Session 1394

# A Pedagogy to Support Modern Concepts in Distributed Systems Courses

**Bina Ramamurthy, Eric Crahen**
**Computer Science and Engineering Department**
**University at Buffalo**
**Buffalo, NY 14260-2000**
**bina@cse.buffalo.edu, crahen@cse.buffalo.edu**
**716-645-3180 (108)**

## Abstract

Recently, there have been many advances in technology and infrastructure to meet the changing needs of the application domains. These changes have resulted in the development and adoption of a rich set of novel concepts in distributed systems. For example, lookup, discovery, custom event and event handling, runtime reflection, callback and service leasing are just a few of these new ideas. Traditional approaches to teaching Distributed Systems courses do not cover these newer concepts for reasons ranging from lack of support from the existing framework to the fear sacrificing coverage of some fundamental concepts. Moreover, many of the textbooks used do not cover these concepts. In this paper, we present a pedagogy that seamlessly integrates the modern concepts to the existing conventional methods for teaching distributed systems. We propose a set of laboratory experiments that will not only illustrate how to integrate the newer concepts into existing framework but will also provide the students with hands-on experience in the application of these concepts. The design and description of three laboratory projects that cover newer topics in Distributed Systems, namely, (i) location-independence, (ii) active discovery and (iii) interoperability and persistence are shown. These projects will serve as models for development of similar projects illustrating other concepts of interest. A major contribution of this paper will be the pedagogy that will build bridges between the rapidly advancing modern technologies and the traditionally rigid curricula.

## 1. Introduction

A distributed system is a collection of autonomous computers linked by a network and supported by software that enables the collection to operate as an integrated facility [14]. A course in distributed systems covers the design principles, the architecture, the components, services, and the issues in concurrency, transactions and security, client-server models, and integration models and other related material. This course is typically offered as an upper division (senior level) undergraduate course at some schools (See

Table 1) and also as an entry-level graduate course. The two of the textbooks commonly used for the course are the one by Steen et al [12] and the other one by Colouris et al [2]. With the advent of the Internet and the information technology revolution it brought about, distributed systems have evolved from a simple file server or data server into a multi-tier system integrating applications, data and resources. The size, complexity and the scale of the computations and data grew exponentially. To handle the complexity and scale, newer distributed architectural models such as Common Object Request Broker Architecture (CORBA) [7], Remote Method Invocation (RMI) [4], Microsoft's .NET [5] and Java 2 Enterprise Edition (J2EE) [3] have been introduced. Associated with these architectures are a set of protocols, procedures and algorithms to support the features of the new architectures. The technology wave created by these and the demands of the current application domains have greatly increased the importance of proactively preparing our students for this technological workplace. One of the ways of preparing them is to introduce these modern concepts into appropriate courses in their curriculum. We feel that the distributed systems course is the most appropriate of them all to introduce such concepts. The challenge then is to introduce newer concepts without cannibalizing the existing core curriculum of the course. Traditional textbooks do not provide a solution. It is understandable that the authors of textbooks find it difficult to keep up with the pace of introduction of these newer concepts and technologies. We describe in this paper a pedagogy that utilizes the laboratory component of a distributed systems course to teach the modern concepts. This solution for teaching modern concepts has a broader impact than on a distributed systems course. It can be used for other courses with laboratory component and also for adapting and testing the viability of other concepts in the future.

We studied the current status of distributed systems courses offered at selected schools and the details we discovered are explained in the Section 2. The laboratory-based pedagogy is explained in Section 3. This section gives the templates for the laboratory projects including the source code and structure, and methods for adapting the pedagogy. Finally we summarize our experience with the distributed system course in our curriculum.

## 2. Current Status

Many universities do have a distributed systems course even though the coverage varies widely from a theoretical course at Yale University to a hands-on practical course at John Hopkins University. In the Table 1 we have put together a random list of schools that have a well-established programs in the distributed systems area. This allows us to study the trend and the variety of distributed systems curriculums. We have provided links to most of the programs for the readers to examine these courses for themselves. Most of the schools use a textbook. Others have a recommended reading list. Both these approaches do use laboratory-based projects in the courses. The pedagogy we propose is applicable to both the approaches.

| School Name | Course Number and link | Text |
|---|---|---|
| Stanford University | CS 244 Distributed Systems www.stanford.edu/class/cs244b/#Materials | [10] |
| U. of Illinois, Urbana-Champaign | CSE 328 Distributed Systems: www-courses.cs.uiuc.edu/~cs328 | [11] |
| Brown University | CS176 Intro. to Distributed Computing: www.cs.brown.edu/courses/cs176 | [15] |
| Yale University | CS 425 Theory of Distributed Systems: zoo.cs.yale.edu/classes/cs425 | [15] |
| Rice University | COMP 520 Distributed Systems: not accessible | [12] |
| Georgia Tech. | CS7210 Distributed Computing: www.cc.gatech.edu/classes/AY2003/cs7210_fall | [9] |
| U. of North Carolina, Chapel Hill | COMP243 Distributed Systems: www.cs.unc.edu/~kimk/comp243 | [6, 13] |
| John Hopkins University | CSE337/437 Distributed Systems: www.cnds.jhu.edu/courses/cs437/ref.html | [1] |
| U. of Texas at San Antonio | CS 5523 Operating Systems: vip.cs.utsa.edu/classes/cs5523s2001 | [2] |

**Table 1: Sample Distributed Systems Courses**

## 3. The Laboratory-based Pedagogy

The stepwise description of the proposed pedagogy, a case study involving a senior level distributed systems course with short descriptions of the projects, the project templates and some suggestions for adapting the pedagogy are given in this section.

### 3.1 Proposed Pedagogy

1. During the planning stages of the course decide on the one or more major concepts or ideas that you desire to introduce. For example, application level interoperability between J2EE platform [3] and .NET platform [5].
2. Establish the context and importance of each of the concepts for the course goals and make sure they warrant a laboratory project. In our course the concepts or ideas are solutions to many unsolved issues or problems with existing processes and systems. For example, transforming a simple naming service into a location-independent naming service.
3. Design the project description for the laboratory that explains the problem. The details such as the goals, requirements, protocols, procedures, documentation, example programs, expected output, outcome assessment methods and due date are provided. This is a critical step where students are encouraged to apply their knowledge, originality and creativity to create various alternate solutions.
4. Typical projects in the distributed systems area deal with elaborate environment and file structures. Provide a skeleton structure for the directory of files and environment. Explain the usage with an example. See the illustration in Section 3.2

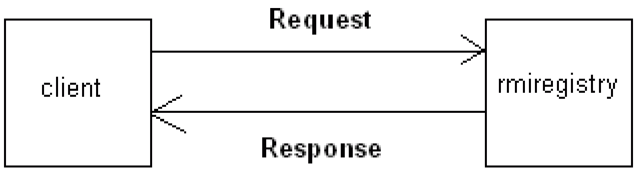
for more details of this step.
5. Explain any special tools such a make utility or build tools that may facilitate the development process.
6. Incorporate the project experiences into the lecture material and the assessment for the course.

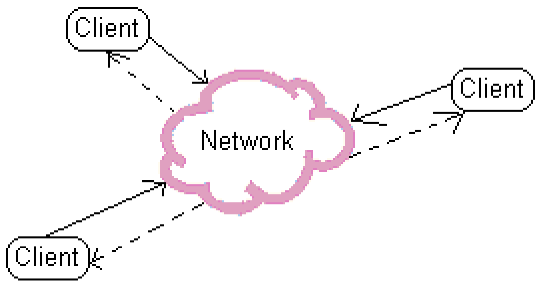
### 3.1 Case Study: Senior Level Distributed Systems Course

The proposed pedagogy was used in a senior level distributed systems course (CSE486) in the Computer Science and Engineering department at State University of New York at Buffalo (UB). Before introduction of the new pedagogy the projects used were on topics such as concurrency using multithreading, simple distributed file system, and simple use of CORBA objects. The problem with this approach is that there is no room for scientific inquiry, or application of creativity or originality. Moreover, in many of these cases cookie cutter solutions are available online for these projects. The pedagogy we stated above to addresses these issues.

We planned for three laboratories for the course and we chose the concepts first as prescribed by the pedagogy. The concepts chosen were: (i) location independent naming service, (ii) active discovery and (iii) combination of interoperability and persistence. For the first two we decided Java RMI as the technology for the first two and added on the enterprise java beans (EJB) and CORBA technology for the last one. The next step is establishing the context and provide a problem description with an explanatory diagram as shown in the Figure 1.
The Naming Service:

The RMI registry is based on a client/server system. The clients send a request to a known destination and await a response. All participants must be aware of the URL of the naming service to locate any services registered.





Creating a distributed system eliminates this client/server dependency. In such a system, a client does not need to know the exact location to send a request to. It can broadcast a request to a network and await its response. Knowledge of a services exact location is no longer an issue.

**Figure 1: A Location Independent Naming Service**

The first project requires the students to implement a location independent naming service using multicast discovery protocol. Sample programs are provided to enable the students to understand the multicast concept. The project description typically has an introduction, the brief but precise problem description, the plan, any new protocols and procedures to be followed, approaches to be followed, overall project requirements and resources available for the design and development of the solution. A project template as described in Section 3.2 that provides the directory structure and environment definition is also provided.

The second laboratory project builds upon the first project by enhancing simple discovery of services into proactive announcement by the naming service whenever objects representing the services are bound or unbound. In this case, the new discovery protocol and the new behavior are provided in the project description besides the usual material.

The third laboratory project illustrates the method to address two important issues in a single project. In this case the two are interoperability and persistence. The application is a news server. The clients in this system are CORBA-based, working with a Java RMI server which uses callbacks to deliver the news to the clients. Persistence is realized through EJB technology. The details are shown in Figure 2. The initial simple client-server architecture that is taught in a traditional course has been enhanced to have location independence, active discovery, callback features, interoperability and persistence. Working with these projects also exposes the students to the various technologies that they use to solve the problems.

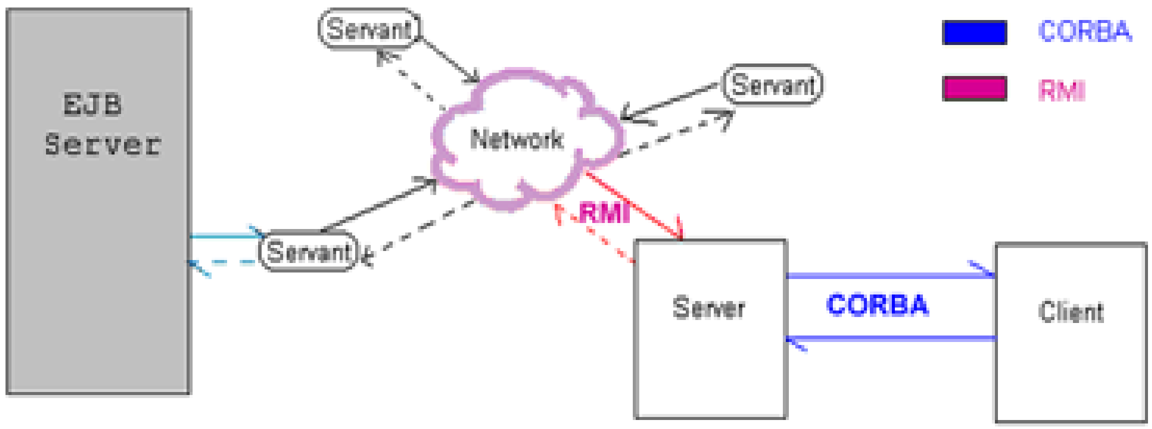


**Figure 2: A Persistent News Server with EJB, RMI Server and CORBA Client**

## 3.2 Project Templates

To focus the studies for these labs a set of project templates were created and distributed to the students. The project template contains a *Makefile* and a directory structure that will simply the process of building and packaging the sources. Providing this allows the students to focus on the real task at hand, understanding the distributed systems concepts, and not on mundane tasks of project management. Another benefit is that the instructor is provided with a uniform interface for all the projects that can help to automate some of the testing of the project and streamline grading them. See the sample template available at http://www.cse.buffalo.edu/~crahen/asee

The directory structure is simple, and laid out briefly as follows. The **bin** directory contains some utilities that assist in the compilation process. The **classes** directory hold the class files generated from the students source code. The **doc** directory holds any documentation for the project. The **html** directory holds the project descriptions as they were posted on the class website. The **lib** directory contains the main library the students are developing throughout the course of these projects. The **sources** directory contains the source code for the library and for the demos that the students will create to test their library.

In order to build the project, all a student needs to do is to run the GNU make utility in the root of their project. The *Makefile* provided would automatically detect and compile all sources, as well as locate any classes that need to have CORBA or RMI stubs generated. All the students need to do is edit or create files in the source directory.

Once the sources are compiled, the students will find a jar file in the **lib** directory. To use the Naming class replacement they are implementing, the use the **Xbootclasspath** option to tell the Java Virtual Machine (JVM) to search their jar (java archive) for the sdk (system development kit) classes before the standard location. For example,

```
java –Xbootclasspath/p:lib/project1.jar example.aClassName
```

can be used to run a demo that uses their projects Naming implementation.

## 3.3 Adapting the Pedagogy

For the initial adaptation of the pedagogy for a distributed systems course the package of the three projects described in this paper along with the solved samples can be used. In the subsequent offering of the course the concepts, coverage and the solutions can be varied as needed. This pedagogy works well not only for distributed systems courses but also for other courses in a state of flux. We have made use of this pedagogy for a Computer Science introductory course (CS2) to introduce the concept of design patterns [8].

## 3.4 Successes and Failures

We practiced the pedagogy in the Distributed Systems course of 20 students in the Spring 2002. From the course evaluations we could see that students really learned many new concepts. If the success could be measured by a more tangible metric, we found that 25% of the class found employment because of the skills they acquired in the course. At least two of the employers were asking for students who have taken this course. There was but one student who found it very difficult to meet the rigors of the laboratories. Later we found out that this student did not complete the prerequisite for the course before taking the course and so was unprepared.

In the Fall2002 we extended the methodology described in this paper to a critical course (CS2) in our curriculum with multiple sections and an enrolment of 200 students per semester. In general our experience was good. We are yet to analyze the data in more detail.

## 4. Summary

We presented a pedagogy that helps bridge the gap between rigid curricula and rapidly evolving technologies and application domains. The pedagogy exploits the existing laboratory component available in almost all engineering courses. This pedagogy was successfully used in a distributed systems courses where we measured the success by the employability. We have summarized our successes and failures. The complete project descriptions and sample code are packages and is available online at http://www.cse.buffalo.edu/~crahen/asee for instructors to readily apply this pedagogy in their courses. For future work we are planning a more comprehensive laboratory package for the distributed systems course and also experimentation with other courses that may benefit from this pedagogy.

## Bibliography

[1.] K.P. Birman. Building Secure and Reliable Network Applications. Manning Publications Co., 1996.

[2.] G. Coulouris, J. Dollimore, and T. Kindberg. Distributed Systems: Concepts and Design, Addison-Wesley Publishing Co., 2002.

[3.] Java 2 Platform, Enterprise Edition, http://java.sun.com/j2ee/, November 2002.

[4.] Java Remote Method Invocation (Java RMI): Distributed Computing for Java, White paper, http://java.sun.com/marketing/collateral/javarmi.html

[5.] Microsoft's .NET: Microsoft XML Web Services Platform, http://www.microsoft.com/net/defined/default.asp.

[6.] S. Mullendar. Distributed Systems. Addison Wesley Publishing Co., 1993.

[7.] Object Management Group, *The Common Object Request Broker:Architecture and Specificatio*n, 2.5 ed., Sept. 2001.

[8.] B. Ramamurthy, and P. Ventura. A Practical Approach to Introducing Design Patterns in CS1 and CS2", submitted to 8th Annual Conference on Innovation and Technology in Computer Science Education (ITiCSE2003), Macedonia, Greece, June 30 - July 2, 2003.

[9.] Reading list for Stanford Univeristy's Distrbuted Systems Course CS244B: http://www.stanford.edu/class/cs244b/#Materials

[10.] Reading list for Georgia Tech's Distributed Systems Course CS 7210:
www.cc.gatech.edu/classes/AY2003/cs7210_fall

[11.] P. Sinha. Distributed Operating Systems: Concepts and Design. IEEE Computer Society Press, 1996.

[12.] M.V. Steen and A. S. Tannenbaum. Distributed Systems: Principles and Paradigm. Prentice Hall Inc., 2002.

[13.] W.R. Stevens. Advanced Programming in Unix® Environment. Additon-Wesley Pulbishing Co., 1992.

[14.] Technical University of Vienna, The Distributed Systems Group,
http://www.infosys.tuwien.ac.at/Teaching/DS/1997/ds1/tsld003.htm

[15.] J. L. Welch, J. Welch and H. Atiiya. Distributed Computing: Fundamentals, Simulations, and Advanced Topics, McGraw-Hill Co., 1998.