

MQTT

Gabriel R. Caldas de Aquino

Universidade Federal do Rio de Janeiro

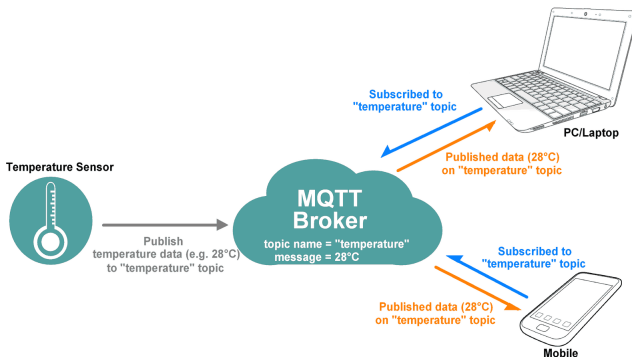
gabriel@labnet.nce.ufrj.br

1 de maio de 2018

- Criado em 1999 para conectar oleodutos por conexão via satélite;
- Características:
 - Open Source
 - Simples e fácil de implementar
 - Leve e consome pouca banda
 - Provê QoS
 - Agnóstico de dados: depende de como o payload é estruturado
- Interessante para dispositivos de recursos limitados;
 - Comunicação M2M na IoT;
- Versão atual 3.1.1;

Introdução

- Protocolo de transporte de mensagens cliente servidor do tipo publish/subscribe:
 - Existem tópicos;
 - Clientes publicam em tópicos;
 - Clientes assinam determinados tópicos.



- Tanto publisher quanto subscriber se conectam a um broker
- Clientes escrevem em um tópico
- Broker recebe as mensagens e as distribui
- Se um tópico receber uma nova mensagem, o broker envia aos clientes

O broker

MQTT broker

O broker é o responsável por receber mensagens, filtrar, decidir quem está interessado e enviar a todos os clientes inscritos.

- Parte principal do Publish/Subscribe
- Mantém a sessão dos clientes (assinaturas e mensagens perdidas)
- Autenticação e autorização de clientes
- É geralmente extensível: código aberto

É importante que:

O Broker seja *escalável*, *integrável*, *monitorável* e *tolerante a falhas*.

Exemplos de brokers

- Eclipse Mosquitto: <https://mosquitto.org/>
 - Open source (EPL/EDL licensed)
 - Implementa o protocolo MQTT ver. 3.1 e 3.1.1.
- Hivemq: <https://www.hivemq.com/>
 - Compatível com MQTT 3.1 e 3.1.1
 - Solução mais completa
- Lista: <https://github.com/mqtt/mqtt.github.io/wiki/brokers>

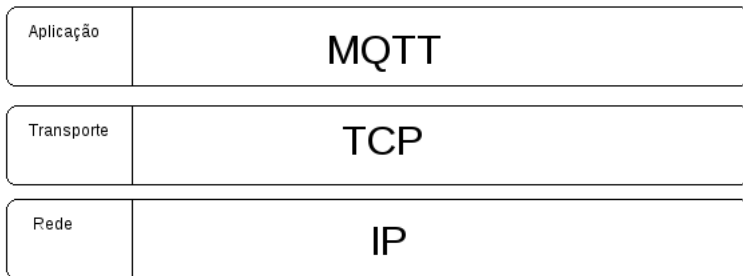
Cliente MQTT

Qualquer dispositivo que execute as bibliotecas do MQTT e está conectado a um broker MQTT em qualquer tipo de rede.

- Basicamente qualquer dispositivo conectado a um broker:
- Dispositivo pequeno e com recursos limitados conectado por uma rede sem fio.
- Computador típico executando um cliente MQTT.

MQTT e a Pilha TCP/IP

- MQTT funciona sobre a pilha TCP/IP
- TCP/IP porta 1883



Formato do pacote de controle MQTT

- O pacote consiste em até três partes (na seguinte ordem):

Figure 2.1 – Structure of an MQTT Control Packet

Fixed header, present in all MQTT Control Packets
Variable header, present in some MQTT Control Packets
Payload, present in some MQTT Control Packets

Header Fixo MQTT

- O Header Fixo do pacote MQTT é dividido em três pedaços:

Figure 2.2 - Fixed header format

Bit	7	6	5	4	3	2	1	0
byte 1	MQTT Control Packet type				Flags specific to each MQTT Control Packet type			
byte 2...	Remaining Length							

Header Fixo MQTT - Tipos de pacote de controle

- 4-bit unsigned values (byte 4 ao 7):

Table 2.1 - Control packet types

Name	Value	Direction of flow	Description
Reserved	0	Forbidden	Reserved
CONNECT	1	Client to Server	Client request to connect to Server
CONNACK	2	Server to Client	Connect acknowledgment
PUBLISH	3	Client to Server or Server to Client	Publish message
PUBACK	4	Client to Server or Server to Client	Publish acknowledgment
PUBREC	5	Client to Server or Server to Client	Publish received (assured delivery part 1)
PUBREL	6	Client to Server or Server to Client	Publish release (assured delivery part 2)
PUBCOMP	7	Client to Server or Server to Client	Publish complete (assured delivery part 3)
SUBSCRIBE	8	Client to Server	Client subscribe request
SUBACK	9	Server to Client	Subscribe acknowledgment
UNSUBSCRIBE	10	Client to Server	Unsubscribe request
UNSUBACK	11	Server to Client	Unsubscribe acknowledgment
PINGREQ	12	Client to Server	PING request
PINGRESP	13	Server to Client	PING response
DISCONNECT	14	Client to Server	Client is disconnecting
Reserved	15	Forbidden	Reserved

Header Fixo MQTT - Flags

- 4-bit unsigned values (byte 0 ao 3):

Table 2.2 - Flag Bits

Control Packet	Fixed header flags	Bit 3	Bit 2	Bit 1	Bit 0
CONNECT	Reserved	0	0	0	0
CONNACK	Reserved	0	0	0	0
PUBLISH	Used in MQTT 3.1.1	DUP ¹	QoS ²	QoS ²	RETAIN ³
PUBACK	Reserved	0	0	0	0
PUBREC	Reserved	0	0	0	0
PUBREL	Reserved	0	0	1	0
PUBCOMP	Reserved	0	0	0	0
SUBSCRIBE	Reserved	0	0	1	0
SUBACK	Reserved	0	0	0	0
UNSUBSCRIBE	Reserved	0	0	1	0
UNSUBACK	Reserved	0	0	0	0
PINGREQ	Reserved	0	0	0	0
PINGRESP	Reserved	0	0	0	0
DISCONNECT	Reserved	0	0	0	0

Header variável MQTT

Figure 2.3 - Packet Identifier bytes

Bit	7	6	5	4	3	2	1	0
byte 1	Packet Identifier MSB							
byte 2	Packet Identifier LSB							

Table 2.5 - Control Packets that contain a Packet Identifier

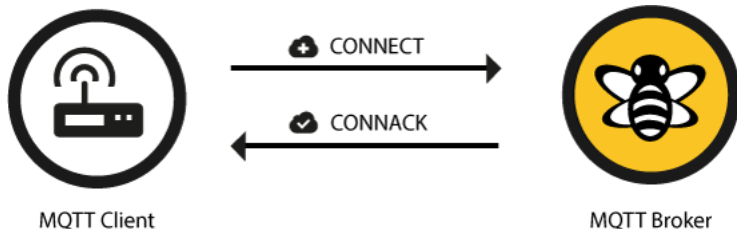
Control Packet	Packet Identifier field
CONNECT	NO
CONNACK	NO
PUBLISH	YES (If QoS > 0)
PUBACK	YES
PUBREC	YES
PUBREL	YES
PUBCOMP	YES
SUBSCRIBE	YES
SUBACK	YES
UNSUBSCRIBE	YES
UNSUBACK	YES
PINGREQ	NO
PINGRESP	NO
DISCONNECT	NO

Table 2.6 - Control Packets that contain a Payload

Control Packet	Payload
CONNECT	Required
CONNACK	None
PUBLISH	Optional
PUBACK	None
PUBREC	None
PUBREL	None
PUBCOMP	None
SUBSCRIBE	Required
SUBACK	Required
UNSUBSCRIBE	Required
UNSUBACK	None
PINGREQ	None
PINGRESP	None
DISCONNECT	None



Estabelecendo uma conexão MQTT

- A conexão MQTT é sempre entre um cliente e o broker
- Não há conexão entre clientes



- Uma vez que a conexão é estabelecida o Broker a mantém aberta
- A conexão só fecha com uma mensagem de disconnect ou quando a conexão é perdida

Formato dos pacotes Connect e Connackdo MQTT

MQTT-Packet: CONNECT 		MQTT-Packet: CONNACK 	
contains:	Example	contains:	Example
clientId	"client-1"	sessionPresent	true
cleanSession	true	returnCode	0
username (optional)	"hans"		
password (optional)	"letmein"		
lastWillTopic (optional)	"/hans/will"		
lastWillQos (optional)	2		
lastWillMessage (optional)	"unexpected exit"		
lastWillRetain (optional)	false		
keepAlive	60		

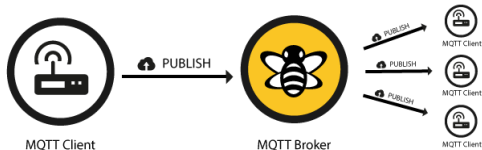
Return Code	Return Code Response
0	Connection Accepted
1	Connection Refused, unacceptable protocol version
2	Connection Refused, identifier rejected
3	Connection Refused, Server unavailable
4	Connection Refused, bad user name or password
5	Connection Refused, not authorized

- Após conectado, o cliente pode publicar no Broker
- MQTT filtra e passa as mensagens por tópicos
 - Cada mensagem *publish* tem que conter um tópico
 - Este tópico é usado pelo broker para encaminhar as mensagens aos clientes interessados.
- Cada mensagem contém os bytes dos dados para transmitir

O MQTT não possui conhecimento acerca dos dados. Ou seja, o uso depende totalmente do caso e de como a mensagem é estruturada. Cabe ao usuário como estruturar o envio dos dados: dados binários, dados textuais ou mesmo XML ou JSON completos.


Publicando no MQTT

Quando um cliente publica, o broker lê a publicação, confirma de acordo com o QoS e, em seguida processa. O processamento é determinar os clientes inscritos e depois enviar aos clientes selecionados.



- Quem publica a mensagem está preocupado apenas em entregar a mensagem ao broker.
- Então, é responsabilidade do broker entregar a mensagem a todos os subscribers
- O cliente que publicou não recebe nenhum feedback dos assinantes

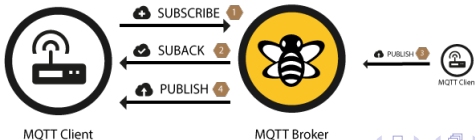
Estrutura do pacote *publish*

MQTT-Packet:	
PUBLISH	
	
contains:	Example
packetId (always 0 for qos 0)	4314
topicName	"topic/1"
qos	1
retainFlag	false
payload	"temperature:32.5"
dupFlag	false

- *Packet Identifier*: É relevante para o QoS
- *Topic Name*: Uma string estruturada hierarquicamente. Ex: casa/sala/luz
- *QoS*: Determina a garantia de entrega ao cliente ou broker
- *Retain-Flag*: Determina se a mensagem será salva pelo broker. Novos clientes recebem a mensagem após a inscrição.
- *Payload*: É o conteúdo da mensagem
- *DUP flag*: indica que esta mensagem é duplicada e é reenviada porque a outra extremidade não confirmou a mensagem original

Assinando e cancelando a assinatura

<p>MQTT-Packet:</p> <h2>SUBSCRIBE</h2> <p>contains:</p> <table><thead><tr><th></th><th>Example</th></tr></thead><tbody><tr><td>packetId</td><td>4312</td></tr><tr><td>qos1 } (list of topic + qos)</td><td>1</td></tr><tr><td>topic1</td><td>"topic/1"</td></tr><tr><td>qos2 }</td><td>0</td></tr><tr><td>topic2</td><td>"topic/2"</td></tr><tr><td>...</td><td>...</td></tr></tbody></table>		Example	packetId	4312	qos1 } (list of topic + qos)	1	topic1	"topic/1"	qos2 }	0	topic2	"topic/2"	<p>MQTT-Packet:</p> <h2>SUBACK</h2> <p>contains:</p> <table><thead><tr><th></th><th>Example</th></tr></thead><tbody><tr><td>packetId</td><td>4313</td></tr><tr><td>returnCode 1 (one returnCode for each topic from SUBSCRIBE,</td><td>2</td></tr><tr><td>returnCode 2 in the same order)</td><td>0</td></tr><tr><td>...</td><td>...</td></tr></tbody></table>		Example	packetId	4313	returnCode 1 (one returnCode for each topic from SUBSCRIBE,	2	returnCode 2 in the same order)	0
	Example																								
packetId	4312																								
qos1 } (list of topic + qos)	1																								
topic1	"topic/1"																								
qos2 }	0																								
topic2	"topic/2"																								
...	...																								
	Example																								
packetId	4313																								
returnCode 1 (one returnCode for each topic from SUBSCRIBE,	2																								
returnCode 2 in the same order)	0																								
...	...																								
<p>MQTT-Packet:</p> <h2>UNSUBSCRIBE</h2> <p>contains:</p> <table><thead><tr><th></th><th>Example</th></tr></thead><tbody><tr><td>packetId</td><td>4315</td></tr><tr><td>topic1 } (list of topics)</td><td>"topic/1"</td></tr><tr><td>topic2</td><td>"topic/2"</td></tr><tr><td>...</td><td>...</td></tr></tbody></table>		Example	packetId	4315	topic1 } (list of topics)	"topic/1"	topic2	"topic/2"	<p>MQTT-Packet:</p> <h2>UNSUBACK</h2> <p>contains:</p> <table><thead><tr><th></th><th>Example</th></tr></thead><tbody><tr><td>packetId</td><td>4316</td></tr></tbody></table>		Example	packetId	4316										
	Example																								
packetId	4315																								
topic1 } (list of topics)	"topic/1"																								
topic2	"topic/2"																								
...	...																								
	Example																								
packetId	4316																								



- Estrutura hierárquica (Lembra pastas de um S.O.) em forma de string UTF-8
- usa a barra (/) como um delimitador
- Tópicos são case-sensitive e devem ao menos conter um caracter válido
- Exemplos:
 - house/room/main-light
 - house/room/side-light
 - house/garage/main-light
 - house/garage/alarm

Diferentes tópicos podem ser assinados usando *wildcards*

- Existem dois tipos:
 - #: *wildcard* de vários níveis
 - +: *wildcard* de um único nível
- Exemplo: house/+ /main-light
 - (cobre) house/room1/main-light
 - (cobre) house/room2/main-light
 - (não cobre) house/room1/side-light
- Exemplo 2: house/#
 - (cobre) house/room1/main-light
 - (cobre) house/room1/alarm
 - (cobre) house/garage/main-light
 - (cobre) house/main-door

Um pouco além: QoS

3 níveis de QoS no MQTT:

- 3 níveis de QoS no MQTT:
 - No máximo uma vez (0) (Best-Effort): Uma mensagem não será confirmada pelo destinatário nem será armazenada e devolvida pelo remetente
 - Pelo menos uma vez (1): É garantido que uma mensagem será entregue pelo menos uma vez ao destinatário. Mas a mensagem também pode ser entregue mais de uma vez.
 - Exatamente uma vez (2): Garante que cada mensagem seja recebida apenas uma vez pela contraparte. É a qualidade mais segura e também a mais lenta do nível de serviço.

Importante:

É importante dizer que quanto maior o nível de QoS, maior é a troca de mensagens e o consquente overhead na rede.

- <http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/mqtt-v3.1.1.html>
- <https://www.hivemq.com/blog/mqtt-essentials/>
- <http://www.steves-internet-guide.com/understanding-mqtt-topics/>

The End