Data link system via Financial Information Exchange protocol using Simple Binary Encoding

Kashkynbek Islam*

Computer Science, Software Engineering and Telecommunication,
International Information Technology University,
Manas street #34A, Almaty050019, Republic of Kazakhstan
ikashkynbek@gmail.com

UmurzakovBulat

Assistant-professor, Department ofComputer Science, Software Engineering and Telecommunication,
International Information Technology University,
Manas street #34A, Almaty050019, Republic of Kazakhstan
bnu0907@gmail.com

Abstract

The main goal of this project is to make transactions faster between stock exchange and traders. Financial Information Exchange (FIX) is a dominant standard for data link between the participants of exchange trades in real time around the world, but binary protocols such as ProtocolBuffer, Simple Binary Encoding (SBE) have takenrevolution among high-programmers. To get a faster trading platform and comply with the international standards of the stock market,this project combines FIX with a new binary protocol SBE.

Keywords: Stock exchange, FIX, SBE

1. Definition of Problem and Protocols

Over the past few years the list of Stock Exchange investors has beenactively growing. As we know, trading on the stock market in the 21st century is an extremely high-tech process. For instance,for the investorstoget high returns, developed a variety of trading platforms, brokerage systems that can cope with a heavy load, implemented API to him, laid high-speed communication channels, etc. According to the rules of the stock exchanges only members of the market can send trading orders directly to the exchange, bypassing the broker system. However, direct access costs money that not everyone can afford, but everyone wants to make transactions with maximum speed. This is not surprising, because between success and failure, gain or loss in the stock market is often only a fraction of a second. Therefore, it should work like clockwork and very fast.

FIX protocol - data transfer protocol, an international standard for data exchange between the participants of exchange trades in real-time. It is now widely used trading system for the exchange of financial data and making transactions. FIX protocol is supported by most major banks and electronic trading systems, as well as the major exchanges.

SBE is OSI layer 6 presentation for encoding and decoding application messages in binary format for low-latency applications.[1] Message encoding/decoding is one of the costly things. Many applications spend a lot of time to parsing and transforming FIX orXML than executing business logic. SBE is designed to make this part of a system more faster. SBE follows a number of design principles to achieve this goal. Sometimes adhering to these design principles means features available in other codecs will not be offered.

2. Problem Solution

By following the international standards of the stock market this project decided to combine FIX with SBE to get faster trading platform. To do this server client application is written. Server side of the program gets market data from Stock Exchange by FIX API and propagates it to clients and store to local storage. Clients are connected with server using FIX engine. For connection between client and server libraries as Apache MINA Core and QuickFIX are used, but in structure of messages Simple Binary Encoding is used.

2.1. *Defining SBE Message.*

As specified by the standard, to use SBE first of all we need to define a XML based schema for the messages. SBE supports data types asint, floate, char, array, enum, composites,repeated groups, strings and blobs.

A message schema should be compiled to generate stubs in a round of programming languages like Java, C++, and C# using SBE compiler and following command:

```
java-jar sbe-tool.jar <sbe-message-file.xml>
```

2.2. *Programming with Stubs*

The generated stubs contain Market data incremental refresh message and the structure of message based on FIX Specification. When a listener of Stock Exchange API handles the message,server side of program should convert this message to SBE, encode it, then convert encoded binary data to HEX and propagate this hexadecimal message to client. Let’s consider encoding process:

First of all fill message header

```
header.wrap(encBuffer, index, 0)
        .blockLength(marketDataIncRef.sbeBlockLength())
        .templateId(marketDataIncRef.sbeTemplateId())
        .schemaId(marketDataIncRef.sbeSchemaId())
        .version(marketDataIncRef.sbeSchemaVersion());
```

Then fill the body of the message

```
marketDataIncRef.wrapForEncode(encBuffer, index+ header.size())
        .transactTime(System.currentTimeMillis())
        .matchEventIndicator(MatchEventIndicator.END_EVENT);
finalMarketDataIncrementalRefresh.MdGrpmdGrp = marketDataIncRef.mdIncGrpCount(500);
for (ObjectQuote q : instr.getQuotes().getObjects()) {
mdGrp.next();
mdGrp.securityId(instr.getId());
mdGrp.mdEntryPx().mantissa((long) q.getMessage().getPrice());
mdGrp.mdEntrySize().mantissa((int) (q.getMessage().getBuyValue() > 0 ?
q.getMessage().getBuyValue() :
q.getMessage().getSellValue()));
mdGrp.numberOfOrders((int) instr.getMessage().getOrderCnt());
mdGrp.mdUpdateAction(MDUpdateAction.DELETE);
mdGrp.aggressorSide(instr.getMessage().hasBuyQuantity() ?
Side.BUY :
Side.SELL);
}
```

The stubs wrap data to buffer provided by SBE. The DirectBuffer class isused to work with byte[], heap or directByteBuffer buffers. When messages are often encoded and decoded in memory mapped files via MappedByteBuffer and can be transferred to a network channel by the kernel thus avoiding user space copies.

In client side of the program, when the message is received, firstly, we should convert from hexadecimal to byte array and append it to instance ofDirectBufferclass.

```
CharBuffercharBuffer = hexToCharBuffer(inHex);
buffer.byteBuffer().asCharBuffer().append(charBuffer);
```

Then decode using SbeTool and following commands:

First of all fill message header

```
header= header.wrap(decBuffer, index, 0);
intblockLength= header.blockLength();
int version = header.version();
```

Then decode the message body

```
marketData.wrapForDecode(decBuffer, index+ header.size(),blockLength, version);
```

Now we can get data from studs and output to console.

```
for (final MarketDataIncrementalRefresh.MdGrpmdGrp : marketDataIncRef.mdIncGrp()) {
System.out.println(mdGrp.tradeId());
System.out.println(mdGrp.mdEntryPx().mantissa());
System.out.println(mdGrp.mdEntrySize().mantissa());
System.out.println(mdGrp.numberOfOrders());
System.out.println(mdGrp.mdUpdateAction());
System.out.println(mdGrp.aggressorSide());
}
```

3. Conclusion

According to the tests, using SBE message with FIX is really faster than FIX or ProtocolBuffer messages. After 2 thousand message transferring between server and client, SBE message encoding isapproximately 20 times, decoding 30 timesfaster thanProtoBuf and encoding 40 times, decoding 50 times faster than FIX message. This application at this time has no analogues.

References

[1] https://github.com/real-logic/simple-binary-encoding
[2] http://mechanical-sympathy.blogspot.com/2014/05/simple-binary-encoding.html
[3] http://www.fixtradingcommunity.org/FIXimate/FIXimate3.0/