Uncertainty-Aware Event Analytics over Distributed Settings: Industry Paper

Nikos Giatrakos
Athena Research & Innovation Center
Technical University of Crete
ngiatrakos@imis.athena-innovation.gr
ngiatrakos@softnet.tuc.gr

Alexander Artikis
University of Piraeus
NCSR Demokritos
a.artikis@unipi.gr
a.artikis@iit.demokritos.gr

Antonios Deligiannakis
Athena Research & Innovation Center
Technical University of Crete
adeli@imis.athena-innovation.gr
adeli@softnet.tuc.gr

Minos Garofalakis
Athena Research & Innovation Center
Technical University of Crete
minos@imis.athena-innovation.gr
minos@softnet.tuc.gr

ABSTRACT

In complex event processing (CEP), simple derived event (SDE) tuples are combined in pattern matching procedures to derive complex events (CEs) of interest. Big Data applications analyze event streams online and extract CEs to support decision making procedures. At massive scale, such applications operate over distributed networks of sites where efficient CEP requires reducing communication as much as possible. Besides, events often encompass various types of uncertainty assigned on event attribute values, occurrence or detection rules. Therefore, massively distributed Big event Data applications in a world of uncertain events call for communication-efficient, uncertainty-aware CEP solutions, which is the focus of this work. As a proof-of-concept, we show how we bridge the gap between two recent CEP prototypes which utilize IBM PROactive Technology ONline as their CEP engine and each extend it towards only one of the dimensions of distribution and uncertainty.

CCS CONCEPTS

• **Information systems → Data streams**; • **Applied computing → Event-driven architectures**; • **Computer systems organization → Distributed architectures**.

KEYWORDS

Complex Event Processing, Distributed Streams, Uncertainty

ACM Reference Format:
Nikos Giatrakos, Alexander Artikis, Antonios Deligiannakis, and Minos Garofalakis. 2019. Uncertainty-Aware Event Analytics over Distributed Settings: Industry Paper. In *DEBS '19: ACM International Conference On Distributed and Event-based Systems, June 24–28, 2019, Darmstadt, Germany.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/1122445.1122456

DEBS '19, June 24–28, 2019, Darmstadt, Germany

ACM ISBN 978-1-4503-9999-9/18/06…$15.00
https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

Complex event processing (CEP) refers to a generic computational paradigm where simple derived event (SDE) tuples are combined in pattern matching procedures so as to derive higher level, complex events (CE) of interest. Consider, for exhibition purposes, a mobile fraud detection application. A rule (pattern) of the form "Report long (lasting more than Y minutes) calls to premium locations" requires two SDEs to occur in sequence. The SDE of the initiation of a call to a premium location should be followed by the SDE of its duration surpassing a certain threshold Y. In case this sequence of SDEs occurs, a full pattern match exists. The corresponding output CE constitutes a high level representation of the business event, which in this particular occasion captures a mobile fraud incident.

Modern Big Data applications analyze event streams in an online fashion and aim at extracting CEs in real-time so as to support critical decision making procedures. In our simple running example, streams refer to ongoing call records and CEs correspond to mobile fraud detection pattern matches. The decision that needs to be made in real-time involves the cut-off of an ongoing call in case it matches a fraud pattern, so that further monetary losses for the telecom provider are avoided.

Big Data applications at massive scale usually operate over distributed, networked architectures. This is because data are not generated within corporate data centers, public or hybrid clouds but instead, event streams originate from a number of geo-distributed sites. Distributed architectures are ubiquitous in Big Data application scenarios ranging from Internet-of-Things (IoT) and wireless sensor networks to smart city, smart energy grid or smart factory settings. In all these scenarios, data gathering and processing devices of any type (wearables, smartphones, sensors on machine particles, antennas) are physically present near the sources of the data streams or constitute such sources. The event generating sites in our example are mobile devices moving in city centers and rural areas, which transmit call related information to one or more communication antenna sites serving these calls as users commute. Moreover, the query source site may lie at a corporate data center regulating mobile fraud issues.

In such settings optimizing the delivery of event analytics requires reducing communication as much as possible. This is because should we let all sites continuously transmit sampled quantities involved in various analytics procedures towards the central query site, the whole network will become overloaded hindering the online, real-time delivery of detected CEs. Moreover, for battery-powered sensors communication is a major cause of energy drain.

Additionally, Big event Data often encompass different levels of uncertainty. As past works have pointed out [5], uncertainty may come on the event attribute values, propagate on the event occurrence or the rule that defines the pattern to be checked for matching may be applied with a certain level of confidence. In our mobile fraud scenario, uncertainty on event attribute may come from attaching a confidence (probability) in characterizing a call destination as premium, which inevitably propagates on the corresponding CE occurrence. Furthermore, domain experts, based on prior knowledge, may have defined that the CEs detected by the fraud pattern truly indicate a mobile fraud incident only at a percentage of the observed cases. The latter is a probability value attached on the rule. One can add another source of uncertainty that stems from the need of forecasting events with a certain probability before they occur, instead of reporting their appearance [4], or from performing approximate CEP with quality guarantees [26].

Therefore, what is required in massively distributed Big event Data applications operating in a world of uncertain events are solutions that can provide communication-efficient, uncertainty-aware CEP. This is the focus of this paper. In particular, given an uncertainty-aware event analytics query posed over a network of sites, the techniques we propose construct in-situ filters for each site. Our in-situ filters suppress communication among sites by being constructed in a way so that, if their conditions are not satisfied, a CE **cannot** have occurred even upon combining event data from the various sites. Events whose communication is suppressed are initially cached at sites but, if they continue to be suppressed as time passes, they expire due to window constraints. Thus, their communication may be avoided altogether.

Our approach is backed by probability theory and communication protocols. Past research efforts and prototypes have focused on one of these aspects, either reducing communication during CEP procedures over distributed settings [12, 13] or synthesizing probabilities of SDEs so as to produce an overall probability of the output CE (for instance, see [9, 32]). Hence, to our knowledge, our techniques are the first that elaborate on communication-efficient, uncertainty-aware event analytics over distributed settings.

The contributions of this work are summarized below:

- We propose, to our knowledge for the first time in CEP, uncertainty-aware in-situ filters which are installed on the sites of a distributed setting and suppress unnecessary communication of uncertain events during the execution of CEP queries. The filters achieve that by being constructed in such a way so that, if they are not violated by at least one site, a global CE over the network of sites cannot have occurred even upon combining event data from all of them. Thus, communication can be safely avoided.
- We propose a novel monitoring protocol for executing uncertainty-aware CEP over distributed settings, composed of a number of sites, in a communication-efficient manner. Our protocol fully exploits the installation of the constructed in-situ filters on sites.
- As a proof-of-concept of the applicability of our techniques in real systems, we show how we bridge the gap between two recent prototypes. The FERARI streaming multi-cloud platform [12, 13], which provides communication-efficient event analytics over distributed settings but neglects uncertainty aspects, and the work by Correia et al [7] which elaborates on handling event uncertainty but requires continuous event transmissions, not accounting for efficiency over distributed settings. Both prototypes utilize IBM PROactive Technology ONline (Proton) [1] as their CEP engine and each extend it towards the respective aspect. To illustrate the proposed approach in real world scenarios, we use a mobile fraud detection use case from the telecom domain.

2 OVERVIEW & MOTIVATION

The following motivating example better exhibits our contributions and helps us provide an overview of our approach. Consider a more intriguing, compared to our introductory example, query of the form: "Trigger an alarm when the number (count) of calls to premium locations made by a specific caller during the past Y minutes is above T with probability higher than C". Note that each call of a *callerID* can be handled by a different antenna site and thus, the total call count for that *callerID* may be spread over the network (Figure 1). Hence, one needs to aggregate respective data for each *callerID* at a central site to answer the query, which is daunting from a communication perspective. Assume, for exhibition purposes, that each international call destination is attributed a confidence value (probability) p of being premium. This confidence value may be set by a domain expert or get derived from past data [6, 28]. Therefore, each call to a potentially premium location is a Bernoulli random variable. The total number of calls (Bernoulli trials) made by a *callerID* is the sum of the number of calls that have been handled by all antennas $\{A_1, \ldots, A_N\}$: $\sum_{i=1}^{N} n_i = n$, with n_i being the local count of calls for *callerID* at antenna A_i. Let X denote the random variable representing the count of such calls to potentially premium locations throughout the network of antennas. Then X follows a Binomial distribution, i.e., $X \sim B(n, p)$, where n is the total number of such calls made by *callerID* the last Y minutes and p the probability of a call destination being premium.

The approach in FERARI [12, 13] will attempt to avoid continuous central data collection by constructing in-situ filters that are not uncertainty-aware, in the sense that they consider all potentially premium locations as surely premium. Assuming a city with N antennas, an in-situ filter for each site will be: each antenna A_i suppresses communication if the local count X_i of calls of *callerID* is at most $\frac{T}{N}$, where T is the threshold on the call count in the given query. This is because if $X_i \leq \frac{T}{N}$ at all antenna sites, then globally $X = \sum_{i=1}^{N} X_i \leq \sum_{i=1}^{N} \frac{T}{N} = T$. Thus, if an antenna A_i finds $X_i > \frac{T}{N}$ for calls within the last Y minutes, it will transmit this data to the query site, because only then it is possible that X has exceeded the T threshold. Afterwards, the central site will prompt the rest of the $N-1$ antennas to communicate the call count for that *callerID* within the last Y minutes. But the "with probability higher than C" criterion is not applied anywhere in the in-situ filters. Thus, instead of narrowing down transmitted SDEs based on two criteria, i.e., count of calls to premium locations and the confidence on

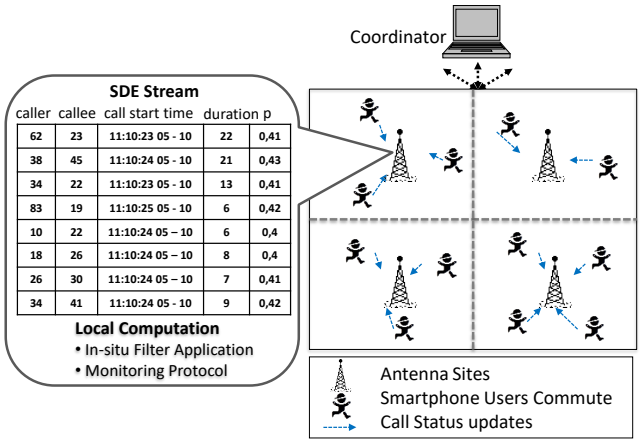


Figure 1: An instance of site network organization into a bottom tier (telecom antennas) and a top tier (coordinator - query site). Each site applies the in-situ filter and executes the monitoring protocol proposed by our approach.

these locations being premium, only one such criterion is used. This increases potentially unnecessary transmissions.

On the other hand, should one simply use the extension of Proton provided by Correia et al [7], she will need to accumulate all data to the query site before being able to check whether the uncertainty criterion applies and whether events can be pruned because of it.

Our contribution comes exactly because of the ability of our in-situ filters to account for both the distribution and uncertainty dimensions of the posed query. In particular, our in-situ filters will recognize the fact that if globally $X \sim B(n,p)$, then for each antenna $A_i, X_i \sim B(n_i,p)$. Our techniques exploit probability theory to recognize that, for common p, the binomial distribution is self-decomposable, i.e., if $X_i \sim B(n_i,p) \rightarrow X = \sum_{i=1}^{N} X_i \sim B(n,p)$. Therefore, the in-situ filter constructed by our approach will be (as explained in Section 4.3): each A_i suppresses communication if the probability of the local count X_i of calls of *callerID* is at most $\frac{T}{N}$ with probability above $\sqrt[N]{1-C}$: $Pr[X_i \leq \frac{T}{N}] \geq \sqrt[N]{1-C}$, where C is the (un)certainty threshold in the posed query. The latter filter accounts for both the criteria (call count, confidence) included in the posed query. Further details on the generic approach for in-situ filter construction, decomposable distributions and communication protocol operation follow throughout our study.

3 PRELIMINARIES

Network of Sites. We assume a distributed, two-tiered setting composed of N sites $\{A_1, \ldots, A_N\}$ at the bottom tier and a query source site (top-tier), which we term as the coordinator site. The coordinator poses a query involving event data that are gathered across the N sites and engages the CEP operators outlined below. An instance of this architecture, for the motivating example of Section 2, is shown in Figure 1.

Target Queries. Our algorithms engage popular operator categories [11]. All operators that are admissible in our setup bear a window W, which expresses the maximum time window within which the input events of an operator should appear. This operator list includes:

- NON_AGGRegation operators with NON_AGGR $\in$ {AND, OR, SEQ}:
  - AND outputs a CE when all participating SDEs occur in W.
  - SEQ outputs a CE when all participating SDEs occur in specified sequence in W.
  - OR outputs a CE whenever any participating SDE occurs in W.
- AGGRegation operators, AGGR $\in$ {AVERAGE, COUNT, SUM}.

Note that the naming may differ from one CEP engine to another. For instance, IBM Proton employed in our proof-of-concept uses ALL instead of AND, supports OR via ALL and ABSENCE (logical negation) operators as well as uses SEQUENCE rather than SEQ.

In this work we target thresholded CEP queries on both the aggregation and uncertainty values. Generalizing the rationale of our motivating example in an SQL-like syntax ($m \geq 1$, $0 < C \leq 1$):

```
PATTERN NON_AGGR(AGGR_1 ≶ T_1, ..., AGGR_m ≶ T_m) Q
[WHERE list_conditions ]
[PARTITION BY partition_key ]
HAVING Q.certainty>C
WITHIN window_constraints
```

where $\lessgtr \in \{>, <, \leq, \geq\}$. Compartments embraced in [] are optional. The WHERE clause refers to filter conditions on individual attributes or predicates, the PARTITION BY clause partitions the incoming events based on some key attribute, WITHIN expresses the desired window constraints and HAVING accounts for our uncertainty criterion. Instantiating the above for the example of Section 2 gives:

```
PATTERN (COUNT(Call_Detail_Record CDR) ≥ T) Q
WHERE CDR.destination ∈ possibly_premium_locations
PARTITION BY CDR.callerID
HAVING Q.certainty>C
WITHIN Y minutes
```

where no NON_AGGR operator is needed.

Uncertainty Assumptions. As in the vast majority of related works [5], we employ an event independence assumption and also assume that sites handle independent streams. Other common practices, such as employing a Markovian or Bayesian [5] hypothesis for event and stream dependence, are left for future work.

4 OUR APPROACH

4.1 Decomposition to Individual Aggregations

Given the independence assumption, for NON_AGGR $\in$ {AND, SEQ}, Q.certainty>C holds if the same uncertainty criterion is satisfied in each $\text{AGGR}_j \lessgtr \text{T}_j$, $j \in \{1, \ldots, m\}$ engaged in the query Q. This is because the certainty of AND, SEQ results as a product of the certainties of $\text{AGGR}_j \lessgtr \text{T}_j$. Thus, these operators can output a CE when the certainty of each $\text{AGGR}_j \lessgtr \text{T}_j$ is higher than C. For an OR operator, we require at least one $\text{AGGR}_j \lessgtr \text{T}_j$ to have a certainty value higher than C for the rule to output a CE. Therefore, the initial query can be decomposed to individual sub-queries:

```
PATTERN (AGGR_j ≶ T_j) Q_j
[WHERE list_conditions ]
[PARTITION BY partition_key ]
HAVING Q_j.certainty>C
WITHIN window_constraints
```

Based on the above discussion, the uncertainty criterion C is checked for every sub-query and, thus, we henceforth focus on constructing in-situ filters for each $\mathtt{Q_j}$. Note that, as in the example of Section 2, the variable X describing $\mathtt{AGGR_j}$ follows a probability distribution $f(.)$, i.e., $X \sim f(.)$. The parameters that are included in the parenthesis may differ from one probability distribution to another. $f(.)$ becomes known to the coordinator, obtained either theoretically (as in the example of Section 2) or empirically based on gathered data samples as described in Section 4.4.

4.2 Decomposable Probability Distributions

Our techniques are applicable when X can be decomposed to individual $X_i \sim f(.)$ for each site A_i, so that either $X = \sum_{i=1}^{N} X_i$ or $X = \prod_{i=1}^{N} X_i$. Linear or weighted combinations of X_is are also supported. This property holds for large families of distributions. Table 1 summarizes some popular, supported distributions.

Both discrete and continuous probability distributions are supported by our approach. Discrete distributions, include Binomial, Poisson (shown in the table) and Geometric together with the Negative Binomial or Poisson Binomial distributions (not shown in the table). Continuous cases involve the Normal, Gamma, Exponential, Cauchy, Chi-square and Logistic (approximated via Normal). Furthermore, logarithmic versions of supported distributions include, among others, the popular cases of Log-Normal and Log-Logistic. For the log-versioned distributions $X = \prod_{i=1}^{N} X_i$ holds, instead of $X = \sum_{i=1}^{N} X_i$, which holds in the rest of the provided examples.

With respect to the telecom domain employed in our motivating example (Section 2) and our proof-of-concept (Section 5), besides the Binomial distribution, Exponential [20], LogNormal and LogLogistic distribution [10] can be used to model (abnormal) time intervals between calls of a particular caller, handled by different antennas in the network. Moreover, the Poisson distribution can be applied in modeling suspicious traffic [17], for instance, caused in the network by numbers of a certain subscriber. Finally, in our proof-of-concept we exploit the Normal distribution to assess potentially fraudulent calls when only the charged, instead of the actual, duration of calls are known to data analysts due to caller privacy constraints.

The first three columns of Table 1 include the name of the distribution, its Probability Density Function (PDF) and an explanation of its parameters, respectively. The last column of Table 1 refers to our proof-of-concept (Section 5), highlighting which of the cited distributions are present in the uncertain extension of IBM Proton [7]. We believe that as uncertainty-aware CEP gets more and more attention, such functionality will be provided by other engines as well. Note that the fourth column of Table 1 just presents examples of plausible decompositions. Other ways of breaking up X to X_is may also be admissible. For instance, the work of Moschopoulos [27] can serve as a generic tool for decomposing the Gamma distribution even when the scale parameters of X_is differ. We will comment of the fifth column of Table 1 shortly.

Finally, note that $X_i \sim f(.)$ allows the probability distribution to incorporate different parameters in $f(.)$ compared to $X \sim f(.)$. A study on self-decomposable and infinitely divisible distributions, which admit our in-situ filters (Section 4.3) and can be supported by our approach, is included in the book by Steutel and Harn [31]. We will shortly discuss how to encounter situations where $f(.)$ may differ among the sites.

4.3 In-situ Filter Construction

Let us now examine how in-situ filters are constructed when X can be expressed either as a sum or a product of X_is. The probabilistic criterion in our query Q_j can be written as $Pr[X \lesseqgtr T_j] > C$, thus, no CE can be outputted if this probability is lower or equal to C.

Definition 4.1. (**Global Filter**) The global filter is the condition:

$$Pr[X \lesseqgtr T_j] \leq C \tag{1}$$

where $\lesseqgtr \in \{>, <, \leq, \geq\}$, such that no CEs can be outputted by the evaluation of Q_j even upon combining event data from $\{A_1, \ldots, A_N\}$.

This is the global filter which we break into in-situ filters for each site A_i. Lemma 4.2 explains how this is performed. In Lemma 4.2 notice that Inequalities 2,3 use $X_i \gtreqless \frac{T_j}{N}$ and $X_i \gtreqless \sqrt[N]{T_j}$, respectively, while the global filter in Inequality 1 says $X \lesseqgtr T_j$. In particular, the direction and the inequality used in Inequality 1 is complementary to that of Inequalities 2,3. Recall that $\gtreqless, \lesseqgtr \in \{>, <, \leq, \geq\}$. Thus, if $\geq$ or $>$ is present in Inequality 1, the in-situ filter in Inequalities 2,3 should use $<$ or $\leq$ respectively, and vice versa.

Lemma 4.2. (**In-situ Filters**)

(i) When $X = \sum_{i=1}^{N} X_i$, the in-situ filter that should be locally installed at each site $A_i \in \{A_1, \ldots, A_N\}$ is given by:

$$Pr[X_i \gtreqless \frac{T_j}{N}] \geq \sqrt[N]{1 - C},\ \forall A_i \in \{A_1, \ldots, A_N\} \tag{2}$$

(ii) When $X = \prod_{i=1}^{N} X_i$, for positive T_j and random variables (see Table 1 for log-versioned distributions), the in-situ filter that should be locally installed at each site $A_i \in \{A_1, \ldots, A_N\}$ is given by:

$$Pr[X_i \gtreqless \sqrt[N]{T_j}] \geq \sqrt[N]{1 - C},\ \forall A_i \in \{A_1, \ldots, A_N\} \tag{3}$$

Proof. (i) Analyzing $\geq$ from the set $\{>, <, \leq, \geq\}$ suffices as the rest of the cases are analogous. Using $\geq$, Inequality 1 turns to $Pr[X \geq T_j] \leq C$, while Inequality 2 turns to $Pr[X_i < \frac{T_j}{N}] \geq \sqrt[N]{1 - C}$. Thus:

$$(2) \Rightarrow \prod_{i=1}^{N} Pr[X_i < \frac{T_j}{N}] = Pr[\nexists X_i \geq \frac{T_j}{N}] \geq 1 - C \Leftrightarrow$$

$$Pr[\exists X_i \geq \frac{T_j}{N}] \leq C \Rightarrow C \geq Pr[\exists X_i \geq \frac{T_j}{N}] \geq$$

$$Pr[\sum_{i=1}^{N} X_i \geq T_j] = Pr[X \geq T_j]$$

Hence, Inequality 2 $\Rightarrow$ Inequality 1. No CE can be outputted by Q_j and communication can safely be avoided by all sites.

(ii) Again for the case of $\lesseqgtr$ corresponding to $\geq$:

$$(3) \Rightarrow \prod_{i=1}^{N} Pr[X_i < \sqrt[N]{T_j}] = Pr[\nexists X_i \geq \sqrt[N]{T_j}] \geq 1 - C \Rightarrow$$

$$C \geq Pr[\exists X_i \geq \sqrt[N]{T_j}] \geq Pr[\prod_{i=1}^{N} X_i \geq T_j] = Pr[X \geq T_j]$$

□

Distribution	PDF	Remarks	Decomposition Example	In-situ Filter for $1 - CDF(X, T) > C$	CEP Engine
Normal	$\frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$,	$\forall x \in \mathbb{R}$	$X_i \sim Normal(\mu_i, \sigma_i^2)$ $X = \sum_{i=1}^{N} X_i \sim Normal(\sum_{i=1}^{N} \mu_i, \sum_{i=1}^{N} \sigma_i^2)$	$\sqrt[N]{1-C} \leq CDF(X_i, \frac{T}{N})$	✔
Log-Normal	$\frac{1}{x\sigma\sqrt{2\pi}} e^{-\frac{(\ln x-\mu)^2}{2\sigma^2}}$,	$\forall x > 0$ $\mu \in \mathbb{R}$(≠mean) $\sigma > 0$ (≠ st.dev.)	$X_i \sim LogNormal(\mu_i, \sigma_i^2)$ $X = \prod_{i=1}^{N} X_i \sim LogNormal(\sum_{i=1}^{N} \mu_i, \sum_{i=1}^{N} \sigma_i^2)$	$\sqrt[N]{1-C} \leq CDF(X_i, \sqrt[N]{T})$	✔
Chi-Square	$\frac{1}{2^{\nu/2}\Gamma\left(\frac{\nu}{2}\right)} x^{\frac{\nu}{2}-1} e^{-\frac{x}{2}}$	$\forall x > 0$ $\nu \in \mathbb{N}^+$degrees of freedom	$X_i \sim x^2(\nu_i)$ $X = \sum_{i=1}^{N} X_i \sim x^2(\sum_{i=1}^{N} \nu_i)$	$\sqrt[N]{1-C} \leq CDF(X_i, \frac{T}{N})$	✘
Cauchy	$\frac{1}{\pi s\left[1+(\frac{x-\nu}{s})^2\right]}$	$\forall x \in \mathbb{R}$ $\nu \in \mathbb{R}$(location) $s > 0$ (scale)	$X_i \sim Cauchy(\nu_i, s_i)$ $X = \sum_{i=1}^{N} X_i \sim Cauchy(\sum_{i=1}^{N} \nu_i, \sum_{i=1}^{N} s_i)$	$\sqrt[N]{1-C} \leq CDF(X_i, \frac{T}{N})$	✘
Poisson	$\frac{\lambda^x e^{-\lambda}}{x!}$	$\forall x \in \mathbb{N}$ $\lambda > 0$	$X_i \sim Poisson(\lambda_i)$ $X = \sum_{i=1}^{N} X_i \sim Poisson(\sum_{i=1}^{N} \lambda_i)$	$\sqrt[N]{1-C} \leq CDF(X_i, \frac{T}{N})$	✘
Gamma	$\frac{1}{\Gamma(\alpha)\theta^\alpha} x^{\alpha-1} e^{-\frac{x}{\alpha}}$	$\forall x > 0$ $\alpha > 0$(shape) $\theta > 0$ (scale)	$X_i \sim Gamma(\alpha_i, \theta)$ $X = \sum_{i=1}^{N} X_i \sim Gamma(\sum_{i=1}^{N} \alpha_i, \theta)$	$\sqrt[N]{1-C} \leq CDF(X_i, \frac{T}{N})$	✔
Logistic	$\frac{e^{-\frac{x-\nu}{s}}}{s\left(1+e^{-\frac{x-\nu}{s}}\right)^2}$	$\forall x \in \mathbb{R}$ $\nu \in \mathbb{R}$(location) $s > 0$ (scale)	$X_i \sim Logistic(\nu_i, s_i)$ (approx.) $X = \sum_{i=1}^{N} X_i \sim Logistic(\sum_{i=1}^{N} \nu_i, \sqrt{\sum_{i=1}^{N} s_i^2})$	$\sqrt[N]{1-C} \leq CDF(X_i, \frac{T}{N})$	✔
Log-Logistic	$\frac{(\beta/\alpha)(x/\alpha)^{\beta-1}}{\left(1+(x/\alpha)^\beta\right)^2}$	$\forall x > 0$ $\alpha > 0$(scale) $\beta > 0$ (shape) $\nu = log(\alpha)$ $s = 1/\beta$	$X_i \sim LogLogistic(\nu_i, s_i)$ (approx.) $X = \prod_{i=1}^{N} X_i \sim LogLogistic(\sum_{i=1}^{N} \nu_i, \sqrt{\sum_{i=1}^{N} s_i^2})$	$\sqrt[N]{1-C} \leq CDF(X_i, \sqrt[N]{T})$	✘
Exponential	$\lambda e^{-\lambda x}$	$\forall x > 0$ $\lambda > 0$ (rate)	$X_i \sim Gamma(\frac{\alpha_i}{N}, \frac{1}{\lambda})$ $\sum_{i=1}^{N} \alpha_i$ $X = \sum_{i=1}^{N} X_i \sim Exp(\lambda)$	$\sqrt[N]{1-C} \leq CDF(X_i, \frac{T}{N})$	✔
Binomial	$\binom{n}{x} p^x (1-p)^{n-x}$	$x = 0, 1, \ldots, n$ $p \in [0, 1]$ $n \in \mathbb{N}$	$X_i \sim Binomial(n_i, p)$ $X = \sum_{i=1}^{N} X_i \sim Binomial(\sum_{i=1}^{N} n_i, p)$	$\sqrt[N]{1-C} \leq CDF(X_i, \frac{T}{N})$	✔

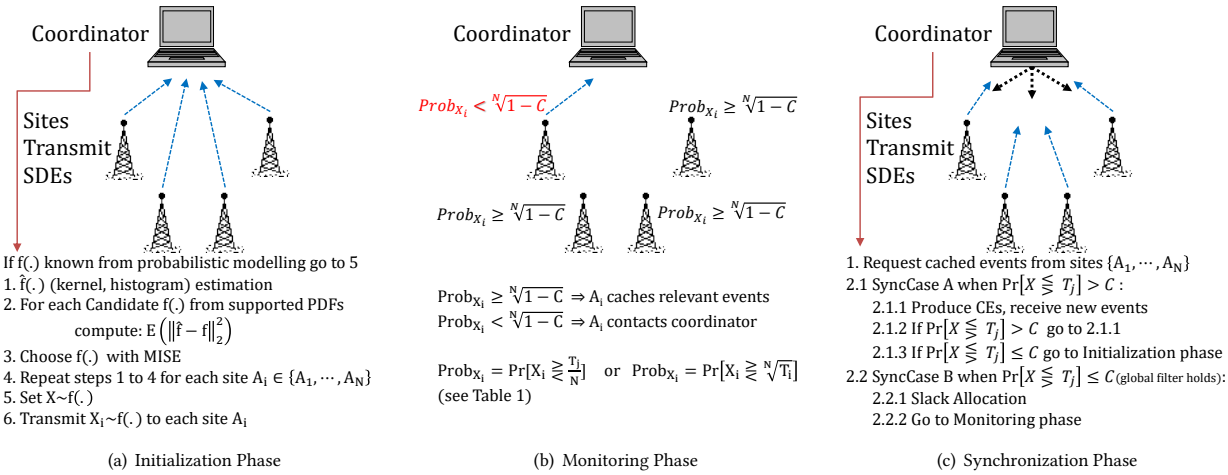
Table 1: Some supported probability distributions, uncertainty criteria and respective in-situ filter examples. PDF: Probability Density Function, CDF: Cumulative Distribution Function. For log-versioned distributions $X = \prod_{i=1}^{N} X_i$ instead of $X = \sum_{i=1}^{N} X_i$ holds due to the use of logarithms. $1 - CDF(X, T) > C \Leftrightarrow P[X > T] > C$ in the fifth column exemplifies the query uncertainty criterion, which corresponds to the global filter: $P[X > T] \leq C$ based on Inequality 1. The column then shows the instantiations of Inequality 2 or Inequality 3 per distribution. The last column indicates which of these distributions are supported by the CEP Engines used in our Proof-of-Concept.

Inequality 2 and Inequality 3 are the in-situ filters we construct based on how X can be decomposed to individual X_is, one for each site A_i. The fifth column of Table 1 illustrates further application examples of Inequality 2 and Inequality 3 for the cited distributions. When the query poses an uncertainty criterion: $1 - CDF(X, T_j) > C \Leftrightarrow P[X > T_j] > C$, it interprets to the global filter $P[X > T_j] \leq C$ (Inequality 1). CDF stands for Cumulative Distribution Function.

Handling Variations in Local Probability Distributions. So far, we have assumed that $X \sim f(.)$ and each $X_i \sim f(.)$ follow the same probability distribution f (in Table 1 the Exponential distribution is an exception to that observation) although various parameters, means and variances among $X \sim f(.)$ and $X_i \sim f(.)$ can differ. The natural question that arises regards what if the PDF of X_i differs for each site and from that of X, i.e. for some (one or more) site A_k, say $X_k \sim g_k(.)$ where g_k differs from f.

It is important to note that X comes from combining data from the various sites $\{A_1, \ldots, A_N\}$. Therefore, if $X \sim f(.)$, it is impossible for X_is to follow arbitrary probability distributions. It is only a very narrow subset of PDFs that are allowed for X_is so that their synthesis in the form of either $X = \sum_{i=1}^{N} X_i$ or $X = \prod_{i=1}^{N} X_i$ yields $X \sim f(.)$. Given this, even if there exists $X_k \sim g_k(.)$, there should be a variable transformation to turn $X_k \sim g_k(.)$ into $X'_k \sim f(.)$ and then X'_k should be used in the respective sum or product for X.

Popular transformations include Linear transformations such as multiplying by, dividing by or adding a constant to X_i and non-linear transformations such as Logarithmic, Square root, Power, Inverse, Reciprocal, Cube Root and Exponential transformation. Please refer to Kutner et al [22] and Leemis and McQueston [24] for more details on transforming one probability distribution to another. For instance, applying a square root-like transformation to a Chi-square distributed random variable provides a Normal approximation [19]. Both these distributions are supported by our techniques as shown in Table 1. Thus, if $X \sim Normal(\mu, \sigma^2)$ but X_k is initially found (see Section 4.4) to follow a Chi-square distribution, then applying such a transformation on X_k produces $X'_k \sim Normal(\mu_k, \sigma_k^2)$ and X'_k should be used in producing X in an additive form (see first row of Table 1).



(a) Initialization Phase (b) Monitoring Phase (c) Synchronization Phase

Figure 2: Distributed Monitoring Protocol Operation

4.4 Distributed Monitoring Protocol

Our proposed framework divides its operation into three phases, namely (a) Initialization, (b) Monitoring and (c) Synchronization phases, as described below.

1. Initialization Phase: A central event data collection takes place at the coordinator after the reception of the posed query, gathering events from sites for a predefined time interval. These will be used so that the distribution which X follows is estimated, if it cannot be imposed by the probabilistic model (e.g., as in Section 2). Hence, the coordinator applies a density estimation method (such as kernel density estimation, histograms) to obtain an empirical estimation of the PDF of X. Then, using the set of supported decomposable distributions, some of which are presented in Table 1, the coordinator computes the Mean Integrated Squared Error (MISE) for each: $E(\|\hat{f} - f\|_2^2)$ where $E(.)$ denotes the expected value, $\hat{f}$ is the estimated PDF and f the candidate one from the set of decomposable distributions. It then picks the one with the minimum MISE and sets $X \sim f(.)$. If the global filter in Inequality 1 holds, the coordinator follows the same density estimation process for each site A_i and computes $X_i \sim f(.)$ as well. Then, the PDF of each X_i is transmitted to the respective A_is and the network proceeds to the Monitoring phase (outlined below). Otherwise, if the global filter does not hold, central data collection should keep taking place as CEs are indeed produced. We emphasize that, if the local PDF of X_i differs from the one of X, the discussion about handling variations in local probability distributions (see end of Section 4.3) applies and possible variable transformations take place. Furthermore, the chosen PDF $f(.)$, used for both X and X_is, is allowed to change to another supported distribution only after a Synchronization phase (also outlined below). However, X_is may update the parameters, such as mean and variance, of their own PDF $f(.)$ as more data arrive locally at A_is.

2. Monitoring Phase: Each A_i receives timestamped uncertain event tuples. A_i computes any alterations on the parameters of the PDF of X_i using only local data that are included in the current window and satisfy conditions of the WHERE clause. Given this, A_i checks if the in-situ filters of Inequality 2 or Inequality 3, depending on the nature of the decomposition as described in Section 4.3, are satisfied. In case the in-situ filters are satisfied in all the sites, no communication takes place and each site A_i caches its relevant events. If the local filter is violated in at least one site A_i, a synchronization phase takes place. Otherwise cached events expire as the window slides.

3. Synchronization Phase: A central data collection takes place at the coordinator. This involves all arriving or cached event tuples that are included in the current window and satisfy the conditions of the WHERE clause. We distinguish the following two cases:

- *Sync Case A*: If Inequality 1 does not hold and, thus, CEs are indeed produced, continuous communication of events observed at $\{A_1, \ldots, A_N\}$ takes place. Event tuples are used at the coordinator to produce CEs and to update the parameters of the PDF of X, if needed. Inequality 1 is checked again with the reception of new tuples. In case at some point Inequality 1 holds true, the coordinator enters the Initialization phase.
- *Sync Case B*: If Inequality 1 holds true, the central event collection took place in vain, since no CEs are produced. We call such synchronizations as False Positive (FP) ones. However, if no CE is produced, the data each A_i communicated have not being used in pattern matches. If the coordinator does nothing and simply tells the sites to switch to the Monitoring phase again, another FP synchronization will be very likely. For instance, if no event tuple arrives or expires in the sites which caused the previous synchronization, their in-situ filters will be violated as soon as they re-enter the Monitoring phase. A situation with continuous, most likely FP, synchronizations should be avoided, if possible. In order to achieve that, the coordinator performs a *slack allocation* effort which is outlined below. If slack allocation is successful, the protocol returns to the Monitoring phase with new in-situ filters. Otherwise, the protocol simply switches to the Monitoring phase.

Slack Allocation: During slack allocation, the coordinator attempts to decrease the uncertainty threshold in Inequality 2 or

Inequality 3 for the violating sites, so that they are less likely to be violated immediately after switching to the Monitoring phase. For the non-violating sites the corresponding threshold is increased. To illustrate how this is performed, for ease of exposition, let us denote with $Prob_{X_i}$ either $Pr[X_i \gtrless \frac{T_j}{N}]$ or $Pr[X_i \gtrless \sqrt[N]{T_j}]$ depending on how decomposition is achieved for X. Slack allocation works when $\prod_{i=1}^{N} Prob_{X_i} > 1 - C$. This is because there is the requirement that $\prod_{i=1}^{N} Prob_{X_i}$ does not fall below $1 - C$, as explained in the proofs of Lemma 4.2(i) and Lemma 4.2(ii). Given this and the fact that the protocol is in Sync Case B, the coordinator computes $\frac{1-C}{\prod_{i=1}^{N} Prob_{X_i}} = \Delta < 1$. This is a ratio expressing the quantity (slack) remaining for $\prod_{i=1}^{N} Prob_{X_i}$ to approach $1 - C$. To avoid short-term FP synchronizations, the coordinator distributes Δ among the, say $K < N$, sites whose in-situ filter was violated during the current synchronization. Therefore, the in-situ filter for those sites is loosened to $\sqrt[N]{1-C} \cdot \sqrt[K]{\Delta}$, while for the remaining sites the in-situ filter becomes stricter: $\sqrt[N]{1-C} \cdot \frac{1}{\sqrt[N-K]{\Delta}}$. Notice that, by adjusting the uncertainty thresholds as above, $\prod_{i=1}^{N} Prob_{X_i} \geq 1 - C$ still holds, provided the new in-situ filters hold in all sites. Hence, if slack allocation leads to a situation where all new in-situ filters hold, the protocol switches to the Monitoring phase by a corresponding message from the coordinator to sites including the new in-situ filter for each. The slacks $\sqrt[K]{\Delta}$ and $\frac{1}{\sqrt[N-K]{\Delta}}$ remain fixed until the next synchronization. Otherwise, a Monitoring phase with no new slack allocation (i.e., using the usual filters) takes place.

Example 4.3. Assume a set up of $N = 3$ sites with $C = 0.8$ and $K = 1$. Thus, $1-C = 0.2$, $\sqrt[N]{1-C} = 0.58$. In particular, consider only A_1 violates its in-situ filter with $Prob_{X_1} = 0.5 < \sqrt[N]{1-C} = 0.58$, while for the other two sites $Prob_{X_2} = 0.9$, $Prob_{X_3} = 0.9$. These yield $\prod_{i=1}^{N} Prob_{X_i} = 0.405 > 0.2 = 1 - C$ and therefore $\Delta = 0.49$. Based on this, the decreased threshold for A_1 will be $\sqrt[N]{1-C} \cdot \Delta = 0.28$, while the increased thresholds for A_2, A_3 will be set to $\sqrt[N]{1-C} \cdot \frac{1}{\sqrt{\Delta}} = 0.82$. At the time the slack allocation is performed all the three thresholds are satisfied by $Prob_{X_1}, Prob_{X_2}, Prob_{X_3}$ and slack allocation leaves enough "space" for the probabilities to change (upon switching to the Monitoring phase) and still avoid a violation of the new in-situ filters.

5 PROOF-OF-CONCEPT

As a proof-of-concept of the applicability of our approach on real CEP engines, we comment on how our techniques bridge the gap between two recent prototypes that extend the same CEP engine, that is IBM Proton [1], one towards uncertainty handling [7], while the other [12, 13] towards distributed settings.

5.1 Proton and Uncertainty Handling

In our discussion so far, for ease of presentation, we used CEP query formulations that resemble SQL syntax. Individual CEP engines may adopt query languages that deviate from such syntax and still support our techniques. For instance, Proton, abiding by the concepts discussed by Etzion and Niblett [11], organizes CEP in an Event Processing Network (EPN) which is composed of Event Processing Agents (EPAs). With respect to our previous discussion, one may think of EPAs as CEP operators linked together when one EPA provides input to another. Since a detailed description of the main concepts of Proton is included in the work of Correia et al [7], to avoid overlap, we here only describe the basic structure of an EPA so as to exhibit at which point our filters are installed. An EPA performs from at least one and up to three logical steps, as shown in Figure 3:

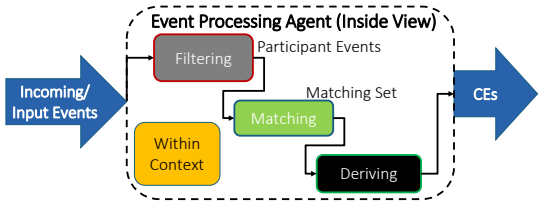


Figure 3: EPA Basic Structure.

- The filtering step has nothing to do with our in-situ filters. It concerns simple filter conditions for selecting relevant events from the input. In our motivating example of Section 2, such filter conditions involve that the call must be outgoing and the call destination should be in the set of potentially premium locations.
- The matching step takes all events that passed the filtering and looks for matches between these events, using an event processing pattern or some other kind of matching criterion. The output of this step is the matching set. Compared to the target queries described in Section 3, this corresponds to the `PATTERN` compartment of the posed query. To be more precise, an EPA can host either a `NON_AGGR` or an `AGGR` operator. So, to form the `PATTERN` compartment of the generic query formulation presented in Section 3, one needs m EPAs, one for each $\texttt{AGGR}_j$, to provide input to a `NON_AGGR` EPA.
- The derivation takes the output from the matching step and produces the output events by applying a derivation formula.

Note that in Figure 3, "within context" interprets to the window constraints of the query formulation in Section 3 but, in general, it may involve other kinds of dynamic context as well. Please see Proton's documentation [1] for a more detailed presentation.

The uncertain version of Proton incorporates uncertainty either on the matching or the derivation step within an EPA. Assigning uncertainty in the derivation step essentially tags each output CE with a certainty value. So, this involves the probability of the CE to have occurred or the experts' confidence on the rule (EPA) itself. On the other hand, uncertainty in the matching step involves uncertain conditions and comes exactly on the pattern matching process. That is, an AGGR (or NON_AGGR) operator is examined during the matching step and a full pattern match for it can only occur if the (un)certainty threshold holds. Therefore, this is also where the `HAVING Q.Certainty> C` of our generic query formulation in Section 3 corresponds to. For supporting such kind of uncertainty criterion within the matching step of an EPA, each SDE or CE possesses a built-in Certainty attribute that stores the certainty of this event. An event has a default certainty value equal to 1, while it can have any value between (0-1].

With respect to the supported distributions, the uncertain version of Proton supports all the distributions that are marked with a

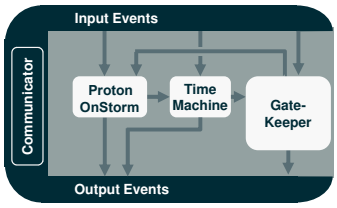


Figure 4: Intra-site structure in FERARI. An Apache Storm topology runs at each site. The CEP Engine module is ProtonOnStorm [2].

✔ in Table 1. These include [7] the Normal, Log-Normal, Exponential, Gamma, Logistic (a.k.a. Sigmoid) and Binomial distributions. Moreover, the uncertain version of Proton implements built-in functions for computing the PDF, CDF, Mean, Var(iance) of each of these distributions.

5.2 The FERARI Platform

Complementarily to the approach by Correia et al [7], which enriches Proton with uncertainty-aware querying and processing functionality, the FERARI platform [12, 13] presents extensions to Proton that enable it to operate both (a) exploiting parallelization opportunities within corporate data centers, public or hybrid clouds and (b) in a geo-distributed manner, i.e., performing CEP across a network of sites.

In FERARI, every site in the network runs an Apache Storm [16] topology composed of the following Spouts and Bolts as illustrated in Figure 4:

- Input Spout: or Input Events in Figure 4, where streaming tuples arrive or events from other sites are fed into the CEP Engine.
- CEP Engine (multiple Bolts): IBM ProtonOnStorm [2] is the utilized CEP engine. ProtonOnStorm breaks its functionality to multiple Bolts which allows for different degrees of parallelization in different event extraction steps. Please refer to the documentation of ProtonOnStorm [2] for further details on its internals. ProtonOnStorm receives the input events from the Input Spout and, having processed them according to the respective EPA(s), it emits CEs towards the Time Machine Bolt.
- Time Machine Bolt: caches events from the CEP Engine and solves out-of-order issues.
- Gatekeeper Bolt: responsible for handling generic streaming operators, such as non-linear functions [18].
- Communicator Bolt: responsible for communication (of Output Events in Figure 4) among sites and towards the coordinator.

A query is submitted at a central site and a FERARI Optimizer acts as the coordinator. It rewrites the query so that it is decomposed to individual EPAs - one for each site. These EPAs incorporate in-situ filters for communication-efficient execution of the distributed CEP process, but the filters are not uncertainty-aware. An example of such a filter is presented in Section 2 and additional examples are included in our case study (Section 5.3). Finally, the optimizer transmits a JSON file to every site, including configuration of in-situ filters and of the EPAs where they are installed, as well as other Storm topology and communication parameters.

5.3 Towards Uncertainty-Aware FERARI

Let us now see how our techniques bridge the gap between the two prototypes. First, we focus at the level of the network of sites. Should the FERARI optimizer act as the coordinator, it should be extended to account for the probabilistic filters we propose in this work together with the monitoring protocol that we introduce in Section 4.3 and Section 4.4, respectively. Given the fact that the optimizer already supports (rewrites queries and constructs site configurations using) in-situ filters that are not uncertainty-aware and also executes some kind of distributed monitoring protocol with equivalent number of phases, implementing the proposed functionality essentially involves: (a) density estimation per site and overall during the initialization phase, (b) support for uncertainty-aware EPAs per site during the Monitoring phase and (c) implementation of the Synchronization phase, as described in Section 4.4.

(a) and (c) simply involve extending the code of the optimizer so that proper site configurations are transmitted by the coordinator at the beginning and end of the Initialization and the Synchronization phase. During these phases, sites only need to know if the events of the posed query should be cached in the Time Machine Bolt or transmitted via the Communicator Bolt. The rest is done in the coordinator. More precisely, at the beginning of the Initialization phase, the coordinator instructs the sites to transmit every relevant event broadcasting a corresponding message. At the end of this phase, it transmits separate site configurations including uncertainty-aware EPAs formed according to the query and the in-situ filters we propose. Similarly, during the Synchronization phase, the coordinator informs the sites whether they need to transmit every event that has been cached or currently arrives (Sync Case A), or switch again to the Monitoring phase, potentially with new in-situ filters should the slack allocation succeeds (Sync Case B).

The job that needs to be done at individual sites is to independently execute the Monitoring phase and apply the in-situ filters we introduce in this work. If a site finds that its in-situ filter is violated, it informs the coordinator. Note that EPAs are defined and executed withing the CEP engine. Therefore, we should practically find a way to combine the functionality of the prototype in Correia et al [7] and FERARI [13] so that we can install the in-situ filters we propose and execute our distributed monitoring protocol. However, the prototype in Correia et al [7] accepts definitions of uncertainty-aware EPAs, but can only be installed at the coordinator and let sites transmit events as soon as they occur. On the other hand, FERARI does not include support for defining uncertainty-aware EPAs but supports distributed monitoring over the network and parallel processing within each site.

There are three ways to go to implement (b) combining the virtues of each of the prototypes:

- Option 1: The first option is to replace ProtonOnStorm with the uncertainty-aware version of Proton, encapsulating the latter version in a single Bolt. The pros of such an approach is that it directly exploits the built-in functionality regarding uncertainty handling as well as the readily available infrastructure for distributed processing over a number of sites provided by FERARI. Nonetheless, the cons come from the fact that the parallel processing of various EPAs within each of the sites should be manually configured as the uncertainty-aware Proton does not take care

of that and its functionality is not split into separate Bolts. Furthermore, contrary to ProtonOnStorm, the uncertainty-aware version of Proton is not open source.
- Option 2: A second option would be to implement the required functionality for uncertainty handling within ProtonOnStom, but this requires expert insights on the internals of both versions of the CEP engine.
- Option 3: A third option is to use ProtonOnStorm as is in the FERARI platform, configure each non uncertainty-aware EPA to output the events that produced a CE packed together with the CE itself and implement the part of the evaluation which involves uncertainty handling outside the CEP engine. In particular, this functionality can be implemented within the GateKeeper Bolt as with non-linear function handling. This approach preserves the open source nature of the platform and unleashes support for all the distributions cited in Table 1. Importantly, it decouples uncertainty handling from the adopted CEP Engine. Thus, one can then easily use Proton or alternative CEP engines, such as Esper which has already been incorporated in the FERARI architecture [23]. On the down side, this option overburdens the Time Machine Bolt with queuing the input events of each CE until they are dequeued and re-processed by the GateKeeper.

From the application perspective, these options do not make a difference. The application just needs to build the query using, for instance, an authoring tool [7] and the coordinator should cooperate with the sites to accomplish the rest. In the next section, we present a use case scenario which better exhibits the functionality of each of the prototypes discussed in Section 5.1 and Section 5.2 as well as the breakthrough our in-situ filters bring.

5.4 Mobile Fraud Detection: A Case Study

In our case study, we utilize (masked due to company's security policies) fraud detection rules from the telecommunications domain, resembling the ones used in FERARI [13]. Event patterns expressing masked rules for mobile fraud detection are analyzed in separate sections below. We form a scenario where uncertainty comes from incomplete data due to subscriber privacy policies. In particular, because we have access to a privacy-aware version of each Call Detail Record (CDR), we are not aware about the actual duration of the call handled by each antenna, but we only know the charged duration in minutes. This, together with the fact that neither the plan of the caller nor the number of charged units for each call are made known to us, introduces uncertainty with respect to the actual duration of a call. Moreover, we are not aware about the actual expense of the call, since the plan of a user may impose zero charges up to a certain call duration or prepaid card deposit and only start charging the parts of the calls that deviate from the plan.

We study calls to Voice over IP (VoIP) destinations and in one of the queries we exploit knowledge about the fact that such calls may be characterized as suspicious with a certain probability (confidence) $0 < p < 1$. This introduces another type of uncertainty similar to the one in Section 2. Recall that in all such cases the call count, sum of duration or sum of monetary cost of each call may be spread across the network of antennas handling these calls, as subscribers commute.

Without any further knowledge, to confront this situation, for each update on the duration of the call (and the sums of such updates) we consider the charged duration of the call as the actual duration plus a certain amount of error incorporated in it. Therefore, we model the actual duration of the call as a random variable following a Normal distribution centered in the middle of the duration interval with a standard deviation such that $[\mu - 3\sigma, \mu + 3\sigma]$ covers 99% of the duration. We apply a similar model for the call cost assuming charges would come per second from the beginning of the call.

We present sample analytics queries in natural language, while in Figure 5 and Figure 6 we present the respective EPAs along with specifying where in the network they are executed. It is important to note that in all the cases the application is oblivious to the underlying site distribution and it only specifies the EPA as if it would be executed locally in a machine. The coordinator (FERARI optimizer) is then responsible for locally installing this EPA (marked with "at coordinator" in the figures) and decompose it to EPAs installed on individual sites in the scope of the distributed execution (marked with "at each site A_i" in the figures).

For each EPA we only show how the query is conveyed from natural language to the steps that take place inside the EPA, exemplifying the generic structure of Figure 3. For the filtering step, we show the filtering expression; for the matching step, we note the pattern variables and the respective thresholds; while for the deriving step, we mention the value assignments. Note that sites essentially forward modified (excluding irrelevant to the CEP query attributes) input events when their in-situ filters do not hold. Hence, they do not communicate any final CEs to the coordinator. Therefore, the EPAs that are to be executed at individual sites have an output marked as temporary CEs, i.e.,"TempCE". Importanty, in the matching step of each EPA we note the matching condition which has the reverse inequality compared to the in-situ filter. This is because, if $P[X_i \leq T/N] \geq \sqrt[N]{1-C}$ is the corresponding in-situ filter, the matching condition for potentially producing a CE is $P[X_i \leq T/N] < \sqrt[N]{1-C}$. The correspondence of variables to X and X_is are highlighted in red in Figure 5 and Figure 6.

LongToVoIPCalls: For each callerID, provide an alert when (i) within the last Y minutes, (ii) she makes calls with a VoIP prefix, (iii) and the total (sum) duration of these calls is larger than T minutes with probability above C. Figure 5(a) shows the uncertainty-aware EPA as posed by the application. Since the prototype in Correia et al [7] does not account for distributed processing, this EPA will be executed only at the coordinator and all sites will transmit all relevant events as soon as they occur. The filtering step of the EPA includes the simple conditions of the call being outgoing ('O') and of 'VoIP' prefix. The matching step expresses the criteria for full pattern matches to occur: total (sum) duration of these calls is larger than T minutes with probability above C, and finally the derivation step assigns a value to the total sum along with the certainty of the extracted CE.

Figure 5(b) shows the FERARI version of the same EPA, i.e., without support for uncertainty, while Figure 5(c) exhibits how the coordinator will decompose it to individual EPAs installed at each site. In particular if $SumLengthIsX_i > T/N$ in at least one site, this site will inform the coordinator which will perform a non uncertainty-aware

synchronization process. If $SumLengthIsX_i \leq T/N$ for all sites, no communication takes place because $X = \sum_{i=1}^{N} SumLengthIsX_i \leq \sum_{i=1}^{N} T/N = T$.

Finally, Figure 5(d) illustrates the contributions of the in-situ filters we propose. In Figure 5(d) we show how the coordinator, having received the EPA of Figure 5(a) (the coordinator's EPA is the same for our approach and that of Correia et al [7]) as input, will decompose it to individual, this time uncertainty-aware, EPAs per site.

ExpensiveToVoIPCall: For each callerID, provide an alert when (i) within the last Y minutes, (ii) she makes calls with a VoIP prefix, (iii) and the total (sum) cost of these calls is larger than T monetary units with probability above C. The above query is equivalent with respect to the EPA representation with the "LongToVoIPCalls" one, since they both involve thresholded summations of uncertain durations and monetary costs, respectively. Therefore in Figure 5 we only present the case of the "LongToVoIPCalls" query.

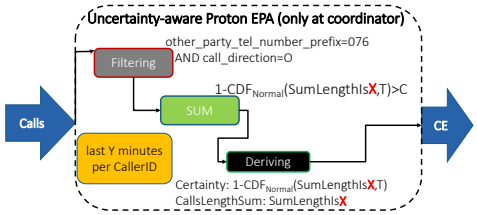
FrequentToVoIPCalls: For each callerID, provide an alert when (i) within the last Y minutes, (ii) she makes calls with a VoIP prefix, (iii) and the probability of more than T, each suspicious with p, calls is higher than C. Similar observations can be extracted for this query illustrated in Figure 6. The difference here is that we have a count of calls to monitor and since each such call is suspicious at a certain level of confidence (Bernoulli trial with p success probability), this count essentially corresponds to a Binomial variable that exceeds the threshold T with probability above C. This is the new pattern that is noted in the matching step of Figure 6(a).

Figure 6(b) shows the FERARI version of the same EPA without support for uncertainty, while in Figure 6(c) the coordinator decomposes the EPA of Figure 6(b) to individual EPAs installed at each site. If $CallCountIsX_i \leq T/N$ for all sites, $\sum_{i=1}^{N} CallCountIsX_i \leq \sum_{i=1}^{N} T/N = T$ and thus no synchronization takes place.

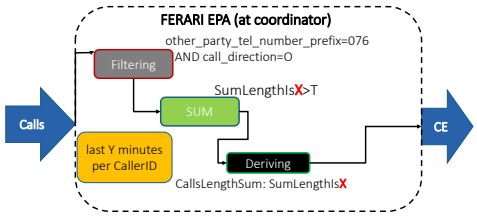
Finally, Figure 6(d) illustrates how the coordinator will decompose the EPA of Figure 6(a) to individual, uncertainty-aware EPAs and send respective configurations to sites, as entailed by this work.

5.5 Preliminary Evaluation Results

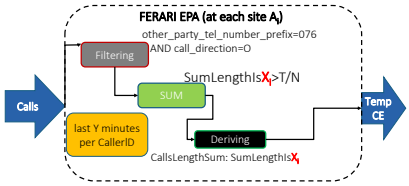
For our evaluation we used anonymized data provided by a large telecom provider [12, 13]. More precisely, we utilized approximately 160.000.000 Call Detail Records from the provider's network from the period between 01.01.2015 and 12.01.2015. In our preliminary evaluation, we compare the communication cost entailed by our approach to one that mixes FERARI with an uncertainty aware coordinator. More precisely, recall that FERARI applies in-situ filters which are not uncertainty aware (see Figures 5(b),5(c) and Figures 6(b),6(c)). What passes through those filters goes to the coordinator where an installation of what was previously termed as "Uncertainty-aware Proton" applies all evaluation aspects related to uncertainty. On the contrary, our approach applies uncertainty-aware in-situ filters (Section 4.3) and the introduced monitoring protocol (Section 4.4). We then measure the ratio of the communication cost (number of transmitted messages) entailed by our techniques versus this mixed approach. Note that, from our comparative analysis, we exclude the naive approach of centralizing all data to the coordinator and then let "Uncertainty-aware Proton" process event tuples, since it entails an amount of communication



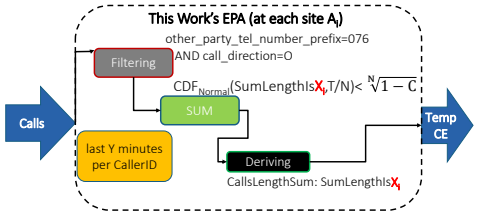
(a) EPA Structure under Correia et al [7] executed only at the coordinator



(b) EPA Structure in FERARI's Coordinator



(c) EPA Structure in FERARI's Sites

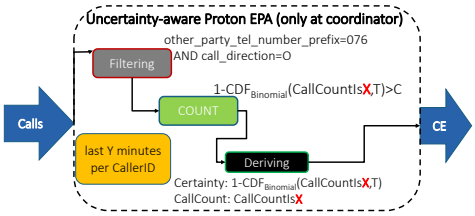


(d) EPA Structure in This Work's Sites. Coordinators EPA is equivalent to Fig. 5(a)

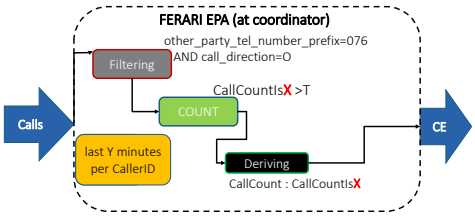
Figure 5: EPA Structure for LongToVoIPCalls Query in each Prototype

analogous to the number of relevant records in the utilized dataset, precisely known even before running any actual experiment. Further note that we do not evaluate the accuracy of the rules since our techniques do not affect that accuracy. Our in-situ filters and communication protocol do not affect how well real mobile fraud cases are pinpointed, which depends on how much suitable is the rule, but reduce the communication cost while evaluating these rules. Our preliminary experiments simulated Option 3 among those mentioned in Section 5.3, mainly because it is more easy to patch the open source code of the GateKeeper Bolt with libraries of decomposable distributions along with built-in functions for PDF, CDF etc. For instance, one may import the JDistlib Library [1] for supporting such distributions.

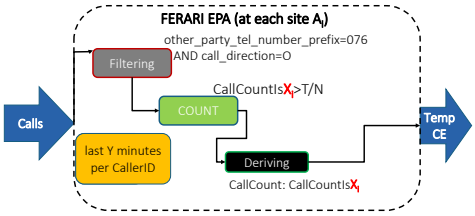
[1] http://jdistlib.sourceforge.net/



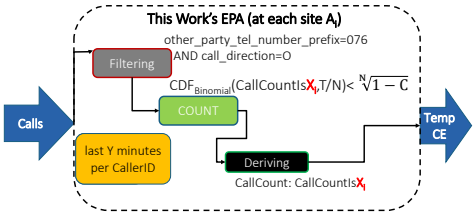
(a) EPA Structure under Correia et al [7] executed only at the coordinator



(b) EPA Structure in FERARI's Coordinator



(c) EPA Structure in FERARI's Sites



(d) EPA Structure in This Work's Sites. Coordinators EPA is equivalent to Fig. 6(a)

Figure 6: EPA Structure for FrequentVoIPCalls Query in each Prototype

We experiment with different values of C and N. We vary C between 0.5 and 0.9, while $3 \leq N \leq 10$. To study the effect of N on the constructed in-situ filters, we initially set $N = 3$ and monitor all calls that are handled by no more than 3 antennas. Then, at each step h, we increase the value of N to $N = 3 + h \leq 10$. Thus, in the next steps all calls handled by at most $3 + h$ antennas are included in the evaluation and the respective value of N is uniformly set in their in-situ filter. Our results show that the ratio of messages communicated by the mixed FERARI approach over those of this work, i.e, $r = \frac{\#MesgFERARI}{\#MesgThis}$ exceeds a factor of 10 for ($C = 0.9, N = 3$) and averages to a factor of 4 across the ranges of used values for C and N. The positive, regarding communication reduction, effect of our in-situ filters fades for values of C approaching 0.5, in which case the ratio r is to approach a value of 1. Furthermore, it fades as N approaches 10 (i.e., when subscribers commute fast and the number of antennas potentially handling their call can grow large), because then (a) the quantity $\frac{T}{N}$ used in Inequality 2 approaches zero and (b) the quantity $\sqrt[N]{1-C}$ in the same inequality approaches 1. Again then, the ratio r is to approach a value of 1. To handle high N values, one can incorporate sampling techniques such as those discussed in Giatrakos et al [18], which we leave for future work.

6 ADDITIONAL APPLICATIONS

Our techniques are directly applicable for serving uncertainty-aware event analytics (a) across various Big Data platforms and (b) over heterogeneous clusters with CPU and GPU accelerators.

Cross-platform Event Analytics: the need to provide event analytics across a number of Big Data platforms may arise for a number of reasons. First, because Big Data platforms evolve over time. Thus, some event processing pipelines may be set up over an Apache Storm [16] installation, while some others may exploit the more recent FlinkCEP API in Apache Flink [14]. Second, because parts of the CEP task may be more efficiently executed in a specific platform. Consider a CEP application which engages SDEs involving the processing of graph interactions, such as likes or friend requests in social networks, together with non-graph related SDEs. The application can exploit both the GraphX API in Spark [15] and the Spark Streaming API to process each SDE category. Third, because cross-platform execution may be mandatory. Consider for instance information pipelines that span coalitions of organizations each using its preferred Big Data platform. In all these cases, naively communicating event data across platforms could affect the time performance of the overall event analytics execution. For instance, one needs to repartition and convert event tuples to micro-batches in order to get cross-platform analytics across Flink and Spark Streaming. This may not only affect the logical organization of the processing, but also the physical task execution.

Event Analytics over Heterogeneous Clusters/Clouds: There is a recent trend in major cloud providers, such as Amazon AWS, to provide High Performance Computing (HPC) cloud infrastructure by allowing clients use CPU and GPU servers on-demand, optimized for specific applications. Prior work [8] on hardware accelerated CEP has pointed out that the benefits provided by GPUs strongly depend on the amount of event data transferred among the main (CPU) and the GPU memory. By reducing such communication to a minimum, our techniques can efficiently serve this aspect of distributed, uncertainty-aware CEP execution as well.

7 RELATED WORK

CEP under Uncertainty. A key survey on uncertain CEP is provided by Alevizos et al [5]. There, the work by Wang et al [32] is reported as the only one that considers uncertain CEP over distributed settings. However, contrary to our work, the work by Wang et al does not impose in-situ filters in order to avoid communication. Instead, the proposed technique lets every site compute probabilities of full or partial pattern matches locally per site and then accumulates these results to a central site to compute the final CEs. These final CEs are then forwarded to the CEP query source. The techniques we develop significantly differ because, by employing in-situ filters, we totally suppress communication among sites in

case these local filters indicate that a CE cannot have occurred even upon synthesizing data from other sites.

Distributed CEP. The seminal work of Akdere et al [3] elaborates on communication- efficient evaluation of SEQ, AND operators, based on the application of a push-pull rationale under deterministic event occurrence. The coordinator sets the most rare events that are input to an operator in push mode and the more frequent ones in pull mode. Events in push mode are transmitted by the various sites towards the coordinator when they occur, while the rest are cached until their state changes to push mode. The state of an event input changes to push mode, by a corresponding message from the coordinator, upon the occurrence of the more rare events. Apart from the fact that the proposed technique does not account for uncertainty, it is further restricted to SEQ, AND. This holds because the push-pull rationale can only be applied to operators which require all their inputs to occur before they output a CE and thus, sites can suppress communication unless rare events occur.

Communication-efficient, distributed CEP also appears in Hermes [29], PADRES [25], Cordies [21] and DHCEP [30]. The focus, there, is to choose an alternative site in the network that lies closer to the event sources to act as the coordinator. Because the coordinator is closer to event sources, the total communication in the network is reduced by avoiding having event data being routed in long paths towards the query source, which finally receives only the produced CEs. Among them, Hermes uses a Distributed Hash Table (DHT) and picks the coordinating site in order to minimize the hop count in the paths event data follow. DHCEP uses a network usage metric in order to choose a proper coordinator. Network usage is the sum of products of $dataRate \times latency$ on communication links. Hence, all these techniques that choose the position of the coordinator are orthogonal to our in-situ filters and distributed monitoring protocol.

8 CONCLUSIONS AND FUTURE WORK

In this work, we analyzed the distributed execution of uncertainty-aware event analytic queries. Our queries engage both aggregation and non-aggregation operators and incorporate thresholds on both the aggregation and uncertainty values. To reduce the amount of communicated data when such queries are executed, we introduced uncertainty-aware in-situ filters to be installed on individual sites of the distributed, networked architecture. We further proposed a novel distributed monitoring protocol that exploits these in-situ filters to suppress communication as much as possible. Finally, we elaborated on system aspects and explained how our work bridges the gap between two recent prototypes that have been proposed in the literature. Our future work heads towards further exploring Options 1-3 mentioned in Section 5.3, dealing with high values of N, potentially via sampling among sites (Section 5.5), as well as exploiting our techniques in the context of event forecasting [4] across Big Data platforms and heterogeneous clusters (Section 6).

ACKNOWLEDGMENTS

This work has received funding from the EU Horizon 2020 research and innovation program INFORE under grant agreement No 825070.

REFERENCES

[1] 2019. IBM Proactive Technology Online. "https://github.com/ishkin/Proton/tree/master/IBMProactiveTechnologyOnline". [Online; accessed 31-January-2019].
[2] 2019. IBM Proactive Technology Online on STORM. "https://github.com/ishkin/Proton/tree/master/IBMProactiveTechnologyOnlineonSTORM". [Online; accessed 31-January-2019].
[3] Mert Akdere, Ugur Çetintemel, and Nesime Tatbul. 2008. Plan-based complex event detection across distributed sources. *PVLDB* 1, 1 (2008), 66–77.
[4] Elias Alevizos, Alexander Artikis, and Georgios Paliouras. 2018. Wayeb: a Tool for Complex Event Forecasting. In *LPAR*. 26–35.
[5] Elias Alevizos, Anastasios Skarlatidis, Alexander Artikis, and Georgios Paliouras. 2017. Probabilistic Complex Event Recognition: A Survey. *ACM Comput. Surv.* 50, 5 (2017), 71:1–71:31.
[6] Michael H Cahill, Diane Lambert, José C Pinheiro, and Don X Sun. 2002. Detecting fraud in the real world. In *Handbook of massive data sets*. Springer, 911–929.
[7] Ivo Correia, Fabiana Fournier, and Inna Skarbovsky. 2015. The uncertain case of credit card fraud detection. In *DEBS*. 181–192.
[8] Gianpaolo Cugola and Alessandro Margara. 2012. Low latency complex event processing on parallel hardware. *J. Parallel Distrib. Comput.* 72, 2 (2012), 205–218.
[9] Gianpaolo Cugola, Alessandro Margara, Matteo Matteucci, and Giordano Tamburrelli. 2015. Introducing uncertainty in complex event processing: model, implementation, and validation. *Computing* 97, 2 (2015), 103–144.
[10] Pedro OS Vaz De Melo, Leman Akoglu, Christos Faloutsos, and Antonio AF Loureiro. 2010. Surprising patterns for the call duration distribution of mobile phone users. In *ECML/PKDD*. 354–369.
[11] Opher Etzion and Peter Niblett. 2010. *Event Processing in Action*. Manning Publications Company.
[12] Ioannis Flouris, Vasiliki Manikaki, Nikos Giatrakos, Antonios Deligiannakis, and Minos N. Garofalakis et al. 2016. Complex event processing over streaming multi-cloud platforms: the FERARI approach: demo. In *DEBS*. 348–349.
[13] Ioannis Flouris, Vasiliki Manikaki, Nikos Giatrakos, Antonios Deligiannakis, and Minos N. Garofalakis et al. 2016. FERARI: A Prototype for Complex Event Processing over Streaming Multi-cloud Platforms. In *SIGMOD*. 2093–2096.
[14] Apache Software Foundation. 2019. Apache Flink. https://flink.apache.org/. [Online; accessed 31-January-2019].
[15] Apache Software Foundation. 2019. Apache Spark. https://spark.apache.org/. [Online; accessed 31-January-2019].
[16] Apache Software Foundation. 2019. Apache Storm. http://storm.apache.org/. [Online; accessed 31-January-2019].
[17] Vijay Garg. 2010. *Wireless communications & networking*. Elsevier.
[18] Nikos Giatrakos, Antonios Deligiannakis, Minos N. Garofalakis, Daniel Keren, and Vasilis Samoladas. 2018. Scalable approximate query tracking over highly distributed data streams with tunable accuracy guarantees. *Inf. Syst.* 76 (2018).
[19] Douglas M Hawkins and RAJ Wixley. 1986. A note on the transformation of chi-squared variables to normality. *The American Statistician* 40, 4 (1986), 296–298.
[20] Jaakko Hollmén et al. 2000. *User profiling and classification for fraud detection in mobile communications networks*. Helsinki University of Technology.
[21] Gerald G. Koch, Boris Koldehofe, and Kurt Rothermel. 2010. Cordies: Expressive Event Correlation in Distributed Systems. In *DEBS*. 26–37.
[22] Michael H Kutner, Christopher J Nachtsheim, John Neter, William Li, et al. 2005. Applied linear statistical models.
[23] Konstantinos Kyriakopoulos. 2018. *Distributed Complex Event Processing (CEP) System Based on the Esper Engine*. Master's thesis. School of Electrical and Computer Engineering, Technical University of Crete.
[24] Lawrence M Leemis and Jacquelyn T McQueston. 2008. Univariate Distribution Relationships. *The American Statistician* 62, 1 (2008), 45–53.
[25] Guoli Li and Hans-Arno Jacobsen. 2005. Composite Subscriptions in Content-based Publish/Subscribe Systems. In *Middleware*. 249–269.
[26] Zheng Li and Tingjian Ge. 2016. History is a mirror to the future: Best-effort approximate complex event matching with insufficient resources. *PVLDB* 10, 4 (2016), 397–408.
[27] Peter G Moschopoulos. 1985. The distribution of the sum of independent gamma random variables. *Annals of the Institute of Statistical Mathematics* 37, 1 (1985), 541–544.
[28] Clifton Phua, Vincent Lee, Kate Smith, and Ross Gayler. 2010. A comprehensive survey of data mining-based fraud detection research. *arXiv preprint arXiv:1009.6119* (2010).
[29] Peter R. Pietzuch, Brian Shand, and Jean Bacon. 2003. A Framework for Event Composition in Distributed Systems. In *Middleware*. 62–82.
[30] Bjorn Schilling, Boris Koldehofe, and Kurt Rothermel. 2011. Efficient and Distributed Rule Placement in Heavy Constraint-Driven Event Systems. In *HPCC*. 355–364.
[31] Fred W Steutel and Klaas Van Harn. 2003. *Infinite divisibility of probability distributions on the real line*. CRC Press.
[32] Y. H. Wang, K. Cao, and X. M. Zhang. 2013. Complex event processing over distributed probabilistic event streams. *Computers and Mathematics with Applications* 66, 10 (2013), 1808–1821.