UNIVERSITY OF CINCINNATI

Date: July 20, 2005

I, Wen Huang **,**

hereby submit this work as part of the requirements for the degree of:

Master of Science

in:

Computer Science

It is entitled:

Development of Web-browser Based Thin-client Low Bandwidth Workspace Sharing System with Real Time Display Of Remote Mouse Movement

This work and its defense approved by:

Chair: Dr. Yizong Cheng

Dr. Dieter Schmidt

Dr. Raj Bhatnagar

Development of Web-browser Based Thin-client Low Bandwidth Workspace Sharing System with Real Time Display of Remote Mouse Movement

A thesis submitted to the

Division of Research and Advanced Studies
of the University of Cincinnati

in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE

in the Department of Electrical and Computer Engineering and Computer Science
of the College of Engineering

2005

by

Wen Huang

B.S., People's University of China, Beijing, 1988

Committee Chair: Dr. Yizong Cheng

Abstract

Desktop sharing is a key function in Internet conferencing solutions. There are many such software products available in this emerging market. The key players are WebEx, Microsoft (with NetMeeting), Citrix (with GotoMeeting), to name just a few. These commercial products allows remote users to view the same desktop as the host's, allowing remote presentation and training, remote technical support and troubleshooting, and so on. These commercial products are fully featured, enabling remote users to share the whole desktop content in real time. But the disadvantages are slow transmission and hefty cost, making it inconvenient for general purpose.

Due to the shortcomings of current products, such as high cost, complicated installation, slow remote desktop display, no true color capability, one-way sharing and security concern, there is a need to develop a better communication tool that can address these issues and meet the needs of certain user groups. I took the initiative to develop such an innovative system. This system, called Web-browser Based Thin-client Low Bandwidth Workspace Sharing System with Real Time Display of Remote Mouse Movement, is now widely used by Triton Global Sources, Inc., an Internet-based manufacturing outsourcing agent. It is highly commended by Triton's staff and its customers all over the country.

The thesis illustrates the technology in developing this system and a methodology to improve viewing comfort and effectiveness, utilizing JAVA Servlet/Tomcat, XML&DOM, VML, object-oriented JavaScript, etc. The following are two main research areas:

1. Development of a Workspace Sharing System with Real Time Display of Remote Mouse Movement. With this system two remote users can simultaneously view each other's mouse movement over a still picture. This is a server-side environment, which requires no client-side download of any special program. The picture to be viewed by both parties is pre-loaded to each individual's computer and requires no refreshing. The only constantly refreshed data is the mouse position, which is very small in size. This logic makes the mouse position refresh more frequently and allows two remote users to see each other's mouse movement simultaneously.

2. Improvement for viewing under Internet traffic jamming condition. This is particularly useful for inter-continental communication where Internet traffic is limited by undersea cable capacity. Two methods are explored:

    A. Data Buffer/Delay – Mouse position data are received (GET) and sent out (SEND) in a larger interval so that the chance of losing data is reduced. Instead of sending a single mouse point, a sequential of mouse points are buffered first, then sent out as a group. Although this method sacrifices real-time effect by a little bit, the result of continuity is significantly improved.

    B. Data Interpolation – Mouse points are sparsely sampled and sent out in order to reduce the amount of the transmitted data. When the other client receives the mouse position data, it performs a data interpolation to regenerate more points before displaying them. So the user can see a more continuous mouse movement. Using this method, the

number of sampled mouse points is reduced while the frequency of SEND and GET operation is decreased, which helps to prevent the mouse data from getting lost at server.

Above two improvements are considered from two different view angles. In A, the improvement is done before the mouse position data are sent out to the server. In B, the improvement is done after the mouse position data are received but before they are displayed.

Keywords: Remote desktop sharing, web-based screen sharing

Acknowledgement

I would like to express my sincerest gratitude to my advisor Dr. Yizong Cheng for his guidance, encouragement and kindness. His insightful comments and inspiration have been invaluable throughout the preparation of this thesis.

I would also like to thank Dr. Dieter Schmidt and Dr. Raj K. Bhatnagar, who have spent their precious time in their busy schedule to read the draft of this thesis thoroughly and serve as my defense committee members.

I am also very grateful to my parents for their long-distance support and caring. Without their encouragement I might not even have the courage and determination to continue my thesis work.

Finally, I would like to give my special thanks to my husband Junping He, whose patient love, understanding and sacrifice enabled me to complete this work. I also want to thank my son Robin for his help proofreading my thesis and PowerPoint presentation, and my little daughter Megan, whose sweet smiles always cheered me up and gave me relief during this difficult time.

Table of Content

LIST OF FIGURES

CHAPTER 1 INTRODUCTION

Desktop sharing is a key function in Internet conferencing solutions. There are many such software products available in this emerging market. The key players are WebEx, Microsoft (with NetMeeting), Citrix (with GotoMeeting), to name just a few. These commercial products allows remote users to view the same desktop as the host's, allowing remote presentation and training, remote technical support and troubleshooting, and so on. These commercial products are fully featured, enabling remote users to share the whole desktop content in real time.

However, in order to achieve the best performance for commercial use, these companies have to build special infrastructure based on client server architecture. Despite many powerful features, there exist quite a few drawbacks in these commercial products. One of the drawbacks is that the client user has to download a special heavy program and install it on their own computer. The inconvenience of client-side download, dependence on high bandwidth of Internet data, and the hefty usage charge limit the popularization among massive low-end users.

The project revealed in this thesis is particularly to address the problems identified from above – how can we develop a product that is light and easy to use? This became an urgent need for Triton Global Sources, Inc., whose business requires frequent technical communication with its customers all over the country and around the world. Triton finds it very cumbersome to use the desktop sharing software mentioned above to communicate with its customers. They had tried some of them, but felt none of them fit their need. Triton's customers are mostly quality engineers, product engineers, or mechanical design engineers. They are neither IT professionals

nor computer savvy. These people, in general, are either lack of patience or slow to learn the technology. They are afraid, or hate to follow the computer setup procedures. They are often intolerant for the slow desktop displaying. Many people are also alerted whenever they are required to download something in their computer. Moreover, Triton's contact people are often changed: they might switch roles or switch jobs. So Triton has to constantly deal with new contacts with different computers. The scenario would be like this: for a one-minute-long simple question to discuss, it may take up to 10 minutes to instruct the client to set up the software. This is really a headache for Triton. Realizing that there is a unique need for certain user groups, I used a new logic to design a so-called Web-browser Based Thin-client Low Bandwidth Workspace Sharing System with Real Time Display of Remote Mouse Movement. The system is now widely used by Triton staff and their customers and is highly commended by everyone for its ease of use.

The thesis illustrates the technology in developing this system and a methodology to improve viewing comfort and effectiveness, utilizing JAVA Servlet/Tomcat, XML&DOM, VML, Object-oriented JavaScript. The following are two main research areas:

1. Development of a Workspace Sharing System with Real Time Display of Remote Mouse Movement. The remote users can simultaneously view each other's mouse movement over a JPG/GIF picture. This is mainly a thin client environment, which requires no client-side download of any special program. The picture to be viewed by both parties requires no refreshing. The only constantly and automatically refreshed data is the mouse position,

which is very small in bytes. Compared to traditional remote desktop sharing, this system significantly reduces the amount of data transmitted since what is transmitted is only mouse position instead of the whole desktop image. This results in faster data transferring between two remote computers. Moreover, while conventional desktop sharing is one way sharing, i.e. Client 1 can see Client 2's desktop, but Client 2 can't see Client 1 at the same time, this system allows two remote users to see each other's mouse movement over the same image simultaneously which we call two way sharing.

2. Improvement for viewing under Internet traffic jamming condition. This is particularly useful for inter-continental communication where Internet traffic is limited by undersea cable capacity. Two methods are explored:

   A. Data Buffer/Delay – Mouse positions are received (GET) and sent out (SEND) in a larger interval so that the chance of losing data is reduced. Instead of sending a single point, a sequential of mouse positions are buffered first, and then sent out as a group. Although this method sacrifices real-time effect by a little bit, the result of continuity is significantly improved.

   B. Data Interpolation – Mouse points are sparsely sampled and sent out in order to reduce the amount of the transmitted data. When the other client receives the mouse position data, it performs a data interpolation to generate more points before displaying them. So the user can see a more continuous mouse movement.

The structure of each chapter is as following:

Chapter 2 briefly reviews current remote desktop sharing products and their advantages and disadvantages.

Chapter 3 discusses the system requirement and the project background

Chapter 4 reviews software platform such as Apache Tomcat, Servlet, XML&DOM, VML etc.

Chapter 5 discusses system architecture, system functionality and workflow, and explains the main procedures in details.

Chapter 6 addresses the Internet traffic jam situation and explains how the advanced work improves the viewing comfort.

Chapter 7 further explores buffer optimization and the relationship between Internet speed and refreshing frequency.

Chapter 8 concludes the thesis and outlines areas for future study.

Chapter 2 Overview of Current Desktop Sharing Products

As broadband Internet becomes more and more popular and Internet related products mushroom, more and more people are working remotely, either at different work sites, or during traveling or at home, for convenience and/or for cost reduction. In order for the remote users to communicate effectively, a remote communication system must meet the following basic requirement:

1. They can see the same thing;
2. They can talk with each other;
3. They can see each other's pointing spot.

Item 3 is the most important feature and the key topic of this thesis. Without this capability it is not a true or effective remote communication tool. There are many commercial desktop-sharing products available in the market. The typical makers and their products are listed below.

Manufacturer	**Product**
Microsoft	NetMeeting
Webex	Webex
Citrix	GotoMeeting/GotoAssist

These and other unlisted products are all similar in functionalities. The most common functionalities of these products are as following:

- Presentation -

  Allows a company to present new product information, marketing and sales strategy to its customer online. The remote customers can see the presentation on their own computer as well as the presenter's action.

- Online Training -

  Allows a trainer to give training in a virtual classroom. The remote users can see the trainer's desktop and follow the presentation easily just as if he is standing in front of them. The trainer can reach more people, more frequently, without the costs of travel.

- Remote Technical Support -

  Allows an IT professional to access and control a remote client's computer to fix the problem. With remote user's permission, the IT professional can gather a standard set of information about the end user's environment, including operating system details; total and available memory; applications and services currently running; or can gather additional custom system or application information specific to his support needs, such as version details for applications, key information from the registry and the contents of specific text files, and more. Based on this information, the IT professional could be able to fix the problems.

All of these commercial products require a download of supporting software of at least a few hundred kilobytes on the client's computer. Many companies claim their software doesn't require download, but the truth is that the download is done on the background without user's manipulation. For example, Glance claims that guest never download, but in fact, each time a remote user joins the sharing session, the user has to wait for about 50 seconds for the java applet

to download on the background. Without this download, it is impossible to handle the speedy desktop image transferring.

The other common feature is, all the remote desktop sharing is one way, i.e. it only allows one remote user, normally an IT professional, or the initiator, to control or see the other remote user's desktop, not vise versa.

These desktop sharing systems are generally not suitable for dial-up connection because traffic tends to burst up to several hundred kilobytes-per-second for several seconds whenever the screen changes. Many companies claims their system can work with dial-up connection, but they also admit that the user will experience greater latency, i.e., there will be a longer time lag during each screen change.

Moreover, Microsoft Net Meeting admits that true color sharing capability does not work during its Remote Desktop Sharing sessions. The reason is that in order to transfer desktop image in real-time, the image size must be kept small. With true color image, whose size can go up to several hundred kilobytes, it is hard to cope with the average Internet traffic speed. That is why 100% companies use low bit (8-bit, 12-bit) pseudo color image for shared desktop image, even though they claimed using patented compression technology and the screen capture algorithm.

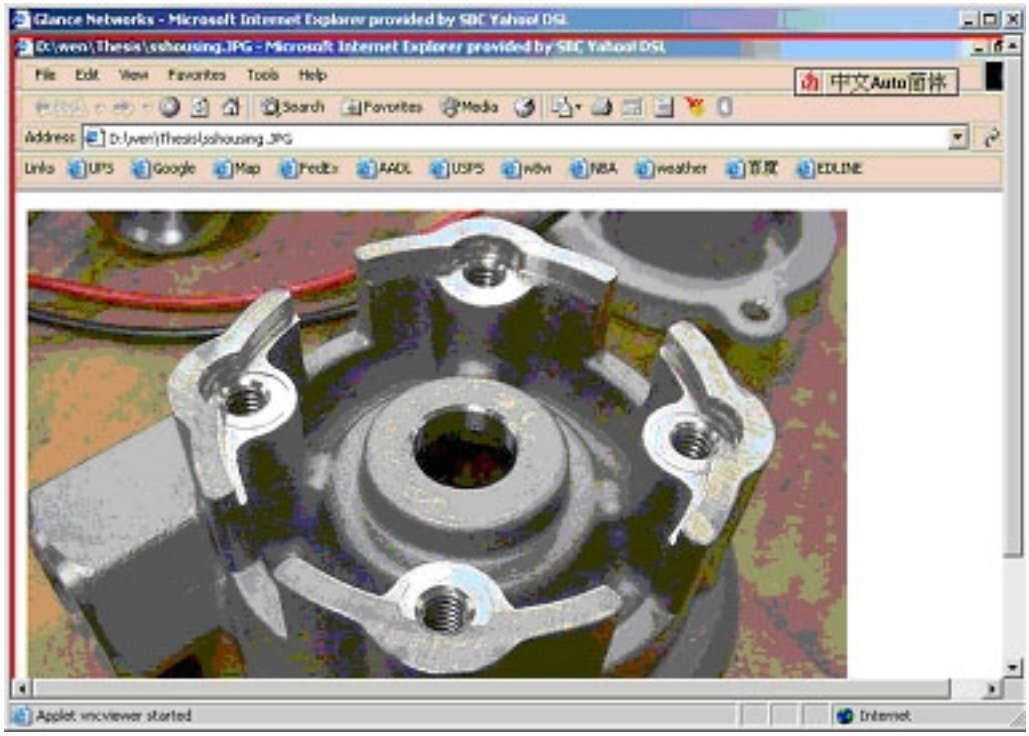

Figure 1 Pseudo Color Image

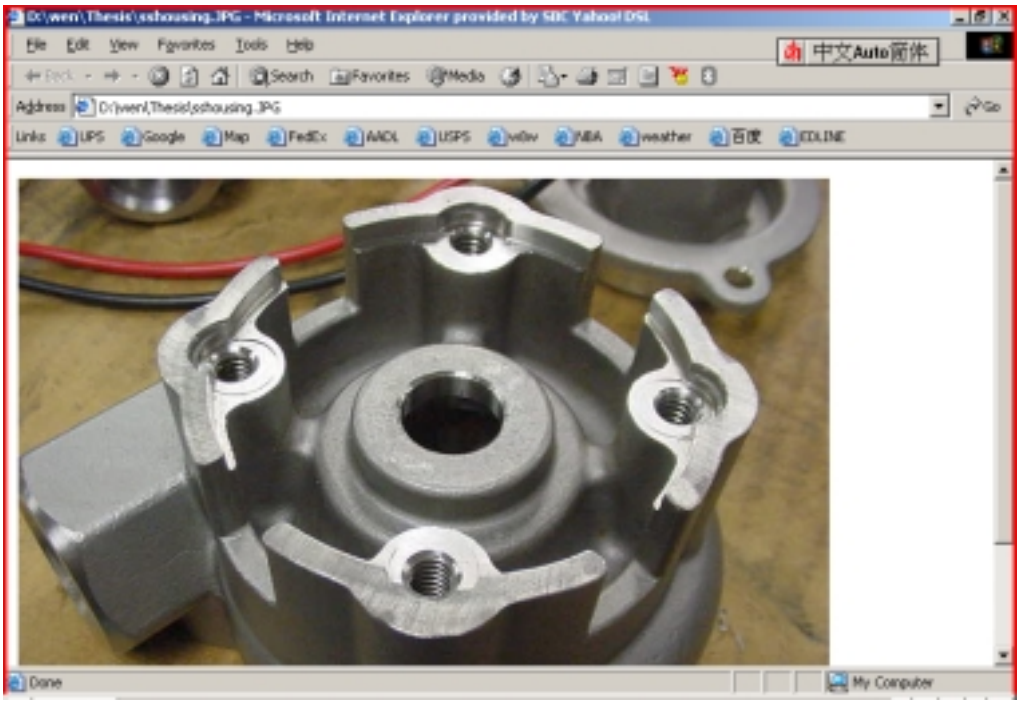

Figure 2 True Color Image

It is proved that these commercial products are ideal for collaboration within a corporation where Internet infrastructure can be best laid out and employees can receive proper training before using the system.

For example, as Microsoft claims, Boeing designers, engineers, managers, supervisors and machinists use NetMeeting to collaborate on creating two demonstrator jet fighters; Deere & Company employees worldwide use NetMeeting to share information and work together in real-time with standards-based features, such as application sharing and electronic whiteboards, all from their desktop PC's. Ford Motor Company takes advantage of NetMeeting's powerful tools for effective communication and collaboration over their corporate intranet. And so on.

However, for unexpected business-to-business communication it would be difficult to use these sophisticated desktop-sharing systems because the remote users are mostly low end users who are neither properly trained, nor computer savvy. Plus, the contact person may be constantly changed. On the other hand, for the security concern, the corporate intranet may not generously allow an outsider to access to their system.

In summary, despite the advantages, the drawbacks of these commercial products are the roadblocks for popularizing the use of remote desktop sharing. The major drawbacks are:

1. High cost: one-time software purchase, license fee, by-minute charge,
2. Complicated installation: not suitable for constantly changed user group or their computers.
3. Slow or latent remote desktop display.
4. No true color desktop display.

5. One-way sharing – cannot perform mutual sharing.

6. Security issues: client program is prone to virus/hacker attacks.

CHAPTER 3 Workspace Sharing System with Real Time Display of Remote Mouse Movement

3.1 The Unique Need

As mentioned in chapter 2, there really exists a unique need for a better remote communication tool for certain user groups. In this thesis, I'd like to introduce a particular user group and their need. Triton is an internet-based manufacturing outsourcing agent with its supplier base in China and clients all over the country and around the world. As a sourcing agent, Triton staff needs to discuss blue prints or photos of products or manufacturing process with their remote suppliers or customers all the time. Unlike a high-tech company, Triton is in the traditional low-margin manufacturing business. In order to cut cost, they strongly feel they need a simpler and easier-to-use tool to communicate with their customers and suppliers. They first tried Microsoft NetMeeting and later Webex. However the cost, the complicated set-up and the potential hacker attack on client's system does not make it a viable option at all. Especially recently Internet virus and Spyware program make many companies nervous on any outside program. For small size businesses without dedicated IT staff, this becomes a major issue.

Based on all these issues and concerns, I believe there exists a large demand for a unique system that can fulfill the following requirements:

1. Extremely simple to use, no setup, no download;
2. Zero or very low cost;
3. Fast – no interrupted or slow image display or mouse movement

4. True color, no distorted or simplified color. Since the users need to discuss the real part, true color must be displayed on the computer screen of both parties.
5. Two way viewing. Remote two parties can see each other's mouse movement at the same time.

3.2 Project Background

The above requirement seems to be very challenging. If we were still having current desktop sharing concept in our mind to tackle this problem, we would probably never have a breakthrough.

By studying Triton's real need, I soon jumped out of this standard desktop sharing box -- Triton's remote communication is actually not requiring a full-functioned desktop sharing: it is neither a fancy PowerPoint presentation nor a computer helpdesk system. All it needs is to discuss blue prints or still pictures with their remote customers or suppliers. Triton wanted to be able to see the other side's mouse movement so that they know exactly where the other side is pointing at, and vice versa. The application does not require sharing a whole desktop image or a constantly changed desktop image, but just a static image (print or photo) along with each other's mouse movement. With this unique need in mind, I designed a different desktop sharing system. To differentiate it with other conventional desktop sharing products, I named it Web-browser Based Thin-client Low Bandwidth Workspace Sharing System with Real Time Display of Remote Mouse Movement. The system is innovative in the sense that it breaks down and simplifies a complicated process, identifies and separates each function and finally uses the best techniques to accomplish it. In this system, the mouse movement sharing, image sharing and

voice communication are all separated and tackled with different techniques. This unique logic effectively avoids the complication caused by the bundled requirements and maximizes the share-viewing performance.

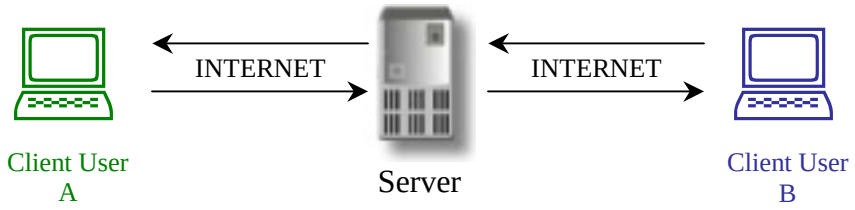

Figure 3 Client / Server Architecture

CHAPTER 4 SOFTWARE PLATFORMS REVIEW

In this chapter, the following system implementation environment and tools are reviewed: Apache Tomcat web server, Java Servlet, XML, VML and Object-oriented JavaScript.

4.1 Apache Tomcat

Tomcat is the Servlet container that is used in the official Reference Implementation for the Java Servlet and JavaServer Pages technologies. As part of Apache's open source Jakarta project, it has nearly become the industry accepted standard reference implementation for both the Servlets and JSP API. Tomcat can function both as a web server in its own right, and in conjunction with other servers such as Apache and IIS.

The version used in this system is Tomcat 4.1.30. Tomcat 4 implements the Servlet 2.3 and JavaServer Pages 1.2 specifications from Java Software.

4.2 Servlet

Servlets are the Java platform technology of choice for extending and enhancing Web servers. Servlets provide a component-based, platform-independent method for building Web-based applications, without the performance limitations of CGI programs. And unlike proprietary server extension mechanisms (such as the Netscape Server API or Apache modules), Servlets are server and platform independent. Today Servlets are a popular choice for building interactive Web applications.

Servlets have access to the entire family of Java APIs, including the JDBC API to access enterprise databases. Servlets can also access a library of HTTP-specific calls and receive all the benefits of the mature Java language, including portability, performance, reusability, and crash protection.

Servlets make use of the Java standard extension classes in the packages javax.servlet (the basic Servlet framework) and javax.servlet.http (extensions of the Servlet framework for Servlets that answer HTTP requests). Since Servlets are written in the highly portable Java language and follow a standard framework, they provide a means to create sophisticated server extensions in a server and operating system independent way.

Typical uses for HTTP Servlets include:

- Processing and/or storing data submitted by an HTML form.
- Providing dynamic content, e.g. returning the results of a database query to the client.
- Managing state information on top of the stateless HTTP, e.g. for an online shopping cart system which manages shopping carts for many concurrent customers and maps every request to the right customer.

Java Servlets also offer a number of advantages over Java applets. Java applets require the Java code to be run in the user's browser, and suffer from complicated compatibility problems due to differences between browsers and possibly slow startup due to the need to download many

compiled classes into the browser. Java Servlets, since they run in the server, work well with any browsers and do not result in any code being downloaded to the user's machine.

It is for this particular reason that Java Servlet, instead of Java Applet, was used in this system design.

4.3 XML & DOM

XML, the Extensible Markup Language, is a W3C-endorsed standard for document markup. It defines a generic syntax used to mark up data with simple, human-readable tags. It provides a standard format for computer documents. It allows designers to create their own customized tags, enabling the definition, transmission, validation, and interpretation of data between applications and between organizations.

XML specifies neither semantics nor a tag set. In fact XML is really a meta-language for describing markup languages. In other words, XML provides a facility to define tags and the structural relationships between them. Since there's no predefined tag set, there can't be any preconceived semantics. All of the semantics of an XML document will either be defined by the applications that process them or by Stylesheets.

XML is an extremely flexible format for data. In theory, any data that can be stored in a computer can be stored in XML format. In practice, XML is suitable for storing and exchanging any data that can be plausibly be encoded as text.

The XML Document Object Model (DOM) is a programming interface for XML documents. As a W3C specification, it defines the way an XML document can be accessed and manipulated. The XML DOM is designed to be used with any programming language and any operating system.

With the XML DOM, a programmer can create an XML document, navigate its structure, and add, modify, or delete its elements. The DOM represents a tree view of the XML document.

The Microsoft XML parser supports all the necessary functions to traverse the node tree, access the nodes and their attribute values, insert and delete nodes, and convert the node tree back to XML.

4.4 VML

VML**,** Microsoft's Vector Markup Language, is an XML application for vector graphics that can be embedded in Web pages in place of the bitmapped GIF and JPEG images loaded by HTML's IMG element. VML provides faster graphic downloads and a better user experience. It allows the delivery of high-quality, fully integrated, scalable vector graphics to the Web, in an open text-based format. Rather than referencing graphics as external files, VML graphics are delivered inline with the HTML page, allowing them to interact and scale with user interaction.

Vector graphics take up less space and thus display much faster over slow network connections than traditional GIF and JPEG bitmap images.

VML was designed to help developers address problems with binary graphic files (GIFs, JPGs, and so on). These limitations of binary graphic files include:

- Their large file size causes them to download slowly, especially over slow links
- They're difficult to maintain and change
- They're external to the rest of the HTML document, making them difficult to distribute

VML is currently supported by Microsoft Internet Explorer 5 or greater for Windows® 95, Windows 98, and Window NT® 4.0 or greater.

4.5 Object-Oriented JavaScript

JavaScript, as with many other modern-day computer languages, is *object-oriented*. This means you can create these special named collections of properties that include variables and special built-in functions, called *methods*. These objects can make working with chunks of data quite a bit easier.

In more practical terms, an object is a named container of sorts. In most cases, you create an object with a special function, in a way that's similar to creating an array. You can then refer to the object using special notation that enables you to dig into the object and access certain variables or methods.

You can create self-contained JavaScript objects, storing and manipulating the data on the client side, rather than the procedural code. This also allows you to reuse code more often.

CHAPTER 5 SYSTEM ARCHITECTURE DESIGN

This chapter presents the system architecture and functionality of the Workspace Sharing System with Real Time Display of Remote Mouse Movement.

5.1 System Architecture

The proposed Workspace Sharing System has a client / server architecture as in Figure 4.

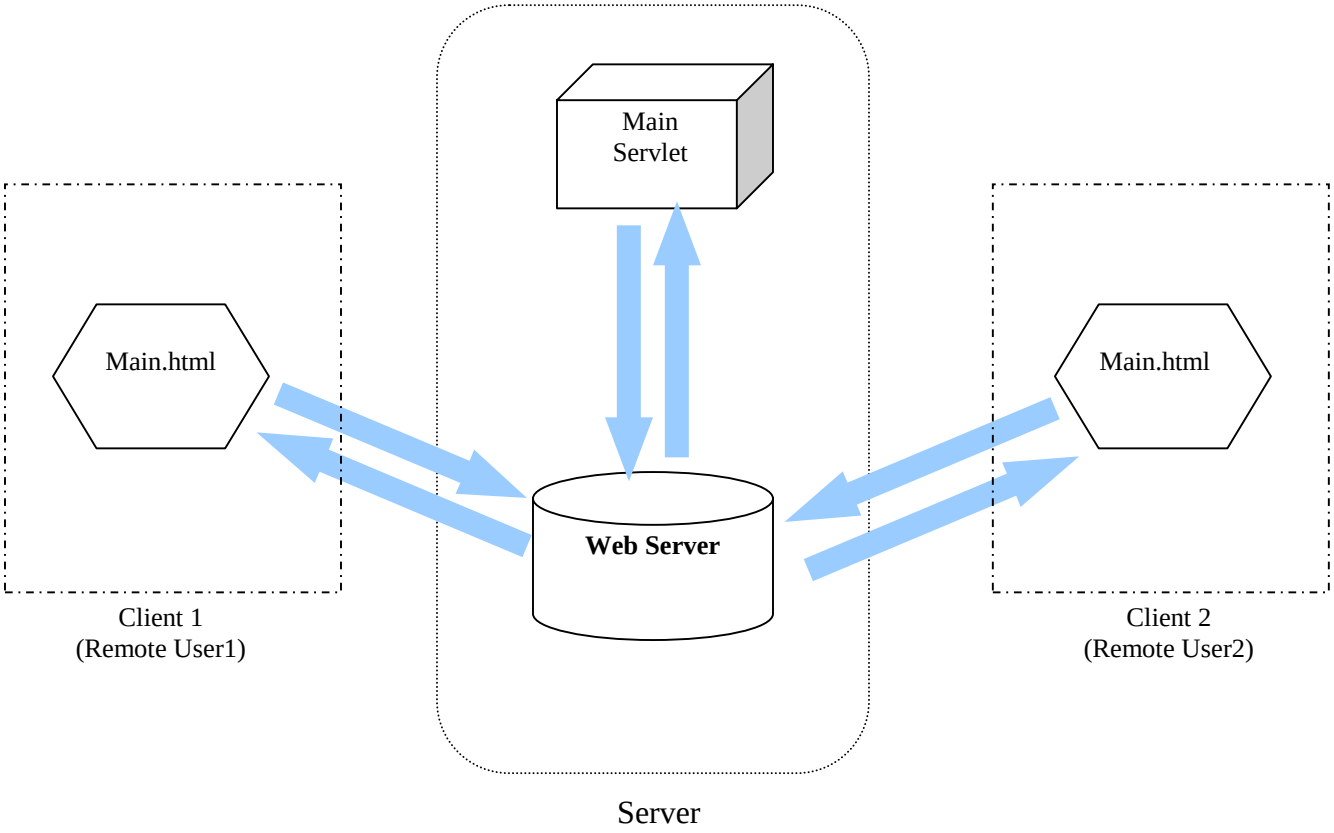

Figure 4 System Architecture

5.2 Design Issues and Solutions

In the system design, a few decisions need to be made, among which the following three items are highlighted and explained:

1. Real-time capture of the mouse trail

   I first thought of the most straightforward design - simply refreshing the web page to capture the mouse movement. But this would cause large data transmission. Since the image picture we are looking at is a static one, the only dynamic part is the mouse position, we only need to refresh the mouse position to reduce the data transmit amount. It is a waste of resource to refresh the page again and again. So I started to think about Hidden IFrame technique which can refresh a small portion of the whole page. This could greatly reduce the transfer speed and improved the viewing performance. However after some tests on different versions of IE browser, I found that the result is varied and not consistent when working in different versions of IE browser. Finally I located Microsoft's XML DOM (Document Object Model) technology to access and manipulate the data. Using this method, we can achieve the high frequency refreshing. The other very efficient strategy I used in my system design is: at the same time of the client sending (SEND) the mouse position to the server, the client also gets (GET) the other client's mouse information back. This doubles the efficiency of capturing the mouse positions.

2. Transparent display of mouse

   To display the other client's mouse position, I use a mouse image to simulate the other site's mouse. At the beginning I used a solid image in JPG format. But the mouse image

would block the picture being viewed. After changing to GIF format from JPG and making the mouse image transparent, the problem was solved.

3. Displaying simple graphs (rectangle and oval)

   In our system there is a simple graphing tool. Using this tool we can select an area of interest by drawing rectangle (includes square) or oval (includes circle). At the beginning, I tried to use a transparent rectangle (or an oval) GIF file. Through changing the size of the GIF file, changing the position and turning on/off the image, we draw rectangle (or an oval) on the webpage. However, the shortcoming by using this scaled image is, when the rectangle (or an oval) image is small, the image border becomes too thin, causing the rectangle (or an oval) invisible; when the image is big the border becomes too wide that the rectangle (or an oval) looks ugly. After some investigation and research, I found out VML is the best solution to implement the real time graphing. VML draw graph with vectors. In our case, the vectors are the mouse positions. When you click the mouse twice, the size and the position of the graph is defined. The simple graphing is easy to handle using VML.

   In order to identify your own graph from the other client's graph, I defined two VML rectangles, one is blue and one is red. The blue one is yours and the red one is the other client's.

5.3 System Functionality and Work Flow

The system functionality and workflow is illustrated in Figure 5.

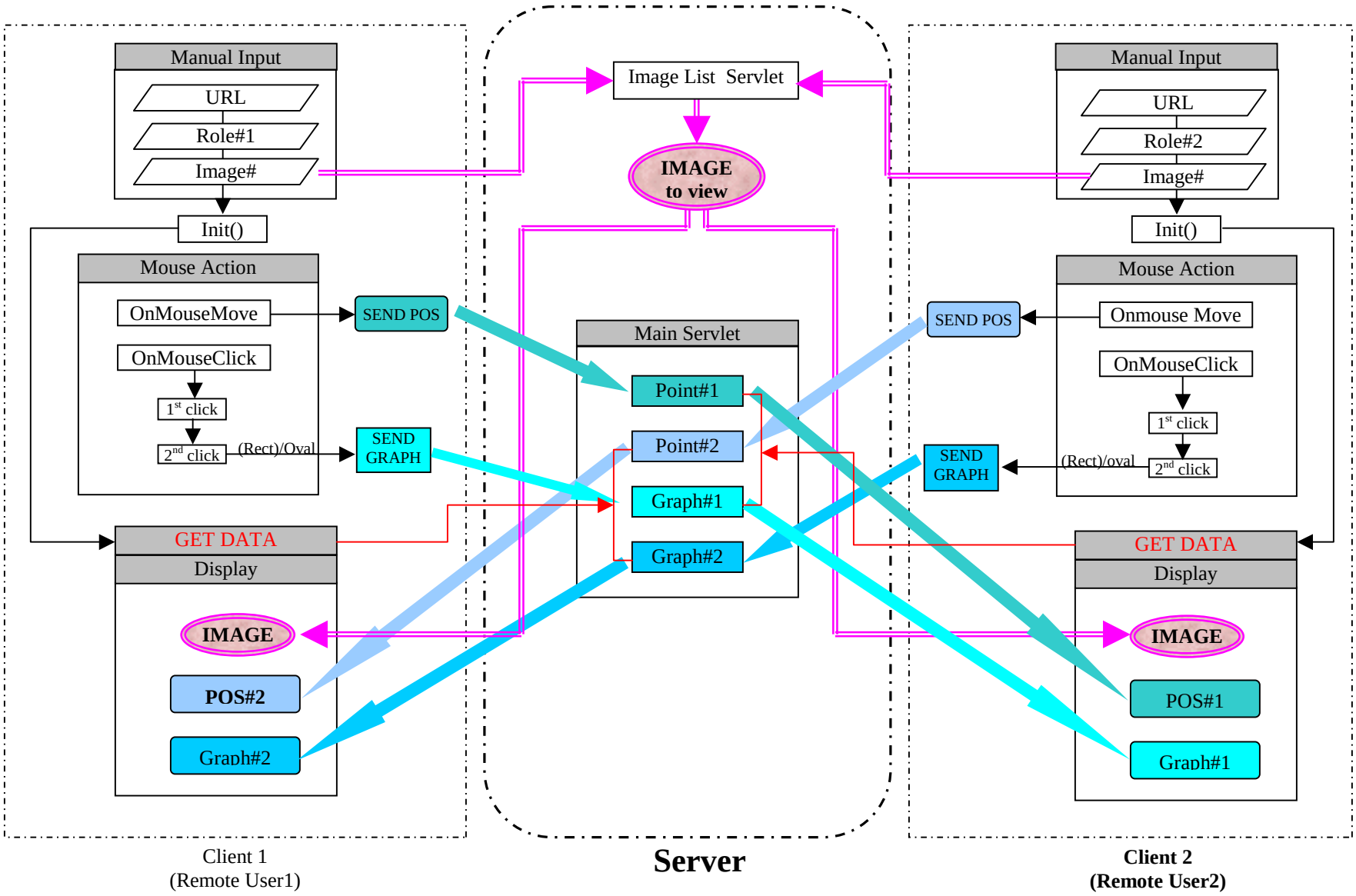

Figure 5 System Functionality & Workflow (Basic Design)

5.4 Function Descriptions

The following sections are the detailed explanation for the major system components:

5.4.1 Web page HTML files

A) Index.html

Index.html is the portal page both clients access from their browsers. It contains a form, asking the users to choose a “role”, value of 1 or 2. The “role” is a name we give to the clients. One party chooses role = 1, the other chooses role = 2. When the mouse positions are being transferred to the server, role number is also passed to the server as a parameter, so the server always knows where the information comes from, and is able to send the information to the other side.

After choosing the role and submitting the form, FileLister Servlet was called. FileLister.java will return to the browsers the list of all the pictures stored in the specified folder in the server. Users click and choose the same picture from the list. Then the users are directed to main.html page.

B) Main.html

This DHTML page is the main page both sides will work on. Users are able to see each other’s mouse movements and drawings over the same picture. Layer concept is used in the page design. They can. The page contains the following layers:

- DrawingType form:

  The user can choose the drawing type. Two types can be chosen, rec (rectangle) and oval (circle). rec is the default value.

- ImageLayer:

  It holds the picture which the users have chosen to discuss on.

  ```
  <div id="ImageLayer" align='left' onmousemove='sendPosition()'  onmousedown="clicked()"
  onmouseout="mouseout()"></div>
  ```

- MouseLayer:

  It holds the mouse image to trace other side's mouse movement. When Client 1 gets Client 2's mouse position coordinates (x, y), this layer with mouse image is shown in the position(x, y) in the browser. So by resetting mouse image in the different position, Client 1 can see Client 2's mouse movement in real time. Same as Client 2.

  ```
  <div id="MouseLayer" style='position:absolute; visibility:hidden;'>
  <img src="cursor.gif"  border="0"  width="24" height="24" >
  </div>
  ```

- CircleLayer:

  It holds the blue circle/oval VML graphic -. This layer is turned on and the blue circle/oval is drawn when the users choose the oval drawing type and click mouse twice over the picture.

  ```
  <vml:oval id="CircleLayer"  style="position: absolute;visibility:hidden;"  filled="false"
  strokecolor="blue"   strokeweight="1" >
  </vml:oval>
  ```

- SquareLayer:

It holds the blue rectangle VML graphic. This layer is turned on and the blue rectangle is drawn when the users choose the rectangle drawing type and click mouse twice over the picture.

```
<vml:rect id="SquareLayer" style="position: absolute;visibility:hidden;"   filled="false"
strokecolor="blue" strokeweight="1">
</vml:rect>
```

- OutCircleLayer:

  It holds the red circle/oval VML graphic. This layer is turned on and the red circle/oval is drawn when the other client is drawing the circle/oval.

  ```
  <vml:oval id="OutCircleLayer"  style="position: absolute;visibility:hidden;"    filled="false"
  strokecolor="red" strokeweight="1" >
  </vml:oval>
  ```

- OutSquareLayer:

  It holds the red rectangle VML graphic. This layer is turned on and the red rectangle is drawn when the other client is drawing the rectangle.

  ```
  <vml:rect id="OutSquareLayer" style="position: absolute;visibility:hidden;"   filled="false"
  strokecolor="red" strokeweight="1">
  </vml:rect>
  ```

  The refreshing of the mouse position is actually the layer page refreshing and so does the refreshing of the drawing.

  The Figure 6 is the illustration of the layer design.

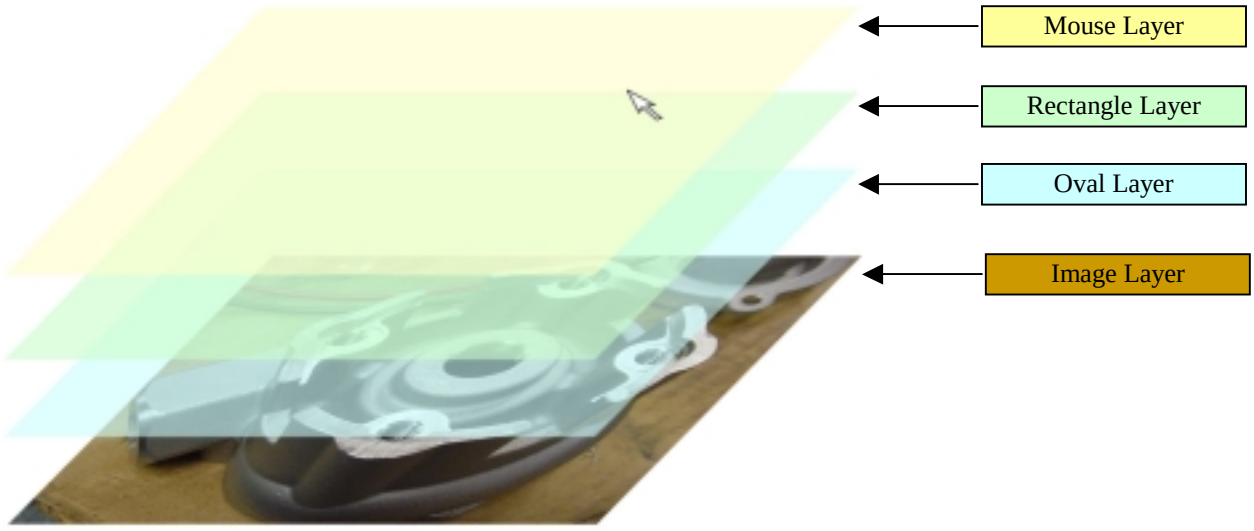

Figure 6 Layer Illustration

5.4.2 JavaScript files

There are two java script files in the program.

A) MouseMovement.js

This file defines a JavaScript class called MouseMovement. It stores mouse and graph information and defines send/get mouse and graph information methods.

class properties:

x, y: mouse position

top, left, width, height: describe oval or rectangle

type: drawing type, rectangle or oval

class methods:

- getData(role):

It calls Servlet main.java, passing parameter action=get. The Servlet returns back the other role's mouse position and graph information in XML format.

- sendData(role):

It calls Servlet main.java, passing parameter action=put and mouse position (x, y)

- sendGraph(role):

It calls Servlet main.java, pass parameter action=put and drawing information (top, left, width, height).

- MouseMovement_refresh(url):

  Load an XML document from the specified url, and replaces the contents of the specified XML object with the downloaded XML data. Then store the information back the object MouseMovement.

  The following is the class definition.

```
function MouseMovement(x,y){
        this.x= x;
        this.y = y;
        this.top=0;
        this.left=0
        this.width=0;
        this.height=0;
        this.type='rect';
        this.sendData=MouseMovement_sendData;
        this.getData=MouseMovement_getData;
        this.sendGraph=MouseMovement_sendGraph;
        this.refresh=MouseMovement_refresh;
}

function MouseMovement_sendData(role){
        this.refresh(servletRoot+"/Main?sAction=put&x="+this.x+"&y="+this.y+"&role="+role);
}
```

```
function MouseMovement_getData(role){
	this.refresh(servletRoot+"/Main?sAction=get&role="+role);
}

function MouseMovement_sendGraph(role){
	this.refresh(servletRoot+"/Main?sAction=putGraph&top="+this.top+"&left="+this.left+"&role
="+role+"&width="+this.width+"&height="+this.height+"&type="+this.type);
}

function MouseMovement_refresh(url){
try{
		var oXML=new ActiveXObject("Microsoft.XMLDOM");
		oXML.async=false;
		oXML.load(url);
		var oNode = oXML.documentElement.selectSingleNode("X");
		if(oNode!=null){
			this.x = oNode.text;
		}
		oNode =oXML.documentElement.selectSingleNode("Y");
		if(oNode!=null){
			this.y = oNode.text;
		}
		oNode =oXML.documentElement.selectSingleNode("L");
		if(oNode!=null){
			this.left = oNode.text;
		}
		oNode =oXML.documentElement.selectSingleNode("T");
		if(oNode!=null){
			this.top = oNode.text;
		}
		oNode =oXML.documentElement.selectSingleNode("H");
		if(oNode!=null){
			this.height = oNode.text;
		}
		oNode =oXML.documentElement.selectSingleNode("W");
		if(oNode!=null){
			this.width = oNode.text;
		}
		oNode =oXML.documentElement.selectSingleNode("P");
		if(oNode!=null){
			this.type = oNode.text;
		}
	}
	catch(e){
		window.status=e.description;
		return;
	}
}
```

B) MouseCapture.js

This file is the place where all the JavaScript functions are defined. Also, a MouseMovement object oMouseMovement is initiated.

- Init()

  It is called in HTML tag <body onload=init()> , loads the picture and then calls getData().

```
function init(){
        parastr=window.location.href;
        pos = parastr.indexOf("?");
        if (pos<0){
                alert("No param");
        }
        parastr = parastr.substring(pos+1);
        para = parastr.split("&");
        for(i=0;i<para.length;i++) {
                var tempstr = para[i];
                pos = tempstr.indexOf("=");
                if (tempstr.substring(0,pos)=="role"){
                        role=tempstr.substring(pos+1);
                }else if(tempstr.substring(0,pos)=="pic"){
                        pic=tempstr.substring(pos+1);
                        document.all.ImageLayer.innerHTML="<img  src='../pics/"+pic+"'
border=0>";
                }
        }
        getData();
}
```

- getData():

  It invokes JavaScript object oMouseMovement method getData(role) to get other client's mouse position, followed by setPosition() and setGraph() to set the mouse position and draws the graph in the browser. And then it calls itself getData() again in every 0.01 seconds interval to keep getting the other side's information.

```
function getData(){
        oMouseMovement.getData(role);
        setPosition();
```

```
        setGraph();
        eval("setTimeout('getData();', "+INTERVAL+")");
    }
```

- setPosition()

  It turns on the MouseLayer to show the mouse image, sets the mouse image to the coordinates (x, y) which is retrieved from the server and stored in the client object oMouseMovement.

```
function setPosition(){
document.all.MouseLayer.style.top=oMouseMovement.y;
document.all.MouseLayer.style.left=oMouseMovement.x;
document.all.MouseLayer.style.visibility = 'visible';
 }
```

- setGraph()

  It turns on the OutSquareLayer if the drawing type is “rec” and OutCircleLayer if type is “oval”, and draws the graph based on values of top, left, width and height which is retrieved from the server and stored in the client object oMouseMovement.

```
function setGraph(){
        …

        if(oMouseMovement.type=="rect"){
                document.all.OutSquareLayer.style.top=oMouseMovement.top;
                document.all.OutSquareLayer.style.left=oMouseMovement.left;
                document.all.OutSquareLayer.style.width=oMouseMovement.width;
                document.all.OutSquareLayer.style.height=oMouseMovement.height;
                document.all.OutSquareLayer.style.visibility = 'visible';
                document.all.OutCircleLayer.style.visibility = 'hidden';
        }else if(oMouseMovement.type=="oval"){
                document.all.OutCircleLayer.style.top=oMouseMovement.top;
                document.all.OutCircleLayer.style.left=oMouseMovement.left;
                document.all.OutCircleLayer.style.width=oMouseMovement.width;
                document.all.OutCircleLayer.style.height=oMouseMovement.height;
                document.all.OutSquareLayer.style.visibility = 'hidden';
                document.all.OutCircleLayer.style.visibility = 'visible';
        }
 }
```

- sendPosition()

It is triggered whenever the mouse moves. It is embedded in the HTML tag * *

It retrieves the current mouse position coordinates (x, y), assigns it to oMouseMovement JavaScript object and invokes the object method sendData(role) to send to the server. Parameter "role" has to be passed to the server to notify the server where the information comes from.

```
function sendPosition() {
        …

        oMouseMovement.x=nowx+document.body.scrollLeft;
        oMouseMovement.y=nowy+document.body.scrollTop;
        oMouseMovement.sendData(role);
}
```

- sendGraph()

  It is similar to sendPosition(). It retrieves the graph information (top, left, width, height) and assigns them in to the object oMouseMovement, and invoke the method sendGraph(role) to send to the server

```
function sendGraph(){
        oMouseMovement.top=oldY+document.body.scrollTop;
        oMouseMovement.left=oldX+document.body.scrollLeft;
        oMouseMovement.width=nowx-oldX;
        oMouseMovement.height=nowy-oldY;
        oMouseMovement.type=document.main.figure.value;
        oMouseMovement.sendGraph(role);
 }
```

- click()

  It is called when the users click the mouse (not move the mouse). It is embedded in the HTML tag * *

In order to define the top left and the bottom right positions to draw the rectangle or oval, we need to click mouse twice in different spots. First click, records one position. Second click, together with the first position, draws the rectangle or oval in your own browser. Then calls sendGraph() function to send the graph information out to the server.

```
function clicked(){
        …

        clickcount++;
        if(clickcount%2==1){
                recordPosition();
        }else{
                drawGraph();
                sendGraph();
        }
}
```

- recordPosition()

  Record the mouse position on the first click when drawing the graph.

```
function recordPosition(){
        getNowPosition();
        oldX=nowx;
        oldY=nowy;
        document.all.CircleLayer.style.visibility = 'hidden';
        document.all.SquareLayer.style.visibility = 'hidden';
}
```

5.4.3 Java Classes

There are two Java classes defined in this program:

A) Point

Store mouse position coordinates (x, y) in the server.

```
public class Point {
	String x, y;
	public Point(String x, String y) {
		this.x = x;
		this.y = y;
	}
}
```

B) Graph

Store graph information (top, left, width, height, type) in the server.

```
public class Graph {
	String top, left, width, height, type;
	boolean isNew;
	public Graph(String top, String left, String width, String height,
			String type,boolean isNew) {
		this.top = top;
		this.left = left;
		this.height = height;
		this.width = width;
		this.type = type;
		this.isNew=isNew;
	}
}
```

5.4.4 Servlets

There are two Java Servlets in the program:

- FileLister.java

It lists all the available pictures stored in the browser for users to choose one in interest.

```
import java.io.File;
import java.io.IOException;
import java.io.FileInputStream;
import java.io.PrintWriter;
import java.util.Properties;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
public class FileLister extends javax.servlet.http.HttpServlet {
	private static int i = 1;
	public void service(HttpServletRequest request, HttpServletResponse response)
			throws IOException, ServletException {
```

```
…

File dir = new File(rootDir + "pics");

String[] fileList = dir.list();
response.setContentType("text/html;charset=UTF-8");
PrintWriter out = response.getWriter();
String role = request.getParameter("role");
out.println("<HTML>");
out.println("<HEAD><TITLE>Picture List</TITLE></HEAD>");
out.println("<style>body{font-family:Arial,Verdana,Helvetica,sans-serif;}</style>");
out.println("<BODY>");
if ("1".equals(role) || "2".equals(role)) {
        out.println("<table width='640' border='0' cellspacing='0' cellpadding='0'
        align='center'>");
        if (fileList == null) {
                out.println("<tr height='150'><td>"
                        + new File(".").getAbsolutePath() + "</td></tr>");
        } else {
                for (int i = 0; i < fileList.length; i++) {
                        if (i % 4 == 0)
                                out.println("<tr height='150'>");
                                out.println("<td><a
                                href='"+rootURL+"pages/main.html?role=" + role
                                        + "&pic=" + fileList[i] + "'><img
                                src='"+rootURL+"pics/"
                                        + fileList[i]
                                        + "' width=80 height=80 border=0><br>"
                                        + fileList[i] + "</a></td>");
                        if (i % 4 == 4 - 1)
                                out.println("</tr>");
                }
        }
        out.println("</table>");
} else {
        out.println("The role is invalid.");
}
out.println("</BODY></HTML>");
    }
}
```

- main.java

  This is the Servlet which does the main work, retrieving and storing the mouse and graph information to the server objects, at the same time returning back to the client the other side's mouse and graph information. Four Java objects are initiated at the beginning, two Points and two Graphs. Client 1's information is stored in Point [0] and Graph [0], Client

2's is stored in Point [1] and Graph [1]. Client 1 gets information from Point [1] and Graph [1] and Client 2 get information from Point [0] and Graph [0].

To make the program simple and efficient, every time when the Servlet is called, no matter if the client is trying to send the data or get the data, it returns the other client's mouse and graph information back to the browser. The data is sent back in XML format.

```
import java.io.PrintWriter;
import java.io.IOException;
import java.util.Vector;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
import java.util.GregorianCalendar;
import java.util.Calendar;

public class Main extends javax.servlet.http.HttpServlet {
        static Point[] points = {new Point("-100", "-100"),
                        new Point("-100", "-100")};
        static Graph[] graphs = {new Graph("-100", "-100", "0", "0", "rect", true),
                        new Graph("-100", "-100", "0", "0", "rect", true)};

        public void service(HttpServletRequest request, HttpServletResponse response)
                        throws IOException, ServletException {
                int index1 = Integer.parseInt(request.getParameter("role")) - 1;
                int index2 = 1 - index1;
                String sAction = request.getParameter("sAction");
                if ("put".equals(sAction)) {//if the request is to put the current position of the mouse
                        points[index1].x = request.getParameter("x");
                        points[index1].y = request.getParameter("y");
                } else if ("putGraph".equals(sAction)) {//if the request is to put the current graph
                        graphs[index1].top = request.getParameter("top");
                        graphs[index1].left = request.getParameter("left");
                        graphs[index1].width = request.getParameter("width");
                        graphs[index1].height = request.getParameter("height");
                        graphs[index1].type = request.getParameter("type");
                        graphs[index1].isNew = true;
                }

                //return the other role's mouse position and graph info
                //stored in xml format
                String retXML = "<Position>";
                retXML += "<X>" + points[index2].x + "</X><Y>" + points[index2].y
                                + "</Y>";
                if (graphs[index2].isNew) {
                        retXML += "<L>" + graphs[index2].left + "</L><T>"
                                        + graphs[index2].top + "</T><W>" + graphs[index2].width
```

```
                                        + "</W><H>" + graphs[index2].height + "</H><P>"
                                        + graphs[index2].type + "</P>";
                }
                retXML += "</Position>";
                try {
                        response.setContentType("text/xml;charset=UTF-8");
                        response.getWriter().println(retXML);
                } catch (Exception ex) {
                        System.out.println(ex);
                }
        }
}
```

Chapter 6 Advanced Research– Addressing Internet Traffic Jam

As described in the Basic Design in the previous chapter, the mouse position data from Client 1 are constantly sent out to the server and stored in a server Java object. When the server receives the GET request from Client 2, it releases the stored mouse data and passes them to Client 2. In this design, each of two Point objects only holds one mouse position data, and one GET request fetches only one mouse data at a time. If the GET request arrives at the server too late, the mouse position data stored in the Point object will be overwritten by the next mouse data and will never be fetched, resulting in one mouse point loss. Under normal Internet speed and traffic condition, the timing of the GET and SEND matches very well, so the mouse position data are transferred from one remote computer to the other smoothly and continuously, without much mouse data loss. The system is proved successful when both clients are within the United States,

However this situation does not always exist. Internet traffic has been developing at a very fast rate. Coupled with the Internet is the Intranet, which has resulted from the globalization of business. There is a growing requirement for services such as video-conferencing, real-time data transmission, and multi-media applications such as the transmission of colored graphic images, video images, and high-fidelity sounds. E-mail, remote computing, file transfer, and conferencing and voice over Internet are becoming more and more popular. These services, along with others to come, demand larger and larger amounts of traffic bandwidth.

As an example of growth, over the next 15 years, internet traffic between Asia (excluding Australia) and North America is predicted to increase anywhere from 10 to 100 percent. For the

trans-continental communication, where the Internet connections go through undersea cables, the Internet traffic jam is inevitable. The typical problem encountered here is the communication between China and US. Even both party use broadband Internet connections, the speed is still slow. This is due to the bottleneck traffic condition caused by underwater optical cable that connects two continents (Asian and North America). One popular phenomenon is, if you try to access any major Chinese websites, no matter how fast your computer's Internet connection speed is, the web pages you're trying to view is displaying very slowly.

In our case, if Client 1 is in the States and Client 2 is in China, Client 2 would have difficulty in seeing Client 1's mouse movement. This is because the server is located in US and the connection speed from Client 2 (China) to the server is much slower than that from Client 1 to the server. The GET request from Client 2 (China) cannot arrive at the US server in time to fetch the mouse position data from Client 1(US), causing those mouse position data being constantly overwritten and vanishing, hence they will never be transferred and displayed on Client 2's screen.

To solve this problem, I designed and implemented the following two schemas: Data Buffer/Delay and Data Interpolation.

6.1 Data Buffer/Delay

6.1.1 Introduction

The basic idea of the Data Buffer/Delay schema is to reduce the frequency of SEND and GET to cope with the Internet traffic jam condition. In the Basic Design, the frequency has been maximized as highest as the machine can allow. Normally the refreshing frequency can go as

high as 100 times per second. This makes mouse re-display smoothly and instantly when Internet connection is swift. But if the GET request could not get to the server in time due to the Internet traffic jam while the other side keeps sending the mouse data, it could cause significant loss of mouse points. So in the Data Buffer/Delay schema, the SEND and GET frequency is reduced to 2 times per second, *i.e.* the refreshing interval is increased to 0.5 seconds. With this 0.5-second interval, the GET request should have ample time to get to the server and retrieve the mouse data. On the other hand, the mouse data are no longer sent out one point at a time. Instead, they are sent out as a series of mouse points that had been constantly collected during the 0.5-second period. This special design helps to preserve the mouse trail from Client 1 and ensure the continuous mouse trail re-display on client 2's screen.

The result of this algorithm is satisfactory. It greatly improves the view performance in the remote communication between China and US.

6.1.2 System Functionalities and Work Flow

The Buffer Design's system functionality and workflow is illustrated in Figure 7.

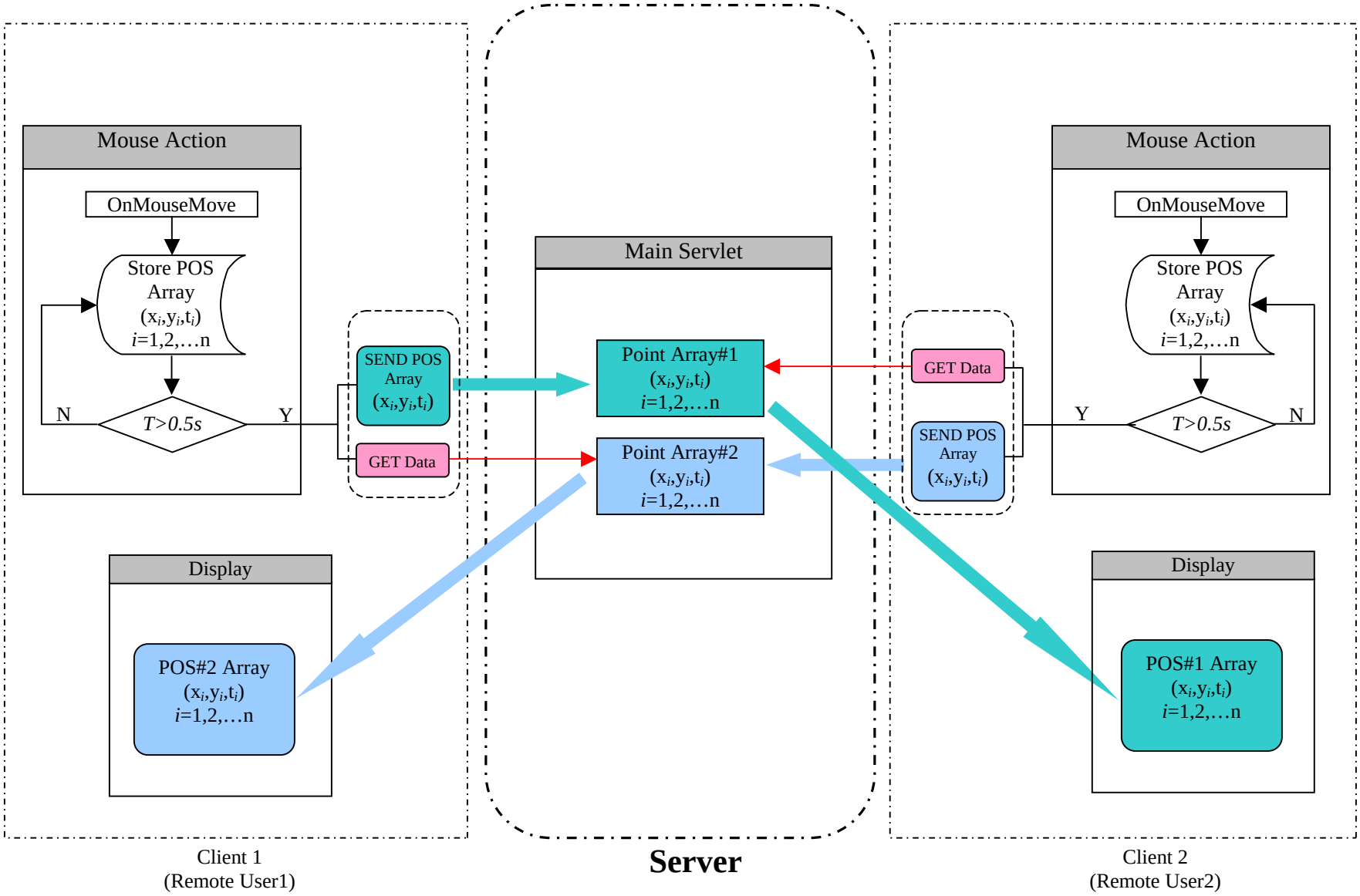

Figure 7 System Functionalities and Work Flow (Buffer Design)

6.1.3 Comparision of GET in Basic and Buffer Designs

Illustration of GET in Basic and Buffer Designs are shown in Figure 8 and Figure 9.

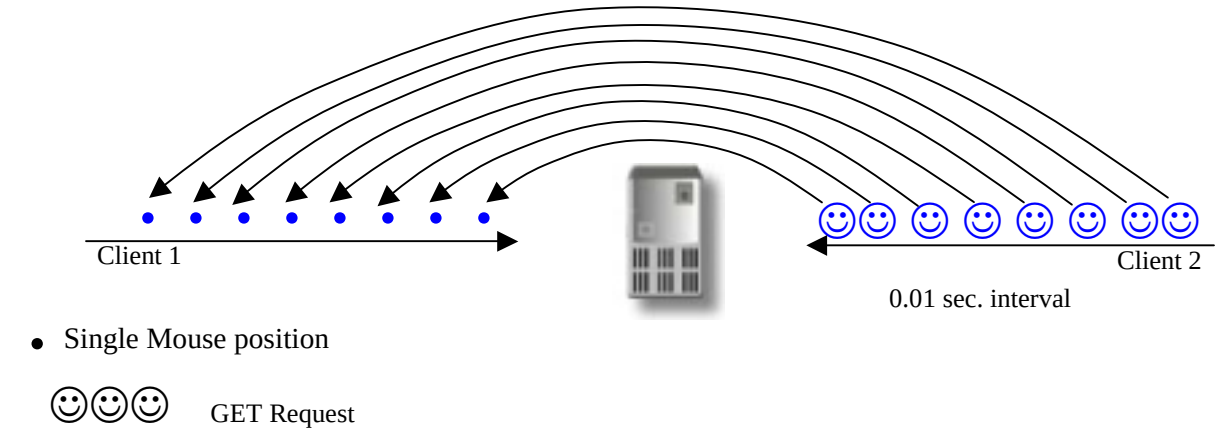

Figure 8 GET Illustration (Basic Design)

In the above Basic Design (Figure 8), when client 1 moves the mouse, the mouse position data is sent out to the server one by one and continuously; while client2, once the web page is loaded, sends GET command to the server at a preset interval, say every 10 milliseconds. When the server receives one GET request, it releases mouse data to client 2. In other word, one GET command, one mouse point released to client 2. This cycle continues, so client 2 can see client 1's mouse movement instantly and precisely.

In the below Data Buffer Design (Figure 9), the SEND and GET frequency is reduced to 2 times per second, which means the refreshing interval is increased to 0.5 seconds. With this 0.5-second interval, the GET request should have ample time to reach the server and retrieve the mouse data. On the other hand, the mouse data are no longer sent out to the server one point at a time. Instead, they are sent out as a series of mouse points that had been recorded during the 0.5-

second period. This special design helps to preserve the mouse trail from Client 1 and ensure the continuous mouse re-display on client 2's screen.

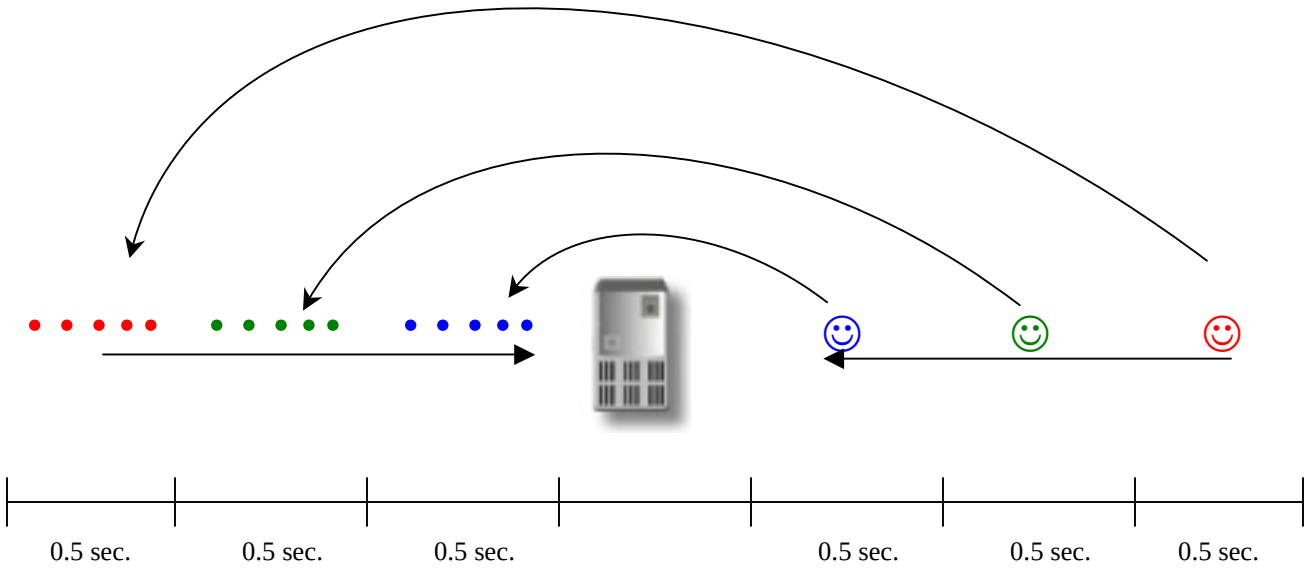

Figure 9 GET Illustration (Buffer/Delay Design)

6.1.4 Function Descriptions

Only the modified objects and functions from the Basic Design are described here.

- JavaScript class MouseMovement()

Timestamp is added as another class property to record the time when the mouse position is stored.

- Java class Point()

Same as JavaScript class MouseMovement(), timestamp is added as another class property to store both mouse position's (x, y) coordinate and the timestamp in server side.

- JavaScript MouseCapture.js

At the beginning, instead of initiating one MouseMovement() object to hold mouse position information, two MouseMovement() objects are initiated like below:

```
var oTrans = new MouseMovement(0,0);
var oDisp = new MouseMovement(0,0);
```

oTrans stores your own desktop's mouse positions which are going to be sent out. Unlike Basic Design, in which property X holds only one mouse position's x coordinate, oTrans holds a series of mouse positions' x coordinate as a string delimited by comma. It is the same with property Y and TimeStamp.

oDisp stores the other client's mouse positions released from the server to be displayed. Like oTrans, a series of mouse positions are stored as strings.

- sendPosition()

This function is called whenever the mouse moves. But unlike Basic Design, in which every mouse position stored in MouseMovement is to be sent out to the server, the new mouse position and timestamp are appended to oTrans' X, Y, Timestamp properties and wait for the next SEND executed in every 0.5s.

```
function sendPosition() {
        getNowPosition();
```

```
        if(Math.abs(nowx-oldX)+Math.abs(nowy-oldY)>2){
          oldX=nowx;
          oldY=nowy;
                oTrans.x+=","+(nowx+document.body.scrollLeft);
                oTrans.y+=","+(nowy+document.body.scrollTop);
                oTrans.timestamp+=","+(new Date().getTime()-start);
        }
}
```

- setPosition()

This function positions the mouse image in the browser according to the mouse position data received from the server, simulating the other client's mouse movement. Since the mouse position data are stored in a string, we need to split them into an array as playXs, playYs and playStamps. Then these mouse position data points will be display on the other client's screen one by one. Since the timestamp is also recorded for each mouse position, when the mouse points are re-displayed, they are not only displayed at the same position as original but also in the same moving rate as original.

```
function setPosition(){
        var playStart=new Date().getTime();
        playXs=oDisp.x.split(",");
        playYs=oDisp.y.split(",");
        playStamps=oDisp.timestamp.split(",");
        if(playXs.length>0)
          play(0);
}

function play(i){
  if(playYs[i]<0 &&playXs[i]<0){
    document.all.MouseLayer.style.visibility = 'hidden';
  }else{
    document.all.MouseLayer.style.visibility = 'visible';
    if(playYs[i]>2 &&playXs[i]>2){
                        document.all.MouseLayer.style.top=playYs[i]-2;
                        document.all.MouseLayer.style.left=playXs[i]-2;
    }
  }
 if(i<playXs.length-1)
  setTimeout('play('+(i+1)+');',playStamps[i+1]-playStamps[i]);
```

}

6.2 Data Interpolation

6.2.1 Introduction

Another schema to cope with Internet traffic jam condition is using data interpolation to manipulate mouse positions. In this schema, not every mouse move would trigger the SEND command. Instead, the program only sends out mouse data in a controlled interval, say, send out every other mouse point to the server. At the same time, client 2 sends its GET request in a slow interval too, say, 5 times per second. When client 2 receives the mouse data from client 1, the program uses a linear interpolation to add back more points between any two adjacent mouse points, then plots all these mouse points, original and interpolated ones, on client 2's computer screen. Therefore, even client 1 didn't send out every single mouse point, client 2 can still maintain a continuity of the mouse movement.

Below is a brief explanation about interpolation. The most practical and simplest form is linear interpolation, which is used in our design. To interpolate any points (x_i, y_i) based on two known points (x_0, y_0) and (x_1, y_1), use the following formula:

$$Y_i = aX_i + b$$

$$\text{Where } a = \frac{y_1 - y_0}{x_1 - x_0},\ b = y_1 - x_1 \frac{y_1 - y_0}{x_1 - x_0}$$

Figure 10 illustrates the linear interpolation for the mouse trail.

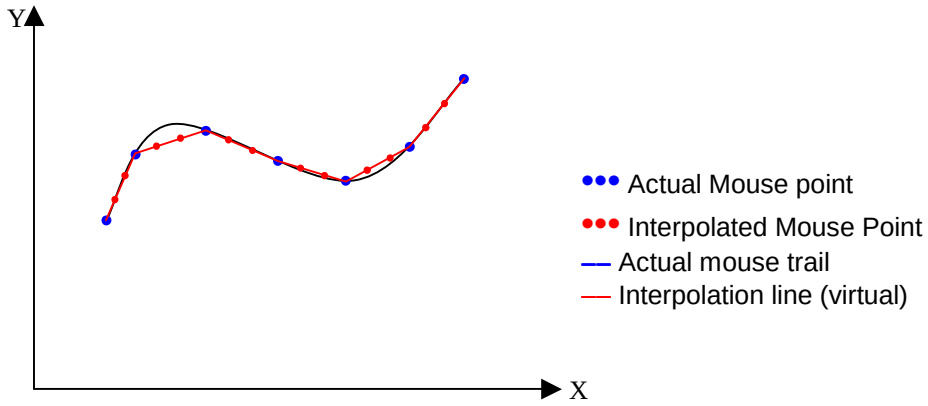

Figure 10 Linear Interpolation Illustration

6.2.2 System Functionalities and Work Flow

The system functionality and workflow for Interpolation Design is illustrated in Figure 11.

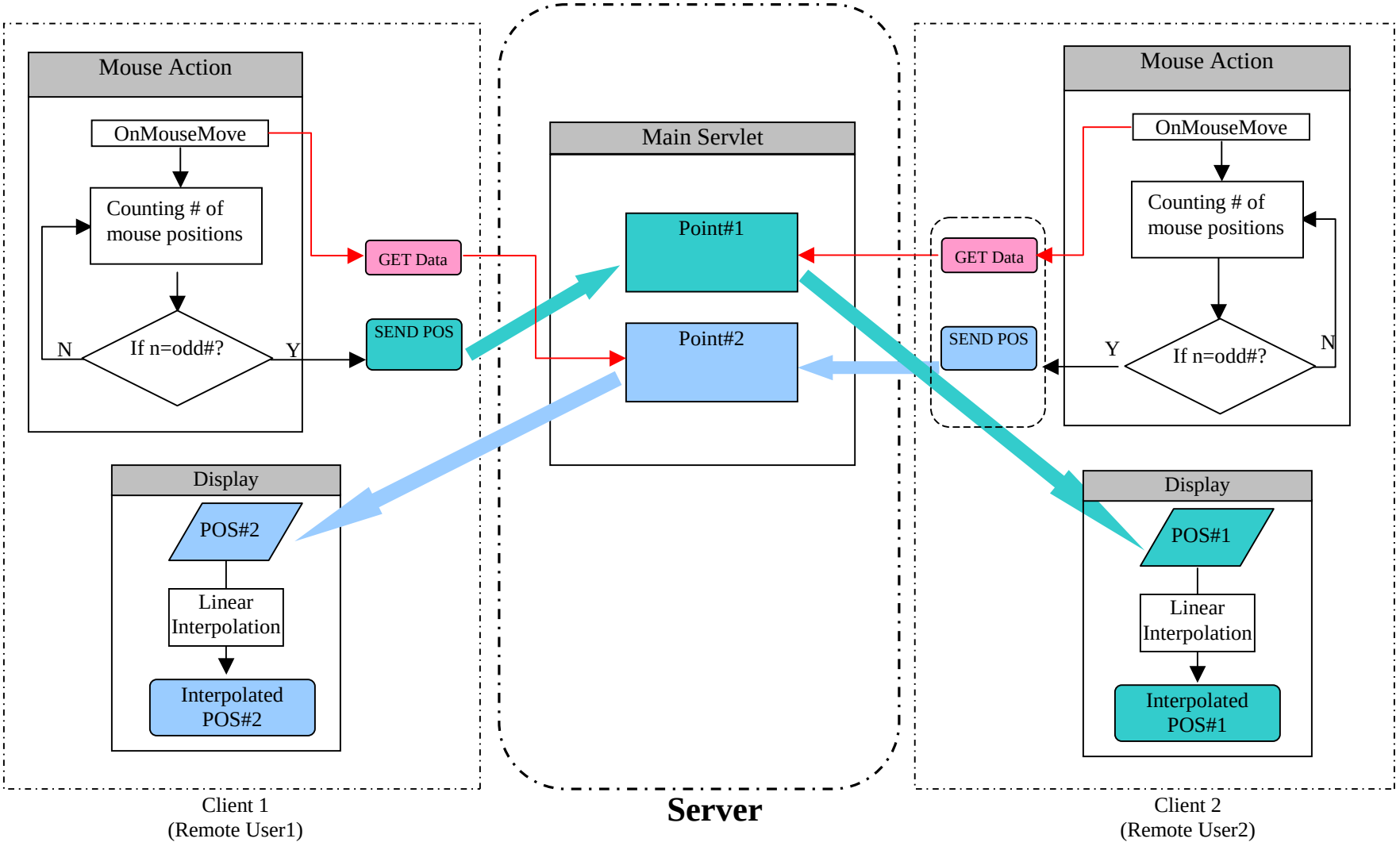

Figure 11 System Functionality & Workflow (Interpolation Design)
-Mouse Position Movement Only

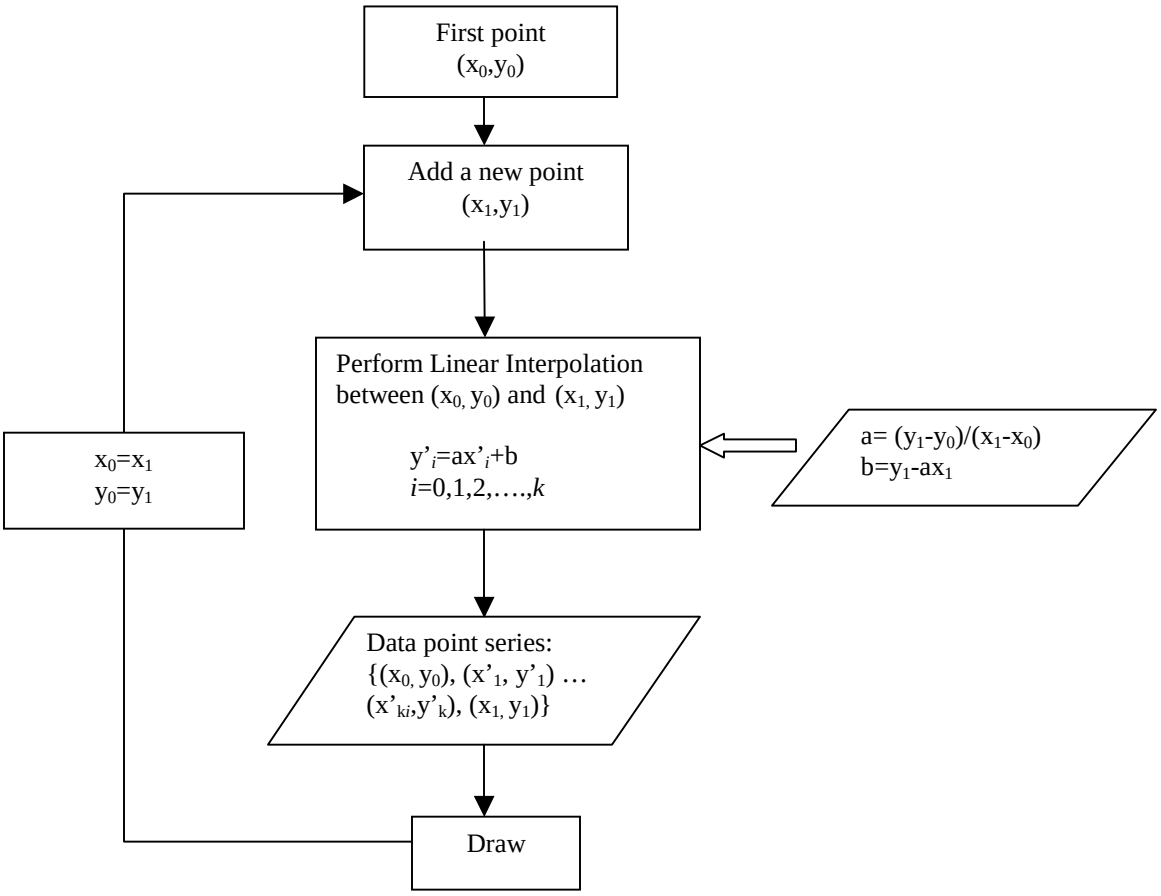

Figure 12 Interpolation Workflow Chart

6.2.3 Function Description

A new JavaScript class MouseDisplay is defined in the JavaScript file MouseDisplay.js since we display not only the original mouse points retrieved from the server, but also the interpolated ones. Similar to Class MouseMovement in the Basic Design, which holds the mouse points received from the server, the new class MouseDisplay holds both the original points and interpolated ones for display.

The class is defined as:

```
function MouseDisplay(){
    this.x=new Array();
    this.y=new Array();
    this.dispX=new Array();
    this.dispY=new Array();
      this.index=0;
      this.MaxLen=2;
      this.calc=MouseDisplay_calc;
      this.add=MouseDisplay_add;
      this.reverse=MouseDisplay_reverse;
}
function MouseDisplay_add(posX,posY){
      this.dispX=new Array();
      this.dispY=new Array();
      var len=this.x.length;
     if(posX<0||posY<0){
            document.all.MouseLayer.style.visibility = 'hidden';
            return;
       }
      var i=0;
      if (len<this.MaxLen){
            this.x[len]=posX;
            this.y[len]=posY;
     }else{
           for (i=0;i<len-1;i++){
              this.x[i]=this.x[i+1];
              this.y[i]=this.y[i+1];
           }
          this.x[len-1]=posX;
          this.y[len-1]=posY;
     }
}
function MouseDisplay_calc(){
      var len=this.x.length;
```

```
 this.dispX=new Array();
this.dispY=new Array();
if(len<2)
      return;
if(this.x[len-1]==this.x[len-2]&&this.y[len-1]==this.y[len-2]){
      return;
 }
   var i=0;
   if(len==2&&Math.abs(this.x[1]-this.x[0])>0){
      if(this.x[1]-this.x[0]>0){
          for(i=0;i<this.x[1]-this.x[0];i++){
              this.dispX[i]=Math.abs(this.x[0])+i+1;
              this.dispY[i]=Math.abs(this.y[0])+Math.round((i+1)*(this.y[1]-this.y[0])/(this.x[1]-this.x[0]));
          }
       }else{
          for(i=0;i<this.x[0]-this.x[1];i++){
              this.dispX[i]=Math.abs(this.x[0])-i-1;
              this.dispY[i]=Math.abs(this.y[0])-Math.round((i+1)*(this.y[1]-this.y[0])/(this.x[1]-this.x[0]));
          }
       }
   }else if(len==2&&Math.abs(this.y[1]-this.y[0])>0){
          if(this.y[1]-this.y[0]>0){
              for(i=0;i<this.y[1]-this.y[0];i++){
                  this.dispY[i]=Math.abs(this.y[0])+i+1;
                  this.dispX[i]=Math.abs(this.x[0])+Math.round((i+1)*(this.x[1]-this.x[0])/(this.y[1]-
                   this.y[0]));
              }
          }else{
              for(i=0;i<this.y[1]-this.y[0];i++){
                  this.dispY[i]=Math.abs(this.y[0])-i-1;
                  this.dispX[i]=Math.abs(this.x[0])-Math.round((i+1)*(this.x[1]-this.x[0])/(this.y[1]-
                   this.y[0]));
              }
          }
   }
}
```

- setPosition()

Notice that in the Basic Design, when a client computer receives a mouse point (x, y), it immediately draws it. In the interpolation design, when the client computer receives mouse point, it adds it to the oMouseDisplay object, until there are two mouse points, then a linear interpolation is performed and more points are generated. The original points and the interpolated points are all stored in oMouseDisplay object and then plotted one by one on the screen.

```
function setPosition(){
        if(oMouseMovement.x<0||oMouseMovement.y<0){
                return;
        }
        oMouseDisplay.add(oMouseMovement.x,oMouseMovement.y);
        oMouseDisplay.calc();
        if(oMouseDisplay.dispX.length>0){
                displaySingle(0);
        }
}

function displaySingle(i){
        if(oMouseDisplay.dispY[i]<0 &&oMouseDisplay.dispX[i]<0){
                document.all.MouseLayer.style.visibility = 'hidden';
        }else{
                document.all.MouseLayer.style.visibility = 'visible';
                if(oMouseDisplay.dispY[i]>2 &&oMouseDisplay.dispX[i]>2){
                        document.all.MouseLayer.style.top=oMouseDisplay.dispY[i]-2;
                        document.all.MouseLayer.style.left=oMouseDisplay.dispX[i]-2;
                }
        }
        if(i<oMouseDisplay.dispX.length-1){
```

```
            setTimeout('displaySingle('+(i+1)+')',Math.round((INTERVAL)/oMouseDisplay.dispX.length));
        }else{
            oMouseDisplay.dispX=new Array();
            oMouseDisplay.dispY=new Array();
        }
}
```

6.3 SYSTEM PERFORMANCE

Both Data Buffer and Data Interpolation designs further more improve the performance of the workspace sharing system. After applying 6.1 or 6.2, the system is now producing a satisfactory viewing effect for the remote communication between U.S. and China.

Chapter 7 Optimize Refreshing Frequency

7.1 Background

In the Data Buffer design, the key is the looping interval, which is the time for the system to complete one cycle of GET and SEND of the mouse points. If the interval is set too short, data might get lost. But if the interval is set too long, the user would experience delay or latent mouse movement.

The loop interval in the Basic Design is set at 0.01s, which is probably the smallest a computer allows in order to achieve the best real-time performance. In reality, the Internet traffic jam condition would slow down the SEND-GET cycle, therefore in the Buffer Design, a larger loop interval, say 0.5s, was set so that it allows the whole actual SEND-GET cycle to complete before the next cycle starts. During the enlarged interval, the system is still recording the mouse points into a series, which will be sent to the server at the end of each interval. This non-stop recording preserves the mouse position data from client 1.

Figure 13 shows a complete loop of GET and SEND cycle.

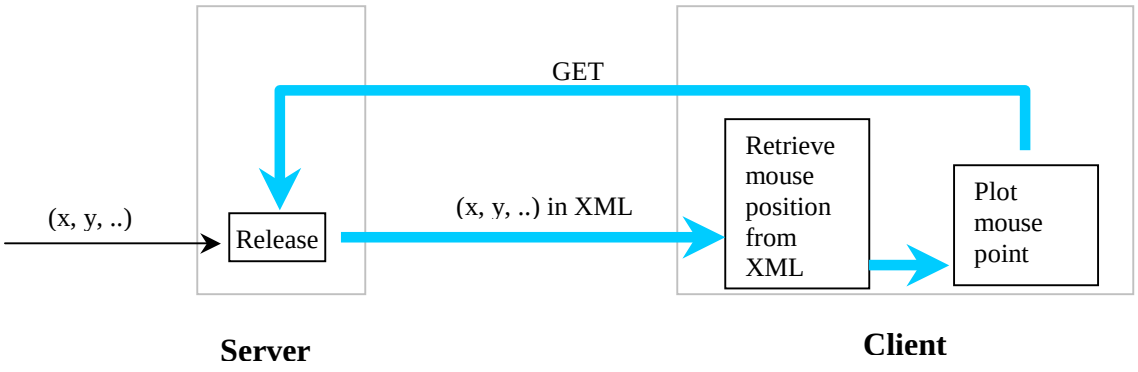

Figure 13 Cycle of GET and SEND of mouse position data

Therefore, there exists an optimal looping interval that gives the best remote viewing performance. The optimal interval is related to the Internet speed between client and server.

Internet speed refers to the average speed of data transmission between your computer (client) and the server you visit. In this thesis Internet speed is measured by the round trip time using *ping* command. This speed is related to number of nodes between client and server as well as the connection speeds between each node. Using *tracert* command, it is easy to find out how many nodes between client and server as well as the Internet speed. Figure 14 shows a *tracert* result:

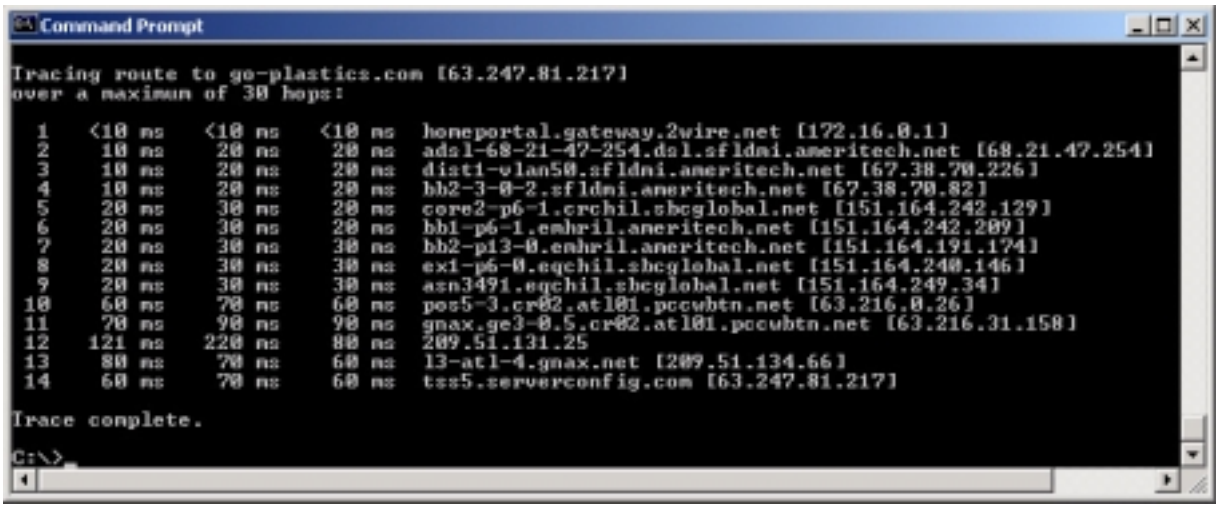

Figure 14 Example of transmission time and nodes that a packet travels to the web server

In Figure 14, the first column is the node number. Each of the next three columns shows the round-trip times in milliseconds for an attempt to reach the destination. The fourth column is the host name (if it was resolved) and IP address of the responding system.

In order to achieve the best viewing performance under different Internet speed, the looping interval needs to be optimized. An optimization test was conducted, which includes:

- Selecting test locations
- Detecting Internet speed at each test site
- Experimenting out a minimum loop interval that would result in a best viewing performance under different Internet speed.
- Plotting data points to find out the relationship between Internet speed and the optimal looping interval.

7.2 Optimization Test

7.2.1 Selecting test locations

The web server hosting our application is located in Atlanta, Georgia, USA. Four client locations are picked from Ann Arbor, MI, New York, NY, San Diego, CA, and Ningbo, China, as listed in Table 3. The Internet connection type and distance to the web server are also listed. Geographic locations are indicated in Figure 15.

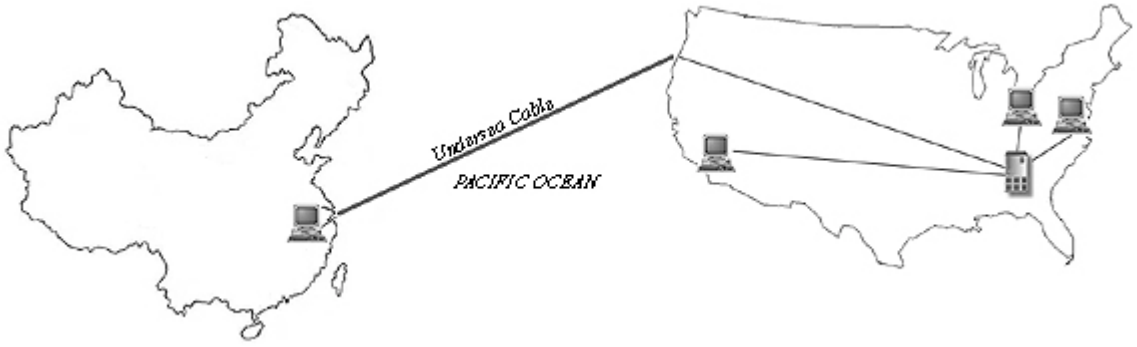

Figure 15 Geographic locations of client computers and server

7.7.2 Detecting Internet speed at each test site

In this study, Internet speed is measured in terms of round trip travel time of a packet. When using *ping* command, the computer measures a round trip travel time for a packet of 32 bytes, traveling to and from the web server. The screen shot in Figure 16 shows a test result of *ping*ing the web server from location 1, Ann Arbor, Michigan, with DSL connection:

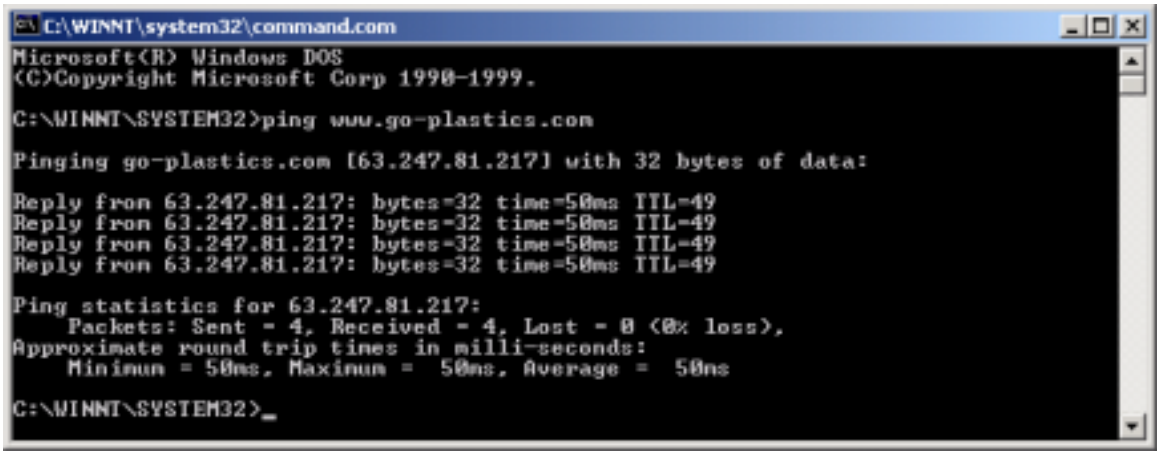

Figure 16 Example of Pinging the web server

By using this method, I recorded the transmission time of the packet for each location, as shown in Table 3, listed as packet travel time.

During the test I also observed that the Internet speed is not constant. Instead, it fluctuates about 3~8% from time to time, as shown in Table 2. This is reasonable since Internet traffic varies from time to time.

Table 1 Internet speed varies with time

	From Location#6 (ms)	From Location#1 (ms)
Trial #1	1445	60
	Timeout (1500+)	60
	1426	52
	Timeout	60
Trial #2	Timeout	60
	1428	50
	1464	62
	Timeout	60
Trial #3	1404	60
	Timeout	60
	1448	50
	Timeout	60

Note:

- Timeout indicates that the round trip time exceeds the pre-defined number, which is 1500 in this case.
- Each trial is about 5~10 seconds apart.

7.2.3 Optimizing looping interval

This test is to find out the relationship between Internet transmission time and the minimum looping interval.

First, we need to define the following rating to quantify the viewing performance result:

Table 2 Criterion for optimal viewing performance

Description of Mouse Movement	**Rating**
Very smooth	4
Smooth	3
Intermittent (jerky, latent, etc.)	2
Mouse is visible but almost no trace of movement	1
No mouse can be seen at all	0

Next, we communicate with each remote user from the locations we chose. By setting different loop intervals in the program, we can have different viewing results. The minimum interval that can achieve a rating of 3 and above was recorded in Optimized Interval column in Table 3.

Table 3 Results of Optimization Test

#	Location of Client 2	Connection	Approx. Distance to Server (miles)	Packet travel time (ms)	Optimized Interval (s)
1	Ann Arbor, MI	DSL	735	50	0.01
2	Ann Arbor, MI	56K Modem	735	240	0.1
3	New York, NY	Cable	870	30	0.01
4	San Diego, CA	DSL	2166	50	0.01
5	San Diego, CA	T1	2166	40	0.01
6	Ningbo, China,	DSL	8500	1500	0.5

7.2.4 Plotting data

By using the last two columns we plot a graph in Figure 17.

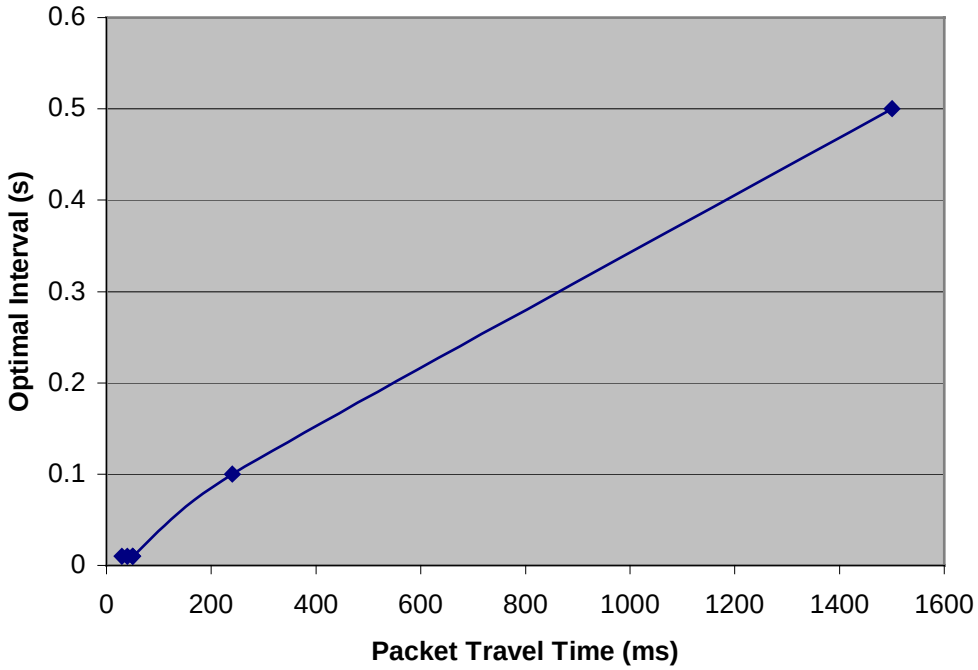

Figure 17 Relationship between Optimized Interval and Internet Speed

From Figure 17 we can see the relationship between optimized looping interval and the packet travel time. The shorter the packet travel time, which means the higher the Internet speed and the smaller the looping interval, the better the real time performance is. Although the data are too scattering to give out an accurate regression line, it can still provide the guidance in choosing the best looping interval under different Internet connection. On the other hand, the relationship between Internet speed and distance is not strong. All across the states, from east coast to west coast, the Internet speed is not much different. Although in the Far East, such as China, the Internet speed connecting to our server is slow, it is mainly due to the under-capacity of undersea cable, not just because it is far away.

7.3 Conclusion and Future Study

The optimal looping interval is a linear function of the Internet speed at which a remote client communicates with the server. The higher the Internet speed, the smaller the looping interval is. It is not strongly related to the distance.

By finding the Internet connection speed of a remote client user using *ping* command, we can obtain an optimal looping interval from the function diagram in Figure 17.

In our current implementation we manually set the looping interval, roughly based on the estimate of the Internet connection speed.

For future study, a dynamic algorithm should be used to automatically adjust the looping interval based on the real-time Internet speed detected by the program.

Chapter 8 Conclusion

8.1 Conclusion

This thesis used innovative thinking and implementation to address real business need. By implementing the schemas discussed in the previous chapters, I have fulfilled all the requirements set by certain user groups. The following are the unique features of this workspace sharing system:

1. Extreme simplicity

   The viewing is exactly the same as viewing any Internet page: no setup, no software downloads. This is by far one of the simplest form for any desktop sharing.

2. Very low cost

   The application is hosted by an ISP with Java/Tomcat support. Except for a small monthly hosting fee, there is no other cost.

3. Fast

   Since the only data needed to refresh is the mouse position, not the whole desktop image, the transmitted data is very small, just a few bytes. Plus, the image being viewed is a static picture and can be pre-loaded into each remote user's own computer by the Internet browser. There is no interrupted or slow display of the mouse movement or image during viewing.

4. True color

All the images being viewed are the original picture. It can be true color or gray scale. The screen will 100% re-display the image as original color, with the same hue, saturation and lightness.

5. Two way viewing

   With our unique design, remote two parties can see each other's mouse movement at the same time. This feature is especially convenient for two remote users to have a technical conversation.

In summary, the system illustrated in this thesis is

1. Innovative: Compared to the traditional method of remote desktop sharing, which focus on transmitting the whole desktop image, the system proposed in this thesis focuses on transmitting only mouse position data between clients. This significantly increases the speed of data stream and improves the viewing performance. Also, a couple of additional innovative algorithms are developed to optimize the system, further enhancing the performance.
2. Challenging: Besides a set of stringent requirement of the original problems to solve, there is also a challenge imposed by the trans-continental Internet connection, where bottleneck Internet traffic jam is a big issue. The system architectures and algorithms in this thesis are all none trivial but well solved all those challenges.
3. Practical: with simplicity and robustness, this system has been used very well and will become a popular tool for a specific user group that needs a remote interactive technical discussion

8.2 Future study

The thesis pioneered a new method of web browser based workspace sharing. More features and capabilities can be explored in a later stage by people who are interested. A few topics are listed below:

7.2.1 Optimization of Looping Interval

As described at the end of Chapter 7, a dynamic algorithm should be developed to automatically adjust the looping interval based on the real-time Internet speed detected by the program.

7.2.2 Picture format

Currently the system only works with pictures in JPG (JPEG or JPE) and GIF format. It does not work with pictures in TIF (TIFF), PDF and other image formats. This is mainly because the program is working inside the web browser. If an image is not displayed inside the browser, the program cannot relate the mouse position over the image's coordinate system. Most current web browser does not have plug-in viewer for TIF and PDF file. When you click a link for image in TIF and PDF format, the browser will trigger an external program, like Microsoft/Kodak's Image Viewer or Adobe, the display of pictures in PDF, TIF is run by their own program outside the browser. However in higher version browser it starts to have PDF or tiff plug-in, so the pictures of these formats can be display within the browser.

Figure 18 and Figure 19 show the TIF viewing software and PDF viewing software, respect.

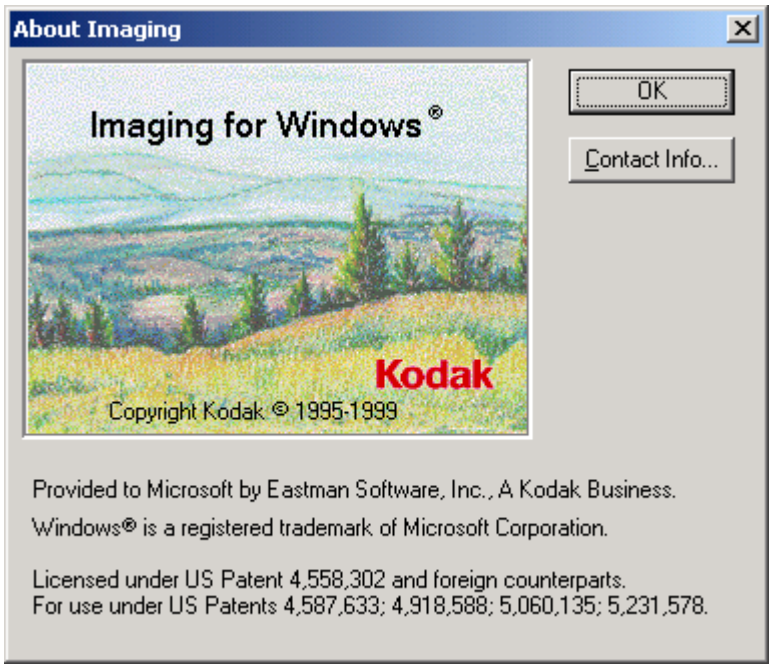

Figure 18 TIF Image Viewing Software

Figure 19 PDF Image Viewing Software

8.2.3 Multiple users

Another topic for future study is how to share a picture with more than 2 persons.

8.2.4 Other browsers

Due to limited time for the development and popularity of the Microsoft Internet Explore, the thesis did not test the validity of other browsers such as Netscape and AOL, etc.

8.2.5 More features

Add whiteboard function, so that users from both ends can type texts or draw any graph on the screen, like Microsoft whiteboard and instant messenger. This would make the communication more convenient and effective.

References

1. Microsoft NetMeeting www.microsoft.com/NetMeeting
2. WebEx www.webex.com
3. Citrix GotoMeeting http://www.citrix.com/English/PS/products/product.asp?familyID=13991&productID=13976
4. NetMeeting -True Color, http://ercrmsweb.engin.umich.edu/tutorials/Netmeeting.htm
5. True Color, http://en.wikipedia.org/wiki/True_color
6. The pseudo color generation program, http://www.geocities.com/~special_effect/sw_pseudo_colorize.html
7. Tomcat User's Guide, http://jakarta.apache.org/tomcat
8. Using Tomcat, http://www.onjava.com/pub/ct/33
9. Java by Example, by Jerry R. Jackson, Alan L. McClellan, The Sunsoft Press
10. XML In a Nutshell, by Elliotte Rusty Harold & W. Scott Means, O'REILLY
11. VML - the Vector Markup Language, http://www.w3.org/TR/1998/NOTE-VML-19980513
12. Web Workshop - Specs & Standards -- VML Overview, http://msdn.microsoft.com/library/default.asp?url=/workshop/author/vml/default.asp
13. JavaScript, The Definitive Guide, by David Flanagan, O'REILLY
14. Dynamic HTML: The Definitive Reference (2nd Edition), by Danny Goodman, O'REILLY
15. Remote Scripting with IFRAME, http://developer.apple.com/internet/webcontent/iframe.html
16. Jean-Marie Beaufils, Undersea Cable Technology in the Pacific, winter 1999, Underwater Magazine http://www.diveweb.com/telecom/

17. An Oversimplified Overview of Undersea Cable Systems, http://davidw.home.cern.ch/davidw/public/SubCables.html
18. Conclusion of China-U.S. Cable Network, http://www.kddi.com/english/corporate/news_release/archive/kdd/press-e97/97-030-c.html
19. Global Communications Submarine Cable Map 2004, http://www.telegeography.com/products/map_cable/index.php
20. Numerical Methods for Scientists and Engineers, R. W. Hamming, Second Edition
21. Mathematics at Work, Hilbrook L. Horton, Fourth Edition
22. Mathematics - Its Content, methods and Meaning, A. D. Aleksandrov, A. N. Kolmogorov, M. A. Lavrent'ev
23. Essential Ping, http://compnetworking.about.com/od/softwareapplicationstools/l/aa100700a.htm
24. Understanding a Tracert, http://www.wurd.com/misc_tracert.php
25. Networking Basics: Traceroute and Ping Overview, http://www.visualware.com/resources/tutorials/tracert.html