## Appendix A: Redline NBBO, API, and Code Implementation

### Redline NBBO

We wrote customized code on the Redline server to record the tick data of the Redline NBBO with time accuracy in nanoseconds. We first describe the details of the implementation. We further consolidate the data and output in the format as shown in Appendix B. To be noted first is that there are two sets of similar data, one from SIP NBBO (SIP in the NBBO field) and the other from Redline NBBO (RL in the NBBO field). To make it easy to compare, we put them in the same context, and sort according to timestamp value. The exchange where the update occurs is also listed. The prices are updated due to several trigger events, including new bid / ask quotes, cancellation and trading activities. If an update is due to a new bid quote, which is higher than the status quo bid price, the last column has a value of 1002. If it is a new ask quote then the last column has a value of 1004. If a trade event triggered the update, the last column has a value of 1108.

As shown in Appendix B, the trade events do not have to match the NBBO bid / ask price. This is reasonable in that the trade can be an ISO trade or even a dark pool trade. Nevertheless, the types of trades we are interested in are regular trades, which should be executed on a NBBO price.

In Appendix B, there are approximately two sets of the same bid/ask/trade sequence, with slightly different timestamps. Since the timestamp records when the server receives the update, there is a delay compared with the real moment that it happens. The delay depends on the source that the update is collected.

If the source of the data is UTP, the update information is first generated in the corresponding exchange, is then sent to the UTP SIP data processor, gets calculated, and is then further sent to the trader’s server. The Redline server works as the trader’s end. But we still need to measure the time when the information is received by the trading application.

If the source of the data is Redline NBBO, there are two possible routes depending on the data feed of the Redline server. The server we work on has four feeds: exchange BATS, exchange EDGE, exchange NASDAQ, and UTP. The feeds from BATS, EDGE, and NASDAQ are direct feeds. UTP is the consolidated data processor that provides SIP NBBO. The Redline system consolidates these updates in real-time to generate a local NBBO. If an update is from BATS, EDGE, or NASDAQ and can make to the top of the book as NBBO, it can be updated in Redline NBBO immediately. In this case the date route is from the exchange to the Redline server, where it gets calculated and then an update is made to the system directly. If the update is from an exchange other than BATS, EDGE, and NASDAQ, the Redline system can only get the information through UTP. Then the route of the information is from the exchange to UTP, where it gets calculated, is sent to the Redline system, gets calculated again, and then an update is made to the system. As a result, the total delay is shorter on Redline if the update originates from BATS, EDGE, or NASDAQ and vice versa if the update originates from an exchange other than BATS, EDGE, or NASDAQ.

Appendix B illustrates these two types of latencies. Our example uses a numberof exchange PIDs. Some of them, including FH_PID_NYSE_ARCA, FH_PID_NSX, FH_PID_NASDAQ, are not directly connected with the Redline system. As a result, a SIP NBBO update has a timestamp earlier than that of the corresponding RL NBBO update. On the other hand, three PIDs, including FH_PID_BATS_BZX, FH_PID_BATS_BYX, FH_PID_EDGA, are directly fed into Redline system. Then a RL NBBO update has a timestamp earlier than that of the corresponding SIP NBBO update.

Appendix C shows the distribution of top of book exchanges on the two NBBOs. It only lists 6 major exchanges. According to SIP NBBO, AAPL has 2.99% of the best asks from BATS_BYX, about 8.54% from BATS_BZX. They add up to 13.53% in total from BATS. On the other hand, 20.2% of them are from EDGE, 16.66% from NASDAQ, 39.22% from NYSE ARCA. These numbers do not have to add up to 100% because there are other exchanges that are not shown.

One observation is that for the percentage number, the values are different in the two NBBO cases. This is related to different latencies in these two cases and the approach by which these numbers are calculated. Every time the NBBO ask/bid price changes, the corresponding exchange is recorded as the top of the book exchange. If later updates show up with a different exchange but the same top of book price, that exchange is ignored because we do not know whether it is other type updates, not a new ask/bid quote, that bring up the exchange. As a result, we may over-count BYX in the case of Redline NBBO. For example, if two asks with the same prices are quoted from NASDAQ first and then BYX 1 millisecond later, BYX will show up early on Redline NBBO and then be counted as top of book exchange. However in the case of SIP it should be NASDAQ.

Redline develops the C library to control the network connection, collect data and calculate NBBO. It also provides an API so that customers can build their own applications based on C or C++ language. They also provide a series of sample application to analyze data.

One function provided by the API can completely print out the received quote or trade update in the following format,

```
symbol: AAPL
  trigger: FH_TRIGGER_ASK
  updates included: 1
  status: FH_STATUS_BOOK_SUSPECT
  subscription flags: FH_REFRESH_FULL | FH_AGGREGATE_PRICE | FH_DEPTH_LIMITED
| FH_DELIVERY_CALLBACK
  feed: FH_FEED_UQDF pid: FH_PID_NYSE_ARCA sequence number: 22572032  line number:
0
  time: 17:25:23.080000000
  timestamp: 96078685308100
bids (1):
  level[01]:  numOrders:    0, volume:     100, price:   476.7000, time: 17:24:31.681000000
asks (1):
  level[01]:  numOrders:    0, volume:     400, price:   476.7600, time: 17:25:23.080000000
```

The key information includes the *symbol*, the *trigger* type, the *feed* (through which channel we obtain the data), the exchange (shown as *pid*) where the quote is initiated, the *time* that the exchange updates the information, the *timestamp* that the Redline server receives the update, the *bid* and *ask* prices, *volumes*, and the *time* they are updated. In the example above, it is a new ask quote, which happens at 17:25:23.08 in exchange NYSE ARCA. The moment it happens is recorded in both the main *time* field and the ask *time* field.

The data frequency is very high, with the lag between two updates as small as a few nanoseconds. In order to record the data arrival time correctly, the program needs to have a dedicated software thread to handle the information so that it processes the current data very quickly and becomes available for the next data. As a result, the CPU may be idle for more than 90% of the time. By dedicating computational resources to this we can reduce the possibility that a new update arrives at the moment that the CPU is still processing other jobs, so that it cannot be processed immediately. Since the data process includes obtaining the timestamp, it will cause the recorded update arrival time to be delayed.

We develop a multi-threaded code to deal with this. First, there is a thread #0 hidden behind the Redline API to communicate with the hardware such as the network interface card. It sends out call back function to thread #1, which saves the update information into a FIFO queue that temporarily stores information. Another thread (#2) then sequentially checks the FIFO queue. If there is new data then remove it from the FIFO queue and print out to IO devices. Furthermore, we apply specific technology so that threads #0 and #1 are assigned to fixed CPUs (The server is a multi-core machine with 24 CPUs). In this case thread #1 is the dedicated CPU that conducts the minimum amount of work, including receiving the update, getting the timestamp and saving it to the FIFO queue.

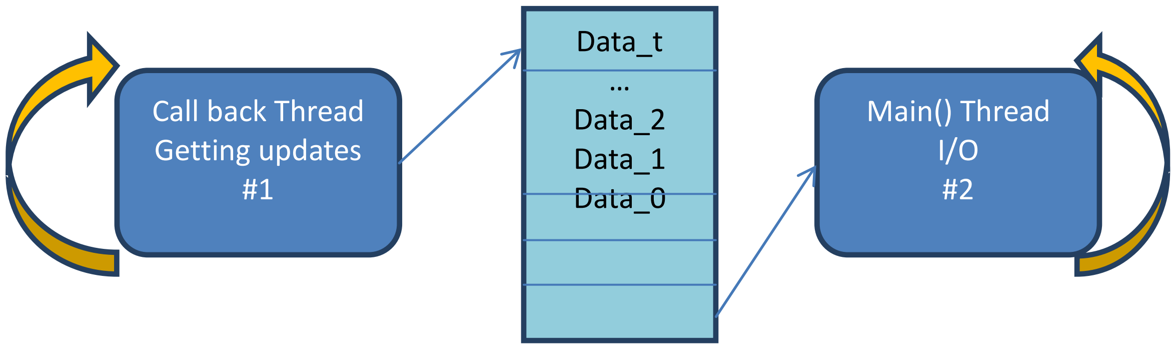

Fig 2. Multi-thread program diagram

Another issue to handle is the correctness and accuracy of the timestamp. As a fact, the clocks of different exchanges are not synchronized. So the exchange update time cannot be used as a guideline on when the update happens. On the other hand, we are more interested in the time that the trading system receives the updates. Thus we employ a software clock to record the information arrival time. Specifically, we use the *timestamp* field as shown in the example, which is available through the Redline API. Note that it is stored as a 64-bit integer value, which is actually the count of CPU ticks starting from system reboot time. This value can be converted into real clock up to the accuracy of one nanosecond.

We collect daily data for six symbols: AAPL, AMZN, SOHU, SQQQ, FSLR, DNDN. The size of daily data file ranges in 200-500 MegaBytes. We further wrote Perl and Unix Shell scripts to pre-process the data and transfer the data to a local Windows machine through ftp. The main data analysis procedure is coded in Matlab.

## Appendix B: Sample Data

| Symbol | Exchange PID | NBBO | Time stamp | Ask price | Ask size | Bid price | Bid size | Trade price | Trade size | Trigger |
|---|---|---|---|---|---|---|---|---|---|---|
| AAPL | FH_PID_NYSE_ARCA | SIP | 09:30:10.544572841 | 547.1 | 100 | 547.29 | 200 | 0 | 0 | 1004 |
| AAPL | FH_PID_NYSE_ARCA | RL | 09:30:10.544573019 | 547.1 | 100 | 547.29 | 200 | 0 | 0 | 1004 |
| AAPL | FH_PID_NYSE_ARCA | SIP | 09:30:10.618426062 | 547.1 | 100 | 547.31 | 300 | 0 | 0 | 1004 |
| AAPL | FH_PID_NYSE_ARCA | RL | 09:30:10.618426317 | 547.1 | 100 | 547.31 | 300 | 0 | 0 | 1004 |
| AAPL | FH_PID_NSX | SIP | 09:30:10.644072413 | 547.1 | 100 | 547.31 | 300 | 0 | 0 | 1000 |
| AAPL | FH_PID_NSX | RL | 09:30:10.644072613 | 547.1 | 100 | 547.31 | 300 | 0 | 0 | 1000 |
| AAPL | FH_PID_NASDAQ | RL | 09:30:10.667109263 | 547.1 | 100 | 547.29 | 400 | 0 | 0 | 1004 |
| AAPL | FH_PID_NYSE_ARCA | SIP | 09:30:10.699665334 | 547.1 | 100 | 547.29 | 200 | 0 | 0 | 1004 |
| AAPL | FH_PID_NYSE_ARCA | RL | 09:30:10.699665455 | 547.1 | 100 | 547.29 | 200 | 0 | 0 | 1004 |
| AAPL | FH_PID_BATS_BZX | RL | 09:30:10.699665993 | 547.1 | 100 | 547.28 | 300 | 0 | 0 | 1004 |
| AAPL | FH_PID_BATS_BZX | SIP | 09:30:10.699666122 | 547.1 | 100 | 547.28 | 300 | 0 | 0 | 1004 |
| AAPL | FH_PID_BATS_BYX | RL | 09:30:10.699666636 | 547.1 | 100 | 547.27 | 200 | 0 | 0 | 1004 |
| AAPL | FH_PID_BATS_BYX | SIP | 09:30:10.699666895 | 547.1 | 100 | 547.27 | 200 | 0 | 0 | 1004 |
| AAPL | FH_PID_EDGA | RL | 09:30:10.699667199 | 547.1 | 100 | 547.31 | 300 | 0 | 0 | 1004 |
| AAPL | FH_PID_EDGA | SIP | 09:30:10.699667601 | 547.1 | 100 | 547.31 | 300 | 0 | 0 | 1004 |
| AAPL | FH_PID_NYSE_ARCA | SIP | 09:30:10.701459264 | 547.1 | 100 | 547.29 | 200 | 0 | 0 | 1004 |
| AAPL | FH_PID_NYSE_ARCA | RL | 09:30:10.701459282 | 547.1 | 100 | 547.29 | 200 | 0 | 0 | 1004 |

* NBBO - SIP vs. Redline (RL)

* Triggers - 1002: TRIGGER_BID; 1004: TRIGGER_ASK; 1108: TRIGGER_TRADE;

## Appendix C: Percentage of time an exchange is on top of NBBO book

| Symbol | Feed &bid/ask | FH_PID_BATS_BYX | FH_PID_BATS_BZX | FH_PID_EDGX | FH_PID_EDGA | FH_PID_NASDAQ | FH_PID_NYSE_ARCA |
|---|---|---|---|---|---|---|---|
| AAPL | SIP ask | 2.99% | 8.54% | 16.39% | 3.81% | 16.66% | 39.22% |
| | SIP bid | 3.51% | 7.45% | 14.92% | 5.17% | 15.99% | 38.83% |
| | Redline ask | 5.21% | 8.63% | 16.29% | 3.34% | 14.74% | 39.08% |
| | Redline bid | 5.59% | 7.74% | 14.68% | 4.48% | 14.37% | 38.80% |
| GOOG | SIP ask | 2.97% | 8.72% | 3.13% | 5.91% | 26.44% | 18.63% |
| | SIP bid | 12.32% | 10.73% | 2.50% | 5.72% | 24.93% | 18.04% |
| | Redline ask | 4.70% | 8.65% | 2.95% | 4.88% | 25.38% | 18.61% |
| | Redline bid | 13.91% | 10.74% | 2.44% | 5.68% | 23.85% | 17.92% |
| ADRE | SIP ask | | 11.87% | 3.11% | | 7.82% | 75.13% |
| | SIP bid | | 0.48% | 0.03% | | 74.41% | 25.04% |
| | Redline ask | | 11.53% | 3.11% | | 8.83% | 75.13% |
| | Redline bid | | 0.48% | 0.03% | | 74.41% | 25.04% |
| KNDI | SIP ask | | 0.04% | 0.27% | | 61.25% | 38.41% |
| | SIP bid | | 11.17% | 18.60% | | 23.76% | 45.92% |
| | Redline ask | | 0.04% | 0.27% | | 61.25% | 38.41% |
| | Redline bid | | 11.17% | 18.60% | | 23.76% | 45.91% |
| QQQC | SIP ask | | | | | 99.71% | 0.28% |
| | SIP bid | | | | | 2.29% | 97.70% |
| | Redline ask | | | | | 99.66% | 0.28% |
| | Redline bid | | 2.29% | | | | 97.70% |
| QQQM | SIP ask | | 25.15% | | | 36.74% | 37.85% |
| | SIP bid | | 35.16% | | | 27.73% | 37.09% |
| | Redline ask | | 42.83% | | | 17.01% | 39.91% |
| | Redline bid | | 38.70% | | | 24.20% | 37.10% |