A Method of Fast Synchronizing Asynchronous Clocks in Distributed Applications

Jun Liu
Computer Science Department
University of North Dakota
Grand Forks, ND 58202-9015
jliu@cs.und.edu

Abstract

In many distributed applications, identifying the sequence of event occurrence is useful. When local clocks lack of synchronization, determination of the order of event occurrence is difficult or time-consuming without developing new methods for synchronizing clocks. In this paper, we describe a pseudo-synchronization method for projecting occurrence time of events onto a common timeline such that the direct "happened-before" relations are maintained. The basic idea of this technique is to estimate the clock shift between two clocks from the minimum difference of occurrence time of two events included in a direct "happened-before" relation. Evaluation results suggest that this method can pseudo-synchronize non-base clocks with a base clock without violating the direct "happened-before" relations.

Jun Liu
Computer Science Department
University of North Dakota
Grand Forks, ND 58202
jliu@cs.und.edu

1 Introduction

In many distributed application scenarios, identifying the instantaneous state of a distributed application is useful in matching requirements of a distributed application to the characteristics of the underlying system. The instantaneous state of a distributed application at a given time moment is the set of events that occurred at that moment. High-level applications are allowed to express their requirements, and they are notified by the underlying system when events that match their requirements are available [1, 2, 5]. For instance, in ubiquitous computing applications running on mobile hosts equipped with wireless transceivers, tasks running on different hosts are difficult to be coordinated because of the opportunistic communication between hosts. Hence, discovering the state of a ubiquitous computing application could facilitate the execution of operations issued by upper-level computing agents so that computing tasks can be coordinated.

Identifying an instantaneous state of a distributed or parallel application is generally difficult. Even though monitoring events occurred in distributed or parallel systems has been studied in a number of studies [7, 3, 9], but it is still difficult to design an efficient tracing tool to make different components in a system coordinate in recording events and in extracting useful information from event traces recorded by different hosts. The task of tracing events becomes even more complicated when occurrence time is involved due to lack of a universal clock [7].

In a distributed application consisting of multiple hosts each of them is equipped with its own local clock, an event is recorded only once by the host where the event occurs, along with its occurrence time-stamp with respect to the local clock of the recording host. Events occurred at a host are recorded in sequence with respect to the chronological order of their occurrence. When the overall chronological order of the occurrence of events is to be revealed, events recorded at different hosts need to be unified onto a common time line. When local clocks lack of synchronization, it is generally difficult to unify events, that are time-stamped with respect to different local clocks, onto a common time line without synchronizing local clocks.

When lacking of synchronized local clocks, determining the order of occurrence of events satisfies needs in many distributed applications. In this case, the order of occurrence of events can be determined with respect to a logical clock which can be formally characterized using the "happened-before" relations [6] by signifying the mutual order of occurrence of two events. The two events having a direct "happened-before" relation could be two events occurred consecutively at a same host, or two events involved in transmitting a message between different hosts. Here, we focus on the "happened-before" relations each of which consists of two events: a send event and a receive event occur in different hosts.

The order of occurrence of events can be determined incrementally by exploring "happened-before" relations between events through scanning a long history of occurrence of events. Incrementally sequentializing the order of occurrence of events can achieve a very good serialization of events, since many seemingly concurrent events could be serialized. (Two events are called *concurrent* events when there is not a "happened-before" relation between them.) However, sequentializing events incrementally could take a long processing time.

The order of occurrence of events can also be determined point-wisely. In this approach, only a small number of events that occur around the time point of interest are sequential-

ized, such that the processing time can be limited. For example, in order to discover the cause to an event occurred at a particular moment, only those events occurring in the neighborhood of this moment need to be sequentialized. It is inefficient and inapplicable to do this task by serializing a long history of events using the method of incremental serialization.

In this paper, we propose a new technique to pseudo-synchronize local clocks in distributed applications such that sequence of event occurrence can be serialized under the synchronized clocks. Pseudo-synchronizing one clock with another clock is to discover the shift between the two clocks. The basic idea of pseudo-synchronizing two clocks is to estimate the shift between the clocks by examining the difference of occurrence time between events having a direct "happened-before" relation. The term pseudo-synchronization comes after the fact that the estimated shift between two clocks might not reflect the actual shift between them.

Under pseudo-synchronized clocks, it is ideal to maintain the original order of occurrence of events. However, shifts between clocks estimated using the procedure of pseudo-synchronization can be over estimated due to lack of knowledge of actual transmission delays between hosts. When the shift between clocks at two hosts is discovered, relations of direct transmission involving only these two hosts are examined; the shift is estimated as the minimum difference between the occurrence time of two events having a relation of direct transmission. This estimation is accurate only when the transmission delay between the two hosts is zero. The accuracy of the estimated shift between two clocks can be improved if the knowledge of transmission delay between the two hosts where the two clocks are install is available. If the occurrence time, with respect to the reference clock, of events is estimated based on over-estimated shifts, some relations of direct transmission could be violated. Therefore, the occurrence time of events needs further adjustments in order to resolve relations that have been violated.

The rest of this paper is organized as follows. The previous works relating to this subject in Section 2. The procedure of pseudo-synchronization of clocks is described in Section 3, and the procedure of estimating occurrence time of events with respect to a common reference clock is described in Section 4. The evaluation to the method of deriving momentary states of distributed applications is in Section 5. Our method is summarized in Section 6.

2 Related Work

Serialization of events with respect to the order of their occurrence has been widely used in performance analysis and in error debugging. Due to lack of a universal clock in most distributed application scenarios, one method of serializing the occurrence of events in distributed applications is by making use of a logical clock in place of a universal clock. Lamport [6] presented an approach of partially serializing events by making use of a logical clock that is formally defined as the "happened-before" relation. The "happened-before" relations are defined under two assumptions: 1) all events, that occur on the same process, form a sequence, *i.e.* they are *a priori* totally ordered; 2) sending or receiving a message is an event in a process.

Even though the Lamport logical clock satisfies the clock condition, but it is not strongly

consistent and not being able to always capture concurrency. To overcome the deficiency of the Lamport logical clock, a concept of vector clock was later proposed by a number of researchers, most notably Fidge [4] and Mattern [8]. A vector clock is an array of integers $VT[n]$, where n is the number of processes in the system. Each processor maintains its own vector clock that assigns time stamps to events by three rules: 1) all events that occur consecutively on the same processor are time-stamped sequentially; 2) the time stamp of a sending event is carried in the message being sent; 3) upon receipt of a message, the event of receiving a message is time-stamped by the maximum of the time stamp carried in the message and the local clock of the receiver.

Events having "happened-before" relations have been made use of in our methods of pseudo-synchronization of clocks, and of adjustments to occurrence time of events with respect to a common reference clock. Moreover, the method of adjusting occurrence time of events has a flavor of the elastic method because not all occurrence time of events occurred at a same host is adjusted consistently.

Srinivasan *et al.* proposed the Near-Perfect State Information (NPSI) adaptive protocols [11] and the Elastic Time Algorithm (ETA). In parallel computing systems, in order for the logical processes (LPs) to schedule their executions, the correct state information of the system needs to be informed to the LPs. However, the overhead of delivering the correct state information of the system is not acceptable in reality, a protocol of propagating of good approximation of perfect state information is desired. Both NPSI and ETA are control mechanisms to guide LPs to schedule their next events. The difference between NPSI and ETA is that NPSI defines a class of algorithms with controlled optimist, whereas ETA is an instance belonging to this class. Quaglia [10] proposed the scaled version of ETA to speed up the execution of LPs by taking into account of execution delays of events in the optimism control in LPs.

3 Pseudo-Synchronization

When all hosts share a universal clock, the occurrence time of an event recorded on different hosts is identical. In this case, the order of occurrence of events could be naturally determined with respect to their occurrence time. However, when local clocks at different hosts lack of synchronization, an event can have different occurrence time with respect to different clocks. Without having the local clocks synchronized with respect to a universal clock, it is difficult to serialize the occurrence of events according to their occurrence time recorded with respect to different local clocks. Hence, the efficient method of clock synchronization is highly desired in application scenarios in which a universal clock is unavailable.

Since it is very difficult or time-consuming to make asynchronous clocks to be precisely synchronized, a method of synchronizing clocks imprecisely is described in this section. This method focuses more on maintaining "happened-before" relations than precisely synchronizing local clocks. Doing so reduces the computational overhead in the synchronization. The basic idea of this method is to determine the shift between two clocks imprecisely making use of the "happened-before" relations. In a "happened-before" relation, the send event must occur prior to the receive event, no matter how their occurrence time are

recorded. Hence, the shift between two local clocks is reflected in the difference of the occurrence time of the send and receive events in a "happened-before" relation.

3.1 Pair-wise Pseudo Synchronization of Clocks

For two hosts involved in a "happened-before" relation, their local clocks are assumed to be asynchronous. The two hosts are called hosts 1 and 2. A time point t with respect to a universal clock is assumed to map into t_1 and t_2 on host 1 and 2, respectively. The shift between the local clock in host 1 and the universal clock can be derived as $S_1 = t_1 - t$, and the shift between the local clock in 2 and the universal clock is $S_2 = t_2 - t$. Since t_1 and t_2 should represent the same time point if the two local clocks are perfectly synchronized, the shift between the two clocks is expressed as $|t_1 - t_2| = |S_1 - S_2|$. A "happened-before" relation $e_{1,t_a} \rightarrow e_{2,t_b}$ means that the send event e_{1,t_a} occurs in host 1 at a local time t_a, and the receive event e_{2,t_b} occurs in host 2 at a local time t_b. The relation $e_{1,t_a} \rightarrow e_{2,t_b}$ can be used to synchronize the clock in host 2 to the clock in host 1. In this case, the clock in host 1 is treated as the base clock, and the delay between the occurrence of events e_{1,t_a} and e_{2,t_b} is denoted as d_{ab}. The local time t_b in host 2 can be expressed in terms of the local time t_a in host 1, *i.e.*, $t_b = t_a + d_{ab} - S_1 + S_2$. Hence, if the value of delay d_{ab} is known, the shift from the clock in host 2 to the base clock in host 1, denoted as S_{21}, can be precisely measured as $S_{21} = t_2 - t_1 - d_{ab} = S_2 - S_1$. If the value of delay d_{ab} is unknown, then S_{21} can be approximated using $t_2 - t_1$. The approximation to shift S_{21} using $t_2 - t_1$ is called *pseudo-synchronization* of two clocks. The approximated value of shift S_{21} is denoted as S'_{21} which is bigger than the actual shift S_{21}, *i.e.*, $S'_{21} > S_{21}$. Likewise, S_{12} can be measured using a pair of events in a relation $e_{2,t_u} \rightarrow e_{1,t_v}$ when the clock in host 2 is treated as the base clock. Even though $S_{12} = -S_{21}$, but $S'_{12} \neq -S'_{21}$ if $d_{ab} \neq d_{uv}$.

Under two pseudo-synchronized clocks, the main concern is whether "happened-before" relations can be violated. The set of "happened-before" relations with the send and receive events occurring in host 2 and 1, respectively, is denoted as $E_{21} = \{e_{2,t_a} \rightarrow e_{1,t_b}\}$ where t_a and t_b are the time-stamps recorded with respect to the local clocks in hosts 1 and 2, respectively. After the clock in host 2 has been synchronized with the clock in host 1 by a pseudo-shift S'_{21}, the receive event in a relation $e_{1,t_a} \rightarrow e_{2,t_b}$ has a new occurrence time $t'_b = t_b - S'_{21}$ (ref. Figure 1). The relation $e_{1,t_a} \rightarrow e_{2,t_b}$ is violated under the pseudo-synchronized clocks if $t'_b < t_a$. It is possible that some "happened-before" relations can be violated under two pseudo-synchronized clocks, if the pair of send and receive events used in estimating the pseudo-shift have not been selected carefully.

When the clock in host 2 needs to be synchronized with the base clock in host 1, an appropriate relation needs to be selected from a set $E_{12} = \{e_{1,t_a} \rightarrow e_{2,t_b}\}$ which is the set of "happened-before" relations with the send and receive events occurring in host 1 and 2, respectively. In this case, the appropriate relation to be used is the one with the minimum value of $t_b - t_a$. This fact can be formally claimed into Proposition 3.1.

Proposition 3.1 No relation in set E_{12} is violated after the clock in host 2 is pseudo-synchronized with the clock in host 1 by the pseudo-shift $S'_{21} = \min_{e_{1,t_a} \rightarrow e_{2,t_b}} \{t_b - t_a\}$. □

Proof: Consider an arbitrary relation $e_{1,t_x} \rightarrow e_{2,t_y}$ in set E_{12}. After the clock in host 2 is

pseudo-synchronized with the clock in host 1 by a pseudo-shift $S'_{21} = \min_{e_{1,t_a} \rightarrow e_{2,t_b}} \{t_b - t_a\}$, the occurrence time of event e_{2,t_y} becomes $t'_y = t_y - S'_{21}$. Since $t_y - t_x \geq S'_{21}$, $t'_y - t_x = t_y - t_x - S'_{21} \geq 0$. Hence, all relations $e_{1,t_x} \rightarrow e_{2,t_y}$ in E_{12} will not be violated under the pseudo-shift S'_{21}. □

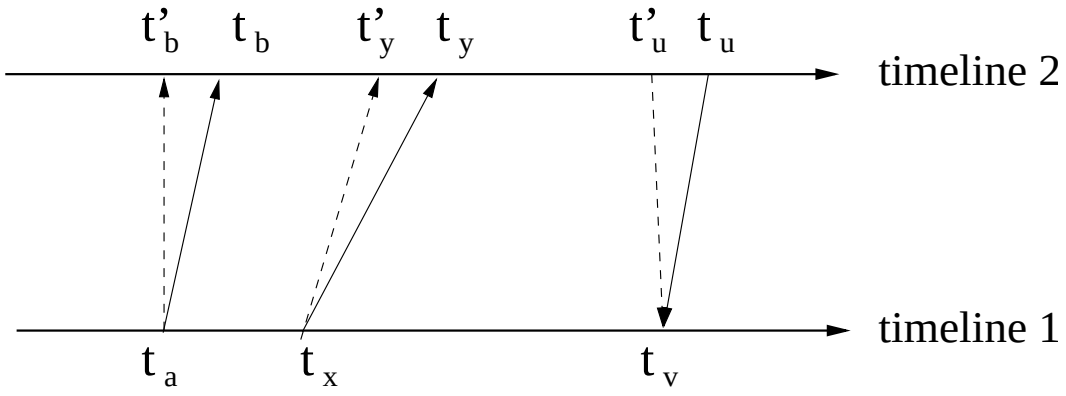

Figure 1: Estimation of a pseudo-shift by maintaining "happened-before" relations when the clock in host 2 is synchronized with the clock in host 1.

Under the same pseudo-shift S'_{21}, another concern is whether the relations in form of $e_{2,t_u} \rightarrow e_{1,t_v}$ will be violated. The answer to this concern is formally described in Proposition 3.2.

Proposition 3.2 No relation in set E_{21} is violated after the clock in host 2 is pseudo-synchronized with the clock in host 1 by the pseudo-shift S'_{21}. □

Proof: Consider an arbitrary relation $e_{2,t_u} \rightarrow e_{1,t_v}$ in set E_{21}. $t_v = t_u + d_{u,v} - S_2 + S_1$. After the clock in host 2 is pseudo-synchronized to the clock in host 1 with a pseudo-shift S'_{21}, the occurrence time of event e_{2,t_u} becomes $t'_u = t_u - S'_{21}$. Since the pseudo-shift S'_{21} can be expressed $S'_{21} = S_2 - S_1+$. Thus, $t_v - t'_u = t_v - t_u + S'_{21} = d_{u,v} - (S_2 - S_1) > 0$ because the delay in transmission $d_{x,y}$ is always assumed to be bigger than 0. Hence, all relations in E_{21} will not be violated under the pseudo-shift S'_{21}. □
Combining the facts stated in Proposition 3.1 and Proposition 3.2, no relation in set E_{12} or E_{21} is violated after the clock in host 2 is pseudo-synchronized with the clock in host 1 by a pseudo-shift $S'_{21} = \min_{e_{1,t_a} \rightarrow e_{2,t_b}} \{t_b - t_a\}$.
The estimated pseudo-shift is always no less than the actual shift, and the amount of over-estimation is bounded by the maximum delay from the base host to the non-base host. For instance, the median values of round-trip time (RTT) measured in the Internet [12] are mostly less than 150 ms, thus, the uni-directional transmission delay between two hosts should mostly be less than 75 ms. Hence, the maximum delay should be no more than 75 ms most of the time, and a 75 ms over-estimation in excess to the actual shift is still reasonably small. Meanwhile, the knowledge of minimum delay between two hosts can be obtained in many application scenarios. There are network measurement tools for obtaining delay information between two hosts in the Internet. Moreover, these measurement tools

are gradually becoming a part of designs of operating systems to facilitate decision making at the application level.
The over-estimation of the shift between two clocks will cause problems when multiple clocks are pseudo-synchronized with respect to a common base clock.

3.2 Pseudo Synchronization of Multiple Clocks

When the pseudo-synchronization method is applied to synchronize multiple clocks, "happened-before" relations can be violated. The fundamental reason for this fact is due to the unknown delays. That is, if the delays are known, then no violation is made possible. However, if the delay information is unknown, then some combination of delay values can lead to violations. For example, in an application scenario consisting of 3 hosts, when the clocks in hosts 2 and 3 are pseudo-synchronized with the clock in host 1 by respective pseudo-shifts S'_{21} and S'_{31}, some "happened-before" relations involving only hosts 2 and 3 might be violated. This claim is formally stated in Proposition 3.3. Following the same notion on the set of "happened-before" relations, the set E_{23} and E_{32} can be defined accordingly. The minimum delay going from host i to host j is denoted as D_{ij} ($1 \leq i, j \leq 3$, $i \neq j$).

Proposition 3.3 After the clocks in host 2 and 3 are pseudo-synchronized with the clock in host 1, respectively, a relation $e_{2,t_x} \rightarrow e_{3,t_y}$ in E_{23} is violated if $d_{xy} < D_{13} - D_{12}$. □

Proof: Consider an arbitrary relation $e_{2,t_x} \rightarrow e_{3,t_y}$. The occurrence time of event e_{2,t_x} is t_x with respect to the local clock in host 2, and the occurrence time of event e_{3,t_y} is t_y with respect to the local clock in host 3. When the shift from the local clock in host 2 to the local clock in host 3 is $S_{32} = S_3 - S_2$, $t_y = t_x + d_{x,y} - S_2 + S_3$ where d_{xy} is the transmission delay between the two events e_{2,t_x} and e_{3,t_y}.
After the clock in host 2 is pseudo-synchronized with the clock in host 1 by a pseudo-shift S'_{21}, the occurrence time of event e_{2,t_x} becomes $t'_x = t_x - S'_{21}$. Correspondingly, the occurrence time of event e_{3,t_y} becomes $t'_y = t_y - S'_{31}$ after the clock in host 3 is pseudo-synchronized with the clock in host 1 by a pseudo-shift S'_{31}. Since $t'_y - t'_x = d_{xy} + D_{12} - D_{13}$, $t'_y < t'_x$ by the assumption that $d_{xy} < D_{13} - D_{12}$. Hence, under the pseudo-synchronized clocks, the relation $e_{2,t_x} \rightarrow e_{3,t_y}$ is violated because the the receive event has a smaller value of occurrence time than the send event. □

3.3 Resolving the Violations

Over-estimation of the actual shift between a pair of clocks is the fundamental cause to violations of "happened-before" relations. In order to resolve the violations, the over-estimated values of shifts need to be adjusted. Consider the case when both clocks in hosts 2 and 3 have been synchronized with the clock in host 1 by pseudo-shifts S'_{21} and S'_{31}, respectively. Under the pseudo-synchronized clocks, "happened-before" relations in E_{23} and/or E_{32} can be violated. The new values of occurrence time of events in a relation $e_{2,t_x} \rightarrow e_{3,t_y}$ become $t'_x = t_x - S'_{21}$ and $t'_y = t_y - S'_{31}$.
When violations only occur in set E_{23}, only the value of S'_{31} needs to be adjusted. For a violated relation $e_{2,t_x} \rightarrow e_{3,t_y}$ in E_{23}, the difference between the new values of occurrence

time is negative, *i.e.*, $t'_y - t'_x < 0$. Thus, the violated relations in set E_{23} can be resolved if a negative value $\min\{t'_y - t'_x\}$ is compensated to S'_{31}. The value $\min\{t'_y - t'_x\}$ is derived from those violated relations in set E_{23}. No compensation is necessary for S'_{21}, *i.e.*, $S''_{21} = S'_{21}$. After this compensation, the clock in host 3 is synchronized with the base clock in host 1 by a new pseudo-shift $S''_{31} = S'_{31} + \min\{t'_y - t'_x\}$ ($S''_{31} \leq S'_{31}$). Under new pseudo-shifts S''_{21} and S''_{31}, it can be shown that $t''_y - t''_x \geq 0$ for any relation $e_{2,t_x} \rightarrow e_{3,t_y}$ in E_{23} as follows.

$$\begin{aligned}
t''_x &= t_x - S''_{21} \\
&= t_x - S'_{21} \\
&= t_x - S_2 + S_1 - D_{12} \\
t''_y &= t_y - S''_{31} \\
&= (t_x + d_{xy} - S_2 + S_3) - (S_3 - S_1 + \min\{d_{xy}\} + D_{12}) \\
&= t_x + d_{xy} - S_2 + S_1 - \min\{d_{xy}\} - D_{12} \\
t''_y - t''_x &= d_{xy} - \min\{d_{xy}\} \\
&\geq 0
\end{aligned} \tag{1}$$

In the meantime, no relation in set E_{32} will be violated under the new pseudo-shifts S''_{21} and S''_{31}. For an arbitrary relation, $e_{3,t_u} \rightarrow e_{2,t_v}$, in E_{32}, the new occurrence time of e_{3,t_u} and e_{3,t_v} with respect to the clock in host 1 are:

$$\begin{aligned}
t''_u &= t_u - S''_{31} \\
&= t_u - S'_{31} - \min\{t'_y - t'_x\} \\
&= t_u - S_3 + S_1 - \min\{d_{xy}\} - D_{12} \\
t''_v &= t_v - S''_{21} \\
&= (t_u + d_{uv} - S_3 + S_2) - (S_2 - S_1 + D_{12}) \\
&= t_u + d_{uv} - S_3 + S_1 - D_{12} \\
t''_v - t''_u &= (t_u + d_{uv} - S_3 + S_1 - D_{12}) - (t_u - S_3 + S_1 - D_{23} - D_{12}) \\
&= d_{uv} + \min\{d_{xy}\} \\
&> 0
\end{aligned} \tag{2}$$

Therefore, no relations in set E_{32} is violated under new pseudo-shifts S''_{21} and S''_{31}.
Similarly, if violations only occur in set E_{32}, only the value of S'_{21} needs to be adjusted. In this case, a negative value $\min\{t'_v - t'_u\}$ can be estimated from the violated relations $e_{3,t_u} \rightarrow e_{2,t_v}$ in set E_{32}. After the adjustment, the new pseudo-shifts are $S''_{21} = S'_{21} + \min\{t'_v - t'_u\}$ and $S''_{31} = S'_{31}$. Following the same line of reasoning in Equations (1) and (2). It can be shown that no violation is present in set E_{23} and E_{32}.
When violations occur in both E_{23} and E_{32} after both clocks in host 2 and 3 have been synchronized with the clock in host 1 by pseudo-shifts S'_{21} and S'_{31}, respectively, one of S'_{21} and S'_{31} needs to be adjusted in order to resolve the violations. In this case, $\min\{t'_y - t'_x\}$ and $\min\{t'_v - t'_u\}$ are derived from the violated relations in set E_{23} and E_{32}, respectively. If $\min\{t'_y - t'_x\} < \min\{t'_v - t'_u\}$, then the new pseudo-shifts are $S''_{31} = S'_{31} + \min\{t'_y - t'_x\}$ and $S''_{21} = S'_{21}$. If $\min\{t'_y - t'_x\} > \min\{t'_v - t'_u\}$, then the new pseudo-shifts are $S''_{21} =$

$S'_{21} + \min\{t'_v - t'_u\}$ and $S''_{31} = S'_{31}$. In both cases, it can also be verified that no violation in E_{23} and E_{32} is possible.
Therefore, in all the above described cases, no violation in set E_{23} and E_{32} is made possible under adjusted pseudo-shifts S''_{21} and S''_{31}.

4 Serialization of Events Under Pseudo-Synchronized Clocks

In a distributed environment consisting of m hosts, each host is equipped with a local clock, and all clocks are no synchronized. There is not a universal clock. Each host only time stamps events occurred locally with respect to its local clock. Each host also periodically sends the incremental segment of event trace recorded locally from last transmission of trace segment. Upon receiving trace segments sent by other hosts, each host tries to reconstruct a sequence of overall event occurrence. The events in this sequence have to make sure that no "happened-before" relation is violated.
The basic idea of serializing the occurrence of events from trace segments is to first pseudo-synchronize local clocks, and then to serialize the occurrence of events with respect to the new occurrence time of events under pseudo-synchronized clocks. Each host treats its local clock as the base clock and pseudo-synchronize other clocks with respect to its own clock. The segment of event trace recorded in host i ($1 \leq i \leq m$) is denoted as $G_i = \{e_{i,t_k} : 1 \leq k \leq n_i\}$ where n_i is the number of events in G_i and t_k is the occurrence time of e_{i,t_k} with respect to the local clock in host i. The set of "happened-before" relations with the send events occur in host i and the receive events occur in host j is denoted as $E_{ij} = \{e_{i,t_u} \rightarrow e_{j,t_v} : 1 \leq u \leq n_i, 1 \leq v \leq n_j\}$. The local clock in host i is assumed to differ from an imaginary universal clock with a shift s_i. The shift from the clock in host i to the clock in host j is denoted as $S_{ij} = s_i - s_j$ such that a time point t_i in host i and the time point $t_i - S_{ij}$ in host j mean the same time point with respect to the imaginary universal clock. The delay of transmission between a send event e_{i,t_u} and a receive event e_{j,t_v} is denoted as d_{uv}, and the minimum delay sending from host i to host j is denoted as D_{ij}, *i.e.*, $D_{ij} = \min_{\forall e_{i,t_u} \rightarrow e_{j,t_v}} \{d_{uv}\}$. In the following, the description of the procedure of pseudo-synchronization and serialization assumes that the clock in host 1 is the base clock.

4.1 Procedure of the Pseudo-Synchronization

Pseudo-synchronization of other clocks to the base clock in host 1 is to estimate the pseudo-shift S'_{i1} from the clock in host i ($2 \leq i \leq m$) to the clock in host 1. This procedure takes the following steps.

(1) Computing the difference of occurrence time between each pair of events in E_{1i}, *i.e.*, $t_v - t_u$ for all relations in the form of $e_{1,t_u} \rightarrow e_{i,t_v}$ in E_{1i};

(2) Taking the minimum value of these differences as the pseudo-shift S'_{i1}, *i.e.*,

$$S'_{i1} = \min_{\forall \{e_{1,t_u} \rightarrow e_{i,t_v}\} \in E_{1i}} (t_v - t_u).$$

When the clock in host i is synchronized to the clock in host 1 with the pseudo-shift S'_{i1}, no violation of "happened-before" relations in sets E_{1i} and E_{i1} will be resulted under pseudo-synchronized clocks. This fact has been shown in Proposition 3.1 and Proposition 3.2. The processing time spent in pseudo-synchronizing two clocks is the amount of time needed to scan through the two relevant trace segments.

4.2 Estimating the Adjustments to the Pseudo-Shifts

After all local clocks have been pseudo-synchronized to a base clock, violations to "happened-before" relations are made possible between non-base clocks. This fact is stated in Proposition 3.3. Adjustments to the estimated pseudo-shifts S'_{i1} ($2 \leq i \leq m$) are necessary in order to resolve the violations under pseudo-synchronized clocks. The procedure of estimating the adjustments and compensating the adjustments to the pseudo-shifts is described as follows.

(1) Computing the new occurrence time of events in G_i ($2 \leq i \leq m$), *i.e.*, $t'_x = t_x - S'_{i1}$ for all events e_{i,t_x} in G_i; (an event e_{i,t_x} with a new occurrence time t'_x is denoted as e_{i,t'_x}.)

(2) Detecting violations to the "happened-before" relations in E_{ij} ($2 \leq i, j \leq m$) making use of new occurrence time, *i.e.*, a relation $e_{i,t'_x} \rightarrow e_{j,t'_y}$ is violated if $t'_y < t'_x$;

(3) Recording in variable q_{ij} ($2 \leq i, j \leq m$, $i \neq j$) the minimum value of $(t'_y - t'_x)$ for those violated relations $e_{i,t'_x} \rightarrow e_{j,t'_y}$;

(4) Estimating the adjustment a_{ij} and a_{ji} ($2 \leq i, j \leq m$, $i \neq j$) based on q_{ij} and q_{ji}:

   (a) If both $q_{ij} \geq 0$ and $q_{ji} \geq 0$, then $a_{ij} = 0$ and $a_{ji} = 0$;

   (b) If $q_{ij} \geq 0$ and $q_{ji} < 0$, then $a_{ij} = 0$ and $a_{ji} = q_{ji}$;

   (c) If $q_{ij} < 0$ and $q_{ji} < 0$, then:

      (i) If $q_{ij} < q_{ji}$, then $a_{ij} = q_{ij}$ and $a_{ji} = 0$;

      (ii) otherwise, $a_{ji} = q_{ji}$ and $a_{ij} = 0$;

4.3 Derivation of New Pseudo-Shifts

After all a_{ij}'s ($2 \leq i, j \leq m$, $i \neq j$) have been determined, the new pseudo-shifts S''_{j1}'s ($2 \leq j \leq m$) can be obtained as that

$$S''_{j1} \;=\; S'_{j1} + \min_{2 \leq i \leq m} \{a_{ij}\}.$$

4.4 Serialization of Segments of Event Traces

The new occurrence time of events included in G_i, a segment of event trace recorded in host i ($2 \leq i \leq m$), can be derived as $t''_j = t_j - S''_{j1}$ where t_j is the original occurrence time of an event e_{i,t_j}. The original values of occurrence time of all events included in event sets

G_i's ($2 \leq i \leq m$) can be projected into new values of occurrence time onto the time-line with respect to the base clock in host 1. Hence, the events in G_i's can be naturally serialized based on the new values of occurrence time projected onto the time-line with respect to the base clock.

5 Evaluation

The method of serializing occurrence of events by pseudo-synchronizing local clocks has also been validated by an experiment. In this experiment, a distributed application runs on 8 hosts each of which only records events occurring locally. Each host is equipped with its own clock which is lack of synchronization with other clocks. The clock at host 1 is used as the base clock. The non-reference clocks are set asynchronous with the reference clock by clock shifts shown in Table 1.

host i	2	3	4	5	6	7	8
S_{i1}	1.9800	0.2950	1.0617	0.9566	0.3391	0.0967	1.7971

Table 1: The actual clock shifts used in the experiment.

5.1 Setting of the Experiment

An event generator is attached to each host, which generates LOCAL or SEND events. When a LOCAL event, which does not involve sending or receiving a message, occurs at a host, the host just records this event along with the occurrence time of this event with respect to the local clock at the host. When a SEND event occurs at a host, the host has to physically send out a message to the destination host prescribed by the event generator, and records the SEND event along with its occurrence time with respect to its own clock. When a host receives a message from another host, a RECEIVE event occurs at the host and is recorded with respect to the clock at the receiving host.

Every event generator randomly generates LOCAL events and SEND events with equal probabilities, and the generated series of events by a generator follows a *Possion* distribution with the mean inter-arrival time a_i for each host i (shown in Table 1). The choice of a destination host upon a SEND event is also randomly made with equal probabilities for every remote host. Each event generator is set to generate a total of 1000 LOCAL or SEND events. A minimum transmission delay is also randomly chosen between each pair of hosts as shown in Table 2. The actual transmission delay of a message transmitted is the sum of the minimum delay between two corresponding hosts and a random queueing delay which is drawn uniformly in $[0, 0.5s]$.

Before any performing our method, the numbers of violations to direct "happened-before" relations are shown in Table 3.

i\j	1	2	3	4	5	6	7	8
1	0	0.0631	0.0608	0.3495	0.2291	0.2849	0.4302	0.2466
2	0.0153	0	0.1269	0.0872	0.0669	0.4545	0.2586	0.0261
3	0.1563	0.2048	0	0.4451	0.1286	0.1837	0.0911	0.1217
4	0.4145	0.4455	0.4380	0	0.3777	0.2321	0.1713	0.4325
5	0.1073	0.2895	0.2293	0.4774	0	0.0240	0.2511	0.3437
6	0.4231	0.3467	0.2469	0.2918	0.3950	0	0.2245	0.3702
7	0.0529	0.4947	0.2412	0.4404	0.1057	0.0265	0	0.4819
8	0.1694	0.4137	0.2678	0.2088	0.4051	0.3332	0.3010	0

Table 2: The minimum transmission delays D_{ij} from the send host i to a receive host j.

$i \setminus j$	1	2	3	4	5	6	7	8
1	0	0	0	0	0	0	0	0
2	39	0	29	34	21	28	39	17
3	17	0	0	0	0	0	13	0
4	34	0	24	0	0	30	33	0
5	21	0	36	0	0	35	30	0
6	0	0	0	0	0	0	2	0
7	2	0	0	0	0	0	0	0
8	33	0	36	36	26	38	30	0

Table 3: Under unsynchronized clocks, the numbers of violations to relations of direct "happened-before" relations between an arbitrary pair of hosts i, j in the form of $e_{i,t_x} \rightarrow e_{j,t_y}$.

5.2 Simple Pseudo-Synchronization

Making use of the procedure of pseudo-synchronization of clocks, estimations to the shifts S'_{i1} from the base clock 1 to the non-base clock i are shown in Table 4. Compared to S_{i1}, the original shifts from the base clock 1 to a non-base clock i, the estimated shifts (S'_{i1}'s) are over-estimated. The estimation to the clock shifts could be made more close to the actual shifts when the knowledge of minimum transmission delays is available. (ref. the column $S'_{i1} - D_{1i}$ in Table 4)

Under the estimated pseudo-shifts S'_{i1}'s, the occurrence time of events recorded with respect to non-base clocks can be projected onto the timeline with respect to the base clock. Using the new occurrence time of events, violations to direct "happened-before" relations can be evaluated, and numbers of violations are shown in Table 5. It is clear that no violations happen between non-base hosts and the base host because number of violations are all 0 for the row of $i = 1$ and the column of $j = 1$ in Table 5. Meanwhile, it is still possible that violations can happen between non-base hosts.

$i\backslash$	S_{i1}	S'_{i1}	D_{1i}	$S'_{i1} - D_{1i}$
2	1.9800	2.0581	0.0631	1.9950
3	0.2950	0.3679	0.0608	0.3071
4	1.0617	1.4202	0.3495	1.0707
5	0.9566	1.1904	0.2291	0.9613
6	0.3391	0.6293	0.2849	0.3444
7	0.0967	0.5389	0.4302	0.1087
8	1.7971	2.0465	0.2466	1.7999

Table 4: The estimated shifts from the base clock 1 to the non-base clocks i's. (All metrics are in unit of a second.)

$i \setminus j$	1	2	3	4	5	6	7	8
1	0	0	0	0	0	0	0	0
2	0	0	0	27	5	0	18	17
3	0	0	0	0	6	3	25	8
4	0	0	0	0	0	0	0	0
5	0	0	0	0	0	7	0	0
6	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0

Table 5: After non-base clocks are pseudo-synchronized with the base clock, the numbers of violations to direct "happened-before" relations between a pair of hosts i, j.

5.3 Resolving Violations by Adjusting Pseudo-Shifts

Violations to direct "happened-before" relations can be resolved by adjusting the estimated pseudo-shifts shown in Table 4. The adjustments a_{ij}'s are shown in Table 6. A new pseudo-shift S''_{i1} can be obtained by applying the minimum value of a_{ij}'s ($2 \leq j \leq m$) on the initial pseudo-shift S'_{i1}. The values of new pseudo-shifts are shown in Table 7. Indeed, after non-base clocks i's are synchronized with the base clock 1 by pseudo-shifts S''_{i1}'s, all violations to direct "happened-before" relations are eliminated.

5.4 Serialization of Events

After the pseudo-synchronization of non-base clocks with the base clock by shifts S''_{i1}, new occurrence time of events can be evaluated. The comparison of the original occurrence time to the new occurrence time after pseudo-synchronization is shown in Figure 2. The new values of occurrence time of events are very close to the values of their original occurrence time, because a pair of the new and original occurrence time of an event is located very to the diagonal line which illustrates the ideal case when the two time values are equal.

i\j	2	3	4	5	6	7	8
2	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0
4	-0.1799	0	0	0	0	0	0
5	-0.0870	-0.0286	0	0	0	0	0
6	0	-0.0246	0	-0.0308	0	0	0
7	-0.0988	-0.2658	0	0	0	0	0
8	-0.1421	-0.0426	0	0	0	0	0

Table 6: Adjustments to pseudo-shifts. The values of adjustment are in unit of a second.

$\backslash j$	2	3	4	5	6	7	8
S''_{j1}	2.0581	0.3679	1.2403	1.1034	0.5985	0.2731	1.9044
S_{j1}	1.9800	0.2950	1.0617	0.9566	0.3391	0.0967	1.7971
S'_{j1}	2.0581	0.3679	1.4202	1.1904	0.6293	0.5389	2.0465

Table 7: The pseudo-shifts S''_{j1} after the adjustment.

6 Conclution

In this paper, a method of pseudo-synchronization of local clocks in distributed applications is proposed. Pseudo-synchronization of local clocks with a base clock is to estimate the pseudo-shift between each non-base clock and the base clock without violating direct "happened-before" relations. The pseudo-shifts are over-estimated when the knowledge of delays is unknown. Over-estimated values of pseudo-shifts introduce the violations to the direct "happened-before" relations. In order to resolve the violations, adjustment to the pseudo-shifts is necessary. This method has been validated in a distributed application running on 8 hosts. The results show that the values of new occurrence time of events projected onto the base timeline are very close to the values of actual occurrence time with respect to a universal clock.

References

[1] Context-sensitive access control for open mobile agent systems. In *Proceedings of the 3rd International Workshop on Software Engineering for Large-Scale Multi-Agent Systems (SELMAS'2004)*, pages 42–48, Edinburgh, Scotland (UK), May 2004.

[2] Marcos Kawazoe Aguilera, Robert E. Strom, Daniel C. Sturman, Mark Astley, and Tushar Deepak Chandra. Matching events in a content-based subscription system. In *Symposium on Principles of Distributed Computing*, pages 53–61, 1999.

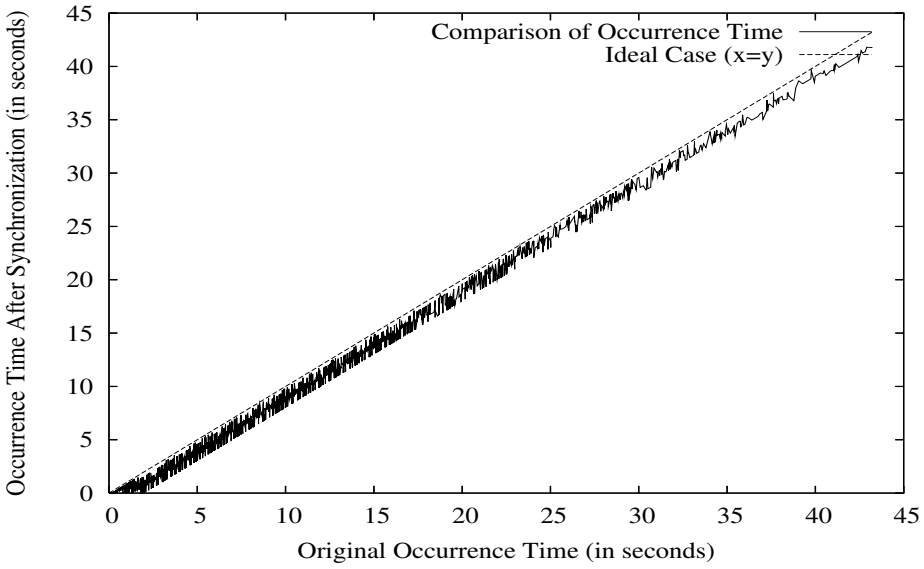

Figure 2: The comparison of the original occurrence time to the new occurrence time after pseudo-synchronization.

[3] Karthikeyan Bhargavan and Carl A. Gunter. Requirements for a practical network event recognition language. In *Proceedings of the Runtime Verification Workshop*, July 2002.

[4] C. Fidge. Timestamps in message-passing systems that preserve the partial ordering. *Australian Computer Science Communications*, 10(1):56–66, February 1988.

[5] John Heidemann, Fabio Silva, and Deborah Estrin. Matching data dissemination algorithms to application requirements. In *Proceedings of the first international conference on Embedded networked sensor systems*, pages 218–229. ACM Press, 2003.

[6] L. Lamport. Time, clocks and the ordering of events in a distributed system. *Communications of the ACM*, 27(7):558–565, July 1978.

[7] Zoltán Ádám Mann. Tracing system-level communication in object-oriented distributed systems. Winner of the 2001 Student Paper Contest of IEEE Hungary Section, 2001.

[8] Friedemann Mattern. Virtual time and global states of distributed systems. In M. Cosnard et. al., editor, *Parallel and Distributed Algorithms: proceedings of the International Workshop on Parallel and Distributed Algorithms*, pages 215–226. Elsevier Science Publishers B. V., 1989.

[9] L. Mummert and M. Satyanarayanan. Long term distributed file reference tracing: Implementation and experience. *Software Practice and Experience*, 26(6):705–736, 1996.

[10] Francesco Quaglia. A scaled version of the elastic time algorithm. In *Proceedings of the fifteenth workshop on Parallel and distributed simulation table of contents*, pages 157 – 164, Lake Arrowhead, California, 2001.

[11] Sudhir Srinivasan and Paul F. Reynolds Jr. NPSI adaptive synchronization algorithms for PDES. In *Winter Simulation Conference*, pages 658 – 665, 1995.

[12] the Cooperative Association for Internet Data Analysis (CAIDA). Round-trip time internet measurements from caida's macroscopic internet topology monitor. http://www.caida.org/analysis/performance/rtt/walrus0202/.