EFFICIENT PROTOCOL FOR LIVE WEB-BASED TEACHING

By

RAVINDER PAL SINGH

A THESIS PRESENTED TO THE GRADUATE SCHOOL
OF THE UNIVERSITY OF FLORIDA IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE

UNIVERSITY OF FLORIDA

2002
Copyright 2002

by

Ravinder Pal Singh
To my parents and sister, for their love, encouragement and support
ACKNOWLEDGMENTS

First of all I would like to express my heartfelt gratitude to my advisor, Dr. Jose Principe, for giving me an opportunity to work under his supervision and providing valuable insight during the course of my research. Without his guidance and support, this thesis would not have been possible.

Next, I would like to express my deep gratitude to my family in India for their constant moral support and encouragement, which still motivate me to achieve my goals.

I would like to sincerely thank Dr. Randy Chow and Dr. Richard E. Newman, members of my committee, for their invaluable contribution to my thesis.

I would also like to thank Dr. Michael P. Frank for his timely help for my defense.

Lastly, I sincerely thank Shayan, Yadu, Geetha and all CNELities, for providing an excellent environment around me.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACKNOWLEDGMENTS</td>
<td>iv</td>
</tr>
<tr>
<td>LIST OF FIGURES</td>
<td>viii</td>
</tr>
<tr>
<td>ABSTRACT</td>
<td>x</td>
</tr>
<tr>
<td>INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>1.1 Introduction to Interactive Teaching and Learning on the Web</td>
<td>1</td>
</tr>
<tr>
<td>1.2 Interactive Teaching Laboratory</td>
<td>2</td>
</tr>
<tr>
<td>1.3 Objective of Interactive Teaching on the Web</td>
<td>4</td>
</tr>
<tr>
<td>1.3.1 Server</td>
<td>4</td>
</tr>
<tr>
<td>1.3.2 Communication Media</td>
<td>4</td>
</tr>
<tr>
<td>1.3.3 Client</td>
<td>5</td>
</tr>
<tr>
<td>1.4 Organization of Thesis</td>
<td>5</td>
</tr>
<tr>
<td>1.5 Summary</td>
<td>6</td>
</tr>
<tr>
<td>MULTICASTING AND REAL-TIME TRANSPORT PROTOCOL</td>
<td>8</td>
</tr>
<tr>
<td>2.1 Broadcasting and Multicasting</td>
<td>8</td>
</tr>
<tr>
<td>2.1.1 Advantages of Multicasting</td>
<td>9</td>
</tr>
<tr>
<td>2.1.2 Internet Protocol (IP) and Multicast Addresses</td>
<td>9</td>
</tr>
<tr>
<td>2.1.3 Group Participation in IP Multicast</td>
<td>10</td>
</tr>
<tr>
<td>2.1.4 Multicast Prerequisites</td>
<td>11</td>
</tr>
<tr>
<td>2.1.5 Time To Live (TTL)</td>
<td>11</td>
</tr>
<tr>
<td>2.1.6 Multicast Filtering Switches</td>
<td>13</td>
</tr>
<tr>
<td>2.2 The Internet Group Management Protocol (IGMP)</td>
<td>13</td>
</tr>
<tr>
<td>2.3 Multicast Routing Concepts</td>
<td>16</td>
</tr>
<tr>
<td>2.3.1 Multicast Routing</td>
<td>16</td>
</tr>
<tr>
<td>2.3.2 Spanning Trees</td>
<td>17</td>
</tr>
<tr>
<td>2.4 IP Multicast Backbone on the Internet (MBone)</td>
<td>17</td>
</tr>
<tr>
<td>2.5 Streaming Media and Real-Time Transport Protocol</td>
<td>18</td>
</tr>
<tr>
<td>2.5.1 Streaming Media</td>
<td>18</td>
</tr>
<tr>
<td>2.5.2 Protocols for Streaming Media</td>
<td>20</td>
</tr>
<tr>
<td>2.6 Real-Time Transport Protocol</td>
<td>21</td>
</tr>
<tr>
<td>2.6.1 RTP Services</td>
<td>21</td>
</tr>
<tr>
<td>2.6.2 RTP Architecture</td>
<td>22</td>
</tr>
<tr>
<td>2.6.3 Data Packets</td>
<td>23</td>
</tr>
</tbody>
</table>
RESULTS

5.1 Bandwidth Requirements

5.2 Ability to Reconstruct Smart Board Annotations

5.3 Ability to Synchronize the Audio and the Video Streams

5.4 Summary

CONCLUSION AND FUTURE WORK

6.1 Conclusion

6.2 Future Work

6.2.1 Features to Synchronize Teacher and Student’s I-Books

6.2.2 Features to Enable Two Way Communication

6.2.3 Web-Enabled Interface

LIST OF REFERENCES

BIOGRAPHICAL SKETCH
## LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 1-1.</td>
<td>Interactive teaching laboratory (ITL)</td>
<td>3</td>
</tr>
<tr>
<td>Figure 2-1.</td>
<td>Address format in IP</td>
<td>10</td>
</tr>
<tr>
<td>Figure 2-2.</td>
<td>Multicasting basics</td>
<td>12</td>
</tr>
<tr>
<td>Figure 2-3.</td>
<td>Multicast filtering switches</td>
<td>13</td>
</tr>
<tr>
<td>Figure 2-4.</td>
<td>Details of multicasting IP addresses</td>
<td>14</td>
</tr>
<tr>
<td>Figure 2-5.</td>
<td>Architecture of RTP</td>
<td>22</td>
</tr>
<tr>
<td>Figure 2-6.</td>
<td>RTP data-packet header format</td>
<td>23</td>
</tr>
<tr>
<td>Figure 3-1.</td>
<td>Smart board</td>
<td>30</td>
</tr>
<tr>
<td>Figure 3-2.</td>
<td>High-level JMF architecture</td>
<td>33</td>
</tr>
<tr>
<td>Figure 3-3.</td>
<td>Working of player</td>
<td>35</td>
</tr>
<tr>
<td>Figure 3-4.</td>
<td>Working of processor</td>
<td>36</td>
</tr>
<tr>
<td>Figure 3-5.</td>
<td>Working details of processor</td>
<td>37</td>
</tr>
<tr>
<td>Figure 4-1.</td>
<td>Abstract system design</td>
<td>43</td>
</tr>
<tr>
<td>Figure 4-2.</td>
<td>Details of the server and its components</td>
<td>47</td>
</tr>
<tr>
<td>Figure 4-3.</td>
<td>Three-tier architecture for getting smart board events</td>
<td>48</td>
</tr>
<tr>
<td>Figure 4-4.</td>
<td>Synchronization details</td>
<td>52</td>
</tr>
<tr>
<td>Figure 4-5.</td>
<td>Details of the client and its components</td>
<td>53</td>
</tr>
<tr>
<td>Figure 4-6.</td>
<td>Master/Slave relationship between players for synchronization purposes</td>
<td>54</td>
</tr>
</tbody>
</table>
Abstract of Thesis Presented to the Graduate School of the University of Florida in Partial Fulfillment of the Requirements for the Degree of Master of Science

EFFICIENT PROTOCOL FOR LIVE WEB-BASED TEACHING

By

Ravinder Pal Singh

May 2002

Chairman: Dr. Jose C. Principe
Major Department: Computer and Information Science and Engineering

Web-based teaching is an efficient tool to reach students in remote locations or asynchronously at home. Because of technological limitations (such as transmission bandwidth, network congestion and amount of data flow between server and client), this concept is still far from reality because students still require high-speed connections such as T1 or Digital Subscribers Line (DSL). Removing technological hindrances can help implement the concept of distributed classes in various parts of world using simple modems.

In this thesis we focus on the design, development and testing of a protocol. The protocol uses customized hardware to efficiently transmit and synchronize annotations on special white board with the voice of the teacher to simulate the actual classroom on the web. The main motivation for using white boards is to reduce the amount of data flowing between the server and the client. This saves bandwidth, which results in a better quality of service and a highly interactive teaching/learning environment.
This protocol provides an effective method of online teaching. Bandwidth requirements for this protocol are much less than bandwidth requirements for present day protocols for audio/video transmission.
CHAPTER 1
INTRODUCTION

1.1 Introduction to Interactive Teaching and Learning on the Web

Conventional classroom teaching is the best example of interactive teaching and learning because it provides an environment of direct interaction between students and the teacher. One of the main factors responsible for the success of conventional classroom is the instant delivery of information from teacher to student and vice versa. This results in immediate solutions to students’ questions and instant feedback from the students.

However the conventional classroom poses many restrictions on the flexibility of the class (for example, the class should meet in one place, the size of the class depends on the space available, and attendance is difficult for students who live far away). Ideally, we need a new way of teaching that is as effective as the conventional classroom but free of its restrictions.

The Internet is currently the dominant medium for information flow and is free of all geographical limitations. Information can flow from one part of the world to another within seconds. Our main idea is to replace the conventional classroom with an Internet-based “web classroom” and still have all the benefits of the conventional classroom.

If we are able to implement the “web classroom” successfully, the results will be amazing as all the students will have liberty to join class despite any geographical or time restrictions. The main challenge of the “web classroom” is to make it interactive, which depends on how effectively data flows from classroom (server) to students (clients).
A successful effort has been made in this thesis to achieve this by using customized hardware and communication protocols. Results obtained from the simulated “web classroom” are very encouraging.

1.2 Interactive Teaching Laboratory

To realize the concept of Interactive teaching on the web, a highly sophisticated computing environment has been created in the Electrical Engineering building at the University of Florida in which all the necessary technical equipment and hardware are provided.

The Interactive Teaching Laboratory (ITL) is part of the Computational NeuroEngineering Laboratory in the Department of Electrical and Computer Engineering at the University of Florida. It is dedicated to advancing the state of the art in scientific education by using computer technology to create an interactive learning environment -- not just a multimedia presentation.

The ITL is sponsored by the National Science Foundation. The equipment was purchased with an ILI-IP grant and the concept recently received a multi-site grant for implementing and testing of an Internet-based team-teaching infrastructure. The same system also can be used for distance learning. One of the goals is to use low bandwidth communication (modems).

The ITL currently has 15 networked Pentium machines and a high-end Pentium II server. The lab has an LCD projector and a smart board, which is an interactive white board that allows the instructor to actively control the computer and annotate/highlight the contents while standing in front of the large projected view of the instructor's computer.
In the classes offered in the ITL, each student has her/his own computer with interactive hypertext material and examples/simulations. The book used for the course is by Dr. Principe and is called, *Neural and Adaptive Systems* (published by John Wiley and Sons) [1].

![Interactive teaching laboratory (ITL)](image)

Lectures use the studio format: each topic is presented and illustrated with a computer example. The simulator is live, allowing the student to run the examples independent of the teacher for 10 to 15 minutes [2]. The teacher summarizes and goes into the next topic, and the cycle repeats. The big advantage of this format is that students must ask and answer their own questions. This normally leads to a much deeper understanding of the material. To further facilitate the learning process, work is being
done on designing and developing a Web based Graphical user interface. This will be used to implement Neural Network applications in an Interactive Web-based book.

1.3 Objective of Interactive Teaching on the Web

The main objective of this project is to simulate classroom teaching as closely as possible on the Web. Looking into the technical details of the project we can divide the whole process into three parts. Then we can aim to meet each individual objective separately so that when we combine these we get a system with satisfactory performance. Here we discuss all the three components of the project and their individual objectives.

1.3.1 Server

The server is the most important component of the system as it is responsible for serving all clients. In addition to having high-end processing capabilities it should meet the following requirements:

- Capable of getting annotations from smart board with minimum possible time lag.
- Capable of collecting an audio (teacher’s voice) stream and encoding it efficiently to minimize bandwidth requirements.
- Capable of sending an audio stream, which is independent of hardware platforms.
- Ideally, capable of sending synchronization signals to clients so that they can synchronize with an i-book running on the server.
- Capable of serving all the clients efficiently.

1.3.2 Communication Media

Communication media plays an important part in this process. As we know, for our project the communication media is the Internet and there is nothing we can do to boost its speed. But if one uses the existing infrastructure with intelligent protocols (RTSP/RTP for audio transmission), which can help reduce network traffic with
minimum time lag, we can achieve our objective. While discussing communication media we concentrate on the following points:

- Our system should be able to serve all of the clients with the same effectiveness despite differences in communication mediums. For example some clients may connect through modems (56K). Some may connect through DSL or LAN.

- An optimal technique of transmission should be used. This includes a thorough evaluation of different techniques such as broadcasting, multicasting or different threads serving respective clients.

- Time to live (TTL) parameter should be set carefully to avoid unnecessary traffic on the network.

1.3.3 Client

The client side or student side is the destination of the of the data stream. The main objectives, which must be met on this side of the system are as follows:

- One of the most important objectives on the client side is to achieve synchronization between the professor’s voice and annotations on the smart board. This is because we expect audio-visual actions to happen simultaneously, and the human eye is not tolerant of contents presented out of sync [3].

- Clients should be able to get smooth streams of annotations on the smart board and voice of the professor, which is the sole purpose of this project.

- Clients should receive exactly what is happening in the classroom (voice and annotations). This can be achieved if we are able to transmit data streams in real time.

- Annotations on the client’s machine should be exactly replicated at the same places as on the server.

- The client’s I-book should be synchronized with the one on the server. In this thesis we met most of the requirements of the system. Few are left and are the result of limitations of the I-book.

- Audio and video stream at the client side should be synchronized.

1.4 Organization of Thesis

The thesis consists of five chapters. Chapter 2 discusses the various technologies used in the implementation of the project, for example the multicasting and the Real-
Time Transport Protocol (RTP). Chapter 3 discusses in detail the various hardware and software components without which it was impossible to implement the project. Chapter 3 also presents details about the smart board and Java™ Media Framework (JMF) technology. Chapter 4 provides details of the project from the programmer’s point of view. It gives an overview of all the source codes and also explains the whole process using abstract code segments. Performance analysis of the working project is presented in chapter 5. Final chapter presents the conclusion and discusses future work to extend interactive web classroom.

1.5 Summary

In this chapter we discussed the Interactive teaching laboratory and the equipment available in the laboratory for successful implementation of the project. The main objectives, which should be met in the different parts of the project, were also discussed.
CHAPTER 2
MULTICASTING AND REAL-TIME TRANSPORT PROTOCOL

This chapter provides a technical introduction to the IP Multicast and the Real-Time Transport Protocol (RTP) concepts and their technical features. It discusses the requirements for IP Multicast delivery, addressing and host group management, and approaches to multicast routing. It also discusses other techniques for transmission and the reason for choosing multicasting and the Real-Time Transport Protocol for our project.

2.1 Broadcasting and Multicasting

To transmit data to more than one node on a network, we use either broadcasting or multicasting. Broadcasting is the process of sending one copy of the datagram to all nodes on a network. Broadcasting can be an efficient way of communicating same information to all nodes on a network. However, if all nodes on the network do not need this information, broadcasting is very inefficient and leads to additional traffic on the network. It also results in processing overhead by nodes that do not need the broadcast information. In our project we need to transmit audio and visual information to a select group of students, who are participating in the web-classroom. Students participating in the web-classroom are not on the same network. So broadcasting cannot be used in our implementation. To avoid this problem we use multicasting. Multicasting is the process of sending a copy of the datagram to a group of nodes on the network. The most interesting feature of multicasting is that the server sends only one datagram to the group, which is received by all the clients who have joined the multicasting group. In the
following sections we discuss the advantages and various technical details of multicasting.

2.1.1 Advantages of Multicasting

From above discussion, it is very clear that broadcasting cannot be used when sending the same information to several recipients outside the local subnet. Unicast can be the other technique. But if the number of the clients increases, unicast consumes server resources and generates excessive network traffic. This is illustrated by an example: Let us assume that we have to transmit same information to 500 clients. Each transmission takes about 4 seconds. Unicasting this information to all the clients will take $500 \times 4 = 2,000$ seconds, whereas multicasting this information will take only 4 seconds because we have to send only one copy of the information to the multicasting group. This problem can further aggravate if this data changes frequently and retransmissions to all recipient must be made a few times. From above example we see that multicasting is an efficient method to transmit information to more than one client.

2.1.2 Internet Protocol (IP) and Multicast Addresses

All nodes on a TCP/IP network are defined by a unique address. This address is a standard 32-bit IP address that contains sufficient information to uniquely identify a network and a specific host on that network [4]. As shown in figure 2-1, an IP address consists of two parts i.e. netid, which identifies the network and hostid, which identifies the host on that network.

All the IP addresses are divided into five classes namely A, B, C, D and E. The class E is reserved for future use. This classification is based on the number of bits used to identify the netid field of the IP address format.
• **Class A.** The first byte is used for the netid and the remaining three bytes are used for the *hostid*.

• **Class B.** The first two bytes are used for the netid and the remaining two bytes are used for the *hostid*.

<table>
<thead>
<tr>
<th>netid</th>
<th>hostid</th>
</tr>
</thead>
<tbody>
<tr>
<td>N Bits</td>
<td>32-N Bits</td>
</tr>
</tbody>
</table>

Figure 2-1. Address format in IP

• **Class C.** The first three bytes are used for the netid and the remaining one byte is used for the *hostid*.

• **Class E** This class is reserved for multicasting and identified by the pattern “11110” at the most significant bits of the IP address format.

### 2.1.3 Group Participation in IP Multicast

We can use IP multicasting to send a datagram to a select group of nodes. The class D address format is reserved for multicasts. A multicast address is similar to radio channel that operates at a particular frequency. Radio receivers can tune in to the channel that operates at a particular frequency and hear the radio broadcast at that frequency, and they can also tune out [5]. The members of a multicast group join the group dynamically and can leave the group at any time.

Membership in a multicast group is required to receive a multicast datagram sent to the group’s multicast class D address. A host can simultaneously be a member of several multicast groups. A host does not need to be a member of multicast group to send a multicast. Multicast addresses that are available for temporary use are called transient multicast groups.
2.1.4 Multicast Prerequisites

To send a multicast on networks connected by routers requires that all the routers in that network be able to forward multicast datagrams. Conventional routers that are enabled for multicasting can forward multicast datagrams. The following are the requirements for multicast at the end node hosts [6]:

- All routers between sender and the receiver must be multicast enabled.
- Firewalls should be configured for multicasting.
- Support for multicast transmission and reception in the TCP/IP protocol stack.
- Software supporting IGMP to communicate requests to join a multicast group(s) and receive multicast traffic.
- Network interface cards that can filter LAN data link layer addresses mapped from network layer multicast addresses.
- IP multicast application software such as video conferencing.

The multicast has broad and growing industry backing, and is supported by many vendors of network infrastructure elements such as routers, switches, TCP/IP stacks, network interface cards, desktop operating systems and application software. Figure 2-2 shows components that must be multicast-enabled. The direction of traffic shown is for multicast datagrams.

2.1.5 Time To Live (TTL)

The time-to-live (TTL) parameter is measured in seconds and represents the maximum time an IP datagram can live on the network. Each multicast packet has a time-to-live (TTL) field as a part of the IP header. The TTL field specifies the number of hops that an IP multicast packet can make before it expires. Each time a router forwards a packet, its TTL is decremented. When the TTL field becomes 0, the TTL timer expires.
This expiration causes a multicast packet to be discarded by a router, but not by the destination host. This mechanism prevents unnecessary traffic on the Internet.

The following are the other consideration for the TTL.

- A host must not send a datagram with a TTL value of 0, and a host must not discard a datagram just because it was received with a TTL less than 2.

- A fixed value TTL must be at least big enough to handle the longest possible path on the Internet.

Figure 2-2. Multicasting basics
The IP layer must provide a means for the transport layer to set the TTL field of every datagram that is sent. When a fixed TTL is used, that value must be configurable.

2.1.6 Multicast Filtering Switches

IP Multicast can be optimized in a LAN by using multicast filtering switches. A multicast-aware switch provides the same benefits as a multicast router, but in the local area [6]. Without one, the multicast traffic is sent to all segments on the local subnet. A multicast aware switch can automatically set up multicast filters so the multicast traffic is only directed to the participating end nodes.

Figure 2-3. Multicast filtering switches

2.2 The Internet Group Management Protocol (IGMP)

Multicast packets from remote sources must be relayed by routers, which should only forward them on to the local network if there is a recipient for the multicast host.
group on the LAN. The Internet Group Management Protocol (IGMP) is used by multicast routers to learn the existence of host group members on their directly attached subnets. It does so by sending IGMP queries and having IP hosts report their host group memberships [6].

Hosts that join or leave a multicast group use the Internet Group Management Protocol (IGMP) to report their group memberships to neighboring routers. The IGMP is implemented by the IP module, but it uses IP to carry messages. So is loosely analogous to Internet Control Message Protocol [5]. The IGMP has only two kinds of packets: Host Membership Query and Host Membership Report, with the same simple fixed format containing some control information in the first word of the payload field and a class D address in the second word.

The IGMP message format is shown in figure 2-3.

<table>
<thead>
<tr>
<th>Version Bits 0-3</th>
<th>Type Bits 4-7</th>
<th>Code Bits 8-15</th>
<th>Checksum 16-31</th>
</tr>
</thead>
<tbody>
<tr>
<td>Group Address</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 2-4. Details of multicasting IP addresses

The version field is the version number of the protocol and is currently set to 1. The “Type” field identifies whether the IGMP message is a query sent by a multicast router or a response sent by a host to a multicast router query:

- Host Membership Query (Type = 1)
- Host Membership Report (Type = 2)
The unused field must be set to 0 upon sending and is ignored upon reception. The checksum field is the usual 16-bit one’s complement of the one’s complement sum of the 8-octet IGMP message.

To determine if any hosts on a local subnet belong to a multicast group, one multicast router per subnet periodically sends a multicast IGMP Host Membership Query to all IP end nodes on its network, asking them to report back on the host, group memberships of their processes. The IGMP Host Membership Query is sent to the multicast group address 224.0.0.1 in an IP header with a TTL value of 1. The TTL of 1 is used so that these queries are not propagated outside of the LAN. Hosts respond to this query by sending an IGMP host membership report message. The response message reports each multicast group address for which the host is a member on the network interface from which the query was received. Each host sends back one IGMP Host Membership Report message per host group, sent to the group address, so all group members see it (thus only one member reports membership). To avoid simultaneous response messages and to reduce the number of response messages, the following technique is used:

- Host does not send the IGMP host membership report message immediately after receiving an IGMP host membership query. Instead it starts a report timer delay for each of its group memberships on the network interface of the incoming query. When a timer expires, a response message for the corresponding multicast group is sent.

- A response message is sent with an IP destination address equal to the host group address being reported and with an IP TTL of 1. This address ensures that other members of the same multicast group on the same network can receive the response message.
Typically, only one IGMP host membership report message will be generated for each group present on the network. This message is generated by the host whose delay timer expires first.

The IGMP updates are used by multicast routing protocols to communicate host group memberships to neighboring routers, propagating group information through the internetwork. The IGMP is used to identify a designated router in the LAN for this purpose. The bandwidth needed to transmit host group information is usually slight compared to the multicast application traffic, so this propagation method is workable. More sophisticated methods enable routers to determine dynamically how to best forward the multicast application traffic [6].

2.3 Multicast Routing Concepts

2.3.1 Multicast Routing

Routing multicast traffic is not as simple as IP routing. A multicast address does not specify a physical machine but a particular transmission session consisting of many hosts. These hosts can be on the same or different networks. An individual host joins a multicast session by sending IGMP message to its subnet router. A naive approach to sending data to multiple receivers would be for the source to maintain a table identifying all the receiving subnets participating in the session and to send a separate copy of the data to each receiving subnet. However, this would be an extremely inefficient use of bandwidth, since many of the data streams would follow the same path throughout much of the network [6].

New techniques have been developed to address the problem of efficiently routing multicast traffic. Since the number of receivers for a multicast session can potentially be quite large, the source should not need to know all the relevant addresses. Instead the
network routers must somehow be able to translate multicast addresses into host addresses. The basic principal involved in multicast routing is that routers interact with each other to exchange information about neighboring routers. To avoid duplication of effort, a single router is selected (via IGMP) as the Designated Router for each physical network [6].

2.3.2 Spanning Trees

For efficient transmission, Designated Routers construct a spanning tree that connects all members of a multicast group. A spanning tree has just enough connectivity so that there is only one path between each pair of routers, and it is loop-free. If each router knows which of its lines belong to the spanning tree, it can copy an incoming multicast datagram onto all of its outgoing branches, generating only the minimum needed number of copies. Messages are replicated only when the tree branches, thus minimizing the number of copies of the messages that are transmitted through the network [6].

Because multicast groups are dynamic, with members joining or leaving a group at any time, the spanning tree must be dynamically updated. Branches in which no listeners exist must be discarded (pruned). A router selects a spanning tree based on the network layer source address of a multicast packet, and prunes that spanning tree based on the network layer destination address.

2.4 IP Multicast Backbone on the Internet (MBone)

MBone is defined as a set of IP multicast enable network on the Internet, which are interconnected. MBone provides a faster technology for transmitting real-time audio and video programs, and for videoconferencing. All Internet backbone routers do not support native IP multicast. Due to this MBone is constructed with tunnels across
networks that do not support multicast routing. MBone therefore is a virtual network layered on top of Internet.

Figure 2-4 describes how tunneling is used to carry the multicast traffic across networks that do not support native IP multicasting.

Encapsulation in tunnels is IP in IP, i.e. multicast IP packets are attached to an IP IP header, where source and destination addresses are addresses of the tunnel endpoints. The tunnel endpoints are routers or workstations with mrouted software.

Routing protocol used in MBone is DVMRP. Some parts may also use locally MOSPF or PIM. Implementing multicast routing in the whole Internet is not yet possible, because none of the current multicast routing protocols scale to such a big network. Development is going towards shared spanning-tree algorithms (PIM-SM and CBT protocols), which are designed to scale to the whole Internet [7].

2.5 Streaming Media and Real-Time Transport Protocol

In this section we will discuss streaming media and the Real-Time Transport Protocol, which is used to transmit audio and visual data from our Interactive Teaching Laboratory (ITL). To send or receive a live media broadcast or conduct a video conference over the Internet or Intranet, we need to be able to receive and transmit media streams in real-time. This section introduces streaming media concepts and describes the Real-time Transport Protocol the Java™ Media Framework (JMF) uses for receiving and transmitting media streams across the network.

2.5.1 Streaming Media

When media content is streamed to a client in real-time, the client can begin to play the stream without having to wait for the complete stream to be downloaded. In fact, the stream might not even have a predefined duration--downloading the entire stream
before playing it would be impossible. The term streaming media often used to refer to both this technique of delivering content over the network in real-time and the real-time media content that is delivered [8].

Figure 2-4. Tunnelling in non multicast networks
Streaming media is everywhere we look on the web-live radio and television broadcasts and webcast concerts and events are being offered by a rapidly growing number of web portals, and it is now possible to conduct audio and video conferences over the Internet. By enabling the delivery of dynamic, interactive media content across the network, streaming media is changing the way people communicate and access information [8].

2.5.2 Protocols for Streaming Media

Transmitting media data across the net in real-time requires high network bandwidth. It is easier to compensate for lost data than to compensate for large delays in receiving the data [8]. This is very different from accessing static data such as a file, where the most important thing is that all of the data arrive at its destination. Consequently, the protocols used for static data don't work well for streaming media.

The HTTP and FTP protocols are based on the Transmission Control Protocol (TCP). TCP is a transport-layer protocol designed for reliable data communications on low-bandwidth, high-error-rate networks. When a packet is lost or corrupted, it's retransmitted. The overhead of guaranteeing reliable data transfer slows the overall transmission rate.

For this reason, underlying protocols other than TCP are typically used for streaming media. One that's commonly used is the User Datagram Protocol (UDP). UDP is an unreliable protocol; it does not guarantee that each packet will reach its destination. There's also no guarantee that the packets will arrive in the order that they were sent. The receiver has to be able to compensate for lost data, duplicate packets, and packets that arrive out of order.
Like TCP, UDP is a general transport-layer protocol—a lower-level networking protocol on top of which more application-specific protocols are built. The Internet standard for transporting real-time data such as audio and video is the Real-Time Transport Protocol (RTP) [8].

2.6 Real-Time Transport Protocol

RTP provides end-to-end network delivery services for the transmission of real-time data. RTP is network and transport-protocol independent, though it is often used over UDP.

RTP can be used over both unicast and multicast network services. Over a unicast network service, separate copies of the data are sent from the source to each destination. Over a multicast network service, the data is sent from the source only once and the network is responsible for transmitting the data to multiple locations. Multicasting is more efficient for many multimedia applications, such as video conferences. The standard Internet Protocol (IP) supports multicasting [8].

2.6.1 RTP Services

RTP enables us to identify the type of data being transmitted, determine what order the packets of data should be presented in, and synchronize media streams from different sources.

RTP data packets are not guaranteed to arrive in the order that they were sent—in fact, they are not guaranteed to arrive at all. It's up to the receiver to reconstruct the sender's packet sequence and detect lost packets using the information provided in the packet header [8].
While RTP does not provide any mechanism to ensure timely delivery or provide other quality of service guarantees, it is augmented by a control protocol (RTCP) that enables us to monitor the quality of the data distribution. RTCP also provides control and identification mechanisms for RTP transmissions. If quality of service is essential for a particular application, RTP can be used over a resource reservation protocol that provides connection-oriented services.

### 2.6.2 RTP Architecture

An RTP session is an association among a set of applications communicating with RTP. A session is identified by a network address and a pair of ports. One port is used for the media data and the other is used for control (RTCP) data [8]. A participant is a single machine, host, or user participating in the session. Participation in a session can consist of passive reception of data (receiver), active transmission of data (sender), or both.

Each media type is transmitted in a different session. For example, if both audio and video are used in a conference, one session is used to transmit the audio data and a separate session is used to transmit the video data. This enables participants to choose which media types they want to receive--for example, someone who has a low-bandwidth network connection might only want to receive the audio portion of a conference.
2.6.3 Data Packets

The media data for a session is transmitted as a series of packets. A series of data packets that originate from a particular source is referred to as an RTP stream [8]. Each RTP data packet in a stream contains two parts, a structured header and the actual data (the packet's payload). The header of an RTP data packet contains the following information:

- The RTP version number (V), which is of 2 bits. The version defined by the current specification is 2.

- Padding is 1 bit. If the padding bit is set, there are one or more bytes at the end of the packet that are not part of the payload. Last byte in the packet indicates the number of bytes of padding. The padding is used by a few encryption algorithms.

- Extension (X) is also 1 bit. If the extension bit is set, the fixed header is followed by one header extension. This extension mechanism enables implementations to add information to the RTP Header.

- CSRC count is 4 bits. The number of CSRC identifiers that follow the fixed header. If the CSRC count is zero, the synchronization source is the source of the payload.

- Marker is 1 bit. A marker bit defined by the particular media profile.

- Payload type (PT) is 7 bits. An index into a media profile table that describes the payload format. The payload mappings for audio and video are specified in RFC 1890.

- Sequence number is 16 bits. A unique packet number that identifies this packet's position in the sequence of packets. The packet number is incremented by one for each packet sent.

<table>
<thead>
<tr>
<th>v</th>
<th>p</th>
<th>x</th>
<th>cc</th>
<th>m</th>
<th>pt</th>
<th>Sequence Number</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Timestamp</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Synchronization Source (SSRC)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Content Source (CSRC) (0-15)</td>
</tr>
</tbody>
</table>

Figure 2-6. RTP data-packet header format [8]
• Timestamp is 32 bits. Reflects the sampling instant of the first byte in the payload. Several consecutive packets can have the same timestamp if they are logically generated at the same time—for example, if they are all part of the same video frame.

• SSRC is 32 bits. It identifies the synchronization source. If the CSRC count is zero, the payload source is the synchronization source. If the CSRC count is nonzero, the SSRC identifies the mixer.

• CSRC is 32 bits each. It identifies the contributing sources for the payload. The number of contributing sources is indicated by the CSRC count field; there can be up to 16 contributing sources [8]. If there are multiple contributing sources, the payload is the mixed data from those sources.

2.6.4 Control Packets

In addition to the media data for a session, control data (RTCP) packets are sent periodically to all of the participants in the session. RTCP packets can contain information about the quality of service for the session participants, information about the source of the media being transmitted on the data port, and statistics pertaining to the data that has been transmitted so far.

There are several types of RTCP packets:

• Sender Report
• Receiver Report
• Source Description
• Bye
• Application-specific

RTCP packets are "stackable" and are sent as a compound packet that contains at least two packets, a report packet and a source description packet. All participants in a session send RTCP packets. A participant that has recently sent data packets issues a sender report. The sender report (SR) contains the total number of packets and bytes sent
as well as information that can be used to synchronize media streams from different sessions.

Session participants periodically issue receiver reports for all of the sources from which they are receiving data packets [8]. A receiver report (RR) contains information about the number of packets lost, the highest sequence number received, and a timestamp that can be used to estimate the round-trip delay between a sender and the receiver.

The first packet in a compound RTCP packet has to be a report packet, even if no data has been sent or received an empty receiver report is sent.

All compound RTCP packets must include a source description (SDES) element that contains the canonical name (CNAME) that identifies the source. Additional information might be included in the source description, such as the source's name, email address, phone number, geographic location, application name, or a message describing the current state of the source.

When a source is no longer active, it sends an RTCP BYE packet. The BYE notice can include the reason that the source is leaving the session [8]. RTCP APP packets provide a mechanism for applications to define and send custom information via the RTP control port.

### 2.6.5 RTP Applications

RTP applications are divided into those that need to be able to receive data from the network (RTP Clients) and those that need to be able to transmit data across the network (RTP Servers). Some applications do both--for example, conferencing applications capture and transmit data at the same time that they're receiving data from the network [8].
2.6.6 Receiving Media Streams From the Network

Being able to receive RTP streams is necessary for several types of applications. For example:

- Conferencing applications need to be able to receive a media stream from an RTP session and render it on the console.

- A telephone answering machine application needs to be able to receive a media stream from an RTP session and store it in a file.

- An application that records a conversation or conference must be able to receive a media stream from an RTP session and both render it on the console and store it in a file.

2.6.7 Transmitting Media Streams Across the Network

RTP server applications transmit captured or stored media streams across the network. For example, in a conferencing application, a media stream might be captured from a video camera and sent out on one or more RTP sessions. The media streams might be encoded in multiple media formats and sent out on several RTP sessions for conferencing with heterogeneous receivers [8]. Multipart conferencing could be implemented without IP multicast by using multiple unicast RTP sessions.

2.7 Summary

In this chapter we discussed the IP multicasting. The IP multicasting enables many new types of applications and reduces network congestion and server loads. IP multicast products and services are receiving widespread industry attention because of their potential benefits. Advances are being made in areas such as reliable multicasting, real-time applications support, and network management and diagnosis.
CHAPTER 3
VARIOUS HARWARE AND SOFTWARE COMPONENTS OF THE PROJECT

In this chapter we discuss various technical components, which together lead to an overall working project. We also discuss present technologies used to implement the various system components. Detailed working knowledge of all the hardware (customized and conventional) is also provided.

In this section we start with the different components of the system.

3.1 Technical Considerations for Project

As discussed in Chapter 1, we see that whole system can be divided in the following three major parts

- Server (Machine in Interactive Teaching Laboratory, which serves all the clients)
- Communication Medium (Network, which connects the server to the clients)
- Client (Machine at student side)

Now we discuss each of these with respect to their technical as well as implementation details.

3.1.1 Server

The server plays a crucial part in this set-up as there is direct relationship between the server speed and its capability to serve clients. The faster it processes data from the smart board and microphone, the less time it takes to multicast the stream on the net and thus clients are better served. While thinking about the machine configuration of the server we should make sure that it meets minimum requirements so that it can server the clients efficiently. The following points must be kept in mind while setting up server.
• Processor speed
• RAM
• Sound card
• Network card
• Microphone
• Server program up and serving all the requests

Because hardware is affordable, a system with a Pentium IV processor, 512MB RAM, 10/100Mbps fast PCI Ethernet card and 20GB hard disk would be an ideal server for our requirements. Also it must have a good quality full-duplex sound card and wireless microphone so the teacher has the freedom to move in the class while giving a lecture.

3.1.2 Communication Media

This is the most critical component in the project. In fact the whole project is based on the success we achieve in utilizing this media efficiently. As we go into the detail of the media we find out that we have different kind of medium for different clients. Below is the exact classification of the types of medium for the clients.

3.1.3 Ethernet LAN

As in the Interactive Teaching Laboratory, we have 10 Mbps Ethernet Local Area Network (LAN) and all the clients are connected to the server through this link so we would not find any problem in implementing a local interactive classroom in which the clients machine is connected to the server through high-speed LAN.

For remote clients the configuration varies as the clients can have different types of connection to the Internet. It also depends on the Internet Service Provider (ISP). In
countries where each ISP handles large geographic areas so the effective bandwidth for each user decreases which has to taken care of while designing communication protocol. We discuss most common equipments to connect to Internet, in detail.

3.1.4 Cable modem

Cable modems link computer to the Internet through local cable television (i.e., CATV) networks. The CATV network may provide unidirectional as well as bi-directional access to the Internet. With both unidirectional and bi-directional services, information can be downloaded at the speed of 10 Mbps and potentially 28 Mbps in the future. At 10 Mbps, a cable modem can transmit nearly 1000 pages of text in about 2 seconds while a 28.8 Kbps modem- which is still the most common way of accessing the internet-would not get pass the third page within same period of time. As cable modem can support multiple computers, so if we are able to reduce bandwidth requirements of the client/server communication, it is possible to have small group of students attending the class at same time, this would be very beneficial for technical colleges across the world.

3.1.5 Modem connections

Basically modems are not suitable for streaming media because of their low data transfer rates (28.8K and 56K), but modem is the most widely used medium to connect to the Internet, so we are designing the system in such a way that it works well with 56K modems. Recently, some vendors have announced the “dual modem” solutions that combine data from several modems and combine them into single stream for greatly improved throughput, but main disadvantage of this system is that many telephone lines have to be reserved for a single system.
3.2 Hardware Components and their Technical Details

In the previous section we discussed various components of the system. In this section we will discuss different hardware components, which make the project feasible. Main motivation for the project comes from smart board, which gives an alternative way of transmitting visual data by providing very low bandwidth communication. Here follows details of every hardware component of the system:

![Smart board](image)

Figure 3-1. Smart board

3.2.1 Smart Board

The smart board in Interactive classroom is replacement of black board in conventional classroom. Smart board is an interactive whiteboard that improves the way you meet, train, present and teach. It combines the look and feel of a regular blackboard with the power of a computer so we can save and print notes, collaborate on electronic documents, share information and run multimedia materials [9].

3.2.2 Components of Smart Board

1. Touch-sensitive screen. A touch-sensitive screen is basically an interface between teacher and computer. It is a transducer that converts the movements on its surface to the electrical signals. It can be used in two modes.
• Mouse mode. In this mode touch screen just acts as a touch pad, which is usually present in modern laptops. The finger can be used as a mouse pointer. One taps the board to emulate a mouse click.

• Board mode. In this mode, the smart board acts as an actual board. We pick up a pen from the pen tray and write on the board and whatever we write on smart board is actually written on the server’s screen attached to the smart board. A point worth mentioning is that nothing gets written on the board itself; only on the server’s monitor.

2. Three pen trays (red, green, black). These pen trays hold pens. The trays have sensors that recognize the different colors of pens used to write on the board. Three colors of pens (red, green and black) are available to write with.

3. Electronic duster tray. This tray is used to hold duster and it has sensor that recognizes duster pickup event. In this mode duster is used to erase annotations on the board.

4. Power supply. A 12-volt direct current (DC) power supply is used to power up the touch-sensitive panel and various sensors present on the smart board.

5. Serial interface. Serial interface cable is used to connect smart board to the server. It is very important to mention that smart board components (touch sensitive panel) basically emulate mouse actions. For example picking up a pen from the pen tray emulates a mouse left click and holding and writing on the smart board emulates mouse dragging. This interface used a data rate of 9600 Bd.

When combined with an LCD panel or projector, the smart board becomes a large, touch-sensitive screen. We can control Windows or Macintosh applications by simply touching the board. We can move our finger on the board just as we would use a mouse at your desktop to move between spreadsheets, word-processing documents,
presentation software, CD ROMs or Web sites. Also we can pick up a pen from the smart pen tray and write notes over your applications in electronic ink [9].

3.2.3 Projector

As we know that when we move electronic pen on the smart board annotations are not visible on the smart board but on the computer screen. It is because the smart board is not a display screen but just a touch pad. In this scenario it is very difficult for the teacher to make out what he is writing unless he keeps a constant vigil on the server’s monitor. A solution to this problem is to project the server screen on the smart board so that it gives an illusion of smart board actually showing what we write on it. So a projector is an excellent tool for people who are sitting in Interactive classroom.

3.3 Java<sup>TM</sup> Media Framework

In this section we discuss various software technologies, which made it possible, the successful completion of interactive web-based classroom. Java<sup>TM</sup> was an obvious choice because it provides an independent platform for application development. This was preferred platform because it offers excellent environment to handle time-based media.

The Java<sup>TM</sup> Media Framework (JMF) is an application programming interface (API) for incorporating time-based media into Java applications and applets.[jmf reference]. The JMF provides a unified architecture and messaging protocol for managing the acquisition, processing, and delivery of time-based media data. JMF is designed to support most standard media content types, such as AIFF, AU, AVI, GSM, MIDI, MPEG, QuickTime, RMF, and WAV [10]. In our project we made extensive use of JMF as it provide excellent ways of processing and synchronizing media streams.
This chapter discusses in detail architecture of JMF and different components related to our project.

### 3.3.1 Architecture of JMF

The JMF uses same basic model as of conventional techniques of capturing and presenting time based media. A data source encapsulates the media stream just like a video tape [8]. A player provides processing and control mechanisms similar to a VCR. Playing and capturing audio and video with JMF requires the appropriate input and output devices such as microphones, cameras, speakers, and monitors.

Data sources and players are integral parts of JMF’s high-level API for managing the capture, presentation, and processing of time-based media. JMF also provides a lower-level API that supports the seamless integration of custom processing components and extensions. This layering provides Java developers with an easy-to-use API for incorporating time-based media into Java programs while maintaining the flexibility and extensibility required to support advanced media applications and future media technologies [8].

![High-level JMF architecture](image)

**Figure 3-2.** High-level JMF architecture

### 3.3.2 JMF Managers

The JMF API consists mainly of interfaces that define the behavior and interaction of objects used to capture, process, and present time-based media.
Implementations of these interfaces operate within the structure of the framework. By using intermediary objects called Managers, JMF makes it easy to integrate new implementations of key interfaces that can be used seamlessly with existing classes [8].

The JMF uses four managers:

- **Manager**: Manager handles the construction of Players, Processors, DataSources, and DataSources. This level of indirection allows new implementations to be integrated seamlessly with JMF. From the client perspective, these objects are always created the same way whether the requested object is constructed from a default implementation or a custom one.

- **PackageManager**: PackageManager maintains a registry of packages that contain JMF classes, such as custom Players, Processors, DataSources, and DataSinks.

- **CaptureDeviceManager**: CaptureDeviceManager maintains a registry of available capture devices.

- **PlugInManager**: PlugInManager maintains a registry of available JMF plug-in processing components, such as Multiplexers, Demultiplexers, Codecs, Effects and Renderers.

To write programs based on JMF, we use the Manager’s create methods to construct the Players, Processors, DataSources, and DataSinks. In our project we extended JMF functionality by implementing a new plug-in, CustomPacketizer and CustomDePacketizer. To do so we have to register it with the PlugInManager to make it available to Processors that support the plug-in API [10].

In this section we saw the high level architecture of JMF and its various components. In order to understand details of the source codes in our project, we need to understand some technical terms used by JMF, which are used extensively in chapter 4. In the following sub-sections we discusses all these technical terms in detail.
3.3.3 **Data Source**

Data source is an object that implements the DataSource interface in Java™ Media Framework to encapsulate the location of media and the protocol and software used to deliver the media [10]. In our project we use data source to capture smart board coordinates and professor’s voice.

3.3.4 **Player**

A Player processes an input stream of media data and renders it at a precise time [11]. A data source is used to deliver the input media-stream to the Player. The rendering destination depends on the type of media being presented. Functioning of the Player is shown in figure 3-3.

![Diagram of Player](image)

Figure 3-3. Working of player

A Player does not provide any control over the processing that it performs or how it renders the media data. So it can only be used where we don’t need to access the contents of media stream and contents should be recognized by the Player.

3.3.5 **Processor**

Processor is an extended form of a Player and can be used to present media data. Processor has a quality that it provides control over what processing is performed on the input media stream [8]. A Processor supports all of the same presentation controls as a Player.
In addition to rendering media data to presentation devices, a Processor can output media data through a DataSource so that it can be presented by another Player or Processor, further manipulated by another Processor, or delivered to some other destination, such as a file [10].

Figure 3-4. Working of processor

Detailed information about how processor can used to process the contents of the media stream is given in figure 3-4. The different steps shown in figure 3-4 are explained below.

- **Demultiplexing** is the process of parsing the input stream. If the stream contains multiple tracks, they are extracted and output separately. For example, a QuickTime file might be demultiplexed into separate audio and video tracks. Demultiplexing is performed automatically whenever the input stream contains multiplexed data.

- **Pre-processing** is the process of applying effect algorithms to the tracks extracted from the input stream.

- **Transcoding** is the process of converting each track of media data from one input format to another. When a data stream is converted from a compressed type to an uncompressed type, it is generally referred to as decoding. Conversely, converting from an uncompressed type to a compressed type is referred to as encoding.

- **Post-Processing** is the process of applying effect algorithms to decoded tracks.

- **Multiplexing** is the process of interleaving the transcoded media tracks into a single output stream. For example, separate audio and video tracks might be multiplexed into a single MPEG-1 data stream.

- **Rendering** is the process of presenting the media to the user.
3.3.6 **Codec**

Codec is short form for Coder/Decoder. In JMF implementation it is used to process the output tracks from the processor. A series of Codecs can be applied to the same track in our project we use Codec to get the smart board information from false audio stream.

3.3.7 **Renderer**

Renderer processes the media data in a track and delivers it to a destination such as a screen or speaker [8]. When we use renderer we don’t do any processing to the media track but media track is directly destined for audio or video playback. In our project we use renderer to play professor’s voice on the client machine.

3.4 **Summary**

In this chapter we discussed various technical consideration for the project. Details of the hardware and software components, which made the project feasible, are also provided.
CHAPTER 4
SYSTEM ARCHITECTURE AND IMPLEMENTATION

In the last chapter we discussed the main hardware and software components of the project and their technical details. In this chapter we see how we can combine all these components in a working system by designing our customized protocol. This chapter mainly deals with the design issues from the programmers point of view i.e. all the technical details such as underlying system architecture, system logical flow, details of communication between server and client, synchronization between visual and audio data streams and step by step system implementation process have been discussed in full details.

4.1 Introduction To Common Communication Methodologies on Internet

At present we have following technologies for communication over the Internet, Transmission Control Protocol (TCP), User Datagram Protocol (UDP), Real-Time Transport Protocol (RTP) and Real Time Streaming Protocol (RTSP).

- Transmission Control Protocol most widely used communication over the Internet for reliable communication. TCP provides end-to-end reliability between an application process running on one computer system to another application process running on another computer [5]. It does error-correction, which in the case of streaming audio and video means the stream is effectively stopped while the error-correction takes place. The stream may then sound unnatural. It can also increase network traffic considerably. Looking in to all these points we think that TCP is not suitable for our project implementation.

- User Datagram Protocol (UDP), on the other hand, does not do error correction. UDP is useful in applications that are command/response oriented and in which the commands and responses can be sent in a single datagram [5]. Bad packets are just dropped, which can cause a noticeable dropout when listening to the stream, but the flow of sound continues uninterrupted, so the effect is minimal. The downside of
UDP is that it does not always work within firewalls. Most of the high level protocols such as RTSP and RTP are based on UDP.

- Real Time Streaming Protocol is a new standard developed specifically to enable more efficient streaming of audio and video data. It has the potential to simplify and enhance the transfer of streaming audio. RTSP is an application-level client-server protocol for control over multi-media delivery and control. Its goal is to offer robust protocol that can stream multimedia in "one to many"-applications. It also supports interoperation between clients and servers from different vendors. RTSP is a so-called out-of-band protocol. The RTSP messages are sent out-of-band, whereas the media stream, whose packet structure is not defined by RTSP, is considered "in-band". RTSP cannot ensure in-time delivery, but it does provide timestamp and control mechanisms for synchronizing different streams with timing properties. Also it doesn’t define any mechanisms for recovering for packet, but does supply the necessary header information (like sequence numbers) for some forms of error recovery by retransmission. RTSP breaks data into many packets sized according to the bandwidth available between client and server. When enough packets have been received by the client, the user's software can be playing one packet, decompressing another and downloading the third. The user is able to start listening almost immediately without having to get the entire media file. RTSP is meant to control multiple data-delivery sessions and provide a way to choose delivery channels such as UDP, TCP and IP-multicast. The use of a particular compression method is not enforced by RTSP.

- Real-time Transport Protocol (RTP) is an excellent way of transferring data for applications, which require response in real time over the Internet. RTP is defined in detail in Chapter 2 and is used for our project implementation.

### 4.2 Problem Solving Approach

In this section we discuss fundamental design issues and various approaches, which can be used to implement interactive web-based classroom. We also discuss various tradeoffs between different approaches and their effects on the overall system performance.

#### 4.2.1 Design Issues

The most important objective of this thesis was to develop an effective protocol for the communication between the student and the teacher on the web. This implements an interactive web classroom. To realize this concept, we need to come up with a technique to send the mouse clicks and the audio information to the students. This should
be done with minimum communication delay while preserving synchronization between audio and visual streams. While developing software solution for this we have to take care of the fact that different students use different platforms (Windows NT /98/Unix). Our solution should be independent of the operating system platform at student’s end. We discuss all these issues in the following section.

4.2.2 Approach 1

The first approach is to use technology, available for visual and audio transmission over the Internet. We discuss how we can utilize this technology in our project implementation. We also discuss factors that made us think of our own protocol for the system implementation.

At present we can send audio and video signal using commercially available products (such as Microsoft’s “NetMeeting” or Real Player server). The major advantage of using these products is that they are able to send audio and video signals simultaneously in a single stream. This means that we do not have to worry about synchronizing audio and video streams as they are implicitly synchronized. In this case we do not even need a smart board because we can put a camera in classroom and everything can be made visible to students. But to transfer video as well as audio stream, we need a high bandwidth network. This also results in the poor quality of the video at the client side because of video compression schemes used. The second big hurdle in this implementation is that student can not run interactive simulations from “Neural and Adaptive System” book, which is one of the long-term objectives of the project.

We see that high bandwidth is required for this approach. The clarity of the annotations is also sacrificed. Obviously we need to device a solution free from these hindrances of this approach, and provides a better interface.
4.2.3 Approach 2

The second approach is to use customized hardware and software solution for effectively managing the bandwidth requirements and visual quality of the annotations.

As discussed in Chapter 2, the smart board is an effective tool for converting handwritten text to electronic form and the information used to do this, is the coordinates of the huge smart board touch pad. This information is passed to the computer using a serial interface, which requires very little bandwidth. Idea behind the second approach is to transmit these coordinates to the client and reconstruct original annotations. The audio stream in this case can be transmitted using Real Time Protocol (RTP), which ensures good quality of service while dealing with streaming media.

Although this approach seems to be quite simple and effective, the biggest challenge in this approach is to synchronize the smart board coordinates with the audio stream so that they are presented at the client side, in exactly the same relative time frame as in the Interactive Teaching Laboratory (ITL).

Features and Tradeoffs

This approach though very effective when we consider the bandwidth requirements, does not provide the complete visual environment of the classroom. A student can only see what is being written on the smart board.

In the following sections we see the complete system design and implementation that made it possible to synchronize the audio and the visual stream using low bandwidth, compatible with modems (56K).
4.3 System Design

In this section, we discuss details of the system design and various considerations for the interactive teaching on the web. It also discusses the division of the project into different parts, depending on the modules functionality.

The foremost important phase is the requirements identification phase of the problem. In this phase we identify different requirements based on the information we need in different project components. As discussed earlier in Chapter 1, we know that the project can be divided into three components, which are the server, the communication medium and the client. First we discuss the abstract design of the system and then later in this section we discuss each one of these three components separately.

4.3.1 Abstract Design

As shown in the figure 4-1, system is classified in three different parts. Although all the technical details and the system implementation from the programmer point of view, are given in section 4.4 named “System Architecture”, here we give a very brief description of each component so that basic functionality becomes clear.

It is very clear from the figure that the server is the most important component of the system as it responsible for the following functions

- To get useful information from the smart board so that the client can reconstruct the annotations.
- To acquire the audio information from the microphone so that the client can listen to professor’s voice.
- To convert the smart board information to audio format in order to ensure synchronization.
- To compress the raw audio information from the microphone, which saves lot of bandwidth while transmitting over the Internet.
- To transmit both streams over the Internet in such a way that the load on the server is minimized.

- To transmit both streams in such a way that these can be synchronized easily at the client side.

**Figure 4-1. Abstract system design**

After the server starts transmitting the stream, the communication medium comes in the picture, as it is responsible for carrying information from the server to the client. In this component of the system, we mainly consider how clients are connected to the Internet (for example, modems/local area networks/digital subscriber link). The connection speed needs to be handled carefully to maintain satisfactory performance of the system. In our case, we are using a multicasting technique, which offloads the server...
by sending a single stream for all the clients. Otherwise sending different stream to all clients will unnecessarily overloads the server thereby degrading its performance. Streams are multicasted using Real Time Protocol (RTP), an effective method for transmitting real time data over the Internet.

The third important component of the system is the client. The client is responsible for receiving the streams and playing them to produce audio and annotations. Validity of the protocol is decided by performance we get on the client. In our system the client joins a multicasting group and waits for the real time stream. Once streams reach the client, client processes streams, extracts the information contained in these streams and passes it to different processes responsible for audio and visual reproduction.

In this section we provided big picture of how the system is divided into different components, which combined together make this system feasible. Details of the technology used to implement these components are discussed in the next section.

### 4.4 System Architecture

In the previous section we discussed the basic system design by dividing it into three components and also discussed fundamental approach to achieve desired functionality of each component. In this section we discuss technical implementation details of each component from the programmer’s point of view.

As discussed in Chapter 3, the implementation revolves around Java™ technology as it provides an independent platform for Internet based applications. Dealing specifically with our project, we made extensive use of two application programmers interfaces (API) offered by Java™. These technologies are Java™ Media Framework (JMF) application programmers interface and Java™ real time transport protocol application programmers interface for multicasting over the Internet.
4.4.1 Server

The server is the main component of the architecture. All the critical issues like audio and information from the smart board is captured and transmitted from the server. For our technical discussion we divide the server in two parts, one, which handles professor’s voice and another, which generates a false audio stream to transmit the smart board coordinates. These transmitted coordinates are eventually used to reproduce the annotations at the client side. We discuss details of these two server Components.

4.4.1.1 Capturing data from microphone

As shown in the figure 4-2, the professor’s voice is captured through a microphone. The data captured straight from the microphone is raw data so first we convert this data to valid format that can be used by JMF API. This is done by a JMF plug-in, which takes the raw audio data as input and encodes it to produce output.

4.4.1.2 Capturing smart board coordinates

To replicate annotations from the smart board to the client machines through communication media (local area network/Internet) we need to find out some mechanism, which keeps track of every event on the smart board. We found that the smart board actually behaves as a large mouse pad with some extra functionality added for the different event like choosing pen colors and erasing annotations. Coordinates from the smart board and special events are captured using the smart board application programming interfaces provided by SMART Technologies Inc., Calgary, Alberta, manufacturer of smart board.

Smart board driver, which runs on the machine connected to the smart board acts as a core, which provides smart board functionality and detection of various events.
Smart board API kit is only usable when smart board driver is up and running on the machine otherwise it is not possible to get special events from smart board.

The three-tier architecture used to get coordinates from smart board is shown diagrammatically in figure 4-3.

4.4.1.3 Converting coordinates to audio format

In the previous sub section, we discussed how to get smart board coordinates from the smart board. In this sub section we discuss technical details of how to convert these coordinates to the JMF compatible audio format for purposes of synchronization and RTP transmission.

To convert data from smart board to audio stream, first we need to understand the audio format that is supported by the Java™ Media Framework (JMF). Any audio data have basic characteristics such as sampling rate and bits/sample. JMF uses some additional parameters to describe a valid audio format. These parameters are explained below.

- **Encoding-audio.** Data is coded to save bandwidth as coded audio takes less space and hence can be transmitted effectively. Many coding schemes are used in practice such as GSM, LINEAR, MAC3 etc
- **Sample rate.** The sample rate represents how many sample are taken per second
- **Sample size.** The sample size defines how many bits are used to represent an audio sample
- **Channels.** The number of channels in audio stream for example 1or 2.
- **Endian.** The sample byte ordering used for this audio format
- **Signed.** Indicates whether the samples are stored in a signed or unsigned format.
Figure 4-2. Details of the server and its components
Frame size. The frame size is taken in bits

Frame rate. The frame rate

Data type. The type of the data received in audio stream. For example, byte array or integer array

Most of the parameters stated above, are used to identify the stream but the useful data is stored in an array, which is defined by the data type (byte or integer array). So the trick to convert smart board coordinates to audio format is to put x and y coordinates from the smart board to this data type which is defined to be a byte array and the rest of the parameters can have some valid values suitable for transmission on the net. By doing so we achieve a desired audio format, which contains smart board coordinates. The Real Time Protocol (RTP) recognizes this format and can easily transmit it on the net.

Merging Data Sources

The Java Media Framework media players use data sources to manage the transfer of media-content. A data source encapsulates the media stream much like a music CD. In JMF, a data source object represents the audio media, video media, or a combination of the two. A data source can be a file or an incoming stream from the
Internet. Once we determine its location or protocol, the data source encapsulates both the media location, and the protocol and software used to deliver the media. Once created, a data source can be fed into a player to be rendered, with the player unconcerned about where the data source originated or what was its original form [11].

Media data can be obtained from various sources, such as local or network files, or live Internet broadcasts. As such, the data source can be classified into the following two types according to how a data transfer initiates:

- **Pull data source.** The client initiates the data transfer and controls the data flow from the source. HTTP and FILE serve as examples of established protocols for this type of data.

- **Push data source.** The server initiates the data transfer and controls the data flow from a push data source. Push data source examples include broadcast media and video on demand. Our project implements this type of data source.

In our project we need to handle two kinds of information (visual and audio), so we need to create two different data sources, which handle both type of streams. To achieve synchronization and ease of multicast using Real-Time Transport Protocol (RTP), we combine these two audio streams by creating a merged data source. The data source created in this way provides us both the audio streams at the same time.

### 4.4.2 Communication Medium

After conditioning and merging the two audio streams the next step is to transmit the combined stream on the Internet. In our project, the Internet is the communication medium for remote students and local students are connected to the server through local area network (LAN).

We use the Real-Time Transport Protocol (RTP) to multicast combined audio stream. RTP is an efficient protocol for transmission on the Internet. Details of the RTP
transmission in our project are already covered in the chapter 2 under sub heading “Transmitting RTP media streams.”

4.4.3 Client

Client is the machine on the student side. The success of the Interactive web-classroom is measured by the efficiency achieved at the client side. Efficiency in our case is determined mainly by two factors.

- Level of synchronization achieved
- Ability to reconstruct annotations

After receiving the RTP stream from the net. The first step is to get the useful information from it and then passing this information to audio/video devices to simulate the classroom environment. We discuss each one of these steps in detail.

Receiving RTP Media Streams

JMF players and processors provide the presentation, capture, and data conversion mechanisms for RTP streams [8]. A separate player is used for each stream received by the session manager.

To receive an RTP stream we make the client, a member of the mulcasting group by creating an RTP session manager for a session, which is identified by address and port pair. The session manager then listens for different events. On receiving new stream, a processor is created to extract the two audio streams for audio/annotations reconstruction. As the false audio stream contains the smart board information, we pass this stream through a Codec to access this information. The Codec accesses the relevant information from the stream and passes to the different program, which is responsible for
reconstruction of annotations. On the other hand the audio stream containing professor’s voice is directly sent to the renderer for the audio reproduction.

4.6 Synchronization

Synchronization in this context is defined as the process of ensuring that two or more media streams are presented at the exact time specified by the author of the content [3].

4.6.1 Need for Synchronization

Humans expect audio-visual actions to happen simultaneously, and human eye is not tolerant of the contents that are presented out of sync. For example many people become disconcerted during movies when the actor’s mouth moves, but no audio is heard for several seconds [3].

In our project synchronization is even more crucial because we need to teach online, a student needs to see what the teacher is trying to explain. For this to happen, audio and visual effects should occur at the same time, otherwise it could be possible that teacher writes an equation for velocity but student receives an audio in which teacher speaks about acceleration. This occurs when there is no synchronization between drawing and audio stream and drawing stream reaches student’s machine before corresponding audio stream or vice-versa. If this happens it defeats the purpose of virtual classroom. So synchronization is extremely necessary between these two streams.

4.6.2 How We Achieved Synchronization

We converted integer coordinates to audio stream not only to transmit it effectively using real time transport protocol but also to simplify synchronization between two streams. The Java™ Media Framework accomplishes synchronization by leveraging the capabilities of the player’s clock interface. Clock interface has two time
representations media time and time base. Media time starts and stops with the presentation whereas clock’s time base is an uninterruptible flow of time from a specific starting point and it enables the clock to map a media-specific time to a media-independent time source.

As a result, if two or more clocks use the same time base, they have a common time, which can be used for synchronization purposes. The following diagram illustrates this process.

![Diagram](image)

**Figure 4-5. Synchronization details**

Using the clock’s API we create a common time base between two players but before we do this we have to decide which player contains master clock for synchronization purposes.

Once we have the master device we can inform the slave players of the time base with which should synchronize [3]. In our case the audio captured from the microphone acts as master where as the false audio stream that is used to transmit coordinates acts as the slave. This master slave relationship to set each player’s time base is shown in figure 4-6.
Figure 4-4. Details of the client and its components
4.7 Implementation Process

The following figure 4-7 presents the step-by-step implementations of the web-based interactive classroom. These steps have been developed during the implementation of this thesis and provide framework for integrating future enhancements. We discuss the major step involved in the project.

**Step 1. Write Source Code to Capture Smart Board Events**

The very first step is to write the code to listen to all the smart board events (for example, picking up pen from the pen tray, choosing pen color, writing with the electronic pen, picking eraser, and erasing from the smart board). The manufacturer of the smart board, “SMART Technologies Inc.” provided a Software Development Kit (SDK), which provides an excellent interface to the smart board. We used this kit in our source code to get all these events.

**Step 2. Pass Smart Board Coordinates to Custom Data Source**

In this phase, we write the code to pass captured coordinates to the customized data source. This is done with Java™ API for handling input/output streams for Java™ Media Framework applications. This code creates an object of all the relevant
information, which is required at the client side to exactly reproduce the annotations on
the smart board. The following code shows how we encapsulate all information in this
object.

```java
public class TestFrame2 extends Frame {

    public static ObjectOutputStream outData;

    { String co[] = { Integer.toString(lastPoint.x), // Coordinate to improve
                    reconstruction
```

Figure 4-7. Implementation process on the server side

Step 1. Write source code to capture smart board events

Step 2. Pass smart board co-ordinates to custom data source

Step 3. Capture professor’s voice from microphone

Step 4. Create a merged data source on streams created in
Step 1 and Step 3

Step 5. Create RTP session manager to transmit streams
```java
Integer.toString(lastPoint.y), // Coordinate to improve reconstruction
Integer.toString(e.getX()), // X Coordinate
Integer.toString(e.getY()), // Y Coordinate
Integer.toString(ToolType), // Tool type (Pen/Eraser)
Integer.toString(pencolor) // Pen color
};
outData.writeObject(co); // This writes object to the file
```

This object is then written to a temporary file. File created in this manner keeps on appending all the objects. In this way it provides a real time bridge between the smart board and the Java™ Media Framework data source. This example shows the code segment from “Transmit.java”, which is used to capture the smart board information and writing to a file stream.

The data source created is a custom plug-in, which extends the JMF’s PushBufferDataSource. The output of this custom data source is a false audio stream on which we again create a data source, which is used to create the processor for the transmission. The implementation details of the first data source and the false audio stream are given in source code “DataSource.java” and “FalseAudio.java”.

**Step 3. Capture Professor’s Voice from the Microphone**

This step is performed in parallel with the Step 1 and is used to capture the professor’s voice. Again we create a data source on the microphone. In this case we choose a built-in plug-in to create a custom audio stream. Characteristics of this audio stream are defined by the coding format, sampling rate, number of channels and the bits per sample etc. For our project we use the following parameters.
At the end of this phase we have two audio streams, one of which represents the professor’s voice and another represents the smart board information.

**Step 4. Create a Merged Data Source on Streams Created in Step 1 and Step 3**

After getting the two audio streams, we need to create the data source on them. This is done because we have to create a processor. A processor is a player that takes data source as an input, performs some user-defined processing on the media and then outputs the processed media data. In this case we use the processor for making the streams suitable for the RTP transmission. The following paragraph illustrates how we create a merged data source on these streams.

We can create a merged data source only on the two other data sources. So first of all we create two different data sources on each of these two streams and then create a merged data source on these two data source. This is implemented in source code “Trasmit.java.”

A code segment that is used to create data source on both streams, a single merged data source on these two streams and the processor on this merged data source, is shown below. A merged data source is created in order to synchronize audio and video streams and for the ease of transmission.

```java
private String createProcessor()
{
    if (locator == null)
        return "Locator is null";
```
DataSource ds=null;
DataSource ds1=null;
DataSource ds2=null;
DataSource clone;
    try
    {

// locator is the location of false media stream which contains smart board coordinates
    ds1 = javax.media.Manager.createDataSource(locator);

// locator2 is the location of false media stream which contains professor’s voice
    ds2 = javax.media.Manager.createDataSource(locator2);
    DataSource dsa[]= {ds1,ds2};

// Creating merged data source
    ds = javax.media.Manager.createMergingDataSource(dsa);
    }
      catch (Exception e)
    {
        System.out.println("Couldn't create DataSource");
    }

// Try to create a processor to handle the input media locator
try
{

    processor = javax.media.Manager.createProcessor(ds);

//processor = javax.media.Manager.createProcessor(ds1[1]);

}catch (NoProcessorException e)
 {
     return "Couldn't create processor ";

    }

    catch (IOException e)
{

    return "I/O error creating processor";

    }
    }
Table 4.1: Description of Source Codes

<table>
<thead>
<tr>
<th>Program Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DataSource.java</td>
<td>This program extends PushBufferDataSource and is used in conjunction with FalseAudio.java to create DataSource for smart board information. It provides all the methods to handle the DataSource.</td>
</tr>
<tr>
<td>FalseAudio.java</td>
<td>This program generates the false audio stream which contains the smart board information like X&amp;Y coordinates, tool type, Pen color etc. It provides all the methods to access audio stream information.</td>
</tr>
<tr>
<td>Transmit.java</td>
<td>This program is main the server side program, which provides all the methods to capture, process and transmit media stream using the Real-Time Transport Protocol.</td>
</tr>
<tr>
<td>Receive.java</td>
<td>This program is the main client side program that contains all the methods for receiving, processing and presenting received media stream from internet.</td>
</tr>
<tr>
<td>CustomPacketizer.java</td>
<td>This program defines custom plug-in for RTP Transmission on the server side</td>
</tr>
<tr>
<td>CustomDPacketizer.java</td>
<td>This program defines custom plug-in for RTP Reception on the client side</td>
</tr>
<tr>
<td>Codec</td>
<td>This class is a subclass of “Receive.java”. It is a Codec used on output of the processor. Its main function is to get the smart board information from audio stream and pass the relevant information to “Draw.java”, which eventually reconstruct annotations</td>
</tr>
</tbody>
</table>

Step 5. Create RTP Session Manager to Transmit streams

This is the final step taken on the server side. In this step we create an RTP manager on each of the stream got from the processor created in Step 4. The RTP session manager is responsible for querying the format of each source stream to determine if it has a registered payload type for this format. If the format of the data is not a valid RTP format or a payload type cannot be located for the RTP format, an UnsupportedFormatException is thrown with appropriate message. The RTP session managers for our project is implemented in source code file named “Transmit.java”.

All of these five steps were taken on the sever side to transmit the combined audio stream on the internet. Now we discuss the step-by-step implementation of the client side. Figure 4-8 shows the all the implementation details of the client.

Step 6. Receiving Media Stream on the Client

The very first step on the client machine is to receive the combined media stream. For that we need to specify the IP address and port number of the multicasting group, which we join in order to receive an RTP stream. We create an RTP session manager for the address and it looks for all the events associated with it. For example, ReceiveStreamEvent, NewReceiveStreamEvent and ByeEvent etc.

Step 7. Extracting the Audio Streams

After the session manager detects new stream, it extracts both the streams and sends them it to the different components based on whether a particular stream has to be processed or rendered. Step 8 and Step 9 explain this process in detail.

Step 8. Processing the False Audio Stream

In our case the stream that contains professor’s voice doesn’t need any processing so it can be rendered directly to the speakers. But in case of the false audio stream we have to extract the smart board information. For that we create a processor on it. The processor itself does not do anything to the stream, but provides stream access to the Codec, which actually processes the stream. In our project this is taken care of by “Codec.java”. The Stream output of the processor is passed through a Codec, which actually accesses the smart board information from the buffer object passed to it. This Codec in-turn passes relevant information to the program “Draw.java” that handles annotation reconstruct on the client side.
Step 9. Producing Visual Effects

This is an important step as it is responsible for the reconstruction of annotations on the clients. In our project this step is taken care of by “Draw.java”. This program takes the smart board information from Step 8 and draws annotations on the clients screen exactly as on the server.

The “Draw.java” program takes seven parameters to reconstruct annotations on the smart board. As we are getting only the coordinates from the smart board so it is tricky to construct continuous lines or drawing. This is accomplished as follows.

Constructor for “Draw.java” program looks like

```java
public void draw(int X, int Y, int X1, int Y2, int linewidth, int tool, int color)
{
    // Initialization code
}
```

The parameters X, Y, X1 and Y2 are used to reconstruct the annotations and the parameters linewidth, tool and color represent width of line to be drawn, tool type (Pen/Eraser) and color of annotations, respectively. The client can configure “linewidth” parameter to get desired thickness. The abstract code, which draws annotations on the client machine is shown below.

```java
public void paint(Graphics g) { 
    double angle;
    double Delta_x = (double)(x2 - x1);
    double Delta_y = (double)(y2 - y1);
    double R = ((double)lineWidth)/2.0;
    double cosAngle, sinAngle, Double_I;
    if (x1 == x2)
    {
        angle=Math.PI;
    }
    else
    {
```
angle = Math.atan(Delta_y/Delta_x) + Math.PI/2;
}
cosAngle = Math.cos(angle);
sinAngle = Math.sin(angle);

Figure 4-8. Details of the implementation process on the client side

\[
\begin{align*}
x_1 &= x_1 - (\text{int})(R \times \cos \theta);
y_1 &= y_1 - (\text{int})(R \times \sin \theta);
x_2 &= x_2 - (\text{int})(R \times \cos \theta);
y_2 &= y_2 - (\text{int})(R \times \sin \theta);
\end{align*}
\]

for(int I=0; I<lineWidth; I++)
{
    Double_I = (double)I;
g.drawLine(x1 + (int)(Double_I*\cos \theta),
y1 + (int)(Double_I*\sin \theta),
x2 + (int)(Double_I*\cos \theta),
y2 + (int)(Double_I*\sin \theta));
if((I+1)<lineWidth)
Step 10. Rendering Professor’s Voice Audio Stream

The second stream received is an audio stream, which contains the professor’s voice. This stream does not need any processing as it has to be presented unchanged. So we pass this stream into the inbuilt Java™ Media Framework “Renderer”. A “Renderer” delivers media in its final processed state. It is a single-input processing component with no output [8]. In our project we pass the second stream directly to the “Renderer” to present the professor’s voice to the clients.

4.8 User Interface

A very simple user interface has been provided for the server and the client. An icon has been created on both sides to start the application with the parameters specified in the batch files. The server side has batch file named “Transmit.bat” and the client side has “Receive.bat”.

The “Transmit.bat” executes the “Transmit.java” source code, which accepts the following parameters to start the transmission.

- Source for the smart board annotations (value is “smartboard:"”)
- Source for receiving the professor’s voice (default value is “javasound://:"”)
- Class-D Multicasting address (default value is “234.4.5.6”)
- Port number (default value is “45000”)

```java
{ g.drawLine(x1 + 1 + (int)(Double_I*cosAngle),
    y1 +     (int)(Double_I*sinAngle),
    x2 + 1 + (int)(Double_I*cosAngle),
    y2 +     (int)(Double_I*sinAngle));
}
}```
The first two parameters should be kept unchanged. The user can change the last two parameters if there is any conflict in the address or the port number.

On the other hand the “Receive.bat” on the client side executes the “Receive.java”. The Receive.java takes three parameters. The first two parameters are the two pairs of the multicasting group addresses and the port numbers. The first multicasting address and the port number should be exactly the same as specified in the “Transmit.bat” on the server side i.e., “234.4.5.6/45000”. The second pair has the same multicasting group address but we increase the port number by a factor of 2, which makes the default value of “234.4.5.6/45002 for the second pair. We need two pairs because we have to receive the two audio streams on the client side.

The third parameter is for the pen thickness. This parameter can be set by the user to increase or decrease the pen thickness on the client side. The valid values for this parameter are 1,2,3,4,5,6 and 7.

4.9 Summary

This chapter provided the design and the architecture for the general framework developed for the interactive web classroom. The step-by-step system implementation process and sample code discussed in the chapter gives a way for the further enhancement and improvement of the development framework.
CHAPTER 5
RESULTS

The interactive web based classroom has been implemented and tested for its
performance in the Computational and NeuroEngineering Laboratory at University of
Florida. In this chapter we discuss the performance of the interactive web based
classroom and various factors, which affect its performance.

5.1 Bandwidth Requirements

As the whole idea for the web based classroom is to make it available to all the
clients, who connect to the Internet through different connection speeds. The most
common equipments to connect to the Internet are Modems and Digital Subscriber Line
(DSL). While designing the system we kept all these constraints in mind and designed a
system, which works on all the above said equipments.

For the purposes of showing the efficiency with which we can transmit smart
board annotations using the false audio stream, we split the bandwidth requirements in
following three cases.

- Bandwidth of the professor’s voice
- Bandwidth requirements of the false audio stream (smart board annotations)
- Bandwidth requirement of the whole system

Tables 5-1 through 5-3 show the total number of Bytes transmitted for different
audio streams. All these reading were taken on Interactive Teaching Laboratory server,
which is a Pentium II Processor (MHz) with 98 MB of RAM. When we performed these
tests no other application was running on the server.
Table 5.1 shows the total number of Bytes transmitted for 1, 5 and 10 seconds respectively, by actual audio stream, which carries the professor’s voice.

Table 5-1. Data transmission statistics for the audio stream containing professor’s voice

<table>
<thead>
<tr>
<th>Serial number</th>
<th>Time for which audio stream containing professor’s voice transmitted (sec)</th>
<th>Data transmitted</th>
<th>Effective bandwidth (Kbps)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.00</td>
<td>3416</td>
<td>27.32</td>
</tr>
<tr>
<td>2</td>
<td>5.00</td>
<td>19764</td>
<td>31.62</td>
</tr>
<tr>
<td>3</td>
<td>10.00</td>
<td>40016</td>
<td>32.01</td>
</tr>
</tbody>
</table>

Table 5-2 presents total number of Bytes transmitted for 1, 5 and 10 seconds respectively, by the false audio stream, which carries the smart board annotations.

Table 5-2. Data transmission statistics for the audio stream containing the smart board annotations

<table>
<thead>
<tr>
<th>Serial number</th>
<th>Time for which audio stream containing the smart board annotations transmitted (sec)</th>
<th>Data transmitted (Bytes)</th>
<th>Effective Bandwidth (Kbps)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.00</td>
<td>1952</td>
<td>15.61</td>
</tr>
<tr>
<td>2</td>
<td>5.00</td>
<td>9272</td>
<td>14.83</td>
</tr>
<tr>
<td>3</td>
<td>10.00</td>
<td>17568</td>
<td>14.05</td>
</tr>
</tbody>
</table>

Table 5.3 presents the total number of Bytes transmitted for 1, 5 and 10 seconds respectively, for the combined stream, which contains both streams i.e. the audio stream carrying the smart board annotations and the professor’s voice. From smart board technical specifications in chapter 3, we observe that smart board communicates with computer with a speed of 9600 Bd. Where as in our case we are transmitting the same information using nearly 14-15 Kbps. The reason behind this increase in transmission rate is that we are transmitting information even when there is no movement on the smart board. Also we are sending more coordinates to the client for smooth reconstruction of annotations. These two factors result in effective increase in transmission rate.
Table 5-3. Data transmission statistics for the audio stream containing the smart board annotations and the professor’s voice

<table>
<thead>
<tr>
<th>Serial number</th>
<th>Time for which audio stream containing both smart board annotations and professor’s voice transmitted (sec)</th>
<th>Data transmitted (Bytes)</th>
<th>Effective bandwidth (Kbps)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.00</td>
<td>5368</td>
<td>42.94</td>
</tr>
<tr>
<td>2</td>
<td>5.00</td>
<td>29036</td>
<td>46.45</td>
</tr>
<tr>
<td>3</td>
<td>10.00</td>
<td>57584</td>
<td>46.06</td>
</tr>
</tbody>
</table>

From the above collected bandwidth requirements we see that the average bandwidth requirement of the system is 45.15 Kbps, which can easily be achieved by Modems/DSL on the client side and still have good performance.

The one good thing about the client side program is that we can tune the buffer length to get the desired smoothness based on the Internet connection. For example if we are connecting through Modems we can increase the buffer length, which will ensure that we have enough data to display. Doing so might end up in getting a burst of information at once but as said before we can fine tune the buffer length parameter.

At present we have set buffer length on client side to 0. This setting gives an excellent performance on Local Area Network (LAN).

5.2 Ability to Reconstruct Smart Board Annotations

The second most important issue is the ability to reconstruct smart board annotations on the client side.

Figure 5-1 shows screen capture of the smart board annotations on the server side. After comparing the screen captures of the server and the client side we see that we were able to reconstruct the annotations and the professor’s voice exactly without any loss of information.
Figure 5-1. Annotations on the server side

Figure 5-2 shows screen capture on the client side after we receive audio stream containing smart board annotations.

5.3 Ability to Synchronize the Audio and the Video Streams

During the testing phase we saw that visual and audio effects are reconstructed on the client side with the same relative time frames as on the server. This was the crucial
objective of our project, which we have met by converting the smart board information to audio stream and playing in sync with actual audio stream of the professor’s voice.

Figure 5-2. Annotations received on the client side

5.4 Summary

This chapter evaluated the performance of interactive web-based classroom and showed that we get good performance with this system. We also discussed various factors that affect the performance of the system. All the data are collected on the Interactive Teaching Laboratory server.
CHAPTER 6
CONCLUSION AND FUTURE WORK

A successful attempt was made to design and implement an efficient protocol for interactive teaching on the web. The results are encouraging enough to proceed further and to develop an ideal interactive teaching system that can be put into practical use.

6.1 Conclusion

This thesis presents a framework for an efficient protocol to simulate an actual classroom on the web. This system has totally been tested for its performance and usability on different platforms (Windows NT/Windows 98/Linux) and different communication channels such as Modems (56 K) and Local area Network.

The basic objective of achieving low bandwidth for annotations reconstruction has been achieved by using the smart board, which provides an excellent replacement for conventional black board. It uses a large touch sensitive screen to communicate with the computer. This communication has less bandwidth requirements (serial port which connects the smart board to the computer) and this motivated us to implement the web classroom on the Internet.

The Java™ Media Framework technology has been chosen to implement the project as it provides us platform independence and the ease of implementation. The original coordinate stream from the smart board was converted to the audio stream to synchronize it with the actual voice audio stream. Both streams are multicasted on the Internet using the Real-Time Transport Protocol (RTP), which provides a very effective way of sending audio streams.
This project once fully implemented can be really helpful in overcoming the existing limitations of the conventional classroom. The basic approach of converting the data stream to the audio stream and multicasting over the Internet using the Real-Time Transport Protocol (RTP) can be used for other applications as well where we need to handle large number of clients with minimum network delay. The only limitation of this approach is that we cannot apply the same solution to the problems where we need exactly bit by bit replication of the data on the client side as this approach uses the Real Time Protocol which is a lossy data transfer protocol.

6.2 Future Work

Although this framework is fully functional from the teacher’s side as synchronized streams of the smart board annotations and audio data can be transmitted efficiently on the student’s side but when the format of the I-book changes to HTML, we can make some additions to the existing system to make it more powerful and effective. Some of the features, which will make the existing system an excellent tool for the online interactive teaching, are discussed below.

6.2.1 Features to Synchronize Teacher and Student’s I-Books

As in our present system setup we project I-book on the smart board with the projector and write on it to simulate making notes on the text book. This I-book gives an interactive graphical user interface on the server to change, forward or backward or browse through pages, but on the client side student does not know which particular page the teacher is currently on. So if we can integrate this feature of synchronizing the teacher’s book with the client’s book, it would make a nice interface.
6.2.2 Features to Enable Two Way Communication

At present our system allows us to multicast the audio and the smart board annotations streams from the server to the clients. This poses a problem if students have some questions. So if we add a feature to enable the student side to communicate with the teacher and ask her questions it will make an excellent learning experience. While designing this feature we have to take care of the factors such as two or more students asking questions at the same time. So there must be some rule to assign priorities to the students so that the teacher can respond to students’ questions one at a time.

6.2.3 Web-Enabled Interface

Presently, the electronic version of the book is a hypertext document in the windows help format. The hypertext documentation has been created using RoboHELP™. The hypertext is linked to the simulator NeuroSolutions™ through Object Linking and Embedding (OLE) to run custom simulations. This limits e-book to run on Windows platform only. So we should get rid of this limitation and design an interface with the following capabilities:

- User graphical interface would be web-enabled.
- The simulation examples provided in the book should run in the popular browsers such as Netscape and Internet Explorer.
- Protocols for communication between the new and the existing NeuroSolutions™ application should be provided.

After studying into all these aspects, we conclude that this project can lead to a highly efficient way of teaching interactively on the web.
LIST OF REFERENCES


BIOGRAPHICAL SKETCH

Ravinder Pal Singh was born on November 18, 1975, in Kapurthala, Punjab, India. He received his Bachelor of Technology degree in computer science and engineering from Regional Engineering College, Jalandhar, India, in June 1996. After graduation he worked for Pertech Computers Limited as a system engineer. He joined the Department of Computer and Information Science and Engineering at the University of Florida in spring 2000. He worked as a research assistant in the Computational NeuroEngineering Laboratory. He received his Master of Science degree in May 2002. His research interests include TCP/IP and mobile networks.