PlanetLab@UOC: A Real Lab Over the Internet to Experiment With Distributed Systems

JOAN MANUEL MARQUÈS, DANIEL LAZARO, ANGEL A. JUAN, XAVIER VILAJOSANA, MARC DOMINGO, JOSEP JORBA

Computer Science, Multimedia and Telecommunication Studies, Universitat Oberta de Catalunya, Barcelona 08018, Spain

Received 17 March 2010; accepted 21 May 2010

ABSTRACT: This paper presents an innovative academic experience regarding the use of PlanetLab—an open large-scale platform over the Internet—in engineering courses on distributed systems. By integrating a live-deployment environment such as PlanetLab into distributed-systems courses, software implementations developed by students can interact with instructors' implementations in a real Internet-distributed scenario. To the best of our knowledge, this is the first experience regarding the use of the PlanetLab nodes available over Internet to teach distributed computing concepts to engineering students, since most of the existing literature on distributed-systems courses describes only learning experiences in lab-simulated or lab-emulated scenarios. As discussed in this paper, experimenting with a real laboratory over the Internet adds some complexity to students' practical activities, but it also facilitates a deeper and more intensive learning experience. A survey among students shows some clear benefits of this approach over the use of simulation labs, as well as some pitfalls that must be avoided whenever possible. The paper ends with a set of recommendations for instructors that might want to design similar courses as well as with an overview of future work to be performed in this field. © 2010 Wiley Periodicals, Inc. Comput Appl Eng Educ; Published online in Wiley InterScience (www.interscience.wiley.com); DOI 10.1002/cae.20468

Keywords: real laboratories; Internet; engineering education; distributed systems; PlanetLab

INTRODUCTION

Students from introductory courses on distributed systems usually learn the complexities and limitations of distributed systems as well as basic notions of how to deal with these complexities. Programming assignments provide them with some hands-on experience in a variety of distributed-systems technologies and helps them to better understand the algorithmic complexities. However, these programming assignments do rarely provide them with enough understanding of the complexities related with fully implementing and running a distributed application in a real environment over the Internet, which in our opinion should constitute an essential part of most computer science and engineering curricula.

In addition to dealing with intrinsic complexities of distributed algorithms, real implementations have to deal with hidden complexities related to issues such as concurrent access to local data, rare sequences of failures, etc. Traditionally, simulators and emulators have been used to deal with these complexities. Simulators or emulators may be used to run the assignments under different topologies and with different simulated network conditions but, as some authors point out, it is difficult to model or emulate the real conditions of a wide area network [1]. In addition, in the case of simulators, the application has to be adapted to the simulation framework, which is not a trivial task to perform. In this context, it seems interesting to explore the use of tools like PlanetLab (www.planet-lab.org), "an open platform for developing, deploying, and accessing planetary-scale services." As of March 2010, PlanetLab consisted of 1,073 nodes at 495 sites worldwide. Table 1 shows a comparison among simulation, emulation, and PlanetLab, summarizing the strong and weak points of each alternative [2]. Notice that the main advantage of PlanetLab-based laboratories over simulators and emulators is the fact that they offer both flexible and realistic environments to experiment with.

According to our experience, live-deployment environments such as PlanetLab offer new academic opportunities to deploy and test applications in real environments [3]. Applications are run on a set of geographically dispersed hosts connected to a real network. As in the case of emulators, applications do not need to be adapted.

In this paper we introduce PlanetLab@UOC, a platform that has been designed, implemented, and used at the Universitat Oberta de Catalunya (www.uoc.edu) to run students' assignments on distributed-systems courses by means of PlanetLab nodes.

Correspondence to J. M. Marquès (jmarquesp@uoc.edu).

Table 1 Simulation Versus Emulation Versus PlanetLab

Property	Fast network simulator	Emulated nodes and net	PlanetLab
System			
Scale	1,000s	~1,000	~500
Network topology and link characteristics	Flexible, latency only	Flexible, all effects	Hard-wired, all effects
Node effects	No	Yes	Yes
Stimulus			
Workload	Flexible	Flexible	Flexible and realistic
Operator actions	No	Flexible	Realistic
Faults	Net only	Flexible	Realistic
Measures			
Reproducibility	High	Medium	Low
Experiment management	Easy	Medium	Hard

Therefore, the paper presents our academic experiences as well as some relevant lessons learned from them. PlanetLab@UOC has been developed by the DPCS research group (dpcs.uoc.edu) and has been used for two semesters in an undergraduate course on distributed systems with around 100 registered students per semester. As illustrated in Figure 1, first these students implement a distributed-systems algorithm in a given distributed application developed by the course instructors. Then, students must run their instance of the application such in a way that it interacts with other instances of the application implemented by the instructor and deployed in PlanetLab. During this execution, several tests are carried out to verify the correct behavior of the student's application. Finally, results of each test are stored in a university server so that they are accessible to both students and instructors throughout a web application.

This paper is structured as follows: Second section motivates the convenience of developing learning systems such as PlanetLab@UOC and reviews some related works. Third section describes the academic environment where this system has been employed as well as the system architecture. Fourth section provides specific details regarding the use of PlanetLab@UOC in our courses. Fifth section presents and discusses the results of a survey among students of these courses. Sixth section introduce some future work we plan to develop to follow up the one described here. Finally, seventh section highlights some of the most important aspects of our system and of the related teaching experiences.

MOTIVATION AND RELATED WORK

As with many other experimental disciplines, courses on distributed systems require laboratory environments where students can apply the concepts and empirically verify theoretical results provided during lectures [4,5]. Since it is not trivial to design and implement a distributed application—in fact, it can be a highly complex, error-prone, and time-consuming task—it becomes necessary to provide lab facilities that support the teaching process and promote the active participation of students in their own learning process [1,6]. Typically, experimental testbeds in distributed-systems courses make use of one of the following approaches: (a) emulation of networking infrastructure, (b) network simulators, or (c) real live-deployment environments. Each approach supports different abstractions of distributed middleware levels.

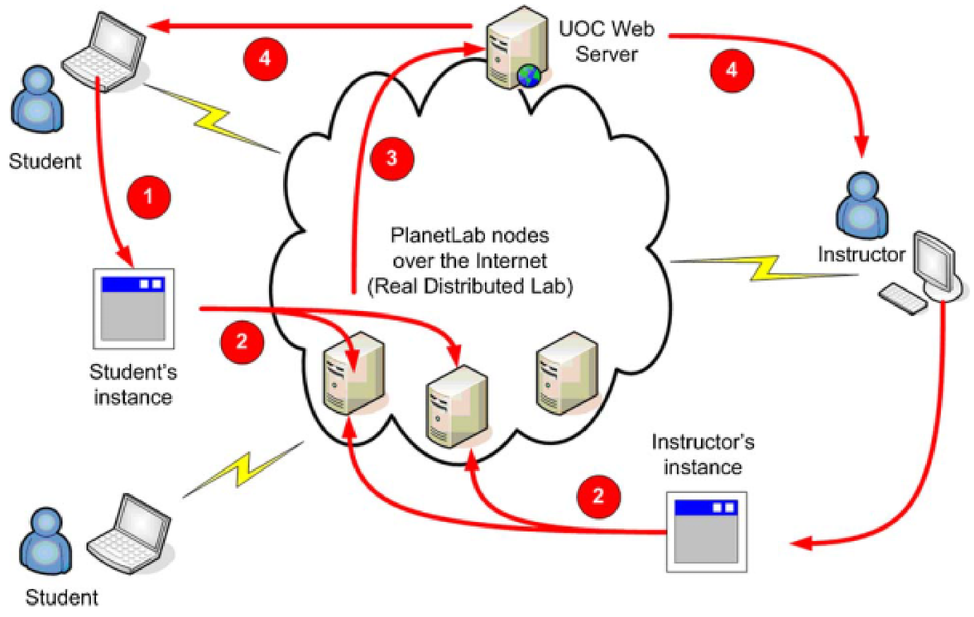


Figure 1 Using PlanetLab as a real laboratory for distributed-systems courses. [Color figure can be viewed in the online issue, which is available at www.interscience.wiley.com.]

Some of the studied experimental facilities integrate one or more of these approaches by combining parameters with different levels of control and realism. In environments such as Emulab [7], ModelNet [8], or Netbed [7], the facility may even combine emulated components, simulated components, and real components (nodes and links) forming a hybrid environment. However, it is difficult to obtain a well-balanced testbed system, suitable for learning, by just combining different approaches since each environment tends to focus on only one or two of these approaches:

- *Network emulators* provide abstraction of network devices, so there is no need to deal with specific hardware. However, emulators tend to focus on packet-level discrete mechanisms and, therefore, they do not offer the programmer a real distributed-middleware experience.
- *Network simulators*—for example, NS2 [nsnam.isi.edu/nsnam], OPNET [www.opnet.com], or OMNeT++ [www.omnetpp.org]—allow to test network performance under different parameter values (latency, intercommunication protocols, service policies, etc.). Furthermore, they provide an environment that can be easily controlled and reproduced if necessary, so that different scenarios can be analyzed. They are used sometimes in a metaphoric gameplay of distributed application stability, like the PDConsole environment [9]. However, network simulators are difficult to configure so that they could offer a realistic behavior of individual components in large-scale distributed systems.
- *Real-life experimental platforms* allow students to deal with intrinsic problems of distributed systems, for example: application deployment, resource discovery, and resource management. Many of these computer testbeds are driven by different communities like PlanetLab (which is a dedicated platform) [10], Seattle [https://seattle.cs.washington.edu] (which is a non-dedicated environment), SatelliteLab [www.mpi-sws.org] (which follows an approach similar to that of Seattle but adding mobile devices), or StarHPC [11] (which uses the Cloud as a background platform). In these communities people contribute resources when they do not need them and thus can access community resources when needed [12]. These communities can use either physical machines or virtual machines. They keep track of the available resources and also the number of resources that have been used by each member. In some cases, they even provide means to deploy services in widely used grid or cloud infrastructures, such as Amazon EC2 in the case of StarHPC.

Aside from general-purpose remote job execution tools, such as Cfengine [13], Gexec [www.theether.org/gexec], or Vxargs [vxargs.sourceforge.net], there are some projects that focus on managing distributed applications as well as on their automatic assessment in realistic environments:

- Inside the PlanetLab platform, the PlanetLab Application Manager (PAM) [appmanager.berkeley.intel-research.net] provides many features for long-running services on PlanetLab. It is designed to maintain applications that provide a service requiring a high availability level. However, PAM does not provide a way to interactively manage life-cycle of applications on remote machines and does not offer support tools for short-live applications. In addition, it does not offer support for synchronization of distributed applications.
- SmartFrog [14] also manages distributed applications. It is a framework for describing, deploying, and controlling software components. It consists of a description language and a collection of agents that manage applications deployed in a distributed manner. Like PAM, SmartFrog is not a tool that can be used to interactively to control distributed applications.
- Plush [plush.cs.williams.edu] offers functionalities to manage distributed applications, their lifecycle, their deployment and their configuration. Besides, it offers functionalities to enable synchronization of distributed applications by means of barriers. While some of the Plush functionalities are similar to the ones provided by PlanetLab@UOC the scope and the way they are used significantly differ from our approach. In particular, PlanetLab@UOC hides complexities derived from distributed applications such as synchronization and group communication.
- Splay [15] includes tools for simplifying the design of large-scale distributed applications and overlay networks. Applications can be described in a specialized scripting language and, after that, they can be deployed into different platforms running in a safe environment with restricted access to local resources.

From the literature review, it can be pointed out that PlanetLab@UOC addresses different objectives than most of the existing approaches. PlanetLab@UOC is purposely oriented to learning environments, for example, it focuses more on the learning-oriented functionalities than on the middleware-oriented functionalities provided by other proposals. Assessment capabilities as well as distributed execution with instantaneous feedback are singular points of PlanetLab@UOC. Besides, the nature of the system enables students to execute their assignments on their local machines and, at the same time, interacting with instructors' implementations deployed in PlanetLab. As it will be discussed later in this paper, this enables rich scenarios to learn and apply concepts related with distributed systems.

ASSIGNMENTS ENVIRONMENT

The Universitat Oberta de Cataluya (UOC) is a purely online university that has offered undergraduate and graduate degrees since 1995. UOC uses a proprietary e-learning platform called Virtual Campus, which has been developed and improved over the last decade to fulfill online students' and instructors' needs. UOC basically uses an asynchronous learning model, which follows a student-centered educational paradigm: at the beginning of the semester, students access the course's online classroom and, following the recommendations provided by their instructors, they download a complete syllabus of the course along with all associated learning materials and resources. Throughout the course, students are encouraged to participate actively in discussion forums, to develop collaborative learning projects and, specially, to follow a scheduled continuous assessment process, which typically consists of four or five homework activities. Currently, UOC has more than 37,000 enrolled students and more than 400 tutors and lecturers.

In this scenario, about 100 students per semester complete our course on distributed systems. Students' assignments are presented as a distributed application that uses a set of well-known distributed-systems algorithms. Students have to extend the provided template of the application implementing the required distributed algorithms or mechanisms, that is, all or some of the distributed algorithms or mechanisms that the application uses. The application template uses a distributed framework to start the execution of an experiment. Then, it gets the necessary information to interact with other application instances deployed in PlanetLab (Fig. 1). Thus, for example, students might be required to implement a mechanism for consensus without worrying about group membership, which is provided in the application template through the framework. While implementing the solution, students

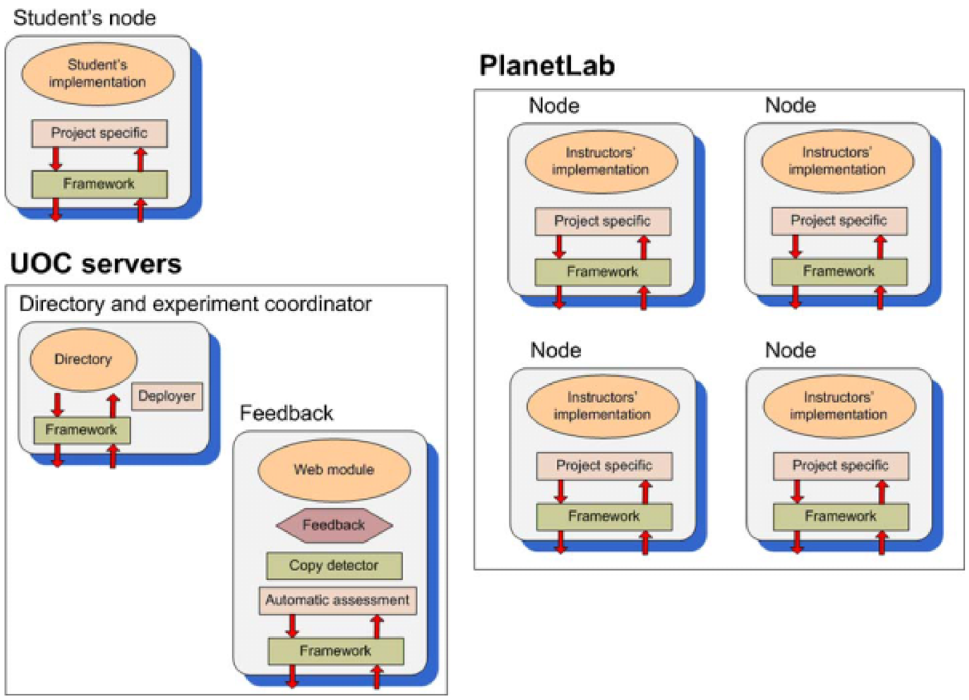


Figure 2 PlanetLab@UOC architecture. [Color figure can be viewed in the online issue, which is available at www.interscience.wiley.com.]

can test the application using a local execution mode provided by instructors. Once implemented, their solution is executed and it interacts with other instances of the same application that have been developed by instructors and that are currently deployed in PlanetLab. At this moment, several tests are performed. The results of these tests—which have been provided by a large subset of PlanetLab nodes—are collected and compared among them. Students and instructors can consult registered information on performed runs to know whether the tests were correctly executed or, alternatively, to get more information about possible errors that might have occurred. Assignments are divided into a number of different phases, for example, a simple assignment could have two phases: activity simulation and results collection.

System Architecture

Figure 2 shows the distributed architecture of PlanetLab@UOC. This architecture is divided into three different environments: (a) student's computer, (b) university servers, and (c) PlanetLab nodes. The student's computer executes the student's implementation of the assignment—extending the template provided by the instructors, which uses the framework functionalities. PlanetLab nodes execute the instructors' implementation, which is based on the same template previously provided to students at the beginning of the course. Finally, university servers are in charge of controlling the live deployment, the execution and the evaluation processes. University servers also host a web application that manages the historical information regarding all executions performed by each student.

Student's Computer. The components that are executed in the student's computer are described next:

- *Student's implementation of the assignment's algorithms*: This is done by the student by completing the template code.
- *Project-specific implementation*: Implementation of the functionalities which are specific of the application. This implementation is provided to the student by the instructors. It includes parts of the application that go beyond the assignment goals and, therefore, do not need to be implemented by students. It also includes the code that is needed to execute the tests and validate the assignment.
- *Framework*: Generic layer of communication and coordination which is common to all components. This layer is responsible for starting and coordinating the execution of an experiment and also of getting the necessary information, such as membership, to interact with other application instances deployed in PlanetLab. It also collects the final results and sends them to the *feedback* service hosted in *university* servers.

University Servers. As regards as the university servers, they host two different services:

- *Directory and experiment's coordination service*: This service is in charge of keeping the required number of deployed application instances. Students' instances contact it to start the execution of an experiment. Furthermore, it has the following components:

1. *Deployer*: It is responsible for deploying an application in the PlanetLab nodes. Its tasks include: (a) selecting active nodes in PlanetLab that might be used for deploying the application, (b) installing the application on the selected nodes and starting its execution, and (c) monitoring the selected nodes to detect failures and disconnections.
2. *Directory*: It is in charge of the activation and coordination of experiments. Its tasks include: (a) selecting a group of nodes to participate in an experiment, (b) sending group and membership information to the nodes involved in the experiment, and (c) coordinating the execution of the experiment, that is, its different phases and tests.

3. *Framework*: This is a generic layer of communication and coordination. It is the same as the one included in the student's component.

- *Feedback*: This service collects, evaluates, and stores the results of each execution. Students and instructors can consult it. It has the following components:

4. *Web module*: It is a web application that students and instructors may use to check the results of each execution. Instructors can track results of individual students, as well as aggregate information. Students can check the historical of their executions.
5. *Feedback*: It stores results associated with each execution.
6. *Copy detector*: It stores the classes used in each student's execution. When the assignment deadline has expired, students' classes are compared to detect possible plagiarism.
7. *Automatic assessment*: It contains the logic to evaluate the results of experiments for a specific application.
8. *Framework*: It is a generic layer of communication and coordination. It is the same as the one included in the student's component.

PlanetLab Nodes. Nodes in PlanetLab execute the same components that are present in students' nodes. The difference is that they execute an instructor's implementation of the assignment. This way, the compatibility of the student's implementation with the proposed protocol can be tested, and its results can be compared with the results obtained by the instructor's implementation.

Overall PlanetLab@UOC. Components of PlanetLab@UOC are reusable. The only parts that change are those related with the application and algorithms of each assignment, that is, *project-specific*, *student's implementation*, and *instructors' implementation*. On the one hand, big portions of the *Project-specific* component are common to different assignments although they have to be adapted to the specific characteristics of each one. On the other hand, *students' implementations* and *instructors' implementation* are completely dependent of each algorithm and must be widely re-implemented for each assignment.

Student's Flow of Assignment Implementation and Execution

Figure 3 shows a typical student's flow when developing and testing his/her implementation. Two modes of execution, local and remote, are provided to students:

- *Local mode of execution*: All instances involved in the experiment run in a single node. This node may be the student's computer, where he has full control over resources and configuration, or a PlanetLab node.
- *Remote mode of execution*: Students execute their implementation interacting with remote instructors' instances deployed in PlanetLab. Students' implementation may be executed in their own computer or in a PlanetLab node. The latter is used for the final assessment of the assignment.

Automatic Assessment and Feedback

One of the most noticeable functionalities of PlanetLab@UOC is its ability to perform automatic assessment of students' assignments and also to provide feedback reports related with these assessments to both students and instructors.

To this end, the *project-specific* component—which is available both at the student's computer and at PlanetLab nodes—implements a logic in charge of executing one or more tests that check the correct behavior of the implemented algorithm. Once the execution ends, the student's component—or, alternatively, the node that executes the student's implementation—and the PlanetLab nodes participating in the experiment—either all or a large subset of them—send the results of each test to the *automatic assessment* module in the university servers. This module compares the results given by each test and generates an assessment report that, for each test, indicates if it has been passed or not. In case a test has not passed, the report provides descriptive information about the reason why it has failed. This information is sent back to the student as an output of the executed experiment and is also stored at the university server to be accessed, if necessary, by the student or his/her instructors in a future time using the web application included in PlanetLab@UOC (Fig. 1). Therefore, each student has online access to the results of his/her own experiments and, additionally, instructors can also access the results associated with each student and experiment. Instructors also receive summarized and aggregated reports that help them to know how students perform each practical assignment, for example: a list of best executions for each student (Fig. 4) or statistical graphs including percentages of students that have tried and succeed in each test. This kind of information is especially interesting in the case of distance universities like ours since students and instructors never meet face-to-face [16].

Finally, once the assignment deadline has expired student's executions stored at the university server are compared among

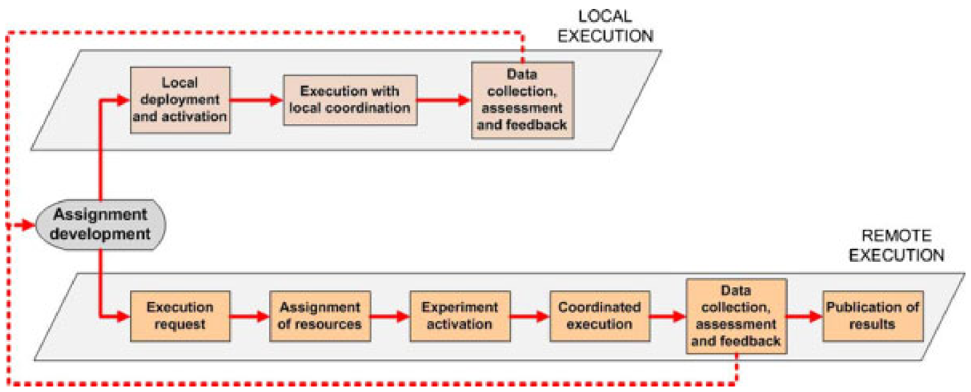
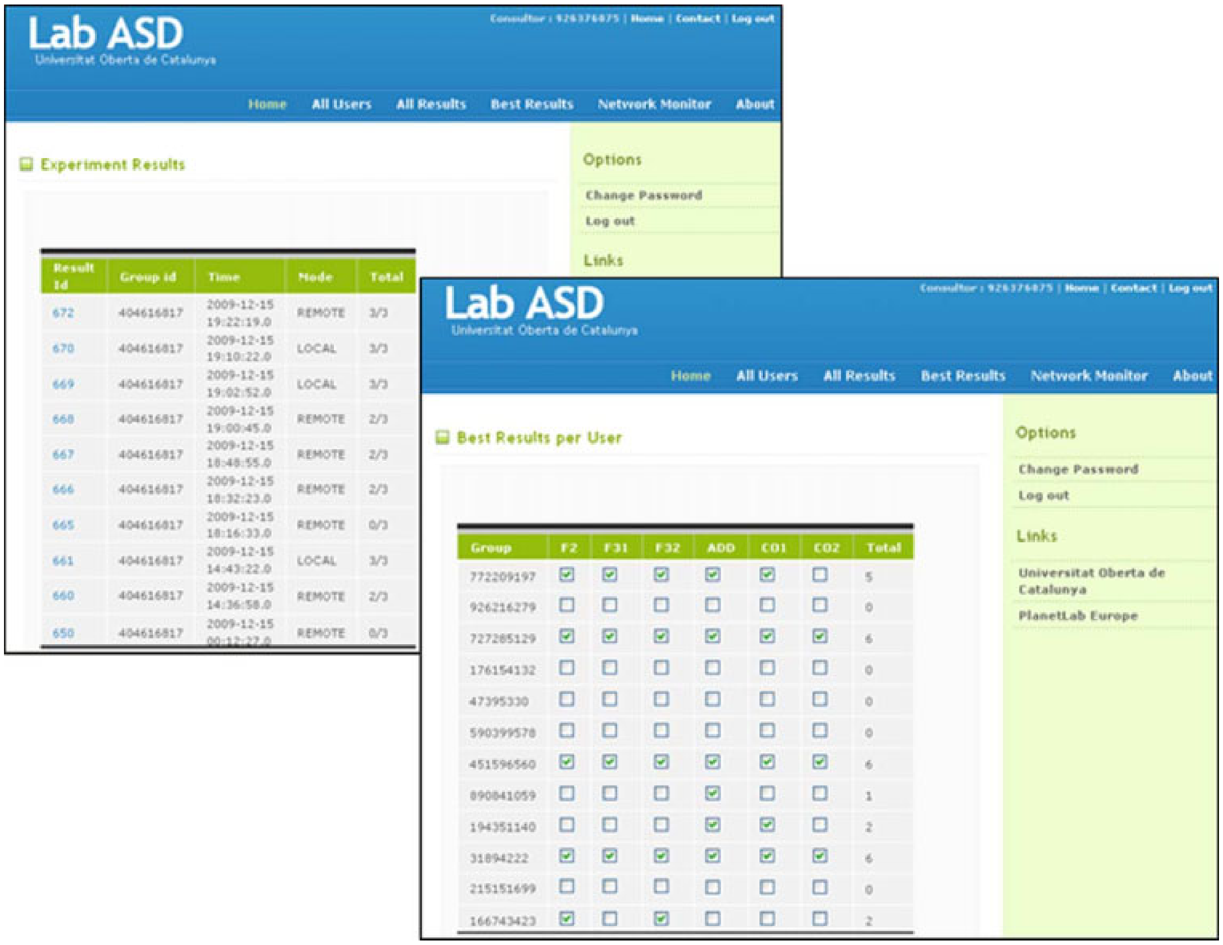


Figure 3 Student's flow of assignment implementation and execution. [Color figure can be viewed in the online issue, which is available at www.interscience.wiley.com.]



Experiment Results

Result Id	Group id	Time	Mode	Total
672	404616817	2009-12-15 19:22:19.0	REMOTE	3/3
670	404616817	2009-12-15 19:10:22.0	LOCAL	3/3
669	404616817	2009-12-15 19:02:52.0	LOCAL	3/3
668	404616817	2009-12-15 19:00:45.0	REMOTE	2/3
667	404616817	2009-12-15 18:48:55.0	REMOTE	2/3
666	404616817	2009-12-15 18:32:23.0	REMOTE	2/3
665	404616817	2009-12-15 18:16:33.0	REMOTE	0/3
661	404616817	2009-12-15 14:43:22.0	LOCAL	3/3
660	404616817	2009-12-15 14:36:58.0	REMOTE	2/3
650	404616817	2009-12-15 00:12:27.0	REMOTE	0/3

Best Results per User

Group	F2	F31	F32	ADD	C01	C02	Total
772209197	☑	☑	☑	☑	☑	☐	5
926216279	☐	☐	☐	☐	☐	☐	0
727285129	☑	☑	☑	☑	☑	☑	6
176154132	☐	☐	☐	☐	☐	☐	0
47395330	☐	☐	☐	☐	☐	☐	0
590399578	☐	☐	☐	☐	☐	☐	0
451596560	☑	☑	☑	☑	☑	☑	6
890841059	☐	☐	☐	☑	☐	☐	1
194351140	☐	☐	☐	☑	☑	☐	2
31894222	☑	☑	☑	☑	☑	☑	6
215151699	☐	☐	☐	☐	☐	☐	0
166743423	☑	☐	☑	☐	☐	☐	2

Figure 4 All executions (left) and best execution for each student (right). [Color figure can be viewed in the online issue, which is available at www.interscience.wiley.com.]

them to detect possible plagiarism, which is recognized as an important issue in this kind of assignments [17,18]. To this end, we use JPlag (www.ipd.uni-karlsruhe.de/jplag) which, according to their authors: (a) "is a system that finds similarities among multiple sets of source code files," (b) "does not merely compare bytes of text, but is aware of programming language syntax and program structure and hence is robust against many kinds of attempts to disguise similarities between plagiarized files," and (c) "is typically used to detect and thus discourage the unallowed copying of student exercise programs in programming education."

EXPERIENCES AT THE UOC

The described assignment environment has been used in our course on Distributed Systems Architecture at UOC. Table 2 contains a list of competences to be acquired by students of the course which, on the average, has about 100 registered students per semester.

These practical assignments, which last for about 2 months, are carried out by groups, each group composed by two students. The assignments are divided in three phases of incremental difficulty. Each of these phases is associated with a set of competences. By correctly completing each phase, students receive accumula-

Table 2 Competences to be Acquired During Practical Assignments

No.	Keyword	Competence description
1	Design	To know the fundamental issues and challenges associated with the design of distributed systems
2	Development	To possess basic notions on development of distributed systems
3	Algorithms	To be able to understand simple distributed algorithms and to deal with algorithmic challenges in the context of a distributed application
4	Remote invocation	To be able to use remote invocation technologies (RMI, SOAP, etc.)
5	Web applications	To know the basics of building web applications (CGI, servlets, etc.)

Table 3 Assignments

Semester	Phase 1: simple distributed application	Phase 2: simple distributed algorithm	Phase 3: more complex distributed algorithm	Live deployment
Autumn 2007	Distributed record of entries in a government office	Paxos algorithm [19]	Bully algorithm for leader election [20]	No
Spring 2008	Distributed record of clients in a corporation	Primary backup with confirmation in two phases [21]	LCR algorithm for leader election [22]	No
Autumn 2008	Distributed record of orders in a warehouse	Primary-backup with confirmation in three phases [23]	Afek and Gafni's algorithm for leader election [24]	No
Spring 2009	Distributed file sharing	Golding's time-stamped anti-entropy (with no purge of logs) [25]	Add the delete operation and the purge of logs	Yes (PlanetLab)
Autumn 2009	Distributed file sharing	Improvement of Golding's Time-Stamped Anti-Entropy protocol to improve dissemination of information for commutative operations [26]	Casual ordering between non-commutative operations (a deletion must be executed after all previous operations from the same node)	Yes (PlanetLab)

tive marks that will eventually determine their final grades. This division of the practical assignment into three different phases allows instructors to evaluate the level of achievement of each group in each competence. It also helps students to better organize their work and focus more on those areas which are of interest for them.

We have used the same structure for the assignments during five semesters. From autumn 2007 to autumn 2008 assignments were executed locally. During spring and autumn 2009 semesters PlanetLab was used. Table 3 presents the details for each semester.

Some details about each of these phases and how they contribute to students' final marks are given next:

- *Phase 1*: It consists on implementing a simple web interface and a servlet to access the application's functionalities. It is associated with the competence of being able to build simple web applications. Its completion is not enough by itself to pass the assignment, but it gives students the right to complete a final exam where they could pass the course if they prove to possess enough knowledge on the subject. Thus, phase 1 is a simple but necessary task for all students to perform.
- *Phase 2*: It consists on the implementation of a well-known distributed algorithm. The pseudo-code of the algorithm—as well as a detailed explanation and some basic references—are provided to students by instructors. The completion of this phase is associated with competences 1–4 in Table 2. By correctly completing this phase, students can pass the practical assignment and, in fact, those students who complete both phase 1 and phase 2 are eligible to obtain a B grade.
- *Phase 3*: It consists on either implementing another distributed algorithm or extending/improving the algorithm in phase 2—so that the applications' functionalities will be increased or enhanced. In this phase instructors do not give as much details to students as and in the previous phases and, therefore, students need to resort to scientific papers where the original algorithm has been described in detail. This, in turn, allows students to get a deeper insight into competences 1–3 (Table 2). Completing the three phases is a necessary condition for students to attain the maximum possible grade.

The live deployment structure (used during the 2009 spring and autumn semesters) uses 36 nodes from PlanetLab to host the instructors' implementation. Whenever a student starts an experiment, 10 of these nodes are selected at random and form a group that communicates using the mechanisms of the assignment application—the TSAE protocol in our case. Then, experiments run the activity phase, for example, during the 2009 spring, this phase consisted in a 10-min simulation activity where the nodes were generating random events and simulating random failures and recoveries; after that, a stabilization 5-min period—where no new events were generated—was executed. After this activity phase, the information related with the student's node and some of the PlanetLab nodes was collected and compared to check for correctness of the student's implementation. Results associated with each assessment were then stored at the university servers, and from that moment on they were made available online to both students and instructors.

STUDENTS' SURVEY

In order to evaluate the students' experience and opinions regarding this course, at the end of the semester we perform a survey study. Table 4 shows the questionnaire we utilized during the semesters of autumn 2008 and spring 2009. Notice that there is a significant difference between these two semesters: while no live-deployment environment was used during autumn 2008, during spring 2009 we introduced the use of PlanetLab@UOC. Therefore, significant methodological and technological changes took place between both semesters. Of course, we were especially interested in analyzing to which extend—if any—these changes modified students' perception of our course.

The questionnaire was sent to each student the same day he/she received his/her final marks. In autumn 2008, 29 out of 89 students who had completed the assignment answered the questionnaire (33% of participation). The total number of registered students this semester was 107. In spring 2009, 28 out of 79 students who had completed the assignment answered the questionnaire (35% of participation). The total number of registered students this semester was 100. It is important to highlight that the questionnaire was not anonymous, since students had to send an e-mail to the course coordinator with their answers.

From the survey results, the following conclusions can be made:

- The overall difficulty of the practical assignment is considered as being high, both by students of autumn 2008 (average value of 2.4 in a 1–3 scale) and by students of spring 2009 (average value of 2.5). Notice that the average score in both semesters is practically the same.
- Figure 5 shows the main difficulties that students found when completing their practical assignments. The most cited prob-

Table 4 The Questionnaire

No.	Question	Autumn 2008	Spring 2009	*P*-value
1	Rate from 1 (lowest) to 3 (highest) the overall difficulty of implementing a distributed algorithm in a realistic distributed environment	2.4	2.5	—
2	Which are the main complexities of the assignment? (more than one option is allowed)			
	Understanding the problem	41%	36%	—
	Understanding the algorithms	52%	46%	—
	Implementing the algorithms	41%	39%	—
	Developing an application that works as intended	59%	57%	—
	Interacting with the template provided by the instructors	28%	32%	—
	Interacting with instructors instances deployed in PlanetLab	N/A	75%	—
	Other	0%	39%	—
3	Splitting the activity in several phases helps me to manage the different tasks	89%	81%	—
4	Splitting the activity in several phases allows me to organize and regulate my effort	75%	75%	—
5	In case you have not fully completed all the phases, which was the reason? (more than one option is allowed)			
	Not interested in the course	0%	0%	—
	Lack of time due to personal or professional issues	45%	39%	—
	Excessive difficulty of some phases	45%	14%	—
	Excessive time requirements	21%	39%	—
	Interested in distributed systems but not in implementation	3%	4%	—
	Other	0%	29%	—
6	This practical assignment has been . . . (more than one option is allowed)			
	Useful to understand important issues on distributed systems	66%	89%	—
	Useful to acquire practical knowledge on distributed systems	62%	61%	—
	Helpful to understand how to implement real distributed systems	55%	50%	—
	Other	0%	14%	—
7	Is the assignment well related with the course goals? (the percentage of positive answers is shown on the right)	83%	75%	—
8	The statement of the assignment was . . .			
	Clear and well detailed	21%	29%	—
	Easy to follow but not too much detailed	55%	39%	—
	Not clear and poorly detailed	24%	29%	—
9	About the statement of the assignment . . .			
	I appreciated that it was closed and the algorithms to be used were given beforehand by instructors	66%	61%	—
	It was too much closed and I would have preferred a design phase to decide which algorithms to use	10%	11%	—
10	Difficulty of phase 1 (1 = lowest; 3 = highest)	1.7	1.8	—
11	Difficulty of phase 2 (1 = lowest; 3 = highest)	2.4	2.6	—
12	Difficulty of phase 3 (1 = lowest; 3 = highest)	2.7	2.5	—
13	Do you prefer to execute your application in a realistic environment instead of using your PC? (the percentage of positive answers is shown on the right)	52%	70%	0.155
14	Rate from 0 (completely unsatisfied) to 10 (completely satisfied) your overall satisfaction with the practical assignment	6.7	5.6	0.180

lem in the case of spring 2009 students was "interacting with instructors instances deployed in PlanetLab" (cited by 75% of participating students). Attention should also be paid to other difficulties like "developing an application that works as intended" (59% in autumn 2008 and 57% in spring 2009), and "understanding the algorithms" (52% in autumn 2008 and 46% in spring 2009). About the first difficulty, we think that it could be due to the following facts: (a) by having to interact with instructors' implementation, students are forced to follow a given interface and adapt their codes to it; and (b) when tests did not work properly, it was really difficult to know which part of the code was generating the problem. This lack of feedback when interacting with instructors' instances was one of the mains students' complain during spring 2009. On the contrary, in autumn 2008 students ran locally their experiments, interacting with other instances of their implementation and, thus, executions were easier to track.

- Students from both semesters also agree in that by splitting the activity in several phases help them to better manage their tasks (89% of participants in autumn 2008 and 81% in spring 2009) and to better regulate their effort (75% in autumn 2008 and also 75% in spring 2009).
- With regard to the reasons why some students could not complete all phases of the project (Fig. 6), it should be noted that the "lack of time due to personal or professional issues" was the most cited (by 45% of participant students in autumn 2008 and by 39% of participants in spring 2009). Regarding the "lack of time due to personal or professional issues," it should be noticed that 95% of UOC combine their studies with their professional activity. Moreover, about 75% of our students are between 25 and 45 years old and, therefore, most of them have to attend familiar issues while studying. Other causes highly cited were referred to an "excessive difficulty of some phases" (45% in autumn 2008 but only 14% in spring 2009) and also "excessive time requirements" (21% in autumn 2008 and 39% in spring 2009). About the excessive difficulty of some phases, we have detected in some students a lack of programming capacities, which is a requisite of this course.

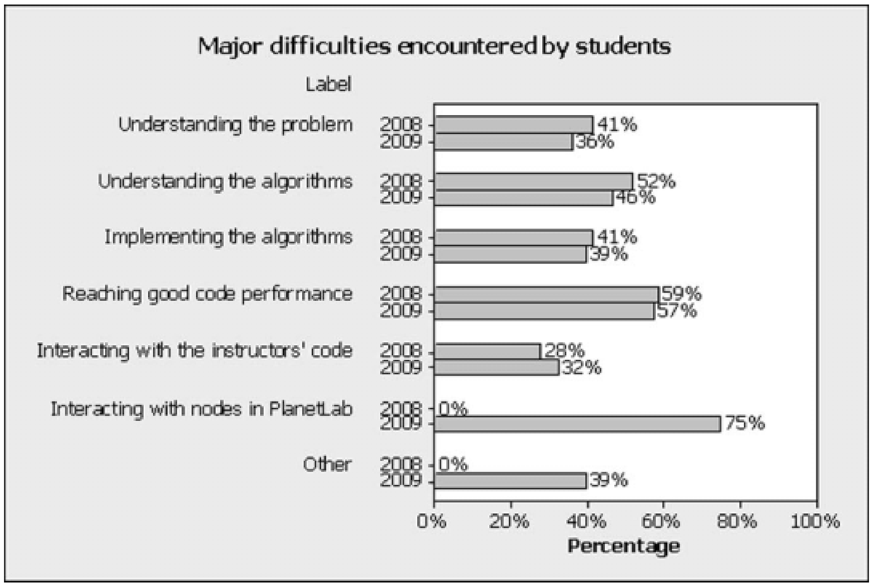


Figure 5 Main difficulties encountered by students. (More than one option is allowed.)

- Notice also that most students from both semesters agree in that the practical assignment was useful to understand important issues on distributed systems (66% in autumn 2008 and 89% in spring 2009), to acquire practical knowledge on the subject and also to implement real distributed systems.
- According to students' opinions, it is also possible to state that the practical assignment is well related with the course goals. However, it seems that improvements should be made in the statement of the assignment, since an important percentage of students do not consider it to be enough clear and well detailed. On the contrary, about 60–65% of students appreciated the fact that the statement was closed and that the algorithms to be used were already provided by instructors.
- Regarding the difficulty of each phase, it seems obvious that both phase 2 (average difficulty about 2.5 over 3) and phase 3 (average difficulty about 2.6 over 3) are much more difficult than phase 1—which can not be considered as easy since it receives an average difficulty level of 1.75 over 3. Notice that average rates are approximately equal across semesters.
- It is interesting to notice that only about 52% of autumn 2008 students preferred to execute their application in a realistic environment, while this percentage increased up to 70% in the case of spring 2009 students. While this difference cannot be considered as statistically significant ($P = 0.155$), it seems that there is a tendency among those students who have already used realistic environments to prefer them over non-realistic ones.
- Finally, the overall satisfaction of students with the practical assignment is neither too low nor too high. The average score given by autumn 2008 participants was of 6.7 while this

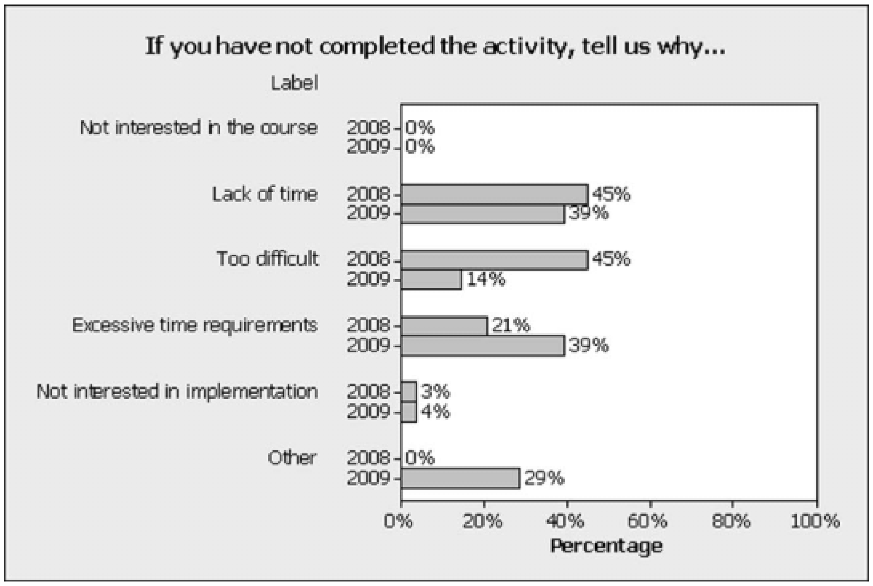


Figure 6 Reasons for not completing the assignments. (More than one option is allowed.)

score decreased to 5.6 in spring 2009. Even when the difference between these two scores is not statistically significantly ($P = 0.180$), it clearly shows that some methodological innovations should be addressed in this course to increase the overall satisfaction of its students.

FUTURE WORK

We are currently moving towards a deeper integration of PlanetLab in our courses. In particular, we are designing new practices where students will have to measure their algorithms' performance in different working conditions of the distributed system, for example: different locations of nodes, random noise, random connections and disconnections, etc. Another line we are working on is the development of new modules for the PlanetLab@UOC environment, for example: a new module that allows users to select those PlanetLab nodes which satisfy certain properties, a new module that takes care of the random generation of nodes failures, a new module that provides automatic assessment of students' codes and also that offers a more personalized feedback to each student, etc.

CONCLUSIONS

This paper presents an innovative academic experience regarding the use of a real distributed laboratory over the Internet based on the PlanetLab open platform. In this experience, students execute their implementations using PlanetLab@UOC, an environment that allows their implementations to interact with instructors' instances also deployed over the Internet by using the PlanetLab nodes. Moreover, PlanetLab@UOC makes it possible for students and instructors to deploy, execute, and assess students' assignments. As discussed in this paper, our environment also provides automatically generated feedback reports to students regarding the performance of their instances.

Based on the results obtained from a two-semester survey among students, the major difficulties found by students when working in such a real distributed environment are: (a) the complexity of the interaction with instructors' implementations deployed in PlanetLab, which are mainly due to the lack of debugging feedback; (b) the excessive time requirements to complete most practices; (c) the difficulty to fully understand and implement the proposed distributed algorithms; and (d) the challenge of developing an application that reaches an efficient overall performance.

The two main contributions of the PlanetLab@UOC environment are automatic assessment and, especially, the use of a real laboratory over the Internet. Both students and instructors agree in their academic benefits: students appreciate having a "real-time" automatic-assessment feedback while instructors agree in that the environment save them a lot of time, since they are not required to execute and test each student's implementation. Moreover, by being able to access a web application which contains reports on all students' executions, instructors can keep track of how students are working and progressing in their learning process. Finally, students also seem to prefer the use of a real-live distributed environment over simulated or emulated labs, since this helps them to better understand distributed-systems complexities and also to keep their motivation up during their learning process.

ACKNOWLEDGMENTS

This research work has been partially funded by the Catalan Government under grant 2009MQD00121 and by the UOC Vice-Chancellor of Innovation under grant IN-PID0904. This work has also been supported by the HAROSA Knowledge Community (dpcs.uoc.edu).

REFERENCES

[1] I. Calvo, M. Marcos, D. Orive, and I. Sarachaga, Building complex remote learning laboratories, Comput Appl Eng Educ 18 (2010), 53–66.

[2] D. Oppenheimer, J. Albrecht, D. Patterson, and A. Vahdat, Distributed resource discovery on PlanetLab with SWORD, First Workshop on Real, Large Distributed Systems (WORLDS '04), 2004.

[3] J. Albrecht, C. Tuttle, A. C. Snoeren, and A. Vahdat, PlanetLab application management using plush, SIGOPS Oper Syst Rev 40 (2006), 33–40.

[4] B. Balamuralithara and P. C. Woods, Virtual laboratories in engineering education: The simulation lab and remote lab, Comput Appl Eng Educ 17 (2008), 108–118.

[5] T. Achalakul, B. Sirinaovakul, and N. Nuttaworakul, Virtual laboratory: A distributed collaborative environment, Comput Appl Eng Educ 12 (2004), 44–53.

[6] A. Kayssi, S. Sharafeddine, and H. Karaki, Computer-based laboratory for data communications and computer networking, Comput Appl Eng Educ 12 (2004), 84–97.

[7] B. White, J. Lepreau, L. Stoller, R. Ricci, S. Guruprasad, M. Newbold, M. Hibler, C. Barb, and A. Joglekar, An integrated experimental environment for distributed systems and networks, Proceedings of the 5th Symposium on Operating Systems Design and Impl. (OSDI), Boston, MA, 2002, pp 255–270.

[8] A. Vahdat, K. Yocum, P. Mahadevan, D. Kostic, J. Chase, and D. Becker, Scalability and accuracy in a large-scale network emulator, Proceedings of the 5th Symposium on Operating Systems Design and Implementation, 2002, pp 271–284.

[9] J. Wein, K. Kourtchikov, Y. Cheng, R. Gutierez, R. Khmelichek, M. Topol, and C. Sherman, Virtualized games for teaching about distributed systems, ACM SIGCSE Bull Arch 41 (2009), 246–250.

[10] A. Baviera, M. Bowman, B. Chun, D. Culler, S. Karlin, S. Muir, L. Peterson, T. Roscoe, T. Spalink, and M. Wawrzoniak, Operating system support for planetary-scale network services. Proceedings of the 1st conference on Symposium on Networked Systems Design and Implementation, 2004, p 19.

[11] C. Ivica, J. Ridley, and C. Shubert, Teaching Parallel Programming within Elastic Compute Cloud, Proceedings of the ITI 2009 31st Int. Conf. on Information Technology Interfaces, 2009, pp 353–356.

[12] X. Vilajosana, D. Lazaro, J. Marques, and A. Juan, DyMRA: A decentralized resource allocation framework for collaborative learning environments, Intelligent Collaborative e-Learning Systems and Applications, 2009, pp 147–169 (Springer Series in Studies in Computational Intelligence, Vol. 246) Springer-Verlag, Berlin, Germany.

[13] M. Burgess, Cfengine: A site configuration engine, USENIX Comput Syst 8 (1995), 309–337.

[14] P. Goldsack, J. Guijarro, A. Lain, G. Mecheneau, P. Murray, and P. Toft, SmartFrog: Configuration and automatic ignition of distributed applications, 10th OpenView University Association Workshop, 2003.

[15] L. Leonini, E. Rivière, and P. Felber, SPLAY: Distributed systems evaluation made simple (or how to turn ideas into live systems in a breeze), Proceedings of the 6th USENIX Symposium on Networked Systems Design and Implementation Table of Contents, 2009, pp 185–198.

[16] A. Juan, T. Daradoumis, J. Faulin, and F. Xhafa, A data analysis model based on control charts to monitor online learning processes, Int J Bus Intell Data Mining 4 (2009), 159–174.

[17] S. Rodriguez, J. Pedraza, A. Dopico, F. Rosales, and R. Mendez, Computer-based management environment for an assembly language programming laboratory, Comput Appl Eng Educ 15 (2007), 41–54.

[18] D. Spinellis, P. Zaharias, and A. Vrechopoulos, Coping with plagiarism and grading load: Randomized programming assignments and reflective grading, Comput Appl Eng Educ 15 (2007), 113–123.

[19] L. Lamport, The part-time parliament, ACM Trans Comput Syst 16 (1998), 133–169.

[20] H. Garcia-Molina, Elections in a distributed computing system, IEEE Trans Comput 31 (1982), 48–59.

[21] D. Skeen and M. Stonebreaker, A formal model of crash recovery in a distributed system, IEEE Trans Softw Eng 9 (1983), 219–228.

[22] E. Chang and R. Roberts, An improved algorithm for decentralized extrema-finding in circular configurations of processes, Commun ACM 22 (1979), 281–283.

[23] D. Skeen, A formal model of crash recovery in a distributed system, IEEE Trans Softw Eng 9 (1983), 219–228.

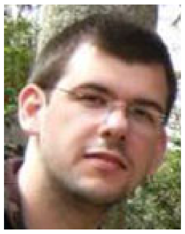
[24] Y. Afek and E. Gafni, Time and message bounds for election in synchronous and asynchronous complete networks. Proceedings of the Fourth Annual ACM Symposium on Principles of Distributed Computing (PODC ’85), pp 186–195.

[25] R. Golding and D. Long, The performance of weak-consistency replication protocols. UCSC Technical Report UCSC-CRL- 92-30. (1992).

[26] R. Golding, “Weak-consistency group communication and membership.” Ph.D. thesis, published as technical report UCSC-CRL- 92-52. Computer and Information Sciences Board, University of California, Santa Cruz. www.soe.ucsc.edu/∼golding/papers/ucsc-crl-92-52.pdf (1992).

BIOGRAPHIES

Joan Manuel Marquès is an associate professor of distributed systems in the Computer Science Department at the Open University of Catalonia as well as Researcher at the Internet Interdisciplinary Institute. He holds a PhD and MS degrees in computer science from the Technical University of Catalonia. His research interests include design of scalable and cooperative Internet services and applications, distributed computing, and collaborative learning (dpcs.uoc.edu/marques).



Daniel Lázaro is a PhD student at the Open University of Catalonia. He holds a MS in Computer Science from the Technical University of Catalonia. His research interests include scalable distributed algorithms and applications, service and task deployment in decentralized and cooperative environments and peer-to-peer systems (dpcs.uoc.edu/lazaro).

Angel A. Juan is an associate professor of simulation and data analysis in the Computer Science Department at the Open University of Catalonia as well as a Researcher at the Internet Interdisciplinary Institute. He also collaborates, as a Lecturer of Computer Programming and Applied Statistics, with the Department of Applied Mathematics I at the Technical University of Catalonia. He holds a PhD in Applied Computational Mathematics (UNED), an MS in Information Systems & Technology (Open University of Catalonia), and an MS in Applied Mathematics (University of Valencia). His research interests include both service and industrial applications of modeling & simulation, quantitative data analysis, and computer supported mathematical e-learning. He has published several papers in international journals, books, and proceedings regarding these fields. As a researcher, he has been involved in several international research projects. He is an editorial board member of the International Journal of Data Analysis Techniques and Strategies as well as of the International Journal of Information Systems and Social Change, and a member of the INFORMS society (ajuanp.wordpress.com).

Xavier Vilajosana Guillen is an associate professor at the Universitat Oberta de Catalunya (UOC) and CTO at WorldSensing. Xavier received his MS degree of Computer Science Engineering at the Universitat Politecnica de Catalunya, in 2004, and his Ph.D. in Computer Science from the Universitat Oberta de Catalunya at early 2009. During 2007 and 2008, he was a visiting researcher at the France Telecom R&D labs (Issy les Moulineaux, Paris, France) where he developed a Grid resource scheduler using economic mechanisms which also was the topic of research of his thesis and one of the research results of the Grid4All European project. His current interests include efficient scheduling and resource allocation techniques in distributed systems, distributed algorithms and concurrent programming. He is currently responsible for the Computer Networks and Advanced computer networks subjects at UOC.

Marc Domingo is a MS student at Universitat Politècnica de Catalunya. He received a degree in Computer Sciences from Universitat Politècnica de Catalunya. His research interests include security and peer-to-peer systems.

Josep Jorba is an associate professor of Computer Architecture and Operating Systems at the Universitat Oberta de Catalunya. His research interest include design of middleware for distributed and Grid/Cloud Computing environments, and design of tools for automatic performance analysis of distributed and Parallel applications. He has a PhD in computer engineering from Universitat Autònoma de Barcelona.