Poza-Lujan, J.; Posadas-Yagüe, J.; Simó Ten, JE. (2009). From the Queue to the Quality of Service Policy: A Middleware Implementation. En Distributed Computing, Artificial Intelligence, Bioinformatics, Soft Computing, and Ambient Assisted Living. Springer Verlag (Germany). 432-437. doi:10.1007/978-3-642-02481-8_61.

# From the Queue to the Quality of Service Policy: a Middleware Implementation

José L. Poza, Juan L. Posadas, José E. Simó

Institute of Industrial Control Systems
Polytechnic University of Valencia
Camino de Vera s/n, 46022, Valencia, Spain
{jopolu, jposadas, jsimo}@ai2.upv.es

**Abstract.** Quality of service policies in communications is one of the current trends in distributed systems based on middleware technology. To implement the QoS policies it is necessary to define some common parameters. The aim of the QoS policies is to optimize the user defined QoS parameters. This article describes how to obtain the common QoS parameters using message queues for the communications and control components of communication. The paper introduces the "Queue-based Quality of Service Cycle" concept for each middleware component. The QoS parameters are obtained directly from the queue parameters, and Quality of Service Policies controls directly the message queues to obtain the user-defined parameters values.

## 1. Introduction

Manage the quality of service (QoS) on the middleware layer is one of the current trends in the field of distributed systems. Each component of a distributed system has some particular characteristics so it is difficult the distributed management of the QoS parameters. To make easier the QoS management appears the concept of QoS policy.

There are a lot of middleware with QoS support [1]. Among the current middleware architectures, stands out the Data Distribution Service (DDS) model proposed by the Object Management Group (OMG) [2]. DDS introduces the concept of QoS policy instead of QoS parameters to manage communications between the components of a distributed system. The concept of QoS policy is extended to the control layer. The control layer is based on the Sensor Web Enablement (SWE) model, proposed by Open Geospatial Consortium (OGC) [3]. The model presented in this article, called FSA-Ctrl [4], is based on the synergy of two standards models: DDS and SWE.

The remainder of the paper is organized as follows. In Section 2, we overview the definition on QoS, with particular attention to the parameters widely used in the main bibliography. Section 3 describes the FSACtrl architecture and how to manage the QoS policies based on the "QoS-cycle" concept. Conclusions and future work are in Section 5.

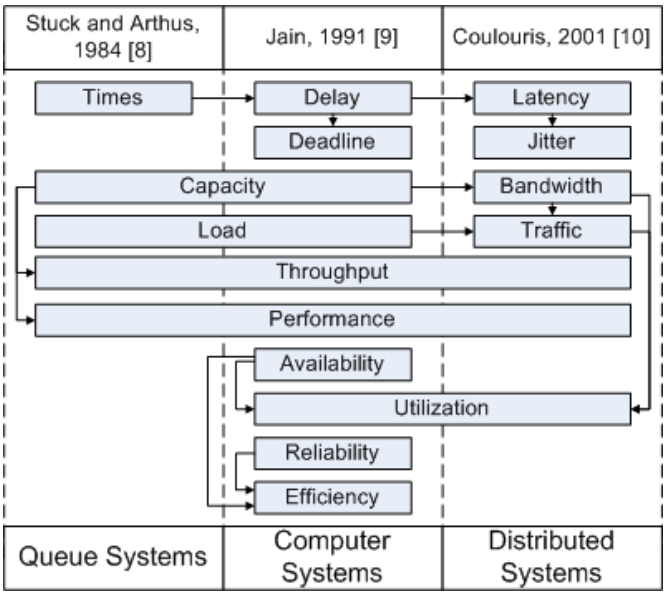


**Fig. 1.** Evolution of QoS parameters and relations between them.

## 2. Quality of Service, Parameters and Policies

### 2.1 Quality of Service

Is difficult to find a QoS definition, there are a lot of authors that defines Quality of Service based on the context of application. In the communications field, the following definitions are interesting.

- Quality of service represents the set of those quantitative and qualitative characteristics of a distributed multimedia system necessary to achieve the required functionality of an application [5].
- Quality of Service is a set of service requirements to be met by the network while transporting a flow [6].
- The collective effect of service performance, which determines the degree of satisfaction of a user of the service [7].

In the above definitions, services must have a set of measureable characteristics. To measure the characteristics, the middleware uses the parameters.

### 2.2 QoS Parameters

Initially, the QoS is measured directly from the parameters of the message queues [8], like service time, capacity or throughput. If the QoS is applied to the computation performance, most of the parameters are the same as the message queues parameters [9], related to the computation like delay or deadline. However, in the distributed systems, is usual to use more complex parameters [10], like availability, reliability or efficiency. The user viewpoint about the QoS is a difficult problem, some questions about the translation between user and application can be obtainer from [11]

Figure 1 show how the concept of QoS parameters has changed depending on the context in which the parameter is applied. It is interesting to note, how the parameters are closely related to each other. Moreover, from the message queue parameters can be obtained the usual QoS parameters. For example, the throughput of a component can be obtained from the occupancy rate of the message queue associated with the component. [8], this relationship is shown in equation 1.

$$\lambda_i = (1- p_0)\ \mu_i \quad (\mathbf{1})$$

$$\mu = 1 / E[S] \quad (\mathbf{2})$$

The throughput of a component is represented by $\lambda_i$. "The probability of finding a message queue occupied is $(1- p_0)$. Finally $\mu$ represents the service rate. If the equation is extended to the relations among the components, the common throughput can be obtained from the throughput of each single component. Equation 2 shows how the service rate ($\mu$) can be obtained from the average of messages in the queue (E[S]).

### 2.3 QoS Policies

The concept of QoS is used to measure all relevant characteristics of a system. Generally, QoS is associated with a set of measurable parameters. QoS policy can be defined as the dynamic management of the QoS parameters whit a negotiated values. Next, we try to define both concepts: QoS and Parameters. The aim of the FSACtrl architecture is unify both concepts: QoS policies and message queues using a set of well defined parameters. In [12] there is an example of other middleware with QoS policies support.

DDS specification proposes 22 different QoS policies that cover all aspects of communications management: message temporal aspects, data flow and metadata. For example, by means the “Deadline” policy, that determines the maximum time for the message arrival, and the “TimeBasedFilter” policy, that determines the minimum time between two messages, a component can establish a temporal window to receive messages from other components.

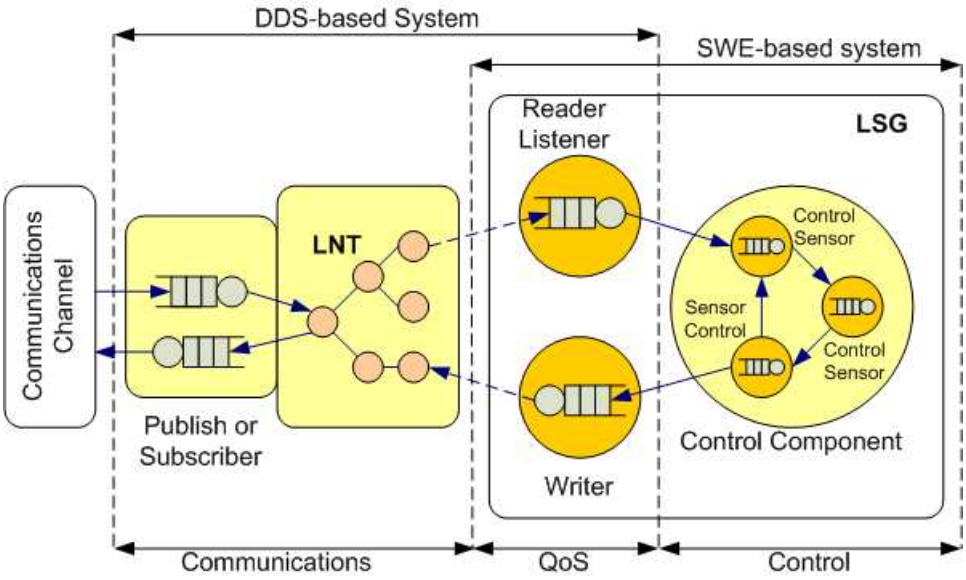


**Fig. 2.** Major components of the FSACtrl architecture, and his message queues.

## 3. Queue-based Quality of Service Cycle

### 3.1 Where are the messages queues placed?

The research group has developed a middleware with QoS support. The architecture is called FSACtrl [4]. All components of FSCtrl are based on a common component that contains a unique message queue. Figure 2, shows how communications layer and control layer has similar messages queues. QoS policy management acts in the negotiation between the elements of control and communication.

There are two important components: reader and writer. Readers and Writers are the common components from DDS and SWE model. Readers and Writers are placed on the intersection of the DDS and SWE model. Their primary function is to manage the message flow between the control layer and the communication layer. The QoS layer is the responsible of the managing of the message flow and time restrictions.

### 3.2 Steps to control the Message Queues

Each of the components of the FSACtrl architecture has a unique message queue. With the combination of the message queues behaviour, system can be adjusted to accomplish the user-defined requirements.

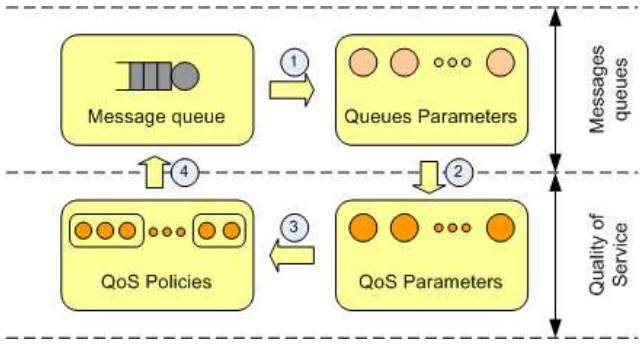


**Fig. 3.** Graphic of the queue-based quality of service cycle concept.

Figure 3 shows the four steps of the Queue-based Quality of Service Cycle. The steps are repeated for all the communication process. Next, the steps shown in the figure 3 will be described.

1. Initially the queue provides the simple parameters, like the number of messages waiting in the queue, or the time difference between the arrival and the departure of a message to be processed.
2. From the simple formulas, like the formula shown in the equation 1, component can obtain the QoS parameters based on the relations displayed on the figure 1.
3. QoS parameters are analyzed and combined by the QoS policy algorithm. The result determines if the user-defined requirements are between the correct values.
4. If the result is out of limits, the policy acts on the message queue. Message queues allow changes as the priority to send the messages to the rest of the system queues or the message buffers sizes.

The previous steps, provides to the control layer an important feedback about how the values of a queue can be used to determine a QoS policy.

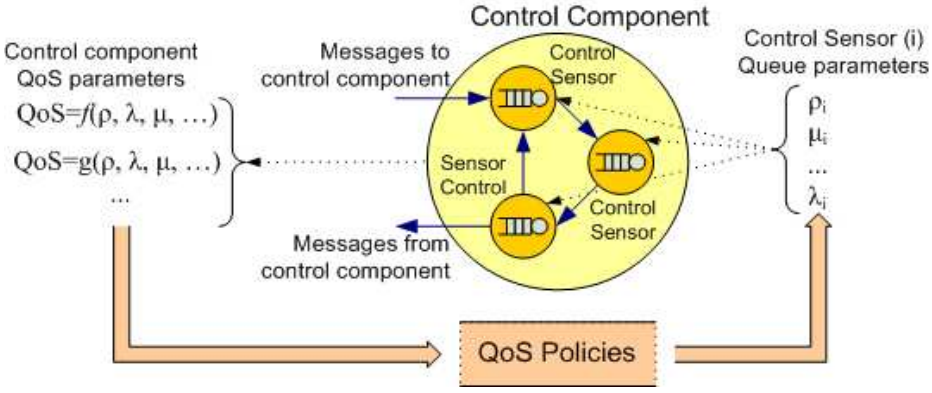


**Fig. 4.** Local QoS parameters and common QoS Policy.

Figure 4, shows how each single component contributes with a small part of the final component QoS values. Each QoS policy processes the relevant values and sends the results to each control sensor. If a component needs increase the throughput, a single control sensor can modify his service average rate, through the prioritization of messages in the queue or decreasing the number of messages processed, although this involves a decrease in the accuracy of the result.

## 5. Conclusions

This article has presented a concept called "Queue-based QoS Cycle," by which a distributed system can be managed from the parameters obtained from their individual components.

The QoS is based on the DDS model. Its main use is to predict the temporal needs and message flow to each component of a control algorithm. Defining the values of the QoS parameters, a system can self-configure the behaviour of the message queues. Moreover, the load of each control component can be calculated with a simple simulation. However, the overload produced by the use of a message queue for each component, makes the architecture difficult to use in a embedded systems.

Currently are being implemented all the QoS policies specified in the DDS model. The next step is determining the main formulas to obtain QoS parameters from the message queues.

**Acknowledgements.** The middleware architecture described in this article is a part of the coordinated project SIDIRELI: Distributed Systems with Limited Resources. Control Kernel and Coordination. Education and Science Department, Spanish Government. CICYT: MICINN: DPI2008-06737-C02-01/02

## References

1. Aurrecoechea, C., Campbell, A.T. and L. Hauw, "A Survey of QoS Architectures", ACM/Springer Verlag Multimedia Systems Journal, Special Issue on QoS Architecture, Vol. 6 No. 3, pg. 138-151, May 1998.
2. OMG. “Data Distribution Service for Real-Time Systems, v1.1.” Document formal/2005-12-04. December 2005.
3. Botts M, Percivall G, Reed C, Davidson J, (2006) OGC®. Sensor Web Enablement: Overview And High Level Architecture, OpenGIS Consortium Inc
4. Poza, J.L., Posadas, J.l. and Simó, J.E. QoS-based middleware architecture for distributed control systems. International Symposium on Distributed Computing and Artificial Intelligence. Salamanca. 2008
5. Andreas Vogel, Brigitte Kerherve, Gregor von Bochmann, Jan Gecsei. Distributed Multimedia and QoS: A Survey. Vol.2,. No. 2, 1995, pp.10-19.
6. Crawley, E.; Nair, R; Rajagopalan, B. “RFC 2386: A Framework for QoS-based Routing in the Internet”. August. 1998, pp. 1-37, XP002219363.
7. ITU-T Recommendation E.800 (0894). Terms and Definitions Related to Quality of Service and Network Performance Including Dependability, 1994.

8. B.W. Stuck and E. Arthurs. A Computer & Communications Network Performance Analysis Primer. Prentice Hall. 1984.
9. Raj Jain. The art of Computer Systems Performance Analysis. John Wiley & Sons Inc. New york. 1991.
10. Coulouris, G., Dollimore, J., Kindberg, T. Distributed Systems. Concepts and Design. Third Edition. Addison Wesley. Madrid. 2001.
11. Jae-ll Jung, "Quality of Service in Telecommunications Part II: Translation of QoS Parameters into ATM Performance Parameters in B-ISDN", IEEE Comm. Mag. Aug. 1996, pp.112-117
12. Eric Wohlstadter, Stefan Tai, Thomas Mikalsen, Isabelle Rouvellou, Premkumar Devanbu, "GlueQoS: Middleware to Sweeten Quality-of-Service Policy Interactions," icse,pp.189-199, 26th International Conference on Software Engineering (ICSE'04), 2004