Web-Based Multimedia Tools for Sharing Educational Resources

Shervin Shirmohammadi [1], Abdulmotaleb El Saddik [2],
Nicolas D. Georganas [1], and Ralf Steinmetz [2,3]

[1]Multimedia Communications Research Laboratory,
School of Information Technology and Engineering,
University of Ottawa,
Ottawa, Canada

[2]Industrial Process and System Communications,
Dept. of Electrical Eng. & Information Technology,
Darmstadt University of Technology,
Darmstadt, Germany

[3]GMD IPSI,
German National Research Center for Information Technology,
Darmstadt, Germany

Abstract

Many educational resources and objects have been developed as Java applets or applications. One can access these resources by simply downloading them from different repositories. In many instances, it is necessary to share these resources in real-time, such as when an instructor teaches remote students how to use a certain resource or explaining to them the theory behind it. We have developed some tools for this purpose. These tools, which emulate a virtual classroom, are primarily designed for synchronous sharing of resources. They enable participants to share Java objects in real time and also allow the instructor to dynamically manage the tele-learning session.

1. Introduction

Since the advent of the Internet, the computing and communications industry has progressed very rapidly. Today, any user with a desktop computer can access and share multimedia documents with others through the Internet. It seems certain that in the near future every person, no matter where located geographically, will be equipped with some sort of network computing capability, either by means of conventional desktop computing or through information appliances. This not only means that geographically-distributed people will be able to easily communicate, but also “collaborate”; i.e., share multimedia documents and applications. Examples are joint editing, whiteboarding, joint browsing, and multi-user presentations, used in a variety of applications such as conferencing, collaborative design, training and telelearning.

Having noticed the above paradigm, the education community has started to conduct research on how to best utilize this global connectivity to facilitate and evolve education.

One of these advancements is the creation of repositories of educational resources, where Java applets and applications can be downloaded and used for educational purposes [14] . Java is used to eliminate the platform/operating system problem of heterogeneous environments, such that users wouldn't have to be restricted in their choice of a resource. This is specially important for tele-learning and distance education since some users might choose UNIX-workstations, while others might prefer Windows 95/98/NT or Macintosh. But with the introduction of Java it became possible to overcome these problems.

Another interesting issue that arises is how to share these resources among geographically distributed people? For example, imagine a virtual classroom, with participants from different parts of the world, where an instructor wants to discuss a specific topic using one of these Java resources. It is necessary for the participants to have the same view of the Java applet or application in real time. This signals the need for a sharing tool.

1.1 Web-based learning

There has been much interest in Web-based tools for telelearning. One can use e-mail, newsgroup, chat rooms, and other tools which are readily available. But for efficient resource sharing one needs tools specifically designed for such purpose. These tools can be used to share educational resources and they can generally be categorized into two groups: *synchronous* tools and *asynchronous* tools. Synchronous tools are those that allow sharing of resources in real-time, such as [1] [3] [7] [12] , and require users to be online at the same time. Asynchronous tools, such as Virtual U. [9] , allow for mostly off-line communication and sharing. Notice that these two methods are complementary technologies, not necessarily competing. Some researchers mistakenly choose to adopt only one method, but in fact you need both synchronous and asynchronous resource sharing for a complete system. Our research; however, is focused on the synchronous type. With this in mind, let us have a look at some basics about synchronous sharing.

1.2 Fundamentals of Real-Time Sharing

Basically speaking, the core technology behind any synchronous collaboration tool is a mechanism to enable a user to send updates to other users about the interactions that are made to a shared application, as illustrated in figure 1. For example, when one user draws a line on a whiteboard, the system informs the whiteboards of other users so that they also draw the same line. The mechanisms to propagate these "updates" vary according to the design or intended use of the system. Some systems send graphical display updates of the portion of the screen that was changed; the receiver simply redraws that portion using the graphics update. Some other systems send the system's graphical events that were generated as a result of a user's interaction, the receivers then process the events as if generated locally; hence reproducing the interaction at every user [6] [7]. Another approach is the use of object tokens, whereby an update message is preceded by a token that defines the semantic of that update message. By looking at the token, the receivers can determine what action to perform; for example draw a line, erase an area, etc. [12].

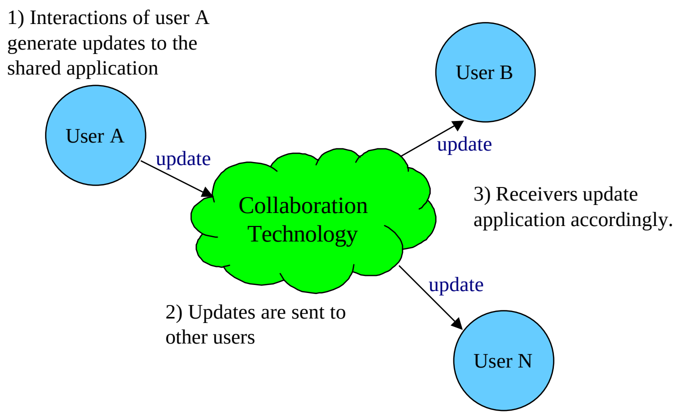

1) Interactions of user A generate updates to the shared application
User A
update
User B
update
Collaboration Technology
3) Receivers update application accordingly.
update
2) Updates are sent to other users
User N

Figure 1. A generic collaboration system

All of these approaches can be implemented using a centralized or fully distributed communication infrastructure. Furthermore, they can be implemented as real-time or near-real-time systems. However, they all have one thing in common: they all must use reliable communication, such as TCP or Reliable Multicast (RM), for their update messages. Although suitable for real-time video/audio data transfer, unreliable communication such as UDP or regular multicasting is not suitable for the transfer of application update messages since these applications, by nature, cannot afford to lose any update data.

To optimize the use of bandwidth and compensate for latency, we have to choose an approach that sends as small amount of information as possible for the updates. Graphics updates are therefore not suitable because of their bulkiness and heavy use of bandwidth.

Event updates and object tokens are better candidates. Object tokens are heavily based on the specific application, and must in fact be hard-coded into the shared application - an approach, which is not transparent. We have used both the event update and the object token technique in our tools.

In addition to dissemination of update messages, a collaboration system must also address issues such as latecomers, floor control, awareness, synchronization, and management. However, detailed discussion about these issues is outside the scope of this paper and can be found in other literature such as [12] .

In the next two sections we will present two resource sharing tools. The *Java-Enabled Telecollaboration System (JETS)*, has been developed at the Multimedia Communications Research Laboratory (MCRLab) at the University of Ottawa, Canada, under funding from the national Telearning Network of Centers of Excellence. The second system, *Java Application Sharing in Multiuser INteractive Environments (JASMINE)*, is a collaborative effort between the MCRLab and the KOM Laboratory at the Darmstadt University of Technology in Germany.

2. JETS 2000

Figure 2 below shows a screenshot of a sample JETS session.

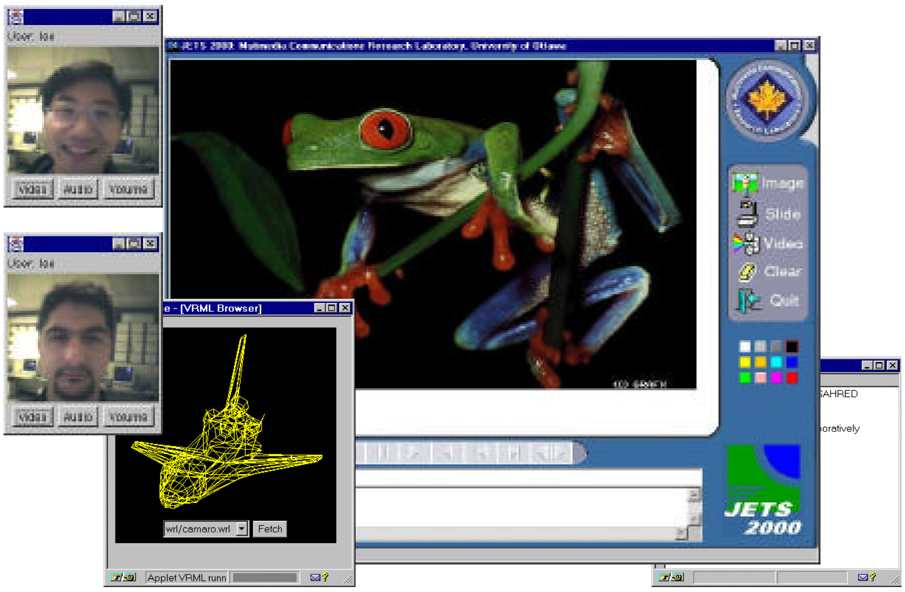


Figure 2. A sample JETS session with shared applets and A/V conferencing.

As can be seen from figure 2, JETS consists of many utilities that enable multimedia viewing and sharing. The following section briefly describes these utilities.

2.1 Utilities

Presentation

Using the JETS whiteboard, which can be used for shared color drawings, one has the ability to bring up presentations. JETS is capable of bringing slides from the Web server and displaying them. These slides can be PowerPoint slides saved in HTML format, or simple sequences of images. Participants can annotate on these slides in the same way as the images.

Pictures

The whiteboard is capable of bringing up images in JPEG or GIF format from the Web server and displaying them inside the whiteboard. The users can then annotate on these images and start a discussion. The built-in locking mechanism of JETS is used to forbid modification of the same object at the same time by more than one user.

Chat

The whiteboard also facilitates user conversations by a having a shared chat space which can be used by participants to exchange textual massages. The chat space is useful in cases where audio access is not available.

Video

A very useful feature of JETS is its ability to play ITU-T H.263 [13] compliant video in the whiteboard. The video must be stored on the Web server. When a user opens a video file and starts playing it, the video data is streamed down to all participants, decoded in real-time (processor permitting), and displayed in their whiteboard. Users can also play the video frame-by-frame as well as annotate on a frozen frame.

3D Viewing

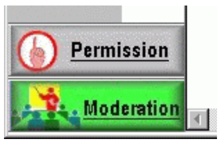
Another applet is a simple shared 3D viewer for VRML files which permits real-time collaborative interaction with simple VRML objects. The applet brings from the server simple VRML 1.0 files and displays them in wireframe mode. A user can then collaboratively interact with the 3D scene, with all the rotations, moving, and zooming reflected on all participants' screens.

A/V Conferencing and Recording

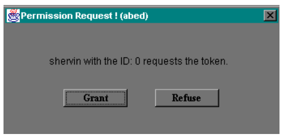
The *Java Video Conference Recorder (J-VCR)* tool further enhances JETS by providing services for audio/video conferencing, recording a session, and playback of a recorded session. J-VCR can record the session in the Synchronous Multimedia Integration Language (SMIL) format, which is a World Wide Web Consortium (W3C) standard. As a result, any SMIL-player such as RealNetwork's RealPlayer can be used to playback the recorded session [12] .

2.2 Moderation

Managing a shared session is a very important ability, specially for telelearning. An instructor can use this ability to moderate a session, as in the real world moderation of a classroom or meeting. Figure 3 below shows these abilities in JETS.

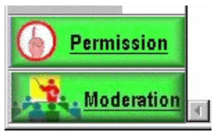


a

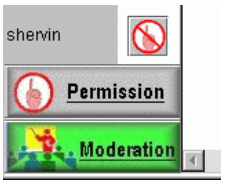

Permission Request ! (abed)

shervin with the ID: 0 requests the token.

b



c



d

Figure 3. Moderation Abilities in JETS 2000.

As shown in figure 3a, an instructor, who has logged into the session with special password, can enable moderation by pressing the moderation button. After this, any participant wishing to interact with the shared applications must ask for permission by pressing the permission button. The moderator will then receive a permission request which can be granted or denied (fig 3b). Should the moderator decide to grant permission, the participant sees a green light on the permission button (fig 3c) and can interact with the application. The moderator can "cut off" any participant by pressing the cut button next to the participant's name.

2.3 Architecture

JETS is a framework that permits sharing of Java applets. Since JETS uses the core Java packages, users don't need to install any additional Java classes on their system. This allows any user to access JETS and share applets with a Java-enabled browser. From a developer's point of view, JETS can be regarded as a set of Application Programming Interfaces (API) that the developer can use to build shared resources. It provides the developer with built-in consistency, access control, and data passing.

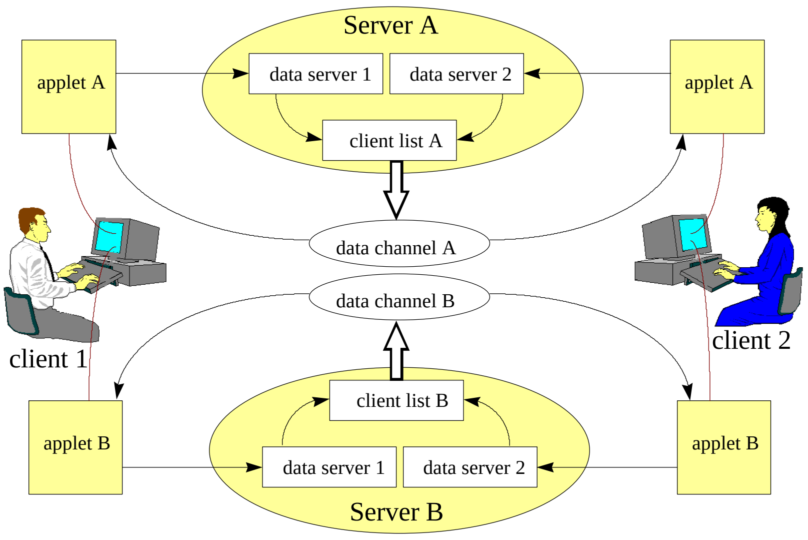


Figure 4. Client-Server communication in JETS

JETS uses a multithreaded server as shown in figure 4, where the main server launches a sub-server for each user joining the session. The sub-server is responsible for processing only the update messages or requests coming in from its own client. Once the sub-server receives the update message, it will send it to all other clients in the session. This will create a fast system response, at the expense of more resources utilized due to sub-server threads. However, usually only one client at a time can control and interact with an application (due to floor control as we will see), and most threads will simply be waiting For its client-server communication, JETS uses TCP/IP and UDP/IP sockets. In figure 4, when client 1 does some interaction with application A, his actions are reflected to data server 1 which runs as part of Server A for application A. Next, data server 1 relays the actions of client 1 to other clients which are listed in a client list on Server A. Finally, application A of client 2 receives those actions and reflects them on the screen of client 2.

3. JASMINE

JETS requires Java applets or applications to explicitly utilize its API in order to become shareable. While this is doable most of the time, it is sometimes difficult or impossible to modify the source code of a resource. There's therefore a need for a system that can transparently share resources without the need to modify the code for the resource.

Enter JASMINE: a Java resource sharing tool that allows applets and applications to be shared transparently without any modification [2] . Figure 5 illustrates the overall concept, where the JASMINE framework wraps around an applet that is to be shared (fig 5a). The framework listens to all events occurring in the graphical user interface of the applet and transmits these events to all other participants in order to be reconstructed there. The framework captures both Java AWT-based and Swing-based events. After capturing the event, it is sent to the communication module where the event is sent to all other participants in the session (fig 5b).

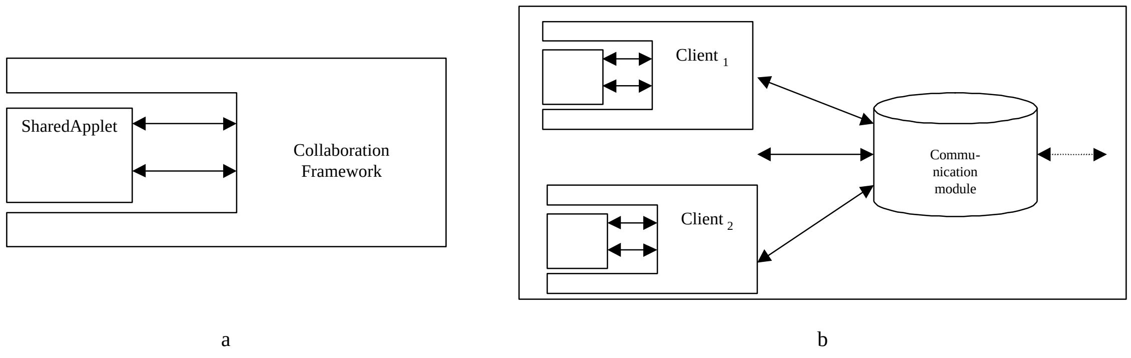


Figure 5. JASMINE's Architecture.

In the next section we describe how JASMINE is able to share resources without any modification to the resources' source code.

3.1 Client side

The JASMINE client can be seen as a component adapter. Every event occurring at the graphical user interface of the application is sent to this adapter, which then sends the events to the collaboration server (JASMINE-Server). The client is a Java application, which consists of the following components:

- Collaboration Manager
- Component Adapter
- Listener Adapter
- Event Adapter

These components are discussed next.

3.1.1 Collaboration Manager

The Collaboration Manager is the main component on the client side and provides the user with a graphical interface offering options such as joining the session, starting and sharing applications/applets and chatting with other participants. The collaboration manager is also responsible for dispatching external events coming from the communication module and forwarding them to the component adapter, as well as receiving internal events from the component adapter and sending them to the communication module.

3.1.2 Components Adapter

The Component Adapter maintains a list of the GUI-components of all applications and applets. This list is created with the help of the *java.awt.Container* class, which allows us to get references of all applet components [13]. With the help of the main window of an application, a list of the GUI components in the application can directly be created. Therefore, the main window of an application loaded by the Collaboration Manager is registered by the Component Adapter. However, Java applets do not use stand-alone windows. They are an extension of the class *java.applet.Applet* and thus of *java.awt.Panel*. Hence, applets can be easily placed into a window, which can then be registered as the main window for the applet. All these registrations are done at the Component Adapter.

After the registration is done a list of all Swing and/or AWT-components within the loaded application/applet is created. This task is done in the same order on each client, so that a component has the same reference identification at all clients. These references are used to point to specific components, which are the source of the events generated internally and the recipient of the events generated externally. With the help of the references, the recipient of an incoming event is located and the event is reconstructed on each client, as if it occurred locally.

3.1.3 Listener Adapter

The Listener Adapter implements several AWT listeners, which listen to *MouseEvent* and *KeyEvent* for all AWT-components except of *java.awt.Scrollbar*, *java.awt.Choice* and *java.awt.List*. For these components the Listener Adapter listens to *AdjustmentEvent*, *ItemEvent* and *ActionEvent*. When an event occurs on the GUI of the application, the Listener Adapter catches it, converts it to an external event, and forwards it to the Collaboration Manager. The Collaboration Manager in turn sends this event to the communication module, which propagates the event to all other participants.

3.1.4 Event Adapter

The Event Adapter works opposite to the Listener Adapter: it converts incoming external events to AWT events, which can then be processed locally.

3.1.5 Data Flow

Let us summarize the client side's architecture through the following data flow diagram. Figure 6 shows the overall event circulation of the system.

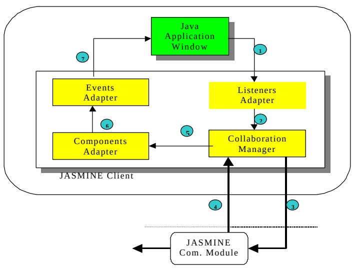
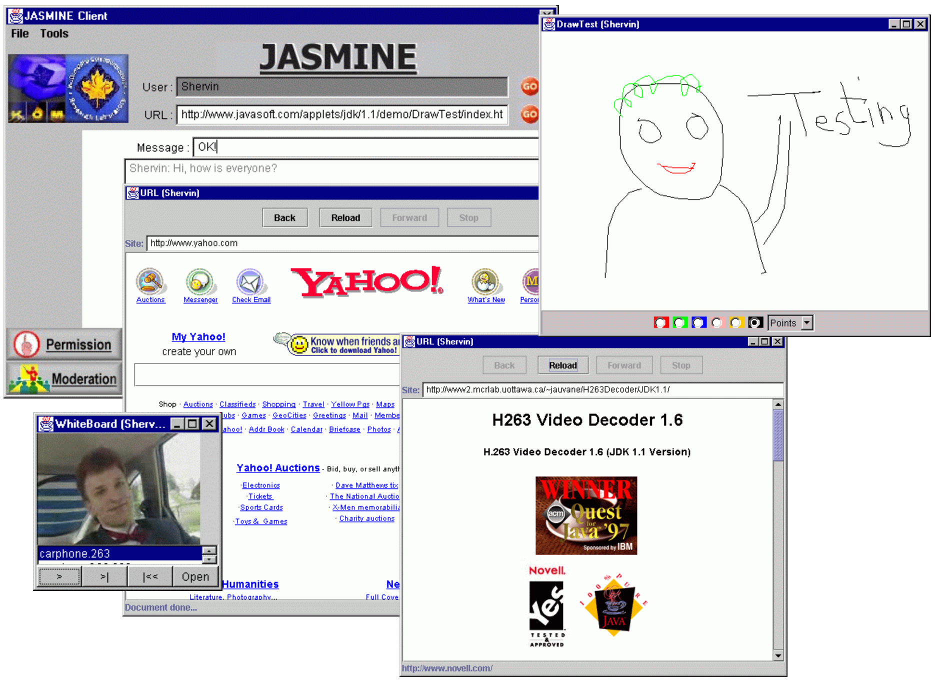


Figure 6: Events Circulation

There are two main data paths in the system: the first path is labeled with numbers 1,2 and 3. This path is used to send the internal AWT events to the communication module, and it works as follows: any Event occurred in a Java-application is caught by the Listener Adapter. The Listener Adapter first tests whether the event is an external or an internal event. It then sends only the internal events, which were not received from other clients, to the Collaboration Manager, which in turn sends the events to the communication module.

Via the second data path shown in figure 5 with numbers 4, 5, 6 and 7, the external AWT events received from the communication module are captured by the Collaboration Manager and the Component Adapter in order to reconstruct the event locally. After receiving the remote event, the Component Adapter extracts the information about its target component and sends this information together with the events to the Event Adapter. The Event Adapter converts the event to normal AWT events and sends them to the application, which then reacts to the event in the same manner as it would to a local user's interaction with the application's GUI.

3.2 Utilities



User : Shervin

URL : http://www.javasoft.com/applets/jdk/1.1/demo/DrawTest/index.ht

Message : OK!

Shervin: Hi, how is everyone?

Site: http://www.yahoo.com

Site: http://www2.mcrlab.uottawa.ca/~jauvane/H263Decoder/JDK1.1/

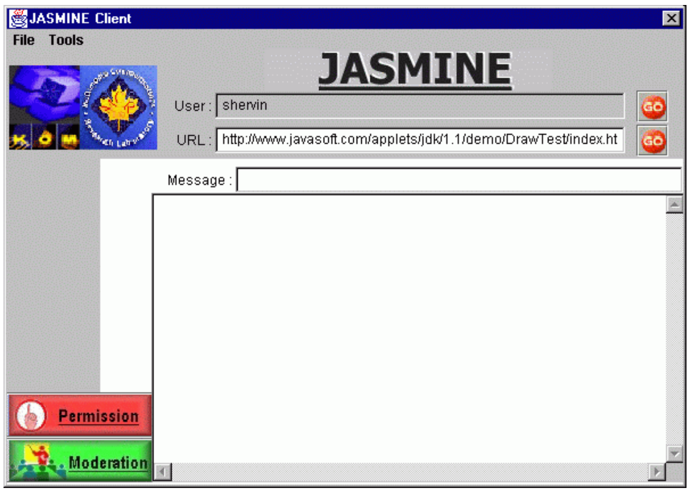
H263 Video Decoder 1.6

H.263 Video Decoder 1.6 (JDK 1.1 Version)

http://www.novell.com/

Figure 7. Screenshot of a JASMINE session.

Figure 7 shows a sample JASMINE session, where arbitrary applets and resources from the Internet have been brought into the session dynamically. The starting point with JASMINE is the collaboration browser shown in figure 8 below.



User : shervin

URL : http://www.javasoft.com/applets/jdk/1.1/demo/DrawTest/index.ht

Figure 8. Main JASMINE Application.

This is the interface that is presented to the participant. It includes a chat box where participants can exchange textual messages in real-time. Using the “URL field” at the top of the interface, one can type in the URL of any resource on the Web. That resource s then brought into the session dynamically and is shared among all participants in real time. In addition, a number of resources can be predefined before the session starts and can be started during the session from the “Tools” menu.

Since any Java applet or application can be brought into the session, JASIMNE has a potential unlimited extendibility property, since each resource enhances JASMINE dynamically.

As can be seen from figure 8, JASMINE has the same moderation capabilities as JETS, allowing for dynamic and real-time moderation of a shared session.

4. Related Work

There are many Java-based collaboration systems, none of which offer a management or moderation feature similar to JETS or JASMINE. Kuhmünch [10] has developed a Java Remote Control Tool, which allows the control and synchronization of distributed Java applications and applets. Similar to JETS, this approach uses an API that Java applets and applications must use to become shareable by the system. The Java Shared Data Toolkit

(JSDT) from JavaSoft is also an API-based framework. Habanero [1] is an approach that supports the development of collaborative environments. Habanero is in its terms a framework that helps developers create shared applications, either by developing a new one from scratch or by altering an existing single-user application which has to be modified to integrate the new collaborative functionality. Instead of using applets, which can be embedded in almost every browser, the Habanero system uses so-called “Happlets” which need a proprietary browser to be downloaded and installed on the client site. Java Collaborative Environment (JCE) has been developed at the National Institute of Standards and Technology (NIST) coming up with an extended version of the Java-AWT [2] called Collaborative AWT (C-AWT). In this approach AWT-components must be replaced by the corresponding C-AWT components [3].

All these approaches propose the use of an API, which has the cost of modifying the source-code of an application, re-implementing it or to design and implement a new application from scratch in order to make it collaborative.

Java Applets Made Multiuser (JAMM) [8] is a system, which is similar to JASMINE in terms of its objective: the transparent collaboration of single-user applications. The difference between JAMM and JASMINE is the way collaboration is achieved. In JAMM [6], the set of applications that can be shared is constrained to those that are developed using Swing user interface components as part of Java Foundation Classes, which are part of the standard JDK since version 1.2. JAMM’s set of applications is furthermore restricted to those which implement the Java serializable interface.

5. Conclusion

We have successfully deployed and tested both JETS and JASMINE in shared sessions. Based on our experience, the most interesting part of both systems is the moderation from the user perspective. This enables a useful and practical session to be carried out without interruptions. The ability to dynamically bring any resource into a JASMINE session is another key feature enjoyed and emphasized by all users.

Since the JETS system has an API, it gives more control over resources, which allows for fine-grained tasks such as MM synchronization which are not possible with JASMINE or any other transparent collaboration tool. On the other hand, JASMINE’s transparency to

[11] S. Shirmohammadi, L. Ding, and N.D. Georganas, "An Approach for Recording Multimedia Collaborative Sessions: Design and Implementation", Journal of Multimedia Tools and Applications (accepted, to appear).

[12] S. Shirmohammadi, J.C. Oliveira, and N.D. Georganas, “Applet-Based Telecollaboration: A Network-Centric Approach”, IEEE Multimedia, Vol. 5, No. 2, April-June 1998, pp. 64-73.

[13] K. Rijkse, "H.263: Video Coding for Low-Bit-Rate Communication", IEEE Communications Magazine, December 1996.

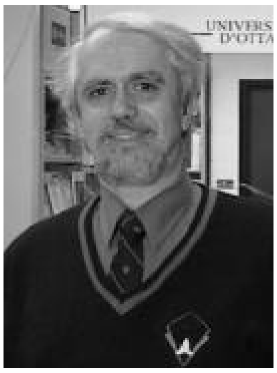
URL:

[14] Computer Science Teaching Center, Digital Library of Peer Reviewed Teaching Resources, http://www.cstc.org/

Author Biographies:

Shervin Shirmohammadi received his M.A.Sc. and Ph.D. in Electrical Engineering from the University of Ottawa in 1996 and 2000, respectively, where he conducted research at the Multimedia Communications Research Laboratory in various projects. His research interests include collaborative virtual environments, multimedia communications, and telecommunications software. He is the 1995 University of Ottawa Gold Medallist for highest standing in Engineering, 1998 Canadian Advanced Technology Association Award winner in Telecommunications Software, and Natural Sciences and Engineering Research Council of Canada scholarship holder (NSERC-PGS B).

Abdulmotaleb El Saddik is pursuing a Ph.D. in electrical engineering in the Department of Electrical Engineering and Information Technology at Darmstadt University of Technology, Germany. He is currently working on the Multibook project, founded by the German Federal Ministry for Education and Research (BMBF). His research interests include interactive multimedia visualizations, collaborative learning environments, multimedia communications, and telecommunications software. A. El Saddik received his M.Sc. (Dipl.-Ing.) degree in electrical engineering from Darmstadt University of Technology in 1995. He is a member of the Association for Computing Machinery (ACM), The Gateway to Educational Materials (GEM) and the Interactive Multimedia Electronic Journal of Computer-Enhanced Learning (IMEJ). He is also co-author of the Open Java book, published 1999 by Springer.



Nicolas D. Georganas is Professor and Director of the Multimedia Communications Research Laboratory (MCRLab) at the School of Information Technology and Engineering, University of Ottawa, Canada. He has been leading multimedia application development projects since 1984. He was General Chair of the IEEE Multimedia Systems'97 Conference (ICMCS97) (June 1997, Ottawa) . He has served as Guest Editor of the IEEE Journal on Selected Areas in Communications, issues on "Multimedia Communications" (1990) and on "Synchronisation Issues in Multimedia Communications" (1996). He is on the editorial boards of the journals Multimedia Tools and Applications, ACM/Springer Multimedia Systems, ACM Computing Surveys, Performance Evaluation, Computer Networks, Computer Communications, and was an editor of the IEEE Multimedia Magazine. He is Fellow of IEEE, Fellow of the Canadian Academy of Engineering, Fellow of the Engineering Institute of Canada and Fellow of the Royal Society of Canada. In 1998, he was honoured as the University of Ottawa Researcher of the Year and also received the University 150th Anniversary Gold Medal for Research. In 1999, he received the T.W.Eadie Medal of the Royal Society of Canada, funded by Bell Canada. In 2000, he received the A.G.L. McNaughton Medal and Award, the highest distinction of IEEE Canada, the Julian C. Smith Medal of The Engineering Institute of Canada and the President's Award of the Ottawa Center for Research and Innovation.

Ralf Steinmetz is professor at the "Electrical Engineering and Information Technology" as well as "Computer Science" departments of the Darmstadt University of Technology, Germany, since 1996. There he is in charge of a new chair position in the area of process communications and multimedia networking sponsored mainly by the Volkswagen-Stiftung. He is also one of the directors of the Information Transfer Office at the university. In late 1996 he was appointed director of the Integrated Publications and Informations Institute of the German National Research Center (GMD) in Darmstadt, Germany. Here he is in charge of co-operative work, workspaces of the future, interactive learning, media processing and mobile networking. His research interests are networked multimedia systems, co-operative applications, as well as mobile and service gateways for multimedia data. Dr. Ralf Steinmetz studied electrical engineering with the focus on communications at the University of Salford, England, and at the Technical University of Darmstadt, Germany, where he received the M.Sc. (Dipl.-Ing.) degree in 1982. Working as scientific assistant, he received the Ph.D. degree (Dr.-Ing.) in 1986 at this university. He mainly worked in the area of Petri-Nets and concurrent programming languages. Subsequently he joined the "Advanced Development Department" of "Philips Kommunikations Industrie" in Siegen-Eiserfeld, Germany, where he was involved in ISDN multimedia project workstations development activities. From 1988 until 1996 he worked at the IBM European Networking Center in Heidelberg, Germany. There he has been involved in various multimedia communication activities. He started his first activities on multimedia and networking and was in charge of a multimedia laboratory. In integrated multimedia projects that followed he acted as a key technical coordinator. He was the leader of the whole multimedia transport system development and subsequently was in charge of several application projects and their respective application support issues. He managed the multimedia department in Heidelberg which at that time became IBM's European Multimedia Center. He is editor and co-author of a multimedia course, which reflects the major issues of a first in-depth technical book on multimedia technology, 1993, (in German). He is editor of the magazine "Computer Communications" published by Butterworths-Heinemann and "Distributed Systems Engineering". He was in charge of a worldwide IEEE Multimedia Taskforce working group for magazine publication. There he is associate-editor in-chief of the IEEE Multimedia Magazine. He has served as chair, vice-chair and member of various program and steering committees of multimedia workshops and conferences. He is a member of ACM, GI, ITG, as well as senior member of IEEE.