Distributed Systems

Assignment

John Lennon

C10321265

DT228/4

Date: 21st November 2013

I declare that this work, which is submitted as part of my coursework, is entirely my own, except where clearly and explicitly stated.

Signed:

____

Introduction:

The following is a documentation report on my implementation of a distributed application that is a simulation of an online auction client and server. This report will cover the specifications for both the client and the server. It will also include the documentation and design of each class used for in the application. This implementation allows for multiple clients to connect to the server and for the user to bid for items on sale. This application will be implemented using Java sockets and Java mulit-threading.

When creating the client there is some specifications it must fulfill:

- The client must be able to connect to the server. The item currently being offered for sale and the current bid are displayed.
- A user is able to enter a bid. The amount entered should be greater than the current highest bid.
- After a new bid is placed, the new highest bid amount is displayed on the client's window.

For the server there is also a list of specifications:

- Can receive connections from multiple clients.
- After a client connects, notify the client which item is currently on sale and the highest bid (or reserve price).
- Specify the bid period. Max allowed 1 minute. When a new bid is raised, the timer is reset back.
- When a new bid is placed, all clients are notified. Clients should be notified about the time left for bidding.
- If the bid period elapses without a new bid, then the auction closes. The successful bidder (if any) is chosen and all clients are notified.
- When one auction finishes, another should start.

To start the server for this application the following commands must be entered in the command prompt window or execute the AuctionServer.bat file :

Java –classpath . AuctionServer port e.g. Java –classpath . AuctionServer 4421

To start the client the following commands must be entered or the AuctionClient.bat file can be used also:

Java –classpath . AuctionClient localhost port name e.g. Java –classpath . AuctionClient localhost John

Design Documentation:

AuctionServer:

This Java file is responsible for the majority of the functionality of the auction application. The server is set up first by binding a socket to the port number entered by the user. The server then calls the start function which assigns a thread to the server. Once the server has started, the user then enters in the name of the item for auction and the price for it. While this is being entered the server is listening out for any clients that are waiting to connect and accepts them providing the limit for the number of clients hasn't been reached. The startAuction function is where the code for entering the item details can be found. While the bidding is in process, the users are updated about the time left to bid by the timerBroadCast function. For communicating to the clients the broadcast function handles this detail. The user is told if their bid was successful or not. The winning user is told they won the bidding at the end of the auction from the winnerBroadCast function. As a client connects to the server they are added to the AuctionServerThread which handles each client. When a client is leaving the application, this is executed in the remove function. The client thread is removed and the connection to the client is closed off. In the main function, a new instance of the auction server is initialized and also the countdown timer for the bidding time is initialized and started.

AuctionClient:

The auction client file is responsible for handling all of the client based code. The client first connects to the server on the specified port and then welcomes the user to the auction. The user can then enter their chosen bid amount for the item that is currently on sale. The user will have a time limit of one minute for each item to bid on. If they are successful in their bidding the server informs the user of their winning bid and also of their winning item at the end of the auction. If a user wishes to leave they must type .bye into the command window. Once this is entered the handle function calls the stop function in the client class. This then closes the console for the user input, the output stream for data and the socket for the connection to the server.

AuctionServerThread and AuctionClientThread:

These classes are responsible for handling the threads created for the server and also for each client when they connect to the server. Both classes allow for threads to be set up for their use and handle for the closing and stopping of each thread. The server sets up input and output streams which handles data going to and from the server to client. The client thread handles similar functionality but is mainly focused on listening in on the correct port and getting or closing the input stream correctly.

TimerThread:

This class is responsible for the timer for during the bidding process. The timer runs while it is greater than zero and outputs the time to the server. At the thirty second mark the time is outputted to the user as a warning for the time left in the auction. The value bidTime is decremented every 1000 milliseconds or every second. When the timer reaches 0 the auction ends and the winner is informed that they have won. The auction then restarts and the user can enter in a new item. If a user makes a successful bid higher than the item price or current bid, the bid timer is reset to 60 seconds.

Problems Encountered:

Over the course of this assignment I encountered a few problems that challenged my learning skills and knowledge. One problem I encountered was setting the timer up so that it would restart when a successful bid was made. After careful consideration and thinking about what was trying to become, a solution to the problem eventually came to.

Another problem I had to overcome during this assignment was the use of multi-threads in Java. The concept of using threads and how to implement them was an area I wasn’t too familiar with. This challenge was eventually overcome after learning about threads from resources provided and from spending time to understand this concept.

For testing and also design purposes, after running the AuctionServer batch file the user can also run the AuctionClients batch file. This will create three clients and connect them to the server for the auction. This helps to save time opening multiple command windows and entering in the necessary information to connect to the server.