An Adaptive Control Mechanism for Management of Networked Information Consistency

Mintang Lin
Department of Management Information Systems
Chin Min College
Miaoli, Taiwan

E-mail:
mtlin@cse.nsysu.edu.tw

Chungnan Lee
Institute of Computer and Information Engineering
National Sun Yat-Sen University
Kaohsiung, Taiwan

E-mail:
cnlee@mail.nsysu.edu.tw

Tainchi Lu
Institute of Computer and Information Engineering
National Sun Yat-Sen University
Kaohsiung, Taiwan

E-mail:
tclu@cse.nsysu.edu.tw

Hung-Che Shen
Institute of Computer and Information Engineering
National Sun Yat-Sen University
Kaohsiung, Taiwan

E-mail:
hcshen@cse.nsysu.edu.tw

Abstract

To automatically maintain appropriate level of interested information for a large number of participants in networked virtual environments is of paramount importance. In this paper we propose an adaptive control mechanism to manage the information consistency and control the dynamic shared state for a large-scale virtual environment. The approach uses three major techniques for dynamic shared state maintenance - the shared repository technique, blind broadcasting method, and dead reckoning model to throttle the data transmission over the network. Depending upon the system load, the approach can automatically switch from one level to another level to maintain maximum performance in server. Performance evaluation on the proposed adaptive control mechanism, the three-level consistency approach, and the shared repository technique were conducted. Performance evaluation results show that both the proposed mechanism and the three-level consistency approach can maintain low system load over a wide range of participants, but the adaptive control mechanism can adapt to the state of server load and provide a better service quality.

Keywords: *Dynamic shared state; Networked virtual environment; Information consistency; Blind broadcasting; Distributed interactive simulation; Dead reckoning model (DRM)*

1 Introduction

The primary consideration for a flexible networked virtual environment (NVE) [3,5] is to adapt to a variety of participants using an appropriate strategy and control the consistency of the information. However, the common NVE systems encounter several problems as the number of participants or clients grows, such as managing information consistency among servers and clients, guaranteeing real-time interactivity [13], and contending with limited network bandwidth. Therefore, many research agencies devote their energies to the design of workable protocols, such as the Distributed Interactive Simulation (DIS) [9], the High Level Architecture (HLA) [6], and the VR-protocol [7], to maintain the information consistency. Some typical systems are the Shastra system [1], the VEOS system [2,8], and NPSNET [18], and so on. The Shastra system developed at Purdue University is a collaborative environment using a shared repository to provide the absolute consistency. The VEOS system developed at the University of Washington uses the blind broadcasting method to reduce message overheads. The NPSNET research group has developed a large-scale virtual environment (LSVE) based on DIS. The NPSNET uses the dead reckoning model to reduce network traffics. The goal of the LSVE is to study and develop the technologies for generating real-time and interactive networked virtual worlds for military simulations. They focus on the complete breadth of human-computer interaction and software technology for implementing the LSVE. Basically the systems mentioned above use only one type of dynamic shared state techniques to send messages.

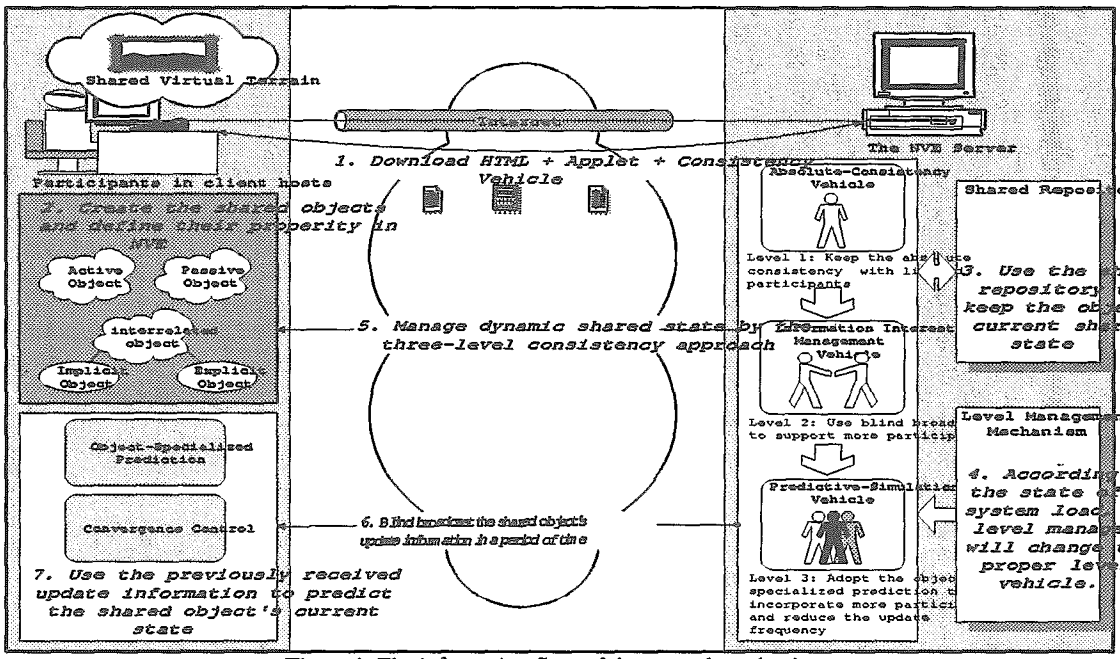
Generally speaking, participants may not join an NEV through the network simultaneously. When there are only a few participants or a light load in the NVE, high overheads and network bandwidth limitations will not be the major problems for NVE. Hence, it is possible for an NVE server to provide absolute consistency and to maintain reasonable network latency at the same time. Absolute consistency has the advantage of providing an identical view of the project to all client hosts. When the server load becomes heavier in the networked virtual environment, it becomes impractical to maintain absolute information consistency. Under the circumstances, some absolute consistency can be sacrificed in order to maintain high interaction. Hence, we use the blind broadcasting method to update those less important shared objects and adopt the broadcasting approach to update those more important shared objects in the NVE. In order to further reduce the frequency for transmitting updates in the entire NVE, we may consider the DIS dead reckoning model (SIMNET [12] was the spawning of a standard networking protocol, known as DIS), which is used to predict the true state of virtual space in client hosts. Conceptually, this model is suitable for a large-scale NVE with a large number of participants. Actually, it is not easy for an NVE to attract so many simultaneous participants all day long. In addition, the dead reckoning model does not guarantee that all client hosts will share the identical state. It requires participants to tolerate and accept potential discrepancies. So it is inappropriate to adopt dead reckoning model for the NVE when there is only a few participants or a light server load.

Based on the considerations mentioned above, we had proposed the three-level consistency approach to maintain the information consistency of an NVE in previous research [11]. This approach adopted a predefined threshold to enable switching from one level to another level based on the maximum participant's number derived from the approximate Poisson discrete distribution. The main consideration for switching mechanism in the three-level consistency approach is the number of participants. When the number of participants that joined the networked virtual environment exceeds the predefined range, the NVE server switches from one level to another. The advantage of the approach is simple to switch and implement for the NVE

developers. Unfortunately, the state of networked virtual environment such as the load of an NVE server cannot be always judged from the amount of participants. Although the number of participants may influence on the load of the NVE server, it is not the only factor to determine the load of the NVE server. For example, few participants with frequently activity may cause a heavy load of the NVE server or network traffic; on the contrary, many participants with few activities may not always cause a heavy load. Since the amount of participants in the NVE cannot always guarantee the right timing to switch to a proper level. In this paper, we use the load of the NVE server to enable the switching control in the control mechanism.
The remainder of the paper is organized as follows. Overview of information consistency techniques is presented in Section 2. In Section 3, we describe the proposed adaptive control mechanism. Section 4 presents performance evaluation, simulation design, and results. In the last section some conclusions are given.

2 Overview of Information Consistency Techniques

In this section, we first describe the information flow of the proposed NVE infrastructure using the control mechanism. As shown in Figure 1, participants can connect to the NVE server via the Internet and download the html, applets, and consistency vehicle (i.e. an applet that is responsible for sending the information updates to the server) from the server. Based on the control mechanism within the NVE server, the proposed approach can dynamically switch to an appropriate level to maintain the information consistency. It is an adaptive and flexible mechanism to meet the requirements for different levels of information consistency. Thereafter, we adopted dynamic shared state techniques to raise the consistency-throughput tradeoff issues, and propose the adaptive mechanism to dynamically maintain the state of the shared objects in a networked virtual environment. Each level of the fundamental concept will be briefly reviewed. The performance characterization of each dynamic shared state technique will be given in Section 3.



Shared Virtual Terrain
Internet
Participants in client hosts
1. Download HTML + Applet + Consistency Vehicle
2. Create the shared objects and define their property in NVE
Active Object
Passive Object
interrelated object
Implicit Object
Explicit Object
5. Manage dynamic shared state by three-level consistency approach
Object-Specialized Prediction
Convergence Control
6. Blind broadcast the shared object's update information in a period of time
7. Use the previously received update information to predict the shared object's current state
The NVE Server
Absolute-Consistency Vehicle
Level 1: Keep the absolute consistency with limited participants
Information Interest Management Vehicle
Level 2: Use blind broadcast to support more participants
Predictive-Simulation Vehicle
Level 3: Adopt the object-specialized prediction to incorporate more participants and reduce the update frequency
Shared Repository
3. Use the shared repository to keep the objects' current shared state
Level Management Mechanism
4. According to the state of the system load, the level manager will change to a proper level vehicle.

Figure 1. The information flow of the control mechanism.

2.1 The First Level: Absolute Consistency

In the first level, the major assessment is to achieve absolute consistency, including accurate dynamic shared state, identical view, low network latency, and high reliability. Particularly, the accurate dynamic shared state is fundamental to creating realistic environments. In order to provide accurate information and absolute consistency throughout the network, each client host must send update information to the server when the shared objects are modified; then the server notifies other clients immediately. In the implementation, we used Java language to build a shared repository in the server and spawn the Java server thread for each participating client to listen the client's requests. The shared repository contains the current value for the shared state used in the NVE and stores the NVE's current state in a centralized database. Consequently, the shared repository guarantees that each participant can see the updates in the same order and maintain absolute consistency among the participating client hosts. A typical example of using a shared repository to provide absolute consistency is the Shastra system, which was developed at Purdue University and uses a central server in the collaborative environment.

2.2 The Second Level: Information Interest Management

When the load of an NVE server is heavy and network latency is high, the NVE encounters a nasty problem, because the NVE can not afford the high communications overhead to support absolute consistency and identical view at all client hosts. In such a situation, this is the time to switch from the first level to the second level to maintain the dynamic shared state. In the second level, we use the blind broadcasting approach [4] to send and receive the update information through the network for those less important objects. The blind broadcasting approach has been used to support many critical NVE applications. A typical example of using blind broadcasting can be found in the VEOS system [2,8], that was developed at the University of Washington and uses an "epidemic" approach to reduce the broadcasting message overheads.

2.3The Third Level: Predictive Simulation

IEEE Standard 1278.1 [8] defines the protocol data units (PDUs) of Distributed Interactive Simulation (DIS). The DIS protocol uses 27 different kinds of messages (i.e. PDUs) to exchange information among simulation nodes. The DIS uses a predictive algorithm, known as the dead reckoning algorithm. It transmits state update packets less frequently and approximates the true shared state based on previously received update information. A dead reckoning model consists of two major components, a prediction technique and a convergence technique. The prediction technique is used to estimate the future value of the shared state by referring to the past and current update information. As for the convergence technique, it is used to correct the predicted state without incurring visual distortion. Generally speaking, a second-order polynomial prediction is the most popular prediction technique for the dead reckoning model. For example, each client host predicts the next position of the shared objects by using the equation, $x' = x_0 + v_0 * t + \frac{1}{2} a t^2$, where x' is the next position, x_0 is the initial position, v_0 is the velocity, and a is the acceleration. We can obtain the initial position x_0, velocity v_0, and acceleration a from the pervious update information of this shared object.

3 The Proposed Adaptive Control Mechanism

In this section, we describe the proposed mechanism to automatically switch from one level to another level depending on the load of an NVE server. It maintains the information consistency and allows maximum participants to join the networked virtual environment simultaneously. The main functions of this mechanism are to analyze and estimate the load of NVE server in order to determine the timing to switch the level. The load of the NVE server can be characterized by the process service time [14], the process waiting time in a queue, and the length of the queue [15]. In other words, we use a queuing model to serve as the analysis and estimation model. We define the switching criteria for three levels based on the load of the NVE server. In the following, we describe the analysis and estimation modeling confirming the NVE server in Subsection 3.1 and present the adaptive control mechanism in Subsection 3.2.

3.1 Analysis and Estimation of an NVE Server Load

We use a centralized system consisting of one CPU with shared memory as an NVE server [16,7]. Client hosts communicate with the server using massege passing through the network. A server is modeled as a queuing system, such as M/M/1, M/D/1, and so forth. We denote that λ is the mean arrival rate, μ is the mean service rate, traffic intensity ρ is the ratio of λ/μ, L_q is the expected queue length (excluding the current executed task) and W_s is the waiting time (including the service time) in the NVE server. Once both λ and μ are given, the values of L_q and W_s can be calculated from Little's formula [10]. Let $L_q(t)$ be the mean queue size and $W_s(t)$ be the mean waiting time, which are estimated from the system load at time t. In order to quantify these notations, we designate two values L and H, where $0 \le L < H$. Both L and H are arbitrary values, which refer to the ratio of $\frac{L_q(t)}{L_q}$ and $\frac{W_s(t)}{W_s}$, and they are the system parameters to be set by the system designer based upon the capacity of an NVE server.

Definition of the Range for Three Levels

Currently, we use three types of dynamics shared state techniques that are used to support different information consistency. We use L and H to divide the range and define the relationships among these three levels.

Definition 1: The first level is "Absolute Consistency," denoted as AC, if $\frac{L_q(t)}{L_q} \le L$ and $\frac{W_s(t)}{W_s} \le L$.

Definition 2: The second level is "Information Interest Management," denoted as IIM, if $L < \frac{L_q(t)}{L_q} \le H$ and $L < \frac{W_s(t)}{W_s} \le H$.

Definition 3: The third level is "Predictive Simulation," denoted as PS, if $H < \frac{L_q(t)}{L_q}$ and $H < \frac{W_s(t)}{W_s}$.

The basic concept for the three definitions is to measure the current server's mean queue size $L_q(t)$ and mean waiting time $W_s(t)$ and then compare them with the predefined L_q and W_s that are calculated from the system parameters λ and μ.

When the load of the NVE server is light, the absolute consistency is used to provide accurate dynamic shared state. Under the circumstances, the mean queue length and the mean waiting time should be short in the NVE server at time t. In other words, the values of $\frac{L_q(t)}{L_q}$ and $\frac{W_s(t)}{W_s}$ are much lower than L in the first level. In contrast to the first level, we adopt the predictive simulation approach in the networked virtual environment, when the load of an NVE server is much heavy. In other words, the values of $\frac{L_q(t)}{L_q}$ and $\frac{W_s(t)}{W_s}$ are much higher than H in the third level. In general, the load in the second level is between the levels of absolute consistency and of the predictive simulation.

Definition of the Server Load Function

In addition to the predefined range of three levels, we need to determine the values of threshold for different cases in the adaptive control mechanism. Because when we determine the level switching based on Definitions 1, 2, and 3, some conditions may be ambiguous. For example, there are ($\frac{L_q(t)}{L_q} > L$ and $\frac{W_s(t)}{W_s} \le L$) or ($\frac{L_q(t)}{L_q} \le L$ and $\frac{W_s(t)}{W_s} > L$) in the first level. Hence, we further define a server load function LOAD($\cdot$) that is a function of the waiting time, queue length, L_q and W_s as follows:

$$LOAD(L_q(t), W_s(t), m) = \frac{\frac{L_q(t)}{L_q}}{m} + \frac{\frac{W_s(t)}{W_s}}{m}, \text{ where } m = H \text{ or } L \quad \text{..........(1)}$$

Furthermore we define two thresholds for LOAD($\cdot$) as X_m, where m=L or H. When $LOAD(L_q(t), W_s(t), m) \le X_m$, it means that the mean server load is smaller than the threshold X_m. So the control mechanism can switch to a lower level or has no change. In other words, it can maintain a higher information consistency as the server has a light load. When $LOAD(L_q(t), W_s(t), m) > X_m$, the mean load of the server system is higher than the threshold X_m. Hence, the control mechanism can switch the current level to a higher level or no change.

3.2 The Principle of Adaptive Control Mechanism

The First Level: Absolute Consistency

In Table 1, we list the principle for the first level by substituting parameter L for *m* in equation (1). There are two possible cases as defined in the following:

$$LOAD\ (L_q(t), W_s(t), L) = \frac{\frac{L_q(t)}{L_q}}{L} + \frac{\frac{W_s(t)}{W_s}}{L} \le X_L \quad \text{................(2)}$$

$$LOAD\ (L_q(t), W_s(t), L) = \frac{\frac{L_q(t)}{L_q}}{L} + \frac{\frac{W_s(t)}{W_s}}{L} > X_L \quad \text{................(3)}$$

When the condition of LOAD(·) satisfies equation (2), it means that the server load is light, and queue length and waiting time are short in the NVE server. So the NVE server can have capacity to maintain absolute consistency. When the condition of LOAD(·) is defined by equation (3), it means that the server load is higher. Therefore, the NVE server cannot maintain absolute consistency. Then, the control mechanism needs to switch the level from the first to the second by adopting blind broadcasting approach to handle the dynamic shared state in the networked virtual environment.

Table 1. The control principle using the function LOAD(·) and parameter X_L in the first level.

Waiting time ratio / Queue length ratio	$\frac{W_s(t)}{W_s} \le L$	$\frac{W_s(t)}{W_s} > L$
$\frac{L_q(t)}{L_q} \le L$	!	LOAD(·)ª X_L, ! LOAD(·)>X_L, ! #
$\frac{L_q(t)}{L_q} > L$	LOAD(·)ª X_L, ! LOAD(·)>X_L, ! #	#

The Second Level: Information Interest Management

In Table 2, we list the control principle for the second level by substituting parameter L or H for *m* in equation (1). There are three possible cases as defined in the following:

$$LOAD\ (L_q(t), W_s(t), L) = \frac{\frac{L_q(t)}{L_q}}{L} + \frac{\frac{W_s(t)}{W_s}}{L} \le X_L \quad \text{................(4)}$$

$$LOAD(L_q(t), W_s(t), L) = \frac{\frac{L_q(t)}{L_q}}{L} + \frac{\frac{W_s(t)}{W_s}}{L} > X_L$$

and

$$LOAD(L_q(t), W_s(t), H) = \frac{\frac{L_q(t)}{L_q}}{H} + \frac{\frac{W_s(t)}{W_s}}{H} \le X_H \quad \text{........................(5)}$$

$$LOAD\ (L_q(t), W_s(t), H) = \frac{\frac{L_q(t)}{L_q}}{H} + \frac{\frac{W_s(t)}{W_s}}{H} > X_H \quad \text{............(6)}$$

In the first case, when the condition of LOAD(·) satisfies equation (4), it means that the server load is light. In other words, the amounts of network message traffics decrease and the waiting time in server reduces. Therefore, the control mechanism can switch from the second level back to the first level to maintain absolute information consistency. In the second case, when the condition of LOAD(·) is defined by equation (5), it does not change the level by Definition 2. In the third case, when the condition of LOAD(·) is defined by equation (6), it means that the server load is very heavy. The server cannot use the information interest management to maintain the information consistency. So the control mechanism needs to switch from the second level to the third level by adopting dead reckoning approach to maintain the dynamic shared state in the networked virtual environment.

Table 2. The control principle using the function LOAD(·) and two parameters, X_L and X_H, in the second level.

Waiting time ratio / Queue length ratio	$\frac{W_s(t)}{W_s} \le L$	$\frac{W_s(t)}{W_s} > L$	$\frac{W_s(t)}{W_s} \le H$	$\frac{W_s(t)}{W_s} > H$
$\frac{L_q(t)}{L_q} \le L$	!	LOAD(·)ª X_L, # ! LOAD(·)>X_L, #	▨	▨
$\frac{L_q(t)}{L_q} > L$	LOAD(·)ª X_L, # ! LOAD(·)>X_L, #	#	#	▨
$\frac{L_q(t)}{L_q} \le H$	▨	#	#	LOAD(·)ª X_H, # LOAD(·)>X_H, # $
$\frac{L_q(t)}{L_q} > H$	▨	▨	LOAD(·)ª X_H, # LOAD(·)>X_H, # $	$

The Third Level: Predictive Simulation

In Table 3, we list the control principle for the third level by substituting parameter H for *m* in equation (1). There are two possible cases as defined in the following:

$$LOAD\ (L_q(t), W_s(t), H) = \frac{\frac{L_q(t)}{L_q}}{H} + \frac{\frac{W_s(t)}{W_s}}{H} \le X_H \quad \text{............(7)}$$

$$LOAD\ (L_q(t), W_s(t), H) = \frac{\frac{L_q(t)}{L_q}}{H} + \frac{\frac{W_s(t)}{W_s}}{H} > X_H \quad \text{............(8)}$$

When the condition of LOAD(·) is defined by equation (7), it means that the value of the server load function is lower than the threshold X_H and the server can use information interest management to maintain information consistency. So the control mechanism needs to switch from the third level to the second level. In contrast, when the condition of LOAD(·) is defined by equation (8), the server load is still very heavy. So the control mechanism should remain in the third level.

Table 3. The control principle using the function LOAD(·) and parameter X_H in the third level.

Waiting time ratio / Queue length ratio	$\frac{W_s(t)}{W_s} \le H$	$\frac{W_s(t)}{W_s} > H$
$\frac{L_q(t)}{L_q} \le H$	#	LOAD(·)ª X_H,$ # LOAD(·)>X_H, $
$\frac{L_q(t)}{L_q} > H$	LOAD(·)ª X_H,$ # LOAD(·)>X_H, $	$

Parameters L and H play an important role to determine the accuracy of dynamic shared state in the control mechanism. The values of L and H will affect the probability to adopt the first level and the third level. And the difference between L and H will also affect the probability to adopt the second level. Therefore, according to the capacity of the server, one can define appropriate values for L and H to be the lower bound and the upper bound. For simulation in the next section we let L=0.3 and H=0.7 for our NVE server.

The value of X_L and X_H can be calculated by Definitions 1, 2, and 3 in the following:

1. From Definition 1 :

$$\frac{L_q(t)}{L_q} \le L \quad \text{and} \quad \frac{W_s(t)}{W_s} \le L$$

$$\frac{\frac{L_q(t)}{L_q}}{L} \le 1 \quad \text{and} \quad \frac{\frac{W_s(t)}{W_s}}{L} \le 1$$

$$LOAD\ (L_q(t), W_s(t), L) = \frac{\frac{L_q(t)}{L_q}}{L} + \frac{\frac{W_s(t)}{W_s}}{L} \le 2 = X_L$$

2. From Definition 2 :

$$L < \frac{L_q(t)}{L_q} \le H \quad \text{and} \quad L < \frac{W_s(t)}{W_s} \le H$$

$$\left(\frac{\frac{L_q(t)}{L_q}}{L} > 1 \ \text{or} \ \frac{\frac{L_q(t)}{L_q}}{H} \le 1 \right) \text{and} \left(\frac{\frac{W_s(t)}{W_s}}{L} > 1 \ \text{or} \ \frac{\frac{W_s(t)}{W_s}}{H} \le 1 \right)$$

$$LOAD\ (L_q(t), W_s(t), L) = \frac{\frac{L_q(t)}{L_q}}{L} + \frac{\frac{W_s(t)}{W_s}}{L} > 2 = X_L$$

or

$$LOAD\ (L_q(t), W_s(t), H) = \frac{\frac{L_q(t)}{L_q}}{H} + \frac{\frac{W_s(t)}{W_s}}{H} \le 2 = X_H$$

3. From Definition 3 :

$$H < \frac{L_q(t)}{L_q} \quad \text{and} \quad H < \frac{W_s(t)}{W_s}$$

$$\frac{\frac{L_q(t)}{L_q}}{H} > 1 \quad \text{and} \quad \frac{\frac{W_s(t)}{W_s}}{H} > 1$$

$$LOAD\ (L_q(t), W_s(t), H) = \frac{\frac{L_q(t)}{L_q}}{H} + \frac{\frac{W_s(t)}{W_s}}{H} > 2 = X_H$$

Hence, the value for both X_L and X_H should be equal to 2 from the description above.

4 Performance Evaluation

This section presents performance characterization of three types of dynamics share state techniques and performance evaluation of the proposed control mechanism, the three-level consistency approach, and the shared repository. As we have known, the length of server queue and waiting time of the jobs in the server are uncertain parameters in the networked virtual environment. Based on the discussion in Section 3, we assume that the simulation model is based on the (M/M/1) : (FCFS/% /%) model.

Environment

Both $L_q = \dfrac{\rho^2}{1-\rho}$ and $W_s = \dfrac{1}{\mu(1-\rho)}$ are calculated from Little's formula. We plot the relationships of L_q and ρ, W_s and ρ in Figures 2 and 3. From Figures 2 and 3, we can see that L_q, W_s, and ρ have positive relationships. When the value of the traffic intensity ρ is low, L_q and W_s, could approach to zero, because the job arrival rate is much lower than the job service rate. As a result, the queue size L_q and the waiting time W_s are small, and the server load is light. When ρ is large (i.e. ρ approaches to 1), the job arrival rate is almost equal to the job service rate. The values of both L_q and W_s, and the server load will be very high. In our simulation, we let ρ be 0.7 and & be 4, then the L_q and W_s can be calculated from ρ and &

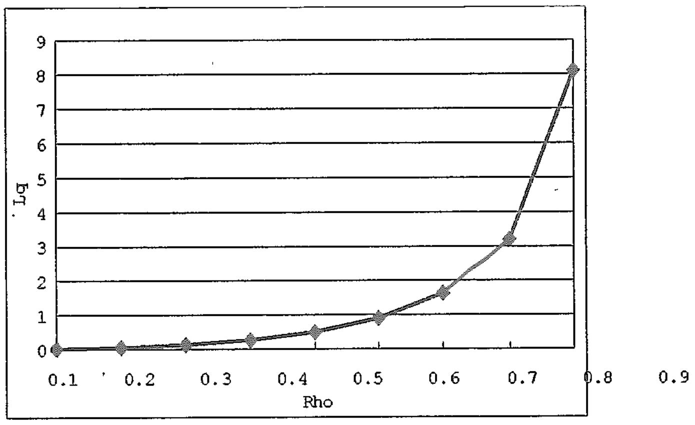


Figure 2. The relationship between L_q and Rho (Rho=ρ).

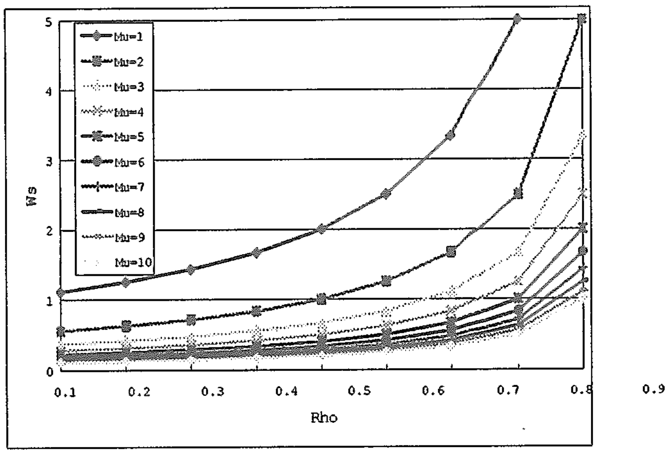


Figure 3. The relationship between W_s and Rho (Rho=ρ, Mu=μ).

Parameters and Load Models

The parameters used to evaluate the performance are the types of control mechanisms and the types of server load models, which are related to the number of clients, service time, and broadcast time. When the number of clients increases, it will increase the number of threads spawned by the server. At the same time, the service time and broadcast time will be also increased. We use two types of load models. The first one assumes that the load increases in proportional to the number of joined clients. The first load model is (1 + 3*n%). It means that for each new joined client it will contribute three percent of the original load. The second load model assumes that the load increasing rate is less than the first one and is modeled as $(1 + 0.056 * n^2 \%)$. The curves of two load models are shown in Figure 4. We use these two load models to evaluate the performance of three dynamic shared state techniques and two control mechanisms. To emulate the information interest management using the blind broadcasting approach, we assume that the updated rate will be delayed. In other words, we reduce the frequency to send the update information to clients. The reduction rate is 15 % of the quantity of that of the shared repository. We further reduce the updated rate to emulate the predictive simulation using the dead reckoning model. The reduction rate is 15 % of the quantity of that of the blind broadcasting. The server load variation under two load models for three dynamic shared state techniques is shown in Figure 5. Depending on the server capacity, the load of server increases exponentially as the number of participant client hosts increases. Under the second load model the server load increases less dramatic. This explains that the supporting of a large number of participants is a big problem for all NVEs.

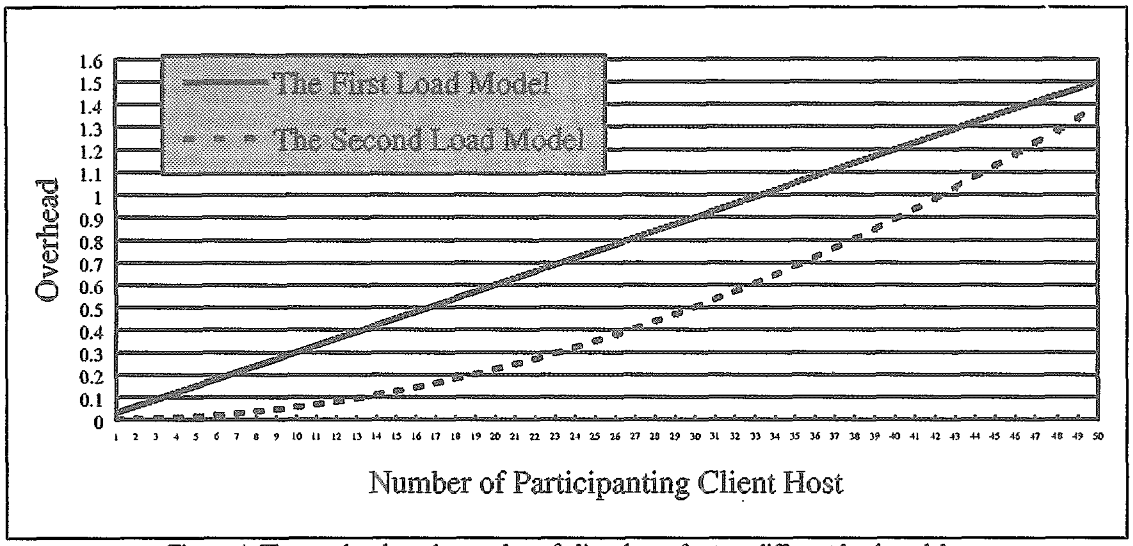


Figure 4. The overhead vs. the number of client hosts for two different load models.

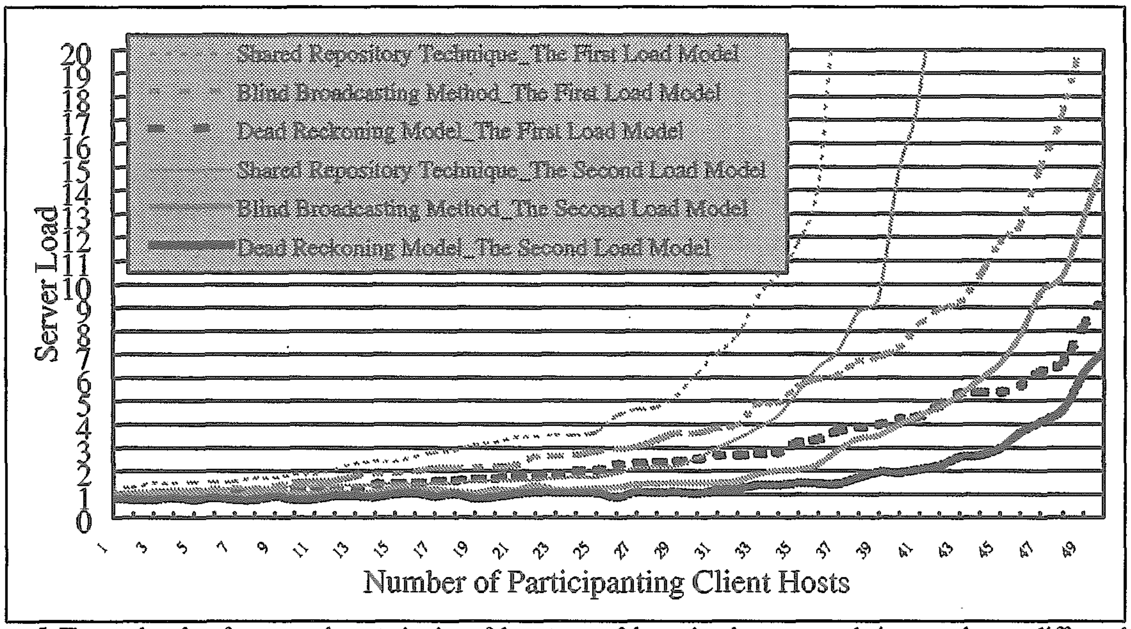


Figure 5. The results of performance characterization of three types of dynamics share state techniques under two different load models.

Performance Evaluation Using the First Load Model
In the first experiment we use the first server load model to simulate the load behavior in the server, when the number of participants increases in the networked virtual environment. We have experimented on three approaches – the shared repository technique, the three-level consistency approach [11], and the adaptive control mechanism. The results are shown in Figure 6, where the vertical axis is the value of the server load and the horizontal axis is the number of participants. We can see that the overhead of the shared repository technique increases very significantly as the number of participant increases. But, both of the three-level consistency approach and the adaptive control mechanism maintain very low server load for a large number of participants. They not only can control levels of information consistency but also refer to the server load to provide more accurate dynamic shared state. The vertical dash lines divide the intervals of the level of information consistency for the three level consistency approach. The threshold values are 10 and 25. The solid vertical lines indicate the change of the level of information consistency for the adaptive control approach. We observe that the adaptive control mechanism using the state of the server load provides higher information consistency than the three-level consistency approach using the number of participants does.

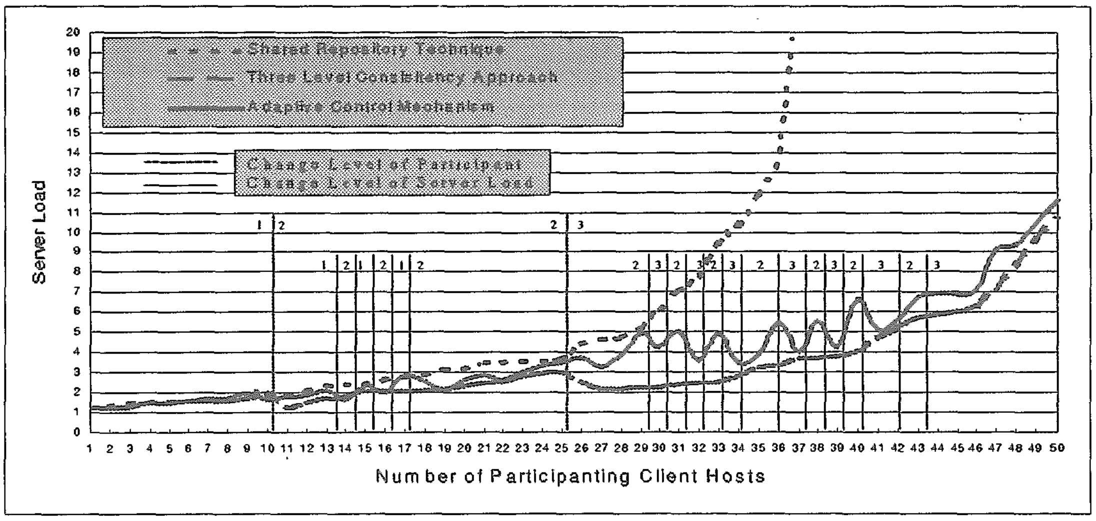


Figure 6. Server load versus the number of simultaneous participants in the server site.

Figure 7, we show the service quality of shared repository and two control approaches, the vertical axis is the rate of information consistency and the horizontal axis is the number of participants. Though the shared repository cannot support too many participants, it always provides the best information consistency all the time. Hence, we use it as reference to calculate the rate of information consistency provided by two types of control mechanisms. Since the shared repository is the reference, it maintains 100% for all range of the number of participants. The curves show that the rate of information consistency using the adaptive control mechanism is better than that of the three-level consistency approach.

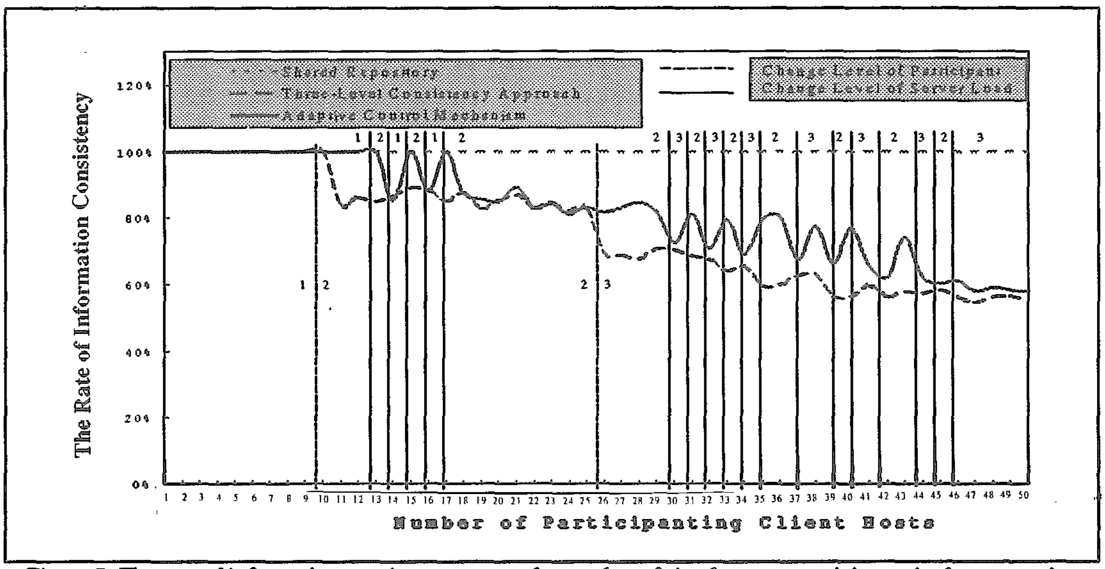


Figure 7. The rate of information consistency versus the number of simultaneous participants in the server site.

● **Performance Evaluation Using the Second Load Model**

In the second experiment we use the second server load model to simulate the server load behavior. In this experiment the client's behavior is different. In this case the load does not linearly increase with the number of participants. The performance results are shown in Figure 8. Figure 8 shows the service quality of shared repository and two types of control mechanisms. It shows that the estimation of the adaptive control mechanism can provide the highest information consistency as the shared repository until the number of participants is 30, but the evaluation of the maximum simultaneous participants' approach only can support the same level information consistency for 10 participants. The rate of information consistency using the estimation of the server load is better than that of the evaluation of the maximum simultaneous participants.

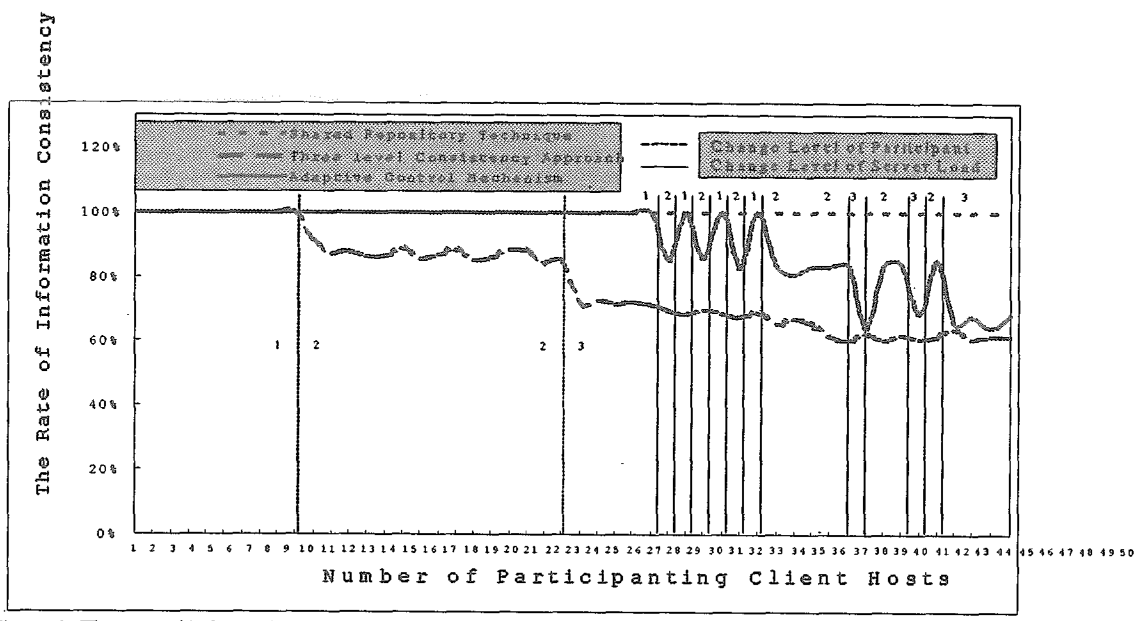


Figure 9. The rate of information consistency versus the number of simultaneous participants in the server site.

Discussion

The experiments show that both control mechanisms can significantly improve in the management of the number of participants and the level of information consistency. The three-level consistency approach has the advantages of simple in implementation and low overhead in switching. Though the adaptive control mechanism using the estimation of server load performs better, it may have oscillation problem, when the variation of the number of participants is very frequent. It also has a higher overhead due to estimating the server load.

5 Conclusions

Putting reality into a virtual environment requires collaborative interactions among humans and machines. From the above concept, in this paper we have presented an adaptive control mechanism to maintain information consistency and control the dynamic shared state for a large-scale virtual environment. Two types of load models are used to evaluation the performance of the shared repository technique, the three-level consistency approach, and the adaptive control mechanism. Evaluation results show that the proposed approach and the three-level consistency approach maintain lower server load than the shared repository technique over a larger number of participants. However, the adaptive control mechanism makes level switching based on the statues of a realistic virtual environment and can provide a better service quality.

References

[1] Anupam, V. and Bajaj, C. "Shastra: An Architecture for Development of Collaborative Applications." International Journal of Intelligent and Cooperative Information systems (IJICIS), Vol. 3, No. 2, pp. 155-166, 1994.

[2] Bricken, W. and Coco, G. "The VEOS Project." Presence: Teleoperators and Virtual Environments, Vol. 3, No. 2, pp. 30-39, 1994.

[3] Bridges, A. and Charitors, D. "On Architectural Design in Virtual Environments." Design Studies, Vol. 18, No. 2, pp. 143-154, 1997.

[4] Carlsson, C. and Hagsand, O. "DIVE – A Platform for Multi-User Virtual Environment." Computers & Graphics, Vol. 17, No. 6, pp. 663-669, 1993.

[5] Cremer, J., Kearney, J. and Ko, H. "Simulation and Scenario Support for Virtual Environments." Computers & Graphics, Vol. 20, No. 2, pp. 199-206, 1996.

[6] Department of Defense Modeling and Simulation Office "High Level Architecture Interface Specification Version 1.3." 1998. Available through the Internet: http://www.dmso.mil/.

[7] Duenyas, I., Gupta, D., and Olsen, T. L. "Control of a Single-server Tandem Queuing System with Setups." Operations Research, Vol. 46, No. 2, pp. 218-230, 1998.

[8] Escobar, J., Partridge, C. and Deutsch, D. "Flow Synchronization Protocol." IEEE/ACM Transaction on Networking, Vol. 2, No. 2, pp. 111-121, 1994.

[9] Institute for Electrical and Electronics Engineers "IEEE Standard for Distributed Interactive Simulation - Application Protocols." IEEE Standard 1278.1, 1995.

[10] Kleinrock, N. *Queuing Systems*, Vol. 1, 1976, New York: John Wiley & Sons.

[11] Lu, T. C., Lin, M. T., and Lee, C. N. "Control Mechanism for Large-Scale Virtual Environments." Journal of Visual Languages and Computing, Vol. 10, No. 1, pp. 69-85, 1999.

[12] Miller, D. and Thorpe, J. A. "SIMNET: The Advent of Simulator Networking." In Proceedings of IEEE, Vol. 83, No. 8, pp. 1114-1123, 1995.

[13] Pullen, J. M. "Networking for Distributed Virtual Simulation." Computer Networks and ISDN Systems, Vol. 27, pp. 387-394, 1994.

[14] Rommel, C.G. "The probability of load balancing success in a homogeneous network." IEEE Transactions on Software Engineering, Vol. 17, pp. 922-933, 1991.

[15] Shivaratri, N.G, Krueger, P., and Singhal, M. "Load distributing in locally distributed systems." IEEE Computer, Vol. 25, pp. 33-44, 1992.

[16] Sriram, M.G and Singhal, M. "Measures of the Potential for Load Sharing in Distributed Computing Systems." IEEE Transaction on Software Engineering, Vol. 21, No. 6, pp. 468-475, 1991.

[17] Taylor, D. "The VR-Protocol and What It Offers HLA." In Proceedings of the 1997 Spring Simulation Interoperability Workshop, Vol. 2, pp. 665-672, 1997.

[18] Zyda, M. et al. "NPSNET – Large-Scale Virtual Environment Technology Testbed." In Proceedings of International Conference on Artificial Reality and Tele-Existence, pp. 18-26, 1997.