

PROTOCOLO MQTT

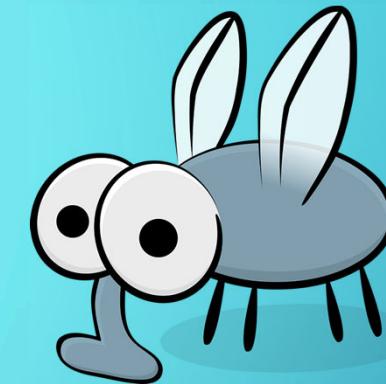
MQTT – Message Queuing Telemetry Transport

HISTÓRIA

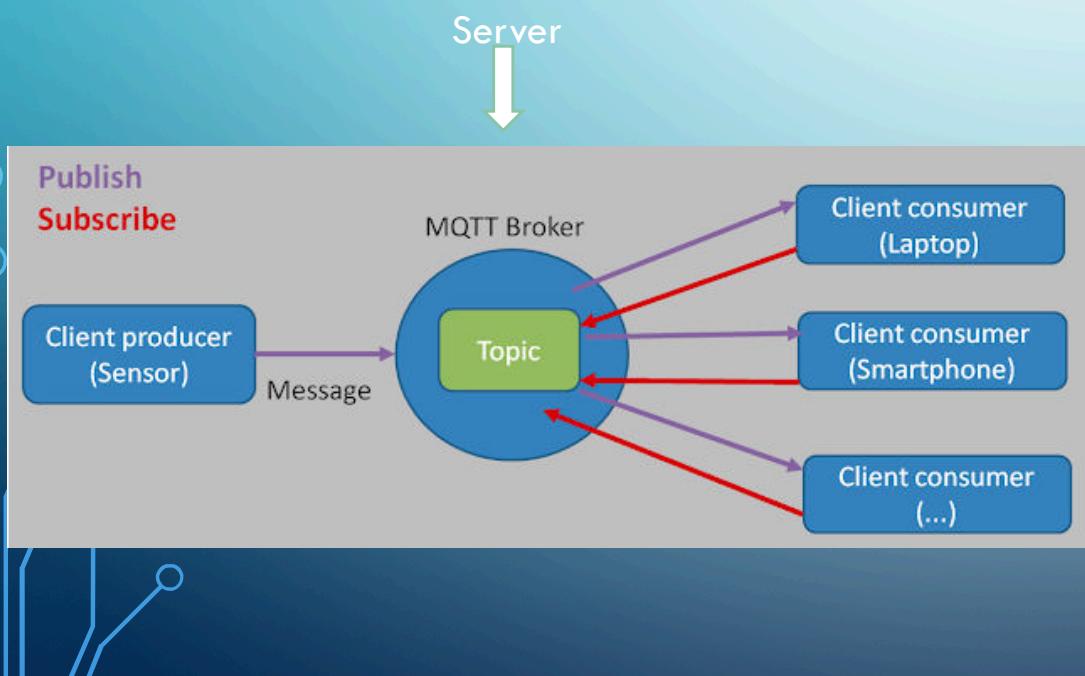
- Nos anos 90 a IBM e Eurotech criaram o protocolo MQTT. Sua origem se deu à necessidade de um protocolo simples e leve que conseguisse comunicar várias máquinas entre si, uma comunicação que ocorreria utilizando microcontroladores para a obtenção de dados que tivesse uma taxa de transmissão leve para a comunicação entre as máquinas e os sensores (telemetria de oleodutos via satélite).
- Devido à tão importante funcionalidade de “criar uma conexão” entre pequenos sensores e as máquinas onde esses dados serão tratados, o MQTT é visto como uma das principais tecnologias que podem tanto participar quanto impulsionar o desenvolvimento de uma nova “rede”, a chamada Internet das Coisas (IoT), que já está mostrando sua importância há muito tempo e, com certeza, ganhará ainda mais foco nos próximos anos.

PRINCÍPIO DE FUNCIONAMENTO

- MQTT (Message Queuing Telemetry Transport) é um protocolo de comunicação máquina para máquina (M2M - Machine to Machine) com foco em Internet of Things (IoT) que funciona em cima do protocolo TCP/IP. Um sistema MQTT se baseia na comunicação entre cliente e servidor, em que o primeiro pode realizar tanto “postagens” quanto “captação” de informação e o segundo administra os dados a serem recebidos e enviados.



PRÍNCIPIO DE FUNCIONAMENTO



- **Publicador/ Publisher:** Quem envia dados para um tópico, emissor.
- **Subscrito/ Subscriber:** Pessoa que está inscrita no tópico e recebe os dados, receptor.
- **Broker:** Intermédio de comunicação entre Publicador e Subscrito, responsável por receber, enfileirar e enviar as mensagens.
- **Payload:** Conteúdo da mensagem enviada.
- **Cliente/Client:** Elemento capaz de interagir com o Broker, seja para enviar, receber ou os dois.
- **Mensagem:** Pacote de dados trocados entre Clientes e Broker.

CARACTERÍSTICAS

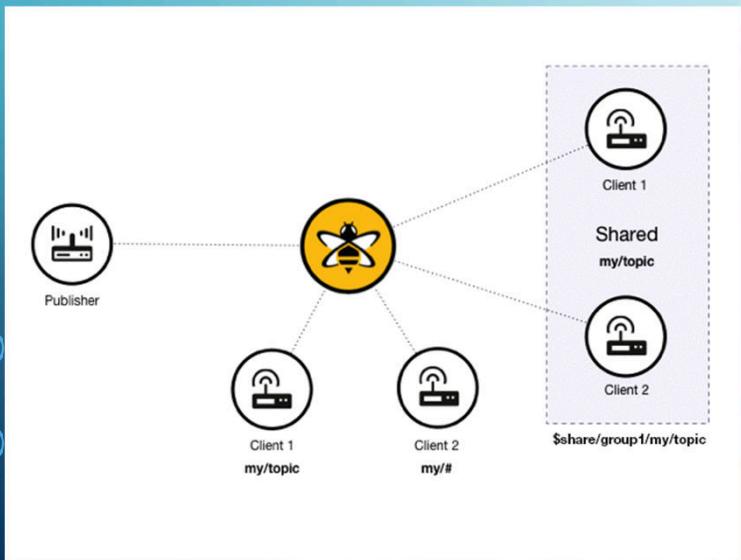
QoS 1(at most once): Assemelha-se ao protocolo UDP, onde não há confirmação de entrega da mensagem e quem envia não tem obrigação de manter a mensagem armazenada para futuras retransmissões.

QoS 2 (at least once): Aqui temos a confirmação de entrega de uma mensagem. Quem envia também mantém a cópia da mensagem em caso de time out. Quando a mensagem chegar ao destino, o destinatário envia uma mensagem ao emissor confirmado o recebimento.

QoS 3 (exactly once): Por último, garante que a mensagem seja entregue exatamente 1 vez, com o envio da confirmação de recebimento e a confirmação da confirmação do recebimento. Resumindo, existe confirmação nos 2 sentidos para tudo que é trafegado. Enquanto uma mensagem não é confirmada, ela é mantida.

CARACTERÍSTICAS

- Um cliente pode fornecer um **LWT (Last Will and Testament)** da primeira vez que ele se conectar com um broker. Se tal cliente se desconectar da rede, o broker envia uma mensagem de broadcast aos outros clientes. Usado geralmente para saber quando um dispositivo fica offline em uma rede.

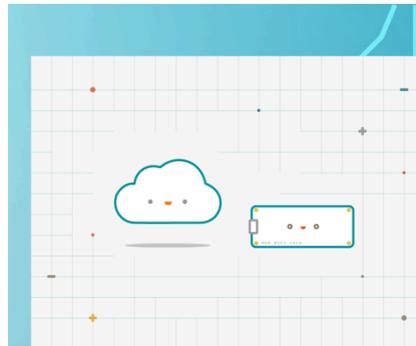


Segurança:

- ✓ Mensagem Criptografada
- ✓ Segurança por Autenticação
- ✓ Segurança MQTT pelo ACL (Access Control List)



Name	Broker Address	TCP Port	TLS Port	WebSocket Port	Message Retention	Persistent Session	Sign Up Required
Eclipse	mqtt.eclipse.org	1883	N/A	80, 443	YES	YES	NO
Mosquitto	test.mosquitto.org	1883	8883, 8884	80	YES	YES	NO
HiveMQ	broker.hivemq.com	1883	N/A	8000	YES	YES	NO
Flespi	mqtt.flespi.io	1883	8883	80, 443	YES	YES	YES
Dioty	mqtt.dioty.co	1883	8883	8080, 8880	YES	YES	YES
Fluux	mqtt.fluux.io	1883	8883	N/A	N/A	N/A	NO
EMQX	broker.emqx.io	1883	8883	8083	YES	YES	NO



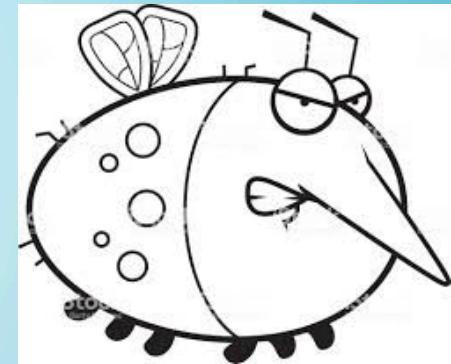
Private MQTT Brokers

Name	TCP Port	TLS Port	WebSocket Port	Message Retention	Persistent Session	QoS Levels	Free Limits	Link
Azure	NO	8883	443	NO	Limited	0, 1	8000 messages/day	Link
AWS	NO	8883	443	NO	Limited	0, 1	250,000/month	Link
CloudMQTT	Custom Port	Custom Port	Custom Port	NOT SURE	YES	0, 1, 2	5 Connections & 10 Kbit/s	Link



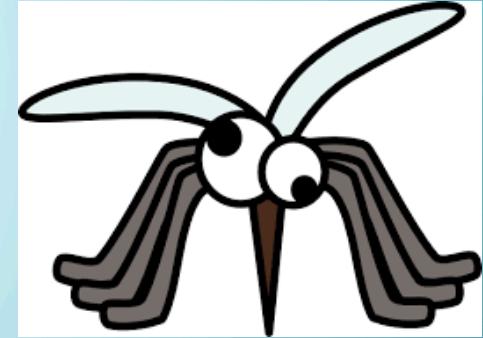
VANTAGENS

- Super leve
- Necessita de pouco hardware
- Baseado em TCP/IP com payload menor que HTTP
- Open source
- Flexível e simples
- Desenvolvido para sistemas de comunicação instáveis e críticos
- Controle de desconexão com notificação
- Criptografia da conexão e dos dados



DESVANTAGENS

- Não é síncrono
- Foco na entrega não na velocidade
- Não existe histórico (ou armazenamento) dos valores
- Alto consumo no subscriber (keep alive)
- Muitos subscribers conectados sobrecarregam o broker
- Todas as informações devem passar pelo broker, se ele falha tudo para (mas já existem brokers redundantes).

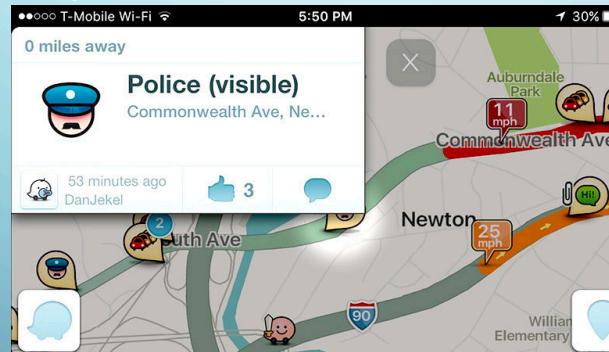


USO FORA DO ESCOPO

Sincronismo é importante?



Velocidade de transmissão é importante?



Capacidade de tráfego é importante?



OBRIGADO!

FATEC Pompéia

Disciplina : Internet das Coisas
Prof. João Ricardo Favan
Aluno
Cássio Félix de Moura