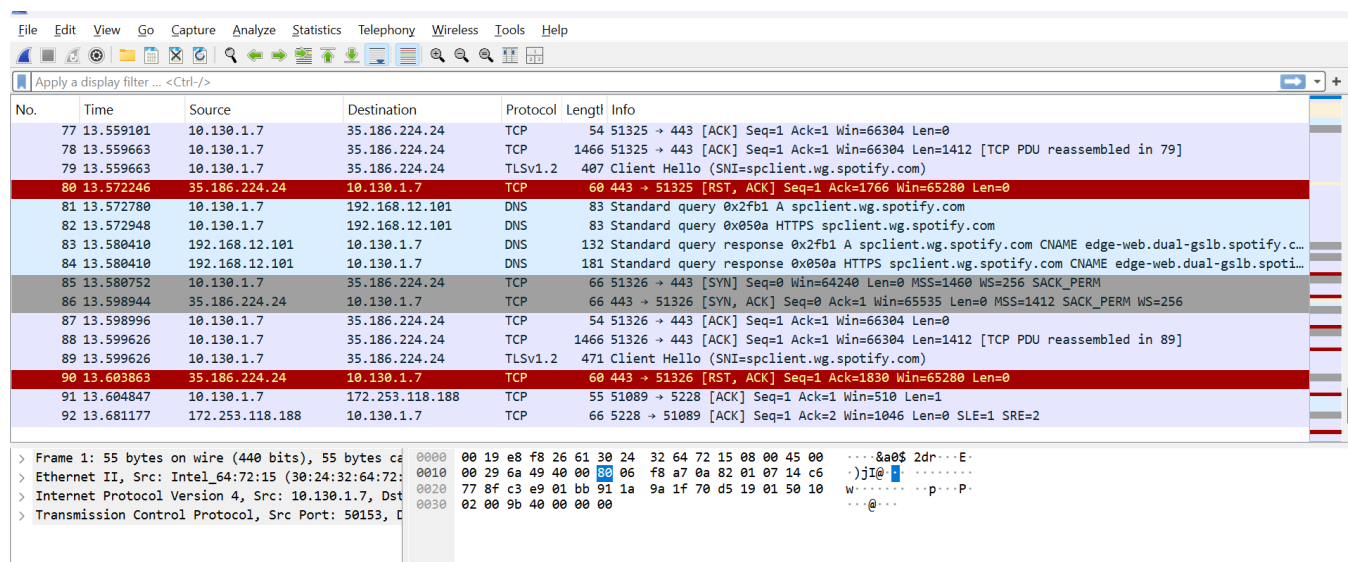
 <b>Marwadi University</b> Marwadi Chandarana Group	<b>Marwadi University</b> <b>Faculty of Engineering and Technology</b> <b>Department of Information and Communication Technology</b>	
<b>Subject: Computer Networks (01CT0503)</b>	<b>Aim:</b> Monitor the live/real time network and analyze the concepts of various networking protocols like IP, TCP, UDP, etc.	
<b>Experiment No: 11</b>	<b>Date: 17/11/2024</b>	<b>Enrolment No: 92200133029</b>

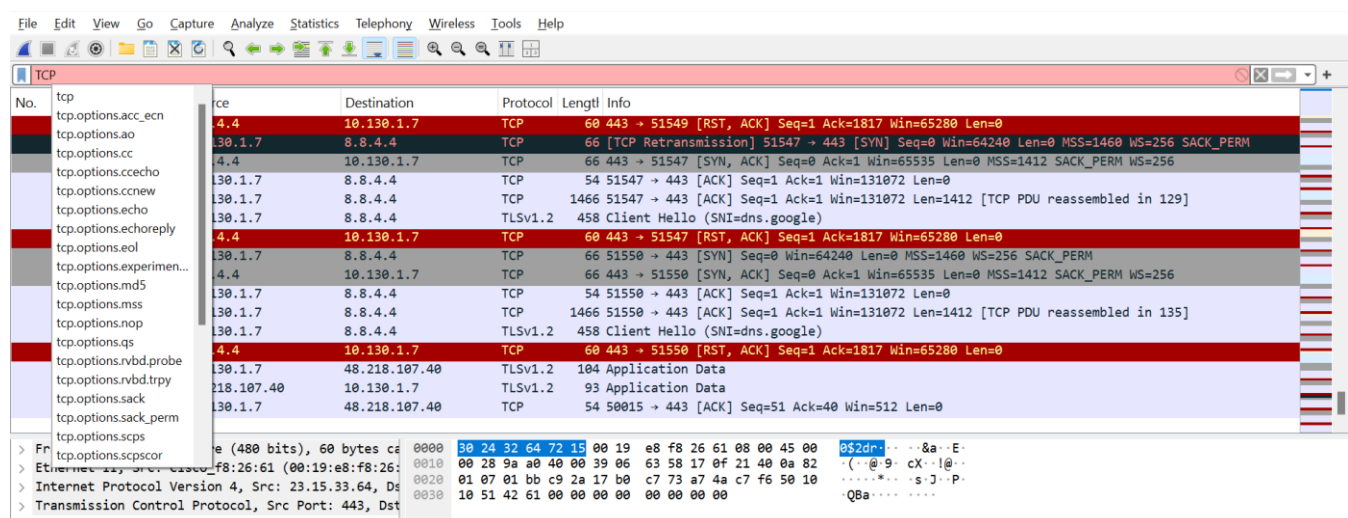
**Aim : Monitor the live/real time network and analyze the concepts of various networking protocols like IP, TCP, UDP, etc.**

Step – 1 : Open the wireshark. After opening Wireshark, you will see several options on your screen. Select the **Wi-Fi interface** to start capturing packets. Once the capture begins, you can observe various types of packets being routed, similar to those shown in the image.




No.	Time	Source	Destination	Protocol	Length	Info
77	13.559101	10.130.1.7	35.186.224.24	TCP	54	51325 → 443 [ACK] Seq=1 Ack=1 Win=66304 Len=0
78	13.559663	10.130.1.7	35.186.224.24	TCP	1466	51325 → 443 [ACK] Seq=1 Ack=1 Win=66304 Len=1412 [TCP PDU reassembled in 79]
79	13.559663	10.130.1.7	35.186.224.24	TLSv1.2	407	Client Hello (SNI=spotify.wg.spotify.com)
80	13.572246	35.186.224.24	10.130.1.7	TCP	60	443 → 51325 [RST, ACK] Seq=1 Ack=1766 Win=65280 Len=0
81	13.572780	10.130.1.7	192.168.12.101	DNS	83	Standard query 0x2fb1 A spotify.wg.spotify.com
82	13.572948	10.130.1.7	192.168.12.101	DNS	83	Standard query response 0x2fb1 A spotify.wg.spotify.com CNAME edge-web.dual-gslb.spotify.c...
83	13.580410	192.168.12.101	10.130.1.7	DNS	132	Standard query response 0x2fb1 A spotify.wg.spotify.com CNAME edge-web.dual-gslb.spotify.c...
84	13.580410	192.168.12.101	10.130.1.7	DNS	181	Standard query response 0x050a HTTPS spotify.wg.spotify.com CNAME edge-web.dual-gslb.spotify.c...
85	13.580752	10.130.1.7	35.186.224.24	TCP	66	51326 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
86	13.598944	35.186.224.24	10.130.1.7	TCP	66	443 → 51326 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1412 SACK_PERM WS=256
87	13.598996	10.130.1.7	35.186.224.24	TCP	54	51326 → 443 [ACK] Seq=1 Ack=1 Win=66304 Len=0
88	13.599626	10.130.1.7	35.186.224.24	TCP	1466	51326 → 443 [ACK] Seq=1 Ack=1 Win=66304 Len=1412 [TCP PDU reassembled in 89]
89	13.599626	10.130.1.7	35.186.224.24	TLSv1.2	471	Client Hello (SNI=spotify.wg.spotify.com)
90	13.603863	35.186.224.24	10.130.1.7	TCP	60	443 → 51326 [RST, ACK] Seq=1 Ack=1830 Win=65280 Len=0
91	13.604847	10.130.1.7	172.253.118.188	TCP	55	51089 → 5228 [ACK] Seq=1 Ack=1 Win=510 Len=1
92	13.681177	172.253.118.188	10.130.1.7	TCP	66	5228 → 51089 [ACK] Seq=1 Ack=2 Win=1046 Len=0 SLE=1 SRE=2

Step – 2 : Next, go to the **filter bar** in Wireshark and enter **tcp** as the filter. By applying this filter, you will see only the packets related to the **TCP protocol**, as shown in the image.



No.	Time	Source	Destination	Protocol	Length	Info
4.4		10.130.1.7	8.8.4.4	TCP	60	443 → 51549 [RST, ACK] Seq=1 Ack=1817 Win=65280 Len=0
30.1.7		10.130.1.7	8.8.4.4	TCP	66	[TCP Retransmission] 51547 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
4.4		10.130.1.7	8.8.4.4	TCP	66	443 → 51547 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1412 SACK_PERM WS=256
130.1.7		10.130.1.7	8.8.4.4	TCP	54	51547 → 443 [ACK] Seq=1 Ack=1 Win=131072 Len=0
130.1.7		10.130.1.7	8.8.4.4	TCP	1466	51547 → 443 [ACK] Seq=1 Ack=1 Win=131072 Len=1412 [TCP PDU reassembled in 129]
130.1.7		10.130.1.7	8.8.4.4	TLSv1.2	458	Client Hello (SNI=dns.google)
4.4		10.130.1.7	8.8.4.4	TCP	60	443 → 51547 [RST, ACK] Seq=1 Ack=1817 Win=65280 Len=0
130.1.7		10.130.1.7	8.8.4.4	TCP	66	51550 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
4.4		10.130.1.7	8.8.4.4	TCP	66	443 → 51550 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1412 SACK_PERM WS=256
130.1.7		10.130.1.7	8.8.4.4	TCP	54	51550 → 443 [ACK] Seq=1 Ack=1 Win=131072 Len=0
130.1.7		10.130.1.7	8.8.4.4	TCP	1466	51550 → 443 [ACK] Seq=1 Ack=1 Win=131072 Len=1412 [TCP PDU reassembled in 135]
130.1.7		10.130.1.7	8.8.4.4	TLSv1.2	458	Client Hello (SNI=dns.google)
4.4		10.130.1.7	8.8.4.4	TCP	60	443 → 51550 [RST, ACK] Seq=1 Ack=1817 Win=65280 Len=0
130.1.7		48.218.107.40	10.130.1.7	TLSv1.2	104	Application Data
118.107.40		10.130.1.7	48.218.107.40	TLSv1.2	93	Application Data
130.1.7		48.218.107.40	10.130.1.7	TCP	54	50015 → 443 [ACK] Seq=51 Ack=40 Win=512 Len=0

Here we can see detailed information of TCP that how it routes the packets and what ack it gives etc.

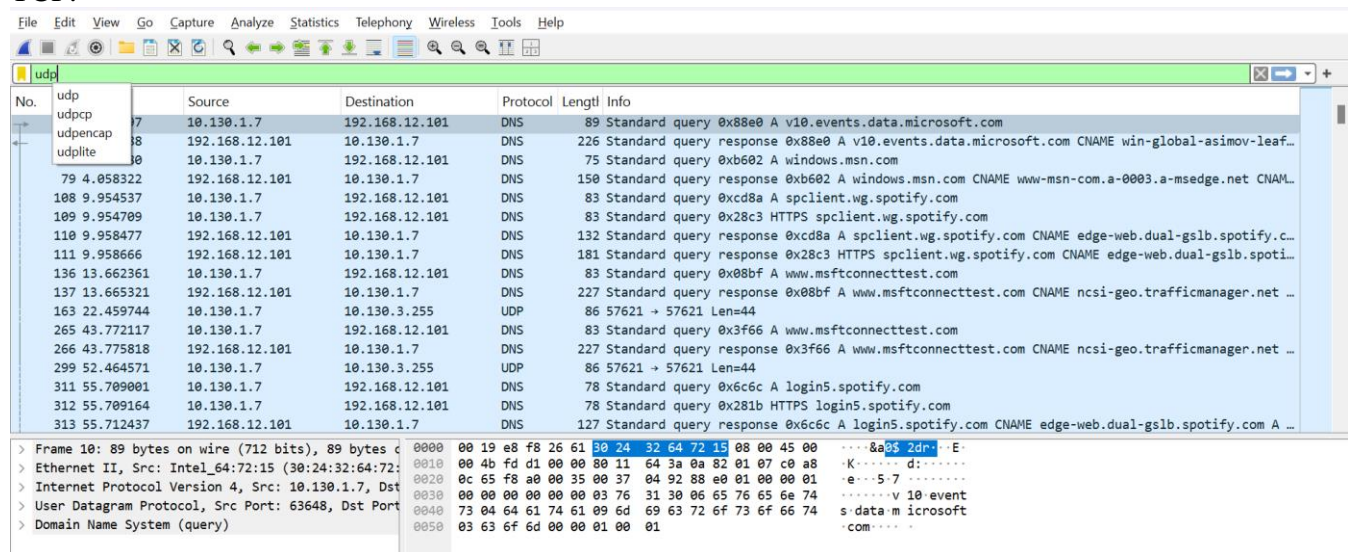
 <b>Marwadi University</b> Marwadi Chandarana Group	<b>Marwadi University</b> <b>Faculty of Engineering and Technology</b> <b>Department of Information and Communication Technology</b>	
<b>Subject: Computer Networks (01CT0503)</b>	<b>Aim:</b> Monitor the live/real time network and analyze the concepts of various networking protocols like IP, TCP, UDP, etc.	
<b>Experiment No: 11</b>	<b>Date: 17/11/2024</b>	<b>Enrolment No: 92200133029</b>

```

✓ Transmission Control Protocol, Src Port: 443, Dst Port: 52886, Seq: 1941, Ack: 7085, Len
  Source Port: 443
  Destination Port: 52886
  [Stream index: 4]
  [Stream Packet Number: 22]
  > [Conversation completeness: Incomplete (28)]
  [TCP Segment Len: 0]
  Sequence Number: 1941 (relative sequence number)
  Sequence Number (raw): 1252068900
  [Next Sequence Number: 1941 (relative sequence number)]
  Acknowledgment Number: 7085 (relative ack number)
  Acknowledgment number (raw): 2161475922
  0101 .... = Header Length: 20 bytes (5)
  > Flags: 0x010 (ACK)

```


Step – 3 : Now, in the **filter bar**, type **udp** and apply the filter. This will display only packets using the **UDP protocol**. In the image, you can see DNS packets because DNS (Domain Name System) typically operates over UDP for queries and responses, as it is faster and requires fewer resources compared to TCP.



The screenshot shows the Wireshark network protocol analyzer. The filter bar at the top is set to 'udp'. The packet list pane displays a series of DNS packets between 10.130.1.7 and 192.168.12.101. The packet details pane for the selected packet (No. 10) shows the following structure:

- Frame 10: 89 bytes on wire (712 bits), 89 bytes captured (712 bits) on interface 0
- Ethernet II, Src: Intel\_64:72:15 (30:24:32:64:72:15), Dst: 10.130.1.7
- Internet Protocol Version 4, Src: 10.130.1.7, Dst: 192.168.12.101
- User Datagram Protocol, Src Port: 63648, Dst Port: 53
- Domain Name System (query)
  - Standard query query 0x8000 A v10.events.data.microsoft.com CNAME win-global-asimov-leaf...

Here you can see various information about udp like udppayload and flags etc.

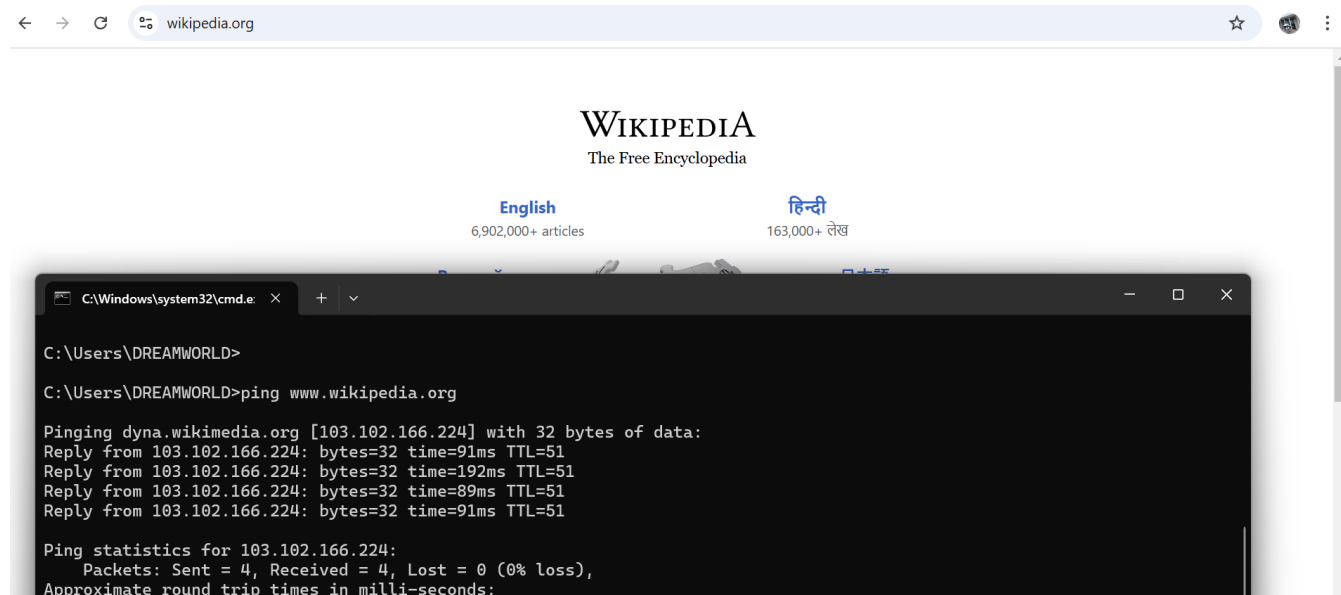
 <b>Marwadi University</b> Marwadi Chandarana Group	<b>Marwadi University</b> <b>Faculty of Engineering and Technology</b> <b>Department of Information and Communication Technology</b>	
<b>Subject: Computer Networks (01CT0503)</b>	<b>Aim:</b> Monitor the live/real time network and analyze the concepts of various networking protocols like IP, TCP, UDP, etc.	
<b>Experiment No: 11</b>	<b>Date: 17/11/2024</b>	<b>Enrolment No: 92200133029</b>

```


[Stream index: 0]
[Stream Packet Number: 1]
> [Timestamps]
UDP payload (48 bytes)
✓ Domain Name System (query)
  Transaction ID: 0x9afd
  > Flags: 0x0100 Standard query
  Questions: 1
  Answer RRs: 0
  Authority RRs: 0
  Additional RRs: 0
  > Queries
    [Response In: 56]

```

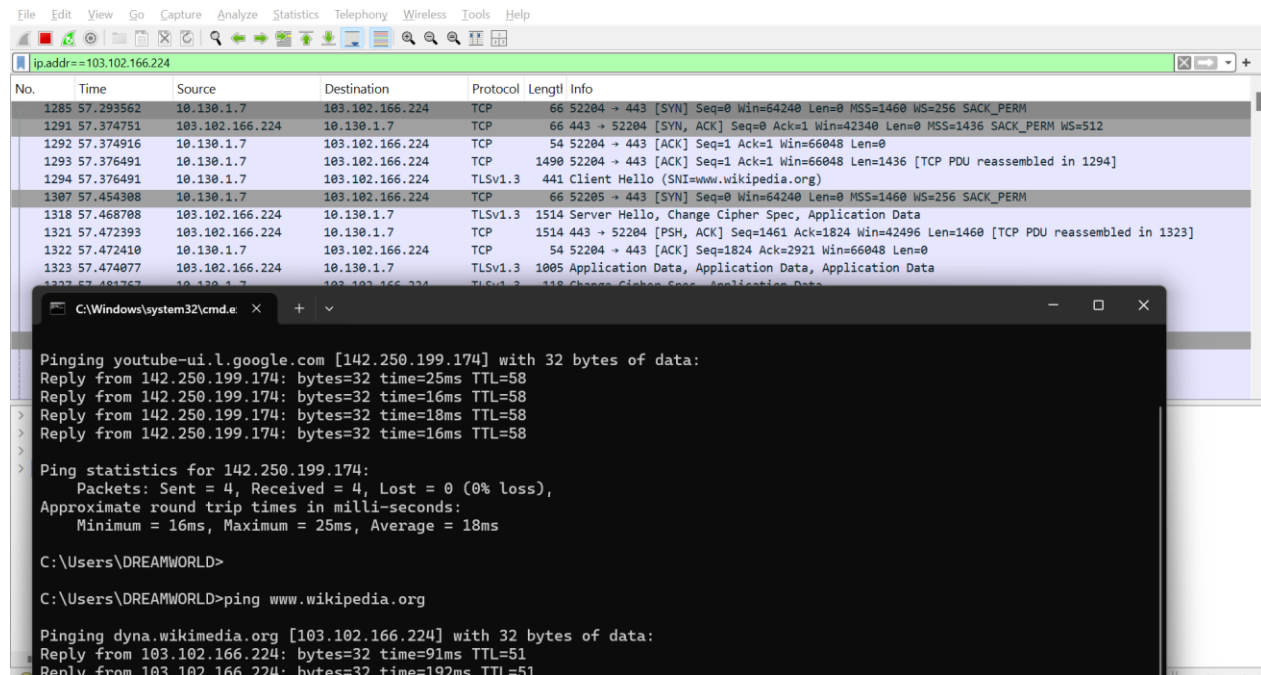
Step – 4 : To view the **IP protocol in action**, I opened the **Wikipedia website** and launched the **Command Prompt**. I then used the **ping command**, which is commonly used to check the routing of packets and measure the response time between your device and the target server. From this command, I was able to retrieve the **IP address of Wikipedia**.



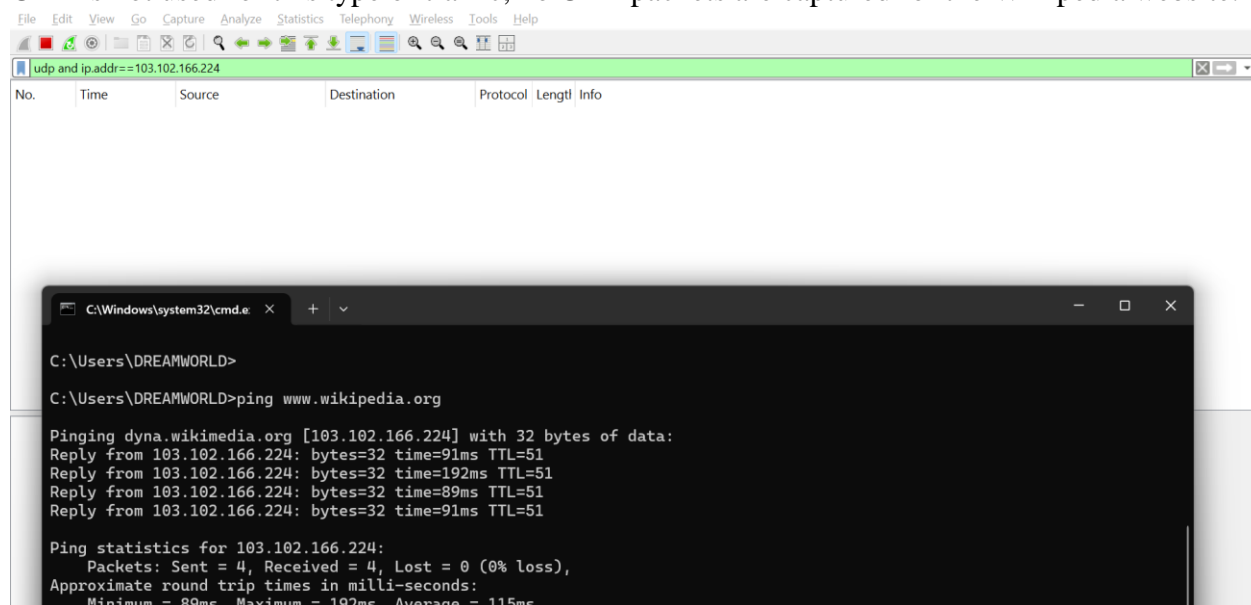
**Step – 5 :** Using the retrieved **IP address of Wikipedia**, go to the **filter bar** in Wireshark and type **ip.addr == [Wikipedia's IP address]** . This filter will show all the packets routed to and from the specified IP address, allowing you to analyze how the packets are being routed and the communication

 <b>Marwadi University</b> Marwadi Chandarana Group	<b>Marwadi University</b> <b>Faculty of Engineering and Technology</b> <b>Department of Information and Communication Technology</b>	
<b>Subject: Computer Networks (01CT0503)</b>	<b>Aim:</b> Monitor the live/real time network and analyze the concepts of various networking protocols like IP, TCP, UDP, etc.	
<b>Experiment No: 11</b>	<b>Date: 17/11/2024</b>	<b>Enrolment No: 92200133029</b>

details.




**Step – 6 :** When you enter udp in the filter bar for the Wikipedia website, no packets are displayed because Wikipedia primarily uses the **TCP protocol** for communication. Websites like Wikipedia rely on **HTTP/HTTPS**, which are based on TCP, as TCP ensures reliable, ordered delivery of data. Since UDP is not used for this type of traffic, no UDP packets are captured for the Wikipedia website.



## Conclusion :

In this Wireshark experiment, I analyzed network traffic using different protocols such as TCP, UDP, and IP. By applying filters for TCP and UDP, I observed how TCP ensures reliable, ordered data

 <b>Marwadi University</b> Marwadi Chandarana Group	<b>Marwadi University</b> <b>Faculty of Engineering and Technology</b> <b>Department of Information and Communication Technology</b>	
<b>Subject: Computer Networks (01CT0503)</b>	<b>Aim:</b> Monitor the live/real time network and analyze the concepts of various networking protocols like IP, TCP, UDP, etc.	
<b>Experiment No: 11</b>	<b>Date: 17/11/2024</b>	<b>Enrolment No: 92200133029</b>

transmission, making it ideal for applications like web browsing, while UDP, being faster and connectionless, is typically used for applications like DNS queries. I also observed the routing of IP packets when accessing a website like Wikipedia. Since Wikipedia uses the TCP protocol for its HTTP/HTTPS communication, no UDP packets were captured, highlighting the role of TCP in ensuring reliable web traffic. This experiment provided valuable insights into how different protocols function and interact within live networks.