Probabilistic Timing Join over Uncertain Event Streams *

Aloysius K. Mok, Honguk Woo
Department of Computer Sciences,
The University of Texas at Austin
{mok,honguk}@cs.utexas.edu

Chan-Gun Lee
Intel Corporation,
chan-gun.lee@intel.com

Abstract

This paper addresses the problem of processing event-timing queries over event streams where the uncertainty in the values of the timestamps is characterizable by histograms*. We describe a stream-partitioning technique for checking the satisfaction of a probabilistic timing constraint upon event arrivals in a systematic way in order to delimit the "probing range" in event streams. This technique can be formalized as a probabilistic timing join* (PTJoin) *operator where the join condition is specified by a time window and a confidence threshold in our model. We present efficient* PTJoin *algorithms that tightly delimit the probing range and efficiently invalidate events in event streams.*

1 Introduction

The monitoring of data streams requires a new class of data management systems [2] that has a far different scope than conventional active databases or reactive programs that monitor and react to external events in a control loop. For example, suppose the images of a monitor camera in a hazardous environment are sent back every 5 minutes for processing, and the difference between two consecutive images reveals that a dangerous event e has happened within the time between the two images. In such a scenario, we do not know for certain when exactly e occurs; so if we are to tag the occurrence of e with a timestamp, the timestamp should have as its domain a time interval and not a time point. Yet the provision of precise timing information of events is often crucial for supporting a unified view of the monitored environment. In this paper, we address the problem of performing query processing of the timing relationship between events whose occurrence time can be determined accurate to at most within an interval.

The following example illustrates an anomaly that arises when events are correlated by using as input the time points at which the events are detected from processing the data streams. This mode of query processing is said to be by *detection semantics*. If the exact times of occurrence of the events are known and are used as input, then we say that the mode of query processing is by *occurrence semantics*. Unqualified use of the detection semantics can easily yield a wrong answer where occurrence semantics is intended.

Example 1 *Consider a continuous query* cq *that joins two input events whenever they occur within* 40*ms. Figure 1 shows that events* e_i *and* e_j *are detected and stamped by sensors* s_i *and* s_j *at time* 60 *and* 110 *respectively. Since* cq *is defined as a timing relationship between event occurrences, the execution of* cq *relying on the detection timestamps may end up missing a join result of* e_i *and* e_j*. This may happen if for example, the sensors* s_i *and* s_j *operate adaptively in low-power sleeping mode, thereby inducing variable detection delays equal to 20ms and 40ms respectively. The event pair* e_i *and* e_j *actually satisfy the join condition of* cq*. Unfortunately, the detection delay may not be known exactly.*

It has been shown that probabilistic data models can be effectively adopted for dealing with time-varying data attributes [3, 4]. However, these models do not address the issue of the temporal uncertainty in the occurrence times of the data update events themselves. There are two key factors that need to be considered regarding temporal monitoring over event streams. First, *time* is monotonically increasing along the data stream. Second, the correlation of the temporal relationship among event occurrences often needs to be performed in real time because the result of the correlation calculation may determine whether certain

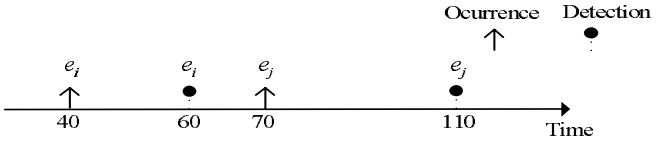

Figure 1. Event occurrences and detections

*This research is partially supported by ONR grant N00014-03-1-0705.

Proceedings of the 12th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA'06)
0-7695-2676-4/06 $20.00 © 2006 IEEE

timing constraint violation/satisfaction has occurred, which may in turn decide whether data attributes and occurrence times further down the data stream are valid or not. We shall show that the calculation of timing correlation can be cast into the problem of performing a join operation that creates an output stream, where each output tuple must satisfy the probability that the events in the join specification occur together within a specified time interval. We refer to this class of band joins over streams with temporal uncertainty as *PTJoin* (Probabilistic Timing Join). *PTJoin* can be made efficient by exploiting the key factors of monotonicity and instantaneity as mentioned above.

Example 2 *Consider a set of sensor nodes that are spread densely in a surveillance region where the detection latency of a suspicious activity by an intruder in the region is probabilistically known according to specific sensor settings (i.e., sensing modalities, power level, hardware mechanisms) and environmental conditions (i.e., variations on terrain and weather) via experiments [1]. To make a more accurate hypothesis on possible intrusions, events from different modalities and nearby sensor nodes can be joined based on their temporal proximity, that is, "if the probability that acoustic and seismic sensor events occur within, say,* 2000*ms exceeds the confidence threshold* 0.6*, then raise alarm for possible intrusion.". To adjust the false alarm rate, this threshold must be updated efficiently.*

The above example specifies a join calculation that is predicated on temporal proximity. A confidence threshold is embedded in a join condition to denote the uncertainty of the temporal correlation. We note that the special case of a uniform distribution over the interval (named *interval timing join*) has been studied in [7]. In this paper, we incorporate the uncertainty in event timing by using a histogram timestamp model. A timestamp is created by partitioning a time interval I into a finite number of subintervals and specifying the probability that the event (instance) occurs during each subinterval, such that the probability of the event occurring in I is 1.0. With this model, we shall show how to systematically partition a temporally ordered event stream upon the arrival of a event, by computing the minimum/maximum satisfaction times of a timing constraint so that we can optimize *PTJoin* the calculation with the necessary "probing range".

The contributions of this paper are as follows. First, we generalize the temporal monitoring problem under event timing uncertainty by allowing arbitrary probability distributions of event occurrence times to be specified by histogram timestamps (Section 2). Second, using the histogram timestamp model, we describe a partitioning scheme which can be used for efficiently checking a timing constraint against event streams. Based on the partitioning scheme, we develop a *PTJoin* operator that can handle event timing uncertainty in processing the band join over event streams. More specifically, we show how to tightly delimit the probing range and efficiently invalidate events in event streams (Section 3).

2 Probabilistic Event Timing Data

In this section, we present our histogram timestamp model and a method for evaluating a probabilistic temporal relationship.

2.1 Histogram Timestamp

First, we introduce some notations and functions in our histogram timestamp model. We use the symbol I to denote a time interval $I = [t_1, t_2]$ where t_1 and t_2 are time points such that $t_1 < t_2$. We assume that the domain of time points is the set of non-negative real numbers. The following auxiliary functions are defined for time intervals: $min(I) = t_1$, $max(I) = t_2$, $len(I) = t_2 - t_1$, $x \in I$ iff $min(I) \leq x \leq max(I)$, and $I + x = [min(I) + x, max(I) + x]$ for a real number x.

Definition 1 *A histogram timestamp of an event is an n-tuple, $H = ((I_1, p_1), (I_2, p_2), \ldots, (I_n, p_n))$ where (I_k, p_k) is called the kth bucket consisting of the time interval I_k and the probability p_k that the event occurs during I_k. A histogram timestamp has the following properties:*
- $\sum_{k=1}^{n} p_k = 1$,
- $\forall 1 \leq k < n, max(I_k) = min(I_{k+1})$,
- $\forall 1 \leq k \leq n$, *the occurrence time is uniformly distributed over each I_k. Formally, the probability density function (*pdf*) is described by $f(x) = \frac{p_k}{len(I_k)}$ if $x \in I_k$.*

Similar to those defined for intervals, the following auxiliary functions are defined: $|H|$ = the number of buckets, $min(H) = min(I_1)$, $max(H) = max(I_{|H|})$, $len(H) = max(H) - min(H)$, $H + x = ((I_k + x, p_k))_{1 \leq k \leq |H|}$ for a real number x. Unless stated otherwise, given an event stream A consisting of events $a_1, a_2, a_3, \ldots$, we denote the occurrence time of an event a_i by H_{a_i} and the random variable (r.v) on H_{a_i} by X_{a_i}[1]. The time interval and the probability of the kth bucket in H_i are denoted by I_{ik} and p_{ik} respectively, and thus we may write $p_{ik} = P(X_i \in I_{ik})$ for $k = 1, \ldots, |H_i|$. The following example illustrates how we can capture the histogram timestamp of an event by using the template histogram that models the detection latency of the event. For clarity, we shall use the symbol ϕ to denote a template histogram that represents a *pdf* when the the associated event is not named. Without loss of generality, we assume that $min(\phi_i) = 0$ for all template histograms ϕ_i.

[1] If there is no ambiguity, we drop the letter for an event name in the subscript of symbols H and X, e.g., for an event a_i, we may use H_i and X_i instead of H_{a_i} and X_{a_i}.

Proceedings of the 12th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA'06)
0-7695-2676-4/06 $20.00 © 2006 IEEE

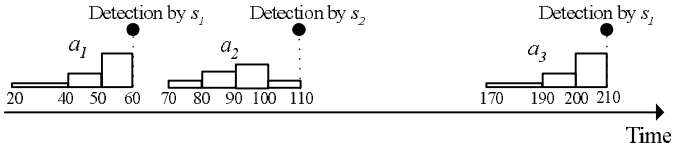

Figure 2. An event stream with histogram timestamps

Example 3 *Consider a stream A consisting of events from multiple sensor nodes $s_1, s_2, \ldots, s_n$ where a sensor node s_i is observed to empirically satisfy a probability distribution represented by the template histogram $\phi_i = ((I_{i1}, p_{i1}), \ldots, (I_{im}, p_{im}))$ with respect to the detection latency of its sensing modality for $i = 1, \ldots, n$. For example, suppose the detection latency of s_1 satisfies the distribution $((0, 10, 0.6), (10, 20, 0.3), (20, 40, 0.1))$[2]; so the template histogram of s_1 is defined as $\phi_1 = ((0, 20, 0.1), (20, 30, 0.3), (30, 40, 0.6))$. In this case, for an event a_j that is detected at time t_{det} by s_1, the data stream processor can probabilistically approximate the occurrence time of a_j by $H_j = \phi_1 + t_{det} - max(\phi_1)$. Figure 2 depicts three timestamps captured from two sensor nodes, i.e., s_1 detects two events a_1 and a_3 at time 60 and 210, and s_2 detects an event a_2 at time 110 where $\phi_2 = ((0, 10, 0.15), (10, 20, 0.3), (20, 30, 0.4), (30, 40, 0.15))$.*

2.2 Temporal Relationship

In general, we shall assume that the histogram distributions associated with different event occurrences are independent of one another as in [5].

Definition 2 *Given two histogram timestamps H_i and H_j, $ge(H_i, H_j)$ is defined as the probability:*

$$P(X_i \geq X_j) = \int_{H_i} F_j(t) f_i(t) dt \quad (1)$$

where $F_j(t) = P(X_j \leq t)$ and $f_i(t) = P(X_i = t)$.

Even though Equation 1 is conceptually simple, it is nontrivial to compute in practice because both $f_i(t)$ and $F_j(t)$ are piecewise functions that depend on the buckets of H_i and H_j. For computational purposes, we shall use the following alternative equation that makes use of joint probabilities:

$$ge(H_i, H_j) = \sum_{u=1}^{|H_i|} \sum_{v=1}^{|H_j|} P(X_i \geq X_j, X_i \in I_{iu}, X_j \in I_{jv}) \quad (2)$$

[2]Note that a bucket may be written in a triple without the inner brackets enclosing its time interval.

Lemma 1 *Given the buckets (I_{iu}, p_{iu}) from H_i and (I_{jv}, p_{jv}) from H_j, let i_1, j_1, i_2, and j_2 denote $min(I_{iu})$, $min(I_{jv})$, $max(I_{iu})$, and $max(I_{jv})$ respectively. The joint probability $P(X_i \geq X_j, X_i \in I_{iu}, X_i \in I_{jv})$ is given as follows:*

$$\begin{cases} 0 & :(1)\ \textit{if } i_2 \leq j_1, \\ p_{iu}p_{jv} & :(2)\ \textit{if } i_1 \geq j_2, \\ \frac{p_{iu}p_{jv}(i_2-j_1)^2}{2(i_2-i_1)(j_2-j_1)} & :(3)\ \textit{if } i_2 > j_1 \wedge i_2 \leq j_2 \wedge i_1 < j_1, \\ \frac{p_{iu}p_{jv}(2i_2-j_1-j_2)}{2(i_2-i_1)} & :(4)\ \textit{if } i_2 \geq j_2 \wedge i_1 < j_1, \\ p_{iu}p_{jv} - \frac{p_{iu}p_{jv}(j_2-i_1)^2}{2(i_2-i_1)(j_2-j_1)} & :(5)\ \textit{if } i_2 > j_2 \wedge i_1 \geq j_1 \wedge i_1 < j_2, \\ \frac{p_{iu}p_{jv}(i_2-2j_1+i_1)}{2(j_2-j_1)} & :(6)\ \textit{otherwise.} \end{cases}$$

Proof: *Since the proof for non-overlapping intervals is straightforward, let us show the proof for the cases when the time intervals I_{iu} and I_{jv} overlap. We can divide the timeline of I_{ai} into three subintervals: I_1 where I_{iu} definitely precedes I_{jv}, I_2 where I_{iu} and I_{jv} intersect, and I_3 where I_{iu} definitely follows I_{jv}. Let us consider the case (3). In this case, we find I_1 and I_2 but not I_3. Since there is no way to satisfy $X_j < X_i$ in I_1, we only need to consider the probability in $I_2 = [j_1, i_2]$ such that $P(X_i \geq X_j | X_i \in I_2, X_j \in I_2)$ $P(X_i \in I_2)$ $P(X_j \in I_2)$ which is equivalent to $\frac{P(X_i \in I_2)P(X_j \in I_2)}{2}$. This is because we have $P(X_i \geq X_j | X_i \in I_2, X_j \in I_2) = \frac{1}{len(I_2)^2} \int_{min(I_2)}^{max(I_2)} t - min(I_2) dt = \frac{(max(I_2)-min(I_2))^2}{2 \cdot len(I_2)^2} = \frac{1}{2}$. Owing to the uniform density on each time interval, the probability for subintervals can be easily calculated by $P(X_i \in I_2) = \frac{p_{iu}(i_2-j_1)}{len(I_{iu})}$ and $P(X_j \in I_2) = \frac{p_{jv}(i_2-j_1)}{len(I_{jv})}$. The other cases can be similarly established. In particular, we can derive the fomular for the case (5) (or the case (6)) from that of the case (3) (or the case (4)) using the inverse relation, i.e., $P(X_i \geq X_j) = 1 - P(X_j \geq X_i)$.* □

Given a time interval I_{iu} of H_i, let I_{jv}= $<I_{jl}, I_{jl+1}, \ldots, I_{jr-1}, I_{jr}>$ be the list of time intervals of H_j that overlap with I_{iu}. As part of the task of computing $ge(H_i, H_j)$, the probability from each pair of I_{iu} and I_{jv} needs to be determined by the formulae in Lemma 1. As explained in the proof, the formulae are obtained by the process that divides an interval into subintervals and distributes the probability to the subintervals.

Definition 3 *Let T_{iu}= $<min(I_{i1}), min(I_{i2}), \ldots, min(I_{in}), max(I_{in})>$ be the list of the time points in H_i and let T_{jv}= $<min(I_{j1}), min(I_{j2}), \ldots, min(I_{jm}), max(I_{jm})>$ where $n = |H_i|$ and $m = |H_j|$. By merging T_{iu} and T_{jv} in ascending order without repetition of any time point that appears more than once, we can obtain the time points $t_1 < t_2 < \ldots < t_l < t_{l+1}$, and construct the time intervals $I_1 = [t_1, t_2], \ldots, I_l = [t_l, t_{l+1}]$. The histogram consisting of the buckets $(I_1, P(X_i \in I_1)), \ldots, (I_l, P(X_i \in$*

$I_l))$ *will be called the adjusted histogram of* H_i *and similarly the histogram consisting of the buckets* $(I_1, P(X_j \in I_1)),\ldots,(I_l, P(X_j \in I_l))$ *will be called the adjusted histogram of* H_j.

The idea of histogram adjustment is to perform a union on all time points and to distribute the probabilities to new time intervals accordingly so that the histograms share the identical time interval set while each histogram maintains its original probability density.

Theorem 1 *Given two histogram timestamps* H_i *and* H_j, *let the corresponding adjusted histograms be* $\tilde{H}_i = ((I_u, \tilde{p}_{iu}))_{1\leq u\leq l}$ *and* $\tilde{H}_j = ((I_v, \tilde{p}_{jv}))_{1\leq v\leq l}$ *respectively where* $|\tilde{H}_i| = |\tilde{H}_j| = l$. *Then, we have*

$$ge(H_i, H_j) = \sum_{u=1}^{l}\sum_{v=1}^{u-1}\tilde{p}_{iu}\tilde{p}_{jv} + \frac{1}{2}\sum_{u=1}^{l}\tilde{p}_{iu}\tilde{p}_{ju}. \quad (3)$$

3 *PTJoin*: Probabilistic Timing Join

In this section, we describe *PTJoin* in our histogram timestamp model. A SQL-style query representation for Example 2 takes the following form:

```
select * from SEISMIC A, ACOUSTIC B
where WINDOW(A, B) = 2000 with THRESHOLD 0.6
```

The join condition of *PTJoin* above is specified by two parameters, the time window $d = 2000$ and the confidence threshold $\delta = 0.6$. In general, we shall denote such a join condition by ***jc***: (d, δ); an event a_i in a stream A and an event b_j in a stream B qualify ***jc*** iff $P(X_i + d \geq X_j \wedge X_j + d \geq X_i) \geq \delta$ holds.

A straightforward method to perform *PTJoin* is as follows. Upon a new event arrival in a stream A (or B), first retrieve all events in a stream B (or A) and then keep evaluating the join condition with each of the possible event pairs. The evaluation of the join condition may require performing the task of histogram adjustment to check the temporal relationship between two events. Three problems arise with this approach. First, how can we tightly delimit the set of event pairs being probed upon an event arrival? Second, how can we efficiently evaluate the join condition for an event pair? Third, how can we effectively manage stream buffers? Based on the nested loop join model for sliding time windows [6], we investigate the stream partitioning technique that tightly identifies the probing range for an arriving event by parameterizing the uncertain temporal properties of streams and keeping events in the temporal order. It is possible to keep such a temporal order in a stream without incurring significant overheads owing to the fact that events normally arrive in ascending order of their occurrence times [9]. The partitioning allows us to prune unnecessary join tests and furthermore derive the tightest sliding time window over a stream, rendering the events outside the sliding time window obsolete.

Obviously, if network delays and the degree of timing uncertainty in streams cannot be bounded, it is impossible to determine the probing range for an arriving event to within a finite interval. Henceforth, we consider only streams where the temporal uncertainty of all the events can be parameterized (bounded). We shall introduce several parameters for capturing the temporal uncertainty of the events in a stream and call them *stream parameter*; a stream parameter denotes some quantitative property that applies everywhere in a stream.

Definition 4 *We use the stream parameter* $\mathcal{H}_A = \{\phi_1, \phi_2, \ldots, \phi_n\}$ *to denote the set of template histograms in a stream A, i.e., if an event* a_i *in A is detected at time t, then it must be the case* $\exists\phi_j \in \mathcal{H}_A$, $H_i = \phi_j + t - max(\phi_j)$. *We shall write* $a_i \models \phi_j$ *to denote that* ϕ_j *is the template histogram of the event* a_i.

Definition 5 *The stream parameter* π_A *denotes the maximum length of template histograms that can occur in a stream A, i.e.,* $\forall\phi_i \in \mathcal{H}_A$, $len(\phi_i) \leq \pi_A$ *holds.*

Definition 6 *The stream parameter* $\mathcal{D}_A$ *denotes the maximum transmission latency in a stream A to the stream processor, i.e., for any event* a_i *in A, the datum must arrive at the stream processor by time* $max(H_i) + \mathcal{D}_A$.

Definition 7 *Given a timing constraint* $\boldsymbol{c}$: $T_1 + d \geq T2$ *where* T_1, T_2 *are time terms corresponding to occurrence times of events and d is a delay or deadline constant, we define a probabilistic timing constraint (*PTC*) as* $prob(\boldsymbol{c}) \geq \delta$ *where* $0 < \delta < 1$ *is the confidence threshold. The term* $prob(\boldsymbol{c})$ *is known as the satisfaction probability of* $\boldsymbol{c}$ *[8].*

In the following, we restrict our discussion to the cases where (1) the stream parameters in Definition 4-6 are known *a priori* as part of system specification, say, via sensor calibrations and (2) the time window size d in *PTJoin* queries is relatively large such that $d \geq max(\pi_A, \pi_B)$ holds. In particular, with such d values, the evaluation of the join condition ***jc*** in *PTJoin* transforms to that of its associated *PTC*: $prob(H_i + d \geq H_j) \geq \delta$ or $prob(H_j + d \geq H_i) \geq \delta$ for a pair of events a_i and b_j.

Lemma 2 *Consider a join condition* ***jc***:$(d \geq max(\pi_A, \pi_B), \delta)$, *and a pair of events* a_i *in a stream A,* b_j *in a stream B such that* $max(H_j) \geq max(H_i)$. *The event pair satisfy the join condition* ***jc*** *iff* $prob(H_i + d \geq H_j) \geq \delta$. *Likewise, given a pair of events* a_i *and* b_j *such that* $max(H_i) > max(H_j)$, *it must be the case that they satisfy the join condition* ***jc*** *iff* $prob(H_j + d \geq H_i) \geq \delta$.

Proof: *Due to the given condition* $max(H_j) \geq max(H_i)$

Proceedings of the 12th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA'06)
0-7695-2676-4/06 $20.00 © 2006 IEEE

*and $d \geq \pi_A \geq len(H_i)$, we obtain $min(H_j) + d \geq max(H_i)$, which is equivalent to $P(X_j + d \geq X_i) = 1$. Thus, $P(H_i + d \geq H_j) \geq \delta$ ensures the satisfaction of **jc**; conversely, $P(H_i + d \geq H_j) < \delta$ ensures the violation of **jc**. We omit the proof for the other case.* □

Evaluating an event pair for *PTJoin* requires checking the *PTC* imposed on the event pair by the join condition. Notice that we can check a *PTC* such as $prob(H_i + d \geq H_j) \geq \delta$ by computing $ge(H'_i, H_j)$ where $H'_i = H_i + d$.

For convenience, we shall assume the availability of *max-time sorted streams*, for which stream buffers are set up to contain events in ascending order of their maximum possible occurrence times, that is, $max(H_i)$ for an event a_i.

3.1 Partitioning Max-time Sorted Streams

Here, we describe the partitioning technique to be applied to max-time sorted streams A and B for performing *PTJoin* with the join condition ***jc***: (d, δ). Upon arrival of event a_i in the stream A, we partition the stream B into

- *Satisfaction* range (**SR**) which contains all events b_j in B for which $max(H_j) \in$ **SR** satisfies ***jc*** with a_i,
- *Violation* range (**VR**) which contains all events b_j in B for which $max(H_j) \in$ **VR** violates ***jc*** with a_i,
- *Probing* range (**PR**) which contains all events b_j in B for which $max(H_j) \in$ **PR** may or may not satisfy ***jc*** with a_i.

With these partitions, we can limit probing events in B to only those in **PR**. Regardless of the value of δ, we can have the partitions constructed by two **PR**s:

$$\mathbf{LPR} = [min(H_i) - d, max(H_i) - d + \pi_B), \quad (4)$$

$$\mathbf{RPR} = (min(H_i) + d, max(H_i) + d + \pi_B] \quad (5)$$

Therefore, only events b_j in B such that $min(H_i) - d, \leq max(H_j) < max(H_i) - d + \pi_B \vee min(H_i) + d < max(H_j) \leq max(H_i) + d + \pi_B$ will be checked with a_i. Not surprisingly,

$$\mathbf{SR} = [max(H_i) - d + \pi_B, min(H_i) + d] \quad (6)$$

is surrounded by two **PR**s, and **VR** is necessarily confined by the ranges that are neither **PR**s nor **SR**.

Given a specific δ, we can find the tightest **PR**s for an event that triggers the probing over a stream of its join partners. Lemma 3 will show that the satisfaction probability of a timing constraint monotonically decreases (or increases) as one histogram timestamp H_j in the timing constraint slides forward (or backward) in time while the other histogram timestamp H_i in the timing constraint is fixed in time. Motivated by this observation, we construct a partitioning method based on the satisfaction time (in Definition 8 below) of template histogram pairs.

Lemma 3 *Suppose $0 < prob(H_i + d \geq H_j) = \delta < 1$. It must be the case that $prob(H_i + d + \theta \geq H_j) > \delta > prob(H_i + d - \theta \geq H_j)$ holds for any $\theta > 0$.*

Definition 8 *Consider the histogram timestamps H_i, H_j, and $0 < \delta < 1$. The satisfaction time, $stime(H_i, H_j, \delta)$ is a time point $\mathcal{T}$ such that $prob(H_i + max(H_j) - \mathcal{T} \geq H_j) = \delta$ holds.*

It is clear that for any real number x, we have

$$stime(H_i, H_j, \delta) + x = stime(H_i + x, H_j, \delta). \quad (7)$$

By Lemma 3, given any time point $t > stime(H_i, H_j, \delta) + d$, we can conclude $prob(H_i + d + max(H_j) - t \geq H_j) < \delta$ holds; likewise, given any time point $t \leq stime(H_i, H_j, \delta) + d$, we can conclude $prob(H_i + d + max(H_j) - t \geq H_j) \geq \delta$ holds.

Before proceeding to our partitioning method, let us first explain how to find the satisfaction time $stime(H_i, H_j, \delta)$. Consider the histograms H_i and H_j. Let T_{iu} and T_{jv} be the list of the time points in H_i and H_j where $n = |H_i|$ and $m = |H_j|$ as in Definition 3. We use the term *configuration* to denote a tuple $(c_1, c_2, \ldots, c_{m+1})$ where $c_k = 0$ if $T_{jk} < T_{i1}$, $c_k = l$ if $T_{il} \leq T_{jk} < T_{il+1}$ $(l = 1, \ldots, n)$, and $c_k = n$ if $T_{jk} \geq T_{in+1}$ for $k = 1, \ldots, m + 1$. Then, for $prob(H_i - x \geq H_j)$ where x is a variable, we may enumerate all possible configurations that pertain to H_i and $H_j + x$ by increasing x where $min(H_i) - max(H_j) \leq x \leq max(H_i) - min(H_j)$. For each configuration, we compute the lower and upper bounds of $prob(H_i - x \geq H_j)$. By Lemma 3, these bounds in a configuration are achieved by calculating $prob(H_i - x \geq H_j)$ with the minimum and maximum possible values of x in the configuration. Once the configuration bounding δ is identified, we formulate the quadratic equation corresponding to $prob(H_i - x \geq H_j) = \delta$ in the configuration and then we get $stime(H_i, H_j, \delta) = x + max(H_j)$ exactly by solving the equation[3].

In practice, we can find the configuration bounding δ efficiently by using the binary search, delimiting the range of x to $max(min(H_i) - t_j, t_i - max(H_j)) \leq x \leq min(max(H_i) - t_j, t_i - min(H_j))$ where t_i is the time point such that $P(X_i \leq t_i) = 1 - \delta$ and t_j is the time point such that $P(X_j \leq t_j) = \delta$. In particular, this method efficiently prunes unnecessary configurations in case that the number of buckets in histogram timestamps is large.

Example 4 *Consider the satisfaction time $stime(H_3, H_2, 0.8)$ where H_2 and H_3 are the histogram timestamps of a_2 and a_3 in Example 3. From $prob(H_3 - 90 \geq H_2) = 0.925$ and $prob(H_3 - 100 \geq H_2) = 0.769$, we obtain the bounding configuration* $(0, 1, 1, 2, 3)$

[3] We restrict *overlapping* histograms to the case where there are at least a pair of intersecting buckets with non-zero probability.

Proceedings of the 12th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA'06)
0-7695-2676-4/06 $20.00 © 2006 IEEE

for $prob(H_3 - x \geq H_2) = 0.8$ *where* $90 \leq x < 100$. *Let* $y = min(H_2)+x = 70+x$ *and* $H_2' = H_2+x = H_2+y-70$ *for notational convenience, and we obtain the adjusted histograms* $\tilde{H}_3 = ((I_1, 0), (I_2, 0.005y - 0.8), (I_3, 0.05), (I_4, -0.005y + 0.85), (I_5, 0.03y - 4.8), (I_6, -0.03y + 5.1), (I_7, 0.06y - 9.6), (I_8, -0.06y + 10.2))$ *and* $\tilde{H}_2' = ((I_1, -0.015y + 2.55), (I_2, 0.015y - 2.4), (I_3, 0.3), (I_4, -0.04y + 6.8), (I_5, 0.04y + -6.4), (I_6, -0.015y + 2.55), (I_6, 0.015y - 2.4), (I_7, 0))$ *of* H_3 *and* H_2' *respectively where* $I_k = [y, 170], [170, y + 10], [y + 10, y + 20], [y + 20, 190], [190, y + 30], [y + 30, 200], [200, y+40], [y+40, 210]$ *for* $k = 1, \ldots, 8$. *Then, following Theorem 1, we have*

$$ge(H_3, H_2') = \sum_{u=1}^{8}\sum_{v=1}^{u-1} \tilde{p}_{3u}\tilde{p}_{2v} + \frac{1}{2}\sum_{u=1}^{8} \tilde{p}_{3u}\tilde{p}_{2u}$$
$$= -0.0007625y^2 + 0.236y - 17.315 = 0.8$$

where $\tilde{p}_{3u}$ *and* $\tilde{p}_{2v}$ *denote the probabilities in* $\tilde{H}_3$ *and* $\tilde{H}_2'$. *By solving the equation, we have* $y = 168.59$ *and subsequently we have* $stime(H_3, H_2, 0.8) = x + max(H_2) = y - min(H_2) + max(H_2) = 168.59 - 70 + 110 = 208.59$.

Theorem 2 *Let* $a_i \models \phi_u$ *in a stream* A *and* $b_j \models \phi_v$ *in a stream* B. *If the inequality*

$$stime(\phi_u, \phi_v, \delta) + d + max(H_i) - max(\phi_u) \geq max(H_j) \quad (8)$$

holds, then it must be the case $prob(H_i + d \geq H_j) \geq \delta$. *Otherwise,* $prob(H_i + d \geq H_j) < \delta$ *holds.*

Proof: *Let* $t_s = stime(\phi_u, \phi_v, \delta) + d + max(H_i) - max(\phi_u)$. *Then we have* $t_s = stime(\phi_u + d + max(H_i) - max(\phi_u), \phi_v, \delta)$ *by Equation 7 and subsequently we have* $prob(H_i + d - t_s + max(\phi_v) \geq \phi_v) = \delta$ *according to Definition 8. Consider the case* $t_s \geq max(H_j)$ *and let* $\theta = t_s - max(H_j) \geq 0$. *Then we have* $prob(H_i + d \geq H_j) = prob(H_i + d - max(H_j) + max(\phi_v) \geq \phi_v) = prob(H_i + d - t_s + max(\phi_v) + \theta \geq \phi_v) \geq prob(H_i + d - t_s + max(\phi_v) \geq \phi_v) = \delta$ *by Equation 7 and Lemma 3. We omit the proof for the other case.* □

Theorem 2 and Lemma 2 provide a join method that uses the satisfaction time of the respective template histograms for evaluating the join condition ***jc***. Consider events $a_i \models \phi_u$ and $b_j \models \phi_v$. If $max(H_j) \geq max(H_i)$ holds, as shown above, we can check Equation 8 for evaluating $prob(H_i + d \geq H_j) \geq \delta$; otherwise, we can check if the inequality

$$stime(\phi_u, \phi_v, 1-\delta) - d + max(H_i) - max(\phi_u) \leq max(H_j) \quad (9)$$

holds for evaluating $prob(H_j + d \geq H_i) \geq \delta$. Due to the inverse relation such as $prob(H_j + d \geq H_i) = 1 - prob(H_i - d \geq H_j)$, we have $prob(H_j + d \geq H_i) \geq \delta$ holds iff $prob(H_i - d \geq H_j) \leq 1 - \delta$ holds.

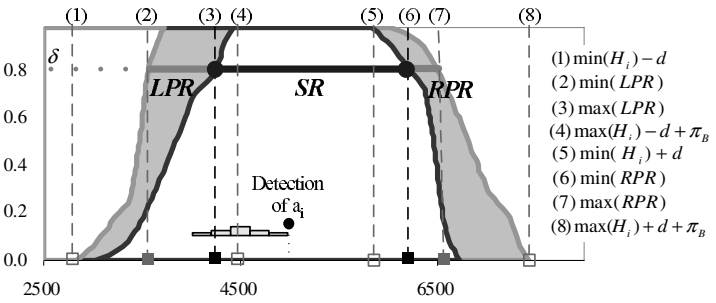

Figure 3. An example graph for partitions: The time points $(2), (3), (6)$ and (7) represent the partitions by Theorem 3 while the time points $(1), (4), (5)$ and (8) represent the partitions by Equation 5 and 6.

Notice that such a satisfaction time for a specific threshold, i.e., δ and $1 - \delta$, can be pre-computed for a set of template histograms at query installation time, and thus evaluating the join condition for an event pair in **PR**s can be done in constant time. Given a set of template histograms, we can determine **PR**s tightly by the following theorem.

Theorem 3 *Consider* ***jc****:* $(d \geq max(\pi_A, \pi_B), \delta)$. *Upon arrival of* $a_i \models \phi_u$ *in a stream* A, *we have the partitions in a stream* B *as follows:*

$$\texttt{LPR} = [\min_{\phi_v \in \mathcal{H}_B} stime(\phi_u, \phi_v, 1-\delta) + g(-d), \max_{\phi_v \in \mathcal{H}_B} stime(\phi_u, \phi_v, 1-\delta) + g(-d)),$$
$$\texttt{RPR} = (\min_{\phi_v \in \mathcal{H}_B} stime(\phi_u, \phi_v, \delta) + g(d), \max_{\phi_v \in \mathcal{H}_B} stime(\phi_u, \phi_v, \delta) + g(d)]$$

where $g(x) = max(H_i) - max(\phi_u) + x$.

Proof: *Let us denote two time points of* **RPR** *by* $rpr_1 = \min_{\phi_v \in \mathcal{H}_B} stime(\phi_u, \phi_v, \delta) + g(-d)$ *and* $rpr_2 = \max_{\phi_v \in \mathcal{H}_B} stime(\phi_u, \phi_v, \delta) + g(-d)$. *Consider an event* $b_r \models \phi_k$ $(\phi_k \in \mathcal{H}_B)$ *such that* $max(H_r) > rpr_2$ *and* $max(H_r) \geq max(H_i)$ *hold. And further let* $t_k = stime(\phi_u, \phi_k, \delta) + max(H_i) - max(\phi_u) + d$. *By Theorem 3, then, we conclude that since* $max(H_r) > rpr_2 \geq t_k$ *holds,* $prob(H_i + d \geq H_r) < \delta$ *holds, and thus the event pair* a_i *and* b_r *do not qualify* ***jc***. *Conversely, consider an event* $b_l \models \phi_k$ *such that* $max(H_l) \leq rpr_1$ *and* $max(H_l) \geq max(H_i)$. *By Theorem 3, then, we conclude that since* $max(H_l) \leq rpr_1 \leq t_k$ *holds,* $prob(H_i + d \geq H_l) \geq \delta$ *holds, and thus the event pair* a_i *and* b_l *qualify* ***jc***. *As explained in Lemma 2,* $max(H_l) \geq max(H_i)$ *guarantees* $prob(H_l + d \geq H_i) = 1$. *We omit the further proof since we can use the inverse relation for* **LPR**. □

Proceedings of the 12th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA'06)
0-7695-2676-4/06 $20.00 © 2006 IEEE

Example 5 *Figure 3 depicts an example graph of partitions over the max-time sorted stream B for performing* PTJoin *with $d = 1500$ upon arrival of event a_i as formulated in Theorem 3. In this example, we randomly create 500 template histograms for $\mathcal{H}_A$ and $\mathcal{H}_A$, and get a_i's timestamp by $H_i = \phi_j + t_{det} - max(\phi_j)$ where t_{det} is the detection time of the event and ϕ_j is a template histogram randomly chosen from $\mathcal{H}_A$. In the graph, the X-axis represents the timeline of the max-time sorted stream B in milliseconds and Y-axis represents the probability. The figure clearly shows two* **PR**s *and* **SR** *when $\delta = 0.8$ and $t_{det} = 5000$, highlighting their tightness with the comparison of those by Equation 5 and 6.*

3.2 Managing Event Buffers

Whereas most conventional sliding window joins include an explicit clause for the time window in their join specification and rely on event buffering for dealing with any possibly delayed event arrivals, we can derive the sliding time window tightly by exploiting the join condition of *PTJoin* and stream parameters.

Theorem 4 *Consider* ***jc****: $(d \geq max(\pi_A, \pi_B), \delta)$. When the join operation triggered by a_i in a stream A is processed, any event b_j in a stream B such that $max(H_j) < max(H_i) - \mathcal{D}_A - d + t_{min}$ must be obsolete where $t_{min} = \min_{\phi_u \in \mathcal{H}_A, \phi_v \in \mathcal{H}_B} stime(\phi_u, \phi_v, 1-\delta) - max(\phi_u)$.*
Proof: *According to Definition 6, when processing a_i in A, we may have the maximum possibly delayed event a_j in A such that $max(H_j) = max(H_i) - \mathcal{D}_A$. Assuming such an event $a_j \models \phi_k$, consider time $t_l = \min_{\phi_v \in \mathcal{H}_B} stime(\phi_k, \phi_v, 1-\delta) + max(H_i) - max(\phi_k) - d$ as similarly shown in Theorem 3. Any event whose max time is earlier than t_l cannot be a join partner of a_j due to the property of* PR*s. The minimum possible t_l by all possible template histograms $\phi_u \in \mathcal{H}_A$ guarantees that any possible pair of a_j in A and b_l in B such that $max(H_l) < t_l$ must violate the join condition.* □

A stream buffer can be implemented as a circular buffer such that front and rear pointers are maintained to locate the valid sliding time window over a max-time sorted stream. A stream buffer grows as a new event is placed at the rear position, but it shrinks as obsolete events are removed by advancing the front pointer. Based on Theorem 4, the sliding time window over a stream for *PTJoin* can be tightly constructed whenever a newly arrived event is processed and inserted into the stream buffer by setting the front pointer to just after the latest obsolete event. Notice that the time offset to the latest obsolete events is a constant that can be calculated at query installation time by comparing the satisfaction time of all template histogram pairs. It should be noted that the stream parameter for denoting the maximum possible network delays, i.e., $\mathcal{D}_A$ is needed to ensure that the invalidated events cannot be joined even if some event triggering the join operation may arrive late within the parameter value.

4 Conclusion

In this paper, we have addressed the issue of processing timing joins over temporally uncertain event streams where the exact time of event detection may be known only probabilistically. This paper generalizes all previous work in that our probabilistic model allows arbitrary histograms to be used for quantifying the uncertainty in event timing. For this model, we develop a partitioning scheme for checking the satisfaction of a timing constraint by exploiting the probabilistic properties and the temporal ordering of event arrivals in a data stream. By using this partitioning scheme, we showed how to implement a timing join operator efficiently.

References

[1] T. Anderson, M. Moran, S. Ketcham, and J. Lacombe. Tracked Vehicle Simulations and Seismic Wavefield Synthesis in Seismic Sensor Systems. *IEEE Computing in Science and Engineering*, 6(6):22–28, November/December 2004.

[2] D. Carney, U. Cetintemel, M. Cherniack, C. Convey, S. Lee, G. Seidman, M. Stonebraker, N. Tatbul, and S. Zdonik. Monitoring streams: A new class of data management applications. In *Proc. of the International Conference on Very Large Data Bases(VLDB)*, pages 215–226, 2002.

[3] R. Cheng, D. V. Kalashnikov, and S. Prabhakar. Evaluating Probabilistic Queries over Imprecise Data. In *Proc. of the ACM SIGMOD International Conference on Management of Data*, pages 551–562, 2003.

[4] A. Deshpande, C. Guestrin, and S. Madden. Using Probabilistic Models for Data Management in Acquisitional Environments. In *Proc. of Conference on Innovative Data Systems Research(CIDR)*, pages 317–328, 2005.

[5] C. E. Dyreson and R. T. Snodgrass. Supporting Valid-time Indeterminacy. *ACM Transactions on Database Systems*, 23(1):1–57, March 1998.

[6] J. Kang, J. F. Naughton, and S. Viglas. Evaluating Window Joins over Unbounded Streams. In *Proc. of the International Conference on Data Engineering*, pages 341–352, 2003.

[7] C.-G. Lee, P. Konana, and A. K. Mok. Applying Interval Timing Monitoring to Stream Data Processing: Interval Timing Join for Streaming Data. *ready for journal submission*, 2006.

[8] C.-G. Lee, A. K. Mok, and P. Konana. Monitoring of Timing Constraints with Confidence Threhold Requirements. In *Proc. of IEEE Real-Time Systems Symposium(RTSS)*, pages 178–187, December 2003.

[9] U. Srivastava and J. Widom. Flexible Time Management in Data Stream Systems. In *Proc. of the ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pages 263–274, 2004.

Proceedings of the 12th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA'06)
0-7695-2676-4/06 $20.00 © 2006 IEEE