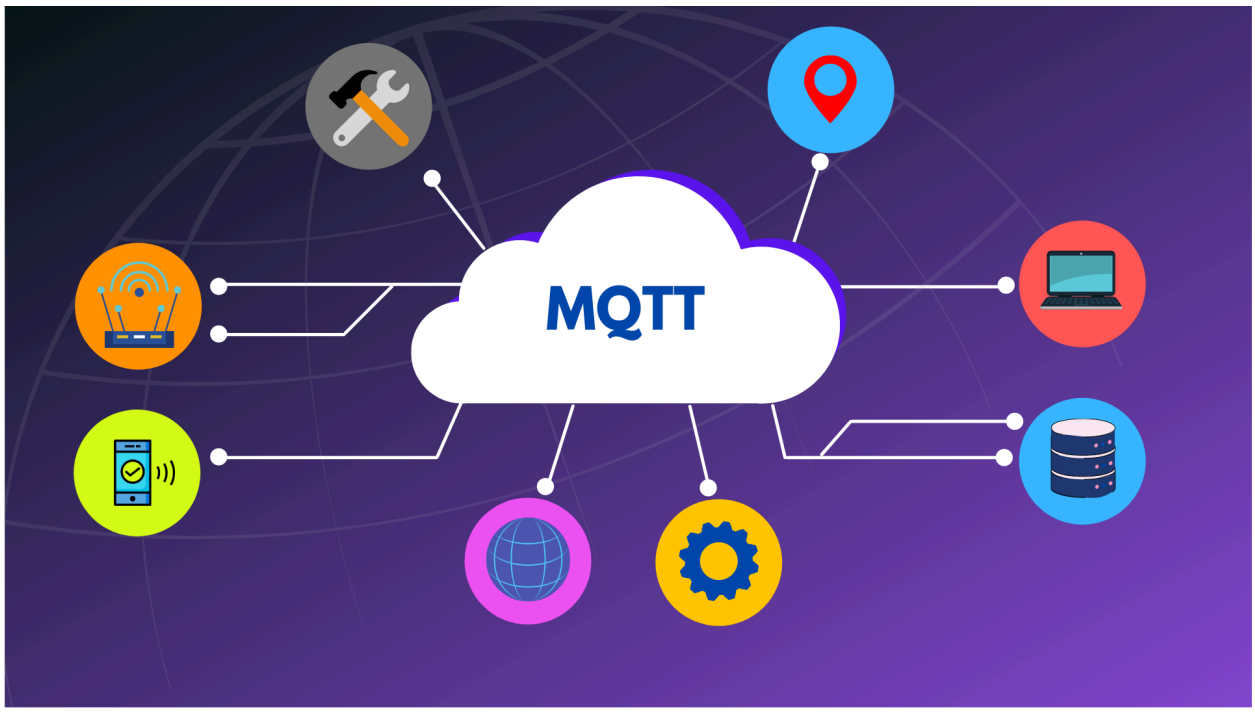
SIT782 Capstone Team Project(B)

Research Report

HiveMQ - Introduction and Implementation



Ayush Kumar Som
222198016

1. Introduction to HiveMQ

HiveMQ is a MQTT broker designed for enterprise solutions that require robust, scalable, and reliable messaging. It supports MQTT 3.x and MQTT 5.0 protocols and is particularly noted for its ability to handle high-throughput scenarios and massive numbers of connections. HiveMQ is built to ensure seamless data flow across devices, making it an ideal choice for projects involving real-time sensor data communication.

2. Project Requirements Recap

Sensors Overview

- **Neo 6M GPS:** Transmits location data.
- **Accelerometer (Built-in Arduino Nano):** Captures motion-related data.
- **Oximeter MAX30100:** Monitors blood oxygen saturation and pulse rate.
- **Heart Rate Sensor:** Tracks the heart rate.
- **Camera Module - Pi Cam:** Streams video and still images.

Data Flow Needs

- **Volume and Frequency:** High data volume from the camera; frequent updates from the health sensors.
- **Connectivity Requirements:** High reliability, real-time data transmission, low latency.

3. HiveMQ Features Relevant to the Project

- **High Scalability:** Supports clustering to manage large-scale deployments.
- **Robust Security:** Offers SSL/TLS encryption and advanced authentication mechanisms.
- **Extension Framework:** Allows customizations and integrations, including integration with backend systems and databases.
- **Bridge and Cluster:** Facilitates connections with other MQTT brokers and redundant server setups for enhanced reliability.

4. Implementation Plan

Step 1: HiveMQ Installation

- **Environment Setup:** Choose a server environment (cloud, on-premise). For this project, a cloud-based instance (e.g., AWS, Azure) is recommended for scalability and ease of management.
- **Installation:** Download and install HiveMQ. Docker containers are also available for easier deployment and scalability.

Step 2: Configuration

- **Basic Configuration:** Configure hivemq.xml to set up basic settings such as listener ports.
- **Security Configuration:** Set up SSL/TLS for secure communication. Implement client certificate authentication for devices.
- **Clustering Configuration:** If high availability is critical, configure HiveMQ clustering.

Step 3: Integration with Sensors

- **Sensor Setup:** Each sensor should be configured to publish data to the HiveMQ broker. Arduino and Raspberry Pi can use libraries like PubSubClient or paho-mqtt for this purpose.

Step 4: Data Handling

- **Topic Structure Design:** Design an MQTT topic hierarchy that categorizes data logically (e.g., sensor/gps, sensor/heart_rate).
- **Data Subscription:** Set up subscription logic on the server or other subscribing clients to handle incoming data.

Step 5: Monitoring and Maintenance

- **Broker Monitoring:** Utilize HiveMQ's monitoring tools to track broker performance and client interactions.
- **Data Logging and Analysis:** Implement logging for troubleshooting and data analysis.

5. Sample Code

Users > ayushsom > Documents > HIVEMQ.cpp > setup_wifi()

Click here to ask Blackbox to help you code faster

```cpp
#include <PubSubClient.h>
#include <ESP8266WiFi.h>

const char* ssid = "yourSSID";
const char* password = "yourPassword";
const char* mqtt_server = "broker.hivemq.com";

WiFiClient espClient;
PubSubClient client(espClient);

Codiumate: Options | Test this function
void setup_wifi() {
  delay(10);
  // Connect to WiFi
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
  }
}

Codiumate: Options | Test this function
void reconnect() {
  while (!client.connected()) {
    if (client.connect("arduinoClient")) {
      // Subscribe or publish if necessary
    }
  }
}

Codiumate: Options | Test this function
void setup() {
  setup_wifi();
  client.setServer(mqtt_server, 1883);
}

Codiumate: Options | Test this function
void loop() {
  if (!client.connected()) {
    reconnect();
  }
  client.loop();

  // Assume getHeartRate() is a function that reads heart rate data
  String heartRate = String(getHeartRate());
  client.publish("sensor/heart_rate", heartRate.c_str());
}
```

6. Conclusion

Deploying HiveMQ as the MQTT broker for this project will significantly enhance the reliability, scalability, and security of data communications between the sensors and the central server. By following the outlined steps and using the provided sample code, the project can efficiently handle the data requirements of each sensor, ensuring real-time performance and robustness.