ATLANTIS PRESS

Advances in Computer Science Research, volume 87

3rd International Conference on Mechatronics Engineering and Information Technology (ICMEIT 2019)

Design of IoT Web Server Communication Platform based on Netty and WebSocket

Xiongfei Liu[1, a, *], Jiakang Liu[1], Beiping Liao[2], Yunyi Zhu[1] and Huimin Liu[1]

[1]School of Physics and Electronics, Central South University, Changsha 410083, China.

[2]Hunan Hengmao High-Tech Co., Ltd., Liling, Hunan 412200, China.

[a, *] x.f.liu@163.com

Abstract. With the development of the Internet, in the home, medical, transportation and other aspects of the Internet of Things large-scale application, more and more IoT gateways are connected to the cloud. Users interact with the Internet of Things gateway through the Internet of Things web server. The traditional Internet of Things web server communication platform has the disadvantages of large system resource consumption, long system response time and poor real-time information transmission. This paper provides a design of Internet of Things web server communication platform based on Netty and websocket, which realizes high-performance communication between the Internet of Things gateway, web client and server, and real-time information between web client and gateway client, transmission.

Keywords: Internet of Things Web Server; Netty Framework; Protobuf Framework; Websocket Protocol; Internet of Things Gateway.

1. Preface

With the rapid development of the Internet of Things, IoT devices are widely used. Currently, IoT servers in the industry use traditional input/output (IO) based Transmission Control Protocol (TCP) communication and asynchronous Ajax polling to push information. In the process of interaction between a large number of web clients and the Internet of Things gateway and the Internet of Things web server, there are mainly problems such as the number of concurrent connections of the Internet of Things gateway, the performance of the Internet of Things web server system [1], and the lack of timely feedback of web client messages. Based on the Netty framework and WebSocke protocol, the Internet of Things web server communication platform uses Netty's high performance, event-driven, asynchronous non-blocking and other performance to build high-performance communication between the IoT gateway and the Web server, and communicates using the long-connected Websocket protocol [2]. , complete the connection between the web client and the Internet of Things web server. Realize the high-quality stable communication between the Internet of Things web server and a large number of IoT gateways and the real-time information interaction of the web client.

2. The Overall Design of the IOT Web Server Communication Platform

The block diagram of the IoT web server communication platform system is shown in Figure 1: The Netty framework and the Websock protocol are used in the SSM framework of the built Internet of Things web server. The web server performs IP port monitoring and data processing on the IoT gateway client, and realizes high-quality stable communication between the IoT Web server and the large number of IoT gateway clients through the Netty framework, and stores the received data in the web server database. The web client establishes a Websocket connection with the web server. The Netty server on the Internet of Things web server implements the command control of the IoT web client push and the web client to the IoT gateway client through the web client SessionID and the Websocket protocol. Achieving Real-time interaction of information between the web client and the IoT gateway client [3].

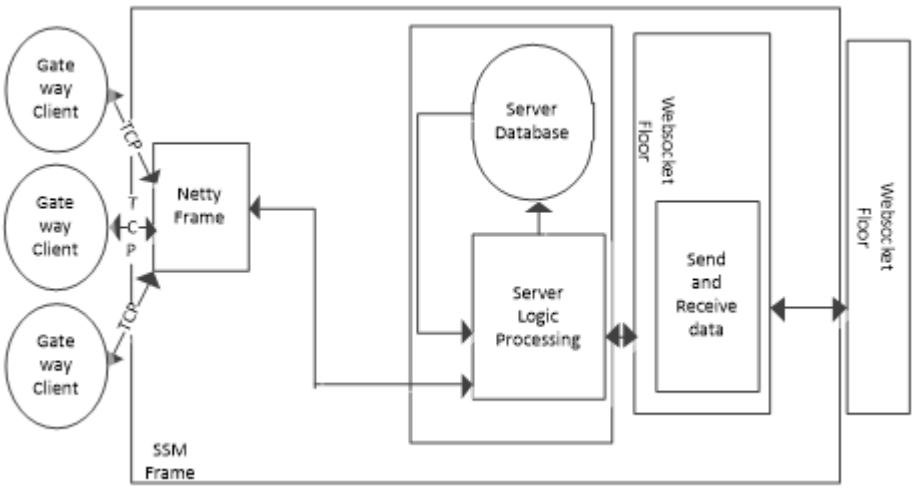

Figure 1. System block diagram

3. Web Server Netty Framework and Websocket Protocol Design

3.1 Web Server Communication Platform Netty Framework Design

Netty work flow chart shown in Figure 2: The IoT gateway client connects to the IoT web server via Tcp. The Netty server is built on the web server. After the boss thread pool listens to the IoT gateway client port connection, it binds a socketchannel channel to the worker thread pool. The IoT gateway client sends information to the worker thread pool. After processing by the channelhander class in the work thread pool, the data sent by the IoT gateway client contains the User_ID, and the User_ID is used as the keyword query table. The corresponding sessionID in the User_Index table determines the corresponding web client through the sessionID, and pushes the client information from the IoT gateway and stores the data in the database for persistence processing [4]. At the same time, from the web server database through the User_ID query table User_Order corresponding command information, read and sent back to the IoT gateway client.

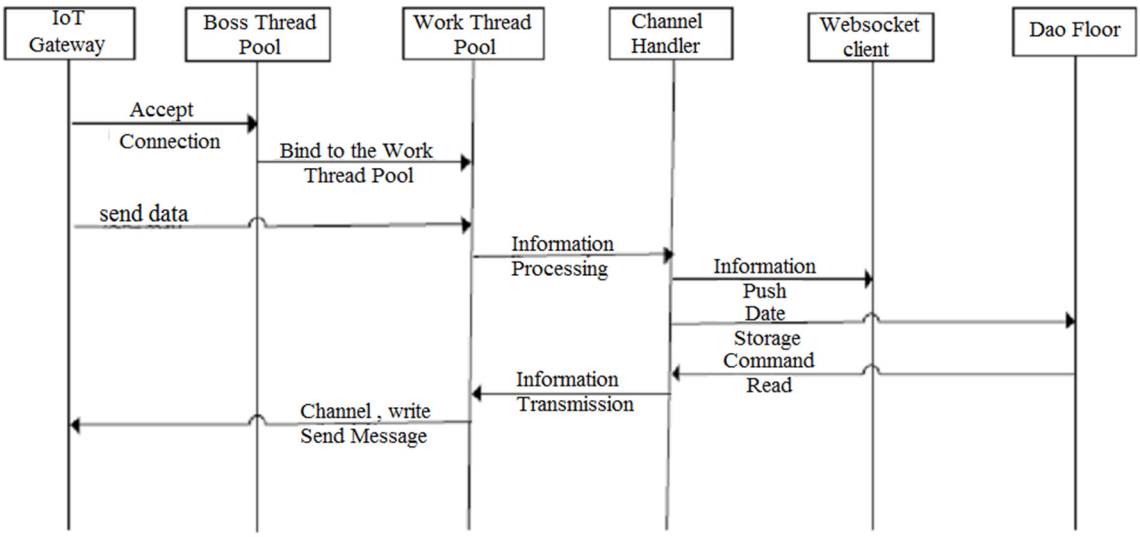

Figure 2. Netty work flow chart

The web server processes the transmitted information through the channelHandler. ChannelHandler is a key interface that Netty provides to developers for customization and extension [5]. ChannelHandler provides a variety of methods to support different operations at different stages of message reading. The ChannelPipeline in this paper mainly adds a decoder to decode the request

message in the server channel. An encoder implements encoding of response messages within the server channel. A logical handle handles the encoding and decoding and logical processing of data.

3.1.1 Message Codec

In the Internet of Things web server communication platform, a large number of gateway clients communicate with the intelligent web server through the Tcp protocol. On network transmission, transmission data serialization tends to increase the transmission rate of the network [6], and is also easy for machine parsing and generation. Xml and json serialization have become the preferred protocol for most communication systems due to their platform independence and small memory footprint, but they increase the space overhead for good readability. When the number of IoT gateway clients is relatively large, a large amount of web server memory resources will be consumed, which affects the normal operation of the web server. Netty provides a powerful codec framework and supports the decoding and encapsulation of protobuf by default. Protobuf is a platform-independent, language-independent structured codec tool. Compared to xml and json, its serialization and deserialization processing time is shorter, and the serialized code stream is smaller, which is more conducive to network transmission and persistence. . By introducing the netty jar package, the Netty framework's Channel Pipeline adds the class Protobuf Varint 32 Frame Decoder to solve the half-packet and sticky-package problems before decoding. Adding the decoding class Protobuf Decoder decoding converts the specified gateway client to the web server's UserInform byte number group into UserInform class. Add the encoding class ProtobufVarint32engthFiel depender to the encoded byte array with a simple header to indicate that the encoded byte length supports half-packet and sticky-packet processing during decoding. Add the encoding class ProtobufEncoder to encode the UserOrder class that will be passed from the web server to the gateway client command information.

The UserInform class structure defined by the gateway client and the Internet of Things web server is defined in Table 1. The data structure consists of five elements: identifier, user ID, sensor type, sensor ID, and sensor value. When the IoT web server receives the data, it can immediately save the sensor type, sensor ID, and sensor value values to the sequence of the User_inform target user according to the user ID.

Table 1. UserInform protobuf class structure definition

Field Name	Field Type	Field length (bits)	Field Description
Identifiers	int32	32	Identifier
User_ID	int32	32	UserID
Sensor_type	int32	32	Sensor Type
Sensor_ID	int32	32	SensorID
Sensor_value	int32	32	Sensor Value

3.1.2 Message Logic Processing

In the Netty framework, the received data stream is decoded into the entity class UserInform. The entity class UserInform handles the logical processing of decoding data through the ChannelPipeline handle handler NettyServerHandler class. The logic processing in the class NettyServerHandler is mainly implemented in its internal channelRead function. The program flow chart of the channelRead function is shown in Figure 3. The information sent by the IoT gateway to the web server is stored in an entity class UserInform . UserInform.User_ID is used as a keyword to query the sessionID of the corresponding Websocket web client in the user login table User_index. The table User_index contains three elements, User_name, User_ID, and Session_ID. The element User_ID in the query table User_index can query the corresponding session_ID value. The Session.getBasic().sendText() function sends a message to the corresponding web client to implement the push of the gateway client information to the corresponding web client. The entity class request persistence is stored in the sequence of the user object of the table User_Inform through the User_ID. The channelRead function queries and reads the contents of the command in the target user sequence of the table User_Order through the User_ID. The ctx. channel. Write and Flush. function sends a message to the target user gateway client to implement the release of the command release.

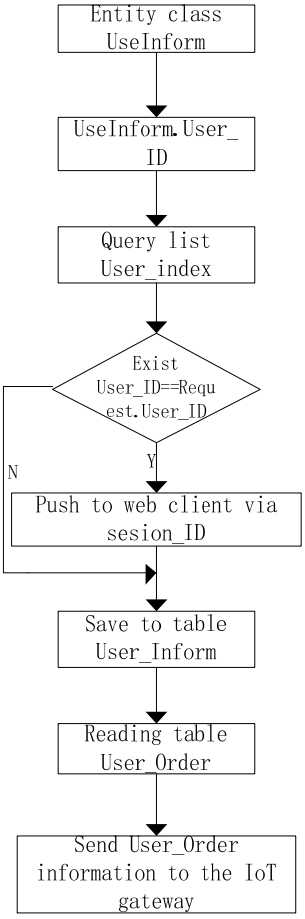

Figure 3. Channelread function program flow chart

3.2 Web Server Websocket Protocol Design

3.2.1 Web Server Communication Platform Websocket Protocol Design

The Websocket protocol is a communication protocol based on a TCP long connection and a new generation of client and server for full duplex communication. The communication platform uses the Websocket protocol to implement interaction between the web client and the web server. The websocket server in the Internet of Things web server communication platform will persist the User_ID and sessionID information of the web client. The websocket server @onopen annotation function saves the user Session_ID in the table User_index, and the @onClose annotation function clears the Session_ID in the User_index sequence, clearing the offline user Session_ID information. During the session, the Websocket server @onMessage annotation function saves the command information of the web client in the table User_Order.

The work flow chart of the Internet of Things web server communication platform Websocket is shown in Figure 2: the web client logins, the websocket handshake is completed by verification, and the sessionID is stored in the User_index sequence by the keyword User_ID. The web server Netty server pushes the information to the websocket client via sessionID and User_ID. The command information sent by the Websocket client is stored in the table User_Order and persisted by the Netty service.

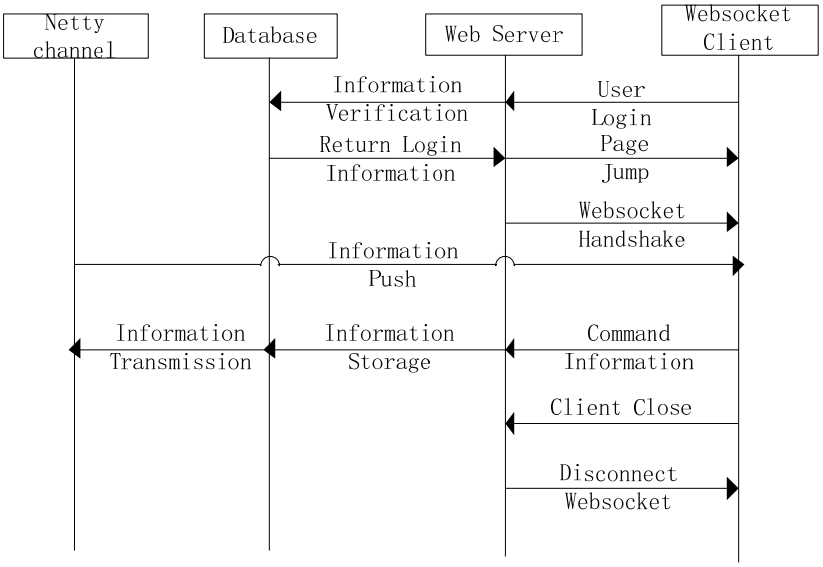

Figure 4. Websocket work flow chart

3.2.2 Analytical Tests and Results

The stress test uses the open source server stress test tool Loadrunner software. TCP connection is used to simulate a large number of IoT gateway clients. Websocket connects the simulated web client to stress test of the Internet of Things web server. The Internet of Things web server communication platform based on Netty and websocket is compared with the traditional IO and Ajax IoT web server in terms of system response time and web server CPU usage.

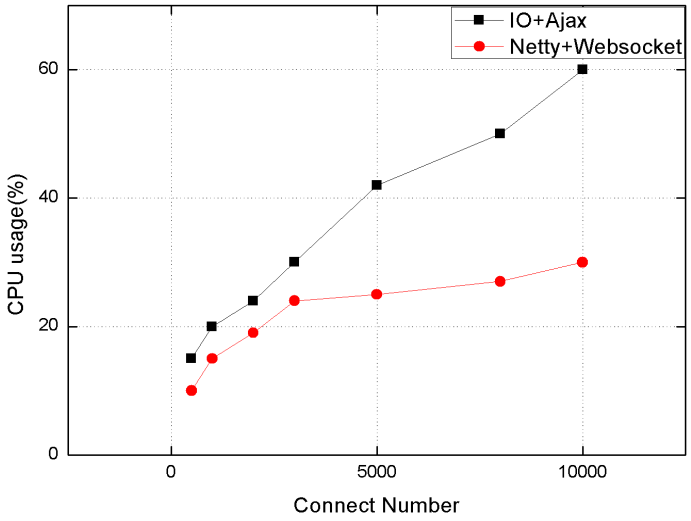

Figure 5. Different ways of web server gateway and client connection CPU usage

In the CPU usage of the number of connections between the web server gateway and the client, it can be seen from FIG. 5 that, as the number of connections increases, the CPU usage of the proposed web server does not increase sharply when the number of connections reaches 10000. At the same time, the CPU usage is still less than 40%, and the system can run stably and efficiently.

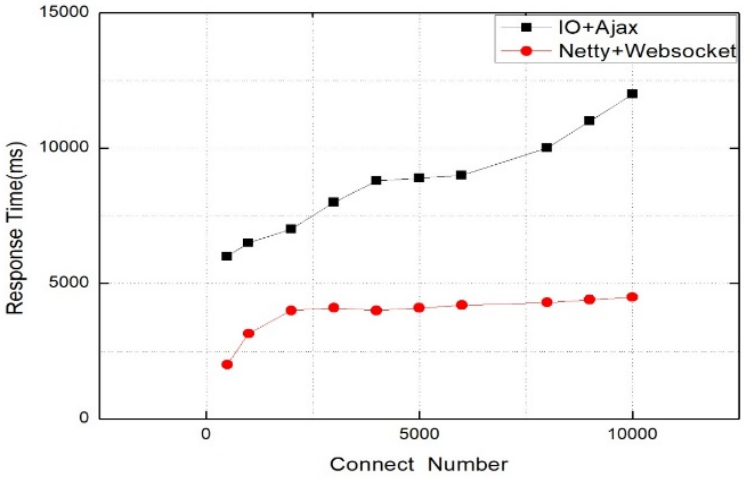

Figure 6. Comparison of response time between different system gateways and web client information

As can be seen from Figure 6, in the system response time unit is milliseconds, each gateway client sends 50 byte data requests to the same web client on average every time, due to the traditional IO-based TCP communication and Ajax polling push information. Ajax re-establishes an HTTP connection every time it connects. It is a waste of bandwidth, and when the number of concurrency is very large, the stack memory and CPU thread switching overhead occupied by the thread increases sharply, consuming a lot of system resou [7].

Rces. When the number of connections of the IoT gateway client is about 6000, the information delay between the gateway client and the web server is obvious and the connection fails abnormally. Based on Netty and websocket IoT web server communication platform, since the thread pool does not need to create additional threads, the web server and web page client only need one Http handshake during the whole communication process, the system overhead is small, and the number of connections is 10000. The system can still process requests from a large number of gateway clients simultaneously. Based on Figure 5 and Figure 6, the response time and CPU usage of the Internet of Things web server communication platform based on Netty and websocket are far superior to those of the traditional Io and Ajax-based IoT web server communication platform. The system runs more efficiently and reliably.

4. Conclusion

Compared with the traditional smart home web server design, the smart home web server is built with the SSM framework, adopts the asynchronous non-blocking network Netty communication framework, and the long-connected Websocket protocol. The system has stable operation, robust communication, small memory footprint and response time. Fast, efficient and so on, it proves to be an excellent smart home web server design.

Acknowledgments

Fund Project: Project supported by the National Natural Science Foundation of China (61490702).

References

[1]. Chan M, Campo E, Estève D, et al. Smart homes—Current features and future perspectives[J]. Maturitas, 2012, 64(2):90-97.

[2]. Tian Li. Realizing the centralized sharing of data for backup card holders [N]. People's Public Security News, 2010-10-11(8).

[3]. Souza A M C, Amazonas J R A. An Outlier Detect Algorithm using Big Data Processing and Internet of Things Architecture [J]. Procedia Computer Science, 2015, 52:1010–1015.

[4]. Kailas A, Cecchi V, Mukherjee A. A survey of communica -tions and networking technologies for energy management in buildings and home automation [J]. Journal of Computer Net -works and Communications, 2012, 932181.

[5]. Jiang Ni, Zhang Yu, Zhao Zhijun. Based on MQTr Internet of Things message push system [J].Network New Media Technology, 2014, 3(6): 62-64.

[6]. Li Rui, Xiao Xiangqiang. Application research of embedded web technology in traffic monitoring system[C]// 2nd Interna -tional Symposium Computer Science and Computational Tec -hnology. Huangshan, China, 2017: 94 – 97.

[7]. Liang Yanjie, Lian Dongben. Data Exchange Platform Transmission Framework Based on Message Middleware Design [J]. Computer System Applications, 2012, 21(4): 10-13.

[8]. Zhang Yizhen, Zhang Zhibin, Zhao Wei, et al. Comparative analysis of TCP and UDP network traffic Research [J]. Computer Applied Research, 2010, 27(6): 2192-2197.