uOttawa

L'Université canadienne
Canada's university

FACULTÉ DES ÉTUDES SUPÉRIEURES ET POSTOCTORALES

uOttawa

L'Université canadienne
Canada's university

FACULTY OF GRADUATE AND POSDOCTORAL STUDIES

Md. Abdur Rahman

AUTEUR DE LA THÈSE / AUTHOR OF THESIS

M.A.Sc. (Electrical Engineering)

GRADE / DEGREE

School of Information Technology and Engineering

FACULTÉ, ÉCOLE, DÉPARTEMENT / FACULTY, SCHOOL, DEPARTMENT

Design and Development of P2P Multilingual Collaborative Multimedia Tool

TITRE DE LA THÈSE / TITLE OF THESIS

A. El Saddik

DIRECTEUR (DIRECTRICE) DE LA THÈSE / THESIS SUPERVISOR

CO-DIRECTEUR (CO-DIRECTRICE) DE LA THÈSE / THESIS CO-SUPERVISOR

EXAMINATEURS (EXAMINATRICES) DE LA THÈSE / THESIS EXAMINERS

D. Petriu

J. Zhao

Gary W. Slater

LE DOYEN DE LA FACULTÉ DES ÉTUDES SUPÉRIEURES ET POSTDOCTORALES /
DEAN OF THE FACULTY OF GRADUATE AND POSTDOCORAL STUDIES

Design and Development of P2P Multilingual Collaborative Multimedia Tool

by

Md. Abdur Rahman

A Master's thesis submitted to the
Faculty of Graduate and Postdoctoral Studies
in partial fulfillment of the requirements for the degree of

Master of Applied Science
in
Electrical and Computer Engineering

Ottawa-Carleton Institute for Electrical and Computer Engineering
School of Information Technology and Engineering
University of Ottawa

Your file Votre référence
ISBN: 0-494-11388-X
Our file Notre référence
ISBN: 0-494-11388-X

Abstract

With the technological advancements, both client-server-based and fully decentralized collaborations are gaining popularity. Throughout the last decade, many collaborative tools have emerged based on the former technology or in some cases a combination of both technologies, which is often called hybrid collaboration system. The need for fully decentralized or Peer-to-Peer (P2P) collaboration over the Internet is increasing nowadays. Moreover, due to the rapid growth of multimedia technology over the web, more and more P2P Environments are adopting numerous multimedia applications such as audio, video, text, etc. to enhance the quality of collaboration. However, there are many factors that make this task challenging. These factors include not only the limited bandwidth (especially of peers having dialup connections), the presence of firewalls, proxy servers and Network Address Translation (NAT's) in Intranets, but also how to efficiently design such a fully decentralized system that supports different multimedia. A flexible architecture needs to be defined that overcomes these problems and permits a smooth and rich collaborative multimedia environment that would satisfy the needs of end users. This thesis illustrates the architecture, design and implementation of a fully decentralized collaborative multimedia system called PECOLE (PEer-to-peer COLlaborative Environment). PECOLE supports real-time collaboration and provides the following collaborative facilities: 1) Shared browsing 2) Multilingual collaboration 3) Shared telepointer 4) Moderation 5) Multipoint-to-multipoint audio/video conferencing and 6) Chat.

Acknowledgements

My sincere gratitude goes to my academic supervisor Dr. Abdulmotaleb El Saddik, who helped, encouraged, and guided me towards my academic as well as my personal success. The thesis would not be successful without enormous help from him. He provided me a substantial amount of help and suggestions throughout my work. A special gratitude also goes to fellow students of the Multimedia Communication Research Laboratory team specially Souhail Abdala, Bogodan Solomon and Md. Anwar Hossain for many helpful discussions and their constant encouragement and support.

I would like to give my sincere thanks to my beloved parents and wife for their support throughout my graduate studies. I am also grateful to all my friends and family who helped me either with suggestions or with positive affirmation.

I would like to acknowledge the financial assistance of the National Capital Institute of Telecommunications (NCIT) and of the Natural Sciences and Engineering Research Council Canada (NSERC) through its Learning Objects Repositories Network (LORNET).

Table of Contents

List of Figures

List of Abbreviations

ADSL: Asymmetric Digital Subscriber Line

API: Application Programming Interface

AWT: Abstract Window Toolkit

DHCP: Dynamic Host Control Protocol

GUI: Graphical User Interface

HTML: Hyper Text Markup Language

HTTP: Hyper Text Transfer Protocol

HTTPS: Secure Hyper Text Transfer Protocol

IE: Internet Explorer

I/O: Input/Output

IP: Internet Protocol

ISP: Internet Service Provider

JASMINE: Java Application Sharing in Multi-user INteractive Environment

JDK: Java Development Kit

JETS: Java Enabled Telecollaboration System

JMF: Java Media Framework

JNLP: Java Network Launching Protocol

JVM: Java Virtual Machine

JXTA: JuXTApose

LAN: Local Area Network

MD5: Message-Digest algorithm 5

MPEG-4: Moving Picture Experts Group

NAT: Network Address Translation

NTP: Network Time Protocol

OS: Operating System

P2P: Peer-to-Peer

PECOLE: Peer to pEer COLlaborative Environment

QoS: Quality of Service

SDK: Standard Development Kit

SWB: Shared Web Browser

SWT: Standard Widget Toolkit

TCP: Transmission Control Protocol

TOMSCOP: Technology Of Multi-user Synchronous COllaboration Platform

TTL: Time-To-Live

UDP: User Datagram Protocol

UI: User Interface

URI: Uniform Resource Identifier

URL: Uniform Resource Locator

XML: eXtensible Markup Language

Chapter 1 Introduction

1.1 Thesis Background and Motivation

Collaborative environments through networks have become a very popular research area for many years [25] [37]. The growth of the Internet makes the world smaller and more and more people would like to work together from places that are geographically distributed. Several applications demand this kind of distributed collaborative environments. Some of such applications are group chat, shared browsing [45], telecollaboration [23], multilingual collaboration, telepointer [2], shared whiteboard [10] [24] [31] [32] and instant messaging etc. In order to achieve such functionalities, a shared workspace similar to a physical meeting room in the real world is required [26]. The shared workspace can provide an identical visual and operable working area among geographically separated participants [42]. It is one of the most important features of synchronous collaboration systems [15]. There are two basic approaches to implement such shared workspace. The first one is based on client-server model which is most commonplace nowadays and lots of tools have emerged in the last decade that adopts this model. Another approach is based on fully decentralized architecture such as a P2P network. The greatest drawback of the former approach is that it requires huge and complex server to maintain groups, communication, security etc. resulting high costs, bottleneck around the server and maintenance overhead. On the other hand, the latter approach has attracted lot of researchers because it does not need any central server and basically no maintenance is required. Several tools have also emerged in the market that works on a P2P network and provide collaborative facilities. One of the shortcomings of pure P2P network is that developers of P2P applications had to develop their own network protocol from scratch. JXTA [40], an open source, language-neutral, system

independent platform for P2P, has solved the shortcomings of P2P computing, reducing the time-to-solution for P2P applications. JXTA makes P2P interoperable by defining a basic communications substrate in XML. As such, it crosses language boundaries. It is also network independent, and can operate over TCP/IP, Bluetooth and smart phones. JXTA also can route packets across NAT, DHCP (Dynamic Host Control Protocol), and firewall boundaries.

Several collaborative applications have emerged recently based on JXTA framework due to its enhanced features and emerging technologies. A widely used example is the chat room. People in different places can chat with others synchronously through the Internet. There are also many frameworks to support group work on a shared workspace. For example, the whiteboard [10] is a commonly used synchronous collaborative tool that allows users to work collaboratively over the network. Users can share files on the whiteboard and draw and annotate on it as well.

However, there are still lots of problems and drawbacks with existing products, prototypes and frameworks. Some of these systems provide only simple tools, such as chat rooms, whiteboards, shared browsing etc. For instance, users cannot share Java programs transparently in the shared workspace in the TOMSCOP [15], NetMeeting [9] [31] and Groove [38]. None of them provide multipoint-to-multipoint audio/video conferencing based on pure P2P network. Multilingual collaboration is another feature which is not present in any collaboration tool till to date as far as we know. The session discovery, joining and leaving the session in distributed P2P networks is another challenge. Most existing products that support session management, such as JASMINE [2], JETS [23], Microsoft NetMeeting [9] [31], and GROOVE [38] heavily depend on central servers to handle session management. Although TOMSCOP [15] uses a pure P2P based session discovery protocol, it takes unexpectedly long time to discover and join the session.

The thesis is motivated by the facts that these problems have limited the promotion of real-time multimedia collaboration, so it is necessary to find flexible solutions for them.

Among various characteristics, we choose the following features as objective of our design and implementation: multipoint-to-multipoint audio/video conferencing, multilingual collaboration, telepointer, shared browsing, moderation, and chat.

1.2 Thesis Objective and Contribution

The objective of this thesis is to enhance and optimize typical synchronous collaborative environments with several new functionalities leveraging cutting edge technologies and make them as complete and generic as possible. The challenge in the development of several useful collaboration functionalities for real-time multimedia collaboration environment including multipoint-to-multipoint audio/video conferencing in pure P2P network that overcomes firewalls and NATs and a multilingual collaboration tool is considered as one of the main objectives of the proposed work. Another objective of the thesis was to implement a Standard Widget Toolkit (SWT) [52] based shared browser which enables the sharing of both java application [21] as well as web documents of virtually most formats for synchronous collaboration systems. The final objective was to incorporate as many features as possible to make the multimedia collaboration appealing such as the introduction of moderation capabilities, chat, and shared telepointer over P2P network with reduced complexity.

The main contribution of this thesis is the design and development of PECOLE, a P2P-based multimedia collaborative environment. PECOLE is based on JXTA framework and it provides, in addition to the standard features such as chat and multi session, some novel collaborative features such as multilingual shared browsing, multipoint-to-multipoint audio/video conferencing, shared telepointer across several applications and session moderation capability for floor control.

Particularly the two new features: ***multilingual shared browsing*** and ***multipoint-to-multipoint audio/video conferencing*** are designed and developed on fully decentralized

architecture. The specialty of the audio/video conferencing system module is that it can overcome firewalls and NATs.

Although Shared Browser is being implemented by some researchers and some collaborative tools, the design architecture we followed gives the browser a unique perspective. We used SWT technology, which gives the browser the power to share web documents with theoretically all the scripts, Java applications and applets, 3D virtual world etc. Our SWT-base browser is capable of using the properties of any underlying Operating System (OS), thus making use of already installed Internet Explorer or Mozilla Firefox capabilities in a transparent way.

Finally, our novel peer discovery mechanism, implemented in order to find appropriate peers to join a group or session, is designed in such a way that it needs significantly less time than existing peer discovery techniques proposed by other researchers.

1.3 Thesis Organization

In chapter 2, background of the thesis and related work about the development of multimedia collaboration system is introduced. In chapter 3, the introduced PECOLE system design is presented in the order of system architecture, and software design. Chapter 4 provides the implementation details of PECOLE while chapter 5 provides the measurement data we have found along with comparison of features with other tools mentioned in chapter 2. In chapter 6, a conclusion is given and some future work is mentioned.

1.4 Publications

Two journal papers have been submitted.

1. Abdulmotaleb El Saddik, **Md. Abdur Rahman**, Souhail Abdala and Bogdan Solomon, "PECOLE: P2P Multimedia Collaborative Environment", ACM Transactions on Multimedia Computing, Communications, and Applications (submitted).

2. Md. **Abdur Rahman** and Abdulmotaleb ElSaddik, "Modified Syntactic Method to Recognize Bengali Handwritten Characters", IEEE Transactions on Instrumentation and Measurement (submitted).

Five papers have been published in conferences.

1. **Md. Abdur Rahman**, M. Anwar Hossain and Abdulmotaleb El Saddik, "Authoring Multimedia Objects in Collaborative Ambient Intelligent Virtual Environment", In Proceedings of the 4th IEEE International Workshop on Haptic Virtual Environments and their Applications (HAVE2005), Ottawa, Canada, October 1-2, 2005.

2. **Md. Abdur Rahman,** Souhail Abdala and Abdulmotaleb El Saddik, "A Framework for Two Phase Reconciliation in Mobile Databases", In proceedings of the NOuvelles TEchnologies de la Rŝpartition (NOTERE 2005), Montreal, Canada, August 30-September 1, 2005.

3. M. Anwar Hossain, **Md. Abdur Rahman,** Abdulmotaleb El Saddik and Pierre Lévy, "Architecture for 3D Navigation and Authoring of Distributed Learning Object Repositories", In Proceedings of the 3rd IEEE International Workshop on Haptic Virtual Environments and their Applications (HAVE2004), Ottawa, Canada, October 2-3, 2004.

4. **Md. Abdur Rahman,** M. Anwar Hossain, Abdulmotaleb El Saddik, "LORNAV: A Demo of a Virtual Reality Tool for Navigation and Authoring of Learning Object Repositories", In proceedings of the 8th IEEE International Symposium on Distributed Simulation and Real Time Applications (DS-RT 2004), Budapest, Hungary, October 21-23, 2004.

5. Md. Shamim Hossain, **Md. Abdur Rahman**, and Abdulmotaleb El Saddik "A Framework for Repurposing Multimedia Content". In Proceedings of the Canadian Conference on Electrical and Computer Engineering, Niagara, Ontario, Canada, May 2-5, 2004.

Chapter 2 Background and Related work

2.1 Literature Background

In this section, background information of the thesis will be presented. The development of any collaboration system has to handle operation sequences on shared objects, so the first concept to introduce is the shared workspace that will be designed and developed in this thesis. Although shared workspace is an abstract entity, which can be applied to many domains wherever collaboration is concerned, we will be describing the shared workspace in view of fully decentralized P2P domain only. Secondly, as the basic framework of multimedia collaboration is based on JXTA, we will discuss the technical basics related to this technology.

2.1.1 Shared Workspace

The shared workspace is a peers' joint working area for viewing and manipulating objects collaboratively [42]. Shared workspace systems can include those that support chat rooms, shared whiteboards, video conferencing, telepointer etc. The main capabilities of such systems include:

- Allowing peers and users to discover and to be aware of each other;
- Allowing peers and users to communicate synchronously and asynchronously with each other;
- Allowing peers and users to be uniquely identified on the network;
- Ensuring a high Quality of Service (QoS) for communications;
- Providing security within the system; and

- Allowing data to be stored and managed on the peers

Communication in real-time is a crucial aspect for most decentralized shared workspace applications [43]. Figure 2.1 shows basic shared workspace architecture for a pure decentralized P2P network [30]. The Real Time Connection Monitor layer is responsible for acting upon information obtained from the Awareness Monitor, Repository Manager and Check in/out data layers to keep shared workspaces synchronized. Once any data or request has passed through the P2P Network Layer and is stripped down by the Message Resolver, it may make a decision to pass the data/request through the Real Time Connection Monitor layer in case there are time constraints attached to this kind of data/request (i.e. a global update of a workspace, video conferencing may be in use or streaming of some form of media). The Real Time Connection Monitor could then make a decision to attach a high priority to the data/request.

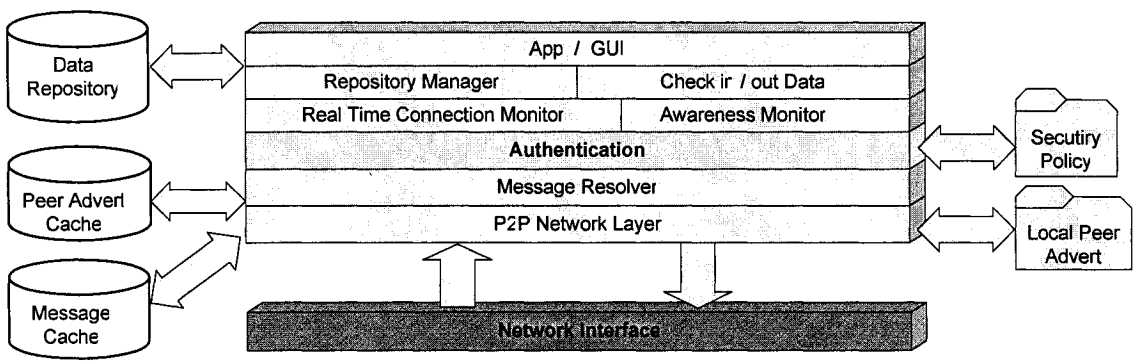


Figure 2.1 Decentralized shared workspace architecture [30]

Achieving a truly decentralized shared workspace system is particularly difficult due to the fact that the workspace itself needs to be managed. One possible solution is to allow any peer on the network to create and manage a shared workspace. This, in essence, means that each peer can effectively take on a role as a server within the network (handling authentication, QoS issues, etc). Architecture for such a peer would be similar not only to that of a server node, but also including the requisite mechanisms required for decentralized systems (e.g. publication, discovery and message routing). Thus, in order to provide a dynamic shared workspace, each peer/user in a decentralized

system must have both server and client utilities embedded in them [30]. JXTA comes with an architecture where each peer acts both as a server and as a client, making it a favorable framework for shared workspace design.

2.1.2 Introduction to JXTA

JXTA (JuXTApose) is an open-source project [40] that provides generic building blocks for the development of any type of P2P systems. JXTA creates a virtual overlay network on top of existing physical networks upon which services and applications are built (see Figure 2.2).

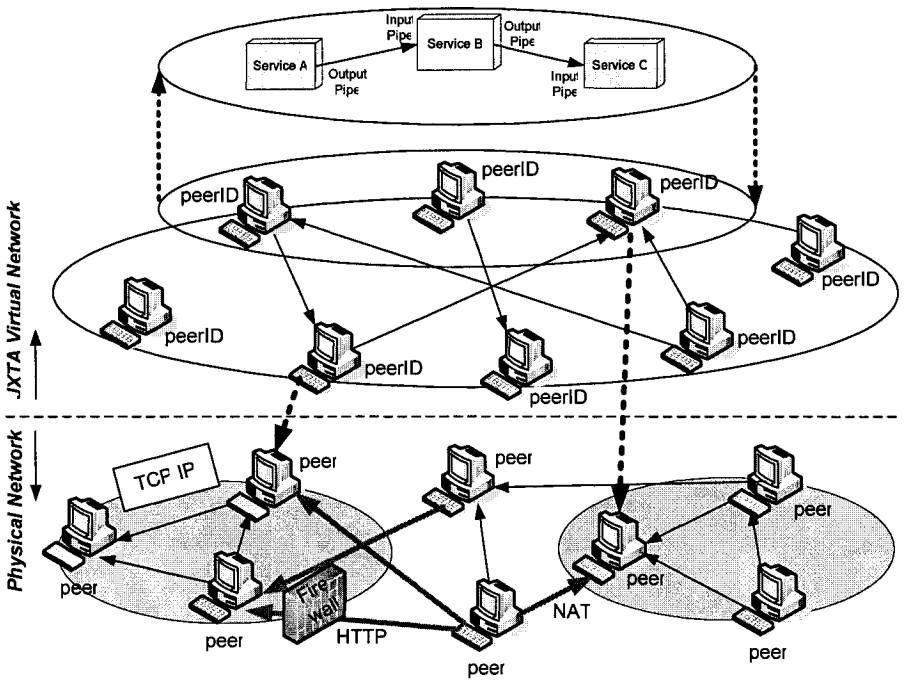


Figure 2.2 JXTA virtual network

A peer is any networked device (sensor, phone, PDA, PC, server, and so on) that implements one or more of the JXTA protocols [4]. A peer works independently and asynchronously of other peers. It can be both a client and a server. A peer group is a collection of peers that have a common set of interests.

The basic communication means used in JXTA are pipes [8]. A pipe is a virtual unidirectional connection between peers. A thorough understanding about different JXTA pipes can be found in [5] [8] and [40]. JXTA does not define or enforce any underlying transport protocol, such as TCP or UDP. JXTA relies on the features provided by the transport protocols and does not add any own functionalities which are already available by those protocols, such as routing or error detection and correction. All entities (resources), such as peers, peer groups, pipes, or services, are described by advertisements [5]. These are XML structured documents and follows programming language neutral metadata structure.

The JXTA network is an ad hoc, multi-hop, and adaptive network composed of connected peers. Connections in the network may be transient, and message routing between peers is nondeterministic. Peers may join or leave the network at any time, and routes may change frequently. The organization of the network is not mandated by the JXTA framework, but in practice four kinds of peers are typically used: a minimal edge peer, a full-featured peer, a rendezvous peer and a relay peer. Details about the functionality of different types of JXTA peers can be found in [5]. JXTA network is an adaptive network composed of connected peers as shown in figure 2.3.

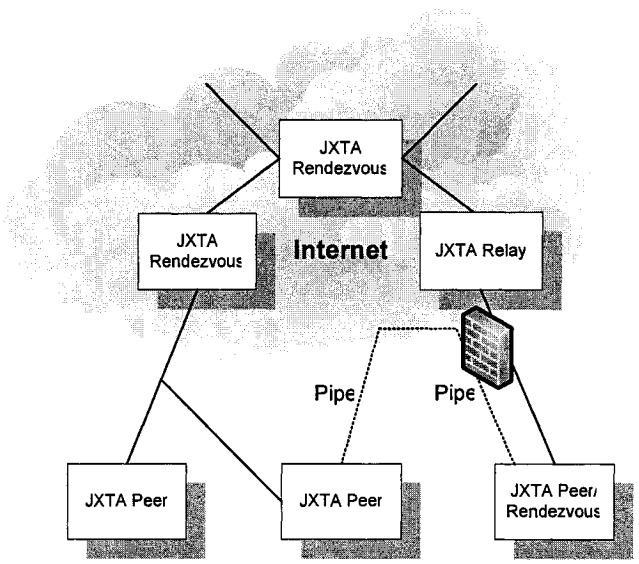


Figure 2.3 JXTA network

In order for JXTA peers to communicate with each other across a firewall [14], at least one peer in the peer group inside the firewall must be aware of at least one peer outside of the firewall which is typically a relay peer, and they can send or receive messages via HTTP as almost all firewalls allow HTTP data transfer, typically using port 80.

Another important feature of JXTA, towards the ubiquitous communication, is the JXTA Socket [51] which offers an Internet where everyone can create and consume network services, even if they have a dynamic IP address or no IP address, are behind a Network Address Translation (NAT) device, or blocked by an ISP's firewall. JXTA Socket hides the complexity of pipes by introducing an abstract layer on top of the pipes and provides an interface similar to that of the more familiar Berkeley Software Distribution (BSD) socket API. Compared to the JXTA pipes, JXTA sockets add reliability and bi-directionality to JXTA communications [4] [6]. The features embedded with JXTA socket is very much suitable for real-time multimedia communication especially audio, video, synchronous content sharing etc. Each peer can communicate with other peer through JXTA sockets and different dedicated sockets can host different multimedia channels like audio through one socket, video through another etc. (see Figure 2.4).

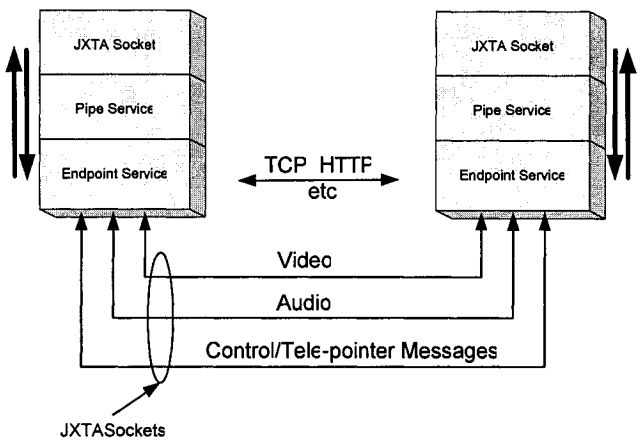


Figure 2.4 Communication based on JXTA sockets

2.2 Related Work

Collaborative computing, usually known as groupware or computer supported collaborative work (CSCW), refers to technologies and systems that support a group of people engaged in a common task or goal and that provide an interface to a shared environment. Grudin [34] in 1994 defined a time/location matrix to generally categorize collaborative systems as four types, among which there is one called a distributed synchronous collaborative system that can support a group of people in different locations to conduct a common task or activity at the same time. A necessary and fundamental element in a synchronous collaborative system is the shared application that multi-users can synchronously view and manipulate following the mode of what you see is what I see (WYSIWIS) [35]. Different degree of coupling in WYSIWIS system can be found in [44].

Shared applications fall into two categories, screen-copy system and event-aware system. The former allows many existing single-user applications to be used by multi-users in cooperative fashion via capturing an application window and sending it as image data similar to that of a video camera. Examples of transparent collaborative systems are Microsoft NetMeeting [9] [31], and Intel ProShare [41]. In the latter system, only events related to an application are captured and sent out. The event-aware systems make more efficient use of networks, and can support more advanced groupware functions. The designed system in this thesis falls into this category with a complete different connection topology and other special features. There are generally three types of connection and message passing topologies among multiple users' computers/devices used for their collaborations. One is called a centralized topology in which there is no direct connection between computers and all messages are mediated by an inter-mediator generally known as a group server. JASMINE [2], JETS [23], NetMeeting [31], IBM SameTime [32] and several others have adopted this topology. However, systems built on the above platforms suffer common problems that a communication bottleneck may arise since all messages must first go to and then get out from the centralized server, and the whole system may be down when the server has some trouble. Actually such connection topology follows

the ordinary client-server model. The P2P model is used for the other two topologies: the hybrid topology and the decentralized topology [3] [36] [37]. The hybrid topology is one in which a peer needs to connect to both a group server and other peers. In this connection, some group administration messages are passed via the server and other messages are sent directly to others. Groove [38] and Endeavors' Magi [39] have adopted the hybrid topology. Although they overcome some drawbacks of client-server based systems, a peer has yet to go first to the server and strictly follow the procedures defined by a particular system. Peers have not enough flexibility to quickly find each other and easily form a group by themselves. Furthermore, the two systems only work within Microsoft Windows environments. The decentralized topology is one in which every peer is able to directly connect to all other peers and messages are sent without intermediation via a server. Similar to TOMSCOP [15], we designed our collaborative environment based on the decentralized topology and both systems are able to work in any environment due to their Java implementation [21].

Although there is no border, adding more and more components in a decentralized multimedia collaborative environment makes any collaborative environment much more appealing to the users. Among some of appealing add-ons are shared browsing, moderation, multilingual collaboration, shared telepointer, and multipoint-to-multipoint audio/video conferencing. Because there are several collaboration tools that already exist, we will introduce some of the above works in details that closely relate to our requirements in terms of features or topology or architecture and will focus on what these environments/tools have done and the basic concepts and architectures behind their work. The differences between these works and our solutions, as well as the advantages of our solutions, will also be presented.

2.2.1 JASMINE

JASMINE (Java Application Sharing in Multi-user INteractive Environment) [2], developed jointly by the University of Ottawa and the Darmstadt University of Technology, enables users to share Java applications or applets in real-time. The system

also provides basic moderation control and enables diverse views of the same visualization in a moderated session, in which the moderator can see more than others [1]. The idea behind JASMINE is that all the events happening at the GUI of an application or applet on the client side can be caught, distributed and reconstructed. As a result, Java applications or applets can be shared transparently. This type of collaboration allows users to interact in real-time [1].

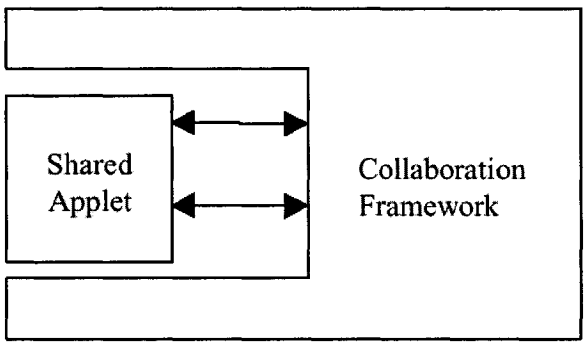


Figure 2.5 The concept of JASMINE

Figure 2.5 shows the concept of JASMINE. It wraps around the application to be shared. The framework listens to all the events occurring on the GUI of the application and distributes these events to all other participants and then remote users reconstruct the events at their own environment. The framework is based on Java Abstract Window Toolkit (AWT) and Swing classes. JASMINE can share Java applets and applications synchronously. However, JASMINE framework has several drawbacks. First of all, JASMINE uses a client-server model for the collaborative session management like group management, event sharing, floor control, moderation control etc. When the number of users increases, the system performance decreases [2]. Secondly, Applet or Swing based system on top of TCP or UDP shows poor performance in comparison with that of SWT based system integrated with JNLP on top of Java Web Start [49]. Third, it does not have several features like multilingual collaboration, telepointer and multipoint-to-multipoint audio/video conferencing.

2.2.2 JETS

JETS (Java Enabled Telecollaboration System), developed at the University of Ottawa, is a client-server based system that permits sharing of Java applets. Because of security restrictions, applets are allowed to establish network connections only on the machine from which they are downloaded. This issue forces JETS to use a central server, so applet clients can exchange data and multimedia documents through the server. For the client-server communication, JETS uses TCP/IP and UDP/IP sockets. Scalability becomes an issue since too many clients would overwhelm the server, so JETS has been designed in such a way that scalability becomes a hardware issue. Furthermore, JETS makes it possible to use more than one computer to perform duties of the server. Figure 2.6 below shows how JETS works [23].

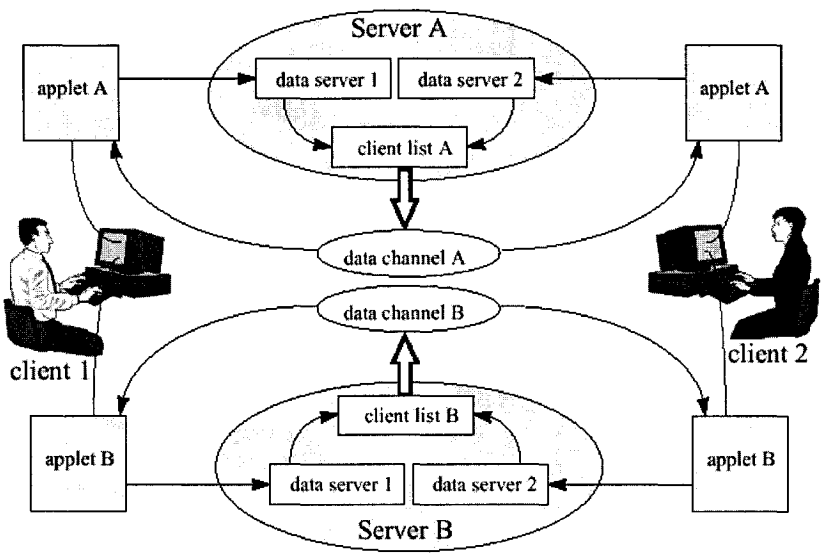


Figure 2.6 The architecture of JETS

JETS is a groupware toolkit, written fully in Java that offers an API which a developer can use to create multimedia applets for collaboration [23]. The whiteboard provided by JETS is an interactive space where clients in a virtual session can share pictures, Microsoft PowerPoint slides, text, video and drawings. Users can annotate on images and start a discussion. The built-in locking mechanism of JETS is used to avoid modification of the same objects at the same time by more than one user. Clients can use the chat area

to communicate. All other members who have input access to the whiteboard will see the originator of the message followed by the message itself. Another way of interacting through the whiteboard is by drawing. A client may paste a picture found in the archive. Any member who has full access can freely comment or draw on the picture. A client may also start a slide show found in the archive. Any member who has full access can freely comment or draw on the slide show as well as go to the next or previous slide [23].

A very useful feature of JETS is its ability to play ITU-T H.263 compliant video on the whiteboard. That is accomplished by using jStreaming's API. When a user opens a video file and starts playing it, video data are streamed down to all participants, decoded in real-time, and displayed on their whiteboards [24].

JETS also implements a management system that enables monitoring of the session. One of the session clients has to log in as a moderator. Once the session is established, initially only the moderator has the right to access the shared whiteboard. Other clients have no access. Every session client can ask the moderator for access permission. Once the moderator approves a session client's request, this client becomes a session participant, and gains the right to access the shared whiteboard. Any session participant can put his/her notation on the shared whiteboard. The moderator has the right to revoke access privileges to any client at any time [24].

However, JETS has several limitations in comparison to our proposed system. First of all, JETS uses a centralized server for the collaborative session management like group management, event sharing, floor control, moderation control etc. When the number of users increases, the system performance decreases. Secondly, Applets when shared over network on top of TCP or UDP, shows poor performance in comparison with that of SWT based system integrated with JNLP on top of Java Web Start [49]. Finally, it does not have many features like multilingual collaboration, telepointer and multipoint-to-multipoint audio/video conferencing.

2.2.3 Microsoft NetMeeting

NetMeeting is a powerful tool that allows real-time communication and collaboration over the Internet or corporate Intranets. Users can communicate over a network with real-time voice and video technology. Users can work together virtually on any window-based program, exchange or mark up graphics on an electronic whiteboard, transfer files, use the text-based chat program or share any program or remote desktop. NetMeeting supports multiple sessions based on a centralized server, but in a session, only two users can have an audio and video conversion at a moment. It does not support multipoint-to-multipoint audio/video conferencing.

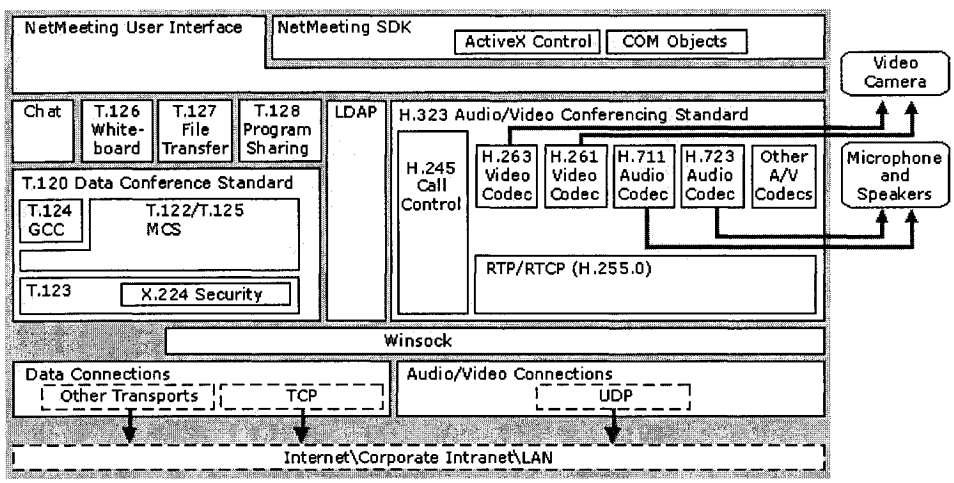


Figure 2.7 The architecture of Microsoft NetMeeting

NetMeeting functions both as a client and a platform. The NetMeeting client provides users the benefits of real-time audio, video and multipoint data conferencing. At the core of the NetMeeting architecture is a series of data, audio and video conferencing and directory service standards. Figure 2.7 shows how these standards work together with transport applications, user interfaces, and the Window NetMeeting Software Development Kit (SDK) components to form the NetMeeting architecture [31].

NetMeeting is a very popular and it supports multiple sessions based on heavy-duty centralized servers. However, it is impossible to remove the centralized server from the

system, so NetMeeting architecture is not suitable for P2P network. It also does not have telepointer and multilingual collaboration facilities. Another biggest disadvantage of NetMeeting is that for the audio/video conferencing one need to disable all the firewall or need to open the communication ports for NetMeeting which is very much vulnerable for any corporate computer.

2.2.4 Lotus SameTime

Lotus SameTime represent IBM major mainstream effort to enter the web conferencing, live presentation and virtual classroom niche with a competitive product [32]. The collaboration services, as shown in Figure 2.8, provided by SameTime can fall into the following categories:

Conferencing services: These services include a shared whiteboard; IP based audio and video conferencing; and the ability to share programs and documents online.

Secure instant messaging services: These services include team awareness, instant messaging, and chat. A "presence list" makes users aware of who is available (and who is online but unavailable) to receive an instant message or participate in a chat".

Shared whiteboard: The shared whiteboard supports interactive presentations which closely resemble a slide show. In a whiteboard presentation, one participant presents images in the white-board tool of the SameTime Meeting Room client on the participant's local computer. Remote meeting participants can also view the images and annotate the images using the whiteboard tools running on their local computers.

IP audio and video: IP audio and video enables multiple users to transmit and receive it over an IP network in a SameTime meeting. It is either "interactive" or "broadcast". The "interactive" IP audio and video enables all participants in a meeting to both transmit and receive IP audio and video packets on the network. In an interactive IP audio and video meeting, one user transmits a stream of audio and video packets to the server and the

server disseminates this stream to all other meeting participants. The "broadcast" IP audio and video enables a large group of users (or audience members) to receive the audio and video from a meeting but not transmit audio and video to other users in a meeting.

IBM also hosts a live demo area which allows to fully testing all the features of SameTime, while allowing emulating a live web conferencing session that can be found at [33]. Although SameTime offers several collaboration facilities, it requires huge expenditure and operational costs, maintenance overhead, complex configuration and above all, it is proprietary. In order to get the collaboration services, clients need to connect to its central WebSphere Edge Server and 10 multiplexor (MUX) servers that

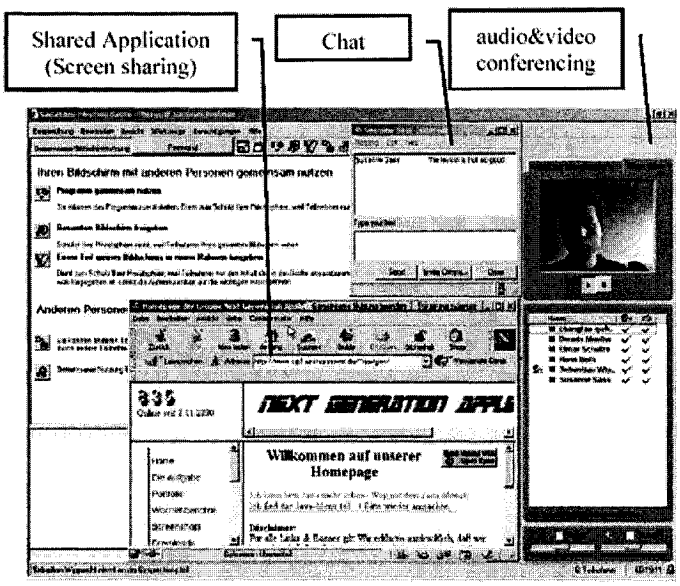


Figure 2.8 Screen shot of part of synchronous collaboration services offered by SameTime

disperse chat sessions between four Community servers. These all lead to the same server bottleneck issue. Above all, SameTime is not based on P2P architecture, does not provide any multilingual collaboration facilities and lacks shared telepointer option.

2.2.5 Groove

Groove provides a virtual space for a small group of people with a context as rich as the web in an environment as spontaneous as e-mail. The goal of Groove is to provide a virtual workspace for users to share their ideas and information [38]. Figure 2.9 illustrates the virtual organization of Groove. The Groove Network consists of three components:

- Pure P2P network, which is the place where the peers can do group meeting, scheduling and having discussion;
- Groove agents, which do the roles of backing up peer information, querying and updating peer requests and event monitoring; and
- Center-based server bots, which does save customer history, transaction processing, and workflow/business process automation

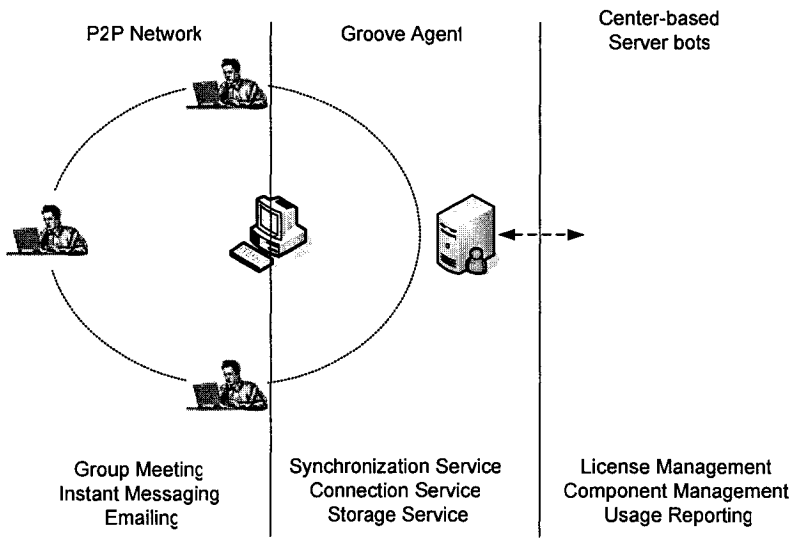


Figure 2.9 Groove network virtual organization

Groove works by sending messages through the Internet, telling other clients about changes that have been made by one member of a workspace to documents or other information being shared. Once the other clients receive these messages, they can update their local versions of the documents or other information.

Groove is an intermediate platform between P2P network and client server architecture. Groove architecture integrates P2P and client server by putting agents and servers in the middle of peer networks. Groove tries to support IT managers to control the peer network maintenance and save peers history and security when a peer is offline. In the other hand, Groove has the disadvantage of both architectures. Groove is still having problems inherited from client server architecture such as high server maintenance cost, fixed role by separation of client and server, exponential traffic increase as users increase, heavy traffic to the agents and servers, and lack of scalability.

For these reasons, Groove has to admit that it is suitable only for small group of people, not for large-scale communities. GROOVE, a virtual workspace, does not have most of the collaborative features like multipoint-to-multipoint audio/video conferencing, multilingual collaboration and shared telepointer.

2.2.6 TOMSCOP

TOMSCOP (Technology Of Multi-user Synchronous COllaboration Platform) is based on the elementary peer group services offered by the JXTA general framework. TOMSCOP provides four types of services: synchronous message transportation, peer room administration, peer communication support and application space management (see Figure 2.10).

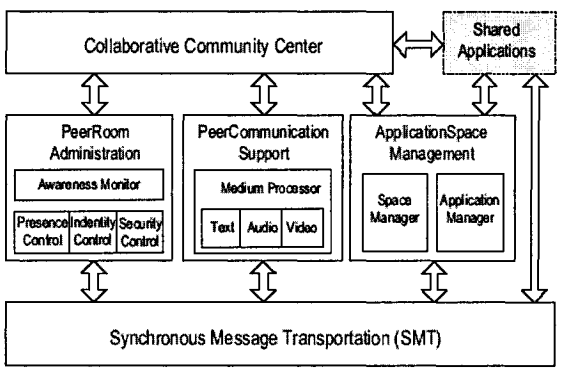


Figure 2.10 TOMSCOP architecture

By using the four services, different kinds of shared applications for various specific purposes can be relatively easily developed and associated collaborative cyber spaces or communities can be quickly built across the JXTA virtual network overlaid on top of the existing physical networks.

TOMSCOP is designed using the metaphor of center-room-facility. Users or peers gather in a virtual community center to meet each other, enter some rooms corresponding to specific groups of interests, and work together via using available facilities, i.e., shared applications. However, TOMSCOP suffers from inefficient pipe advertisement problems. It takes long time, around 7 to 8 minutes to discover a peer. Also there are some features that are not available in TOMSOP, although demanded by a rich collaborative environment, such as multilingual collaboration, and multipoint-to-multipoint audio/video conferencing.

Chapter 3 System Design

In this chapter we will present the design and architecture of PECOLE from two different view points. First of all, we will illustrate the layered architecture of PECOLE. Here we will present different logical layers of PECOLE and the applications running in each layer. Next we will present PECOLE software design technique with the aid of UML diagrams including package diagram, use case diagram, class diagrams and sequence diagrams.

3.1 PECOLE Architecture

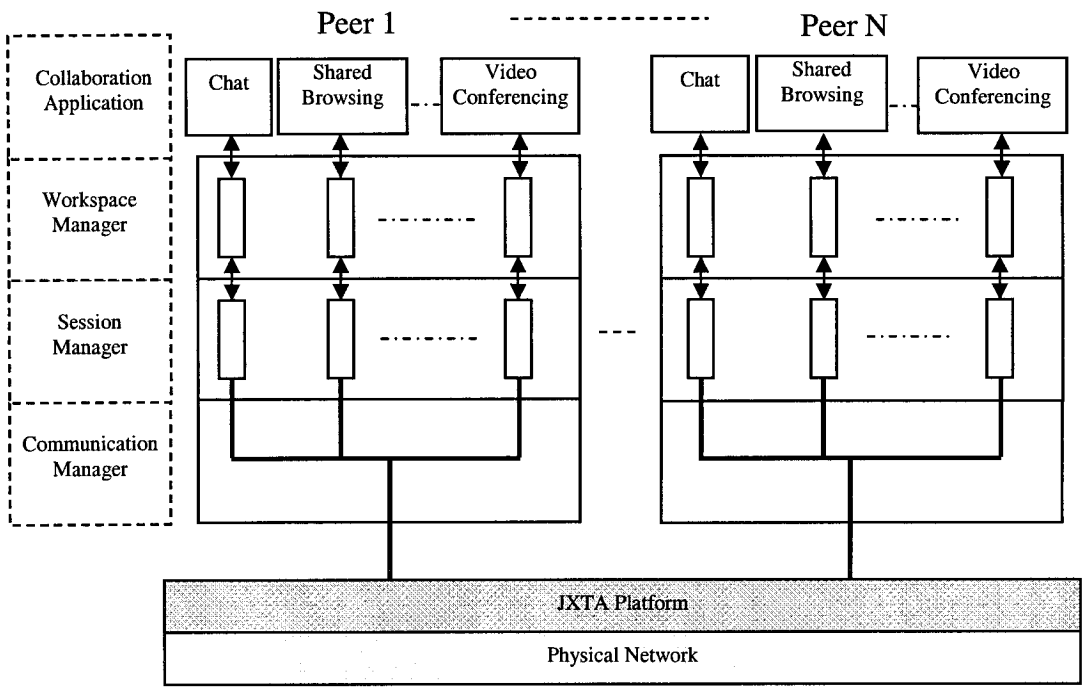


Figure 3.1 PECOLE system architecture

Figure 3.1 depicts our proposed PECOLE system architecture. Every PECOLE peer has four logical layers: (1) Collaborative Application (CA), (2) Workspace Manager (WM), (3) Session Manager (SM), and (4) Communication Manager (CM). All theses four layers reside on top of the JXTA platform, which hides the physical network underneath it. We will next describe the services offered by each layer.

3.1.1 Collaborative Application (CA) Layer

The CA layer hosts the PECOLE collaborative applications and any application-specific protocols. Events generated at the CA layer are of two types: (1) local and (2) remote. Local events are handled locally by the collaborative applications. Local events are directly sent to the local application layer for reconstructing the local GUI. However, remote events are passed to the lower layers for transmission to the remote peers. This happens in the case of a collaborative session where session chair wants to send the same URL's and UI's he is navigating. Remote events encompass all such events and messages that should be sent to remote peers of the same session to keep them synchronized. Among the applications of the CA layer are shared browsing, multilingual collaboration, telepointer, moderation, multipoint-to-multipoint audio/video conferencing, and chat. In the following section we will be discussing them in details.

3.1.1.1 Shared Browser

Most of the reviewed shared applications including [2] [17] [18] [19] [20] [22] [23] and [24] use the client-server model in which the server is usually very complex and heavy, and sometimes becomes a communication bottleneck. This is due to the fact that all the data exchanged among the group members are mediated by the server. To solve the above problem, the shared browser in PECOLE adopts a pure P2P architecture, which does not require any server. A group member or a device – also called a peer – dynamically finds other peers via distributed searching, and directly exchange data with other peers. It supports not only sharing a web document in a peer group but also synchronously viewing the document and manipulating the browser with further support

of some group users' awareness information like a moderator peer is moving a cursor, entering a new URL and clicking a hyperlink. It is implemented using JXTA protocol and Sun's SWT technology. To make the system applicable over the Internet bypassing firewalls and NATs, the HTTP protocol is used to transfer data via pipes, a communication mechanism in JXTA. Because of platform and transport independent features of JXTA as well as our system implementation with the JAVA, the SWT based shared browser can be used in any machine and over any network environment. Furthermore, JXTA provides flexible, scalable and secure group management.

Figure 3.2 shows the architecture of PECOLE shared web browser. The shared browser consists of seven standard modules, adopted by several P2P based systems including [15] and [16], and is briefly explained below:

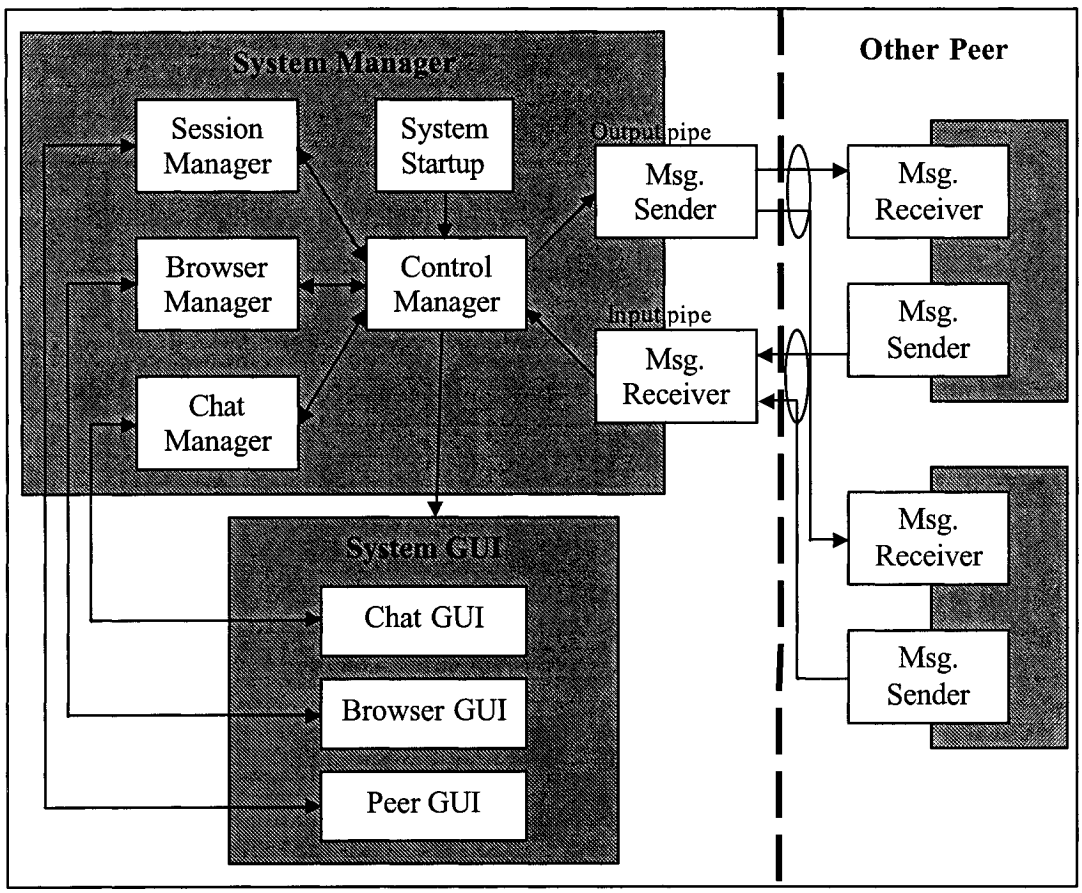


Figure 3.2 P2P shared browser system

- *System Startup* – It manages network and security configuration and connection with the JXTA generic net group. It also captures the session and moderation information like session name, moderator password from the user as initial input;
- *Session Manager* - It deals with session advertisement, discovery, joining and leaving;
- *Chat Manager* - It controls sending and receiving chat messages;
- *Browser Manager* - It deals with basic browser's functions like hyperlink, and browser's operations such as sharing a telepointer;
- *Control Manager* - It controls the Browser Manager, the Chat Manager, the Session Manager, the GUI components, and the input and output pipes. This unit co-ordinates various modules and makes sure to call responsible modules for any state of the system. If any unit wants to communicate with other unit, they must go through *Control Manager;*
- *System GUI* - It includes Chat GUI, Browser GUI, and Peer GUI; and
- *Msg. Sender and Msg. Receiver* - They are implemented using JXTA propagate pipes [8]. As shown in Figure 3.2, the propagate pipes connect one output pipe to multiple input pipes. Messages flow from the output pipe (the propagation source) into the input pipes in the same peer group. MsgSender acts as an output pipe and MsgReceiver acts as an input pipe.

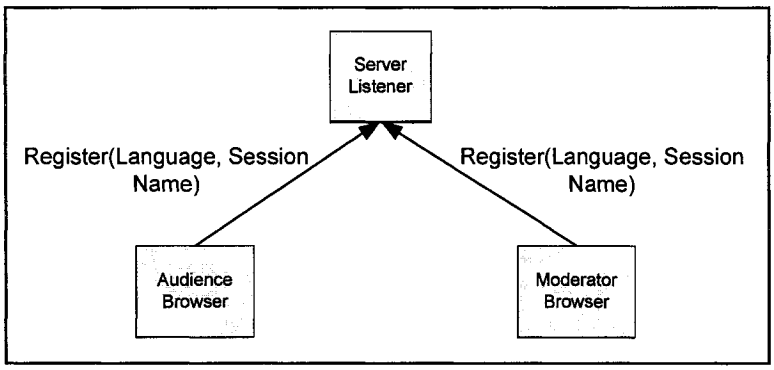
The browser exchanges or shares URL addresses with other peers through the JXTA propagate pipe. The session manager gets a URL address from a URL field, hyperlink or navigation button, and passes them to Msg.Sender. In the Msg.Sender, the URL address is converted to a pipe message. Conversely, when Msg.Receiver receives the message, the current browsing URL address is extracted from this message. Then, it is passed to the browser manager to access the corresponding web server that finally passes the URL to Browser GUI.

3.1.1.2 Moderation

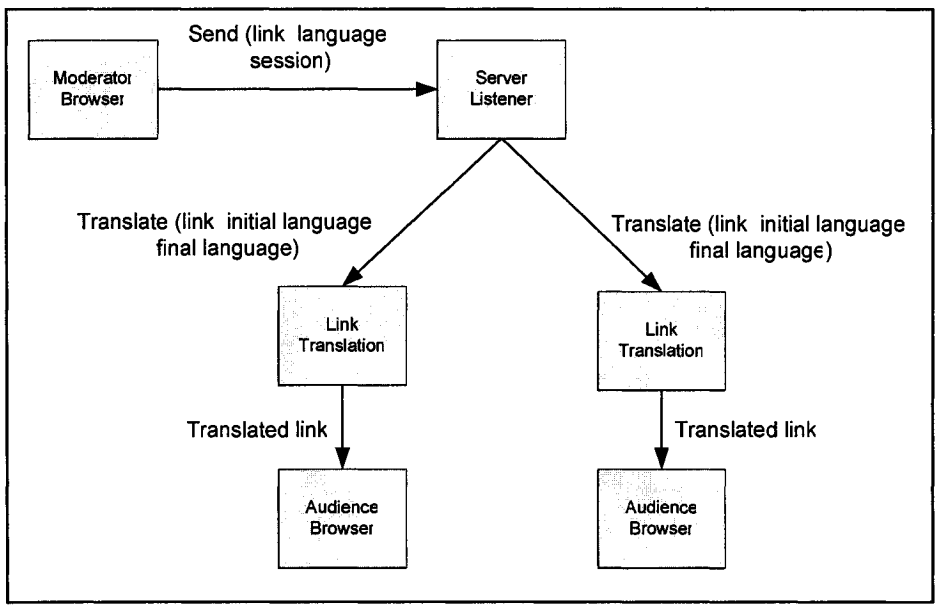
Similar to JASMINE [2] and JETS [23], PECOLE comes with a standard moderation system that facilitates the online peers to use the moderator/audience role while they are taking part in illustration of a shared document. This moderator role can be thought of as a token in a token ring network in which the node/peer having the token can only send the document to others. Likewise, the moderator has only the privilege to navigate the browser content while the audience peers only observe the same content [2] [15]. In PECOLE, we have two types of sessions: moderated, like telepointer, shared browsing, multilingual collaboration etc. and non-moderated, like chat and audio/video conferencing. Each session falls of any one of the categories. If it is moderated then, while entering the session, any peer with the moderator privilege, sometimes called session chair, has to log in the session. While the session is going on, moderatorship can be requested by any peer of the session, provided the current moderator releases the lock. Above all, the session moderator/chair has the super privilege to discard or take away this temporary privilege that was assigned to other peers in any time if he needs to do so. In summary, we have 3 types of peers in each moderated session. First one is the session chair, who has the super privilege of any session. The second one is the moderator, who has the floor control privilege when any peer acquires moderatorship. The third type of peer of any session, excluding the current moderator, is the audience peer.

3.1.1.3 Multilingual Collaboration

In multilingual browsing architecture (see Figure 3.3), the basic language translation service is provided by a translation server in which all the peers need to register initially (see Figure 3.3 (a)). All the subsequent messages exchanged among the peers pass through this server. The server then changes the web content, if necessary, depending on each client's language preference and forwards the content to the client (see Figure 3.3 (b)).



(a)



(b)

Figure 3.3 Multilingual collaboration architecture (a) Registration (b) Translation service

First of all, the system saves the peer's chosen language in the web server's session for that peer. The multilingual browser is a java based browser that starts through Java Web Start (JWS) from a web page. When the peer starts the java browser, the browser receives the user's language, role and session from which the browser was started, the peer's nickname, the translation server's IP and the starting webpage for the browser through the JNLP protocol [49]. When a peer having moderator privilege browses a URL, a message is sent to the server specifying the session name, moderator's language and the actual link that was clicked. The server receives the message, parses it and then verifies the list of audiences who are in the same session. Depending on the peers'

language preferences, the server translates the web content based on peers' preferences and sends the page to all the peers' browsers with everyone's chosen language.

3.1.1.4 Telepointer

Telepointers are replicated cursors that track the location and interactive movement of a peer's mouse pointer and replay it in all the peers attending the session of a collaborative application. Telepointers are one of the most useful elements of real-time groupware: they are simple to implement, but provide embodiment, awareness, and gestural communication. However, telepointers often suffer from severe performance problems on real-world networks like the Internet. When the network becomes congested, telepointers become slow, often to the point where they are no longer useful for the collaboration. In situations where people use telepointers to coordinate closely-coupled interactions, these incorrect representations of the other person's actions can lead to frustration and errors in the collaborative activity. Disruptions to these qualities are caused by network latency, jitter, and loss of packets – all of which happen frequently, even on high-bandwidth networks. As a result, real-time telepointers are virtually unusable in groupware that operates on real-world wide area networks, and most common groupware applications do not even attempt to provide them. For example, NetMeeting [31], Groove [38], and nearly all multi-player games provide no telepointers at all; some whiteboard systems such as MSN Messenger [46] provide a single draggable arrow. Some screen-sharing systems such as VNC [47] or Citrix GoToMeeting [48] that do provide a shared real-time cursor suffer obvious performance problems when network difficulties arise. Even though telepointers are not currently successful, it is not the case that they are fundamentally unsuited to Internet groupware. The root of the problem is that little attention has been paid to telepointer performance, and as a result, current telepointer implementations are inefficient and slow. Although performance issues have been considered by a few CSCW researchers, there are very few currently available implementations in P2P network. We propose a new telepointer implementation strategy that greatly improves telepointer performance with the aid of SWT technology.

The telepointer function provides two modes, moderator mode and audience mode as described in Table I, to control whose cursor will be displayed in the shared browser among all group members.

Table I Telepointer mode

Mode	Telepointer Action
Moderator mode	Move mouse pointer and send captured pointer position
Audience mode	Receive pointer position

3.1.1.5 Multipoint-to-multipoint Audio/Video Conferencing

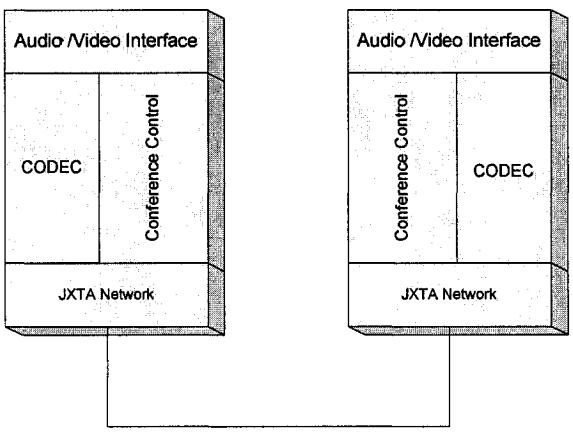


Figure 3.4 Basic audio/video conferencing architecture

Although a few video applications are developed over JXTA overlay network including [12], multipoint-to-multipoint audio/video conferencing is none at all. It is due to the fact that the service that can be provided is limited by number of ports and protocols because of the presence of firewalls and NATs. Four main technical components are necessary to provide basic set of conferencing services [7]. These components are: a user interface, efficient coder-decoders (CODEC) for audio and video, efficient networking support, and a set of conference control functions as shown in Figure

3.4. The logical relationship among these components defines the architecture of such conferencing system.

Figure 3.5 displays the architecture of PECOLE audio/video conferencing system. It is largely consistent with several architectures proposed in the literature [7] [28] and [31]. However, the audio/video conferencing system designed in this thesis is based on JXTA framework. As shown in Figure 2.4, PECOLE uses separate JXTA sockets for audio and video for scalability purposes. Streaming of audio and video is based on multicasting fully connected graph algorithm that ensures maintaining the bandwidth QoS. Before streaming audio/video, the peers form a fully connected graph structure using PECOLE's novel discovery protocol (see Figure 3.11 and Figure 3.12). PECOLE's GUI facilitates any peer to give access to his audio/video to any number of peers he likes.

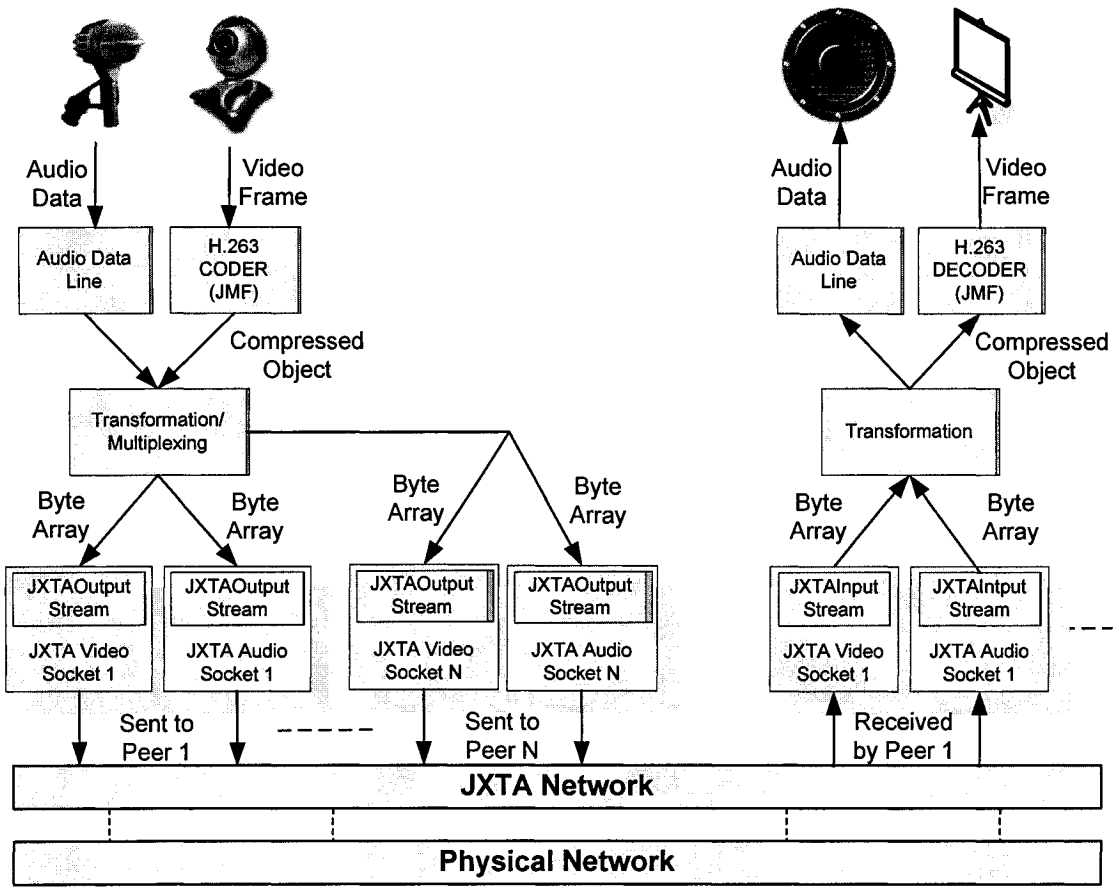


Figure 3.5 PECOLE multipoint-to-multipoint audio/video conferencing architecture

Video Conferencing

The video message is created in a three step process. First of all, the video frames grabbed by, e.g., a web camera using Java Media Framework (JMF) utility libraries [29] are sent to an H.263 CODEC [29], which generates the compressed video objects. This compression technique is very important for the low bandwidth network. Secondly, the compressed videoFrame objects need to be converted to byte arrays in order to be able to send them through JXTAOutputStream.

Finally, this stream is passed through JXTA sockets which convert the byte array to output streams that eventually use the JXTA network for transportation. In order to send the video stream to N number of peers simultaneously, PECOLE uses N number of JXTA sockets to multiplex the video stream. At the receiver end, the dedicated JXTA socket receives the stream and generates byte array, which is in turn passed to the transformation engine to reconstruct the compressed video object. Before playing the video by the player plug in, the image panel receives the compressed video frames sent by the decoder after constructing the video frames.

Audio Conferencing

Figure 3.5 also shows the multipoint-to-multipoint audio conferencing architecture of PECOLE. As the figure portrays, the audio data is first captured by the audio data line from the audio capture device like microphone or web camera. Because audio data requires less bandwidth in comparison with that of video data, the audio data is left uncompressed. It is received and stored in the audio buffer, constructed as a special audio object, multiplexed for multipoint dispatching, and sent as byte arrays to JXTAOutputStream unit. The JXTA audio socket then streams the audio data to JXTA network. The following code snippet shows the audio acquisition format (in Java) used in PECOLE.

```
AudioFormat audioFormat = new AudioFormat
                          (AudioFormat.Encoding.PCM_UNSIGNED, 8000,
                          8, 1, 1, 8000, false);
```

Figure 3.6 PECOLE audio acquisition format

As shown in Figure 3.6 we used PCM audio format with sample rate 8000 Hz, 8 bits/sample, 1 byte in each frame, 1 frame/second, and are stored in little-endian style, which refers to an addressing scheme for byte programming adopted by Intel Corporation where additional bytes expands either left or right without affecting it's already stored bytes.

On the receiver end, the receiving peer's dedicated JXTA audio socket tunnels the stream into the JXTAInputStream which converts the stream to byte array. The byte arrays are then passed to the transform unit for the audio data conversion that is processed by the audio data line to supply the audio data to audio play device.

3.1.1.6 Chat

PECOLE adopts a standard P2P based chat style where peers can enter rooms/sessions and communicate with text messages [11]. There is no chat server at all, no central peer. All clients just run independently, listening for chat messages and sending them to each other. JXTA discovery provides the capability that makes it possible. When a client starts up, it creates a pipe for itself. However, instead of using JXTA discovery to find a server and send the pipe advertisement to it, the client just publishes the advertisement itself, to the peer group. When clients want to find each other they no longer need to ask a server, instead they simply use JXTA discovery to find the client directly. Everything becomes much simpler and the code becomes terse. Figure 3.7 shows the architecture of chat message communication. It shows that once the communication channel is formed among collaborating peers, chat messages are just multicasted to the group.

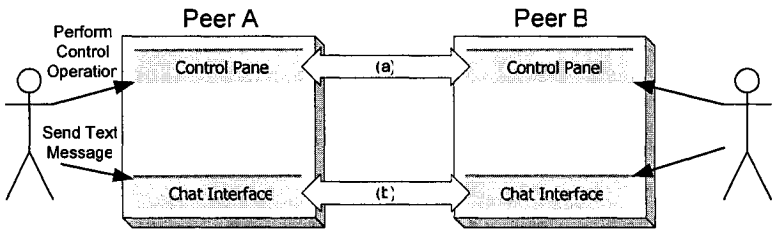


Figure 3.7 PECOLE chat architecture (a) Co-ordination control messages (b) Chat messages [11]

3.1.2 Workspace Manager (WM) Layer

A workspace is a collection of objects, like text, graphs, and images, that belong to some shared application and the tools necessary to access and manipulate these objects. When multiple users collaborate through jointly manipulating a shared object, the need for synchronization arises. Users can synchronize their workspaces by exchanging events. The WM layer is responsible for intercepting user events and passing them to the underlying SM layer which in turns broadcast them to all participants belonging to the same session. On the receiver side, the WM layer is responsible for receiving remote events from the SM layer and dispatching them to the appropriate shared applications.

3.1.3 Session Manager (SM) Layer

Users collaborating in a shared workspace are said to form a session. A session is a period of synchronous interaction. In every active session, one user is identified as the session chairman or moderator. The session chairman/moderator manages the session by invoking appropriate tools and admitting new participants to the session.

The SM layer is responsible for establishing, managing, and terminating sessions between peers. It is also responsible for floor and telepointer control. Further, it is responsible for maintaining a list of the current participants in every session. The list is used to broadcast session messages to all participants using the services of the underlying CM layer.

Our session management system works as follows. The initiating user (session chairman/moderator) starts a new session. Users who want to participate in the session must first join the session. There are two ways to find a session. First, a user can find a session by browsing a list of currently active sessions. Second, a user knows a priori that the session will be taking place. Once a user knows the session name, he can attempt to join the session.

The shared browser is intended to synchronously browse web document among multiple users in a group, thus each user must first discover and then join the appropriate session to share information with others. The Session Manager provides functions of session discovery, joining and leaving. In the following we briefly describe the actions associated with the joining and leaving of a session.

Join-Session Action

When a peer joins a session, all the peers in the session view him in the Peer GUI (see label 8 of Figure 4.1) and other peers in the same group can send related information like a current browsing URL, window size and scrollbar position to the new peer.

Leave-Session Action

After clicking the "Exit" button in a group GUI, first, a logout command is sent to other peers of a same session through the output pipe in the form of a popup window (see label 12 of Figure 4.1).

3.1.4 Communication Manager (CM) Layer

The CM layer provides the services necessary for message transportation between peers. It is implemented using JXTA sockets. The following are some of the services provided by the CM layer:

1. CM layer provides reliable transportation of messages using sockets. The CM layer creates JXTA sockets over the JXTA pipes, which provide the transportation of messages between peers. Since JXTA sockets are an extension of normal Java

sockets, they provide the same reliability as Java sockets. The CM is responsible for creating a server socket on each client listening for incoming connections; once a connection is received a JXTA socket is created for the connection between the two peers which is used until one of the two peers closes the connection.

2. This layer facilitates connecting to the JXTA network. The CM layer also ensures the initial connection to the JXTA network by joining the remote peer group when the application starts.

3. CM layer also helps in creating and joining of peer groups. The CM layer is responsible for the creation and joining of peer groups by creating, posting and finding JXTA advertisements for groups or users.

4. Finally, this layer allows fetching web pages from the Internet, chatting or sending and receiving multimedia content. Furthermore, the CM layer receives requests from the other components to send chat messages or multimedia content. On the receiving end, the CM layer gets the message and passes it back to the appropriate component for display or processing. Processing HTTP requests and responses is also the responsibility of the CM layer.

3.2 PECOLE Software Design

3.2.1 Package Diagram

PECOLE implementation is based on the Java binding of the JXTA protocol suite [21]. Given that the low level communication functions are handled using Java, it seems natural to use the Eclipse Java SWT library [52] for the Graphical User Interface (GUI). Furthermore, SWT offers a graphical component that embeds Internet Explorer or Netscape Navigator in a Java Window.

PECOLE jar file, the main package of the system, needs to run the main JAVA standard libraries included in the Java virtual machine, the JMF library, the graphical library of SWT and of course the JXTA library as shown in Figure 3.8. PECOLE's multipoint-to-multipoint audio/video conferencing feature is implemented using JMF package. We strive to design our application such that the GUI and workspace management logic is decoupled from the JXTA communication logic. This increases code modularity and clarity.

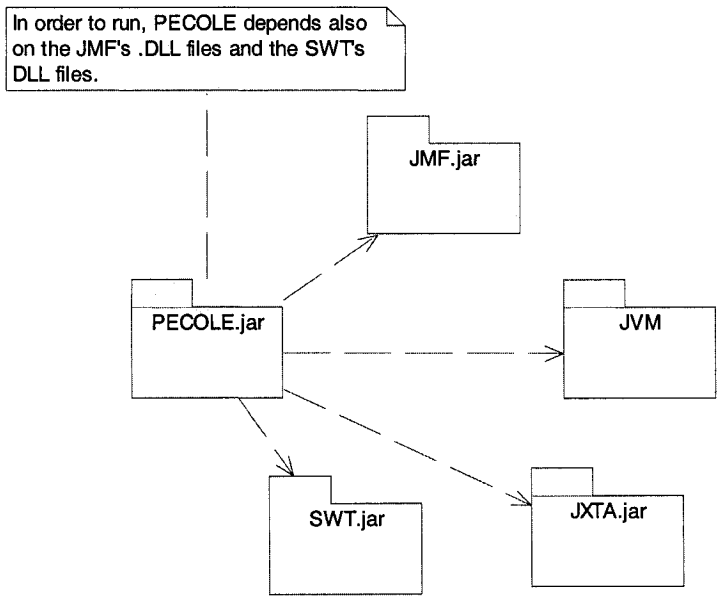


Figure 3.8 Package diagram

3.2.2 Use Case Diagram

Figure 3.9 provides a high level view of the main use case provided by the PECOLE environment. We can distinguish two main external actors: (1) User and (2) JXTA peers. By user we mean any peer while JXTA peers we mean the remaining peers in the collaborative session. The PECOLE collaborative system provides the following key functionalities:

1. Login and connect my JXTA peer to the JXTA overlay network using JXTA communication services and protocols;
2. Discover existing JXTA peers and sessions that are defined in the same domain;
3. Create / Join / Leave sessions;

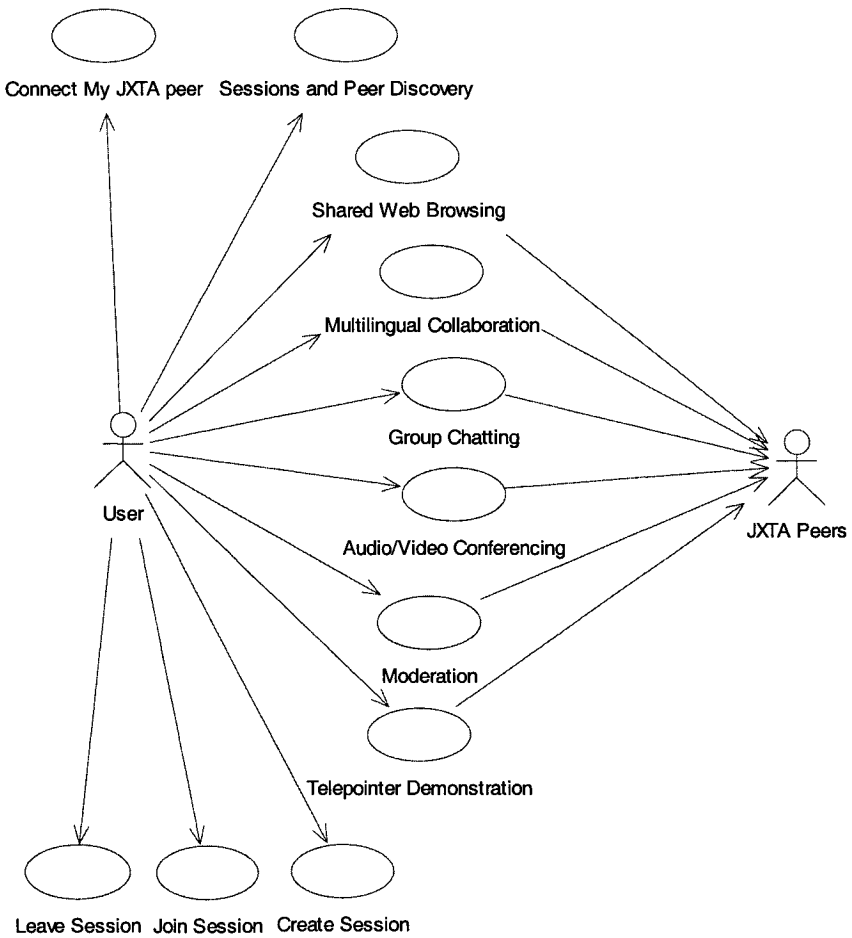


Figure 3.9 Main use case diagram

4. Collaborative web browsing;
5. Collaborative group chatting;
6. Collaborative audio/video Conferencing;
7. Multilingual Collaboration;
8. Moderation for floor control; and

9. Shared telepointer demonstration

Figure 3.10 shows the use case pertaining to the joining to the JXTA overlay network and discovering other resources or peers inside the JXTA network. The process of discovery applies to any JXTA resource: peers can discover other peers, and they also can discover peergroups, pipes, advertisements, and other resources.

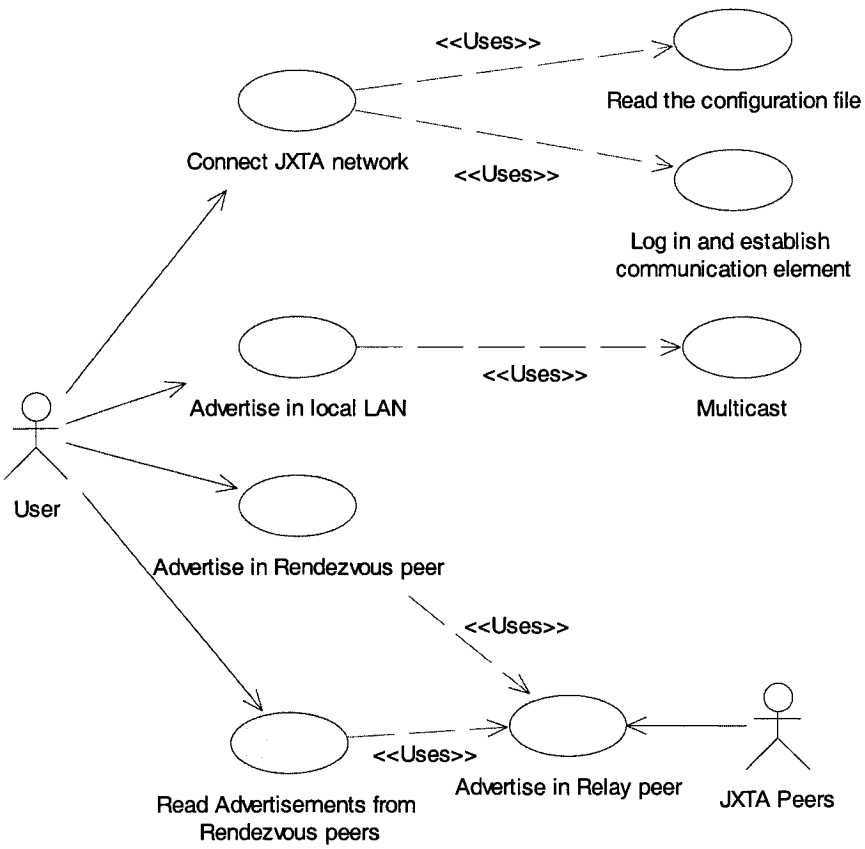


Figure 3.10 Use case diagram for peer discovery

As shown in the diagram, a peer first connects to the generic JXTA network as directed by a JXTA configuration file. The system also takes some initial parameters like user name, session name etc. to publish them to the outside world of JXTA virtual network. The peer first publishes his advertisement to the LAN he is connected by multicasting or broadcasting the packets. This advertisement in the local LAN helps fast discovery of peers who are in the same LAN. However, to discover the peers outside his

LAN, the peer publishes his advertisement to the rendezvous peer he knows, which acts like a lookup services: they keep a list of peers (and other JXTA resources) that they know about, and other peers query them for that list. A single rendezvous peer may have thousands of peers connected to it. An important characteristic of this kind of peer is that one rendezvous peer knows other rendezvous peers on the JXTA network. This hierarchical organization leads to scalable procedures for searching and publishing resources in the JXTA network. To access some resource, a peer asks for the resource to its rendezvous peer. If the contacted rendezvous peer knows the desired resource, it returns the resource address to the requesting peer. If it does not know the resource, it will propagate the search request to all rendezvous peers it knows. If any of these contacted rendezvous peers knows the requested resource, it will answer to the first rendezvous peer which in turn will respond to the peer that made the request. At this time the first rendezvous peer caches the received resource information to speed up future requests. If none of these rendezvous peers knows the resource, each one will pass on the request to all rendezvous peers they know, and so on. It is important to note that the search process affects only the rendezvous peers, and not every peer. Since each rendezvous peer can have thousands of peers connected to it, the number of peers involved in a search query is quite small. When a peer wants to publish some resource, all it has to do is to send relevant information about the resource to its rendezvous peer.

However, to traverse firewalls, we need another special kind of peer: the relay peer. A relay peer must be located outside of firewall or NAT boundaries, in a machine with a real IP address. It serves the purpose of relaying messages to peers located behind a firewall or NAT gateway. The relay peer usually listens on a port number that is open on the firewall for outgoing connections. Given the widespread use of the World Wide Web nowadays, the port 80 is open for outgoing connections on virtually every firewall.

In PECOLE, firewall traversal works in the following way: a peer behind a firewall or NAT gateway (which will be called inner peer) must know a relay peer. To connect to another peer on the JXTA network, the inner peer sends a request to its relay. This relay acts as a proxy, sending and receiving messages on behalf of the inner peer. Since the

relay peer is outside firewall or NAT boundaries, it can freely communicate with other peers in the JXTA network. A request issued by the inner peer is then propagated to the JXTA network by the relay peer. The corresponding response is sent back to the relay peer. (It cannot be sent directly to the inner peer, which is unreachable due to the firewall or NAT gateway.) In order to get the response from the relay peer, the inner peer periodically contacts the relay peer and checks if the response arrived. When the response arrives, the inner peer pulls it out.

PECOLE comes with a novel session discovery algorithm. The following pseudo code explains the underlying steps that take place when a peer wants to advertise its resources or to discover other peers or resources like peer-groups, pipes, and advertisements.

```
Set pipe type to multicast;
Create pipe ID using MD5ID to hash the peer group ID, the user ID and the pipe
    type (video pipe for video, audio pipe for audio and message pipe for chat
    messages);
Create pipe advertisement using the pipe ID;
Set session name as the pipe advertisement's name;
Set user's peer name as the description of the pipe advertisement;
Publish the pipe advertisement using the discovery service;
Create a JXTA Server Socket using the pipe;
```

Figure 3.11 PECOLE's Peer advertisement pseudo code

In order to calculate a well-formed unique ID to create the PECOLE JXTA server we used a form of hash function based on the Peer ID entered by the user. MD5 hash is used since it is long and random enough to generate a good ID. This ID is then used to create the JXTA server *Advertisement*. JXTA peers use these *Advertisements* to find each others over the network. Once the JXTA server is created, the peer has to publish its *Advertisements* locally and remotely as shown in Figure 3.11.

In order to discover other peers, a thread is launched to search periodically for new available peers and sessions and update the application's model and graphical interface. A callback function acts as a listener to receive all new sessions and peers' notifications. Figure 3.12 shows a portion of code in terms of pseudo code that first activates the listener and then starts sending discovery requests over the JXTA network to all surrounding rendezvous peers [14].

```
Start listening
   Add a listener to the discovery service
   Wait until we get a discovery message
      If discovery message is obtained
         retrieve group, remote and local advertisements from the discovery service
      End If
  End
End listening
```

Figure 3.12 PECOLE's discovery initiation pseudo code

Once the listener is called, which means that a new session or peer has been found, the name and other relevant information are extracted from its *Advertisement.* If it is a peer, then a handshake communication protocol is initiated before adding it in the list of new discovered peers.

3.2.3 Class Diagrams

We group the important java classes according to their functionalities that results communication layer class diagram, message class diagram, application layer class diagram, utilities class diagram and audio/video conferencing class diagram. We will now briefly describe each class diagrams.

Figure 3.13 shows the application layer class diagram of PECOLE. The main class is the *CollaborationWindow* class which represents the main SWT window for the peer to

interact with the collaboration sessions, to see the inputs and to create outputs. The *LauncherFrame* is the frame window for creating new collaboration sessions. The *CBListener* and *MyGlassPane* classes are used to provide the telepointer capability to the system.

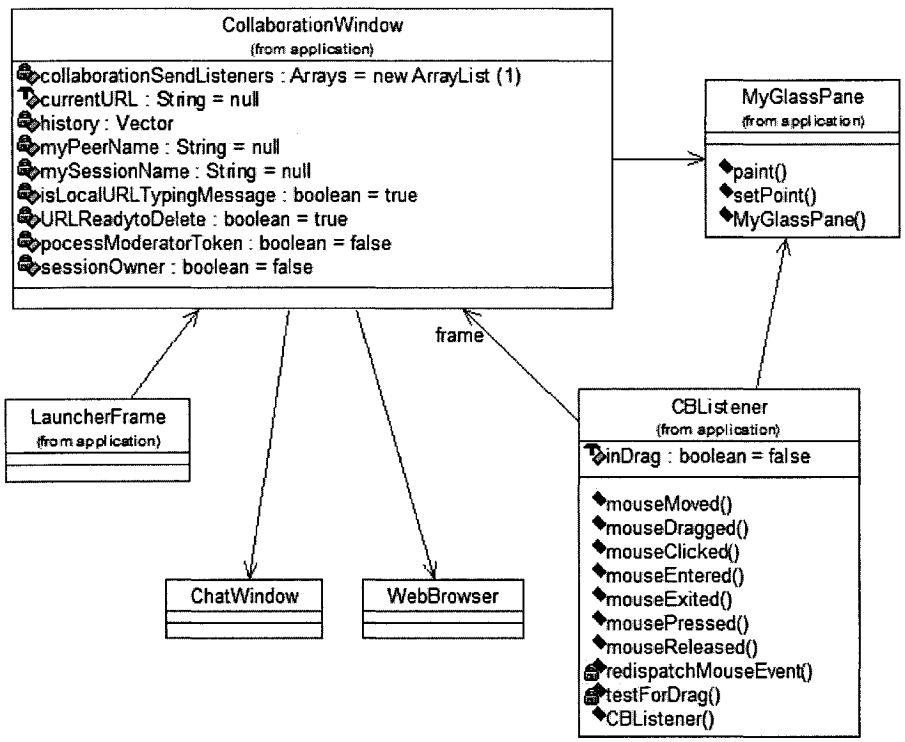


Figure 3.13 PECOLE Interface Class Diagram

Figure 3.14 depicts the Audio/Video conferencing class diagram. Since PECOLE enables the multipoint-to-multipoint audio/video conferencing feature, a set of instances of all of these classes is created for every connection made to a peer. If a peer requests an audio/video conferencing session to be started with another peer, then two threads are launched. The first one will be serving the video channel and the second one the audio channel. *ClientServerVideoFrame* is the main video class that handles the sending/receiving and encoding/decoding of video frames at the application layer. It also enables the user to see his video before sending it and receiving the other people's videos in a multi panel video window. *AudioStreamer* is the responsible class for the Audio part of the video conferencing session. Both classes initiate the creation of the threads mentioned earlier.

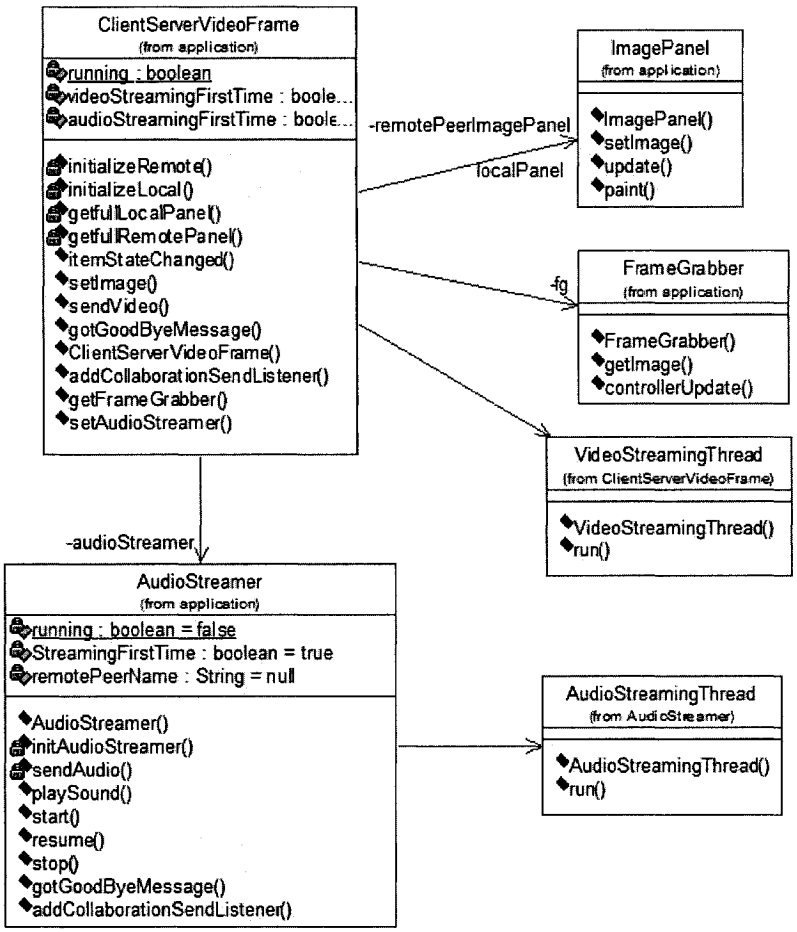


Figure 3.14 Audio/Video Conferencing Class Diagram

Figure 3.15 shows the communication layer 'CM' class diagram. The main class in this diagram is *CommunicationManager*. Its main role is to connect the peers in the virtual P2P JXTA network, and to create a collaboration server and JXTASockets needed for different multimedia applications. The CM Layer is using the *JxtaServerSocket* as its principal server. *JxtaServerSocket* is a bi-directional JXTA Pipe that behaves very much like ServerSocket. It creates an input pipe and listens for pipe connection requests. JxtaServerSocket also defines it own protocol. Requests arrive as a JXTA Messages. *ReaderThread* is the thread launched as a service for listening to the other peer's messages. Another important thread is *ConnectionServerThread.* This thread keeps watching the server socket. As other peers connect to the socket, this class creates a new connection session. Every connection a peer creates has a special manager. Its name is *SenderReceiverConnection.* It has the core code that starts a collaborative connection and

then handles the send/receive on the created I/O streams. The *CollaborationSendListener* is the interface that links the GUI to the session. It has a send method that is used to pass the *MessageObjects* to the session manager for transmission. Finally, the *ConnectionListener* interface announces that a new socket connection was created.

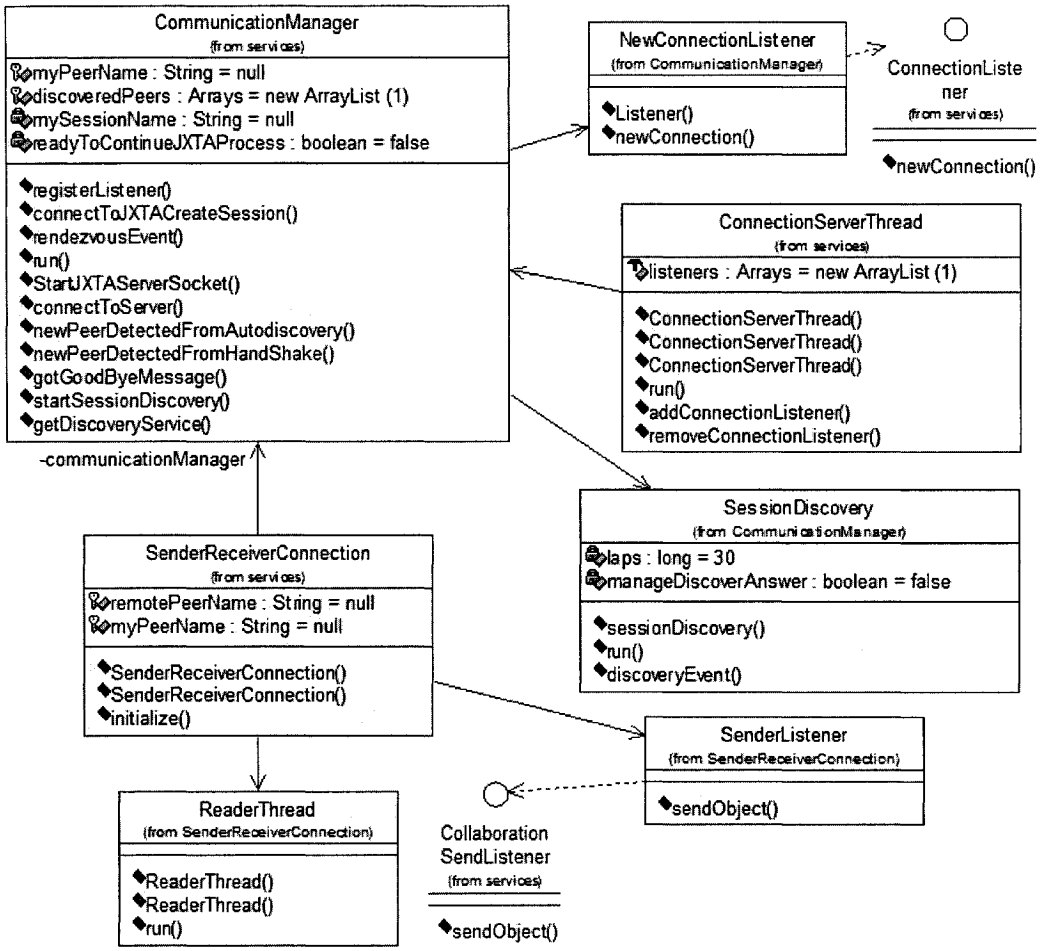


Figure 3.15 Communication Layer Class Diagram

All the communication in PECOLE is carried out in the form of messages. Messages class diagram (see Figure 3.16) groups the classes that handle different message types. *Chat Message* is the data sent between peers as chat message. *TelepointerMessage* contains mouse X, Y co-ordinates of the moderator that is sent to all the audiences. *URLTypingMessage* refers to the characters typed in the URL text field of the shared browser by the moderator, which is synchronously sent to all the audiences.

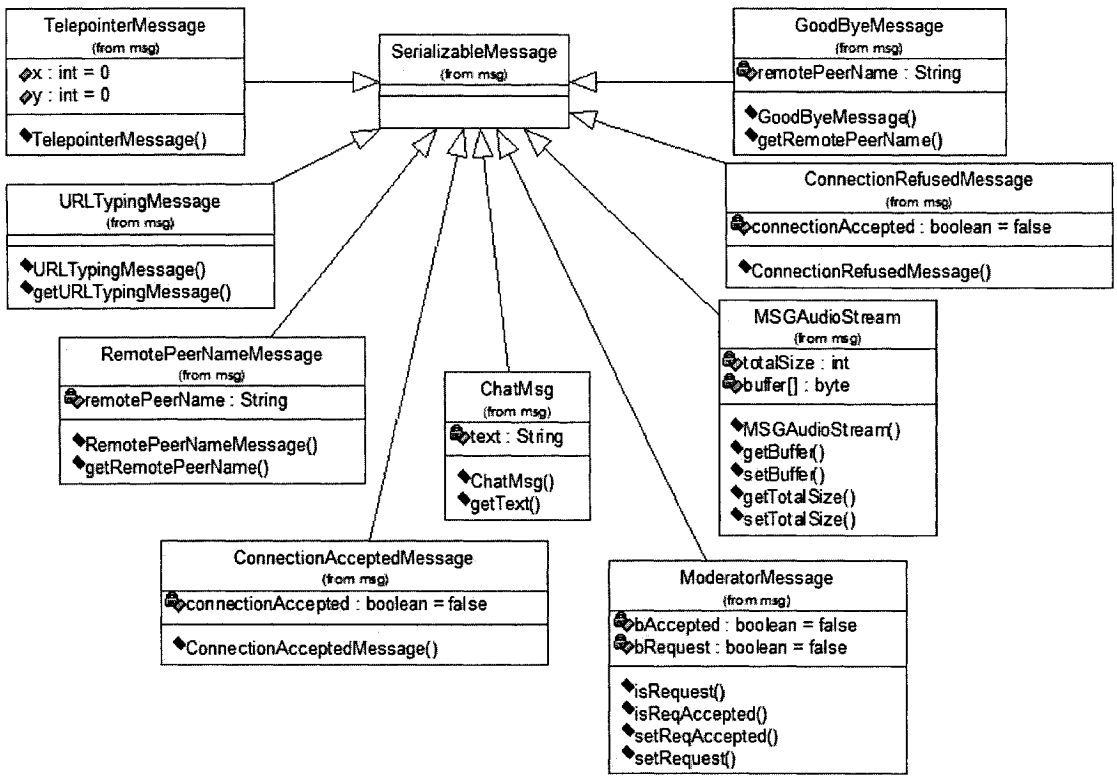


Figure 3.16 Massages class diagram

RemotePeerNameMessage is a handshake message sent by a peer to establish a connection with another peer. If handshake successful, the second peer sends *ConnectionAcceptedMessage* to the requesting peer. If handshake is unsuccessful, the second peer sends *ConnectionRefusedMessage* to the requesting peer. *GoodByeMessage* is sent to all the peers of the same session when a peer leaves a session. *ModeratorMessage* consists of two distinct parts. A *Response* message is sent by the moderator to any audience who is requesting for the moderator privilege. A *Request* message is sent by any audience to the moderator asking for the moderator privilege. *MSGAudioStream* is the audio message sent between two peers.

PECOLE uses some utility classes (see Figure 3.17). *MD5ID* class creates pipe and group IDs from MD5 Hashes. *PeerGroupTool* is a class of utility methods that create the peer group. *Utility* class facilitates in configuring the buffer size, timeout etc. parameters.

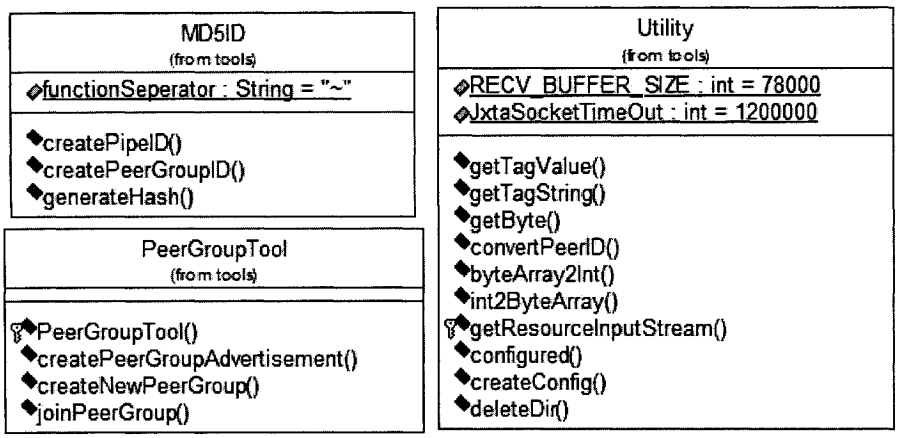


Figure 3.17 Utilities class diagram

3.2.4 Sequence Diagrams

In general all peers perform the same sequence of initialization when started:

1. Connect to the JXTA network;
2. Create and join a session;
3. Connect to or create a rendezvous peer group (if it does not exist);
4. Start the JXTAServerSocket in a separate thread listening for new connections from other peers in the same peer group; and
5. Create one collaboration session per connection request. The collaboration session handles the send/receive operations.

The sequence diagram in Figure 3.18 shows the interactions between the main actors and the system. Basically, the user triggers the start of this sequence diagram. This is done by entering the website URL that he wants to navigate to and clicking on the "Go" button. From a high level point of view, the System, which is an abstraction of the Shared Web Browser (SWB) being developed, receives this event, displays the web page to the user, and then broadcasts the same message to all JXTA peer group members in parallel. The same sequence of interactions happens for the chat system.

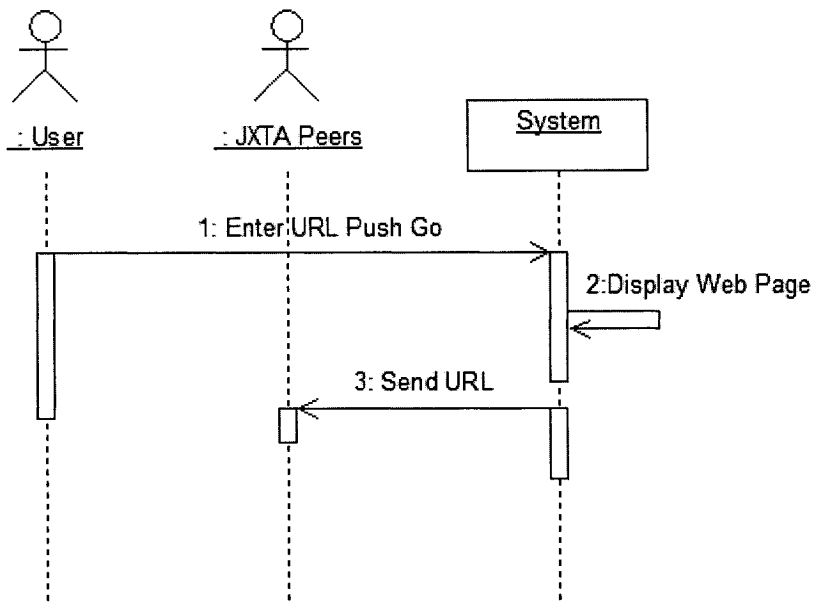


Figure 3.18 Main sequence diagram

Figure 3.19 depicts the sequence diagram of interactions that happen between users in more details. The "Peer" component in this diagram is an abstract class representing all the underlying layers till the JXTA platform. This is for the sake of clarity and simplification of the diagram.

The shared web browsing session scenario will produce the following sequences:

1. User enters a URL and clicks on "Go" or press "Enter";
2. The collaboration window receives the event and then renders the web page locally by fetching it from the Internet;
3. The collaboration window informs the underlying layer of the successful completion of that task;
4. The workspace layer passes the request to the service layer, which creates a URL event message and then broadcasts this message in the JXTA network; and

5. The RemoteMessage is received by the external actor ":JXTA Peers". Then, every peer reproduces the behavior of the event received locally by dispatching it to the right GUI component.

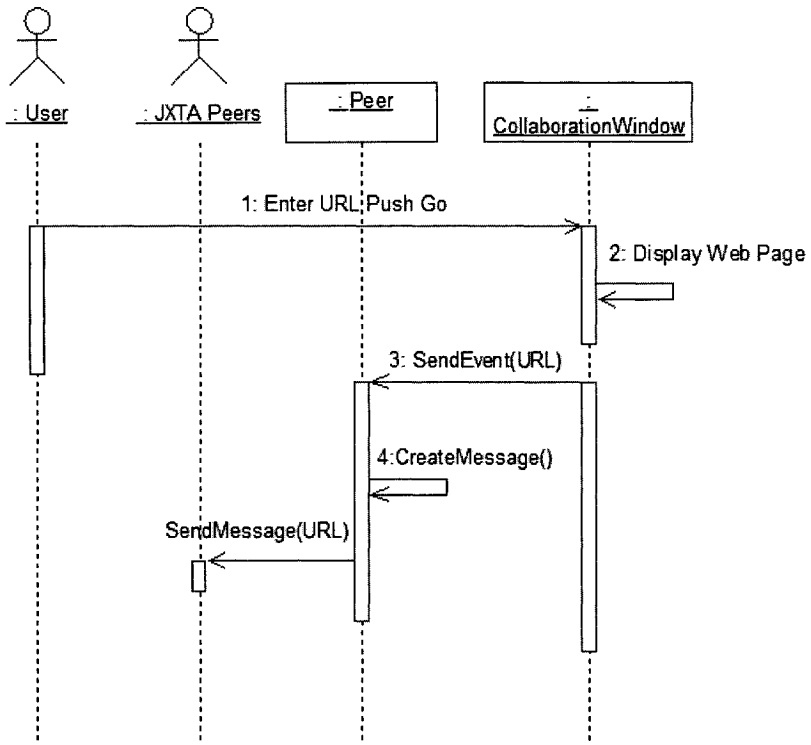


Figure 3.19 Collaborative browsing sequence diagram

Chapter 4 Implementation

There are several applications integrated with the current version of PECOLE as described in chapter 3. These are shared browsing, multilingual collaboration, telepointer, moderation, multipoint-to-multipoint audio/video conferencing, and chat. The following sections provide the implementation of each of the applications mentioned above.

4.1 Shared Browsing

The shared browser is implemented with the Eclipse project's Java-based SWT technology and extends all the properties of the Operating System (OS) default browser. If the default browser is Internet Explorer (IE), then PECOLE extends all the properties of IE in addition to its own set of browser properties. The same is true for any underlying web browser, irrespective of the OS in which PECOLE is operating [52]. So, it supports javascript, vbscript, xml, plugins, applet, external viewers etc. This property helps sharing not only html documents using different scripts but also sharing video and audio, even playing in an external player.

As shown in Figure 4.1, the shared browser window covers most of the PECOLE applications. We now briefly introduce some of the GUI functionalities of the browser. Label 1 shows the basic web content area where the body of a document appears while Label 2 refers to the area where users type the URL of a resource. Label 3, 4 and 5 indicates 'Go', 'Back' and 'Home' buttons respectively. In order for requesting moderatorship, a peer has to click on 'Moderator Request' button as shown by Label 6. Label 7 shows the 'Exit' button and Label 8 lists the number of peers joining the session

while Label 9 shows the current session name. Chat messages appear in the area shown by Label 10. Label 11 indicates a button group consisting of three buttons: 'Searching Peers', 'Join A Peer', and 'Start 3D Collaboration'. 'Search Peers' is used to find search other peers who are in the same session. This helps in finding other peers dynamically. By clicking on 'Join A Peer' one can initiate a one to one collaboration. Label 12 shows a pop up window that appears to all other peers of the same session when any peer leaves a session.

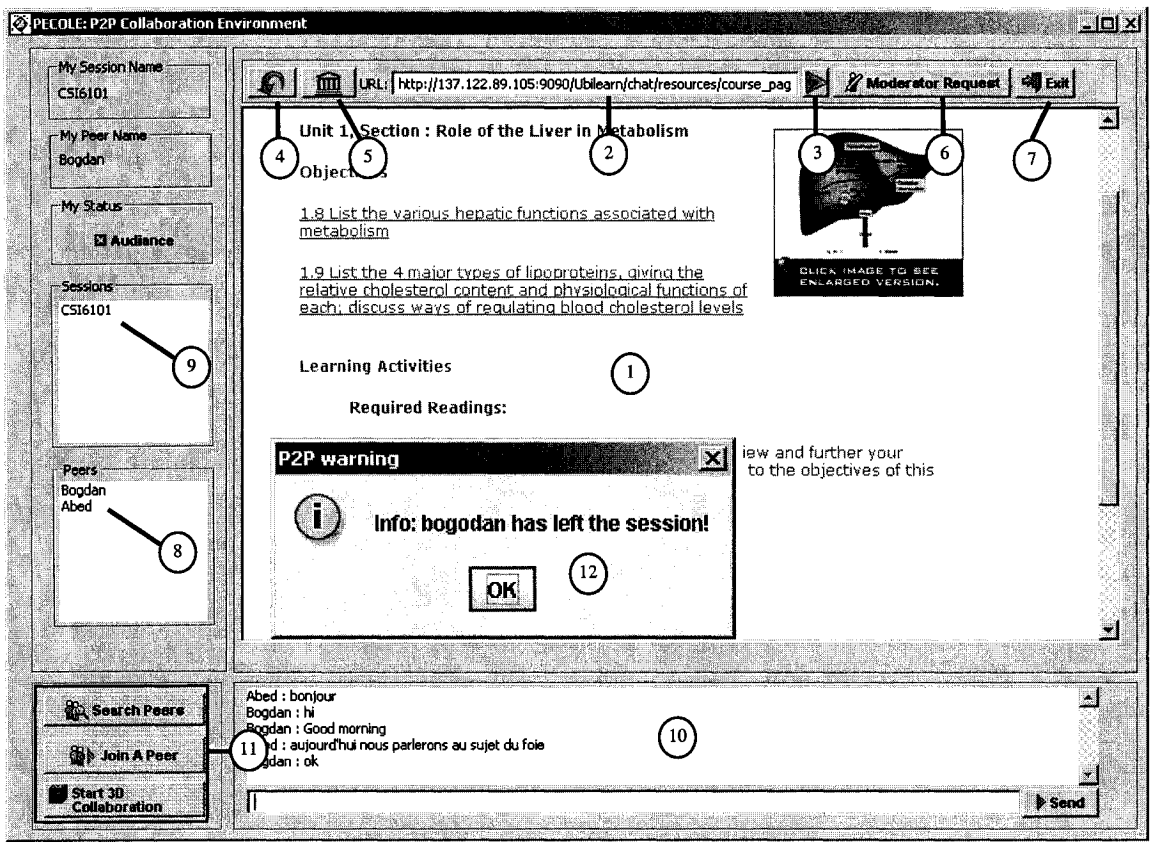


Figure 4.1 Shared browser GUI

At present, PECOLE provides the following shared operations concerning browser's contents:

- Entering a URL from the URL field;

- Pressing navigation buttons, such as Go, Back, Home, Moderator Request, exiting from the system;
- Clicking a hyperlink in a web page to go to another page; and
- Other basic operations that are provided by popular web browsers to date

4.2 Multilingual Collaboration

Using the shared browser shown in figure 4.1, peers can browse multi language web documents. We take a bilingual classroom, English and French, as an application of PECOLE. In this scenario, we assume that the students are divided into two language groups, English and French. The teachers or TAs have moderator privilege. In the course registration server, the students are registered beforehand who want to receive the service. Initially the teacher or TA starts the session with their language of preference while the student peers also start their session with their language of preference. The moderator has the control over the browsing.

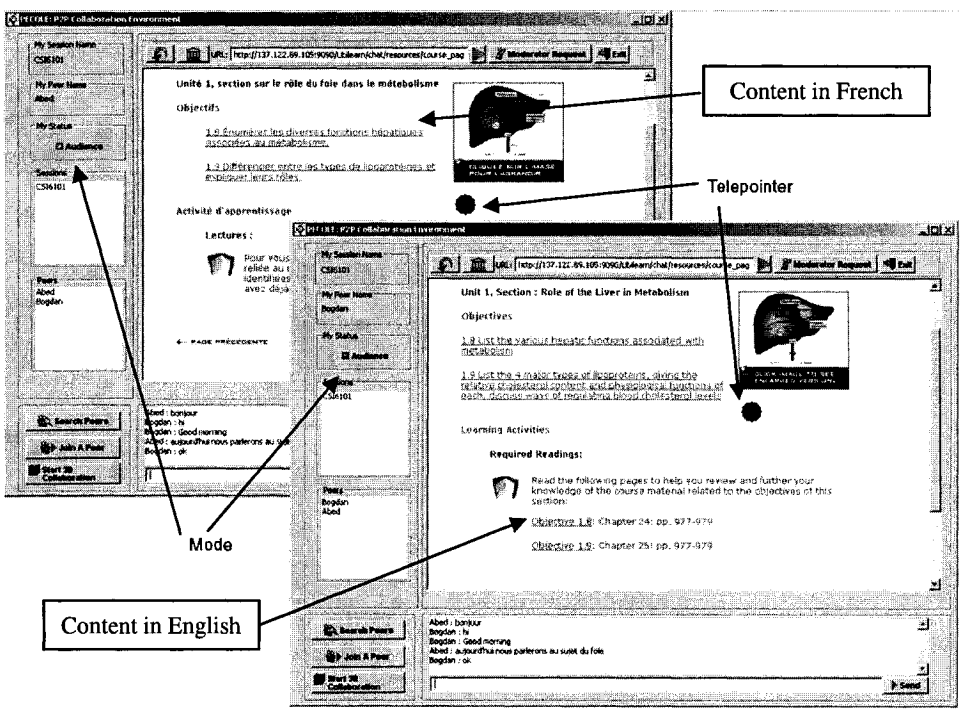


Figure 4.2 Telepointer and multilingual collaboration demonstration

Figure 4.2 shows two peers, one with language English and other with language French, browse anatomy lecture notes. The peer having language option English is the moderator. When he browses, the other peer i.e. student views the same content in French.

4.3 Telepointer

The shared telepointer is intended to provide awareness information of moderator user's actions by showing the movement of moderator mode user's cursor in a browser area (see red pointers in Figure 4.2). Audience users can change from audience mode to a moderator mode. This will send a request to the current moderator that user A wants the moderator privilege. If current moderator gives permission by releasing the telepointer lock then user A holds the lock for the telepointer. In the moderator mode, the captured cursor position will be sent to others. A user in the audience mode can only see parent user's (moderator) cursor. A telepointer position is sent to others through the JXTA socket.

4.4 Moderation

We have two types of sessions in our collaborative system: non-moderated and moderated. A non-moderated session does not require any specific peer to be a moderator. In such session, the peers can perform any available operations such as chatting with other peer and starting audio and video conferencing. A moderated session is initiated by a session chair. A moderator may be the first peer entering the session, or be a pre-defined session chair. Multimedia collaboration, shared browsing and demonstrating telepointer are performed under moderated session to ensure that only one peer at any moment can have moderator privilege. However, while the session is going on, moderatorship may be requested by any peer of the session. If the current moderator accepts the request, he releases the write lock and becomes an audience while the

requesting peer becomes the new moderator. Overall, the session chair has the super privilege to take away this temporary moderation privilege that was assigned to other peers. Figure 4.3 shows a moderated-session time message communication where the following steps happen.

1. 'mehran', the moderator of the session logs in the session;
2. One of the peers, 'bogodan' requests current moderator 'mehran' to assign him moderator privileges;
3. 'mehran' rejects 'bogodan' to become the moderator of the session; or
4. 'mehran' agrees to give 'bogodan' moderator privileges and 'bogodan' becomes the moderator of the session SYSC 5800

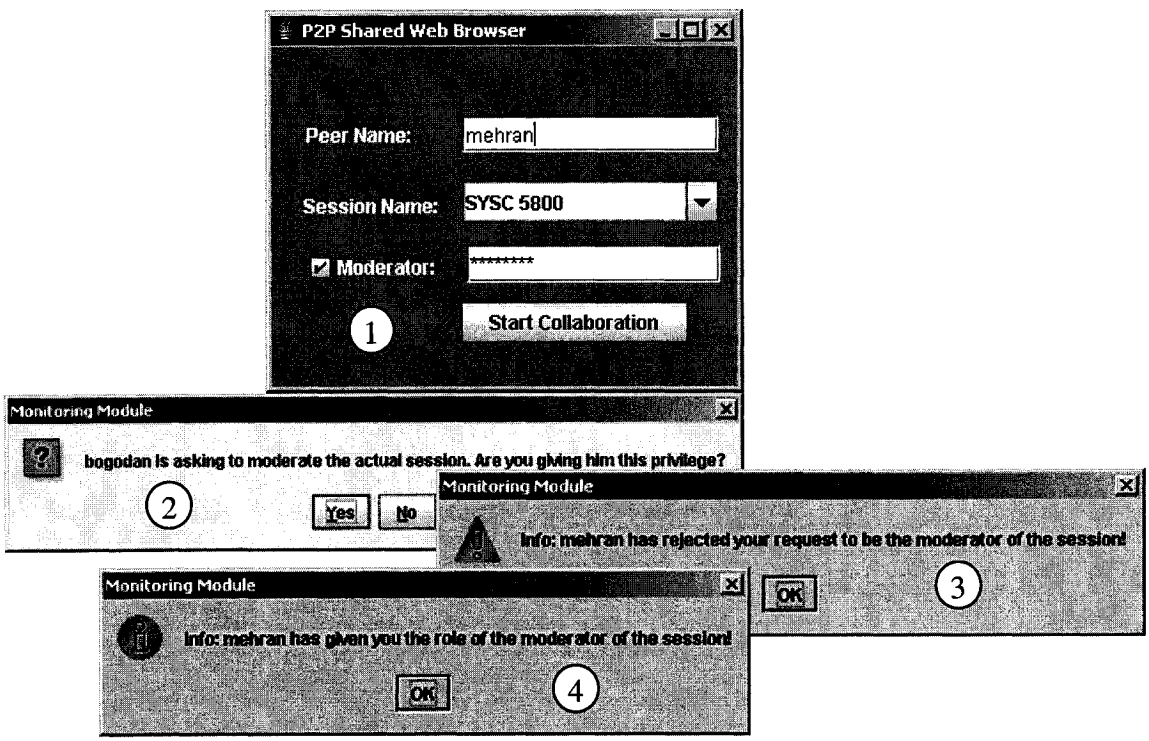


Figure 4.3 Moderation facilities in PECOLE

4.5 Multipoint-to-Multipoint Audio/Video Conferencing

When any peer starts PECOLE and try to join the session, the system first searches for necessary audio and video hardware (like web camera, microphone or speakers) for conferencing. If anything is missing, it warns the peer to mount the hardware for successful conferencing. If the system passes the hardware test successfully, then upon entering inside any PECOLE session, the peer sees the audio/video interface. The numbers of audio and video interfaces are dependent upon the number of peers in the session including himself.

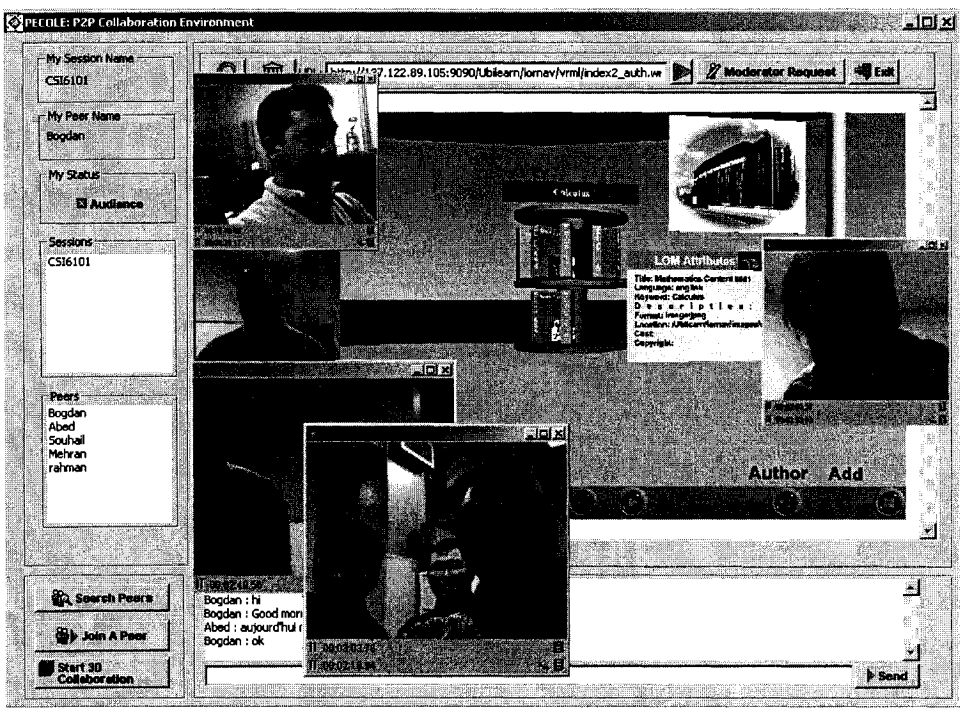


Figure 4.4 Five (5) peers in an audio/video conferencing session

Figure 4.4 shows five peers have joined in a collaboration session while engaged in an audio and video session. Each peer has the control whether to stream his video and/or audio to any specific peer or not. Peers have also the control on how many of the peers they want their audio/video to be streamed to. At the extreme end, he can send his audio and video stream to every peer of the session forming a multicasting channel. If every peer follows the same scenario, they form a multipoint-to-multipoint audio/video

conferencing session. Again, any peer has choice, he can either stream audio and block video or vice versa. This feature is pretty helpful in saving bandwidth of the network. The top timer panel at the bottom end of each video interface of figure 4.4 is the video timer while the bottom one is audio timer that shows the time elapsed in the conferencing.

One of the unique features of PECOLE is the streaming of audio/video bypassing the firewalls and NATs. We tested PECOLE having peers in Germany, Montreal and Ottawa. But we got the best streaming performance while peers are inside same the LAN.

4.6 Chat

Like any P2P system, PECOLE provides existing rooms for chatting and if needed, any developer peer acting as system developer can also create new rooms so that others can join the room. The key issue here is that all the peers that are chatting with each other are part of the same session. In principle, all they have to know about the session is the session name. Label number 10 of Figure 4.1 shows the chat area of PECOLE.

Chapter 5 Evaluation and Results

To evaluate the performance of the proposed system, we tested PECOLE from different regions including Germany and different places in Canada. The peers connected to each other in real-time. Figure 5.1 shows a typical test venue of PECOLE.

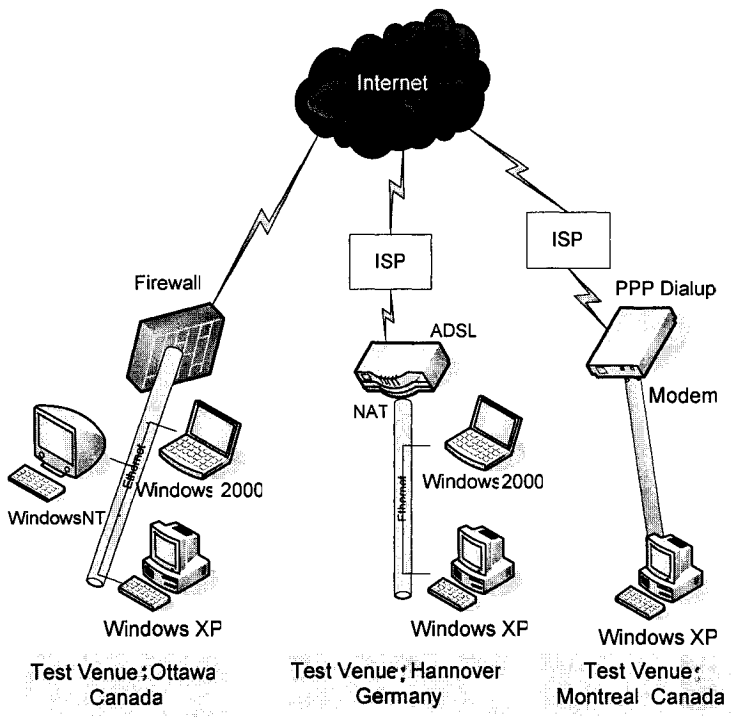


Figure 5.1 Test environment

In the following section we describe the characteristics of each of the venues used in the performance tests:

Ottawa, Canada:

- Multimedia Communication Research Laboratory (MCRLab),
- Under strict firewall (University firewall),
- Number of peers used in the test: three,
- OS of each peers: Windows XP/2000,
- Network: 100 Mbps Ethernet LAN.

Hannover, Germany:

- Under NAT,
- Number of peers used in the test: two,
- Using ADSL,
- OS of each of the peers: Windows XP/2000,
- Network: 10 Mbps Ethernet LAN.

Montreal, Canada:

- Dial up user connected to ISP through modem.
- OS of the peer: Windows XP.

To evaluate a fully distributed P2P-based collaborative system, peer discovery time is one of the most crucial factors because it is a basis for group/session formation. Once the group is formed, the next factor that affects the communication is the application to application layer message communication delay. A third element that also qualifies a multimedia distributed system is jitter, which is the difference in delay between two successive messages or packets. Here, by the notion of jitter, we mean the delay between two consecutive messages arriving at the application layer after being received by another peer. Finally, as the audio and video is a real-time application, the delay in frames is an important evaluation criterion to measure the quality of experience as defined in [50].

To measure the delay for different applications, we took the data in different times of a day including day and night. Communication delay depends mostly on Internet traffic

and the traffic around the external relay peer, which is variable throughout a day. So, for better evaluation of the system, we tested the system at different times of a day.

The following items and/or packages were installed and configured for the smooth startup of PECOLE.

- JXTA: Jar files that come with the stable builds version 2.3.3 released in 03/14/2005.
- Peer configuration: Simple peer using a relay (as the peer is behind firewall/NAT).
- Transport protocol: TCP and HTTP (as the peer is behind firewall/NAT).
- Streaming audio and video: JMF 2.1.1e
- Virtual machine: Java Virtual Machine that comes with JDK 1.5.
- System distribution and server processing in each peer: JNLP 1.5 and Java Web Start.

5.1 Test Results

5.1.1 Peer discovery time

Peer discovery time is the time that is required to discover a peer of the same session before starting the collaboration. We tested peer discovery time extensively in different portion of a day for about 4 months. According to the test data we have found so far, the peer discovery time varies based on two scenarios. In case of the first scenario, the peer is behind firewall or NAT while in case of the second scenario the peer is inside the same LAN. It took approximately 30 seconds to 4 minutes to discover a peer when the peer was behind firewall or NAT. This extra overhead time is required because, in this case, we use HTTP to connect to the external relay peer for the peer advertisement in order to bypass the firewalls or NATs. In this scenario, peer discovery time is dependent on the traffic around the external relay peer itself and thus, the time varies between 30 seconds

to 4 minutes. For the second scenario, where all the peers were inside the same LAN, the peer discovery time was almost negligible.

5.1.2 Average message communication delay

In order to measure the delay associated with different applications like chat, shared web browsing, shared telepointer and moderation, we have run PECOLE in testing mode for approximately 14 hours, which includes both day and night time so that the delay takes into account varying traffic conditions of Internet. Figure 5.2 shows the average application to application layer round trip delay between peers for different PECOLE's application. It shows that shared browsing suffers the highest delay which is approximately 104.35 milliseconds. The reason is that each peer's browser needs to download the web content that is currently viewed by the moderator by sending an HTTP request to the server where the web content is located. So, in addition to URL message communication delay, the web content loading time is also added with it. It makes the delay higher in comparison with other types of application delays. We wrote a test program in java running in the moderator peer that starts a timer once the moderator clicks on a link in the shared browser and it goes on incrementing the timer till all the peers in the group gives response back to the moderator after the URL is received and downloaded by the application layer of the audiences. The test program itself introduces insignificant delay compared with the application delay because it takes only a single or two instruction execution cycles of microprocessor. However, we defined a Time To Live (TTL) for a peer's response. If we do not get any response from any peer by this threshold time, we stop the counter and write the round trip delay in the log file. We took 1000 such instances of delay for each of the applications, which took approximately 14 hours. We then calculated the average delay in milliseconds for each specific application. We used PECOLE with 3, 4 and 5 users in different instances, but the average delay was almost the same, which is shown in Figure 5.2. Similar type of test was conducted for other applications (see Figure 5.2). According to the data we found, telepointer message took approximately 29.25 milliseconds while moderation message made a delay of 20.567 milliseconds. Chat data took the least time, approximately 17.366 milliseconds.

The reason for this small delay (for chat, telepointer and moderation messages) in comparison with that of the web request is that in these cases, the update messages are sent directly between the peers and therefore, no extra delay of requesting web content from a web server is required.

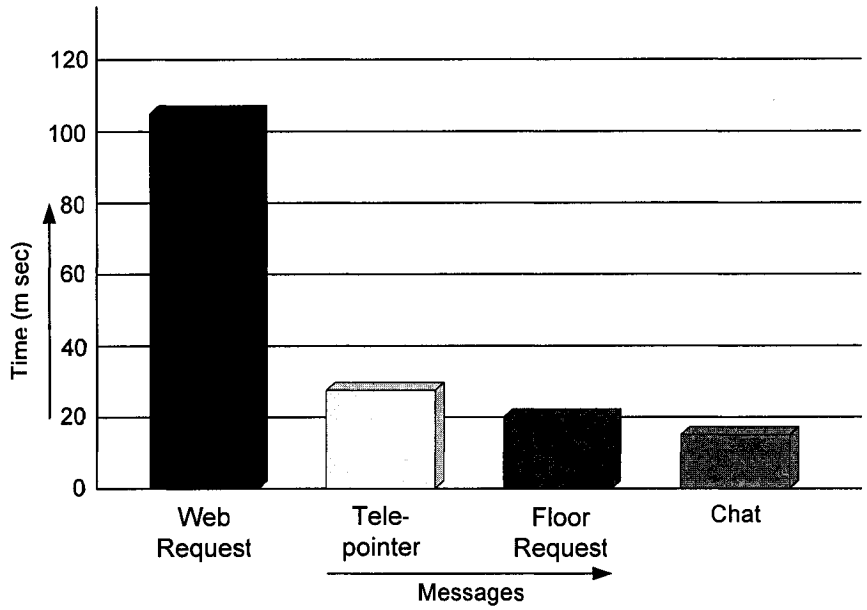


Figure 5.2 Average application layer to application layer message round trip time.

5.1.3 Application layer to application jitter

Jitter is a measure of the variability of the latency over time across a network. Jitter can be measured for a tightly coupled system where whole network follows the same system clock or a loosely coupled system where no network clock is maintained. For a fully decentralized system, as in the case of PECOLE where the peers can be from anywhere in the world employing different clock synchronization servers having different protocols like Network Time Protocol (NTP), Daytime Protocol, Time Protocol etc. and under different firewall settings, it is not feasible to maintain synchronization of jitter with respect to any central timing server. Moreover, as we are measuring the jitter from the application layer, we did not take into account the synchronization issue with respect to the computer clock cycles either. Because we are using the JXTA network and JXTA network hides the physical network, we did not attempt to account the impact of JXTA

network delay on jitter, which is out of scope of this thesis work. We took similar approaches of testing PECOLE for calculating jitter as discussed in the previous section. Figure 5.3 shows the average application layer to application layer jitter, which we calculate in the following way:

For every two messages M1 and M2, we define

$$\text{Jitter } (J_i) = (R_2 - S_2) - (R_1 - S_1) \text{ ---------- (1)}$$

where R refers to the receiving time of an acknowledgement message recorded by a timer after the message travels from the application layer of other peer and S refers to the sending time of a timer when the message is just sent from the application layer of a peer destined towards other peers.

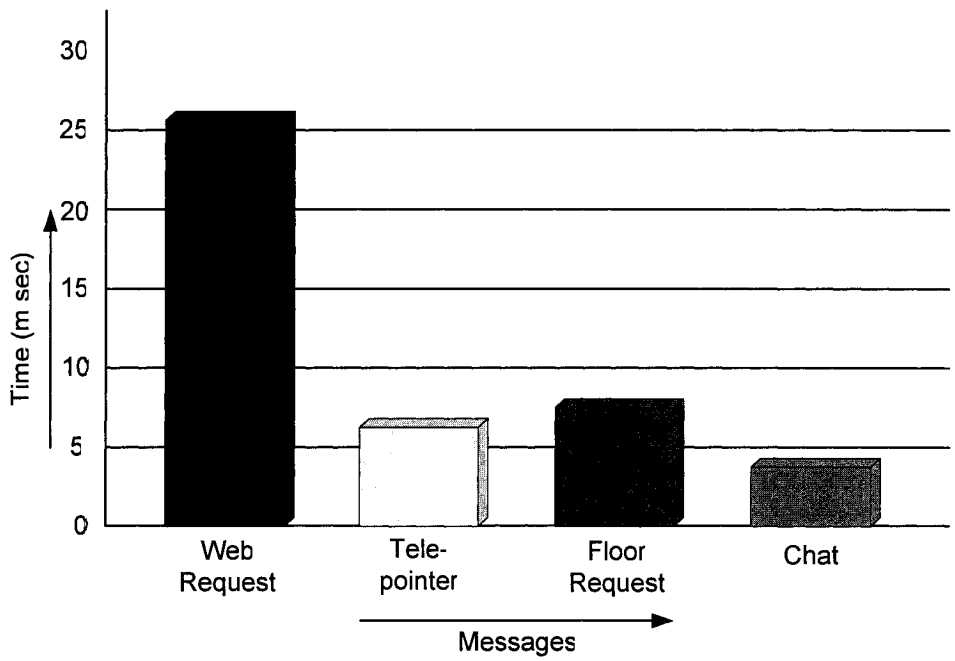


Figure 5.3 Average application layer to application layer jitter

Now we get the

$$\text{Average Jitter} = \frac{\sum_{i}^{N-1} Ji}{N-1} \text{ ---------- (2)}$$

where N= total number of messages, which is in our case 1000.

The data we found shows that the mean value of jitter for the shared browsing is still the highest which is followed by moderator messages, telepointer messages, and chat message causes the least jitter.

5.1.4 Video frame transmission delay

We tested the video frame delay between application layer to application layer while 3, 4 and 5 peers were in a multipoint-to-multipoint video conferencing session. Following is the specification of video frames we used for the test:

- Video: H.263 compliant,
- Resolution: 320x240,
- Frame rate: 15 fps,
- Bit rate: 70-80 kbps.

The test was also carried out at two different times of a day; during day time and during night time. In each case we took 11,000 sample video frames as test sample. Figure 5.4 shows the average transmission delay of sending a frame and receiving the response from all the peers of the group, taking into consideration the frame travels from the application layer of one peer and reaches to the application layer of other peers in the same session and finally the acknowledgement messages again reaches to the application layer of the sender. We call the delay application layer to application layer round trip delay. The red column shows the delay during day time and the blue column shows the night time delay. The average round trip delay of 5 peers in a video session during day time was the highest, which was approximately 37.82222 milliseconds while at night

time it was approximately 35.68708 milliseconds (see Figure 5.4 (a)). In case of 4 peers the delays were 30.43259 milliseconds and 30.74383549 milliseconds during day and night time respectively (see Figure 5.4 (b)). For 3 peers, the delay we found was 29.25693 milliseconds during daytime while 30.89625 milliseconds during night time (see Figure 5.4 (c)).

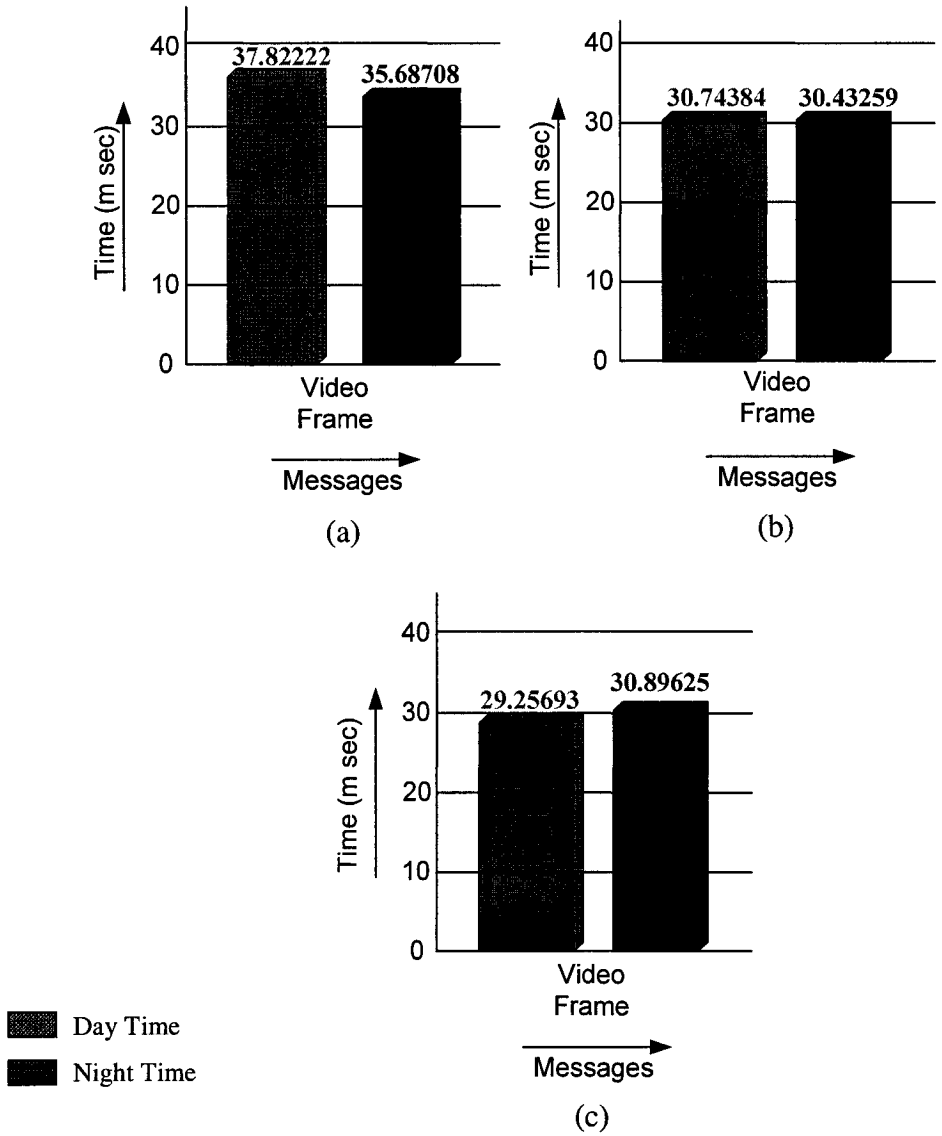


Figure 5.4 Average application layer to application layer round trip delay for a compressed video frame for (a) 5 peers (b) 4 peers and (c) 3 peers

Because the test results reflect the delay in terms of round trip delay, the end-to-end delay will indeed be less than the delay for each case once we subtract the returning time of the trip. The returning time is the time required for the acknowledgement message of the video application from remote peers that the frame is successfully received. For the sake of measurement, we set up this acknowledgement provision and we use TCP for the transmission of frames. However, to make the communication faster, the real audio/video conferencing module does not acknowledge the video frame receipt by remote peers, and thus allow us to use UDP.

As suggested by the authors in [50], maximum tolerable end-to-end delay (including propagation) for natural hearing is 100 milliseconds hence, the maximum delay for picture frame display should also be 100 milliseconds because both audio and video should be synchronized in real-time,

i.e., processing delay + network delay + network resynchronization delay + processing resynchronization delay + processing delay = Constant <= 100ms [50]

which is in all the cases <= 37.82222 milliseconds. This average delay makes the audio and video conferencing quality tolerable.

The time delay of different collaborative applications of PECOLE varies and little bit unpredictable because all the communication goes through the relay and rendezvous peers. The traffic of the relay and rendezvous peers determines the overall communication delay. So we can conclude that the overall time delay can be further minimized if the JXTA network delay can be minimized.

5.2 Comparison between PECOLE and Other Collaborative Environments

This section briefly compares each collaborative tool, introduced in details in section 2.2, with our developed tool PECOLE. Although it is very hard to compare the features

among the tools due to their diversified nature of features, Table II shows comparison based on some of the collaborative aspects they provide.

Table II Comparison between PECOLE and other collaborative environments

	PECOLE	**Groove Network**	**JASMINE**	**NetMeeting**	**SameTime**	**TOMSCOP**
Goal	Collaborative environment	Virtual workspace	Java application sharing	Collaborative environment	Collaborative environment	Collaborative environment
Multipoint-to-multipoint Audio/Video Conferencing	YES	NO	NO	NO	NO	NO
Multilingual Collaboration	YES	NO	NO	NO	NO	NO
Firewall Tunneling	YES	Yes	NO	NO	NO	YES
Telepointer Support	YES	NO	NO	NO	NO	YES
Shared Browsing	YES	YES, but limited options*	YES	NO	NO+	YES
Backup Server	N/A	YES	NO	YES	YES	N/A
Chat	YES	YES	YES	YES	YES	YES
Platform	JXTA	Hybrid!	Client-server	Client-server	Client-server	JXTA

*Cannot browse HTTPS links, cannot share applet and other java applications etc.

+Only screen sharing in windows environment.

!Uses both client-server and P2P architecture.

Because most of the terms in the first column of Table II are self explanatory, as they are discussed in earlier sections, we only define three terms with respect to this scope; goal, backup server and firewall tunneling. The first row refers to the goal of the tool while the second row signifies either they need any back up server for maintaining users logs, caches etc. or not. Firewall tunneling refers to the fact that most firewalls prevent the socket communication, except for HTTP. Therefore, by wrapping all messages in HTTP requests, one can overcome the firewall. This is sometimes called firewall/HTTP tunneling.

As shown in Table II, the 2nd row compares the multipoint-to-multipoint audio/video conferencing feature, which is not present in any of the tools except PECOLE. The same thing is for multilingual collaboration feature that is implemented by PECOLE while no other tools provide shared multilingual browsing facility. In case of firewall tunneling, only the tools built on top of P2P framework, can overcome firewall such as PECOLE, Groove and TOMSCOP. Because of latency problem with the client-sever model, only PECOLE and TOMSCOP, which are built on top of JXTA framework, implements shared telepointer. Shared browsing and chat are common to most of the tools while back up server is not applicable to P2P-based tools because P2P paradigm uses local cache of each peer for data storage.

Chapter 6 Conclusion and Future Work

6.1 Conclusion

In this thesis we described the design and implementation of a new tool, called PECOLE. PECOLE is a P2P collaborative multimedia environment which supports several applications such as shared web browsing, chat, moderation, multipoint-to-multipoint audio/video conferencing, telepointer operation and multilingual collaboration. These applications make the collaboration system more user-friendly and the support of multimedia adds a new dimension in collaboration. Compared to other similar solutions, our implementation has some significant features.

To make the communication faster, PECOLE uses three separate JXTA sockets. One is used for audio, one for video and the third one for control and/or data messages. These different sockets for different data types result in the whole communication system more scalable and faster.

To the best of our knowledge, the multipoint-to-multipoint audio/video conferencing system over P2P network introduced in this thesis is not implemented by any P2P system before. The audio/video stream can overcome firewalls and/or NAT's. Introducing multipoint-to-multipoint audio/video conferencing will indeed enhance the quality of collaboration system much more and will lead to ubiquitous conferencing system. Also, the end-to-end delay of video frames is within tolerable limit. However, when the peers are behind the firewalls, then PECOLE uses HTTP protocol, resulting in slower audio/video communication. This is a drawback of the system. If peers using PECOLE are in the same LAN, it gives the best performance.

The shared browser of PECOLE is the initial interface for the whole collaborative environment and the browser depends on the successful launching of JWS which is a locally installed helper application in each peer. As the Java browser is based on SWT technology, whenever a peer wants to access PECOLE using JNLP, the application is automatically downloaded to the user's workstation and starts the Java run time environment [49]. The whole process requires little to no interaction from the end-user and greatly simplifies the effort to distribute PECOLE.

Many research groups are working on collaboration systems that support dynamic session discovery. However, their approaches mostly depend on powerful centralized servers. The session management is a heavy-duty module in their systems. With our novel peer search technique, any peer can join any session of PECOLE dynamically with least discovery time.

Multilingual collaboration is a research area that is gaining interests among researchers and software industries. In this thesis we have shown the architecture of a bilingual, English and French, collaborative browsing system. Adopting the architecture, a collaboration service can be built that accommodates more languages which will be helpful in online classrooms or virtual meeting places where participating students or meeting members have different languages.

Many researchers and software industries are working to provide shared telepointer inside the virtual collaboration workspace. The shared telepointer that is presented in this thesis is based on SWT technology and shows improved performance in comparison with that designed with Swing based technology. Shared telepointer provides a significant role in a collaborative session.

Optimizing existing collaborative systems in P2P network is always the goal to researchers who are working in this area. The applications added with PECOLE, which are multipoint-to-multipoint audio/video conferencing, multilingual and multi-session collaboration, telepointer support, moderation support, and shared browsing will provide

more complete and precise support to collaborative users since their requirements are fully considered.

6.2 Future Work

Some important work still remains to be done. First, we need to work on dynamic group management in PECOLE. In the current version of PECOLE, we assume that the system developer creates the sessions or rooms of collaboration and publishes the session names to all the peers through other means of communication like e-mail or telephone. For our context students and teachers/TAs join the classroom session, where students register beforehand. PECOLE does not support dynamic creation of rooms yet. This is only possible if all the peers including the relay peers are within the same group. We need to configure our own relay peer instead of Sun Microsystems' relay peer that we are using currently.

Second, adding voice recognition system within the collaboration interfaces to give a new voice dimension in the system.

Third, we intend to implement some other popular collaborative applications with PECOLE like whiteboard, instant messaging, shared document authoring, email, forum etc. We also have planned to implement WebDAV [13] protocol soon.

Fourth, we plan to implement a pure P2P based authoring tool so that the authors working on mobile or ad-hoc networks can visualize and author. This might lead to a huge research area of authoring any system that has XML file format like visualizing and authoring Web Services, authoring in sensor network environment etc.

Fifth, we did not consider the synchronization issue between the audio and video sockets. Because JXTA socket is introduced very recently and it is in its premature phase, we will work on it to synchronize both audio and video in real-time in the future version

of PCEOLE. Also, we will try to improve the video compression ratio using MPEG-4 [27] CODEC.

The application of XML based distributed multimedia collaborative environment becomes wider with the development of the Internet and multimedia communication. To develop a qualified and practical multimedia collaborative system for P2P network is always a challenge to all the researchers working in this field. We hope that the work we have done has contributed to this research and the work we will be doing in future will help us achieving higher goals to provide more powerful, realistic and generic collaborative systems to users.

Bibliography

1. El Saddik, A.: Interactive Multimedia Learning – Shared Reusable Visualization-based Modules. Springer, Berlin, pp. 101-132 (2001).

2. El Saddik, A., Shirmohammadi, S., Georganas, N. D., Steinmetz, R.: JASMINE: Java Application Sharing in Multiuser Interactive Environment. IDMS2000 (October, 2000).

3. Ma, J., Shizuka, M., Lee, J., Huang, R.: A P2P Groupware System with Decentralized Topology for Supporting Synchronous Collaborations. In Proceedings of the International Conference on Cyber Worlds (CW'03), Singapore (December, 2003).

4. Antoniu, G., Hatcher, P., Jan, M., Noblet, D. A: Performance Evaluation of JXTA Communication Layers. In Proceedings of Workshop on Global and Peer-to-Peer Computing (GP2PC 2005) held in conjunction with the 5th IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGRID 2005), IEEE TFCC., Cardiff, UK (May, 2005).

5. Halepovic, E., Deters, R.: The costs of using JXTA. Proceedings of third International Conference on Peer-to-Peer Computing, pp. 160 – 167 (September 1-3, 2003).

6. Jere, A., Meža, M., Marušič, B., Dobravec, Š., Finkšt, T., Tasič, J. F.: Peer to Peer Search Engine and Collaboration Platform Based on JXTA Protocol. The IEEE EUROCON on Computer as a tool, vol. A, pp. 256 - 260 (September 22-24, 2003).

7. Gong, F.: Multipoint Audio and Video Control For Packet-Based Multimedia Conferencing, Proceedings of the second ACM international conference on Multimedia, (October, 1994).
8. Seigneur, J. M., Biegel, G., Jensen, C. D.: P2P with JXTA-Java pipes, Proceedings of the 2nd international conference on Principles and practice of programming in Java, pp. 207 – 212, Kilkenny City, Ireland (June, 2003).

9. Cadiz, J., Balachandran, A., Sanocki, E., Gupta, A., Grudin, J., Jancke, G.: Distance Learning Through Distributed Collaborative Video Viewing. ACM CSCW'00, Philadelphia (December, 2000).

10. Kangassalo, M., Hietala, P., Ovaska, S.: Electronic whiteboard in kindergarten: opportunities and requirements. Proceeding of the 2003 conference on Interaction design and children, pp. 15-22 (2003).

11. Margaritis, M., Fidas, C., Avouris, N., Komis, V.: A peer-to-peer architecture for synchronous collaboration over low-bandwidth networks. In K. Margaritis, I Pitas (edn.) Proceedings of 9th PCI 2003, pp. 231-242, Thessaloniki (November 2003).

12. Tsuchiya, T., Yoshinaga, H., Koyanagi, K.: STARCast: Streaming Collaboration Architecture on Heterogeneous Environment Everywhere, Proceedings of the 2004 ACM workshop on Next-generation residential broadband challenges, pp. 57 – 62, (October 2004).

13. Dridi, F., Neumann, G.: How to implement Web-based Groupware Systems based on WebDAV, Proceedings of the 8th Workshop on Enabling Technologies on Infrastructure for Collaborative Enterprises, pp. 114 – 119 (June 1999).

14. Halepovic, E., Deters, R., Traversat, B.: Performance Evaluation of JXTA Rendezvous. International Conference on Distributed Objects and Applications (DOA), pp. 1125-1142, Agia Napa, Cyprus (October 25-29, 2004).

15. Kawashima, T., Ma, J.: TOMSCOP -A Synchronous P2P Collaboration Platform over JXTA, IEEE CS proceeding of the International Workshop on Multimedia Network Systems and Applications (MNSA'2004), in conjunction with The 24th International Conference on Distributed Computing Systems (ICDCS-2004), Tokyo, Japan (March, 2004).

16. Nakamura, M., Ma, J., Chiba, K., Shizuka, M., Miyoshi, Y.: Design and Implementation of a P2P Shared Web Browser Using JXTA. IEEE CS Proceedings of the International Conference on Advanced Information Network and Applications (AINA'03), pp.111-116, Xi'an (March, 2003).

17. Wahab, H. A., Kim, O., Kabore, P., Favreau, J. P.: Java-based Multimedia Collaboration and Application Sharing Environment. CFIP '99 (April, 1999).

18. Wahab, H. A., Kabore, P., Kim, O., Favreau, J. P. Replication Management of Application Sharing for Multimedia Conferencing and Collaboration. IFIP/IEEE Management of Multimedia Networks and Services (November, 1998).

19. Wahab, H. A., Kvande, B., Nanjangud, S.: Using Java for Multimedia Collaborative Applications. PROMS'96 (October, 1996).

20. Wahab, H. A., Kvande, B., Kim, O., Favreau, J. P.: An Internet Collaborative Environment for Sharing Java Application. IEEE FTDCS '97 (October, 1997).

21. JAVA SPECIFICATION: http://java.sun.com. Last visited: April 2005.

22. Kim, O., Kabore, P., Favreau, J. P., Wahab, H. A.: Issues in Platform-Independent Support for Multimedia Desktop Conferencing and Application Sharing. IFIP HPN'97 (April 1997).

23. Shirmohammadi, S., and Georganas, N. D.: JETS: Java-Enabled TeleCollaboration System. Proceedings of IEEE Multimedia Systems'97, Ottawa (June, 1997).

24. Shirmohammadi, S., Oliveira, J.C., Georganas, N.D.: Java-based Multimedia Collaboration: Approaches and Issues. International Conference on Telecommunications (ICT'98), Chalkidiki, Greece (June, 1998).

25. Dorohonceanu, B., Marsic. I.: A Desktop Design for Synchronous Collaboration. Graphics Interface `99, pp. 27-35 (June, 1999).

26. Dorohonceanu, B., Sletterink, B., Marsic, I.: A Novel User Interface for Group Collaboration. IEEE System Sciences (January, 2000).

27. MPEG-4 VIDEO FOR JAVA MEDIA FRAMEWORK: http://www.alphaworks.ibm.com/tech/mpeg-4. Last visited: February 2005.

28. Wilcox, J.: Videoconferencing, The Whole Picture. Telecom Books, N.Y., ISBN 1-57820-054-7 (2000).

29. Gordon, R., Talley, S.: Essential JMF: Java Media Framework. Prentice-Hall, Englewood Cliffs, N.J (1999).

30. P2P ARCHITECTURE: http://www.atc.gr/p2p_architect/results/0309F05_ReferenceArch.pdf. Last visited: April 2005.

31. MICROSOFT NetMeeting: http://www.microsoft.com/windows/netmeeting/. Last visited: January 2005.

32. IBM LOTUS SAMETIME: http://www.lotus.com/products/product3.nsf/wdocs/Homepage. Last visited: October 2004.

33. SAMETIME DEMO: http://stdemo3.dfw.ibm.com/. Last visited: April, 2005.

34. Grudin, J.: Computer-Supported Cooperative Work. IEEE Computer (1994).

35. Steinmetz, R., Nahrsted, K.: Multimedia: Computing, Communications and Applications. Prentice Hall PTR, Upper Saddle River, NJ 07458 (1995).

36. Barkai, D.: Peer-to-Peer Computing: Technologies for Sharing and Collaborating on the Net. INTEL PRESS (2001).

37. Leuf, B.: Peer To Peer: Collaboration and Sharing over the Internet. Addison-Wesley, (2002).

38. Edwards, J.: Peer-to-Peer Programming on Groove. Addison-Wesley (2002).

39. MAGI: http://www.endeavors.com/. Last visited : January 2005.

40. PROJECT JXTA: http://www.jxta.org/. Last visited: November 2004.

41. INTEL PROSHARE: http://www.intel.com/support/proshare/. Last visited: September 2004.

42. Shiah, C. W., Chen, W. C.: A Generic Shared Window Architecture and Some Issues. Department of Computer Science and Information Engineering, National Taiwan University.

43. Bhandarkar, P.: Replicating Distributed Events for Real-Time Collaboration. Special Problems Research report. Department of Electrical and Computer Engineering. Rutgers University (1998).

44. Suthers, D. D.: Architectures for Computer Supported Collaborative Learning. The IEEE International Conference on Advanced Learning Technologies (ICALT) (August, 2001).

45. Twidale, M. B., Nichols, D. M., Paice, C. D.: Browsing is a collaborative Process. Information Process & Management, vol. 33, no. 6, pp. 761-83 (1997).

46. MSN MESSENGER: http://messenger.msn.com/. Last visited: July 2004.

47. VIRTUAL NETWORK COMPUTING: http://www.realvnc.com/. Last visited: October 2004.

48. CITRIX GOTOMEETING: https://www.gotomeeting.com/. Last visited: October 2004.

49. JNLP SPECIFICATION: http://java.sun.com/products/javawebstart/1.2/docs/Javadoc /index.html. Last visited: February 2005.

50. Baldi, M., Ofek, Y.: End-to-End Delay Analysis of Videoconferencing Over Packet-switched Networks. IEEE/ACM Transactions on Networking, vol. 8, no. 4, pp. 479-492 (August, 2000).

51. JXTA SOCKETS: http://p2psockets.jxta.org/. Last visited: April 2005.

52. ECLIPSE SWT: http://www.eclipse.org/swt/. Last visited: April 2005.