International Journal of Smartcare Home
Vol.1, No.1 (2021), pp.61-74
https://dx.doi.org/10.21742/26531941.1.1.06

Research on Real-time Data Transmission between IoT Gateway and Cloud Platform based on Two-way Communication Technology

David Taniar[1], Johan Barthelemy[2], and Lin Cheng[3]
[1,2,3]Department of Electrical and Computer Engineering, McGill University, Montreal, Canada
[1]david.taniar@mail.mcgill.ca

Abstract

To realize the real-time two-way communication between the IoT gateway and the cloud platform and improve the transmission efficiency, this paper adopts the designed open IoT platform, uses the Node.js server as the operating platform, and uses the Redis database fast access technology. The design is based on Modularized processing of real-time two-way communication system of Internet of Things gateway and resource platform based on Socket.IO communication protocol. Compared with the traditional Internet of Things communication protocol, the advantage of Socket.IO is that only a handshake between the client and the server is needed to quickly establish a Socket two-way channel, which can realize the real-time two-way transmission of data and effectively save broadband resources. Thereby improving the system's real-time update efficiency of the data model, and ensuring the stability and reliability of the system. For the system function and performance test, the experimental results show that the use of Socket.IO two-way communication technology can realize the simultaneous update of the physical model of the IoT gateway and the virtual model of the IoT cloud platform, and effectively improve the data transmission efficiency of the system.

Keywords: *Two-way communication technology, IoT gateway, Cloud platform, Data model*

1. Introduction

Based on information carriers such as the Internet and traditional telecommunications networks, the network that enables all ordinary physical objects that can be independently addressed to achieve interconnection is the Internet of things (IoT) [1][2]. Literature [3] proposes the role of gateways in IoT systems. In the choice of the communication protocol of the Internet of Things, the HTTP protocol uses the "request-response" method to achieve communication. The advantage is that it is convenient, agile, and easy to expand. The disadvantage is that it cannot realize the real-time two-way transmission of data. The MQTT mentioned in the literature [5] is a "lightweight" communication protocol based on the publish/subscribe model built on TCP/IP. It has the advantages of open source, simplicity, and easy implementation. [6], the disadvantage is high power consumption and lack of encryption mechanism.

Aiming at the problem of data transmission in the Internet of Things system, Node.js is used as the operating platform, using Socket.IO communication mechanism and Redis database cache technology [7] to achieve a two-way communication system for real-time data

Article history:
Received (February 14, 2021), Review Result (March 15, 2020), Accepted (June 10, 2021)

transmission between the gateway and the cloud platform. The gateway is divided into functional modules to realize the interaction process between the gateway and the platform regarding the authentication of the underlying device, the data transmission of the device, and the heartbeat of the device [8].

2. Internet of things open platform and gateway

From top to bottom, the Internet of Things system is the web front-end interface, the Internet of Things cloud platform, the Internet of Things gateway, and the sensor device. By constructing the IoT system architecture, cloud platform, and gateway, real-time two-way communication, and transmission of the IoT system are realized.

2.1. IoT system architecture

To facilitate the safe, stable, and efficient transmission and storage of data from the underlying sensor equipment, the Internet of Things system architecture is established, as shown in [Figure 1].

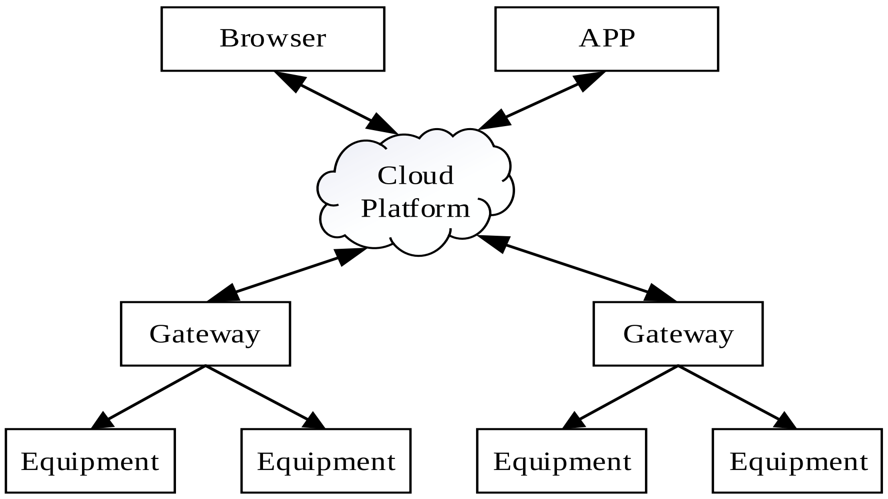

Figure 1. Internet of things system architecture

The IoT system architecture is mainly composed of the underlying equipment, the IoT gateway, the IoT cloud platform, and the browser. The Internet of Things system uses an open device system based on a platform + gateway to complete the access, processing, and transmission of the underlying sensor device data, and finally, display it to PC users or mobile users in the form of an interface through the front end.

2.2. IoT cloud platform

The cloud platform adopts a unified resource description and resource-oriented architecture (ROA) and uses asynchronous non-blocking characteristics, which has obvious advantages in a long connection and multi-request environment [9]. Use Node.js language's high-efficiency data event engine and high-concurrency processing mechanism characteristics [10] to complete the design and implementation of the cloud platform. The architecture of the IoT cloud platform is shown in [Figure 2].

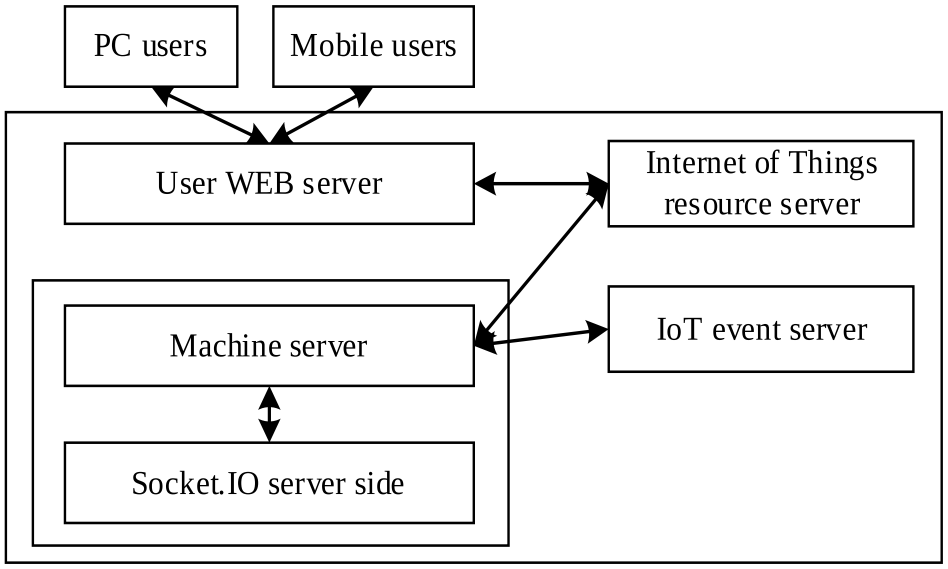

Figure 2. IoT cloud platform architecture

To complete the real-time transmission of the underlying device data and have a more efficient processing mechanism, it solves the shortcomings that the HTTP protocol does not have real-time performance and the MQTT protocol can only run in hardware devices with a limited processor and memory resources. Therefore, this article uses a real-time two-way transmission mechanism based on the Socket.IO communication protocol to divide the cloud platform into modules, which can complete the processing and application of the underlying device data more quickly and efficiently.

The cloud platform is divided into four parts: machine server, resource server, event server, and user server. Among them, the user web server is used to realize the interaction between the platform and the website, and the IoT resource server is used to complete the addition, deletion, modification, and inspection of platform data, platform event detection, and event operation task distribution, and the IoT event server is used to maintain events and operation processing Queue, process events in real-time, and complete the real-time two-way transmission of data between the platform and the gateway by building a Socket.IO server in the machine server. After the device is successfully connected, the device data is analyzed and processed at the gateway, and the device data is transmitted to the cloud platform through the IoT gateway to complete the data transmission process between the device and the platform, making the platform's operation of the device easier and convenient, the workload can be reduced, and the work efficiency is higher.

2.3. IoT gateway

The IoT gateway serves as a bridge connecting the cloud platform and the underlying equipment and completes the management and application of authentication, data upload, and data update of the underlying equipment. To realize the conversion, management, and extensive access of the gateway to the device protocol, we have designed a hierarchical architecture and system module for the gateway.

In the gateway layered architecture, it can be divided into two parts: the internal gateway module composed of the platform communication layer and the data model management synchronization layer, and the device access module composed of the protocol adaptation layer and the sensing interface layer. The two modules use TCP for communication to realize the access authentication of the equipment, the management of the equipment, and the protocol conversion of the equipment data. The hierarchical architecture of the gateway is shown in [Figure 3].

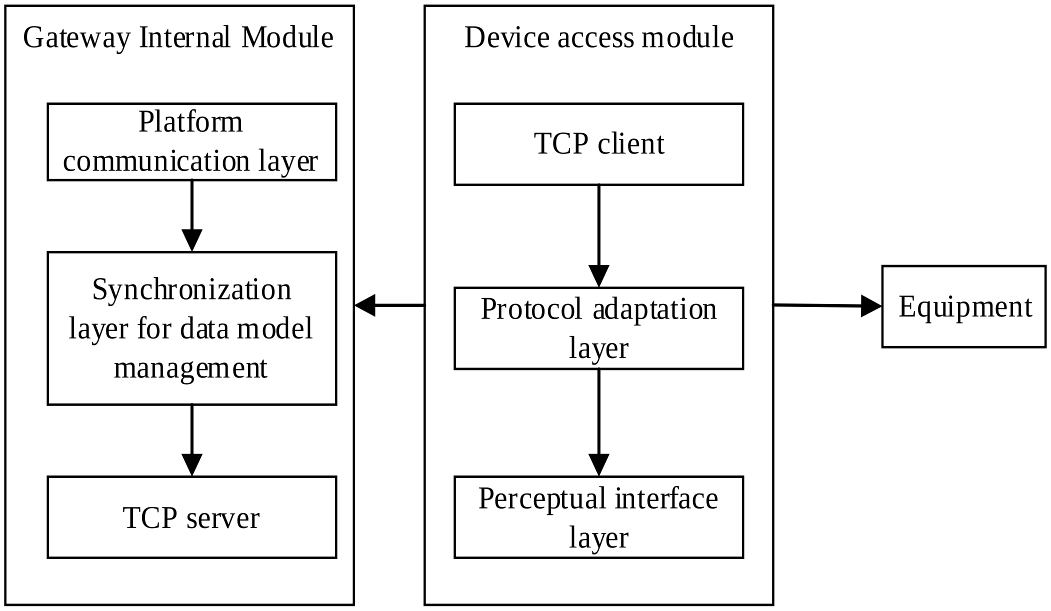
Gateway Internal Module
Platform communication layer
Synchronization layer for data model management
TCP server
Device access module
TCP client
Protocol adaptation layer
Perceptual interface layer
Equipment

Figure 3. Gateway layered architecture

In [Figure 3], the IoT gateway layered architecture is mainly composed of a perception interface layer, a protocol adaptation layer, a data model synchronization and processing layer, and a platform communication layer. Among them, the realization of the two-way communication system of the Internet of Things platform is completed by constructing the Socket.IO client in the platform communication layer and the server in the machine server. The data model synchronization and processing layer are used to implement the process of platform data distribution and device data upload and realize the synchronization update of the platform model and the gateway model. The protocol adaptation layer completes the data conversion between the device data model and the actual device and realizes the access of the IoT platform to various devices. The perceptual interface layer implements the device's access driver, and for most general-purpose devices, it provides a general-purpose adaptation program. Develop programs to connect non-standard equipment.

`underlying sensor equipment data, accelerate the event processing speed on the original basis, reduce the pressure on the cloud platform, and realize the application and management of the underlying sensor equipment by the IoT gateway. The gateway system module is shown as in [Figure 4].

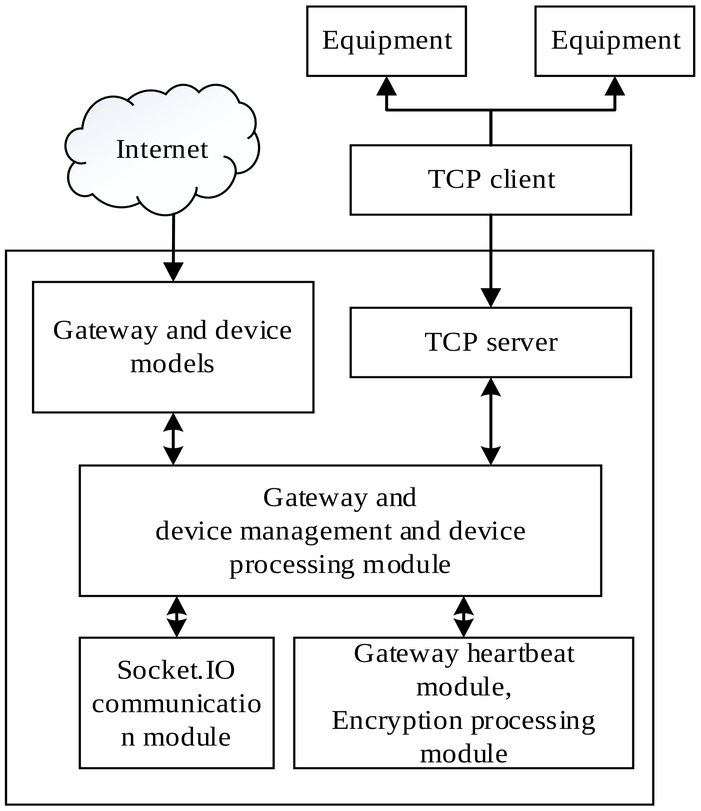
Equipment
Equipment
Internet
TCP client
Gateway and device models
TCP server
Gateway and device management and device processing module
Socket.IO communication module
Gateway heartbeat module, Encryption processing module

Figure 4. The gateway system module

In [Figure 4], the gateway system module is composed of 5 parts: Socket.IO communication module, gateway and device model module, gateway and device management and event processing module, device communication module, gateway heartbeat, and encryption processing module. The function of the Socket.IO communication module is to realize the real-time two-way communication between the IoT platform and the gateway and complete a series of operations such as device access. The function of the gateway and device model module is to automatically communicate with the platform after the gateway is powered on, create a physical model of the gateway and device according to the platform registration data model, and maintain real-time synchronization and update with the platform. The function of the gateway and device management and event processing module is to complete the state management of the gateway and the device, the configuration management of the gateway, and the location and operation management of the device, and to realize the processing of the event processing module and event monitoring module of the gateway device model. The function of the device communication module is to realize the access of the underlying device through TCP communication and complete the conversion of sensor device data and physical gateway device model data. The function of the gateway heartbeat management module is to periodically upload heartbeat messages to the machine server, otherwise, the machine server thinks that the gateway is down and uploads the status to the platform database. The function of the encryption processing module is to use the MD5 encryption method to authenticate and authorize the gateway for the userKey and gateway ID allocated during gateway access authentication.

3. Gateway platform two-way communication system

Through the construction of a machine server, a two-way communication system for the Internet of Things, and a sub-module design for the Internet of Things gateway, the two-way communication, and transmission of the Internet of Things system is realized.

3.1. Machine server design

Build a machine server inside the IoT resource platform to implement gateway device access, identity verification, and model synchronization. The machine server module is shown in [Figure 5].

In [Figure 5], the machine server module is mainly composed of the Redis message queue module, message processing module, interface module, and Socket.IO server module.

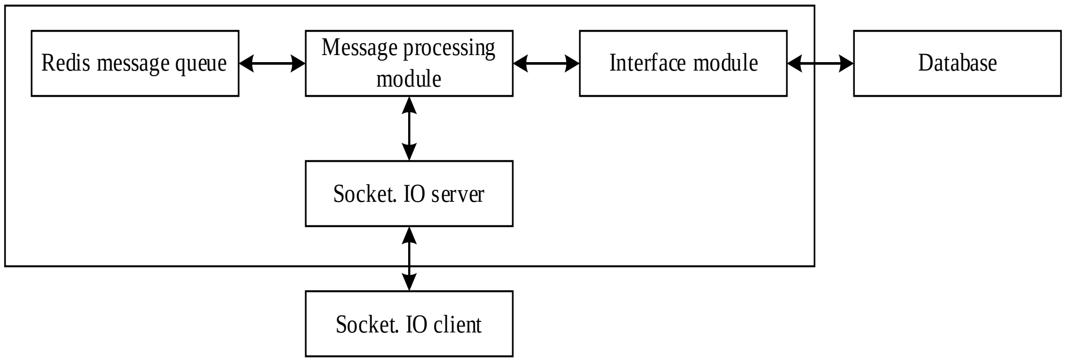

Figure 5. Machine server module

The Redis message queue can cache data to solve the high concurrency of data. The Socket.IO server and the Socket.IO client conduct real-time two-way data interaction. Message processing module to realize the analysis, processing, and distribution of messages. The interface module is used to realize data interaction and cache reading.

By establishing the Socket.IO server in the IoT platform machine server and the Socket.IO client in the gateway, the two-way real-time communication between the platform and the gateway is completed, and the authentication of the gateway device, the upload and update of the gateway device data, and the gateway/ Heartbeat transmission of the device. After the authentication access of the gateway device is completed, the registered model is read from the Internet of Things resource platform, and the synchronous update of the gateway device model is completed.

3.2. Two-way communication system design

Socket.IO is a real-time communication module based on the WebSocket protocol under the Node.js platform. It is composed of JavaScript on the client-side and Node.js on the server-side [11]. The difference between its communication method and the traditional HTTP protocol is that only one handshake process is required between the client and the server, and then a fast Socket two-way channel is established, which in principle effectively saves the server's broadband resources and improves System transmission efficiency.

In the Internet of Things system, the data transmission between the Internet of Things gateways and the cloud platform is completed based on the Socket.IO two-way communication protocol, including the access authentication of the underlying sensor equipment, data, and heartbeat message transmission. The equipment authentication process mainly realizes the secure access of the underlying equipment, guarantees the security of the system, and returns the corresponding data model to the gateway side. The data transmission process not only realizes the effective collection of equipment data and stores it in the system database but also completes the synchronization update and modification of the data model according to the needs of the platform. Finally, you need to upload device heartbeat messages regularly to ensure that the device stays online and sends it to the cloud platform at the same time.

3.3. Gateway modular design

According to different requirements, the functional module division of the gateway is completed, and the divided modules are packaged, managed, and developed, and there is no coupling between the modules, which is convenient for subsequent software programming. The modular design of the gateway is shown in [Figure 6].

As shown in [Figure 6], for this system, the gateway mainly completes three parts of operations: One is the gateway device authentication module. When an underlying device is connected, the gateway sends an authentication request to the platform, and the authentication is successful, and the physical gateway data model is returned. The second is the data upload and instruction issuance module. The underlying sensor equipment can upload data to the platform in real-time and store it in the system's database. At the same time, the synchronization of the platform simulation data type and the device data type is completed, and the platform can deliver the updated data model to the gateway. The third is the heartbeat management module of the gateway device. It regularly sends heartbeat messages of the gateway and device to the machine server. If it is disconnected for a long time, it will be considered to be in the shutdown state. At the same time, the state needs to be uploaded to the database to complete the heartbeat management process of the gateway and device.

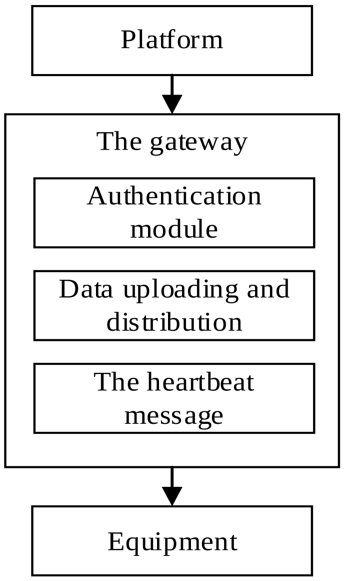

Figure 6. Gateway modular design

When the device is connected for the first time, it needs to be authenticated twice to complete the authority authentication management of the system to the underlying device. Its purpose is to ensure reasonable equipment access and the safety of the entire system. The authentication process is shown in [Figure 7].

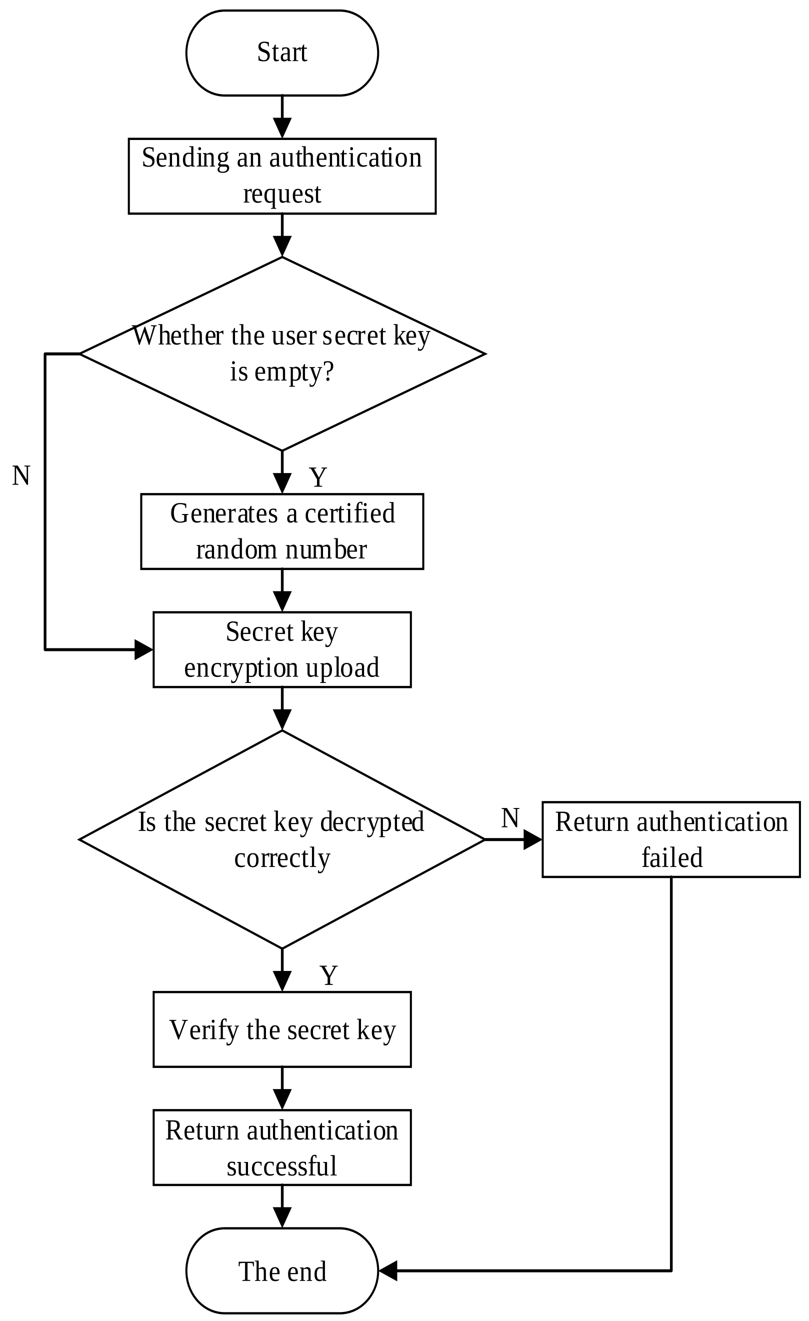
Start
Sending an authentication request
Whether the user secret key is empty?
N
Y
Generates a certified random number
Secret key encryption upload
Is the secret key decrypted correctly
N
Return authentication failed
Y
Verify the secret key
Return authentication successful
The end

Figure 7. Device registration and authentication process

In [Figure 7], when the underlying sensor device is connected and registered, the gateway side needs to send an authentication request packet to the platform side, and the platform side listens to the data sent from the gateway side by establishing a port. When registering for authentication for the first time, the gateway sends a data packet without userKey to the platform. The platform receives the data packet and parses it to determine whether the userKey in the data packet is empty. If it is empty, an authentication random number is generated and stored in the Redis database. In, the gateway userKey and the generated authentication random number are encrypted through MD5, and the encrypted user-Key is obtained. Otherwise, the generated authentication random number is read from the database according to the gateway ID, and the encryption process is completed according to the above encryption method. Decrypt the above-mentioned encrypted content. If the decrypted userKey is consistent with the userKey in the secondary authentication request, the authentication is successful, and the authentication success identifier and data model are returned to the gateway; otherwise, the authentication failure is returned and the process ends.

The device needs to establish a connection with the platform to complete the real-time transmission of device data and to synchronize the update of the platform and the device data model after the device is successfully authenticated. The bottom device data is transmitted to the gateway via the TCP protocol to realize the synchronization update process of the data model. Whenever the virtual data model registered by the platform changes, the corresponding physical data model must also be changed to complete the synchronization conversion between the real-time data and status update data of the equipment and the defined data model. The data upload process is shown in [Figure 8].

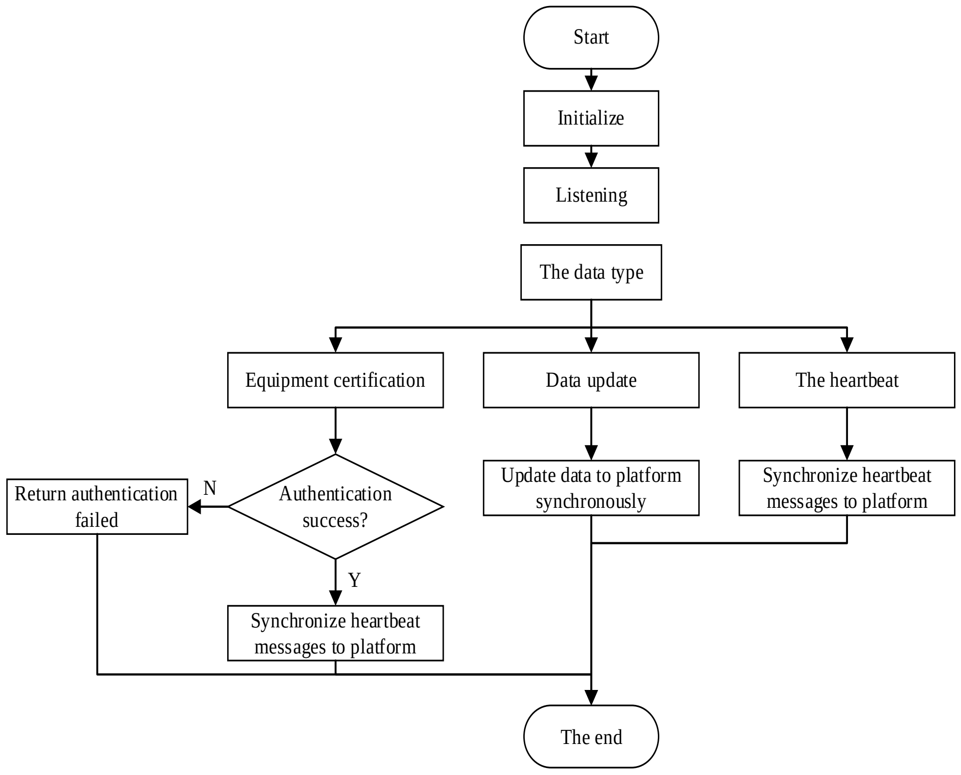
Start
Initialize
Listening
The data type
Equipment certification
Data update
The heartbeat
Return authentication failed
N
Authentication success?
Update data to platform synchronously
Synchronize heartbeat messages to platform
Y
Synchronize heartbeat messages to platform
The end

Figure 8. Device data upload process

In [Figure 8], before uploading device data, first, complete the process initialization work. The gateway client monitors the data uploaded from the underlying device, and judges and processes the data type. If the monitored message is a device authentication message, the device authentication is completed through the device authentication process, and the gateway device model in the platform is returned. If the device data update message is monitored, the data needs to be updated to the platform side through the data upload process to complete the real-time update process of the underlying device data. If the heartbeat message of the device is monitored, the heartbeat messages of the gateway and device need to be uploaded to the machine server regularly to determine whether the two are shut down, and upload the gateway and device status to the platform database.

Construct a JSON format device control instruction on the platform side, and send the instruction message to the specific device through the gateway to realize the operation of the underlying device. The operation flow is shown in [Figure 9].

In [Figure 9], when the platform needs to operate the device, it is initialized, the JSON message data sent by the platform is monitored, the type of the received message is judged and corresponding processing and the platform is issued to it through the Socket.IO protocol Gateway. If the monitored message is an instruction message, it will be sent to the specific underlying sensor device through the gateway to perform specific operations on it. If the monitored message is a model message, the platform will return the corresponding device model and gateway model to the gateway when the authentication is completed. At this point, the platform data distribution process is over.

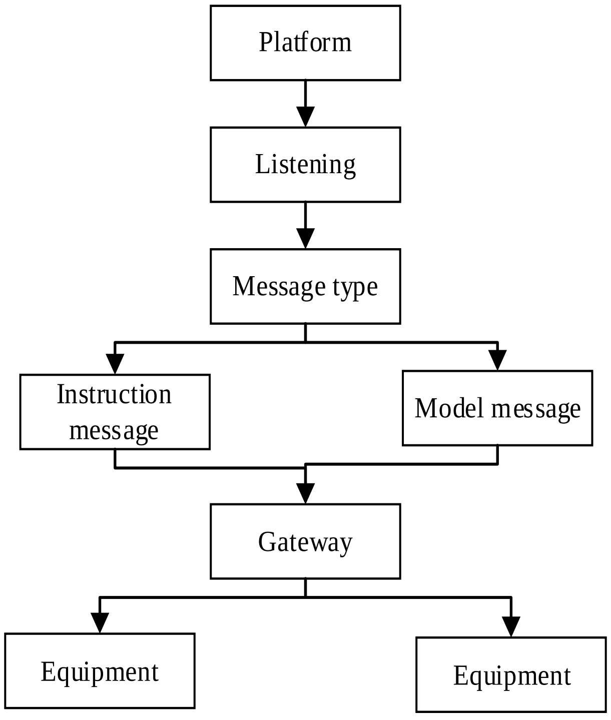

Figure 9. Data issuance

After the system is connected, set the gateway heartbeat interval to 30s, which mainly includes two functions: one is to verify whether the connection between the device, the gateway, and the platform is maintained. The second is to ensure that the gateway stays online, complete the orderly transmission of data, and upload the heartbeat message as shown in [Figure 10].

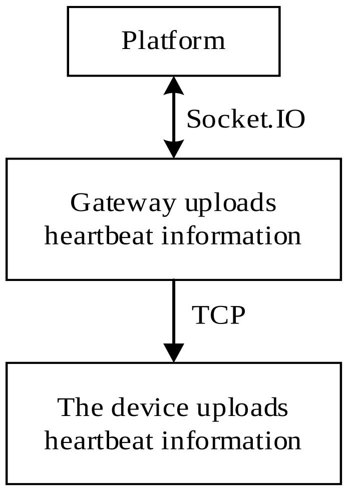
Platform
Socket.IO
Gateway uploads heartbeat information
TCP
The device uploads heartbeat information

Figure 10. Device heartbeat upload

In [Figure 10], the device needs to periodically send corresponding heartbeat messages to the gateway through the TCP communication protocol to determine whether the device maintains contact with the gateway. At the same time, the gateway is required to periodically send the heartbeat message of the device to the platform through the Socket.IO communication protocol to determine the connection between the device and the platform and upload the real-time status of the device to the platform database.

3.4. Modularization of a real-time two-way communication system of gateway platform

The Socket.IO two-way communication mechanism and the data caching function of the Redis database are used to complete the realization of the modular design of the real-time two-way communication of the IoT gateway.

Socket.IO implements a real-time and two-way communication mechanism. It solves the real-time communication problem and unifies the programming method of the server and the client. After starting the socket, it is like establishing a client and server pipeline, and the two sides can communicate with each other. With or without [12]. The connection process includes: the server starts a socket service and monitors the ‘connection’ event. The client creates a Websocket, connects to the socket on the server-side, and binds methods to receive socket events. After the client is successfully connected, the server socket can send messages to the client. Socket communication can be simply understood as a long link in the true sense. If the channel is not actively disconnected, the channel will always exist, and the two ends of the channel can talk to each other.

The Socket.IO mechanism based on the WebSocket protocol is used for two-way communication between the IoT gateway and the cloud platform. The Socket.IO message mechanism is shown in [Figure 11].

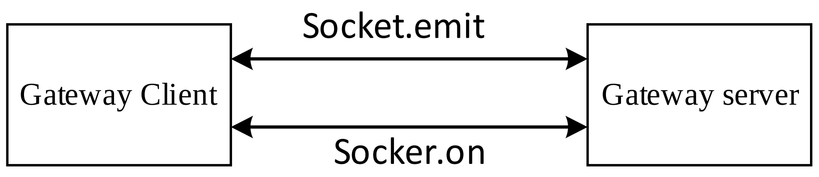

Figure 11. Socket.IO message mechanism diagram

In [Figure 11], the data transmission between the IoT gateway and the cloud platform is realized through the Socket.IO two-way communication message mechanism. The Socket.IO two-way communication process is: the gateway server is successfully created, and then the corresponding port number and IP parameters are set, and the gateway client completes the connection with the server according to the IP parameters and port number set for access. When the underlying sensor device needs to be authenticated, data update, and gateway/device timing heartbeat upload, the data is transmitted by the socket.meit ("message name", JSON format message content) through the Socket.IO message mechanism in Figure 11 To the resource platform. At the same time, the server can also push data to the client and monitor whether the data is successfully received through the socket.on ("message name", JSON format message content) to complete the two-way communication between the IoT gateway and the resource platform.

Redis is a NoSQL database based on key-value, which is widely used in distributed applications and is generally used to store cached data [13]. Through consistent hashing of the key, the distribution of the key corresponding to the Redis node is realized. Redis supports asynchronously writing the data in the memory to the hard disk without affecting the continued service. To ensure a certain operating efficiency, its data is temporarily stored in memory, so it supports high-speed operations on complex data structures. In the cloud platform, the Socket.IO server is connected to the database Redis to complete the cache storage of device data. When there are many data types, large amounts of data, or high concurrency, the data is temporarily stored in the Redis cache queue to improve the efficiency of the system and reduce memory usage. The cached data includes the authentication random number generated by the server during the first authentication, the Socket.ID number of each successfully connected and online gateway, the gateway/device status, and the timeout time of the gateway/device heartbeat.

4. Test

Complete functional test and performance test of the whole system to verify the feasibility of the system. The hardware equipment includes a temperature and humidity sensor/camera (DM365), a Sugon server, and a PC. The performance test uses Jmeter software to test the related parameters of the entire two-way communication system.

4.1. Function test

Design related test cases to perform specific tests on each module of IoT gateway communication. The specific functions of the test include device authentication, device data upload and status update, platform instruction data delivery, and platform model data delivery, gateway/device heartbeat.

(1) Equipment certification

When there is a bottom-level device that needs to be connected to the IoT platform, the bottom-level device needs to be authenticated. After the authentication is successful, the platform returns to the gateway/device data model.

(2) Image data upload

When the bottom device is a camera, the collected pictures need to be sent to the platform and displayed through the front-end interface for system functional testing.

4.2. Performance test

To ensure the effective operation of the system, relevant performance tests are carried out on the system to verify the feasibility of the system, as shown in [Table 1].

Table 1. Comparison of aggregation reports

Letter of agreement	Socket.IO protocol	HTTP protocol
Number of samples	3000	3000
Average response time/ms	273	212
90% user response time/ms	603	544
Error rate/%	0.17	0.21
Packet loss rate/%	0.015	0.03
Transmission rate/(KB/s)	8.23	8.94

In [Table 1], select 3000 samples, and use Jmeter software to perform performance testing under the two methods of the HTTP protocol and Socket.IO protocol. The performance test report of the entire system is mainly composed of the adopted communication protocol, the number of samples, the average response time, the probability of error, the probability of packet loss, and the transmission rate. It can be seen from the experimental results that, in contrast, when the Socket.IO protocol is adopted, the data transmission efficiency is significantly improved, the packet loss rate is significantly reduced, and the working efficiency of the entire system is greatly improved. It can be seen from the test results that the use of Socket.IO two-way communication protocol as the two-way communication protocol between the IoT gateway and the platform can not only realize the real-time data transmission between the gateway and the platform but also increase the operating speed of the entire system and ensure the system security and stability.

5. Conclusion

Based on the Node.JS operating platform, the communication part adopts the Socket.IO communication protocol, and the system database uses Redis to complete the cache access function for the underlying sensor device data, completing the modularization of the real-time two-way communication system between the gateway and the platform design and implementation. First, complete the framework of the IoT communication system through the design of the IoT system, cloud platform, and gateway architecture. Secondly, according to different functional requirements, for the communication system to complete the design of different modules to achieve different functions, use the Socket.IO two-way communication protocol to achieve real-time authentication of the underlying device, the underlying device data, and the heartbeat data between the IoT gateway and the cloud platform Transmission process. Finally, the acceptance of the system is completed by performing functional and performance tests. From the above test results, it can be known that the two-way communication system between the gateway and the platform based on the Socket.IO communication protocol can more effectively improve the working efficiency of the system and the stability of the system during operation compared to the traditional protocol.

References

[1] J. Gubbi and Buyya, "Internet of Things (IoT): A vision, architectural elements, and future directions, Future Generations Computer System, **(2013)**

[2] I. Lee and K. Lee, "The internet of things (IoT): Applications, investments, and challenges for enterprises," Business Horizons, vol.58, no.4, pp.431-440, **(2015)**

[3] T. M. Georgescu, B. Iancu, and M. Zurini, "Named-entity-recognition-based automated system for diagnosing cybersecurity situations in IoT networks," Sensors (Basel, Switzerland), vol.19, no.15, **(2019)**

[4] C. S. Cho, Y. R. Oh, and Y. H. Lee, "Design and implementation of a two-way real-time communication system for audio over CATV networks," International Society for Optics and Photonics, **(2007)**

[5] D. Thangavel, X. Ma, A. C. Valera, "Performance evaluation of MQTT and CoAP via a common middleware," IEEE Ninth International Conference on Intelligent Sensors, IEEE, **(2014)**

[6] R. A. Light, "Mosquitto: Server and client implementation of the MQTT protocol," The Journal of Open-Source Software, vol.2, no.13, **(2017)**

[7] M. Kusuma, Widyawan, and R. Ferdiana, "Performance comparison of caching strategy on wordpress multisite," International Conference on Science and Technology-computer, IEEE, **(2017)**

[8] A. Syaefulloh and Y. F. Implementasi, "Dan Analisa Performa DataBase Cache Redis," **(2019)**

[9] R. H. Sarnol, "Developing distributed system with service resource-oriented architecture," Telkomnika, vol.10, no.2, **(2012)**

[10] J. Chaves and S. Freitas, "A systematic literature review for service-oriented architecture and agile development," Springer, Cham, **(2019)**

[11] A. Rahmatulloh, I. Darmawan, and R. Gunawan, "Performance analysis of data transmission on web socket for real-time communication," 2019 16th International Conference on Quality in Research (QIR): International Symposium on Electrical and Computer Engineering, **(2019)**

[12] F. U. Damayanti, "Research of web real-time communication - The unified communication platform using node.js signaling server," Journal of Applied Information Communication and Technology, vol.5, no.2, pp.63-72, **(2018)**