Secure mobile replication for collaborative virtual reality

Work supported by the European Commission project IST AIT-VEPOP[1]

Fabien Costantini[1], Christian Toinard[1], Nicolas Chevassus[2]
[1] CNAM-CEDRIC 292 rue Saint-Martin 75141 Paris Cedex 03 France, {costanti,toinard}@cnam.fr
[2] EADS Corporate Research 12 rue Pasteur BP 76 92152 Suresnes France, nicolas.chevassus@eads-nv.com

Keywords

Collaborative virtual reality, mobility, security.

Abstract

Our paper describes how to share and distribute a scene tree among mobile workers. Thus, a participant can work on a disconnected basis to improve a sub-set of the global scene tree. The scene tree reforms automatically and consistently during a meeting and distant workers cooperate in real-time to build the shared scene. The system rests on a replication mechanism to provide private and shared workspaces. A distributed designation and protection of the scene tree is proposed. Moreover, work consistency is provided in a distributed way without limiting the ability of parallel working. A refreshment protocol enables to maintain accurate state for the replicas with few network consumption. Intensive usage of the User Datagram Protocol and multicasting enables to reach good performances. At last, authentication and confidentiality is provided for User Datagram Protocol and multicasting. Thus, the solution can be deployed easily, efficiently, securely and at low price over the Internet.

1 Motivation

Groupware tools enable to share resources. Microsoft NetMeeting [13] uses TCP (Transmission Control Protocol) connections to share windows. Sharing is slow since the window content has to be transmitted. [5] proposes a room metaphor that ease natural social interaction. The solution does not address sharing and distribution of a scene tree. It does not help workers to reunify disconnected works.

Different solutions [1,7,10,12] make all use of a central server. The server receives an interaction from a replica, processes the message, and sends out further messages to the other replicas. [14] uses the same principle but broadcasting can be replaced by a reliable multicast protocol. The main drawback is a poor performance. First, a central server introduces a bottleneck in the system. Second, these solutions use generally multiple TCP (Transmission Control Protocol) connections. Third, using a reliable multicast does not suit for real-time updates and does not guaranty a consistent progression of the work.

Distributed virtual reality environments [2,3,6,9,11] consider how to reduce the network traffic due to a high number of moving objects. These systems do not support parallel working [2] or do not guaranty the work consistency [3,6,9]. Moreover, client-server pitfalls [3,9] or quality of service requirements [6] limits these solutions. [3,9] and [8] in another context consider freshness as a consistency property. But, they do not guaranty that a modification is carried out on the latest state on the corresponding object.

So, the solutions do not consider how a global scene tree can be distributed among different workers for disconnected improvements and further conciliation. They solutions do not integrate security properties. They do not define how to authenticate participants and achieve confidentiality of best effort communication (i.e. multicast or point-to-point datagram). At last, they do not define how to reserve a multicast address, resolve the multicast connectivity problem and/or locate mobile designers that use a dynamic host configuration.

[1] Contract Number IST 1999-13346, EADS Corporate Research, EADS Airbus, CNAM-CEDRIC, Flow-Master, University of Oulu

2 Solution

2.1 Project management

Project negotiation: a project manager and selected participants communicate using the email to set-up a project. Security is achieved using S/MIME for authentication and confidentiality through X509 certificates. The manager can transfer the project responsibility with a signed text certifying the transfer.

Scheduling: a subset of the project members uses the email to schedule a meeting. A multicast address is negotiated within the different emails. That address allocation can be omitted. Moreover, conflicting address will be detected and resolved automatically during the meeting.

Distribution of a session key: scheduling serves also to distribute a private key PK_S. PK_S will be used as a session key during the scheduled meeting. That private key is transmitted securely using S/MIME.

2.2 Meeting

Shared scene tree: the application maintains a tree of objects. Each object inherits collaboration attributes from our distribution classes. Each peer maintains a replica of that global scene. No server maintains the global scene that is accessible during a meeting. If participants are absent the shared scene is a subset of the global scene.

The scene tree gives a high level and concise information about the graphical objects. Collaboration attributes include ownership and protections attributes. Protection attributes enable to prevent distant participants from observing or modifying the corresponding object. There is only one owner at a time of a given object. Only the owner can modify the corresponding object. But, the ownership can be transmitted as described in the sequel.

Real time membership and global scene merging: an entering peer multicasts the participant name. Distant participants reply as multicasting their names. Thus, a new participant maintains locally his knowledge of the meeting membership. Moreover, the protocol allocates a session color to each participant.

Afterwards, a global scene merging is processed. Each peer multicasts a list part [*LIST*: V_L, *TotParts*, *NPart*, (G_1, V_1),…, (G_N, V_N)] where V_L is the version number of the list, *TotParts* is the total number of parts forming that list, *NPart* is the part number and each couple (G_X, V_X) defines the global name and the version number for an object *X*. Using *TotParts* list messages, a peer announces the objects it enters into the meeting. Afterwards, the announcing peer multicasts an object state [*State*: V_L, G_X, V_X, T_X, S_X] for each object *X* of the list where T_X is the type of *X* and S_X is the state associated with V_X. At the end of the scene merging, each peer has a copy of the shared scene. Since a distant peer has the list of the objects, it can ask the retransmission of a missing object *X* by requesting a version number equal or higher to V_X.

Real time operations: a peer modifies an object *X* by multicasting a message [*State*: V_L, G_X, V_X, T_X, S_X]. A receiver uses the event either to create a new object or to update an existing object. The receiving peers do not acknowledge that message.

Distributed designation and protection: a peer computes locally a unique name when a designer creates a new node. A unique name contains the *IP* (Internet Protocol) address (*@IP*) of the creation machine, a local date and the position within the tree (e.g. *@IPA,dateA,1.3* corresponds to the first child and third grandson of the node *@IPA,dateA*). Only the owner of a node can create/delete a child (e.g. *A* creates the node *@IPA,dateA,1.3* as owner of *@IPA,dateA,1*). That way, each computation peer defines unique names in a distributed way. Moreover, those distributed names can be created on a disconnected basis. It should be noted that a peer can change its *IP* address without any difficulty since the names remain unique in time and space. Creation/deletion are carried out according to the node ownership. Thus, a distributed protection is provided satisfying participants' rights.

Refreshment: since the updates and deletions are not acknowledged, a distant peer can lose events. Using the refresh service, a peer requests a list to a distant peer. The requester replies by multicasting the list messages. The requesting peer uses the received list to remove or add objects. If the objects exist already, the received list enables to ask retransmission for newer versions.

Consistency through the ownership transfer: a precise semantic of consistency is proposed. The ownership transfer guaranties that a modification is processed starting from the latest state of the corresponding object. The owner refuses the ownership transfer when a write attribute is disabled. Otherwise, the requesting peer receives at a time the current state and ownership.

A transfer is running in three phases. First, the requester multicasts a message to locate the owner. Second, the owner replies to the requester with a point-to-point message including the ownership and the current state of the node. Third, when receiving the reply, the granted peer sends a point-to-point acknowledgement that terminates the transfer. A reliable transfer is provided using the same sequence number for the different messages plus retransmissions.

Security and address allocation: by using the session key PK_S, each peer achieves a symmetric encryption of each message. The shared secret PK_S was distributed securely with S/MIME during the scheduling of that meeting. Moreover, the peer with the smallest network address will transmit frequently a new session key through the secure multicast channel. Since the session key is changing frequently, the system limits cracking.

If another application uses the same multicast address, the system will detect automatically the situation because the message cannot be decrypted using the session key PK_S. In that case, a peer will propose a new address checking first that there is no activity on the proposed address.

Multicast connectivity: the system detects automatically if a distant participant is not accessible with multicast. When a peer *A* enters a meeting, it sends an email to the group of participants including its current *IP* address. In the mean time, *B* waits for multicast messages to decide if *A* can be reached through multicast. If multicast is available between *A* and *B*, the session starts without waiting any email. Otherwise, *A* and *B* communicate through point-to-point transmissions using the IP address included in the received email.

Leaving: a leaving participant selects the nodes that he wants in his isolated workspace. Generally, all the owned nodes are automatically included in his isolated workspace. Other nodes can also be selected. Thus, the leaving participant defined the sub-set of the shared scene he includes in his isolated workspace. At the end of the selection, the peer multicasts a leaving message to inform the distant participants of its departure. Then, the peer leaves the multicast address. Leaving message does not need any acknowledgment. An unreliable departure suffices since the peer has already the selected nodes within his isolated workspace. The isolated workspace is a private space that can be improved on a disconnected basis and reunified automatically as a global scene during a further meeting.

References

[1] Abdel-Wahab,H., Favereau,J., Kim, O., Kabore, P. 1997. *An Internet Collaborative Environment for Sharing Java Applications.* IEEE Computer Society Workshop on Future Trends of Distributed Systems (FTDCS'97), Tunis, Tunisia.

[2] Barrus,J.W., Waters,C., Anderson,D.B. 1996. *Locales: Supporting Large Multiuser Virtual Environments*, IEEE Computer Graphics and Application. November, Vol.16, No.6, pp.50-57.

[3] Broll,W., Schick,D. 1998. *DWTP-a basis for networked VR on the Internet*, Electronic Imaging'98, San Jose, conference proceedings.

[4] Costantini,F., Sgambato,A., Toinard,C., Chevassus,N., Gaillard,F. 2000. *An Internet Based Architecture Satisfying the Distributed Building Site Metaphor.* IRMA2000 Multimedia Computing Track, Conf. proceed. IDEA Group Publishing.

[5] Greenberg,S., Roseman,M. 1998. *Using a Room Metaphor to Ease Transitions in Groupware.* Research report 98/611/02, University of Calgary.

[6] Defense Modeling and Simulation Office. 1997. *HLA Data Distribution Management Design Document Version 0.5*, February, U.S. Department of Defense, Washington D.C., http://www.dmso.mil/project/hla.

[7] Eleftheriadis, A., Herpel, C., Rajan, G., Ward, L. 1998. *Text for ISO/IEC FCD 14496-1 Systems,* ISO/IEC JTC1/SC29/WG11CODING OF MOVING PICTURES AND AUDIO.

[8] Handley,M., Crowcroft,J. 1997. *Network Text Editor (NTE) A scalable shared text editor for the MBone.* Proceedings of ACM Sigcomm 97, Canne, France.

[9] Hagsand,O. 1996. *Interactive Multiusers VEs in the DIVE system*, IEEE Multimedia, Vol.3, No.1, pp.30-39.

[10]Kuhmünch,C., Fuhrmann,T., Schäppe,G. 1998. *Java Teachware - The Java Remote Control Tool and its Applications*'. In proceedings of ED-MEDIA'98.

[11]Macedonia,M.R., Zyda,M.J., Pratt,D.R., Brutzman,D.P., Barham,P.T. 1995. *Exploiting Reality with Multicast Groups*, IEEE Computer Graphics and Applications, Vol.15, No.5, pp.38-45.

[12]Shirmohammadi,S., Oliveira,J. C., Georganas,N. D. 1998. *Java-Based Multimedia Collaboration: Approaches and Issues.* Proc. International Conference On Telecommunications (ICT '98), Vol. I, Porto Carras, Greece.

[13]NetMeeting, http://www.netmeet.net, 2000.

[14]Sun Microsystems. Java Shared Data Toolkit. http://www.sun.com/software/jsdt/index.html,1999.