Middleware with QoS support to control intelligent systems

José Luis Poza Luján, Juan Luis Posadas Yagüe, José E. Simó Ten
Institute of Industrial Control Systems. Polytechnic University of Valencia
{jopolu, japosadas, jsimo}@ai2.upv.es

Abstract

This paper presents the architecture of a middleware for control component-based distributed systems. The fundamental idea underlying this architecture is 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). The aim of FSA-Ctrl is to provide a simple interface to specify the control actions and ensure a communication with some specified QoS parameters restrictions. In this architecture both systems, communication and control, manages the QoS policies. The communication system is based on the Data Distribution Service (DDS), a standard proposed by Object Management Group (OMG). Control system is derived from the Sensor Web Enablement (SWE) model proposed by Open Geospatial Consortium (OGC). The system introduced in this paper is focused on the use of QoS parameters of the distributed systems components to manage the information redundancy.

1. Introduction

Nowadays, one of the most relevant technological challenges is the management of peer-to-peer quality of service (QoS) for component-based distributed intelligent control systems.

This aspect of distributed systems goes beyond the real-time requirements because involves considerations such as: computational resources availability, security, control algorithms cooperation, stability, task control performance and the redundant information management.

Current communications system are not designed to provide complex QoS, just very simple features of QoS like message sequencing, traffic congestion relieving, and so on.

To manage complex component-based distributed intelligent control systems, much more QoS features are needed. A way of getting these new features are communications based on message queuing technologies and publish-subscribe paradigm and control system based on services paradigm. 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 implementation 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. An example of the use of QoS distribution based in the FSA-Ctrl architecture is given in Section 4. Section 5 presents concluding remarks.

2. Fundamentals: DDS and SWE

Most of the communications systems that provide support to the distributed control architectures need a module that hides the details of the components connections. Usually, when this module is separated from control components, it 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 are: 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]. There are a lot of interfaces and tools to develop a middleware. Some of the tools like JMS [5] and MSMQ [6] are generic protocols, and widely used on distributed systems.

To develop a distributed multi-agent system based on components, some components of the agents must adapt their technology to the communication interfaces. For example, if communications are based on CORBA [7], the multi-agent system must be implemented with the object-oriented programming technology. To avoid the use of a particular technology is usual the use of standardized protocols like FIPA [8].

2.1. Communications: the DDS model

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 [9]. 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 its communications 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 its 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 send the message without waiting for the application request.

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 [10]. From the communications viewpoint, QoS is defined as all the requirements that a network must meet to message flow transport [11].

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. Control: the SWE model

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 [12]. 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.

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 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” which 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 keep these aspects.

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) [13]. The architecture has two distinct areas: communications and control. QoS Policies connects both areas. Figure 3 shows details of the architecture.

The “Frame” component of the FSA architecture takes the same role of the “DomainParticipant” 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 communications 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 [14].

Control layer organizes the “Sensors” on a graph, called Logical Sensor Graph (LSG); details can be obtained from [15]. 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 communications components. A “Logical Sensor” can receive, or send, messages from, or to, another “Logical Sensors”.

3.1. Communications: adapters and LNT

Platforms that supplies communications 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 fits 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. Figure 4, shows the details of LNT structure.

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/*” and close a concrete window writing the close order at the logical data “root/home/effectors/site_A/windows/window_1”.

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 5). 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.
3.3. Conceptual model

Figure 6 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 LogicalSensors, 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.

4. Use case: redundancy management

One of the most common problems in distributed systems is the redundant information. Complex sensors produce a great amount of information. When a system moves a large amount of information, data can be replicated. Likewise, control components may require information with specific temporal characteristics and is possible that some messages are discarded. System must be able to manage these redundancy aspects. QoS policies can provide support to manage the redundant information on system.

Proposed architecture has on the QoS one of his strengths, because all components can manage time and message flow with the same interface, which allows detecting the redundant information in the most appropriate level. Figure 7 shows a very simplified case. The example uses a QoS policy called “TimeBasedFilter” that defines the temporary boundary in which intermediate messages are discarded. This policy has a unique parameter with the time boundary. Messages discarded can be different, but the fact that control sensors cannot process it, makes them redundant. The example describes a simple navigation speed control to a mobile robot.

In the example, DataWriter provides data to the motors at the same frequency, regardless the message arrival frequency from the control action. Likewise, is possible to select the frequency at which provide the speed information at different control components. In this example, control components, only acts when the value of time, negotiated in the QoS policy, is done. Combination of different values on different QoS policies provides many opportunities to manage complex systems.
5. Conclusion

This paper presents the implementation of component-based distributed intelligent control architecture with QoS support. Communications based on publish-subscribe paradigm and the DCPS standard endorsed by OMG allows to convert easily the system into another. Control based on SWE standard endorsed by OGC allows the high-distributed control algorithms.

The FSA-Ctrl architecture especially focuses on setting QoS management policies. With these policies, the system can make decisions about questions regarding the discrimination of redundant information or taking decisions about agents’ movement.

The FSA-Ctrl architecture gives several advantages, some of them are 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.

Moreover, the architecture 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 the agents’ control processes according to the restrictions.

The first version of a 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.

Acknowledgements

Middleware developed 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