CVIEW: A Real-Time Interactive Conferencing System

IBM Research Report RC 16138
9/28/90

Larry Koved

IBM Research
T. J. Watson Research Center
P.O. Box 704
Yorktown Heights, New York 10598

Abstract

Computer-based tools for supporting collaboration, from informal conversations to structured meetings, are becoming commonplace. Different technologies support various techniques for computer conferencing. On the one extreme is asynchronous communication where the participants do not need to be available at the same time in order to communicate. Traditionally, this has taken the form of computer conferences, bulletin boards, mail and forms management. In contrast, video conferencing provides the ability to have a conference where the participants can see each other and possibly make impromptu presentations. Video conferencing is progressing from freeze frame to full motion technology.

An emerging area in computer supported cooperative work is real-time computer conferencing. This technology allows two or more people in the privacy of their own offices to see a *shared* view of a computer application. That is, when any person moves a mouse, uses the keyboard, or uses other input devices, the result of these actions are immediately shown to all people in the conference (limited by network bandwidth). This technology is especially useful in consulting, teaching, programming, meetings, marketing technical writing and a variety of other applications.

This paper discusses a system called CVIEW that implements a real-time interactive conferencing system. The architecture of the system is described along with its implications and directions for future systems based on User Interface Management Systems (UIMS).

Introduction

Computer-based tools for supporting collaboration, from informal conversations to structured meetings, are becoming commonplace. Different technologies support various techniques for computer conferencing. On the one extreme is asynchronous communication where the participants do not need to be available at the same time in order to communicate. Traditionally, this has taken the form of computer conferences, bulletin boards, mail and forms management. In contrast, video conferencing provides the ability to have a conference where the participants can see each other and possibly make impromptu presentations. Video conferencing is progressing from freeze frame to full motion technology.

An emerging area in computer supported cooperative work is real-time computer conferencing. This technology allows two or more people in the privacy of their own offices to see a *shared* view of a computer application. That is, when any person moves a mouse, uses the keyboard, or uses other input devices, the result of these actions are immediately shown to all people in the conference (limited by network bandwidth). This technology is especially useful in consulting, teaching, programming, meetings, marketing technical writing and a variety of other applications.

A number of systems have been created to provide real-time conferencing. One of the earliest was available in Doug Engelbart's NLS/Augment system [Engelbart68, Engelbart82]. This provided the ability to share a window, as well as display real-time video images. Recent systems have been oriented more towards specific applications rather than generalized systems to support existing, unmodified, applications [Pferd79, Foster84, Forsdick86, Aguilar86, Poggio85].

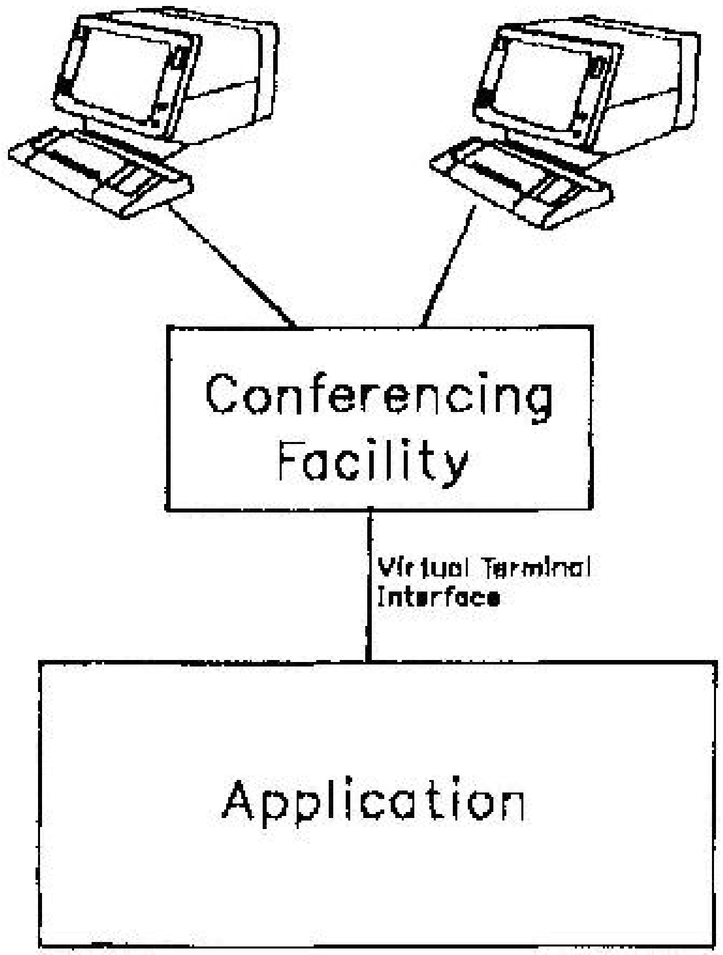
From these systems and current research, fundamental issues in the design of real-time conferencing systems are emerging. In the past, applications were incorporated as an integral part of the real-time conferencing facility. This design approach did not provide immediate reusability of any of the components of the system for other applications. The integration of new applications into the real-time conferencing system could require substantial programming effort. Each new application may need custom tailoring for the conferencing system environment. Recent proposals and systems [Sarin84, Sarin85, Lantz86, Poggio85, Aguilar86, Foster84] have been moving in the direction of developing a structure or framework with tools to support the development of real-time conferencing systems. This is a promising approach for the development of new applications, particularly for distributed computing environments [Sarin84, Sarin85, Lantz86, Poggio85, Aguilar86].

However, a large base of existing applications is not built using tools to facilitate real-time conferencing. Users of these applications have a need for real-time conferencing. The conferencing facility has, as a minimum, the following requirements:

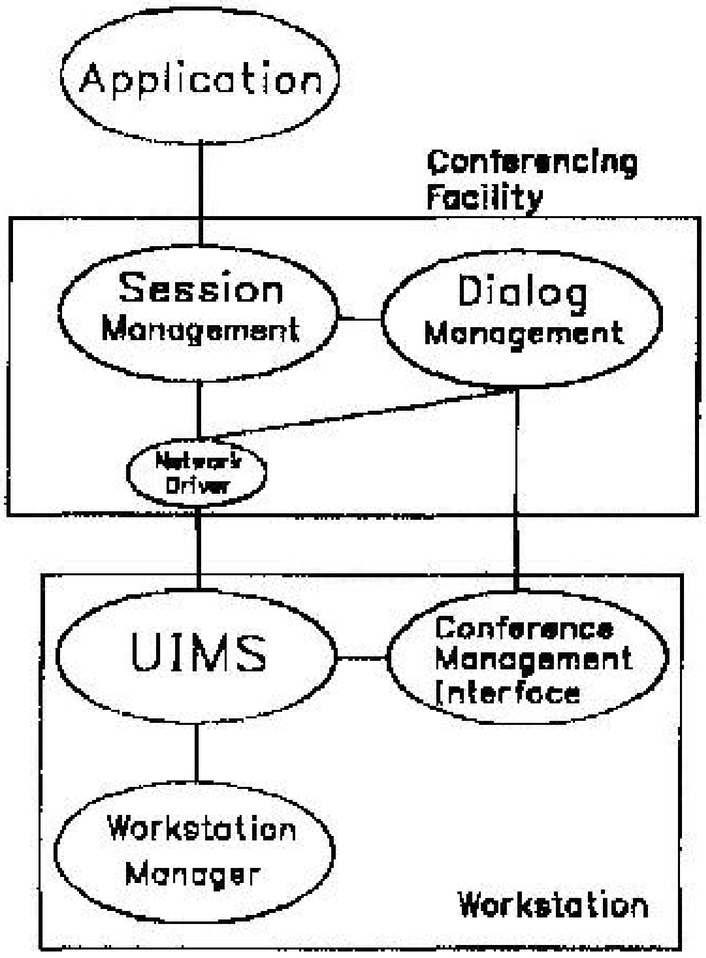
- To the application, there appears to be only one user.
- The conferencing facility supports access by two or more people.

This approach has been referred to as *terminal linking* or *terminal sharing* (see terminal linking figure). The particular advantage of this approach to conferencing is the relative ease of construction, preservation of existing applications without modification. Special coding and new tools are not needed to create new applications that can be used in conferencing contexts. People building applications for conferencing can build them as though they were building an application for single user use. This design objective leads to a clear and clean separation between the conferencing facility and the application [IBM, Lantz86]

Terminal Linking



Original computing environment



Cooperative Viewing Facility (CVIEW)

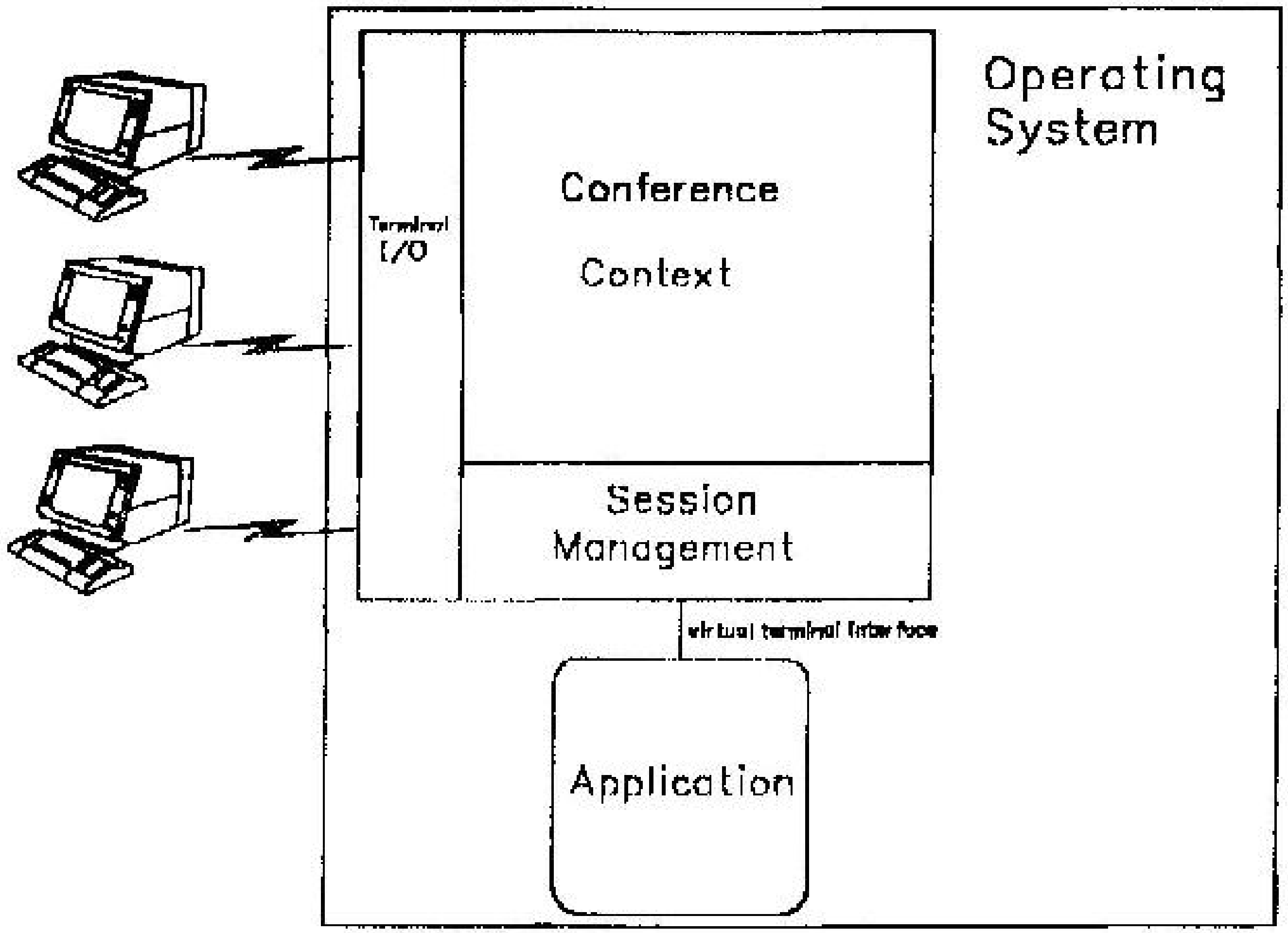
Background

The CVIEW project began in early 1982 as a means of providing consulting and educational support for a geographically distributed user community. The computing environment consisted of several IBM 370 architecture CPU's (3033's or 3081's), and 4341 or 4381 processors at satellite locations. Users had 3270 terminals or terminal emulators for logging onto the systems. These terminals are half duplex with the input buffered at the terminal or terminal control unit. Input is transmitted to application after the user presses one of the predefined function keys (which includes Enter and Clear-screen). Multi-media I/O is supported through escape sequences in the 3270 data stream. One example of this capability is the attachment of a Tektronix 614 storage tube display to a 3270 display or terminal emulator. Another is the loading of fonts into the display to generate color graphics output. Access, including login and file transfer capabilities, from the satellite locations to the central computing facility was provided over high-speed leased lines (see computing environment figure). Typically, when a user of the system had a problem that could not be solved through online information or through conversing with a colleague, the user would call the Help Desk or walk to a consultant's office. Walk in consulting was a common method of getting problems resolved.

CVIEW was motivated by the need to provide consulting to a user community that was 15 to 30 minutes from the central site. When a user at a remote site had a problem, the consultant did not have access to the user's login account to examine the environment, perform problem determination and work on problem resolution. A phone conversation was frequently inadequate because the consultant did not always get sufficient information from the user about what was on the display screen or some specific information about the run time environment. Consultants frequently commented about errors in command syntax that was not being adequately communicated via the telephone. The round trip cost to the central site was 30 minutes or more for a problem that often could be resolved in a couple of minutes at the terminal. Having the user log off the account and the consultant log onto the account was not a viable alternative. From an educational point of view, the user would not know how to fix the problem if it was encountered again because the corrective procedures used by the consultant were not observed. From an administrative point of view, the sharing of passwords was frowned upon.

One of the initial assumptions was that the users of the computing systems at the research lab were working with a wide range of software tools. Most of these tools could not be rewritten if that were the only means by which they could work with a conferencing facility. Another major assumption was that the conferencing facility should not be part of the operating system. The separation of the facility from the operating system would provide the flexibility to adapt the conferencing software to meet the needs of specific user requirements (i.e., consulting vs. education).

CVIEW architecture



System Structure

CVIEW uses the terminal linking approach to provide real-time conferencing (see CVIEW figure). The major components are the session management, terminal I/O and conference context. The *session management* provides CVIEW with access to the operating system's virtual terminal interface. This code also maps the differences between the different terminal types in the 3270 family (slight variations in data stream format, screen size, graphics capabilities, etc.). The *terminal I/O* interface drives the real display devices or the network interface for remote terminals. The *conference context* provides the means of entry and departure from the conferencing system, as well as dialogues for controlling the conference. These are the conference management functions. Application programs run in a separate process (userid) and interface to the conferencing system through the virtual terminal interface. The applications are given the appearance that there is only one terminal attached. All of the multiplexing of the terminal I/O is handled by the session management and conference context code. In addition, the session management code and all of the conferencing applications support multiple concurrent (independent) conferences.

Conference Contexts

The design of CVIEW provides flexibility in applying conferencing facilities to a variety of uses (contexts). The conference context is the set of policies governing the types of actions and functions available to the user while in the conference. These include how to enter and leave a conference, who is given permission to provide input to the shared application, security and access control, and transcript facilities. Typically, when users enter the facility, they are presented with a set of menus. These menus guide them through the conferencing facility.

Three different contexts were prototyped, of which two became the IBM product. The prototypes were for pair-wise consulting, multi-user conferencing and a classroom presentation tool. Extensions to the multi-user conferencing and classroom presentation were planned but not developed.

Consulting

One of the initial uses for CVIEW was for pair-wise consulting. A person who needed assistance from another person would connect to the consulting facility through the use of existing networking and operating system facilities. Through a set of menus, the user would specify with whom they wanted to confer. Similarly, the other person would connect to the conferencing facility and identify themselves. The conferencing context would recognize that the two people wanted to start a conference and would send the request to the session management code. The session management code would create a virtual terminal, associate the two real terminals with the virtual terminal and perform the I/O multiplexing with any necessary terminal data stream translations. One of the two conferees would log onto a userid and begin the conference. Either person could send input to the application. All output from the application was sent to both person's terminals. Since 3270 terminals are half duplex devices, a function key could (optionally) be dedicated to allow input from one terminal to be displayed at the other person's terminal before it was sent to the application.

When the conference was completed, the conferees log off the userid and the virtual terminal is closed. The session management code reflects the session termination to the conference application which takes over control of the terminals. Another dialog is initiated to permit departure from the facility or to start another conference.

A transcript capability was prototyped but not included in the product. The conference context could make a request to the session management code to begin recording all I/O to and from the virtual terminal. Each I/O was time stamped to record actual timing delays. The transcript file could be edited to modify the timing delays in order to produce a smooth presentation on playback. Through a separate stand-alone application, the session could be played back on any terminal similar to the ones used in the conference.

Multi-user Conferences

In a multi-user conference setting, additional controls are needed for entry, departure, viewing ability, and "passing the floor" to different conference participants. Each participant in the conference supplies the conference name and their name to the conference context. If the person is an attendee, they wait for the moderator to begin the conference. The moderator sees a screen with the names of the people who have requested to attend the conference. For each attendee requesting access, the moderator has four possible actions:

1. Full participation - view and data entry capability
2. View only
3. Drop from the conference
4. Suspended - none of the above (access permission neither given nor denied)

When all of the participants have arrived, the moderator can request conference initiation. The conference context requests that the session management code create the virtual terminal and link the terminals into a session. The moderator or whoever was given full participation capabilities, can start an application by logging onto a userid and starting a program. When the application is terminated (logging off the userid), the conferees return to the conference context. The moderator can change the participants and their level of access to the next session.

Additional Multi-User Conference Controls

A number of additional functions had been proposed for the multi-user conferencing system. While some detailed planning was done for an extended prototype, it was never implemented. Even so, the functions that were envisioned are worth noting here.

In a conference, where there may be a number of arriving and departing participants, greater control is needed by the moderator. The moderator and participants should have a function key that allows them to switch from the shared display mode back to the conference context. From there, attendees can leave the conference or join a different conference. Moderators can change the participants or the access rights the current participants have to the shared session. While participants are all connected via some shared voice communication channel (i.e., conference call), participants would like the ability to pass private messages. This is a function that could be contained in a separate window. Or, several participants can start up a private real-time conference on the side. Other forms of shared communication could be provided by the conference context, such as voting [Begeman86].

As people begin to explore real-time conferencing, there may be considerations for the support of multiple concurrent conference sessions. At many professional conferences, people wander from room to room where papers are being presented, tutorials are being given, panel discussions are taking place, etc. In the future, we may find that people will hold these conferences, or something analogous, electronically. Additional tools in the conference context will be necessary to support larger, less personal, conferences.

Teaching

The third conference context that was prototyped was for a classroom setting. Usually there would be one teacher presenting information to all of the students in the class. The teacher would create a class with a name. Each of the students would select the class and wait for the teacher to start the shared session. Students were typically in a read-only mode. They could easily get out of the class by pressing one of the function keys. This design was very effective for straight forward demonstrations of how to use some specific software on the system or for marketing presentations.

A number of useful extensions to this system were envisioned:

- Each of the participants should be able to concurrently log onto their own userid to work on exercises and run some of the examples provided by the instructor. Controls to switch between the shared session and the private session are needed.
- Participants should be able to signal the instructor that they have a question or need assistance.
- Be able to show the instructor what they are doing.
- Allow the instructor unobtrusively to "look over the shoulder" of the students as the students are doing work in private sessions.
- Allow the instructor to start a session on the side to run applications independent of the presented material. This is useful for information searches or trying new scenarios without confusing the students.
- When a student runs across something interesting, the instructor should be able to switch from the original shared session to the student's session and make it viewable by all participants.
- Transcripts of the class could easily be made by the session management code and sent to each of the participants after the class has completed. This reduces the need to take notes. Additional note taking can be done in the student's private session.

Some of these features can be performed through the use of a video display sharing switch. The distinct disadvantage of the hardware solution is its lack of flexibility and the inability to use it in a geographically dispersed user community.

Social Interaction

The social interaction between participants in real-time conferencing is quite different from face-to-face meetings. The people usually make initial contact through the telephone. For first-time users of the conferencing facility, there is a short period of education (1/2 to 1 minute) to familiarize them with a new tool. After conference initiation, protocol for control of the shared session takes place. Transfer of control is established through explicit verbal exchanges or implicitly through the context of the conversation.

Gesturing is different when using a real-time computer conferencing system. When people are in physical proximity and have visual contact, they use hands and body movement to draw attention to items of interest. When using the conferencing facilities, the contact between people was usually limited to the audio channel (telephone) and the computer display. In particular, the 3270 class of terminals is half duplex with buffered input. This severely constrains the range of gestures available to the conferees. Frequently, the audio channel is used to draw attention to objects viewable on the screen. When this mode of interaction breaks down, the objects on the screen were manipulated to draw attention to areas of interest. Several screen-based techniques were frequently employed. First, a blinking cursor is moved to an object of interest. Through the use of a key defined at the start of the session, the new position of the cursor could be transmitted to all of the participants without sending the cursor position information to the application. In many of the IBM text editors, there is a concept known as the "current line." Often the current line is highlighted in some way (bright intensity, color, flashing, etc.). To draw attention to specific information in the file, one of the participants scrolls the file to move the items of interest to the current line. Similar techniques are used in other applications. Unfortunately, in terms of providing inter-user communication, the 3270 class of terminals severely restricted the level of possible interaction between conferees.

With 3270 class terminals, the half duplex I/O with buffered input does have a few distinct advantages. All input devices (keyboard, light pen, mouse, etc.) can be given input capability. Through verbal communication, the transfer of control can take place. However, two or more participants can make data entry at the same time. The information is not transmitted to the application until one of the function keys is pressed. Through verbal protocols, the participants can decide whose input is to be sent. This allows local editing by several people at the same time and the decision as to whose input is to be used can easily be deferred. This leads to the situation where the participants would like to see the input before it is to be sent to the application. Through the use of a function key, the session management code is signaled to read the input from one terminal and redisplay the input on all of the participants' displays. The input can be edited by another participant before being sent to the application.

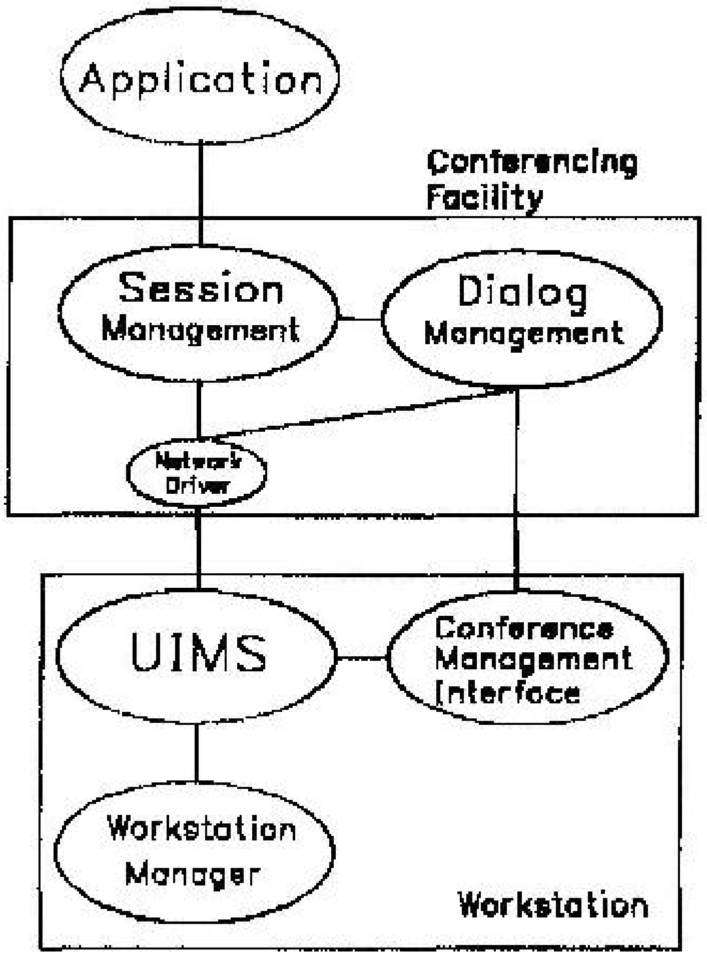
The 3270 terminal protocol is straight forward to use in a real-time conferencing context. It greatly simplifies a number of issues in the area of I/O multiplexing. Questions of how to transfer control of input from one user to another is not a necessarily a technical one. Rather, it can be deferred to the users involved. Thus, it becomes a social interaction issue.

A Model For Real-Time Conferencing

The work on CVIEW and other recent research [Sarin85, Lantz86] provides insight into the technical issues surrounding real-time conferencing. Lantz argues that application should be running on all of the user's workstations at the same time. The problems that arise include the management of a computing environment that is homogeneous in many respects for all participants, and the development of applications that support sophisticated transaction management [Lantz86]. As noted by Sarin and Greif, systems can be designed and developed using a set of tools for conferencing, or a conferencing facility can be constructed to take advantage of single user applications [Sarin85].

From a pragmatic point of view, a system to support single user applications may be the better choice. To achieve some of the other objectives stated by Sarin and Greif, an architecture is needed that is slightly different then the one proposed by Lantz. One of the key additions to the proposals by these researchers is a User Interface Management System [Pfaff85, Hill86, Jacob86, Olsen86, Szekely88]. This addition will permit each user to have a different view of the same information coming from the application. This eliminates many of the problems of incompatible terminal types or system functional capabilities, especially with systems from multiple vendors.

Proposed architecture
One or more instances of the workstation components would be replicated, one for each conference attendee.



The architecture of a real-time conferencing system has the following components (see proposed architecture figure):

Application	User developed code. The output of the application is a data stream to be interpreted by a User Interface Management System. No special coding is needed for multi-user conferencing.
Session Management	Interface between the application and the end-user's User Interface Management System (UIMS). This code implements policies defined by the Dialog Management code (see next item). The policies may include who is allowed to send input to the application, who is allowed to see the output, etc. This code may also need to perform protocol translations if the end-user UIMS' are not identical.
Dialog Management	This is a set of functions that create the context in which the conferencing system exists. This may be one of the conference contexts implemented by CVIEW or other contexts. This code sets the access policies that are enforced by the Session Management code.
Network Driver	Acts as a multiplexor (multi-cast) for the Session Management code. It also handles network functions for the Dialog Management code.
UIMS	The User Interface Management System provides a media independent interface to an application. Each user can have a separate view of the data presented by the application. The user can define the interaction techniques that they desire to use.

Conference Management Interface	The user interacts with the Dialog Management through the Conference Management Interface. The user accesses the conferencing system and performs conferencing functions through this interface. Part of the information sent by the Conference Management Interface to the Dialog Management is a description of the current view being used by the UIMS. This gives everyone the ability to select the same view so they have the same frame of reference for discussion and interaction.
Workstation Manager	The physical resources of the workstation are handled by this code [Lantz86b]. Part of the Workstation Manager is the windowing system.

This architecture provides many of the features needed to meet the goals set forth in prior discussions of real-time conferencing systems [Sarin85, Lantz86]. With a simple extension, Lantz's model is subsumed by this model. We are currently looking at prototyping the model presented here in order to validate the concept.

Summary

Many of the issues in real-time computer conferencing are surfacing. The issues are both technical and social. Several implementations of application independent conferencing now exist. These are giving us an opportunity to explore this new technology. On the technical side, there remain questions on how these systems should be structured in the future. Windowing systems, User Interface Management Systems and their successor technologies will greatly affect the ways in which we design conferencing systems. As we develop distributed computing technologies, additional opportunities will appear. There are certainly tradeoffs between current technologies and those we expect to be available in the future.

The surface of the social aspects of real-time conferencing has barely been scratched. These may be the more difficult to recognize and analyze. Yet the benefits of real-time conferencing can be tremendous. Even in a relatively specialized area such as remote debugging and maintenance, the economic payoff of using this technology can be substantial. The same is true for education, where students can remain in their offices while the instructor can be trans- or inter-continental. The uses of this technology are really limited by the imagination of the people who choose to exploit it.

Acknowledgements

Many people helped to shape my thinking on the design of CVIEW. Thanks are due to Leon Nemerever who introduced me to a conferencing facility he had been working with for several years. Some of his insights into multi-user conferencing have been included in this paper. Joyce Graham and David Boloker at the IBM Cambridge Scientific Center gave me inspiration. My manager at the time, Jerry Waldbaum, was patient and supportive during the design and development of CVIEW. Finally, Keith Lantz, Sunil Sarin and Irene Grief wrote articles that helped clarify my thoughts on this subject.

Bibiography

[Engelbart68] English, W. K. & Engelbart, D. C. (1968). A Research Center for Augmenting Human Intellect. *Proceedings of the National Computer Conference*, pp. 395-410, IFIPS.

[Engelbart82] Engelbart, D. C. (1982). Toward High-Performance Knowledge Workers. *Office Automation Conference Digest*, April 1982, pp. 279-290.

[Pferd79] Pferd, W., Peralta, L. A. & Prendergast, F. X. (1979). Interactive Graphics Teleconferencing. *Computer*, vol. 12, no. 11, pp. 62-72, IEEE.

[Foster84] Foster, G. (1984). *CoLab, Tools for Computer-Based Cooperation: A Proposed Research Program*. Berkeley: University of California, Berkeley, UCB/CSD 84/215.

[Forsdick86] Forsdick, H. (1986). Explorations into Real-time Multimedia Conferencing. *Proceedings of the Second International Symposium on Computer Message Systems*, Sept. 1986, pp. 331-347, Elsevier Science Publishers B.V. (North-Holland).

[Aguilar86] Aguilar, L., Garcia-Luna-Aceves, J. J., Moran, D., Craighill, E. J. & Brungardt, R. (1986). Architecture for a Multimedia Teleconferencing System. *Proceedings of the SIGCOMM'86 Symposium on Communications Architectures and Protocols*, pp. 126-136, ACM, August 1986.

[Poggio85] Poggio, A., Aceves, J. J. G. L., Craighill, E. J., Moran, D., Aguilar, L., Worthington, D. & Hight, J. (1985). CCWS: A Computer-Based, Multimedia Information System. *Computer*, October 1985, pp. 92-103.

[Sarin84] Sarin, S. K. (1984). *Interactive On-Line Conferences*. PhD Thesis, Massachusetts Institute of Technology, MIT/LCS/TR-330.

[Sarin85] Sarin, S. & Greif, I. (1985). Computer-Based Real-Time Conferencing Systems. *Computer*, October 1985, pp. 33-45, IEEE.

[Lantz86] Lantz, K. A. (1986). An Experiment in Integrated Multimedia Conferencing. *Conference Proceedings of CSCW '86*, Austin, Tx., December 1986, pp. 267-275.

[IBM] IBM, (1985). *Cooperative Viewing Facility: Version 2 General Information*. Cary, N.C.: International Business Machines Corporation, Cary, N.C.

[Begeman86] Begeman, M., Cook, P., Ellis, C., Graf, M., Rein, G. and Smith, T. (1986). PROJECT NICK: Meetings Augmentation and Analysis. *Conference Proceedings of CSCW'86*, Austin, Tx., December 1986, pp. 1-6.

[Pfaff85] Pfaff, G. (ed.). (1985). *User Interface Management Systems*. New York: Springer-Verlag.

[Hill86] Hill, R. (1986). Supporting Concurrency, Communication, and Synchronization in Human-Computer Interaction: The Sassafras UIMS. *ACM Transactions on Graphics*, 5(3), July 1986, pp. 179-210.

[Jacob86] Jacob, R. (1986). A Specification Language for Direct-Manipulation User Interfaces. *ACM Transactions on Graphics*, 5(4), October 1986, pp. 283-317.

[Olsen86] Olsen, D. (1986). MIKE: The Menu Interaction Kontrol Environment. *ACM Transactions on Graphics*, 5(4), October 1986, pp. 318-344.

[Szekely88] Szekely, P. (1988). *Separating the User Interface from the Functionality of Application Programs*. PhD Thesis, Carnegie Mellon University, CMU-CS-88-101.

[Lantz86b] Lantz, K. A. (1986). On User Interface Reference Models. *SIGCHI Bulletin*, 18(2), October 1986, pp. 36-42.