CMPU4021 Distributed Systems

Assignment

Due Date: 21st of November 2024, 11:30am

This assignment represents the practical assessment for this subject, 30% of the overall mark.

Deliverables:

Source code, Design Document, Demo

Overview

For this assignment, you are required to design and implement a distributed system that is a simulator of an electronic **food market place.**

The market has two types of people: **buyers** and **sellers**. Each seller sells the following **goods** (i.e. **items**): flower, sugar, potato and oil; and has limited supply of it. Each buyer in the **market** is looking to buy these goods.

Each seller can sell one of the goods (hereafter referred as **items**) at a time. Each seller starts with N amount of an item to sell (e.g., 5 kilos of flower). After selling all N amount of an item, the seller picks another item to sell. The seller can also start selling anther item after the maximum time for the item sale has elapsed (e.g. 60 seconds).

Each buyer decides to buy an item and tries to purchase it.

Each seller should be able to accept multiple requests at the same time. This means that a seller should be able to process multiple concurrent buy requests and decrement the amount/number of items in stock using synchronization.

Command line interfaces are sufficient. There is no need to provide a graphical user interface, but you are allowed to provide it, if you wish.

There must be at least **1** seller and multiple buyers (minimum **4** needed for the demo).
You may assume that each buyer wants to buy **1** piece/amount of an item. For example, 1kg of flower/item. You are also allowed to implement in such a way that a seller can buy 4kg, if they wish.

All buyers and sellers should be registered with unique ID, i.e. *nodeID*.

Seller Requirements

The seller should:

- Receive connections from multiple buyers (or connect with multiple sellers/buyers). This means you need to support concurrent multiple connections.
- Notify buyers which item is currently for sale and the current amount left.
- When a new purchase request is received from a buyer and if it is successful, the purchase confirmation is sent to the successful buyer, i.e. it should display/print a message: '***nodeID* bought '*an amount*' of *productID* from *nodeID***"
  All buyers should also be notified as how many pieces (or amount) of the item is left or if the item is sold out.
- As appropriate, buyers should also be notified about the time left for selling – if any amount of goods is left.
- Specify the selling period for an item. Max allowed is 60 seconds. If the maximum selling period elapses and not all pieces/amount of the item is sold, the next item is put for sale.
- Any item not sold out, should be put for sale again (automatically).

Buyer Requirements

The buyer should be able to do the following:

- Join market
- Leave market
- Try to buy an item
- List items available for sale

Buyer should:
- Connects to the seller (or find an item seller).
- Enter the amount of item to buy. The amount entered should be smaller or the same than the number of items left. If the purchase was successful, the buyer receives the purchase confirmation.

Implementation Restrictions

The assignment can be implemented in Java or Python. You can choose to use either **sockets**, **RMI** or **multicast**. You should use threads where necessary. You can choose an appropriate design for your systems and implements it, for example as a client/server, peer-to-peer or some other model.

Basic Java JDK and Python installations should be used. No additional libraries/packages/frameworks/containers should be used to support. If data storage is necessary, **files** should be used.
A configuration file can be used for connections setup and discovery, containing seller/buyers integer ID, addresses and ports.

Design document

You must provide **a design document** describing:

1. Your application architecture description and diagrams.
2. Setup, code directory structure.
3. Detailed instructions how to run the system on Windows platform, that it can be demo-ed in the lab (i.e. include a content of a Readme file). Containers should NOT be used (or necessary) to run/demo the assignment.
4. If you wish, you may include screen shots of a working application (only as an info, not as a proof of working correctly).
5. Your design document should also include 'Alternative design solution' section. 'Alternative design solution' section should contain an alternative/different system design description (if there was no stated assignment implementation restrictions). It should be a description of your second viable architectural design, with the difference that it was not implemented.
   Your alternative design should include:
   - A brief overview of components and technologies used.
   - How the components communicate and the placement of components.
   - System/software architecture diagram.

The document should **not exceed 10 pages**, and must also include the following:

- Your **name** and **student number**.
- The subject title, i.e. **Distributed Systems**
- The assignment title i.e. **Assignment**
- The date, i.e. **21st of November 2024**
- The following declaration**: "I declare that this work, which is submitted as part of my coursework, is entirely my own, except where clearly and explicitly stated."**

What should you submit?

1. The **design document**. It MUST be submitted as **ONE** word/pdf document.
2. **Code** (all necessary files, including a Readme file that provides details of how to deploy and run your code).
3. Put all the code and the design document in a directory named **YOURNAME (i.e. John Smith), ZIP it** and upload it in Brightspace by **11:30am, 21st of November 2024.**

Assessment Criteria

The application **must** be demo-ed in the labs. The demos will be scheduled.

Failure to provide code, documentation or demo will result in a mark of 0%.

You will be penalised an absolute value of 10% for every day that your assignment is late.

Marking Scheme

- System implementation (55%)
- General coding (error handling, comments, code structure) (10%)
- Design document (15%)
- Quality of the outcome (10%)
- Setup/Demo (10%)