December 1997

An X-window-based Distributed Application for Cooperative Work

D. Hoang
University of Sydney

An X-window-based Distributed Application for Cooperative Work

D. B. Hoang
Basser Department of Computer Science, University of Sydney, NSW 2006
doan@cs.usyd.edu.au

Practitioner-oriented Executive Summary
This paper presents the design and implementation of OpenBoard Tutorial System. OpenBoard is a client/server distributed application that supports concurrent meetings among users of X-window-based workstations. It manages "boards" which are windows that users at various workstations can write or draw on simultaneously, with each seeing an up-to-date image of the board's state on his/her own screen. The OpenBoard is built on top of the TCP/IP protocol suite, it uses SUN RPC to support interactions between server and clients; and it allows stored images to be broadcast between users. The OpenBoard Tutorial System (OTS) is an application structured around the OpenBoard. The OpenBoard Tutorial System implements additional mechanisms which allow users to conducting real-time tutorial sessions effectively in a distributed environment.

The system has been evaluated by several groups of students. One such group consists of four evaluators who evaluated the system by commenting on its usability and interface. They had no prior knowledge on how to use the system but knew what it is supposed to do. The evaluators all thought that OpenBoard had a nice look to it, and they found the interface easy to use. They had no problems in drawing objects onto the screen. The evaluators were most impressed with the feature which allows users to display and send stored images to different clients and they had no problem using it. However some features are necessary but not intuitive as expected. For example, the DrawQueue feature which is required to maintain the order of the session, but was not strongly appreciated by students involved in the evaluation. Other features suggested include a useful pointing device to draw attention to certain object; a help button; the ability to enter text on the drawing area, and the ability to draw freehand.

Overall, the paper presents the design and implementation of a very useful computer-supported tutoring system: the OpenBoard Tutoring System. The system performs very well in terms of functionality as well as in terms of fast response time. Its availability suggests a feasible, powerful, alternative style of tutoring which becomes increasingly important in a distributed learning environment. Furthermore, the OpenBoard system can easily be adapted for other applications including conference, design teamwork, etc. The application can also be extended for real-time multimedia applications.

Abstract
The paper presents the design and implementation of a distributed application called OpenBoard Tutorial System (OTS) that supports cooperative work among users of window-based workstations. The application is based on the X window system and the SUN RPC protocol. The paper describes OTS as a system designed for conducting real-time tutorial in a distributed environment by allowing students to share text, graphics and stored images, and tutor to facilitate the session.

1. Introduction

Cooperation and coordination are essential to accomplished a complex task that requires participation of many individuals. Face-to-face meeting is an efficient way to achieve such cooperation and coordination. One major drawback of this type of meeting is that it requires participants to be present at the meeting physically. This requirement is sometime impossible to meet and in many cases not desirable. Participants cannot attend several meetings simultaneously. Another drawback is due to the use of traditional media like chalkboards for carrying out activities in the meeting. It is difficult to rearrange, store, and keep records of the meeting with the chalkboard. Furthermore, the only information that can be shared at a face-to-face meeting is that which was brought to the meeting or composed during the meeting. Participants cannot access the increasing multitude of private, corporate, and public databases to satisfy unanticipated information needs.

Computer-supported meetings can overcome many of the problems associated with conventional face-to-face meetings. Computer-supported meeting systems can be employed in many different applications, from simple one-way seminar presentation, unstructured multiparty communications,

distributed cooperative design work, to more structured conference (Ahuja, Ensor, and Horn 1988; Stefik et al. 1987; Goscinski 1993).

This paper presents the design and implementation of OpenBoard Tutorial System. OpenBoard is a distributed application that supports concurrent meetings among users of X-window-based workstations. It manages "boards" which are windows that users at various workstations can write or draw on simultaneously, with each seeing an up-to-date image of the board's state on his/her own screen. The application supports concurrent meetings by allowing users to concurrently join several different boards and communicates via them in text and drawings. An earlier version of OpenBoard was developed by Liu and Hoang (1994). The early version suffered a number of drawbacks: 1) it was implemented mainly to test the implementation of the OSI Remote Procedure Call Protocol by Liu and Hoang (1997); 2) the system response time was slow since it is built on top of the 7-layer OSI protocol suit; 3) it did not support the exchange of stored images between users; 4) it only supported unstructured communications, i.e., the board is free for all , no restriction or coordination is imposed on users.

This paper also describes the OpenBoard Tutorial System which is based on a new version of OpenBoard. The new OpenBoard is implemented 1) using SUN RPC; 2) built on top of the TCP/IP protocol suite; 3) allowing stored images to be broadcast between users. The OpenBoard Tutorial System implemented additional mechanisms which allow users to conducting real-time tutorial sessions in a distributed environment.

The paper is organized as follows. Section 2 gives a brief description of how OpenBoard operates. Section 3 presents the design and section 4 presents the implementation of the OpenBoard Tutorial System. Conclusion is in section 5.

2. OpenBoard Operation

The OpenBoard system consists of a client program and a server program. The server acts as a manager to manage boards for clients. The clients send requests to the server to open, join, or close boards and to draw on the opened boards. The server program is started as a background process. A user wishing to participate in a meeting executes a copy of the client program. If a user wants to join several meetings simultaneously, he/she has to run several copies of the client programs as one instance of the client program only provides a single board (window) for a user to carry out discussion in a particular meeting. The client program uses the X-window system to provide a graphical user interface (GUI) to the users of OpenBoard so that they can write and draw on the windows, they can also broadcast stored images to all other users. The server program, however, requires nothing from the X-window system. The interactions between clients and the server are supported by the SUN RPC system, which provides OpenBoard with a convenient interface to the underlying communication system.

When a user wants to open or join a board, the client program sends the server an RPC request with the board name as a parameter. If the board exists, the server adds the user to the board's user list and uses a callback to send a copy of the board's current state to the client program, which displays an image of the board on the user's display. Otherwise, the server creates a new board with an initial state, establishes an initial board's user list and sends a callback to inform the client program to display an image of a blank board. When a user writes or draws something on a board, the client program first updates the local image on the user's display and then sends the changes in an RPC request to the server. The server will update the state of that board and send the new state to each client registered with that board using a callback. The clients then update their boards.

The OpenBoard Tutorial System utilizes fully the OpenBoard facilities and adds necessary session management mechanisms so that the students can request for participation and the tutor can organize the discussion in the most effective manner.

3. Design

The OpenBoard Tutorial System is designed to meet the following requirements:

- A real-time tutorial can be conducted in an orderly manner with a tutor having rights to manage the tutorial,
- Users can joint or leave the tutorial subject to certain access control,

- Users can create and modify text and graphic documents,
- Users can communicate stored images to all participants,
- All users see identical shared document on their screen,
- Each user can have his/her on private view of the document besides/addition to the common shared information.

There are three major aspects in designing the OpenBoard Tutorial System. The first aspect is the interaction with the X-windows environment to provide a graphical user interface for text, drawing and image communications. The second aspect is the session management which allows a tutor to control and organizes the distributed tutorial session. The third aspect deals with a mechanism for users to communicate with each other.

3.1 Graphical User Interface

A user interface is more than just a graphical display. It is a medium for dialogue between users and computers. It is important to center the design around user attitudes, and provide them with an environment representing the reality of their conceptual model. In the last decade, the state of the art moved from simple command text interfaces to interactive visual interfaces. Consequently, the user interaction style has changed from conversational to direct manipulation of graphical objects.

A number of principles has been observed in designing the OpenBoard Tutorial System:

- The interface has to be simple. The tendency is to provide as many features as possible for the application. However, experiences showed that doing so would make the interface complicated and inevitably has negative effects on the users.
- The interface must be easy to use. This simply means that if the users find it hard to manipulate or "complicated" and they will not use it.
- The interface should be efficient in the sense that it can be supported adequately in terms of response time and reliability by the underlying distributed communications mechanism.
- The interface must take into account the users' needs for the particular application. In the context of a tutorial system, the tutor must be able to manage and control and facilitate the tutorial, and the students must be able to explore their own idea as well as participate in the discussion without overly constrained.

The main constraint is the limited amount of space available for the OTS window. This constraint translates directly into the number of graphical features that can be simultaneously visible on the interface without cluttering the window.

It was determined that the most intuitive graphical WYSIWYG (what you see is what you get) interface be adopted. The interface uses a layering technique, similar to CAD applications, to allow shared view as distinct from a user's private view of the document.

The overall interface has three main areas: the drawing window with its buttons, the text window and the draw queue areas and session management buttons (Figure 1). The drawing window follows the conventions of many drawing programs. The file menu, edit menu and the other menus are placed at the top of the window, the drawing tools are represented as icons on the left side of the window, and the drawing area is in the center of the window. The text window is below the drawing area. The draw queue and session management buttons are placed to the right of the drawing area. The GIF pictures are displayed in a separate dynamically sized window to reduce the clutter in the main window.

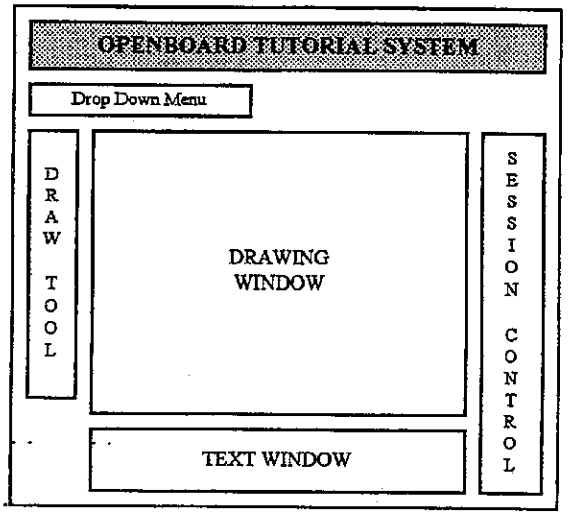

Figure 1 The OpenBoard User Interface Design

To conform with the simplicity and easy to use principles, only common shape elements of almost every drawing package are made available. These include lines, rectangles, squares, circles and ellipses.

The interface uses a layering technique, similar to CAD applications, to allow shared view as distinct from a user's private view of the document. When a shape is drawn it is set to one of three colours, depending on the current state of the user. There are three distinguishable states of objects in the drawing area, identifiable by the colour of the object. Drawing objects can appear in one of the following colours:

- **Black** - objects are permanent and visible to all users at all time.
- **Blue** - objects which belong to the student who currently has the right to draw in the drawing area, and is presenting his/her contributions to all participants. This is the main colour used to interact with other users of OpenBoard. This allows users to draw objects and to submit work they have done previously for all participants to see.
- **Red** - objects are only visible to the user who has drawn them. This allows students to explore alternative solutions and to express their own ideas freely without disturbing the main discussion. When a user's turn comes to participate in the main drawing, active red objects are changed to blue and hence they become visible to all.

An example usage scenario of this system is a tutor drawing an incomplete circuit and transforming it into a permanent black object. Each student in turn begins drawing solutions on his/her own drawing space in red. When each student has finished he/she can request to display the result to all users. When each user finishes he/she yielded control to another user.

3.2 Session Management

The OpenBoard Tutorial System (OTS) as the name suggested, is designed for conducting a distributed tutorial. The emphasis is placed on how such a tutorial can be conducted flexibly and orderly. This philosophy is reflected in our interface, where the emphasis is placed on controlling and ordering of the tutorial session. Operations implemented in the OTS can be divided into 2 groups. Group 1 consists of operations that can be performed by all participants, and group 2 consists of operations that are only available to the tutor.

Operations that can be performed by all include file management operations (New, Load, Save, Save As, Quit); drawing commands (Cut, Copy, Paste, Select, Delete, Move, Redraw); network commands (Connect, Disconnect, Who, Refresh); commands to request for submission, and commands for sending image file.

Operations that are available only to the tutor include Upgrade User, Yield User, and Remove User. These operations aim to keep the tutorial session in order and to facilitate the discussion in a fair manner.

3.3 Communication Mechanism

To handle communication between students and tutor in a distributed environment, the client/server model is adopted, using SUN Remote Procedure Call (SUN RPC). This protocol is chosen to maximize portability, and to provide better response time. Our experience in implementing the OpenBoard using OSI RPC and with the support of an OSI 7-layer communications protocol suite indicated that the system response time was unacceptably slow.

Server:

In the earlier implementation, each client periodically made a remote procedure call to the server to get the updated information such as the list of drawn objects, or the list of users on the system. This resulted in many wasted calls to the server and placed a great deal of strain on the network, even when no change has been made.

With this project, a different approach is adopted which stresses that a remote procedure call should be made only when it is necessary. This approach utilizes a mechanism known as **Callback** RPC. It allows each client to offer services on which the server can call back. Thus, when someone draws a shape or joins the tutorial, a remote procedure call is made from this client to the server passing it the new information. However, rather than all other clients calling the server to get this new information, the server actually calls these clients through their services to pass them the new information. This ensures that calls are made only after there have been changes and the clients need to be informed. Callback technique is illustrated in Figure 2.

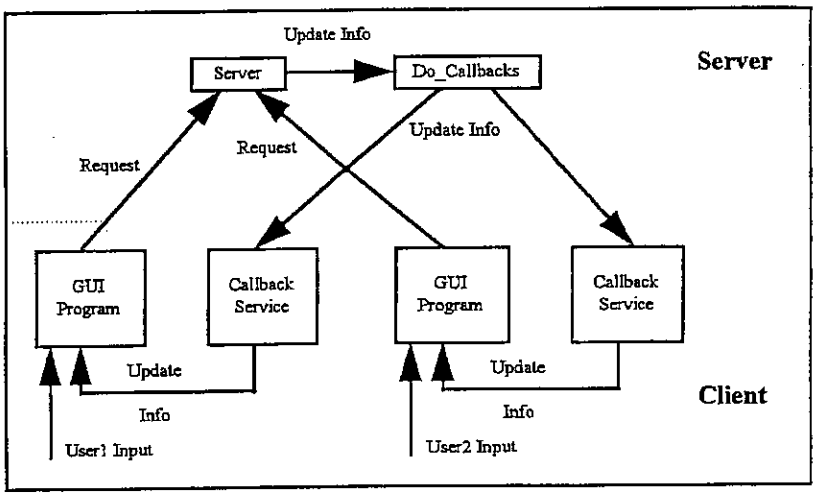

Figure 2. Callback feature of the server

Client:

The client is responsible for three basic functions: calling on the server when it wants to update the board, handling the callbacks from the server, and handling local X-Window events. Whenever the client receives a draw request either from its user (students or tutor) or from a callback, it does the drawing and updates its local drawing record list. Callbacks result from other users' requests. The client has to listen for both user input from the keyboard/mouse and input from the network for callbacks so that it will not miss requests from its users and those from other via the server.

4. Implementation

The complete interface for the OpenBoard Tutoring System is realized as shown in Figure 3.

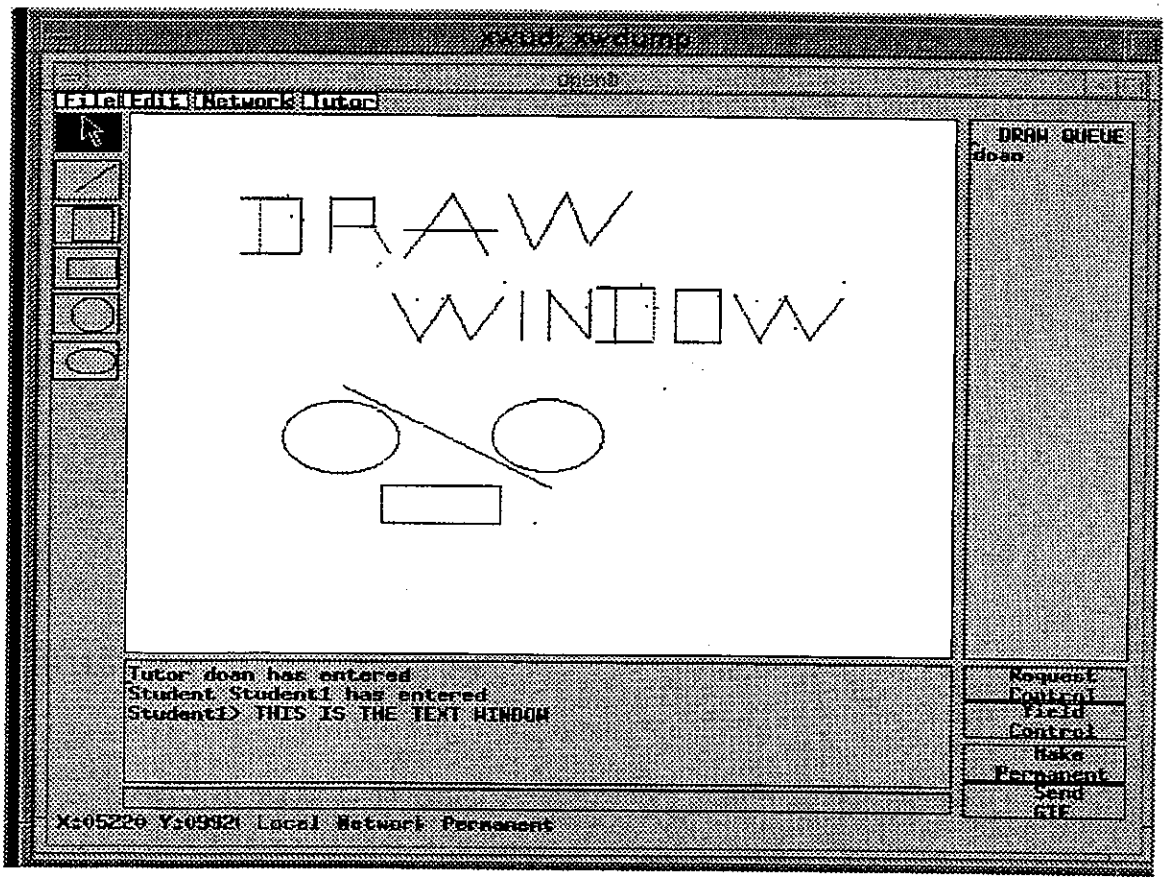

Figure 3. The OpenBoard Tutorial System user interface

In most drawing programs, when the users draw a shape, it appears immediately on the screen to show them exactly what they have drawn. Also, when drawing a shape, the shape is updated as it is being drawn (a method known as *rubber-banding*) so that the shape can be previewed as it is being drawn, and the shape can be placed precisely. These two concepts are implemented in OpenBoard.

The selection tool in OpenBoard works in a way similar to most modern drawing programs. Once selected, the user can move a shape by clicking and dragging the selected objects. When an object is selected, it displays two of its *control points* which can be dragged by the mouse, allowing the shape to be re-sized.

With the **Callback** mechanism employed, the server has to perform many tasks in a short time. It must process the new information and then calls each of the other clients to inform them of the new information. In the mean time, the server cannot service any other request from clients. This delay can be intolerable if there are many clients, since the time required to make a remote procedure call is quite significant. Furthermore, if a client has crashed, the server actually waits till the call to that client has timed out before making a call to the next client.

The solution to this problem is solved by using *thread* programming technique. Creating a thread is similar to "forking" a new process with the exception that a thread shares with its peer threads its code section, data section and operating resources such as open files and signals. The use of threads allows the server to continue with servicing incoming requests while it is making callbacks to the clients. Furthermore, each call to the client is made by creating a new thread to handle it, thus preventing delay if one of the client has crashed.

Figure 4 shows the OpenBoard interface and a pop-up window which holds the image sent by one of the users (the Figure is wrapped around and has to be fixed). The image can be positioned in any place on the workstation screen, however it is placed on top of the interface window for convenience.

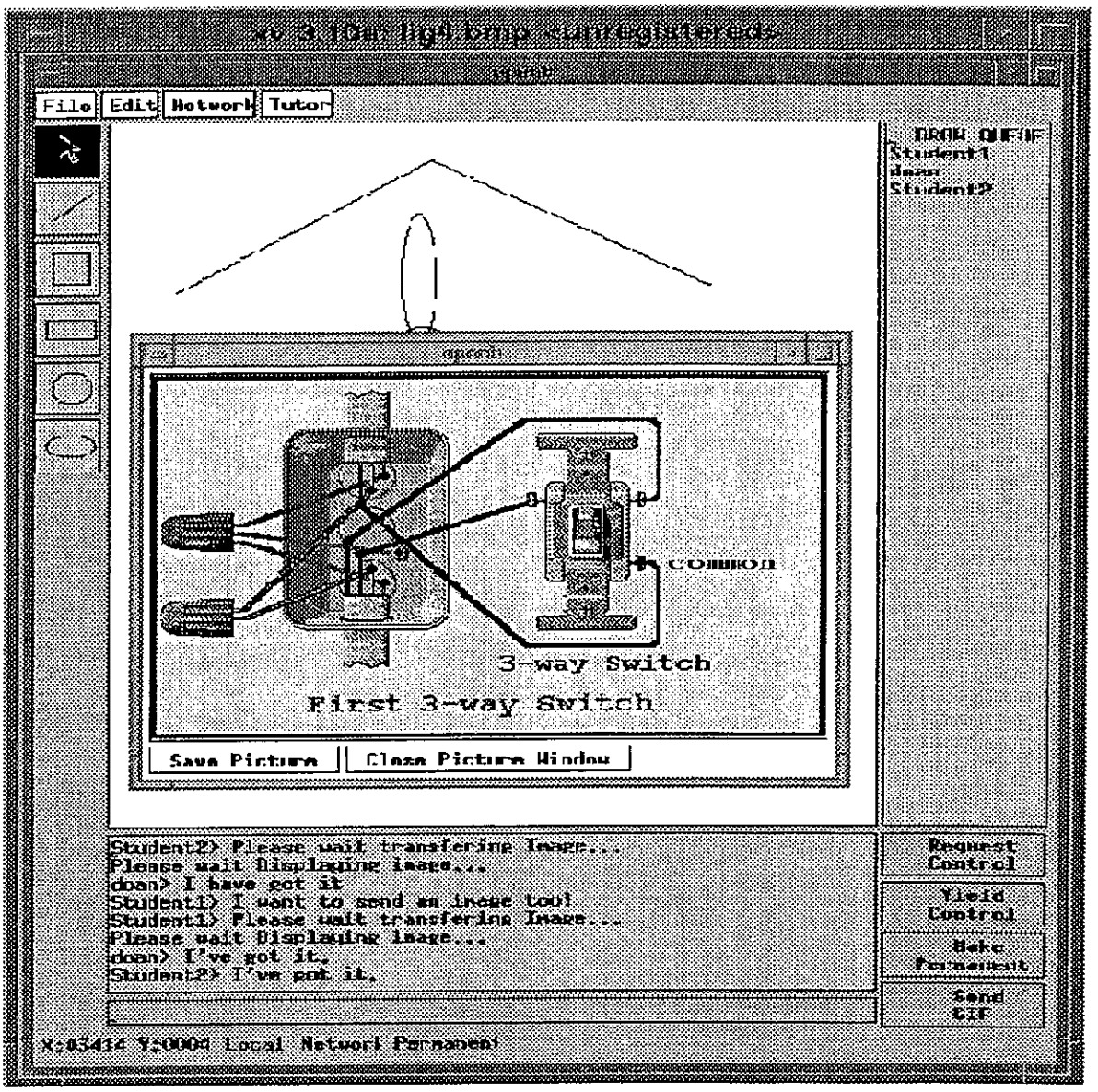

Figure 4. OpenBoard and Image capability.

On the tutorial management aspect, OTS implemented the following operations:

Operations that can be performed by all participants:

- All OpenBoard users can enter an OpenBoard session by using the "Connect" command. The user then must provide the user's name and correct password before he/she is allowed to join.
- All users participate in a discussion by typing their comments and questions in the text dialogue area.
- All participants can request to draw by placing themselves on the drawing queue.
- All participants can make any object "permanent" if they decide it is necessary and it is their turn to draw.
- When a particular participant is on the top of the drawing queue, all drawing objects that were red will change to blue and these objects will become visible to all OpenBoard users. All object drawn thereafter will also be in blue until such time the user is no longer on top of the drawing queue.
- Any user can save the drawing area or text dialogues to a file for later use and they can load a previously saved drawing for discussion.

- The "WHO" command displays a list of all participants in the tutorial.
- Any user can "Disconnect" and work locally or quit.
- A user on top of the draw queue is allowed to send a picture in GIF format to other users for discussion.

Operations that are only available to the tutor:

- The tutor has the ability to remove a student out of the tutorial if that student is disruptive.
- The tutor can place any student to the top of the drawing queue if he/she decides that a student has something important to show.
- The tutor can remove any student currently on the top of the drawing queue if the tutor feels that this student has been there for too long.

5. Discussion

The system has been evaluated by several groups of students. Ond such group consists of four evaluators who evaluated the system by commenting on its usability and interface. They had no prior knowledge on how to use the system but knew what it is supposed to do. The evaluators all thought that OpenBoard had a nice look to it. They found the interface easy to use . They had no problems in drawing objects onto the screen. However some features are necessary but not intuitive as expected. For example, the DrawQueue feature which is required to maintain the order of the session, but was not strongly appreciated by students involved in the evaluation

The evaluators were most impressed with the feature which allows users to display and send stored images to different clients and they had no problem using it.

It was found that arrows as an object would be useful to draw attention to certain object, and that different thickness of objects would also be an advantage. The most important thing which was found lacking was the ability to enter text on the drawing area, and the ability to draw freehand. The problem was that these features were hard to implement with the same level of support such as move, select, cut, etc. This is especially true for text and freehand objects, and is a major limitation of Libsx.

It is found that the response time of the system is more than adequate for the tutoring application. Some extensions were suggested include a help button feature, an arrow for drawing user's attention to a certain object.

The paper presents the design and implementation of a very useful computer-supported tutoring system: the OpenBoard Tutoring System.

Overall, the OpenBoard Tutoring System performed very well. Its availability suggests a feasible, powerful, alternative style of tutoring which becomes increasingly important in a distributed learning environment.

Acknowledgments

The project has evolved from the design and implementation of the OSI RPC through the development of an unstructured OpenBoard in an OSI communications environment, to the present Tutoring Systems in the SUN RPC - Internet environment. The author would like to acknowledge the contributions of Y. Liu and S. Mok in the development of an earlier version of OpenBoard, and T. Shen, D. Young, T. Ballantyne, and C. Nguyen in the development of the OpenBoard Tutorial System.

References

Ahuja, S. R, Ensor, J. R., and Horn, D. N., "The Rapport Multimedia Conferencing System," Proceedings of Office Information Systems, Palo Alto, California, 1988, pp. 1-8.

Goscinski, A., "A Distributed Computer System Supporting Remote and Concurrent Meetings." Proceedings of the IEEE Malaysia International Conference on Communications, Kuala Lumpur, Malaysia, 1993, pp. A4.1.1-A4.1.8.

Liu, Y and Hoang, D. B., "A Window, OSI-Based Distributed Application Supporting Concurrent Meetings." Proceedings of the International Conference on Communication Technology (ICCT'94), Shanghai, China, 1994, pp. 1411-1415.

Liu, Y and Hoang, D. B., "OSI Remote Procedure Call - Standardization issues, design and implementation, to appear in *Computer Communications* journal, 1997.

Stefik M, Foster G., Bobrow D. G., Kahn K., Lanning S. and Suchman L., "Beyond the Chalkboard: Computer Support for Collaboration and Problem Solving in Meetings." Communications of the ACM, 30, 1, 1987, pp. 32-47.