

Wireless Networking Fundamentals Project-3:

Real Time video broadcast supporting multiple clients using Socket Programming , OpenCV and PyShine.

Team:

Amandeep Singh (PES1UG19EC034)

Anvita Ramesh (PES1UG19EC051)

Amrita Das (PES1UG19EC037)

Aditi Adhikary (PES1UG19EC013)

Theory:

This project demonstrates a real time broadcasting system that allows connection to multiple clients from the server.

Algorithm:

Receiver side:

- Setting up of the TCP socket and binding it with the socket address
- Accept all the incoming connection requests and run the show_client as soon as the connection request is accepted. Threading library is used to run multiple tasks simultaneously.
- The show_client function receives the video as a byte stream and converts it back by decoding it.
- Pyshine function helps in creating the window for the video and OpenCV in displaying the frames or the images as a video.

Transmitter side:

- The transmitter uses the function VideoCapture() of the OpenCV library to capture the video from the inbuilt webcam of the laptop. We can also transmit a locally stored video.
- The socket is set up and bound with the address.
- The connection request is sent to the receiver.
- When the client leaves, an appropriate message saying the Client ,along with its IP address, has disconnected, while the other users are still transmitting the data.

Code:

Server:

```
import socket, cv2, pickle, struct
import imutils
import threading
import pyshine as ps
import cv2

server_socket = socket.socket(socket.AF_INET,socket.SOCK_STREAM)
host_name = socket.gethostname()
host_ip = "192.168.1.86"
print('HOST IP:',host_ip)
port = 9999
socket_address = (host_ip,port)
server_socket.bind(socket_address)
server_socket.listen()
print("Listening at",socket_address)
```

```

def show_client(addr,client_socket):
    try:
        print('CLIENT {} CONNECTED!'.format(addr))
        if client_socket:
            data = b""
            payload_size = struct.calcsize("Q")
            while True:
                while len(data) < payload_size:
                    packet = client_socket.recv(4*1024)
                    if not packet: break
                    data+=packet
                packed_msg_size = data[:payload_size]
                data = data[payload_size:]
                msg_size = struct.unpack("Q",packed_msg_size)[0]

                while len(data) < msg_size:
                    data += client_socket.recv(4*1024)
                frame_data = data[:msg_size]
                data = data[msg_size:]
                frame = pickle.loads(frame_data)
                text = f"CLIENT: {addr}"
                frame = ps.putText(frame,text,10,10,vspace=10,hspace=1,font_scale=0.7,
background_RGB=(255,0,0),text_RGB=(255,250,250))
                cv2.imshow(f"FROM {addr}",frame)
                key = cv2.waitKey(1) & 0xFF
                if key == ord('q'):
                    break
            client_socket.close()
        except Exception as e:
            print(f"CLIENT {addr} DISCONNECTED")
            pass

while True:
    client_socket,addr = server_socket.accept()
    thread = threading.Thread(target=show_client, args=(addr,client_socket))
    thread.start()
    print("TOTAL CLIENTS ",threading.activeCount() - 1)

```

Client:

```

import socket,cv2, pickle,struct
import pyshine as ps
import imutils
camera = True
if camera == True:
    vid = cv2.VideoCapture(0)
#else:
#    vid = cv2.VideoCapture('videos/mario.mp4')
client_socket = socket.socket(socket.AF_INET,socket.SOCK_STREAM)
host_ip = '192.168.1.11'

port = 9999
client_socket.connect((host_ip,port))

```

```

if client_socket:
    while (vid.isOpened()):
        try:
            img, frame = vid.read()
            frame = imutils.resize(frame,width=380)
            a = pickle.dumps(frame)
            message = struct.pack("Q",len(a))+a
            client_socket.sendall(message)
            cv2.imshow(f"T0: {host_ip}",frame)
            key = cv2.waitKey(1) & 0xFF
            if key == ord("q"):
                client_socket.close()
        except:
            print('VIDEO FINISHED!')
            break

```

Results:

1) Wireshark Captures:

| | | | | | | |
|-----|----------|----------------|----------------|-----|------|---|
| 57 | 5.192144 | 52.111.252.2 | 192.168.1.86 | TCP | 60 | 443 → 52478 [ACK] Seq=2738798263 Ack=1475277936 Win=2052 Len=0 |
| 59 | 5.336852 | 192.168.1.86 | 157.240.192.52 | TCP | 54 | 52610 → 443 [ACK] Seq=2701055873 Ack=2660300286 Win=511 Len=0 |
| 64 | 6.314278 | 157.240.192.52 | 192.168.1.86 | TCP | 60 | 443 → 52610 [ACK] Seq=2660300286 Ack=2701055954 Win=667 Len=0 |
| 75 | 7.781530 | 192.168.1.86 | 52.5.112.135 | TCP | 82 | [TCP Retransmission] 52669 → 443 [PSH, ACK] Seq=1666695116 Ack=4064281807 Win=512 Len=28 |
| 76 | 7.987095 | 52.5.112.135 | 192.168.1.86 | TCP | 66 | 443 → 52669 [ACK] Seq=4064281807 Ack=1666695144 Win=531 Len=0 SLE=1666695116 SRE=1666695144 |
| 78 | 8.437516 | 192.168.1.82 | 192.168.1.86 | TCP | 66 | 62430 → 9999 [SYN] Seq=2335240678 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1 |
| 79 | 8.437698 | 192.168.1.86 | 192.168.1.82 | TCP | 66 | 9999 → 62430 [SYN, ACK] Seq=72645978 Ack=2335240679 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM=1 |
| 80 | 8.438324 | 192.168.1.82 | 192.168.1.86 | TCP | 60 | 62430 → 9999 [ACK] Seq=2335240679 Ack=72645979 Win=1051136 Len=0 |
| 82 | 8.816826 | 192.168.1.82 | 192.168.1.86 | TCP | 1514 | 62430 → 9999 [ACK] Seq=2335242139 Ack=72645979 Win=1051136 Len=1460 |
| 83 | 8.816846 | 192.168.1.86 | 192.168.1.82 | TCP | 54 | 9999 → 62430 [ACK] Seq=72645979 Ack=2335243599 Win=1051136 Len=0 |
| 84 | 8.817014 | 192.168.1.82 | 192.168.1.86 | TCP | 1514 | 62430 → 9999 [ACK] Seq=2335243599 Ack=72645979 Win=1051136 Len=1460 |
| 86 | 8.817088 | 192.168.1.86 | 192.168.1.82 | TCP | 54 | 9999 → 62430 [ACK] Seq=72645979 Ack=2335246519 Win=1051136 Len=0 |
| 89 | 8.817364 | 192.168.1.86 | 192.168.1.82 | TCP | 54 | 9999 → 62430 [ACK] Seq=72645979 Ack=2335249439 Win=1051136 Len=0 |
| 90 | 8.817422 | 192.168.1.82 | 192.168.1.86 | TCP | 1514 | 62430 → 9999 [ACK] Seq=2335249439 Ack=72645979 Win=1051136 Len=1460 |
| 92 | 8.817602 | 192.168.1.86 | 192.168.1.82 | TCP | 54 | 9999 → 62430 [ACK] Seq=72645979 Ack=2335252359 Win=1051136 Len=0 |
| 94 | 8.817830 | 192.168.1.82 | 192.168.1.86 | TCP | 1514 | 62430 → 9999 [ACK] Seq=2335253819 Ack=72645979 Win=1051136 Len=1460 |
| 95 | 8.817866 | 192.168.1.86 | 192.168.1.82 | TCP | 54 | 9999 → 62430 [ACK] Seq=72645979 Ack=2335255279 Win=1051136 Len=0 |
| 98 | 8.818103 | 192.168.1.86 | 192.168.1.82 | TCP | 54 | 9999 → 62430 [ACK] Seq=72645979 Ack=2335258199 Win=1051136 Len=0 |
| 99 | 8.818174 | 192.168.1.82 | 192.168.1.86 | TCP | 1514 | 62430 → 9999 [ACK] Seq=2335258199 Ack=72645979 Win=1051136 Len=1460 |
| 100 | 8.818323 | 192.168.1.82 | 192.168.1.86 | TCP | 1514 | 62430 → 9999 [ACK] Seq=2335259659 Ack=72645979 Win=1051136 Len=1460 |
| 101 | 8.818351 | 192.168.1.86 | 192.168.1.82 | TCP | 54 | 9999 → 62430 [ACK] Seq=72645979 Ack=2335261119 Win=1051136 Len=0 |
| 104 | 8.818606 | 192.168.1.86 | 192.168.1.82 | TCP | 54 | 9999 → 62430 [ACK] Seq=72645979 Ack=2335264039 Win=1051136 Len=0 |
| 105 | 8.818654 | 192.168.1.82 | 192.168.1.86 | TCP | 1514 | 62430 → 9999 [ACK] Seq=2335264039 Ack=72645979 Win=1051136 Len=1460 |
| 107 | 8.818811 | 192.168.1.82 | 192.168.1.86 | TCP | 54 | 9999 → 62430 [ACK] Seq=72645979 Ack=2335266959 Win=1051136 Len=0 |
| 109 | 8.819044 | 192.168.1.82 | 192.168.1.86 | TCP | 1514 | 62430 → 9999 [ACK] Seq=2335268419 Ack=72645979 Win=1051136 Len=1460 |
| 110 | 8.819057 | 192.168.1.86 | 192.168.1.82 | TCP | 54 | 9999 → 62430 [ACK] Seq=72645979 Ack=2335269879 Win=1051136 Len=0 |
| 111 | 8.819167 | 192.168.1.82 | 192.168.1.86 | TCP | 1514 | 62430 → 9999 [ACK] Seq=2335269879 Ack=72645979 Win=1051136 Len=1460 |
| 113 | 8.819309 | 192.168.1.86 | 192.168.1.82 | TCP | 54 | 9999 → 62430 [ACK] Seq=72645979 Ack=2335272799 Win=1051136 Len=0 |
| 116 | 8.819548 | 192.168.1.86 | 192.168.1.82 | TCP | 54 | 9999 → 62430 [ACK] Seq=72645979 Ack=2335275719 Win=1051136 Len=0 |
| 119 | 8.819799 | 192.168.1.86 | 192.168.1.82 | TCP | 54 | 9999 → 62430 [ACK] Seq=72645979 Ack=2335278639 Win=1051136 Len=0 |
| 120 | 8.819905 | 192.168.1.82 | 192.168.1.86 | TCP | 1514 | 62430 → 9999 [ACK] Seq=2335278639 Ack=72645979 Win=1051136 Len=1460 |
| 121 | 8.820028 | 192.168.1.82 | 192.168.1.86 | TCP | 1514 | 62430 → 9999 [ACK] Seq=2335280099 Ack=72645979 Win=1051136 Len=1460 |
| 122 | 8.820045 | 192.168.1.86 | 192.168.1.82 | TCP | 54 | 9999 → 62430 [ACK] Seq=72645979 Ack=2335281559 Win=1051136 Len=0 |

Wireshark - Packet 214 - Ethernet

```

Internet Protocol Version 4, Src: 192.168.1.82, Dst: 192.168.1.86
  0100 .... = Version: 4
  .... 0101 = Header Length: 20 bytes (5)
  Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    0000 00.. = Differentiated Services Codepoint: Default (0)
    .... 0000 = Explicit Congestion Notification: Not ECN-Capable Transport (0)
  Total Length: 1500
  Identification: 0x14e7 (5351)
  Flags: 0x40, Don't fragment
    0... .... = Reserved bit: Not set
    .1.. .... = Don't fragment: Set
    ..0. .... = More fragments: Not set
  Fragment Offset: 0
  Time to Live: 128
  Protocol: TCP (6)
  Header Checksum: 0x5c3c [validation disabled]
  [Header checksum status: Unverified]
  Source Address: 192.168.1.82
  Destination Address: 192.168.1.86
Transmission Control Protocol, Src Port: 62430, Dst Port: 9999, Seq: 2335370619, Ack: 72645979, Len: 1460
  Source Port: 62430
  Destination Port: 9999
  [Stream index: 5]
  [TCP Segment Len: 1460]
  Sequence Number: 2335370619
  [Next Sequence Number: 2335372079]
  Acknowledgment Number: 72645979
  0101 .... = Header Length: 20 bytes (5)
  > Flags: 0x010 (ACK)
  Window: 4106
  [Calculated window size: 1051136]
  [Window size scaling factor: 256]
  Checksum: 0xdcc9 [unverified]
  [Checksum Status: Unverified]
  Urgent Pointer: 0
  > [SEQ/ACK analysis]
  > [Timestamps]
  TCP payload (1460 bytes)
Data (1460 bytes)
  Data: 5e6d595d6b595868595568585368584e6b594b6b5a4e6a595065585063564e64534f644f...
  [Length: 1460]

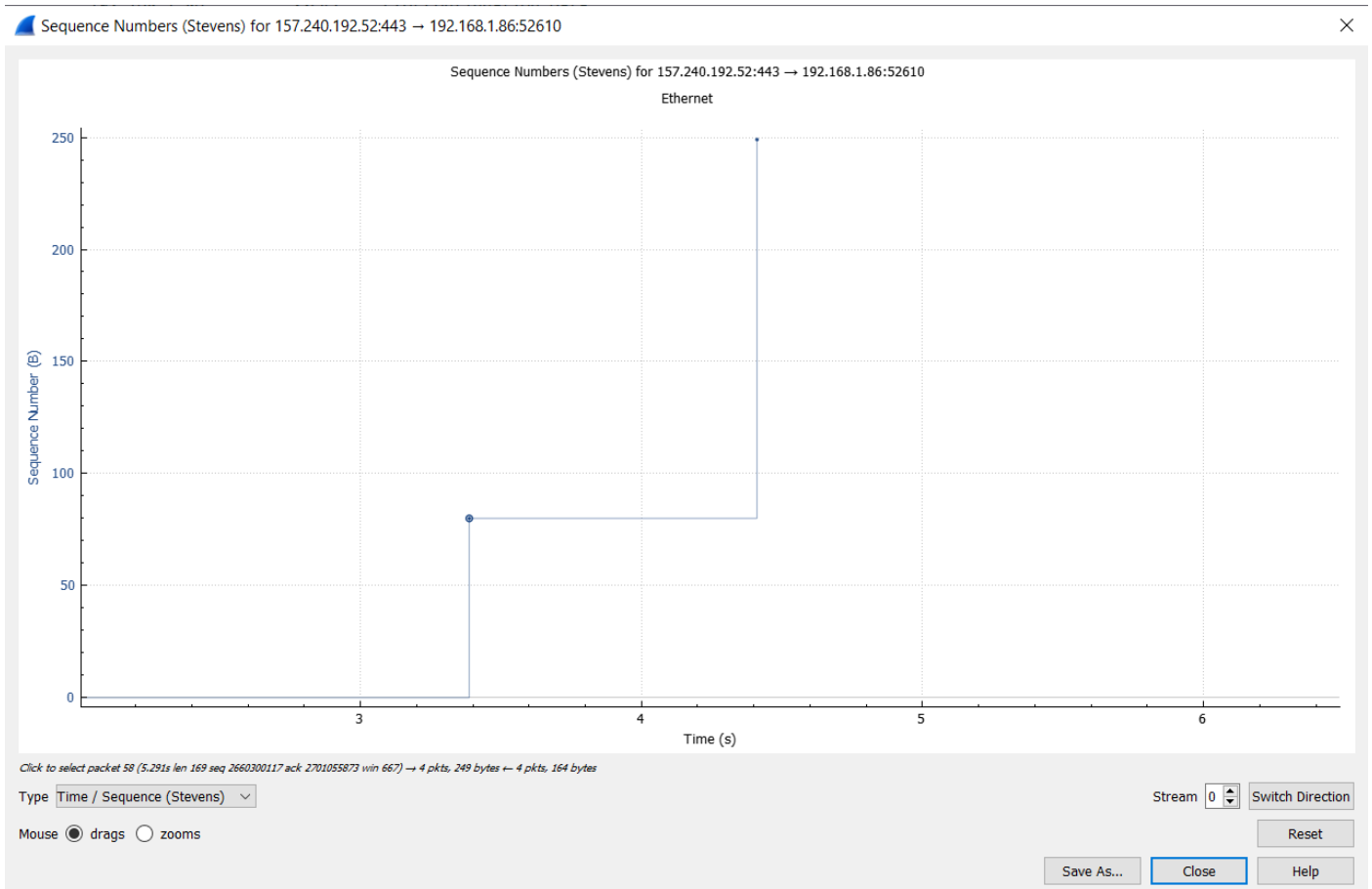
```

No.: 214 • Time: 8.627642 • Source: 192.168.1.82 • Destination: 192.168.1.86 • Protocol: TCP • Length: 1514 • Info: 62430 → 9999 [ACK] Seq=2335370619 Ack=72645979 Win=1051136 Len=1460

Close Help

2) Steven's Graphs:

Steven's graph shows the increase in sequence numbers with time. This helps us know when the delays occurred or when the transfer of packets hung.



3) Throughput:

Throughput for 157.240.192.52:443 → 192.168.1.86:52610 (MA)

