Implementation and Management of Web-Based Collaboration Using Java

Shervin Shirmohammadi, Jauvane C. de Oliveira, and Nicolas D. Georganas
Multimedia Communications Research Laboratory
School of Information Technology and Engineering
University of Ottawa, Canada

Real-time collaboration systems, in which participants share multimedia documents and applications in real time, have been a subject of interest for many years. Although computer supported cooperative work (CSCW) systems have existed for a long time [1], Web-based collaboration tools, such as Microsoft NetMeeting, have started to emerge relatively recently. Moreover, there has been much advancement in Internet-related technologies lately. Among these, Java applets seem to have introduced the most exciting notion: platform-independent applications provided by the network. To run applets, users only have to make sure they have a Java-enabled Web browser on their platform.

From CSCW point of view, the next step in the evolution of applets is their real-time sharing among many users. Although applets can usually be converted to applications and be shared using conventional techniques, there is a fundamental difference between sharing applets and applications. In practice, applets can be shared through a Web browser or a network station. This is not only platform-independent but also requires no downloading and installation of the specific application on the user side. A user simply navigates to a given URL and joins a session. Furthermore, users will always have the latest version of the applet. This is not the case with applications. Applications cannot be shared using a Web-browser. This means that a package must be downloaded and installed on the machine of every user who wants to take part in the collaboration session; an approach which is very similar to collaboration using non-Java applications. Hence,

the emphasis in the application sharing method is the portability feature of Java, whereas the applet sharing method is more a network-centric approach.

Today, there are some experimental Java-based collaboration tools available, such as the NCSA Habanero and the Java Collaborative Environment (JCE), which allow sharing of basic multimedia applications; however, very few of these tools use applets as their base for collaborative applications.

There are many issues that such collaboration systems must address. Consider figure 1 which shows a simple sketch of a client-server collaboration system. When one user interacts with the applet, by for example clicking a button, one or more events are generated. Each event is sent to the server, which multicasts it to all other clients. The other clients then process the event as if it was generated locally and therefore recreate the intended actions of the original client. Another approach is to multicast screen shots of the application instead of the events; however, the literature [11] shows that in general, event broadcasting is more efficient than display broadcasting.

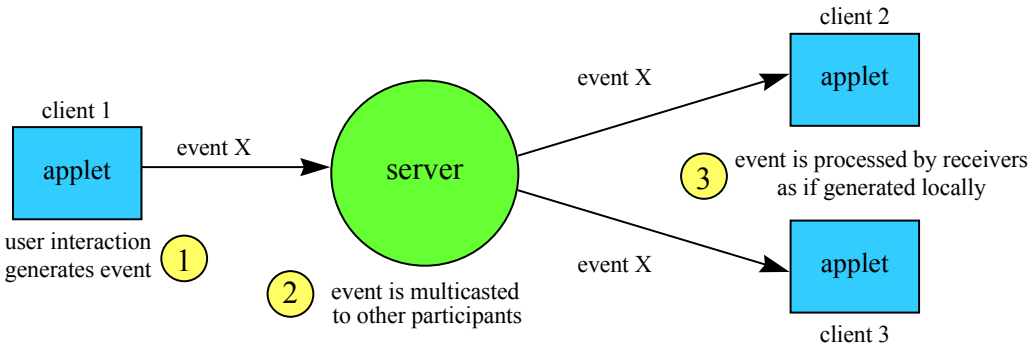

Figure 1. A simple client-server architecture for event-multicasting

The main challenges created by such system are latecomers, participants' awareness, management, event collision, and synchronization.

Latecomers: The problem here is as follows: when a user joins a session already in

progress, the user must be presented with the current states of the shared applications.

Awareness: Awareness is the ability of a given participant of a collaborative session to have feeling of both existence and actions of the others. There are as many as eleven different elements that provide such "feeling" [5].

Management: Often overlooked by existing implementations, a session manager is a key component of a functional telecollaboration system. There is a need for a chairperson who controls participants' access rights for different applications. This person can dynamically set individual user's access rights such as no-access, view-only, and view/interact.

Event Collision: The problem occurs when the events of two or more users which are generated at approximately the same time affect the same data and create unwanted results.

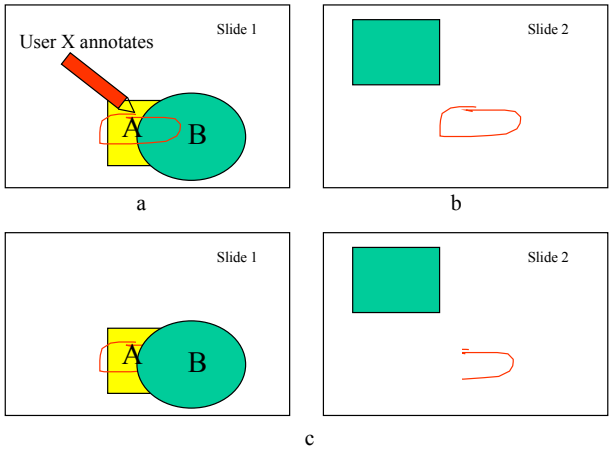

Figure 2. Event collision a)what user X intended b)what actually happened because another user changed the slide c)another scenario of what happened: half of the annotation is on slide 1, the other half on slide 2

For instance in an on-line presentation, one user might advance the presentation to the next slide while another is trying to annotate on the previous slide. As a result, the annotation might appear on the new slide, or half of it might appear on the previous slide and the other half on the new slide. This situation is depicted in figure 2. Event collision

is usually addressed by some type of pessimistic access control such as locking.

Synchronization: Typically, individual users in the same session have different processing powers and different network access in terms of network bandwidth and network latency. This creates the possibility that the state of one copy of the application is different form the state of another copy. Lack of synchronization will lead to both temporal and application state inconsistencies.

In this article, we present a client-server architecture for sharing Java applets. We will show the collaboration issues that we came across while designing our Java-Enabled Telecollaboration System (JETS) prototype and propose solutions based on our experience with JETS applets consisting of a whiteboard, video player, 3D viewer, and text editor. In addition, we will illustrate an applet-based management scheme that is the result of our experimentation with JETS' Management System.

Also, looking at other Java collaboration tools, JAMM is a system that more closely compares with JETS since it also tries to achieve collaboration using Java applets rather than Java applications. The main difference between JAMM and JETS is that JAMM uses transparent collaboration, which means that the events are automatically intercepted and sent to the other copies by the collaboration engine; whereas JETS requires the developer to handle the events once they are intercepted.

We will show through examples that even though a transparent collaboration system makes it easier for a developer to write typical applets, it doesn't always work for the case of multimedia applets. Additional information must be sent together with the event in order to avoid lack of consistency due to the difference in access parameters such as network bandwidth and delay.

References

1. J. Grudin, "Computer-Supported Cooperative Work: History and Focus", IEEE Computer, Vol. 27, No. 5, pp 19-26, May 1994.

2. O. Kim, P. Kabore, J. Favereau and H. Abdel-Wahab, “Issues in Platform-Independent Support for Multimedia Desktop Conferencing and Application Sharing”, Proceedings of the Seventh IFIP Conference on High Performance Networking (HPN'97), White Plains, NY, April 18 - May 2, 1997

3. S. Shirmohammadi and N. D. Georganas, “JETS: a Java-Enabled Telecollaboration System”, Proc. IEEE ICMCS, IEEE Computer Society, Los Alamitos, Calif., 1997, pp. 541-547.

4. T. B. Downing, *Rmi : Developing Distributed Java Applications With Remote Method Invocation and Object Serialization*, IDG Books Worldwide, California, 1998.

5. C. Gutwin and S. Greenberg, “Workspace Awareness for Groupware”, Proc. ACM Computer-Human Interface (CHI ’96), ACM press, New York, 1996.

6. H. P. Dommel and J. J. Aceves, “Floor Control for Multimedia Conferencing and Collaboration”, ACM Multimedia Systems, Vol. 5, No. 1, 1997, pp. 23-38.

7. J.C. Oliveira, "TVS - A Videoconferencing System"; Master Dissertation (in Portuguese), Computer Science Department, Pontifical Catholic University of Rio de Janeiro, Brazil, August 1996

8. R. Elmasri and S. B. Navathe, *Fundamentals of Database Systems*, 2nd edition, Benjamin/Cummings Publishing Company, California, 1994, Chapter 8.

9. W. Robbins and N. D. Georganas, “Shared Media Space Coordination: Mixed Mode Synchrony in Collaborative Multimedia Environments”, Proc. IEEE ICMCS, IEEE Computer Society, Los Alamitos, Calif., 1997, pp. 466-473.

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

11. J. Begole, C. Struble and C. Shaffer, “Leveraging Java Applets: Toward Collaboration Transparency in Java”, IEEE Internet Computing, March-April 1997, pp. 57-64

12. Multimedia Communication Forum Inc., “Multimedia Communication Quality of Service”, MMCF document MMCF/95-010, Approved Rev 1.0, September 24, 1995.

13. H. Abdel-Wahab, J. Favereau, O. Kim and P. Kabore, J. Favereau “An Internet Collaborative environment for Sharing Java Applications” IEEE Computer Society

Workshop on Future Trends of Distributed Computing Systems (FTDCS'97), Tunis, Tunisia, October 29 - 31, 1997

14. J. Begole, C. Struble, C. Shaffer and R. Smith, "Transparent Sharing of Java Applets: A Replicated Approach," Proceedings of the 1997 Symposium on User Interface Software and Technology (UIST'97), ACM Press, NY, 1997, pp. 55-64.

URLs:

JAMM (Java Applets Made Multiuser): http://simon.cs.vt.edu/JAMM/

JCE (Java Collaborative Environment): http://snad.ncsl.nist.gov/madvtg/Java/Java.html

JETS (Java-Enabled Telecollaboration System) http://www.mcrlab.uottawa.ca/jets

NSCA Habanero: http://www.ncsa.uiuc.edu/SDG/Software/Habanero/