WESTERN UNIVERSITY
FACULTY OF ENGINEERING
DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

SE 3314B – COMPUTER NETWORKS APPLICATIONS

Assignment 1: Online Auction

Due Date: April 7th , 2014

This assignment is to be done individually. Do not show your code to any other student and do not look at any other student's code. Do not put your code in any public domain. However, you may get all the help you need from the TAs or the instructor. Also you are encouraged to discuss your problems (if any) with other students.

Objectives

The goal of this assignment is to build a client/server application that takes benefit from the power of single-threaded programming approach. The application in this assignment examines how you would go about constructing an online auction a similar vein to eBay or online stock markets.

In this assignment, you need to come up with an Online Auction Protocol (OAP) that uses TCP and UDP. Node.js modules, C# and .NET framework libraries will be your tools to build your protocol and application. You need to implement the protocol, and develop a client and a server that communicate using your protocol through TCP and UDP sockets.

Your online auction protocol (OAP) should be able to facilitate the following functionalities:

1. A buyer can join the auction freely at any time.
2. A buyer can leave the auction with no restrictions.
3. A buyer may bid to one or more items advertised by the server (online store).
4. The server shall send all items on sale to the client (buyer) once joined the auction.
5. The server shall update the price of an item as soon as anybody bids a valid price for that item.
6. The server shall identify the status of the bidders and communicate with them accordingly.

Installing Node.js

The first thing to do before you begin is install Node.js, if you haven't done so already. Node can be downloaded from the project home page: http://nodejs.org/ where you will be presented with a package installer for your operating system. Binary installers are available for Unix, Mac, and Windows. The stable release version of Node at the time of this assignment is v0.10.25.

Node provides a JavaScript runtime environment, which you can access at any time by going into your command prompt or terminal window and typing node. For the purposes of the assignment you will be running your main program (JavaScript source file) rather than typing code directly into Node. To run a JavaScript file, you run the node command with a parameter like this: **node filename.js**.

Overview & Requirements

Your application includes two main components: an online store and the clients. The clients may appear at any time, but the online store is a long-lived 'server' component. Buyers (clients) can join the auction, leave the auction, and bid for an item in the auction. The current price of an item gets updated as soon as anybody bids a valid price for the item. This is similar to the online stock market where the current prices of the shares are flashed as soon as their value changes. Let us look at some of the details.

Server Side

You are required to develop a Node.js based online store server that can handle each client request individually and at the same time it services many (hundreds or thousands) of buyers (clients).

This means, as each client connects to the server port and host, the main program will fire a pre-defined **Callback** function asynchronously. Because the **server.listen()** method creates a new **Socket** object for the connection, the original Socket object will be free to accept more client connections.

Your software architecture should follow the OO approach as possible. The server should support at least the following methods:

- join auction - to connect to the server and join to the current auction
- leave auction - the client is no longer in the system
- bid on an item - a client offers a new (higher) price for an item

Server can read the initial list of items, minimum prices and closing time from a text file (given). When a buyer joins the auction, the server needs to send the items information (Item description, Price, closing time) through TCP connection (using Node.js Net module) and the item image through UDP connection (using Node.js UDP/Datagram module). For example: (we are going to consider 4 and only 4 different items)

Item Description:	Toshiba, Laptop
Current bid:	$70.00
Closing time:	7:00 pm 17 February 2014

Server should respond to client operations by one of the following messages:

- **Enter a higher value and click Bid.**
  when a client has joined the auction.

- **Congratulations, the item is yours.**
  when auction time is expired and the client were the higher bidder.

- **You are currently the high bidder.**
  when a client has bid with the highest during the auction life time.

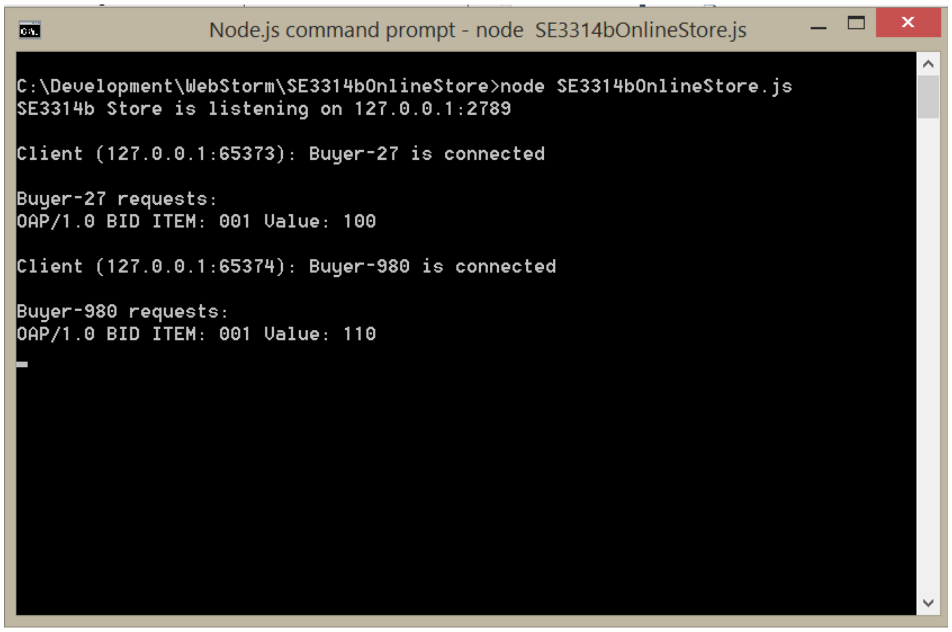
- **You are out bid, enter higher value and re-bid again.**
  when a client has lower bid during the auction life time.

- **You have lost.**
  when auction time is expired and the client were not the higher bidder.

To make your life easier, the server will be a console based application and consists of the following modules.

Console Interface

Node.js command console will be an acceptable UI for your application in this assignment. Your application will display the core operations of the server into sequential and scrolling up form of interaction. Figure 1 shows a sample look of your server interface.



```
C:\Development\WebStorm\SE3314bOnlineStore>node SE3314bOnlineStore.js
SE3314b Store is listening on 127.0.0.1:2789

Client (127.0.0.1:65373): Buyer-27 is connected

Buyer-27 requests:
OAP/1.0 BID ITEM: 001 Value: 100

Client (127.0.0.1:65374): Buyer-980 is connected

Buyer-980 requests:
OAP/1.0 BID ITEM: 001 Value: 110
```

Figure 1:   Sample look of your server interface

Image Manipulation Module

The required server functionality for this module is to maintain the UDP connection, read the items images from the server's local storage and send them to the connected clients through UDP packets. Please name this module as "ImageIO.js".

OAP Module

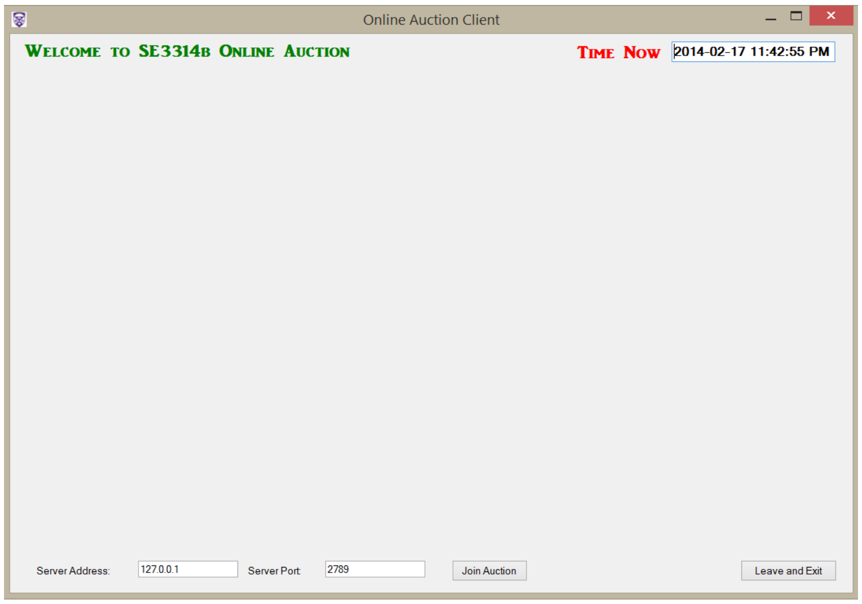
This module requires the Net module to provide all the functionality to create the TCP socket, listen to TCP connections, receive Online Auction Protocol (OAP) commands, and send the corresponding OAP responses. Please name this module as "OAP.js".

Main Application

The main application of the server, it requires the above modules (ImageIO.js, and OAP.js), in order to function. The basic functionality of this component is to implement the above modules (and other of course like timers, fs, util, etc.) to accept connections, receive client command and respond accordingly. The component is also responsible to display (log) into the console all information about the connected/disconnected clients, client's requests and their current bidding. (See Figure 1). Please name this module as "SE3314bOnlineStore.js".

Client Side

Then, you need to create a client as a windows form application, see Figure 2. Your form should have four equal parts to display four items when client joins the auction. This form should have two fields to accept port and IP address for server application.



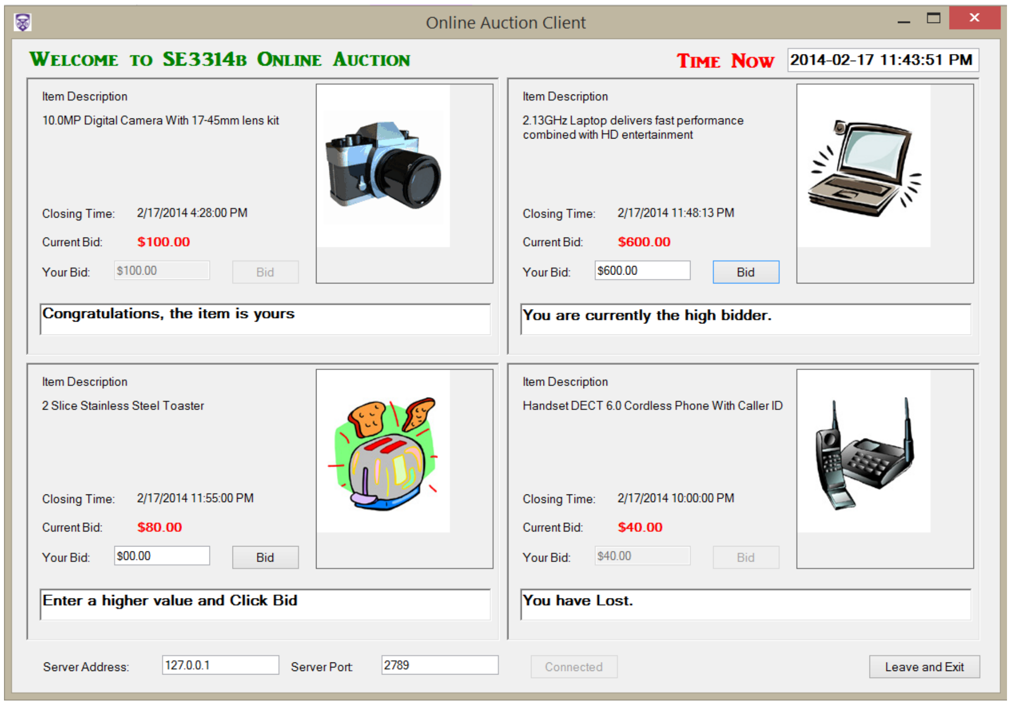
Time Now: 2014-02-17 11:42:55 PM

Server Address: 127.0.0.1

Server Port: 2789

Figure 2: The client form when run your client application and before joining the server

When the client successfully joins the auction, your application should display the images of items for sale along with their description, current price and closing time, and lets the user place a bid for a particular item. Figure 3 shows the minimum requirements for the client application.


Online Auction Client

WELCOME TO SE3314B ONLINE AUCTION

TIME NOW 2014-02-17 11:43:51 PM

Item Description
10.0MP Digital Camera With 17-45mm lens kit
Closing Time: 2/17/2014 4:28:00 PM
Current Bid: $100.00
Your Bid: $100.00
Congratulations, the item is yours

Item Description
2.13GHz Laptop delivers fast performance combined with HD entertainment
Closing Time: 2/17/2014 11:48:13 PM
Current Bid: $600.00
Your Bid: $600.00
You are currently the high bidder.

Item Description
2 Slice Stainless Steel Toaster
Closing Time: 2/17/2014 11:55:00 PM
Current Bid: $80.00
Your Bid: $00.00
Enter a higher value and Click Bid

Item Description
Handset DECT 6.0 Cordless Phone With Caller ID
Closing Time: 2/17/2014 10:00:00 PM
Current Bid: $40.00
Your Bid: $40.00
You have Lost.

Server Address: 127.0.0.1
Server Port: 2789

Figure 3: The client form after joining the server

Since multiple clients will be bidding at the same time, the methods on the server must be properly synchronized. Also client ID will be generated automatically once a client joined. Whenever any client makes a successful bid, the current price at all the clients must reflect this change, and the related messages need to be modified accordingly. Keep in mind that, all the exceptions must be properly handled.

Hand In

Submit the final version of your source code (Fully commented and formatted) through Sakai by the due date mentioned above. This will include a compressed achieve file (zip file) for all JS files and the package.json file to be named assignment1_yourUWOID.