**Logotipo

Descrição gerada automaticamente**

Serviço Nacional de Aprendizagem Industrial

SENAI “Gaspar Ricardo Junior”

Curso

TÉCNICO EM DESENVOLVIMENTODE SISTEMAS

*MQTT*

*(Message Queuing Telemetry Transport)*

**Nome:** miguel. s.

**Nome Professores:** Vedilson prado; leandro rosa.

Sorocaba

Abril – 2024

1. **O que é o MQTT?**

MQTT(Message Queuing Telemetry Transport) é um protocolo de comunicação máquina para máquina (M2M - Machine to Machine) com alvo em Internet of Things (IoT) que funciona em cima do protocolo TCP/IP. Um sistema que tem base em comunicação entre o usuário e o servidor, em que o primeiro pode realizar tantos publicações quanto captação de informação e o segundo administra as referências a serem recebidos.

1. **Como usar o protocolo MQTT?**

Conforme mencionado, o protocolo MQTT é amplamente utilizado em aplicações IoT devido à sua simplicidade e facilidade de operação. Além dos aplicativos IoT, outro aplicativo muito popular é a coleta de dados em tempo real.

Podemos citar o exemplo apresentado no artigo “Como funciona” acima, onde o protocolo MQTT é utilizado para coletar informações sobre temperatura e umidade do solo. Por exemplo, se o solo não estiver molhado e a temperatura estiver alta, o sistema de irrigação funcionará.

O uso deste protocolo fica inteiramente a critério do desenvolvedor. Podem ser criados sistemas para controle de estoque, processamento automatizado, controle de fluxo, controle de eficiência energética e muito mais.

**Como começar com o protocolo MQTT**

Atualmente, existem muitas implementações de fornecedores e clientes de MQTT em diferentes linguagens, como **Python, JavaScript, C#,** entre outras. Podemos encontrar facilmente algumas dessas implementações na Internet, muitas delas são de código aberto e podem ser facilmente acessadas e baixadas pelo público.

Uma dessas implementações é o corretor Mosquitto. Um dos dispositivos de consumo mais utilizados atualmente devido à sua simplicidade e facilidade de uso, essa tecnologia de código aberto ganhou muito espaço no mercado de IoT por estar intimamente relacionada ao protocolo MQTT.

Usaremos o Mosquitto como exemplo de alternativas e também mostraremos como instalá-lo e como você pode começar a usá-lo em um aplicativo real. Caso você tenha interesse em outros exemplos de Brokers, escreveremos no final outras opções que podem ser utilizadas.

Mosquitto pode ser facilmente baixado para vários aplicativos. Você pode encontrar os downloads na página oficial do provedor, na aba Downloads. Depois de baixar a corretora escolhida, você se acostumará a utilizar o servidor onde será tratado o sistema e o relacionamento entre o cliente e a corretora.

A partir de agora você pode consultar seu agente. Este teste pode ser realizado através do comando do próprio computador, onde o cliente pode se cadastrar sobre determinados temas. Além disso, se você já possui um aplicativo da web, pode vinculá-lo ao provedor (seu servidor), adicioná-lo como cliente de assinatura e cliente editor.

Conforme mencionado anteriormente, vários sistemas podem ser utilizados como clientes deste protocolo. Por exemplo, podemos utilizar sensores, que podem pegar alguns dados do ambiente e enviá-los ao vendedor. Podemos usar o RaspberryPi, que consegue obter esses dados do fornecedor – enviados por sensores – para controlar um determinado dispositivo. Também podemos ter um sistema que, como mencionado acima, pode utilizar os mesmos dados enviados pelo sensor ao provedor, mas para uma finalidade que pode ser diferente ou diferente do RaspberryPi.

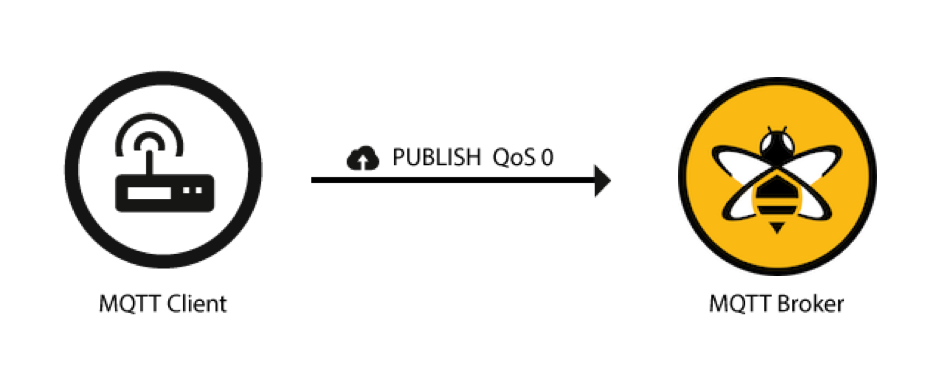
1. **Qualidade de serviço**

No protocolo MQTT nós temos 3 qualidades de serviço(QoS) e cada conexão com o broker pode especificar qual será utilizada., sendo estas: "no máximo uma vez", "no mínimo uma vez" e "exatamente uma vez".

**QoS 0 - No máximo uma vez**

Neste modo de transmissão a mensagem é enviada pelo menos uma vez, aguardando uma resposta ao envio da mensagem, denominada PUBACK. Se você não receber um PUBACK, as mensagens continuarão sendo enviadas até que uma resposta seja recebida. Neste QoS, uma mensagem pode ser enviada diversas vezes e processada diversas vezes.

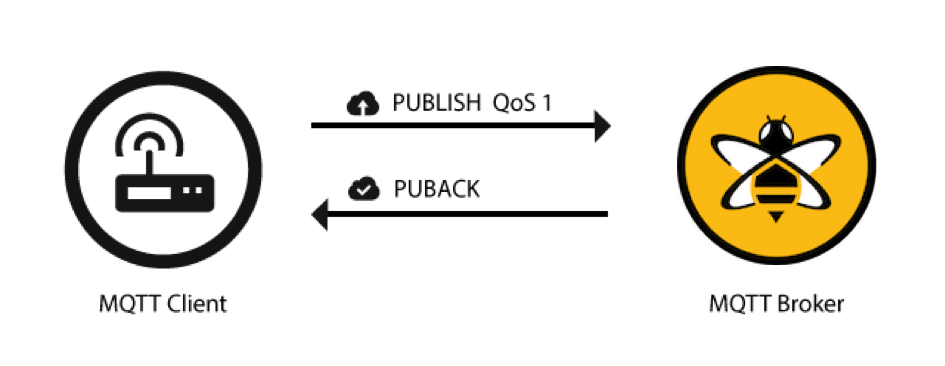
Para que uma mensagem seja enviada várias vezes, ela precisa ser salva. Ele será retirado do destinatário após receber uma resposta para confirmar o envio.



**QoS 1 - Pelo menos uma vez**

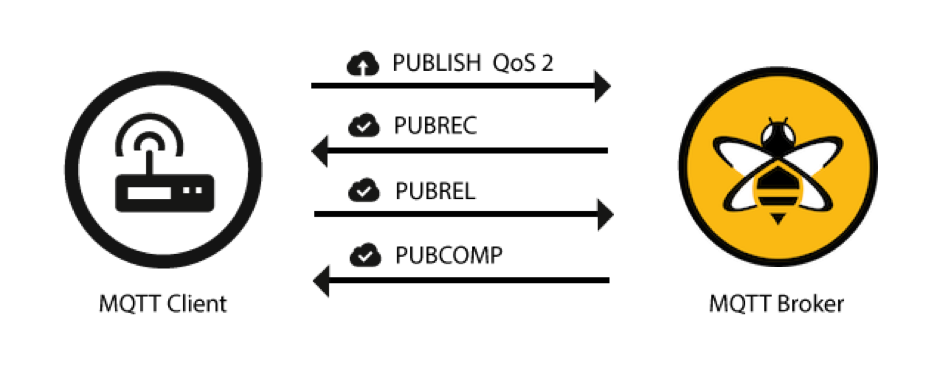
Neste modo de transmissão a mensagem é enviada pelo menos uma vez, aguardando uma resposta ao envio da mensagem, denominada PUBACK. Se você não receber um PUBACK, as mensagens continuarão sendo enviadas até que uma resposta seja recebida. Neste QoS, uma mensagem pode ser enviada diversas vezes e processada diversas vezes.

Para que uma mensagem seja enviada várias vezes, ela precisa ser salva. Ele será retirado do destinatário após receber uma resposta para confirmar o envio.



**QoS 2 - Exatamente uma vez**

Neste modo de transmissão, a mensagem é enviada diretamente uma vez, exigindo que a mensagem seja armazenada localmente no remetente e no destinatário até ser processada. Para garantir esta QoS, é necessário enviar 2 pares solicitação-resposta (chamados de handshake de quatro partes), nos quais podemos enviar mensagem (PUBLICATE), receber resposta (PUBREC), confirmar o recebimento recebido. do PUBREC (PUBREL) e para confirmar se o processo foi concluído e a remoção pode ser realizada (PUBCOMP). Assim que o PUBREL for recebido, o destinatário poderá excluir a mensagem e, assim que o remetente receber o PUBCOMP, a mensagem poderá ser excluída.



**Cliente**

Os clientes dispõem de duas áreas de atuação para informações: publicação e recebimento, respectivamente “editor” e “assinante”. Ele pode escolher em qual área trabalhar, podendo atuar em delivery, recepção ou ambos. Mas não importa o caso que ele escolha, a presença do corretor é sempre necessária para intermediar os dados entre todos os clientes.

**Relação Cliente x Broker**

Todas as informações recebidas e entregues pelo agente são organizadas em formato hierárquico de acordo com o seu tema. Isso significa que os dados capturados e enviados à corretora pertencerão a apenas 1 tópico, enquanto dados diferentes pertencerão a outro tópico e assim por diante. Podemos dar um exemplo:

“Dois sensores são colocados na plantação. Um mede a umidade do solo e o outro mede a temperatura local. Ambos estão conectados ao servidor e os dados capturados são enviados a cada 30 minutos.”

Neste exemplo, ambos os sensores são publicadores, ou seja, enviam dados para um agente que armazena e controla esses dados. No entanto, eles não são armazenados no mesmo local (no mesmo tópico). Por exemplo, os dados relacionados à temperatura local seriam armazenados no tópico “Temperatura”, enquanto os dados relacionados à umidade do solo seriam armazenados no tópico “Umidade”. Além destes 2 clientes, teremos outros clientes, nomeadamente assinantes atuais. Por exemplo, seriam Raspberry Pis conectados ao sistema de irrigação do local. O Raspberry receberá do corretor dados sobre umidade e temperatura do solo no local e executará suas tarefas.

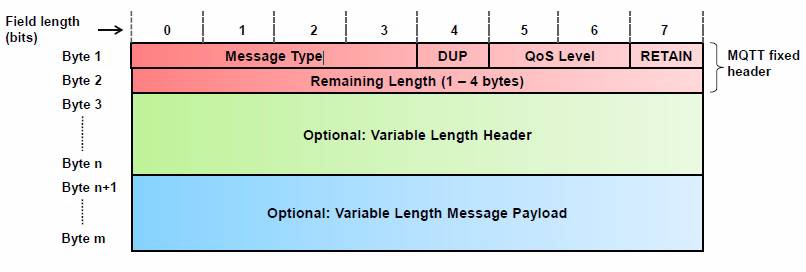
Quanto a “receber dados da corretora”, é isso. Não é o cliente que solicita dados do servidor. Por já estar listado no tópico apropriado, a corretora sabe que deve enviar esses novos dados para aquele destinatário sem exigir solicitação. Além desse benefício, essa abordagem permite que o cliente editor não precise saber para quem esses dados devem ser enviados, pois esse é o trabalho do proxy.

**Trasmissão de dados**

O protocolo MQTT usa outro protocolo chamado TCP para transmissão de dados. Além do TCP, o MQTT-SN também é utilizado para outros tipos de transporte, como UDP ou Bluetooth.

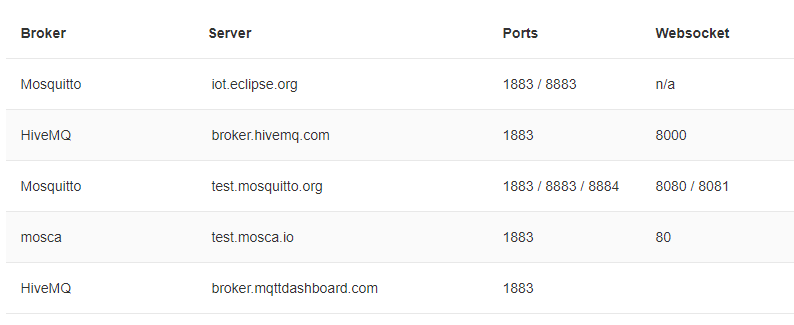
**Header**

Os cabeçalhos MQTT podem variar em comprimento de 2 a 5 bytes. Em relação ao primeiro byte obrigatório, os primeiros 4 bits representam o tipo de mensagem, o próximo bit representa o indicador de mensagem duplicada, dois bits representam a QoS (Qualidade de Serviço) do pacote e um bit representa a QoS (Qualidade de Serviço). do pacote. As mensagens devem ser persistidas ou não para que quando alguém se conectar receba a última mensagem enviada. Os próximos 4 bytes definirão o tamanho do restante do pacote, variando de 0 a 268.435.455 bits. O restante das informações pode variar e não existe um padrão.



**Broker MQTT**

É um servidor intermediário de informações. Ele recebe dados de sensores, processa e transmite esses dados. Pode haver vários corretores no sistema e eles compartilharão os dados recebidos entre eles com base nos clientes que possuem e nos dados que solicitaram.



5 Serviços de broker online para teste

1. **IoT**

**Comunicação de dispositivos:** o MQTT foi projetado para permitir a comunicação entre dispositivos, como sensores, atuadores e gateways, em um ambiente de baixa largura de banda e alta latência. Isso o torna um protocolo ideal para aplicações IoT em que os dispositivos geralmente estão conectados à Internet por meio de links de baixa largura de banda.

**Modelo Publicar-Assinar**: MQTT usa um modelo publicar-assinar, onde os dispositivos podem publicar mensagens em um tópico e outros dispositivos podem se inscrever para receber mensagens desse tópico. Este modelo permite uma comunicação eficiente entre dispositivos e permite que os dispositivos se comuniquem entre si de maneira descentralizada.

**Baixo consumo de energia:** o MQTT foi projetado para ser eficiente em termos energéticos, o que é fundamental para dispositivos IoT que geralmente possuem fontes de energia limitadas. O protocolo utiliza um mecanismo keep-alive para manter conexões e enviar mensagens em pequenos pacotes, reduzindo o consumo de energia.

**Escalabilidade**: o MQTT foi projetado para escalar horizontalmente, permitindo lidar com um grande número de dispositivos e mensagens. Isso o torna adequado para implantações de IoT em larga escala.

**Segurança**: MQTT fornece recursos de segurança integrados, como criptografia e autenticação, para garantir a comunicação segura entre dispositivos.

BIBLIOGRAFIA

Disponível em: < <https://www.gta.ufrj.br/ensino/eel878/redes1-2019-1/vf/mqtt/> > Acessado em: 30/04/2024.

Disponível em: < <https://www.ibm.com/developerworks/br/library/iot-mqtt-why-good-for-iot/index.html> > Acesso em: 30/04/2024