http://ansinet.com/itj

ITJ

ISSN 1812-5638

INFORMATION TECHNOLOGY JOURNAL

ANSInet
Asian Network for Scientific Information
308 Lasani Town, Sargodha Road, Faisalabad - Pakistan

Information Technology Journal 12 (20): 5941-5946, 2013
ISSN 1812-5638 / DOI: 10.3923/itj.2013.5941.5946

A Real-time Causal Order Control Approach in a Large-scale DVE System with Asynchronous Clocks

[1,2]Hangjun Zhou, [1]Zhongli Liu, [1]Sha Fu and [1]Guang Sun
[1]Department of Information Management, Hunan University of Finance and Economics, Chang Sha, Hu Nan, 410205, People's Republic of China
[2]School of Mechanical Engineering, Nanjing University of Science and Technology, Nan Jing, Jang Su, 210094, People's Republic of China

Abstract: In a large-scale peer-to-peer DVE (Distributed Virtual Environment) system running on the WAN (Wide Area Network) without the central control node, it is one of the most challenging problems to preserve the causality of all the simulation events on each node with asynchronous clock. However, the most existing asynchronous causality control methods seem to be inefficient to resolve the problem for applying either too closely coupled causal control information with system scale to achieve the real-time preservation or too little control information in a message to maintain the correct causality. In this paper, a novel causal order control approach with the deduction of the minimal calculable moment and the distributed preservation algorithm is proposed to achieve the correct cause-effect relationship among all the simulation events on each node. The experimental results demonstrate that comparing with the representative existing methods the proposed approach can cost lower transmission and calculation overhead with proper amount of causal control information in a message and ensure the real-time preservation of causality with asynchronous clocks.

Key words: Distributed virtual environment, asynchronous clock, causal order, distributed preservation algorithm

INTRODUCTION

One of the most remarkable differences between a peer-to-peer fully distributed DVE system and a server-client DVE system is that the former doesn't have a central controlling and coordinating server. Therefore, the distributed nodes or sites in the former one are autonomous and equal to one another. Each node can solely communicate with the remote ones by directly exchanging messages. Any site in the decentralized DVE system doesn't have the shared memory with another one (Fujimoto, 2000; Plch *et al*., 2013).

As the fast development of network and virtual reality technologies, the decentralized network systems are increasingly established on WAN so as to satisfy the application demands. What is more, the scale of distributed sites in a DVE system expands significantly which brings about the frequent data exchanging and message transmission in the network. Due to the geographical dispersity of the participant sites in a network system running on WAN, the time advance of each site is usually asynchronous with one another (Fujimoto, 2000).

Because of the large transmission latency of WAN, it is probable that the distributed sites receive messages in different orders which may not be identical with the sending order of the messages. If the decentralized network systems are sensitive to the causality of the delivery order of messages, the causal order violations might happen without the proper control mechanisms (Cai *et al*., 2005; Schwarz and Mattern, 1994). Thereby, it is necessary to ensure the causal order delivery for this kind of systems. Furthermore, in the case that they are real-time network systems, the data in each message has the valid time limitation so that the causal order delivery of messages must be accomplished in real time at each site. Currently, this is one of the most challenging problems for the decentralized DVE systems on WAN.

However, the most existing causality control mechanisms seem to be inefficient to resolve the problem. Immediate Dependency Relation (IDR) (Hernandez *et al*., 2004; Dominguez *et al*., 2008) provides the real-time message ordering delivery but if there is any cause message could not arrive at the destination sites within the valid time of the received effect message, it is hard for IDR to maintain the causality of received messages.

Corresponding Author: Hangjun Zhou, Department of Information Management, Hunan University of Finance and Economics, Chang Sha, Hu Nan, 410205, People's Republic of China

Vector Time (Raynal *et al*., 1991; Raynal, 2013) method, such as SES, can select adequate control information for each message to ensure the causality delivery, however, the amount of the control information is closely coupled with the system scale so that it cannot meet the real-time requirement in the large-scale systems on WAN. Δ-Causal Order (Baldoni *et al*., 1998) is another real-time method but it assumes that every message has the same valid time "Δ" and the time advances of all sites are perfectly synchronized.

In this article, we analyze the fundamental reasons of the problem and propose an original causality communication algorithm based on discerning the minimal calculable moment during the course of the asynchronous time advances at the message receiving sites. The results of groups of experiments demonstrate that our algorithm has a low overhead with respect to the amount of control information appended to each message and ensures that the received messages are delivered with causality in real time.

The rest of the paper is organized as follows. Section 2 presents the system model of a decentralized DVE network. The minimal calculable moment is analyzed and defined in Section 3. The new distributed causality preservation algorithm is proposed in Section 4. Experiments are implemented to evaluate the performance of our algorithm compared with the existing ones in Section 5. Section 6 concludes the article.

SYSTEM MODEL

We assume that a decentralized network system comprising n equal sites: $P = \{p_1, p_2,..., p_n\}$. The occurrence of actions performed at a site is called events which are assumed to be atomic. An event is sent to other related sites through message transmission. For simplicity, sometimes sending (receiving) a message is also expressed as sending (receiving) an event.

For any event e_x generated at site p_i, we use $r(e_x) = (i, a)$ as its identification where i is the number of the site and a is the logical time when ex generates. $t_i x$ is the local time when pi can identify ex, i.e., if ex is generated at pi, $t_i x$ is the time when pi generates ex; if ex is generated at some remote site and received by p_i, $t_i x$ is the time when p_i receives e_x.

MINIMAL CALCULABLE MOMENT

In decentralized DVE systems, the causality of events is defined by the "happened before" relation (Lamport, 1978) which is denoted as "¬".

Definition 1:
"Happened before" relatio

- If e_1 and e_2 are events in the same sites and e_1 comes before e_2, then $e_1 \neg e_2$
- If e_1 is the sending event of a message by one site and e_2 is the receipt event of the same message by another site, then $e_1 \neg e_2$
- If $e_1 \neg e_2$ and $e_2 \neg e_3$, then $e_1 \neg e_3$

For two distinct events e_1 and e_2, if both $e_1 \neg e_2$ and $e_2 \neg e_1$ are unavailable, then e_1 and e_2 are said to be concurrent.

With the "happened before" relation, the causality of events could be described in the cause-effect graph. For instance, e_1, e_2, e_3, e_4 and ey are the events of causality. The cause-effect graph of them is shown in Fig. 1. Obviously, this graph is a directed graph.

As can be seen, the causality of the events in Fig. 1 is mainly consisting of the cause-effect relation on tow directed paths: $e_1 \neg e_2 \neg e_3 \neg e_y$ and $e_1 \neg e_4 \neg e_y$. Because e_2 and e_4 are concurrent which is the same to the relation of e_3 and e_4, Fig. 1 can be identically transformed to Fig. 2 as far as the cause-effect relation is concerned.

It is not difficult to understand that as long as the causality of events in each directed cause-effect path is well preserved, the causality in the whole cause-effect graph is ensured. Therefore, we intend to find out the proper way to maintain the "happened before" relation of events on a directed path in the rest of this section.

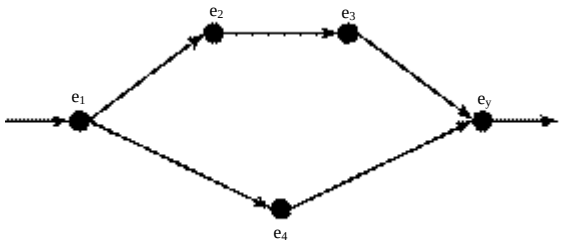

Fig. 1: An example of the cause-effect graph

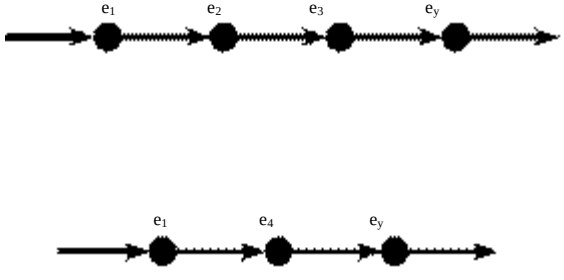

Fig. 2: Two directed cause-effect paths

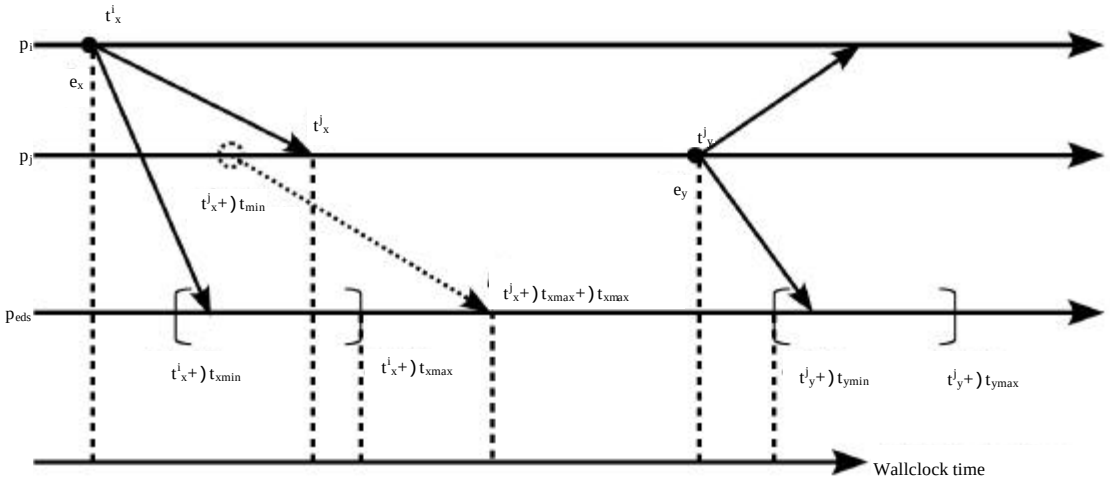

Fig. 3: Minimal calculable moment of tix at p_j

Due to the large transmission latency on WAN, the cause events may not arrive at the destination sites within the valid time of the received effect event. In order to keep the "happened before" relation of received events in this case, a sending site should select the causal control information of cause events one by one along the reversed direction of a directed cause-effect path. Thus, even though there are some lately arrived cause events, the receiving sites could use the causal control information of the effect to identify the nearest cause on a path among the received events and keep the causal order delivery.

However, during the course of selecting the control information at a sending site, when should the site terminate the selection? Before answer this question, a conception should be defined at first.

Definition 2

Minimal calculable moment: $\forall e_x, e_y$, if e_x is generated at p_i and e_y is generated at p_j and $e_x \rightarrow ey$, then the minimal moment in the time advance of pj which is equal or later than tix in physical time and can be identified by p_j, is called the minimal calculable moment of t^i_x at p_j.

In a decentralized DVE system, the arriving range of ey sent from p_j toward different destination sites could be denoted as $[t^j_y+\Delta t_{ymin}, t^j_y+\Delta t_{ymax}]$, where Δt_{ymin} is the minimal transmission time of ey in the network and Δt_{ymax} is the maximal transmission time of ey. Δt_{ymin} and Δt_{ymax} could be calculated by the decentralized network coordinate system (Dabek *et al*., 2004; Agarwal and Lorch, 2009). Thus, an example of the minimal calculable moment is illustrated in Fig. 3. For t^j_x is the arriving time of e_x at p_j, the minimal calculable moment tix of should be $t^j_x-\Delta t_{xmin}$ currently which is the earliest and identifiable moment later than tix in the time advance of p_j. Evidently, $t^j_x-\Delta t_{xmin}+\Delta t_{xmax}$ is later than $t^i_x+\Delta t_{xmax}$ in physical time (also expressed as wallclock time [1]). Thereby, if $(t^j_x-\Delta t_{xmin}+\Delta t_{xmax}) \leq (tjy+\Delta t_{ymin})$ is achieved on a directed path of ey, it would indicate that there is an ex that arrives all destination sites, denoted as pdes, earlier than e_y. If each cause of the events between e_x and e_y on the path is delayed by the network latency in the worst case, pdes could utilize the control information of e_x to ensure delivering e_x and e_y in causal order. Hence, when the information of this kind of e_x is selected, p_j can terminate the selection.

The use of minimal calculable moment $(t^j_x-\Delta t_{xmin})$ is better than use of t^j_x for the less transmission overhead with respect to the amount of the control information for each event.

DISTRIBUTED CAUSALITY COMMUNICATION ALGORITHM

With the minimal calculable moment, we design and propose a distributed causality preservation algorithm containing two sub-algorithms in this section to preserve causality of events on the whole cause-effect graph. At First, some data structures at one site need to be introduced.

- **LT(p_j):** The vector to record the current logical time
- **CM(p_j):** The adjacency list recorded by p_j to denote the cause-effect relation of delivered events. There are five variables: $(r(e_x), t^j_x, \Delta t_{xmin}, \Delta t_{xmax}, p\_cs[num])$ in an element of CM(p_j). e_x is a cause of e_y and p cs[num] is a pointer array in which each pointer points to an element that is corresponding to an im- mediate dependent event of e_x. CM(p_j) would be periodically updated by p_j

Sub-algorithm 1: Send and event eij in a message M

Input: cij,)t_{ymin}, t_{ymax}
Output: M
1: LT (p_j)[j] = LT(p_j)[j]+1
2: t^j_y = time get time()
3: r(e_y) = (j, LT (p_j)[j])
4: iter = 0
5: for all the e_x that is the immediate cause of e_y do
6: If !find ln (CM (r(e_x)) then
7: p_cs[iter] = null;
8: else
9: p_cs[iter = find lnCM (r(e_x))
10: end of
11: iter++
12: end for
13: ptr = Cn(p_j).add(r(e_y), t^j_y, Δt_{ymin}, Δt_{ymax}, p_cs[num])
14: Selection (null, null, ptr, t^j_y)t_{ymin}); // select control information for e_y
15: M←(ey, CB(e_y))
16: Send Message (M)
17: CB(e_y). Clear ()
18: exit()

Fig. 4: Sending message sub-algorithm 1

- **CB(e_y):** The adjacency list to record the causal control information of e_y. Each element in CB(e_y) also has five variables: (r(e_x), t^j_x, Δt_{xmin}, Δt_{xmax}, p_cs[num]). r(e_x), t^j_x, Δt_{xmin}, Δt_{xmax} could be replicated from the corresponding element in C M (p_j) but the structure p cs[num] needs to be revalued to point to the immediate dependent elements in C B(e_y)
- **MQ(p_j):** The buffering queue of p_j to temporarily store the messages which are received but still not delivered. p_j would periodically check MQ(p_j) to find out whether there are messages that can be deliveredpresently

Based on the above analysis, the real-time and distributed communication algorithm is proposed which mainly includes two parts: Sending a message with control information from p_j and receiving and processing it in accordance with causality at pdes.The sending part is described in the Fig. 4 with Sub-Algorithm 1.

When p_j is going to send e_y to other remote sites, it needs to select causal order control information for e_y which is accomplished in line 14 of Sub-Algorithm 1. The procedure Selection (ptrup, nl, ptrjoin, tjy, Δt_{ymin}) of line 14 is described in Fig. 5.

After the selection, the replication of CB(e_y) would be added into the message M with e_y and sent to pdes. The receiving part is described in Fig. 6 with Sub-Algorithm 2.

The preservation of causal order delivery of received events at pdes is ensured by the procedure depicted in the line 10 of Sub-Algorithm 2 which is stated in Fig. 7.

Input: ptr_{up}_pointer to the element in cb(e_y) that calls this procedure; nl∈[0, num-1]; ptr_{join}-[pointer to the element (r(e_x), t^j_x, Δt_{xmin}, Δt_{xmax}, p_cs[num]) in CM (p_j) joining to CB(e_y) during this calling; t^j_y; Δt_{ymin}
Output: CB(e_y)
1: p = find lnCB(ptr join→r(ex));
2: if p then
3: ptrup→p_cs[nl] = p
4: return true;
5: end if
6: p = CB(e_y).add(r(e_x), t^j_x, Δt_{xmin}, Δt_{xmax}, p_cs[num])
7: if ptrup≠null and nl≠null then
8: ptrup→p_cs[nl] = p
9: if p→(t^j_x-)t_{xmin}+Δt_{xmax}Δ≤t^j_y+Δt_{ymin}) then
10: for all ir such that 0≤ir≤num_1 do
11: p→p_cs [ir] = null
12: end
13: else
14: for all ir such that 0≤Ir≤num-1 do
15: P_{nt} = → p_cs[ir]
16: if P_{nt}≠null then
17: Selection (p, ir, P_{nt}, t^j_y, Δt_{ymin})
18: end if
19: end for
20: end if
21: else
22: for all ir such that 0≤ir≤num-1 do
23: P_{nt} = p→cs[ir]
24: if Pnt≠null then
25: Selection (p, ir, Pnt, tjy, Δt_{ymin})
26: end if
27: end for
28: end if
29: return true

Fig. 5: Causal control information selection procedure

Sub-algorithm 2: Receive M containing e_y and CB(e_y)

Input: M

1: If LT (p_j)[j]??LT(p_{des})[j] then
2: For go msg (M)
3: exit()
4: Else if not all the immediate dependent causes of e_y are delivered at p_{des} then
5: t^{des}_y = time get time()
6: Store Msg (M, t^{des}_y)
7: exit()
8: else
9: t_{desy} = time get time()
10: Process Msg (r(e_y), M, t^{des}_y)
11: end of
12: exit()

Fig. 6: Receiving message sub-algorithm 2

EXPERIMENTS

With the aim to evaluate and verify the performance of the distributed communication algorithm proposed in the above section, several experiments are implemented on the PC cluster in the key laboratory of distributed and parallel processing at our university. We use a Spirent

Input: $r(e_y)$, Mt^{des}_y
1: p = find lnCB($r(e_y)$)
2: if p then
3: ptr = CM(p_{des}).add(P→($r(e_y)$, tiy,)t_{ymin}, (t_{ymax}, p_cs[num]))
4: else
5: return false
6: end if
7: iter = 0
8: For all the immediate ex of e_y such that $r(e_x)$ = (i, s) do
9: If $r(e_x).a \leq LT(P_{des})[r(e_x).i]$ then
10: ptr→p_cs[iter] = Find lnCM($r(e_x)$)
11: else
12: M_x = Find ln MQ ($r(e_x)$)
13: If Mx then
14: ptr→p_cs[iter] = Process Msg ($r(e_x)$, M_x, t_x^{des})
15: else
16: ptr→p_cs[iter] = Process Msg ((re_x), M, t_y^{des})
17: end if
18: end if
19 end for
20: if $LT(p_{des})[j] < LT9(p_j)[j]$ then
21: $LT(p_{des})[j] = LT(p_j)[j]$
22: end if
23: Process event (e_y)
24: Return ptr

Fig. 7: Causal order preservation procedure

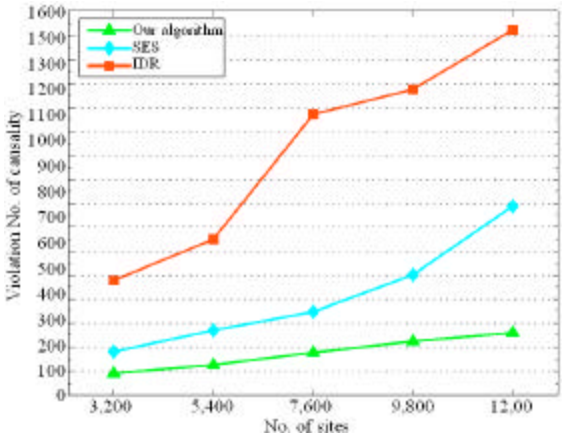

Fig. 8: Violation numbers of causality

ConNIE (Converged Network Impairment Emulator) to really emulate the impairments of the message transmission delay on WAN. In the decentralized network system, the time advance of each site is set to be asynchronous with one another.

What's more, an air battle simulation system is developed as test instance which is applied with different scales: 3200, 5400, 7600, 9800, 12000. To compare with our algorithm in the experiments of asynchronous time advances of all sites, we chose the typical asynchronous causal control algorithms: SES and IDR. The mean network delay time is conFig.d as 200 ms.

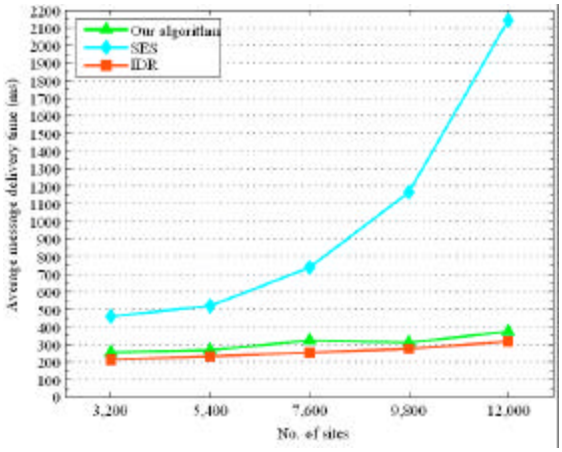

Fig. 9: Average message delivery time

As we can see, the violation numbers of causality of events under different scales of sites is shown in Fig. 8. Because of the lack of sufficient causal control information, the violations of IDR are the most in each scale. Due to the large message transmission amount, the violation numbers of SES under message valid time limitation increases a lot with the expanding of number of sites. The violations of our algorithm are less than those of the other two in the experiments.

Figure 9 illustrate the average message delivery time of these three algorithms in the experiments with different scales of sites. For the large amount of transmission overhead, the increment of the average message delivery time of SES raises significantly as the number of sites expands. The amount of the control information of our algorithm is irrelevant to the system scale, therefore, the average message delivery time of it is similar to or merely a little bit higher than that of IDR in different experiments.

CONCLUSION

As for a decentralized DVE system running on WAN, it is a challenging problem to preserve the causality of received events with real-time property at each site. In this article, a causal order control approach consisting of two sub-algorithms based on the minimal calculable moment is proposed to resolve the problem. The algorithm could select the effective causal control information which merely costs small transmission overhead and has the good scalability. Experimental results and analysis demonstrate that through applying our algorithm, the sites in the system could ensure the causality of events in real-time.

ACKNOWLEDGMENTS

This study work is supported by the Hunan Science and Technology Project (No. 2012FJ6011) and the Construct Program of the Key Discipline in Hunan Province, China.

REFERENCES

Agarwal, S. and J.R. Lorch, 2009. Matchmaking for online games and other latency-sensitive P2P systems. Proceedings of the Conference on Data communication, August 17-21, 2009, Barcelona, Spain, pp: 315-326.

Baldoni, R., R. Prakash, M. Raynal and M. Singhal, 1998. Efficient Ä-causal broadcasting. Int. J. Comput. Syst. Sci. Eng., 13: 263-271.

Cai, W., S.J. Turner, B.S. Lee and J. Zhou, 2005. An alternative time management mechanism for distributed simulations. ACM Trans. Model. Comput. Simul., 15: 109-137.

Dabek, F., R. Cox, F. Kaashoek and R. Morris, 2004. Vivaldi: A decentralized network coordinate system. Proceedings of the Conference on Applications, Technologies, Architectures and Protocols for Computer Communications, August 30-September 3, 2004, Portland, Oregon, USA., pp: 15-26.

Dominguez, E.L., S.E. Hernandez and G.R. Gomez, 2008. MOCAVI: An efficient causal protocol for cellular networks. Int. J. Comput. Sci. Network Security, 8: 136-152.

Fujimoto, R.M., 2000. Parallel and Distributed Simulation Systems. John Wiley & Sons, New York.

Hernandez, S.P., J. Fanchon and K. Drira, 2004. The Immediate dependency relation: An optimal way to ensure causal group communication. Ann. Rev. Scalable Comput., 6: 61-79.

Lamport, L., 1978. Time, clocks and the ordering of events in a distributed system. Commun. ACM., 21: 558-565.

Plch, T., T. Jedlicka and C. Brom, 2013. HLA Proxy: Towards Connecting Agents to Virtual Environments by Means of High Level Architecture (HLA). In: Cognitive Agents for Virtual Environments, Dignum, F., C. Brom, K. Hindriks, M. Beer and D. Richards (Eds.). Springer, USA., ISBN: 978-3-642-36443-3, pp: 1-16.

Raynal, M., 2013. Distributed Algorithms for Message-Passing Systems. Springer-Verlag, Berlin, USA., ISBN 978-3-642-38123-2, Pages: 500.

Raynal, M., A. Schiper and S. Toueg, 1991. The causal ordering abstraction and a simple way to implement it. Inform. Process Lett., 39: 343-350.

Schwarz, R. and F. Mattern, 1994. Detecting causal relationships in distributed computations: In search of the holy grail. Distributed Comput., 7: 149-174.