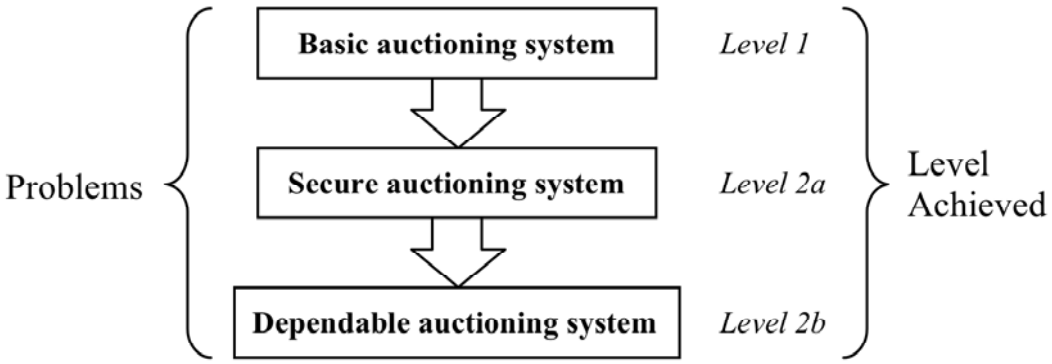
Coursework for SCC 311: Auctioning System

1. Description of the Problem

You are asked to design and implement a simple *distributed auctioning system* using **Java RMI**. Crucially, this overall problem is organised into three sub-problems of increasing difficulty: *basic auctioning system* (level 1), *secure auctioning system (level 2-a)*, and *dependable auctioning system (level 2-b)*. Each problem builds on the previous one. The level you reach determines the maximum mark you can possibly obtain for the coursework. How well you reach a level determines your actual mark.

Achieving level 1 satisfactorily is sufficient to obtain a reasonably good 2.2 mark if you completed the project retrieval exercise. Precise marking details are provided at the end of this document.



2.1 Basic Auctioning System (Level 1)

You are asked to implement a distributed auctioning system. This auctioning system should consist of a central auctioning server and two separate client programs:

- The ***first client*** program should enable a seller to create a new auction for an item offered for sale. The seller should be able to provide **a *starting price*** and ***a minimum acceptable price*** (reserve price). Creating the auction will return an auction id. At some point in the future, the seller should be able to close the auction using the same client program, by quoting the auction id. When the seller closes an auction, the client program should either indicate who the winner is, along with their details (see below), or else indicate that the reserve price has not been reached.

- The ***second client*** program should enable potential buyers to bid for auctioned items. Firstly, the program should enable buyers to browse the list of currently active auctions with their current highest bid (but not the reserve price, which is secret). The client program should then enable a buyer to bid for a selected item, by entering the buyer's details (name and e-mail).
- The ***auctioning server*** should then deal with requests from the two client programs and maintain the state of the ongoing auctions.
- **Note that we do NOT expect you to use a database system** (e.g. MYSQL) to store the auction data. In memory java containers, e.g. List, Hash Tables, should be sufficient enough.

2.2 Advanced Features for Auctioning System

Now, suppose this Auctioning system is a big success, and attracts a dramatic increase in traffic, along with threats from a powerful global crime organisation. You are asked to *consider* and *implement* techniques to enhance the *security*, *scalability* and *availability* of your system

Level 2a: **Secure Auctioning System**

The implementation of level 1 is insecure: for instance, it has no mechanism to protect the system against external attacks such as the interception of messages (thus violating the confidentiality of the reserve price for instance), tampering with bids (one buyer modifying or stopping another buyer's bid), or bidding in another buyer's name (see discussion on security vulnerabilities in the Security lecture - fundamental stream - in week 4).

Modify you system to provide a level of security. In particular, you are expected to provide the following properties:

(i) cryptographic authentication - in particular, you should use an appropriate challenge-response protocol based on either symmetric or asymmetric cryptosystems (and be prepared to defend your choice).

(ii) access control - in particular, ensuring the right level of access for different users, e.g. only the initiator and hence owner of an auction can close the auction, etc.

Note that for both of the above, you can assume that a set of users are already registered with the system and associated keys as required by your authentication solution are also already distributed to the clients and servers.

Level 2b: Dependable Auctioning System

To ensure a larger scale use of the auctioning system you are required to enhance the *availability* of your system by using replication techniques.

You should implement ***an active replication system*** to meet these requirements (thereby increasing both dependability and scalability). Please refer to the fundamental stream lectures of Fault Tolerance & Dependability for the definition of active replication. *The client and auction servers should communicate using Java RMI.*

Auction data, such as the list of available auction items and the current highest bit for each item etc., are hold by replicas. The server implementation should have *at least three replicas* and should allow the user to easily add a new replica.

All replicas must maintain *a consistent view* of the auction data (i.e. consistency). JGroups can provide a useful foundation for this purpose through its reliable group communication service. Therefore, you are expected to use JGroups to implement the group communications among replicas.

Servers may crash from time to time due to either hardware or software faults. Therefore, you solution should *detect and handle* the failure of server replicas. As long as there is one replica remains alive, the auctioning system should function correctly. You should be prepared to justify your choice of design for failure detection and handling.

For the purpose of demonstration, your solution must allow the marker to close *an arbitrary replica*, e.g. through sending a command to the replica manager, or by closing the console window where a particular replica process runs.

Note:

1. *You are not expected to provide a sophisticated user interface for the two client programs (a simple text based interface will suffice).*
2. *Crucially, this overall problem is organised into three sub-problems of increasing difficulty: basic auctioning system (level 1), and secure & dependable auctioning system (levels 2a & 2b).*
3. *Each problem builds on the previous one. The level you reach determines the maximum mark you can possibly obtain for the coursework. How well you reach a level determines you actual mark.*

3. Marking Scheme

The coursework mark for SCC 311 will be calculated as follows:

Overall Component	Additional Information	Weighting (%)
Coursework Retrieval	**Week 4**	**20%**
	Use of username/password authentication	*10%*
	Or Use of key-based authentication	*20%*
Basic Server (Level 1)	**Week 7**	**35%**
	Completion of clients and server as specified	*30%*
	Quality of the solution	*5%*
Secure & Dependable Server (Level 2a & Level 2b)	**Week 10**	**45%**
Level 2	*Authentication + access control*	*22.5%*
Level 3	*Implementation of Replication*	*22.5%*
TOTAL		**100**

This mark will then be combined with the exam mark in a ratio of 60% exam: 40% coursework to provide an overall grading for the course.

4. Deadline, marking, and submission:

IMPORTANT NOTE:
Your solution must be formally submitted through Moodle by
Wednesday (4pm) of week 7 for level 1 and of week 10 for level 2.
You will be asked to demonstrate your work from your Moodle submission.

1. Level 1 will be marked in the practical sessions of **week 7**. Levels 2a and 2b will be marked in the practical sessions of **week 10**.

2. **You need to demonstrate your work** to a member of the course team. Failing to do so will result in a grade of 0.

3. You need to come to the marking session **with your exercise completed**. We will not provide any support during marking sessions.

4. You should be able to explain what you have done clearly, to show that you understand the concepts introduced.

5. Checks for plagiarism and collusion will be carried out on all work (you are advised to see the university student handbook if you're not sure about the potential consequences of plagiarism).