Instruções Aplicação MQTT

Henrique C. Oliveira[1], **Antonio B. Pagotto**[1], **Marcelo M. Silva**[1]

[1]Instituto de Matemática e Computação – Universidade Federal de Itajubá (UNIFEI)
Caixa Postal 37.500-903 – Itajubá – MG – Brasil

Abstract. *This article aims to present a simple application that uses the MQTT (Message Queuing Telemetry Transport) protocol to communicate between a client and a server. This application was developed to perform three distinct operations: perform mathematical calculations, respond to text, and modify a text file on the server. MQTT is a lightweight and efficient messaging protocol that allows internet-connected devices to communicate asynchronously and in real-time. The application presented in this article is a practical demonstration of the use of MQTT in creating a simple and functional solution. Details about the implementation of the application will be presented, as well as information on how to set up the environment to test MQTT communication between client and server.*

Resumo. *Este artigo tem como objetivo apresentar uma simples aplicação que utiliza o protocolo MQTT (Message Queuing Telemetry Transport) para realizar a comunicação entre um cliente e um servidor. Essa aplicação foi desenvolvida para realizar três operações distintas: fazer cálculos matemáticos, responder um texto e alterar um arquivo de texto no servidor. O MQTT é um protocolo de mensagens leve e eficiente que permite que dispositivos conectados à internet possam se comunicar de forma assíncrona e em tempo real. A aplicação apresentada neste artigo é uma demonstração prática do uso do MQTT na criação de uma solução simples e funcional. Serão apresentados detalhes sobre a implementação da aplicação, bem como informações sobre como configurar o ambiente para testar a comunicação MQTT entre cliente e servidor.*

1. MQTT (Message Queuing Telemetry Transport)

O MQTT é um protocolo de mensagens leve e eficiente desenvolvido para permitir a comunicação entre dispositivos conectados à internet em tempo real e de forma assíncrona. Ele foi projetado para ser usado em redes com largura de banda limitada ou instável, tornando-o uma escolha popular para a Internet das Coisas (IoT) e outras aplicações que exigem troca de dados em tempo real. O MQTT utiliza o modelo publisher-subscriber, onde os dispositivos são conectados a um broker MQTT, que é responsável por rotear as mensagens para os clientes interessados. O protocolo é fácil de implementar, possui baixo consumo de recursos e é altamente escalável, tornando-o uma solução eficaz para a comunicação entre dispositivos IoT e outras aplicações distribuídas.

1.1. Broker

O Broker é um componente central em sistemas que utilizam o modelo de comunicação publisher-subscriber, como o MQTT. Ele é responsável por intermediar a comunicação entre os dispositivos que publicam mensagens (publishers) e os dispositivos que se inscrevem para receber essas mensagens (subscribers).

O broker recebe as mensagens publicadas pelos publishers e as encaminha para os subscribers inscritos nos tópicos relevantes. Ele também gerencia as inscrições e desinscrições de clientes nos tópicos, além de garantir a entrega confiável de mensagens aos subscribers.

Em outras palavras, o broker funciona como um servidor de mensagens, que permite que os dispositivos conectados à rede possam trocar informações de forma assíncrona e em tempo real. É importante escolher um broker confiável e escalável para garantir a eficiência e a confiabilidade do sistema de comunicação.

1.2. Pub/Sub

O modelo publisher-subscriber é um padrão de comunicação que permite que um conjunto de dispositivos se comuniquem de forma assíncrona. Nesse modelo, os dispositivos são divididos em dois tipos principais: publishers (publicadores) e subscribers (assinantes). Os publishers são responsáveis por enviar mensagens para um tópico específico, enquanto os subscribers se inscrevem nesses tópicos para receber as mensagens que estão interessados.

Dessa forma, os publishers e subscribers não precisam saber da existência um do outro, e a comunicação é intermediada por um broker que roteia as mensagens para os subscribers adequados. Esse modelo de comunicação é amplamente utilizado em sistemas distribuídos, como a Internet das Coisas (IoT), onde diversos dispositivos precisam trocar informações de forma assíncrona e em tempo real.

1.3. DLQs (Dead Letter Queues)

No modelo publisher-subscriber, é comum o uso de filas, também conhecidas como ”Dead Letter Queues” (DLQs), para lidar com as falhas no recebimento de mensagens pelos subscribers. As DLQs são filas em que as mensagens são armazenadas temporariamente quando não são entregues aos subscribers por algum motivo, como falta de conexão ou sobrecarga de dados. Essas mensagens podem ser recuperadas posteriormente pelos subscribers, garantindo que nenhuma informação seja perdida. As DLQs são importantes para garantir a confiabilidade e a robustez do sistema, especialmente em casos de comunicação em redes instáveis ou sujeitas a interrupções.

2. Desenvolvimento da Aplicação

Essa aplicação em Python utiliza a biblioteca Paho-MQTT para comunicação cliente-servidor e um broker público da EMQX para conexão em nuvem. O virtual environment para Python é usado para gerenciamento independente de pacotes e dependências. A solução permite a realização de operações como contas matemáticas, respostas a texto e alteração de arquivos de texto.

2.1. Python

Python é uma linguagem de programação de alto nível, interpretada e orientada a objetos. Ela foi criada no final dos anos 80 por Guido van Rossum e tem se tornado cada vez mais popular nos últimos anos, especialmente em aplicações de inteligência artificial, análise de dados e desenvolvimento web. Python é conhecida por sua sintaxe clara e legível, o que a torna fácil de aprender e de manter. Além disso, ela possui uma vasta biblioteca padrão,

que oferece recursos para diversas tarefas comuns de programação, como manipulação de arquivos, comunicação em rede, processamento de strings e muitas outras.

Outra vantagem de Python é sua grande comunidade de desenvolvedores, que contribuem com pacotes e bibliotecas que expandem as funcionalidades da linguagem. Essa comunidade ativa também oferece suporte e recursos para a resolução de problemas, tornando a linguagem uma escolha popular para desenvolvedores de todos os níveis de experiência.

Em resumo, Python é uma linguagem de programação poderosa, fácil de aprender e com uma grande comunidade de desenvolvedores. Sua flexibilidade e eficiência a tornam uma escolha popular em muitas áreas da programação, desde a análise de dados e machine learning até o desenvolvimento web e de jogos.

2.2. Paho-MQTT

Paho-MQTT é uma biblioteca de código aberto para a implementação de protocolos MQTT em linguagens de programação como Python, Java, C e outras. Ela foi desenvolvida pela Fundação Eclipse e é amplamente utilizada em projetos de Internet das Coisas (IoT) e outras aplicações de comunicação em rede.

Com o Paho-MQTT, os desenvolvedores podem facilmente criar aplicações que se comuniquem com dispositivos IoT e outros sistemas de comunicação baseados em MQTT. A biblioteca oferece suporte a recursos avançados, como o uso de tópicos wildcard, garantia de entrega de mensagens e a comunicação criptografada através do uso de SSL/TLS.

2.3. Variáveis de Ambiente

Para que a aplicação funcione corretamente, é necessário definir algumas variáveis de ambiente, como o endereço do broker, sua porta, usuário e senha de conexão. Esses dados são essenciais para estabelecer uma conexão segura e confiável entre cliente e servidor.

Como se trata de uma aplicação de demonstração, os valores utilizados são apenas mocks públicos. No exemplo apresentado, as variáveis de ambiente definidas são:

BROKER=”broker.emqx.io”;

PORT=1883;

USERNAME=”emqx”;

PASSWORD=”public”.

3. Utilização da Aplicação

1. A aplicação foi dividida em três arquivos .py, sendo um para o servidor, outro para o cliente e um terceiro para funções e configurações compartilhadas entre os dois primeiros. Essa estrutura permite uma maior organização e facilidade de manutenção do código.
2. Para utilizar a aplicação, é necessário definir algumas variáveis de ambiente na máquina do cliente e servidor. Essas variáveis incluem o endereço do broker, a porta, usuário e senha de conexão ao broker. É importante ressaltar que, como se trata de uma aplicação de demonstração, os valores utilizados são apenas mocks públicos.

3. Após definir as variáveis de ambiente necessárias, é possível executar a aplicação através do comando python3 CAMINHO-PARA-ARQUIVO. Com o servidor e pelo menos um cliente em execução, é possível interagir com os prompts no terminal do cliente para realizar as operações desejadas, como contas matemáticas, respostas a texto e alteração de arquivos de texto no servidor.
4. Em resumo, a aplicação em questão oferece uma solução simples e eficiente para comunicação cliente-servidor utilizando o protocolo MQTT. Sua estrutura modular permite uma fácil manutenção e atualização do código, enquanto as variáveis de ambiente e instruções para utilização garantem uma conexão segura e confiável entre cliente e servidor.

4. References

https://www.emqx.com/en/blog/how-to-use-mqtt-in-python
https://www.eclipse.org/paho/index.php?page=clients/python/docs/index.php