

Advanced Programming – Assignment

Part 2 - Dublin Events

Student: Maximiliano Herrera

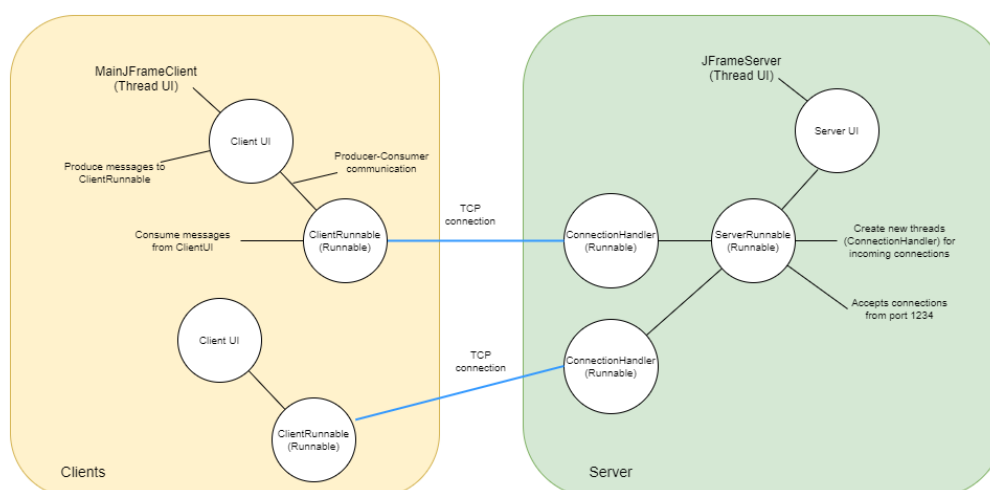
Student number: 20103212

Contents

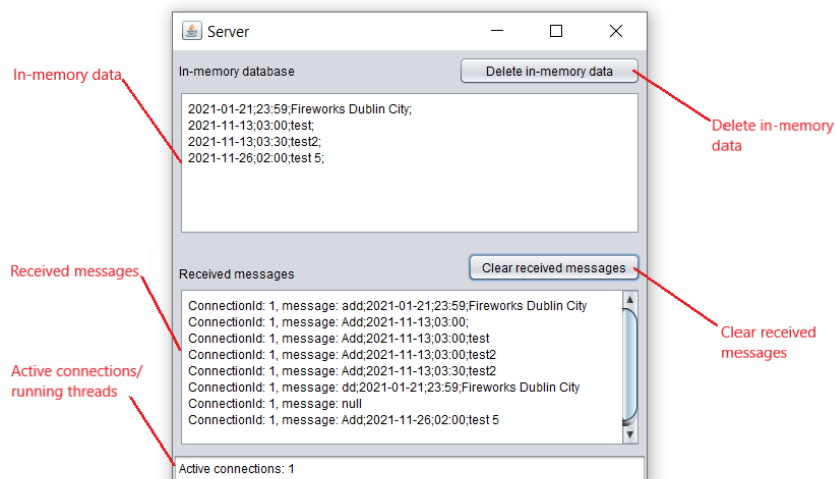
Architecture	1
Server	2
Client	2
Multiple connections	2
Adding an event	3
Removing events.....	3
Before removing event	3
After removing event.....	3
Multiple messages within a connection	3
Links	4
Client	4
Server	4

Architecture

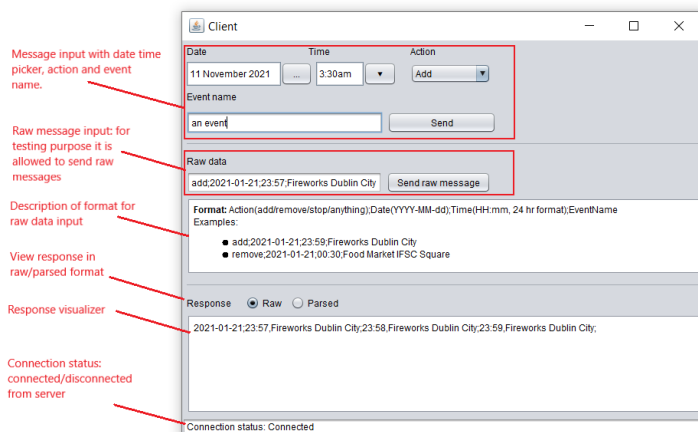
Client-Server architecture which supports multiple clients, each of them handling its own TCP connection to the only one server. Below it is shown the architecture for an example of two clients and one server from a Runnable point of view, which means each circle is a thread.



Server



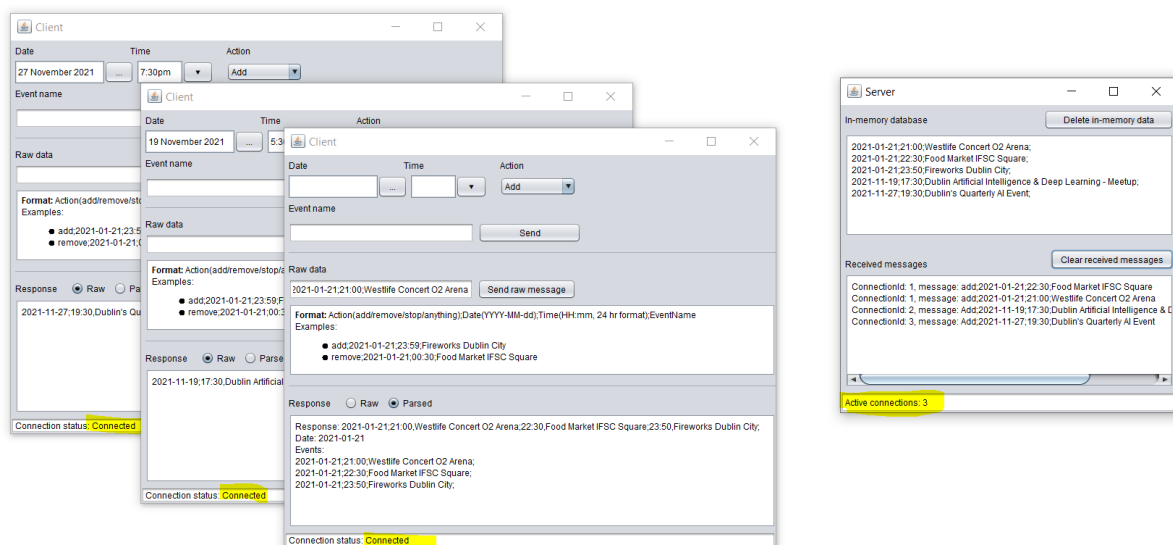
Client



Multiple connections

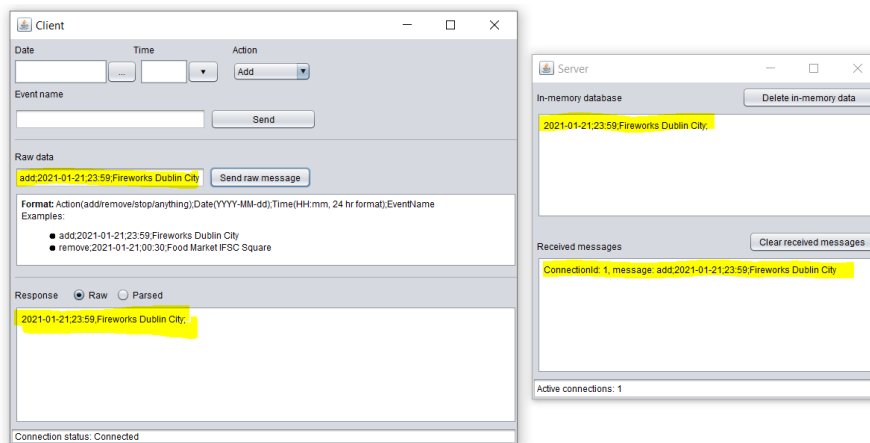
The server displays how many clients are currently connected through the "Active connections" status bar.

The Client shows if it is connected or disconnected to a server on the "Connection status" bar.



Adding an event

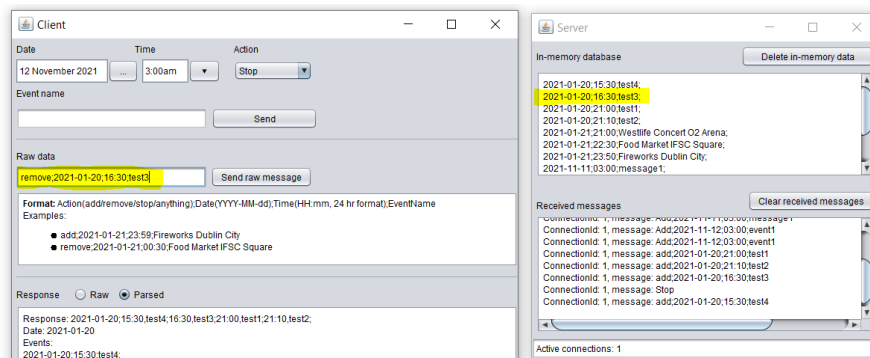
Example of adding an event and the response shown on client side as well as the in-memory database content on server side.



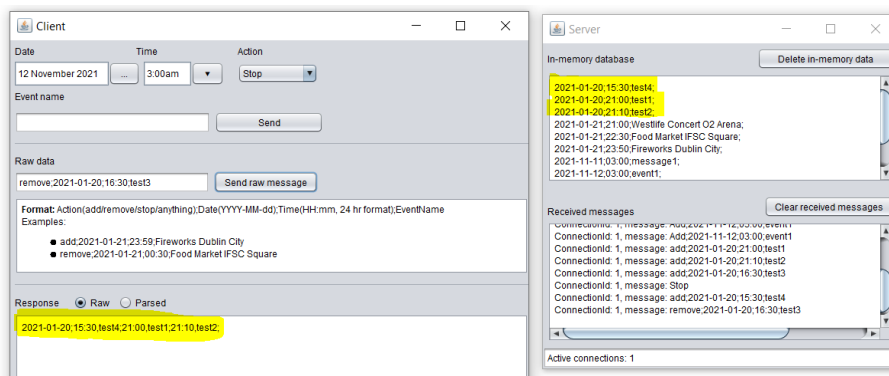
Removing events

This is an example of removing an event and showing how the UI looks like before and after in both client and server.

Before removing event



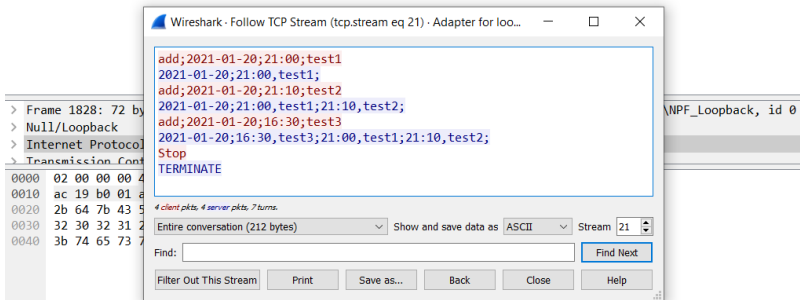
After removing event



Multiple messages within a connection

We can see from Wireshark that Client can send multiple messages within a TCP connection until the Stop-Terminate sequence is performed and the TCP connection is closed. As evidence of this please visualize the package No. 5918 and TCP stream window which are shown below:

No.	Time	Source	Destination	Protocol	Length	Info
1828	33.430805	172.25.176.1	172.25.176.1	TCP	72	53198 → 1234 [PSH, ACK] Seq=1 Ack=1 Win=10231 Len=28
1829	33.430931	172.25.176.1	172.25.176.1	TCP	44	1234 → 53198 [ACK] Seq=1 Ack=29 Win=10232 Len=0
1830	33.431853	172.25.176.1	172.25.176.1	TCP	69	1234 → 53198 [PSH, ACK] Seq=1 Ack=29 Win=10232 Len=25
1831	33.431914	172.25.176.1	172.25.176.1	TCP	44	53198 → 1234 [ACK] Seq=29 Ack=26 Win=10231 Len=0
3838	109.642...	172.25.176.1	172.25.176.1	TCP	72	53198 → 1234 [PSH, ACK] Seq=29 Ack=26 Win=10231 Len=28
3839	109.643...	172.25.176.1	172.25.176.1	TCP	44	1234 → 53198 [ACK] Seq=26 Ack=57 Win=10232 Len=0
3840	109.643...	172.25.176.1	172.25.176.1	TCP	81	1234 → 53198 [PSH, ACK] Seq=26 Ack=57 Win=10232 Len=37
3841	109.643...	172.25.176.1	172.25.176.1	TCP	44	53198 → 1234 [ACK] Seq=57 Ack=63 Win=10231 Len=0
5770	161.960...	172.25.176.1	172.25.176.1	TCP	72	53198 → 1234 [PSH, ACK] Seq=57 Ack=63 Win=10231 Len=28
5771	161.960...	172.25.176.1	172.25.176.1	TCP	44	1234 → 53198 [ACK] Seq=63 Ack=85 Win=10232 Len=0
5772	161.961...	172.25.176.1	172.25.176.1	TCP	93	1234 → 53198 [PSH, ACK] Seq=63 Ack=85 Win=10232 Len=49
5773	161.961...	172.25.176.1	172.25.176.1	TCP	44	53198 → 1234 [ACK] Seq=85 Ack=112 Win=10231 Len=0
5914	178.251...	172.25.176.1	172.25.176.1	TCP	50	53198 → 1234 [PSH, ACK] Seq=85 Ack=112 Win=10231 Len=6
5915	178.251...	172.25.176.1	172.25.176.1	TCP	44	1234 → 53198 [ACK] Seq=112 Ack=91 Win=10232 Len=0
5916	178.254...	172.25.176.1	172.25.176.1	TCP	55	1234 → 53198 [PSH, ACK] Seq=112 Ack=91 Win=10232 Len=11
5917	178.254...	172.25.176.1	172.25.176.1	TCP	44	53198 → 1234 [ACK] Seq=91 Ack=123 Win=10231 Len=0
5918	178.254...	172.25.176.1	172.25.176.1	TCP	44	53198 → 1234 [FIN, ACK] Seq=91 Ack=123 Win=10231 Len=0
5919	178.254...	172.25.176.1	172.25.176.1	TCP	44	1234 → 53198 [ACK] Seq=123 Ack=92 Win=10232 Len=0
5920	178.255...	172.25.176.1	172.25.176.1	TCP	44	1234 → 53198 [FIN, ACK] Seq=123 Ack=92 Win=10232 Len=0
5921	178.255...	172.25.176.1	172.25.176.1	TCP	44	53198 → 1234 [ACK] Seq=92 Ack=124 Win=10231 Len=0



Links

[Github repository](#)

Client

[MainJFrameClient.java](#): Starting point, Client UI.

[ClientRunnable.java](#): Receive messages from Client UI and send those messages to server through a TCP connection. The TCP connection is open until receiving TERMINATE message from server.

[ResponseCommandBuilderTest.java](#): Test cases for command response builder component.

Server

[JFrameServer.java](#): Starting point, Server UI.

[ServerRunnable.java](#): Listen on port 1234 for incoming connections and create a tuple socket-thread (ConnectionHandler) for each of them.

[ConnectionHandler.java](#): Handle the messages from client and execute commands until receiving Stop message.

[InMemoryDatabase.java](#): Store in-memory data, defines add and remove methods both have linear time complexity $O(N)$.