MQTT y Node Js.

El concepto de MQTT

MQTT es un protocolo de transmisión de publicación / suscripción de mensajes basado en cliente-servidor. El protocolo MQTT es ligero, simple, abierto y fácil de implementar, estas características lo convierten en una amplia gama de aplicaciones. En muchos casos, incluidos los entornos restringidos, como la comunicación de máquina a máquina (M2M) y la Internet de las cosas (IoT). Se ha utilizado ampliamente para comunicar sensores a través de enlaces satelitales, ocasionalmente equipos médicos de acceso telefónico, hogares inteligentes y algunos dispositivos miniaturizados.

Diferencia con otros protocolos de transporte

MQTT es un protocolo Pub / Sub basado en TCP diseñado para escenarios de IoT. Tiene muchas características optimizadas para IoT, como la adaptación a diferentes QoS de red, temas jerárquicos, últimas palabras, etc.

WebSocket es un protocolo diseñado para aplicaciones HTML5 para facilitar la comunicación bidireccional con el servidor. Protocolo de enlace HTTP y luego transferencia al protocolo TCP para reemplazar las implementaciones antiguas como Server Push, Comet y sondeo largo. Todos los dos tienen intersecciones debido a un escenario de aplicación: cómo usar la aplicación HTML5 como cliente de MQTT para recibir mensajes de dispositivo o enviar información a dispositivos, entonces MQTT sobre WebSocket se convierte naturalmente en la forma más razonable.

Soporte de idiomas para el cliente MQTT

- Java
- Javascript
- C/C++
- Python
- Ruby
- Objective-C

Crear un servidor

1. Instalar dependencias **npm install mosca mqtt**
2. Crear directorio raíz mqtt.js

```
const mosca = require("mosca");
const MqttServer = new mosca.Server({
  port: 1883
});
MqttServer.on("clientConnected", function(client) {
  // Devolución de llamada cuando hay una conexión de cliente.
  console.log("client connected", client.id);
});
/**
   * Supervisar mensajes de temas MQTT
   * Cuando el cliente tiene una conexión para publicar un mensaje de tema
 **/
MqttServer.on("published", function(packet, client) {
  var topic = packet.topic;
  switch (topic) {
    case "temperature":
      // console.log('message-publish', packet.payload.tostring());
      // MQTT puede reenviar mensajes de temas a otros temas
      //MqttServer.publish({ topic: 'other', payload: 'ssssss' });
      break;
    case "other":
      console.log("message-123", packet.payload.toString());
      break;
  }
});

MqttServer.on("ready", function() {
  // Devolución de llamada cuando se inicia el servicio
  console.log("mqtt is running...");
});
```

3. Crear publicador pub.js

```
const mqtt = require("mqtt");
const client = mqtt.connect("mqtt://127.0.0.1:1883"); // Conéctese al servidor mqtt
// Escriba un temporizador para enviar información meteorológica regularmente cada 3 segundos, este servicio puec
setInterval(function() {
  const value = Math.ceil(Math.random() * 40);
  client.publish("temperature", value.toString(), { qos: 0, retain: true });
}, 3000);
```

4. Crear suscriptor sub.js

```
const mqtt = require("mqtt");
// const mqtt = require('./node_modules/mqtt/dist/mqtt.min.js')
const client = mqtt.connect("mqtt://127.0.0.1:1883"); // Especifique la dirección y el puerto del servidor

client.on("connect", function() {
  console.log("La conexión al servidor es exitosa");
  // connected = client.connected
  client.subscribe("temperature", { qos: 1 }); // Suscríbete a noticias con prueba de tema
});
client.on("message", function(top, message) {
  console.log("Tema actual:", top);
  console.log("Temperatura actual:", message.toString());
});
```