SCHEME OF IOT MIDDLEWARE BASED ON MESSAGE SERVICE

Weiqing QU[1], Chunliang ZHOU[1], Qi ZHANG[2], Jun LIU[3*]

Subscription-based message-pushing can effectively reduce the processing pressure of data servers in the Internet of Things (IOT) system. This paper describes the architecture of an IOT middleware based on message service. The middleware integrates processing chips for data acquisition and Wi-Fi modules for networking. It provides hardware and software interfaces for locale applications and web services. Both debugging mode and working mode is supplied. Using this middleware to build an IOT system, the development cost will be greatly reduced. The stress testing of a module of middleware-based lighting landscape system is described. With 90 concurrent threads, the response speed and throughput are normal in a stable network environment. At the same time, using the web service framework nginx+ tomcat + microservice + zookeeper based on EMQTTD, a distributed system can be built easily by adding servers in a flat way, and cluster services be developed.

Key words: Message Service, Middleware, IOT

1. Introduction

In the era of internet, kinds of devices are connected to the network. The Internet of Things (IOT) system based on traditional equipment came into being. In order to develop this kind of IOT system, developers must be competent at the technologies involved locale signal acquisition, processing and transmission, network programming, wifi module programming, web application design etc. That is, to master the service application environment configuration and technology involves from hardware, underlying data processing to front-end [1].

In the IOT system, the locale application and web service should be customized according to the requirements. But in the transmission process, after data acquisition and processing, each system has common points while the data transmitted the Internet through wireless routing from the locale subsystem [2]. Therefore, if an effective solution is ready for this common part, it will reduce the development cost, and the technical requirements of the development team. The

[1] A/Prof., School of Information Engineering, Ningbo University of Finance & Economics, China
[2] Eng., School of Information Engineering, Ningbo University of Finance & Economics, China
[3] A/Prof., School of Management Science and Engineering, Nanjing University of Finance and Economics, China
* Corresponding Author's Email: 852721090@qq.com

developers may focus on the solution of locale requirements and web services, and development efficiency will be greatly improved. Middleware includes basic services (functions) for application software, connect various parts of network application system, realize resource sharing. Middleware technology can provide common modules in various application system [3], which is widely used in various system [4].

General, middleware refers to software. Considering that there are a lot of "things" in IOT, a middleware covering hardware and software may be designed to meet the common needs of these systems: the middleware provides interfaces for locale applications and web services respectively, through which the processed data may be transmitted between the two sides subsystems. Efficient and reliable message transmitting mechanism can exchange data across platforms and languages. The traditional implementation mode of messaging middleware has some characteristics such as peer-to-peer, message queue, remote procedure call and shared memory [5]. Subscription-based messaging service may effectively reduce the processing pressure of data server in IOT [6].

2. Technical framework

Messaging-based IOT middleware mainly includes: message service, database service, web service, communication protocol between device and server, as shown in Fig. 1:

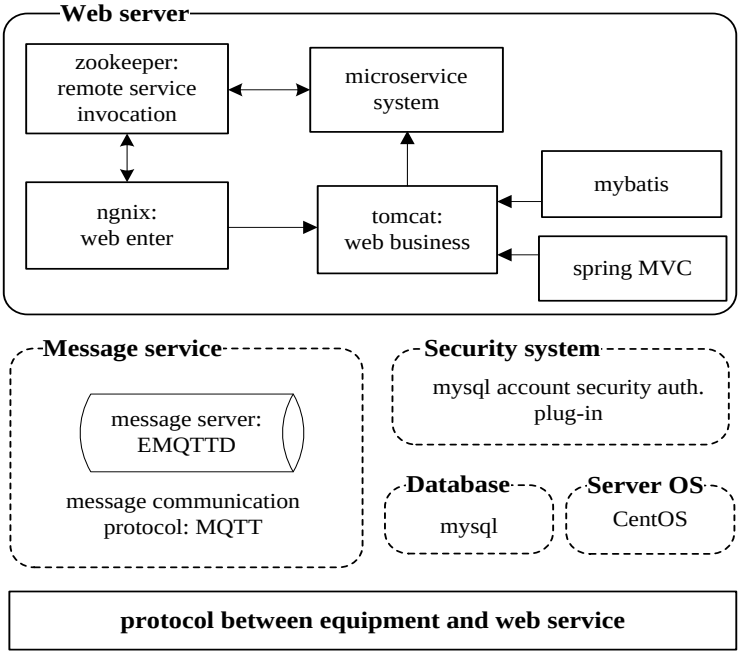

Fig. 1. Technical framework of IOT middleware

2.1 Distributed open source IOT messaging servers EMQTTD

MQTT is used as the message communication protocol. It is designed for a large number of remote sensors and control devices with limited computing power and working in low bandwidth and unreliable networks [7]. It supports all kinds of platforms and can connect almost all networked devices and networks. Compared with XMPP, MQTT is easy to implement and low less bandwidth [8]. It supports the mode of message publishing and subscribing [9]. In the locale of IOT, such as sensor-server communication and information collection, MQTT is one of good solutions [10]. EMQTTD is a MQTT server, which mainly supports publishing and subscribing functions, and supports for cluster deployment.

MQTT was originally designed for IOT [11]. It is good at communication between client and server. The data of IOT is mainly state information and control signals. Relatively, the amount of data transmitted is not very large, but the performance of real-time is required. MQTT can quickly send or receive data between devices [12].

2.2 Account security

The computational complexity of authentication process is reduced because of the lightweight security authentication protocol [13]. In the middleware, Mysql database account security authentication plug-in is used to ensure system security and maintain the unification of web server and MQTT account.

2.3 Lightweight web server: nginx + tomcat + microservice + zookeeper

In order to help developers to build IOT system quickly based on the middleware, help managers maintaining the system easily, a lightweight web server is constructed based on the frame of nginx +tomcat + microservice + zookeeper:

- Nginx: web request entry

It is a lightweight web server/reverse proxy server, which occupies less memory while having strong concurrency ability [14]. In the middleware, nginx acts as the web request entry, and performs the request accessing.

- Tomcat: web business processing unit

Tomcat server is a free, open source, and lightweight web application server [15]. The middleware adopts spring MVC as the technology framework and Mybatis as the database middleware.

The business unit accepts commands from the client, invokes message services, and pushes messages to specific devices.

- Microservice system

The server supports multi-cluster deployment. The Frame of Springboot is adopted to implement business manage and message processing functions. Business and message services are registered in the registry, and service is remotely invoked by business processing unit.

Message services include receiving commands from business processing units and sending messages to specific devices group or all devices. Subscribe the message from the device and process it accordingly, then send it to the business processing service for subsequent business processing.

- Zookeeper: configuration and server

It can provide consistent services for distributed applications, such as configuration maintenance, domain name services, distributed synchronization, and group services. In the middleware, zookeeper provides a remote service registry of Dubbo, establishes service remote call docking for tomcat and microservice system, and supports cluster deployment [16].

2.4 Server operating system: CentOS

CentOS (Community Enterprise Operating System), also known as the Community Enterprise Operating System. CentOS can build enterprise Linux system environment just like RHEL [17].

3. Functions and working mode of middleware

3.1 Functions of Middleware

An important principle of designing middleware is to provide the application system developers with simple and easy-operating interfaces as far as possible.

To improve the efficiency of development, middleware may integrate processing chips for locale data acquisition and wifi modules for networking, provide hardware and software interfaces for locale applications and software interfaces for web services. Therefore, middleware may be designed based on the following functions:

-Data is collected from the locale sensors and sent to the message server EMQTTD after processed.

-Web applications can receive the subscribed data from EMQTTD.

-In general, the locale control will be required in the IOT, that is, data should be transmitted in two-way.

-Considering that the main function of the data server is data transmission, not data processing or storage, the data server may be defined as a message server. All data may be designed as messages with certain format, and transmitted through the network.

In order to meet these functional requirements, the middleware may be

composed of processing chip, wifi module and message server. It transmits data or control signal between the middleware and locale application, and message between the middleware and web application. The architecture of the middleware-based IOT system is shown in Fig.. 2.

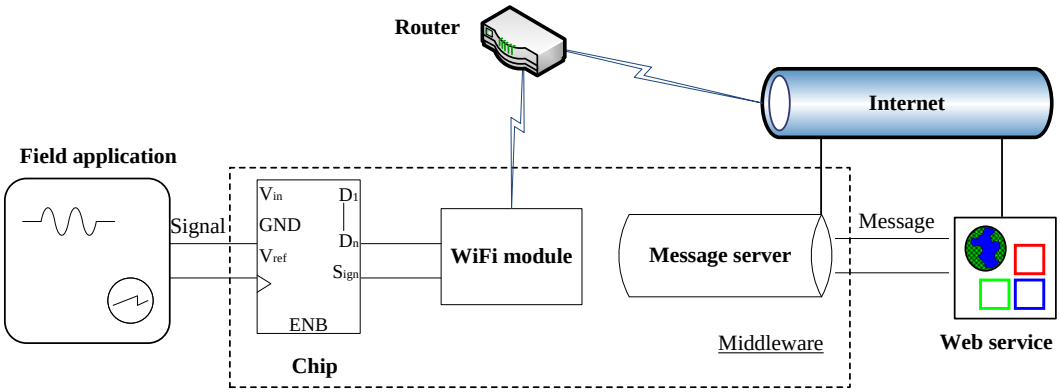

Fig. 2. Architecture of IOT based on middleware

3.2 Working mode of Middleware

In order to meet the needs of application, real-time data interaction between locale applications and web services, the middleware should be designed to ensure that programs can be written to 32U4 through PC, and ESP8266 can be connected to TCP/IP by setting or scanning routers. So, the middleware may be designed to supply debugging mode for to build an executable IOT application system:

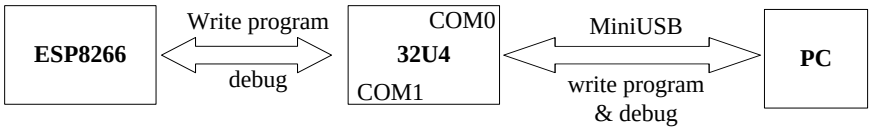

Fig. 3. Debugging model of middleware

The middleware based on 32U4 can be applied in kinds of IOT systems. The whole system includes web application, message server, sensors or controllers in the locale, and the middleware containing 32U4 and ESP8266. In the system, the chip 32U4 connects sensors and controlling mechanism, and connects with internet by ESP8266. Data or signal is transmitted between EMQTTD and web application. In Fig. 4, the flow chart shows that the operation instructions sent by web application to the locale controller, and the web application receives data from locale sensors, which is a reverse flow of data.

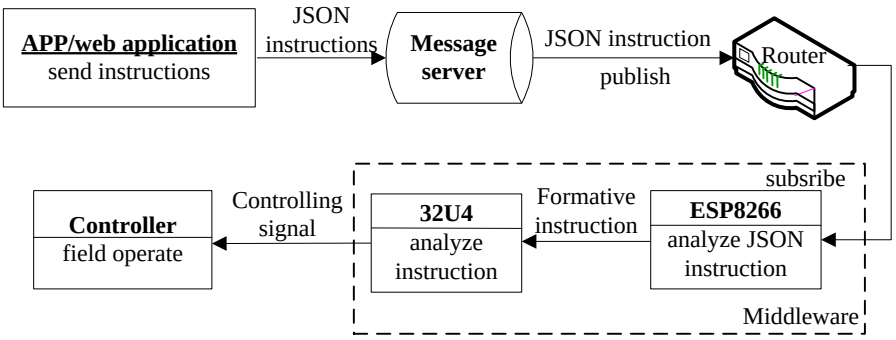

Fig. 4. Controlling model of middleware

Message pushes and subscription:

In the process of controlling, many kinds of message is required to be pushed. EMQTTD is associated with users' web application and locale control. For the web application, subscription requests and user identity are sent to the server, while the server pushes the locale status to web application according to the subscription requests, which can be timed or counted according to users' command. For application sites, the server pushes updated messages, status of web subscriptions, execution instructions to ESP8266, while ESP8266 sends device identity, locale data, and subscription requests to the server. The message transmission is shown as Fig.. 5.

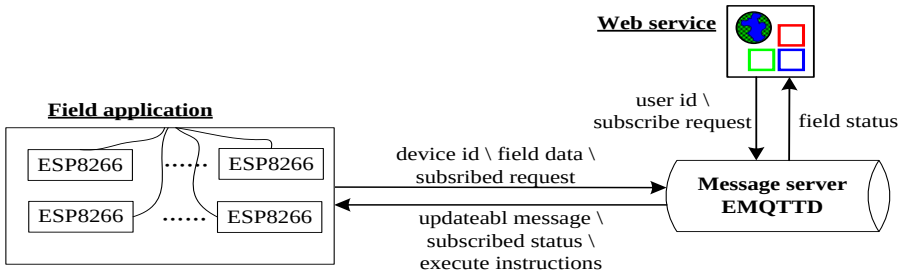

Fig. 5. Message transmission

In the process of message transmission, the chip identity reading and verification are indispensable, which is an important measure for system security.

4. Design for IOT Middleware

According to the function of middleware, the chip 32U4 of arduino is selected as the main processing chip, and the ESP8266 as the wifi module. And the two are integrated into a circuit board. So, this middleware should be designed separately in hardware and software.

4.1 Hardware

- Connecting with PC:

In order to debug the software on 32U4 conveniently, one miniUSB port is set.

- A/D expanded pin:

Set pins as many as possible, through which the signals of various sensors may be received, the processed data can be displayed in the locale monitor, and the controlling operation can be carried out according to the instructions of web service.

- Internal connection between 32U4 and ESP8266

Programming status: PC can program on ESP8266 through 32U4.

In the operation process of the application system, ESP8266 is a communication channel. The 32U4 sends the processed data to ESP8266, and ESP8266 will transmit the data, after appending its own identification number, to EMQTTD located on the internet through HTTP protocol. At the same time, ESP8266 may subscribe interested messages from EMQTTD and analyze them identifiable for 32U4.

4.2 Software

The software in middleware includes four parts:

- Software in 32U4

User's software runs in 32U4, which generally implements the acquisition and processing of sensor signals and controlling of devices linked to the ports. The processed data is sent from 32U4, and messages receives from ESP8266. In this process, the user can operate the interface for sending and receiving data, and parse the data meaning according to the predetermined format.

The functions of all pots of 32U4 should be defined clearly.

- Software in ESP8266

ESP8266 connects 32U4 at one end and internet at the other, so the messages can be transmitted between them. During the process, for the data received from 32U4, the device number of ESP8266 should be appended and sent to EMQTTD on the Internet; at the same time, the message subscribed from EMQTTD must be parsed for 32U4.

Therefore, the software in ESP8266 mainly deals with the sending and receiving of message, including the definition of messages.

- Software Interface

According to the functions defined for the middleware, software interface can be divided into two parts:

At first, software of 32U4 for locale application to initialize the chip 32U4, including its ports definition, and the functions of sending and receiving data

should be supplied.

The second, interface for web application is designed mainly for the application server, including the nodes definition of sensors and the information definition displayed on web pages.

- Definition of messages

The message types includes subscription request, control signal, status data, etc.. A complete message contains device number, message content, message value and its type. Software architecture of middleware is shown in Fig.. 6.

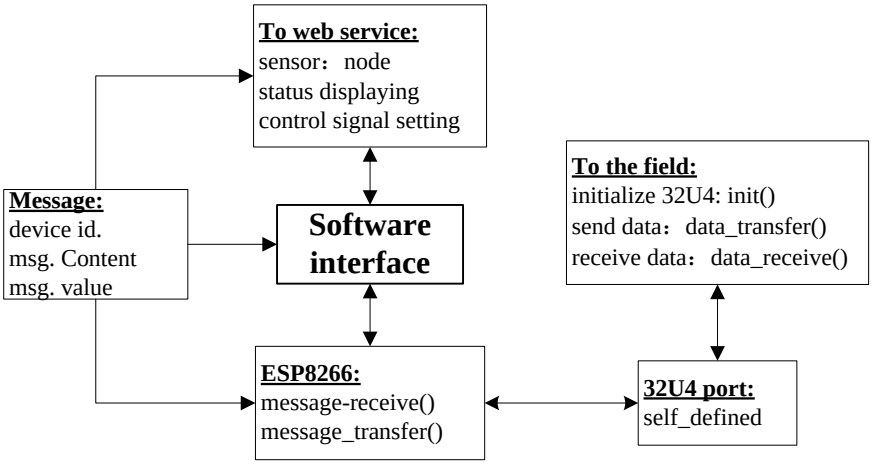

Fig. 6. Software architecture of middleware

4.3 Protocol between device and server

Protocol between devices and server is composed of 4 parts:

At first, every device must be registered in the system, to call the web service to complete the registration of the device and obtain the MQTT authorization code and the group number. The 2nd part is timed heartbeat report from device to Server. The 3rd one is status report from device to Server. And the 4th one is the subscribed messages from devices to server: three kinds of messages may be subscribed to all devices, the defined device with serial number, or group devices with group number.

5. Application of Middleware

Using the above middleware, a special IOT can be easily accomplished by simply defining the interface between hardware and software.

5.1 Hardware

According to the needs of locale applications, in principle, the relevant sensors or control lines should be connected to the predetermined pins. The

problem, how to deal with the anti-interference of the locale signal and reduce the loss of the signal, should be solved by the locale engineers.

For servers, they can be conveniently selected according to the needs of applications. Logically, they mainly include application servers, message servers and data servers. They can be physically separated or placed on the same machine according to the needs of applications too.

5.2 Software

- Design of User-Level Protocol

Software developers program on the interface function according to the set of the sensor signal and the control function of pins. In order to achieve unhindered communication between the locale and the web application, programmers must define the mean of each data and signal to ensure reading and writing correctly between both ends.

The problem of time delay in the process of data collection in wireless sensor networks can be dealt with adaptive technology similar to OFDM system [16]. It is not permitted that the delay waiting occur in the internet application, to avoid the loss of subscribed message. It should be processed according to the predefined interval time (refresh time) of receiving data from network.

After compiling, the program will connect to the network independently, send the sensor data to the message server, and do controlling operation after receive message from the message server.

- Design of web application

The functions of the web application includes two parts: one is the subscription of message; the other is the definition and display of node message. The former is accomplished by programmers with code modification in the web program, and the latter may be set directly in the web page.

Middleware provides node definitions on web pages, and the customized sensor status subscribed from message server supported by middleware.

5.3 Lighting Landscape System

For a lighting landscape system, when each RGB lamp bead can be controlled by programming, users will hope to find their favorite effects regularly through the network. Every effect file needs certain storage space and will consume amount of flow, so it should be stored on the field controller instead of being provided by the server on line. In addition, the landscape system is located outdoors generally, and there is demand for signals such as temperature, humidity, and others. Therefore, the corresponding sensors should be equipped. The middleware can be applied in the system.

When there are many users, the number of concurrent threads will rise, and the response time and throughput will be affected. Therefore, the maximum

number of concurrent threads will be limited for each server. For example, in the lighting landscape system, the maximum number of concurrent threads of will be 200. If the number of threads exceeds, they are queued.

A stress test for the effect setting module is done in Apache JMeter 5.1, and the summary is presented in table 1. The test data is set as: the number of concurrent threads is from 10 to 160, each visit is cycled 60 times to reduce the interference caused by the uncertainty of the network and obtain reasonable data.

Table 1

Stress testing summary of effect-setting module

Number of threads	Samples	Average (ms)	Min (ms)	Max (ms)	Std. Dev.	Throughput (KB/min)
10	600	55	22	150	24.1	136.2
20	1200	62	23	177	26.2	250.1
30	1800	63	24	153	22.2	371.5
40	2400	59	23	143	19.5	508.5
50	3000	73	23	163	23.3	541.1
60	3600	93	23	3156	120.3	447.7
70	4200	115	22	481	57.9	405
80	4800	132	22	3071	82.3	460
90	5400	153	6	3197	241.4	432
100	6000	136	4	3414	101.1	623
110	6600	187	3	3661	603.3	384
120	7200	181	3	4003	408	379
13	7800	235	24	3475	483	285
140	8400	235	3	3830	480	409
150	9000	501	2	9225	944	216
160	9600	345	3	9447	777	275

In the table it is shown that when the number of concurrent threads is less than 90, the average access time, standard deviation and throughput are normal. When the number exceeds 90, the data is unstable. The long response time will appear and the throughput is reduced.

The number of concurrent threads that a single server can deal with is limited. In order to solve this problem, the service architecture of nginx + tomcat + microservice + zookeeper based on EMQTTD is applied to support cluster deployment. Developers can add web servers in a flat way and realize the cluster-controlled IOT management system easily.

6. Conclusion

Owing to the standard and definable software and hardware interface of the middleware, it can be applied in the lighting landscape system, the building monitoring system, air monitoring system, etc. It will greatly shorten the development cycle of the application system and improve the development efficiency. At the same time, with the architecture of nginx + tomcat + microservice + zookeeper based on EMQTTD, the cluster deployment of IOT management system can be realized to build a large-scale IOT management platform.

Acknowledgements

This paper was supported by: Zhejiang public nonprofit technology applied research plan project (No. 2015C33236), Zhejiang university youth discipline leader academic climbing program (No. pd2013443), Ningbo natural science foud.((No. 2017A610126, No. 2012A610071), Zhejiang basic public welfare research program (LGF19G020001), Ningbo intelligent team business plan project (Ningbo World Information Technology Development Co., Ltd.), Ningbo leader and top-notch talent and Ningbo Wisdom team project, Ningbo DaHongYing College Sciences support project, Ningbo Soft Science Fund (2016A10053), 2017 Zhejiang science and technology innovation program for college students. National undergraduate innovative training program (201713001015), Ningbo science and technology benefiting people project (2017C50024).

R E F E R E N C E S

[1]. *W.H. Chen*, The Design and Implementation of a User Centric Middleware for Wireless Sensor Networks, Master Thesis, Zhejiang University of Technology, 2017.

[2]. *L. Wang, J. Jiang*, Research on Middleware for Wireless Sensor Network, Computer Engineering &Science, **vol. 36**, no. 2, Feb. 2014, pp. 244-249.

[3]. *L.S. Cui*, Overview of Middleware Technology, Science & Technology Vision, **vol. 4**, no. 3, Jan. 2014, pp. 198,288.

[4]. *Y. Mesmoudi, M. Lamnaour, Y. El Khamlichi et al.*, A Middleware based on Service Oriented Architecture for Heterogeneity Issues within the Internet of Things (MSOAH-IoT), Journal of King Saud University – Computer and Information Sciences, https://doi.org/10.1016/j.jksuci. 2018.11.011.

[5]. *J. Wang*, Application of Business Bus Based on Message Middleware in Airport Information System, Master Thesis, Xidian University, 2016.

[6]. *C.N. Wang, Z.T. Wang, Z.G. Bao, H.W. Xing*, Design of Telemetry and Command Message-oriented Middleware System with Publish/Subscribe Model, Journal of Computer Applications, **vol. 35**, no.3, March 2015, pp. 878-881.

[7]. *H. Bai*, Design and Implementation of Internet of Things Platform Based on MTQQ Protocol, Master Thesis, Xian University of Posts & Telecommunications, 2018.

[8]. *R.L. Gai, Y.L. Qian, H.B. Li, J.Y. Jia*, Enterprise Push Notification System Based on MQTT, Computer System & Applications, **vol. 24**, no. 11, Nov. 2015, pp. 69-75.
[9]. *H. Ren, Y. Ma, H.B. Yang, Z.F. Jia*, Message Pushing Server Based on the MQTT Protocol, Computer System & Applications, **vol. 23**, no. 03, Mar. 2014, pp. 77-82.
[10]. *S.T. Li, Q.S. Yin*, Application of Message Middleware in IoT Gateway, Internet of Things Technologies, **vol. 9**, no. 12, Dec. 2018, pp.48-49, 54.
[11]. *C.D. Lyu, Y.C. Li*, Lightweight Authentication Protocol for Security Vehicle Network of Railway Freight Train, Chinese Journal of Network and Information Security, **vol. 4**, no. 11, Nov. 2018, pp.23-31.
[12]. *K. Xu, Q. Ding,* An Internet of Things Communication Gateway Based on MQTT Protocol, Instrumentation Technology, **vol. 26**, no.1, Jan. 2019, pp.1-4, 43.
[13]. *D.C. Chen*, Research and Application of High-concurrent Access Server Based on Nginx, Master Thesis, University of Chinese Academy of Sciences, 2018.
[14]. *Z.Q. Qie*, High Concurrency Optimized Processing of Tomcat Application Server, Computer Programming Skills & Maintenance, **vol. 26**, no. 2, Feb. 2018, pp.129-136.
[15]. *C. Yang*, Research and Implementation of Cluster Server Based on Distributed Service Framework Dubbo, Master Thesis, Beijing University of Posts and Telecommunications, 2017.
[16]. *W.Q. Qu, Y.D. Qi, Q. Zhang, C.L. Zhou*, Multi-terminal Data Integration Analysis of Internet of Things Based on Middleware, Acta Technica CSAV, **vol. 62**, no. 2, Dec. 2017, pp. 263-272.
[17]. *L. Jiang,* Design and Implementation of Virtual Host Technology for Apache Server Based on CentOS, Journal of Changsha University, **vol. 27**, no. 5, Sept. 2013, pp. 67-68.