Implementing Remote Laboratories for Control Engineering: Foundations for Distance Learning

by

Carisa Bohus

A THESIS

submitted to

Oregon State University

in partial fulfillment of the

requirements for the degree of

Master of Science

Completed March 15, 1996

Commencement June 1996

AN ABSTRACT OF THE THESIS OF

Carisa Bohus for the degree of Master of Science in Computer Science presented on March 15, 1996.

Title: Implementing Remote Laboratories for Control Engineering: Foundations for Distance Learning.

Abstract Approved: Lawrence Crowl

Lawrence A. Crowl

Distance learning enables students to study effectively from remote locations. With televised lectures now commonplace, many disciplines can provide effective distance learning. While the lecture format serves many disciplines, it does not serve those that require laboratory work. Currently, there is little distance learning support for laboratory work.

Second Best to Being There (SBBT) is a system that supports the implementation of remote laboratories. In this thesis, we define a paradigm for interaction with remote laboratories. This paradigm is based on experiences in the local laboratory, and provides experiment control, laboratory presence, laboratory environment control, safety, and collaboration facilities. We describe the overall system architecture and the implementation of the experiment-independent system software. In particular, we describe how our software provides assured, safe, real-time control of experiments. We also describe the lessons learned participating in a multi-disciplinary development project.

The SBBT team has demonstrated remote use of SBBT on five separate occasions. These demonstrations show that SBBT is successful in making the remote laboratory experience effective.

Implementing Remote Laboratories for Control Engineering: Foundations for Distance Learning

by

Carisa Bohus

A THESIS

submitted to

Oregon State University

in partial fulfillment of the

requirements for the degree of

Master of Science

Completed March 15, 1996

Commencement June 1996

Master of Science thesis of Carisa Bohus presented on March 15, 1996

APPROVED:

Major Professor, representing Computer Science

Head of Department of Computer Science

Dean of Graduate School

I understand that my thesis will become part of the permanent collection of Oregon State University libraries. My signature below authorizes release of my thesis to any reader upon request.

Carisa Bohus, Author

TABLE OF CONTENTS

ACKNOWLEDGMENTS

I am very grateful to acknowledge the contributions of the SBBT project team. Burcin Aktan was responsible for the graphical user interface for the Lab Environment Control. He cabled the PC and workstation, and created all of the routines to transfer data between the machines, as well as writing demonstration code for the robot arm. He conducted several demonstrations in other cities, participated in many design discussions and helped me understand control-engineering concepts. Steve Wilcox, an Electrical and Computer Engineering Department technician, designed and built the Motor Control Interface, participated in many design discussions and gave me a lot of practical and personal support. Dr. Molly H. Shor has been a wonderful faculty sponsor to me. She provided support and advice for laboratory logistics, grant logistics, opportunities to show my work, and an understanding of control engineering pedagogy. I would also like to thank Dr. H. Rebecca Callison for her early and continued support for the real-time and safety aspects of the project. Thanks also to Dr. Bill Robinson, Ken Ferschweiler, John Sechrest, Steve Fulling, Tad Reynales, and Dr. Wojtek Kolodziej for their help and interest.

I would especially like to express my gratitude to my advisor, Dr. Lawrence A. Crowl, for all the work that he has done on my behalf. I now have a well-developed appreciation for computer science theory. His advice in software organization and implementation have made this work solid. His careful reviews have definitely made me a better writer. Thank you for working with me.

My committee members have been very supportive of my work, and I would like to thank Dr. Roger Graham for his encouragement. My longest standing committee member, Dr. Margaret Burnett supervised several of my research and writing efforts during my time at OSU. I thank you for expecting me to do my very best.

Several colleagues took the time to read my thesis and I appreciate their good humor! Sherry Yang, Michelle Franz, Mike Herbert, and Laurie Wayne have all given me valuable perspectives. Juliet Hyams deserves a heartfelt thanks for her constructive support of my writing process. I thank you all very much.

I also wish to acknowledge the State of Oregon Employment Division, and the NERO program for their financial support of my work.

Finally, to all my family and friends who never seemed to tire of hearing about my work, my warmest thanks!

TABLE OF CONTENTS (Continued)

LIST OF FIGURES

LIST OF TABLES

This thesis is dedicated to the loving memory of my grandmother, Lillian McKay Pullen.

Implementing Remote Laboratories for Control Engineering: Foundations for Distance Learning

CHAPTER 1
SECOND BEST TO BEING THERE (SBBT)

Distance learning has been an exciting development over the last several years. It is now commonplace to transmit a lecture on broadcast television, or over a computer network. Distance learning is an effective way for people to learn new material without travelling to a specific location. While the lecture format serves most disciplines, it does not serve those that require laboratory work. Currently, there is little distance learning support for classes with laboratory components.

Consider the situation of a control engineering student. Once a student designs and writes control code to run on an experiment, she must test it on the actual experiment. If she is on the same campus with the experiment, she can walk into the laboratory and turn the experiment on. However, if she is not within easy commuting distance, she will need software and hardware to provide for her remote lab experience. Due to the popularity of computer games, virtual reality, animation, and remote control of toys and television, many people are ready for the concept of working on equipment at a location different from their own.

This work explores the implementation of remote laboratories. We defined a paradigm for interaction with a remote laboratory based on experiences in existing labs. As a multi-disciplinary team, we implemented a system that supports the paradigm with new and existing software and hardware. We call our system Second Best to Being There (SBBT), in recognition that while being there in person is best, learning from a remote location is also a worthwhile experience. Our demonstrations from other cities in Oregon and California show that SBBT is successful.

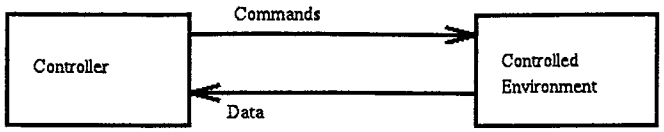

Figure 1. Simple Control Diagram.

1.1 Control Engineering and the Laboratory

Control engineering is an applied discipline, encompassing dynamical modeling, motion control, sensing, communications, and data acquisition. Generally there is a process or object to control, and there is some mechanism to control it (Figure 1). A very simple example is a thermostat. To heat a room, a person adjusts the thermostat dial to the desired setting. Next, the setting is compared to the most recent temperature reading. If it is too cold, the controller program sends a signal to the furnace to heat the room. In due time, the temperature test will match the thermostat setting, and the controller signals the furnace to turn off. This is an example of a control loop: Test the environment, compare results to the goal, transmit orders to make changes if needed, and repeat (Figure 2). Typical control engineering experiments in our lab are *x-y* positioning tables, DC motors, and robot arms.

Control engineering is an appropriate discipline to introduce distance learning laboratories. Many of the concepts needed to conduct a remote lab, such as controlling distant equipment, are paradigms control engineers already understand. Distant control of equipment is a concept that can now be applied to their learning environment, as well as to what they are learning. See Figure 3 for an example with a robot.

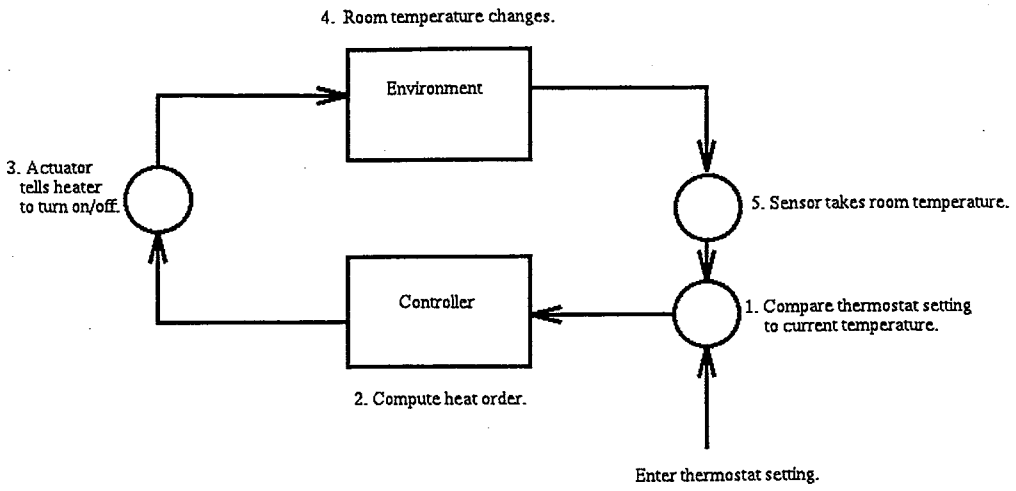

Figure 2. Thermostat Control Loop.

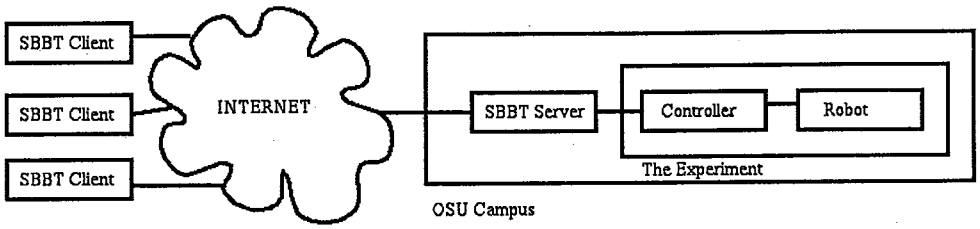

Figure 3. Internetwork context for SBBT.

1.2 The Internet: Infrastructure For SBBT

The Internet is a wildly successful experiment. It began as a project of the U.S. Department of Defense called the ARPANET (Advanced Research Projects Agency Network). Starting with four sites in December of 1969 [Tanenbaum, 1989], the Internet now connects 9.5 million hosts. In 1993, the Clinton administration of the U.S. government announced the National Information Infrastructure (NII), a new internetworking vision [Cochrane, 1994]. On a local level, the State of Oregon has received federal funds to develop the Network for Education and Research in Oregon

(NERO), which expands and increases the performance of the Internet. The Internet (Figure 4) provides the communications infrastructure for our work.

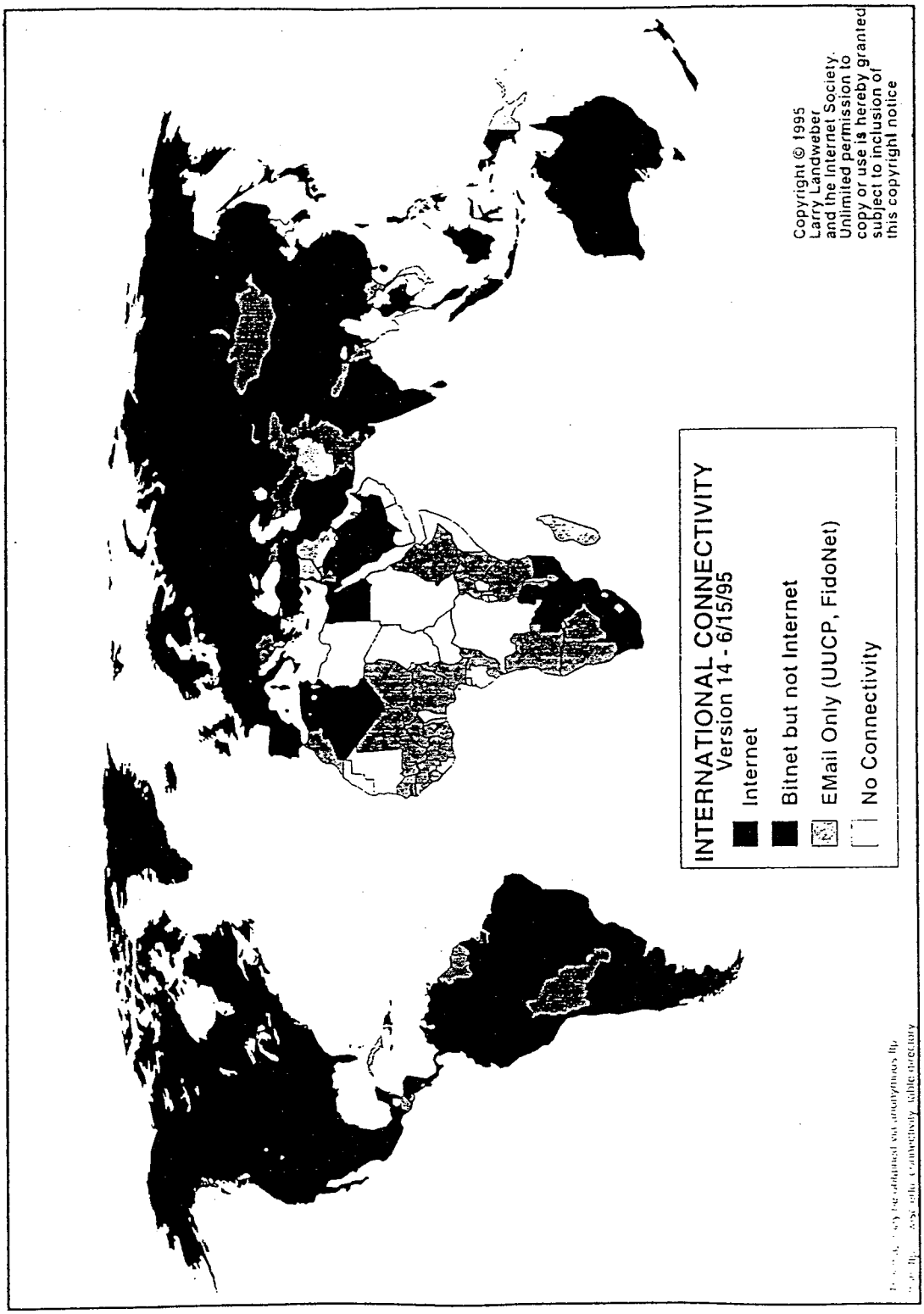

Figure 4. World Map of Network Connectivity.

1.2.1 National Information Infrastructure (NII)

At a recent technical forum, David D. Clark, Chief Protocol Architect for the Internet from 1981 to 1989, observed that, while the U.S. government has the vision for a gigabit network, it will not pay for it. Therefore, he concluded, the people who will do the work will also define the Information Superhighway itself. We provide a realistic application, not a benchmark, for evaluation of multimedia network traffic characteristics, and definition of the new functionality we want in future applications.

1.2.2 Network for Education and Research in Oregon (NERO)

Our project supports all the development goals of NERO [NERO, 1993]:

- *Collaborative tools for instruction or research*: SBBT is designed as a collaborative tool and an instructional laboratory. In addition, collaboration is a specific component of the application architecture.
- *Engineering and Computer Science Courses*: SBBT may be used as part of the lab component for control engineering classes.
- *Video Applications*: SBBT uses video as an important part of the application presentation. SBBT has a modular design and can easily accommodate advances in network video technology.
- *Real-time equipment/process control*: SBBT controls lab equipment in real time.
- *Access to resources (databases, supercomputer, telescope, etc)*: SBBT makes control engineering lab facilities at OSU available to other universities and researchers.
- *ATM transport and other network issues*: SBBT requires several kinds of network traffic with varying degrees of end-to-end reliability. When existing software did not meet our needs, new networking software was developed.

1.3 Application Design Approach

To create an application for remote laboratory work, we identified our goals and then shaped some strategies to guide us to a successful implementation. First, we wanted a working prototype as soon as possible. Second, we knew that there would be opportunities for follow-on work, so the system needed to have a solid base. Third, since different students would work on system development, it needed to contain robust components. A modular design would also make it easy to incorporate software and hardware improvements with little disruption. Fourth, the SBBT system itself needed to be general enough to be useful on a variety of equipment. Fifth, we recognized that student-to-student communication needs explicit support. Finally, there were real concerns to overcome from administrators as to whether a long-distance application that controls moving parts was allowed in an educational setting. Keeping these needs in mind, we shaped several strategies to guide our implementation choices. These strategies were: striving for modularity, achieving transparency to the user, using existing hardware and software whenever possible, encouraging collaboration, ensuring safety, and using real-time computing. Establishing clear guidelines helped us build a working implementation quickly.

1.3.1 Strategy 1: Striving for Modularity

Designing a modular system makes porting SBBT to other computer systems and different experiments more manageable. SBBT must be flexible in new environments since there are a wide variety of engineering experiments. Using modular components also simplifies transitions to new software and hardware. We planned to use tools we already had. They enabled us to build a flexible prototype quickly, knowing that we could upgrade easily when we needed more performance or functionality. Modularity also makes it easier to partition system development between people. By dividing up the development tasks, we could implement the system in parallel.

1.3.2 Strategy 2: Achieving Transparency to the User

So that the student could focus on her work, we kept the SBBT system as transparent as possible. We used standard user-interface conventions from the PC and UNIX worlds to keep the process predictable. For example, SBBT comes up on a UNIX workstation as a series of windows to place; students already know how to place windows from other applications. For user feedback from the PC controller, we used DOS-like responses. For the user, there is virtually no manual needed.

1.3.3 Strategy 3: Using Existing Hardware and Software

As stated earlier, we used numerous existing tools in order to have a working prototype on schedule. This strategy also benefits from modular design. As new and better technology becomes available, we can take advantage of it by easily swapping in new modules.

1.3.4 Strategy 4: Encouraging Collaboration

We made the controversial decision that scheduling the experiment resource should involve a social interaction between students. Using a computer scheduling program would serve the purpose, but not add to the lab experience. Remote learners are isolated and have fewer opportunities to form collegial relationships. Providing a reason for students to meet may lead to other conversations as well. Bradley et. al.[1993] observes that:

> While we concern ourselves with the impact of computerization on 'man-machine' or 'human-machine' communication, we must also direct our attention towards the qualitative aspects of communication 'human to human.' Some researchers identified these aspects as early as the 1960s, emphasizing their importance for well-being at work, though it is only during the last 2 or 3 years that a well-

> functioning and good communication in this broader sense has been recognized as an essential productivity factor.

The social aspects of working in a lab are also part of the training for future professional communications.

1.3.5 Strategy 5: Ensuring Safety

In order to deploy an application that controls moving parts semi-autonomously, we had to take safety very seriously. We conducted a high-level hazards analysis to reassure administrators on a technical level that we would protect students and equipment in the lab. Our safety approach is broad and makes sense for our lab.

1.3.6 Strategy 6: Using Real-Time Computing

When a student is working in a laboratory, everything happens in real time. Nearly every aspect of our remote lab paradigm requires a real-time component. If equipment is ordered to stop motion, that must happen as soon as possible. Real-time methodology ensures that critical traffic is predictable.

1.4 Related Work

There are three areas of engineering research that have bearing on SBBT: simulation systems, telerobotics, and large multi-location industrial applications. We were able to use the experience in these fields to make SBBT more responsive to distance learners. We examine projects in each of these areas and note how they influenced the design of SBBT.

1.4.1 Simulation Systems

Simulation systems abstract and represent important aspects of actual systems. Simulation is a cost-effective method to determine the characteristics for an optimal physical system more flexibly than building and testing numerous prototypes. Simulation can also be used to train operators on dangerous or unavailable equipment.

Lumelsky [1991], uses simulation systems to determine why human operators are ineffective at motion planning for telerobotic applications. In telerobotics, a human operator is part of the control loop. He proposes that there is a need for "effect of presence" so that the operator can be more accurate in environments with obstacles. He argues that people can be more effective if they have a better idea of the surroundings that the remote robot is operating in. He used simulations to test the kind of information people need to make more accurate movement with telerobots. We incorporated the idea of giving the remote student information about the laboratory setting into our system.

In [Stark, et. al., 1987], simulation was used to discover the visual interactions between the human operation and the manipulator system in an environment with communications delay. The human operators used joysticks to manipulate a simulation of a four Degree of Freedom (DoF) cylindrical-type robot arm. In their simulations they found human response time, (200 ms) to be the major time delay. This timing data helped us set our timing goals for SBBT system response.

Lee and Lee [1993], use simulation to monitor telerobotic force feedback and validate their techniques. They found to stabilize actions, a goal of three seconds latency, or delay, maintains fidelity of remote operations. This latency information was factored into our system response calculations. Human response time captured from systems in use provided a realistic starting point for SBBT.

Another important use of simulation is training. These types of simulation systems are very application specific and often require elaborate user interfaces. Miner and Stansfield [1994] describe a simulation system to train operators in retrieval of hazardous waste in underground storage tanks, such as those located at Hanford, Washington. On-the-job training is obviously undesired in this situation. In their training exercises, they found that making task-level commands available to operators made them more efficient.

Operators using their command set did not have to remember detailed sequences to do routine tasks. This information helped us define commands for SBBT at the right level of detail.

In [Woolf and Hall, 1995], several teaching simulation systems are reviewed and evaluated. They describe an "active learning environment" consisting of three key ingredients: 1) system parameters that the student may change, 2) enough domain knowledge to understand the assignment, and 3) simulation system response to student actions. We used these concepts in our design of SBBT.

Simulation systems are important, but do not take the place of working with actual systems. Not only is it more exciting to use a real experiment, a simulation by necessity will exclude some physical realities, and therefore consequences. Real experience is based on working with real equipment. SBBT allows remote use of an actual system.

1.4.2 Telerobotics

Telerobotics is an industrial field "where a human must be part of the control and decision-making loop...." [Lumelsky, 1991]. Bhatia and Uchiyama [1994] used a telerobotic system to study the time-delay effects on motion planning. They found that delays beyond one second make operators tentative. This timing information gave us additional insight that helped us set our timing goals.

Chan and Dubey [1994] used a telerobotic system to fine tune the amount of effort, or force, that the human operator must use. Their goal is to reduce the fatigue in the human operator. This paper helped us realize that the system itself should not take too much effort to understand and operate.

We know of two telerobotic applications on the World Wide Web. These applications [Bekey, et. al., 1995] and [Taylor and Trevelyan, 1995] are very amusing, but due to the nature of the web, there is no liveness. They use still pictures, which are downloaded and displayed after each user-directed move. When an application is fun to use, it can be a learning motivator.

1.4.3 Large Multi-location Industrial Applications

Klein, Lehoczky, and Rajkumar [1994], focus on resource management, or scheduling, in large distributed systems. These systems, which monitor manufacturing facilities, vessel traffic systems (harbors and airports), medical equipment, weather and seismic activity must be distributed but also must have tight controls. They use the divide and conquer method to establish tasks and their end-to-end deadlines. Even though SBBT is designed for one person to carry out an experiment entirely on their own, dividing tasks into subtasks makes development and operation more manageable.

Graves, Ciscon, and Wise [1992] and Kondraske, et. al. [1993] describe a distributed telerobotics operation spanning five sites in four cities. This testbed is being used to prepare for space and lunar stations by evaluating different communication protocols and control techniques. They require a modular environment in order to change operations dynamically. The system itself is under constant scrutiny. We don't want the student to even notice the SBBT system, but the flexibility due to modular components has proved to be a useful method for hardware and software upgrades.

Large multi-location industrial applications are not generally available to students for experimentation. SBBT is specifically designed for student use.

1.5 Outline of this Thesis

In the next chapter, we present a functional overview of the SBBT application. Chapter 3, describes the implementation, focusing on the main contribution of our work. In Chapter 4, Software Engineering Strategies, we describe the engineering management choices we made to implement SBBT. The final chapter concludes with a summary of our approach and accomplishments.

CHAPTER 2

FIVE-PART ARCHITECTURE: FUNCTIONAL OVERVIEW

To define a model of our remote lab, we remembered all of our lab experiences and attempted to partition the experience and characterize what each component represents. We identified the important aspects and imagined multiple computer-transmittable facsimiles. After each iteration of establishing the design, we reviewed our ideas with colleagues in as many fields as possible. This review process clarified what vocabulary effectively describes the ideas. Furthermore, it confirmed that the design would be general enough to support a variety of experiments. From this partitioning exercise, we defined a five-part architecture, each part capturing a significant aspect of laboratory interaction. The five parts are:

- The Experiment: This is the conventional experiment that the students need to do their work.
- Lab Presence: This part gives the remote students the feeling that they are in the lab.
- Lab Environment Control: This part replaces what local students do for themselves in the lab, such as turning the experiment on.
- Safety: This part protects the local students and equipment in the laboratory.
- Collaboration: This part represents the social component of working in a laboratory.

2.1 The Control Engineering Experiment

In this section, we describe our process for selecting an experiment for a remote laboratory. We base this description, with some modifications, on [Bohus, et. al. 1995].

The control engineering experiment is basically unchanged from its conventional form. Criteria for an experiment for remote use include: economics, logistics, and appearance. Current experiments in the OSU laboratory are *x-y* positioning tables, robot

arms, DC motor control, inverted pendulums, and magnetic suspension systems. We will evaluate these experiments against the criteria for illustration.

2.1.1 Economics

If substantial time, human, or financial resources are dedicated to design and build an experiment, it is a worthy candidate for remote students. For example, most labs will have an *x-y* positioning table, but a robot arm is usually more expensive. For the greatest economic value, the cost of simply replicating an experiment should be compared against the effort and expense of installing SBBT, keeping in mind that most SBBT costs occur only once, while replication costs scale. If replicas cost more than SBBT, it makes sense to use SBBT.

2.1.2 Logistics

The logistical considerations, which can be automatic or manual, are (1) remote power control, (2) safety for people and property in the lab, (3) the ability to run without human intervention, (4) a stable start position, (5) at least one reset position,[1] and (6) the ability to download control code. The importance of finding the proper solution to each of these concerns should not be underestimated. To keep an experiment running, many people play a role in the maintenance and upkeep of the equipment. Although time-consuming, representational views from students, professors, and supporting technicians are critical to a successful remote experiment. At each stage, from design to implementation, all procedures and interfaces should be reviewed by a representative group. These evaluations not only improve the overall design, but also provide informal training.

1. The reset position may be the same as the start position.

2.1.3 Appearance

If video capabilities are available, any experiment that moves is a candidate for remote lab use. Another appearance criterion, mostly for demonstration purposes, is whether the equipment is unique or interesting to watch. From watching the experiment, the visual information should give the student a good grasp of the overall behavior. For a robot experiment the visual qualities are obvious; however, in the case of a DC motor, alterations such as notches on the rotor could give additional information. Although there are textual and graphical techniques to represent a live experiment, we find that video transmissions are effective invitations to use long-distance experimentation.

2.1.4 Our Choice of Experiments

We had several candidate experiments available to test the feasibility of SBBT: an inverted pendulum, an *x-y* table, and a robot arm. In all cases, the control code could be downloaded to each of the experiments, a mandatory logistical criterion. The pendulum experiment was ruled out because it required modifications to support the reset operation (a separate set of arms that close and set the pendulum upright). The *x-y* table was in active use by other students. The 3-DoF robot arm was the remaining choice. It was a good first choice because it contained no loose pieces, it was easily reset, and it would allow visually interesting demonstrations.

2.2 Lab Presence

In a real (local) lab, a student feels himself present when he opens the lab door and walks in. He sees the equipment, hears the voices of colleagues, and gets a general sense of the activity in the lab, just by opening the door. We wanted to find ways to replicate the laboratory presence for the remote student. SBBT provides audio and video, computer-transmittable facsimiles satisfying the senses of sight and sound.

We used vic [McCanne and Jacobson, 1994] and vat [Jacobson and McCanne, 1994] from Lawrence Berkeley Laboratories (LBL) for video and audio, respectively. We chose these applications because they were freely available via FTP in the beta release phase. We knew that video and audio applications require a large amount of network bandwidth, but both vic and vat provide an interface for network resource tuning. Due to the bandwidth concerns, we built modularly and could easily install less demanding presence applications such as straight-line drawings, or text-only feedback. However, bandwidth was not an insurmountable problem, and we found through our demonstrations that video and audio in particular invite interest.

2.3 Lab Environment Control

Since the remote student is not in the lab, he cannot manually turn on the power to the experiment, among other actions required while testing control code. The Lab Environment Control component enables remote use of the equipment. We determined a minimal set of commands by surveying students and professors in control engineering, electrical engineering, mechanical engineering, and computer science. See Appendix A for the survey results. The Lab Environment Control commands are very basic and all experiments need them (Table 1). Our demonstrations show that this minimal command set is sufficient to conduct long-distance experiments.

Table 1. Lab Environment Control Commands [Bohus, et. al., 1995]

Main Functions	**Explanation**
gosbbt	Start up the application for a work session.
quit	Release all SBBT resources.
stop	Immediate shutdown of controller motors.
reset	Put the experiment in a predefined, stable state.
download	Transfer control code or data to the target controller.
reboot	Turn off power to PC for several seconds, forcing a reboot.
compile	Compile and link the control code on the target machine.
run	Execute the most recently compiled control code.
getdata	Transfer experiment output data to the user.

2.4 Safety

Safety issues arise when building a system to control machines and processes from a remote location. First, SBBT must guarantee the safety of the people who are working locally in the lab. Second, SBBT must confirm that the network is up, and that the distant student is in timely contact with the equipment. Third, since this is a learning situation, the student needs tools to stop machines when he sees a mistake occur. We provide a mechanism for each of these situations.

We have implemented three different levels of safety for the most comprehensive coverage possible. There is a trade-off between guaranteeing each safety mechanism through redundancy, or developing a broader set of safety mechanisms. Redundancy is often considered a good strategy, but in this context it adds complexity. Additional complexity adds more modes of failure to the overall system [Leveson, 1984]. Our mechanical, software-controlled, and student-controlled safety features provide a responsible model for remote laboratories.

2.4.1 Safety Mats: Mechanical and Automatic

Since the remote student cannot be seen, some way of alerting and protecting students working locally is mandatory. Our technician, Steve Wilcox, reported that, in industry, a warning buzzer sounds when power to an apparatus is turned on. While the remote student is working, new local students may wander into the local lab. These students may be unaware of the remote student, because they were absent when the experiment power was engaged and the buzzer sounded. To protect the local students from sudden movement of an experiment, we placed a pressure-sensitive mat in front of the moving parts of the experiment. The mat is connected to the experiment power supply. If any weight is detected on the floor mat, the motors to the experiment are automatically turned off. The distant student must go through a reinitialization sequence to continue

working. The only way to foil this mechanism is to cut the wires. This safety feature is mechanical and automatic.

2.4.2 Network Heartbeats: Software Controlled and Automatic

Networks have latencies (time delays) and limited bandwidth. Furthermore, they sometimes go down. Initially, administrators were concerned about students working on moving systems from a remote location, since no one would be in the room to prevent any mishaps. We designed a heartbeat signal that constantly checks the network, to ensure that it is up, and that delays can be tolerated. If heartbeats are missing because delays are too long, or the network is unreliable, the power to the experiment motors is automatically turned off and SBBT sends an explanatory message to the student. The heartbeats are software controlled and automatic.

2.4.3 Panic Stop Button: Student Controlled and Manual

Assuming the student is actively using the experiment, and going through the normal routines to test his control code, he will be the first to react to many problems. In the local situation, if a student sees something undesired about to happen, he lunges for the power button, or some other trigger to stop the action. We provide the STOP button on the screen (Figure 5), for the remote users, which can be clicked on using the mouse. When the remote student clicks on the STOP, SBBT sends a message across the network to cut power to the motor, stopping it; this preserves the situation for student analysis. The STOP button is under manual student control.

2.5 Collaboration

Distance learners are potentially isolated collegially as well as physically. If computer-mediated distance learning is to minimize the effect of collegial isolation, SBBT must explicitly address theses issues. Straus and McGrath, [1994] emphasize the need for human considerations when designing computer-mediated experiences:

> There is a substantial lag between development and implementation of technological systems, on the one hand, and systematic research on the social and behavioral consequences of using such systems, on the other. Research in a variety of work contexts has shown the negative effects of designing technical systems without regard for the social systems in which they are embedded (e.g. Foushee, 1984; Trist & Bamforth, 1951).

SBBT supports social interactions with tools that make communication convenient and social protocols that encourage communication.

There are many network conference tools available for students to use to communicate with one another. We chose wb [Jacobson and McCanne, 1994], a shared whiteboard application developed at LBL. A student can draw or type on a portion of the computer screen, and everyone who is connected to that session sees the drawings and text. There are other wb application features to take advantage of, such as saving the communication sessions to review later. If latency, limited bandwidth, or software unavailability prevents students from using an audio tool, then the shared whiteboard, the UNIX utility *talk*, or even the telephone can be used to discuss homework problems or negotiate experiment use.

As often happens in a lab, there may be more demand for equipment than is available. Labs at universities handle this situation in various ways, such as using time limits and sign-up reservations systems. We chose the less formal negotiation approach to allocate resources. A social protocol provides a reason for students to communicate, that is, to negotiate the use of an experiment. This introduction may inspire continued technical discussion. SBBT provides the tools to compensate for the lack of traditional lab communications.

2.6 A Typical Scenario

The remote student needs an xterminal or workstation (Figure 5). The remote student starts the system by typing "gosbbt" on the command line. The system puts up all the applications (wb, vic, vat, Lab Environment Control) one after the other, as the student places the windows. While these window placements occur, the SBBT system checks to see if the experiment is currently in use. If the experiment is already in use, the remote student will get an abbreviated Lab Environment Control window. He can negotiate access to the equipment, or click on the QUIT button of the Lab Environment Control window and check again later. This is analogous to a student coming into the lab and, upon finding the experiment occupied, either discussing when he may use the experiment, or just looking and moving on.

If the experiment is not in use, the remote student has the full-functionality Lab Environment Control window. All other users will be locked out of control, so the integrity of the system is maintained. The network heartbeats have started, to ensure that the network is delivering an adequate level of service.

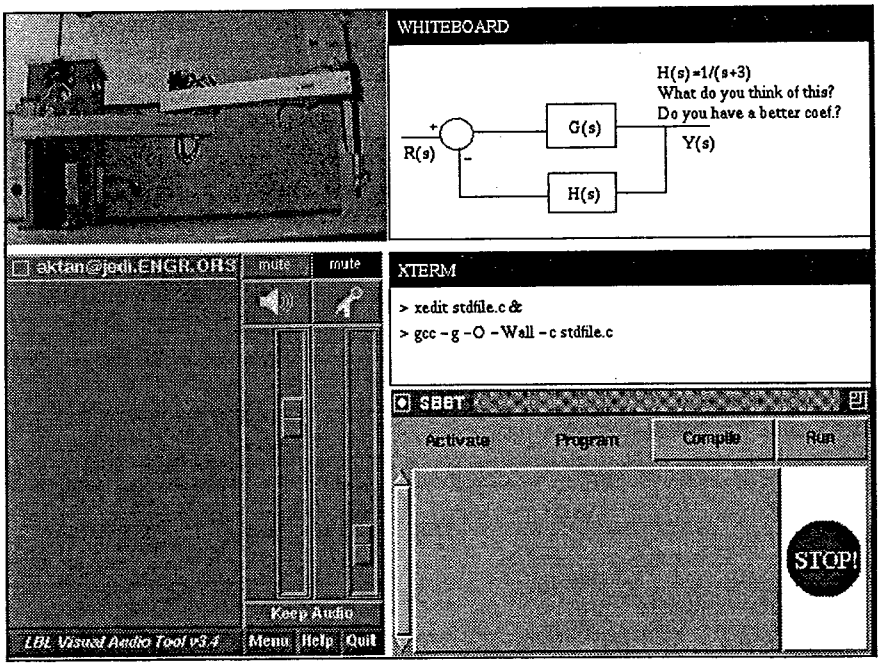

Figure 5. The Remote Lab User Interface for SBBT. [Bohus, et. al., 1995]

To start working, the student brings a model for the experiment he is working with, and a developed control program to try. He downloads his code to the experiment, where it is compiled and linked. The experiment apparatus is at an initial, stable starting position. At this point, the student has control of the experiment, his code is ready to run, the experiment is in a starting position, and the power is on. The student runs his code by clicking on RUN in the Lab Environment Control window and watches the experiment go through its paces.

Another student, in the lab or elsewhere, may observe the experiment and chat with the remote student via the communication tools. They may exchange ideas for ways to improve the performance of the controller. The remote student can consider these ideas, change his code accordingly, and go through another round of experimentation.

Finally, when satisfied that the control code is his best effort, or when receiving enough pressure from peers to relinquish control of the experiment, he can sign off. The experiment is put in a safe, neutral position, the motor is turned off, and all the remote lab software exits. The experiment becomes available to another student, remote or local, to use.

2.7 The SBBT Approach

Using the five-part architecture to design a remote laboratory offers numerous benefits. The architecture provides a conceptual foundation for developing remote laboratories. For example, a complete experience requires that each of the five parts must have some representation. However, each component can be provided in multiple ways, according to costs and needs. The lab facility can offer a selection of tools for each component. This way, students can chose tools they are comfortable with. If there is a problem with any particular software or hardware piece, students can temporarily move to another tool. New tools can be added, yet not displace existing applications. Flexibility allows students to develop their own environment. The SBBT approach accommodates diverse experiments and connectivity, allowing versatile hardware and software investments.

CHAPTER 3

ARCHITECTURE IMPLEMENTATION: LAB ENVIRONMENT CONTROL

To get our application working as soon as possible, we used existing software and hardware wherever there was a reasonable fit. We found ready-made solutions for each piece of the architecture except the Lab Environment Control. That component takes the place of the student in the lab. It has the specific constraints of safety, reliability and speed to maintain confidence in the SBBT system.

We conducted a safety survey of professors, technicians, graduate students, and a class of undergraduate control engineers (see Appendix A for the survey and results). With our safety requirements and the surveys, we identified a safety definition for our project, and safety mechanisms to enforce our policy.

In section 3.1, we define safety for our project. In section 3.2, we explain how real-time computing supports safety in our application. The hardware configuration is described in section 3.3; my colleagues Burcin Aktan and Steve Wilcox are responsible for the accomplishments in this section. Following this background information, we present a detailed description of the Lab Environment Control component. The last section of this chapter presents our timing results.

3.1 Safety

Ironically, software is inherently safe; it cannot affect people directly. It is the hardware that software controls that may pose hazards. Safety is a system-level concern [Leveson, 1986]. SBBT is a system with which students will learn the fundamentals, and they need to make mistakes in order to learn. We have the opposing goals of making a system flexible enough so students can learn, yet eliminating any dangers.

In a laboratory where students work on moving systems, many events are unanticipated. It is impossible to perform an exhaustive hazard analysis, since all the possible events cannot be known in advance. Even if we could list all hazards and find mechanisms to avoid them, research shows that trying to eliminate all risks usually just displaces them [Leveson, 1986]. Therefore, the first line of responsibility is in the student's hands, as if she were in the lab supervising the equipment in person. People are better than machines at judging what to do in an emergency. As part of the training a lab experience offers, we want students to learn good safety techniques and consider safety at all times.

Our definition of safety for this project is to protect the students working in the local lab from physical injury, protect equipment from damage, and prevent any uncontrolled periods. However, to encourage the natural learning process we allowed harmless mistakes.

All of our safety features, when activated, go to a fail-safe mode. From our high-level hazard analysis, we found that when power is cut to the robot motor, there is very little drift due to momentum. Also there is no locking break; if the robot arm presses against something, it relaxes when power is turned off. In all safety events, SBBT turns off the power, which puts the equipment in a safe, neutral state. The apparatus is left as is, so the student can analyze the situation.

3.2 Real-Time Computing

Real-time computing provides the methodology for ensuring safety. Real-time computing is *predictable* computing, not necessarily *fast* computing [Shin and Ramanathan, 1994]. However, since most computers are optimized for the general case, special policies must be defined to satisfy real-time demands. General computers try to give fast overall service to all of the jobs currently running. Fast service for all jobs cannot be guaranteed, because demand on computing resources cannot be predicted. In our case, the goal is to assure the distant user that she has tight control of the distant experiment by continually checking the network. The real-time strategy we relied on was round-trip

network signals called heartbeats which complete a circuit between the lab and remote student at regular intervals. The amount of time a heartbeat requires to traverse the circuit measures the network latency. The device that supplies power to the experiment expects regular heartbeats assuring system integrity. SBBT will verify that the network latency is adequate, or it will turn the experiment off.

A safety scenario might involve a student watching the experiment via video, running her control code and witnessing the experiment going through its paces. Upon seeing an actual or potential mishap, she can click on the STOP button of the Lab Environment Control window (Figure 5). The Lab Environment Control window and the video window are separate applications. Nevertheless, if the student clicks on the STOP button, she needs to see the experiment stop in the video window. The relationship between the heartbeats and the critical panic STOP order verifies that the student, *seeing* trouble, can *stop* the experiment within the specified timing constraint.

To determine the heartbeat rate, we timed actions in the lab and examined answers from a survey of control engineering students. Appendix A contains the survey and responses. An initial goal was a 3-4 second latency or less. We measured the time required locally to click on the STOP button and hear the motor stop. This action averaged roughly 0.5 seconds. During SBBT demonstrations in different cities, the response time seemed instantaneous.

The critical time slice to monitor our network is four seconds. Dividing this interval in half for the heartbeat periods lets one heartbeat fail and still leaves time to execute another heartbeat check within the critical time slice. For data transport, we chose User Data Protocol (UDP), a connectionless, best-effort transport level protocol. Recall that networks can be volatile. Dividing the critical time slice in half ignores some of the characteristics of our network infrastructure. We decided to divide the critical time-slice into thirds, thus allowing for one lost UDP packet due to the best-effort network transport, one actual lost packet, and still providing time to get one legitimate heartbeat through before passing the threshold and shutting SBBT down (Figure 6). The lost heartbeat threshold and the heartbeat interval are software adjustable (at compilation time), to accommodate characteristics of any network or experiment.

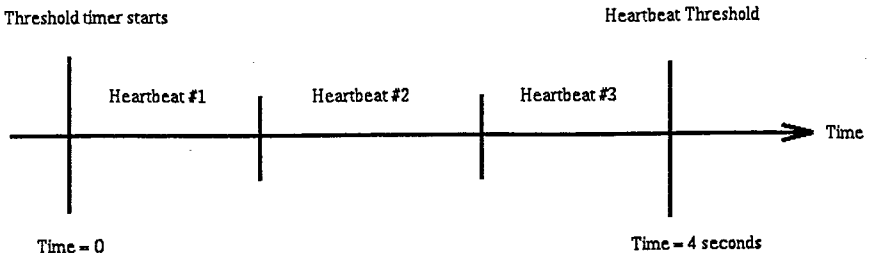

Figure 6. SBBT Timeline Schematic.

SBBT uses local enforcement of timing constraints to tolerate unpredictable network delays safely. The network delay must be within the established threshold to ensure that the remote student is indeed in control. The remote student has complete responsibility for the equipment, as long as the network latency is acceptable.

3.3 Hardware Configuration

The hardware configuration directly supports the five-part user interface specification. Two important hardware functions, besides the control experiment itself, are network connectivity and basic power access (Figure 7). We discuss the hardware requirements and describe the hardware configuration in this section.

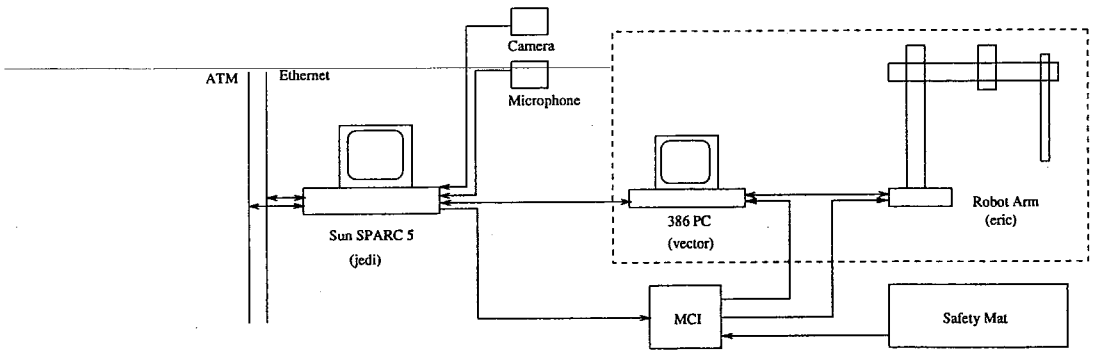

Figure 7. SBBT Hardware Configuration. [Bohus, et. al. 1995]

3.3.1 Hardware Requirements

As we specified the functionality for the remote student, we could identify our hardware requirements:

- construction of the control engineering experiment
- availability of the experiment 24 hours a day
- ability to download/upload code/data independent of local help
- support for audio, video, and collaboration applications
- support for safety at every phase of the experiment

3.3.2 Hardware Description

The control engineering experiment we used to demonstrate SBBT is a 3-DoF (Degrees of Freedom) robot arm, named eric (Figure 5). It has a 4.8 ampere stepping motor for movement. Eric also has a disabled pneumatic (vacuum) system to manipulate objects. Control signals to the robot come from the PC, and power is supplied via the Motor Control Interface (MCI).

The PC is a 386 MHz machine, running DOS 6.1, named vector. It receives signals from the workstation via a serial cable and a full duplex null modem connector. Control programs are loaded from the workstation to the PC through a keyboard emulator. Feedback from the experiment transmits back to the student over the same line. The PC receives power via the MCI. By using the MCI to power the PC, we enable the long distance learner to reboot the PC easily. My colleague, Burcin Aktan, completed the cabling between the workstation and PC, and wrote the routines for program, data, and command transfer.

The MCI was custom built by Steve Wilcox to provide basic power control. It routes power to the experiment, the safety mat, and the PC. It receives signals from the workstation, and from the pressure-sensitive safety mat, placed in front of the experiment's moving parts. The MCI makes the experiment available 24 hours a day since the remote student can turn the experiment on independently. The MCI sounds a warning

buzzer when powered on. It features a manual stop button and sensor light indicators which indicate state (Figure 8). The MCI supports all the safety features through basic power access.

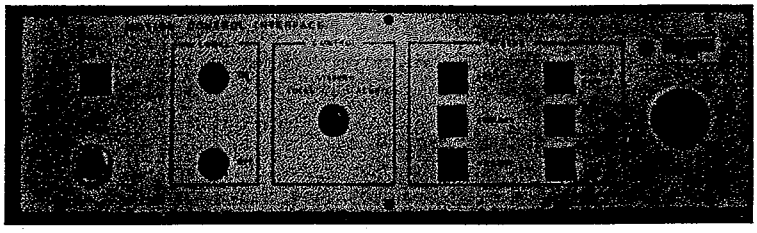

Figure 8. Motion Control Interface.

The workstation is a Sun Sparc 5 running Solaris 2.4, named jedi. It provides the hardware support to digitize and transmit video camera and audio microphone data. The audio and video require add-on boards plugged into the workstation chassis. The workstation is dual ported for ATM (Asynchronous Transfer Mode) and ethernet for network communications. The OSU NERO staff performed the network connection cabling for the workstation. To use the LBL conference tools (wb, vic, and vat) for group dialog, some multicast mechanism must be enabled for the network. The workstation connects to the MCI (RS232/RS485) and the PC (RS232) with serial cable. The workstation provides connectivity, audio, and video of the lab to the remote student.

3.4 Lab Environment Control

Abstracting the experiment into a resource reveals the client/server paradigm as a natural fit. The client/server paradigm is defined by Comer [1988] as a "pattern of interaction among cooperative applications." Lab Environment Control is a set of client programs, and a set of server programs. A server program runs continuously on the local laboratory machine, waiting for requests to service. The client program runs at remote

sites only when invoked by students who want to use SBBT. We produced two releases of software: the first, an iterative server handling only sequential requests; the second, a concurrent server that handles multiple requests and the network heartbeats. By going through two development phases, we demonstrated SBBT earlier and incorporated more user-directed improvements in the second release.

In this section, we detail our main software contribution. First, we describe both of the client/server architectures. Next, we explain our communications protocol. In subsection 3.4.4, we describe how we award access to the experiment. The remaining subsections present the logic for each piece of the client/server architecture using finite state machines.

3.4.1 Early Prototype: Iterative Client/Server

The first SBBT prototype was implemented as a connectionless iterative client/server (Figure 9). The client program runs at the remote location. It executes only when the student clicks on a command in the Lab Environment Control window. When a student clicks on a command request, the client program is invoked with parameters to indicate the service desired. The client program sends a request packet out on the Internet addressed to the server. The client then waits for a reply from the server. It returns the reply information to the user and then exits. The client program only runs when it has a control command request.

The iterative server fulfills one request at a time. Most operating systems queue a small number of requests for iterative servers, which saves programming effort. The iterative server continuously listens for requests, it performs the requested action, sends the results back to the requester, and returns to listening.

This type of server is good for short requests that arrive sporadically. It worked well to implement a prototype quickly. But, ultimately, it was not appropriate for SBBT, which issues command requests that can take some time to complete. An iterative server was not an adequate long-term solution.

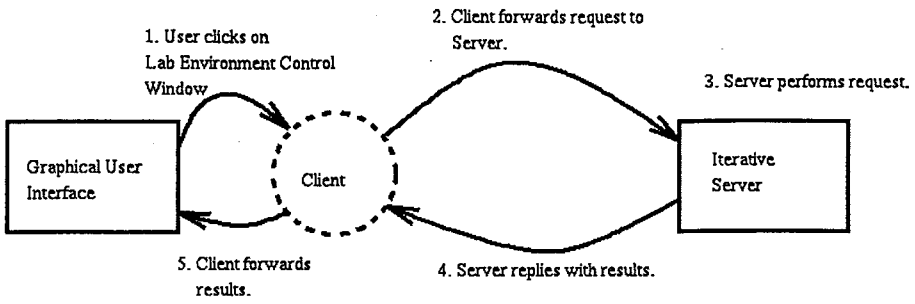

Figure 9. Iterative Client/Server.

3.4.2 Concurrent Client/Server

In the first SBBT release, the client was activated only to issue a request and wait for the response. Now, we needed a client that was continuously active to receive and answer heartbeats and forward Lab Environment Control commands. We renamed the iterative client the Pony Express (PE). It is activated when the student clicks any selection in the Lab Environment Control window. The PE carries the student request to the new SBBT client, which is always running while the student is using the experiment. The new, more complex concurrent client must forward requests to the server, relay feedback to the student, and answer network heartbeats (Figure 10).

When requests take a long time to fill, using a concurrent server is a better strategy than using an iterative server. In a concurrent server configuration, there is a *parent* server, which is also called the *listening* server. The parent server in SBBT is the SBBT Lab Manager. The Lab Manager tracks the state of the experiment resource, that is, whether it is in use or not. When a parent server detects a request that will take some time to service, the parent creates a *child* server. The child server assumes responsibility for completing the service, freeing the parent server to listen for more requests. The child server acts as the SBBT Session Manager. Once a student receives control of the experiment, the Session Manager handles all further requests, and monitors the safety heartbeats.

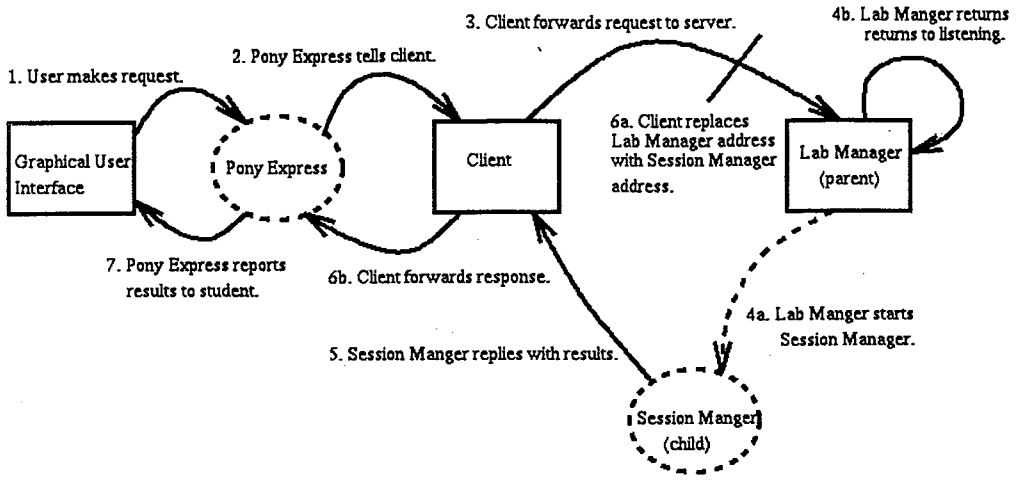

Figure 10. Concurrent Server.

3.4.3 Lab Environment Control Communications

Once the client/server design decision is made, the next step is to define the protocol for communication between the programs. The communication tasks are to: convey user directives, relay back the results, manage the heartbeat signals, and update the parent/child server states. These tasks exhibit different communication characteristics. For example, user requests cannot be predicted; they are sporadic. The heartbeat signals, however are generated at regular intervals; they are periodic. The protocol needs to accommodate a variety of communication services.

We chose User Data Protocol (UDP) for several connectivity benefits. UDP is best described as a connectionless, best-effort, transport-level protocol. It is a low-level protocol consuming little processing overhead. The communication traffic for the Lab Environment Control is critical and benefits from little overhead processing. Our network is also quite reliable, and the end-to-end reliability of stream-oriented transport was not necessary. After a year of demonstrations, we believe there has only been one lost communications packet. Also, we designed SBBT requests to fit in one UDP packet and thus they are not stream-oriented data. Stream-oriented transport is designed to transmit

multiple data packets. Finally, we have an Asynchronous Transfer Mode (ATM) network available to us, and UDP packet characteristics compare similarly to ATM cells. Current ethernet-based networks transmit data at 10 mbps (megabits per second). New experimental protocols, such as ATM, coupled with faster and more reliable media like optical fiber, can push transmission rates to 155 mbps.

We considered a release of SBBT that ran directly over ATM, as opposed to indirectly on top of the Internet Protocol (IP). An initial evaluation reveals that running directly on raw ATM will not increase performance commensurate with the programming effort. Since we use the NERO network, which is based on ATM and fiber, we already benefit indirectly. In keeping with our development strategies, we remain compatible with possible ATM development.

All the client/server communications complete a circuit. In the case of control commands (see Figure 11(a)), the student initiates a command by clicking on an item in the Lab Environment Control window. This invokes the PE with a parameter indicating the student's choice. The PE forwards this request to the client and waits for reply information. The client sends the request to the server, but does not wait for a reply. Instead, the continuously-running client waits for any traffic to process. When the server completes the request, it formats and sends a reply to the client. The client forwards this information to the waiting PE. The PE completes the circuit by handing the results to the user interface, where it is displayed for the student.

Simultaneous with any control command sequence, the server initiates the network heartbeats to verify the network is up (see Figure 11(b)). This circuit runs between the client and server only. This is the long-distance link more likely to experience delays or downtime. During each critical interval, three heartbeats are sent. The critical time interval can be adjusted to suit the characteristics of any network.

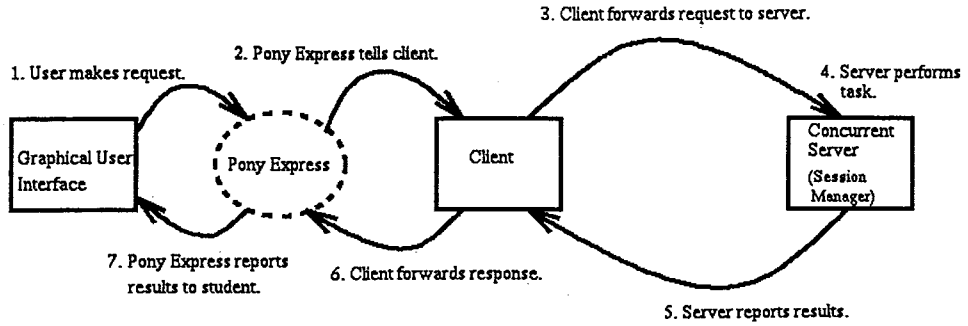

Figure 11(a) Communications for Lab Environment Control Commands.

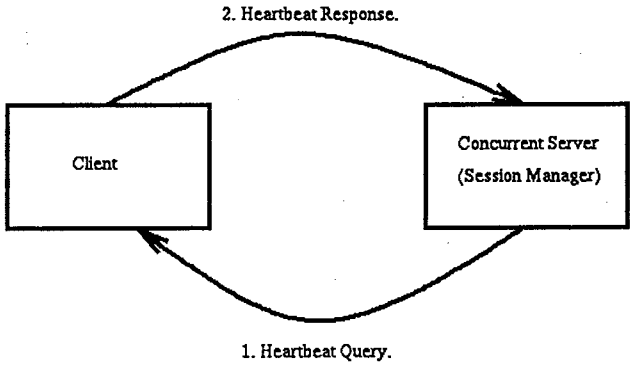

Figure 11(b). Simultaneous Communications for Lab Environment Heartbeat.

3.4.4 Who Gets SBBT Control?

Naturally in a lab, one person controls an experiment at a time. We copy the physical laboratory model by bringing up SBBT for anyone who invokes it, just as anyone walking into the lab stands in the room whether the experiment they want is occupied or not. In our current implementation, the remote student has audio, video, and shared whiteboard connections to communicate with others, whether they receive control of the experiment or not.

For safety and security, we want only one student to control the experiment resource. This policy requires two different Lab Environment Control user interfaces. My colleague Burcin Aktan wrote the graphical user interfaces. One user interface has all the facilities listed in Table 1. This is depicted in Figure 5. The other Lab Environment Control interface has only a QUIT button. SBBT users do not know whether they will receive control until they see which window comes up. All SBBT sessions for the same experiment are connected via video, audio, and whiteboard, providing an introduction among students.

3.4.5 Finite State Machine (FSM) Descriptions for Concurrent Client/Server

A finite state machine is a formal diagram showing the logic and possible states of a program. The states are depicted as spheres, and the arrows show what input causes a transition, which changes the state of the program [Hopcroft and Ullman, 1979]. In the following subsections we present a finite state machine for the Lab Manager, the Session Manager, and the SBBT Client.

3.4.5.1 Lab Manager FSM Description

The Lab Manager has two states: Unallocated (the start state), when no control is allocated to any user, and Control Allocated, when a single user does receive control (Figure 12). A request for the experiment moves the Lab Manager from the Unallocated state to the Control Allocated state. All subsequent requests for the experiment are denied and the Lab Manager remains in the Control Allocated State. When the student indicates she is ready to release the experiment, the Lab Manager acknowledges this request and returns to the Unallocated state.

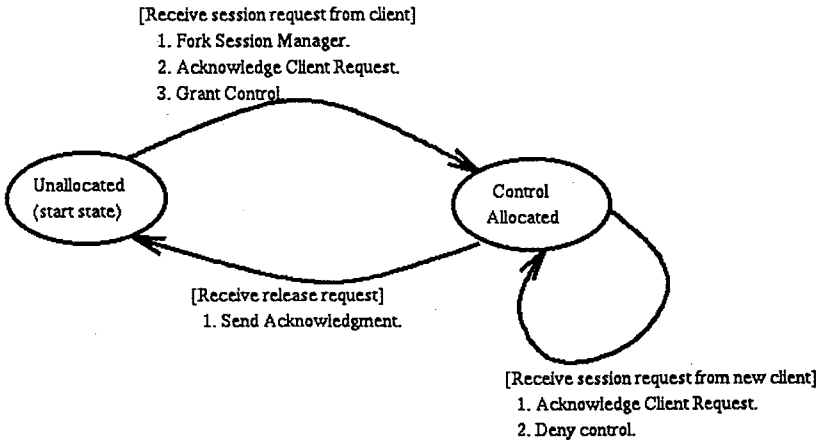

Figure 12. Finite State Machine for SBBT Lab Manager.

3.4.5.2 Session Manager FSM Description

The Session Manager fulfills all of the remote users' requests. Most time is spent in the Manage and Await states (Figure 13). Once control is allocated to a distant learner, the state changes to Manage. In the Manage State, special signals remind the Session Manager to send a network heartbeat to the client, verifying the connection. Once a heartbeat is sent, the state changes to Await, until the return heartbeat is received. Each heartbeat sequence cycles through the Manage and Await states. Lab Control Environment requests are accepted and serviced only in the Manage and Await states. The finite state machine shows that tracking the heartbeat sequence is the driving force.

If too many heartbeats are missed (the network malfunctions), the session manager, by policy, forces a shutdown of the experiment and a release of the experiment. This moves the Session Manager to the Wait to Die State. Once it receives exit confirmation from the Lab Manager, the Session Manager exits. SBBT attempts to inform the distant student. However, it does not wait for an acknowledgment since we have indications that the network is down or clogged.

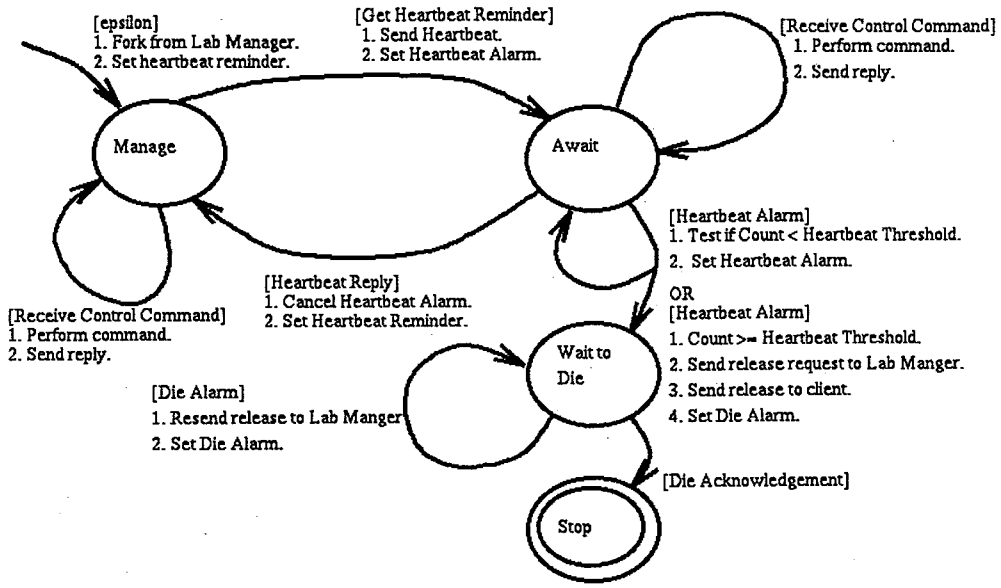

Figure 13. Finite State Machine for SBBT Session Manager.

3.4.5.3 Client FSM Description

Two independent paths lead through the Client finite state machine (Figure 14). As soon as the student invokes SBBT, the SBBT client requests control of the experiment resource from the Lab Manager. Experiment Control is awarded or denied, according to whether the experiment is already in use or not. In either case, the SBBT system automatically starts all of the constituent applications and connects to the SBBT session for that particular experiment.

If the resource request is denied, the student will be an observer. Since the SBBT system does not track observers, when the Quit is sent by the observing student, all the components simply exit. If the experiment is available, the remote student's request is granted, and the client spends most time in two states: Conduct Experiment, and Process Heartbeat. When the student who has control is ready to leave, clicking the mouse button on Quit makes the client ask the Session Manager for release confirmation. Upon receipt, the client forwards the information to the PE and exits. Once the PE presents the information, it also exits.

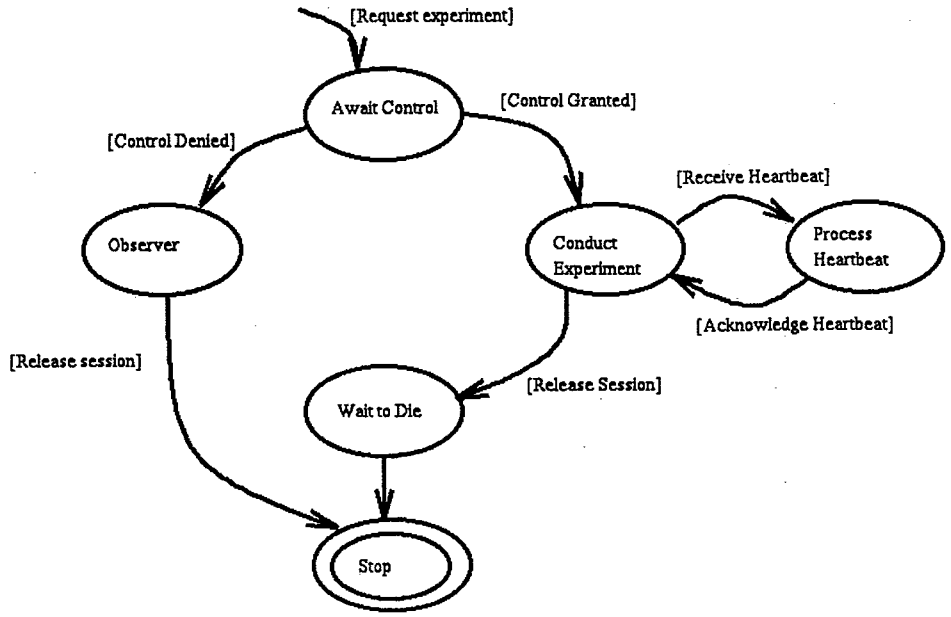

Figure 14. Finite State Machine for SBBT Client.

3.5 Timing Results

We timed the Lab Environment Control communications exclusively on Sun Sparc 5s, running Solaris Operating System 4.0. The Sun workstations are all connected to the NERO network. Our tests were conducted on one night, during light network traffic hours. We used the high resolution real-time clock and time routines (gethrtime()). The graphical user interface processing was bypassed for these time trials.

The term latency indicates the time it takes for a communications packet to traverse the network. We used round-trip times, which includes operating system processing time as well as the time on the network. The machine configurations we used for timing were host-to-self, host-to-host at the same site, and host-to-host at different sites. Table 2 reports our timing data. Note that it takes longer for a host-to-self time than for two-host traffic at the same site. This reflects that, for the host-to-self communications, the same processor handles both sending and receiving processing overhead. In the two-host communications, each host completes half of the processing. To get a rough idea of the

network latency, we subtract the host-to-host (same site) from the host-to-host (different sites) and find that a Lab Environment Control packet takes about 47 milliseconds for a round trip between Corvallis, Oregon and Portland, Oregon. Network latency of the Lab Environment Control is not a serious concern for SBBT.

Table 2. Timing results for Lab Environment Control communications reported in nanoseconds.

Configuration	**Average Time**	**Longest Time**	**Shortest Time**	**Difference**
Host to Self (jedi.engr.orst.edu)	3,241,650 ns	4,121,500 ns	3,127,500 ns	994,000 ns
Host to Host (jedi.engr.orst.edu) (zero.engr.orst.edu)	3,053,050 ns	3,070,000 ns	3,034,500 ns	35,500 ns
Host to Host (jedi.engr.orst.edu) (sampo.cs.pdx.edu)	50,454,800 ns	418,278,000 ns	8,968,000 ns	409,310,000 ns

CHAPTER 4

ENGINEERING STRATEGIES

The SBBT project was a multi-disciplinary effort. That is, the people involved range over academic departments, ranks, universities, and levels of investment. When different disciplines gather, they must allow time to establish common definitions. Such projects are ambitious due to the complex human factors involved.

For example, during review of the control commands for the Lab Environment Control, we discovered very different definitions of the term *crash*. For computer scientists, this term describes a program that has worked its way to a non-operative state; it is stuck. For engineers who work in a more physical and applied world, a *crash* occurs when two objects physically smash together, such as a robot arm hitting the wall. It took several meetings to discover this important discrepancy. Communication among the team participants building and using SBBT was crucial to the project's success. In this chapter, we highlight three areas that require correct communications: 1) scheduling decisions, 2) interface issues and 3) quality assurance.

4.1 Scheduling Decisions

In a large project, scheduling is crucial to speedy project completion. Some activities can be planned ahead, but some scheduling is forced by outside influences. For instance, delivery of the workstation and several parts for the MCI were delayed. That, in turn, delayed communications cabling. Meanwhile, our meetings with administrators emphasized the need to guarantee safety. We used the extra time to further develop the specification and safety analysis.

To meet our project timeline, we accomplished our tasks as follows:

- While we waited for the delivery of the workstation, we researched safety and conducted a wider review of our specification.

- Once we received the workstation, we could start on the communications cabling. We wanted to test the communications quickly, so we used operating-system--level utilities. This turned out to be a very flexible method, and became a key to ensuring portability.
- While we waited for the outside communications to be set up, we tested the workstation to PC communications and cabling.
- To test quickly, we used a very simple iterative server. We provided demonstrations within just two months of receiving the workstation by keeping initial development efforts simple.
- We deliberately scheduled early demonstrations which provided an excellent way to focus the team. It clearly fostered cross-departmental commitment and resources. The emotional payback of doing demonstrations also gave the team a feeling of success.
- During the first few demonstrations we masked the fact that the custom-built MCI was not ready. We compensated by scheduling another student in the physical lab to turn equipment on for demonstrations. The remote lab student was now joined by the local lab assistant and they established a rapport. This highlighted the important collaboration aspects, and we continued the practice.

Delays and roadblocks are inevitable in a multi-discipline project. We tried to adjust the schedule to accommodate changes and use our time efficiently. We deferred all solved problems and concentrated only on the unsolved ones. By giving early demonstrations, we met many interested people and could use their observations to improve SBBT.

4.2 Interface Issues

Our interface issues fell into four categories: 1) LBL tool start-up, 2) graphical user-interface and the Lab Environment Control coupling, 3) communication between the client and server programs, and 4) the MCI and the Lab Environment Control commands. We resolved these issues so that all interfaces worked smoothly. The following tactics ensured this success:

- Conducting many interface reviews with the people responsible for each side of the interface. Each review defined the interface more precisely.
- Conducting reviews with many different kinds of people who were not directly involved in implementation. Often others would point out something we had not considered.
- Conducting a careful safety analysis. Looking at the interfaces from another point of view often revealed any misunderstandings.

Spending time together, and reviewing the various interface specifications gave us the opportunity to clarify our understandings and avoid costly mistakes.

4.3 Quality Assurance

Since we often had the opportunity to do ad-hoc demonstrations, it was important that our system be stable and working at all times. We required a solid quality assurance process. The quality assurance plan specified the following steps: 1) integrate one new feature at time 2) test it locally, and 3) test it from a remote location. This method pinpointed problems very quickly. For instance, when we integrated the PE and new client, we first tested them against the old server. The protocol was exactly the same and so only the new part was unknown. Locally, the testing succeeded, but the next step, testing from another location, failed. It took less than five minutes to find the invalid program variable, correct it and recompile the whole system. Had we integrated the new server, along with the PE and new client, many other bug-solving paths would have been necessary to isolate the cause.

We also had the code developer turn over tested code to the integrator. The integrator usually completed one testing session independently, followed by one testing session with the developer. The integrator's testing provided both objective feedback and the perspective of a student user. Using this quality assurance method, we were always ready for demonstrations.

CHAPTER 5

CONCLUSION

This thesis describes the process we used to produce our prototype for remote laboratory access. We considered local laboratory use and distilled the lab experience into five components: the experiment, lab environment control, lab presence, safety, and collaboration. We produced a multimedia application, named Second Best to Being There (SBBT), to test our distance learning paradigm for remote labs. We examined each component and selected a computer transmittable facsimile in accordance with our strategies, which were: striving for modularity, achieving transparency to the user, using existing hardware and software, encouraging collaboration, ensuring safety, and using real-time computing. Using the Internet for the communication infrastructure also allows us to participate in defining how the National Information Infrastructure (NII) vision will play out. We paid special attention to safety and computer-mediated communication. Our prototype is a useful multimedia application for remote laboratory access.

5.1 Five Demonstrations of Success

We demonstrated SBBT at different locations throughout our development. Each audience stimulated useful suggestions that we could incorporate into the next software release. Our confidence in our application grew with each opportunity to exercise it. Each of SBBT's demonstrations proved successful.

- Modern Communications Center Dedication at LaSells Auditorium, OSU Campus, Corvallis, Oregon, March 26, 1995. *SBBT's debut proved that it worked between buildings on the OSU campus.*
- Portland State University Electrical Engineering Colloquium, PSU Campus, Portland, Oregon, April 14th, 1995. *Part of the colloquium demonstrated SBBT from 85 miles away.*

5.3 Future Work

Since SBBT is a prototype, many ideas will surface with use. We note here possibilities of product-level additions, more support for collaboration, additional feedback data from the experiment, and using SBBT on more experiments. The product-level development that seems appropriate now are installation routines, a guidebook for lab assistants, and increased security. More collaboration support might include tracking observers, more video cameras, an ability to re-position cameras, and establishing distance collaboration etiquette. Additional feedback data would encompass more sensors on the experiment, a way to organize the measurement data, and tools to visualize the data. As SBBT is incorporated into different labs and experiments, the experiences of a class of control engineers will give priority to enhancements.

5.4 Future Applications

We believe our approach to defining a remote lab experience can be used on general distance applications. By abstracting the essence of what we want, we can provide a natural and comfortable setting for it. People are ready to experience more from home, and the government is encouraging remote exploration. It is clear that the applications developed now will define our future network experiences. We have shown that using the computer need not be a sterile experience. By dividing the desired experience into components and satisfying each component separately, we ensure each part has a rewarding element. By using practical and theoretic strategies over all implementation choices, we came very close to meeting all of our ideals.

BIBLIOGRAPHY

[1] Bhatia, Praveen, and Masaru Uchiyama. 1994. Shared Intelligence for Telerobots with Time Delay: Theory and Human Interface with Local Intelligence, *Proceedings of 1994 IEEE International Conference on Robotics and Automation*, 1994, Volume 2, pp. 1441-1448.

[2] Bekey, George, Steven Gentner, Rosemary Morris, Carl Sutter, and Jeff Wiegley, The Tele-garden, http://www.usc.edu/dept/garden/, 1995.

[3] Bohus, Carisa, Burcin Aktan, Molly H. Shor, and Lawrence A. Crowl. 1995. Running Control Engineering Experiments over the Internet, Department of Computer Science Oregon State University, Corvallis, Oregon 97331 USA. Technical Report 95-60-07. Also to appear in *IFAC World Congress Proceedings*, San Francisco, CA, July 1-5, 1996.

[4] Bradley, Gunilla, Peter Holm, Marcia Steere, and Gorel Stromqvist. 1993. Psychosocial Communication and Computerization, *Computers in Human Behavior*, 1993, Volume 9, pp. 157-169.

[5] Chan, Tan Fung, and Rajiv V. Dubey. 1994. Design and Experimental Studies of a Generalized Bilateral Controller for a Teleoperator System with a Six DoF master and a Seven DoF Slave, *Proceedings of IEEE International Conference on Robotics and Automation*, Volume 3, pp. 2612-2818.

[6] Cochrane, Peter. 1994. Dark Fiber will Transform Telecommunications, *IEEE Spectrum*, January, 1994, pp. 24-25.

[7] Comer, Douglas. 1988. *Internetworking with TCP/IP, Principles, Protocols, and Architecture*, Prentice-Hall, Inc. 1988.

[8] Graves, Sean, Larry Ciscon, and J.D. Wise. 1992. A Modular Software System for Distributed Telerobotics, *Proceedings of the 1992 IEEE International Conference on Robotics and Automation*, Nice, France, May 1992, pp. 2783-2785.

[9] Hopcroft, John E., and Jeffrey D. Ullman. 1979. *Introduction to Automata Theory, Languages, and Computation*, Addison-Wesley Publishing Company, Inc., 1979.

[10] Klein, Mark, John P. Lehoczky, and Ragunathan Rajkumar. 1994. Rate-Monotonic Analysis for Real-Time Industrial Computing, *IEEE Computer*, January 1994, Volume 27, pp. 24-33.

[11] Kondraske, George V., Richard A. Volz, Don H. Johnson, Delbert Tesar, Jeffrey C. Trinkle, and Charles R. Price. 1993. Network-Based Infrastructure for Distributed Remote Operations and Robotics Research, *IEEE Transactions on Robotics and Automation*, Volume 9, Number 5, October 1993, pp. 702-704.

[12] Lee, Sukhan, and Hahk Sung Lee. 1993. Modeling, Design, and Evaluation of Advanced Teleoperator Control Systems with Short Time Delay. *IEEE Transactions of Robotics and Automation*, Volume 9, Number 5, October 1993, pp. 607-623.

[13] Leveson, Nancy G. 1984. Software Safety in Computer-Controlled Systems, *IEEE Computer*, February 1984, pp. 48-55.

[14] Leveson, Nancy G. 1986. Software Safety: Why, What, and How, *Computing Surveys*, Volume 18, Number 2, June 1986, pp. 125-163.

[15] Lumelsky, Vladimir. 1991. On Human Perfomance in Telerobotics, *IEEE Transactions on Systems Man, and Cybernetics*, Volume 21, Number 5, September/October 1991, pp. 971-982.

[16] Miner, Nadine E., and Sharon A. Stansfield. 1994. An Interactive Virtual Reality Simulation System for Robot Control and Operator Training, *Proceedings of 1994 IEEE International Conference on Robotics and Automation*, Volume 2, pp. 1428-1435.

[17] NERO. 1993. Network for Education and Research in Oregon, version 1.2 of the *Guidelines for NERO Projects*.

[18] Papadopoulos, Christos, and Gurudatta M. Parulkar. 1993. Experimental Evaluation of SUNOS IPC and TCP/IP Protocol Implementation, *IEEE/ ACM Transactions on Networking*, Volume 1, Number 2, April 1993, pp. 199-216.

[19] Shin, Kang G. and Parameswaran Ramanathan. 1994. Real-Time Computing: A New Discipline of Computer Science and Engineering, *Proceedings of the IEEE*, Volume 82, Number 1, January 1994, pp. 6-23.

[20] Stark, Lawrence, Won-Soo Kim, Frank Tendick, Blake Hannaford, Stephen Ellis, Mark Denome, Mary Duffy, Tim Hayes, Ted Jordan, Mark Lawton, Tim Mills, Robert Peterson, Kathleen Sanders, Mitch Tyler, and Steven Van Dyke. 1987. Telerobotics: Display, Control, and Communication Problems, *IEEE Journal of Robotics and Automation*, Volume RA-3, Number 1, February 1987, pp. 67-75.

[21] Straus, Susan G. and Joseph E. McGrath. 1994. Does the Medium Matter? The Interaction of Task Type and Technology on Group Perfomance and Member Reactions, *Journal of Applied Psychology*, 1994, Volume 79, Number 1, pp. 87-97.

[22] Tanenbaum, Andrew S. 1989. *Computer Networks*, Second Edition, Prentice-Hall, Inc., 1989.

[23] Taylor, Ken and James Trevelyan. 1995. A Telerobot on the World Wide Web, *Proceedings of the 1995 National Conference of the Australian Robot Association*. http://telerobot.mech.uwa.edu.au.

[24] Woolf, Beverly Park, and Wendy Hall. 1995. Multimedia Pedagogues, *IEEE Computer*, May 1995, pp. 74-80.

APPENDIX

CONTROL ENGINEERING SURVEY AND RESULTS

Table A-1. Survey Results for Hazards in a Laboratory. [Bohus, et. al. 1995]

Hazard	Possible Causes
Hazard due to students learning	The software or controller (hence robot) may not do what students think has been programmed.
	The hardware or software initialization or calibration may be off, resulting in unintended robot action.
	The gains or controller may be inappropriate, resulting in instability of the closed-loop system, hence undesirable robot behavior.
Hardware setup incomplete or incorrect	The hardware connection may be loose, missing, or improperly prepared, resulting in incorrect or noisy signals from or to robot, resulting unintended controller or robot action.
	The robot may not be left in inoperable condition by the on-site users, causing problems in the first hazard category or rendering the robot inaccessible to distance users.
Observers in the way	A person or object my obstruct the robot in the lab, resulting in an accident during normal robot action.
Environment hazards	A person tripping over cables may pull down the attached equipment.
Unsafe experiment start-up	Someone may turn on power before all switches and mechanical parts are on the safe power-on positions, disrupting safe power-on sequence.

Table A-2. Safety Precautions [Bohus, et. al. 1995]

Electrical	**Mechanical**
1. Check all switches for default positions before main power is turned on.	1. Verify no person can obstruct any moving parts.
2. Verify all electrical connections are intact, including cards.	2. Verify no wires or cables are in the way of human or machine movement.
3. Verify that no electrical conducting media (wires, people) are touch any live wire.	3. Verify no piece of equipment can obstruct other equipment.
4. Check power-on sequence.	
5. Verify start-up transients do not damage equipment.	

Here is a copy of the survey that was administered to a senior control engineering class in the Electrical and Computer Engineering department at OSU in the Spring of 1994.

Working in a Control Engineering Lab Spring Term 1994

ANONYMOUS SURVEY

Good Day,

Over the summer there will be an experiment to use the control engineering lab from a remote location. To make the remote lab experience useful for the remote student, we need to provide some of the things that come naturally from working locally in a lab. Please answer the questions below about your lab experience which will help create a decent environment for future remote students.

Questions

1. Do you generally acknowledge other people in the lab when you enter for the first time?

2. In a typical hour in the lab, how much time do you spend talking to classmates?

3. If using shared equipment, how do you schedule time on the machines between groups and/or individuals?

4. How long is a typical lab work session?

5. What is the time range of a work session? (The longest, the shortest)

6. How often do you use the lab during odd hours in a term? (before 8am, after 6pm, weekends)

7. List safety measures you follow regularly.

8. List safety measures you know of, but don't do consistently.

9. For the experiment you're working on now, how many views angles do you use to see whats going on? Please list the experiment, # of views, and a general description of the view. (e.g., from both sides)

10. How many seconds does it take you to stop an experiment?

Survey Results

There were 10 responses. We list the various replies to each question.

1. **Do you generally acknowledge other people in the lab when you enter for the first time?**

Yes: 8 respondents. One respondent did not reply, and one respondent acknowledges people if s/he knows them.

2. **In a typical hour in the lab, how much time do you spend talking to classmates?**

"*10 minutes*"

"*usually with lab partners, because other classmates are in the lab.*"

"*At least 1/2 to 3/4 of the time we are talking and working together.*"

"*10 minutes*"

"*Talking is constant as communication is vital during sessions.*"

"*45 minutes*"

"*Plenty- we talk about how we are doing our lab.*"

"*50%, discussion of possible solutions and trials.*"

"*30*"

3. **If using shared equipment, how do you schedule time on the machines between groups and/or individuals?**

"*By writing names and time on the board.*"

"*There are no problems in using shared equipment. We come whatever time in the day.*"

"*Sometimes we talk to each other in class to coordinate schedules. Sometimes we make tentative schedules on the chalk board in lab.*"

"*Through the TA ahead of time.*"

"*This has never been a problem. 1) There are only a couple teams using the same equipment. 2) If another team is present, we can share the equipment.*"

"*Could sign up for a time before using lab.*"

"*Never had that problem.*"

"*Sign-up sheet.*"

Question 3 continued.

"30-40 min. per group."

"For the first question: N/A, there is no fixed lab time."

4. **How long is a typical lab work session?**

"At least 3 hours."

"For our lab, it has been a term. For each day however about 3-4 hours."

"Anywhere from 1 to 5 hours at a time depending on what we're doing."

"2-3 hours."

"3 hours."

"90 minutes +"

"Who knows- they vary and we do not have set times."

"3-5 hours."

"3-4 hours."

5. **What is the time range of a work session? (The longest, the shortest)**

"Shortest = 3 hours, longest = 8 hours."

"About 3-4 hours."

"1 to 5 hours."

"6 hours - 1/2 hr."

"30 min. - 6 hrs."

"15 min. - 2 hrs."

"5 min to 10 hours."

"5 min - 10 hours."

"8 hours max - 30 min. minimum."

6. **How often do you use the lab during odd hours in a term?**

"Always."

"Sometimes, when my group has an idea as how to do the lab. Come in prepared."

"Anywhere between 5-10 times a term depending on how large a project is."

"Very often."

"Often these are the best times."

"~ Three times."

"All the time."

Question 6 continued.

"Frequently if not always."

"Usually after 3pm- Late weeknights. As often as necessary. 2-3 nights/week."

"Often."

7. **List the safety measures you follow regularly.**

"Avoid burning the equipment."

"When changing a measurement- say from current to voltage- turn the power off."

"Don't touch hot wires. Keep fingers out of equipment while it is in operation."

"Check connections for proper polarity. Check conditions of wires. Monitor equipment for heat."

"Careful of full line voltage connections that are not enclosed."

"Turn off power before touching wires."

"common sense."

"Turn off power before making changes in system (re: electronic components) Keep drinks away from equipment."

8. **List safety measures you know of, but don't do consistently.**

"I know none."

"Never disregard safety measures if I know about."

"?"

"None."

"None."

"In case of emergency, pull the plug!"

9. **For the experiment you're working on now, how many view angles do you use to see what's going on?**

"Magnetic levitation, 3 views."

"Our experiment right now is about moving the table. View is not necessary unless there are something not working. Like why the table is not moving, is the supply on, connections ok. we are controlling."

"We're controlling the position of a table in one dimension. We look at the top and side views of the table. We also look at the outputs from various meters and scopes. So, approximately 5-7 things are observed."

Question 9 continued.

"Magnetic levitation system (3 views) top, front, side."

"all views- under equipt, even small gaps between equipt. As well as all 360 degrees above the device."

"DC motor lab. Computer view. motor and table view. (possibly) oscilloscope view."

"360 degrees."

"360 degrees. x-y table with computer controller. The rotation of the lead screw needs to modeled."

"DC motor (oscilloscope) 2 views, top view/side vied of x-y table. 2 views, front view of dc motor/ side view."

10. **How many seconds does it take you to stop an experiment?**

"5 minutes."

"About 10 seconds."

"2-5 seconds on average."

"Magnetic levitation system = 1 sec."

"Stopping our experiment is immediate since it is left assembled."

"~Five."

"3-5 sec."

"Up to 10 seconds."