SCC.311 Coursework Stage 3 Specification
Due: Friday Week 10, 5 pm via Moodle
Total mark allocation: 45%

Level 5: Passive Replication (20%)

To ensure the dependability of the auctioning system, you are required to enhance the availability of your system by using replication techniques. You should implement a **passive replication** system to meet these requirements, thereby increasing dependability. Please refer to the lectures for the definition of passive replication.

The server implementation should have at least three replicas and allow the user to easily add a new replica. You must design your server program so that any replica can function as the primary replica. The clients will communicate with a **stateless** *front-end* program which should then simply forward client requests to the *primary replica*. Your front end should initially pick one replica and keep using that replica as the primary replica.

The primary replica must ensure that all replicas maintain a *consistent view* of the auction data before responding to each request. Your front-end is required to implement the Auction interface, and you can choose for yourself exactly how the front end and replicas interact.

Requirements: Each replica must run in a separate process, and your replica server program must support a command-line parameter to assign a unique identifier (i.e., an integer) to each replica. For example, you should be able to start a replica from the command line with ID 1 as shown below:

- java Replica 1

Your front-end must use the exact interface as shown below, which introduces a new method *getPrimaryReplicaID()* that returns the identifier of the replica that is currently used as the primary one.

```
public interface Auction extends Remote {

  public Integer register(String email, PublicKey pubKey)throws
  RemoteException;

  public ChallengeInfo challenge(int userID, String
  clientChallenge) throws RemoteException;

  public TokenInfo authenticate(int userID, byte signature[])
  throws RemoteException;

  public AuctionItem getSpec(int userID, int itemID, String
  token) throws RemoteException;

  public Integer newAuction(int userID, AuctionSaleItem item,
  String token) throws RemoteException;
```

```
    public AuctionItem[] listItems(int userID, String token)
    throws RemoteException;

    public AuctionResult closeAuction(int userID, int itemID,
    String token) throws RemoteException;

    public boolean bid(int userID, int itemID, int price,
    String token) throws RemoteException;

    public int getPrimaryReplicaID() throws RemoteException;
}
```

Note: You are not required to implement the 3-way authentication for this part of the coursework if you have not completed Level 4 of Stage 2. In that case, you can simply leave the implementations of challenge() and authenticate() empty, i.e., simply return null for both methods and omit the Token verification in the rest of your code.

Level 6: Fault Tolerance (20%)

The server replicas may crash unexpectedly due to either hardware or software faults. Your solution should be able to handle such failures. As long as one replica remains alive, your auctioning system should continue to function correctly. You should be prepared to justify your choice of design for failure detection and handling.

Your solution must allow the automated testing to kill all but one of the replicas (including the primary replica). Even when the current primary replica fails, your system should continue to function correctly. Your system should also allow replicas to recover and re-join the system. Ideally, your system should work even with a complete replica turnover; that is, you want your system to start with 3 replicas (say with IDs 1, 2, and 3), add another replica (with ID 4), kill all original replicas (with IDs 1, 2, 3) and have the system continue to function properly.

Your front-end should initially pick a replica as the primary replica, and only in the event of a failure detection, the front-end should select a different (i.e., a working one) replica as the primary.

Level 7: Design (5%)

This element of the coursework relates to your overall design to date and how well-argued it is. In this diagram, you will need to include in your submission a diagram (as a PDF) in the root directory of your submission. It is desirable for the diagram to demonstrate the layering/tiering structure of your implementation.

Coursework submission instructions

Your coursework will be partly marked using an automated test system. For the test system to work properly, your submission must be contained in a zip file and have the following:

1. A shell script called *server.sh* in the root directory of your submission, which performs any set of operations necessary to get your front-end and at least three replicas (each with a unique ID) running.
2. An RMI service advertised with the name **"FrontEnd".**
3. An RMI interface which **exactly matches** the specification given in this document.

4. Your entire system must be up and running within 5 seconds (our test client is launched five seconds after your server.sh script)

You can test if your submission is valid (i.e., markable by the actual marking system) using the **ClientTest program** (to be provided in week 8). This code will check whether or not the above basic checks pass, but does not perform any logic testing or provide any indication of your mark. If ClientTest.java is unable to test your code, you will usually receive a mark of zero.

Marking Scheme:

Level 5:

Working code and the correct implementation of passive replication — 15 marks

Correctly perform all the operations, i.e. bid, list, close, etc. using 3 replicas — 5 marks

Level 6:

Handle primary replica failure — 6 marks

Handle multiple replica failures — 7 marks

Handle complete replica turnover — 7 marks

Level 7:

Architecture diagram — 2 marks

Use of layering and tiering — 3 marks
