CCF: Collaborative Computing Frameworks*

V. Sunderam, S. Cheung, M. Hirsch, M. Grigni, S. Chodrow,
A. Krantz, I. Rhee, J. Sult, S. Olesen, P. Gray, P. Hutto, T. Goddard

Emory University
Atlanta, GA 30322, USA

Abstract

CCF (Collaborative Computing Frameworks) is a suite of software systems, communications protocols, and tools that enable collaborative, computer-based cooperative work. CCF constructs a virtual work environment on multiple computer systems connected over the Internet, to form a Collaboratory. In this setting, participants interact with each other, simultaneously access and operate computer applications, refer to global data repositories or archives, collectively create and manipulate documents or other artifacts, perform computational transformations, and conduct a number of other activities via telepresence. Research issues addressed in this project include problem solving environments and methodologies for laboratory and instrument-based scientific disciplines, and computer science issues in heterogeneous distributed systems. New approaches are being investigated and developed for fast multiway communication, robust geographically distributed data management methodologies, high-performance computational transforms inlined within collaboration sessions, and related auxiliary issues such as active documents, security, archival storage, and experiment management and control. In this paper, we discuss the design philosophy and systems rationale behind CCF, describe the major subsystems of the collaborative computing environment, and discuss the salient features of the system.

1 Introduction

Systems enabling and supporting collaborative endeavors via the fusion of computers and telecommunications are rapidly gaining importance. The formal beginnings of such systems and their study began in the mid-1980's with the emergence of "Computer Supported Cooperative Work" or CSCW [4]; since then, significant advances have been made in understanding the ways in which remote collaboration is carried out and issues relating to dynamics of cooperative work, as well as in pragmatic and technical issues such as distributed concurrency control mechanisms and building efficient shared computer applications. Recently, attention has been focused on one manifestation of CSCW systems, namely *Collaboratories* [6], which may be informally defined as the emulation of a scientific laboratory in which cooperative scientific or technical work may be carried out without regard to geographical location. While many of the features desired in collaboration systems are similar in concept, scientific collaboratories have a number of additional requirements that are often significantly advanced or sophisticated. They include facilities which permit access to scientific instrumentation, electronic laboratory notebooks, computational harnesses, access to large data sets and the ability to efficiently access them, and support for near-realtime distributed scientific visualization. Several research projects aimed at addressing these issues and building working collaboratories have been initiated in the past few years; in this paper, we describe the Collaborative Computing Frameworks (CCF) environment and system that has been developed at Emory University.

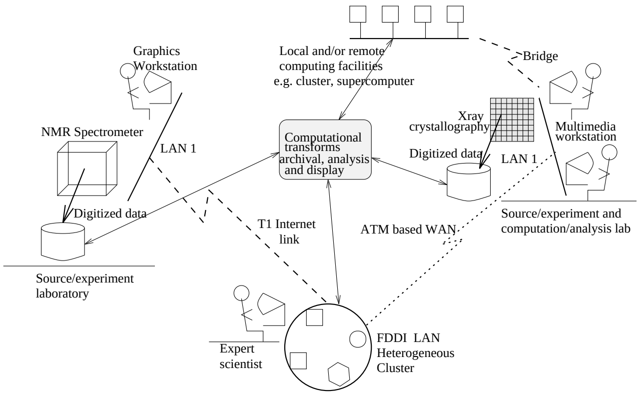

Figure 1: CCF Operational Overview

The overall goal of CCF is to enable the construction of efficient and flexible collaboratories. CCF is intended to be comprehensive and complete; it supports the essential elements of any collaborative activity, viz. application sharing, compute sharing, data sharing, and commmunications. Although we expect that CCF will

*Research supported by NSF grant ASC-9527186

be useful in a number of scenarios, a guiding principle in its design was to identify and meet the needs of natural science investigators with a diverse set of computing, instrument interfacing, and collaboration requirements. Thus, the primary target domain for CCF is collaborative research in chemistry, physics, and biochemistry, and closely related areas. In these sciences and others, it is an increasing trend for many research activities (a) to be jointly performed by collaborators at different sites; and (b) to be highly dependent upon computers – for computational transformations, data management, and graphical displays. In a typical collaborative effort, one investigator prepares or isolates the material to be studied (e.g., a protein, a lipid, cells, a synthetic compound, etc.) and the material is sent to another lab. In this second lab, the material is then studied by one or more of a number of types of instruments (usually a spectrometer of some type) which are frequently computer controlled and produce digital output. Often the initial processing of the data is done on the instrument; for more expensive instruments, the data may be first transported to a workstation or server, and archived. The data is then typically processed with standard instrument-specific or experiment-specific programs, often for extended periods of time; different types of transforms and calculations, parametric variations, comparison to archived data sets, and graphical viewing of structures are performed multiple times until all relevant results are obtained. In all phases of such research but especially in the latter stages, collaboration and computation are fundamental to the work process. The CCF system is designed to facilitate and enhance all such aspects of collaborative work; it provides a distributed computing and collaboration framework that integrally supports human audio/visual (AV) communication, efficient and fast computational transforms, and distributed data management facilities. A typical operational scenario of such a collaborative environment is shown in Figure 1.

2 Background

The CCF project views collaboration to be comprised of a number of aspects, including access to a shared workspace or work area, communication among members of a collaborative work session, cooperative computation as part of the work process, and shared access to data and information artifacts. In an abstract and general sense, these elements are likely to be present in one form or another in most collaboration systems or methodologies. However, CCF has certain unique aspects and features that distinguish it from other comparable frameworks:

- CCF adopts the view that collaborative computing involves communication, computation, and data manipulation. The integral provision of interfaces for computing, and a built in shared file and data object space, distinguish CCF from systems that concentrate on application sharing or conferencing facilities alone.
- Collaborative computing requires "many-to-many" or multiway data exchange at varying qualities of service for different aspects of a session. The core of CCF is the special multicast transport layer protocol suite (CCTL) that has been expressly designed to cater to distributed and collaborative applications.
- A completely distributed architecture in CCF greatly enhances scalability and performance and reduces the potential for bottlenecks and failures.
- CCF caters both to "blind" or collaboration-unaware applications (i.e. legacy applications) and also provides interfaces and a programming API for developing collaboration aware applications.
- A number of other features, including multiway mixed audio, annotations of dynamic entities, totally ordered textual chat, directory services, etc, are integrated into CCF, thereby presenting a complete and coherent framework for cooperative work.

Several other software systems and toolkits that enable or support collaboration have been developed or are evolving. For example, TeamWave is a groupware system that combines synchronous groupware technologies such as shared whiteboards, chat rooms, and customizable groupware applets with a persistent work environment. However, TeamWave is a relatively closed system that is based on the GroupKit[5] groupware toolkit, and only subsystems and applications developed specifically with GroupKit can be used with TeamWave. GroupKit is an extension to Tcl/Tk and is a generic toolkit using which collaboration applications as well as support subsystems such as session managers can be built. Apart from the disadvantage of being unable to support GroupKit-unaware applications, this toolkit is built on other Tcl/Tk and TCP-IP whose drawbacks it therefore inherits. However, developing simple applications is extremely easy and quick with Tcl/Tk; when only generic facilities such as text windows, scrolling, and simple graphics are required, GroupKit applications can be written using several tens of lines of code.

While GroupKit is an example of a collaborative system built on existing toolkits, another class of systems is based on the World Wide Web. An example is Tango[3], which is a prototype of an open system that provides mechanisms to rapidly integrate applications into a multiuser collaborative environment on the Web. Tango uses a client-server architecture and uses a central server to control and distribute information among the clients. Local daemons on each participating machine intercept application traffic and distribute them via the server and browser plugins; the Web browser serves as the shared work surface. A similar approach is adopted by the EMSL Collaboratory project. This suite of tools consists of a session manager that controls the operation of various tools within a Web browser context; tools include audio and video conferencing, synchronized Web "tour"ing, televiewers, electronic notebooks, and electronic instrument control.

A third category of collaborative environments includes X-multiplexing systems such as Xmx and Xtv [7]; these systems support only existing applications (i.e. they do not provide a framework for developing collaboration aware applications) and are based on centralized architectures. The Habanero system from NCSA does permit the creation of new applications but only supports Java objects. Netscape's Conference and Microsoft Netmeeting may also be considered contemporary collaboration systems, but are limited in their functions especially with regard to computation and shared data space support. CCF provides all the major elements of collaboration, is based upon a completely distributed architecture, and supports sharing of existing as well as collaboration-aware applications.

3 CCF Architecture

The CCF system is an extensible collection of core modules, tools, and applications. Core modules are the subsystems that provide infrastructural support for the four major aspects of CCF, viz. communication, data management, computation, and a shared workspace. Tools include auxiliary subsystems that assist in the collaboration process; in the current version of CCF, an audiotool, a chat tool, and a clearboard are included, and a data space manager, video conference tools, and telepointers are under development. Applications are situation- and domain-specific; examples are molecular simulators and viewers, computational fluid dynamics applications, and visualization systems, but could also include generic applications like text terminals, spreadsheets, games, and Web browsers. Applications may either be CCF-unaware, i.e. single-user applications that made shareable by CCF via external means, or they may be CCF-aware, i.e. those explicitly written to be distributed and shared.

A schematic of the CCF software architecture is shown in Figure 2. As indicated, CCF operates over standard Internet protocols or over high speed local networks, but builds its own fast and efficient multiway protocols thereupon – these core modules are identified as CCTL and CCF-API in the diagram. The data man-

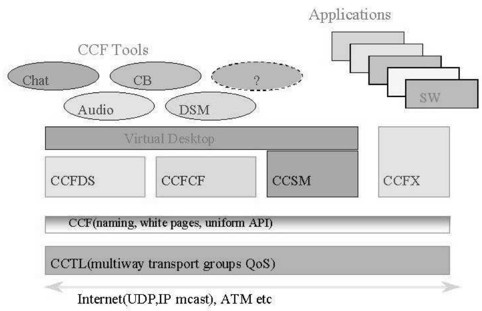

Figure 2: CCF System Architecture

agement and compute management modules provide infrastructural support for these functions, and the CCSM module is responsible for creating and managing the shred workspace. CCFX is an auxiliary subsystem that is used to multiplex CCF-unaware applications. Tools and CCF-aware applications interface to the different core modules as appropriate. Most CCF subsystems have been designed so that they may be used in isolation or in smaller subsets. For example, CCTL provides a standalone multicast protocol library suite and the computation toolkit may be used by itself for heterogeneous distributed computation. Similarly, the audiotool and clearboard may be used independently of the rest of the CCF system.

4 The CCTL Subsystem

Based on the observation that collaborative and distributed computing systems require many-to-may protocols that support different qualities of service, as opposed to traditional pairwise stream protocols such as TCP-IP, we have designed and implemented a full-featured protocol framework for the CCF system. The Collaborative Computing Transport Layer (CCTL) [1] is a communications substrate consisting of a suite of multiparty protocols, providing varying service qualities among process groups. CCTL explicitly supports distributed distributed, collaborative, and multimedia applications. CCTL is based on a two-level group hierarchy. Logical interconnections among entities, called *channels*, define an efficient and light-weight group mechanism. Channels support a variety of service qualities such as reliability and message ordering. Related channels can also be combined to form *sessions*, heavy-weight groups which provide a default atomic multicast service. CCTL supports membership protocols tailored to the quality of service offered by a channel, including a relaxed form of *virtual synchrony*.

At the lower layer, CCTL utilizes IP multicast whenever possible. In the current Internet, as not every site is capable of IP multicast over WANs, CCTL uses a tunneling technique similar to the one adopted in MBONE. At each local subnet containing a group member is a multicast relay, called *mcaster*, that receives a UDP feed from different subnets and multicasts it to its own subnet. A sender first multicasts a message to its own subnet, and then sends the tunneled message to remote mcasters at other subnets where other group members are located. The tunneled UDP messages contain a multicast address to which the message has to be multicast in a subnet. TCP-Reno style flow control schemes and positive acknowledgments are used for data transfer, resulting in high bandwidths as well as low latencies. This scheme has proven to be very effective in the provision of fast multiway communications both on local networks and on wide area networks. On a single LAN, since IP multicast (augmented by CCTL flow control and fast acknowledgment schemes) greatly reduces sender load, throughput at each receiver is maintained near the maximum possible limits (approximately 800kB/s on Ethernet) with the addition of more receivers. For example, with a 20 member group, CCTL can achieve 84% of the throughput of TCP to one destination. If TCP is used in such a situation, the replicated transmissions that are needed at the sender cause receiver throughput to deteriorate as the number of participants increases. A similar effect is observed on WAN's. These results from our preliminary use of CCTL are very encouraging, and we expect this to be beneficial to the other subsystems of CCF that use CCTL. Additionally, we expect to make CCTL available for standalone use by other distributed systems.

5 Virtual Desktop and Tools

The user interface to the CCF framework is a "virtual desktop" realized by a system component called CCSM. All user interaction, program display, application launching, and annotation occurs in this replicated graphical area. CCSM typically executes as a window within a user's X-based workstation, but behaves as a replicated X-server across all CCF session participants for the purpose of collaboration. It serves as the console and control panel for CCF; a user starts up CCSM, and by using menus and text forms, joins or creates a collaborative session. Session information is maintained by an organization-administered white pages server, and session membership is subject to authentication and access control by the session creator. Once enrolled, all participants interact within the collaborative session via facilities provided in the CCSM desktop. Details regarding the facilities supported by this CCF virtual desktop subsystem, as well as design and implementation schemes are described in [8]; here we briefly outline a few salient features.

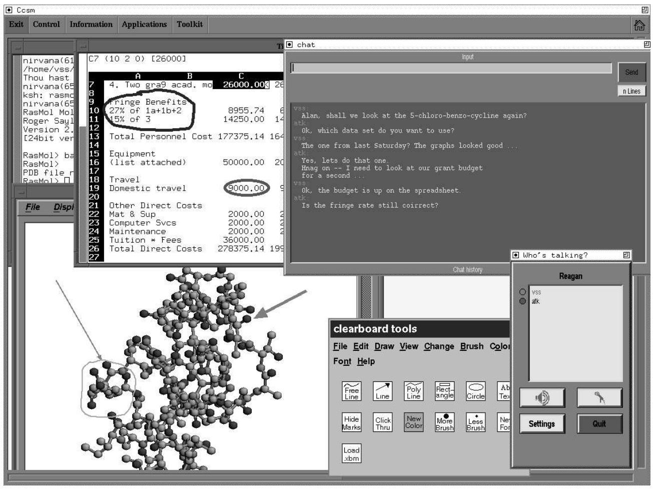

Figure 3: CCSM Desktop Snapshot

Figure 3 shows a snapshot of a CCSM desktop from an active session. The title bar of the CCSM window contains several pulldown menus, including those for session control (joining, leaving, authentication) and application launching. CCSM supports WYSIWYS (what you see is what I see), thereby ensuring that all participants have the same desktop (although participants may choose to, or have to, move or resize their individual "viewport"). Each application is controlled by one participant at a time, and individual title bars provide indications and means for requesting the control token. Applications are shared by (a) X-multiplexing, i.e. connecting to a pseudo-X-server CCFX which then relays requests and responses to all CCSM's; (b) Input multiplexing or collaboration via application replication; or (c) CCF-aware, distributed applications that are written using the CCF-API and shared model. In addition, several native tools are provided in CCF. One of these is the Clearboard (CCFCB) that permits participants to freely draw annotations on a specified area of the desktop. CCFCB is more powerful than whiteboard systems because overlay annotations of any window are possible. This facility is analogous to having multiple transparent marking surfaces placed over the desktop. Further, annotations are object oriented i.e. they may be moved, hidden, deleted, or their properties (color, size, orientation) modified.

CCF facilitates participant interaction via audio and text communication. The CCF Audiotool [10] is unique in many respects, including support for multiple users with realtime software mixing, hands-free operation, silence detection for minimizing network traffic, full-duplex operation with configurable anti-clipping, proxy mixing for slow links, and multiple selectable compression algorithms. The CCFChat tool, a multiway text-based subsystem, provides user-tagged messages that are fully ordered with respect to all participants, single- or multi-line input, scrolling, and an archival/logging capability. A video-based tool is under development. Together, these support subsystems enable effective collaboration and cooperative work which may be conducted within the unified CCSM desktop setting.

6 Data Management

In addition to sharing applications, interacting, and communicating, the CCF system considers comprehensive facilities for data management as an integral aspect of collaborative environments. Data management in CCF is handled by the CCDS (data space) subsystem [2] whose function is to session participants with uniform access to shared data objects, and controlled concurrent access to these objects. The CCDS is comprised of "virtual objects" of various types that may be accessed by a group of participants (either processes or users) in a collaborative session. Objects in CCDS are classified by type. CCDS supports *file, directory, pipe, active,* and *device* object types. A file holds a sequence of bytes of definite length. A directory object is a file containing a list of objects that is updated atomically. A pipe provides access to a bidirectional stream of bytes, of indefinite length; a pipe might provide access to a remote program, and could also involve multiple readers and writers. Writes to a pipe are non-blocking and writes from multiple writers are updated atomically. An active object is an executable in some portable format, such as Java bytecode. A device object provides an interface to a physical device; a device might be a scientific instrument such as a microscope where one participant writes to the microscope (ie. controlling its operation), and with multiple participants reading and viewing the results. A device object may be implemented via an active object that serves as an intermediary between the actual device and the CCDS subsystem.

A virtual object may be created in the CCDS in two ways: (1) a participant may import a data object from his/her own file system into the CCDS, or (2) a program may create a native virtual object using the CCDS programming interface. An imported object is typically an object that is created in an underlying file system and then imported into the CCDS to be shared with other participants in a collaborative computation. For example, consider a data file created by one collaborator which is to be viewed by the entire group. This file is imported into the CCDS and then may be simultaneously viewed by the group with the appropriate application. A data object that has been created with the import function may also be "unimported" back to the originating file system, thus removing it from the CCDS, or it may be "exported" while retaining a local copy. Every virtual object that is created in the CCDS has a set of attributes including permission and ownership indicators, object-specific attributes, and persistence information. CCDS functions are implemented by set of servers that execute on each machine involved in a CCF collaborative session. Application programs, such as file-managers, collaborative editors, and directory services access the CCDS via a library-based API that contains facilities for generic as well as object-specific operations. Where applicable, concurrency control at various levels (e.g. serialized access or explicit locking of file sections) is provided by the CCDS system, thereby enabling effective data object sharing and access in a CCF collaborative session.

7 Computation in CCF

During a collaborative session, especially one involving scientific or technical computation, inline facilities for executing programs are particularly useful. CCF facilitates computation within a collaborative session in a number of ways. First, application-specific computations may be launched by any participant in a session at any machine that is part of a session (subject to permission from the owner of the software, hardware, and the session). This facility, coupled with the ability for such programs to access data from the shared CCDS data space and display output at all participant locations simultaneously, has proven to be very useful in a number of situations. Second, CCF provides interfaces to standard computational tools such as MATLAB and Mathematica; in addition to providing replicated windows to these packages, CCF permits input data and output files to be shared via the data space subsystem.

CCF also supports a complete metacomputing subsystem that may be used standalone, or within collaborative sessions for high-performance parallel distributed computing. This framework, termed IceT [9], is based on the model of concurrent processes distributed over a network that cooperate via message exchange to solve problems using the distributed memory multiprocessing paradigm. The IceT computing environment is a dynamically changing virtual machine which is constructed by assembling multiple hosts or by merging of, or recursive aggregation of existing virtual machines. As in any metacomputing system, IceT applications are composed of multiple distributed processes cooperating to solve a given problem. In IceT, processes may be instantiations of executable binaries (e.g. compiled programs), scripts (e.g. Perl, `sh`), or intermediate code (e.g. Java bytecode, "`.o`" files). In the latter two cases (and sometimes in the case of executables) the IceT process model enables the "uploading" of code or code segments, as necessary, dynamically, and at minimal cost, thereby ensuring currency, availability, and compatibility, and avoiding complex or insecure disk installations. Uploading also permits processes to migrate or move to the data thereby optimizing communication traffic. Further, when done recursively, uploading code and data can be an effective method of creating complex load balanced process interaction structures, a traditionally difficult task in metacomputing. Finally, the uploading mechanism can form the basis for implementing *agents* or *compubots* that are especially useful in computations that are distributed in space and in time. By design, IceT is a multi-user environment which allows for the dynamic transfer of data *and* processes to a dynamic, extended, and adaptable pool of computational resources.

8 Summary

The CCF system described herein is a comprehensive suite of tools and systems for distributed collaboration and cooperation. Rather than focus upon isolated aspects of cooperative work the approach adopted by CCF has been to provide a coherent and integrated framework in which computation, data management, communication, and a common workspace complement each other. The design of the CCF system has also been deliberately modular, so that most components may be used independently or selectively, as the needs of a particular collaborative session dictates. CCF is also unique in that it may be used to share existing collaboration-unaware applications, while providing efficient interfaces for the development of collaboration-aware tools and applications.

The CCF project has been in progress for a little over a year, and alpha testing of the software is in progress. A beta release is imminent, and we expect that substantial experiences with its use will have been gained within the next few months. Simultaneously with release, testing, and refinement of the CCF system as presented in this paper, we also intend to continue enhancements to the system. Among the immediately planned projects are (1) facilities for secure communication and session participation to complement the current owner-permissions based authentication schemes; (2) provisions for multiple modes of token-control for applications; (3) design of a high-level toolkit for rapid development of collaboration-aware applications; and (4) performance improvements and optimizations, particularly in situations involving network multicast of large graphics images.

References

[1] I. Rhee, S. Y. Cheung, P. Hutto, V. Sunderam "Group Communication Support for Distributed Multimedia and CSCW Systems" *Proc. 17th ICDCS*, Baltimore, MD, May 1997.

[2] S.P. Olesen, S.E. Chodrow, M. Grigni, and V.S. Sunderam, "Distributed Data Management Support for Collaborative Computing", *Proc HPCN '97*, Vienna, Austria, April 1997.

[3] L. Beca, et. al., "TANGO - a Collaborative Environment for the World-Wide Web", `http://trurl.npac.syr.edu/tango/`, 1996.

[4] J. Grudin, "CSCW: History and Focus", *IEEE Computer*, Vol. 27, No. 5, pp. 19-27, May 1994.

[5] M. Roseman, S. Greenberg, "Building Real Time Groupware with GroupKit, A Groupware Toolkit", *ACM Transactions on CHI*, Vol. 3, No. 1, pp. 66-106, March 1996.

[6] R. Kouzes, J. Meyer, W. Wulf, "Collaboratories: Doing Science on the Internet", *IEEE Computer*, Vol. 29, No. 8, pp. 40-46, August 1996.

[7] O. Jones, "Multidisplay software in X", *The X Resource*, Vol. 6, pp. 97-113, 1993.

[8] A. Krantz, et. al. "Ccfx-Ccsm: A Distributed X-Multiplexer", submitted to *UIST 1997*, April 1997.

[9] P. Gray, V. Sunderam, "Metacomputing with IceT", `http://www.mathcs.emory.edu/g̃ray/IceT3.ps`, March 1997.

[10] S. E. Chodrow, M. D. Hirsch, I. Rhee, S. Y. Cheung, "Design and Implementation of a Multicast Audio Conferencing Tool for a Collaborative Computing Framework", *Proc. 3rd Joint Conference of Information Sciences*, Durham, NC, March 1997.