DISTRIBUTED VIRTUAL REALITY

Virtual Reality: A Distributed Perspective

Simon J. E. Taylor and Shirley Moody

Department of Computer Science and Information Systems
Brunel University
Uxbridge
Middlesex
UB8 3PH, U.K

Abstract: This paper considers the implications of Distributed Virtual Reality in terms of user interaction and networking. It is argued that the allocation strategy of data in such a system is one of the fundamental problems faced by designers of Distributed Virtual Reality applications. The need to use experience and techniques drawn from distributed databases is highlighted.

Keywords: distributed virtual reality, user interaction, networking, distributed databases.

1 Introduction

Two of the key themes in virtual reality (VR) today are user interaction and sensory sophistication. Many virtual reality applications are driven towards environments in which users can interact with other users in real time within a sensory domain mimicking the real world in as much detail as possible. As awareness of the huge range of stimulating possibilities that VR presents to end-users in industry, education and entertainment, greater and greater demands will be placed on this technology.

Developments in networking have made possible more and more *distributed* applications in which both users and data interact in an integrated but geographically disjoint computer system. As evidenced by research initiatives such as MOVIE (Furmanski, 1994), NPSNET (Zyda *et al.*, 1992) and examples found in (Zyda and Macedomia, 1995), Distributed VR is proving to be one of the most challenging of future distributed applications.

The purpose of this paper is therefore to highlight exactly why Distributed VR is so challenging. To this end we briefly review single- and multi-user VR architectures and introduce a general distributed VR architecture. We then indicate the heart of the distribution problem. The paper concludes with a recommendation as to how this problem might be addressed.

2 Distributed Virtual Reality

The architecture for a typical single-user VR application is shown in Figure 1. A user of this environment will "enter" the virtual world by linking up to appropriate VR interface devices (headsets, gloves, etc.). Exploration of the various possible locations and the various objects therein comes from an exchange of sensory data (visual, auditory, tactile, haptic, etc.) transmitted to and from the user via an interface to the virtual reality engine (VRE). The VRE interprets this data and, via reference to an Environment Database (ED), issues responses to the user. The responsibilities of creating the virtual world and the timely interaction with the data generated from the sensory interface is divided between the Environment Generator and the Real-time Operating System of the VRE.

In a single-user application, the sophistication of the virtual world fundamentally lies with the processing capability of the computer system on which the VR system is implemented and the response rate of the VR peripheral devices. Changing the mode of operation from single-user to multi-user adds a new factor as interaction between different users must be accurately controlled. The processing demands made by the Environment Generator will consequently increase as will the need for

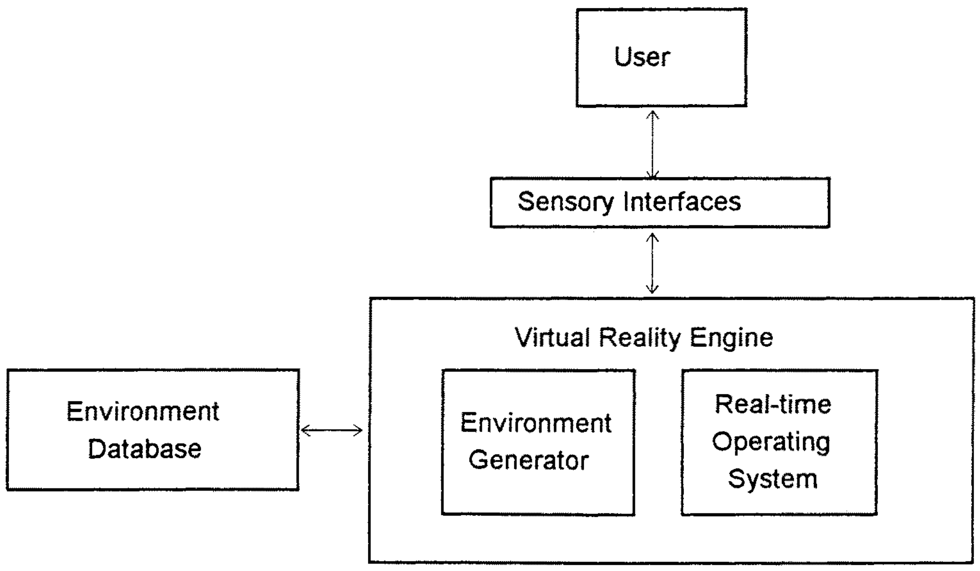

Figure 1 Architecture of a Single User Virtual Reality Application.

control exerted by the real-time operating system. The typical legacy of this move is a slow down in the response time of the VR system which can only be balanced by the purchase of faster hardware. For an excellent discourse on VR systems, we refer the reader to Vince (1995).

A major leap in complexity comes when a multi-user VR application is moved from a centralised scheme to a distributed one. Figure 2 shows such a system. Notionally, a Distributed VR (DVR) application is one in which geographically disparate users interact as if they were connected in a centralised multi-user VR application. The application in Figure 2 has sixteen users, eight of which are at one site and eight at another. At each site, the users are connected to one of two VR systems. Using examples of generally available technology, the VR systems at each site could be connected to each other via Ethernet, and to systems at another site via modem and the Internet (telephone lines).

Distributed systems are composed of separate hardware and software components which must interact. Software components (such as those comprising the VRE) are typically implemented as a collection of processes which interact by the coordinated transfer of data. Processes are hierarchically organised into a *client-server* relationship (typical of operating systems such as UNIX). In the example shown in Figure 2, the group of processes which would represent the VRE form a server to the client users. Another server process would exist to police access to the ED. The VRE server therefore is a client to the ED server. Server processes would also control access to local and remote communication facilities. This hierarchical approach to communication imposes a clear programming model based on *message passing* and has led to major innovations in portable distributed programming such as PVM and MPI[1].

Although this software organisation promotes an excellent programming and development model, it does not solve some of the more fundamental problems of DVR. Consider the following scenario. The sixteen users shown in Figure 2 are immersed in a virtual world. Each user can interact with each other and with the objects that exist with the world. Given that the position of each user must be held within some part of the ED, and that user interaction will be recognised by the VRE on the basis of changes to that database, what effect does this have on the placement of data within this distributed system?

If each ED contained all the data describing the virtual world in which the users interacted, the interaction of users of each individual VRE would be satisfied. However, to control the interaction of users of different individual VREs, some kind of mechanism would be required to automatically keep the data of each VRE *consistent*. For example, each if a user moved an object within the virtual world, the actions of that user would have to be communicated to *all* the VREs operating within the world so that their own EDs contained the new position of that object. A three-phase commit protocol would be enough to ensure the commitment of each and every action. Unfortunately, this would also promote delays to the VR system in terms of the communication overhead.

A possible alternative to this would be for each Environment Database to only hold timely information relating to the location of users local to the VRE of the database. Updates to the database would only therefore have to be performed when a user passed a defined "boundary" into the domain of another database. This

[1] A wealth of information concerning both PVM and MPI can be found via the Internet.

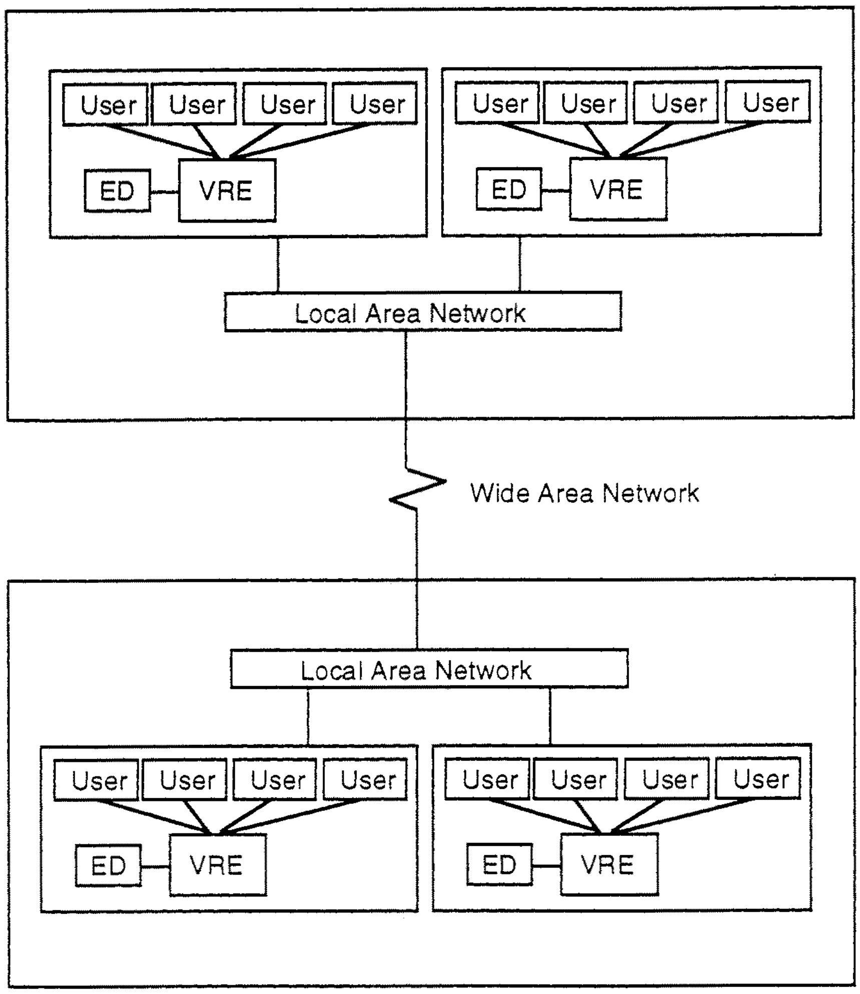

Figure 2 Architecture of a Distributed Virtual Reality Application

would, depending on the movement of the users, have the effect of decreasing the rate of communication but push up the amount of data that had to be transferred thus loading the communications network.

A wide spectrum of options are available to the developer of a DVR application by making combinations of the above two example policies. Choice depends on the application type and the demands of the user. Consideration of factors such as data transfer speeds (message latency, bandwidth, etc.) and processing times of each computer system on which the DVR application executes, also have to be taken into account due to the physical implications of a distributed system.

3 Conclusions

The problems faced by designers of DVR applications are very similar to those faced by designers of distributed databases (Bell and Grimson, 1992) and distributed systems (Coulouris and Dollimore, 1988) and are a synthesis of data access control, distributed transparency, fault tolerance and recovery control. However, a major difference is the real-time constraints imposed by the user of a DVR application.

Many analytical techniques exist to determine the optimum placement of data measured against user access profile and hardware processing constraints. Computer simulation is also used to analyse performance bottlenecks resulting from data access. Consideration of these techniques will inevitably lead to VR systems better capable of meeting user expectations and shorter development times through the elimination of performance barriers.

References

Bell, D. and Grimson, J. (1992). *Distributed Database Systems* (Addison-Wesley).

Coulouris, G.F. and Dollimore, J. (1988). *Distributed Systems: Concepts and Design* (Addison-Wesley).

Furmanski, W. (1994). MOVIE - Multitasking Object-oriented Visual Interactive Environment, in Fox, G.C., Messina, P. and Williams, R. (eds), *Parallel Computing Works* (Morgan and Kaufman).

Vince, J.A. (1995). *Virtual Reality Systems* (Wokingham: Addison-Wesley).

Zyda, M., Pratt, D., Monahan, J. and Wilson, K. (1992). NPSNET: Constructing a 3D Virtual World, *Computer Graphics,* ACM SIGGRAPH, March.

Zyda, M. and Macedomia, M. (eds) (1995). Special Issue. *Presence: Teleoperators and Virtual Environments* 4(2). MIT Press.