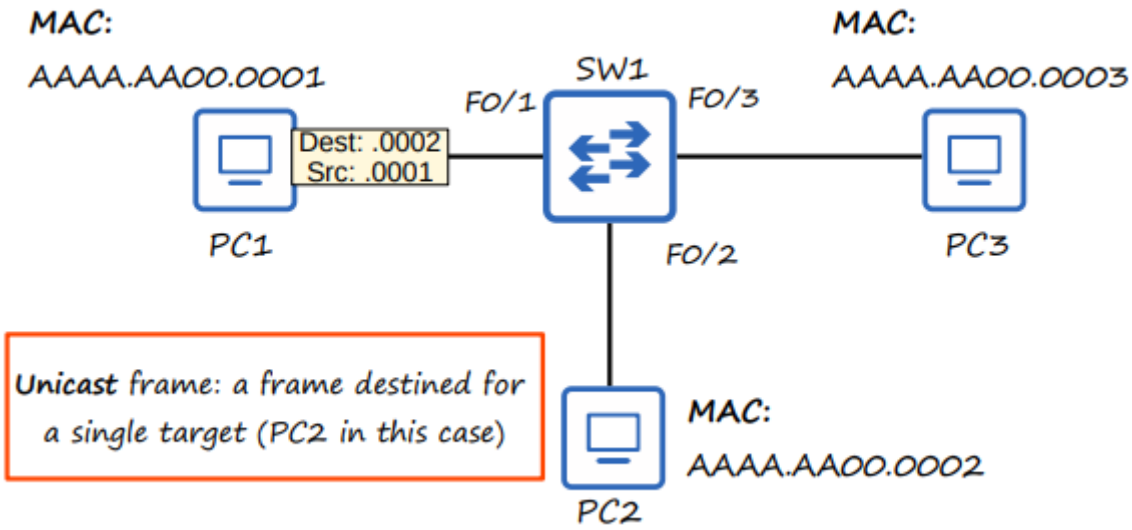
- The address is **6 bytes (48 bits)** long
- It is also known as **Burned In Address (BIA)**
- The **first three bytes** of the address is the **OUI (Organisationally Unique Identifier)**, which is the company/manufacturer address
- The **last three bytes** of the address is unique to the device
- Written as **12 hexadecimal characters**.

DEC.	HEX.	DEC.	HEX.	DEC.	HEX.	DEC.	HEX.
0	0	8	8	16	10	24	18
1	1	9	9	17	11	25	19
2	2	10	A	18	12	26	1A
3	3	11	B	19	13	27	1B
4	4	12	C	20	14	28	1C
5	5	13	D	21	15	29	1D
6	6	14	E	22	16	30	1E
7	7	15	F	23	17	31	1F

Study the table to grasp how hexadecimals work

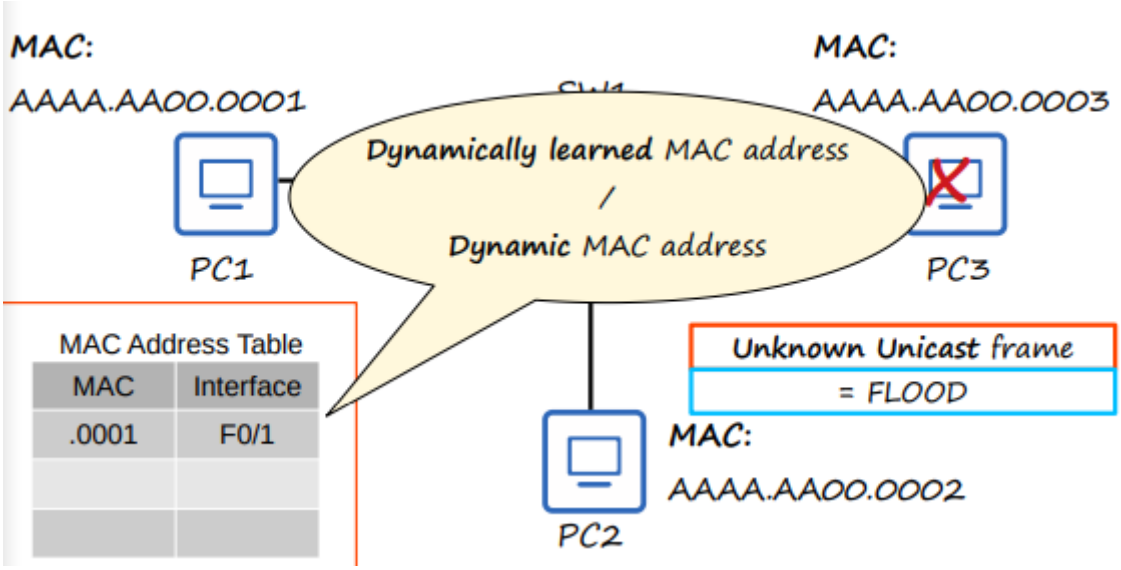
Sending Data Within Switches

Now lets look at how switches use the MAC address to send and receive data.



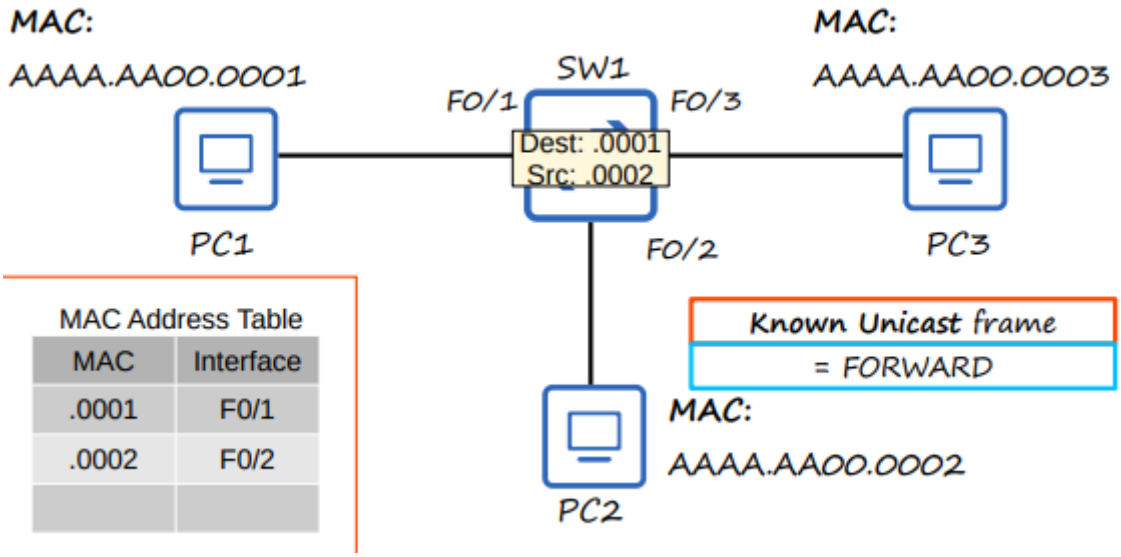
PC1 wants to send data to PC2

`AAA.AA` is the manufacturer's address **(OUI)** and `00.0001` is PC1's unique identifier. PC1 is sending data to PC2 - this is called a **unicast frame** because the frame (data) is being sent to only 1 recipient.



When the frame is sent from PC1, first it goes to the switch. The switch takes the **MAC address and interface** of the incoming frame, then **floods** the rest of the nodes with the data until the correct one receives it. This is known as an **unknown cast frame**.

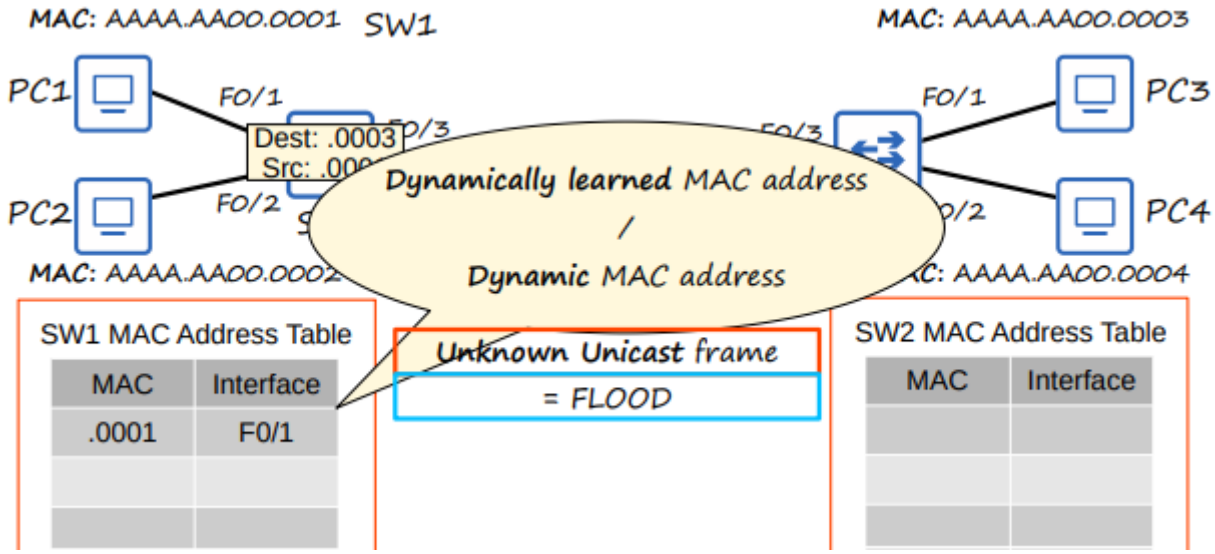
The switch will send an unknown cast frame to both PC3 and PC2. PC3 rejects the frame because the destination address does not match. PC2 matches the destination within the frame, so the frame is accepted.



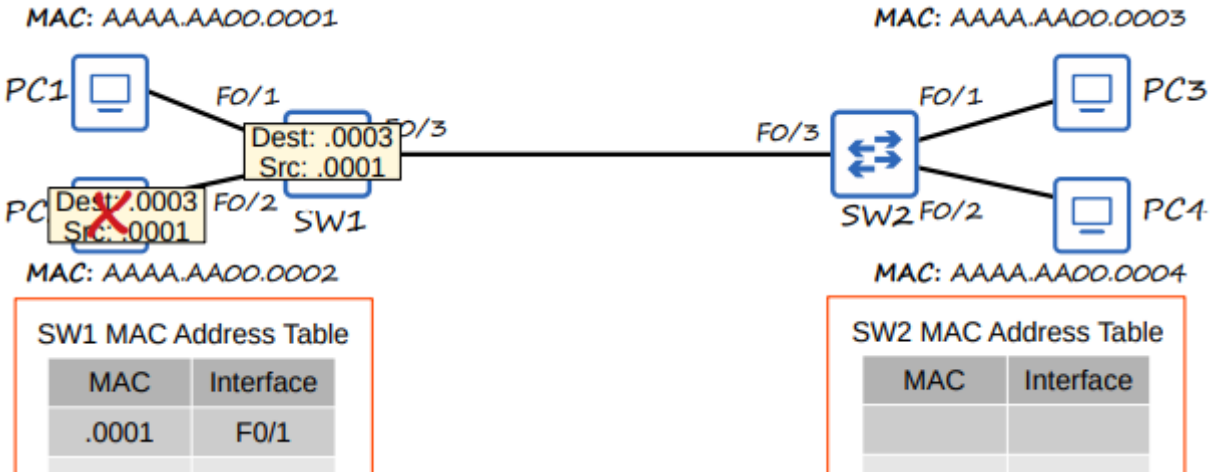
When PC2 sends data back to PC1, the switch stores the **MAC address and interface** of the incoming frame. It checks if the destination of the frame is in the **MAC address table**; the address is found and the switch **forwards** the frame directly to PC1. This is called a **known unicast frame**.

Sending Data Between Switches

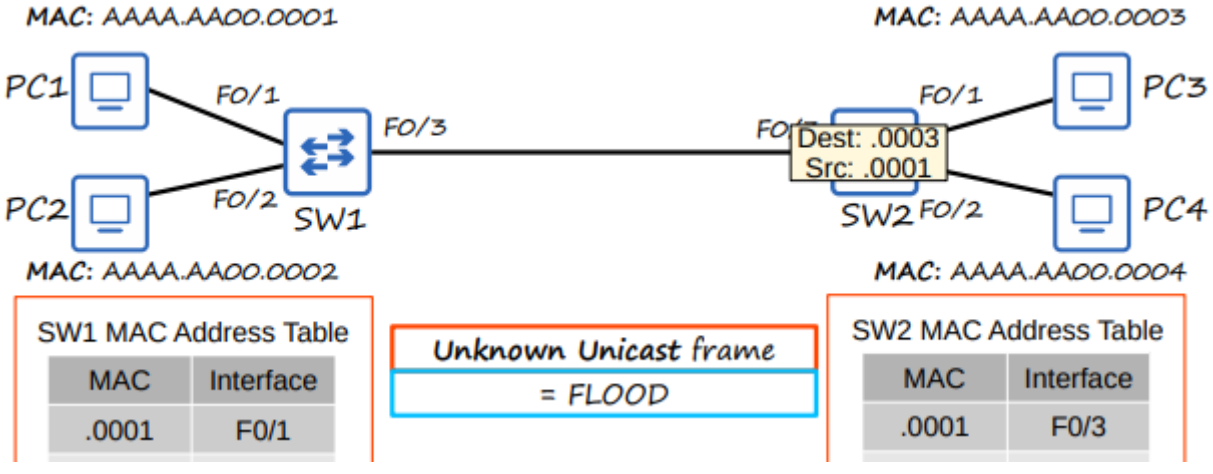
We will look at how data is sent between multiple switches within a network. The example above is when we have two nodes communicating within the same LAN.



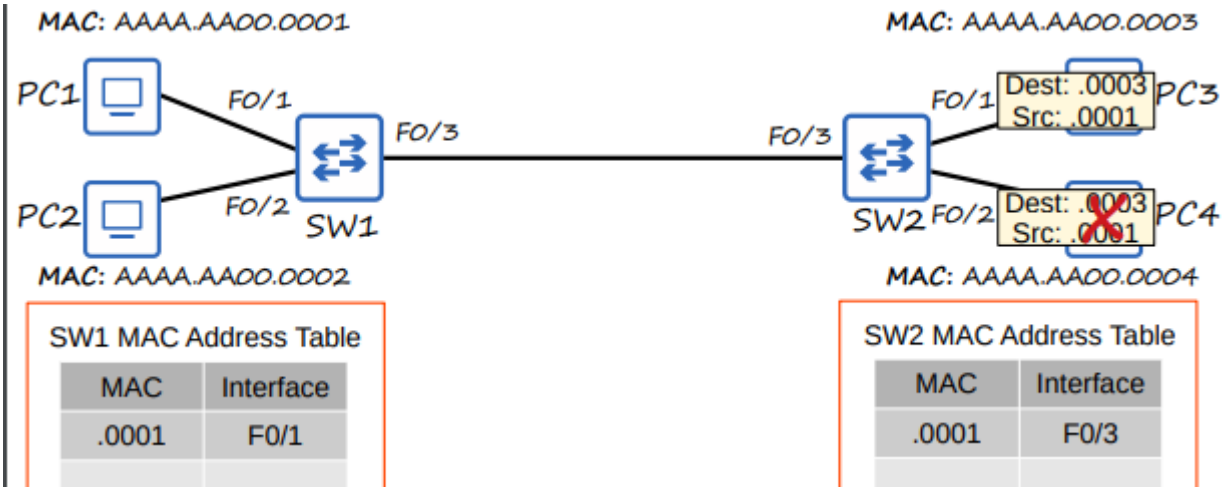
But what if PC1 wanted to send data across to PC3 over two switches? The same methods would apply. SW1 stores the MAC address & interface of the incoming data then broadcasts the data via an **unknown unicast frame** to PC2 but it is rejected.



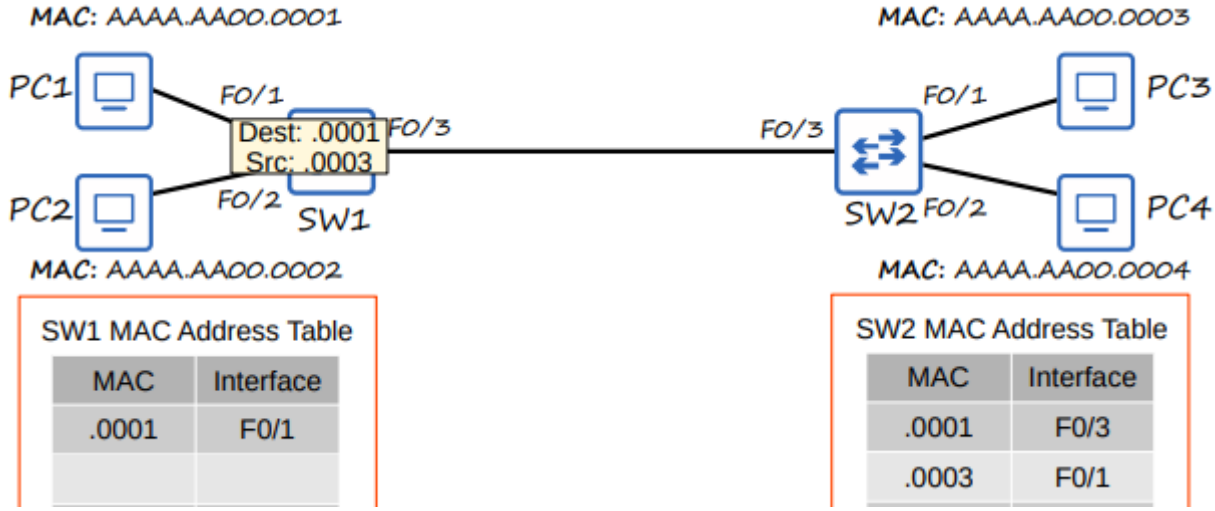
The data will be then sent to SW2, which will again store the MAC address & interface in it's MAC address table.



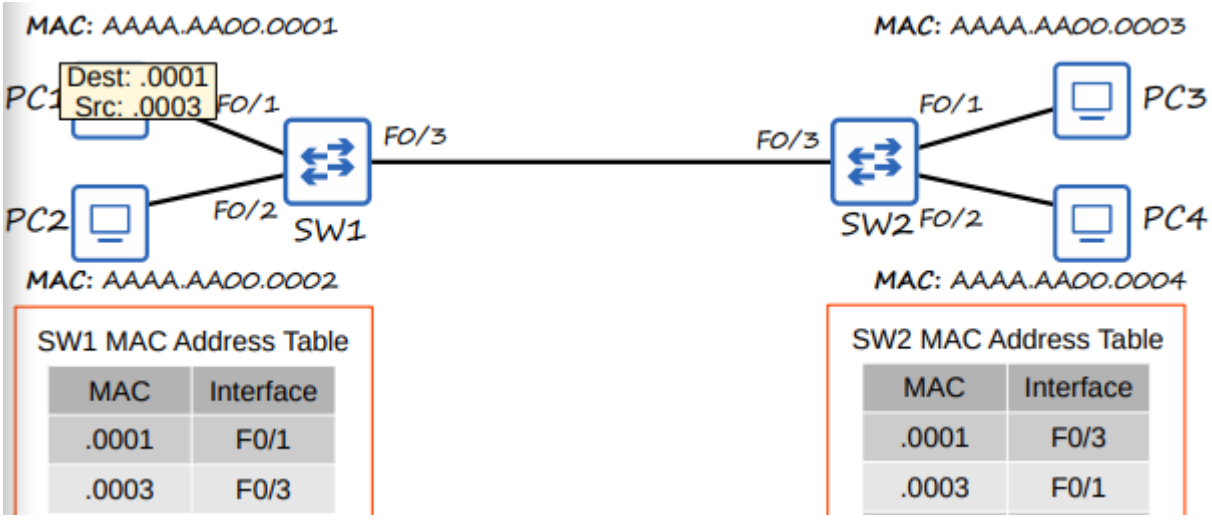
It will send an unknown unicast frame to both PC3 & PC4. PC4 will reject the frame and the frame will be then accepted by PC3.



Now when PC3 sends data back to PC1, the data is sent via a **known unicast frame** because the destination address has been previously stored in the MAC address table of the switch.



SW2 stores the MAC address & interface data from PC3 before sending the frame over to SW1 via a **known unicast frame**.



SW1 looks at the source address of the frame and stores it in the SW1 MAC address table, before sending the data over to PC1.

REMEMBER

- The cables connected to the nodes are **Layer 1 of the OSI Model**.
- **CRC (Cyclic Redundancy Check)** looks for corrupted data within the FCS (trailer) of a frame.
- A **unicast frame** is a frame for just one end host

- A **dynamic MAC address** in a switch learns which interface the data is being sent from once the data is sent. It does not know where the data is sent to so it floods all other nodes with the data until the right node is found.
- The MAC address table **removes** the MAC address after **5 minutes of inactivity**