QoS-based middleware architecture for distributed control systems

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

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

Abstract. This paper presents an implementation of a middleware architecture to control distributed systems. The main objective is providing a QoS level between the communications layer and the control layer. This architecture is based on the use of a hierarchical communications structure called "logical namespace tree" and a structured set of control processes interconnected, called "logical sensors graph". This architecture is named Frame Sensor Adapter Control (FSA-Ctrl). In this architecture communication layer and control layer can manage the QoS policies. The communication layer is based on the Data Distribution Service (DDS), a standard proposed by Object Management Group (OMG). Control layer is derived from the Sensor Web Enablement (SWE) model proposed by Open Geospatial Consortium (OGC). Middleware components use messages queues to manage components QoS requirements. By means of QoS policies, control components can take important decisions about distributed questions, like components mobility or information redundancy detection.

Keywords: Distributed system, control architecture, quality of service.

1 Introduction

Usually quality of service (QoS) used in the communication layer provides simple temporal parameters like messages delay or easy message flow control like congestion control. When communication system has important requirements, like real-time support, information optimization or components hidden, then communication layer should use more QoS policies, with more complex parameters.

A middleware must control all communication parameters involved in message management like the message producers and consumers or all questions related to the use of message queues. Most of the communication paradigms are designed to improve message speed or hide the communications components to the application layer. Nevertheless, QoS covers some more aspects, like optimizing messages or metadata management. Publish/Subscribe (P/S) communications paradigm provides an appropriate environment for information distribution, and the messages queues provides a flow control extending middleware QoS policies.

Among the current communication architectures standards, the Data Distribution Service (DDS) provides a system based on publish-subscribe paradigm [1] with ability to manage a large amount of QoS policies [2]. Likewise, among the current control architecture standards, the Sensor Web Enablement (SWE) allows a simple intelligent control model based on services capable of managing complex sensor networks [3]. Joining DDS and SWE standards is interesting because it provides a unique environment to implement a solution to manage component-based distributed intelligent control system, based on the combination of several QoS policies.

This paper presents the model of an architecture called Framed Sensor Adapter Control (FSA-Ctrl) whose communication components are based on the DDS model and its control components are based on the SWE model. The QoS merges communications and control components. The communication layer is called Logical Namespace Tree (LNT) and it is a hierarchical abstraction of the real communications channels, like TCP/IP, EIB bus, and they adapters. Each information item in the LNT can be identified by means of a topic called Logical Data (LD). The control layer is known as Logical Sensor Graph (LSG) and it is a structured set of small control processes interconnected by means the LNT or internally. Each control process unit is called Logical Sensor (LS) and they are an abstraction of the control components.

The rest of the paper is organized as follows. Section 2 briefly describes the functional structure of DDS and SWE standards. Section 3 presents in detail the LNT and LSG components and the role of QoS in the system. Section 4 presents concluding remarks.

2 Fundamentals: DDS and SWE

Most of the communication systems that provide support to the distributed control architectures need a module that hides the details of the components connexions. Usually, when this module is separated from control components, is known as "middleware". The main responsibility of middleware is providing, to control components, the necessary services to increase efficiency of communication. Among the required services is outstanding identification of components, authentication, authorization, hierarchical structuring or components mobility.

Moreover, the underlying programming technology like object-oriented programming, component-based programming or service-based programming determines partially the resultant control architecture and its ability to provide more QoS features [4].

To develop a distributed system based on components, some components must adapt his technology to the communication interfaces. For example, if communication is based on CORBA [5], the distributed system must be implemented with the object-oriented programming technology. To avoid the use of a particular technology usually distributed systems use standardized protocols like FIPA [6]. To support distributed protocols, usually systems use a message-based service, like JMS [7], that offers some control to QoS.

2.1 Data Distribution Service

Data Distribution Service (DDS) provides a platform independent model that is aimed to real-time distributed systems. DDS is based on publish-subscribe communications paradigm. Publish-subscribe components connect information producers (publishers) and consumers (subscribers) and isolate publishers and subscribers in time, space and message flow [8]. To configure the communications, DDS uses QoS policies. A QoS policy describes the service behaviour according to a set of parameters defined by the system characteristics or by the administrator user. Consequently, service-oriented architectures are recommended to implement QoS in his communication modules.

DDS specifies two areas: Data-Centric Publish-Subscribe (DCPS) witch is responsible for data distribution and DLRL which is responsible for the data adaptation to the local applications level. The DLRL area is optional due to the DCPS components can work directly with the control objects without data translations.

DCPS has a lot of components, or classes in the case of the object-oriented (OO) model, in his formal model. However, there are mandatory components, presented at figure 1. When an producer (component, agent or application) wants to publish some information, should write it in a "Topic" by means of an component called "DataWriter" witch is managed by another component called "Publisher". Both components, DataWriter and Publisher, are included in another component called "DomainParticipant". On the other side of the communication, a Topic can be received by two kinds of components: "DataReaders" and "Listeners" by means a "Subscriber". A DataReader provides the messages to application when the application request-it, in lieu a "Listened" sends the messages without waiting for the application request. An application will use as many DataWriters, DataReader or Listeners as distributed topology requires. DataReaders, DataWriters and Listeners will be the elements responsible for maintaining message queues, and consequently responsible for implementing QoS policies.

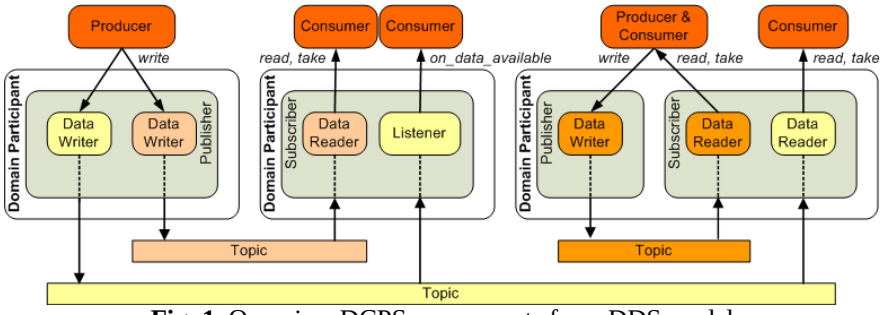

Fig. 1. Overview DCPS components from DDS model.

2.2 Quality of Service

QoS is a concept that defines a set of parameters by which to evaluate a service offered. In the field of control architectures, there are many definitions of quality of service. From the viewpoint of processing, QoS represents the set of both: quantitative and qualitative characteristics of a distributed system needed to achieve the functionality required by an application [9]. From the communication viewpoint, QoS is defined as all the requirements that a network must meet to message flow transport [10].

In DCPS model, all objects may have associated a set of policies QoS. At present the DCPS specification defines 22 different QoS policies that can be classified into four areas: times, flows, components and metadata management. All components of a DCPS based communication, can establish a set of QoS policies into them independently the others components.

2.3 Sensor Web Enablement

The main objective of Sensor Web Enablement (SWE) is providing a unique and revolutionary framework of open standards for exploiting Web-connected sensors and sensor systems of all types [11]. SWE was developed in 2004 as part of an initiative by the OpenGIS Consortium (OGC). At present SWE is used especially for monitoring and management of sensor networks. The proposed model is currently used by many organizations, like NASA and computer weather systems. SWE assign control functions to several interconnected elements.

The components of SWE are divided in two groups: information models and services. Information models are standard specifications in XML, processes interchanges messages with these specifications. Services are control components that process the information models. Control processes are based on components interconnected, those receive information models from other components, and send the results to connected components.

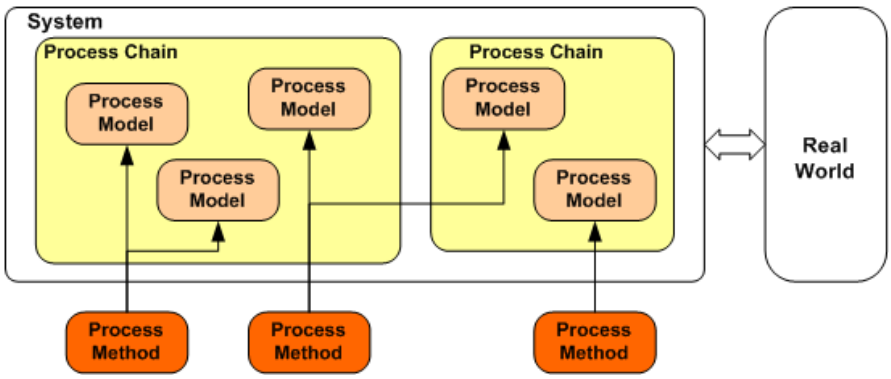

Fig. 2. SWE control architecture components overview.

From SWE viewpoint, a component is a particular physical process that transforms information. Simple examples of SWE components are sensors, effectors or physical process filters. Complex examples of SWE components are control kernels or sensor data fusion algorithms.

As shown in the Figure 2, a “Process Model” is a single component, used into a more complex structure, called “Process Chain”. Moreover, a “Process Model” is based on a “Process Method” witch acts as a “Process Model” template. A “Process Method” specifies the interface and how to implement the “Process Model”, also define inputs, outputs and the operating parameters. The model proposed by SWE is very interesting because allows to specify reusable process patterns. This scheme provides a highly scalable control system based on singles control kernels.

Anyway, it should take some precautions when using this scheme. The highly interconnected model increases redundant information because the model hides the data sources. Also, repetition in control patterns can lead to control actions repeated. Finally, the interconnection of control models can generate undesirable control cycles. Any SWE based architecture must prevent these aspects.

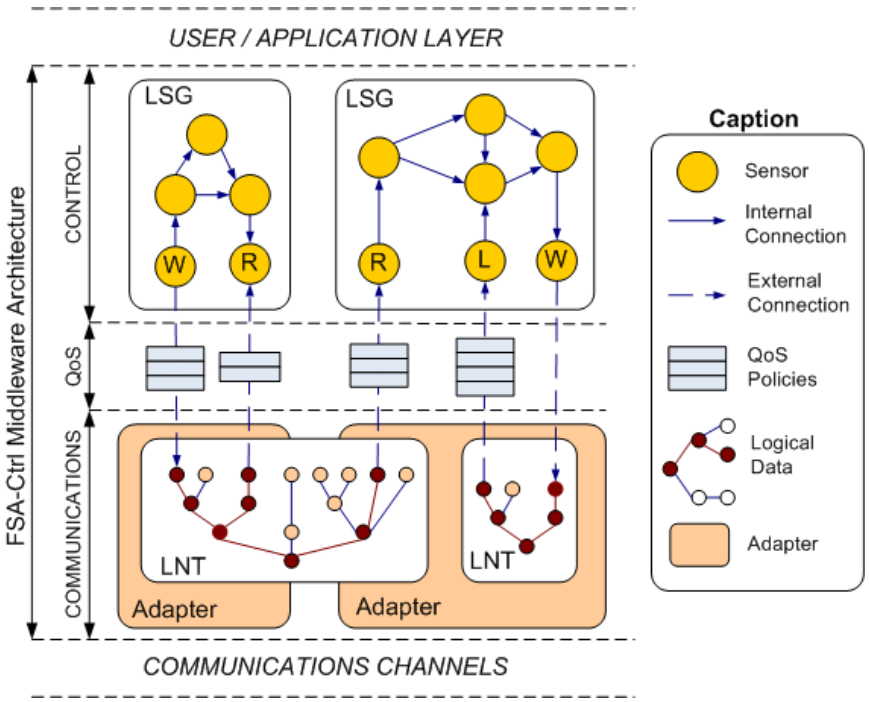

Fig. 3. FSA-Ctrl architecture overview.

3 FSA-Ctrl Architecture

Frame Sensor Adapter to Control (FSA-Ctrl) is an architecture inspired by DDS and SWE models and is an evolution of an architecture developed by the research group called Frame Sensor Adapter (FSA) [12]. The architecture has two distinct areas: communication and control. QoS Policies connects both areas. Figure 3 shows the details of the architecture.

The “Frame” component of the FSA architecture takes the same role of the “DomainPaticipant” component of the DDS architecture. The “Adapter” component takes the role of both DDS components, “Publisher” and “Subscriber”. A specialization of “Sensor” component takes the role of the “DataWriter”, “DataReader” and “Listener” DDS components. The function of the “Topic” DDS component is performed by the “LogicalData” component of the FSA-Ctrl architecture. The communication layer organizes the “LogicalData” in a hierarchical structure to hide any type of communication channel like the TCP/IP protocol, EIB or CAN bus. The structure is a symbolic tree called Logical Namespace Tree (LNT), details and examples can be obtained from [13].

Control layer organizes the “Sensors” on a graph, called Logical Sensor Graph (LSG); details can be obtained from [14]. This model is based on SWE “Process Chain". The process units are known as “Logical Sensors”, and some of this “Logical Sensors” takes the role of some communication components. A “Logical Sensor” can receive, or send, messages from, or to, another “Logical Sensors”.

3.1 Communication: adapters and LNT

Usually, platforms that offer communication to distributed control architectures use several methods to locate the components. Frequently the communications systems provide to control system a name or an address that represents the component. The name of the component can’t offer more information, like type of component, location and other features. FSA-Ctrl architecture uses a layer, which hides the protocol-dependant location method and organizes the components depending on user -defined characteristics.

To connect the communications channels, like TCP/IP, EIB or CAN bus, FSA-Ctrl uses a type of component called adapter. Adapter hides details of the media and fit messages to a SWE message format standardized, like SensorML. Messages are dispatched to the control components that are interested in them.

To manage system components, FSA-Ctrl organizes the information in a tree structure called LNT. The LNT locates both main FSA-Ctrl components: adapters and sensors, by a concept named Logical Data (DL).

A logical data is a sequence of names separated by the symbol ‘/’. Every name is known as “logical node”. Symbol ‘*’ represents the sub-tree derivates form a node. This structure provides a common meta-information about the type of message or type of component involved in the communication. For example, is possible obtain all temperatures from an home automation system by a subscription to the logical data “root/home/sensors/temperature/*”.

3.2 Control: sensors and LSG

Components witch implements the control system are named Logical Sensors (LS) and contains the control algorithms. LS can implement from a simple process or single operation, like an arithmetic addition or a logical comparison, to complexes tasks like the obstacle avoidance in a robot navigation algorithm.

Communication into Logical Sensors can be of two kinds: internal and external, depending on location in the distributed system (Figure 3). To communicate two sensors into the same node execution, adapters employs internal messages provided by the operating system. If two Logical Sensors resides in separated execution nodes, adapter uses the communication channel that connects the execution nodes. In both cases, Logical Sensors uses the same communications interface (LNT).

By means of the LNT, the boundary between communications and control can change and provides a simple framework to move components into the distributed system. Through QoS, Logical Sensors can make decisions about the best sources or destinations to work. One of the strengths of the control system is that Logical Sensors may include others Logical Sensors to create more complexes Logical Sensors than gives more complex algorithms.

Adapters maintain the message queues between communications layer and the LNT. These queues only give restrictions of the QoS parameters that adapters can offer to control. The DDS-based QoS policies are responsibility of the communication sensors.

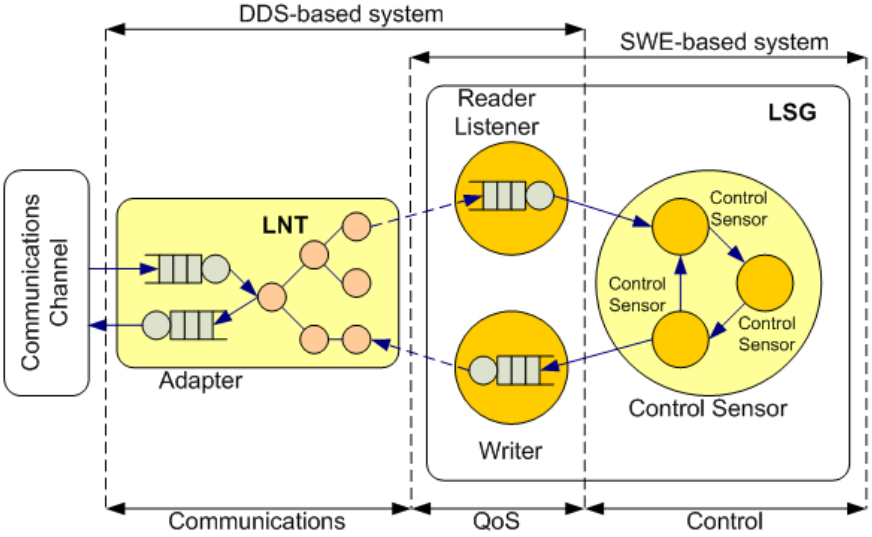

Fig. 4. FSA-Ctrl architecture overview.

3.3 Conceptual model

Figure 5 shows a formal description of the FSA-Ctrl architecture by means of a UML class diagram. “Entity” is the class base for all components, except for the QoS policy. Each component can have associated several QoS policies. This relationship is performed at the class base level to standardize the QoS to all components derived from “Entity” class.

The classes “Frame” and “FrameEntity” are derived from “Entity” class, these classes contain the elements of the architecture. “Frame” class represents the execution framework to sensors and adapters components and “FrameEntity” is the base class for “Adapter” class and “Sensor” class. “LogicalData” is a data model managed by the Adapter “class”. A LogicalSensor object can be connected with some others LosgicalSensors, but only one LogicalSensor can be connected with an Adapter. Making a similarity with DCPS model of DDS standard, the “Frame” component of FSA-Ctrl takes the same role as the “DomainParticipant”, “Adapter” is similar than “Publisher” and “Subscriber” and “Logical Sensors” makes the same role as the “DataWriter”, “Datareader2 and “Listener” components. The role of a “LogicalData” is the same that “Topic” in DCPS. When a “Logical Sensor” does not have an associated “Adapter”, then is a control component, and can be associated with others control components.

In FSA-Ctrl architecture, QoS is used by all components. “Adapters” and “Logical Sensors” specialized on communications uses the QoS policies to manage times and messages flow parameters, although “Logical Sensors” uses the QoS policies to manage the control action computation efficiency. DDS and QoS becomes a common interface between communications and control.

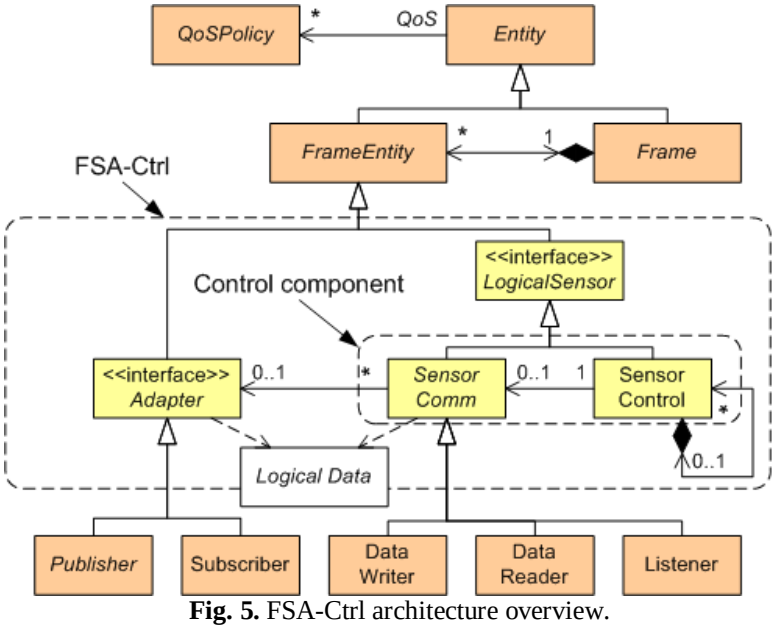

Fig. 5. FSA-Ctrl architecture overview.

4 Conclusions

This article has presented the UML specification of a middleware with QoS support. The middleware hides the communications details and offer simple control components. The architecture, called FSA-Ctrl, is based in two standards architectures, DDS and SWE, and takes the benefits of a QoS-based communication. DDS, based on publish-subscribe paradigm, is a standard supported by OMG. SWE standard is endorsed by OGC. The FSA-Ctrl architecture especially focuses on the use of QoS policies.

FSA-Ctrl architecture may have different uses. A publish-subscribe-based system provides highly scalable systems. The fact that FSA-Ctrl is based on DDS will be able to adapt easily at many systems, and the simple set of communications components allows converting easily the communications interface system into another.

Moreover, due to communications components can implement part of the control system, control applications can perform another tasks. In addition, the management of the QoS policies is performed on the middleware, which gives to applications a large number of calculated parameters. With these parameters, the system can make decisions about questions regarding the discrimination of redundant information or taking decisions about agents' movement. The use of the LNT provides a structured and hierarchical abstraction of the system to control, a very detailed topology of the control algorithms and a great ability to tune the system performance based on QoS policies.

SWE-based control components allow the high-distributed control algorithms. Moreover, the logical sensor-based can be applied to evaluate the proper location for the components. Since certain control components with a set of QoS restrictions for each, the architecture could distribute automatically agents' control processes according to a set of parameters values.

System has some drawbacks, usually related to the use of a middleware. The main disadvantage is a slight speed reduction intrinsic to the use of queues and the LNT structure.

Currently, a beta version of the middleware, based on the FSA-Ctrl architecture, is being tested on a home automation project. System is used to obtain the most relevant QoS parameters from the basic algorithms, used on home automation control. The next step in the project is a real-time testing of the middleware to determine a set of QoS parameters to model performance aspects. The library has a set of adapters that can communicate an application with real-time CAN bus, the home automation bus EIB, TCP/IP services and a MySQL database service.

Middleware can be used in distributed systems to measure the communications QoS parameters. The obtained values can be used to distribute the components according to the communications performance, with the aim of achieve a system able to auto distribute its components according to some relevant changes in the environment.

Acknowledgements. The architecture described in this article is a part of the coordinated project KERTROL: Kernel control on embedded system strongly connected. Education and Science Department, Spanish Government. CICYT: DPI2005-09327-C02-01/02

References

1. Matteucci, M. Publish/Subscribe Middleware for Robotics: Requirements and State of the Art. Technical Report N 2003.3, Politecnico di Milano, Milano, Italy. (2003)
2. OMG. Data Distribution Service for Real-Time Systems, v1.1. Document formal/2005-12-04. (2005)
3. OGC. Sensor Web Enablement: Overview and High Level Architecture. OGC White Paper. OGC 06-050r2. Edited by Botts, M., Percivall, G. Reed, C. and John Davidson. (2006)
4. Coulouris, G., Dollimore, J. and Kindberg, T. Distributed systems, concepts and design. Third edition. Addison Wesley. (2001)
5. OMG. Real-Time Corba Specification version 1.1. Document formal /02-08-02. (2002)
6. FIPA. Specfication. Part 2, Agent Communication Language. Foundation for Intelligent Physical Agents. (1997)
7. Hapner, M., Sharma, R., Fialli, J. and Stout, K. *JMS specification*. Sun Microsystems Inc. Santa Clara, CA 95054 USA, Vol. 1.1. (2002)
8. Pardo-Castellote, G. OMG Data-Distribution Service: architectural overview. Proceedings of 23rd International Conference on Distributed Computing Systems Workshops. Providence, USA. Vol. 19-22, pp. 200-206. (2003)
9. Vogel, A., Kerherve, B., von Bochmann, G. and Gecsei, J. Distributed Multimedia and QoS: A Survey. IEEE Multimedia.Vol.2, No. 2, pp. 10-19. (1995)
10. Crawley, E., Nair, R., Rajagopalan, B. RFC 2386: A Framework for QoS-based Routing in the Internet. IETF Internet Draft. pp. 1-37. (1998)
11. Botts M., Percivall G., Reed C. and Davidson J. OGC. Sensor Web Enablement: Overview and High Level Architecture, OpenGIS Consortium Inc. (2006)
12. Posadas, J. L., Perez P., Simo J.E., Benet G. and Blanes F. Communication structure for sensory data in mobile robots. Engineering Applications of Artificial Intelligence. Vol. 15, No 3-4, pp 341-350. (2002)
13. Poza, J. L., Posadas, J.L., Simó, J.E. and Benet, G. Hierarchical communication system to manage maps in mobile robot navigation. Proceedings of International Conference on Automation, Control and Instrumentation. Valencia, Spain. (2006)
14. Poza, J.L., Posadas, J.l. and Simó, J.E. Distributed agent specification to an Intelligent Control Architecture. 6th International Workshop on Practical Applications of Agents and Multiagent Systems. Salamanca, Spain. pp. In Press. (2007)