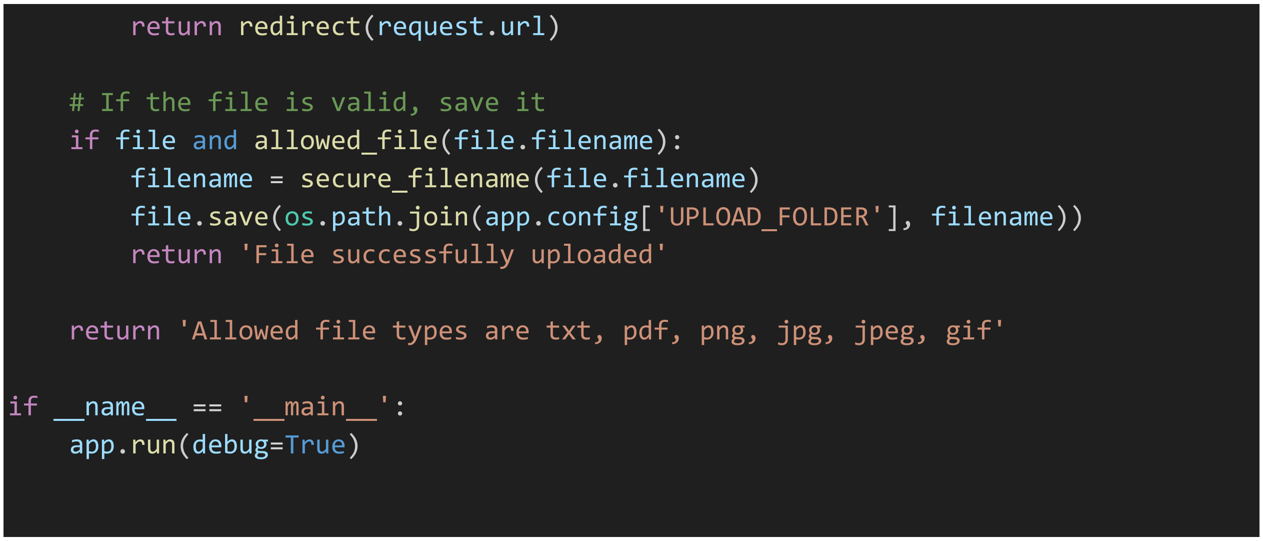
Assignment 8
WRC078BEI047

A. In the Programming language of you choice to write the web app that allows you to upload files.

Ans:

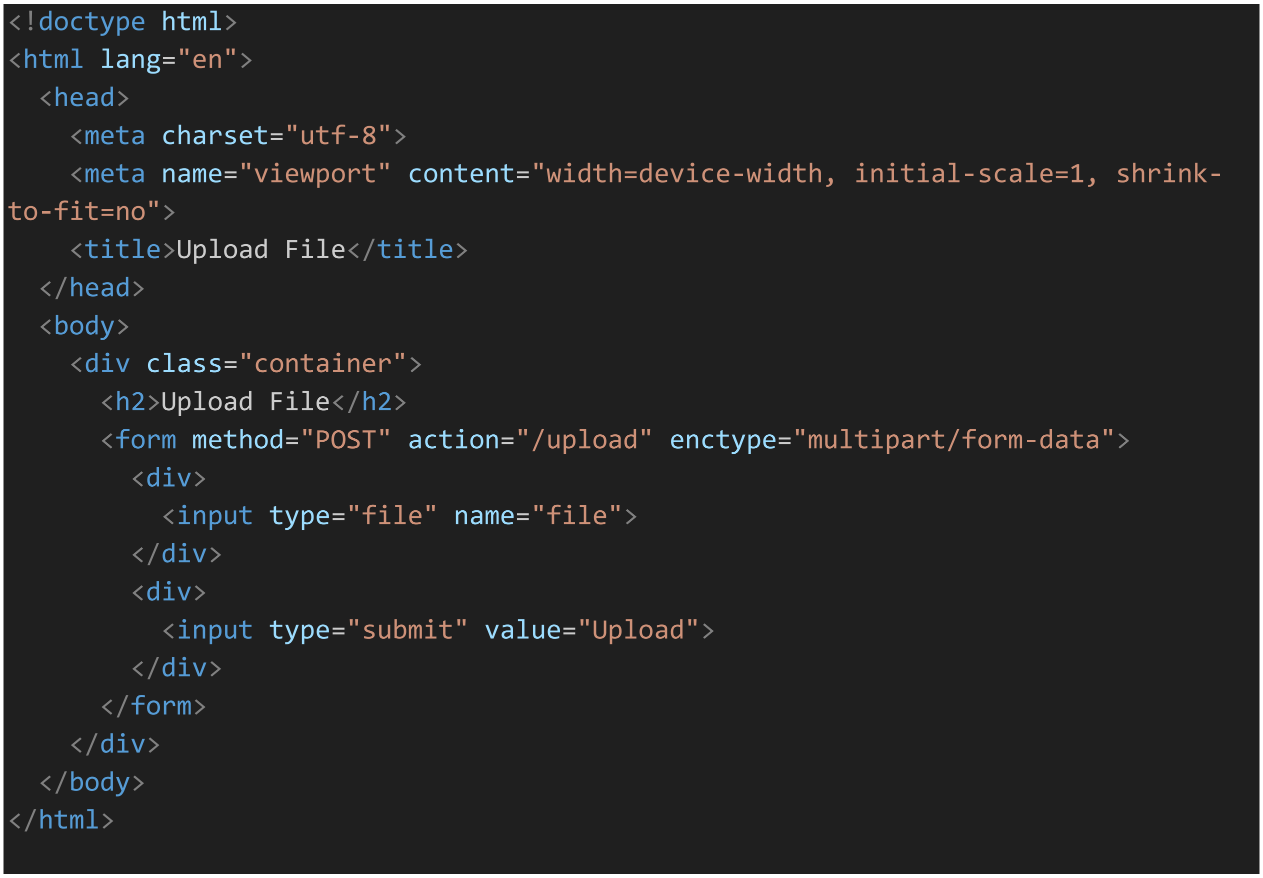
App.py

```
from flask import Flask, request, redirect, url_for, render_template
from werkzeug.utils import secure_filename
import os

app = Flask(__name__)

# Configurations
UPLOAD_FOLDER = 'uploads'
ALLOWED_EXTENSIONS = {'txt', 'pdf', 'png', 'jpg', 'jpeg', 'gif'}
app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER

if not os.path.exists(UPLOAD_FOLDER):
    os.makedirs(UPLOAD_FOLDER)

def allowed_file(filename):
    return '.' in filename and filename.rsplit('.', 1)[1].lower() in ALLOWED_EXTENSIONS

@app.route('/')
def upload_form():
    return render_template('upload.html')

@app.route('/upload', methods=['POST'])
def upload_file():
    # Check if the post request has the file part
    if 'file' not in request.files:
        return redirect(request.url)

    file = request.files['file']

    # If user does not select file, browser also submits an empty part without filename
    if file.filename == '':
```



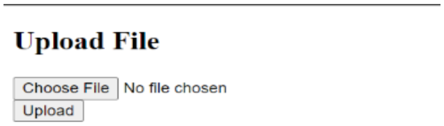
```python
        return redirect(request.url)

    # If the file is valid, save it
    if file and allowed_file(file.filename):
        filename = secure_filename(file.filename)
        file.save(os.path.join(app.config['UPLOAD_FOLDER'], filename))
        return 'File successfully uploaded'

    return 'Allowed file types are txt, pdf, png, jpg, jpeg, gif'

if __name__ == '__main__':
    app.run(debug=True)
```

upload.html



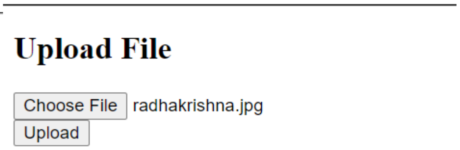
```html
<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
    <title>Upload File</title>
  </head>
  <body>
    <div class="container">
      <h2>Upload File</h2>
      <form method="POST" action="/upload" enctype="multipart/form-data">
        <div>
          <input type="file" name="file">
        </div>
        <div>
          <input type="submit" value="Upload">
        </div>
      </form>
    </div>
  </body>
</html>
```

Web App that allows us to upload files:



Upload File

No file chosen



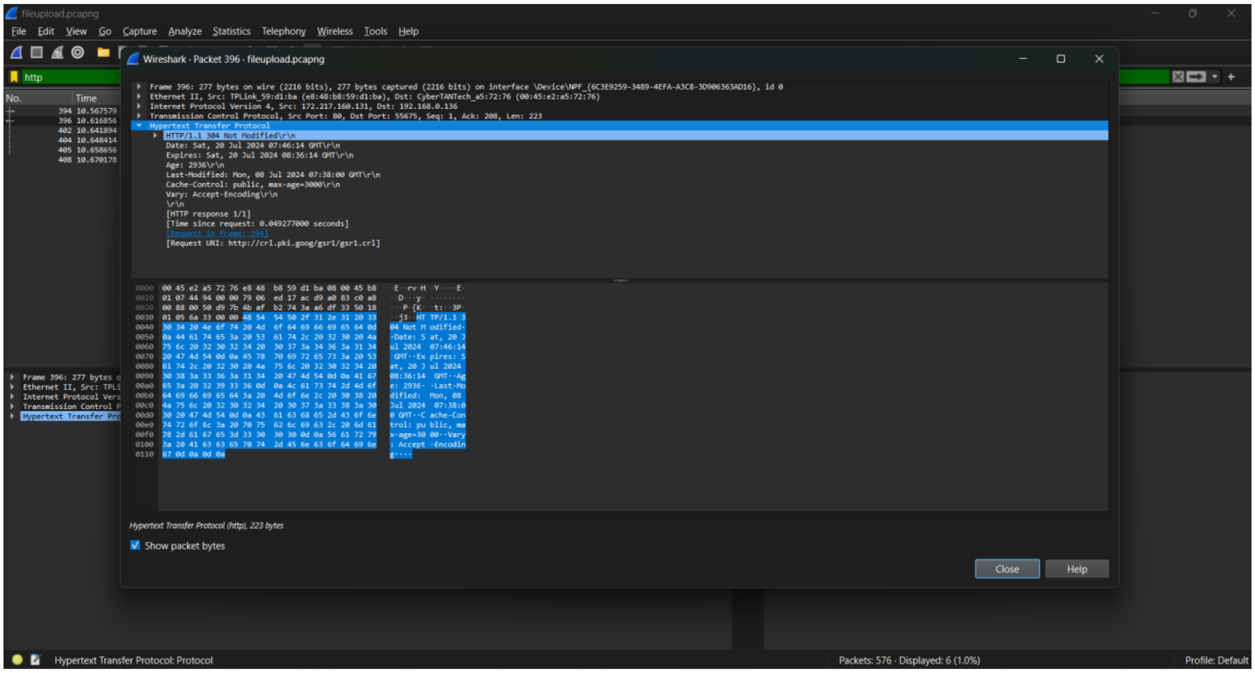
Upload File

radhakrishna.jpg

File successfully uploaded

B. Capture the traffic in wireshark while uploading the file.

Ans:



fileupload.pcapng
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Wireshark · Packet 396 · fileupload.pcapng
No. Time
Close
Help

http

```
394 10.567579
396 10.616856
402 10.641894
404 10.648414
405 10.658656
408 10.670178
```

```
Frame 396: 277 bytes on wire (2216 bits), 277 bytes captured (2216 bits) on interface \Device\NPF_{6C3E9259-3489-4EFA-A3C8-3D906363AD16}, id 0
Ethernet II, Src: TPLink_59:d1:ba (e8:48:b8:59:d1:ba), Dst: CyberTANTech_a5:72:76 (00:45:e2:a5:72:76)
Internet Protocol Version 4, Src: 172.217.160.131, Dst: 192.168.0.136
Transmission Control Protocol, Src Port: 80, Dst Port: 55675, Seq: 1, Ack: 208, Len: 223
Hypertext Transfer Protocol
    HTTP/1.1 304 Not Modified\r\n
    Date: Sat, 20 Jul 2024 07:46:14 GMT\r\n
    Expires: Sat, 20 Jul 2024 08:36:14 GMT\r\n
    Age: 2936\r\n
    Last-Modified: Mon, 08 Jul 2024 07:38:00 GMT\r\n
    Cache-Control: public, max-age=3000\r\n
    Vary: Accept-Encoding\r\n
    \r\n
    [HTTP response 1/1]
    [Time since request: 0.049277000 seconds]
    [Request in frame: 394]
    [Request URI: http://crl.pki.goog/gsr1/gsr1.crl]
```

```
0000  00 45 e2 a5 72 76 e8 48  b8 59 d1 ba 08 00 45 b8   ·E··rv·H ·Y····E·
0010  01 07 44 94 00 00 79 06  ed 17 ac d9 a0 83 c0 a8   ··D···y· ········
0020  00 88 00 50 d9 7b 4b af  b2 74 3a a6 df 33 50 18   ···P·{K· ·t:··3P·
0030  01 05 6a 33 00 00 48 54  54 50 2f 31 2e 31 20 33   ··j3··HT TP/1.1 3
0040  30 34 20 4e 6f 74 20 4d  6f 64 69 66 69 65 64 0d   04 Not M odified·
0050  0a 44 61 74 65 3a 20 53  61 74 2c 20 32 30 20 4a   ·Date: S at, 20 J
0060  75 6c 20 32 30 32 34 20  30 37 3a 34 36 3a 31 34   ul 2024  07:46:14
0070  20 47 4d 54 0d 0a 45 78  70 69 72 65 73 3a 20 53    GMT··Ex pires: S
0080  61 74 2c 20 32 30 20 4a  75 6c 20 32 30 32 34 20   at, 20 J ul 2024 
0090  30 38 3a 33 36 3a 31 34  20 47 4d 54 0d 0a 41 67   08:36:14  GMT··Ag
00a0  65 3a 20 32 39 33 36 0d  0a 4c 61 73 74 2d 4d 6f   e: 2936· ·Last-Mo
00b0  64 69 66 69 65 64 3a 20  4d 6f 6e 2c 20 30 38 20   dified:  Mon, 08 
00c0  4a 75 6c 20 32 30 32 34  20 30 37 3a 33 38 3a 30   Jul 2024  07:38:0
00d0  30 20 47 4d 54 0d 0a 43  61 63 68 65 2d 43 6f 6e   0 GMT··C ache-Con
00e0  74 72 6f 6c 3a 20 70 75  62 6c 69 63 2c 20 6d 61   trol: pu blic, ma
00f0  78 2d 61 67 65 3d 33 30  30 30 0d 0a 56 61 72 79   x-age=30 00··Vary
0100  3a 20 41 63 63 65 70 74  2d 45 6e 63 6f 64 69 6e   : Accept -Encodin
0110  67 0d 0a 0d 0a                                     g····
```

```
Frame 396: 277 bytes o
Ethernet II, Src: TPLi
Internet Protocol Vers
Transmission Control P
Hypertext Transfer Pro
```

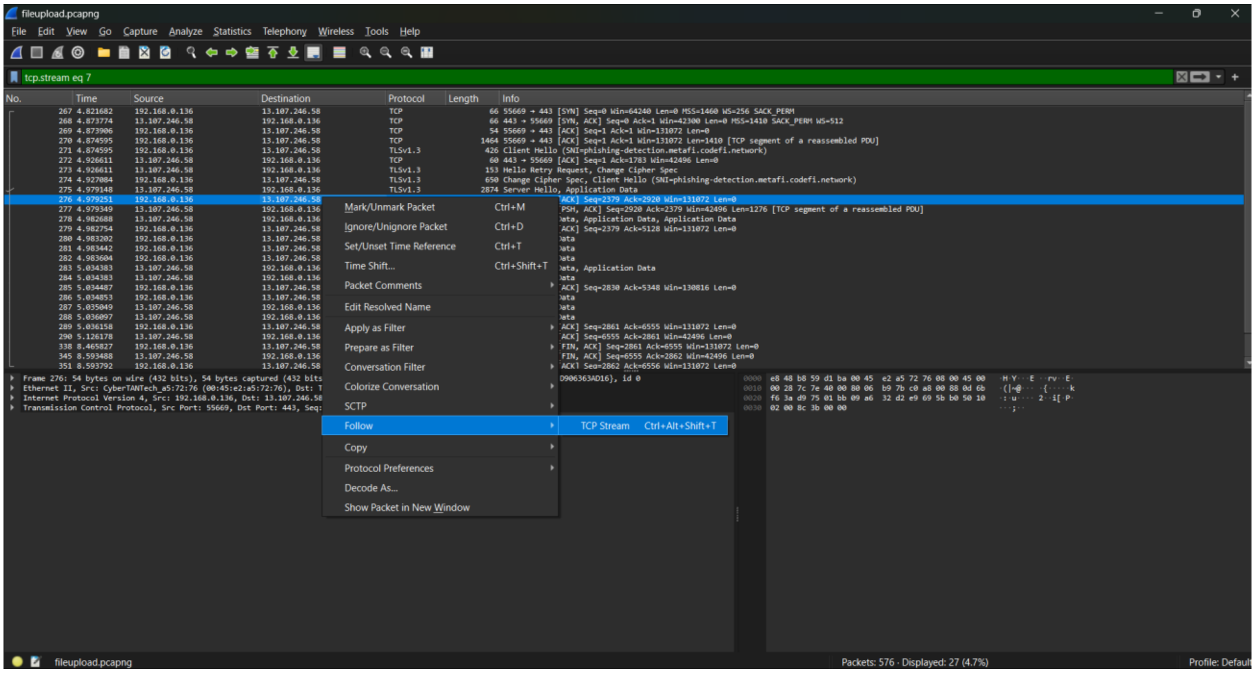
Hypertext Transfer Protocol (http), 223 bytes

☑ Show packet bytes

Hypertext Transfer Protocol: Protocol   Packets: 576 · Displayed: 6 (1.0%)   Profile: Default

3. Follow the TCP stream and explain connection initiation, connection maintenance and connection termination

Ans:


fileupload.pcapng
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Mark/Unmark Packet Ctrl+M
Ignore/Unignore Packet Ctrl+D
Set/Unset Time Reference Ctrl+T
Time Shift... Ctrl+Shift+T
Packet Comments
Edit Resolved Name
Apply as Filter
Prepare as Filter
Conversation Filter
Colorize Conversation
SCTP
Follow
TCP Stream Ctrl+Alt+Shift+T
Copy
Protocol Preferences
Decode As...
Show Packet in New Window
fileupload.pcapng
Profile: Default

```
tcp.stream eq 7
```

```
No.   Time      Source          Destination     Protocol  Length Info
  267 4.821682  192.168.0.136   13.107.246.58   TCP          66 55669 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
  268 4.873774  13.107.246.58   192.168.0.136   TCP          66 443 → 55669 [SYN, ACK] Seq=0 Ack=1 Win=42300 Len=0 MSS=1410 SACK_PERM WS=512
  269 4.873906  192.168.0.136   13.107.246.58   TCP          54 55669 → 443 [ACK] Seq=1 Ack=1 Win=131072 Len=0
  270 4.874595  192.168.0.136   13.107.246.58   TCP        1464 55669 → 443 [ACK] Seq=1 Ack=1 Win=131072 Len=1410 [TCP segment of a reassembled PDU]
  271 4.874595  192.168.0.136   13.107.246.58   TLSv1.3     426 Client Hello (SNI=phishing-detection.metafi.codefi.network)
  272 4.926611  13.107.246.58   192.168.0.136   TCP          60 443 → 55669 [ACK] Seq=1 Ack=1783 Win=42496 Len=0
  273 4.926611  13.107.246.58   192.168.0.136   TLSv1.3     153 Hello Retry Request, Change Cipher Spec
  274 4.927084  192.168.0.136   13.107.246.58   TLSv1.3     650 Change Cipher Spec, Client Hello (SNI=phishing-detection.metafi.codefi.network)
  275 4.979148  13.107.246.58   192.168.0.136   TLSv1.3    2874 Server Hello, Application Data
  276 4.979251  192.168.0.136   13.107.246.58   ACK] Seq=2379 Ack=2920 Win=131072 Len=0
  277 4.979349  13.107.246.58   192.168.0.136   PSH, ACK] Seq=2920 Ack=2379 Win=42496 Len=1276 [TCP segment of a reassembled PDU]
  278 4.982688  13.107.246.58   192.168.0.136   Data, Application Data, Application Data
  279 4.982754  192.168.0.136   13.107.246.58   ACK] Seq=2379 Ack=5128 Win=131072 Len=0
  280 4.983202  192.168.0.136   13.107.246.58   Data
  281 4.983442  192.168.0.136   13.107.246.58   Data
  282 4.983604  192.168.0.136   13.107.246.58   Data
  283 5.034383  13.107.246.58   192.168.0.136   Data, Application Data
  284 5.034383  13.107.246.58   192.168.0.136   Data
  285 5.034487  192.168.0.136   13.107.246.58   ACK] Seq=2830 Ack=5348 Win=130816 Len=0
  286 5.034853  192.168.0.136   13.107.246.58   Data
  287 5.035049  13.107.246.58   192.168.0.136   Data
  288 5.036097  13.107.246.58   192.168.0.136   Data
  289 5.036158  192.168.0.136   13.107.246.58   ACK] Seq=2861 Ack=6555 Win=131072 Len=0
  290 5.126178  13.107.246.58   192.168.0.136   ACK] Seq=6555 Ack=2861 Win=42496 Len=0
  338 8.465827  192.168.0.136   13.107.246.58   FIN, ACK] Seq=2861 Ack=6555 Win=131072 Len=0
  345 8.593488  13.107.246.58   192.168.0.136   FIN, ACK] Seq=6555 Ack=2862 Win=42496 Len=0
  351 8.593792  192.168.0.136   13.107.246.58   ACK] Seq=2862 Ack=6556 Win=131072 Len=0
```

```
Frame 276: 54 bytes on wire (432 bits), 54 bytes captured (432 bits  D906363AD16}, id 0
Ethernet II, Src: CyberTANTech_a5:72:76 (00:45:e2:a5:72:76), Dst: T
Internet Protocol Version 4, Src: 192.168.0.136, Dst: 13.107.246.58
Transmission Control Protocol, Src Port: 55669, Dst Port: 443, Seq:
```

```
0000  e8 48 b8 59 d1 ba 00 45  e2 a5 72 76 08 00 45 00   ·H·Y···E ··rv··E·
0010  00 28 7c 7e 40 00 80 06  b9 7b c0 a8 00 88 0d 6b   ·(|~@··· ·{·····k
0020  f6 3a d9 75 01 bb 09 a6  32 d2 e9 69 5b b0 50 10   ·:·u···· 2··i[·P·
0030  02 00 8c 3b 00 00                                  ···;··
```

Packets: 576 · Displayed: 27 (4.7%)

1. Connection Initiation

Goal: Identify the TCP handshake that establishes the connection between the client and server.

Steps:

1. **Find the SYN Packet**:
   - Look for a packet with the SYN flag set, usually the first packet sent by the client to start the connection.
   - **Example Packet Details**:
     - **Flags**: SYN
     - **Sequence Number**: X
2. **Find the SYN-ACK Packet**:
   - Find the packet with both SYN and ACK flags set, which is the server’s response to the client’s SYN request.
   - **Example Packet Details**:
     - **Flags**: SYN, ACK
     - **Sequence Number**: Y
     - **Acknowledgment Number**: X + 1 (acknowledges the client’s SYN)
3. **Find the ACK Packet**:
   - Look for the final packet in the handshake with the ACK flag set, sent by the client to acknowledge the server’s SYN-ACK.
   - **Example Packet Details**:
     - **Flags**: ACK
     - **Sequence Number**: X + 1

        - **Acknowledgment Number**: Y + 1 (acknowledges the server’s SYN)

2. Connection Maintenance

Goal: Analyze the data transfer and how TCP maintains the connection.

Steps:

1. **Identify Data Packets**:
    - Look at TCP packets with the PSH (Push) flag set, indicating data being pushed to the application layer.
    - **Example Packet Details**:
        - **Flags**: PSH, ACK
        - **Sequence Number**: Starting byte of the data segment.
        - **Acknowledgment Number**: Next expected byte from the other side.
2. **Check Flow Control**:
    - Observe the window size in the TCP header to see how much data the receiver can accept.
    - **Example Field**:
        - **Window Size**: Amount of buffer space available.
3. **Monitor Congestion Control**:
    - Look for signs of congestion control mechanisms like Slow Start or Congestion Avoidance, though detailed analysis may require deeper inspection of TCP behavior over time.
4. **Check Keep-Alive Packets (Optional)**:
    - If configured, check for occasional TCP keep-alive packets sent to ensure the connection remains active.
    - **Example Packet Details**:
        - **Flags**: ACK (with no payload)

3. Connection Termination

Goal: Identify the sequence of packets that close the TCP connection.

Steps:

1. **Find the FIN Packet**:
    - Look for a packet with the FIN flag set, indicating that one side wants to close the connection.
    - **Example Packet Details**:
        - **Flags**: FIN
        - **Sequence Number**: U
2. **Find the ACK Packet**:
    - Locate the packet that acknowledges the FIN request from the other side.
    - **Example Packet Details**:
        - **Flags**: ACK

        - **Acknowledgment Number**: U + 1 (acknowledges the FIN)

3. **Find the Second FIN Packet**:
    - The side that acknowledged the initial FIN now sends its own FIN packet to close its side of the connection.
    - **Example Packet Details**:
        - **Flags**: FIN
        - **Sequence Number**: W
4. **Find the Final ACK Packet**:
    - Look for the final ACK packet that acknowledges the second FIN request.
    - **Example Packet Details**:
        - **Flags**: ACK
        - **Acknowledgment Number**: W + 1

How to View This in Wireshark

1. **Start Capture**:
    - Begin capturing traffic on the appropriate network interface.
2. **Follow TCP Stream**:
    - Right-click on a TCP packet related to the connection you want to analyze.
    - Select "Follow" > "TCP Stream" to view the entire conversation.
3. **Examine Packets**:
    - Use filters and packet details to find SYN, SYN-ACK, and ACK packets for initiation.
    - Look for data packets, flow control, and keep-alive packets during maintenance.
    - Identify FIN and ACK packets for termination.