Research Article

A Method of the Active and Passive Event Service Based on the Sensor Web

Lan Liu ¹, Jingjing Fan,² Chengfan Li ³ and Xuefeng Liu ²

¹School of Electronic and Electrical Engineering, Shanghai University of Engineering Science, Shanghai 201620, China

²School of Communication and Information Engineering, Shanghai University, Shanghai 200444, China

³School of Computer Engineering and Science, Shanghai University, Shanghai 200444, China

Correspondence should be addressed to Xuefeng Liu; lx02@shu.edu.cn

Received 7 December 2021; Accepted 10 January 2022; Published 31 January 2022

Academic Editor: Yuyu Yin

Copyright © 2022 Lan Liu et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

The intelligent information system constructed by the sensor web can monitor all kinds of sudden abnormal events, improve the ability of event discovery and rapid disposal, and promote the development of smart city and the construction of the Internet of Things (IoT). In this paper, we consider the problem of complex integration and poor expansibility in the large-scale full-network operation and maintenance system construction and propose an active and passive event service (APES) method based on the sensor web. In the APES, a system framework with the perception layer, data service layer, event service layer, and user layer is firstly defined and constructed. Secondly, a middleware with the ability to active and passive event service (APES) is designed and implemented based on the system framework. Finally, taking abnormal weather and fire warning as examples, the performance of the proposed event service middleware is tested, respectively. Experimental results show that the proposed APES model in this paper has the advantages of high precision, stable operation, and strong practicability and solves “Information Island” and low reusability in the whole network operation and maintenance system. This is an attempt at the structural design of a similar intelligent information system.

1. Introduction

With the commercially developing industrial Internet of Things (IIOT), the original scattered industrial network is rapidly evolving into a large-scale industrial interconnection network via the integration of information and communications technology (ICT) and operational technology (OT) [1–3]. How to ensure the stable and efficient operation of information communication and digital applications and build a network management system with distributed monitoring and intensive management has become a difficulty in related research and application. Meanwhile, more and more enterprises and industries are paying attention to the economic benefits brought by remote operation and predictive maintenance of network infrastructure.

At present, the combination of the sensor network and event service has become an important content of smart city construction [4, 5]. In 2005, the open geospatial consortium

(OGC) proposed a novel sensor web enablement (SWE) framework, and the corresponding sensor web standards and protocols have also been presented. In view of the standard-based discovery, access, subscription, and other operations within the sensor web, the structural differences in communication protocols, data acquisition, processing, and storage in different sensor networks can be eliminated, and then the island effect of sensor networks can be decreased too. Sensor observation server (SOS) mainly focuses on data services. As one of the data sources of event services, SOS manages and stores the data collected by front-end sensors in a unified manner and returns the qualified data to users facing requests.

The event service dedicated to the discovering and alerting of events is usually composed of event providers and event consumers [6, 7]. There are two ways to communicate between the event provider and the event consumer, for example, request-response mode and

subscription-publishing mode [8–10]. Based on the active and passive relationship between the event provider and consumer, the event service is correspondingly divided into active service and passive service [11]. Traditionally, event services are active or passive with respect to service providers. In the event service binary structure composed of event providers and consumers, providers actively “push” services to consumers as active services, and consumers actively “pull” services as passive services. In fact, it is difficult to determine whether these two modes are active or passive under the above definition because the act does not clearly define the “push” or “pull.”

To address the above problems and simplify and clarify the definition of the active and passive relationship of event service, in this paper, the definitions of the active and passive service can be preset as follows: whether the active service or the passive service still takes the event service provider as the reference subject, which emphasizes the core of the event service initiator. If an event service is initiated by the consumer of the service, that is, the service consumer first requests or subscribes to the service from the service provider, the service is regarded as a passive service regardless of the way the service provider provides the service to the service consumer. On the contrary, if the initiator of the service is the provider of the service, the consumer only passively accepts the service, and this mode should be regarded as an active service relative to the reference subject of the service provider. For example, both request-response and subscription-publication event service modes are passive services, and push daily weather forecast, air quality status, and other public service behaviours to the public are the active service. Our proposed active and passive event service (APES) model based on the sensor web has the advantages of high precision, stable operation, and strong practicability and effectively solves the island effect and low reusability in the whole network operation and maintenance system.

The major contributions are concluded as follows:

- (1) Based on the sensor web framework, the APES scheme is presented from the aspects of service architecture, service mode, and event discovery, which is composed of the perception layer, data service layer, event service layer, and user layer.
- (2) An event service middleware (ESM) in the APES method is designed and implemented based on the sensor web. It manages multisource sensor observation data and simplifies the interaction between middleware, data service layer, and users.

The rest of this paper is structured as follows. Section 2 discusses the related work of the APES method based on the sensor web. Section 3 presents details of the event service model including overall architecture and detailed design of the event service layer. Section 4 describes the implementation of the ESM, and Section 5 presents our experimental results. Our conclusions and future work are presented in Section 6.

2. Related Work

The object management group (OGM) first formulated the common object request broker architecture (CORBA) specification and event service specification [12–14]. CORBA event service architecture is mainly composed of the event provider, event channel, and event consumer. Due to the intermediary (event channel) between providers and consumers of the CORBA architecture, the providers and consumers do not interact directly. The CORBA event service model has no clear relationship between the active and passive services.

With the development of the Internet of Things (IoT), the event service model still faces the unprecedented challenges. In the CORBA’s trinity event service model, the discovery of events is at the forefront, and the event channel middleware only plays the role of event delivery. The front-end sensors are mainly responsible for the observation and collection of data and are not directly responsible for the discovery (decision) of events. It not only serves as a delivery event but also can discover and filter events. Therefore, the CORBA event service model cannot meet the above requirements for event services on the IoT. OGC complies with the demands of the development of the IoT and proposes a new type of SWE, which satisfies the intelligent requirements for event services in the IoT era to some extent [15]. As an important part of the SWE, the sensor event service (SES) combines the sensor web with event services to deliver perceptive information that meets filter conditions to consumers [16]. The SES specification adopts a subscription-notification pattern of real-time event services and, in essence, is a passive service. As a member of the SWE, SOS is a data service component, which has no event service capability and can only save sensor observation data for a long period of time to facilitate data backtracking [17, 18]. It provides data support for the construction of such event filters when historical observations are required as a reference for the occurrence of an event.

Based on the standard CORBA specification, an active service system framework is proposed, and the active service is defined as the service provider providing the service to the user without the user giving the order [19]. In this framework, the active service corresponds to the “push” pattern in the CORBA specification, the passive service corresponds to the “pull” pattern, the subscription pattern is classified as the active service, and the request-response pattern is classified as the passive service. To some extent, it extends the service mode of traditional event service. A new notification service based on the traditional event service was first presented in 2000. It is compatible with event services, except that notification services add filtering capabilities. Real-time event service is to extend the standard event service to meet the requirements of real-time applications. And it realizes the real-time scheduling strategy, and the information can be allocated and processed in time.

Then, a real-time event detection service with data service middleware (DSWare) was developed [20]. The middleware can handle unreliable individual reports in

real-time event services, sensor observations with different correlations, and real-time events with special attributes. It can support data semantics information including the relative importance and historical patterns of events. However, the accuracy is low, and the practicability is poor.

The SES specification in the overall sensor web framework has been applied to Zanzibar Island in northeast Tanzania, Africa, and successfully realized the monitoring and notification service of water resources [21]. It avoids wasting a lot of time on polluting water sources, indirectly contributing to the region's workforce and economic development. By integrating multiple object-oriented architecture standards and the real-time event-driven architecture (EDA) package, it sends the data to a central communication middleware, namely, enterprise service bus (ESB), and finally diffuses the data into SES components. By extending the basic CORBA event service specification, a middleware that can complete a real-time event service is designed when consumers can firstly receive notification when an event occurs. It can improve the timeliness and continuity of event service and use efficiency. Since events occur continuously and dynamically in the IoT environment, a hybrid complex event service model based on the IoT resource model is presented, and it has great flexibility and wide application range in the actual applications.

In this paper, we design and implement ESM in APES and use for event discovery and event service. The sensor observation information from the SOS is converted into events so that hot events of public concern can be actively pushed to registered users, while the specific events can be monitored according to the user's personalized subscription requirements after passively accepting the user's subscription request. When an event occurs, an alert notification is sent to the user. This scheme not only makes up for the shortage of the CORBA but also solves the defects that SES cannot provide pure active event service.

3. Event Service Model

3.1. Overall Architecture. The APES model consists of four parts, i.e., perception layer, data service layer, event service layer, and application layer. And the overall architecture of the APES model is shown in Figure 1.

The sensor layer consists of a series of sensors that send the collected data to the data service layer. The core of the data service layer is SOS. The event service layer is the core of this model. The ESM is composed of the data request and parsing module, active service module, and passive service module. The main body of the application layer is the user (consumer), which requests service, receives, and displays alarm information by the consumer's intelligent mobile device.

For the event M , the metamodel can be defined as

$$\text{Event } M = (I \text{ } d, A, E, L, T), \quad (1)$$

where Id is the number of sensors, and each Event M can be defined by a unique Id ; A is the attribute of Event M , $A = (\text{attr}1, \dots, \text{attr}n)$, $n \geq 0$; R is the EventSet that contains

the sets of all the operations OPRSet that caused the Event M , $R = (\text{EventSet}, \text{OPRSet})$; L is the location of Event M and can be expressed by the function boundingbox; and T is the occurrence time of Event M , real-time or interval events. In event monitoring, the attribute A contains at least the sensor Id and observation results.

The data samples collected by sensor nodes in the network have both autocorrection and cross-correlation. In return, for a measurement data series of the same physical quantity $X_1, X_2, \dots, X_N$, the sampling time is $T_1, T_2, \dots, T_N$, and then the correlation of sensor data can be written as follows:

$$R_x(k) = \sum_{i=1}^{N-k} X_i \cdot X_{i+k}, \quad (2)$$

where k is the sequence step. Although the time variable T does not appear in formula (2), the assumption of the formula is that the continuous data samples are sampled at equal intervals.

Similarly, for the two measurement data series with different physical quantities $X_1, X_2, \dots, X_N$ and $Y_1, Y_2, \dots, Y_N$, the sampling time is $T_1, T_2, \dots, T_N$, and then the cross-correlation of sensor data can be written as follows:

$$R_{xy}(k) = \sum_{i=1}^{N-k} X_i \cdot X_{i+k}, \quad (3)$$

where k is the sequence step. Cross-correlation reflects the situation where two random numerical variables change simultaneously.

3.2. Detailed Design of the Event Service Layer. The event service layer is the core component of the APES model. It is an intermediary between the data service layer and the application layer (consumer) and a bridge between the data service layer and the application layer, which distinguishes between the active and passive services. In this paper, we implement the function of the event service layer by constructing an ESM composed of three modules. The composition of the event service layer is shown in Figure 2.

3.2.1. Data Request and Parsing Module. The main functions of the module are as follows:

- (1) When the active service module or the passive service module is called, the request parameters transmitted by the active service module and the passive service module are encapsulated into an XML format request document conforming to the SOS specification and sent to the SOS
- (2) Parsing the key data in the XML format response document returned by SOS and returning the parsed data to the corresponding event service module

3.2.2. Active Service Module. The active service does not require the consumer to subscribe or send a request. After the consumer registers the personal information to the event

FIGURE 1: Overall architecture of the APES model.

service layer and applies for membership of the event provider, the event service layer will default the consumer to accept the active event service initiated by the event service middleware. The event filtering component in the active service module performs event filtering according to a preset event judgment expression based on the data returned by the data request and parsing module. If the event was established, the event notification component is invoked to send the alarm information. After the event notification component receives the request that the alarm information needs to be sent, the corresponding alarm information is dynamically generated according to the specific event in the request, and the event alarm information is pushed to the consumer through a short message or an e-mail.

3.2.3. Passive Service Module. The passive service is initiated by the consumer, i.e., the consumer is required to subscribe to the event of interest to the event service layer after the registration is completed. The passive service module consists of the event generation component, event filtering component, and event notification component. The event generation component dynamically generates an event discrimination expression according to the subscription condition of the consumer and transmits the logical expression to the event filtering component. The event filtering component filters the event according to the event discrimination expression based on the data returned by the data request and analysis module, and if the event is established, the event notifying component is called to send alarm information. The event notification component publishes the event alarm information to the user who subscribed to the event.

4. Implementation of ESM

4.1. ESM Design. ESM is a component between the data service layer and the application layer, which is the bridge between the above two parts. ESM is in the event service layer, which is the core component of this model. ESM is

composed of three parts, parsing module, active service module, and passive service module, which cooperate to complete all the functions of the event service layer. The composition of the ESM is shown in Figure 3.

4.1.1. Parsing Module. As shown in Figure 3, all functions of the parsing module are implemented by the parser. The parser is the bridge of information exchange between the data service layer and event service layer. The information exchange between active (passive) modules and the data service layer needs to be completed by the parser, and the parser is responsible for interacting with the SOS.

4.1.2. Active Service Module. The event service controller, event filter, and timer logic jointly realize data processing and event filtering of active service. The composition of the active service module is shown in Figure 4.

The event service controller controls the on or off specific event monitoring in the active service. That is to say, it mainly reflects the initiative of the event service by the event service controller and is not restricted to the received active service.

The timer logic is used to control the frequency at which event alert messages are sent. If the active service cycle T_s (data requests, document parsing, event filtering, and judgment after every T_s time) is short and events continue to occur for a period, middleware will push the same alert message to consumers every T_s time without timer logic components, which can result in wasted resources and poor user experience. This module in the timer logic component presets an alarm period T_0 ; when an alarm is issued, the next alarm interval should not be less than the time.

The event filter is responsible for the discovery of the event. It constructs an event filter expression based on the observed attributes of one or more sensors and uses the sensor observations parsed by the parser as the input to the

FIGURE 2: The composition of the event service layer.

FIGURE 3: The composition of the ESM.

expression. When the expression is established, the event which satisfies the condition will be generated, and the event alarm program will be triggered.

The initiator of the active service is the ESM, and the consumers passively accept the services from the ESM. When starting an active event service, the initial state of the event service controller will be turned on, and the event service controller sends a data request to the parser. After receiving the request, the parser first judges whether the parsed document corresponding to the request exists.

4.1.3. Passive Service Module. The passive service module consists of two parts: a user service controller deployed on the user's smart phone and an event filter deployed on the server. The composition of the passive service module is shown in Figure 5.

The user service controller is arranged on the smart phone of the user and is responsible for subscribing events to the event service layer to initiate an event service process. Once the event service subscription is successful, the filter of the event service layer starts a passive event service process

FIGURE 4: The composition of the active service module.

until the event is established, and an alarm notification is sent to the user. Visible, passive service is initiated by consumers, service enabled or not controlled by the user.

Event filter is the core of the passive event service, which is responsible for receiving user's event subscription, constructing event filter expression in real time, sending data service request to the parser, receiving the parsing result returned by the parser, and judging whether the expression is established according to these results. Once the expression is established, the event to which the user subscribes occurs, and an alert notification is immediately sent to the user.

When a consumer subscribes to the ESM service, the consumer's registration information is verified first. If the consumer has not previously registered personal information with the SOS service, there is no right to subscribe to the service because middleware only provides services for registered users. If the user has already registered, the service subscribed by the user will be sent to the middleware.

FIGURE 5: The composition of the passive service module.

4.2. Middleware Implementation

4.2.1. Data Request and Parsing Module Implementation.

A class named parser is created to implement the functions of the data request and parsing module, and Table 1 illustrates a detailed structure diagram of the class.

It can be seen from Table 1 that the class contains two member variables and two member methods. The access permission of the GetData method is public, which is mainly used by the outside world to obtain the data of the corresponding sensor. Invoking this method needs to pass in two parameters, namely, the ID number of the sensor and the observation attribute (offering) that needs to be obtained. Then, the method will generate an XML request document conforming to the SOS standard and send a request to the SOS. The SOS returns an XML-formatted document in response to the request. This method returns the document name as a string and stores it in the XMLDoc member variable. The XMLDoc variable access permission is private to prevent the external modification of the document to ensure the authenticity of the data and event judgment results. The access permission of the AnalyseData method is also public, which can be invoked by the outside class. The main function of this method is to parse the content of the returned XML document. The XMLDoc member property is passed to the method as an argument. The AnalyseData method uses the SAX parsing technique to return the parsing results in the string format and store them in the

TABLE 1: The structure of the parser class.

Member variables and member methods	Function
+ Result: string	Storing parsed data
– XMLDoc: string	Storing the document name of the return document by the SOS
+ AnalyseData(string doc): string	Parsing the documents returned by the SOS
+ GetData(string ID, string offering): string	Sending data request to SOS service

Class member access rights: + public; – private.

Result member variable. The access permission of the Result member variable is public which can be obtained by the outside world for filtering events. By the above four members of the class, the parser class implements the functions of request document generation, sending, and response document parsing.

4.2.2. Active Service Module Implementation. A class named *ActiveEventService* is created to implement the function of the active service module of the ESM, and Table 2 illustrates a detailed structure diagram of the class.

It can be seen from Table 2 that there are two member variables and two member methods in this class. The *CycleControl* method is used to control the thread loop process. If T_s is passed as a parameter to the method, all the methods in the *ActiveEventService* class will execute the cycle in T_s period. The *EventJudgement* method mainly filters events. When this method runs, it first calls the *GetData* method of the *Parser* class to request data and then calls the *AnalyseData* method to perform data parsing. Finally, a series of logical combinations are judged based on the data stored in the *Result* variable.

4.2.3. Passive Service Module Implementation. A class named *PassiveEventService* is created to implement the functionality of the passive service module for the ESM, and Table 3 illustrates a detailed structure diagram of the class.

It can be seen from Table 3 that the class has three member methods and two member variables. The *CycleControl* method is used to control the thread loops of this class. If you enter a time parameter of T_s , the methods in that class cycle through T_s . The *DynamicJudgement* method is triggered when a user subscribes to a passive service module for an event. The user's subscription conditions are passed in as arguments to the method, which parses the subscription and dynamically generates an event filter. Finally, the filter's logical discriminate expression is returned in the string format and passed to the *EventJudgement* method.

4.2.4. Alarm Notification Implementation. ESM provides two ways to send alerts, i.e., SMS and e-mail. Create *SMSSend* class and *EmailSend* class to realize the function of SMS notification and e-mail notification, respectively. There is only one implementation method in both classes. In this work, in the implementation of mail function by the *JDK* e-mail functions, the ESM just needs to send the message subject and content to the method in this class, and then mail

will be automatically generated and sent out through the SMTP protocol. The realization of SMS alarm function uses the third-party platform. By encapsulating the implementation interface given by the third-party platform, the ESM only needs to transmit the user number and the content of the received EMS to the method in the *SMSSend* class, and if the transmission is successful, the server receives the returned status code 200 in the background.

5. Experimental Results

Figure 6 demonstrates the system architecture diagram of the simulation experiment. It shows that there are sensor, server, and mobile smart device in the experiment.

As shown in Figure 6, PC1, PC2, and PC3 use the TCP protocol to transmit data, and PC4 and PC5 use the UDP protocol for data transmission. PC1 simulates a temperature sensor, PC2 simulates a humidity sensor, PC3 simulates a wind speed sensor, PC4 simulates a temperature sensor, and PC5 simulates a CO₂ concentration sensor. Both the SOS and the ESM are deployed on PC6. The client uses the Android-based mobile smart device to receive and display alarm information.

5.1. Active Service Simulation and Testing. The performance of the active service module is tested by abnormal weather warning. In the test, there are two conditions which can be defined as follows:

- (1) High temperature event: the temperature is higher than 35°C, which is a simple event involving real-time observations of only one sensor
- (2) Cold wave event: the temperature drops more than 8°C, and the current temperature is less than 4°C within 24 hours, which is a complex event, involving not only the real-time observation data of the sensor but also the historical data of the sensor

PC1 is used to simulate the temperature sensor with ID number temp_S-1-1, and a 96-size array is designed to store a set of temperature data representing the 48-hour real-time observations of the sensor, and then PC1 sequentially extracts one temperature data from the array at 30-minute intervals and sends it to the SOS. SOS receives this data and saves it to the database table. Table 4 demonstrates the part of the simulated temperature data. The first 24 hours will trigger a high temperature anomaly, and the second 24 hours will trigger a cold wave anomaly as the temperature drops.

TABLE 2: The structure of the ActiveEventService class.

Member variables and member methods	Function
– TimeFlag: Boolean	Identifying whether this method is executed
+ ServiceFlag: Boolean	Identification whether the time difference is greater than T_0
+ EventJudgement(): void	Event filtering
+ CycleControl (int time): void	Controlling the thread loop of the class

Class member access rights: + public; – private.

TABLE 3: The structure of the PassiveEventService class.

Member variables and member methods	Function
+ ServiceFlag: Boolean	Identifying whether this method is executed
+ ServiceTerminate: Boolean	Identifying whether the event service ends
+ EventJudgement (string judge): void	Event filtering
+ CycleControl (int time): void	Controlling the thread loop of the class
+ DynamicJudgement (string request): string	Dynamically generating the event filter condition

Class member access rights: + public; – private.

FIGURE 6: The system architecture diagram of the simulation experiment.

After the active service starts, the middleware sends a request to the SOS to obtain the latest observation data, the SOS returns the first real-time temperature (15°C) (serial number: 7801) to the middleware, and the middleware performs event filtering; no high temperature event occurs because the temperature is lower than 35°C. Thereafter, the middleware performs data request and event filtering in a 30-minute cycle. The temperature rises gradually as time goes on. When the temperature reaches 36°C (serial number: 7822), a high-temperature abnormal event is triggered, and the active event service module successfully sends the alarm information to the consumer by SMS and e-mail, respectively, as shown in Figure 7.

After 24 hours, the simulated temperature gradually drops. When the temperature obtained by the sensor is 8°C

(serial number: 7855), the condition of “temperature drop over 8°C in 24 hours” has been met. However, the current temperature has not reached below 4°C, and the cold wave weather alarm is not triggered by the system. With the passage of time, when the temperature obtained by middleware is 3°C (serial number: 7895), the second condition of cold wave weather is met, and the cold wave weather alarm is triggered at this time. The active event service module successfully sends the alarm information to the consumer by SMS and e-mail, as shown in Figure 8.

5.2. Passive Service Simulation and Testing. In the test, the condition of the fire event can be defined as follows:

- (1) The temperature rises rapidly above 80°C ($t > 80$)

TABLE 4: Simulated partial temperature data.

Sensor ID	Observation value (°C)	Observation serial number
Temp_S-1-1	15	7801
Temp_S-1-1	16	7802
Temp_S-1-1	19	7803
...	...	...
Temp_S-1-1	30	7820
Temp_S-1-1	32	7821
Temp_S-1-1	36	7822
Temp_S-1-1	34	7823
...	...	...
Temp_S-1-1	10	7854
Temp_S-1-1	8	7855
Temp_S-1-1	6	7856
...	...	...
Temp_S-1-1	3	7895
Temp_S-1-1	1	7896

FIGURE 7: The alarm information of high temperature weather by SMS and e-mail.

FIGURE 8: The alarm information of the cold weather event by SMS and e-mail.

TABLE 5: Simulated partial temperature and carbon dioxide concentration data.

Sensor ID	Observation value (°C)	Observation serial number	Sensor ID	Observation value (°C)	Observation serial number
Temp_S-1-2	20	8801	Carb_S-1-1	400	8801
Temp_S-1-2	20	8802	Carb_S-1-1	389	8802
...	...	...	...	...	...
Temp_S-1-2	48	8840	Carb_S-1-1	760	8840
Temp_S-1-2	57	8841	Carb_S-1-1	840	8841
...	...	...	...	...	...
Temp_S-1-2	116	8884	Carb_S-1-1	1480	8884
Temp_S-1-2	134	8885	Carb_S-1-1	1920	8885
...	...	...	...	...	...
Temp_S-1-2	186	8895	Carb_S-1-1	2140	8895
Temp_S-1-2	192	8896	Carb_S-1-1	2420	8896

FIGURE 9: The alarm information of the fire event by SMS and e-mail.

- (2) The CO₂ concentration exceeds 2000 ppm ($c > 2000$) in five minutes

The temperature sensor with ID number temp_S-1-2 and the carbon dioxide sensor with ID number Carb_S-1-1 installed in a user's home are simulated by PC4 and PC5, respectively. The user first registers the two sensors in the SOS and subscribes the fire event service to the ESM according to the fire event subscription conditions. As shown in Table 5, two arrays of size 120 were designed to store a simulated set of temperature and carbon dioxide concentration data (5 s/time data acquisition cycle) monitored over 5 minutes, respectively, and the two sets of data were synchronously sent to the SOS at the same cycle in an associated database.

After the service begins, the middleware requests the SOS for the latest observation data at a frequency substantially synchronized with the SOS acceptance data and constructs an event filter according to the user subscription conditions for event filtering. Initial temperature is 20°C, carbon dioxide concentration is about 400 ppm (serial number: 8801), belonging to the normal range, and no alarm occurs. When there are objects burning in the room, the indoor temperature and carbon dioxide concentration increase significantly. When the monitored temperature reaches 116°C, the first condition of the fire event (temperature difference $t > 80^\circ\text{C}$ in 5 minutes) is met, but the carbon dioxide concentration has not reached the warning condition (serial number: 8884), and then there is no event alarm which is triggered. When the monitored temperature and the carbon dioxide concentration reach 186°C and 2140 ppm, respectively, two conditions of a fire event are met, an alarm is successfully triggered, and the passive service module sends alarm information to a consumer by SMS and e-mail, as shown in Figure 9.

6. Conclusions and Future Work

In this paper, we propose a novel event service framework named APES based on the sensor web for the elimination of "Information Island" and low reusability in the whole network operation and maintenance system. In the model, there are perception layer, data service layer, event service layer, and application layer, and the core part is the ESM module. Finally, taking abnormal weather and fire warning as an example, the performance of the ESM module is tested, and the experimental results verify the effectiveness of the proposed method and the practicality of the ESM.

Meanwhile, the SOS information is used as the data source of event service; that is, the front-end sensor does not interact directly with the ESM module to realize event monitoring but sends the collected data to the SOS and stores in the database. To some extent, it results in the delay of event alarm and affects the real-time performance of event discovery. In addition, the performance of the proposed method is tested in the experimental environment instead of real sensor data. In the future, we will exploit APES for a wider range of tests, such as the interactive mode of data and the real sensor web data environment.

Data Availability

The data used to support the findings of this study are included within the article.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

The authors appreciate the China Vision Cloud platform of Shanghai University and Shanghai Engineering Research Centre of Intelligent Computing System. This work was partially supported by the Science and Technology Commission of Shanghai Municipality (nos. 21142202400 and 19142201600) in China and Graduate Innovation and Entrepreneurship Program in Shanghai University in China (no. 2019GY04).

References

- [1] L. Lan, B. Wang, L. Zhang, R. Shi, and F. Li, "An event-driven service-oriented architecture for internet of things service execution," *International Journal of Online Engineering*, vol. 11, no. 2, pp. 68–73, 2015.
- [2] Y. Huang, H. Xu, H. Gao, X. Ma, and W. Hussain, "SSUR: an approach to optimizing virtual machine allocation strategy based on user requirements for cloud data center," *IEEE Transactions on Green Communications and Networking*, vol. 5, no. 2, pp. 670–681, 2021.
- [3] X. Zhou, X. Xu, W. Liang, Z. Zeng, and Z. Yan, "Deep-learning-enhanced m detection for end-edge-cloud surveillance in smart IoT," *IEEE Internet of Things Journal*, vol. 8, no. 16, pp. 12588–12596, 2021.
- [4] M. Sorrentino, V. Cirillo, D. Panagrosso, A. Trifirò, and F. Bedogni, "Development of free-cooling detection procedures to support energy intelligence actions within telecommunication environments," *Applied Thermal Engineering*, vol. 144, no. 1, pp. 1037–1048, 2018.
- [5] C. Rodríguezdomínguez, K. Benghazi, M. Noguera, J. Garrido, M. Rodríguez, and T. RuizLópez, "A communication model to integrate the request-response and the publish-subscribe paradigms into ubiquitous systems," *Sensors*, vol. 12, no. 6, pp. 7648–7668, 2012.
- [6] M. Fan, H. Fan, N. Chen, Z. Chen, and W. Du, "Active on-demand service method based on event-driven architecture for geospatial data retrieval," *Computers & Geosciences*, vol. 56, no. C, pp. 1–11, 2013.

- [7] C. Guo, J. Liu, and P. Zou, "The study and implementation of active service based on CORBA event service," *Journal of National University of Defense Technology*, vol. 21, no. 4, pp. 83–86, 1999.
- [8] X. Liu, F. Ye, Y. Liu, X. Xie, and J. Fan, "Real-time forecasting method of urban air quality based on observation sites and Thiessen polygons," *International Journal on Smart Sensing and Intelligent Systems*, vol. 8, no. 4, pp. 2065–2082, 2015.
- [9] X. Ma, H. Xu, H. Gao, and M. Bian, "Real-time multiple-workflow scheduling in cloud environments," *IEEE Transactions on Network and Service Management*, vol. 18, no. 4, pp. 4002–4018, 2021.
- [10] X. Zhou, X. Xu, W. Liang et al., "Intelligent small object detection for digital twin in smart manufacturing with industrial cyber-physical systems," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 2, pp. 1377–1386, 2022.
- [11] Y. Sun, T. Zhao, E. Li et al., "Radiometer calibration of airborne L-band active and passive microwave detector," *National Remote Sensing Bulletin*, vol. 25, no. 4, pp. 918–928, 2021.
- [12] R. Liu, N. Li, G. Wang et al., "Structure design and analysis of airborne active and passive microwave sounder," *Electro-Mechanical Engineering*, vol. 36, no. 1, pp. 14–21, 2020.
- [13] X. Y. Qian, Q. W. He, D. Y. Li, and C. Huang, "Research on the design and test of a novel hybrid vibration isolator," *Ship Electronic Engineering*, vol. 39, no. 6, pp. 176–187, 2019.
- [14] Y. Xu, Y. Wu, H. Gao, S. Song, Y. Yin, and X. Xiao, "Collaborative APIs recommendation for artificial intelligence of things with information fusion," *Future Generation Computer Systems*, vol. 125, no. 1, pp. 471–479, 2021.
- [15] X. Zhou, X. Yang, J. Ma, and K. I. K. Wang, "Energy efficient smart routing based on link correlation mining for wireless edge computing in IoT," *IEEE Internet of Things Journal*, vol. 8, p. 1, 2021.
- [16] B. Arne, E. Johannes, J. Simon et al., "New generation sensor web enablement," *Sensors*, vol. 11, no. 3, pp. 2652–2699, 2011.
- [17] N. Chen, L. Di, G. Yu, and M. Min, "A flexible geospatial sensor observation service for diverse sensor data based on web service," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 64, no. 2, pp. 234–242, 2009.
- [18] H. Gao, Y. Zhang, H. Miao, R. J. D. Barroso, and X. Yang, "SDTIOA: modeling the timed privacy requirements of IoT service composition: a user interaction perspective for automatic transformation from BPEL to timed automata," *Mobile Networks and Applications*, vol. 26, 2021.
- [19] J. B. Liang, F. S. Tian, C. Jiang, and T. S. Wang, "Survey on task offloading techniques for mobile edge computing with multi-devices and multi-servers in internet of things," *Computer Science*, vol. 19, no. 1, pp. 17–24, 2020.
- [20] S. Jirka, A. Bröring, P. Kjeld, J. Maidens, and A. Wytzisk, "A lightweight approach for the sensor observation service to share environmental data across Europe," *Transactions in GIS*, vol. 16, no. 3, pp. 293–312, 2012.
- [21] J. Pradilla, M. Esteve, and C. Palau, "SOSFul: sensor observation service (SOS) for internet of things (IoT)," *IEEE Latin America Transactions*, vol. 16, no. 4, pp. 1276–1283, 2018.