PROTOCOLO MQTT

Especificação e Aplicação

Agenda

1. *Por que MQTT*
2. *Características gerais*
3. *Arquitetura*
4. *MQTT-SN*
5. *Visão resumida em baixo nível*
6. *Implementações*
7. *Aplicações*
8. *Código de exemplo*

Por que MQTT

- Protocolo HTTP tem overhead demais para muitas aplicações
  - Especialmente ao se enviar muitos payloads pequenos
- QoS variável
  - Útil para protocolos de transporte sem garantias de entrega
  - Aplicações em que se pode perder pacotes, menor overhead
- Paradigma diferente
  - Publish/Subscribe
  - Comunicação N para N
  - Modelo requisição/resposta nem sempre é o mais adequado
  - Ideal para comunicação M2M e IoT

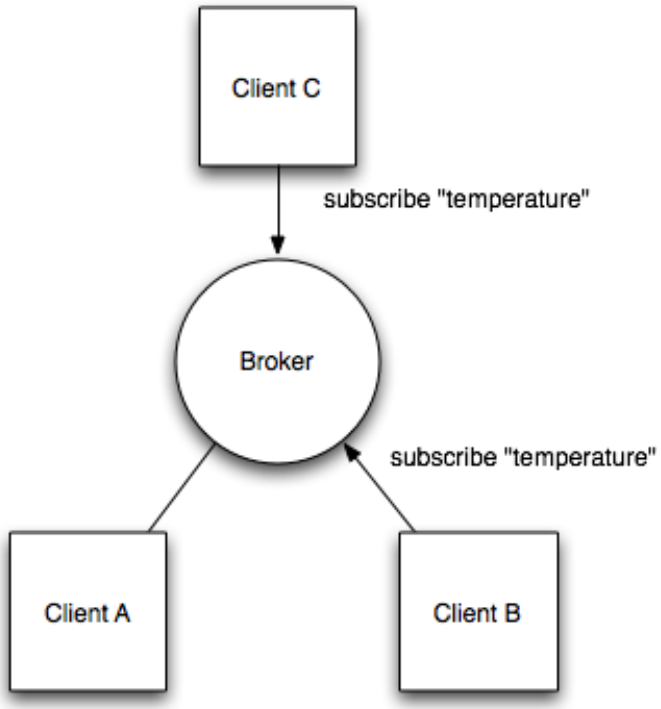
MQTT - Características

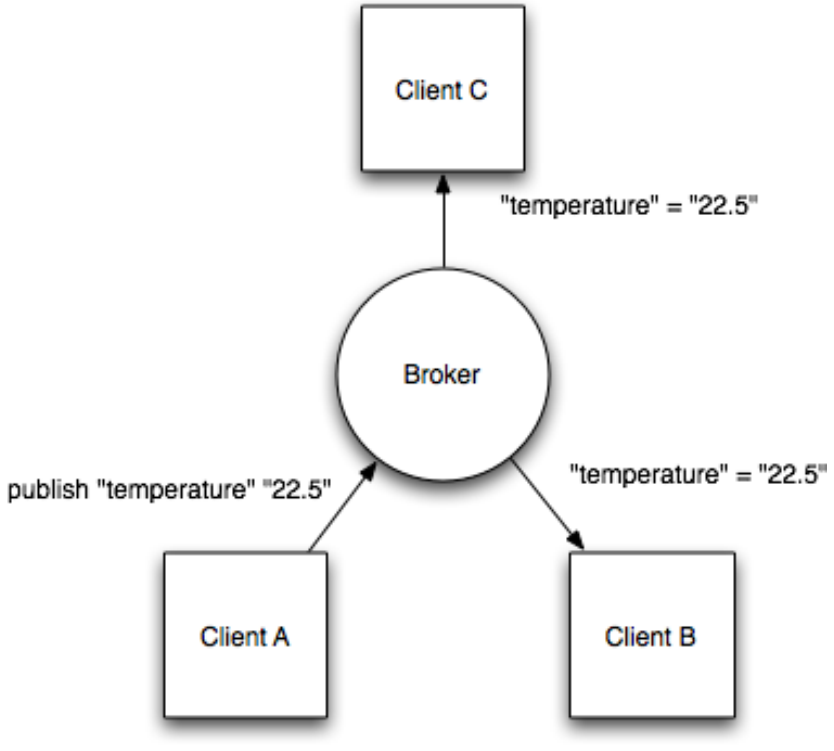
- IBM - Padrão OASIS
- 1999 – Oleoduto no deserto
- Camada de aplicação
  - sobre o protocolo TCP
- Modelo publish/subscribe
  - oposto ao request/response
  - baseado em eventos
  - sem filas
- Originalmente “Message Queue Telemetry Transport”, agora simplesmente MQTT

MQTT - Características

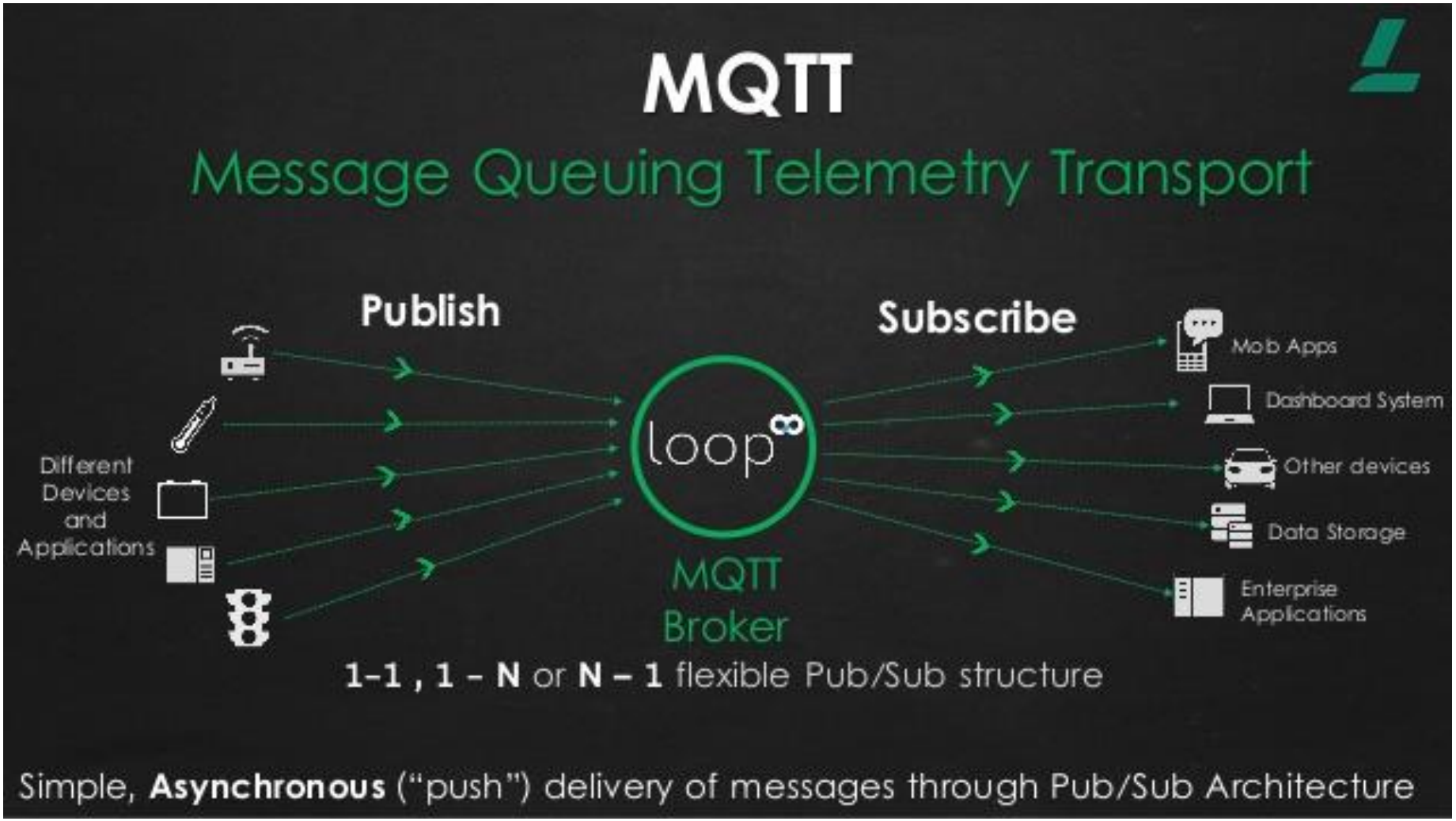
- Broker e tópicos
  - Menor dependência entre produtores e consumidores de dados
  - As duas partes não precisam estar “online” ao mesmo tempo
  - Comunicação N-para-N
- Relativamente leve e simples
- Projetado para usar a banda de maneira eficiente
  - Overhead pequeno
- Pode ser usado sobre Websockets
  - Permite o uso em browsers
- Suporte a autenticação
  - Sem suporte a criptografia; Usar TLS
- Qos Variável
  - At most once
  - Al least once
  - Exactly once

Arquitetura





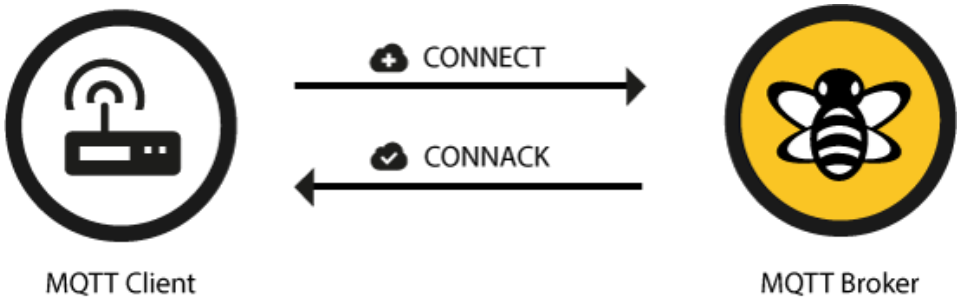
Arquitetura



MQTT-SN

- Possui uma versão alternativa, MQTT-SN, específica para redes de sensores
- MQTT-SN foi idealizada para operar sobre ZigBee (802.15.4) em vez de TCP, é muito próximo de mas adaptado para peculiaridades de RSSF (WSN) – Largura de banda, falhas, tamanho da mensagem, bateria, processamento e memória limitados.
  - Resolve os problemas de nomes dos tópicos (strings longas) e não é necessária uma conexão TCP aberta a todo momento
- Além de clientes (nós SA) e brokers MQTT(servidores), MQTT-SN também inclui Gateways em sua arquitetura

Visão resumida em baixo nível



MQTT-Packet:

CONNECT

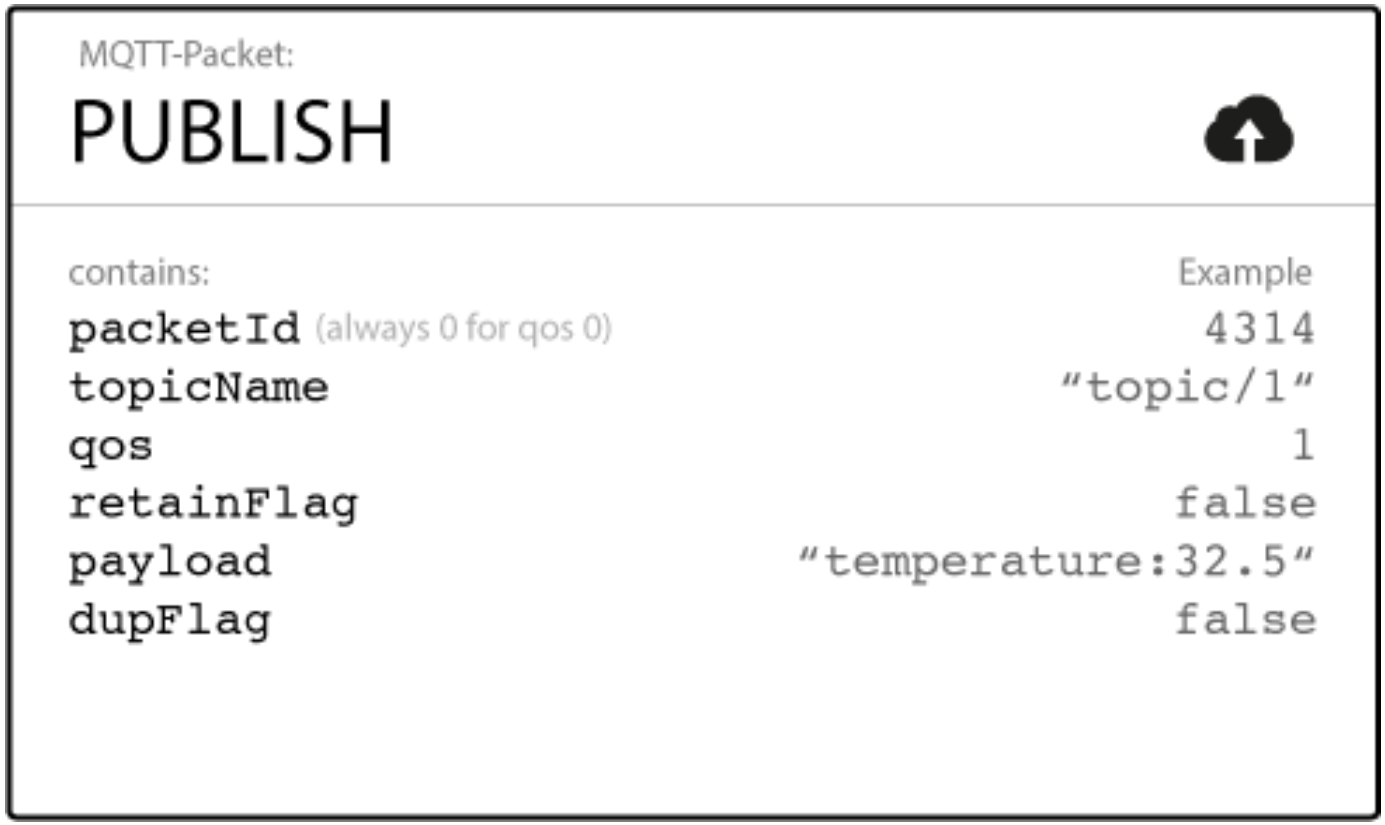
contains:	Example
clientId	"client-1"
cleanSession	true
username (optional)	"hans"
password (optional)	"letmein"
lastWillTopic (optional)	"/hans/will"
lastWillQos (optional)	2
lastWillMessage (optional)	"unexpected exit"
keepAlive	60

MQTT-Packet:

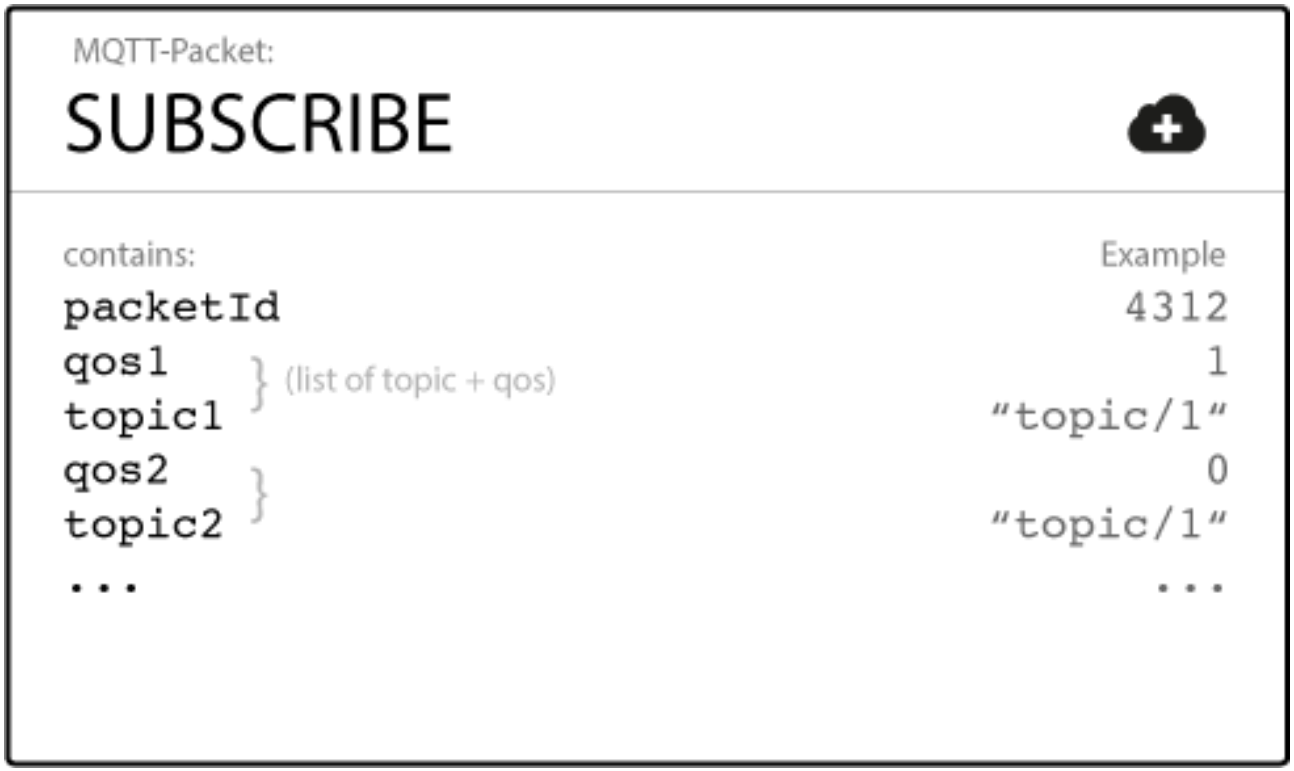
CONNACK

contains:	Example
sessionPresent	true
returnCode	0

Visão resumida em baixo nível

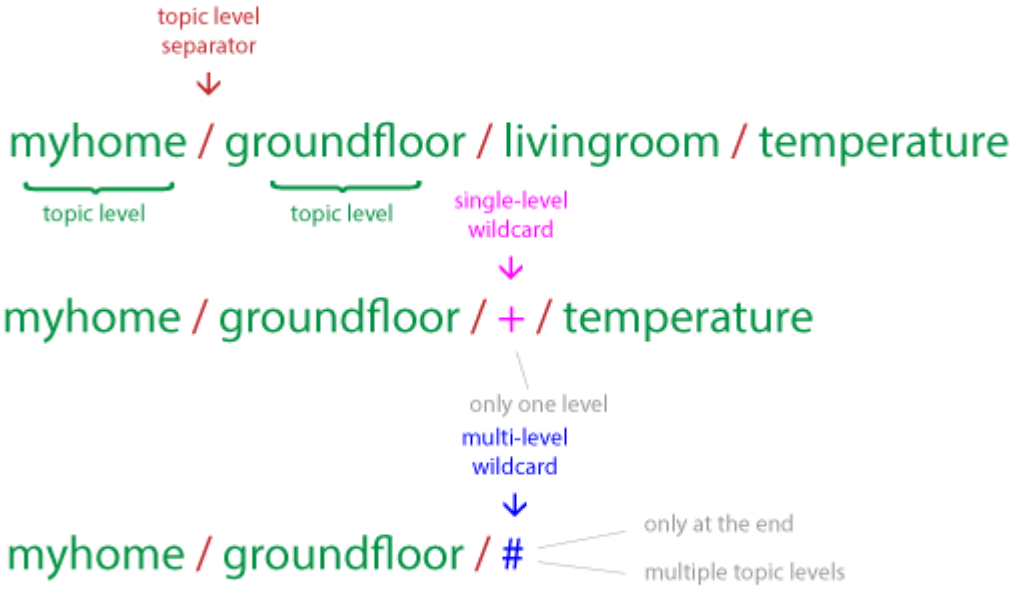


Visão resumida em baixo nível



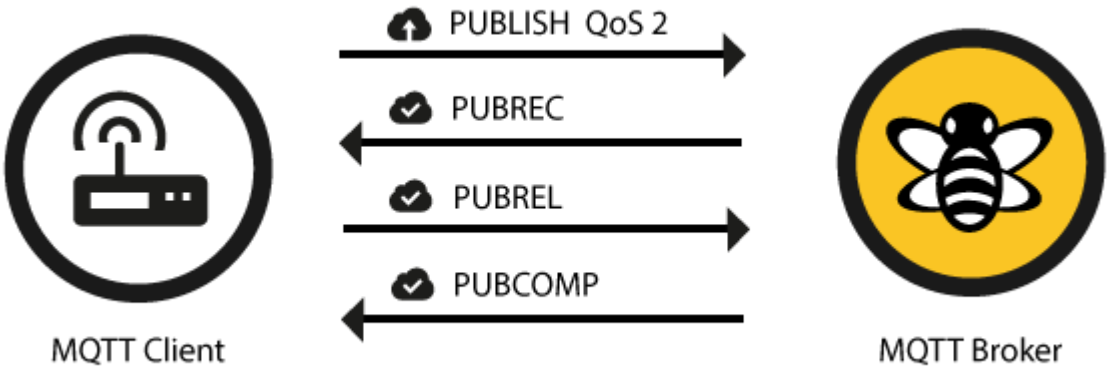
Visão resumida em baixo nível

- SubAck
- Unsubscribe
- UnsubAck
- Formato dos tópicos: string hierárquica



Visão resumida em baixo nível

- PubAck – QoS 1
- PubRec, PubRel e PubComp – QoS 2



Visão resumida em baixo nível

- Opções de sessão
  - Clean Session
  - Persistent Session
- Retained messages
  - Persistir a última mensagem publicada no tópico
- Last Will and Testament
  - Mensagem a ser enviada caso o cliente seja desconectado

Implementações

- Clientes:
  - Projeto Eclipse Paho: Java, C, C++, JavaScript, Lua, Python, Go, Android
  - Outras implementações independentes na mesmas linguagens
  - Implementações independentes em Arduino, Dart, Clojure, Delphi, Erlang, Haskell, Objective-C, OCaml, Perl, PHP, Prolog, Ruby
- Dezenas de Brokers: HiveMQ, RabbitMQ, Mosquito, Mosca (módulo Node), IBM Websphere

Aplicações

- Facebook Messenger (variação)
- Amazon AWS IoT
- IoT em geral
- M2M em geral

Exemplo de cliente (código)

```
public class Publisher
{
    public static final String BROKER_URL = "tcp://broker.mqttdashboard.com:1883";
    private MqttClient client;

    public Publisher()
    {

        String clientId = Utils.getMacAddress() + "-pub";
        try
        {
            client = new MqttClient(BROKER_URL, clientId);
        }
        catch (MqttException e)
       {
           e.printStackTrace();
           System.exit(1);
        }
    }
}
```

Código

```
MqttConnectOptions options = new MqttConnectOptions();
options.setCleanSession(false);
options.setWill(client.getTopic("home/LWT"),
"I'm gone".getBytes(), 2, true);

client.connect(options);
```

Exemplo de cliente (código)

```
public static final String TOPIC_TEMPERATURE = "home/temperature";

//...
while (true)
{
    publishBrightness();
    Thread.sleep(500);
    publishTemperature();
    Thread.sleep(500);
}
//...

private void publishTemperature() throws MqttException {
    final MqttTopic temperatureTopic = client.getTopic(TOPIC_TEMPERATURE);

    final int temperatureNumber = Utils.createRandomNumberBetween(20, 30);
    final String temperature = temperatureNumber + "°C";

    temperatureTopic.publish(new MqttMessage(temperature.getBytes()));
}
```

Dúvidas?

- Obrigado!

Referências

- MQTT and CoAP, IoT protocols

Eclipse Newsletter http://www.eclipse.org/community/eclipse_newsletter/2014/february/article2.php

- MQTT 101 – How to Get Started with the lightweight IoT Protocol http://www.hivemq.com/blog/how-to-get-started-with-mqtt
- IBM developerWorks http://www.ibm.com/developerworks/br/cloud/library/cl-bluemix-arduino-iot2/#download
- MQTT Version 3.1.1. Editado por Andrew Banks and Rahul Gupta. 2014. OASIS Standard. http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html.