PAPER

Online Timing Correlation of Streaming Data with Uncertain Timestamps*

Chan-gun LEE†, *Member*, **Aloysius K. MOK**††, *and* **Prabhudev KONANA**††, *Nonmembers*

SUMMARY We introduce the interval timing correlation, which can establish timing correlation conditions to handle interval timing timestamps. Interval timestamps are adopted to handle the temporal uncertainties in the timestamps of stream data. A probabilistic querying approach is taken in order to support timing predicates such as deadline, delay, and within over interval timestamps. A timing correlation condition entails a desired confidence threshold (minimum satisfaction probability). We define the interval timing correlation and discuss how to implement the algorithm. We present an analysis result which can effectively identify only tuples that need to be considered in determining the correlation. The performance of the proposed algorithm is shown.

key words: *timing, correlation, uncertainty, timestamp, interval*

1. Introduction

There are many emerging applications requiring the functionality of processing streaming data with timestamps. Typical applications are sensor data processing, position tracking for moving objects, log analysis for telephone records, network packet analysis, and stock trend analysis. The streaming data are semi-ordered by their timestamps and arrive in real-time [1].

We address the problem of determining the timing correlation of streaming data especially with interval timestamps in this paper. The timing correlation resembles typical stream join operator but it is extended to handle the temporal uncertainty. We can specify the degree of certainty that a temporal condition should satisfy, which is assessed over the pairs of interval timestamps of streaming data from different sources. We adopt interval timestamps rather than point-based timestamps because the former can represent the temporal uncertainties that may reside in the timestamps of stream data. The uncertainties can be included due to various reasons such as different granularities for time stamping between systems, imprecise occurrence times of events, etc [2].

The point-based timestamps fail to capture the uncertainty information in their representation and may lead the system to produce erroneous answers regarding the temporal properties. As noted in [2], few studies have been done to support incomplete temporal information (temporal indeterminacy) for database systems compared to the abundance of research on the incomplete information. This trend seems to continue in research on streaming data processing despite the imperative of supports for the emerging applications. We imagine there will be many practical situations where the data from sources of stream data, typically sensors, have incomplete temporal information due to the inherent limitations of their hardware or unexpected hardships from its running environment.

Before presenting our approach in detail, we present a scenario where the timing correlation can be applied. Assume that there are two types of sensors, each of which is designed for sensing temperature and noise respectively. They are randomly distributed in a region. Each sensor emits an event containing the measured value and time whenever the measured value deems significant. The limitations of the sensors force the timestamps of the events to have interval forms. We want to collect only the cases where the two different type of sensor detect an object simultaneously, e.g. each detection time is within 2 second. Since the timestamp is in the interval form, we also want to limit to the cases where the satisfaction probability is over 60 percent.

The above example requires handling streaming data and temporal uncertainties. At first glance, the studies on the stream query processing [3] for incomplete values may seem to be directly applied to the above example. In case the processing is done in off-line, obviously timestamps can be treated in the same manner as regular attributes; however, timestamps should be handled differently in case of on-line processing. There are extra information (or constraints) we can exploit when processing is done in on-line. For example, "current time" is always related to the timestamps of "currently incoming" stream data. There must be some bounds on the maximum discrepancies between timestamps of data from different streams. When to discard stream data from the stream buffers is also dependent on their timestamps. We shall use these properties in designing the algorithm for timing correlations later in this paper.

An interval timestamp consists of two time points delimiting the possible occurrence ranges of a data. The probability distribution in an interval is assumed to be uniform; any time point in it has the same chance of occurrence.

As mentioned above, the timestamps of stream data are intervals; thus, we need to devise a mechanism to quantitatively compare interval timestamps. We adopt a probabilistic approach in evaluating and assessing timing conditions

Manuscript received April 25, 2008.
Manuscript revised February 11, 2009.

†The author is with the Faculty of Chung-Ang University, Seoul, 156–756 Korea.
††The authors are with the Faculty of Univ. of Texas at Austin, TX 78712, U.S.A.

*A preliminary version of this paper is included in Lee's dissertation.

DOI: 10.1587/transinf.E92.D.1260

on timestamps of data. For example, we can say that a timestamp I_1 is less than I_2 by 30 percent if the probability of the occurrence of such an event is 30 percent. We shall show how to compute such a probability[†] in Sect. 3.

Main contributions made in this paper are shown in the following:

- Defining the interval timing correlation and showing how to calculate the satisfaction probabilities of timing predicate over interval timestamps
- Analyzing the satisfaction probability for the "within" predicate
- Showing the effectiveness of using the analysis of satisfaction probability in performing interval timing correlations

The rest of this paper is organized as follows. Section 2 presents related work. In Sect. 3, we define the interval timing correlation and present how to evaluate timing predicates. Section 4 presents the analysis result of the satisfaction probability. In Sect. 5 we validate the idea of using the analysis result in performing interval timing correlations. Future work and summary are presented in Sect. 6.

2. Related Work

Recently there have been many efforts for developing various components and architectures of stream data management systems (SDMSs). We shall briefly introduce only some of them which discuss stream join operators. Interested readers are invited to refer to [4], [5] for the extensive introduction to stream data processing and recent advances.

In Aurora [6], users can express their queries by arranging boxes and connecting them with links. Each box corresponds to a stream query operator and each link directs a stream output of a box to an input of another box. A join operator includes parameters such as a join predicate, a time window size, and ordering information about each stream. Ordering information about a stream specifies the tolerance of disorder[††].

Hammad et al. [7] introduces variations of sliding window joins (referred to as W-joins). Two types of multiway join algorithms, the BEW-join and the FEW-join, are shown and their characteristics are discussed. Their underlying assumption is that streaming data are always delivered in timestamp-sorted order although there can be delays in transmission; there is no out-of-order data. Utilizing this assumed property, the FEW-join computes a time point F_t based on a "period" composed of tuples from all streams; if a newly arrived tuple t has the timestamp earlier than the F_t, then t is guaranteed to join with other data in the period. Hence repeated evaluations for the time window can be avoided.

Kang et al. [8] designs a unit-time basis cost model for sliding window joins and study efficient join strategies based on the analysis of their cost model and simulation results. In addition, effective resource-allocation schemes for improving the efficiency of the join processing under various scenarios are presented.

Srivastava and Widom [9] addresses the need for heartbeat in stream data management systems. On receiving a heartbeat with a timestamp τ from a stream, SDMS can assure that any incoming data from the stream will have timestamps greater than τ. A novel modeling technique capturing the bounds of timestamp skew, out-of-ordered timestamps, and transmission latency is introduced and a heartbeat generation algorithm is shown.

In [2] there is a study similar to our problem in the context of valid-time databases. In valid-time databases, a data is associated with a timestamp during which the data is valid. In order to cover indeterminacy in timestamps, the authors adopt a probabilistic approach like ours. They present an algorithm to compute the probability of the comparison operator *before* for interval timestamps with arbitrary probability distributions. Note that their algorithm is designed for off-line use. Hence timestamps are treated as regular attributes in their algorithm..

In order to handle the inherent uncertainties in timestamps, we adopt interval timestamps and assume that the probability distribution in a given interval is uniform. In our own earlier work [10], [11], we proposed the algorithms for monitoring timing constraints where timestamps of events are time intervals. [10] presents two new modalities, *certain* and *possible* for specifying timing constraints on interval timestamps. [11] adopts a probabilistic approach in monitoring timing constraints. A timing constraint is defined over interval timestamps with a deadline/delay and a desired minimum satisfaction probability. The formulae for calculating the satisfaction probability from given two interval timestamps and a deadline/delay constraint are shown.

Recently Mok et al. [12] proposes a general approach for probabilistic timing join over streaming data with uncertain timestamps. In contrast to the approach, we limit our focus to special cases where the probabilities of the timing uncertainties can be modeled in uniform distributions in this paper. The solutions presented in this paper are in closed form and are easy to use in practice. Various implementations of the timing correlation are discussed and their performances are compared.

3. Interval Timing Correlation for Streaming Data

Given two sets of stream data with interval timestamps, a typical condition for interval timing correlation is specified with a timing predicate on interval timestamps and a minimum satisfaction probability[†††] for which the timing predicate should meet. An interval timing correlation produces streams of joined data satisfying the timing condition with a probability not less than the confidence threshold.

Each data from stream sources comes with its own in-

[†]We use the term satisfaction probability.

[††]slack in [6] is different from our maximum latency in that slack is defined in terms of the maximum number of tuples which can be out of order.

[†††]It is referred to as a confidence threshold later.

terval timestamp. There are two possible ways to timestamp a stream data; at the stream processor or at the source of stream data. Our assumption is that the time stamping is done at the source of stream data, which we believe is more realistic and has many relevant applications.

Before presenting the definition of the interval timing correlation formally, let us introduce an example and explain its meaning informally. The following SQL-like statement is an example query of the interval timing correlation:

```
SELECT *
FROM Stream A, Stream B
WHERE | @A - @B | <= 10 sec over 80%
DELAY Stream A < 1 sec, Stream B < 2 sec
```

It specifies that we want to join stream data if their timing difference is less than 10 seconds and the satisfaction probability is at least 80 percent. It also states that there can be delays during the transmission to the stream data management system from the sources; the maximum one second for the data from stream A and two seconds from stream B.

We use the event model proposed in [10] to capture the uncertainties in event occurrence times. In this event model, the timestamp of an event is represented as a time interval, which consists of a pair of time values (min_time, max_time) where min_time is the earliest possible time of occurrence of the event and max_time is the latest possible time of occurrence of the event. The probability distribution in an interval is assumed to be uniform. **min(I)** function is defined to extract the *min_time* from the timestamp $I = (min\_time, max\_time)$. Similarly, **max(I)** function returns *max_time*. **length(I)** function returns the value $max(I) - min(I)$. Two important parameters needed to be determined in system design time are π and ρ. They represent the maximum and the minimum possible length of any timestamp in the system respectively. Formally, $\forall I$, $\rho \leq len(I) \leq \pi$ where I is an interval timestamp.

Definition 1: We define three types of **timing predicates**; **within** (e.g., $|I_1 - I_2| \leq d$), **deadline** (e.g., $I_1 + d \geq I_2$), and **delay** (e.g., $I_2 - d \geq I_1$) where I_1 and I_2 are timestamps of data. d is zero or an positive constant. The **within** predicate imposes a mutual range condition d on two timestamps. The **deadline** predicate requires the timestamp I_2 to be no later than I_1 by d. The **delay** predicate requires the timestamp I_2 not to be within d after timestamp I_1.

Definition 2: Satisfaction probability of a timing predicate tp, $prob(tp)$ is the probability for which tp is satisfied.

Definition 3: A **timing condition** is a pair of tp and ct, denoted by (tp, ct), where tp is a timing predicate and ct is a confidence threshold requirement for tp ranging from 0% to 100%. A timing condition (tp, ct) is satisfied iff $prob(tp) \geq ct$. A timing condition (tp, ct) is violated iff $prob(tp) < ct$.

In what follows, we use the symbol "@" in front of a stream name to denote the timestamp of any tuple from that stream. In case "@" is used with a tuple e, it means the timestamp of the tuple e. For example a timing correlation condition $(|@A-@B| < 30, 50\%)$ means that we want to join any tuple pairs (e_a, e_b) where e_a and e_b are tuples from stream A and B respectively and the difference of their timestamps is less than 30 with the minimum satisfaction probability of 50%. A timing predicate $@e_1 + 10 \geq @e_2$ specifies a deadline 10 from $@e_1$ to $@e_2$.

We now show how to calculate the satisfaction probability of a within predicate. The main idea is to convert a within predicate into two deadline predicates by following the definition of the within predicate.

Theorem 1: Assume there are timing predicates $c : |I_1 - I_2| \leq d$, $c_1 : I_1 + d \geq I_2$, and $c_2 : I_2 + d \geq I_1$. Then $prob(c) = prob(c_1) * prob(c_2|c_1) = prob(c_2) * prob(c_1|c_2)$.

In this paper we assume that any deadlines specified in timing predicates are larger than π the maximum interval length in the system. Similarly any delays are smaller than $-\pi$. This assumption makes the computation of a within predicate simple as shown in the following:

Corollary 1: Given two timestamps I_1 and I_2, $prob(|I_1 - I_2| \leq d) = prob(I_1 + d \geq I_2)$ if $min(I_1) \leq min(I_2)$ and $d \geq \pi$.
Proof:
$min(I_1) + \pi \leq min(I_2) + \pi \leq min(I_2) + d$ and $max(I_1) \leq min(I_1) + \pi$. Hence, $max(I_1) \leq min(I_2) + d$. Therefore, $prob(I_2 + d \geq I_1) = 100\% \geq prob(I_1 + d \geq I_2)$ holds. By Theorem 1, $prob(|I_1 - I_2| \leq d) = prob(I_1 + d \geq I_2)$. □

4. Efficient Timing Correlation

Assume an interval timing correlation condition $c = (|@A - @B| \leq d, ct)$ is given for streams A and B. Upon receiving a tuple e_1 with a timestamp I_1 from stream A[†], we need to do the following three jobs.

1. Insert e_1 into the buffer for stream A.
2. Find tuples from stream B satisfying the timing condition c with I_1 and output the paired results.
3. Drop expired tuples from the buffers.

In what follows, we call the newly arrived tuple, e.g. tuple e_1 in the above example, *base tuple* and the stream from which the base tuple came is referred to as *base stream*. Assuming that there are two streams, the other stream is called *target stream*.

To improve the performance of the interval timing correlation, we present an efficient technique; upon receiving a base tuple with a timestamp I_1 from stream A, the proposed technique finds the regions in the target stream B determining the satisfiability of the within timing condition, $c : (|@A - @B| \leq d, ct)$. Specifically, we identify three regions R_s, R_v, and R_p in the target stream and they have the following properties:

1. A target tuple with a timestamp I_2 belonging to R_s is guaranteed to satisfy the given timing condition c, i.e. $prob(|I_1 - I_2| \leq d) \geq ct$.

[†]The case receiving a data from stream B is symmetric, hence is omitted here.

2. A target tuple with a timestamp I_2 belonging to R_v is guaranteed to violate the given timing condition c, i.e. $prob(|I_1 - I_2| \leq d) < ct$.
3. A target tuple with a timestamp I_2 belonging to R_p cannot be decided unless the satisfaction probability is computed.

We define that a timestamp I belongs to a range $[t_{min}, t_{max}]$ iff $t_{min} \leq max(I) \leq t_{max}$, $(t_{min}, t_{max}]$ iff $t_{min} < max(I) \leq t_{max}$, $[t_{min}, t_{max})$ iff $t_{min} \leq max(I) < t_{max}$, and (t_{min}, t_{max}) iff $t_{min} < max(I) < t_{max}$ respectively. A tuple with a timestamp I belongs to a region $R = \{r_1, r_2, \ldots, r_n\}$ iff $\exists i$ such that I belongs to a range r_i where $1 \leq i \leq n$.

To find such regions for interval timestamps, we extend the results in [11]. One of the main results in those paper is the maximum satisfaction probability (MSP) function for a timing predicate c. Given the input parameters I_1 and t where I_1 is a interval timestamp and t is a time value, $MSP(I_1, t) \mid_c$ returns the maximum satisfaction probability of $c : @e_1 + d \geq @e_2$ provided that $@e_1 = I_1$ and $max(@e_2) = t$. Also, it is shown that $MSP(I_1, t) \mid_c$ is a monotonic non-increasing function w.r.t. a time value t for any timestamp I_1 and timing predicate c. By rearranging the MSP, given a confidence threshold value ct, we can compute the maximum x such that there exist a timestamp I_2, $max(I_2) = x$, and $prob(@e_1 + d \geq I_2) \geq ct$. Thus, it can be identified that $R_v = (x, \infty)$ from the above.

One limitation of this result is that it provides the information about "future" tuples only; it is always the case that $max(@e_1) \leq x$. Another limitation is that R_s is not identified. This is because ρ is not assumed in those previous work.

Now we extend our prior analysis to address those limitations as shown in Fig. 1. We assume that the system has a timing correlation condition (tp, ct) where the timing predicate $tp = |@A - @B| \leq d$ and the confidence threshold ct for stream A and B. In the figure, we assume that a base tuple with the timestamp $I_1 = (min_1, max_1)$ has arrived from stream A. It presents the maximum and minimum achievable satisfaction probabilities (in the Y-axis) for each possible $max_2 = max(I_2)$ where I_2 is the timestamp of a tuple in the target stream. The formulae used in the figure are shown in below:

$$\alpha\beta'_{min} = \frac{(min_1 - (d + max_2))^2}{2\pi len(I_1)},$$

$$\alpha\gamma'_{min} = \frac{2d + 2max_2 - min_1 - max_1}{2\pi},$$

$$\beta\gamma'_{min} = 1 - \frac{(d - max_1 + max_2 - \pi)^2}{2len(I_1)\pi},$$

$$\alpha\beta'_{max} = \frac{(min_1 - (d + max_2))^2}{2\rho len(I_1)},$$

$$\beta\beta'_{max} = \frac{2d + 2max_2 - \rho - 2min_1}{2len(I_1)},$$

$$\beta\gamma'_{max} = 1 - \frac{(d - max_1 + max_2 - \rho)^2}{2len(I_1)\rho},$$

$$\beta\gamma_{max} = 1 - \frac{(d - max_2 + min_1)^2}{2\pi len(I_1)},$$

$$\beta\beta_{max} = \frac{2d + max_1 + min_1 - 2max_2 + 2\pi}{2\pi},$$

$$\alpha\beta_{max} = \frac{(d + max_1 - max_2 + \pi)^2}{2len(I_1)\pi},$$

$$\beta\gamma_{min} = 1 - \frac{(d - max_2 + min_1)^2}{2\rho len(I_1)},$$

$$\alpha\gamma_{min} = \frac{2d + 2max_1 - 2max_2 + \rho}{2len(I_1)},$$

$$\alpha\beta_{min} = \frac{(max_2 - \rho - (d + max_1))^2}{2len(I_1)\rho}.$$

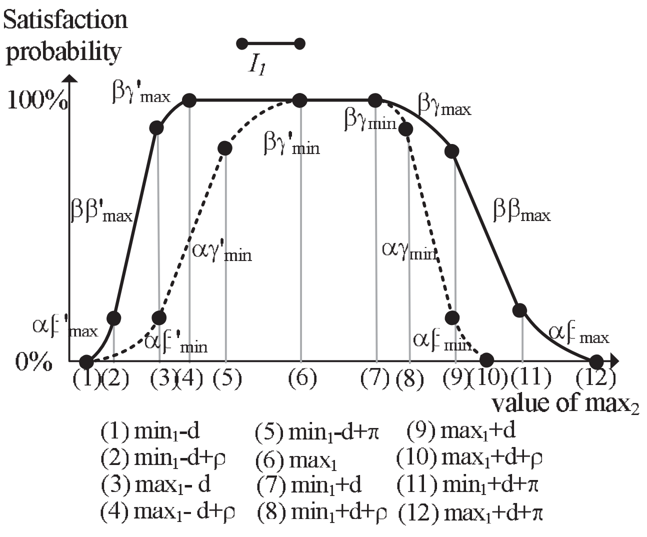

Fig. 1 The maximum and minimum achievable satisfaction probabilities of $|@e_1 - @e_2| \leq d$ where $@e_1 = I_1$ and $max(@e_2) = max_2$.

Figure 2 identifies the following three regions; $R_s = \{[L_L, R_L]\}$ - satisfaction region, $R_v = \{(-\infty, L_H), (R_H, \infty)\}$ - violation region, and $R_p = \{[L_H, L_L), (R_L, R_H]\}$ - probe region. It is easy to see that any target tuple with a timestamp belonging to the region R_s is guaranteed to satisfy the given

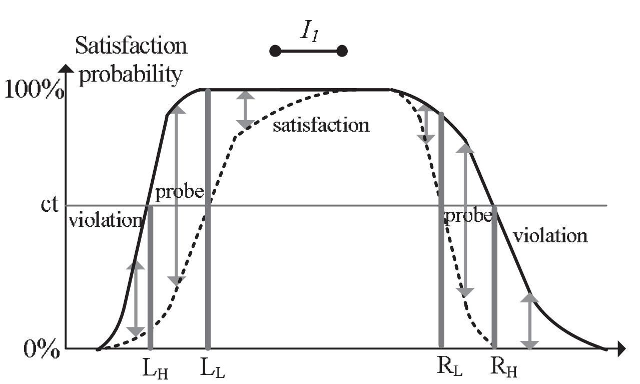

Fig. 2 Finding regions of violation, probe, and satisfaction.

join condition because its minimum satisfaction probability is above the requested confidence threshold. Similarly, any target tuple with a timestamp belonging to R_v is guaranteed to violate the given join condition. However, the target tuple with a timestamp belonging to the R_p needs further computations because its satisfaction probability can be either above or below the confidence threshold. In the following we show how to derive the formulae used in Fig. 1.

Lemma 1: Suppose a deadline predicate, $c : I_1 + d \geq I_2$ where $d \geq 0$. Let $I_1^{le} = (min(I_1) - \delta, max(I_1))$ where $\delta > 0$, i.e. an interval created by decreasing $min(I_1)$ by δ. Let c^{le} be the predicate $I_1^{le} + d \geq I_2$. Similarly, assume the followings hold. $I_1^{ls} = (min(I_1)+\delta, max(I_1))$, $I_1^{re} = (min(I_1), max(I_1)+\delta)$, $I_1^{rs} = (min(I_1), max(I_1)-\delta)$, $c_1^{ls} = I_1^{ls}+d \geq I_2$, $c_1^{re} = I_1^{re}+d \geq I_2$, $c_1^{rs} = I_1^{rs} + d \geq I_2$. Then, $prob(c_1^{le}) \leq prob(c)$, $prob(c_1^{ls}) \geq prob(c)$, $prob(c_1^{re}) \geq prob(c)$, and $prob(c^{rs}) \leq prob(c)$ hold.
Similarly, assume the following timestamps and timing predicates: $I_2^{le} = (min(I_2) - \delta, max(I_2))$, $I_2^{ls} = (min(I_2) + \delta, max(I_2))$, $I_2^{re} = (min(I_2), max(I_2) + \delta)$, $I_2^{rs} = (min(I_2), max(I_2) - \delta)$, $c_2^{le} = I_1 + d \geq I_2^{le}$, $c_2^{ls} = I_1 + d \geq I_2^{ls}$, $c_2^{re} = I_1 + d \geq I_2^{re}$, $c_2^{rs} = I_1 + d \geq I_2^{rs}$. Then, $prob(c_2^{le}) \geq prob(c)$, $prob(c_2^{ls}) \leq prob(c)$, $prob(c_2^{re}) \leq prob(c)$, and $prob(c_2^{rs}) \geq prob(c)$ hold.
Proof:
The proof is done in [11] by showing $prob(c) - prob(c^{le}) \geq 0$. *The other cases can be proved similarly.* □

Given a timing predicate, the following two corollaries identify the upper bound and lower bound of the value of $min(I_1)$ to achieve the maximum or minimum satisfaction probability of a deadline predicate $I_1 + d \geq I_2$. The following two corollaries can be derived from Lemma 1 and the definitions of π and ρ. They identify the lower bound and upper bound of the satisfaction probability of a timing predicate.

Corollary 2: Assume timing predicates $c_h : I_1 + d \geq I_h$ where $I_h = (t_{max} - \pi, t_{max})$ and $c_l : I_1 + d \geq I_l$ where $I_l = (t_{max} - \rho, t_{max})$. For any timing timing predicate $c : I_1 + d \geq I_2$ where $I_2 = (t_{min}, t_{max})$ and t_{min}, t_{max} are time values, it is always the case that $prob(c_l) \leq prob(c) \leq prob(c_h)$.

Corollary 3: Assume timing predicates $c_h : I_h + d \geq I_2$ where $I_h = (t_{max} - \rho, t_{max})$ and $c_l : I_l + d \geq I_2$ where $I_l = (t_{max} - \pi, t_{max})$. For any timing timing predicate $c : I_1 + d \geq I_2$ where $I_1 = (t_{min}, t_{max})$ and t_{min}, t_{max} are time values, it is always the case that $prob(c_l) \leq prob(c) \leq prob(c_h)$.

Now assume a tuple with a timestamp $I_1 = (min_1, max_1)$ arrived from stream A. Our goal is to determine ranges of max_2 which can tell whether the given timing predicate $(|I_1 - I_2| \geq d, p)$ is satisfied or violated where $I_2 = (min_2, max_2)$. Let the timing predicates $|I_1 - I_2| \geq d$, $I_1 + d \geq I_2$, $I_2 + d \geq I_1$ be c, c_1, and c_2 respectively.

We first identify two extreme cases where $max_2 < min_1 - d$ and $max_2 > max_1 + d + \pi$. In either case, clearly $prob(c)$ is zero. Let us derive some common properties which will be used in the derivation. By Corollary 2, $prob(c_2)$ reaches its upper bound and lower bound when $min_2 = max_2 - \rho$ and $min_2 = max_2 - \pi$ respectively. Similarly, $prob(c_1)$ reaches its upper bound and lower bound when $min_2 = max_2 - \pi$ and $min_2 = max_2 - \rho$ respectively. In case I_2 belongs to the range [(1),(5)], $min_2 \leq min_1$ holds because $min_2 \leq max_2 \leq min_1 - d + \pi \leq min_1$. $prob(c) = prob(c_2)$ in this range by Corollary 1. In case I_2 belongs to the range [(7),(12)], $min_1 \leq min_2$ holds because $min_1 + d \leq max_2 \leq min_2 + d$. $prob(c) = prob(c_1)$ in this range by Corollary 1.

In what follows, we shall first focus on the the maximum curve (drawn with a solid line) in the figure.
(1)-(2): $min_1 - d \leq max_2 \leq min_1 - d + \rho$. $prob(c) = prob(c_2)$ in this range. A timing predicate c_2 is in $\alpha\beta$ configuration and we get the upper bound of $prob(c_2)$ when $min_2 = max_2 - \rho$. Therefore, $\alpha\beta'_{min}$ represents the maximum satisfaction probabilities in this range.
(2)-(3): $min_1 - d + \rho \leq max_2 \leq max_1 - d$. The timing predicate c_2 is in $\beta\beta$ configuration when $min2 = max_2 - \rho$. Therefore, $\beta\beta'_{min}$ represents the maximum satisfaction probabilities in this range.
(3)-(4): $max_1 - d \leq max_2 \leq max_1 - d + \rho$. c_2 is in $\beta\gamma$ configuration when $min2 = max_2 - \rho$. $\beta\gamma'_{max}$ represents the maximum satisfaction probabilities in this range.
(4)-(7): $max_1 - d + \rho \leq max_2 \leq min_1 + d$. In this range if we set $min_2 = max_2 - \rho$, then $min_2 + d = max_2 - \rho + d \geq max_1$; it means that $prob(c_2) = 1$. In case $max_1 - d + \rho \leq max_2 \leq min_1 + \rho$, $min_2 \leq max_2 - \rho \leq min_1$ holds. Therefore, $prob(c) = prob(c_2) = 1$. In case $min_1 + \rho \leq max_2 \leq min_1 + d$, $min_1 + d \geq max_2$ holds, which means that $prob(c_1) = 1$. Since $prob(c_2) = 1$, $prob(c) = 1$. Therefore, the maximum of $prob(c)$ is one in this range.
(7)-(12): $min_1 + d \leq max_2 \leq max_1 + d + \pi$ In this range $prob(c) = prob(c_1)$ as explained in the beginning of this proof. The derivation of maximum satisfaction probabilities in this range for $prob(c_1)$ is in [11], hence is omitted.

Similarly, we can derive the formulae identifying the minimum satisfaction probabilities.

5. Experimental Results

In this section, we measure the effectiveness of the use of the minimum-maximum satisfaction probability for interval timing correlation.

We implement a stream simulation system as shown in Fig. 3. This system is a common test bed on which various implementations of the interval timing correlation are running. Including this simulation system, all implementations are coded in Java. The experiments were done on a Xeon 1.8 Mhz based PC with 1 GB main memory running Sun JDK 1.3.2 on Windows XP professional. To measure the performances of various algorithms for the interval timing correlation on the same conditions, we create data streams on data files before run-time and feed the same data files to the interval timing correlation operator. In the figure, each stream provider thread reads a data file, generates data stream, and insert the data to the corresponding stream

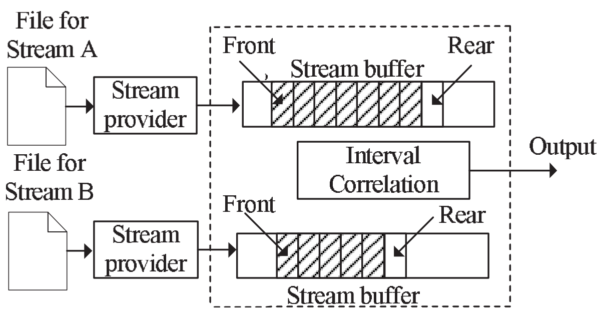

Fig. 3 Experiment set-up.

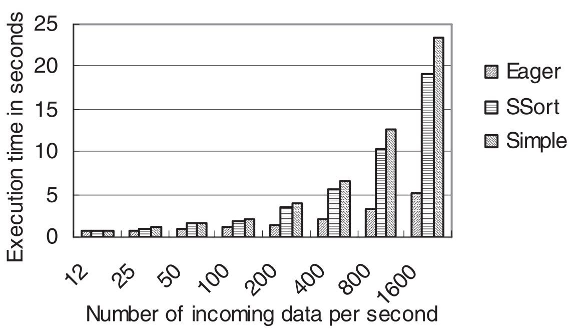

Fig. 4 The total execution times under various arrival rates.

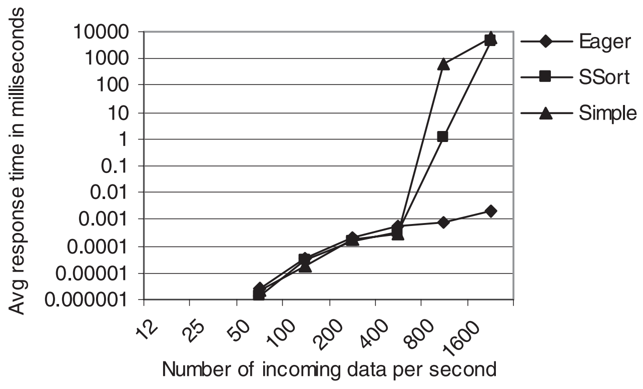

Fig. 5 The average response times under various arrival rates.

buffer. The stream buffer is implemented in a circular array. *Front* points the location of the first tuple[†] in the buffer and *Rear* points the position where a new tuple is going to be placed. A stream buffer grows by advancing its *Rear* to the clockwise direction. Similarly, a stream buffer shrinks by advancing its *Front* clockwise.

In the following, we use *base stream buffer* to denote the buffer for a stream from which a new tuple (or a block of tuples) arrived. The buffer for the other stream is referred to as *target stream buffer*.

The *simple correlation* is the most simplest one among the interval timing correlations discussed here. When a tuple e arrives at the base stream, every tuple in the target stream buffer is examined and the satisfaction probability is calculated. While the system is visiting the tuples in the target stream buffer, obsolete tuples are marked. Then the marked tuples are removed from the target stream buffer and e is inserted into the base stream buffer.

It is too expensive to remove obsolete tuples from stream buffers and shift the remainder physically. The implementation for removing obsolete tuples in the simple correlation is done by finding consecutively located obsolete tuples from the front of a stream buffer and advancing the pointer *Front* beyond the end of such tuples. The *simple sort correlation* expects longer consecutively located obsolete tuples than the simple correlation does by keeping tuples in stream buffers sorted in their *max* timestamps order.

The *eager correlation* uses the analysis result of satisfaction probability presented in the previous section. Every time a tuple e arrives, the system computes L_H, L_L, R_L and R_H based on e. We shall use $L_H(e)$ to denote the value of L_H computed based on e[††]. $L_L(e)$, $R_L(e)$, and $R_H(e)$ can be interpreted similarly. As illustrated in the previous section, all tuples belonging to $[L_L(e), R_L(e)]$ in the target stream buffer are guaranteed to be in the result. The tuples belonging to $[L_H(e), L_L(e))$ or $(R_L(e), R_H(e)]$ in the target stream buffer should be probed further.

To determine the block of invalid tuples, we first set $@e_{inv}$ to $(CurrentTime - delay_{base} - \pi, CurrentTime - delay_{base})$ and compute L_H based on e_{inv}. Since the tuples belonging to $(-\infty, L_H(e_{inv}))$ in the target buffer are guaranteed not to match with any incoming future tuples from the base stream, they can be removed from the buffer. For the efficient searching in stream buffers, we use binary search hence stream buffers should be sorted; after completing a correlation, a new tuple is inserted into the base stream buffer in a sorted order.

To measure the performance of each algorithm under different workloads, we use the data files r12.dat, r24.data, ..., r1600.dat of which arrival rates are from 12 tuples/second to 1600 tuples/second. We measured the total execution times (Fig. 4) and the average response times (Fig. 5). In this experiment, we set $\pi = 300$, $\rho = 20$. The response time of a tuple (e_a, e_b) is computed by the correlation completion time minus MAX(arrival time of e_a, arrival time of e_b).

From the figures, it is evident that the eager correlation outperforms the simple correlation and the simple sort correlation (SSORT in the figure). The main reason is that the eager correlation full utilizes the analysis result of satisfaction probability. Hence it saves significant execution time by avoiding the computation of satisfaction probabilities for the tuples belonging to the satisfaction and violation regions.

We can also notice that the simple sort correlation is faster than the simple correlation. This is mainly because the invalidation process is much effective in the simple sort correlation.

Figure 6 shows the performance under varying confi-

[†]Roughly speaking it is the oldest tuple, i.e., the one with the smallest *max* timestamp, if the buffer is sorted.

[††]The parameters $ct, distance, \rho$, and π are also used in the computation, however we shall omit these parameters in expressions for the brevity.

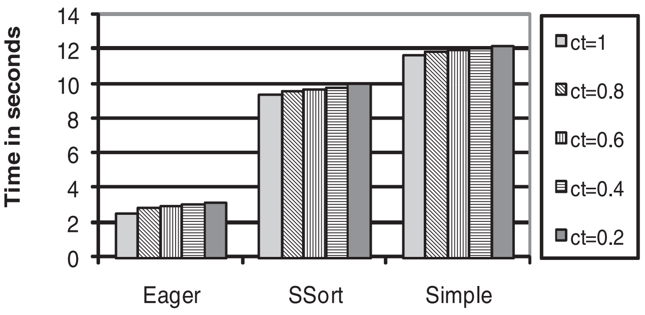

Fig. 6 The execution times under various confidence thresholds.

dence thresholds (*ct*). We deliberately composed the data file such that all tuples are ordered in the max values of their timestamps. In this experiment, we varied *ct* from 1.0 to 0.2. The execution time increases as *ct* decreases. This can be explained by the fact that the candidate range for timing correlation becomes larger as *ct* decreases. Also, the probe region is increased proportionally.

6. Conclusions

We introduced interval timing correlation, a variant of the join operator for streaming data. The interval timing correlation enables users to assess timing conditions over streaming data with interval timestamps. The interval timestamps are adopted to express temporal uncertainties which may exist in the timestamps of stream data. The proposed timing condition consists of a timing predicate and a confidence threshold. The timing predicate specifies a timing correlation between two timestamps in forms of “deadline”, “delay”, and “within”. The confidence threshold states the minimum satisfaction probability for the associated timing predicate. If the satisfaction probability of the timing predicate is lower than the confidence threshold, then the timing condition is evaluated to be false.

Based on the results in previous work [11], this paper extends the analysis of satisfaction probability for the within predicate. In designing algorithms for the interval timing correlation, we utilized this information to effectively identify only data to evaluate. The performances of the proposed algorithm under different workloads were measured and analyzed.

For the future work, applying the techniques developed in this paper to practical applications such as intrusion detection would be interesting. For example, there have been efforts for utilizing temporal correlations between the events related to the security for detecting intrusion detection [13], [14].

Acknowledgements

This research was supported by a grant (CR070019) from Seoul R&BD Program funded by the Seoul Development Institute of Korean government.

References

[1] L. Golab and M.T. Ozsu, “Processing sliding window multi-joins in continuous queries over data streams,” Proc. International Conference on Very Large Data Bases (VLDB), pp.500–511, 2003.
[2] C.E. Dyreson and R.T. Snodgrass, “Supporting valid-time indeterminacy,” ACM Trans. Database Syst., vol.23, no.1, pp.1–57, March 1998.
[3] R. Cheng, D.V. Kalashnikov, and S. Prabhakar, “Evaluating probabilistic queries over imprecise data,” Proc. ACM SIGMOD International Conference on Management of Data, pp.551–562, 2003.
[4] B. Babcock, S. Babu, M. Datar, R. Motwani, and J. Widom, “Models and issues in data stream systems,” Proc. ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, pp.1–16, 2002.
[5] L. Golab and M.T. Ozsu, “Issues in data stream management,” SIGMOD Rec., vol.32, no.2, pp.5–14, 2003.
[6] D.J. Abadi, D. Carney, U. Cetintemel, M. Cherniack, C. Convey, S. Lee, M. Stonebraker, N. Tatbul, and S.B. Zdonik, “Aurora: A new model and architecture for data stream management,” VLDB Journal, vol.12, no.2, pp.120–139, 2003.
[7] M.A. Hammad, W.G. Aref, and A.K. Elmagarmid, “Stream window join: Tracking moving objects in sensor-network databases,” Proc. International Conference on Scientific and Statistical Database Management, pp.75–84, 2003.
[8] J. Kang, J.F. Naughton, and S. Viglas, “Evaluating window joins over unbounded streams,” Proc. International Conference on Data Engineering, pp.341–352, 2003.
[9] U. Srivastava and J. Widom, “Flexible time management in data stream systems,” Proc. ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, pp.263–274, 2004.
[10] A.K. Mok, C.G. Lee, H. Woo, and P. Konana, “The monitoring of timing constraints on time intervals,” Proc. IEEE Real-Time Systems Symposium (RTSS), pp.191–200, 2002.
[11] C.G. Lee, A.K. Mok, and P. Konana, “Monitoring of timing constraints with confidence threshold requirements,” IEEE Trans. Comput., vol.56, no.7, pp.977–991, 2007.
[12] A.K. Mok, H. Woo, and C.G. Lee, “Probabilistic timing join over uncertain event streams,” Proc. IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA), 2006.
[13] A. Jones and S. Li, “Temporal signatures for intrusion detection,” Proc. Annual Computer Security Applications Conference, pp.252–261, 2001.
[14] V.C.S. Lee, J.A. Stankovic, and S.H. Son, “Intrusion detection in real-time database systems via time signatures,” Proc. Real-Time Technology and Applications Symposium (RTAS), pp.124–133, 2000.

Chan-gun Lee is Assistant Professor in the Dept. of Computer Science and Engineering at Chung-Ang University, Korea. He received the BS degree in 1996 from Chung-Ang University, Korea, the MS degree in 1998 from Korea Advanced Institute of Science and Technology (KAIST), Korea, and the PhD degree in 2005 from the University of Texas at Austin, all in computer science. He was a recipient of a scholarship from the Korea Foundation for Advanced Studies during his Ph.D. study. He worked as a senior software engineer at Intel, Hillsboro for two years after getting his PhD degree. His research interests include real-time systems, software engineering for time-critical systems, and streaming data processing.

Aloysius K. Mok is Quincy Lee Centennial Professor in Computer Science at the University of Texas at Austin. He received the S.B. in electrical engineering, the S.M. in electrical engineering and computer science and the Ph.D. degrees in computer science, all from the Massachusetts Institute of Technology. Since 1983, Dr. Mok has been on the faculty of the Department of Computer Sciences at the University of Texas at Austin. Professor Mok has done extensive research on computer software systems and is internationally known for his work in real-time systems. He is a past Chairman of the Technical Committee on Real-Time Systems of the Institute of Electrical and Electronics Engineers, and has served on numerous national and international research and advisory panels. His current interests include real-time and embedded systems, robust and secure network-centric computing and real-time knowledge-based systems. Dr. Mok received in 2002 the IEEE TC on Real-Time Systems Award for his outstanding technical contributions and leadership achievements in real-time systems.

Prabhudev Konana is Associate Professor of MIS and Distinguished Teaching Professor at the McCombs School of Business, the University of Texas at Austin. He has an MBA and a Ph.D. from the University of Arizona. His research interests are in electronic business value, electronic brokerages, online investor satisfaction, outsourcing and offshoring of IT and business processes, and online supply chain management. His research is supported by grants from NSF CAREER Award, NSF Information Technology Research, Dell, Intel and IBM. He has received numerous teaching awards. He has published over 60 papers in journals and conference proceedings including Management Science, Sloan Management Review, Management Information Systems Quarterly, Information Systems Research, Communications of the ACM, INFORMS Journal on Computing, Information Systems. He serves in the program committee of numerous conferences in the management of information technology and systems area.