Method of Performance Analysis of Time-Critical Applications Using DB-Nets

Anton Rigin
Faculty of Computer Science
HSE University
Moscow, Russia
amrigin@edu.hse.ru

Sergey Shershakov
Faculty of Computer Science
HSE University
Moscow, Russia
sshershakov@hse.ru

***Abstract*—These days, most of time-critical business processes are performed using computer technologies. As an example, one can consider financial processes including trading on stock exchanges powered by electronic communication protocols such as the Financial Information eXchange (FIX) Protocol. One of the main challenges emerging with such processes concerns maintaining the best possible performance since any unspecified delay may cause to a large financial loss or other damage. Therefore, a performance analysis of time-critical systems and applications is required. In the current work, we develop a novel method for a performance analysis of time-critical applications based on the db-net formalism which combines the ability of colored Petri nets to model a system control flow with the ability to model relational database states. This method allows to conduct a performance analysis for time-critical applications that work as transactional systems and have log messages which can be represented in the form of table records in a relational database. One of such applications is a FIX protocol-based trading communication system. This system is used in the work to demonstrate applicability of the proposed method for time-critical systems performance analysis. However, there are plenty of similar systems existing for different domains, and the method can also be applied for a performance analysis of these systems. The software prototype is developed for testing and demonstrating abilities of the method. This software prototype is based on an extension of Renew software tool, which is a reference net simulator. The testing input for the software prototype includes a test log with FIX messages, provided by a software developer of testing solutions for one of the global stock exchanges. One of possible applications of the method is presented. The developed method can be used in the further research in this domain as well as in testing a performance of real time-critical software systems.**

Keywords—performance analysis, time-critical applications, db-nets, FIX protocol, software modeling, software testing

I. Introduction

Nowadays, most of time-critical business processes are performed using computer technologies. Nuclear reactor control, medical equipment control, spaceship control are some obvious examples of such processes. However, different financial processes including trading on stock exchanges also can demand strict performance requirements.

In the previous century, trading on stock exchanges was primarily performed through phone calls and with use of paper-based order books [1]. Working this way did not allow traders to compete for the best price that is generally offered during very short period. This was the reason of beginning of automatization of trading on stock exchanges. In order to guarantee compatibility of software systems of different traders, brokers, and exchanges, there were financial protocols for electronic communication between trading participants created. Financial Information eXchange (FIX) Protocol maintained by the FIX Trading Community [2] is one of the most known and widely used protocols of such type. There exist different approaches to encode messages transferring with the FIX protocol. In this paper we focus on the *FIX TagValue Encoding*, which is the main standard of encoding FIX messages [2].

The FIX protocol allows traders, brokers, and exchanges to create and fill (execute) orders for buying or selling securities in several milliseconds using electronic communication channels such as Internet [2]. It is a great driver for competence in the global stock markets, however it creates new challenges for financial software vendors. One of such challenges is maintaining the best possible performance. Any unspecified delay may cause to a large financial loss for a trader due to the best price is missed. Such delays may create unequal and unfair conditions for different participants, lead to local or global economic problems as well as public scandals and reputational problems for the exchange or some traders or brokers.

Financial protocol-based communication systems are considered in this work to demonstrate applicability of the proposed method for time-critical systems performance analysis. However, there are plenty of similar systems existing for different domains, and the method can also be applied for a performance analysis of these systems.

Any FIX message consists of a set of tag-value pairs [3]. In fact, it means that we can represent these messages in the form of records of a table in some relational database. Therefore, some methods of system modeling, which rely on relational database states, can be considered here. The same is valid not only for messages of the FIX protocol, but for any messages of transactional systems that are represented as sets of tag-value pairs.

In 2020, we developed a software simulator for the db-net formalism [4] introduced by Montali and Rivkin in 2017. This formalism is represented by the layer with modified colored Petri net modeling a control flow of a process system, and two inner layers for working with an attached relational database modeling a persistent storage [5] as shown in Fig. 1. This simulator is developed as a plugin for Renew (Reference Net Workshop) software tool which is a Java-based reference net simulator [6].

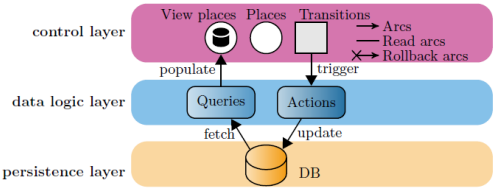

Fig. 1. The db-net structure [5].

Generally, the lowest layer of the db-net (the persistence layer) is represented by an ordinary relational database [5]. However, it can be replaced with any other information storage, which is accessible through a custom relational DML interface that is to be implemented.

One can model a tag-value message sending by using the "insert" database operation, where tags are represented as attributes of a relational table and values are represented as attributes of a record in the table. A tag-value message receiving can be modeled similarly using the "select" database operation.

In the current time, there are some performance analysis research works focused on distributed software systems such as [7, 8], however a performance analysis using db-nets has its advantages for transactional systems which send and receive messages that are representable in the form of records in relational tables. Firstly, this method allows to apply well-known approaches used in the relational database domain to the wide set of transactional systems supporting time-critical applications. Secondly, it allows to combine performance analysis of transactional systems with other methods for their verification and validation, based on Petri nets and their modifications, especially db-nets (e.g., checking safety, liveness, fairness, and similar properties).

All the above provides the motivation for the research.

The purpose of the research is development of a method of performance analysis of time-critical applications using db-nets.

The objectives of the research are as follows.

1) Developing a method for performance analysis of time-critical applications using db-nets.
2) Developing a software prototype for performance analysis of time-critical application logs using db-net models.
3) Checking the method by testing the developed software prototype on a test log of FIX messages provided by a software developer of testing solutions for one of the global stock exchanges.

The rest of the paper is organized as follows. The Section II presents the theoretical foundations and concepts of the work and the developed method. In the Section III, the developed software prototype and its testing are described. After this, the main points of the paper are summarized in the conclusion.

II. Performance Analysis Using DB-Nets

A. DB-Nets

The db-net formalism is a modification of the colored Petri net, which allows to model a system control flow together with relational database states. The db-net consists of three layers: (1) the control layer, (2) the data logic layer which connects the control layer and the persistence layer together, and (3) the persistence layer [5]. The scheme of db-net structure is shown in Fig. 1.

The persistence layer allows to store the persistent data and is formally defined by a relational database schema and constraints that declare the data consistency rules [5].

The data logic layer is defined by two sets: (1) set of *queries* for retrieving records from a database in the persistence layer and (2) set of *actions* for insertion and deletion of records in the persistence layer database. Each action includes sets of added and deleted *facts* (records in relational tables) [5].

The control layer allows to model a system control flow and is defined by a colored Petri net with the following modifications [5].

1) Queries defined in the data logic layer are assigned to places of a colored Petri net in the control layer. Such places are called *view places*. View places cannot contain *tokens* (resources modeled in a Petri net) such as other places, but they produce new tokens by retrieving data from the persistence layer through assigned queries.
2) Actions defined in the data logic layer are assigned to transitions of the net in the control layer. When a transition with the assigned action is *fired* (executed), the action is performed on a database in the persistence layer.
3) In addition to traditional Petri net arcs, there exist *read arcs* and *rollback arcs* in the db-net control layer. The former is used for connecting view places with transitions and the latter is used for defining a flow for a case of rollback of an action due to violation of the data consistency rules in a database of the persistence layer after performing the action.

The db-net control layer's net and persistence layer's database schema example for the taxi booking software system is shown in Fig. 2.

B. Method of Performance Analysis Using DB-Nets

The developed method implies analyzing messages sent or received by the application or its modeled part (*request* and *response messages*, respectively) and stored in a log of the application. We analyze those messages for which a maximum delay between sent message and received response is restricted. The method utilizes the db-net formalism.

The method consists of two parts: (1) set of requirements for implementing the method in a software tool and (2) sequence of stages and steps for using the method after being implemented.

1) Implementing the Method in a Software Tool

The following set of requirements specifies how the method of performance analysis using db-nets is to be implemented as a software tool. These requirements extend general principles of the db-net behavior, which are described in [5].

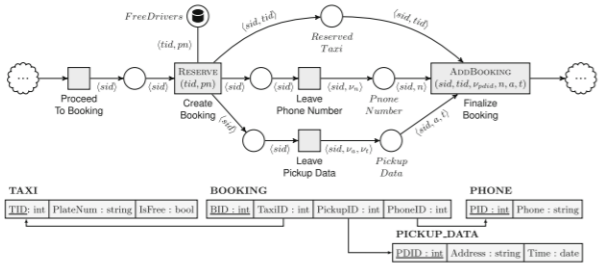

Fig. 2. The control layer's net and persistence layer's database schema example for the taxi booking software system [5].

1) When a *request message* is inserted in an action assigned to a db-net transition, it should be stored in the memory (RAM or persistent storage) for further retrieving when the corresponding response message is retrieved.

2) When a *response message* is retrieved by a "select" query assigned to a db-net view place *and* the connected by a read arc db-net transition contains parameters for performance analysis as specified in the step 6 of the stage 1 of the method (the Section II.B.2.a), the following sequence of steps is to be executed:

   a) The corresponding request message (with the same *id* attribute value) is to be retrieved through the specified query from the memory/storage (as specified in the item 1 of the current set of requirements).

   b) If there is no stored corresponding request message, then this sequence is to be stopped and the token with the response message is to be moved to the places connected by output arcs.

   c) The *sending timestamps* of the request and response messages are to be parsed using a specified pattern or a regular expression.

   d) A *delay* that is a difference (in milliseconds) between these two *sending timestamps* is to be calculated. If it exceeds the specified *maximum acceptable value of a delay*, then the validation is to be considered as failed – information about the *id* and *message type* of the problematic messages is to be displayed or stored in the report (depending on the requirements and implementation), for the first violation or for each violation (also depending on the requirements and implementation).

3) If there are several response messages for one request message, only the first response message is considered.

4) If the simulation is finished (no transitions can be *fired* – executed) and the validation did not fail, then such validation is considered as succeeded.

2) Use of the Method

After implementing the software tool, the method is to be used by following the sequence of steps divided into three stages, as follows.

a) Stage 1. Modeling a DB-Net

A db-net that matches a system/a modeled part of a system is to be modeled using the following steps.

1) A scope of the modeled system is to be defined. It should include considered components of the system which send *request messages* (messages sent by the system or its considered component) and get responses to them (*response messages*), and considered types of *request messages* and corresponding types of *response messages*. From now on, we will call a modeled system/part of the system a *time-critical application* (or just an *application*).

2) It is necessary to make sure that the application works as a transactional system and satisfies the ACID (atomicity, consistency, isolation, durability) properties [9], and a log with its request and response messages can be represented in the form of tables in a relational database. It means that each message includes a set of *tags* (attributes) together with their *values*. *Tags* are represented as attributes of a relational table, messages are represented as records of the table, and *values* are represented as attributes of a record in the table. Types of messages and parts of the application which do not satisfy these properties, if any, are to be removed from the scope.

3) A persistence layer of the modeled db-net is to be defined. To do this, a relational database schema is to be created and populated with necessary tables. The table attributes reflect the tags of considered request and response messages.

4) A data logic layer of the modeled db-net is to be defined. The "insert" queries, which model insertion of the request messages into the modeled relational database, are to be specified. The "select" queries, which model retrieving the request and response messages from the modeled relational database, should similarly be specified.

5) A model of a system control flow (a control layer of the modeled db-net) is to be defined. After that, “insert” and “select” queries from the modeled data logic layer are assigned to transitions and view places, respectively.

6) For each db-net transition connected by a read arc with a view place that is assigned with a "select" query for retrieving the response messages, the following parameters for conducting a performance analysis are to be specified:

   a) The name of a variable in the control layer that stores a value of the *id* attribute of a response message, which allows to find a corresponding request message by the same value of the same *id* attribute.
   b) The name of a variable in the control layer that stores a value of the *sending timestamp* attribute of a response message.
   c) An ordering number of the *sending timestamp* attribute of a message in results of a "select" query for retrieving the corresponding request message, that is mentioned in the item “f” of the current list.
   d) A pattern or a regular expression for parsing the *sending timestamp* string in a message.
   e) An ordering number of the *message type* attribute of a message in results of a "select" query for retrieving the corresponding request message, that is mentioned in the item “f” of the current list.
   f) The name of a declared "select" query for retrieving the corresponding request message.
   g) The *maximum acceptable value of a delay* between *sending timestamps* of corresponding request and response messages (in milliseconds).

b) Stage 2. Preprocessing the Log

Preparing a log of the application includes the following steps.

1) It is necessary to make sure that the messages in a log are represented in a form satisfying properties

described in the step 2 of the stage 1. Any messages that are not represented in a valid form as well as broken messages are to be removed.

2) The log should be prepared in a format compatible with a software tool implementing the method.

c) Stage 3. Conducting a Performance Analysis Using DB-Nets

A simulation of the modeled db-net is to be run in the software tool implementing the method.

C. Example of Performance Analysis Using DB-Nets for the FIX Protocol

The developed method is illustrated by an example modeling a trading order creation with use of the FIX protocol. The example includes the analysis of two types of FIX messages: (1) *create_order_single* (*msg_type = "D"*) which is used for request messages sent from a trader or a broker to the exchange, to create an order for buying or selling securities, and (2) *execution_report* (*msg_type = "8"*) which is used for response messages sent from the exchange to the trader or the broker as a confirmation of the order creation (or information about the order rejection with clarification of a reason). For each message, the attributes *msg_type*, *cl_ord_id* and *sending_time* are considered in the model. The *msg_type* attribute defines a type of the message. The corresponding request/response messages are connected by a key (id), whose role is played by the *cl_ord_id* attribute. The *sending_time* attribute is a sending timestamp of the message.

The db-net modeling this example is shown in Fig. 3. A schema of a relational database in the db-net persistence layer includes a *msg* relational table for storing FIX messages. The table contains *msg_type*, *cl_ord_id* and *sending_time* attributes. The *create_order_single* action models the "insert" DML query for insertion of the *msg_type*, *cl_ord_id* and *sending_time* attributes of the *create_order_single* FIX message. The *create_order_single* and *execution_report* queries model the "select" SQL query for retrieving the same attributes of the *create_order_single* and *execution_report* FIX messages, respectively. The *create_order_single_corr_req* query models the "select" SQL query for retrieving the same attributes of the *create_order_single* FIX message by the given *cl_ord_id*. It is used for retrieving the corresponding request message for a previously retrieved response message.

The view place assigned with *create_order_single* query (Fig. 3) is responsible for retrieving messages of *create_order_single* type. The following transition executes the *create_order_single* action, modeling insertion of the messages into the *msg* table. Then the transition transfers the messages to the *Processed messages* place.

The view place assigned with *execution_report* query (Fig. 3) is responsible for retrieving messages of *execution_report* type. After retrieving an *execution_report* message, the following transition retrieves the corresponding *create_order_single* message (with *msg_type = "D"* and the same *cl_ord_id*) using the *create_order_single_corr_req* query and calculates a delay between these two messages as a difference between their sending timestamps (the *sending_time* attribute). If the calculated delay exceeds *max_delay* (it is 100 ms in the example), then the validation fails. Otherwise, the *execution_report* message is transferred to the *Processed messages* place. After all messages are retrieved from the log, and validation did not fail, the validation of the log is considered succeeded.

We consider two followng FIX messages (these messages are presented below in the human-readable form, not in the original FIX tag-value form): (1) *create_order_single (msg_type = "D", cl_ord_id = "12345", sending_time = "20190218-02:14:45.490000")* and (2) *execution_report (msg_type = "8", cl_ord_id = "12345", sending_time = "20190218-02:14:45.492787")*. Firstly, the *create_order_single* message is retrieved by the view place assigned with the *create_order_single* query. The following transition performs the *create_order_single* action with the "insert" DML query for this message and transfers the message to the *Processed messages* place. Secondly, the *execution_report* message is retrieved by the view place assigned with the *execution_report* query. By the *cl_ord_id = "12345"* attribute value of the message, the following transition retrieves the corresponding *create_order_single* message (with *msg_type = "D"* and the same *cl_ord_id = "12345"*) using the *create_order_single_corr_req* query and calculates a delay between these two messages as a difference between their sending timestamps (the *sending_time* attribute). This delay equals 3 ms (rounding up). A maximum acceptable delay linked with the transition is defined to 100 ms. The delay does not exceed the maximum acceptable delay, so the validation does not fail, and the *execution_report* message is transferred to the *Processed messages* place. However, if the *sending_time* attribute value of the *execution_report* message was, for example, *"20190218-02:14:45.592787"*, then the delay would be equal to 103 ms (rounding up) and the maximum acceptable delay would be exceeded which would lead the validation to fail.

III. Software Prototype

A. Software Prototype Features and Implementation

For testing and illustrating abilities of the method, the latter is implemented in the form of a software prototype. For doing this, we developed the db-net software simulator (Renew DB-Nets Plugin) in 2020 [4] and then extended it with features for conducting a performance analysis of time-critical applications using the proposed method. The simulator has a form of a plugin for Renew software tool which is a Java-based reference net simulator [6]. The simulator has a graphical user interface as shown in the screenshot in Fig. 4.

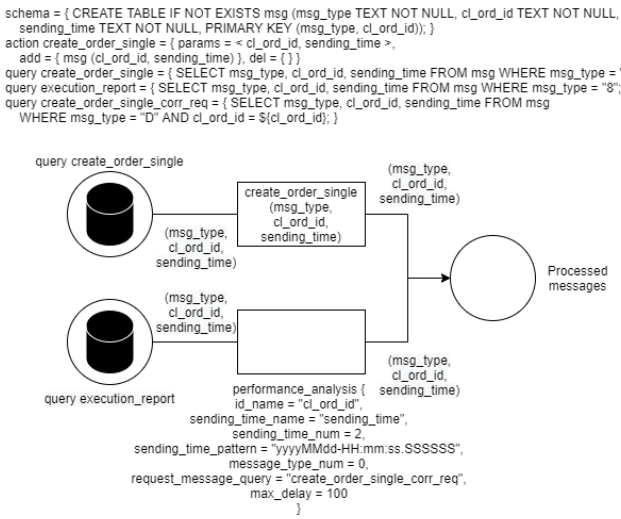

Fig. 3. Example of a db-net model for a performance analysis of a FIX protocol-based system.

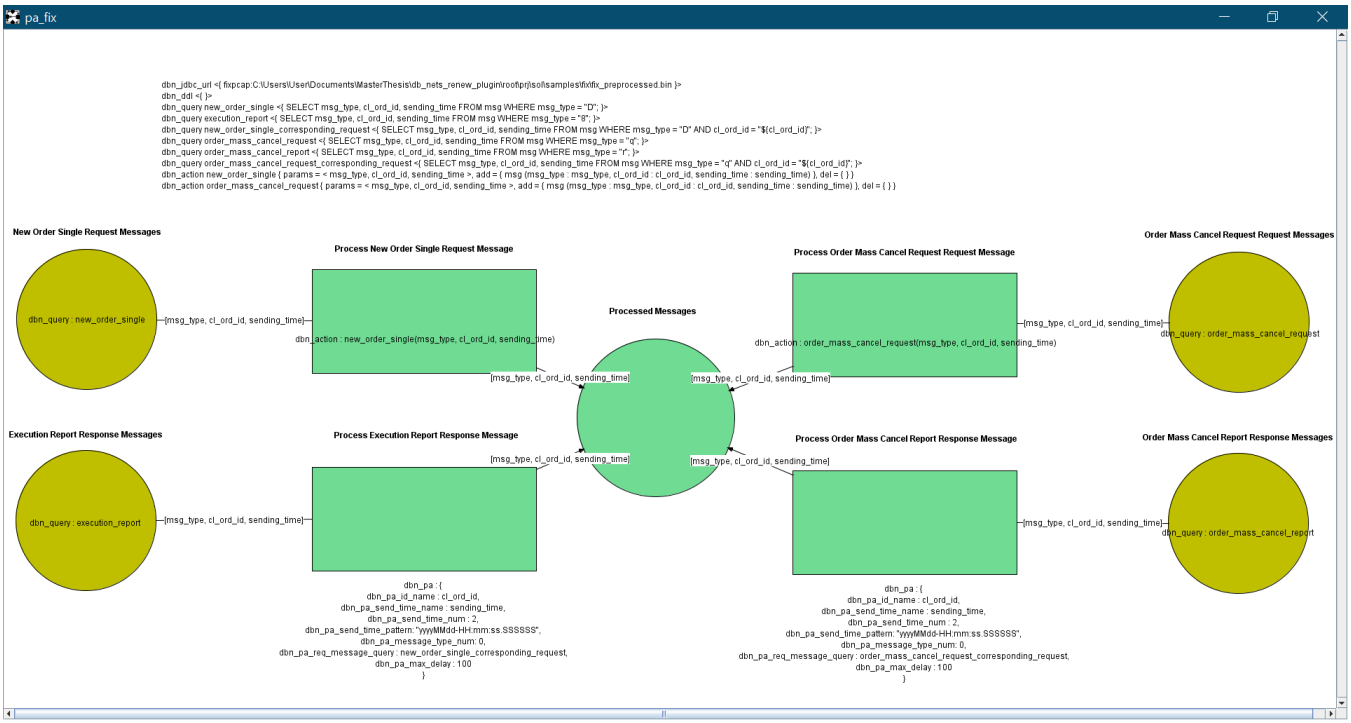

Fig. 4. Screenshot of a graphical user interface of the developed software prototype.

The prototype allows to (1) model a db-net for a considered system, (2) specify parameters for conducting a performance analysis of time-critical applications, as described in the step 6 of the stage 1 of the described method (the Section II.B.2.a), (3) conduct a performance analysis of an application in parallel with a db-net model simulation using the proposed method and (4) work with a FIX log (raw binary data of the FIX protocol packages captured as a Wireshark PCAP file [10] with further filtering) through a relational DML interface.

An implementation of the developed db-net simulator is described in [4]. This implementation is based on an implementation of Renew software tool, a reference Petri net simulator. The Renew code which was suitable for the db-net behavior is reused. Other code is overridden by a custom db-net implementation. Classes representing elements of the db-net control layer are inherited from Renew classes representing similar elements of traditional colored Petri nets and necessary methods are overriden. The prototype is implemented as a pure plugin for Renew tool, without modifying existing Renew source code [4]. The plugin code, UML class diagram and documentation are available in the project GitHub repository[1].

For working with a FIX log through a relational DML interface, the alternative implementation of the database connection interface is created. It is used if the JDBC URL in a db-net model starts from the "*fixpcap:"* prefix. All messages that are read from file through this connection are stored in RAM (in the *java.util.HashMap* container, where keys, which are pairs of *message type* and *id*, are stored in a hashtable). When the message is being retrieved through this connection, it is firstly searched in RAM. If it is found in RAM, it is returned and removed from RAM. If it is not found in RAM, then the file is scanned until finding this message (and all scanned messages are stored in RAM). This approach allows to scan each line of the file only once and to minimize the RAM usage.

For goals of a performance analysis, the prototype follows the set of requirements described in the Section II.B.1. When the first *maximum acceptable delay* violation is detected while simulating a db-net model, the dialog window with an information message describing this violation is shown and the corresponding CSV report is created. All *maximum acceptable delay* violations that are detected during the current simulation are written into the created CSV report. The format of a CSV report is presented in Table I.

B. *Testing the Prototype on the FIX Log and Quantitative Analysis of Maximum Acceptable Delay Violations*

The developed software prototype is tested on a log with FIX protocol messages, which is represented by the raw binary data extracted from a Wireshark PCAP file with some FIX protocol messages captured in the testing environment.

TABLE I. COLUMNS OF THE CSV REPORT

Column Name	Description	Type	Example
#	Order number of the row in the CSV report (starting from 1)	Integer	1
Request Message Type	Type of the request message	String	D[a]
Message ID	ID of the request and response message pair	String	15504
Delay	Difference (in milliseconds) between request and response message sending timestamps	Integer	493
Max Delay	Maximum acceptable delay	Integer	100
Diff	Difference between detected delay and maximum acceptable delay	Integer	393

[a.] In the FIX Protocol, the *D* message type is used for the *New Order Single* messages.

[1] Link: https://github.com/Glost/db_nets_renew_plugin

The file is provided by a software developer of testing solutions for one of the global stock exchanges.

The screenshot in Fig. 4 shows the db-net model for performance analysis applied to the FIX protocol messages for the *New Order Single* scenario (request message: *New Order Single*, message type: *"D"*; response message: *Execution Report*, message type: *"8"*) and the *Order Mass Cancel Request* scenario (request message: *Order Mass Cancel Request*, message type: *"q"*; response message: *Order Mass Cancel Report*, message type: *"r"*). The total number of processed messages in this model equals 321671.

Using this model, the quantitative analysis of *maximum acceptable delay* violations is conducted based on the CSV reports with information about violations. The plots in Fig. 5 show (1) counts of *maximum acceptable delay* violations and (2) percentages (ratios) of message pairs with *maximum acceptable delay* violations (where 100 % is all processed message pairs), in the db-net model described above, with a breakdown to the request message types (*"D"* is used for the *New Order Single* messages and *"q"* is used for the *Order Mass Cancel Request* messages) for maximum acceptable delay values from 1000 ms to 9000 ms.

The significant decrease in count of violations between *maximum acceptable delay* values 3000 ms and 4000 ms is notable. The plots in Fig. 6 show the same metrics for *maximum acceptable delay* values from 3100 ms to 3900 ms. We can conclude that the most of delays larger than 1 second are between 3 and 4 seconds.

Such quantitative analysis is an example of possible applications of the developed method. For instance, requirements and service level agreements (SLAs) can be specified and adjusted basing on some statistics on ratio of message pairs violating each *maximum acceptable delay*. This information with a breakdown to the request message types allows to focus on improving the speed of the most critical scenarios.

IV. Conclusion

In the current work, a novel method of performance analysis of time-critical applications based on the db-net formalism is developed. This method allows to apply well-known approaches used in the relational database domain to the wide set of transactional systems supporting time-critical applications. Moreover, the method combines performance analysis of transactional systems with other approaches for their verification and validation, based on Petri nets and their modifications, especially db-nets (e.g., checking safety, liveness, fairness, and similar properties).

A software prototype implementing the method is developed. The prototype is checked on a test log with FIX

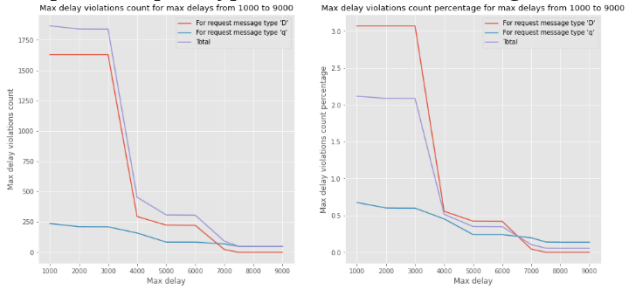

Fig. 5. Quantitative analysis of maximum acceptable delay violations for maximum acceptable delay values from 1000 ms to 9000 ms.

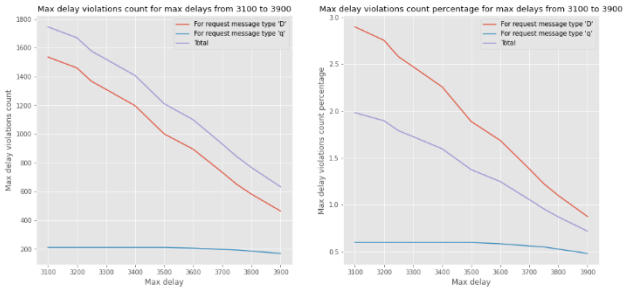

Fig. 6. Quantitative analysis of maximum acceptable delay violations for maximum acceptable delay values from 3100 ms to 3900 ms.

messages provided by a software developer of testing solutions for one of the global stock exchanges. The quantitative analysis of maximum acceptable delay violations is conducted based on this log. This demonstrates how the method can be applied for similar analysis.

The developed method can be used in the research in this domain as well as in testing a performance of real time-critical software systems. Further steps include extending the method for use with hierarchical Petri nets and more complex variants of performance analysis of transactional systems.

Acknowledgment

This work is supported by the Basic Research Program at the HSE University.

References

[1] L. Harris, “Back Office Operations,” in *Trading and Exchanges: Market Microstructure for Practitioners*, New York, NY, USA: Oxford Univ. Press, 2003, ch. 7, sec. 7.2.2, pp. 148 – 149.

[2] “Introduction.” FIX Trading Community. https://www.fixtrading.org/online-specification/introduction/ (accessed Mar. 28, 2021).

[3] “FIX TagValue Encoding.” FIX Trading Community. https://www.fixtrading.org/standards/tagvalue-online/ (accessed Mar. 28, 2021).

[4] A. Rigin and S. Shershakov. “Data and Reference Semantic-Based Simulator of DB-Nets with the Use of Renew Tool,” in *Analysis of Images, Social Networks and Texts (AIST 2020)*, W. M. P. van der Aalst et al., Eds., Springer LNCS, vol. 12602, Berlin, Heidelberg, Germany: Springer, 2021, pp. 453 – 465, doi: 10.1007/978-3-030-72610-2_34.

[5] M. Montali and A. Rivkin. “DB-Nets: On the Marriage of Colored Petri Nets and Relational Databases,” in *Transactions on Petri Nets and Other Models of Concurrency XII*, M. Koutny, J. Kleijn, W. Penczek, Eds., Springer LNCS, vol. 10470, Berlin, Heidelberg, Germany: Springer, 2017, pp. 91 – 118, doi: 10.1007/978-3-662-55862-1_5.

[6] “Renew – The Reference Net Workshop.” Renew.de. http://www.renew.de/ (accessed Mar. 28, 2021).

[7] J. Vetter, “Performance analysis of distributed applications using automatic classification of communication inefficiencies,” in *Proceedings of the 14th international conference on Supercomputing (ICS '00)*, J. Reynders, A. Veidenbaum, Eds., New York, NY, USA: Association for Computing Machinery, May 2000, pp. 245 – 254, doi: 10.1145/335231.335255.

[8] M. A. Marsan, A. Bianco, L. Ciminiera, R. Sisto and A. Valenzano, "A LOTOS extension for the performance analysis of distributed systems," in *IEEE/ACM Transactions on Networking*, vol. 2, no. 2, April 1994, pp. 151 – 165, doi: 10.1109/90.298433.

[9] T. Haerder and A. Reuter, “Principles of transaction-oriented database recovery,” in *ACM Computing Surveys*, vol. 15, no. 4, New York, NY, USA: Association for Computing Machinery, Dec. 1983, pp. 287 – 317, doi: 10.1145/289.291.

[10] “5.2. Open Capture Files.” Wireshark.org. https://www.wireshark.org/docs/wsug_html_chunked/ChIOOpenSection.html (accessed Mar. 28, 2021).