ATLANTIS PRESS

Advances in Computer Science Research, volume 62

2nd Joint International Information Technology, Mechanical and Electronic Engineering Conference (JIMEC 2017)

An Asynchronous Control Method For Reducing Inconsistency In DVE

Wei Zhang *, Hangjun Zhou †

**College of Computer, National University of Defense Technology, Changsha, China*
†Department of Information Management, Hunan College of Finance and Economics, Changsha, China

Keywords: DVE systems; inconsistency; asynchronous control method

Abstract

Maintaining the consistency of each node is the key factor to preserve system usability in a distributed virtual environment (DVE) system. However, due to the existence of communication delay between nodes, it is hard to sustain the absolute consistency of the system while providing acceptable responsiveness of the nodes, thus affecting the interactive experience of the user in the virtual world. In this paper, we propose a new consistency control method based on the asynchronous clock model, which can satisfy the responsiveness requirement of each node and reduce the overall inconsistency of the system. In order to evaluate the performance of the proposed method, the experimental process and results analysis are also displayed in this paper.

1 Introduction

Distributed virtual environment [1] is a real-time network interaction environment based on distributed simulation technology [2,3]. Through the communication among nodes, it can share local or global information of the virtual world to help users collaboratively complete the tasks distributed to varying nodes, and provide users with a holistic, real, and immersive virtual space. Distributed virtual environment has rapidly developed in recent years, and has been widely used in online games, military simulation, and distance education and so forth.

The problem of consistency is the most crucial and central problem of distributed virtual environment. Due to the existence of the delay in network transmission, the messages transmitted between nodes cannot arrive at the target in real time. Therefore, an event will have different execution times in separate nodes, which brings about the phenomenon of disorder and affects normal running of the system. On the other hand, the responsiveness of the system is also an important factor affecting the usability of the system. Responsiveness is defined as the time interval between initiating an action and observing the effect of the action. Normally, users hope this response time will be consistent with their real-world perceptions. Otherwise, the interaction experience of the user in the virtual world will be significantly affected.

Most existing consistency control methods primarily try to optimize either of these two key elements. However, since there is a trade-off relationship between system consistency and responsiveness, it is often hard to provide satisfactory responsiveness for the methods that enhance the consistency, and vice versa. For this reason, we propose an asynchronous consistency control method to reduce the overall inconsistency of the system, while also ensuring the responsiveness of each node. The contribution of this paper mainly includes two aspects, one is to propose a calculation method of the system inconsistency, which can quantify the consistency degree of the system; the other is to propose an asynchronous consistency control method, which can optimize the overall consistency of the system, thereby improving the usability of the system.

The rest of this paper is organized as follows: Section 2 reviews the existing consistency control methods. Section 3 briefly describes the system model and gives the consistency models and responsiveness requirement model used in this paper. In Section 4, we present our asynchronous consistency control method and show how to use it to optimize the overall consistency of DVE systems. Section 5 exhibits and discusses some experimental results. Finally, section 6 contains our conclusions.

2 Related work

When the initial state of each node in the virtual environment system is consistent, as long as all events that change the state of the system are correctly executed at each node, it is sufficient to maintain consistency for the system. Because their states only change when events are executed, we only need to make sure each node handles events in the same order for the discrete models in the DVE system. This kind of problem has been studied in-depth [4,5]. As for the continual model in the virtual environment system, it is necessary to ensure that each event is executed at the correct time point because the system state changes with the time.

Existing consistency control methods for maintaining continuous models focus on how to eliminate inconsistencies in the system. In [6], the authors propose a lock-step synchronization technique, which avoids the occurrence of inconsistency by forcing the clocks of the blocking nodes to

advance until all of the nodes finish the calculation in the last cycle, and they then enter into the next cycle. This method can ensure the correctness of the system function, but the performance cannot meet the requirements of the interaction of the user. In [7], the authors propose a method to improve the performance of the system by delaying the remote events without delaying the local events. This model could provide an excellent reaction, but it would sacrifice the functions of the system. In many cases, the control result of this method cannot satisfy the consistency needs of the messages, which leads to the occurrence of various inconsistencies.

In [8], the author considers that the consistency control of the continuous model is a compromise between the correctness and performance of DVE system, and the optimization for one aspect will inevitably lead to the deterioration of another aspect of the DVE system. In addition, in [9] the author proposes a definition of time-space inconsistency. They analyze the factors that cause inconsistency and give some suggestions to reduce the inconsistency.

In addition, some information management techniques are introduced into the distributed virtual environment system, such as Relevance Filtering [10], Dead Reckoning [11], and Packet Bundling [12]. These methods can also indirectly improve the consistency of the system by reducing the amount of transmission data and transmission delay.

In order to improve the responsiveness of each node, our previous work [13] proposed a consistency control method based on a callback clock, which can enhance the overall responsiveness of the system while also fulfilling the consistency needs of each node. However, when the method is running, there are still some nodes whose responsiveness cannot meet the demand, which leads to the destruction in the interactive experience of the user in the virtual world. Therefore, in this paper, we will try to reduce the system inconsistency on the premise that the responsiveness of the user needs can be achieved.

3 System Model

In this section, we first describe the system model used in this paper, and then give the definitions of the consistency model and the responsiveness model, as well as the computational method of inconsistency used.

In this paper, we use V to represent the set of all nodes involved in the DVE system, and O to represent the set of all events generated and executed in the system. Given a $v_i \in V$, G_i denotes the set of all events generated by node v_i, R_i denotes the set of all events that node v_i can receive. We let $E_i = G_i \cup R_i$, where E_i denotes the set of all events that node v_i can perceive. Given an $o_m \in O$, if $o_m \in G_i$, $TG_i(o_m)$ indicates the generate time of event o_m at node v_i, $TS(o_m)$ denotes the excepted execution time of event o_m set by node v_i. If $o_m \in R_i$, $TR_i(o_m)$ indicates the receive time of event o_m at node v_i. If $o_m \in E_i$, $TE_i(o_m)$ denotes the actual execution time of event o_m at node v_i.

In order to keep a consistent view among all nodes in the virtual environment, it is necessary to satisfy certain constraints when the events in the system are executed at each node. As mentioned in [14], we divide the consistency requirements into three levels. These levels are categorized as basic time stamp order (BTSO), interval time stamp order (ITSO), and absolute time stamp order (ATSO).

Definition 1. (Basic Time Stamp Order, BTSO)

$$\forall v_i, v_i \in V; o_m, o_n \in E_i \cap E_j$$
$$TE_i(o_m) \le TE_i(o_n) \rightarrow TE_j(o_m) \le TE_j(o_n) \quad (1)$$

BTSO can guarantee that for any two nodes v_i and v_j, which can perceive events o_m and o_n, these two events will be executed at v_i and v_j in the same order.

Definition 2. (Interval Time Stamp Order (ITSO))

$$\forall v_i, v_i \in V; o_m, o_n \in E_i \cap E_j$$
$$TE_i(o_m) - TE_i(o_n) = TE_j(o_m) - TE_j(o_n) \quad (2)$$

ITSO can guarantee that for any two nodes v_i and v_j, which can perceive events o_m and o_n, the interval between execution times of these two events will be same at v_i and v_j. The demand of ITSO is stricter than that of BTSO.

Definition 3. (Absolute Time Stamp Order, ATSO)

$$\forall v_i, v_i \in V; o_m \in E_i \cap E_j \rightarrow TE_i(O_m) = TE_j(O_m) \quad (3)$$

ATSO has the most severe demand among all consistency requirements. It requires that any event will be executed at the same time at all nodes that can perceive the event.

For discrete models in the DVE system, BTSO can achieve the system consistency requirements. But for continuous models, the general requirements are to keep ATSO, only a few applications need to just keep ITSO [8].

In addition to the functional correctness of the system, the performance requirements should also be considered. Because it represents the ability of the system to respond to the interaction, the responsiveness is an important factor affecting system usability in general. Given an event o_m, which is generated from node v_i, we use $CF_i(o_m)$ to represent the response time of the event at node v_i, which is equal to the difference between the excepted execution time $TS(o_m)$ and the generate time of the event $TG_i(o_m)$.

Definition 4. (Event Responsiveness)

$$CF_i(O_m) = TS(O_m) - TG_i(O_m), \forall v_i \in V, o_m \in G_i \quad (4)$$

For each node in the system, the responsiveness of a node is defined as the average responsiveness of all events it generated.

Definition 5. (Node Responsiveness)

$$CF_i = \frac{\sum_{\forall o_m \in G_i} CF_i(O_m)}{|G_i|}, \forall v_i \in V \quad (5)$$

We use CFM to denote the maximum tolerance responsiveness of a node. If $CF_i \le CFM$, we consider that the responsiveness requirements of the node are fully attained, otherwise the responsiveness requirements of the node are not satisfied and the the interactive experience of the user is affected.

Given an event o_m , if we want to guarantee that its responsiveness $CF_i(o_m)$ is not greater than CFM according to Definition 4, the expected execution time $TS(o_m)$ needs to satisfy the condition $TS(O_m) \le TG_i(O_m) + CFM$. Moreover, to meet the requirements of ATSO described in Definition 3, the actual execution time of the event at each node needs to be equal to the excepted execution time set by the event generation node, that is to say, $\forall v_j, o_m \in E_j \Rightarrow TE_j(o_m) = TS(o_m)$. However, attributable to the existence of network transmission delay, there may be a node v_j that received the event at a time that is later than the excepted execution time, that is $TR_j(o_m) > TS(o_m)$. In this case, we cannot guarantee that the event is executed at the same time at all other nodes. Here, we define the inconsistency degree of event o_m at node v_j as the difference between the actual execution time at v_j and the excepted execution time.

Definition 6. (Event Inconsistency Degree)

$$IC_j(O_m) = \begin{cases} 0 & , \ TE_j(O_m) \le TS(O_m) \\ TE_j(O_m) - TS(O_m), & TE_j(O_m) > TS(O_m) \end{cases} \quad (6)$$

We then define the inconsistency degree of the whole system as the sum of the inconsistency degree of all of the events at all nodes.

Definition 7. (System Inconsistency Degree)

$$IC_{all} = \sum_{\forall v_i \in V, o_m \in O} IC_i(o_m) \quad (7)$$

In [9], the authors also proposed a definition of time and space inconsistency, and quantified the inconsistency in the virtual environment system. However, that definition emphasizes inconsistencies in spatial positional deviations attributable to temporal inconsistencies, which are different from the objectives of this paper. The inconsistency in this paper is based on the definition of the ATSO consistency model, and it calculates the degree of deviation between the real system and the ideal situation. As the responsiveness of the system has a decisive influence on the interactive experience of the user, the optimal goal of this paper is to minimize the inconsistency of the system under the premise of ensuring the responsiveness requirement of the system is satisfied. In the next section, we will introduce the asynchronous clock model and the consistency control method used within this paper.

4 Asynchronous Consistency Control Method

4.1 Asynchronous Clock Model

Due to the existence of real-time interaction in the distributed virtual environment, most existing methods make the assumption that the simulation time of each node needs to be strictly synchronized with the external wall clock time. In order to ensure that all events are simultaneously executed at all nodes, the responsiveness cannot be less than the maximum network transmission delay under this assumption, which results in the contradictions between system consistency and performance. In fact, the strict synchronization between the simulation time and the wall clock time is unnecessary, and we can observe what happens if the simulation time and the wall clock time are inconsistent.

In Figure 1, there are two nodes, v_1 and v_2 . Node v_1 sends an event o_1 to node v_2 at time T , the delay time of the event is ΔT . Node v_2 will receive the event at time $T + \Delta T$, so node v_1 must wait ΔT to execute the event. Therefore, the responsiveness of node v_1 is ΔT . In this case, if we adjust the simulation time of node v_2 to make it ΔT slower to the wall clock time, then the simulation time of node v_2 is T-ΔT when the event is sent by node v_1 , and the receive time of this event at node v_2 is T . Therefore, node v_1 can execute the event immediately after sending the event because both node v_1 and v_2 can execute the event at time T , the responsiveness of node v_1 becomes 0. In turn, if node v_2 sends messages to node v_1 , it has to wait more time, so the responsiveness of node v_2 becomes $2\Delta T$.

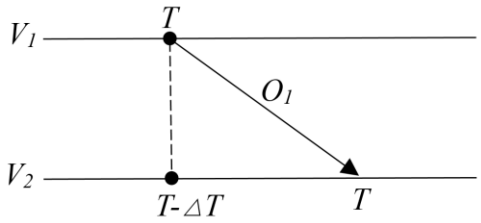

Fig. 1. An Example of Asynchronous Clock Model

From the above example, we can see the system time resources can be redistributed by adjusting the deviation time of the simulation time of individual nodes and the wall clock time. Simultaneously, the system consistency situation can be

optimized in the process. We define the difference between the simulation time of a node and the wall clock time as the deviation time of the node.

Definition 8. (Node Deviation Time)

$$\forall v_i \in V, d_i = t_w - t_i \tag{8}$$

In the case where the deviation time exists, for a given event o_m and the node v_i that generates the event, we can obtain its actual execution time at the receiving node v_j is $TE_j(O_m) = TG_i(O_m) + e_{ij} + d_i - d_j$, where e_{ij} represents the communication delay between node v_i and node v_j . According to Definition 4 and Definition 6, it can be seen that the event inconsistency in the asynchronous clock model can be calculated according to the following formula.

Definition 9. (Event Inconsistency Degree in asynchronous clock model)

$$IC_j(O_m) = \begin{cases} 0 & , \ e_{ij} + d_i - d_j \le CFM \\ e_{ij} + d_i - d_j - CFM & , \ e_{ij} + d_i - d_j > CFM \end{cases} \tag{9}$$

Therefore, the goal of this paper is to choose and adjust the deviation time value of each node to reduce the overall system inconsistency under the asynchronous clock model. In the next subsection, we will detail the consistency control method used.

4.2 Asynchronous Consistency Control Method

The key issue of asynchronous consistency control method is to choose proper values of deviation time for all nodes. In fact, the deviation time of a node affects both the inconsistency degree of events it generated, as well as the inconsistency degree of the events it received. When the deviation time increases, the degree of inconsistency degree of received events decreases, and the inconsistency degree of sent events increases, and vice versa. If the frequencies of receiving events and sending events are not equal, we can adjust the deviation time to reduce the system overall inconsistency. Due to the difference of the physical properties of the entities, specifically the difference of the user's operating habits and the difference of the area of interest (AOI), it is common that the frequency of sending and receiving events do not match in the DVE system. In order to optimize the system as a whole target, we need to consider the deviation time values for all nodes. For each node, the deviation time is related to the difference between the frequency of its receive event and the frequency of its generate event. The larger the difference, the greater the deviation time of the node should be. Moreover, there is also a constraint for the deviation times of neighbor nodes.

Condition 1. (Deviation Time Constraints)

$$\forall v_i, v_j, G_i \cap R_j \neq \varnothing \Rightarrow d_j \le e_{ij} + d_i - CFM \tag{10}$$

According to Definition 9, the reason for the existence of this constraint is, for any node v_i and v_j , when the deviation time of the node v_j has reached $e_{ij} + d_i - CFM$, the inconsistency degree of the reception event of the node is already 0. The additional increasing of the deviation time will not further decrease the inconsistency degree of receiving events, but rather will increase the inconsistency degree of sending events. Under the above constraint, the deviation time of the entire system can be subjected to a linear programming problem, the overall operation of the process is shown in Figure 2.

Algorithm 1.
1: ***set*** d_i=0 *at initial stage*
2: ***if*** *TimerCounter > Counter_TH* ***then***
3: *TimerCounter = 0*
4: *Compute average communication delay* e_{ij} *for every* v_j *from* Int_G_i *and* Int_R_i
5: *Compute event generate frequency* gf_i *from* Int_G_i
6: *Compute event receive frequency* rf_i *from* Int_R_i
7: ***clear*** *set* Int_G_i *and* Int_R_i
8: *send* $\{\{e_{ij}\}, gf_i, rf_i\}$ *to center node and wait for response*
9: *receive new_*d_i *from center node and* ***set*** d_i=*new_*d_i
10: ***else***
11: *TimerCounter = TimerCounter + 1*
12: *Collect all* $o_m \in G_i$ *in* Int_G_i
13: *Collect all* $o_n \in R_i$ *in* Int_R_i
14: ***end if***

Fig. 2. Asynchronous Consistency Control Method

To optimize the control results of the asynchronous clock model, we set up a central tuning node in the system. In the initial stage of the distributed virtual environment system, we first set the deviation time of each node to 0. Then, during the running period, we periodically gather statistics of the communication delay and event sending frequency among nodes and their neighbor nodes, and gather this information to the central node. The central node will use the linear programming method according to the current system running state, calculate the new deviation time of each node, and broadcast the results. After each node receives the message, it will adopt the new deviation time in the following cycle. The process runs periodically during the system run-time, and continues to optimize overall system consistency. In the next section, we will show the experimental results of the method.

5 Experimental Results

In order to verify the effectiveness of the proposed method, we constructed a simulated distributed virtual environment system. Our test platform is a PC with an Intel i7 2600 2.8GHz processor and 12GB RAM. The number of nodes in the system ranges from 100 to 500.

We use GT-ITM to generate simulated internet topologies to simulate the communication characteristics of the network. We generated a Waxman1 random model and a Transit-Stub model ranging from 100 to 500 nodes, respectively. For the Waxman1 model, we use $\alpha = 0.6$, $\beta = 0.4$. For the Transit-Stub model, we set the transit-transit link delays to [20ms, 40ms], the transit-stub link delays to [50ms, 100ms], and the communication delays between nodes in the same stub domain to [20ms, 40ms].

We ran the simulated virtual environment for 10000 cycles, and in each cycle, every node has a certain probability to send events to its neighbors. We ran our asynchronous consistency control algorithm one time every 100 cycles, and compared the statistical results to the one without using the consistency control algorithm. The results are shown in Fig. 3 and Fig. 4.

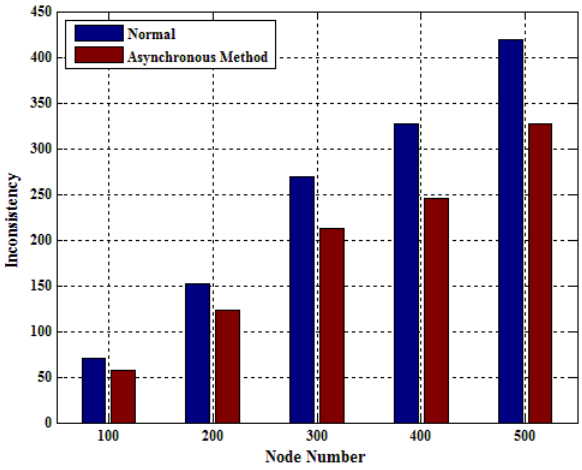

Fig.3. Comparison results on Waxman Model

In Figure 3, we first compare the results in the Waxman1 model. The x-axis represents the number of nodes in the virtual environment, varying from 100 to 500, and the y-axis represents the average inconsistency degree of the system for each cycle throughout the run. The blue columns represent the results without using the consistency control method, and the red columns represent the results using the consistency control method. From the results, we can observe that our method can effectively reduce the total system inconsistency in several cases from 100 to 500, and the overall inconsistency degree can be decreased by at least 15%.

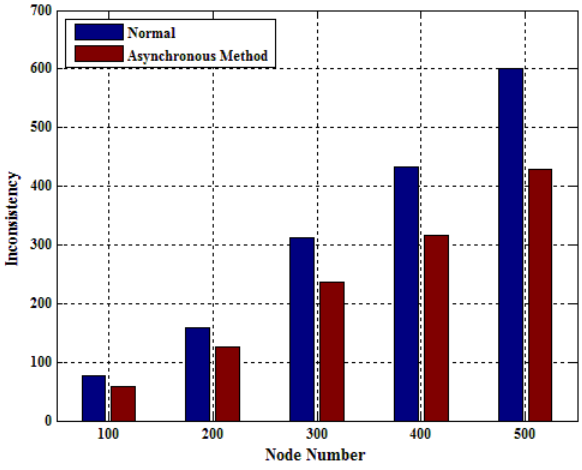

Fig. 4. Comparison results on Transit-Stub Model

Subsequently, we compared the statistical results in the hierarchical model. In Figure 4, the x-axis also represents the number of participating nodes in the virtual environment, varying from 100 to 500, and the y-axis denotes the average inconsistency degree of the system for each cycle throughout the run. From the results, we can see that the method in this paper is also better in the hierarchical model, and the reduction ratio of the system inconsistency can reach over 20%. The experiments show that the asynchronous consistency control method can reduce the overall inconsistency of the system and improve the usability of the virtual environment system under the premise of guaranteeing the system responsiveness.

6 Conclusion

Consistency and responsiveness are two key factors in determining the usability of the distributed virtual environment system. Because of the trade-off relationship between these two factors, it is hard to optimize both aspects at the same time. In this paper, we proposed an asynchronous clock model based control method. By periodically adjusting the deviation times of the participating nodes during the run-time, the time resources of the system are reallocated, reducing the whole system inconsistency and improving the overall performance of the system. The experimental results proved the effectiveness of the proposed method.

The definition of inconsistency proposed in this paper can quantitatively calculate the degree of consistency within the system. However, whether the causal violation exists in the inconsistency phenomenon also has a great influence on the sense of reality. Therefore, we will further consider how to incorporate causal factors into the consistency metrics in future works.

Acknowledgements

The work described in this paper was supported by the National Natural Science Foundation of China (Grant No. 61303187).

References

[1] Stytz M R. Distributed virtual environments[J]. IEEE Computer Graphics and Applications, 1996, 16(3): 19-31.

[2] Tang Y H, Zhang B D, Wu J J, et al. Parallel architecture and optimization for discrete-event simulation of spike neural networks[J]. Science China-Technological Sciences, 2013, 56(2), pp. 509-517.

[3] Hou B, Yao Y, Wang B, et al. Modeling and simulation of large-scale social networks using parallel discrete event simulation[J]. Simulation-Transactions of The Society for Modeling and Simulation International, 2013, 89(10) , pp. 1173-1183.

[4] Zhou S, Cai W, Turner S J, et al. Critical causal order of events in distributed virtual environments[J]. ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM), 2007, 3(3): 15.

[5] Cai W, Turner S J, Lee B S, et al. An alternative time management mechanism for distributed simulations[J].

ACM Transactions on Modeling and Computer Simulation (TOMACS), 2005, 15(2): 109-137.

[6] Hernndez S P, Fanchon J, Drira K. The immediate dependency relation: an optimal way to ensure causal group communication[J]. Annual Review of Scalable Computing, 2004, 6(3): 61-79.

[7] Qin X. Delayed consistency model for distributed interactive systems with real-time continuous media[J]. Journal of Software, 2002, 13(6): 1029-1039.

[8] Mauve M, Vogel J, Hilt V, et al. Local-lag and timewarp: providing consistency for replicated continuous applications[J]. IEEE transactions on Multimedia, 2004, 6(1): 47-57.

[9] Zhou S, Cai W, Lee B S, et al. Time-space consistency in large-scale distributed virtual environments[J]. ACM Transactions on Modeling and Computer Simulation (TOMACS), 2004, 14(1): 31-47.

[10] Bassiouni M A, Chiu M H, Loper M, et al. Performance and reliability analysis of relevance filtering for scalable distributed interactive simulation[J]. ACM Transactions on Modeling and Computer Simulation (TOMACS), 1997, 7(3): 293-331.

[11] Blau B, Hughes C E, Moshell M J, et al. Networked virtual environments[C]. Proceedings of the 1992 symposium on Interactive 3D graphics. ACM, 1992: 157-160.

[12] Liang L A H, Cai W, Lee B S, et al. Performance analysis of packet bundling techniques in DIS[C]. Distributed Interactive Simulation and Real-Time Applications, 1999. Proceedings. 3rd IEEE International Workshop on. IEEE, 1999: 75-82.

[13] Zhang W, Zhou H J, Peng Y X, et al. Asynchronous time consistency control methods in distributed interactive simulation[J]. Journal of Software, 2010, 21(6): 1208-1219.

[14] Zhang W, Zhou H, Peng Y, et al. Providing Responsiveness Requirement Based Consistency in DVE[C]. Parallel and Distributed Systems (ICPADS), 2009 15th International Conference on. IEEE, 2009: 594-601.