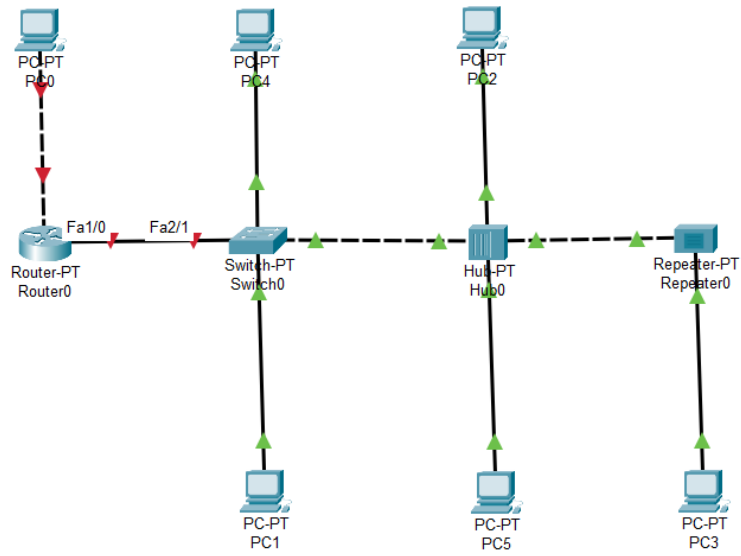
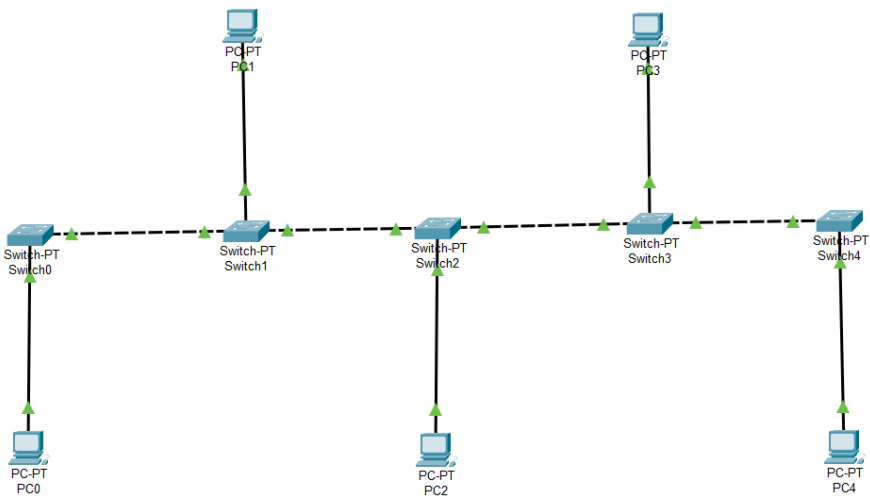


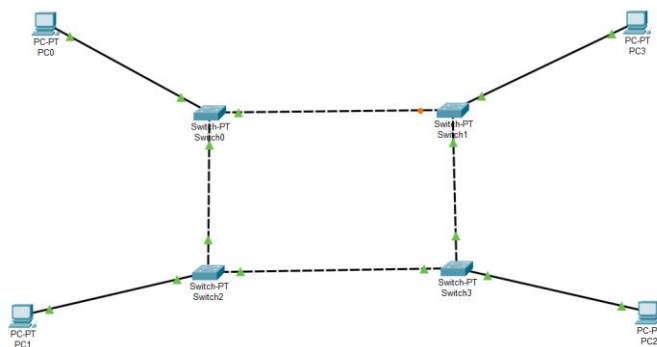
Network devices



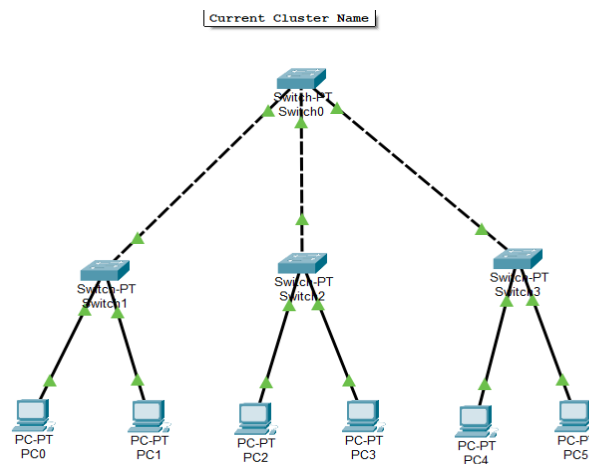
Bus topology



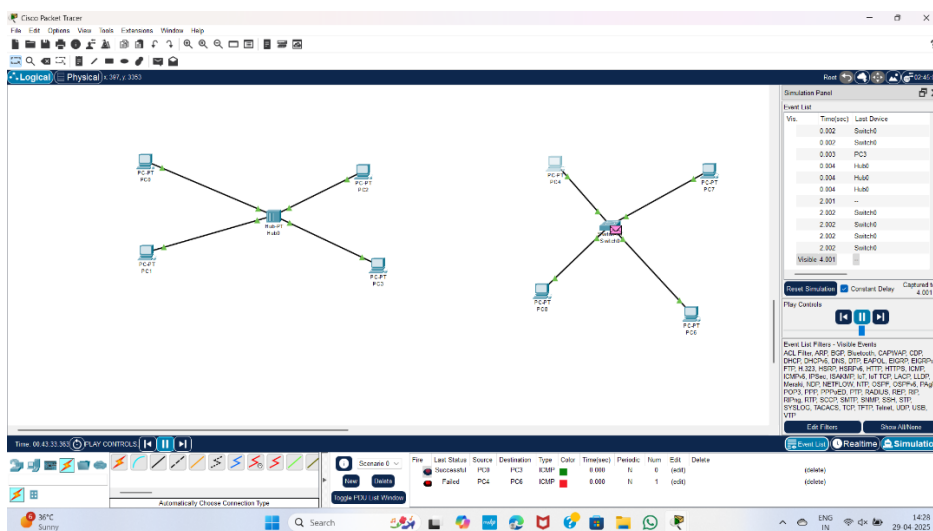
Ring topology



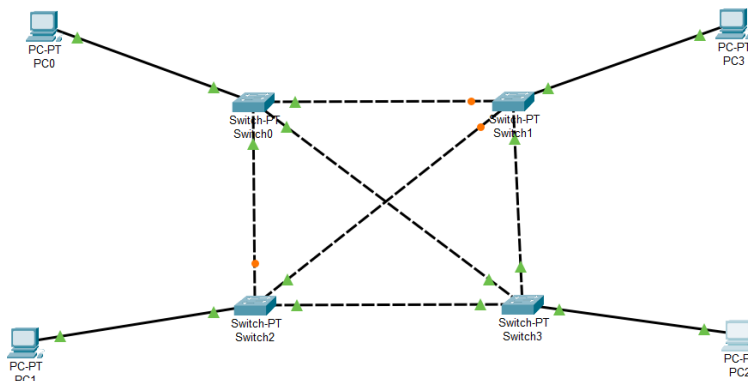
Tree topology



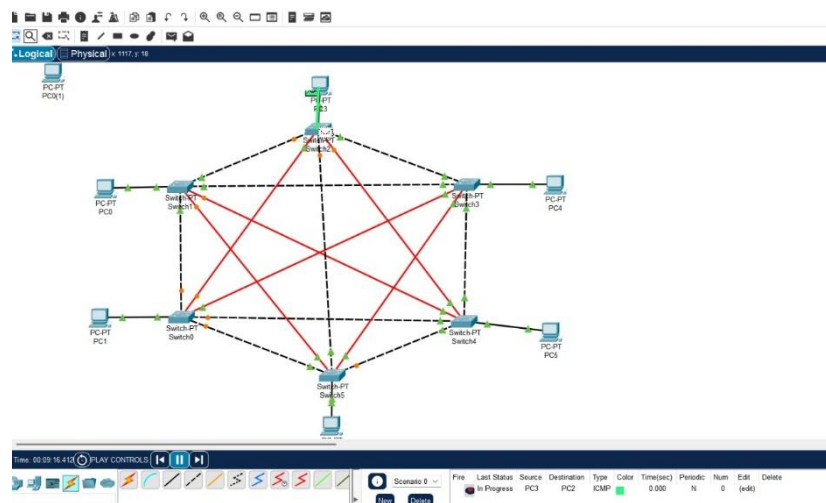
Star topology



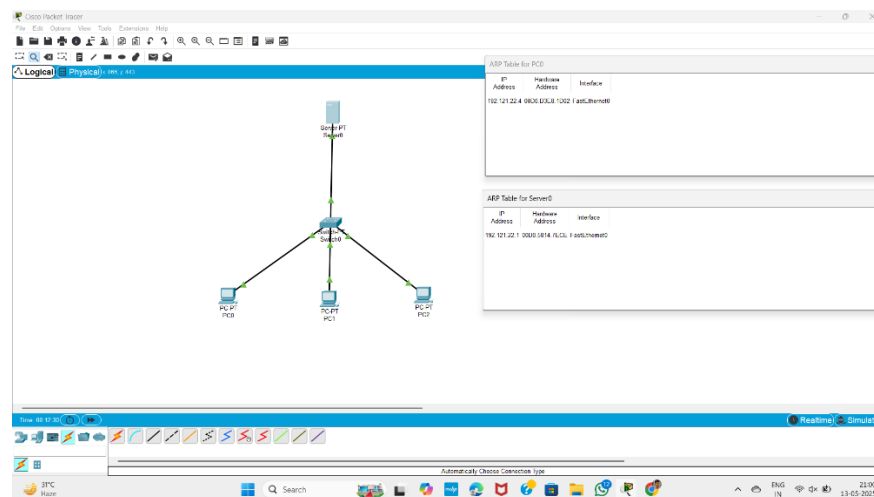
Mesh topology



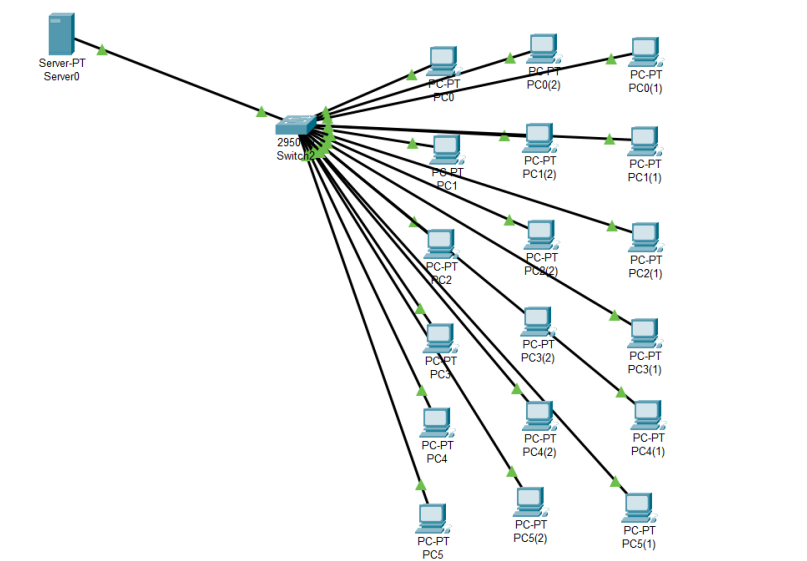
Hybrid topology



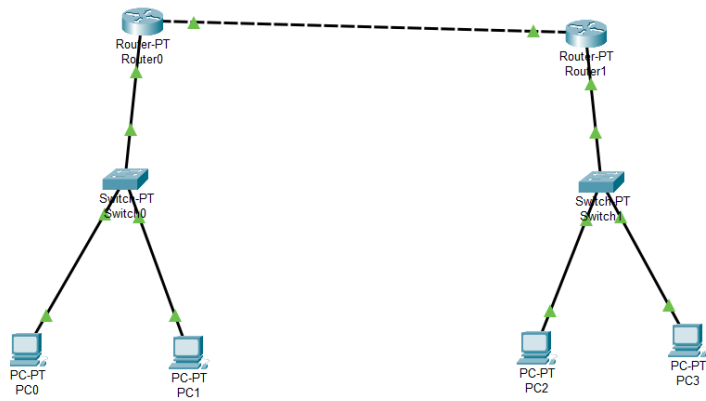
Data link layer traffic simulation of ARP



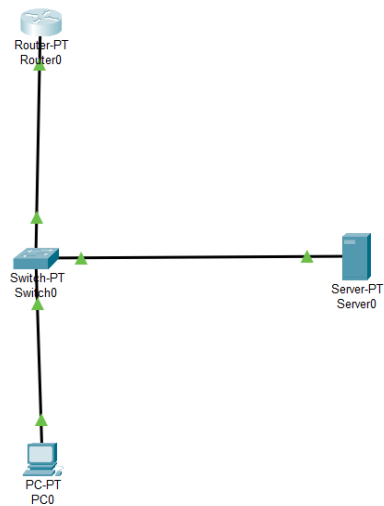
Computer lab



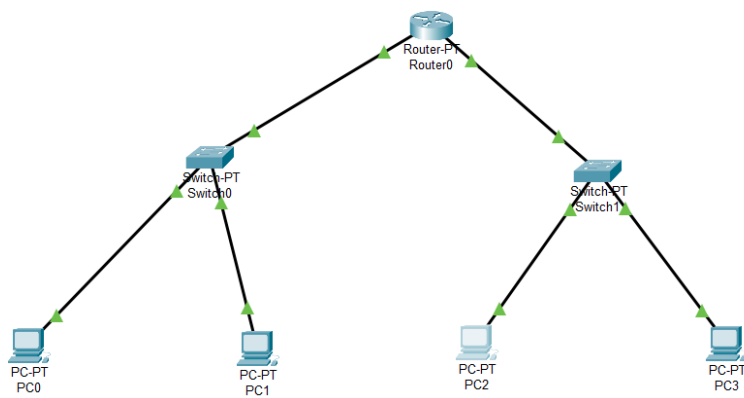
Static router



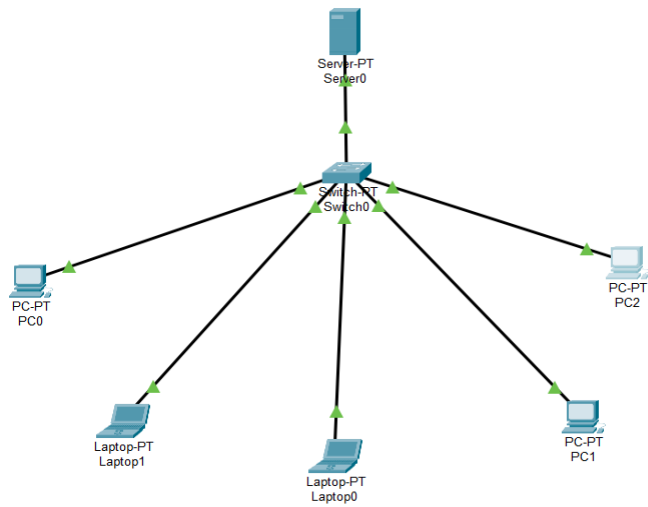
TCP



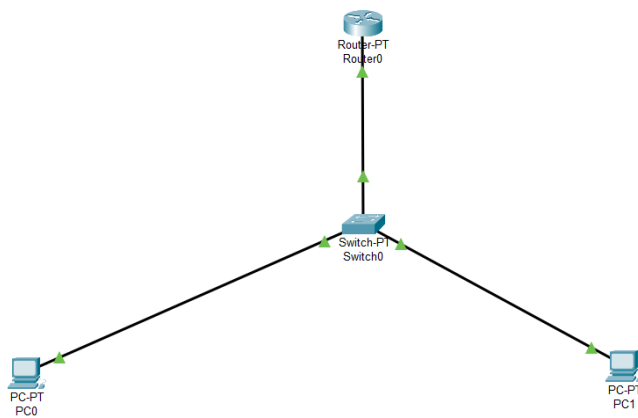
Subnetting



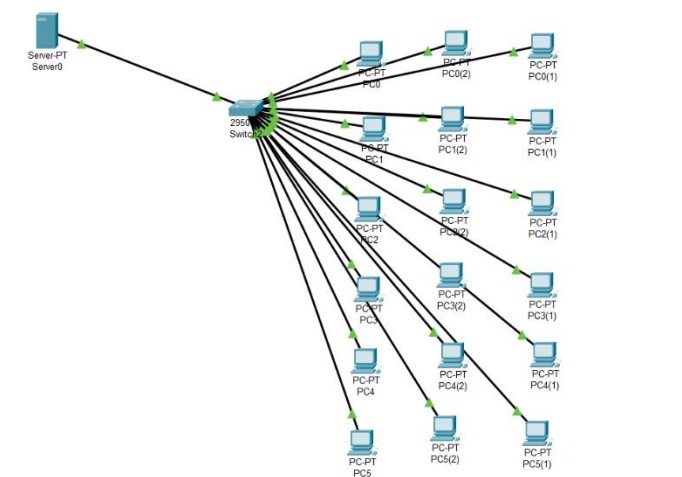
DHCP

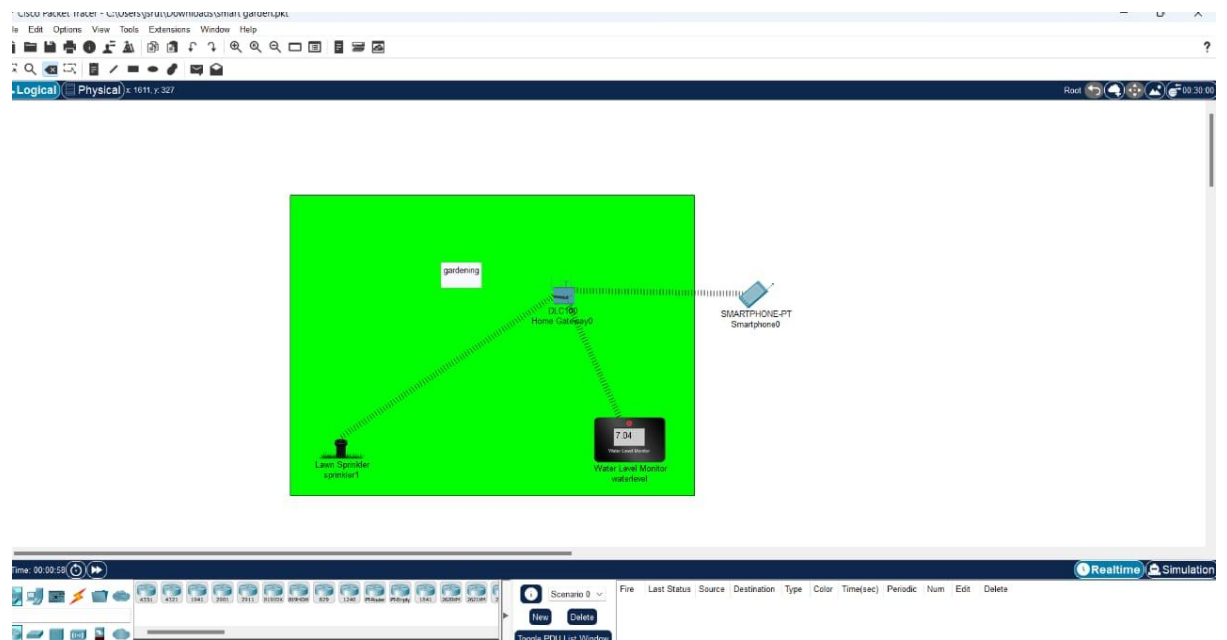
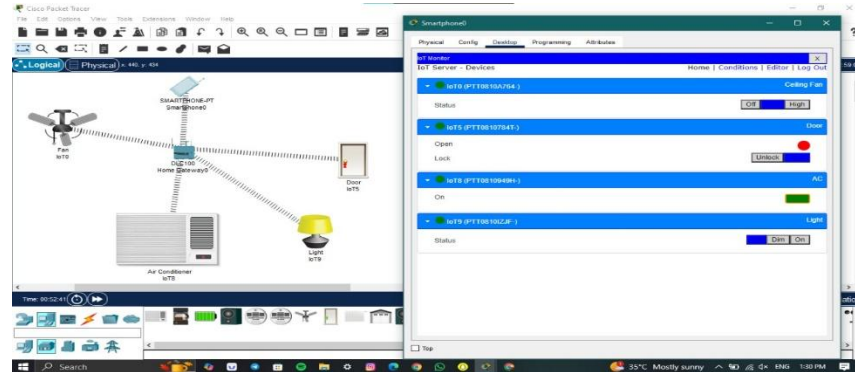
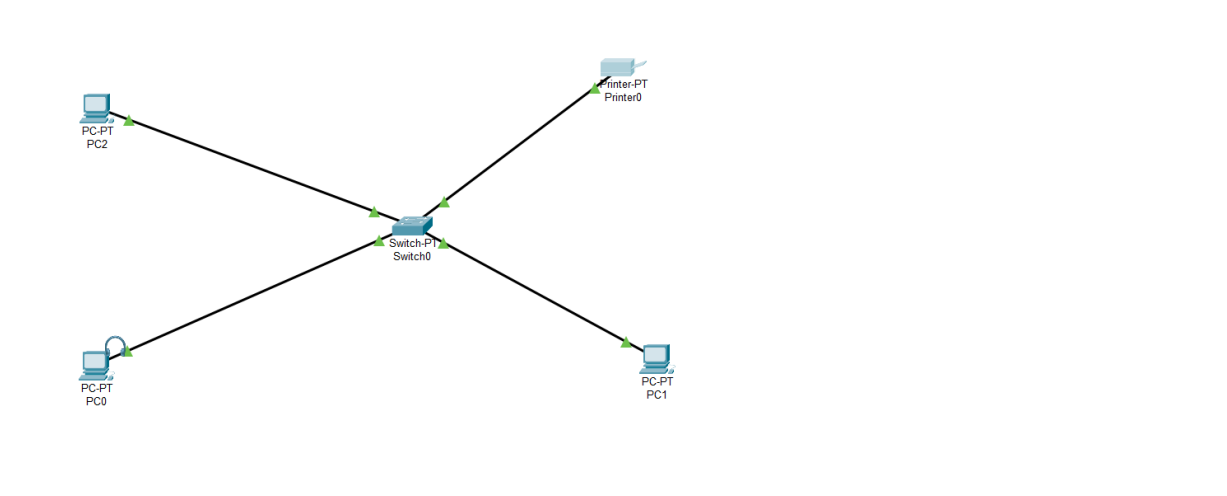


FIREWALL

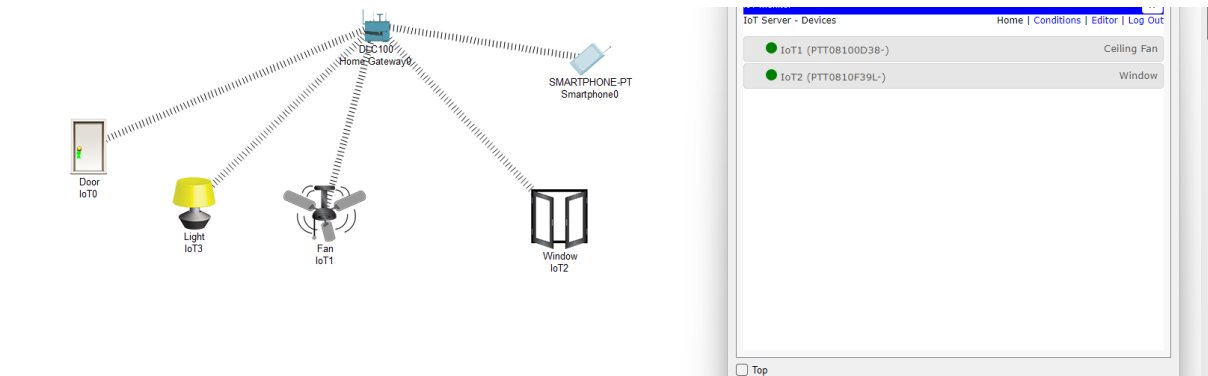


COMPUTER LAB

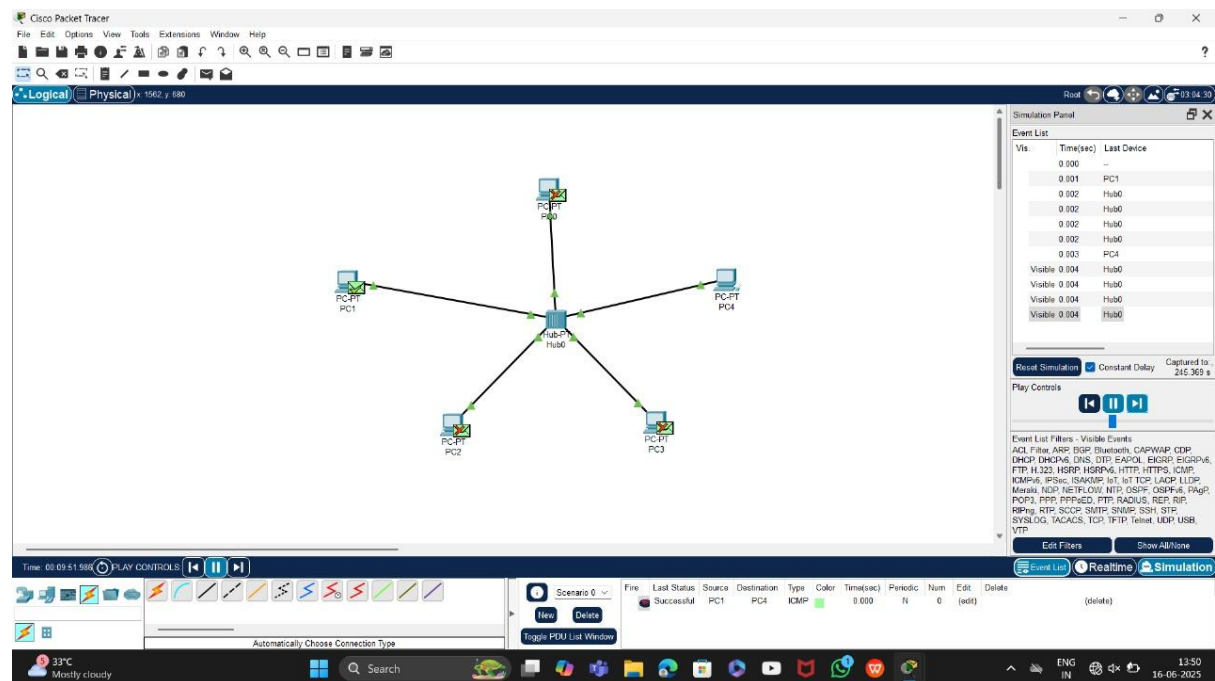




SMART NETWORK DEVICES



CSMA/CD



HTTP

The image shows a Wireshark packet capture of an HTTP 400 Bad Request. The packet list on the left shows a packet of length 558 bytes. The packet details pane on the right shows the following structure:

- Internet Protocol Version 4, Src: 188.243.34.67, Dst: 192.168.1.7
- Transmission Control Protocol, Src Port: 26801, Dst Port: 54262, Seq: 1, Ack: 216, Len: 558
- Hypertext Transfer Protocol
 - HTTP/1.1 400 Bad Request (text/html)
 - Date: Thu, 10 Jul 2025 14:49:45 GMT
 - Server: exchange.prodigidcp.com
 - X-Frame-Options: SAMEORIGIN
 - Content-Security-Policy: script-src 'self' 'unsafe-inline' 'unsafe-eval'; object-src 'self'; worker-src 'self' blob:; r/n
 - Content-Length: 226
 - Connection: close
 - Content-Type: text/html; charset=iso-8859-1

The packet bytes pane at the bottom shows the raw data of the packet, including the HTTP status bar and the body of the response.

Arp

The image shows a Wireshark packet capture of an ARP request. The packet list on the left shows a packet of length 68 bytes. The packet details pane on the right shows the following structure:

- Frame 303: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface \Device\NPF_{871D9A99-2B94-4843-B7A9-C5D1676B45AF}, id 0
- Ethernet II, Src: c0:35:32:7e:eb:0f (c0:35:32:7e:eb:0f), Dst: f8:0d:a9:f4:01:8b (f8:0d:a9:f4:01:8b)
- Address Resolution Protocol (request)
 - Hardware type: Ethernet (1)
 - Protocol type: IPv4 (0x0800)
 - Hardware size: 6
 - Protocol size: 4
 - Opcode: request (1)
 - Sender MAC address: c0:35:32:7e:eb:0f (c0:35:32:7e:eb:0f)
 - Sender IP address: 192.168.1.7
 - Target MAC address: f8:0d:a9:f4:01:8b (f8:0d:a9:f4:01:8b)
 - Target IP address: 192.168.1.1

The packet bytes pane at the bottom shows the raw data of the packet, including the Ethernet II header and the ARP request body.

TCP

The image shows a Wireshark packet capture of a TCP connection. The packet list on the left shows several TCP segments, including a Reset (RST) packet with sequence number 58120 and length 68. The packet details pane on the right shows the structure of the TCP segment, including the source and destination ports (57882 and 6881), the sequence number (1576603141), and the acknowledgment number (0). The packet bytes pane at the bottom shows the raw data of the packet, including the TCP header and the RST flag.

No.	Time	Source	Destination	Protocol	Length	Info
11609	286.178178	2401:4900:889f:baa9... 2a03:ec00:b9a9:3209	2a03:ec00:b9a9:3209	TCP	142	Handshake
11610	286.336152	2401:4900:889f:baa9... 2a03:ec00:b9a9:3209	2a03:ec00:b9a9:3209	TCP	142	[TCP Retransmission] Seq=1 Ack=1 Win=65535 Len=68
11611	286.414885	2401:4900:889f:baa9... 2a03:ec00:b1a0:2ba6	2a03:ec00:b1a0:2ba6	TCP	142	[TCP Retransmission] Seq=1 Ack=1 Win=65535 Len=68
11612	286.508463	192.168.1.7	213.243.53.19	TCP	422	[TCP Retransmission] Seq=1 Ack=1 Win=65535 Len=368
11613	286.634019	2401:4900:889f:baa9... 2a03:ec00:b17b:e8c7	2a03:ec00:b17b:e8c7	TCP	74	58120 → 16254 [RST, ACK] Seq=69 Ack=1 Win=0 Len=0
11614	286.758061	2401:4900:889f:baa9... 2a03:ec00:b979:55d6	2a03:ec00:b979:55d6	TCP	142	[TCP Retransmission] Seq=1 Ack=1 Win=65535 Len=68
11615	286.758154	2401:4900:889f:baa9... 2a03:ec00:b9a9:3209	2a03:ec00:b9a9:3209	TCP	142	[TCP Retransmission] Seq=1 Ack=1 Win=65535 Len=68
11616	286.804049	192.168.1.7	112.198.27.2	TCP	509	[TCP Retransmission] Seq=1 Ack=1 Win=65535 Len=455
11620	286.938998	2401:4900:889f:baa9... 2a03:ec00:b17b:e8c7	2a03:ec00:b17b:e8c7	TCP	142	[TCP Retransmission] Seq=1 Ack=1 Win=65535 Len=68
11622	287.069656	2401:4900:889f:baa9... 2a03:ec00:b9a8:2b27	2a03:ec00:b9a8:2b27	TCP	142	[TCP Retransmission] Seq=1 Ack=1 Win=65535 Len=68
11624	287.194226	2401:4900:889f:baa9... 2a03:ec00:b1a0:2ba6	2a03:ec00:b1a0:2ba6	TCP	142	[TCP Retransmission] Seq=1 Ack=1 Win=65535 Len=68

Frame 302: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface \Device\NPF_{871D9A99-2B94-4843-B7A9-CD01676B45AF}, id 0
> Ethernet II, Src: c0:35:32:7e:eb:0f (c0:35:32:7e:eb:0f), Dst: f8:0d:a9:f4:01:8b (f8:0d:a9:f4:01:8b)
> Internet Protocol Version 4, Src: 192.168.1.7, Dst: 141.95.160.248
v Transmission Control Protocol, Src Port: 57882, Dst Port: 6881, Seq: 0, Len: 0
Source Port: 57882
Destination Port: 6881
[Stream index: 16]
[Conversation completeness: Complete, WITH_DATA (31)]
[TCP Segment Len: 0]
Sequence Number: 0 (relative sequence number)
Sequence Number (raw): 1576603141
[Next Sequence Number: 1 (relative sequence number)]
Acknowledgment Number: 0
0000 f8 0d a9 f4 01 8b c0 35 32 7e eb 0f 08 00 45 005 2-----E
0010 00 00 cf cd a0 00 00 06 3a ef c0 a8 01 07 8d 5f@ - 2-----
0020 a0 f8 e2 1a 1a e1 5d f9 0e 05 00 00 00 00 02]:
0030 ff ff 16 0f 00 00 02 04 05 b4 01 03 03 08 01 01
0040 04 02 ..

BIT STUFFING

The image shows a C program named `main.c` that demonstrates bit stuffing. The program prompts the user to enter data bits, which are then stuffed with zeros to ensure the total length is a multiple of 5. The output shows the original data bits, the stuffed data bits, and a success message.

```
main.c
#include<stdio.h>
#include<string.h>
int main()
{
    int i=0,count=0;
    char databits[80];
    printf("Enter Data Bits: ");
    scanf("%s",databits);
    printf("\nData Bits After Bit stuffing: ");
    for(i=0; i<strlen(databits); i++)
    {
        if(databits[i]!='1')
            count++;
        else
            count=0;
        printf("%c",databits[i]);
        if(count==5)
        {
            printf("0");
            count=0;
        }
    }
    return 0 ;
}
```

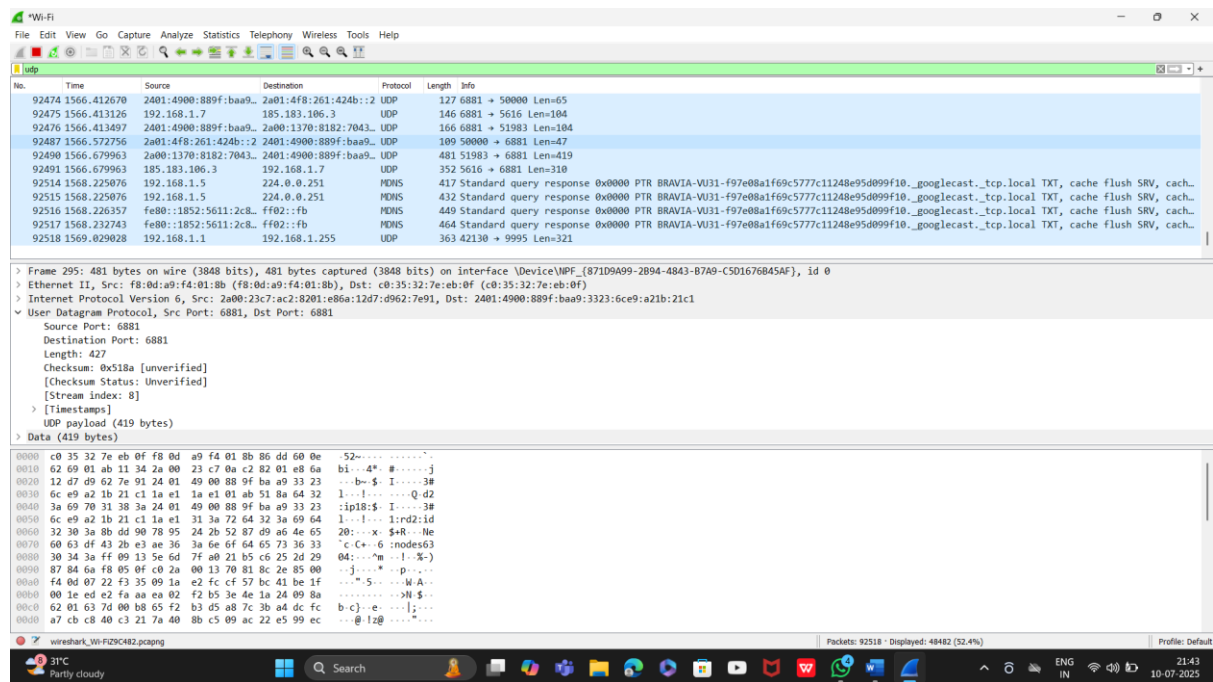
Output

Enter Data Bits: 101010111111

Data Bits After Bit stuffing: 1010101111101

=== Code Execution Successful ===

UDP



The image shows a Wireshark packet capture of UDP traffic. The top pane displays a list of packets, with packet 295 selected. The middle pane shows the details of the selected packet, including the Ethernet II header, Internet Protocol Version 6 header, and User Datagram Protocol header. The bottom pane shows the raw packet data in hexadecimal and ASCII.

Packet 295: 481 bytes on wire (3848 bits), 481 bytes captured (3848 bits) on interface \Device\NPF_{871D9A99-2B94-4843-B7A9-C5D1676B45AF}, id 0

Ethernet II, Src: f8:0d:a9:f4:01:8b (f8:0d:a9:f4:01:8b), Dst: c0:35:32:7e:eb:0f (c0:35:32:7e:eb:0f)

Internet Protocol Version 6, Src: 2a00:23c7:ac2:8201:e86a:1d7:d962:7e91, Dst: 2401:4900:889f:baa9:3323:6ce9:a21b:21c1

User Datagram Protocol, Src Port: 6881, Dst Port: 6881

Source Port: 6881

Destination Port: 6881

Length: 427

Checksum: 0x518a [unverified]

[Checksum Status: Unverified]

[Stream index: 8]

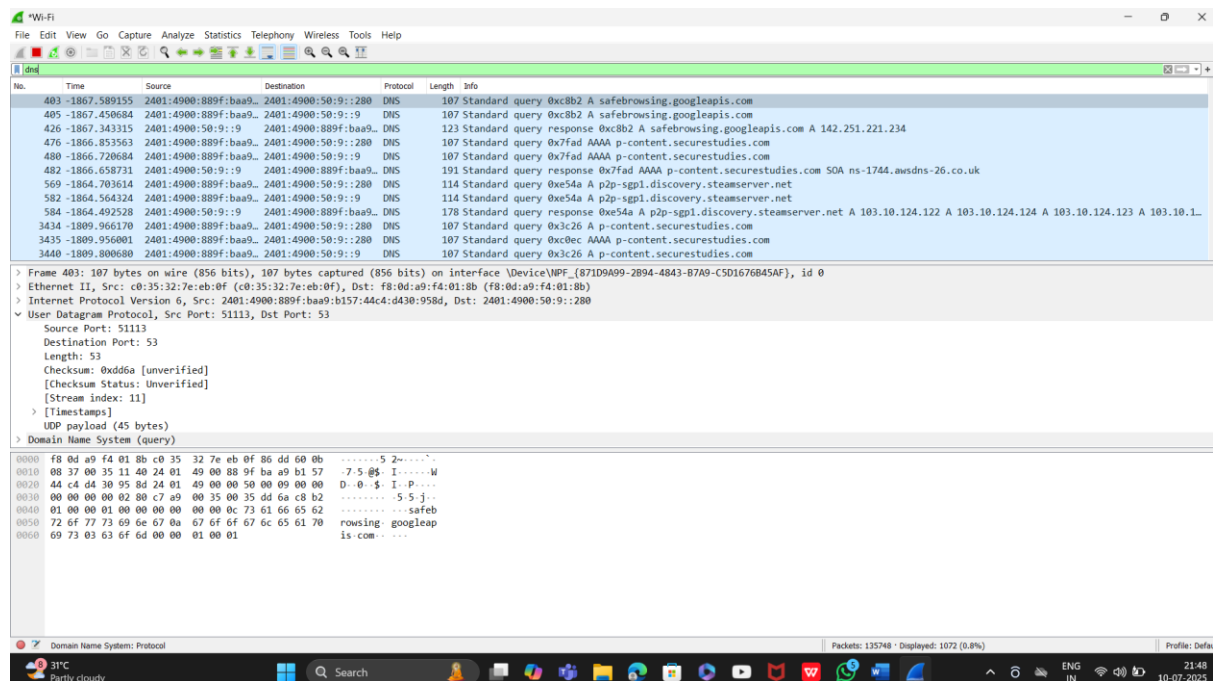
[Timestamps]

UDP payload (419 bytes)

Data (419 bytes)

0000 c0 35 32 7e eb 0f f8 0d a9 f4 01 8b 86 dd 60 0e 52.....
0010 62 69 01 ab 11 34 2a 00 23 c7 0a c2 82 01 e8 6a bi...4*#.....
0020 12 d7 d9 62 7e 91 24 01 49 00 88 9f ba a9 33 23 ---b-\$I...3#
0030 6c e9 a2 1b 21 c1 1a e1 1a e1 01 ab 51 8a 64 32 1---l---Q d2
0040 3a 69 70 31 38 3a 24 01 49 00 88 9f ba a9 33 23 :ip18:\$I...3#
0050 6c e9 a2 1b 21 c1 1a e1 31 3a 72 64 32 3a 69 64 1---l---1rd2id
0060 32 30 3a 8b dd 90 78 95 24 2b 52 87 d9 a6 4e 65 20---x-\$R...Ne
0070 60 63 df 43 2b e3 ae 36 3a 6e 6f 64 65 73 36 33 "c(C+...6 :nodes63
0080 30 34 3a ff 09 13 5e 6d 7f a0 21 b5 c6 25 2d 29 04---m...l-(\$-)
0090 87 84 6a f8 05 0f c0 2a 00 13 70 81 8c 2e 85 00 -j...-p...
00a0 f4 0d 07 22 f3 35 09 1a e2 fc cf 57 bc 41 be 1f ---*5---W.A...
00b0 00 1e ed e2 fa aa ea 02 f2 b5 3e 4e 1a 24 09 8a>N.\$...
00c0 62 01 63 7d 00 b8 65 f2 b3 d5 a8 7c 3b a4 dc fc b c)---e---|;...
00d0 a7 cb c8 c0 c4 21 7a 80 8b c5 09 ac 22 e5 99 ec ---@-1z@.....

DNS



The image shows a Wireshark packet capture of DNS traffic. The top pane displays a list of packets, with packet 403 selected. The middle pane shows the details of the selected packet, including the Ethernet II header, Internet Protocol Version 6 header, and User Datagram Protocol header. The bottom pane shows the raw packet data in hexadecimal and ASCII.

Packet 403: 107 bytes on wire (856 bits), 107 bytes captured (856 bits) on interface \Device\NPF_{871D9A99-2B94-4843-B7A9-C5D1676B45AF}, id 0

Ethernet II, Src: c0:35:32:7e:eb:0f (c0:35:32:7e:eb:0f), Dst: f8:0d:a9:f4:01:8b (f8:0d:a9:f4:01:8b)

Internet Protocol Version 6, Src: 2401:4900:889f:baa9:b157:44c4:d430:958d, Dst: 2401:4900:50:9::280

User Datagram Protocol, Src Port: 51113, Dst Port: 53

Source Port: 51113

Destination Port: 53

Length: 53

Checksum: 0xdd6a [unverified]

[Checksum Status: Unverified]

[Stream index: 11]

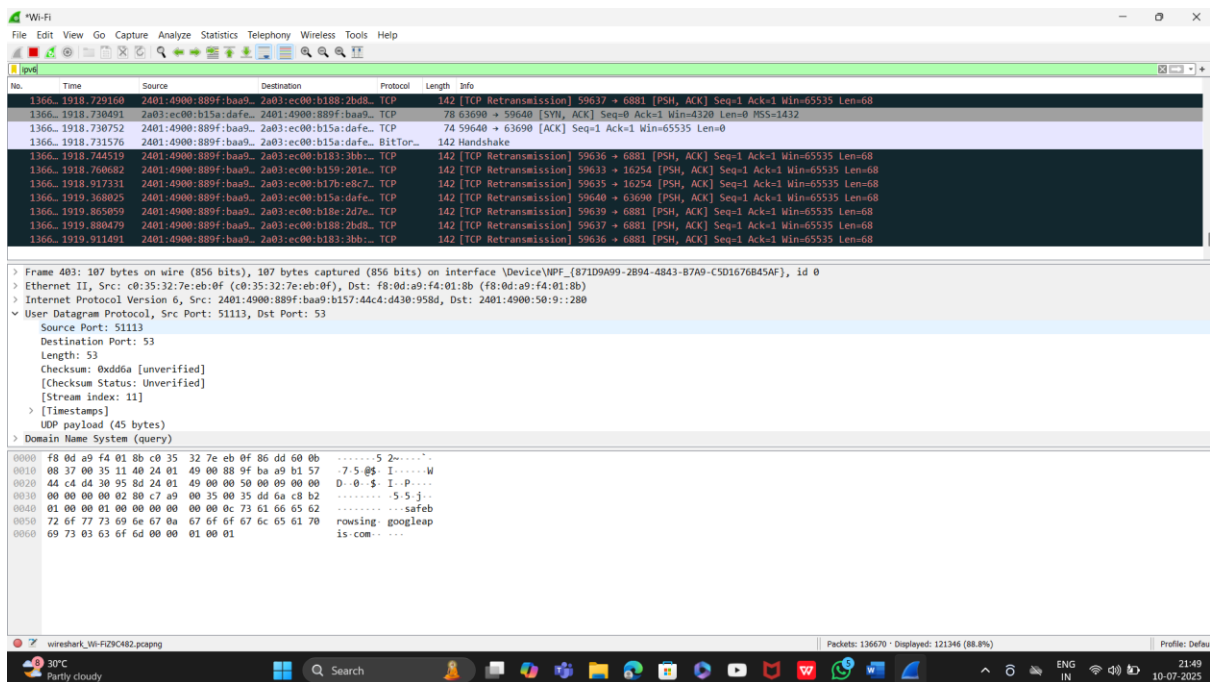
[Timestamps]

UDP payload (45 bytes)

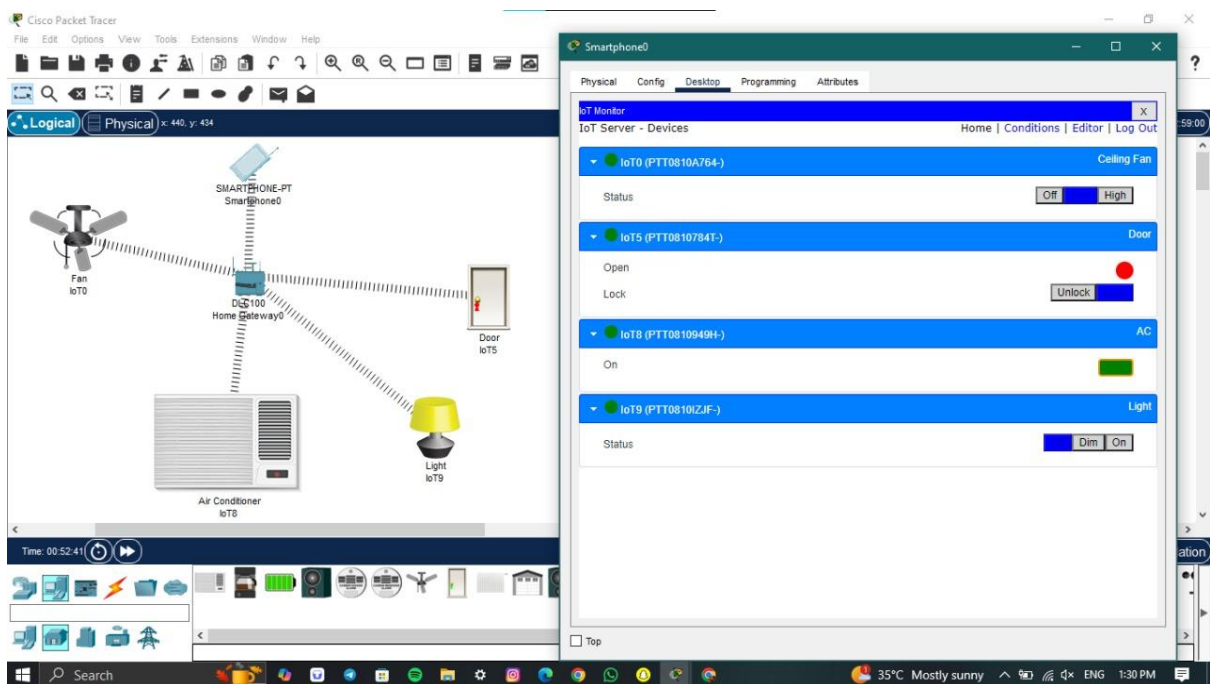
Domain Name System (query)

0000 f8 0d a9 f4 01 8b c0 35 32 7e eb 0f 86 dd 60 0b5 2.....
0010 08 37 00 35 11 40 24 01 49 00 00 9f ba a9 81 57 -7.5@-I...-W
0020 44 c4 d4 30 95 8d 24 01 49 00 00 50 00 09 00 00 D...0-\$I...P...
0030 00 00 00 00 02 80 c7 a9 00 35 00 35 dd 6a c8 b25.5-j..
0040 01 00 00 01 00 00 00 00 00 00 0c 73 61 66 65 62safeb
0050 72 6f 77 73 69 6e 67 0a 67 6f 6f 67 6c 65 61 70 rowing: googleap
0060 69 73 63 63 6f 6d 00 00 01 00 01 is-com-...

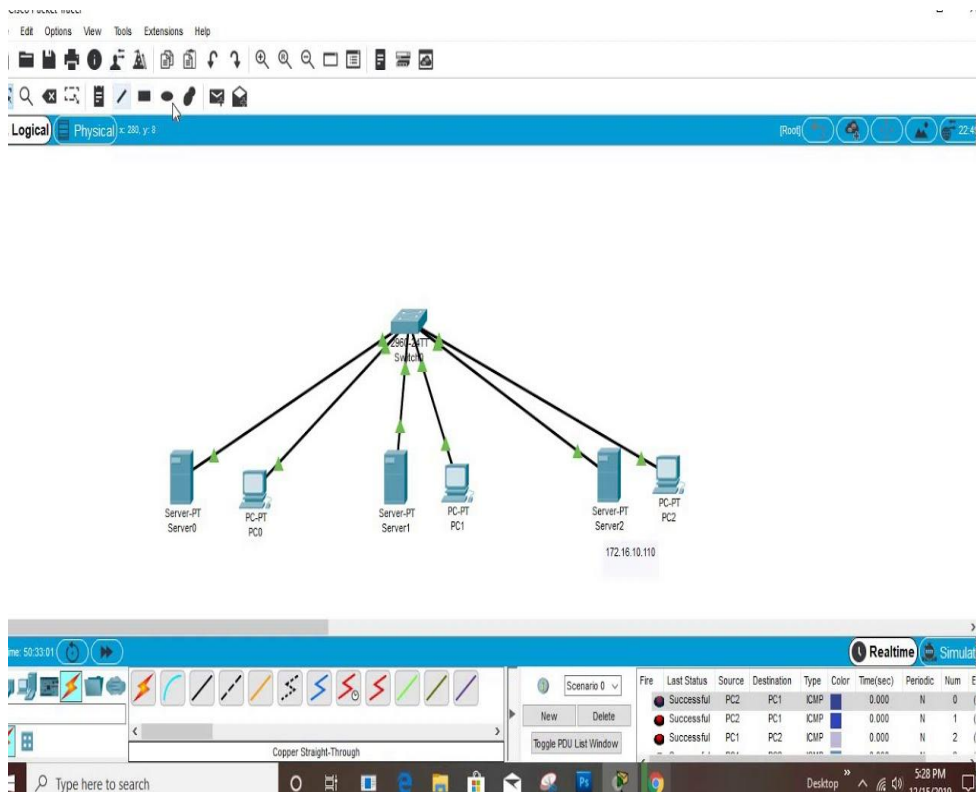
IPV6



Control of fan,light



WLAN



AAA server

