International Journal of Trend in Research and Development, Volume 8(5), ISSN: 2394-9333
www.ijtrd.com

Design and Implementation of High-Performance MQTT Broker Based on Netty

Bing Lv
College of Computer Science and Technology, Shan Dong University of Technology, Zibo, Shandong, China

Abstract: At present, the Iot(Internet of Things) mainly uses HTTP (Hypertext Transfer Protocol), MQTT (Message Queuing Telemetry Transport, Message Queuing Telemetry Transport) or CoAP (Constrained Application Protocol) protocols for data transmission. During the period, MQTT is the most widely used. This system uses the Spring Boot framework and implements the MQTT protocol based on Netty. The main functions are: message publishing and subscription, heartbeat maintenance, topic filtering, Qos message confirmation, data persistence. The system fully supports the MQTT V3.1.1 protocol specification, has the characteristics of low latency and high concurrency, supports connections within 10,000, and fully meets the needs of small-sized enterprises.

Keywords: *IoT; Spring Boot; Netty; MQTT; Qos*

I. Introduction

The publishing and subscribing of messages is the core of the MQTT protocol. The MQTT Broker is responsible for receiving messages from IoT devices and sending them to the message subscribers. This system fully complies with the MQTT V3.1.1 protocol, realizes the core functions of MQTT Broker, and uses the Spring Boot and Netty frameworks with lower development costs, reduces the difficulty of development. The system inherited Netty's easy-to-use, high-performance, and safe features.

II. Related Technology

This system is developed based on Java, using Spring Boot framework, and network communication is based on Netyy framework.

1. Spring Boot

Spring Boot makes it easy to create stand-alone, production-grade Spring based Applications that you can "just run"[1].The Spring Boot framework is further optimized on top of the Spring framework. Spring Boot retains the excellent features inherent in the Spring framework, simplifies the construction and development process of Spring programs, and makes it convenient for developers to use.

2. Netty

Netty is a NIO client server framework which enables quick and easy development of network applications such as protocol servers and clients. It greatly simplifies and streamlines network programming such as TCP and UDP socket server[2].

3. MQTT

MQTT is located in the application layer of the network protocol. It has a unique function of publishing and subscribing to messages. It plays an indispensable role in this system. It is precisely because of the unique communication function of the MQTT protocol that it can act as a " The role of “message broker”. It can provide messaging services for other different clients with few codes and limited bandwidth.

III. System Design

According to different functions, the MQTT Broker is divided into five modules.

1. Subscription Module

The server will receive one or more subscription requests from the client. If the server has not received the message sent by the publisher before, it will send a request to the publisher client to send the corresponding operation instructions or data. If the message sent by the publisher is received before, it will directly perform the corresponding operation or send a message to the subscriber.

2. Publish Module

The server will accept messages from the publisher that other clients may subscribe to. When the server receives the subscription request from the client, it will send the information received from the publisher's client to the subscriber, or directly execute the received operation instruction. If the message from the publisher has not been received before, a request will be sent to the publisher, and then the publisher will send relevant information to the server.

3. Heartbeat Maintenance Module

The server can receive the heartbeat sent from the client, as well as the specific time sent and the heartbeat timeout time.

4. Subject Filtering Module

When the server receives a message from the client, it will have its specific topic name. The server can only send the message through the same topic name, and the client can receive the corresponding message, preventing message confusion and leakage.

5. Qos message confirmation mechanism Module

Whenever the server sends a message to the client, there will be related quality of service qos0, qos1, qos2, and the level from low to high is at most once, at least once, and only once.

IV. System Implementation

1. Import Related Packages

Add the following code to the pom.xml file

```
<dependency>
  <groupId>org.projectlombok</groupId>
  <artifactId>lombok</artifactId>
  <version>1.16.16</version>
</dependency>
<dependency>
  <groupId>com.google.guava</groupId>
  <artifactId>guava</artifactId>
  <version>23.3-jre</version>
</dependency>
```

```
<dependency>
  <groupId>io.netty</groupId>
  <artifactId>netty-all</artifactId>
  <version>4.1.12.Final</version>
</dependency>
<dependency>
  <groupId>org.apache.commons</groupId>
  <artifactId>commons-lang3</artifactId>
  <version>3.5</version>
</dependency>
```

2. System Start

The main entry code of the system is as follows:

```
@SpringBootApplication
@EnableAspectJAutoProxy
public class ServerApplication {
   public static void main(String[] args) {
      ConfigurableApplicationContext run
            = run(ServerApplication.class, args);
   }
}
```

3. MQTT Message Handler

The MQTT message handler is as follows:

```
public abstract class ServerMqttHandlerService implements IMqttHandler
{
   public abstract boolean login(Channel channel,
            MqttConnectMessage mqttConnectMessage);
   public abstract  void  publish(Channel channel,
            MqttPublishMessage mqttPublishMessage);
   public abstract void subscribe(Channel channel,
            MqttSubscribeMessage mqttSubscribeMessage);
   public abstract void pong(Channel channel);
   public abstract  void unsubscribe(Channel channel,
            MqttUnsubscribeMessage mqttMessage);
   public abstract void disconnect(Channel channel);
   public abstract void doTimeOut(Channel channel,
            IdleStateEvent evt);
}
```

4. Faculty development

The processing method of half-packet and sticky packet of message is as follows:

```
public  abstract  class MqttHander extends
SimpleChannelInboundHandler<MqttMessage>
{
   private IMqttHandler mqttHandler;
   public MqttHander(IMqttHandler mqttHandler){
      this.mqttHandler=mqttHandler;
   }
   @Override
   protected void channelRead0(ChannelHandlerContext
   channelHandlerContext, MqttMessage mqttMessage)
   throws Exception {
      MqttHeader mqttHeader = mqttMessage.fixedHeader();
      Optional.ofNullable(mqttHeader) .ifPresent
      ( t -> doMessage(channelHandlerContext,mqttMessage));
   }
   .....
}
```

CONCLUSION

This system uses the current mainstream development technology, uses the Netty framework to encapsulate the MQTT protocol, and realizes the core functions of MQTT Broker, such as publish and subscribe, heartbeat maintenance, topic filtering, qos message confirmation mechanism, data persistence, etc. The system has the characteristics of high concurrency and low latency.

References

[1] Computers; Researchers from University of Jordan Describe Findings in Computers (Augmented Whale Feature Selection for Iot Attacks: Structure, Analysis and Applications)[J]. Information Technology Newsweekly,2020.

[2] Information Technology - Data Analytics; Findings on Data Analytics Discussed by Investigators at Newcastle University (Multiobjective Deployment of Data Analysis Operations In Heterogeneous Iot Infrastructure) Information Technology Newsweekly,2020.

[3] Information Technology - Cloud Computing; Data on Cloud Computing Described by Researchers at School of Electrical Engineering (Iot Embedded Cloud-based Intelligent Power Quality Monitoring System for Industrial Drive Application). Information Technology Newsweekly,2020.