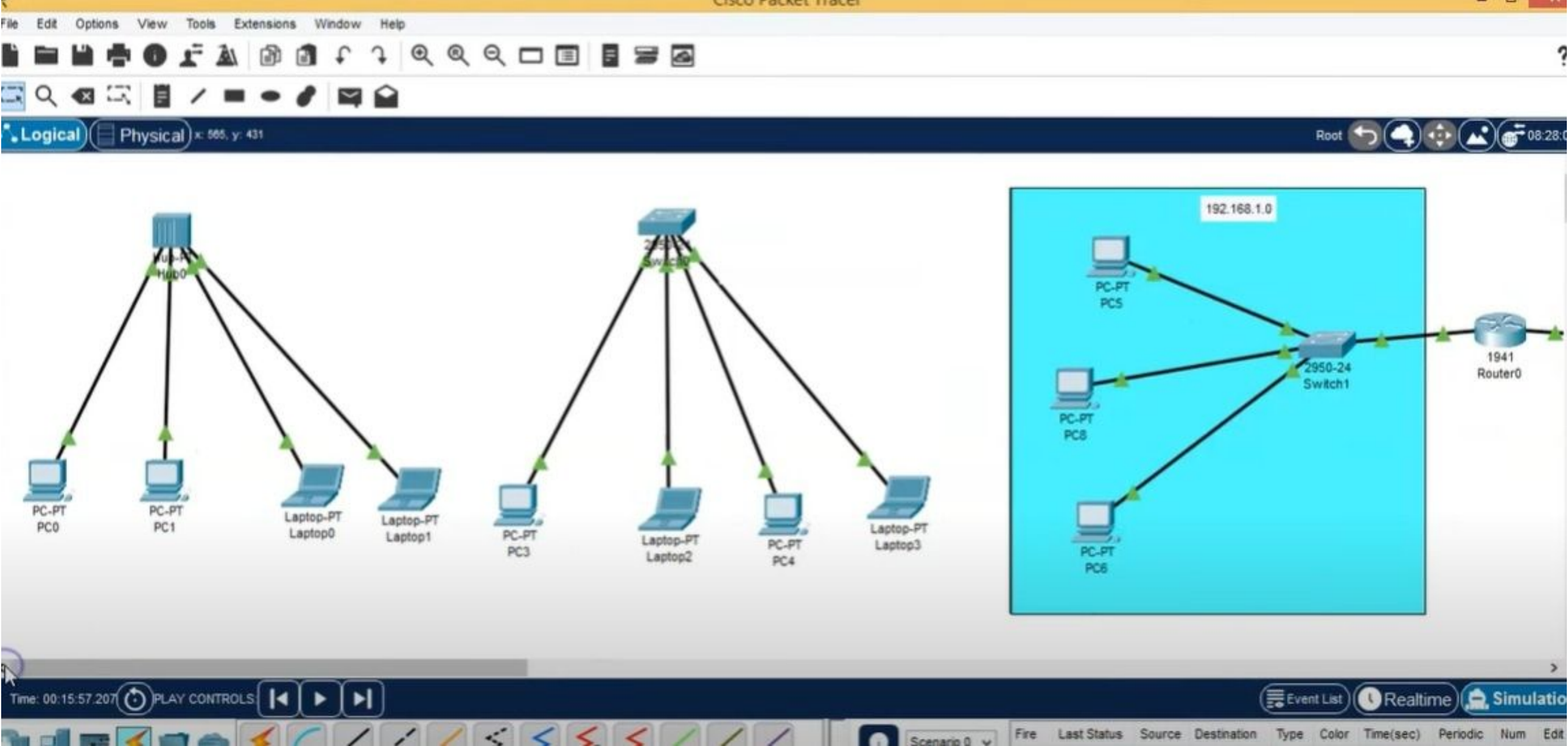
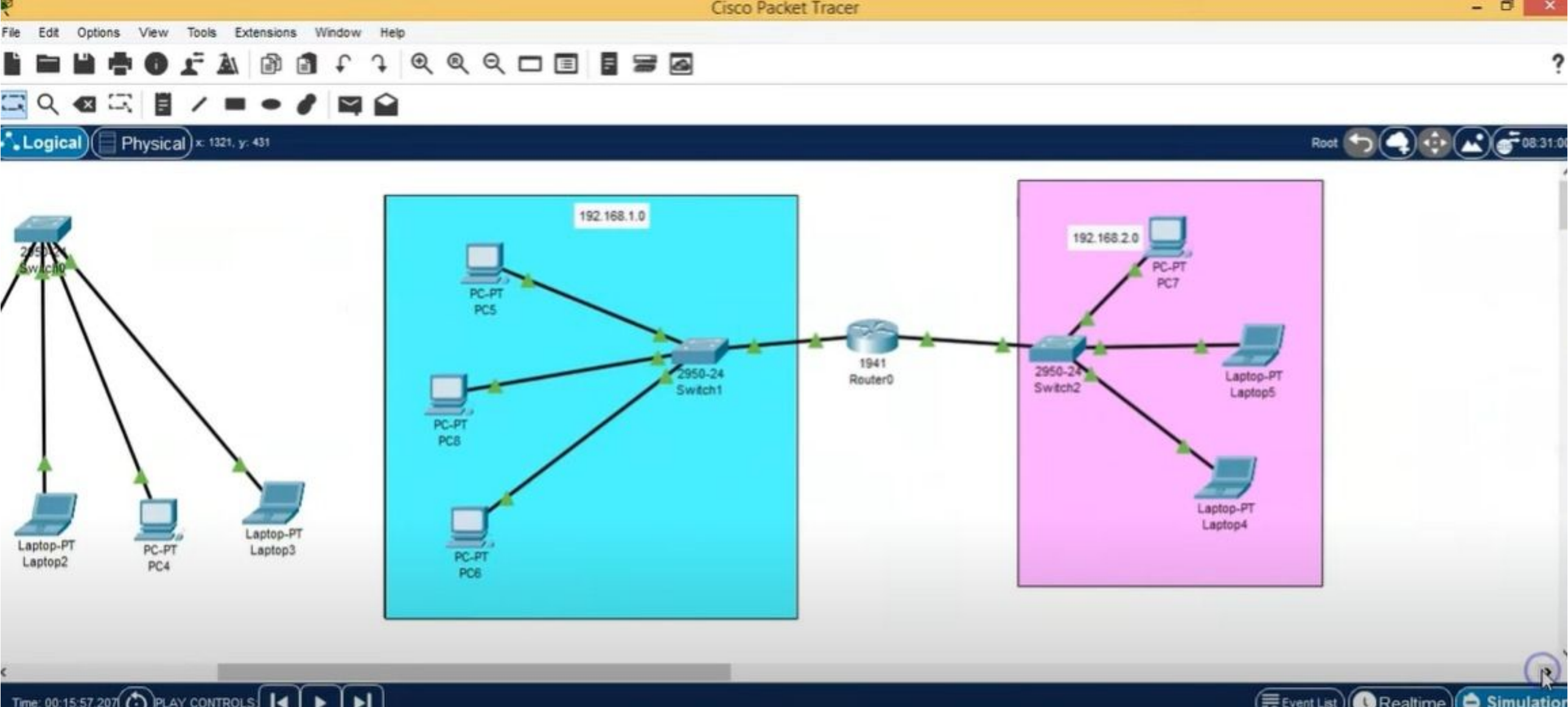

Cisco Packet Tracer
File Edit Options View Tools Extensions Window Help
Logical
Physical
x: 565, y: 431
Root
08:28:0
PC-PT PC0
PC-PT PC1
Laptop-PT Laptop0
Laptop-PT Laptop1
PC-PT PC3
Laptop-PT Laptop2
PC-PT PC4
Laptop-PT Laptop3
192.168.1.0
PC-PT PC5
PC-PT PC8
PC-PT PC6
2950-24 Switch1
1941 Router0
Time: 00:15:57.207
PLAY CONTROLS:
Event List
Realtime
Simulatio
Fire Last Status Source Destination Type Color Time(sec) Periodic Num Edit

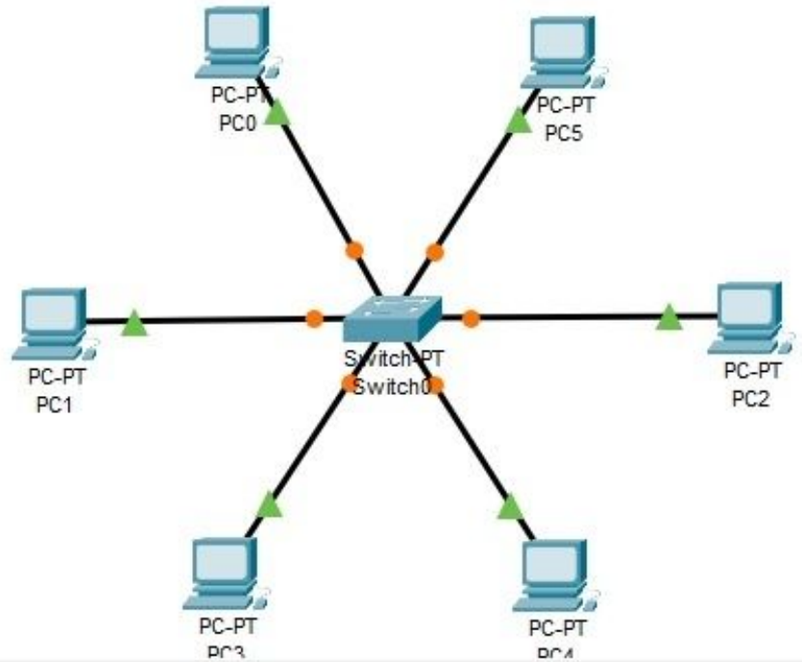

Cisco Packet Tracer
File Edit Options View Tools Extensions Window Help
Logical
Physical
x: 1321, y: 431
Root
08:31:00
192.168.1.0
192.168.2.0
PC-PT PC5
PC-PT PC8
PC-PT PC6
2950-24 Switch1
1941 Router0
2950-24 Switch2
PC-PT PC7
Laptop-PT Laptop5
Laptop-PT Laptop4
Laptop-PT Laptop2
PC-PT PC4
Laptop-PT Laptop3
Time: 00:15:57.207
PLAY CONTROLS:
Event List
Realtime
Simulation

File Edit Options View Tools Extensions Window Help

Logical Physical x: 153, y: 0

Logical Mode (Shift+L)

Root 00:02:30



Time: 00:00:04

Realtime Simulation

4331 4321 1941 2901 2911 819IOX 819HGW 829 1240 PT-Router PT-Empty 1841 2620

(Select a Device to Drag and Drop to the Workspace)

Scenario 0

New Delete

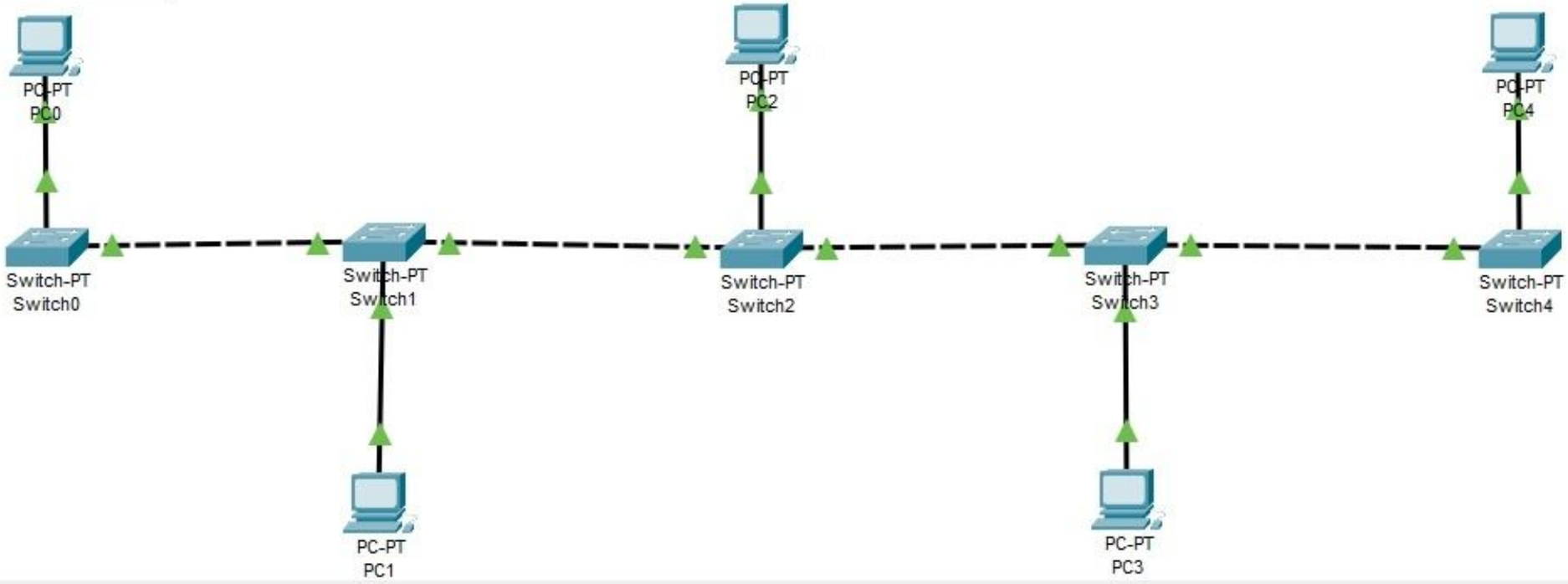
Toggle PDU List Window

Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic
	--	PC0	PC2	ICMP		0.000	N
	--	PC5	PC3	ICMP		0.000	N
	--	PC1	PC2	ICMP		0.000	N

File Edit Options View Tools Extensions Window Help

Logical Physical x: 6, y: 86 Root 08:03:30

Logical Mode (Shift+L)



Time: 00:15:55 Realtime Simulation

4331 4321 1941 2901 2911 819IOX 819HGW 829 1240 PT-Router PT-Empty 1841 2620

CGR1240

Scenario 0

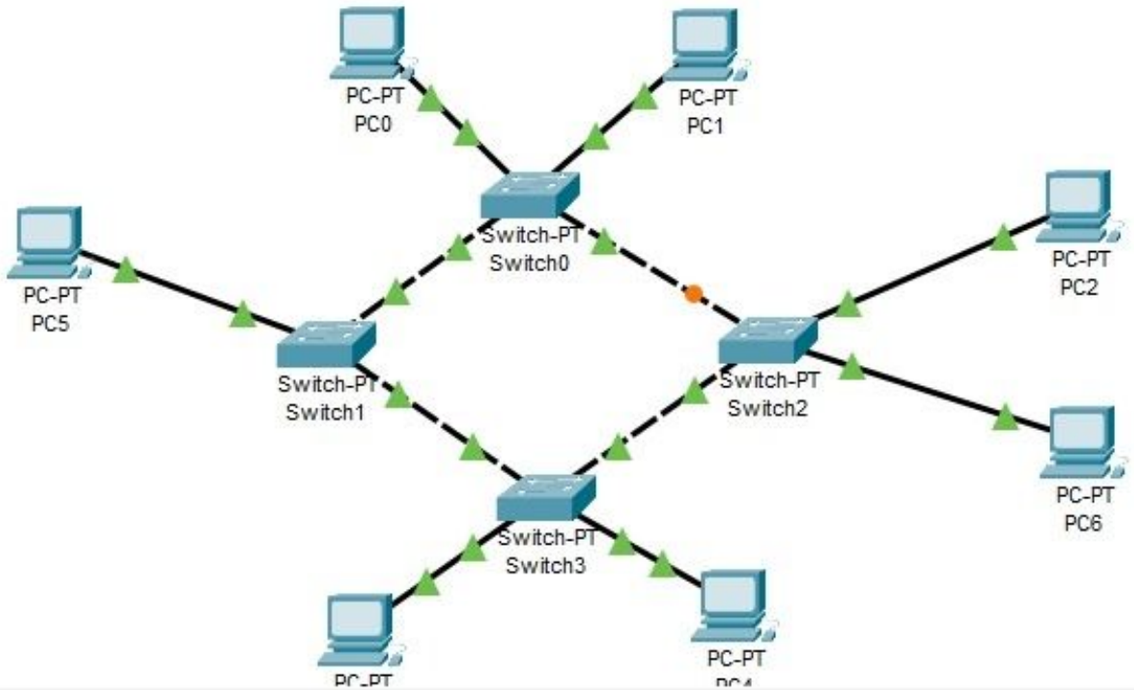
New Delete

Toggle PDU List Window

Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Peri
	--	PC0	PC1	ICMP		0.000	
	--	PC1	PC2	ICMP		0.000	
	--	PC2	PC3	ICMP		0.000	

File Edit Options View Tools Extensions Window Help

Logical | Physical | x: 1013, y: 403 | Root | 00:17:00



Time: 00:00:33 | Realtime | Simulation

PC | Laptop | Server | Meraki Server | Network Controller | Printer | IP Phone | VoIP Device | Phone | TV | Tablet | Smart Phone

PC-PT

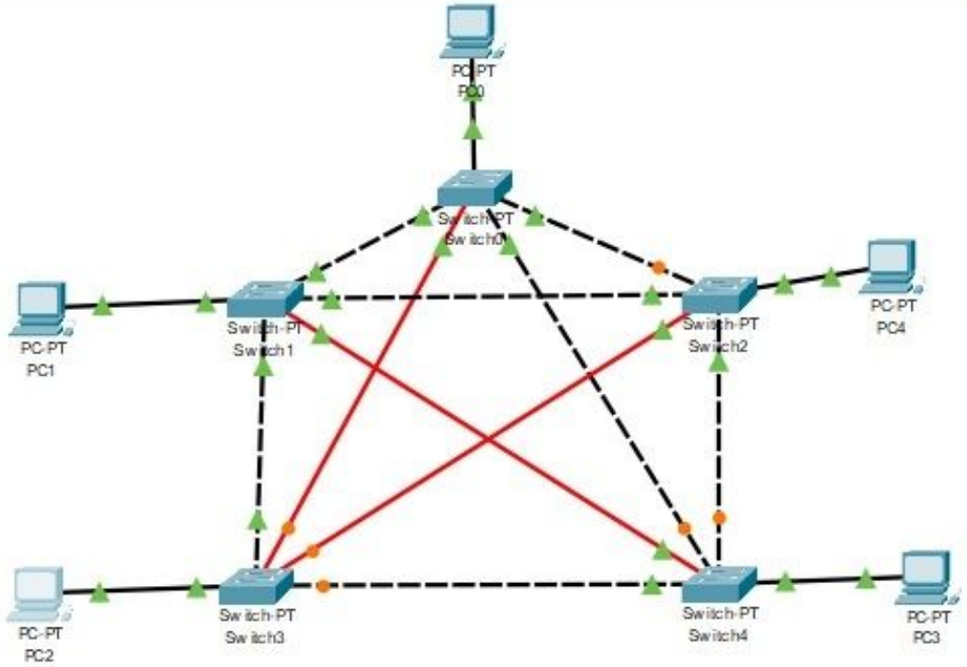
Scenario 0

New | Delete | Toggle PDU List Window

Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic
	--	PC0	PC3	ICMP		0.000	N
	--	PC1	PC4	ICMP		0.000	N
	--	PC2	PC4	ICMP		0.000	N

File Edit Options View Tools Extensions Window Help

Logical Physical x: 193, y: 136 Root 00:20:00



Time: 00:00:38

Realtime Simulation

4331 4321 1941 2901 2911 819IOX 819HGW 829 1240 PT-Router PT-Empty 1841 2620

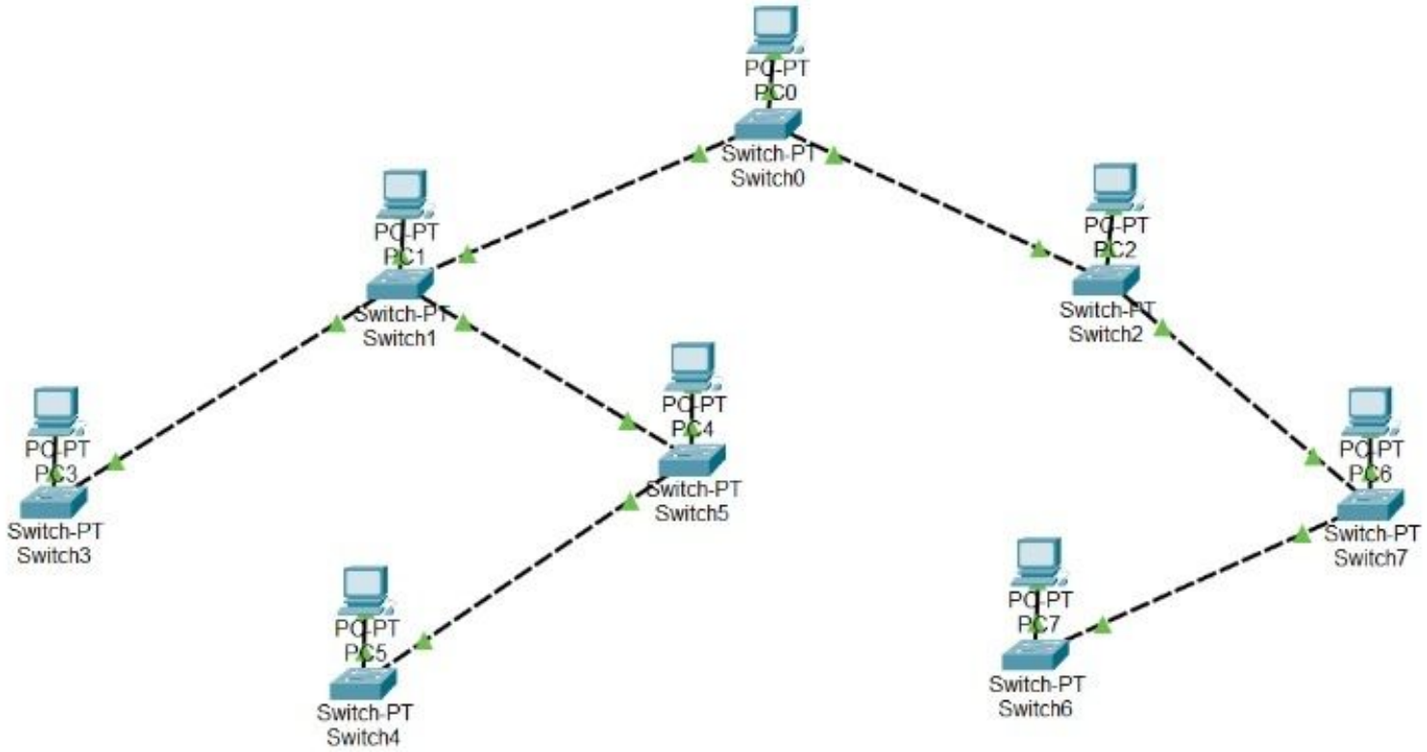
(Select a Device to Drag and Drop to the Workspace)

Scenario 0

New Delete

Toggle PDU List Window

Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Period
	--	PC1	PC4	ICMP		0.000	N
	--	PC0	PC2	ICMP		0.000	N
	--	PC0	PC3	ICMP		0.000	N



Time: 00:32:57

Realtime | Simulation

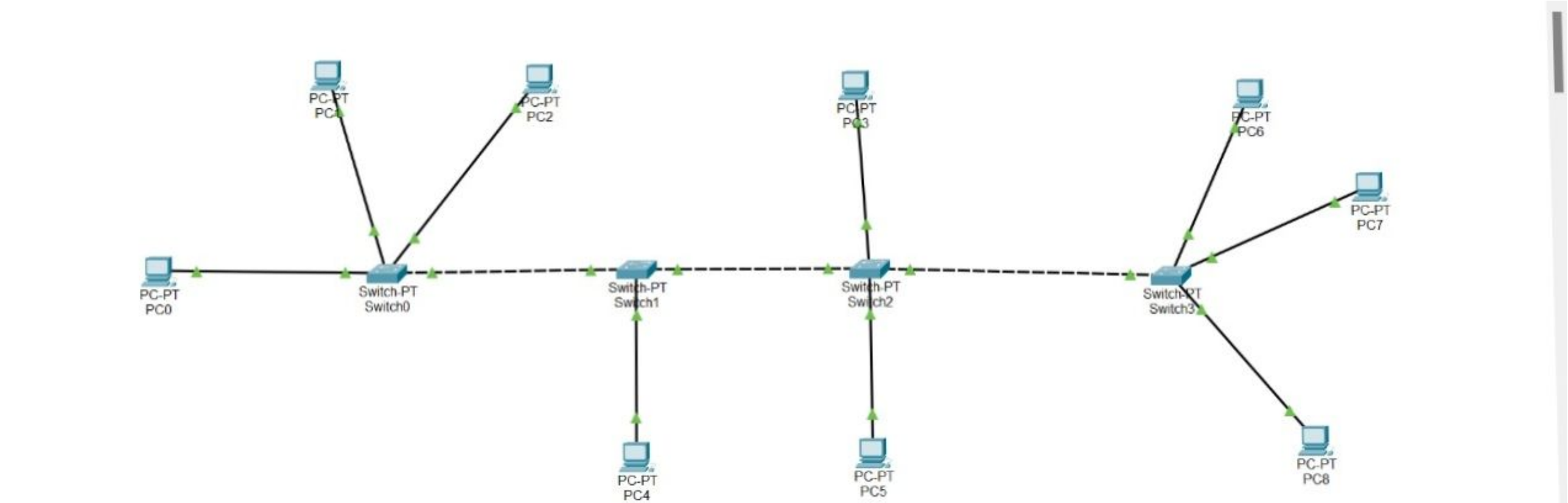
Scenario 0

New | Delete

Toggle PDU List Window

Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic	Num	Edit	Delete
	Successful	PC5	PC7	ICMP		0.000	N	4	(edit)	(delete)
	Successful	PC6	PC3	ICMP		0.000	N	5	(edit)	(delete)
	Successful	PC2	PC4	ICMP		0.000	N	6	(edit)	(delete)

Automatically Choose Connection Type



Time: 00:25:40

Realtime | Simulatio

Scenario 0

New | Delete

Toggle PDU List Window

Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic	Num	Edit	Delete
	Successful	PC7	PC0	ICMP		0.000	N	5	(edit)	(delete)
	Successful	PC8	PC6	ICMP		0.000	N	6	(edit)	(delete)
	Successful	PC7	PC1	ICMP		0.000	N	7	(edit)	(delete)

File Edit Options View Tools Extensions Window Help

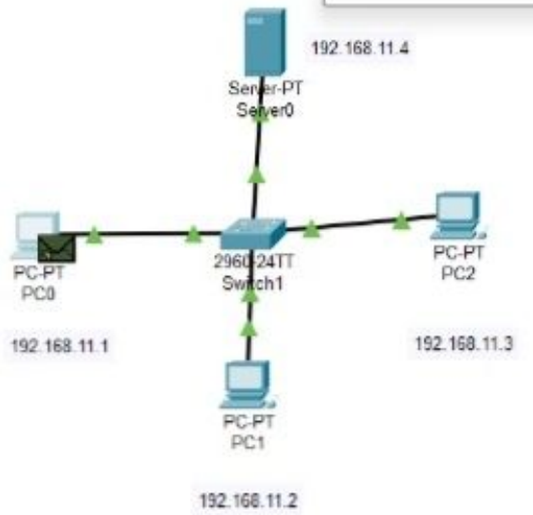
Logical | Physical x: 1227, y: 353 | Root | 07:39:00

ARP Table for Server0

IP Address	Hardware Address	Interface
192.16...	0000.0C...	FastEth...

ARP Table for PC0

IP Address	Hardware Address	Interface
192.16...	0003.E4...	FastEth...



Simulation Panel

Event List

Vis.	Time(sec)	Last Device	At Device
	0.000	--	PC0
	0.000	--	PC0
	0.001	PC0	Switch1
	0.002	Switch1	PC1
	0.002	Switch1	PC2
	0.002	Switch1	Server0
	0.003	Server0	Switch1
	0.004	Switch1	PC0
	0.004	--	PC0
	0.005	PC0	Switch1
	0.006	Switch1	Server0
	0.007	Server0	Switch1
Visible	0.008	Switch1	PC0

Reset Simulation | Constant Delay | Captured to: 0.008 s

Play Controls

Event List Filters - Visible Events

ACL Filter, ARP, BGP, Bluetooth, CAPWAP, CDP, DHCP, DHCPv6, DNS, DTP, EAPOL, EIGRP, EIGRPv6, FTP, H.323, HSRP, HSRPv6, HTTP, HTTPS, ICMP, ICMPv6, IPSec, ISAKMP, IoT, IoT TCP, LACP, LLDP, Meraki, NDP, NETFLOW, NTP, OSPF, OSPFv6, PAgP, POP3, PPP, PPPoED, PTP, RADIUS, REP, RIP, RIPng, RTP, SCCP, SMTP, SNMP, SSH, STP, SYSLOG, TACACS, TCP, TFTP, Telnet, UDP, USB, VTP

Edit Filters | Show All/None

Time: 00:08:01.383 | PLAY CONTROLS: | Event List | Realtime | Simulation

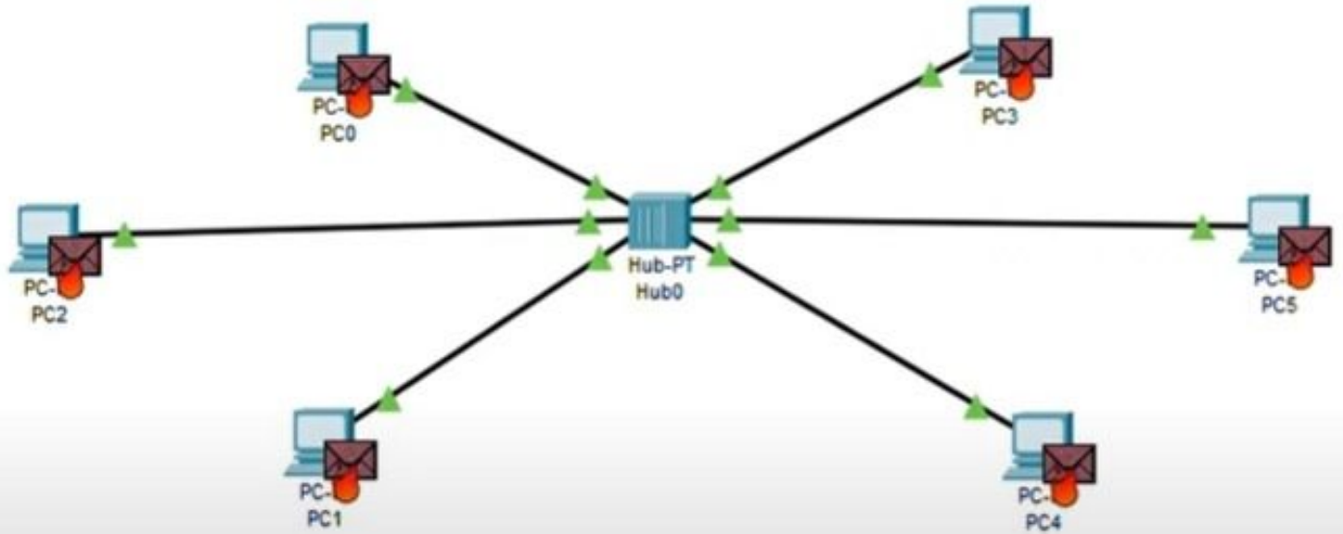
Copper Straight-Through

Scenario 0 | New | Delete | Toggle PDU List Window

Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic	Num	Edit	Delete

File Edit Options View Tools Extensions Window Help

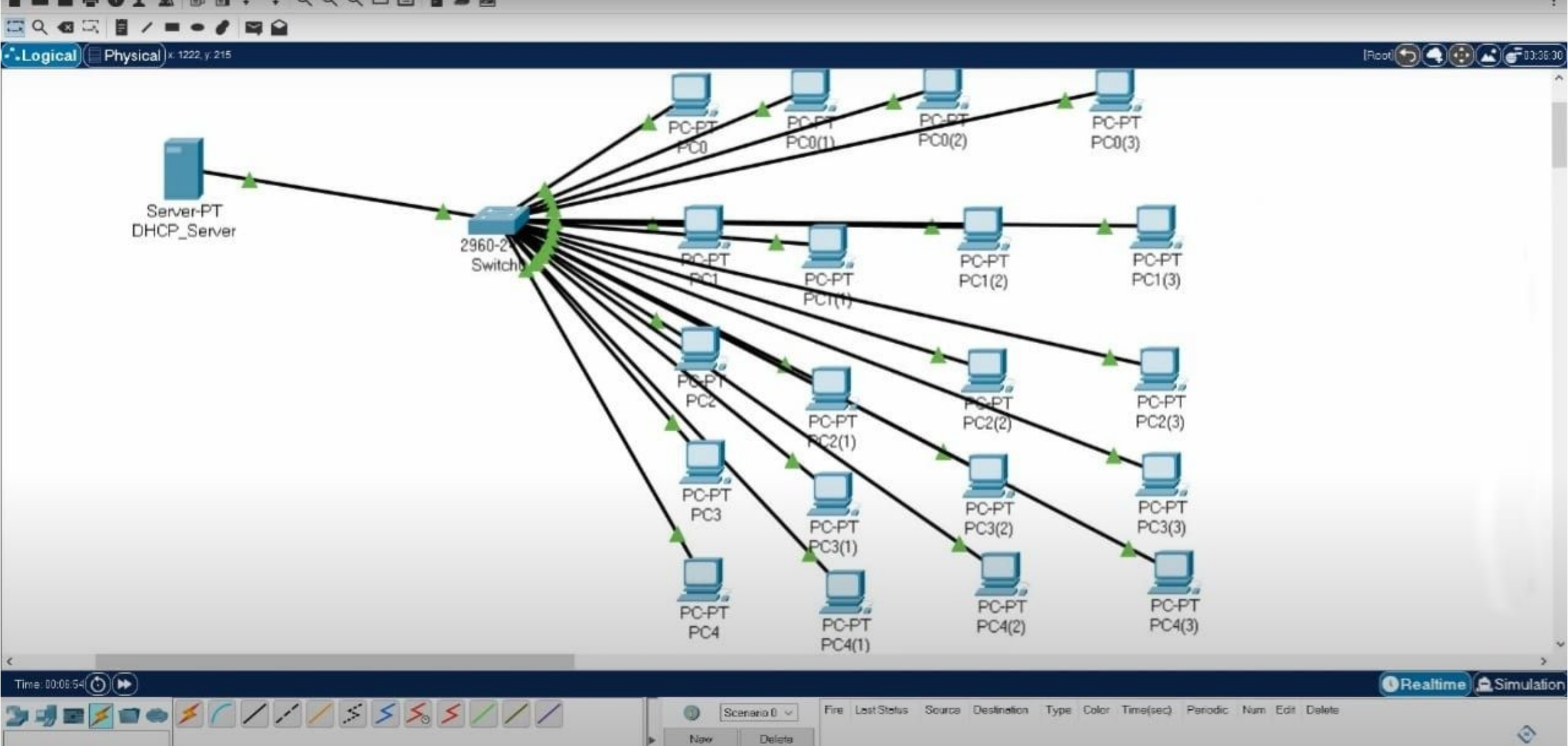
Logical Physical x: 1195, y: 420 Root 09:44:00

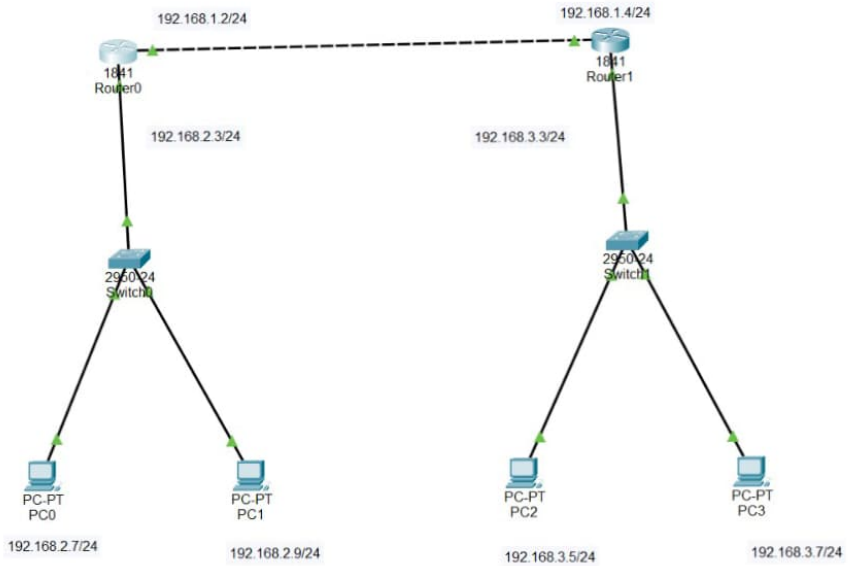


Time: 00:08:27.172 PLAY CONTROLS: Event List Realtime Simulation

Scenario 0

Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic	Num	Edit

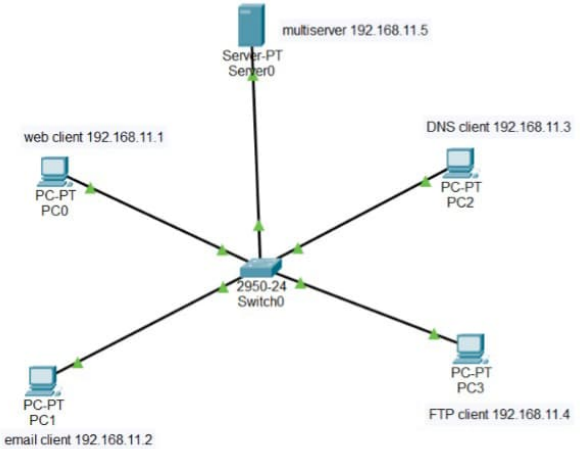

Logical
Physical
x: 1222, y: 215
Root
Server-PT
DHCP_Server
2960-2
Switch0
PC-PT
PC0
PC-PT
PC0(1)
PC-PT
PC0(2)
PC-PT
PC0(3)
PC-PT
PC1
PC-PT
PC1(1)
PC-PT
PC1(2)
PC-PT
PC1(3)
PC-PT
PC2
PC-PT
PC2(1)
PC-PT
PC2(2)
PC-PT
PC2(3)
PC-PT
PC3
PC-PT
PC3(1)
PC-PT
PC3(2)
PC-PT
PC3(3)
PC-PT
PC4
PC-PT
PC4(1)
PC-PT
PC4(2)
PC-PT
PC4(3)
Time: 00:06:54
Realtime
Simulation
Scenario 0
New
Delete
Fire
Last Status
Source
Destination
Type
Color
Time(sec)
Periodic
Num
Edit
Delete


192.168.1.2/24
192.168.1.4/24
1841
Router0
1841
Router1
192.168.2.3/24
192.168.3.3/24
2950-24
Switch0
2950-24
Switch1
PC-PT
PC0
PC-PT
PC1
PC-PT
PC2
PC-PT
PC3
192.168.2.7/24
192.168.2.9/24
192.168.3.5/24
192.168.3.7/24

Cisco Packet Tracer - C:\Users\Harshini Darshan\Cisco Packet Tracer 8.2.2\saves\exp12.pkt

File Edit Options View Tools Extensions Window Help

Logical Physical x: 1295, y: 399 Root 00:51:00

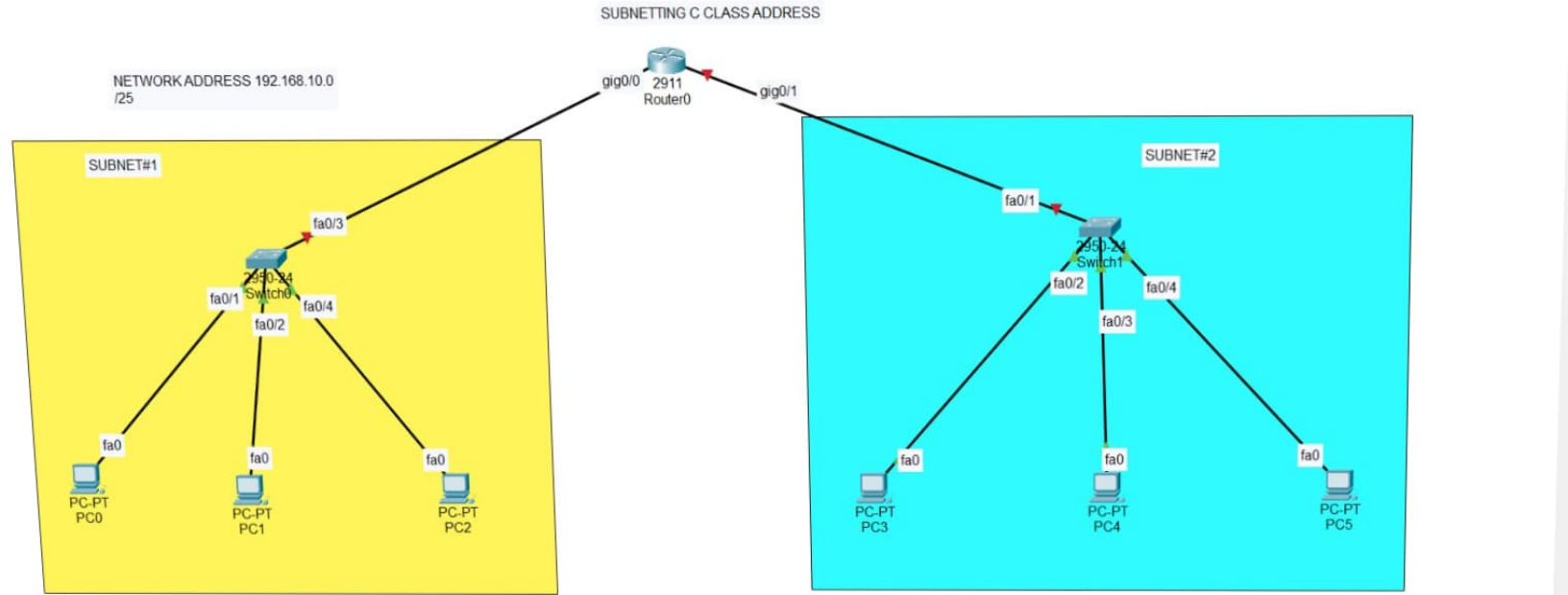


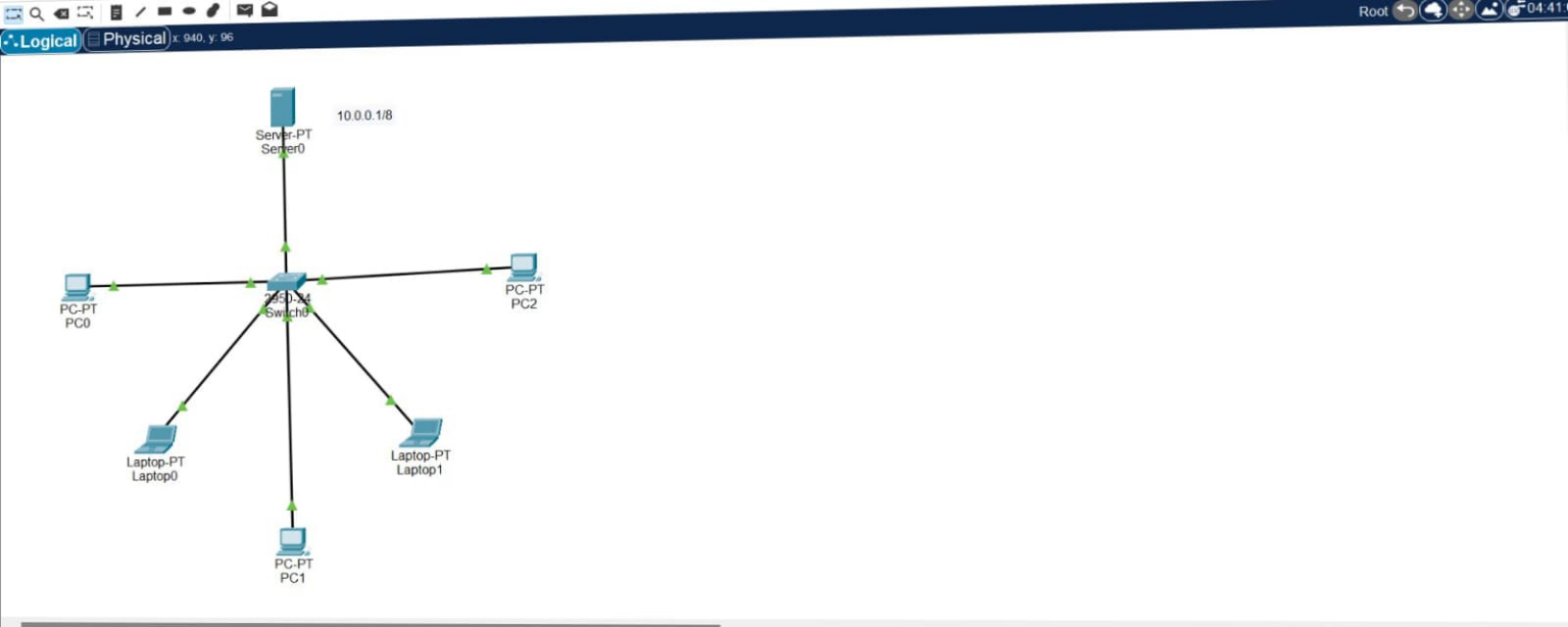
Time: 00:01:41 Realtime Simulation

Copper Straight-Through

Scenario 0 New Delete Toggle PDU List Window

Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic	Num	Edit	Delete

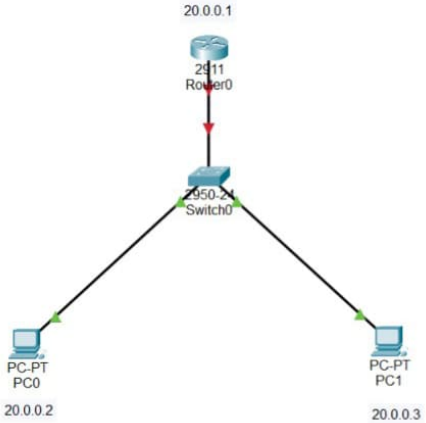

SUBNETTING C CLASS ADDRESS
NETWORK ADDRESS 192.168.10.0
/25
gig0/0
2911
Router0
gig0/1
SUBNET#1
fa0/3
2950-24
Switch0
fa0/1
fa0/2
fa0/4
fa0
PC-PT
PC0
fa0
PC-PT
PC1
fa0
PC-PT
PC2
SUBNET#2
fa0/1
2950-24
Switch1
fa0/2
fa0/3
fa0/4
fa0
PC-PT
PC3
fa0
PC-PT
PC4
fa0
PC-PT
PC5


Root
04:41
Logical
Physical
x: 940, y: 96
10.0.0.1/8
Server-PT
Server0
PC-PT
PC0
2950-24
Switch0
PC-PT
PC2
Laptop-PT
Laptop0
Laptop-PT
Laptop1
PC-PT
PC1

Cisco Packet Tracer

File Edit Options View Tools Extensions Window Help

Logical Physical x: 1091, y: 108 Root 04:29:00

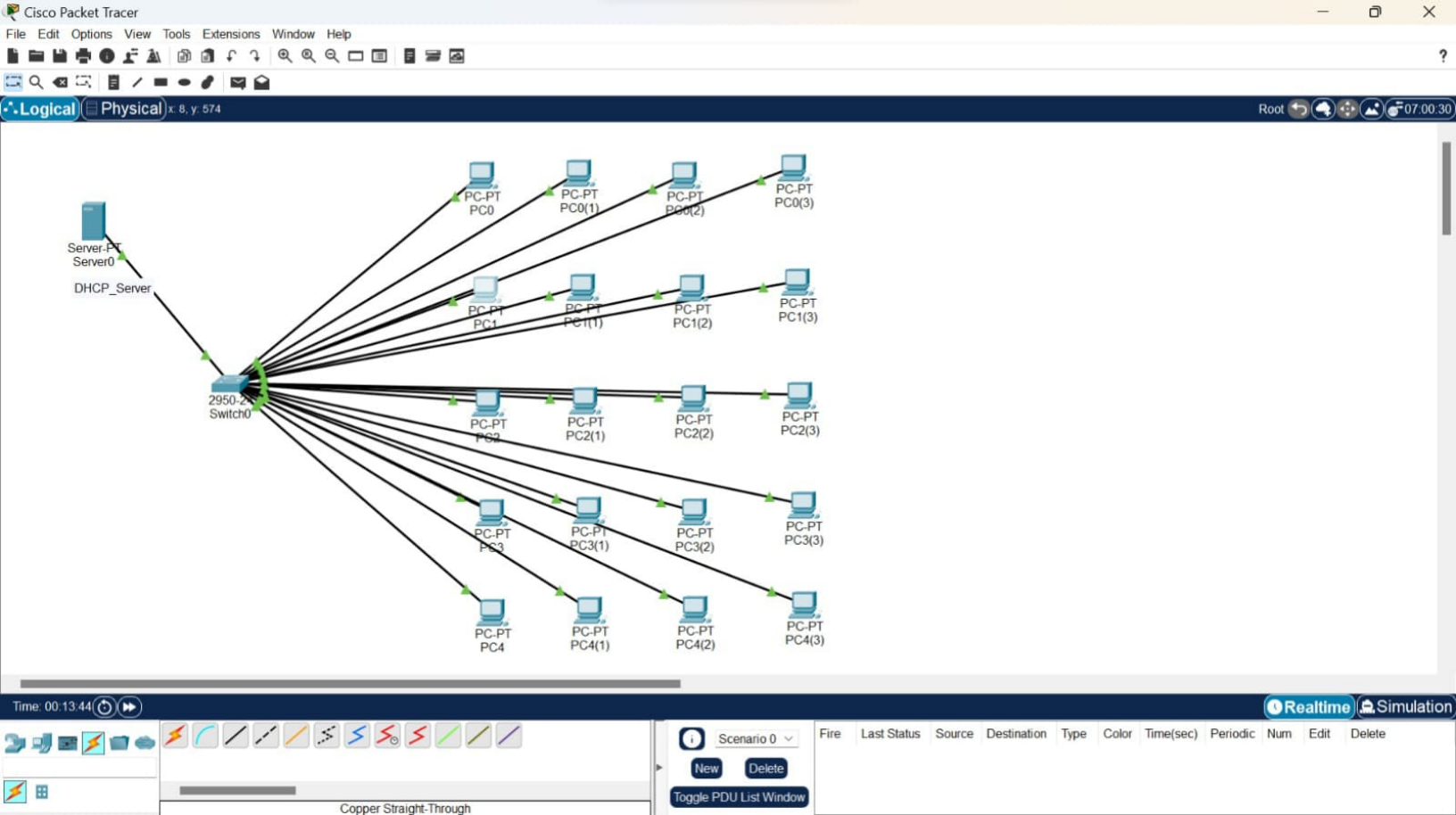


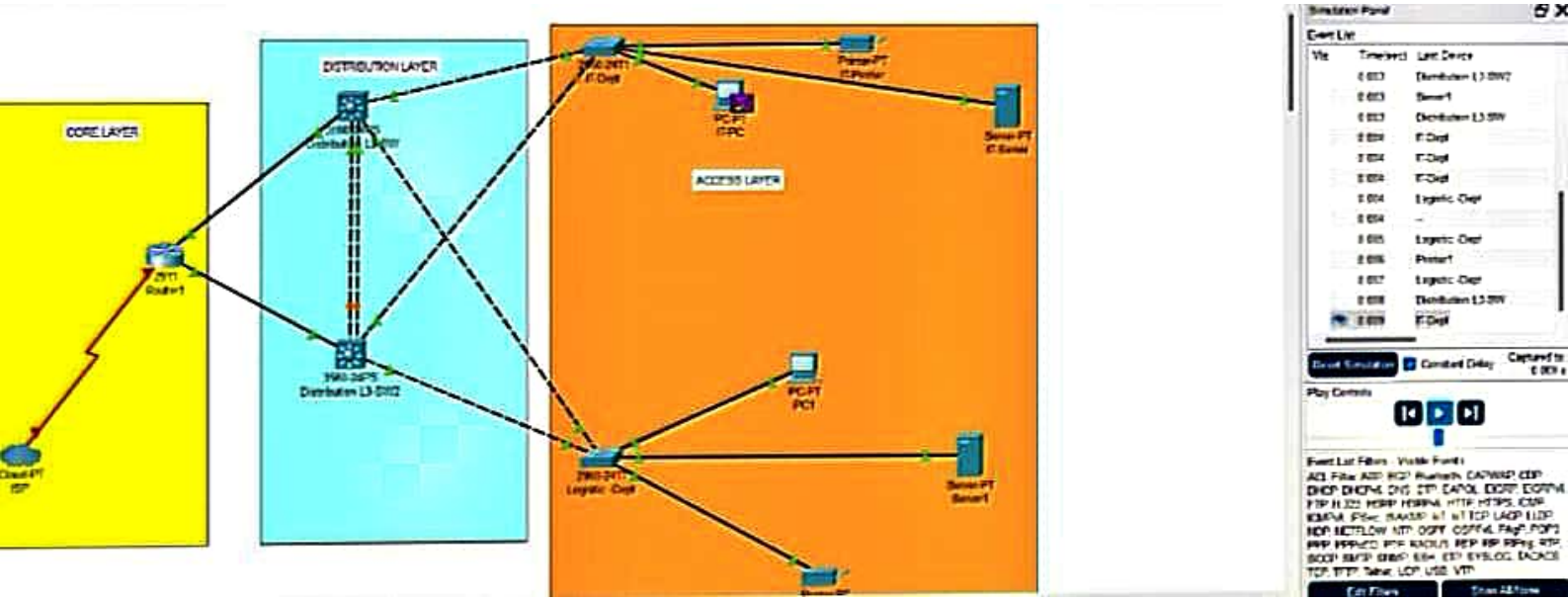
Time: 00:08:57 Realtime Simulation

Copper Straight-Through

Scenario 0 New Delete Toggle PDU List Window

Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic	Num	Edit	Delete

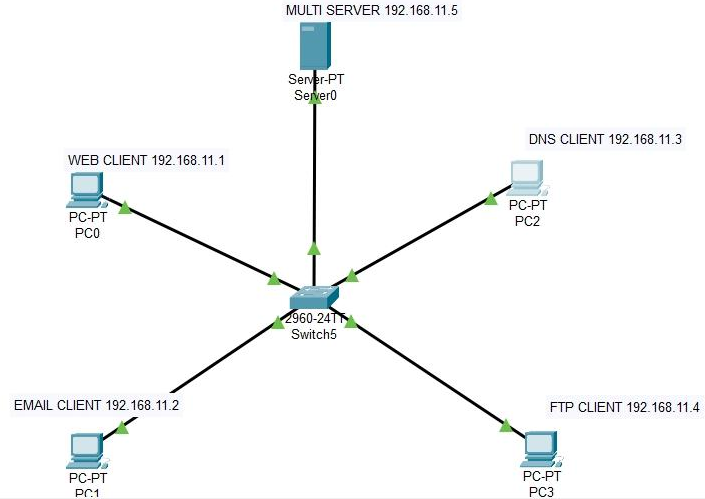

Cisco Packet Tracer
File Edit Options View Tools Extensions Window Help
Logical Physical x: 8, y: 574
Root 07:00:30
Server-PT Server0
DHCP_Server
2950-24 Switch0
PC-PT PC0
PC-PT PC0(1)
PC-PT PC0(2)
PC-PT PC0(3)
PC-PT PC1
PC-PT PC1(1)
PC-PT PC1(2)
PC-PT PC1(3)
PC-PT PC2
PC-PT PC2(1)
PC-PT PC2(2)
PC-PT PC2(3)
PC-PT PC3
PC-PT PC3(1)
PC-PT PC3(2)
PC-PT PC3(3)
PC-PT PC4
PC-PT PC4(1)
PC-PT PC4(2)
PC-PT PC4(3)
Time: 00:13:44
Realtime Simulation
Copper Straight-Through
Scenario 0
New
Delete
Toggle PDU List Window
Fire Last Status Source Destination Type Color Time(sec) Periodic Num Edit Delete


CORE LAYER
DISTRIBUTION LAYER
ACCESS LAYER
PC-PT
PC1
Server-PT
Server1

Cisco Packet Tracer

File Edit Options View Tools Extensions Window Help

Logical Physical x: 936, y: 3799 Root 19:28:30



Time: 00:38:35 Realtime Simulation

Copper Straight-Through

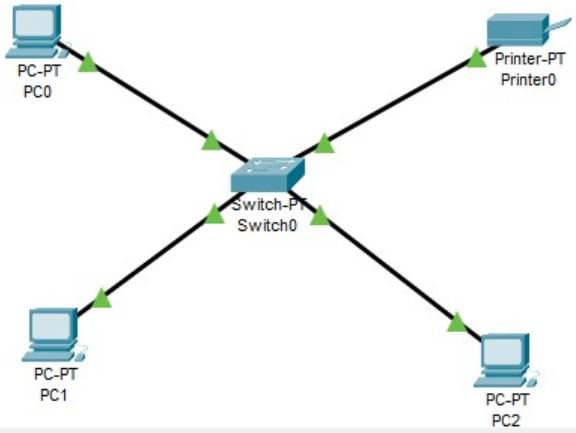
Scenario 0 New Delete Toggle PDU List Window

Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic	Num	Edit	Delete

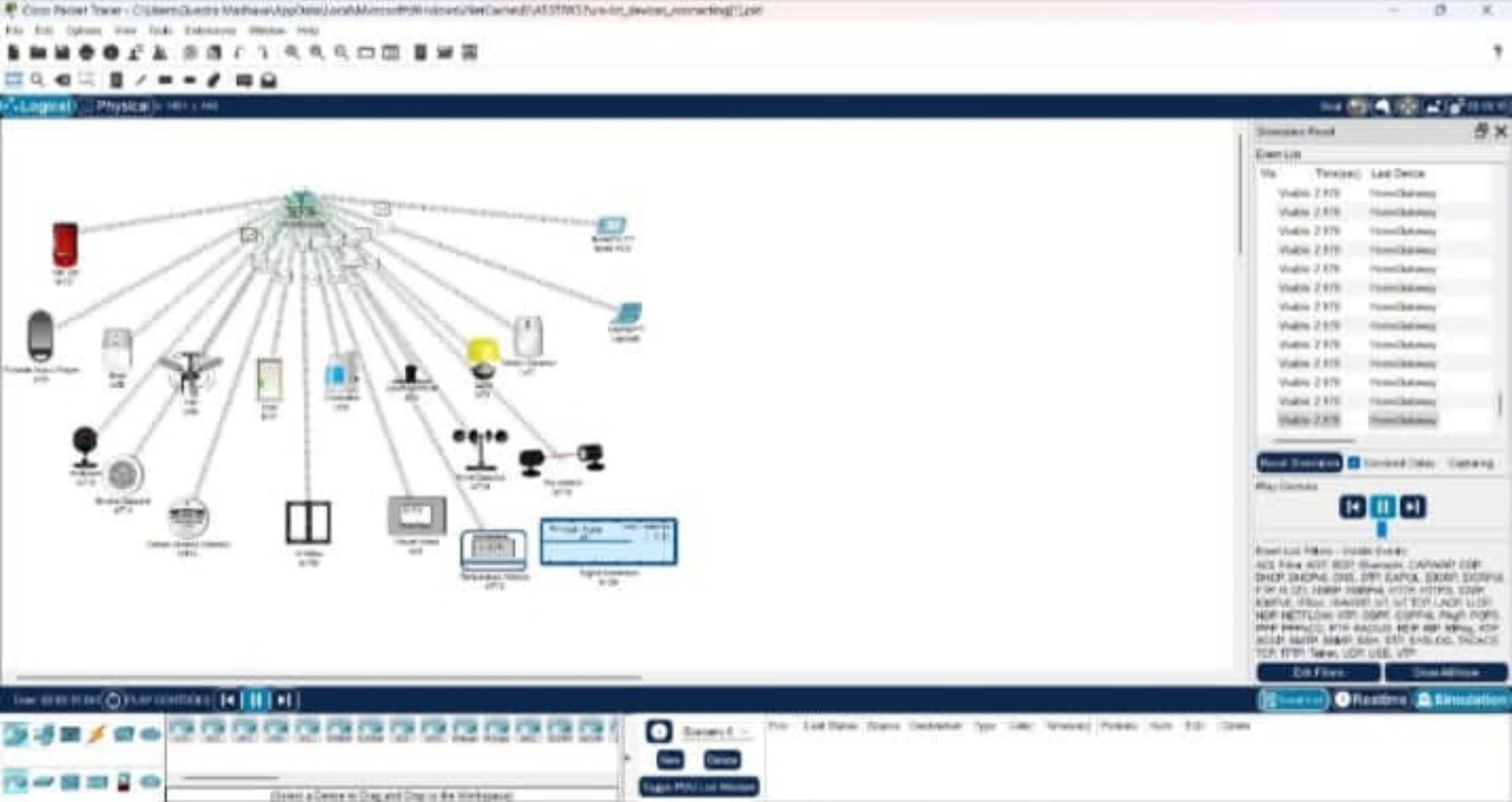
86°F Haze Search ENG IN 14:09 12-02-2025

File Edit Options View Tools Extensions Window Help

Logical Physical x: 522, y: 124 Root 18:05:30

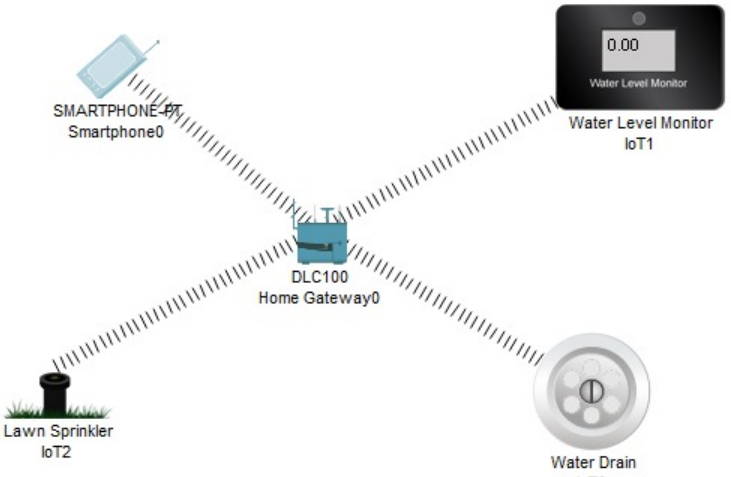


Time: 00:36:15 Realtime Simulation



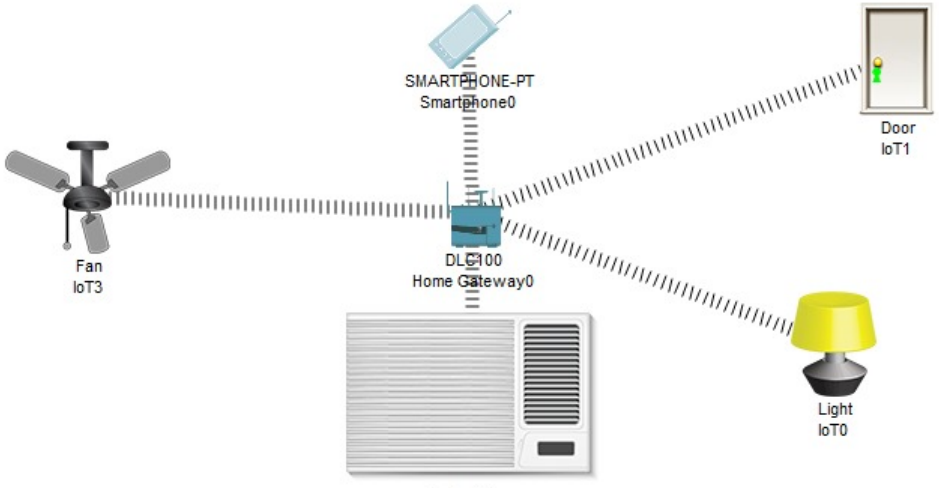
File Edit Options View Tools Extensions Window Help

Logical Physical x: 616, y: 197 Root 01:57:00



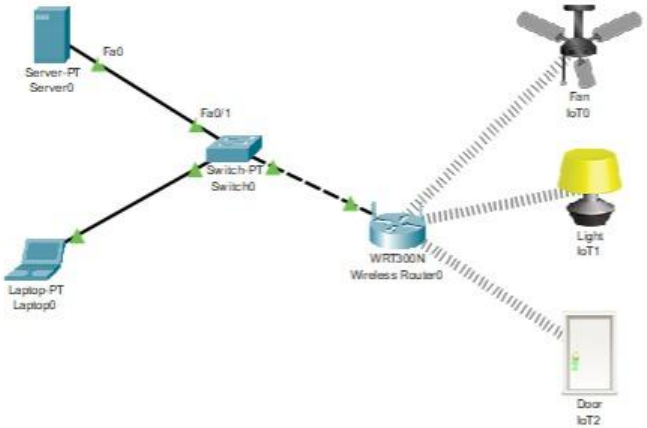
File Edit Options View Tools Extensions Window Help

Logical Physical x: 782, y: 55 Root 01:09:30

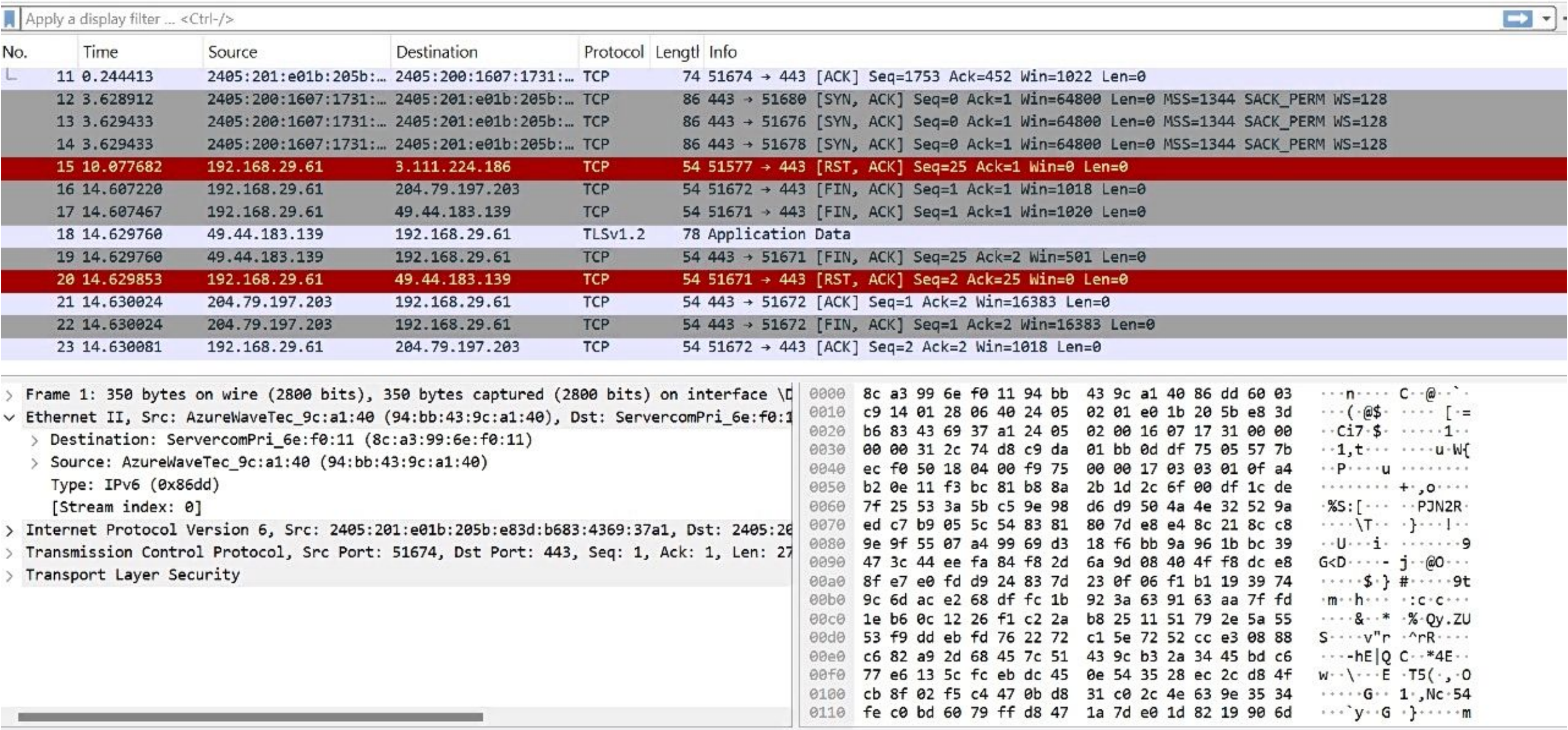


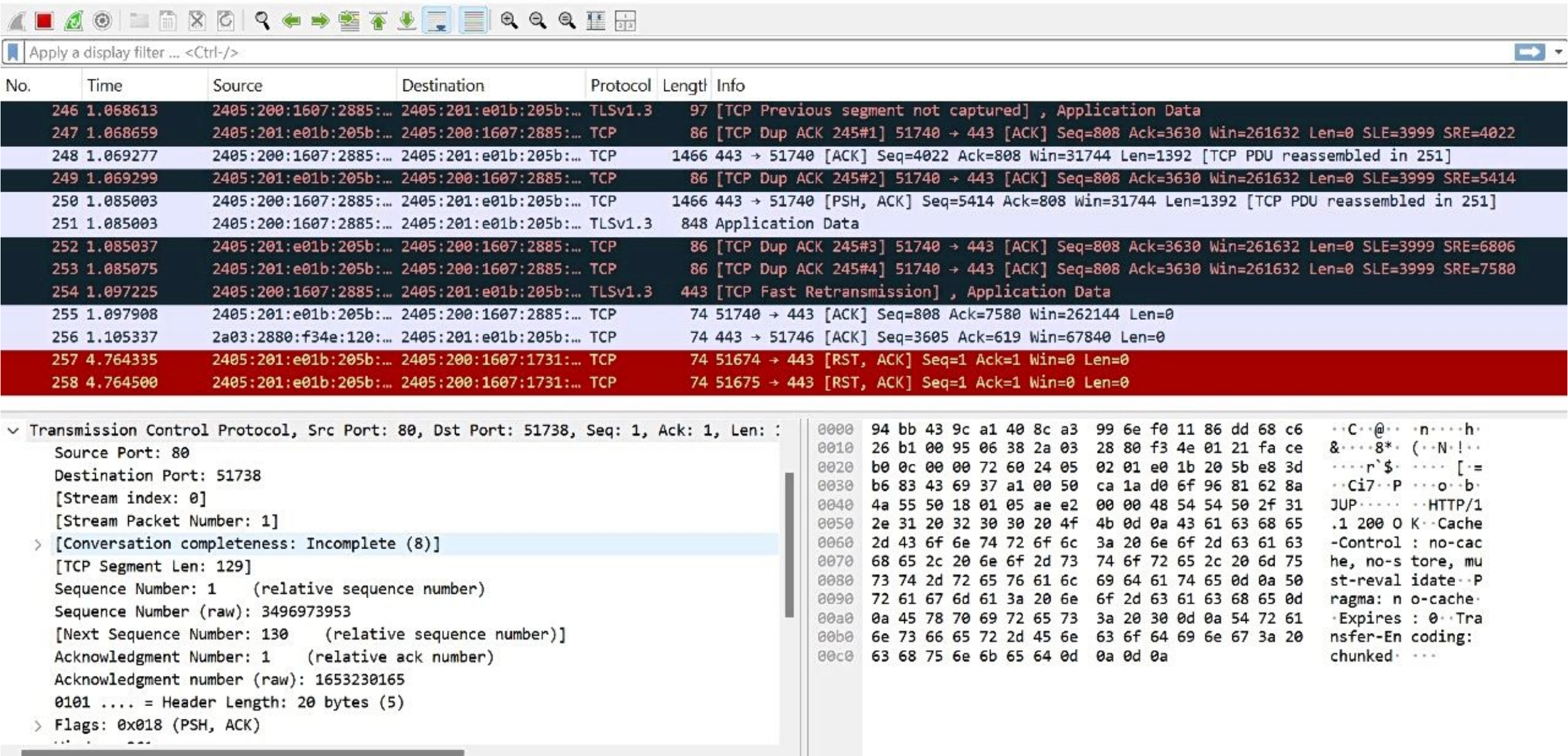
File Edit Options View Tools Extensions Window Help

Logical Physical x: 519, y: 188 Root 07:06:30



Time: 00:13:51 Realtime Simulation


Apply a display filter ... <Ctrl-/>
No. Time Source Destination Protocol Lengtl Info
11 0.244413 2405:201:e01b:205b:... 2405:200:1607:1731:... TCP 74 51674 → 443 [ACK] Seq=1753 Ack=452 Win=1022 Len=0
12 3.628912 2405:200:1607:1731:... 2405:201:e01b:205b:... TCP 86 443 → 51680 [SYN, ACK] Seq=0 Ack=1 Win=64800 Len=0 MSS=1344 SACK_PERM WS=128
13 3.629433 2405:200:1607:1731:... 2405:201:e01b:205b:... TCP 86 443 → 51676 [SYN, ACK] Seq=0 Ack=1 Win=64800 Len=0 MSS=1344 SACK_PERM WS=128
14 3.629433 2405:200:1607:1731:... 2405:201:e01b:205b:... TCP 86 443 → 51678 [SYN, ACK] Seq=0 Ack=1 Win=64800 Len=0 MSS=1344 SACK_PERM WS=128
15 10.077682 192.168.29.61 3.111.224.186 TCP 54 51577 → 443 [RST, ACK] Seq=25 Ack=1 Win=0 Len=0
16 14.607220 192.168.29.61 204.79.197.203 TCP 54 51672 → 443 [FIN, ACK] Seq=1 Ack=1 Win=1018 Len=0
17 14.607467 192.168.29.61 49.44.183.139 TCP 54 51671 → 443 [FIN, ACK] Seq=1 Ack=1 Win=1020 Len=0
18 14.629760 49.44.183.139 192.168.29.61 TLSv1.2 78 Application Data
19 14.629760 49.44.183.139 192.168.29.61 TCP 54 443 → 51671 [FIN, ACK] Seq=25 Ack=2 Win=501 Len=0
20 14.629853 192.168.29.61 49.44.183.139 TCP 54 51671 → 443 [RST, ACK] Seq=2 Ack=25 Win=0 Len=0
21 14.630024 204.79.197.203 192.168.29.61 TCP 54 443 → 51672 [ACK] Seq=1 Ack=2 Win=16383 Len=0
22 14.630024 204.79.197.203 192.168.29.61 TCP 54 443 → 51672 [FIN, ACK] Seq=1 Ack=2 Win=16383 Len=0
23 14.630081 192.168.29.61 204.79.197.203 TCP 54 51672 → 443 [ACK] Seq=2 Ack=2 Win=1018 Len=0
Frame 1: 350 bytes on wire (2800 bits), 350 bytes captured (2800 bits) on interface \D
Ethernet II, Src: AzureWaveTec_9c:a1:40 (94:bb:43:9c:a1:40), Dst: ServercomPri_6e:f0:1
Destination: ServercomPri_6e:f0:11 (8c:a3:99:6e:f0:11)
Source: AzureWaveTec_9c:a1:40 (94:bb:43:9c:a1:40)
Type: IPv6 (0x86dd)
[Stream index: 0]
Internet Protocol Version 6, Src: 2405:201:e01b:205b:e83d:b683:4369:37a1, Dst: 2405:20
Transmission Control Protocol, Src Port: 51674, Dst Port: 443, Seq: 1, Ack: 1, Len: 27
Transport Layer Security
0000 8c a3 99 6e f0 11 94 bb 43 9c a1 40 86 dd 60 03
0010 c9 14 01 28 06 40 24 05 02 01 e0 1b 20 5b e8 3d
0020 b6 83 43 69 37 a1 24 05 02 00 16 07 17 31 00 00
0030 00 00 31 2c 74 d8 c9 da 01 bb 0d df 75 05 57 7b
0040 ec f0 50 18 04 00 f9 75 00 00 17 03 03 01 0f a4
0050 b2 0e 11 f3 bc 81 b8 8a 2b 1d 2c 6f 00 df 1c de
0060 7f 25 53 3a 5b c5 9e 98 d6 d9 50 4a 4e 32 52 9a
0070 ed c7 b9 05 5c 54 83 81 80 7d e8 e4 8c 21 8c c8
0080 9e 9f 55 07 a4 99 69 d3 18 f6 bb 9a 96 1b bc 39
0090 47 3c 44 ee fa 84 f8 2d 6a 9d 08 40 4f f8 dc e8
00a0 8f e7 e0 fd d9 24 83 7d 23 0f 06 f1 b1 19 39 74
00b0 9c 6d ac e2 68 df fc 1b 92 3a 63 91 63 aa 7f fd
00c0 1e b6 0c 12 26 f1 c2 2a b8 25 11 51 79 2e 5a 55
00d0 53 f9 dd eb fd 76 22 72 c1 5e 72 52 cc e3 08 88
00e0 c6 82 a9 2d 68 45 7c 51 43 9c b3 2a 34 45 bd c6
00f0 77 e6 13 5c fc eb dc 45 0e 54 35 28 ec 2c d8 4f
0100 cb 8f 02 f5 c4 47 0b d8 31 c0 2c 4e 63 9e 35 34
0110 fe c0 bd 60 79 ff d8 47 1a 7d e0 1d 82 19 90 6d


Apply a display filter ... <Ctrl-/>
No. Time Source Destination Protocol Lengtl Info
246 1.068613 2405:200:1607:2885:… 2405:201:e01b:205b:… TLSv1.3 97 [TCP Previous segment not captured] , Application Data
247 1.068659 2405:201:e01b:205b:… 2405:200:1607:2885:… TCP 86 [TCP Dup ACK 245#1] 51740 → 443 [ACK] Seq=808 Ack=3630 Win=261632 Len=0 SLE=3999 SRE=4022
248 1.069277 2405:200:1607:2885:… 2405:201:e01b:205b:… TCP 1466 443 → 51740 [ACK] Seq=4022 Ack=808 Win=31744 Len=1392 [TCP PDU reassembled in 251]
249 1.069299 2405:201:e01b:205b:… 2405:200:1607:2885:… TCP 86 [TCP Dup ACK 245#2] 51740 → 443 [ACK] Seq=808 Ack=3630 Win=261632 Len=0 SLE=3999 SRE=5414
250 1.085003 2405:200:1607:2885:… 2405:201:e01b:205b:… TCP 1466 443 → 51740 [PSH, ACK] Seq=5414 Ack=808 Win=31744 Len=1392 [TCP PDU reassembled in 251]
251 1.085003 2405:200:1607:2885:… 2405:201:e01b:205b:… TLSv1.3 848 Application Data
252 1.085037 2405:201:e01b:205b:… 2405:200:1607:2885:… TCP 86 [TCP Dup ACK 245#3] 51740 → 443 [ACK] Seq=808 Ack=3630 Win=261632 Len=0 SLE=3999 SRE=6806
253 1.085075 2405:201:e01b:205b:… 2405:200:1607:2885:… TCP 86 [TCP Dup ACK 245#4] 51740 → 443 [ACK] Seq=808 Ack=3630 Win=261632 Len=0 SLE=3999 SRE=7580
254 1.097225 2405:200:1607:2885:… 2405:201:e01b:205b:… TLSv1.3 443 [TCP Fast Retransmission] , Application Data
255 1.097908 2405:201:e01b:205b:… 2405:200:1607:2885:… TCP 74 51740 → 443 [ACK] Seq=808 Ack=7580 Win=262144 Len=0
256 1.105337 2a03:2880:f34e:120:… 2405:201:e01b:205b:… TCP 74 443 → 51746 [ACK] Seq=3605 Ack=619 Win=67840 Len=0
257 4.764335 2405:201:e01b:205b:… 2405:200:1607:1731:… TCP 74 51674 → 443 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
258 4.764500 2405:201:e01b:205b:… 2405:200:1607:1731:… TCP 74 51675 → 443 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
Transmission Control Protocol, Src Port: 80, Dst Port: 51738, Seq: 1, Ack: 1, Len:
Source Port: 80
Destination Port: 51738
[Stream index: 0]
[Stream Packet Number: 1]
[Conversation completeness: Incomplete (8)]
[TCP Segment Len: 129]
Sequence Number: 1 (relative sequence number)
Sequence Number (raw): 3496973953
[Next Sequence Number: 130 (relative sequence number)]
Acknowledgment Number: 1 (relative ack number)
Acknowledgment number (raw): 1653230165
0101 .... = Header Length: 20 bytes (5)
Flags: 0x018 (PSH, ACK)
0000 94 bb 43 9c a1 40 8c a3 99 6e f0 11 86 dd 68 c6 ··C··@·· ·n····h·
0010 26 b1 00 95 06 38 2a 03 28 80 f3 4e 01 21 fa ce &····8*· (··N·!··
0020 b0 0c 00 00 72 60 24 05 02 01 e0 1b 20 5b e8 3d ····r`$· ···· [·=
0030 b6 83 43 69 37 a1 00 50 ca 1a d0 6f 96 81 62 8a ··Ci7··P ···o··b·
0040 4a 55 50 18 01 05 ae e2 00 00 48 54 54 50 2f 31 JUP····· ··HTTP/1
0050 2e 31 20 32 30 30 20 4f 4b 0d 0a 43 61 63 68 65 .1 200 O K··Cache
0060 2d 43 6f 6e 74 72 6f 6c 3a 20 6e 6f 2d 63 61 63 -Control : no-cac
0070 68 65 2c 20 6e 6f 2d 73 74 6f 72 65 2c 20 6d 75 he, no-s tore, mu
0080 73 74 2d 72 65 76 61 6c 69 64 61 74 65 0d 0a 50 st-reval idate··P
0090 72 61 67 6d 61 3a 20 6e 6f 2d 63 61 63 68 65 0d ragma: n o-cache·
00a0 0a 45 78 70 69 72 65 73 3a 20 30 0d 0a 54 72 61 ·Expires : 0··Tra
00b0 6e 73 66 65 72 2d 45 6e 63 6f 64 69 6e 67 3a 20 nsfer-En coding:
00c0 63 68 75 6e 6b 65 64 0d 0a 0d 0a chunked· ···

No.	Time	Source	Destination	Protocol	Length	Info
388	48.696789	2405:200:1630:caf::…	2405:201:e01b:205b:…	TLSv1.2	1514	Ignored Unknown Record
389	48.696789	2405:200:1630:caf::…	2405:201:e01b:205b:…	TLSv1.2	1514	Ignored Unknown Record
390	48.696789	2405:200:1630:caf::…	2405:201:e01b:205b:…	TLSv1.2	1514	Ignored Unknown Record
391	48.696789	2405:200:1630:caf::…	2405:201:e01b:205b:…	TLSv1.2	1514	Ignored Unknown Record
392	48.696789	2405:200:1630:caf::…	2405:201:e01b:205b:…	TLSv1.2	2954	Ignored Unknown Record
393	48.696941	2405:201:e01b:205b:…	2405:200:1630:caf::…	TCP	74	51754 → 443 [ACK] Seq=1038 Ack=62165 Win=66048 Len=0
394	48.710219	2405:200:1630:caf::…	2405:201:e01b:205b:…	TLSv1.2	1514	Ignored Unknown Record
395	48.710288	2405:201:e01b:205b:…	2405:200:1630:caf::…	TCP	74	51754 → 443 [ACK] Seq=1038 Ack=63605 Win=66048 Len=0
396	48.711452	2405:200:1630:caf::…	2405:201:e01b:205b:…	TLSv1.2	1514	Ignored Unknown Record
397	48.711452	2405:200:1630:caf::…	2405:201:e01b:205b:…	TLSv1.2	1514	Ignored Unknown Record
398	48.711452	2405:200:1630:caf::…	2405:201:e01b:205b:…	TCP	1514	443 → 51754 [PSH, ACK] Seq=66485 Ack=1038 Win=64128 Len=1440
399	48.711554	2405:201:e01b:205b:…	2405:200:1630:caf::…	TCP	74	51754 → 443 [ACK] Seq=1038 Ack=67925 Win=66048 Len=0
400	48.736571	2405:200:1630:caf::…	2405:201:e01b:205b:…	TCP	1514	443 → 51754 [ACK] Seq=67925 Ack=1038 Win=64128 Len=1440

```
Acknowledgment Number: 1    (relative ack number)
Acknowledgment number (raw): 1653230165
0101 .... = Header Length: 20 bytes (5)
> Flags: 0x018 (PSH, ACK)
Window: 261
[Calculated window size: 261]
[Window size scaling factor: -1 (unknown)]
Checksum: 0xaee2 [unverified]
[Checksum Status: Unverified]
Urgent Pointer: 0
> [Timestamps]
> [SEQ/ACK analysis]
TCP payload (129 bytes)
TCP segment data (129 bytes)
```

```
0000  94 bb 43 9c a1 40 8c a3  99 6e f0 11 86 dd 68 c6   ··C··@·· ·n····h·
0010  26 b1 00 95 06 38 2a 03  28 80 f3 4e 01 21 fa ce   &····8*· (··N·!··
0020  b0 0c 00 00 72 60 24 05  02 01 e0 1b 20 5b e8 3d   ····r`$· ···· [·=
0030  b6 83 43 69 37 a1 00 50  ca 1a d0 6f 96 81 62 8a   ··Ci7··P ···o··b·
0040  4a 55 50 18 01 05 ae e2  00 00 48 54 54 50 2f 31   JUP····· ··HTTP/1
0050  2e 31 20 32 30 30 20 4f  4b 0d 0a 43 61 63 68 65   .1 200 O K··Cache
0060  2d 43 6f 6e 74 72 6f 6c  3a 20 6e 6f 2d 63 61 63   -Control : no-cac
0070  68 65 2c 20 6e 6f 2d 73  74 6f 72 65 2c 20 6d 75   he, no-s tore, mu
0080  73 74 2d 72 65 76 61 6c  69 64 61 74 65 0d 0a 50   st-reval idate··P
0090  72 61 67 6d 61 3a 20 6e  6f 2d 63 61 63 68 65 0d   ragma: n o-cache·
00a0  0a 45 78 70 69 72 65 73  3a 20 30 0d 0a 54 72 61   ·Expires : 0··Tra
00b0  6e 73 66 65 72 2d 45 6e  63 6f 64 69 6e 67 3a 20   nsfer-En coding:
00c0  63 68 75 6e 6b 65 64 0d  0a 0d 0a                  chunked· ···
```

main.c

Share Run

Output Clear

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>

#define PORT 8080
#define BUFFER_SIZE 1024

void send_file(FILE *fp, int sockfd) {
    char buffer[BUFFER_SIZE];

    while (fgets(buffer, BUFFER_SIZE, fp) != NULL) {
        if (send(sockfd, buffer, strlen(buffer), 0) == -1) {  // Send only valid data
            perror("Error sending file");
            exit(1);
        }
        memset(buffer, 0, BUFFER_SIZE);
    }
}

```

```
Listening on port 8080...


=== Session Ended. Please Run the code again ===
```

main.c

Share Run

Output Clear

```
21
22 int main() {
23     int sockfd, new_sock;
24     struct sockaddr_in server_addr, new_addr;
25     socklen_t addr_size;
26
27     sockfd = socket(AF_INET, SOCK_STREAM, 0);
28     if (sockfd < 0) {
29         perror("Socket creation failed");
30         exit(EXIT_FAILURE);
31     }
32
33     server_addr.sin_family = AF_INET;
34     server_addr.sin_addr.s_addr = INADDR_ANY;
35     server_addr.sin_port = htons(PORT);
36
37     if (bind(sockfd, (struct sockaddr *)&server_addr, sizeof
           (server_addr)) < 0) {
38         perror("Binding failed");
39         close(sockfd);
40         exit(EXIT_FAILURE);
41     }
```

```
Listening on port 8080...


=== Session Ended. Please Run the code again ===
```

main.c

Share Run

Output Clear

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>

#define PORT 8080
#define BUFFER_SIZE 1024

void receive_file(int sockfd) {
    char buffer[BUFFER_SIZE];
    FILE *fp = fopen("received_file.txt", "w");
    if (fp == NULL) {
        perror("File open failed");
        exit(1);
    }

    int n;
    while ((n = recv(sockfd, buffer, BUFFER_SIZE, 0)) > 0) {
        fwrite(buffer, sizeof(char), n, fp);
        memset(buffer, 0, BUFFER_SIZE);
    }
```

```
Connection failed: Connection refused


=== Code Exited With Errors ===
```

main.c

Share Run

Output Clear

```
    }

    if (n < 0) {
        perror("Error receiving file");
    }

    fclose(fp);
}

int main() {
    int sockfd;
    struct sockaddr_in server_addr;

    sockfd = socket(AF_INET, SOCK_STREAM, 0);
    if (sockfd < 0) {
        perror("Socket creation failed");
        exit(EXIT_FAILURE);
    }

    server_addr.sin_family = AF_INET;
    server_addr.sin_port = htons(PORT);
    server_addr.sin_addr.s_addr = inet_addr("127.0.0.1"); //
```

```
Connection failed: Connection refused


=== Code Exited With Errors ===
```

main.c

Share Run

Output Clear

```
        exit(EXIT_FAILURE);
    }

    server_addr.sin_family = AF_INET;
    server_addr.sin_port = htons(PORT);
    server_addr.sin_addr.s_addr = inet_addr("127.0.0.1"); //
        Change to server IP if necessary

    if (connect(sockfd, (struct sockaddr *)&server_addr, sizeof
        (server_addr)) < 0) {
        perror("Connection failed");
        close(sockfd);
        exit(EXIT_FAILURE);
    }

    receive_file(sockfd);
    printf("File received successfully.\n");

    close(sockfd);
    return 0;
}
```

```
Connection failed: Connection refused


=== Code Exited With Errors ===
```

main.c

Share Run

Output Clear

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>

#define PORT 5353
#define BUFFER_SIZE 512
#define DOMAIN "example.com"
#define IP_ADDRESS "93.184.216.34"

int main() {
    int sockfd;
    struct sockaddr_in server_addr, client_addr;
    socklen_t addr_len = sizeof(client_addr);
    char buffer[BUFFER_SIZE];

    if ((sockfd = socket(AF_INET, SOCK_DGRAM, 0)) < 0) {
        perror("Socket creation failed");
        exit(EXIT_FAILURE);
    }

```

```
DNS Server is listening on port 5353
```

main.c

Share Run

Output Clear

```
22
23      memset(&server_addr, 0, sizeof(server_addr));
24      server_addr.sin_family = AF_INET;
25      server_addr.sin_addr.s_addr = INADDR_ANY;
26      server_addr.sin_port = htons(PORT);
27
28      if (bind(sockfd, (const struct sockaddr *)&server_addr,
            sizeof(server_addr)) < 0) {
29          perror("Bind failed");
30          close(sockfd);
31          exit(EXIT_FAILURE);
32      }
33
34      printf("DNS Server is listening on port %d\n", PORT);
35
36      while (1) {
37          int n = recvfrom(sockfd, buffer, BUFFER_SIZE - 1, 0,
38                           (struct sockaddr *)&client_addr,
                                 &addr_len);
39          if (n < 0) {
40              perror("Receive failed");
41              continue;
```

```
DNS Server is listening on port 5353
```

main.c

Share Run

Output Clear

```
        continue;
    }

    buffer[n] = '\0'; // Null-terminate safely
    printf("Received request for: %s\n", buffer);

    if (strncmp(buffer, DOMAIN, BUFFER_SIZE) == 0) {
        if (sendto(sockfd, IP_ADDRESS, strlen(IP_ADDRESS),
            0,
                   (struct sockaddr *)&client_addr,
                       addr_len) < 0) {
            perror("Send failed");
        } else {
            printf("Responded with IP: %s\n", IP_ADDRESS);
        }
    } else {
        const char *not_found = "Not found";
        if (sendto(sockfd, not_found, strlen(not_found), 0,

                   (struct sockaddr *)&client_addr,
                       addr_len) < 0) {
```

```
DNS Server is listening on port 5353
```

main.c

Share Run

Output Clear

```
                        addr_len) < 0) {
            perror("Send failed");
        } else {
            printf("Responded with IP: %s\n", IP_ADDRESS);
        }
    } else {
        const char *not_found = "Not found";
        if (sendto(sockfd, not_found, strlen(not_found), 0,
                (struct sockaddr *)&client_addr,
                    addr_len) < 0) {
            perror("Send failed");
        } else {
            printf("Responded with: Not found\n");
        }
    }
}

close(sockfd);
return 0;
}
```

```
DNS Server is listening on port 5353
```

main.c

Share Run

Output Clear

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>

#define SERVER_IP "127.0.0.1"
#define PORT 5353
#define DOMAIN "google.com"
#define BUFFER_SIZE 512

int main() {
    int sockfd;
    struct sockaddr_in server_addr;
    char buffer[BUFFER_SIZE];
    socklen_t addr_len = sizeof(server_addr);

    // Create UDP socket
    if ((sockfd = socket(AF_INET, SOCK_DGRAM, 0)) < 0) {
        perror("Socket creation failed");
        exit(EXIT_FAILURE);
    }
```

```
Sent query for: google.com
```

main.c

Share Run

Output Clear

```
21          exit(EXIT_FAILURE);
22      }
23
24      memset(&server_addr, 0, sizeof(server_addr));
25      server_addr.sin_family = AF_INET;
26      server_addr.sin_port = htons(PORT);
27
28      // Convert IP address
29      if (inet_pton(AF_INET, SERVER_IP, &server_addr.sin_addr) <=
            0) {
30          perror("Invalid address/ Address not supported");
31          close(sockfd);
32          exit(EXIT_FAILURE);
33      }
34
35      // Send DNS-like query
36      if (sendto(sockfd, DOMAIN, strlen(DOMAIN), 0, (struct
            sockaddr *)&server_addr, sizeof(server_addr)) < 0) {
37          perror("Send failed");
38          close(sockfd);
39          exit(EXIT_FAILURE);
40      }
```

```
Sent query for: google.com
```

main.c

Share Run

Output Clear

```
        sockaddr *)&server_addr, sizeof(server_addr)) < 0) {
        perror("Send failed");
        close(sockfd);
        exit(EXIT_FAILURE);
    }

    printf("Sent query for: %s\n", DOMAIN);

    // Receive response
    int n = recvfrom(sockfd, buffer, BUFFER_SIZE - 1, 0,
        (struct sockaddr *)&server_addr, &addr_len);
    if (n < 0) {
        perror("Receive failed");
    } else {
        buffer[n] = '\0'; // Null-terminate the received data
            safely
        printf("Received response: %s\n", buffer);
    }

    close(sockfd);
    return 0;
}
```

```
Sent query for: google.com
```

main.c

Share Run

Output Clear

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <arpa/inet.h>
#include <unistd.h>
#include <netinet/in.h>

#define DNS_PORT 53
#define DNS_SERVER "8.8.8.8" // Google's public DNS server
#define BUFFER_SIZE 65536

// DNS header structure
struct DNS_HEADER {
    unsigned short id;
    unsigned char rd :1;
    unsigned char tc :1;
    unsigned char aa :1;
    unsigned char opcode :4;
    unsigned char qr :1;
    unsigned char rcode :4;
    unsigned char cd :1;
    unsigned char ad :1;
```

```
Usage: /tmp/gyme5pnJbB/main.o <hostname>


=== Code Exited With Errors ===
```

main.c

Share Run

Output Clear

```
    unsigned char ad :1;
    unsigned char z :1;
    unsigned char ra :1;
    unsigned short q_count;
    unsigned short ans_count;
    unsigned short auth_count;
    unsigned short add_count;
};

// DNS question structure
struct QUESTION {
    unsigned short qtype;
    unsigned short qclass;
};

// DNS resource data structure
struct R_DATA {
    unsigned short type;
    unsigned short _class;
    unsigned int ttl;
    unsigned short data_len;
};
```

```
Usage: /tmp/gyme5pnJbB/main.o <hostname>


=== Code Exited With Errors ===
```

main.c

Share Run

```
};

// Structure of a response record
struct RES_RECORD {
    unsigned char *name;
    struct R_DATA *resource;
    unsigned char *rdata;
};

// Convert hostname into DNS format
void ChangeToDnsNameFormat(unsigned char *dns, const char
    *host) {
    char host_copy[256]; // Safe buffer for modification
    strcpy(host_copy, host);
    strcat(host_copy, ".");

    int lock = 0;
    for (int i = 0; i < strlen(host_copy); i++) {
        if (host_copy[i] == '.') {
            *dns++ = i - lock;
            for (; lock < i; lock++) {
```

Output Clear

```
Usage: /tmp/gyme5pnJbB/main.o <hostname>


=== Code Exited With Errors ===
```

main.c

Share Run

```
            *dns++ = i - lock;
            for (; lock < i; lock++) {
                *dns++ = host_copy[lock];
            }
            lock++;
        }
    }
    *dns = '\0';
}

// Perform a DNS query
void PerformDnsQuery(const char *host) {
    unsigned char buf[BUFFER_SIZE], *qname;
    struct sockaddr_in dest;
    struct DNS_HEADER *dns;
    struct QUESTION *qinfo;

    // Create UDP socket
    int sock = socket(AF_INET, SOCK_DGRAM, IPPROTO_UDP);
    if (sock < 0) {
        perror("Socket creation failed");
        return;
```

Output Clear

```
Usage: /tmp/gyme5pnJbB/main.o <hostname>


=== Code Exited With Errors ===
```

main.c

Share Run

Output Clear

```
    return;
}

// Setup DNS query packet
memset(buf, 0, BUFFER_SIZE);
dns = (struct DNS_HEADER *)buf;
dns->id = (unsigned short)htons(getpid());
dns->qr = 0; // Query
dns->opcode = 0; // Standard query
dns->aa = 0;
dns->tc = 0;
dns->rd = 1; // Recursion desired
dns->ra = 0;
dns->z = 0;
dns->ad = 0;
dns->cd = 0;
dns->rcode = 0;
dns->q_count = htons(1); // One question

// Setup query name
qname = &buf[sizeof(struct DNS_HEADER)];
ChangeToDnsNameFormat(qname, host);
```

```
Usage: /tmp/gyme5pnJbB/main.o <hostname>


=== Code Exited With Errors ===
```

main.c

Share Run

Output Clear

```
111     // Setup query name
112     qname = &buf[sizeof(struct DNS_HEADER)];
113     ChangeToDnsNameFormat(qname, host);
114
115     // Setup query type and class
116     qinfo = (struct QUESTION *)&buf[sizeof(struct DNS_HEADER)
            + (strlen((const char *)qname) + 1)];
117     qinfo->qtype = htons(1);   // Type A query
118     qinfo->qclass = htons(1);  // Class IN
119
120     // Send the query
121     if (sendto(sock, buf, sizeof(struct DNS_HEADER) + (strlen
            ((const char *)qname) + 1) + sizeof(struct QUESTION),
            0,
122                (struct sockaddr *)&dest, sizeof(dest)) < 0) {
123         perror("Query sending failed");
124         close(sock);
125         return;
126     }
127
128     printf("Sent query for: %s\n", host);
129
```

```
Usage: /tmp/gyme5pnJbB/main.o <hostname>


=== Code Exited With Errors ===
```

main.c

Share Run

```
    printf("Sent query for: %s\n", host);

    // Receive the response
    socklen_t addr_len = sizeof(dest);
    int n = recvfrom(sock, buf, BUFFER_SIZE, 0, (struct
        sockaddr *)&dest, &addr_len);
    if (n < 0) {
        perror("Response reception failed");
        close(sock);
        return;
    }

    dns = (struct DNS_HEADER *)buf;
    unsigned char *reader = &buf[sizeof(struct DNS_HEADER) +
        (strlen((const char *)qname) + 1) + sizeof(struct
        QUESTION)];

    printf("Resolved IP Addresses:\n");

    // Read each answer record
    for (int i = 0; i < ntohs(dns->ans_count); i++) {
        struct RES_RECORD answer;
```

Output

Clear

```
Usage: /tmp/gyme5pnJbB/main.o <hostname>


=== Code Exited With Errors ===
```

main.c

Share Run

Output Clear

```
for (int i = 0; i < ntohs(dns->ans_count); i++) {
    struct RES_RECORD answer;
    answer.name = reader;
    reader += 2; // Move past name

    answer.resource = (struct R_DATA *)reader;
    reader += sizeof(struct R_DATA);

    if (ntohs(answer.resource->type) == 1) { // IPv4
        address
        answer.rdata = (unsigned char *)malloc(ntohs
            (answer.resource->data_len));
        if (!answer.rdata) {
            fprintf(stderr, "Memory allocation failed\n");
            close(sock);
            return;
        }

        memcpy(answer.rdata, reader, ntohs(answer.resource
            ->data_len));
        reader += ntohs(answer.resource->data_len);
```

```
Usage: /tmp/gyme5pnJbB/main.o <hostname>


=== Code Exited With Errors ===
```

main.c

Share Run

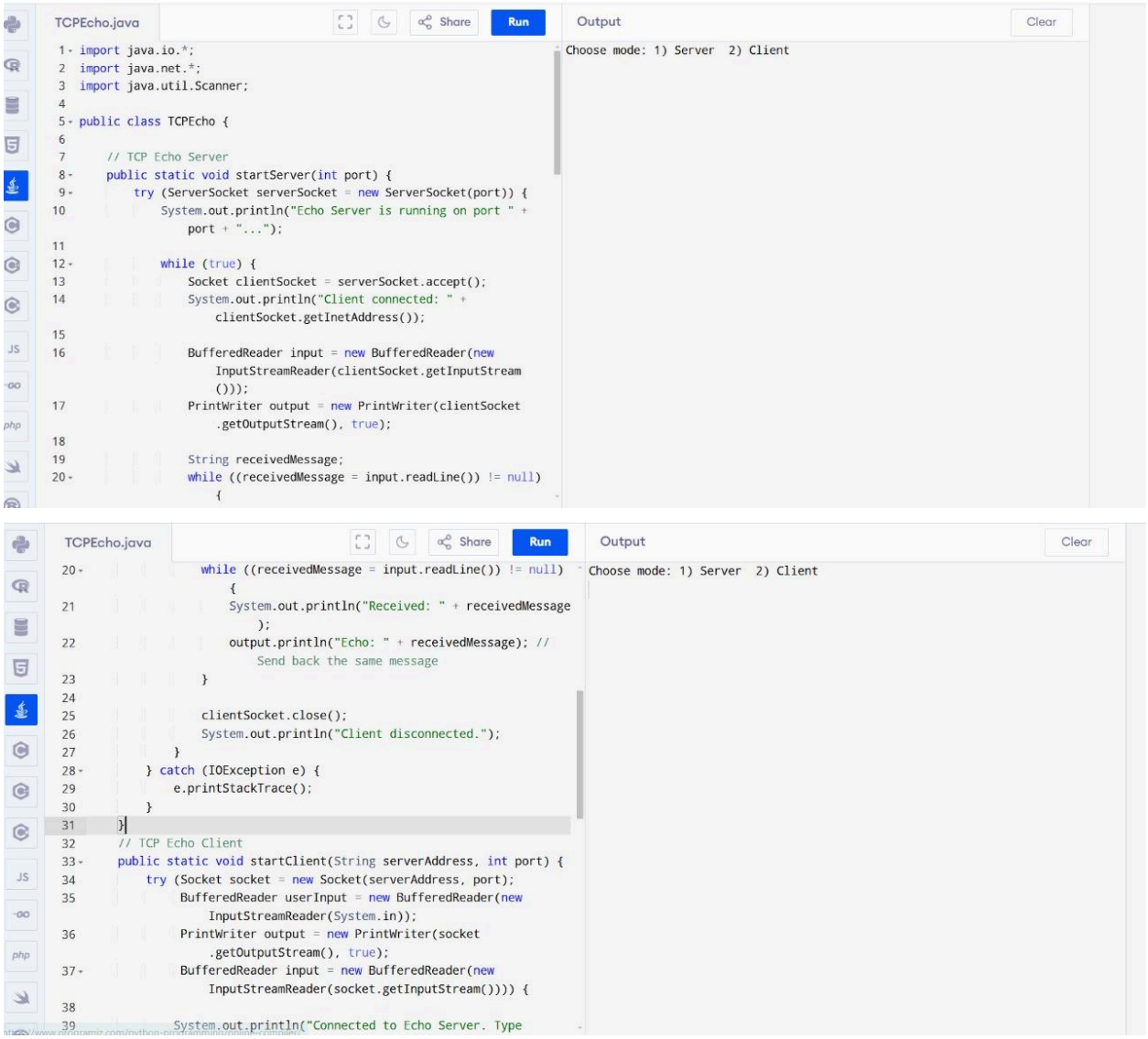
```
160
161             memcpy(answer.rdata, reader, ntohs(answer.resource
                    ->data_len));
162             reader += ntohs(answer.resource->data_len);
163
164             // Print the resolved IP address
165             printf("%d.%d.%d.%d\n", answer.rdata[0], answer
                    .rdata[1], answer.rdata[2], answer.rdata[3]);
166
167             free(answer.rdata);
168         }
169     }
170
171     close(sock);
172 }
173
174 int main(int argc, char *argv[]) {
175     if (argc < 2) {
176         printf("Usage: %s <hostname>\n", argv[0]);
177         return 1;
178     }
179
```

Output

Clear

```
Usage: /tmp/gyme5pnJbB/main.o <hostname>


=== Code Exited With Errors ===
```

29)

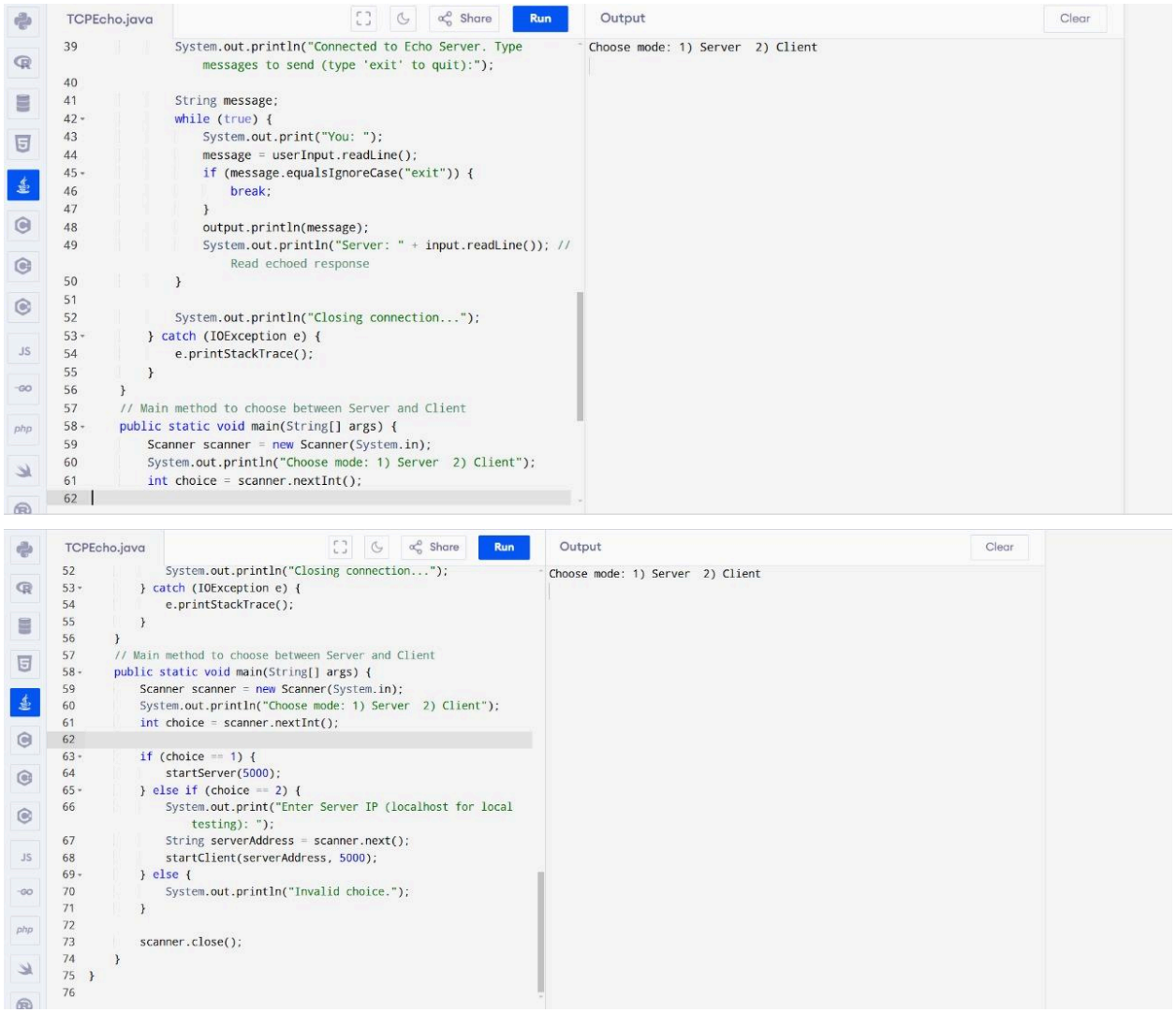


TCPEcho.java

Share Run Output Clear

```
import java.io.*;
import java.net.*;
import java.util.Scanner;

public class TCPEcho {

    // TCP Echo Server
    public static void startServer(int port) {
        try (ServerSocket serverSocket = new ServerSocket(port)) {
            System.out.println("Echo Server is running on port " +
                port + "...");

            while (true) {
                Socket clientSocket = serverSocket.accept();
                System.out.println("Client connected: " +
                    clientSocket.getInetAddress());

                BufferedReader input = new BufferedReader(new
                    InputStreamReader(clientSocket.getInputStream
                    ()));
                PrintWriter output = new PrintWriter(clientSocket
                    .getOutputStream(), true);

                String receivedMessage;
                while ((receivedMessage = input.readLine()) != null)
                    {
                    System.out.println("Received: " + receivedMessage
                        );
                    output.println("Echo: " + receivedMessage); //
                        Send back the same message
                }

                clientSocket.close();
                System.out.println("Client disconnected.");
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
    // TCP Echo Client
    public static void startClient(String serverAddress, int port) {
        try (Socket socket = new Socket(serverAddress, port);
             BufferedReader userInput = new BufferedReader(new
                 InputStreamReader(System.in));
             PrintWriter output = new PrintWriter(socket
                 .getOutputStream(), true);
             BufferedReader input = new BufferedReader(new
                 InputStreamReader(socket.getInputStream()))) {

            System.out.println("Connected to Echo Server. Type
```

```
Choose mode: 1) Server  2) Client
```


TCPEcho.java
Share
Run
Output
Clear
Choose mode: 1) Server 2) Client
System.out.println("Connected to Echo Server. Type messages to send (type 'exit' to quit):");
String message;
while (true) {
System.out.print("You: ");
message = userInput.readLine();
if (message.equalsIgnoreCase("exit")) {
break;
}
output.println(message);
System.out.println("Server: " + input.readLine()); // Read echoed response
}
System.out.println("Closing connection...");
} catch (IOException e) {
e.printStackTrace();
}
}
// Main method to choose between Server and Client
public static void main(String[] args) {
Scanner scanner = new Scanner(System.in);
System.out.println("Choose mode: 1) Server 2) Client");
int choice = scanner.nextInt();
TCPEcho.java
Share
Run
Output
Clear
Choose mode: 1) Server 2) Client
System.out.println("Closing connection...");
} catch (IOException e) {
e.printStackTrace();
}
}
// Main method to choose between Server and Client
public static void main(String[] args) {
Scanner scanner = new Scanner(System.in);
System.out.println("Choose mode: 1) Server 2) Client");
int choice = scanner.nextInt();
if (choice == 1) {
startServer(5000);
} else if (choice == 2) {
System.out.print("Enter Server IP (localhost for local testing): ");
String serverAddress = scanner.next();
startClient(serverAddress, 5000);
} else {
System.out.println("Invalid choice.");
}
scanner.close();
}
}

30) server

Programiz
C Online Compiler

Program

main.c

Share

Run

Output

```
Server is listening on port 8080...
```

```c
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <unistd.h>
#include <arpa/inet.h>
#include <sys/socket.h>

#define PORT 8080
#define BUFFER_SIZE 1024

void handle_client(int client_socket) {
    char buffer[BUFFER_SIZE];
    int bytes_read;

    while (1) {
        memset(buffer, 0, BUFFER_SIZE);
        bytes_read = recv(client_socket, buffer, BUFFER_SIZE, 0
            );

        if (bytes_read <= 0) {
            printf("Client disconnected.\n");
            break;
        }

        printf("Client: %s\n", buffer);
        send(client_socket, buffer, bytes_read, 0);
    }

    close(client_socket);
}

int main() {
    int server_socket, client_socket;
    struct sockaddr_in server_addr, client_addr;
    socklen_t client_len = sizeof(client_addr);

    // Create socket
    server_socket = socket(AF_INET, SOCK_STREAM, 0);
    if (server_socket == -1) {
        perror("Failed to create socket");
```

Server is listening on port 8080...

main.c

Share Run

Output

```
39          perror("Failed to create socket");
40          exit(EXIT_FAILURE);
41      }
42
43      // Define server address
44      server_addr.sin_family = AF_INET;
45      server_addr.sin_addr.s_addr = INADDR_ANY;
46      server_addr.sin_port = htons(PORT);
47
48      // Bind the socket to the specified IP and port
49      if (bind(server_socket, (struct sockaddr*)&server_addr,
            sizeof(server_addr)) < 0) {
50          perror("Bind failed");
51          close(server_socket);
52          exit(EXIT_FAILURE);
53      }
54
55      // Listen for incoming connections
56      if (listen(server_socket, 3) < 0) {
57          perror("Listen failed");
```

```
Server is listening on port 8080...
```

main.c

Share Run

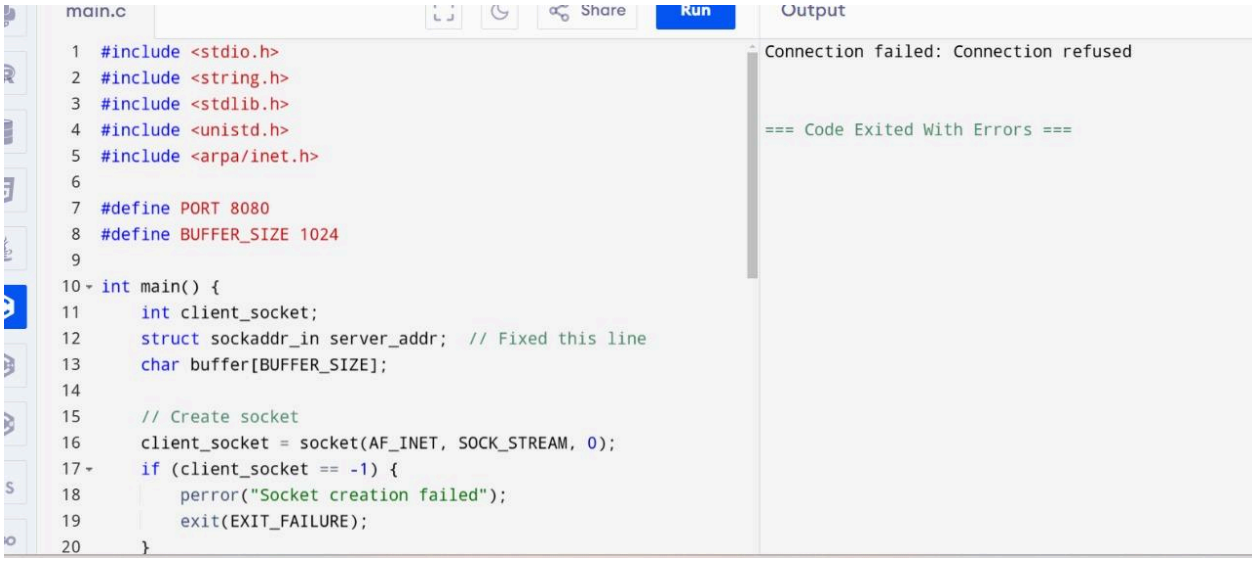
Output

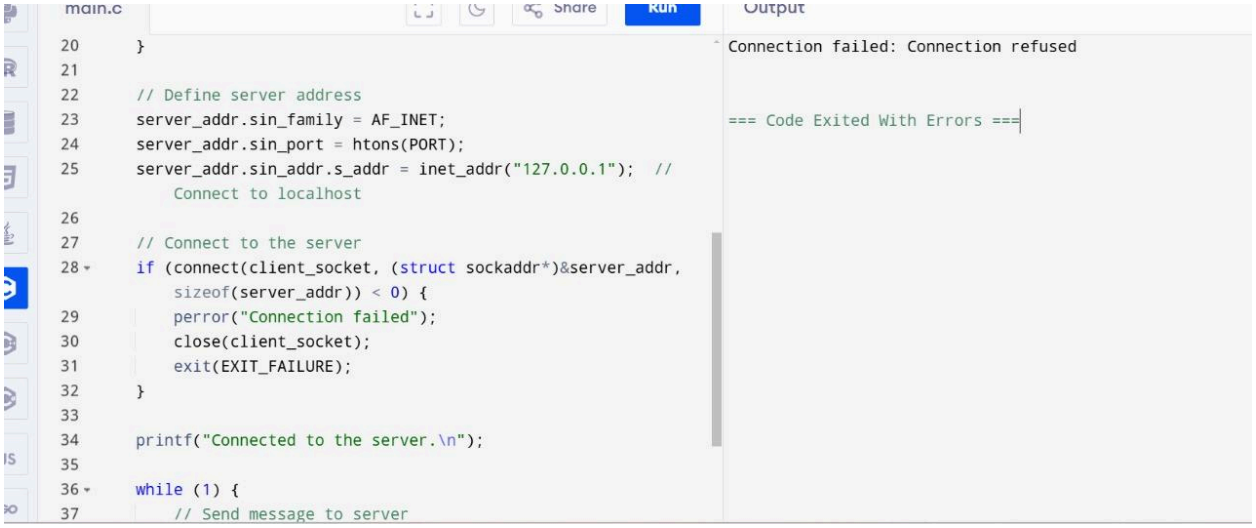
```
57          perror("Listen failed");
58          close(server_socket);
59          exit(EXIT_FAILURE);
60      }
61
62      printf("Server is listening on port %d...\n", PORT);
63
64      while (1) {
65          // Accept a new connection
66          client_socket = accept(server_socket, (struct sockaddr
                *)&client_addr, &client_len);
67          if (client_socket < 0) {
68              perror("Accept failed");
69              close(server_socket);
70              exit(EXIT_FAILURE);
71          }
72
73          printf("New client connected.\n");
74          handle_client(client_socket);
75      }
```

Server is listening on port 8080...

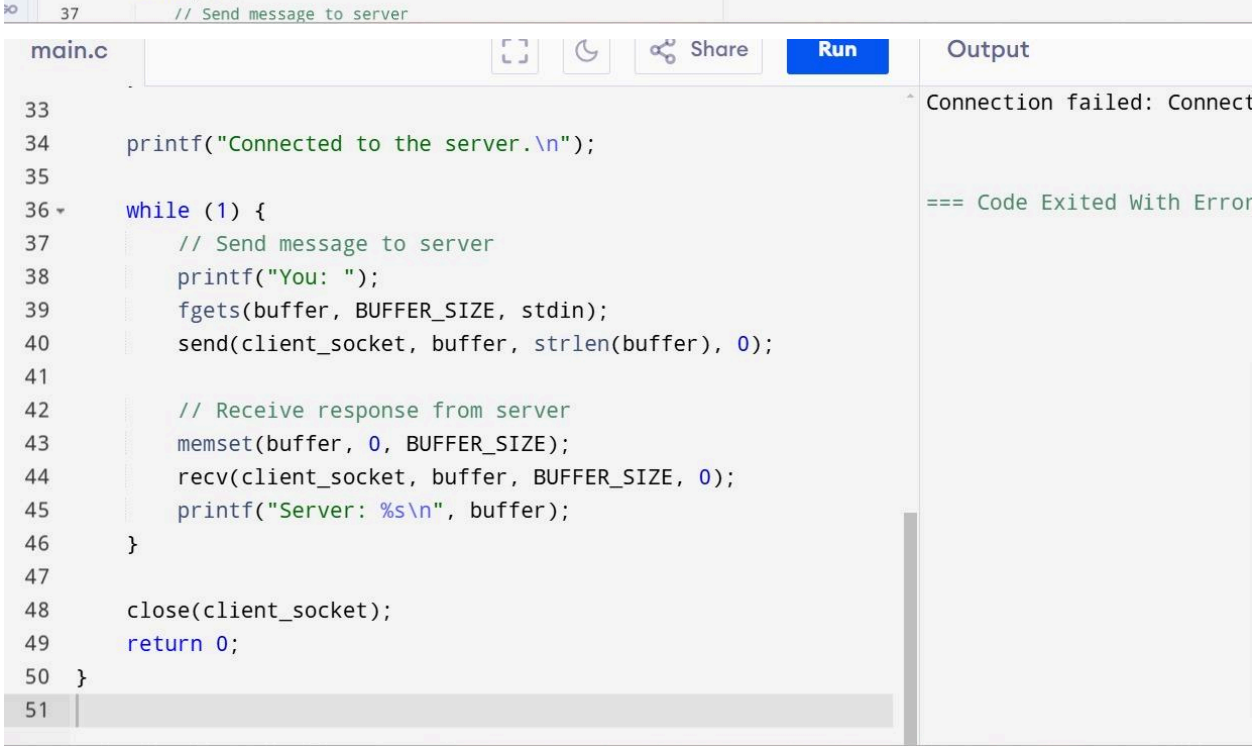
```
    while (1) {
        // Accept a new connection
        client_socket = accept(server_socket, (struct sockaddr
            *)&client_addr, &client_len);
        if (client_socket < 0) {
            perror("Accept failed");
            close(server_socket);
            exit(EXIT_FAILURE);
        }

        printf("New client connected.\n");
        handle_client(client_socket);
    }

    close(server_socket);
    return 0;
}
```

Client response





```
main.c                                                    Share   Run   Output

20      }
21
22      // Define server address
23      server_addr.sin_family = AF_INET;
24      server_addr.sin_port = htons(PORT);
25      server_addr.sin_addr.s_addr = inet_addr("127.0.0.1");  //
            Connect to localhost
26
27      // Connect to the server
28      if (connect(client_socket, (struct sockaddr*)&server_addr,
            sizeof(server_addr)) < 0) {
29          perror("Connection failed");
30          close(client_socket);
31          exit(EXIT_FAILURE);
32      }
33
34      printf("Connected to the server.\n");
35
36      while (1) {
37          // Send message to server

Connection failed: Connection refused


=== Code Exited With Errors ===
```



```
main.c                                                    Share   Run   Output

33
34      printf("Connected to the server.\n");
35
36      while (1) {
37          // Send message to server
38          printf("You: ");
39          fgets(buffer, BUFFER_SIZE, stdin);
40          send(client_socket, buffer, strlen(buffer), 0);
41
42          // Receive response from server
43          memset(buffer, 0, BUFFER_SIZE);
44          recv(client_socket, buffer, BUFFER_SIZE, 0);
45          printf("Server: %s\n", buffer);
46      }
47
48      close(client_socket);
49      return 0;
50  }
51

Connection failed: Connect


=== Code Exited With Error
```

main.c 31 Share Run

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <arpa/inet.h>
#include <netinet/if_ether.h>
#include <sys/socket.h>
#include <unistd.h>

#define CACHE_SIZE 10

typedef struct {
    char ip[INET_ADDRSTRLEN];
    unsigned char mac[ETH_ALEN];
} arp_cache_entry;

arp_cache_entry arp_cache[CACHE_SIZE];

void add_to_cache(const char *ip, unsigned char *mac) {
    for (int i = 0; i < CACHE_SIZE; i++) {
        if (strlen(arp_cache[i].ip) == 0) {
            strcpy(arp_cache[i].ip, ip);
            memcpy(arp_cache[i].mac, mac, ETH_ALEN);
            break;
        }
    }
}
```

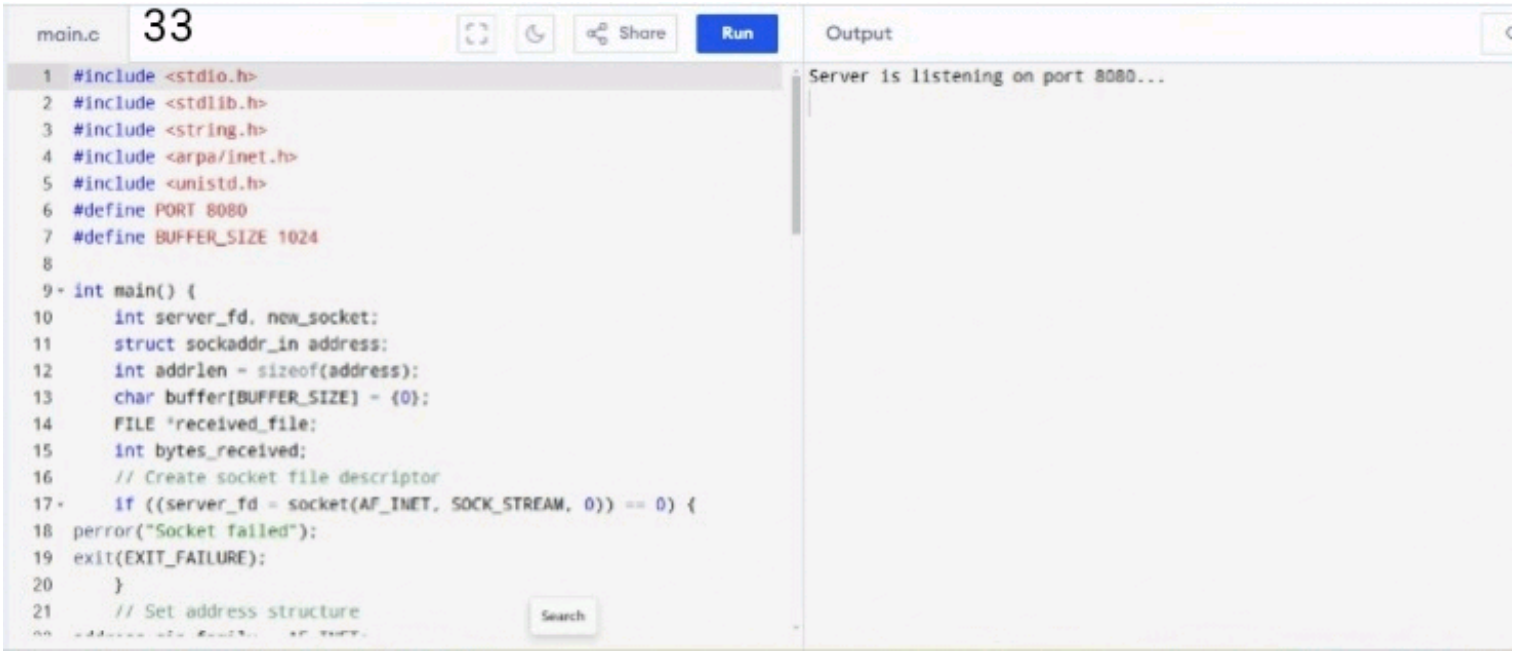
Output Clear

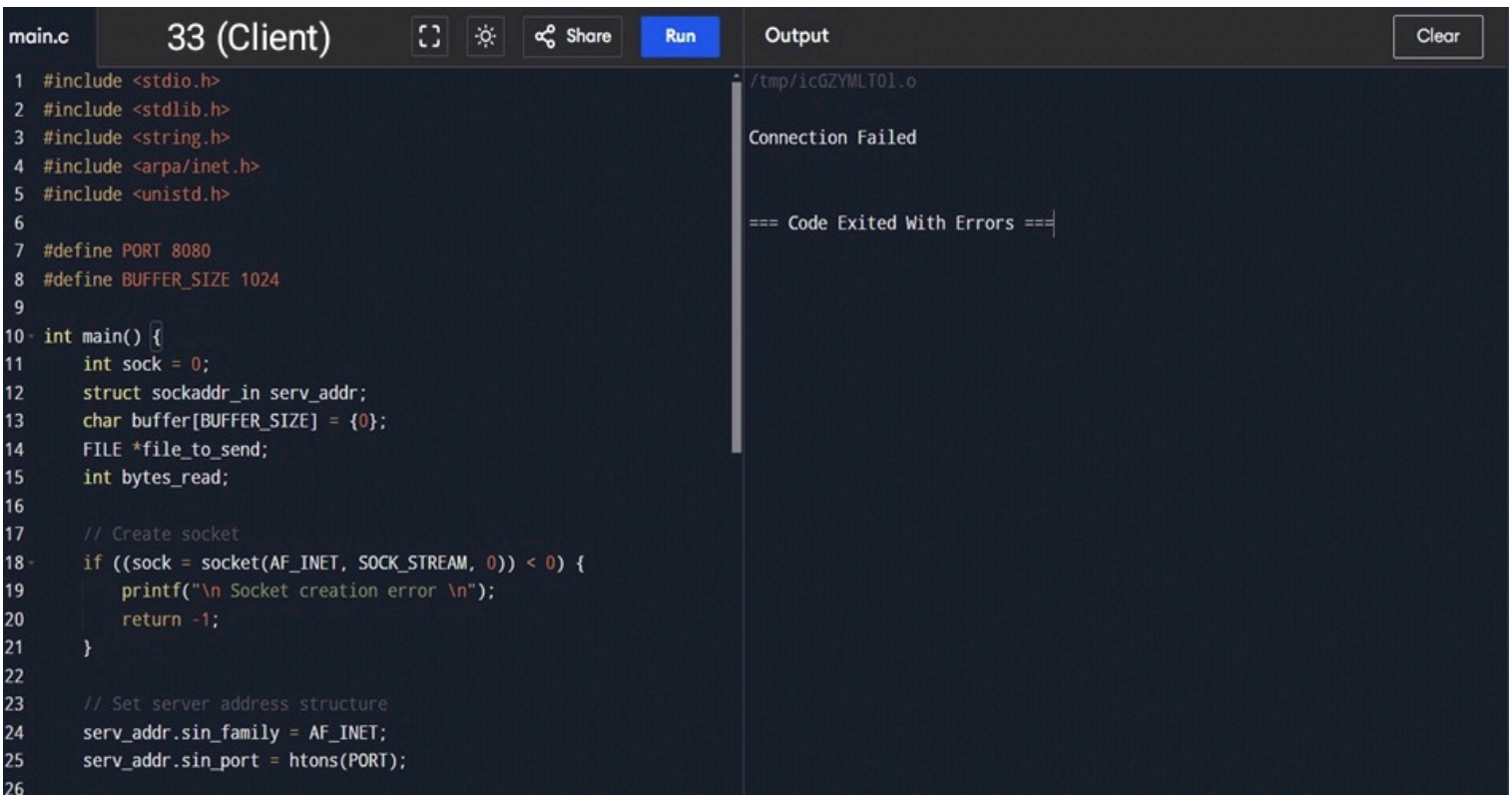
```
/tmp/8J6GoxASNj.o
IP: 192.168.1.1, MAC: 00:11:22:33:44:55


=== Code Execution Successful ===
```

main.c 32 Share Run Output Clear

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define FLAG_SEQUENCE "01111110"

// Function to perform bit stuffing
void bit_stuffing(const char *input, char *output) {
    int count = 0; // Count of consecutive '1's
    int j = 0; // Index for output array

    for (int i = 0; input[i] != '\0'; i++) {
        output[j++] = input[i]; // Copy the current bit to output

        // Count consecutive '1's
        if (input[i] == '1') {
            count++;
        } else {
            count = 0; // Reset count if '0' is found
        }

        // If we have five consecutive '1's, insert a '0'
        if (count == 5) {
            output[j++] = '0'; // Stuff a '0'
            count = 0; // Reset count after stuffing
```

```
/tmp/gudr89sSou.o
Input Data: 11111011101111101011111
Stuffed Data: 111110011101111101010111110
Destuffed Data: 11111011101111101011111


=== Code Execution Successful ===
```


main.c
33
Share
Run
Output
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <arpa/inet.h>
#include <unistd.h>
#define PORT 8080
#define BUFFER_SIZE 1024
int main() {
int server_fd, new_socket;
struct sockaddr_in address;
int addrlen = sizeof(address);
char buffer[BUFFER_SIZE] = {0};
FILE *received_file;
int bytes_received;
// Create socket file descriptor
if ((server_fd = socket(AF_INET, SOCK_STREAM, 0)) == 0) {
perror("Socket failed");
exit(EXIT_FAILURE);
}
// Set address structure
Search
Server is listening on port 8080...

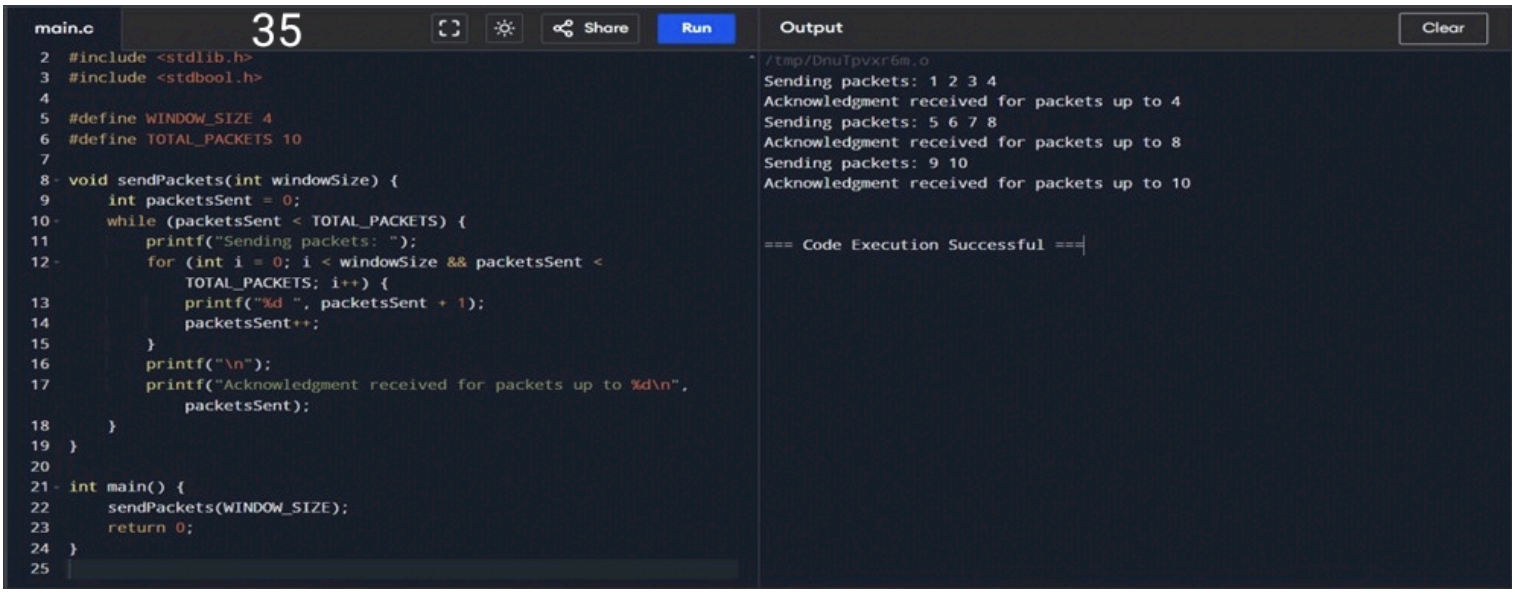

main.c
33 (Client)
Share
Run
Output
Clear
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <arpa/inet.h>
#include <unistd.h>
#define PORT 8080
#define BUFFER_SIZE 1024
int main() {
int sock = 0;
struct sockaddr_in serv_addr;
char buffer[BUFFER_SIZE] = {0};
FILE *file_to_send;
int bytes_read;
// Create socket
if ((sock = socket(AF_INET, SOCK_STREAM, 0)) < 0) {
printf("\n Socket creation error \n");
return -1;
}
// Set server address structure
serv_addr.sin_family = AF_INET;
serv_addr.sin_port = htons(PORT);
/tmp/icGZYMLTOl.o
Connection Failed
=== Code Exited With Errors ===

main.c 34 Share Run

```c
#include <stdio.h>
#include <string.h>

#define POLYNOMIAL 0x07

unsigned char compute_crc(const char *data) {
    unsigned char crc = 0; // Initial CRC value
    size_t len = strlen(data);

    for (size_t i = 0; i < len; i++) {
        crc ^= (data[i] & 0xFF);

        for (int j = 0; j < 8; j++) {
            if (crc & 0x80) {
                crc = (crc << 1) ^ POLYNOMIAL;
            } else {
                crc <<= 1;
            }
        }
    }
    return crc;
}

void simulate_error(char *data) {
    size_t len = strlen(data);
    if (len > 0) {
```

Output Clear

```
/tmp/GJirRrSc5d.o
ERROR!
Original Data: Hello, World!
Computed CRC: 0x87
Data after error: Iello, World!
Error detected in data!


=== Code Execution Successful ===
```


main.c
35
Share
Run
Output
Clear
#include <stdlib.h>
#include <stdbool.h>
#define WINDOW_SIZE 4
#define TOTAL_PACKETS 10
void sendPackets(int windowSize) {
int packetsSent = 0;
while (packetsSent < TOTAL_PACKETS) {
printf("Sending packets: ");
for (int i = 0; i < windowSize && packetsSent < TOTAL_PACKETS; i++) {
printf("%d ", packetsSent + 1);
packetsSent++;
}
printf("\n");
printf("Acknowledgment received for packets up to %d\n", packetsSent);
}
}
int main() {
sendPackets(WINDOW_SIZE);
return 0;
}
/tmp/DnuTpvxr6m.o
Sending packets: 1 2 3 4
Acknowledgment received for packets up to 4
Sending packets: 5 6 7 8
Acknowledgment received for packets up to 8
Sending packets: 9 10
Acknowledgment received for packets up to 10
=== Code Execution Successful ===