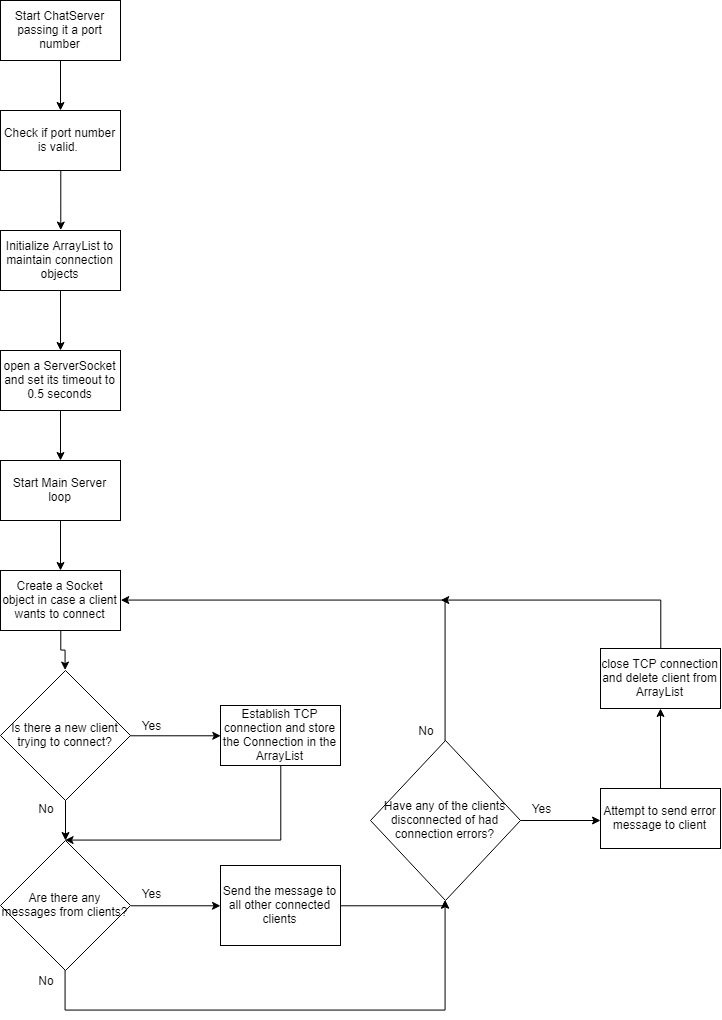
Program 1 Result report

Joshua Landron

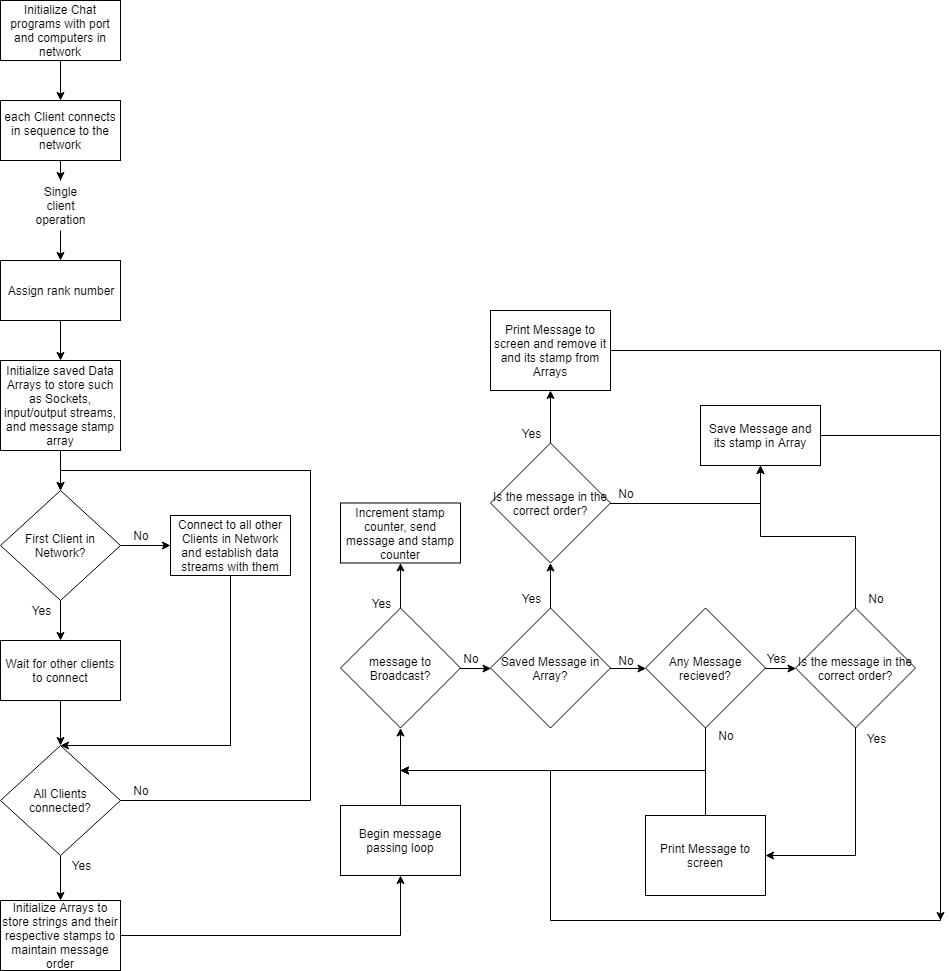
CSS434 Spring19

# Algorithm Documentation:

Part 1: Chat-Server Overview and decision flow



Part 2: Serverless Chat Overview and decision flow



# Execution Output:

Figure 1: ChatServer and ChatClients execution.

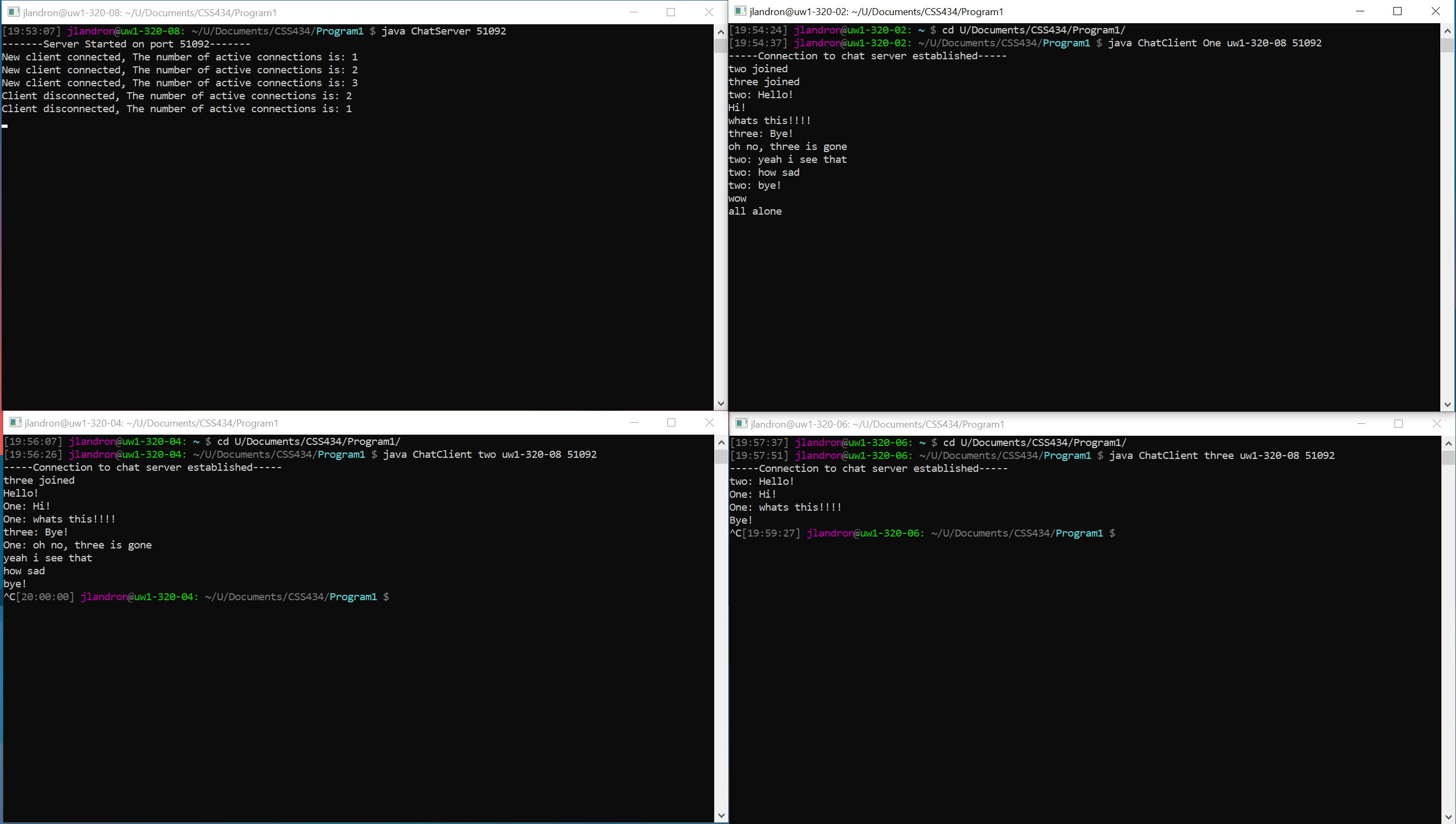
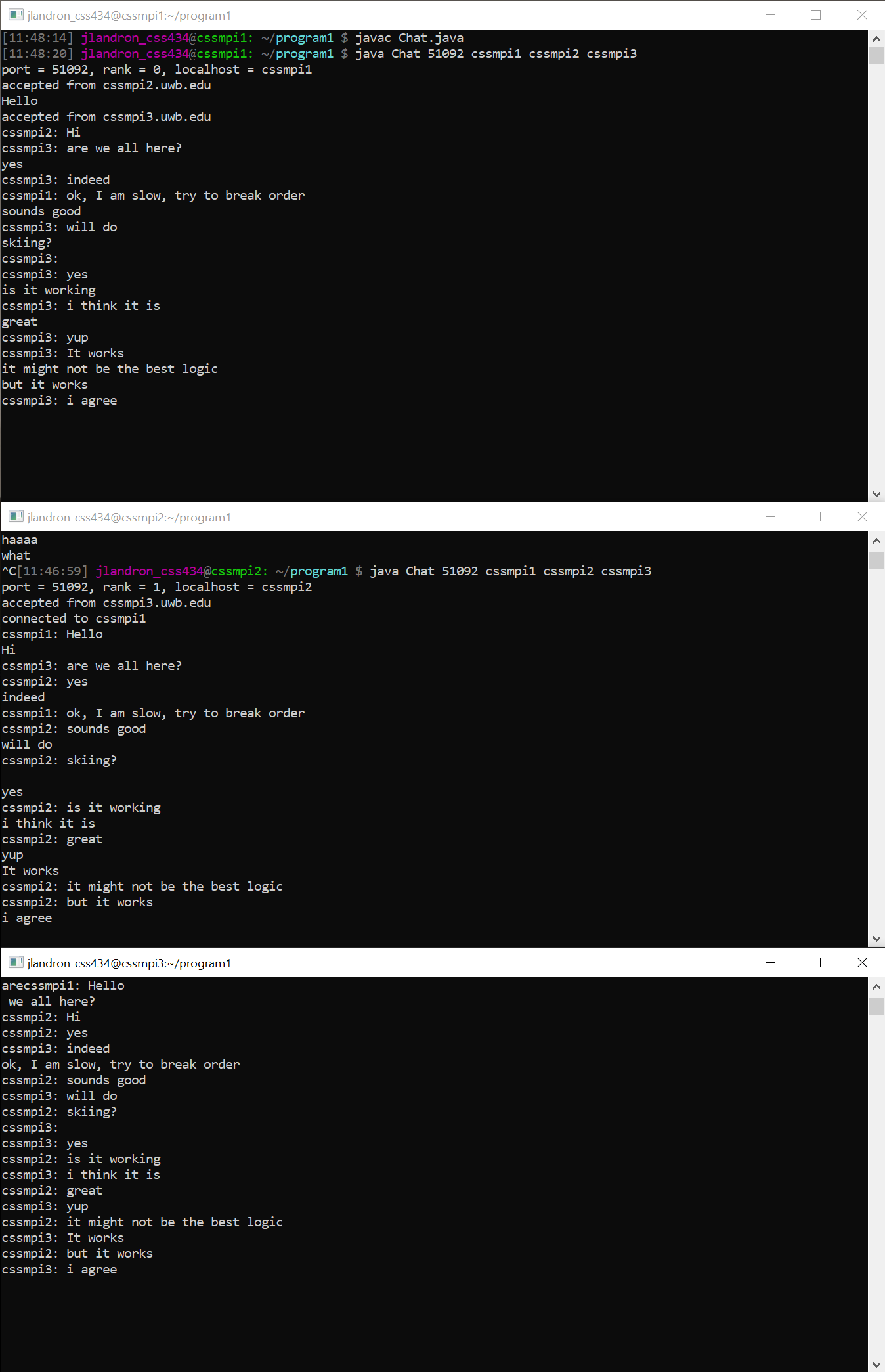


Figure 2: Causal ordering maintained.

# Discussions:

## Part 1:

My Chat-Server loops through the array of connected clients checking if they have sent a message, and this is a rather simple algorithm that maintains ordering of messages inherently. It does not try to read a message unless there is one waiting to be read, so it will not get stuck trying to read a message. The Server continues this loop while checking for new Server connections with a 0.5 second delay before continuing the loop. One improvement that I think would help is to only check for new connections at a predetermined number second instead of every loop. This way it can focus on sending and receiving messages, while checking its port for new connections regularly. This change may show in an increase of active chat response times, while new Clients may see a slowdown in establishing a new connection to the Chat-Server.

## Part 2:

The Serverless Chat program is more complex than a Chat Server, and therefore has a few downsides to address. The number of Clients that will connect to the network must be established and all clients connected in the order that they were declared. This is a large problem that I think makes this type of program worse that the Chat-Server style of message passing. My algorithm for finding messages and maintaining causal ordering is simple and keeps a small Array of out of order messages in the background that are waiting to be printed. This action happens on a per-client basis and helps to decentralize the computing power needed. I think the best change that could be made to this program, if it is possible using standard TCP connections, would be to assign the rank 0 computer the role of server in the background, and always pass messages to that client as well as allowing them to chat. This would eliminate the need to check for message order but would cause the rank 0 client to always have a larger workload assigned to them.