Dynamic verification of input and output data streams for market data aggregation and quote dissemination systems (Ticker Plant)

Alyona Bulda
Exactpro, LLC
alyona.bulda@exactpro.com

Maria Orlova
Exactpro, LLC
maria.orlova@exactpro.com

***Abstract* - Market data aggregation and quote dissemination systems (Ticker Plants) are widely used across the electronic trading industry. A Ticker Plant is responsible for distributing information about multiple execution venues over a normalized protocol. This paper presents a dynamic verification approach for such systems. Based on a set of programs developed by the authors, it allows processing large data sets, including those collected during non-functional testing of trading platforms and using them in real-live production. The paper also outlines benefits and shortcomings of the selected approach for real-time and historical transactions analysis.**

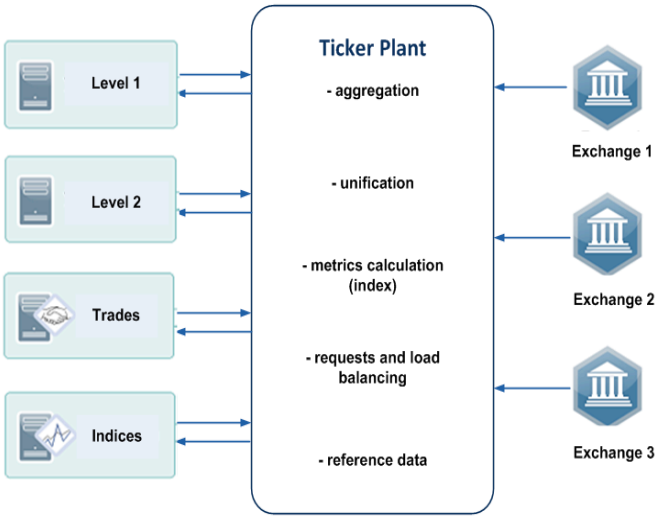
Keywords - market data, ticker plant, trading platform, exchange, non-functional testing, dynamic verification

I. Introduction

It is impossible to imagine modern day trading without up-to-date information about financial instruments, orders, and trades… Electronic trading is characterized by huge amount of data processed by a multitude of systems and algorithms. In its turn such systems and algorithms generate vast amounts of other data disseminated in various forms for a variety of purposes. One of the systems disseminating large amounts of data is a system of quotes aggregation and distribution (Ticker Plant).

A Ticker Plant is a system of aggregating market data information from various electronic trading platforms (or exchanges) and its dissemination. The system provides market data to the traders in a normalized or unified format [6]. Based on accumulated market data, Ticker Plant systems often calculate additional parameters, i.e. they enrich the static data about stocks and derivatives. One other characteristic of Ticker Plant systems is the ability to unite homogeneous values of disseminated quotes - Price Levels, real-time quote data provision based on requests from clients (e.g., such widely used services as Level 1, Level 2 [7], Index [9], T&S, News) - and storing of disseminated market data.

A schematic representation of a Ticker Plant system is provided below as Picture 1.



Picture 1. A Ticker Plant system schema

II. High Level Requirements for Ticker Plant Systems

Based on the description of a Ticker Plant system and its main characteristics noted above, we provide a set of requirements the system should comply with. We will follow this set of characteristics when testing it. A classification of such requirements was provided in [18], and in this paper we are providing a more complete list of such requirements. In order to make the assessment of each of the characteristics easier, we have divided them into functional and non-functional ones. A Ticker Plant system should:

• From a functional standpoint:

1. Collect quotes information from several sources (the suppliers of market data: exchanges, banks);
2. Process reference data provided by the exchanges;
3. Process quotes information disseminated via various data transmission protocols in real-time;
4. Convert the collected information into one format;
5. Aggregate quotes information according to various described methods;
6. Process this data in order to enrich the system's functionality: e.g., provide statistics (VWAP, Turnover, Trade High/Low, 52 week Trade High/Low) [1];

7. Provide data according to clients' requests: Level 1, Level 2, T&S, News, Index, Option chains, etc. [7] [9];
8. Provide recorded historic data about quotes.

• From a non-functional standpoint:

1. Provide fast processing of quotes data streams received from the exchanges in real-time;
2. Provide fast processing of requests received from clients and quotes data depending on the type of a client request (e.g., separately for a traded instrument, or a group of instruments, or the entire market);
3. Provide continuous working efficiency of the system;
4. Provide system operability;
5. Provide the ability of system monitoring (i.e. the availability of applications to monitor the system and operate its components);
6. Provide throughput;
7. Provide latency;
8. Provide fault tolerance.

Having defined a set of functional and non-functional characteristics of a Ticker Plant, we need to understand the process of testing each of the characteristics. The following two approaches to testing such systems become obvious.

In this paper, we would like to focus on the correctness of the logic of order construction in a Ticker Plant system and its reliability under load throughout an extended period of time.

Thus we have a specific task of verifying the outgoing data flow in Ticker Plant systems under load or real-time.

Market Data, Replay and Recovery

Highly loaded exchange and brokerage systems provide market data concerning traded financial instruments by means of their own components called Market Data Feeds. Every financial instrument comprises a fair amount of information generated every second. Therefore the ability to disseminate the entire stream of market data and the speed of disseminating market data for each financial instrument are the main characteristics for this type of components of highly loaded trading systems.

Normally market data includes the following set of parameters that are specific for a certain financial instrument: Ticker Symbol, Last Trade Price, Best Bid & Offer, ISIN, exchange code, Trade Time, Close Price. Depending on the complexity of highly loaded exchange systems, market data can be processed by the electronic exchange's internal components and enriched with additional information: e.g., daily turnover, VWAP, and more detailed information about the stock or derivative, i.e. Reference Data including, for instance, parameters of traders, market, trading sessions, and instruments. Reference Data or Static Data is instrument information, which does not change real-time: e.g., International Securities Identification Number (ISIN), price at the close of previous day's trading session (Close Price), Currency, the parameters of so called "Circuit Breakers" that are normally presented as percentages of last trade price (Dynamic or Static Circuit Breaker Tolerances (%)), and so on [1]. There is a number of standard quote dissemination protocols, such as, for instance, FIX/FAST [2], the so called quote dissemination protocols with fixed length of messages (e.g., ITCH [3]), or coded data dissemination protocols (e.g., HTTPS (HyperText Transfer Protocol Secure) [4], [5]) for providing the above mentioned information.

To transmit quote information, many electronic exchanges use both standard and bespoke protocols. Often traders cannot afford developing computer applications that would collect quote information required for their work. Therefore there is a need of creating systems allowing collection and aggregation of market data from various exchanges disseminated by using different financial data transmission protocols.

Market Data Feeds continue to disseminate data on multicast channels, however these will are active with a primary feed published on a primary network path and an identical secondary feed published on the resilient path. Recipients have access to two identically sequenced multicast feeds: Feed A and Feed B. Recipients may process both feeds and arbitrate between them to minimize the probability of a data loss.

If a gap in sequence numbers is detected on the multicast channel, the recipient should assume that some or all of the order books maintained on its systems are incorrect and initiate one of the recovery processes outlined below.

Replay Channel – The TCP replay channel should be used by clients to recover from a small-scale data loss on a particular channel. It permits to request the retransmission of a limited number of messages already published on the multicast channel. The channel supports the retransmission of the last small amount of messages published [3].

Recovery Channel – The TCP recovery channel should be used by clients to recover from a large-scale data loss. It permits to request a snapshot of the order book for the active instruments in the market data group [3].

Each multicast channel has its own dedicated instance of both a Replay and a Recovery service; each service is accessible by individual TCP sessions to dedicated IP addresses.

Protocol UDP

The two main transport layer protocols are used in the Internet, one of which requires establishing a connection and the other one doesn't. These protocols complement each other. The protocol that does not require establishing a connection is UDP (User Dataram Protocol) [20]. UDP is not a reliable protocol as it does little other than sending the packets between applications allowing them to complement that with their own protocols. UDP is an interface for IP by way of de-multiplication of several processes using ports and optional straight through error detection.

The Main Characteristics of UDP are:

- *UDP is unreliable* – it does not guarantee the delivery of packets. There is no error detection, flow control or re-transmission of lost packets. It just sends them and doesn't care whether they arrive or not.
- *UDP is a connection-less protocol* – Data is just sent and no socket needs to be established first. Data can flow one way (as in a radio broadcast) or both ways (a 2 way phone call). It also allows you to traverse some NAT devices without using port forwarding using a technique called UDP hole punching.
- *UDP is not ordered* – it does not use sequence numbers and therefore it cannot guarantee that they will reconstructed in the right order.
- *UDP is fast* – Because UDP doesn't have the additional overhead as TCP it is a faster protocol ideal for streaming.

For processes that need to drive the flow, control errors and time intervals UDP is ideal. One of the areas where it is especially useful is client-server applications. Often times the client sends a short request to the server and expects a short response. If the request and the response are lost, the client may attempt to send another request after a certain amount of time. This allows for code simplification and the reduction of the amount of required messages compared to protocols that need an initial tuning (such as TCP).

Keeping in mind the main differences between TCP and UDP it is sometimes better to use UDP over TCP. TCP should be used when no packet loss is a requirement and the data must be correct and free from errors. This is obviously useful when viewing web pages, e-mails and most forms of networking communication we are used to. When data is sent over UDP the packets can be lost.

Considering all available channels and all details and peculiarities of these channels, we have developed an approach to verify streams of the data for logical correctness via these channels. The descriptions of this approach will be started from separate parts.

III. Tools used for Ticker Plant dynamic verification

MListener is a program-model based on the mechanism of restoring "lost" updates disseminated by a channel. The program-model was developed due to the specifics of the channels used to disseminate market information. The UDP protocol is a more common way of distributing market information over such channels [20]. The protocol does not guarantee the delivery of the packets to the addressees. [20]. This program-model plays the role of a subscriber to the UDP market information channel. The logic of the program-model is simple – if a packet containing a change in the market flow is lost the program goes into "recovery' mode and re-requests the necessary information via special recovery channels provided by the exchanges. As a result, the program creates up to 2 additional files with market information per each such channel. If examined in their entirety, a complete picture of changes taking place at the exchange can be obtained.

Agglutination tool – is an instrument utility which is a part of MListener program-model. Its main function is to combine real-time market information and restored data if a small amount of real-time information was lost delivered in a queue.

Book Reader – is a program-model allowing to construct a book of quotes according to the logic of order construction in a Ticker Plant system. The implementation logic is based on the information seen by the external user only rather than on Ticker Plant system logic. This allows finding defects in such logic.

JSON (*JavaScript Object Notation*) - is a text format of data exchange. The format is based on JavaScript. There are 2 main advantages of using the format: 1) it is easily human-readable, and 2) it is easily generated and processed by the machines. This provides us with advantages from the standpoints of 1) flow analysis, and 2) the speed of program-models [19].

This method of verification based on program-models the text format of data exchange JSON is used thanks to its lightness (compared to xml for instance). Due to that the format allows for fast serializing / de-serializing and has the advantage of being inter-platform (the libraries exist for all platforms and all programming languages), flexible enough (it is possible to describe the types of data and their structures), compact enough (which is critical for large flows of data), has easily understandable method syntax if there is a need to understand the problems in the interaction of the components.

Book Checker – is a program-model allowing comparative analysis of each level of quotes added to the book in the Book Reader program-model, further pausing the work if differences are found and showing the differences as a result of performing the comparative analysis.

IV. Methods used for Ticker Plant dynamic verification

In this chapter we are going to examine the process of testing a Ticker Plant system by using the method of verifying a system of data aggregation and dissemination based on models.

To achieve better clarity, we are going to separate the testing process into the following stages:

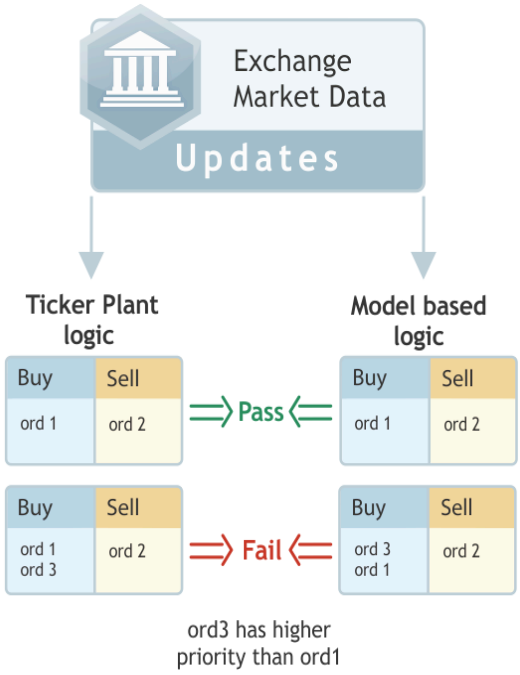
- Collection of market information about quotes from trading platforms (exchanges) delivered via a quote dissemination protocol with fixed length of messages (ITCH [3]). A "test" trading platform which is a full copy of a real electronic Ticker Plant system and MListener program-model receive incoming orders and trades information from the exchanges. At this stage, the "test" exchange [11] processes the quotes

information real-time. The MListener program-model keeps the market information in a queue without processing it.

- The collected historic information about changes allows us to build an order book for each instrument traded at the exchange after each quote change that occurred both at the exchange and the Ticker Plant system. Therefore, we receive 2 fully independent order book flows. The independence of the flows allows us track the defects that can be present in all algorithms (including the Ticker Plant algorithm).

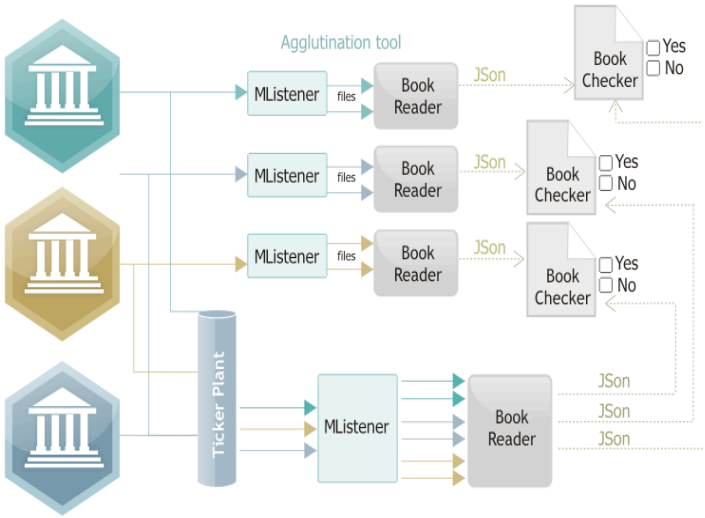
- The algorithm of the outgoing Ticker Plant flow and the outgoing exchange flow processed by program-models. Pic. 2 shows a schematic view of how the program-model works. The received Ticker Plant and exchange flows are verified in such a way that the book flow received by the program-model is the expected result for verification, while the flow built by the Ticker Plant system is the actual result to compare it with the expected one. Further, if a comparison of the two flows finds a difference, an analysis of the succession of changes that could have lead to a defect in an algorithm is performed. If the difference was caused by a defect in the program-model the same analysis is performed as to the logic and correctness of the change. If a defect in the test tool is found, it is easy to fix it and continue processing the flows.

This approach allows verifying a fairly large spectrum of scenarios to check the correctness of a Ticker Plant system operation. Such verification analysis is based on data from quote books. Such verifications include the following: that the values of added quotes are correct (price, quantity, order direction, etc.); that the order change is correct; that the book changes when an order is removed, or in case of an aggressive order change, or in case of executed trades are correct; that the book is constructed correctly in case of multi-level trades; that the priority of constructed orders is correct, that the addition of multi-level orders (10 price levels) is correct.



Pic.2 The algorithm of verifying the outgoing exchange flow processed by program-models.

We are providing a more detailed diagram of Ticker Plant system verification done by using program-models showing all of the steps of going through the program-models explained in this paper in succession.



Pic.3 The verification of Ticker Plant systems by using models.

The test approach is necessary and viable due to the fact that there is a multitude of situations involving various price positions and their changes as well as the large number of exchanges that disseminate quotes from different platforms. All such situations and combinations are structured in their entirety by using the program-models therefore eliminating the

need to spend time on arranging static pre-conditions. It is enough to generate a random set of trading combinations leading to various change combinations in quote aggregation and dissemination systems (Ticker Plant). On the other hand, this can be an emulation of trading cycles including all changes from the production platform, whose outgoing flow is also processed by program-models as described above.

V. Conclusion

This paper provides a description of quote aggregation and dissemination system (Ticker Plant), a list of its basic characteristics, its main functional components that need to be covered by testing. The paper describes the main large functional component for which a verification approach has been developed and the program-models implementing the approach have been built. The proposed method has been successfully used in practice. As the result of its implementation, a large amount of border conditions defects in order book construction has been detected.

References

1. Watson Wheatley Financial Systems, *Reconciliation Best Practice,* http://www.watsonwheatley.com/literature.html
2. M, Cochinwala, V. Kurien, G. Lalk, D. Shasha, "Efficient data reconciliation", *The Journal of Information Science,* vol.137, issue 1-4, pp. 1-15, Sep. 2001
3. Roger Nolan, The Informatica Blog, *Even 'The Most Interesting Man In The World' Won't Do This…* http://blogs.informatica.com/perspectives/2012/03/06/even-the-most-interesting-man-in-the-world-wont-do-this/
4. London Stock Exchange. *How UnaVista Works*: http://www.londonstockexchange.com/products-and-services/matching-reconciliation/how-unavista-works/index.html
5. Official site B2Bits <epam>. // [Electronic resource]. –http://www.b2bits.com/trading_solutions/market-data-solutions.html
6. Official site FIXprotocol. // [Electronic resource]. – Режим доступа http://www.fixprotocol.org/
7. Level1 Market Data Documentation / Official site London Stock Exchange // [Electronic resource]. – http://www.londonstockexchange.com/products-and-services/millennium-exchange/millennium-exchange-migration/mit303.pdf
8. P. Karlton, P. Kocher.: The Secure Sockets Layer (SSL) Protocol Version 3.0 // [Electronic resource]. – Режим доступа http://tools.ietf.org/html/rfc6101
9. Official site FTSE. // [Electronic resource]. – http://www.ftse.com/Indices/Data_Licenses/Real-time_Constituent_Data.jsp
10. Official site B2Bits <epam>. // [Electronic resource]. –http://www.b2bits.com/trading_solutions/market-data-solutions.html
11. Building the Book: A Full-Hardware Nasdaq Itch Ticker Plant on Solarflare's AoE FPGA Board / Sherman,M., Sood,P., Wong,K., Iakovlev,A., Parashar,N. // [Electronic resource]. – http://www.cs.columbia.edu/~sedwards/classes/2013/4840/reports/Itch.pdf
12. Official site Tokyo Stock Exchange // [Electronic resource]. – http://www.tse.or.jp/english/market/mkinfo/mains.html
13. Day Trading. / Official site About.com // [Electronic resource]. – http://daytrading.about.com/od/daytradingmarketdata/a/MarketDataDefin.htm
14. Customer Development Service (CDS). / Official site– London Stock Exchange // [Electronic resource]. – http://www.londonstockexchange.com/products-and-services/technical-library/customer/customerdevelopmentservice/customerdevelopmentservice.htm
15. Trading Floor Architecture / Official site Cisco // [Electronic resource]. – http://www.cisco.com/en/US/docs/solutions/Verticals/Trading_Floor_Architecture-E.html
16. Official site BSE India // [Electronic resource]. – http://www.bseindia.com/markets/MarketInfo/DispNoticesNCirculars.aspx?page=20120531-22&pagecont=0,31/05/2012,31/05/2012,,All,All,All,Scrip%20Name%20/%20Code
17. Official site MICEX // [Electronic resource]. – http://rts.micex.ru/s437
18. Статья К.В.Воронцов (ВЦ РАН), С.Б. Пшеничников (ММВБ): "Имитационное моделирование торгов: новая технология биржевых тренажеров." / Журнал «Индикатор», №2 (42). М. 2002 год // [Electronic resource]. — http://www.forecsys.com/site/about/press/exchange_simulator/
19. Official site Oslo Bors Stock Exchange. // [Electronic resource]. –http://www.oslobors.no/ob_eng/Oslo-Boers/Trading/Delta/Millennium-Exchange/Guide-to-Testing-Services-updated-issue
20. Zverev,A., Bulda,A.: Exchange Simulators for SOR. Algo Testing: Advantages vs. Shortcomings. / Конференция ExTENT // [Electronic resource]. – http://www.slideshare.net/extentconf/exchsims-forsoralgotestingadvantagesvsshortcomings29102011111113011104phpapp02
21. Official site Exactpro Systems. // [Electronic resource]. –http://www.exactprosystems.com/
22. Bulda A., Buyanova O., Zverev A., - Article "Usage of Exchange Simulators and Test Exchanges as Tools for Ticker Plant Systems", https://yadi.sk/d/eKKxTtEGCCeF6
23. Official site JSON. // [Electronic resource]. – http://www.json.org/index.html

24. Andrew S. Tanenbaum, David J. Wetherall.: Computer Networks. // [Electronic resource]. – http://cse.hcmut.edu.vn/~minhnguyen/NET/Computer%20Networks%20-%20A%20Tanenbaum%20-%205th%20edition.pdf