Testing and Analysis

1. TCP Connection Setup and Handshake Verification

The TCP three-way handshake process is visible between the client and server IP addresses (e.g., `192.168.1.248` and `127.0.0.1`). The handshake sequence (SYN, SYN-ACK, ACK) confirms a reliable connection setup, as each client and server successfully establish sessions on the designated port (e.g., `5000`).

2. Message Relay and Broadcasting

We can see TCP packets transferring data, indicating message exchange between clients and the server. Client `Josh` (IP: `192.168.1.248`) sends messages to the server, which are relayed back to connected clients like Tony (IP: `127.0.0.1`). Confirming that the server is successfully broadcasting messages to all active clients. TCP packets labeled `[PSH, ACK]` typically signify data transmissions, confirming that the messages are relayed as expected.

3. Client Connection and Disconnection

The logs display when clients connect and disconnect. In the screenshots, it shows clients joining and leaving, along with messages in the server logs, which match the TCP reset packets (e.g., RST). The RST packets indicate disconnections. This signifies the end of a session. The server logs correctly reflect the changes in client connections, which align with the RST packets in Wireshark.

4. Private Messaging Verification

Although the exact packet contents are not visible in the screenshots, it would be expected that private messages would have unique identifiers in the data payload to indicate their targeted recipient. If the chat application includes private messaging, you could look for specific data strings or payloads addressed to individual IPs without a broadcast indicator. Unfortunately, without seeing this in the capture, the functionality cannot be fully verified.

5. Errors or Reset Packets

Some RST (Reset) packets are visible in the capture.

- These packets may be due to clients disconnecting or application errors. It's essential to monitor these occurrences, as frequent resets could indicate an issue with session management or network stability.

6. Data Integrity and Complete Message Delivery

TCP packets show ACK for each segment, which confirms delivery. No visible packet fragmentation or FIN-based interruptions for incomplete transmissions. Message delivery is stable and complete without issues, as TCP inherently ensures the complete and ordered delivery of data segments.

7. Filtering for Relevant Data

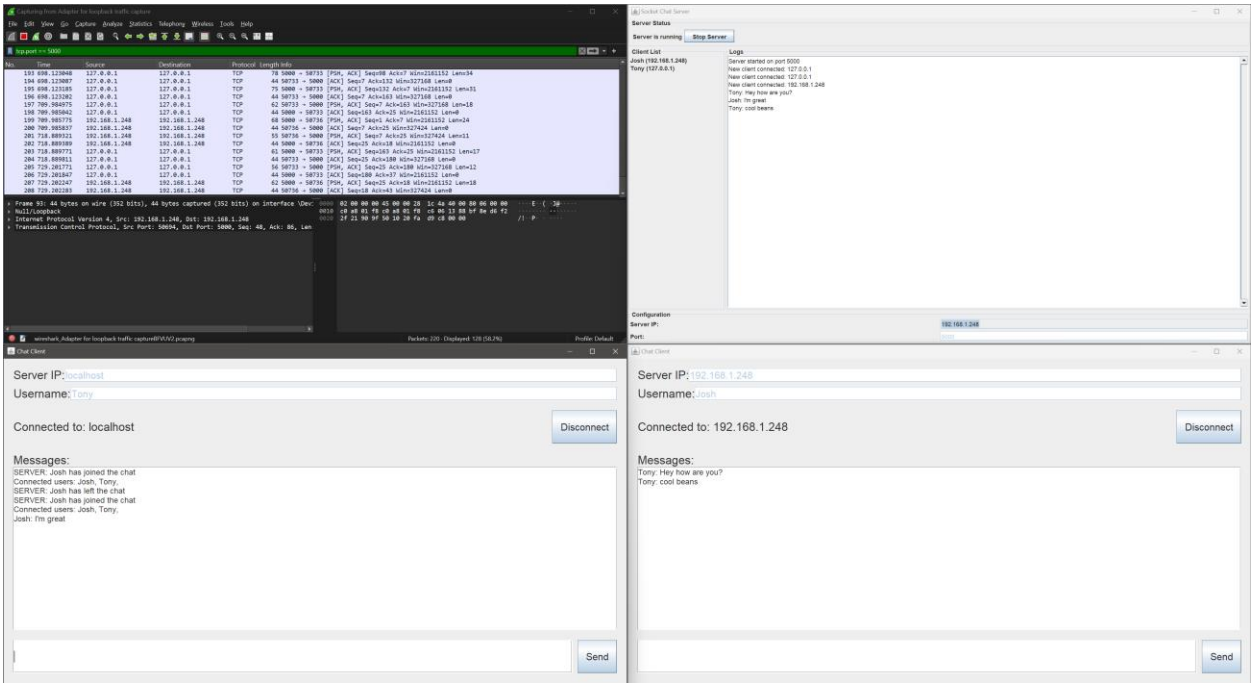
Filters such as `tcp.port == 5000` have been applied in Wireshark to narrow down relevant traffic. This filter is effective for isolating traffic on port 5000, which your application uses. Additionally, filtering by IP addresses (e.g., `192.168.1.248` for the server) can help analyze client-server exchanges and confirm targeted communication.

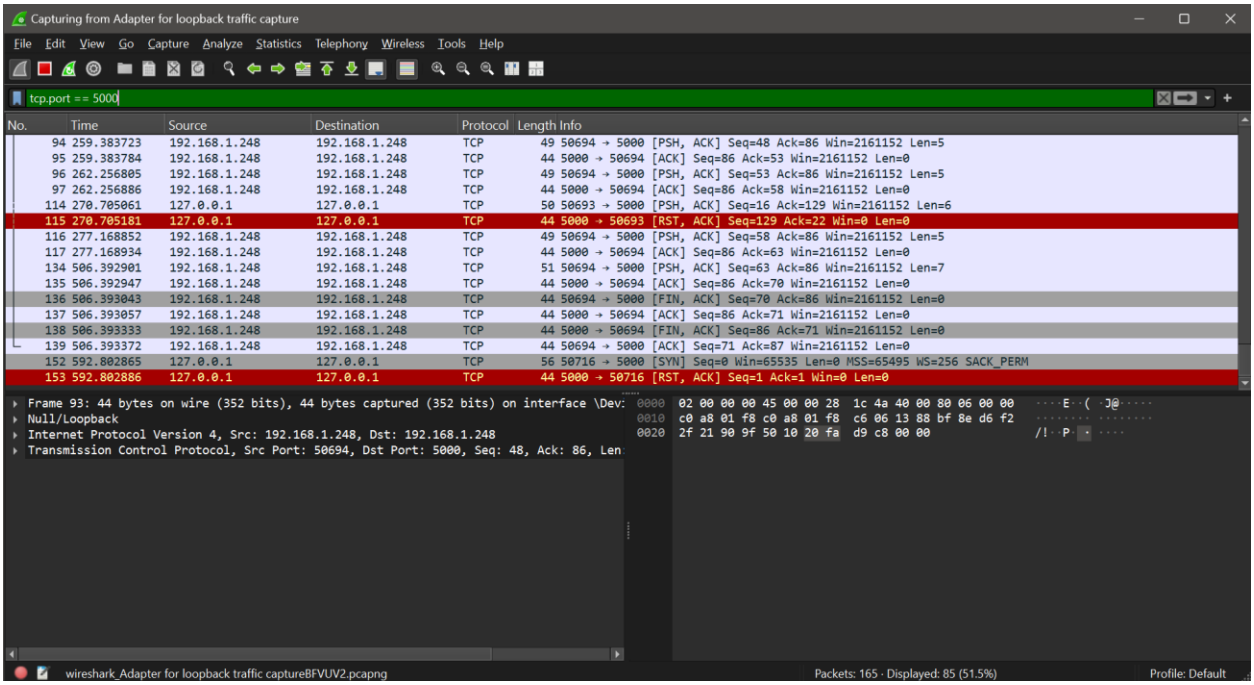
Conclusion

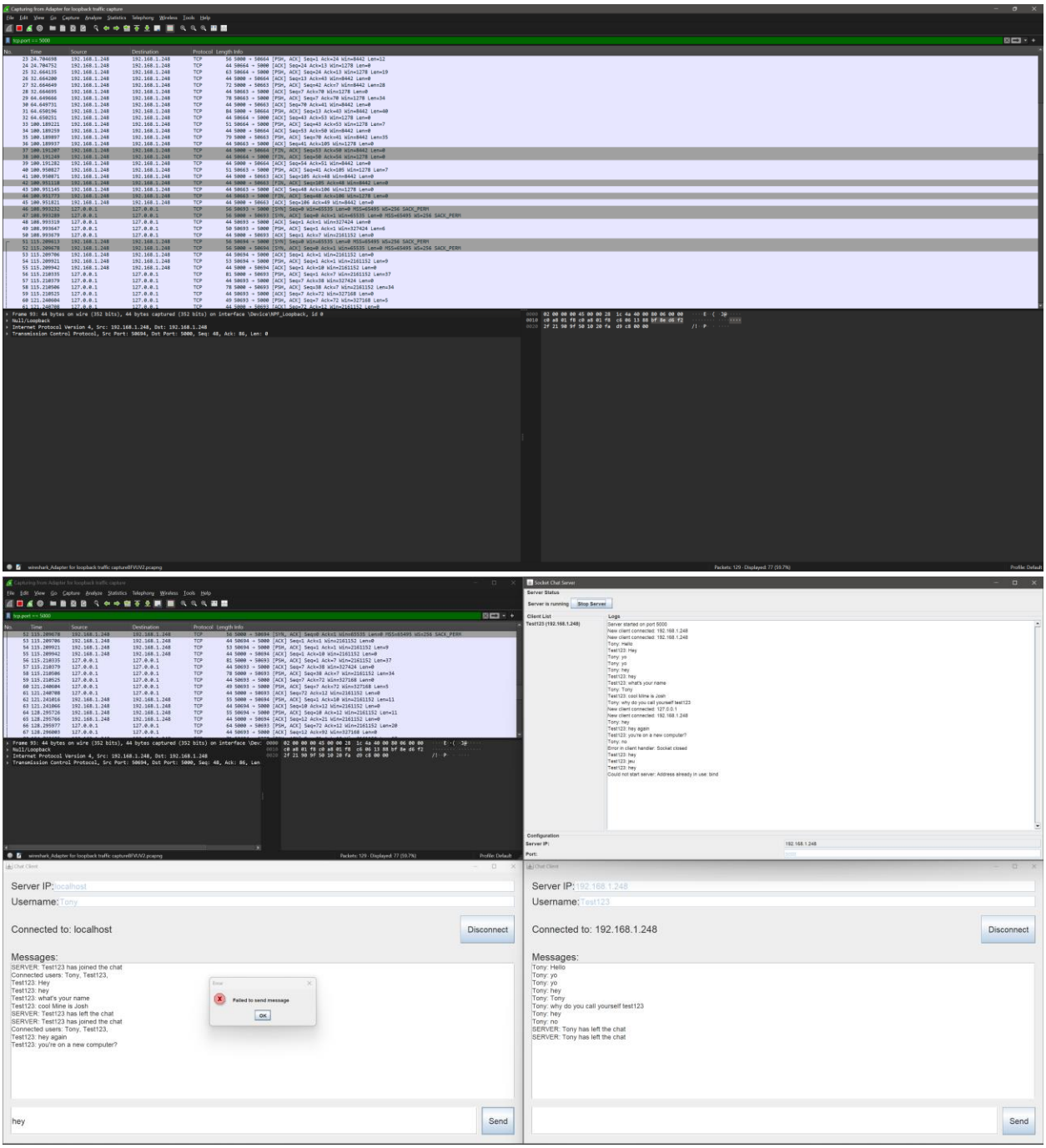
The captured data demonstrates a functioning chat server where TCP handshakes, data exchanges, connection/disconnection logs, and message relay appear to operate correctly. For further analysis:

- **Check** private messaging by inspecting payloads or unique identifiers in messages.
- **Monitor** RST packets closely to understand if they correlate with intentional disconnections or potential issues.
- **Ensure** secure handling of credentials, if applicable, to prevent exposure over the network.

Photos


Server IP: localhost
Username: Tony
Connected to: localhost
Disconnect
Messages:
SERVER: Josh has joined the chat
Connected users: Josh, Tony,
SERVER: Josh has left the chat
SERVER: Josh has joined the chat
Connected users: Josh, Tony,
Josh: I'm great
Send
Server IP: 192.168.1.248
Username: Josh
Connected to: 192.168.1.248
Disconnect
Messages:
Tony: Hey how are you?
Tony: cool beans
Send


Capturing from Adapter for loopback traffic capture
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
tcp.port == 5000
No. Time Source Destination Protocol Length Info
94 259.383723 192.168.1.248 192.168.1.248 TCP 49 50694 → 5000 [PSH, ACK] Seq=48 Ack=86 Win=2161152 Len=5
95 259.383784 192.168.1.248 192.168.1.248 TCP 44 5000 → 50694 [ACK] Seq=86 Ack=53 Win=2161152 Len=0
96 262.256805 192.168.1.248 192.168.1.248 TCP 49 50694 → 5000 [PSH, ACK] Seq=53 Ack=86 Win=2161152 Len=5
97 262.256886 192.168.1.248 192.168.1.248 TCP 44 5000 → 50694 [ACK] Seq=86 Ack=58 Win=2161152 Len=0
114 270.705061 127.0.0.1 127.0.0.1 TCP 50 50693 → 5000 [PSH, ACK] Seq=16 Ack=129 Win=2161152 Len=6
115 270.705181 127.0.0.1 127.0.0.1 TCP 44 5000 → 50693 [RST, ACK] Seq=129 Ack=22 Win=0 Len=0
116 277.168852 192.168.1.248 192.168.1.248 TCP 49 50694 → 5000 [PSH, ACK] Seq=58 Ack=86 Win=2161152 Len=5
117 277.168934 192.168.1.248 192.168.1.248 TCP 44 5000 → 50694 [ACK] Seq=86 Ack=63 Win=2161152 Len=0
134 506.392901 192.168.1.248 192.168.1.248 TCP 51 50694 → 5000 [PSH, ACK] Seq=63 Ack=86 Win=2161152 Len=7
135 506.392947 192.168.1.248 192.168.1.248 TCP 44 5000 → 50694 [ACK] Seq=86 Ack=70 Win=2161152 Len=0
136 506.393043 192.168.1.248 192.168.1.248 TCP 44 50694 → 5000 [FIN, ACK] Seq=70 Ack=86 Win=2161152 Len=0
137 506.393057 192.168.1.248 192.168.1.248 TCP 44 5000 → 50694 [ACK] Seq=86 Ack=71 Win=2161152 Len=0
138 506.393333 192.168.1.248 192.168.1.248 TCP 44 5000 → 50694 [FIN, ACK] Seq=86 Ack=71 Win=2161152 Len=0
139 506.393372 192.168.1.248 192.168.1.248 TCP 44 50694 → 5000 [ACK] Seq=71 Ack=87 Win=2161152 Len=0
152 592.802865 127.0.0.1 127.0.0.1 TCP 56 50716 → 5000 [SYN] Seq=0 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM
153 592.802886 127.0.0.1 127.0.0.1 TCP 44 5000 → 50716 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
Frame 93: 44 bytes on wire (352 bits), 44 bytes captured (352 bits) on interface \Dev
Null/Loopback
Internet Protocol Version 4, Src: 192.168.1.248, Dst: 192.168.1.248
Transmission Control Protocol, Src Port: 50694, Dst Port: 5000, Seq: 48, Ack: 86, Len
0000 02 00 00 00 45 00 00 28 1c 4a 40 00 80 06 00 00
0010 c0 a8 01 f8 c0 a8 01 f8 c6 06 13 88 bf 8e d6 f2
0020 2f 21 90 9f 50 10 20 fa d9 c8 00 00
wireshark_Adapter for loopback traffic captureBFVUV2.pcapng
Packets: 165 · Displayed: 85 (51.5%)
Profile: Default


Server IP: localhost
Username: Tony
Connected to: localhost
Disconnect
Messages:
SERVER: Test123 has joined the chat
Connected users: Tony, Test123,
Test123: Hey
Test123: hey
Test123: what's your name
Test123: cool Mine is Josh
SERVER: Test123 has left the chat
SERVER: Test123 has joined the chat
Connected users: Tony, Test123,
Test123: hey again
Test123: you're on a new computer?
Failed to send message
OK
hey
Send
Server IP: 192.168.1.248
Username: Test123
Connected to: 192.168.1.248
Disconnect
Messages:
Tony: Hello
Tony: yo
Tony: yo
Tony: hey
Tony: Tony
Tony: why do you call yourself test123
Tony: hey
Tony: no
SERVER: Tony has left the chat
SERVER: Tony has left the chat
Send