Practical – 8

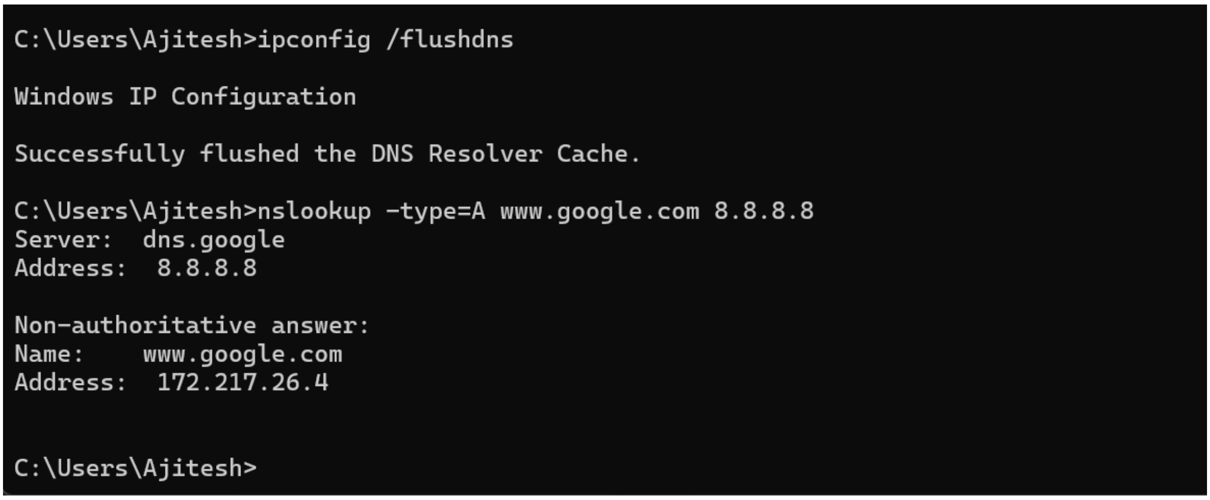
Aim of the Practical :- To analyse network traffic using Wireshark – Capture and Study TCP and UDP Packets

Practical:

The objective of this lab is to practically explore and understand the structure of two core protocols of the transport layer: the User Datagram Protocol (UDP) and the Transmission Control Protocol (TCP). By capturing live network traffic with Wireshark, we can analyse the headers of each protocol to see how they function in real-world applications like DNS and web browsing.

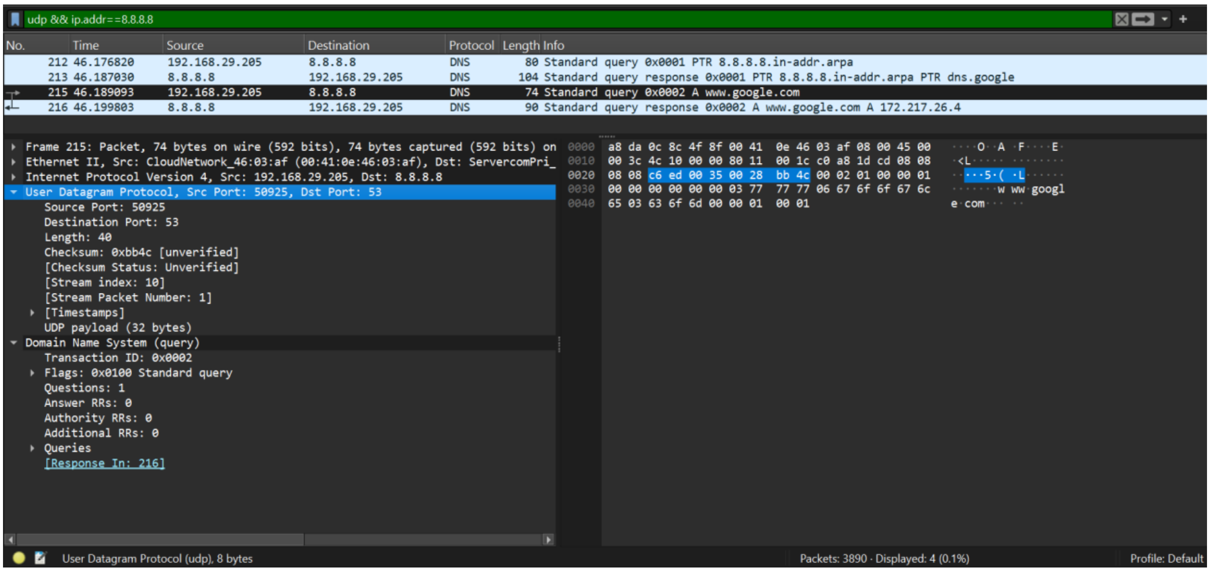
Task 1: Generate UDP Packet using DNS and Analyse It

This section details the analysis of a UDP packet generated by a Domain Name System (DNS) query. UDP is used for DNS because it is a fast, connectionless protocol suitable for quick request-response transactions.



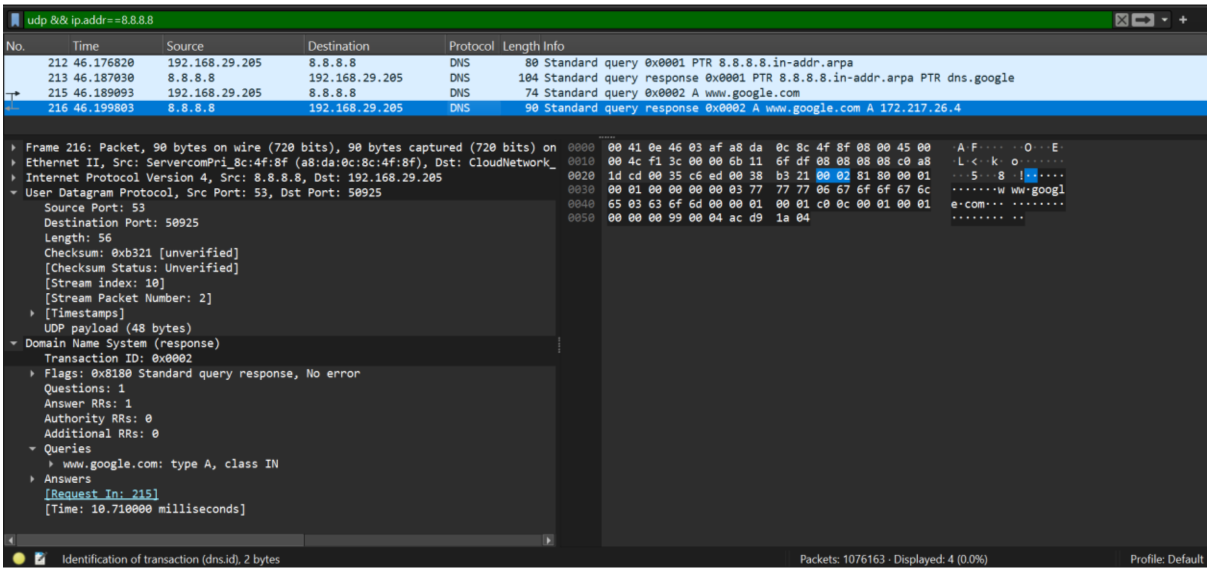
```
C:\Users\Ajitesh>ipconfig /flushdns

Windows IP Configuration

Successfully flushed the DNS Resolver Cache.

C:\Users\Ajitesh>nslookup -type=A www.google.com 8.8.8.8
Server:  dns.google
Address:  8.8.8.8

Non-authoritative answer:
Name:    www.google.com
Address:  172.217.26.4


C:\Users\Ajitesh>
```

Image 1: This Command Prompt screenshot shows the DNS cache being cleared using *ipconfig /flushdns*. A *nslookup* command is then used to successfully resolve *www.google.com* to its IP address.



```
udp && ip.addr==8.8.8.8

No.  Time       Source          Destination     Protocol Length Info
 212 46.176820  192.168.29.205  8.8.8.8         DNS        80 Standard query 0x0001 PTR 8.8.8.8.in-addr.arpa
 213 46.187030  8.8.8.8         192.168.29.205  DNS       104 Standard query response 0x0001 PTR 8.8.8.8.in-addr.arpa PTR dns.google
 215 46.189093  192.168.29.205  8.8.8.8         DNS        74 Standard query 0x0002 A www.google.com
 216 46.199803  8.8.8.8         192.168.29.205  DNS        90 Standard query response 0x0002 A www.google.com A 172.217.26.4

Frame 215: Packet, 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on
Ethernet II, Src: CloudNetwork_46:03:af (00:41:0e:46:03:af), Dst: ServercomPri_
Internet Protocol Version 4, Src: 192.168.29.205, Dst: 8.8.8.8
User Datagram Protocol, Src Port: 50925, Dst Port: 53
    Source Port: 50925
    Destination Port: 53
    Length: 40
    Checksum: 0xbb4c [unverified]
    [Checksum Status: Unverified]
    [Stream index: 10]
    [Stream Packet Number: 1]
  > [Timestamps]
    UDP payload (32 bytes)
Domain Name System (query)
    Transaction ID: 0x0002
  > Flags: 0x0100 Standard query
    Questions: 1
    Answer RRs: 0
    Authority RRs: 0
    Additional RRs: 0
  > Queries
    [Response In: 216]

0000  a8 da 0c 8c 4f 8f 00 41  0e 46 03 af 08 00 45 00   ····O·A ·F····E·
0010  00 3c 4c 10 00 00 80 11  00 1c c0 a8 1d cd 08 08   ·<L····· ········
0020  08 08 c6 ed 00 35 00 28  bb 4c 00 02 01 00 00 01   ·····5·( ·L······
0030  00 00 00 00 00 00 03 77  77 77 06 67 6f 6f 67 6c   ·······w ww·googl
0040  65 03 63 6f 6d 00 00 01  00 01                     e·com··· ··

User Datagram Protocol (udp), 8 bytes          Packets: 3890 · Displayed: 4 (0.1%)          Profile: Default
```

Image 2: This Wireshark capture displays the outgoing DNS query packet sent over UDP.



```
udp && ip.addr==8.8.8.8

No.  Time       Source          Destination     Protocol Length Info
 212 46.176820  192.168.29.205  8.8.8.8         DNS        80 Standard query 0x0001 PTR 8.8.8.8.in-addr.arpa
 213 46.187030  8.8.8.8         192.168.29.205  DNS       104 Standard query response 0x0001 PTR 8.8.8.8.in-addr.arpa PTR dns.google
 215 46.189093  192.168.29.205  8.8.8.8         DNS        74 Standard query 0x0002 A www.google.com
 216 46.199803  8.8.8.8         192.168.29.205  DNS        90 Standard query response 0x0002 A www.google.com A 172.217.26.4

Frame 216: Packet, 90 bytes on wire (720 bits), 90 bytes captured (720 bits) on
Ethernet II, Src: ServercomPri_8c:4f:8f (a8:da:0c:8c:4f:8f), Dst: CloudNetwork_
Internet Protocol Version 4, Src: 8.8.8.8, Dst: 192.168.29.205
User Datagram Protocol, Src Port: 53, Dst Port: 50925
    Source Port: 53
    Destination Port: 50925
    Length: 56
    Checksum: 0xb321 [unverified]
    [Checksum Status: Unverified]
    [Stream index: 10]
    [Stream Packet Number: 2]
  > [Timestamps]
    UDP payload (48 bytes)
Domain Name System (response)
    Transaction ID: 0x0002
  > Flags: 0x8180 Standard query response, No error
    Questions: 1
    Answer RRs: 1
    Authority RRs: 0
    Additional RRs: 0
    Queries
      > www.google.com: type A, class IN
  > Answers
    [Request In: 215]
    [Time: 10.710000 milliseconds]

0000  00 41 0e 46 03 af a8 da  0c 8c 4f 8f 08 00 45 00   ·A·F···· ··O···E·
0010  00 4c f1 3c 00 00 6b 11  6f df 08 08 08 08 c0 a8   ·L·<··k· o·······
0020  1d cd 00 35 c6 ed 00 38  b3 21 00 02 81 80 00 01   ···5···8 ·!······
0030  00 01 00 00 00 00 03 77  77 77 06 67 6f 6f 67 6c   ·······w ww·googl
0040  65 03 63 6f 6d 00 00 01  00 01 c0 0c 00 01 00 01   e·com··· ········
0050  00 00 00 99 00 04 ac d9  1a 04                     ········ ··

Identification of transaction (dns.id), 2 bytes          Packets: 1076163 · Displayed: 4 (0.0%)          Profile: Default
```

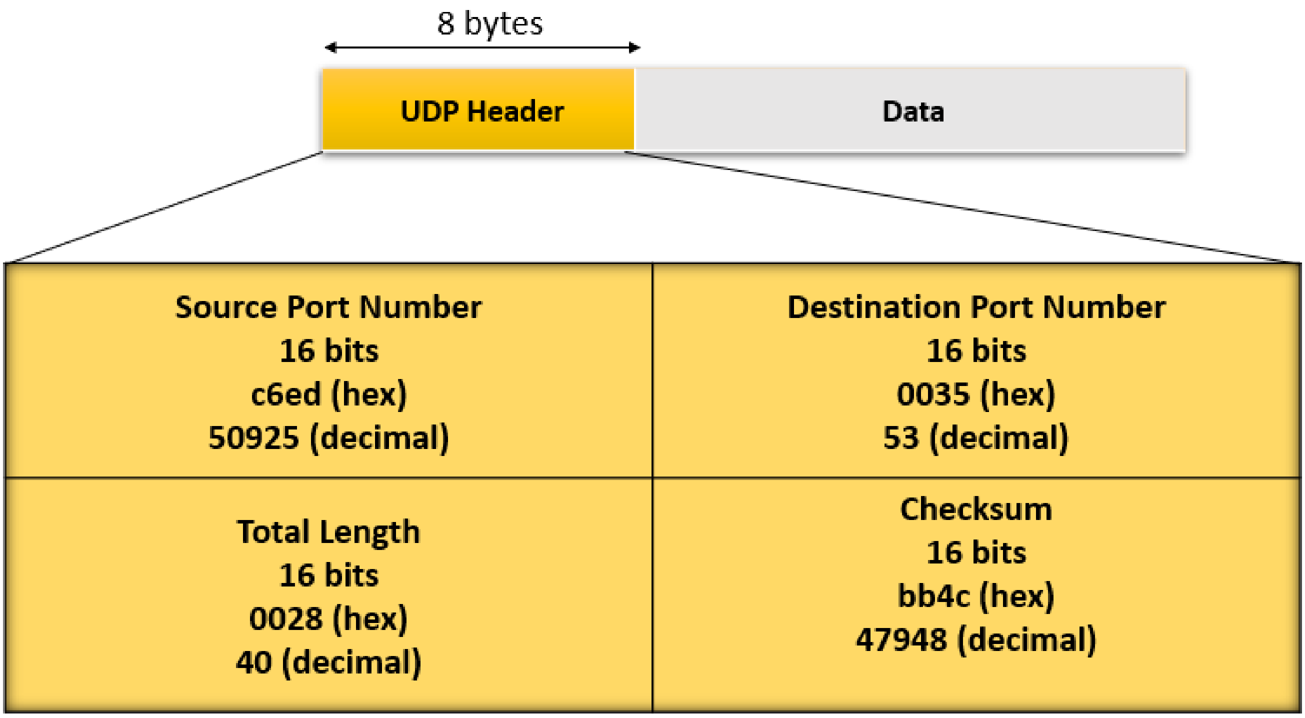
Image 3: This screenshot shows the DNS response packet received from the server in Wireshark. The details confirm the successful resolution, providing the IP address for www.google.com in the answer section.

UDP datagram header in hexadecimal format

The 8-byte hexadecimal dump header of the captured UDP packet is:

c6ed 0035 0028 bb4c.

The diagram below illustrates the 8-byte structure of the UDP header from the captured query packet. It breaks the header down into its four 16-bit fields, showing the specific values for the Source Port, Destination Port, Total Length, and Checksum in both hexadecimal and decimal formats.



Source Port Number **16 bits** **c6ed (hex)** **50925 (decimal)**	**Destination Port Number** **16 bits** **0035 (hex)** **53 (decimal)**
Total Length **16 bits** **0028 (hex)** **40 (decimal)**	**Checksum** **16 bits** **bb4c (hex)** **47948 (decimal)**

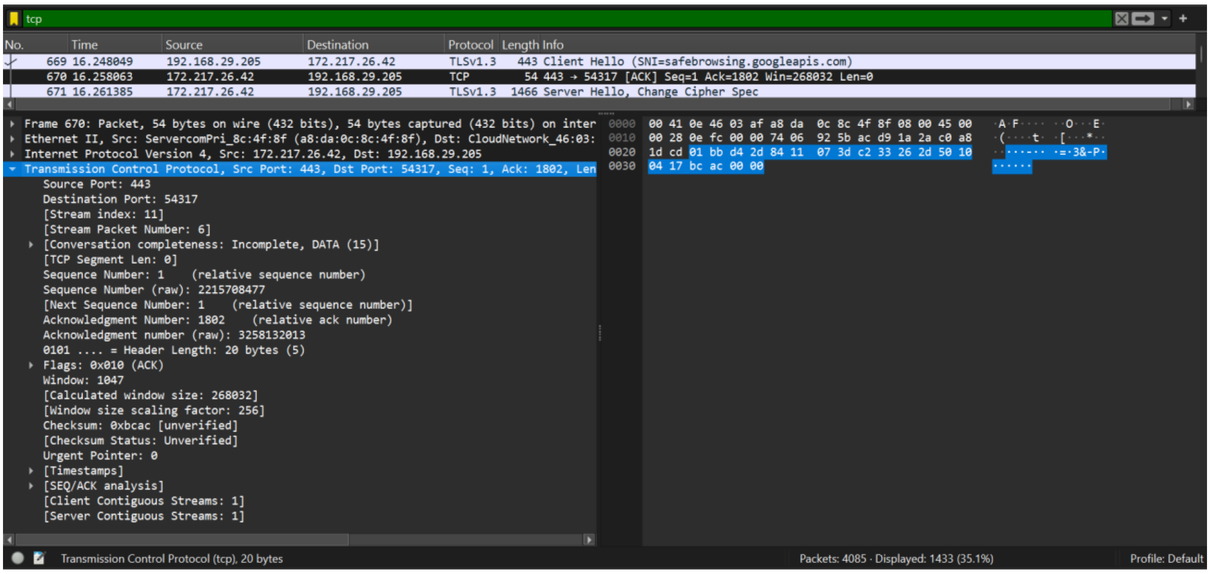
Table 1: UDP Packet Analysis

1.	a.	Source port number:	50925
	b.	Destination port number:	53
	c.	Total length of UDP datagram:	40 bytes
	d.	Length of data:	(40 - 8) = 32 bytes
	e.	Is the packet from client or server?	Client
	f.	Application-layer protocol:	DNS (Domain Name System)
	g.	Is checksum calculated?	Yes, and value is bb4c

2.	Are answers in No. 1 verified by the information in the detail pane?	Yes, all the values are confirmed in Wireshark's User Datagram Protocol details pane.
3.	Source and destination IP addresses in the query message:	Source: 192.168.29.205, Destination: 8.8.8.8
	Source and destination IP addresses in the response message:	Source: 8.8.8.8, Destination: 192.168.29.205
	Relation between IP addresses:	Addresses are swapped.
4.	Source and destination port number in the query message:	Source: 50925, Destination: 53
	Source and destination port number in the response message:	Source: 53, Destination: 50925
	Relation between port numbers:	The port numbers are swapped.
	Which port number is well-known?	Port 53
5.	Length of the first UDP packet:	40 bytes
	How many bytes of payload are carried by the first UDP packet?	32 bytes
6.	Number of bytes in the DNS message:	32 bytes
	Does the count agree with the answer to question 5?	Yes
7.	Is the checksum calculated for the first UDP packet? Value of the checksum:	Yes bb4c

Task 2: Generate TCP Packet using Web Browsing and Analyse It

This section details the analysis of a TCP segment captured during a secure web browsing session. TCP is a connection-oriented protocol that ensures reliable data delivery, making it suitable for applications like the World Wide Web.



```
No.   Time       Source          Destination     Protocol Length Info
  669 16.248049  192.168.29.205  172.217.26.42   TLSv1.3   443 Client Hello (SNI=safebrowsing.googleapis.com)
  670 16.258063  172.217.26.42   192.168.29.205  TCP        54 443 → 54317 [ACK] Seq=1 Ack=1802 Win=268032 Len=0
  671 16.261385  172.217.26.42   192.168.29.205  TLSv1.3  1466 Server Hello, Change Cipher Spec
```

```
Frame 670: Packet, 54 bytes on wire (432 bits), 54 bytes captured (432 bits) on inter
Ethernet II, Src: ServercomPri_8c:4f:8f (a8:da:0c:8c:4f:8f), Dst: CloudNetwork_46:03:
Internet Protocol Version 4, Src: 172.217.26.42, Dst: 192.168.29.205
Transmission Control Protocol, Src Port: 443, Dst Port: 54317, Seq: 1, Ack: 1802, Len
   Source Port: 443
   Destination Port: 54317
   [Stream index: 11]
   [Stream Packet Number: 6]
   [Conversation completeness: Incomplete, DATA (15)]
   [TCP Segment Len: 0]
   Sequence Number: 1    (relative sequence number)
   Sequence Number (raw): 2215708477
   [Next Sequence Number: 1    (relative sequence number)]
   Acknowledgment Number: 1802    (relative ack number)
   Acknowledgment number (raw): 3258132013
   0101 .... = Header Length: 20 bytes (5)
   Flags: 0x010 (ACK)
   Window: 1047
   [Calculated window size: 268032]
   [Window size scaling factor: 256]
   Checksum: 0xbcac [unverified]
   [Checksum Status: Unverified]
   Urgent Pointer: 0
   [Timestamps]
   [SEQ/ACK analysis]
   [Client Contiguous Streams: 1]
   [Server Contiguous Streams: 1]
```

```
0000  00 41 0e 46 03 af a8 da  0c 8c 4f 8f 08 00 45 00   ·A·F·····O··E·
0010  00 28 0e fc 00 00 74 06  92 5b ac d9 1a 2a c0 a8   ·(····t··[···*··
0020  1d cd 01 bb d4 2d 84 11  07 3d c2 33 26 2d 50 10   ·······=·3&-P·
0030  04 17 bc ac 00 00                                  ······
```

```
Transmission Control Protocol (tcp), 20 bytes          Packets: 4085 · Displayed: 1433 (35.1%)          Profile: Default
```

Image 4: This Wireshark screenshot shows a TCP packet captured, with the pane below detailing its TCP header fields like source/ destination ports, flags, and sequence numbers.

TCP segment header in hexadecimal format

The Hexadecimal dump of the captured TCP packet is:

01 bb d4 2d

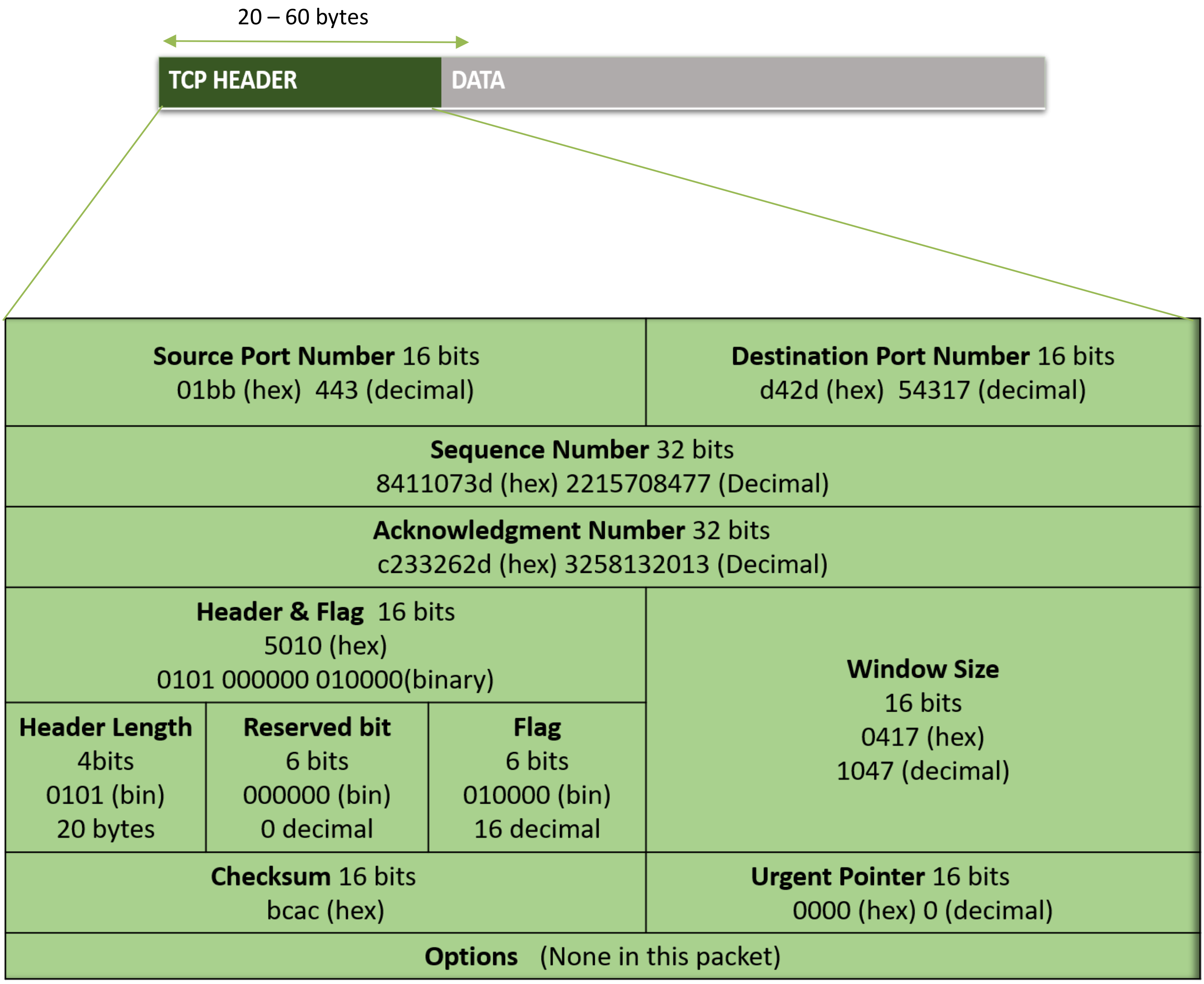
84 11 07 3d

c2 33 26 2d

50 10 04 17

bc ac 00 00

The diagram below provides a detailed, field-by-field analysis of a specific 20-byte TCP header. It's populated with data from a captured network packet, showing the values for the source port, destination port, sequence/acknowledgment numbers, and other fields.



Source Port Number 16 bits 01bb (hex) 443 (decimal)			**Destination Port Number** 16 bits d42d (hex) 54317 (decimal)
Sequence Number 32 bits 8411073d (hex) 2215708477 (Decimal)			
Acknowledgment Number 32 bits c233262d (hex) 3258132013 (Decimal)			
Header & Flag 16 bits 5010 (hex) 0101 000000 010000(binary)			**Window Size** 16 bits 0417 (hex) 1047 (decimal)
Header Length 4bits 0101 (bin) 20 bytes	**Reserved bit** 6 bits 000000 (bin) 0 decimal	**Flag** 6 bits 010000 (bin) 16 decimal	
Checksum 16 bits bcac (hex)			**Urgent Pointer** 16 bits 0000 (hex) 0 (decimal)
Options (None in this packet)			

Table 2: TCP Packet Analysis

1.	a. Source port number:	01bb (hex) 443 (Deci)
	b. Destination port number:	d42d (hex) 54317 (Deci)
	c. Sequence number:	8411073d (hex) 2215708477 (Deci)
	d. Acknowledgment number:	c233262d (hex) 3258132013 (Deci)
	e. Header length:	20 bytes
	f. Set flags:	0100 (16 Deci)
	g. Window size:	0417 (hex) 1047 (Deci)
	h. Urgent pointer:	0000 (hex) 0 (Deci)
2.	Are answers in question 1 verified by the information in the detail pane?	Yes

Conclusion

This lab provided hands-on experience with Wireshark, demonstrating the structural differences between the connectionless UDP protocol and the connection-oriented TCP protocol. By examining the headers of live packets, we identified key fields such as port numbers, sequence/acknowledgment numbers, and flags. This analysis confirms the theoretical roles of UDP in providing fast, low-overhead communication and of TCP in ensuring reliable, ordered data delivery.