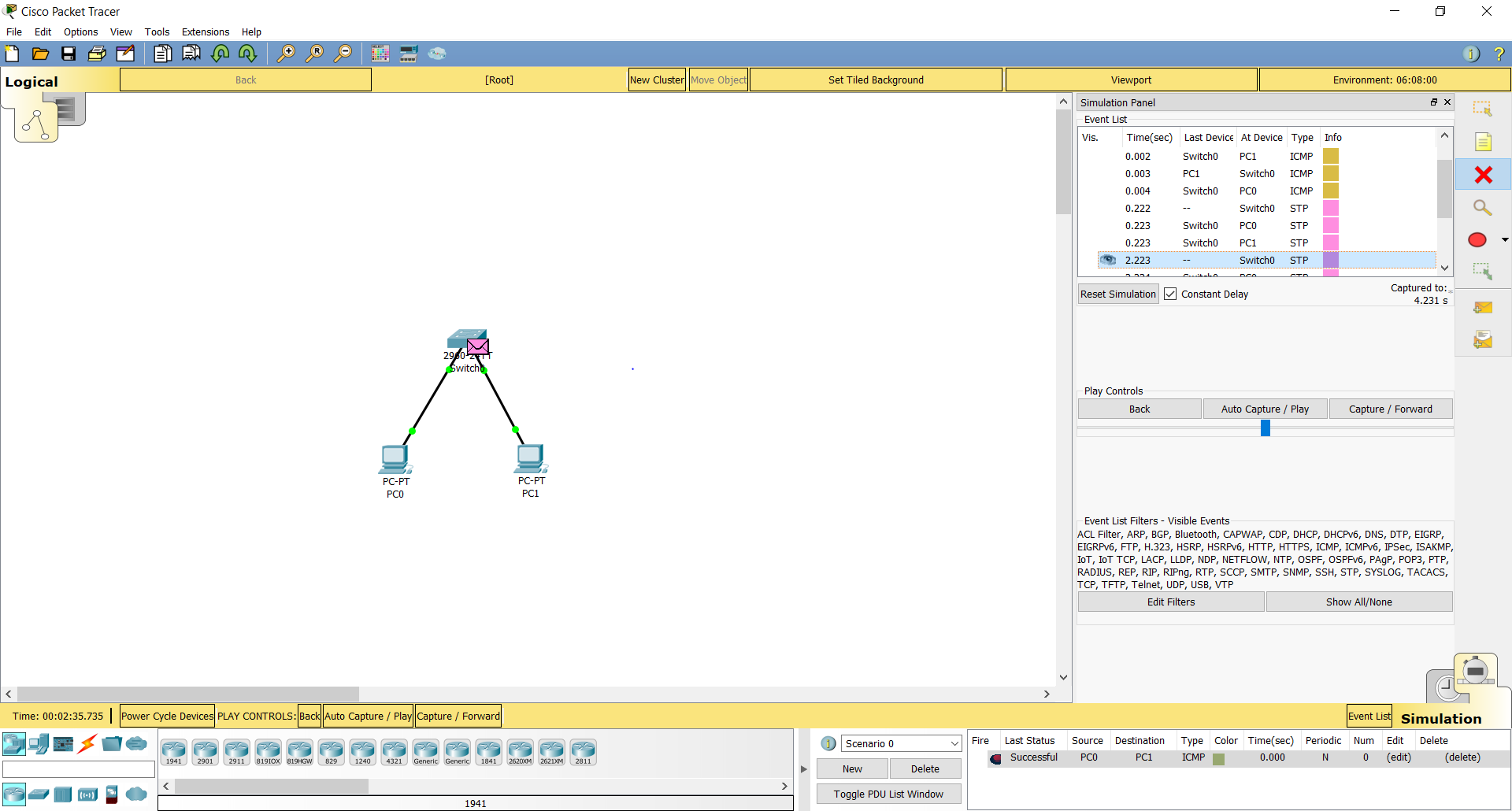
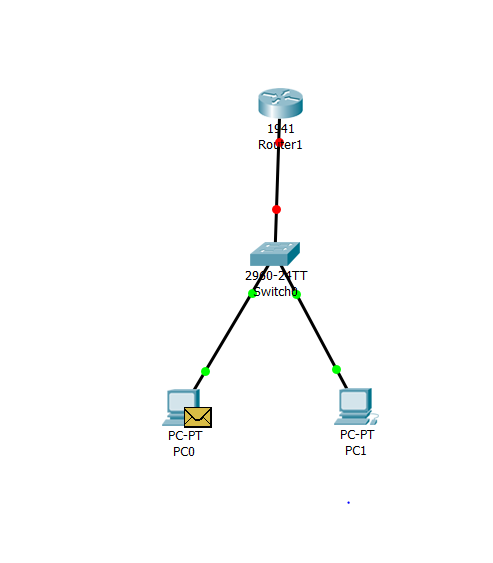
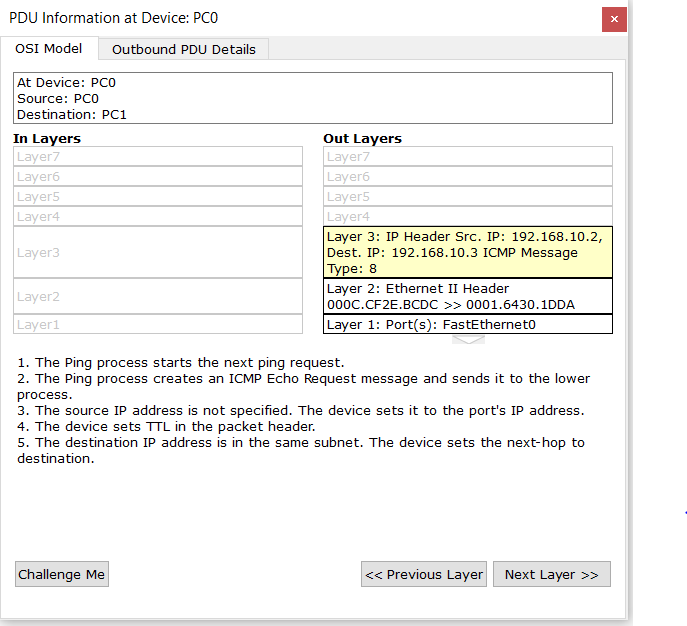
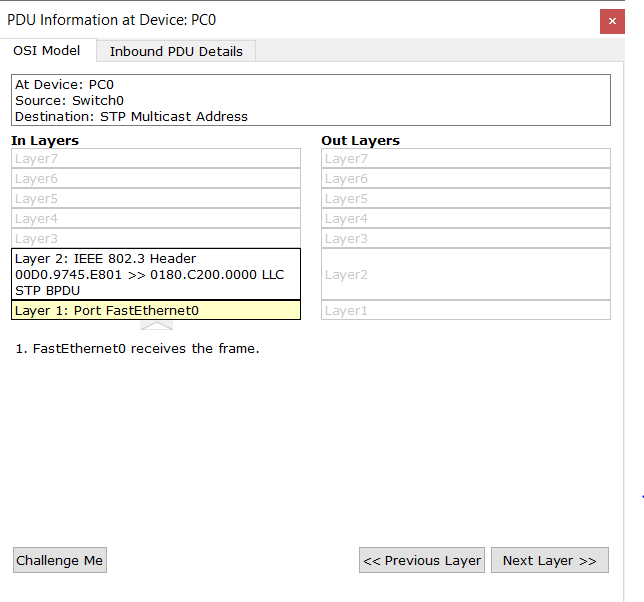
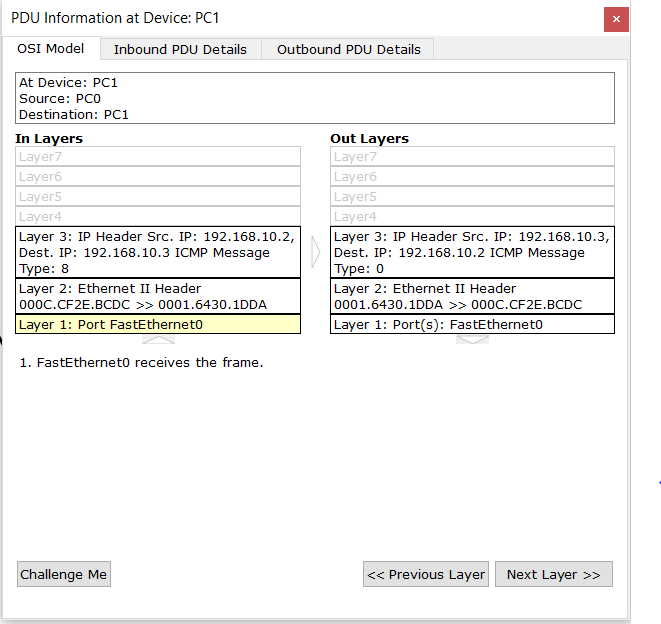
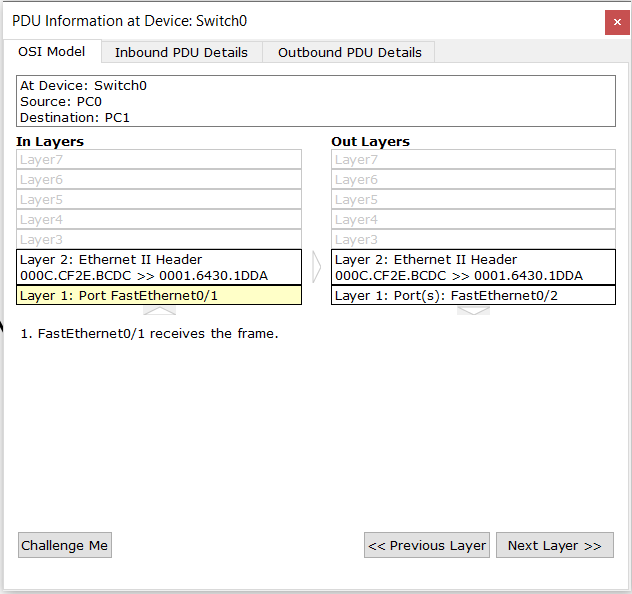
ARYAMAN MISHRA

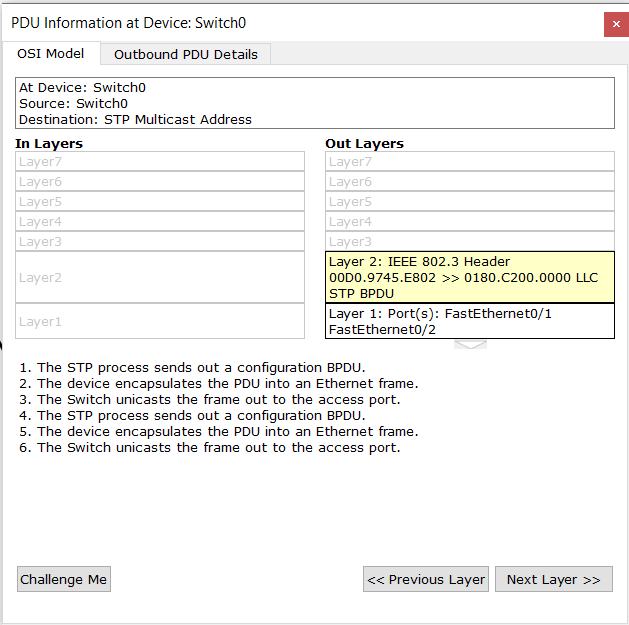
19BCE1027

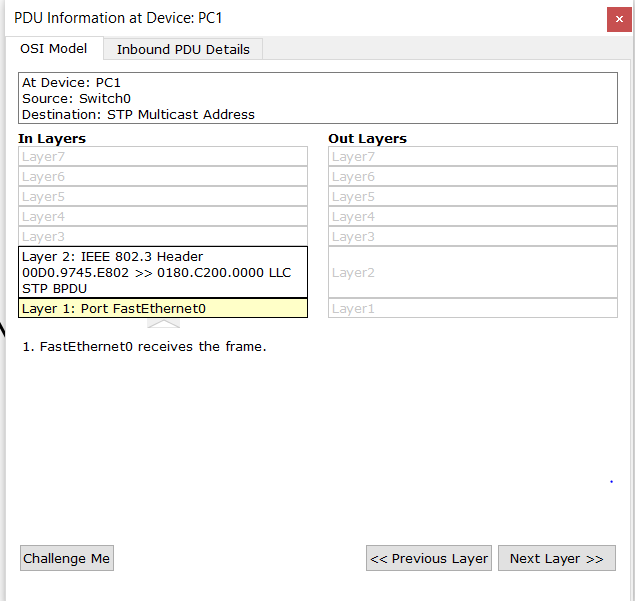




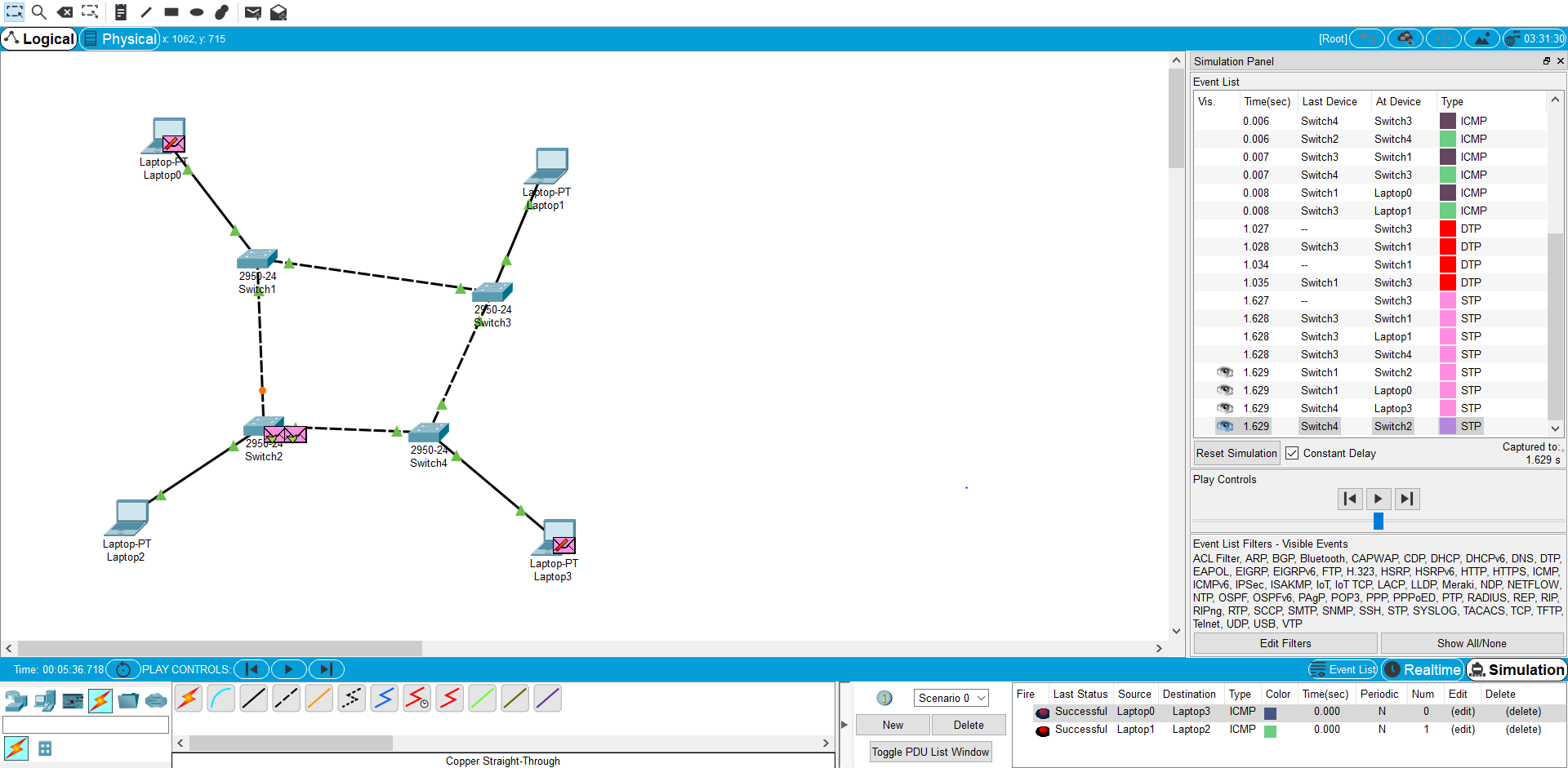


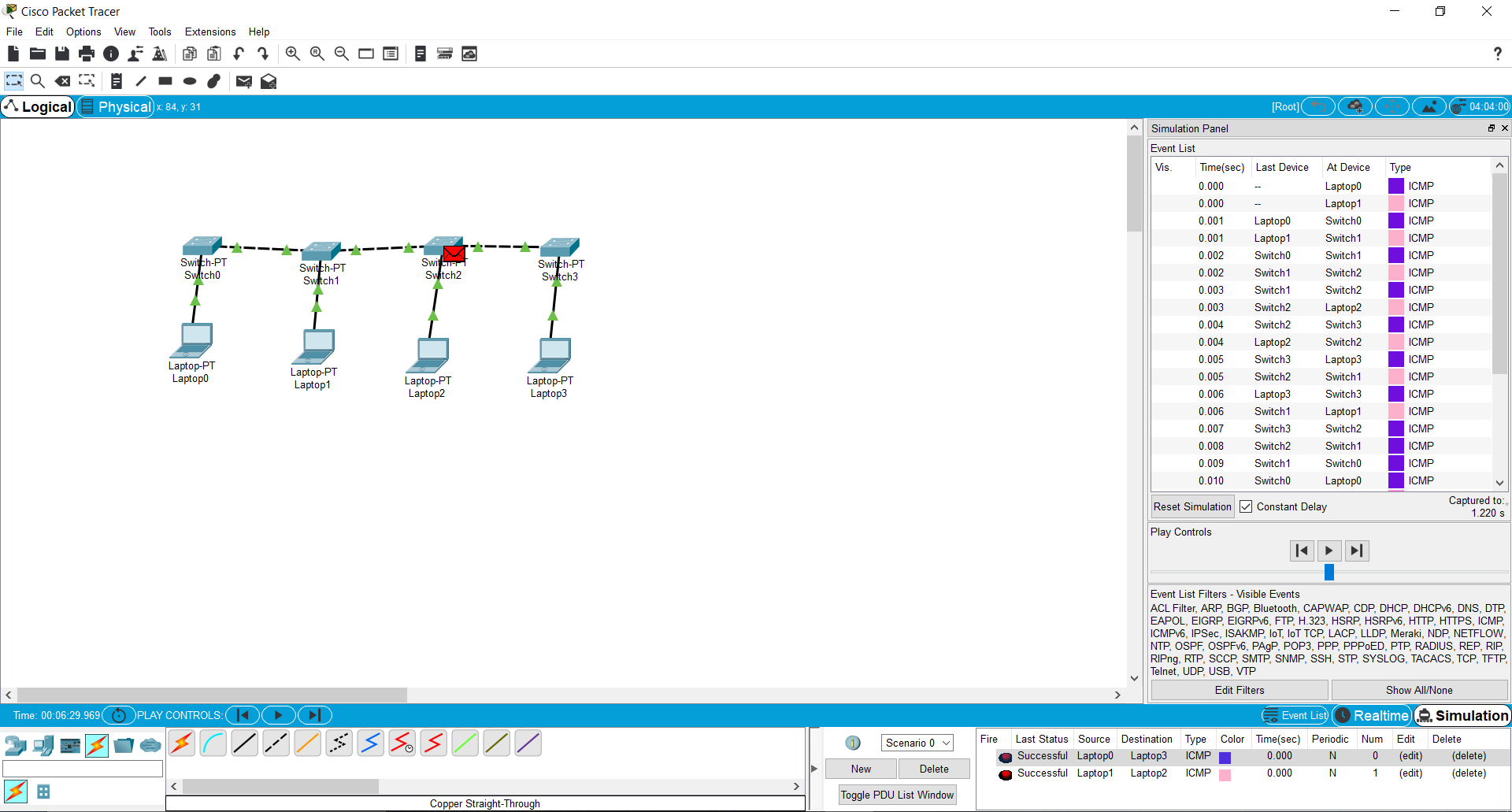


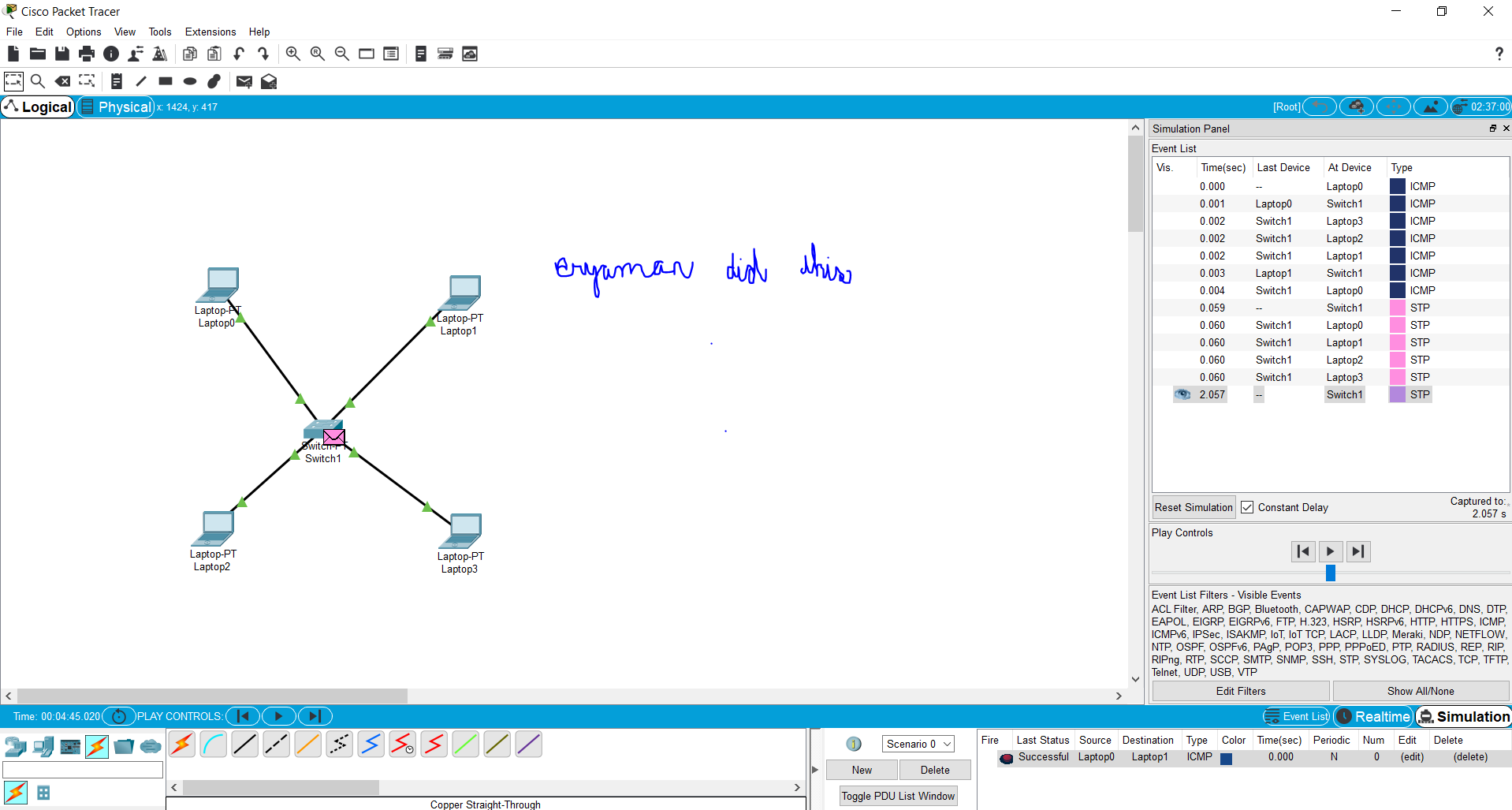




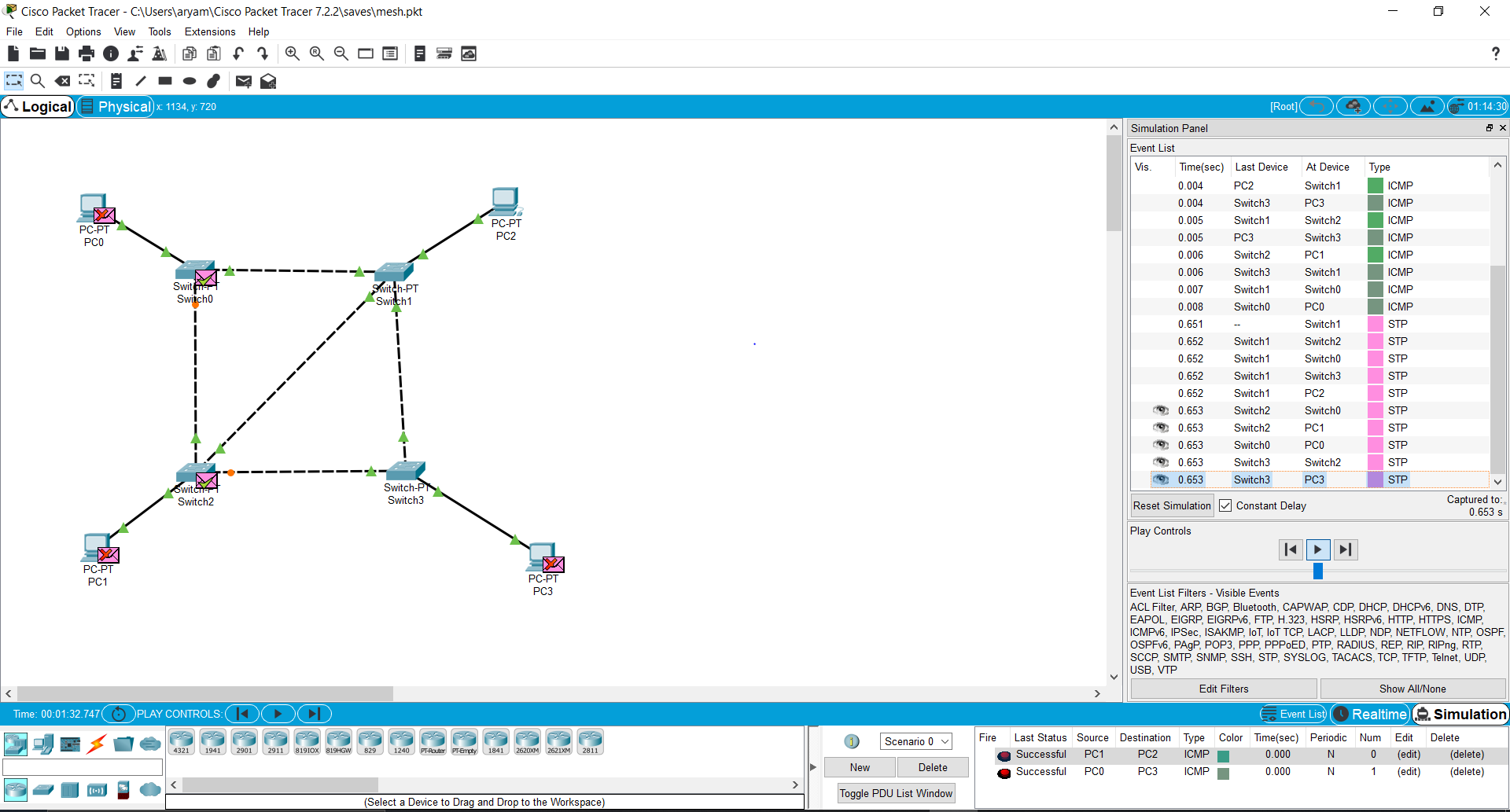
3)STAR,BUS,RING:

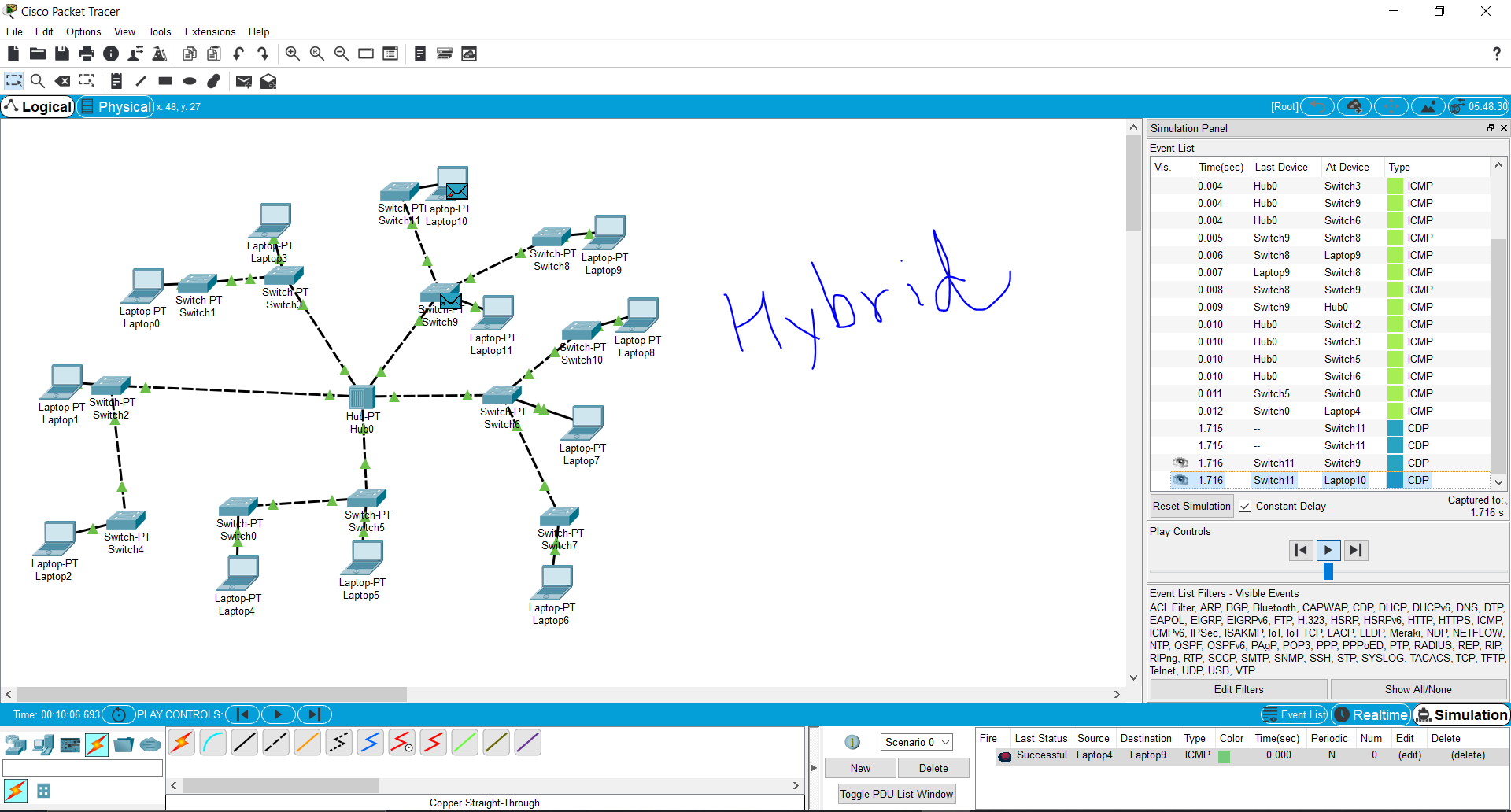




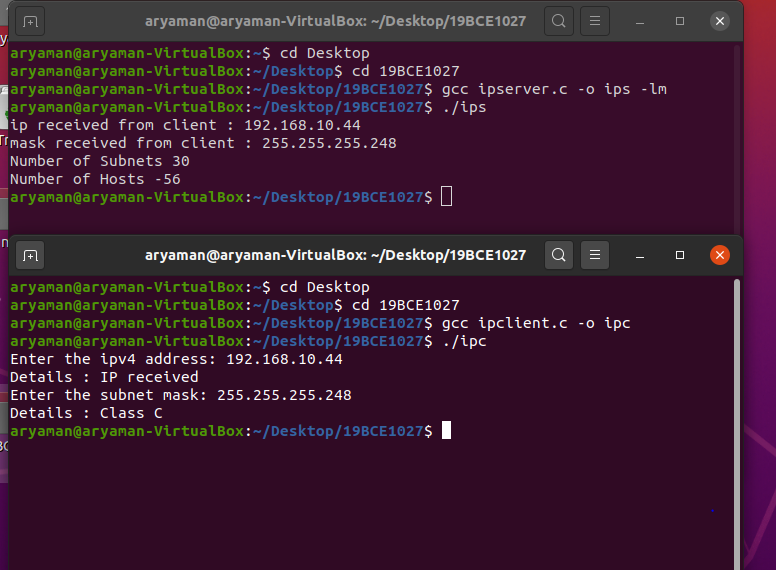


MESH,HYBRID





ASSIGNMENT:

2

IPCLIENT.C

#include<stdio.h>

#include<sys/socket.h>

#include<netinet/in.h>

#include<string.h>

#include<unistd.h>

int main()

{

char msg\_s[200], msg\_r[200];

int sockfd=socket(AF\_INET, SOCK\_STREAM,0);

struct sockaddr\_in SA;

SA.sin\_family = AF\_INET;

SA.sin\_port = htons(7011);

SA.sin\_addr.s\_addr = INADDR\_ANY;

int c1 = connect(sockfd, (struct sockaddr \*)&SA, sizeof(SA));

printf("Enter the ipv4 address: ");

scanf("%s",msg\_s);

//printf("The entered ip is %s",msg\_s);

send(sockfd,msg\_s,200,0);

recv(sockfd,msg\_r,200,0);

printf("Details : %s\n",msg\_r);

printf("Enter the subnet mask: ");

scanf("%s",msg\_s);

//printf("The entered ip is %s",msg\_s);

send(sockfd,msg\_s,200,0);

recv(sockfd,msg\_r,200,0);

printf("Details : %s\n",msg\_r);

close(sockfd);

}

IPSERVER.C

#include <unistd.h>

#include <stdio.h>

#include <sys/socket.h>

#include <stdlib.h>

#include <netinet/in.h>

#include <string.h>

#include<math.h>

void subnetA(int arr[])

{

int a[10],i,n=arr[1],count =0;

for(i=0;n>0;i++)

{

if(n%2==1)

count++;

n=n/2;

}

float host= pow(2,(24-count)) - 2\*pow(2,count);

printf("Number of Subnets %.0f",(pow(2,count)-2));

printf("Number of Hosts %.0f \n",host);

}

void subnetB(int arr[])

{

int a[10],i,n=arr[2],count =0;

for(i=0;n>0;i++)

{

if(n%2==1)

count++;

n=n/2;

}

float host= pow(2,(16-count)) - 2\*pow(2,count);

printf("Number of Subnets %.0f \n",(pow(2,count)-2));

printf("Number of Hosts %.0f \n",host);

}

void subnetC(int arr[])

{

int a[10],i,n=arr[3],count =0;

for(i=0;n>0;i++)

{

if(n%2==1){

count++;

//a[i] = n%2;

}

n=n/2;

}

float host= pow(2,(8-count)) - 2\*pow(2,count);

printf("Number of Subnets %.0f \n",(pow(2,count)-2));

printf("Number of Hosts %.0f \n",host);

}

int main()

{

struct sockaddr\_in serv\_addr, cli\_addr;

char msg\_recv[200], msg\_recv2[200] , msg[200];

int arr[4],arr2[4];

int sockfd = socket(AF\_INET, SOCK\_STREAM, 0);

serv\_addr.sin\_family = AF\_INET;

serv\_addr.sin\_addr.s\_addr = INADDR\_ANY;

serv\_addr.sin\_port = htons(7011);

int b = bind(sockfd, (struct sockaddr \*)&serv\_addr, sizeof(serv\_addr));

int l = listen(sockfd,3);

int clen = sizeof(cli\_addr);

int newsockfd = accept(sockfd, ( struct sockaddr \*)&cli\_addr, &clen);

recv(newsockfd,msg\_recv,200,0);

printf("ip received from client : %s\n",msg\_recv);

send(newsockfd,"IP received ",12,0);

recv(newsockfd,msg\_recv2,200,0);

printf("mask received from client : %s\n",msg\_recv2);

int i=0;

char \* buffer1 = strtok(msg\_recv,".");

int buffer = atoi(buffer1);

while(buffer1 != NULL)

{

arr[i++]=atoi(buffer1);

buffer1 = strtok(NULL,".");

}

i=0;

char \* buffer2 = strtok(msg\_recv2,".");

buffer = atoi(buffer2);

while(buffer2 != NULL)

{

arr2[i++]=atoi(buffer2);

buffer2 = strtok(NULL,".");

}

if(arr[0] <=127 && arr[0] > 0){

strcpy(msg,"Class A");

subnetA(arr2);

}

else if(arr[0] <= 191){

strcpy(msg,"Class B");

subnetB(arr2);}

else if(arr[0] <=223){

strcpy(msg,"Class C");

subnetC(arr2);}

else if(arr[0] <= 239)

strcpy(msg,"Class D");

else if(arr[0] <= 255)

strcpy(msg,"Class E");

else

strcpy(msg,"invalid IP" );

send(newsockfd,msg,200,0);

close(newsockfd);

close(sockfd);

}

TO VALIDATE IP ADDRESS:

// Program to check if a given string is valid IPv4 address or not

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define DELIM "."

/\* function to check whether the string passed is valid or not \*/

bool valid\_part(char\* s)

{

int n = strlen(s);

// if length of passed string is more than 3 then it is not valid

if (n > 3)

return false;

// check if the string only contains digits

// if not then return false

for (int i = 0; i < n; i++)

if ((s[i] >= '0' && s[i] <= '9') == false)

return false;

string str(s);

// if the string is "00" or "001" or "05" etc then it is not valid

if (str.find('0') == 0 && n > 1)

return false;

stringstream geek(str);

int x;

geek >> x;

// the string is valid if the number generated is between 0 to 255

return (x >= 0 && x <= 255);

}

/\* return 1 if IP string is valid, else return 0 \*/

int is\_valid\_ip(char\* ip\_str)

{

// if empty string then return false

if (ip == NULL)

return 0;

int i, num, dots = 0;

int len = strlen(ip);

int count = 0;

// the number dots in the original string should be 3

// for it to be valid

for (int i = 0; i < len; i++)

if (ip[i] == '.')

count++;

if (count != 3)

return false;

// See following link for strtok()

// http:// pubs.opengroup.org/onlinepubs/009695399/functions/strtok\_r.html

char \*ptr = strtok(ip\_str, DELIM);

if (ptr == NULL)

return 0;

while (ptr) {

/\* after parsing string, it must be valid \*/

if (valid\_part(ptr)) {

/\* parse remaining string \*/

ptr = strtok(NULL, ".");

if (ptr != NULL)

++dots;

}

else

return 0;

}

/\* valid IP string must contain 3 dots \*/

// this is for the cases such as 1...1 where originally the

// no. of dots is three but after iteration of the string we find it is not valid

if (dots != 3)

return 0;

return 1;

}

// Driver program to test above functions

int main()

{

char ip1[] = "128.0.0.1";

char ip2[] = "125.16.100.1";

char ip3[] = "125.512.100.1";

char ip4[] = "125.512.100.abc";

is\_valid\_ip(ip1) ? printf("Valid\n") : printf("Not valid\n");

is\_valid\_ip(ip2) ? printf("Valid\n") : printf("Not valid\n");

is\_valid\_ip(ip3) ? printf("Valid\n") : printf("Not valid\n");

is\_valid\_ip(ip4) ? printf("Valid\n") : printf("Not valid\n");

return 0;

}

1)

import java.util.\*;

import java.io.\*;

import java.net.\*;

import java.lang.Math;

class Ip {

// Converts IP address to the binary form

public static int[] bina(String[] str)

{

int re[] = new int[32];

int a, b, c, d, i, rem;

a = b = c = d = 1;

Stack<Integer> st = new Stack<Integer>();

// Separate each number of the IP address

if (str != null)

{

a = Integer.parseInt(str[0]);

b = Integer.parseInt(str[1]);

c = Integer.parseInt(str[2]);

d = Integer.parseInt(str[3]);

}

// convert first number to binary

for (i = 0; i <= 7; i++)

{

rem = a % 2;

st.push(rem);

a = a / 2;

}

// Obtain First octet

for (i = 0; i <= 7; i++) {

re[i] = st.pop();

}

// convert second number to binary

for (i = 8; i <= 15; i++) {

rem = b % 2;

st.push(rem);

b = b / 2;

}

// Obtain Second octet

for (i = 8; i <= 15; i++) {

re[i] = st.pop();

}

// convert Third number to binary

for (i = 16; i <= 23; i++) {

rem = c % 2;

st.push(rem);

c = c / 2;

}

// Obtain Third octet

for (i = 16; i <= 23; i++) {

re[i] = st.pop();

}

// convert fourth number to binary

for (i = 24; i <= 31; i++) {

rem = d % 2;

st.push(rem);

d = d / 2;

}

// Obtain Fourth octet

for (i = 24; i <= 31; i++) {

re[i] = st.pop();

}

return (re);

}

// cls returns class of given IP address

public static char cls(String[] str)

{ int block;

int a = Integer.parseInt(str[0]);

if (a >= 0 && a <= 127)

{System.out.println("block=128");

return ('A');}

else if (a >= 128 && a <= 191) {System.out.println("block=16384");

return ('B'); }

else if (a >= 192 && a <= 223) {System.out.println("block=2097152");

return ('C'); }

else if (a >= 224 && a <= 239) {System.out.println("block=1");

return ('D'); }

else{System.out.println("block=1");

return ('E'); }

}

// Converts IP address

// from binary to decimal form

public static int[] deci(int[] bi)

{

int[] arr = new int[4];

int a, b, c, d, i, j;

a = b = c = d = 0;

j = 7;

for (i = 0; i < 8; i++) {

a = a + (int)(Math.pow(2, j)) \* bi[i];

j--;

}

j = 7;

for (i = 8; i < 16; i++) {

b = b + bi[i] \* (int)(Math.pow(2, j));

j--;

}

j = 7;

for (i = 16; i < 24; i++) {

c = c + bi[i] \* (int)(Math.pow(2, j));

j--;

}

j = 7;

for (i = 24; i < 32; i++) {

d = d + bi[i] \* (int)(Math.pow(2, j));

j--;

}

arr[0] = a;

arr[1] = b;

arr[2] = c;

arr[3] = d;

return arr;

}

public static void main(String args[])

{

Scanner sc=new Scanner(System.in);

int i;

String[] str = new String[4];

String ipr=sc.nextLine();

// You can take user input here

// instead of using default address

// Ask user to enter IP address of form(x.y.z.t/n)

System.out.println("IP address CIDR format is:" + ipr);

// Separate IP address and n

String[] str1 = ipr.split("/");

// IP address

String tr = str1[0];

// Split IP address into 4 subparts x, y, z, t

str = tr.split("\\.");

int[] b = new int[32];

System.out.println();

// Convert IP address to binary form

b = bina(str);

int n = Integer.parseInt(str1[1]);

int[] ntwk = new int[32];

int[] brd = new int[32];

int t = 32 - n;

// Obtanining network address

for (i = 0; i <= (31 - t); i++) {

ntwk[i] = b[i];

brd[i] = b[i];

}

// Set 32-n bits to 0

for (i = 31; i > (31 - t); i--) {

ntwk[i] = 0;

}

// Obtaining Broadcast address

// by setting 32-n bits to 1

for (i = 31; i > (31 - t); i--) {

brd[i] = 1;

}

System.out.println();

// Obtaining class of Address

char c = cls(str);

System.out.println("Class : " + c);

// Converting network address to decimal

int[] nt = deci(ntwk);

// Converting broadcast address to decimal

int[] br = deci(brd);

// Printing in dotted decimal format

System.out.println("First Address : " + nt[0]

+ "." + nt[1] + "." + nt[2] + "." + nt[3]);

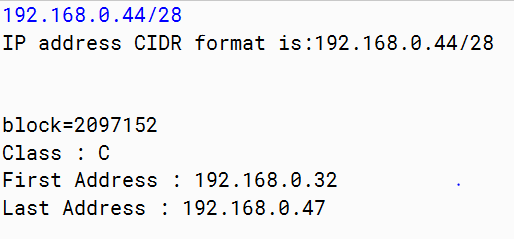
// Printing in dotted decimal format

System.out.println("Last Address : "

+ br[0] + "." + br[1] + "." + br[2] + "." + br[3]);

}

}



// Java program to determine class, Network

// and Host ID of an IPv4 address

import java.util.\*;

class NetworkId{

String findClass(String str){

// Calculating first occurrence of '.' in str

int index = str.indexOf('.');

// First octate in str in decimal form

String ipsub = str.substring(0,index);

int ip = Integer.parseInt(ipsub);

// Class A

if (ip>=1 && ip<=126)

{

return "A";

}

// Class B

else if (ip>=128 && ip<=191) {

return "B";

}

// Class C

else if (ip>=192 && ip<223) {

return "C";

}

// Class D

else if (ip >=224 && ip<=239) {

return "D";

}

// Class E

else{System.out.println("block=1");

return "E";

}

}

void seprate(String str, String ipClass){

// Initializing network and host empty

String network = "", host = "";

if(ipClass == "A"){

int index = str.indexOf('.');

network = str.substring(0,index);

host = str.substring(index+1,str.length());

}else if(ipClass == "B"){

//Position of breaking network and HOST id

int index = -1;

int dot = 2;

for(int i=0;i<str.length();i++){

if(str.charAt(i)=='.'){

dot -=1;

if(dot==0){

index = i;

break;

}

}

}

network = str.substring(0,index);

host = str.substring(index+1,str.length());

}else if(ipClass == "C"){

//Position of breaking network and HOST id

int index = -1;

int dot = 3;

for(int i=0;i<str.length();i++){

if(str.charAt(i)=='.'){

dot -=1;

if(dot==0){

index = i;

break;

}

}

}

network = str.substring(0,index);

host = str.substring(index+1,str.length());

}else if(ipClass == "D" || ipClass == "E"){

System.out.println("In this Class, IP address"+

" is not divided into Network and Host IDs");

return;

}

System.out.println("Network ID is "+network);

System.out.println("Host ID is "+host);

}

public static void main(String[] args) {

Scanner sc=new Scanner(System.in);

NetworkId ob=new NetworkId();

System.out.println("Enter IP Address.");

String str=sc.nextLine();

String ipClass = ob.findClass(str);

System.out.println("Given IP address belings to Class "+ipClass);

ob.seprate(str,ipClass);

}

}

