CS 315 : Computer Networks Lab
Assignment - 6
UDP, Socket Programming & SMTP

Ayush Mallick
CS22BT008

Part-0

```
ayushm@ayushm-HP-Pavilion-x360-Convertible-14-cd0xxx:~$ ifconfig
lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 239  bytes 24118 (24.1 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 239  bytes 24118 (24.1 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

wlo1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.200.240.35  netmask 255.255.240.0  broadcast 10.200.255.255
        inet6 fe80::ffd0:a85f:6677:d0b  prefixlen 64  scopeid 0x20<link>
        ether 28:3a:4d:63:21:71  txqueuelen 1000  (Ethernet)
        RX packets 19199  bytes 24594968 (24.5 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 6925  bytes 790279 (790.2 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Part-1

1.

DNS (Domain Name System)

```
▸ Frame 66: 80 bytes on wire (640 bits), 80 bytes captured (640 bits) on interface wlo1, id 0
▸ Ethernet II, Src: CloudNet_63:21:71 (28:3a:4d:63:21:71), Dst: Cisco_0a:9a:f3 (44:b6:be:0a:9a:f3)
▸ Internet Protocol Version 4, Src: 10.200.240.35, Dst: 10.250.200.3
▸ User Datagram Protocol, Src Port: 59412, Dst Port: 53
▸ Domain Name System (query)
```

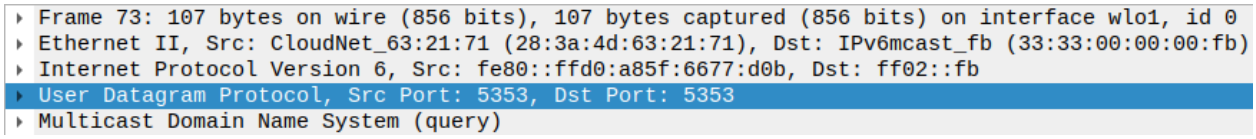
QUIC (Quick UDP Internet Connections)

```
▸ Frame 43: 1292 bytes on wire (10336 bits), 1292 bytes captured (10336 bits) on interface wlo1, id 0
▸ Ethernet II, Src: CloudNet_63:21:71 (28:3a:4d:63:21:71), Dst: Cisco_0a:9a:f3 (44:b6:be:0a:9a:f3)
▸ Internet Protocol Version 4, Src: 10.200.240.35, Dst: 142.250.195.234
▸ User Datagram Protocol, Src Port: 54334, Dst Port: 443
▸ QUIC IETF
```

UDP (User Datagram Protocol)

```
▸ Frame 70: 71 bytes on wire (568 bits), 71 bytes captured (568 bits) on interface wlo1, id 0
▸ Ethernet II, Src: CloudNet_63:21:71 (28:3a:4d:63:21:71), Dst: Cisco_0a:9a:f3 (44:b6:be:0a:9a:f3)
▸ Internet Protocol Version 4, Src: 10.200.240.35, Dst: 142.250.195.238
▸ User Datagram Protocol, Src Port: 35486, Dst Port: 443
▸ Data (29 bytes)
```

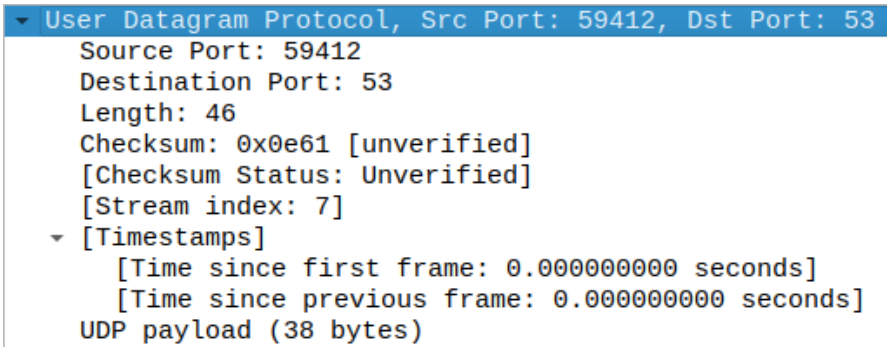
MDNS (Multicast Domain Name System)



2.

There are a total of *4* header fields and the payload in the UDP.

Source Port (*2* bytes)
Destination Port (*2* bytes)
Length (*2* bytes)
Checksum (*2* bytes)



3.

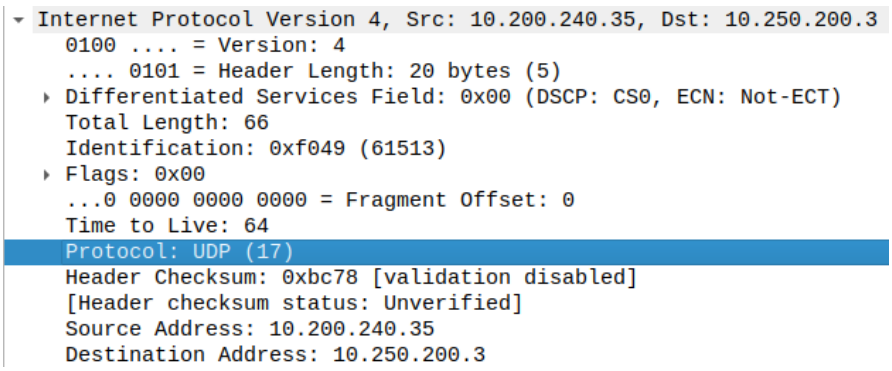
The Length field in the UDP header specifies the total length (in bytes) of the UDP packet, including both the UDP header, and the UDP payload.

4.

The maximum number of bytes that can be included in the UDP payload is the maximum size of an IP packet, minus the bytes for the UDP and IP headers, giving *(2^16 - 1) - 8 - 20 bytes = 65535 - 8 - 20 bytes = 65507 bytes* (for IPv4), or *(2^16 - 1) - 8 - 40 bytes = 65535 - 8 - 40 bytes = 65487 bytes* (for IPv6).

5.

Protocol number for UDP : *17*



Part-2

server.py :

cs315_iitdh - server.py

```python
import socket

# Server Configuration
HOST = '0.0.0.0'  # Listen on all available interfaces
PORT = 12345      # Port to bind to

# Create and bind the server socket
server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server_socket.bind((HOST, PORT))
server_socket.listen(1)
print(f"Server listening on {HOST}:{PORT}")

while True:
    conn, addr = server_socket.accept()
    print(f"Connection from {addr}")

    # Open a file to store received data
    with open("received.txt", "wb") as f:
        while True:
            data = conn.recv(1024)
            if not data:
                break
            f.write(data)

    print("File received successfully")

    # Read the received file and extract the first and last 10 lines
    with open("received.txt", "r", encoding="utf-8") as f:
        lines = f.readlines()
        first_10 = lines[:10]
        last_10 = lines[-10:]
        response = "".join(first_10 + last_10)

    # Send the extracted lines back to the client
    conn.sendall(response.encode())
    print("Sent first and last 10 lines back to the client")

    # Close connection
    conn.close()
    print("Connection closed\n")

    # Exit after one transaction (modify for continuous listening)
    break

server_socket.close()

```

client.py :

cs315_iitdh - client.py

```python
import socket

# Server Configuration
HOST = '127.0.0.1'  # Server IP (localhost for testing)
PORT = 12345        # Must match the server's port

# Create a client socket and connect to the server
client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
client_socket.connect((HOST, PORT))
print(f"Connected to server at {HOST}:{PORT}")

# Open and send the file in chunks
with open("alice.txt", "rb") as f:
    while chunk := f.read(1024):
        client_socket.sendall(chunk)

# Signal end of transmission
client_socket.shutdown(socket.SHUT_WR)

# Receive and display the response from the server
response = client_socket.recv(4096).decode()
print("Received from server:\n", response)

# Close the connection
client_socket.close()

```

terminal :

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\ayush\Documents\github\cs315_iitdh\assignments\assignment6> python server.py
Server listening on 0.0.0.0:12345
Connection from ('127.0.0.1', 52077)
File received successfully
Sent first and last 10 lines back to the client
Connection closed

PS C:\Users\ayush\Documents\github\cs315_iitdh\assignments\assignment6>
```

```
PS C:\Users\ayush\Documents\github\cs315_iitdh\assignments\assignment6> python client.py
Connected to server at 127.0.0.1:12345
Received from server:
                ALICE'S ADVENTURES IN WONDERLAND

                        Lewis Carroll

              THE MILLENNIUM FULCRUM EDITION 3.0



                           CHAPTER I
hers would, in the after-time, be herself a grown woman; and how
she would keep, through all her riper years, the simple and
loving heart of her childhood:  and how she would gather about
her other little children, and make THEIR eyes bright and eager
with many a strange tale, perhaps even with the dream of
Wonderland of long ago:  and how she would feel with all their
simple sorrows, and find a pleasure in all their simple joys,
remembering her own child-life, and the happy summer days.

                           THE END
PS C:\Users\ayush\Documents\github\cs315_iitdh\assignments\assignment6>
```

Part-3

mailclient.py :

cs315_iitdh - mailclient.py

```python
import socket
import ssl
from base64 import b64encode

# User credentials and email details
userEmail = "220010008@iitdh.ac.in"
userPassword = "[redacted]"  # Use the generated app password
userDestinationEmail = input("Enter Email Destination: ")
userSubject = input("Enter Subject: ")
userBody = input("Enter Message: ")

# msg = '{}.\r\n I love computer networks!'.format(userBody)
msg = f"Subject: {userSubject}\r\nFrom: {userEmail}\r\nTo: {userDestinationEmail}\r\n\r\n{userBody}\r\n I love computer networks!"

# Choose a mail server (e.g. Google mail server) and call it mailserver
mailserver = ("smtp.gmail.com", 587)

# Create socket called clientSocket and establish a TCP connection with mailserver
clientSocket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
clientSocket.connect(mailserver)

# Receive and check server response
recv = clientSocket.recv(1024).decode()
print(recv)
if recv[:3] != '220':
    print("220 reply not received from server.")

# Send HELO command and print server response.
heloCommand = "HELO Alice\r\n"
clientSocket.send(heloCommand.encode())
recv1 = clientSocket.recv(1024).decode()
print(recv1)
if recv1[:3] != '250':
    print("250 reply not received from server.")

#account authentication
clientSocket.send("STARTTLS\r\n".encode())
clientSocket.recv(1024)

# sslClientSocket = ssl.wrap_socket(clientSocket)   # deprecated in newer versions of python
context = ssl.create_default_context()
sslClientSocket = context.wrap_socket(clientSocket, server_hostname="smtp.gmail.com")

sslClientSocket.send("AUTH LOGIN\r\n".encode())
print(sslClientSocket.recv(1024))
sslClientSocket.send(b64encode(userEmail.encode()) + "\r\n".encode())
print(sslClientSocket.recv(1024))
sslClientSocket.send(b64encode(userPassword.encode()) + "\r\n".encode())
print(sslClientSocket.recv(1024))

# Send MAIL FROM command and print server response.
mailFromCommand = f"MAIL FROM:<{userEmail}>\r\n"
sslClientSocket.send(mailFromCommand.encode())
recv3 = sslClientSocket.recv(1024).decode()
print(recv3)

# Send RCPT TO command and print server response.
rcptToCommand = f"RCPT TO:<{userDestinationEmail}>\r\n"
sslClientSocket.send(rcptToCommand.encode())
recv4 = sslClientSocket.recv(1024).decode()
print(recv4)

# Send DATA command and print server response.
sslClientSocket.send("DATA\r\n".encode())
recv5 = sslClientSocket.recv(1024).decode()
print(recv5)

# Send message data.
sslClientSocket.send(msg.encode())

# Message ends with a single period.
sslClientSocket.send("\r\n.\r\n".encode())
recv6 = sslClientSocket.recv(1024).decode()
print(recv6)

# Send QUIT command and get server response.
sslClientSocket.send("QUIT\r\n".encode())
recv7 = sslClientSocket.recv(1024).decode()
print(recv7)

# Close connection
sslClientSocket.close()

```

terminal :

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\ayush\Documents\github\cs315_iitdh\assignments\assignment6> python mailclient.py
Enter Email Destination: ayushmallick2507@gmail.com
Enter Subject: Test Subject
Enter Message: Test Message
220 smtp.gmail.com ESMTP 98e67ed59e1d1-2fc13ac0a84sm6433840a91.16 - gsmtp

250 smtp.gmail.com at your service

b'334 VXNlcm5hbWU6\r\n'
b'334 UGFzc3dvcmQ6\r\n'
b'235 2.7.0 Accepted\r\n'
250 2.1.0 OK 98e67ed59e1d1-2fc13ac0a84sm6433840a91.16 - gsmtp

250 2.1.5 OK 98e67ed59e1d1-2fc13ac0a84sm6433840a91.16 - gsmtp

354 Go ahead 98e67ed59e1d1-2fc13ac0a84sm6433840a91.16 - gsmtp

250 2.0.0 OK  1739721654 98e67ed59e1d1-2fc13ac0a84sm6433840a91.16 - gsmtp

221 2.0.0 closing connection 98e67ed59e1d1-2fc13ac0a84sm6433840a91.16 - gsmtp
```

test mail :

Gmail

Ayush Mallick <ayushmallick2507@gmail.com>

Test Subject

1 message

220010008@iitdh.ac.in <220010008@iitdh.ac.in> 16 February 2025 at 21:30
To: ayushmallick2507@gmail.com

Test Message
I love computer networks!