A Brief Overview of the DCS Distributed Conferencing System

R. E. Newman-Wolfe, C. L. Ramirez, H. Pelimuhandiram,
M. Montes, M. Webb and D. L. Wilson
Computer and Information Sciences
CSE-301
University of Florida
Gainesville, FL 32611
nemo@cis.ufl.edu

Abstract

The Distributed Conferencing System (DCS) at the University of Florida is a distributed package providing real-time support for cooperative work. In this system, a set of mechanisms for conference management supports a wide range of floor control paradigms. The small, flexible message-passing interface to the conferencing management processes permits shared applications to be installed easily in DCS. Currently, DCS has applications that support concurrent development of text and graphic documents; remote demonstration, testing and debugging of programs; and automatic creation of transcripts of meetings including motions made and voting results.

1. Introduction

Research in distributed conferencing and shared applications has usually focused on a particular application, or on sharing an existing application without modifying the application code. We believe that cooperative work generally involves several kinds of shared applications, and that the framework for coordination of effort is significant in itself. Sharing should not be transparent for all applications, nor is the same architecture appropriate for all applications. DCS not only provides a cohesive set of shared applications suitable for collaborative preparation and demonstration of text, figures and software, but also a rich set of mechanisms for directing the collaborative effort. DCS monitors user behaviors to assist our study of how people interact when using these distributed collaboration tools.

DCS is written using Unix 4.3bsd sockets [Bach 86] [Leffler et al. 89] and XWindows [Johnson & Reichard 89] with the X Athena Widgets package. It runs concurrently with other tools; thus other programs and sources of information are available to conference participants. Versions of DCS have been running at the University of Florida since August 1990.

A unique aspect of DCS is the voting mechanism, which provides flexible, decentralized control of conferences. Within the text editing and graphics editing applications, fine granularity write and view

---
1 This work is partially supported by the University of Florida - Purdue University Software Engineering Research Center.

2 Unix is a trademark of AT&T, of course.
locks are provided (see below). Each user may define the degree to which she wishes to share views, or see ongoing work. This is a generalization of What-You-See-Is-What-I-See (WYSIWIS) [Stefik et al. 86, 87] that we call "What-You-See-Is-What-I-May-See" (WYSIWIMS). The WYSIWIMS paradigm is similar to the filtering provided in Suite, which propagates changes depending on syntactic or semantic integrity [Dewan & Choudhary 90]. Discussion windows facilitate communication among members of a conference by allowing a channel of "out-of-band" communications. A transcript of the discussion is preserved, allowing both asynchronous and synchronous cooperation. Users have access to control functions and information through a status window. In addition to these control features, DCS provides a suite of applications with the flexibility to add others easily.

The remaining sections briefly describe related work in groupware [Ellis & Gibbs 88], and some details of the structure of DCS. The abstract is necessarily brief, and more thorough explanations may be found in [Ramirez 91], [Wilson 91], and [Newman-Wolfe et al. 91].

2. Related Work in Groupware

Earlier systems for distributed collaboration date back to the 1960's and Engelbart's work with NLS [Engelbart & English 68] and have continued through MCC's Project Nick [Begeman et al 86]. Most office automation systems focus on only one application, and those that belong to a package usually have limited facilities for supporting shared work. Typical applications include: shared calendars and schedulers (RTCAL, MTICAL [Greif & Sarin 86]), shared draw screens (NLS [Engelbart & English 68], MBlink [Sarin & Greif 85], Colab [Foster & Stefik 86, Stefik et al. 87], Commune [Bly & Minneman 90], TWS [Ishii 90]), shared graphics editors (Xsketch [Lee 90]), shared text editors (Collaborative Editing System (CES) [Greif & Sarin 86, Sarin & Greif 85], Shared Books [Lewis & Hodges 88]) and hypertext browsers (Contexts [Delisle & Schwartz 86], NoteCards [Trigg et al. 86], TEXTNET [Trigg & Weiser 86], Intermedia [Garret et al. 86]). Some of these offer more than one medium for communication and cooperation. Another approach uses shared terminal emulators and other facilities for sharing existing serial applications without modification (Cantata [Chang 87], Augment [Engelbart 84], Dialogo [Lantz 86], Rapport [Ahuja et al. 88a], Timbuktu [Farallon 88]). These independent shared applications are more tightly coupled in DCS, reflecting the cohesive nature of the conference. DCS more closely resembles Share [Greenberg 90], MMConf [Crowley et al. 90], and Mermaid [Watabe et al. 90] in dealing with conferences as a whole. An excellent introduction to work in this area may be found in [Greif 88]. Additional sources include the proceedings of the conferences on Computer Support for Cooperative Work (CSCW 86, 88 and 90) as well as the Conference on Organizational Computing Systems (nee Conference on Office Information Systems).

3. DCS Description

3.1. Goals of DCS

DCS provides a distributed, real-time conferencing facility for the cooperative creation, review and execution of papers, proposals, and programs. Conferences may be long-term or short-term and are dynamic in that their constituency may change over time; they may also split or merge. Conference control is hierarchically democratic, and DCS's architecture is hierarchically distributed.
DCS uses Unix 4.3 sockets [Bach 86], [Leffler et al. 89], [Kernighan & Pike 84] and X-Windows [Johnson & Reichard 89] for implementation to allow it some degree of independence. By using "middleware" DCS will be able to run on a heterogeneous, distributed environment. Porting the system should be easy.

Shared applications may be made accessible within DCS by adherence to a small, message-passing interface within the application code, and by providing a minimal amount of information to DCS regarding the application name and usage. This allows DCS to be extended and customized.

Our intent is not only to provide an easily portable and extensible conferencing system, but to study the ways in which people interact when using a distributed conferencing facility of this type. From this, we hope to learn what features are critical to the success of groupware, and what resources are needed to support them. To this end, we are currently incorporating monitoring code in DCS to measure system performance parameters (such as number of messages, message size, response times, etc.), user behaviors (facilities used, conference interactions, preferences) and conference behaviors (control modes used, time spent sharing objects, lock collisions, lock preemptions). Direct user feedback will also contribute to our evaluation of DCS.

3.2. Functional Description of DCS

Important concepts in DCS are conference typing, user roles, voting, and independence. Conference types define the desired level of stability of the conference and how long the conference will last. User roles describe what a conference member may do. Voting resolves conference control and user defined issues. Finally, independence means that each user has control over the degree to which she shares with others.

DCS provides two types of conferences: short-term and long-term. A long-term conference persists, even when members are not logged in, until it is terminated by the last member resigning. Over this period, a record is kept of the members, the files, and access information. An effort is made to keep conference objects stable over system disruptions. Short-term conferences have no such records; they are automatically cancelled when the last member exits.

Within a conference, members may have the role of a voter, a non-voter or an observer. The roles are created to maximize the number of participants, yet preserve the integrity of the conference by limiting rights given to members. Voters will be permitted to participate in group decisions about conference actions, such as whether to allow a new user to join the conference, or what the disposition of a conference object should be. Both voters and non-voters have read and write access to all files within their conference. Observers are denied write access, but may view conference files and other objects.

All conference control activities are resolved using a general voting mechanism. These actions include changing the role of a user, merging two conferences, splitting a conference, and disposing of conference objects. In addition, DCS supports user-defined motions, which are not interpreted by the system but are only recorded and handled through the voting mechanism.

Off-line voting is the general control paradigm, so that the conference need not be suspended while a vote is under way. When a motion is made, a small dialog box with the motion and a checklist appears on
the screen of each active user. The user may then vote for or against, abstain, or defer on the motion. Unresolved motions are placed on a referendum list and members may vote on any motion for which they have not yet voted (deferral is not a vote). The referendum list may be examined by any member at any time. Once enough votes for a motion have accumulated, action may be taken. Conversely, if enough negative votes accumulate against the motion, then it will be removed from the referendum list and no action will be taken. Voting may be rollover or anonymous, short circuit or exhaustive. The results of voting are appended to the conference control log. By adjusting the roles of users, conference control ranges from dictatorship (one voter) to total democracy (all members are voters) to anarchy (no voters).

3.2.1. Extra-conference Activities

DCS provides some operations to users who are not active participants in a conference. Any user may ask for a list of conferences, initiate a conference, or request to join a conference. All operations other than direct conference file access through usual Unix mechanisms require the user to invoke DCS.

Each conference has its own directory with its own files. Members of the conference may access these files through DCS without having access through Unix [Kernighan & Pike 84]. Conversely, any user may gain access to conference files according to the usual Unix access modes without involving DCS. The default mode is no access, but the protection mode may be changed by the members of a conference.

3.2.2. Intra-Conference Control Activities

Within a conference, a member may make queries, make motions, and vote. Queries allow the member to review the status of the current conference or other conferences without leaving the current conference. Motions are either requests for actions within a conference, or are user-defined and are not interpreted by the system. Motions are usually subject to voting by the members (see above).

Control activities within a conference consist of changing user roles (joining, resigning, upgrading to voter, etc.); sending invitations to join; splitting and merging conferences; changing conference parameters; and determining the disposition of conference objects.

3.3. DCS Architecture

DCS is implemented as a group of cooperating processes. We term these the Central Conference Server (CCS), the Conference Manager (CM), the User Manager (UM). Of these, only the CCS need always exist and only the CCS needs to have an easily found port. Each active conference has a CM, and each user active in a conference has a UM. In addition, each application has its own logical Application Manager (AM) and architecture internal to itself. Currently, the applications we have developed all have an AM process with an application user process for each user working with that application, but this architecture is not required.

3.3.1. The Central Conference Server

The Central Conference Server (CCS) is a daemon that listens at a fixed address known to the process invoked by users. It provides coordination between and access to conferences. The CCS also monitors the performance of CMs and does error recovery.
When a user first invokes DCS, she obtains a UM that runs locally and is connected to the CCS. Only the status window is displayed, and the CCS may be queried about existing conferences or the user may request to join or create a conference. Requests to join a conference are forwarded to the appropriate CM, which will allow members to connect immediately, or will make a motion to allow a new user to become a member. Create requests cause DCS to prompt the user for descriptive information (for external listing) and conference type. The CCS then creates a new CM and a new directory, and connects the user’s UM to the new CM.

![Diagram of CCS connections to multiple conferences](image)

**Figure 1.** CCS connections to multiple conferences.

### 3.3.2. The Conference Manager

Each active conference has a Conference Manager (CM). The CM is responsible for providing basic conference control and for the discussion window. The main access point to the CM is the status window. Through the status window, the user may perform conference control activities described earlier, make queries, or request application windows. When an application window is requested by a user, the CM must determine whether an appropriate AM exists, invoke one if it does not exist, and connect the user to that AM. The CM also provides the AMs with access to voting and other conference control mechanisms through the message-passing interface. Finally, the CM is responsible for checking on AMs and the CCS to verify their operation.

### 3.3.3. The User Manager

The User Manager (UM) acts as a dialog manager for the individual user active in a conference. It routes information between the user’s input devices, the CM, and the user’s output devices. When the user requests an application window, the UM asks for information needed by the CM to connect the user to the
right AM. Once the CM insures the existence of the appropriate AM, that AM’s port is sent to the UM along with instructions for invoking the application window (AW) that will be connected to the AM. Once the UM starts the AW, it is no longer concerned with that application, leaving all the intra-application communication to be handled by the AM and AWs attached to it.

3.3.4. Application Managers

Each application in DCS has its own (logical) application manager. There may be only one AM for a particular application in a conference, or there may be several for the same application within a conference. For example, the execute window has only one manager per conference regardless of the number of execute sessions in progress. On the other hand, the graphics edit application has one Graphics Manager (GM) for each graphics file that is being edited, so there may be several GMs for a single conference. Conceptually, each graphics file has a GM and together, these form a graphics object.

By having AMs, the design of DCS is modular: stand-alone shared applications are easily introduced into its framework. This also distributes the workload within DCS in two ways. First, each process that handles multiplexing within an application can be relatively simple since they only handle a fraction of the communication load. Second, a new AM may be started on a different machine from the CM or existing AMs, so no one machine will become overloaded.

While the shared applications we have developed so far have all had an explicit AM connected to AWs running on user’s workstations, this architecture is not required by DCS. DCS does require that each application have a liaison communicating with the CM that acts as a virtual AM. This is needed for instance to clean up orphaned AWs when a user quits DCS without closing AWs first. Since the application may not have a fixed AM, but must have some internal communication abilities, DCS provides a message type for migration of the logical AM if needed. Thus an application could be a set of peer processes that pass AM duties among themselves as the set changes.
3.4. Base Windows

A user running DCS always has a status window. If the user has joined a conference, then she also has a discussion window. These are the base windows of DCS, and they provide support for distributed decision making and conference control.

3.4.1. The Status Window

The status window provides conference control and information. It is the only window that always exists, whether or not a user is inside a conference. Members may make motions, request information, and vote through the status window and its menus. Notices of actions that are requested, pending, or completed, as well as requested information, are displayed in the status window. The output of the status window that is relevant to the whole conference is buffered in a file as a record of conference actions. Users may scroll within the status window independently of one another.
3.4.2. The Discussion Window

Discussion windows allow real-time communication among collaborators. Each member has access to a discussion window that is local to the conference to which she belongs. The window, which is similar to a gossip window [Ramirez & Pelimuhandiram 90] is opened by default when the member joins the conference. All members of a conference have append-only rights to the top half, which may be viewed synchronously or asynchronously. By default, the user’s login name is prepended to all contributions she makes to the discussion window, but an anonymous mode is available to omit this if attribution would inhibit free discussion.

3.5. Application Windows

In addition to the base windows, several application windows may be brought up in DCS from a menu on the status window.

3.5.1. The Execute Window

Every member of a conference may create an execute window through which she has access to a shell. Currently, users may only execute a program producing line-oriented ASCII output and taking serial ASCII input. This window may be exported to other users, allowing them to see the execution of the program taking place in the source execution window. The execution window is the only window with a conference pointer. Only one member at a time controls the pointer and the input of each execution window. This seems to be a necessity for maintenance of consistency when serial applications are shared.

When the execution window is opened, the input control to the window belongs to the user who opened it. The user who has control of the input may pass control to another user at any time. The controlling member may specify which other active members may view the execution window, except that the member who created the execution window cannot be excluded. All other windows are synchronized in a WYSIWIS fashion to reflect every input and output of the controlling window.
Since only the output of the program's execution is sent to the slave execution windows, people can participate in conferences in spite of some differences in software and hardware [Ahuja et al. 88a],[Ahuja et al. 88b]. This tool provides the facilities for activities such as joint debugging of programs, and conducting tutorials or demonstrations remotely.

### 3.5.2. The Text Edit Window

The DCS system provides a concurrent text editor, permitting concurrent access to a shared document using fine granularity locks. Locked portions of text may begin and end between any pair of adjacent characters in the file, or at its beginning or end.

Two types of lock are used: the write lock and the view lock. The write lock is allocated on a first-come, first-served basis and it ensures exclusive write permission to a section of the file. Only one write lock may be held per text edit window. A write lock is held by the writer until it is explicitly released, at which time the changes will overwrite the old version of the section. The user holding a write lock may also place a view lock on the locked section. View locks prevent other users from 'looking over the shoulder' of the writer while the edit is in progress (i.e., neither update nor synchronous viewing of the
edited section is allowed). They do not prevent other users from viewing the 'old' version of the text in the locked region.

User have three modes available for viewing the contents of a text file: static, snapshot, and synchronous. Static viewing allows locked portions of files to be viewed as they were before modification began. Snapshot viewing causes the member's window to capture a snapshot of the current state of a locked portion of a file, while synchronous viewing causes changes to be seen as they occur. If the reader is viewing synchronously, she is not able to scroll within or edit the file until she returns to an asynchronous viewing mode. These latter two modes are disabled if the source is view-locked. By default, the reader views the text in static mode, seeing updates only when locked portions of text are released.

3.5.3. The Graphics Edit Window

Users may invoke a concurrent, object-oriented graphics editor from DCS. The stand-alone version of this editor is called Ensemble, and it is based on the tgif public domain graphics editor from UCLA. Objects may be manipulated in the usual ways.

Locking is used to prevent conflicts in the graphics editor. Locks are placed implicitly when a user selects an object: if a locked object is selected, an error message is printed in the banner and the selection is rejected. Changes are shown when the selection (and thus the lock) is released.

Ensemble allows users to preempt locks held by others, in the event that a user holds locks on some objects for too long while another user wishes to work with them. While we initially handled lock preemption through voting, it was cumbersome to obtain sufficient votes quickly enough to be worthwhile.
Abstract

The Distributed Conferencing System (DCS) at the University of Florida is a distributed package providing real-time support for cooperative work. In this system, a set of mechanisms for conference management supports a wide range of floor control paradigms. The small, flexible message-passing interface to the conferencing management processes permits shared applications to be installed easily in DCS. Currently, DCS has applications that support concurrent development of text and graphic documents; remote demonstration, testing, and debugging of programs; and automatic creation of transcripts of meetings including motions made and voting results.

Research in distributed conferencing and shared applications has usually focused on a particular application, or on sharing an existing application without modifying the application code. We

Instead, any editor in DCS may preempt locks at will, and we rely on their desire to cooperate to damp these urges.

Each user has a pointer, which may be exported to other users. Each user also has the option of viewing other users’ exported pointers. A user importing pointers will see another user’s pointer only if the other user exports her pointer. Pointers may be used for gesturing while discussing the contents of a graphics window, a process that we found demanded an audio connection as we note in the conclusions section. Although we attempted to avoid use of special equipment, close interaction while editing a canvas is one instance in which it is very desirable.

When a graphics edit window (or Ensemble window) is opened within DCS, it is automatically and transparently connected to any other Ensemble windows editing the same file. These AWs coordinate through the Graphics Manager (GM), one of which exists per graphics file for which an Ensemble window is open. The GM maintains the last stable version of the file, coordinates locking and updating, and multiplexes pointer information. Each Ensemble window process has its own copy of the object data base that it manipulates locally for display purposes, sending and receiving object updates when necessary by exchanging messages with the GM. Since messages are received serially by the GM, there are no ties and all requests are handled in a first-come, first-served manner. The rate of information exchange is very low compared to systems in which low level information is broadcast to application windows. Each AW may also deal with the updates in a manner appropriate to the section of the canvas displayed locally, since each Ensemble window is able to view the canvas independently. This provides excellent response time for
edits performed locally, with good response time for those performed remotely.

![Figure 8. Graphics edit window.](image)

4. Conclusions

DCS is a distributed real-time conferencing system running at the University of Florida. It provides flexible conference control and has the concept of multiple conferences. Within DCS, fine-grained concurrent text and graphics editors are available, as well as a discussion window and a general-purpose execute window for sharing a limited class of serial applications. Applications planned for DCS include software engineering tools (a code review tool) and a more general execute window.

Applications within DCS use a variety of paradigms for access: immediate and implicit access is granted for appending to the discussion and status windows, exclusive floor control is passed serially in the execution window, fine-grained locks are used in the graphics and text edit windows. We have not found locks to be restrictive, as has been feared [Ellis & Gibbs 8]; this may be due to the selection of the applications in which locks are used and the capability to remove locks that have been held ‘too long.’

Through the paradigm of the Application Manager, an interface to the Conference Manager of DCS, each shared application can implement the architecture most appropriate to its functioning. New applications are currently easy to add to DCS, provided that they follow an undemanding message passing protocol. We plan to automate the installation process so that even the small changes to the program required on addition of a new application may be performed quickly and correctly.

In using DCS we found occasions when an audio connection among users would have been desirable. For example, gesturing with pointers in a graphics window while conversing in the discussion window is clumsy. In future versions of DCS, we plan to incorporate a voice channel for communication that will enhance the interaction among users.
Our experience with DCS is still preliminary, and we intend to perform extensive monitoring in order to determine which features are most desirable and which are not. DCS gathers information on user behaviors and system performance with internal statistics gathering code. It is dangerous to presuppose the features or functionality that users will find most useful, so we plan to beta test DCS soon. The cost of using DCS will be to respond to a user evaluation questionnaire after three months of use. This information will guide future development of DCS.

5. References

[Ahuja et al. 88a]

[Ahuja et al. 88b]

[Bach 86]

[Begeman et al. 86]


[Crowley et al. 90]

[Delisle & Schwartz 86]

[Dewan & Choudhary 90]

[ Ellis & Gibbs 88]

[Engelbart & English 68]
[Engelbart 84]

[Farallon 88]

[Foster & Stefik 86]

[Garret et al. 86]

[Greenberg 90]

[Greif & Sarin 86]

[Greif 88]

[Ishii 90]

[Johnson & Reichard 89]

[Kernighan & Pike 84]

[Lantz 86]

[Lee 90]

[Leffler et al. 89]
Leffler, S. J., McKusick, M. K., Karels, M. J., Quarterman, J. S., The Design and Implementation of

[Lewis & Hodges 88]

[Newman-Wolfe et al. 91]

[Ramirez & Pelimuhandiram 90]

[Ramirez 91]

[Sarin & Greif 85]

[Stefik et al. 86]

[Stefik et al. 87]

[Trigg et al. 86]

[Trigg & Weiser 86]


[Wilson 91]