Implementing Scalable Networked Virtual Environments using Replicated VRML Servers

T. Rischbeck and G. Morgan
Department of Computing Science, Newcastle University,
Newcastle upon Tyne, NE1 7RU, England.

1. Introduction

The Virtual Reality Modeling language (VRML97) [1,2] is a file format for the description of audio-visual three-dimensional worlds. A VRML97 browser is capable of downloading such worlds from the Internet for local visualisation and exploration by the user. Contrary to previous versions of the language, world content may be dynamic and responsive to user interaction. Therefore, a VRML97 browser is responsible for evaluating and updating the world state.

Maximum world size in VRML97 is limited by download time, rendering costs, execution costs (for the dynamic world update) and memory requirements. Moreover, there is no support for multi-user shared worlds or for world state persistence. We have developed a VRML97 client-server architecture to overcome these restrictions. A VRML server maintains world state and performs updates. Multiple users (clients) may interact by informing the server of their actions (e.g., movement in the virtual environment) and receive updates to the shared world state from the server. The server is implemented in Java on top of the System Of Dynamic Active Objects (SODA) [3]. SODA adopts a programming model of communicating active objects, which may proceed in parallel and are dynamically load balanced over processing resources in a LAN. The server employs these features to compute world state in parallel. This provides scalability to many clients and support for large, complex, virtual worlds.

When only a single server exists, clients may suffer from low fidelity and a lack of realism due to the slow communication links that may exist between server and a geographically remote client. To overcome this problem, we provide client access to the shared world state via a group of replicated VRML servers. Clients are serviced by a "local" server, reducing the possibility of clients being separated from servers by large distances. Replica servers collaborate to present all clients with a single shared world state. Server collaboration is enabled via the Newtop object group service [4]. Newtop provides mechanisms that enable servers to maintain mutually consistent states, ensuring clients, irrespective of which local server they are connected to, receive mutually consistent views of the shared world state. Furthermore, in the presence of server failure, Newtop provides mechanisms allowing clients to migrate from a failed server to a correctly functioning server, increasing the availability of the shared world state.

Section 2 gives a brief outline of the client-server architecture and parallel execution of VRML97 using SODA active objects. Section 3 describes the system of replicated servers, describing how Newtop is used to ensure states of the replica servers remain mutually consistent. Section 4 briefly summarizes conclusions and directions for future study.

2. A Local Parallel VRML97 Server

2.1. VRML Execution Model

A VRML world is described as a collection of *VRML nodes*, which represent various real-world concepts and abstractions (e.g., geometric objects). World state is captured in the attributes of the nodes comprising a scene.

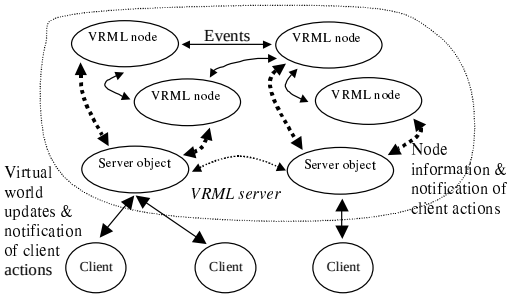

Figure 1 – Parallel client-server model

During runtime, node attributes may change to reflect updates to world state (e.g., a moving object may change its coordinates). VRML97 adopts an event-driven execution model to control these dynamic aspects. Sensor nodes serve as event generators when stimulated by user interaction or the passage of time. Events are then propagated along a *routing graph* to a set of nodes interested in a particular sensor event. In response to event reception, nodes may update their attributes as determined by the node's control logic. Additionally, a node may respond with the creation of secondary events. In this fashion, an initial sensor event might cause an *event cascade*, possibly involving a substantial subset of the nodes interconnected by the routing graph. Evaluation of complex event cascades requires powerful computational resources.

2.2. Parallel VRML97 Execution Using Active Objects

A conventional VRML97 browser will compensate lacking machine performance with reduced frame rates and therefore jerky visualisation. Our VRML97 implementation reaches scalability (i.e., high frame rates for complex and highly dynamic worlds) by exploiting parallelism between branches of the event cascade. This is compliant with the VRML97 execution model (see [3] for details).

Parallelism within the event cascade is expressed through a mapping of VRML nodes onto SODA active objects. Active objects proceed concurrently and the SODA runtime system may exploit parallelism by distributing active objects over processors available on a set of workstations on the LAN. Automatic object migration distributes load evenly over participating machines. Therefore, processing and maintenance of world state is achieved in parallel, utilising processing and memory resources of all participating machines.

2.3. Client-Server Model

We decouple world state evaluation from world state visualisation via a client-server architecture. Server objects may be allocated on any of the machines participating in the world update. They mediate the flow of information between clients and the VRML execution mechanism.

A client communicates with a server object to send client messages. There are two types of client messages: requests for world state information and notification of client actions that affect the world state. Notification messages are dispatched to the appropriate VRML sensor nodes, possibly triggering event cascade processing. To satisfy client requests for world state information, server objects collect information about updated world state from VRML nodes. Server objects perform culling of update information based on a client's position and viewing parameters before replying to a client request. In this way clients are only informed about a fraction of the world state, considerably reducing message size. By using powerful hardware on the server side, client machines with restricted processor and memory resources can therefore participate in highly complex virtual worlds. Multiple clients can connect at any time, sharing the same world state.

3 A Distributed Virtual Environment

Our distributed architecture consists of a number of interconnected VRML server replicas providing access to the shared world state for geographically dispersed clients (figure 2). Clients access their local VRML servers in the same manner as described previously and are unaware of server replication.

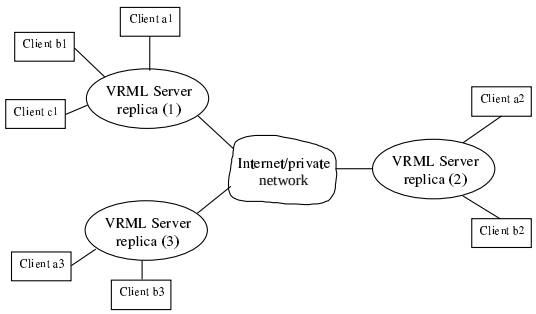

Figure 2 – Interconnected VRML server replicas

The VRML servers are actively replicated (in active replication all correctly functioning replicas perform processing). When a replica receives an action notification message from a client (i.e., a message that may result in a change to the shared world state) the message is multicast to the replica group. To ensure the states of the replicas remain mutually consistent, multicasts must be reliable and each replica must have a mutually consistent view in which events (such as receiving action notification messages) have taken place. In particular, to implement a successful active replication scheme we require the property that a given multicast be atomic: either all the functioning replicas are delivered the message or none; an additional property of interest is total order: all

functioning replicas are delivered messages in causality preserving total order. Newtop satisfies these requirements of a replica group.

As server objects mediate the flow of messages between VRML nodes and clients, they are well suited to the role of distributing client action notification messages throughout the replica group. A server object has an equivalent in each VRML97 server replica. Such objects form a group with each member multicasting action notification messages within this group.

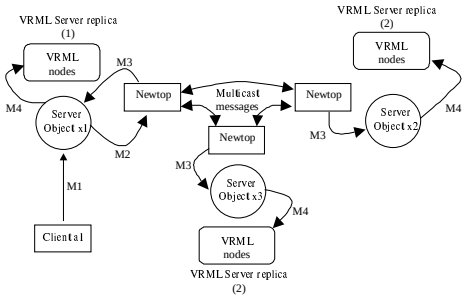

Figure 3 – Distributing client messages.

With the aid of the diagram in figure 3, we shall describe the message passing that takes place to ensure a client's action notification message is distributed to all replicas. M1 represents the initial message. Server object x1 forwards this message to the Newtop service for multicasting to the server object group (M2). Newtop multicasts this message and ensures that all server objects are delivered the message in the same order (M3). All server objects then deliver this message (M4) to the VRML nodes.

A server object satisfies request for world state messages locally. No communications with other replicas are involved. This is because the nature of the replication scheme ensures that all other replicas will generate identical messages independently of each other. This guarantees that all clients maintain a mutually consistent view of the shared environment.

Comparing our replication based solution to a single server solution, users may experience a greater delay between the sending of an action notification message and the resultant changes to world state such a message may have. However, the ability of local servers to directly communicate world state changes to clients provides an overall improvement in performance as the majority of world state changes originate from within a VRML replica. For example, a single user action may cause multiple state changes to occur over a period of time.

4. Conclusions and future work

We have described an architecture appropriate for use in large scale networked virtual environments. A parallel VRML97 server has been implemented using active objects distributed over workstations in a LAN. Exploiting the potential for computing the world state in parallel enables our system to satisfy the requirements of large numbers of clients and to model large and/or complex worlds. Furthermore, we have used active replication to ensure our system is suitable for geographically dispersed users.

Future work will concentrate on the reliability issues of our system. Newtop provides group membership services, allowing groups to be dynamic (membership may change during the lifetime of a group) and a failure suspector based on the causal relationships between messages and timeouts. With the aid of these services, the failure of a replica may be realised by the group of remaining, correctly functioning, replicas and removed from the replica group. By ensuring each replica maintains a list of all clients present in the shared world state, clients of a failed server may be migrated to a correctly functioning replica.

References

[1] Carey R, Bell G, The Annotated VRML 2.0 Reference Manual. Addison-Wesley, 1997.

[2] Carey, Rikk, Bell, Gavin, and Marrin, Chris. ISO/IEC 14772-1: 1997 Virtual Reality Modelling Language (VRML97). 1997.

[3] Rischbeck, Thomas and Watson, Paul. A Parallel VRML97 Server Based on Active Objects. VECPAR 2000 Proceedings. pp. 169-182. To be published in Lecture Notes in Computer Science, Springer Verlag

[4] G. Morgan, S.K. Shrivastava, P.D. Ezhilchelvan and M.C. Little, "Design and Implementation of a CORBA Fault-tolerant Object Group Service", Proceedings of the Second IFIP WG 6.1 International Working Conference on Distributed Applications and Interoperable Systems, DAIS'99, Helsinki, June 1999