**Virtual Local Area Network**

Table of Contents

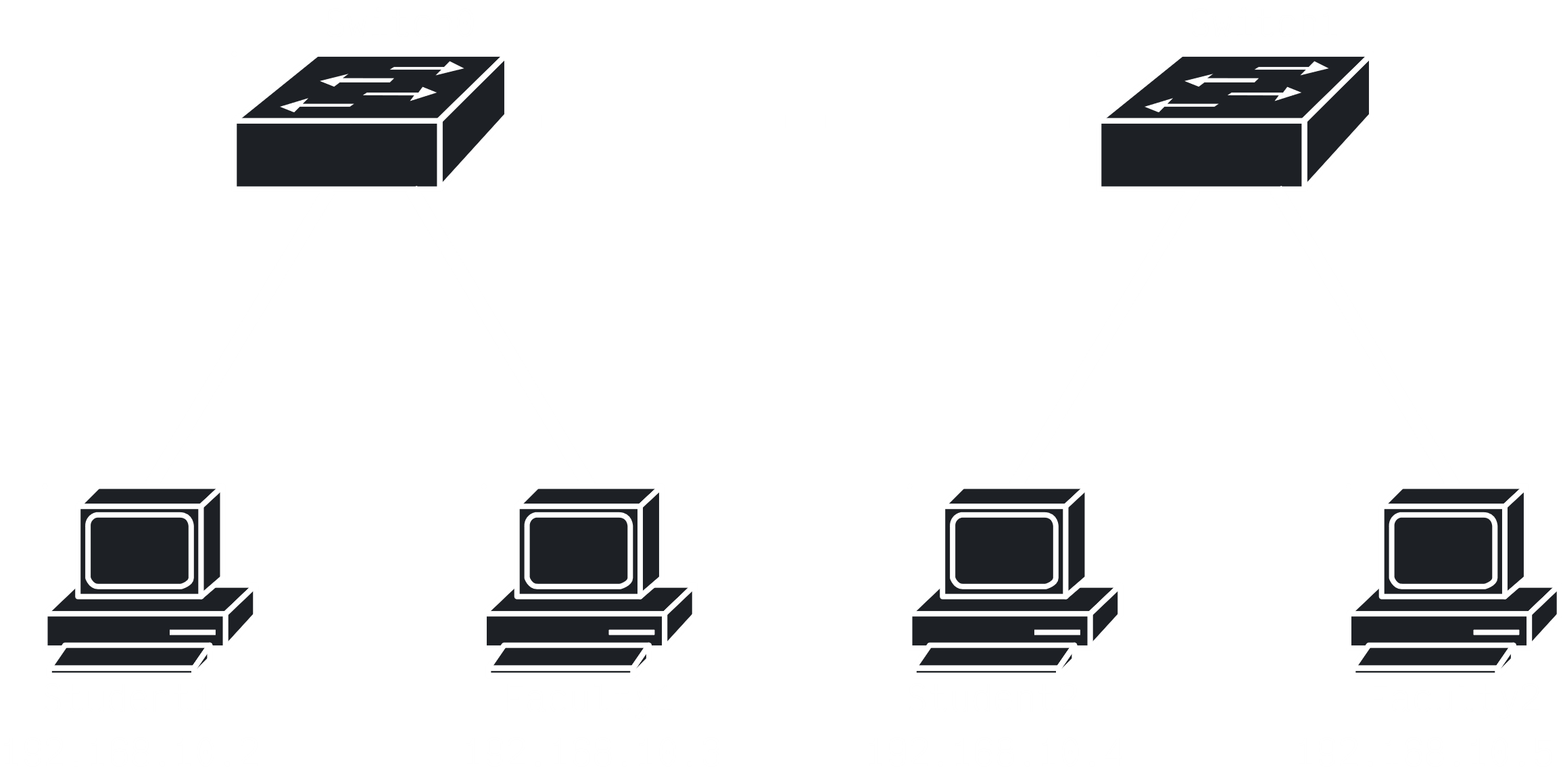
[Setup 3](#_Toc86832918)

[Inter-VLAN Routing 6](#_Toc86832919)

[Router-on-a-Stick 6](#_Toc86832920)

[Multi-Layer Switch 9](#_Toc86832921)

Say we have the following setup:



All the PCs will be able to communicate with each other. However, what if we want **just the Faculty PCs** to communicate with each other. We can of course choose a specific IP address to send data. All the other PCs will get the message but will ideally reject it since it is not meant for them, but that is **not safe**.

One way to bypass this issue is to create a separate network for the Faculty PCs, but as we scale up, this quickly becomes impractical. Instead, we use **Virtual Local Area Networks** (VLANs). Essentially, we create a **virtual** network for the Faculty PCs on which they can communicate.

## Setup

So far, we have never explicitly configured any **switches**. It has not been necessary. Now, we will have to.

Switch>enable  
Switch#configure terminal  
Enter configuration commands, one per line. End with CNTL/Z.  
Switch(config)#hostname switch0  
switch0(config)#vlan 10  
switch0(config-vlan)#name student  
switch0(config-vlan)#exit  
switch0(config)#vlan 20  
switch0(config-vlan)#name faculty  
switch0(config-vlan)#exit

CLI

Above, we have created **two VLANs**, one for the Student PCs and one for the Faculty PCs. Each VLAN has been assigned an **ID**, 10 and 20 respectively.

IDs can range from 1 to 4094, but note that ID 1 and ID 99 is generally reserved. ID 1 is the **default VLAN**, which is assigned to any new PC in the network that does not yet have an explicitly assigned VLAN. ID 99 is for the **native VLAN**, which deals with untagged frames, which we will be looking into soon.

Unlike the default VLAN, the **native VLAN** must be **explicitly configured**, so let’s do that next.

switch0(config)#vlan 99  
switch0(config-vlan)#name native  
switch0(config-vlan)#exit

CLI

We can **view** all the VLANs we have created like this:

switch0#show vlan brief

CLI

However, we still need to **assign ports** to the VLANs we created. By doing this, we will identify data travelling between which ports are allowed to use a particular VLAN.

switch0(config)#interface fastEthernet0/1  
switch0(config-if)#switchport mode access  
switch0(config-if)#switchport access vlan 10  
switch0(config-if)#no shutdown  
switch0(config-if)#exit

CLI

Notice that we identified a **switchport mode**. For this case, the mode we have used is **access**. This essentially means that this interface will only be able to access **one specific VLAN**. We could also have used **trunk** mode, which would allow access to **multiple VLANs**.

For the diagram provided above, the connections between the PCs and the respective switches are all **access links**, while the connection between the switches is a **trunk link**. Thus, we have to set up the access links in a similar manner as shown above, and for the trunk link, we can assign the link like this:

switch0(config)#interface fastEthernet0/3  
switch0(config-if)#switchport mode trunk  
  
switch0(config-if)#  
%LINEPROTO-5-UPDOWN: Line protocol on Interface FastEthernet0/3, changed state to down  
  
%LINEPROTO-5-UPDOWN: Line protocol on Interface FastEthernet0/3, changed state to up  
  
switch0(config-if)#switchport trunk allowed vlan all  
switch0(config-if)#no shutdown  
switch0(config-if)#exit  
switch0(config)#

If we view all the VLANs again, the ports that can use each VLAN should also be shown. Note that the trunk connection will not be shown in the CLI tab, but can be verified from the Config tab.

The above setup was for just **one switch**. The other switch needs to be appropriately setup as well.

## Inter-VLAN Routing

Once the setup is complete, PCs on the **student VLAN** will be unable to ping PCs on the **faculty VLAN** and vice versa. However, they will still be able to ping PCs on the **same VLAN** as themselves.

To allow devices on different VLANs to communicate with each other, we need to implement **Inter-VLAN Routing**. This can be achieved through two approaches:

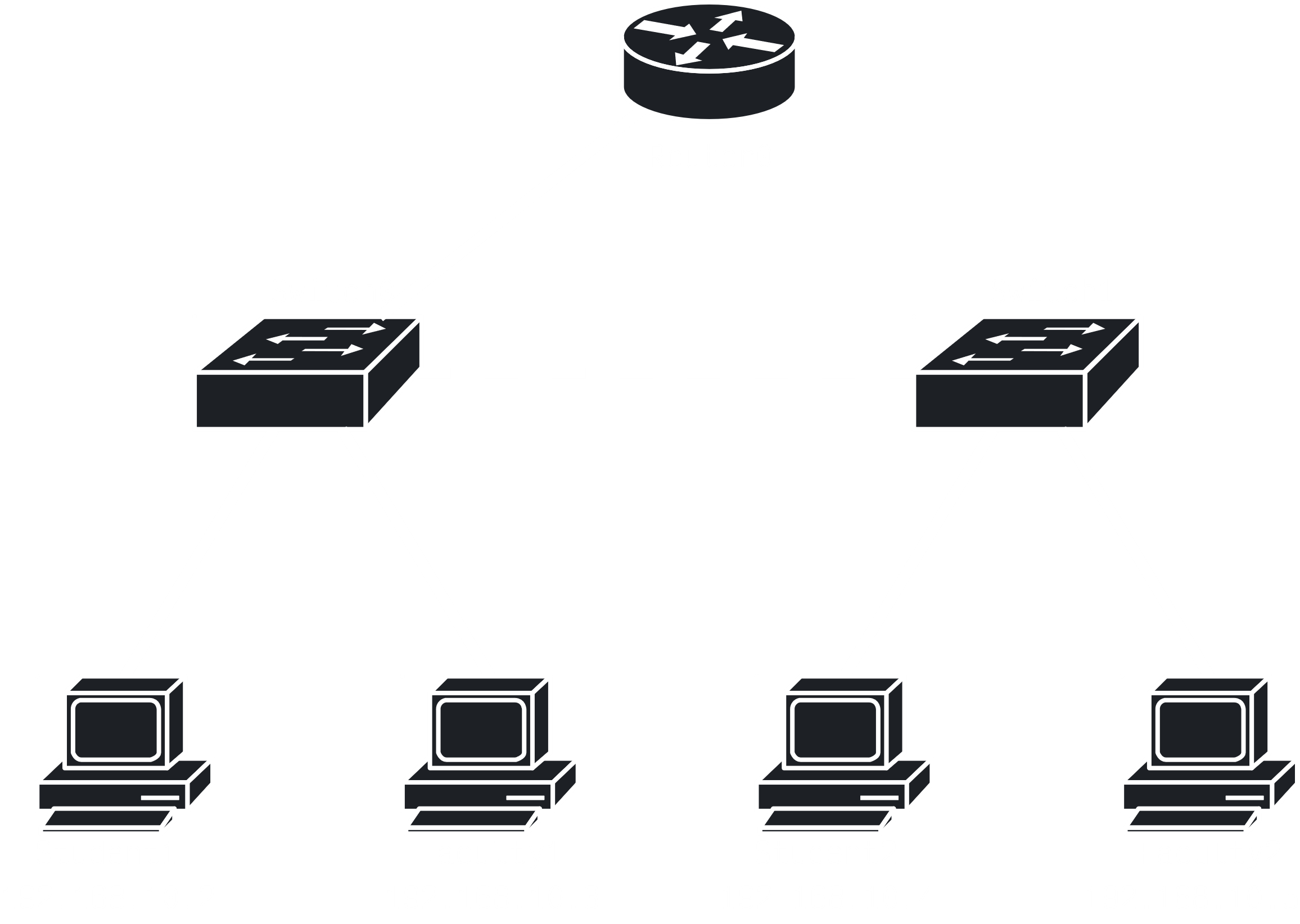
1. Router-on-a-Stick
2. Multi-Layer Switch

### Router-on-a-Stick

If we use the **Router-on-a-Stick Approach**, we will have to introduce a **router**. In that router, we take one of the interfaces and configure it as a **trunk port** so that it can understand **VLAN tags**. The VLAN tags are what allow us to differentiate between data from different VLANs.

For each VLAN on the trunk port, we will be creating a separate **sub-interface**, each with a separate IP address from the VLAN it will be working with. Essentially, the sub-interface for a VLAN will be its **default gateway**.

After introducing the router, the new diagram looks like this:



Notice that there is only **one new connection** to one of the switches, not with both. For that switch, we need to configure the new interface at the switch, just like we did before. The connection will be a **trunk connection**.

Another issue is that, for inter-VLAN routing to work, the PCs on each VLAN need to be on **separate networks**. This does not require any physical changes of course, but we do need to change the **IP addresses**.

|  |  |  |
| --- | --- | --- |
| **PC** | **IP Address** | **Default Gateway** |
| Student1 |  |  |
| Student2 |  |  |
| Faculty1 |  |  |
| Faculty2 |  |  |

Finally, we can move on to configuring the **router**. In the CLI tab for the router, we first configure each of the **sub-interfaces**.

Router>enable  
Router#configure terminal  
Enter configuration commands, one per line. End with CNTL/Z.  
Router(config)#hostname router1  
router1(config)#interface gigabitEthernet0/0.10  
router1(config-subif)#encapsulation dot1Q 10  
router1(config-subif)#ip address 192.168.10.1 255.255.255.0  
router1(config-subif)#no shutdown  
router1(config-subif)#exit

CLI

The command encapsulation dot1Q 10 essentially defines a protocol, the **Encapsulation Dot1Q Protocol**, which will be used for inter-VLAN communication.

In a similar manner, we must configure the second sub-interface.

Lastly, remember that we also need to activate the main interface, i.e.

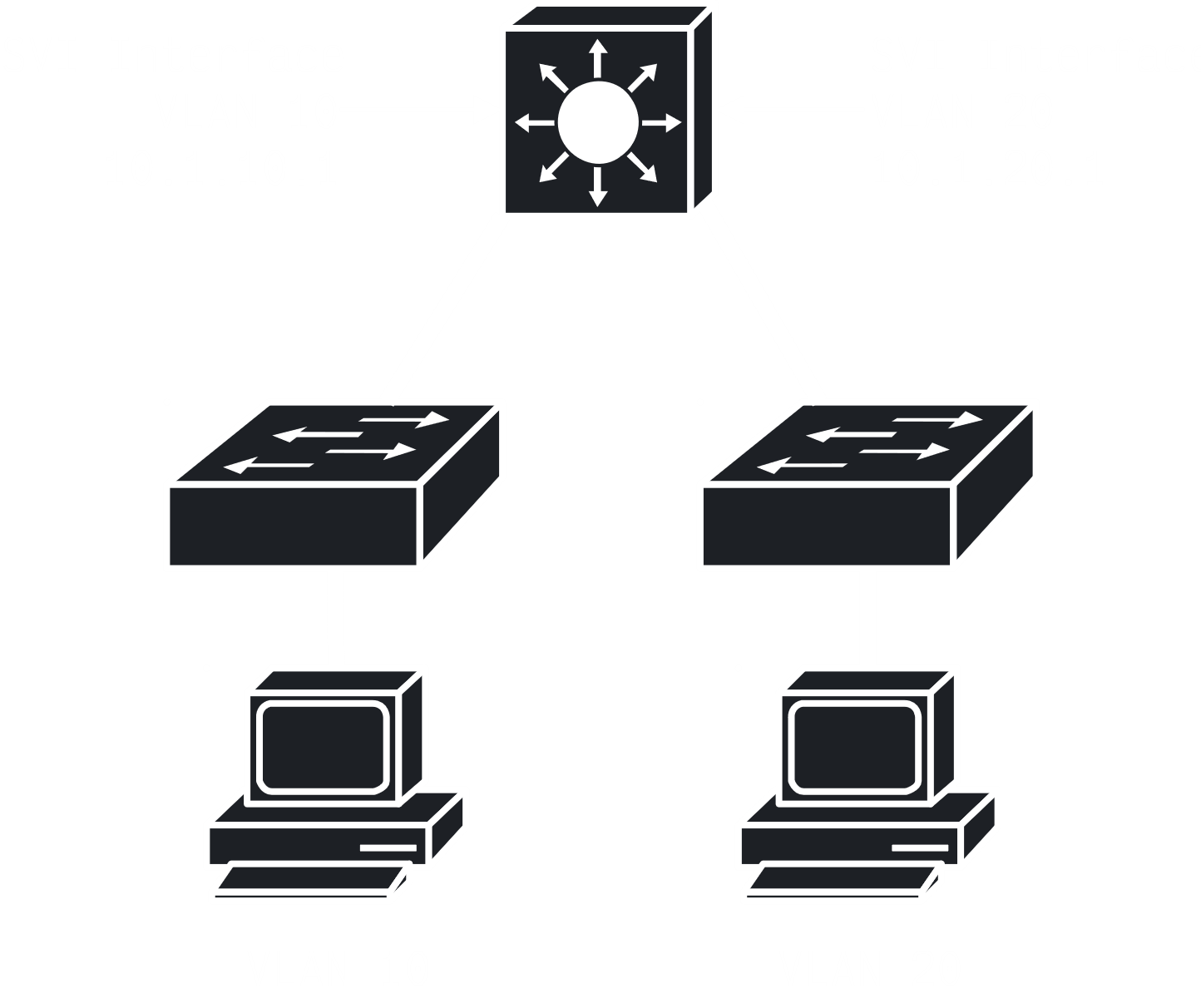
router1(config)#interface gigabitEthernet0/0  
router1(config-if)#no shutdown

CLI

Once all of this is done, all the PCs will again be able to communicate with each other. However, since they are now on different networks, **broadcast** messages will not travel between them. Thus, we have achieved both security and complete communication.

### Multi-Layer Switch

We can also implement Inter-VLAN Routing using a **Multi-Layer Switch**, which works in both layers 2 and 3. This approach is used more often in industries because it provides more VLANs and is faster.



Similar to how we created sub-interfaces in the previous approach, now we will be creating **Switch Virtual Interfaces** (SVIs), which works similarly. We will have one SVI for each VLAN with an IP address that is the default gateway for the VLAN. The only difference between this and creating sub-interfaces in the Router-on-a-Stick Approach, is that packets remain within the switch and are not being routed to a different network, which increases performance.