An Intelligent Presentation System

HU LIDAN

KTH Information and
Communication Technology

Master of Science Thesis
Stockholm, Sweden 2008

COS/CCS 2008-14

An Intelligent Presentation System

Hu Lidan

lidan@kth.se

Supervisor: Prof Gerald Q. Maguire Jr.

Examiner: Prof Gerald Q. Maguire Jr.

Department of Communication System

Royal Institute of Technology (KTH)

Stockholm Sweden

August 7th, 2008

Abstract

The master thesis project - Personal video: An Intelligent Presentation System (IPS) - was conducted at the Department of Communication Systems, Royal Institute of Technology (KTH), Stockholm, Sweden. The focus of this thesis is designing, building, and evaluating an intelligent environment for giving presentations which makes all the technologies of the existing devices invisible to the presenter, i.e., to remove difficulties of configuration, compatibility, etc. from hindering the user from being able to give their presentation.

IPS is an application in a hot research field - Ambient Intelligence (AmI) which is influenced by user – centered design. Therefore, the goal of IPS is to improve the user's experience. As compared to a traditional presentation system, IPS integrating several independent applications hence improving the user's efficiency and offering greater user friendliness.

In this thesis, a prototype of IPS was designed which combines an online presentation room booking system (running on a context server), a CGI presentation control module (running on a presentation server), and a cross platform control panel (which could be running on a PC, laptop, PDA, cellular phone, etc.). This prototype realized our goals of providing an intelligent and comfortable environment for both the presenters and the system administrators.

Acknowledgement

I would like to sincerely thank my examiner Professor Gerald Q. Maquire Jr for his kind help, considerable encouragement, and great suggestions. During the whole master thesis, he showed great patience by helping me on tasks ranging from the hardware and software configuration, English writing and speaking, to problems solutions and coding. His positive attitude and excellent insights in work and life rewarded me a valuable experience at Wireless@KTH.

I would like to thank my husband for his sincerely encouragement and help in both my life and thesis. I would like also to thank my classmate Ren Xueliang for his solid support in helping me with hardware configuration.

Moreover, thanks go to all the staff at Wireless@KTH for their friendliness and to all my friends for their encouragement and help.

Table of Contents

List of Figures

Table of Acronyms and Abbreviations

Acronyms and abbreviations	Meaning
AmI	Ambient Intelligence
CGI	Common Gateway Interface
FTP	File Transfer Protocol
IPS	Intelligent Presentation System
MRBS	Meeting Room Booking System
NFS	Network File System
PUC	Personal Universal Controller
SER	SIP Express Router
SP	Smart Projector
UDP	User Datagram Protocol

Chapter 1: Introduction

1.1 Problem Statement

It is said that the one thing people fear almost more than death itself is speaking publicly. So, when people give a public presentation, some mess technical problems such as configuring their display setting or activating the external video on their notebook, increase the stress on the presenters and cause their audience to lose interest. In order to address these kinds of problems three guidelines should be followed:

1. Make the environment context–aware: For example, when Lidan walks into a room to give a presentation, she should be recognized by an intelligent sensor. This sensor sends a signal to a local or central room management system. The system could greet the speaker by displaying "Hello! Lidan" on the room's display screen. When she wants to start her presentation, she should find that all her slides are already prepared for presentation. She starts by press a button on the control panel of her PDA or cellular phone or some other simple device which is used for her interactions with the presentation system during her presentation. The problem of initially recognizing the user (in this case Lidan) will be not included in this thesis.

2. Make the user interfaces easy and simple to use: This means that user does not need specific knowledge of the room's technical infrastructure (i.e., IP address, device ID, etc). Similarly, for the handheld device itself, the system should not require users to read long and complex instructions in order to operate it. The user interface should utilize simple widgets, organized in graphic panels which most users will understand at the first sight. This part will be a central part of my work. Hence, this is also a focus of the related studies on user interfaces given in section 2.4.1. The further development of the intelligent presentation system (both methodology and architecture) will be addressed in the following chapters.

3. Hide the internal technologies: Successfully hiding the internal technologies from the user is a key point in measuring the user – friendliness of the proposed intelligent presentation system. All the details of underlying technologies should be invisible. When the presenter has booked a room in advance, she should be able to upload her presentation slides to a server which will take responsibility for seeing that these slides can be presented (i.e., that the correct software is available to render them, that they are ready for when the presentation is to start, and that the slides are removed when they are no longer necessary). When the presentation occurs, the presenter does not need to utilize her own laptop. Additionally, the presenter does not need to arrive in advance to set up their laptop, connect it to the display, etc. -- as the server and the room's infrastructure provide all the necessary resources. The presenter does not need to bring her slides. Therefore, she can focus on giving her presentation.

The Intelligent Presentation System (IPS) is designed with these guidelines in mind. It hides the technologies between the user's PC and the projector, between the handheld device and the projector, and also between the projector and the server PC. Moreover, IPS provides a personalized configuration for each presenter which will be introduced in section 2.2. Details of the methodology will be presented in the following chapters. IPS builds upon the principles of Ambient Intelligence System (AmI), i.e., it is an intelligent context – aware system. AmI, discussed on section 2.1, is the vision of a world where people are surrounded by many intelligent and smart appliances which are seamlessly embedded in the environment [1, 2].

Some related research concerning intelligent presentation is introduced in section 2.3. Key network issues are discussed on section 2.4. The discussion of previous work and some suggestions for the design of a new system are in section 2.4.5.

1.2 IPS in our Context-aware System

This thesis project was conducted at a computer communications system lab the KTH Center for Wireless Systems (Wireless@KTH). IPS is not a single and isolated project, but is connected with other projects which have been or are being developed in this lab. Figure 1.1 shows the relationships between IPS and other context –aware projects, as well as the architecture of the whole context-aware system which we are going to work with.

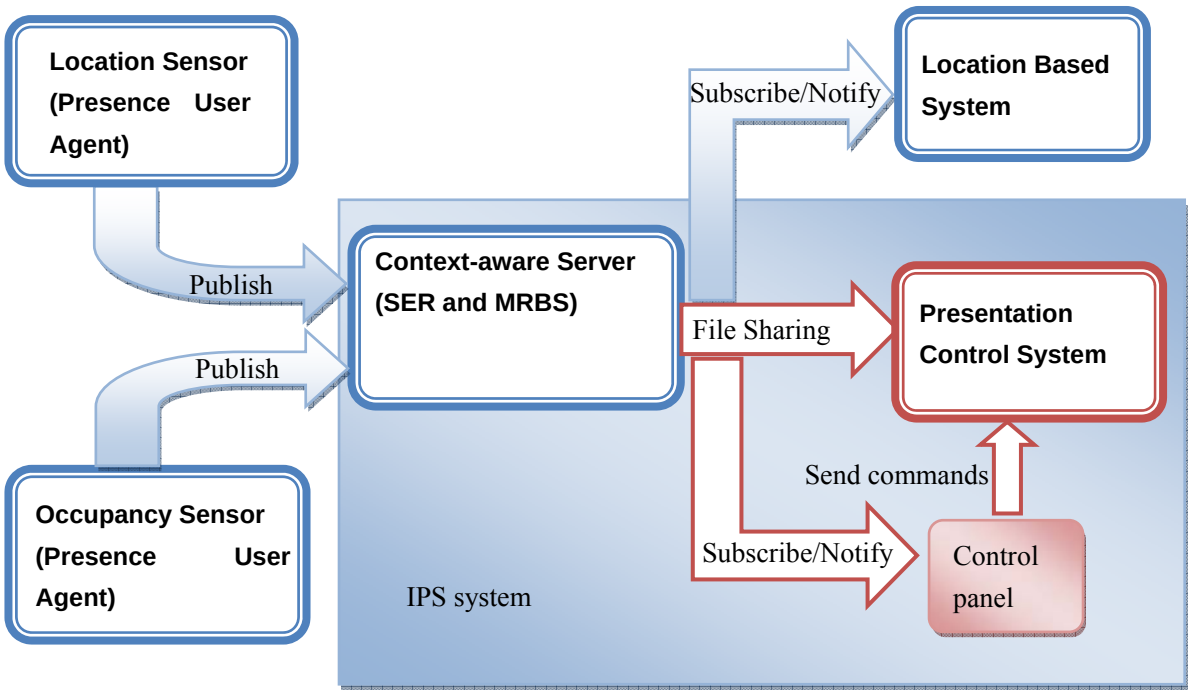


Figure 1.1 The whole context – aware system and the IPS part

Figure 1.1 clearly shows the architecture of context aware system and its relationship to other context-aware projects under this system. The Location Sensor part is a WLAN based

presence user agent which was developed by Haruumi Shiode and is described in his thesis [38]. The details of the Location Based System can be found in the thesis of Sun Yu [39]. The functionality of the SIP Express Router (SER) could be found in the thesis of M. Z. Eslami [40]. To create an Occupancy Sensor System, Daniel Hübinette has developed a prototype of such system in his master thesis project [41] in 2007 and now Ren Xueliang continues this work to develop a meeting detector; which is a presence user agent which utilizes information from the occupancy sensors and the room booking system to infer that a meeting is taking place. Ke Wang is exploiting this context information to automatically cancel rooms' bookings for rooms which are not being used and to take the output of Ren's presence user agent to automatically configure the user's SIP proxy based upon their being in a meeting.

The red highlighted parts are our specific components in this system. They are a context server running the Meeting Room Booking System (MRBS) [42], a presentation control system running a presentation control module, and a control panel providing a user interface for presentation control. These components together with the data interaction between them constitute the whole IPS environment which is contained in the blue area. Thus the IPS builds upon the earlier efforts of other students, most importantly the context – aware server is being used by M. Z. Eslami and configured of this environment. Note that MRBS is open source software which can be downloaded from a website [43]. We have extended MRBS and integrated it into our context –aware system by adding additional functionalities such as file uploading, speaker authentication, etc (see section 2.3.3 and Chapter 3). However, the IPS will not end with simply the three function modules in the blue area in figure 1.1; we expect that in the future all of the other modules could be integrated into the IPS, leading to an ambient intelligent system.

1.3 Organization of this Thesis

In this thesis, we present the IPS project within the context of a context-aware system. Chapter 1 describes the basic topic of IPS and presents IPS as a part of work in a larger context aware project conducted in our lab. Chapter 2 introduces Ambient Intelligence which provides a conceptual basis for IPS, some related research which has close relationship with IPS design, and some issues to be noticed for implementing IPS. Following this, in Chapter 3, we discuss how MRBS was embedded into IPS and how the presentation control module as well as the control panel were designed and integrated together with a context-aware infrastructure in order to enable presentation scheduling service and presentation control. A test and analysis of experimental results as well as suggested improvements will be presented in Chapter 4. An evaluation and the degree to which I achieved my goals will be presented in Chapter 5. The thesis concludes with a summary of our conclusions and suggests future work in Chapter 6.

Chapter 2:Background

The Intelligence Presentation System (IPS) is an example of an important application of an Ambient Intelligence System. So, before we talk about the Intelligence Presentation System, we first introduce the idea of an Ambient Intelligence System, to expose its architecture and to examine its important parts.

2.1 Ambient Intelligence System (AmI)

"We believe that, in the year 2020, people will relate to electronics in more natural and comfortable ways as we do now."

----------Phillips Research. [3]

2.1.1 Overview of Ambient Intelligence System

An Ambient Intelligence System is a smart electronic environment that is sensitive and responsive to people's needs, personalized to their requirements, proactive of their behavior, and reactive to their presence. [3, 4] Ambient Intelligence is an interdisciplinary subject covering intelligent computing, telecommunications, and the human interactions with machines. The emphasis on simple to use user–interface experience prompts the development of AmI and has become a main design goal when designing an AmI system. Typically, in an AmI system, all the internal technologies are hidden inside the environment or the control system. The user sees only the outside interface of each appliance which makes the complex technologies "transparent" and is believed to create a more comfortable environment for the users.

Ambient Intelligence System was originally developed during the 1990s. Phillips was one of the earliest proponents of Ambient Intelligence [3]. In recent years, more and more organizations have adopted this model. The recent European Conference on Ambient Intelligence provides a comprehensive overview of AmI applications [5]. At Carnegie Mellon University, CyLab has released a series successful business products build as AmI system. [6] Some common characteristics of an AmI system are [4, 7]:

Embedded	networked devices and appliances are merged into environment.
Context-awareness	systems can sense their physical environment and people's presence.
Personalized	device provides personalized services to different users
Adaptive	device reacts according to people's needs.
Anticipatory	device anticipate your requirements and pro-actively execute it.

2.1.2 System Architecture of an Ambient Intelligence System

Chen Rui, et al. gives a clear definition of ambient intelligent system architecture in their paper: A Framework for Local Ambient Intelligence Space: The AmI-Space Project. [7] Figure 1 shows their hierarchal model.

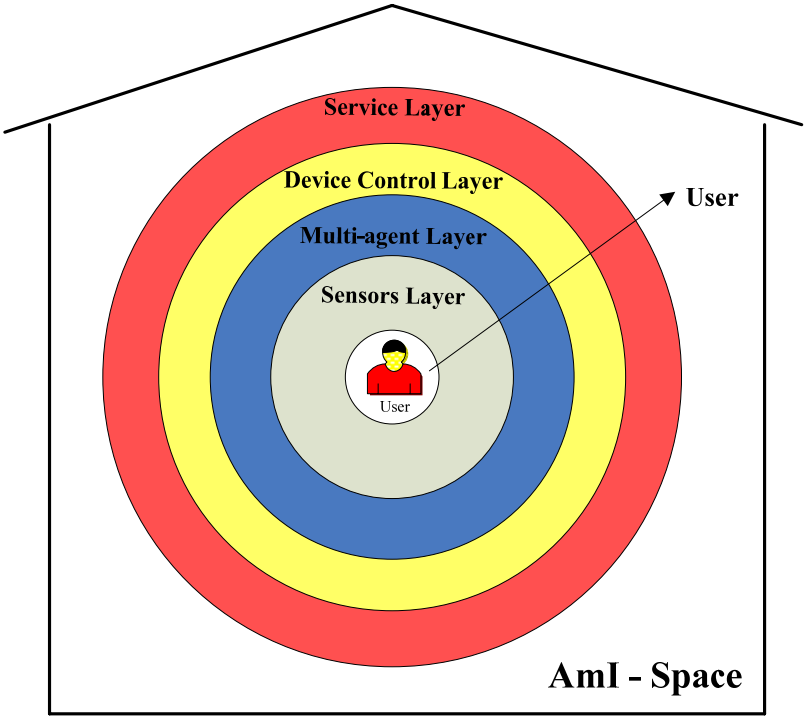


Figure 2.1 The system architecture of an AmI system

Figure 2.1 shows that in an AmI system, the user is placed at the center of the design. All the services are designed to encompass users' needs. [4, 7] This is because an AmI system aims to create and operate a smart environment from user's point of view. The functions of each of the layers shown in Figure 2.1 are:

• Service layer: located at the outer layer. It includes all the different services which users directly utilize. For example, a Location Based Service provides location information to users moving from one location to another location. A context based service provides context aware applications based upon user context, time context, computing context, physical context, etc. [8]. For example, in a smart meeting room, we can use handhelds to access physical devices and control the ambient media stored in the devices in the meeting room. Therefore, education within such environment becomes more interesting and efficient. Many AmI based educational projects use handhelds and mobile computing, as does our proposed Intelligent Presentation System. Examples of these projects include the Palm Education Pioneers Program [9], the Pittsburgh Pebbles PDA Project [6, 10], etc.

• Indoor device control layer: this layer controls all the appliances inside the environment, such as entertainment appliances (such as TV, projector, etc.), cooking appliances (Oven, refrigerator, etc.), and home/building appliances (air-conditioner, water heater, etc.). All the electronic appliances can be controlled by the software in this layer. [7]

• AmI Multi-Agent layer: it is said to be the most important layer in this architecture since it provides the intelligence of the system. This layer includes multi-sensor agent, central context-based agent, device-control agent, function agent, mobile agent etc. In an AmI

system, every agent has its own role and all of these agents work together. In addition, to accomplishing their own work, they frequently exchange data and communicate with each other. [7]

• Sensors layer: the closest layer to the user utilizes a physical interface to monitor users' actions and transmit context information to its upper multi-sensor agents or mobile agents. The three main types of sensors are: embedded sensors, wearable sensors, and portable sensors.

2.1.3 Business Opportunities

Ambient Intelligence provides seamless integration of technologies into everyday life which is viewed by many as being a significant market advantage. As mentioned in section 2.1.1, many companies and research centers have already developed successful business models and some of them have achieved profits. Phillips, the leading market research organization with respect to concerning AmI systems, has used AmI to create a fresh and exiting TV experience [11]. Another pioneer in AmI system is Carnegie Mellon University's CyLab whose product Slideshow Commander is starting to earn money now. [12] They have formed strong business partnerships with handsets companies, to sell Palm and Pocket PC versions of their application *together* with the handset platform. Additionally, Microsoft has been experimenting with ambient technologies for years. Microsoft's "Easy Living" project and their "Concept Home" model were designed to showcase the company's vision of the role new software will play in improving people's quality of life [13, 14]. Their aim is now to commercialize these technologies [14]. The Margi presenter-to-go, developed by Margi Company, is also a successful product to provide a user with a simple, fast, and flexible tool to give presentations using their handheld device, which saves users from lugging around their laptop computer [15]. Moreover, HP uses new wireless technologies to produce a more intelligent presenter-to-go, which enable users to make presentations by using devices in a seamlessly manner [16].

There are many obstacles that exist for business applications. "The main technical hurdles are making computers see, hear, speak, and understand language," according to Robert Laddaga, a research scientist at the MIT's Artificial Intelligence Lab. [17] While recognizing faces and human gestures are not an easy task. There has been plenty of progress in recent years in sensors and wireless, which are needed to make "intelligent" spaces possible. For example, the location sensor developed by Harumi in Wireless@KTH is a wireless based sensor to detect the signal strength to identify the presence of the user agent.

In summary, we can see that there is a lot of research work and some initial products that are exploiting Ambient Intelligence. In addition, there are EU projects, such as the Ambient Network project [18] which are trying to enable wireless devices to be able to discover each other and offer services to each other by dynamically composing an ambient network. It will also enable an integrated, scalable, and transparent control of network capabilities.[18]

2.2 Intelligent Presentation System (IPS)

2.2.1 Relationship between IPS and AmI

Our Intelligent Presentation System (IPS) is an application of AmI. The goal of IPS is to enable the presenter to focus on their presentation by freeing them from dealing with the details of computer and projector configuration. IPS users do not have to worry about matching the resolution of their laptops with the projector. IPS also provides Identification Based Service which means that the system could recognize the presenter when he/she enters the meeting room. The meeting room where the presentation will be held should be context–aware, thus a user simply utilizes one or more portable devices that store the presentation files. IPS will also provide an identification-aware presentation service which will be the topic of future thesis work.

The service agent in our case is software running on a presentation server which is physically connected to the projector and controls this projector. The details of this will be presented in the next sections.

2.2.2 General Introduction

IPS helps users save a lot of time by avoiding having to deal with the technical issues of connecting their computer and the projector which they are using. Hence the presenter simply focuses on giving their presentation! There are four main devices used in providing this service: a handheld device, a projector, a presentation server which connects the projector and the handheld device, and a context server which is used to management presentation files. The basic interconnection between these devices is shown in figure 2.2.

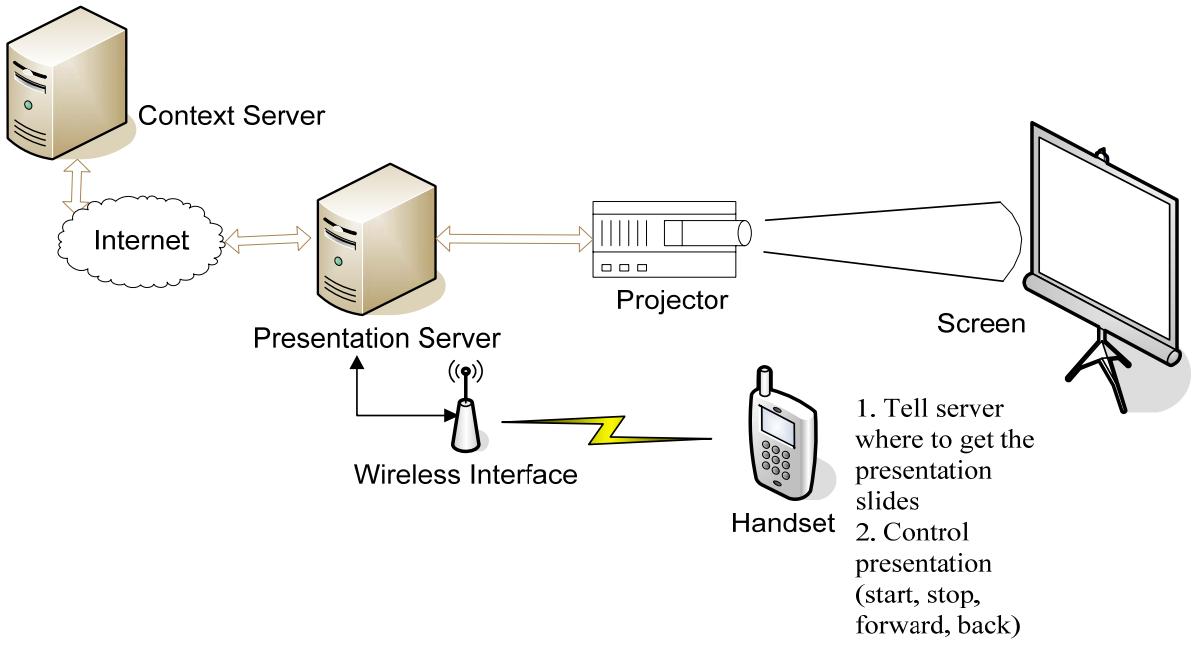


Figure 2.2 Presentation system structure

The user sends his/her presentation slides to the context server before the presentation starts. The administrator who schedules the presentation indicates the time and place of the presentation and passes this information to the user who will give the presentation at this time and place. Before the presentation starts, when the user enters the room, a sensor detects that the user who is to give the presentation has arrived. The sensor (or sensors) communicate with the context server, which in turn sends a NOTIFY message to the presentation server. This presentation server now begins listening for commands from the user. The user uses their choice of handset. On this handset they can press a button to start the presentation. During the presentation, the presenter's handset is responsible for controlling the presentation. After the presentation is finished, the user uses the handset to terminate the presentation and turn off the system. When the user leaves the presentation room, the presentation slides stored in the machine will be automatically deleted (this may be based upon the scheduled time of the presentation, the termination of the presentation, or at some particular time each day.)

Because the user need not bother about configuring his or her computer and the projector, he or she can focus on the presentation itself. The whole process is expected to be much more efficient and convenient than the current 5-10 minutes required for each presenter to connect their computer to the projector and get started [74].

2.2.3 Application Scenarios

Scenario 1
Users use their PDA or PC to upload their presentation slides. Later they can use the same PDA or another intelligent handheld device to control the presentation (to start, stop, go back and forward in their slides during their presentation). Here, PDA is used both for uploading presentation files and presentation control. In order to do that, the user should first put presentation files on PDA.

Advantages: this approach avoids many difficulties for a user who is familiar with a PDA and the presentation control system. The user primarily interacts with software which they are familiar with or is very simple to use. However, it requires that the presentation client (pClient) be easily and simply deployed to the user's choice of device.

Disadvantages: this approach requires the user know how to upload their presentation files to the server through PDA. In addition, user needs to know when the room is booked; thus notifying the user of this room booking and giving them a simple means of referring to their scheduled presentation will be an important part of IPS. Sometimes the user is more familiar with his or her PC than with another device - thus the user should be able to use their PC to control their presentation. However, the user will avoid needing to configure his or her computer's display resolution to match that of the projector. Thus an important aspect of the design, implementation, and evaluation is to reduce the difficulty of the user deploying and using this client. Additionally, this removes the need to connect the user's computer with the projector by cable in the presentation room. Therefore if the user uses a wirelessly connected device to control their presentation, they are free to move about the presentation room.

Scenario 2
We separate the process of uploading the presentation slides and the controlling presentation, by distributing these functions over two machines. Thus, the user uses their own computers to upload their presentation slides to their presentation server. When they want to give their presentation, they simply use a PDA or other handheld device to start the presentation (opening the file previously stored at the server) and to control the presentation slides. When the user finishes their presentation, they can use their handheld device to delete the slides from the server or allow an automatic deletion process to perform this task.

Advantage: It is much more user–friendly (as compared to scenario 1). This process mirrors the common way we currently give presentations. The user is generally most familiar with his or her computer, thus the process of uploading presentation slides should be quite straightforward (i.e. similar to uploading other files to a server). Additionally, learning to use a PDA or other handheld devices to control the slides should not be too difficult for the user (This assumes that we can provide an easy means to download the software to do this to the device which the user would prefer to use, be it a PDA, their laptop, or a PDA.).

Disadvantages: The user must manually delete their presentation slides from the server or trust that their presentation slides will be automatically deleted at some point in time. If the user forgets to delete the presentation files after the presentation or the presentation files are not automatically deleted, their presentation might be available to others - which might be objectionable to users.

Scenario 3
Compared to scenario 2, here the user simply uploads their slides to the context server. After the user's presentation (the time and place of the presentation) has been booked through a presentation management system, as part of this booking, the user sends the URL of the presentation slides to the booking system The booking system in conjunction with the presentation system must retrieve the presentation before the scheduled time of the presentation and cache the presentation at the presentation server. At the time of the presentation, the handheld device in the presentation room is used to indicate which presentation slides are to be used based on the previously sent URL. When the user goes to presentation room, the slides should be already downloaded based upon the information the user supplied when he or she booked the room using the booking system.

Advantages: this approach reduces the amount of effort required of the user. The user simply provides a URL to their slides using the combination room booking and presentation management system. This approach reduces the actions required of the user, unifies the booking and uploading of the presentation slides, and ensure that the user's presentation is ready in the room when and where they have booked their presentation is booked.

Disadvantages: This method assumes that the system will be able to retrieve the presentation using the URL prior to the presentation time and does not provide a way for the user to know if the presentation has been cached by the presentation system or not.

Scenario 4
Summarizing the scenarios above, here in IPS, we run a presentation management system on the context server to handle presentation room and schedule booking, slides uploading, and speaker authentication. Between the context server and the presentation server, there is an instantly file sharing system which makes the presentation slides immediately get by the presentation server once the speaker uploads them to the context server. When it comes to the presentation time, the speaker can log into the control panel through any devices as long as the device can access to the Internet (the control panel is required to support cross platforms). Through the control panel, the speaker can control the presentation. After the presentation, the speaker can use the control panel to exit the displaying and delete the slides stored on the presentation server.

Advantages: this approaches implemented a timely file sharing between the context server and presentation server, which reduces the efforts for file transmission. Well, it also reduces the amount of effort required for the user. Clearly defines the functionality of the presentation management system and adds the speaker authentication for security reason. Additionally, we provide flexible choices on presentation control devices by offering a cross platform control panel.

Disadvantages: no disadvantages have been identified at the present. After we realize it, we will test, analyze, and improve the weaknesses of this scenario later.

2.3 Related Research

There are some related researches about how to intelligent control a presentation which has close relationship with our later research and implementation on IPS. They are the Pebbles Slideshow Commander, the presenter-to-go and MRBS. The Pebbles Slideshow Commander is a software application and allows the user to control a PowerPoint presentation running on a PC by using a PDA. The presenter-to-go is a software and hardware package. It enables a user to use a PDA directly communicate to a projector which stores the slides. However, the difference between the Pebbles Slideshow Commander and the Presenter-to-go is that the second one avoids the need to use a computer. The meeting room booking system (MRBS) which provides online room booking,checking and cancelling. This software is used in IPS to act as the presentation management system.

2.3.1. The Pebbles Slideshow Commander

The Slideshow Commander is an application developed as part of Carnegie Mellon University's Pebbles research project. The goal of this project was to explores how hand-held devices can be used to communicate with a PC, with other hand-held devices, and with smart computerized appliances such as telephones, radios, microwave ovens, automobiles, and factory equipment [17]. There are both Palm and Windows CE / Pocket PC versions of Slideshow Commander. Both versions have been licensed for sale by Installigent (formerly Synergy Solutions, Inc). [12, 19].

The Slideshow Commander allows the presenter to control a Microsoft PowerPoint presentation running on a PC by using a PDA. Using this PDA, the presenter can easily navigate to any slide. The presenters can see not only the current slide, but also their notes for the slide. The user can also browse a list of the titles of their slides to quickly and convenient quickly go to a specific slide (for example, to answer a question or to provide additional information). The user can also annotate the slide by drawing pictures on the current slide. Additionally, with a wireless link equipped handheld device this application allows presenters to convey their message from anywhere in the room, giving the presenter freedom to move around the room enabling more natural interaction with their audience. Additionally, the audience in the presentation room can also use their own laptop to see the current slides and annotate their own copy of the current slide which helps the audience remain synchronized with the presentation and to produce their own notes concerning the presentation. Moreover, the presenter can display the annotations of the audience on the main screen to encourage questions and discussion, which facilitates increased interaction between the audience and the presenter. There is also a timer embedded in the PDA recording the time of the presentation - in order to allow a timed presentation - either by defining a time per slide or a total amount of time for the presentation

2.3.2. The Presenter-to-go

Margi Presenter-to-go is a software and hardware package which enables a user to use a handheld device such as PDA to control his or her presentation slides while displaying them through a projector. Margi has already introduced products based upon this and related products (for example smart projectors- which come with their hardware and some of the software pre-installed in them).

Margi Presenter-to-go offers two important services. First, using Presenter-to-go's Mirror function and a projector, the presenter transmits the contents of the PDA’s screen directly to a projector to display the contents of their device’s screen. Second, the PDA can be used to control a PowerPoint presentation avoiding the need to use a computer [20]. In this second case, the projector or other hardware has to perform the rendering of the Microsoft Power Point slides to the projector.

The Presenter-to-go hardware might use a MARGI Systems SD card, which is an expansion device that allows presenters to connect directly to projectors equipped with a VGA port [21]. This VGA port uses a SD card which implements a simple VGA frame buffer and to which a VGA cable may be attached. The MARGI Systems SD card provides a powerful handheld solution – but requires a direct connection to a projector (i.e., the PDA directly provides the video for each slide to the projector - generally the frame buffer implemented in the SD card utilizes a greater resolution that the PDA's own screen). The device is a small box about 1.5 x 2 inches with a VGA connector on one end (to connect to the projector), a cable with SDIO (Secure Digital Input/output) card on the other end (to connect to the PDA), and two IR ports on either side. Figure of the Presenter-to-go SD card can be found in Pocket PC Central [28].

The client software can be installed on a PDA through ActiveSync [20] or other synchronization method from a PC or laptop. This client software includes two components: the presenter part controls the presentation enabling the user to navigate between slides, start and stop the presentation, etc.. (the configuration of this client allows the user to set one of four different resolutions and convert their PowerPoint format to the specific Margi format); the other part of the client is a mirror function which displays any program running on this PDA via the attached projector[20, 21].

Several drawbacks exist in Margi Presenter–to–go. First, the presenter has to use the cabled Margi SD card with its wired connect to projector, which limits the movements of the presenter during the presentation. Second, before the presentation, the user needs to transfer his or her presentation files into Margi format. Additionally, if the user wants to use a handheld device, he or she must send files to the handheld device using HotSync™, ActiveSync, or PC Connect. These procedures are time consuming for the presenter to perform before their presentation. However, HP provides a better solution by using their variant which is called the HP Wireless Presenter-to-go Software which was specifically designed for professionals who wish to present wirelessly or PC-free [16, 22]. The users only needs to install a Margi System's client software on his or her laptop. Then, use the laptop to communicate (via a wireless or wired network interface) with a projector equipped with the HP Advanced Connectivity Module or with the Smart Attachment Module (SAM).[22, 23] The HP Presenter–to–go is compatiable with a variety of IEEE 802.11b devices such as notebooks, Pocket PCs, and Palm devices which give users flexible choices in choosing their preferred device to use when giving their presentation. Additionally, by using a memory card, the presenter can do a presentation in a PC–free mode, more details about such PC-free mode presentation can be found in the HP presenter-to-go manual on HP website [22].

2.3.3 The Meeting Room Booking System

The Meeting Room Booking System (MRBS) is an existing open source software package for multi-site meeting room booking. It is used at Wireless@KTH, the Department for Communication Systems, and a number of other departments at KTH. In our IPS architecture, we assume that this software is run on our context–aware server and it has been extended to provide support for uploading the presentation files temporally before the presentation. Note that the MRBS system could be running on another computer, the only requirement is that we have to be able to add some software to provide our extended functions and that the server has to be available on an external network in order for external user to be able to upload their presentation; while at the same time it needs to be connected to the internal network in order to communicate with the presentation servers which are **only** attached to the internal to KTH network. Actually, the MRBS act as the presentation management software (i.e., it comprises the server side of the scheduling, management, and file uploading functions). The modules which we have added are described in Chapter 3.

2.4 Issues to be Considered

2.4.1 Personal Interfaces: PDA, Cellular phones and Shortcutter

As to the personal interface to loading control panel, we firstly analyze several optional devices to find their common points and their respectively distinctive features in controlling presentation. Then we will fix how to build a cross platform control panel. Therefore, the initial solution which we will use is PDA. IPS should support PDA in that PDA is involved in all the other projects in the context-aware system shown in figure 1.1. Another device we consider is the Shortcutter which inspired the design of our user interface. Additionally, we also think of the cellular phone interface which is now the most common device used for people.

2.4.1.1 PDA

In IPS, we plan to initially use an HP iPAQ 5550 PDA as the handset and for the user to control the presentation. We will use Microsoft's XP or Linux (or both) in the rendering server. The initial approach will be to use a Linux (rendering) server running Open Office - as this package can read a wide variety of different formats and the sever is expected to be more robust under Linux than under Windows XP. The HP iPAQ is running Microsoft's Pocket PC operating system. We can easily create a user control panel and deploy it on the PDA.

The integrated program development environment (in this case Microsoft's Visual Studio 2005) makes the development of a simple graphic user interface simple to do. Note that this same tool can generate a device platform specific executable (in this case an ARM binary for use on the HP iPAQ 5550) or a .NET CF version which can be executed on any device supporting the .NET virtual machine (this includes most widely available operating systems - given the appropriate virtual machine).

2.4.1.2 Shortcutter

Carnegie Mellon University's Human Computer Interaction Institute's Shortcutter allows a programmer to easily create panels of controls on a handheld device by drawing buttons, scroll bars, knobs, and other widgets which the programmer believes most suit the user's needs. This tool includes dynamically linked libraries which enable the software to control any PC application. The buttons can be any size you (as the programmer) like. A key feature of Shortcutter is that it gives the programmer the necessary tools to easily build a user-friendly interface on a PDA [24].

Shortcutter exists in two versions to generate applications for either the Palm or Pocket PC platforms. For a Pocket PC equipped PDA, the programmer can create different panels to control different applications. Each panel is a data file which can be renamed or deleted. Currently, there are three kinds of panels included in the sample download [25, 26]. These are a Number Pad, the Windows Media Player, and an X10 lighting control panel. Figure 2.4 (a,b, and c) shows these three types of panels.

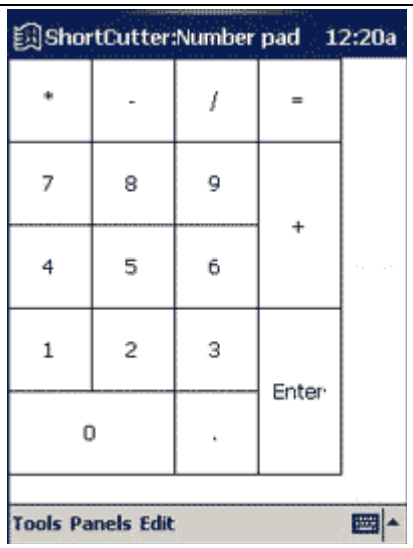
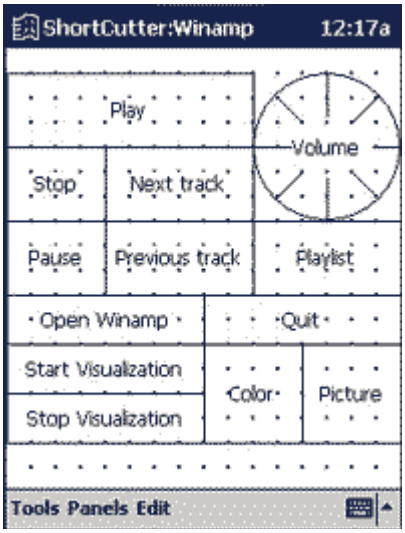
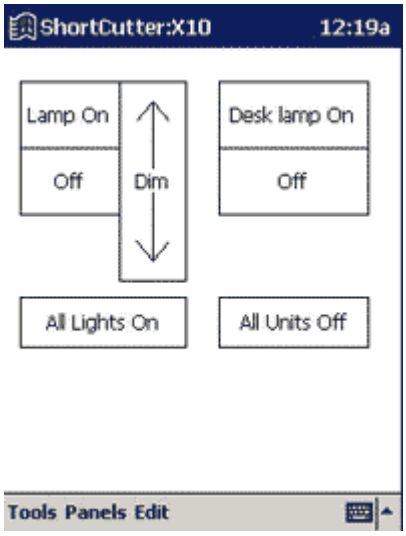


Figure 2.4. (a) Number Pad panel [25] - with the permission of Brad A. Myers



(b) Windows Media Player panel[25] - with the permission of Brad A. Myers



(c)X10 lighting control panel [25] - with the permission of Brad A. Myers

If the programmer wants to create his or her own panel, he or she can utilize the edit mode -- draw buttons or other widgets and to couple these to actions by switching to running mode [25]. Each of the widgets supports one or more actions which have been predefined. Programmers can also create their own actions by doing advanced programming.

2.4.1.3 Smart Phone

The mobile phone communicates with the presentation server through a Bluetooth, WLAN, or wide area network interface. The person giving the presentation just presses a button on their cellular phone to control the presentation of their slides.

I set up an experimental environment using a Sony Ericson W880i cellular phone, running the Symbian OS 6.0 operating system[27] and supporting Bluetooth wireless technology . I used the mobile to wirelessly communicate with a laptop which had a Bluetooth interface inserted into it. I set the main manu button (Figure 2.5) as the start and stop button to control the presentation slides and the small snap button (Figure 2.5) as the navigation button to navigate between slides.

Figure2.5. Button settings in a smart phone to control a presentation

In this experiment, I invited five test subjects to use the mobile to control a presentation. The presentation files were stored in the presentation server. After several trials, I found several obvious inconveniences when using a cellular phone as the control device. First the buttons of most cellular phones are so small that the user is very likely to mis-press buttons. When the presenters are nervous during their presentations, mis–pressing a button will tend to make them even more nervous; which could in turn lead to a further mis–pressing of the buttons, resulting in a vicious cycle of increasing nervousness and errors. Another problem is the mobile's sleep time out, as in order to save battery power the cellular phone tries to quickly enter sleep mode. However, if the user spends a long time about talking one slide, then the cellular phone will go to sleep. Now the user has to press the button twice to continue his or her presentation -- potentially also needing to wait for the device to return from sleep mode, power up the wireless interface, etc. However, these problems can be solved by manually prolong the awake time longer in cellular phone user settings or even programming the cellular phone to stay awake longer.

While using a cellular phone, the user can use it to navigate, start, and stop the presentation of their slides; any more advanced operations such as editing, renaming, etc. requires installing special software into the cellular phone. Such additional application software will be introduced in the next section.

2.4.1.4 Other Interfaces for Controlling a Presentation

Additional tools (beyond those found in the Slideshow Commander) are included Carnegie Mellon University's personal universal controller (PUC), a software framework running on a PDA or other mobile device that is capable of providing an interface to allow users to control any appliance in a room [26, 65]. The concept of using a handheld device to control the presentation is that when a user points the handheld device at a projector, the projector will send to the handheld device a description of its control parameters. An application running in the handheld device uses this information to create an appropriate control panel, taking into account the properties of the handheld device and the preferences of the user. The user can then control the projector using the handheld device. PUC has not yet become a mature product, thus it is not available to control real appliances today. Additionally when it becomes available, it will still require the user to learn how to perform specific operations which adds some inconvenience [29].

Another tool is PECo which provides 3D-based interaction with an UPnP meeting room [32]. Using PECo, a user can access his or her physical world and its devices, as well as his or her personal media regardless of the storage location of this media. In order to archive these goals, PECo uses a two dimensional interface for document access and a three dimensional interface for environmental device access [33]. For the personal media, such as presentation slides, PECo integrates the virtual personal world into the real physical environment. The relationship between personal video and real physical environment is shown in figure 2.6. The user simply drags a presentation from his or her computer and drops the slides onto the server connected with projector.

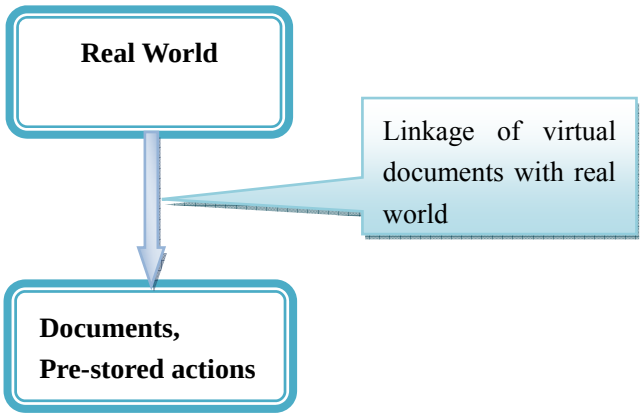


Figure 2.6 Relationship between user documentation

Also, both Philips' iPronto and Logitech's Cordless Presenter [34] can be used to control presentations. However, they do not provide intelligent interfaces for editing or interacting with the slides or interacting with the presentation environment.

In table 2.1, we compared the features and the limitations of each interface for controlling a presentation.

Table 2.1 Features and limitations of each device

Device	Features	Limitations
PDA	It is simple to develop a simple graphic user interface for control presentation.	The screen size is limited so that not everyone get use to utilize a PDA for presentation control
Cellular Phones	It is the most common handset device we use in everyday life. If it is well configured, it is easy to navigate between the presentation slides.	The buttons on cellular phone is very small which may cause mis-press on buttons. The screen size is limited.
Shortcutter	It is easy to build a user-friendly interface on a PDA	We need to pay for the license.
PUC	It provides an interface to allow users to control any appliance in a room.	It is not a mature software package now
PECo	It provides 3D-based interaction with a smart meeting room. (e.g. a two dimensional interface for document access and a three dimensional interface for environmental device access)	We needs extra hardware support in a smart meeting room to communicate with PECo.
iPronto and Logitech's Cordless Presenter	Can be used to control presentation wirelessly	They do not provide intelligent interface for interacting with presentation slides

Table 2.1 describes the strengths and weaknesses of each kind of device. It is very hard to decide which one will be most suitable device for loading the control panel to interact with slides during the presentation. Therefore, we consider making a web based interface to the presentation server available - as this can be used on any device with a web browser and connectivity to the presentation systems. (Note that issues due to the presentation system being attached only to an internal network are addressed in seciton 2.4.3.)

2.4.2 Networked Projector and Networked Display

A networked projector is a smart device which has a direct connection to the local area network (LAN). You can configure such a projector to have its own IP address. Such

networked projectors offer two main improvements compared to the earlier projectors. First, presenters can upload their presentation slides directly to the projector's internal storage and display presentations without worrying about on how to connect their own laptop to the projector [30]. Some client software installed on the networked projector makes the projector appear as a network neighbor (i.e., as a networked storage device). So, the presenter simply drags the files associated with their presentation into the projector‘s storage from his or her own computer. Even, if there is no specific client software installed, the presenter can make use of onboard projector’s firmware once peer to peer file sharing is established between the user PC and the projector. The embedded PC in the networked projector acts like the user’s PC to render the user’s presentation to the projected display without any need for an attached computer and the difficulties of cabling, adjusting resolution, timing, and synchronization issues. [30]

The second improvement is that a network projector supports remote management. By checking parameters such as lamp life, power status, etc. via the network, administrators can remotely monitor and manage a large number of projectors. SONY sells a “wisdom” solution which helps integrate projectors into any of the many available SNMP based management packages. If the projector implements self diagnostics, these can generate a SNMP alert (which is sent via the network to a network management system) when abnormal conditions occur. Moreover, based upon a schedule, the projectors can be automatically switched off and on remotely, so that projectors are not left on all day and all night [30, 31]. Similarly, InFocus offers software which enables a centralized solution for managing multiple projectors. Using their software, you can run presentations simultaneously on multiple projectors from a central location.

Many companies (including SONY) provide wireless connectivity which facilitates communications between the projector and handheld device [31]. Wirelessly connected to a handheld device, the projector adds more flexibility for presenters. Thus, the networked display eases use by users, management by administrators, and support by staff. This is both because many details are hidden and because the device is now controlled, managed, and used via the network. Presenters can upload slides or videos into the projector via the network. Additionally, they can connect their PDA to the projector through one or more wired or wireless interfaces, and display whatever they want from their PDAs to a big screen.

There are many different existing handheld interfaces which can be used to control a presentation. They can be divided into two categorizes: one is a fixed interface such as the Logitech Cordless Presenter [34], InFocus Navigator™ Remote [66], and cellular phone. These devices have fixed buttons which allow the presenter to navigate between their slides and start or stop the presentation. The limited functions of these controls are simple enough that presenters do no need to learn any special skills to operate the projector. These devices are the most common ones we use today. The other category of handheld device has an intelligent or programmable interface such as a PDA, Carnegie Mellon University’s Shortcutter or PUC, etc. Because it is programmable, the control panel can be changed according to a user's needs, rather than having a fixed functionality. With this feature, the

presenter can not only navigate their slides, start and stop the presentation; but the presenter can also annotate their slides in real-time during their talk. Additionally, the software can change the control panel or give users the ability to create their own panels (as Shortcutter does). Therefore, an intelligent presentation system should support a handheld device that is user–friendly (i.e., does not need much time for the user to learn how to use it), offers suitable functionality (i.e., easy to operate, reminds presenters when something happens), and with a wireless connection to the projector.
Thus in this thesis, we will assume that a PDA is used to provide this handheld interface device as part of an intelligent presentation system. For the user control panel, before users learn how to create panels of their own, a predefined simple and easy to use panel will by default be made available for presenters. PUC's design offers a good choice as a starting point for this interface. We could pre-program the control panel for the projector and use a hard coded interface for the projector, then, everything should be ok [29]. However, in an effort to have both a cost effective (due to its portability to multiple platforms) and extensible solution, I plan to implement this predefined control panel by using C and deploy it firstly into a PDA (as PUC does).

2.4.3 Network Firewall and NAT Issues in File Transmission

In IPS, at first we assume that there is a FTP file transmission between the context server and the presentation server. Therefore an FTP server should be built in the presentation server or in some other network attached computer - this enables the presentation file to be transmitted from the user's PC. In any case, the machine to which the user's presentation file is transferred should be attached to the network, thus it can make the presentations available via a distributed file system to the IPS presentation servers. When uploading files from the user's PC, if the user's PC is behind a NAT, there would be a problem for a user to gain access to the FTP server. Because of how NAT works, in FTP active mode, the FTP server initiates the data connection by connecting to the external address of the NAT gateway on the port chosen by the user's client. The NAT machine will receive this, but because it has no mapping for the packet in its state table, it will drop the packet and will not deliver it to the user's client. Alternatively, in passive mode, when the user's client requests that the server pick a random port to listen on for the data connection, the server informs the client of the port it has chosen, and the client connects to this port to transfer the data. Unfortunately, it is not always possible to connect to the file server, for example because there is a firewall between the client and the FTP server; the firewall may block the incoming data connection. [35]

There are simple solutions to this problem. First, if the user's PC is behind a firewall / NAT, then a packet filter in the NAT could be used to redirecting FTP traffic to an FTP proxy server. A packet filter is a piece of software which looks at the header of packets as they pass through, and decides whether to drop or accept the packet or doing other operations of the packet. Under Linux, packet filtering is built into the kernel which acts to "guide" your FTP traffic through the NAT gateway/firewall, by actively adding needed rules to PF system and removing them when done, by means of the Packet Filter anchors system [35, 36]. Alternatively the firewall could have a ported opened for incoming FTP connections. Or the firewall could have a port opened for incoming HTTP connections and we could use an

HTTP connection to upload the file. In IPS, I will configure all the devices to each have a single public IP address in order to avoid some of these problems.

2.4.4 Temporary Storage of Presentation Files

In the IPS architecture, when it is time for the presentation, the software running on the server side will automatically download the slides in advance (based upon the schedule in the room booking system). This assumes that all the files for the presentation will be stored in advance in a network attached server which works together with the IPS to offer a context aware environment for presentations. We assume that the control panel software is already loaded into the handheld device which the presenter will use. The presenter begins by pressing a button to initiate their presentation. After the presenter finishes his or her presentation and the whole session has ended, the slides and other related files will be deleted from the server (for example, based upon the time period for which the presentation was booked). Meanwhile, the IPS should automatically update the projector for the next presentation.

To begin with we will assume (like most existing products) that we must support OpenOffice presentation(see for example the application which Athanasios Karapantelakis has written - see section 2.4.5). However, we will design the system so that it is not difficult to support additional document types, such as Microsoft's PowerPoint document types (As OpenOffice compatible with PowerPoint, it will be not a big problem for support on both the *odp and *ppt document types). As to video and audio files, they are not supported by the existing products; but should be handled correctly as part of the presentations which will be the future work.

2.4.5 Prior Work and Some Suggestions

Athanasios Karapantelakis created a simple presentation system. It consists of two parts: a client called pClient, and a server part, called the consoleAgent. The console agent is a simple UDP server application that works in conjunction with OpenOffice (while it currently supports OpenOffice, it could easily be extended to support other document types) to implement a presentation application. The pClient software runs on a PDA and uses the File Transfer Protocol (FTP) to transfer the presentation from the user's device to the service agent's local storage, then it uses its own Projector Control protocol to allow the user to control the presentation of slides by remotely controlling the projector [37]. Figures 2.7 (a) and (b) illustrate the structure and the function of these two operations:

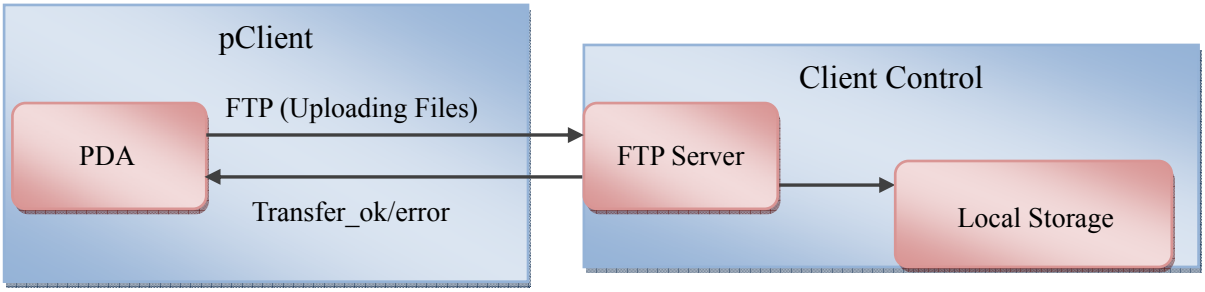


Figure 2.7 (a) Uploading files from user device (PDA) to local server

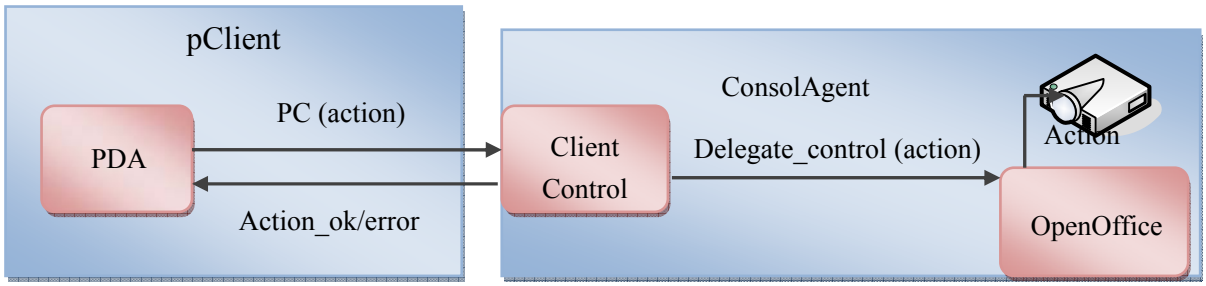


Figure 2.7 (b) Controlling message sending from PDA to the presentation

It is possible to run his pClient either in a PDA or a PC. Initially, I ran both this pClient and the consoleAgent on the same computer using .NET software (to provide a virtual machine for both applications).

An important first step was to separate the two parts to enable them to run on two separate platforms and to separate the file transfer part of the client from the presentation control part of the client. This required rethinking both parts of the current application. The result is that giving a presentation consists of:

1. A file transfer of the presentation file(s) from the user's device of the presentation file(s) to a networked file server.
2. Transfer of this file from the file server to the computer attached to the projector
3. Rendering of the presentation file(s) by the computer attached to the projector under the control of a projector control client running in a handheld device which provides the user interface during the presentation.

The improvements and the resulting new structure of the system are shown in figure 2.8. Note that the file transfer between the context server and the presentation server were replaced in my final implementation by using a distributed file systems between the context server and the presentation sever (i.e., the server connected to the projector). The functionalities inside the client control module running on the server attached to the projector

are shown in figure 2.9.

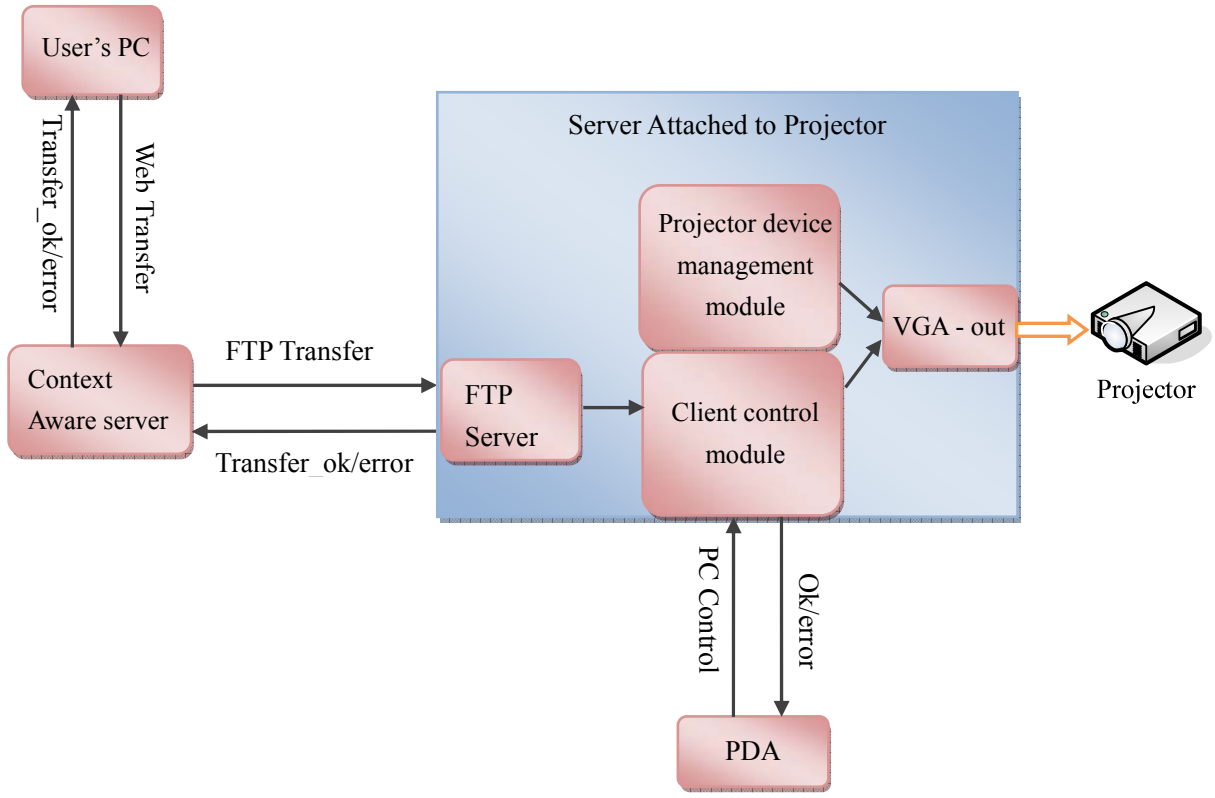


Figure 2.8 A new structure of new IPS

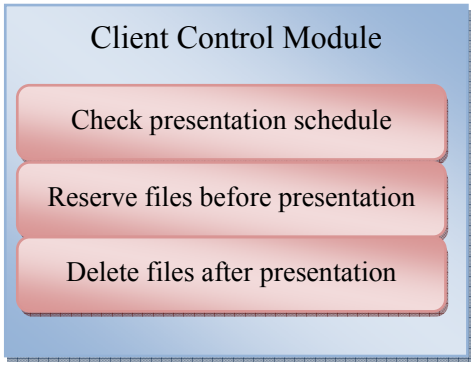


Figure 2.9 Functionalities of a client control module

Several metrics will be used to measure the new system IPS. The initial three metrics are:

1 User-friendliness: I will measure the user – friendliness in terms of the number of steps which the presenter conducts and the required skills that the presenter needs to configure the IPS versus the old presentation system. The fewer the number of steps required and the less skills needed, the better the system is.

2 Average time: I will first measure the average time the presenter needs to prepare for a presentation both in IPS and the old presentation system. A comparison of the average times will used to examine the efficiency of IPS.

3 Hardware and software costs: for the software, time and memory consumed will be considered for running software in networked file server, the computer attached to the

projector, and the PDA. For the hardware side, the cost of devices and ease of management will be considered.

Chapter 3: Goals and Implementation

In this chapter, we explain the details of prototype's design and the interface design. We will begin by first stating our methods and goals with regard to IPS as a whole (section 3.1). Then based on the IPS physical structure (as figure 2.2 showed), we built a prototype in our lab environment for both the hardware and software. After that, we will give a very detailed analysis of the prototype's design and interface design (section 3.3 and 3.4). These two parts are the core parts of our implementation. Finally, we will give a summary of the implementation - before describing the testing of this implementation in the following chapter.

3.1 Method and Goals

According to figure 2.8 and figure 2.9, the overall work flow of the IPS should be: An administrator of the IPS management system books a presentation session (including the time and place where the presentation will be held) for the presenter. If the booking is successful, a random code will be generated to be used as an authentication code (ID-code) by the presenter. The administrator who has scheduled the presentation will send a URL to the IPS web interface and the ID-code to the presenter. The presenter can use the ID-code to login to the uploading page to upload his / her presentation slides before the presentation. After the files are successfully uploaded, he /she can download the files by visiting the uploading folder through a URL to the context server. At the same time, the files should be sent from the context server to the presentation server (in this case we will uses the Network File System (NFS)). Finally, the presenter can control the presentation by using the client control panel in a PDA or any other devices which can log in to the control panel.

Thus, the division of work with regard to each component in IPS is clear. The context server is responsible for the presentation booking and file uploading. The presentation server receives and executes the control of the presentation based upon signals from the PDA, and outputs the video to display to the projector through the VGA port. The PDA interacts with the presentation server during a presentation, so that the presenter can start their presentation, move forward and backward through their slides, terminate the presentation, etc. Note that here we just refer to the PDA as the user's control device to load the control panel. But actually, the user is allowed to use any kinds of devices as the controlling device as long as it supports to run our control panel. Later in section 3.3.5, we will detailedly describe how the control panel we designed supports cross platform and what the conditions are required for the controlling devices to run the control panel.

In the implementation, we will use an open source software package called MRBS (introduced in section 2.3.3) to act as our presentation booking and management software. MRBS already has a template used to check for room availability, booking of a room, and canceling a meeting. However, in order to embed into our IPS, we need to add an

authentication mechanism for presentation speakers and to allow these speakers to upload their presentation file (s). Both mechanisms will be introduced in sections 3.3.1 and 3.3.2. In section 3.1.2, we will describe how to configure MRBS for using in the lab environment at Wireless@KTH. These implementations are meant to integrate the separate parts to create a smart management system for presentation booking and presentation file management.

In order to implement the presentation server, we need to build a subsystem to receive control signals from the control panel displayed on the presenter's device (such as a PDA) and send these signals to another subsystem running on the presentation server to actually execute the user's commands. The details of using a wireless interface to communicate with the presenter's PDA as will be introduced in section 3.1.1. Additionally, we need to transfer the uploading files from the context server to the presentation server (the details of this are presented in section 3.3.3).

Another important implementation part is the implementation of a control panel using an application running on a PDA to send commands to the presentation server. The details of the prototype's design are given in section 3.3.4. The goal for this subsystem is to provide an effective way for the user to control their presentation using a handheld device.

Based on the three guidelines for designing the IPS as presented in section 1.1, we need to design an easy to use interface which hides the complex internal technology details. Therefore, for the context server, we have used MRBS which already provides a friendly web interface. Thus we extend this web interface for our newly added parts on the context server. For the presentation control panel on the PDA, we flow the rules which we have discussed in section 2.4.1.2 and designed a simple control panel as the interface, further details are given in section 3.4.

3.2 Lab Environment

In order to set up the IPS, we need both the hardware and software. Figure 2.2 (on page 10) gave an overview of the hardware which was used. In the next subsection we will give a detailed description of this hardware. After this we will describe the software running on each machine, including its dependence on the underlying operating system.

3.2.1 Hardware built

As shown in figure 2.2, at least two computers are needed for the infrastructure: one acts as a context server and the other a presentation server. We can also use a personal computer as the user client, which is only used by the presenter to prepare his or her slides and then to upload these presentation files to IPS. In this thesis project a HP 6515b laptop was used for user client and a DELL GX620 desktop PC as the context server. However, for development purposes we used a self built computer based upon a Via Technology EN1200EG fanless mini ITX motherboard [67, 68, and 69, 70], a SATA disk, and a small fanless power supply as the

presentation server. The result is that the presentation server does not make any noise; hence it is very usable even in a small meeting room. Additionally, it takes very little power, so there is no reason not to leave it on at all times. We use a HP iPAQ 5550 PDA (see section 2.4.1.1) as the handset for the speaker to control their presentation. The PDA has a wireless internet access interface which enables wireless communication with the presentation server during the presentation. The context server and the presentation server are connected to the public internet. Additionally, we connect the VGA interface of the presentation server to the projector for display of the presentation. To support IPS, it was important to have an environment with both the wired and wireless internet access. In our lab, we use the KTHOPEN wireless network connection (i.e., an IEEE 802.11b WLAN) and a wired broadband internet connection (connected via 1Gbps Ethernet connections to a local ethernet switch). As mentioned in section 2.4.3, all of these devices were each configured with a single public IP address to avoid some issues due to firewalls/NAT.

3.2.2 Software

3.2.2.1 Operating Systems

In order to build a stable system environment and not be bother by the virus infections, Linux was chosen as the operating system for both the context server and the presentation server (openSUSE 10.3 for the context server and Fedora 7 for the presentation server). For the handset PDA, the originally installed system Windows CE is used as the operating system (as this the standard operating system which is preinstalled in this device).

3.2.2.2 Application Software

To create the context server, first MRBS was downloaded and installed. As mentioned in section 2.3.3, it is a free, web application using PHP and MySQL/pgsql for booking meeting rooms. MRBS is very reliable and very simple to use. Some common and useful features are [41]:

Easy to use	Very little or even no instruction is needed for users.
Flexible choices for users	Supports selectable DAY / WEEK / MONTH views, multiple languages settings and repeating bookings.
Easy to management	Supports multiple authorization levels (read-only, user, administrator) and various reporting options.
Stable	Ensures that conflicting entries cannot be entered and allow authentication using an existing user database (such as Netware, NT Domain, NIS, etc.) or via user name/password

Moreover, all the features are designed for day-to-day operation to enable an efficiency management and timely room booking/checking for users [41]. MRBS can be downloaded

freely from its official website [42]. However, to install the MRBS, Apache is needed to provide a web server, MySQL is needed to store data, and the PHP is needed to utilize dynamic web pages. Both Apache and PHP are open source software which can be downloaded freely from their official websites [43, 44]. MySQL provides free community downloads for developers via their official website [45]. Therefore, we downloaded and installed all of these software packages and configured them using the user guides in their support documents. Additionally, we used some specific configurations details provided in Mohammad.Z.Eslami's thesis [46]. Next, phpMyAdmin was installed to administer the MySQL via a Web interface [46]. Currently, phpMyAdmin can be used to create and drop databases, create/drop/alter tables, delete/edit/add fields, execute any SQL statement, manage keys on fields, manage privileges, and export data into various formats [46]. The biggest advantage of using phpMyAdmin is to make the administration of MySQL database much easy and more efficient through its graphical user interfaces. This interface is shown in figure 3.1.

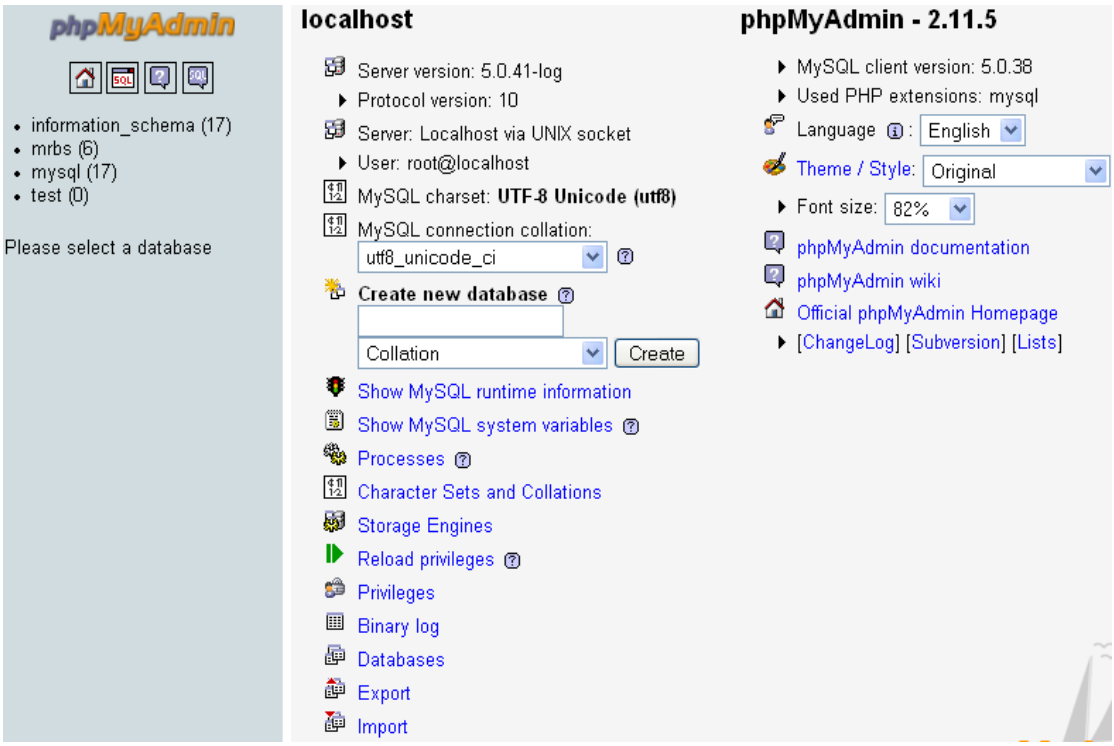


Figure 3.1 the phpMyAdmin Interface

In figure 3.1, the left column is a list of databases in the current MySQL server; the middle column shows the basic information about the MySQL server and all the executable operations on the MySQL database; the right column lists some information about phpMyAdmin and links to the help documentation. As we can see from this figure, there are four database in this MySQL server. The database *mrbs* contains six data tables and was created to support MRBS.

After the Apache+PHP+MySQL platform was successfully installed and MRBS was installed, the MRBS needs to be configured to be an IPS. Followed by the instruction in MRBS's

INSTALL file, MRBS was placed in the directory - */srv/www/htdocs/mrbs* on the openSUSE 10.3 system. Then, using phpMyAdmin, the sample table structures (including the area structure *mrbs_areas and* the room structure *mrbs_rooms*) offered by MRBS were loaded into the database *mrbs*. Two areas (*wireless@KTH1* and *wireless@KTH2*) were defined in table *mrbs_areas* to create the meeting room environment at Wireless@KTH. Four midsize meeting rooms (Grimeton, MINT, Horby, Motala) were put in the area *wireless@KTH1* and one big-size meeting room (Open Area) was put in the area *wireless@KTH2* in table *mrbs_rooms*. This classification based on the size of the presentation rooms gives the flexible choice on room booking. After doing this, the basic MRBS database configuration was finished. The resulting table structure of *mrbs* is shown in figure 3.2.

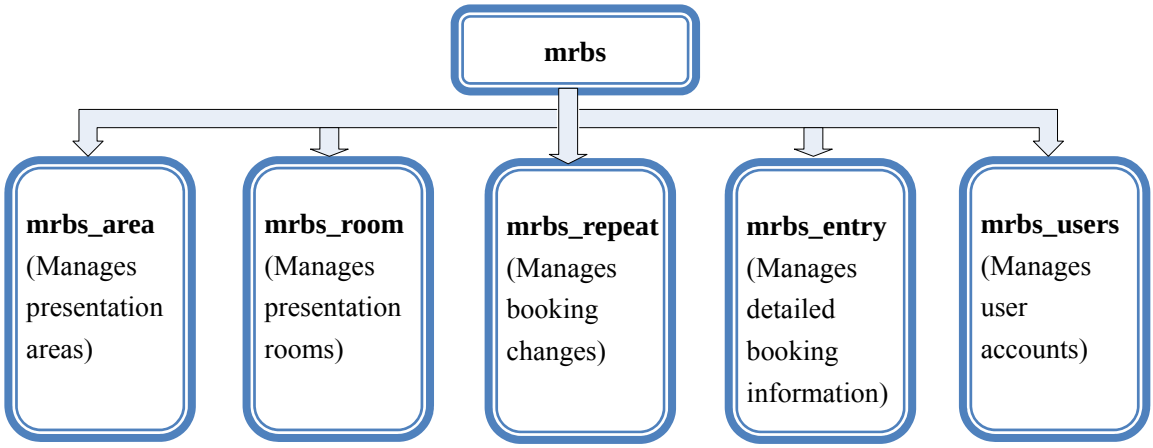


Figure 3.2 Database table structure of MRBS

Figure 3.2 lists five database tables now contained in database *mrbs*. To facilitate data management, the same prefix "*mrbs_*" is used to name the data tables. Table *mrbs_area* and *mrbs_room* store the name and id of the presentation area and room. Table *mrbs_entry* manages the presentation booking information regarding the time, place, topic and brief description of the presentation. Table *mrbs_repeat* records any changes in the booking. And *mrbs_users* is used for managing user accounts which enables the system to grant different rights to different users.

The server's hostname, database name, login user name and password were defined in the configuration file of MRBS to set up the connection entry for the database. In the Site Identification field of configuration file, the "KTH School of ICT" was set as the company/department name. Now, everything is ready to run the MRBS. Figure 3.3 in the next page shows the MRBS home page.

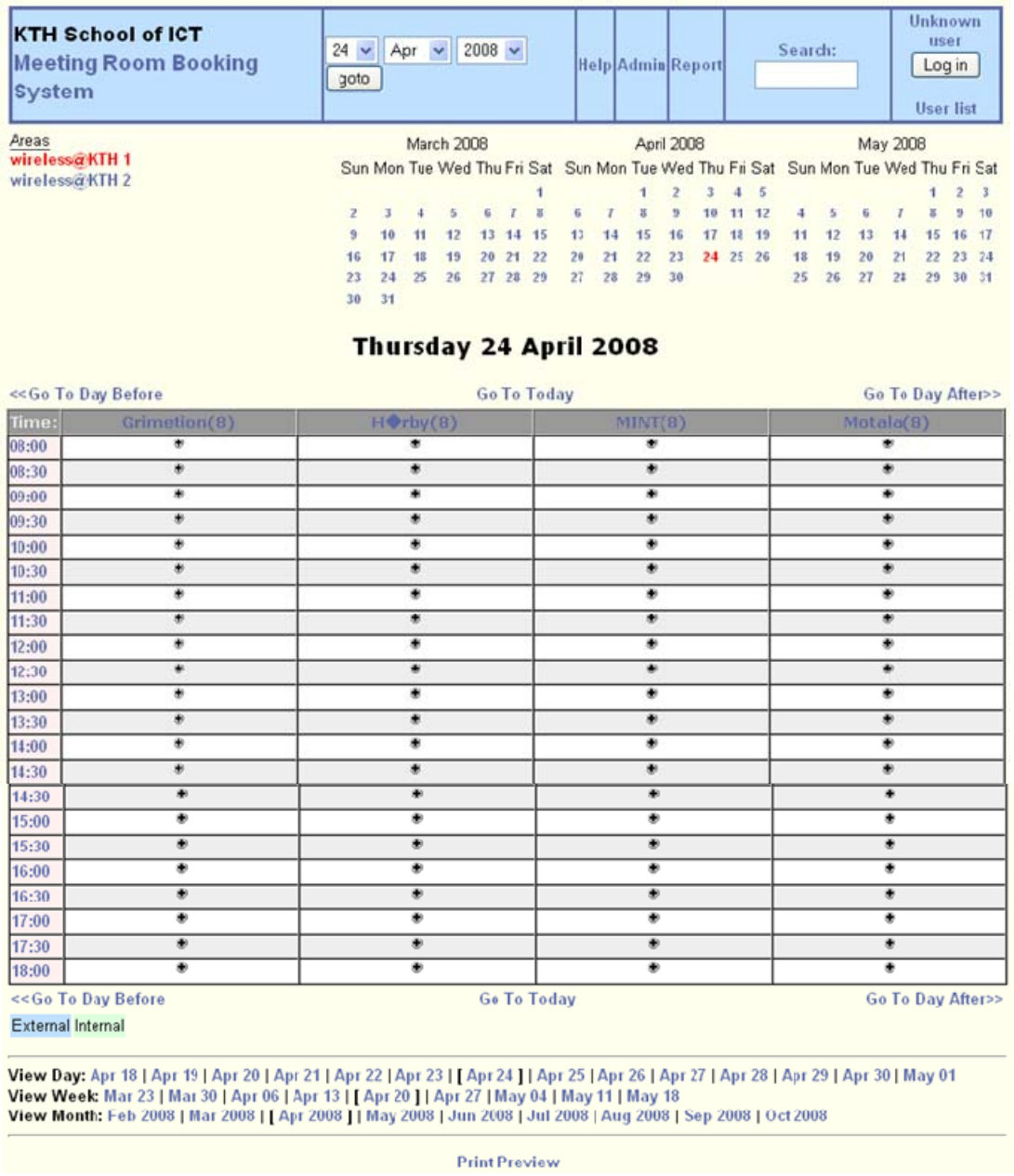


Figure 3.3 the main interface of MRBS

In figure 3.3, we can see a menu in the header of this page showing the title of the MRBS, the domain name (e.g. KTH School of ICT), the date, and the identity of the current user. Below the header menu, there are a list of areas and the calendars. If we select the first area, *wireless@KTH1*, all the rooms in this area will be listed in the middle table of this page. Thus, the administrator can book any presentation session by click ✚ under the room column.

The details of the booking information stored in *mrbs_entry* can be checked by clicking the topic of the presentation displayed in the main page to redirect the web browser to the *view_entry* page. Interested readers can try out the demonstration version of MRBS installed on its official website. [47]

Figure 3.4 shows an example of how a presentation named Introduction for KTH was booked in Grimeton (8) between 08:00 to 09:00. And figure 3.5 shows the main functions and the file structure of MRBS.

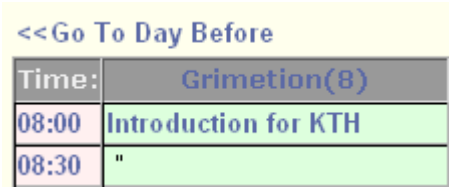


Figure 3.4 An example for room booking

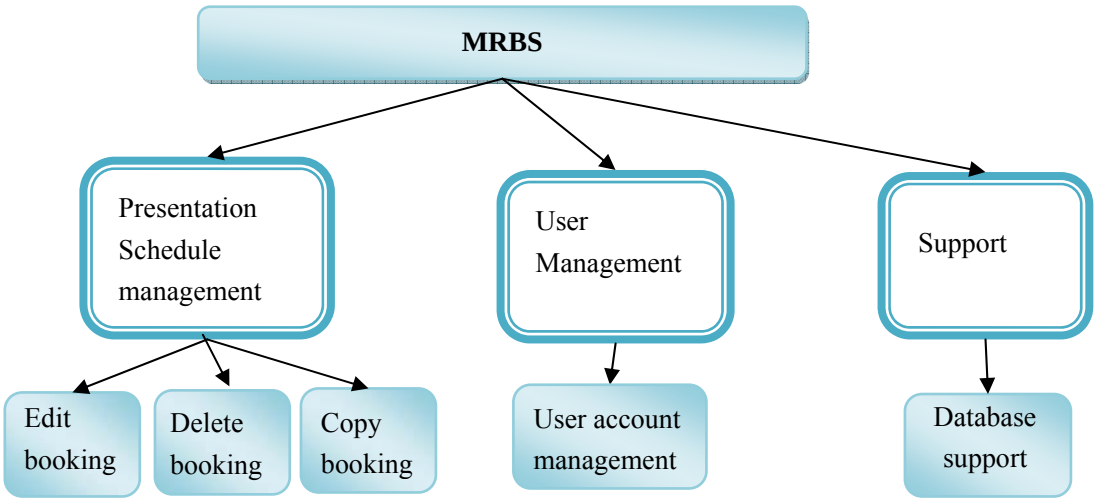


Figure 3.5 Functionalities and structure of MRBS

3.3 Prototype Design

Designing the prototype of IPS was divided into four steps. The first was to add authentication and file management modules into the MRBS to enable authentication of speakers (the presenters) and support file uploading for these speakers – as will be elaborated in section 3.3.1 and 3.3.2. The second was to facilitate an effective file sharing between the context server and the presentation server – as will be explained in section 3.3.3. The third step which is the core of IPS was to implement a presentation control module in the presentation server to receive and execute control commands from the user interface device used by the speaker. This part will be explained in section 3.3.4. The last step was to create a control panel to provide an interface for the speakers to control their presentation (further details of this will be presented in section 3.3.5).

3.3.1 Authentication of Speakers

For security reasons in IPS, the speakers need to be authenticated before they upload the presentation slides into the context server. The most common way to do that is to give the speaker a username and a password, so that a unique authentication pair (username, password) can be used to identify the speaker. However, the authentication pair (username, password) could be simplified to be simply a unique ID-code which is generated when an administrator books a presentation for a speaker. After the ID-code is generated, the administrator can send the ID-code together with the URL of uploading page to the speaker to enable the speaker to upload their file. Note that part of the reason to control uploads is to make sure that only the speaker can upload their presentation and that it is know who is responsible for the upload. Additionally, only the speaker should be able to present this material. The present system does not ensure that only the speaker can access this material - this is left for future work.

The ID-code should be randomly generated at the time of the room booking. Therefore, we created a random alphanumeric ID-code using a function randomWord (n), as shown below:

```
function randomWord(n){
var baseStr =
"ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789";
for (var i = 0, r =""; i < n; i++) r += baseStr.charAt ( Math.floor(Math.random() * 62));
return r;
}
```

In this function, a long base string is defined to include all the capital and small letters and the digits from 0 to 9. When the random number is calculated, the duosexagesimal system (a radix of sixty-two) is used rather than the common decimal system to allow a short ID-code, but reduce the probability of assigning duplicate codes to different speakers. Therefore, to get a random 10 character ID-code, the number “10” is passed as the parameter to the function randomWord (n), i.e., randomWord(10).

Thus the probability of assigning two users the same ID-code is reduced to as low as $\frac{1}{839299365868340224}$ which is a very low probability event. Thus the probability of assigning two users the same ID-code is lower than the probability of a first time golfer getting a “Hole in One” in golf☺ [71].

In this regard, the speaker can use this ID-code and the URL which they are provided with to login to the web page to upload his or her presentation slides. To support such an authentication process, a new module *upload_login.php* was added into MRBS. In the next section, another new PHP module named *upload_file.php* will be added together with *upload_login.php* to support the file uploading to the context server. The *upload_login.php*

looks very simple and only includes a single input text area and a submit button (shown in figure 3.6).

Figure 3.6 The authentication page for speakers

In addition to these PHP modules, a new table named *mrbs_speakers* was created in the *mrbs* database which contains the speaker's ID-code, the presentation topic, the filename of the presentation slides, the start time of the presentation, and the presentation room. Figure 3.7 shows the new table *mrbs_speakers* added into *mrbs* and the resulting database structure.

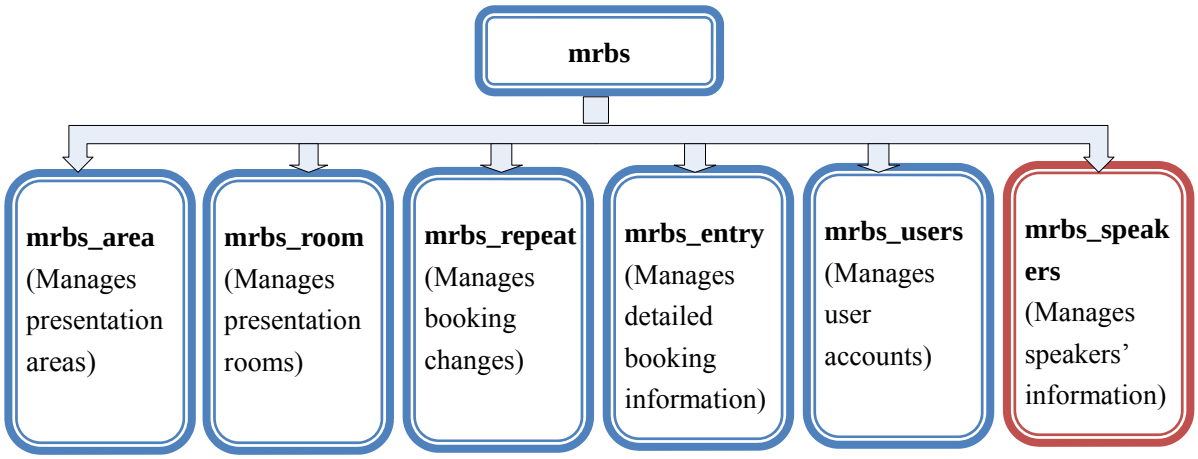


Figure 3.7 The new structure of the mrbs database

3.3.2 Uploading a File to the Context - Server

If the speakers are successfully authenticated, they are allowed to upload their presentation slides. The original MRBS does not support file uploading. Thus, we need the extend it to support file uploading. Figure 3.8 shows a HTML form generated by *upload_file.php* as viewed in a web browser. This form enables speakers to select which files from their own machine to upload.

Figure 3.8 HTML form for selecting and uploading files

After the speaker clicks "Upload Files", the data will be posted to the server and the speaker will be redirected to *upload_file.php* to receive feedback about their upload.

In *upload_file.php,* the $_FILES array is used to collect the information about the files. There are several elements of this array that are necessary to understand [48, 49]:

- **upfile** – it is the reference we assigned in our HTML form. It is included to tell the $_FILES array which file we wish to manipulate. This can be any file name.
- **$_FILES['upfile']['name']** - The original file name on the client's machine.
- **$_FILES['upfile']['tmp_name']** - This contains the path to the temporary file that resides on the server. The file should exist on the server in a temporary directory with a temporary name.
- **$_FILES['upfile']['type']** – The mime type of the file to be uploaded.
- **$_FILES['upfile']['size']** – The size, in bytes, of the uploaded file.

A new directory called "*uploadfile*" was created under the path */srv/www/htdocs/mrbs/files*. The presentation files will be saved in this directory. The PHP function below shows how to get the path as well as the full file name of the uploading files. [50]

```
//Get the Extension Name of the uploading file.
$exname=strtolower(substr($_FILES['upfile']['name'],(strrpos($_FILES['upfile']['name'], '.') +1)));

//Generate the full name of the uploading file together with its storage path
function getname($exname){

    $dir = "../uploadfile/"; //new directory that the files will be stored
    $i=1;

    while(true){
        if(!is_file($dir.$i.".".$exname)){
            $name = $_FILES['upfile']['name'];
        }
        break;
    }
    $i++;
      }
       return $dir.$name; //the result contains the path and the full file name
}
//Get the full name of the file
$uploadfile = getname($exname);
```

After we have successfully uploaded the file we need to validate the file uploads. When the *upload_file.php* is executed, the uploaded file is placed in a temporary storage area named *uploadfile_temp* on the server. If the file is not moved to a different location it may be destroyed. To solve this problem, *move_uploaded_file ()* is used to check that whether the uploaded file has been successfully moved to a designated location - */srv/www/htdocs/mrbs/files*. If the file is valid and been successfully moved to the designated

location, a successful upload message will be generated and the uploaded file name will be stored into the database. Otherwise, a failed upload message will be generated. The function below implements this process.

```
if (move_uploaded_file($_FILES['upfile']['tmp_name'], $uploadfile)) {
    echo "<h2><font color=#ff0000>Uploading success\uff01</font></h2><br>
    <br>";
    $sql="UPDATE $tbl_speakers SET fileName= '$filename' WHERE
                $tbl_speakers.IDcode='$IDcode' LIMIT 1";
    if (sql_command($sql) < 0) return 0;
}
else {
    echo "<h2><font color=#ff0000>Uploading failed\uff01</font></h2><br>
    <br>";
}
```

However, a key decision should be made with all uploads: keep the file or throw it away. A file might be thrown away for many reasons, including: [48, 49]

- The file is so large that you do not want to keep it on your server.
- The file type they uploaded is a file type that you do not want, for example, an executable file - (.exe).
- An error happen during the uploading which causes the server can not keep the file(s).

To solve these problems, a maximum data length is defined in *upload_file.php* to limit the file size. Upload errors will be returned via *upload_file.php* if they happen during the uploading or if the user attempts to upload an executable file (or other type of prohibited file). The source code for this can be found in Appendix A.

3.3.3 File Sharing between Context and Presentation Servers

In sections 3.3.1 and 3.3.2, we saw that how the presenter is authenticated and how their files could be uploaded from their own computers. However, the uploaded files need to be shared by the context server and the presentation server; otherwise the speaker can not present and control their presentation via the presentation server.

Initially, we considered having the context server send the presentation slides to the presentation server using a FTP file transfer. However, we found that the speakers may upload their presentation slides anytime before the presentation. Therefore, it is very hard to decide when this FTP file transfer should be executed (e.g. one hour before the presentation starts? or half an hour before?). Additionally, if the speaker re-uploads their files multiple times, then the FTP transfer will need to be repeated. For these reasons, it is better to utilize file sharing rather than FTP to transfer the file from the context server to the presentation server. To implement this we utilize an NFS (Network File System) between these two computers; thus the file is available to the presentation server as soon as it is uploaded to the

context server. Note that if the speaker's presentation is re-scheduled to another room, the presentation is easily available to the presentation server responsible for this other room. There may be problems when all of the presentations in the different rooms start at the same time as there potentially could be correlated load on the NFS file server. However, we leave the investigation of this and the solution to this to future work.

NFS allows a client computer to access files over a network as easily as if the files were stored on its local disks. [51, 52] The NFS protocol is specified in RFC 1094, RFC 1813, and RFC 3530 [52]. RFC 1094 defines version 2 of the protocol which just supports only UDP communication [53]. In RFC 1813, the support for TCP was added as a transport protocol for NFS; along with other improvements [54]. In April 2003, IETF released version 4 of the NFS protocol in RFC 3530 – this specifies various performance improvements, mandates strong security, and introduces a stateful protocol [51]. In IPS, the context server acts as a NFS server and the presentation server as a NFS client. So that any files uploaded to the context server can be immediately available to the presentation server. The choice of the context server rather than the presentation server as the NFS file server was made because this facilitates the presentation being re-booked to another room (and therefore another presentation server). However, for performance and also reliability reasons the opposite choice should be made (see section 6.2.5 for more discussion of this as potential future work).

3.3.3.1 Setting up a NFS server

Setting up a Linux NFS server on the context server was straight forward. However, there are a few security issues worth examining: especially regarding the firewall configuration [55, 56]. In this regard, we are going to place the NFS server behind the built-in IP-tables firewall. This also facilitates the future change to place the presentation server only on the internal network, with only the user authentication and file uploading functions available via the public Internet.

First, the NFS configuration files need to be set up. Typically, there are three main configuration files that need to be edited to set up a NFS server. They are */etc/exports*, */etc/hosts.allow*, and */etc/hosts.deny*. */etc/exports* is necessary for NFS to work. While */etc/hosts.allow* and */etc/hosts.deny* specify which computers on the network can use the NFS services (and other services) this NFS server [56].

- /etc/exports:
  This file stores the information about your NFS shares on your server, specifically: the name and path of the folder to share, the IP address (es) of hosts to be allowed to access your shared folder, and the rights permitted to your shared folder. [55, 56] Therefore, for IPS, this file is configured as shown below:

```
/srv/www/htdocs/mrbs/files vid1_IP (rw)
```

The first column of the configuration contains the full path on the NFS server that you want to share; the second column is the IP address of the presentation server (NFS client) you are sharing to this directory with. The letters in parentheses specify the privileges being granted to this client to access the shared folder. In this example, */srv/www/htdocs/mrbs/files* is shared to the presentation server with the IP address *vid1_IP*, and this client has *read-and-write* privileges on the shared folder. (Writing privileges on the folder ma to be able to edit the files during the presentation or delete the files after the presentation)

- /etc/hosts.deny
  This file describes the names of the hosts which are **not** allowed to access the shared folder. As the */etc/hosts.allow* (see the next section) overwrites this file, it is best to list all servers here. [55] Therefore, this file is configured as below:

```
portmap:ALL
lockd:ALL
mountd:ALL
rquotad:ALL
statd:ALL
```

  With */etc/hosts.deny* set like this and without a */etc/hosts.allow*, no hosts are allowed to access the shared folder on the NFS server. [55, 56]

- /etc/hosts.allow
  This file describes the names of the hosts which are allowed to access the shared folder. The following lines were added to allow the presentation server access to the shared directory on the context server:

```
portmap:vid1_IP
lockd:    vid1_IP
rquotad: vid1_IP
mountd: vid1_IP
statd:     vid1_IP
```

Then, we need to set up NFS to work from behind an IP-tables based firewall.

- /etc/sysconfig/nfs
  This file controls the ports which NFS is going to use. The following lines show how we configured it:

```
LOCKD_TCPPORT=48620
LOCKD_UDPPORT=48620
MOUNTD_PORT=48621
STATD_PORT=48622
RQUOTAD=no
RQUOTAD_PORT=48623
```

This configuration does not create a security risk since the context server is behind a firewall and NAT; and the firewall does not open these ports to external hosts.

After that, a folder named *tool* is created in the root directory. Several scripts are placed inside this folder. The names of these scripts and their functions can be found in Appendix D. To get NFS working, you invoke the *nfstart* script by typing:

```
[root@ccsleft tool]# sh ~/tool/nfstart
```

Then, if it is successfully started, the feedback will looks like lines shown below.

```
Starting NFS services:                                    [ OK ]
Starting NFS daemon:                                      [ OK ]
Starting NFS mountd:                                      [ OK ]
Mounting other filesystems:                               [ OK ]
```

Actually, if the context server were configured with two different network interfaces and one were connected to the Internet and the other connected to an internal network (where the presentation servers were placed), that we would not need these scripts, but could simply configure the NFS service to only operate via the internal network -- thus the system would manage the opening of the appropriate firewall ports (as necessary).

3.3.3.2 Setting up a NFS Client

The NFS client is configured on the presentation server which runs Fedora 7. Before setting it up, it is necessary to check whether the *mount* program is up to date and the *nfs-common* is installed. In Fedora, all the system packages can be installed or updated through the Package Manager or installed by *apt-get install*.

- Mounting the remote directories

The first step is to mount the remote shared directories of the NFS server [57, 58]. The generic command to perform this is: *sudo mount server.mydomain.com: /Serverfiles /Clientfiles*. Thus, in our specific case, the mount command looks like this:

```
root@vid1:~# mount ccsleft_IP: /srv/www/htdocs/mrbs/uploadfile /usr/local/share/files
```

This means that the mount point on the NFS client is */usr/local/share/files* and the shared folder of the NFS server (with IP address ccsleft_IP) is */srv/www/htdocs/mrbs/files*. Thus whatever the speaker uploads s their file to the shared directory it will be immediately available to the presentation server.

Next, we restart the portmap and nfs-common services:

```
root@vid1:~# /etc/init.d/portmap restart
root@vid2:~# /etc/init.d/nfs-common restart
```

- Mounting NFS file systems at boot time

After the services are successfully restarted, the NFS file systems should be set up to be mounted at boot time. The NFS file systems can be added to your */etc/fstab* file on the presentation server in the same way as the local file systems, thus they will be automatically mounted when your system starts up [57, 58]. This can be done by adding the following line to your /etc/fstab file with an editor:

```
ccsleft_IP:/srv/www/htdocs/mrbs/uploadfile /usr/local/share/files nfs rsize=8192, wsize =8192, timeo=14, intr
```

The rsize and wsize mount options specify the size of the chunks of data that the NFS client and server pass back and forth to each other [57]. While “nfs” specifies the file system type.

Now NFS file sharing has been successfully configured between the context server and the presentation server. In other words, the speaker can freely upload their presentation slides anytime before the presentation and those slides will be available to the presentation server.

3.3.4 Presentation Control Module in the Presentation Server

The presentation control module consists of two programs running on the presentation server - a client part (we call pres-client.cgi) and the server part (called presentation-server.cgi). The pres-client.cgi is responsible for receiving commands sent by speakers from their control panel and sending it to the presentation-server.cgi. The presentation-server.cgi receives the commands sent from the pres-client.cgi and executes the command.

Here, for security reasons, the speaker is required to authenticate using by entering his/her ID-code. The speaker will only get a control panel if he/she is successfully authenticated. After this, MRBS will redirect the speaker to the control panel to which it passes as environmental values - the file name of the slides and the ID of the presentation room (we will explain the details regarding the control panel platform in next section). When the speaker wants to initiate the presentation, he / she presses the button “INIT” on the control panel of the handheld device (or any other device). The pres-client.cgi opens a socket, puts the command together with the presentation file name into the packet and sends it to the presentation-server.cgi. Then, after the presentation-server.cgi receives this packet, it first

extracts the file name and executes the commands necessary to initiate the presentation; in this case we have used the OpenOffice impress program [72]. Consequently, the presentation files stored in the shared folder of the presentation server will be opened. Then the projector displays the slides on a big screen. Next, the speaker presses the button “Full Screen” on the control panel. The pres-client.cgi sends the command and the presentation-server.cgi receives and executes the command, so that the slides will be displayed in full screen style. Similar operations occur when the speaker presses the “Previous_Slide” and “Next_Slide” to navigate between slides and “Stop Full Screen” to exist the full screen display mode. A difference between “INIT” and the other operations are that the filename is only transmitted with the “INIT” command and pseudo key operations are needed in the following operations (as will be explained later in this section). Figure 3.9 describes the main process of how the control panel, the pres-client.cgi and presentation-server.cgi work together to render the presentation.

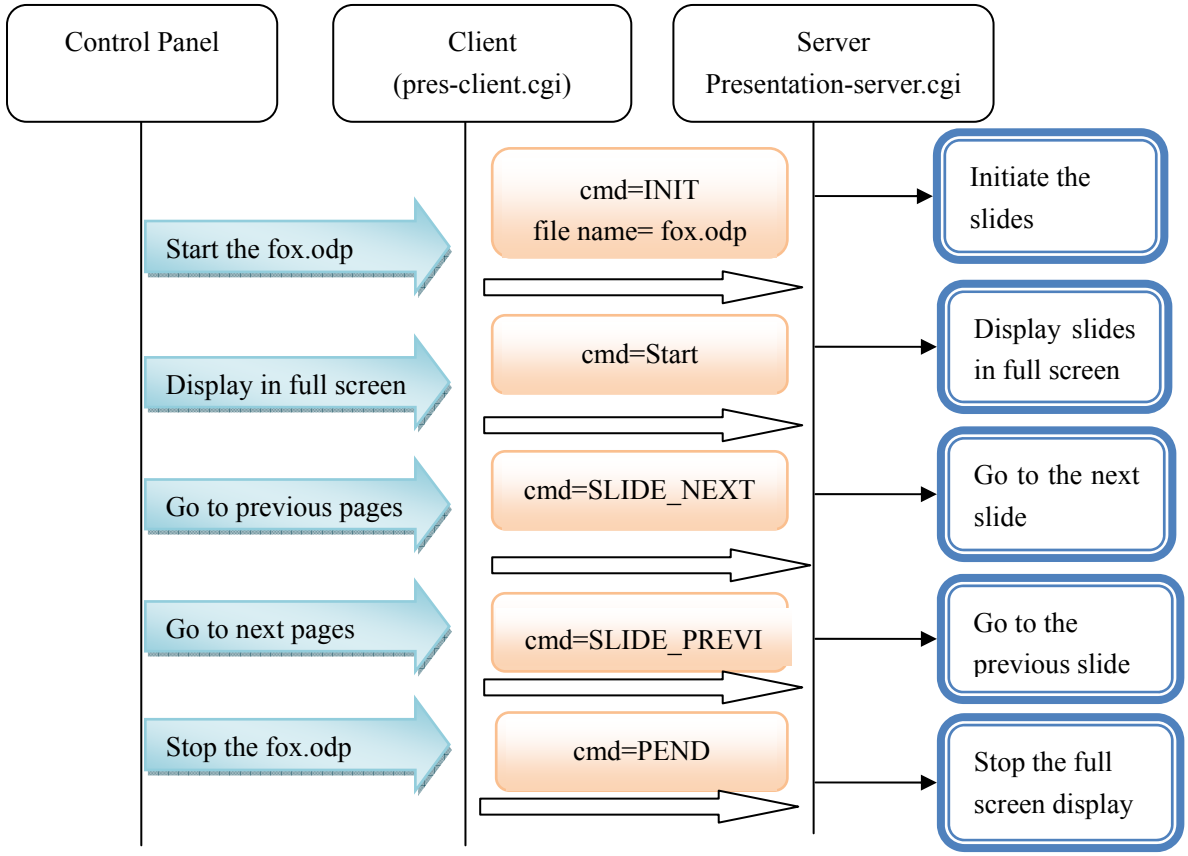


Figure 3.9 Work processes of the pres-client.cgi and the presentation-server.cgi

Figure 3.9 shows that the pres-client.cgi implements the communication between the control panel and the presentation-server.cgi. The control panel provides the human interface to the speaker and offers only the operations which the speaker can perform. While the presentation-server.cgi is only responsible for executing these commands and does not need to provide a human interface. The main tasks of pres-client.cgi are translation (translating the request from the control panel into executable commands) and transfer (transferring the

command to the presentation-server.cgi). The partitioning of the system in this way was made to facilitate replacing the control panel and client with other programs, for example, to allow a web based interface to be used on a laptop or a WAP (or web) based interface via a cellular phone.

The communication between the control panel and the client utilizes CGI (Common Gateway Interface) web interface. CGI is a standard protocol for interfacing external applications with information servers, commonly Web servers [62, 63]. CGI defines how the information about the web server and the request is passed to the command in the form of arguments and environment variables and how to execute the command to return the output [63]. For example, when the speaker wants to initiate their presentation, he or she presses the button "INIT" on the control panel. Then, the pres-client.cgi will be invoked. The code below shows how the pres-client.cgi is invoked through the URL behind the "INIT" button. The command "INIT" and the file name "wisdom.odp" are transferred to the pres-client.cgi as environment variable after the "?" mark in the URL.

```
<a href=\"http://vid1_IP/cgi-bin/pres-client.cgi?command=INIT&name=
wisdom.odp\">INIT</a>
```

The communication between the client and the server are conducted through UDP (User Datagram Protocol) sockets. The UDP processes in the pres-client.cgi and presentation-server.cgi are shown in Figures 3.10(a) and 3.10 (b) [59]. The code is shown in Appendix C.

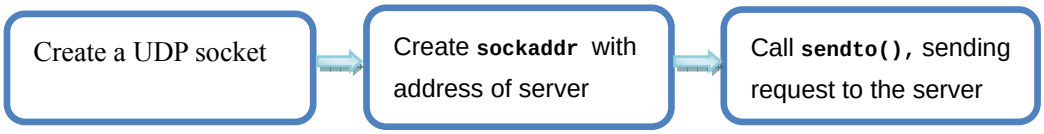


Figure 3.10 (a) Creating a UDP socket in the pres-client

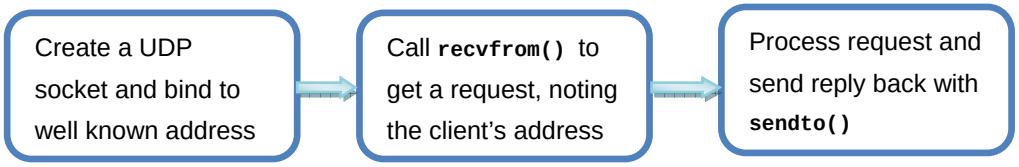


Figure 3.10 (b) Creating of the UDP socket in the

If the UDP packet is successfully received by the presentation-server.cgi, the server will output the message (including the command, the file name, the source address of the packet coming from and the data length) to a process which is executing the OpenOffice impress applications (this is the binary program "ooimpress"). As noted before file name is only transmitted with "INIT" command. The presentation-server.cgi will initiate the OpenOffice's impress application by invoking the ooimpress process located in - */usr/bin/ooimpress* and open the file containing the presentation. The C code below shows (logically) how the presentation-server.cgi starts this presentation (contained in the file "wisdom.odp").

```
sprintf(CMD, "/usr/bin/ooimpress -open /tmp/wisdom.odp"); /*command to start ooimpress */
```

To implement the remaining commands (such as *Full Screen*, *Slide_Next*, etc.) the presentation-server.cgi will invoke the function *sendKey (KeySym keysym)* to simulate the key-press events. Here, X11 functions are used to provide the basic framework, or primitives, for the implementation on a Linux system, since the X11 windowing system is being run on the display of the presentation server (a similar implementation mechanism can be used by sending messages to windows on a Microsoft Windows system). X11 implements the functions to draw and move windows on the screen and to interact with a mouse and/or keyboard [60, 61]. The X windowing system uses a client-server model. In our case, the X server is the graphical interface running on the Linux system running on the presentation-server.cgi and the X client here is the OpenOffice ooimpress application which has a window open on the screen. Take the command "Full Screen" as an example. When the presentation-server.cgi receives the command, to perform the actions which would be invoked by a real key – press event, it invokes the function *sendKey (XK_F5)* to do a pseudo key-press event to simulate a real F5 key-press event (The function key F5 is used by the impress program to cause the program to use full screen mode.). On X server side, this pseudo key-press operation is no different than a real F5 key-press, thus, the correct action will be performed by the X client (ooimpress). Figure 3.11 shows how this pseudo-operation is performed and how the X server passes this key-press event to the X client.

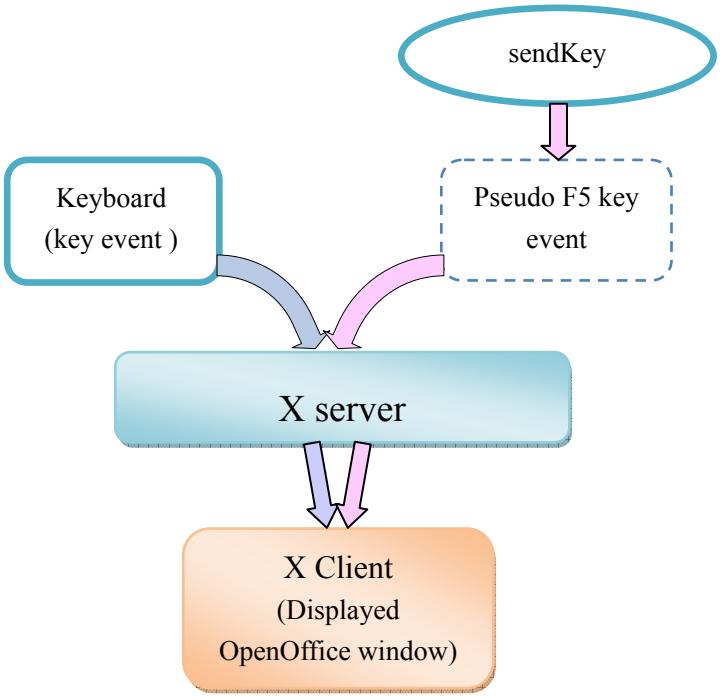


Figure 3.11 the Pseudo operation in sendKey function and the output

Figure 3.11 clearly illustrates that the Pseudo key event generates the same input as the real key event does to X server and for this reason, the output and the operation in the

presentation (indicated in figure 3.12 as "the slides") window will be the same as the real key event would cause. This pseudo operation is actually divided into two steps: key-press simulation and key-release simulation. However, before the key-press simulation, a local display has to be opened as an interface to X client to get the results returned from key – press and release simulation. Then, to target the window of which has ooimpress running in it, the X system function – *XgetInputFocus* is used, which returns the current window ID by keeping track of the focus window and the current focus status. When the OpenOffice is initiated, the window running the ooimpress is automatically set as the top-most window. And the *XgetInputFocus* is programmed to get such ID of the top-most window. After the correct ID is figured out, we set the event type as *KeyPress*, the event status as 0 and the event key code as the F5 code. Finally, the key-press event is invoked through the function *XSendEvent.* In this manner, a key-press event is simulated. For the key-release simulation, we simply change the event type to *KeyRelease* and again invoke *XSendEvent*. Figures 3.12(a) and (b) show the processes of the key-press and release simulation.

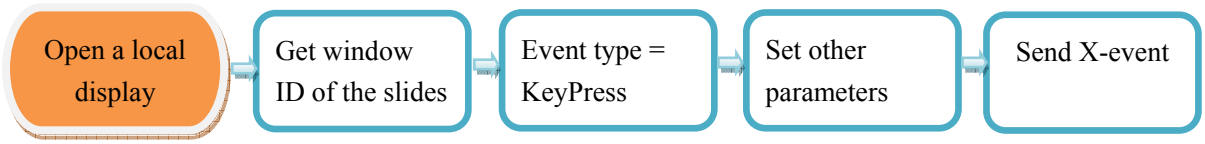


Figure 3.12 (a) Key-press simulation process

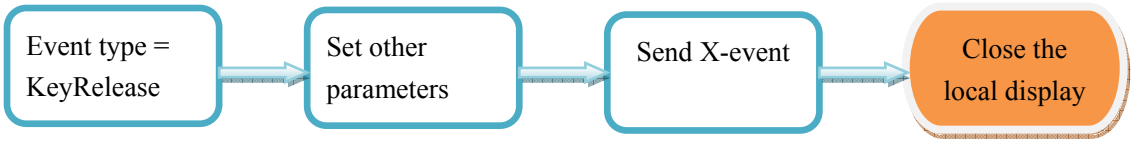


Figure 3.12 (b) Key-release simulation process

3.3.5 Presentation Control Panel

Initially, we wanted to make the control panel run only on a handheld device (e.g. PDA). Later we recognize that not all the speakers are willing to use handheld devices, especially for those who are not familiar with a PDA or have not yet got used to using PDA. Our concern was motivated by our earlier experiment using a smart phone to control a presentation (see section 2.4.1.3). To avoid to this negative effect and to give more flexible choices of control panel to speakers, we decide to make our presentation control panel machine-independent, which means that the control panel can run on different platforms

A cross platform control panel should operating system independent. Therefore, building the control panel using web pages is a good choice; thus the speakers can use any kind of computer which has a web browser (e.g. PC, laptop, PDA, smart phone, etc.) to control the presentation as long as they can access the Internet. Nevertheless, simply making a web interface is not enough. It is very important to have suitable server support to execute the commands sent by the web interface. Therefore, we decided to use CGI. In IPS, the control panel is built dynamically using web pages which are written in PHP. The presentation control module is written in C is compiled as a CGI module (pres-client.cgi and

presentation-server.cgi respectively). To make the CGI module executable on the presentation server, a directory named *cgi-bin* was created under the path */var/www/cgi-bin*. Then the CGI module was put into this directory. Now, the CGI module can respond to the commands sent from web browsers and returning a new web page. Every time a command is received by the pres-client.cgi, it analyzes the request, extracts the file name, opens a UDP socket and sends a packet containing a command to presentation-server.cgi. After received the packet, the presentation-server.cgi will execute the command and return the appropriate output. Figure 3.13 shows the web interface for Lidan when using her Grimeton control panel - *Grimeton.php*. The code for *Grimeton.php* can be found in Appendix B.

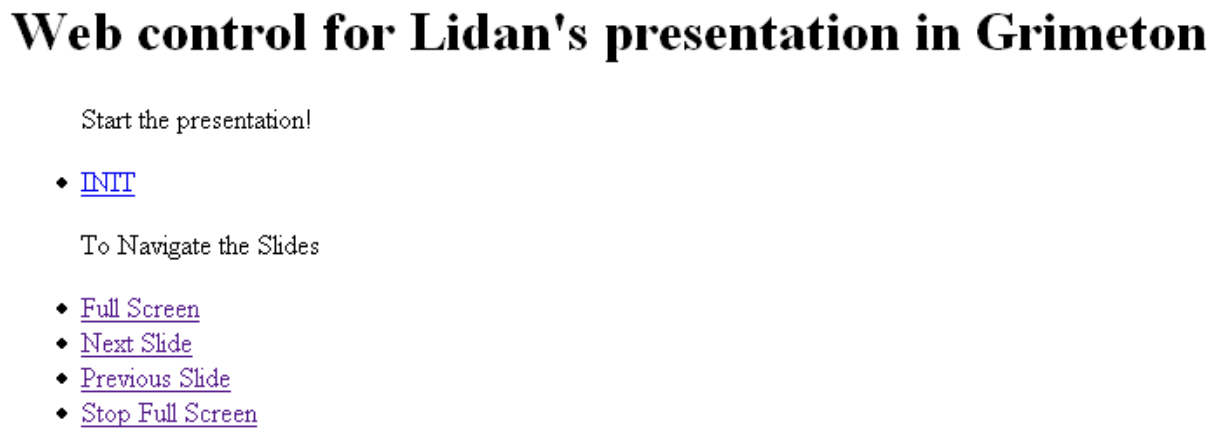


Figure 3.13 Control panel of IPS

The code below is part of the web page of the control panel. It shows the web interface between the command button on the web page and the executable CGI module in *cgi-bin* (here it is pres-client.cgi).

```
<a href=\"http://vid1_IP/cgi-bin/pres-client.cgi?command=INIT&name=
wisdom.odp\">INIT</a>
```

From the code above, we can see that the pres-client.cgi is invoked by the URL behind the INIT command button in the web pages. The presentation_server_IP is the address of the presentation server located in room Grimeton. Likewise, the other command buttons will invoke the pres-client.cgi in the same way as the INIT command does.

Each of the meeting rooms which are equipped for presentations has a control panel configured specifically for this room. While these control panels are associated with different rooms, the web pages look much the same as Grimeton's shown in figure 3.13, but the web interfaces between the control panel and the CGI module differ due to the different IP addresses of the different presentation servers. For example, in *Motala.php* (the control panel of Motala), the web interface is shown below (here we use the symbolic name ccsmoto_IP as the IP address of the presentation server in room Motala).

```
<a href=\"http:// ccsmoto_IP /cgi-bin/pres-client.cgi?command=INIT&name=
wisdom.odp\">INIT</a>
```

Now the IP address of each presentation room is pre-defined in the web interface. In the table mrbs_spekaers (see figure 3.7), the room id and the presentation file name of each presentation are stored together with the ID-code of the speaker. This database and the pre-defined IP address in the web interface of a control panel enable IPS to redirects the speaker to the correct control panel for the room where they are presenting - along with the correct presentation file – once a speaker is authenticated.

Chapter 4: Testing and Improvements

In this chapter, we will first test the IPS in single speaker mode. This means that the context server works with only a single presentation server. Following this we will test IPS in multiple speakers mode; which means that many speakers present at the same time and the context server operates in parallel with different presentation servers. After that, several improvements are proposed to solve the problems detected from the system test.

4.1 Testing IPS in Single speaker mode

In our test environment, the context server is named CCSLEFT and has an IP address of ccsleft_IP. The presentation server in Grimeton is named Vid1 and has an IP address of id1_IP. To test the whole process of the IPS, the administrator books a presentation session on June 18th, 2008, between 8:00 and 9:00 in the meeting room Grimeton. The presentation topic is "*Introduction of Business Strategy*". After that, the session can be seen from the main page of the MRBS by visiting from internet. Figure 4.1 below shows the result.

Figure 4.1 Presentation session booking on MRBS

Then, the administrator sent the ID-code together with the URL of uploading page to the speaker. The speaker logged in to the MRBS and uploaded his slides named "*Business_Strategy.odp*" to the context server using his own laptop. Due to NFS, files on the context server are available on the presentation server in Grimeton, i.e., presentation slides are in the shared folder (*/usr/local/share/files*) as soon as the speaker uploaded them. Figure 4.2 shows this.

```
[root@vid1 ~]# cd /usr/local/share/files
[root@vid1 files]# ls
Business_Strategy.odp  fox.odp  lidan.odp  wisdom words.odp
[root@vid1 files]#
```

Figure 4.2 slides exist in presentation server after the speaker uploaded them

Before the presentation starts, the administrator initiates the presentation-server.cgi on the presentation server. The code below shows the initiation command.

```
root@vid1:~# ./presentation-server.cgi
```

For our first test, the speaker used an HP iPAQ 5550 PDA as the handset (as described in section 2.4.1.1). He first logs into the control panel by entering his ID-code. Figure 4.3 shows the log in window.

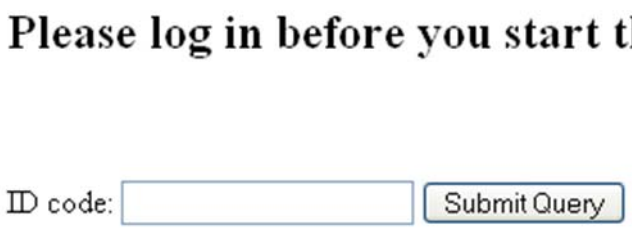


Figure 4.3 log-in window for speakers to open the control panel

We assume that the speaker was successfully authenticated. His presentation room is Grimeton. So the control panel of Grimetion.php was sent to his PDA. The speaker then presses the command "INIT" on the web control panel. Subsequently, the OpenOffice's impress in the presentation server was initiated and the speaker's presentation file name "*Business_Strategy.odp*" was opened. The presentation-server.cgi received the correct transmission information in the UDP packet sent from pres-client.cgi also running in presentation server. Figure 4.4 below is a screen shot about the debugging output from the presentation server.

```
[root@vid1 cgi-bin]# ./presentation-server.cgi
Agent server initiated
Received packet from 130.237.15.224:32775
Data: PNAME/usr/local/share/files/Business_Strategy.odp
String length=1024
Presentation name: /usr/local/share/files/Business_Strategy.odp
Received packet from 130.237.15.224:32775
Data: INIT
String length=20
```

Figure 4.4 Packet receiving from pres-client.cgi

In figure 4.4 we can see that the presentation server (Vid1) had successfully received the UDP packet and extracted the file name and storage path. Figure 4.4 also shows the IP address and the UDP socket port used for communication between the two CGI programs.

Next, the speaker pressed "Full Screen" on his control panel. The slides are now displayed in full screen size. Now the speaker presses the "Next_Slide" and "Previous_Slide", causing the system to present the next or the previous slide correctly. Finally, he pressed "Stop Full Screen". The presentation program exited full screen display mode. All the operations were successfully executed. Figure 4.5 is also a screen shot which shows the outputs showing the commands and their corresponding execution feedback.

```
Received packet from 130.237.15.224:32775
Data: START
String length=20
Received packet from 130.237.15.224:32775
Data: null
String length=1024
Received packet from 130.237.15.224:32775
Data: SLIDE_NEXT
String length=20
Received packet from 130.237.15.224:32775
Data: null
String length=1024
Received packet from 130.237.15.224:32775
Data: SLIDE_PREV
String length=20
Received packet from 130.237.15.224:32775
Data: null
String length=1024
Received packet from 130.237.15.224:32775
Data: PEND
String length=20
```

Figure 4.5 Commands received and the execution feedback by presentation-server.cgi

4.2 Testing IPS in Multiple Speaker Mode

As there are multiple presentation rooms in our locate, it is reasonable to test whether the IPS supports simultaneous presentations or not. So we equipped another presentation room (named Motala) with a presentation server for multiple-speakers test. The presentation server in Motala is called CCSMOTO and has an IP address of 130.237.15.252. We booked two presentations at the same time session, but in these two different rooms. Figure 4.6 shows the booking results.

Tuesday 24 June 2008

<<Go To Day Before Go To Today Go To Day After>>

Time:	Grimetion(8)	H◆rby(8)	MINT(8)	Motala(8)
08:00	Introduction for Grimeton	✚	✚	Introduction for Motala
08:30	"	✚	✚	"

Figure 4.6 two presentations booked in MRBS

Then, we uploaded two presentation slides (Grimeton.odp and Motala.odp) to the context server through MRBS. As expected, the files uploaded to the shared file folder in context server were visible in the NFS client's shared folder on both Vidl and CCSMOTO. After this, the CGI module in Vidl and CCSMOTO were initiated (note that in normal operation they would always be running). Next, two speakers use their laptops to log to IPS to get their respective control panels to make two different presentations in parallel. This result in both

presentations running on different presentation servers being initiated, the presentation files correctly opened, and correctly displayed. These results prove that the IPS can support multiple presentations simultaneously. However, no measurements were made as to the maximum number of presentations which could be carried out in parallel using the existing context server. This remains for future work as discussed in section 6.2.5.

4.3 Problems Detected and Improvements Made

During testing we found a problem: if the presentation server is abnormally halted or if the presentation-server.cgi is manually stopped without firstly closing the presentation file, the OpenOffice impress will generate an error window to ask if the presentation should be recovered when it is initiated again. The recovery message window is shown in Figure 4.7.

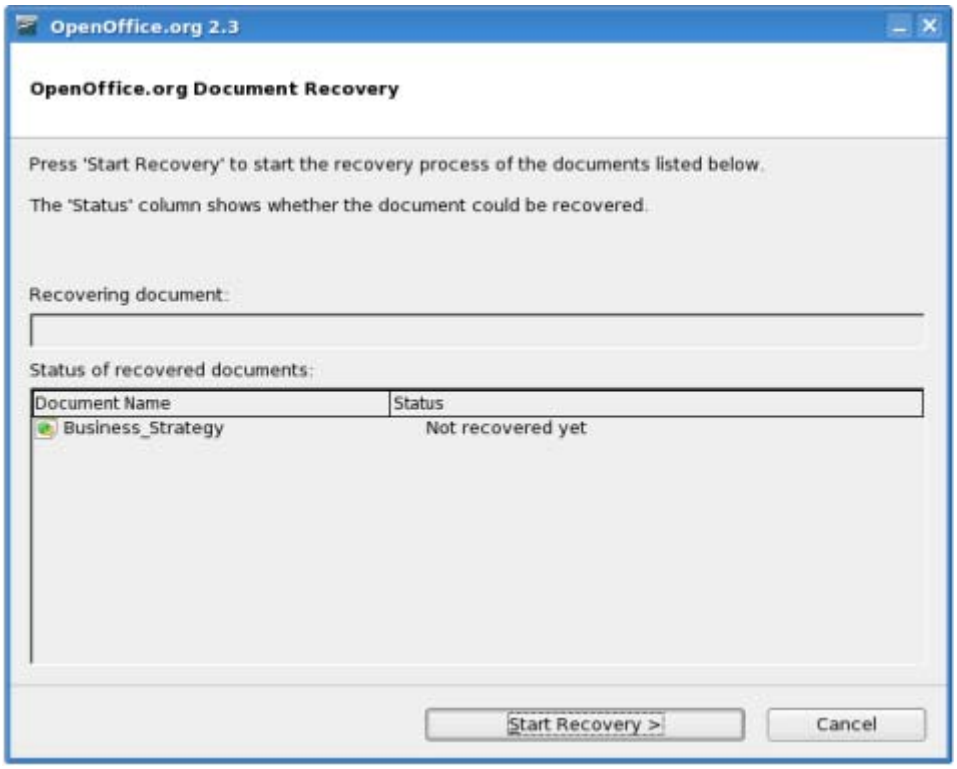


Figure 4.7 recovery window of OpenOffice slides

The reason comes from asynchronous close in parent and child processes in presentation-server.cgi. Normally, the presentation-server.cgi runs as the parent process. It forks a child process through the command *popen(CMD, "w")* to open the presentation slides when the OpenOffice is initiated. However, if the presentation-server.cgi is abruptly stopped, the child process will lose the communication pipe with the parent process and the marker bit in the OpenOffice presentation file will be not properly reset. Therefore, if the next time the OpenOffice is initiated, it wants to perform a recovery operation to set the marker bit properly. We can simply press the *enter* key to start recovery. After this when the OpenOffice is executed, it will be correctly initiated. There is an easy way to solve this problem: we add a command button “Enter” to recovery the OpenOffice slides when a recovery window comes out. Additionally, we add a command button”Close” on the control panel to let the speaker

exit and close OpenOffice. Figure 4.8 shows the new control panel. An alternative would be to automatically reset the marker bit, but we have not implemented this solution.

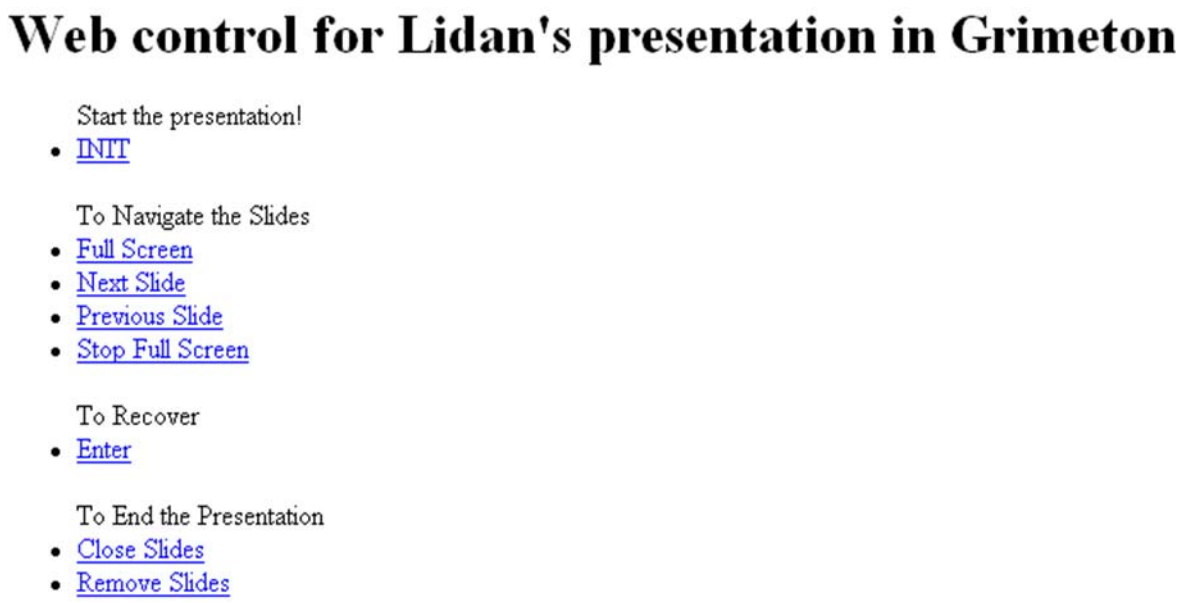


Figure 4.8 A new control panel

Actually, to simulate a key press event of "Enter" is not a problem. *sendKey(XK_Return)* can be used to execute pseudo key event "Enter". However, the key event simulation of "Close" is different from others in that it is a combination key event (Ctrl+Q). The key "Ctrl" should be pressed earlier and released later than the key "Q" does. To simulate this combination key event, we first perform a pseudo key press on Ctrl; then do a pseudo key press on Q; third do a pseudo key release on Q; lastly do a pseudo key release on Ctrl (show in figure 4.9 (a)). However, this did not work as we expected. We realize that the way we designed meets the time sequence of these four key events but did not show their logical relationships between each other. Therefore, the four events were executed one by one, but did not operate as a combination key event. Later we found a better solution by using an Xlib constant – ControlMask provides a combination key event. Here, we set the event.state of pseudo key-press event on Ctrl as ControlMask, now the logical relationship of those four pseudo operations is shown in figure 4.9 (b) below.

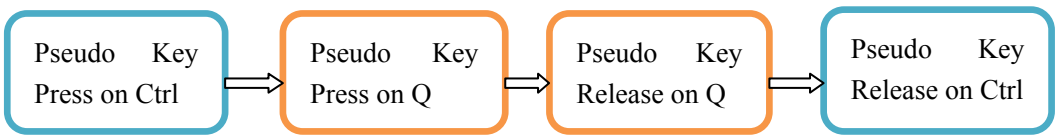


Figure 4.9 (a) Four pseudo key events

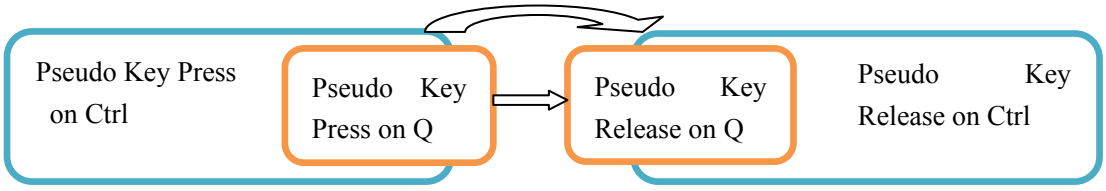


Figure 4.9 (b) Improved four pseudo key events

After making this change, we tested the improvements based in two scenarios: First, we manually stopped the presentation-server.cgi, then closed the OpenOffice presentation slides. After that, we started the presentation-server.cgi again and pressed “INIT” to initiate the OpenOffice. As expected, the recovery window appeared. So we pressed the newly added button “Enter”, subsequently the slides were opened correctly – as expected.

In the second scenario, we pulled the electrical plug out of the presentation server to simulate an unpredictable close-down of presentation-server.cgi. When we restarted the machine, ran the presentation-server.cgi and pressed the press “INIT” in the control panel again, we see that the presentation proceeds smoothly. This confirmed that our improvements work effectively and successfully!

Chapter 5: Evaluation

5.1 Achievement of the Goals

At the beginning of this thesis, three goals are presented regarding the development of in developing the IPS (see section 1.1). Now we examine to what extent IPS fulfilled these goals. The first goal was to build a context-aware environment so that if the speaker walks into the presentation room, the intelligent sensor should recognize him/her identity by sending an acknowledgement signal to the speaker on screen. However, this was not included in the development of IPS and is being addressed by Ren Xueliang in his thesis – "A Meeting Detector to Provide Context to a SIP Proxy" [64]. Our development of IPS presentation begins at the point is when the speaker starts the presentation, his/her presentation files should be already prepared on the presentation server using IPS control panel sent to him/her . This part of the project was completed. We also support the user's uploading of their presentation to the context server and making this file available to the presentation server. Another point is that the speaker only uses simple buttons on a web page to interact with the IPS. We have shown that a speaker just press the buttons set on their control panel and the presentation server reacts properly (see figure 4.8).

The second goal is that the user interface should easy and simple to use. In this regard, the tasks for the speaker to upload files and control presentation use a simple web based application and are available to any machine with network connectivity and a web browser. The speaker does not need to learn a new technology to use IPS (see figure 4.8).

Another critical point for an intelligent system is that the internal technologies should be hidden. Here NFS and CGI in IPS support this goal. Using NFS avoids needing to consider file transmission between the context server and the presentation server. While CGI supports a simple to create and simple to use web based control panel, giving the speaker flexibility in choice of which device they wish to use to control their presentation. MRBS hide the complex management of room bookings by offering an easy to use web based interface; thus making booking a presentation simple for both the administrator making the booking and for the speaker who will give the presentation.

In addition to these goals, we also added a security mechanism to IPS to authenticate the identity of the speaker both before their presentation and before permitting them to upload files. The details of this mechanism can be found in section 3.3.1.

5.2 The Architecture of IPS and Proposed Improvements

The new improved architecture of IPS is shown in Figure 5.1. The main functionalities of MRBS in the context server are shown in Figure 5.2.

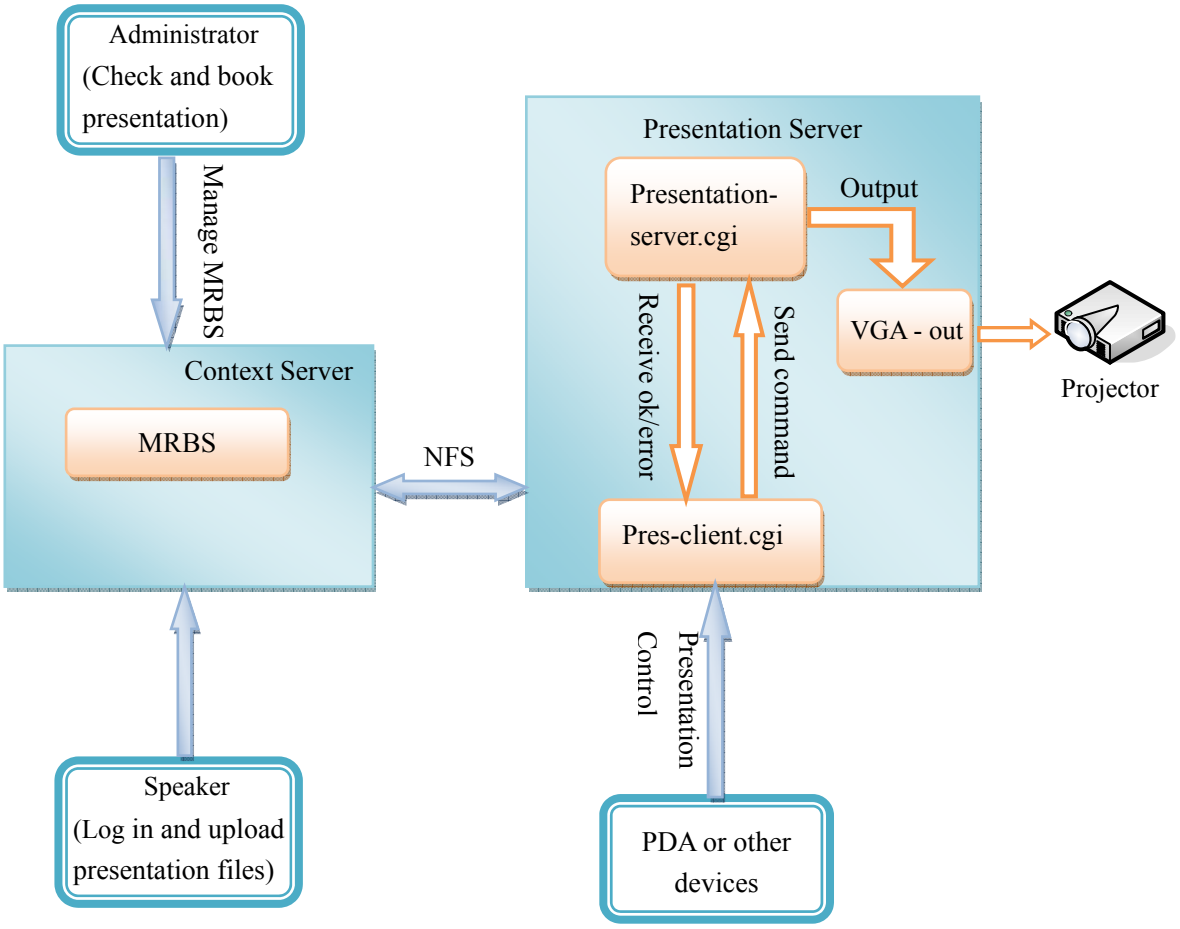


Figure 5.1 New Architecture of the IPS

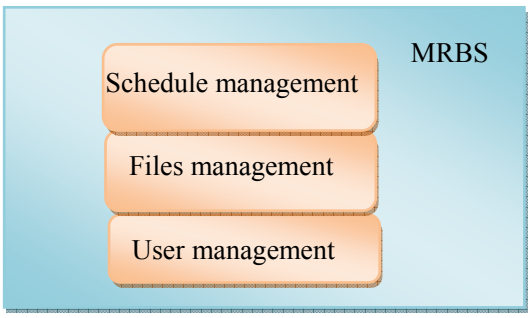


Figure 5.2 Management functionalities of MRBS in Context server

Figure 5.1 shows that there are four main changes in our new architecture of IPS compared with the originally IPS architecture (shown in figure 2.8). First we add an administrator role in IPS to book a presentation for the speaker and to send an ID-code and URL to the speaker.

Now the speaker only uses IPS for uploading their files and controlling their presentation. This allows better management of the MRBS database and limits the authority of administrator and speaker. For example, the speaker need not be an MRBS user – which avoids the administrator needing to add and remove lots of presenters to/from the authorized user database. Second, we simplify the functionality of presentation server by removing the client control module from the presentation server and use MRBS in context server to provide presentation management. This shifts the central server of IPS from the presentation server to context server. Additionally, it splits the presentation server into two parts one which should be local to the actual computer providing the presentation video (and possibly audio) and the other which could be run on another computer which has public Internet access (to receive commands from the user) and has internal network connectivity to the presentation server. The main functionalities of MRBS are shown in figure 5.3. However, the relationship between the context server and presentation server is one-to-many. If there are many presentation rooms, there is a presentation server in each room, but they are all managed by one context server. Third, we eliminated the need for FTP between the context server and presentation server and utilize NFS instead. Obviously, using a distributed file system such as NFS between these two servers is better than FTP file transfer – as it is both simpler and if properly configured safer. Moreover, NFS saves us from the complexity of the network firewall and NAT issues which would need to be addressed if using a FTP file transfer. Last but not least, we no longer restrict the speaker to control the presentation using only a handheld device (e.g. PDA). Now the speaker can easily choose which device they wish to use to control their presentation – all that is required is a web browser and internet connection.

5.2 Comparison with the Existing Method of Giving Presentation

We will assume that the existing presentation approach is to connect a laptop computer with the presentation software to the projector in the room. Here we assume the following configuration: a user laptop connected with an InFocus (LP 130) projector. This projector is typical of the type of device which is commonly used to display a speaker's slides as displayed on the user's laptop on a big screen. However, our IPS avoids the user needing to connect their laptop to the projector and the user can control their presentation using a device which need not contain any presentation software. Hence, the user can use a wide variety of devices to control their presentation. Therefore, we believe that IPS is more efficient and convenient to use. In table 5.1 below we use several metrics to measure the IPS performance and compare it with the typical means of making a presentation today.

Table 5.1 Evaluate the IPS

		IPS	Old Presentation System
Efficiency		Very efficient, do no need users to deal with any technology configuration before their presentation	Efficient to use, but requires the speaker to configure their computer's settings before a presentation
Performance	Time	Extra time is required when the user uploads their presentation file and is authenticated by the IPS.	Real time executive. Costs extra time to connect the user's laptop and the projector.
	Memory	128MB for running OpenOffice	128MB for running OpenOffice
Usability		Easy to use. Control panel and MRBS are web based applications. They are simple to understand and only require the speakers and administrators to know how to browse to a web page on to the internet.	Not so easy to use. Sometimes, it requires extra time for the speaker due to the need to adjust the resolution, timing, and sometimes select another screen than displayed on their laptops when connecting directly with the projector
Ease of development	Available software	OpenOffice and MRBS needed in the IPS are available for free. The Linux operating system is also free	Do not need any software to develop and configure the system, but must have the presentation software installed on their laptop.
	License	Do not need any license	Do not need any license
	Setting it all up	Require that the presentation server, the context server, and the projector keep running during the presentation time. The presentation-server.cgi should be initiated before the presentation.	Require that the projector should be started and adjusted well before the presentation.
Cost	Software cost	All the software needed is open source and free to use.	No software cost
	Hardware cost	At least two PCs (one is used for context server and the other is for presentation server.) and one projector.	One projector. Need the speakers to bring their own laptops.

From table 5.1, we can see that the IPS creates a comfortable and intelligent environment for people to use to give presentations. Therefore, the speakers need not bother their laptop configuration during their presentation. Additionally, the cost of development and use of IPS is very low. All the work is based on open source software and freely available for everyone.

More importantly, the IPS really provides an intelligent environment which hides the internal technology. However, IPS requires that there be network connectivity which the speaker can use to connect to the presentation (control) system.

Chapter 6: Conclusions and Future work

6.1 Conclusions

We have successfully implemented the complete architecture of an initial IPS. As shown in figure 2.2, our IPS now consists of one context server, at least one presentation server, one projector, and one handset for controlling the presentation (Note that with suitable network connectivity the control of the presentation could be done by any other computer with a web browser such as PC, smart phone, laptop, etc.). IPS supports not only presentation control but also room & presentation scheduling and files management. The intelligence comes through three aspects we developed in IPS. First, the MRBS provides a friendly web interface for the presentation administrator to book, cancel, and edit the presentation schedules. Second, MRBS together with NFS provide timely file sharing between the context server and presentation server which enables the speaker to easily upload his/her presentation files at any time before the presentation. When it comes to the presentation, the IPS opens the correct file after the speaker is authenticated. Third, the control panel operates across most platforms. The speaker can use any computer with a web browser to load the control panel as long as the machine can connect to the internet. Moreover, IPS has no requirements on the internet interface (other than the ability to support HTTP based interactions), so the network connection can be wired or wireless. The final configuration of IPS is shown in figure 6.1. The planned modules indicated in red have been implemented and operate as expected.

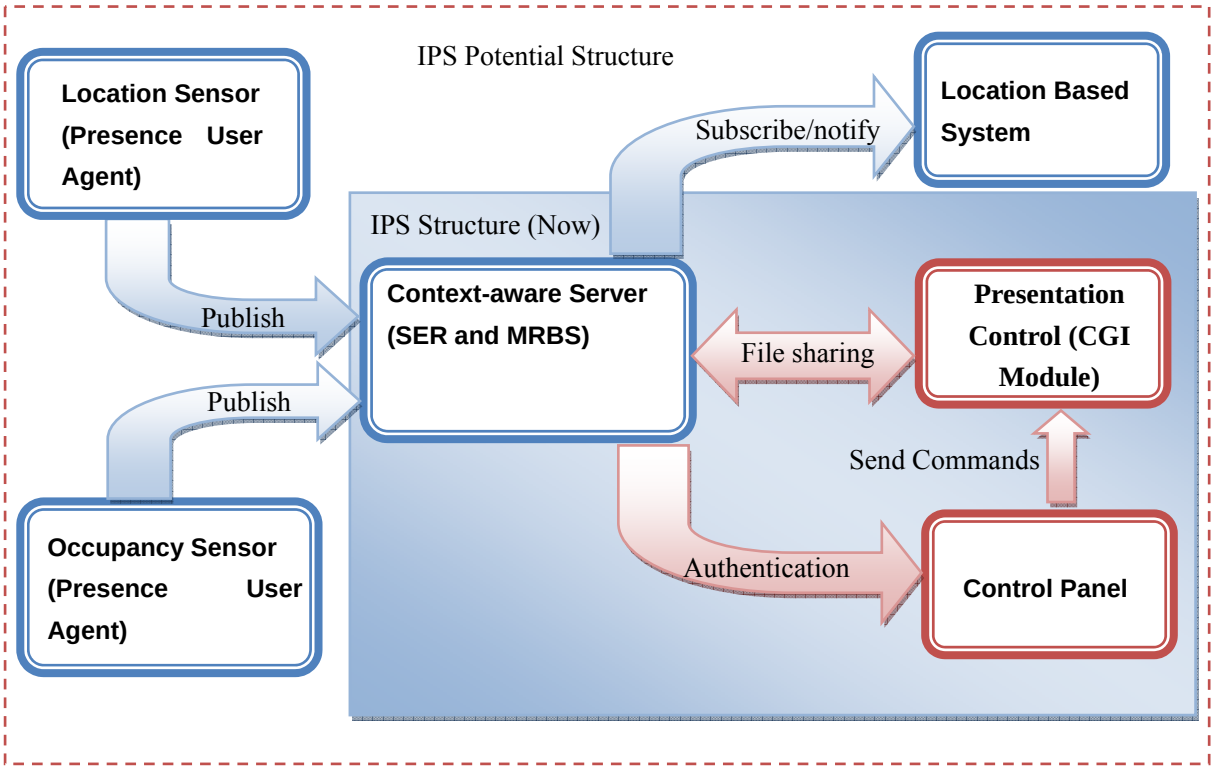


Figure 6.1 The whole context aware project and IPS structure inside

Figure 6.1 shows the IPS structure in the context of the larger context aware project. The dotted-line box indicates the potential parts which will be integrated into IPS in the future. While, the IPS developed in this thesis is only a prototype. It does operate and could already be used without the other components.

In summary, IPS development as described in this thesis is a successful project which supports an efficient and comfortable way to control a presentation and avoids the speaker having to worry about device configuration. IPS offers boundless potential for integration with other smart context aware modules to provide a more intelligent environment; as will be expected of an Ambient Intelligence System in the near future.

6.2 Future work

6.2.1 MRBS

The MRBS running in the context server was mainly used for file management, user management, and schedule management. In IPS, MRBS not only supports the presentation schedule management, but also was extended to provide file management and user authentication. The new structure of MRBS is shown in figure 6.2.

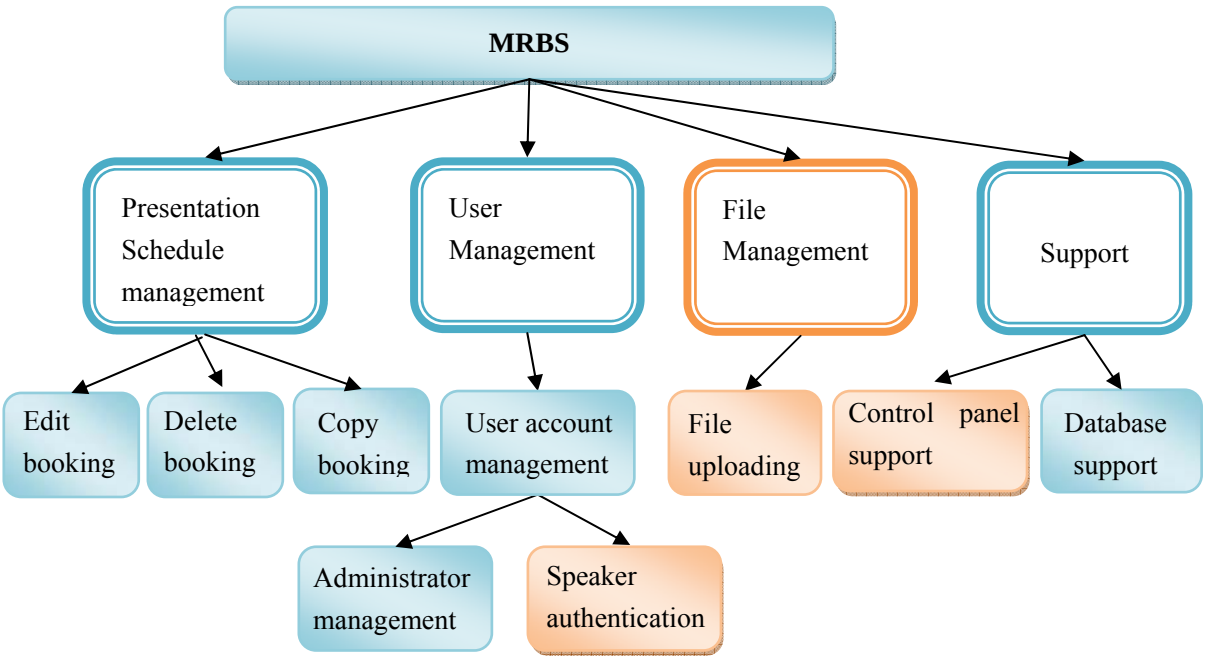


Figure 6.2 the New Structure of MRBS in IPS

Figure 6.2 shows that the new structure of MRBS. The orange boxes describe the new functions we added in embedding into our IPS. Thus MRBS acts as the central management software in IPS to provide online booking and speaker authentication.

There remain some improvements that could be made. First, better support for finding an available room and automatic booking cancellation when a meeting room is booked but unused after some period of time. By integrating with Xueliang Ren and Ke Wang's two thesis projects work [64,73], the MRBS can be extended to exploit real time room occupation information with location of an available meeting room, automatic cancellation of under booking, and even automatic redirection of calls to the user when they are actually in a meeting room.

6.2.2 CGI Presentation Control Module

The CGI module running on the presentation server receives commands sent from the control panel and executes the command. Currently the CGI modules consist of pres-client.cgi and presentation-server.cgi. The pres-client.cgi is initiated by the web interface of the control panel. It sends the command and file name to the presentation-server.cgi. Then, the presentation-server.cgi simulates the correspondent key operations to execute the command. The details of their working can be found in figure 3.9.

However, there are still some limitations of these CGI modules. If there are multiple presentation servers, we need to compile a CGI module for each presentation server - which adds extra work for the administrator. Therefore, it would be better to make a portable interface for the CGI module. Thus the CGI module could be installed on another presentation server with just the press of a button. Additionally, the CGI module should be automatically configured for this presentation server.

Another improvement would be to make the web interface between the control panel and the pres-client.cgi more portable. Currently, we need to pre-define the IP address of the presentation server into the interface. If we change the presentation server, we need to change the IP address in the web interface which currently requires re-coding and re-compiling of the CGI module. A proper solution on these issues is to program a module to automatically detect the IP address and put the new IP address into the interface in order to produce the necessary code. However, we can also remove the compiling work and make the CGI module automatically check the machine's IP before placing the module on the presentation server.

For the file types of presentation slides, now the presentation control module only supports OpenOffice *odp and PowerPoint *ppt. It is necessary to extend the control module to support on audio and video files.

6.2.3 Control Panel

The control panel for speakers in IPS is a web page with command links on it. The appearance of the control panel was shown in figure 4.8. Following its improvements, the command buttons are divided into two groups. One group named "To Navigate" which consists of four buttons: Full Screen, Next Slide, Previous Slide, Stop Full Screen. The other

group named “To Finish” which consists of three buttons: Enter, Close Slide, Remove Slide. Four buttons in the first group are designed for navigating slides during the presentation and the rest three buttons in the second group are used for controlling error recovery and closing and removing the slides after the presentation is finished. From the test we are conducted (see Chapter 4) and the improvements we made, now all the buttons work together well with the CGI module. The speaker can use the control panel to control the whole presentation without configuring the presentation server.

However the control panel still needs improvements. The appearance of the control panel can be improved by adding image button to substitute the links. The control panel would look nicer by editing including a CSS style and the background image on this page.

MRBS is already designed to support multiple languages, thus it would be very straight forward to use the existing mechanisms in this software to implement a web interface in the user's choice of language. However, this remains for future work.

6.2.4 Securing the Presentation

As noted in section 3.3.1, only the presenter should be able to upload and present their material, i.e., no one else should be able to access the contents of the presentation, and the presenter should be able to know that the presentation is secure. One means of doing this is for the speaker to encrypt the file and only provide the decryption key to the presentation server at the time of the presentation. This could be done in a variety of ways, and the best method of doing so should be examined in a future thesis.

6.2.5. Correlated Load on the NFS File Server

As multiple presentations may start at the same time (for example at 13:00) the presentation servers might all try to load their presentations from the same NFS file server at the same time.This correlated load could lead to poor performance. One way of solving this is to put the presentation on the local disk of the presentation server, in advance of the presentation.

However, this re-introduces the problem of when to do this transfer. An alternative is to initially place the file on the presentation server, but this might cause problems if the user's presentation is re-scheduled to another meeting room (when the file could be copied to this new presentation server). Details of this should be investigated in a future project.

6.2.6. Better File Distribution on NFS Clients.

Now each presentation server is connected to the context server through NFS. When a presenter uploads files to the context server, these files are automatically distributed to all presentation servers. It considers neither which room the presentation files are meant for, nor

at what time the presentation is to take place. It might cause redundant documents on some presentation servers if IPS was used on a larger scale.

The problem of sending the files to the right server could be solved by the context server sorting the files into different folders when they are uploaded. Each presentation server could then just sync with its own personal folder. This would make the system more scalable.

References

[1] Thomas Kirste: Ambient Intelligence: Towards Smart Appliance Ensembles (December, 2004), Dept. of Computer Science, Rostock University, Germany. http://www.informatik.uni-rostock.de/mmis/paper.pdf (last visited: July 10th, 2008)

[2] Emile Aarts, Rick Harwig, and Maarten Schuurmans: Chapter 3 Ambient Intelligence, in *The Invisible Future: The Seamless Integration Of Technology Into Everyday Life*, McGraw-Hill Companies, 2001.

[3] Phillips Research – Technology, "What is an Ambient Intelligence", http://www.research.philips.com/technologies/syst_softw/ami/index.html(last visited: July 10th, 2008)

[4] Ambient intelligence: http://en.wikipedia.org/wiki/Ambient_intelligence (last visited: July 10th, 2008)

[5] AmI-07: http://www.ami-07.org/home.html (last visited: July 10th, 2008)

[6] Pebbles Project: http://www.pebbles.hcii.cmu.edu/index.php (last visited: July 10th, 2008)

[7] Chen Rui, Hou Yi-bin, et al. : A Framework for Local Ambient Intelligence Space: The AmI-Space Project. *Embedded Software and Systems Institute, Beijing University of Technology, Beijing 100022, China.* Published at: 31st Annual International Computer Software and Applications Conference(COMPSAC 2007), 0-7695-2870-8/07, 2007 IEEE.

[8] Soraya Kouadri Mostéfaoui and Béat Hirsbrunner: Towards a Context-Based Service Composition Framework. Department of Computer Science University of Fribourg, Switzerland. Published at: Proceedings of the international Conference on Web Service, ICWS' 03, 23-26 June 2003. pp. 42-45.

[9] Mark Finn: The Handheld Classroom: Educational Implications of Mobile Computing, Media and Communications at Swinburne University of Technology published at: Australian Journal of Emerging Technologies and Society Vol. 2, No. 1,2004

[10] Brad A. Myers: Using Handhelds and PCs Together. Communications of the ACM/Vol. 44, No. 11, November 2001, http://www.cs.cmu.edu/~pebbles/papers/pebblescacm.pdf (last visited: July 10th, 2008).

[11] Philips Research – Technologies: Ambilight - the new TV experience. http://www.research.philips.com/technologies/syst_softw/ami/ambilight.html (last visited: July 10th, 2008).

[12] The Pittsburgh Pebbles PDA Project - Slideshow Commander: http://www.pebbles.hcii.cmu.edu/software/slideshow/index.php (last visited: July 10th, 2008).

[13] Microsoft Research: Easy Living. http://research.microsoft.com/easyliving/ (last visited: July 10th, 2008).

[14] Microsoft PressPass: Microsoft Debuts Concept Home: http://www.microsoft.com/presspass/press/2000/Jan00/HomeDebutCESpr.mspx (last visited: July 10th, 2008).

[15] Brighthand: Presentations from your PDA-Margi Systems Presenter-to-Go SD card. http://www.brighthand.com/default.asp?newsID=12064 (last visited: July 10th,

2008).

[16] HP: HP Wireless Presenter To Go Software V2.1 – Product details & specifications. http://h20392.www2.hp.com/portal/swdepot/displayProductInfo.do?productNumber=L1694A (last visited: July 10th, 2008).

[17] The Pittsburgh Pebbles PDA Project: Pebbles Project Overview: http://www.pebbles.cs.cmu.edu/ (last visited: July 10th, 2008).

[18] Ambient Networks: An introduction. http://www.ambient-networks.org/about.html (last visited: July 10th, 2008).

[19] Brad A. Myers, Jeff.Nichols, and Rob Miller: User Interfaces that Span Hand-Held and Fixed Devices. Human Computer Interaction Institute, School of Computer Science,Carnegie Mellon University: http://www.cs.cmu.edu/~pebbles/papers/chi2001workshop4.html (last visited: July 10th, 2008).

[20] Brett Powers: Electronic Resources Reviews – Presenter-to-go. Wright State University Libraries Dayton, Ohio: http://www.pubmedcentral.nih.gov/picrender.fcgi?artid=153173&blobtype=pdf (last visited: July 10th, 2008).

[21] Margi: Presenter-to-go: http://www.margi.com/products/prod_ptg_featcomp.htm (last visited: July 10th, 2008).

[22] HP technical support: HP Digital Projectors - Wireless Presenter-to-go Frequently Asked Questions. http://h10025.www1.hp.com/ewfrf/wc/ fastFaqLiteDocument?lc=en&cc=uk&dlc=en&docname=c00209592 (last visited: July 10th, 2008).

[23] HP [158]: a Advanced Connectivity Module – User's Guide. http://h10032.www1.hp.com/ctg/Manual/c00064012.pdf (last visited: July 10th, 2008).

[24] Carnegie Mellon University: Shortcutter. Developed by Human Computer Interaction Institute School of Computer Science: http://www.pebbles.hcii.cmu.edu/software/shortcutter/index.php (last visited: July 10th, 2008).

[25] Brad A. Myers "Using Hand-Held Devices and PCs Together," Communications of the ACM. Volume 44, Issue 11. November, 2001. pp. 34 – 41.

[26] Brad A.Myers, Robert C. Miller, et al.: *Taking Handheld Devices to the Next Level*. Published by the IEEE Computer Society, 0018-9162/04/$20.00. 2004 IEEE

[27] ZDNet: Sony Ericsson W880：http://www.zdnet.com.tw/review/handheld/0,2000086602,20031721,00.htm (last visited: July 10th, 2008).

[28] Pocket PC Central: Pocket PC Display Adapters-Presenter-to-go SD card. : http://pocketpccentral.net/dispadps.htm (last visited: July 10th, 2008)

[29] Jeffrey Nichols and Mathilde Pignol: Personal Universal Controller- Appliance - Specification Language Documentation: http://www.pebbles.hcii.cmu.edu/puc/specification.html (last visited: July 10th, 2008).

[30] NetComm Solutions™: Networked Projectors. July 2002. http://www.cdw.com/webcontent/editorial/hardware/070302_NetworkedProjectors.asp#3 (last visited: July 10th, 2008).

[31] Sony: Press Release - SONY Introduces Network Strategies for The Conference Room. June 15, 2000. http://www.sony.com/news. (last visited: July 10th, 2008).

[32] Ali A. Nazari Shirehjini: An Evaluation of mobile 3D-based interaction with complex Multimedia Environments. Fraunhofer-IGD, Darmstadt, Germany. Published In: HCI International 2007. Proceedings and Posters [DVD-ROM]: With 8

further Associated Conferences. Berlin, Heidelberg, New York : Springer Verlag, 2007, LNCS 4556, pp. 108-115

[33] Ali Asghar Nazari Shirehjini, Michael Hellenschmidt, and Thomas Kirste: An Integrated User Interface Providing Unified Access to Intelligent Environments and Personal Media. Published at: EUSAI2004, 8-10 November 2004, Eindhoven, the Netherlands.Copyright 2004 ACM 1-58113-992-6

[34] Logitech: 2.4 GHz Cordless Presenter http://www.logitech.com/index.cfm/notebook products/presenters/devices/175&cl=us,en (last visited: July 10th, 2008).

[35] OpenBSD: PF-Issues with FTP: FTP Modes http://www.openbsd.org/faq/pf/ftp. html#natserver (last visited: July 10th, 2008).

[36] Linux 2.4 Packet Filtering HOWTO: So What's A Packet Filter? http://www.netfilter. org/documentation/HOWTO/packet-filtering-HOWTO-3.html (last visited: July 10th, 2008).

[37] Athanasios Karapantelakis, "Presentation Scenario", Support document, Communication Systems, Royal Institute of Technology (KTH), Stockholm, Sweden, October 2007.

[38] Haruumi Shiode, "In-building Location Sensing Based on WLAN Signal Strength", Master thesis, Communication Systems, Royal Institute of Technology (KTH), Stockholm, Sweden, March 2008.

[39] Sun Yu, "Context-aware applications for a Pocket PC", Master thesis, Communication Systems, Royal Institute of Technology (KTH), Stockholm, Sweden, December 2007.

[40] M. Z. Eslami, "A Presence server for Context-aware applications", Master thesis, Communication Systems, Royal Institute of Technology (KTH), Stockholm, Sweden, December 2007.

[41] Daniel Hübinette, "Occupancy Sensor System: For Context-aware Computing", Master thesis, Communication Systems, Royal Institute of Technology (KTH), Stockholm, Sweden, December 2007.

[42] MRBS: Introduction and downloaded. Available at: http://mrbs.sourceforge.net/ (last visited: July 10th, 2008).

[43] Apache: Download Mirrors: Available at: http://www.apache.org/dyn/closer.cgi (last visited: July 10th, 2008).

[44] PHP: downloads. Available at: http://www.php.net/downloads.php (last visited: July 10th, 2008).

[45] MySQL: Community Downloads. Available at: http://dev.mysql.com/downloads/ (last visited: July 10th, 2008).

[46] The phpMyAdmin Project - Effective MySQL Management.

[47] MRBS: Try MRBS: http://mrbs.sourceforge.net/demo.html (last visited: July 10th, 2008).

[48] PHP Tutorial – File Upload. Available at: http://www.tizag.com/phpT/fileupload.php (last visited: July 10th, 2008).

[49] PHP Manual: Handling file uploads. Available at: http://se2.php.net/features.file-upload(last visited: July 10th, 2008).

[50] W3schools: PHP File Upload. Available at: http://www.w3schools.com/PHP/php_file_upload.asp (last visited: July 10th, 2008).

[51] S. Shepler, B. Callaghan, D. Robinson, R. Thurlow, C. Beame, M. Eisler, and D. Noveck. "Network File System (NFS) version 4 Protocol"; RFC 3530 (Proposed Standard) obsoletes RFC 3010. April 2003. [Online] Available at: http://tools.ietf.org/ html/rfc3530 (last visited: July 10th, 2008).

[52] Network File System (protocol) available at: http://en.wikipedia.org/wiki/Network_File_System_(protocol) (last visited: July 10th, 2008).

[53] Sun Microsystems, Inc. "NFS: Network File System Protocol Specification"; RFC 1094 (INFORMATIONAL), March 1989, updated by 1813, 3010, and 3050. [Online] Available at: http://tools.ietf.org/html/rfc1094 (last visited: July 10th, 2008).

[54] B. Callaghan, B. Pawlowski, and P. Staubach, "NFS Version 3 Protocol Specification", RFC 1813, June 1995, updated by 3010 and 3050. [Online] Available at: http://tools.ietf.org/html/rfc1813 (last visited: July 10th, 2008).

[55] Magyarkúti Barna, "Setting up a Fedora NFS server". [Online] Available at: myfedora- http:// fconfig.wordpress.com/2006/08/17/setting-up-a-fedora-nfs-server/ (last visited: July 10th, 2008).

[56] Linux NFS-HOWTO, Section 3: Setting Up an NFS Server. Available at: http://nfs.sourceforge. net/nfs-howto/ar01s03.html (last visited: July 10th, 2008).

[57] Linux NFS-HOWTO, Section 4: Setting Up an NFS Client. Available at: http://nfs.sourceforge.net/nfs-howto/ar01s04.html (last visited: July 10th, 2008).

[58] Ubuntu Forums, Tutorial and Tips: HOWTO: NFS Server/Client. Available at: http://ubuntuforums.org/showthread.php?t=249889 (last visited: July 10th, 2008).

[59] Netprog: "Socket Programming over UDP", available at: http://www.cs.rpi.edu/~hollingd/netprog/notes/udp/udp.pdf (last visited: July 10th, 2008).

[60] X.org Foundation: X11R7.3. Available at: http://www.x.org/wiki/Releases/7.3 (last visited: July 10th, 2008).

[61] WIKIPEDIA: X Window System. Available at: http://en.wikipedia.org/wiki/X_Window_System#User_interface_features (last visited: July 10th, 2008).

[62] The Common Gateway Interface: Introduction. Available at: http://hoohoo.ncsa.uiuc.edu/cgi/intro.html (last visited: July 10th, 2008).

[63] Common Gateway Interface. Available at: http://en.wikipedia.org/wiki/Common_Gateway_Interface (last visited: July 10th, 2008).

[64] Ren Xueliang: "A Meeting Detector to Provide Context to a SIP Proxy v2.7", Master thesis, Communication Systems, Royal Institute of Technology (KTH), Stockholm, Sweden, 2008.

[65] Jeffrey Nichols, Brad A. Myers, Michael Higgins, Joseph Hughes, Thomas K. Harris, Roni Rosenfeld, Mathilde Pignol. "Generating Remote Control Interfaces for Complex Appliances," In Proceedings of UIST'2002, Paris, France. Oct 27-30. pp. 161-170. Available at: http://www.jeffreynichols.com/papers/pucUIST_2002.pdf (last visited: July 10th, 2008).

[66] InFocus Accessories: InFocus Navigator™ Remote. Available at: http://www.infocus.com/Accessories/Remote/HW_NAVIGATOR_GEN.aspx (last visited: July 10th, 2008).

[67] mini-itx.com (The Next Small Thing): mini-itx FAQ. Available at: http://www.mini-itx.com/faq.asp (last visited: July 10th, 2008).

[68] VIA-A World of Digital Brilliance: VIA Mini-ITX Mainboard Form Factor: 17cm x 17cm. Available at: http://www.via.com.tw/en/initiatives/spearhead/mini-itx/ (last visited: July 10th, 2008).

[69] VIA-A World of Digital Brilliance: Mini-ITX 2.0: The Mini PC Platform of the Future. Available at: http://www.via.com.tw/en/initiatives/spearhead/mini-itx_2.0/ (last visited: July 10th, 2008).

[70] VIA-A World of Digital Brilliance: VIA EPIA EN-Series Mini-ITX Board. Available at: http://www.via.com.tw/en/products/mainboards/motherboards.jsp?motherboard_id=399 (last visited: July 10th, 2008).

[71] Office of DOE Science Education- Ask A Scientist: Hole in One Probability. http://www.newton.dep.anl.gov/askasci/math99/math99243.htm (last visited: July 10th, 2008).

[72] OpenOffice.org: Impress-More power to your presentations. Available at: http://www.openoffice.org/product/impress.html (last visited: July 10th, 2008).

[73] Ke Wang: “Exploit Presence”, Master thesis, Communication Systems, Royal Institute of Technology (KTH), Stockholm, Sweden, (expected Fall) 2008.

[74] BBC Home: Configuring and connecting your laptop to a projector for a presentation. Available at: http://www.bbc.co.uk/dna/h2g2/brunel/A999110 (last visited: July 10th, 2008).

Appendix A

These are newly added modules in MRBS.

- upload_login.php

```
<?php
# $Id: upload_login.php,v 1.30.2.8 2008/05/28 15:50:10 Lidan Hu $

require_once('grab_globals.inc.php');
include "config.inc.php";
include "functions.inc";
include "$dbsys.inc";
include "mrbs_auth.inc";

global $twentyfourhour_format;

#If we dont know the right date then make it up
if(!isset($day) or !isset($month) or !isset($year))
{
    $day   = date("d");
    $month = date("m");
    $year  = date("Y");
}
if(empty($area))
    $area = get_default_area();
if(!isset($edit_type))
    $edit_type = "";

# if(!getAuthorised(0))
# {
#   showAccessDenied($day, $month, $year, $area);
#   exit;
# }

print_header($day, $month, $year, $area);

 ?>

<?php
# ob_start();
    #    mysql_connect('localhost:3306','root','ccslab1');
   #     mysql_select_db("mrbs");

  if(isset($_POST['speaker_upload']))
```

```
  {
        if(isset($_POST['IDcode'])&&!empty($_POST['IDcode']))
        {
              //username and password are stored as plain text
              $sql   =   "SELECT   COUNT(*)  AS rcnt FROM   $tbl_speakers
WHERE  IDcode='".$_POST['IDcode']."'";
              $res   =  sql_query($sql);
              $arr   =  sql_fetch_array($res);
              $num   =  $arr['rcnt'];
              $IDcode = $_POST['IDcode'];
              if($num == 1)
              {
                    #  header("Location: upload_file.php");
                    #  die
('<script>document.location="'.$_POST['http://localhost/mrbs/web/upload_log
in.php'].'";</script>');
                    echo   "<META   HTTP-EQUIV=REFRESH
CONTENT=0;URL=upload_file.php?IDcode=$IDcode>";
                    exit();
              }
              else
              {     #  fatal_error(1, sql_error());
                    echo   "No matching ID-code";

              }
        }
        else
        {
              echo   "Please input the correct speaker and ID-code";
        }
  }
  ?>
  <br>
  <br>
  <br>
  <br>
  <form   method="post"   action="<?php  echo $_SERVER['PHP_SELF'];   ?>">
  ID-code: <input   type=password   name=IDcode>
  <input   type=submit   name=speaker_upload>
  </form>
  <br>
  <br>
<?php include "trailer.inc" ?>
```

- upload_file.php

```
<?php
# $Id: edit_entry.php,v 1.30.2.8 2008/05/28 15:50:10 Hu Lidan $

require_once('grab_globals.inc.php');
include "config.inc.php";
include "functions.inc";
include "$dbsys.inc";
include "mrbs_auth.inc";

global $twentyfourhour_format;

#If we dont know the right date then make it up
if(!isset($day) or !isset($month) or !isset($year))
{
    $day   = date("d");
    $month = date("m");
    $year  = date("Y");
}
if(empty($area))
    $area = get_default_area();
if(!isset($edit_type))
    $edit_type = "";

# if(!getAuthorised(0))
# {
#   showAccessDenied($day, $month, $year, $area);
#   exit;
# }

print_header($day, $month, $year, $area);

?>
<br>

<form enctype="multipart/form-data" action="upload_file.php?IDcode=<?php
echo $IDcode;?> " method="post">

<input type="hidden" name="MAX_FILE_SIZE" value="2000000">

<input type=file name=upfile size=20>

<input type=submit value='Upload Files'>
</form>
```

```
<?php

 function getname($exname){

   $dir = "../files/";

   $i=1;

  while(true){

      if(!is_file($dir.$i.".".$exname)){

         $user = getUserName();
         if ($user){
               $name = $user.".".$exname;
         }
        else {
            $name = $_FILES['upfile']['name'];
        }

        break;

      }

      $i++;

  }


   return $dir.$name;

 }

$exname=strtolower(substr($_FILES['upfile']['name'],(strrpos($_FILES['upfil
e']['name'],'.')+1)));

  $uploadfile = getname($exname);

#if the upload user is Administrator, then we name the file as the administrator's
name + file type
$fileuser = getUserName();
$filename= $fileuser.".".$exname;
```

```
if (!$fileuser)
{
       $filename = $_FILES['upfile']['name'];
}


    #$uploadfile.$_FILES['file']['name']
    if (move_uploaded_file($_FILES['upfile']['tmp_name'], $uploadfile)) {

     echo "<h2><font color=#ff0000>Uploading success!/font></h2><br><br>";
     $sql  = "UPDATE $tbl_speakers SET fileName= '$filename' WHERE
$tbl_speakers.IDcode='$IDcode' LIMIT 1";

     if (sql_command($sql) < 0) return 0;


}
    else if (!isset($id)) {
       echo "<h2><font color=#ff0000>Welcome to upload
files!</font></h2><br><br>";
       global $IDcode;
       $IDcode= $_REQUEST['IDcode'];

}
    else {
    echo "<h2><font color=#ff0000>Uploading failed!/font></h2><br><br>";
}
echo "These are some information about the files:”;
echo "<br><br>File Name:”.$_FILES['upfile']['name'] .

     "<br><br>File Type:” .$_FILES['upfile']['type'] .

     "<br><br>File Name Stored:”;

echo $filename;
echo $uploadfile;

echo  "<br><br>File Size:”.$_FILES['upfile']['size'] .

    "<br><br>Error Code:”.$_FILES['upfile']['error'];

 ?>
<?php include "trailer.inc" ?>
```

- presenter-login.php

```
<?php
# $Id: upload_login.php,v 1.30.2.8 2008/05/28 15:50:10 Hu Lidan $

# require_once('grab_globals.inc.php');
include "config.inc.php";
include "functions.inc";
include "$dbsys.inc";
include "mrbs_auth.inc";

 ?>

<?php
# ob_start();
   #    mysql_connect('localhost:3306','root','ccslab1');
   #    mysql_select_db("mrbs");

  if(isset($_POST['speaker_upload']))
  {
     if(isset($_POST['IDcode'])&&!empty($_POST['IDcode']))
     {
      //username and password are stored as plain text
        $sql  =  "SELECT room_id FROM $tbl_speakers WHERE
IDcode='".$_POST['IDcode']."'";
       $res  =  sql_query1($sql);

      $sql1 = "SELECT fileName FROM $tbl_speakers WHERE
IDcode='".$_POST['IDcode']."'";
       $filename =  sql_query1($sql1);
      if($res == 1)
        {
           echo   "<META   HTTP-EQUIV=REFRESH
CONTENT=0;URL=Grimeton.php?filename=$filename>";
           exit();
        }
     else if ($res == 2)
        {
           echo   "<META   HTTP-EQUIV=REFRESH    CONTENT=0;URL=H枚
rby.php?filename=$filename>";
           exit();
        }
    else if ($res == 3)
       {
           echo   "<META   HTTP-EQUIV=REFRESH
```

```
CONTENT=0;URL=MINT.php?filename=$filename>";
         exit();
        }
        else if ($res == 4)
        {
          echo   "<META   HTTP-EQUIV=REFRESH
CONTENT=0;URL=Motala.php?filename=$filename>";
          exit();
         }
         else if ($res == 5)
        {
         echo   "<META   HTTP-EQUIV=REFRESH
CONTENT=0;URL=Open_Area.php?filename=$filename>";
          exit();
         }
         else
         {
            echo "###";
            echo $res;
            echo $_POST['IDcode'];
            echo   "No matching ID-code";

         }
     }
    else
     {
         echo   "Please input the correct speaker and ID-code";
      }
  }
  ?>
  <br>
  <form   method="post"   action="<?php  echo $_SERVER['PHP_SELF'];   ?>">
   <H2>Please log in before you start the presentation!</H2>
<br>
<br>
  ID-code: <input   type=password   name=IDcode>
  <input   type=submit   name=speaker_upload>
  </form>
  <br>
  <br>
```

Appendix B

Control panel in Grimeton

- Grimeton.php

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<HTML LANG="en-us">
<HEAD>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<meta name="author" content="root">
<META NAME="description" CONTENT="page for controlling grimeton projector">
<META HTTP-EQUIV="pragma" CONTENT="no-cache">
<META NAME="ROBOTS" CONTENT="NOARCHIVE">
<META NAME="copyright"  CONTENT="Copyright &copy; 2008 G. Q. Maguire Jr. and
Lidan Hu">


<TITLE>Control Panel of Grimeton</TITLE>
</HEAD>
<Body>
<!-- <img src= "http://vvv.it.kth.se/misc/people/faces/maguire.gif"> -->

<H1>Web control for Lidan's presentation in Grimeton</H1>

<ul>

<?php
# Display day name checkboxes according to language and preferred weekday start.
 $filepath = "PNAME/usr/local/share/mrbs_files/";
 $name = $filepath.$_REQUEST['filename'];

?>

<p>Start the presentation!
<li><a
href="http://130.237.15.224/cgi-bin/pres-client.cgi?command=INIT&name=<?php
echo $name;?>">INIT</a></li>


<p>To Navigate the Slides
<li><a
href="http://130.237.15.224/cgi-bin/pres-client.cgi?command=START&name=null
">Full Screen</a></li>
<li><a
```

```
href="http://130.237.15.224/cgi-bin/pres-client.cgi?command=SLIDE_NEXT&name
=null">Next Slide</a></li>
<li><a
href="http://130.237.15.224/cgi-bin/pres-client.cgi?command=SLIDE_PREV&name
=null">Previous Slide</a></li>
<li><a
href="http://130.237.15.224/cgi-bin/pres-client.cgi?command=PEND&name=null"
>Stop Full Screen</a></li>


<p>To Recovery
<li><a
href="http://130.237.15.224/cgi-bin/pres-client.cgi?command=ENTER&name=null
">Enter</a></li>


<p>To End the Presentation
<li><a
href="http://130.237.15.224/cgi-bin/pres-client.cgi?command=CLOSE&name=null
">Close Slides</a></li>
<li><a
href="http://130.237.15.224/cgi-bin/pres-client.cgi?command=REMOVE&name=nul
l">Remove Slides</a></li>
</ul>


</BODY>
</HTML>
```

Appendix C

- CGI Presentation Control Module (pres-client.c)

```
/*
 *pres-client.c
 * send commands given as an argument on a UDP socket
 *
 *
 */

#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <string.h>
#include <unistd.h>


#ifndef TRUE
#define TRUE 1
#endif

#ifndef FALSE
#define FALSE 0
#endif

#define debug FALSE

#define Length_of_Presentation_ID 128
#define CmdBufferSize 20
#define NameBufferSize 1024
#define my_port       9999

#define destination_host "130.237.15.224"

/* commands sent to the presentation software by the presentation control application */
#define Presentation_Init     "INIT"
#define Start_Presentation    "START"
#define Next_Slide            "SLIDE_NEXT"
#define Previous_Slide        "SLIDE_PREV"
#define Presentation_End      "PEND"
```

```
#define Presentation_Name     "PNAME"
#define Enter                 "ENTER"

#define Maximum_cmd_length 1024


int main(argc, argv)
int argc;
char **argv;
{
  int client_socket_fd; /* Socket to client, server */

  struct sockaddr_in server_addr;       /* server's address */
  //struct sockaddr_in client_addr;       /* client's address */
  //int client_addr_len;

  //char presentationID[Length_of_Presentation_ID];
  char data_cmd[CmdBufferSize];
  char data_name[NameBufferSize];
  //char CMD[Maximum_cmd_length]; /* buffer to hold the command to execute to start impress */
  char *cmdstring;

  int sendto_flags=0;

  printf("%s%c%c\n","Content-Type:text/html;charset=iso-8859-1",13,10);

  printf("<title>pres-client running</title>\n");

  printf("<H1>Web control for Lidan's presentation in Grimeton</H1>\n");

  printf("<p>To Navigate the Slides\n");
  printf("<li><ahref=\"http://130.237.15.224/cgi-bin/pres-client.cgi?command=START&name=null\"> Full
Screen</a></li>\n");
  printf("<li><ahref=\"http://130.237.15.224/cgi-bin/pres-client.cgi?command=SLIDE_NEXT&name
=null\">Next Slide</a></li>\n");

printf("<li><ahref=\"http://130.237.15.224/cgi-bin/pres-client.cgi?command=SLIDE_PREV&name=null\">Pr
evious Slide</a></li>\n");
  printf("<li><ahref=\"http://130.237.15.224/cgi-bin/pres-client.cgi?command=PEND&name=null\">  Stop
Full Screen</a></li>\n");

  printf("<p>To Recovery\n");
  printf("<li><ahref=\"http://130.237.15.224/cgi-bin/pres-client.cgi?command=ENTER&name=null\">
Enter</a></li>\n");
```

```
  printf("<p>To End the Presentation\n");
  printf("<li><ahref=\"http://130.237.15.224/cgi-bin/pres-client.cgi?command=CLOSE&name=null\">
Close Slides</a></li>\n");
  printf("<li><ahref=\"http://130.237.15.224/cgi-bin/pres-client.cgi?command=REMOVE&name=null\">
Remove Slides</a></li>\n");

  cmdstring=getenv("QUERY_STRING");
  if (debug)
  {
     printf("<H3>results is</H3>\n");
     fprintf(stdout, "%s\n", cmdstring);
  }
  printf("<br>\n");

  /* create a UDP socket */
  if ((client_socket_fd = socket(AF_INET, SOCK_DGRAM, IPPROTO_UDP)) == -1) {
    perror("Unable to open socket");
    exit(1);
  };


  /* initialize the server address structure */
  memset( (char*)&server_addr, 0, sizeof(server_addr));
  server_addr.sin_family=AF_INET;
  server_addr.sin_port=htons(my_port);
  if (inet_aton(destination_host, (struct in_addr*)&server_addr.sin_addr) == 0) {
    fprintf(stderr, "could not get an address for: %s", destination_host);
    exit(1);
  }

  /* put the the command into the buffer and send it */

  sscanf(cmdstring, "command=%[^&]&name=%s",data_cmd,data_name);
  //because the amount of file names are infinity, we cannot use "name matching" in Pres-Server.cgi
  //so, just add some special symbols at the beginning of the file names. i.e. #file_name

  if ((sendto(client_socket_fd, data_name, NameBufferSize,
          sendto_flags, (struct sockaddr*)&server_addr, sizeof(server_addr))) == -1) {
    perror("Unable to send to socket"); close(client_socket_fd);
    exit(1);
  }
```

```
  if ((sendto(client_socket_fd, data_cmd, CmdBufferSize,
           sendto_flags, (struct sockaddr*)&server_addr, sizeof(server_addr))) == -1) {
    perror("Unable to send to socket"); close(client_socket_fd);
    exit(1);
  }

  if (debug)
  {
     fprintf(stdout, "sent file name: %s\n", data_name);
          printf("<br>\n");
     fprintf(stdout, "sent command: %s\n", data_cmd);
     printf("<br>\n");
     }

  close(client_socket_fd); /* close the socket */
  exit(0);
}
```

- CGI Presentation Control Module (presentation-server.c)

```
/*
 * presentation-server.c
 * Based upon consoleAgent.cs
 *
 */

#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <string.h>

#include <X11/Xlib.h>
#include <X11/Xutil.h>
#include <X11/cursorfont.h>
#include <X11/keysymdef.h>
#include <X11/keysym.h>
#include <X11/extensions/XTest.h>

#include <unistd.h>
#include <arpa/inet.h>



#ifndef TRUE
#define TRUE 1
#endif

#ifndef FALSE
#define FALSE 0
#endif

#define debug 0

#define Length_of_Presentation_ID 128
#define DataBufferSize 1024
#define my_port       9999

#define destination_host "130.237.15.224"

/* commands sent to the presentation software by the presentation control application */
#define Presentation_Init     "INIT"
```

```
#define Start_Presentation    "START"
#define Next_Slide            "SLIDE_NEXT"
#define Previous_Slide        "SLIDE_PREV"
#define Presentation_End      "PEND"
#define Presentation_Name     "PNAME"
#define Enter                 "ENTER"
#define Close                 "CLOSE"
#define Remove                "REMOVE"

/* commands to send to impress the presentation application */
/* key code is in <X11/keysymdef.h>, and transfer it to Dec, and send to sendKey()*/
#define ESC_character XK_Escape//"65307" //0xff1b
#define P_character   XK_p//"112" //'p'
#define N_character   XK_n//"110" //'n'
#define F5_character  XK_F5//"65474" //0xffc2
#define Return_character XK_Return
#define Q_character   XK_q
/*
 * note that sending a "B" or "." to impress will cause it to show a blank slide
 * sending a HOME key goes to first slide
 * sending a END key goes to the last slide
 */


#define Maximum_cmd_length 1024
//#define got_presentation_file 0
FILE *impress_child;        /* global file handle to child process's stdin */
FILE *remove_child;

int sendKey(KeySym keysym, int if_ctrl_key);

int main(argc, argv)
int argc;
char **argv;
{
  int server_socket_fd;          /* server's socket */
  struct sockaddr_in server_addr;       /* server's address */
  struct sockaddr_in client_addr;       /* client's address */
  socklen_t client_addr_len;

  int receivedDataLength;
  char presentationID[Length_of_Presentation_ID];
  char data[DataBufferSize];
  char CMD[Maximum_cmd_length]; /* buffer to hold the command to execute to start impress */
```

```
int sendto_flags = 0;
int got_presentation_file = 0;

/* create a UDP socket */
if ((server_socket_fd = socket(AF_INET, SOCK_DGRAM, IPPROTO_UDP)) == -1) {
  perror("Unable to open socket");
  exit(1);
};


/* initialize the server address structure */
memset( (char*)& server_addr, 0, sizeof(server_addr));
server_addr.sin_family=AF_INET;
server_addr.sin_port=htons(my_port);
server_addr.sin_addr.s_addr = htonl(INADDR_ANY);

if (bind(server_socket_fd, (struct sockaddr*)&server_addr, sizeof(server_addr))==-1) {
  close(server_socket_fd);
  exit(1);
}



fprintf(stdout, "Agent server initiated\n");

while (TRUE) {
  memset( (char*)&data, 0, DataBufferSize); /* clear the buffer */
   client_addr_len = sizeof(client_addr);
  if ((receivedDataLength=recvfrom(server_socket_fd, data, DataBufferSize,
                    sendto_flags, (struct sockaddr*)&client_addr, &client_addr_len)) == -1) {

    perror("Unable to receive from socket");
    close(server_socket_fd);
    exit(1);
  }

  if (1) {
    printf("Received packet from %s:%d\nData: %s\nString length=%d\n",
       inet_ntoa(client_addr.sin_addr), ntohs(client_addr.sin_port), data, receivedDataLength);
  }

  if ((strncmp(Presentation_Init, data, strlen(Presentation_Init)) == 0) && got_presentation_file ==
1 ) {
    if (debug)
```

```
    {
        fprintf(stdout,"init\n");
    }
    else
    {
        sprintf(CMD, "/usr/bin/ooimpress -open %s", presentationID);   /* command to start impress */
        impress_child=popen(CMD, "w");  // run the command and pass single character commands to it


            }
  } else if (strncmp(Start_Presentation, data, strlen(Start_Presentation)) == 0) {
    // View slide show is invoked by the F5 key
    // all the short cut keys in presentation mode are described in the impress manual Appendix A, page
231
    if (debug)
    {
        fprintf(stdout,"start presentation\n");
    }
    else
    {

        sendKey(XK_F5,0);
        //fputs(F5_character, sendKey_child);
   }
  } else if (strncmp(Next_Slide, data, strlen(Next_Slide)) == 0) {
    // Next slide show is invoked by the Right arrow, Down Arrow, Page Down, Enter, Return, or "N" keys
    if (debug)
    {
        fprintf(stdout,"next slide\n");
    }
    else
    {
        sendKey(XK_Right,0);
        //fputs(N_character, sendKey_child);
    }
  } else if (strncmp(Previous_Slide, data,strlen(Previous_Slide)) == 0) {
    // Next slide show is invoked by the Left arrow, Up Arrow, Page Up, Enter, Backspace, or "P" keys
    if (debug)
    {
        fprintf(stdout,"previous slide\n");
    }
    else
    {
        sendKey(XK_Left,0);
```

```
        //fputs(P_character, sendKey_child);
    }
} else if (strncmp(Presentation_End, data, strlen(Presentation_End)) == 0) {
    // To exit send ESCAPE
    if (debug)
    {
        fprintf(stdout,"exiting\n");
        got_presentation_file = 0;
    }
    else
    {
        sendKey(XK_Escape,0);
        //fputs(ESC_character, sendKey_child);
        got_presentation_file = 0;
            }
} else if (strncmp(Presentation_Name, data, strlen(Presentation_Name)) == 0) {
            if (debug)
    {
        fprintf(stdout,"file name\n");
        got_presentation_file = 1;
    }
    else
    {
        memset( (char*)&presentationID, 0, Length_of_Presentation_ID);
        strncpy(presentationID, &data[5], strlen(data)-5);
        presentationID[strlen(data)-5]='\0';
        fprintf(stdout,"Presentation name: %s\n", presentationID);
        got_presentation_file = 1;
    }
} else if (strncmp(Enter, data, strlen(Enter)) == 0) {
    // To send Enter if any
    if (debug)
    {
        fprintf(stdout,"Enter......\n");
    }
    else
    {
        sendKey(XK_Return,0);
            }
} else if (strncmp(Close, data, strlen(Close)) == 0) {
    // To send Close if any
    if (debug)
    {
        fprintf(stdout,"Close......\n");
```

```
        }
        else
        {
            sendKey(XK_q,1);
                }
            } else if (strncmp(Remove, data, strlen(Remove)) == 0) {
        // To send Remove if any
        if (debug)
        {
            fprintf(stdout,"Remove......\n");
        }
        else
        {
            sprintf(CMD, "rm  %s", presentationID);   /* command to remove slides */
            remove_child=popen(CMD, "r");
            pclose(remove_child);
                }
            }
    }

      pclose(impress_child);
    close(server_socket_fd);  /* close the socket */
    exit(0);

}

int sendKey(KeySym keysym, int if_ctrl_key) {

    Display * localDpy;
    int Event;
    int Error;
    int Major;
    int Minor;
    XKeyEvent event;
    Window winFocus;
       int    revert;

    localDpy = XOpenDisplay ( 0 );

    if (! localDpy) {
      fprintf(stderr, "unable to open the local display\n");
      exit(0);
    }
```

```
  if ( ! XTestQueryExtension (localDpy, &Event, &Error, &Major, &Minor ) ) {
     // nope, extension not supported
     fprintf(stderr, "XTest extension not supported on server %s", DisplayString(localDpy));
     XCloseDisplay ( localDpy);
     exit(0);
  }


  event.display = localDpy;
  //event.window = WindowID;
  event.root =  RootWindow (localDpy, DefaultScreen(localDpy));
  event.subwindow = None;
  event.time = CurrentTime;
  event.x = 1;
  event.y = 1;
  event.x_root = 1;
  event.y_root = 1;
  event.same_screen = 1;


     /****************** press ********************/
     /* Get the window ID of the current window */
     XGetInputFocus(localDpy, &winFocus, &revert);
     event.window = winFocus;
     event.type = KeyPress;
     event.keycode = XKeysymToKeycode(localDpy,keysym);

     if (if_ctrl_key==0)
     event.state = 0;
     else
     event.state = ControlMask;

     XSendEvent(event.display, event.window, 1, KeyPressMask|KeyReleaseMask, (XEvent *)&event);

     /****************** release ***********************/
     //XGetInputFocus(localDpy, &winFocus, &revert);
     //event.window = winFocus;
     event.type = KeyRelease;
     XSendEvent(event.display, event.window, 1, KeyPressMask|KeyReleaseMask, (XEvent *)&event);

  XFlush(localDpy );
  XCloseDisplay(localDpy);
  return 1;
}
```

Appendix D

Script files in setting up a NFS server.

- nfs_firewallopen (used to open your firewall for nfs and portmap ports)

```
/sbin/iptables -I RH-Firewall-1-INPUT -m state --state NEW \
-m tcp -p tcp --dport 111 -j ACCEPT |
/sbin/iptables -I RH-Firewall-1-INPUT -m state --state NEW \
-m tcp -p tcp --dport 2049 -j ACCEPT |
/sbin/iptables -I RH-Firewall-1-INPUT -m state --state NEW \
-m tcp -p tcp --dport 48620 -j ACCEPT |
/sbin/iptables -I RH-Firewall-1-INPUT -m state --state NEW \
-m tcp -p tcp --dport 48621 -j ACCEPT |
/sbin/iptables -I RH-Firewall-1-INPUT -m state --state NEW \
-m tcp -p tcp --dport 48622 -j ACCEPT |
/sbin/iptables -I RH-Firewall-1-INPUT -m state --state NEW \
-m tcp -p tcp --dport 48623 -j ACCEPT |
/sbin/iptables -I RH-Firewall-1-INPUT -m state --state NEW \
-m udp -p udp --dport 111 -j ACCEPT |
/sbin/iptables -I RH-Firewall-1-INPUT -m state --state NEW \
-m udp -p udp --dport 2049 -j ACCEPT |
/sbin/iptables -I RH-Firewall-1-INPUT -m state --state NEW \
-m udp -p udp --dport 48620 -j ACCEPT |
/sbin/iptables -I RH-Firewall-1-INPUT -m state --state NEW \
-m udp -p udp --dport 48621 -j ACCEPT |
/sbin/iptables -I RH-Firewall-1-INPUT -m state --state NEW \
-m udp -p udp --dport 48622 -j ACCEPT |
/sbin/iptables -I RH-Firewall-1-INPUT -m state --state NEW \
-m udp -p udp --dport 48623 -j ACCEPT
```

- nfs_servicestart (used to start nfs services)

```
/etc/init.d/portmap start
/etc/init.d/nfslock start
/etc/init.d/nfs start
/etc/rc.d/init.d/netfs start
```

- nfstart (to execute the 2 scripts above)

```
sh ~/tool/nfs_servicestart
sh ~/tool/nfs_firewallopen
```

- nfs_firewallclose (used to close your opened nfs and portmap ports)

```
/sbin/iptables -I RH-Firewall-1-INPUT -m state --state NEW \
    -m tcp -p tcp --dport 111 -j REJECT |
/sbin/iptables -I RH-Firewall-1-INPUT -m state --state NEW \
    -m tcp -p tcp --dport 2049 -j REJECT |
/sbin/iptables -I RH-Firewall-1-INPUT -m state --state NEW \
    -m tcp -p tcp --dport 48620 -j REJECT |
/sbin/iptables -I RH-Firewall-1-INPUT -m state --state NEW \
    -m tcp -p tcp --dport 48621 -j REJECT |
/sbin/iptables -I RH-Firewall-1-INPUT -m state --state NEW \
    -m tcp -p tcp --dport 48622 -j REJECT |
/sbin/iptables -I RH-Firewall-1-INPUT -m state --state NEW \
    -m tcp -p tcp --dport 48623 -j REJECT |
/sbin/iptables -I RH-Firewall-1-INPUT -m state --state NEW \
    -m udp -p udp --dport 111 -j REJECT |
/sbin/iptables -I RH-Firewall-1-INPUT -m state --state NEW \
    -m udp -p udp --dport 2049 -j REJECT |
/sbin/iptables -I RH-Firewall-1-INPUT -m state --state NEW \
    -m udp -p udp --dport 48620 -j REJECT |
/sbin/iptables -I RH-Firewall-1-INPUT -m state --state NEW \
    -m udp -p udp --dport 48621 -j REJECT |
/sbin/iptables -I RH-Firewall-1-INPUT -m state --state NEW \
    -m udp -p udp --dport 48622 -j REJECT |
/sbin/iptables -I RH-Firewall-1-INPUT -m state --state NEW \
    -m udp -p udp --dport 48623 -j REJECT
```

- nfs_servicestop (used to stop nfs services)

```
/etc/init.d/portmap stop
/etc/init.d/nfslock stop
/etc/init.d/nfs stop
/etc/rc.d/init.d/netfs stop
```

- nfstop (to execute the 2 scripts above)

```
sh ~/tool/nfs_servicestop
sh ~/tool/nfs_firewallclose
```

COS/CCS 2008-14

www.kth.se