An Efficient Constraint Mapping-based Fault-causing Process Localization Method for Intelligent Dispatching Systems of Smart Grid: Case Study

Xinpeng Li[1,2], Yuexing Peng[3], Guangjie Han[4,5], Hanxu Sun[1], Bo Yan[2]

[1] School of Automation, Beijing University of Posts and Telecommunications, China
[2] Dispatching and Control Center, State Grid Jibei Electric Power company Limited, China
[3] Key Lab of Universal Wireless Communication, MoE, Beijing University of Posts and Telecommunications, China
[4] College of Engineering, Nanjing Agricultural University, China
[5] Department of Information and Communication Systems, Hohai University, China
Xinpeng_li@126.com, yxpeng@bupt.edu.cn, hanguangjie@gmail.com, hxsun@bupt.edu.cn, yan.bo.c@jibei.sgcc.com.cn

Abstract

As the next generation power grid, smart grid, utilizing modern information technologies, is able to deliver power more efficiently and response to wide-ranging conditions and events. Intelligent dispatch system (IDS) is regarded as the core link of the smart grid to control all the systems, and then its smooth running is vital to the smart grid. With numerous functionalities including data aggregation, data analysis, and decision making, the IDS platform is of high complexity, and then the reliable online fault diagnosis of the IDS platform becomes a big challenge when the IDS platform becomes too complicated to avoid faults such as functionality logic fault (FLF), software bug (SB), and hardware fault (HF). In order to decouple the services of IDS for easing understanding, transplanting, and debugging, the software design and implementation are under the principle of modular design and independent service calling. However, due to the sharing of the software and hardware resources of the platform, the running services are coupled tightly, which calls for efficient and reliable fault-causing process localization methods. In this paper, a constraint mapping-based method is proposed for efficient localization of the fault-causing process. Firstly, the complicated interactions among processes are analyzed and categorized into inter-service and intra-service constraints, where inter-service constraints include database operation constraint (DOC), software resource constraint (SRC), and hardware resource constraint (HRC), wihle the intra-service constraint is summarized as functional logic constraint (FLC). Then corresponding constraint mapping tables can be constructed for the IDS platform. Originating from the abnormal alerts of processes and fault messages of services, the fault-causing process localization is decomposed into the track of source abnormal process within each service and the localization of fault-causing process among the services regarding intra- and inter-constraint mapping tables, respectively. At last, the fault and alarming logics are reproduced in order to achieve both verification of the result of the fault-causing process localization and acquisition of new knowledge from the fault-causing logics. Case studies are presented, and the results show the efficiency of the proposed method.

Keywords: Fault-causing process localization, Intelligent dispatch system, Smart grid

1 Introduction

The power grid is an inter-connected network that transports the generated electricity from generators to consumers. By utilizing modern information technologies, smart grid (SG), regarded as the next generation power grid, is capable of delivering power in more efficient ways and responding to wide-ranging conditions and events. More specifically, the SG is regarded as a `smart' electric system that makes full use of modern communication technologies and computational intelligence in an integrated fashion across electricity generation, transmission, substations, distribution and consumption to achieve a system that is safe, secure, reliable, resilient, efficient, and sustainable [1].

Intelligent dispatch system (IDS) is the control system of the SG with functionalities of data aggregation, data analysis, and decision-making [2]. There are fruitful researches on the data aggregation [3-6], data analysis, fault diagnosis, and security for the smart grid [7-12, 22].

IDS platform is a complicated software and hardware platform, on which the IDS functionalities are performed. It is vitally important to ensure the smooth running of the IDS platform, so that a monitoring system is a must to monitor the running state of the IDS platform by abnormality early warning,

*Corresponding Author: Guangjie Han; E-mail: hanguangjie@gmail.com
DOI: 10.3966/160792642019092005010

fault-causing process localization, and troubleshooting. For easing functionality expansion, transplanting, modifying, debugging, and understanding, the IDS is designed in an architectural and modular fashion, that is, a complicated functionality is decomposed into several simple and independent services. Moreover, the software implementation accords to isolation principle, i.e, a common service supporting several functionalities will be generated multiple independent versions such that they are isolated from each other. In this way, the IDS software is decoupled to facilitate the fault diagnosis for the monitoring system. However, the IDS software is running on the hardware platform, and the sharing of hardware resources (i.e., CPU, memory, hard disk, and network bandwidth) introduces complicated coupling among the running services, which raises a big challenge for fault-causing process localization. Compared to the fruitful fault diagnosis in smart grid, the research on the fault-causing process localization for the IDS platform is not sufficient. Roughly, the existing fault-causing process localization methods can be categorized into three types.

Knowledge-based method. Based on the knowledge of functionality logics and the software implementation, inference engine is employed to analyze the fault generation logics. Knowledge-based methods include thresholding-based method, fault-tree method, fuzzy logic, expert system, and so on. Thresholding-based method is simple but efficient in alarming and fault-causing process localization. When a parameter of a process exceeds a preset threshold, an alert is initiated. For example, an active process inevitably occupies memory. If the available memory is less than a threshold so that this process cannot work properly, it will trigger off an abnormal alert to show the shortage of memory. Clearly, if the threshold is set accurately, the fault can be traced efficiently. The thresholding-based method is widely deployed in equipment working condition monitoring, such as in [13]. However, it is of big challenges for applying this method into the IDS platform. Since all active processes running at the same network element share the hardware resources, i.e., memory, central processing unit (CPU), one resource shortage fault will cause avalanche type chain reaction of alarms because other running processes will be also short of hardware resource. One real case is: a process periodically calculates some intermediates for other services, which may require a huge memory to store the data and speed up the calculation. When this process has a syntax error so that it does not release the memory after the calculation, the memory will be used up rapidly, which causes the memory shortage alerts for all other active processes. Moreover, all active processes may have to postpone their proceedings due to the shortage of memory, which results in the burst of other types of abnormal alerts. In this case, avalanche type alerts are observed, but it is very hard to localize the source fault-causing process by the thresholding-based method solely due to the tight coupling of the processes in the form of hardware resource sharing. Fault tree method is based on the rules which are originated from experts' knowledge, i.e., timing relationship between processes and reasonable region of a key parameter, then the inference engine uses rules to infer conclusions with respect to certain faults. Fault tree method is widely applied to localize the fault-causing process within a service, but it becomes infeasible for large-scale coupling system like IDS platform due to the exponentially increased complexity. Fuzzy logic is a method to partition a feature space into fuzzy sets and utilize fuzzy rules for reasoning [14]. Fuzzy logic has been applied successfully for fault diagnosis [15-16]. The knowledge based expert system methods can provide insight for the IDS, however, its performance is limited by the expert knowledge and known rules. The ever-increasing expansion and updating of the IDS and the obscure coupling among the services make the solo use of knowledge-based methods degrade severely in the fault-causing process localization of the IDS. Naturally, the knowledge-based methods are combined with other techniques to overcome the problem of large scale and high coupling.

Software testing methods. Before the practical application, software testing is a must for the IDS. From the testing level, there are unit testing, integration testing, system testing, and operational acceptance testing [17]. As for testing methods, there are static testing, dynamic testing, and the box approach which includes white-box testing, black-box testing, and gray-box testing [18]. However, even after strict software testing, software errors are inevitable for the tightly coupled IDS. With the expansion of the grid and the increase of functionality, the IDS needs constant updating, which severely hardens the software testing. In other words, software testing cannot avoid the software errors of the IDS, and what is more severe for software testing methods is that they cannot localize the fault of IDS on the spot.

Machine learning methods. Owing to its powerful ability of nonlinear approximation and adaptive learning, machine learning has been the most well-established data-driven fault diagnosis tool for hardware and network [19-21]. When there are enough fault localization samples to illustrate the connection between the source fault-causing process and the following alerts, machine learning method is an efficient tool to model the obscure relation between the input and the output instead of modeling the cause and the effect of the fault. When the fault examples or the monitored parameters are not sufficient, however, machine learning method degrades badly. In the case of the IDS platform, the fault samples usually are seldom, which severely hinders the application of machine learning methods in the fault localization of the IDS.

Concerning the big challenges on the fault-causing process localization of the complicated IDS platform, in this paper we study the constraint mapping-based method to localize the fault-causing process. It is notable that the localization of faults caused by functionalities of the smart grid is out of the scope of this method. The main contributions include:

- The complicated couplings among the processes and services in the IDS system are described and analyzed, and most critical constraints are then refined and categorized into inter-service constrains and intra-service constrains, where inter-service constraints include database operation constraint (DOC), hardware resource constraint (HRC), and software resource constraint (SRC), and intra-service constraint is summarized as functional logic constraint (FLC). Based on the constraint types of services, four classes of constrain mapping tables can be constructed for the IDS platform, which are the basis of the proposed method.
- An efficient fault-causing process localization method is proposed, which is divided into two phases: (*i*) intra-service tracing phase traces the source abnormal process within every service with alerts based on the intra-service constraint mapping tables; (*ii*) inter-service localization phase localizes the source fault-causing process from the set of traced source abnormal processes on the base of inter-service constraint mapping tables.
- The alarming logic of the fault event is reproduced to achieve both the verification of the fault-causing process localization result and discovery of either new knowledge or software bug.

The rest of paper is organized as follows. Section II, the IDS platform, including its software and hardware architecture, functionalities, and causes and types of couplings among services, are presented. In Section III, the proposed decoupling-based fault-causing process localization method is detailed, and practical case is studied to illustrate the effectiveness of the proposed method in Section IV. Section V concludes the paper.

2 IDS Platform and Constraint Mapping

2.1 IDS Software and Hardware Platform

The IDS platform is a complicated software and hardware systems. The software architecture can be layered to five layers, i.e., operation system layer, data storage layer, transport layer, public service layer, application layer. Based on the layered architecture, the IDS software platform provides functionalities in a manner of decoupling via (*i*) layered architecture, modular design, and functionality abstraction mechanism, i.e., functionality-service-process, for easy transplanting and understanding; (*ii*) independent calling, that is, independent new public service is initiated for a new caller to ensure the independent running of multiple implementations of the same services; (*iii*) message and service bus mechanism for message transferring between services in order to avoid direct calling. In this way the IDS software platform features: (*i*) complicated functional logic among the services and processes due to the inner interconnection among functionalities; (*ii*) independent running of services as the result of independent calling of the common services by functionalities.

The hardware platform of the IDS consists of four parts, i.e., data storage and database management by the storage array and data server; data processing by the application server; networking by routers, switches, and firewalls; and data visualization, decision making, and dispatching by command & dispatch center. The layout of the IDS hardware has much to do with the software implementation, and affects the coupling of the services. The features of the hardware platform of IDS include: (*i*) server cluster-based parallel computing mechanism to greatly improve the processing capability for ever-increasing traffic data in smart grid; (*ii*) hot backup-based redundancy mechanism for robustness of the IDS; and (*iii*) distributed data storage mechanism for bandwidth save and quick response.

Although the isolated mechanism is deployed for the software platform of IDS, the sharing of hardware resource of IDS introduces complicated coupling among processes and services, which is detailed in the following section. In summary, two big challenges are arose for the fault-causing process localization in the IDS, i.e., (*i*) the sharing of both software and hardware resource of IDS induces tight coupling among processes and services, which makes it impossible to exhaust all possible coupling among processes and services for knowledge-based inference methods; (*ii*) besides those services with routine timing, there are lots of event-driving services in IDS, which results in the dynamic running of services and processes. Consequently IDS requires real-time monitoring and fault localization of the IDS, which causes a big problem for software testing methods; (*iii*) After careful software testing before practical implementation of IDS, the fault events, though inevitable, are seldom. In this case, machine learning-based methods degrade severely.

2.1.1 Coupling Analysis and Constraint Mapping

The complicated coupling among the processes of the IDS services is the result of software and hardware resources sharing of the IDS platform. More specifically, from the viewpoint of the layered architecture of IDS software, the following operations will cause the coupling of processes. To ease the coupling analysis, the function of real-time visual monitoring of a key parameter is set as an example. As

illustrated in Figure 1, there are four services running at three servers: data aggregation service at the front end server (FES), data harvest service and data visualization service at the managing server, data transport service at the data server. Firstly, data aggregation service calls data harvest service via service bus to collect data. At receiving the data request, data transport service locates the data and then transmits the requested data via message bus to data harvest service. These monitored data are written to the real-time database and then are transferred to the history database by data transport service. At last, data visualization service reads the data from the history database and then transfers them to human-machine interface for visualization. The following operations will cause coupling of processes.

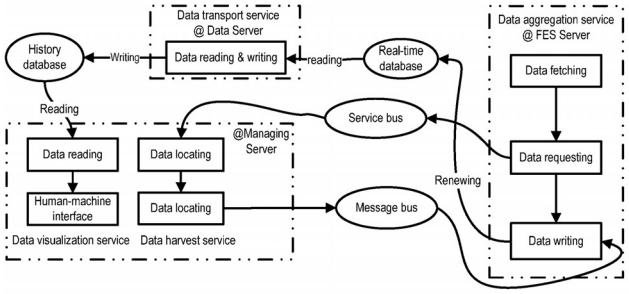


Figure 1. Illustration of the coupling among processes of three services

- Collaboration among various services to support a function. As illustrated in Figure 1, the function of real-time visual monitoring of a key parameter requires the collaboration of four services. In this case, timing constraint exists among the processes of these services.
- Database reading and writing. As illustrated in Figure 1, *data writing* process of the data aggregation service at the FES server renews the RT database, and then *data reading & writing* process of the data transport service at the data server writes the history database with the data reading from the real-time database, and finally *data reading* process of the data visualization service at the managing server reads the history database and outputs the monitored data to human-machine interface for visualization. All data writing and reading processes have to occupy the read-write lock before their reading/writing action, and then the read-write lock becomes a constraint for these processes. From this simple example, it is clearly that the database reading and writing connects several services.
- Public service calling via service bus and data/message transportation via message bus. These two types of connections are also depicted in Figure 1. More specifically, the data aggregation service at the FES server is responsible for the renewing of the monitored data in the real-time database, which is achieved by calling data harvest service at the managing server via service bus, and the data harvest service collects the requested data and feeds them back to the data aggregation service via a message bus. At receiving the requested data, the data writing process of the data aggregation service writes the monitored data into the real-time database to renew the data. Consequently the data transport service copies the monitored data and stores the copy at the history database, which is read and then visualized by the data visualization service at the managing server.

The deployment of IDS software on a hardware platform will also introduce complicated coupling among the processes of services. Some typical causes are listed below, which also illustrate the hardware-related coupling types.

- The ever-increasing massive data are always stored in a distributed way with redundancy to ensure data security. In the fog computing aided cloud storage mode, a large amount of data will be stored at the network edge which is near to the data generators so as to facilitate local intelligence and reduce the bandwidth consumption. Then the data access might visit multiple data servers.
- Hot backup mechanic is a standard technique for the stability of the IDS. Some key application servers may have several backup servers, and this hot backup mechanic introduces bi-direction communication between the host server and the backup ones. Although the same service is running at both master and backup servers, but there are often constraints on the service, i.e., only the service at the master server can write data to the database. Clearly, these constraints introduce coupling among the independent services due to the backup mechanism.
- With the rapid expansion of the grid scale and its functionalities, the IDS scale increases correspondingly. Then more hardware resources are required, including computing, storage, and networking. Parallel computing is inevitable, which coordinates the resources to fulfil the tasks efficiently, but at the same time it couples the services involved in the parallel computing.
- In the large-scale IDS, networking becomes more important, which introduces interaction among the connected servers.
- All processes running on the same network element (i.e., server, router) will share the hardware resources, like CPU, memory, hard disk, and networking bandwidth. By sharing the hardware resources, all active processes are coupled. Since the hardware resources have hard constraints, excessive occupancy of hardware resources by one process may cause avalanche type chain reaction of alerts due to the shortage of resource to fulfil their tasks.

From the knowledge graph and the deployment of the IDS, couplings among processes and services can be categorized into four types of constraints, i.e., the functional logic constraints of services (FLC), the hardware resource constraints of network elements (HRC), the software resource constraints of IDS (SRC), and the database operation constraints of the IDS (DOC).

FLC. The implementation of a service requires the reasonable assembling and the fulfilling of some rules for the ensemble of the involved processes. There are mainly two kinds of FLC, i.e., the order of operation constraint and the value range constraint. Traditionally the operation order of processes includes sequential, parallel, and conditional selective operations in both event-driving and periodic operation mode of the services. The order of operation constraints depends on both functional logic of functionality and software implementation mode, and they keep constant when the IDS software is implemented. Value range constraints, such as the state query should be finished within one second, are always determined by the expert knowledge of the IDS functionalities, and often reflected as the threshold in anomaly detection. Different from the order of operation constraints, which can be determined by the logic of the software, the value range constraints are variable according to the hardware capacity.

HRC. The hardware resource of a network element (i.e., server, workstation, router, switch, gateway) mainly includes the CPU, memory (both physical and virtual memory), hard disk, and network bandwidth. An active process would occupy some resources, and the limits are usually predetermined as the threshold for anomaly detection to ensure the smooth running of the service and the available resource of the hardware.

SRC. SRC in the IDS is the communication constraints, where there are two types of bus mechanisms for communication. The first one is the inter-server communication. For example, when two servers collaborate to fulfill a service, the processes running in different servers will communicate with each other via message bus. The other one is the inter-service communication. As shown in Figure 1, when a service needs to call a public service, it will send a call message via service bus to the public service. The SRC mainly consists of the communication constraints on the query length and the delay spread.

DOC. In IDS, there are history database and real-time database, and all data reading and writing operations are correlated with these two databases. For safe, there are always constraints on the read-write lock of the sheet, operation concurrency of a database, and length of the database operation waiting queue, which contribute to DOC.

In order to facilitate the localization of the fault-causing process by our proposed method, these four types of constraints can be further categorized into intra- and inter-service constraints. When the processes involved in a constraint belong to different services, this constraint then can be split into several intra-service constraints. This splitting operation greatly increases the number of constraints, but it is necessary for the proposed method because the FLC is mainly single service based, and classic knowledge-based fault diagnoses methods are also single service-based. On the other hand, as stated above, HRC, SRC, and DOC are the main cause of inter-service coupling in the IDS. These inter-service constraints are the base of final fault-causing process localization.

After the categorization of coupling types of processes, four types of intra- and inter-service constraint-mapping tables can be further constructed, which act as the basis of the proposed method and will be explained in the following two sections.

3 Constraint Mapping-based Fault-causing Process Localization Method

In order to ease the understanding of the proposed method, a practical case is studied as an example to explain the four steps of the proposed method.

3.1 Fault Case: A Monitored Parameter Stops Refresh

Supervisory control and data acquisition (SCADA) application is responsible for the online RT monitoring of key grid parameters. In this case, the fault is that one of key parameters stopped refreshing during 17:45-18:00 and 19:19-19:41 at the SCADA host server, but this key parameter tele-metering service worked properly at the SCADA backup server. The flow chart of the related services at the host server during the faulting period is presented in Figure 2. Specifically, the procedure of the related services is that: service 1 performs periodical data reading and plotting by calling public service 2 to read data from the RT database, while service 4 performs model updating with the aid of public service 3 to download the model from the RT database. Obviously there are two independent functionalities: reading the value of the key parameter and showing it periodically; installing a new model and broadcasting the model renewing messages to other node. These two functionalities are coupled via DOC and HRC.

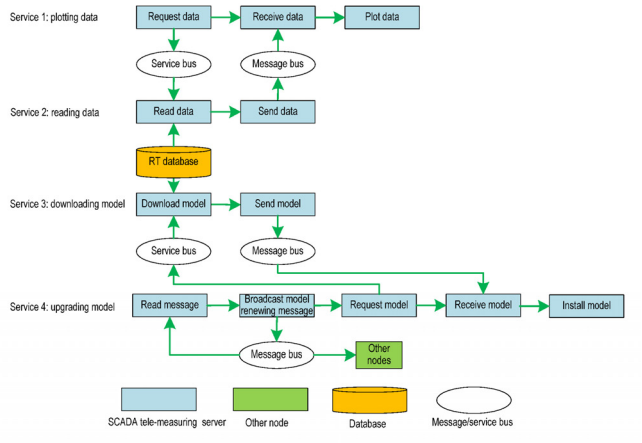


Figure 2. Flow chart of tele-metering data refresh-related services

When the monitored parameter stops refresh, three types of alert messages are reported, which are listed below.

- Database alerts: too many read-lock occupation requests in the waiting list of the RT database from service 2 and 5 ; the read-lock of RT database is held by the service 4 for long time.
- Software resource alerts: message bus is crowded by too many model renewing message from *broadcast message* process of service 3; service bus is crowded by *calling model download* process of service 3.
- Function logic alerts: *receive data* process and *plot data* process of service 1 are delayed; *read data* process and *send data* process of service 2 are delayed; *receive model* process of service 4 is delayed.

3.2 The Proposed Method

Based on the functional logics of services and the monitored fault and alert messages, the proposed method localizes the fault-causing process via four steps: (*i*) constructing process state mapping tables according to the intra- and inter-service constraints; (*ii*) source abnormal process tracing within every service based on intra-service constraints; (*iii*) fault-causing process localization among the set of source abnormal processes based on inter-service constraints; (*iv*) the fault and overall alarming process reproduction for both verification of the fault-causing process localization result and new knowledge discovery. The flow chart of the proposed method is shown in Figure 3.

We take the above case as an example to explain the four steps of the proposed method.

Step 1: Constructing fault tree model of every service of the IDS platform. Fault tree model will be employed to trace the intra-service source abnormal process, and it is also the basis of FLC mapping table construction. We firstly construct the fault tree model for every service based on the expert knowledge of IDS software and hardware platform.

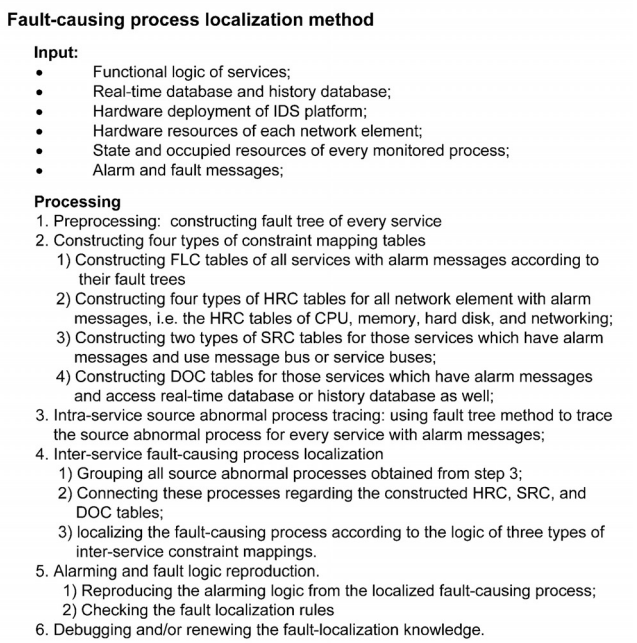


Figure 3. Block diagram of the procedures of the proposed fault-causing process localization method

Step 2: Constructing intra- and inter-service constraint mapping tables. Guided by the knowledge of IDS software and hardware platform, intra- and inter-service constraint mapping tables can be constructed.

Firstly, functional logic constrain (FLC) of the related four services are constructed in the form of fault tree model and illustrated in Figure 4, which facilitates the tracing of source abnormal process within service by fault tree method.

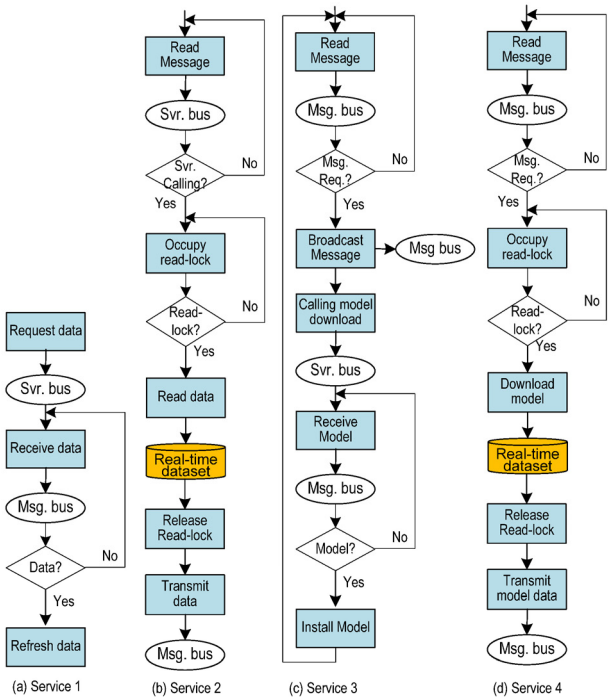


Figure 4. Fault tree models for the services related to the fault that a monitored data stops refreshing

Then the HRC, SRC, and DOC-mapping tables are constructed according to the resources that the related services occupy. These three types of constraints are summarized below.

HRC. The hardware resource of a network element (i.e., server, workstation, router, switch, gateway) mainly includes the CPU, memory (both physical and virtual memory), hard disk, and network bandwidth. An active process would occupy some resources, and the limits are usually preset as the threshold for abnormal alarming. The HRC mapping tables are constructed and updated for every network element timely, and Table 1 depicts the CPU-type HRC mapping table. It is natural to construct memory, hard disk, and network bandwidth-type HRC mapping tables, which are omitted here.

Table 1. CPU-type HRC mapping table at backup server

Process	Service	Ratio	Constraint	State
Request data	1	0.01	0.05	Normal
Receive data	1	0.02	0.05	Normal
Refresh data	1	0.01	0.05	Normal
Read message	2	0.01	0.05	Normal
Occupy real-lock	2	0.01	0.05	Normal
Read data	2	0.01	0.05	Normal
…	…	…	…	…

SRC. SRC is the networking type including service bus and message bus. From the functional logics of the related services, it is easy to construct SRC table, which is listed in Table 2.

Table 2. SRC mapping table at backup server

Req. Proc.	Req. Srv.	Resp. proc.	Resp. Srv.	Bus	State
Req. data	1	Read Msg	2	Srv.	Normal
Send data	2	Rec. data	1	Msg.	Normal
Req. model	4	DL model	3	Srv.	Normal
Send model	3	Rec. model	4	Msg.	Normal
B.C model	4	Read Msg.	5	Msg.	Normal

DOC. DOC includes the read-write lock of the sheet, operation concurrency of a database, length of the database operation waiting queue. The DOC mapping tables can be constructed accordingly as shown in Table 3. Notable that there are several independent copies of services 4 at the backup server.

Table 3. DOC mapping table at backup server

Req. Proc.	Req. Srv.	Ratio	State
Occupy read-lock	2	0.01	Delayed
Read data	2	0.02	Delayed
Occupy read-lock	4	0.01	Delayed
Download model	4	0.01	Delayed

Then HRC, SRC, and DOC mapping tables are constructed, which are summarized below.

Step 3: Intra-service source abnormal process tracing. Firstly, all alert messages are arranged into their belonged services, and then the intra-service HLC-based source abnormal process is traced within every service with alerts according to the inter-service constraint mapping tables and alerts. Here the fault tree method is employed such that the source abnormal process is traced on the basis of fault tree models for the four related services, which are shown in Figure 4. Specifically, for service 1, both *request data* and *receive data* processes are reported with alert of "delayed", but the *receive data* process is traced as the source abnormal process due to their functional logic. For service 2, among the three processes with alerts, *occupy read-lock* process is traced as the source abnormal process. In service 3, too many messages in message bus is due to *broadcast message* process sends too many model renewing messages via message bus, then the *broadcast message* process is traced as the source abnormal process. For service 4, *occupy read-lock* process is obvious the source abnormal process due to its taking read-lock a long time.

Step 4: Inter-service fault-causing process localization. When the source abnormal processes have been traced for all services with alerts, the fault-causing process can be localized via the coupling relationship among the services in the form of inter-service constrain mapping tables. In the concerned case, the four services are coupled by the RT database reading. From the DOC mapping table, the *broadcast message* process of service 3 is localized as the fault-causing process, because it broadcasts too many model renewing messages with result of the read-lock of the RT database is took by the *download model* process of service 4, which further delays the data reading of service 2.

Step 5: Alarming and fault logic reproduction. In order to verify the fault localization, the overall alarming process and the fault event are reproduced, that is, how does the fault-causing process produce the overall alerts. This overall alarming process, as well as the fault-causing process, would be visualized to help the understanding of the IDS.

The logic of the fault event is depicted in Figure 5, that is, the upgrading model service at the host server broadcasts the model upgrading message to all servers via message bus. After receiving this message again, the host server calls *downloading model* service 3 to download the model consequently. As such there are several *download model* processes, which hold the read-lock of the RT database for a long time. At the same time, the *read data* process of service 2 is delayed because it cannot hold the read-lock thus cannot read the RT database within its reading period, which causes the interruption of the monitored data refresh.

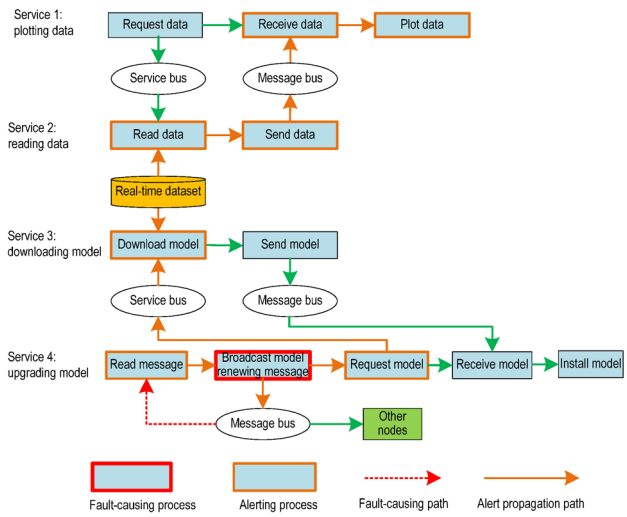


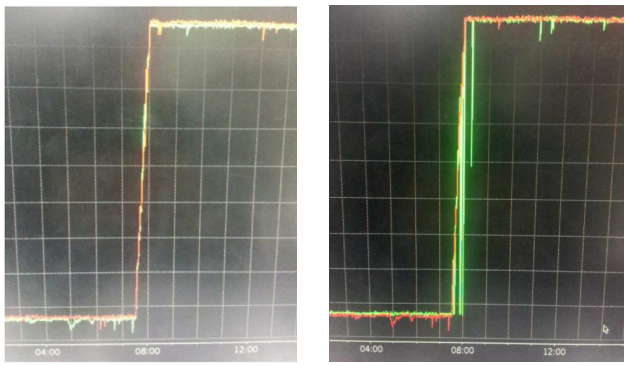
Figure 5. Fault causing logic of interruption of a telemetering data refresh

Step 6: Debugging and/or renewing the fault localization knowledge. When new fault-alert logic is found, new knowledge is acquired and then can be appended to the expert system and the FLC.

In the studied case, from the results of both fault-causing process localization and its alerting logic reproduction, we trace the syntax error of service 4, that is, the model updating message should not be sent back to itself. After the patching and updating, the monitored key parameter is periodically refreshed during the period of model updating.

4 Case Study

In this section, we employ the proposed method to localize the fault-causing process of a complicated case. In this case, the monitored key parameter in SCADA application is the generated power (GP) of the grid. Usually, the GP value changes steadily without sharp fluctuation, as shown in Figure 6(a). However, a fault is detected at 7:40, 7:53, 7:55, and 8:20, when the measured value fluctuated sharply as illustrated by the green curve in Figure 6(b).



(a) Normal trend (b) Abnormal trend

Figure 6. trend of the generated power of grid

Before the localization of the fault-causing process, we first present the services related to the GP monitoring. As illustrated in Figure 7, three servers are involved in the GP monitoring, i.e., the IDS managing server (in white), the host SCADA application server (in blue), and the SCADA application backup server (in green). The default host SCADA application server is preset, but its state can be changed to backup when some conditions are met, and the server's state determines its function. For example, only the server in the host state can write GP data to the history database in order to avoid data overwriting. There are eight services running at the two SCADA application servers and one human-machine interface (HMI) visualization service in the IDS system managing server, which are listed below.

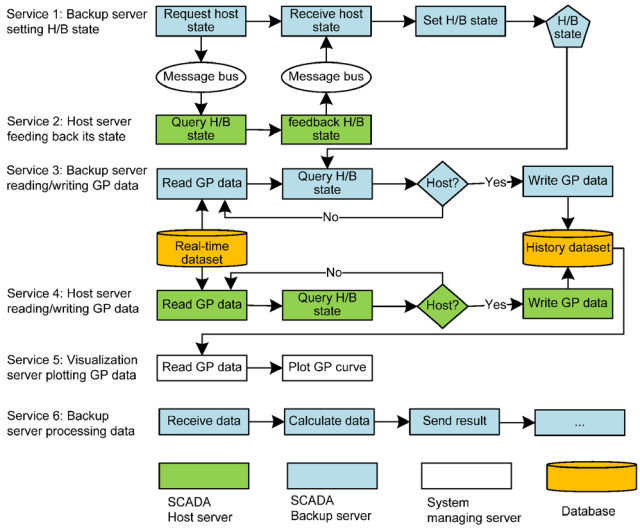
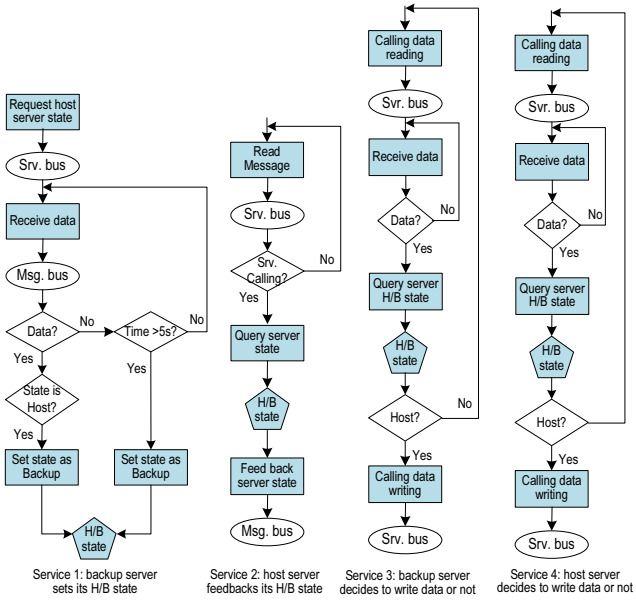


Figure 7. Flow chart of GP monitoring-related services

- Service 1: the backup server determines its host or backup (H/B) state. The backup server requests the host server's state information periodically via service (Srv.) bus and sets its state as backup when host state is fed back by the host server. When it does not receive the feedback from host server in 5 seconds or the feedback state is backup, the backup server changes its state into host.
- Service 2: the host server feeds back its H/B state. At receiving the querying message from the backup server via service bus, the host server queries its own state and feeds back its working state to the backup server via a message bus.
- Service 3: the backup server reads GP data and writes the GP data into history database if necessary. The backup server periodically calls the common service of data reading from the RT database, and writes the GP data to the history database only if its state is host.
- Service 4: the host server reads GP data and writes this GP data into the history database if necessary. Similar to the Service 3, the host server periodically

reads GP data from the RT database and writes them to the history database when its state is host.

- Service 5: the system managing server plots the GP data for data visualization. The system managing server periodically calls the common service of data reading to get GP data from the history database and then plots them at the monitor.
- Service 6: the backup server computes data. This service receives data from message bus, calculates an intermediate, and then sends it out via message bus.
- Common service of data reading from RT database. This service reads data as requested from the RT database and then feeds back the data via message bus.
- Common service of data writing to history database. This service receives data from message bus and then writes them into the history database as requested.



The alert messages are categorized according to the constraint types and listed below.

Hardware resource alerts. At the backup server, *calculate data* process of service 6 alarms overuse of the memory, while all other processes alarm the shortage of memory, and the system information monitoring service reports that SWAP memory has been used up.

Functional logic alerts. The system managing process of the backup server alarms the processing delay; all processes at the backup server alarm processing delay.

Software resource alerts. At the backup server, the system managing process alarms the H/B state frequent switching.

It is notable that all alerts are related to the backup server and visualization server, while no alarm message is reported from the host server. Then it is believed that the host server works properly.

Based on the knowledge of GP monitoring logic and the alerts, we can localize the fault-causing process by the proposed four-step method.

Step 1: constraint mapping table construction. Firstly, FLC of the eight services in the form of fault tree models are constructed for intra-service source abnormal process tracing by fault tree method, which is shown in Figure 8.

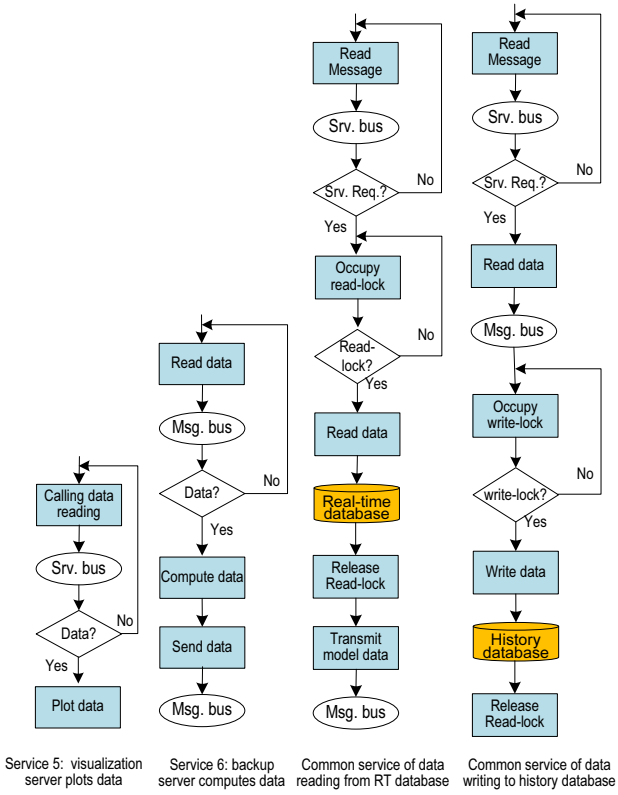


Figure 8. Fault tree of GP monitoring-related services

Then HRC, SRC, and DOC mapping tables are constructed, which are summarized in Table 4 to Table 7. For brevity, the number of occupied resource is omitted and only the state is listed.

Table 4. Memory-type HRC mapping table at the backup server

Proc.	Belonged Srv.	State
Request host state	1	insufficient
Receive host state	1	insufficient
Read GP data	3	insufficient
Calculate data	6	excessive
…	…	…

Table 5. SRC mapping table at the backup server

Type	Caller	Callee	State
Msg. bus	Req. host state	Query H/B state	delayed
Msg. bus	Feedback H/B state	Receive host state	delayed
…	…	…	…

Table 6. DOC mapping table of the RT database

Proc.	Srv.	Operation
Read GP data	3	Reading
Read GP data	4	Reading

Table 7. DOC mapping table of the history database

Proc.	Srv.	Operation
Write GP data	3	Writing
Write GP data	4	Writing

Step 2: source abnormal process tracing within service. Since no alert is reported by the host server, the source abnormal process tracing is only implemented at the backup server. Based on the alerts and fault-tree models, the source abnormal process of service 1 is the *Request host server state* process due to being the starting process and no abnormality from the host server to introduce alert at the backup server thereafter. For service 3, the source abnormal process *read GP data* is traced due to the same logic as that in service 1. Obviously the source abnormal process is regarded as *Calculate data* in service 6 regarding the alert from memory-type HRC mapping table.

Step 3: fault-causing process localization among services via inter-service constraints. Besides the FLCs, HRCs connect different services at the backup server. Specifically, among the source abnormal processes set, the memory-type HRC alerts have been reported by all these three processes. It is straight to infer that *Calculate data* process is the fault-cause process which occupies too much memory with result of memory shortage for other two processes. This deduction is further confirmed by the earliest alarming time index of the *Calculate data* process.

Step 4: alarming logic reproduction. From the fault-causing process and the source abnormal processes of services with alerts, we can reproduce the story of alarming, which is illustrated in Figure 9. First, *Calculate data* process of service 6 occupies too much memory, then all other active processes at the backup server are short of memory, which causes the following results: *i*) both *request host state* and *receive host state* processes of service 1 are delayed, then the backup server changes its state into host because it cannot receive the state message of the host server in the request period and then believes in the host server being off-line; *ii*) *read GP data* process of service 3 is delayed and then reads the overdue data. When the state of the backup server is set as host, it will write the overdue GP data to the history database and then overlap the up-to-the-date data written by the host server which works regularly; *iii*) *read GP data* process of service 5 will read the overdue GP data and the plotted GP curve shows a sharp fluctuation; *iv*) when SWAP memory is used up, the out of memory (OOM) killer mechanic of Linux will kill some active processes to release memory.

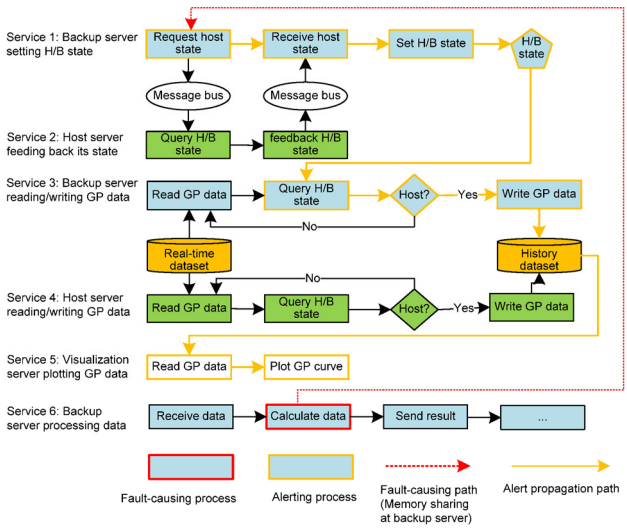


Figure 9. Fault causing logic of sharp fluctuaton of GP data

From the fault localization result, the missing of memory release at the *Calculate data* process of service 6 is detected. Moreover, from the alarming logic reproduction, two new constraints can be appended to the FLC, i.e., the GP field of the history database cannot be written twice within a data writing period, and only one server with host state is permitted to appear at the system. After the patching and updating, the sharp fluctuation fault disappears, which verifies the correctness of the fault localization by the proposed method.

5 Conclusion

In this paper, a fault localization method is proposed for the IDS of smart grid. To decouple the complicated coupling among the services, four types of constraints are synthesized and constraint mapping tables are constructed accordingly. Based on the constructed constraint mapping tables and alert messages, the obscure fault localization is simplified greatly into three steps. In step 1, the source abnormal process is traced within every service with alerts without concerning other services. In step 2, fault-causing process is localized among the abnormal processes by only concerning the four types of constraints. In the last step, the alarming logic is reproduced to verify the fault localization result. Two practical case studies are presented and the results show the effect of the proposed method.

References

[1] X. Fang, S. Misra, G. Xue, D. Yang, Smart Grid - The New and Improved Power Grid: A Survey, *IEEE Communication Survey & Tutorials*, Vol. 14, No. 4, pp. 944-980, April, 2012.

[2] P. Zhang, F. Li, N. Bhatt. Next-generation Monitoring, Analysis, and Control for the Future Smart Control Center,

IEEE Transactions on Smart Grid, Vol. 1, No. 2, pp. 186-192, 2010.

[3] A. Aburabou, A Wireless Communication Architecture for Smart Grid Distribution Networks, *IEEE System Journal*, Vol. 10, No.1, pp. 251-261, January, 2016.

[4] Z. Guan, J. Li, L. Wu, Y. Zhang, J. Wu, X. Du, Achieving Efficient Ad Secure Data Acquisition for Cloud-Supported Internet of Things in Smart Grid, *IEEE Internet of Thing Journal*, Vol. 4, No. 6, pp. 1934-1944, June, 2017.

[5] F. Aalamifar, L. Lampe, Cost-efficient QoS-aware Data Acquisition Point Placement for Advanced Metering Infrastructure, *IEEE Transactions on Communications*, Vol. 66, No. 12, pp. 6260-6274, December, 2018.

[6] Z. Wang, An Identity-based Data Aggregation Protocol for the Smart Grid, *IEEE Transactions on Industrial Informatics*, Vol. 13, No. 5, pp. 2428-2435, 2017.

[7] A. Singh, R. Singh, B. Pal, Stability Analysis of Networked Control in Smart Grids, *IEEE Transactions on Smart Grid*, Vol. 6, No. 1, pp. 381-390, 2015.

[8] K. Sun, Q. Chen, Z. Gao, An Automatic Faulted Line Section Location Method for Electric Power Distribution Systems Based on Multisource Information, *IEEE Transactions on Power Delivery*, Vol. 31, No.4 pp. 1542-1551, August, 2016.

[9] U. Choi, J. Lee, F. Blaabjerg, K. Lee, Open-circuit Fault Diagnosis and Fault-tolerant Control for a Grid-connected NPC Inverter, *IEEE Transactions on Power Electronics*, Vol. 31, No. 10, pp. 7234-7247, October, 2016.

[10] M. Rana, W. Xiang, G. Wang, IoT-based State Estimation for Microgrids, *IEEE Internet of Things Journal,* Vol. 5, No. 2, pp. 1345-1346, April, 2018.

[11] M. M. Rana, L. Li, S. W. Su, W. Xiang, Microgrid State Estimation: A Distributed Approach, *IEEE Transactions on Industrial Informatics*, Vol. 14, No. 8, pp. 3368-3375, August, 2018.

[12] G. Han, X. Miao, H. Wang, M. Guizani, W. Zhang, CPSLP: A Cloud-based Scheme for Protecting Source-location Privacy in Wireless Sensor Networks Using Multi-sinks, *IEEE Transaction on Vehicular Technology*, Vol. 68, No. 3, pp: 2739-2750, Mary, 2019.

[13] Z. Li, Q. Li, Z. Wu, J. Yu, R. Zheng, A Fault Diagnosis Method for on Load Tap Changer of Aerospace Power Grid Based on the Current Detection, *IEEE Access*, Vol. 6, No. 4, pp. 24148-24156, April, 2018.

[14] C. Nan, F. Khan, M. Iqbal, Real-time Fault Diagnosis Using Knowledge-based Expert System, *Process Safety Environ. Protection*, Vol. 86, No. 1, pp. 55-71, January, 2008.

[15] F. Zidani, D. Diallo, M. Benbouzid, R. Nait-Said, A Fuzzy-based Approach for the Diagnosis of Fault Modes in a Voltage-fed PWM Inverter Induction Motor Drive, *IEEE Transactions on Industrial Informatics*, Vol. 55, No. 2, pp. 586-593, February, 2008.

[16] O. Linda, D. Wijayasekara, M. Manic, C. Rieger, FN-DFE: Fuzzy Neural Data Fusion Engine for Enhanced Resilient State-awareness of Hybrid Energy Systems, *IEEE Transactions on Cybernetics*, Vol. 44, No. 11, pp. 2065-2075, November, 2014.

[17] W. Lewis, *Software Testing and Continuous Quality Improvement*, 3rd ed., CRC Press, 2016.

[18] P. Ammann, J. Offutt, *Introduction to Software Testing*, Cambridge University Press, 2016.

[19] Y. Shatnawi, M. Al-Khassaweneh, Fault Diagnosis in Internal Combustion Engines Using Extension Neural Network, *IEEE Transactions on Industrial Electronics*, Vol. 61, No. 3, pp. 1434-1443, Mary, 2014.

[20] S. Toma, L. Capocchi, G. Capolino, Wound-rotor Induction Generator Inter-turn Short-circuits Diagnosis Using a New Digital Neural Network, *IEEE Transactions on Industrial Electronics*, Vol. 60, No. 9, pp. 4043-4052, September, 2013.

[21] M. Valtierra-Rodriguez, R. Romero-Troncoso, R. Osornio-Rios, A. Garcia-Perez, Detection and Classification of Single and Combined Power Quality Disturbances Using Neural Networks, *IEEE Transactions on Industrial Electronics*, Vol. 61, No. 5, pp. 2473-2482, May, 2014.

[22] G. Han, H. Wang, M. Guizani, S. Chan, W. Zhang, KCLP: A k-means Cluster-based Location Privacy Protection Scheme in WSNs for IoT, *IEEE Wireless Communication Magazine*, Vol. 25, No. 6, pp. 84-90, June, 2018.

Biographies

Xinpeng Li received the B.S. and M.S. degree from Shanghai Jiaotong University in 2012 and Hongkong University of Science and Technology in 2013, respectively, both in Electronics Engineering. He is currently pursuing the Ph.D. degree with the Beijing University of Posts and Telecommunications (BUPT), Beijing, China. Currently he is also with the State Grid Jibei Electric Power Company Limited as a senior engineer.

Yuexing Peng received the Ph.D degree from Southeast University, Nanjing, China, in 2004. From 2006 to 2008, he was a Postdoctoral Research Associate with BUPT. Since May 2008, he has been a faculty member with BUPT, where he is currently an Associate Professor. His research interests include machine learning, intelligent signal processing and Internet of Things.

Guangjie Han received the Ph.D. degree from Northeastern University, Shenyang, China, in 2004. From October 2010 to 2011, he was a Visiting Research Scholar with Osaka University, Suita, Japan. He is currently a Professor with the Department of Information and Communication System, Hohai University, Nanjing, China.

Hanxu Sun is a professor of BUPT, China. He was granted a special government subsidy of the State Council, and was awarded the Prize for the first successful manned spaceflight and the May 1 labour medal of Beijing Municipality. His research areas include space robotics and automatic control theory.

Bo Yan received the B.S. and Ph.D degree from Zhejiang University, China, in 2007 and 2012, respectively, both in Electrical Engineering. He was a visiting scholar with Missrouri University of Science and Technology from 2011 to 2012. Currently he is with the State Grid Jibei Electric Power Company Limited as a senior engineer.