COMPUTER NETWORKS LAB (CSL5403)

Name: Lakhan Kumawat

Roll: 1906055

Program: B.Tech CSE
(5th Sem JUL-DEC 2021)

Test B

Name : Lakhan Kumawat Roll No : 1906055 Course Code : CSL5403

🧪 Experiment -01

🎯 AIM: Write a program for hamming code error detection and correction.

✔ Solution:

```
#include <bits/stdc++.h>
using namespace std;

// Function to generate hamming code
vector<int> generateHammingCode(
    vector<int> msgBits, int m, int r)
{
    // Stores the Hamming Code
    vector<int> hammingCode(r + m);

    // Find positions of redundant bits
    for (int i = 0; i < r; ++i)
    {

        // Placing -1 at redundant bits
        // place to identify it later
        hammingCode[pow(2, i) - 1] = -1;
    }

    int j = 0;

    // Iterate to update the code
    for (int i = 0; i < (r + m); i++)
    {

        // Placing msgBits where -1 is
        // absent i.e., except redundant
        // bits all positions are msgBits
```

```
        if (hammingCode[i] != -1)
        {
            hammingCode[i] = msgBits[j];
            j++;
        }
    }

    for (int i = 0; i < (r + m); i++)
    {

        // If current bit is not redundant
        // bit then continue
        if (hammingCode[i] != -1)
            continue;

        int x = log2(i + 1);
        int one_count = 0;

        // Find msg bits containing
        // set bit at x'th position
        for (int j = i + 2;
             j <= (r + m); ++j)
        {

            if (j & (1 << x))
            {
                if (hammingCode[j - 1] == 1)
                {
                    one_count++;
                }
            }
        }

        // Generating hamming code for
        // even parity
        if (one_count % 2 == 0)
```

```
        {
            hammingCode[i] = 0;
        }
        else
        {
            hammingCode[i] = 1;
        }
    }

    // Return the generated code
    return hammingCode;
}

// Function to find the hamming code
// of the given message bit msgBit[]
void findHammingCode(vector<int> &msgBit)
{

    // Message bit size
    int m = msgBit.size();

    // r is the number of redundant bits
    int r = 1;

    // Find no. of redundant bits
    while (pow(2, r) < (m + r + 1))
    {
        r++;
    }
    // Generating Code
    vector<int> ans = generateHammingCode(msgBit, m, r);

    // Print the code
    cout << "Message bits are: ";
    for (int i = 0; i < msgBit.size(); i++)
        cout << msgBit[i];
```

```
    cout << "\nGenerated codeword is: ";
    for (int i = 0; i < ans.size(); i++)
        cout << ans[i];
}

// Main function
int main()
{
    cout << "Enter Input string: ";
    // Input message bits
    string s;
    cin >> s;
    vector<int> msgBit;
    for (auto ch : s)
    {
        msgBit.push_back(ch - '0');
    }
    // Function Call
    findHammingCode(msgBit);

    return 0;
}
```

Output ScreenShot 🖼️:

```
F:\Computer Networks\Lab>"f:\Computer Networks\Lab\main.exe"
Enter Input string: 0101
Message bits are: 0101
Generated codeword is: 0100101
F:\Computer Networks\Lab>
```

Name : Lakhan Kumawat Roll No : 1906055 Course Code : CSL5403

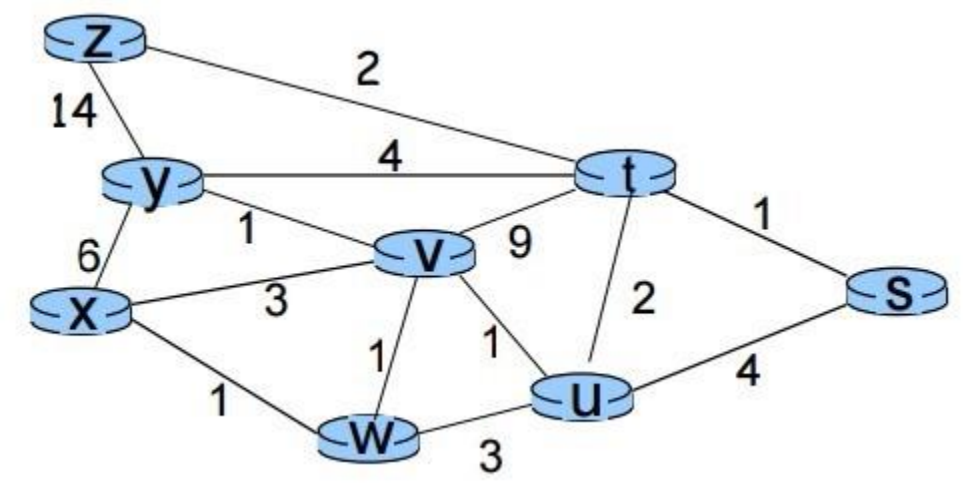
🧪 Experiment -02

🎯 AIM:

Consider the network shown below Fig. 1

and assume that each node initially knows

the costs to each of its neighbours.
Consider the link state routing algorithm and
show the distance table entries at node X (Source node).



✔ Solution:

```
import math
def dijkstra(nodes, distances,source):
    # These are all the nodes which have not been visited yet
    unvisited = {node: None for node in nodes}
    # It will store the shortest distance from one node to
another
    visited = {}
    current = source
    # It will store the predecessors of the nodes
    currentDistance = 0
    unvisited[current] = currentDistance
    # Running the loop while all the nodes have been visited
```

```
    while True:
        # iterating through all the unvisited node
        for neighbour, distance in distances[current].items():
            # Iterating through the connected nodes of
current_node (for
            # example, a is connected with b and c having
values 10 and 3
            # respectively) and the weight of the edges
            if neighbour not in unvisited: continue
            newDistance = currentDistance + distance
            if unvisited[neighbour] is None or
unvisited[neighbour] > newDistance:
                unvisited[neighbour] = newDistance
        # Till now the shortest distance between the source
node and target node
        # has been found. Set the current node as the target
node
        visited[current] = currentDistance
        del unvisited[current]
        if not unvisited: break
        candidates = [node for node in unvisited.items() if
node[1]]
        current, currentDistance = sorted(candidates, key =
lambda x: x[1])[0]
    return visited

nodes = ('s', 't', 'u', 'v', 'w', 'x', 'y','z')
distances = {
    's': {'t': 1, 'u': 4},
    't': {'s': 1, 'u': 2, 'v': 9, 'y':4, 'z':2},
    'u': {'v': 1, 's': 4, 't': 2, 'w': 3},
    'v': {'t': 9, 'u': 1, 'w': 1, 'x':3,'y': 1 },
    'w': {'u': 3, 'v': 1, 'x':1},
    'x': {'w': 1, 'v': 3, 'y': 6},
    'y': {'t':4, 'v':1, 'x':6, 'z':14},
    'z':{'t':2,'y':14}
```

```
    }
print(dijkstra(nodes, distances,'x'))
```

Output ScreenShot 🖼️:

```
PS F:\Computer Networks\Lab> & "C:/Users/LAKHN KUMAWAT/AppData/Local/Programs/Python/Python310/python.exe"
"f:/Computer Networks/Lab/main.py"
{'x': 0, 'w': 1, 'v': 2, 'u': 3, 'y': 3, 't': 5, 's': 6, 'z': 7}
PS F:\Computer Networks\Lab>
```

🧪 Experiment -03

🎯 AIM:

Simulate in packet tracer environment.

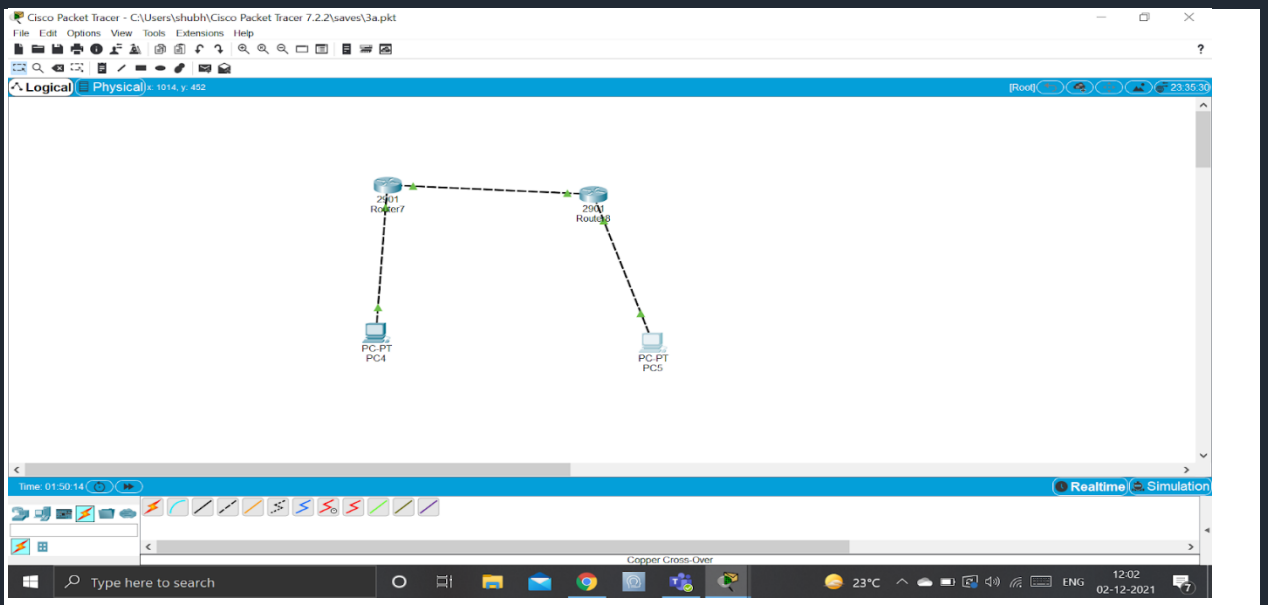
A> Two routers connected with two different Pc.
Configure all IP addresses assigned and check connectivity using ping command.

B> Implement DHCP server with two networks.
Use network devices as reqired.

SOLUTION A:

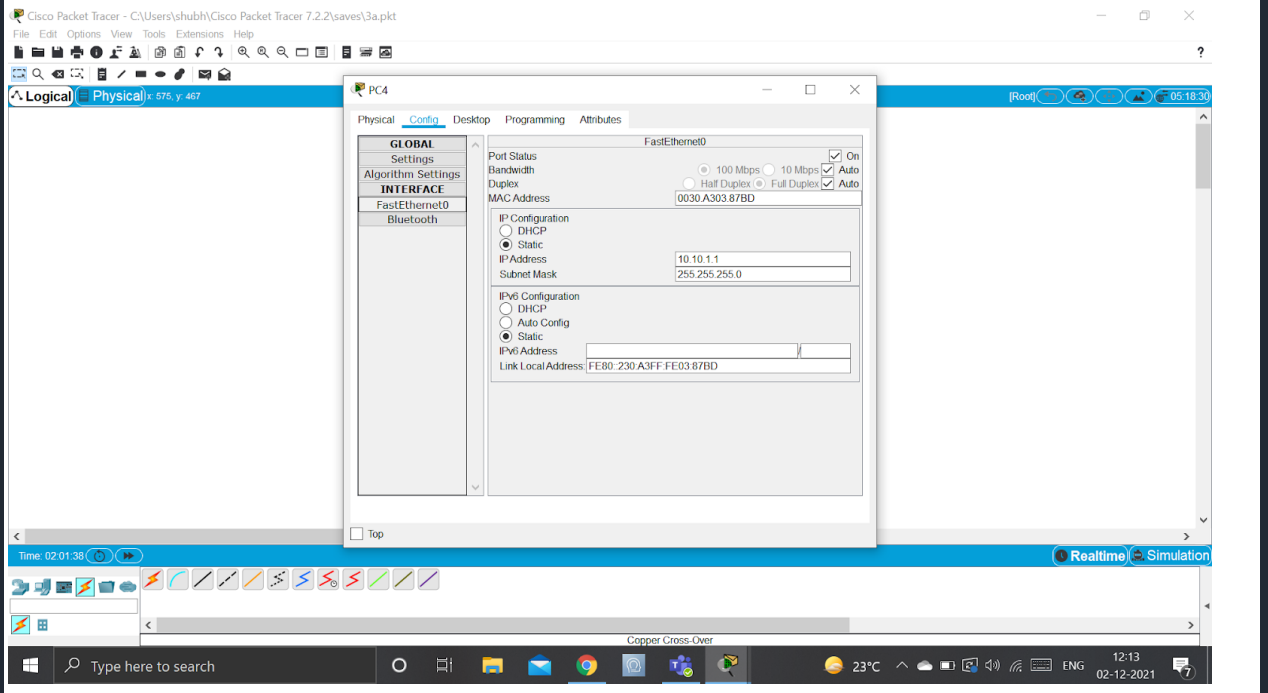
Network Established:

Name : Lakhan Kumawat Roll No : 1906055 Course Code : CSL5403



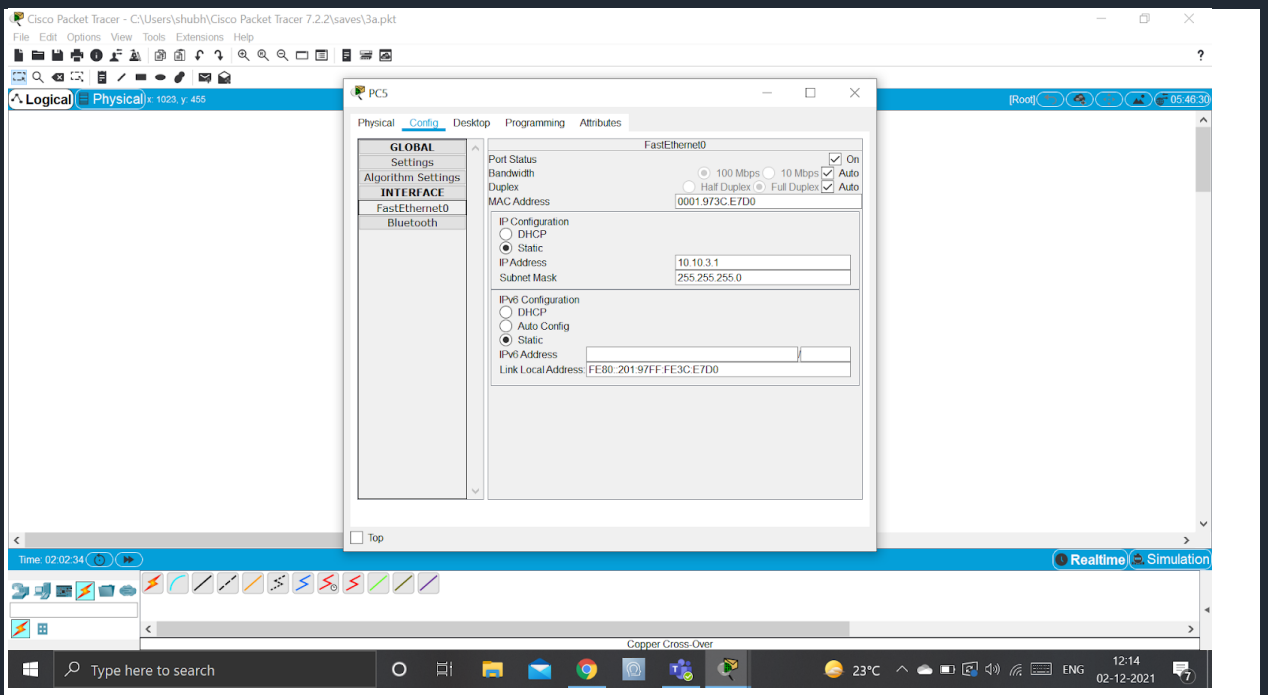
PC configuration :

PC1:



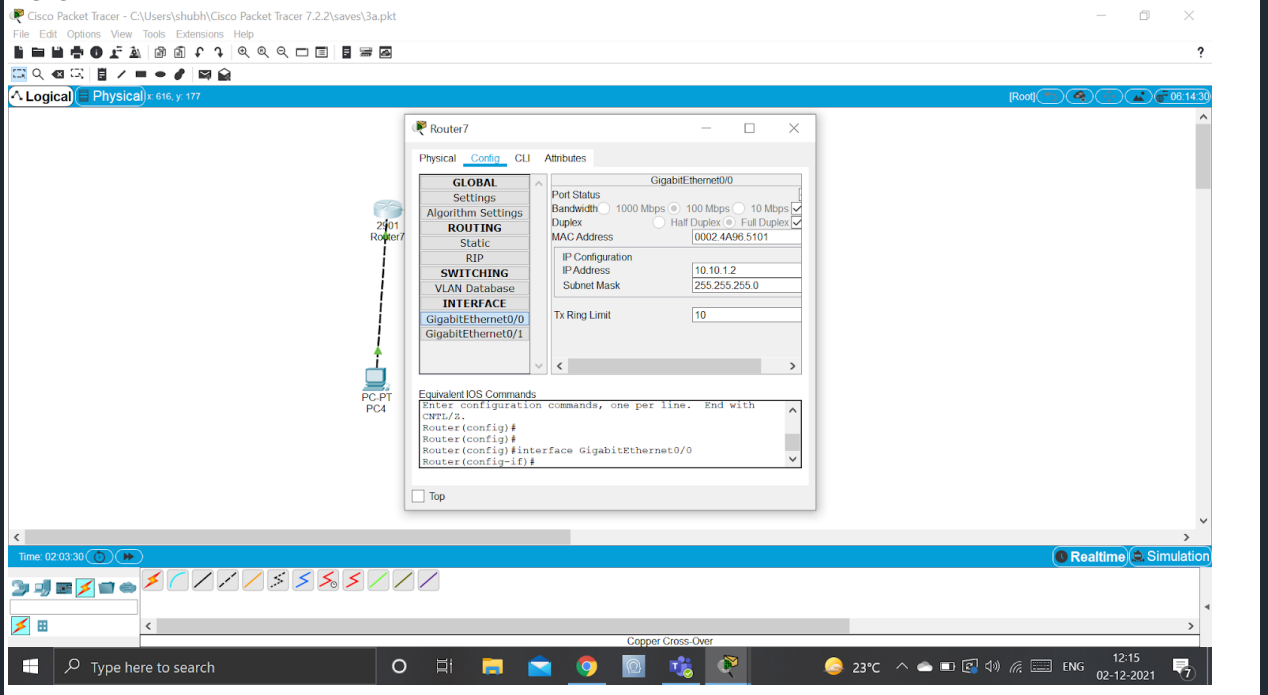
PC2 :

Name : Lakhan Kumawat Roll No : 1906055 Course Code : CSL5403



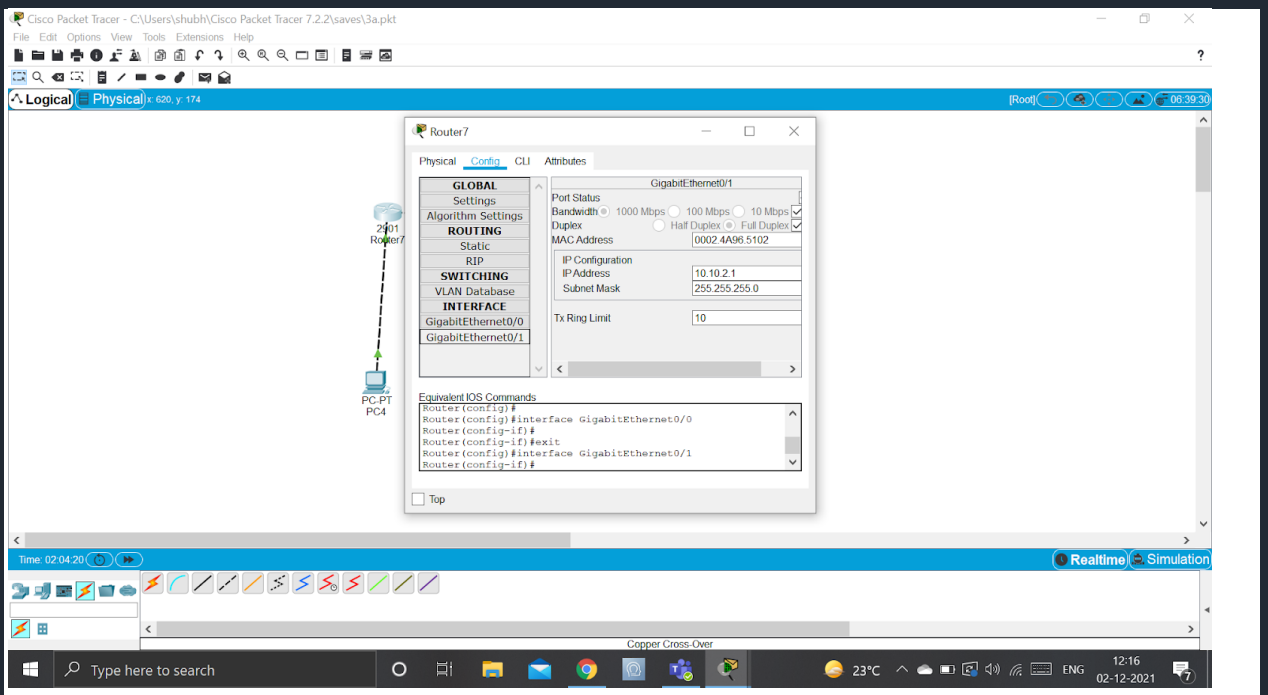
Router Configuration :

Router 1:
Port 1:



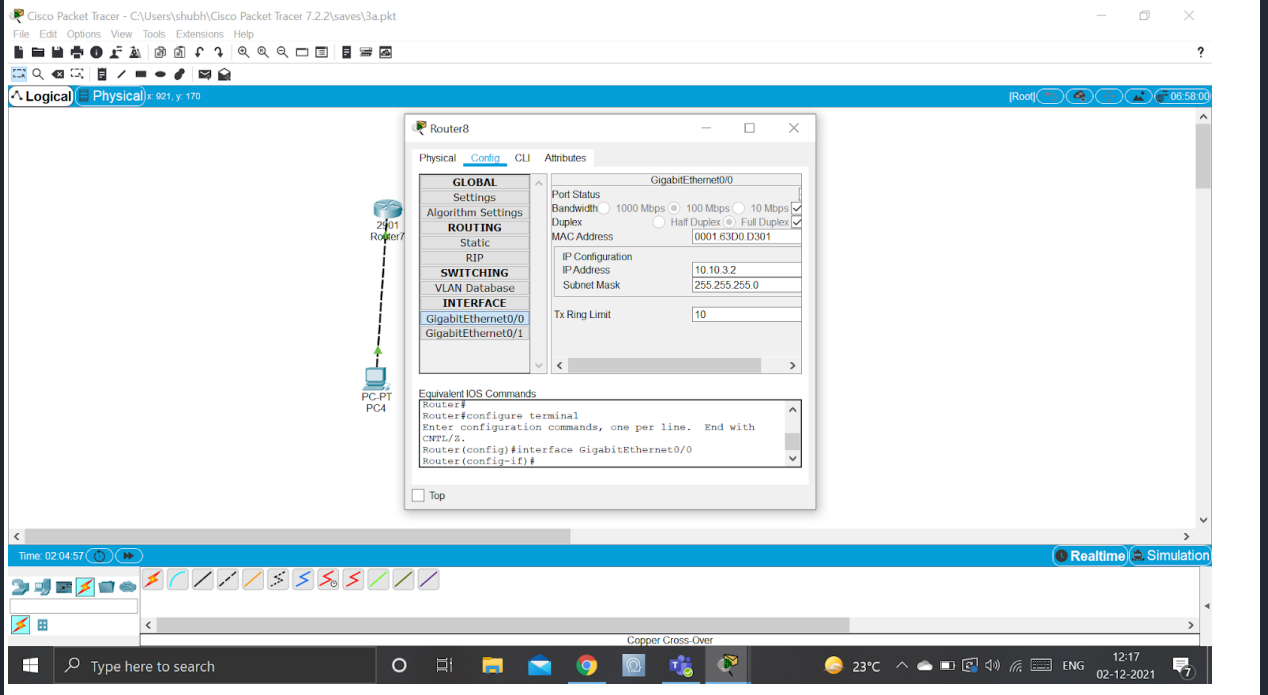
Port 2:

Name : Lakhan Kumawat Roll No : 1906055 Course Code : CSL5403



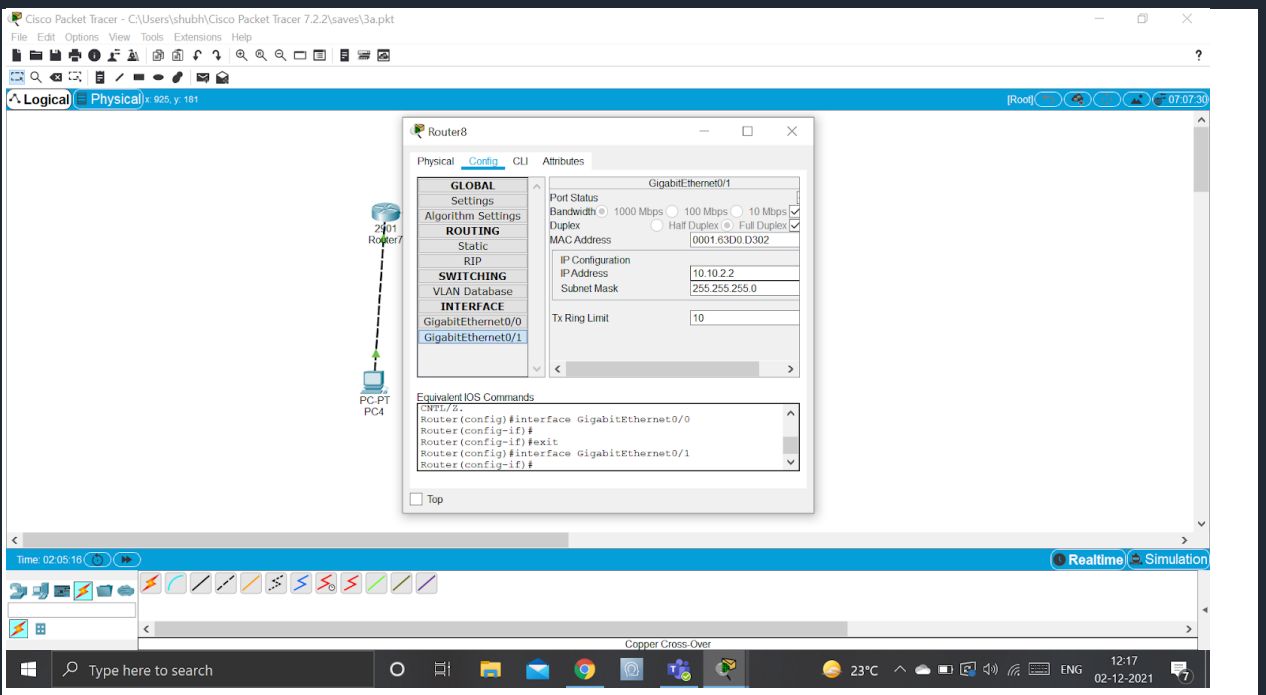
Router 2 :

Port 1:



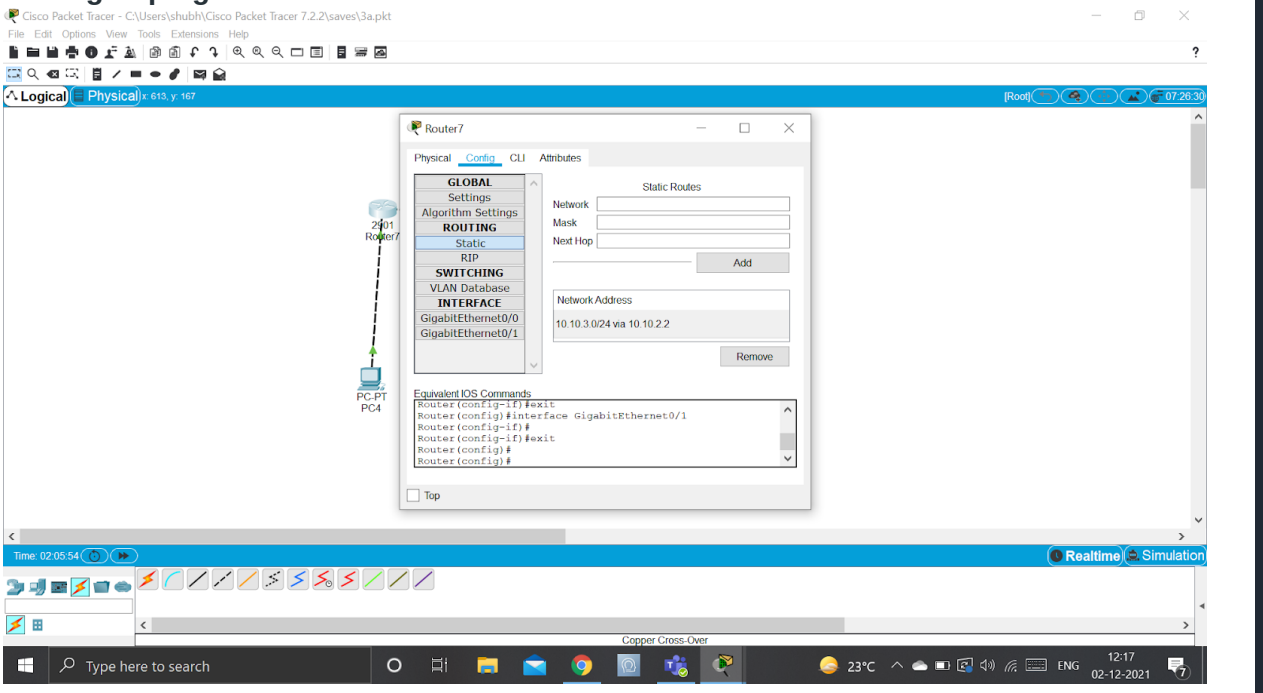
Port 2 :

Name : Lakhan Kumawat Roll No : 1906055 Course Code : CSL5403



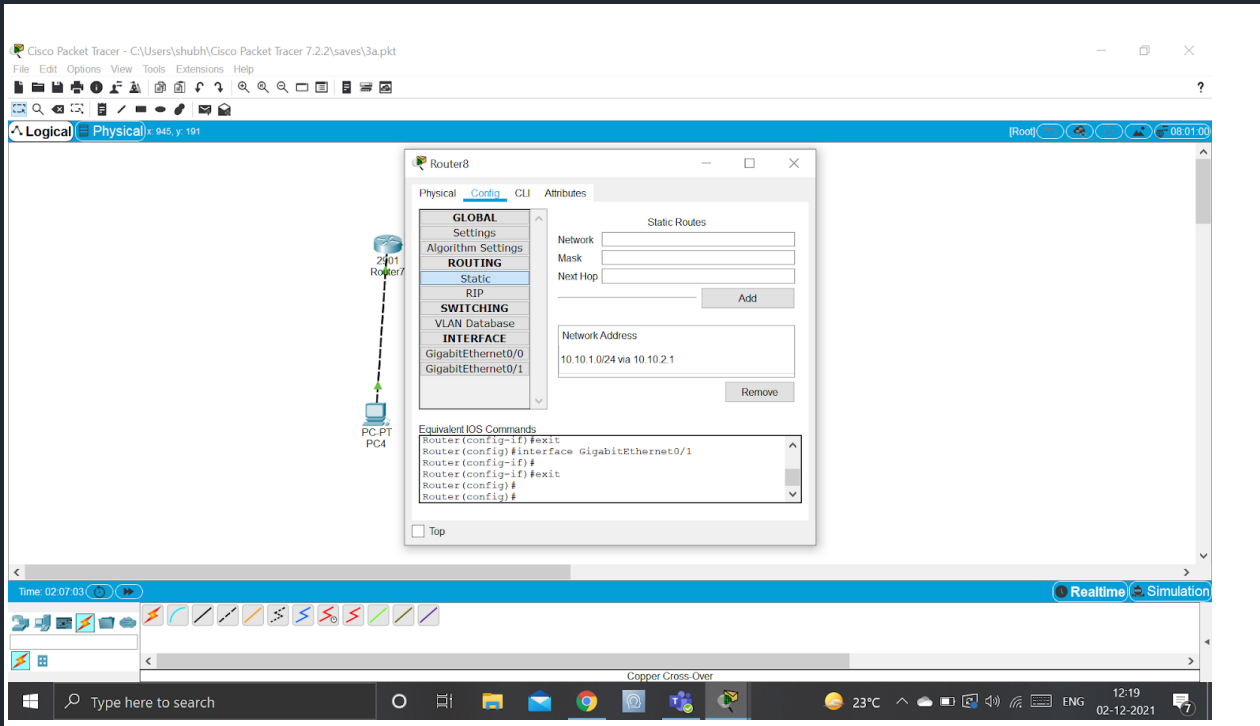
Configuring Static Routing :

For router 1 :
Adding hoping Address via network:



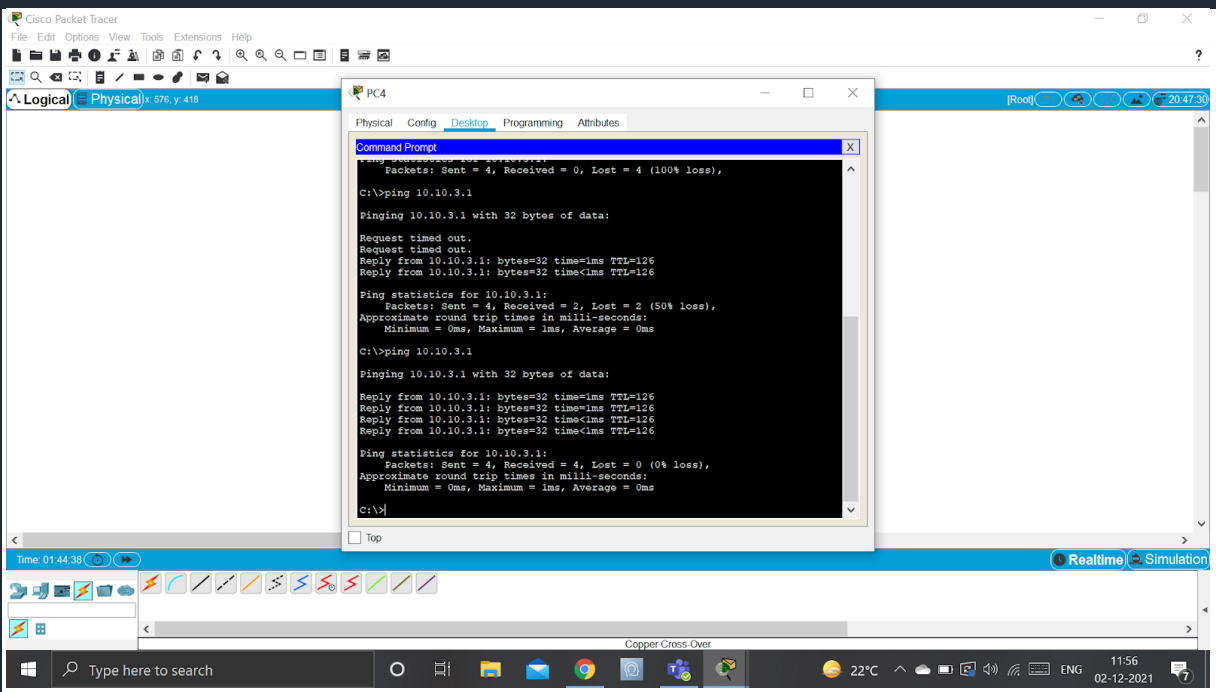
For Router 2 :
Adding hoping address via network :

Name : Lakhan Kumawat Roll No : 1906055 Course Code : CSL5403

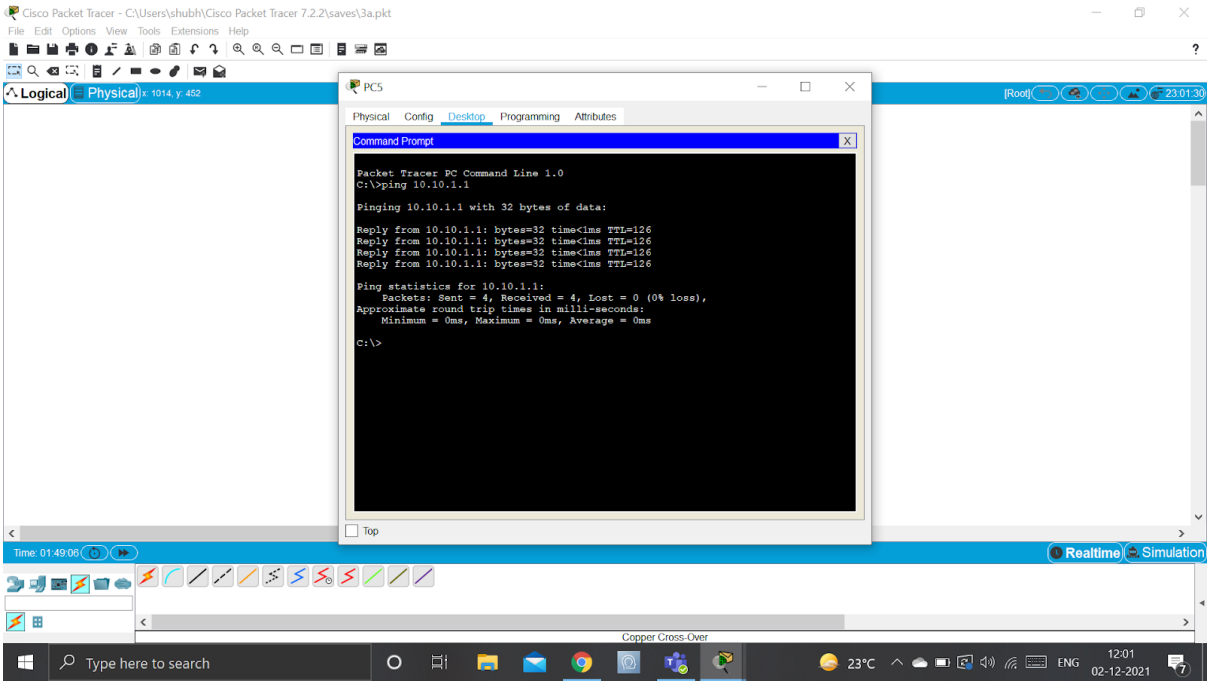


Pinging PC2 from PC1 :

Name : Lakhan Kumawat Roll No : 1906055 Course Code : CSL5403



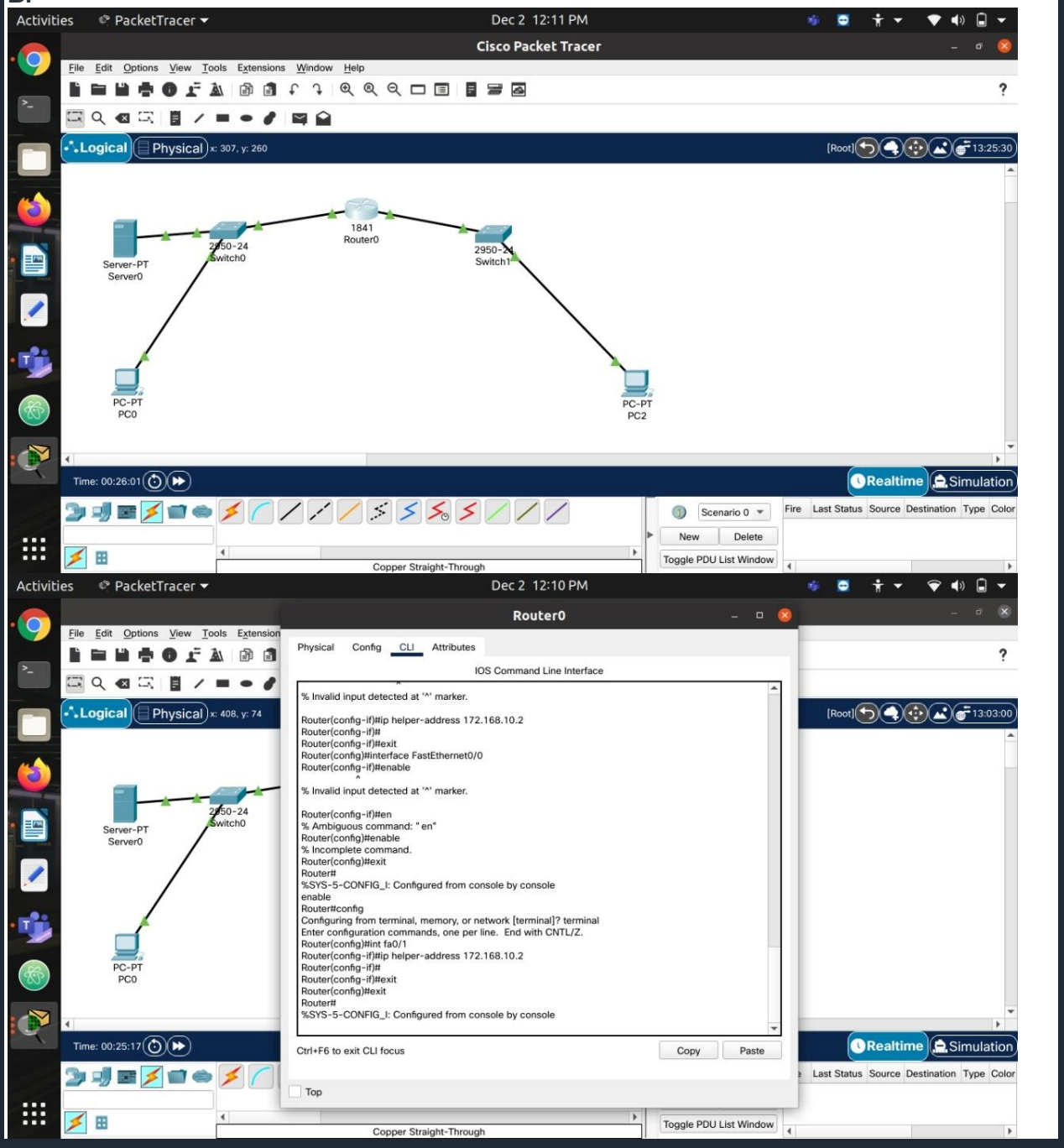
Pinging PC1 from PC2 :



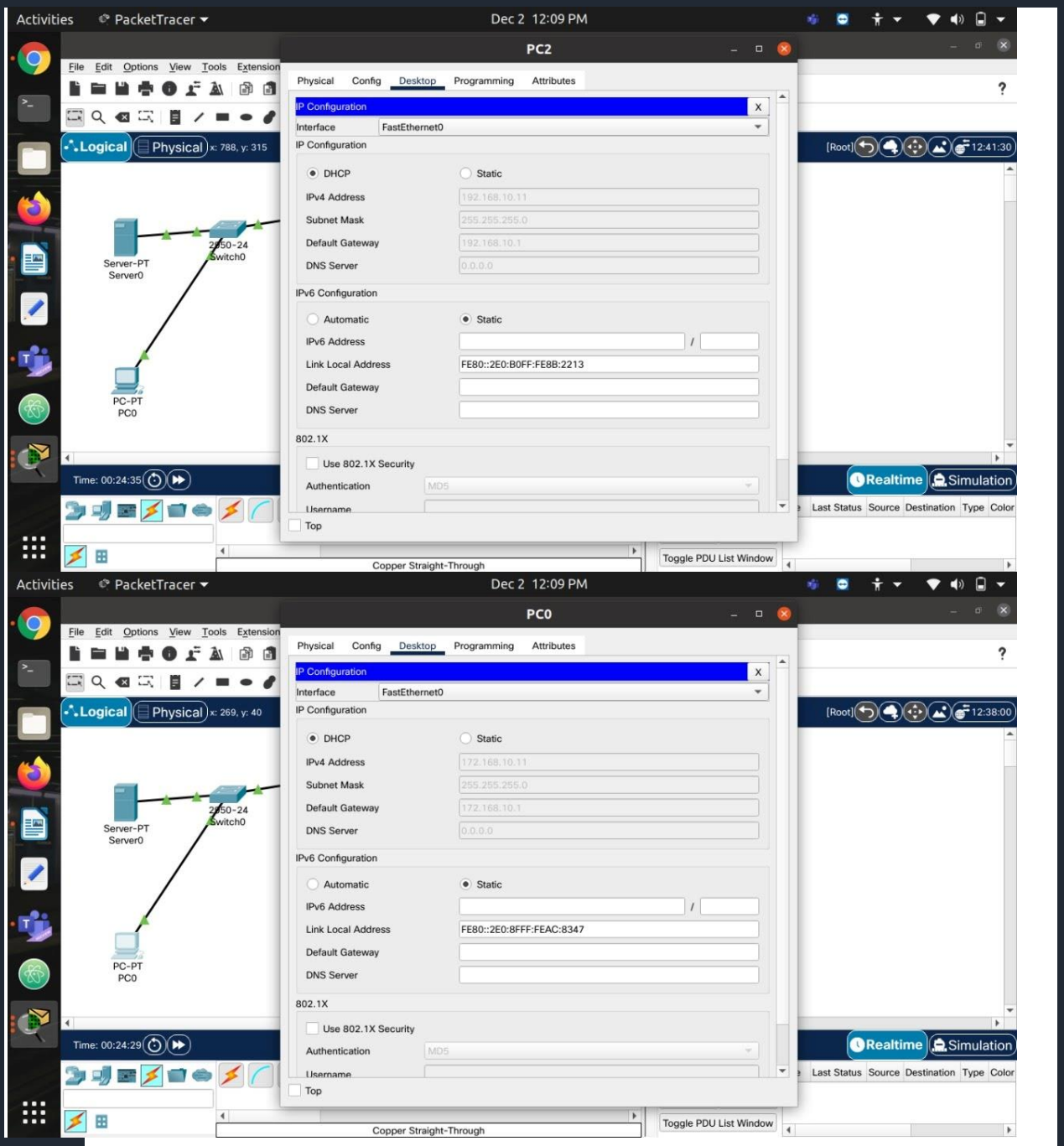
Name : Lakhan Kumawat Roll No : 1906055 Course Code : CSL5403

SOLUTION B:

B.



Name : Lakhan Kumawat Roll No : 1906055 Course Code : CSL5403



---- END OF ASSIGNMENT ----