A Platform for Supporting Multimedia Interactive Tutoring

Jim M. Ng, Edward Chan, Peter H.H. Tsang
Department of Computer Science
City Polytechnic of Hong Kong, 83 Tat Chee Ave., Kowloon, Hong Kong

Abstract: In most educational institutions, tutoring and consulting services are provided to the students by having a special consulting room where students can meet the counselor or tutor face to face. Since an increasing number of students are preparing their work with the aid of computers, queuing up for the tutors with printouts of error messages or partial results every time is not an efficient way to get assistance. In this paper, a platform for supporting multimedia interactive tutoring will be described. Using the facilities provided, the students can seek advice without leaving their workstations. The tutor can view error messages or documents using the shared screen, and point to any location on the shared screen using a telepointer. Furthermore, a voice channel is set up so that verbal discussion can be exchanged between the student and the tutor. Extensive multimedia conferencing facilities are also provided to allow the tutor to conduct on-line group tutorials.

1. Introduction

In most educational institutions, a variety of consulting and tutoring services are provided to the students. These services can in general be classified into personal consulting, such as career planning and course selection, and academic consulting, such as providing help on lecture material or homework assignments. For personal consulting, students will usually arrange for an appointment and meet the counselor privately in a room as a face-to-face session is more preferable. On the other hand, for academic consulting, if a student has some problems with the lecture material, he may discuss with the lecturer personally during office hour, or he can post an e-mail message to the lecturer. Also, a special consulting room is usually set up for students to bring in their work and ask for advice. For instance, many computer science departments have some of their graduate students on duty in the consulting room as tutors. Students who need assistance will typically print out the computer programs together with any error messages before seeing the tutors.

As most of the students' assignments are now prepared on computer, printing them out every time is clearly very inefficient. If there are any error messages displayed when the task is carried out, the tutor will not be able to see them unless they are captured or written down by the students.

Furthermore, in order to see the tutor, the students have to leave the workstations and no useful work can be done while queuing up.

MITS, a Multimedia Interactive Tutoring System, is designed to improve the communication between tutors/lecturers and the students. When a student encounters a problem and would like to discuss it with the tutor, he can initiate a remote tutoring session by sending a request to the tutor from his workstation. The tutor, who serves the students on a first come first serve basis, will respond with a start session message. A shared screen and a voice channel will then be established between the student and the tutor. The student can create one or more shared screens to show any documents, program listings, error messages, and pictures to the tutor. The design of MITS is based on much research effort in the area of desktop multimedia conferencing systems (Abdel-Wahab, 1988; Ahuja, 1988; Karmouch, 1990; Ng, 1993; Ohkubo, 1990; Stefik, 1987; Watabe, 1991).

In the next section, the system requirements will be discussed. Then in section 3, the system architecture will be presented. An open architecture is adopted so that different applications can be built: the interactive tutoring system is just one of them. The MITS system components and facilities will be discussed in detail in section 4. Some future enhancements are included in the conclusion of the paper.

2. System Requirements

In this section, the system requirements for a real-time interactive tutoring system will be discussed. The major requirements identified include: a mechanism for the exchange of multimedia information, a mechanism for the control of the conferencing session, a concurrency control mechanism for simultaneous editing of public documents, and a floor control mechanism.

For a tutoring session, it is often necessary for both the tutor and the student to view and access the same set of information. For instance, the student may want to show the tutor a program or segment of a program listing, and any changes made by the tutor should be reflected on both the student's and the tutor's screens, i.e. a What-You-See-Is-What-I-See (WYSIWIS) environment is often desired. However, during the tutoring session, the tutor may want to look up other information which does not need to be presented to the students; hence, the WYSIWIS requirement is too restrictive, and both the tutor and the student should be allowed to create *private windows* for the display and manipulation of private information. User application programs mapped to the private windows will then have local significance, and all editing and information manipulation will only be seen by the local user. On the other hand, once a file (or part of a file) is moved to the *public window*, both the tutor and the student can edit the file and any update will be seen by both parties.

By opening a public window, the tutor can exchange textual information with the student, and he can also amend the student's document or program. Furthermore, the student/tutor can load in any picture if necessary. With an image viewer, one can draw on or point at any location of the picture, so that the tutor can put marks on the picture during discussion. Besides exchanging data in textual and graphical forms, a voice channel may also be established between the tutor and the student so that verbal communication is possible.

Concurrent access to shared data is a key design issue in general conferencing systems. While some argue that simultaneous editing of a file should be allowed, others prefer to have only one

person edit the file at any one time. In the tutoring environment, the tutor will be the one manipulating the file most of the time, and it is often undesirable to allow the student to edit the file simultaneously. Hence, in MITS, a floor control mechanism is used so that only the person who has the floor can modify the shared data.

In a simple situation where there is only one student in the session, the floor will be passed back and forth between the student and the tutor. As it is desirable to cater for the situation where other students can join in the discussion during a tutoring session, a floor control mechanism is needed. Since the tutor in a tutoring session has the highest authority, all requests of the floor will be routed to the tutor, and he will then decide to pass the floor to one of the students. The tutor can also take back the floor at any time whenever he wants to talk or manipulate the files. Furthermore, since there may be more than one student requesting for service at any one time, a first come first serve discipline will be used to regulate the service order. However, while waiting, a student may have his problem solved, and he should hence be allowed to withdraw his request.

3. System Architecture

MITS is developed on SUN Sparc workstations interconnected by an Ethernet. Each station has at least 12 Mbyte RAM, a color monitor, a speaker, a microphone, and its own hard disk. Currently, there are ten workstations connected together. The MITS system software runs on top of an industry standard platform of UNIX and X11R5 to ensure portability.

MITS is composed of two subsystems: the applications subsystem and the conferencing subsystem. The applications subsystem provides the user interface and application services, such as text editor and image viewer. The conferencing subsystem provides the low level conference

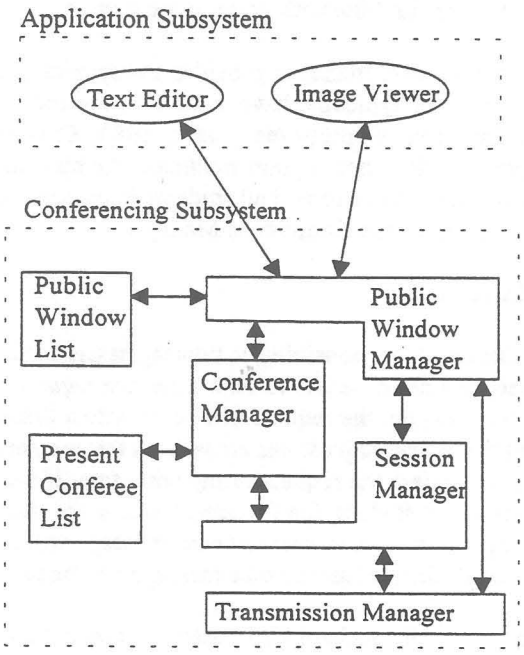

Figure 1. Architecture of MITS

management and data transmission services to the users. A distributed model is adopted for our system where the tutor and the student(s) keep their own set of system status and parameters as well as copies of public window contents. Hence, when a public window is opened by the floor holder, say the student, a copy of the corresponding application will be invoked at the tutor's site, and the file opened or information put onto the public window will be passed from the student to the tutor. Hereafter, any updates made by the floor holder will be passed to the other party via the conferencing subsystem which will distribute only the update operations. Since all stations see the same input and carry out the same operations, their states will remain synchronized yielding identical output at each site. Note that if more than one student is involved in the same tutoring session, information will be distributed to all students via the conferencing subsystem.

Figure 1 shows the system architecture of MITS. The conferencing subsystem is composed of four main units, namely the Transmission Manager, Session Manager, Conference Manager, and Public Windows Manager.

3.1 Transmission Manager

The Transmission Manager provides the functions of establishing a channel and transmitting data for the various applications. In multimedia applications, different data types usually require different qualities of service. For instance, audio and video information streams require data to be received within a tight time constraint and with a small delay variance; however, they do not require 100% reliable transmission, i.e. data loss can be tolerated. On the other hand, text data will tolerate higher delay but require reliable transmission. In order to cater for these different requirements, the transmission manager will provide different types of services accordingly. Furthermore, synchronization of multiple streams of information is typically required in a multimedia environment. For instance, the position of the cursor in the public window, called the *telepointer*, should synchronize with voice in most cases.

Currently the transmission manager provides transparent transmission of data to all conferees. Since current networks typically have a very low error rate, a UDP-like protocol has been developed for transmission purposes (Clark, 1990; Crowcroft, 1992; Ng, 1994). Since the transmission protocol does not support multicast, the transmission manager provides this service by sending messages to the tutor and all students in the same session based on a *Present Conferee List (PCL)* maintained by the Conference Manager.

3.2 Session Manager

The Session Manager is responsible for tutoring session management. In MITS, when a student wants to connect to a tutor, he has to send a *service request* message to the tutor. If the tutor has an active session going on, the request will be put into a first come first serve queue, and the tutor will be notified that a new request has arrived. At this moment, a session has not been established yet, and the tutor can pass the request to any other tutor if necessary, and a *change tutor* message will be sent back to the student. On the other hand, while waiting for the response, the student can withdraw his request, and a *withdraw request* message will be sent to the tutor. When the tutor is ready to serve the student, a *start service* message will be sent to establish a session. .

In MITS, a simple poll mechanism is used to monitor the status of conferee stations. In the simplest situation where there is one student in the session, the tutor's station and the student's station will poll each other. If an acknowledgement is not received after a predetermined timeout

period, the polling station will assume the other station has failed and the connection will be dropped.

When there is more than one student in the same tutoring session, the polling scheme makes use of the PCL maintained by the conference manager at each station. The basic scheme is that the stations on the PCL form a logical ring as far as error detection purpose is concerned. Each station is responsible for polling the next station on the PCL list. If an acknowledgement is not received after a predetermined timeout period, the station will assume its successor has failed, an error message will be sent to all conferees, and it will try to poll the successor of the next station on the list. On the other hand, if a poll message has not been received by a successor after a timeout period, the station will assume its predecessor has failed and it will try to poll the predecessor of the failed station. After it has received an acknowledgement, normal polling according to the updated PCL continues. If the failed node is the tutor, the tutoring session will be closed automatically. On the other hand, if the failed node is the floor holder, the tutor will then take back the floor control.

3.3 Conference Manager

The Conference Manager is responsible for the floor control passing mechanism as well as the maintenance of some key information regarding the status of the tutoring session. It works closely with the Session Manager to provide the basic conferencing facilities.

Before a tutoring session starts, a *Present Conferee List (PCL)* will be set up at each site. The PCL stores the information of all members involved in a tutoring session. Also, the current floor holder will be marked on the list. This list is used by the session manager for session establishment and maintenance as explained in the previous section.

In MITS, only one floor passing mode is supported, namely *tutor selects next.* A *current floor request list (CRL)* is maintained by the tutor. Although the CRL is arranged in first-in-first-out order, the tutor can decide which student can have the floor control next. The tutor also has the authority to get the floor back from the student whenever necessary.

3.4 Public Window Manager

The Public Window Manager interfaces with the applications by means of a set of Application Programming Interfaces (API) and hides the details of the conferencing subsystem from the applications. The clear demarcation between the basic conferencing platform and the applications is what makes the platform open and extensible: new applications can be added without affecting the existing ones. There is a one-to-one mapping between public applications and public windows, and multiple public windows can be supported. The tasks handled by the public window manager includes public application invocation, peer public application communication, and initialization of the application content of the new conferee.

A *Public Window List (PWL)* is maintained by the manager at each site. Since each application communicates with the conferencing system through a UNIX named pipe, the *PWL* provides a mapping between the pipe id of each application and its corresponding logical id. This mapping is needed because the pipe id is generated locally when an application is invoked. All communications between peer public applications will be done by multicasting a message to other conferees by means of the logical id. A remote public window manager receiving the message

will then map the logical id to its own pipe id, and send the message to the application accordingly. In addition, since only the floor holder is allowed to manipulate the content of a public application, the public window manager is also responsible for informing the application if the floor has been granted or removed from the station.

The API provided by the Public Window Manager currently consists of the following function calls:

int IsFloorHolder()
int App_init()
void QuitApplication()
*int SendBuffer(char *buffer, int type, unsigned long bufSize, int sendmode)*
*int OpenConnection(char *argv[], Widget parent)*
int CloseConnection()

OpenConnection allows data connections of different quality of service to be opened and used by the application, it also inserts an application id into the public window list; in the same manner the *QuitApplication* function removes the application from list. The *SendBuffer* function allows data to be passed transparently to all or certain conferees with the support of the transmission manager, which will handle the actual transmission and synchronization of the data streams. *IsFloorHolder* and *App_init* are used by the application to check its status and to create a message receiving agent respectively. While the API outlined is very simple, it is found to be adequate for the development of most applications, including all those outlined in the following section.

4. The MITS System Services

A number of applications to facilitate the tutoring session have been built on top of the underlying conferencing subsystem framework described in the previous sections. As discussed in section 2, a tutoring session will require the exchange of textual, graphical, and even audio data. The tutor may want to view some documents, point to various locations in a picture, discuss the problem with the student verbally, and make changes to some of the files. Information can be moved between public windows as well as user's own local windows, i.e. private windows. Generally speaking, the users enjoy an interactive environment similar to that of a face-to-face discussion session.

The transmission of voice is an integral part of the system, so that when a session is established, a voice channel between the tutor and the student(s) will be setup automatically. As in most conferencing systems, only the floor holder can speak. Besides the audio channels, currently there are two other basic applications developed based on the requirements specified: a text editor, and an image viewing system. All application subsystems are developed using Motif to provide a user-friendly interface.

4.1 Text Editor

A screen editor which supports basic editing functions has been developed. After opening a public window, the window content can be loaded by opening a file or copying some text segments from another text window. The clipboard function provides a convenient way for a user to copy text between different windows, and in particular, between public and private windows.

Hence, one can control and select only relevant information to be presented on the public window for the tutor/student.

When a new file is loaded in a public window, the editor uses the underlying conferencing subsystem to distribute the information to all members. Thereafter, only update operations are sent to reflect the editing operations.

4.2 Image Viewer

The image viewer supports a small set of image and graphics processing primitives. It now supports four basic geometrical operations, namely image mirroring, flipping, rotating, and scaling. In addition, several special image enhancement operations are provided. A *histogram equalization* function is provided for viewing images, such as X-ray, where the pixel values are usually skewed to the left, i.e. the screen shows a dark image. After performing histogram equalization, the pixel values are re-distributed to create an image with higher contrast. *Pseudo coloring* adds colors to a black and white X-ray image or satellite picture so that different components can be shown and distinguished clearly. *Contrast changing* is provided to change the intensity ratio from 1.01 to 1.1 in ten steps. *Image thresholding* and *edge detection* functions are also provided.

Besides the functions specified above, the image viewer also provides graphics overlays functions for users to mark or highlight various location on the image. A telepointer is also provided for the floor holder to point to any location on the image so as to draw the attention of the students to a specific area or object.

4.3 An Interactive Session

MITS provides a friendly menu driven interface to the user. After invoking the system, a control panel will be displayed for the student to initiate a request to the tutor. The panel will display the student's current status, which is either *waiting, rejected,* or *start session.* The student will be in the *waiting* state if the tutor is busy with other students and he has to wait for his turn. If the tutor to whom he wants to connect is not logged on or has MITS turned off, a *rejected* status will be returned, and the student may choose to drop the connection or connect to another tutor. When

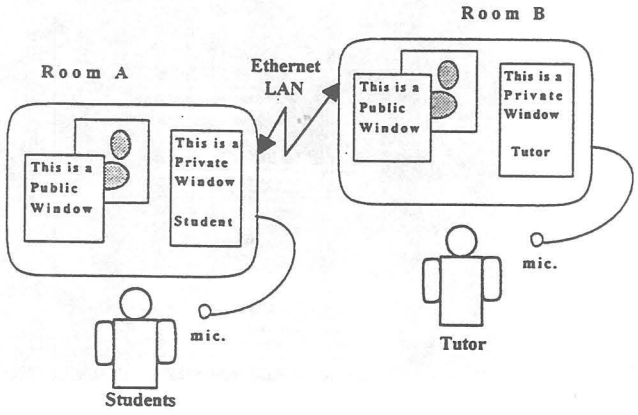

Figure 2. A Tutoring Session Scenario

the tutor is ready to serve the student, the status will be changed to *start session*, and a voice channel and a public text window will be set up by the system. The student and the tutor can then communicate through either the voice or the text window.

After getting the floor from the tutor, the student can then open one or more public windows and load documents and pictures. If the student has tried to compile a computer program, all error messages can be copied onto the public window for the tutor to examine. The tutor can discuss the problem with the student and amend the computer program, and all amendments can be seen by the student instantaneously. Furthermore, a medical student may load in an X-ray image and discuss with the tutor while pointing at various locations on the screen using the telepointer. The tutor can mark and write on the image in order to highlight various positions. Figure 2 shows a scenario of a tutoring session where the tutor and the student use two shared screens, i.e. public windows, and each maintains his own private window. Figure 3 shows a couple of snapshots of the MITS system. A session can be terminated by either the tutor or the student through a *disconnect* message.

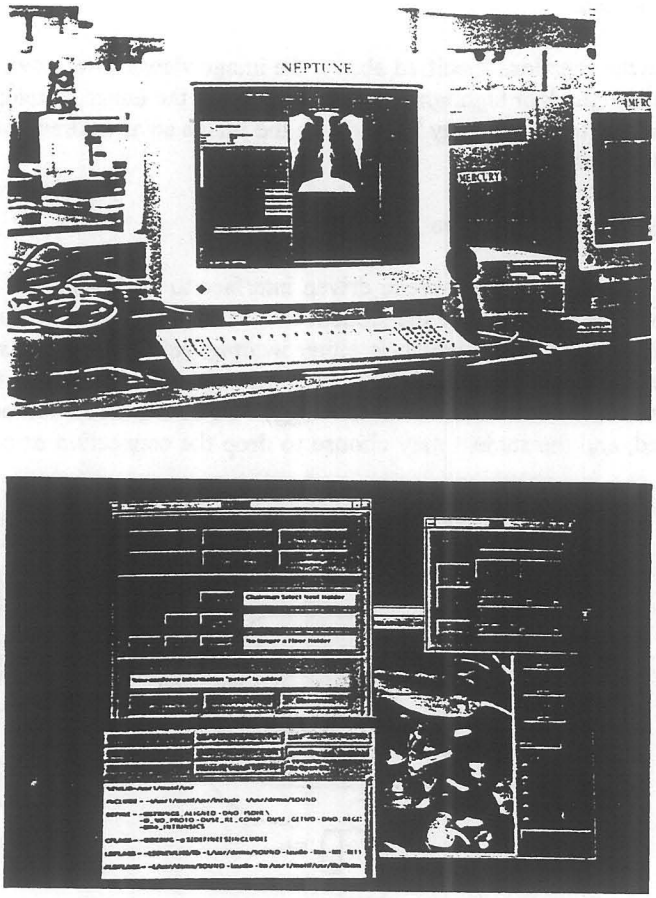

Figure 3. Snapshots of the MITS

5. Conclusion and Future Enhancements

A prototype system has now been developed with two applications: a text editor and an image viewer. The users can open multiple copies of the applications whenever necessary. Although only two applications are provided, it has been found that they are sufficient for our purpose in most situations. Real-time response is achieved, and with the user-friendly interface, users are provided with many capabilities of a real conferencing system. The provision of such facilities enables tutors to hold on-line small group tutorials.

The polling mechanism in the session manager has proven to be very useful especially in the case where students tend to log off without properly removing themselves from the MITS system. When a student logs off, the system will detect a connection failure, and remove the entry from the tutor's waiting list automatically. Duplicating a copy of a file in the public window also helps the student to recover the original file if there are some undesirable changes on the document during the tutoring session.

Although the floor control mechanism is found to be very useful in situation where there is more than one student involved in a session, it becomes a bit clumsy and inefficient when there is only one student discussing with the tutor. With voice communication, it is very natural to have the two persons take turns in talking; hence, it is awkward to require the users to ask for the floor every time. So, the system may have to distinguish the case when there are only two persons, one tutor and one student, in a session, and the case when there are more than two persons in the session, and uses different controlling mechanism. However, one major disadvantage with this approach is that an inconsistent environment is presented to the users.

Another problem with the current system is that the student requesting the service does not know how long he has to wait. The student has no information on the queue length which is now maintained only at the tutor's site. The queue length information is actually needed before the student makes a request so that he may not want the service if the queue is too long. Also, after he makes a request, the student may want to know the current status to estimate the waiting time. As a result, we intend to modify the system slightly so that when a student makes a request to a tutor, the current queue length will be returned. The wait queue of all the tutors can also be returned as an option. The student will then be prompted to ask if he still wants the tutoring service. Depending on the student's response, the request will be dropped or a *request commit* message will be sent to the tutor to confirm the request for service.

We are now considering the usefulness of allowing the tutor to open multiple sessions. Although the public windows can be handled easily by slightly changing the conferencing subsystem, the audio channel between the students must be handled with more care. Since it is impossible to listen or talk to multiple sessions simultaneously, or talk to a session and listen to another session, it is perhaps best to restrict the system to support audio communication for one session only.

Another useful feature we are contemplating is a session recording facility, which would allow part or all of a tutoring session to be recorded and stored in an easily accessible manner. A tutoring session is a valuable resource since it often captures common student mistakes, as well as attempts by the tutor to deal with problems raised by students. Review of these sessions can be useful in helping tutors to improve their tutoring skills. The large storage requirement of such a facility is a potential problem, and careful investigation into compression techniques is needed.

Currently synchronization between multiple streams of different information media is still being developed and refined. This implies enhancement in the capabilities of the transmission manager. One of the advantages of the current platform is the modular structure of its key components. This means that advances in technology will only result in changes in individual components. For example, we are planning to experiment with full video conferencing facilities and high speed networking technology such as Asynchronous Transfer Mode (ATM), but we do not expect that the modifications in MITS will be extensive. We do expect a rather different user experience if a video channel is provided, and we hope to report on our enhancements in a future paper.

References

Abdel-Wahab, H.M., Guan, S.U. and Nievergelt, J. (1988). Shared Workspaces for Group Collaboration: An Experiment Using Internet and UNIX Interprocess Communications. *IEEE Communications Magazine,* 26(11), pp.10-16.

Ahuja, S.R., Ensor, J. and Horn, D. (1988). The Rapport Multimedia Conferencing System. *Proceedings of the Conference on Office Information Systems*, pp.1-8.

Clark, D.D. and Tennenhouse, D.L. (1990). Architecture Considerations for a New Generation of Protocols. *Proceedings of ACM SIGCOMM '90,* pp. 200-208. Philadelphia, Pennsylvania: ACM.

Crowcroft, J., Wakeman, I., Wang, Z. and Sirovica, D. (1992). Is Layering Harmful? *IEEE Network*, 6(1), pp. 20-25.

Karmouch, A., Orozco-Barbosa, L., Georganas, N.D. and Goldberg, M. (1990). A Multimedia Medical Communications System. *IEEE Journal on Selected Areas in Communications,* 8(3), pp. 325-339.

Ng, J. M., Chan, E., Ip, H.H.S., Tsang, H.H., Kwok, K.Y. and Lee, Y.K. (1993). A Distributed Multimedia Conferencing System. *Proceedings of IEEE TENCON'93*, pp. 70-73. Beijing, China.

Ng, J. M. and Yu, T.Y. (1994). Transport Protocol for Real-Time Multimedia Communication. Proceedings of the 19th IFAC/IFIP Workshop on Real-Time Programming. Lake Constance, Germany.

Ohkubo, M. and Ishii, H. (1990). Design and Implementation of a Shared Workspace by Integrating Individual Workspaces. *Proceedings of the Conference on Office Information Systems*, pp. 142-146.

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

Watabe, K., Sakata, S., Maeno, K., Fukuoka, H. and Ohmori, T. (1991). Distributed Desktop Conferencing System with Multiuser Multimedia Interface. *IEEE Journal on Selected Areas in Communications,* 9(4), pp. 531-539.