Hindawi
Wireless Communications and Mobile Computing
Volume 2018, Article ID 8475604, 11 pages
https://doi.org/10.1155/2018/8475604

Research Article

A Service-Based Method for Multiple Sensor Streams Aggregation in Fog Computing

Zhongmei Zhang,[1,2,3] **Chen Liu**,[2,3] **Shouli Zhang**,[1,2,3] **Xiaohong Li**,[1] **and Yanbo Han**[2,3]

[1]*School of Computer Science and Technology, Tianjin University, Tianjin 300350, China*
[2]*Beijing Key Laboratory on Integration and Analysis of Large-Scale Stream Data, North China University of Technology, Beijing 100144, China*
[3]*Institute of Data Engineering, School of Computer Science and Technology, North China University of Technology, Beijing 100144, China*

Correspondence should be addressed to Zhongmei Zhang; gloria_z@126.com

Received 23 February 2018; Revised 12 April 2018; Accepted 19 April 2018; Published 28 May 2018

Academic Editor: Xuyun Zhang

A surge in sensor data volume has exposed the shortcomings of cloud computing, particularly the limitation of network transmission capability and centralized computing resources. The dynamic intervention among sensor streams also brings challenges for IoT applications to derive meaningful information from multiple sensor streams. To handle these issues, this paper proposes a service-based method with fog computing paradigm based on our previous service abstraction, which can capture meaningful events from multiple sensor streams. In our service abstraction, we utilize correlation analysis method to capture events as variations of correlation among sensor streams. Facing inconsistent frequency and shift of correlation, we propose a Dynamic Time Warping- (DTW-) based algorithm to obtain sensor streams' lag-correlation. For adaptively aggregating related events from different services, we also propose an event routing algorithm to assist the composition of cascaded events through service collaboration. This paper reports the tryout use of our method in Chinese power grid for detecting abnormal situations of power quality. Through a series of experiments based on real sensor data in power grid, we verified that our method can reduce the network transmission and computing resource with high accuracy.

1. Introduction

With the rapid development of Internet of Things (IoT), numerous sensors are deployed for event monitoring, ecological observation, and so forth, which produce an overwhelming amount of stream data [1]. Individual sensor streams are usually fine-grained, have relatively lower value density, and intervene with each other dynamically. For obtaining more valuable information, many IoT applications try to capture events from sensor streams with multiple sources in different ways [2]. The big real-time sensor streams lead to the emergency of a new computing paradigm, that is, fog computing. Compared with cloud computing, it does not require data to be stored and processed in a centralized way and can greatly lower the computation and storage costs of the cloud server through putting processing of sensor streams to the edge nodes.

Although fog computing has lots of advantages, its related researches have just started and are far from being mature. Most of current researches still focus on the architecture, wireless sensor networks, hardware, and underlying applications [3]. Few works attach enough importance on what software abstractions should be fit to put on a fog/edge node and how to program them. Actually, it is very important for an IoT application to build with the fog computing paradigm.

In an IoT application, sensor streams usually are not independent with each other. They need to cooperate for achieving complicated business goals. For capturing meaningful events from multiple sensor streams, event detection techniques [4–6] are already used in various real-time

enterprise solutions. State-of-the-art event processing models provide frameworks to represent and reason events. However, they require the detailed event definitions and work on fixed set of sensor streams [7]. Facing large amount of sensor streams, it is impractical to define all detailed definitions of each variety of events. Hence, it is crucial to adaptively choose relative stream sources and capture events without relying on predefined definitions.

Correlation analysis methods have traditionally been used to process data streams for classifying and identifying events. In many fields, such as anomaly detection and electronic trading, the variation of correlation among sensor streams can be regarded as one kind of abnormal events, which do not need detailed definitions on raw sensor streams. Meanwhile, for multiple sensor streams, the inconsistency of frequency and shift of correlation will affect the accuracy of correlation analysis and cause wrong identification of events.

A remarkable effort is to encapsulate sensor data into Web services, in which the open Web standards are supported for information sharing [8–10]. But ad hoc association and streaming-oriented queries can hardly be reflected. Following the thought of software-defined sensors, we proposed a service abstraction called ***proactive data service*** in our previous works [11, 12]. With proactive data service, the *sensor events* generated directly by sensors can be transformed into multiple *service events*, which reveal more meaningful information. In this paper, we integrate our service abstraction with fog computing paradigm. In our service abstraction, we utilize the correlation analysis method that works directly on sensor streams for capturing events as the variation of correlations. Facing the inconsistency of frequency and shift of correlation of different sensor streams, we propose a Dynamic Time Warping- (DTW-) based algorithm to obtain the lag-correlation and capture dynamic events.

With proactive data services, the sensor events generated directly by sensors can be transformed into multiple service events that reveal more meaningful information. Since the capturing of some complicated events depends heavily on the cooperation of different sensor streams, we propose an event routing algorithm for indicating the targets of certain service events and define a set of declarative rules in our service abstraction for triggering corresponding processing. Based on changeable event routing paths and declarative rules, proactive data services can adaptively and proactively collaborate with each other and generate more meaningful events from dynamic sensor streams.

Proactive data services can be deployed in edge nodes, do some preliminary analyses, and report events to cloud servers for further analyses. Relying on our DTW-based event capturing and event routing algorithms, we can capture events from multiple sensor streams and obtain complicated events through service collaboration. We preliminarily applied our method in a real-world scenario: power quality event detection for Chinese power grid; and we verified the feasibility of our method. Based on real sensor data from Chinese power grid, we evaluate the effectiveness and efficiency of our method through a series of experiments. The experimental results show that our method can greatly reduce the transformation of primitive sensor streams and improve the efficiency of event capturing with high accuracy.

2. Scenario

The Chinese power grid consists of various electric devices with complicated electrical connections. In most electric devices of Chinese power grid, a series of sensors are deployed to monitor indicators of power quality, such as voltage, current, and frequency. Due to the relationship of electric device and disturbance source, the devices can be divided into two types: disturbance source device and general device. A disturbance source device is deployed close to certain disturbance source and is directly influenced, while a general device is indirectly influenced by the disturbance source. Each sensor is in charge of monitoring one indicator and produces corresponding sensor stream. These sensor streams are affected by various disturbance sources, such as wind farm, photovoltaic power station, and electrified railway and show some abnormal status; that is, originally correlated sensor streams become uncorrelated or out of limit. A key problem of a power grid is to capture abnormal events of power quality based on these sensor streams.

To users, the primitive sensor streams are somewhat meaningless and unreadable. It is crucial to increase the value density of sensor streams, that is, to capture abnormal events from primitive sensor streams. An abnormal power quality event caused by certain disturbance generally involves multiple devices, and the sensor streams in each device show the same abnormal status (i.e., involving the same set of sensor streams). Figure 1 shows an example of abnormal power quality events, in which each column indicates a sensor stream, and the darker columns indicate abnormal and light columns indicate normal. As shown in Figure 1, the event involves multiple sensor streams from one disturbance source device and 3 general devices, and for different devices, the sensor streams that indicate the same indicators are involved. For single device, the details of involved sensor streams are shown in Figure 2.

As shown in Figure 2, the three phase voltages ($stream_1$, $stream_2$, and $stream_3$) in power quality are generally correlated, while two trains are passing by an electrified railway; the correlation among these three streams is changed, and an abnormal event "unbalanced three phase voltages" occurs in the electric device. For each device, it is challenging to capture events from multiple sensor streams, because the correlation among them may be lagged. Figure 2 also shows some examples of lag-correlation; the trends of rising and falling of $stream_4$ and $stream_5$ are similar but not synchronous strictly. And the frequency and timestamps of data records in different sensor streams are different. For example, $stream_2$ and $stream_4$ have lower frequency than other streams. There could be error correlation analysis if we do not consider these issues.

It is obvious that the sensors in each device and the devices involved in abnormal power quality events are changeable because of the influence of various disturbance sources. The sources of sensor streams for generating

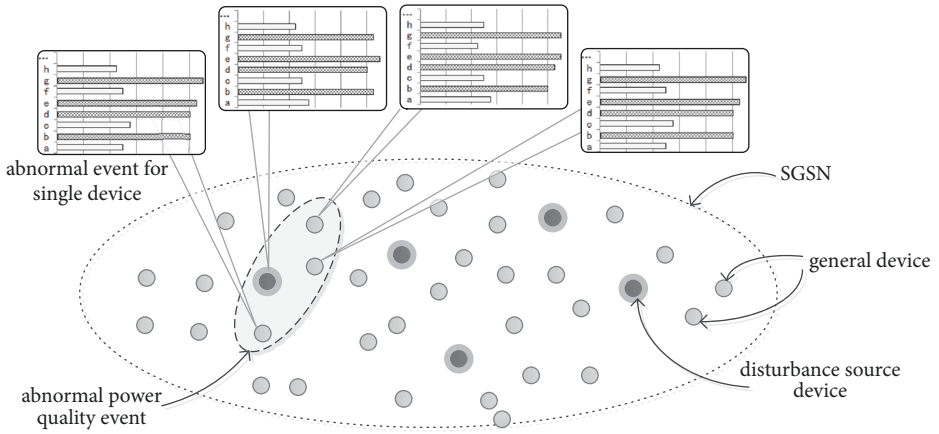

Figure 1: An example of abnormal power quality event.

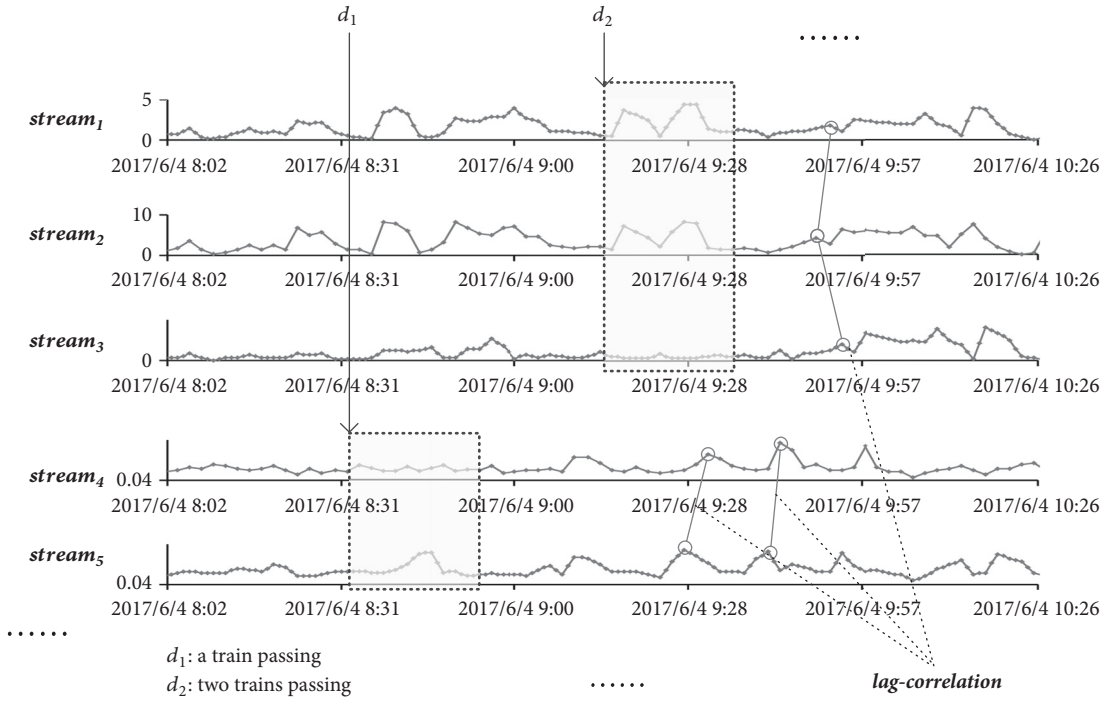

Figure 2: Examples of disturbances' affection in sensor streams.

meaningful events are uncertain. Hence, for deriving meaningful event, we need to adaptively choose related sensor stream sources, that is, sensors in single device and electric devices, and capture abnormal events for each device and aggregate abnormal events of multiple devices for obtaining final events.

Because of uncertain stream sources of events, existing methods need to collect and analyze all sensor streams in cloud server for generating events. The analysis of all sensor streams will cause vast and unnecessary consumption of computational and communication resources and cannot guarantee the requirement of timeliness. In this paper, we aim to utilize the fog computing paradigm and deal with two problems: how to capture events more accurately through considering the characters of sensor streams and how to adaptively choose relative sensor streams relying on different disturbances.

3. Overview of Our Service-Based Method

Figure 3 shows the rationale of our service-based method for dynamically aggregating multiple sensor streams and capturing meaningful events. Based on our previous proactive data service model, we can create services whose basic function is to generate events (regarded as service events in the proactive data service model) from certain set of primitive sensor events. Each proactive data service can be deployed in edge nodes and do preliminary processing for

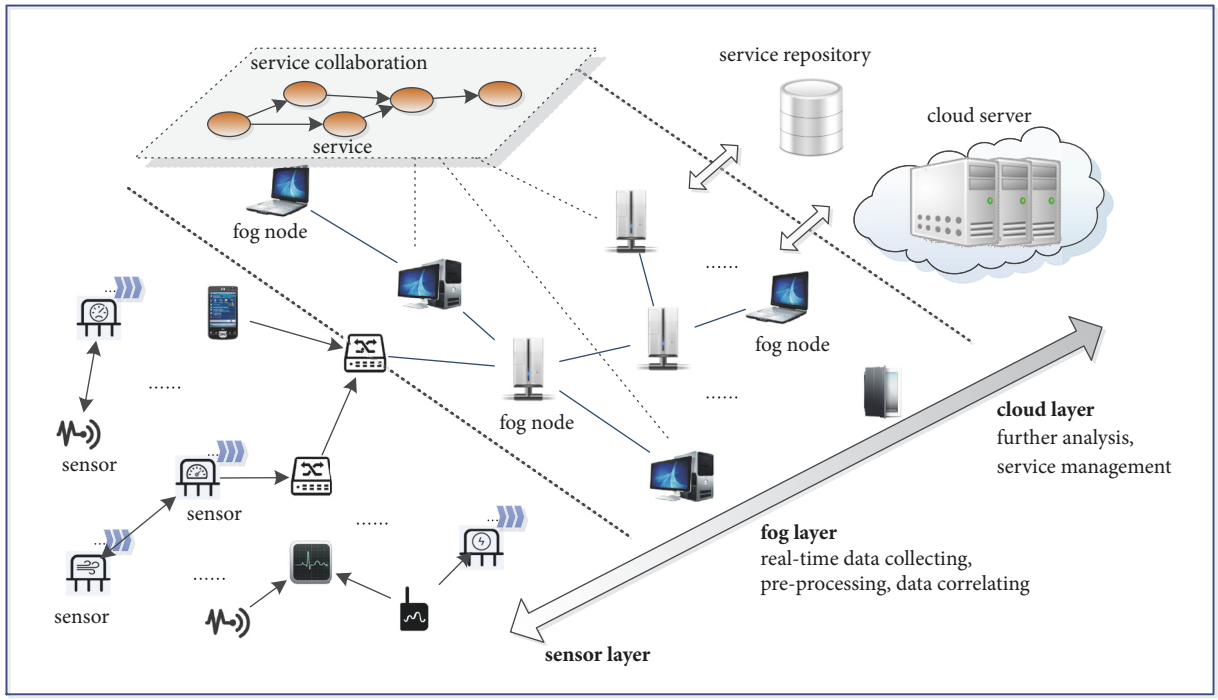

Figure 3: Rationale of our service-based method.

increasing value density of sensor streams. In this case, only meaningful events are transmitted to cloud server. For generating more meaningful and complicated events involving dynamic sources, the proactive data service also supports routing its service events to related services and adaptively collaborates with other services. Based on proactive data services, the fog layer can provide abilities such as real-time data accessing, preprocessing, and sensor data correlating. The detailed information of services can be stored in cloud service for service discovering.

A sensor stream can be defined as follows.

Definition 1 (sensor stream). A sensor stream can be represented as $ssd = \langle source_{id}, A, R\rangle$, in which $source_{id}$ is the id of stream source, A are the attribute sets, and $R = \{r_1, r_2, \ldots, r_i, \ldots\}$ is an infinite series of data record $r = \langle\{\langle a_i, v_i\rangle \mid a_i \in A\}, t\rangle$which is a set of key-value pairs with a timestamp.

Typically, events are of a certain type, have a timestamp, and hold specific data. Many event detection technologies utilize event patterns on sensor streams to represent events. For example, a set of predicates are utilized to describe the event pattern in [13]. In this paper, we regard the correlation's variations among multiple sensor streams as events. Hence, we can utilize a set of sensor sources to represent a primitive event pattern. And a more complicated event can be composited by multiple primitive events and represented as a composite event pattern $p_{com} = \langle\{source_{id}\}, R\rangle$, in which $\{source_{id}\}$ indicates the sensor stream sources in one primitive event pattern and R is a set of relations between the primitive event patterns. For example, an abnormal event "unbalanced three phase voltages" occurring in single device can be represented by three streams indicating three phase voltages, and an integrated abnormal power quality event can be composited by several "unbalanced three phase voltages" events in multiple devices.

We regard each record with its source and timestamp in sensor stream as a sensor event and abstract the ability of event processing as a proactive data service, which can access and process multiple sensor streams and generate service events with higher value. The services can communicate with each other and provide service events to cloud server based on their *uris*. Figure 4 shows the structure of our proactive data service model.

Definition 2 (proactive data service). A proactive data service can be formalized as $pds = \langle pds_{id}, event_{in}, event_{out}, DR, EP, hyperlinks\rangle$, in which pds_{id} is the unique identifier, $event_{in}$ represents the input event streams, $event_{out}$ represents the output event streams generated by EP, DR are the declarative rules, EP means the event processing function, and the *hyperlinks* are an optional parameter that indicates the targets of service events.

Presently, we utilize Pearson's correlation coefficient (PCC) [14] to measure the sensor streams' correlations for capturing the change of correlation as event. The correlation among sensor data usually shifts in time, which can be regarded as lag-correlation [15]. We consider the lag-correlation analysis problem as to find the time lag vector to maximize the PCC and proposed a DTW-based event capturing algorithm in our service.

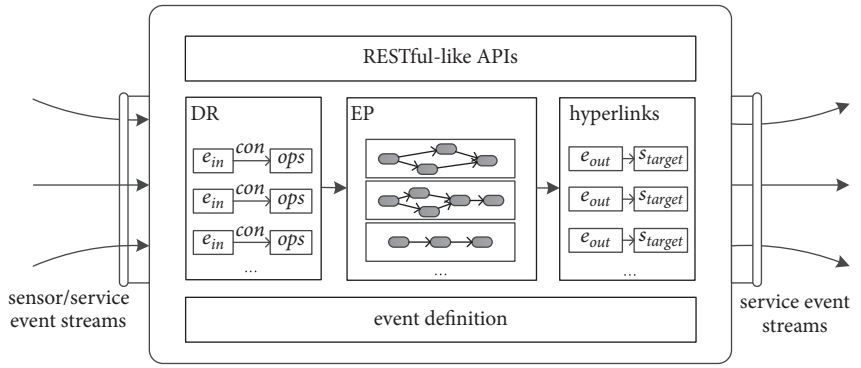

Figure 4: The structure of the proactive data service.

For supporting dynamic service collaboration, we propose an event routing algorithm for setting the hyperlinks in service, which can indicate the target services for certain service event. And we also utilize declarative rules in service to handle external events. In this paper, we define declarative rules based on Event-Condition-Action (ECA) rules, which are originally used in active database systems to provide event-driven, instantaneous responses for conventional database systems [16]. In this way, the entire process requires no intervention from users or external applications.

4. Event Capturing in Proactive Data Service

The proactive data service is created based on fixed set of sensor streams. We calculate the correlations between each two sensor streams based on time-based slide window, and once any two sensor streams' correlation is changed, an event is through to be captured. The pattern of captured event can be represented by the set of sensor streams whose correlation is changed.

When analyzing correlation among sensor data, misjudgment of correlation will also occur if correlation lags are not considered. The lag-correlation analysis problem can be considered to find the time lag vector to maximize the PCC, which can be formalized as follows.

Definition 3 (lag-correlation analysis problem). Given two sensor data sets E_i and E_j, suppose that $E_i = \{e_{i\_t1}, e_{i\_t2}, \ldots, e_{i\_tn}\}$ and $Ej = \{e_{j\_t1}, e_{j\_t2}, \ldots, e_{j\_tn}\}$ in a slide window; if there exists a time lag vector $\Delta = \{t_1', t_2', \ldots, t_n'\}$, one has

$$\mathrm{MAX}\left(cor_{(E_i,E_j,\Delta)}\right) = \frac{\sum_1^n \left(e_{i\_t} - \overline{e_i}\right)\left(e_{j\_(t+tk')} - \overline{e_j}\right)}{\sqrt{\sum_1^n \left(e_{i\_t} - \overline{e_i}\right)^2} \times \sqrt{\left(e_{j\_(t+tk')} - \overline{e_j}\right)^2}} \quad (1)$$

To analyze the correlation between two sensors' data, we can firstly obtain a time lag vector that makes $\mathrm{MAX}(cor_{E_i,E_j,\Delta})$ and then justify the lag-correlation based on the calculated coefficient. In this paper, we calculate the time lag vector making the minimum Euclidian distance of the normalized series and then calculate the PCC of the original series based on the time lag vector.

DTW algorithm [17] is a robust method used to measure similarity of time series, which can shift and distort the time series to ignore the problem of time axis scaling and shifting. In this paper, we adopt a DTW-based algorithm to find the time lag vector of two sensor data series. To ignore the problem of amplitude scaling, we align sensor data series with DTW algorithm based on normalized sensor data series. To avoid excessive time warping, we set a maximum value for the time lag vector.

Because of the inconsistent frequency of different sensor streams, the numbers of their data records in the same window may be different, and their PCC cannot be calculated directly. Hence, we firstly utilize linear interpolation method to increase the data records of the sensor stream with lower frequency. Then we utilize DTW algorithm to align the data series and calculate the correlations. The pseudocode of our event capturing algorithm executed in each window is shown in Algorithm 1 .

As shown in Algorithm 1, after choosing sources of sensor streams, we can obtain a set of data series. We firstly get all pairs of data series for calculating the correlation between each pair (line 1). For the data series pair that has different numbers of records, we utilize interpolation method to increase the number of data records for the shorter one (line 7). In this way, these two data series could have the same number of data records. Then we utilize DTW algorithm to align the data series for analyzing the lag-correlation (lines 12 and 13). At last, we verify the difference of current correlation coefficient and previous coefficient; if the difference is bigger than given threshold, an event will be captured and the execution in this window is ended.

5. Dynamic Collaboration among Services

With our proactive data service model, we can create one proactive data service for each electric device. Based on our event capturing algorithm, we can capture the abnormal status of each device. In this section, we will introduce the dynamic service collaboration for correlating sensor streams

Input:
$\boldsymbol{E} = \{E_1, E_2, \ldots E_k\}$: a set of data series in the same window;
δ_{cha}: the change threshold of PCC;
limit: the maximum value for the time lag vector;
output:
event;
(1) obtain all pairs $\{\langle E_i, E_i\rangle\}$in $\boldsymbol{E}$;
(2) for each $\langle E_i, E_i\rangle$
(3) if E_i and E_j have different numbers of records
(4) $d \leftarrow$ the difference of their records' numbers;
(5) $E_m \leftarrow$ getLongerSeries(E_i, E_j);
(6) $E_n \leftarrow$ getShorterSeries(E_i, E_j);
(7) $E_n' \leftarrow$ interpolation(E_n, d);
(8) else
(9) $E_m \leftarrow E_i$;
(10) $E_n' \leftarrow E_j$;
(11) end if
(12) $\langle E_m', E_n''\rangle \leftarrow$ aligned data series from $\langle E_m, E_n'\rangle$ based DTW
(13) calculate correlation coefficient *cor* of $\langle E_m', E_n''\rangle$;
(14) if $|cor\text{-}cor_{previous}| >= \delta_{cha}$
(15) an *event* is captured;
(16) break;
(17) end if
(18) end for

Algorithm 1: Event capturing algorithm.

from dynamic devices and obtaining the abnormal power quality events.

Each collaboration procedure is beginning with a proactive data service corresponding to one disturbance source device. When a service event is generated, it will be routed to related services and triggers corresponding processing of events. Since the collaboration is emanative from disturbance source service, we set up one event detection service (ED service) for each disturbance source for collecting the collaboration results and obtaining the final abnormal power quality events.

5.1. The Event Routing Algorithm. Since there is no electrical connection information, we propose an algorithm through considering service correlation relying on both the location of physical devices and existing collaboration results. The algorithm has two phases. In the initial phase, we regard all proactive data services near to given one as its related services based on physical devices' location. In the runtime phase, we keep updating the correlation of proactive data services based on their collaboration results for certain event pattern.

Specifically, we regard the neighbour proactive data services as the total set of routing targets $S_{possible}$ for given service. We assume that devices frequently cooccurring in power quality events are very likely to be involved in an event again. Hence, we will keep routing events to the successful collaborated set S_{valid} and reduce the possibility to route events to the failure collaborated set $S_{invalid}$. Because of the dynamic nature of events, devices that did not cooccur in one power quality event still have possibility to form a power quality event. Hence, the remaining service set $S_{alternative}$ in $S_{possible}$ excepting the actual target set $S_{practical}$ also needs to be considered.

We set up weight ω_i for each proactive data service s_i in $S_{possible}$ and obtain the actual target set $S_{practical}$ through comparing ω_i with a given threshold δ. In procedure 1, we firstly set up the weight of all proactive data services in $S_{possible}$ as δ and regard $S_{possible}$ as actual targets $S_{practical}$ in default. Any events generated by a disturbance source service will be sent to their related services. Algorithm 2 shows the pseudocode of the second phase in our algorithm.

In procedure 2, we update the weights of target proactive data services for certain service and certain event pattern based on newly received collaboration result, and because the target pattern is the same as source pattern, we then can obtain the declarative rules. After each event routing, we increase target proactive data services' weights of valid routing paths and reduce the weights of invalid paths. Meanwhile, we also increase the weight of services in $S_{alternative}$ for considering their possibility. Through comparing the new weights with given threshold, we will obtain new $S_{practical}$ and update the event handlers and links in the proactive data service.

5.2. Case Study. Taking the detection procedure of an abnormal power quality event caused by *wind farm* as an example, this section describes our service-based method. Figure 5 shows the service collaboration procedure. Due to limited space, only parts of proactive data services are shown. We create one proactive data service for each electric device and

```
(1)  while receiving collaborate result
(2)    extract source service s_i;
(3)    if the result is a composed event
(4)      extract the valid routing path ⟨pattern, s_ij⟩;
(5)      s_ij.weight = s_ij.weight+ α;
(6)    else
(7)      extract the invalid routing path ⟨pattern, s_ij⟩;
(8)      s_ij.weight = s_ij.weight- β;
(9)      if s_ij.weight< δ
(10)       remove s_ij from s_ipractical;
(11)     end if
(12)   end if
(13)   for each s_ij in s_alternative
(14)     s_ij.weight = s_ij.weight+γ;
(15)     if s_ij.weight ≥ δ
(16)       add s_ij in s_ipractical;
(17)     end if
(18)   end for
(19) end while
```

Algorithm 2: Updating the routing targets.

deploy it in edge node. The event detection service can be deployed in cloud server to collect the collaboration results. Proactive data service A is corresponding with a disturbance source device, whose event patterns, output events, and routing paths can be set up at design time. Other proactive data services are corresponding with general devices, and their pattern needs to be matched and output events and routing paths are updated at runtime.

As mentioned before, each proactive data service can capture abnormal events for single device, which can be represented by a set of sensor streams, while an integrated abnormal power quality event can be captured by multiple services, in which each service indicates one device. Table 1 shows the details of some proactive data services in the collaboration procedure.

The collaboration is beginning with proactive data service A; when an event e_1, that is, the correlation among $\{a, b, c\}$, is generated, A sends it to service B, C, and the ED service. When B and C receive this event, they will calculate the correlation among $\{a, b, c\}$, and events generated by B and Cwill be composed with event received from A to generate more meaningful events $\langle\{A, B\}, \{a, b, c\}\rangle$ and $\langle\{A, C\}, \{a, b, c\}\rangle$. The new events will be continuously routed. Since F does not generate new event after receiving event from C, it sends a failure message to ED service and stops collaborating with other proactive data services. Event generated by B is sent to D and E, and only D generates new event and sends it to G and H. Since both G and H fail to collaborate with D, the whole collaboration is finished, and a power quality event involving proactive data services A, B, C, D is generated.

At each step of the collaboration, the ED service collects collaboration results and updates corresponding routing paths. For proactive data service A and event e_1, routing paths $e_1 \rightarrow \{B, C\}$ are valid, so the weights of targets will be increased and the routing paths can be kept. For service B and event e_2, the routing path $e_2 \rightarrow E$ is invalid, so the weight of E will be decreased and the routing path may be removed. Besides, there may be proactive data services that are not in the routing paths, while they are neighbours of certain proactive data service. Their weight will also be increased, so they can be considered for detecting dynamic events.

Table 1: The details of proactive data services in our scenario.

ID	*Pattern*	$event_{out}$	*Routing paths*	*Set-up time*
A	$\{a, b, c\}$	e_1: $\langle A, \{a, b, c\}\rangle$	$e_1 \rightarrow \{B, C\}$	*Design time*
B	$\{a, b, c\}$	e_2:$\langle\{A, B\}, \{a, b, c\}\rangle$	$e_2 \rightarrow \{D, E\}$	*Runtime*
C	$\{a, b, c\}$	e_3:$\langle\{A, C\}, \{a, b, c\}\rangle$	$e_3 \rightarrow \{F\}$	*Runtime*
D	$\{a, b, c\}$	e_4:$\langle\{A, B, D\}, \{a, b, c\}\rangle$	$e_4 \rightarrow \{G, H\}$	*Runtime*

6. Evaluation

In this section, we evaluate the effectiveness and efficiency of our proactive data services with our event capturing and routing algorithms. We built one proactive data service for each electric device and evaluated the average performance of the services.

6.1. Experiment Setup

Data Set. The data set used in our experiments is from real sensor data in Chinese power grid. We selected sensor data from 2017-05-01 to 2017-06-05. There are 2653 devices with 438 disturbance source devices. Each disturbance source device can cause 2–6 types of events for single device, and there are totally 1978 kinds of event patterns. And each electric device at most deploys 2051 sensors. We simulated

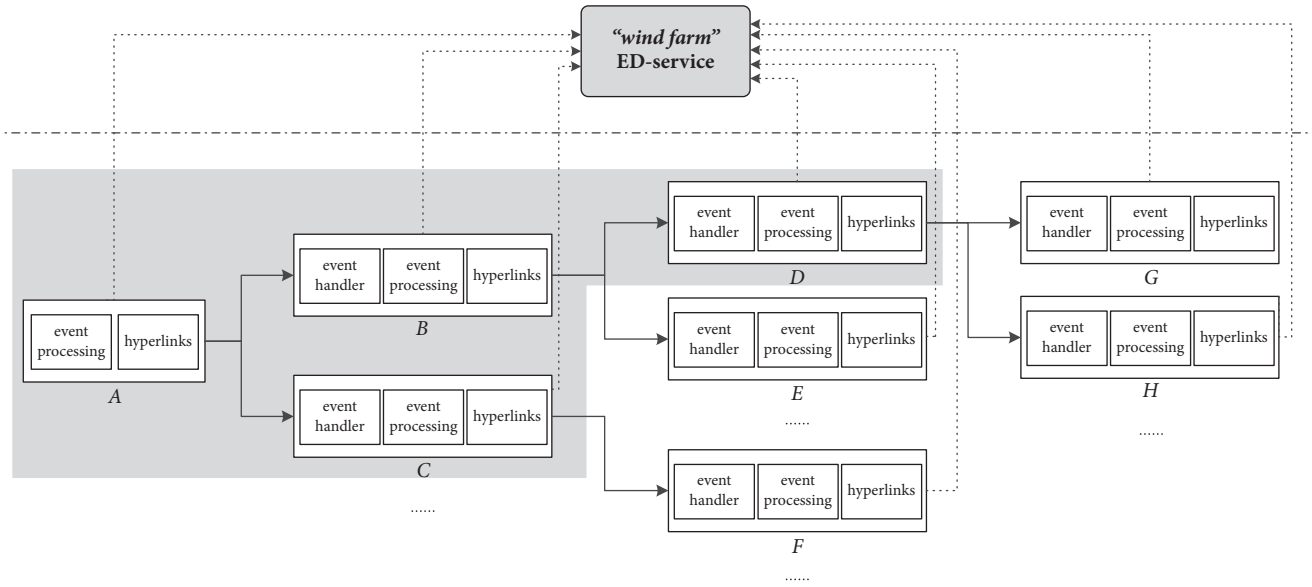

Figure 5: Collaboration procedure of abnormal power quality event detection.

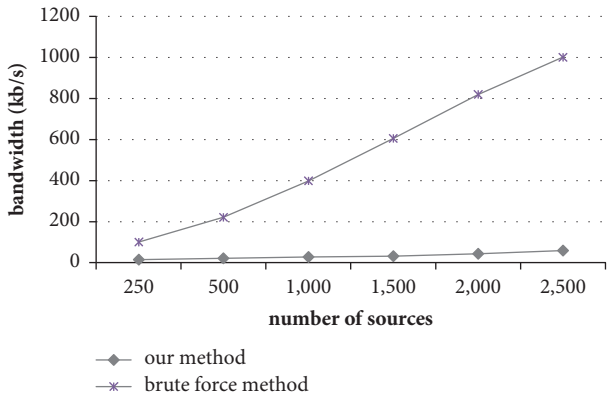

Figure 6: The bandwidth consumptions of different methods.

sensor stream for each sensor strictly according to the real timestamp.

Environment. We implemented our method in virtual machines with CentOS release 6.4, four Intel Core i5-2400 3.10 GHz CPUs and 4.00 GB RAM. All the algorithms are implemented in Java with JDK 1.8.0.

Evaluation Criteria. We design the following criteria:

(1) Bandwidth consumption: the bandwidth consumption means the data volume received by cloud server from stream sources

(2) System load: the system load of the proactive data service can be divided into the load of CPU and memory

(3) Event execution time: for each generated event, its execution time is the difference between the time when its corresponding sensor event is firstly received and the time when it is generated

(4) Accuracy: the accuracy includes precision and recall, which can be calculated as follows:

$$precision = \frac{|D \cap T|}{|D|} \times 100\%$$
$$recall = \frac{|D \cap T|}{|T|} \times 100\% \quad (2)$$

where $D = \{d_1, d_2, \ldots, d_m\}$ is detected event list and $T = \{t_1, t_2, \ldots, t_n\}$ is the actual event list.

6.2. Experiment Result and Analysis. Existing works barely focused on capturing events from fixed set of sensor streams and merely considered dynamic sensor streams correlation. At present, power quality detection system has been built to detect abnormal power quality events. Because the sources involved in power quality events are changeable, the existing method is kind of brute force method, which needs to collect and process all sensor streams in cloud server with a centralized manner. We utilize brute force method practically used as comparative and compare our method with different sensor stream sources, that is, device sets in power grid.

Figure 6 shows the bandwidth consumptions of different methods with different source sets. The brute force method needs to process all sensor streams, while in our method the disturbance source sensor stream only needs to match with corresponding patterns, and the processing of general sensor stream is relying on the processing results of previous service. In this case, only sensor streams from related sources are processed. Hence, the bandwidth consumption of our method is far less than existing brute force method. Because more events may occur with more sensor streams, the bandwidth consumption of our service abstraction has a slight increase with the growth of stream sources.

Figures 7 and 8 show the system loads and execution times of different methods with different stream sources. It

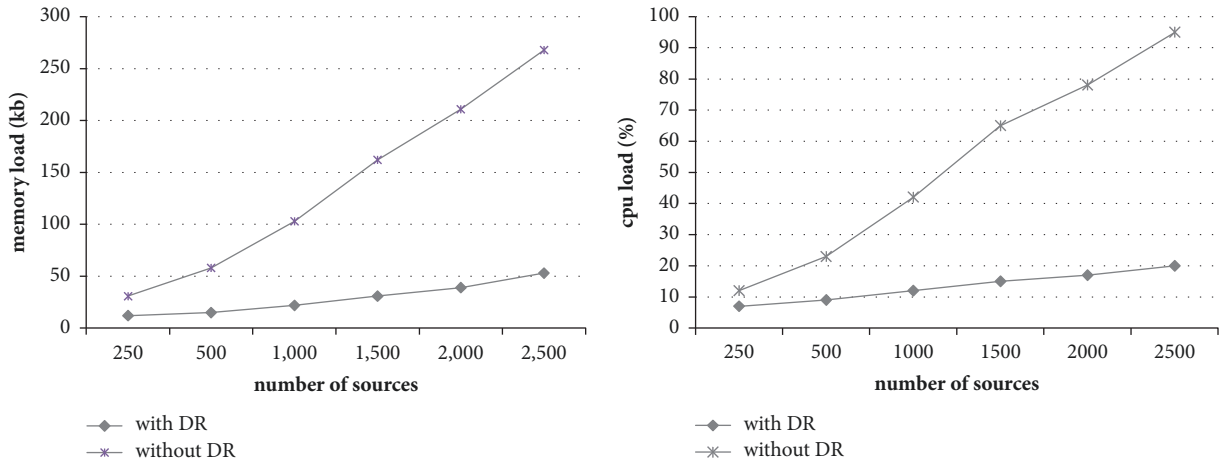

Figure 7: The system loads of different methods.

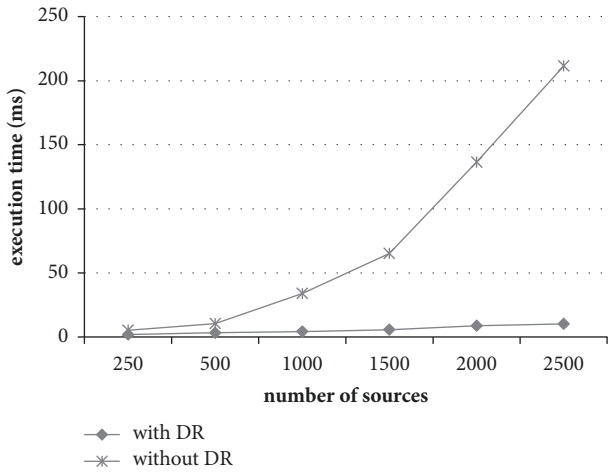

Figure 8: The execution times of different methods.

is obvious that our method has much lower system loads and less execution time, because the senor data that need to be processed are much less.

Then we evaluate our event capturing algorithm through calculating the precision and recall of the events capturing by our DTW-based algorithm and normal correlation analysis algorithm, which does not consider the inconsistency of frequency and shift of correlation among multiple sensor streams. For verifying the accuracy of our method, we analyzed all the sensor data and consulted domain expert and obtained 1783 abnormal events as the actual event list.

Figure 9 shows the average precision and recall of different event capturing algorithms with different stream sources. It is obvious that our event capturing algorithm has much higher precision and recall. This is because our event capturing algorithm adopted DTW-based correlation analysis algorithm and considers different frequency of sensor streams and the lag of correlation. As shown in Figure 9, our method has an average precision over 89% and average recall over 91%. By consulting domain experts, existing event capturing method in power grid has the precision and recall of about 60%–75%, which indicates that our method can increase the accuracy by about 20%.

In future, we aim to update the hyperlinks and declarative rules at runtime based on real-time analysis and learning and further improve the accuracy through considering the dynamic and intelligent correlations among events.

7. Related Works

Fog computing aims at bringing back partial computation load from the cloud to the edge devices. Gupta et al. [18] proposed a system, which abstracted connected entities as services and allowed applications to orchestrate these services with end-to-end QoS requirements. Researchers presented Vigil, a real-time distributed wireless surveillance system supporting real-time tracking and surveillance in enterprise campuses, retail stores, and across smart cities. Vigil utilized laptops as fog computing nodes between cameras and cloud to save wireless capacity [19]. Yuriyama and Kushida [20] virtualized a physical sensor as a virtual sensor on the cloud and automatically provided the virtual sensors on demand. Mohamed et al. [21] abstracted services and components involved in smart city applications as services accessible through the service-oriented model and enhanced integration and allow for flexible inclusion and utilization of the various services needed in a smart city application. In our research, we borrow ideas from Vigil [19] and utilize laptops as fog computing nodes. Each computing node can maintain multiple proactive data services and transmit generated event streams to the cloud server.

Many works utilized various event patterns to abstract events based on spatiotemporal relation of sensor data and transformed event detection problem into a pattern matching problem. A basic method is to detect event based on preset threshold value defined by domain expert [4]. To describe more complex event, Xue et al. [5] integrated the pattern-based approach with an in-network sensor query processing framework, which focused on the event patterns for single sensor and defined 5 common basic patterns to describe events. Xue et al. [22] also defined a spatiotemporal pattern of

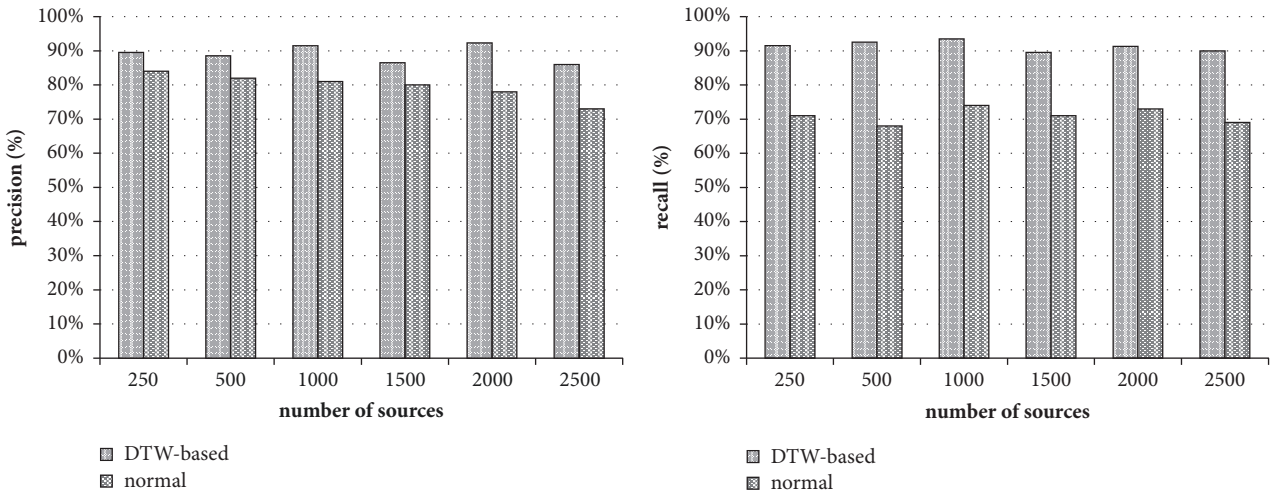

Figure 9: The accuracy of different event capturing algorithms.

events, which described events with a set of regression models over multiple spatial regions. Mao et al. [23] utilized Bayesian Network and Markov Chain to model the spatiotemporal correlation among sensors. However, the above pattern-based methods are mostly based on predefined event patterns which indicated precise sources of sensor streams, and the learning-based methods also need fixed inputs for model training and classification. Hence, they are inappropriate for extracting meaningful information from dynamic sensor streams.

One trend to promote the development of IoT is viewing IoT as Web of Thing (WoT), where the open Web standards are supported for information sharing and device interoperation. Paganelli et al. [10] proposed a framework that supported developers to model smart things as Web resources and exposed them through RESTful APIs. Facing large amount of Web service in IoT age, Qi et al. [24] proposed a novel privacy-preserving service recommendation approach based on locality-sensitive hashing to handle the distributed recommendation problem. DSWare [25] was a data-centric service middleware that defined a compound event specification, which consists of maximum detection range, time interval, and a confidence function. Cheng et al. [26] proposed a situational-aware service coordination method, which included a situational event definition language, a situational event detection algorithm, and an event-driven service coordination behaviour model based on ECA mechanism. Most existing service-based methods needed to predefine the collaboration goals for service composition or collaboration and did not support adaptive collaboration. Hence, they cannot be applied to increase value density for dynamic sensor streams. In this paper, we refer to existing methods to encapsulate sensor data as services and propose a service-based method to support adaptive aggregation of sensor streams.

8. Conclusions

With more and more sensors deployed in physical world, an overwhelming amount of stream data is produced. It is meaningful to set up suitable abstraction to increase value density and to promote widespread use for business applications and data visualization. The paper proposes a service-based method with fog computing diagram for adaptive aggregation of multiple sensor streams. In this paper, we can create a proactive data service by generating service events from fixed sensor streams defined in design time and capture events based on correlation analysis method. Facing the inconsistency of frequency and shift of correlation of different sensor streams, we utilize a Dynamic Time Warping- (DTW-) based algorithm to obtain the lag-correlation and capture dynamic events. For cooperating dynamic sensor streams and capturing more complicated events, we utilize an event routing algorithm for indicating the targets of service events and define a set of declarative rules in our service abstraction for triggering corresponding processing.

To verify our method's feasibility, we applied it for power quality event detection in Chinese power grid. In this scenario, we have to deal with dynamic sensor stream sources. We set up event detection services that can be deployed in cloud server to collect the collaboration results by obtaining finial abnormal events and assisting the adaptive collaboration of proactive data services. Based on the real sensor data set in Chinese power grid, a series of experiments demonstrate that our method have much higher efficiency than existing methods with high accuracy.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (no. 61672042), Models and Methodology of Data Services Facilitating Dynamic Correlation of Big Stream Data; Beijing Natural Science Foundation (no. 4172018), Building Stream Data Services for Spatiotemporal Pattern Discovery in Cloud Computing Environment; and the Program for Youth Backbone Individual, supported by Beijing Municipal Party Committee Organization Department, Research of Instant Fusion of Multisource and Large-Scale Sensor Data.

References

[1] J. Heidemann, M. Stojanovic, and M. Zorzi, "Underwater sensor networks: applications, advances and challenges," *Philosophical Transactions of the Royal Society A: Mathematical, Physical & Engineering Sciences*, vol. 370, no. 1958, pp. 158–175, 2012.

[2] L. Qi, X. Zhang, W. Dou, and Q. Ni, "A distributed locality-sensitive hashing-based approach for cloud service recommendation from multi-source data," *IEEE Journal on Selected Areas in Communications*, vol. 35, no. 11, pp. 2616–2624, 2017.

[3] S. Yi, C. Li, and Q. Li, "A survey of fog computing: concepts, applications and issues," in *Proceedings of the Workshop on Mobile Big Data (Mobidata '15)*, pp. 37–42, ACM, Hangzhou, China, June 2015.

[4] K. Kapitanova, S. H. Son, and K. Kang, "Using fuzzy logic for robust event detection in wireless sensor networks," *Ad Hoc Networks*, vol. 10, no. 4, pp. 709–722, 2012.

[5] W. Xue, Q. Luo, and H. Wu, "Pattern-based event detection in sensor networks," *Distributed Parallel Databases*, vol. 30, no. 1, pp. 27–62, 2011.

[6] Y. Singh, S. Saha, U. Chugh, and C. Gupta, "Distributed event detection in wireless sensor networks for forest fires," in *Proceedings of the 15th International Conference on Computer Modelling and Simulation*, pp. 634–639, Cambridge, UK, April 2013.

[7] O. P. Patri, A. V. Panangadan, V. S. Sorathia, and V. K. Prasanna, "Sensors to events: semantic modeling and recognition of events from data streams," *International Journal of Semantic Computing*, vol. 10, no. 4, pp. 461–501, 2016.

[8] D. Guinard and V. Trifa, "Towards the web of things: web mashups for embedded devices," in *Proceedings of the International World Wide Web Conferences*, pp. 1506–1518, 2009.

[9] D. Zeng, S. Guo, and Z. Cheng, "The web of things: a survey," *Journal of Communications*, vol. 6, no. 6, pp. 424–438, 2011.

[10] F. Paganelli, S. Turchi, and D. Giuli, "A web of things framework for RESTful applications and its experimentation in a smart city," *IEEE Systems Journal*, vol. 10, no. 4, pp. 1412–1423, 2016.

[11] Y. Han, G. Wang, J. Yu, C. Liu, Z. Zhang, and M. Zhu, "A service-based approach to traffic sensor data integration and analysis to support community-wide green commute in China," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 9, pp. 2648–2657, 2016.

[12] Y. Han, C. Liu, and S. Su, "A decentralized and service-based apporach to proactively correlating stream data," in *Proceedings of the International Conference on Internet of Things*, pp. 93–100, 2016.

[13] H. Wu, J. Cao, and X. Fan, "Dynamic collaborative in-network event detection in wireless sensor networks," *Telecommunication Systems*, vol. 62, no. 1, pp. 43–58, 2016.

[14] T. Guo, S. Sathe, and K. Aberer, "Fast distributed correlation discovery over streaming time-series data," in *Proceedings of the 24th ACM International Conference on Information and Knowledge Management*, pp. 1161–1170, October 2015.

[15] S. Wu, H. Lin, W. Wang et al., "RLC: ranking lag correlations with flexible sliding windows in data streams," *Pattern Analysis & Applications*, vol. 20, no. 2, pp. 601–611, 2016.

[16] A. Paschke, "ECA-RuleML: an approach combining ECA rules with temporal interval-based KR event/action logics and transactional update logics," *Computer Science*, 2006.

[17] R. J. Kate, "Using dynamic time warping distances as features for improved time series classification," *Data Mining and Knowledge Discovery*, vol. 30, no. 2, pp. 283–312, 2016.

[18] H. Gupta, S. B. Nath, S. Chakraborty, and S. S. K. Ghosh, "SDFog: a software defined computing architecture for qos aware service orchestration over edge devices," https://arxiv.org/abs/1609.01190.

[19] T. Zhang, A. Chowdhery, P. Bahl, K. Jamieson, and S. Banerjee, "The design and implementation of a wireless video surveillance system," in *Proceedings of the 21st Annual International Conference on Mobile Computing and Networking*, pp. 426–438, September 2015.

[20] M. Yuriyama and T. Kushida, "Sensor-cloud infrastructure—physical sensor management with virtualized sensors on cloud computing," in *Proceedings of the 13th International Conference on Network-Based Information Systems (NBiS '10)*, pp. 1–8, September 2010.

[21] N. Mohamed, J. Al-Jaroodi, I. Jawhar, S. Lazarova-Molnar, and S. Mahmoud, "SmartCityWare: A service-oriented middleware for cloud and fog enabled smart city services," *IEEE Access*, vol. 5, pp. 17576–17588, 2017.

[22] W. Xue, Q. Luo, and H. K. Pung, "Modeling and detecting events for sensor networks," *Information Fusion*, vol. 12, no. 3, pp. 176–186, 2011.

[23] Y. Mao, X. Chen, and Z. Xu, "Real-time event detection with water sensor networks using a spatio-temporal model," in *Database Systems for Advanced Applications*, pp. 194–208, 2016.

[24] L. Qi, H. Xiang, W. Dou, C. Yang, Y. Qin, and X. Zhang, "Privacy-preserving distributed service recommendation based on locality-sensitive hashing," in *Proceedings of the IEEE International Conference on Web Services*, pp. 49–56, Honolulu, Hawaii, USA, June 2017.

[25] S. Li, Y. Lin, S. H. Son, J. A. Stankovic, and Y. Wei, "Event detection services using data service middleware in distributed sensor networks," *Telecommunication Systems*, vol. 26, no. 2-4, pp. 351–368, 2004.

[26] B. Cheng, D. Zhu, S. Zhao, and J. Chen, "Situation-aware IoT service coordination using the event-driven SOA paradigm," *IEEE Transactions on Network and Service Management*, vol. 13, no. 2, pp. 349–361, 2016.