ShareMe: Running a Distributed Systems Lab for 600 Students With Three Faculty Members

Clemens Kerer, *Student Member, IEEE*, Gerald Reif, *Student Member, IEEE*, Thomas Gschwind, *Member, IEEE*, Engin Kirda, *Member, IEEE*, Roman Kurmanowytsch, and Marek Paralic

***Abstract*—The goal of the distributed systems (DS) laboratory is to provide an attractive environment in which students learn about network programming and apply some fundamental concepts of distributed systems. In the last two years, students had to implement a fully functional peer-to-peer file sharing system called ShareMe. This paper presents the approach the authors used to provide the best possible support and guidance for the students while keeping up with ever-rising participant numbers in the laboratory course (approximately 600 last year), as well as managing budget and personnel constraints. The learning environment is based on Web and Internet technologies and not only offers the description of the laboratory tasks but also covers electronic submission, a discussion forum, automatic grading, and online access to grading and test results. The authors report their experiences of using the automated grading system, the amount of work required to prepare and run the laboratory, and how they deal with students who submit plagiarized solutions. Furthermore, the results of student feedback and evaluation forms are presented, and the overall student course satisfaction is discussed. Detailed information about the DS laboratory is available at http://www.dslab.tuwien.ac.at**

***Index Terms*—Automated grading, distributed systems, e-learning, peer-to-peer, web-based education.**

I. Introduction

THE main teaching responsibility of the Distributed Systems (DS) group at the Vienna University of Technology, Vienna, Austria, is to hold an introductory course on distributed systems for fifth-semester computer science students. The course consists of a lecture and a laboratory. According to the curriculum, the learning objectives in the laboratory comprise deepening the theoretical concepts taught in the lecture (e.g., various models of distributed systems, unicast versus multicast, connection-oriented versus connectionless communication, concurrency, name services, distributed objects, remote method invocation, and network security) and applying them in practice. Because of the popularization of information technology (IT)-related studies, the number of computer science students has increased over recent years. With no enrollment limits and free access to Austrian universities, the number of students taking the DS laboratory reached more than 500 in 2002 and almost 600 in 2003.

Consequently, the authors had to think about how to reach the desired teaching objectives and keep a high level of student support in light of the continuously growing number of students and limited human resources (three faculty members and a few undergraduate laboratory assistants). Thus, one of the goals was to minimize the workload for grading, while providing high-quality and timely support to the students.

The space limitations of the DS laboratory was another important consideration. It consists of about 20 personal computers and two server machines. Because of the high number of students registered for the laboratory, most of whom work shortly before the laboratory deadlines, support for working outside the laboratory (e.g., on other machines at the university, at home, etc.) was crucial. The laboratory assignments were, therefore, designed in such a way that they could be implemented with a computer with a Java software development kit 1.4.x. Java is used as the programming language because the students learn the language in other courses before they take the DS laboratory.

One of the methods previously used to handle a large number of students was group assignments. As a result, teams of three or four students would work together, distribute the various tasks, and implement the group's solution in a joint effort. Developing software in such a small team also mimics the situation in many companies; the idea was to get an additional benefit for the students from this experience. In many cases, however, only one or two students in the group did all the work, while the rest of the group just earned the points. Since the goal in the laboratory is to ensure that every single student knows how to do network programming, this situation was unacceptable, and individual assignments were introduced. Although necessary, this decision resulted in three to four times the number of submissions to grade and a heightened temptation for some students to submit plagiarized solutions.

The remainder of this paper is structured following the step-by-step approach of instructional design [1]. Section II discusses the possible instructional strategies to meet the learning objectives introduced in this section. It further illustrates the ShareMe distributed application the students have to implement in the laboratory. Section III details the course material and the laboratory environment developed to support the students. Section IV discusses various grading schemes, the automatic grading system, and the experiences gained.

Manuscript received September 17, 2003; revised January 27, 2005.

C. Kerer was with the Vienna University of Technology, 1040 Vienna, Austria. He is now with Microsoft, Seattle, WA 98004 USA (e-mail: kerer@infosys.tuwien.ac.at; reif@infosys.tuwien.ac.at; kirda@infosys.tuwien.ac.at).

G. Reif and E. Kirda are with the Vienna University of Technology, 1040 Vienna, Austria (e-mail: reif@infosys.tuwien.ac.at; kirda@infosys.tuwien.ac.at).

T. Gschwind is with the IBM Zurich Research Laboratory, CH-8803, Rūschlikon, Zurich, Switzerland, and also with the University Zurich, Switzerland (e-mail: gschwind@infosys.tuwien.ac.at).

R. Kurmanowytsch is with Comprendium, 1020 Vienna, Austria (e-mail: kurmanowytsch@infosys.tuwien.ac.at).

M. Paralic is with the Technical University of Kosice, Slovakia (e-mail: marek.paralic@tuke.sk).

Digital Object Identifier 10.1109/TE.2005.849740

Section V presents the evaluation results and reports on the overall effort to develop and run the laboratory. Section VI points out planned changes, and Section VII concludes the paper.

II. THE SHAREME FILE-SHARING APPLICATION

To address the requirements presented in the previous section, the laboratory design combines two models of distributed systems (peer-to-peer and client–server), demonstrates concurrency and synchronization issues in several multithreaded servers, and explains how to secure network communication using digital signatures and a simple public key infrastructure (PKI). Furthermore, it covers different communication styles in distributed systems with the help of programming techniques, such as sockets, the user datagram protocol (UDP), multicast, remote method invocation, and the Common Object Request Broker Architecture (CORBA).

When the authors started developing the laboratory, they first formulated the instructional strategy [2]. One option was to structure the laboratory into small, independent assignments, each demonstrating a separate network technology. Using small assignments has the advantage that the students concentrate on the technology at hand without worrying about integration problems. On the other hand, small assignments are often artificial and do not serve a practical purpose. The other option was to use a larger application that combines all the technologies and shows how they work together. It also allows for a more interesting application that, in the end, fulfills a useful task. In addition, incremental development of an application provides practical experience with a software engineering technique. The problem in the latter case is that 1) the application has to be structured in a way that it can be gradually implemented by the students and 2) the students have to integrate the various subtasks.

Eventually, the authors decided that the benefits of seeing multiple technologies working together and implementing a useful application outweigh potential integration problems. The laboratory team thus tried to come up with an up-to-date scenario that not only meets the learning objectives, but also interests the students. Currently, ubiquitous peer-to-peer file sharing seems an adequate application domain. In a peer-to-peer system, each peer is a computing device that serves as a server and as a client.

The challenge when designing the peer-to-peer application was twofold. First, the learning objectives introduced in the first section needed to be met. Second, the system had to be structured in such a way that the students can incrementally implement, test, and use their applications.

The following four subsystems make up the final ShareMe system:

1) the peer-to-peer network infrastructure;
2) the search facility within the peer-to-peer network;
3) a way to secure the communication among peers;
4) the user interface that enables file exchange.

The next sections present the separate tasks in greater detail, demonstrate how the tasks build on each other, and show how the learning objectives map onto the tasks.

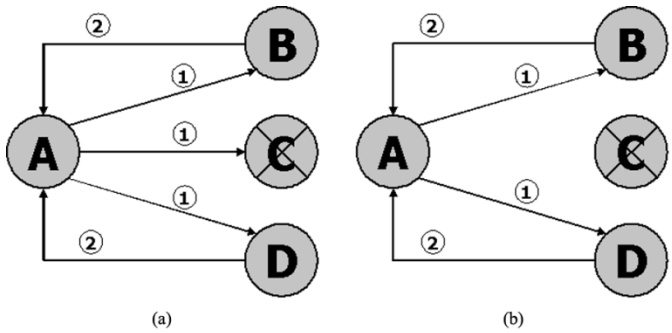

Fig. 1. (a) Exchange of multicast IAmAlive messages among peer **A** and the other peers (messages exchanged among the other peers are omitted for clarity). (b) Issuing search requests and responding with search.

A. Laboratory 1: The Peer-to-Peer Network Infrastructure

The aim of the first laboratory assignment is to introduce the peer-to-peer model, work with connectionless communication (UDP unicasts and multicasts), and demonstrate synchronization issues in multithreaded servers. First, the students create their own "peer" that participates in the peer-to-peer infrastructure. Each "peer" is a multithreaded server with threads for stopping the peer upon receiving an UDP shutdown message, making the peer known in the network, finding other peers, and reacting to changes in the network topology.

To maintain a list of active peers in the peer-to-peer network, every peer periodically sends an IAmAlive UDP datagram packet to a common multicast address. All peers that listen on the multicast address then know what other peers are currently active in the network. The design choice of using multicast communication obviously limits the system's scalability to local area networks. While this restriction is not severe for teaching purposes and does give the students a chance to use multicast communication in their programs, the design choice provides a good exercise for students to think about the advantages and disadvantages of using multicast communication for file sharing and discussing alternatives.

In Fig. 1(a), first, peer **A** sends multicast UDP IAmAlive messages to all other peers in the local network. Peers **B** and **D** receive the message and know that peer **A** is still available; peer **C** is currently not active. Second, at the same time, the other available peers (**B** and **D**) also send IAmAlive messages, which are received by peer **A** which, in turn, knows that apart from itself, two other peers are running.

B. Laboratory 2: Searching for Files

After the students have established connectivity in the peer-to-peer network, the next task is to search for files. In general, search criteria in peer-to-peer networks can be complex meta data queries. In the ShareMe system, simple filename-based queries are used. Thus, if the name of a file matches or contains the entered search string, the file is added to the search result. This task covers remote method invocation, the client–server model, and naming services from the learning objectives.

Searching is done by iteratively querying all peers that are currently in the list of active peers. Search queries are implemented as remote method invocations (Java RMI). Hence, every

peer implements an RMI server (to answer incoming search queries) and an RMI client (to issue search requests to other peers). Accessing RMI servers requires some sort of naming service being available to look up the server implementation (i.e., stubs for remote method invocations). Java RMI uses a registry for this purpose. Instead of running a central RMI registry that the students could use, the students had to concern themselves with naming services since the task description requires them to run their own registry and register their server implementation with it.

Fig. 1(b) illustrates a typical search request. First, based on the list of known (and active) peers, peer **A** iteratively issues search requests to peers **B** and **D**. Second, the contacted peers search their file base, and each responds with a list of matching filenames.

C. Laboratory 3: Making Searches Secure

Laboratory 3 addresses securing the communication between peers using digital signatures and a key server as trusted third party. More precisely, the goal is to sign search requests and responses digitally to make sure that no fake queries and/or fake responses can occur within the system. The signatures are based on MD5 (Message Digest 4, one-way hash function) hash sums and the Rivest, Shamir, and Adleman (RSA) public key encryption algorithm, which requires the creation and administration of a public–private key pair.

The private key of a peer is only stored locally on the peer machine. The public key, however, has to be exchanged with the other peers in order to allow them to verify signed search requests. For this purpose, the laboratory team runs a central key server where the public keys of peers are registered and can be retrieved by other peers. This server is implemented as a CORBA service. The students implement the corresponding CORBA client to register public keys and retrieve keys of other peers.

Eventually, the Laboratory 2 implementation is extended to sign and verify search requests and results. Fig. 2(a) illustrates this extended behavior. First peer **B** creates and signs a search request and sends it to peer **A**. Second, peer **A** contacts the key server **KS** to retrieve the public key of peer **B** and verifies the signature of the search request. Third, in the next step, peer **A** searches its file base, constructs a search result, and signs it. Finally, it sends the response back to peer **B**, where the digital signature is verified once again before the result is processed.

D. Laboratory 4: Adding an HTML User Interface

In the previous laboratory assignments, the students create the core functionality of the ShareMe system but do not include a user interface (other than the command line) or a possibility to actually exchange (download) files. In Laboratory 4, the students implement their own multithreaded Web server and, in doing so, get to know the internals of the HTTP protocol, the difference between static and dynamically generated Web pages, and the remaining technology from the learning objectives—TCP sockets.

The Web server provides user interface (HTML) pages, processes search requests submitted via HTML forms, and supports HTTP downloads. Search requests are forwarded to the existing

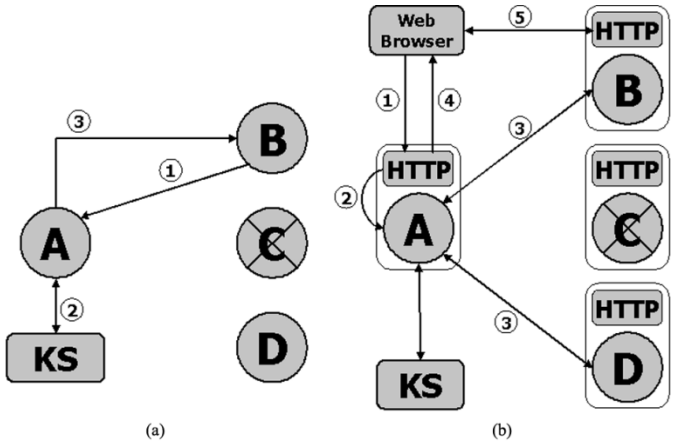

Fig. 2. (a) Exchanging secure search requests and responses using digital signatures. (b) Searching and downloading files using an HTTP server and a Web browser.

infrastructure, and the search results are converted back into HTML pages and displayed in the browser. Each file included in the search result is rendered as a hyperlink that supports the direct download of the file to the local machine. The students can then use any Web browser to interact with the ShareMe system.

Fig. 2(b) shows the integration of the HTTP component into the existing infrastructure.

1) From a Web browser, a search request is sent to a peer.
2) The Web server component decodes the search request and forwards it to the search component.
3) The same process as before is then executed. The request is created and signed and sent to all known peers in the network; the responses are collected and verified.
4) An HTML page is created to display the search result in the browser.
5) If the user clicks a file in the search result, it is downloaded directly from the peer that provides it (since that peer also has an integrated Web server).

III. Supporting the Students

One of the biggest problems in the distributed systems laboratory is coping with up to 600 students registered for the laboratory. With so many students, one of the key requirements is to fully support working remotely, be it online or offline.

A first measure for people with permanent Internet connections (e.g., other machines at the university or machines with broadband access) is to grant remote access to the laboratory machines via secure shell (SSH). Furthermore, the laboratory Website (http://www.dslab.tuwien.ac.at) provides access to all the documentation and required software.

Many students also opt for working offline at home. The laboratory team provides them with a CD-ROM containing all the documentation, software packages, the key server, tutorials, and API references. Thus, they can fully implement the laboratory assignments without being online. The laboratory Website is designed and implemented in a similar way to a distributed interactive learning (DIL) environment as described in [3] using the MyXML technology. MyXML [4] is a Web content management and generation application based on XML

(extensible markup language) and XSL (extensible stylesheet language) technologies. Using MyXML for building and maintaining the Website also facilitates the automated generation of a PDF version of all the documentation, which is offered for download and easy printing.

When several hundred students work on an application such as the ShareMe system, problems and questions are inevitable. Having ensured that the students could work in the laboratory on machines connected to the Internet, and even offline at any reasonable personal computer, the next issue is how to handle the problems and questions raised by the students. A frequently asked questions (FAQ) document covers the most likely ones and is included in the Website. A laboratory newsgroup where faculty members answer questions and where students can help each other is another major source of information. Finally, a group of graduate laboratory assistants are present in the laboratory for two to four hours per day to solve problems that are difficult to discuss electronically. As a last resort, the faculty members also offer e-mail support if the laboratory assistants cannot solve the problem.

Further services offered on the laboratory Website include the grading service, the laboratory status overview, and a Web interface to the public key server. With the grading service, students can query the number of points they have earned so far. The laboratory status overview shows all the peers currently active in the laboratory, including their names and the registry information. To ensure that at least some peers are running in the laboratory at all times, the laboratory team provides several peers that the students can use to test their implementations. The Web interface of the public key server lists the set of keys currently registered at the server and offers a possibility to revoke the registration after giving the correct password (set when registering the key earlier).

IV. GRADING STUDENT PERFORMANCE

The method used to grade students is always controversial since it requires taking several conflicting factors into account. The government that finances the university tries to keep the costs for the laboratory low. The university has to teach the students the principles of a distributed system and should ensure that it passes only those students who have mastered the subject and are able to apply it in practice as set forth in the curriculum. Some students, however, try to pass the lectures with minimal effort, resulting in a number of plagiarized solutions. From colleagues around the world, the authors know that the situation is similar at other universities.

The distributed systems course has already been offered for ten years at the Vienna University of Technology (albeit with adaptations because of new developments), and during that time the laboratory team experimented with different ways to grade student performance.

A. Automated Grading System

Korhonen *et al.* state in [5] that no significant difference exists between Web-based and classroom learning environments as long as 1) the training material is of high quality and 2) the feedback is timely and accurate. Building on these results, the DS laboratory provides extensive information on the Website and—to provide timely feedback—the whole management of deadlines, submissions, e-mail feedback, and grading of submissions is automated and does not require manual interaction. The submission system packs all the student files into an archive; sends it via e-mail to the test environment; and unpacks, compiles, and checks it there. Immediately after submitting a solution, the student receives a feedback e-mail with details on the files submitted, the time of the submission, whether it could be successfully compiled, and all necessary files included. In theory, the laboratory environment would provide a fully automated submission and grading system fulfilling the requirements listed in [6]. In the first year, however, 5%–10% of the students still managed to produce code that could not be graded automatically (e.g., because it crashed, hung, did not obey interfaces, or blocked the test engine). These cases were used to improve the submission system. The number of manually corrected submissions has dropped to less than 2% since then.

B. Group Versus Individual Assignments

Until 2001, about 350 students were registered for the laboratory, and the assignments were still solved in groups. Group assignments can become troublesome if the groups are required to manage themselves. The supervisors had frequent discussions with groups that failed because some of the group members did not implement their part, or copied the solution from other groups, or failed to show up to group meetings. These failures resulted in some students ending up on the losing side even though they had fulfilled their part.

In 2002, the ShareMe system was developed, and the laboratory team abandoned the idea of letting students solve assignments within groups since the additional benefit of team work is also the focus of a separate course in the curriculum. This decision resulted in four times as many submissions and led to about eight times as many plagiarized solutions.

C. Dealing With Cheaters

Plagiarized submissions are another major source of effort as also reported in [7]. Typically, students do not submit an exact copy of another solution but change it in a variety of ways. In most submissions containing plagiarism, at least the source code comments and the order of the Java `import` statements have been changed. Also renaming variables and parameter names, changing the order of variable declarations and/or methods, or changing the spacing of the source code are popular among students. In order to be able to detect and handle cheaters, a cheater detection system is used in the DS laboratory that first builds the abstract syntax trees of the students' code and then applies a proximity measure to estimate the probability that a specific submission was copied. Similar systems for detecting software plagiarism such as Moss [8], Plague [9], and JPlag [10] exist. The technique used in the laboratory is similar to that described in the literature ([11]) to identify code that has been "cloned" (i.e., copied) in mainframe systems. In older systems, software developers often cloned code so as not to have to write code from scratch. The software maintenance research that has been

done to identify cloned code has also proven to be useful in identifying students who cheat.

If the laboratory team suspects that a submission is directly copied or derived from another submission, the involved students are invited to a private hearing to resolve the issue. Even though comparing the submitted solutions with each other is done automatically, the preparation and execution of the hearings consumes considerable time and is equally unpopular among faculty and students.

D. Open Versus Closed Book Tests

For their final grade, the students can earn points for the programming assignments and the mandatory written test. The assignments and the test each contribute 50% of the points, and the students have to succeed in both parts. The written test usually consists of conceptual questions about the ShareMe system (e.g., to discuss scalability or fault tolerance issues of the system) and small programming assignments.

In real-world situations, programmers frequently use example code and look up API documentation. Therefore, the written exam was held as an open-book test. The students were allowed to bring their code to the exam. Some students, however, simply copied code (including parts that were not relevant to the problem) instead of using the code to get hints to solve the test questions. This situation was a signal that these students were not able to solve simple network programming tasks and, therefore, failed the teaching goal. It turned out that most of these students did not study for the exam but relied on the code they were allowed to use.

As a consequence, the exam was changed to be a closed-book test in the fall term 2003 and handed out the API documentation needed to solve the programming questions. In addition, the focus of the exam was the intent of the code and not the correct syntax. This was still a realistic test environment, since the students were able to look up the API if in doubt (e.g., to find out the name of a function or the order of its arguments). Hence, the students were forced to come better prepared to the exam, which resulted in more students meeting the teaching goals.

E. Required Versus Optional Assignments

In 2003, the distributed systems course was moved from the spring term to the fall term. The course was therefore offered in two consecutive terms, resulting in fewer students registering for the laboratory in the fall 2003 term, which allowed the authors to do some more experiments. The laboratory assignments were changed to be voluntary except for the first one. Consequently, the grades are only given on the basis of the written exam held at the end of the course, and students are encouraged to solve the laboratory exercises as preparation for the exam. Copied submissions are not an issue in this case, since the submitted laboratory assignments do not earn any points, and only the performance in the written exam determines the grade. The advantage is that it allows students to learn the subject in whichever form is suitable for them (alone or in groups) in a similar way to long-distance learning courses. Ragan *et al.* state in [12] that this change encourages the students to take personal responsibility for their learning.

The laboratory team conducted an online survey among the students in 2003 that asked them for their opinion as to whether the laboratory assignments should be mandatory. In contrast to Ragan *et al.*'s results, the survey showed that 109 out of 158 students are against the new setup while only 49 are in favor of the change. The students against optional laboratory assignments argue that this change would require them to take more responsibility in preparing for the test and that they do not believe that students would implement the laboratory if they are not forced to do so. They believe that one consequence of this change might be considerably poorer test results. The students in favor of the change say that questions and problems can be discussed more openly (e.g., source code could be posted to the forum) and that they would not have to implement the laboratories if they already knew the technologies.

F. Discussion

The authors have to stress that none of the above choices is a particularly bad one. With each choice, most of the students managed to acquire the skills set forth in the curriculum. The only difference is how the laboratory team deals with the relatively small number of students who are disinterested and only want to pass the laboratory without any effort on their part. Unfortunately, these students require the most time and work from the laboratory team members.

Based on the fall 2003 laboratory, the authors came to the conclusion that optional laboratory assignments in combination with a closed book test that determines the students' grades is a good tradeoff. On the one hand, the optional laboratory assignments allow students to solve the laboratory exercises alone or in groups. On the other hand, the written exam tests programming skills, conceptual understanding, and an assurance that each student's individual performance is graded.

V. Experiences With the ShareMe Laboratory

The distributed systems laboratory has been running in the described setup for two years and has had more than 1000 students participating in it. This section gives more information on the experiences of running the laboratory and presents an analysis of e-mail feedback and student evaluations. Since the feedback and overall rating was basically the same in both years, this section focuses on the data from the 2003 laboratory with 597 registrations.

One clear trend is that students increasingly prefer to work at home rather than in the laboratory. This observation is supported not only by the number of downloads from the Website or the number of CD-ROMs distributed (i.e., downloaded from the Website or sold) but also by analyzing the usage of the laboratory computers. Most of the students who did not work offline at home connected remotely to the two server machines (labsrv01 and labsrv02). The personal computers in the laboratory (labpc01 to labpc19) were hardly used. Note that all figures refer to the time the students were logged in on the laboratory machines; no information was collected on the time they worked offline at home.

TABLE I
USAGE OF THE LABORATORY EQUIPMENT PER MONTH

Month	labsrv01	labsrv02	19 lab PCs (labpcXX)
March	69.3%	15.7%	15.0% (=0.79% per PC)
April	69.7%	14.3%	16.0% (=0.84% per PC)
May	68.4%	14.5%	17.1% (=0.9% per PC)
June	68.1%	14.1%	17.8% (=0.94% per PC)
Avg.	68.9%	14.9%	16.2% (=0.85% per PC)

In total, the laboratory computers were used for approximately 30 000 hours from March to June 2003. This figure reflects the time that students were logged in on any of the laboratory servers or personal computers. Table I shows the usage of the laboratory machines. Obviously, the server machines were used most, whereas the 19 personal computers together (labpcXX) are responsible for a mere 15%–17% of the utilization. In other words, people were logged in on the server machines six times as much as on all the laboratory PCs together. Furthermore, labsrv01 was used much more than labsrv02, which has no technical reason since both machines are identical. It seems that the students just preferred the number 01 over 02. As a consequence, the servers were renamed to "pizza" and "pasta," which solved this discrepancy in the following term.

The students also made heavy use of the offered support services, such as the newsgroup or e-mail messages. During the 2003 spring term, the laboratory team received a total of 960 e-mail inquiries and ended up with 1532 postings in the newsgroup. A total of 494 (32.2%) of the postings (which is about seven per day) were replies from the faculty members. One serious problem with supporting the students via the newsgroup was that they apparently got feedback and replies to their questions too quickly. Soon, several students abused the newsgroup as a remote debugger; instead of thinking about a solution first, they just posted their problem (or even error message, stack trace, etc.) and waited for an answer. This resulted in an explosion of messages, which meant that many students could not keep up with the number of postings/threads in the newsgroup. As a consequence, the students kept asking the same questions that had already been answered.

Another source of feedback is an evaluation study that is conducted by the university. This survey usually does not have high participation. In the 2003 spring term, 51 students participated in this study. Table II shows the most important results of the study where a 1 means total agreement and a 6 means total disagreement.

At the end of the study's questionnaire, the students had the possibility anonymously to give positive and negative comments that were not covered by the prepared questions. On the positive side, the students appreciated the selected topic (peer-to-peer file sharing) and that they built a useful application as opposed to simple programs such as quote-of-the-day or echo servers. They also commended the good laboratory organization and liked the automatic submission/grading system. Some students, however, requested more extensive test reports. According to them, the information in the test reports was not detailed enough to understand easily what caused a problem in some test cases. In summary, the study shows that the students are satisfied with the laboratory overall.

While the laboratory team received many positive reviews saying that the documentation and task descriptions were detailed and extensive, some students criticized this fact and argued that the assignments were less of a challenge because everything was exactly specified. The problem here is that less experienced students need the detailed specification to be able to implement the laboratory; more experienced programmers, however, would prefer more flexibility in implementing their solutions. Another reason for the elaborateness of the specification is that the submissions were automatically tested and graded. This automation is only possible if the design of the application is well specified and the interfaces are fixed.

Another area of improvement pointed out by student feedback was the handling of updates of the documentation. When the documentation was updated to clarify the specification or fix a bug, a new version of the online and offline Websites and the PDF documentation was generated. Although the updates were indicated on the Website, the students requested some sort of change notification (e.g., via e-mail and in the newsgroup) that would inform them about the update and the changes. For the next year's laboratory, an e-mail notification service will be integrated.

From the authors' point of view, the major effort of running the laboratory falls into three categories: the student support, the grading effort, and the time spent handling cheaters. The one-time effort of developing and creating the laboratory assignments, the laboratory infrastructure, and the test environment is not considered here. The authors estimate that the total preparation effort amounts to approximately five to six person-months.

The effort going into student support (e-mail, newsgroup, personal meetings) inevitably comes with running such a laboratory and, while being considerable, seems to be within tolerable bounds. The grading effort is reduced significantly by automatically grading the submissions. What remains is to check manually the submissions that cause a problem during automatic grading and to grade the written laboratory test. Again, much work is required, but it can hardly be done more efficiently if the written test is not completely abandoned. Working with students who cheated, however, is the area where the laboratory team is not willing to spend so much time. In the 2003 spring term laboratory, the abstract syntax tree comparisons identified 76 submissions of the first laboratory assignment (out of 507, i.e., 15%) as potential copies. After manually rechecking the suspicious submissions to avoid wrong accusations, the students were invited to a private hearing. Of the 76 students who were interviewed, 65 were consequently excluded from the laboratory. The remaining 11 students were allowed to continue the laboratory since one could not conclusively prove that they cheated (although several clues existed). In total, the laboratory team spent more than a person-month on all the aspects in conjunction with copied submissions for the first laboratory. This excessive time is clearly unacceptable.

TABLE II
RESULT OF THE UNIVERSITY'S STUDY ON THE 2003 DISTRIBUTED SYSTEMS LABORATORY

Question	1	2	3	4	5	6	Avg.
The documentation is concise and informative	21	18	5	0	4	0	1.9
The faculty members are suffi ciently available	23	8	4	1	1	1	1.9
The faculty members are competent	21	7	6	4	0	0	1.8
The faculty members clearly explain the tasks	22	8	7	2	1	1	1.9
The effort for the lab is too high (1) - too low (6)	11	10	28	0	0	0	2.3
The level of the lab too high (1) - too low (6)	4	11	29	3	1	0	2.7
The lab helped me deepen my understanding	18	18	5	4	5	0	2.2
The grading scheme is adequate	16	15	5	6	3	4	2.5
I mark the lab as excellent (1) - very bad (6)	21	14	8	5	2	0	1.9

VI. FUTURE PLANS

For the next year's distributed systems laboratory, some short tutorials at the start of the semester are planned to reduce the amount of trivial questions in the newsgroup at the beginning of the laboratory. As mentioned, a notification service for updates to the documentation or download packages will also be integrated in the laboratory environment.

Another planned modification is to replace the popular newsgroup with a moderated Web forum. In such a forum, the messages can be better structured by topic or laboratory assignment, and the moderator has the capability of removing trivial, personal, or outdated postings. As a result, only questions relevant to many people should stay in the forum, and all students should be able to follow the discussions without being overwhelmed by the sheer amount of postings in the newsgroup.

To help the students focus on the programming tasks at hand, the laboratory team is currently evaluating the use of Eclipse [13] as a development environment. The advantage of such a system would be the automation of common tasks (compilation and execution of code, the submission of deliverables, etc.) and the benefits of a full Java integrated development environment (IDE) (syntax highlighting, autocompletion, code assist, source code refactoring, etc.). A prototype that extends the Eclipse platform with functionality convenient for the purposes of the DS laboratory has been well accepted by beta testers. This system will be installed on the machines in the laboratory and can also be downloaded and installed by the students at home.

Beyond the scope of the Vienna University of Technology, the laboratory environment is deployed at the Technical University of Kosice, Slovakia. The laboratory team also intends to submit the material for use by the ECET consortium [14] of which they are a member. The ECET group is a European-funded Europe-wide network defining a common curriculum in computer science for European universities. Further, the authors were contacted by a company that would like to use the laboratory environment to train their programmers internally. The future goal, thus, is to create a ready-to-install and well-documented package of the laboratory environment including the Website, all the source code, the test and grading environment, and the additional scripts (e.g., for cheater detection).

VII. CONCLUSION

Taking into account the growing number of students and the limited resources available at the Vienna University of Technology, personal support of each student is no longer possible. Nevertheless, to reach the teaching objectives, the laboratory team must ensure that each student receives adequate experience with the topics and technologies of network programming. To enable students to do so in the most flexible way, the laboratory environment not only provides the online Website with its services but also fully supports offline working at home. The submission and the grading of laboratory assignments is automated, and the students immediately get e-mail feedback. Further support with problems and questions is provided via the newsgroup which is much appreciated by the students and offers the opportunity for students to benefit from each other.

Although most students found the topic of the laboratory interesting, the number of copied submissions caused a major workload in the last two years. As explained in Section IV, optional assignments with a closed-book exam are a good way to support group work while filtering cheaters.

Based on discussions with colleagues from other universities, the authors expect the concepts and experiences reported in this paper to be useful for other educational organizations that have high numbers of students and budget and personnel constraints.

The student feedback that the authors have received so far shows that the ShareMe peer-to-peer scenario is appreciated and encourages the laboratory team to further improve the laboratory organization and environment.

REFERENCES

[1] Reference Guide for Instructional Design (2001, Sep.). [Online]. Available: http://www.ieee.org/organizations/eab/tutorials/refguide/mms01.htm

[2] W. Dick, L. Carey, and J. O. Carey, *The Systematic Design of Instruction*. Boston, MA: Pearson Allyn & Bacon, 2000.

[3] M. Khalifa and R. Lam, "Web-based learning: Effects on learning process and outcome," *IEEE Trans. Educ.*, vol. 45, no. 4, pp. 350–356, Nov. 2002.

[4] C. Kerer and E. Kirda, "Layout, content, and logic separation in Web engineering," in *Proc. 9th Int. World Wide Web Conf., 3rd Web Engineering Workshop*, Amsterdam, The Netherlands, 2001, pp. 135–147.

[5] A. Korhonen, L. Malmi, P. Myllyzelk, and P. Scheinin, "Does it make a difference if students exercise on the Web or in the classroom?," in *Proc. 7th Annu. Conf. Innovation Technology Computer Science Education*, 2002, pp. 121–124.

[6] D. G. Kay, T. Scott, P. Isaacson, and K. A. Reek, "Automated grading assistance for student programs," in *Proc. 25th SIGCSE Symp. Computer Science Education*, 1994, pp. 381–382.
[7] B. Cheang, A. Kurnia, A. Lim, and W. Oon, "On automated grading of programming assignments in an academic institution," in *Comput. Educ.*, 2003, vol. 41, pp. 121–131.
[8] (2004, Aug.) Moss home page. Moss: A System for Detecting Software Plagiarism. [Online]. Available: http://www.cs.berkeley.edu/aiken/moss.html
[9] Plague home page (2004, Aug.). [Online]. Available: http://www.csse.monash.edu/projects/plague/software.shtml
[10] JPlag home page (2004, Aug.). [Online]. Available: http://www.jplag.de/
[11] I. D. Baxter, A. Yahin, L. M. D. Moura, M. Sant'Anna, and L. Bier, "Clone detection using abstract syntax trees," in *Proc. Int. Conf. Software Maintenance*, 1998, pp. 368–377.
[12] P. Ragan, A. Lacey, and R. Nagy, "Web-based learning and teacher preparation: Lessons learned," in *Proc. Int. Conf. Computers Education*. Green Bay, WI, 2002, pp. 1179–1180.
[13] The Eclipse Project home page (2004, Aug.). [Online]. Available: http://www.eclipse.org/
[14] (2004, Aug.) ECET home page. European Computing Education and Training (ECET). [Online]. Available: http://ecet.ecs.ru.acad.bg/

Clemens Kerer (S'03) received the M.S. and Ph.D. degrees in computer science from the Distributed Systems Group at the Vienna University of Technology, Vienna, Austria, in 1999 and 2003, respectively.

He is currently with Microsoft, Seattle, WA. His interests include distributed systems, Web-based education, Web engineering, and software components.

Gerald Reif (S'03) received the M.Sc. degree in computer science from the University of Technology Graz, Graz, Austria, in March 2000. He received the Ph.D. degree from the Technical University of Vienna, Vienna, Austria.

After working as a Master's student at the Institute for Information Processing and Computer-Supported New Media at the University of Technology Graz, he worked at the Distributed System Group, Institute for Information Systems, Vienna, Austria, as a Research Assistant for the European Union IST project MOTION (MObile Teamwork Infrastructure for Organizations Networking). He is currently an Assistant Professor at the Technical University of Vienna.

Thomas Gschwind (M'03) received the M.S. and Ph.D. degrees in computer science from the Vienna University of Technology, Vienna, Austria, in 1997 and 2002, respectively, where he was working on the composition and adaptation of software components.

He was an Assistant Professor at Vienna University of Technology. In 2004, he joined the IBM Zurich Research Laboratory, Switzerland, where he is working on event correlation with IBM's Security Group. He is also affiliated with the University Zurich, Switzerland.

Engin Kirda (M'03) received the M.Sc. and Ph.D. degrees in computer science from the Vienna University of Technology, Vienna, Austria, in 1999 and 2002, respectively.

He is an Assistant Professor at the Distributed Systems Group, Vienna University of Technology. His interests include software, network, and Web application security; device-independent Web engineering; and distributed systems.

Roman Kurmanowytsch received the M.S. and Ph.D. degrees in computer science from the Vienna University of Technology, Vienna, Austria, in 1999 and 2004, respectively.

After working as a Visiting Researcher at HP Laboratories and as an Assistant of the Department of Distributed Systems of the Vienna University of Technology, he joined the software company Comprendium, Vienna, Austria, where he works as a Software Architect.

Marek Paralic received the Master's degree in informatics and Ph.D. degree in programming and information systems from the Technical University of Kosice, Slovakia, in 1995 and 2002, respectively.

His research currently focuses on distributed systems, especially mobile-agent-based systems and multiagent systems. He is an Assistant Professor at the Department of Computers and Informatics at the Technical University of Kosice, where he heads the Software Engineering Group. He teaches undergraduate course in programming and graduate course in distributed programming.