# On Pseudo-synchronizing Clocks in Distributed Applications

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

## Abstract

Many distributed applications demand the underlying distributed systems to accommodate their needs. Matching requirements of distributed applications compels the prompt identification of momentary characteristics of an underlying distributed system. Thus, a method is needed to promptly derive a momentary state of a distributed system by identifying a set of events occurred across hosts and occurred in the neighborhood of the given time moment. We propose a novel method of pseudo-synchronizing local clocks to a common reference clock so to aid the derivation of the set of distributed events occurred at a given time moment. Pseudo-synchronizing local clocks to a common reference clock is discover the shifts between local clocks and the reference clock by making use of the ”happened-before” relations. Then, local clocks can be pseudo-synchronized to the reference clock by compensating the shifts discovered. However, compensating the shifts to local clocks might affect the original order of occurrence of events. In order to maintain the original order, further adjustments to the shifts are needed before compensating the shifts to local clocks.

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. A direct "happened-before" relation involving a pair of events in sending and receiving a message is also called a relation of direct transmission, and the two events are called the sending event and the receiving event, respectively.

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 sequentialized, 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 the synchronized clocks can be used for timing-sensitive applications, like deriving the momentary state of a distributed application at a given time moment. Pseudo-synchronizing one clock to 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 procedure of pseudo-synchronization of clocks is described in Section 2, and the procedure of estimating occurrence time of events with respect to a common reference clock is described in Section 3. The evaluation to the method of deriving momentary states of distributed applications is in Section 4. The previous works relating to this subject in Section 5. Our method is summarized in Section 6.

# 2 Pseudo-Synchronization

## 2.1 Motivation

In a distributed application scenario consisting of $m$ hosts, the occurrence time of an event is expressed as an $m$-dimensional time stamp $(t_1, t_2, \cdots, t_m)_m$, where $t_i$ $(1 \leq i \leq m)$ is the occurrence time of the event with respect to the local clock at host $i$. If all hosts share a universal clock, then the occurrence time of an event recorded by different hosts is identical. When all hosts share a universal clock, the order of occurrence of events could be naturally determined using their occurrence time. However, when local clocks at different

hosts lack of synchronization, an $m$-dimensional time stamp of the occurrence of an event is not directly available from the measurement. For example, when an event is recorded by host 1 at time $l$ with respect to the local clock, its occurrence time with respect to clocks at other hosts is uncertain, and thus, the $m$-dimensional occurrence time of this event is expressed as $(l, -, -, \cdots, -)$ where $-$ denotes an uncertain value. The uncertain values can be determined when all local clocks are synchronized to a common reference clock.

Pseudo-synchronizing a local clock to a reference clock is to find out the shift between the two clocks. The shift between two clocks can estimated by making use of relations of direct transmissions which are "happened-before" relations with one sending event occurring at one host and one receiving event occurring at the other host. The basic idea is that the shift between two clocks can be determined by the minimum difference of the occurrence time between pairs of events having a direct transmission relation. Without loss of generality, the local clock at host 1 is treated as the reference clock in pseudo-synchronization of local clocks, and host 1 is called the reference host. Consecutive events recorded by host $i\,(1 \leq i \leq m)$ can be denoted as a time series, $\{t_1^i, t_2^i, \cdots, t_{p_i}^i\}$, and an event occurred at host $i$ at time $t_v$ $(t_1^i \leq t_v \leq t_{p_i}^i)$ is denoted as $e_{t_v}^i$. A relation of direct transmission involving hosts $i$ and $j$ can be expressed as $e_{t_v}^i \rightarrow e_{t_w}^j\,(i \neq j)$ where $e_{t_v}^i$ and $e_{t_w}^j$ is a sending event at host $i$ and a receiving event at host $j$, respectively.

If the clock at host $i$ is well synchronized to the reference clock, *i.e.* the clock at host 1, $t_v^1$ should precede $t_w^i$ for the two events in $e_{t_v}^1 \rightarrow e_{t_w}^i$, and the difference between $t_v^1$ and $t_w^i$ is the delay between events $e_{t_v}^1$ and $e_{t_w}^i$. Under all circumstances, $e_{t_v}^1$ should always occur before $e_{t_w}^i$, and $t_v^1$ should precede the occurrence time of event $e_{t_w}^i$ with respect to the reference clock. If the transmission delay between events $e_{t_v}^1$ and $e_{t_w}^i$ is assumed to be zero, then the shift between the local clock at site $i$ and the reference clock can be roughly estimated as $t_w^i - t_v^1$. The shift between local clock at host $i$ and the reference clock can be used to adjust the local clock $i$.

However, not every relation of direct transmission is suitable for estimating the shift between two clocks. When an inappropriate relation of direct transmission is used in estimating the shift, some relations of direct transmission might be violated with respect to the reference clock. For example, consider two relations of direct transmission, $e_{t_v}^1 \rightarrow e_{t_w}^i$ and $e_{t_x}^1 \rightarrow e_{t_y}^i$, satisfying $t_v^1 < t_x^1$, $t_w^i < t_y^i$, and $t_x^1 - t_v^1 < t_y^i - t_w^i$. The relation $e_{t_v}^1 \rightarrow e_{t_w}^i$ is appropriate for estimating the shift between clock $i$ and the reference clock, however, the pair of events $e_{t_x}^1 \rightarrow e_{t_y}^i$ is not appropriate for the purpose. This fact can be validated as follows. When the relation $e_{t_v}^1 \rightarrow e_{t_w}^i$ is used, $t_w^i$ is projected onto the reference timeline as $t_v^1$, correspondingly, $t_y^i$ is projected onto the reference timeline as $t_v^1 + (t_y^i - t_w^i)$. Since the relation $e_{t_x}^1 \rightarrow e_{t_y}^i$ has to be maintained, $t_v^1 + (t_y^i - t_w^i) > t_x^1$ is mandated and is guaranteed by the assumption of $t_x^1 - t_v^1 < t_y^i - t_w^i$. However, when the relation $e_{t_x}^1 \rightarrow e_{t_y}^i$ is used for this purpose, the relation $e_{t_v}^1 \rightarrow e_{t_w}^i$ is violated because the occurrence time of event $e_{t_w}^i$ with respect to the reference clock precedes $t_v$.

## 2.2 Procedure

When estimating the shift between the local clock at host $i$ and the reference clock, the estimation procedure takes as input two sequences of consecutive events recorded by host

$i$ and by host 1, respectively. This procedure takes the following steps.

(1) Finding out pairs of events having a relation of direct transmission $e^1_{t_v} \rightarrow e^i_{t_w}$ and denoting them by a set $E_i$ that

$$E_i = \left\{ (e^1_{t_v}, e^i_{t_w}) \middle| e^1_{t_v} \rightarrow e^i_{t_w}, t^1_1 \leq t^1_v \leq t^1_{p_1}, t^i_1 \leq t^i_w \leq t^i_{p_i} \right\};$$

(2) Computing the difference of occurrence time between each pair of events in $E_i$, *i.e.* $t^i_w - t^1_v$ for $(e^1_{t_v}, e^i_{t_w}) \in E_i$;

(3) Taking the minimum value of these differences as the shift between two clocks, *i.e.* $\min\limits_{(e^1_{t_v}, e^i_{t_w}) \in E_i} (t^i_w - t^1_v)$.

The validation of this procedure is shown in the Appendix.
The processing time spent in pseudo-synchronizing two clocks is the amount of time needed to scan through both sequences of consecutive events recorded by respective host. The procedure of estimating the shift between two clocks can be performed on-line, which keeps the running minimum value of differences while a distributed application progresses. After all local clocks have been pseudo-synchronized to the reference clock, the occurrence time of an event with respect to each individual local clock can be determined, hence, an $m$-dimensional occurrence time stamp for the occurrence of each event can be obtained. It is worthy of noting that, under this estimation procedure, the occurrence time of an event with respect to remote clocks might not be accurate because the shift between two clocks could be over-estimated. Moreover, pseudo-synchronizing local clocks to a reference clock could only guarantee that relations of direct transmission involving the reference host are maintained, and it is possible that some relations of direct transmission between non-reference hosts could be violated. This fact is stated more specifically in Proposition 2.1.

**Proposition 2.1** When local clocks at host $i$ and $j$ $(2 \leq i, j \leq m; i \neq j)$ are pseudo-synchronized to the reference clock at host 1, the shift between clock $i$ and clock 1 can be denoted as $d_i = \min\limits_{e^1_{t_v} \rightarrow e^i_{t_w}} (t^i_w - t^1_v)$, and the shift between clock $j$ and clock 1 can be denoted as $d_j = \min\limits_{e^1_{t_v} \rightarrow e^j_{t_w}} (t^j_w - t^1_v)$. Relations of direct transmission between host $i$ and $j$ are maintained only if $\min\limits_{e^i_{t_p} \rightarrow e^j_{t_q}} (t^j_q - t^i_p) \geq (d_j - d_i)$. □

## 2.3 Discussion

Due to lack of knowledge of the actual minimum transmission delay between two hosts, it is difficult to precisely estimate the shift between two clocks by the procedure of pseudo-synchronization. In most application scenarios, the transmission delay between two hosts should not be very large. 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, a value of zero is a close estimate of the transmission delay between two hosts when the actual transmission delay is reasonably short.

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.

Over-estimating the shift $d$ between two clocks is a potential problem for the procedure of pseudo-synchronization. The fact that the shift between two clocks might be over-estimated can be formally justified as follows. We assume that $d$ is derived from a relation of direct transmission in the form of $e^1_{t_v} \rightarrow e^i_{t_w}$, *i.e.* $d = t^i_w - t^1_v$. We denote the actual shift between the two clocks to be $d'$, and the actual transmission delay is denoted as $\delta\,(\delta > 0)$. Following the same reasoning, we have $t^i_w - (t^1_v + \delta) = d'$ and further have $d = d' + \delta$. Hence, $d > d'$. If the actual transmission delay $\delta$ is available, then the shift between two clocks becomes $d - \delta$.

# 3 Deriving the Momentary Snapshot

When the current state of a distributed application consisting of $m$ hosts is to be discovered at a host, the local clock at the host is treated as the reference clock and other clocks are pseudo-synchronized to the reference clock. After the pseudo-synchronization is performed, a set of events occurred in the neighborhood of a time point with respect to the reference clock is to be extracted. Then, the occurrence time of the events extracted is estimated with respect to the reference clock. Lastly, the set of events is naturally serialized with respect to their estimated occurrence time and is treated as the snapshot of the distributed application at the given moment.

## 3.1 Motivation

Without loss of generality, it is assumed that the current state of a distributed application at a time moment $t_{q_1}$ is to be discovered at host $1$. The clock at host 1 is treated as the reference clock. This task is performed by first extracting a set of events occurring within a small time interval $[t_{q_1} - u/2, t_{q_1} + u/2]\,(u > 0)$ with respect to the reference clock; then, the occurrence time of this set of events is estimated with respect to the reference clock. Hence, the derivation of the state of the distributed application at the time moment $t_{q_1}$ only involves events occurring within time interval $[t_{q_1} - u/2, t_{q_1} + u/2]$. This set of events can only be identified after all non-reference clocks are pseudo-synchronized to the reference clock. Under the pseudo-synchronized clocks, the time point $t_{q_1}$ can be projected as $t_{q_i} = t_{q_1} + d_i$ with respect to clock $i$ where $d_i$ is the shift between clock $i\,(2 \leq i \leq m)$ and the reference clock. If a set of events occurring at host $i\,(1 \leq i \leq m)$ within time interval $[t_{q_i} - u/2, t_{q_i} + u/2]$ with respect to clock $i$ is denoted as $E_i$, then the union of $E_i$'s, denoted as $E = \bigcup_{i=1}^{m} E_i$, constitutes a set of events occurring, across all hosts, within time interval $[t_{q_1} - u/2, t_{q_1} + u/2]$ with respect to the reference clock.

If there is no violated relations of direct transmission after pseudo-synchronization of clocks, then the set of events derived is the desired snapshot of the distributed application

at the given moment. However, some relations of a direct transmission could be violated under pseudo-synchronized clocks due to a fact stated in Proposition 2.1. In order to overcome violated relations of direct transmission, some adjustments need to be made. For example, for a relation of $e^i_{t_v} \rightarrow e^j_{t_w}$, the occurrence time of the pair of events is projected onto the reference timeline as $t^i_v - d_i$ and $t^j_w - d_j$, respectively, and this relation is violated if $t^i_v - d_i > t^j_w - d_j$. In order to overcome the violation, the shift between clock $j$ and the reference clock needs to be adjusted by a minimum amount of $d_j - d_i - (t^j_w - t^i_v)$. Consequently, the new shift between the clock $j$ and the reference clock becomes $d''_j = d_i + (t^j_w - t^i_v)$.
Adjusting the shift between a local clock and the reference clock could also possibly introduce new violations. Therefore, long processing time could be resulted in order to fully resolve violations. In order to limit the processing time used in resolving violations, the occurrence time of events involved in violated relations of direct transmission need to be selectively adjusted. Selective adjustments on occurrence time should not introduce new violations.

## 3.2 Procedure

The procedure of deriving a momentary snapshot of a distributed application assumes that all local clocks have been pseudo-synchronized to the reference clock. This procedure consists of four steps. In the first step, the occurrence time of every event in $E$ is projected onto the timeline with respect to the reference clock. In the second step, violated relations of direct transmission are identified based on the occurrence time projected onto the reference timeline. In the third step, the adjustments on shifts between local clocks and the reference clock are determined. In the fourth step, selective adjustments on the occurrence time of events are made in order to resolve violations. In order to estimate the amount of processing time in each step, the number of events included in set $E$ is assumed to be $N$.
In the first step, the occurrence time of an event $e^j_t \in E_j\,(2 \leq j \leq m)$ is projected into $t^j - d_j$ with respect to the reference clock. The amount of processing time in this step is $\mathcal{O}(N)$.
In the second step, the set $V$ of violated relations of direct transmission under projected occurrence time is identified as

$$V = \left\{ (e^i_{t_v}, e^j_{t_w}) \middle| e^i_{t_v} \rightarrow e^j_{t_w}; 2 \leq i, j \leq m; t^i_v - d_i > t^j_w - d_j \right\}.$$

The set $V$ can be partitioned into $m-1$ components each of which is a set of event pairs with the trailing event occurred at a same host. For example, the $j$-th component $(2 \leq j \leq m)$ is denoted as

$$V_j = \left\{ (e^i_{t_v}, e^j_{t_w}) \middle| (e^i_{t_v} \rightarrow e^j_{t_w}) \in V, 2 \leq i \leq m; t^i_v - d_i > t^j_w - d_j \right\}.$$

Hence, $V = \bigcup_{2 \leq j \leq m} V_j$ and $V_i \bigcap V_j = \emptyset\,(\forall i \neq j)$. In order to find out items in set $V_j$, each event in $E_j$ is scanned exactly once, in turn, the processing time in the second step is $\mathcal{O}(N)$.
In the third step, a new shift between clock $j$ and the reference clock is estimated. For each element $(e^i_{t_v}, e^j_{t_w}) \in V_j$, $d_i + (t^j_w - t^i_v)$ is computed. Furthermore,

$$d''_j = \min_{(e^i_{t_v}, e^j_{t_w}) \in V_j} [d_i + (t^j_w - t^i_v)] \tag{1}$$

is treated as the new shift between clock $j$ and the reference clock. Under the new shift $d''_j$, the occurrence time of event $e^j_u$ is projected into $u^j - d''_j$ with respect to the reference clock. It can be shown that $u^j - d''_j > u^j - d_j$, *i.e.* an over-estimated shift is adjusted into a smaller value of shift and the new projected occurrence time of event $e^j_u$ is bigger than the original projection. Under new shifts, no all violated relations of direct transmission can be resolved, and the projected occurrence time needs to be further adjusted using a method described in step four. The processing time of the third step is also $\mathcal{O}(N)$.

In the fourth step, projections of occurrence time is adjusted in order to resolve violated relations of direct transmission. The adjusted occurrence time of events $e^j_t$ with respect to the reference clock is denoted as $T(e^j_t)$, and an event $e^j_t$ obtains its initial $T(e^j_t)$ as $t^j - d_j$ $(d_1 = 0)$. The adjustment aims to satisfy two conditions:

(i) $T(e^j_{u'}) > T(e^j_u)$ if event $e^j_u$ occurs prior to event $e^j_{u'}$ at host $j$;

(ii) $T(e^j_{t_w}) \geq T(e^i_{t_v})$ if $e^i_{t_v} \rightarrow e^j_{t_w}$ $(2 \leq i, j \leq m$ and $i \neq j)$.

Condition (i) states that the adjustment should not revert the order of occurrence for events occurred at a same host, and condition (ii) states that the adjustment should serve to resolve violations of relations of direct transmission. In order to satisfy condition (i), a barrier $B_j$ is set on the timeline with respect to clock $j$ to prevent introducing new violations. The barrier $B_j$ is initially set to be the right border of the interval, *i.e.* $t^1_0 + u/2$ and moves to the left as projections of occurrence time is adjusted.

In the adjustment procedure, relations of direct "happened-before" among the $N$ events are examined by scanning events in the order from later occurrence time to earlier occurrence time, thus, the $B_j$'s are initialized as the right boundary of the time interval with respect to the reference clock.

For each event under scanning, the following operations are performed:

(1) `if` event $e^j_u$ is the leading event in a non-violated relation of direct "happened-before" in the form of $e^j_u \rightarrow e^k_{u'}$ $(1 \leq k \leq m)$

(2) `then` $T(e^j_u) = \min\{T(e^j_u) + (d_j - d''_j), T(e^k_{u'}), B_j\}$;

(3) `else` $T(e^j_u) = \min\{T(e^j_u) + (d_j - d''_j), B_j\}$;

(4) `endif`

(5) $B_j = T(e^j_u)$.

Under this adjustment procedure, no new violations to relations of direct transmission will be introduced. However, it is still possible that some violations still exist after the adjustment. In order to eliminate all violations, the $N$ events are scanned for a second round to resolve violations. A violated relation of direct transmission could be resolved by simply swapping the occurrence time of the two events. The argument is as follows. The time interval between the occurrence time of two events in a violated relation of direct transmission can be imagined to first condense into a single point, and no new violation is introduced in the condensation because the order of occurrence of events does not change due to the condensation. Next, the condensed single time point is expanded into the original time interval

by simply swaping the occurrence time of the two events occurred at the boundary of the interval, and keeping the occurrence time of events occurred within this time interval intact. It is clear that no new violation to relations of direct transmission is introduced in resolving this violated relations of direct transmission. The processing time in the fourth step is still $\mathcal{O}(N)$. Hence, the overall processing time in the point-wise serialization is $\mathcal{O}(N)$.

# 4 Evaluation

The method of deriving momentary snapshot of a distributed application has also been validated by experiments. The goal of this evaluation is to examine the effectiveness this method in deriving the set of events occurred at a given moment in order to constitute the snapshot. A distributed application runs on $5$ hosts each of which records events occurring locally. Each host is equipped with its own clock which is out of synchronization with other clocks. The state of the distributed application at time moment $5$s is to be derived at host 1. Thus, the clock at host 1 is used as the reference clock, and host 1 is the reference host. The non-reference clocks are set unsynchronized to the reference clock with their shifts to the reference clock shown as $d'_i$ in Table 1. The shifts are drawn uniformly in $[0, 1.0s]$.

| host $i$ | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| $a_i$ | 0.0036 | 0.0006 | 0.0024 | 0.0059 | 0.0067 |
| $d'_i$ | n/a | 0.1997 | 0.3135 | 0.5890 | 0.4713 |

Table 1: The parameters used in generating event trace at each host.

## 4.1 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 $[0, 0.5s]$.

| S\D | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | n/a | 0.3393 | 0.0182 | 0.1833 | 0.0478 |
| 2 | 0.2601 | n/a | 0.3358 | 0.1458 | 0.0668 |
| 3 | 0.1364 | 0.0787 | n/a | 0.4961 | 0.2113 |
| 4 | 0.3343 | 0.3013 | 0.2637 | n/a | 0.2485 |
| 5 | 0.3161 | 0.2584 | 0.3554 | 0.4082 | n/a |

Table 2: The minimum transmission delays (in seconds) between an arbitrary pair of hosts.

Before any processing is made by our method, the number of violated relations of direct transmission between an arbitrary pair of hosts is shown in Table 3, when the occurrence of an event is time-stamped with respect to the local clock at the host where the event occurs.

| $i \setminus j$ | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 | 0 |
| 2 | 0 | 0 | 0 | 0 | 0 |
| 3 | 70 | 8 | 0 | 0 | 0 |
| 4 | 94 | 42 | 5 | 0 | 0 |
| 5 | 65 | 8 | 0 | 0 | 0 |

Table 3: Under unsynchronized clocks, numbers of violations to relations of direct transmission between an arbitrary pair of hosts $i, j$ $(i \neq j)$ in the form of $e_t^i \rightarrow e_{t'}^j$.

## 4.2 Effect of Pseudo-Synchronization

Under the procedure of pseudo-synchronization of clocks, the shifts between non-reference clocks and the reference clock can be estimated and are shown in Table 4. $d_i$ is the estimated shifts between clock $i$ and the reference clock, *i.e.* clock $1$. Compared to $d_i'$ (the original shifts between clock $i$'s and the reference clock), the estimated shifts ($d_i$'s) have been over-estimated by offsetting the original shifts ($d_i'$) by minimum transmission delays ($\min \delta_{1,i}$) as shown by $d_i' + \min \delta_{1,i}$ in Table 4. The shifts could be more accurately estimated when the knowledge of minimum transmission delays is available.

| $i\setminus$ | $d_i$ | $d_i'$ | $\min \delta_{1,i}$ | $d_i' + \min \delta_{1,i}$ |
|---|---|---|---|---|
| 2 | 0.5407 | 0.1997 | 0.3393 | 0.5390 |
| 3 | 0.3325 | 0.3135 | 0.0182 | 0.3316 |
| 4 | 0.7741 | 0.5890 | 0.1833 | 0.7723 |
| 5 | 0.5227 | 0.4713 | 0.0478 | 0.5191 |

Table 4: The estimated shifts between non-reference clocks to the reference clock. (All metrics are in unit of second.)

After non-reference clocks are pseudo-synchronized to the reference clock, the occurrence time of events recorded at non-reference hosts can be estimated with respect to the reference clock. Hence, violations to relations of direct transmission can be evaluated under the estimated occurrence time of events, and numbers of violations are shown in Table 5. It is clear that no violations happen between non-reference hosts and the reference host because number of violations are all $0$ for the row of $i = 1$ and the column $j = 1$ in Table 5, *i.e.* there is no violation to relations in the form of $e_t^1 \rightarrow e_{t'}^i$ or $e_t^j \rightarrow e_{t'}^1$. Meanwhile, it is still possible that violations happen between non-reference hosts.

| $i \setminus j$ | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 | 0 |
| 2 | 0 | 0 | 0 | 0 | 0 |
| 3 | 0 | 96 | 0 | 0 | 0 |
| 4 | 0 | 0 | 0 | 0 | 0 |
| 5 | 0 | 20 | 0 | 0 | 0 |

Table 5: Under pseudo-synchronized clocks, numbers of violations to relations of direct transmission between an arbitrary pair of hosts $i, j$ $(i \neq j)$ in the form of $e_t^i \rightarrow e_{t'}^j$.

## 4.3 Extraction of Events Occurred Around A Time Moment

In order to derive the snapshot at a given time moment, a small set of events occurred in a small time interval centered at the given moment needs to be derived. In our evaluation, the time moment is set to be $5$s with respect to the reference clock, and the size of the time interval is set to be $0.5$s. The numbers of events occurred within this interval with respect to the reference clock is shown in Table 6, in contrast to the total number of events occurred at each host with the duration of the experiment. Indeed, compared to numbers of events occurred in the duration of the experiment, only a small number of events occurred at each host need to be considered in forming the set of events occurred in the neighborhood of the given time moment.

| $\setminus i$ | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| $[4.75s, 5.25s]$ | 34 | 38 | 40 | 26 | 42 |
| Overall | 1396 | 1384 | 1371 | 1399 | 1376 |

Table 6: Numbers of events occurred within the time interval $[4.75s, 5.25s]$ and numbers of events occurred in the duration of the experiment.

## 4.4 Resolving Violations in A Small Set of Events

In order to derive the snapshot at a given time moment, violations to relations of direct transmission (only involving events in the small set) between non-reference hosts are to be

eliminated. Hence, the estimated shifts shown in Table 4 need to be adjusted by applying Equation (1), and new shifts $d_i''$'s between non-reference clocks $i$'s and the reference clock are estimated. The adjustments to shifts $d_i$'s are shown in Table 7. The adjustments $d_i - d_i''$ can be applied to adjust occurrence time of events in the small set using the procedure described in step four in Section 3. After the occurrence time of events in the derived set has been adjusted, all violations to relations of direct transmission (only involving events in the small set) can be resolved (shown in Table 8).

| $\backslash i$ | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| $d_i - d_i''$ | n/a | 0.1066 | 0 | 0 | 0 |

Table 7: Adjustments to shifts estimated in Table 4.

| $i \setminus j$ | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 | 0 |
| 2 | 0 | 0 | 0 | 0 | 0 |
| 3 | 0 | 0 | 0 | 0 | 0 |
| 4 | 0 | 0 | 0 | 0 | 0 |
| 5 | 0 | 0 | 0 | 0 | 0 |

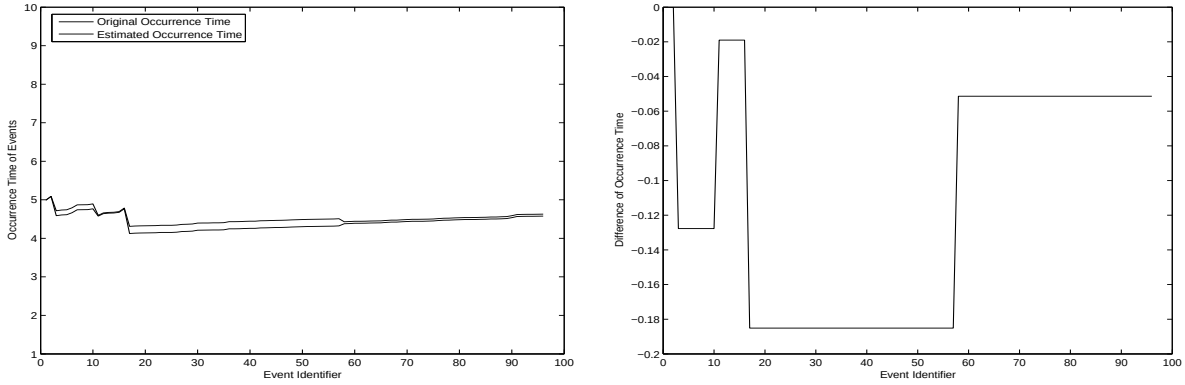
Table 8: After the adjustment to occurrence time of events occurred in $[4.75s, 5.25s]$, numbers of violations to relations of direct transmission are all 0's.

## 4.5 Serialization of Events

After violations to relations of direct transmission have been resolved, events occurred in the neighborhood of the given time moment are naturally serialized with respect to the reference clock. The occurrence time of an event after adjustments is expected to be close to the occurrence time with respect to a conjectured universal clock. In the experiment, a universal clock is assumed for verification purpose, and the occurrence time of each event with respect to the universal clock is recorded, as well as the occurrence time with respect to the local clock at the host where the event occurred. In our experiments, the clock at host 1 is used as the universal clock. Hence, it is possible to compare the degree of similarity between the estimated occurrence time of an event and the recorded occurrence time with respect to the universal clock. Shown in Figure 1, the estimated occurrence time stamps of events occurred within $[4.75s, 5.25s]$ are very close to their ideal occurrence time stamps, and the difference between an estimated occurrence time stamp and the corresponding occurrence time stamp is small.

## 4.6 Overhead

The processing time in deriving the snapshot occurred within a small time interval consists of the processing time for pseudo-synchronizing non-reference clocks to the refer-




(a) The occurrence time estimated vs. ideal

(b) Estimation error (estimated occurrence time - ideal occurrence time)

Figure 1: The comparison between the estimated occurrence time and the ideal occurrence time with respect to a universal clock. The occurrence time is only for those events occurred in $[4.75s, 5.25s]$. The identifiers of these events are shown on the $x$-axis, and the occurrence time is shown on the $y$-axis.

ence clock and the processing time for adjusting occurrence time of events under pseudo-synchronized clocks. Even though the processing time spent in pseudo-synchronization could be long since long event traces recorded at participating hosts need to be scanned to discover the minimum difference between the occurrence time of a RECEIVE event and the occurrence time of the corresponding SEND event, but this operation can be made on-line such that a running minimum difference is maintained as the distributed application progresses. Hence, the processing time in discovering shifts between clocks can be amortized into the processing demand for each operation of deriving a snapshot. After the pseudo-synchronization, all processing for serialization only involve a small number of events compared to the number of overall event occurrence (see Table 6), thus, the processing overhead is low for each derivation of a snapshot. Therefore, the overall processing overhead in point-wise serializing events is low.

# 5 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.

# 6 Conclution

In this paper, a method of pseudo-synchronizing local clocks in distributed applications is proposed. Pseudo-synchronization of local clocks to a common reference clock is to estimate the shifts between local clocks to the reference clock without violating relations of direct transmission. The accuracy of the estimated shifts affect the quality of estimation on the occurrence time of events based on the pseudo-synchronized clocks. The cause to inaccurate estimates of shifts is due to unknown transmission delays between hosts. In order to prevent relations of direct transmission from being violated, adjustments on the occurrence time of events with respect to the reference clock are performed after pseudo-synchronization of clocks. This method has been validated in an application scenario distributed on $5$ hosts. The results demonstrate its effectiveness in that the estimated occurrence time of events occurred in the neighborhood of a given time point is very close to actual occurrence time of these events.

# 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.

[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/.