Pertanika J. Sci. & Technol. Supplement 9(1): 87 - 95 (2001)
ISSN: 0128-7680

A Java Based Multimedia Distributed Computer Supported Collaborated World (CSCW) Environment Over the Internet

Chee Boon Kok, Malay R. Mukerjee, Borhanuddin Mohd Ali & Abdul Rahman Ramli
Department of Computer & Communication System Engg
Universiti Putra Malaysia, 43400 UPM Serdang, Selangor, Malaysia
Email: cbkok@cae.upm.edu.my

ABSTRAK

Computer Supported Collaborative Work (CSCW) and distributed computing have been termed as the next paradigm of computing which allows users to interact and collaborate with each other seamlessly over the network. As a result, many collaborative distributed CSCW system have been proposed and developed. However, most of the system fall short in providing platform independent system that address many real-world needs of users working together in a CSCW system. In this paper, efforts have been focused on creating a Distributed Computer Supported Collaborative Work environment for efficient collaborative editing and interaction among the participating users over the internet that support asynchronous and synchronous collaboration. Java has been used developed language.

Keyword: Distributed, Collaboration, CSCW, Java

INTRODUCTION

CSCW is the study of how people may work together using computer technology to facilitate human interaction for problem solving. It looks at the way people interact and collaborate with each other, and attempts to suggest guidelines for developing technology to assist in the communication process. The advance of networking and distributed technology like CORBA[1], DCOM[2], Voyager[3] and RMI[4] has made the CSCW concepts an ideal and viable solution for improving productivity and innovation. People can now communicate, interact, collaborate, search or send information with just a simple mouse click regardless of location and time. Internet and Java™[5] technology further accelerate the adoption of computer technology in research and industry.

This trend might be extended to a flexible-working trend in the future. In contrast to the traditional working style (with employees actually clocking in and off of shifts controlled by factory klaxons), future employees might come and go within certain hours, and would be free to organize their time as long as they meet the deadlines. In another scenario, employees might just work from their home, eliminating the need to commute to the office and meet their colleagues face to face.

CSCW system plays an important role on the success of teamwork and flexible-working concepts. CSCW system will assist groups in communicating, collaborating and coordinating their activities. While the idea of supporting these activities is not new, the realization of any form of computer support has been more difficult than most would

1. OMG Home page CORBA http://www.omg.org/
2. Microsoft Distributed Component Object Model (DCOM). http://www.microsoft. com/com/dcom.asp
3. Object Space Voyager. H=http://www.objectspace.com/products/voyager/index.html
4. JavaTM Remote Method Invocation. http//java.sun.com/products/jdk/rmi/index.html
5. JavaTM Technology Home. Http://java.sun.com/
6. Flexible JAMM (Java Applet Made Multiuser). http://simon.cs.vt.edu/jamm
7. Tango2. TANGO 2 Documentation." http://trurl/krzysztof/tangodoc/

have anticipated. This is because of the diverse nature of task undertaken by groups, diffrerent styles of group working, and a host of behavioral factors which are difficult to overcome. The challenge to the CSCW designer will be to come out with a solution to solve different problem domains of a workgroup environment. Recent (Shirmohammadi & Georgamas 1997), projects like JAMM[6], Tango[7] and JETS by MCRLab of University of Ottawa[8], show various approaches utilizing different techniques and models to create CSCW system. Java Distributed Collaborative Environment (JDCE) is an attempt to offer different solution for solving problem in a general system.

COLLABORATION MODEL

Asynchronous Collaboration

In asynchronous collaboration, the interaction between users happens regardless of the time factor. Users are not required to be online simultaneously for collaboration. Users can log on to the system at different times but still can collaborate on the shared data as long as the collaboration system is running. There is also no strict timing requiremet for data delivery. Users can log in into the system and modify the shared data at any time while the system will maintain the consistency of the data. The system updates the modified data to other users at later times when they log in into the system again. It is the main responsibility of the asynchronous system to make sure that all the users get the updated shared data. In this case, the reliability of the data delivered becomes the key factor of asynchronous system. Example of asynchronous collaboration system is email, Network News Transfer Protocol (NNTP) (Kantor & Lapsley 1986).

Synchronous Collaboration

In synchronous collaboration, users must be online simultaneously for collabration. A message sent from one users is immediately sent to all others in the same collaboration session. There are two general approaches to provide awareness for synchronous collaboration. One is the sharing of legacy single-users applications. This is referred to as collaboration transparency because the sharing is provided by a mechanism that is unknown, or transparent to the advantage of allowing the users to utilize legacy application that they are familiar with and, thus increases the productivity of the users. Several collaboration transparency systems that have been implemented include Old Dominion University's XTV(Abdel - Wahab and Feit 1991), Sun Microsystem' ShowMe SharedApp(1994) and Microsoft's Netmeeting (1998).

The second approach, known as collaboration awareness, is where the application are specifically developed to support cooperative work. These applications may be developed ad hoc or with a groupware toolkit such as the University of Calgary's GroupKit (Roseman & Greenberg 1996) and Old Dominion University's Java Collaborator Toolset (JCT) (Abdel - Wahab *et al.* 1996). This approach offers the freedom of building more visual awareness to the applications targeting special collaborative activities. This type of synchronous collaboration model has been adopted in JDCE.

Hybrid Collaboration Model

JDCE is a CSCW system that implements the hybrid model were both the asynchronous and synchronous model are supported. This provides the maximum flexibility for users working in this environment.

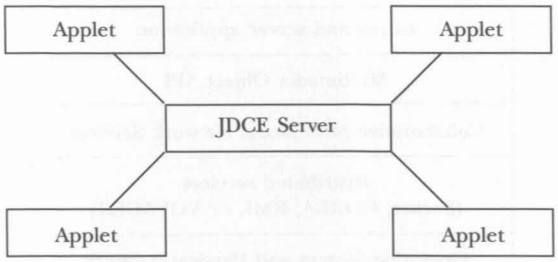

Fig 1. JDCE client-server architecture

NETWORK ARCHITECTURE

In a collaborative environment, the resources and users are normally distributed over a vast network. The system needs to provide the communication channels for efficient interaction between users and the system. Various architectures requirements of a particular system and application. Generally, collaboration architecture can be divided into three main models:

- Client-server architecture - centralized model
- Server network architecture - partially distributed model
- Peer-to-peer architecture - fully distributed model

JDCE has been implemented using the client-server architecture since the client of JDCE is built as Java applet where it can only connect to the originating server. Fig. 1 shows the diagram of client-server architecture implemented in JDCE.

A central points of communication makes it easy for message routing and state management. Access to the system can be controlled easily and different floor control can be applied for effective collaboration. This architecture also allows the implementation of shared object that are accessible by all the users. Shared object helps in keeping the consistency of the application shared-data state. The server control access to the shared object through an object looking mechanism. A user who wants to change the share object state must look at the shared object first, perform the modification and release the lock. Other users are only allowed to modify the shared object when the lock on the object is released. This approach is also good for latecomers who can get the latest shared object states which are synchronized with other users in the same collaboration session.

SYSTEM ARCHITECTURE

This environment has been built using an object-oriented layering architecture API. The system layering architecture is presented as illustrated in Fig. 2. The purpose of using the layering architecture is to partition the system into layers responsible for different tasks. A clear interface is defined between layers, which provide a public protocol for communication between them. Thus, changes in one layer will not have any effect on other layer.

At the top layer, there exist various multimedia application like video conference, electronic notebook, shared whiteboard and distance learning applications. These applications use the services of Multimedia Object API.

The second layer is a Multimedia Object API. This layer has been constructed using object-oriented approach and consists of object classes that can be used to generate,

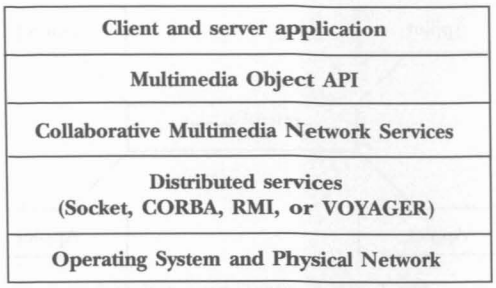

Fig 2. Object-oriented layering architecture API

display and modify the content of multimedia data. Example of the multimedia object classes are text, graphics, audio and video object classes. Most of the new Java Media APIs have been incorporated into this layer. An open approach is used to accommodate future media that are not supported now.

The third layer consists of distributed multimedia services that are needed to support the operation and management of distributed multimedia applications. This layer is further divided into different subsystem that provide different services to the multimedia applications. The subsystem are Communication Service, Session Service, Naming Service and Management Service.

The fourth layer offers distributed network services. Currently, Java socket has been used as the vehicle of the distributed services. However, these services can be easily extended to include the support for CORBA, Java RMI or voyager distributed system.

SYSTEM INDEPENDENT COLLABORATION PROTOCOL (SICAP)

The performance of this environment has been improved by the introduction of a system-independent Collaboration Access Protocol (SICAP). SICAP is a high level protocol. Fig. 3 illustrates the communication between SICAP and other distributed services like CORBA, RMI, Voyager and network protocols.

The key design objective of SICAP is to increase the efficiency, utilization and performance of client-server communication by reducing the bandwidth usage and yet offer a wide range of functions and facilities that are needed for distributed collaborative

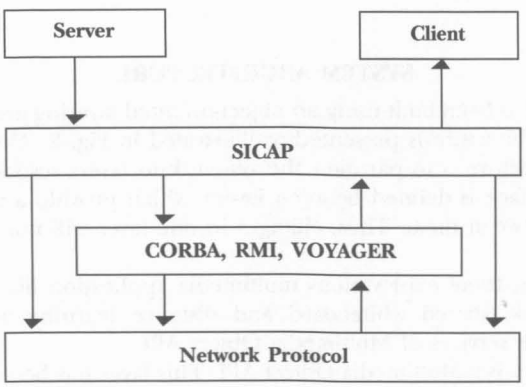

Fig 3. Communication between SICAP, client and server

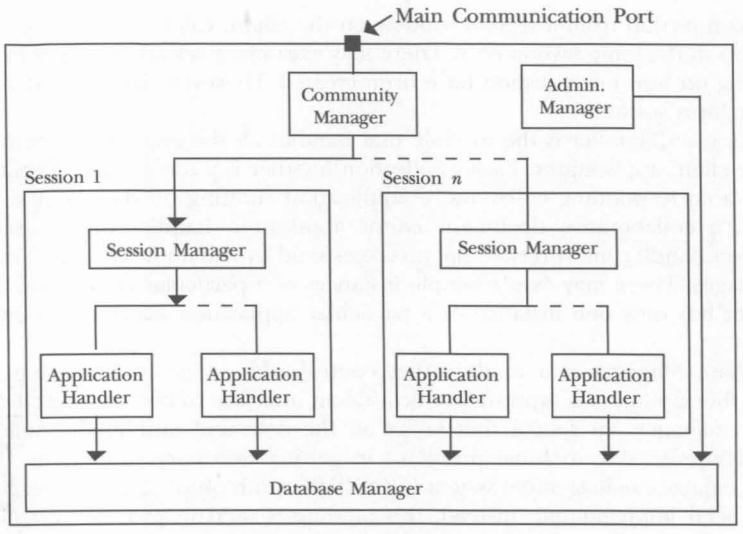

Fig 4. Communication between SICAP, client and server

environments. More detail description about SICAP are available in (Boon Kok *et al.* 1998).

IMPLEMENTATION

JDCE Server

The JDCE server is the main component of the JDCE system. The JDCE server consists of several modules. Fig. 4 illustrates the general structure of the server and relationship between them.

Community Manager is a module that controls the operation of the whole server. It accepts messages from a client and responds to the client's request through a well known port number. This module has direct control of the session manager modules. It keeps a lists of all active sessions currently in the system. Actions such as joining or leaving a session and session creation are handled by the community manager module. Any changes in the system state are broadcast to all members in the community. The module is also responsible for terminating the session when there is no users left in the session. It communicates with the administration manager to receive configuration setting and system data. Only one instance of such a module exists in running server.

Administration Manager is a module that provides the functionality for system administrator to change and set up the configuration of the server. In the current implementation, only the system administrator has direct acces to this module. This is to protect any changes to the system level state by unauthorized users. As with the community manager, each running server has only one instance of the administration manager module.

Session manager is a module responsible for controlling and providing services for all activities in a session. Each session created is associated with one instance of this module. Each module keeps a list of application handler that can be referred by the client applications. This module accepts control messages from the community manager directly and its controls all the application handler running under the same session. It

accepts connection from a session window on the client. Client messages are broadcast to all users in the same session only. There may exist many session manager in the server depending on how many session have been created. However, there is only one session manager for a session.

Application Handler is the module that handles all the application event messages from the client applications. Each application handler is a unique implementation that matches a corresponding collaborative application running on the client side. For an example, a collaborative document editor application handler on the server. This application handler interprets all the messages send by this application and responds to the messages. There may exist multiple instances of a particular application handler in the server but only one instance of a particular application handler can exist in one session.

Database Manager is a module that controls the access to the system database. Initially, this module was supposed to provide an interface to control an object-oriented database management system that keeps all the data and multimedia objects in the system. However, due to some problems in getting necessary equipment, the object-oriented database management system (OODBMS) (with object query language support) has not been implemented. Instead, this module is used to provide persistent object services to store and retrieve the multimedia object and data. A simple object-oriented database management system (without support for object query language) is implemented partially and is controlled by this module. All application handler send request to the database manager directly for getting any multimedia object. The administration manager accesses the database manager directly when updating the system database.

The server functions are discussed in (Boon Kok *et al.* 1998a) and (Boon Kok *et al.* 1998b). A snapshot of the running server are shown in Fig. 5.

JDCE Client

The JDCE client consists of three important components for the proper operation of the client; Mainapplet, Session Window and Collaborative Application. Fig. 6 shows the three components of the client and relationship between them.

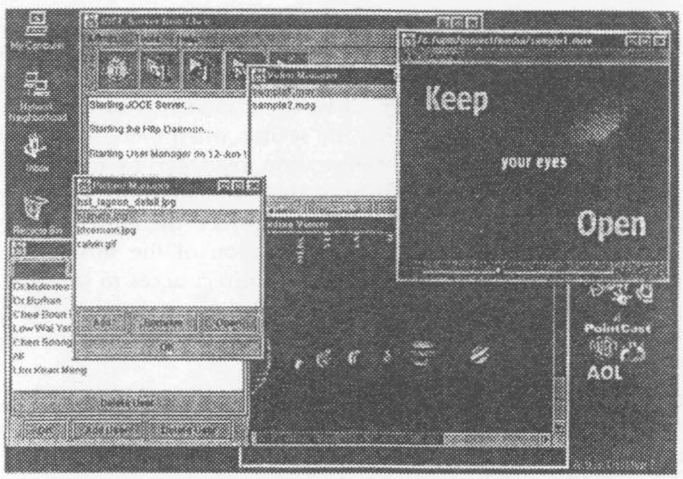

Fig 5. JDCE server in action

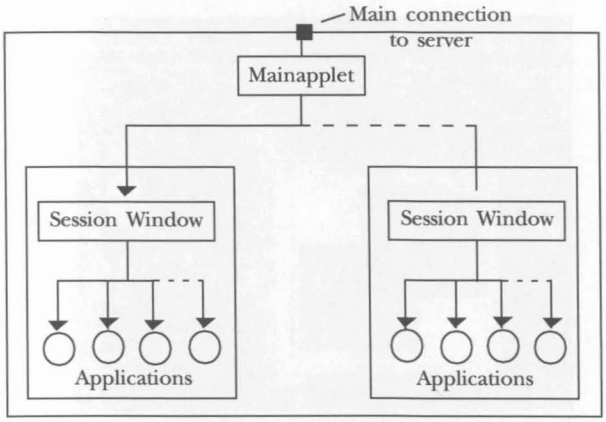

Fig 6. General structure of JDCE client

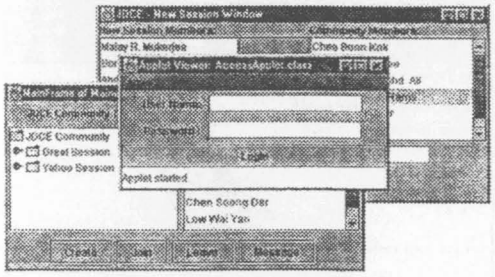

Fig 7. Mainapplet in action

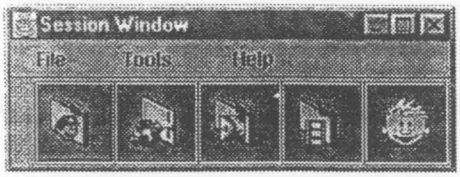

Fig 8. Session Window of JDCE client

The Mainapplet is the main component of the JDCE client. Mainapplet is the central control for the JDCE client. This component provides the necessary user interface for a user to log in remotely to the server. This component communicates with the server directly to establish a connection for control message distribution. It is also displays all running session in the server and all the active users in the system. Any changes to the system state are send to Mainapplet by the server. This component controls the creation of Session Window and passing the necessary data to the Session Window like port number of the corresponding session manager on the server. There is only one instance of Mainapplet in every running client. Mainapplet screen shot is shown in Fig. 7.

The Session Window is a component that groups a set of collaborative application that are available for a particular session. When a user join a particular session, this

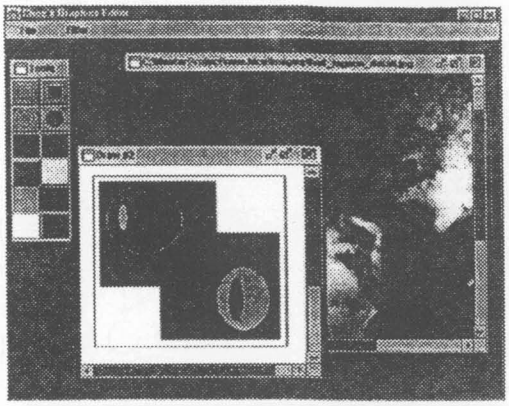

Fig 9. Graphics editor in action

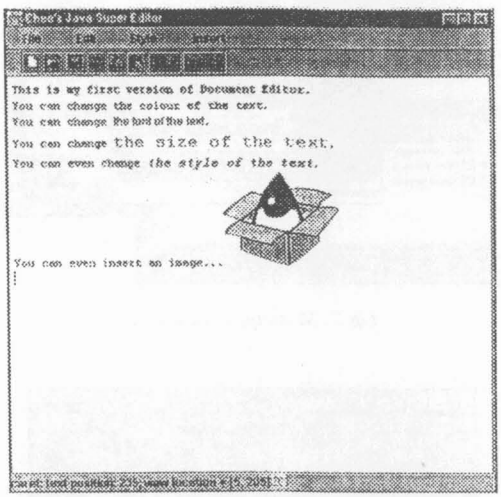

Fig 10. Word editor in action

window is created dynamically by the Mainapplet. When created, this component opens a connection to the corresponding session manager and passes it to the right application ID that comes with the message. The Session Window is shown in Fig. 8.

The Collaborative Application is a tool that provides facility for collaborative work. A set applications which provides different functionalities has been included in each session to help the users to finish their tasks. These application can be activated through the Session Window. Any application event message received from the Session Window. JDCE provides various collaborative applications such as chat client, graphics editor and word editor. Figs. 9 and 10 show snapshot of the graphics editor and word editor respectively.

CONCLUSION

A new multimedia distributed CSCW enviroment has been presented which permits workers in different physical location to work together on a multimedia document using internet resources. The set-up of user interface is an improvement in earlier proposals in that a reduction in bandwidth has been achieved, more collaborative- aware user interface has been designed and text and graphics editing is possible as a collaborated activity.

Though the system has been developed using Java, it is flexible enough to incorporated different kind of existing user environment. The effectiveness of the operation with CORBA and similar ORB environments is still under investigation.

ACKNOWLEDGEMENT

The authors would like to acknowledge the research support provided by the University Putra Malaysia through financial support to the first author and the use of existing facilities.

REFERENCES

ABDEL-WAHAB H. and M. FEIT. 1991. XTV: A Framework for X Window Clients in Remote Synchronous Collaboration. In *Proc. of IEEE TriComm:* 159-167.

ABDEL-WAHAB H., B. KVANDE S. NANJANGUD, O. KIM and J. P. FAVREAU. 1996. Using Java for Multimedia Collaborative Applications. In *Proc. of the 3rd Int. Workshop on Protocols for Multimedia Systems (PROMS'96),* October 1996, Madrid.

CHEE BOON KOK, MALAY R. MUKERJEE, BORHANUDDIN MOHD. ALI and ABDUL RAHMAN RAMLI,"SICAP: A system- independent Collaborative Access Protocol For Distributed Collaborative Environment," Proc ICIMU 98, UNITEN September 1998.

CHEE BOON KOK, MALAY R. MUKERJEE, BORHANUDDIN MOHD. ALI and ABDUL RAHMAN RAMLI. A Java Based Object-Oriented Multimedia Distributed Collaborative Environment Over the Internet. In *Proc. NCTT'98,* UPM, December 1998.

KANTOR B. and PHIL LAPSLEY. 1986. Network News Transfer Protocol: A Proposed Standard for the Stream- Based Transmission of News. RFC 977, U.C. San Diego and U.C Berkeley.

Microsoft Net Meeting. 1998. Microsoft. http://www.microsoft.com/netmeeting/

ROSEMAN M. and S. GREENBERG. 1996. Building real time groupware with groupKit, a groupware toolkit. Transaction on computer interaction. *Computer Human Interaction.* **3(1)**: 66-106.

SHIRMOHAMMADI, SHERVIN and NICOLAS D. GEORGANAS. 1997. JETS: a Java-Enable Telecollaboration System. 541-547. *Proc. IEEE Conference of Multimedia Computer and System* (ICMCS'97), Ottawa.

The Complete Guide to ShowMe 2.0.1. Sun Microsystem Inc., Mountain View, CA. 1994. http://www.sun.com/product-n-solution/sw/ShowMe/product/ShowMe_SharedApp.html.