Usage of DDS Data-Centric Middleware for Remote Monitoring and Control Laboratories

Marisol García Valls, *Senior Member, IEEE*, and Pablo Basanta Val

***Abstract*—Communication middleware technologies are powerful instruments to develop and deploy remote laboratories for students to practice the control of real application cases. Middleware allows the rapid development and deployment of actual distributed settings since it abstracts the specifics of the different hardware platforms and communication media involved in the access of students to the lab material. This paper describes the practical settings carried out in the a case of a successful laboratory experience that allows students to simulate the on-line monitoring and control of remote systems, i.e., the train traffic in a simulated metro system. The innovative contribution of this experiment is the usage of data-centric middleware based on the publish/subscribe paradigm that has real-time properties. Up to our knowledge, such deployment with this particular middleware type has not been done, and it has given rise to a flexible setting that grows with the contributions of students. We have validated the idea by gathering the feedback of students with respect to the operation of the assignment, showing that this motivates the students' learning and enrollment in the assignment.**

***Index Terms*—DDS, middleware, networked embedded systems, remote laboratory, remote monitoring and control.**

I. Introduction

THE usage of information and communication technologies plays a significant role in distance learning, and this has been specially recognized and implemented over the last few decades. Scientific education has been enhanced by the progressive adoption of information and communication technologies (ICT) that have enabled the setting and deployment of laboratory-based courses that can be attended remotely [1], [2]. Although it cannot replace the traditional learning as described in [3], remote education can allow students to freely manage their learning time, and it can impressively motivate their learning experience if the proposed exercises are stimulated by adequate graphical means emulating the real world.

Different types of technologies have appeared that enable this possibility [2], [4]: from procedural to object-oriented languages, easy-to-use and easy-to-program environments with powerful user friendly "graphical" interactive means, and tools for enabling web-based communication between the lab setting and the student. Client and server technology has also improved considerably, aided by the increase in the computation power of the hardware nodes, the improvement of the network communication, and the de facto globalization of the Internet access.

Manuscript received August 07, 2011; revised February 14, 2012 and April 13, 2012; accepted June 08, 2012. Date of publication August 01, 2012; date of current version December 19, 2012. Paper no. TII-11-402.

The authors are with the Department of Telematics Engineering, Universidad Carlos III de Madrid, 28911 Leganés, Madrid, Spain (e-mail: mvalls@it.uc3m.es; pbasanta@it.uc3m.es).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TII.2012.2211028

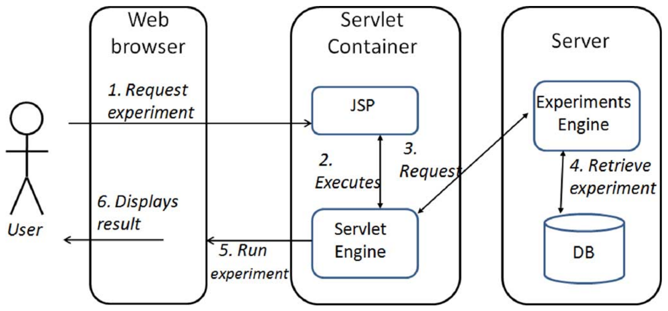

Fig. 1. Web technology for remote execution of experiments.

Middleware technologies are the basic building blocks of distance learning deployments, and they have been extensively applied over the last two decades in different types of distributed applications. Middleware is a high-abstraction software layer enabling the development of networked and distributed systems in an easy way, by abstracting programmers of the specifics of the platforms that must be interconnected [5]. The details of the different operating systems and hardware that must be interconnected are overcome by the fact that the middleware takes care of the transformation between different network protocols, data formats, and most basic low-level synchronization problems.

Over the last decade, enhanced technologies over remote procedure call middleware and socket-based middleware have appeared as web technologies. They have gained an impressive popularity since they allow using web browsers to access any resource that can be located, for instance, in a remote server, download part of its content, and execute it locally in a transparent way, as shown in Fig. 1.

Multi-tier enterprise architectures such as J2EE [6] have brought about a new vision for client systems by embedding the application logic fully in the server side. This way, clients have become lightweight front-ends that are user-friendly interfaces capturing the requests of users and forwarding them to a central server that executes the solicited service or functionality. This powerful paradigm has also scaled to the remote education platforms where the students practically only need a web browser to be able to connect to the remote laboratory.

In the same way, professors are able to perform most of the software maintenance in a remote way as well as the remote tutoring of the students by means of instant chat systems or forums. Some tools for this are already available as the virtual learning environment of Moodle [7].

This paper presents a remote laboratory deployment where the emphasis is placed on the enabling communication middleware technologies and how they have made possible the partial deployment of the presented educational experience. This setting is specially meant for master thesis students that engage on a remote monitoring and control master thesis topic, and the deployment is enhanced by the contribution of the different enrolled students. The setting contains easy to use middleware technologies and graphical emulation of real settings that have proved to motivate the student learning experience.

This educational setting is briefly explained in Section II. Section III elaborates on the selection of middleware technologies that has been made for this laboratory. Section IV describes the architecture of the laboratory. Section V presents the conclusions on the increased motivation of students. Section VI presents a survey of related work describing some similar deployments and highlighting their differences with respect to the one presented in this paper. Finally, Section VII concludes the paper.

II. System Description

This practical laboratory was designed as part of the graduate degree of Electrical Engineering as a basic technological setting for students to carry out the final master thesis. For this experience, the contributions of the different master thesis add to its implementation and, especially, the feedback about the idea of the lab experience is enriching for its progressive enhancement. Typically, the feedback provided by students gives rise to the assignments that follow.

The lab is offered to master thesis students. Typically, the master thesis is an additional subject where the students receive an assignment as a mainly practical laboratory to be worked on under the guidance of the advising professor. Most of the time, students are performing personal work towards the goal of finalizing the assignment and obtaining their diploma. If they are faced with some blocking points or lack of knowledge on some specific part, they are guided by the advisor professor in order to unblock their learning experience allowing them to continue to find the solution by themselves.

In such an autonomous effort, video technologies can help the student to find their way in a more didactic and motivating way. As a result, the practical setting to be described combines graphical technologies for rapid feedback and visualization of the result of their work and a set of technologies for enabling the communication framework between the student, the advisor, and the feedback that the assignment gives to both, the student and the advisor.

The basic structure of this laboratory has been used for four consecutive academic years in the subject of Final Master Thesis of the Electrical Engineering degree. It began in September 2007 as part of a pilot experimentation to motivate students to take assignments on distributed real-time systems for control and monitoring of remote embedded infrastructures. The remote infrastructures to be monitored can vary depending on the student's interest: railway traffic control and monitoring and remote surveillance and actuation of some physical process have been the ones proposed up to present.

The motivation of engineering students is likely to be increased when the laboratory combines two fundamental characteristics: 1) it is hands-on-work involving a high degree of practical effort; and 2) it emulates, simulates, or is a real-world environment. Since the type of exercise to be implemented by students is the remote control of a simulated real-life networked embedded system, the assignment focuses on two main targets, given here.

- Remote real-time video surveillance. In this version of the lab, the students must synchronize different data flows of video, prioritize them according to different parameters (video source, type of video and compression format), and tune the real-time parameters of the enabling backbone middleware for each frame or frame set. Also, different nodes have distinct computation requirements; therefore, the nature of different nodes (e.g., sensors or PC) must also be considered, and there resource usage must be efficient (e.g., memory, CPU cycles, and network bandwidth).
- Remote monitoring and control of a simplified metro system. When this exercise is used, the students focus on the fast detection of alarms, the real-time communication of alarms, the timely monitoring of the train traffic in the system, and the correct control of the train circulation on the tracks.

Both use cases teach the students the basics on the following.

- Usage of ICT for remote real-time communication. They have to implement the remote communication with the lab servers, and also they must tune the communication parameters of the selected laboratory middleware (e.g., message delays, priorization, or queues). The interface to the servers functionality is specified in the assignment.
- Programming of reduced footprint embedded nodes. The lab also deals with the programming of sensors. Sensors are used to trigger alarms in the system (motion detection and temperature measurements).
- Applying the logic of control of concurrent functionality over a real-time operating systems. Students must be able to correctly synchronize the operation of the system by controlling the semaphores or other synchronization constructs to adequately control the train tracks or the video transmission.

The overview of the laboratory deployment is shown in Fig. 2. The assignment can be both performed locally at the university premises or at the student site. Only a network connection is required. The used middleware enables to operate from remote sites taking care of the low level communication details. One of the basic tools that is provided in the lab is the DB (a *data base* of the experiments) with the logic of the experiment, programmed by the professors with a well specified logic. In this paper, we focus on the assignment of the trains control system to illustrate this educational experience. Therefore, the DB system contains the actuation pattern, which is given here.

- Timing of train operation, train delays, number of trains per line, number of lines, and graphical setting
- Number and type of video streams, relative priority, etc.
- In both cases (train control system and video surveillance), a video camera is connected to the lab for remote visualiza-

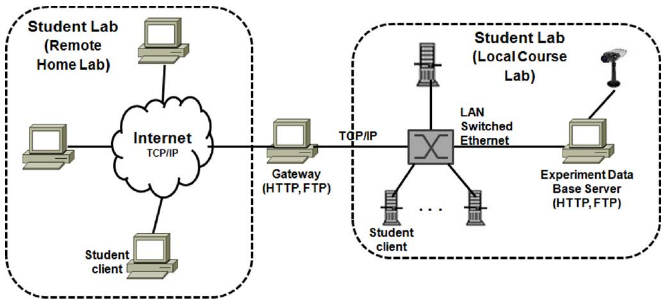

Fig. 2. Remote laboratory deployment.

tion of the local premises and interaction with the professor at the specified times when s/he is physically in the lab.

III. Middleware Technology Selection

The main enabler of this setting is the *communication middleware* technology to be selected. Students must develop the control operations of the trains and the communication establishment using the laboratory middleware. There are two possibilities for enabling the communication between nodes:

- Working at *network protocol* level using RTP [16] or a higher level with RTSP [17]. This can be very efficient since the processing is done directly in the network protocol stack. However, it lacks flexibility since any change may imply re-coding the protocol as needed; therefore, it does not scale well even for minor modifications.
- Working at *middleware* level. This is a very flexible solution since the middleware abstracts the end nodes from the details of the lower level communication network [5]. However, a thorough selection of the middleware technologies must be made for two main reasons. On the one hand, most middleware rely on the increasing computation power of the hardware and therefore do not consider efficient resource consumption as a driving force. Consequently, they become heavy software layers only concerned about interoperability. This is, for instance, the case of OSGi [8], CORBA [12], AJAX [10], even RT-CORBA [11], and JMS among others. On the other hand, most application domains do not impose restrictions on timely issues, although users are becoming less tolerant to any delays in interactivity or responsiveness. For this purpose, new middleware technologies have appeared as the simplifications of CORBA for embedded systems [13], ICE and ICE Storm [19], or DDS [15].

Also, the programming language characteristics as well as the facilities that it brings to easy programmability are one of the key issues that influences its success. The Java technology framework allows the easy implementation of sophisticated architectures with powerful interfacing, communication, and graphical capabilities. It is one of the most widely used in remote laboratory deployments for setting up of the experimental engine and user interactive framework as evidenced in [1]. It is also easily pluggable to the specific platforms for different control experimentation such as MATLAB or SIMULINK by means of language bridges. Moreover, it contains different versions for different node types with heterogeneous computation power as sensors, actuators, servers, laptops, or cameras. Also, the different packages as J2EE [6], J2SE [22], and J2ME [21] integrate various middleware versions. For example, Java's RPC (called Remote Method Invocation in Java: RMI [9]) is available in J2SE and the publish/subscribe version (JMS [14] or the Java Messaging Service) is only available in the J2EE version as an optional package.

A. Communication Middleware Paradigms

The selection of the main communication middleware is of great importance in a remote laboratory architecture since the level of flexibility, decoupling, and efficiency depends on the underlying communication model.

Clients and servers can communicate either synchronously or asynchronously depending on the selected and needed characteristics. Synchronous communication may be done at clock-level (as in time-triggered communication or in schemes where, at least, logic clock synchronization is required) or at a higher level based on RPC-like communication. Asynchronous communication allows information exchange where there are no bounds on either process execution speed nor message transmission delays, and clocks can have arbitrary drift rates. Asynchronous communication can be implemented in two ways. On the one hand, with events there is no need to have queue at the receiving end, so the communication fails if the receiver is not available. On the other hand, by using messages, senders do not need to know the identity of the receivers but only of the message queue where the receivers are waiting. The middleware architecture related to this interaction model is MOM (Message Oriented Middleware, MOM) based on data exchange via higher abstraction entities: data objects or messages.

How the different middleware technologies make use of the different interaction models and communication paradigms is important to efficiently implement the communication. Following, the basic middleware interaction models are presented: RPC, DOM, and MOM.

RPC provides a synchronous communication model based on invocation of code which resides in a remote machine in the form of a procedure (function or method, depending on the programming language used). This scheme is available on a huge number of operating systems. RPC-based middleware (such as CORBA [12] or ICE [19]) provides three different services: definition of interfaces, marshalling and unmarshalling services, and network communication. Therefore, clients use the remote calls in the same way as a local call. However, a purely synchronized middleware as an RPC-based one offers low efficiency in large networks, and it only allows the development of tightly-coupled applications.

Distributed Object Middleware (DOM) was the next natural step of RPC. It brought in the facilities of object-oriented programming as encapsulation, support for inheritance, object references, and exceptions. This is the case of Java RMI [9]. However, it lacks scalability since object middleware may not always be applicable to non-object-oriented environments and programming languages. This is a synchronous scheme in its basic form, and later it evolved to support asynchrony.

Message Oriented Middleware (MOM) [23] allows the communication of data stored in the form of messages. In the middleware, messages are software structures that encapsulate data to be exchanged. MOM relies on the usage of message queues which provide asynchronous communication between sender and receiver. They have no natural support for synchronous communication directly between sender and receiver. MOM is usually used to provide event notification and requests for service execution.

B. Publish/Subscribe Interaction Paradigm

Publish/Subscribe (P/S) interaction paradigm provides an asynchronous means for communication of specific data types. P/S architectures allow constructing a communication cloud among several nodes that are interested in specific data contents. Where data comes from and where it goes to is not the focal point. This is a powerful means for building decoupled systems. In P/S, there are *publishers* that are the entities that produce information and publish data in the system, and *subscribers* that are the consumers of such information generated in the system. Subscribers interested in particular data types will register to them.

As a result, P/S paradigm offers a flexible interaction model in distributed applications, allowing the development of a complex system made of remote decoupled components that interact based on their interests for specific data contents. DDS standard [15] (*Data Distribution Service for Real-Time Systems*) implements this model of interaction, and it also offers specification of QoS parameters. Also, Java Messaging Service [14] offers a centralized version of the P/S paradigm, as well as ICE Storm [19].

IV. Software Architecture

A. Overview

The software setting follows a hybrid architecture comprising a client/server structure that uses as underlying communication paradigm a publish/subscribe model.

- The client-server part is for the experimental data delivery and graphical structure parameterization.
- The publish-subscribe model is the main building block of the communication and coordination among the different experimental components allowing also the data exchange between student and advisor. Interested nodes subscribe to the data objects that are of their interest; for example, every time that a train is prepared to arrive to a station, the local control of the station receives the *train data object* with its associated parameters of relevance.

In the following, the software architecture is specified showing the precise technologies selected for the experiment for the train control system. It must be noted that students have to control the operation of trains of two metro lines that intersect in different stations. It is assumed that all metro lines are located at the same ground altitude, so that each metro station uses the same tracks for trains of different lines (see Fig. 3).

The software units shown in Fig. 3 communicate via message exchange where publishers and subscribers have been identified as well as the data types to be exchanged. Video data is also transmitted at run-time which allows live monitoring of the remote premises, and it can also be used for online advisor assistance to the students that need support to unblock a particular point in their work.

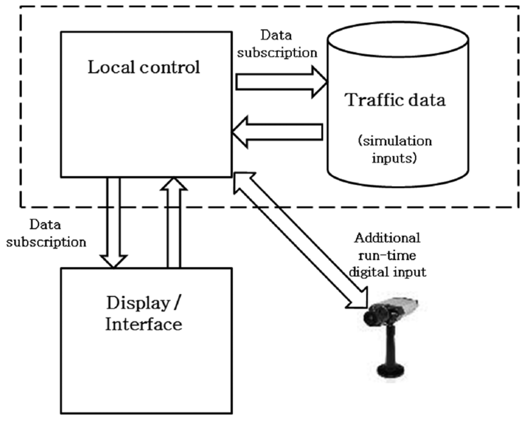

Fig. 3. Functional blocks in the laboratory architecture.

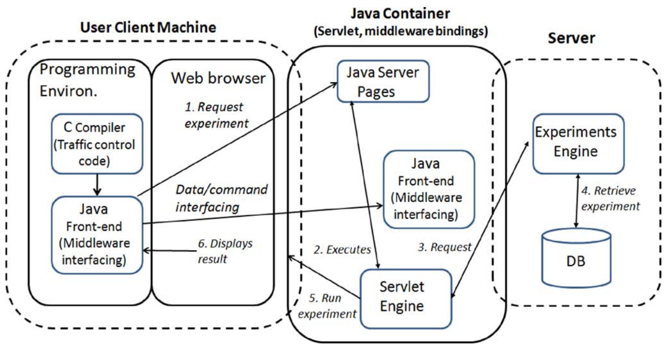

Fig. 4. Technology platform for remote execution of experiments.

The software units deployment is shown in Fig. 4 containing the view of the client machine (the user) and the server. They use Java libraries as JSP (Java Server Pages) and other web related technology embedded in the basic Java distribution to offer flexible web-based visualization and communication.

Therefore, the key element used is Java framework. The client is a lightweight element based on a web browser front-end, to which the student control code is hooked to operate against the database (DB) that contains the traffic information that is generated.

B. Data-Centric Elements

The relation of the software architecture with its data-centric elements is shown in Fig. 5, and it is later extended in Fig. 6. As the data-centric middleware, we have used Data Distribution Service (DDS) [15] for distributed real-time applications, which is an OMG standard and is becoming very popular for real-time communication through the inclusion of a comprehensive set of quality of service (QoS) parameters to fine tune the communications (enabling different levels of reliability, liveliness, timeliness, among other options). The basic event/command information and trains positioning is communicated via DDS.

The display units are client operator nodes. They allow the manual control of a human operator. Therefore, they receive information on the train positioning and display it to the operators.

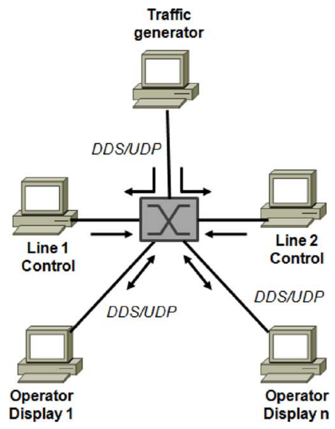

Fig. 5. Laboratory deployment at the university premises. Switched Ethernet is used for control over traffic shaping.

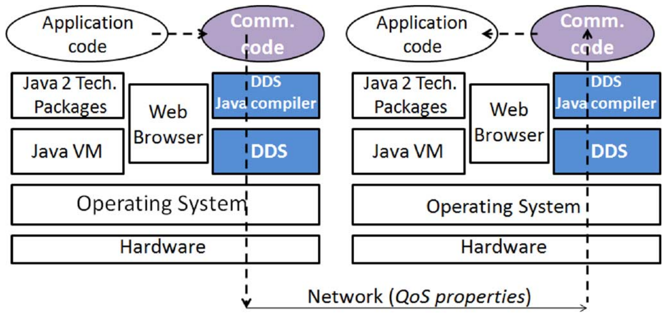

Fig. 6. Integration of P/S middleware and Java technology.

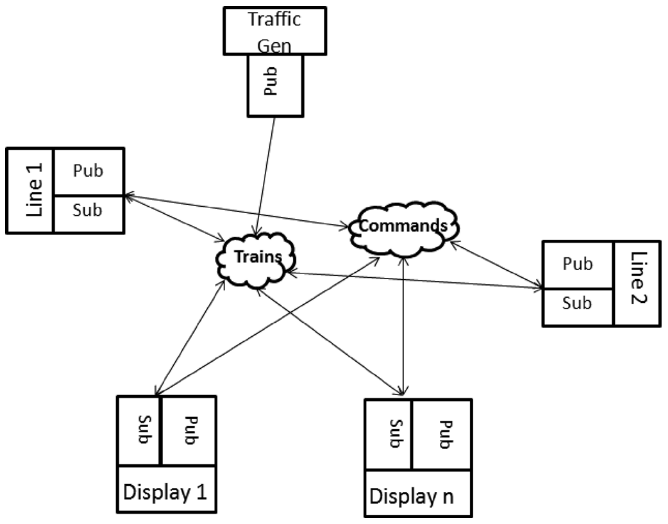

Fig. 7. Data-centric communication for traffic control.

Also, they provide a basic interface for displaying information on the line traffic and basic station control operations. The interface is lightweight, developed in Java using Swing and AWT.

The predominant technology chosen has been Java on a multi-tier architecture based on a modified version of lightweight clients that also contain the control execution programmed by students to control the traffic of trains.

Integrating DDS code in the software architecture of our lab deployment has the structure shown in Fig. 6. In it, the Java 2 technology packages contain the Java front-end interfacing libraries and the processing libraries as the JSP (Java Server Pages and its corresponding engine).

The data-centric middleware based on DDS allows to use data "topics" as the center of the communication interactions between end-nodes; this is later shown in Fig. 7. In the case of our lab deployment, end nodes are student computers and laboratory servers connected by DDS middleware. If new students join the laboratory experience, their software automatically joins lab by subscribing to the data topics that are created.

The traffic data contains the circulation pattern of the trains. This is typically a server running an ix86 connected via network links using TCP/IP to the different remote stations. It receives the positioning information of all trains and any alert or abnormal event that must be handled. This unit is also connected to the local control units that are the client nodes at each station storing the information on train traffic, and setting the emergency semaphores in special situations. The local control unit would, in reality, be connected to a central control unit that would generate the real commands for the trains circulation. The data topics used are shown in Fig. 7. They are mainly train data objects and commands issued to control the operation

In the case of unexpected errors, local controls at stations could decide to set to red some entrance or outgoing semaphores to stop train traffic. Students have to control the traffic flow and system execution of both types of stations (*isolated* stations that are part of just one metro line and *shared* stations that are part of two metro lines). There are semaphores located at the entrance of each station that must be controlled and set to red or green depending on the state of the station, state of the trains that arrive and depart from it, and the events (alarms and extra delays) generated by the traffic data subsystem (EIS). Special attention is put by students to shared stations.

Fig. 8 shows a display of the metro system with two lines (red and blue) and two trains in each. Green stations are shared.

Trains are represented as flags; each train circulates over a specific metro line. In this way, empty flags are the trains of line 1 and dark colored flags are the trains line 2. If ever two dark colored flags happen to be in the same node, the problem has not been correctly solved since line 2 does not allow bidirectional traffic. Also, if two different flags coincide in the same metro station, the solution is erroneous, and it should be reprogrammed.

Following, a photograph of a student site at the lab is shown in Fig. 9.

Clients nodes can be heterogeneous nodes with very different processing power. The main nodes are PCs with ix86 architecture; also, smaller nodes can be Java bytecode Ajile processors with an attached LCD display for basic interaction (shown in left part of Fig. 9, bottom left corner). Moreover, students can model their control solution using a UML-based tool. The interesting property of these tools is that they show at a glance the structure of the code that will be executed. The professor can easily extract the reasoning of the student about the solution that he or she has programmed and give the feedback in a fast way to students.

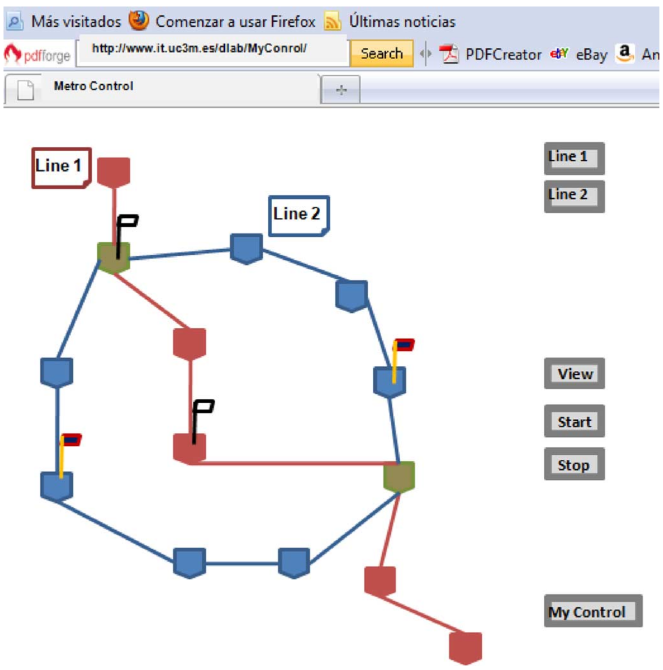

Fig. 8. Display node visualization.

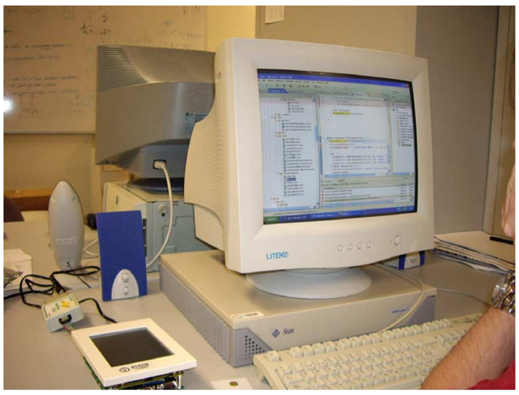

Fig. 9. Client node examples.

V. Results of the Experience

The main result that can be stated after the years of experience is that students have shown to be motivated by these experimental setting to perform their Master Thesis Work. Results of the laboratory experience are given in the following axes: 1) students motivation source for choosing this particular laboratory (since it is an optional assignment); 2) the elements that have influenced mostly the selection of this assignment; 3) the number of students that have joined the assignment due to the different particular motivation sources; and 4) the difficulty estimated by the students with respect to each particular key element of the lab. All of these axes together directly influence the number of students that have joined the lab voluntarily (shown in Table I as "*# Students acquired*"). Following, results based on experience after some years of this setting are shown in Table I.

Students joining this assignment have a major background on information and communication technologies, electrical engineering, and computer science. As a consequence, the goal has been to carry out a project on the application of ICT to remote

TABLE I
Laboratory Results

Motivation Source	Influence on choice	# Students acquired	Difficulty
Reality emulation	25%	15	1
Control aspects	10%	9	5
Middleware Tech.	20%	13	2
Graphics	10%	10	3
Remote access	15%	14	1
Java tools	20%	12	1

control of physical processes; therefore, the assignment constitutes a step towards their attraction to the control area from an ICT point of view, which is at the level of their knowledge and motivates clearly their enrollment. As shown in Table I, most students were mainly motivated to choose this assignment by the emulation of a real-world case; this circumstance that applied to the 25% of the enrolled students together with the fact that they were really motivated to use middleware technologies and deepen in their knowledge (the later applying to 20% of them) and the fact that using Java tools was also appealing to them (20%) shows that the use of ICT was a clear point in favor for acquiring students (a total of 15 during these years). The level of difficulty perceived by students (1 being the easiest to use/program and 5 the value for the highest difficulty) also shows that middleware technologies appear to them as easy to use tools to obtain the most out of a simple interface for communication.

It is, therefore, a suitable way to attract students towards the remote control of systems from an ICT perspective which is appears to be the most appealing subject our students. Their knowledge is therefore enhanced in a domain which proposes a small scale real-world case. As a summary, it has shown to be an attractive assignment for students.

VI. Related Work

Typically, remote laboratories have been architected as a specific solution to solve a specific problem. It is possible to find in the literature a large number of experiences reported on a wide range of subjects (e.g., electronics, robotics, industrial informatics, and mechanics) as shown in [2]–[4]. However, it is very common to find these reported experiences bounded to a limited set of requirements, providing solutions that are fit for a specific setting. In this side, we can find a considerable number of work and experiences reported in the literature as [25], [27], [29], among others. Due to their specific nature and precise target domain, little attention has been paid on very important characteristics for software intensive domains as scalability, maintainability, security, and flexibility of the deployed infrastructure. A laboratory deployment that lacks these characteristics is rather closed to new experiments and it may even be case that it does not support any minor modifications to its current structure that might be of educational interest for students.

Over the last years, some joint efforts have been undertaken with the goal of studying the road towards obtaining more reusable laboratory systems and deployments. In this way, works such as [24], [26], and [28] describe a number of general aspects to be considered in the design of remote laboratory architectures that are based mainly on an iterative process for the technology selection. Also, the structure of the deployments is, in some cases, designed around some specific software technology; this is the case of [25], where the Java programming language and associated APIs is used for development of a user front-end connected via COM to a MATLAB-based control engineering node; this mainly shows the possibilities of a remote control a Java client server application for interfacing with the user over a Windows operating system.

Also, the connectivity issues from a front-end user desk to the controlled object or controller process (that resides in a remote node) has been targeted by some works as [26], where important implications of using Internet protocols for communication are put forward.

As a consequence and, up to the best of our knowledge, the existing remote laboratory experiences do not cover the properties that we have presented in the experience described in this paper. On the one hand, existing solutions focus mainly in providing powerful front-ends and the middleware technology used for enabling the communication between the server side that provides the experience and the user is not further considered; further examples of this are given in [29] and [30]. In this paper, we focus and describe the benefits of using a data-centric middleware for the remote laboratory deployments since this paradigm supports naturally the scalability problem of the end-user population. Also, our work is based on a standard solution that has QoS parameters as DDS that allows to configure the communications to provide quality interaction between the user or student and the back-end server. Currently, we have extended the proposed middleware with real-time reconfiguration capacities as described in [31]–[34]. The data-centric nature of the middleware used allows to have a flexible deployment that can easily grow with the contribution and feedback of students. Finally, we provide a unique laboratory experience where students program the train traffic control of a simulated metro line that, up to the best of our knowledge, has not been presented in this format (neither with our technology selection nor with our goal of teaching distributed control and middleware programming to students).

VII. CONCLUSION

We have presented a laboratory for educational purposes for the final master thesis of engineering students that wish to deepen in their knowledge of ICT for control and monitoring infrastructures. We have deployed an infrastructure that is intensive in ICT usage, and that has been partially deployed during four years. The infrastructure is growing with the students involvement and feedback.

Our experience using this educational setting has shown that the usage of flexible ICT-based infrastructures is highly appealing for students, and, as a consequence, their learning process becomes clearly more effective and motivating.

The presented laboratory experience has an important dimension on the facilitation of the remote education by means of the distance interaction with the university central deployment and also, it is important the interaction with the professor and advisor. Students can autonomously set their home client that will interface with the remote facilities.

We have focused on the innovative aspect of applying data-centric middleware to offer a lab setting for exercising the remote monitoring and control of systems. We have not found this idea in any other similar deployment. We have also described and measured some relevant quantifiable axis of it.

This experience has shown to be attractive to students and motivating. Their knowledge on control and monitoring of infrastructures is gain at the same time as knowledge on middleware technologies that, at the end, become a simple tool to seamlessly achieve the goal that is desired: to correctly control and monitor the train traffic. The experiment is alive and continues to grow with the feedback of students and professors. In the future, it is expected to gain on self maintenance.

REFERENCES

[1] P. Penfield and R. C. Larson, "Education via advanced technologies," *IEEE Trans. Education*, vol. 29, no. 3, pp. 436–443, Aug. 1996.
[2] L. Gomes and S. Bogosyan, "Current trends in remote laboratories," *IEEE Trans. Ind. Electron.*, vol. 56, no. 12, pp. 4744–4756, Dec. 2009.
[3] P. Joo-Hyun, K. Pang-Ryong, and L. Hong-Woo, "Empirical study on the enhancement of the quality of cyber education," in *Proc. Technol. Management for the Global Future*, Jul. 2006, pp. 1273–1284.
[4] M. J. Callaghan, J. Harking, T. M. McGinnity, and L. P. Maguire, "Intelligent user support in autonomous remote experimentation environments," *IEEE Trans. Ind. Electron.*, vol. 55, no. 6, pp. 2355–2367, Jun. 2008.
[5] R. Schantz and D. Schmidt, "Middleware for distributed systems. Evolving the common structure for network-centric applications," in *Encyclopedia of Software Engineering*, J. Marciniak and G. Telecki, Eds. New York: Wiley, 2001, pp. 43–48.
[6] SUN Microsystems, Reference J2EE v1.4 Documentation Jul. 2011 [Online]. Available: http://java.sun.com/j2ee/1.4/docs/
[7] "Moodle community," Moodle in Education, Jul. 2011 [Online]. Available: http://docs.moodle.org
[8] OSGi Service Platform Release 4. ver. 4.2, OSGi Alliance, May 2011 [Online]. Available: http://www.osgi.org/download/osgi-early-draft-2011-05.pdf
[9] Java Remote Method Invocation. ver. 1.5.0, Sun Microsystems, Jul. 2011 [Online]. Available: http://download.oracle.com/javase/1.5.0/docs/guide/rmi/spec/rmiTOC.html
[10] L. D. Paulson, "Building rich web applications with AJAX," *Computer*, vol. 28, no. 10, pp. 14–17, Oc. 2005.
[11] "Real-Time CORBA Specification v1.2 Formal/05-01-04," OMG, May 2005.
[12] "Common Object Request Broker Architecture: Core Specification," OMG, Object Management Group, 3.0.3 formal/04-03-12 edition, Mar. 2004.
[13] "OMG's CORBA/e Middleware Specification for Distributed Real-Time and Embedded Systems," OMG, Aug. 2009 [Online]. Available: http://www.omg.org/corba-e/index.htm
[14] "JMS, Java Messaging Service v1.0.2," Sun Microsystems, Jul. 2011 [Online]. Available: http://download.oracle.com/docs/cd/E17802_01/products/products/jms/javadoc-102a/index.html
[15] "Data Distribution Service for Real-Time Systems," OMG, 1.2 formal/07-01-01 edition, Jan. 2007.
[16] H. Schulzrinne, S. Castner, R. Frederik, and V. Jacobson, "RTP: A transport protocol for real-time applications," *Network Working Group*, Jan. 1996, RFC 2326.
[17] H. Schulzrinne, A. Rao, and L. Ranphier, "Real-time streaming protocol (RTSP)," *Audio-Video Transport Working Group*, Apr. 1998, RFC 1889.
[18] J. A. Clavijo, M. Segarra, R. Sanz, A. Jiménez, C. Baeza, C. Moreno, R. Vázquez, F. Díaz, and A. Díez, "Real-time video for distributed control systems," *Control Eng. Practice*, vol. 9, no. 4, pp. 459–466, Apr. 2001.

[19] "Distributed Programming With ICE," ZeroC, ver. 3.3.1, Mar. 2009.

[20] D. Brookshier, N. Krishnan, D. Govoni, and J. C. Soto, "JXTA: Java peer to peer programming," *Sun Microsystems*, Jun. 2002.

[21] "Java ME and Java Card Technology," SUN Microsystems, Jul. 2011 [Online]. Available: http://www.oracle.com/technetwork/java/javame/index.html

[22] JSR-000270 Java (TM) SE 6 Release. SUN Microsystems, Jul. 2011 [Online]. Available: http://jcp.org/en/jsr/detail?id=270

[23] P. Maheshwari, H. Tang, and R. Liang, "Enhancing web services with message oriented middleware," in *Proc. IEEE Int. Conf. Web Services*, Jun. 2004, pp. 524–552.

[24] J. García-Zubía, P. Orduña, I. Angulo, J. Irurzun, and U. Hernández-Jayo, "Towards a distributed architecture for remote laboratories," *Int. J. Online Eng.*, vol. 4, pp. 11-14–, 2008.

[25] P. Bisták and P. Folvar, "Remote laboratory java server based on JACOB project," *Int. J. Online Eng.*, vol. 7, no. 1, pp. 33–36, 2011.

[26] A. Azad, M. Auer, and V. Judson Harward, "Internet accessible remote laboratories: Scalable E-learning tools for engineering and science disciplines," *Int. J. Online Eng.*, vol. 6, no. 1, pp. 63–66, 2010.

[27] G. Schaf and C. E. Pereira, "Integrating mixed-reality experiments into virtual learning environments using intechangeable components," *IEEE Trans. Ind. Electron.*, vol. 56, no. 12, pp. 4768–4783, Dec. 2009.

[28] M. Helander and M. Emami, "Engineering eLaboratories: Integration of remote access and eCollaboration," *Int. J. Eng. Education*, vol. 24, no. 3, pp. 466–479, Mar. 2008.

[29] M. Casini, D. Prattichizzo, and A. Vicino, "Operating remote laboratories through a bootable device," *IEEE Trans. Ind. Electron.*, vol. 54, no. 6, pp. 3134–3140, Dec. 2007.

[30] J. Fayolle, C. Gravier, M. Ates, and J. Lardon, "Remote laboratories framework: Focus on reusability and security in m-learning situations," *Int. J. Online Eng.*, vol. 5, no. 3, pp. 19–24, Aug. 2009.

[31] M. García-Valls, I. Rodríguez-López, and L. Fernández-Villar, "iLAND: An enhanced middleware for real-time reconfiguration of service oriented distributed real-time systems," *IEEE Trans. Ind. Inform.*, vol. PP, no. 99, p. 1, 2012.

[32] M. García-Valls, P. Basanta-Val, and I. Estevez-Ayres, "Real-time reconfiguration in multimedia embedded systems," *IEEE Trans. Consumer Electron.*, vol. 57, no. 3, pp. 1280–1287, Mar. 2011.

[33] M. García-Valls, R. Castro-Fernández, I. Estévez-Ayres, P. Basanta-Val, and I. Rodríguez-López, "A time-bounded service composition algorithm for service based distributed real-time systems," in *Proc. IEEE Conf. Embedded Software Syst.*, Liverpool, U.K., Jun. 2012.

[34] M. García-Valls, A. Alonso, and J. A. de la Puente, "A dual-band priority assignment algorithm for QoS resource management," *Future Generation Comput. Syst.*, vol. 28, no. 6, pp. 902–912, Jun. 2012.

Marisol García Valls (SM'12) was born in Castellón de la Plana, Spain. She received the Computer Science Engineering degree from Universitat Jaume I de Castellón, Castellón de la Plana, Spain, in 1996 and the Ph.D. degree from the Technical University of Madrid, Madrid, Spain, in 2001.

She is currently a Professor with Universidad Carlos III de Madrid, Madrid, Spain, in the Telematics Engineering Department, where, since 2002, she has been the head of the Distributed Real-Time Systems Lab of the GAST Group. Her research interests focus on resource management and quality of service for distributed embedded real-time systems, real-time middleware, operating systems, and service oriented architectures. She is reviewer of a number of SCI/JCR journals in these fields; also, she is member of the PC of a number of conferences in these areas. She has been enrolled in a number of European, national, and regional projects as the EU projects ARTIST and ARTIST2, and the European Network of Excellence ARTISTDesign. Currently, she is the scientific and technical coordinator of the EU project iLAND (ARTEMIS-1-100026) and the Principal Investigator and coordinator of the Spanish national project REM4VSS (TIN-2011-28339).

Pablo Basanta Val was born in Lugo, Spain. He received the Electrical Engineering degree from University of Vigo, Vigo, Spain, in 2002, and the Ph.D. degree from Universidad Carlos III de Madrid, Madrid, Spain, in 2007.

Currently, he is an Assistant Professor with the Telematics Engineering Department, Universidad Carlos III de Madrid, Madrid, Spain, and he is a member of the Distributed Real-Time Systems Lab (Drequiem Lab) of the GAST group. He has been a member of national and European projects as REM4VSS. His main research interest is the real-time specification for Java.