­­

PSAN3

**Remote Desktop Application for iPad**

by

Hua Guo, Yu Jiang and Haixiang Liu

**PSAN3**

Advised by

Prof. Pedro SANDER

Submitted in partial fulfillment

of the requirements for COMP 397

in the

Department of Computer Science and Engineering

The Hong Kong University of Science and Technology

2010-2011

Date of submission: September 21, 2010

Table of Contents

[1. Introduction 4](#_Toc272556876)

[1.1 Overview 4](#_Toc272556877)

[1.2 Objective 4](#_Toc272556878)

[1.3 Literature Survey 5](#_Toc272556879)

[1.3.1 Products Survey 5](#_Toc272556880)

[1.3.2 Technology Survey 5](#_Toc272556881)

[2. Methodology 6](#_Toc272556882)

[2.1 Design 6](#_Toc272556883)

[2.1.1 Transmission Protocol 6](#_Toc272556884)

[2.1.2 Software Architecture 6](#_Toc272556885)

[2.1.3 GUI 6](#_Toc272556886)

[2.1.4 Hand Gestures 7](#_Toc272556887)

[2.2 Implementation 7](#_Toc272556888)

[2.3 Testing 7](#_Toc272556889)

[2.4 Evaluation 8](#_Toc272556890)

[3. Project Planning 8](#_Toc272556891)

[3.1 Division of Work 8](#_Toc272556892)

[3.2 GANTT Chart 8](#_Toc272556893)

[4. Required Hardware & Software 9](#_Toc272556894)

[5. References 9](#_Toc272556895)

1. Introduction

1.1 Overview  
The newly released iPad has brought with it a new direction for improving productivity when working with computers: combining iPad’s portability and relatively large desktop space (compared to iPhone/iTouch) with PC/Mac’s computing power. This synergy could be created with the help of a type of applications, which are usually referred to as remote desktop applications.  
Remote desktop applications could be further divided into server side and client side applications, and with server side and client side applications installed on two different machines, the user on the client side can view the desktop of the server side computer and control it. Thus, with a remote desktop application installed on the iPad, the user could perform tasks on his iPad away from his desk, even if the tasks require computing power that exceeds the capability of iPad.  
Currently several products providing remote desktop services have been developed for iPad, such as Air Display, iDisplay, ScreenRecycler, Mocha VNC, VNC Viewer, iTeleport and MaxiVista. However, all these products suffer from problems such as lag, proper display or ease of use to different extents.  
Our goal is to first develop a remote desktop software on iPad that is easy to use, with smaller delay and then to extend the current functions of remote desktop control such as assisting multiple users getting access to a single PC game and developing more gestures representing different commands.

1.2 Objective  
The primary objective of this project is to allow the user to port their original desktop onto the iPad seamlessly and intuitively, under a tidy user Interface. Once the desktop components are on the iPad, the user should be able to perform the normal desktop controls as they would on their PC. One thing noteworthy is that when interacting with this iPad remote desktop application, mouse clicks for PC will be replaced by touch gestures. The overhead of transformation of finger gestures to actual commands and the transfer of data between iPad and PC should be negligible after an action is performed on the original display.   
  
The second main objective, which is also a distinguishing feature of this project, is to support multiple user access, i.e., applications on the PC will be simultaneously accessible to multiple users. Furthermore, modifications and gestures performed by each user will be immediately reflected on the iPads of the other users. To date, we have not found any existing products that support this kind of feature. This feature will be significantly useful for circumstances such as a collaborative business meeting, where everyone will be able to contribute concurrently, or a classroom, where students will be able to participate more actively by have control over the same application running on the main PC from their iPads.   
  
The final distinctive feature that we want to develop is to expand the set of gesture controls currently available on iPad to make it more user-friendly. For example, one setback that exists with remote desktop application is that the corner of the iPad screen is often insensitive to touch as the user attempts to close an application, and to allow better usage, we can develop a three-finger drag gesture that closes the current window.

1.3 Literature Survey

The literature survey consists of two parts: existing products survey and relevant technology survey. The products covered in products survey could be further categorized as either VNC client or RDP client.

### ****1.3.1 Products Survey****

**VNC Clients**

1.3.1.1 Air Display  
Air Display, developed by Avatron Software, allows its users to have an extra display for their Mac/Windows PC, using a unique dynamic compression algorithm based on a WiFi network. This is by far the most user-friendly extended monitor application providing relatively reliable functionalities. However, there still exists some defects such as a low speed and a grey screen appeared on the iPad when another iPhone is trying to connect to the PC.

1.3.1.2 iDisplay  
iDisplay enables a second monitor for Mac/PC, but suffers from severe user dissatisfaction described as “unbearably slow and choppy” and easy to crash.  
1.3.1.3 Mocha VNC   
Mocha VNC provides access to VNC server so that user can see an exact mirror of their PC from their iPhone or iPad. Its free version has been widely accepted by the users but not its paid version and it suffers from some problems as well. For example, it maintains a poorly structured interface and it fails to provide a stable connection if the distance between iPhone and PC exceeds around 20 meters.

1.3.1.4 iTeleport: Jaadu VNC   
iTeleport provides basically the same functionalities as Mocha VNC does, but it is more user-friendly because it provides more reliable and comprehensive gesture sets, and it will search for the nearby PCs automatically. Nevertheless, the current version of iTeleport can sometimes be buggy: it may crash or stop responding to some input demand occasionally. Another problem is that iTeleport is 4 times more expensive than the non-free version of Mocha VNC does.

1.3.1.5 MaxiVista  
MaxiVista provides a second monitor for windows PC, but it does not receive as good reputation as the previous ones in terms of either speed or reliability. Also, it can only align with the left edge of the iPad but not free placement.

**RDP Client**1.3.1.6 Mocha RDP  
Mocha RDP is a remote desktop app based on RDP protocol. Except that, it is very similar to Mocha VNC. It has several limitations including low color bit rate, absence of sound support and unattractive visual style.  
1.3.1.7 iTap  
iTap is the most powerful RDP client on iPad. It supports 32bit color with aero effect. But it is very expensive and hard to use for new users due to the complexity of its configuration.

1.3.2 Technology Survey

Several protocols related to remote desktop applications are around, with the two most popular ones being Remote Framebuffer Protocol (RFB) and Remote Desktop Protocol (RDP). RFB is a simple protocol for remote access to graphical user interface, and it is applicable to all windowing systems such as windows, macintosh and X11 since it works at the framebuffer level. Virtual Network Computing (VNC) refers to the type of remote desktop technology built on top of the RFB Protocol. RDP is a proprietary protocol developed by Microsoft, and in contrast, RFB is first developed [Olivetti Research Laboratory](http://en.wikipedia.org/wiki/Olivetti_Research_Laboratory) (ORL) in the idea of as simple as possible. There are limitations for both protocols.  
RDP is faster, requires less bandwidth, but more complex to implement and rendering must be performed on client side. Also, no GPU acceleration is allowed. There are only two RDP clients for iPads (Mocha RDP and iTap), and all have limitations in terms of their functionalities and visual styles.  
VNC/RFB is slower, which means to provide equivalently smooth experience, VNC requires more bandwidth than RDP does. The advantage of VNC is that it is simple to implement, and no client side rendering is needed. Several products are available for iPad that are based on VNC as mentioned before, and some of them are quite sophisticated, especially Mocha VNC.   
Considering the above, we plan to develop a remote desktop client on iPad based on RDP, since this approach is more promising in producing smoother experience and more additional functionalities. We will also explore the possibility of enabling GPU acceleration at an acceptable speed.

# 2. Methodology

2.1 Design  
The design phase will start in late September. It will cover the following aspects:

2.1.1 Transmission Protocol  
Our current plan is to follow the Remote Desktop Protocol as close as possible. However, considering the complexity of the protocol, it is likely that we will choose to work out a simplified version, only implementing the rules and functionalities which are essential in building our software. In this case, we will carefully consider which part of the RDP should be employed and produce a specification.

2.1.2 Software Architecture  
We will first decompose the software into several relatively independent modules, and then produce design specification for each module. The specification should describe the expected behavior of the module, including the input and output format. Since we will use Objective-c as the primary programming language which is object-oriented, the specification should also describe major classes and the class hierarchy.

2.1.3 GUI  
Though most of the functionalities should be achieved without explicit user intervention,  a user friendly GUI is still essential to provide a clean and smooth experience during client configuration process and client start/exit. The style and layout of the GUI will be determined during this phase.

2.1.4 Hand Gestures  
Inputs to iPad are conveyed through hand gestures, which is very different from the keyboard-and-mouse input scheme. One aim of our software is to provide simple, intuitive and powerful gestures, so that the users will have a smooth transition from keyboard and mouse to touch screen.

2.2 Implementation  
The first stage in the implementation phase is to build a prototype based on the existing Remote Desktop Protocol (RDP). We will evaluate the performance of the prototype and try different ways to improve speed.   
The program will be developed under MAC OS in the XCode environment using Objective-C. A simulator for iPad is available within Xcode, which could be used for testing purpose.  
The implementation will largely follow the design specifications produced in the design phase. According to the preliminary plan, the project will be divided into 4 parts:  
1.  Read and process the network data sent from the server.  
2.  Display the data processed on ipad screen.  
3.  Recognizing different input gestures correctly and sending corresponding commands back to the server.  
4.  Support different configurations.  
Trial and error is expected in this phase. It is possible that we will identify some unrealistic aspects in the design specifications, modify the design specs after discussion, and then continue the implementation with the updated design specs.

2.3 Testing

Different parts of this project have different bottlenecks, and the design of testing cases will be based on different parts.

For the networking part, minimizing the data flow to provide a smooth screen display is the pivotal goal. Therefore, in addition the correctness of the screen display, different configuration and requesting schemes also need to be tested and tuned for creating a smooth experience.

For the gesture part, creating an intuitive and simple input method to replace keyboard and mouse is the ultimate goal. In the beginning phase, different gestures will be created to replace the normal input methods of PC. In the later phase, we will try different gestures and try to choose the most user-friendly ones and refine them.

For the multi-user part, stability and lag can be the biggest issue. We will first test on correctness of different job dividing algorithms and evaluate their performance, then try to choose the most effective one.

At the final testing stage, more black box testing for will be performed, especially for special cases.

## 2.4 Evaluation

The final evaluation will be based on the following aspects:

1. Reliability: how fluent the image transmission and display are under normal circumstances

2. Accuracy: how accurate the gestures could be recognized

3. Consistency: how consistent the final product is with the objectives stated above in term of functionalities

4. Competitiveness: overall performance compared with similar applications

5. Usability: how classmates, professors and relatives judge the project

To evaluate the fourth and fifth aspect, we will conduct a user study, asking participant to try both our application and some other applications mentioned in 1.3.1 to obtain feedback.

# 3. Project Planning

## 3.1 Division of Work

|  |  |  |  |
| --- | --- | --- | --- |
|  | Hua Guo | Yu Jiang | Haixiang Liu |
| Overall Plan | A | A | L |
| Literature Review | L | A | A |
| Prototype | A | A | L |
| Project Proposal | A | L | A |
| Networking Module Design | L | A | A |
| Server Module Design | A | L | A |
| Client Control Module Design | A | A | L |
| Client User Input Module Design | A | L | A |
| Networking Module Implementation | L | A | A |
| Server Module Implementation | A | L | A |
| Client Control Module Implementation | A | A | L |
| Client User Input Module Implementation | A | L | A |
| Networking Module Unit Testing | L | A | A |
| Server Module Unit Testing | A | L | A |
| Client Control Module Unit Testing | A | A | L |
| Client User Input Module Unit Testing | A | L | A |
| System Testing | A | A | L |
| Implementing Optional Features | A | A | L |
| Progress Report | L | A | A |
| Final Testing | L | A | A |
| Evaluation and Fine Tuning | A | L | A |
| Final Report | A | A | L |
| Poster Design | A | L | A |
| Presentation and Demonstration Preparation | L | A | A |

Note: “L” stands for “leader” and “A” stands for “assistant”.

## 3.2 GANTT Chart



# 

# 4. Required Hardware & Software

## 4.1 Hardware

## Intel –based Mac – for running IDE

* Running Mac OS X 10.6 Snow Leopard
* PC – for running the server side application
* Running Windows Vista or Windows 7
* iPad – for running the client side application

## 4.2 Software

* XCode – IDE for developing applications for iPhone OS, including support for objective – c as the programming language

# 5. References

[1] Microsoft. (2010, Jan 06). Remote Desktop Protocol. *MSDN Library* [Online]. Available: <http://msdn.microsoft.com/en-us/library/ee484339.aspx>

[2] RealVNC Ltd. (2009, Nov 24). The RFB Protocol. *RealVNC* [Online]. Available: <http://www.ieee.org/portal/cms_docs_iportals/iportals/publications/authors/transjnl/stylemanual.pdf>

[3] Wikipedia. (2010, Sep 14). RFB Protocol. *Wikipedia*[Online]. Available: <http://en.wikipedia.org/wiki/RFB_protocol>