**SCHOOL OF COMPUTER SCIENCE**

**UNIVERSITY OF PETROLEUM AND ENERGY STUDIES**

**DEHRADUN, UTTARAKHAND**



**Data Communication and Networks**

**LABORATORY FILE**

**(2024-2025)**

**For**

**Vth Semester**

**Submitted To: Submitted By:**

Prof. Abhishek Yadav Mr. Akshat Negi

Assistant Professor S.S. 500106533(SAP ID)

[Vth Semester] R2142220414(Roll No.)

School of Computer Sciences B.Tech. CSF (Batch-1)

**LAB EXPERIMENT – 1**

**OSI MODAL AND DEVICES**

**OSI MODAL**

The OSI Modal consist of seven different interconnected layers: -

1. Physical Layer
2. Data Link Layer
3. Network Layer
4. Transport Layer
5. Session Layer
6. Presentation Layer
7. Application Layer

1. Physical Layer: - It’s the lowest layer of the OSI Modal which helps to create the physical layer connections between multiple devices. It is use to transmit data in the form of bits connecting one node to another. On receiving data signals from somewhere it converts it into the 0s and 1s form. Hubs, Repeaters, Modems are the physical layer devices.

2. Data Link Layer: - It basically helps to connect the nodes and helps to convert the packets into frames for further transacts to the physical layer. Switch and Bridges are the devices of the Data Link Layer.

3. Network Layer: - It helps in the transmission of the data from a host to different other locations. The data are in the form of packets in the network layer. It also helps the routing connection of the devices using their IP Addresses as the headers of the Data.

4. Transport Layer: - It helps in conversion of segments to the smaller units using headers as the locations of the data.

5. Session Layer: - It usually is use for establishment of connection, maintenance of sessions, and authentication, and also helps in secure environment between the user and the server.

6. Presentation Layer: - Encryption/ Decryption: Data encryption translates the data into another form or code. The encrypted data is known as the ciphertext and the decrypted data is known as plain text. A key value is used for encrypting as well as decrypting data.

7. Application Layer: The Application layer which is implemented by the network applications. These applications produce the data to be transferred over the network. This layer also serves as a window for the application services to access the network and for displaying the received information to the user.

**The OSI MODAL DEVICES**

**HUB: -**

* It basically belongs the first layer of the OSI modal that is the physical layer.
* It usually belongs to broadcast data in the interconnected devices of a LAN.
* These are not smart enough to circulate the data from a particular host to a particular device as it transits the data to all the devices present in a LAN.
* These are cheaper than the switches and routers.
* These are insecure nowadays as it broadcast the data to every layer in the network.

**Switch: -**

* It basically belongs to the 2nd layer of the OSI Modal that is the data link layer.
* It is quite smarter than the hub as it is the next gen. modal with some new functionalities.
* It helps to transit the data from a particular device to another particular device using IP Addresses without broadcasting like HUBS.
* These are quite expensive than HUBS and quite cheaper than routers.

**ROUTERS: -**

* It basically belongs to the third layer of the OSI modal that is the network layer.
* It usually belongs to transmit data using the MAC addresses of the devices and create a interconnection between the two LAN Area Networks which can communicate between each other.
* It has the latest technology and most advanced features as its intelligence is more as compared to hubs and switches.
* These devices are expensive that the HUBS and Switches.

**LAB EXPERIMENT – 2**

**BIT STUFFING AND DE-STUFFING**

**Bit Stuffing** and **De-Stuffing** are techniques used in synchronous data transmission to prevent the occurrence of consecutive ones that could be misinterpreted as control characters. In certain protocols, such as HDLC, a continuous stream of ones is used to indicate the end of a frame. If a series of ones appears unintentionally within the data, it can lead to premature termination of the frame.

**Example**

Consider the following data sequence: 1111111111

**Bit Stuffing:**

111110111110

**Bit De-Stuffing:**

1111011110

As you can see, the extra zero inserted during bit stuffing is removed during de-stuffing, restoring the original data.

**Program for Bit Stuffing**

**Process:**

* The data is transmitted bit by bit.
* If five consecutive ones are encountered, an extra zero is inserted into the data stream.
* This process continues until the end of the frame.

**Purpose:** To ensure that a series of five consecutive ones cannot appear within the data, preventing false termination of the frame.

#include <stdio.h>

#include <string.h>

int main()

{

    char stuff[100];

    int i, count = 0;

    printf("Enter the BIT--> ");

    scanf("%s", stuff);

    printf("Bit-stuffed--> ");

    for (i = 0; stuff[i] != 0; i++)

    {

        printf("%c", stuff[i]);

        if (stuff[i] == '1')

        {

            count++;

            if (count == 5)

            {

                printf("0");

                count = 0;

            }

        }

        else

        {

            count = 0;

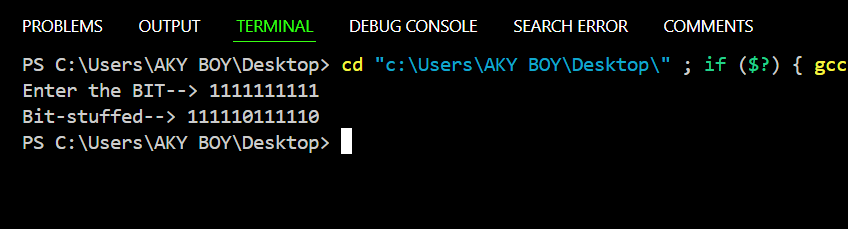
        }

    }

    printf("\n");

    return 0;

}



**Program for Bit De-Stuffing**

**Process:**

* The received data is examined bit by bit.
* If five consecutive ones are encountered followed by a zero, the zero is removed.
* This process continues until the end of the frame.

**Purpose:** To restore the original data by removing the extra zeros inserted during bit stuffing.

#include <stdio.h>

#include <string.h>

int main() {

    char data[100], dest[100];

    int i, j, count = 0;

    printf("Enter the BIT--> ");

    scanf("%s", data);

    printf("Bit De-stuffed--> ");

    for (i = 0, j = 0; data[i] != 0; i++)

    {

        if (data[i] == '1')

        {

            count++;

            if (count == 5 && data[i + 1] == '0')

            {

                count = 0;

                i++;

            }

        }

        else

        {

            count = 0;

        }

        dest[j++] = data[i];

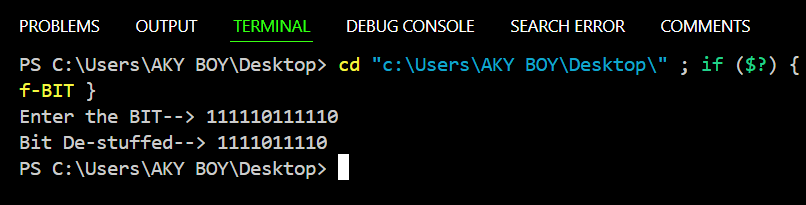
    }

    dest[j] = '\0';

    printf("%s\n", dest);

    return 0;

}



**LAB EXPERIMENT – 3**

**CRC and Hamming code**

A **CRC (Cyclic Redundancy Check) Experiment Lab** is a practical exercise to demonstrate error detection in digital communication. It involves implementing and testing CRC algorithms using a sender-receiver model. Here's how you can structure the lab:

**Objective**

1. Understand how CRC works for error detection.
2. Implement a CRC algorithm in C.
3. Simulate data transmission with errors and validate error detection.

**Requirements**

1. A C compiler (e.g., GCC).
2. A computer to run the code.

#include<stdio.h>

#include<string.h>

#define N strlen(gen\_poly)

char data[28];

char check\_value[28];

char gen\_poly[10];

int data\_length,i,j;

void XOR()

    {

        for(j = 1; j < N; j++)

        check\_value[j] = (( check\_value[j] == gen\_poly[j])?'0':'1');

    }

void receiver()

{

    printf("Enter the received data: ");

    scanf("%s", data);

    printf("Data received: %s", data);

    crc();

    for(i=0;(i<N-1) && (check\_value[i]!='1');i++);

        if(i<N-1)

            printf("\nError detected\n\n");

        else

            printf("\nNo error detected\n\n");

}

void crc()

{

    for(i=0;i<N;i++)

        check\_value[i]=data[i];

    do{

        if(check\_value[0]=='1')

            XOR();

        for(j=0;j<N-1;j++)

            check\_value[j]=check\_value[j+1];

        check\_value[j]=data[i++];

    }

    while(i<=data\_length+N-1);

}

int main()

{

    printf("\nEnter data to be transmitted: ");

    scanf("%s",data);

    printf("\nEnter the CRC polynomial: ");

    scanf("%s",gen\_poly);

    data\_length=strlen(data);

    for(i=data\_length;i<data\_length+N-1;i++)

        data[i]='0';

    printf("Data padded with n-1 zeros : %s",data);

    crc();

    for(i=data\_length;i<data\_length+N-1;i++)

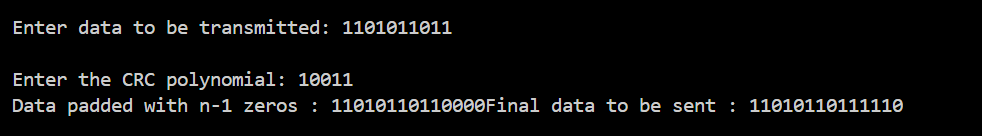
        data[i]=check\_value[i-data\_length];

    printf("Final data to be sent : %s\n\n",data);

    receiver();

        return 0;

}



A **Hamming Code Lab Experiment** focuses on implementing and testing error detection and correction using Hamming codes. This practical exercise demonstrates the ability to detect and correct single-bit errors in data transmission.

**Objective**

1. Understand the principles of Hamming codes for error detection and correction.
2. Implement Hamming code generation, error simulation, and error correction.
3. Experiment with single-bit error correction in transmitted data.

**Requirements**

1. A C compiler (e.g., GCC) or any programming environment.
2. Basic understanding of binary representation.

#include <string.h>

#include <math.h>

#include <stdlib.h>

#include <stdio.h>

    int MaxLength;

    int length;

    int parity;

    char \*HammingString=NULL;

    void EnterParameters(int \**length*, int \**parity*)

    {

        printf("Enter the maximum length: ");

        scanf("%d", *length*);

        printf("Enter the parity (0=even, 1=odd): ");

        scanf("%d", *parity*);

    }

    void CheckHamming(char \**HammingString*, int *parity*)

    {

        int i, j, k, start, length, ParityNumber;

        printf("Enter the Hamming code: ");

        scanf("%s", *HammingString*);

        int ErrorBit = 0;

        length = strlen(*HammingString*);

        length--;

        if (length > MaxLength)

        {

            printf("\n\*\* Invalid Entry - Exceeds Maximum Code Length of %d\n\n", MaxLength);

            return;

        }

        ParityNumber = ceil(log(length)/log(2));

        for(i = 0; i < ParityNumber; i++)

        {

            start = pow(2, i);

            int ParityCheck = *parity*;

            for(j = start; j < length; j=j+(2\*start))

            {

                for(k = j; (k < ((2\*j) - 1)) && (k < length); k++)

                {

                    ParityCheck ^= (*HammingString*[length - k] - '0');

                }

            }

                ErrorBit = ErrorBit + (ParityCheck \* start);

            }

        if(ErrorBit == 0)

        {

            printf("No error \n");

        }

        else

        {

            printf("There is an error in bit: %d\n", ErrorBit);

            if(*HammingString*[length - ErrorBit] == '0')

            {

*HammingString*[length - ErrorBit] = '1';

            }

            else

            {

*HammingString*[length - ErrorBit] = '0';

            }

            printf("The corrected Hamming code is: %s \n", *HammingString*);

        }

    }

    int main()

    {

        int parity;

        int choice = 0;

            printf("Error detection/correction: \n");

            printf("----------------------------\n");

            printf("1) Enter parameters \n");

            printf("2) Check Hamming code \n");

            printf("3) Exit \n");

            printf("\nEnter selection: ");

            scanf("%d", &choice);

            while (choice != 3)

            {

                if (choice == 1)

                {

                    EnterParameters(&MaxLength, &parity);

                    HammingString = (char\*) malloc (MaxLength \* sizeof(char));

                    main();

                }

                else if (choice == 2)

                {

                    CheckHamming(HammingString, parity);

                    main();

                }

                else

                {

                    printf("Valid options are 1, 2, or 3. Quitting program. \n");

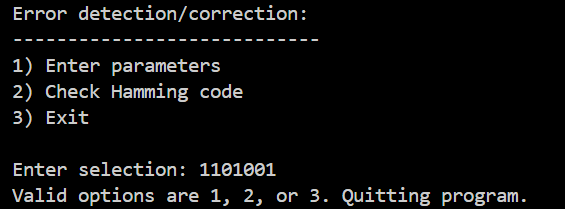
                    exit(0);

                }

            }

            exit(0);

    }



**LAB EXPERIMENT – 4**

**Familiarization of Network IP & Subnetting & Supernetting**

**Aim:** Study of Network IP and Sub Netting & Super Netting.

* Classification of IP address

**Apparatus (Software):** No Software or hardware needed.

**Theory:** An Internet Protocol address (IP address) is a numerical label assigned to each device (e.g., computer, printer) participating in a computer network that uses the Internet Protocol for communication. An IP address serves two principal functions: host or network interface identification and location addressing. Its role has been characterized as follows: "A name indicates what we seek. An address indicates where it is. A route indicates how to get there."

A sub network, or subnet, is a logically visible subdivision of an IP network. The practice of dividing a network into two or more networks is called sub netting. A super network, or super net, is an Internet Protocol (IP) network that is formed from the combination of two or more networks (or subnets) with a common Classless Inter-Domain Routing (CIDR) prefix. The new routing prefix for the combined network aggregates the prefixes of the constituent networks. It must not contain other prefixes of networks that do not lie in the same routing path. The process of forming a super net is often called super netting, prefix aggregation, route aggregation, or route summarization.

**Procedure:** Following is required to be study under this practical.

* Classification of IP address
* Sub netting

Why we Develop sub netting and How to calculate subnet mask and how to identify subnet address.

* Super netting

Why we develop super netting and how to calculate super net mask and how to identify super net address.

**Conclusion:** Gain the knowledge about the IP address and Sub netting & Super netting.

1. **CLASSIFICATION OF IP ADDRESS: -**

**IP Address Classification**

IP addresses are divided into five classes: A, B, C, D, and E. Each class is assigned a specific range of addresses and has a different purpose.

**Class A**

* **Range:** 0.0.0.0 to 127.255.255.255
* **Purpose:** Primarily used for large networks, such as those of major organizations or countries.
* **Number of hosts:** Up to 16,777,214 hosts per network.

**Class B**

* **Range:** 128.0.0.0 to 191.255.255.255
* **Purpose:** Used for medium-sized networks, such as those of universities or large corporations.
* **Number of hosts:** Up to 65,534 hosts per network.

**Class C**

* **Range:** 192.0.0.0 to 223.255.255.255
* **Purpose:** Used for small networks, such as those of small businesses or homes.
* **Number of hosts:** Up to 254 hosts per network.

**Class D**

* **Range:** 224.0.0.0 to 239.255.255.255
* **Purpose:** Reserved for multicast addressing, which allows data to be sent to a group of hosts simultaneously.

**Class E**

* **Range:** 240.0.0.0 to 255.255.255.255
* **Purpose:** Reserved for experimental use and future expansion.

**SUBNETTING –**

Subnetting is the process of dividing a network into smaller, more manageable subnetworks. It's done by using some bits of the host ID to create a subnet ID.

**Why we Develop sub netting and how to calculate subnet mask and how to identify subnet address.**

* Subnetting is a network technique that divides large networks into smaller, more manageable subnets. Subnetting is used to improve network performance, security, and IP address utilization.

To calculate a subnet mask and identify a subnet address, you can:

* Convert the IP address and mask to binary
* Determine the network and host portions of the address
* Find the subnet ID in binary by changing all host bits to 0s
* Convert the binary representations to decimals

You can also use CIDR (Classless Inter-Domain Routing) notation to represent an IP address and its subnet mask. CIDR is a common way to represent the network address and the number of bits in the network portion of the address.

**SUPERNETTING -**

Super-netting is a computer networking technique that combines multiple smaller networks into one larger network to improve routing efficiency. It's used to reduce the size of routing tables and the strain on routers, which can be severely affected by large groups of networks.

To calculate a super-net mask and identify a super-net address, you can follow these steps:

1. Choose the required subnets, which must be contiguous.
2. Identify the super-net's network ID by translating the numerical IP addresses of the subnets' IDs into binary form.
3. Calculate the super-net's new subnet mask.

Here's an example of how to calculate a super-net mask:

1. Subtract the number of Class C networks you want from 256. For example, if you want eight Class C networks, the result is 248.
2. Place the value of 248 into the third octet of the mask, resulting in 255.255.248.0.

Super-netting, also known as CIDR, is a way to represent multiple networks with a single network prefix and mask. CIDR notation uses a forward slash (/) and the number of bits in the subnet mask to append the number of subnet mask bits to the network address.

**LAB EXPERIMENT – 5**

**Basic Network Commands**

**Set Up the Devices**

1. Drag and drop:
   * **2 PCs** (e.g., PC0 and PC1).
2. **Connect the devices**:
   * Use a automatic cables to connect PC0 and PC1 directly.

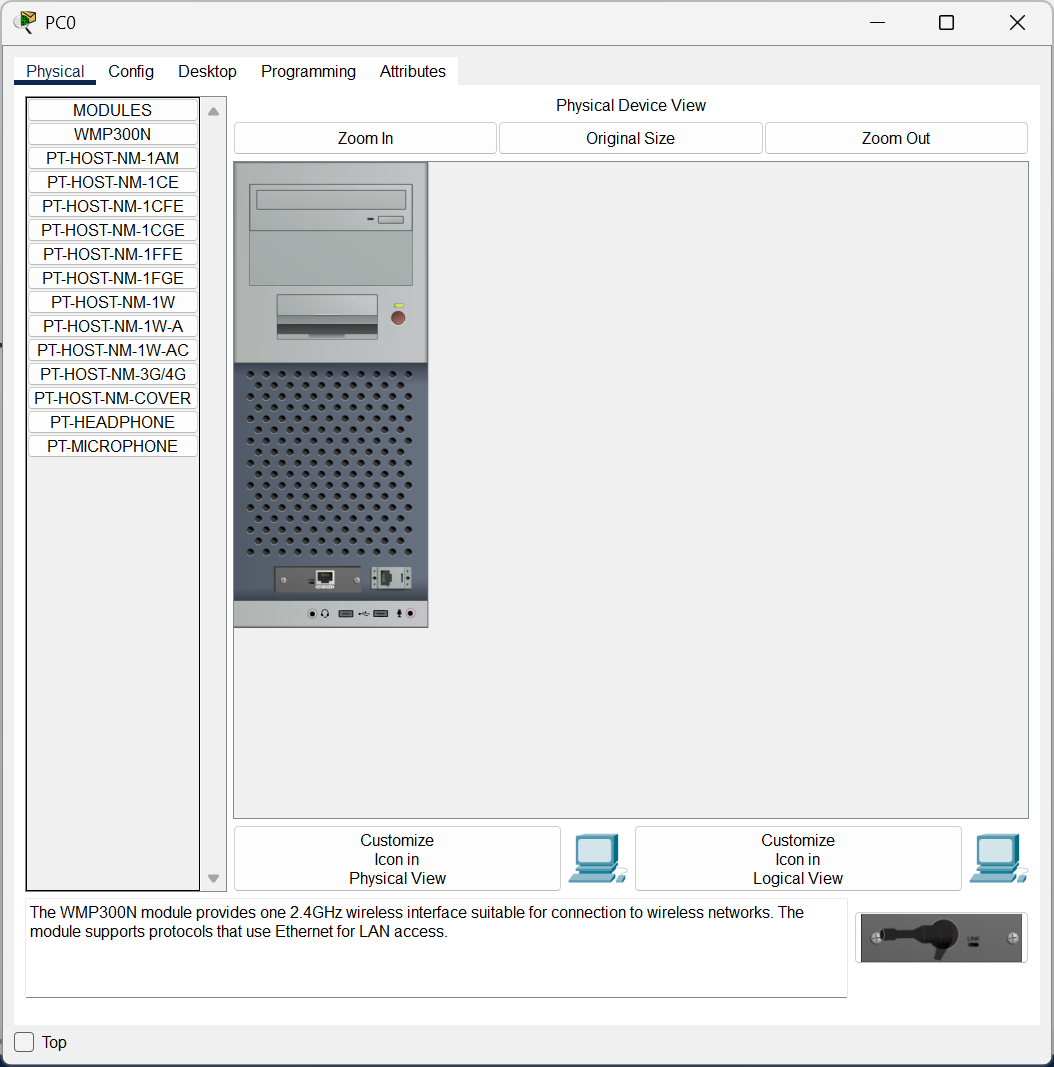
**Assign IP Addresses to the PCs**

Manually assign IP addresses to each PC using the **IP Configuration tab**:

**Steps for Each PC:**

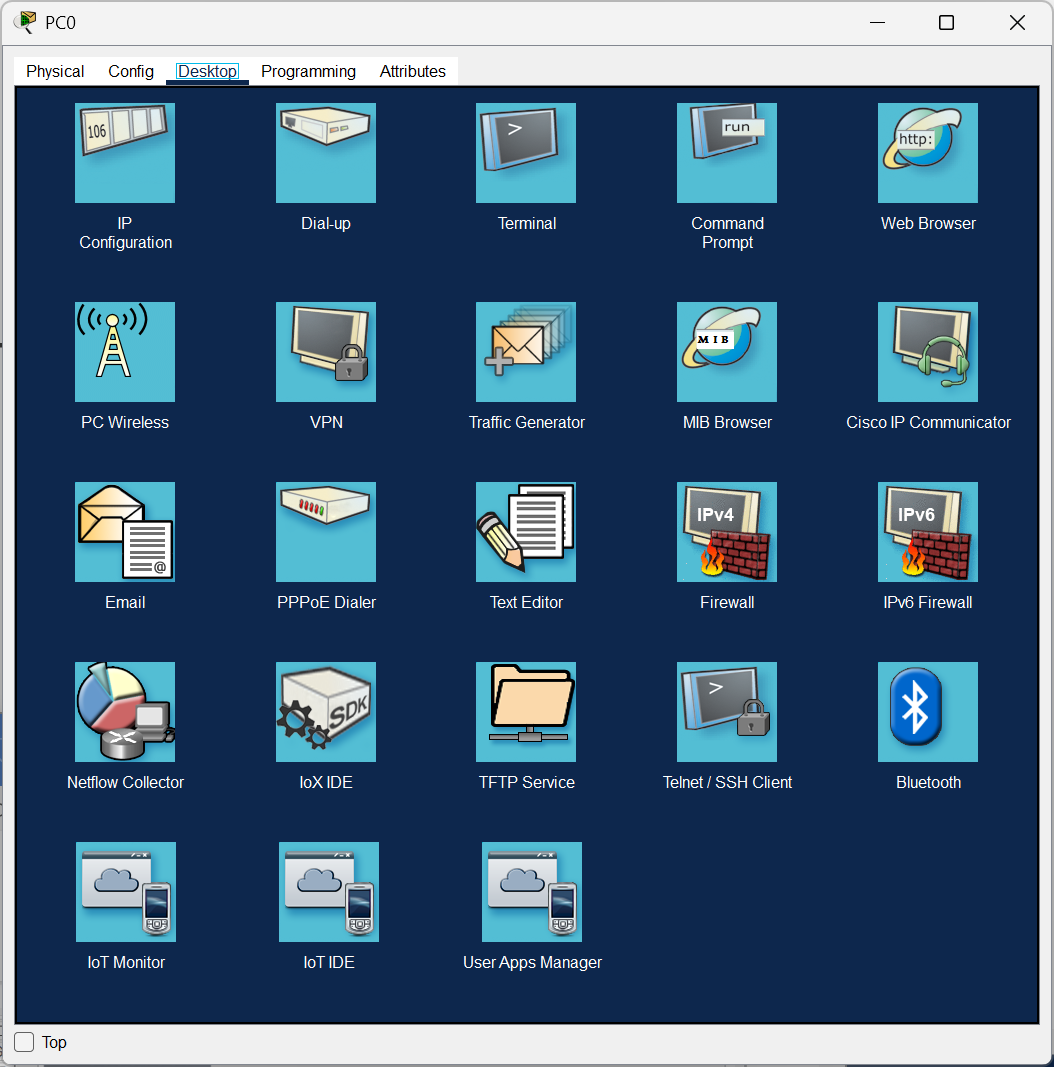
1. Click on **PC0**, go to **Desktop > IP Configuration**, and set:
   * **IP Address**: 10.0.0.1
   * **Subnet Mask**: Leave blank for auto configuration to default.
   * **Default Gateway**: Leave blank if no router is used.
2. Click on **PC1**, go to **Desktop > IP Configuration**, and set:
   * **IP Address**: 10.0.0.2
   * **Subnet Mask**: Leave blank for auto configuration to default.
   * **Default Gateway**: Leave blank if no router is used.

**PC Configuration Setup**

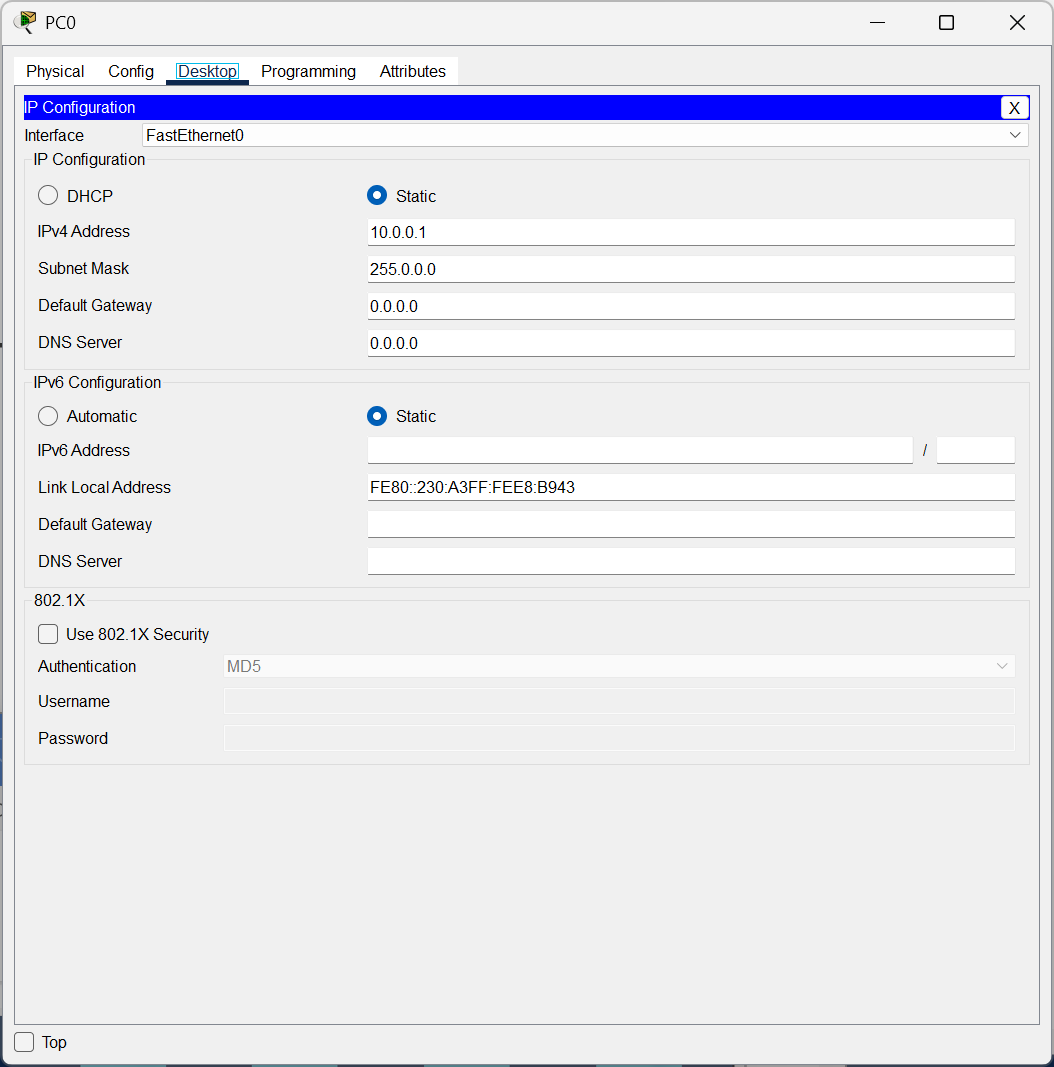


**Open the Command Prompt for IP Configuration Check.**

1. Click on the PC (e.g., **PC0**).
2. Navigate to **Desktop > IP Configuration**.

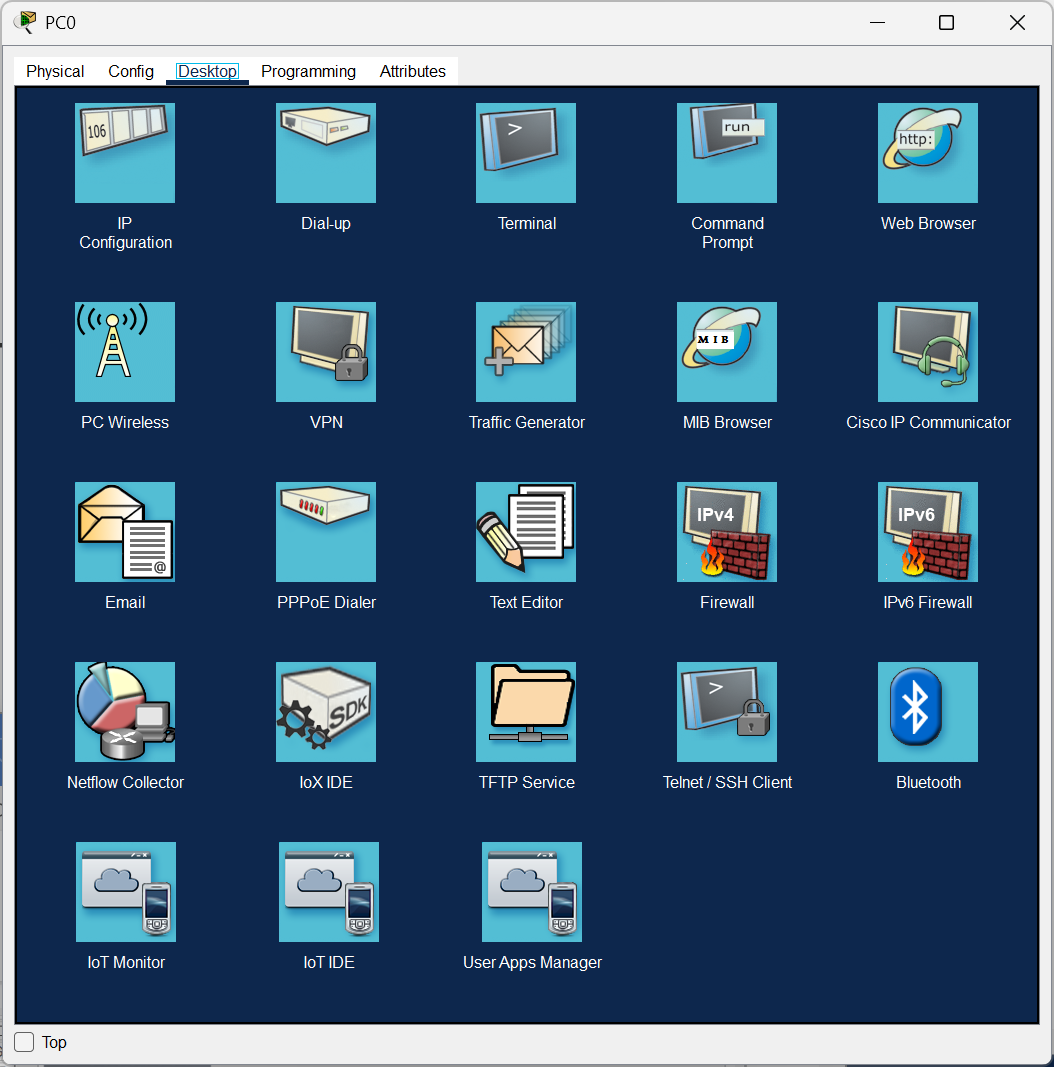


Configure PC0 with giving IP Address as **10.0.0.1**



**Open the Command Prompt for IP Configuration Check.**

1. Click on the PC (e.g., **PC0**).
2. Navigate to **Desktop > Command Prompt**.



Run the **ipconfig** Command

1. Type the following command and press Enter:

**ipconfig**

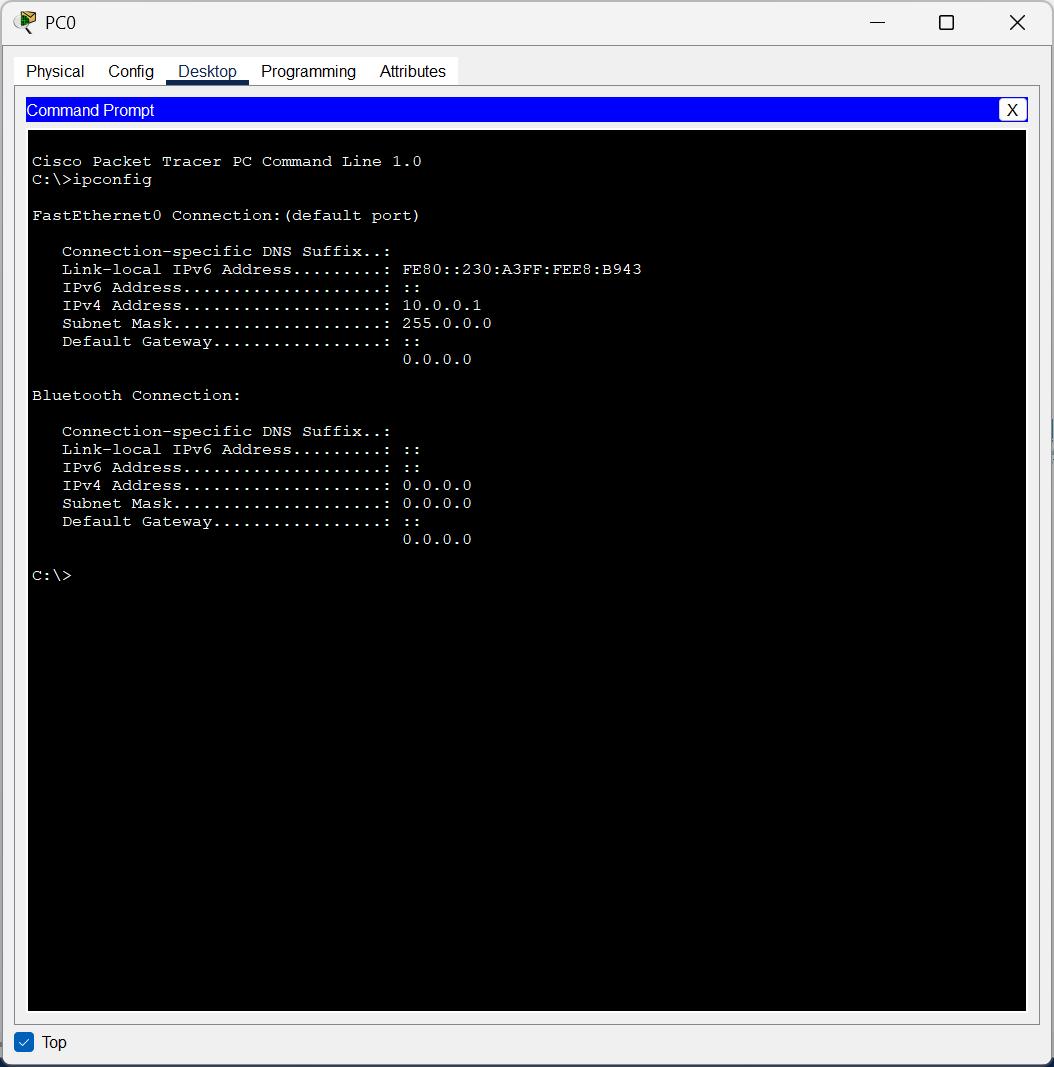
1. The output will display the IP configuration of the PC. Example output:

FastEthernet0 Connection:

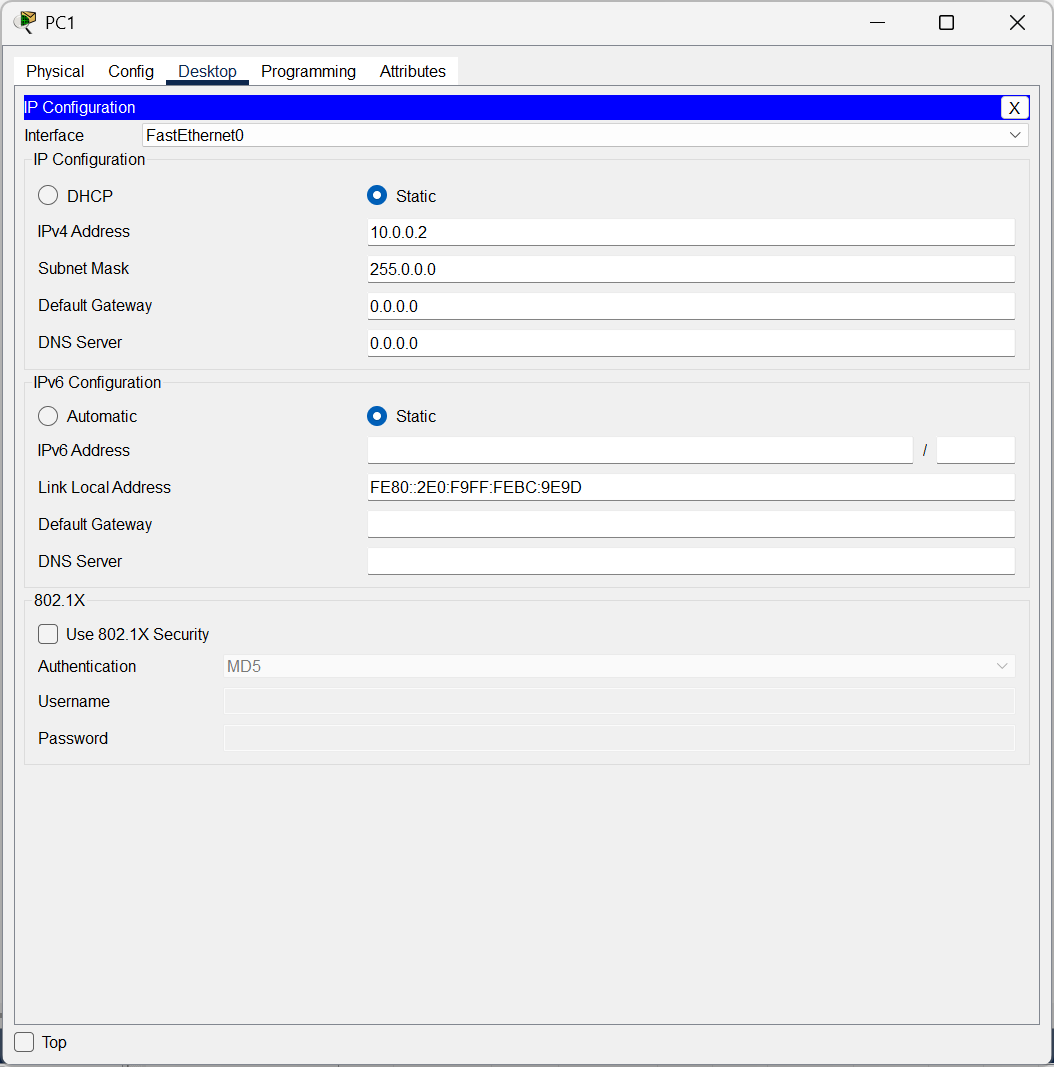
IP Address: 10.0.0.1

Subnet Mask: 255.255.255.0

Default Gateway: (none)



Configure PC1 with giving IP Address as **10.0.0.2**



Run the **ipconfig** Command

* Type the following command and press Enter:

**ipconfig**

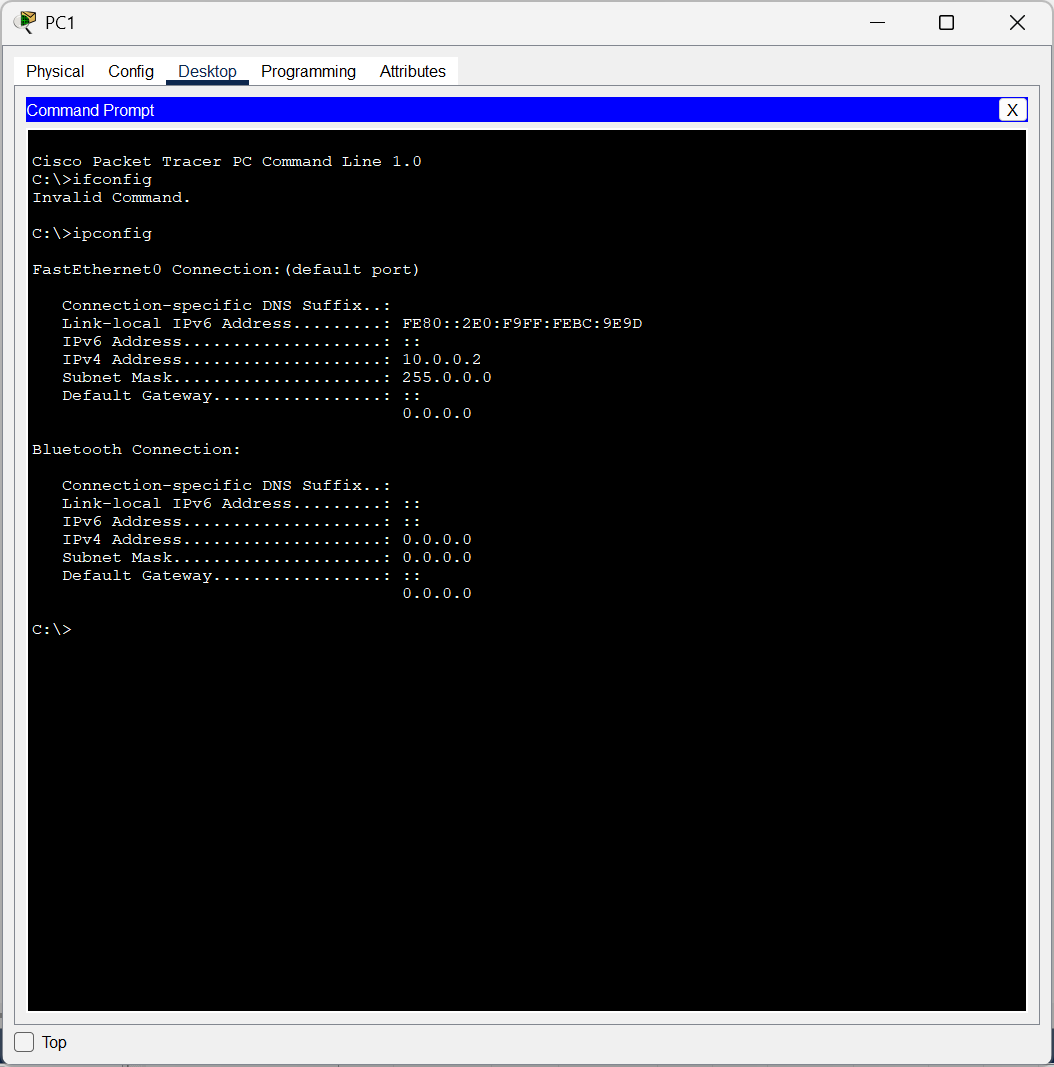
* The output will display the IP configuration of the PC. Example output:

FastEthernet0 Connection:

IP Address: 10.0.0.2

Subnet Mask: 255.255.255.0

Default Gateway: (none)



**Test Connectivity with Ping Command**

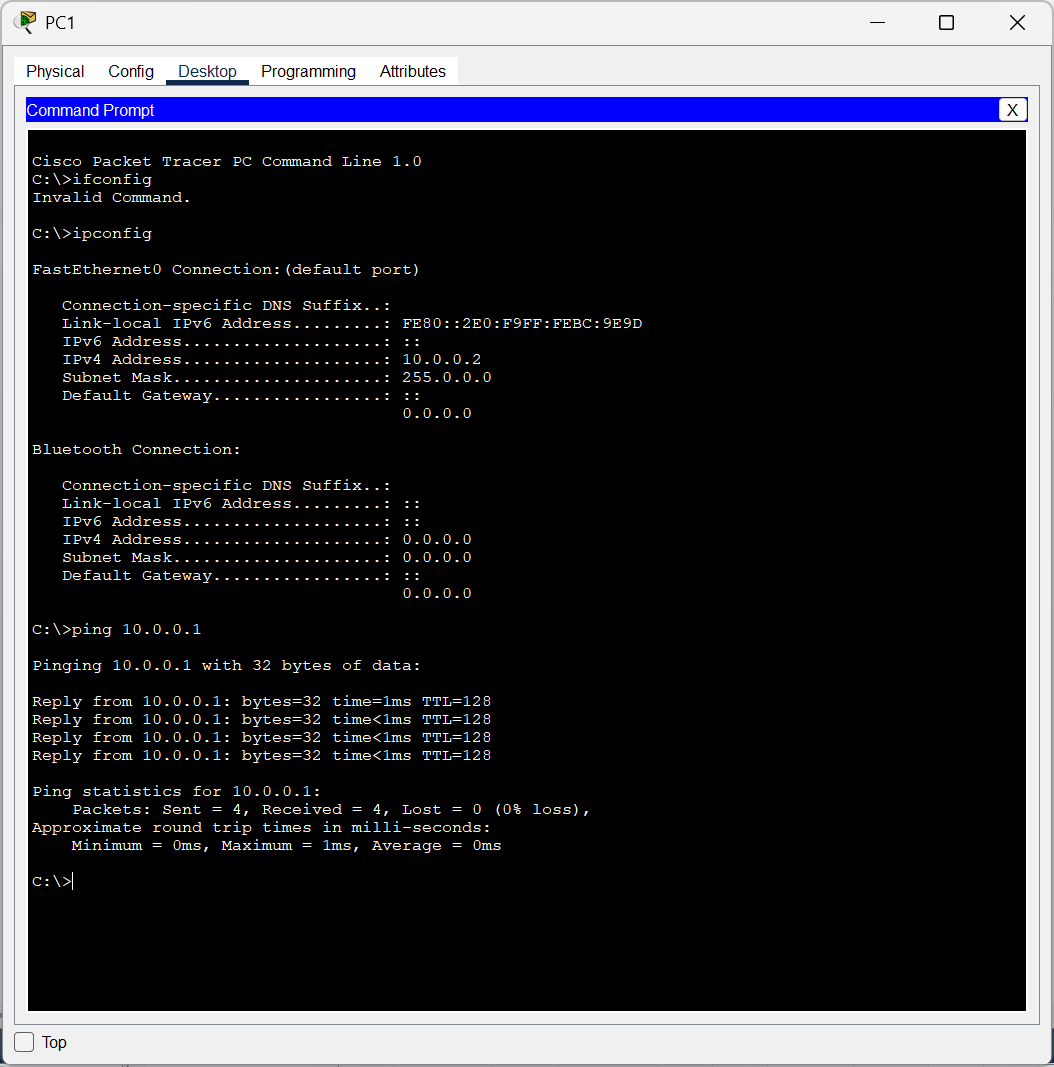
1. On **PC1**, go to **Desktop > Command Prompt**.
2. Type the following command and press Enter:

ping 10.0.0.1

* + If the connection is successful, you will see replies like:

Reply from 10.0.0.1: bytes=32 time<1ms TTL=128

* + If the connection fails, you will see "Request timed out." Check the cable connection and IP configuration.



**Test Path Using Traceroute Command**

1. On **PC1**, go to **Desktop > Command Prompt**.
2. Type the following command and press Enter:

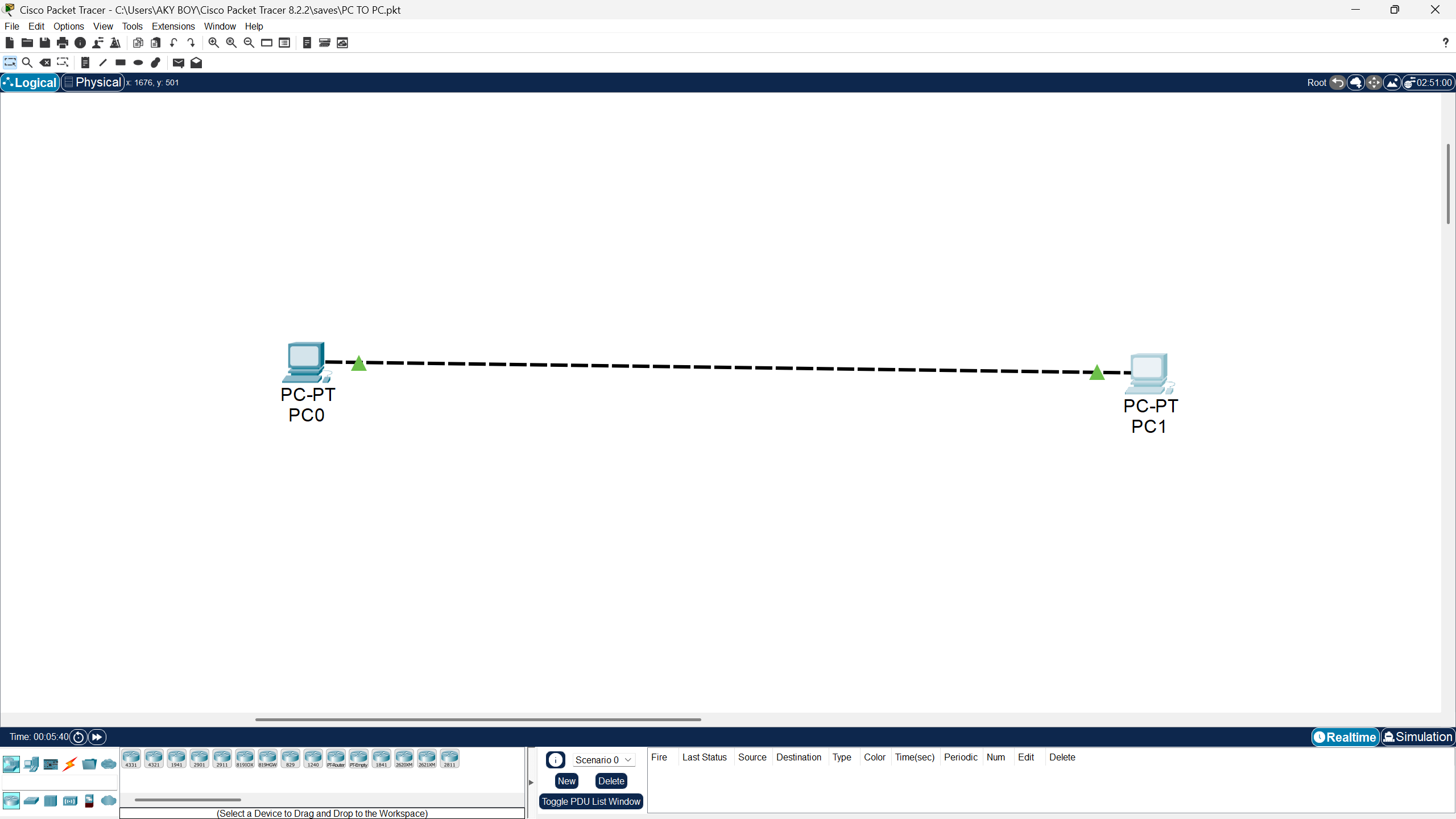
tracert 10.0.0.1

* + The output will show the path taken by the packets to reach the destination. Since there is only one hop in this setup, you will see:

Tracing route to 10.0.0.1 over a maximum of 30 hops: 1 <1 ms <1 ms <1 ms 192.168.1.2 Trace complete.

**CISCO PACKET TRACER DIGITAL REPRESENTATION**

***Representation of 2 PCs connected to each other and performing Network Connection Basic Commands.***



**LAB EXPERIMENT – 6**

**Hub - STAR TOPOLOGY**

To set up a **star topology** with 6 PCs connected through a **hub**.

**Set Up the Devices**

1. Open **Cisco Packet Tracer**.
2. Drag and drop:
   * **1 Hub** (e.g., Generic Hub).
   * **6 PCs**.
3. Use straight-through cables to connect:
   * Each PC to the Hub port using straight-through cables.

**Assign IP Addresses to PCs**

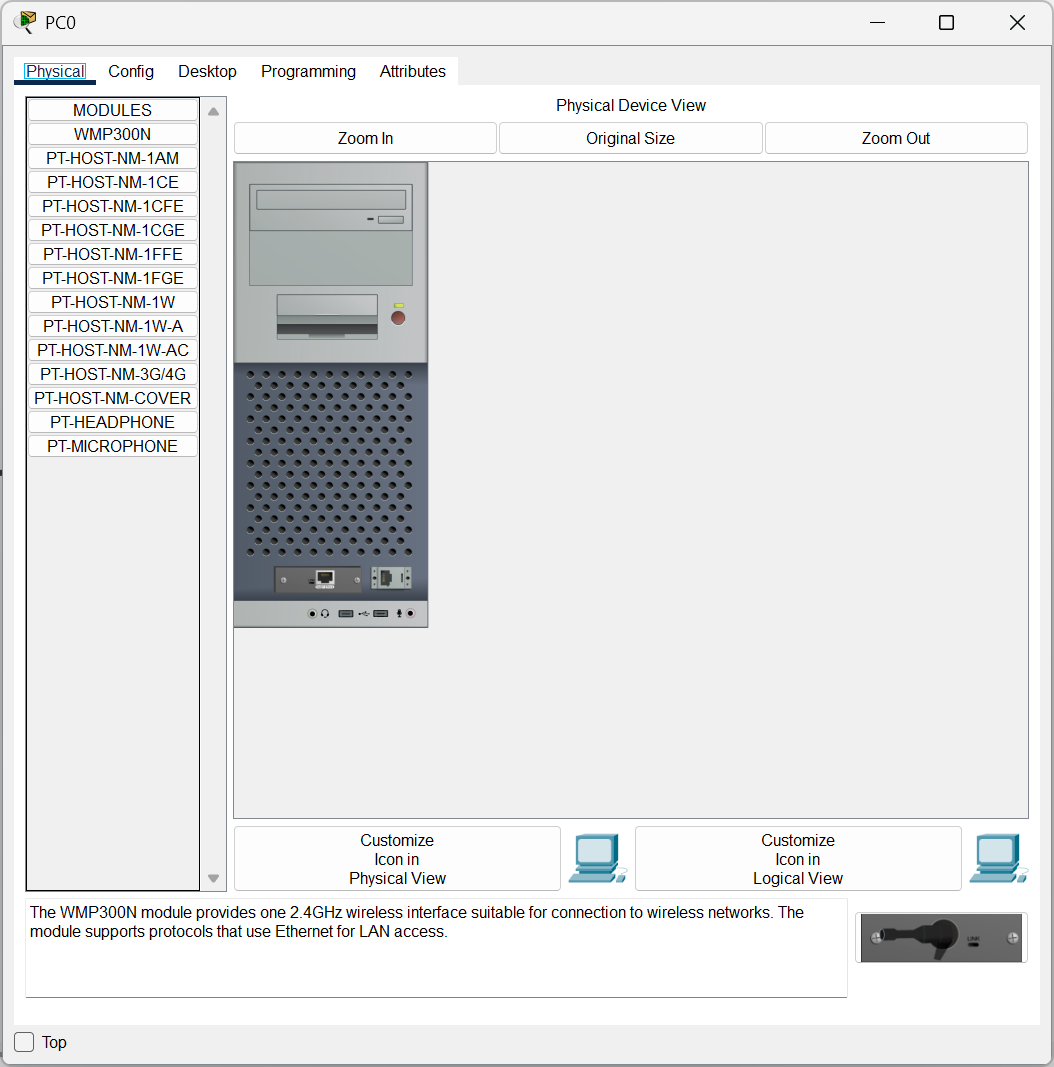
Each PC needs an IP address. Use the **IP Configuration tab** for each PC:

**Steps for Each PC:**

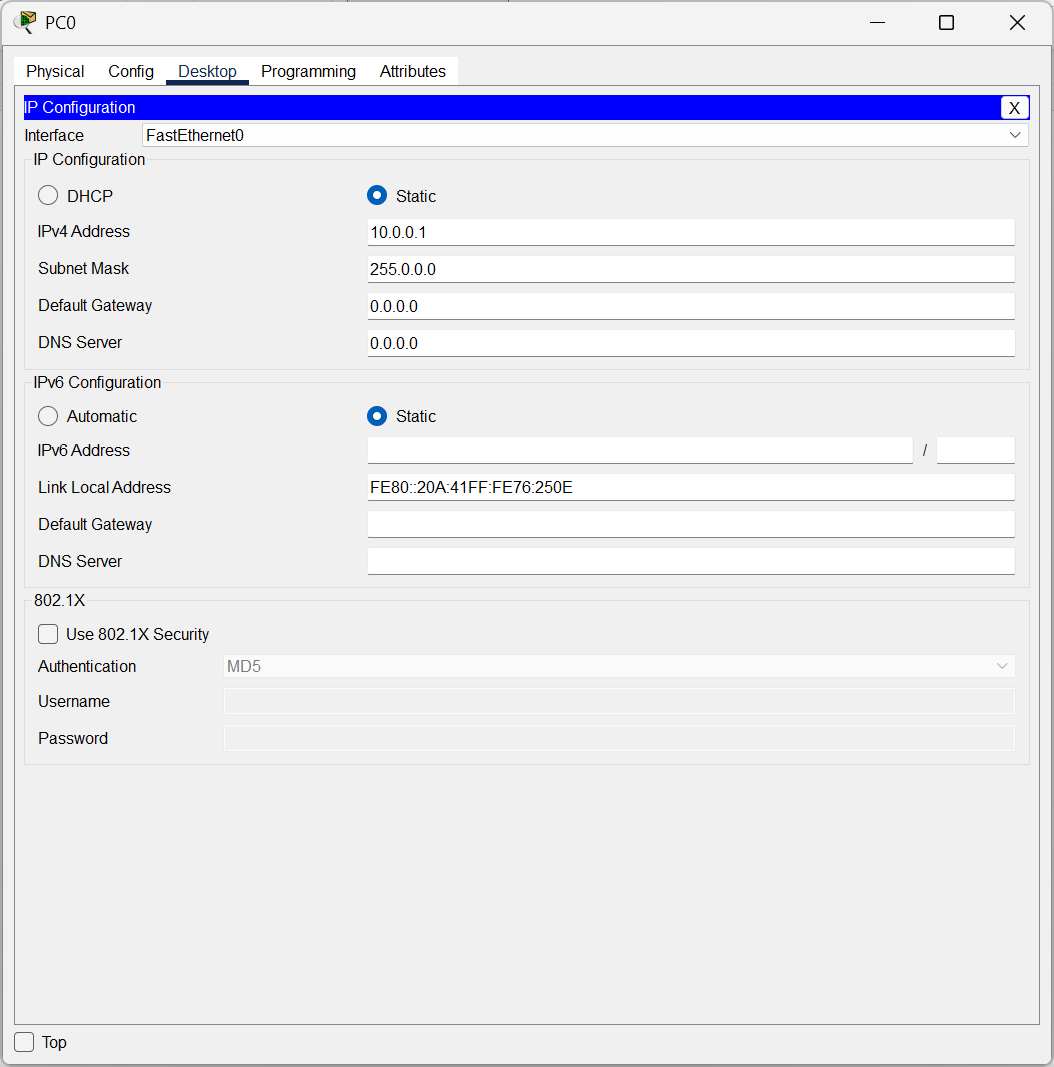
1. **Click on a PC** (e.g., PC0).
2. Navigate to **Desktop > IP Configuration**.
3. Set the following IP addresses:

|  |  |  |
| --- | --- | --- |
| **PC Name** | **IP Address** | **Default Gateway** |
| PC0 | 10.0.0.1 | 10.0.0.254 |
| PC1 | 10.0.0.2 | 10.0.0.254 |
| PC2 | 10.0.0.3 | 10.0.0.254 |
| PC3 | 10.0.0.4 | 10.0.0.254 |
| PC4 | 10.0.0.5 | 10.0.0.254 |
| PC5 | 10.0.0.6 | 10.0.0.254 |

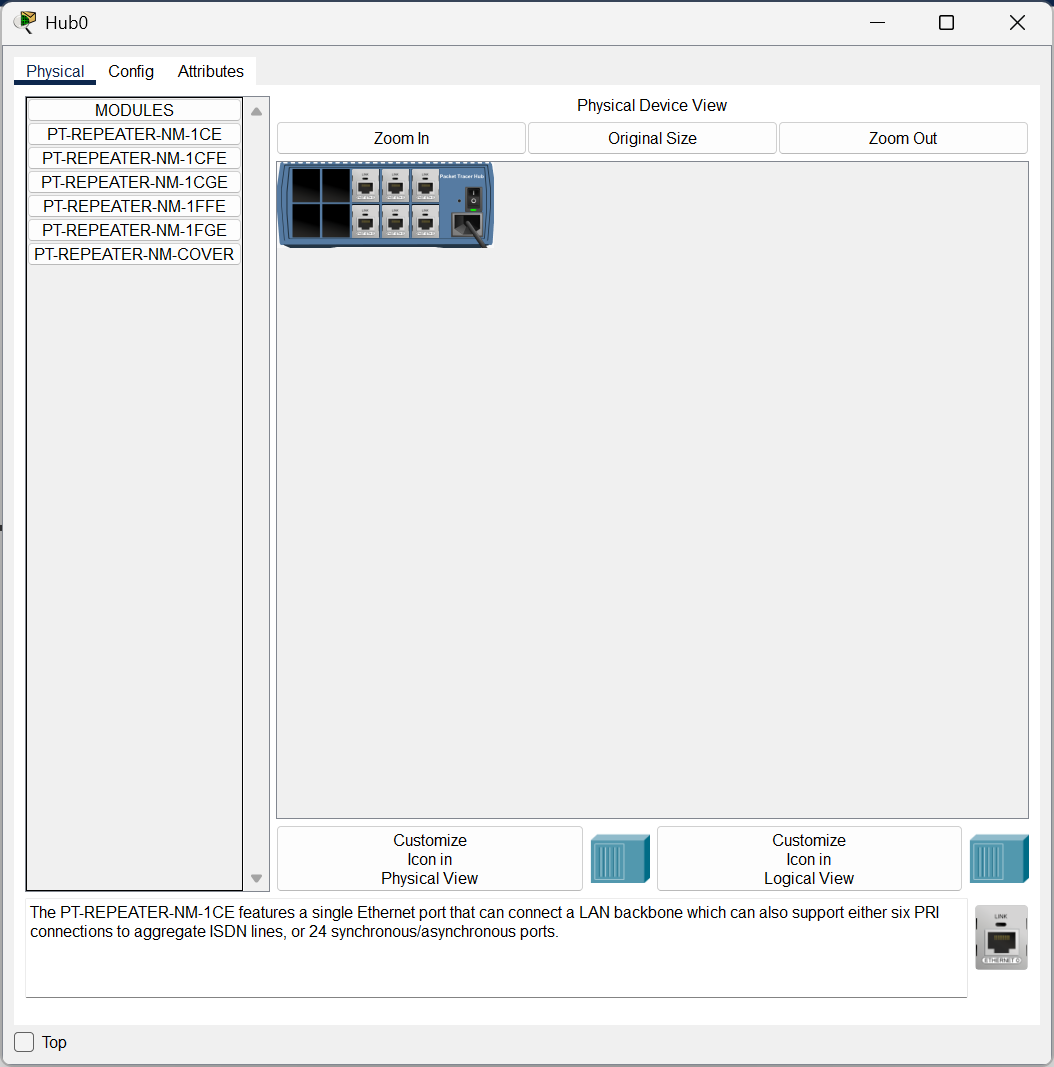
**PC Configuration Setup**



Configure PC0 with giving IP Address as **10.0.0.1**

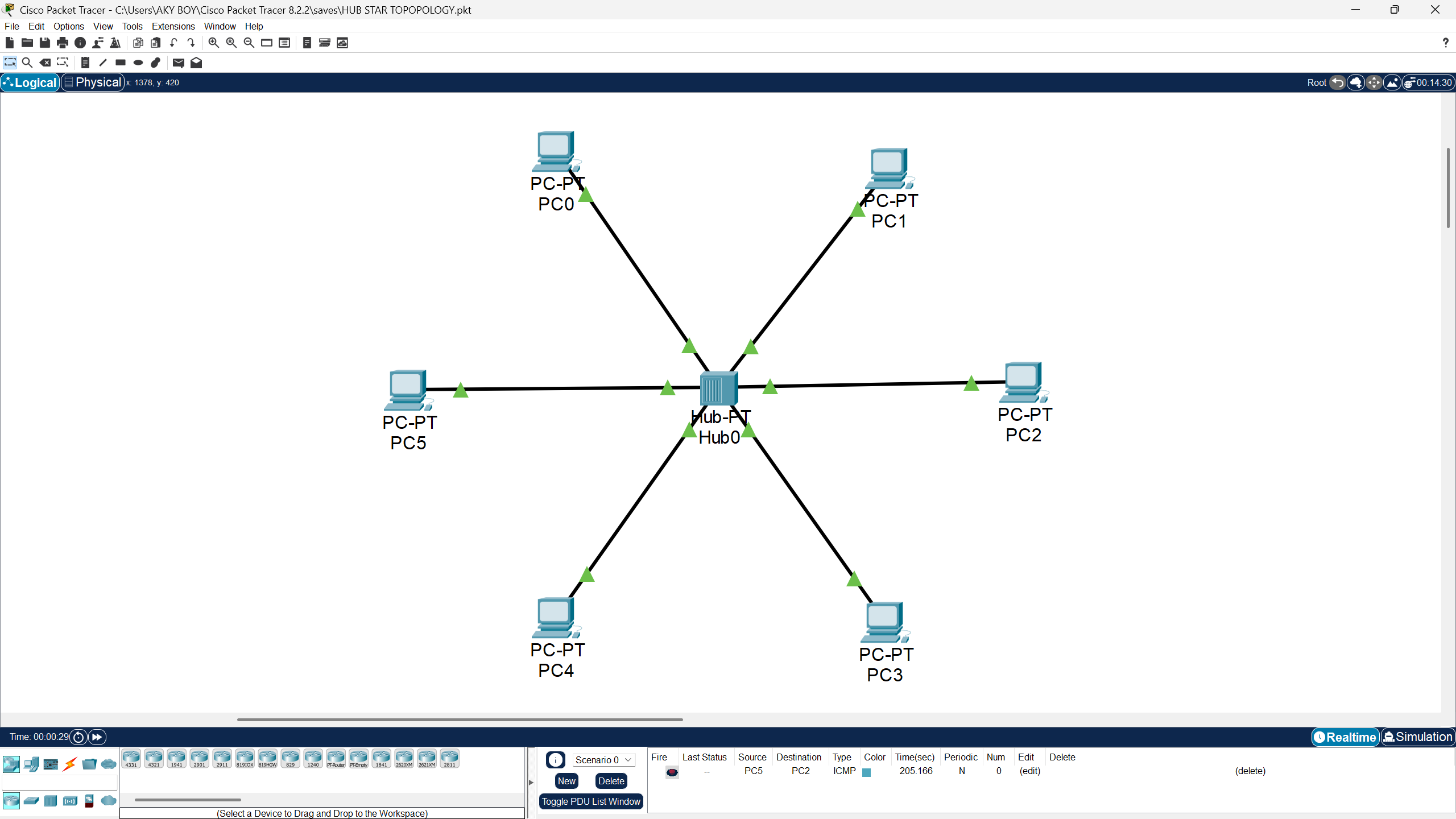


**Physical Representation of a Hub**



**CISCO PACKET TRACER DIGITAL REPRESENTATION**

***Representation of 6 PCs connected through Hub to each other following Star Topology.***



**Switch - STAR TOPOLOGY**

**Set Up the Devices**

1. Open **Cisco Packet Tracer**.
2. Drag and drop:
   * **1 Switch**.
   * **6 PCs**.
3. Use **straight-through cables** to connect:
   * Each PC to one of the switch ports (e.g., FastEthernet 0/1 to FastEthernet 0/6).

**Assign IP Addresses (Static Configuration)**

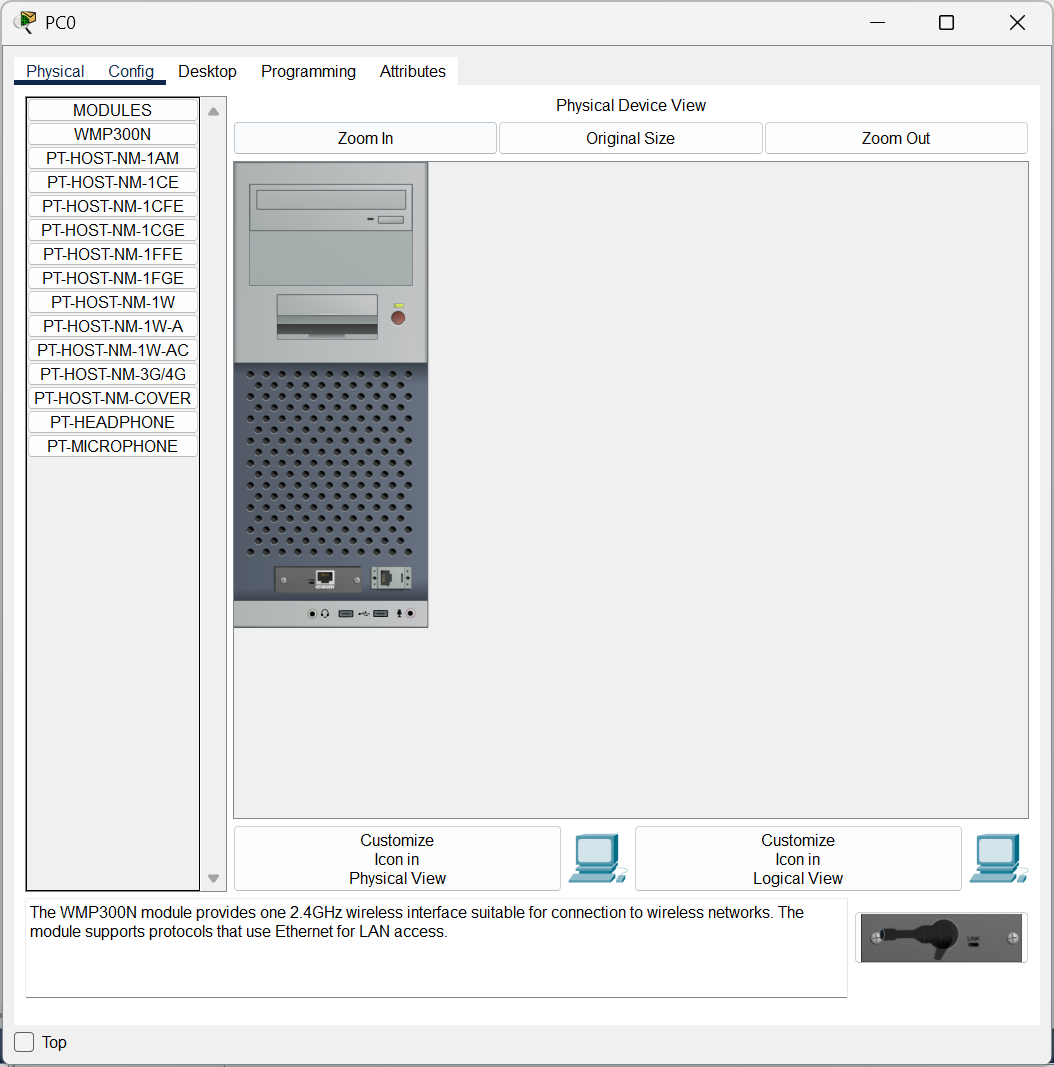
Manually assign IP addresses to each PC using the **IP Configuration tab**:

**Steps for Each PC:**

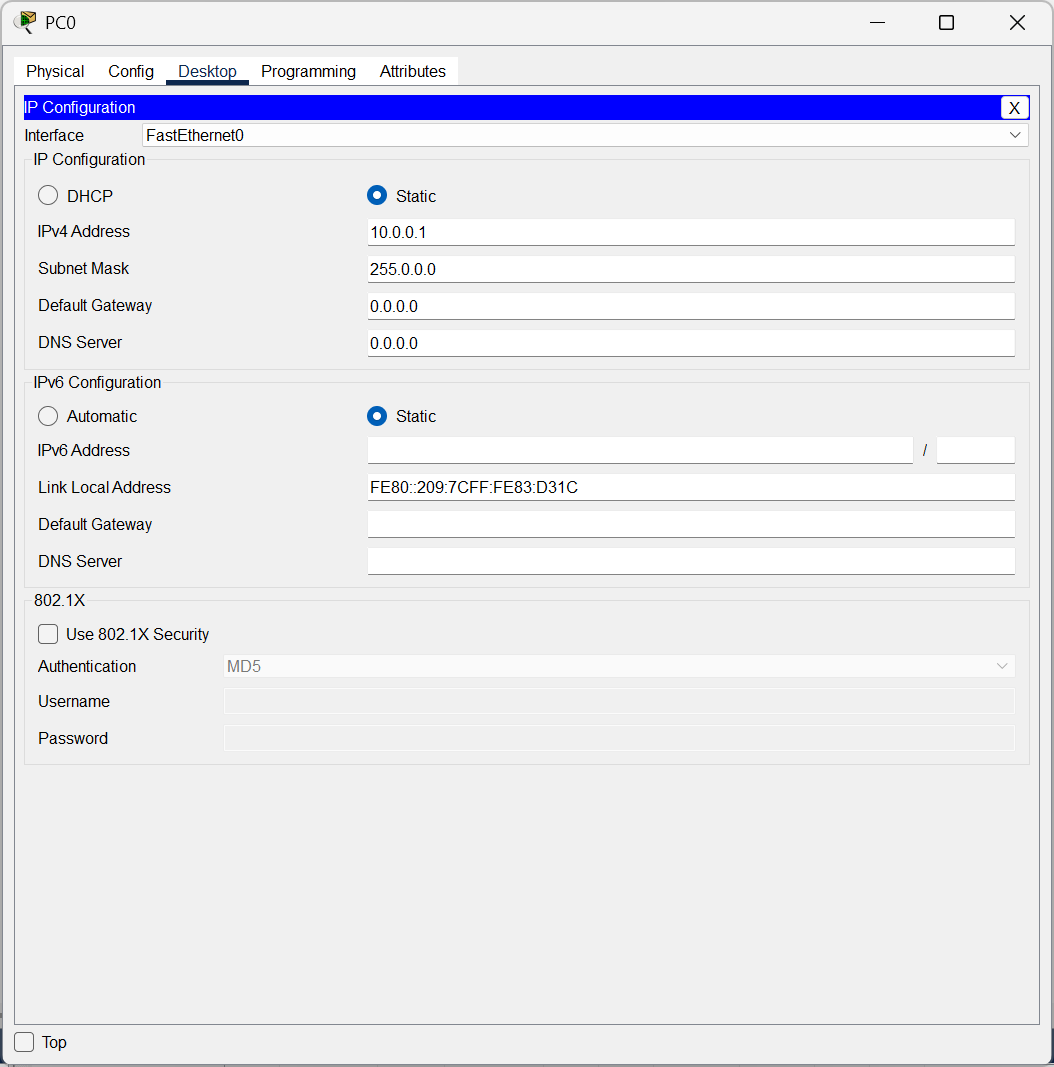
1. **Click on a PC** (e.g., PC0).
2. Navigate to **Desktop > IP Configuration**.
3. Assign the following settings:

|  |  |  |
| --- | --- | --- |
| **Device** | **IP Address** | **Default Gateway** |
| PC0 | 10.0.0.1 | 0.0.0.0 |
| PC1 | 10.0.0.2 | 0.0.0.0 |
| PC2 | 10.0.0.3 | 0.0.0.0 |
| PC3 | 10.0.0.4 | 0.0.0.0 |
| PC4 | 10.0.0.5 | 0.0.0.0 |
| PC5 | 10.0.0.6 | 0.0.0.0 |

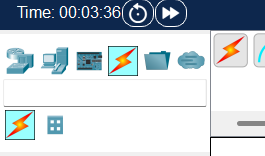
**PC Configuration Setup**



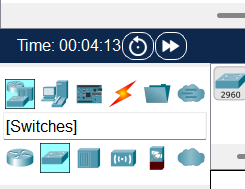
Configure PC0 with giving IP Address as **10.0.0.1**



Connect the Switch to the PCs by connecting switch with automatic cables.

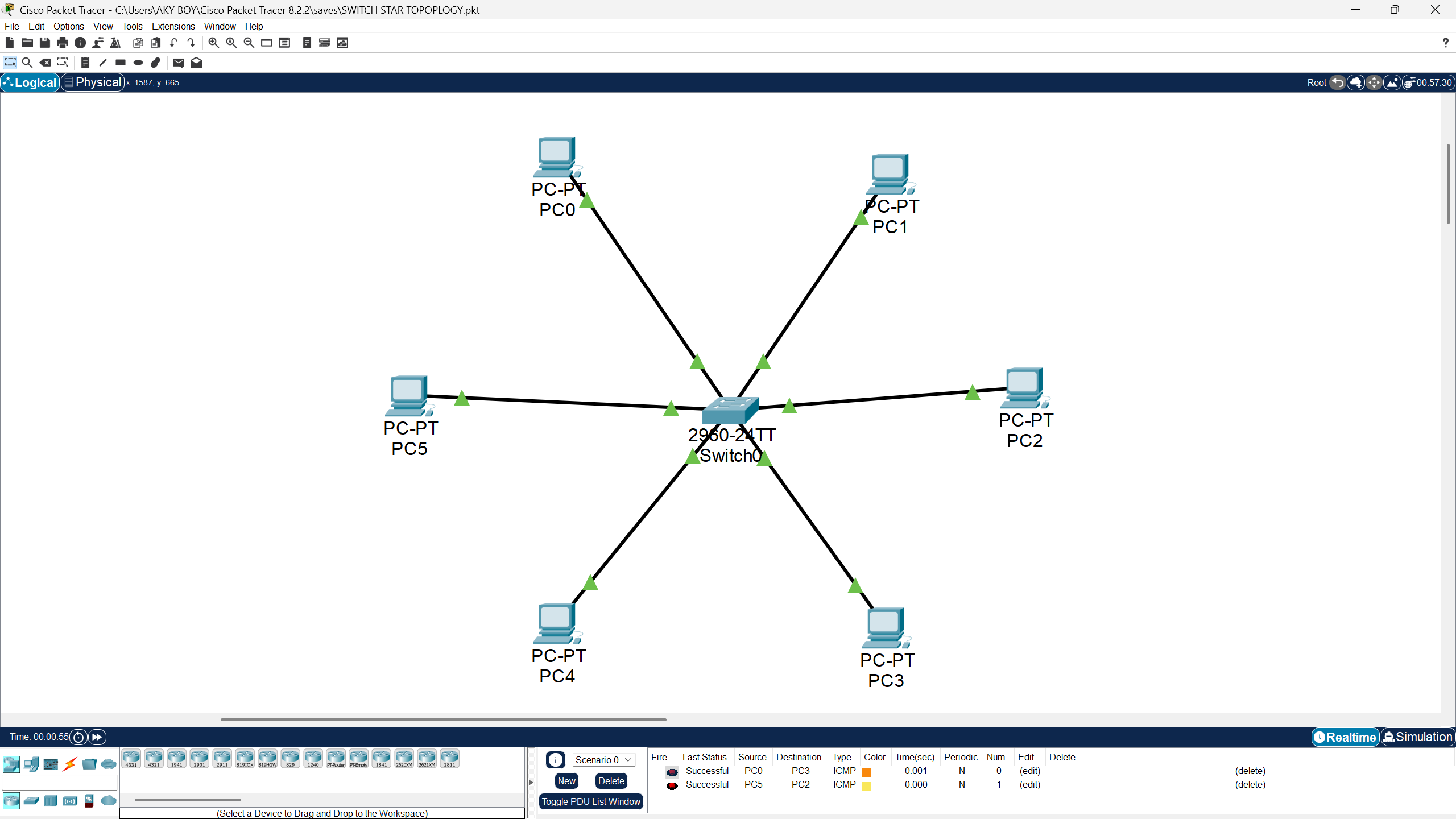


Drag and drop the Switch and connect them with PCs



**CISCO PACKET TRACER DIGITAL REPRESENTATION**

***Representation of 6 PCs connected through switches to each other following Star Topology.***



**LAB EXPERIMENT – 7**

**ROUTER - STAR TOPOLOGY**

**Set Up the Devices**

1. **Open Cisco Packet Tracer.**
2. Drag and drop the following devices:
   * **1 Router**
   * **2 Switches**
   * **4 PCs**
3. Connect the devices using the appropriate cables:
   * **Router to Switches**: Use a **crossover cable** or an **automatic cable** to connect the router's GigabitEthernet port to each switch.
   * **Switch to PCs**: Use a **straight-through cable** or an **automatic cable** to connect each PC to a switch.

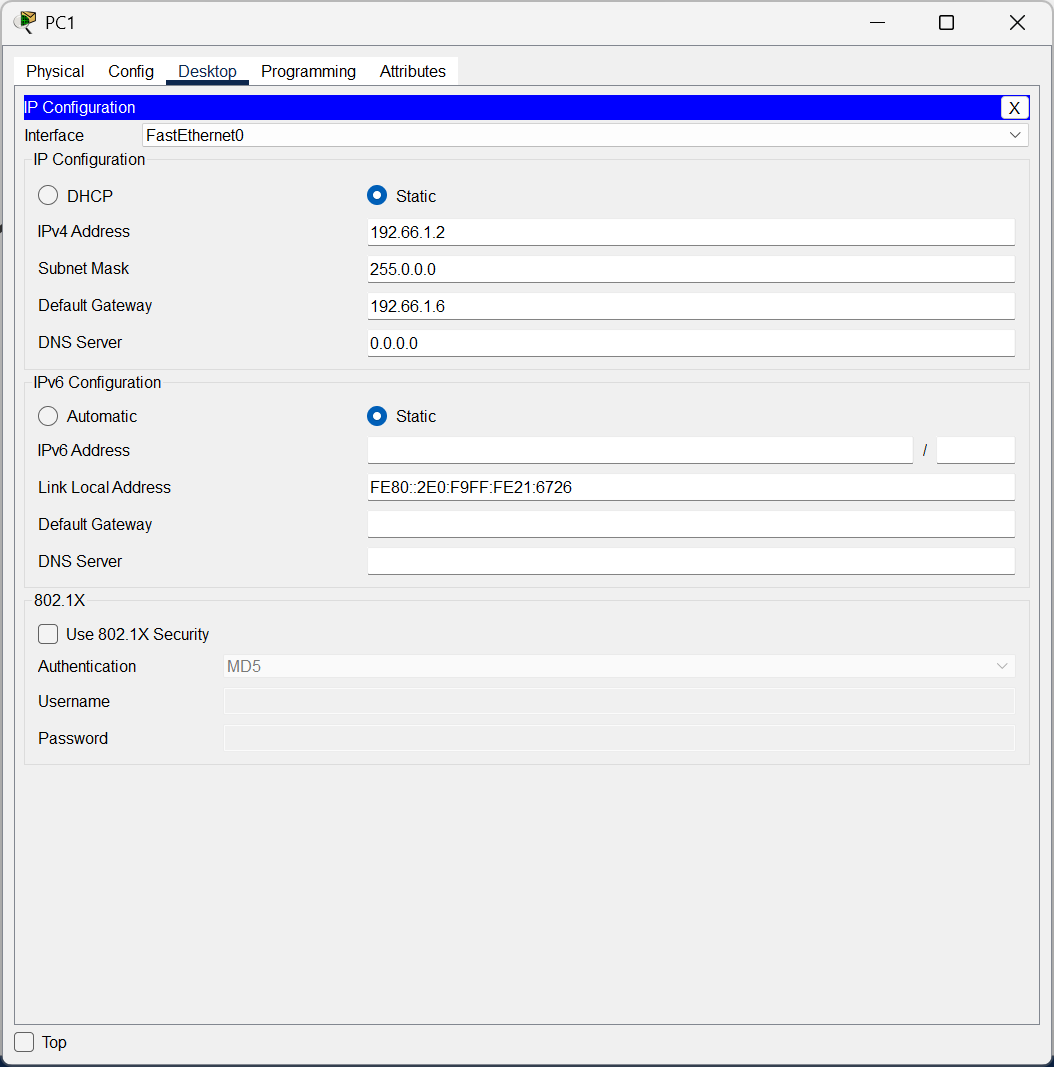
**Assign IP Addresses**

Assign IP addresses to the PCs in two subnets. Assume:

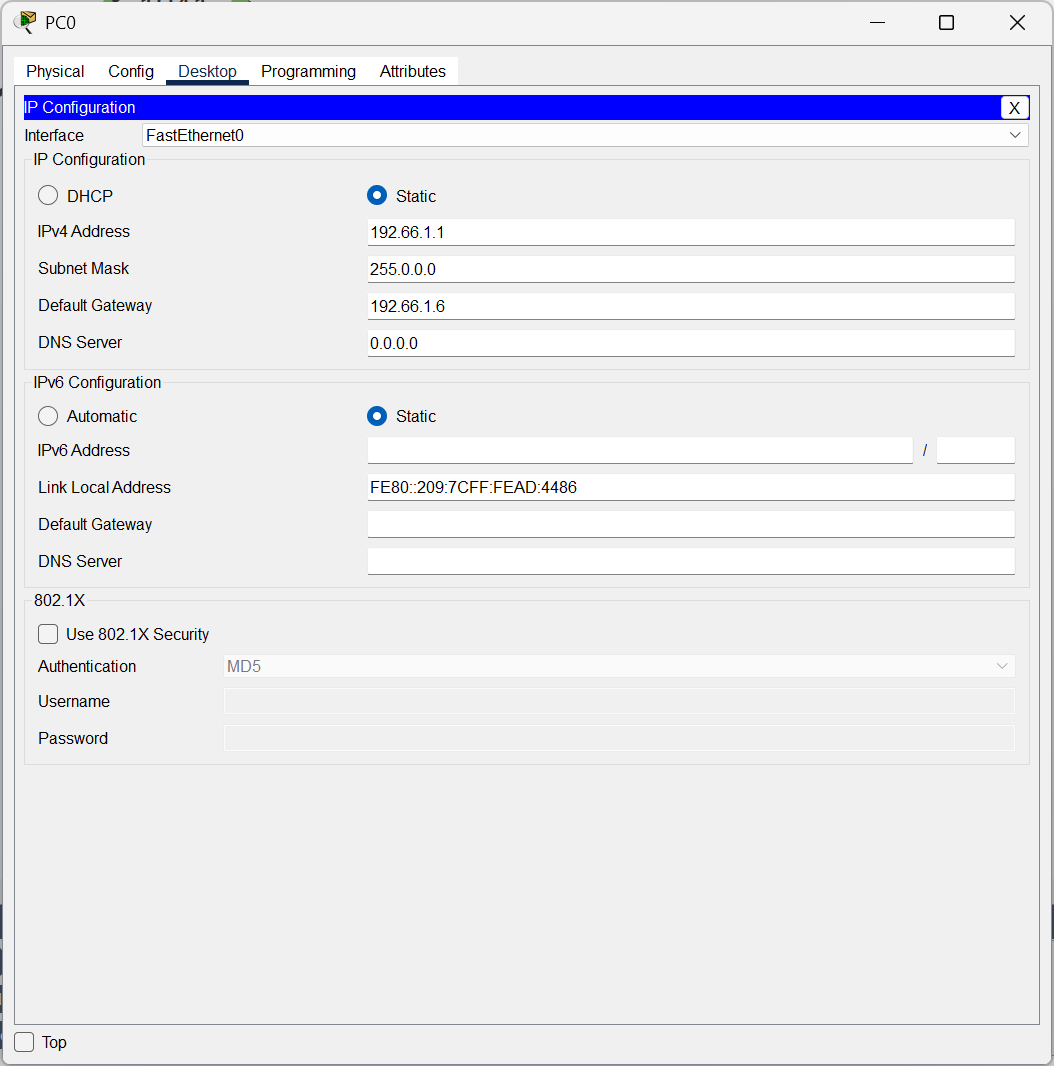
* **Subnet 1**: 192.66.1.0/24
* **Subnet 2**: 10.0.0/24

|  |  |  |
| --- | --- | --- |
| **Device** | **IP Address** | **Default Gateway** |
| PC0 (Switch 1) | 192.66.1.1 | 192.66.1.6 |
| PC1 (Switch 1) | 192.66.1.2 | 192.66.1.6 |
| PC2 (Switch 2) | 10.0.0.3 | 10.0.0.4 |
| PC3 (Switch 2) | 10.0.0.2 | 10.0.0.4 |

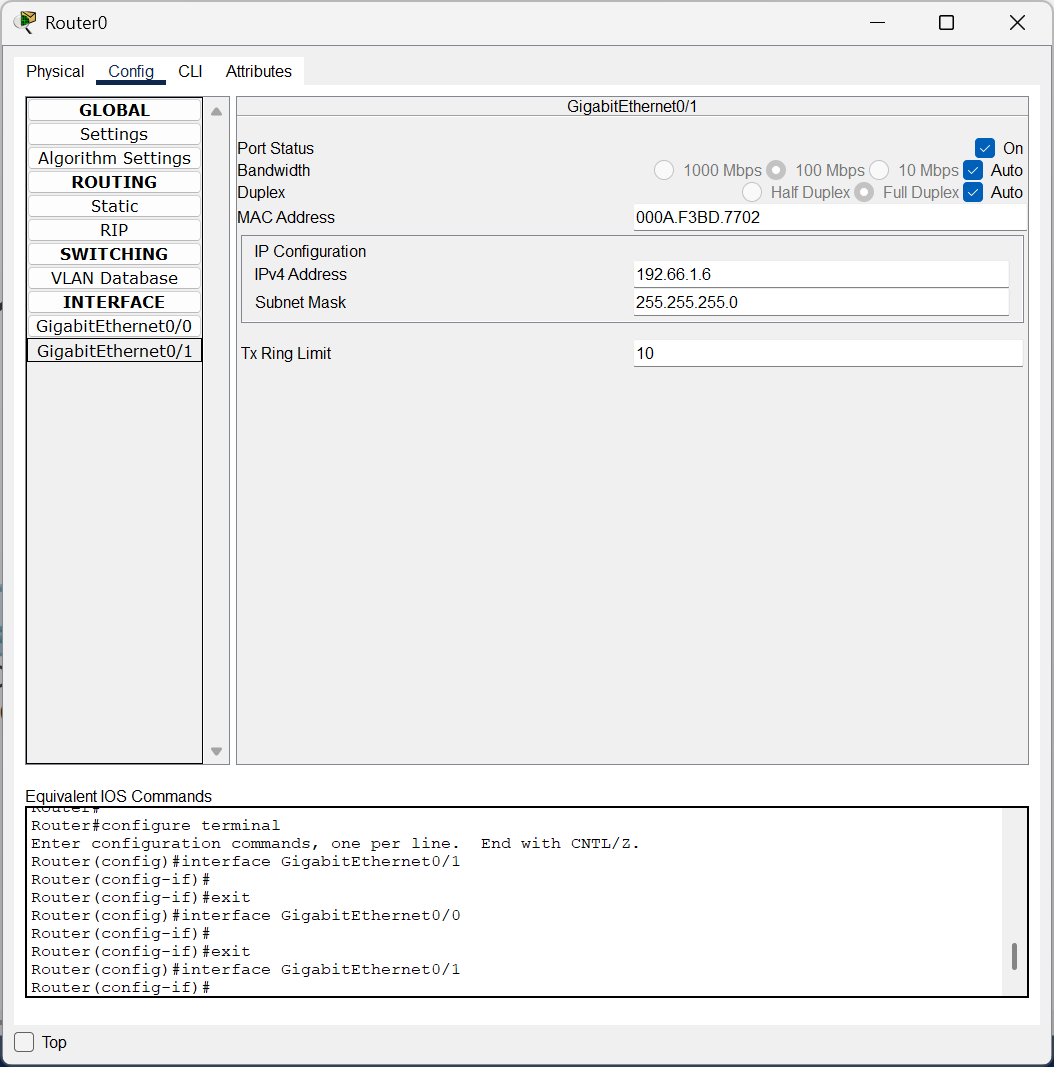
Configure PC1 with giving IP Address as **192.66.1.2**



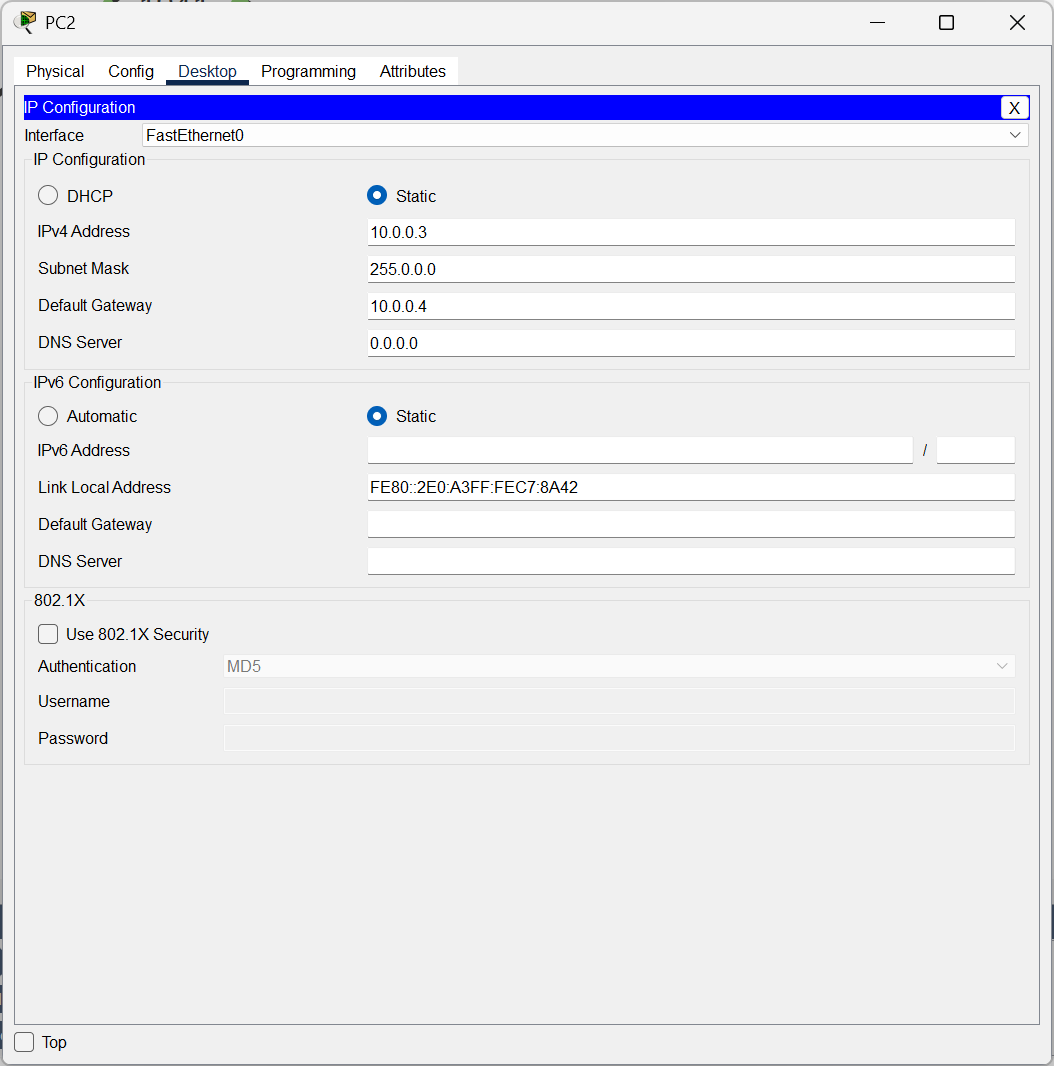
Configure PC0 with giving IP Address as **192.66.1.1**



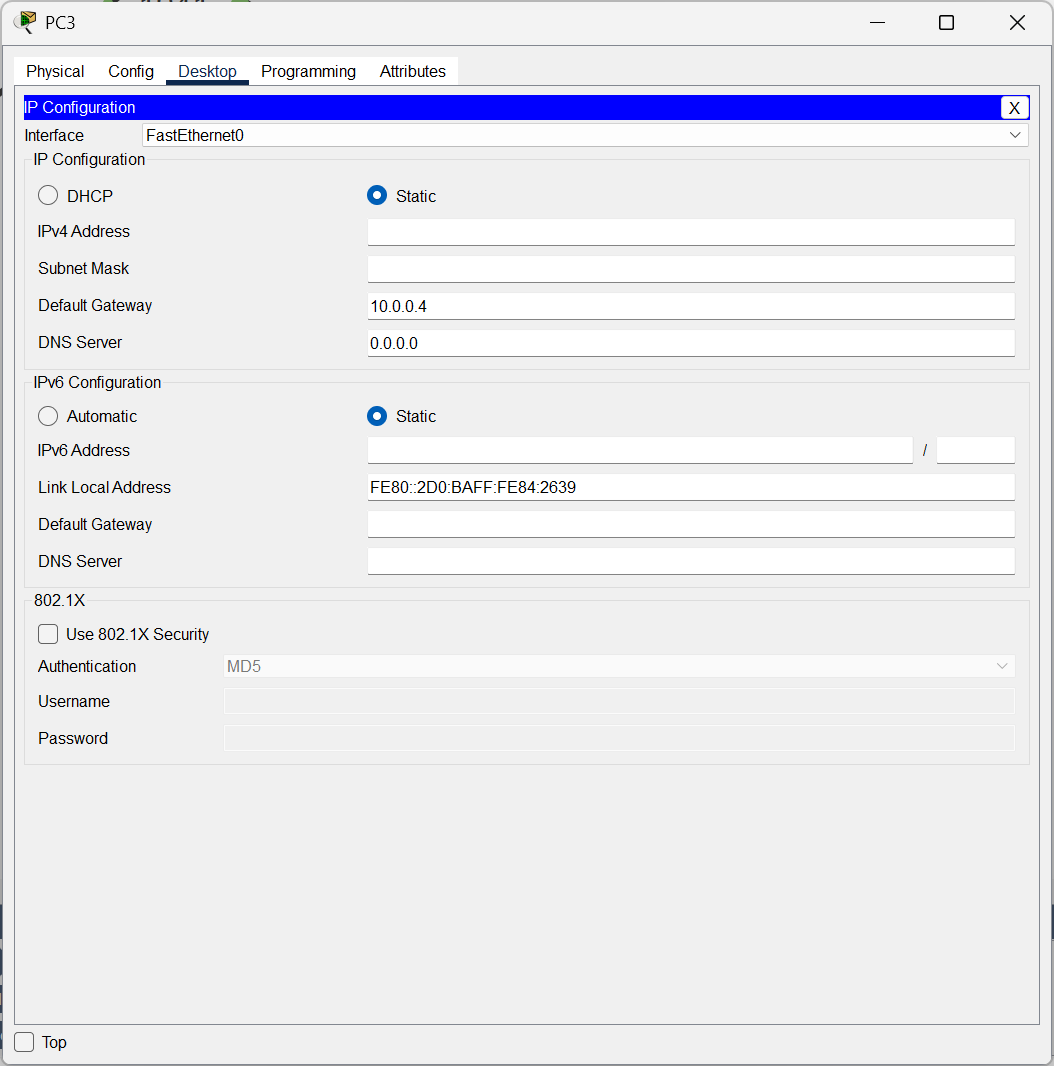
Configure Router0 with giving IPv4 Address as **192.66.1.6 for the network 1.**



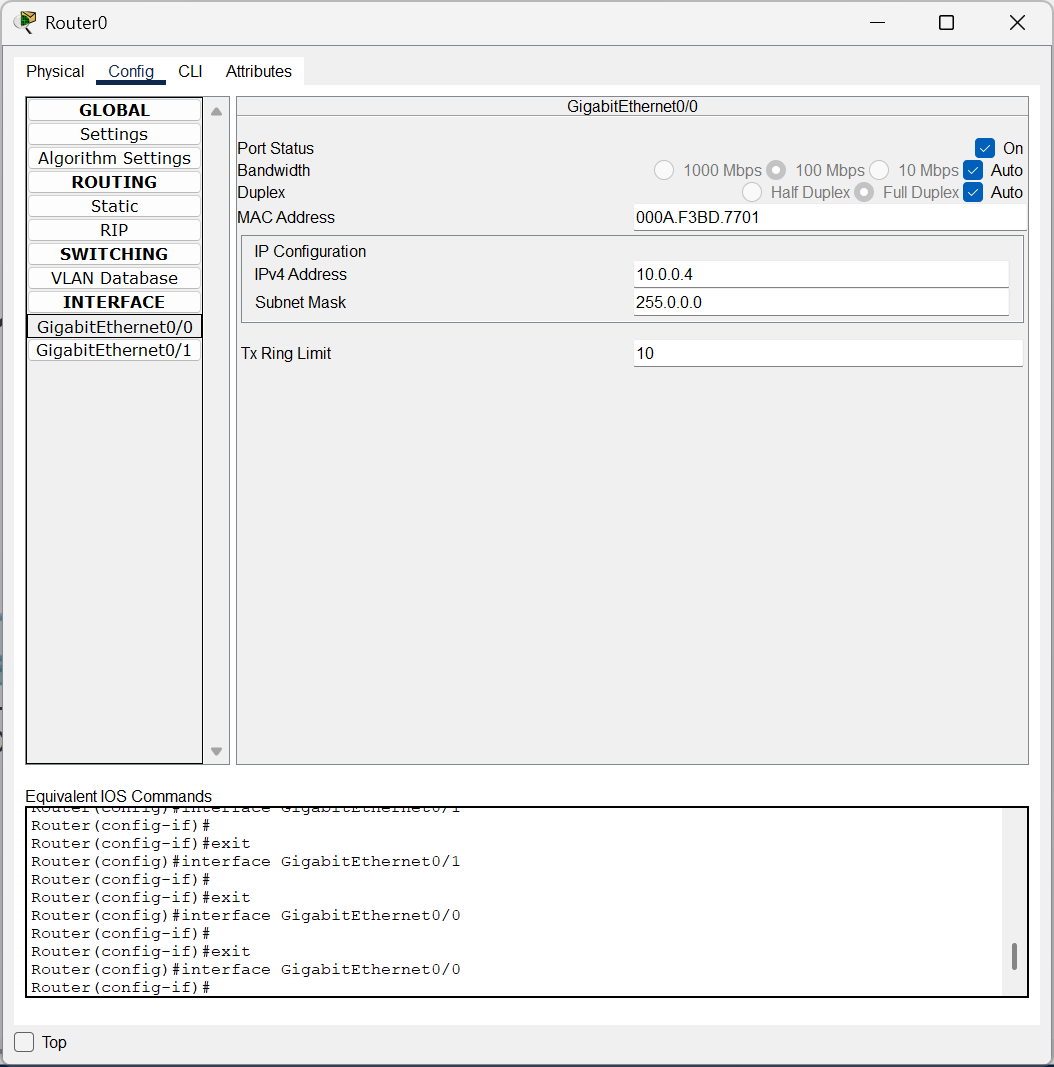
Configure PC2 with giving IP Address as **10.0.0.3**



Configure PC3 with giving IP Address as **10.0.0.2**

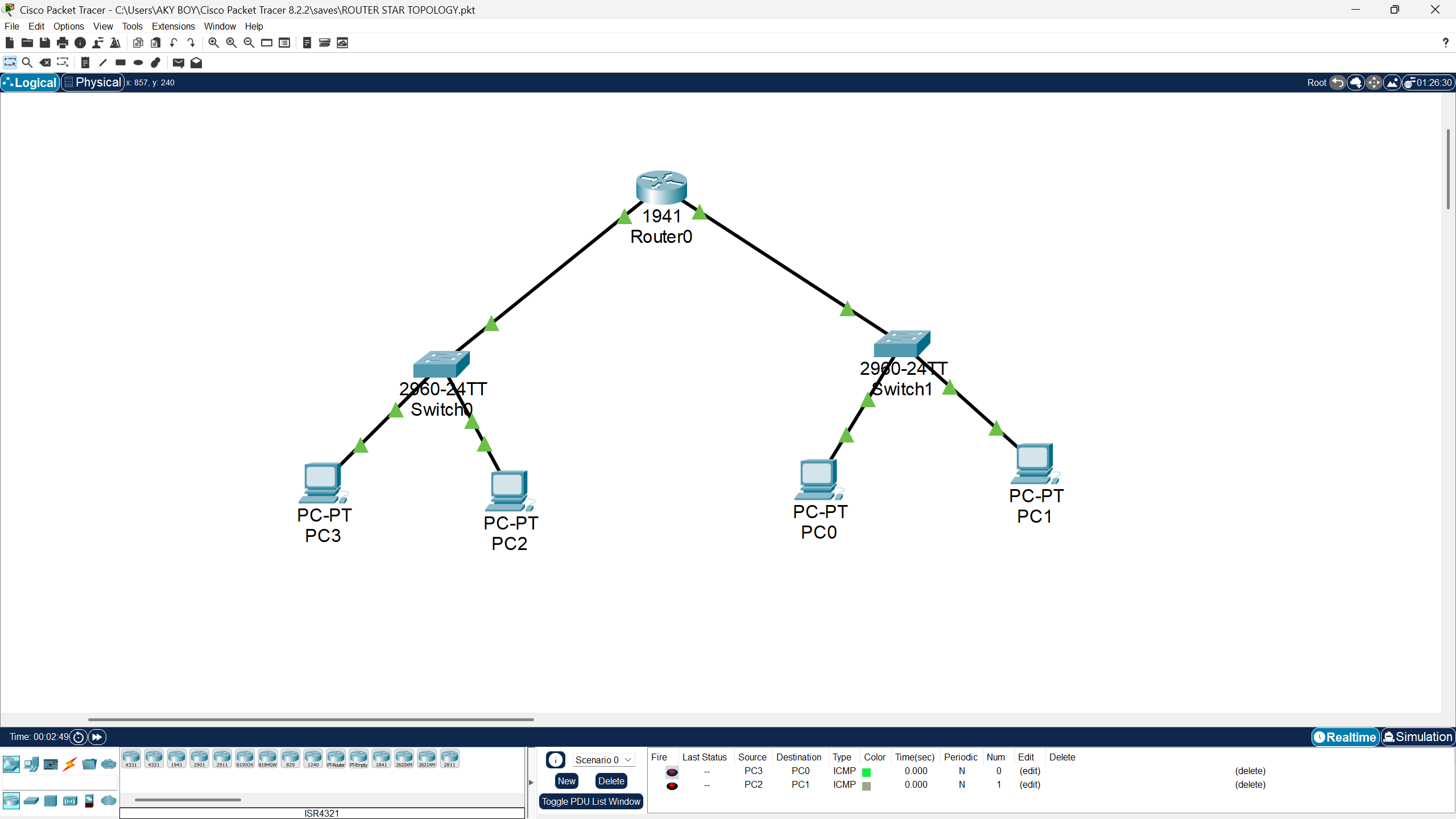


Configure Router0 with giving IPv4 Address as **10.0.0.4 for the network 0.**



**CISCO PACKET TRACER DIGITAL REPRESENTATION**

***Representation of 2 different networks connected through switches to a Router.***



**LAB EXPERIMENT – 8**

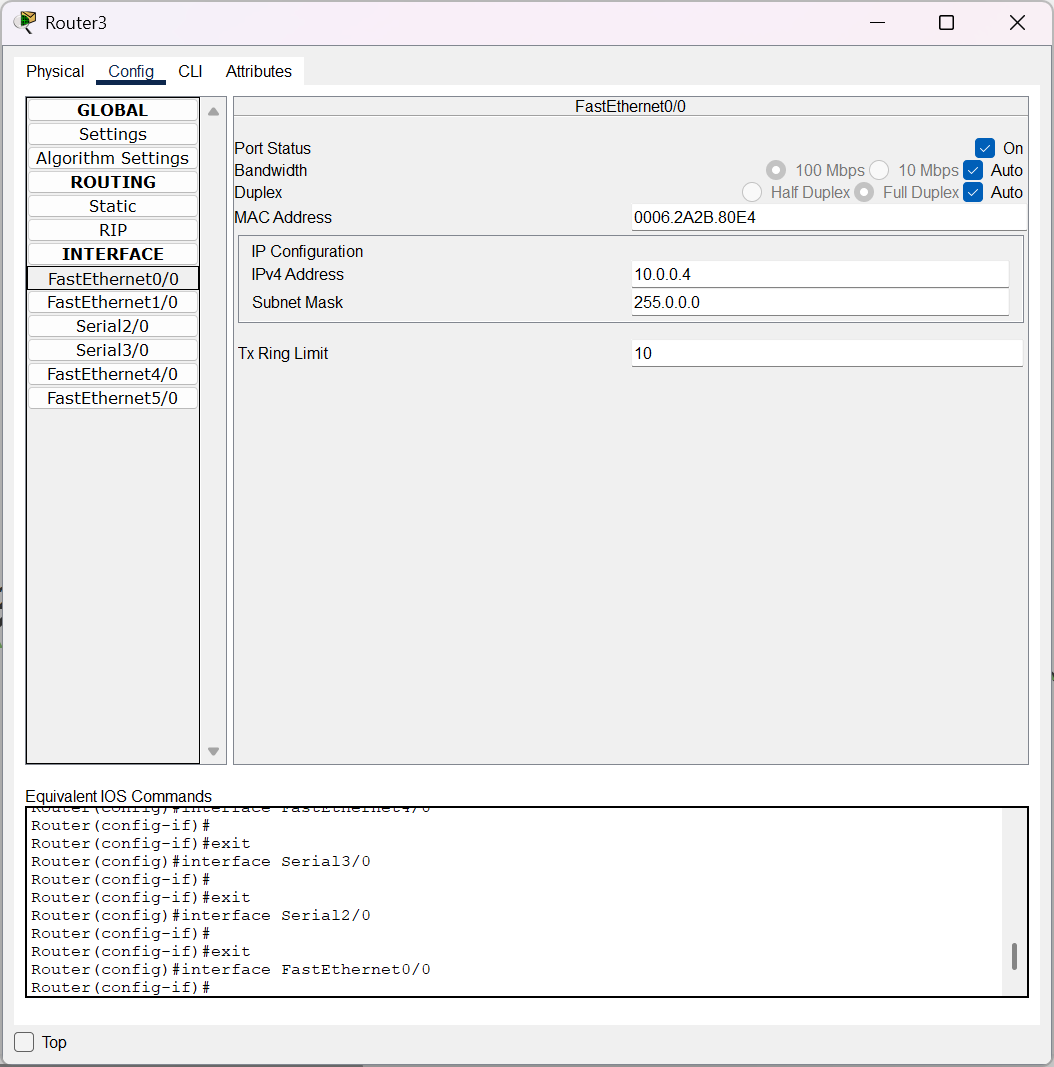
**Distance Vector Routing Simulation Using Cisco Packet Tracer – Static Routing**

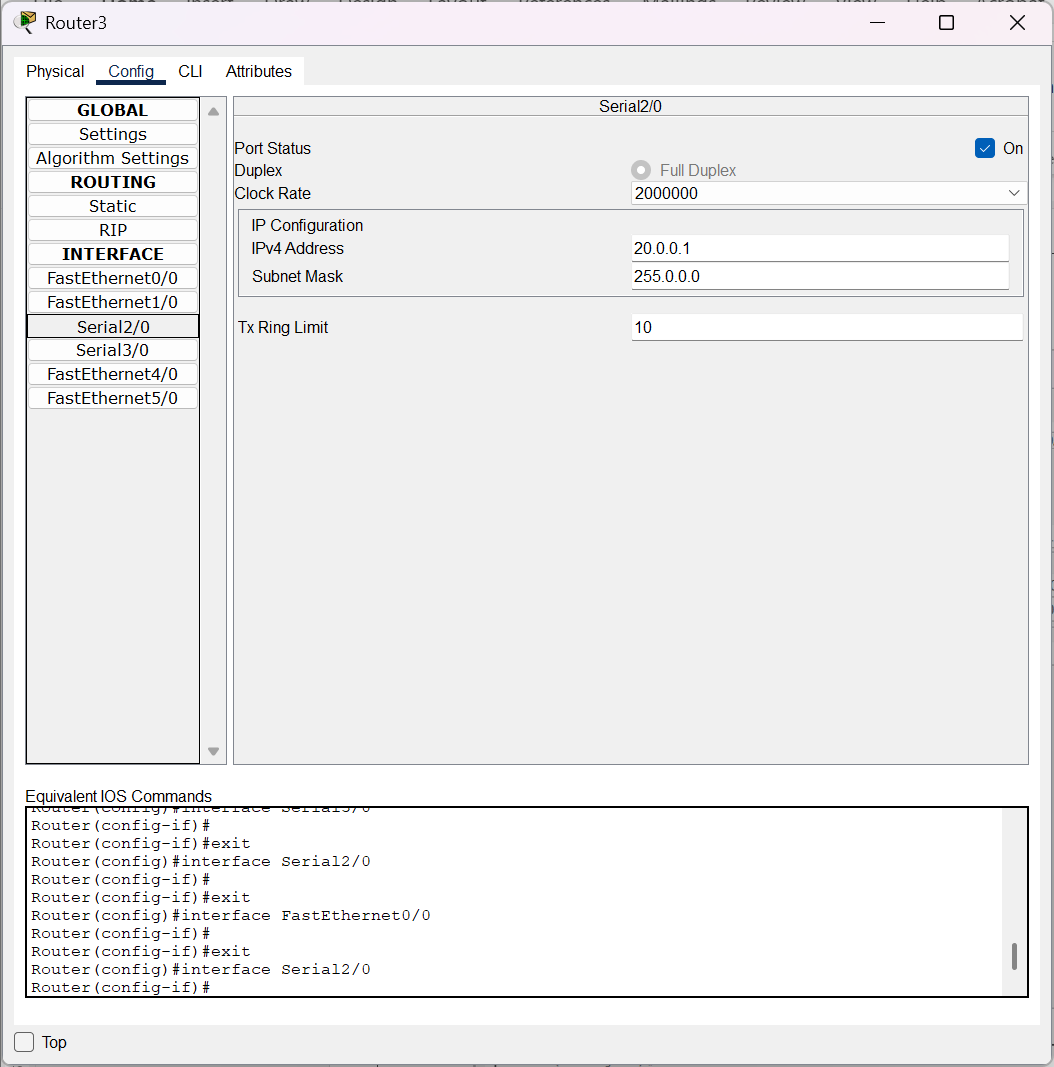
**Objective**

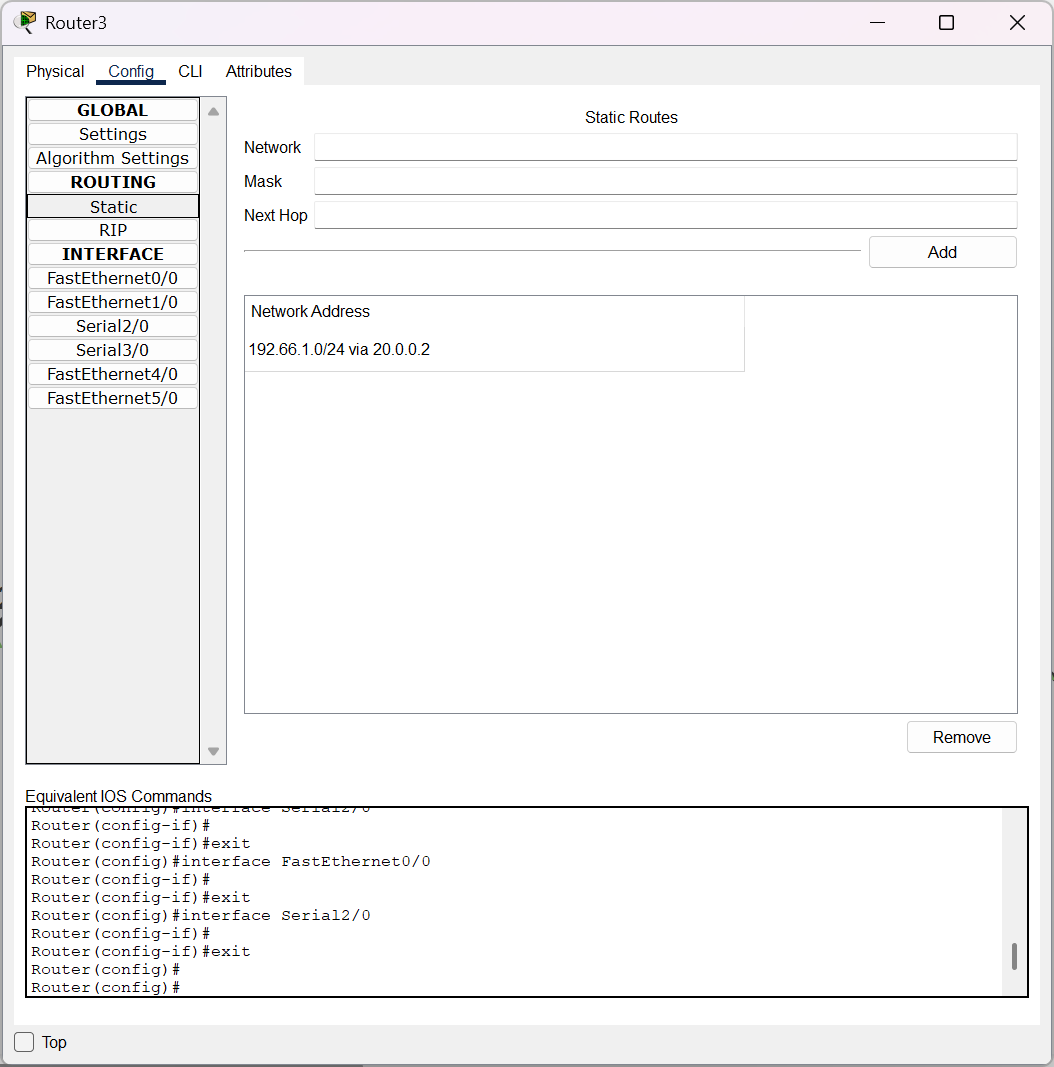
Simulate and verify the Distance Vector Routing protocol in a network using Static Routing tables in Cisco Packet Tracer. The focus is on creating a network topology, configuring static routes, and observing how routers forward packets.

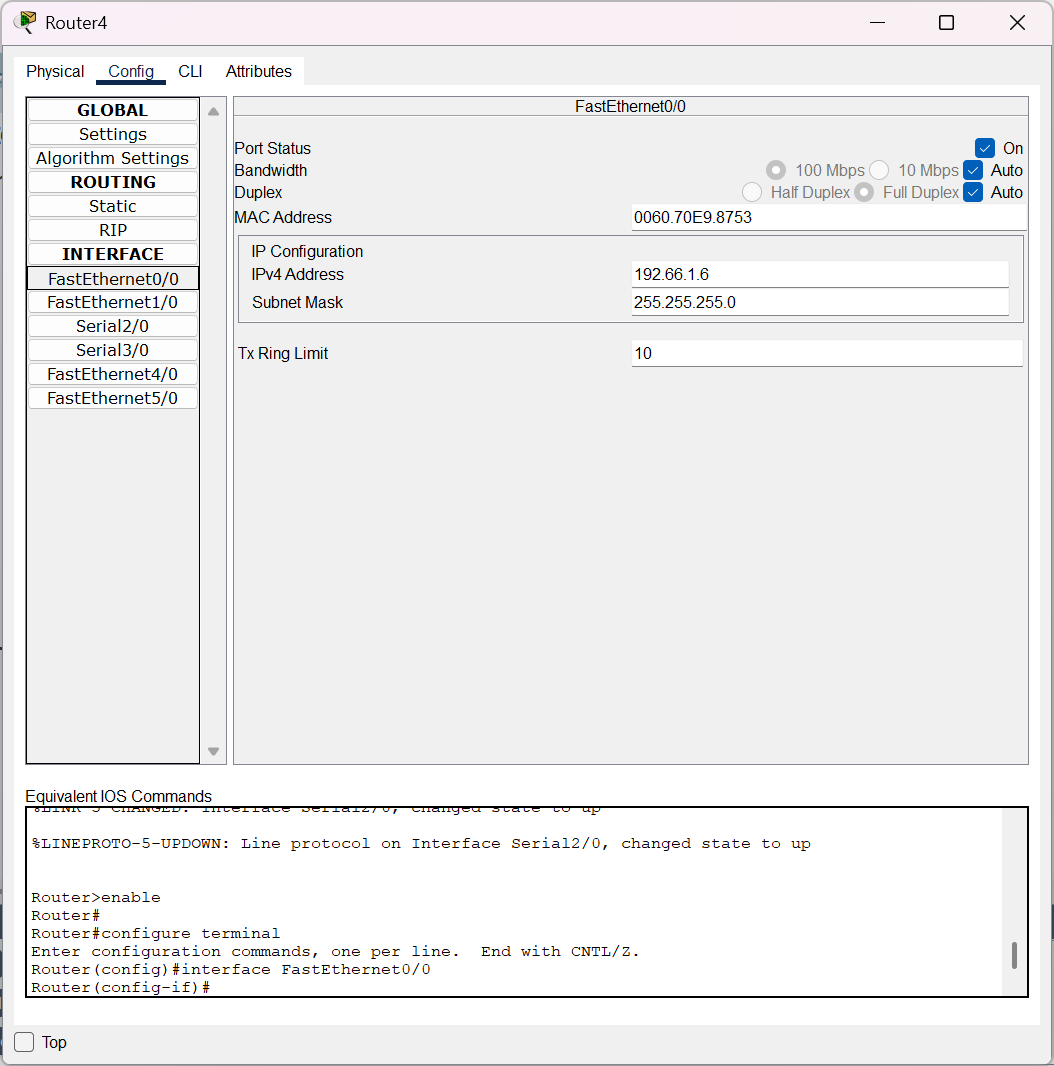
**Set Up the Devices**

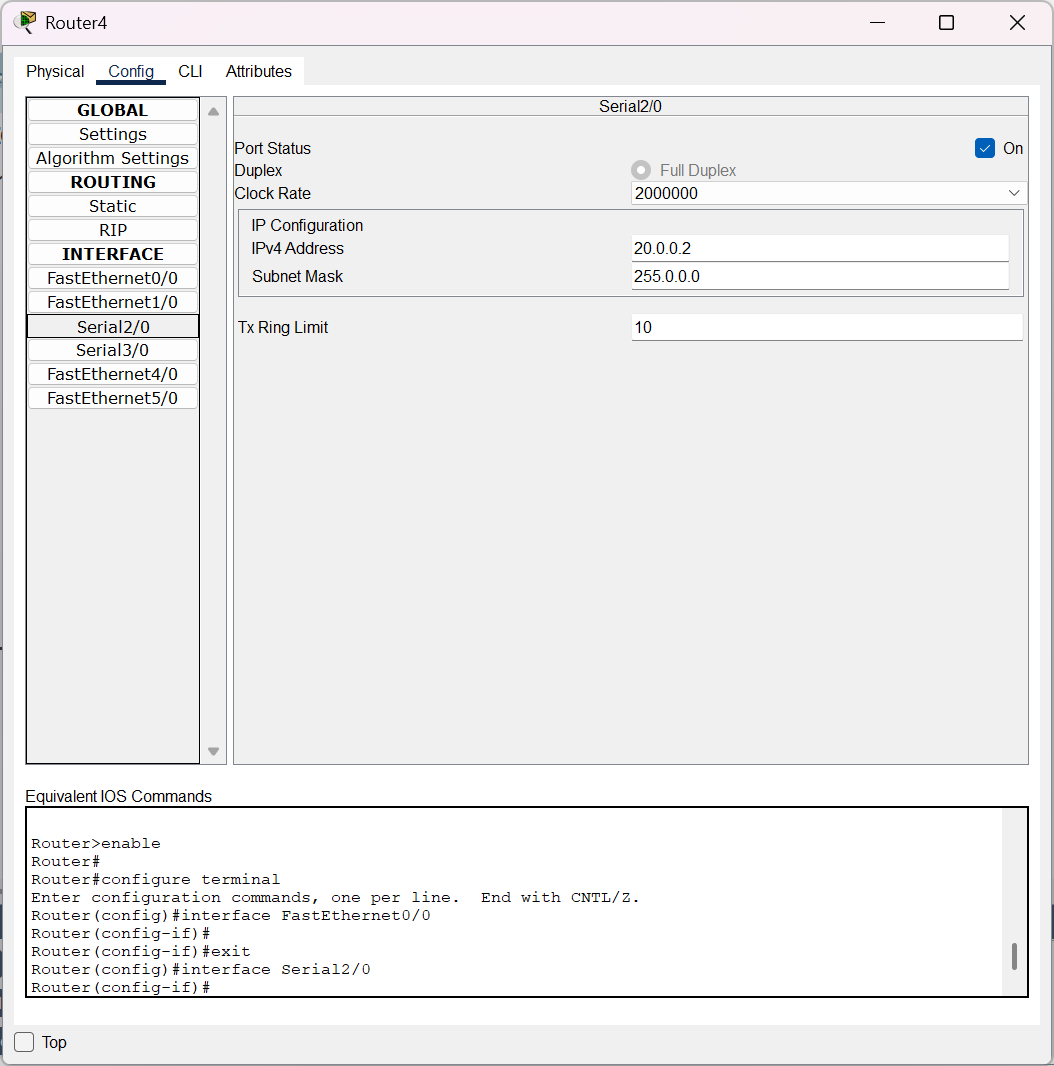
1. **Open Cisco Packet Tracer.**
2. Drag and drop the following devices:
   * **2 Router**
   * **2 Switches**
   * **4 PCs**
3. Connect the devices using the appropriate cables:
   * **Router to Switches**: Use a **crossover cable** or an **automatic cable** to connect the router's GigabitEthernet port to each switch.
   * **Switch to PCs**: Use a **straight-through cable** or an **automatic cable** to connect each PC to a switch.

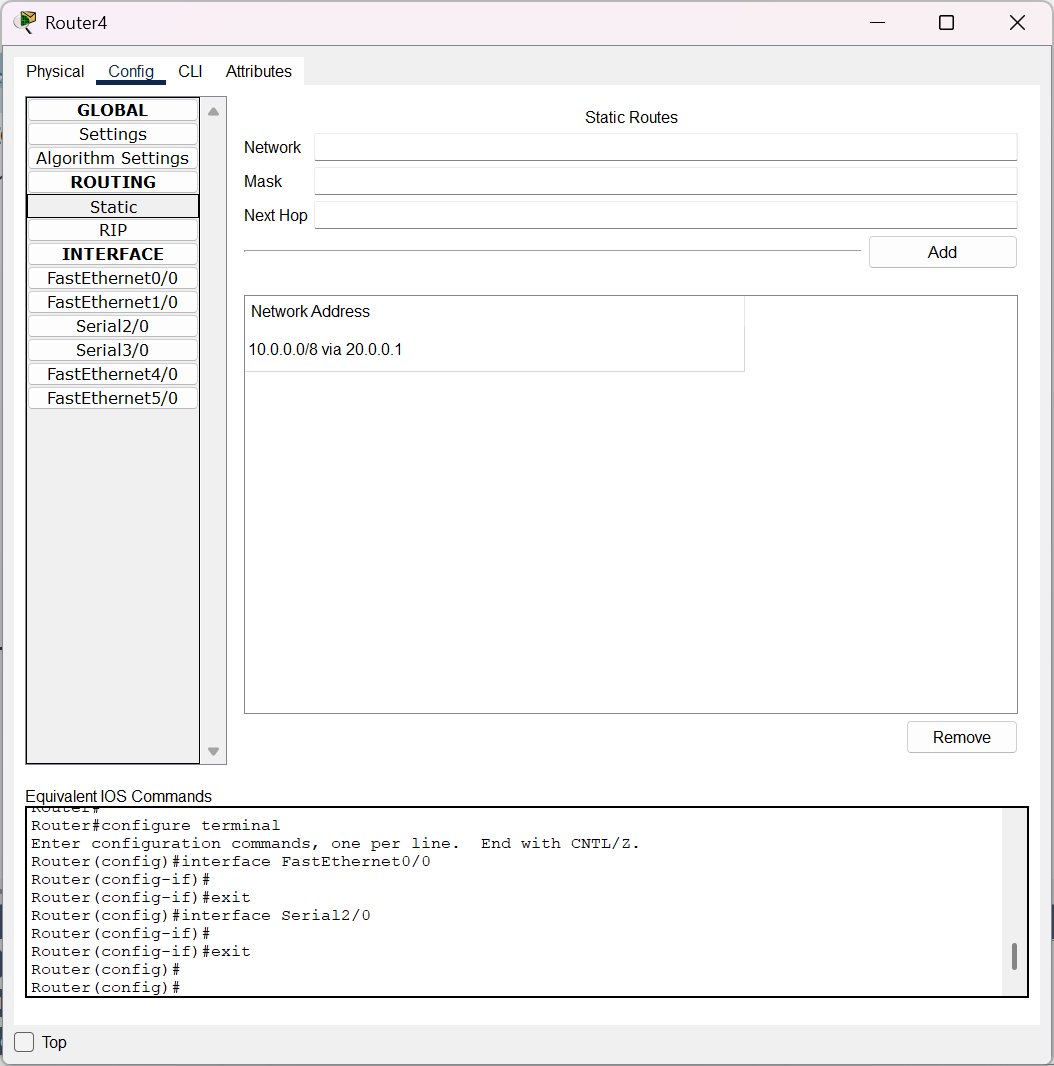


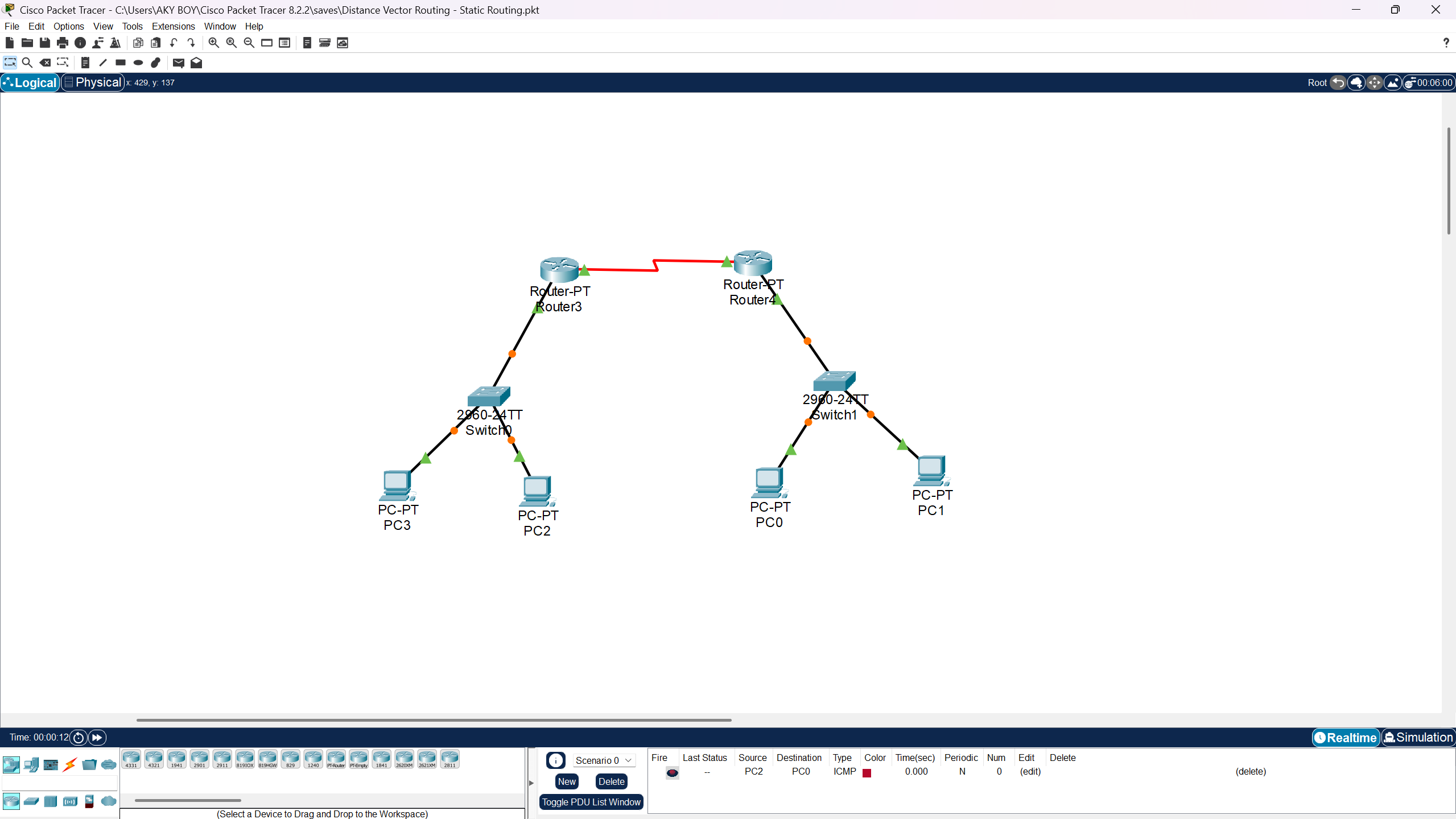










****

**LAB EXPERIMENT – 9  
Distance Vector Routing Simulation Using Cisco Packet Tracer – RIP Table**

**Set Up the Devices**

1. **Open Cisco Packet Tracer.**
2. Drag and drop the following devices:
   * **1 Router**
   * **2 Switches**
   * **4 PCs**
3. Connect the devices using the appropriate cables:
   * **Router to Switches**: Use a **crossover cable** or an **automatic cable** to connect the router's GigabitEthernet port to each switch.
   * **Switch to PCs**: Use a **straight-through cable** or an **automatic cable** to connect each PC to a switch.

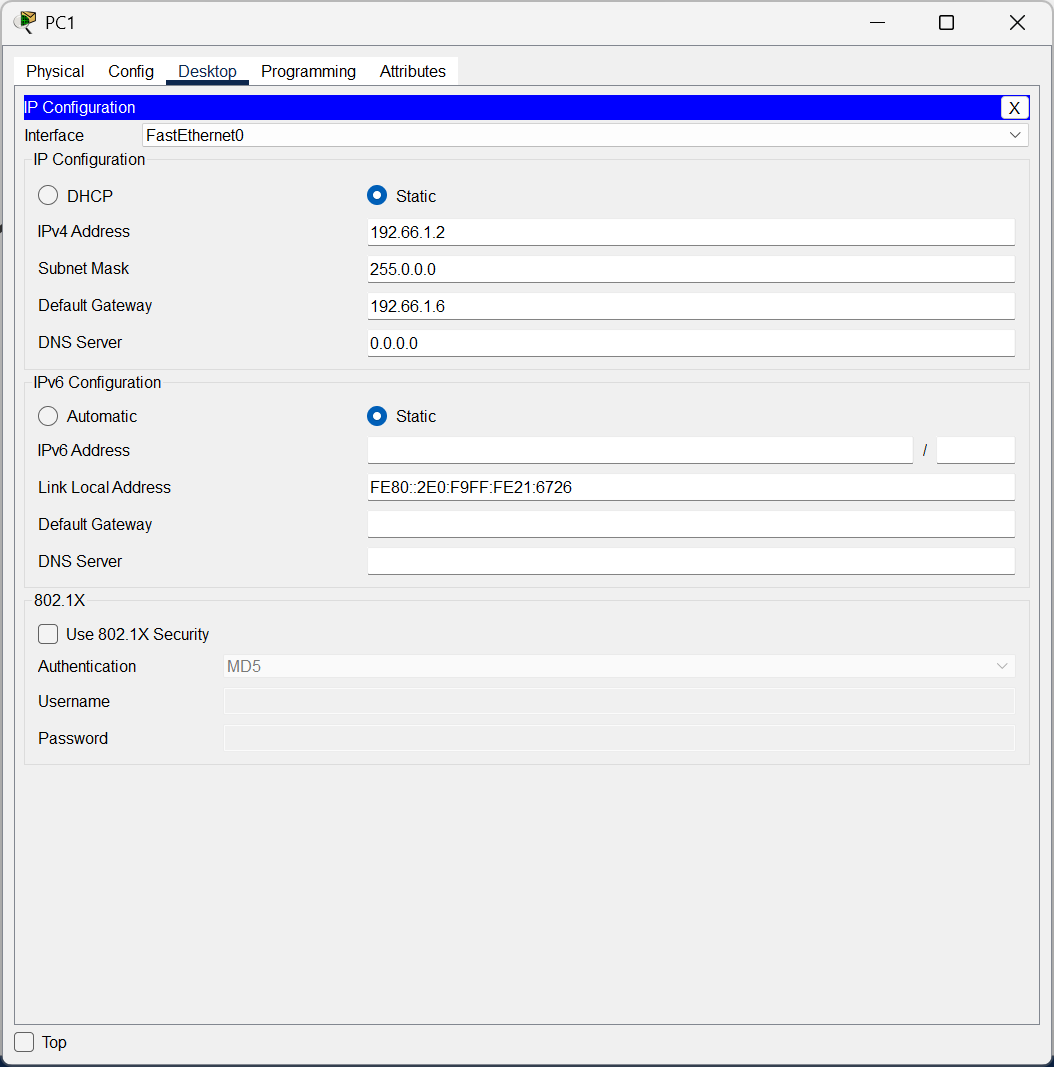
**Assign IP Addresses**

Assign IP addresses to the PCs in two subnets. Assume:

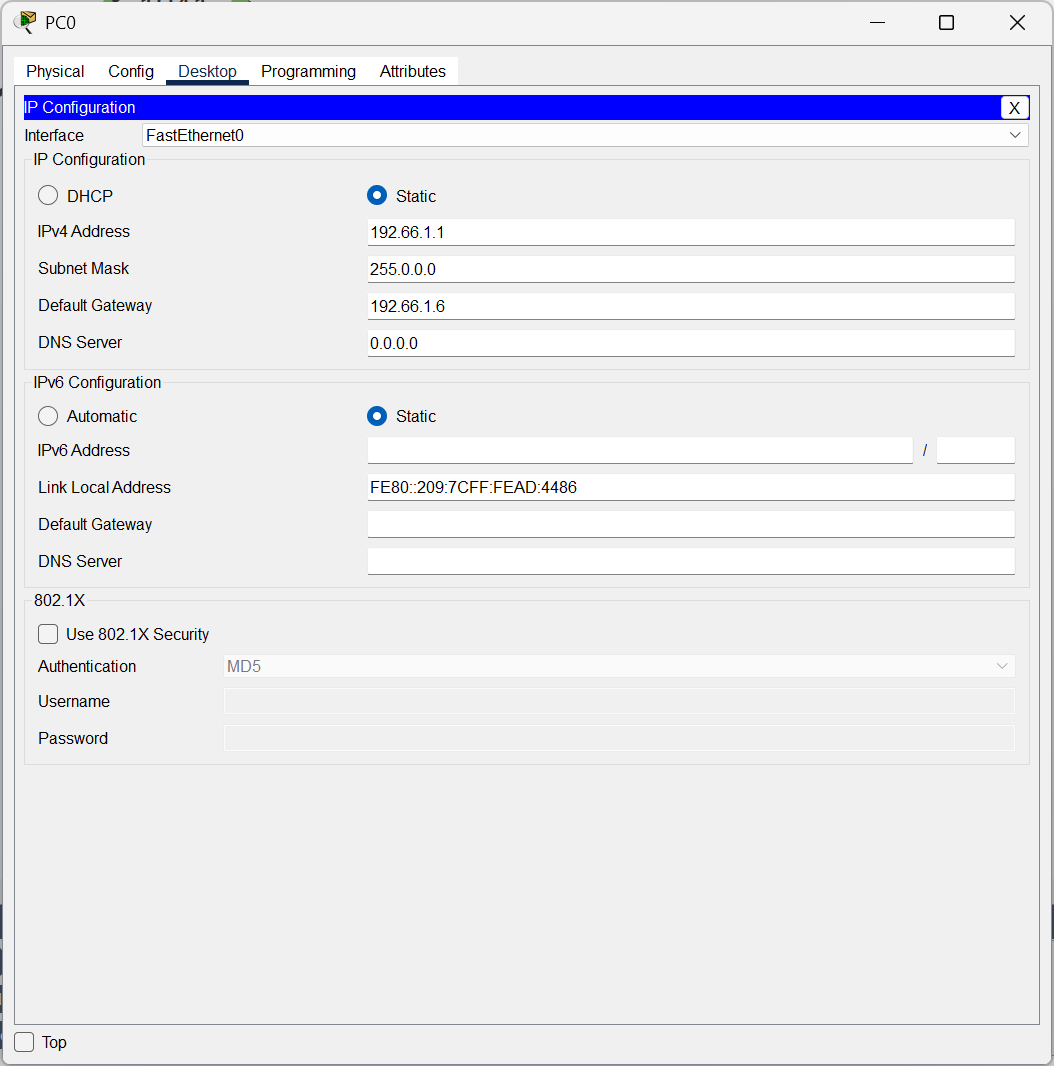
* **Subnet 1**: 192.66.1.0/24
* **Subnet 2**: 10.0.0/24

|  |  |  |
| --- | --- | --- |
| **Device** | **IP Address** | **Default Gateway** |
| PC0 (Switch 1) | 192.66.1.1 | 192.66.1.6 |
| PC1 (Switch 1) | 192.66.1.2 | 192.66.1.6 |
| PC2 (Switch 2) | 10.0.0.3 | 10.0.0.4 |
| PC3 (Switch 2) | 10.0.0.2 | 10.0.0.4 |

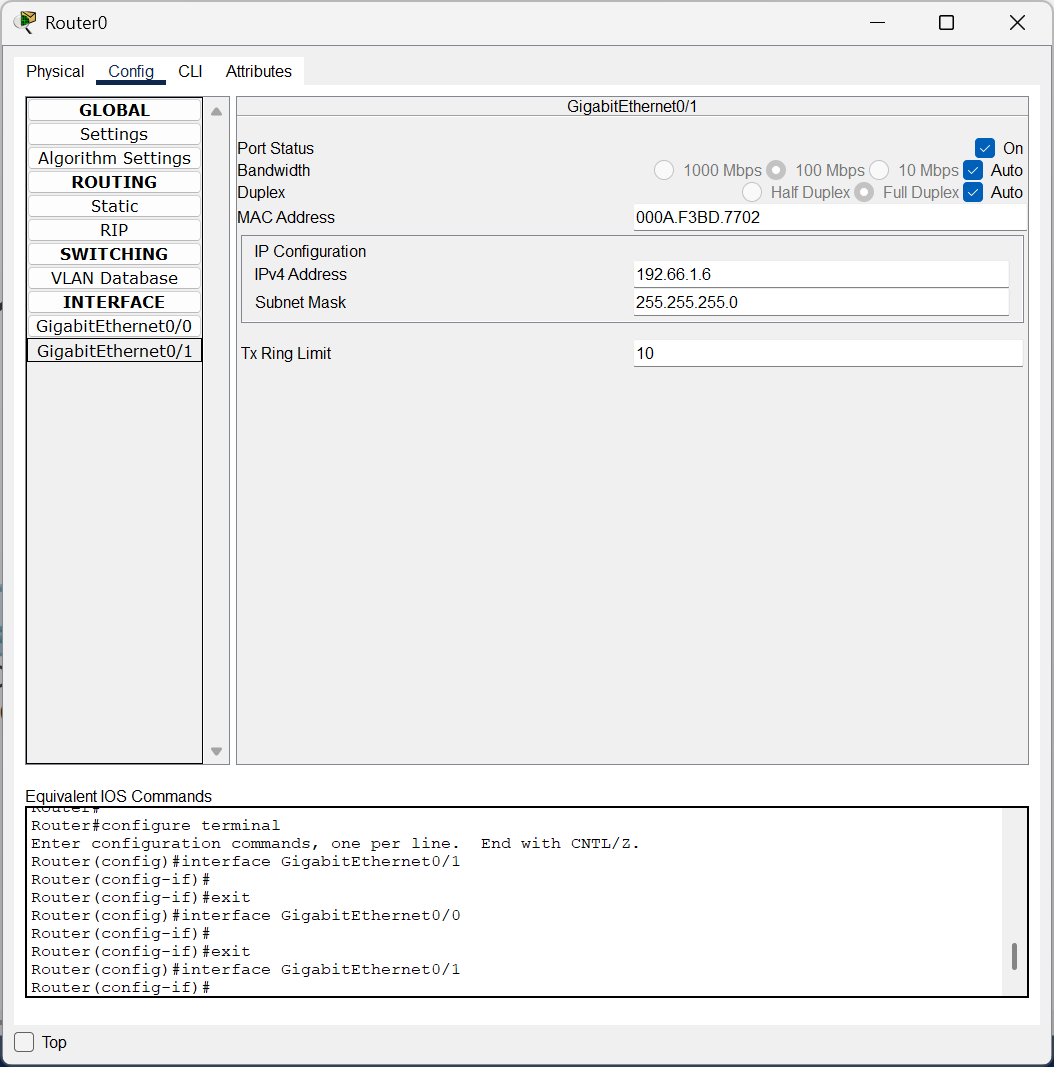
Configure PC1 with giving IP Address as **192.66.1.2**



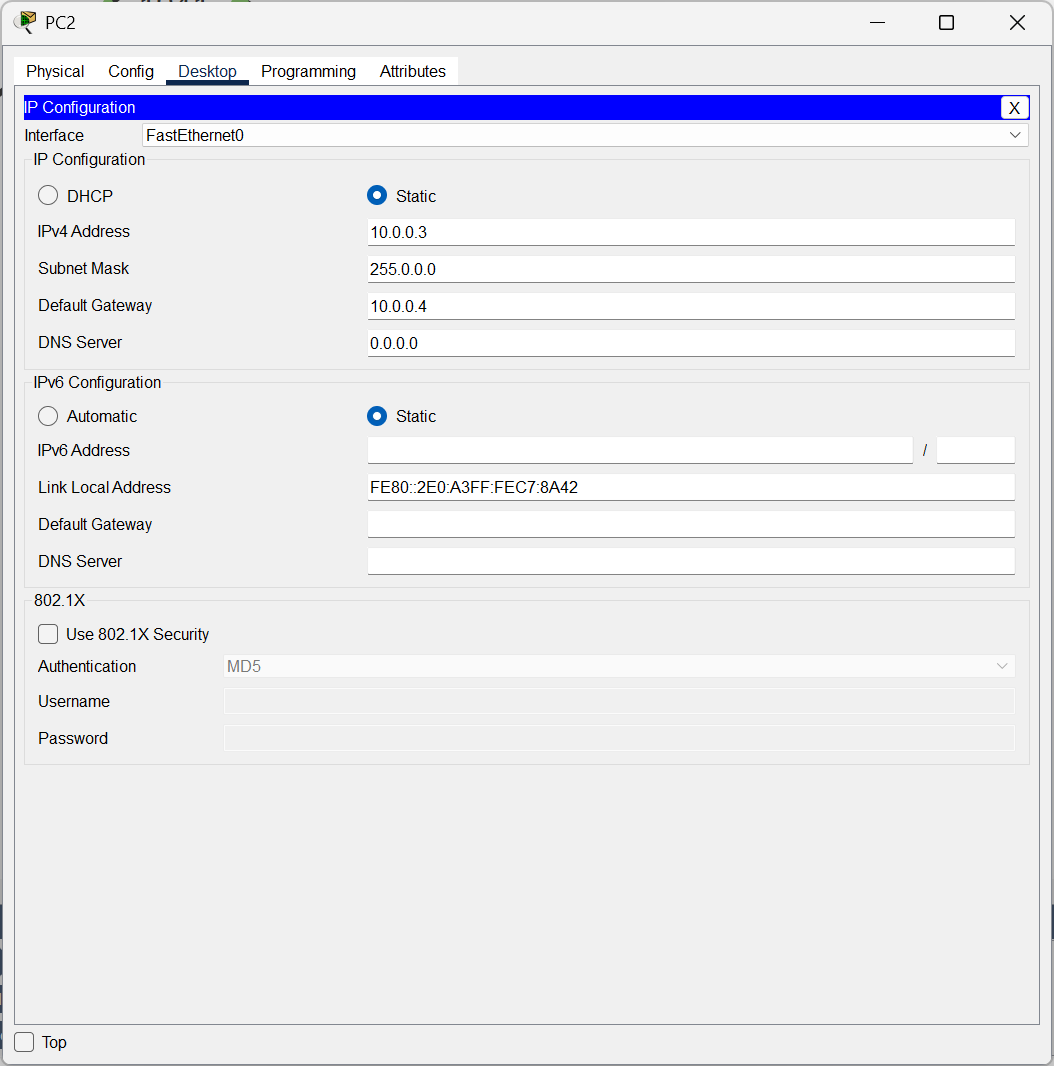
Configure PC0 with giving IP Address as **192.66.1.1**



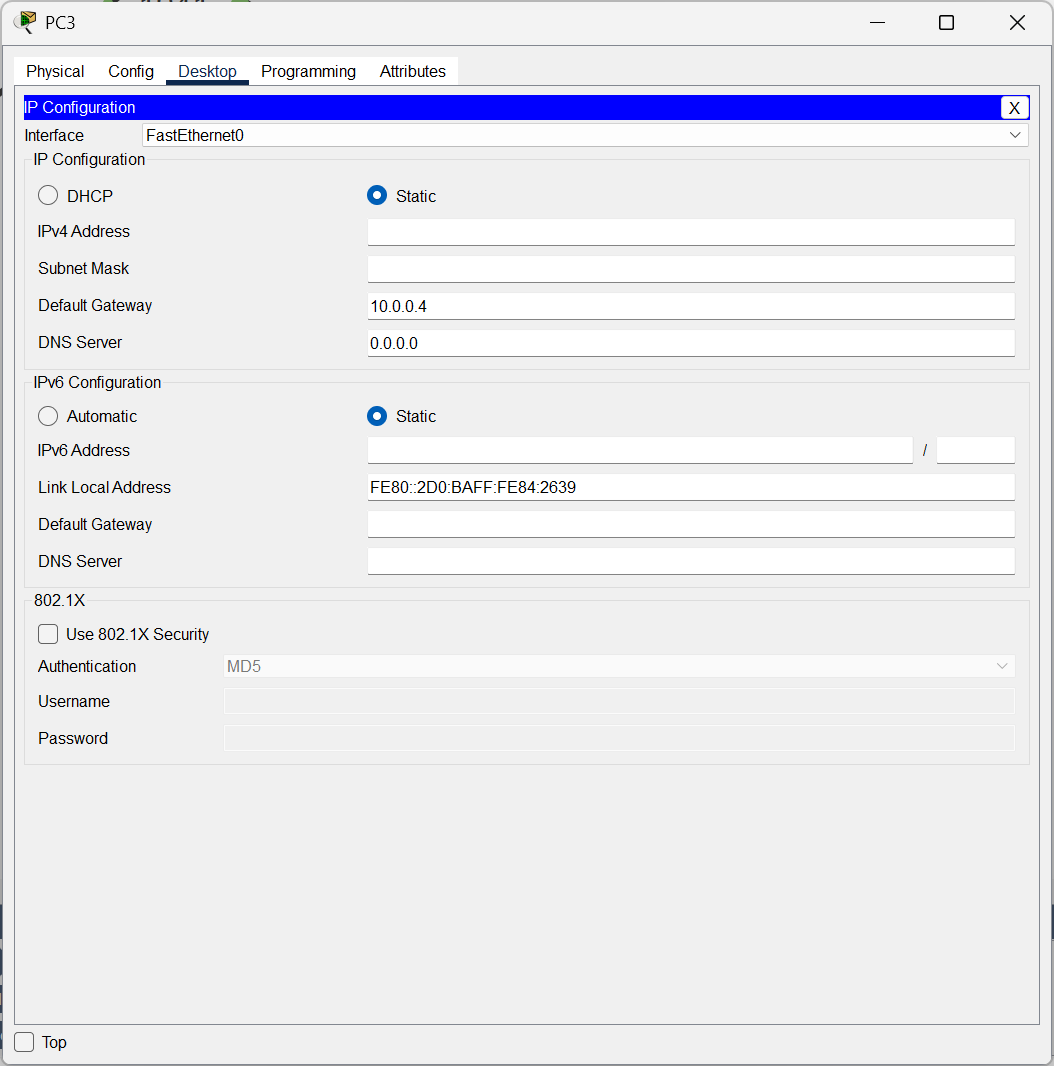
Configure Router0 with giving IPv4 Address as **192.66.1.6 for the network 1.**



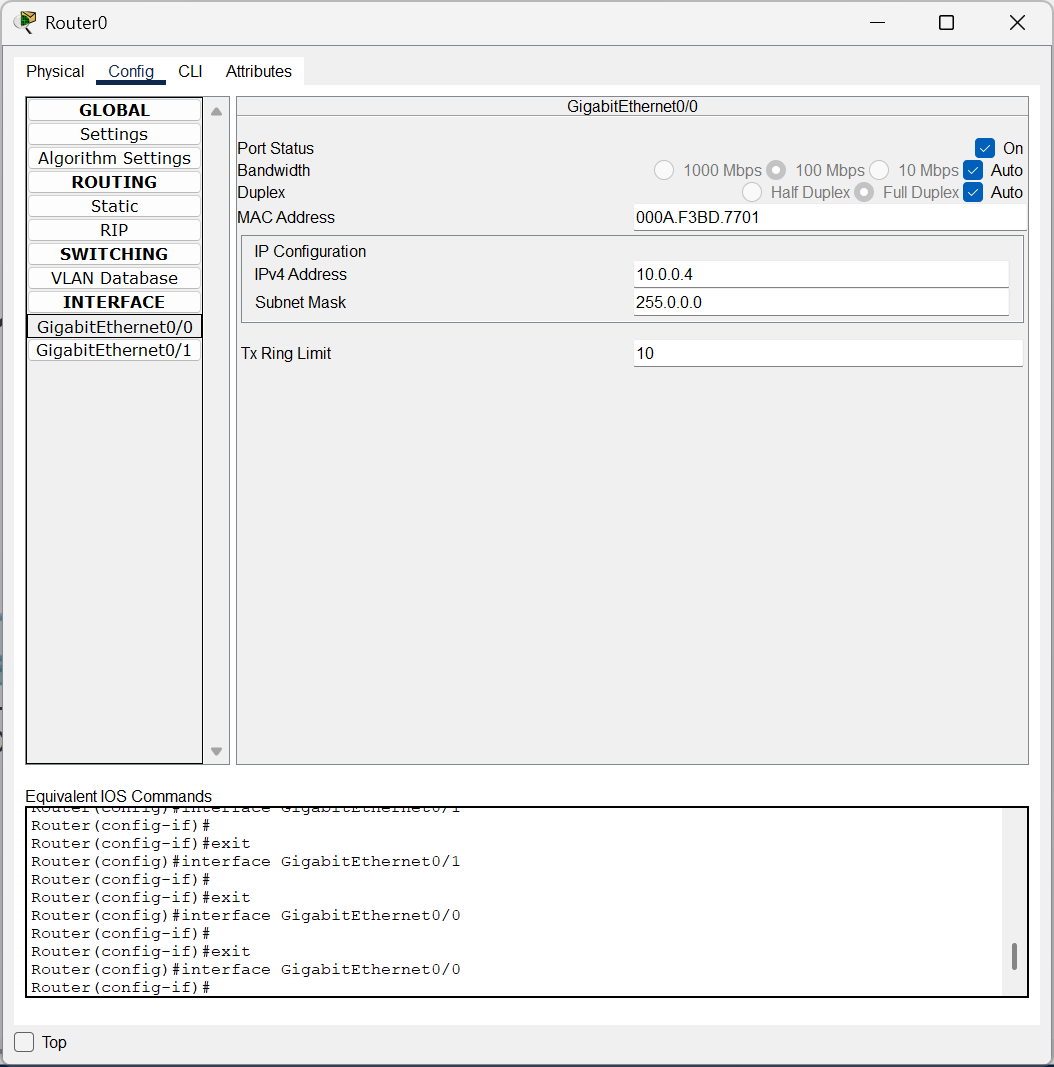
Configure PC2 with giving IP Address as **10.0.0.3**



Configure PC3 with giving IP Address as **10.0.0.2**



Configure Router0 with giving IPv4 Address as **10.0.0.4 for the network 0.**



**CISCO PACKET TRACER DIGITAL REPRESENTATION**

***Representation of 2 different networks connected through switches to a Router.***

