Primarily, the project has 3 major components:

1. Exchange B Market Data Feed Server (MarketDataFeedServer.java)
2. Micro-price service collocated at Exchange-B working as Market Data Handler receiving Market Data prices from the feed server(MicropriceMDHandler.java)
3. Firm's TCP Server collocated at Exchange A receives micro-price data (ExchangeA_TCPServer.java)

Exchange B's Market Data Feed server uses a mock/random price-engine aka 'pricer' to provide up to 10 levels of prices. In the real world, the price-engine would be a separate entity, perhaps a micro-service, providing price-objects to many different market data feed servers, each supporting different format e.g. FIX, FIXML, ITCH/OUCH or some proprietary formats etc. The source of the price data for the price-engine could be external sources or it could be home grown. Price engines typically use IPC or UDP connections if running on remote hosts. For this project, to keep things simple only blocking-queues are used to depict the same producer-consumer model. For this project, the pricer runs on a separate thread, managed via ScheduledExecutorService and generates and provides prices to MD Feed Server, which translates that to the required binary format and sends the data to the micro-price MD handler via UDP multicast.

Micro-price MD Handler (MicropriceMDHandler.java) collocated at Exchange B primarily does following:

1. Listens to incoming market Data multicast messages supporting up to 10-levels (tiers) of market data
2. Translates the incoming price data to an Order Book object.
3. Using the price data from the top of the book calculates micro-price
4. Via TCP client, sends the newly calculated micro-price along with the security-id to a service collocated at Exchange A.

TCP-Server collocated at Exchange-A receives security-id and micro-price and save stores as key-value pair in a Hash Map.

Note: Please note in a real production system not all securities have all levels/tiers of prices available. It is possible some instruments/securities do not have up to 10 levels of prices – therefore, this project is built to show support of different number of levels coming from the feed server max limit to 10.

Java Processes Execution:

There are shell scripts created to show how to run the processes – please use following order:

1. `exchangeA_TcpServer.sh – tcp server collocated at Exchange A`
2. `microprice_handler.sh – multicast price-layers receiver; tcp-client`
3. `md_feed_server.sh – multicast sender`

For Verification:

One can run all scripts with last argument providing the size of samples – all script needs to run with the same sample size e.g. Market Data Feed Server runs as following:

export BLVDR_HOME=/c/blvdr

export CP=.:$BLVDR_HOME/lib/micro-price-service-1.0.jar:$BLVDR_HOME/lib/agrona-1.14.0.jar

java -cp $CP com.microprice.MarketDataFeedServer 239.100.0.1 7445 NetworkInterface **100000**

If the last argument is provided, the feed-server captures the outgoing market data in a file, named: **ExchangeB_MDFeedServer.dat**, **Location: /tmp (or C:\tmp)** folder.

On the receiving end, MD Handler also saves all incoming market data messages in another file: **ExchangeB_MDFeedHandler.dat**

In my xterm, I can run diff or cmp as below to check if there are any differences in the two output files; difference would represent potential error conditions.

```
cmp ExchangeB_MDFeedServer.dat ExchangeB_MDFeedHandler.dat
diff ExchangeB_MDFeedServer.dat ExchangeB_MDFeedHandler.dat
```

Since MD Handler also sends the micro-price (& securityId) to a TCP server collocated at Exchange-A, it also captures that data in a separate file: **ExchangeB_TCPData.dat**

The TCP-Server at Exchange-A, also saves the incoming data in a file: **ExchangeA-TCPData.dat**

cmp, diff or similar utilities should show no difference in these two files.