AQuA-CEP: Adaptive Quality-Aware Complex Event Processing in the Internet of Things

Majid Lotfian Delouee
Bernoulli Institute
University of Groningen
Groningen, The Netherlands
m.lotfian.delouee@rug.nl

Boris Koldehofe
Institute of Applied Computer Science
Technische Universität Ilmenau
Ilmenau, Germany
boris.koldehofe@tu-ilmenau.de

Viktoriya Degeler
Informatics Institute
University of Amsterdam
Amsterdam, The Netherlands
v.o.degeler@uva.nl

ABSTRACT

Sensory data profoundly influences the quality of detected events in a distributed complex event processing system (DCEP). Since each sensor's status is unstable at runtime, a single sensing assignment is often insufficient to fulfill the consumer's quality requirements. In this paper, we study in the context of AQuA-CEP the problem of dynamic quality monitoring and adaptation of complex event processing by active integration of suitable data sources. To support this, in AQuA-CEP, queries to detect complex events are supplemented with consumer-definable quality policies that are evaluated and used to autonomously select (or even configure) suitable data sources of the sensing infrastructure. In addition, we studied different forms of expressing quality policies and analyzed how it affects the quality monitoring process. Various modes of evaluating and applying quality-related adaptations and their impacts on correlation efficiency are addressed, too. We assessed the performance of AQuA-CEP in IoT scenarios by utilizing the notion of the quality policy alongside the query processing adaptation using knowledge derived from quality monitoring. The results show that AQuA-CEP can improve the performance of DCEP systems in terms of the quality of results while fulfilling the consumer's quality requirements. Quality-based adaptation can also increase the network's lifetime by optimizing the sensor's energy consumption due to efficient data source selection.

CCS CONCEPTS

• **Computer systems organization → Real-time systems**; • **Networks → Network dynamics**; • **Software and its engineering → Publish-subscribe / event-based architectures**.

KEYWORDS

Complex Event Processing, Stream Processing, Adaptation, Quality, Internet of Things.

ACM Reference Format:
Majid Lotfian Delouee, Boris Koldehofe, and Viktoriya Degeler. 2023. AQuA-CEP: Adaptive Quality-Aware Complex Event Processing in the Internet of Things. In *Proceedings of ACM Conference (DEBS '23).* ACM, New York, NY, USA, 12 pages.

1 INTRODUCTION

Reacting to unstable situations is a fundamental requirement in the Internet of Things (IoT) scenarios like traffic monitoring, healthcare systems, and smart homes. Distributed Complex Event Processing (DCEP) is a widely employed paradigm to support efficient situation detection based on a variety of different sensors and a step-wise transformation from *primary events* to situations of interest for consumers in the form of *complex events*. The resulting quality, usually expressed in the form of Quality of Service (QoS) and Quality of Results (QoR), highly depends on the origin of primary events, especially in IoT scenarios where the primary events are generated often based on distributed sensor readings from the environment. These sensing deployments are vulnerable to the immense dynamicity in the environment (e.g., availability of the sensors), and more than a single sensing deployment is often needed to meet quality requirements determined by the system or its consumers.

An established solution to react appropriately to environmental dynamics is to adapt the detection logic's placement to the available computing resources, which are part of the DCEP framework. Also, such an adaptation needs to deal with the limitations that the allocated resources might have during the query execution (cf. [6, 21–23, 31, 33]). This idea provides essential means to maintain or improve the QoS-related measures, e.g., by reducing the imposed end-to-end delay or regulating the bandwidth consumption. On the other hand, sacrificing QoR to keep QoS at an acceptable level can already benefit the DCEP systems by combining these mechanisms with other runtime approaches such as load shedding techniques [29, 30]. Although influencing QoR will lead to a degradation in consumer requirements' satisfaction, these techniques find it crucial to impact QoR as less as possible, e.g., by dropping events from partial matches in the event streams.

The state-of-the-art approaches decouple the detection procedure and adaptation strategies from sensing deployment configuration and operate only based on information existing in the design time. Such an idea limits the system's capabilities to react correctly and timely to dynamics in the sensing layer, e.g., the quality of sensor readings or their battery level. Hence, the degradation in QoR can be propagated due to these limitations, while the DCEP system cannot actively influence and prevent the consequences of hardwired sensing configuration. Moreover, even if the sensing deployment adaptation is considered at runtime, adaptation strategies' outcome can affect the QoR [14]. For example, in the case of updating the data sources from a camera to a motion detector, the motion event's accuracy would be degraded. In this vein, due to attaining the most elevated quality grade in the adaptation decisions, the inputs are mandated to have an acceptable level of quality. In this regard, the input data can be assumed of insufficient quality if inaccurate, precise, fresh, or truthful. Events are also evaluated as inadequate quality if they do not hold a certain level of confidence, are received out of order, are wrongly detected, or are not detected.

Therefore, measuring to what extent the consumer's quality requirements are met should be taken into account when applying any adaptation strategy.

In this paper, we analyze and present concepts on expanding flexibility and adaptivity by proposing quality-aware event processing to enhance QoR and QoS. In particular, AQuA-CEP is a mechanism that is designed based on the idea of dynamically exchanging sensing deployment concerning the demarcated requirements by the consumer that influence how sensor data is processed [19]. We enhance DCEP with the concept of so-called quality policies and corresponding quality monitoring mechanisms. Upon any change occurring in the environment or in the sensing deployment observed by our designed DCEP system, AQuA-CEP will autonomously adapt the current sensing deployment according to the available sensing infrastructure, if required. It can be performed by defining sensing configuration restrictions (e.g., cost constraints) considering quality requirements expressed by consumers. Consequently, a utility metric concerning the defined restrictions performs an efficient sensing deployment assignment.

For more details, AQuA-CEP provides the following contributions:

1. We provide a new representation of the quality demands of a query in DCEP systems by proposing a policy-driven specification of complex events to boost data processing performance and more promising utilization of IoT resources.
2. We devise how quality monitoring can be applied in DCEP by presenting concepts allowing the dynamic reconfiguration of appropriate data sources while fulfilling consumer's quality requirements.
3. We explore strategies for configuring quality monitoring agents that trigger adaptation strategies upon any quality policy violation and address the impacts each configuration might have on the DCEP system's performance in terms of QoS and QoR.
4. We evaluate the performance of our proposed mechanism with a real-world dataset alongside using synthetic data to show the ability of AQuA-CEP to boost the performance of the DCEP system in adapting the sensing deployment while observing consumer quality requirements.

The remainder of this paper is structured as follows. We further detail the problem, particularly for an IoT surveillance scenario, and motivate the need for a mechanism for event source adaptation in Section 2. We introduce an overview of the AQuA-CEP system model in Section 3. We elaborate on the problem statement in Section 4. In Section 5, the detailed overview of AQuA-CEP is presented. The evaluation results of AQuA-CEP are exhibited in Section 6. The related work is presented in Section 7. Finally, Section 8 concludes our paper and points to our future work.

2 CASE STUDY: IOT SECURITY MONITORING APPLICATION

In this section, we introduce an IoT scenario that demonstrates the applicability of AQuA-CEP in the context of IoT security surveillance. In this scenario, a state-of-the-art DCEP system fails to couple the DCEP middleware with the available event sources by concepts for (i) quality requirement description, (ii) quality Monitoring, and (iii) sensing deployment adaptation. In other words, by a hardwired sensing configuration in an IoT environment, most DCEP systems fail to exploit multiple sensing deployments. Consequently, they do not benefit from event sources adaptation to make a trade-off between QoR and QoS and react appropriately to the scenario dynamics, e.g., exceeding the coverage of a sensor due to the user's mobility. It indicates that DCEP systems must rethink how switching between event sources at runtime can benefit the system and improve the generated results.

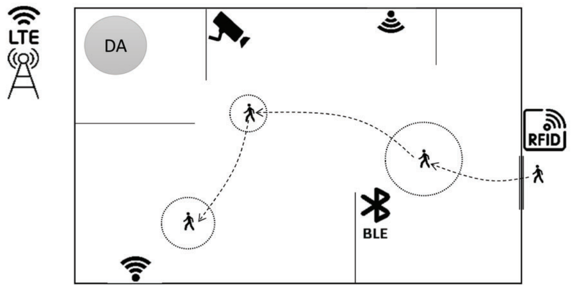


Figure 1: An example of event source adaptation at runtime in an IoT security monitoring scenario.

We consider a continuous query to detect and warn any person approaching a Dangerous Area (DA) in an industrial zone depicted in Figure 1. Any consumer, e.g., a factory's manager, can issue a query to mark a specific point within the area as a DA. The query will check different conditions - the current location of each moving person or object and the list of DAs - observed within the industrial zone. Suppose a person is located closer than a predefined threshold to the Central Point (CP) of a dangerous area (i.e., within the Alarm Region (AR)). In that case, an alarm is triggered and sent to his device to inform him about the security violation. The location events are generated by five distinct sensing deployments, namely RFID, BLE, WIFI, Camera, and LTE signals available in the IoT infrastructures.

By employing AQuA-CEP, a consumer can specify its quality requirements, e.g., high accuracy of location detection close to DA. The location tracking of each target is performed by actively integrating the selected event source (i.e., turning on the mobile data to receive the LTE signals). Therefore, the assignment of any sensing deployment influences the battery life of each target's cellphone. On the other hand, due to the quality levels of each sensing deployment, the location events can be generated with various quality levels that might not satisfy the consumer's expected quality of results. Hence, the DCEP system can monitor the quality and trade the QoR against QoS, e.g., reduce the accuracy of location detection in places far from the DA to achieve energy efficiency. Such a goal can be achieved by adapting the assigned sensing deployment, e.g., switching from highly accurate location sensors like LTE signals to WIFI signals at locations far from DA.

3 SYSTEM MODEL

In this section, we present the system model by introducing AQuA-CEP components, the model of IoT devices, and the provided adaptation models.

3.1 AQuA-CEP Model

We consider a DCEP system to consist of multiple *producers* (e.g., mobile phones, etc.) that generate streams of primary events from the received sensory data and announce them to the system as their advertisements. Correspondingly, *consumers* (e.g., users, applications, services, etc.) create situations of interest as subscriptions and submit them to the system as continuous queries where the set $\mathcal{Q} = \{q_1, \ldots, q_n\}$ denotes the set of currently deployed queries. Moreover, a group of *brokers* (i.e., CEP engines) performs computational tasks (e.g., filter, join, etc.) by hosting a set of *operators* and forward the result to the next step that can be another operator or the consumers.

A query q_i explains the logic by which a complex event can be detected over primary event streams. It can be performed by applying standard CEP operators like pattern matching, aggregation, or windowing over primary events or their attributes. To do so, the imposed complex event detection logic should be applied to the specific brokers for execution. In the meantime, consumers are also allowed to specify their quality requirements as part of the query (e.g., the location accuracy of less than one meter). We denote the set of *consumer-side constraints* of all deployed queries in $\mathcal{Q}$ by $\mathcal{G} = \{g_1, g_2, ..., g_k\}$. Once the query registration is completed, AQuA-CEP is able to run a lookup service over the producers to discover those sensor(s) whose attributes conform to the query's quality requirements. A data source is considered an eligible candidate to feed the system if it can meet all related consumer-side constraints.

3.2 IoT Resource Model

In AQuA-CEP, sensors are the origin of data that measure a specific phenomenon (e.g., temperature) in the environment. The sensory data sources (e.g., Bluetooth) that are used at a given time t form the set of active sensing deployments $\mathcal{SD} = \{sd_1, ..., sd_j\}$. Here, sd_i refers to a specific data source among all available options in the environment that can be participated reliably in the process of sensor assignment for a deployed query. The availability of data sources is dynamic, meaning the set $\mathcal{SD}$ might change over time. In the IoT environment, a data source can be mobile (e.g., sensors embedded in a smartphone) or stationary (e.g., a surveillance camera). We assume that the mobility status of data sources does not negatively or positively affect the quality of their readings.

In our scenario, IoT devices (e.g., smartphones carried by query targets) are interconnected to the system over a wireless sensor network and demanded to register their sensing deployment in the AQuA-CEP in advance. These devices represent the CEP producers who generate primary event streams from sensory data. Also, some of these devices are eligible to issue queries acting as CEP consumers. Moreover, CEP operators can be placed on IoT resources with sufficient computing capabilities, e.g., on the cloud or fog nodes.

3.3 Adaptation Model

For coordinating adaptation and selecting its correct triggers, AQuA-CEP needs to monitor the quality of produced events as well as the state of the sensing infrastructure. In this regard, we build on a sensing middleware (e.g., [1]) that offers the possibility to identify, configure, and access the physical sensors. The system samples the quality level for a subset of the produced events and evaluates potential alternative configurations.

In AQuA-CEP, the adaptation decisions need to serve multiple objectives, e.g., the cost for fulfilling the query's quality policies includes the expenses for utilizing the sensing infrastructure and completing reconfigurations. Besides, adaptation should guarantee a level of stability, which means how often the achieved quality of the detected event stays inside a predefined threshold region after applying the adaptation strategy. It would avoid oscillation and inessential switching costs. Also, with AQuA-CEP, we are looking to adapt the event processing to the environmental dynamics by switching the sensing deployment. For example, given a new set of sensors registered in the network by which some of the currently running queries would be answered. In this case, AQuA-CEP generates new query models considering these new sensors and checks for the costs imposed by transitioning from the current sensing deployments to the newly selected deployments.

4 PROBLEM STATEMENT

Consider the dynamic availability and multitude of event sources for the mentioned use case in Section 2. Due to the mobility of each target or the environmental changes (e.g., a blocked camera), the quality of captured simple events changes over time; thereby, the quality of detected complex events (e.g., alarms) is different. Adapting event sources as a solution to fulfill the required QoR imposes costs that should be considered in adaptation strategies.

In this work, AQuA-CEP selects suitable data sources, i.e., a sensing deployment $\alpha(\mathcal{SD}) \subset \mathcal{SD}$, where α determines which members of $\mathcal{SD}$ will be used. AQuA-CEP is required to meet the consumer constraints (e.g., high accuracy close to DA) or notify consumers when no proper sensing deployment is feasible. Furthermore, each sensor source sd_i of a sensing deployment imposes a *System-Side Cost (SSC)* denoted by $C_{SSC}(sd_i)$ as well as the cost for performing *Quality Monitoring (QM)* for every query q_k denoted by $C_{QM}(sd_i, q_k)$. C_{SSC} includes those quality metrics that are more important for the system (i.e., energy consumption, reusability, resource utilization, etc.). C_{QM} is the cost imposed by the configuration model of the monitoring agent in terms of time (i.e., the delay related to the processing events in the operator and delay for performing the transition between sensing deployments) and computation (i.e., the required computation resources to conduct monitoring process).

More formally, AQuA-CEP aims to find α which minimizes the cost factors imposed by system-side costs and quality monitoring costs subject to the quality constraints of consumers, i.e.,

$$\begin{aligned}
\min \quad & w_s \sum_{sd_i \in \mathcal{SD}} \alpha(sd_i) C_{SSC}(sd_i) \\
& + \; w_q \sum_{sd_i \in \mathcal{SD}} \alpha(sd_i) \sum_{q_k \in \mathcal{Q}} C_{QM}(sd_i, q_k) \\
s.t. \quad & \alpha(\mathcal{SD}) \text{ satisfies constraints in } \mathcal{G} \\
& \alpha(sd_i) = 1 \text{ iff } sd_i \text{ is selected.} \\
& \alpha(sd_i) \in \{0, 1\}
\end{aligned}$$

Here, w_s indicates the weight related to system-side costs, and w_q is the weight associated with monitoring costs.

5 THE AQUA-CEP SYSTEM DESIGN

In Figure 2, we depict the foundational components employed in AQuA-CEP. By utilizing Software-Defined Networking (SDN), we make use of a *controller* to function as the coordinator module and enforce a quality-driven DCEP. This component is logically

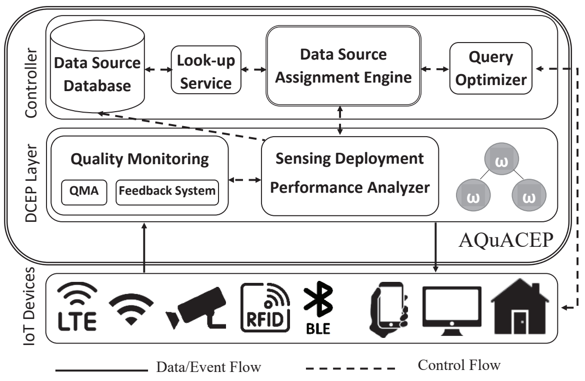


Figure 2: The AQuA-CEP System Design.

centralized but physically distributed and is in charge of exchanging control messages to synchronize the event detection procedure. To do so, the controller owns the principal role in matching the subscriptions to advertisements. A *query optimizer* component receives consumer queries, transforms the query description into a set of event types, and passes the list of required data sources to the *data source assignment engine*. A *look-up service* is triggered by the engine to explore the potential candidates for each event type in *data source database*, where the currently available data sources are previously registered themselves. The records in this database are dynamic and can be registered or canceled at runtime.

Moreover, the controller's functionality is enriched by employing *quality monitoring* agents in the DCEP layer. Upon any predefined situation (e.g., data source disconnection), a so-called *control event* is created by the responsible agent. It notifies the controller to execute corresponding steps as adaptation scenarios in order to maintain the QoR. To do so, a *sensing deployment performance analyzer* investigates the current state of assigned data sources using information acquired by quality monitoring agents and updates the assignment engine to reconfigure sensing deployment, if necessary. The performance analyzer component also updates the data source database's records based on the quality monitoring results. It influences the characteristics of data sources or their availability.

Besides, parts of our system are built on existing concepts for the flexible execution of event processing operators, as proposed in TCEP [22] and CEPLESS [20]. This allows AQuA-CEP to modify the deployment and configuration of operators and integrate a wide range of additional event processing engines, e.g., Apache Flink. With such flexibility, AQuA-CEP can revise the operators' deployment and influence QoS metrics, e.g., bandwidth usage.

5.1 Quality Requirement Description

The foremost step in acquiring the consumers' expectations in terms of quality is the technique by which they can elaborate on their requirements. Such a method must be not only easy to use for the consumers, but also sufficiently comprehensive to cover all aspects and flexible to fit different types of consumers' quality requirements. Hence, we enable consumers to express their specific requirements related to a given query in the form of *quality policy*.

Quality Policy (QP). AQuA-CEP extends traditional query specification of event processing systems by providing the possibility to specify quality requirements. A quality policy can determine various quality metrics essential for the consumers, e.g., the accuracy of delivered values as part of the event detection process. Also, a quality policy might comprise *threshold levels* and *priorities* that can be exploited to optimize and trade between conflicting demands of multiple deployed queries. According to the type of thresholds, the quality policy can be categorized as *static* or *dynamic*.

Quality requirements in the static type of quality policies are specified based on static thresholds as the exact amounts stated clearly in the query, e.g., the temperature data is requested to be delivered with an interval of 10 seconds. Only one type of quality metric can be involved in each static quality policy. Therefore, one expression is required in the query definition to assess multiple aspects of each event. For each data type, the system needs to provide a manner for the consumers to define acceptable values. For instance, to determine the resolution of images, the consumer should have the possibility to specify the thresholds based on PPI (i.e., pixels per inch). Thus, expressing quality policies for consumers will be as easy as possible.

Consider a situation in which a consumer is willing to specify various but related quality requirements based on a second parameter, e.g., various location accuracy thresholds based on the distance to DA for the same query in our use case. In those circumstances, defining multiple static quality policies will enlarge the list of consumer constraints, increasing our problem's complexity. Since the system needs to fulfill only one of those multiple but related static quality policies at a time, we define a dynamic threshold that varies depending on a second factor which can be time or a context-related parameter.

For instance, a dynamic threshold based on the location factor looks like "the location accuracy of an object should be less than 2 meters if it is within 100 meters of a particular area. Otherwise, 10-meter accuracy would be sufficient". By utilizing dynamic thresholds, more intricate descriptions for quality requirements can be explainable, enhancing the flexibility of query definition. Such sort of flexibility will improve consumer satisfaction while optimizing our data analytic system to avoid utilizing more complex procedures to fulfill quality requirements.

In order to validate the admissibility of thresholds, the controller inspects the capability of the data sources and adjusts their characteristics based on the requested thresholds in the query, if applicable. In case there are no appropriate data sources available in the environment concerning the quality requirements, the controller revises the query model with the acceptable thresholds and notifies the consumer about the new query model. Then, based on the feedback obtained from the consumer, the controller will deploy the newly produced query model or cancel the query processing. On the other hand, priorities can also be determined in the query definition. It is worth noticing that a higher query priority will impose higher costs for the query issuer. Such costs can be defined by the system designer depending on the use case.

Moreover, the data source conditions and consumer's quality requirements might be varied over time. That is why the quality policies have to be revised during the runtime. For example, the consumer may need the result of a running query very urgently, such as the current blood pressure of a patient who may have an acute condition with lower intervals. Therefore, the consumer needs to inform the system about this change by raising the query's priority as well as changing the sensing interval's threshold. To do so, AQuA-CEP employs a feedback process for maintaining and updating the quality policies and renewing the policies whenever the quality requirements or the data sources' status is altered.

5.2 Quality Monitoring

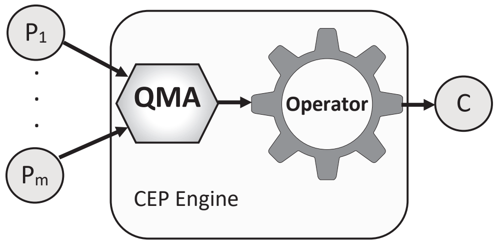
Quality observation should be performed with the lowest possible delay to ensure the expressed quality requirements can be met and adaptation decisions are conducted timely. On the other hand, the available resources for computation are usually bounded. Thus, an event monitoring process must ponder both of these aspects simultaneously.

5.2.1 ***Quality Management Agent (QMA).*** One of the novel traits of AQuA-CEP is to employ QMAs that are liable for inspecting the predefined quality metrics over the event streams and triggering warnings in the matter of quality degradation. On the other hand, the utilization of QMA and where it is hosted in the data plane might influence the quality of service. So, we defined the concept of QMA's configuration model and discussed how it improves the quality in various processing stages.

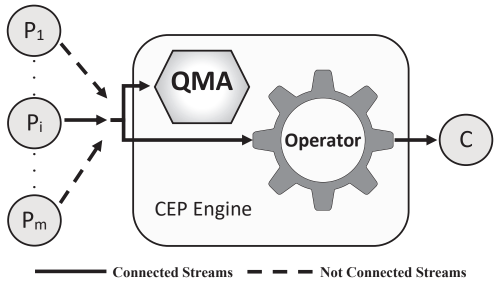
5.2.2 ***QMA Configuration Models.*** As we discussed earlier, the form of QMA configuration in the DCEP layer can yield different results. There are two kinds of configuration models; *sequential* and *parallel* as shown in Figure 3. In a sequential configuration model, the events from all producers that fulfill the requirements of a specific query are aggregated into one joint event stream and fed into the corresponding QMA for quality evaluation. In this case, QMA is in charge of filtering events and allows those events that possess the required level of quality to be delivered to the respective CEP engine. The primary advantage of this model is that the transition time (i.e., handover), the time required to swap data sources, is ensured to remain small, i.e., in the order of milliseconds. Moreover, joining events from two or more producers is feasible to boost the quality by utilizing redundancy. On the other hand, processing all events from the producers indeed imposes considerable latency in query processing that should be taken into account.

The parallel configuration model will impose the minimum possible latency in processing since QMA analyzes the event attributes' quality in parallel. However, connecting one producer to the respective operator is only possible. Thus, if the quality of the event produced by this data source degrades, it takes time for the QMA to trigger an alert and request the controller to link another data source to the CEP engine that fulfills the quality requirements. Such a transition undoubtedly imposes a noticeable overhead on the event processing system. Nevertheless, both processing and transition delay should be considered as the major costs when the system wants to decide on the configuration model of QMAs.

Moreover, the QMA costs for monitoring each data source can be reviewed based on time and computation, directly dependent on the QMA's configuration model. In terms of time, if the parallel configuration model is chosen, the cost is the delay related to switching between the current data source (i.e., sd_i) and the next option (i.e., sd_l) as ($C_s(sd_i, sd_l, q_k)$). On the contrary, if the sequential configuration model was chosen, the time overhead is the delay caused by quality analysis ($C_a(sd_i, q_k)$). In terms of computation costs, the overhead in both parallel and sequential models is almost the same. We called this ($C_w(sd_i, q_k)$) includes the required resources for analyzing an event stream using sliding windows and the computing resources for the data source assignment ($C_{ads}(q_k)$). Therefore, the total cost of monitoring



(a) QMA Sequential Configuration



(b) QMA Parallel Configuration

Figure 3: QMA Configuration Models.

the data source sd_i using a QMA is as follows.

$$\begin{aligned} C_{QM}(sd_i, q_k) = \quad & w_{seq}\, C_s(sd_i, sd_l, q_k) \\ + \quad & w_{par}\, C_a(sd_i, q_k) \\ + \quad & C_w(sd_i, q_k) + C_{ads}(q_k) \\ s.t. \quad & w_{seq} = 1 \text{ iff 'Sequential mode is selected.'} \\ & w_{par} = 1 \text{ iff 'Parallel mode is selected.'} \\ & w_{seq} \in \{0,1\} \text{ and } w_{par} \in \{0,1\} \end{aligned}$$

5.3 Sensing Deployment Adaptation

Adaptation decisions in AQuA-CEP are performed following the MAPE-K feedback loop model [4] building on the previous two steps (i.e., quality description and monitoring). Consequently, the monitoring outcomes are analyzed within every loop to determine adaptation decisions and finally apply them to the stream processing infrastructure.

5.3.1 ***Controller.*** In Algorithm 1, we represent the controller's core functionality in AQuA-CEP.

The first step of query processing is to initialize the corresponding variables, i.e., the set of event types, their quality policies, and their thresholds. When the consumer with ID C_{cid} registers the query (e.g., security monitoring of DA), the controller initiates a subscription for each query's simple event (e.g., location event for each query's target). Such a subscription comprises details regarding the event type of simple event (i.e., et_i) and the respective query (i.e., Qid). On the other hand, a producer P_{pid} registers its available sensing deployments (i.e., $\mathcal{SD}(P_{Pid})$), e.g., LTE signals for location tracking. Then, the controller generates an advertisement for each sensing deployment of the producer. It includes information about the respective data source and the ID of the producer. Finally, the controller investigates possible solutions to match current advertisements to subscriptions (Refer to Algorithm 2).

Algorithm 1 Controller Functionality

```
1:  Initialization:
        $Q_{Qid} = et_1, ..., et_l \leftarrow$ User Query Qid;
        $QP \leftarrow \{qp_1, ..., qp_n\}$;
        $SUB \leftarrow \emptyset$;
        $ADV \leftarrow \emptyset$;
2:  upon ($C_{Cid}$.Submit($Q_{Qid}$)) do
3:      for $et_i \in Q_{Qid}$ do
4:          $SUB \longleftarrow SUB \cup sub^{et_i}_{Qid}$ ;
5:      AssignDataSource($ADV$, $SUB$);                ▷ Algorithm 2
6:  upon ($P_{Pid}$.Register($\mathcal{SD}(P_{Pid})$)) do
7:      for $sd_i \in \mathcal{SD}(P_{Pid})$ do
8:          $ADV \longleftarrow ADV \cup adv^{sd_i}_{Pid}$;
9:      AssignDataSource($ADV$, $SUB$);                ▷ Algorithm 2
10: upon (QMA.Alarm) do
11:     ProcessAlarm(QMA.Alarm);                       ▷ Algorithm 3
```

5.3.2 ***Data Source Management.*** In Algorithm 2, we represent the process of assigning data sources.

Algorithm 2 Data Source Management

```
1:  function AssignDataSource($ADV$, $SUB$)
2:      for $sub^{et_i} \in SUB$ do
3:          $QP_{sub^{et_i}} \longleftarrow$ Related quality policies to $et_i$;
4:          $M_{ADV^{et_i}} \longleftarrow$ Matching advertisements to $et_i$;
5:          for $adv^{sd_i} \in M_{ADV^{et_i}}$ do
6:              if $adv^{sd_i}$.MeetAllQP($QP_{sub^{et_i}}$) then
7:                  $C_{sd_i} \longleftarrow C_{SSC}(sd_i) + C_{QMA}(sd_i, q_k)$;
8:                  $\mathcal{SD}(sub^{et_i}) \longleftarrow \mathcal{SD}(sub^{et_i}) \cup (adv^{sd_i}, C_{sd_i})$;
9:      $\alpha_t(\mathcal{SD}) \longleftarrow$ HACS($\mathcal{SD}$, $\mathcal{G}$);        ▷ Solution
10:     PerformTransition($\alpha_t(\mathcal{SD})$);
11: function PerformTransition( $\alpha_t$)
12:     for ($sub^{et_j}$, $adv^{sd_k}$) $\in \alpha_t$ do
13:         if $sub^{et_j}.sd_{previous} = \emptyset$ then
14:             $Immediate\_Transition(sd_k)$;
15:         else
16:             $Seamless\_Transition(sd_{previous}, sd_k)$
```

To match advertisements to subscriptions in Algorithm 2, the function AssignDataSource looks for potential candidates for each event type (e.g., location event) in the list of related advertisements. In this regard, a data source announced by an advertisement is examined by the MeetAllQP function, which compares the data source's current quality characteristics and the related quality policies' current thresholds (e.g., the accuracy level of fewer than 2 meters). If this data source meets all related quality policies, the corresponding advertisement will be considered a qualified candidate for this subscription (e.g., LTE signal meets the 2-meter accuracy requirement).

The costs of applying this sensing deployment include systems-side and monitoring costs (e.g., energy consumption for utilizing LTE signal) which are calculated for each candidate and paired with its advertisement to form members of a list showing the eligible sensing deployments for each subscription (i.e., $\mathcal{SD}$). Then, a heuristic approach is applied on $\mathcal{SD}$ considering satisfying the constraints in $\mathcal{G}$ (i.e., HACS) to realize an approximate solution for the current situation. This approach checks the currently available sensing deployments and assigns them to the running queries considering the optimization parameter (e.g., maximizing the battery life of targets' cellphones). According to this newly generated solution, if the previous sensing deployment is changed for any query, the transition between data sources can be done in two ways using the function PerformTransition.

The *immediate* transition model occurs when the previous sensing deployment is not available anymore or notably unreliable (e.g., target goes out of the coverage of BLE). So, the controller should perform the transition as fast as possible with minimum delay (e.g., from BLE to WIFI). On the other hand, in *seamless* transition, the previous data source is still available. Still, it cannot satisfy all quality requirements due to changing the quality policy threshold (e.g., more accurate location event close to DA). Therefore, the controller performs the transition smoothly (e.g., from WIFI to LTE). For this type of transition, AQuA-CEP will process both data streams from previous and current producers concurrently in a period of β seconds in which the transition is happening from its invocation to its completion.

5.3.3 ***Adaptation Strategies.*** Algorithm 3 shows the capabilities of the AQuA-CEP to adapt dynamically to the changes in the environment, in the quality of data streams, or process the queries based on the dynamic quality policy thresholds. Hence, a feedback system continuously inspects the conditions in both the data plane and the control plane to trigger alarms upon any noteworthy changes. Each QMA alarm has attributes such as type, corresponding sensing deployment (i.e., sd), quality policy (i.e., qp), and query identifier (i.e., Qid).

Algorithm 3 Alarm Processing

```
1:  function ProcessAlarm($A$)
2:      switch ($A.type$) do
3:          case SDUnavailability:
4:              for $adv^{sd_i} \in ADV$ do
5:                  if $A.sd == sd_i$ then
6:                      $ADV \longleftarrow ADV - \{adv^{sd_i}\}$;
7:              AssignDataSource($ADV$, $SUB$);
8:          case ReducedQuality:
9:              Wait-Monitor($A.sd$);
10:             if $A.sd$ Not Recovered then
11:                 for $adv^{sd_i} \in ADV$ do
12:                     if $A.sd == sd_i$ then
13:                         Update($adv^{sd_i}$)
14:                 AssignDataSource($ADV$, $SUB$);
15:         case ChangedQualityThreshold:
16:             for $sub^{et_i} \in SUB$ do
17:                 if $A.qp \in QP_{sub^{et_i}}$ then
18:                     Update($sub^{et_i}$)
19:             AssignDataSource($ADV$, $SUB$);
20:         case QueryEnded:
21:             for $sub^{et_i} \in SUB_{A.Qid}$ do
22:                 $SUB \longleftarrow SUB - \{sub^{et_i}\}$;
23:             AssignDataSource($ADV$, $SUB$);
```

Upon the arrival of a QMA alarm, if the alarm's type indicates that the connection to a data source is lost and this sensing deployment is not available anymore (i.e., SDUnavailability), the controller removes all the related advertisements and performs data source re-assignment using the global optimizer. The next type of alarm is triggered by a reduction in the quality of data streams concerning the current thresholds of quality policies (i.e., ReducedQuality), e.g., when an obstacle blocks part of a motion detector's vision. In this case, the system performs a *Wait-Monitor* procedure in a specific period, in which AQuA-CEP checks the quality of produced events. If the data source can recover from this situation timely, our mechanism will continue with the current sensing deployment. Contrarily, suppose the lack of sufficient quality remains for the event stream. In that case, firstly, the related advertisements to this sensing deployment will be updated with the new quality characteristics, and then, a re-assignment procedure will start. The main goal of the Wait-Monitor procedure is to prevent oscillation between data sources since it will lead to more switching costs and might produce a worse global solution.

Since the consumer is able to adjust the quality policy threshold at runtime, various ranges are possible for thresholds according to the query model. In such cases, an alarm is triggered (i.e., ChangedQualityThreshold) to indicate that a new threshold should be taken into consideration. Hence, each subscription related to the changed quality policy has to be updated, and a new global re-assignment should be performed. Finally, if a query is finished on time or even ahead of time manually, the corresponding subscriptions will be removed from the set of subscriptions. In addition, the producers and CEP operators should disconnect from each other. Since the absence of those subscriptions may change the global solution, executing the AssignDataSource function on the available advertisements and subscriptions is necessary.

6 EVALUATION

In this section, we experiment with different ways of monitoring the quality of event detection and its corresponding adaptation strategies. The main goals of the evaluation are to figure out 1) can dynamic event source assignment provide a trade-off between QoR (i.e., fulfilling consumer constraints) and QoS (i.e., minimizing the system-side costs) and 2) what limitations are involved in performing a transition among sensing deployments.

6.1 Simulation Setup

We created a single Virtual Machine (VM) in *Oracle VM Virtual Box Manager* [8] in which we installed *Ubuntu version 20* OS. We allocated 6 CPU cores with 100 percent execution capacity and 24 GB of main memory to the VM. We run complex event processing with multi-threading in this machine and create a thread for each of the issued queries; thereby, we could manage them simultaneously using Java. For future work, for making performance-related studies, we would need indeed to implement all components over separate computational resources.

For publish/subscribe communication of AQuA-CEP, we build on *Apache Kafka* [10] as a distributed platform. Furthermore, for detecting complex events, we build on *FlinkCEP* [7], which is a library implemented on top of Apache Flink. In our simulation[1], a Kafka server acts as an event broker that serves both data events and control events, as depicted in Figure 4. We monitor the quality of produced event streams in the QMA, which is located as an Apache Flink operator using the parallel QMA configuration model in the scenarios described below.

[1]https://github.com/majid-lotfian/AQuA-CEP-code

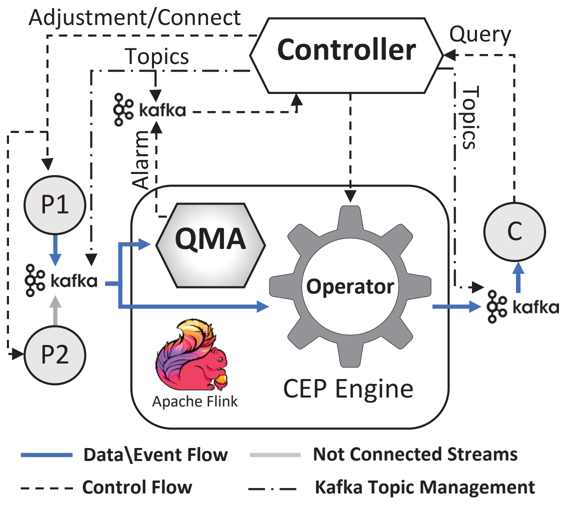


Figure 4: The evaluation model of AQuA-CEP.

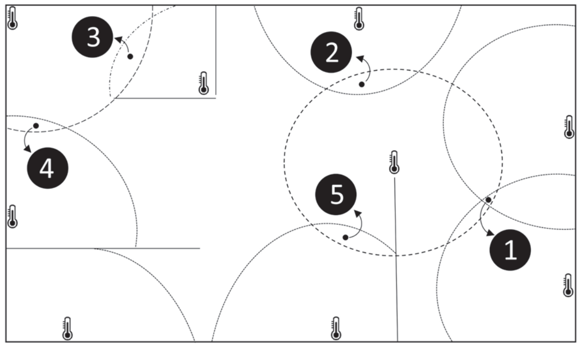


Figure 5: Modified Coverage Problem with Static Quality Policy.

6.1.1 Static Quality Policy Scenario. Consider an area where several temperature sensors are embedded in different locations shown in Figure 5. In such a scenario, the temperature in some locations can be captured by multiple sensors. Each of these sensors has its own characteristics (e.g., sensing accuracy, coverage, etc.) that influence the generated data. We consider a set of continuous queries, each detecting a high temperature in a specific location. We also generated random variations in each sensor's attributes, which simulate the environmental dynamics.

To make our motivation clear in this scenario, we began with a *coverage problem* scenario [28], in which the objective is to solve the sensing placement problem to maximize the coverage of m important points using n sensors. To adequately capture the capabilities of AQuA-CEP, we modified the coverage problem so that sensors are already located in the environment, and their location cannot be changed. Moreover, instead of a set of m critical points, we have a dynamic group of queries to be answered. The goal is to optimize the total energy consumption by activating the set of sensors that can cover all queries.

We analyze the performance of AQuA-CEP in terms of the event loss rate since assumed dynamics directly influence this quality metric. We compared our approach with two baseline mechanisms. The first approach is called *Optimal Dynamic Loss*

Table 1: Applied Queries in the Static Quality Policy Scenario

Q#	Location	Temp Threshold	Loss Rate Threshold
Q1	42:18	> 20.8	< 10
Q2	31:7	> 20.1	< 15
Q3	11:6	> 21.1	< 12
Q4	2:11	> 19.1	< 15
Q5	29:21	> 22.5	< 10

Rate (ODLR), which selects the best data source in terms of Loss Rate at the start of processing a query for each event type. Then, once the monitored loss ratio for produced event stream in runtime goes above its predefined characteristic, the controller first updates this feature with the new assessment and then performs a reassignment check. The second approach is called *Optimal Static Loss Rate (OSLR)*, which selects the best data source in terms of the event loss rate again. But, it stays with this data source until the end of the query and does not switch to another data source even in the case of an event loss ratio increase of its event stream.

We applied the same query with different threshold values, detailed in Table 1, on each approach with the same simulation setup. The points targeted in each query are marked in Figure 5, and two or more sensors can be hired for each. The query definition determines the temperature thresholds, and the minimum required Loss Rate is pinpointed in the respective quality policy. We acquire the consumer queries utilizing a designed JFrame on runtime containing the definition and a set of quality policies. Then, the query is transformed into a CEP-enriched SQL format. An example of a continuous query is shown in Listing 1, which aims to collect the temperature in a specific location. We defined a static quality policy by specifying a threshold value for the event loss rate characteristics of the sensing deployment in the *PATTERN* clause, which will be used in the data source assignment procedure.

```
SELECT  event.*
FROM
    // Selecting event stream
    SELECT  ds.stream
    FROM    DS
    PATTERN
        lossrate < QualityPolicy.threshold
    Within    window_size
    WHERE
        ds.type = 'Temperature'  AND
        ds.coverage(target_location) = True
WHERE  event.value > Query.value
```

Listing 1: Applied query with static quality policy.

In addition, the temperature data in [3] is used, which contains two datasets. We chose one that includes the sensory data about temperature, pH, and turbidity from 30 cm below the water surface. We utilized this dataset because it has a real-world distribution of temperature data that makes our calculation more realistic. In our mechanism, each functioning sensor takes the data from this dataset and transmits its own transformed data according to its predefined quality characteristics. It means that each sensor will produce a unique temperature data stream according to its own features.

In addition, we estimated the amount of energy consumed by each sensor as described in [13]. This includes the energy for sensing the phenomena, processing the measurement, logging (i.e., reading data and writing it into the memory), communication, and transient energy (i.e., the transition energy to go from the idle state to the active state, and vice versa). We presumed that the distance between sensors to the gateway is the same, and they transmit packets of the same size.

Table 2: Sensing Configurations and Their Characteristics

Name	Range (m)	EC (mW)	Accuracy (m)
BLE	70 - 100	426	1 - 3
RFID	1 - 12	375	0.1 - 2
WIFI	50 - 100	817	1 - 5
Camera	N/A	374	< 1
LTE	> Km	1634	< 1

6.1.2 *Dynamic Quality Policy Scenario.*

Consider the mentioned use case in section 2 where the specified dynamic quality policy is "the quality of the person's location can degrade, as the person moves away from the borders of the dangerous area, but it should be as much accurate as possible when it is in a proximity of the target location". In this policy, the quality metric is the location *accuracy*, and the second parameter is the target's *location* (i.e., the accuracy level is determined according to the target's current location). In this scenario, the consumer in the factory needs to submit a query to control people's access to the dangerous area. To do so, the query should consist of the dangerous area's details and the alarm region.

AQuA-CEP can access a sensing infrastructure based on a multitude of sensors deployed on a target's devices (e.g., smartphones), such as Bluetooth Low Energy (BLE), LTE, WiFi, and RFID sensors. Moreover, the system can also benefit from other positioning infrastructure embedded in the environment, like cameras. Each of these sensing deployments has its own characteristics, and in order to estimate the Energy Consumption (EC) in this scenario, we reuse the measurements collected from [9, 17, 26, 32, 36], as indicated in Table 2. Among all sensors, only *Camera* does not utilize the mobile phone's battery since it is a separate camera placed on the wall. Therefore, we assumed this sensor's energy consumption is equal to placing a target's mobile phone in airplane mode.

Since a publish/subscribe mechanism is used to manage people in this factory, a notification event of prohibition to approach a dangerous area is generated. All targets within the factory will be notified of the prohibition. That's because all targets in AQuA-CEP have already subscribed to these notification events when performing the admission process. Moreover, during admission, each target explains its attributes, including its identity and role in the factory. Also, it describes its carrying devices with their sensing capability (e.g., smartphones, tablets, smartwatches, wearables, etc.). In addition, it should be clearly detailed what type of data each device can produce and what sensing and communication technologies it can provide. Also, we assume that each target will give continuous access to their registered sensing deployments and not deliberately block the connection.

The query correlates specific conditions - the target's position, the boundaries of the dangerous area, the alarm region, and the authorization status of the target - to detect a complex event of dangerous area violation. For this case, the perception of a location event could use different sensors to get an approximate position with the required quality level specified in the query's quality policies while optimizing the energy consumption of

Table 3: An example of applied queries with dynamic quality policy

Q#	Definition		Quality Policies	
	CP	AR(m)	Q-Metric	Condition
Q1	60 : 185	70	Accu < 2m	0 < DTDA < 100
			Accu < 5m	100 < DTDA < 200
			Accu < 10m	200 < DTDA < 1000

the application installed in the smartphone. Dependent on various aspects like coverage, location uncertainty, and sensing frequency, various query models can be utilized to meet the QoR and QoS trade-off requirements. In this specific use case, since we do not need a very accurate position when the person is far from the dangerous area, AQuA-CEP uses the sensor, which first meets the quality level of the query and then has the smallest amount of energy consumption for the smartphone.

Alike to the static quality policy scenario, we applied the query to the simulation environment multiple times considering the dynamic quality policy and analyzed the results to gain more insights. With a fixed route for a target person, we randomly generated coordination for the dangerous area, including the details for *CP* and *AR*. That means if a person nears *CP* by less than *AR* meter, a violation will happen. Besides, such a location estimation should be done using a sensor with the accuracy (Accu) level denoted in Table 3, considering the conditions related to the Distance To the Dangerous Area (DTDA).

The CEP-enriched SQL format of the applied query with a dynamic quality policy is represented in Listing 2.

```
SELECT  event.*
FROM
    // Selecting Event Stream
    SELECT  ds.stream
    FROM    DS
    PATTERN
      Accuracy < CurrentQualityPolicy.threshold
    Within    window_size
    WHERE   ds.type = 'Location'
          AND
          // Selecting Current Quality Policy
          CurrentQualityPolicy = (
                SELECT  qp
                FROM    Query.QP
                WHERE   qp.InRange(target_loc))
WHERE  Distance(event.loc, Query.DA) < Query.AR
```

Listing 2: Applied query with dynamic quality policy.

In the first *WHERE* clause (line 9), the current quality policy is determined based on the target's location from the list of quality policies. Based on the threshold of the chosen quality policy, the candidate streams (e.g., LTE signals) are listed and delivered as possible options for this query to a constraint satisfaction global optimization algorithm.

In our simulation, we chose to apply *Choco-solver* [24], an open-source Java library for constraint programming. By employing this library, we are able to solve our optimization problem while considering consumer-side constraints. We expressed the event source assignment task as a variable where the domain of all variables is the available sensing deployments. Therefore, the Choco-solver considers the constraints to find a global solution covering all variables. This optimization solution will determine which stream from the list of candidates should be assigned to each query.

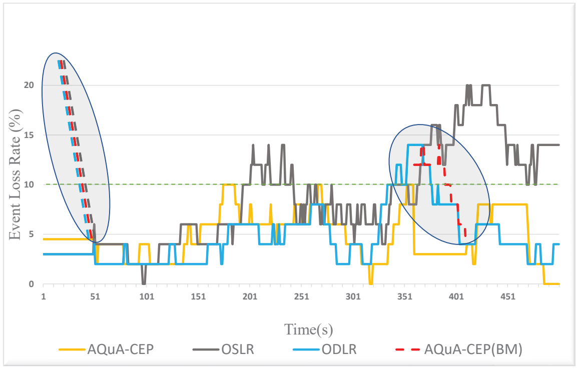


Figure 6: Event loss rate ratio during the execution for query 1 with the static quality policy.

6.2 Simulation Results

In this section, we will analyze our findings compared to the other two mentioned baseline strategies in both categories of static and dynamic quality policies. It should be noted that the results regarding energy efficiency and quality of results do not depend on the number of consumers, but indeed on the number of queries only since a consumer can issue multiple queries. That's why increasing the number of involved queries can help evaluate the scalability of each approach.

6.2.1 ***Static Quality Policy.*** The static quality policy scenario results have been illustrated in Figure 6 and Figure 8. We executed the simulation ten times for each query, and the results were approximately similar. To challenge our approach, we selected the results with AQuA-CEP's worst performance.

Regarding the event stream loss rate in Figure 6, the chart displays the event loss rate for each approach in one execution. It can be seen that the OSLR approach shows the worst performance and proves the idea that each mechanism requires adapting to the dynamics. Both AQuA-CEP and ODLR select those sensing deployments meeting the threshold, which is illustrated as a green dashed line. Only two times our approach exceeds the threshold of the event loss rate highlighted by the gray circles.

In the first quality policy violation, at the start of query execution, we must wait for the window to be completed since we are employing a 50-sec window size over the event stream. So, we can not rely on the monitoring results, and we called this period as *Blind Monitoring (BM)* period. Such a period started once we chose a new sensing deployment. We masked this period with data from the characteristic of the new sensor for all three approaches and showed the simulation values with dashed lines. The second gray oval indicates the sensing deployment switching time for AQuA-CEP. Again, we showed the simulation results with the red dashed line. We cannot rely on this window information because the window comprises events that partially belong to the previous sensor, and the rest belongs to the newly selected sensor until the BM period is ended. We can not perform adaptations within this period since the outputs are unreliable. Therefore, the event loss rate results can go above the predefined threshold, and no sensing deployment switching would be triggered.

Hence, a lower number of BM periods results in a higher percentage of query duration being monitored. Having this fact in mind, AQuA-CEP achieves better results than the ODLR since

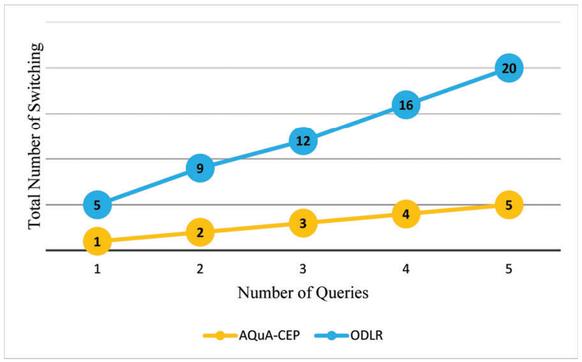


Figure 7: Switching counts between sensing deployments for different sets of queries with the static quality policy.

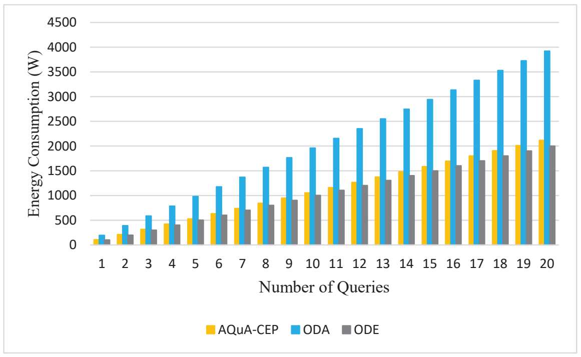


Figure 9: Total energy consumption for different sets of queries enriched by the dynamic quality policy.

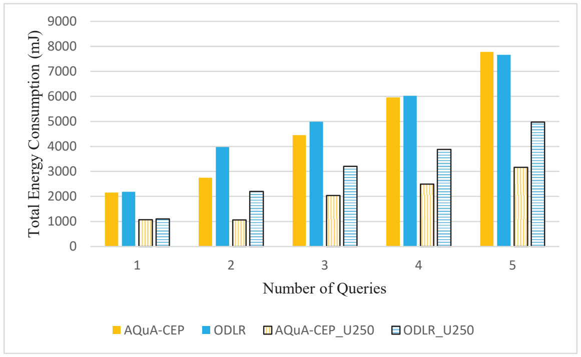


Figure 8: Total energy consumption for different sets of queries with the static quality policy.

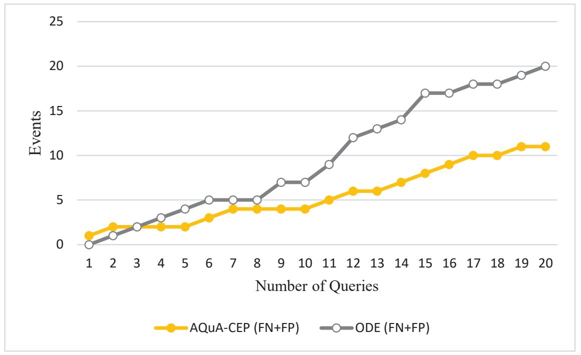


Figure 10: Sum of FP and FN for different sets of queries enriched by the dynamic quality policy.

it has fewer switching counts. To better portray this advantage of AQuA-CEP, more queries are involved in the comparison, and the results for the number of sensing deployment switching are depicted in Figure 7. The chart shows that the difference between AQuA-CEP and ODLR is escalated considerably once more queries come to the system. That means the number of BM periods is increased significantly, leading to less reliability in quality monitoring that also influences the adaptation decisions.

In addition, a higher number of sensing deployment switching results in taking more actions to activate or deactivate sensors. This means that in the ODLR approach, more time overhead is imposed due to stopping the analysis of event streams and monitoring the event quality on the previous sensor. Moreover, initializing these two functionalities over the event stream of the new sensing configuration increases the time overhead.

From the energy consumption point of view, there is also a remarkable dissimilarity between these two approaches exhibited in Figure 8. The graph shows that the amount of energy consumed in AQuA-CEP is less than the ODLR approach. From four queries onward, the total consumed energy seems equal. The reason is one or more queries in ODLR stopped processing; thereby, the consumed energy for them was equal to zero. On the other hand, queries in AQuA-CEP keep being answered until the end of simulation time. To be fair, we illustrated the results for both approaches until 250 sec of execution, when all queries were still active. Hence, the consumed energy for all sensors in AQuA-CEP is dramatically lower than the ODLR approach, respectively, proving the ability of our approach to optimize the server-side costs. It also can be seen that the difference between the two approaches is increasing by involving more queries in the execution. This confirms that AQuA-CEP is more reliable than ODLR in dealing with involving more queries.

6.2.2 ***Dynamic Quality Policy.*** Figures 9 and 10 exhibit the simulation's outcome in the case of applying dynamic quality policy over the query processing.

With energy consumption in mind as the comparison parameter, one can observe from Figure 9 that there is a dramatic distinction between AQuA-CEP and ODA. The rationale is ODA prefers the LTE sensor at all times for location tracking since it delivers the most accurate results while consuming the highest amount of energy among all sensing deployments. From this point of view, it can be concluded that employing the ODA mechanism can quickly lead to the phone's battery exhaustion. On the other hand, although the ODE approach is assumed to be the optimal approach in terms of energy, it consumes slightly less energy than AQuA-CEP. That's why we can claim the performance of AQuA-CEP regarding energy consumption is near-optimal. The discrepancy between the results of the ODE approach and our mechanism is increasing slightly by initiating more queries, but it is still negligible.

In event-based systems, False Positives (FP) and False Negatives (FN) are widely used to compare detection accuracy. In our example, an FN denotes a violation by entering a red-flagged area that occurred in the real world, but the event processing system could not catch it. Besides, an FP indicates a violation wrongly detected by the system. Since both of these errors are feasible in our use case with small counts, we form a single number of their summation that makes the differences more distinguishable, as illustrated in Figure 10.

Since ODA permanently answers queries with the most accurate sensors, it serves better than other methods. It could catch all the complex events without any FP or FN, but with the cost of draining the phone's battery. With fewer queries, ODE acts nearly the same as AQuA-CEP. But once more queries are involved, the situation becomes worse for this approach. It probably happens because of the lower detection capacity of data sources with minor energy consumption when the dangerous area is located in their close vicinity. The sensors with less energy consumption level have less room for consumers and might deliver less accurate data leading to more FPs and FNs.

Similar to FN and FP, *F-score* is a well-known performance measure in machine learning approaches that combines the other two measures, *precision* and *recall*, which are mainly employed to distinguish between classifiers [18] in terms of accuracy. Therefore, F-score can be used in stream processing to compare mechanisms in terms of accuracy. Analyzing the F-score shows an *ascending trend* in the reports, while the outcome for ODE displays a *descending trend*. This proves the ability of AQuA-CEP to deal with involving more queries while maintaining the accuracy of event detection. The possible reason is that involving more queries increases the overlap in data sources; thereby, multiple queries can benefit from our proposed switching mechanism. Consequently, the F-score values for our approach become better.

7 RELATED WORK

Responding to environmental dynamics is highly important while using DCEP systems in IoT scenarios for maintaining the QoR and QoS at a satisfactory level based on consumers' quality requirements. To select the suitable sensing deployments for each query, AQuA-CEP performs a global optimization in sensing deployment configuration that takes into account the consumer-side constraints as their quality requirements and monitors the quality of produced events to assess and make adaptation decisions to maintain the quality of the produced outcomes. In this section, we compare our work to the related work in two key areas, sensor selection and quality monitoring.

7.1 Sensor Selection

In the context of IoT networks, the Sensor Selection Problem (SSP) is a leading research direction to select the best set of sensors to achieve energy efficiency due to power limitations in sensor nodes [12, 34, 35]. More precisely, the selection in SSP-oriented research works is performed by picking a set of homogeneous sensors and executing the sensor aggregation. In the context of stream processing (e.g., DCEP systems), most runtime adaptive approaches concentrate on the operator networks, including adaptation on topology, deployment, processing, overload, fault tolerance, and infrastructure [6]. There are a few approaches similar to AQuA-CEP which are called *data source switching* mechanisms [5, 16] and most of them only applied to video streaming applications [25].

Although both of the mentioned related approaches study the dynamic selection of sensors in an IoT environment to optimize the QoR, such optimizations are performed respecting specific data attributes or a set of specific fused sensor sources. However, integrating such methods in the context of DCEP requires linking them dynamically to different configurations of heterogeneous sensors. Only in this way, the flexibility of current DCEP systems in reconfiguring and rewriting the detection logic of complex events can be used to optimize for QoR.

7.2 Quality Monitoring

In DCEP systems, quality assessment has been primarily studied in the placement of detection logic over the available computing resources (e.g., [22]), and only a few research works focus on the adaptation of sensing deployment to react dynamics. Although these mechanisms are quite similar in performance to our proposed approach, they are focused only on one aspect of the system (e.g., CEP query language in [33]).

In the IoT environment, the *service composition* mechanisms consider the sensory data streams as services provided by the connected objects to be analyzed and deliver results to the corresponding applications and allow the interaction between consumers and smart objects of IoT environment [2, 11, 15]. Considering the vulnerability of IoT service quality to environmental dynamics, service composition techniques try to specify a set of quality metrics to analyze the quality of delivered streams to target applications. By employing heuristic and metaheuristic techniques (e.g., [27]), several approaches attempt to find a global service composition solution while fulfilling QoS demands. These mechanisms tend to monitor and assess the quality of IoT services and adapt the system to maintain the quality, e.g., by training the Hidden Markov Models (HMM) to predict QoS. However, their quality expressivity is limited to defining the static thresholds causing inflexibility in acquiring more complex consumers' quality requirements. Moreover, AQuA-CEP is more efficient in energy consumption since it endeavors to minimize the number of active sensing deployments and eventually fulfills the requested quality requirements.

8 CONCLUSION AND FUTURE WORK

In this work, we proposed AQuA-CEP, which represents how to enable dynamic adaptation of sensing deployment configuration while observing the quality of produced events *and* their data sources. In addition, by proposing optimization criteria for the dynamic activation of sensors, our mechanism can help save resources in the sensing infrastructure. Our evaluation results demonstrated that AQuA-CEP outperforms two baseline approaches in switching counts between sensing deployment and performed near-optimal concerning the total energy consumed by the sensing network in static quality policy scenario. Moreover, by applying a dynamic quality policy, AQuA-CEP achieved near-optimality in terms of energy consumption and quality measured in the form of FP and FN. Moreover, the F-score results proved that AQuA-CEP has sufficient capabilities to fulfill consumers' quality requirements when more queries are involved.

In our future work, we will consider *priority* in the quality policy definition and investigate its impacts to support concurrent queries. Estimating the switching overhead is another point of interest that requires further research. In addition, minimizing the blind monitoring periods can be attainable by predicting the data source switching time. We believe building on a statistical analysis of the data sources' performance will be a promising direction. Finally, we plan to extend our proposed research by taking into account dynamic factors like quality degradation of data sources over time in dynamic quality policies.

ACKNOWLEDGEMENTS

This work has been supported by the Parrot project (Research Council of Norway, project number 311197). The authors would like to thank the four anonymous reviewers for their valuable feedback on this work, and Saad Saleh and Bochra Boughzala for their contributions to the review process.

REFERENCES

[1] K. Aberer, M. Hauswirth, and A. Salehi. 2006. *The Global Sensor Networks middleware for efficient and flexible deployment and interconnection of sensor networks.* Technical Report.

[2] A.N. Abosaif and H.S. Hamza. 2020. Quality of service-aware service selection algorithms for the internet of things environment: A review paper. *Array* 8 (2020), p. 100041.

[3] A.I. Arafat, T. Akter, M.F. Ahammed, M.Y. Ali, and A.A. Nahid. 2020. A dataset for the Internet of Things based fish farm monitoring and notification system. *Data in Brief* 33 (2020), p. 106457.

[4] P. Arcaini, E. Riccobene, and P. Scandurra. 2015. Modeling and Analyzing MAPE-K Feedback Loops for Self-Adaptation. In *Proceedings of 10th International Symposium on Software Engineering for Adaptive and Self-Managing Systems.* IEEE/ACM, pp. 13–23.

[5] P. Asghari, A.M. Rahmani, and H.H.S. Javadi. 2018. Service composition approaches in IoT: A systematic review. *Journal of Network and Computer Applications* 120 (2018), pp. 61–77.

[6] V. Cardellini, F. Lo Presti, M. Nardelli, and G.R. Russo. 2022. Runtime Adaptation of Data Stream Processing Systems: The State of the Art. *ACM Computing Surveys (CSUR)* 54, 11s (2022), pp. 1–36.

[7] Apache Flink Community. 2017. *FlinkCEP.* Retrieved February 9th, 2023 from https://nightlies.apache.org/flink/flink-docs-master/docs/libs/cep/.

[8] Oracle Corporation. 2008. *Oracle VM VirtualBox Manager.* Retrieved February 9th, 2023 from https://www.oracle.com/nl/virtualization/technologies/vm/downloads/virtualbox-downloads.html.

[9] L. Corral, A.B. Georgiev, A. Sillitti, and G. Succi. 2013. A method for characterizing energy consumption in Android smartphones. In *Proceedings of the 2nd International Workshop on Green and Sustainable Software (GREENS).* IEEE, pp. 38–45.

[10] Apache Software Foundation. 2011. *Apache Kafka.* Retrieved February 9th, 2023 from https://kafka.apache.org/

[11] F. Gao and E. Curry. 2020. Quality of Service-Aware Complex Event Service Composition in Real-time Linked Dataspaces. In *Real-time Linked Dataspaces.* Springer, pp. 169–190.

[12] Y. Gao, M. Diao, and T. Fujii. 2019. Sensor Selection Based on Dempster-Shafer Evidence Theory under Collaborative Spectrum Sensing in Cognitive Radio Sensor Networks. In *Proceedings of 90th Vehicular Technology Conference (VTC2019-Fall).* IEEE, pp. 1–7.

[13] M.N. Halgamuge, M. Zukerman, K. Ramamohanarao, and H.L. Vu. 2009. An Estimation of Sensor Energy Consumption. *Progress In Electromagnetics Research B 12* (2009), pp. 259–295.

[14] I. Kolchinsky and A. Schuster. 2018. Efficient Adaptive Detection of Complex Event Patterns. *VLDB Endowment* 11, 11 (2018), pp. 1346–1359.

[15] Ş. Kolozali, M. Bermudez-Edo, N. FarajiDavar, P. Barnaghi, F. Gao, M.I. Ali, A. Mileo, M. Fischer, T. Iggena, and D. Kuemper. 2018. Observing the Pulse of a City: A Smart City Framework for Real-Time Discovery, Federation, and Aggregation of Data Streams. *Internet of Things Journal* 6, 2 (2018), pp. 2651–2668.

[16] S.K. Lee, S. Yoo, J. Jung, H. Kim, and J. Ryoo. 2015. Link-Aware Reconfigurable Point-to-Point Video Streaming for Mobile Devices. *ACM Transactions on Multimedia Computing, Communications, and Applications* 12, 1 (2015), pp. 1–25.

[17] C.T. Li, J.C. Cheng, and K. Chen. 2020. Top 10 technologies for indoor positioning on construction sites. *Automation in Construction* 118 (2020), p. 103309.

[18] L. Li, B. Zhong, C. Hutmacher Jr, Y. Liang, W.J. Horrey, and X. Xu. 2020. Detection of driver manual distraction via image-based hand and ear recognition. *Accident Analysis & Prevention* 137 (2020), p. 105432.

[19] M. Lotfian Delouee, B. Koldehofe, and V. Degeler. 2022. Towards adaptive quality-aware Complex Event Processing in the Internet of Things. In *Proceedings of the 18th International Conference on Mobility, Sensing and Networking (MSN).* IEEE.

[20] M. Luthra, S. Hennig, K. Razavi, L. Wang, and B. Koldehofe. 2020. Operator as a Service: Stateful Serverless Complex Event Processing. In *Proceedings of International Conference on Big Data (Big Data).* IEEE, pp. 1964–1973.

[21] M. Luthra and B. Koldehofe. 2019. ProgCEP: A Programming Model for Complex Event Processing over Fog Infrastructure. In *Proceedings of the 2nd International Workshop on Distributed Fog Services Design.* ACM, pp. 7–12.

[22] M. Luthra, B. Koldehofe, P. Weisenburger, G. Salvaneschi, and R. Arif. 2018. TCEP: Adapting to Dynamic User Environments by Enabling Transitions between Operator Placement Mechanisms. In *Proceedings of the 12th International Conference on Distributed and Event-based Systems (DEBS).* ACM, pp. 136–147.

[23] M. Nardelli, V. Cardellini, V. Grassi, and F.L. Presti. 2019. Efficient Operator Placement for Distributed Data Stream Processing Applications. *IEEE Transactions on Parallel and Distributed Systems* 30, 8 (2019), pp. 1753–1767.

[24] C. Prud'homme and J.G. Fages. 2022. Choco-solver: A Java library for constraint programming. *Journal of Open Source Software* 7, 78 (2022), p. 4708.

[25] C. Qin, H. Eichelberger, and K. Schmid. 2019. Enactment of adaptation in data stream processing with latency implications—a systematic literature review. *Information and Software Technology* 111 (2019), pp. 1–21.

[26] R. Riesebos, V. Degeler, and A. Tello. 2022. Smartphone-Based Real-Time Indoor Positioning Using BLE Beacons. In *Proceedings of 18th International Conference on Automation Science and Engineering (CASE).* IEEE, pp. 1281–1288.

[27] S. Sefati and N.J. Navimipour. 2021. A QoS-Aware Service Composition Mechanism in the Internet of Things Using a Hidden-Markov-Model-Based Optimization Algorithm. *IEEE Internet of Things Journal* 8, 20 (2021), pp. 15620–15627.

[28] S. Singh, S. Kumar, A. Nayyar, F. Al-Turjman, and L. Mostarda. 2020. Proficient QoS-Based Target Coverage Problem in Wireless Sensor Networks. *IEEE Access* 8 (2020), pp. 74315–74325.

[29] A. Slo, S. Bhowmik, and K. Rothermel. 2020. HSPICE: State-Aware Event Shedding in Complex Event Processing. In *Proceedings of the 14th International Conference on Distributed and Event-based Systems (DEBS).* ACM, pp. 109–120.

[30] A. Slo, S. Bhowmik, and K. Rothermel. 2022. State-Aware Load Shedding From Input Event Streams in Complex Event Processing. *IEEE Transactions on Big Data* 8, 5 (2022), p. 1340–1357.

[31] E. Volnes, T. Plagemann, B. Koldehofe, and V. Goebel. 2022. Travel light: state shedding for efficient operator migration. In *Proceedings of the 16th International Conference on Distributed and Event-based Systems (DEBS).* 79–84.

[32] Y. Wang, S. Han, Y. Tian, C. Xiu, and D. Yang. 2020. Is Centimeter Accuracy Achievable for LTE-CSI Fingerprint-Based Indoor Positioning? *IEEE Access* 8 (2020), pp. 75249–75255.

[33] P. Weisenburger, M. Luthra, B. Koldehofe, and G. Salvaneschi. 2017. Quality-Aware Runtime Adaptation in Complex Event Processing. In *Proceedings of 12th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS).* IEEE/ACM, pp. 140–151.

[34] I. Younas and A. Naeem. 2022. Optimization of sensor selection problem in IoT systems using opposition-based learning in many-objective evolutionary algorithms. *Computers & Electrical Engineering* 97 (2022), p. 107625.

[35] J. Zhang, J. Du, and L.R. Dai. 2021. Sensor Selection for Relative Acoustic Transfer Function Steered Linearly-Constrained Beamformers. *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 29 (2021), pp. 1220–1232.

[36] L. Zou, A. Javed, and G.M. Muntean. 2017. Smart mobile device power consumption measurement for video streaming in wireless environments: WiFi vs. LTE. In *Proceedings of International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB).* IEEE, pp. 1–6.