ON REDUCING ENTITY STATE UPDATE PACKETS IN DISTRIBUTED INTERACTIVE SIMULATIONS USING A HYBRID MODEL

Declan Delaney*, Tomás Ward, Séamus McLoone
National University of Ireland, Maynooth, Co. Kildare, Ireland.
*decland@cs.may.ie

ABSTRACT

A key component in Distributed Interactive Simulations (DIS) is the number of data packets transmitted across the connected networks. To reduce the number of packets transmitted, DIS applications employ client-side predictive contracts. One widespread client-side predictive contract technique is dead reckoning. This paper proposes a hybrid predictive contract technique, which chooses either the deterministic dead reckoning model or a statistically based model. This results in a more accurate representation of the entity's movement and a consequent reduction in the number of packets that must be communicated to track that movement remotely. The paper describes the hybrid technique and presents results that illustrate the reduction in packet transmissions. This hybrid technique is compared to the standard dead reckoning method.

KEY WORDS
Distributed Interactive Simulations, Dead Reckoning, Hybrid Models, Latency

1. INTRODUCTION

Distributed Interactive Simulation (DIS) involves the participation of multiple participants communicating over a computer network [1]. The objective of the DIS application is to provide a realistic interactive experience to the participants. A typical example of such an application is a distributed computer game [2]. However, a number of technical problems combine to make delivery of such an experience difficult [3,4,5]. One such problem is latency, which is the time it takes for information to propagate across the network to all participants. Another closely related issue is the problem of network bandwidth. Within a DIS we refer to these problems as the information updating issue. Several methods have been devised to reduce the quantity of data that needs to be transmitted between participants [6,7,8]. The DIS standard defines a client predictive contract mechanism called dead reckoning [9].

This paper proposes a hybrid predictive contract technique, which dynamically switches between a short-term dead reckoning model and a longer-term statistical strategy model. This results in a reduction in the number of packets that must be communicated to track that movement remotely compared to a pure dead reckoning contract. The reduction is dependent on the model error threshold, as will be explained.

In section two of this paper we describe the information updating issue as it applies to distributed interactive applications. Existing solutions to this issue, including dead reckoning, are also outlined. Section three describes our proposed hybrid switching technique. A test environment was developed to compare the new technique with existing dead reckoning techniques. This environment is described in section four. Example results are presented in section five for both the hybrid and dead reckoning techniques. The paper ends with the conclusions and suggestions for future research.

2. THE INFORMATION UPDATING ISSUE

Network latency and bandwidth restrictions can combine to provide poor interactive experience in a distributed application. If an entity is any element that can be controlled then it will have a state that can change with time. To understand the mechanism involved in communicating entity state information to all participants in a distributed application, we will consider the typical case shown in Figure 1. Here we have a central server $\boldsymbol{S}$ maintaining the definitive state of a DIS. It communicates with two clients, $\boldsymbol{C_1}$ and $\boldsymbol{C_2}$, separated by network links with latencies $\boldsymbol{T_1}$ and $\boldsymbol{T_2}$ respectively. Figure 1 depicts a map of the virtual environment and the position of entities within that environment. The state of the DIS, which we will call $\boldsymbol{H}$, will be represented by a set of x-y coordinates for each entity in the simulation. For the scenario below we would have $\boldsymbol{H = \{(x_1, y_1), (x_2, y_2)\}}$. The definitive state of the DIS is that held by the server, $\boldsymbol{H_s}$. This state is updated regularly using entity state packets transmitted from the client set, $\boldsymbol{C=\{C_1, C_2\}}$, according to some underlying protocol. We will assume the impractical case that a new packet is transmitted from a client once per rendering frame.

In Figure 1 we assume that $\boldsymbol{T_1} >> \boldsymbol{T_2}$ and that $\boldsymbol{T_2}$ is negligible compared to the velocity of the entities. As a

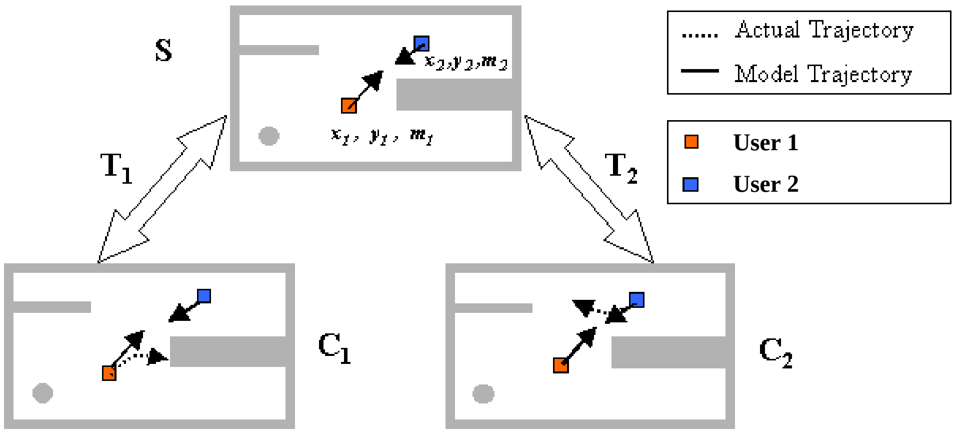

Figure 1: *State of DIS at instant* ***t=t***

result we can say that the state of the DIS as far as $\boldsymbol{C_2}$ is concerned, $\boldsymbol{H_2}$, is tightly coupled via a short latency link to the server such that

$$H_2 \approx H_s \qquad (1)$$

$\boldsymbol{C_1}$ on the other hand is connected via a high latency link and therefore $\boldsymbol{H_1}$ is tending to lag $\boldsymbol{H_s}$.

We can say,

$$H_1(t_n) = \{(x_1, y_1)_{t_n}, (x_2, y_2)_{t_n - T_1}\} \qquad (2)$$

The main consequence of this is that the user at $\boldsymbol{C_1}$ is reacting to an environment state $\boldsymbol{H}$ that is not that of the server $\boldsymbol{H_s}$. But events in the environment are determined globally and distributed by the server and its notion of the environment state $\boldsymbol{H_s}$. This tends to lead to disruption of the user interactivity for client $\boldsymbol{C_1}$, which we will describe as *localized interactivity distortion*.

We now define a measure of this distortion. A convenient one may be

$$\boldsymbol{D} = \sum_{i=1}^{N} (H_s - H_i)^2 \qquad (3)$$

where N is the number of clients participating in a DIS. We can call this a *global DIS distortion figure*.
Given the problems manifest in such a DIS how do we solve the problem such that

$$H_c \rightarrow H_s \qquad (4)$$

In other words, how do we maintain a consistent DIS state across all clients or at least dynamically track $\boldsymbol{H_s}$ as fast as possible for all participants with minimum distortion? Existing solutions are presented in the next section.

2.1 PREDICTIVE SOLUTIONS

The most common solution to the information updating issue involves a client side prediction contract mechanism called dead reckoning [9]: all participating clients agree to maintain the same low order local models of the dynamics of all other participating entities. This is the contract. Each participant also maintains a model of its own entity dynamics, which it continuously compares to its actual dynamics. When these differ by a pre-defined threshold, update information is broadcast to all other participants. These then update their models for that entity. Convergence algorithms are necessary to allow a natural transition to occur between the modeled and actual motion when update data arrives [9,10].

Alternative methods have also been explored, and these include:

- Relevance Filtering techniques: These seek to reduce the information being transmitted over the network by filtering the data based on criteria such as geographical proximity or rate of change [8].
- Network transmission protocol: Multicasting and reliable multicasting allow hosts to subscribe and unsubscribe to any of possibly several multicast groups. Multicast groups might be created based on entity type or geographical location in the virtual environment [12].
- Packet bundling: This involves combining a number of data packets to create a larger data packet because network devices can only process a limited number of packets per unit time [8].
- Data Compression: These techniques allow the reduction in the size of the information packet being transmitted. One technique is to encode differences

between successive data packets instead of transmitting the absolute state [8].

- Time Management: This involves pre-empting events and then locking up the system so that the event can occur. Alternatively, the execution of local user input is delayed and disguised as something else until the local user input can be relayed to all participants [13,14]
- Priority Scheduling: A transmission priority can be assigned to information based on criteria such as speed of movement or rate of error change [15]. This also includes Quality of Service protocols [5].
- Visibility Culling: The environment is divided into cells and multicasting updates are provided to all entities that are visible to each other in each cell [3].

The above techniques are based on network management and network partitioning policies. We are going to look at a packet-reduction method based on client behavioral modeling, where we switch between a short-term dead reckoning model and a long-term statistical-based strategy model.

3. THE HYBRID TECHNIQUE

3.1 TERMINOLOGY

A *goal* is the aim or objective a person has in moving through any environment. For example, the goal might be to go from point A to point B. Goals can be classified as either *static* or *dynamic*. Static goals are stationary in time and space whereas dynamic goals develop over time. In achieving a goal a person can adopt a number of strategies, so that any one strategy is an expression of the goal. Strategies can be either steady state, transient or alternative. Steady state strategies are obvious strategies that can be modeled on past data and arise out of user familiarity with the DIS. Transient strategies relate to new users in the environment who behave in a somewhat erratic way because they are unfamiliar with the environment. Alternative strategies are strategies that lead to the goal but are neither transient or steady state strategies. The idea underlying the hunt for strategies is to train a system to expect certain strategies based on past user behavior or based on expected user behavior. Each strategy comprises one or more trajectories – a set of trajectories can be identified with any strategy. A *trajectory* is an instance of entity motion in achieving a goal. As with strategies, trajectories can be steady state or transient. Multiplicity can refer to either strategies or goals. *Strategy Multiplicity Index* (SMI) refers to the number of goals a strategy leads to. *Goal Multiplicity Index* (GMI) refers to the number of steady-state strategies that lead to the goal. This terminology is illustrated in Figure 2. In this paper we present results for a static GMI of 1.

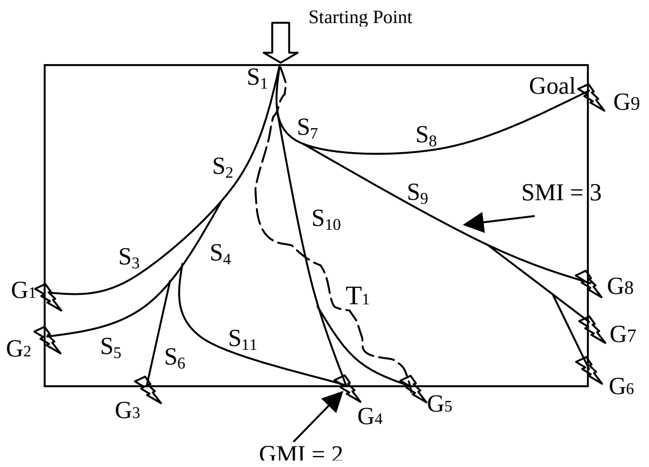

Figure 2: *Terminology – S1 to S11 are strategies;G1 to G9 are goals; T1 is a sample trajectory; GMI is the Goal Multiplicity Index – how many strategies reach that goal; SMI = Strategy Multiplicity Index – how many goals this strategy lead to.*

3.2 THE HYBRID MODEL

The hybrid model ***M*** is of the following form;

$$M = p\chi + (1-p)\Gamma \qquad (5)$$

where χ is any conventional dead reckoning model, Γ is a long term model of entity strategy and p is a binary weighting factor governed by:

$$\begin{aligned} p &= 1 \text{ for } \|M-\Gamma\| \geq \theta \\ &= 0 \quad \text{otherwise} \end{aligned} \qquad (6)$$

where θ represents a distance measure threshold between the modeled behavior and the long term model.

The model given by ***M*** is used by participating clients in a DIS. The parameters and initial entity state used by the model are updated every time the state deviates from the true state by a predefined threshold amount T_m.

While alternative soft blending techniques could be employed here, we have opted for a simple switching technique to illustrate the principles involved.

4 DEVELOPING THE STRATEGY MODEL

4.1 THE TEST ENVIRONMENT

The long-term strategy model employed in the hybrid prediction technique can be constructed in various ways: (1) by recording past actual entity movements in the environment, (2) by heuristically identifying possible strategies based on the examination of the environment and (3) by employing automatic path-finding techniques.

For this paper we developed a Java, game-type application that recorded user trajectories in a controlled two-dimensional environment – see figure 3.

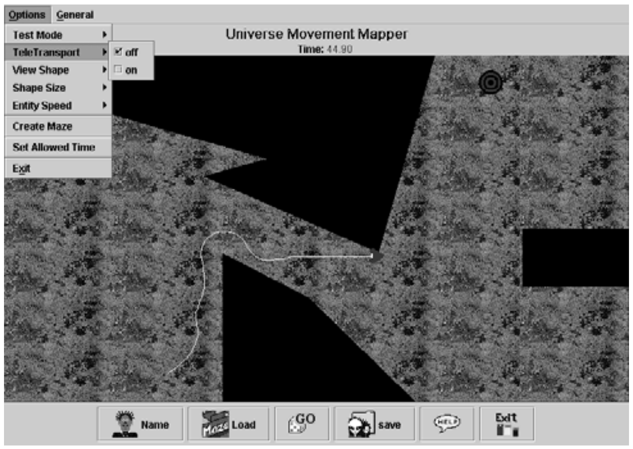

***Figure 3**: A screen shot of the trajectory recording software. The target is shown as a circle to the top right. The black areas are obstacles that do not allow users to pass. The white trail represents the past motion.*

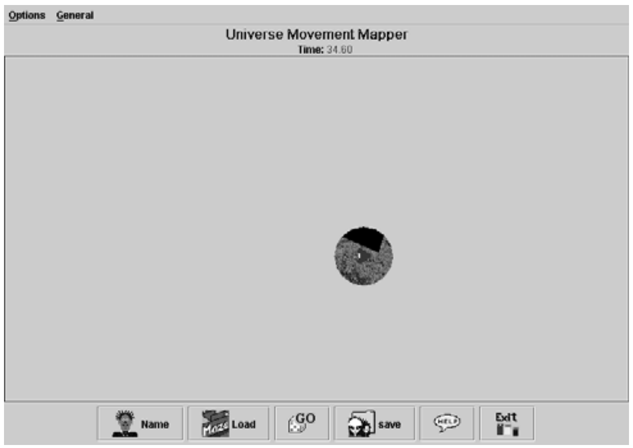

***Figure 4**: A screen shot of the trajectory recording software showing the user-restricted view.*

Users were asked to navigate from a fixed starting position to a fixed target position in as short a time as possible. Their view of any obstacles in their path was restricted to a circular area around their immediate position. This is illustrated in Figure 4. The user began with no knowledge of the location of the target and repeated the exercise until they had produced the quickest time possible. Each attempt constituted a trajectory. Data was collected and stored for each trajectory. This provided the basis of the statistical-based strategy model that we will use in the hybrid contract technique.

4.2 THE STRATEGY MODEL

Using the software application, a minimum of five and maximum of fourteen trajectories were recorded from fourteen different users. The final trajectory for ten of the fourteen users is plotted in Figure 5.

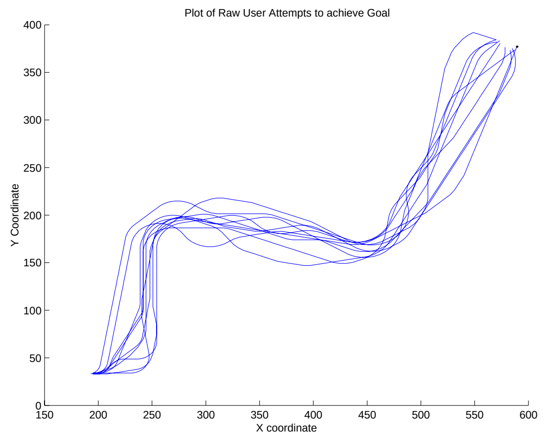

***Figure 5**: A plot of the last user trajectory for 10 users. The trajectory indicated in bold is the strategy chosen to be the strategy model.*

Observation of this plot shows that the trajectories of the users converge to a recognizable steady-state strategy. This observation underlies the motivation behind the hybrid contract approach. In this paper we will select and use one of these trajectories as representative of the steady-state strategy. In future work, we intend to obtain a better strategy model based on all ten data sets. The results obtained are described in the following section.

5. RESULTS

The strategy model was obtained as described in the previous section. Of the remaining four data sets, two were chosen and analyzed to compare the hybrid and first order dead reckoning contract techniques. The same threshold value was set for both. Table 1 shows the number of packets sent for all trials for both user datasets and for both methods.

	User 1		**User 2**	
Trial no.	**Dead Reckoning**	**Hybrid**	**Dead Reckoning**	**Hybrid**
1	19	13	31	32
2	19	17	26	27
3	28	25	18	15
4	20	19	8	3
5	14	12	10	7
6	17	13	14	4
7	9	6	13	13
8	14	12	8	4

***Table 1**: The number of packets transmitted for two user sets for both pure dead reckoning and the hybrid method. Trial 1 is the initial trial. Threshold value: 25.*

A selected number of trials for user 2 are shown in Figures 6a to 6c. Each plot shows the model strategy (dotted line), the user trajectory (continuous line), the packets transmitted (asterisks) and the trajectory as reconstructed by the remote client.

Figure 6a plots the initial user trajectory. The user wanders around the environment seeking the target, which is located at the top right end of the strategy model curve. The strategy model is employed on only one occasion, where it overlaps the trajectory. Most packets are therefore the result of the threshold value between the trajectory and the dead reckoning model being exceeded. It was expected that the strategy model would have almost no relevance since the user had never seen the environment before. The reconstructed trajectory at the remote client is jagged because a first order dead reckoning algorithm is used. Thirty-two packets are transmitted.

In Figure 6b trajectory five of user 2 is plotted. In this case the user has had four previous attempts and is more familiar with the environment. The trajectory and the strategy model are in agreement for all but two sections of the trajectory. In these two sections dead reckoning is employed. The remote client uses the strategy model for reconstruction purposes except for these two sections. Only seven packets are transmitted. If pure dead reckoning were used, ten packets would be required.

Figure 6c shows user 2's final trajectory. The user trajectory meanders about the model strategy, but remains within the threshold distance from the strategy for all but one section of the trajectory. The reconstructed trajectory is therefore superimposed on the strategy curve except for this section. Only four packets were transmitted. If pure dead reckoning were used, eight packets would be required.

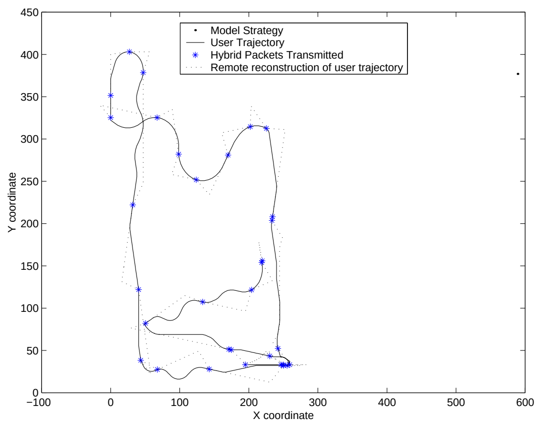

Figure 6a*: The first user 2 trajectory attempt. The user wanders, attempting to locate the target (to the top right). The rugged remote reconstruction results from using a first order dead reckoning algorithm. The strategy model is used on one occasion.*

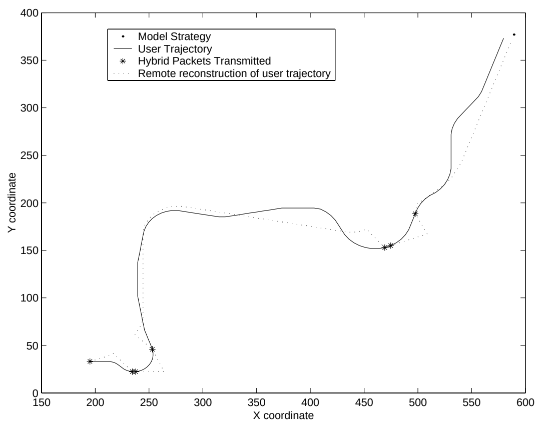

Figure 6b*: This is the fifth user 2 attempt. The user trajectory reaches the target. Where the remote reconstruction (dashed line) appears to disappear, it is in fact superimposed on the model strategy. Dead reckoning is employed on two occasions only.*

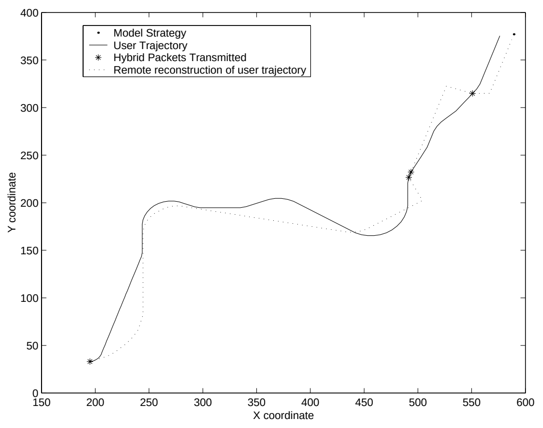

Figure 6c*: This is the eighth trajectory for user 2. Dead reckoning is only employed on one occasion. For most of its length, the trajectory is remotely represented as the strategy.*

It should be noted the performance of the hybrid technique is based on a relatively high threshold error value. Reducing this value results in a poorer performance from the hybrid model, although it is never any worse than the pure dead reckoning model. It is conceivable that employing a multiple-model strategy, where the number of models is related to threshold, would significantly improve modeling performance.

6. Conclusions and Future Work

In this paper we have described a technique called hybrid switching that reduces the number of packets that need to be transmitted to maintain state fidelity across a simple distributed application.

Using a test environment developed in Java we compared the dead reckoning technique with a hybrid switching technique and we illustrated that the hybrid technique required fewer packets compared to dead reckoning for remote trajectory reconstruction. However, as previously noted, this was based on a high threshold value.

Future work will involve looking at improving performance for lower values of threshold through the use of multiple model strategies and multiple model blending techniques.

ACKNOWLEDGEMENT
This work was funded by Enterprise Ireland Basic Research Grant SC/2002/129/.

REFERENCES

[1] S. K. Singhal and M. Zyda, *Networked Virtual Environments* (S. Spencer ACM Press, 1999).

[2] D. Kushner, The Wizardry of ID, *IEEE Spectrum* (39) 8 August 2002, 42-47.

[3] C. Faisstnauer, D. Schmalstieg, and W. Purgathofer, Scheduling for very large Virtual Environments and Networked Games using Visibility and priorities, *Proceedings of the 4th IEEE International Workshop on Distributed Simulation and Real-Time Applications,* San Francisco, California 24 - 26 August 2000, 31-38.

[4] L.A.H. Liang, Wentong Cai, u-Sung Lee and S. J. Turner, Performance Analysis of Packet Bundling Techniques in DIS, *Proceedings of the 3rd IEEE International Workshop on Distributed Interactive Simulation and Real-Time Applications*, College Park, Maryland 23 - 24 October 1999, 75-82.

[5] A.S. Tannenbaum, *Computer Networks Third Edition*, (Prentice-Hall, 1996).

[6] Wentong Cai, F.B.S. Lee, and L. Chen, An Auto-adaptive Dead Reckoning Algorithm for Distributed Interactive Simulation, *Proceedings of the 13th Workshop on Parallel and Distributed Simulation*, Atlanta, Georgia 1 - 4 May 1999, 82-89.

[7] G.J. Valentino, S.T. Thompson, T. Kniola and C.J. Carlisle, An SMP-based, Low-latency, Network Interface Unit and Latency measurement System: the SNAPpy system, *Proceedings of the 2nd IEEE International Workshop on Distributed Interactive Simulation and Real-Time Applications*, Montreal, Canada, 19 - 20 July 1998, 62-70.

[8] M.A. Bassiouni, Chiu Ming-Hsing, M. Loper, M. Garnsey and J. Williams, Performance and reliability analysis of relevance filtering for scalable distributed interactive simulation, *ACM Transactions on modeling and computer simulation*, (7)3, July 1997, 293-331.

[9] IEEE Standard for Distributed Interactive Simulation - Application Protocols IEEE Std 1278.1-1995.

[10] C. Durbach and J.M. Fourneau, Performance evaluation of a dead reckoning mechanism, *Proceedings of the 2nd IEEE International Workshop on Distributed Interactive Simulation and RealTime Applications,* Atlanta, Georgia, 1 - 4 May 1999, 23-29.

[11] T.K. Capin, J. Emeraldo and D. Thalmann, A dead reckoning technique for streaming virtual human animation, *IEEE Transactions on Circuits and Systems for Video Technology,* (9)3 April 1999, 411-414.

[12] J. M. Pullen, Reliable Multicast network Transport for Distributed Virtual Simulation, *Proceedings of the 3rd IEEE International Workshop on Distributed Interactive Simulation and Real-Time Applications,* College Park, Maryland, 23 - 24 October 1999, 59-66.

[13] B. G. Worthington and D.J. Roberts, Encapsulating Network Latency Compensators in VRML, *Proceedings of VWSIM (Virtual Worlds and Simulation Conference) Int. Conf. SCS Western Multi-conference,* San Diego, USA, January 2000.

[14] D.J. Roberts and P. M. Sharkey, Maximising Concurrency and Scalability in a Consistent , Causal, Distributed Virtual Reality System, whilst minimising the effect of Network Delays, *Proceedings of the 6th IEEE Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises*, Cambridge, MA , June 1997, 161-166.

[15] C. Faisstnauer, D. Schmalstieg, and W. Purgathofer, Priority Round-Robin Scheduling for very large Virtual Environments, *Proceedings of the IEEE Virtual Reality 2000 Conference,* New Brunswick, New Jersey, 18 - 22 March 2000, 135-142.