CISTER
Research Center in
Real-Time & Embedded
Computing Systems

Technical Report

QoS enabled Middleware for Real-time Industrial Control Systems

Luis Lino Ferreira
Michele Albano
Luis Miguel Pinho

CISTER-TR-131003
Version:
Date: 10-09-2013

QoS enabled Middleware for Real-time Industrial Control Systems

Luis Lino Ferreira, Michele Albano, Luis Miguel Pinho

CISTER Research Unit

Polytechnic Institute of Porto (ISEP-IPP)

Rua Dr. António Bernardino de Almeida, 431

4200-072 Porto

Portugal

Tel.: +351.22.8340509, Fax: +351.22.8340509

E-mail:

http://www.cister.isep.ipp.pt

Abstract

In this paper we analyze some of the existing solutions for Message-Oriented Middleware (MOM), which can be used on industrial environments, and thatare, at the same time, capable of handling large quantities of data and of providing adequate Quality-of-Service (QoS) levels for its supported applications. We also make a proposal for the generic structure of a middleware layer supported on a MOM.

QoS enabled Middleware for Real-time Industrial Control Systems

Luis Lino Ferreira, Michele Albano, Luis Miguel Pinho
CISTER/INESC-TEC, ISEP, Polytechnic Institute of Porto
R. Dr. António Bernardino de Almeida, 431
4200-072 Porto / Portugal
{llf, mialb, lmp}@isep.ipp.pt

Abstract

In this paper we analyze some of the existing solutions for Message-Oriented Middleware (MOM), which can be used on industrial environments, and that are, at the same time, capable of handling large quantities of data and of providing adequate Quality-of-Service (QoS) levels for its supported applications. We also make a proposal for the generic structure of a middleware layer supported on a MOM.

1. Introduction

Industrial Systems have evolved from centralized systems to fully distributed systems with a mix of decentralized, cooperative and hierarchical applications. One of the main problems of orchestrating together an industrial system is the integration and interoperation of the different components and technologies. The problem is aggravated when QoS requirements are taken into account, since some of the subsystems must support real-time operations and/or reliability guarantees.

The distributed application driving the system will potentially be hierarchically structured into layers. Each subsystem belonging to a layer provides a set of interfaces, can communicate with higher hierarchical layers, and is also capable of cooperating with other systems in the same layers. Examples of these subsystems are: fieldbuses, sensor networks, industrial machines, geographical interconnections, mobile robots, control and monitoring applications.

The complexity of the interactions in the described system becomes much more acute when the environment is dynamic, such as current industrial plants, e.g. if robots connect with different plant areas and move between them. Consequently, complex communication, able of handling multiple different protocols and, at the same time, provide adequate QoS levels systems, must be implemented.

In this paper we propose an approach to solve the communications problem by layering on Message Oriented Middleware (MOM), which facilitates the integration of highly heterogeneous systems. The main goals motivating its use are to simplify distributing applications across heterogeneous operating systems, programming languages, computer architectures, networking protocols, and, at the same time, reducing the complexity on the interconnection functionalities, while providing a high level of scalability.

2. Message Oriented Middleware Solutions

A MOM communication bus can contribute with the following capabilities: i) hiding of the complexity of communication systems; ii) improved scalability; iii) reduces communications design complexity; iv) asynchronous and synchronous communication mechanisms; v) data format transformation (i.e. a MOM can change the format of the data contained in the messages to fit the receiving application [3]); vi) loose coupling among applications; vii) parallel processing of messages; and viii) intrinsic Quality of Service support.

A MOM usually supports one or more among three different communication paradigms: i) message passing (direct communication between applications); ii) message queuing (indirect communication via a queue); iii) Publish/Subscribe mediated interaction, where messages are published to "topics" and then subscribers receive all messages published to the topics they subscribed to.

One of the most interesting communication paradigm provided by a MOM is the Publish/Subscribe one, which provides asynchronous and highly scalable many-to-many communication model [1, 4]. One of its main characteristics is that the sender of the message, called publisher, is not aware of the identity of recipients (subscribers), since it publishes its messages to the MOM. Subscribers are enabled to receive the messages from the MOM by performing subscriptions of the information they are interested in.

The publish/subscribe communication paradigm decouples systems in terms of space, time and synchronization.

Several technologies implement these concepts, of particular interest, due to the maturity of solutions, are the Data Distribution Service for Real-Time Systems (DDS), Extensible Messaging and Presence Protocol

(XMPP) and the Advance Messaging Queue Protocol (AMQP), which are thoroughly analysed in [10].

3. Architecture for Industrial Control

The control of distributed industrial systems can be achieved by a collaborative hierarchically structured system, where interaction occurs through an external Message Oriented Messaging (MOM) bus, which connects actors on the same hierarchical level and connects to higher level MOMs.

The architecture for such system can comprise several hierarchical levels, where the components of each level are interconnected by a MOM-based messaging bus and the levels are interconnected through an extension of the lower level MOM until a Middleware Gateway Module on the higher level MOM. Figure 1 gives a high level view of the proposed architecture. In this architecture each messaging bus of level 1 links all devices in one area with one or more control applications (AP1 and AP2). In an industrial system an area can correspond to a manufacturing cell with its subsystems: machines, robots, human operator interfaces, fieldbus and sensor networks, etc. Level 0 corresponds to all middleware systems that exist inside mobile robots, industrial machines or any other kind of industrial subsystem.

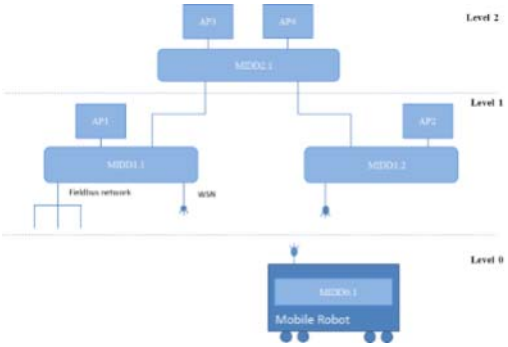

Figure 1. High level overview

We advocate using in each hierarchical level different MOM technologies, adapted to the specific requirements of the level. Inside each Middleware level there is also the need for all communications to adhere to a standard for information representation, to ease interoperability and extendibility of the resulting system. These standards might already be part of the MOM. For example DDS transfers data structures directly mapped into programming languages between the constituents of the distributed system. On the other hand, MOMs like RabbitMQ and XMPP enable just the transfer of custom messages, whose content are defined by the application layer and possibly conformant to XML-based standards.

The dynamics of modern industrial installations must also have the capability of easy and rapid system reconfiguration and addition of new devices, and robustness when confronted with the (possibly abrupt) removal of devices. These last two use cases are pertinent to mobile robots, which can move between different areas due to their intrinsic mobility. Therefore, the Middleware should provide the mechanisms for the fast removal of a device from an area, and its fast insertion into another area. This functionality implies the transparent reconfiguration of the device's communication connections. Devices that are driven by a control application (e.g. AP1 or AP2 in Figure 2) also require the transfer of their configuration and status information between the two areas. Also note that it might be necessary to transfer complex devices which internally are structured around another middleware, e.g. a mobile robot.

Industrial systems should also be capable of providing QoS guarantees to the applications. This latter requirement is hindered by the large scale of the systems at hand, due to their resulting complexity. Some works had addressed these issues [5, 6], but only considered performance issues and lacked on obtaining other QoS metrics.

3.1. Middleware Modules

To provide value, the Middleware must contribute on the aspects briefly described in Section 2. To that purpose we propose the internal architecture depicted in Figure 2, aimed at describing a level 1 middleware.

The main structural component of the architecture is the MOM, which connects all components and makes available a rich set of communication paradigms, supporting several different kinds of QoS parameters. Each level should have a different type of MOM, more adapted to the available resources and to the required performance. E.g. a level 0 MOM, which runs inside a robot, should be more focused on giving real-time guarantees and should consume fewer resources.

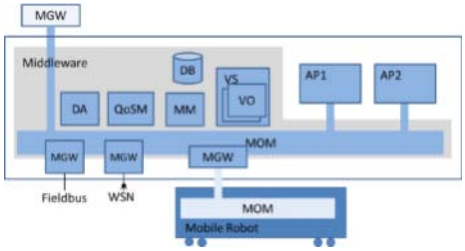

Figure 2. Main Middleware components

The virtualization of devices and subsystems will also benefit from the use of a Data Aggregator module that will be responsible for aggregating events from data obtained from multiple sources, in real-time, and also to perform some reasoning based on the data values, their temporal relation, etc. As an example, let us consider a leak in a pipe: it can be inferred from a sudden reduction on the pressure of the water in the pipe, together with sensor data notifying a higher humidity in the exterior.

The Middleware also has to connect to existing subsystems that communicate in custom protocols. The

connections are performed by the Middleware Gateway Modules (MGM), which translate the subsystem protocols to the protocol used within the Middleware. This module is also used to communicate with between a lower and upper level Middleware, meaning that the lower level middleware extends until the (more resourceful) upper level middleware. Usually this connection is used to receive high level commands and send reports. The rationale behind these modules is that they are the entrance point to the MOM for external data.

3.1.1. Virtual Object/Subsystem Abstraction

The Middleware provides the capability of abstracting devices and subsystems. A Virtual Object can represent a simple sensor or actuator, with its readings or commands, and its configurations. It might also represent a complex device, e.g. a mobile robot. A Virtual Subsystem represents a set of devices that operate in close cooperation. An example of this is a set of sensors and actuators in an automated warehouse. Additionally, the interface with such devices can benefit from the provision, by the Virtual Subsystem abstraction of common interfacing functionalities.

The communication with a device can be performed by other devices in the area, or by any of the control applications on the middleware (AP1 or AP2), always through the Virtual Object that abstracts the physical device. This architecture enhances the robustness of the system, its performance and its composability.

When a device is absent from a system, its Virtual Object will receive and store changes to the device state. These abstractions also provide a cache system that enables the fast access to device status and readings by several applications, thus avoiding sending reading requests to physical devices. Therefore, it should be possible to configure the Virtual Object with a sampling periodicity, or the Virtual Device should be capable of configuring the real device to send data periodically. Thus, the Virtual Object should also be capable of delivering the data liveness properties to the applications.

The Virtual Subsystem purpose is twofold: it is used to request data and/or services from a set of devices, or to a more anonymous “any device in an area”. In fact, direct communication between devices belonging to different areas is prohibited, i.e. all communications should be directed to the controlling applications of the area, which will dispatch the service requests to the proper robot(s)/devices. This way, requests to devices that are far away will be easier to manage, and the interactions will be loosely coupled, since the serving devices will be changed as needed by the control application.

3.1.2. The Message Oriented Middleware

The Messaging Infrastructure module links all components, connecting the applications with the devices and subsystem, with other modules and with upper level control systems. The use of a Message Oriented Middleware (MOM) allows reducing the complexity of inter-application communications. Such a communication infrastructure is capable of supporting both publish/subscribe and request/response communication paradigms. The request/response paradigm involves a client that produces a request and contacts a server, and a server that processes the request and sends back a response to the client. The publish/subscribe paradigm is more complex, it was mentioned in Section 2 and it allows a better scalability of the overall system since it allows for parallel operation, message caching and improved message routing.

Existing MOMs, such as the ones described in[10], provide functionalities to filter events. In such cases, the event subscriber can configure the MOM in order to be only notified if simple rules are met. For more complex rules the Middleware can make use of event processing applications [7]. A MOM is also capable of providing different kinds of QoS properties for the message queues administered by it.

3.1.3. Data Aggregation

The Data Aggregation (DA) module is capable of collecting data from different sensors and performing simple analysis over these data, like determining the maximum and minimum of a set of measurements.

It can also be used to configure the sensory system to autonomously perform such kind of actions, as proposed in [8]. The latter approach has the advantage of being capable of achieving higher performance while consuming less resources system-wide.

3.1.4. Mobility Control

In each area the robots are able to cooperate with each other through the MOM and they can also travel to other areas. The Mobility Control module will have to handle the mobility procedure necessary to transparently move the robot from one area to another. When a robot (represented as a Virtual System) moves from one area to another, a new Virtual System is created and all connections with the MOM must be reestablished, i.e.: the robot will start receiving commands from a different control application, but that change of control is transparent to the robot since all connections are semantically equal. Additionally, the data published by the robot, (e.g. its status) should start following to different, but equivalent queues, on the new level 1 middleware.

Therefore, the Mobility Control module will also have to check if the QoS requirements of the mobile robot can be guaranteed on the destination area and handle the case when it cannot guarantee the QoS requirements. This behaviour is similar to what has been proposed in [2].

3.1.5. QoS Manager

The QoS Manager is responsible for handling QoS requests for every message flow in the system. To that purpose, the establishment of a message flow (e.g. between one producer and 2 subscribers) should be authorized by this module.

This module should work based on a mathematical model of the system message flows and associated loads on CPUs executing the Middleware functions, which enables inferring if a new message flow can be admitted into the system or not. An alternative path is to use approaches similar to what has been proposed in [9], where each intervenient indicates a set of possible QoS configurations, each one with a different reward to the overall system. Then, algorithms calculate the QoS degradation needed to accommodate the requirements of the message flow.

3.1.6. Applications

We can expect to have applications that take care of avoiding collisions between robots, system monitoring and controlling the interaction between mobile robots and industrial machines, just to mention a few.

These applications can benefit from the services provided by the middleware, namely, the Data Aggregator, the Virtual Object or the Virtual System, which can make application development easier. Anyway, it is also important to note that the Middleware, particularly the MOM, can be used to efficiently support distributed application without centralized control, like the ones proposed in [10].

3.1.7. Hierarchical interaction

The architecture being proposed requires the interconnection of different hierarchical levels. We assume that each level can use a different kind of middleware, in which nodes communicate using different protocols. Therefore, specific Middleware Gateway Modules (MGMs) converts between different protocols. Also note that each interconnection requires one MGM only, on the upper level middleware, and the lower level middleware is allowed to extend until the broker. The rationale for this design decision is that it is expectable that the upper level middleware has more resources at its disposal, hence it can take care of performing the protocol translation.

The MGM aggregates messages directed to different areas, and it vehiculates them through upper-level messaging buses modules, using industry standards wherever possible.

4. Conclusions

In this paper we proposed a multi-level architecture for the support of an industrial control architecture, where each level is structured upon a MOM. The architecture is particularly suited for systems which comprise mobile robots in an industrial environment, that connect to multiple types of devices and systems, like wireless sensor networks or industrial machines.

This paper clearly identifies that there is a lack of a sufficiently robust timing analysis which can leverage the use of MOMs in real-time systems.

Acknowledgment

This work was supported by National Funds through FCT (Portuguese Foundation for Science and Technology) and by the EU ARTEMIS JU funding, within ENCOURAGE project, ref. ARTEMIS/0002/2010, JU grant nr. 269354 and Arrowhead project, JU grant nr.332987.

References

[1] Embedded iNtelligent COntrols for bUildings with Renewable generAtion and storaGE (ENCOURAGE), http://www.encourage-project.eu

[2] Gonçalves, J., Ferreira, L., Pinho, L., Silva, G., "Handling Mobility on a QoS-Aware Service-based Framework for Mobile Systems", IEEE/IFIP International Conference on Embedded and Ubiquitous Computing, Hong-Kong, China, Dec 2010, pp. 97 – 104

[3] E. Curry, D. Chambers, G. Lyons, “Extending Message-Oriented Middleware using Interception”, 3rd Int. Workshop on Distributed Event-Based Systems (DEBS '04), ICSE '04, Edinburgh, Scotland, UK, 2004.

[4] P. Th. Eugster, P. Felber, R. Guerraoui, A-M. Kermarrec, “The Many Faces of Publish/Subscribe”, ACM Computing Surveys 35(2), 2003, pp. 114-131.

[5] K. Sachs, S. Kounev, A. Buchmann, “Performance modeling and analysis of message-oriented event-driven systems”, Software and Systems Modeling, Springer-Verlag 2012.

[6] Happe, J., Friedrich, H., Becker, S., & Reussner, R. H. “A pattern-based performance completion for Message-oriented Middleware”, 7th international workshop on Software and performance, June, 2008, pp. 165-176.

[7] E. Wu, Y. Diao, S. Rizvi, “High-performance complex event processing over streams”, 2006 ACM SIGMOD international conference on Management of data (SIGMOD '06). ACM, New York, NY, USA, 407-418.

[8] Pereira, N., Tennina, S., Tovar, E., "Building a Microscope for the Data Center", 7th International Conference on Wireless Algorithms, Systems, and Applications (WASA 2012) Aug. 8-10, 2012, Yellow Mountains, China. (Best Paper Award)

[9] Nogueira, L., Pinho, L., Coelho, J., "A Feedback-based Decentralised Coordination Model for Distributed Open Real-Time Systems", Journal of Systems and Software, Volume 85, Issue 9 (2012), pp. 2145-2159.

[10] Alkhawaja, A., Ferreira, L., Albano, M., "Message Oriented Middleware with QoS Support for Smart Grids", InForum 2012 conference on Embedded Systems and Real Time.