Java Synchronous Collaboratory

Vishal Mehra Rajat Mukherjee Daniel Dias
Parallel Commercial Systems
IBM T.J. Watson Research Center
P.O. Box 704
Yorktown Heights, NY 10598
{*vishal,rajatm,dias*}*@watson.ibm.com*

July 2, 1996

Abstract

This document describes *Shaking Hands*, a collaboratory system for the internet that allows multiple collaborators on the internet to participate in various activities, such as chat, shared-browser foil navigation, whiteboard and chess. We discuss the potential applications of this technology and present our model for web-oriented Java-based collaboratory. We describe the system architecture and design and outline instructions on how the system is to be used. The IBM-only URL is: *http://cayuga.watson.ibm.com/shakinghands*.

Introduction

Java Synchronous collaboratory is a paradigm that provides for real-time collaborative applications among clients on the World Wide Web (WWW). We will refer to a set of collaborating clients as a "party". Elements of collaborative applications include provision of a shared white-board for a party; simultaneous/synchronous viewing of the same web page or a set of foils among a party; chat; audio conferencing; games such as chess; joint editing of forms, and so on. Combinations of these elements can be used to create collaborative applications.

Objectives

The objectives of this demo are to:

- Provide a general framework for building synchronous collaboratory applications;
- Put together (useful) synchronous collaboratory applications that illustrate the power of this paradigm;
- Provide Java Servers to enable these functions, to demonstrate their functionality, generality and portability.

Outline of Demo Applications

For this demo we have put together the following applications:

- **Foil Server:** This application provides for foil presentations by allowing synchronous navigation through a set of foils, presenting the same foil to all clients in a party. The application allows the user to specify a starting URL, and sets up a control panel with "next" and "previous" foil buttons. When either of these buttons is hit by the presenter (and currently any member of the party), the same foil appears on the browsers of all clients in the party.
- **White Board::** The white board could be used concurrently with a foil presentation. It allows for pixel, line, and some shapes, that when drawn on the whiteboard of any client in a party, appears on all clients. This could be used to discuss what if's or questions along with the foil presentation.
- **Java Chat:** This is similar to IRC, and can be used in conjunction with the foil presentation.

- **Chess:** This application is included as an example of inter-client multi-party games. In each party, it allows two players, and multiple viewers. A commentator in a party has the capability of drawing on the board seen by viewers, to indicate what-if's.

While we believe that these elements are useful in their own right, and could be used for, say, foil presentations today, they are intended to illustrate the power of the model. The framework embodied in the demonstration, is easily used and extended to add numerous other collaboratory applications.

System Architecture and Design

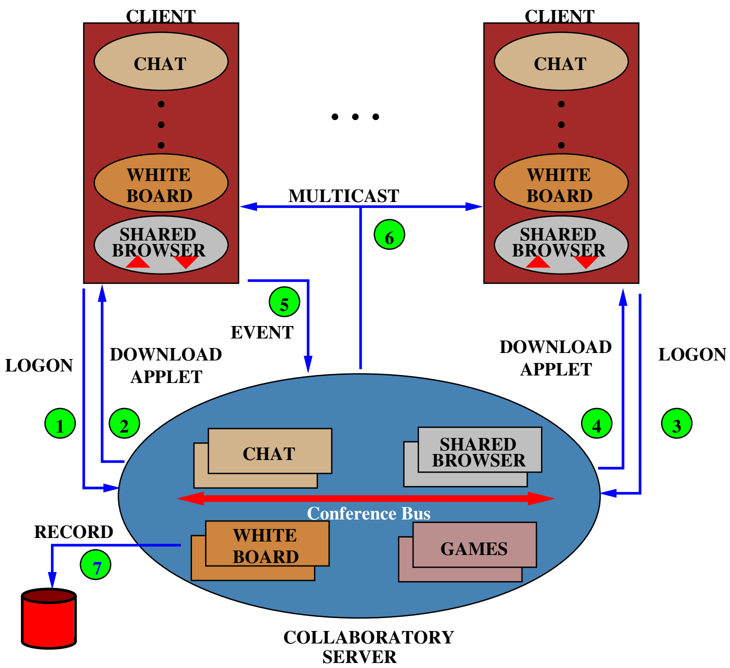

Figure 1: **System Architecture**

The basic model is illustrated in Figure 1. Clients load the URL for the collaboratory system, which downloads an applet from a web server. Java servers for various collaboratory elements run on the same node or cluster as the web server (more on this later, in Section). The applet sets up a connection to the base Java collaboratory server. The user can then choose among collaboratory services and, based on the services selected, the applet makes connections to the required set of collaboratory servers.

Once the connections to the collaboratory servers is set up, the basic paradigm is similar for the various applications. Essentially, user events are marshalled by the applet, and sent to the appropriate Java server; the Java servers propagate them to the clients that are subscribed to that channel and also log the events. Clients are able to use the archive information for later playback. Servers can communicate with each other via a *conference bus*. The servers keep track of the ongoing parties, and multicast data to the appropriate clients. A client applet joining a party is fed either the current event or the sequence of events to date, so that the current view is obtained.

As an example, for the foil server, when the user selects this service, the user first specifies his own URL for his foils, or a default set of foils outlining the Java Synchronous Collaboratory are used. The applet loads the first foil, if this is a new party, or the current foil if it is an ongoing party, and sets up a control panel, with "next" and "previous" foil buttons. When the control panel buttons are hit, an event is sent to the Java foil server, which logs the event, and multi-casts it to the applets of all clients in the party. The client applets use communication between the applet and JavaScript to cause a load of the appropriate URL at each of the clients, so that all clients see the same foil. Similarly, for the whiteboard, events marshalled by the applet correspond to mouse clicks, and for pixel drawing, for mouse movement. For collaborative forms, back-end database transactions would be triggered.

An additional feature of the architecture is the extensibility of the servers. Clients have the capability to dynamically attach new objects/methods to the server and can also change the behavior of the existing objects.

Scalability

Scalability of Java Collaborative servers can be achieved by running multiple server processes on a cluster of processors, as shown in Figure 2. We have previously presented techniques for scaling video and web servers using an IBM SP-2, shared and scalable multimedia file systems, parallel databases, and fine-grained load balancing via *TCP routers* [1]. Similar techniques can be used at the server end. Further, it is possible to configure the TCP router so that different sets of nodes are configured to execute servers for different services. The Architecture of the system includes a *Conference Bus* that allow the collaborative servers to communicate among themselves. Scalability can be achieved by providing inter-server communications on the conference bus and by partitioning parties across servers running on distinct nodes. High-bandwidth switch communications on an SP-2 based server can support efficient collaboration between all servers. Shared file-systems and parallel databases can provide efficient storage for state that requires to maintained.

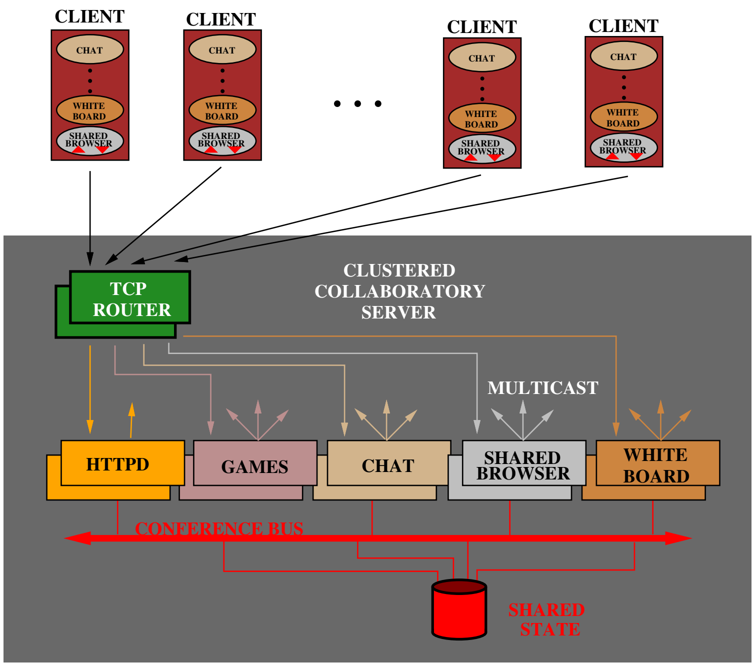

Figure 2: **Scalable Collaboratory Servers**

Demonstration and User Instructions

The current version of the system runs on any Java-enabled web browser, with the caveat that the Shared Browser (Foil Navigation) requires Netscape Version 3.0b2 or later. Please check the documentation at the web site [URL *http://cayuga.watson.ibm.com/shakinghands* for the most recent instructions.

The following user instructions, relatively interactive, are provided at the web-site listed on the title page. **Shaking Hands** is relatively easy, and a user goes through the following steps:

- First, you will be asked politely to log in! You will be presented a user identification menu to type in your collaboration name and a password. For now, it is required that the password and the userid be identical.

- You will see a screen which allows you to either create a new party or join an existing one. The parties are listed by the party type, which include chat,whiteboard and foil navigation. You can also start playing a chess game with someone on the net. The conference URL is applicable only to the case of Foil Navigation, which allows multiple browsers to simultaneously share a set of URLs. This can be the URL of any directory, but the html documents must be named *1.html* through *n.html* in that directory.

- If you create a new party (which you will do unless you have a password to an existing one), you will be prompted to create a party name and a party password.
- When you logon to a party via a party name and a party password, you will join the party. All participants in a shared browser (foil navigation) party will view the same html document at the same time. Furthermore, any client can drive the navigation.

Applications of the Technology and Futures

The Java Synchronous Collaboratory paradigm, and the applications in the demo can be used for numerous applications, some of which are listed here:

- **Foil presentations and Conferences:** The "foil" server outlined earlier, combined with the the white-board and chat servers, can be used for effective foil presentations. Audio conferencing could be added to make this application more compelling.
- **Conference record and playback:** The Java servers can keep a record of events in a party, allowing future playback. Clearly, this would be more effective with audio included.
- **Virtual visitor, department talks, remote classroom:** The foil presentation and record/playback could directly be applied to allow presentations by a virtual net visitor, and other related applications.
- **Collaborative multi-party games:** The demo illustrates the use for games using chess as an example. It is relatively easy to add other multi-party games.
- **Collaborative forms and transactions:** These applications could include joint entry of forms by a client and an agent. For example, a client and an insurance agent could jointly enter a insurance application, which could then be submitted to a back-end OLTP system.
- **Synchronous auctions:** In this application, each party could represent an item or a set of items to be auctioned. The last bid for the item could appear on the screens of all the clients.
- **Broker services:** This could be an extension of auctions, with the asking price being published, and bids being sent only to the broker or seller.

Numerous other business and entertainment applications fit into this paradigm. The Java collaboratory infrastructure can be used or extended to build such applications.

Summary and Conclusions

This demo introduces the Java Synchronous Collaboratory paradigm, with a set of applications built to illustrate the power and possibilities of this paradigm. The applications include a foil server, white-board, chat server, which can be used for presentations, seminars, conferences, and the like. A chess application is also included, to illustrate the use of this paradigm for inter-client multi-party games. We outlined other applications that can be built using this paradigm, to offer a compelling set of real-time applications between web-based clients.

References

[1] Daniel M. Dias, William Kish, Rajat Mukherjee, and Renu Tewari. A Scalable and Highly Available Web Server. In *COMPCON*, March 1996.

[2] Gosling J.,Rosenthal D.S.H., Arden,M., The NeWS Book, Springer Verlag 1989 .

[3] Furmanski W. "MOVIE - Muiltitasking Object-oriented Visual Interface Environtment", NPAC Northeast Parallel Architecture Center Technical Report, June 1992, in Fox G.C., Messina, P. and Williams R., Parallel Computing Works, Morgan and Kaufman , 1994 . http://www.npac.syr.edu/projects/hpsin/movie.html.

[4] Fox,G.C. and Furmanski, W. Web Based HPCC at NPAC. http://www.npac.syr.edu/projects/webspace/webbasedhpcc.html.

[5] K. Mani Chandy, Anand Chelian, Boris Dimitrov, Huy Le, Jacob Mandelson, Matthew Richardson, Adam Rifkin, Paolo A.G. Sivilotti, Wesley Tanaka, and Luke Weisman A World-Wide Distributed System Using Java and the Internet. http://www.cs.caltech.edu/ adam/CALTECH/papers/chandy$_e tal/hpdc.html$.