Zero Knowledge Access to a Smart Classroom Environment

Edward Lank, Amy Ichnowski, and Shahid Khatri

San Francisco State University, 1600 Holloway Avenue, San Francisco, CA,
lank@cs.sfsu.edu

Abstract. A number of current research projects are examining the design of interactive environments. One assumption made in the design of these environments is that the computers have pre-installed software to access the environment. However, participants now bring with them their own personal electronic devices such as laptop computers, PDAs, and cell phones. This paper describes a framework that supports the dynamic delivery of applications to control and interact with a smart room. Using JINI, portable electronic devices such as laptop computers and tablet PCs become fully functional clients in the smart room. Software is never installed on personal devices that participate in the environment. Applications are delivered dynamically, used while participants are present, and discarded when participants leave. HTML is used to store and archive the information created during meetings.

1 Introduction

Several research projects exist on the design and realization of smartrooms, smart buildings, or smart classrooms [1] [4] [8]. One drawback of many of these smart environments is the pre-configured nature of the environments. Computers and other electronic devices must have software pre-installed on them to access the services of the environment.

The goal of the Smart Classroom project at San Francisco State University is to study alternative mechanisms for accessing ubiquitous environments. We have designed an environment where applications that access the smart classroom are viewed as artifacts belonging to the environment, and provided by the environment when, and only when, participants are using the environment. Software is discarded when participants leave. The particular aspect of the environment we stress is that software is never permanently installed on client computers.

In this paper, we describe the software architecture that realizes this smart classroom environment. The environment contains a number of applications that support interaction. These applications are accessed dynamically, used while present and discarded on exit.

2 Background

Research is on-going in the design of intelligent environments. Here, we briefly describe three recent smart classroom initiatives. We also describe the current
stage of research in the iRoom project [4]. Finally, we conclude with a description of the recombinant computing framework in the SpeakEasy project [6].

The Smart Classroom project at Tsinghua University [8] seeks to provide a classroom of the future exhibiting the extreme integration of technology and artificial intelligence processes. The classroom will be designed to accommodate both local and remote users. Using an electronic whiteboard, the professor can lecture. Students can put up their hands locally or request the floor remotely.

A more mature research initiative is the eClass at Georgia Tech, formerly the Classroom 2000 project [1]. The project experimented with live, real-time classroom notetaking in an environment containing electronic whiteboards and video capture technology. The classroom incorporated two main applications, Zen*, a whiteboard application that allowed a professor to lecture from an electronic whiteboard (potentially using pre-existing slides), and StreamWeaver, which generated image maps of the lecture for post-lecture browsing. One aspect of the classroom also involved the personalization of public information using Wacom data tablets embedded in the classroom [1].

Finally, the Active Classroom and Active Campus projects at UC San Diego seek to incorporate handheld devices into a classroom environment [4]. To accomplish communication with these devices, the project uses html pages. They overcome the loss of interactivity in viewed webpages by requiring periodic refreshing. Chen, Myers, and Yaron describe a similar interaction using handheld devices for in-class testing in the Pebbles project at CMU [3].

The Stanford iRoom project includes a layer of middleware called iROS that enables communication of events and information across machines and an iCrafter system that generates device-appropriate user interfaces when requested [5]. There is still an assumption made that some code exists on the client side that can receive events or request user interfaces.

Finally, the SpeakEasy project allows applications to exchange session objects to pass control of devices around. When applications wish to communicate with each other, they request a session object which allows communication. Each component in the SpeakEasy system expresses the data it understands as a set of MIME types [6].

While both the iRoom project and the SpeakEasy project experiment with communication between applications, an important component is the integration of pervasive computing devices into ubiquitous environments. The architecture we describe here solves a different problem, specifically the delivery of applications to clients in a ubiquitous environment. To handle application delivery two approaches have typically been used. Pre-installed software, as in eClass, or web-based services with reduced interactivity as in ActiveCampus. We provide an alternative that eliminates installed software without sacrificing interactivity.

3 Classroom Architecture

In our project, we leverage JINI technology as a third approach [2]. Using JINI, software exists temporarily on the client, while the client is present in the envi-
environment. This decision was motivated for pragmatic reasons, and was based on a careful analysis of our user demographic. Our campus is a commuter campus, so students tend to live elsewhere and attend courses on campus. We also have a mature student body (average age 26). The result of these factors is that many of our students own laptop computers. While students own computers, they are definitely not experts in computer use. The level of computer literacy of a typical first year student involves browsing the web and using basic application software. Understanding this level of knowledge motivated our selection of architecture.

Figure 1 depicts our Java/JINI-based architecture for the smart classroom environment. The motivation for the architecture is to keep interaction with the system similar to web browsing while permitting the delivery of a rich set of classroom-centric applications.

![Diagram](image)

**Fig. 1.** SFSU Smart Classroom architecture.

The interaction with the Smart Classroom proceeds as follows (see Figure 1 for numbers):

1. When applications are developed for the Smart Classroom, they are implemented as services in the environment. These services, when enabled, register themselves with a JINI server.

2. When clients (students with personal electronic devices) enter the smart classroom environment, the students access a website. A script (javascript and vbscript) ensures that they have an appropriate version of the Java Runtime Environment installed. Students click on a link on the webpage. This link activates Java Webstart, and a bootloader application is downloaded onto their computer.

3. The bootloader application performs multicast discovery and locates the JINI servers available in the environment. It downloads a JINI application that allows control of the environment. This application, in turn, discovers other JINI services.

4. Running applications publish information to JavaSpace to describe their activity in the environment.
4 Classroom Applications

We have implemented a series of applications using our architecture. These applications include a shared lecture tool, a Presentation manager tool, a dynamic testing application, a student chat application, and a team note-taking application. These applications are wrapped by a management user interface that students and an instructor use to control the overall behaviour of the environment. Here we describe the user interface to control the environment and the shared lecture tool.

The Management User Interface The management UI is the initial application delivered by JINI. The sole purpose of the bootloader application delivered by webstart is to locate and launch the management UI on the client-side computer. Figure 2 depicts the management UI we use in our Smart Classroom. To the left, a set of active applications are listed. These applications are in use by participants in the environment. Clicking on any button accesses information in the JavaSpace about that service. Currently each button, when pressed, highlights students who are running the indicated application. We are exploring a variety of visualizations that describe activity as well as participants for each application.

![Fig. 2. User interface to control the environment.](image)

To the right, two panes contain PCons, personal icons, of participants in the environment [7]. When students enter the environment, they are required to enter a name, an email, and optionally an image. These names and images are maintained by a special server that tracks participants. Currently, these PCons serve two purposes. First, they validate a student’s access to the environment. Users receive an immediate, visual confirmation that they have successfully accessed the environment. These PCons also mediate communication between participants. To invite someone to chat, you launch the chat application and select PCons when prompted. This action notifies a participant that you wish to chat.
with them by flashing your PCon in their own interface. Clicking on the PCon allows them to launch the chat application and respond to your request.

The management UI performs a multi-cast discovery process and locates all available smart classroom services. It incorporates these services into its "Execute" menu and into a floating toolbar.

**The Shared Lecture Tool** The shared lecture tool is an application that transmits lecture notes captured on an electronic whiteboard to students' personal electronic devices. Students are able to annotate these public notes in two ways. First, if their computer is pen-enabled, they can make free-hand notes on top of the captured notes. If, instead, students are using a traditional laptop computer, a text area at the bottom of the screen allows them to enter typed notes. Figure 3 depicts the professor and student UI for the shared lecture tool.

![Fig. 3. Professor and Student interface for the shared lecture tool.](image)

5 Results

In this chapter, we discuss the results of a pilot study of the Lecture Application and controller interface as performed in a small reference installation of the Smart Classroom at San Francisco State University. The study involved a group of 9 senior level and graduate computer science students. A tutorial was given by a professor, using the application with the management UI. The lecture tool was run on the Smart Board in the classroom, and the student tools were run on a variety of laptop and tablet computers. The students were given a brief introduction to the tools. After the tutorial, a questionnaire was distributed. The survey provided several findings, including insights into the current note-taking styles of students and a consensus on how easy the application is to use.

One interesting result from the survey was that everyone surveyed owned at least one portable device and several had more. Ten participants had a laptop
computer, 2 had PDAs, and one had a tablet computer. Six of the students never used their devices in class for the purpose of taking notes. Only the user with a tablet computer always used the device for note-taking. Two of the laptop owners stated they used their laptop some of the time.

Most comments were confined to the Lecture tool. In general, most of the students praised the application for its simplicity, convenience, and clarity. There were no problems learning to use it, or accessing it. Most liked the tools, and having the option to type or draw. Some claimed that the application allows them to take notes faster and more efficiently. They also agreed in large part that having a written copy of the professor’s notes archived on their own devices is highly valuable.

6 Conclusion

In this paper, we describe an architecture for JINI-based services in a Smart Classroom environment. The architecture temporarily delivers full-featured applications to clients. Client computers need no pre-existing software, and software is never permanently installed on the clients.

JINI has proven beneficial in the design of such an architecture. The biggest challenge in dealing with Java applications involved the security features of the Java language. Typical Java applications are designed to use a local policy file to specify permissions for downloaded code. Version 1.2 of JINI, however, incorporates a complex mechanism to grant dynamic permissions to downloaded code. An application can therefore be verified and can have specific permissions added to support storing information locally and to support the ability to access network services.

References