International Journal of Database Theory and Application
Vol.9, No.11 (2016), pp.317-326
http://dx.doi.org/10.14257/ijdta.2016.9.11.28

A Distributed Consistency Control Method for Multi-level Responsiveness Requirements in DVE Systems

Wei Zhang[1], Hangjun Zhou [2]

[1] *College of Computer, National University of Defense Technology, China*
[2]*Department of Information Management, Hunan College of Finance and Economics, Changsha, China*
[1]*zw_nudt@163.com,* [2] *zhjnudt@gmail.com*

Abstract

In a distributed virtual environment (DVE) system, maintaining system consistency and providing acceptable responsiveness are two core elements for maintaining system usability. Existing consistency control methods either ignore the responsiveness requirement factor or assume that each node has the same responsiveness requirements which cannot meet the real needs of various entities in the virtual environment system. In this paper, we propose a distributed consistency control method for multi-level responsiveness requirements, which can optimize the system consistency according to the real responsiveness needs of different nodes. Our method can achieve a better balance between consistency control and responsiveness optimization, thereby improving the system usability effectively. In order to evaluate the performance of the proposed method, the experimental process and results analysis are also shown in this paper.

Keywords: *Distributed Virtual Environment; Multi-level Responsiveness Requirements; Consistency Control Method*

1. Introduction

Distributed virtual environment [1] and analytic simulation system [2] are two main application scenarios of distributed simulation technology [3, 4]. The distributed virtual environment constructs an interactive virtual world for multiple nodes and provides a holistic and immersive virtual space for the geographically distributed users to accomplish the task in the space. Distributed virtual environment has developed rapidly in recent years, and has been widely used in online games, military simulation, distance education and so on.

In a distributed virtual environment system, maintaining system consistency and providing acceptable responsiveness are two core elements for maintaining system usability. Consistency refers to maintaining the same state of the system at each participating node and providing a consistent, unified view for the users. If the consistency cannot be guaranteed, causal violation, timing chaos and other inconsistency phenomena will appear in the system and affect the user's normal operations. Providing responsiveness is also very important because a user needs to observe the action effect within an acceptable time after an action has been issued in the system. If the interval is too long, it will seriously affect the user's interactive experience in the virtual world. In order to maintain the usability of the system, a large number of researchers work on optimizing these two aspects. However, because of the trade-off relationship between system consistency and responsiveness, it is difficult to optimize both aspects at the same time.

Existing consistency control methods either ignore the responsiveness requirement factor, target the absolute consistency of the system, or consider a relatively simple responsiveness requirement model, for example, assuming that each node has the same

ISSN: 2005-4270 IJDTA

responsiveness requirement. These assumptions make it difficult to match the actual demand model for a large-scale virtual environment system with multiple types of entities. In fact, due to differences in the physical properties of the entity, the differences in user operating habits and other factors, the responsiveness demand of different nodes in the virtual environment will be quite different. In this paper we call it the multi-level responsiveness requirements. Consistency control method needs to consider the multi-level responsiveness requirements of different nodes and meets the actual needs of different users. Therefore, in this paper, we propose a distributed consistency control method for multi-level responsiveness requirements, which can ensure that the multi-level responsiveness requirements of each node can be effectively met. The method can optimize the overall consistency of the system, thereby improving system usability effectively.

The rest of the paper is organized as follows: Section 2 reviews the existing consistency control methods, Section 3 gives the consistency model and multi-level responsiveness model used in this paper. Section 4 describes the proposed distributed consistency control method for multi-level responsiveness requirements, Section 5 shows and discuss the experimental results, and finally, section 6 contains our conclusions.

2. Related Work

The existing consistency control methods can be divided into three categories according to the objectives of optimization. The first category is to ensure the consistency of the system as the goal, and ignore the responsiveness factor. In [5], the authors propose a lock-based consistency control technique, which locks every node until the synchronization process has completely finished in every cycle. A similar method is proposed in [6], the method proposed in that paper controls the status of each node in the system by means of fence synchronization. Although this kind of method can guarantee the consistency of the system, but due to the lack of taking into account the user's interactive experience, its practicality is greatly limited.

The second category of consistency control method is to ensure the consistency of the system as the premise, and optimize the system's responsiveness as far as possible. In [7], the authors propose a local delay based consistency control technique which guarantees the consistency of the system by delaying all the events in the local node and the receiving node for a period of time. A similar approach was proposed in [8], which proposed to maintain the state consistency of each node of the system by utilizing some time buckets. In addition, in [9], the authors propose an asynchronous clock based consistency control method, which can improve the system's responsiveness while ensuring the system's time consistency.

The goal of the aforementioned methods is to achieve the absolute consistency of the system. Some other methods research on some weak consistency models to improve the system's responsiveness capacity. In [10], the authors proposed an interval consistency control method based on estimating network delay, which can ensure the consistency of the message execution time interval at every node in the system. In addition, there is a class of research work [11, 12] for the causal consistency, through the establishment of happened before relationship [13] to construct the causal relationship between events, and to ensure the events causality.

The third category of consistency control method is to ensure system responsiveness as the prerequisite and maximize the consistency degree of the system. In [14], the authors proposed a delayed consistency control method which executes the local events immediately and delays the remote events for a constant time. This method can provide a good responsiveness but cannot guarantee a good consistency of the system. In [15], the authors proposed a centralized consistency control method, which reduces the inconsistency degree of the system by concluding the consistency problem as a linear programming problem. However, the centralized method is difficult to run efficiently

when the system scale is very large. In [7], the authors concluded that the consistency control of continuous model is a compromise between the function and performance of DVE system, and the optimization for one aspect will inevitably lead to the decrease of another aspect of DVE system. In addition, in [16], the authors gave a definition of time space inconsistency, they analyzed the factors that cause the inconsistency, and gave some suggestions to reduce the inconsistency.

In addition, in order to reduce the amount of data transmission and improve the efficiency of data processing, some information management techniques are introduced into distributed virtual environment, such as relevance filtering [19], dead reckoning [20], Package binding [21] and so on.

In order to satisfy the different responsiveness requirements of all nodes, in our previous work [17], we proposed an asynchronous consistency control model for different responsiveness requirements, which can improve the consistency of each node and the overall responsiveness satisfy degree simultaneously. However, due to the uncertainty of the latency in the large-scale DVE system, the responsiveness requirements of some nodes cannot be met at run time, and the interactive experience of the users in the virtual world will be destroyed. Therefore, in this paper, we will first make sure the user's various responsiveness requirements are satisfied, and then try to reduce the system inconsistency and improve the overall system usability.

3. System Model

In this paper, we use V to represent the set of all nodes involved in the DVE system, and E to represent the set of all transmission delays between nodes. Given two nodes $v_i, v_j \in V$, we use $e_{ij} \in E$ to represent the transmission delay between v_i and v_j. We use O to represent the set of all events generated and executed in the system. Given a node $v_i \in V$, 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. Given an event $o_m \in O$, if $o_m \in G_i$, $TG_i(o_m)$ denotes 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 $o_m \in R_i$, $TR_i(o_m)$ denotes the receive time of event o_m at node, $TE_i(o_m)$ denotes the actual execution time of event o_m at node v_i.

In the following, we show some consistency models that can keep all nodes in the same state. The traditional consistency model mainly includes receive order, priority order, causal order, causal and totally order and timestamp order [22]. In [9], we further divide the timestamp order into three levels. They are basic time stamp order(BTSO), interval time stamp order (ITSO) and absolute time stamp order (ATSO).

Definition 1 (Basic Time Stamp Order).

$$\forall v_i, v_i \in V; o_m, o_n \in R_i \cap R_j \Rightarrow TE_i(o_m) \le TE_i(o_n) \rightarrow TE_j(o_m) \le TE_j(o_n) \quad (1)$$

Basic Time Stamp Order is the most basic time consistency demand, which ensures that all events can be executed in the same order at each node.

Definition 2 (Interval Time Stamp Order).

$$\forall v_i, v_i \in V; o_m, o_n \in R_i \cap R_j \Rightarrow TE_i(o_m) - TE_i(o_n) = TE_j(o_m) - TE_j(o_n) \quad (2)$$

Interval Time Stamp Order is stricter than Basic Time Stamp Order. It requires all events to be executed in a consistent sequence, while ensuring that the time intervals between events are consistent.

Definition 3 (Absolute Time Stamp Order).

$$\forall v_i, v_i \in V; o_m, o_n \in R_i \cap R_j \Rightarrow TE_i(O_m) = TE_j(O_m) \tag{3}$$

Absolute Time Stamp Order is the most strict time consistency demand, which requires that all events in the system can be executed simultaneously on all nodes that can receive the event.

In addition to the functional correctness requirements of the system, we also need to consider the performance requirements, that is, response time requirements or responsiveness requirements. It represents the system's ability to respond to the interaction. For an event $\forall o_m \in G_i$, we use $CF_i(o_m)$ to denote the responsiveness of the event on the node v_i, which is equal to the difference between the expected execution time and the generation time of the event.

Definition 4 (Event Responsiveness).

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

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

Definition 5 (Node Responsiveness).

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

For a given node v_i, we use rr_i to denote its responsiveness requirement. Due to the differences in the physical attributes of the entities and the differences in the user's operating habits, there is a big difference in the responsiveness requirements of different nodes. In this paper, we adopt a multi-level responsiveness requirement model. We assume that there are multiple levels of responsiveness requirements in the system, and the requirements of different nodes belong to different levels.

Definition 6 (Multi-level Responsiveness Requirements).

$$rr_i = xRRUnit, x \in [1, RRMax], \forall v_i \in V \tag{6}$$

In the definition, *RRUnit* represents the basic unit between different responsiveness requirement levels, *RRMax* represents the maximum value of the responsiveness requirement level. The larger the value of x is, the weaker the responsiveness requirement of the node is, and vice versa. To ensure that the node's actual responsiveness meets its responsiveness requirements, it is necessary to ensure that the following constraints are met.

Condition 1 (Responsiveness Requirement Satisfaction Constraints).

$$\forall o_m \in G_i, TS(O_m) \le TG_i(O_m) + rr_i \tag{7}$$

However, due to the existence of network transmission delay, there may be a node v_j that receive the event later than the expected execution time, that is $TR_j(o_m) > TS(o_m)$. In this case we cannot guarantee that the event is executed simultaneously at all nodes. Here we define the inconsistency degree of event o_m at the node v_j as the difference between the actual execution time and the expected execution time.

Definition 7 (Event Inconsistency Degree).

$$IC_j(O_m) = \begin{cases} 0 & , \ TE_j(O_m) \le TS(O_m) \\ TE_j(O_m) - TS(O_m), & TE_j(O_m) > TS(O_m) \end{cases} \quad (8)$$

Then, we define the inconsistency degree of the whole system as the sum of the inconsistencies of all the events at all nodes.

Definition 8 (System Inconsistency Degree).

$$IC_{all} = \sum_{\forall v_i \in V, o_m \in O} IC_i(o_m) \quad (9)$$

In this paper, we will first ensure that each node's multi-level responsiveness requirements are met, and then through the distributed consistency control method to reduce the overall system inconsistency. In our previous work [15], we proposed a control method to reduce the inconsistency degree of DVE system, but this method does not take into account the multi-level responsiveness requirements of different nodes in the system, and because that method adopts centralized processing method, it cannot guarantee the efficiency when the scale of the system is very large. In the next section, we will introduce the asynchronous clock model and the distributed consistency control method used in this paper.

4. Distributed Consistency Control Method

4.1 Asynchronous Clock Model

Unlike the traditional synchronous clock model, the asynchronous clock model [9] does not guarantee that each node's simulation time is strictly synchronized with the wall clock time. Instead, in asynchronous clock model the system time resources are reallocated by setting the deviation time for some nodes to optimize the system performance. The deviation time for each node is defined as the difference between the wall clock time and its simulation time.

Definition 9 (Node Deviation Time).

$$\forall v_i \in V, d_i = t_w - t_i \quad (10)$$

In the case where the deviation time exists, for a given event $o_m \in G_i$, its actual execution time at a certain node v_j can be known as $TE_j(O_m) = TG_i(O_m) + e_{ij} + d_i - d_j$. So we can get the calculation method of the event inconsistency in the asynchronous clock model.

Definition 10 (Event Inconsistency Degree in Asynchronous Clock Model).

$$IC_j(O_m) = \begin{cases} 0 & , \ e_{ij} + d_i - d_j \le rr_i \\ e_{ij} + d_i - d_j - rr_i, & e_{ij} + d_i - d_j > rr_i \end{cases} \quad (11)$$

The goal of this paper is to adjust and set the deviation time of each node so that the total system inconsistency is minimized under the asynchronous clock model. The following example describes how to adjust the deviation time to optimize the system consistency.

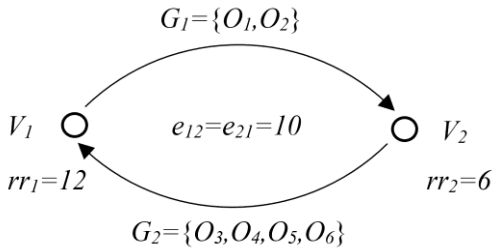

Figure 1. An Example of Asynchronous Clock Model

As shown in Figure 1, there are two nodes in the system, the communication delay between two nodes is 10, and the responsiveness requirements of two nodes are 12 and 6 respectively. During one period, node v_1 sends two events to node v_2, and node v_2 sends four events to node v_1. In the current state, it can be calculated that the inconsistency degrees of these two events v_1 generates are 0, and the inconsistency degrees of the four events v_2 generates are 4, and the total inconsistency degree is 16. At this point, if we set the deviation time of the node v_1 to 2, then the inconsistency degree of the two events are still 0, but inconsistency degree of the four events have become 2, the overall system inconsistency degree is reduced to 8. If the deviation time of node v_1 is set to 4, the inconsistency degree of the two events becomes 2, the inconsistency degree of the four events generated by node v_2 becomes 0, and the total inconsistency degree of the system is reduced to 4, the system inconsistency degree is reduced by 75% compared to the initial state. It can be seen that the system time resources can be reallocated by the asynchronous clock method, which reduces the overall system inconsistency. In the next subsection, we will detail the distributed consistency control method.

4.2 Algorithm Description

The key issue of asynchronous clock consistency control method is how to set the proper deviation time of each node. We need to consider the difference of the responsiveness requirement of each node and the difference between the frequencies of sending and receiving events. For nodes with weak responsiveness requirements, it may be appropriate to increase deviation time, and vice versa. In addition, if the difference between the frequencies of receiving events and sending events is large, the deviation time for that node should also be set to a large value.

When we set the deviation time of each node, the constraints of deviation time between nodes should be taken into account. For a given node v_j and its neighbor node v_i, if the deviation time of v_j exceeds rr_j and reaches $e_{ij}+d_i-rr_i$, the inconsistency degree of node's received events have been 0, further increasing the deviation time of v_j will not reduce the inconsistency degree of the events from node v_i, but will increases the inconsistency degree of events v_j generates. Here, we call $e_{ij}+d_i-rr_i$ the constraint value of node v_i for the node v_j. It is necessary to compare the frequencies of receiving events that removed the element of receiving events from the node v_i and sending events of the node v_j, then we could know if we should continue to increase the deviation time of v_j. For this reason, we designed a distributed deviation time adjustment strategy, the main operation of the process is shown in Figure 2.

Algorithm 1.

set d_i=0 *at initial stage*
if *TimerCounter > Counter_TH* ***then***
  TimerCounter = 0
  $gf_i = |o_m|$, where $o_m \in G_i \cap IntO$
  for *every neighbor node* v_k ***do***
    $rf_i(v_k) = |o_m|$, where $o_m \in G_k \cap R_i \cap IntO$
    $q(v_k)=e_{ij} + d_i - rr_i$
  end for
  rf_i=*sum of all* $rf_i(v_k)$
  if $rf_i > gf_i$ ***then***
    put all neighbor v_k *in array QA in ascending order of* $q(v_k)$
    index = 0
    while $rf_i > gf_i$ ***do***
      index++
      $rf_i = rf_i - rf_i$ (*QA[index]*)
    end while
    $d_i = q$(*QA[index]*)
  else
    d_i=0
  end if
  broadcast new d_i *to all neighbor nodes*
else
  TimerCounter = TimerCounter + 1
  Collect all $o_m \in G_i \cup R_i$ *in IntO*
end if

Figure 2. Distributed Consistency Control Method

The whole method includes three stages: initial stage, information collection stage and adjustment stage. In the initial stage, we first set the deviation time of each node to 0. After that, each node first enters the information collection stage, and continuously collects information such as delays and inconsistency degrees of the sending and receiving events. The adjustment stage logic is executed at the end of the period, and the new deviation time is calculated and adjusted. After that, each node enters the next information collection stage.

In the adjustment stage, each node first compares the numbers of the received and sent events collected in the previous period. If the number of received events is less than the number of sent events, it indicates that the node is generating and sending large numbers of events. In this case, a large deviation time will increase the inconsistency of the sending events, resulting in the increasing overall inconsistency of the system. In this case, it is necessary to set the deviation time of this node to zero. If the number of received events is greater than or equal to the number of sending events, it means that the node is mainly receiving the events from other nodes and we can set a proper value of deviation time for this node to optimize the system's inconsistency degree.

When we choose the value of deviation time, the constraint values between neighbor nodes need to be considered, because once the deviation time value exceeds the constraint value of a neighbor node, increasing the deviation time value of this node will not decrease the inconsistency degree of the events from the neighbor node. Based on this consideration, all neighbor nodes are sorted according to the ascending order of their constraint values, and the received event number of each node in the queue are subtracted one by one in the total received event number until the received event number is not greater than the sent event number. The constraint value of the current node in the queue is used as the new deviation time of this node. At this time, if the deviation time is further increased, since the received event number is already equal to or less than the sent event number, the inconsistency degree of the whole system will be increased. After obtaining

the appropriate deviation time, the node broadcasts the new value to other neighbor nodes. The process runs periodically at run-time, and continues to optimize overall system consistency.

In fact, a centralized approach can be used to solve this problem by collecting all the information into a single node and solving the deviation time for each node on a centralized node. However, when the scale of the system is large, the information gathering and calculation process will become the bottleneck of the whole algorithm, which will seriously affect the efficiency of the algorithm. In the next section, we will compare the experimental results of our distributed method with the centralized method.

8. Experimental Results

In order to verify the effectiveness of our method, we construct a simulated distributed virtual environment system with up to 2000 simulated nodes. We use DS^2 tool [18] to generate the inter-node delay and simulate the communication characteristics of the network. DS^2 is a delay synthesis and generation tool based on the delay measurement of real internet communications. It can effectively simulate the delay model of communications through internet. For responsiveness requirements, we designed five levels, ranging from 100ms to 500ms, to simulate the needs for different types of entities.

We run the virtual environment for 10,000 cycles. Every node in each cycle has a probability of 20%-60% to send an event to its neighbor nodes, and we run our distributed consistency control algorithm one time every 100 cycles. The statistical results are shown in Figure 3.

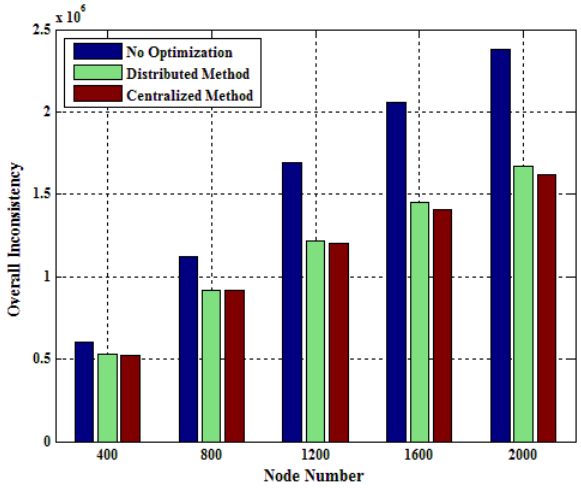

Figure 3. Comparison Results of Inconsistency

In Figure 3, the x-axis represents the number of participating nodes in the virtual environment, varying from 400 to 2000, and the y-axis represents the total system inconsistency throughout the run-time. The blue bars represent the results without using consistency control method. The green bars represent the results obtained by using the consistency control method presented in this paper. The red bars represent the results that all the information are collected by a single node and computed by a centralized way. From the results, we can see that compared with the case without using the consistency control method, our method can effectively reduce the total system inconsistency in all cases from 400 nodes to 2000 nodes. And the more nodes there are in the system, the more optimization we can get from our method. In all cases our method can reduce the overall system inconsistency by at least 15%. Compared with the centralized method, the distributed method in this paper can achieve very close optimization results, the difference

is less than 5%. After that, the performance comparison results between distributed method and centralized method are shown in Figure 4.

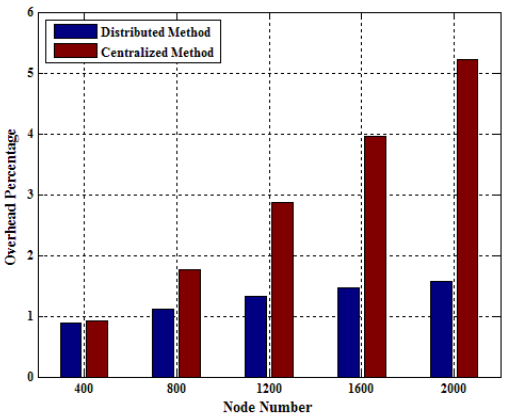

Figure 4. Comparison Results of Overhead

In Figure 4, the x-axis represents the number of participating nodes in the virtual environment, and the y-axis represents the additional time overhead associated with the consistency control approach at run-time. The blue bars represent the distributed consistency control method presented in this paper, while the red bars represent the centralized consistency control method. It can be seen from figure that with the increase of the scale of the system, the overall time overhead of the centralized method is increasing very fast due to the need of collecting a large amount of information and carrying out a large number of solving processes at one node. But the overhead of our distributed consistency control method are not increased fast when the scale of the system increases. So our method has better scalability.

The above experiments show that the asynchronous consistency control method proposed in this paper can reduce the overall inconsistency of the system under the premise of guaranteeing the system responsiveness, so it improves the usability of the DVE system.

9. Conclusion

Due to the difference of the object's natural attributes and the differences of the user's operating habits in the distributed virtual environment system, the responsiveness requirements of the nodes in the system are not the same. However, the existing consistency control method usually neglects this characteristic and adopts a simple responsiveness requirement model, which is hard to match the actual demand of the DVE system. To solve this problem, in this paper we proposed a distributed consistency control method for multi-level responsiveness requirements. By adjusting the distribution of time resources on the participating nodes, the consistency state of the system is optimized. The overall system inconsistency can be effectively reduced and the usability of the system can be improved. The experimental results verify the effectiveness of the proposed method.

The consistency metric model proposed in this paper is based on the assumption that the multi-level responsiveness requirement of each node of the system can be fully satisfied. The main consideration is the deviation between the real execution time of the event and the expected execution time. However, in some distributed virtual environments, consistency need is much more important than the responsiveness requirement need. We will consider how to accommodate the satisfy degree of the multi-level responsiveness

requirements in consistency metric model in our future work, so that the proposed method has better adaptability.

Acknowledgments

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

References

[1] A. Valadares, E. Gabrielova and C. V. Lopes, "On designing and testing distributed virtual environments", Concurrency and Computation: Practice and Experience, **(2016)**.
[2] R. M. Fujimoto, "Parallel and distributed simulation systems", New York: Wiley, **(2000)**.
[3] Y. H. Tang, B. D. Zhang, J. J. Wu, *et al.*, "Parallel architecture and optimization for discrete-event simulation of spike neural networks", Science China-Technological Sciences, vol.56, no.2, **(2013)**, pp. 509-517.
[4] B. Hou, Y. Yao, B. Wang B, *et al.*, "Modeling and simulation of large-scale social networks using parallel discrete event simulation", Simulation-Transactions of The Society for Modeling and Simulation International, vol.89, no.10, **(2013)**, pp. 1173-1183.
[5] T. A. Funkhouser, "RING: a client-server system for multi-user virtual environments", Proceedings of the 1995 symposium on Interactive 3D graphics, ACM, **(1995)**, pp. 85-ff.
[6] B. Di Chen and M. Maheswaran, "A fair synchronization protocol with cheat proofing for decentralized online multiplayer games", Proceedings on Third IEEE International Symposium on Network Computing and Applications 2004 (NCA 2004), IEEE, **(2004)**, pp. 372-375.
[7] M. Mauve, J. Vogel, V. Hilt, *et al.*, "Local-lag and timewarp: providing consistency for replicated continuous applications", IEEE transactions on Multimedia, vol.6, no.1, **(2004)**, pp. 47-57.
[8] L. Gautier, C. Diot and J. Kurose, "End-to-end transmission control mechanisms for multiparty interactive applications on the internet", INFOCOM'99, Proceedings of the Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies, IEEE, vol.3, **(1999)**, pp. 1470-1479.
[9] W. Zhang, H. J. Zhou, Y. X. Peng, *et al.*, "Asynchronous time consistency control methods in distributed interactive simulation", Journal of Software, vol.21, no.6, **(2010)**, pp. 1208-1219.
[10] W. Zhang, H. Zhou, Y. Peng, *et al.*, "An Interval Consistency Control Method Based on Estimating Network Delay in DVE Systems", Computer Engineering & Science, vol.34, no.3, **(2012)**, pp. 55-61.
[11] W. Cai, S. J. Turner, B. S. Lee, *et al.*, "An alternative time management mechanism for distributed simulations", ACM Transactions on Modeling and Computer Simulation (TOMACS), vol.15, no.2, **(2005)**, pp. 109-137.
[12] S. P. Hernandez, J. Fanchon and K. Drira, "The immediate dependency relation: an optimal way to ensure causal group communication", Annual Review of Scalable Computing, vol.6, no.3, **(2004)**, pp. 61-79.
[13] L. Lamport, "Time, clocks, and the ordering of events in a distributed system", Communications of the ACM, vol.21, no.7, **(1978)**, pp. 558-565.
[14] X. Qin, "Delayed consistency model for distributed interactive systems with real-time continuous media", Journal of Software, vol.13, no.6, **(2002)**, pp. 1029-1039.
[15] W. Zhang and H. Zhou, "An Asynchronous Control Method for Reducing Inconsistency in DVE Systems", To appear in JIMET 2017, **(2017)**.
[16] S. Zhou, W. Cai, B. S. Lee, *et al.*, "Time-space consistency in large-scale distributed virtual environments", ACM Transactions on Modeling and Computer Simulation (TOMACS), vol.14, no.1, **(2004)**, pp. 31-47.
[17] W. Zhang, H. Zhou, Y. Peng, *et al.*, "Providing Responsiveness Requirement Based Consistency in DVE", 2009 15th International Conference on Parallel and Distributed Systems (ICPADS), IEEE, **(2009)**, pp. 594-601.
[18] B. Zhang, T. S. Ng, A. Nandi, *et al.*, "Measurement based analysis, modeling, and synthesis of the internet delay space", Proceedings of the 6th ACM SIGCOMM conference on Internet measurement, ACM, **(2006)**, pp. 85-98.
[19] M. A. Bassiouni, M. H. Chiu, M. Loper, *et al.*, "Performance and reliability analysis of relevance filtering for scalable distributed interactive simulation", ACM Transactions on Modeling and Computer Simulation (TOMACS), vol.7, no.3, **(1997)**, pp. 293-331.
[20] B. Blau, C. E. Hughes, M. J. Moshell, *et al.*, "Networked virtual environments", Proceedings of the 1992 symposium on Interactive 3D graphics, ACM, **(1992)**, pp. 157-160.
[21] L. A. H. Liang, W. Cai, B. S. Lee, *et al.*, "Performance analysis of packet bundling techniques in DIS", Proceedings of the 3rd IEEE International Workshop on Distributed Interactive Simulation and Real-Time Applications 1999, IEEE, **(1999)**, pp. 75-82.
[22] R. M. Fujimoto and R. M. Weatherly, "Time management in the DoD high level architecture", ACM SIGSIM Simulation Digest, IEEE Computer Society, vol.26, no.1, **(1996)**, pp. 60-67.

