Sharing views and interactions with single-user applications

Saul Greenberg

Advanced Technologies[†]
Alberta Research Council
6815 8 St NE
Calgary, Alberta, Canada T2E 7H7
phone: (403) 297-2674

Abstract—Although work is frequently collaborative, most computer-based activities revolve around software packages designed to be used by one person at a time. To get around this, people working together often talk and gesture around a computer screen, perhaps taking turns interacting with the running "single-user" application by passing the keyboard around. However, it is technically possible to share these unaltered applications—even though they were originally designed for a single user only—across physically different workstations through special *view-sharing software*. Each person sees the same image of the running application on their own screen, and has an opportunity to interact with it by taking turns. This paper discusses the various roles and responsibilities of the view-sharing software that must be considered during its design and evaluation: view management, floor control, conference registration by participants, and handling of meta-level communications. A brief survey of existing shared view systems is provided for background.

Keywords: computer supported cooperative work, real time conferencing, shared window systems.

1. Introduction

Tele-conferencing and video-conferencing often support only one part of the collaborative process, that of bringing people together. Yet most real meetings require not only the people, but also the materials and on-going work participants wish to share with others. These include notes, documents, plans and drawings, as well as some common work surface that allows each person to annotate, draw, brainstorm, record, and convey ideas during the meeting's progress. Given that an individual's work is commonly centred around a workstation, the networked computer can become a valuable medium for collaborators to share work with each other.

The ideal approach towards building software systems that support real-time sharing of work would recognize the existence of each participant and his or her niche in the collaboration. Such software is called "collaboration aware" (Lauwers and Lantz, 1990). For example, groupware applied to document collaboration could: recognize the roles of the primary writers, reviewers, and readers; adjust access permissions to reflect these roles; keep track of version differences; and enhance communications between the various collaborators (Leland *et al*, 1988). It is unlikely, however, that collaboration aware systems will have a major impact on the market in the next few years. Not only are they technically difficult to build, but the prerequisites for design are lacking—we really know very little about how people work together. As a result, we will probably see "single-user" applications—products designed to be used by one person at a time—far outstripping collaboration-aware systems for years to come.

An alternative approach, and the theme of this paper, stems from the old idea of taking a single-user application and sharing it between participants of an on-line meeting through a "shared screen" or "shared window". Each participant would have an identical view of the running application and an opportunity to interact with it. Special "*view-sharing*" software would allow *any* unaltered single-user application to be brought into a meeting; the application itself would have no awareness that more than one person was using it. The view-sharing software's responsibilities include maintaining consistent shared views, managing floor control between participants wishing to interact with the application, registering participants, and allowing attendees to gesture and annotate around the shared view.

Although simple in idea, sharing views and interactions with single-user applications can augment significantly people's ability to work together. Several possibilities are listed below.

1. Participants can be geographically dispersed. Given an underlying network of sufficient bandwidth and a voice

[†]email: saul@noah.arc.cdn or greenberg.chi@xerox.com

channel, small groups of physically separated people could easily share their computer-based work in real-time.

2. Sophisticated textual and drawing tools become available to the group. When compared to traditional media such as paper or the whiteboard, shared single-user applications provide not only functional richness but increased flexibility for the group to compose, manipulate, and save the objects on display (Stefik *et al*, 1987). Similarly, specialized applications (such as idea outliners) give participants a tool more suited to their task.
3. It is easier for people to take turns accessing the work surface, simply because they do not physically get in each other's way. (Consider the inherent awkwardness of five people drawing on a small physical whiteboard.)
4. Shared views are useful for "over the shoulder" consultation. The expert can quickly provide assistance by asking the novice to point out the problem, by "taking the user for a ride" through the solution, and by watching the novice attempt it himself (Engelbart and Lehtman, 1988).
5. Shared views can be used for presenting material to an audience, and as a medium for allowing the audience to reference and use the material during the course of the meeting.

The list is by no means exhaustive. Any collaborative process—information sharing, coordinating activities, interactive design, joint supervision, keeping tabs on progress—can benefit from shared views.

The rest of this paper will focus exclusively on sharing views and interactions with single-user applications between participants in a computer meeting. We will assume that all meetings are augmented by (at least) a voice channel. The first section provides a brief survey of existing implementations, from a visionary first step in the 1960s, to the sophisticated multi-media conferencing systems of today, through to the new visions for tomorrow. The subsequent section describes the roles and responsibilities of the view-sharing software that must be considered by the designer or evaluator of such systems. The paper closes with a brief discussion of several outstanding issues facing view-sharing systems.

2. A Brief Survey of Shared View Systems

Shared views are far from new. This section surveys some of the work that has been on-going since the mid-1960s.

2.1 The First Steps

NLS. Over twenty years ago, the visionary Doug Engelbart built what is probably the first shared screen conferencing as part of his NLS system (Engelbart and English, 1968). Six displays were arranged on a table so that a group of twenty participants could see the screens. While only one participant could control the screen, other participants could control a large arrow (a telepointer) visible on all screens but invisible to the application.

Augment. As Engelbart's work matured to the Augment system (a commercialized version of NLS), he expanded the shared screen conferencing software to incorporate distance conferencing, true floor control, dynamic entry and departure of participants during the conference, and a virtual terminal screen-image so that dissimilar terminals could be used (Engelbart, 1982 and 1984). For turn-taking, the controller explicitly hands off the floor to another participant.

2.2 Continuation

Doug Engelbart's work aside, most of the early effort in remote conferencing emphasised video. It was not until the mid-1980s that researchers returned to serious consideration of the computer to support group work.

RTCAL is a remote meeting tool that supports meeting scheduling by building a shared information space from participant's on-line calendars (Sarin and Greif, 1985) . During the meeting, each participant sees the shared calendar and their own private calendar. A chairperson oversees all activity, decides who has control of the shared view (each person can request control by entering a special control character), and is the only person who can terminate the conference. A small summary window indicates what the conference is about, who is present, who the chairperson is, and who currently has control. Dynamic registration of participants is allowed.

MBlink allows multiple workstations to share a bitmap (Sarin and Greif, 1985). It is unique in that all participants' cursors are visible on all the shared screens. Distinctive cursors identify their owners.

Cantata is a Macintosh-based suite of programs, featuring a sophisticated multi-person text-based "talk" system; a message broadcasting system; and a shared view system (Chang, 1987). Additionally, a specialized tool called the *Participant Construct System* supports knowledge acquisition from groups of persons working cooperatively. The shared view sub-system, called *switchboard*, shares access to a participant's serial port through a terminal emulator, rather than sharing the standard applications running on the Macintosh. The switchboard's turn-taking mechanism is described in Section 3.2 and illustrated in Figure 3.

Shared X. Rather than share the complete screen, a person can selectively choose and share one or more "public" windows on the display through "window sharing". A participant can then display both private and public windows on his screen, with public windows perhaps representing multiple conferences. Hewlett-Packard's *SharedX* is one example of this genre (Garfinkel *et al*, 1989). Based upon the X window system, SharedX is a centralized window server that interposes itself between a running application and the physical servers controlling each participant's actual workstation. It can be thought of as a switch that redirects X protocol to and from several X servers on behalf of applications.

Registration is primitive. Through a "ShareTool", a participant can make a duplicate of her window appear on another person's display. Conversely, one may "pull" another person's window across to ones own display through a "PullTool". SharedX also provides a primitive protocol for manipulating floor control, on top of which several more sophisticated floor control interfaces have been constructed. Telepointers are also available.

Dialogo. Most shared view systems follow a centralized architecture, where a single executing application is shared amongst all participants. The user's input is directed to that application, and the application's output is sent to every participant in the meeting. An alternative architecture, suggested and implemented by Lantz (1986) in a system now called *Dialogo*, replicates each application at every workstation to minimize network traffic. Although a controller's input is sent to all replicated applications, each application is responsible for generating its own output on its own workstation without going over the network. As long as synchronization is maintained, all views should be consistent. Similar replicated architectures included BBN's *Diamond Multimedia Conferencing System* (Crowley and Forsdick, 1989) and an early version of *Rapport* (Ensor *et al*, 1988).

There are several disadvantages to replicated architectures (Ensor *et al*, 1988). The software must be available at every workstation; there may be contention for physical devices; states local to a specific machine may be referenced; it may be impossible to guarantee consistent views across workstations using this architecture. A good critique of the issues in replicated architectures is provided by Lauwers, Lantz and Romanow (1990).

2.3 Recent Work

The current genre of shared view systems is becoming quite sophisticated. Systems are more complete and polished in appearance. Some include media aside from text and graphics, such as voice, still video, and motion video imagery. Others are tailored for particular meeting styles.

Computer-supported meeting rooms. Xerox PARC pursued the idea of computer support for face-to-face meetings in an experimental meeting room for small groups known as *CoLab* (Stefik *et al*, 1987). The room is arranged with one workstation per participant, as well as a very large touch-sensitive screen and stand-up keyboard. *CoLab* employs customized, collaboration-aware software, rather than the general single-user applications emphasised in this paper. Three systems were built: *Boardnoter*, a shared chalkboard; *Cognoter*, a tool for brainstorming and idea organization; and *Argnoter*, a tool to organize and evaluate arguments.

In contrast, the *Capture Lab*, another computer-supported meeting room, uses quite a simple view sharing setup (EDS, 1988). Each participant has a private workstation (a Macintosh), and has opportunity to gain serial control of a group workstation running a large screen at the front of the room. Work from the private workstation may be copied and pasted to the group workstation and screen. The best part of the Capture Lab is that its construction emphasised the need for careful design of all aspects of the room (Mantei, 1988). The subtle effects of seemingly trivial items such as seating, viewing distances between participants, availability of a front screen, and access protocols had a profound effect on the way the computer-supported meeting is run.

Multi-media conferencing. There is a proliferation of multimedia systems for sharing views. *Rapport*, for example, uses the X window system to support a multimedia conferencing system (Ensor *et al*, 1988; Ensor, 1989). A voice channel is explicitly integrated into the conferencing software (other systems reviewed here assume that the voice channel is handled separately). Rapport also supports multiple telepointers, as well as pictures of the participants themselves. Participants can point to one another and "raise their hands" for attention.

EMCE also integrates a voice channel along with the shared application (Garcia-Luna-Aceves *et al*, 1988). Floor control is handled automatically by the system through a distributed dialogue-activated collision-sensing algorithm, which notes pauses and handles any contention for the floor. Other participants can also signal the current floor-holder when they would like a turn.

There are several other notable examples of shared-view conferencing systems. Suzuki *et al* (1986) describe their real-time electronic conferencing system based on distributed Unix, while Lester Ludwig of Bellcore is furthering a centralized multi-media shared view system that integrates sound and video into the SharedX platform.

Commercial systems. On the commercial front, Farralon Software sells a simple, inexpensive but surprisingly effective shared-screen facility for the Macintosh called *Timbuktu* (Farallon, 1988), described in Section 3.3 and illustrated in Figure 4. A list of earlier commercial products can be found in Sarin and Greif (1985).

2.4 Innovative Directions

VideoDraw. Not all work is performed using a computer. One exciting possibility allows participants to share video projections of a workspace. Tang and Minneman from Xerox PARC built a prototype system called *VideoDraw* for video sharing of a virtual whiteboard between two participants (Tang, 1989). Each participant's work surface is a horizontally-mounted video screen, covered by a translucent drawing surface that can be marked with a pen. A video camera above the surface transmits the corresponding image to the other participant. The effect is a truly-shared "whiteboard". Each person sees not only the annotations and erasures of the other, but also the hands as they draw over the surface. People working together have opportunity to place their arms on top of the projected arm, so that their pen tips are literally on top of each other.

As a variation, several research laboratories are exploring the possibility of "video hallways" for casual interaction between remote sites. Two examples are the *TeleCollaboration* project (Corey *et al,* 1989), and *Cruiser* (Root, 1988; Fish, 1989). Users can "walk" the hallways and offices of the remote sites through remote-controlled video cameras. On their own screen, they can see who is around, start informal conversations, engage others in coffee room chit-chat, and so on. If video hallways were combined with ideas such as *VideoDraw*, we may see the start of the computer-supported spontaneous meeting that allows people to develop, jot down and share ideas in a non-formal situation.

Virtual reality allows people to interact within a three-dimensional world by using a head-mounted display, a data-glove, and a 3-D audio display. The head-mounted display is a helmet that contains a screen for each eye; projected 3-D images are synchronized for true binocular vision. When the person moves his head, his view of the 3-D space is adjusted accordingly. The data glove is an input device that converts hand gestures and relative positions into computer-readable form. When linked into the 3-D space, the user can see his gloved hand, and use it to manipulate his virtual objects. Finally, the audio display synthesizes audio cues so that the sound heard reflects the user's position in the virtual space. Off the shelf equipment is now available from VPL Research Inc (California).

The relevance of virtual reality to shared workspaces becomes clear when two or more people interact within the virtual space. Imagine a conference held in a virtual room, with attendees milling about, holding private conversations, and viewing and manipulating some of the available 3-D entities. Science fiction? Not quite. The first demonstration of VPL's *shared virtual reality* occurred on June 7, 1989 in San Francisco.

Shared Alternate Reality Kit. *SharedArk* is a workstation-based graphical model of a shared virtual yet physical world—a two-dimensional "flatland"—used for teaching students physics (Smith, 1988). Students can wander through flatland and manipulate physical objects with a mouse-operated hand. Unlike most virtual worlds, flatland is populated by all the people travelling in it. Students may accidentally encounter each other (one will see another person's hand). They then have opportunity to open an auxiliary video and audio connection for more direct communication. Within *SharedArk*, students can form collaborations on shared simulated physics experiments and jointly edit text and graphics.

3. Roles and Responsibilities of the View-Sharing Software

There are a number of design decisions that must be considered when building a comprehensive workstation-based view-sharing system. The responsibilities of the system fall into four major roles, as listed below.

A. The View Manager. The distributed view of the running application should follow (more or less) the "what you see is what I see" (WYSIWIS) abstraction, in which everyone sees the same image on their screen or window (Stefik *et al*, 1987). The View Manager is responsible for synchronizing and transmitting these views.

B. The Chair Manager. All participants should be able to interact with the application in a reasonable manner. Since the application is built to handle only one input stream, a floor-control mechanism that gives control to only one person at a time is usually desirable. The Chair Manager is responsible for setting or changing the floor control policy, for coordinating and enforcing turn-taking between participants, and for sending the selected input stream to the application.

C. The Registrar. Conference "registration" by the Registrar addresses four issues: conference set up and tear down; entry and departure of participants while the conference is in progress; access control; and feedback of the conference's current status.

D. The Meta Manager. Participants should be allowed to talk "around" the application through gestures and annotations without actually affecting the application. The Meta Manager is responsible for separating and controlling these meta-interactions.

Each manager is discussed in greater detail below.

3.1 The View Manager

Given a homogeneous terminal-based computing environment, creating a WYSIWIS View Manager is fairly straightforward. One merely has to tap into the application's output stream and send a copy of that stream to each of the other participants' terminals. Assuming that all connected participants are running compatible terminal types with the same initial terminal and screen configuration, each screen would then show the same image (Figure 1).

The real world is heterogeneous, and incompatible terminal types are likely. In this case, the View Manager would probably have the application write to a "virtual terminal", and then translate the output stream so that the same view (or a close approximation) appears across the different terminals (Engelbart, 1982) (Figure 2).

The situation is exacerbated in a windowing environment, where a shared view is represented in one of several windows on a workstation screen. Windows can normally be overlapped, resized, and destroyed at a user's whim. Four such cases are described below.

1. User resizing of a shared window is a problem. The View Manager may "fix" the window to a pre-determined size; allow windows larger (but not smaller) than the one the application expects; or alter the size of all windows across all workstations to reflect the single user's request. Alternatively, the manager may create a window that is a "viewport" to the underlying shared view, where the participant can pan across the image if his window is smaller than the actual size of the virtual view.
2. Window movement is usually handled by the workstation's window manager, and does not usually require intervention by the View Manager. Problems arise when the window is covered and then uncovered, for the window system may send a "window repair" request to the application. If this were to occur, all participants' views may be refreshed unnecessarily. As an alternative, backing store (if available) may be used to save the bitmap of the occluded region, or the View Manager may store and restore the particular window based on its own internally maintained virtual image.
3. Window destruction is analogous to leaving the conference. The registrar should be notified, and none of the other participant's displays should be damaged. "Window destroyed" notifications sent to the application must be intercepted and handled gracefully.

Other issues specific to shared window systems are raised in Lauwers and Lantz (1990).

3.2 The Chair Manager

A truly shared application requires that each participant is able to interact with it. Technically, the simplest approach is to have all input from all users merged into a single stream on a first-come, first-served basis (Figure 1). This sounds chaotic in principle—just imagine several people typing at the same time, with letter insertion interleaved into the text depending upon who struck the last key! However, stream merging can work well in practice. Our experience suggests that voice-mediation is a reasonable way of controlling casual sharing of applications between small groups (2–3 people), provided that the consequences of accidental simultaneous interactions are not disastrous.

As groups become larger, more advanced forms of floor control are usually desired. In Figure 2, the Chair Manager arbitrates turn-taking between the participants, and usually restricts control of the floor to one person at a time. At issue are the several ways that control is relinquished between participants, and what floor control policy is actually implemented. For example, four methods for acquiring and releasing the floor are described below.

Ring-passing. The participant currently in charge must explicitly release control before anyone else can assume it. Although this guarantees that a person can retain control of the floor, tension can arise between a participant who will not release control and others who wish to acquire it. Even in a benign setting, the current controller still has to remember to release the floor after he completes his turn.

Pre-emptive. Any participant may grab control of the floor at any time. This can lead to excessive interruptions in an aggressive conference.

Time slices and time outs. A person may have a set time-slice for control, after which the floor is taken away from him. More reasonably, the floor can be released for anyone's acquisition if the current controller has been idle for a set time period. Although a controller cannot be interrupted while "speaking", the floor is automatically available after a suitable pause.

Moderated. A designated participant may act as a chairperson who is responsible for passing control to and from the other attendees.

Other explicit floor-control protocols include round robin, queuing, random access, and so on. The preferred style will depend upon the group task, the size of the group, the politics of the group's interactions, and the application itself. Surprisingly, there has been no attempt to evaluate these different methods in existing shared view systems.

From the user's perspective, how is explicit floor control activated? In a windowed environment, a separate window, menu, or panel may be used to control turn-taking. Perhaps the name of the active controller is added to that panel, or placed in the shared window's title bar, or appended to the cursor. Consider the Cantata Switchboard (Chang, 1987), shown in Figure 3. The example illustrates four people sharing an electronic bulletin board through a queue floor control policy. The queue icon in the upper left indicates that the viewer (identified as "self") is waiting in line; the name list below shows her position in line in relation to the other participants. As the other participants gain and release control, she advances up the queue until it is her turn. The icon changes to a terminal pictogram, and control is kept until the icon is selected again. The line can be re-joined by another click on the queue icon.

Managing the floor is difficult when full screens are shared. There is no place to display a control panel for turn-taking, nor is there any room for feedback without disrupting the shared view. Although turn-taking may be controlled completely by special keyboard sequences, there is no room to give the participant feedback upon the state of the floor or meeting, such as who currently holds the floor and who the other participants are.

3.3 The Registrar
How do people enter and leave a shared view session? Technically, the easiest approach is to require that all participants be known before-hand to a "Registrar" (Figure 1). Connections are made between participants during conference setup only, and all channels remain open until the meeting terminates. This is reasonable for short meetings with few participants. One example is a person asking for momentary "over the shoulder" assistance in his work.

Dynamic registration is usually desirable for long-term, larger conferences. Any participant should be able to join and leave the shared view conference at any time. The Registrar must keep track of who has entered and left the conference, and alert the other managers to that fact (Figure 2). For example, the View Manager must be sure to synchronize a new participant's view with the established common view (Lauwers and Lantz, 1990).

The Registrar must also handle the differing roles of each attendee. Participants may, for example, have different levels of access to the meeting. These include no permission to join the conference, observer status, full read/write permission, and a special preferred status in floor control. The Registrar must also coordinate the conference start-up and tear-down. Other duties could include making potential participants aware of the meeting.

Figure 4 illustrates a control panel from Farallon's *Timbuktu* screen-sharing package (Farallon, 1988). Through this panel, a user can set permissions for other "guests" (participants) to share his screen. In this case, there are three people sharing Saul's screen—one with observe only permission (designated by the eye icon on the guest list) and two with interaction permission (the hand icon). Access control for guests is controlled either globally (top right panel) or individually (the middle panel).

3.4 The Meta Manager
Conventional approaches to shared workspaces are usually concerned with the text and graphics of the running application. This view is deficient, for it does not include gestures, a vital part of the group interaction (Tang and Leifer, 1988; Tang, 1989). People will often talk around a shared view without interacting directly with the displayed application. Some of these "meta" actions involve:

- directing the group's attention to some aspect of the view (referring, retracing, emphasising);
- allowing more than one person to gesture around the view at the same time;
- providing feedback on the attentiveness of other participants;
- augmenting verbal dialogue; and
- leaving marks on top of the view that are transparent to the application.

The Meta Manager is responsible for handling these meta-level interactions between participants (Figure 2). Simple gesturing, however, is possible when the shared view includes the controller's cursor. In most window systems, moving the cursor without depressing the mouse button does not generate any application input. The controller can gesture with his cursor, and other participants will see the cursor move in their views. Another more general approach is to provide a "telepointer" which is a cursor visible to all participants but invisible to the application (Stefik, 1987). Each participant can turn their cursor into a telepointer on demand (perhaps a large arrow with their name on it), and gesture accordingly. More than one telepointer may be shown at a time, and the person using it does not have to be the application controller.

Leaving marks on the shared view is more complicated and, to our knowledge, has not yet been tried. Using a "tele-pencil" as opposed to a telepointer, a participant may write on top of the application. He may annotate portions of the view, circle items, or erase some or all of the marks. Although this approach will not work well for a constantly changing view, it is a reasonable approach for relatively static images.

3.5 Implementation

Share is a “policy-free” view-sharing system whose architecture closely implements the conceptual design shown in Figure 2 (excepting the Meta Manager). It is a simplified system built for testing purposes—although it employs windows within its interface, it only shares views of a virtual terminal, itself running within its own window.

Most of our efforts in *Share* has been in floor control. *Share* departs slightly from the conceptual design in that the centralized Chair Manager does not enforce a specific floor control policy—it just understands a simple protocol that, amongst other things, controls observe and write status of participants, and sets a time-out period. Instead, policy is defined in a *turntaking interface process*—one for each participant—that converts a specific floor control policy into a protocol stream sent to the chair. For example, pre-emptive floor control is implemented by having the turntaking process request the Chair Manager to assign write permission to the process’ owner and observe-only status to all other meeting participants.

The Chair Manager also mediates direct inter-process communication between the turntaking processes, allowing them to extend the floor control protocol and coordinate their behaviour directly. In one example interface, we have implemented a floor control policy where participants must request the floor from a moderator. Since the Chair Manager’s protocol has no notion of a “pending floor request”, it was implemented as a protocol understood by the turntaking processes themselves.

Although *Share* is a working prototype still under development, initial results are promising. Different floor control and registration policies are almost trivial to implement; each participant's turntaking interface may be specialized to reflect his specific political role in the meeting; floor control policies may be switched on the fly; and the turntaking interface becomes platform-independent.

4. Obstacles and issues

Given the systems and architectures described above, it appears that shared views have reached a reasonable level of maturity using conventional technology. Such is not the case. It is worth considering several obstacles and issues facing current and future systems.

4.1 Standards

One of the greatest limitations of any view sharing system and, for that matter, any multi-user system, is the lack of a presentation standard. View-sharing of text-based terminals is fairly straightforward due to the ASCII standard and integrated databases of terminal drivers. However, most modern workstations support window systems that are, with one or two notable exceptions, proprietary. Given the wide diversity of workstations within and between offices, it is unlikely that all participants will use or be familiar with the same window system.

Many vendors are now recognizing this problem and are embracing X windows as a platform-independent window system standard. Although it will still take several years for X to migrate to the office, it may be reasonable to assume that a X-based window sharing system will support future meetings where participants use a variety of different workstations.

The down-side is that X describes only the underlying window protocol, and not the “look and feel” of the user interface. Sharing applications with non-familiar interfaces is difficult for the user. While standards are being suggested, there is an on-going battle between the Open Software Foundation’s *Motif* and Unix International’s *Open Look* user interface. Similarly, vendors running their own proprietary window systems (such as Apple Macintosh and NeXT) may have too much at stake (and too much inertia with their user audience) to switch window systems in mid-stream.

4.2 Networks and Multimedia

Effective conferencing demands real-time feedback of the view shared between participants. Sharing all but the simplest text-based applications will require fast, high-bandwidth communication channels. When multimedia are included (such as graphics, voice and video), the demands on the network become much more stringent.

Most existing shared view systems are built to run over local area or specialized high-bandwidth networks, which seems at odds with the objective of bringing together participants from remote sites. Normal phone lines are too slow, and special high-bandwidth connections are either unavailable or costly. Until the technology is in place and high-speed lines inexpensive, real-time view-sharing that includes multi-media objects will not be a serious communications device between distant offices.

As a stop-gap measure, some companies are including a fax-like capability in their machines. Rather than supporting interactive sharing of an application, people can mail screen snapshots. One example is Wang’s *Freestyle* multimedia communications system that allows users to take a snapshot of a window, annotate it with voice and hand drawings, and then mail it over the network (Wang, 1989). The recipient sees an animated playback of the voice and drawing annotations.

4.3 Technical difficulties of window-sharing.

There are a variety of technical difficulties still facing effective design of window-sharing. Consider, for example, the difficulty of sharing screen cursors within window systems. Cursors are usually handled deep in the heart of the window manager. The sharing of single cursors and the display of multiple cursors may entail significant changes to the kernel of the system.

Lauwers and Lantz (1990) also raise the issue of workspace management. When the screen contains a mixture of private windows, multiple shared windows attached to one conference, and multiple conferences, the system must not only identify the windows belong together but offer some scheme for managing related windows cohesively.

4.4 Data Sharing
Another problem arises in how people share and manipulate data through the common view (Sarin and Greif, 1985). If the application runs within the data space of one participant (ie within his file space), that person runs the risk of his files being intentionally or inadvertently damaged by the other conference attendees. Additionally, other participants are restricted to the access permissions of that person, which may exclude them from gaining access to and importing their own work to the meeting. Access control remains a problem even when the application runs within a data space owned by the group.

A related issue arises from the possible desire of participants to have "side conversations". A large group may wish to split into sub-groups, each with their own copy of the shared view, perhaps comparing results later on. Since the shared view may reference common data, maintaining different versions is necessary to prevent each sub-group from overwriting the data space of the others.

4.5 The Human Component
There are a variety of non-technical issues concerning effective view-sharing meetings. First, not all participants may have experience with a particular shared application. Consider, for example, the problems of sharing an editor. Perhaps there are several on the system. Each person may favour ones own choice, and be unfamiliar with the others. If a participant's editor is not selected for sharing, he becomes a second-class citizen in the meeting. Even when a common editor is agreed upon, problems will arise if people are used to a customized version.

Second, there is no way to personalize a shared view. WYSIWIS can be overly restrictive, especially when participants have widely differing roles, knowledge and abilities. For example, a view used by a technical person may have too much detail for effective viewing by (say) a manager. This fundamental limitation can only be overcome when applications become collaboration-aware.

Third, we really know very little about meeting dynamics. Floor control in face to face meetings, for example, is often based upon quite subtle yet natural body language. Human conventions for being invited into and joining a meeting are also subtle. In comparison, the explicit floor control mechanisms and registration schemes now implemented in shared view systems seem unduly contrived and restrictive.

5. Summary

Sharing single user applications across workstations is just a first step in eliminating physical restrictions and distances between people working together. It represents a paradigm shift in user interface design for computing, emphasising the collaborative nature of most real work.

This paper has provided a survey of the many existing systems as a snapshot of the work available, and has offered a schema for designing future view-sharing systems. Yet the field is still young. Researchers are still immersed in technical difficulties and have barely begun the empirical studies necessarry to the design of truly effective collaborative interfaces.

References

Chang, E., Kasperski, R. and Copping, T. (1987) "Group coordination in participant systems." Technical report, Alberta Research Council, Alberta, Canada, September.

Corey, D., Abel, M., Bulick, S., Coffin, S. and Schmidt, J. (1989) "Multi-media communication: The US WEST advanced technologies prototype system." Submitted to the *Fifth IEEE Workshop on Telematics*, September 17-21, Denver, Colorado.

Crowley, T. and Forsdick, H. (1989) "MMConf: The Diamond multimedia conferencing system." Proceedings of the Groupware Technology Workshop, Palo Alto, California. August

EDS (1988) "The Capture lab." EDS Centre for Machine Intelligence, Ann Arbour, Michigan. Videotape.

Engelbart, D. and English, W. (1968) "A research center for augmenting human intellect." *Proceedings of Fall Joint Computing Conference*, **33**(1), p395-410, AFIPS Press.

Engelbart, D. (1982) "Towards high-performance knowledge workers." *AFIPS Office Automation Conference*, San Francisco, CA, p279-290, April 5–7.

Engelbart, D. (1984) "Authorship provisions in AUGMENT." In *Proceedings of the IEEE Compcon Conference*.

Engelbart, D. and Lehtman, H. (1988) "Working together." *Byte Magazine*, **13**(13), December.

Ensor, J. R. (1989) “Rapport: A multimedia conferencing system.” *The ACM SIGGRAPH Video Review Supplement to Computer Graphics*, **45**(5). ACM Press, Baltimore, MD. Videotape.

Ensor, J.R., Ahuja, S.R., Horn, D.N. and Lucco, S.E. (1988) “The RAPPORT multimedia conferencing system — a software overview.” In *Proceedings of the IEEE Conference on Computer Workstations*, p52–58, March.

Farallon (1988) “Timbuktu User’s Guide.” *Farallon Computing Inc.*, Berkely, California.

Fish, R. S. (1989) “Cruiser: A multi-media system for social browsing.” *The ACM SIGGRAPH Video Review Supplement to Computer Graphics*, **45**(6). ACM Press, Baltimore, MD. Videotape.

Garcia-Luna-Aceves, J.J., Craighill, E.J. and Lang, R. (1988) “An open-systems model for computer-supported collaboration.” In *Proceedings of the IEEE Conference on Computer Workstations*, p40-51, March.

Garfinkel, D., Gust, P., Lemon, M. and Lowder, S. (1989) “The SharedX multi-user interface user’s guide, version 2.0” Technical report STL-TM-89-07, Hewlett-Packard Laboratories, Palo Alto, March.

Lantz, K. (1986) “An experiment in integrating multimedia conferencing.” In *Proceedings of the conference on Computer-Supported Cooperative Work,* Austin, Texas, December.

Lauwers, J.C. and Lantz, K.A. (1990) “Collaboration awareness in support of collaboration transparency: Requirements for the next generation of shared window systems.” In *Proceedings of the ACM Conference on Human Factors in Computing Systems*, Seattle, Washington, April.

Lauwers, J.C., Lantz, K.A. and Romanow, A.L. (1990) “Replicated architectures for shared window systems: A critique.” In *COIS 90—Proceeding of the Conference on Office Information Systems*, Cambridge, Mass. April 25-27.

Leland, M.D.P., Fish, R.S. and Kraut, R.E. (1988) “Collaborative document production using Quilt.” In *Proceedings of the ACM Conference for Computer-Supported Cooperative Work*, Portland, Oregon, p206–215, September 26–28.

Mantei, M. (1988) “Capturing the Capture Lab concepts: A case study in the design of computer supported meeting environments.” In *Proceedings of the ACM Conference for Computer-Supported Cooperative Work*, Portland, Oregon, p257-270, September 26–28.

Root, W. (1988) “Design of a multi-media vehicle for social browsing.” In *Proceedings of the ACM Conference for Computer-Supported Cooperative Work*, Portland, Oregon, p25–38, September 26–28.

Sarin, S. and Greif, I, (1985) “Computer-based real-time conferencing systems.” *IEEE Computer*, **18**(10), p33-45.

Smith, R. B. (1988) “A prototype futuristic technology for distance education (working draft).” In *NATO Research Workshop on New Directions in Education Technology*, Cranfield, England, November.

Stefik, M., Foster, G., Bobrow, D., 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), p32-47.

Suzuki, T., Taniguchi, H. and Takada, H. (1986) “A real-time electronic conferencing system based on distributed Unix.” In *Proceedings of the Usenix 1986 Conference*, Atlanta, Georgia, p189–199, June 9–13.

Tang, J. C. (1989) “Listing, drawing, and gesturing in design: A study of the use of shared workspaces by design teams.” *Xerox Technical Report SSL-89-3.* April. Also as PhD Dissertation, Stanford University.

Tang, J.C. and Leifer, L.J. (1988) “A framework for understanding the workspace activity of design teams.” In *Proceedings of the ACM Conference for Computer-Supported Cooperative Work*, Portland, Oregon, p244-249, September 26–28.

Wang Corporation (1989) “Freestyle Multimedia Communication System.” Demonstrated at the *ACM Human Factors in Computing Systems SIGCHI conference*, Austin, Texas, June 5-9.

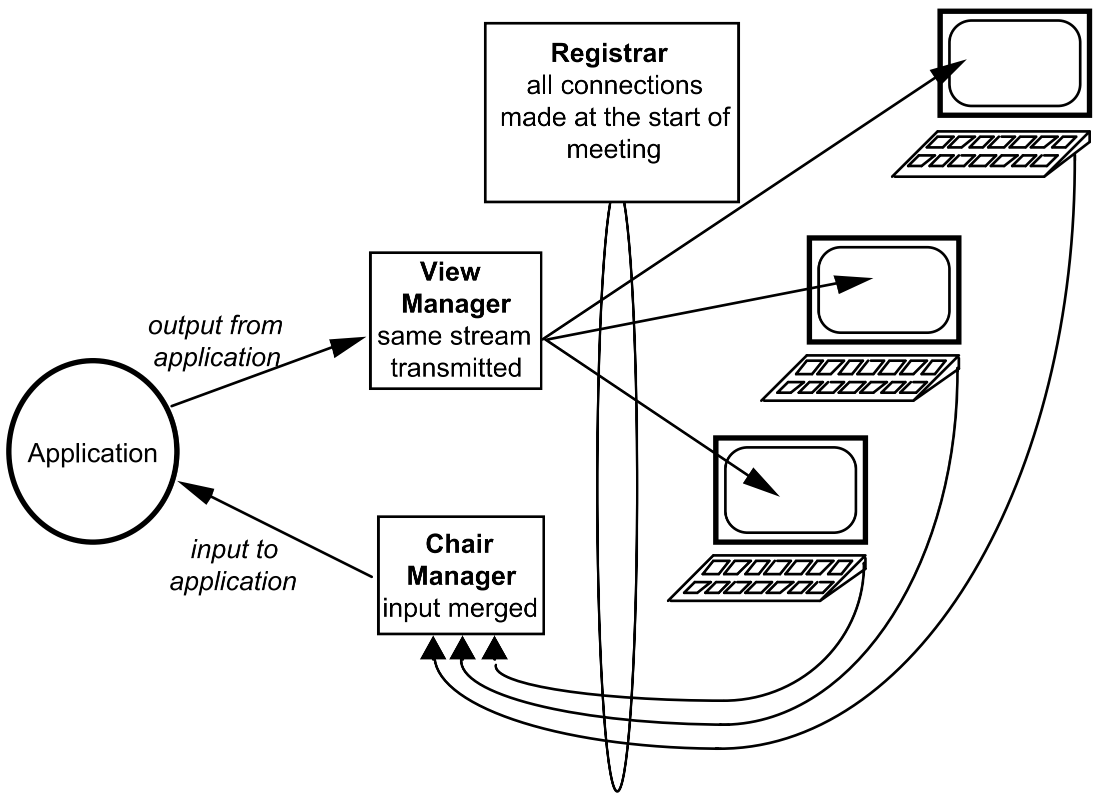


Figure 1: A simple view-sharing system for a homogeneous terminal-based computing environment. There is no explicit floor control, no dynamic entry and departure of participants, and no meta-level communication

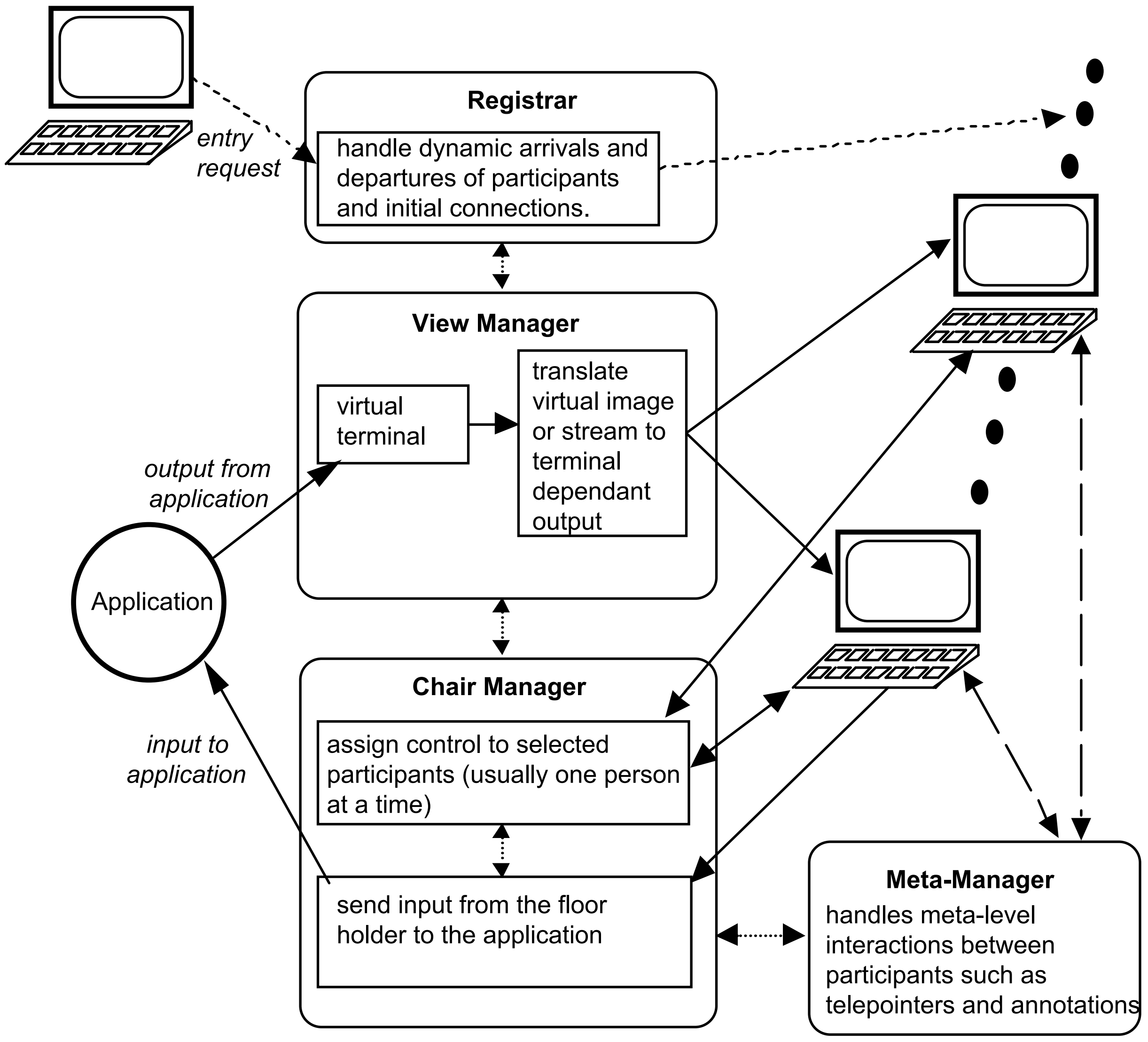
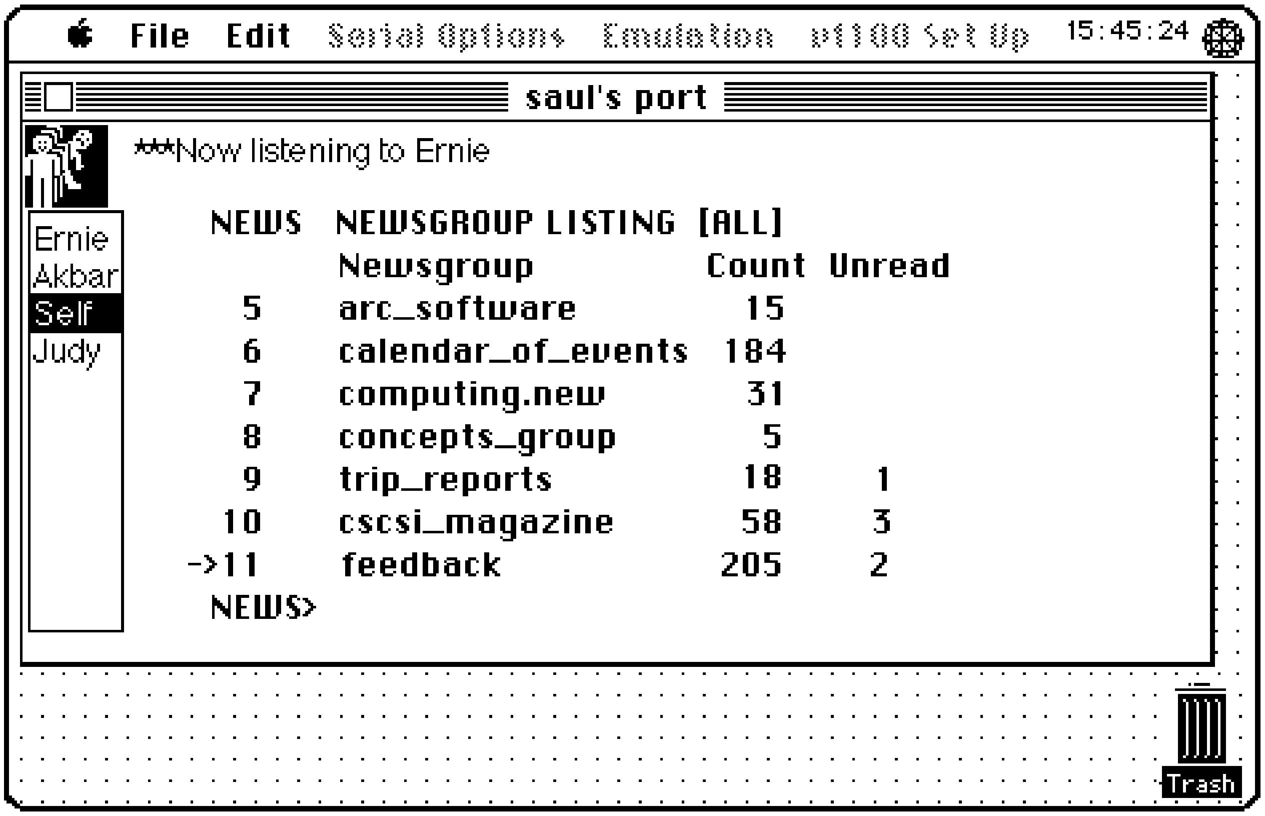

entry request
Registrar
handle dynamic arrivals and departures of participants and initial connections.
View Manager
virtual terminal
translate virtual image or stream to terminal dependant output
output from application
Application
Chair Manager
input to application
assign control to selected participants (usually one person at a time)
send input from the floor holder to the application
Meta-Manager
handles meta-level interactions between participants such as telepointers and annotations

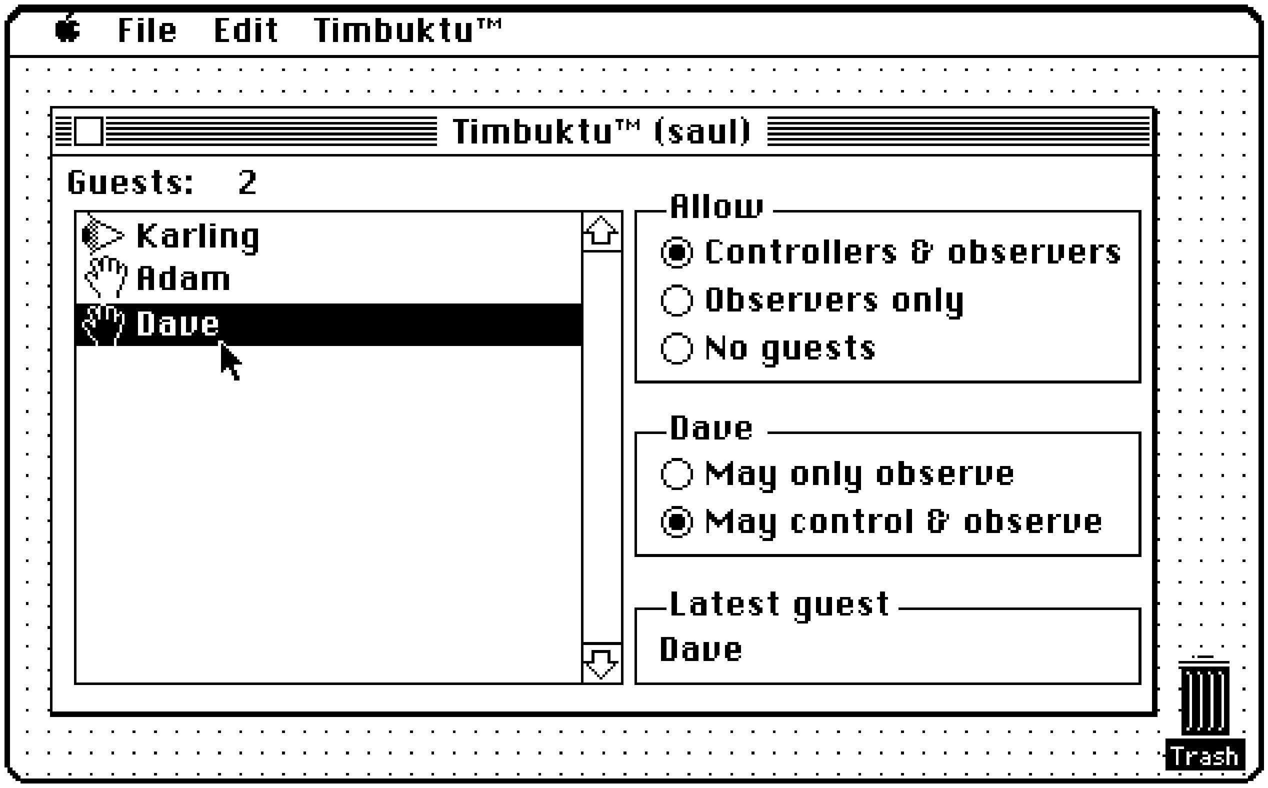
Figure 2: A more complex view-sharing system handling heterogeneous terminals, turntaking, dynamic registration, and meta-level dialog.



```
***Now listening to Ernie

    NEWS  NEWSGROUP LISTING [ALL]
          Newsgroup           Count  Unread
      5   arc_software          15
      6   calendar_of_events   184
      7   computing.new         31
      8   concepts_group         5
      9   trip_reports          18      1
     10   cscsi_magazine        58      3
   ->11   feedback             205      2
    NEWS>
```

Figure 3: The Cantata Switchboard showing four people sharing an electronic bulletin board.



Guests: 2

Karling
Adam
Dave

Allow
[✓ Controllers & observers]
Observers only
No guests

Dave
May only observe
[✓ May control & observe]

Latest guest
Dave

Figure 4: The Farralon Timbuktu control panel for setting access permissions for a shared screen.