Analysis of the Message Waiting Time for the FioranoMQ JMS Server

Michael Menth and Robert Henjes
Department of Distributed Systems, Institute of Computer Science, University of Würzburg, Germany
Email: {menth,henjes}@informatik.uni-wuerzburg.de

***Abstract*— The Java messaging service (JMS) is a means to organize communication among distributed applications according to the publish/subscribe principle. If the subscribers install filter rules on the JMS server, JMS can be used as a message routing platform, but it is not clear whether its message throughput is sufficiently high to support large-scale systems. We perform measurements for the FioranoMQ JMS server and derive a simple model for its message processing time that takes message filters and the message replication grade into account. Then, we analyze the JMS server capacity and the message waiting time for various application scenarios. We show that the message waiting time is not an issue as long as the server throughput is sufficiently high. Finally, we assess the capacity of two different distributed JMS architectures whose objective is to increase the capacity of the JMS beyond the limit of a single server.**

I. Introduction

The Java messaging service (JMS) is a communication middleware for distributed software components. It is an elegant solution to make large software projects feasible and future-proof by a unified communication interface which is defined by the JMS API provided by Sun Microsystems [1]. Hence, a salient feature of JMS is that applications do not need to know their communication partners, they only agree on the message format. Information providers publish messages to the JMS server and information consumers subscribe to certain message types at the JMS server to receive a certain subset of these messages. This is known as the publish/subscribe principle. When messages must be reliably delivered only to subscribers who are presently online, the JMS in the persistent but non-durable mode is an attractive solution for the backbone of a large scale real-time communication applications. For example, some user devices may provide presence information to the JMS. Other users can subscribe to certain message types, e.g., the presence information of their friends' devices. For such a scenario, a high message routing platform needs filter capabilities and a high capacity to be scalable for many users. In particular, the throughput capacity of the JMS server should not suffer from a large number of clients or filters.

In this paper we investigate the maximum throughput of the FioranoMQ JMS server implementation [2] by measurement and derive a model for the message processing time depending on the number of installed filters and the replication grade of a message. The model is useful to predict the server throughput in practice we investigate different application scenarios. We study the message waiting time based on an $M/G/1-\infty$ approximation and perform a sensitivity analysis with respect to the variability of the message replication grade. The analysis shows that the message waiting time is low as long as the server throughput is sufficiently high since the message replication grade does not induce too much variance. Finally, we present two simple distributed architectures based on conventional JMS servers that increase the JMS capacity beyond the capacity provided by a single server and compare their usefulness for different parameter settings.

The paper is organized as follows. In Section II we present JMS basics, that are important for our study, and consider related work. Section III presents our test environment, measurement methodology, and results that we use to derive a an analytical model for the message service time. In Section IV we apply this model to predict the server throughput for various application scenarios, we calculate the distribution of the message waiting time, and compare the system throughput of two new distributed architectures. Finally, we summarize our work in Section V and give an outlook on further research.

This work was funded by Siemens AG, Munich. The authors alone are responsible for the content of the paper.

II. Background

In this section we describe the Java messaging service (JMS) and discuss related work.

A. The Java Messaging Service

Messaging facilitates the communication between remote software components. The Java Messaging Service (JMS) standardizes this message exchange. The so-called publishers generate and send messages to the JMS server, the so-called subscribers consume these messages – or a subset thereof – from the JMS server, and the JMS server acts as a relay node [3], which controls the message flow by various message filtering options. This is depicted in Figure 1. Publishers and subscribers rely on the JMS API and the JMS server decouples them by acting as an isolating element. As a consequence, publishers and subscribers do not need to know each other. The JMS offers several modes. In the persistent mode, messages are delivered reliably and in order. In the durable mode, messages are also forwarded to subscribers that are currently not connected while in the non-durable mode, messages are forwarded only to subscribers who are presently online. Thus, the server requires a significant amount of buffer space to store messages in the durable mode and it achieves a larger

throughput in the non-durable mode. In this study, we only consider the persistent but non-durable mode.

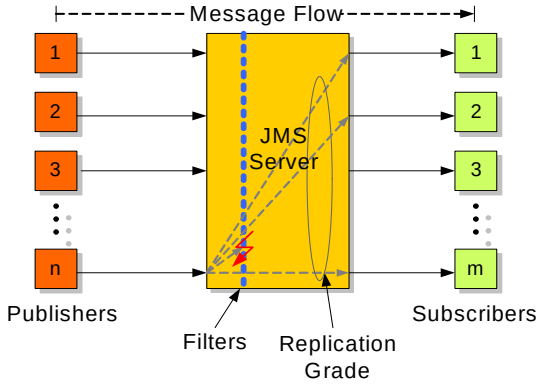

Fig. 1. The JMS server decouples publishers and subscribers.

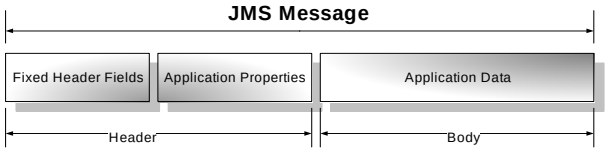

Fig. 2. JMS message structure.

Information providers with similar themes may be grouped together and publish to a so-called common topic; only those subscribers having subscribed for that specific topic receive their messages. Thus, topics virtually separate the JMS server into several logical sub-servers. Topics provide only a very coarse and static method for message selection. In addition, topics need to be configured on the JMS server before system start. Filters are another option for message selection. A subscriber may install a message filter on the JMS server, which effects that only the messages matching the filter rules are forwarded instead of all messages in the corresponding topic. Each subscriber has only a single filter. In contrast to topics, filters are installed dynamically during the operation of the server. A JMS message consists of three parts that are illustrated in Figure 2: the message header, a user defined property header section, and the message payload itself [1]. So-called correlation IDs are ordinary 128 byte strings that can be set in the header of JMS messages. Correlation ID filters try to match these IDs whereby wildcard filtering is possible, e.g., in the form of ranges like [#7;#13]. Several application-specific properties may be set in the property section of the JMS message. Application property filters try to match these properties. Unlike to correlation ID filters, a combination of different properties may be specified which leads to more complex filters with a finer granularity. After all, topics, correlation ID filtering, and application property filtering are three different possibilities for message selection with different semantic granularity and different computational effort.

B. Related Work

The JMS is a wide-spread and frequently used middleware technology. Therefore, its throughput performance is of general interest. Several papers address this aspect already but from a different viewpoint and in different depth.

The throughput performance of four different JMS servers is compared in [4]: FioranoMQ [2], SonicMQ [5], TibcoEMS [6], and WebsphereMQ [7]. The study focuses on several message modes, e.g., durable, persistent, etc., but it does not consider filtering, which is the main objective in our work. The authors of [8] conduct a benchmark comparison for the Sun OneMQ [9] and IBM WebsphereMQ. They tested throughput performance in various message modes and, in particular, with different acknowledgement options for the persistent message mode. They also examined simple filters but they did not conduct parametric studies, and no performance model was developed. The objective of our work is the development of such a performance model to forecast the maximum message throughput for given application scenarios. In [10] the memory requirements of different filtering algorithms for pub/sub systems were studied theoretically and experimentally. A proposal for designing a "Benchmark Suite for Distributed Publish/Subscribe Systems" is presented in [11] but without measurement results. The setup of our experiments is in line with these recommendations. General benchmark guidelines were suggested in [12] which apply both to JMS systems and databases. However, scalability issues are not considered, which is the intention of our work. A mathematical model for a general publish-subscribe scenario in the durable mode with focus on message diffusion without filters is presented in [13] and they are validated by measurements in [14]. In our work a mathematical model is presented for the throughput performance in the non-durable mode including filters and this model is validated by measurements. Several studies address implementation aspects of filters. A JMS server checks for each message whether some of its filters match. If some of the filters are identical or similar, some of that work may be saved by intelligent optimizations. This is discussed, e.g., in [15]. We perform measurements for the FioranoMQ with identical and different filters and both lead to the same results. Thus, FioranoMQ does not implement any optimization for several identical filters.

Apart from single server architectures, there are also distributed approaches like the one in [16] that intend to increase the overall scalability of the system concerning throughput and reliability. We also propose two distributed JMS server architectures to improve the system scalability but in contrast to this approach, our approach is based on off-the-shelf JMS components.

III. Measurement Results

In this section, we investigate the throughput of the FioranoMQ JMS server by measurements. First, we explain the experiment setup, give a summary of previous measurement results, and conduct parameter studies including filters to explore their impact on the JMS server throughput. Finally,

we present a simple model for the message processing time at the JMS server and validate them by our measurements.

A. Experiment Setup and Measurement Methodology

For reasons of comparability and reproducibility we accurately describe our testbed and our measurement methodology.

1) Testbed: Our test environment consists of five computers that are illustrated in Figure 3. Four of them are production machines and one is used for control purposes, e.g., controlling jobs like setting up test scenarios and starting measurement runs. The four production machines have a 1 Gbit/s network interface which is connected to one exclusive Gigabit switch. They are equipped with 3.2 GHz single CPUs and 1024 MB system memory. Their operating system is SuSe Linux 9.1 in standard configuration. To run the JMS environment we installed Java SDK 1.4.0, also in default configuration. The control machine is connected over a 100 Mbit/s interface to the Gigabit switch.

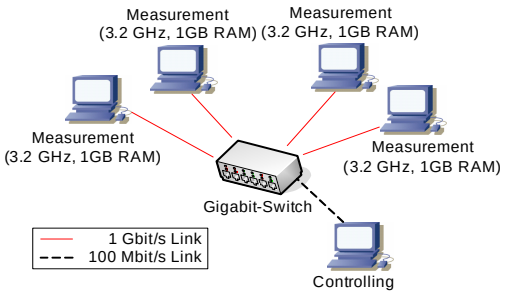

Fig. 3. Testbed environment.

We installed the FioranoMQ version 7.5 server components as JMS server software. We used the vendor's default configuration as delivered with the test version. Our publisher and subscriber test clients are derived from Fiorano's example Java sources for measurement purposes. Each publisher or subscriber is realized as a single Java thread, which has an exclusive connection to the JMS server component. A management thread collects the measured values from each thread and appends these data to a file in periodic intervals. In our experiments one machine is used as a dedicated JMS server, the publishers run on one or two exclusive publisher machines, and the subscribers run on one or two exclusive subscriber machines depending on the experiment. If two publisher or subscriber machines are used, the publisher or subscriber threads are distributed equally between them.

2) Measurement Methodology: Our objective is to measure the capacity of the JMS server. Therefore, we load it in all our experiments closely to 100% CPU load and verify that no other bottlenecks like system memory or network capacity exist on the server machine, i.e., that they have a utilization of at most 75%. The publisher and subscriber machines must not be bottlenecks, either, and they must not run at a CPU load larger than 75%. To monitor these side conditions, we use the Linux tool "sar", which is part of the "sysstat" package [17]. We monitor the CPU utilization, I/O, memory, and network utilization for each measurement run. Without a running server, the CPU utilization of the JMS server machine does not exceed 2%, and a fully loaded server must have a CPU utilization of at least 98%.

Experiments are conducted as follows. The publishers run in a saturated mode, i.e., they send messages as fast as possible to the JMS server. However, they are slowed down if the server is overloaded because publisher side message queuing is used. To save system processing resources during the measurement phase, all JMS messages that will be ever sent by the publisher are created in advance when the publisher test clients are started. For the same reason, all connections are established before measurements are taken. Each experiment takes 100 s but we cut off the first and last 5 s due to possible warmup and cooldown effects. We count the overall number of sent messages at the publishers and the overall number of received messages by the subscribers within the remaining 90 s interval to calculate the server's rate of received and dispatched messages. We call the corresponding rates the received and dispatched throughput and their sum the overall throughput. For verification purposes we repeat the measurements several times but their results hardly differ such that confidence intervals are very narrow even for a few runs.

B. Measurement Results

This paper focuses mainly on the investigation for the message waiting time based on our measurement results and a performance model. In [18] we performed extensive measurements according to the above described testbed and measurement methodology. We first summarize the various aspects of that study and review then the results which are important for this paper in more detail.

1) Summary on Previous Measurement Studies: We briefly summarize the experiments and their results from [**?**]. We investigated the maximum message throughput of the server depending on the number of publishers and subscribers. We found that a minimum number of 5 publishers must be installed to fully load the JMS server and so we conducted the following experiments with at least 5 publishers. When we increase the number of subscribers without filters, the messages are forwarded to all of them. If a message is dispatched to R different subscribers, it is replicated and sent R times by the JMS server and we call R the replication grade of the message. Another experiment showed that the message size has a significant impact on the message throughput. We used a default message body size of 0 bytes, i.e. the full information is contained in the message headers. We found that the message throughput suffers the least from topic filtering, followed by correlation ID filtering and application property filtering, and investigated complex AND- and OR-filter rules.

2) Joint Impact of the Number of Filters and the Message Replication Grade: We have learned from prior experiments that both the number of filters and the replication grade impact the JMS server capacity. In this section, we investigate their joint impact by measurements and present a simple model to forecast the server performance for a given number of filters

and for an expected replication grade. This model is validated by measurements.

a) Experiment Setup and Measurement Results: We set up experiments to conduct parameter studies regarding the number of installed filters and the replication grade R of the messages. We use one publisher and one subscriber machine. Five publishers are connected to the JMS server and send messages with correlation ID #0 or application property value #0 in a saturated way. Furthermore, $n+R$ subscribers are connected to the JMS server, R of them filter for correlation ID or application property attribute #0 while the other n subscribers filter for different correlation IDs. Hence, $n+R$ filters are installed altogether. This setting yields a message replication grade of R. We choose replication grades of $R \in \{1,2,5,10,20,40\}$ and $n \in \{5,10,20,40,80,160\}$ additional subscribers.

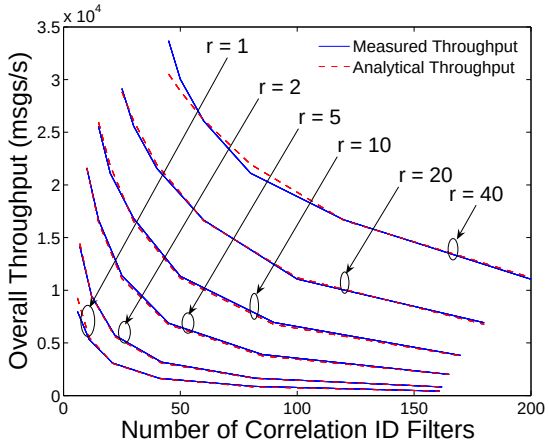

Fig. 4. Impact of the number of filters n_{fltr} and the message replication grade R on the overall message throughput in case of correlation ID filters – measurements and analytical data.

Figure 4 shows the the overall message throughput for correlation ID filters depending on the number of installed filters $n_{fltr}=n+R$ and on the replication grade R. The solid lines show the measured throughput. An increasing number of installed filters reduces obviously both the received and overall message throughput of the system and an increasing replication grade increases the overall system performance to a certain extent. Similar measurements are obtained for application property filtering. The basic performance behavior is the same, but the absolute overall message throughput is about 50% compared to the one of correlation ID filters. We get the same results for both experiments if all the n non-matching filters search for the same value, e.g. for #1, and if they look for different values, e.g. for #1, ..., #n. Thus, we cannot find any throughput improvement if equal filters are used instead of different filters [15].

b) A Simple Model for the Message Processing Time: We assume that the processing time of the JMS server for a message consists of three components. For each received message, there is

- a fixed basic time overhead t_{rcv} independently of filter installations.
- a fixed time overhead $n_{fltr} \cdot t_{fltr}$ caused by the JMS server while checking which different filters are matching. This value depends on the application scenario.
- a variable time overhead $R \cdot t_{tx}$ depending on the message replication grade R. It takes into account the time the server takes to forward R copies of the message.

This leads to the following mean $E[B]$ of the message processing time B.

$$E[B] = t_{rcv} + n_{fltr} \cdot t_{fltr} + E[R] \cdot t_{tx} \quad (1)$$

c) Validation of the Model by Measurement Data: The results in Figure 4 show the overall message throughput. Within time $E[B]$, one message is received and $E[R]$ messages are dispatched on average by the server. Thus, the received and overall throughput is given by $\frac{1}{E[B]}$ and $\frac{E[R]+1}{E[B]}$ and the latter corresponds to the measurement results in Figure 4. The parameters n_{fltr} and R for the message processing time B are known from the respective experiments. We fit the parameters t_{rcv}, t_{fltr}, and t_{tx} by a least squares approximation [19] to adapt the model in Equation (1) to the measurement results. The results are compiled in Table I for correlation ID and application property filters. Note that both filter types require different values for all parameters to approximate the respective experimental measurements by the model.

TABLE I
OVERHEAD VALUES FOR THE MODEL OF THE MESSAGE PROCESSING TIME IN EQUATION (1).

overhead type	$t_{rcv}(s)$	$t_{fltr}(s)$	$t_{tx}(s)$
corr. ID filtering	$8.52 \cdot 10^{-7}$	$7.02 \cdot 10^{-6}$	$1.70 \cdot 10^{-5}$
app. prop. filtering	$4.10 \cdot 10^{-6}$	$1.46 \cdot 10^{-5}$	$1.62 \cdot 10^{-5}$

We calculate the message throughput based on these values and Equation (1) for all measured data points, and plot the results with dashed lines in Figure 4. The throughput from our analytical model agrees very well with our measurements for all numbers of filters n_{fltr} and all replication grades R. Thus, if we know the the number of installed filters n_{fltr} on the JMS server and the mean $E[R]$ of the message replication grade in a certain application scenario, we have a model that allows the prediction of the average message processing time $E[B]$ and the server capacity in terms of message throughput.

IV. Analytical Performance Evaluation

Based on the performance model and parameters obtained in Section III, we investigate now the JMS server capacity in different application scenarios by a rough average calculation and the message waiting time by careful queuing theoretical observations. Finally, we compare design alternatives for distributed JMS systems regarding their capacity to illustrate the usefulness of our findings.

A. JMS Server Capacity

To get a feeling for capacity of the FioranoMQ JMS server, we investigate the mean message processing time depending on the number of filters and predict the server capacity.

1) Average Message Service Time: With Equation (1) it is clear that the message service time increases linearly with the number of filters. Figure 5 illustrates the mean for the message service time $E[B]$ depending on the number of filters n_{fltr} and the average replication grade $E[R]$. The results are shown for both correlation ID filtering and for application property filtering. For small values of n_{fltr}, the average message service time $E[B]$ is dominated by the average replication grade $E[R]$ but for large values of n_{fltr} the linear growth clearly dominates the influence of the message replication grade. Note that both the x- and the y-axis have a logarithmic scale. Thus, the service time for a message ranges over several orders of magnitude, which is due to different message replication grades, to the linear growth of $E[B]$ with n_{fltr}, and to filter type specific values of t_{rcv}, t_{fltr}, and t_{tx}. Hence, it is strongly application scenario specific.

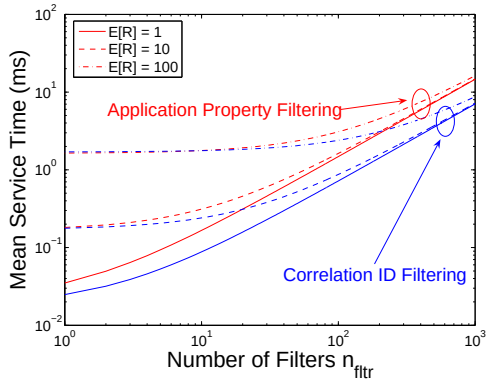

Fig. 5. Impact of the number of filters n_{fltr}, the average replication grade $E[R]$, and the filter type on the average message service time $E[B]$.

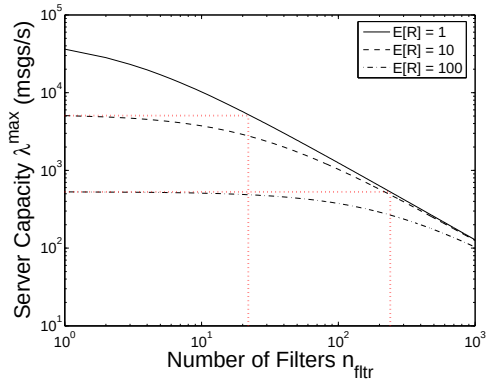

Fig. 6. Impact of the number of filters fn_{fltr} and the average replication grade $E[R]$ on the server capacity λ^{max} for a maximum server CPU utilization of $\rho = 90\%$.

2) Server Capacity: We define the server capacity by the maximum supportable load in terms of messages per second. If we allow a server CPU utilization of ρ, we can compute the server capacity in terms of received message throughput by

$$\lambda^{max} = \frac{\rho}{E[B]} \tag{2}$$

Figure 6 shows the server capacity for a maximum server CPU utilization of 90% for the same application scenarios like above. Figure 6 shows the server capacity λ^{max} depending on the same parameters like above but for the sake of clarity we omitted the results for application property filtering. Like the service time, the server capacity ranges also over several orders of magnitude due to Equation (2). It is obvious that the server capacity decreases both with an increasing number of filters n_{fltr} and with an increasing average replication grade $E[R]$. Filters protect the subscribers from undesired messages, they reduce the replication grade which limits the network traffic and improves potentially the server capacity. However, the latter objective is not always achieved. This is also shown in Figure 6: A message replication grade of $E[R] = 10$ (100) without filters effects the same capacity reduction like a message replication grade of $E[R] = 1$ and $n_{fltr} = 22$ (240) filters.

This leads to the question: when should a filter be applied to increase the server throughput? We consider an information consumer q that has installed n^q_{fltr} filters on the server. Furthermore, we assume that these filters receive the proportion p^q_{match} of all messages. On the one hand, the filters increase the message processing time by $n^q_{fltr} \cdot t_{fltr}$ but on the other hand, they reduce it by $(1 - p^q_{match}) \cdot t_{tx}$. Thus, these filters increase the server capacity if the following inequality holds.

$$n^q_{fltr} \cdot t_{fltr} < (1 - p^q_{match}) \cdot t_{tx} \tag{3}$$

Taking the values of Table I into account, a single or two correlation ID filters ($n^q_{fltr} \in \{1,2\}$) should be used if their match probability is smaller than 58.7% or 17.4%, respectively. Three or more filters per consumer slow down the server more than forwarding any message if no filters are set. A single application property filter ($n^q_{fltr} = 1$) should be used if its match probability is smaller than 9.9%. Like above, two or more filters per consumer cannot lead to a capacity increase of the JMS server. However, filters are primarily used to protect the consumers against too many unwanted messages and the network against overload.

B. Analysis of the Message Waiting Time

The objective of this section is the investigation of the message waiting time. We model first the JMS server by a simple queueing system and discuss various distribution models for the message replication grade which impacts the variability of the service time. Then, we study the mean, the distribution, and in particular the 99% and the 99.99% quantile of the message waiting time depending on the average server utilization.

1) A Simple Queuing Model for JMS Servers: With our version of FioranoMQ, the major part of the messages are queued at the publisher site due to a kind of push-back mechanism. As a consequence, we did not observe any message loss due to

buffer overflow at the JMS server. In our experiments, we used permanently sending publishers that were only slowed down by the push-back mechanism of the JMS server. However, in reality, the arrival process is stochastic, i.e., the publishers do not send in a saturated manner. If the JMS server is not overloaded and if its message buffer is large enough to absorb all arriving messages, we can well approximate the complex overall system by a single message queue at the JMS server site. This is depicted by Figure 7. The arrival rate $\lambda = \sum_{0 \le i < n} \lambda_i$ for that queue is the sum of the message rates λ_i from all publishers.

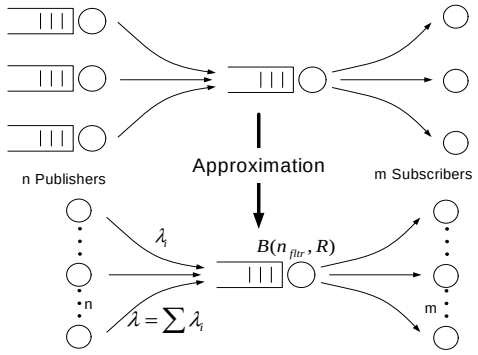

Fig. 7. A simple queueing model for a JMS server: $M/GI/1-\infty$.

Furthermore, we assume a Poisson model for the arrival process in the busy hour, i.e., the inter-arrival times are exponentially distributed and the message arrival rate is denoted by λ. This is a reasonable assumption since technical processes are often triggered by human beings. Messages are served sequentially by the server with their processing time B. This random variable has a general distribution. Thus, we can model the system by an $M/G/1-\infty$ queue. The first and second moment of the message waiting time in this queueing system is given by

$$E[W] = \frac{\lambda \cdot E[B^2]}{2 \cdot (1-\rho)} \quad (4)$$

$$E[W^2] = 2 \cdot E[W]^2 + \frac{\lambda \cdot E[B^3]}{3 \cdot (1-\rho)} \text{ with} \quad (5)$$

$$\rho = \lambda \cdot E[B] \quad (6)$$

being the utilization of the server [20].

2) Model for the Message Service Time: The formulae for the first two moments of the message waiting time (Equations (4) and (5)) require the first three moments of the message service time. The service time B for a message is composed of a constant part $D = t_{rcv} + n_{fltr} \cdot t_{fltr}$ and a variable part $V = R \cdot t_{tx}$ such that the first three moments can be calculated by

$$E[B] = E[D+V] = D + E[R] \cdot t_{tx} \quad (7)$$

$$E[B^2] = E[(D+V)^2] = D^2 + D \cdot t_{tx} \cdot E[R] + t_{tx}^2 \cdot E[R^2] \quad (8)$$

$$E[B^3] = E[(D+V)^3] = D^3 + 3 \cdot D^2 \cdot t_{tx} \cdot E[R] + 3 \cdot D \cdot t_{tx}^2 \cdot E[R^2] + t_{tx}^3 \cdot E[R^3] \quad (9)$$

To conduct a parameter study of the waiting time distribution depending on the mean $E[B]$ of the service time B and its coefficient of variation

$$c_{var}[B] = \frac{\sqrt{E[B^2] - E[B]^2}}{E[B]}, \quad (10)$$

we calculate the required $E[R]$ from Equation (7), and use $E[R]$ and Equation (8) to calculate $E[R^2]$. Depending on the appropriate model for the message replication grade R, we get $E[B^3]$ by using Equation (9) and the third moment of the respective distribution for the replication grade. In the following, we discuss various distributions to model the replication grade R.

a) Deterministic Distribution: If the replication grade is constant, say r, the distribution of the message processing time B is also deterministic and its coefficient of variation is $c_{var}[B] = 0$. Furthermore, the second and third moments of the message replication grade are

$$E[R^2] = E[R]^2 \quad (11)$$

$$E[R^3] = E[R]^3. \quad (12)$$

This model is very static and probably not appropriate to characterize real world scenarios.

b) Scaled Bernoulli Distribution: With a probability of p_{match}, a message is forwarded by all n_{fltr} filters and with a probability of $1-p_{match}$, the message is forwarded not at all. This can be modelled by a scaled Bernoulli distribution. The corresponding first two moments are

$$E[R] = p_{match} \cdot n_{fltr} \quad (13)$$

$$E[R^2] = p_{match} \cdot n_{fltr}^2 \quad (14)$$

The model parameters can be calculated vice-versa by $n_{fltr} = \frac{E[R^2]}{E[R]}$ and $p_{match} = \frac{E[R]}{n_{fltr}}$. Furthermore, the third moment is

$$E[R^3] = \frac{E[R^2]^2}{E[R]}. \quad (15)$$

We are interested in the coefficient of variation $c_{var}[B]$ of the message service time which is based on a message replication grade which is distributed according to this scaled Bernoulli distribution. We calculate it using Equations (10), (7), and (8). Figure 8 shows $c_{var}[B]$ depending on the number of filters n_{fltr}, the match probability p_{match}, and the filter type. The coefficient of variation $c_{var}[B]$ converges for an increasing number of filters to values that depend on p_{match} and the filter type. The coefficient of variation is at most $c_{var}[B] = 0.65$ and we cannot find any larger values for any other parameters of p_{match}.

c) Binomial Distribution: The scaled Bernoulli distribution is probably not realistic enough to model the distribution of the message replication grade. Now, we assume that the n_{fltr} filters match messages independently of each other with a probability of p_{match}. Then, the resulting replication grade follows a Binomial distribution:

$$P(R = k) = \binom{n_{fltr}}{k} \cdot p_{match}^k \cdot (1 - p_{match})^{n_{fltr}-k}. \quad (16)$$

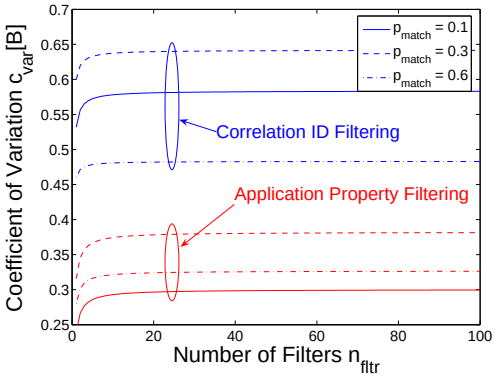

Fig. 8. Impact of the number of filters n_{fltr} and the match probability p_{match} on the coefficient of variation $c_{var}[B]$ of the message processing time B for a replication grade R distributed according to a scaled Bernoulli distribution.

Furthermore, the second and third moments [21] are

$$E[R^2] = n_{fltr} \cdot p_{match} \cdot (1 - p_{match}) \quad (17)$$

$$E[R^3] = E[R]^2 - E[R^2] - E[R] \cdot E[R^2] + 2 \cdot \frac{E[R^2]^2}{E[R]} \quad (18)$$

We conduct the same study like above and observe in Figure 9 that the coefficient of variation $c_{var}[B]$ decreases quickly for an increasing number of filters n_{fltr} to values of 0.064 and 0.033, which depends on the filter type.

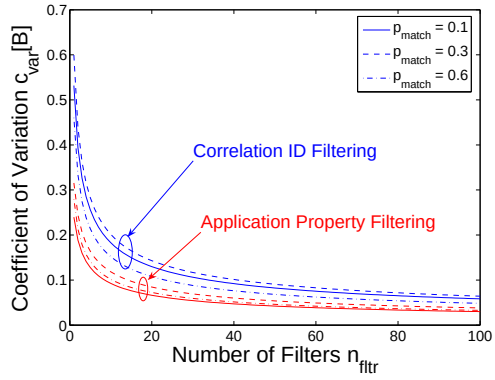

Fig. 9. Impact of the number of filters n_{fltr} and the match probability p_{match} on the coefficient of variation $c_{var}[B]$ of the message processing time B for a replication grade R distributed according to a Binomial distribution.

After all, the second moment of the service time is bound by Equation (8) and the second moment of the replication grade (cf. Equations (11), (14), and (17), respectively). Realistic coefficients of variations of the message service time lie between 0 and 0.2 and coefficients larger than 0.65 are impossible. Therefore, we work in the following exemplarily with the values 0, 0.2, and 0.4 because only they cover realistic scenarios.

3) Average Message Waiting Time: The average message waiting time at the JMS server can be calculated with Equation (4). Figure 10 shows it depending on the server utilization ρ in a specific application scenario with $n_{fltr} = 100$ correlation ID filters and a constant replication grade of $R = 1$. The left y-axis shows the corresponding waiting time in ms. It is a trivial result that the average waiting time $E[W]$ increases with ρ. We can generalize the result by indicating the waiting time as a multiple of the average message processing time $E[B]$ on the right y-axis, which also approximates the mean queue length in packets. Based on this normalized y-axis, we can easily compare the average message waiting time $E[W]$ from different application scenarios that have different means $E[B]$ and coefficients of variations $c_{var}[B]$. Figure 10 illustrates that the mean waiting time is sensitive to the coefficient of variation of the message processing time B and that it increases with $c_{var}[B]$. Note that the normalized diagram in Figure 10 provides also a lookup table for the average message waiting time $E[W]$ in any application scenario with a matching coefficient of variation $c_{var}[B]$.

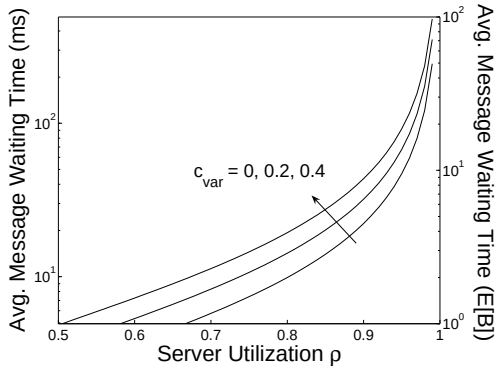

Fig. 10. Impact of the server utilization ρ and the coefficient of variation $c_{var}[B]$ of the message service time on the average message waiting time $E[W]$.

4) Message Waiting Time Distribution: In addition to the mean of the waiting time, its distribution is of interest. According to the $M/G/1-\infty$ queuing formulae, the waiting time probability for a message is $p_w = \rho$. With Equations (4) and (5) we know the first and second moment of the message waiting time such that we can calculate the first and second moment of the waiting time W_1 regarding only delayed calls by

$$E[W_1] = \frac{E[W]}{\rho}, \qquad E[W_1^2] = \frac{E[W^2]}{\rho} \quad (19)$$

The Gamma distribution has a positive range and can be viewed as the continuation of the exponential and Erlang distribution for coefficients of variations different from $c_{var}[X] = \frac{1}{\sqrt{k}}$, $k \in \mathbb{N}$ [22]. We approximate the waiting time distribution of the delayed calls $P(W_1 \leq t)$ by fitting their two parameters $\alpha = \frac{1}{c_{var}[W_1]}$ and $\beta = \frac{E[W_1]}{\alpha}$. Thus, we get the waiting time distribution regarding all calls by

$$P(W \leq t) = (1-\rho) + \rho \cdot P(W_1 \leq t). \quad (20)$$

This Gamma-approximation is exact for an exponentially distributed service time and leads to very good approximation results for other service time distributions [23].

Figure 11 shows the complementary distribution function of the message waiting time W for a server utilization of $\rho = 0.9$ and for a coefficient of variation of $c_{var}[B] = 0$, 0.2, and 0.4 on a normalized x-axis. The distribution functions are clearly shifted towards larger waiting time values with increasing $c_{var}[B]$, which is consistent with the results obtained in Section IV-B.3. The deterministic, the scaled Bernoulli, and the Binomial distribution coincide for $c_{var}[B] = 0$ and lead, therefore, to the same waiting time distribution of the messages. Furthermore, we can hardly see any difference between the waiting time distribution function for the binomial and the Bernoulli distribution of the replication grade R. Thus, we can neglect the exact distribution type of the message service time and work with its first two moments instead. In the following, we assume a messages service time based on a binomially distributed message replication grade R.

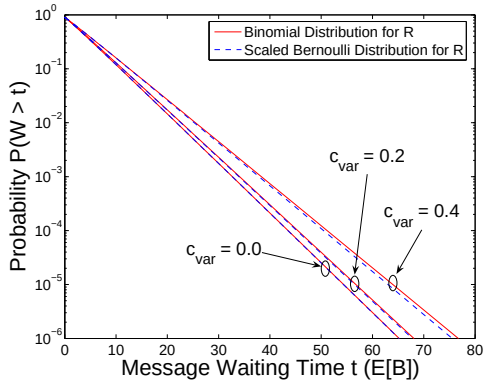

Fig. 11. Impact of the coefficient of variation $c_{var}[B]$ and the distribution type of the message replication grade R on the complementary distribution function of the message waiting time W for a server utilization of $\rho = 0.9$.

5) Message Waiting Time Quantile: The p-quantile or p-percentile $Q_p[W]$ specifies the lowest duration for which $P(W \leq Q_p[W]) \geq p$ holds. It says "$p \cdot 100\%$ of all messages wait shorter than $Q_p[W]$" and yields thereby a "quasi upper bound" on W if p is large. Figure 12 shows the 99% and 99.99% quantile of the waiting time on a normalized y-axis depending on the server utilization ρ and the coefficient of variation $c_{var}[B]$ of the message service time. The 99.99% quantile of the waiting time is substantially larger than the 99% quantile. The quantiles increase with the server utilization ρ and they are substantially larger than the means of the waiting time $E[W]$ in Figure 10. The impact of the coefficient of variation $c_{var}[B]$ is notable but the impact of the server utilization ρ is much larger since the considered coefficients of variation are all quite small. If we limit the server utilization to $\rho = 0.9$, the message waiting time is less than $50 \cdot E[B]$, i.e., a waiting time of $50 \cdot E[B]$ is not exceeded with a probability of 99.99%. With that probability a maximum waiting time of at most 1 s is guaranteed as long as $E[B]$ is smaller than 20 ms[1]. However, in this case, the maximum server capacity is only $\lambda_{\rho=0.9}^{max} = 45$ messages per second which is very low. Hence, at a server utilization of $\rho = 0.9$ or less, the message waiting time is not an issue provided that 1 s is tolerable but a server capacity of 45 msgs/s is not tolerable. Thus, if a sufficiently high throughput is achieved, the waiting time is small. Therefore, we neglect the waiting time in the next section.

[1]If the average replication grade is $E[R] = 1$ in the above scenario, up to 1369 or 2845 filters may be installed on the JMS server for application property or correlation ID filtering, respectively.

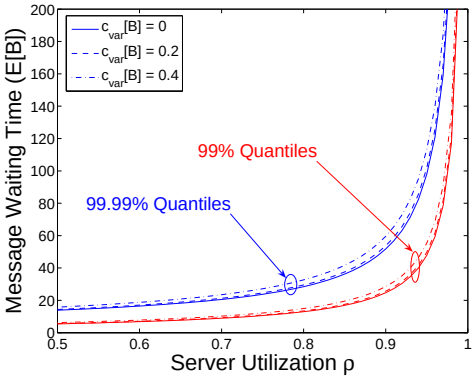

Fig. 12. Impact of the server utilization ρ and the coefficient of variation $c_{var}[B]$ of the message service time on the 99% and 99.99% quantiles of the message waiting time.

C. Performance Comparison of Distributed JMS Server Architectures

The capacity of a JMS server is bounded by the performance of its CPU. If it does not suffice to support a certain message rate from n publishers to m subscribers, a distributed architecture might be useful to alleviate the problem. We consider two basically different simple architectures: publisher-side JMS server replication and subscriber-side JMS server replication.

1) Publisher-Side JMS Server Replication (PSR): With publisher-side JMS server replication (PSR), each publisher has its own local JMS server for which subscribers can register. The concept is visualized in Figure 13. Each publisher-side $M/G/1-\infty$ system supports a message rate λ_i and their average message replication grade is $E[R_i]$. Since the messages are filtered already at the publishers, the traffic load imposed on the network interconnecting publishers and subscribers is $\sum_{0 \leq i < n} \lambda_i \cdot E[R_i]$.

A drawback of this distributed PSR architecture is the fact that all subscribers have to register in parallel for n JMS servers at distributed publisher sites instead of to a single one. This disturbs the elegant communication interface of JMS over a single server. Thus, additional entities must be introduced to allow a transparent communication like with a single server, but this is not scope of this paper.

2) Subscriber-Side JMS Server Replication (SSR): With subscriber-side JMS server replication (SSR), each subscriber has its own JMS server for which publishers can register. The concept is visualized in Figure 14. Since the messages are filtered only at the subscribers, the message rate for each subscriber-side $M/G/1-\infty$ system is $\lambda = \sum_{1 \leq i \leq n} \lambda_i$. Thus, the overall traffic carried in the network is $m \cdot \lambda$. Since m is an upper bound on R_i, SSR produces significantly more traffic in the network than PSR. Like with PSR, the elegant

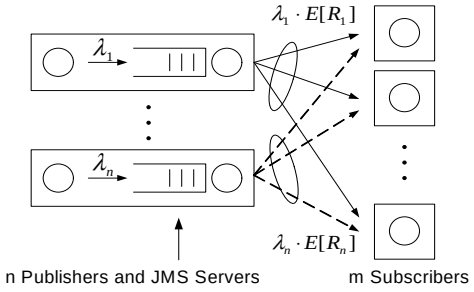

Fig. 13. Publisher-side JMS server replication (PSR).

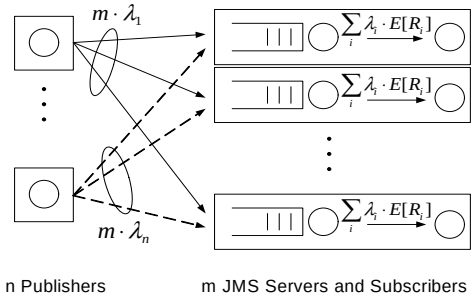

Fig. 14. Subscriber-side JMS server replication (SSR).

communication interface of JMS is also compromised by the SSR architecture because every publisher needs to multicast its messages to all JMS servers at m different subscriber sites instead of to a single one. However, this problem is not our present concern.

3) Capacity Comparison of PSR and SSR: For the performance comparison of the both architectures we consider the following environment. All nodes have the same computation power. In particular, we assume that they have the same capacity as the machines in our experiments in Section III because our numerical study relies on the values t_{rcv}, t_{fltr}, and t_{tx} that were obtained for these machines. Furthermore, the message rates λ_i of all publishers are equal and the average replication grades $E[R_i]$ for their messages are the same such that we can denote them uniformly by $E[R]$. In addition, each subscriber has $n_{fltr} = 10$ different filters.

For PSR, the capacity of the distributed JMS system $\lambda_{PSR}^{max} = n \cdot \min_{1 \le i \le n}(\lambda_i^{max})$ is the n-fold multiple of the minimum of all individual JMS server capacities λ_i^{max}. Similarly to Equation (2), it can be calculated under the above stated assumptions by

$$\lambda_{PSR}^{max} = \rho \cdot n \cdot \left(t_{rcv} + m \cdot n_{fltr} \cdot t_{fltr} + E[R] \cdot t_{tx}\right)^{-1} \qquad (21)$$

Thus, the system capacity depends on n and m and is thereby application scenario specific.

In case of subscriber-side JMS server replication, the capacity of the distributed JMS system $\lambda^{max} = \min_{0 \le i < m}(\lambda_i^{max})$ is the minimum of all individual JMS server capacities λ_i^{max}. It can be calculated under the above stated assumptions by

$$\lambda_{SSR}^{max} = \rho \cdot \left(t_{rcv} + n_{fltr} \cdot t_{fltr} + E[R] \cdot t_{tx}\right)^{-1} \qquad (22)$$

In contrast to λ_{PSR}^{max}, the expression for λ_{SSR}^{max} is independent of n and m.

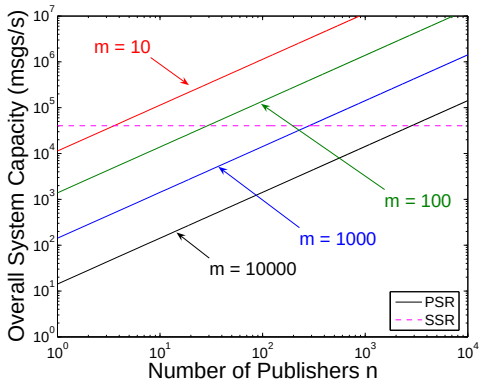

Fig. 15. Capacity comparison for publisher- and subscriber-side JMS server replication for a server utilization of $\rho = 0.9$, an average replication grade of $E[R] = 1$, m subscribers, and correlation ID filtering.

Figure 15 illustrates the impact of the parameters n and m on the capacities λ_{PSR}^{max} and λ_{SSR}^{max} of both distributed JMS systems. The results are calculated for an average replication grade of $E[R] = 1$, a maximum server utilization of $\rho = 0.9$, and correlation ID filtering. The capacity λ_{SSR}^{max} for SSR yields a horizontal line since it is independent of the parameters n and m. The capacity for PSR increases linearly with n and decreases about reciprocally for large values of m. PSR outperforms SSR for medium or large values of n and for small or medium values of m. Note that a large m can reduce the capacity of a single JMS server so much that waiting time problems arise. For example, for $m = 10^4$ and a large n the distributed system has still a large capacity but the capacity of a single publisher-side server is only 7 msgs/s leading to average waiting times of 1 s and to 99.99% quantiles of 10 s. We get similar results for application property filtering.

The capacity lines in Figure 15 intersect where both Equations (21) and (22) yield the same results. Thus, we follow that PSR outperforms SSR if the following inequality holds

$$n > \frac{t_{rcv} + m \cdot n_{fltr} \cdot t_{fltr} + E[R] \cdot t_{tx}}{t_{rcv} + n_{fltr} \cdot t_{fltr} + E[R] \cdot t_{tx}}. \qquad (23)$$

It gives a recommendation under which circumstances PSR or SSR should be implemented to cope with a large number of publishers or subscribers.

After all, PSR achieves system capacity scalability with respect to an increasing number of publishers, but the capacity degrades with an increasing number of subscribers. In contrast, SSR provides system capacity scalability for an increasing number of subscribers but its capacity does not scale with an increasing number of publishers. Hence, neither architecture yields a viable solution for the general scalability of the capacity of JMS servers. Therefore, we are working currently on such a solution.

V. Conclusion

In this work, we have investigated the throughput performance of the FioranoMQ JMS server. We set up a testbed

to measure the throughput for application scenarios with different message replication grade R, different numbers of filters n_{fltr}, and different filter types. From these results, we derived a simple model for the message processing time which provided the base for the further performance evaluation. This formula is especially useful in practice because it predicts the maximum message throughput of a JMS server for a planned application scenario.

We observed from our analytical parameter study that the values for both the message processing time and the corresponding server capacity λ^{max} in msgs/s range over several orders of magnitude depending on the application scenario. Both additional filters and unnecessarily sent messages reduce the JMS server capacity. Thus, we studied this phenomenon and gave a recommendation for the configuration of subscribers to maximize the JMS server capacity based on the message match probability of filters. We modelled the JMS server by an $M/GI/1-\infty$ queuing system and presented three different distributions for the message replication grade, which lead to significantly different variabilities of the message processing time. We showed that the average message waiting time is mainly influenced by the server utilization and our sensitivity analysis showed that the processing time variability plays only a marginal role. We used a normalized diagram which can be used as a lookup table for various application scenarios. The 99.99% quantile gives a "quasi upper bound" on the waiting time and an estimate on the required buffer space at the JMS server. Finally, we concluded that extensive waiting times do not occur as long as the server is not overloaded and as long as its throughput is medium or high. These results are of general nature and are also valid for other servers than the FioranoMQ.

Finally, we introduced two distributed JMS server architectures: publisher- and subscriber-side server replication (PSR, SSR). We compared the capacity of both alternatives by the use of our simple throughput model. The capacity λ_{PSR}^{max} of PSR scales well for an increasing number of publishers and the capacity λ_{SSR}^{max} of SSR scales well for an increasing number of subscribers. However, none of them scales well for both requirements. We gave recommendations for the usage of PSR or SSR depending on the application scenario.

Currently, we validate our model for other JMS servers than FioranoMQ [18], [24]–[26]. In addition, we investigate the message throughput performance of server clusters and work on concepts to achieve true JMS system scalability regarding their capacity.

ACKNOWLEDGEMENTS

The authors would like to thank Andre Bondi and Phuoc Tran-Gia for the fruitful discussions and Sebastian Gehrsitz and Christian Zepfel for the conducted measurements.

REFERENCES

[1] *Java Message Service API Rev. 1.1*, Sun Microsystems, Inc., April 2002, http://java.sun.com/products/jms/.

[2] *FioranoMQ^{TM}: Meeting the Needs of Technology and Business*, Fiorano Software, Inc., Feb. 2004, http://www.fiorano.com/whitepapers/whitepapers_fmq.pdf.

[3] P. T. Eugster, P. A. Felber, R. Guerraoui, and A.-M. Kermarrec, "The Many Faces of Publish/Subscribe," in *ACM Computing Surveys*, 2003.

[4] Krissoft Solutions, "JMS Performance Comparison," Tech. Rep., 2004, http://www.fiorano.com/comp-analysis/jms_perf_comp.htm.

[5] *Enterprise-Grade Messaging*, Sonic Software, Inc., 2004, http://www.sonicsoftware.com/products/docs/sonicmq.pdf.

[6] *TIBCO Enterprise Message Service*, Tibco Software, Inc., 2004, http://www.tibco.com/resources/software/enterprise_backbone/message_service.pdf.

[7] *IBM WebSphere MQ 6.0*, IBM Corporation, 2005, http://www-306.ibm.com/software/integration/wmq/v60/.

[8] Crimson Consulting Group, "High-Performance JMS Messaging," Tech. Rep., 2003, http://www.sun.com/software/products/message_queue/wp_JMSperformance.pdf.

[9] *Sun ONE Message Queue, Reference Documentation*, Sun Microsystems, Inc., 2005, http://developers.sun.com/prodtech/msgqueue/reference/docs/index.html.

[10] S. Bittner and A. Hinze, "A Detailed Investigation of Memory Requirements for Publish/Subscribe Filtering Algorithms," in *International Conference on Cooperative Information Systems (CoopIS)*, Agia Napa, Cyprus, Oct. 2005, pp. 148–165.

[11] A. Carzaniga and A. L. Wolf, "A Benchmark Suite for Distributed Publish/Subscribe Systems," Software Engineering Research Laboratory, Department of Computer Science, University of Colorado, Boulder, Colorado, Tech. Rep., 2002. [Online]. Available: \url{http://serl.cs.colorado.edu/~carzanig/papers/cucs-927-02.pdf}

[12] T. Wolf, "Benchmark für EJB-Transaction und Message-Services," Master's thesis, Universität Oldenburg, 2002.

[13] R. Baldoni, M. Contenti, S. T. Piergiovanni, and A. Virgillito, "Modelling Publish/Subscribe Communication Systems: Towards a Formal Approach," in 8th *International Workshop on Object-Oriented Real-Time Dependable Systems (WORDS 2003)*, 2003, pp. 304–311.

[14] R. Baldoni, R. Beraldi, S. T. Piergiovanni, and A. Virgillito, "On the Modelling of Publish/Subscribe Communication Systems," *Concurrency and Computation: Practice and Experience*, vol. 17, no. 12, pp. 1471–1495, Apr. 2005.

[15] G. Mühl, L. Fiege, and A. Buchmann, "Filter Similarities in Content-Based Publish/Subscribe Systems," *Conference on Architecture of Computing Systems (ARCS)*, 2002.

[16] A. Carzaniga, D. S. Rosenblum, and A. L. Wolf, "Design and Evaluation of a Wide-Area Event Notification Service," *ACM Transactions on Computer Systems*, vol. 19, no. 3, pp. 332–383, 2001.

[17] *Sysstat Monitoring Utilities*, http://perso.wanadoo.fr/sebastien.godard/, http://perso.wanadoo.fr/sebastien.godard/, Feb. 2004.

[18] R. Henjes, M. Menth, and S. Gehrsitz, "Throughput Performance of Java Messaging Services Using FioranoMQ," in 13th*GI/ITG Conference on Measuring, Modelling and Evaluation of Computer and Communication Systems (MMB)*, Erlangen, Germany, Mar. 2006.

[19] C. Moler, *Numerical Computing with MATLAB*. Philadelphia, PA: Society for Industrial and Applied Mathematic (SIAM), 2004, http://www.mathworks.com/moler/chapters.html.

[20] H. Takagi, *Queueing Analysis Volume 1: Vacation and Priority Systems*. North-Holland, 1991.

[21] A. Papoulis, *Probability, Random Variables, and Stochastic Processes*, 2nd ed. McGraw-Hill Book Company, 1984.

[22] A. M. Law and W. D. Kelton, *Simulation Modeling and Analysis*, 3rd ed. McGraw-Hill, 2000.

[23] M. Menth, R. Henjes, C. Zepfel, and P. Tran-Gia, "Gamma-Approximation for the Waiting Time Distribution Function of the $M/G/1-\infty$ Queue," in 2nd*Conference on Next Generation Internet Networks Traffic Engineering (NGI)*, Valencia, Spain, Apr. 2006.

[24] R. Henjes, M. Menth, and C. Zepfel, "Throughput Performance of Java Messaging Services Using Sun Java System Message Queue," in *High Performance Computing & Simulation Conference (HPC&S)*, Bonn, Germany, May 2006.

[25] ——, "Throughput Performance of Java Messaging Services Using WebsphereMQ ," in 5th*International Workshop on Distributed Event-Based Sytems (DEBS) in conjuction with ICDCS 2006*, Lisbon, Portugal, July 2006.

[26] R. Henjes, M. Menth, S. Gehrsitz, and C. Zepfel, "Throughput Performance of Popular JMS Servers," in *ACM SIGMETRICS (Poster)*, Saint-Malo, France, June 2006.