International Journal of Current Engineering and Technology
E-ISSN 2277 – 4106, P-ISSN 2347 – 5161

Available at http://inpressco.com/category/ijcet

Research Article

Measuring Tradeoffs between Performance and QoS in Event Based Systems

Gaurav Singh† and Archana Singh†

†CSE Dept., Sam Higginbottom institute of Agriculture, Technology and Sciences, Allahabad, U.P., India

Accepted 20 June 2015, Available online 26 June 2015, **Vol.5, No.3 (June 2015)**

Abstract

Event based system are message passing systems in which its components are decoupled in nature. They separate the communication and computation, via an event notification service. Because of decoupled nature of these systems, it is necessary to analyze the performance of such systems in order to guarantee reliability and other Quality of Service (QoS) specifications. This paper mainly aims to carry out performance analysis to show the effects of enforcing QoS specification viz., latency, reliability and delivery semantics. The study is based on an event based system used to notify intended subscriber vehicles in case of traffic jams. Latency and reliability is derived with certain relations and delivery semantics is enforced while modeling the system. Stochastic Analysis, using Performance Evaluation Process Algebra (PEPA) is introduced to check the overall behavior of the system.

Keywords: *Event-based systems, Communicating Systems, QoS, Performance Modeling, Stochastic Analysis, PEPA.*

1. Introduction

In today's world we are usually encountered with various message passing system which inform us about various types of event which happens now and then. Event based systems are also message passing systems which have notion of scalability and they are flexible also. These systems consist of different components which are decoupled in nature. Decoupling of components in these systems means that the components are unaware of other components. This message passing system uses a notification service to inform the intended users. There are three major components of an Event Based System namely, producers, consumers and event notification service. Event notification service acts as middleware in between producer and consumer. Producer component identifies an event and publish the information to event notification service, about its happening. Consumers need to subscribe to event notification service. When any event is recorded at event notification service by a producer, this is notified to the intended consumer. The decoupling between components makes the system extensively appropriate for large-scale distributed applications, so event based systems are getting a lot of attention in various domains.

We have proposed a model of event based system for road traffic conditions. We have modeled our system by denoting sensor as producer, subscribers A and B as consumers, and event notification service as ENS given in Fig.1.

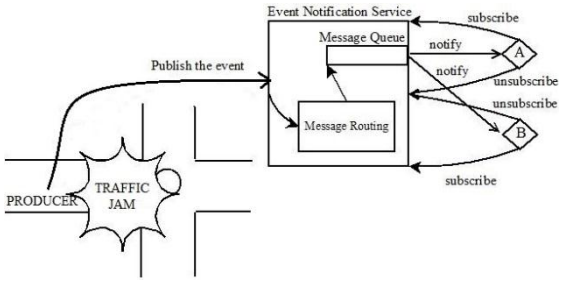

Fig.1 Proposed Event Based System

A vehicle on road will act as a producer and consumer both. It identifies a traffic jam at a certain location. This traffic jam event is notified to the event notification service (ENS). ENS, then, route the message, store it in a queue, and finally notifies the subscribed consumers about the publication of road traffic events. So, here consumers are decoupled by subscribers and vice versa, via ENS. The criticality of the system can be assumed by the nature of the time-saving system and making transportation system more efficient.

Decoupling, instead of providing flexibility, imposes certain requirements need to be analyzed for these systems. As the components in event based system works discretely, the performance of the overall system should be checked when all the components are integrated. In this paper, we are trying

*Corresponding author **Gaurav Singh** is a PG Scholar and **Archana Singh** is working as Assistant Professor

to carry out stochastic analysis on a base model of an event based system. Determining performance of the system on the basis of latency, reliability and delivery semantics, will provide a clear vision to implement such systems which incorporate QoS in event based systems. We have tried to analyze the behavior of the system, while modeling some of the QoS features in the system.

The proposed model is an event based system to handle events occurred as a result of traffic jams and are notified to intended subscribers. The model also propose consumer acknowledgement to the event notification service for every notification received. It will decrease the performance of the system, but we have tried to calculate the tradeoff between reliability and performance decay.

Rest of the paper describes our work to model and analyze event based system and the results we have found during our research. In Section II we will be listing previous researches related to performance modeling of event based systems. Section III will introduce with PEPA language along with its syntax and semantics. In section IV we will be describing our model and carry out stochastic analysis on that model. The analysis result will be presented in Section V and finally Section VI will conclude the paper providing emphasis on future directions.

2. Related Work

Several performance modeling implementations are available in the literature, which aims to identify system behavior. In (Eugster *et al*, 2003), publish/subscribe systems are described with their types. A generic publish/subscribe model is described in (Garlan *et al*,2003), where the reusable framework to capture dispatch policy and run-rime specifications. Stochastic analysis is carried out in (Mühl *et al,* 2009). While (Schröter *et al,* 2009) emphasizes hybrid routing in general peer-to-peer technologies, (Mühl *et al,* 2009) employs identity based hierarchal routing to analyze utilization and delay metrics.(He, Fei, *et al*,2007) uses the probabilistic model checking to deal with the uncertainty issues included in the systems. UML diagrams are used to model various components of publish/subscribe paradigm in (Zanolin *et al*,2003), and these models are validated with help of SPIN model checker. Methodology adopted in (Sachs, Kai *et al*, 2013) is based on the use of Petri nets. It also provides a background for capacitive planning of these systems. A formal model which employs π-calculus is implemented in (Li, Qin *et al*,2008). Here, performance metric is based on the composition and reduction applied to message brokers. Complex models are reduced in order to analyze them clearly. A detailed study on effects of various metrics like link utilization, delay in notification, publication and subscription, etc. is background of (Schröter *et al*,2010).(Ramesh *et al*,2014)(Baldoni *et al*,2005) also talks about performance modeling of event based systems, but they follow different approaches for modeling. Model checking problem in (Garlan *et al*,2000) is solved by dividing it into two parts. This is decomposed in deriving information about behavioral specifications and reusable infrastructure at runtime, in accordance with the delivery policies.

Quality of service is one of the major concerns in event based systems.(Mahambre *et al*,2007)(Hoffert *et al*, 2007) talks about importance of QoS in publish /subscribe or event based systems. Each paper has its own way to judge the implications of adding QoS specifications in the system. As in (Araujo *9t al,* 2000), QoS parameters are handled in the same way as other information regarding events.(Corsaro *et al,* 2006), on the other hand, stresses on the properties which affect QoS. In addition to this, two well-known industrial standards viz. Java Message Service (JMS) and Data Distribution Service (DDS) are also introduced. Identification and evaluation of QoS needs is demonstrated in (Appel *et al*,2010). A comprehensive study of different event based middleware is carried out in (Mahambre *et al*,2007). This study was based on availability of various QoS parameters like latency, reliability, bandwidth, etc. in different systems. Challenges to implement QoS specification in these systems are listed in (Hoffert *et al*,2009).

Our work is to query for QoS features in Event based system depends on the relations derived in (Mahambre *et al*,2007). We have tried to implement a system where we can verify the relations derived in (Mahambre *et al*,2007). In this paper, PEPA model is developed, using which we are able to perform steady state as well as transient analysis. We have calculated the overall utilization probabilities of different states, analyzing which it will be feasible to develop a better system. A relation between performance and reliability is also one of the major aspects of this paper.

3. PEPA

Performance evaluation process algebra (PEPA) is a language which uses formal approach to describe a system. It extends classical process algebra elaborated in(R.Milner,1989)(C.A.R.Hoare,1985), by incorporating timing information to the model. This addition to process algebra provides a background to carry out stochastic analysis and analyze the performance of the system and to confirm the essential characteristics of the system. It provides the method to model concurrent systems and deduce behavior of the system from the results.

A system modeled in PEPA is divided into interacting components, performing some intra or inter components actions. Every action in PEPA has an associated rate, which informs the timing aspect of that action. So, in PEPA components accomplish some activity using some action and an associated rate. A set of combinators is provided by PEPA, which allows the designing of constructs in this language. Formal syntax of PEPA can be introduced as:

$S ::= (\alpha, r).S \mid S1+S2 \mid C_s$
$P ::= P<L>Q \mid P/L \mid C$

where S, S1 and S2 are *sequential component* and P,Q denotes a *model component*. A is the action taken and r defines the rate. C denotes a constant which, depending on the equation, can be sequential as well as model component. C_S is for sequential component. The semantics of component combinators can be presented as follows:

Prefix, (α,r).S

It means the component performs an activity with action α and rate r, after which it behaves like S.

Choice, S1+S2

The + combinator represents a choice between the two associated components. It provides an option for the components to behave either as S1 or S2.

Constant, S=C

By defining S=C, the component S starts behaving like the constant C.

Cooperation, P<L>Q

The construct P<L>Q, defines two components P and Q, which cooperate with each other on L. L is a set of actions on which the two components are cooperating with each other. Concurrent execution of P and Q will affect all the actions which are defined in L, while the individual actions, which are not part of L, remain unaffected. P||Q can be written whenever L is empty set.

Hiding, P/L

Here L is a set of actions carried out by a component P which are required to be kept internal or private to hide these activities from external observers. These actions are considered as unknown type τ.

Apart from these constructs, PEPA also facilitates passive rates, defined by *infty* or T. These rates are assigned to those actions which are affected by the cooperating components. A large number of concurrent systems consider continuous time Markov chain (CTMC), to determine performance and reliability.

4. System Model

The focus of this paper is to analyzing the aspects of QoS in event based system. The proposed model represent an abstract view of event based system visualizing the external components and analyzing overall performance after integrating these components in a singles system. The different component of the system can be listed as producer, consumer and event notification service. We have also proposed a component internal to event notification service which takes care of expiry time of a notification for an event.

Assumption

Some assumptions are made while modeling our system:

- The message queue, internal to the event notification service, is never full.
- Every publication and subscription is matched as per the defined topics and the consumer is getting notifications only for the topics to which is has subscribed.

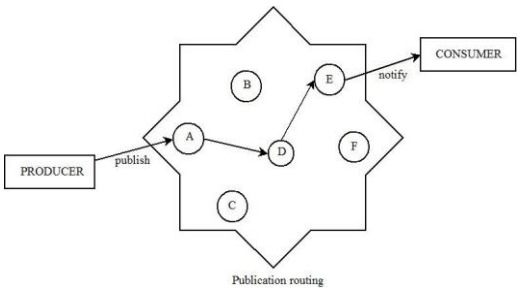

Fig.2 Publication Routing

System Specifications

ENS routes the message from one to another, as shown in Fig.2, using different routing strategies.
After the message reaches the expected ENS the message in analyzed for the content type and then it is queued in respective queue.

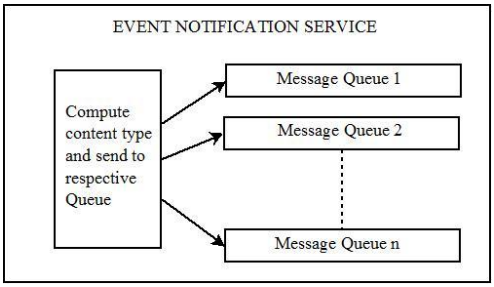

Fig.3 Abstract Model of ENS

The routing of message from one ENS to other, matching of subscription and queuing is modeled as abstract behavior of the system. The abstract model of an ENS is shown in Fig.3.

Producer, P

The producer publishes an event with a rate associated and is again ready to publish after performing the activity.

P = (pub, r_p).P

This expression signifies that the producer publishes the event at a rate r_p and is gain ready to publish the event, which happens next.

Consumer, S

Consumer component can subscribe/unsubscribe to event notification service.
S = (sub, r_s).S1;
S1 = (notify, r_n).S2 + (unsub,r_u).S;
S2 = (ack, r_a).S1;

Once subscribed, it will receive notifications and in order to ensure reliability, it will send acknowledgement for the received notification.

Event notification Service, ENS

Event notification service acts as an intermediate between producer and consumer to ensure decoupling among these two components. This component is modeled as:

B = (pub, r_p).(sub, r_s).(match, r_m).
(notify,r_n).(ack,r_a).ENS +
(sub, r_s).B +
(unsub, r_u).B ;

Producer cooperates with the event notification service with *pub* action, while consumer cooperates with *sub, unsub, notify* and *ack* actions. Table I reflects the number of instances of different components used for modeling.

Table 1 Instances of Components

Component	Instances
Producer	10
Consumer	50
ENS	3

There are two alternate proposed models – one acknowledges for receiving notification while other doesn't. By evaluating these two models we can be able to analyze the trade-off cost between reliability and performance of the notifications in the system.

5. Performance Analysis

A deep analysis of the system model has been done on the basis of QoS aspects. Apart from QoS specification, behavior of system has been analyzed on various fronts.

Capacity Utilization Analysis

Fig.4 shows that ENS is the component which utilizes its maximum capacity. As per capacity utilization analysis

ENS>CONSUMER>PRODUCER (1)

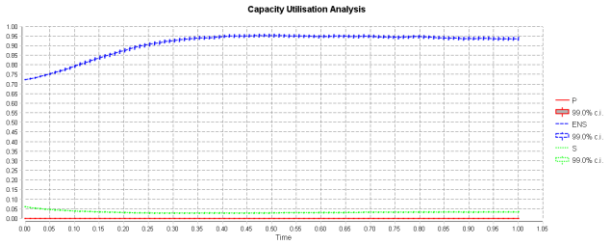

Fig.4 Capacity Utilization Analysis

Out of the total utilization time, probability of ENS for receiving publication, subscription and sending notification is high.

Pr(ENS state) = 0.18
Pr(ENS5 state) = 0.161

ENS state receives publications and/or subscriptions and ENS5 state sends notification. So, maximum time of ENS is for these two states.

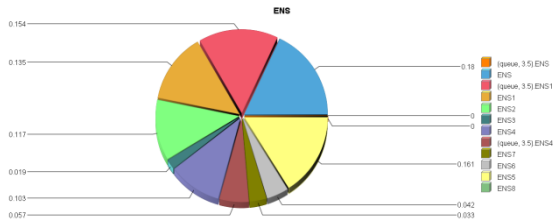

Fig.5 Probabilistic distribution of different states of ENS

Latency

In Fig.6, we can see the variations of *notify* action which is parameterized on the basis of rate of notification r_n, and rate of publication r_p. Assuming, delay in propagation of publication is L_p, queuing delay is L_q and notification delay is L_n. So, the overall latency to deliver notification after happening of event, with the help of Fig.6, can be calculated as

$$L_{total} = L_p + L_q + L_n \quad (2)$$

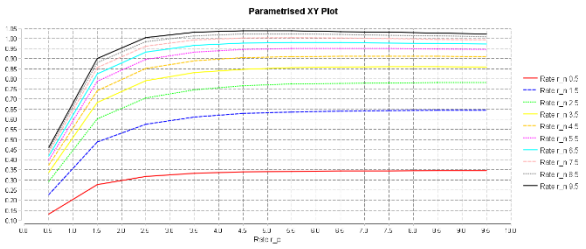

Fig.6 Performance of Notify action on the basis of rate of publication and notification

Fig.6 shows that after a certain time the system reaches obtains such a state where increasing latency of

publication and notification will not increase the overall latency of the system.

Throughput Analysis

For our system, steady state analysis reflects the data shown in Fig.6.

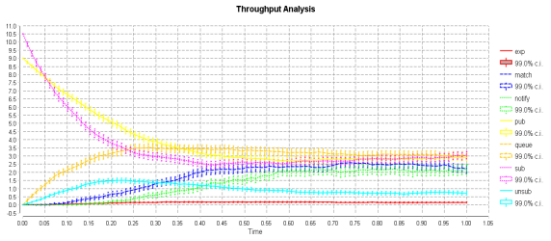

Fig.7 Throughput Analysis

It represents that after integrating the individual components into a single system, the throughput of *pub* and *unsub* action is low as compared to other actions. Activity for subscribing suffers a downfall, which stabilizes after 0.4s of operation.

Reliability

Reliability is directly dependent on the ratio of sent notifications to published events. So, as per our both models, the factor of reliability is calculated as follows:

$$R = \frac{no.of\ notifcations\ sent}{no.of\ events\ published} \quad (3)$$

We have calculated value of R for both the model with *acks* and without *acks*

Reliability with acks, R_{ack}	*0.83*
Reliability without acks, R_{wack}	*0.81*

From the above results, we can conclude that modeling the feature of acknowledgement will make the system more reliable.

Table 2 Variation in throughput of different actions for different models.

Actions	Without Acks	With Acks
Notify	1.275	0.968
Pub	1.569	1.168
Sub	1.569	1.168

From above table we can see that the throughput of the system decreases for different actions like pub, sub and notify in case of adding the feature of acknowledgement in the system.

Calculating average decrease in throughput of the system for our proposed system

% Drop in Throughput = 25.067%

As, throughput is one of the most important measurements of performance, we can say that if we will model such a system where every notification needs an acknowledgement to confirm the delivery, the performance of the system will degrade by 25.067% and there will be 1.22% increase in reliability. So, in case of extremely critical system, acknowledgement should not be a part of consumer component, but in case of general event based system it will increase the reliability.

Conclusion

Latency and reliability are among two important QoS factors. This paper tries to evaluate a proposed system on the basis of QoS. Results originated during analysis states that the trade-off between reliability of the system and system performance is much high. Modeling acknowledgement as a part of the system will decrease the performance to a great extent. In case of critical system as proposed by this paper, we cannot allow increase in reliability, on the cost of such loss in performance. But, in case of general system, which are not that much critical in nature, we can increase reliability by adding acknowledgement facility. Latency, on the other hand, is dependent on notification delay and publication delay. Adding acknowledgement to these systems will not affect latency on a higher side. In future, we will expand our study to consider other QoS factors like bandwidth and message ordering.

References

R.Milner, (1989), Communication and Concurrency. International Series in Computer Science. Prentice Hall, 2nd edition.

C.A.R.Hoare. (1985), Communicating Sequential Processes. Prentice-Hall, Englewood Cliffs, NJ.

Hermanns, Holger, and Joost-Pieter Katoen, (2001), Performance evaluation:(process algebra+ model checking) x markov chains, CONCUR 2001—Concurrency Theory. Springer Berlin Heidelberg, 59-81.

Hermanns, Holger, Ulrich Herzog, and Joost-Pieter Katoen. (2002),Process algebra for performance evaluation.-Theoretical computer science 274.1: 43-87.

Eugster, Patrick Th, *et al*., (2003), The many faces of publish/subscribe. ACM Computing Surveys (CSUR) 35.2 114-131.

Garlan, David, Serge Khersonsky, and Jung Soo Kim, (2003), Model checking publish-subscribe systems. Model Checking Software. Springer Berlin Heidelberg, 166-180.

Schröter, Arnd. (2009), Modeling and optimizing content-based publish/subscribe systems, Proceedings of the 6th Middleware Doctoral Symposium. ACM.

Mühl, Gero, *et al*., (2009), Stochastic analysis of hierarchical publish/subscribe systems.Euro-Par 2009 Parallel Processing. Springer Berlin Heidelberg, 97-109.

Zanolin, Luca, Carlo Ghezzi, and Luciano Baresi. (2003), An approach to model and validate publish/subscribe architectures. Proc. of the SAVCBS. Vol. 3.

Sachs, Kai, Samuel Kounev, and Alejandro Buchmann, (2013), Performance modeling and analysis of message-oriented event-driven systems. Software & Systems Modeling 12.4, 705-729.

He, Fei, *et al*., (2007), Formal analysis of publish-subscribe systems by probabilistic timed automata. Formal Techniques for Networked and Distributed Systems–FORTE, Springer Berlin Heidelberg, 247-262

Li, Qin, *et al*., (2008), Scalable formalization of publish/subscribe messaging scheme based on message brokers. Web Services and Formal Methods. Springer Berlin Heidelberg, 61-76.

Schröter, Arnd, *et al*., (2010), Stochastic performance analysis and capacity planning of publish/subscribe systems. Proceedings of the Fourth ACM International Conference on Distributed Event-Based Systems. ACM.

Garlan, David, and Serge Khersonsky, (2010), Model checking implicit-invocation systems. Software Specification and Design, 2000. Tenth International Workshop on. IEEE

Fiege, Ludger, *et al*., (2002), Engineering event-based systems with scopes. ECOOP 2002—Object-Oriented Programming. Springer Berlin Heidelberg, 309-333.

Araujo, Filipe, and Luís Rodrigues, (2002), On QoS-aware publish-subscribe. Distributed Computing Systems Workshops, 2002. Proceedings. 22nd International Conference on. IEEE

Corsaro, Angelo, *et al*., (2006), Quality of service in publish/subscribe middleware, Global Data Management 19 20.

Appel, Stefan, Kai Sachs, and Alejandro Buchmann. Quality of service in event-based systems. Proceedings of the 22. GI-Workshop on Foundations of Databases, GvD. 2010.

Mahambre, Shruti P., Madhu Kumar, and Umesh Bellur. (2007), A taxonomy of qos-aware, adaptive event-dissemination middleware, Internet Computing, IEEE 11.4 : 35-44

Hoffert, Joe, and Douglas C. Schmidt, (2009) Maintaining QoS for publish/subscribe middleware in dynamic environments. Proceedings of the Third ACM International Conference on Distributed Event-Based Systems. ACM

Ramesh, Chithrupa, Henrik Sandberg, and Karl H. Johansson, (2014) Performance Analysis of a Network of Event-based Systems. arXiv preprint arXiv:1401.4935.

Kounev, Samuel, Christoph Rathfelder, and Benjamin Klatt.(2014) Modeling of Event-based Communication in Component-based Architectures.