 CMC, vol.56, no.1, pp.73-89, 2018

A Distributed LRTCO Algorithm in Large-Scale DVE Multimedia Systems

Hangjun Zhou[1, 2, *], Guang Sun[1], Sha Fu[1], Wangdong Jiang[1], Tingting Xie[3] and Danqing Duan[1]

Abstract: In the large-scale Distributed Virtual Environment (DVE) multimedia systems, one of key challenges is to distributedly preserve causal order delivery of messages in real time. Most of the existing causal order control approaches with real-time constraints use vector time as causal control information which is closely coupled with system scales. As the scale expands, each message is attached a large amount of control information that introduces too much network transmission overhead to maintain the real-time causal order delivery. In this article, a novel Lightweight Real-Time Causal Order (LRTCO) algorithm is proposed for large-scale DVE multimedia systems. LRTCO predicts and compares the network transmission times of messages so as to select the proper causal control information of which the amount is dynamically adapted to the network latency variations and unconcerned with system scales. The control information in LRTCO is effective to preserve causal order delivery of messages and lightweight to maintain the real-time property of DVE systems. Experimental results demonstrate that LRTCO costs low transmission overhead and communication bandwidth, reduces causal order violations efficiently, and improves the scalability of DVE systems.

Keywords: Distributed computing, distributed virtual environment, multimedia system, causality violation, causal order delivery, real time.

1 Introduction

With the rapid development of Cloud Computing, Big Data, Artificial Intelligence and Internet technologies, large-scale multimedia systems are increasingly implemented on the Internet and mobile networks. A DVE multimedia system is a kind of distributed computing system that simulates the real world and regards geographically distributed users as a group of sequential processes among which data are communicated solely by messages [Fujimoto (2000); Balci, Fujimoto, Goldsman et al. (2017)]. One important issue in DVE systems is to preserve the causal

[1] Hunan University of Finance and Economy, Fenglin Road, No. 139, Changsha, 410205, China.
[2] Nanjing University of Science and Technology, Xiaolingwei Road, No. 200, Nanjing, 210094, China.
[3] School of Electrical Engineering and Computer Science, Queen Mary University of London, Mile End Rd, London E1 4NS, UK.
[*] Corresponding Author: Hangjun Zhou. Email: zhjnudt@gmail.com.

CMC. doi:10.3970/cmc.2018.02411

order delivery of messages at each process [Lamport (1978); Zhou, Cai and Turner (2007)]. Moreover, since a DVE is a computer-generated virtual world which simulates the real one, messages are requested to be orderly delivered in real time. Presently, how to effectively maintain the real-time causal order delivery of messages is still one of the important and fundamental issues in large-scale DVE systems [Fujimoto (2000); Zhou, Cai and Turner (2007); Evropeytsev, Dominguez, Hernandez et al. (2017)].

Causality of messages has been widely studied in parallel and distributed computing systems. Previously, many distributed causal order control approaches have been proposed. Vector Time [Schwarz and Mattern (1994); Kshemkalyani and Singhal (1998); Cai,Turner and Lee (2005)] and Immediate Dependency Relation (IDR) [Prakash, Raynal and Singhal (1997); Hernandez (2015); Hernandez, Fanchon and Drira (2004)] are two kinds of traditional approaches to keep causal order. They assume data of all messages have unlimited time validity so that they are not mainly researched with real-time issues [Baldoni, Mostefaoui and Raynal (1996); Zhou, Cai and Turner (2007)]. a Δ-Causal Order [Baldoni, Prakash and Raynal (1998); Yavatkar (1992)] is defined for the distributed systems in which data of messages have limited time validity and real-time constraints. However, this approach can merely set each message with the identical time validity Δ, which may not suit for DVE systems [Rodrigues, Baldoni and Anceaume (2000)]. Moreover, because it uses Vector Time to maintain causal order, the amount of control information with each message is closely coupled with the scale of processes. Hence, when in large-scale DVE systems, Δ-Causal Order needs to attach a large amount of control information to each message, which introduces too high overhead of sending, transmitting and receiving messages to preserve causal order delivery in real time. In Rodrigues et al. [Rodrigues, Baldoni and Anceaume (2000)], another causal order algorithm is proposed to take each message's own time validity into consideration, but it still uses Vector Time to keep causal order which limits the scalability and degrades the real-time property of the algorithm. Zhou et al. [Zhou, Cai and Turner (2007)] gives a real-time causal order mechanism to define and reduce critical causal order violations. The control information of each message is irrelevant to the scale of processes by attaching a critical causal pair in it. However, since critical causal order merely preserves the cause-effect relation existing between two immediate dependent events, other cause-effect relations might not be maintained and would change into concurrent relations, which would violate the causal order. In Evropeytsev et al. [Evropeytsev, Dominguez, Hernandez et al. (2017)], a causal protocol CODM is proposed of which the overhead timestamp in each message is based on immediate dependency relation, but it is oriented to a reliable hierarchical overlay network where the direct communication among peers is disabled.

Thus it can be seen that the key challenge is to preserve causal order delivery of messages with the control information of which the transmitting and processing overhead is subject to real-time property in large-scale DVE systems. In this article, we aim to propose a novel Lightweight Real-Time Causal Order (LRTCO) algorithm for large-scale DVE systems. Different to the previous approaches, LRTCO can compute causal control information by the prediction and comparison of transmission delays of messages, and deduce the reasonable termination condition of the computation. Therefore, the causal control information is unconcerned with the scale of processes and dynamically adapted to the network latency variations.

In this way, LRTCO could exclude the redundant data for the control information so that it is efficient to preserve the causality of messages and lightweight to maintain the real-time property of DVE systems. Experimental results demonstrate that the LRTCO algorithm costs low control information overhead and com-munication bandwidth, effectively reduces causal order violation, and is more efficient than the previous approaches in preserving the causal order delivery of messages in real time.

The rest of the paper is organized as follows. Section 2 introduces the formal definition of real-time causal order delivery. A novel distributed LRTCO algorithm to resolve the problem is proposed in Section 3. In Section 4, experiments are implemented to evaluate the efficiency of the algorithm. Conclusions are summarized in Section 5.

2 Real-time causal order delivery of messages

In a DVE multimedia system, geographically distributed users, who connect with each other via LAN/WAN, are usually regarded as a finite set P of n sequential processes $\{p_1, p_2, \ldots, p_n\}$ that do not have shared memory and communicate merely by exchanging messages(events) through the serverless network. Generally, an event generated at the process p_i is denoted as e which can be identified with $r(e)$ where $r(e) = (i, a)$ and a is the logical time [Lamport (1978)] when e is generated at p_i. E_i denotes all the events that have been generated at p_i, V_i denotes all the events that have been received at p_i, and H_i denotes all the events that have been delivered at p_i. $E = \cup_{i=1}^{n} E_i$ denotes all the events generated at P. As a DVE is real-time multimedia system with wallclock time, t is uesed to denote the current wallclock time of the DVE system and t_x to denote the time when event e_x is generated. Base on the above discussions, we have the following definitions.

Definition 2.1 (Aappen-Before Relationship)**.** I A $e_x \in E_i$, $e_y \in E_j$, $r(e_x) = (i, a)$, $r(e_y) = (j, b)$, then $e_x \to e_y$ iff

(1) $i = j \wedge a < b$;

(2) $i \neq j$, e_x is the sending event of a message and e_y is the corresponding receiving event;

(3) $\exists e_z \in E$, and $(e_x \to e_z) \wedge (e_z \to e_y)$.

If $\neg(e_x \to e_y) \wedge \neg(e_y \to e_x)$, then e_x and e_y are concurrent events denoted as $e_x \| e_y$.

Definition 2.2 (Immediate Dependency Relation)**.** If $e_x \in E_i$, $e_y \in E_j$, e_x and e_y are of immediate dependency relation(denoted as $e_x \downarrow e_y$) iff $e_x \to e_y$ and $\forall e_z \in E, \neg(e_x \to e_z \to e_y)$.

Definition 2.3 (Causal Order Delivery)**.** If $e_x \in E_i$, $e_y \in E_j$, $e_x \to e_y$ and e_x, e_y are sent to the same process p_k, e_x must be delivered before e_y at p_k. In this case, we say that there is a causal order between e_x and e_y regarding p_k, and e_x is called a causal predecessor of e_y.

In DVE multimedia systems, because distinct types of messages usually have different validity time values, the validity time of e_y can be denote as ΔT_y. The real-time causal order delivery of events is defined as follows.

Definition 2.4 (Real-Time Causal Order Delivery)**.** $\forall p_{des} \in P$, the real-time causal order delivery of events in the messages at p_{des} is preserved iff

(1) $\forall e_y \in V_{des} - H_{des}$, at the time when $t = t_y + \Delta T_y$ or e_y is required

to be delivered, $\forall e_x \in V_{des} - H_{des}$, if $e_x \to e_y$, e_x must be delivered before e_y at p_{des};

(2) $\forall e_y \in V_{des} - H_{des}$, let $F_y = \{e_x | \forall e_x \in E \wedge e_x \to e_y\}$, when $t_y \leq t \leq t_y + \Delta T_y$, if $F_y \subseteq H_{des}$, e_y must be delivered immediately at p_{des}.

It is denoted as $e_x \xrightarrow{\Delta T} e_y$.

3 A lightweight real-time causal order algorithm

3.1 Analysis and basic idea

In this section, we analyze the basic idea to preserve $e_x \xrightarrow{\Delta T} e_y$ at each p_{des} of e_y base on the following definitions.

Definition 3 .1 (Cause-Effect Relation Graph)**.** $G = (E', D)$ is a cause-effect relation graph iff the ertex et $E' \subseteq E$ and the irected edge set $D=\{d_{(x,y)} | e_x \downarrow e_y, \forall e_x, e_y \in E'\}$, where e_x is the initial ertex and e_y is the terminal ertex of the irected edge $d_{(x,y)}$.

Definition 3.2 (Cause-Effect Relation P ath)**.** If $G = (E', D)$, $\forall e_x, e_y \in E'$, cause-effect relation path $W = (E'', D')$, $E'' \subseteq E'$, $D' \subseteq D$, linking e_x and e_y is a directed sequence of vertices and edges in the form as $w(e_x, e_y) = e_x d_{(x,z)} e_z \ldots e_v d_{(v,y)} e_y$.

$G = (E', D)$ clearly illustrates the cause-effect relation among events in E', but it is not easy to propose the causal order control approach directly based on the graph. With further analysis, it can be found that $\forall e_x, e_y \in E'$, if $e_x \to e_y$ in G, there is bound to be at least one $W = (E'', D')$ linking e_x and e_y; if $e_x || e_y$, there must not be any directed edge or path between e_x and e_y. Therefore, the cause-effect relations merely exist in directed paths. From Definition 1 , we know the delivery order of concurrent events is not considered in causal order control approaches, thus as long as $e_x \xrightarrow{\Delta T} e_y$ of each path is well preserved, $e_x \xrightarrow{\Delta T} e_y$ of the whole graph is equivalently preserved.

Definition 3 .3 (Causal L ength)**.** If $G = (E', D)$, $\forall e_x, e_y \in E'$ such that $e_x \to e_y$, let $W = (E'', D') = w_r(e_x, e_y)$ denote any one of cause-effect relation paths from e_x to e_y in G, thus the causal length between e_x and e_y is

$$L(e_x, e_y) = \max_{r \geq 1} \left(\left| D'_{w_r(e_x,e_y)} \right| \right). \tag{1}$$

Assuming $r(e_y) = (j, b)$, p_j needs to compute the causal control information for e_y, denoted as $CI(e_y)$, to identify $e_x \in V_{des}$ and recursively preserve $e_x \xrightarrow{\Delta T} e_y$ according to the item (1) or (2) in Definition 4. After receiving e_y, if $t_y \leq t \leq t_y + \Delta T_y$, p_{des} would periodically use $CI(e_y)$ to check whether item (2) in Definition 4 is satisfied. In terms with causal order delivery, if $\forall e_x \in E$ such that $e_x \downarrow e_y$, $e_x \in H_{des}$, item (2) in Definition 4 is satisfied, then p_{des} could delivery e_y immediately for better real-time property. However, it is quite possible, especially on WAN, that When $t = t_y + \Delta T_y$ or e_y is required to be delivered, item (2) is not satisfied due to large transmission delay. For this case, it is necessary for p_{des} to use $CI(e_y)$ to reconstruct the cause-effect relation and preserve $e_x \xrightarrow{\Delta T} e_y$ according to item (1) in Definition 4. Then, the current minimum causal event in a path is defined, and a theorem and its proof about causality preservation is given.

Definition 3.4 (Current Minimum Causal Event)**.** $\forall e_y \in E_j, W(E'', D') = w(e_x, e_y)$, the current minimum causal event, denoted as e_m, is the one such that at the current moment t,

$$L(e_m, e_y) = \min_{e_{x'} \in V_{des} \cap E''} \left(L(e_{x'}, e_y) \right). \tag{2}$$

From the definition, it is known that e_m may change with the variation of t and $V_{des} \cap E''$.

Theorem 3.1 $\forall e_y \in E_j$, $W\,(E'',\, D') = w(e_x,\, e_y)$, at the time when $t = t_y + \Delta T_y$ or e_y is required to be delivered, if item (2) in Definition 4 is not satisfied, p_{des} can preserve $e_x \xrightarrow{\Delta T} e_y$ according to item (1) in Definition 4 through using $CI(e_y)$ to identify e_m of $w(e_x, e_y)$ and reconstructing $e_m \downarrow e_y$ relation for $w(e_x, e_y)$.

Proof. When $t = t_y + \Delta T_y$ or e_y is required to be delivered, assume that there is $e_{x'} \in V_{des} \cap E''$, and $e_{x'} \neq e_m, e_{x'} \neq e_y$, then $1 \leq L(e_m, e_y) < L(e_{x'}, e_y)$, hence $e_{x'} \rightarrow e_m$. Because $\forall e_{x''} \in E''$ such that $L(e_{x''}, e_y) < L(e_m, e_y)$, $e_{x''} \notin V_{des}$, thus if p_{des} reconstruct the cause-effect relation between e_m and e_y, it must be $e_m \downarrow e_y$. If p_{des} uses $CI(e_y)$ to identify $e_{x'}$ and reconstruct $e_{x'} \downarrow e_y$, it can get that new $L(e_{x'}, e_y) = 1$. But $L(e_m, e_y)$ isn't renewed, so $L(e_{x'}, e_y) \leq L(e_m, e_y)$, i.e. $e_m \rightarrow e_{x'}$. That violates the original causal order between $e_{x'}$ and e_m. Therefore, the correct way for p_{des} is to reconstruct $e_m \downarrow e_y$ relation, and after that the new $L(e_m, e_y) = 1$. Recursively, p_{des} uses $CI(e_m)$ to identify the current minimum causal event $e_{m'}$ of e_m and renew $L(e_{m'}, e_y)$. The rest may be deduced by analogy until the events in $V_{des} \cap E''$ are identified. Then, p_{des} would require these events to be delivered in causal order so as to preserve $e_x \xrightarrow{\Delta T} e_y$ according to item (1) in Definition 4. Hence, the theorem. ■

Furthermore, if $CI(e_y)$ can be used to identify e_m for item (1) in Definition 4, it can also be used to identify the special e_m for item (2) in Definition 4. Thereby, the focus is the way p_j constructs $CI(e_y)$ so that p_{des} could utilize it to find out $e_m \in V_{des}$ to preserve causal order delivery. Meanwhile, the amount of $CI(e_y)$ greatly affects the transmission overhead so it also needs to be considered particularly. If p_j selects too much $r(e_x)$ into $CI(e_y)$, such as all the $r(e_x)$ in the causal history of e_y in a extremely case, it is bound for p_{des} to identify e_m with $CI(e_y)$, but it also destructs the real-time property of DVE multimedia systems due to the huge transmission overhead. On the other side, if p_j selects not enough $r(e_x)$ into $CI(e_y)$, it may not sufficient for to p_{des} to identify e_m. Therefore, in order to preserve $e_x \xrightarrow{\Delta T} e_y$, the problem is about how to identify e_m at each p_{des} with the proper amount of $CI(e_y)$ selected by p_j.

3.2 Selection mode of causal control information

For the purpose of dynamically selecting effective $CI(e_y)$ adapted to the network latency variation, it might predict the round-trip transmission delay according to the distributed network coordinate algorithm in Dabek et al. [Dabek, Cox and Kaashoek (2004); Agarwal and Lorch (2009)]. The equation to compute and update the network coordinate is as follows.

$$X_j = X_j + \delta \times (rtt - \|X_j - X_i\|) \times u(X_j - X_i). \tag{3}$$

X_j and X_i respectively denote the distributed network coordinates of p_j and p_i. δ is an adaptive timestep. $u(X_j - X_i)$ is an unit vector giving the direction of the force on p_j. rtt (round-trip time) is the the round-trip latency between p_j and p_i. $\|X_j - X_i\|$ is the distance between the coordinates of p_j and p_i in the chosen coordinate space and it could be regarded as the current round-trip network latency between p_j and p_i. During the course of communications at the application layer, it could approximately admit that the one-way transmission delay Δt_{ji} from p_j to p_i is half of the round-trip one, i.e. $\Delta t_{ji} = \|X_j - X_i\|/2$. However, Δt_{ji} is peer-to-peer, which is merely enough for p_j to compute the $CI(e_y)$ solely effective at p_i. In this way, if e_y is not just sent to one destination process, p_j needs to compute the control information for each p_{des} respectively, and then merge to form $CI(e_y)$, which computes repeatedly in a great deal. In order to speed up the computation course, the unnecessary computing overhead of $CI(e_y)$ needs to be eliminated.

With further analysis, it can be found that the transmission time of e_y should have a certain range, i.e. through prediction, p_j can maintain a transmission time range:$[\Delta t_{min}, \Delta t_{max}]$, where Δt_{min} indicates the minimum transmission time of e_y from p_j to p_{des} and Δt_{max} indicates the maximum transmission time similarly. Thus, $[\Delta t_{min}, \Delta t_{max}]$ includes all the transmission times of e_y. As e_y is sent at time t_y, $[t_y + \Delta t_{min}, t_y + \Delta t_{max}]$ denotes the time range including all the times when e_y arrives at p_{des}. Then, when p_j computes $CI(e_y)$ based on the time range, it needs not to respectively compute the control information for each p_{des}. Instead, p_j can select $CI(e_y)$ suitable for all p_{des} to preserve $e_x \xrightarrow{\Delta T} e_y$ at a time, which is beneficial to enhance the real-time property.

$\forall e_y \in E_j, W(E'', D') = w(e_x, e_y)$, p_j may use the arriving time ranges $[t_y + \Delta t_{min}, t_y + \Delta t_{max}]$ and $[t_x + \Delta t_{xmin}, t_x + \Delta t_{xmax}]$ to select proper $CI(e_y)$ with which p_{des} could identify $e_m \in V_{des} \cap E''$. However, when $t = t_y + \Delta T_y$ or e_y is required to be delivered, $V_{des} \cap E''$ of one p_{des} may be not identical with that of the other p_{des} in that the late events may be different at each p_{des} due to different transmission delays. Therefore, p_j should select this kind of $CI(e_y)$ with which each p_{des} could identify its own e_m from its own $V_{des} \cap E''$.

Definition 3.5 (Immediate Dependency Relation Reconstructibility). $\forall e_y \in E_j, W(E'', D')$ $=w(e_x, e_y)$, $|CI(e_y)| = h$, i.e. h is the number of elements in $CI(e_y)$, $x' \in [2, h]$, if the cause-effect relation among e_y and all the events that can be identified by $CI(e_y)$ has the form of $e_1 \downarrow e_2 \downarrow \ldots \downarrow e_h \downarrow e_y$ and satisfies

$$\begin{cases} t_1 + \Delta t_{1max} \leq t_y + \Delta t_{min} & e_1 \downarrow e_y \wedge h = 1 \\ (t_1 + \Delta t_{1max} \leq t_y + \Delta t_{min}) \wedge (t_{x'} + \Delta t_{x'max} > t_y + \Delta t_{min}) & \neg e_1 \downarrow e_y \wedge h > 1 \end{cases}$$

$CI(e_y)$ has the immediate dependency relation reconstructibility.

Theorem 3.2 $\forall e_y \in E_j, W(E'', D') = w(e_x, e_y)$, $|CI(e_y)| = h$, if the selected $CI(e_y)$ has the immediate dependency relation reconstructibility, each p_{des} can use $CI(e_y)$ to iden-tify its own e_m from its own $V_{des} \cap E''$.

Proof. Assume that the time when $t = t_y + \Delta T_y$ or e_y is required to be delivered at p_{des} is denoted as t' and $\forall x' \in [1, h]$,the time $e_{x'}$ arrives p_{des} is denoted as $t_{x'}^{des}$. Because $CI(e_y)$ has the immediate dependency relation reconstructibility, if $h > 1 \wedge \neg(e_1 \downarrow e_y)$, $\forall x' \in [2, h], t_{x'} + \Delta t_{x'max} > t_y + \Delta t_{min}$. For $e_1 \downarrow e_2 \downarrow \ldots \downarrow e_h \downarrow e_y$, $e_{x'} \rightarrow e_y$. Thus $t_{x'} + \Delta t_{x'min} < t_y + \Delta t_{min}$. Thereby, $[t_{x'} + \Delta t_{x'min}, t_{x'} + \Delta t_{x'max}] \cap [t_y + \Delta t_{min}, t_y + \Delta t_{max}] \neq \Phi$. That is to say, it's possible that $t_{x'}^{des} \geq t_y + \Delta t_{min}$ at some p_{des}. In this case, if $(t_{x'}^{des} \leq t') \wedge (L(e_{x'}, e_y) = 1 \parallel (L(e_{x'}, e_y) > 1 \wedge (\forall x'' \in (x', h], t_{x''}^{des} > t')))$, $e_m = e_{x'}$ would be achieved with $CI(e_y)$ at p_{des}. Otherwise if $t_{x'}^{des} > t'$, $e_{x'} \notin V_{des} \cap E''$ at t', so $e_m \neq e_{x'}$. If $\forall x' \in [2, h], t_{x'}^{des} > t'$, e_1 needs to be considered to identify e_m. For $t_1 + \Delta t_{1max} \leq t_y + \Delta t_{min} \wedge e_1 \rightarrow e_y$, $[t_1 + \Delta t_{1min}, t_1 + \Delta t_{1max}] \cap [t_y + \Delta t_{min}, t_y$ $+\Delta$tmax] = Φ.

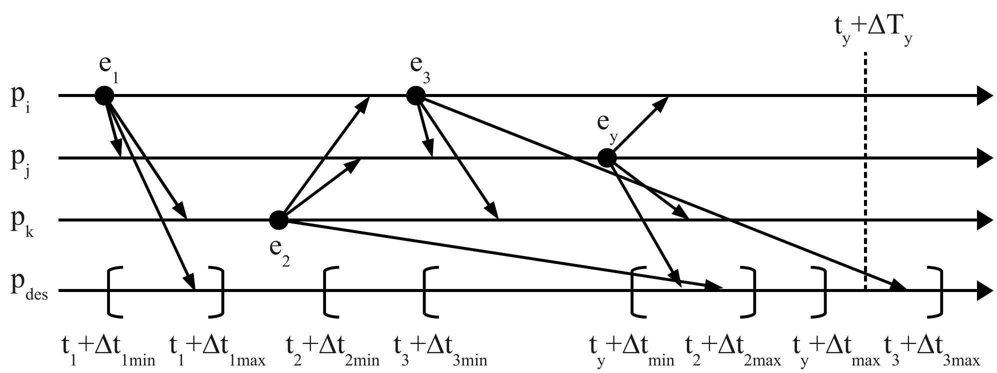

Figure 1: p_j selects $CI(e_y)$ through the comparison of time ranges

Furthermore, $t' \geq t_y + \Delta t_{min}$, thus $t_1 + \Delta t_{1max} \leq t'$, i.e. $e_1 \in V_{des} \cap E''$at t', which is similar to the case when $h = 1 \wedge e_1 \downarrow e_y$. Therefore, at any p_{des}, if

$t_{x'}^{des} > t' \parallel (h = 1 \wedge e_1 \downarrow e_y)$, $e_m = e_1$ can be achieved with $CI(e_y)$. Hence, the theorem. ■

Theorem 3.3 $\forall e_y \in E_j$, $W\ (E'', D') = w(e_x, e_y)$, $|CI(e_y)| = h$, if $CI(e_y)$ has the immediate dependency relation reconstructibility, p_j selects no redundant control information into $CI(e_y)$.

Proof. Assume that the time when $t = t_y + \Delta T_y$ or e_y is required to be delivered at p_{des} is denoted as t' and $\forall x' \in [1, h]$,the time $e_{x'}$ arrives p_{des} is denoted as $t_{x'}^{des}$. For $CI(e_y)$ has the immediate dependency relation reconstructibility, if $h = 1 \wedge e_1 \downarrow e_y$, e_1 is the solely event that can be identified with $CI(e_y)$ and $e_m = e_1$. Obviously, in this case there is no redundant control information in $CI(e_y)$. If $h > 1 \wedge \neg(e_1 \downarrow e_y)$, assume that $\exists x' \in [1, h]$, $r(e_{x'})$ is redundant in $CI(e_y)$. However, it is possible that $(t_{x'}^{des} \leq t') \wedge (L(e_{x'}, e_y) = 1 \parallel (L(e_{x'}, e_y) > 1 \wedge (\forall x'' \in (x', h], t_{x''}^{des} > t')))$ at some p_{des}, thus $e_m = e_{x'}$ is achieved with $CI(e_y)$. Thereby, $r(e_{x'})$ is indispensable in $CI(e_y)$. Furthermore, because $t_1 + \Delta t_{1max} \leq t_y + \Delta t_{min}$, $e_m = e_1$ can be achieved if $\forall x' \in [2, h], t_{x'}^{des} > t'$ at any p_{des}. Therefore, any other $r(e_{x''})$ such that $L(e_{x''}, e_y) > L(e_1, e_y)$ is redundant or unnecessary for $CI(e_y)$. Hence, the theorem. ■

For example, in the scenario shown Fig. 1, $w(e_1, e_y) = e_1 \rightarrow e_2 \rightarrow e_3 \rightarrow e_y$, $t_3 + \Delta t_{3max} > t_y + \Delta T_y$, e_3 may be delayed at some p_{des}. Thus if p_j only selects $r(e_3)$ into $CI(e_y)$, it's not enough to identify e_m at all p_{des}. For $[t_2 + \Delta t_{2min}, t_2 + \Delta t_{2max}] \cap [t_y + \Delta t_{min}, t_y + \Delta t_{max}] \neq \Phi$, if e_2 could arrive in time, p_j may select $r(e_2)$ into $CI(e_y)$. But if e_2 would arrive late either, $e_m \neq e_2$ and $CI(e_y)$ containing $r(e_2)$ and $r(e_3)$ remains not enough for all p_{des}. Then because $[t_1 + \Delta t_{1min}, t_1 + \Delta t_{1max}] \cap [t_y + \Delta t_{min}, t_y + \Delta t_{max}] = \Phi$, e_1 could arrive timely so that even if e_2, e_3 are late, $e_m = e_1$ at p_{des}. Therefore, when $CI(e_y)$ contains $r(e_1)$, $r(e_2)$ and $r(e_3)$, each p_{des} would identify its own e_m and p_j could terminate the selection.

$\forall e_y \in E_j, W(E'', D') = w(e_x, e_y), \forall e_{x'} \in E'', x' \neq y$, before sending e_y, p_j would select $CI(e_y)$ according to the ascending order of $L(e_{x'}, e_y)$ until $CI(e_y)$ has the immediate dependency relation reconstructibility, which is the termination condition of $CI(e_y)$ selection. Correspondently, p_{des} could identify the e_m, whether for item (1) or item (2) in Definition 4, by reading and searching $CI(e_y)$ in the same ascending order. Generally, this kind of computing time overhead is negligible as compared to that of transmission delay. Moreover, because each $W(E'', D')$ in a $G = (E', D)$ can be traversed with Depth-First Search algorithm, the above selection mode and searching method on $W(E'', D')$ are also suitable for the whole $G = (E', D)$. Then, the definition of $CI(e_y)$ of $G = (E', D)$ could be described as follows.

Definition 3.6 (Causal Information on Cause-Effect G raph). $\forall\, e_y \in E_j$,$G = (E', D)$ is the cause-effect graph of e_y, $e_x \in E'$, the causal control information of e_y on $G = (E', D)$ is that $CI(e_y) = \{element_{(i,a)} | element_{(i,a)} = (r(e_x), t_x, [\Delta t_{xmin}, \Delta t_{xmax}], L(e_x, e_y))\}$, where $r(e_x) = (i, a)$ and the subset of $CI(e_y)$ on each directed path has the immediate dependency relation reconstructibility.

Because $CI(e_y)$ is selected based on the comparison of time ranges, the content of $CI(e_y)$ could be dynamically adapted to the network latency variation so as to let p_{des} identify e_m effectively. In the meanwhile, for the amount of $CI(e_y)$ is unconcerned with the scale of processes, the low transmission overhead could be beneficial to preserve $e_x \xrightarrow{\Delta T} e_y$ with real-time property in large-scale DVE multimedia systems.

3.3 Sending and receiving algorithms of LRTCO

3.3.1 Sending messages algorithm

With the selection of $CI(e_y)$, p_j sends it with e_y in a message to p_{des}. Before describing the algorithm of sending the message, several structures of local variants are given as follows.

- $VT(p_j)$—the vector of logical times to track the numbers of messages diffused by processes. The size of $VT(p_j)$ is equal to n. $VT(p_j)$ records the logical time of p_j.

- $CG(p_j)$—the multi-linked list storing the current effective cause-effect relation graph of p_j. To represent an event and its relation, each node of $CG(p_j)$ contains four variants: $(r(e_x), t_x, [\Delta t_{xmin}, \Delta t_{xmax}], ptr[num])$, where $ptr[num]$ is a set of multiple pointers of which each one is pointed to the node that represents the predecessor which has immediate dependency relation with the event represented by this node. Thus $L(e_x, e_y)$ could be indicated by $ptr[num]$. To avoid the unlimited increment of $CG(p_j)$, p_j would periodically delete redundant nodes in it. Assume the current time is t, then $t < t + \Delta t_{min}$. If a node $(r(e_x), t_x, [\Delta t_{xmin}, \Delta t_{xmax}], ptr[num])$ meets that $t_x + \Delta t_{xmax} < t$, it can obtain that $t_x + \Delta t_{xmax} \leq t + \Delta t_{min}$. Thus the other causal nodes of it could be deleted if they are not in multiple paths.

- $CI(e_y)$—the set storing causal information which would be sent to p_{des} to with e_y in a message. Each element in it contains four parts: $(r(e_x), t_x, [\Delta t_{xmin}, \Delta t_{xmax}],$ $lc[num])$. $r(e_x), t_x, [\Delta t_{xmin}, \Delta t_{xmax}]$ could be obtained from $CG(p_j)$ and the function of $lc[num]$ is similar to $ptr[num]$ in $CG(p_j)$ by storing the indexes of its immediate dependent elements in $CI(e_y)$.

- CD_M—the vector storing the indexes of the immediate dependent elements containing $r(e_x)$ such that $e_x \downarrow e_y$ in $CI(e_y)$. An element in CD_M contains two variants: $(r(e_x), ltr)$, where ltr denotes the index of one immediate dependent element containing $r(e_x)$. With the CD_M, p_{des} could begin to traverse $CI(e_y)$ using Depth-First Search algorithm.

e_y	j	$VT(p_j)[j]$	t_y	$[\Delta t_{min}, \Delta t_{max}]$	ΔT_y	CD_M	$CI(e_y)$

Figure 2: The form of message M

The algorithm of sending message M

```
1.  VT(p_j)[j]=VT(p_j)[j] + 1 ;                          % update the logical time of p_j
2.  t_y = timeGetTime() ;                                 % obtain wallclock time t_y
3.  for( a=0; a<CN(p_j).size(); a++ )          % traverse CG(p_j) from each pointer in CN(p_j)
4.        SelectCI( null, null, CN(p_j)[a].ptr );      % recursively compute CI(e_y) and its CD_M
5.  M ← ( e_y, j, VT(p_j)[j], t_y, [Δt_min, Δt_max], ΔT_y, CD_M, CI(e_y) );          % message M
6.  SendMessage(M);
7.  p = CG(p_j).add( r(e_y), t_y, [Δt_min, Δt_max], ptr[CN(p_j).size()] );  % add the new node representing
                                                          % e_y into CG(p_j) and return the pointer to it
8.  for( a=0; a<CN(p_j).size(); a++ )    % each pointer in ptr[num] of the new node points to the
9.      p→ ptr[a]=CN(p_j)[a].ptr;         % node representing the immediate dependent event of e_y
10. CN(p_j).clear;                                  % clear the original elements in CN(p_j)
11. CN(p_j).add( r(e_y), p );                       % add new element ( r(e_y), p ) into CN(p_j)
12. CI(e_y).clear();                                              % clear content of CI(e_y)
13. CD_M.clear();                                                 % clear content of CD_M
14. exit();
```

Figure 3: The algorithm of sending message M

- $CN(p_j)$—the vector storing the pointers to the immediate dependent nodes containing $r(e_x)$ such that $e_x \downarrow e_y$ in $CG(p_j)$. An element in $CN(p_j)$ has two parts: $(r(e_x), ptr)$, where ptr denotes the pointer to one immediate dependent node containing $r(e_x)$. With the $CN(p_j)$, p_j could traverse $CG(p_j)$ using Depth-First Search algorithm.

In order to effectively preserve $e_x \xrightarrow{\Delta T} e_y$ at p_{des}, the form of message M is set as shown in Fig. 2.

The algorithm of sending message M is implemented as described in Fig. 3.

The line 4 of the algorithm described in Fig. 3 is the procedure to recursively compute $CI(e_y)$ and its CD_M. The starting argument of the procedure is a pointer stored in an element of $CN(p_j)$. Then, p_j could begin to traverse $CG(p_j)$ with Depth-First Search algorithm until the subset of $CI(e_y)$ on each directed path has the immediate dependency relation reconstructibility. The procedure is implemented as described in Fig. 4.

```
Procedure SelectCI( up, nm, p )
{
    % the argument up is the index of the element in CI(e_y) which is the caller of this procedure
    % CI(e_y)[up] could be regard as the pointer to the element
    % CI(e_y)[up].lc[nm] would usually store the indexes of its immediate dependent elements
    % p is the pointer to the node in CG(p_j) which is going to be selected into CI(e_y)

    if ( !FindInCG(p) )          % if the node pointed by p is not in CG(p_j), it indicates that the end
        return false ;                          % of a path is reached or an exception occurs
    vi = FindInCI( p.r(e_x) ) ;  % if it's in CG(p_j), check whether it is already selected into CI(e_y)
    if ( vi )
    {                           % if it is already in CI(e_y), vi would record the index of that element
        CI(e_y)[up].lc[nm] = vi ;                             % return the index to the caller
        return ture ;
    }
    vi = CI(e_y).add( *p ) ;     % if it isn't in CI(e_y), add it into CI(e_y) and record the index in vi
    if ( up == null && nm == null )        % if the new added element represents e_x and e_x↓e_y
        CD_M.add( CI(e_y)[vi].r(e_x), vi ) ;           % the ( r(e_x), vi ) should be added into CD_M
    else                                                                  %  if ¬( e_x↓e_y )
        CI(e_y)[up].lc[nm] = vi ;                             % return the index to the caller
    if ( CI(e_y)[vi].( (t_x + Δt_xmax) ≤ (t_y + Δt_min) ) )   % if the subset of CI(e_y) on this path has  the
                                                   % immediate dependency relation reconstructibility
        for (  a=0; a<num; a++ )                          % terminate the selection on this path
            CI(e_y)[vi].lc[a] = null ;
    else                                                   % if the termination condition is not met
        for ( a=0; a<num; a++ )                                % continue to recursively select
            if ( ! SelectCI( vi, a, CG(p_j)[p].ptr[a] ) )      % the new node of CG(p_j) into CI(e_y)
                CI(e_y)[vi].lc[a] = null ;                     % if the end of a path is reached
                                            % or an exception occurs, terminate the selection on this path
    return ture ;
}
```

Figure 4: The procedure to recursively compute $CI(e_y)$ and its CD_M

```
The algorithm of receiving message M

1.  t = timeGetTime() ;                                        % obtain current wallclock time t
2.  if( (t>t_y+ΔT_y) || (VT(p_j)[j]≤VT(p_des)[j]) )           % M is expired or has been discarded
    {
3.      AbandonMessage(M) ;                                                          % abandon M
4.      exit() ;
    }
5.  else                                                                            % M is valid
    {
6.      for( b=0; b<CD_M.size(); b++ )             % check whether item (2) in Definition 4 is satisfied
7.          if( CD_M[b].r(e_x).a > VT(p_des)[ CD_M[b].r(e_x).i ] )    % if there exists e_x such that e_x↓e_y
            {                                                       % and e_x is not delivered at p_des
8.              BufferMessage(M) ;                                            % buffer M to MQ(p_des)
9.              exit() ;
            }
10.     ProcessMessage(M) ;        % if item (2) in Definition 4 is satisfied, process M immediately
11.     exit() ;        }
```

Figure 5: The algorithm of receiving message M

3.3.2 Receiving messages algorithm

After receiving the message M containing e_y, p_{des} would check whether the immediate cause events e_x of e_y, i.e. $e_x \downarrow e_y$, have been delivered. If all the e_x are delivered, it is necessary for p_{des} to deliver e_y immediately according to the item (2) in Definition 4 . In this case, $e_m = e_x$. If there exists any undelivered immediate cause event, p_{des} would buffer M. Then, p_{des} periodically checks the message buffer to find out whether all those e_x have been delivered. If at the moment $t = t_y + \Delta T_y$ or M is required to be delivered, all the e_x are delivered, i.e. $e_m = e_x$, p_{des} could directly deliver e_y. If there are still delayed and undelivered immediate cause events in some paths, then p_{des} would commence to compute e_m in those paths.

Therefore, there exist different cases for M as follows: If M has been discarded at p_{des} or current moment $t > t_y + \Delta T_y$, i.e. M is expired, M would be abandoned; if M is valid and item (2) in Definition 4 concerned with e_y is satisfied at t, p_{des} would process M immediately; if M is valid but item (2) in Definition 4 is not satisfied at current moment t, p_{des} would buffer M into $MQ(p_{des})$.

- $MQ(p_{des})$—messages buffer at p_{des}. p_{des} would periodically scan the buffer, if item (2) in Definition 4 concerned with e_y is satisfied, or current moment $t = t_y + \Delta T_y$, or M is required to be delivered, p_{des} would call the procedure to process M.

The algorithm of receiving message M is implemented as described in Fig. 5.

When p_{des} needs to process M, it would handle M and its undelivered predecessor messages in causal order, which is realized by recursively calling the procedure in line 10 of Fig. 5. If item (2) in Definition 4 concerned with e_y is satisfied, M could be delivered immediately. If there exist delayed messages, p_{des} could use $CI(e_y)$ to create a local message, which contains no actual event but control information, to replace a delayed message so that the recursively calling of the processing procedure can function well.

```
Procedure ProcessMessage( M )
{
    % M contains { e_y, j, VT(p_j)[j], t_y, [Δt_min, Δt_max], ΔT_y, CD_M, CI(e_y) }
    % an element in CD_M contains ( r(e_x), ltr ) such that e_x↓e_y and ltr is the index of the element
    % in CI(e_y) which has the same r(e_x)

    p = CG(p_des).add( r(e_y), t_y, [Δt_min, Δt_max], ptr[ CD_M.size() ] ) ;     % the new node with relative
                                    % data of e_y is added into CG(p_des), and pointer to the node is returned
    for( b=0; b<CD_M.size(); b++ )                                              % search CD_M
    {
        tp = CI(e_y)[ CD_M[b].ltr ] ;   % find the element containing r(e_x) such that e_x↓e_y in CI(e_y)
        if( tp.r(e_x).a ≤ VT(p_des)[ tp.r(e_x).i ] )       % if e_x is delivered at p_des, return the pointer to
            p→ptr[b] = FindInCG( tp.r(e_x) ) ;              % the node representing e_x in CG(p_des)
        else                                                     % e_x is not delivered at p_des
        {
            M' = FindInMQ( tp.r(e_x) ) ;     % check the message M' containing e_x in MQ(p_des)
            if( M' )                                 % if M' is in the buffer, recursively call the
                p→ptr[b] = ProcessMessage( M' ) ;                 % procedure to process M'
            else                              % if M' is not in MQ(p_des), it is a delayed message
            {                                          % create a local message M'' to replace M'
                for( c=0; c<num; c++ )                                 % create CD_M'' and M''
                    CD_M''.add( CI(e_y)[ tp.lc[b]].r(e_x), tp.lc[b] );
                M'' ← ( null, tp.r(e_x).i, tp.r(e_x).a, tp.t_x, tp.[Δt_xmin ,Δt_xmax], null, CD_M'', CI(e_y) ) ;
                p→ptr[b] = ProcessMessage(M'') ;
            }
        }
        for( d=0; d<CN(p_des).size(); d++ )             % if CN(p_des) contains the element pointing
            if( CD_M[b].r(e_x) == CN(p_des)[d].r(e_x) )    % to the node representing e_x in CG(p_des)
                CN(p_des).remove(d) ;                                     % delete the element
    }
    CN(p_des).add( r(e_y), p ) ;                    % add the new element ( r(e_y), p ) into CN(p_des)
    if( VT(p_j)[j] > VT(p_des)[j] )                            % update the logical time at p_des
        VT(p_des)[j] = VT(p_j)[j] ;
    DeliveryEvent(e);                                                       % delivery e_y itself
        return p ;        % return p pointing to the node representing e_y in CG(p_des) to the caller
}
```

Figure 6: The procedure to recursively process M

Thus, p_{des} could identify e_m and preserve $e_x \xrightarrow{\Delta T} e_y$ effectively. Once a message, remotely received or locally created, is delivered at p_{des}, the new node with relative data of the message would be added into $CG(p_{des})$. Then, as p_{des} is going to send a message, it could select correct control information from $CG(p_{des})$. The procedure is described in Fig. 6.

4 Experimental results and analysis

Experiments have been conducted to evaluate the efficiency of LRTCO algorithm in the distributed causality verification environment established on a PC cluster of 30 high performance machines. The framework of the environment is illustrated in Fig. 7. The run time infrastructure of the environment is BH-RTI [Zhao, Zhou and Lu (2008)] developed by Beijing University of Aeronautics and Astronautics following the High Level Architecture (HLA) standard [IEEE (2000, 2001, 2003)]. The middleware between BH-RTI and federates is designed to consist of the network coordinate computation module and the real-time causal order delivery module.

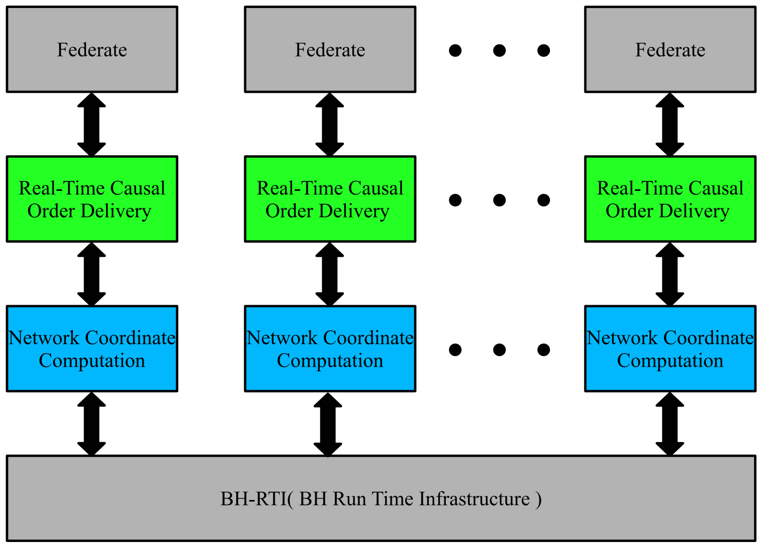

Figure 7: The framework of the distributed causality verification environment

A distributed real-time air battle simulation is developed to run at federates to display the effects of causal order control algorithms. To simulate the transmission delay of WAN, a Spirent ConNIE is utilized to generate network impairments. The ordering mechanism in BH-RTI is set to be Receive Order (RO) in the experiments.

Through the distributed air battle simulation, LRTCO algorithm is compared with the existing real-time causal order control approaches: ERO [Zhou, Cai and Turner (2007)] and DCCO [Rodrigues, Baldoni and Anceaume (2000)]. Multiple experiments are conducted with different scales of entities running as processes and for each scale the three algorithms are implemented in turn. The GUI of distributed air battle simulation is shown in Fig. 8.

For the correctness of the delivery order of events can be evaluated by the numbers of causal order violations, Fig. 9 shows the causal order violations of ERO, DCCO and LRTCO in the experiments. Because each message in ERO merely contains an immediate dependent event as control information, the causal order violations of it are greater than those of DCCO and LRTCO in each scale. When the scale is 3000, the number of violations is approximately 300 while it is over 1400 when the scale is 11000. The complete vector time used by DCCO in each message can reduce causal order violations lower than 100 when scale is 3000, but as the transmission overhead is closely coupled with the scale, the number of violations rises a lot as the scale expands. In 9000 it is approximately 166% of that in 7000 and in 11000 it is about 187% of that in 9000. The violation number of LRTCO is slightly higher than that of DCCO in 3000, but it is lower when the scale is above 3000 due to the control information irrelevant to the scale. Especially when the scale is 11000, the violation number is merely 200 or so, which is approximately 30% of that of DCCO and 15% of that of ERO in the identical scale.

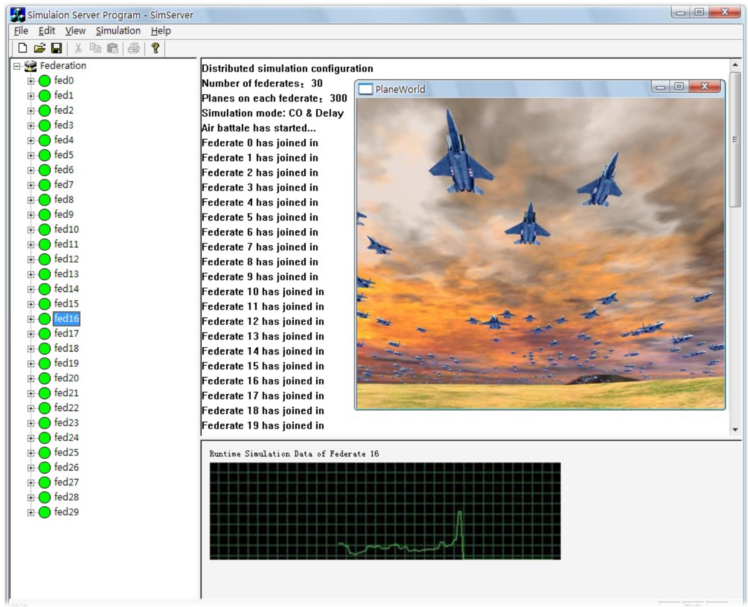

Figure 8: The GUI of the distributed air battle simulation

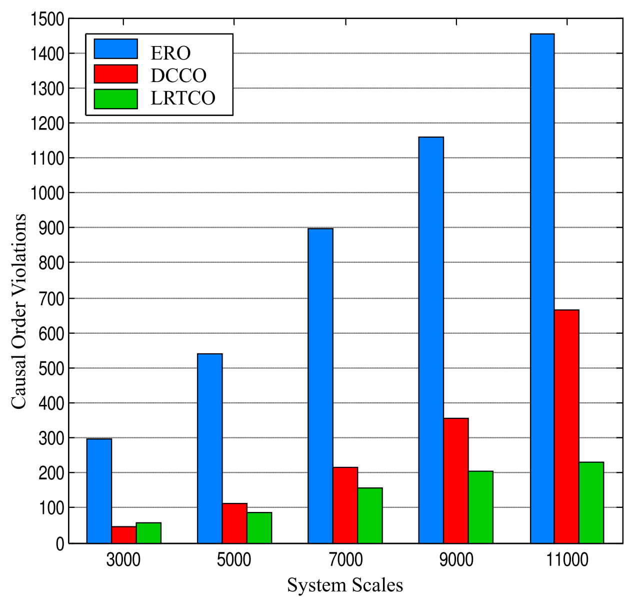

Figure 9: Causal order violations in different scales

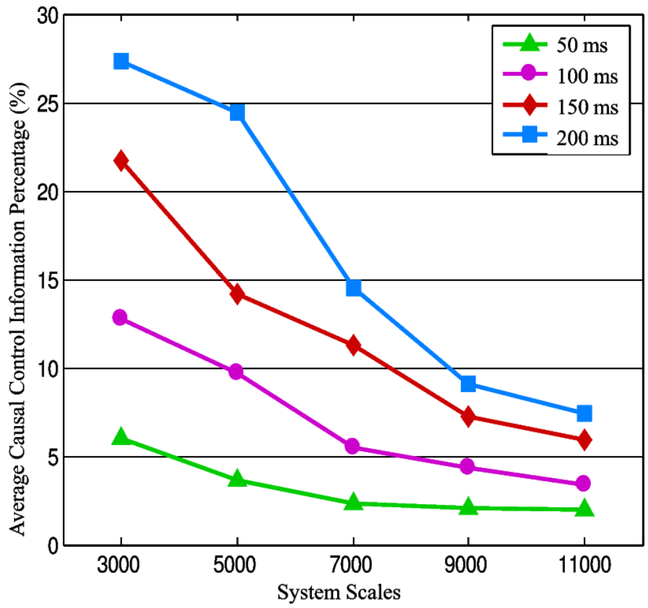

Figure 10: Average causal control information percentage

As the transmission overhead can be estimated by the average amount of causal control information in each message, Fig. 10 shows the percentage of average causal control information of LRTCO compared to the size of vector time of DCCO in different average network transmission delay conditions. As can be seen in scale 3000, when average network transmission delay is 50 ms, the percentage is about 6%. As the transmission delay rises, the number of messages that can not arrive in time may increase, so that the average amount of control information increases either, but when transmission delay is 200 ms, the percentage is merely 27% or so. With the expanding scale, the size of vector time raises a lot, whereas the average amount of causal control information of LRTCO is irrelevant to the scale, so the percentage gradually diminishes. In 3000, the percentage is approximately 6%, 13%, 22% and 27% when the average transmission delay is 50 ms,100 ms, 150 ms and 200 ms. And in 11000, the percentage correspondently decreases to 2%, 3%, 6% and 7%.

5 Conclusions

In the large-scale DVE systems, causal order delivery of events needs to be preserved in real-time. However, some causal events may arrive late due to the large network transmission delay especially on WAN, which would lead to that the cause-effect relations among received events change into concurrent relations and that causal order violations occur if without the causal order control. In this article, we investigate real-time causal order delivery of events. First, the two cases of real-time causal order delivery are defined. Then, we analyze and define the current minimum causal event, and prove that if the proper causal control information selected by the sending process could be used by each destination process to identify its own current minimum causal event in the received events, the two cases of real-time causal order delivery could be preserved. Thus, we discuss the network transmission time range and arriving time range of an event, based on which it is proved that if the selected causal control information has the immediate dependency relation reconstructibility, each destination process could use it to find out its own current minimum causal event in received events and the control information has no redundant data. After the above analysis and proofs, we propose the Lightweight Real-Time Causal Order (LRTCO) algorithm for large-scale DVE systems, which distributedly realizes designing and implementing the procedure to select the causal control information with the immediate dependency relation reconstructibility at sending process, sending messages algorithm, receiving messages algorithm, and the procedure to recursively deliver the received events in real-time causal order at each destination process. At last, multiple experiments are conducted to evaluate the efficiency of LRTCO algorithm compared with the other existing real-time causal order algorithms in the distributed causality verification environment. The experimental results demonstrate that LRTCO could effectively preserve real-time causal order delivery of events in large-scale DVE systems by greatly reducing the causal order violations at destination processes and costing low transmission overhead and communica-tion bandwidth due to the causal control information dynamically adapted to the network latency variation and irrelevant to the system scale.

In our future work, we would like to implement more large-scale DVE application systems using LRTCO programs on Internet, and evaluate the performance of the distributed computing systems, so as to obtain more evaluation results about LRTCO preservation efficiency of real-time causal order delivery.

Acknowledgement: This research work is supported by Hunan Provincial Natural Science Foundation of China (Grant No. 2017JJ2016), Hunan Provincial Education Science 13th Five-Year Plan (Grant No. XJK016BXX001), Social Science Foundation of Hunan Province (Grant No. 17YBA049), 2017 Hunan Provincial Higher Education Teaching Re-form Research Project (Grant No. 564) and Scientific Research Fund of Hunan Provin-cial Education Department (Grant No. 16C0269 and No. 17B046). The work is also sup-ported by Open foundation for University Innovation Platform from Hunan Province, China (Grand No. 16K013) and the 2011 Collaborative Innovation Center of Big Data for Finan-cial and Economical Asset Development and Utility in Universities of Hunan Province. We also thank the anonymous reviewers for their valuable comments and insightful sug-gestions.

References

Agarwal, S.; Lorch, J. (2009): Matchmaking for online games and other latency-sensitive p2p systems. *Proceedings of SIGCOMM Conference*, pp. 1239-1255.

Balci, O.; Fujimoto, R.; Goldsman, D.; Nance, R.; Zeigler, B. (2017): The state of innovation in modeling and simulation: The last 50 years. *Simulation Conference*, pp. 821-836.

Baldoni, R.; Mostefaoui, A.; Raynal, M. (1996): Causal delivery of messages with teal-time data in unreliable networks. *Real-Time Systems*, vol. 10, no. 3, pp. 245-262.

Baldoni, R.; Prakash, R.; Raynal, M. (1998): Efficient δ-causal broadcasting. *Journal of Computer System Science and Engineering*, vol. 13, no. 5, pp. 263-269.

Cai, W.; Turner, S.; Lee, B. (2005): An alternative time management mechanism for distributed simulations. *ACM Transactions on Modeling and Computing Simulation*, vol. 15, no. 2, pp. 109-137.

Dabek, F.; Cox, R.; Kaashoek, F. (2004): Vivaldi: a decentralized network coordinate system. *ACM SIGCOMM Computer Communication Review*, vol. 34, no. 4, pp. 15-26.

Evropeytsev, G.; Dominguez, E.; Hernandez, S.; Trinidad, M.; Cruz, J. (2017): An efficient causal group communication protocol for p2p hierarchical overlay networks. *Journal of Parallel and Distributed Computing*, vol. 102, no. C, pp. 149-162.

Fujimoto, R. (2000): *Parallel and Distributed Simulation Systems*. New York: Wiley Interscience.

Hernandez, S. (2015): The minimal dependency relation for causal event ordering in distributed computing. *Applied Mathematics and Information Sciences*, vol. 9, no. 1, pp. 57-61.

Hernandez, S.; Fanchon, J.; Drira, K. (2004): The immediate dependency relation: an optimal way to ensure causal group communication. *Annual Review of Scalable Computing*, vol. 6, no. 3, pp. 61-79.

IEEE (2000): *IEEE Standard for Modeling and Simulation (M&S) High Level Architecture(HLA)-Framework and Rules (IEEE Std 1516-2000).* The Institute of Electrical and Electronics Engineers, Inc.

IEEE (2001): *IEEE Standard for Modeling and Simulation (M&S) High Level Architecture(HLA)-Federate Interface Specification (IEEE Std 1516.1-2000).* The Institute of Electrical and Electronics Engineers, Inc.

IEEE (2003): *IEEE Recommended Practice for High Level Architecture(HLA) Federation Development and Execution Process (FEDEP) (IEEE Std 1516.3-2003).* The Institute of Electrical and Electronics Engineers, Inc.

Kshemkalyani, A.; Singhal, M. (1998): Necessary and sufficient conditions on information for causal message ordering and their optimal implementation. *Distributed Computing*, vol. 11, no. 2, pp. 91-111.

Lamport, L. (1978): Time, clocks, and the ordering of events in a distributed system. *Communications of the ACM*, vol. 21, no. 7, pp. 558-565.

Prakash, R.; Raynal, M.; Singhal, M. (1997): An adaptive causal ordering algorithm suited to mobile computing environments. *Journal of Parallel and Distributed Computing*, vol. 41, no. 2, pp. 190-204.

Rodrigues, L.; Baldoni, R.; Anceaume, E. (2000): Deadline-constrained causal order. *Proceedings of the 3rd IEEE International Symposium on Object-Oriented Real-Time Distributed Computing*, pp. 234-241.

Schwarz, R.; Mattern, F. (1994): Detecting causal relationships in distributed computations: In search of the holy grail. *Distributed Computing*, vol. 7, no. 3, pp. 149-174.

Yavatkar, R. (1992): MCP: A protocol for coordination and temporal synchronization in multimedia collaborative applications. *Proceedings of the 12th International Conference on Distributed Computing Systems*, pp. 606-613.

Zhao, Q.; Zhou, Z.; Lu, F. (2008): Algorithm of simulation time synchronization over large-scale nodes. *Science in China Series F: Information Sciences*, vol. 51, no. 9, pp. 1239-1255.

Zhou, S.; Cai, W.; Turner, S. (2007): Critical causal order of events in distributed virtual environments. *ACM Transactions on Multimedia Computing, Communications and Applications*, vol. 3, no. 3.