



MQTT

- ▶ MQTT es un protocolo de conectividad Maquina-a-Maquina (M2M)/Internet de las Cosas (IoT).
- ▶ Diseñado para ser un protocolo de transporte de mensajes mediante publicación/suscripción extremadamente ligero.
- ▶ Es útil para conexiones en ubicaciones remotas donde se requiera una huella de código muy pequeña y/o el ancho de banda sea un recurso muy valioso.

# MQTT: MQ Telemetry Transport

# Propósito de MQTT

- ▶ Comunicar dispositivos y aplicaciones heterogéneas
- ▶ Buena elección para escenarios en los que movemos mensajes



- ▶ Protocolo de Capa de Aplicación (modelo OSI)
- ▶ Funciona sobre TCP (1883/8883)
- ▶ Modelo publicación/suscripción
- ▶ Diseñado para dispositivos con pocos recursos
- ▶ Strings codificados en UTF-8
- ▶ Acepta cualquier tipo de carga útil (payload)
- ▶ Mantiene la conexión TCP abierta (keep-alive)
- ▶ QoS

**De un  
vistazo...**

# Modelo de Mensajería Publicación/Suscripción

- ▶ Los clientes se suscriben a *temas (topics)* (SUSCRIPCIÓN)
- ▶ Los publicadores envían mensajes a un topic específico (PUBLICACIÓN)
- ▶ Un servidor *bróker* se encarga del enrutado de mensajes

# Modelo de Mensajería Publicación/Suscripción



## Ventajas

Mejora la escalabilidad y permite una topología de red dinámica  
Desacopla las aplicaciones



## Inconvenientes

Mismo tipo de vulnerabilidades de seguridad que el modelo Cliente/Servidor





# Modelo de Mensajería Publicación/Suscripción

- ▶ Los topics forman una estructura jerárquica
  - ▶ p.ej. *edificio-c/sensores/ID201/temperatura*
- ▶ Podemos suscribirnos a varios topics simultáneamente
  - ▶ Suscripción multinivel: #
    - ▶ p.ej. *edificio-c/sensores/#* hace match con *edificio-c/sensores/ID201*, *edificio-c/sensores/ID391/humedad*
  - ▶ Suscripción de un nivel: +
    - ▶ p.ej *edificio/sensores/+/temperatura* hace match con *edificio-c/sensores/ID201/temperatura* y *edificio-c/sensores/ID391/temperatura*, pero no con *edificio-c/sensores/ID391/humedad*



# Broker (de mensajes)

- ▶ Autentica a los clientes
- ▶ Valida, transforma y enruta mensajes
- ▶ Realiza enrutado de mensajes basado en topic
- ▶ Cachea mensajes para su entrega posterior (Ultima Voluntad, flag RETAIN)
- ▶ Bridges: enlazar brokers entre ellos



# Formato del Mensaje


Cabecera Fija (2 bytes) +  
Cabecera Variable + Payload

# QoS

- ▶ Se especifica en cada mensaje
- ▶ QoS 0: se entrega COMO MAXIMO una vez. Es posible que no se entregue
- ▶ QoS 1: se entrega AL MENOS una vez. Es posible que se entregue por duplicado
- ▶ QoS 2: se entrega EXACTAMENTE una vez. Se garantiza que llegará una vez el mensaje
- ▶ En la mayoría de casos es suficiente QoS 0 por rendimiento, pero depende del escenario concreto (QoS 2 requiere muchos más paquetes TCP que QoS 0)

# Retain, Ultima Voluntad y Testamento (LWT)

- Retain: flag que podemos activar en cada mensaje. Indica al bróker que almacene ese mensaje y que lo envíe a cada cliente que se suscriba (solo puede haber un mensaje retenido por topic)
- Ultima Voluntad y Testamento (*Last Will and Testament, LWT*): mensaje que envía el bróker cuando detecta la desconexión inesperada de un cliente (propiedad del cliente activada al conectar al bróker)

| 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" |
| lastWillRetain (optional)   | false             |
| keepAlive   | 60                |

