Client Server Design for Market Data Dissemination:

High Level Design:

- There are two processes, client & server.

Server:

- Server sends UDP Packets at two ports 'a' then 'z', or 'z' then 'a', based on the input probability parameter.
- Outputs the total time taken in microseconds
- Outputs no. of packets processed by port 'a' prior to port 'z'

Client:

- Client has 2 threads, One listening to port 'a' & another listening to port 'z'
- The two threads synchronize with a Semaphore.

Thread1/2:

- Receives UDP packets from port 'a' or 'z'
- Checks if the received seqno_ of the UDP packet is already processed, then ignores it. (Stale Copy)
    - If not already processed, Checks if a Stock for Buy/Sell by such a name already exists in the Hash Table
        - If already exists, then inserts/modifies/deletes the receive UDP packet info into the correspoding Stock Buy/Sell Linked List
        - If does'nt exist, creates a new entry into the Hash Table by adding the head node of the newly created Linked List as value.
- Outputs the Limit Book into the log file provided for every 100 msg(UDP Packets) processed.

Data Structures:

The following key data structures have been used:

- A Hash Table 'stockmap' with,
  Key: StockName_B or StockName_S (Here 'B' indicates Buy Side, 'S' indicates Sell Side)
  Value: Limit Book implemented as Singly Linked List

- A Hash Table 'processed' with,
  Key: seqno
  Value: Boolean value (True if already Processed else False)

- Singly Linked List having: (Acts as a Limit Book)
  insert() at the level specified
  modify() at the level specified
  del() at the level specified
  display() all the nodes

- A Semaphore used for synchronization between two threads.

Other Data Structures that did not suit for the requirement:

- ***Using a Database, Attempted with MySQL:***

Instead of having Hash Tables & Linked Lists, using a Database might seem to solve a lot of difficulty. The First Design was attempted by connecting to a database & there after (insert/modify/delet)ing based on the packet msg state.

But using a database lead to a lot of packet loss.

- ***Using a Doubly Linked List instead of Singly Linked List in the above described design:***

The advantages of using DLL are:

- We can traverse through either of the ends(head or tail) based on from where the level(position) of the target node for insert/modify/delete is nearest to.
- The head maintains a count value of the total no. of nodes & let's to travel via the shortest distance.
- This improves the performance drastically.

Difficulties while implementing with DLL:

- What if the 'level' of the UDP packet received to be inserted/modified is greater than the count of nodes maintained in the head?

A lot of packets got dropped by DLL DS due to this reason.

Disadvantage of Current Implementaion:

The obvious disadvantage is we are using Singly Linked List & hence cosumes O(n) for insert/delete/modify, whereas a DLL could have consumed O(log n)

ASSUMPTIONS:

- It is assumed that there will be a single entry in the ConfigFile passed to server or client as argument.
- If the requirement is to have multiple entries, having braces for while covering the entire code would do the job.