Auction System

Group 3: Peter Hösch, Sena Tarpan, and Yun Ye

1 Introduction

Our goal in this project is to design an auction system which enables a seller to sell a single product to a group of buyers. The buyers can place bids on the product. The product will be sold to the buyer who placed the highest bid at a predefined end date for the price of said last bid.

2 Project requirements analysis

2.1 Features

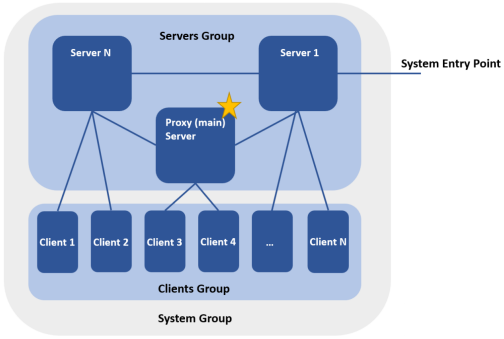

Fig. 1. Architecture diagram

The seller defines a start and end date for the auction, describes the product as well as a starting bid. The buyers get informed about the the details of the product, the end date and the current highest bid every time said bid changes. Nodes can join or leave the auction at any time. The system should be able to recognize if a buyer goes offline or if a server crashes and take appropriate action. The server should be able handle high amounts of buyers by delegating work to supplemental servers if needed.

2.2 Implementation

Architectural Description The system will be implemented as a many servers-many clients design. The servers are the seller who functions as the main server

as well as supplemental servers that provide fault tolerance and scalability. The main server functions as a sequencer. The clients on the other hand works like a thin client machine that provide merely an interface with a very restricted logic and data functionality. The supplemental servers exist to take bids, aggregate them and transfer that data to the main server. For this purpose, each server will be connected to a number of clients. The clients only communicate with this server, in the following called their contact server, not directly with the main server or with each other. Bids will be placed by using UDP connections from a client to a server to ensure that the bid will be reliably transported.

Dynamic discovery of hosts The servers will use UDP with ordered reliable multicast for the host discovery process.

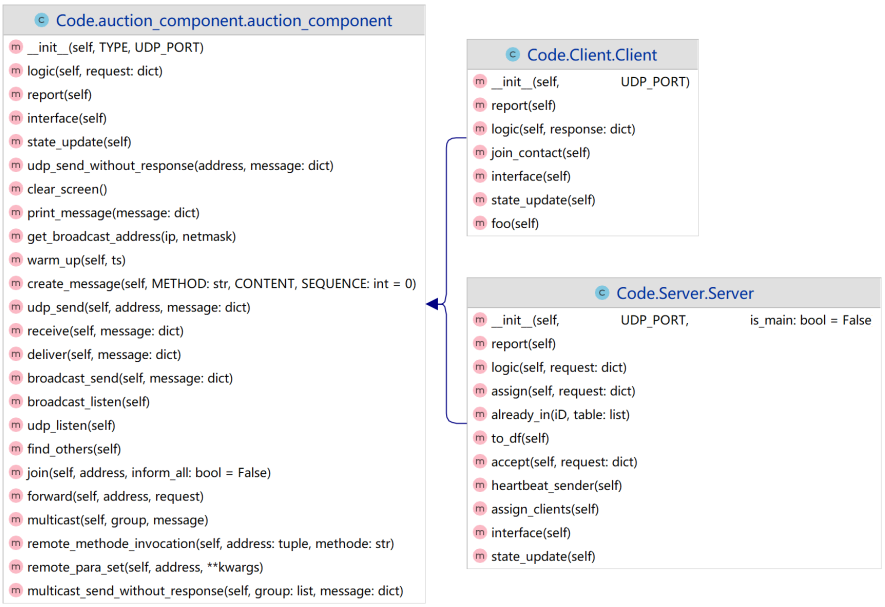

Fig. 2. Object Class in the Implementation

Fault Tolerance Should the main server stop responding, the remaining servers will replace them by using the mechanism described in the voting section. All servers maintain their own data and regularly synchronize with each other. To ensure that we become aware of lost connections or crashes, all participants in the auction are required to send a heartbeat signal. The client send this signal to their contact server, the supplemental servers send it to the main server, and the

main server send it to all the supplemental servers. If this signal doesn't arrives in a certain timeframe, the system will assume that the node that should have sent it is no longer available and proceed to initiate actions to replace them, if possible.

Voting In the case that the main server is no longer available, the other servers will vote for a new main server from among their group. All the servers are organized in a ring structure. The LaLann-Chang-Roberts algorithm. After the election, the information about the new main server will be propagated back to all clients, and the new main server will reorganize the structure of the network.

Ordered Reliable Multicast The servers will use UDP with totally ordered reliable multicast, in addition to the already mentioned for the host discovery process, to inform all interested clients in a raised bid, and to regularly synchronize the current time which is important due to the time critical nature of an auction. If multiple bids with the same value get placed during a short timeframe the system should be able to determine which one was first and as such counts. As mentioned above, the main server serves as the sequencer for this process. When a server gets the information about a raised bid from one of his clients, it will request a sequence number from the main server and then inform all participants of the new highest bid by using a multicast.

3 Current Implementation

Fig. 3. Message passing example in the system

At the moment we've finished the implementation of the dynamic discovery. Both Servers and Clients are able to send out broad-cast messages to either find each other or form the auction group. The messages will be encoded in a json file in a similar way as the http request as shown in figure 3. The main server will recognize different types of request and distribute the loads among the server group. All those processes can be done automatically after the start signal sent out.

And we've also tried remote method invocation and remote parameter changing during the dynamic discovering process to simplify some of the procedures.For further information about the implementation please refer to our Github repository[1].

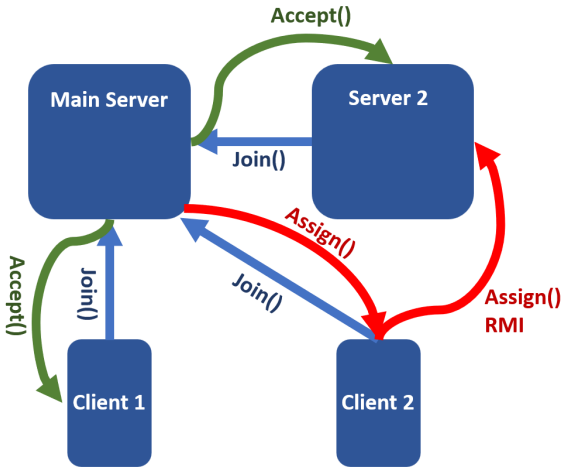

Fig. 4. The logic function that been used in the implementation. The main server is responsible for the coordination among servers and do performance balance if necessary.

[1] https://github.com/Lukasye/ds_project_ws2223

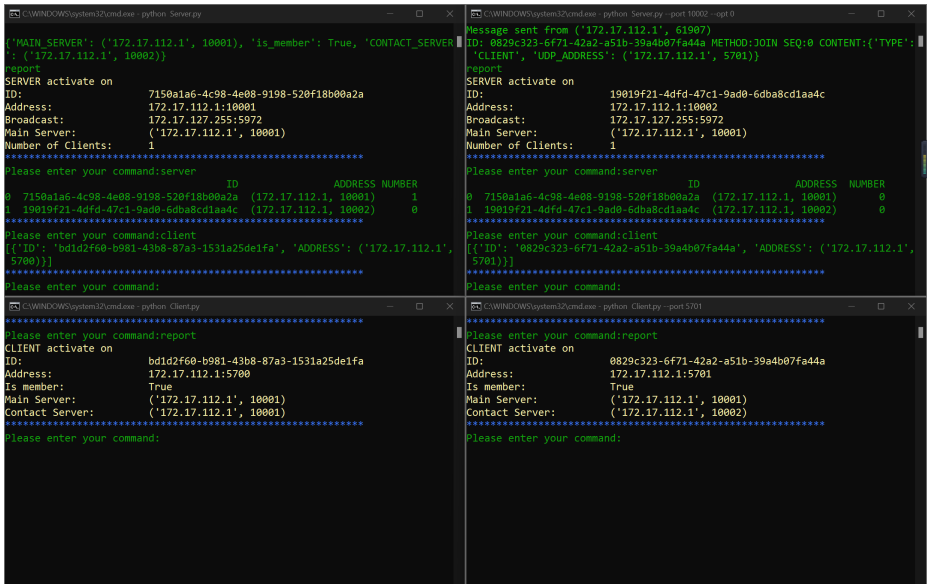

Fig. 5. Result of dynamic discovery: 1. The main server(right upper) send out broad cast to find another server(left upper) and a client(left bottom). 2. Another client(right bottom) trying to join the group and be assigned to the right server.