

Sockets

Introduction.

This report covers the implementation of a simple chat consisting of a server and several clients. ChatClient.java: Has two threads, one to listen for incoming messages from the server and one to send messages to the server. ChatServer.java: Has one thread to each of the clients currently connected but also one thread to listen for new incoming connections from new clients.

Client.

ChatClient.java consist of two classes, the ChatClient class itself and ChatSend. In the main method we create a new socket to localhost port 1234 as well as taking the username as an argument for later use. The newly created socket is passed as an argument to a method called receive. In this method, we create a buffered reader with an input stream reader reading from the sockets input stream of data. this reader is continuously monitored for new incoming messages which are printed to standard output. In case of failing the exception is caught and is being printed as a stack trace.

Back in the main method a new thread is spawned to handle the sending of new messages. This thread is told to execute the ChatSend class. This class takes the same socket as receive as an argument but also the username that was previously saved. In order to for this class to run in a separate thread the class implements the Runnable interface. The class only has one method run() where a new print stream is created on the sockets output stream. In order to read input from the user a buffered reader is created on system.in.

In order to display usernames in the chat the method starts with sending the users username to the server. The method then enters a continuous loop where every new line inserted by the user is sent to the server. In case of failing the exception is caught and is being printed as a stack trace.

The client is now executing two threads, one that is responsible for listening for new messages and the other thread is responsible for sending new messages.

Server.

ChatServer.java consists of two classes, the ChatServer class itself and a ClientThread class. In the main method we create a serversocket on port 1234. An array list named connections is also instantiated in order to save all current connections. A simple message is printed to std out once the server is started displaying on which IP adress and port the server can be found. The main method is then entering a continuous while loop that is checking for incoming connections from new clients. Every incoming connection on port 1234 is accepted. A new object and thread is spawned for each new client connection in order to be able to handle a large number of simultaneous chatting clients. The new socket connection is passed to the newly created object for later use. The new connection is also appended to an unsorted array list called connections.

To keep the list of all connections updated across all clients the ClientThread class has a public method updateList() that is called before spawning the new thread in main. This method takes the new array list as an argument and replaces the old one with the new list.

Each newly created thread starts out in the run() method in the clientThread Class. Here, a buffered reader is created reading from the new clients sockets / connections input stream.

The next thing that is done is that the username that was sent from the client is received and saved in a string for later use. Additionally, the thread calls a method called newConnection with the newly received username. This method is responsible for broadcasting to all other currently connected clients that the new user has joined the chat. This is done by utilizing the array list of connections passed from the main method in the ChatServer class. The method is simply iterating the list and sending an informative message to all clients in the list.

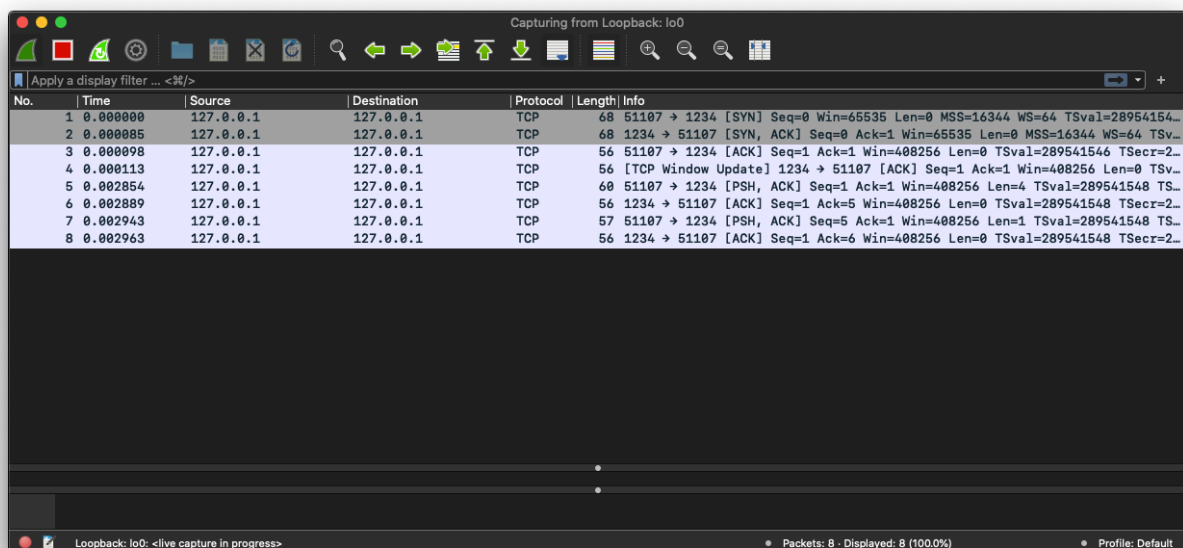
After this, back in the run method it enters a continuous while loop checking for any incoming message from the client, when a new message is received it is being passed to the broadcast method. This method is implemented in a similar manner as newConnection, i.e. broadcasting the message to all clients in the list. The message received from the client is sent to each connected client (except the client that sent the message) with the username prepended in order for all clients to distinguish from which client the message was received. In case of failing in any of the methods the exception is caught and is being printed as a stack trace.

The server is now executing one thread listening for incoming connections as well as one thread per connected client. This makes it possible for the server to handle a large number of simultaneous connected clients.

Extra assignment.

In order to capture network traffic from the chat we use the network sniffer Wireshark. Sniffing is done on the loopback Io0 interface since we are connected to localhost on 127.0.0.1.

Connecting one client to server



Capturing from Loopback: lo0

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	127.0.0.1	127.0.0.1	TCP	68	51107 → 1234 [SYN] Seq=0 Win=65535 Len=0 MSS=16344 WS=64 TSval=28954154...
2	0.000085	127.0.0.1	127.0.0.1	TCP	68	1234 → 51107 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=16344 WS=64 TSv...
3	0.000098	127.0.0.1	127.0.0.1	TCP	56	51107 → 1234 [ACK] Seq=1 Ack=1 Win=408256 Len=0 TSval=289541546 TSecr=2...
4	0.000113	127.0.0.1	127.0.0.1	TCP	56	[TCP Window Update] 1234 → 51107 [ACK] Seq=1 Ack=1 Win=408256 Len=0 TSv...
5	0.002854	127.0.0.1	127.0.0.1	TCP	60	51107 → 1234 [PSH, ACK] Seq=1 Ack=1 Win=408256 Len=4 TSval=289541548 TS...
6	0.002889	127.0.0.1	127.0.0.1	TCP	56	1234 → 51107 [ACK] Seq=1 Ack=5 Win=408256 Len=0 TSval=289541548 TSecr=2...
7	0.002943	127.0.0.1	127.0.0.1	TCP	57	51107 → 1234 [PSH, ACK] Seq=5 Ack=1 Win=408256 Len=1 TSval=289541548 TS...
8	0.002963	127.0.0.1	127.0.0.1	TCP	56	1234 → 51107 [ACK] Seq=1 Ack=6 Win=408256 Len=0 TSval=289541548 TSecr=2...

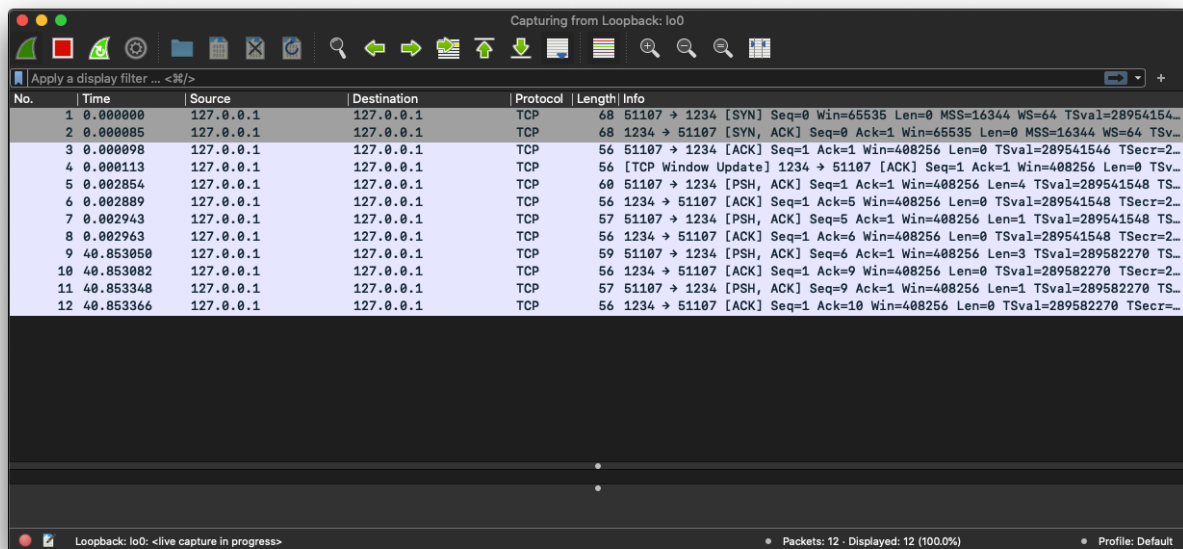
Loopback: lo0: <live capture in progress> Packets: 8 - Displayed: 8 (100.0%) Profile: Default

Here we can see that the first three rows is the standard TCP three-way handshake. After that the window size is updated. Row 5-8 is showing the network traffic of the client sending its name to the server and corresponding acknowledgements. In this case its sending "Emil" which is len=4. A note

on the PSH flag, PSH in the TCP header informs the receiving host that the data should be pushed up to the receiving application immediately.

Sending one message from client to server

Here we can see that sending one message to the server results in four rows in our packet sniffer. Row 9 is the message itself while row 10 is the ack of the message.



Capturing from Loopback: lo0

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	127.0.0.1	127.0.0.1	TCP	68	51107 → 1234 [SYN] Seq=0 Win=65535 Len=0 MSS=16344 WS=64 TSval=28954154...
2	0.000085	127.0.0.1	127.0.0.1	TCP	68	1234 → 51107 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=16344 WS=64 TSv...
3	0.000098	127.0.0.1	127.0.0.1	TCP	56	51107 → 1234 [ACK] Seq=1 Ack=1 Win=408256 Len=0 TSval=289541546 TSecr=2...
4	0.000113	127.0.0.1	127.0.0.1	TCP	56	[TCP Window Update] 1234 → 51107 [ACK] Seq=1 Ack=1 Win=408256 Len=0 TSv...
5	0.002854	127.0.0.1	127.0.0.1	TCP	60	51107 → 1234 [PSH, ACK] Seq=1 Ack=1 Win=408256 Len=4 TSval=289541548 TS...
6	0.002889	127.0.0.1	127.0.0.1	TCP	56	1234 → 51107 [ACK] Seq=1 Ack=5 Win=408256 Len=0 TSval=289541548 TSecr=2...
7	0.002943	127.0.0.1	127.0.0.1	TCP	57	51107 → 1234 [PSH, ACK] Seq=5 Ack=1 Win=408256 Len=1 TSval=289541548 TS...
8	0.002963	127.0.0.1	127.0.0.1	TCP	56	1234 → 51107 [ACK] Seq=1 Ack=6 Win=408256 Len=0 TSval=289541548 TSecr=2...
9	40.853050	127.0.0.1	127.0.0.1	TCP	59	51107 → 1234 [PSH, ACK] Seq=6 Ack=1 Win=408256 Len=3 TSval=289582270 TS...
10	40.853082	127.0.0.1	127.0.0.1	TCP	56	1234 → 51107 [ACK] Seq=1 Ack=9 Win=408256 Len=0 TSval=289582270 TSecr=2...
11	40.853348	127.0.0.1	127.0.0.1	TCP	57	51107 → 1234 [PSH, ACK] Seq=9 Ack=1 Win=408256 Len=1 TSval=289582270 TS...
12	40.853366	127.0.0.1	127.0.0.1	TCP	56	1234 → 51107 [ACK] Seq=1 Ack=10 Win=408256 Len=0 TSval=289582270 TSecr=...

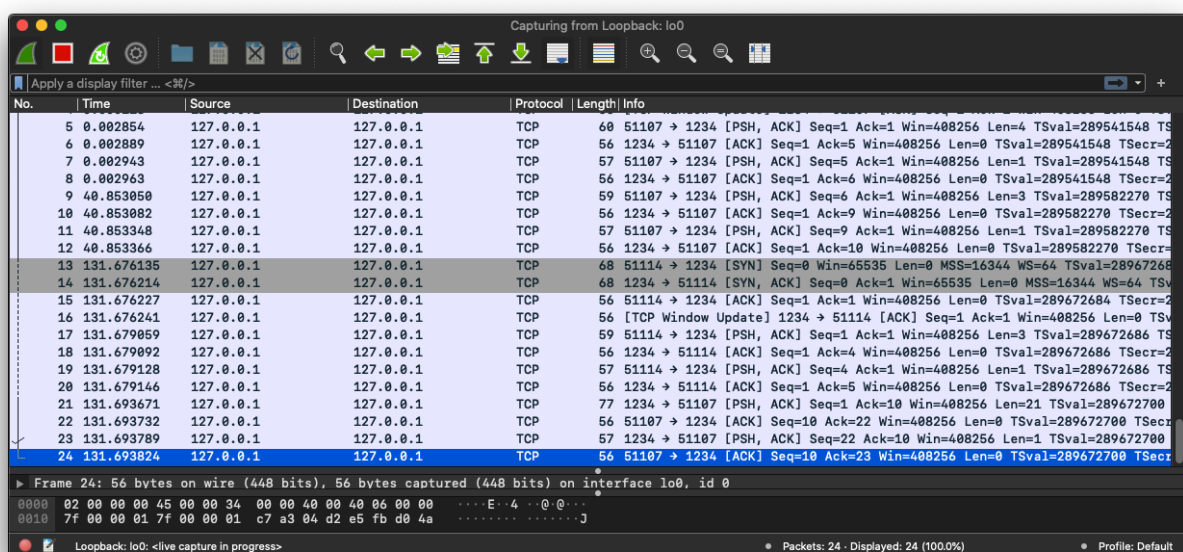
Loopback: lo0: <live capture in progress>

Packets: 12 · Displayed: 12 (100.0%)

Profile: Default

Connecting another client to server. Total of 2 clients.

By connecting another client we see the same TCP three-way handshake at row 13-15. After that the client is sending its username and the program is broadcasting that the new user has joined the chat. It is logical that this results in 8 rows since it previously was 4 rows with one client. The informative message can be found by expanding each row with a PSH flag.



Capturing from Loopback: lo0

No.	Time	Source	Destination	Protocol	Length	Info
5	0.002854	127.0.0.1	127.0.0.1	TCP	60	51107 → 1234 [PSH, ACK] Seq=1 Ack=1 Win=408256 Len=4 TSval=289541548 TS...
6	0.002889	127.0.0.1	127.0.0.1	TCP	56	1234 → 51107 [ACK] Seq=1 Ack=5 Win=408256 Len=0 TSval=289541548 TSecr=2...
7	0.002943	127.0.0.1	127.0.0.1	TCP	57	51107 → 1234 [PSH, ACK] Seq=5 Ack=1 Win=408256 Len=1 TSval=289541548 TS...
8	0.002963	127.0.0.1	127.0.0.1	TCP	56	1234 → 51107 [ACK] Seq=1 Ack=6 Win=408256 Len=0 TSval=289541548 TSecr=2...
9	40.853050	127.0.0.1	127.0.0.1	TCP	59	51107 → 1234 [PSH, ACK] Seq=6 Ack=1 Win=408256 Len=3 TSval=289582270 TS...
10	40.853082	127.0.0.1	127.0.0.1	TCP	56	1234 → 51107 [ACK] Seq=1 Ack=9 Win=408256 Len=0 TSval=289582270 TSecr=2...
11	40.853348	127.0.0.1	127.0.0.1	TCP	57	51107 → 1234 [PSH, ACK] Seq=9 Ack=1 Win=408256 Len=1 TSval=289582270 TS...
12	40.853366	127.0.0.1	127.0.0.1	TCP	56	1234 → 51107 [ACK] Seq=1 Ack=10 Win=408256 Len=0 TSval=289582270 TSecr=...
13	131.676135	127.0.0.1	127.0.0.1	TCP	68	51114 → 1234 [SYN] Seq=0 Win=65535 Len=0 MSS=16344 WS=64 TSval=28967268...
14	131.676214	127.0.0.1	127.0.0.1	TCP	68	1234 → 51114 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=16344 WS=64 TSv...
15	131.676227	127.0.0.1	127.0.0.1	TCP	56	51114 → 1234 [ACK] Seq=1 Ack=1 Win=408256 Len=0 TSval=289672684 TSecr=2...
16	131.676241	127.0.0.1	127.0.0.1	TCP	56	[TCP Window Update] 1234 → 51114 [ACK] Seq=1 Ack=1 Win=408256 Len=0 TSv...
17	131.679059	127.0.0.1	127.0.0.1	TCP	59	51114 → 1234 [PSH, ACK] Seq=1 Ack=1 Win=408256 Len=3 TSval=289672686 TS...
18	131.679092	127.0.0.1	127.0.0.1	TCP	56	1234 → 51114 [ACK] Seq=1 Ack=4 Win=408256 Len=0 TSval=289672686 TSecr=2...
19	131.679128	127.0.0.1	127.0.0.1	TCP	57	51114 → 1234 [PSH, ACK] Seq=4 Ack=1 Win=408256 Len=1 TSval=289672686 TS...
20	131.679146	127.0.0.1	127.0.0.1	TCP	56	1234 → 51114 [ACK] Seq=1 Ack=5 Win=408256 Len=0 TSval=289672686 TSecr=2...
21	131.693671	127.0.0.1	127.0.0.1	TCP	77	1234 → 51107 [PSH, ACK] Seq=1 Ack=10 Win=408256 Len=21 TSval=289672700 TS...
22	131.693732	127.0.0.1	127.0.0.1	TCP	56	51107 → 1234 [ACK] Seq=10 Ack=22 Win=408256 Len=0 TSval=289672700 TSecr=...
23	131.693789	127.0.0.1	127.0.0.1	TCP	57	1234 → 51107 [PSH, ACK] Seq=22 Ack=10 Win=408256 Len=1 TSval=289672700 TS...
24	131.693824	127.0.0.1	127.0.0.1	TCP	56	51107 → 1234 [ACK] Seq=10 Ack=23 Win=408256 Len=0 TSval=289672700 TSecr=...

Loopback: lo0: <live capture in progress>

Packets: 24 · Displayed: 24 (100.0%)

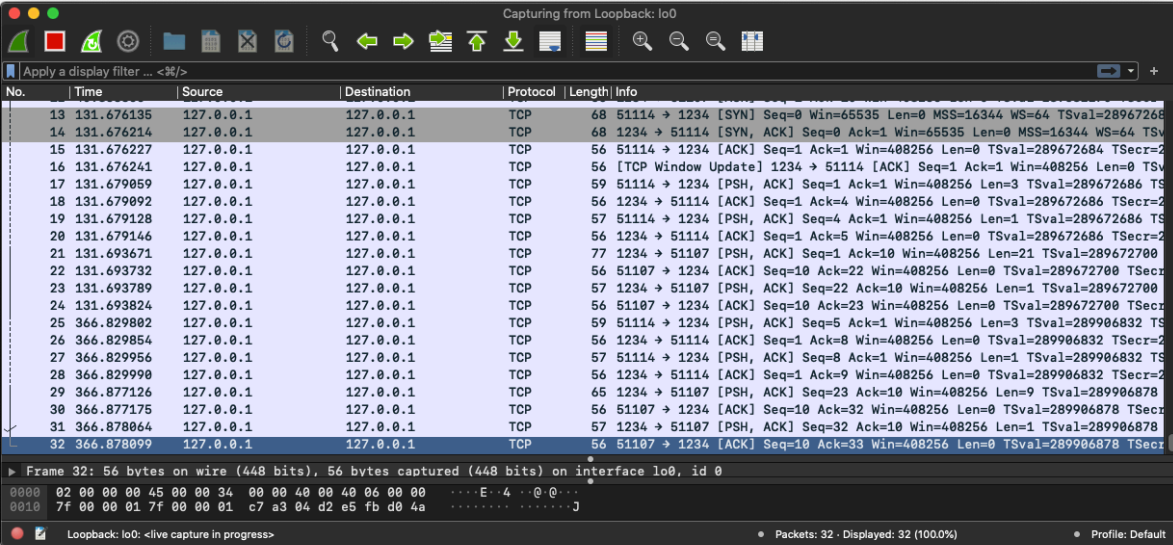
Profile: Default

Frame 24: 56 bytes on wire (448 bits), 56 bytes captured (448 bits) on interface lo0, id 0

0000 02 00 00 00 45 00 00 34 00 00 40 00 40 06 00 00E-4-@-0...

0010 7f 00 00 01 7f 00 00 01 c7 a3 04 d2 e5 fb d0 4aJ

Sending one message to server with two connected clients



No.	Time	Source	Destination	Protocol	Length	Info
13	131.676135	127.0.0.1	127.0.0.1	TCP	68	51114 → 1234 [SYN] Seq=0 Win=65535 Len=0 MSS=16344 WS=64 TSval=28967268
14	131.676214	127.0.0.1	127.0.0.1	TCP	68	1234 → 51114 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=16344 WS=64 TSv
15	131.676227	127.0.0.1	127.0.0.1	TCP	56	51114 → 1234 [ACK] Seq=1 Ack=1 Win=408256 Len=0 TSval=289672686 TSecr=2
16	131.676241	127.0.0.1	127.0.0.1	TCP	56	[TCP Window Update] 1234 → 51114 [ACK] Seq=1 Ack=1 Win=408256 Len=0 TSv
17	131.679059	127.0.0.1	127.0.0.1	TCP	59	51114 → 1234 [PSH, ACK] Seq=1 Ack=1 Win=408256 Len=3 TSval=289672686 TS
18	131.679092	127.0.0.1	127.0.0.1	TCP	56	1234 → 51114 [ACK] Seq=1 Ack=4 Win=408256 Len=0 TSval=289672686 TSecr=2
19	131.679128	127.0.0.1	127.0.0.1	TCP	57	51114 → 1234 [PSH, ACK] Seq=4 Ack=1 Win=408256 Len=1 TSval=289672686 TS
20	131.679146	127.0.0.1	127.0.0.1	TCP	56	1234 → 51114 [ACK] Seq=1 Ack=5 Win=408256 Len=0 TSval=289672686 TSecr=2
21	131.693671	127.0.0.1	127.0.0.1	TCP	77	1234 → 51107 [PSH, ACK] Seq=1 Ack=10 Win=408256 Len=21 TSval=289672700
22	131.693732	127.0.0.1	127.0.0.1	TCP	56	51107 → 1234 [ACK] Seq=10 Ack=22 Win=408256 Len=0 TSval=289672700 TSecr
23	131.693789	127.0.0.1	127.0.0.1	TCP	57	1234 → 51107 [PSH, ACK] Seq=22 Ack=10 Win=408256 Len=1 TSval=289672700
24	131.693824	127.0.0.1	127.0.0.1	TCP	56	51107 → 1234 [ACK] Seq=10 Ack=23 Win=408256 Len=0 TSval=289672700 TSecr
25	366.829802	127.0.0.1	127.0.0.1	TCP	59	51114 → 1234 [PSH, ACK] Seq=5 Ack=1 Win=408256 Len=3 TSval=289906832 TS
26	366.829854	127.0.0.1	127.0.0.1	TCP	56	1234 → 51114 [ACK] Seq=1 Ack=8 Win=408256 Len=0 TSval=289906832 TSecr=2
27	366.829956	127.0.0.1	127.0.0.1	TCP	57	51114 → 1234 [PSH, ACK] Seq=8 Ack=1 Win=408256 Len=1 TSval=289906832 TS
28	366.829998	127.0.0.1	127.0.0.1	TCP	56	1234 → 51114 [ACK] Seq=1 Ack=9 Win=408256 Len=0 TSval=289906832 TSecr=2
29	366.877126	127.0.0.1	127.0.0.1	TCP	65	1234 → 51107 [PSH, ACK] Seq=23 Ack=10 Win=408256 Len=9 TSval=289906878
30	366.877175	127.0.0.1	127.0.0.1	TCP	56	51107 → 1234 [ACK] Seq=10 Ack=32 Win=408256 Len=0 TSval=289906878 TSecr
31	366.878064	127.0.0.1	127.0.0.1	TCP	57	1234 → 51107 [PSH, ACK] Seq=32 Ack=10 Win=408256 Len=1 TSval=289906878
32	366.878099	127.0.0.1	127.0.0.1	TCP	56	51107 → 1234 [ACK] Seq=10 Ack=33 Win=408256 Len=0 TSval=289906878 TSecr

Frame 32: 56 bytes on wire (448 bits), 56 bytes captured (448 bits) on interface lo0, id 0

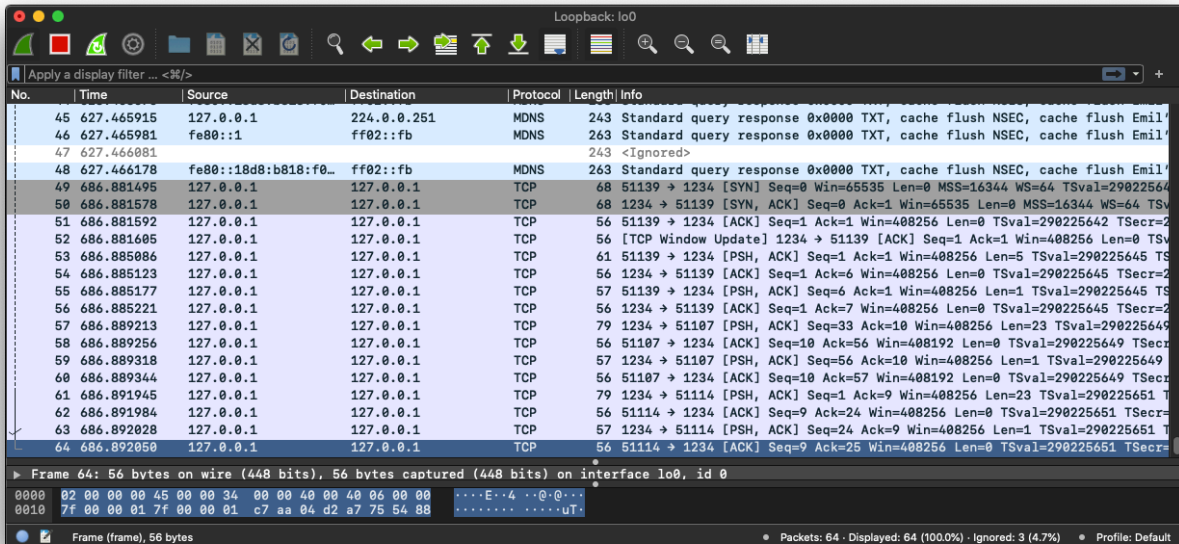
0000 02 00 00 00 45 00 00 34 00 00 40 00 40 06 00 00 ...E..4..@..
0010 7f 00 00 01 7f 00 00 01 c7 a3 04 d2 e5 fb d0 4aJ

Loopback: lo0: <live capture in progress> Packets: 32 · Displayed: 32 (100.0%) Profile: Default

Here we see that sending one message with two connected clients results in 8 messages. Which is reasonable since it was 4 rows with only one client.

Connecting third client to server

When connecting a third client it results in the usual handshake of three rows and 12 rows for the informative message that a new user has connected to the chat.



No.	Time	Source	Destination	Protocol	Length	Info
45	627.465915	127.0.0.1	224.0.0.251	MDNS	243	Standard query response 0x0000 TXT, cache flush NSEC, cache flush Emil
46	627.465981	fe80::1	ff02::fb	MDNS	263	Standard query response 0x0000 TXT, cache flush NSEC, cache flush Emil
47	627.466081				243	<Ignored>
48	627.466178	fe80::18d8:b818:f0...	ff02::fb	MDNS	263	Standard query response 0x0000 TXT, cache flush NSEC, cache flush Emil
49	686.881495	127.0.0.1	127.0.0.1	TCP	68	51139 → 1234 [SYN] Seq=0 Win=65535 Len=0 MSS=16344 WS=64 TSval=29022564
50	686.881578	127.0.0.1	127.0.0.1	TCP	68	1234 → 51139 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=16344 WS=64 TSv
51	686.881592	127.0.0.1	127.0.0.1	TCP	56	51139 → 1234 [ACK] Seq=1 Ack=1 Win=408256 Len=0 TSval=290225642 TSecr=2
52	686.881605	127.0.0.1	127.0.0.1	TCP	56	[TCP Window Update] 1234 → 51139 [ACK] Seq=1 Ack=1 Win=408256 Len=0 TSv
53	686.885086	127.0.0.1	127.0.0.1	TCP	61	51139 → 1234 [PSH, ACK] Seq=1 Ack=1 Win=408256 Len=5 TSval=290225645 TS
54	686.885123	127.0.0.1	127.0.0.1	TCP	56	1234 → 51139 [ACK] Seq=1 Ack=6 Win=408256 Len=0 TSval=290225645 TSecr=2
55	686.885177	127.0.0.1	127.0.0.1	TCP	57	51139 → 1234 [PSH, ACK] Seq=6 Ack=1 Win=408256 Len=1 TSval=290225645 TS
56	686.885221	127.0.0.1	127.0.0.1	TCP	56	1234 → 51139 [ACK] Seq=1 Ack=7 Win=408256 Len=0 TSval=290225645 TSecr=2
57	686.889213	127.0.0.1	127.0.0.1	TCP	79	1234 → 51107 [PSH, ACK] Seq=33 Ack=10 Win=408256 Len=23 TSval=290225649
58	686.889256	127.0.0.1	127.0.0.1	TCP	56	51107 → 1234 [ACK] Seq=10 Ack=56 Win=408192 Len=0 TSval=290225649 TSecr
59	686.889318	127.0.0.1	127.0.0.1	TCP	57	1234 → 51107 [PSH, ACK] Seq=56 Ack=10 Win=408256 Len=1 TSval=290225649
60	686.889344	127.0.0.1	127.0.0.1	TCP	56	51107 → 1234 [ACK] Seq=10 Ack=57 Win=408192 Len=0 TSval=290225649 TSecr
61	686.891945	127.0.0.1	127.0.0.1	TCP	79	1234 → 51114 [PSH, ACK] Seq=1 Ack=9 Win=408256 Len=23 TSval=290225651 T
62	686.891984	127.0.0.1	127.0.0.1	TCP	56	51114 → 1234 [ACK] Seq=9 Ack=24 Win=408256 Len=0 TSval=290225651 TSecr
63	686.892028	127.0.0.1	127.0.0.1	TCP	57	1234 → 51114 [PSH, ACK] Seq=24 Ack=9 Win=408256 Len=1 TSval=290225651 T
64	686.892050	127.0.0.1	127.0.0.1	TCP	56	51114 → 1234 [ACK] Seq=9 Ack=25 Win=408256 Len=0 TSval=290225651 TSecr

Frame 64: 56 bytes on wire (448 bits), 56 bytes captured (448 bits) on interface lo0, id 0

0000 02 00 00 00 45 00 00 34 00 00 40 00 40 06 00 00 ...E..4..@..
0010 7f 00 00 01 7f 00 00 01 c7 aa 04 d2 a7 75 54 88UT.

Frame (frame), 56 bytes Packets: 64 · Displayed: 64 (100.0%) · Ignored: 3 (4.7%) Profile: Default

Message from third client to server

No.	Time	Source	Destination	Protocol	Length	Info
57	686.889213	127.0.0.1	127.0.0.1	TCP	79	1234 → 51107 [PSH, ACK] Seq=33 Ack=10 Win=408256 Len=23 TSval=290225649
58	686.889256	127.0.0.1	127.0.0.1	TCP	56	51107 → 1234 [ACK] Seq=10 Ack=56 Win=408192 Len=0 TSval=290225649 TSecr=
59	686.889318	127.0.0.1	127.0.0.1	TCP	57	1234 → 51107 [PSH, ACK] Seq=56 Ack=10 Win=408256 Len=1 TSval=290225649
60	686.889344	127.0.0.1	127.0.0.1	TCP	56	51107 → 1234 [ACK] Seq=10 Ack=57 Win=408192 Len=0 TSval=290225649 TSecr=
61	686.891945	127.0.0.1	127.0.0.1	TCP	79	1234 → 51114 [PSH, ACK] Seq=1 Ack=9 Win=408256 Len=23 TSval=290225651 TSecr=
62	686.891984	127.0.0.1	127.0.0.1	TCP	56	51114 → 1234 [ACK] Seq=9 Ack=24 Win=408256 Len=0 TSval=290225651 TSecr=
63	686.892028	127.0.0.1	127.0.0.1	TCP	57	1234 → 51114 [PSH, ACK] Seq=24 Ack=9 Win=408256 Len=1 TSval=290225651 TSecr=
64	686.892050	127.0.0.1	127.0.0.1	TCP	56	51114 → 1234 [ACK] Seq=9 Ack=25 Win=408256 Len=0 TSval=290225651 TSecr=
65	766.616612	127.0.0.1	127.0.0.1	TCP	59	51139 → 1234 [PSH, ACK] Seq=7 Ack=1 Win=408256 Len=3 TSval=290305124 TSecr=
66	766.616652	127.0.0.1	127.0.0.1	TCP	56	1234 → 51139 [ACK] Seq=1 Ack=10 Win=408256 Len=0 TSval=290305124 TSecr=
67	766.616680	127.0.0.1	127.0.0.1	TCP	59	51139 → 1234 [PSH, ACK] Seq=10 Ack=1 Win=408256 Len=1 TSval=290305124 TSecr=
68	766.616707	127.0.0.1	127.0.0.1	TCP	56	1234 → 51139 [ACK] Seq=1 Ack=11 Win=408256 Len=0 TSval=290305124 TSecr=
69	766.617675	127.0.0.1	127.0.0.1	TCP	67	1234 → 51107 [PSH, ACK] Seq=57 Ack=10 Win=408256 Len=11 TSval=290305125 TSecr=
70	766.617698	127.0.0.1	127.0.0.1	TCP	56	51107 → 1234 [ACK] Seq=10 Ack=68 Win=408192 Len=0 TSval=290305125 TSecr=
71	766.618130	127.0.0.1	127.0.0.1	TCP	57	1234 → 51107 [PSH, ACK] Seq=68 Ack=10 Win=408256 Len=1 TSval=290305125 TSecr=
72	766.618158	127.0.0.1	127.0.0.1	TCP	56	51107 → 1234 [ACK] Seq=10 Ack=69 Win=408192 Len=0 TSval=290305125 TSecr=
73	766.618304	127.0.0.1	127.0.0.1	TCP	67	1234 → 51114 [PSH, ACK] Seq=25 Ack=9 Win=408256 Len=11 TSval=290305125 TSecr=
74	766.618333	127.0.0.1	127.0.0.1	TCP	56	51114 → 1234 [ACK] Seq=9 Ack=36 Win=408256 Len=0 TSval=290305125 TSecr=
75	766.618356	127.0.0.1	127.0.0.1	TCP	57	1234 → 51114 [PSH, ACK] Seq=36 Ack=9 Win=408256 Len=1 TSval=290305125 TSecr=
76	766.618381	127.0.0.1	127.0.0.1	TCP	56	51114 → 1234 [ACK] Seq=9 Ack=37 Win=408256 Len=0 TSval=290305125 TSecr=

Frame 76: 56 bytes on wire (448 bits), 56 bytes captured (448 bits) on interface lo0, id 0

```

0000 02 00 00 00 45 00 00 34 00 00 40 00 00 06 00 00  ....E..4..@..
0010 7f 00 00 01 7f 00 00 01 c7 aa 04 d2 a7 75 54 88  .....uT.....

```

Frame (frame), 56 bytes

Packets: 76 - Displayed: 76 (100.0%) - Ignored: 3 (3.9%) - Profile: Default

Here we see that sending this message results in 12 rows, which is logical since each new user results in 4 more rows.

Difference between chat and web browser traffic

The most obvious difference between these are that the web browser sends a lot more and different headers and flags than we get in our chat. The browser is also using TLS (transport layer security) which the chat is not using in this implementation. Furthermore, the web browser does not close its connection after every bit of data, i.e. it is using HTTP Keep Alive where it checks the connection/socket periodically (for incoming HTTP requests). The browser is also varying its window since while the chat keeps it the same. The chat is very sequential in its use of acknowledgements while the browser acknowledges in groups of 2-3. We can also see some DNS lookups in the sniffing results. We can also see some lesser known protocols like SSDP and ICMP and some local traffic that uses MDNS and ARP.