Performance Analysis of Message-Oriented Middleware Using Stochastic Petri Nets

Stênio F. L. Fernandes, Wellington J. Silva, Mauro J. C. Silva, Nelson S. Rosa, Paulo R. M. Maciel, Djamel F. H. Sadok

Centro de Informática – Universidade Federal de Pernambuco
PO Box 7851, Cidade Universitária, Recife-PE, 50732-970

{sflf, wjs, mjcs, nsr, prmm, jamel}@cin.ufpe.br

Abstract. *This paper presents how to model and carry out the performance analysis of message-oriented middleware (MOM) using Generalised Stochastic Petri Net (GSPN) models. The results obtained from the Petri Net analysis are compared against ones measured in a commercial MOM. Additionally, some results are presented in order to demonstrate the flexibility and the benefits of the proposed model. This research also focuses on how to improve the MOM performance by suggesting appropriated techniques and adjustments to the MOM basic architecture. Finally, we point out some decisions usually taken by systems administrators that may have a major impact on the performance of MOM systems.*

1. Introduction

The number of middleware products available for building distributed applications is rapidly increasing. Those products are usually implemented according to open standards such as DCE (Distributed Computing Environment) [18], RM-ODP (Reference Model – Open Distributed Processing) [16] and CORBA (Common Object Request Broker Architecture) [17]. Additionally, proprietary middleware such as MQSeries (IBM), .NET (Microsoft) and RMI (Sun) have also been widely used by distributed application developers.

Such variety of standards and products makes difficult a comparison between their performances. In general, it is necessary a careful analysis of the system's architecture in order to achieve high performance [1]. System administrators and researchers should rely on the use of quantitative evaluation and modelling techniques, since it is laborious to accurately predict the performance behaviour of these technologies.

The modelling and performance analysis of middleware systems is a recent research area. Tran [1] presents a performance evaluation analysis of Message-Oriented Middleware (MOM) though measurements, which focused on the IBM's MQSeries V5.2 in a test environment (*Testbed*). The authors have evaluated the impact of some factors such as buffer sizes and persistence mechanisms in variety of scenarios. Khorasani [10] presents a model for Middleware in telecommunications focusing on the performance issues. The analysis focused on the impact of adding security functions to a middleware prototype. Verdickt [11] [9] proposes an extension of the traditional

queue networks systems [5], namely Layered Queuing Networks (LQN), to model the performance of the Common Object Request Broker Architecture (CORBA). Abdul-Fatah [2] has implemented four different test prototypes based on the CORBA architecture, with different configurations for client-server interaction. The experiment results clearly show that there was no better choice amongst them. The authors affirm that a better solution depends on the load requirements of a particular application. Due to the difficulty to set up test environments, we consider such results as an indication to the need of analytical and simulation models for Middleware systems. Pang [3] implements a testbed in order to perform benchmarking tests of two MOM, namely TIB/RV and SonicMQ. The tests evaluated the system's capacity of message delivering (effective throughput), stability under high loads and resource utilization. From the point of view of Petri Net (PN) modelling, we consider that one attractive characteristic of the TIB/RV's architecture is that the component responsible for delivering messages uses User Datagram Protocol (UDP) as the transport protocol. As UDP have not any implicit or explicit control messages to end-user applications, it is possible to simplify a PN model and consequently to increase performance in simulations. Liu [4] proposes a model for performance analysis of middleware Business Process Integration (BPI). The model is based on LQN and its precision has been compared against the results obtained by measurements performed using the IBM CrossWorlds InterChange Server. As far as we know, until the publication of this paper there is no related work to PN-based modelling and evaluation of Message-Oriented Middleware.

The main objective of this work is to model and to evaluate performance behaviour of middleware systems relying on Petri Nets. Particularly, we focus the analysis on Message-Oriented Middleware (MOM) due to its vast utilization in the marketplace and present a GSPN model for the IBM MQ Series v5.2, currently known as WebSphere MQ 5.3 [12].. In turn, the use of Petri Nets models was chosen for the following reasons:

- It is a well-known, widespread and suitable formalism for system performance evaluation;
- It allows a natural modelling of various MOM elements such as queues, senders and receivers application;
- It catches several dynamics aspects such as asynchronous operation, queue and message sizes, failures in components, Quality of Service (QoS) levels, capacity, loss rate, deadlock etc.

The remainder of this paper is organized as follows. Section 2 briefly presents theory of Generalised and Stochastic Petri Nets (GSPN). Fundamentals of Message-Oriented Middleware are described in Section 3. PN-based MOM models used in the evaluation as well as the experiments results from simulations are presented in Section 4. Finally, Section 5 presents some concluding remarks and future works.

2. Generalised Stochastic Petri Nets

Petri nets are a family of formal specification techniques that allows for a graphical, mathematical representation and have powerful methods, which allow designers to perform qualitative and quantitative analysis. Place/transition Petri nets are used to model a logical point of view of the systems, however no formal attention is given to temporal relations and constraints.

GSPN is one of the most extensively adopted classes of stochastic Petri nets. A GSPN is defined by a set of places, a set of transitions, relations describing pre-conditions, post-conditions, and inhibition conditions; and a mapping from the set of places to the natural numbers describing the model's state. The set of places represents the set of resources, local states and system's variables. The set of transitions represents the set of actions. This set is divided into two subsets: the set of immediate transitions that depicts a set of irrelevant actions under the performance point of view; and the subset of timed transitions. Besides, two other functions are taking into account for representing timing and priorities. The timing function associates to each timed transition a non-negative real numbers, depicting the respective exponential transition delay (or rate). The priority function associates to each immediate transition a natural number that represents the respective transition priority level.

Transitions are fired under interleaving firing semantics, a common semantics adopted even in the untimed place/transition model. However, immediate transitions have higher priority than those timed transitions.

From a given GSPN, a reachability graph is generated containing markings of the reachability sets as nodes and rates (or delays) associated to the arcs. Bounded, live and reversible GSPN models allow the generations of reachability graphs from which ergodic Continuous-Time Markov Chains (CTMC) are directly derived. Steady state and transient analysis are carried out on such models. Another possibility is to simulate the GSPN model in order to obtain steady state or transient measures.

3. Message-Oriented Middleware Systems - MOM

Message-oriented middleware (MOM) [8] enables communication by passing information in a message from one application to one or more applications. Two different MOMs are usually implemented: message queuing and publish/subscribe. Message queuing model provides the abstraction of a queue that can be accessed across a network. The publish/subscribe model is more elaborated as it enables a single message (sent by a publisher) to be sent to several subscribers.

The interaction in the MOM model is carried as follows: a component simply gives a message to the queuing service, which takes responsibility to transmit (actually aided by the communication service) the message to another component (or components). In the publish/subscribe MOM, a component subscribes to (register interest in) a subject. Another component publishes messages to the subject and the components that have been subscribed to that subject receive the message. MOMs traditionally provides only the queuing and communication services. The queuing service manages the queue, which stores every message that must be sent from the sender to the receiver. It is worth observing that in the communication model of MOMs, there is not the notion of client and server, but only sender and receiver. The reason for this differentiation comes from the fact that the sender sends a message to the receiver, but the message is not a request that needs to be replied (asynchronous communication).

4. The GSPN Modelling Approach

In this section, we describe the model assumptions and provide a detailed explanation of the GSPN model that we propose for MOMs. Particularly we present a GSPN model for the IBM MQ Series v5.2, currently known as WebSphere MQ 5.3 [12]. We choose that

MOM for the reason that there exists an independent performance evaluation based on measurements, as described in [1]. The IBM WebSphere MQ 5.3 was designed to send messages in an asynchronous manner. Messages sent by applications are queued in message queues and can be read by their receiving applications. These main characteristics of the WebSphere's architecture allow for a direct GSPN modelling. So, it is feasible to compare our results to those measurements, with no need of setting a testbed. Besides it also viable to verify how fair the model captures the actual behaviour of the MOM in a measurement environment. Later, in order to achieve better performance we suggest adjustments on the behaviour of some MOM's components.

4.1. Model Description

The architecture of the IBM WebSphere MQ is depicted in Figure 1. The main components are the Queue Manager (QM) and the Message Control Agent (MCA). The MCA is responsible for transmitting messages between the queue managers. The communication channels (CC) interconnect several WebSphere MQ Clients or Servers. The CCs are an abstraction of the transport, network, link and physical layers. Figure 2 depicts a GSPN model of the WebSphere MQ architecture where we identify its main components. The subsets of places and transitions identified in Figure 2 are:

- App-S – it means a sending application with average message sending rate following a Poisson probability distribution function (pdf). It means that the interarrival times are Independent and Identically Distributed (IID) and are exponentially distributed with mean $\mathbf{1}/\lambda$.
- Queue-S – it models the main buffer at the WebSphere MQ closest to the sending application. The maximum number of messages that can be queued is modelled by the buffer size. Such system capacity is represented by the place **Size** and its initial marking **M**. Needless to say that when the place **Queued** reach its maximum capacity, the new arriving messages are dropped. Additionally, the model accounts the dropping rate. The transition **ServiceTime** represents the process of withdrawing messages from the place **Queued**. The service time is also a random variable IID exponentially distributed with mean $\mathbf{1}/\mu$.
- Network/MCA – it models the communications channel and the MCA component. The transition **Transmission** has a very small firing delay. We consider that such abstraction is very suitable since the network infrastructure in the testbed (as described in [1]) does not add extra delays. Moreover, the network infrastructure is oblivious, since there is no active queue management implemented.
- App-R – it means a receiving application with average receiving rate following a Poisson process with mean $\mathbf{1}/\lambda_2$;
- Queue-R – it models the main buffer at the WebSphere MQ closest to the receiving application. Following the same principles for the **Queue-S**, the maximum number of messages that can be queued is represented by the place **Size-R** and its initial marking **MR**. Transition **ServiceTimeR** represents the process of dequeuing messages from the place **QueuedR**. The service time is also a random variable IID exponentially distributed with mean $\mathbf{1}/\mu_2$.

It is worth stressing the quantities $\rho=\lambda/\mu$, $\rho_2=\lambda_2/\mu_2$ e $\rho_3=\lambda/\mu_2$ that define the stability condition in the sending application side, in the receiving application side and in the whole system, respectively.

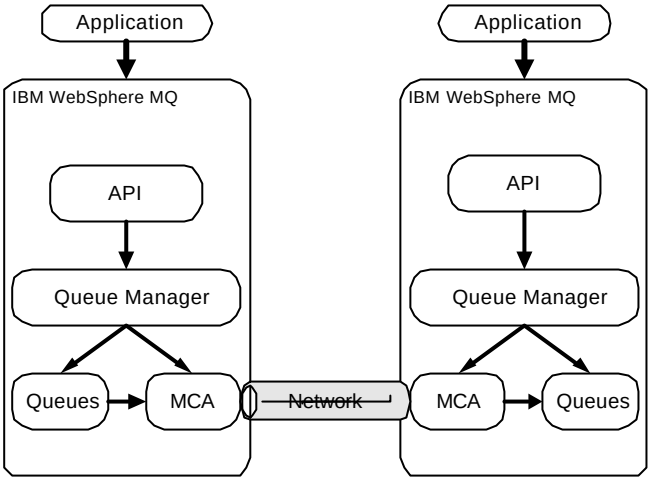

Figure 1 - IBM WebSphere MQ (MQ Series) Architecture

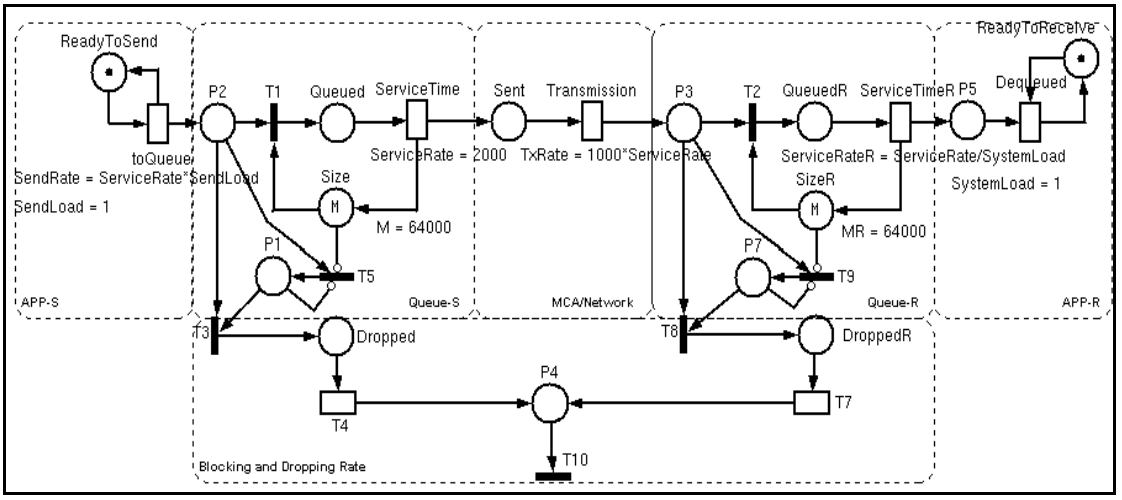

Figure 2. GSPN Base Model of the MQ Series

Some observations must be made to identify the level of correctness of the GSPN model related to the MOM's actual architectures. To do that, we compare our GSPN components and features with the test platform described in [1]. We present in the next paragraphs several differences and similarities for the MOM's architecture and also for the test scenario (*Testbed*).

Concerning to the parameterisation of the IBM WebSphere MQ, a system administrator has access to a large number of parameters to tune. Some of them could be set to optimise the MOM's performance for a particular workload. The main components that affect the performance behaviour are the main buffer size and the persistent buffer area. In our GSPN models, all parameters can be easily tuned. For instance, as presented in the next section, we have made efforts for finding an adequate buffer size.

The test scenarios described in Section 4 of [1] are open loop. The sending and receiving application can independently handle messages at different rates. The difference between such rates influences the main buffer optimal size. Similarly, the GSPN model captures all those features, since the transition firing delays in the sending and receiving application models are independently configured.

Related to workloads, the actual sender's performance behaviour is affected whenever the queue's length grows beyond an optimal threshold. Moreover, the sender's capacity is limited by the queue's processing rate. Our model can also mimic that behaviour by varying the system's workload (ρ). When ρ is below the unit the system is stable. The workload reproduces ratio of the queue's processing rate to the sender's capacity.

When the WebSphere MQ is configured to guarantee message delivery (persistent delivery QoS level), a full main buffer implies in dropping messages leaving the persistent area (log buffer). Hence, the queue manager and the MCA must simultaneously deal with both new messages arriving from the application and old messages buffered in the log files. In such situation, our second GSPN model (Figure 3) reproduces this persistent behaviour. Several adjustments may be carried out in the model in order to consider additional features, such as different priorities to withdraw new and old messages, sender's blocking and so on.

In the Testbed, two different states are rather important: sustainable (ideal) and unsustainable (poor performance). The metric Maximum Sustainable Throughput (MST) represents the throughput in the saturation point. We could easily map such states to the GSPN model as follows: sustainable, $\rho = 1$ and unsustainable, $\rho > 1$.

Some variables (factors) affect the main response variable (MST) in a real environment. The presence of Quality of Service (QoS) features that assure messages delivery is an important attribute. MOMs usually offer three QoS levels in their architectures: Non-Persistent (NP), Persistent (P) and Transactional (T). Accordingly, it is feasible to evaluate all three schemes using GSPN models. In this work we analysed the NP and P cases. System's configuration parameters, namely main and secondary buffer sizes and traffic intensity at the sending application side are other factors that influence performance. Some scenarios were set to analyse the impact of different buffer depths and sending application rates.

Capacity of the receiver, comprising the queue's processing rate and the number of threads, has a direct effect on MOM's performance. In such circumstances, our GSPN model shows its flexibility. It is possible to include multiple threads in the receiver through two strategies: the first one is represented by k concurrent stochastic transitions in order to take messages out the place that represents the main receiver buffer. The quantity k could represent the number of threads available to perform a respective job. The second possible scheme could define an Infinite Server Semantic (ISS) or a K-Server Semantic (KSS) for the transition responsible for taking messages out of the queue. In this work, we did not formally evaluate the impact of multithreads in the global performance. We just evaluated whether the GSPN model would adequately serve to such objectives. Additionally, as message lengths slightly affect the overall performance and as we considered messages in MOMs systems as tokens in GSPN models, their sizes could be regarded in the delay of all timed transitions

responsible for handling them. In this work, we did not evaluate the influence of different messages length.

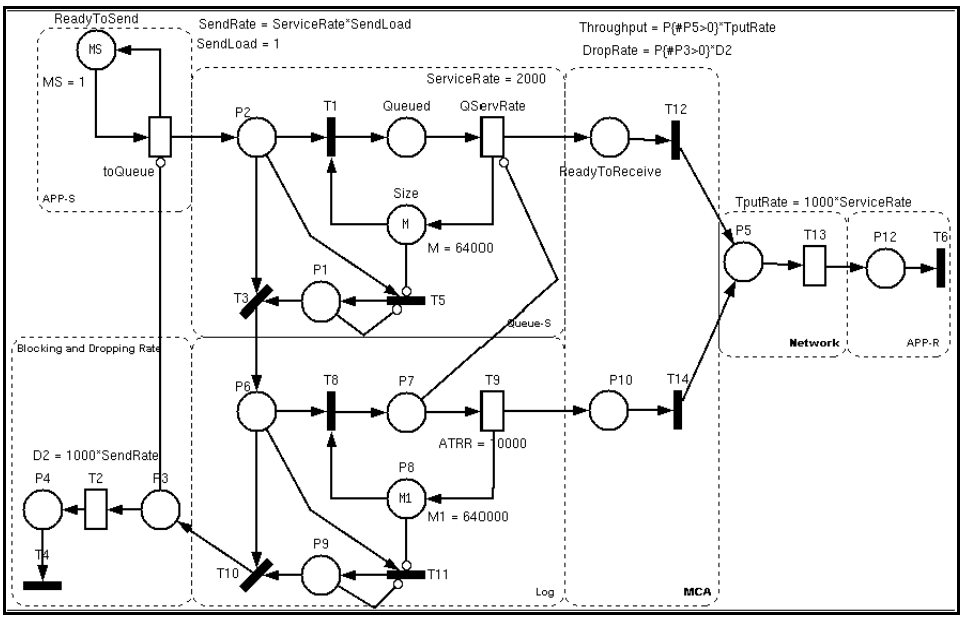

Figure 3 – Basic persistent GSPN model of the IBM WebSphere MQ

Finally, associated to persistent capability of MOM, some results presented in [1] showed that scenarios with persistence had poor performance, since messages have to be recorded to and read from disk, which access times is typically slower than that in random access memories. Accordingly, several adjustments in the GSPN model can be implemented for defining a persistent scheme. One strategy is to inhibit the timed transition in the main buffer as long as exists messages in the persistent log area, as depicted in Figure 4. A second scheme defines weights to transitions that represent the MCA. Such weights could define priorities to withdraw messages from the main buffer or from the log area. One should notice that the summation of these weights must be one. In the computer architecture field, access time is the average time interval between a storage peripheral device (e.g., disk drive or semiconductor memory) receiving a request to read or write a certain location and returning the value read or completing a write. So the ratio between access time in the main buffer and in the log buffer could be taken into account to set the weights for the timed transitions. We name this ratio as Access Time Ratio (ATR). Furthermore, this work suggests an alternative for improving the effective utilization of the system. In this context, we consider effective utilization as the ratio between the sender's sending messages rate and the actual receiver's receiving messages rate or alternatively the dropping messages rate. This work suggests messages in the log buffer to be queued up in the main buffer area once again, as depicted in the transition **T9** in Figure 4.

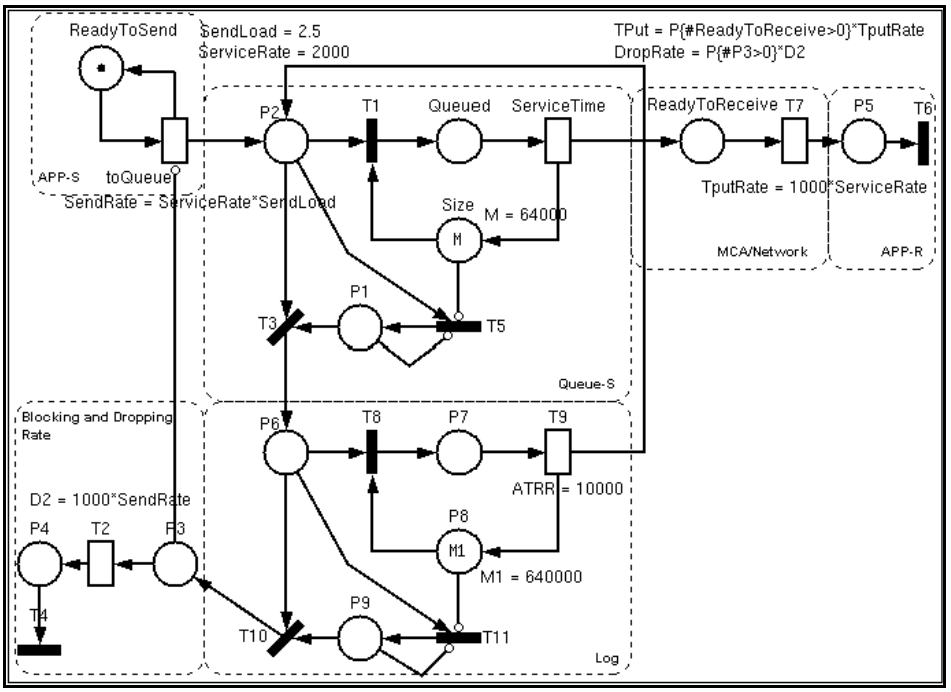

Figure 4 – Alternative GSPN Model

4.2 Simulation Results

In the previous section we presented three GSPN models for MOMs. The first one is the basic model (Figure 2) and the others are able to handle persistent messages delivery (Figure 3 and Figure 4). The second model is capable to reproduce the experiments results conducted in [1]. The alternative proposed model suggests a potential modification in the MOM's components in order to improve the overall performance behaviour. Hence, the experiments carried out using our models aim to analyse the performance behaviour of MOMs under different workload conditions and parameterisations of their constituents.

The performance metrics are described as follows:

- System's Throughput (in messages per second – mps units): this metric represents the average system's throughput, which could be measured in several locations. For instance, considering a receiving messages rate in the stability region ($\rho_2 = 1$), the throughput could be measured directly in the interface between the communication channel output and the receiver's input queue. Moreover the Little's Law [7] could analytically specify the response time of the system;
- Message Dropping Rate (MDR): this metric represents the overall system losses when it is in an unsustainable state. An expected result for ρ, ρ_2 and $\rho_3 = 1$ is the occurrence of no loss. It is important to emphasize that the main goal here is to minimize the dropping rate and to keep the sustainable throughput at maximum.

The results are presented in graphics charts. The GSPN model, metrics and level of the factors are fully identified on figures. Some results, which are precisely described, were obtained through the execution of a number of replications for each scenario in a batch mean simulation technique. In such cases, results represent the mean of observations [7]. The evaluation environment considered was based on a mix of Unix

platforms (Solaris and Linux) and also on a well-known GSPN tool named TimeNET 3.0 [14].

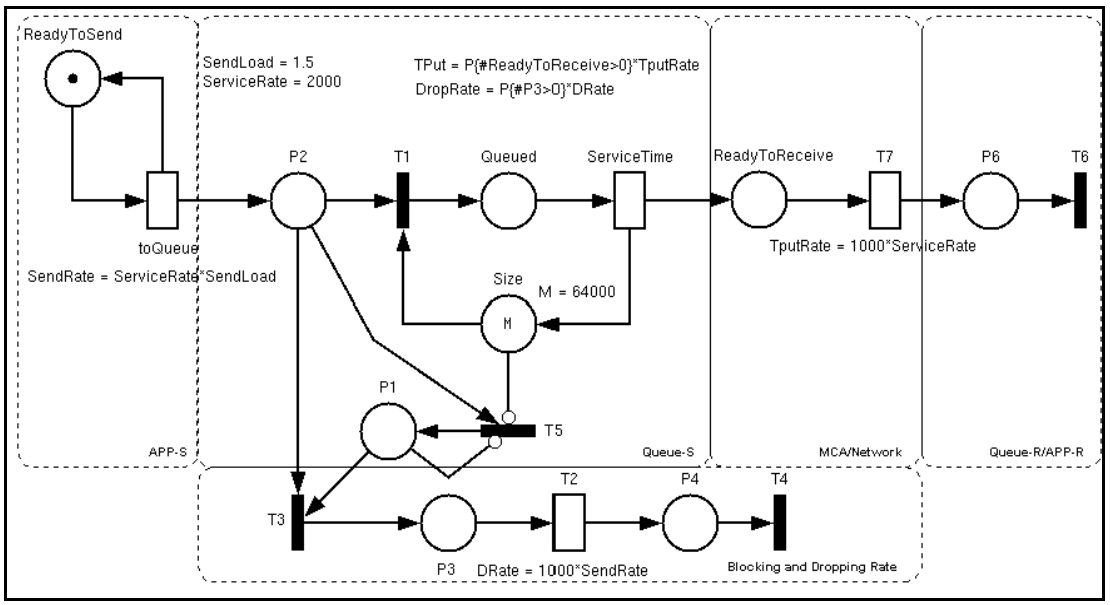

Figure 5 – Simplified GSPN model of the MQ Series

As we briefly described in Section 4.1, Figure 6 shows the simulation results for the GSPN model with a sender's sending rate limited at 4000 mps. We fixed the receiver's receiving rate at 2000 messages per second (mps). Our model (Figure 5) reasonably represents the scenario and precisely reproduces the experimental results (in [1]). In this scenario, results show typical performance behaviour of the MOM system configured with its default parameters. Up to the limit of 2000 mps (saturation point), the receiver's throughput increases linearly with the sender's rate. Beyond this point, messages are cumulated in the queue's receiver.

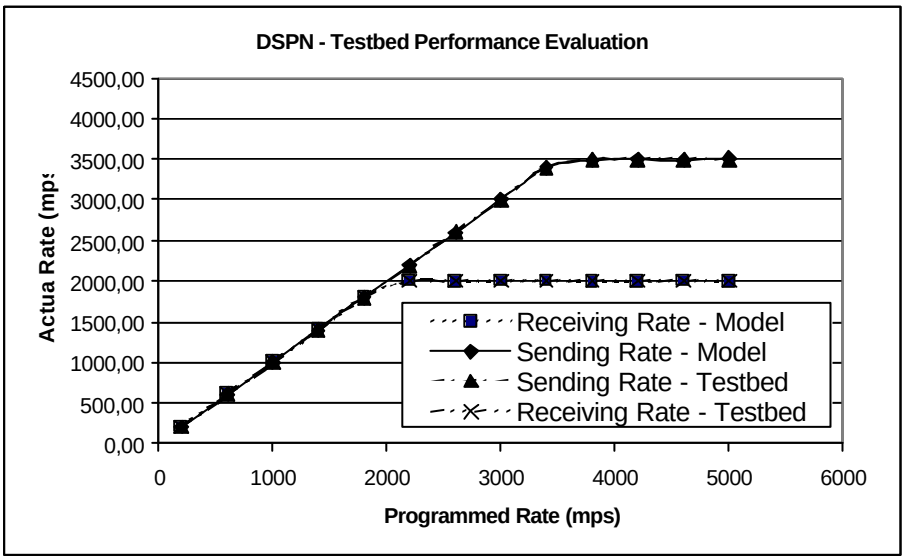

Figure 6 - Comparison of results - GSPN Vs. Testbed

Results presented in Figure 7 shows throughput and dropping rate of the non-persistent GSPN model when we vary the main buffer size, in a sender's stability region

(ρ = 1). We considered that tokens represent messages with 64 bytes length and sender's average sending rate follow a Poisson PDF with mean λ = 2000 mps. It can be seen the maximum throughput and the minimum dropping rate is achieved when the available main buffer depth is around 10^5. Such convergences point out that, depending on the sender or receiver's processing rate, it is useless to configure large buffer sizes. This result supports that a correct parameterisation of MOM systems could minimize losses due to lack of system's resources.

Figure 8 shows simulation results for the throughput metric in two circumstances. One curve pertains to the MOM with the persistent feature enabled (GSPN IBM WebSphere MQ) and the other concerns to the modified persistent GSPN model. The ratio between the firing times of the transitions that represent the main and the secondary buffer corresponds to the access times ratio (ATR). In modern computer architectures, a typical ATR is around 10^5. This means that the access time to withdraw a message from the main queue (RAM) is on the average 10.000 faster than the access time to remove it from the log buffer (files on disk). As commented in [1], we also observed the throughput substantially decreases with the incorporation of messages delivery guarantees. Hence, the trade-off between higher performance and message delivery assurance is an important design issue to be considered.

Our modified GSPN model puts back a logged message in the main buffer. It also conveys the sender to postpone the arriving of new messages when the secondary buffer reaches its maximum capacity. These features allow the system to achieve its maximum capacity faster than the original persistent GSPN, because it gives higher priority to withdraw new messages and occasionally remove old messages that were put back in the main buffer.

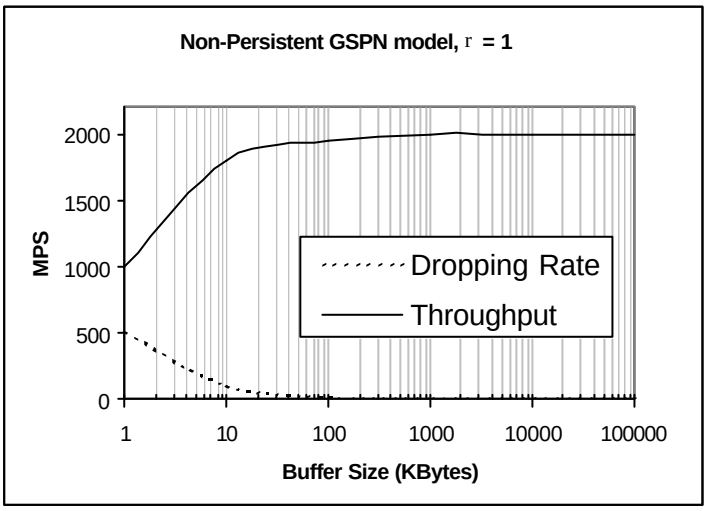

Figure 7 - Dropping Rate and Throughput, Basic Non-Persistent GSPN Model

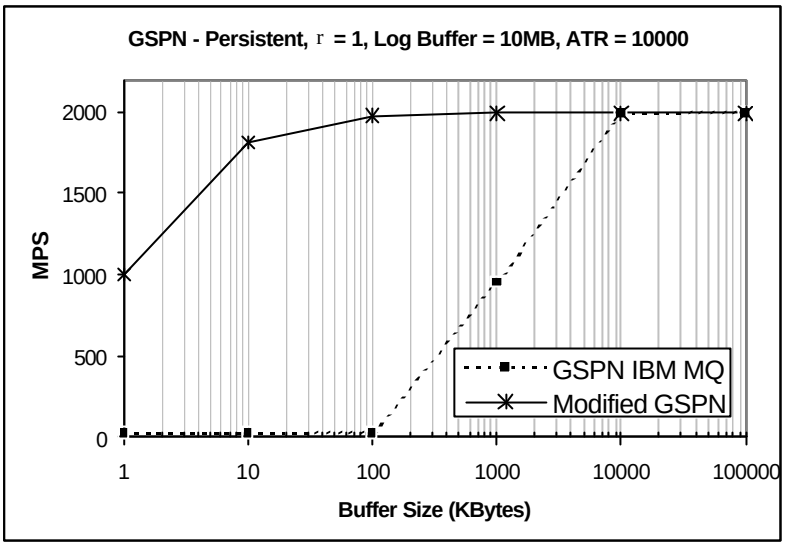

Figure 8 - Throughput vs Buffer Size - Persistent GSPN Models

Results presented in Figure 9 shows throughput and dropping rate of the non-persistent GSPN model when we vary the main buffer size, in a sender's instability region ($\rho > 1$). It is obvious that in such condition the sender quickly reaches its maximum capacity and the system start to drop messages. Moreover, results presented for the persistent GSPN are similar to the non-persistent model, since there will always be messages being deviate from the main buffer to the secondary one, which will soon reach its maximum capacity. Such convergences point out that, depending on the sender or receiver's processing rate, it is useless to configure large buffer sizes. Consequently if the main buffer is small this poor performance will aggravate. Similar to this result, Figure 10 shows throughput of two persistent GSPN model when we vary the main buffer size, in a sender's instability region ($\rho > 1$). The suggested modification in the basic persistent GSPN model still keeps the throughput at maximum. However it clearly does not solve the problem of high dropping rates.

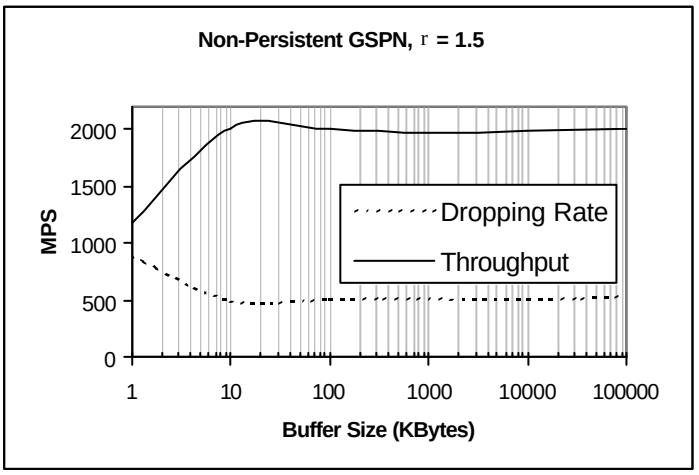

Figure 9 - Throughput and Dropping Rate vs. Buffer Size, Non-Persistent GSPN

Results previously presented use an ATR around 10^5. The following results show the effect of this factor in the metrics. Figure 11 validate our first impression that

the ATR factor has no influence on throughput in a stability region (ρ [1), since there will be rare occurrences of dropped messages to the log buffer. On the other hand, one should notice that in an extreme high workload (ρ = 2.5), the throughput only reaches its maximum if the ATR is around few units.

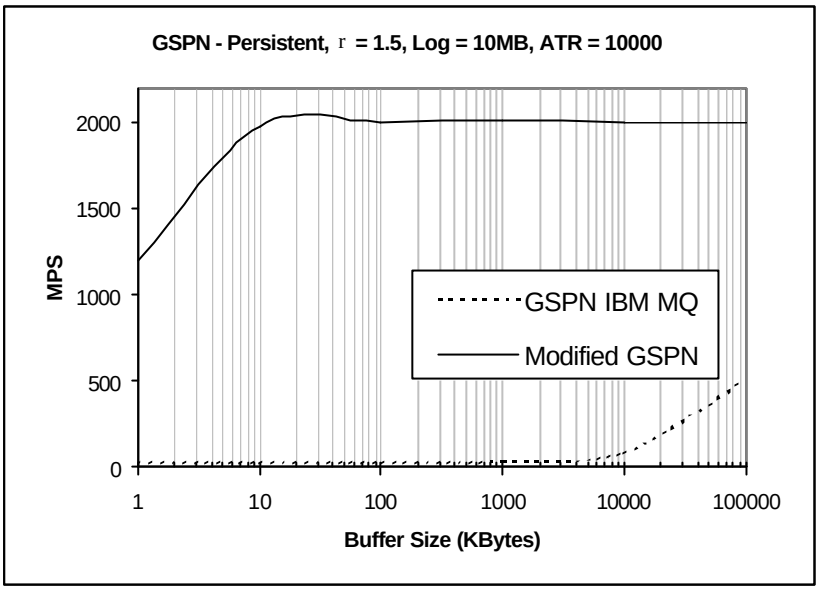

Figure 10 - Throughput vs. Buffer Size - Basic and Modified Persistent GSPN Models - r = 1.5

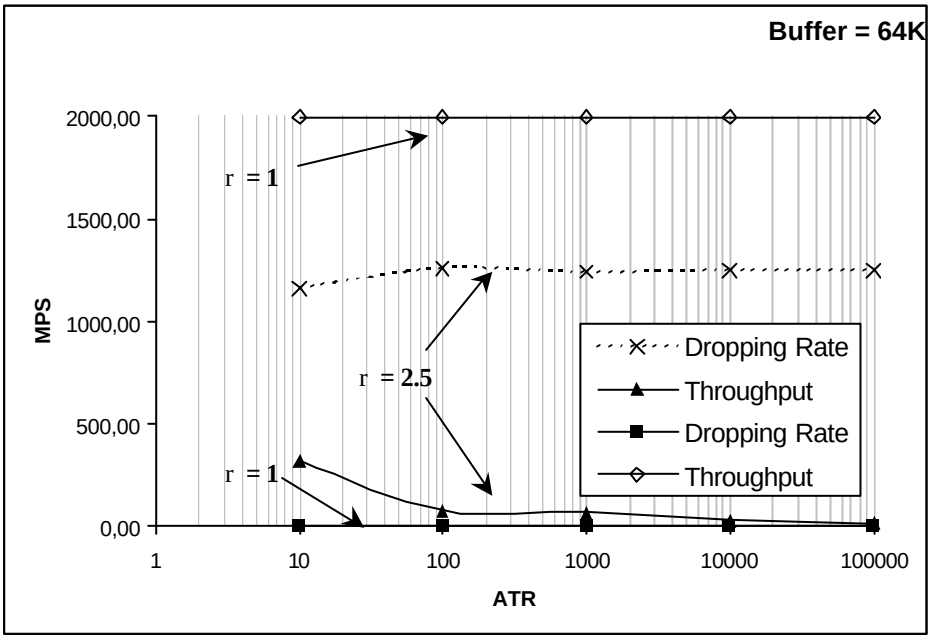

Figure 11 - Impact of ATR on Throughput and Dropping Rate, r = 1

Figure 12 shows the performance behaviour of the persistent GSPN model when we vary the workload. In this simulation, we considered a buffer size with capacity tuned by the previous results, i.e., around 10^5. In both curves, one should notice that throughput increases as the workload does and the absence of dropping messages, up to

the stability limit. As the workload grows, the system's throughput drastically decreases and the dropping rate exponentially increases. In the case of a lower ATR, slower decreasing throughput is apparent.

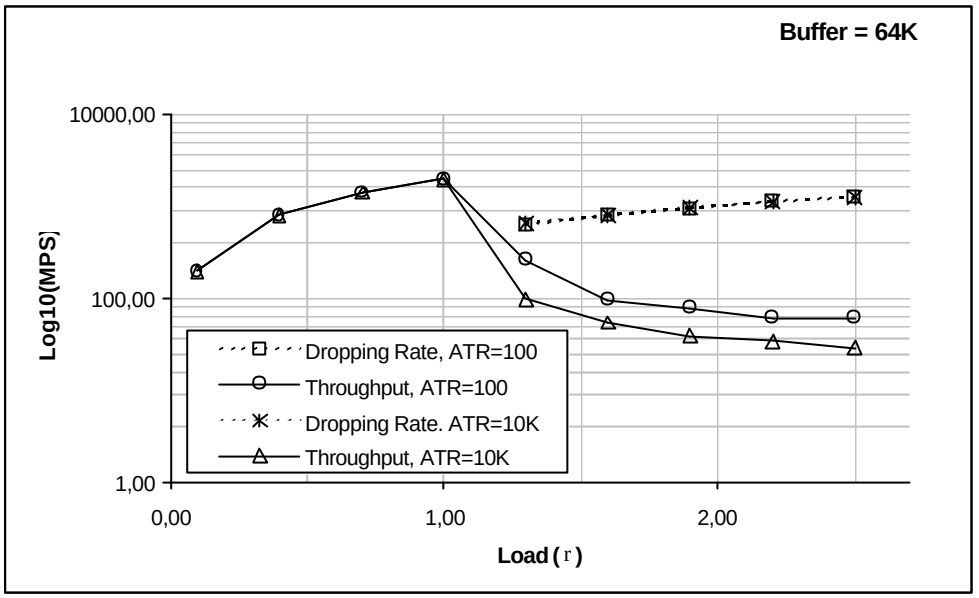

Figure 12 – Persistent GSPN Model, Throughput and Dropping Rate vs. Workload, ATR = 10^4 and 100; Buffer = 64K

5. Concluding Remarks and Further Developments

Middleware has a large diversity of features such as price, complexity, flexibility, and performance, where in each target application, different aspects are more important than others. The process of choosing the correct middleware technology refers first in deciding which qualities have more significance for applications [15]. However, a common requirement for almost all applications is the performance behaviour. Although several ways of evaluating the performance of Middleware are available, most scientific research papers in this area are mainly based on measurements in test environments. The main disadvantage of this approach is related to test scalability. Thus, the search for simulation and analytical models that allows complex system's performance behaviour reproduction has been received increasingly attention by the distributed systems scientific community.

In this paper Message-Oriented Middleware (MOM) was evaluated through Generalised and Stochastic Petri Nets (GSPN) based modelling. The model's accuracy could be evaluated through comparisons with measurements performed in a real test measurement environment. The GSPN model demonstrated to be very accurate and succeed in catching the main performance characteristics of a real MOM. Moreover, some performance analysis could be extended to verify the behaviour of some metrics while changing different factors and their respective levels. In general, results presented the ability of modelling and flexibility in evaluating MOM using GSPN.

The MOM's GSPN-based model is suitable to possible extensions, some of which are currently in progress. Hence, additional performance evaluation can be made in future works. One of them refers to the introduction of threads in the sender or

receiver side. This architectural component can be modelled as *n* stochastic and concurrent transitions that are used to withdraw messages from the place that represents the main sender's or receiver's buffer, where *n* is the amount of available threads. Alternatively, an Infinite-Server Semantic can be defined for transitions that withdraw messages from queues. Another possible extension could be the evaluation of transactional MOMs, with some kind of feedback in the model.

References

[1] Phong Tran, Paul Greenfield, "Behaviour and Performance of Message-Oriented Middleware Systems", Proceedings of the 2nd IEEE International Conference on Distributed Computing Systems Workshops (ICDCSW'02), 2002

[2] Istabrak Abdul-Fatah & Shikharesh Majumdar, "Performance of CORBA-Based Client-Server Architectures", IEEE Transactions on Parallel and Distributed Systems, Vol. 13, No. 2, , pg.111-126, February 2002.

[3] Michael Pang & Piyush Maheshwari, "Benchmarking Message-Oriented Middleware – TIB/RV vs. SonicMQ", 2002

[4] Te-Kai Liu, Amir Behroozi, Santhosh Kumaran, "A Performance Model for a Bussiness Process Integration Middleware", Proceedings of the IEEE International Conference on E-Commerce (CEC'03), 2003.

[5] Gross, D., Harris, C. M., "Fundamentals of Queueing Theory", 3rd Ed., John Wiley & Sons, 1998.

[6] Fernandes, Stenio F. L.; Kamienski, Carlos A.; Sadok, Djamel F. H., "Accurate and Fast Replication on the Generation of Fractal Network Traffic Using Alternative Probability Models". Conference on Performance and Control of Next Generation Communication Networks, part of the SPIE International Conference ITCom 2003, 7-11 September 2003 in Orlando, FL-USA.

[7] Jain, Raj, "The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modelling". John Wiley & Sons. 1991.

[8] Guruduth Banavar, Tushar Chandra, Robert Strom, and Daniel Sturman. A Case for Message Oriented Middleware. Lecture Notes in Computer Science, Vol. 1693, 1999.

[9] Sonic Software Corporation, "Getting Started with SonicMQ", Available by 10/2003.

[10] Mehdi Khorasani, "Middleware in Telecommunications", Lucent Technologies

[11] Tom Verdickt, Bart Dhoedt, Frank Gielen, Piet Demeester, "Modelling the performance of CORBA using Layered Queueing Networks", Proceedings of the IEEE 29th Euromicro Conference (EUROMICRO'03), p. 117, 09/2003

[12] IBM, Websphere MQ Planning Guide, www.ibm.com/software, acessado em Out/2003

[13] SonicMQ V5 Deployment Guide - Sonic Software Corporation, http://www.sonicsoftware.com/products/sonicmq/documentation/index.ssp, acessado em 10/2003.

[14] Zimmermann, Armin, "TimeNET: A Software Tool for the Performability Evaluation with Stochastic Petri Nets", Performance Evaluation Group, TU Berlin, June 2001.

[15] Vinoski, S., "The Performance Presumption", IEEE Internet Computing, April, 2003, pp. 88-91.

[16] Bernstein, Philip A. "Middleware: A Model for Distributed System Services", Communications of the ACM, Vol 39 (2), pp. 87-98, February, 1996.

[17] ISO/IEC/JTC1/SC21/WG7. Reference Model of Open Distributed Processing (Part I) – Overview. ISO 10476-1, July 1995.

[18] OMG. Common Object Request Broker Architecture: Core Specification (CORBA 3.0), December 2002.

[19] Rosenberry, W., Kenney, D. & Fisher, G. Understanding DCE. Ed.O'Reilly & Associates,Inc, 1993.