IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received October 31, 2018, accepted November 26, 2018, date of publication December 4, 2018,
date of current version January 4, 2019.

Digital Object Identifier 10.1109/ACCESS.2018.2884924

A Declarative Service-Based Method for Adaptive Aggregation of Sensor Streams

ZHONGMEI ZHANG[1,2,3], CHEN LIU[2,3], XIAOHONG LI[1], YANBO HAN[2,3], CHEN LV[4], AND WEILONG DING[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
[4]School of Information Science and Engineering, Shandong Normal University, Jinan 250358, China

Corresponding author: Zhongmei Zhang (gloria_z@126.com)

This work was supported in part by the Key Project of the National Natural Science Foundation of China under Grant 61832004, in part by the National Natural Science Foundation of China under Grant 61702014 and Grant 61602286, and in part by the Shandong Key Research and Development Program under Grant 2018GGX101003.

ABSTRACT The dynamic interventions among sensor streams bring new challenges for the Internet of Things applications to derive meaningful information from a large amount of sensor streams. This paper proposes a declarative service-based method for capturing meaningful events from dynamic sensor streams based on our previous service abstraction, which can increase the value density of primitive sensor streams and raise the accessibility for higher level business applications. For aggregating related events from different sources, we set up declarative rules and compose cascaded events through service collaboration. In addition, for dealing with dynamic interventions among sensor streams, we propose a declarative rule evolution method to realize adaptive service collaboration at runtime. This paper also reports the tryout use of our method in the China Power Grid for detecting abnormal situations of power quality. Through a series of experiments based on real sensor data in the Power Grid, we verified that our method can capture events with high and steady precision.

INDEX TERMS Sensor stream, IoT service, adaptive service instantiation and collaboration.

I. INTRODUCTION

With the rapid development of Internet of Things (IoT), numerous sensors are deployed in physical world for event monitoring, ecological observation, etc., which produce an over-whelming amount of stream data [1]. In recent years, the emergence of Shared Sensor Networks [2] and Virtual Sensor Networks [3] in sensor network filed makes sensor streams sharable by multitude of applications. But still, it is challenging for applications to aggregate meaningful information from sensor streams with high efficiency and accuracy.

Individual sensor stream is usually fine-grained, has relatively lower value density, and intervenes with others dynamically. For example, power quality events in China Power Grid generally involve different sensor streams when facing different disturbances. For effectively obtaining valuable information, it is crucial to adaptively choose relative stream sources and effectively capture events from dynamic stream sources instead of fixed set of stream source.

At present, a remarkable effort is to encapsulate sensor data in to Web services, in which the open Web standards are supported for information sharing [4]– [6]. 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 (PD-service) in our previous works [7]– [9]. We can map one or several sensors into a PD-service based on user-defined operations at software-level. With PD-service, the sensor events generated directly by physical sensors can be transformed into multiple service events with more meaningful information.

In this paper, we aim to deal with dynamic interventions among sensor streams and propose a declarative service-based method for capturing meaningful events from dynamic sources based on PD-services. Since each PD-service is initially designed to capture events based on fixed set of sources, we can obtain events with more valuable information from multiple sensor streams through service collaboration. For realizing service collaboration, we firstly define a set of declarative rules in PD-services. One declarative rule can be divided into event handler rule which can trigger corresponding process for new events capturing, and hyperlink which indicates the routing targets for certain event. Then, for handling the dynamic interventions among sensor streams, we propose an evolution method to update declarative rules at runtime. Based on the changeable event handler rules and hyperlinks, PD-services can adaptively collaborate with each other and capture meaningful event from dynamic sensor streams.

We preliminarily applied our method in a real-world scenario as power quality event detection in China Power Grid. Through a series of experiments based on real sensor data, we compared our method with existing brute force method. The experimental results show that our method has higher efficiency and has a precision higher than 90% in average, and the evolution method can support continuous effectiveness facing dynamic sensor streams.

The rest of this paper is organized as follows. Section II defines the problem. Section III presents the overview of our declarative service-based method. Section IV introduces the realization of our method for event detection in China Power Grid. The performance evaluation is given in Section V. Section VI introduces related work. And the final section concludes the work.

II. PROBLEM DEFINITION

In the scenario of IoT, sensor data is a continuous stream of one-time values that measure states such as voltage and harmonic.

From the perspective of data value, stream from things can be classified into two types—a continuously generated stream of values sensed from physical phenomena or a stream of separate values generated in a fixed data type [10]. While from the perspective of sensor stream source, sensor data can be provided from single sensor or from a device (e.g. smart phone, smart bracelet, unmanned aerial vehicle) as a whole.

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 is the attribute sets, and $R = \{r_1, r_2, \ldots, r_i, \ldots\}$ is an infinite series of data record $r = \langle \{\langle a_i, v_i \rangle | a_i \in A\}, t \rangle$ which is a set of key-value pairs with a timestamp.

Taking sensor stream in Power Grid as an example, there are thousands of monitoring devices deployed in China Power Grid. Each power quality monitoring device consists of a set of sensors for monitoring the status of power quality. Such monitoring device is able to provide a vector of indicators such as *voltage*, *current*, *frequency*, *harmonic*, etc.. Hence, the sensor stream in Power Grid is provided at device level with discrete values. Figure 1 shows part of sensor stream produced by certain device. It is obvious that primitive sensor streams have low values density and are not significant and readable for users.

An "event" is a happening of interest [11]. In this paper, we regard the meaningful information derived from primitive

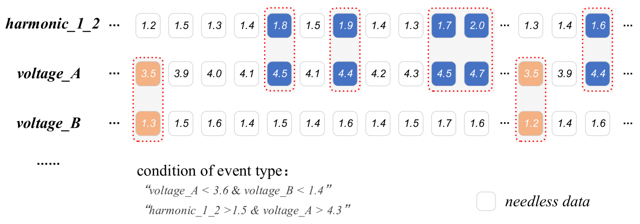

harmonic_1_2 ··· 1.2 1.5 1.3 1.4 1.8 1.5 1.9 1.4 1.3 1.7 2.0 ··· 1.3 1.4 1.6 ···
voltage_A ··· 3.5 3.9 4.0 4.1 4.5 4.1 4.4 4.2 4.3 4.5 4.7 ··· 3.5 3.9 4.4 ···
voltage_B ··· 1.3 1.5 1.6 1.4 1.5 1.4 1.6 1.4 1.5 1.7 1.6 ··· 1.2 1.4 1.6 ···
······
condition of event type:
"voltage_A < 3.6 & voltage_B < 1.4"
"harmonic_1_2 >1.5 & voltage_A > 4.3"
needless data

FIGURE 1. Sensor stream and events example in Power Grid.

sensor stream as events, which can be abstracted as patterns of multiple sensor streams. And we regard pattern of single sensor streams as *primitive event pattern* and pattern of multiple sensor streams as *composite event pattern*, which can be formed by primitive event patterns. To generate meaningful events, we need to aggregate multiple sensor streams.

Primitive event pattern can be defined as a set of attributes are simultaneously in particular range, have particular trends, or belong to particular category, etc.. For example, a set of predicates are utilized to describe the primitive event pattern in [12]. And a composite event pattern can be represented as $p_{com} = \langle \{\langle source_{id}, p_{pri} \rangle\}, R \rangle$, in which $\langle source_{id}, p_{pri} \rangle$ indicates the sensor stream source and its primitive event pattern, R is a set of relations between the primitive event patterns. For example, as shown in Figure 1, only data records that satisfy the condition "$harmonic\_1\_2 > 1.5$ & $voltage\_A > 4.3$" are meaningful for users as events.

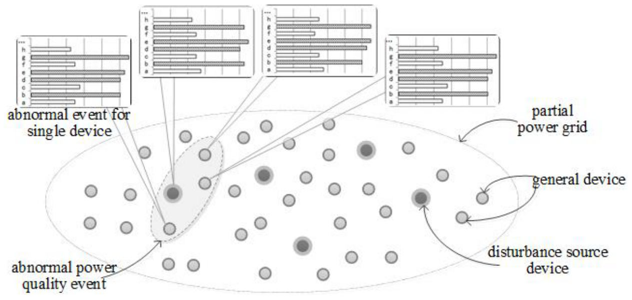

abnormal event for single device
partial power grid
general device
disturbance source device
abnormal power quality event

FIGURE 2. An example of abnormal power quality event.

Figure 2 shows an example of abnormal power quality event in Power Grid. Due to the relationship of equipment and disturbance source, the power quality monitoring devices can be divided into two types: Disturbance source device (DS-device) and General device (G-device). Each device will be influenced directly or indirectly by a disturbance source, and show abnormal (primitive) events as out-of-limit of certain set of indicators. One abnormal power quality event is caused by certain disturbance source and propagates in several equipment (firstly the DS-device deployed near the disturbance source, then other G-devices). Hence, an abnormal power quality event is consisted by several primitive events. In figure 2, the rectangles indicate abnormal events occurred in single device, in which darker columns indicate abnormal indicators and light columns indicate normal indicators. It is showed that the primitive events occurred in each device involved the same attributes, i.e. attribute $\{b, d, e, g\}$, and abnormal events occurred in one DS-device and 3 G-devices form the abnormal power quality event.

Because of the influence of disturbance sources, various abnormal power quality events can occur, which involved dynamic devices. To simplify the primitive event pattern in this paper, we regard it as a set of simultaneous out-of-limit attributes. Through mining history data of each DS-device, we can obtain primitive event patterns caused by a corresponding disturbance source. For G-device, its primitive event patterns are the patterns of all DS-devices since it may be indirectly influenced by any disturbance source.

However, because the dynamic interventions among sensor streams, the abnormal power quality events caused by various disturbances generally involve with dynamic devices, that is the stream sources for capturing meaningful events are not fixed. Because of uncertain stream sources of events, existing method needs to collect and analyze all sensor streams in cloud server for exactly capturing 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. Hence, for deriving meaningful event more efficiently and effectively, we need to adaptively choose related sensor stream sources and aggregate corresponding primitive events on them.

III. OVERVIEW OF THE DECLARATIVE SERVICE-BASED METHOD

A. PRELIMINARIES

We firstly introduce our proactive data service model, whose basic function is to generate events (regard as service event in the proactive data service model) from sensor streams. For generating more meaningful event involving dynamic sources, PD-service also supports to route its service events to related services, recognize the pattern of received events from other services, and adaptively collaborate with others.

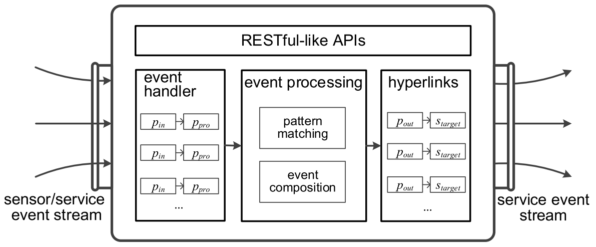


FIGURE 3. **The structure of proactive data service.**

Figure 3 shows the structure of our proactive data service model, a RESTful-like API is utilized for the discovery of PD-service. There are mainly three modules in our model: *event processing* module is in charge of pattern matching and compositing events; *event handler* module is in charge of handling received events through triggering corresponding operations (include capturing new events and composite new events with received events); and *hyperlinks* module is where the event routing paths are placed.

Definition 2 (Proactive Data Service): A PD-service can be formalized as $pds = \langle pds_{id}, in_{event}, eh, ep, out_{event}, hyperlinks\rangle$, in which pds_{id} is the unique identifier, in_{event} is the input events including both sensor event and service event, eh is the event handler which includes a set of rules $\{\langle p_{in}, p_{process}\rangle\}$ to handle received events and trigger the matching of pattern $p_{process}$, ep is the event processing function, out_{event} is the output events, and $hyperlinks = \{\langle p_{out}, T\rangle\}$ describes the routing targets of each output event pattern, in which T is a group of target service' ids.

Based on the proactive data service model, users can create a simple PD-service by specifying the inputs (sensor streams), require events' constraint conditions, and *uri* for the service. It can be encapsulated into a Restful-like API so that other services or applications can use it for obtaining high-level events from sensor streams. For example, we can create an PD-service for certain device in Power Grid through set stream sources $\{voltage\_A, voltage\_B, harmonic\_1\_2\}$ as its input, and set events $\{voltage\_A<3.6\&voltage\_B<1.4, voltage\_A>4.3\&harmonic\_1\_2>1.5\}$ as its output, the event processing module will match predefined pattern with input sensor streams automatically for capturing required events.

In a PD-service, the *hyperlinks* and event handler rules are the keys to support service collaboration. For compositing related service events, we set up *hyperlinks* and event handler rules based on declarative rules, which indicate the relation between service events generated by different services.

Definition 3 (Declarative Rule): A declarative rule can be represented as $dr = \langle pds_{id\_s}, p_s, pds_{id\_t}, p_t\rangle$, in which, pds_{id_s} is the id of source PD-service, p_s is the pattern of certain event generated by service pds_{id_s}, pds_{id_t} is id of target PD-service, and p_t is the pattern of certain event generated by service pds_{id_t}.

With a declarative rule, we can add a hyperlink of $\langle p_s, pds_{id\_t}\rangle$ in service pds_{id_s} to indicate the target services for certain service event, and add an event handler $\langle p_s, p_t\rangle$ in service pds_{id_t}. In this case, when service pds_{id_s} generates an event with pattern p_s, it will be routed to pds_{id_t}. And when pds_{id_t} receives an event with pattern p_s, it will trigger the matching of pattern p_t, and if an event with pattern p_t is generated, it will be composited with received event and a more meaningful event will be generated. Otherwise, the collaboration between these two services is failure. The collaboration will keep working with new events' generation, and a final event will be generated when all involved PD-services are failure to collaborate with their related ones or there is no related service to be collaborated with.

B. THE FRAMEWORK OF OUR METHOD

Specifically, we design two types of services to capture high-level events, which are proactive data service (PD-service) corresponding to physical device or sensor and large-grained Event Detection service (ED-service) based on PD-service collaboration. Each PD-service can capture events from specified sensor streams. And through proactively collaboration, we can obtain higher-level events involving multiple sensor streams. ED-service can collect collaborative results for generating the final composite events and distribute event routing paths to PD-services based on the collaboration result.

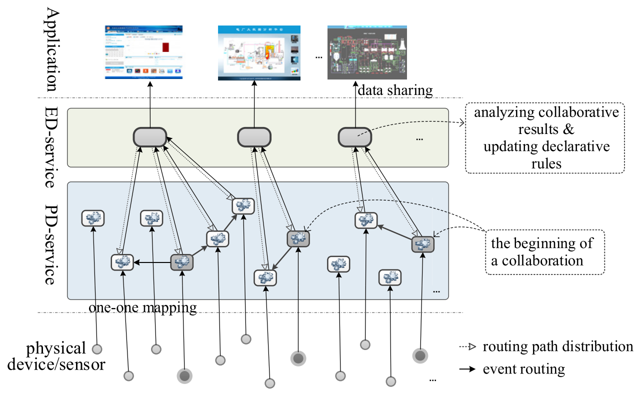

Application
ED-service
PD-service
data sharing
analyzing collaborative results & updating declarative rules
the beginning of a collaboration
one-one mapping
physical device/sensor
routing path distribution
event routing

FIGURE 4. The schematic diagram of declarative service-based event capturing method.

Figure 4 shows a schematic diagram of our method. Each PD-service can receive sensor stream from corresponding device/sensor and events from other PD-services to generate more meaningful events. When an event is captured by a PD-service, it will be sent to related PD-service. And once receives an event, the event handler in PD-service will trigger the capturing of new events. Since the collaboration always begins with one PD-service and diverts to other PD-services, the final composite event will be distributed in different PD-services. For capturing the complete event, the ED-service is needed to collect the collaborative results from all involved PD-services. Since the PD-services involved in one collaboration is unpredictable, the ED-service is also in charge of setting declarative rules (i.e. event handler and hyperlink) for PD-services. When a PD-service can collaborate with previous PD-services, the ED-service distributes event routing paths to it and distributes event handler to its related PD-services. To be mentioned, to lower the computation and storage costs of the cloud server, the PD-service can be deployed in edge nodes, do some preliminary analyses, and report events to cloud servers for further analyses (in ED-service).

As mentioned in section II, the stream sources for capturing meaningful events are not fixed. For handling the dynamic interventions among sensor streams, the declarative rules need to be updated at runtime for supporting adaptive service collaboration and capturing events from dynamic sensor streams. In this paper, we integrate an evolution method in the ED-service to update declarative rules based on parts of priori experience and the actual effect of service collaboration. Based on the changeable rules and hyperlinks, PD-services can adaptively collaborate with each other at runtime.

IV. REALIZATION OF THE POWER QUALITY EVENT DETECTION

The power quality event in Power Grid is a typical kind of event which involves dynamic sensor stream sources. And it has the following characteristics: 1) only partial knowledge is known, which includes the starting sources (disturbance source devices) of power quality events, threshold of sensor data, and primitive event pattern for each sensor stream; 2) the primitive event pattern of each source involved in the power quality event is the same, which can simplify the generation of declarative rules since the source pattern and target pattern are the same; and 3) because of instantaneous transmission of electricity, the temporal relation in an event is simultaneous.

A. THE EXISTING METHOD

At present, power quality detection system has been built to detect abnormal power quality event. Since the sources involved in power quality events are changeable, the existing method is kind of brute force method, which needs to process all sensor streams. It is meaningless and will consume plenty of computing and communicating resources. Figure 5 shows the processes of existing method.

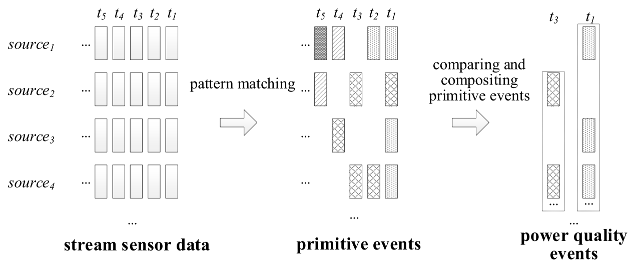


FIGURE 5. The processes of existing power quality event detection method.

As mentioned before, the primitive event patterns for single sensor stream can be obtained through mining history sensor data. As showed in figure 5, existing method firstly matches the real-time sensor stream from all sources with predefined primitive event patterns. Then, it compares all primitive events' patterns with the same timestamp and composites primitive events with the same primitive event pattern as the final power quality event. Due to plenty of sources and large-scale sensor streams, the existing method cannot guarantee the instantaneous requirement of power quality event detection.

B. THE REALIZATION OF OUR SERVICE-BASED METHOD

Through analyzing history sensor data, we can obtain the primitive event patterns (a set of attributes which are out-of-limit) for each DS-device. Then based on our proactive data service model, we can create one PD-service for each sensor stream source (power quality monitoring device), and set up its RESTful-like API based on the identifier of physical device. Each PD-service has following capabilities: 1) recognizing the pattern of input service events and judge whether to process it; 2) matching sensor event with predefined patterns and compositing events; and 3) routing collaborative result. Among these capabilities, event composition only needs to integrate sensors from events with the same pattern. For example, event $\langle\{A, B\}, p_i\rangle$ and $\langle\{C\}, p_i\rangle$ can be composed into $\langle\{A, B, C\}, p_i\rangle$, in which p_i is a primitive event pattern. And routing capability only needs to route the collaborative result based on routing paths in hyperlinks.

Since the power quality event is caused by a disturbance source, each collaboration procedure can be started with one PD-service corresponding to a DS-device. We set up one event detection service (ED-service) for each DS-device to collect collaborative results and generate the final abnormal power quality events.

For realization adaptive service collaboration, we need to define and keep updating declarative rules that indicate relation of service events in PD-services. Since there is no electrical connection information, we define declarative rules through considering both the location of physical devices and existing collaborative results. We integrate a declarative rule evolution algorithm in the ED-service, and the algorithm has two phases, i.e. the initial phase to generate the initial declarative rules and the evolution phase to evolve the declarative rules.

In the initial phase, we regard the neighbor PD-services as the total set of routing target $S_{possible}$ for given service based on physical devices' location.

And in the evolution phase, we keep updating the correlation of PD-services based on their collaborative results for certain event pattern. We assume that devices frequently co-occurred 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 didn't co-occurred in one power quality event still have possibility to form an 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.

Specifically, we set up weight ω_i for each PD-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 PD-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 its related services. Algorithm 1 shows the pseudo-code of the second phase in our algorithm.

In algorithm 1, we update the weights of target PD-services for certain service and certain event pattern based on newly received collaborative result, and because the target pattern is the same with source pattern, we then can obtain the declarative rules. After each event routing, we increase target PD-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 updating the event handlers and links in the PD-service.

Algorithm 1 Updating the Routing Targets

```
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.          $s_{ij}.weight = s_{ij}.weight + \alpha$;
6.      else
7.          extract the invalid routing path <$pattern, s_{ij}$ >;
8.          $s_{ij}.weight = s_{ij}.weight - \beta$;
9.          if $s_{ij}.weight < \delta$
10.             remove $s_{ij}$ from $s_{ipractical}$;
11.         end if
12.     end if
13.     for each $s_{ij}$ in $s_{alternative}$
14.         $s_{ij}.weight = s_{ij}.weight + \gamma$;
15.         if $s_{ij}.weight \geq \delta$
16.             add $s_{ij}$ in $s_{ipractical}$;
17.         end if
18.     end for
19. end while
```

C. CASE STUDY

Taking the detection procedure of an abnormal power quality event caused by disturbance source *wind farm* as an example, this section describes our proactive service-based method. Figure 6 shows the collaboration procedure of PD-services. Due to limited space, only parts of PD-services are shown. Among them, PD-service A is corresponding to a disturbance source device, whose event patterns, output events, routing paths can be set up at design time. Other PD-services are corresponding to general devices, and their event patterns to be matched, output events and routing paths are updated at runtime. Table 1 shows the detail of some PD-services.

The collaboration is beginning with PD-service A, when an event $< A, p_1 >$ matched pattern p_1 is generated, A sends it to service B, C and the event detection service. When B and C receive this event, they will match their sensor event with pattern p_1, and events generated by B and C will composed with event received from A to generate more meaningful event $<\{A, B\}, p_1 >$ and $<\{A, C\}, p_1 >$. The new events will be continuously routed. Since F doesn't generate a new event after receiving event from C, it sends a failure message to ED-service and stops collaborating with other PD-services. Event generated by service B is sent to service D and E, and only service D generates new event and sends it to service G and H. Since both service G and H are failure to collaborate with service D, the whole collaboration is finished, and a power quality event involved PD-services A, B, C, D is generated.

At each step of the collaboration, the ED-service collects collaborative results and updates corresponding routing

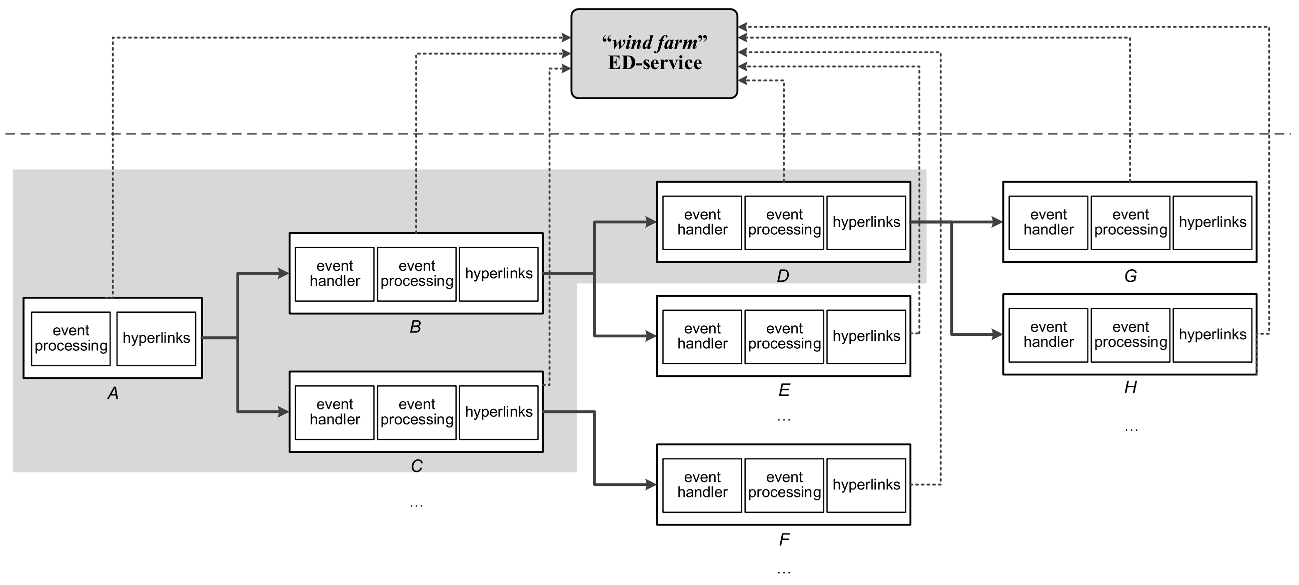


FIGURE 6. **Collaboration procedure of abnormal power quality event detection.**

TABLE 1. **The detail of PD-service in our scenario.**

ID	*pattern*	*eventout*	*routing paths*	*set up time*
A	p_1	$<A, p_1>$	$p1 \rightarrow \{B, C\}$	*design time*
B	p_1	$<\{A, B\}, p_1>$	$p1 \rightarrow \{D, E\}$	*runtime*
C	p_1	$<\{A, C\}, p_1>$	$p1 \rightarrow \{F\}$	*runtime*
D	p_1	$<\{A, B, D\}, p_1>$	$p1 \rightarrow \{G, H\}$	*runtime*

paths. For PD-service A and event pattern p_1, routing paths $p_1 \rightarrow \{B, C\}$ are valid, so the weights of targets will be increased, and the routing paths can be kept. For service B and event pattern p_1, the routing path $p_1 \rightarrow E$ is invalid, so the weight of E will be decreased and the routing path may be removed. Besides, there may be PD-services which are not in the routing paths while they are neighbors of certain PD-service. Their weight will also be increased, so they can be considered for detecting dynamic events.

V. EVALUATION

A. EXPERIMENT SET UP

We used existing brute force method as benchmark to verify the efficiency and effectiveness of our method.

1) DATA SET

The data set used in our experiments is from real sensor data in China Power Grid. We select sensor data of certain province from 2017-06-01 to 2017-06-05. There are 3453 devices with 438 DS-devices. Since we cannot directly access sensor streams from Power Grid, we simulate stream for each device strictly according to the timestamp of sensor data. Based on expert experience, we define 2-6 types of primitive event for each DS-device, and there are totally 2178 primitive event patterns.

2) EVALUATION CRITERIA

For verifying the efficiency and effectiveness of our method, we design following criteria:

(1) System load. The system load of event detection can be divided into the load of CPU and memory.

(2) 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.

(3) Precision. The precision can be calculated as follows:

$$precision = \frac{|D \cap T|}{|D|} \times 100\%$$

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

B. EXPERIMENT RESULTS AND ANALYSES

We firstly compared our method with existing method with different sensor stream sources, i.e. device sets in Power Grid. Based on the 2178 primitive event patterns, we realized the brute force method through processing sensor streams of all devices to capture the power quality events. While with our method, we created one PD-service for each device and one ED-service for each DS-device, and only set 2-6 primitive event patterns in PD-services corresponding to DS-devices. The process in PD-service corresponding to G-device will depend on its received events. Through adaptive PD-service collaboration, related events generated by different services can be gathered and composited in the ED-service to generate the final power quality events.

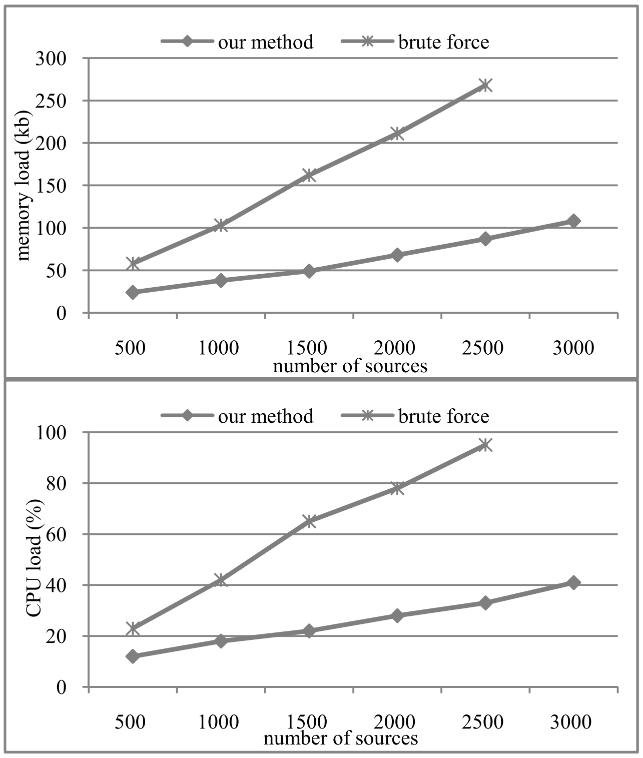


FIGURE 7. The system load of different methods with different device sets.

As shown in figure 7, both methods have higher system load with the enlargement of device set. And our method has much smaller system load with smaller growing rate. This is because the brute force method needs to process all sensor streams with all primitive event patterns. While in our method, the stream from DS-device only needs to be matched with corresponding patterns, and the process of sensor stream from G-device is rely to the process results of previous service. What's more, only sensor streams from related sources are processed with our method, and the reducing of unnecessary processing makes our method have a smaller system load.

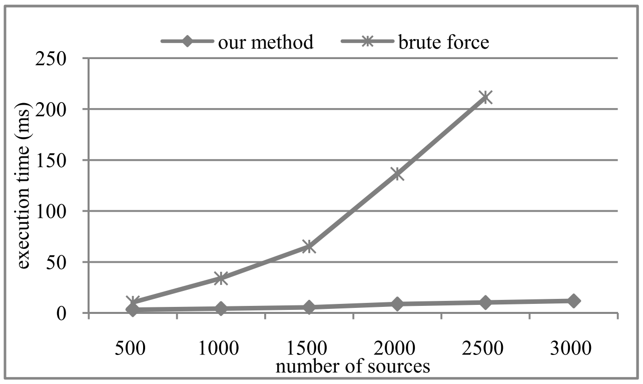


FIGURE 8. The execution time of different methods with different device sets.

Figure 8 shows the execution time of our method and brute force method. Comparing with the brute force method, our method has much shorter execution time, and the growth trend is much gentler.

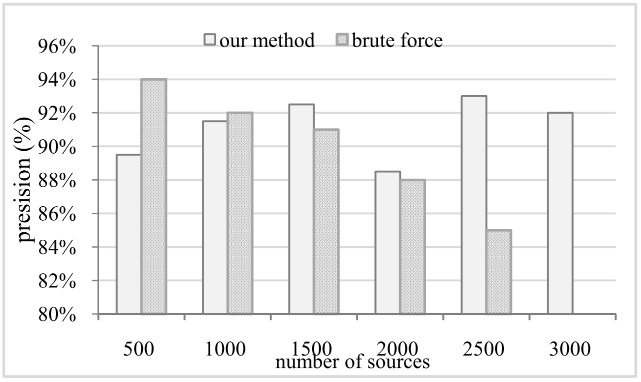


FIGURE 9. The precision of different methods with different device sets.

Figure 9 shows the precision of our method and brute force method. It is observed that the precision of brute force method is decreased with larger device sets. It is because that there may include more devices which have no direct or indirect correlations with DS-device with the growing of devices' number, and the generated power quality events may involve sensor streams from irrelevant sources. The precision of our method is not influenced by the change of device sets and the average precision is reached 91%.

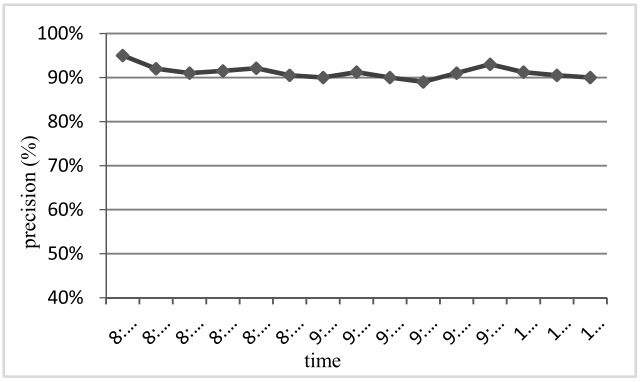


FIGURE 10. The precision of our method in a sustained period.

Then we evaluated the continuous effectiveness of our method facing dynamic intervention among sensor streams. We set the number of devices as 1000 and calculated the precision of event capturing in sequential time points. Figure 10 shows the precision of our method in a sustained period. It is obvious that the precision of our method keeps around in 90%. The experiment result verified that our method can handle dynamic intervention among sensor streams and effectively capture events.

VI. RELATED WORKS

A. EVENT DETECTION

Various techniques have been proposed for efficient event detection which can be classified into threshold-based, pattern-based, and learning-based method.

1) THRESHOLD-BASED APPROACH

In this approach, the events are detected based on pre-set threshold value, which is generally defined through statistical models or experts' experiences. Since rely on a given threshold alone cannot accurately handle the imprecise sensing data, many works [13]– [15] used fuzzy value instead of a precise threshold to describe the event, thereby greatly improved the accuracy of event detection. However, this approach still fails to describe complex events involving multi-sourced data.

2) PATTERN-BASED APPROACH

This approach abstracted events into various patterns of multiple sensor data and transformed event detection problems into pattern matching problems. Xue *et al*. [16] integrated the pattern-based approach with an in-network sensor query processing framework, it focused on the event patterns for single sensor and defined 5 common basic patterns to describe events. Xue *et al*. [17] also defined a spatio-temporal pattern of events, which described events with a set of regression models over multiple spatial regions, and proposed an efficient matching algorithm for detecting events in real time. Wu *et al*. [12] proposed a dynamic and collaborative event detection framework, which allowed sensors to dynamically cooperate with each other based on the temporal relationship of events and spatio-temporal correlation of sensor data. Most of existing pattern-based approaches are based on precise pre-defined event patterns, and didn't consider event detection with event patterns that cannot be pre-defined.

3) LEARNING-BASED APPROACH

Various machine learning techniques have also been used to decide whether an event has occurred or not. These learning-based methods modeled history sensor data based on various classification models, e.g. Neural Network [18], Support Vector Machine [19], Markov Random Field [20] etc., and then classified real-time sensor data. Singh *et al*. [19] proposed a distributed machine learning approach for event detection in two phases, base phase and meta phase. It detects events in a distributed manner using Support Vector Machine (SVM) classification with polynomial kernel. Markov Random Field (MRF) [20] has been adopted to model the spatial relationship of neighboring nodes and improve the accuracy of event detection by considering the data of nearby sensors. The using of MRF and Markov Chain in [21] described the spatio-temporal context of events to further improve the accuracy of event detection.

However, the above pattern-based methods mostly based on pre-defined 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.

B. SERVITIZATION OF STREAM SENSOR DATA

The servitization of stream sensor data is an effective way to support the reusability of sensor data and processing ability across organizations based on the Internet. Guinard and Trifa [4] proposed the concept of ''Web of Thing'' (WoT), which can encapsulate sensor data from different sensor networks into web services. Zeng *et al*. [22] provided a survey on research works in WoT domain, where highlight two main analysis criteria: integrating physical devices (directly or indirectly) on the Web, and providing composable services to enable physical-virtual mashups. Paganelli *et al*. [23] proposed a framework which supported developers to model smart things as web resources, expose them through RESTful APIs. It modeled smart things as a graph of individually addressable web resources whose edges represent simple structural aggregation and reference relations.

Many works took advantage of Complex Event Processing (CEP) services to provide and reuse CEP capabilities through describing event patterns in services and generating complex events by reusing and compositing services [24], [25]. DSware [26] 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*. [27] 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 behavior model based on ECA mechanism.

However, most existing service-based methods also needed to predefine the collaboration goals for service composition or collaboration, and didn't support adaptive service collaboration. Hence, they cannot be applied to increase value density for dynamic sensor streams. In this paper, we refer existing methods to encapsulate sensor data as services, and propose a declarative service-based method to support adaptive aggregation of sensor streams.

VII. CONCLUSION

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 declarative service-based method based on our previous service abstraction for adaptively aggregating multiple sensor streams. We create initial PD-services for capturing service events from fixed sensor streams defined in design time. And to support aggregation of multiple service events, we set up declarative rules in PD-services to form routing paths and event handler rules, with which services can collaborate with each other. For handling dynamic interventions among sensor streams, we propose an evolution method to update the declarative rules at runtime and realize adaptive service collaboration.

To verify the feasibility, we applied our method for power quality event detection in China Power Grid. We set up event detection services to collect the collaborative results by obtaining finial abnormal events and assisting the adaptive collaboration of PD-services. And we obtained the declarative rules based on both physical devices' locations and existing collaborative results. Based on real sensor data in China Power Grid, a series of experiments demonstrate that our method has much higher efficiency than existing method with high and continuous precision.

Power quality event detection in China Power Grid is relatively implementable with our method, since it has a clear requirement and concise business logic. In the future, we will apply our method in more complicated applications such as anomaly event detection in coal-fired power plant, and further improve our method.

REFERENCES

[1] J. Heidemann, M. Stojanovic, and M. Zorzi, "Underwater sensor networks: Applications, advances and challenges," *Philos. Trans. Roy. Soc. London A, Math. Phys. Sci.*, vol. 370, no. 1958, pp. 158–175, Jan. 2012.

[2] C. Farias *et al.*, "A systematic review of shared sensor networks," *ACM Comput. Surv.*, vol. 48, no. 4, p. 51, May 2016.

[3] I. L. Santos, L. Pirmez, F. C. Delicato, S. U. Khan, and A. Y. Zomaya, "Olympus: The cloud of sensors," *IEEE Cloud Comput.*, vol. 2, no. 2, pp. 48–56, Mar. 2015.

[4] D. Guinard and V. Trifa, "Towards the Web of things: Web mashups for embedded devices," in *Proc. Int. World Wide Web Conf. Workshop Mashups, Enterprise Mashups Lightweight Composition Web (MEM)*, Madrid, Spain, 2009, pp. 1–8.

[5] S. De, P. Barnaghi, M. Bauer, and S. Meissner, "Service modelling for the Internet of Things," in *Proc. IEEE Comput. Sci. Inf. Syst.*, vol. 29, Nov. 2011, pp. 949–955.

[6] Z. H. Ali, H. A. Ali, and M. M. Badawy, "A new proposed the Internet of Things (IoT) virtualization framework based on sensor-as-a-service concept," *Wireless Pers. Commun.*, vol. 97, no. 1, pp. 1419–1443, Nov. 2017.

[7] Y. Han, G. Wang, and J. Yu, "A service-based approach to traffic sensor data integration and analysis to support community-wide green commute in China," *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 9, pp. 2648–2657, Sep. 2016.

[8] Y. Han, C. Liu, S. Su, M. Zhu, and Z. Zhang, "A decentralized and service-based approach to proactively correlating stream data," in *Proc. S2 Int. Conf. Internet Things*, Madrid, Spain, 2016, pp. 93–100.

[9] Z. M. Zhang, C. Liu, and S. Su, "SDaaS: A method for encapsulating sensor stream data as services," *Chin. J. Comput.*, vol. 40, no. 2, pp. 445–463, May 2017.

[10] W. Kang, K. Kapitanova, and S. H. Son, "RDDS: A real-time data distribution service for cyber-physical systems," *IEEE Trans Ind. Informat.*, vol. 8, no. 2, pp. 393–405, May 2012.

[11] F. Wang, C. Zhou, and Y. Nie, "Event processing in sensor streams," in *Managing and Mining Sensor Data*, C. Aggarwal *et al.*, Eds. Boston, MA, USA: Springer, 2013, pp. 77–102.

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

[13] M. Collotta, L. L. Bello, and G. Pau, "A novel approach for dynamic traffic lights management based on wireless sensor networks and multiple fuzzy logic controllers," *Expert Syst. Appl.*, vol. 42, no. 13, pp. 5403–5415, Aug. 2015.

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

[15] K. X. Thuc and K. Insoo, "A collaborative event detection scheme using fuzzy logic in clustered wireless sensor networks," *AEU-Int. J. Electron. Commun.*, vol. 65, no. 5, pp. 485–488, May 2011.

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

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

[18] W. Cui *et al.*, "An algorithm for event detection based on social media data," *Neurocomputing*, vol. 254, pp. 53–58, Sep. 2017.

[19] Y. Singh, S. Saha, and U. Chugh, C. Gupta, "Distributed event detection in wireless sensor networks for forest fires," in *Proc. Int. Conf. Comput. Modeling Simulation (Uksim)*, Cambridge, U.K., 2013, pp. 634–639.

[20] T.-Y. Wang, M.-H. Yang, and J.-Y. Wu, "A sliding window approach for dynamic event-region detection in sensor networks," in *Proc. Int. Conf. Inf. Sci., Electron. Elect. Eng.*, Sapporo, Japan, 2014, pp. 2025–2028.

[21] X. Chen, K. T. Kim, and H. Y. Youn, "Integration of Markov random field with Markov chain for efficient event detection using wireless sensor network," *Comput. Netw.*, vol. 108, pp. 108–119, Oct. 2016.

[22] D. Zeng, S. Guo, and Z. Cheng, "The Web of things: A survey," *J. Commun.*, vol. 6, no. 6, pp. 424–438, Jan. 2011.

[23] F. Paganelli, S. Turchi, and D. Giuli, "A Web of things framework for restful applications and its experimentation in a smart city," *IEEE Syst. J.*, vol. 10, no. 4, pp. 1412–1423, Dec. 2016.

[24] F. Gao, E. Curry, and S. Bhiri, "Complex event service provision and composition based on event pattern matchmaking," in *Proc. ACM Int. Conf. Distrib. Event-Based Syst.*, Mumbai, India, 2014, pp. 71–82.

[25] F. Gao, E. Curry, M. I. Ali, S. Bhiri, and A. Mileo, "Qos-aware complex event service composition and optimization using genetic algorithms," in *Proc. 12th Int. Conf. Service-Oriented Comput.*, Paris, France, 2014, pp. 386–393.

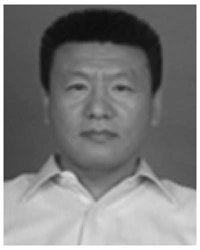
[26] S. Li, Y. Lin, S. H. Sang, J. A. Stankovic, and Y. Wei, "Event detection services using data service middleware in distributed sensor networks," *Telecommun. Syst.*, vol. 26, nos. 2–4, pp. 351–368, Jan. 2003.

[27] B. Cheng, D. Zhu, S. Zhao, and J. Chen, "Situation-aware IoT service coordination using the event-driven SOA paradigm," *IEEE Trans. Netw. Service Manage.*, vol. 13, no. 2, pp. 349–361, Mar. 2016.

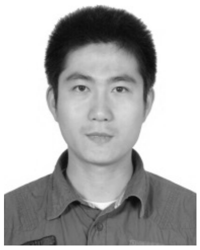
ZHONGMEI ZHANG is currently pursuing the Ph.D. degree with the School of Compute Science and Technology, Tianjin University, China. Her research interests include services computing, and streaming data integration and analysis.

CHEN LIU received the Ph.D. degree in computer science and technology from the Chinese Academy of Sciences, Beijing, China, in 2007. Since 2012, he has been an Associate Professor with the Research Center for Cloud Computing, North China University of Technology, Beijing. His research interests include data integration, service modeling, service composition, and cloud computing.

XIAOHONG LI received the Ph.D. degree from School of Computer Science and Technology, Tianjin University, Tianjin, China, in 2005. Since 2009, she has been a Full Tenured Professor with the School of Computer Science and Technology, Tianjin University. Her current research interests include knowledge engineering, trusted computing, and security software engineering.



YANBO HAN received the Ph.D. degree in computer science from the Technical University of Berlin, Germany. Since 2000, he has been a Full Professor in computer science with the North China University of Technology. His current research interests include streaming data processing, cloud computing, dependable distributed systems, and business process collaboration and management.



CHEN LV received the Ph.D. degree from the Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China, in 2015. He is currently with the School of Information Science and Engineering, Shandong Normal University, China. His research interests include service computing, API usability, and distributed event-based system.

WEILONG DING received the Ph.D. degree from the Institute of Computing Technology, Chinese Academy of Sciences, in 2013. He has been an Associate Professor with the North China University of Technology, Beijing, China. His major research interests include real-time data processing, distributed system, and service computing.