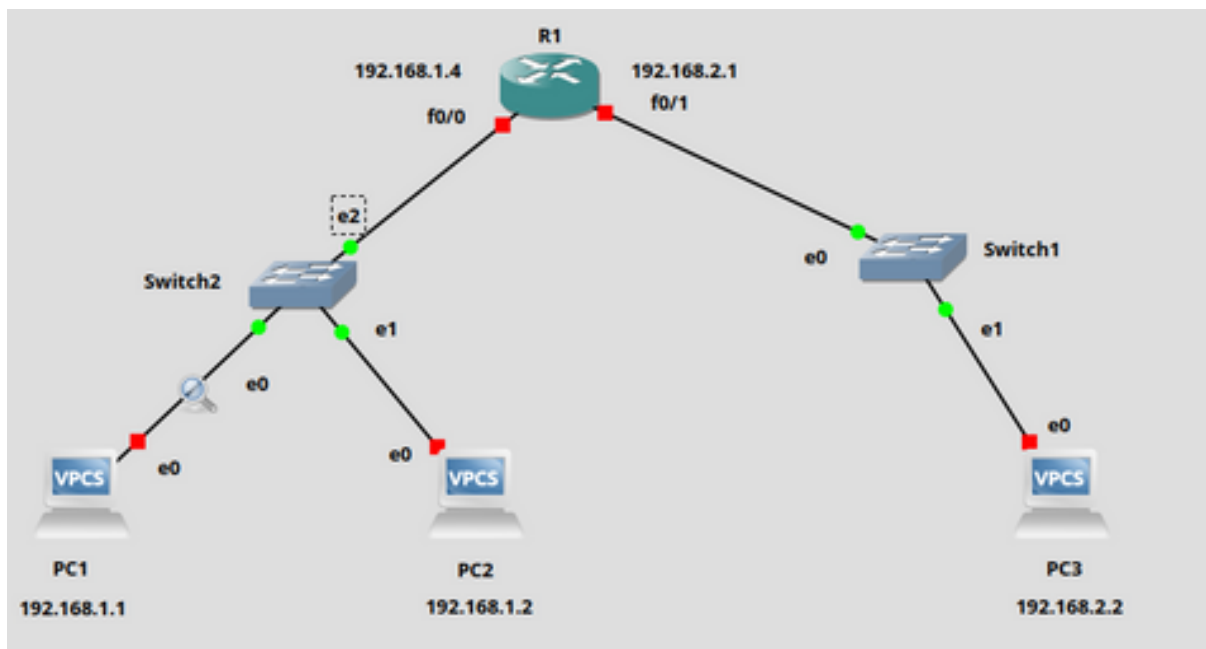


### 3. TCP/UDP Intersection Module: Simulate coexistence, monitoring, and analysis of TCP and UDP traffic in the network.

#### # Background Research for the problem statement:

- Understood TCP and UDP protocol and the fundamental differences between them (connection-oriented and connectionless).
- Understood the TCP and UDP packet structure.
- How Wireshark can dissect and interpret packet headers for both protocols.
- Identified well-known ports for TCP and UDP services (eg. HTTPS, FTP)
- Learned how to set up filters and search for specific packets.



#### # Observations:

- **TCP and UDP Traffic Identification:**

- **Protocol Differentiation:**

Successfully identified and differentiated between TCP and UDP traffic in the Wireshark captures.

Recognized the unique characteristics of TCP and UDP packets.

- **Port Numbers:**

Associated port numbers with specific protocols to understand the nature of the traffic.

Used port numbers to identify common applications or services.

```
-l size           Data size
-P protocol       Use IP protocol in ping packets
                   1 - ICMP (default), 17 - UDP, 6 - TCP
-p port           Destination port
-s port           Source port
-T ttl            Set ttl, default 64
-t               Send packets until interrupted by Ctrl+C
-w ms            Wait ms milliseconds to receive the response
```

Notes: 1. Using names requires DNS to be set.  
2. Use Ctrl+C to stop the command.

```
PC1> ping 192.168.2.2 -t -P 6
```

```
Connect  7@192.168.2.2 seq=1 ttl=63 time=2020.947 ms
SendData 7@192.168.2.2 seq=1 ttl=63 time=16.760 ms
Close    7@192.168.2.2 seq=1 ttl=63 time=18.858 ms
Connect  7@192.168.2.2 seq=2 ttl=63 time=13.363 ms
SendData 7@192.168.2.2 seq=2 ttl=63 time=22.416 ms
Close    7@192.168.2.2 seq=2 ttl=63 time=22.690 ms
Connect  7@192.168.2.2 seq=3 ttl=63 time=15.109 ms
SendData 7@192.168.2.2 seq=3 ttl=63 time=21.718 ms
Close    7@192.168.2.2 seq=3 ttl=63 time=22.810 ms
Connect  7@192.168.2.2 seq=4 ttl=63 time=13.615 ms
SendData 7@192.168.2.2 seq=4 ttl=63 time=12.245 ms
Close    7@192.168.2.2 seq=4 ttl=63 time=13.501 ms
```

```

-P protocol    Use IP protocol in ping packets
                  1 - ICMP (default), 17 - UDP, 6 - TCP
-p port        Destination port
-s port        Source port
-T ttl         Set ttl, default 64
-t             Send packets until interrupted by Ctrl+C
-w ms         Wait ms milliseconds to receive the response

```

Notes: 1. Using names requires DNS to be set.  
2. Use Ctrl+C to stop the command.

```
PC1> ping 192.168.2.2 -t -P 17
```

```

84 bytes from 192.168.2.2 udp_seq=1 ttl=63 time=29.744 ms
84 bytes from 192.168.2.2 udp_seq=2 ttl=63 time=13.978 ms
84 bytes from 192.168.2.2 udp_seq=3 ttl=63 time=13.718 ms
84 bytes from 192.168.2.2 udp_seq=4 ttl=63 time=13.171 ms
84 bytes from 192.168.2.2 udp_seq=5 ttl=63 time=13.392 ms
84 bytes from 192.168.2.2 udp_seq=6 ttl=63 time=14.303 ms

```

## UDP

The image shows a Wireshark packet capture of a UDP Echo service. The packet list on the left shows 15 packets (No. 1103 to 1125) alternating between requests from 192.168.1.1 to 192.168.2.2 and responses from 192.168.2.2 to 192.168.1.1. The selected packet (No. 1057) is a response from 192.168.2.2 to 192.168.1.1. The packet details pane shows the following structure:

- Frame 1057: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface -, id 0
- Ethernet II, Src: Private\_66:68:00 (00:50:79:66:68:00), Dst: cc:01:32:1a:00:00 (cc:01:32:1a:00:00)
- Internet Protocol Version 4, Src: 192.168.1.1, Dst: 192.168.2.2
- User Datagram Protocol, Src Port: 46138, Dst Port: 7
- Echo

The packet bytes pane shows the raw data in hexadecimal and ASCII:

```

0000 cc 01 32 1a 00 00 00 50 79 66 68 00 00 00 45 00  --2...P yfh...E-
0010 00 54 89 11 00 00 40 11 6d 34 c0 a8 01 01 c0 a8  T...@. m4.....
0020 02 02 b4 3a 00 07 00 40 2b 4f 00 50 79 66 68 00  ...:..@ +0 Pyfh-
0030 0e 0f 10 11 12 13 14 15 16 17 18 19 1a 1b 1c 1d  .....
0040 1e 1f 20 21 22 23 24 25 26 27 28 29 2a 2b 2c 2d  .. !"#% &'()*+,-
0050 2e 2f 30 31 32 33 34 35 36 37 38 39 3a 3b 3c 3d  ./012345 6789:;<=
0060 3e 3f

```

## TCP

No.	Time	Source	Destination	Protocol	Length	Info
681	3238.528210	192.168.2.2	192.168.1.1	TCP	54	7 → 21882 [ACK] Seq=1 Ack=57 Win=2920 Len=0
682	3238.537827	192.168.1.1	192.168.2.2	TCP	66	21882 → 7 [FIN, PSH, ACK] Seq=57 Ack=1 Win=2920 Len=0 TSval=1704560857 TSecr=0
683	3238.558825	192.168.2.2	192.168.1.1	TCP	54	7 → 21882 [ACK] Seq=1 Ack=58 Win=2920 Len=0
684	3238.558981	192.168.2.2	192.168.1.1	TCP	54	7 → 21882 [FIN, ACK] Seq=1 Ack=58 Win=2920 Len=0
685	3238.562069	192.168.1.1	192.168.2.2	TCP	66	21882 → 7 [ACK] Seq=58 Ack=2 Win=2920 Len=0 TSval=1704560857 TSecr=0
686	3239.502200	192.168.1.1	192.168.2.2	TCP	74	[TCP Port numbers reused] 21882 → 7 [SYN] Seq=0 Win=2920 Len=0 MSS=1460 TSval=1704560858 TSecr=0 WS=2
687	3239.537807	192.168.2.2	192.168.1.1	TCP	54	7 → 21882 [SYN, ACK] Seq=0 Ack=1 Win=2920 Len=0
688	3239.574945	192.168.1.1	192.168.2.2	TCP	66	21882 → 7 [ACK] Seq=1 Ack=1 Win=2920 Len=0 TSval=1704560858 TSecr=0
689	3239.581970	192.168.1.1	192.168.2.2	ECHO	122	Request
690	3239.594192	192.168.2.2	192.168.1.1	TCP	54	7 → 21882 [ACK] Seq=1 Ack=57 Win=2920 Len=0
691	3239.602937	192.168.1.1	192.168.2.2	TCP	66	21882 → 7 [FIN, PSH, ACK] Seq=57 Ack=1 Win=2920 Len=0 TSval=1704560858 TSecr=0
692	3239.614566	192.168.2.2	192.168.1.1	TCP	54	7 → 21882 [ACK] Seq=1 Ack=58 Win=2920 Len=0
693	3239.618873	192.168.2.2	192.168.1.1	TCP	54	7 → 21882 [FIN, ACK] Seq=1 Ack=58 Win=2920 Len=0
694	3239.616732	192.168.1.1	192.168.2.2	TCP	66	21882 → 7 [ACK] Seq=58 Ack=2 Win=2920 Len=0 TSval=1704560858 TSecr=0
695	3240.617234	192.168.1.1	192.168.2.2	TCP	74	[TCP Port numbers reused] 21882 → 7 [SYN] Seq=0 Win=2920 Len=0 MSS=1460 TSval=1704560859 TSecr=0 WS=2
696	3240.636774	192.168.2.2	192.168.1.1	TCP	54	7 → 21882 [SYN, ACK] Seq=0 Ack=1 Win=2920 Len=0
697	3240.638124	192.168.1.1	192.168.2.2	TCP	66	21882 → 7 [ACK] Seq=1 Ack=1 Win=2920 Len=0 TSval=1704560859 TSecr=0
698	3240.650197	192.168.1.1	192.168.2.2	ECHO	122	Request
699	3240.666954	192.168.2.2	192.168.1.1	TCP	54	7 → 21882 [ACK] Seq=1 Ack=57 Win=2920 Len=0
700	3240.679841	192.168.1.1	192.168.2.2	TCP	66	21882 → 7 [FIN, PSH, ACK] Seq=57 Ack=1 Win=2920 Len=0 TSval=1704560859 TSecr=0
701	3240.697425	192.168.2.2	192.168.1.1	TCP	54	7 → 21882 [ACK] Seq=1 Ack=58 Win=2920 Len=0
702	3240.697579	192.168.2.2	192.168.1.1	TCP	54	7 → 21882 [FIN, ACK] Seq=1 Ack=58 Win=2920 Len=0
703	3240.700239	192.168.1.1	192.168.2.2	TCP	66	21882 → 7 [ACK] Seq=58 Ack=2 Win=2920 Len=0 TSval=1704560859 TSecr=0

Frame 105: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface -, id 0	0000	cc 01 32 1a 00 00 00 50	79 66 68 00 08 00 45 00	..2...P yfh...E
Ethernet II, Src: Private_66:68:00 (00:50:79:66:68:00), Dst: cc:01:32:1a:00:00 (cc:01:32:1a:00:00)	0010	00 3c 88 8f 00 00 40 06	6d d9 c0 a8 01 01 c0 a8	<...@. # .....>
Internet Protocol Version 4, Src: 192.168.1.1, Dst: 192.168.2.2	0020	02 02 55 7a 00 07 2f cf	27 45 00 00 00 a0 02	..uz../. E .....
Transmission Control Protocol, Src Port: 21882, Dst Port: 7, Seq: 0, Len: 0	0030	0b 68 20 8d 00 00 02 04	05 b4 01 01 08 0a 65 99	.h .....e.....
	0040	88 8f 00 00 00 01 03 03	01	.....

## • Traffic Patterns:

- Pattern Analysis:

Observed traffic patterns for both TCP and UDP.

Identified characteristics such as connection-oriented behavior in TCP and connectionless nature in UDP.

- Wireshark Filters:

Successfully applied capture filters to focus on capturing only TCP or UDP traffic.

Explored the effectiveness of filters in isolating specific types of traffic.