Moving Range Queries in Distributed Complex Event Processing

Boris Koldehofe, Beate Ottenwälder, Kurt Rothermel
Institute of Parallel and Distributed Systems
University of Stuttgart
⟨firstname.lastname⟩@ipvs.uni-stuttgart.de

Umakishore Ramachandran
College of Computing
Georgia Institute of Technology
rama@cc.gatech.edu

ABSTRACT

Up to now, correlations in complex event processing (CEP) systems are detected by a well defined set of operators, whose configuration is determined ahead of deployment time. Although CEP operators involve location specific attributes, state of the art systems are heavily constrained in detecting situations where the interest in a situation changes depending on the consumer's location, e.g., with the movement of mobile devices.

This paper adopts the concept of range queries to CEP systems. We propose a mobility-aware event delivery semantics and present a corresponding execution model, which accounts for mobility driven selection of primary event streams to the CEP system. By utilizing the properties of this execution model, we derive an algorithm that establishes low cost and coordinated reconfiguration of CEP operators in a distributed system. The algorithm minimizes the amount of information that needs to be streamed between operators and avoids additional delays as a result of a reconfiguration of CEP operators. We present an analysis of the algorithm's properties and evaluate the efficiency of the proposed reconfiguration algorithm.

Categories and Subject Descriptors

C.2.1 [**Network Architecture and Design**]: Distributed networks; C.2.4 [**Distributed Systems**]: Distributed applications; D.2.11 [**Software Architectures**]: Data abstraction

Keywords

mobility, complex event processing, moving range queries

1. INTRODUCTION

Range queries [19, 27] are one of the most popular coordinate-based queries in the context of location-based services (LBS). They return data relative to a consumer-specified spatial range. Typical examples of such queries are *"retrieve all restaurants close to my current location"* or *"retrieve all objects that were inside a Region R between 11:00 and 12:00"*. In addition, *moving range queries* [32, 7] continuously retrieve all data of interest in a range depending on the location of a moving *focal object*, e.g., a car querying for nearby traffic-lights.

The processing of range queries is currently performed only on primary event streams, i.e., events stemming from dedicated data sources like sensors or databases. Yet range queries would also provide a powerful abstraction for complex event processing in which complex events are the result of correlations over several incoming event streams. For instance, queries such as *"how probable is it that a traffic jam occurs on the roads ahead of my current position"* or *"inform me when I can switch lanes safely within a Region R around my current position"* comprise traffic situations that are restricted to a certain *range of interest*. These situations themselves can be detected from a correlation of several traffic patterns such as movement and acceleration patterns of cars in the range. Processing such correlations can be performed by multiple dependent CEP operators, each detecting a specific traffic pattern.

State of the art CEP systems only have very limited support to deal with mobility driven changes in the range of interest. Operators are configured ahead of their deployment, e.g., to perform correlations on primary events originating from a fixed range. Hence, supporting moving range queries requires to define and predeploy a dedicated set of CEP operators for each potential range of interest. Depending on their current location, moving consumers receive event-streams from the predeployed range that currently matches the most with their interest. Thus, with a higher number of possibly overlapping predeployments, a consumer's range of interest matches more tightly with one of the predeployed ranges, increasing the accuracy.

Depending on the required accuracy, the predeployment of ranges can cause a significant amount of unnecessary processing where none of the produced events are of interest to a deployed moving range query. For example, consider a traffic monitoring system which guarantees, for each moving range query, an accuracy of 100 m maximum deviation from the actual range of interest whereas each range covers a region of size 1 km^2. Then approximately 23,100,000 overlapping ranges have to be predeployed to cover the core road network of Germany. However, due to the large variations in traffic — from 5,000 upto 180,000 cars per day for the

DEBS '12, July 16–20, 2012, Berlin, Germany.

main road segments — we would expect 40% of all ranges to be idle 95% of their deployment time*.

In this paper, we investigate the potential in supporting dynamic reconfigurations of CEP operators in order to cope with mobility driven changes in the range of interest of consumers. This way, the CEP system is obliged to only monitor the ranges which are currently of interest to the consumers of complex events. Supporting low latency and communication efficient reconfiguration, however, comprises several challenges. With each update of a query's range, CEP operators that perform range specific processing of complex events need to be linked to new data sources. Operators are stateful and require with every change in the range of interest also a reconfiguration of the operators' states, i.e., a reconfiguration needs to account for new and obsolete events that result from a changed selection of primary events. Especially in distributed CEP systems, the output of operators also serves as input to other operators, hence a reconfiguration requires the coordinated update of the operators' state. The goal of this paper is to support the mobility driven reconfiguration of CEP operators, which minimizes the time to perform coordinated updates as well as the imposed reconfiguration overhead of mobility aware correlations.

In more detail our contributions are:

1. The definition of a mobility-aware semantics that deals with mobility driven updates of ranges of interest
2. An execution model that realizes the mobility-aware event processing semantics for CEP operators
3. A reconfiguration algorithm and optimization that dynamically reconfigures operators to a new range of interest
4. An analysis and evaluation study of the cost imposed by the mobility-aware reconfiguration algorithm

The remainder of the paper is structured as follows: In Section 2 we introduce the underlying system and event processing model. Section 3 defines a mobility-aware delivery semantics for event streams. To realize the delivery semanticss, we present in Section 4 an execution model which enables the mobility driven reconfiguration of an operator's state. Furthermore, in Section 5 we present the approach to dynamically reconfigure multiple dependent operators and its optimizations. The findings from an analysis and empirical evaluation of the approach is presented in Section 6. Section 7 discusses related work and we conclude the paper with a discussion of the results as well as an outlook on future work in Section 8.

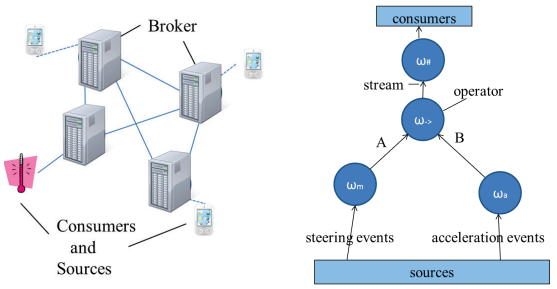
2. SYSTEM AND CEP MODEL

In this section we present the system model and introduce the basic operations of a CEP system.

2.1 System model

We consider a distributed CEP system to consist of a network of brokers (cf. Figure 1(a)). The brokers provide event

*The traffic rates were obtained from a current survey on the German traffic density [1]. We assumed an average speed of 25 m/s and that for 40% of all road segments 10,000 cars are passing per day and a large number of 10% of the traffic participants where interested in traffic events.



(a) System Model (b) CEP

Figure 1: System Model and CEP Operator Tree

streams of interest to a dynamic set of mobile consumers. In order to establish event streams, brokers themselves can connect to event streams from mobile sources or event streams provided by other brokers of the CEP system. Brokers are dedicated infrastructure nodes which support reliable communication. Consumers and sources either use mobile devices, e.g., smartphones, or smart head-units, or static devices, e.g., a thermometer attached to a wall. All mobile devices establish a wireless connection to the nearest broker of the broker network (e.g., via UMTS or WLAN). We further assume that consumers and sources may determine their current time and location, e.g., through a GPS sensor.

2.2 Operator graph and event model

The operation of a distributed CEP system is commonly modeled by an *operator graph*. The operator graph is a directed graph $G = (\Omega \cup S \cup C, L)$, where Ω denotes the set of operators, S the set of sources, C the set of consumers, and $L \subseteq (\Omega \cup S \times \Omega \cup C)$ the event streams between operators, sources, and consumers.

Each operator $\omega \in \Omega$ is hosted by some broker and implements a correlation function f_ω, which defines the mapping of events from the operator's incoming event streams $(in, \omega) \in L$ to its outgoing event streams $(\omega, out) \in L$. The correlation function is applied to a dynamically changing *selection* S of events for which $f_\omega(S)$ determines a possibly empty set of events *produced* in a single correlation step. The selections for a sequence of steps are determined by an operator specific *selection policy*. Furthermore, an operator-specific *consumption policy* determines events to be discarded from future selections. The current selection of events as well as the computational state of f_ω makes up the *state* of the operator.

The events of an event stream are constituted by a finite set of attribute-value pairs, i.e., an event σ is of the form $\sigma = \{(a_1, v_1), \ldots, (a_m, v_m)\}$. Each stream $(s, d) \in L$ is composed of events produced at s using arbitrary restrictions on the attribute values of its events. Moreover, each event has an attached *timestamp* ($\mathrm{t}(\sigma)$) and *location* ($\mathrm{loc}(\sigma)$) information, determining when and where it occurred. To capture their inherent uncertainty, e.g., introduced by GPS sensors, timestamp and location informations are often described by a temporal and spatial interval. For simplicity, but without restricting the generality of the approach, time and location attributes are represented by discrete points in time and space.

Each consumer can register *queries* with the system for streams of interest. When referring to a query we also refer

to its imposed *operator tree*[†], whose root is directly linked to the consumer. The set of all overlapping operator trees forms the operator graph.

Figure 1(b) shows an example of a simple operator tree to detect an accident on a road segment. Primary events stem from sensors deployed in a car, which allow to determine the movement (cf. operator ω_m) and acceleration pattern (cf. operator ω_a) of individual cars. In subsequent correlation steps, a sequence of several similar movement and acceleration patterns (cf. operator $\omega_{\rightarrow}$) determines whether a lane of the road segment is blocked by an accident. We are subsequently refining this example to illustrate the concepts of the paper.

3. MOBILITY-AWARE CEP

As a next step, we introduce the changes in the event delivery semantics of CEP for dealing with moving range queries. Note, in a traditional setting the change in the operator graph is only performed whenever a new query is added or removed. However, in a mobility-aware CEP system the interest in the events provided by the operator graph change with the movement of a consumer. Therefore, the sources and the corresponding primary event streams require a constant update.

In order to support dynamic updates of the operator graph, a consumer registers a *dynamic interest query*, $diq = \{T, mo, R, \delta\}$, where T refers to the querie's imposed operator tree, mo to a monitored focal object that provides discrete updates of its location, R to a range determining the selection of sources of T relative to the position of mo, and δ to the lifetime of a situation, i.e., the lifetime of those complex events which are produced by T and which are directly delivered to the consumer. In the context of the introduced accident example, a car's head-unit can register a diq with the CEP system for all accidents within a region (e.g., of 1 km^2) around its current location. The monitored focal object is associated with the car's location and provides a sequence of discrete location updates[‡] to the CEP system. From the relative range R and the updates of mo, the CEP system can determine a sequence of changing ranges of interest $R_1, .., R_m$ and adapt the operator tree by linking T to the data sources of each range R_i. In addition, the lifetime parameter δ allows us to detect events that occurred δ time-units before the range of interest changed. Consider an accident that occurred 15 minutes before the range of interest comprises the location at which the accident happened. This accident is still of interest to the consumer, as it can still block a lane.

However, dealing with changing ranges of interest requires a different delivery semantics for complex event processing to account for the dynamically changing data sources. In the following, we propose a delivery semantics that defines which events are to be delivered as well as the delivery order in the presence of changing updates. We focus on two important properties, the *completeness* of the delivery and a *space-time ordering* of the event-stream. In order to assure that no consumer misses a potentially relevant event, e.g., an accident that blocks its road, we define the completeness property. A consumer should also be able to match the events of a resulting event-stream of T to a specific range of interest, which is assured by a space-time ordering. For example, a navigational system should display only the accidents that happened in the current range of interest.

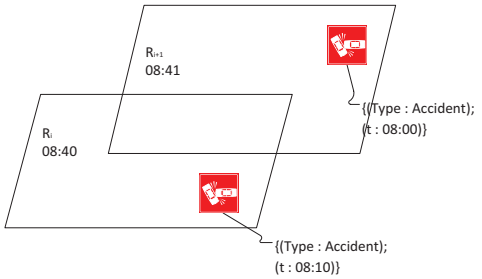


Figure 2: Accident Example

Completeness.

Given a $diq = \{T, mo, R, \delta\}$ and a sequence of range updates $R_1, \ldots, R_m$. For a range R_i, let $[start(R_i), end(R_i)]$ denote the time interval for which R_i was valid, i.e., the location update of R_i occurred at time $start(R_i)$ and the subsequent location update of R_{i+1} occurred at time $end(R_i)$. Furthermore, let $E(R_i) := \{\sigma_1, \ldots, \sigma_r\}$ denote a possible sequence of events to be produced by T, for which T is only linked to data sources in R_i. We say $E(R_i)$ is a *covering sequence* of R_i if for all $\sigma \in E(R_i)$ the temporal restriction $t(\sigma) \in [start(R_i) - \delta, end(R_i)]$ holds. $E(R_i)$ is called a *maximum covering sequence* of R_i if there exists no further covering sequence $E'(R_i) \neq E(R_i)$ s.t. $E(R_i) \subset E'(R_i)$.

Observe, the maximum covering sequence can be the empty set if T cannot detect any events covering R_i. Moreover, a maximum covering sequence is not necessarily uniquely defined. Consider the sequence operator $\omega_{\rightarrow}$ in the example of Figure 1(b). The operator determines for individual events in an incoming stream A, if they happened before individual events in an incoming stream B and consumes this selection after each correlation. Consider a location update of R_i occurs at time $t = 5$ and a subsequent location update of R_{i+1} occurs at $t = 9$. If the lifetime is set to $\delta = 2$, a maximum covering sequence of R_i produced by $\omega_{\rightarrow}$ contains all events with timestamps in the temporal interval $[3, 9]$. Further consider $\omega_{\rightarrow}$ receives after the location update of R_i the following input event sequence, and either initializes the correlation with σ_1, or with σ_2:

$$\begin{aligned}\{\quad \sigma_1 &:= \{(Type : A), (t : 1), (loc : (x_1, y_1) \in R_i)\},\\ \sigma_2 &:= \{(Type : A), (t : 2), (loc : (x_2, y_2) \in R_i)\},\\ \sigma_3 &:= \{(Type : B), (t : 4), (loc : (x_3, y_3) \in R_i)\},\\ \sigma_4 &:= \{(Type : B), (t : 6), (loc : (x_4, y_4) \in R_i)\},\\ \sigma_5 &:= \{(Type : B), (t : 8), (loc : (x_5, y_5) \in R_i)\}\}\end{aligned}$$

[†]Note, this tree does not necessarily need to be specified by the consumer. While giving high flexibility to the user in specifying individual event streams, an administrator can also predefine operator trees for valid event streams.

[‡]To implement location updates there exists already several possibilities in the literature: A location can be either reported directly from the mobile device or by a component tracking it via dead-reckoning [30], either in fixed temporal/spatial intervals [16] or as soon as the underlying set of sources may change, by utilizing safe zones, [7].

Then $E_1(R_i) = \{\{(Value : \sigma_1 \rightarrow \sigma_3), (t : 4))\}, \{(Value : \sigma_2 \rightarrow \sigma_4), (t : 6))\}\}$ and $E_2(R_i) = \{((Value : \sigma_2 \rightarrow \sigma_3), (t : 4))\}$ are two maximal event sequences which result from performing the correlation from the two different initialization points in the input stream.

We adopt a weak and strong formulation for the event delivery semantics:

1. A *weak delivery semantics* will deliver for every range $R_i \in \{R_1, ..., R_m\}$ all events of a maximum covering sequence $E(R_i)$.

2. Let $E_1(R_i)$ and $E_2(R_j)$ be two maximum covering sequences for T w.r.t. two distinct focal objects mo_1 and mo_2. A *strong delivery semantics* ensures for every pair of ranges (R_i, R_j), which refer to the same spatial region, that $E_1(R_i)$ and $E_2(R_j)$ will comprise exactly the same events within their temporal overlap $[\max\{(start(R_i), start(R_j)\}, \min\{end(R_i), end(R_j)\}.]$

The strong semantics assures to consumers that maximum covering sequences of a specific range R_i are consistent to other maximum covering sequences of that range referring to different focal objects. In the above example the strong semantics assures that if the location update of a focal object mo_1 for R_i occurred at $t = 5$ all events σ of a maximum covering sequence with $\mathrm{t}(\sigma) \in [3, 9]$ are generated. In the presence of a location update of a focal object mo_2 for R_j at $t = 6$ which is referring to the same range as R_i, receiving all events σ with $\mathrm{t}(\sigma) \in [4, 10]$, consumers will receive identical covering event sequences within the temporal overlap $[5, 9]$.

Order of events.

In a mobility-aware CEP system the order of events should respect the temporal order relation imposed by the timestamps of events, but also the order of updates of its ranges. In order to make location updates also visible to the consuming application, we favor the order imposed by range updates, i.e., the order of events yields for any pair of events $\sigma_i \in R_i$ and $\sigma_j \in R_j$: σ_i is delivered before σ_j iff either of the following two properties below hold:

i. $i < j$ or

ii. $i = j$ and $t(\sigma_i) < t(\sigma_j)$

Note that this order does not necessarily result in a chronological time order. Consider the example depicted in Figure 2. Two accidents occurred at time 8:00 and 8:10. However, due to the order of the range updates the event with $\mathrm{t}(\sigma_2) = 8{:}00$ is delivered after the event with $\mathrm{t}(\sigma_1) = 8{:}10$.

In order to allow consumers to explicitly deal with changes in the range of interest, we require two subsequent events produced with respect to two different ranges to be separated between by a special *marker* event M.

Timestamps.

Note, that the delivery semantics depends also on the timestamping mechanisms introduced in the CEP system. A complex event can depend on a number of events with different timestamps. Common ways to perform the timestamping is to assign the maximum or minimum timestamp of the events inside the operator's current selection of events. Throughout the paper we use a timestamping scheme that assigns the maximum timestamp of the selection to each complex event. However, in general timestamping can be considered a parameter which can be specified by the domain expert. The results of this paper can be easily adapted if timestamps are assigned deterministically and in monotonously increasing order.

```
1: Requires: Operator ω with S the set of selection windows,
   rw the restriction window, f the correlation function
2: Defines: Buff[IN] ← ∅ //Buffer for individual incoming
   streams
3: upon receive Event(Event σ, Incoming stream in)
4:   Buff[in] ← Buff[in] ⋃ σ
5: end
6: upon selection changed()
7:   f(S)
8: end
9: upon correlation finished()
10:   send partial results over outgoing streams
11:   if (∃σ ∈ Buff[IN] : (t(σ) > end(rw)))
12:      ∧(∃sw ∈ S : notFull(sw)) then
13:      //Initialization of restriction window
14:      close(S)
15:      updateWindows(rw, shiftLength(rw))
16:      open(S)
17:   else
18:      if ∀sw ∈ S : isFull(sw) then
19:         //Consumption
20:         comsumeEvents(Buff[IN], consumptionPolicy)
21:         close(S) //Change Selection
22:         applyShift(S, shiftPolicy) //Selection
23:         open(S)
24:      end if
25:   end if
26: end
27: upon marker selected(Marker M)
28:   send M over outgoing streams
29:   for ∀in ∈ IN : ∃Marker M' ∈ in with t(M') =t(M) do
30:      comsumeEvents(Buff[in],t(M))
31:   end for
32:   close(S)
33:   updateWindows(rw, Δ_in)
34:   open(S)
35: end
36: function updateWindows( Window w, time t)
37:   setBound(w, t)
38:   for ∀sw depending on w do
39:      updateWindows(sw, depend(w))
40:   end for
41: end
```

Figure 3: Selection and Consumption

Problem description.

The problem addressed in the next sections is to establish a mobile CEP system that yields the introduced delivery semantics by providing guarantees for completeness and event delivery order. In addition, the algorithms realizing the delivery semantics should guarantee low reconfiguration time for the operators of T as well as ensure a low reconfiguration overhead.

4. MOBILITY EXECUTION MODEL

In this section, we define an execution model for event processing of a single operator. This execution model provides the foundation for the later described cost efficient reconfiguration of the operator tree in order to establish the introduced delivery semantics. In Section 4.1, we first introduce the concepts for the basic event correlation approach.

These concepts can be easily integrated with most existing CEP correlation engines. However, the reconfiguration of operators bears a huge optimization potential that requires to determine accurate initialization points for each operator as well as the temporal and spatial overlap between the operator state within two subsequent range updates. Therefore, we further propose in Section 4.2 an expressive model to realize selection and consumption policies based on a partitioned window model. The model allows to implement the well known consumption and selection policy of traditional CEP systems. We detail the main properties of the proposed execution model with respect to its expressiveness and the properties that are used in achieving subsequent adaptations in Section 4.3.

4.1 Operator Execution

Recall, that the execution of an operator $\omega \in \Omega$ comprises a sequence of correlation steps. For each correlation step a selection S of events needs to be determined, for which the correlation function f_ω can be applied to produce an outgoing event sequence $f_\omega(S)$. The execution environment of the operator controls the execution of an operator by determining the selections for each correlation step and managing the incoming and outgoing event streams. To determine the sequence of event selections in subsequent correlation steps, the execution environment relies on selection and consumption policies provided as part of the operator description. In addition, the execution environment — in order to account for the mobility-aware delivery semantics — also has to ensure a reconfiguration of the operator's state if updates of ranges for a *diq* occur.

Initialization.

The setup of an operator's execution environment is performed with the deployment of a $diq = \{T, mo, R, \delta\}$. For each $\omega \in T$ a separate execution environment needs to be initialized. The execution environment ensures that the events for a selection will be present locally in a buffer such that the correlation function f_ω can be applied. To achieve this, the execution environment initially determines for each of its incoming streams $(in, \omega) \in T$ the point in time Δ_{in} from which the event stream needs to be processed in order to generate a maximum covering sequence of the initial range R_1. Recall that the timestamp of a complex event comprises the timestamps of a selection of multiple incoming events. Therefore, this point in time needs to be individually determined for each input stream as well as for every operator in T. To guarantee a maximum covering sequence for a range this requires a coordinated decision of all operators in T, which will be introduced in Section 5.1.

Event Correlation.

Let in denote a preceding operator or source of $\omega \in T$ that produces events with respect to range R_i. After an initialization or a reconfiguration step, each operator or source in will produce event streams for which Δ_{in} denotes the timestamp of the first produced event. Observe, that Δ_{in} is a point in time that lies in the past. Hence, the operator performs event processing initially on historic events, before it continues processing on live events. Each incoming stream yields the order semantics introduced in Section 3, i.e., all events in between two marker messages are delivered in chronological order. Also the selection determined from all incoming streams will be streamed to f_ω respecting the delivery order. Hence, the correlation function can already perform processing steps on partial selections while other events still need to be streamed from ω's predecessors. A correlation step is finished whenever the correlation function terminates processing its current selection. As its result $f_\omega(S)$ produces a possibly empty sequence of events. Each $\sigma \in f_\omega(S)$ will be assigned a timestamp, which we choose to be the maximum timestamp of events in the current selection[§]. For each successor of ω, i.e., $out \in T$, the execution environment provides an individual outgoing sequence that matches the specified restrictions on the attribute/value pairs of $f_\omega(S)$ and are delivered respecting the chronological order. Afterwards the selection of the next correlation step is determined by applying the selection and consumption policies that are part of the operator description (cf. Section 4.2).

Reconfiguration.

A reconfiguration of the operator has to be performed after a range update from R_i to R_{i+1} at time $end(R_i)$ occurs. Note, that at this point in time the operator may still need to produce events which are needed for T to generate a maximum covering sequence of R_i. For instance, the operator may still await events with a smaller timestamp $end(R_i)$ from its predecessor in T. Therefore, the reconfiguration of the execution environment is initiated only after a new selection comprises one or more marker message M, whose timestamp equals $end(R_i)$. In this case, the correlation step will only produce a marker message for all of its outgoing streams to initiate the reconfiguration of succeeding operators in T. Subsequently, the operator consumes all events of R_i and initializes itself with events of R_{i+1}. Afterwards, the correlation function and selection are reinitialized to generate a covering sequence of R_{i+1}.

4.2 Selection and Consumption

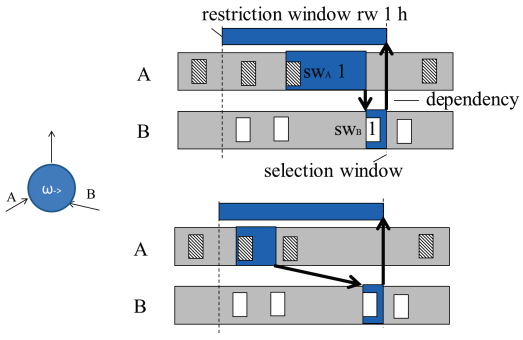


Figure 4: Selection

The basic concepts introduced in Section 4.1 can be integrated with any existing event processing technology. For instance, by realizing a wrapper that encapsulates the respective event correlation engine, and realizes initialization and reconfiguration of operators by selecting appropriate incoming event streams as well as filtering outgoing event streams produced by the respective correlation engine. The

[§]In addition, sequence numbers can be used if also an order for events with identical timestamps needs to be guaranteed.

DHEP framework [23] is a tool to build distributed correlation networks that are realized using heterogeneous correlation engines. Therefore, the basic concepts to realize the mobility-aware delivery semantics as well as their benefits can be achieved to a certain degree with many existing event processing systems. Yet, to further reduce the overhead that comes with a mobility-driven reconfiguration, we show that knowledge on the selection and consumption of events performed by the operator bears a huge potential to reduce this overhead. At the downside, the programmer of a CEP operator now needs to implement the operator against a specific programming model that exposes knowledge on the current state of an operator's selection.

For realizing the selection of events of incoming streams we propose to use a partitioned window model. Each buffered incoming stream is associated with its own *selection window sw* implementing a *selection policy*. However, the sequence of events on which the selection can be performed is restricted by three other mechanisms. First, a *restriction window rw* assures that only temporally relevant events can be selected, e.g., acceleration events from one day ago do not indicate a recent acceleration pattern. Second, dependencies between individual windows assure temporal orders between events of different incoming streams, e.g., for the sequence $\omega_{\rightarrow}$ on two streams A and B no selected event of stream A should have a greater timestamp than a selected event from B. Last, *consumed* events should not be selected. How these concepts are applied for the correlation and initialization is shown in the algorithm in Figure 3, and will be detailed in the remainder of this section.

Selection Windows.

Each selection window $sw = (type, length, shift\_policy, dependencies)$ determines *start* and *end* points in streams, where events in-between are selected for the correlation. The *type* defines if either a given time span (*temporalwindow*) or a count on the number of events (*countingwindow*) is comprised within the window. The *length* determines the respective length of the time span or the count on the number of events. The $shift\_policy$ determines how the starting and end points change after each correlation step. The *dependencies* assure temporal orders between selected events of different streams. For instance, an operator that is connected to two incoming streams A and B associates each of them with a separate selection window sw_A and sw_B. In case of a basic sequence operator $\omega_{\rightarrow}$ (cf. Figure 4), checking for a happens-before relationship $\sigma_A \rightarrow \sigma_B$ of individual events in those streams, these would be counting windows of length one.

The selection S of an operator therefore comprises the set of all selection windows. If this selection changes, the correlation function performs a correlation on a (partial) selection (Line 7) until it indicates the correlation is performed (Line 9). The selection is completed, when all windows are filled — counting windows have selected n events, temporal windows have selected all events within the time span t (Line 17). In this case, the selection windows are *shifted*, according to the $shift\_policy$. Each window is closed, new starting and end points are calculated on its incoming stream before they all are reopened and the correlation starts again (Line 20-22). It either shifts to the $history$, or $future$ of the stream for a fixed amount of non-consumed events or *not* at all. The first window sw_A can shift one event to the $history$, selecting historic events, and the other sw_B can shift one event to the $future$, waiting for the next event to arrive (cf. the second step in Figure 4).

Dependencies.

Other selection windows can limit the shift of a selection window. In our example, no event of stream A should have a timestamp greater than a selected event of stream B. Therefore, the selection policy also has to specify dependencies between selection windows. Each selection window defines that one of its bounds depends on another window's bound, expressing that the execution environment has to assure, that no window shifts beyond the defined dependent bound. However, at least one window has to depend on the restriction window and all other windows depend indirectly on the restriction window, to assure that they are initialized when the restriction window shifts. The end of window sw_A should not shift further than the beginning of window sw_B. In Figure 4 sw_A could not shift further to the future than the point indicated by the black arrow.

Restriction.

As discussed, events of incoming streams are only relevant for a correlation if they happen within a restricted amount of time. Therefore, an additional temporal *restriction window* rw of length $length(rw)$ has to be defined for each operator. Each selection window can only select events and shift within the beginning and end point of rw. Remark, that we assign the timestamp of the outgoing events according to the end point of rw.

The execution environment also assures that the restriction window will only shift if the selection windows cannot shift any further and all events with timestamps between rw's beginning and ending are processed (Line 11). Then rw shifts for either a fixed amount of time, or for a fixed number of events to the future. Each shift of rw initializes the selection windows anew and their bounds are calculated according to the dependencies of the windows (Line 12-15 / Line 36-39).

Consumption.

The *consumption policy* is carried out as soon as the correlation is finished (Line 19). It is defined individually for each selection window. It either consumes no events ($zero$), all events in the window ($window$), or consumes all events that are in the stream before the windows end ($before$). This means that consumed events are marked in the buffer for incoming streams, indicating that they cannot be selected in future rounds. However, they are not dropped from the buffer, because they can be reused for the mobility driven reconfiguration.

Initialization.

After a range update from R_i to R_{i+1}, marker messages are contained in the streams in order to provide the described delivery semantics. While processing these streams, an operator checks if a marker M is contained in the restriction window for any stream, which triggers the reinitialization (Line 26-34). First, all events σ with $\mathrm{t}(\sigma) < \mathrm{t}(M)$ will be consumed, however, only in those streams that contain marker messages with timestamp $\mathrm{t}(M)$, as we reuse the events from the non-reconfigured streams. Then the oper-

ator generates a marker messages for its outgoing stream, including the new range R_{i+1} and $start(R_{i+1})$. Finally, the restriction window rw is shifted to the initialization point of its succeeding operator, starting the correlation from there.

4.3 Properties

Although the model is inspired by traditional stream processing systems such as CQL [2], the introduced selection and consumption policy allows us to implement the typical CEP behaviour. As an example take the recent parameter context of SNOOP [6], where only the most recent occurrences of events are selected and consumed. In case of the previously described $\omega_{\rightarrow}$, each selection window is initialized at the end of the restriction window, as in the first step of Figure 4. Defining a shift-rule that always shifts sw's one event to the history and rw's one event to the future realizes the recent parameter context.

Typical CEP systems work with a single window over all merged input streams, however we use a partitioned stream and window model for an efficient reconfiguration. Recall that only those streams containing a marker require to be reconfigured and consume events of a previous range R_i. Consider an operator with n input streams, receives a marker over only one of the input streams. This would lead to costly insertions and removals, because it requires to remove all old events of the reconfigured incoming stream from the merged one and subsequently adding new events to the merged one. In our partitioned model these costs are circumvented, because only reconfigured incoming streams have to be revised.

The reconfiguration, performed after receiving the marker message, will produce a maximum covering sequence of R_{i+1}. The performed reinitialization assures that only events of R_{i+1} are correlated, because all events of previous ranges are consumed. Recall, that the timestamp mechanism always selects the greatest timestamp of the selection. Therefore, shifting the end of rw to the initialization point Δ_{in} of the succeeding operator makes sure that the first produced event σ is the first possible event with $\mathrm{t}(\sigma) > \Delta_{in}$. Further recall, that the reconfiguration is delayed until the marker is selected by rw. Therefore, it is assured that all events σ of the outgoing sequence with $\mathrm{t}(\sigma) \leq end(R_i)$ are produced, otherwise events with greater timestamps would be produced.

5. DISTRIBUTED RECONFIGURATION

For a $diq = \{T, fo, R, \delta\}$, a range update from R_i to R_{i+1} requires to determine for each $\omega \in T$ the initialization points Δ_{in} of all incoming event streams $(in, \omega) \in T$ in order to generate a maximum covering sequence of each operator. In this section we show how to

1. efficiently determine initialization points of a data stream even if operators are hosted at different brokers inside the network.
2. keep the overhead of an operator's reconfiguration low by proposing several optimizations that allow to minimize the amount of redundant information to be streamed between operators and therefore reduce the cost of a reconfiguration.

5.1 Determining Initialization Points

Since the reconfiguration of operators in T is triggered by receiving incoming Marker messages, finding initialization points for the operators in T breaks down to the problem of finding suitable initialization points of the sources of T. These can depend on the state of operators in T as well as the properties of the incoming event streams.

Observe, that from the size of the restriction windows of operators, we can easily determine an upper bound on the time to reproduce a primary event stream.

LEMMA 1. *Given a $diq = \{T, fo, R, \delta\}$ and a range update from R_i to R_{i+1}. Let p_s be a path in T from a source $s \in R_{i+1}$ to the root of T. If each data source s produces a stream with an individual initialization point Δ_s which yields*

$$\Delta_s \leq \mathrm{start}(R_i) - \delta - \left(\sum_{\omega \in p_s} \mathrm{length}(rw_\omega) \right)$$

then T will produce a maximum covering sequence of R_i.

PROOF. Recall, each operator assigns to each outgoing event σ the maximum timestamp of its selection S of incoming events. If the root operator ω_{root} produces an outgoing event with timestamp $\mathrm{start}(R_i) - \delta$, it requires at most the incoming events within the time span of the restriction window, i.e., incoming events with greater timestamps than $\mathrm{start}(R_i) - \delta - \mathrm{length}(rw_{\omega_{root}})$. All previous incoming events are no longer relevant for the correlation. Thus, any operator ω_1 on the level below the root has to produce outgoing events, starting from $\mathrm{start}(R_i) - \delta - \mathrm{length}(rw_{\omega_{root}})$. Again additionally requiring at most the incoming events within the time span of the restriction window, hence, $\mathrm{start}(R_i) - \delta - \mathrm{length}(rw_{\omega_{root}}) - \mathrm{length}(rw_{\omega_1})$. Thus, on the path $p_s = (\omega_{root}, \omega_1, .., \omega_n, s)$ from ω_{root} to s it sums up to $\mathrm{start}(R_i) - \delta - \mathrm{length}(rw_{\omega_{root}}) - .. - \mathrm{length}(rw_{\omega_n})$. □

In addition, Lemma 2 states that not all operators of T need to be reconfigured when a range update occurs, hence decreasing the overall reconfiguration cost.

LEMMA 2. *Given a $diq = \{T, fo, R, \delta\}$ and a range update from R_i to R_{i+1}. If for a subtree of T there is no dependent data source $s \in (R_{i+1} \setminus R_i) \cup (R_i \setminus R_{i+1})$ that produced (historic) events in $[\Delta_s, start(R_{i+1})]$ then the subtree does not require an update of its opererator's states to produce a maximum covering sequence of R_{i+1}.*

PROOF. If the set of dependant sources does not change, the resulting event stream does not change either. □

Moreover, note, the optimal value for Δ_s depends on the time when a range update occurs. For example, the starting point of stream A of the sequence operator varies according to the dependency to the window of stream B. In Figure 4, the starting point is the event, temporally preceding the point in time depicted by the arrow from sw_A to sw_B.

Provisioning of historic events.

The provisioning of historic events from an initialization point on requires to at least buffer the primary event streams. These streams are organized in distributed, historic *event buffers*, that allow historic access to data streams for a certain time Δ_s. The organization and analysis of such a buffer is not focus of the paper, however, many approaches of the literature provide efficient mechanisms [27, 32]. In particular, for the organization of the primary event streams we use a grid-like structure to organize the event storage, where

```
1: upon receive LocationUpdate()
2:   r ← (R_{i+1} \ R_i) ∪ (R_i \ R_{i+1})
3:   s ← findSources(r, [Δ_s, start(R_{i+1})]);
4:   trigger send COLLECTMsg(s, r, start(R_{i+1}) − δ) to root
5: end

6: upon receive COLLECTMsg(Sources S, Range R, Starting-
   Point sp)
7:   if receiver is operator then
8:     sub ← findSubtrees(S)
9:     Δ ← estimateStartingPoint(sp)
10:    trigger send COLLECTMsg(S, R, Δ) to sub
11:  else
12:    for all s ∈ S do
13:      eventSet ← getEvents(s,R,sp)
14:      trigger send Marker(R,start(R),sp) to successors
15:      trigger send eventSet to successors
16:    end for
17:  end if
18: end
```

Figure 5: Basic Framework

events are stored on brokers near their occurrence and events are selected by moving range queries [32]. However, operators can buffer their in- and outgoing streams locally, to reuse these events, e.g., if no marker arrives over the respective incoming stream.

Reconfiguration Algorithm.

The basic reconfiguration algorithm (cf. Figure 5) relies on the observations stated in Lemma 1-2. Whenever a range update occurs, the brokers providing (historic) access to primary event streams in R_{i+1}, determine the set of event streams $(s, \omega) \in T$ for which events σ produced by (s, ω) yield

$$loc(\sigma) \in (R_{i+1} \setminus R_i) \cup (R_i \setminus R_{i+1})$$

and

$$t(\sigma) \in [\Delta_s, \text{start}(R_{i+1})]$$

Note, according to Lemma 1 it suffices to transmit all events with timestamps greater than Δ_s to produce a maximum covering sequence. Recall, however, that the optimal value for Δ_s depends on the state of the operator, thus we further collect this state information from the individual operators.

Therefore, the sources of the determined event streams are forwarded to the root of T, which can determine from the collected set of sources the subtrees of T which require a reconfiguration of the operator tree. The root initiates the initialization point determination for the subtrees by sending a *COLLECT* message to all its successors that are member of a subtree that requires reconfiguration. The *COLLECT* message comprises information on the window states and an initial estimate on an initialization point of the data stream. Each host receiving a *COLLECT* message will refine the time estimate as well as the collected window state information and sends *COLLECT* messages to all successors that are member of a subtree that requires reconfiguration. Finally, a broker of a primary event stream uses the estimates to initialize its data stream by first sending a marker message followed by the event stream.

Figure 6 illustrates the flow of the algorithm. The range update occurs and the brokers detect that only acceleration events are newly selected in R_{i+1}. Therefore, no streams in the subtree rooted at ω_a need to be reconfigured. The COLLECT message is only sent from the root $\omega_\#$ to ω_m collecting the estimate on the starting point. The brokers providing (historic) primary streams receive the starting point and set the marker message, and now provide the newly determined stream.

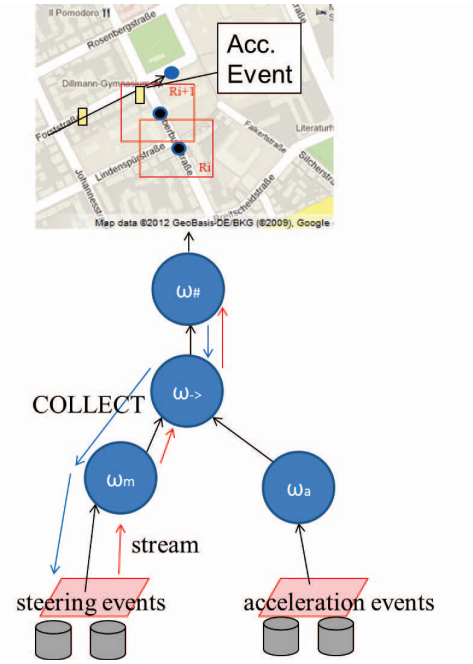


Figure 6: Traffic Scenario

Initialization Point Estimation.

Now we detail how the operators perform the estimation of the initialization points of streams. Consider a pair $(\omega_1, \omega_2) \in T$ where ω_1 needs to send a *COLLECT* message to ω_2. At this time ω_1 has an estimate of the end point $end(rw)$ of its restriction window rw_{ω_1} that will ensure a maximum covering event sequence of T. From $end(rw)$, the state of the selection windows including their dependencies, ω_1 can determine the end point $end(sw_{\omega_2})$ of the selection window sw_{ω_2}. The state information comprised in the *COLLECT* message is $end(sw_{\omega_2})$ and an expression that can be used by ω_2 to find a starting point for sw_{ω_2}. This expression may simply comprise constant time spans $(T : t_{const})$, variable timestamps $(S : t_{var})$ and numbers $(N : n)$[¶] reflecting the different window types supported by the proposed window model. From this information, ω_2 can estimate an endpoint for the restriction window rw_{ω_2}, which provides the estimated endpoint of sw_{ω_2}.

For instance, initially the root of T will receive the expression $((S : start(R_{i+1})\ (T : \delta))$. From this the end point of the restriction window is set to timestamp $start(R) - \delta$, meaning that the initial selection windows comprise timestamps all smaller or equal to $start(R) - \delta$. Therefore, the

[¶]Notice that for each operator can refine such a number by an estimate on how many events are required from its predecessors to produce n events.

first event potentially produced is guaranteed to have a timestamp smaller or equal to $start(R) - \delta$.

5.2 Optimizations

In the following we discuss several possibilities for optimizing the basic reconfiguration algorithm that we integrated in the proposed mobility-aware CEP system.

5.2.1 Incremental Streaming

Note that the maximum covering sequences of two subsequent ranges R_i and R_{i+1} bear a spatial overlap for events produced with respect to $R_{i+1} \bigcup R_i$ as well as a temporal overlap for events produced within $[start(R_{i+1}) - \delta, start(R_{i+1})]$. This can lead to *duplicates*, i.e., the same event is correlated and streamed between operators multiple times. To deal with such overlapping event sequences we distinguish betweeen two classes of operators, *locality preserving operators* and *non-locality preserving operators*. The first class comprises operators that guarantee the following: For any pair of incoming event sequences E and E' which yields $E \subset E'$ all events produced with respect to E are also produced with respect to E'.

For example, an attribute filter is part of that class. On the other hand many aggregation operators are non-locality preserving, i.e., not necessarily all events produced with respect to E are also produced with respect to E'.

Spatial Overlap. Let $\omega \in T$ be a *locality preserving operator*. ω will indicate to all succeeding operators out $(\omega, out) \in T$ that events with location-stamps in $R_{i+1} \bigcup R_i$ will be reused from the incoming buffer until the correlation catches up with the live-data.

Note, that for *non-locality preserving operators* the spatial overlap cannot be used. Consider an operator processes the average number of cars within a time span of 10 min. Further consider the temporal-ordered event-stream $\sigma_1, \sigma_2, \sigma_4$ produced the output event σ_o. If at least one new event σ_3 is integrated during the reconfiguration, i.e., the stream is now $\sigma_1, \sigma_2, \sigma_3, \sigma_4$, the output event σ_o is now changed to σ_0' and therefore σ_o will not be be sent to its successor, although all events that caused o_1 are within the new range R_{i+1}.

Temporal Overlap. To exploit the temporal overlap produced events are also buffered locally at each $\omega \in T$. Each ω verifies for each produced event whether the event has already been sent to its successors by looking it up in its outgoing buffers. If an event was already produced, we log the time and location of this event and postpone the message with this information to the parent operator until a maximum tolerable time span t_s has passed. Within the time span, many more events may be found, so the successors of ω can be notified by sending a log for those events in a single batch.

An operator receiving a log from its predecessor will retrieve these events from its buffer of incoming events to produce the event stream comprising buffered and transmitted events.

5.2.2 Varying the Accuracy of Updates of Ranges

Reducing the accuracy with which a currently deployed range matches the position of the focal point object has additional potential to reduce the reconfiguration overhead. By employing additional update conditions that define an acceptable level of accuracy, the mobile CEP system can decide whether to initiate a reconfiguration.

We distinguish between two update conditions:

1. The *basic update condition* is to reconfigure all operators of the operator tree as soon as the focal point reports a new location. It maintains a high accuracy, as it reflects all location changes, but at the risk of many unnecessary reconfigurations if only a negligible set of event streams need to be linked to T.

2. A *timed or spatial update condition* starts the reconfiguration of all operators of the operator tree, only after some predefined time has passed or the consumer has moved a predefined length away from its old location.

5.3 Analysis

Realizing the Strong Delivery Semantics.

Note that in the reconfiguration algorithms we always used an estimate on the initialization time for the primary event streams. Realizing the strong delivery semantics, however, would need to determine an exact point in time from which to initialize the primary data stream. The current reconfiguration technique needs to be extended, such that at all levels an initialization point is found where the following correlations does not depend on previous correlation steps. For example, we have to make sure that none of the following input events would have been consumed in a previous correlation step.

Comparison to Predeployment.

We analytically compare a predeployment strategy to a mobility driven reconfiguration of operators (for practical evaluation results see also Section 6). Let p be the number of predeployed operator graphs and let q be the number of range queries. Further let the average mobility-rate, i.e., the rate at which the ranges of interest change, be denoted by ν_m. The average number of received events at each level is $\nu_e(l)$ and the average number of correlations at each level of the operator graph is $\nu_c(l)$. The average overhead to update the operator graph C_v is the sum of C_{ev} for event-streams and C_{up} for additional update messages. The operator graph has an average-path length of o. The cost for correlating with an operator graph is therefore $\sum_{l=0}^{o-1} \nu_e(l)\ \nu_c(l)$. In particular, the following inequation must hold for our approach to be better in terms of bandwidth:

$$n\ \textstyle\sum_{l=0}^{o-1}\ \nu_e(l)\ \nu_c(l) > q\ (\sum_{l=0}^{o-1}\ \nu_e(l)\ \nu_c(l) + \nu_m\ C_v) \quad (1)$$

$$\Leftrightarrow (n-q)\ (\textstyle\sum_{l=0}^{o-1}\ \nu_e(l)\ \nu_c(l)) > q\ \nu_m\ C_v \quad (2)$$

Obviously, if the update costs are low and the number of queries is low compared to the number of operator graphs we save bandwidth.

6. EVALUATION

We implemented the mobility-aware CEP (MCEP) system for the Omnet++ simulation environment [28] using SUMO [4] to create realistic mobility traces of cars on a Open StreetMap [13] road map of a subpart of Stuttgart.

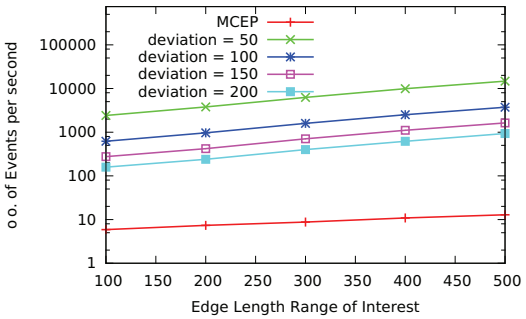


Figure 7: Predeployment vs. MCEP: Streaming

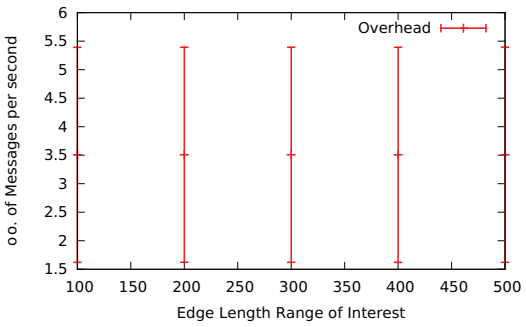


Figure 8: Predeployment vs. MCEP: Overhead

6.1 General setting

We tested our system by evaluating an accident scenario, using an operator tree comparable to the one that we detailed in Section 2. The operators of the tree were randomly placed on the available brokers and we checked on the basis of acceleration and movement patterns, if several cars were involved in an accident.

The scenario was carried out on the road map of a city of roughly 1,5km x 2km size. The brokers were organized geographically as a grid. Each broker was responsible for handling location updates and storing events of consumers and sources residing in a predefined subpart of this city.

The range of interest of consumers was described by rectangular shapes, where the edge length was varying from 100 m up to 500 m.

The simulated city was populated with 1000 cars, which drove from a random location to a random destination. Each car checked each second if its acceleration or steering wheel position changed, and published events with possible changes. Over the course of 1000 simulated seconds this meant that around 250 events per second were published and had to be processed.

6.2 Comparison: Predeployment vs. Reconfiguration

We first highlight the benefits of our approach in comparison with a predeployment approach, where a fixed amount of overlapping ranges and associated operator trees are deployed in advance. For our approach, we associated one car with a *diq*. In this scenario the corresponding operator tree performed the basic reconfiguration with the optimizations described in Section 5.2.

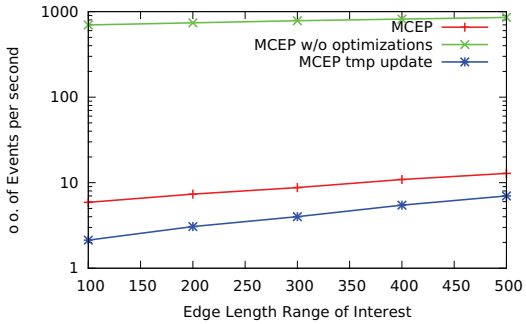


Figure 9: Incremental streaming

The most important distinguishing metrics between these approaches are the *accuracy of the range* of interest and the *number of messages*. The former describes how far the predeployed range deviates from the range of interest. The latter describes the number of streamed events and control messages over all deployed operator trees.

Therefore, we varied two parameters for our simulation. On the one hand the size of the range of interest (edge length) and on the other hand the accuracy in the predeployed range (deviation). Both ranged from a few meters to several hundred meters.

Results.

Figure 7 depicts how many events have to be streamed and processed in the CEP system for different sizes of the range of interest. Figure 8 depicts the overhead in number of control-messages, such as COLLECT messages, that have to be transmitted for the MCEP approach. They show that our approach (MCEP) performs well in contrast to the predeployment approach with different accuracies. The benefit of our approach, even in a small town setting, is that only events of a sub-part of the geographic space, selected by the range of interest, have to be processed. Even the control overhead influences the number of messages only insignificantly, as their number is orders of magnitudes smaller than the number of streamed events.

6.3 Impact of incremental streaming

In this section we highlight the benefits of the optimization of the incremental streaming, where we reduce the number of streamed events by utilizing the spatial and temporal overlap for different operator classes.

The set of experiments varied the size of interest, which shows how the size and potential overlap affects the effectiveness of the event reduction for the accident scenario. Three different scenarios were tested. The reconfiguration was either triggered with each location update, with (MCEP) and without optimizations (MCEP w/o optimizations) or was triggerd after a fixed amount of 10 seconds with optimizations (MCEP tmp update).

Results.

The results, depicted in Figure 9, show that the incremental streaming significantly reduces the amount of events that have to be streamed. This shows that the operators, especially the ones that process primary streams, can reuse many events from previous correlations and thus do not have

to transfer duplicated events. Observe that reconfiguring with each location update also requires more events to be streamed than when we delay the reconfiguration. This is because fewer historic events have to be streamed from an event buffer than in the first case, because the reconfiguration is performed far less often.

7. RELATED WORK

For spontaneous as well as continuous range and moving range queries, a vast amount of approaches has been proposed [19, 7, 10, 27, 12, 8, 5]. The main objective of these approaches is the selection of primary events, e.g., sensor streams, relative to a consumer-specified region in an efficient manner. In contrast, for complex event processing, complex events are the result of multiple dependent operations on primary as well as complex event streams. To account for a meaningful event delivery semantics, i.e., to account for completeness and order of produced event streams, a coordinated reconfiguration of all operators is required.

Some systems like Place* [32] build moving range queries over specific aggregation functions. The approach supports the distributed processing of a single aggregation operator. However, the distributed processing of multiple dependent operators is not addressed by existing work. This way, it becomes possible to apply many optimizations used for distributed CEP systems [20, 22, 24, 29] also in the context of distributed range queries.

Lately, distributed event based systems [9, 26] have proved the benefits for using parametrization of queries to support mobility-aware applications. In [14], a parametrizable publish/subscribe system is presented, which enables location based queries. Yet this work focuses on the reconfiguration of stateless operators like a filter. In order to support stateful operations, our work has shown that it is necessary to determine appropriate initialization points for the chain of dependent operators in order to provide guarantees for the delivery of events.

Note, that recent proposals such as [25] regarding the consistent reconfigurations of stateful operators cannot be applied in a straight forward manner. While [25] provides start and stop semantics for the reconfiguration of operators that operate solely on live events, appropriate initialization points for mobility-aware reconfigurations require to consider historic event streams and again appropriate methods to determine good initialization points.

Many traditional CEP approaches [21, 11, 15, 6, 31, 17] address how to efficiently detect patterns on event streams. However, the operators and sources are statically defined before the query deployment and hence cannot react on location updates. Many different methods for the detection of events have been proposed, such as Petri Nets [11], finite state automata [21], detection trees [6], or disjoint normal form [15]. The introduced partitioned window model allows in contrast to previous work to efficiently determine the temporal and spatial overlap in two subsequent selections and hence especially helps to significantly reduce the reconfiguration overhead of mobility-driven CEP. While many different window semantics were presented for stream-based systems, e.g., [3] or [18] this work can be considered the first work to realize typical CEP operations in a partitioned window model.

8. CONCLUSION

In this paper we have presented a mobility-aware CEP system that delivers complex event streams depending on a dynamically changing interest of a consumer. The system only processes event streams of interest to a consumer and this way avoids the overhead imposed by predeployment of operators. Furthermore, we have introduced a mobility-aware delivery semantics that ensures for a query and each of its ranges to produce a maximum covering sequence. Event streams exhibit a well defined order in which events are delivered to consumers respecting the order of range updates and ensure the preservation of the chronological time order between updates. We have shown that the delivery semantics can be realized cost efficiently by relying on the properties of the proposed event execution model, which partitions incoming event streams in separate selection windows.

While this work adapts a separate operator tree per focal object, in the future we are going to investigate the potential in optimizing the reuse relation between multiple operator trees to further reduce the deployment cost. Furthermore, we did not consider to adapt the placement within the infrastructure, but relied on existing work that aims at optimizing the placement of operators in distributed CEP systems. Additionally, considering the movement patterns of consumers for the placement of operators is an interesting research direction for future work.

Acknowledgement. This work is supported by contract research "CEP in the Large" of the Baden-Württemberg Stiftung. The authors would like to thank S. Schnitzer, A. Benzing, and the reviewers for their helpful comments.

9. REFERENCES

[1] http://www.b30-oberschwaben.de/html/vergleiche.html, February 2012.

[2] M. Akdere, U. Çetintemel, and N. Tatbul. Plan-based complex event detection across distributed sources. *Proc. VLDB Endow.*, 1:66–77, August 2008.

[3] A. Arasu, S. Babu, and J. Widom. The cql continuous query language: semantic foundations and query execution. *The VLDB Journal*, 15:121–142, June 2006.

[4] M. Behrisch, L. Bieker, J. Erdmann, and D. Krajzewicz. Sumo - simulation of urban mobility: An overview. In *SIMUL 2011, The Third International Conference on Advances in System Simulation*, pages 63–68, Barcelona, Spain, October 2011.

[5] A. Benzing, B. Koldehofe, and K. Rothermel. Efficient support for multi-resolution queries in global sensor networks. In *Proceedings of the 5th International Conference on Communication System Software and Middleware*, COMSWARE '11, pages 11:1–11:12, New York, NY, USA, 2011. ACM.

[6] S. Chakravarthy and D. Mishra. Snoop: An expressive event specification language for active databases. *Data & Knowledge Engineering*, 14(1):1 – 26, 1994.

[7] M. Cheema, L. Brankovic, X. Lin, W. Zhang, and W. Wang. Multi-guarded safe zone: An effective technique to monitor moving circular range queries. In *Data Engineering (ICDE), 2010 IEEE 26th International Conference on*, pages 189 –200, march 2010.

[8] T. Do, K. Hua, and C.-S. Lin. Extrange: Continuous moving range queries in mobile peer-to-peer networks. In *Mobile Data Management: Systems, Services and Middleware, 2009. MDM '09. Tenth International Conference on*, pages 317 –322, may 2009.

[9] P. T. Eugster, P. A. Felber, R. Guerraoui, and A.-M. Kermarrec. The many faces of publish/subscribe. *ACM Comput. Surv.*, 35(2):114–131, 2003.

[10] T. Farrell, R. Cheng, and K. Rothermel. Energy-efficient monitoring of mobile objects with uncertainty-aware tolerances. In *Proceedings of the 11th International Database Engineering and Applications Symposium*, pages 129–140, Washington, DC, USA, 2007. IEEE Computer Society.

[11] S. Gatziu and K. Dittrich. Detecting composite events in active database systems using petri nets. In *Research Issues in Data Engineering, 1994. Active Database Systems. Proceedings Fourth International Workshop on*, pages 2 –9, 14-15 1994.

[12] B. Gedik and L. Liu. Mobieyes: A distributed location monitoring service using moving location queries. *IEEE Transactions on Mobile Computing*, 5:1384–1402, 2006.

[13] M. Haklay and P. Weber. Openstreetmap: User-generated street maps. *Pervasive Computing, IEEE*, 7(4):12 –18, oct.-dec. 2008.

[14] K. Jayaram, C. Jayalath, and P. Eugster. Parametric subscriptions for content-based publish/subscribe networks. In I. Gupta and C. Mascolo, editors, *Middleware 2010*, volume 6452 of *Lecture Notes in Computer Science*, pages 128–147. Springer Berlin / Heidelberg, 2010.

[15] G. G. Koch, B. Koldehofe, and K. Rothermel. Cordies: expressive event correlation in distributed systems. In *DEBS '10: Proceedings of the Fourth ACM International Conference on Distributed Event-Based Systems*, pages 26–37, New York, NY, USA, 2010. ACM.

[16] A. Leonhardi and K. Rothermel. A comparison of protocols for updating location information. *Cluster Computing*, 4:355–367, October 2001.

[17] D. C. Luckham. *The Power of Events: An Introduction to Complex Event Processing in Distributed Enterprise Systems*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2001.

[18] K. Patroumpas and T. Sellis. Maintaining consistent results of continuous queries under diverse window specifications. *Inf. Syst.*, 36:42–61, March 2011.

[19] D. Pfoser, C. S. Jensen, and Y. Theodoridis. Novel approaches in query processing for moving object trajectories. In *Proceedings of the 26th International Conference on Very Large Data Bases*, VLDB '00, pages 395–406, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc.

[20] P. Pietzuch, J. Ledlie, J. Shneidman, M. Roussopoulos, M. Welsh, and M. Seltzer. Network-aware operator placement for stream-processing systems. In *Proceedings of the 22nd International Conference on Data Engineering*, ICDE '06, pages 49–, Washington, DC, USA, 2006. IEEE Computer Society.

[21] P. R. Pietzuch, B. Shand, and J. Bacon. Composite event detection as a generic middleware extension. *Network, IEEE*, 18(1):44 – 55, jan/feb 2004.

[22] S. Rizou, F. Dürr, and K. Rothermel. Solving the Multi-operator Placement Problem in Large-Scale Operator Networks. In *Proceedings of the 19th International Conference on Computer Communication Networks*, Zurich, August 2010. IEEE Communications Society.

[23] B. Schilling, B. Koldehofe, U. Pletat, and K. Rothermel. Distributed heterogeneous event processing: enhancing scalability and interoperability of cep in an industrial context. In *DEBS '10: Proceedings of the Fourth ACM International Conference on Distributed Event-Based Systems*, pages 150–159, New York, NY, USA, 2010. ACM.

[24] B. Schilling, B. Koldehofe, and K. Rothermel. Efficient and distributed rule placement in heavy constraint-driven event systems. In *High Performance Computing and Communications (HPCC), 2011 IEEE 13th International Conference on*, pages 355 –364, sept. 2011.

[25] K. Sheykh-Esmaili, T. Sanamrad, P. M. Fischer, and N. Tatbul. Changing Flights in Mid-air: A Model for Safely Modifying Continuous Queries. In *ACM SIGMOD International Conference on Management of Data (SIGMOD'11)*, Athens, Greece, June 2011.

[26] M. A. Tariq, G. G. Koch, B. Koldehofe, I. Khan, and K. Rothermel. Dynamic publish/subscribe to meet subscriber-defined delay and bandwidth constraints. In *Proceedings of the 16th international Euro-Par conference on Parallel processing: Part I*, EuroPar'10, pages 458–470, Berlin, Heidelberg, 2010. Springer-Verlag.

[27] G. Trajcevski, H. Ding, P. Scheuermann, and I. Cruz. Bora: Routing and aggregation for distributed processing of spatio-temporal range queries. In *Mobile Data Management, 2007 International Conference on*, pages 36 –43, may. 2007.

[28] A. Varga. The omnet++ discrete event simulation system. In *Proceedings of the European Simulation Multiconference (ESM'2001)*, June 2001.

[29] M. Völz, B. Koldehofe, and K. Rothermel. Supporting strong reliability for distributed complex event processing systems. In *High Performance Computing and Communications (HPCC), 2011 IEEE 13th International Conference on*, pages 477 –486, sept. 2011.

[30] O. Wolfson, A. P. Sistla, S. Chamberlain, and Y. Yesha. Updating and querying databases that track mobile units. *Distrib. Parallel Databases*, 7:257–387, July 1999.

[31] E. Wu, Y. Diao, and S. Rizvi. High-performance complex event processing over streams. In *Proceedings of the 2006 ACM SIGMOD international conference on Management of data*, SIGMOD '06, pages 407–418, New York, NY, USA, 2006. ACM.

[32] X. Xiong, H. Elmongui, X. Chai, and W. Aref. Place: A distributed spatio-temporal data stream management system for moving objects. In *Mobile Data Management, 2007 International Conference on*, pages 44 –51, May 2007.