ARCHITECTURES AND PROTOCOLS FOR EDUCATIONAL VIRTUAL ENVIRONMENTS

Ch. Bouras [1,2], *Th. Tsiatsos* [1,2]

[1]Computer Technology Institute, Greece

[2]Computer Engineering and Informatics Dept., Univ. of Patras, Greece

ABSTRACT

Shared or networked virtual environments are one of the most promising uses of the virtual reality technology. While a lot of research has been done in the area of multi-user virtual environments corresponding to the sharing of events very few research has been done on specific services and functionality. However both the requirements and the kind of the offered services affect significantly the design of a system. In this paper we present the main issues in the design of a platform to support the distant education process using networked virtual environments. We also introduce the term Educational Virtual Environments and their basic requirements. According to these requirements of the educational virtual environments we propose both a suitable architecture and a protocol for the interaction between the components of this architecture.

1. INTRODUCTION

A simple Virtual Environment (VE) is a computer-generated simulation, which aims to provide its users with a sense of realism. More specifically a VE is a computer system, which generates a 3-dimensional (3D) virtual environment, with which the user can interact, in such a way that he receives real time feedback [8]. If multiple users use the same VE and they are able to interact to each other the above definition would extend to multi-user, shared, or networked VE (SVE or NVE). An extension of an SVE would be a Collaborative VE-system (CVE) which is an SVE aimed at a collaborative task.

According to the above definitions a simple definition of an Educational Virtual Environment (EVE) is a CVE aimed not only at a collaborative task but also at additional educational tasks such as synchronous and asynchronous learning. An EVE is a set of virtual worlds or a virtual world, which offers educational functionality to its users. The avatars (the graphical representation) of these users populate the EVE and they are provided with additional behavior such as gestures, interaction, movements and sound. In order to implement an integrated EVE, it should satisfy some basic requirements. These requirements are listed below:

- *High level of presence:* The user should be represented by an avatar of his choice, which can simulate some basic realistic actions of the users, such as gestures and movement, giving them a shared sense of space, presence and time [10].
- *Interaction:* The EVEs should support two types of interaction: (a) **Multi-modal user-to-user interaction** via chat, voice communication and gestures. (b) **User-system interaction**, which would be based on navigational aid and commands that the system should provide to the user for a specific function as well as the manipulation of 3D objects.
- *Scalability:* The EVE must be scalable to a large number of users in order to support large virtual educational communities.
- *Consistency:* Consistency of the EVE should be realized by distributing and synchronizing user input as well as user independent behavior in order to achieve the impression of a single shared world.
- *Coherence:* Coherence with the sense of a uniform structure of the provided services, concerning mainly the functional and operational characteristics rather than its visual representation in the EVE.
- *Quality of Services*: an EVE should ensure the quality of the provided service

To achieve these goals, we need to develop a platform for educational virtual environments in order to offer educational services to the users in a sufficient way. The main steps in order to realize this are to specify both an architecture for educational virtual environments and a suitable protocol. In this paper we present such an architecture and a protocol.

The remainder of this paper is structured as follows. In the next section we present the design rationale of a platform for educational virtual environments. We then present a suitable communication model and the components of an architecture for this platform. Following this, we present main issues in the design of a protocol for the interaction between the components of this architecture. Finally we present some concluding remarks and our vision for the next steps.

2. DESIGN RATIONALE

In order to provide educational services in an effective way using shared virtual environments we need to satisfy the previous described requirements of an educational virtual environment, as well as to provide specific functionality:

- Transmission of the VE to the users
- Synchronization of the VE
- Transmission of both a user's avatar and users' profile to the rest of the users
- Capability for transmission of educational material (audio, video, pictures, text etc.)
- Support of users with different network characteristics
- Reliable transmission of data in order to ensure QoS
- Effective management of system failure

These reasons guide us to design and implement both an architecture and a protocol for the interaction between the components of this architecture, in order to support educational virtual environments. Many solutions have been proposed, and described in [9], for the network topology of the components of the networked virtual environments, such as peer-to-peer, multicast, client/server, and multiple servers. Each solution has its own advantages and drawbacks, which are presented at [3].

In the design of our platform we also take into account the Networked Virtual Environment Information Principle:
"The resource utilization of a NVE is directly related to the amount of information that must be sent and received by each host and how quickly that information must be delivered by the network" [10].
This principle is described in [10] as follows:

$$Resources = M*H*B*T*P \qquad (A)$$

where
M = number of messages transmitted in the NVE
H = average number of destination hosts for each message
B = average amount of network bandwidth required for a message to each destination
T = timeliness with which the network must deliver packets to each destination
P = number of processor cycles required to receive and process each message

Therefore, in order to reduce the resource utilization we must lower some of the above variables. The problem here is that improving one of the variables we impact another aspect of the system worsening another variable. In order to improve the system performance we propose a new communication model between the components of the system. This model is presented in the following paragraph.

3. REFERENCE ARCHITECTURE

Our basic idea is to divide the processing load for the necessary services of an EVE (such as shared objects, chat and audio communication, etc.) to a set of servers aside from the communication of the users or the management of the virtual worlds as described in other models [3]. In addition, the whole system will serve as a virtual representation of the relevant theme and a presentation mean for the available material. This implies that the 3D community that will be supported by this communication model will consist of a number of smaller VEs. This offers a "segmentation" of the virtual community and led us to design a communication model that consists of a number of message servers. Each message server hosts some specific VEs and it is back-up server for the rest of VEs. The set of message servers constitutes a locus of control of the whole system. More specifically the proposed architecture is based on the following different components:

- *Message Server (MS):* The message server has three main tasks: (a) to transmit virtual world contents, (b) to offer scalability to the system and (c) to keep the 3D world consistent.
- *Application Server:* In order to provide specific applications, we use dedicated application servers. According to the previous described requirements the main applications that should be offered are chat and audio communication as well as shared objects to support the educational process and to represent the educational material. For these reasons we use three types of application servers: (a) **Audio server**, which is responsible to provide real-time streaming audio capabilities to the whole system. (b) **Chat server**, which is responsible for the chat capability. (c) **Shared object server**, which contains all the specific objects that are shared in the 3D virtual environment.
- *The clients:* The client of the system interacts only with the message server and the shared object server.

The interaction between the components of the architecture can be seen in **Fig.1**.

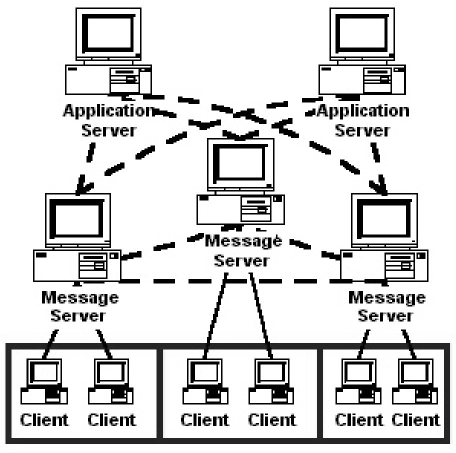

Figure 1. Reference Architecture

The main advantage of this communication model is that the variable **P** of the equation (A) is reduced, examining each message server separately. This occurs because each MS is responsible only for a number of the whole VE and the main processing power for specific applications is divided in the corresponding application servers.
The drawbacks of this solution are (a) that the variable **T** is increased due to the fact that we are using many servers, and (b) that the variable **M** is increased due to the fact that the servers should change additional messages for their synchronization.
In order to deal with the above problems we propose to use multicast communication between the message servers and the user groups that participate in the same VE. With the use of multicast communication we reduce the variable **H** reducing the network traffic.
We believe that this model is well suited for an educational environment for the following reasons: (a) it offers scalability due to the fact that the load is divided, and servers for additional services can be added, without to affect the end user (b) it offers concerted management and authentication procedure, (c) the clients have not excessive system and network requirements, (d) there is no central point of failure, and (e) it is flexible, because if the number of users is small, some of the dedicated servers to one service can be consolidate in a message server.

4. PROTOCOL FOR EDUCATIONAL VIRTUAL ENVIRONMENTS (pLVE)

In order to support the above architecture a suitable protocol should be designed and implemented. Many protocols for shared virtual environments is presented such as *Distributed Interactive Protocol (DIS)* [6], [7], *Distributed Worlds Transfer and communication Protocol (DWTP)* [2], *Interactive Sharing Transfer Protocol (ISTP)* [10], *VS Server Client Protocol (VSCP)* [5], *VRML Interchange Protocol (VIP)* [11] and *Virtual Reality Transfer Protocol (VRTP)* [4]. The most of the above protocols are well suited for shared virtual environments, however in order to support educational virtual environments we should be able to manipulate special data types, which corresponds to the previous described requirements of the EVEs. These data types are the following: (a) *Triggers:* triggers are messages, which need little bandwidth to be transmitted, (b) *Streams:* streams are used for the transmission of audio and video data. (c) *State update messages:* these messages are responsible for the consistency of the virtual world and it includes, among others, the avatar movement and the non-predefined object manipulation. (d) *Files:* files are the 3D virtual worlds, the users' avatars and the additional educational material (3D objects, texts, pictures, etc) provided by the users.

The need of supporting the above data types guides us to design and implement a protocol for the interaction between the components of the previous described architecture. This protocol, which is named pLVE, is presented in [1], however some advanced issues in its the operation is presented in this paper. These issues are concerning the scalability of the system, the reliability of the communication, the connection of a new user and the consistency of the virtual environment.

4.1 Achievement of scalability

One of the main problems in the Networked Virtual Environments is the fact that they are not able to support large number of simultaneous users and/or provided services. For this reason pLVE uses various mechanisms for the achievement of scalability of the overall system. These mechanisms are the following:

- Both multicast and unicast communication is supported. If a client does not support multicast communication, it would connect to the corresponding message server using unicast communication.
- Each message server is back-up server of the rest of message servers.
- Application servers are used in order to reduce the processing load of the message servers.
- Each message server can be "client" of another message server, if the second message server has exhaustive processing load. For the achievement of the load balancing the processing and network load of each message server is computed (according to the shared objects in each VE, the active unicast and multicast connections, etc.) and each new unicast connection is assigned at another message server. If the processing and network load of all message servers is the same, then the new connection will be assigned to the message servers in a round robin way.

4.2 Achievement of reliability

Another problem that the protocol can solve is the packet loss due to network problems. The problem is bigger with the use of UDP multicasting, which does not guarantee reliable transmission of data. The generally used solutions are the following: (a) Usage of ACK message by the sender for the receiving-back of a data packet. (b) Usage of NACK message for the not receiving back of the data packet. In this case there is the problem that the receiver should have a mechanism in order to be aware of a packet loss.

In our case the transmission and receiving of a message is realized as follows:

- Transmission of a message: A client sends the message to a message server using unicast communication. Then, the message server forwards the message to the responsible multicast group or to the rest of clients (that does not support multicast communication).
- Receiving of a message: The participating clients in a multicast group receive the message using the multicast group, and the rest of clients (that does not support multicast communication) receive the message using unicast connection.

Therefore, our protocol uses the following mechanism in order to ensure reliability in each multicast group: only the responsible message server in a specific multicast group can transmit ACK. This ACK should be arrived in the clients in a predefined time. If the sender of a message does not receive this ACK, then this client re-transmits the message to the message server, who should forward this to the multicast group.

4.3 Achievement of connection

When a new client requests a connection with a specific virtual environment, the responsible message server transmits the content of the virtual environment along with the necessary information to achieve the connection: the corresponding multicast group or the unicast address for the specific environment. In addition the message server stores a copy of the virtual environment along with the last changes in the shared objects in the world. The initial connection is a reliable TCP/IP connection for the transmission of the initial necessary data, such as the file of the virtual environment and the avatars of the other users. The message server sends the necessary variables, for the multicast group and port and assigns a unique ID number to the client of the new participant. If the new client is no multicast capable it uses a unicast address to communicate with the message server. If the message server is not able to serve the new client due to excessive network and communicative load, another message server can be used in order to serve the new client. In this case the unicast address of the alternative message server is one of the parameters that they are sending to the new client. As long as the new client has been connected to the multicast group or to (a unicast connection) informs the message server for the connection (and it sends the port that it uses for the communication). Then the message server sends the contents of the virtual environment together with the last changes in the shared object of the environment. Afterwards the client sends to the message server data for the graphical representation of the user in the virtual environment (avatar). After this the TCP/IP connection is used only for reliability reasons and the rest of communication (sharing events, chat, etc.) is executed by the way that the client has chosen (UDP unicast or UDP multicast).

4.4 Achievement of consistency

In order to keep the VE consistent we should send both the shared events (events that concern shared objects) generated by a client to the others and the avatars' movement. For this reason we use triggers and state update messages.

For the realization of the transmission of events we exploit the capabilities of External Authoring Interface (EAI) [13] and VRML 97 [12] specifications. We use the *getEventIn()* method of the shared object node to send events created by a user in the VE, and updating the VEs of the other users (via the message server). Also we use the *getEventOut()* method of the shared object node, to read a shared event. Furthermore EAI provides a mechanism (the EventOut Observer and the callback() function) in order to be notified when an Event Out is generated from the scene. With this way we can send and receive triggers to activate scripts.

In order to serve the next participants of the VE, they should be notified for the last changes of the shared objects in the VE. We deal with this problem storing the last state of the shared objects in the message server and sending it to the new participants when they are connecting to the VE.

The avatar of a user can be added to the VE using both the *createVrmlFromURL* and createVrmlFromString functions of EAI. The avatar of each user is a VRML node that has a unique name plus the ID of the client that uses it. Therefore, in order to update the avatars' movement in the VE we use a proximity sensor in the VRML file to catch the avatar's position and orientation. Using the same functions of EAI as for the shared events we send the values of the position and orientation of the avatar plus the ID of the responsible client. These values are sending to the other clients (by the message server) in order for the other clients to update the position of the avatar, which corresponds to the specific ID in their local VE.

5. SUMMARY

In this paper we introduce the term educational virtual environment and we propose a suitable architecture and a protocol to support this environments. Furthermore we describe some advanced issues concerning the nature of this environments and the way that we deal with them. Currently we are in the implementation phase of our platform In order to realize this platform we use open standards and technologies such as VRML 97 for the VEs, Java for the implementation of the servers and the multi-user clients and H-Anim (Humanoid Animation) [14] for the construction of the avatars.

Our next steps after the implementation of the system are to integrate audio communication (using the RTP protocol and the JMF tools) and to implement a virtual community for testing and measurements.

6. REFERENCES

[1] Bouras C. and Tsiatsos T. "pLVE: Suitable Network Protocol Supporting Multi-User Virtual Environments in Education". *International Conference on Information and Communication Technologies for Education*, Vienna, Austria, December 6-9 2000

[2] Broll W. "DWTP - An Internet Protocol For Shared Virtual Environments". *Proceedings of International Symposium on the Virtual Reality Modelling Language 1998 (VRML'98)*, ACM, ACM SIGGRAPH, pages. 49/56.

[3] Broll W. "Bringing People Together - An Infrastructure for Shared Virtual Worlds on the Internet". *Proceedings of the IEEE WE-TICE '97 (Sixth International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises)*. Cambridge, Massachusetts, USA, IEEE Computer Society Press, Las Alamitos, June 18th - 20th 1997.

[4] Brutzman D., Zyda M. and Watsen K, Macedonia M. "virtual reality transfer protocol (vrtp) Design Rationale". *Workshops on Enabling Technology: Infrastructure for Collaborative Enterprises (WET ICE): Sharing a Distributed Virtual Reality*. Massachusetts Institute of Technology, Cambridge Massachusetts, June 18-20 1997, http://www.stl.nps.navy.mil/~brutzman/vrtp/vrtp_design.ps.

[5] Honda Y., Matsuda K. and Rekimoto J. and Lea, R. "Virtual society: extending the WWW to support a multi-user interactive shared 3D environment". *Proceedings of VRML'95*. San Diego, CA. Aug 1995. Also as SCSL-TR-95-035, http://www.csl.sony.co.jp/project/VS/VRML95.ps.Z.

[6] Locke J. "An Introduction to the Internet Networking Environment and SIMNET/DIS". http://www-nps-net.cs.nps.navy.mil/npsnet/publications/DISIntro.ps.Z.

[7] Macedonia M. R., Zyda M. J. and Pratt D. R. "Exploiting Reality with Multicast Groups: A Network Architecture for Large-Scale Virtual Environments". *Proceedings of the IEEE VRAIS'95*. IEEE Computer Society Press, Las Alamitos, CA, March 1995, pages 2-10.

[8] Normand V., Babski C., Benford S., Bullock A., Carion S., Farcet N., Frecon E., Harvey J., Kuijpers N., Magnenat-Thalmann N., Raupp-Musse S., Rodden T., Slater M., Smith G., Steed A., Thalmann D., Tromp J., Usoh M., Van Liempd G. and Kladias, N. "The COVEN project: exploring applicative, technical and usage dimensions of collaborative virtual environments". *Presence: teleoperators and virtual environments*, MIT Press, Vol.8, No2, 1999, pages218-236.

[9] Pandzic I. S., Joslin Ch. and Magnenat-Thalmann N. "Trends in a Collaborative Virtual Environment". *International Conference on Software, Telecommunications and Computer Networks-SoftCOM 2000*. Split, Rijeka, Dubrovnik (Croatia), Trieste, Venice (Italy), October 11-14 2000, pages 893-901.

[10] Singhal S. and Zyda M. *Networked Virtual Environments: Design and Implementation*. ISBN 0-201-32557-8, ACM Press, 1999.

[11] VRML Interchange Protocol - Specification, web page: http://www.csclub.uwaterloo.ca/~sfwhite/vnet/VIP.html.

[12] Web 3D Consortium. "The Virtual Reality Modeling Language (VRML) - Part 1: Functional specification and UTF-8 encoding". 1997, http://www.web3d.org/technicalinfo/specifications/vrml97/index.htm

[13] Web 3D Consortium. "The Virtual Reality Modeling Language (VRML) - Part 2: External authoring interface". 1999 http://www.vrml.org/WorkingGroups/vrml-eai/Specification/

[14] Web 3D Consortium - Humanoid Animation Working Group "H-Anim 1.1 specification". 1999, http://h-anim.org/spec1.1/