

EdgeKeeper: a trusted edge computing framework for ubiquitous power Internet of Things^{*}

Weiyong YANG^{1,2}, Wei LIU², Xingshen WEI², Zixin GUO²,
Kangle YANG^{†‡2}, Hao HUANG¹, Longyun QI²

¹Department of Computer Science and Technology, Nanjing University, Nanjing 210023, China

²NARI Group Co., Ltd., Nanjing 210003, China

[†]E-mail: yangkangle@sgepri.sgcc.com.cn

Received Nov. 20, 2019; Revision accepted Mar. 16, 2020; Crosschecked May 29, 2020; Published online Jan. 8, 2021

Abstract: Ubiquitous power Internet of Things (IoT) is a smart service system oriented to all aspects of the power system, and has the characteristics of universal interconnection, human-computer interaction, comprehensive state perception, efficient information processing, and other convenient and flexible applications. It has become a hot topic in the field of IoT. We summarize some existing research work on the IoT and edge computing framework. Because it is difficult to meet the requirements of ubiquitous power IoT for edge computing in terms of real time, security, reliability, and business function adaptation using the general edge computing framework software, we propose a trusted edge computing framework, named “EdgeKeeper,” adapting to the ubiquitous power IoT. Several key technologies such as security and trust, quality of service guarantee, application management, and cloud-edge collaboration are desired to meet the needs of the edge computing framework. Experiments comprehensively evaluate EdgeKeeper from the aspects of function, performance, and security. Comparison results show that EdgeKeeper is the most suitable edge computing framework for the electricity IoT. Finally, future directions for research are proposed.

Key words: Internet of Things; Ubiquitous power Internet of Things; Edge computing; Trusted computing; Network security
<https://doi.org/10.1631/FITEE.1900636>

CLC number: TP391


1 Introduction

To accelerate the strategic deployment of a world-class energy Internet company with global competitiveness, the State Grid Corporation of China promoted a comprehensive plan for the “three-type (hub-, platform-, and shared-type) and two-network (strong smart grid and ubiquitous power Internet of Things (IoT))” construction in 2019 and built a “three-type” enterprise, which is an important starting point for building a world-class energy Internet en-

terprise. Construction and operation of the “two-network” constitutes an important material basis for building a world-class energy Internet company. The construction of “three types and two networks” is the company’s specific practice of the network power strategy, an important measure to implement the central government’s deployment and give play to the leading role of central enterprises, and is an inevitable requirement to adapt to internal and external situations and challenges. The ubiquitous power IoT, which fully applies modern information technology (IT) and advanced communication technologies such as mobile Internet and artificial intelligence (AI) to all aspects of the power system, is a smart service system that uses convenient and flexible features. It achieves interconnection and human-computer interaction of all links in the power system, with comprehensive state perception and efficient information processing.

[‡] Corresponding author

^{*} Project supported by the State Grid Corporation Science and Technology Project, China

 ORCID: Weiyong YANG, <https://orcid.org/0000-0001-8430-9168>; Kangle YANG, <https://orcid.org/0000-0001-7646-4336>

© Zhejiang University Press 2021

The creation of the ubiquitous power IoT has opened up a new path for safer grid operation, better learner management, more accurate investment, and better service. At the same time, it can make full use of the unique advantages of the power grid and open up the huge blue ocean market of the digital economy. Building ubiquitous power IoT is the core task in implementing the strategic objectives of a “three-type, two-network, and world-class” system.

With the gradual advancement of the ubiquitous power IoT, edge computing framework has gradually become a research hotspot. The design of an edge computing framework is diverse and generally includes the following functions: resource management based on the edge operating system (OS), access to subdevices, data collection, device control, security management, application management, and IoT platform interaction. According to the design goals and application scenarios, it can be divided into three categories: edge computing for IoT, edge computing for edge cloud services, and edge computing for cloud edge fusion (Liang et al., 2019). The ubiquitous power IoT has both the edge computing for IoT and cloud-edge convergence application scenarios.

In general, in edge computing for IoT, the edge computing framework plays mainly the following roles (Edge Computing Consortium, 2018): (1) Application (APP) controlling. After the IoT management platform issues control commands, the edge agent (edge frame) is made to receive them and acts as an agent to control the APP in the cloud. (2) Data sharing. The data collected by the service APP caches data on the edge association agent and provides a mechanism for data sharing between different APPs. (3) Edge computation. The edge side performs edge computation based on real-time data, cache data, and models issued by the IoT management platform. (4) Cloud-side collaboration. Cloud-side collaboration system covers resource collaboration, data collaboration, intelligent collaboration, application management collaboration, business management collaboration, and service collaboration. (5) APP development. To simplify the development of APP, it is necessary to refine the general interfaces, such as data cache, secure access, data collection, and APP management, to form a unified software development kit (SDK).

Ubiquitous edge computing of power IoT is just in its infancy, and there are currently many problems:

(1) The existing sensing capabilities are not fully used, and the sharing among professionals and the resource reuse are insufficient. A large number of existing sensing devices are not fully functioning; each professional system is self-contained, and the sensing devices are repeatedly deployed, making the system difficult to achieve one-time acquisition and sharing, thus resulting in insufficient data penetration and insufficient data mining to improve the safe operation level, efficiency of power grid, work quality, and so on. (2) The local scene perception depth is insufficient; the user’s energy information is not timely, the distribution information coverage is incomplete, and the emerging business perception is not fully shared. (3) There lack business support capabilities and perceived sources on the Internet side; the network deployment structure cannot support the development of new formats, such as integrated energy services and data operations. It needs standardization of the perception system and rectification of the lack of intelligence. Application requirement cannot be dynamically changed, and a large amount of on-site operations and maintenance are required. Without standardization of construction, it is difficult to centralize, control, and achieve the goal of one source of data. Therefore, it is necessary to construct a unified edge computing framework to help construct the ubiquitous power IoT and support the strategic objective of a “three-type, two-network, and world-class” framework.

Due to the uniqueness of the ubiquitous electric power IoT, traditional edge computing supported software cannot meet the needs of the ubiquitous electric power IoT in terms of edge computing. First, the power business, especially the business related to grid control, has strong real-time requirements. The application-based software cannot meet the real-time requirements of power business alone, especially the maximum response time of system interruption. The terminal response time is usually around 200 μ s, and it is difficult for the general Linux-based application software to meet the requirements of power in real time. Second, the ubiquitous gateways of the electric power IoT are widely distributed in the wild, buildings, plants, and other places. Gateway devices are faced with great network security threats and business reliability challenges at the levels of software, OS, and even hardware. How to ensure the security and

reliability of devices and services based on the edge computing framework has become a major challenge. Third, in the multiservice convergence scenario of the IoT, cloud and edge collaboration is provided to the power business as a whole. Achieving multiservice and multidimensional cloud-side collaboration to meet complex business requirements has become an important issue for the edge computing framework.

In this study, we design and implement a set of trusted edge computing frameworks that meet the edge computing functional requirements of the ubiquitous power IoT, and solve many technical problems. The edge computing framework adheres to many principles: (1) It adheres to the principle of “side-end separation,” and develops and deploys a unified edge IoT agent which separates the functions of the edge IoT agent and the sensing collection terminal. The “end” focuses on perception and collection with a huge scale and simple functions, and the “side” focuses on data convergence, intelligent expansion, resource sharing, one-site one-side, and unified collection of data sources, to achieve front-end business integration, thus forming edge computing and regional autonomy. (2) It adheres to the principle of “shared by side management,” builds a unified management platform for IoT, and achieves unified access, unified operation, and unified control of the edge agent terminals. (3) Under the premise of “reliability, controllability, and customization,” the unified IoT perception system should establish unified technical specification and fully absorb the advanced and mature technologies of the Internet. (4) Considering both the security and ease of use, the existing security protection measures are extended in the information intranet, and the general security protection scheme adapted to the Internet architecture is adopted to the Internet side to achieve safe and convenient access to user terminals. The overall security protection strategy for the IoT perception system requires the development of the IoT management platform and the edge agent association standard. It also needs to embed the relevant security protection design in the design and development stage. Finally, through the IoT management platform and edge material agent device, the problems of repeated acquisition of existing power terminals, multiple protocols, scattered data storage are changed. Through data acquisition at one time and multiple applications, unified standards,

unified management and control, and unified operations are achieved.

2 Related works

Since 2015, edge computing has entered into Gartner’s hype cycle (technology maturity curve). It has set off a wave of industrialization. Various industrial and commercial organizations are actively initiating and promoting research, standards, and industrialization activities for edge computing.

In academic research, IEEE/ACM Symposium on Edge Computing was formally established in October 2016, which formed an academic forum jointly recognized by academia, industry, and government. The application to edge computing and forum’s research directions have been discussed (Shi and Dustdar, 2016; Shi et al., 2016; Satyanarayanan, 2017). In the past two years, special attention has been paid to the performance in IoT scenarios (Maheshwari et al., 2018), security (Ahmed et al., 2018), application scenarios (Chao et al., 2018), cloud-edge collaboration (Ai et al., 2018), and integration with AI and other technologies (Aral and Brandic, 2018; Feng et al., 2018; Jang et al., 2018). In May 2018, the 3rd ACM/IEEE Symposium on Edge Computing was held in China. Many universities and research institutes discussed edge computing interactively to sort out the needs of developers. In addition, many domestic scholars have carried out extensive research on data models (Li JR et al., 2018), computational models (Shi et al., 2017), industrial applications (Wang et al., 2013; Li SN and Luo et al., 2014; Zhang et al., 2018; Zuo et al., 2019), and network security (Yang YM and Song, 2015; Sha et al., 2018) in edge computing scenarios.

In terms of standardization, the International Electrotechnical Commission (IEC) released the Vertical Edge Intelligence (VEI) White Paper (Jang et al., 2018) in 2017, which introduces the importance of edge computing for vertical industries such as manufacturing. The International Organization for Standardization (ISO)/IEC established the Edge Computing Research Group. Edge computing has become an important connotation of the framework in the IEEE P2413 standard for the architectural framework for IoT. The China Communications

Standards Association (CCSA) established the Industrial Internet Ad Hoc Group (ST8).

In terms of industry alliances, in November 2016, Huawei, China Electric Power Research Institute, China Information and Communication Research Institute, Intel, Advanced RISC Machine (ARM), and iSoftStone Information Technology Co., Ltd. jointly launched the Edge Computing Industry Alliance. In 2017, under the Global Industrial Organization Industrial Internet Consortium (IIC), Edge Computing TG was established and a partial edge computing reference framework was defined. In 2019, to accelerate the strategic deployment of a world-class energy Internet company with global competitiveness, the State Grid Corporation of China made a comprehensive promotion of the “three-type and two-network” construction. Many researchers in the power industry began relevant applied research and practice (Cai et al., 2019; Chen et al., 2019; Xu, 2019; Liu et al., 2019).

In terms of the specific edge computing framework, edge computing for IoT, edge computing for edge-cloud services, and edge computing for cloud edge fusion are the mainstream edge computing frameworks.

Edge computing for IoT dedicates to solving problems in the process of developing and deploying IoT applications, such as multiaccess methods. For example, EdgeX Foundry (Saxena and Salem, 2015), a standardized interoperability framework developed for industrial IoT edge computing, provides an extremely simplified and standardized edge computing architecture for industrial IoT around the ecosystem of interoperability components. Apache Edgent (<https://www.oschina.net/p/apache-edgent>) is a programming model and a runtime edge framework with the microkernel style. It focuses on efficiently analyzing data from the edge, which can accelerate the development of edge computing applications in data analysis. Apache Edgent with rich application program interfaces (APIs) can be deployed in the edge computing of running Java virtual machines for real-time analysis of data from devices and for actual accelerated development needs of networks. Predix (Zhou, 2018) is oriented to manufacturing industry. It provides a development framework, supports the access of open field protocols, enhances the function of edge computing, and develops the corresponding

functions of device access and edge computing by partners.

Edge computing for edge-cloud services focuses mainly on optimizing or rebuilding the infrastructure of network edge, to build data centers on the edge of the network and provide similar cloud center services, which are usually found on the edge of network operators such as cellular network base stations. Central Office Re-architected as a Datacenter (CORD), a representative of the Open Networking Foundation (ONF), reconstructs the edges of networks using software-defined networks and Network Function Virtualization (NFV) cloud computing technology. CORD provides edge-cloud services on the edge of operators. For users, it does not need to provide computing resources or build a platform, thus reducing the cost of hardware and software. In addition, the Linux foundation provides an open-source project named “Akraio Edge Stack” for high-performance edge cloud, dedicating to developing a set of open-source software stacks to optimize network construction and management of the edge infrastructure.

For edge computing based on cloud edge convergence, cloud computing service providers are important promoters of edge computing. Based on the concept of “cloud edge convergence,” they are committed to extending cloud service capabilities to the edge of network. Typical examples include AWS Green Grass, Baidu OpenEdge (Shen and Yang, 2015), Ali Link IoT Edge, and Azure IoT Edge, aiming at mixing cloud and edge computing frameworks, as well as expanding cloud capabilities to edge devices to achieve low latency. Edge frameworks on the edge device often use the same programming model on the cloud.

Different frameworks have different understandings, scheme designs, and implementation ideas for edge computing, and are not compatible with each other. The edge framework of the ubiquitous power IoT is dominated by the edge computing scenarios for IoT, and at the same time, there are certain cloud-edge fusion computing scenarios. For the computing frameworks, OpenEdge has limited functions and is closely tied to the Baidu IoT platform, but it can be used for functional calculation. KubeEdge adapting edge computing based on the Kubernetes technology has limitations on platform technology, and is tightly coupled to the platform. EdgeX modules are

decoupled, APP runs in the form of microservices, and APP management is implemented by REST API calls. It is a relatively complete solution to the industrial IoT, but lacks cloud-edge convergence and security considerations. EdgeX provides the interface only for data export and cannot communicate directly with the IoT management platform. EdgeX lacks the functions of application issuance, upgrade, management, business APP control, equipment management control, and monitoring. At the same time, EdgeX lacks the security reinforcement scheme, design in security access, access control, and application command verification. It is necessary to develop an interaction process with an IoT management platform based on the interaction specification.

3 EdgeKeeper

The ubiquitous power IoT is not only a network infrastructure but also an application of IoT technology. It is a comprehensive application of new information and communication technology (ICT). Through mutual penetration and intelligent interaction between information-physics fusion and the new generation power systems, the company can achieve energy and electricity production and consumption. The real-time online connection and integration of people, machines, and objects in each link has gradually formed an infrastructure to support the operation of China's Energy Internet. The ubiquitous power IoT provides horizontal support for the entire business and shields differences in the underlying network

through the IoT agent and the IoT management platform to achieve the first horizontalization for connections of things. Through the capability open center, the user, business, and terminal can be integrated to realize collaboration and achieve the secondary level of operation on the full-service cloud. The platform supports cloud-fog integrated processing, power service data streaming, device data streaming, and secure data streaming based on storage, management, and analysis in the full-service data center. The concept of "collected once, used everywhere" expands the ability of supporting the entire business. The overall functional architecture of the ubiquitous power IoT is shown in Fig. 1.

The ubiquitous power IoT platform is composed mainly of the IoT management platform and the capacity open center. It manages the IoT agent, the terminal, and network resources, supports the business upwards, and provides the API for businesses to become open for outside operations. The IoT management platform supports ubiquitous links, achieves establishment, maintenance, and configuration of network topological links, and enables virtualized orchestration/management of network resources, status monitoring, centralized configuration, remote upgrade of devices such as terminals, and functions of identity authentication and authority management of users inside and outside the network. The capability open center provides the development environment and API for business applications, which support third-party capability integration, business application, rapid development publication, and message push. Edge computing is used mainly in the agent of

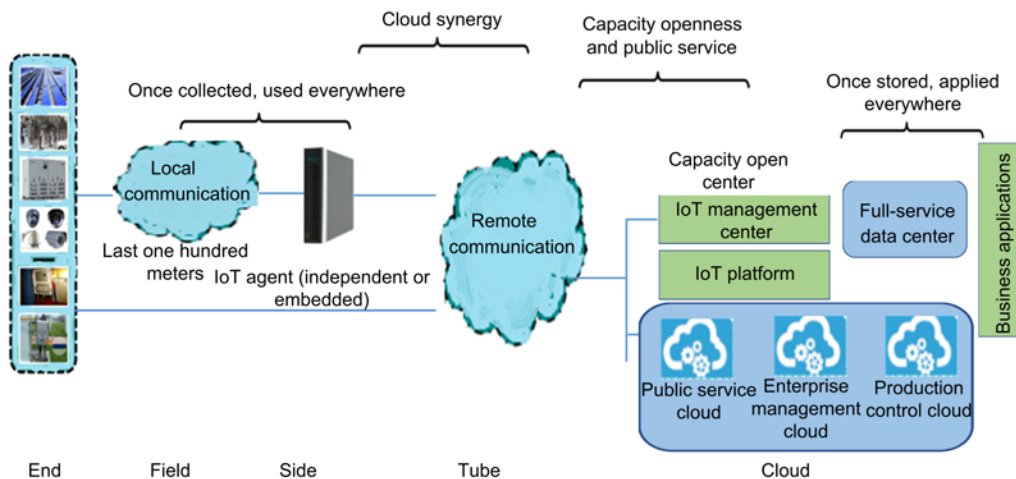


Fig. 1 Ubiquitous power Internet of Things (IoT) functional architecture

object, supporting the ubiquitous power IoT in APP control, data sharing, computing, cloud-edge collaboration, APP development, and other functions.

3.1 Overall framework

The edge computing framework, EdgeKeeper, is divided into the hardware layer, OS layer, basic functional layer, and edge service layer in the functional architecture. The hardware layer includes the unique identifier of the device, trusted computing module, trusted execution environment, and security cryptographic module. It provides a secure and confidential space for the privacy data and for sensitive computing in the execution environment, in addition to implementing the chip in the system. The step-by-step trusted verification of the startup, i.e., the OS layer, includes functions such as system monitoring, secure access, application isolation, and trusted metrics to ensure that only the programs that pass the authentication can run in the system. The hardware layer and OS layer ensure the safety of the framework. The basic functional layer includes subdevice access, object model management, message queue, and other functions. Microservices in the basic functional layer communicate with devices, sensors, actuators, and other IoT objects through the protocols inherent in each IoT object. The generated and transmitted data is converted into a common data structure, and the

matched data of the object model is sent to the upper service. The edge service layer includes functions, such as flow calculation, rule engine, and various microservices, which provide edge analysis and data processing, and supports cloud-side collaboration of resources, data, intelligence, application management, and so on. The functional framework of EdgeKeeper is shown in Fig. 2.

We fully learn from the advantages of EdgeX, OpenEdge, and KubeEdge. EdgeX Foundry locates the industrial IoT and solves the interoperability problems of IoT devices and various business protocol issues. The components implement data to cache and upload, the command controls forwarding and execution, and the rule engine implements complex business processes. OpenEdge locates cloud-edge fusion, function computing can achieve lightweight edge computing, and decoupling between APPs is based on message queuing telemetry transport (MQTT). The agent interacts with cloud, and the engine acts as the background to manage the business APP. Both OpenEdge and EdgeX are based on containers for a business APP, and have complementary business functions. KubeEdge is friendly to cloud-edge integration based on Kubernetes (k8s). In the end, we integrate the design concepts of EdgeX, KubeEdge, and OpenEdge to design the edge computing framework of EdgeKeeper (Fig. 3).

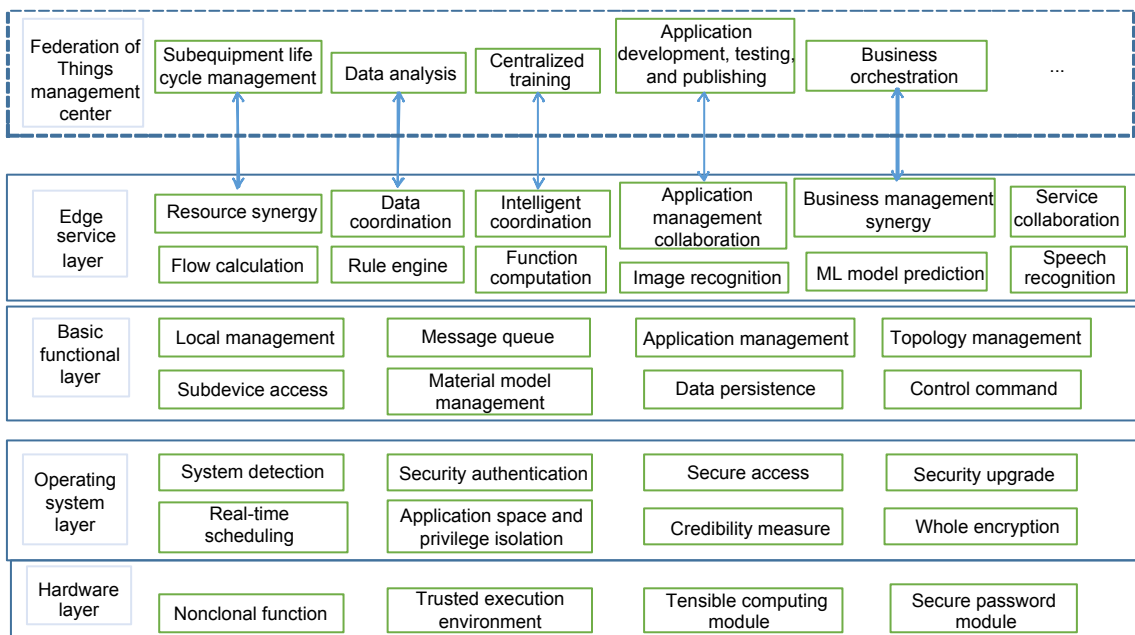


Fig. 2 Ubiquitous IoT edge computing functional framework

3.2 Object model design

The business application senses environmental changes from the terminal device through the underlying driver, generates data, and reports data to the IoT management platform. A large number of nonintelligent sensors send data periodically at different frequencies in the industry, which is forwarded to the International Telecommunication Union (ITU) management platform by the edge computing agent device. Edge agents need to identify semantic information and perform preprocessing, such as data denoising and deduplication, emergency analysis, data format conversion, and retransmission encryption. After receiving the data, the IoT management platform needs to verify the data based on data standards such as integrity and data format, and data protocols generated by different devices are completely different, especially in electric power plausibility check. The above work usually faces great challenges in practical applications. Due to various types of devices, data formats, and transmission industries, industrial equipment, complex industrial control protocols, and diverse business requirements have caused the IoT agent to face difficulties in the preprocessing and edge computing stages. The IoT management platform lacks the basis to check the data after receiving it.

To solve the above problems, the concept of object model is introduced in the design of ubiquitous power IoT, aimed at unifying the terminal equipment model and the data model of the terminal equipment. Through the standardized descriptions of equipment capabilities, functions, attributes, and status, the data uploaded by the agent and intelligent terminal equipment is checked in the management platform of the IoT, and the IoT agent performs information collection and service control on sensors in the grid business application.

The object model must be based on a self-describing grammar format (Boutaud and Ehlig, 1991), which contains all the information needed to describe the device. A complete description of device capabilities is achieved by mapping the device to three-dimensional attributes, interfaces, and events. The attribute contains the static and extended attributes of the IoT terminal entity. A static attribute is a natural attribute value that does not change throughout the life cycle of an IoT terminal. The dynamic attribute refers to the service data that the terminal entity actively reports periodically and that needs to be dynamically added according to the service requirements during the whole life cycle management process. Events refer to the business messages and

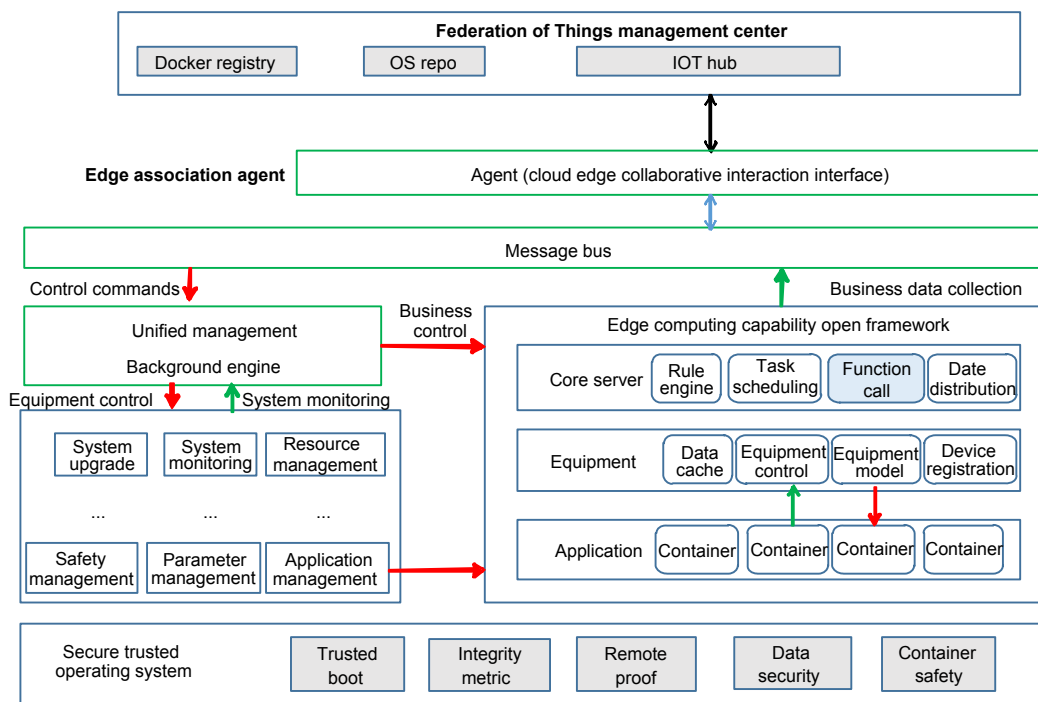


Fig. 3 EdgeKeeper: component interaction

security events reported by the terminal entity on its initiative. Interface means that the IoT terminal accepts the control command and makes a corresponding description. The business unit defines the interface that needs to conform to a certain format, as shown in Table 1.

3.3 Edge calculation

At present, there is still a lack of an accurate and unified definition of edge computing. Industry and academia have described edge computing from their own points of view (Luan et al., 2015; Hu et al., 2015; Mach and Becvar, 2017; Mao et al., 2017). In general, edge computing is a new computing model. Compared with cloud computing, computing and storage resources are deployed on the terminal device side to obtain higher computing real-time performance and improve service responsiveness. In addition, non-critical data processed on the edge side no longer needs to be uploaded to the data center, which greatly reduces network overhead and resource pressure on cloud computing.

EdgeKeeper provides edge computing capabilities based on techniques such as the rule engine (<https://www.progress.com/openedge>) and function calculation (Fultz et al., 2010). The business program running on EdgeKeeper acquires the data actively from a terminal device. The process of edge-free computing involves forwarding edge agents to the IoT management platform. To reduce useless data transmission, the edge calculation model pushes some computing tasks down to the edge side for execution. In a typical edge computing scenario supported by EdgeKeeper, the APP fetches data from the device. EdgeKeeper's internal components publish the data as events, and any internal components and programs

that subscribe to the event will obtain a copy of the event. In fact, EdgeKeeper implements the edge event triggering mechanism through the rule engine to reduce the response time. As the key data service, EdgeKeeper's rule engine subscribes to all internal APP events and receives all the data collected by the APP. The response is triggered by reading and loading the rule file of the IoT management platform. The rule file describes the events of interest, the triggered actions, and the condition of the triggered actions.

EdgeKeeper implements two types of edge computing methods. The first method is based on the event-triggered device to control operations. As shown in Fig. 4, when the built-in rules in the rule engine are compared with the data collected by the marketing APP, the control behavior of the APP on a certain device will be triggered according to the rules. Another kind of edge computing model is more flexible. The data collected by the APP will trigger the calculation of a function if it meets the requirements after comparison with the data collected by the rule engine. The instance of function calculation is managed uniformly by the local function calculation background engine. The rule engine needs only to specify the name of the function that needs to be executed, and this invokes the background engine interface to start the function calculation instance and process the collected data. This process includes data denoising, data format conversion, data encryption before transmission, and complex business logic processing such as line loss calculation. The advantages of function calculation are light weight and flexibility. Usually, the data processing tasks are relatively simple, and do not require complex operations. The instance loading and starting steps are fast, and the edge system resources are less occupied.

Table 1 Interface definition of the object model

Parameter	Description	Mandatory
Interface name	Supporting Chinese, uppercase and lowercase letters, numbers, dashes, and underscores	Yes
Interface identifier	A unique identifier, the service identifier under the same power IoT terminal, cannot be repeated	Yes
Call method	Asynchronous: when the service is called asynchronously; synchronous: when the service is called synchronously	Yes
Inputs	Setting the inputs of the interface	No
Outputs	Setting the outputs of the interface	No
Description	Description of the attribute limited to 100 bytes	No
Extended description	Mapping relationship between the communication protocol supported by the power object terminal and the standard object model	Yes

EdgeKeeper’s edge computing capability is important. Business logic tightly coupled with devices, such as data acquisition and business control, can be encapsulated in the device APP, which can run for a long time without requiring updating unless the device fails. The data collected by these device-class APPs can be based on the rule engine in EdgeKeeper, sharing and flexibly constructing high-level business applications in different edge computing APPs. In this way, data collection and business applications can be completely decoupled. The edge computing APP can be updated at any time to meet business needs, but data collection and equipment control will not be interrupted.

3.4 Interactive protocol

The IoT agent in ubiquitous power IoT has flexible edge computing capabilities. As an important supporting device, it must cooperate with the cloud to achieve cloud-edge collaboration. The interaction protocol aims to eliminate the difference in IoT interconnection (Shen and Yang, 2015). As the standard

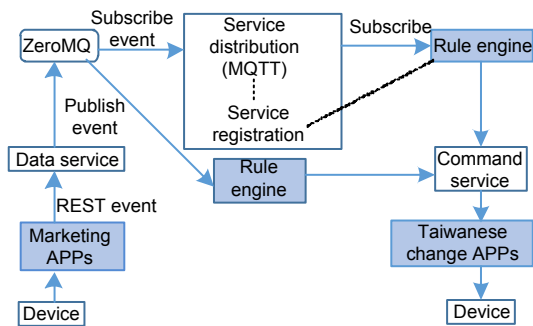


Fig. 4 EdgeKeeper edge computing

of interaction protocol between the IoT agent and cloud, it regulates the transport layer protocol technology, application layer protocol, business message category, message format and message semantics, and request and response time sequence relationship of cloud-side interaction. As specification of the north-bound interface, the interactive protocol includes the management plane and data plane. EdgeKeeper focuses on implementing the interaction specification of the management plane. For the interaction of the data plane, due to its strong business relevance, it is reserved as an extension. The communication protocol is shown in Table 2.

The EdgeKeeper interaction protocol is based on the MQTT protocol, and JavaScript object notation (JSON) is used as the format of the service message, which greatly simplifies the interaction protocol with the cloud. If the cloud issues the device upgrade command, the IoT agent receives the cloud system upgrade command and triggers a complete upgrade operation. The corresponding semantic information is shown in Table 3, and the cloud control message is as follows:

```
{
  "method": "update",
  "params": [{
    "type": "os",
    "name": "glibc",
    "version": "2.17",
    "config": ""
  }]
}
```

Table 2 Communication protocol

Name	Description
Device activation	Interactive process of activation of the agent of the IoT before online
Device access	Interactive process of agent access to cloud
Equipment upgrade	Interactive process of remote upgrading of agent equipment in the IoT
Device configuration	IoT agent accepts a remote configuration interaction process
Equipment monitoring	The cloud monitors the interaction process of the agent
Equipment control	The cloud achieves the interaction process of agent control
Remote proof	Interacting agent completes the interaction process of remote proof
Application management	The interaction process of application management in the cloud through the agent association
Child device management	The cloud achieves the interaction process of child device management through the agent association
Rule management	The interaction process of rules in the cloud management rule engine
Business control	The interaction process of business APP control through the connection agent in the cloud

3.5 EdgeKeeper developer framework

The EdgeKeeper developer framework includes mechanisms for development process control, application auditing and uploading, application deployment and life cycle management, and the necessary development tools for developers' dependencies (such as cross-platform development toolchain, virtual device-based debugging methods, development SDK (to develop business APPs), and secondary development documentation). Device service SDK supports synchronous read and write operations, asynchronous device data, driver interface initialization and destructuring, initialization and destruction, device connectivity, the automatic configuration mechanism framework, multiple types of devices with configuration files, the command triggering action, and cached query response.

After the development is completed, the business APP should be submitted to the application store of the cloud platform, and the platform management personnel should review the security, stability, and operational dependencies of the APP. After the audit is passed, the APP upgrade strategy will be managed by the platform. Business requirements are pushed to the edge association agent by device grouping. The complete process is shown in Fig. 5.

The SDK provided by EdgeKeeper is different from the edge computing SDK provided by cloud computing vendors. For example, the SDK of Alibaba, Huawei, and Tencent encapsulates the MQTT interface, but it is used mainly to encapsulate northward

interfaces and achieve interaction with cloud. The third-party service APP developers use EdgeKeeper's SDK. Based on the SDK, EdgeKeeper can accomplish automatic discovery of child devices, automatic monitoring of the APP running status, and automatic registration when APP starts. Business data such as business data collection and control commands is encapsulated in a unified interface. SDK provides a device profile template based on the object model, a business APP program development interface, and a business APP profile template. The business APP based on SDK development can implement functions of the microservice of APP, automatic caching of business data, configurable business data transmission, edge computing support, and so on. To effectively manage the resource use of the business APP in the IoT agent system, EdgeKeeper provides a container packaging tool and container running environment based on the lightweight base image. Business APP developers can quickly build and package the code into a mirror after completing SDK-based business logic. The SDK functions provided by EdgeKeeper are shown in Fig. 6.

4 Key technologies

The ubiquitous power IoT edge computing framework EdgeKeeper is the basic software of the edge layer. It is connected to the south subdevices involving various heterogeneous sensing devices and to the northbound IoT management platform to

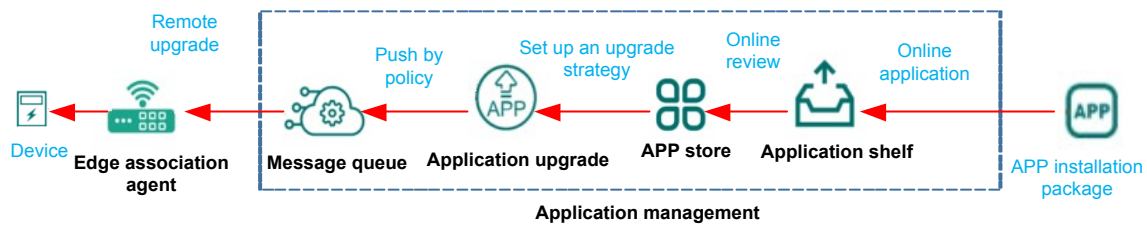


Fig. 5 EdgeKeeper application management

Table 3 Message format for device updating in the communication protocol

Field name	Type	Description
Method	String	Message type "update" indicates that the device is upgraded
Params	List	Indicating the list of parameters required for the upgrade
Params.type	String	Describing the upgrade object (OS/APP)
Params.version	String	Upgrading the target version
Params.name	String	Upgraded software specific name
Params.config	String	Other required configuration files for the upgrade

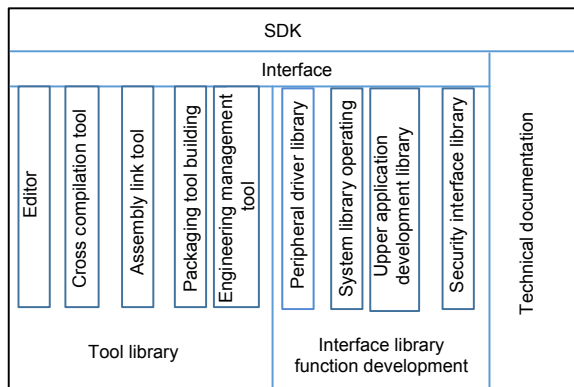


Fig. 6 EdgeKeeper software development kit (SDK)

support cloud-side collaboration. In general, there are four types of key technologies as follows:

1. Key performance. First, the ubiquitous power IoT is oriented mainly to the traditional real-time power industrial system, and the system has clear, hard real-time requirements. For example, the power load control system belongs to the hard real-time system, which needs to deal with external events in a timely manner. Otherwise, it may cause unpredictable consequences. Second, there are strict deterministic requirements for embedded real-time systems. Some of the key businesses of the system must be completed within a certain time.

2. Security and credibility. The open software ecological environment and Internet technology are introduced into the power control system, thus bringing security risks to the closed embedded systems. A large-scale open software ecological environment may contain security vulnerabilities and unknown backdoors; high-speed Ethernet access mode also provides a convenient attack path for hackers. At the same time, the ubiquitous power equipment is usually close to the user side or the transmission path, and thus it has a higher probability to be attacked by the attacker. Therefore, the security of the edge computing node is still a nonnegligible problem. However, the cybersecurity problem of the ubiquitous power IoT cannot directly copy the security solutions in the field of IT. It needs to take into account the real-time and deterministic requirements of embedded systems. At the same time, frequent upgrades and patches will affect the availability of the system, and it is not applicable to the embedded systems; therefore, it is necessary to introduce a secure and credible active defense method.

3. High reliability. Some embedded systems involve industry and personal life safety. For such systems, systematic and comprehensive failure analysis is needed to evaluate the functional safety level of the business modules in terms of the failure probability, hazard size, and hazard controllability. The bottom edge frame is required to meet the functional safety requirements, and it provides functions of fault monitoring and control, fault isolation, and fault recovery.

4. Intelligent ecology for cloud-side collaboration. First, smart IoT equipment introduces new technologies, such as the Internet, big data, and AI. These new technologies require an open intelligent software ecosystem provided by the edge framework. Most of the new technologies have been developed from the IT industry and rely on the open intelligent software ecosystem. Second, smart IoT equipment may be coordinated by multiple systems to complete work tasks. Therefore, edge framework is required to provide interconnection technology and further to provide a mechanism for interoperable mutual invocation. Therefore, excellent systems such as IOS or Android have their unique developer frameworks. On one hand, they are easy for developers to use; on the other hand, they integrate their own unique OS design concepts into business applications.

4.1 Key performance

Some power services have certain real-time requirements, which are particularly significant in the control business. The maximum interrupt response time is the most important indicator reflecting the real-time performance of the system. It represents the longest waiting time for a service interruption task. The maximum terminal response time of a typical Linux OS is often around 200 μ s, which is difficult to meet the real-time requirements. EdgeKeeper is built on NARISecOS (Yang WY et al., 2019). NARISecOS builds a layer of "microkernel OS" (Yang WY et al., 2016) under the kernel layer for distributing interrupt tasks. It also runs the NARISecOS kernel and the NARISecOS real-time kernel. Two domains are used to support legacy applications and real-time services separately, and NARISecOS ensures that real-time tasks must be responded to in real time. Experimental results show that the maximum interrupt response time of EdgeKeeper is about 10 μ s (equivalent to

those of VxWorks and other real-time OSs), which can effectively support real-time services of ubiquitous power IoT. The IoT OS hard real-time support architecture is shown in Fig. 7.

4.2 Security and credibility

4.2.1 Secure access

When the device is registered, the IoT agent EdgeKeeper sends an initialization request to the IoT hub of the IoT management platform. When the device is registered, the agent must send the corresponding device ID. The device ID is built in the system beforehand. After the device registration request is approved by the IoT management platform, the IoT agent will obtain the required three types of certificates, including the certificate authority (CA) certificate, the virtual private network (VPN) certificate, and the IoT management platform certificate. After the device is registered, the business logic can be executed. If the device is restored to factory settings, the device needs to be reregistered, but the information such as the device ID on which the device is registered should remain unchanged. Device registration should include at least three steps to request and obtain a certificate, an encryption certificate, and a remote certificate, and establish a VPN. This ensures the security of subsequent network communications, as shown in Fig. 8.

4.2.2 System security

The agent of IoT is widely distributed, and the OS level faces a greater threat to network security. EdgeKeeper uses the four-level security OS named NARISecOS (the highest security level OS) in the application of the agent, which enables the functions of two-factor authentication, mandatory access control (MAC), separation of three rights, data protection, and other functions, along with a nonclonal function, trusted execution environment (TEE), trusted platform module (TPM) computing chip, and security password module. Based on this, it implements the trusted computing security system and full disk encryption at the system level. At the same time, it focuses on strengthening the security protection of containers to ensure that the entire OS has the ability to deal with high-level security threats and provides a secure execution environment at the OS level for the business. Specifically, it includes the following security functions: (1) separation of three rights and achieving the principle of the minimum privilege; (2) MAC mandatory access control, supporting SELinux and the CAP mandatory access control model, can effectively guarantee business security; (3) two-factor authentication to support the OS; (4) disk encryption to ensure data confidentiality; (5) container protection: supporting container resource limits, access control, and image integrity scan inspection to fully protect

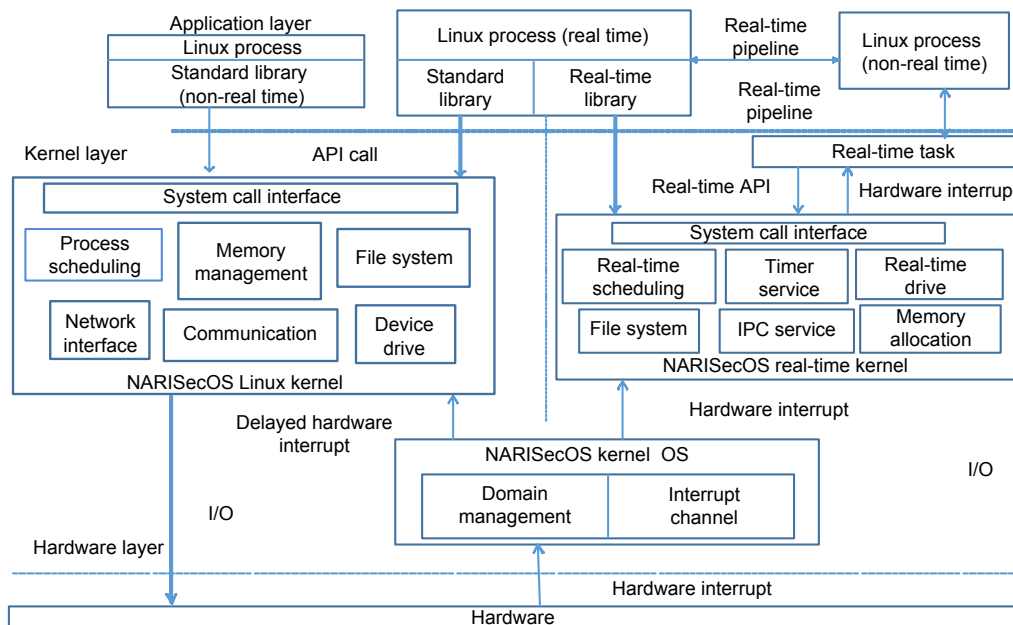


Fig. 7 IoT OS hard real-time support architecture

the container. The security features of the IoT four-level security operating system are shown in Fig. 9.

4.2.3 Trusted authentication

The newly released “level protection 2.0” proposes a clear requirement of “trusted authentication.” In addition, based on trusted computing, most malicious programs such as Trojan horses can be eliminated. Therefore, trusted authentication must be implemented in EdgeKeeper. The following four core functions are achieved:

1. Trusted startup. Firmware and OS are modified based on the board card fuse mechanism and built-in TPM chip, to achieve step-by-step trusted authentication from chip to system startup and efficient full disk encryption, and to prevent devices from being injected malicious code offline or online and the disclosure of sensitive data.

2. Trusted metrics. Based on the secure OS and digital certificate system, a lightweight and reliable metric framework can be implemented, which can ensure that only the authenticated applications can be installed and run in the OS without affecting the availability of the system, and solve the security threats of malicious viruses and Trojan horses.

3. Remote trusted certificate. Based on the TPM and lightweight measurement framework, by remotely collecting the device status of the IoT agent, the IOT management center can remotely evaluate

whether the device is in a trusted state, and support the solution to remote security upgrade problems of firmware, operating system, and application. The flowchart of remote certification based on the TPM chip is shown in Fig. 10.

4. Security upgrade. The firmware, OS, and application in the IoT agent have upgraded requirements. It is urgent to lower the security risks of the remote upgrade. Based on the remote trusted certificate capability, the firmware download is implemented through the platform firmware or uploaded manually to upgrade the update and support the ability to remotely update the device in various ways such as silent, mandatory, and orientation. It can solve the

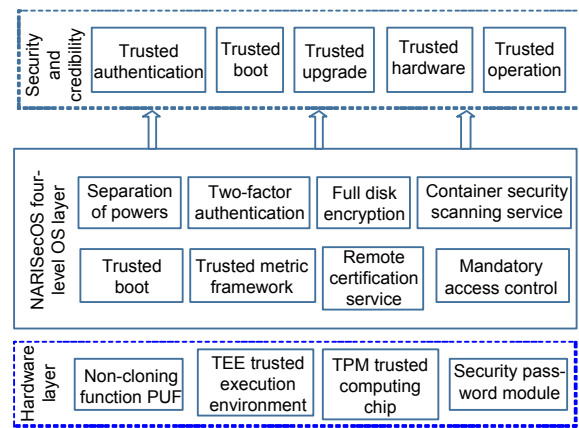


Fig. 9 Security features of the IoT four-level security operating system

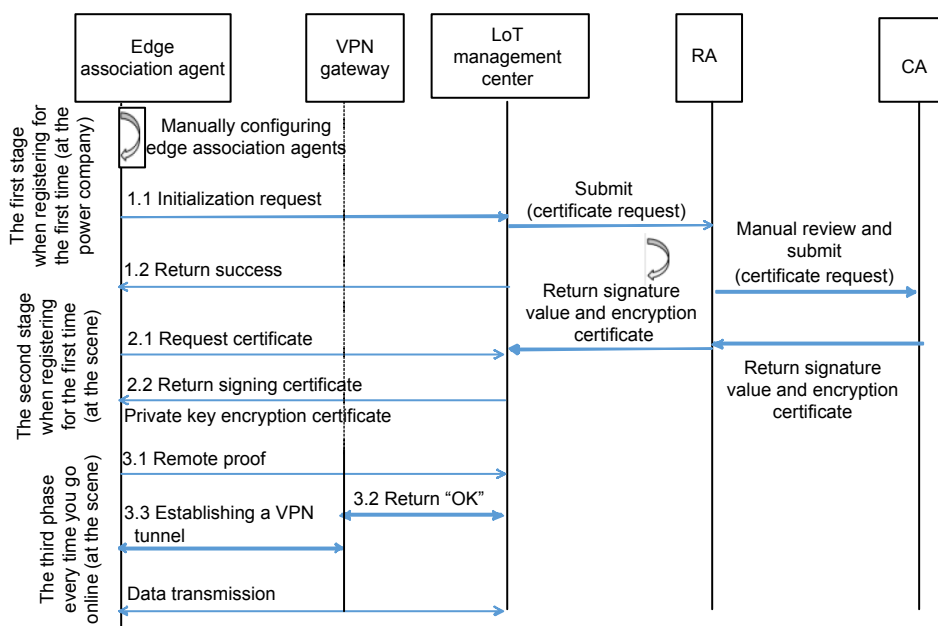


Fig. 8 Initial process of secure access (CA: certificate authority; RA: registration authority)

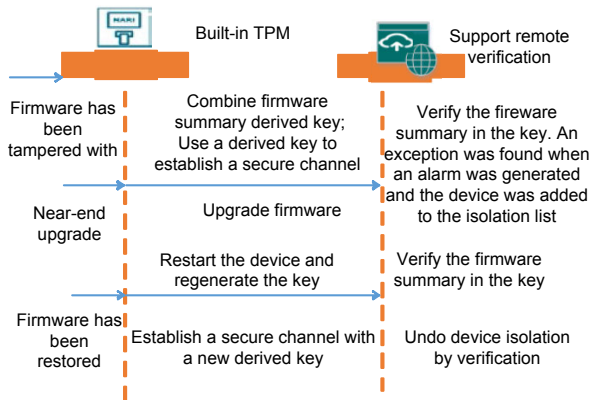


Fig. 10 Remote certification based on a trusted platform module (TPM) chip

upgrade problems of firmware, OS, and application of the agent.

For firmware and OS upgrades, the upgrade service in the IoT agent will take the device status to the over-the-air (OTA) service in the IoT platform, and the OTA service will perform remote certification. After the certificate is verified, the upgrade service receives the upgrade package and uses the device public key to verify the upgrade. The potential security risks of the upgrade are lowered through this two-way authentication. For the upgrade of application, referring to the application management mechanism of IOS, the agent needs only to store the root certificate (public key) of the “detection organization,” and the detection organization can issue the second-level certificate to the business application research and development (R&D) institution. The APP signed by the second-level certificate can be applied directly to the IoT agent, which solves the problems of security and availability.

4.3 High reliability

4.3.1 Partition-based fault isolation technology

Isolation technology supports the robust development of edge computing. Edge devices need to provide effective isolation technology to ensure service reliability and quality. We need to consider two aspects in isolation technology: (1) isolation of computing resources (i.e., applications cannot interfere with each other); (2) isolation of data (i.e., different applications should have different access rights). EdgeKeeper uses a microkernel architecture to provide partition isolation mechanism for different

functional security-level services in user mode. When one service fails, the other partition services are not affected. As shown in Fig. 11, specific technologies include time isolation mechanism between partitions, spatial isolation mechanism, permission isolation, and hardware isolation mechanism. Among them, spatial isolation is one of the most direct control methods. In the past, our process was focused mainly on the isolation of the memory address space. After introducing the containers, we achieved the spatial isolation of the file system, network, process identifier (PID), users, and so on. The namespace is used for spatial isolation, while cgroup is used for resource isolation. In terms of privilege isolation, on one hand, referring to the Linux sandbox mechanism, applications cannot interact with each other. Applications running in the process sandbox are not allotted privileges and cannot access the system or resources. Different EdgeKeeper applications that are restricted to sandboxes do not interfere with each other, and damage to the system and other applications can be minimized. The sandbox mechanism of the EdgeKeeper application is shown in Fig. 11. Applications that do not have a trust relationship are isolated from each other and run alone. On the other hand, based on the MAC at the OS level, the operation of the business domain is minimized, and the scope of access to the business is limited mainly by the access control model. To enrich the access control mechanism of EdgeKeeper, we introduce MAC modes such as consistency availability partition (CAP) tolerance, Bell-LaPadula (BLP), and the BIBA model. The CAP model gives the user only the minimum ability of each privileged process to perform its functions, thus implementing the minimum operational domain of the business and including more than 30 permissions such as allowing access to the network, accessing peripherals, and shutting down. BLP and BIBA are the implementation models of traditional confidentiality and integrity in the system, respectively.

4.3.2 System operation monitoring control and multi-level abnormal processing technology

The edge layer OS provides different granularities of system operating state monitoring and control functions, including task-, partition-, system-, and hardware-level operation monitoring control. Through different levels of health monitoring and

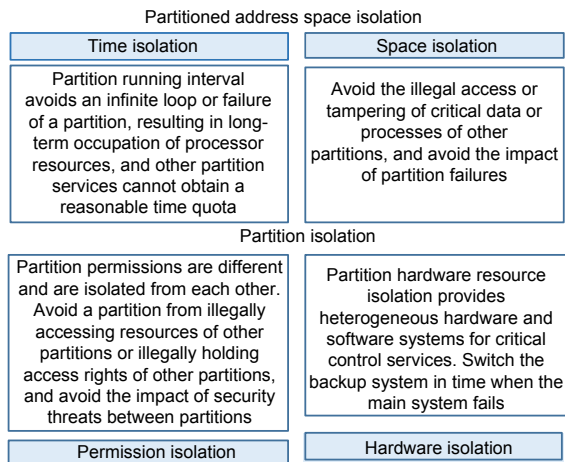


Fig. 11 Partition-based fault isolation

control functions, different ranges of anomalies or alarm events can be discovered in time, and the running status of key tasks can be monitored in time to ensure that their operating status is consistent with expectations. When different types of system failures are found, the OS provides a multilevel exception handling mechanism. The exceptions are processed step by step through the task-, partition-, system-, and hardware-level control to ensure that each abnormal problem can be recovered with the minimum impact on the system and the minimum impact range, avoiding the excessive impact of recovery abnormality. EdgeKeeper designs functional safety features based on high-reliability industry standards to monitor the operation status of a control system in time and restores the operating status of the system in a fine-grained manner, to ensure the continuous and stable operation of high-reliability businesses.

4.4 Intelligent ecology for cloud-edge collaboration

4.4.1 Application development

In application development, EdgeKeeper implements the unified management of the APP running on intelligent terminals and edge agents, including the APP's trusted authentication, APP version management, and APP upgrade policy management. The edge linkage framework provides the basic development framework and the IoT infrastructure enablement for design developers (distributed on the IoT management platform side). After the developers have passed the review by the management department, they can quickly develop applications through

the development framework and the IoT provided by the center within their jurisdiction. The IoT management platform can provide a specific development environment and interfaces as follows: (1) basic components, platform interaction components, and container components, which support the development, compilation, and packaging of the IoT application; (2) development API. Various types of communication interfaces, security management interfaces, and so on support the rapid development and integration of various types of IoT applications. The IoT developer framework is shown in Fig. 12.

4.4.2 Intelligent ecology

EdgeKeeper uses the microkernel architecture for reference. All kernel functions and business functions are provided as services to the outside world, supporting local and remote connections and access. First, the intelligent application service business registers with the local service manager and publishes the service by the service manager. The local business accesses the service manager through the inter-process communication (IPC) mechanism, and the service manager establishes the service connection to achieve the local access service function. The remote service also accesses the service manager (which is responsible for establishing the remote service communication channel) and finds the service node. Through this technology, the function of intelligent interconnection and interoperability across nodes can be achieved, and application services can transparently use local services or remote services. The edge framework EdgeKeeper provides service protocols and security management protocols for application interconnection and interoperability for different scenarios, including constrained application protocol (CoAP) and application interconnection protocols such as the MQTT protocol. Through the above-mentioned application interconnection and management-and-control protocol, a complete ubiquitous power IoT system is formed by achieving the coordination of various heterogeneous systems.

4.4.3 Cloud-edge collaboration

EdgeKeeper implements the edge computing framework supporting function computing, rule engine, and flow calculation based on the application management of basic functions, in addition to providing intelligent services for image recognition,

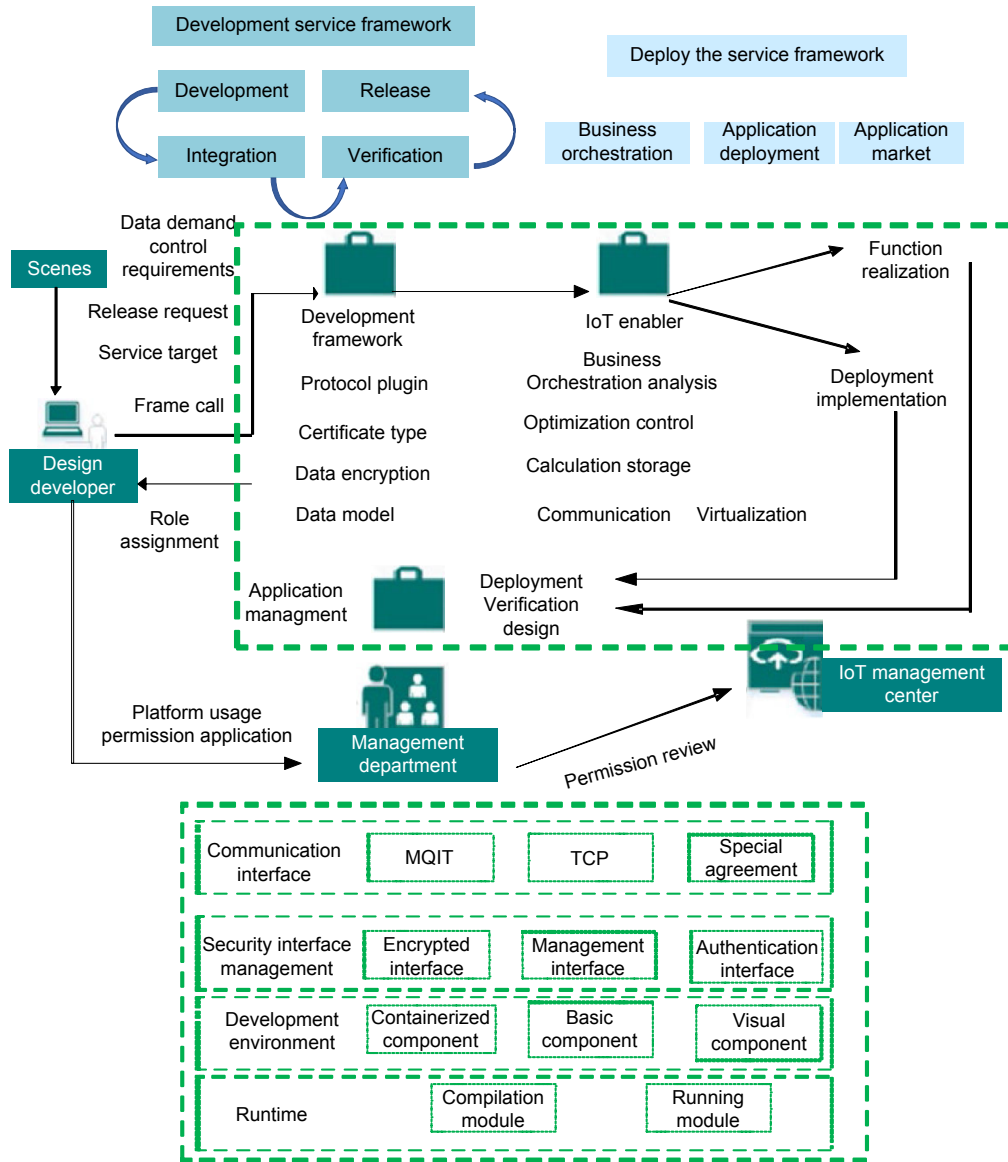


Fig. 12 IoT developer framework

machine learning model prediction, and speech recognition. We believe that the core of cloud-edge collaboration is to further achieve the cloud-side collaboration system, with the coverage of resource collaboration, data collaboration, intelligent collaboration, application management collaboration, business management collaboration, and service collaboration. Among them, (1) resource coordination refers to the life cycle management of the edge node infrastructure, equipment, and southbound resources; (2) data collaboration refers mainly to the edge collection and centralized analysis of data; (3) intelligent collaboration refers to the centralized training of data

on the platform side, distribution of intelligent reasoning, and sending of the trained model to the edge frame side for execution; (4) application management collaboration refers to the full life cycle management of application development and its testing/application; (5) business management collaboration refers to the unified management of business applications; (6) service coordination refers to the unified arrangement of services. These six types of collaboration can basically meet the needs of all cloud-side collaborative application scenarios of ubiquitous power IoT. The edge-based cloud edge collaboration is shown in Fig. 13.

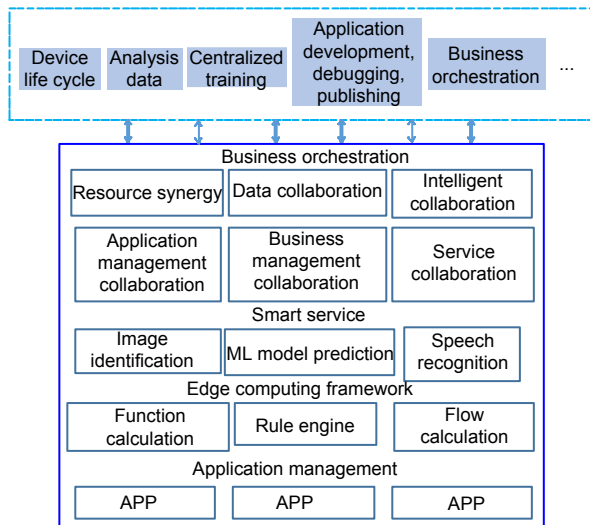


Fig. 13 Edge-based cloud-edge collaboration

5 Experiments

To verify the validity and adaptability of the EdgeKeeper framework, we detail the experimental verification work on function, non-functionality, performance, and application scenario verification. Because the cloud-side interaction protocol between the edge framework (edge-linking agent) and the IoT management platform is suitable for testing by the platform side, some tests are initiated from the side of the IoT management platform to verify the support of the edge computing framework in the IoT agent. The test targets four edge computing frameworks, i.e., EdgeKeeper, OpenEdge, EdgeX, and KubeEdge.

5.1 Main tests

5.1.1 Functional test

The basic functions of the edge framework are verified, i.e., device access, device management capability, model definition and delivery, device shadow, data communication capability, data collection, data distribution, firmware upgrade, APP management, rule engine, and operation and maintenance management, as shown in Table 4.

5.1.2 Nonfunctional test

Based on the special requirements of the IoT application scenario, the test and verification of the remote operation and maintenance capability of the IoT agent and nonfunctional indicators of the edge

framework are carried out (Table 5), including eight test items, such as remote configuration of the agent of the IoT, remote monitoring, remote debugging, reliability, openness, and security of the edge framework.

5.1.3 Performance test

To meet the access of huge-scale devices, we verify mainly the access performance and system performance of the edge framework (Table 6), including 10 indicators, i.e., the maximum number of simultaneous connections supported by a single node, the number of messages that the edge frame can process per second, the number of messages that can be processed by a single node, the performance of commands sent by the edge frame, the performance of commands sent by a single node, the maximum number of online users, the average response time of the core functions, 8-h continuous reporting information, and 8-h continuous issuing instructions. In terms of testing ideas, first, we need to provide professional performance testing tools and complete the writing of test scripts. Testers use test tools to execute test scripts. Second, in the design of test items, considering the limited resources of the laboratory, the test may not be able to meet the full demand of the whole business. Therefore, two types of cases of a single node and edge frame platform (multiedge agent nodes through platform test) are designed for the same test item, and the growth correlation curve between performance indicators and resources is analyzed to evaluate the performance of the edge framework. Third, there are differences in the deployment of edge frameworks among vendors. Therefore, the resources of each edge framework in the laboratory are slightly different.

To ensure the fairness of the test, the resources of the supporting nodes corresponding to the test items are required to be consistent for each framework.

5.2 Application scenario verification

According to the actual business requirements of the power grid side and user side, application scenario verification is carried out and the support capability of the edge object association framework for each application scenario is emphasized. Through building an application scenario simulation environment in the laboratory, we will carry out end-to-end business verification from terminals, IoT agents, and IoT

Table 4 Functional test

Test item	Test point	Requirement
Device access	Device registration and access	After device registration is completed on the platform function page, the device can access the platform; Batch registration of devices is supported, and all registered devices can access the platform
	Provide access to the SDK	Provide basic SDK under different platforms (Linux and Android platforms)
	Data transfer protocol	Support data transfer protocols, such as MQTT, CoAP, HTTP, and WebSocket
Model definition and delivery	Tenant and IoT agency relationship	Support the same tenant to manage multiple IoT agents; The same object agent can be managed by multiple tenants, but the resource requirements of different tenants have permission control
	Object model definition	Object model supports multiple levels; Attributes defined by the model can be added, modified, or deleted; Data format should be standardized
	Object model	Model definition can be issued to the edge object association agent; Modified model definition should be updated and sent to the edge object association agent
Device shadow	Data reporting	Terminal device can report the data to the platform according to the model definition; After the model is updated, the data is reported according to the new model definition
	Device shadow editing	Device shadow data model definition, modification, and deletion; Device shadow status view
	Status change	Business APP modifies the device shadow state data; The device shadow state data is asynchronously sent to the terminal device
Equipment communication capability	MQTT protocol	Support QoS=0 and QoS=1 message characteristics of the MQTT protocol
	Offline storage capability	Support offline storage capabilities of device messages (including reporting and delivery)
Data collection	Import device messages into message queue	Support device message that imports message queue; Implement asynchronous message communication between devices and the third-party services
	Temporary storage of data	Support caching the collected data on the platform
	Data error retransmission	Support data error retransmission
Data distribution	Status change	Support the data reported by the terminal device to be distributed to different message queues and different databases; Upper-layer business APPs can directly use the data
	Data subscription	Support the data directly subscribed by business applications
	Sending data command	Support upper-layer applications to send data commands and consume messages through API interfaces
Firmware upgrade	Firmware upgrade	Platform upgrades firmware for a single device or batch devices
APP management	Management of upper and lower shelves of the APP	Support management of the APP
	Remote upgrade of the APP	Support remote installation and upgrade of APP for a single object agent and grayscale release; Support batch operation of IoT agents with the same APP installed

To be continued

Table 4

Test item	Test point	Requirement
APP management	APP remote configuration	Support APP remote configuration for a single object agent and multiple IoT agents
	APP version management	Support APP version management, including a single device and batch devices
Rule engine	Rule configuration	Add, modify, and delete rules; Enable and stop operations on rules
	Class structured query language (SQL) syntax and underlying semantic operations	Rule description supports class structured query language (SQL) syntax and basic semantic operations
Operation and maintenance management	Operation and maintenance management	Support the operation log viewing and downloading of platform services, edge agent devices, and terminal devices; Require platform service installation and upgrade to support automation operations

Table 5 Nonfunctional test

Test item	Test point	Requirement
Remote configuration	Device remote configuration	Support the connection and modification of the edge object agent device and the terminal device; Support the delivery of the modified configuration file to the edge agent device and the terminal device
	System remote configuration operation	Support patch upgrades and configuration updates for the edge agent device and the terminal device operating system
Remote monitoring	System remote monitoring operation	Support the operation status monitoring of the edge IOT agent equipment and the terminal device operating system, including the CPU utilization rate and memory utilization rate
	Terminal operation status monitoring	Support status monitoring of the edge agent based proxy devices and the terminal devices, such as offline and online conditions
	Application status monitoring	Support monitoring of the operation of the edge IoT proxy devices and the terminal devices, such as log acquisition
	Alarm information management	Support collection and message sending of alarm information of the edge agent device and the terminal device
Remote debugging	Edge proxy device remote debugging	Support remote viewing, analysis, fault recovery, and so on for edge proxy devices
	Remote debugging of terminal equipment	Support remote viewing, analysis, fault recovery, and so on for terminal equipment
Reliability	Batch device online success rate	Time and success rate of re-launching the device after it goes offline
	Cluster high availability deployment	When some devices fail, the services provided by the platform are uninterrupted, and the impacts of the software, hardware, and human-induced faults on the service are minimized
Security	Edge proxy device access security	Register and authenticate the device with a key or other means
	Edge proxy device transmission security	Provide standard transport layer security (TLS) or other high-level encryption for transmission encryption
	Message publishing subscription security and safety	The publishing and subscribing capabilities of messages have strict and secure authority control; Support operation authority control of the same resource under multiple accounts
	API authentication	Have authentication of the IoT API interface

To be continued

Table 5

Test item	Test point	Requirement
Security	Platform login security	Provide a unified login authentication system
	Platform security	Support identity authentication, access control, security audit, software fault tolerance, and resource control; The user is assigned rights, and the management is separated from the business account
Flexibility	Northbound interface	Support upper-layer services to subscribe to messages through different data formats
	Southbound interface	Support push commands or messages to the terminal for configuration
Openness	Northward openness	Northbound API interface supports secondary development
	Southward openness	Southbound API interface supports secondary development
Loose coupling	Support for mainstream databases	Support Oracle, DB2, SQL server, MySQL, and other mainstream databases
	Support grayscale update capability	Loose coupling between platform components and grayscale upgrade
	Support component automatic expansion and contraction	Support multinode multilevel deployment

Table 6 Performance test

Test point	Requirement
The maximum number of simultaneous connections	The platform supports ≥ 5000 connections; The connection lasts 8 h without interruption; Device central processing unit (CPU) and memory usages are $< 85\%$
Single node supporting the maximum number of simultaneous connections	A single device node supports ≥ 1000 connections; The connection lasts 8 h without interruption; Device CPU and memory usages are $< 85\%$
The number of messages that the platform can process per second	The platform supports processing of ≥ 500 messages per second; Device CPU and memory usages are $< 85\%$
The number of messages that a single node can process per second	Single node supports processing ≥ 100 messages per second; Device CPU and memory usages are $< 85\%$
Performance of instructions issued by the platform	Require the platform to support the batch delivery of instructions to 500 devices within 1 min; Success rate of issuing instructions is 100%; Device CPU and memory usages are $< 85\%$
Performance of single-node delivery instructions	A single device node is required to deliver instructions to a batch of 100 devices in 1 min; Success rate of issuing instructions is 100%; Device CPU and memory usages are $< 85\%$
The maximum number of online users	Support 100 users online at the same time (reference value is the maximum number of users in the marketing business, i.e., 18 000); Device CPU and memory usages are $< 85\%$
Average response time of the core function	In the case where the database has > 100 million data volumes in the corresponding core function table, the average response time is within 3 s; Device CPU and memory usages are $< 85\%$
Support 8-h continuous reporting information	1100 devices continuously report information for 8 h at an interval of 1 s, requiring an average response time of 1 s, and transaction volume per second (TPS) is not less than 50; The information processing success rate is 100%; Device CPU and memory usages are $< 85\%$
Support 8-h continuous delivery instructions	The platform continuously delivers instructions to 100 devices in batches for 8 h at an interval of 1 s, requiring an average response time of 1 s, and TPS is not less than 50; The information processing success rate is 100%; Device CPU and memory usages are $< 85\%$

management platforms to business applications, and test the connectivity of the edge framework in the south, north, and complete links. Each business scenario is based on the same southward and northward environment to complete the access of each edge framework. It carries out mainly the application scenario validation of the station area, distribution station, transmission line, integrated energy services, and so on. The verification content is illustrated with the station area scenario as an example. To improve the operation management and customer service level of the substation distribution network, automatic collection of perception information at the substation side, low-voltage line side, and user side is realized, and station status by deploying environmental sensors, monitoring units, smart meters, and other end devices is also realized. By the intelligent distribution terminal transformer terminal unit (TTU), the concentrator and other side devices achieve information collection and processing, and the collected data is transmitted to the intranet system using a wireless private network/public network communication method (Fig. 14).

The test in this project tests and verifies mainly the various power IoT business scenarios, such as meter data reporting, meter data calling, electricity

meter control instructions issued, distribution monitoring data reporting, and remote debugging of the circuit breaker by simulating the low-voltage station area scenario and verifying the adaptability of the edge IoT framework to power professional applications (Table 7).

5.3 Test results

The overall test results of EdgeKeeper, OpenEdge, EdgeX, and KubeEdge are shown in Table 8, where “●” indicates full support, “○” indicates no support, and “◎” indicates partial support. As can be seen from Table 8, EdgeKeeper has passed the tests of all functions, non-functions, performance, and application scenarios, and the performance is the most complete. EdgeX is not satisfactory in terms of nonfunctional test. The KubeEdge does not perform well in non-functional testing and application scenarios. EdgeKeeper edge computing framework has performed well in all tests. OpenEdge does not perform well in some application scenarios, but in other tests, it performs relatively well. In the functional test, EdgeKeeper has the characteristics of perfect basic functions and high reusability; in the nonfunctional test, it has the characteristics of practicality, reliability, flexibility, loose coupling, and so on; in the

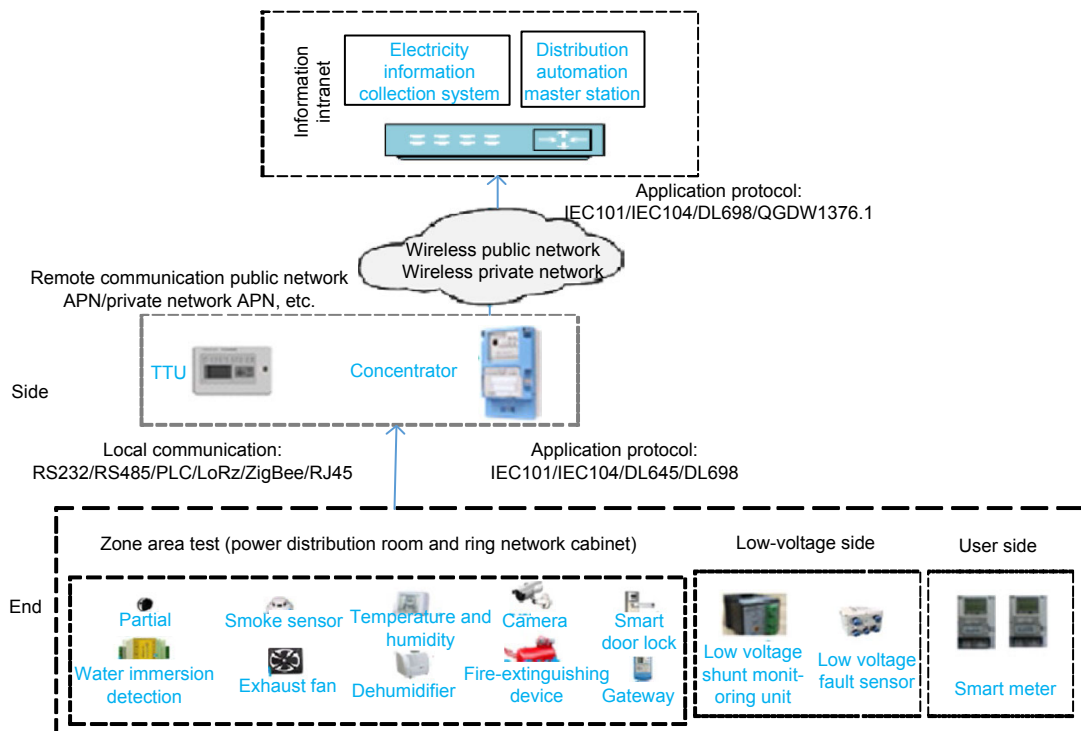


Fig. 14 Overview of the collection of the area

Table 7 Application scenario test

Test item	Test point	Requirement
Zone area scene	Meter data reporting	Smart meter sends data to the mining system through the concentrator, edge object association agent, and object management platform
	Meter data calling	Use the mining system to send an instruction to collect meter data through the IoT management platform; The acquisition instruction is sent to the meter by the agent of the IoT; The smart meter returns data to the mining system through the edge agent and agent management platform
	Electricity meter control instructions issued	Use the mining system to send control commands to the electricity meter through the IoT management platform and the edge object association agent; After the smart meter completes the action, it returns the status signal to the mining system through the edge object association agent and object management platform
	Distribution monitoring data reporting	The smart capacitor, circuit breaker, and branch monitoring unit send status data to the TTU; The TTU terminal sends the collected information such as the hanging device to the object management platform through the edge object association agent
	Remote debugging of the circuit breaker	The DMS system sends control commands to the circuit breaker through the IoT management platform, edge agent, and TTU; After the circuit breaker completes the action, the status signal is returned to the DMS system through the TTU, edge object association agent, and IoT management platform

Table 8 Test results

Verification test	Test item	Test point	OpenEdge	EdgeX	KubeEdge	EdgeKeeper
Functional test	Device access	Device registration and access	●	◎	●	●
		Access to the SDK	●	◎	○	●
		Data transfer protocol	●	◎	●	●
		Tenant and IoT agency relationship	●	◎	●	●
	Model definition and delivery	Object model definition	●	◎	●	●
		Object model launching	●	◎	●	●
		Data reporting	●	●	●	●
	Device shadow	Device shadow editing	●	●	◎	●
		Data reporting	●	◎	◎	●
		Status change	●	●	◎	●
	Equipment communication capability	MQTT protocol	●	●	●	●
		Offline storage capability	●	●	●	●
	Data collection	Device message import message queue	●	●	●	●
		Temporary storage of data	●	●	◎	●
		Data error retransmission	●	●	◎	●
	Data distribution	Status change	●	●	●	●
Data subscription		●	●	●	●	
Send data command		●	●	●	●	
Firmware upgrade	Firmware upgrade	◎	●	◎	●	

To be continued

Table 8

Verification test	Test item	Test point	OpenEdge	EdgeX	KubeEdge	EdgeKeeper
Functional test	APP management	Management of upper and lower shelves of the APP	●	●	◎	●
		Remote upgrade of the APP	●	●	◎	●
		APP remote configuration	●	●	◎	●
		APP version management	●	●	◎	●
	Rule engine	Rule configuration	●	●	◎	●
		Class SQL syntax and underlying semantic operations	●	●	◎	●
	Operation and maintenance management	Operation and maintenance management	◎	◎	●	●
	Remote configuration	Device remote configuration	●	○	○	●
		Operating system remote configuration	●	●	○	●
	Remote monitoring	Operating system remote monitoring	●	●	○	●
Terminal operation status monitoring		●	●	○	●	
Application status monitoring		●	●	○	●	
Alarm information management		●	●	○	●	
Remote debugging	Edge proxy device remote debugging	●	●	○	●	
	Remote debugging of terminal equipment	●	●	○	●	
Reliability	Batch device online success rate	●	●	○	●	
	Cluster high availability deployment	●	●	○	●	
Non-functional test	Edge proxy device access security	Edge proxy device access security	●	●	○	●
		Edge proxy device transmission security	●	●	●	●
	Safety	Message publishing subscription security	●	○	●	●
		API authentication	●	○	●	●
		Platform login security	●	○	●	●
		Platform safety	●	○	●	●
	Flexibility	Northbound interface	●	○	●	●
Southbound interface		●	○	●	●	
Openness	Northward openness	●	○	●	●	
	Southward openness	●	○	●	●	
Loose coupling	Support of mainstream databases	◎	○	○	●	
	Grayscale update capability	●	○	○	●	
	Support of component automatic expansion and contraction	●	○	○	●	

To be continued

Table 8

Verification test	Test item	Test point	OpenEdge	EdgeX	KubeEdge	EdgeKeeper
Performance test	Performance test	The maximum number of simultaneous connections	⊙	○	○	●
		The number of messages a single node that can process per second	●	○	○	●
		Performance of issuing instructions	●	○	○	●
		The maximum number of online users	●	○	○	●
		Average response time of the core function	⊙	○	○	●
		Support of 8-h continuous reporting information	⊙	○	○	●
		Support of 8-h continuous delivery instructions	⊙	○	○	●
Application scenario	Zone area scene	Meter data reporting	⊙	●	●	●
		Meter data calling	○	○	○	●
		Remote meter status	○	○	○	●
		Distribution monitoring data reporting	●	●	●	●
		Remote debugging of the circuit breaker	●	●	●	●

“●” indicates full support, “○” indicates no support, and “⊙” indicates partial support

performance test, it has the characteristics of perfect basic functions and high reusability; in the performance test, it has the characteristics of high load, fast response, and zero error in issuing instructions, which can well meet the requirements of high load and real-time performance; in the application scenario test, it has high adaptability to the professional application scenario of power. It can be seen that EdgeKeeper is currently the most suitable edge computing framework for the ubiquitous power IoT.

6 Conclusions

Edge computing refers to providing the nearest-end service on the side close to the object or data source. Its applications are launched on the edge side, resulting in rapid network service response and meeting the basic needs of the industry in real-time business, application intelligence, security, and privacy protection. Edge computing works between physical entities and industrial connections, or at the top of the physical entities. In cloud computing, historical data of edge calculations can still be accessed. In the construction of ubiquitous power IoT, the core

of the ubiquitous power IoT is to construct an edge computing framework, which is suitable for ubiquitous power IoT. For this purpose, an edge-trusted computing framework named EdgeKeeper is designed and implemented, which completes the design of object model, edge computing, cloud-edge interaction, and breakthrough key technologies, yielding features such as good performance, good security, good reliability, high reliability, and intelligent ecology. Through functional, nonfunctional, performance, and application scenario tests, and comparison with OpenEdge, EdgeX, and KubeEdge, EdgeKeeper illustrates its advantages in business satisfaction and adaptability.

In the future, we will continue to optimize the architecture of EdgeKeeper based on the existing work and continue to enrich the application ecology of the ubiquitous power IoT with the business units and help the construction of the IoT.

Contributors

Weiyong YANG designed the EdgeKeeper framework. Wei LIU analyzed the EdgeKeeper framework and designed the experimental scheme. Xingshen WEI and Huang HAO analyzed the experimental data. Weiyong YANG and Kangle YANG drafted the manuscript. Zixin GUO analyzed the

experimental scheme and provided materials and analysis tools. Longyun QI studied the EdgeKeeper framework in depth and proposed a modification plan which is of constructive significance. Kangle YANG participated in the experiment and revised and finalized the paper.

Compliance with ethics guidelines

Weiyong YANG, Wei LIU, Xingshen WEI, Zixin GUO, Kangle YANG, Hao HUANG, and Longyun QI declare that they have no conflict of interest.

References

- Ahmed R, Zaheer Z, Li R, et al., 2018. Harpocrates: giving out your secrets and keeping them too. *IEEE/ACM Symp on Edge Computing*, p.103-114. <https://doi.org/10.1109/SEC.2018.00015>
- Ai Y, Peng M, Zhang KC, 2018. Edge computing technologies for Internet of Things: a primer. *Dig Commun Netw*, 4(2):77-86. <https://doi.org/10.1016/j.dcan.2017.07.001>
- Aral A, Brandic I, 2018. Dependency mining for service resilience at the edge. *IEEE/ACM Symp on Edge Computing*, p.228-242. <https://doi.org/10.1109/SEC.2018.00024>
- Boutaud F, Ehlig PN, 1991. Series Maximum/Minimum Function Computing Devices, Systems and Methods. US Patent 5 072 418, USA.
- Cai YM, Feng SY, Du HW, et al., 2019. Novel edge-ware adaptive data processing method for the ubiquitous electric power Internet of Things. *High Volt Eng*, 45(6):1715-1722 (in Chinese). <https://doi.org/10.13336/j.1003-6520.hve.20190604005>
- Chao MY, Yang C, Zeng YK, et al., 2018. F-MStorm: feedback-based online distributed mobile stream processing. *IEEE/ACM Symp on Edge Computing*, p.273-285. <https://doi.org/10.1109/SEC.2018.00027>
- Chen XL, Wan S, Zhu YF, et al., 2019. Analysis of distributed power distribution fault processing based on edge computing. *Electromech Inform*, (17):32-33 (in Chinese). <https://doi.org/10.19514/j.cnki.cn32-1628/tm.2019.17.018>
- Edge Computing Consortium, 2018. Edge Computing Reference Architecture 3.0. <http://www.econsortium.org/Uploads/file/20190225/1551059767474697.pdf> [Accessed on Sept. 12, 2019].
- Feng ZQ, George S, Harkes J, et al., 2018. Edge-based discovery of training data for machine learning. *IEEE/ACM Symp on Edge Computing*, p.145-158. <https://doi.org/10.1109/SEC.2018.00018>
- Fultz D, Ramanujan AS, Ibitayo KY, 2010. Rules Engine Architecture and Implementation. US Patent 7 853 786, USA.
- Hu YC, Patel M, Sabella D, et al., 2015. Mobile Edge Computing—A Key Technology Towards 5G. ETSI White Paper No. 11, ETSI, France.
- Jang SY, Lee Y, Shin B, et al., 2018. Application-aware IoT camera virtualization for video analytics edge computing. *IEEE/ACM Symp on Edge Computing*, p.132-144. <https://doi.org/10.1109/SEC.2018.00017>
- Li JR, Li XY, Gao YL, et al., 2018. Review on data forwarding model in Internet of Things. *J Softw*, 29(1):196-224 (in Chinese). <https://doi.org/10.13328/j.cnki.jos.005373>
- Li SN, Luo GJ, 2014. The overview of technologies and applications for industrial IOT. *Telecommun Netw Technol*, (3):26-31 (in Chinese).
- Liang JY, Liu B, Liu F, 2019. The present situation of open source platforms for edge computing. *ZTE Technol*, 25(3):8-14 (in Chinese). <https://doi.org/10.12142/ZTETJ.201903002>
- Liu RL, Liu HT, Xia SF, et al., 2019. Internet of Things technology application and prospects in distribution transformer service area management. *High Volt Eng*, 45(6):1707-1714 (in Chinese). <https://doi.org/10.13336/j.1003-6520.hve.20190604004>
- Luan TH, Gao LX, Li Z, et al., 2015. Fog computing: focusing on mobile users at the edge. <https://arxiv.org/abs/1502.01815>
- Mach P, Becvar Z, 2017. Mobile edge computing: a survey on architecture and computation offloading. *IEEE Commun Surv Tutor*, 19(3):1628-1656. <https://doi.org/10.1109/COMST.2017.2682318>
- Maheshwari S, Raychaudhuri D, Seskar I, et al., 2018. Scalability and performance evaluation of edge cloud systems for latency constrained applications. *IEEE/ACM Symp on Edge Computing*, p.286-299. <https://doi.org/10.1109/SEC.2018.00028>
- Mao YY, You CS, Zhang J, et al., 2017. A survey on mobile edge computing: the communication perspective. *IEEE Commun Surv Tutor*, 19(4):2322-2358. <https://doi.org/10.1109/COMST.2017.2745201>
- Satyanarayanan M, 2017. The emergence of edge computing. *Computer*, 50(1):30-39. <https://doi.org/10.1109/MC.2017.9>
- Saxena H, Salem K, 2015. EdgeX: edge replication for web applications. 8th Int Conf on Cloud Computing, p.1041-1044. <https://doi.org/10.1109/CLOUD.2015.147>
- Sha LT, Xiao P, Chen W, et al., 2018. Leakage perception method for backdoor privacy in industry Internet of Things environment. *J Softw*, 29(7):1863-1879 (in Chinese). <https://doi.org/10.13328/j.cnki.jos.005356>
- Shen SB, Yang Z, 2015. Architecture of Internet of Things and its standardization. *J Nanjing Univ Post Telecommun (Nat Sci)*, 35(1):1-18 (in Chinese). <https://doi.org/10.14132/j.cnki.1673-5439.2015.01.001>
- Shi WS, Dustdar S, 2016. The promise of edge computing. *Computer*, 49(5):78-81. <https://doi.org/10.1109/MC.2016.145>
- Shi WS, Cao J, Zhang Q, et al., 2016. Edge computing: vision and challenges. *IEEE Int Things J*, 3(5):637-646. <https://doi.org/10.1109/JIOT.2016.2579198>
- Shi WS, Sun H, Cao J, et al., 2017. Edge computing—an emerging computing model for the Internet of Everything era. *J Comput Res Dev*, 54(5):907-924 (in Chinese). <https://doi.org/10.7544/issn1000-1239.2017.20160941>
- Wang H, Li Y, Mi MR, et al., 2013. Secure data fusion method

- based on supervisory mechanism for industrial Internet of Things. *Chin J Sci Instrum*, 34(4):817-824 (in Chinese). <https://doi.org/10.3969/j.issn.0254-3087.2013.04.016>
- Xu H, 2019. Implementation of edge calculation in motor monitoring system. *Electron Technol Soft Eng*, (11):190-192 (in Chinese).
- Yang WY, Liu W, Huang H, et al., 2016. Research on power private micro kernel-based secure operating system technology. *Electron Power Inform Commun Technol*, 14(11):22-27 (in Chinese). <https://doi.org/10.16543/j.2095-641x.electric.power.ict.2016.11.004>
- Yang WY, Liu W, Wei XS, et al., 2019. Micro-kernel OS architecture and its ecosystem construction for ubiquitous electric power IoT. *IEEE Int Conf on Energy Internet*, p.179-184. <https://doi.org/10.1109/ICEI.2019.00038>
- Yang YM, Song ZH, 2015. Research on industrial Internet of Things security and protection technology. *Int Things Technol*, 5(3):64-66, 69 (in Chinese). <https://doi.org/10.3969/j.issn.2095-1302.2015.03.028>
- Zhang JX, Wu XL, Yang Z, et al., 2018. Research and application of industrial data acquisition based on industrial Internet of Things. *Telecommun Sci*, 34(10):124-129 (in Chinese). <https://doi.org/10.11959/j.issn.1000-0801.2018271>
- Zhou Q, 2018. GE Industrial Internet five years. *Chin Ind Inform Technol*, (7):32-38. <https://doi.org/10.19609/j.cnki.cn10-1299/f.2018.07.005>
- Zuo PL, Zhou Q, Dai X, 2019. Analysis of industrial Internet of Things technology in smart factory. *Style Sci Technol*, (8):88 (in Chinese). <https://doi.org/10.19392/j.cnki.1671-7341.201908072>