Confidential – Not for Distribution

2010 McGill Code Jam

Proposed Problem

Table of Contents

2010 McGill Code Jam Proposed Problem
Author: John Lobaugh
Date: 30-Sep-10

1. Introduction

You are part of a team of software developers at a large multinational financial institution called *Jay Stanwell*. *Jay Stanwell* has been chosen to run the IPO (Initial Public Offering) of a Web 2.0 company called *Search.NET*. The IPO is unusual because it will be done via a Dutch auction (also called OpenIPO).

IPO: An IPO is when a company makes an issue of common stock or shares that are to be traded on a public exchange. In the IPO the company decides to offer a fixed number of shares at a certain price per share. Investors are given the opportunity to purchase these shares at this predetermined price. After the IPO has occurred the owners of the shares are then free to trade them in a stock exchange where the price of the share will be determined by the ebb and flow of market forces.

Dutch auction: Contrary to the usual procedure in an IPO the purchase price of the shares in a Dutch auction IPO is determined by a sealed bid process. An investor who wishes to purchase shares in the IPO places a bid for a fixed amount of shares at a certain price. The bidders in the auction have no knowledge of each other's bids. Bids take place over a fixed length of time (usually several weeks). Once the bidding phase is complete, the price of the share is determined by tallying all the bids, ranking them from highest to lowest, then choosing the highest price (i.e. bid) that will ensure that all of the company's shares get sold. This share price, called the "clearing" price, is the price that the company (Search.NET) must offer to the investors whose bids won (i.e. those investors that made a bid greater than or equal to the clearing price).

Let's look at an example.

Suppose company X is making an IPO of 1,000 shares. It has a Dutch auction to determine the share price at which winning bidders will be able to purchase shares in the company. It collects bids from interested parties over a week. At the end of the week the bidding is closed. The bids are tallied up as follows:

Bid Price	Number of Shares Bid
$10	252
$11	301
$12	350
$13	405
$14	325
$15	250
$16	150

Starting at the highest price we keep a running total of the number of shares that Company X would be able to sell at each bidding price.

Price	Total
$16	150
$15	400
$14	725
$13	1130

At a share price of $13, Company X would be able to sell all of its 1000 shares, so the "clearing" price is, therefore, $13. Any individual who bid $13 or higher has a winning bid and must purchase the number of shares they bid. Note that for all winning bidders the shares are purchased at the clearing price ($13 per share). So if I bid for 45 shares at $16 I will be able purchase my 45 shares at $13.

Now what about the total of 1130 shares? That's not exactly 1000 shares. To account for this, all the winning bids are prorated by 1000/1130. So if I bid for 45 shares then in the end I only get to purchase 45 * 1000/1130 ≈ 40 shares (at $13). In fact, the company can sell its shares at lower than the clearing price if they choose, in which case all the winning bids are prorated by the total number of shares calculated at the lower price. In this example, if the company decided to sell at $12, there would be a total of 1480 shares. All the winning bids would be prorated by 1000/1480. So at $12 a share, I would be offered 30 shares.

2. Project

The team will code an application that will have the following business goals:

Goals

- Collect bids from multiple clients simultaneously over TCP/IP using a standard protocol.

- Display cumulative totals of bidding data in real time.

- The display should allow for drill down and detailed inspection of individual bids.

- Bidding is closed by a special “bid close” message to the server from a client.

- Once bidding is closed...
  - Determine and display the calculated “clearing” price
  - Present a summary of the bidding data
  - Identify all individual bids that are greater or equal to the clearing price.

The underlying challenge in this competition is to create an application that meets the following goals:

- Scalable, reliable and highly available.
- Displays the data in real time as it enters the application.
- Analyzes and summarizes the data after close of bidding.

You are free to use whatever languages, libraries, and frameworks that you may find useful for the task. Definitions of the terms scalable, reliable and highly available follow.

Scalable

The application should be able to scale to accept as many possible simultaneous connections from bidding clients.

Reliable

Bids should be collected reliably (the definition of reliable will be further explained below). The successfully transmitted bids that are collected by the application will be verified at the end of competition. The definition of a successfully transmitted bid will be clarified below.

Highly Available

The application should exhibit minimal numbers of dropped connections and transmission failures even at the highest loads.

3. Architecture and Specifications

Application

Data Collection

The application collects the bid submissions from clients. It stores this data internally; whether this is done in memory or in a more persistent form is up to the team.

Requirements

- The application must be able to service multiple client requests simultaneously.
- Bids are validated. The share prices must be in a predefined range. Valid share price range is given below.
- A valid bid must be tracked and stored.
- Bidding is closed by a special client transmission.
- Bid submissions should be rejected by the server application once bidding is closed.
- Once bid submission is finished, the program should calculate the clearing price.
- The program should present the collected bid data in a summarized form. The summarized form will be used as verification by the judges to determine the correctness of the data collected.

Definition of a Bid

> A bid consists of three pieces of information: (1) a bid price, (2) number of shares bid for and (3) a bidder ID that uniquely identifies the bidder. (This can be a simple string like “John” or something more complicated like “A003ABC”.)
>
> More than one bid can be submitted by a single bidder. E.g the server MUST accept more than one bid from a single unique bidder id.

Client Transmission Protocol

Bids are collected over TCP/IP. The application should listen on a predetermined port for client connections. A simple protocol (described below) is used between the clients and server. In addition, the close of bidding is initiated by a special client transmission.

Flow of events

The protocol is based on the Transmission Control Protocol, using TCP port 8211 decimal. The bid client opens a TCP connection to the server host on the port 8211. The server application becomes available on the connection to process the request. The client sends the server a one line statement based on the transmission protocol specification (given below), and waits for the server to respond. The server receives and processes the client transmission, returns an answer, then initiates the close of the connection. When the client receives the answer and the close signal, it closes its end of the connection.

Data format

Any data transferred MUST be in ASCII format with lines ending in CRLF (ASCII 13 followed by ASCII 10).

Client transmission submission

The server host MUST accept the entire client transmission specification.

The client transmission is defined in EBNF notation as

```
client transmission = (
                        ( “B”, ‘|’, bid submission ) |
                        ( “C”, ‘|’, close bid ) |
                        ( “S”, ‘|’  bid summary )
                      ) , crlf ;


crlf                = “\r” , “\n” ;
```

There are three types of client transmissions to the server. The type of client transmission is identified by the first character in the transmission (B, C or S). The types of client transmission are:

1. Client transmission prefix: B. Bid submission.
2. Client transmission prefix: C. Close bidding.
3. Client transmission prefix: S. Summary generation. A special request to generate a bid summary for judging purposes.

Specifications for each type of the transmission are given below.

Bid transmission specification

The `bid submission` is given by

```
bid submission   = number of shares, ‘|’,
                   price, ‘|’,
                   bidder id;


(* maximum digits in number of shares is 8 *)

number of shares = digit, {digit};

(* maximum digits in price is 8 *)

price            = digit, {digit};

(* maximum size of bidder id is 32 *)

bidder id        =  { white space }, alphanumeric
                    { alphanumeric | white space },
                    { white space } ;


digit            = "0" | "1" | "2" | "3" | "4" |
                   "5" | "6" | "7" | "8" | "9" ;

alphanumeric     =  ( digit | alpha character )
                    { digit | alpha character }
```

```
alpha character  = "A" | "B" | "C" | "D" | "E" |
                   "F" | "G" | "H" | "I" | "J" |
                   "K" | "L" | "M" | "N" | "O" |
                       "P" | "Q" | "R" | "S" | "T" |
                       "U" | "V" | "W" | "X" | "Y" |
                       "Z" |
                       "a" | "b" | "c" | "d" | "e" |
                       "f" | "g" | "h" | "i" | "j" |
                       "k" | "l" | "m" | "n" | "o" |
                       "p" | "q" | "r" | "s" | "t" |
                       "u" | "v" | "w" | "x" | "y" |
                       "z"
```

```
white space = ? US-ASCII character 32 ?;
```

number of shares is a variable length field containing only ASCII digits (0-9). It specifies the number of shares bid for. The maximum number of shares that can be bid for in a single bid is 10,000. The maximum length of this field is 8 characters.

price is a variable length field containing only ASCII digits (0-9). It specifies the bid price. The bid price must fall within the bid price range that has been chosen for the IPO. The maximum length of this field is 8 characters.

bidder id is a variable length field containing only ASCII alphanumeric characters and the space character. An alphanumeric character is defined to be a character in the ranges 0-9, A-Z and a-z. The space character is defined as the ASCII character 32 decimal. Any spaces that preceed the first alphanumeric character of the bidder id should be ignored. Any spaces trailing the last alphanumeric character should be ignored. Maximum length of the field is 32 characters.

crlf is a carriage return followed by a line feed character (\r\n).

Note the pipe character that separates the submission fields.

Note that zero-filling of price and share numbers is permitted.

Examples of valid bid submission client transmissions are

```
B|1000|12|John Lobaugh\r\n
```

```
B|00100|012| John Lobaugh \r\n
```

The bid submission by the client is answered by the server response. The response in EBNF notation is defined as

```
response     = bid response, crlf;

bid response = “A” | “E” | “C”;

crlf         = “\r”, “\n”;
```

The bid response is one of three ASCII characters: A, E, or C.

- A indicates the server has accepted the bid.
- E indicates an error in the submission. When an E code is returned by the server the bid submission data is discarded and not tracked. An "E" is returned when the submission fails validation or is incomplete:
  - A field contains an unpermitted character.
  - The bid price is outside the range specified for the auction.
  - The pipe character is not present.
  - The crlf is not present at the end of the bid submission.
- C indicates that bidding is closed. Once bidding is closed the C code is the **only** response returned by the server to any bid submission (valid or invalid).

The server application is free to discard bid submissions for which it transmits a "E" or "C" code to the client.

After server transmission of the response the server closes the connection. The client will then close the connection at close signal sent by the server.

Close of Bidding Specification

Close of bidding is done by sending a special client transmission to the bid server. The `close bid` specification is given by

```
close bid         = ‘TERMINATE’;
```

To make it more clear the client would transmit

```
C|TERMINATE\r\n
```

to close the bidding process.

The close bid transmission by the client is answered by the server by a response. The response in EBNF notation is defined as

```
response     = close response, crlf;
```

```
     close response = “A” ;

     crlf           = “\r”, “\n”;
```

The close bid response indicates that the bidding has been closed. The server signals this by sending an A ASCII character. If multiple close bid transmissions are sent to the server each transmission receives the same response, i.e.

```
A\r\n
```

Bidding Summary Generation Specification

This client transmission is for judging purposes. When the server receives a summary generation transmission it should generate a bidding summary to STDOUT or to the screen in some other suitable display format. The summary should be in text format (no graphics). This is to allow the judges to view the summarized bidding data that has been collected by the server. The `bid summary` specification is given by

```
     bid summary = ‘SUMMARY’;
```

Therefore the client would transmit the following for a bidding summary generation transmission

```
S|SUMMARY\r\n
```

The server should respond with

```
     response     = summary response, crlf;

     summary response = “A” ;

     crlf           = “\r”, “\n”;
```

The summarized bidding data that is generated and placed on STDOUT or on the screen should be formatted as follows

```
Auction Status OPEN
Clearing Price $13
Bid Price Total
          Shares
$10       252
$11       301
$12       350
$13       405
$14       325
$15       250
$16       150
```

The actual numerical values shown above are for illustration purposes. The displayed summary data should reflect what has been collected by the application. The server can receive an S transmission at any point in the bidding, before or after close of bidding. If an S is received before sufficient shares have been bid to determine a clearing price (ie the total number of shares bid in the auction is less than the pool of shares up for auction) then the clearing price will be the lowest bid price received by the application. If the auction is currently open then the Auction Status should be OPEN, the auction has been closed then the Auction Status should be CLOSED

Competition Data

Server port for bid collection application	8211
Valid share price range for bids	30-100
Number of shares for Search.NET IPO	10,000,000

Please make these settings easily configurable in our application. The judges may modify them during the testing of your application.

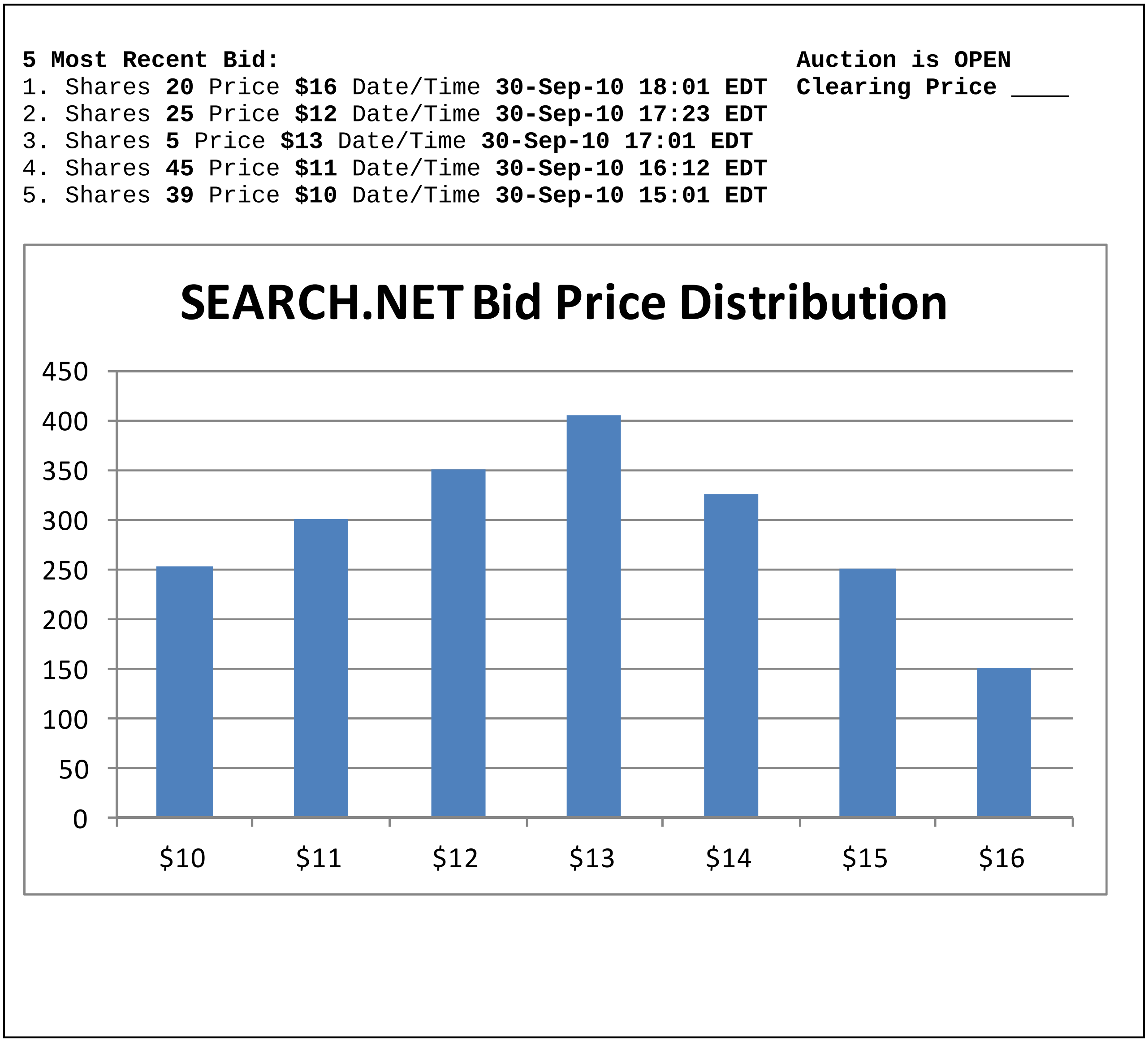
Data Display

The team will be required to build a GUI that displays collected bid data. The data should be displayed in real time. As bids come in from clients the visual interface should be dynamically updated.

Requirement:

- The data should be displayed and updated in real time on the screen.
- The 5 most recently arrived bids must be shown. As new bids come in this list should be updated
- The data should be displayed in summary format in a graphical manner. This summary should be continually updated as new bids arrive.
- The display of summary data should allow the user to drill down to see individual bids.

An example GUI of the bid summary data is shown below. This is only an example, teams are encouraged to explore alternative methods of displaying the data in summary format.



5 Most Recent Bid:
1. Shares **20** Price **$16** Date/Time **30-Sep-10 18:01 EDT**
2. Shares **25** Price **$12** Date/Time **30-Sep-10 17:23 EDT**
3. Shares **5** Price **$13** Date/Time **30-Sep-10 17:01 EDT**
4. Shares **45** Price **$11** Date/Time **30-Sep-10 16:12 EDT**
5. Shares **39** Price **$10** Date/Time **30-Sep-10 15:01 EDT**

Auction is OPEN
Clearing Price ____

In this example the data is displayed in a bar chart format. The bids are “binned” by share price. Each bar indicates the total number of shares bid at each price.

The user should be able drill down and see the individual bids that make up a bin. This could be done in the example above by clicking on a bar in the graph. Doing this action would open another screen that displays the individual bids that make up the bin. The detail screen could display the bids in tabular format or some suitable graphical format.

Clearing Price Calculation and Summary Data

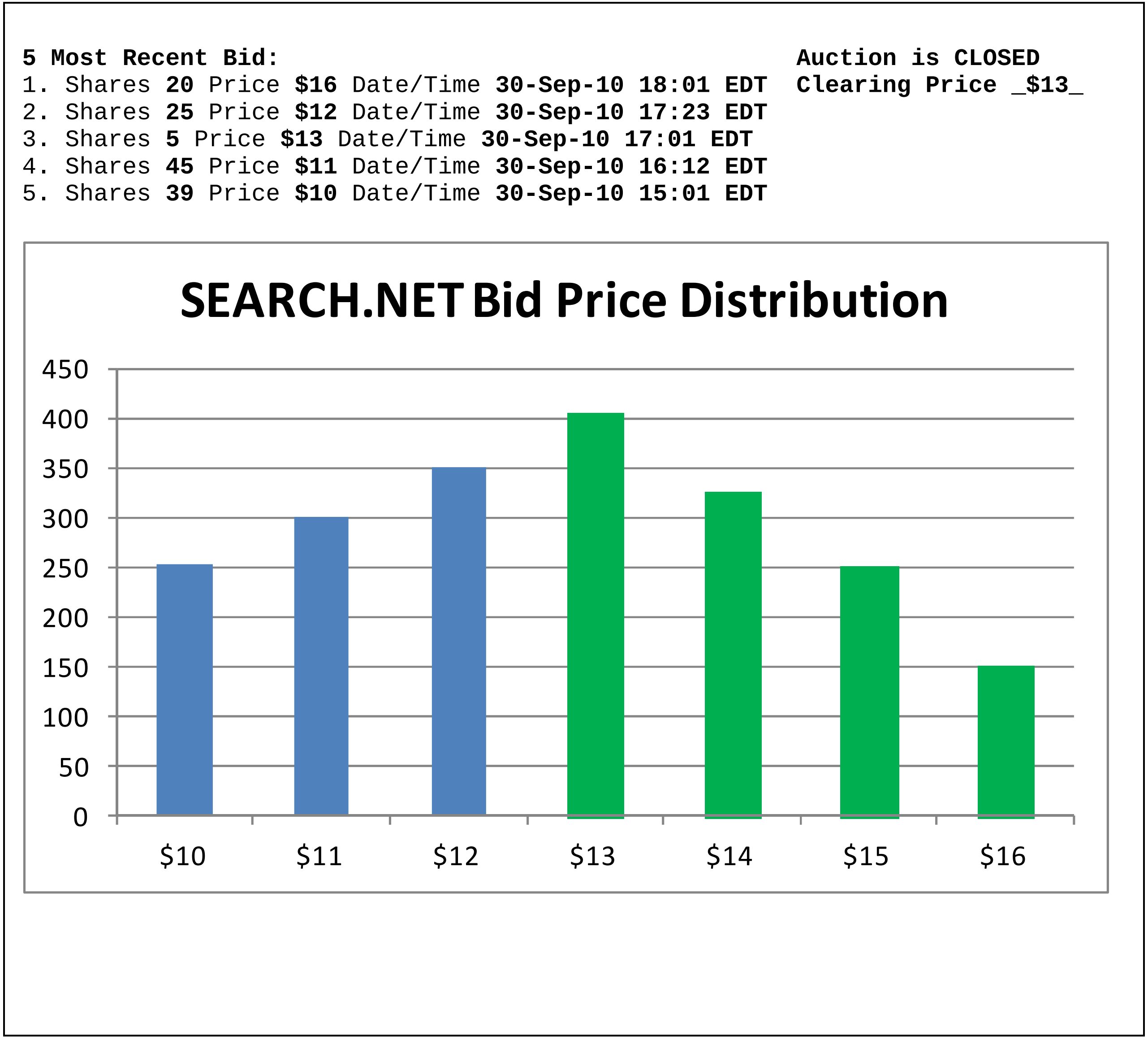
Requirement:

Once the auction is closed...

- After bidding is closed the clearing price is calculated and further client submissions are rejected. Any client transmissions made after close of bidding receive a “C” code in the server acknowledgement transmission.
- Summary data and the clearing price are calculated.

Summary data should be calculated to allow verification of the bids collected by the server.

Using the previous example of a GUI we can see that the auction status and clearing price are displayed in the upper right hand corner of the summary screen. The winning bids are colored in green in the chart.



```
5 Most Recent Bid:                                         Auction is CLOSED
1. Shares 20 Price $16 Date/Time 30-Sep-10 18:01 EDT  Clearing Price _$13_
2. Shares 25 Price $12 Date/Time 30-Sep-10 17:23 EDT
3. Shares 5 Price $13 Date/Time 30-Sep-10 17:01 EDT
4. Shares 45 Price $11 Date/Time 30-Sep-10 16:12 EDT
5. Shares 39 Price $10 Date/Time 30-Sep-10 15:01 EDT
```

The visual formatting of the summary data should allow for clear and unambiguous verification of the summary data by the judges.

4. Bid Submission Client

A bid submission client application, which can simulate one or more bidders, will be provided by the competition judges. This client application will be used to submit bids to the coding team's application. The client application submits bids to the application. The client can be configured to transmit multiple bids in parallel. The client bid data submitted from the client will come from external data files; you will be provided with sample data files that you can test your server application with. The judging team will use the same client application to test each team's application.

5. Competition Procedures

Reliability and Failures

Any software system will have failures. It's possible that at any time during the conversation between client and server a transient condition on the network, on the server machine, or on the client machine could cause the connection to be broken. Also, prior to the client / server connection the client may not be able to initiate a TCP/IP connection with the server (ie the client receives a connection refused error). If the server is unable to handle the load the connection between server and client could time out (ie a connection timeout error). In addition, the local network might not be able to handle the traffic load which could lead to connection time out errors or unsuccessful connection attempts.

Such conditions as those described above may occur in the competition. For judging purposes: a bid transmission where the server transmits an "A", "E" or "C" response to the client is considered to be successful bid transmission. In other words, the client application will consider a successful bid submission to be one where it receives:

- "A" response for a valid bid submission OR
- "E" response for a invalid bid submission
- "C" response when bidding has been closed.

Once an "A" has been received the server must track and include that bid submission in its bid data store. The bid must be included in the clearing price calculation and any summary data presented after bid close.

This implies that the client application will track any successful bid submission. All successful bid submissions will be reported and tracked by the **client** application in order to verify the server application results.

After the coding team's allotted time has elapsed, the application will be tested by the competition judges. This testing will be done using the same client application that was provided to the team during their coding phase.

During the testing phase of the competition one or more instances of the client application will be configured to send many simultaneous connections.

Teams will be judged on

- The number of successful bid transmissions. Obviously the higher number the better for the team.
- The number of failed bid transmissions. A failed bid transmission is defined as one where the client could not connect to the server application or did not receive an acknowledgement transmission after a preset amount of time. The lowest number of these is a deciding factor for the winning team.
- The accuracy of the data. The number of shares that were bid at each share price will be examined. The calculated clearing price will be examined. If this is not correct the team will be penalized.
- Validation failures. A number of the bids that will be transmitted will be invalid (i.e. the bids will fail the validation rules). The **client** application will track any invalid bids. The team will be penalized for any invalid bid which is accepted by the server (ie the server accepts the bid and transmits an "A" for a bid that is invalid).
- The team will be penalized if the application fails to transmit "C" after close of bid. Simply refusing any more connections after bid close would be acceptable as well but transmitting the "C" is better.
- The display of real-time data in the visual client application. The client application should be accurate and as close to real-time as possible.

Appendix A OS Considerations

The maximum number of inbound TCP/IP connections supported by Windows XP Professional is 10 (!). This will have a serious impact on anyone who is testing and implementing their application on Windows XP.

http://support.microsoft.com/kb/314882

The default range of ephemeral ports in Windows is set to 1025 to 5000. You may find it necessary to increase the number of ephemeral ports. Please see the link below for more information

http://msdn.microsoft.com/en-us/library/aa560610(BTS.20).aspx

You server application will be tested on a recent release of Windows Server OS and Linux.