SOFTWARE ARCHITECTURE FOR THE UARC WEB-BASED COLLABORATORY

The Upper Atmospheric Research Collaboratory supports global collaboration among physicists studying space weather, offering access to more than 30 ground- and space-based data sources and supercomputer-class theoretical models.

SUSHILA SUBRAMANIAN, G. ROBERT MALAN, HYONG SOP SHIM, JANG HO LEE, PETER KNOOP, TERRY E. WEYMOUTH, FARNAM JAHANIAN, AND ATUL PRAKASH
University of Michigan

The emergence of the Internet has enabled interactions between people working in the same field across the globe. This has resulted in a number of efforts to solve the problems of supporting group communication, and providing access to information sources that are geographically and administratively scattered. Collaboratories are one way to provide virtual shared workspaces to users. We define a collaboratory as a wide area distributed system supporting a rich set of services and tools that facilitate effective synchronous and asynchronous communication between two or more people who are not co-located.

The Upper Atmospheric Research Collaboratory is an Internet-based system that gives space scientists a virtual shared workspace for conducting real-time experiments as well as various asynchronous collaborations from geographically dispersed facilities.[1] During the past two years, UARC has integrated a significant fraction of the worldwide observational systems devoted to space physics and aeronomy. (Figure 1 shows the sources available to the last data-collection campaign in April 1998.) Moreover, the system now supports real-time outputs from supercomputer-class theoretical models. This has enabled simultaneous comparison of experimental and theoretical data on a global scale. Such comparisons would not be possible without collaboratory technology.

UARC has evolved into a scalable Web-based collaboratory over a period of six years. Since its inception in 1993, UARC has supported more than 10 scientific campaigns, each of which represented a concerted effort to collect real-time data simultaneously from instruments around the world for several days at a time. The first few campaigns ran on a distributed set of Next systems on the NSFNET and could support about a dozen active participants. The current Web-oriented Java-based system supports a more accessible interface by moving away from specific hardware or operating system requirements. This allows UARC to provide sharable multimedia tools across different platforms, support more users, increase the number of sites and instruments feeding data to the collaboratory, and provide access to output from supercomputer-class theoretical models of the thermosphere and ionsphere. The Web-based system continues to evolve, and during the most recent data collection campaign, April 1998, scientists were able to examine data from a suite of 40+ instruments.

In this article, we describe the software architecture and application environment of the UARC system as it emerged from a six-year development effort to support this scientific community in its work. (For a complete list of contributors to the current system, see the sidebar, "The UARC Team.")

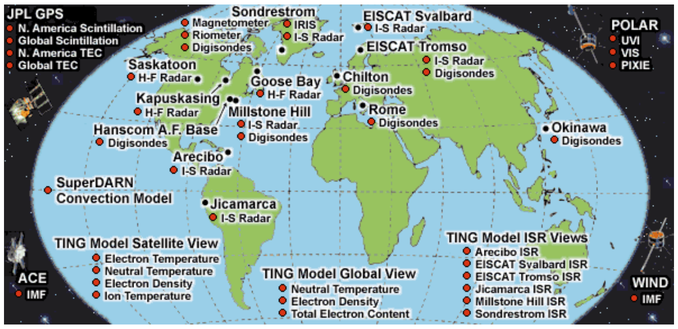


Figure 1. Data sources available to UARC during the last set of campaigns between April 1997 and April 1998. Sources include instruments such as incoherent scatter radars and magnetometers, satellite images, and predictive Thermosphere Ionosphere Nested Grid (Ting) model output.

UARC APPLICATION REQUIREMENTS

The base unit for space science collaboration is data, so providing access to remote data sources is fundamental, along with recording the data for post facto analysis and supporting its annotation. In general, UARC had to establish a coherent system for effective distributed administration of numerous data suppliers and servers.

This coherency included integration of domain-specific visualizers for the remote data resources. Managing the large amount of diverse data available to UARC participants imposed special requirements on the system's data visualization component, such as support for compact representation of multiple graphs and time-synchronized views of multiple data sources. In addition, the design had to be extensible so that new data sources and graphical representations could be added with limited administrator overhead.

The UARC system must provide tools for seamless transitions between synchronous and asynchronous collaboration. In addition to visualizers, scientists must be able to add standard items to the collaboratory workspace including URLs, custom

THE UARC TEAM

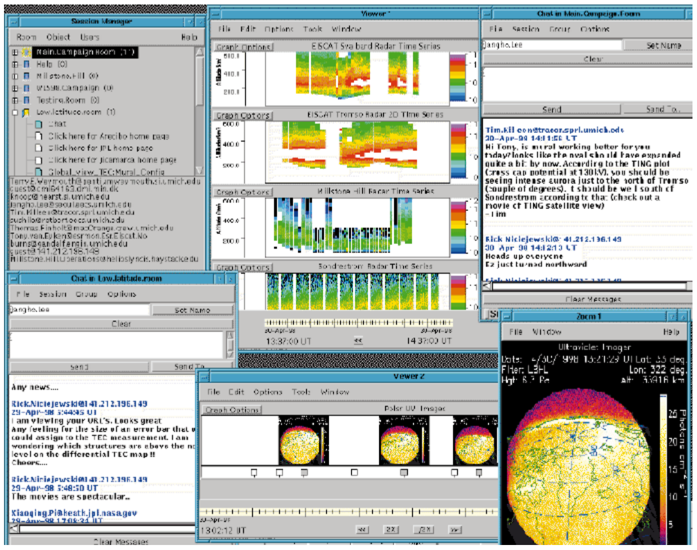
The Upper Atmospheric Research Collaboratory is a multidisciplinary research collaboration supported by the National Science Foundation. UARC studies space phenomena, such as magnetic storms that originate on the sun and can interfere with radio and television reception, disrupt electrical-power transmission, and threaten orbiting spacecraft and astronauts.

The UARC team is made up of faculty, researchers, and graduate students from the fields of computer science, social and behavioral sciences, and space physics. The computer scientists work with space physicists to develop the software systems; the space physicists try out prototypes in live collaborative research settings under the eyes of the behavioral scientists; and the behavioral scientists evaluate successes and failures and give guidance to the computer scientists for the next round of prototypes.

This iterative feedback loop drove the software design process and resulted in a testbed that is also a successful working collaboratory. The principal investigators at the University of Michigan who were actively involved in the design, deployment, and evaluation of UARC include Daniel Atkins, Robert Clauer, Tom Finholt, Farnam Jahanian, Tim Killeen, Gary Olson, Atul Prakash, Joseph Hardin, and Terry Weymouth.

Partnerships with several investigators at other institutions contributed greatly in the creation and evaluation of the testbed. These institutions include SRI International, Danish Meteorological Institute, Millstone Hill Observatory, Arecibo Observatory, EISCAT, NASA JPL, and NASA Goddard Space Flight Center.

The home page for the UARC project is at http://www.si.umich.edu/UARC/. A complete list of partner institutions is at http://www.si.umich.edu/UARC/partners.htm.


Session Manager
Viewer
Chat in Main.Campaign.room
Chat in Low.latitude.room
Viewer 2
Zoom 1
EISCAT Svalbard Radar Time Series
EISCAT Tromso Radar 2D Time Series
Millstone Hill Radar Time Series
Sondrestrom Radar Time Series
Polar UV Imager
Ultraviolet Imager
Send
Send To
Clear
Clear Messages
Set Name

Tim.Killeen@tracer.sprl.umich.edu
30-Apr-98 14:11:56 UT
Hi Tony, is mural working better for you today? looks like the oval should have expanded quite a bit by now. According to the TING plot (cross cap potential at 130kV), you should be seeing intense aurora just to the north of Tromso (couple of degrees). It should be well south of Sondrestrom according to that (check out a movie of TING satellite view)
-Tim

Rick.Niciejewski@141.212.196.149
30-Apr-98 14:12:10 UT
Heads up everyone
Bz just turned northward

Any news....

Rick.Niciejewski@141.212.196.149
29-Apr-98 5:44:45 UT
I am viewing your URL's. Looks great
Any feeling for the size of an error bar that we could assign to the TEC measurement. I am wondering which structures are above the noise level on the differential TEC map !!
Cheers....

Rick.Niciejewski@141.212.196.149
29-Apr-98 5:48:50 UT
The movies are spectacular..

Xiaoqing.Pi@heath.jpl.nasa.gov

Figure 2. Snapshot of a UARC Campaign in April 1998. At the top left corner is the Room Manager showing the active participants in the highlighted room. The bottom left and top right corners show the Chat application from two different rooms (The Low Latitudes Room and Main Campaign room respectively). The top central data visualizer (Mural) displays time synchronized information from 4 radars: EISCAT Tromso and Svalbard in Norway, Millstone Hill near Boston, USA and Sondrestrom in Greenland. The bottom data visualizer shows data from the UVI instrument on the Polar satellite, while the bottom right display is a zoomed-in view of one of the images from the satellite.

GIFs, and text annotations. This workspace must be designed to support both public and private collaboration. Moreover, to allow dynamic groups of scientists to use sharable tools effectively, the tools must support group awareness in terms of knowing who the active participants are and their locations.

UARC supports two modes of operation:

- *Campaign mode*, as noted above, allows scientists to monitor and respond to unfolding phenomena in real time; Figure 2 shows a screen snapshot from the April 1998 campaign. In this mode, a scientist is likely to be interested in observing phenomena and communicating with other scientists about it. For example, during the April 1997 campaign, radar operators responded quickly to the effects of a solar flare and arranged to keep instruments around the world going beyond the usual scheduled times in order to capture a geomagnetic storm of unusual intensity.
- *Electronic workshop mode* supports scheduled cybergatherings with specific goals in mind, such as planning future research or writing papers. This mode requires access to archived data or to data preprocessed by participating scientists as well as to shared access to multimedia tools.

UARC servers had to support continuous campaign-style requests for weeks at a time, and the system included tools for seamless transition between synchronous and asynchronous collaboration modes.

Variation in client bandwidth resources in the heterogeneous Internet environment implicitly imposed additional requirements on the UARC middleware. Users had to be able to set quality of service (QoS) parameters on a per-graph basis, and servers had to support application-level QoS and multicast delivery based on specific data semantics.

UARC SOFTWARE ARCHITECTURE

Figure 3 shows the five main components of the UARC architecture:

- The Salamander server provides data dissemination and archival services, distributing data from remote instruments or transformation modules to client-end data visualizers or archive servers.[2] Salamander also handles variation in bandwidth and network speeds and enables users to manage input data quality through application-layer QoS controls.
- The Corona server supports synchronous sharing of state and events, archive and replay of actions, and group management in terms of joins, leaves, access control, and awareness.[3] Corona is supplemented within UARC by related DistView services, which include a registry service, and Session or Room Managers.
- Cache and Computation Modules are domain specific, transforming raw data from remote instruments into user-specific formats. CCM output is published through Salamander and subscribed to by client data visualizers.
- Client tools allow users to access the collaboratory and view the data in several available formats. Also included in this component are groupware applications based on the DistView Toolkit,[4] such as a multiparty chat or shared whiteboard.
- Data suppliers publish to Salamander from diverse data sources, including remote instruments, satellites, Web pages, or predictive models running on supercomputers.

Figure 3 shows the interconnections among these components, each of which is independently administered and maintained. This approach supports the design goals of scalability, maintainability, and reasonable performance, limiting the propagation of side effects or errors between subsystems.

The remainder of this discussion addresses features of the application environment.

Collaboration Facilities

Users enter UARC through a Session Manager application that uses the Corona server for communication support and the DistView Toolkit for group management and event sharing.

To join a UARC session, a user invokes a Session and Room Manager, which is constructed as a coordinated collection of group-aware Java applets.[5] To support dynamic reconfiguration of shared workspaces and allow access over the Internet, the Room Manager uses *rooms* as the high-level grouping mechanism for objects, such as applets, users, and arbitrary data objects. Rooms can be used for asynchronous and synchronous collaboration because their state persists across synchronous sessions. Room participants can perform different roles (such as administrator, member, and observer), with appropriate access rights. Figure 4 gives a snapshot of room activities.

The Session Manager provides a common interface to the various applets within rooms. It supports queries on the status of the collaboratory or group membership, and can send commands to applets. When a user enters a room, the Session Manager sends a message to the Room Manager, which checks the user's access control rights and privileges. If the user is allowed entry, the Room Manager updates its data structures and notifies the Session Manager. The user may now instantiate objects within this room.

In Session Manager, a room represents a group activity and has a number of objects, which can include shared tools, such as a chat, data visualization tools, or URLs posted by group members. The Session Manager works with the Room Manager, which maintains a database of user whereabouts in the system, for example, who is in room A and running object B. Session and Room Managers are both Corona clients and use Corona's communication and membership notification services to communicate. The Session Manager also communicates with a user registration and authentication server.

The Session and Room Managers feature

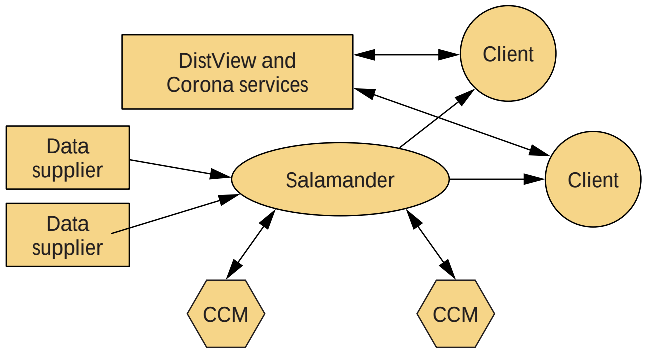
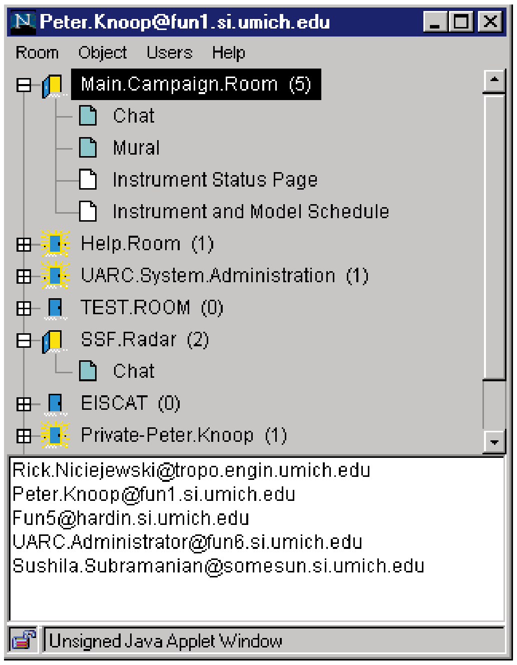


Figure 3. Architecture of the UARC system, showing the interconnections between the data dissemination service (Salamander server), shared state and event management service (Corona server and DistView), Cache and Computation Modules, data suppliers, and client tools.



Main.Campaign.Room (5)
Chat
Mural
Instrument Status Page
Instrument and Model Schedule
Help.Room (1)
UARC.System.Administration (1)
TEST.ROOM (0)
SSF.Radar (2)
Chat
EISCAT (0)
Private-Peter.Knoop (1)

Rick.Niciejewski@tropo.engin.umich.edu
Peter.Knoop@fun1.si.umich.edu
Fun5@hardin.si.umich.edu
UARC.Administrator@fun6.si.umich.edu
Sushila.Subramanian@somesun.si.umich.edu

Figure 4. The Session Manager. The upper window shows a list of seven visible rooms, the first six of which are public and the last, private. The Session Manager automatically creates the list when a user logs into the system. The lower window shows current users in the selected room. An open doorway indicates the room in use, while lit doors indicate rooms in which the user is participating. The number in parentheses following the room name indicates the number of participants within that room.

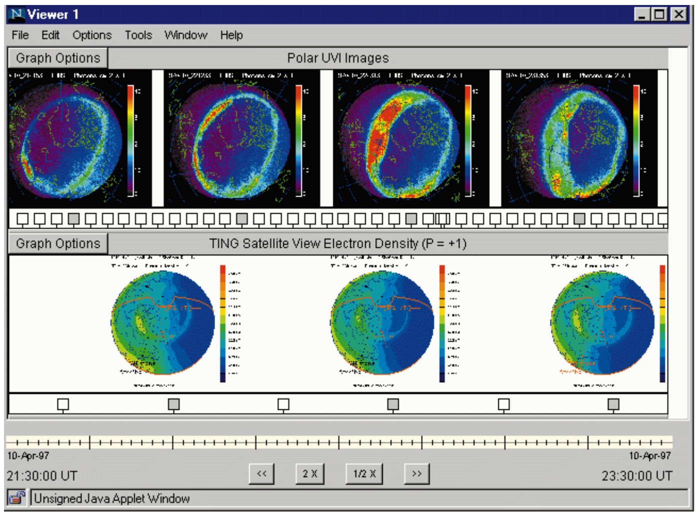

Viewer 1
File Edit Options Tools Window Help
Graph Options
Polar UVI Images
Graph Options
TING Satellite View Electron Density (P = +1)
10-Apr-97
21:30:00 UT
10-Apr-97
23:30:00 UT
<<
2 X
1/2 X
>>
Unsigned Java Applet Window

Figure 5. Stacked graphs with a common time axis. This snapshot of Mural shows real-time satellite images from the UVI instrument on the Polar satellite being compared against images generated by a predictive model (shown in the lower graph) running on parallel computers. The time axis at the bottom shows that the current visible range is for a period of two hours running from 21:30 UT to 23:30 UT on 10 April 1997. The buttons at the bottom allow the user to scroll along the time axis, and reduce or increase the visible time range by clicking the 1/2x and 2x buttons, respectively.

- *User extensibility*. Users can add their own collaborative applets that conform to the published Session Manager interface.
- *Group awareness*. Using the Session Manager API, applets can query about groups, active users, and so on.
- *Link applets*. Users can move among rooms in a hypertext model.
- *Common utility applets*. Tools include shared whiteboards, e-mail support, data visualizers, and multiparty chat.
- *Asynchronous collaborations*. The Session and Room Managers support persistent rooms and applets that retain state across invocations.

Group communication server. Corona is a group communication service provider to UARC collaboration tools such as Session Manager, Chat, and DVDraw.[3] Corona services may be categorized as stateful in that Corona supports communication groups for which shared application state can be defined, and manages shared state.

With its message-based management and independence from application semantics, Corona models shared state as a set of arbitrary objects, each having a client-generated identifier. State update messages are cached on disk to support a robust state transfer service, in which new members can receive the current state of their groups despite network failures and client crashes. Corona also supports persistent groups and locks to allow exclusive access to shared objects. A persistent group outlives its membership, and thus supports long group sessions. Other Corona services include notification of group membership and client and group property changes.

Groupware applications toolkit. DistView provides the interface and libraries to run a stand-alone application in shared mode.[4,6] It supports selective window sharing in which application windows can be individually exported and imported to provide a synchronized view of shared application state while allowing private work in nonshared windows.

To provide quick response times, DistView employs an object-replication scheme, in which objects associated with shared windows are replicated at import sites. DistView also provides a library of reusable, group-aware objects for designing shared windows. DistView-based applications subscribe to Corona services for their communication needs. The Toolkit includes an application component, DVGroup, which provides the Corona client interface.

Data Visualization

Mural is a client tool that supports shared, interactive visualization of data from remote instruments distributed via the Salamander server. A generic tool, Mural presents UARC-specific user options, using information provided in a set of configuration files. Graphs are stacked within a scrollable frame to preserve screen real estate, as well as to display data in a time-synchronized fashion (with the common x-axis being time). Such a stacked model allows users to scroll through multiple graphs simultaneously, as well as to compare data from multiple instruments or between instrument data and predictive model data. Mural has a published interface with predefined formats for configuration files and methods to add new locations, instruments, and graphs. Configurable parameters include location, instrument type, graph type, transport layer (on a per-graph basis), and axes for graphs.

Figure 5 shows two sets of time-synchronized stacked graphs. The highlighted icons below the GIFs show the currently displayed images. Notice that the periodicity for the two graph sources is considerably different (more icons for the first graph than for the second).

Mural can run as an independent or shared application, interacting with other UARC applications via the DistView Toolkit. In shared mode, actions taken by any importer or exporter (such as additions, deletes, and resizes) will be visible to all other users. Mural features include the ability to set data quality on a per-graph basis, zoom on images, view a series of images as a movie, save and restore a configuration of graphs as a single operation, clone graphs and viewers, and cut and paste images to the shared whiteboard for annotating snapshots.

The UARC suite of tools also includes a multiparty, text-based chat tool. Chat creates a stateful, persistent group at the Corona server. A chat process must join that group to receive broadcast messages. Latecomers can download previously broadcast messages to catch up with the conversation. Chat also subscribes to Corona's notification services to provide an up-to-date participant list, broadcast messages to other chat groups, and support private messages.

The shared whiteboard, DVDraw, is used to import images, make annotations, and create shapes and lines. DVDraw also creates a stateful group at the Corona server and defines the canvas contents to be the shared state of the group. To maintain a synchronized state, DVDraw uses Corona's locks to control concurrent accesses to the canvas.

Scalable Data Distribution

A Salamander-based system is composed of two basic units: servers that act as distribution points generally co-located with Web servers, and clients that act as both data publishers and subscribers. These units can be connected in arbitrary topologies to best support a given application.

Salamander supports groupware applications by providing virtual distribution channels in an attribute-based data space. In a Salamander system, a tree of distribution nodes (servers) can be dynamically constructed to provide points of service into the data space. Clients provide persistent queries to the Salamander substrate, using attribute expressions to request data flows, thereby subscribing to a virtual data channel. Salamander connections are first-class objects (that is, objects that are directly addressable) and allow feedback from subscribers to data publishers. Plug-in modules can be added at any point in the distribution tree, allowing application code to affect data distribution, and thereby providing the mechanism to support application-level QoS policies.

Attribute-based data distribution. The Salamander substrate uses flexible attribute sets as its addressing mechanism. Opaque data objects are delivered to receivers based on outstanding subscription queries. Applications subscribe to virtual channels based on a stream of data objects' invariant set of attribute values. These attribute sets, or queries, are propagated throughout the tree of Salamander servers. The use of text-based attribute key-value tuples for data routing is a flexible and powerful abstraction. In general, it supports a variety of application-specific data flows, which can be addressed by an arbitrary hierarchy of channels that can be defined and dynamically extended by the application.

Salamander supports groupware applications by providing virtual distribution channels in an attribute-based data space.

The Salamander substrate provides an abstraction for the distribution of data from publishers to subscribers through its channel subscription interface with both anonymous and negotiated push techniques. In anonymous push, publishers package opaque data objects with text-based attribute lists. Subscribers place persistent queries to the Salamander substrate, using lists of attribute expressions that match both current and future objects published to the Salamander space. Alternatively, this procedure can be thought of as accessing a distributed database where the queries are persistent. The query aspect of Salamander's attribute-based subscription service differs from traditional database systems, in that Salamander queries are dynamic entities that act on both the current state of the system and future updates.

Application-level quality of service. The Salamander architecture provides application-level QoS policies to deliver data to clients, tailoring content to fit client connectivity and processing resources. Application-specific policies allocate the available bandwidth between a client's subscribed flows, providing a client with an effective throughput based on semantic thresholds that only the application and user can specify. These application-level QoS policies are achieved with plug-in policy modules at points in the distribution tree. The UARC application uses discrete delivery, data degradation, and data conversion plug-in modules. By multiplexing the subscribed flows, discrete delivery modules can be used to prioritize, interleave, and discard discrete data objects. The Salamander substrate also provides for on-demand data degradation and conversion of data objects. In order to support real-time collaboration between heterogeneous clients, some mechanism for graceful data degradation must provide useful data to slower participants.

Flexible attribute routing. Salamander's flexible attribute routing allows applications to extend the system with specialized services that act as publishers and respond to specific queries or commands.

Salamander provides data persistence by incorporating a custom lightweight temporal database.

These services register with Salamander, thus enabling clients to query the substrate about a particular supply or service. Two general services have been constructed for use with UARC: a lightweight temporal database and the CCM.

Salamander provides data persistence by incorporating a custom lightweight temporal database, which stores a virtual channel's data as a sequence of write-once updates primarily based on time, and satisfies data requests based on temporal ranges within the update stream. A temporal database generally views a single data record as an ordered sequence of temporally bound updates, which correspond to virtual channels in the Salamander database. Salamander's synergy between real-time data dissemination and traditional temporal and relational databases is one of its main contributions. Our model provides support for persistent queries that act over both present database elements and any real-time updates to the database elements.

Cache and Computation Modules

CCMs are special entities that subscribe to raw data from remote instruments or Web sites, apply a data transform to generate a graph, convert it to a GIF, and then republish it. The Salamander database stores raw data and their corresponding images to support historical queries or time-based scrolling. The GIF images display at the client through Mural for active participants or through a Web page for casual observers. By converting scientific data into simple images, the CCMs offload computationally intensive operations from the Java-based clients, trading network bandwidth for client computational resources.

A CCM registers itself with its Salamander server at startup. Clients can then use the registration information to manipulate a CCM's behavior remotely. A client can start or stop the data supply, change QoS requirements, send domain-specific choices of transformation algorithms, invoke time or other boundaries on requested data, and control data arrival rate.

IMPLEMENTATION FOR THE FINAL CAMPAIGN

The environment for the final April 1998 campaign varied in terms of network connectivity and client processing power. Participants in Greenland and Puerto Rico connected across a slow 56K line, periodically publishing or viewing data, while other data sources or clients were on T3 or, on occasion, vBNS links. Machines also ranged from a 486-based PC to high-end Pentiums and workstations.

The data itself ranged from a few hundred bytes for periodic raw data to 400-Kbyte GIF images published by the Pixie instrument on NASA's Polar satellite. Images generated by the CCMs were generally about 80 Kbytes and were published every four minutes or so.

The raw data was published from once every five seconds from the Sondrestrom Incoherent Scatter Radar to about once in 20 minutes from some Digisondes. Average round-trip times between most servers in Ann Arbor, Michigan, and clients or publishers also varied greatly from a few milliseconds within the University of Michigan campus to about 750 milliseconds to Puerto Rico.

All Mural, CCM, Corona, and DistView code is written in Java. The servers and clients can be connected in arbitrary topologies. All Java subsystems can run on Solaris or Win32 machines. The Salamander server is a Posix thread implementation on Solaris and is written in C. Salamander emulates a multicast service in the absence of ubiquitous support for Mbone. Most of the data suppliers are written in Java and C, or run as Perl scripts at remote instrument sites. Some data suppliers periodically monitor Web pages set up by remote sites

OTHER INTERNET COLLABORATORY EFFORTS

The **Remote Experimental Environment** (REE) allows scientists to conduct fusion energy research using instruments located at General Atomics in San Diego, California.
http://fusionscience.org/collab/REE/

The **Spectro-Microscopy Collaboratory** uses collaboration tools such as Mbone videoconferencing tools and electronic notebook for research in spectro-microscopy.
http://www-itg.lbl.gov/BL7Collab/
http://www.lbl.gov/Notebook/project.html

DOE2000 contains pointers to several collaboratory projects funded by the Department of Energy.
http://www.mcs.anl.gov/DOE2000/

The **Materials MicroCharacterization Collaboratory** provides views and controls of remote instruments as well as a shared electronic notebook to which users can add text and images.
http://tpm.amc.anl.gov/MMC/

The **Collaboratory for Environmental Molecular Sciences** uses a shared electronic notebook, audio- videoconferencing tools, chat, shared whiteboard, and some window sharing capabilities.
http://www.emsl.pnl.gov:2080/docs/collab/

The **Mbone** site lists audio- and videoconferencing tools used in several collaboratory efforts.
http://www.lbl.gov/mbone/

Collaborative Computing Frameworks (CCF) is a suite of software systems, communications protocols, and tools that construct a virtual work environment on multiple computer systems connected over the Internet, to form a collaboratory.
http://emily.mathcs.emory.edu/ccf/

Collaboratory for Microscopic Digital Anatomy (CMDA) provides researchers at a remote site distributed interaction with unique instrumentation for data acquisition, as well as significantly enhanced capabilities to derive and analyze three- dimensional (3D) data by accessing high-performance computers.
http://www-ncmir.ucsd.edu/CMDA/

The **Decisions Systems Group InterMed Collaboratory Project** focuses on the development of health care information systems, and in developing a robust frame for collaboration over the Internet.
http://dsg.harvard.edu/public/intermed/InterMed_Collab.html

The **TeamWave** workplace is a framework that provides support for synchronous and asynchronous collaboration, development of new tools using their libraries, customizing tools and other room objects, and sharing of documents.
http://www.teamwave.com/

and “push” the data from it as it becomes available through the Salamander interface.

The Room Manager and Corona are implemented as Java applications. The Session Manager runs as a Java applet under a Java-enabled browser such as Netscape. Initial communications between the Session and Room Managers assign unique IDs to objects within a room and support shared workspaces via group IDs. These IDs generate specially tailored HTML documents dynamically by CGI scripts, which then provide users with a specialized configuration within which to work. All client end tools, including the data visualizers and groupware applications, are signed with a Netscape Certificate. Signed tools are primarily used to allow Java applets to access special privileges, such as connecting to multiple servers simultaneously.

EXTENDING THE UARC SYSTEM

UARC’s architecture can also support general Internet collaborative applications. For example, the Salamander substrate has been used as a general-purpose data distribution service for several other large Internet projects. Similarly, the DistView Toolkit has been used for collaboration on both LAN and Internet environments. Another Internet performance project uses the Mural package to visualize Internet statistics in real time. As loosely coupled systems, the UARC servers function independently, allowing domain-specific approaches to scalability. Moreover, this approach limits the propagation of any side effects or errors between subsystems. An inoperative server limits, but does not incapacitate, the available system functionality.

Ongoing work includes improving the current system's robustness, as well as extending the system to support a richer set of collaboration tools, enabling more seamless methods of synchronous and asynchronous collaboration, as well as record and replay support. Workspaces will be more hierarchical, allowing multiple Room levels. Data visualization needs to support sophisticated multimedia, such as audio, video and 3D simulation. We will also continue work on networking issues, such as support for disconnected operation, mobile clients, and increased scalability and QoS support. Finally, we are including access to publishing tools and digital libraries from within the UARC environment to support presentation of Campaign results or summaries from an Electronic Workshop. ■

ACKNOWLEDGMENTS

We thank all the UARC team members and participants. Their valuable input in the areas of computer science, space physics, and human computer interface greatly influenced the design of the system and allowed for an iterative process that resulted in a useful and unique collaboratory. This work was supported by funds from the National Science Foundation cooperative agreement IRI-92-16848. We gratefully acknowledge the Intel Corporation for its generous donation of equipment through its Technology for Education 2000 program.

REFERENCES

1. G.M. Olson et al., "The Upper Atmospheric Research Collaboratory (UARC)," *Interactions*, Vol. 3, May-June 1998, pp. 48-55.
2. G.R. Malan, F. Jahanian, and S. Subramanian, "Attribute-Based Data Dissemination for Internet Applications," *J. High Speed Networks* (special issue on Multimedia Networking), Vol. 7, No. 3, 1998, pp. 319-337.
3. R. Hall et al., "Corona: A Communications Service for Scalable, Reliable Group Collaboration Systems," *Proc. CSCW*, ACM Press, New York, 1996, pp. 140-149.
4. A. Prakash and H. Shim, "DistView: Support for Building Efficient Collaborative Applications Using Replicated Objects," *CSCW*, ACM Press, New York, 1994, pp. 153-164.
5. J.-H. Lee et al., "Supporting Multi-User, Multi-Applet Workspaces in CBE," *Proc. Sixth ACM Conf. Computer-Supported Cooperative Work*, ACM Press, New York, 1996, pp. 344-353.
6. H. Shim et al., "Providing Flexible Services for Managing Shared State in Collaborative Systems," *Proc. European Conf. Computer-Supported Cooperative Work*, Kluwer Academic, Lancaster, UK, 1997, pp. 237-252.

Sushila Subramanian received an MSE degree in 1991 from the University of Michigan. She has since worked in the areas of distributed file systems over high-speed networks, Web-based collaboratories such as UARC, and Internet performance measurement.

G. Robert Malan received the BS degree from Carnegie Mellon University in 1990 and the MSE degree in 1996 from the University of Michigan where he continues working toward his doctorate. He has been an active participant in the UARC, IPMA, and Mach projects.

Hyong Sop Shim is a PhD candidate in computer science and engineering at the University of Michigan, Ann Arbor. He now works at Bellcore as a research staff member in the areas of multimedia and groupware systems.

Jang Ho Lee is a PhD candidate in computer science and engineering at the University of Michigan, Ann Arbor. His research interests include computer-supported cooperative work and distributed systems. He is a student member of the IEEE and the ACM.

Peter Knoop is the system administrator and user liaison for UARC. He earned a BSE degree in atmospheric and oceanic sciences in 1989 and an MS degree in marine geology and geochemistry in 1993 at the University of Michigan, where he continues working toward his doctorate.

Farnam Jahanian is an associate professor of electrical engineering and computer science at the University of Michigan. He received a PhD degree in computer science from the University of Texas at Austin in 1989. His current research interests include network protocols and architectures, and distributed computing.

Atul Prakash is an associate professor in the Department of Electrical Engineering and Computer Science at the University of Michigan, Ann Arbor. He received a PhD degree in computer science from the University of California at Berkeley in 1989. His research interests include computer-supported cooperative work, distributed systems, and security.

Terry E. Weymouth is an associate research scientist and has been a software developer and research manager for projects in collaboration technology for the support of medical diagnosis and for the support of distributed remote scientific experimentation. He received a PhD in computer science from the University of Massachusetts in 1986.

Readers can contact the authors at University of Michigan, Ann Arbor, MI 48109; e-mail {sushila, rmalan, hyongsop, jangho, knoop, weymouth, farnam, aprakash}@umich.edu.