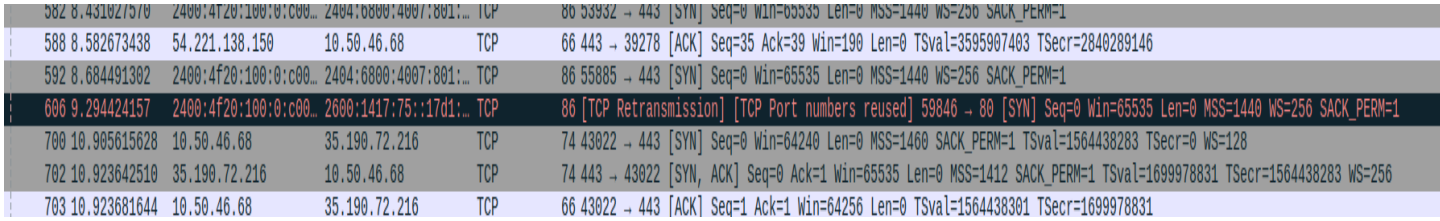
1)



582	8.431027570	2400:4f20:100:0:c00…	2404:6800:4007:801:…	TCP	86 53932 → 443 [SYN] Seq=0 Win=65535 Len=0 MSS=1440 WS=256 SACK_PERM=1
588	8.582673438	54.221.138.150	10.50.46.68	TCP	66 443 → 39278 [ACK] Seq=35 Ack=39 Win=190 Len=0 TSval=3595907403 TSecr=2840289146
592	8.684491302	2400:4f20:100:0:c00…	2404:6800:4007:801:…	TCP	86 55885 → 443 [SYN] Seq=0 Win=65535 Len=0 MSS=1440 WS=256 SACK_PERM=1
606	9.294424157	2400:4f20:100:0:c00…	2600:1417:75::17d1:…	TCP	86 [TCP Retransmission] [TCP Port numbers reused] 59846 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1440 WS=256 SACK_PERM=1
700	10.905615628	10.50.46.68	35.190.72.216	TCP	74 43022 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=1564438283 TSecr=0 WS=128
702	10.923642510	35.190.72.216	10.50.46.68	TCP	74 443 → 43022 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1412 SACK_PERM=1 TSval=1699978831 TSecr=1564438283 WS=256
703	10.923681644	10.50.46.68	35.190.72.216	TCP	66 43022 → 443 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=1564438301 TSecr=1699978831

The above screenshot shows the SYN, SYN ACK and ACk messages flowing between sockets of client and server. The black line with red text shows a retransmitted segment

1)

SERVER-

```
import socket
import threading

def handle_client(conn, addr):
    print(f"[NEW CONNECTION] {addr} connected.")
    try:
        while True:
            data = conn.recv(1024).decode()
            if not data:
                break

            # expected format: "operand1 operator operand2"
            try:
                op1, operator, op2 = data.split()
                op1, op2 = float(op1), float(op2)
                result = None

                if operator == '+':
                    result = op1 + op2
                elif operator == '-':
                    result = op1 - op2
                elif operator == '*':
                    result = op1 * op2
                elif operator == '/':
                    if op2 == 0:
                        result = "Error: Division by zero"
                    else:
                        result = op1 / op2
                else:
                    result = "Invalid operator"
```

```
            conn.send(str(result).encode())
        except Exception as e:
            conn.send(f"Error: {e}".encode())
    finally:
        conn.close()
        print(f"[DISCONNECTED] {addr} disconnected.")

def start_server(host="127.0.0.1", port=8080):
    server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    server.bind((host, port))
    server.listen(2)  # allow up to 2 clients
    print(f"[LISTENING] Server running on {host}:{port}")

    while True:
        conn, addr = server.accept()
        thread = threading.Thread(target=handle_client, args=(conn, addr))
        thread.start()

if __name__ == "__main__":
    start_server()
```

CLIENT-

```
 import socket

def run_client(host="127.0.0.1", port=8080):
    client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    client.connect((host, port))

    while True:
        expr = input("Enter operation (e.g. 10 + 20) or 'exit': ")
        if expr.lower() == "exit":
            break

        client.send(expr.encode())
        result = client.recv(1024).decode()
        print(f"Result: {result}")

    client.close()

if __name__ == "__main__":
    run_client()
```

```
utsav@utsav-victus :~/Desktop/NITK/5th sem/cn_lab/socket-programming $ python3 server.py
[LISTENING] Server running on 127.0.0.1:8080
[NEW CONNECTION] ('127.0.0.1', 60528) connected.
[DISCONNECTED] ('127.0.0.1', 60528) disconnected.
[NEW CONNECTION] ('127.0.0.1', 39214) connected.
```

```
utsav@utsav-victus :~/Desktop/NITK/5th sem/cn_lab/socket-programming $ python3 client.py
Enter operation (e.g. 10 + 20) or 'exit': 10 + 20
Result: 30.0
Enter operation (e.g. 10 + 20) or 'exit': 10 * 20
Result: 200.0
Enter operation (e.g. 10 + 20) or 'exit': 30 - 20
Result: 10.0
Enter operation (e.g. 10 + 20) or 'exit':
```

2)

SERVER-

```
import socket
import threading
import re

def validate_password(password):
    if not (8 <= len(password) <= 20):
        return "Invalid: Length must be between 8 and 20"

    if not re.search(r"[a-z]", password):
        return "Invalid: Must contain a lowercase letter"

    if not re.search(r"[A-Z]", password):
        return "Invalid: Must contain an uppercase letter"

    if not re.search(r"[0-9]", password):
        return "Invalid: Must contain a digit"

    if not re.search(r"[_@$]", password):
        return "Invalid: Must contain at least one special char (_,@,$)"

    return "Valid Password"

def handle_client(conn, addr):
    print(f"[NEW CONNECTION] {addr} connected.")
    try:
        while True:
            data = conn.recv(1024).decode()
            if not data:
                break
```

```
            response = validate_password(data)
            conn.send(response.encode())
    finally:
        conn.close()
        print(f"[DISCONNECTED] {addr} disconnected.")

def start_server(host="127.0.0.1", port=9090):
    server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    server.bind((host, port))
    server.listen(5)
    print(f"[LISTENING] Password server running on {host}:{port}")

    while True:
        conn, addr = server.accept()
        thread = threading.Thread(target=handle_client, args=(conn, addr))
        thread.start()

if __name__ == "__main__":
    start_server()
```

CLIENT-

```
import socket

def run_client(host="127.0.0.1", port=9090):
    client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    client.connect((host, port))

    while True:
        password = input("Enter password (or 'exit'): ")
        if password.lower() == "exit":
            break
        client.send(password.encode())
        response = client.recv(1024).decode()
        print(f"Server: {response}")

    client.close()

if __name__ == "__main__":
    run_client()
```

```
utsav@utsav-victus :~/Desktop/NITK/5th sem/cn_lab/socket-programming $ python3 server.py
[LISTENING] Password server running on 127.0.0.1:9090
[NEW CONNECTION] ('127.0.0.1', 43664) connected.
```

```
utsav@utsav-victus :~/Desktop/NITK/5th sem/cn_lab/socket-programming $ python3 client.py
Enter password (or 'exit'): 123456
Server: Invalid: Length must be between 8 and 20
Enter password (or 'exit'): 123456789
Server: Invalid: Must contain a lowercase letter
Enter password (or 'exit'): abcd12345
Server: Invalid: Must contain an uppercase letter
Enter password (or 'exit'): ABcd12345
Server: Invalid: Must contain at least one special char (_,@,$)
Enter password (or 'exit'): ABcd@1234
Server: Valid Password
Enter password (or 'exit'):
```