Architecture for a Web-based Distributed Virtual Reality System

Zhou Junlan, Yeo Chai Kiat, Lee Bu Sung

Abstract -- **In this paper we describe the architecture for a web-based Distributed Virtual Reality System (DVRS) based on the High Level Architecture RunTime Infrastructure, developed for Distributed Interactive Simulation. The DVRS constructs a virtual world, which exists in a distributed system environment. Each user resides at the local node of the virtual world. The DVRS maintains a consistent view for all users. A web-based viewer using the Virtual Reality Modular Language (VRML) plug-in is used to display the images.**

Keywords -- **Distributed Interactive Simulation, virtual reality environment**

I. Introduction

The astronomical growth of the Internet over the past decade has led to the development of a myriad of new applications. One prevalent application is the development of virtual environments [1], which allow groups of users to interact with one another over the Internet. The Internet Relay Chat is one such popular application. The next phase of development will be to have avatars to represent people and the user can interact with the virtual environment presented. There are already some existing systems [2], but they are centralized in nature and are not scaleable. A better solution is to support the virtual environment in a distributed manner. This approach is basically a primitive form of distributed interactive simulation.

The DoD in the USA has been working on the standardization of protocols for Distributed Interactive Simulation (DIS) over the past few years. One such protocol is the High Level Architecture (HLA) Runtime Infrastructure (RTI) [3-6], which provides a wide range of services to user applications.

This paper proposes an architecture to support a web-based distributed virtual reality system (DVRS) based on the RTI. Section 2 shall describe the proposed architecture while section 3 shall present the implementation of the DVRS. An application example based on the proposed architecture is shown in Section 4. Section 5 will conclude the paper and present some of the future extensions to the DVRS developed.

The authors are affiliated to School of Applied Science, Nanyang Technological University, Nanyang Avenue, S 639798, Singapore.

II. Architecture Of DVRS

Figure 1 shows the architecture of a DVRS. The DVRS consists of a number of components, namely, the end-users, the RTI and the registry server. The design is based on the very objective that every client is to be handled in a distributed manner.

Each user is created as a federate which will inter-operate with the other federates through the RTI. Every federate has a Local Server running to handle all the computing, such as coordinating the information flow among different processing elements, collision detection and message dispatches. User Interface is separated as a single thread to ensure the most effective response to the user input. VRML virtual world viewer displays the virtual world described in Virtual Reality Modeling Language (VRML)

The RTI determines the destination of the message and synchronization of the execution of all federates in the DVRS according to the information provided by the federates. The federates define the data they will require

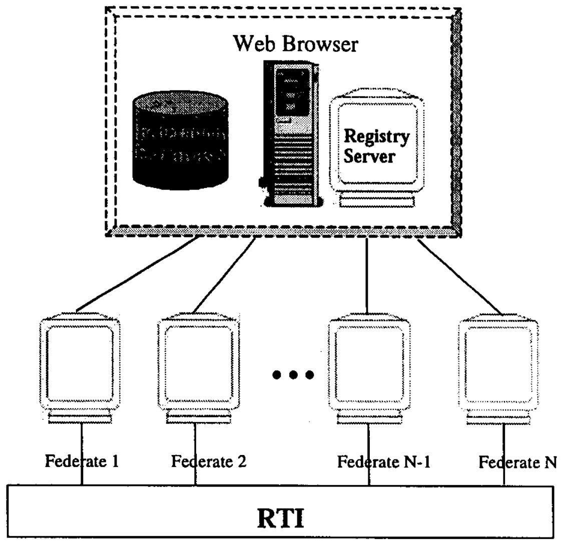

Figure 1 architecture of DVRS

and the information which they will provide during a federate execution.

The Registry Server is a web server which manages and maintains the list of users and the locations of all users in the Federation Database. Together, they keep track of the current states of all federates. Before anyone can join the DVRS, they must log into the Registry Server. It also maintains a current image of all the users and their locations in the virtual world, so that new users will be immediately updated with the latest status of existing users.

A. *The Federate*

Figure 2 shows the internal structure of a federate. Each federate of the DVRS has four processing elements: the Translator, the Local Server, the Virtual World Viewer and the User Interface.

1) *Translator*

The function of the Translator is to enable the system to communicate with the Runtime Infrastructure (RTI) module. The Translator resides between the RTI module and the Local Server. It establishes a communication channel between the RTI and the Local Server. It receives updated information of local objects in the virtual world from the Local Server, and encodes the information in a manner RTI can understand before forwarding it to the RTI. The latter will automatically multicast the updated information to other federates.

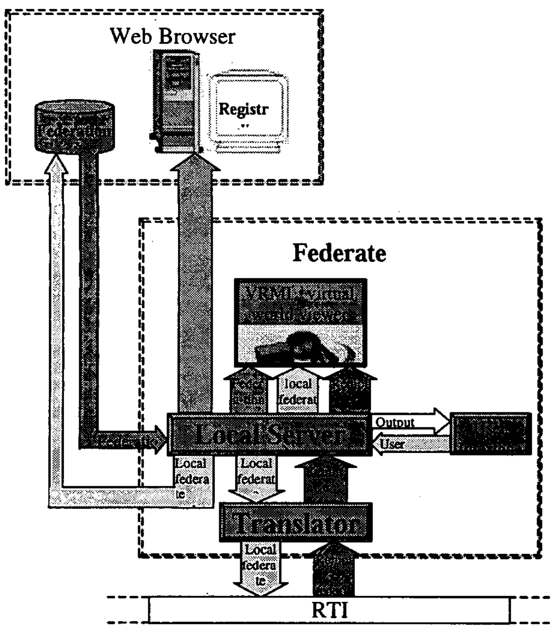

Figure 2. Internal architecture of federate

On the other hand, when receiving messages from the RTI, the Translator decodes them and forwards them to the Local Server. The communication between the Translator and the RTI module is established by function calls and callback functions. The communication between the Translator and the Local Server is established through creating a socket client which connects to the Local Server.

2) *Local Server*

The Local Server functions as a processing server of the federate. It is responsible for all the computation in the federate, such as:

- *Communicating with the Registry Server.* When a user first joins the federation, information on the federation will be requested from the Registry Server. Likewise, all updates of local objects in the virtual environment are being fed to the Registry Server.
- *Message processing and dispatching.* The Local Server handles the commands issued by the user at the local terminal and sends the new information, such as the new position of the local object, to other federates through the Translator and RTI. It also forwards the commands to the VRML Virtual World Viewer which will update the virtual environment accordingly. For example, when receiving user commands from the User Control Interface, such as "join" the virtual environment, or "move up" the local object, it will forward the commands both to the VRML Virtual World Viewer and through Translator to the RTI. The VRML Virtual World Viewer will update the scene graphics according to the new information. RTI will broadcast the information to the other federates which have to be kept updated of the changes in the virtual world.
- *Collision Detection.* When receiving user commands intended to change certain attributes of the objects in the virtual world, such as "moving up" an entity, it will first check the feasibility of the commands to prevent the possibility of a collision.

3) *VRML Virtual World Viewer*

It displays the scene graphics described in VRML and updates the attributes of the objects in the scene graph according to the commands from the Local Server through External Authoring Interface (EAI).

4) *User Interface*

It receives user input from the player, and forwards it to the Local Server for processing of the user commands.

B. RTI

The services provided by the RTI can be grouped into the following classes:

- Federation management offers basic services for the creation and operation of federation
- Declaration management offers support for efficient distribution of data
- Object management provides creation, deletion and identification of object, such as a federate
- Ownership management supports dynamic transfer of ownership of objects
- Time management provides synchronization of run time
- Data distribution management supports efficient routing of data among federates

The RTI is thus responsible for the synchronization of the execution of all federates in the virtual environment and determines the destination of the messages sent by the federates.

C. Registry Server

It resides on a web server, taking care of the Federation Database which stores the description of the current state of the virtual world. It receives the updated information from the federates, and modify the Federation Database accordingly. It is also responsible for providing federates with the federation description when requested.

The Federation Database stores the description of the virtual world constructed by the federation. It is controlled by Registry Server.

III. Implementation

In order to illustrate the feasibility of the proposed architecture for distributed virtual reality system, an implementation is carried out based on the architecture.

All clients and servers are PCs running Windows 95. The Run Time Infrastructure (RTI) version 3[7] module is implemented in C++, which provides services to federates in the form of C++ function calls. The Translator is implemented in C++, to interface to the HLA RTI while the virtual world is described in Java and VRML 2.0. The Local Server is a Java application while the Virtual World Image Viewer runs as a Java Applet. The web viewer is a plug-in called Cosmos Player [8].

The Virtual Reality Modeling Language (VRML) [9, 10,11,12] is the file format standard for 3D multimedia and shared virtual worlds on the Internet. Just as HyperText Markup Language (HTML) led to a population explosion on the Internet by implementing a graphical interface, VRML adds the next level of interaction, structured graphics, and extra dimensions (z and time) to the online experience. For communication between a VRML world and its external environment an interface between the two is needed. This interface is called an External Authoring Interface (EAI) and it defines the set of functions on the VRML browser that the external environment can perform to affect the VRML world. The object orientation, platform-independent, multithreading and other benefits of Java makes it the only practical application language for EAI right now.

The interface between the RTI and the virtual world which is performed by the Translator is accomplished as follows:

- Creates data structure in C++, which can be identified by RTI, to accommodate the objects in the virtual world described in Java and VRML

- Updates the corresponding C++ objects with information of local objects in the virtual world from the Local Server

- Utilizes the service provided by RTI to send interaction message input by Local Server

- Forwards interaction messages and updated information from other federates which come through the RTI, to the Local Server through the socket communication channel set up between them

IV. Application Example

With the implementation of the architecture, a simple web-based distributed virtual reality application is built on top of it. Figures 3a and 3b are the views of two federates each with two entities in the virtual world. Figures 4a and 4b show the results of the activity (moving the ball upwards and changing its color from blue to red) on an entity. The result shows that the activity on an entity will be updated with respected to the other related entities as well, such that all federates in the virtual world, each of which views the virtual world at a certain angle, maintain a consistent view of the virtual world.

V. Conclusion

In this paper, we propose an architecture to support a web-based distributed virtual reality system based on HLA RTI. The DVRS using the HLA RTI, ensures orderly delivery of packets to all users. A web-based VRML plug-in is used as the viewer for the virtual environment. A special server is used as the registry server to manage as well as to maintain the state of the virtual environment. The current system does not support real-time video or audio with the entity. We are presently, looking into the implementation of multimedia entity in the virtual environment.

REFERENCES

[1] Richard C. Waters, John W. Barrus, "The rise of shared virtual environments", IEEE Spectrum March'97

[2] Bernie Roehle, "Channeling the data flood", IEEE Spectrum March'97

[3] Richard M. Fujimoto, Richard M. Weatherly, Time Management in the DoD High Level Architecture, 1087-4097/96 IEEE 1996.

[4] Thomas S. Stark, Richard M. Weatherly, Annette Wilson, The High Level Architecture (HLA) Interface Specification And Applications Programmer's Interface.

[5] Defense Modeling and Simulation Office, Department of Defense High Level Architecture Interface Specification, Version 0.3, 25 January, 1996.

[6] Defense Modeling and Simulation Office, High Level Architecture Application Programmer's Interface, Version 0.1, 15 August, 1995.

[7] http://hla.dmso.mil/hla

[8] http://www.cosmosoftware.com/developer

[9] Chris Marrin, Proposal for a VRML 2.0 Informative Annex: External Authoring Interface Reference, January 21, 1997.

[10] Shel Kimen, VRML Is- Java Does- And the EAI Can Help

[11] Don Brutzman," The Virtual Reality Modeling Language and Java", Communications of the ACM June'98, Vol. 41 No. 6

[12] http://cosmosoftware/developer/moving-worlds