

# HandIn 5: A Distributed Auction System

## Introduction

We have made a program where one or more clients can bid in a virtual auction on a local composite server. This server is composed of multiple servers and is thus more resistant to loss of data as a result of a sudden crash. The goal is for the system to be resilient to a sudden system-failure of all but one of the servers.

## Architecture

The system consists of two kinds of nodes, clients and servers. When a server is started, it will begin listening for requests on a given port, ranging from 5000-5002 (default 5000). From here it awaits commands from any clients that connect to its port.

When a client is created, it will attempt to connect to the servers with the ports 5000-5002 and save these in a list. From here it awaits input from the user using the command line where two different methods can be run, *bid* and *result*.

Running the *bid* method will take a number from the user and package this into a RPC call, together with the client name and ID. This call is made to all of the servers in the list. Each server that successfully receives this call will run their *auction* method and check through the message and if no errors occurred, it will then save or overwrite previous bidders info. Finally, if this is the first bid, it will then start the countdown of the auction.

Running the *result* method will send a RPC request to all servers in the server list and then compare these answers to see which answer is most occurring and print it, depending on what kind of answer it is. The servers receiving it will send two different answers depending if the auction has begun or in process, or if it has ended, where in the first case it will send time left and highest bidding, while in the latter it will send the winner name and final amount. This is done through two different message types.

## Correctness 1

*Argue whether your implementation satisfies linearizability or sequential consistency. In order to do so, first, you must state precisely what such property is.*

Linearizability of a program is when the ordering of calls follows real-time ordering, that is, when a user from one server does something and afterward another user does something on another server, then the sequence on the program is the same as real life. For sequential consistency, the ordering of each individual server has to follow the sequence of calls, but not necessarily follow real-time, so two different servers can each have their own calls, that each are ordered sequentially correct, but compared to each other by real-time they can skew.

Our program is first of all sequentially consistent, since each server that receives a call also immediately reacts to it accordingly, thus the ordering of requests matches the ordering of processes. The system is also partly linearizable, since each request from a user is the same on each server, though there is a tiny time delay between each, though practically this is a small gap.

## Correctness 2

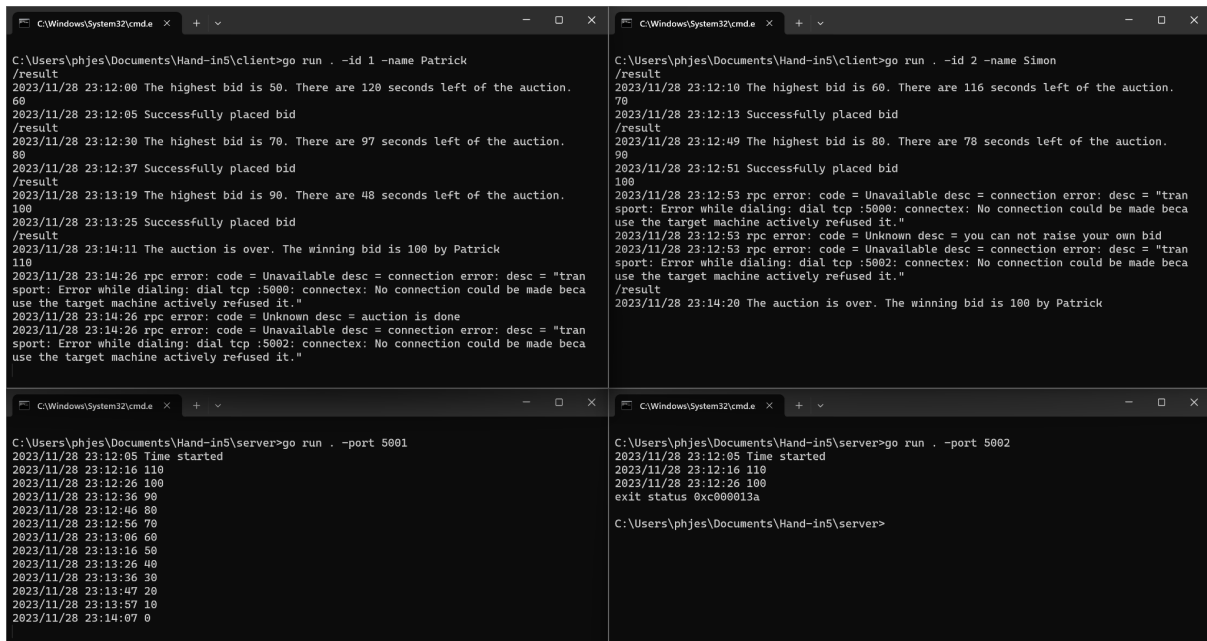
*An argument that your protocol is correct in the absence and the presence of failures*

The program is resilient to failures since we're using a leaderless replication, as a client is sending and receiving requests to and from all servers. If all but one server crashes, the program will remain functioning. It is worth noting that if crashed servers reboot, their individual data will still remain lost since servers aren't communicating to each other.

## Git Repository

<https://github.com/ITU-DISYS23-GROUPPandemicHalfPals/Hand-in5>

# Appendix



```
C:\Windows\System32\cmd.exe x + v x
C:\Users\phjes\Documents\Hand-in5\client>go run . -id 1 -name Patrick
/result
2023/11/28 23:12:00 The highest bid is 50. There are 120 seconds left of the auction.
60
2023/11/28 23:12:05 Successfully placed bid
/result
2023/11/28 23:12:30 The highest bid is 70. There are 97 seconds left of the auction.
80
2023/11/28 23:12:37 Successfully placed bid
/result
2023/11/28 23:13:19 The highest bid is 90. There are 48 seconds left of the auction.
100
2023/11/28 23:13:25 Successfully placed bid
/result
2023/11/28 23:14:11 The auction is over. The winning bid is 100 by Patrick
110
2023/11/28 23:14:26 rpc error: code = Unavailable desc = connection error: desc = "tran
sport: Error while dialing: dial tcp :5000: connectex: No connection could be made beca
use the target machine actively refused it."
2023/11/28 23:14:26 rpc error: code = Unknown desc = auction is done
2023/11/28 23:14:26 rpc error: code = Unavailable desc = connection error: desc = "tran
sport: Error while dialing: dial tcp :5002: connectex: No connection could be made beca
use the target machine actively refused it."

C:\Windows\System32\cmd.exe x + v x
C:\Users\phjes\Documents\Hand-in5\client>go run . -id 2 -name Simon
/result
2023/11/28 23:12:10 The highest bid is 60. There are 116 seconds left of the auction.
70
2023/11/28 23:12:13 Successfully placed bid
/result
2023/11/28 23:12:49 The highest bid is 80. There are 78 seconds left of the auction.
90
2023/11/28 23:12:51 Successfully placed bid
100
2023/11/28 23:12:53 rpc error: code = Unavailable desc = connection error: desc = "tran
sport: Error while dialing: dial tcp :5000: connectex: No connection could be made beca
use the target machine actively refused it."
2023/11/28 23:12:53 rpc error: code = Unknown desc = you can not raise your own bid
2023/11/28 23:12:53 rpc error: code = Unavailable desc = connection error: desc = "tran
sport: Error while dialing: dial tcp :5002: connectex: No connection could be made beca
use the target machine actively refused it."
/result
2023/11/28 23:14:20 The auction is over. The winning bid is 100 by Patrick

C:\Windows\System32\cmd.exe x + v x
C:\Users\phjes\Documents\Hand-in5\server>go run . -port 5001
2023/11/28 23:12:05 Time started
2023/11/28 23:12:16 110
2023/11/28 23:12:26 100
2023/11/28 23:12:36 90
2023/11/28 23:12:46 80
2023/11/28 23:12:56 70
2023/11/28 23:13:06 60
2023/11/28 23:13:16 50
2023/11/28 23:13:26 40
2023/11/28 23:13:36 30
2023/11/28 23:13:47 20
2023/11/28 23:13:57 10
2023/11/28 23:14:07 0

C:\Windows\System32\cmd.exe x + v x
C:\Users\phjes\Documents\Hand-in5\server>go run . -port 5002
2023/11/28 23:12:05 Time started
2023/11/28 23:12:16 110
2023/11/28 23:12:26 100
exit status 0xc000013a

C:\Users\phjes\Documents\Hand-in5\server>
```

**Figure 1:** Log of two clients and two servers having an auction. It also shows that it is possible for all but one server to crash and the auction will still run as if all servers were up and running.