Universitat Oberta de Catalunya

Implementación del protocolo MQTT-S sobre IEEE 802.15.4e en plataformas OpenMOTE

Javier López Molinero
Máster Universitario en Ingeniería de Telecomunicación UOC-URL
Telemática
Junio 2018

INDICE

- Objetivos y motivación.
- MQTT-S
- OpenWSN
- Arquitectura
- Aplicaciones
- Pruebas
- Conclusiones

Objetivos y motivación

- El objetivo es el desarrollo de un interfaz de comunicaciones MQTT dentro del sistema operativo OpenWSN, para que pueda ser integrado dentro de los OpenMotes.
- Desarrollo de la primera aplicación de ejemplo del interfaz MQTT.
- Desarrollo de un Gateway de MQTT-S.
- Desarrollo de un Suscriptor del Broker para el Topic de ejemplo.
- Representación de los resultados de la aplicación de ejemplo en un Grafana.

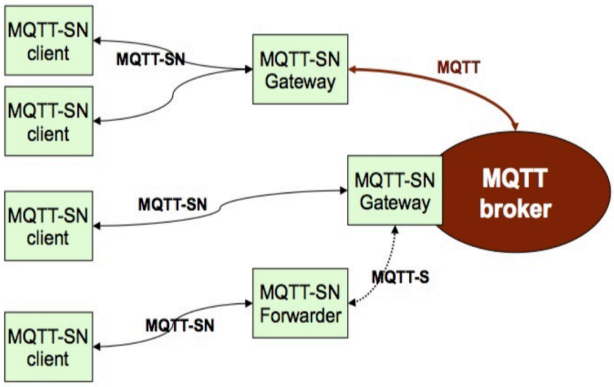
MQTT-S (I)

- MQTT-S o MQTT-SN es una variante de MQTT.
- Utiliza el protocolo de transporte UDP para funcionar, por lo tanto es "no orientado a conexión".
- Este protocolo trata de funcionar en la medida de lo posible como MQTT.
- Utiliza mensajes extra del tipo ACK para suplir a TCP.
- Admite QoS 0,1,2 pero también admite publicación especial 3 y -1.
- Los clientes MQTT-S son capaces de descubrir los Gateways.
- Es necesario el uso de una pasarela de UDP a TCP.
- Topología en estrella.

MQTT-S (II)

Este es el esquema básico de funcionamiento de una arquitectura MQTT-S, se puede ver que existe un paso intermedio entre el Cliente y el broker, que transforma los protocolos.

En la figura, el protocolo que aparece en la parte inferior no se refiere a MQTT-Sensors, sino a MQTT-Secure.

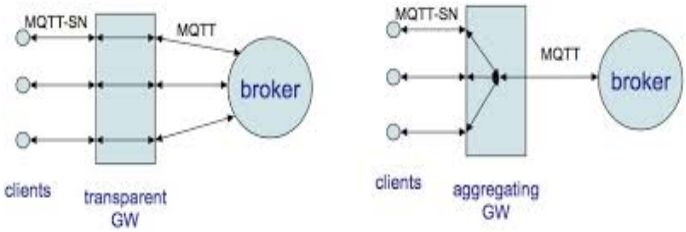


MQTT-S (III)

Gateway MQTT-S puede funcionar de dos maneras diferentes:

- Transparente: Cada cliente tiene su propia conexión frente al broker.
- Agregada: Todos los clientes comparten la misma conexión con el broker.

En el caso de esta arquitectura, funciona de forma agregada.



OpenWSN (I)

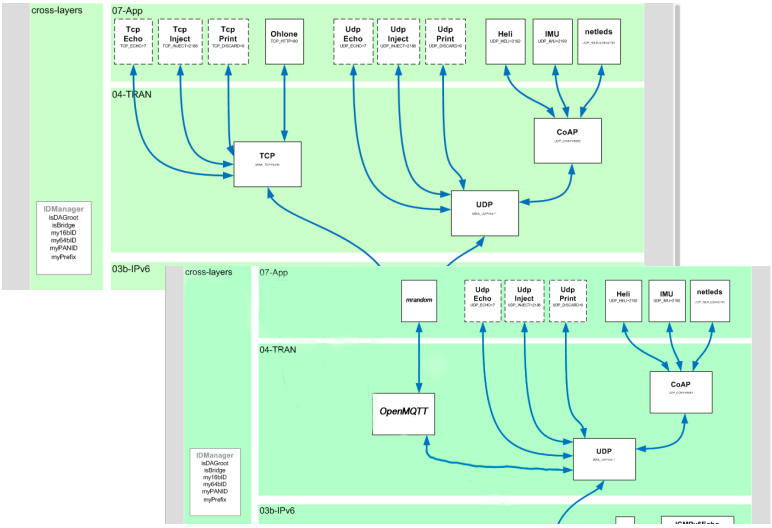
OpenWSN es un sistema operativo de carácter embebido y open-source desarrollado por alumnos de la universidad de Berkeley.

Este sistema operativo se caracteriza por su ligereza. Esta desarrollado en c y python. Utiliza la plataforma Openvisualizer creada expresamente para el uso de estas firmwares de OpenWSN.

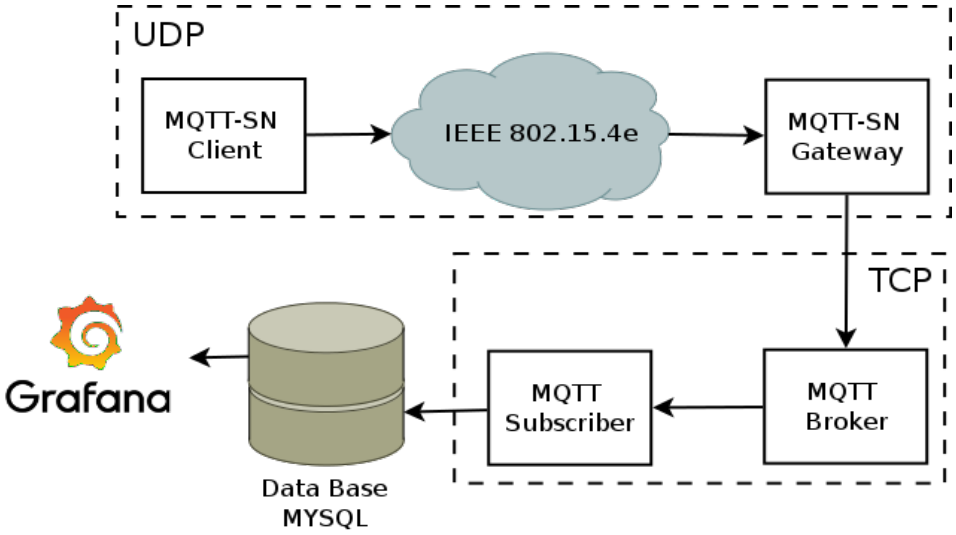
OpenWSN (II)

Analizando el esquema de la estructura del sistema operativo OpenWSN, que se representa en la figura superior se puede observar la modificación realizada frente a la figura inferior correspondiente al nuevo diseño.

Toda comunicación TCP desaparece y aparece un interfaz **openmqtt**, al que se conecta directamente una aplicación llamada **mrandom**.



Arquitectura (I)

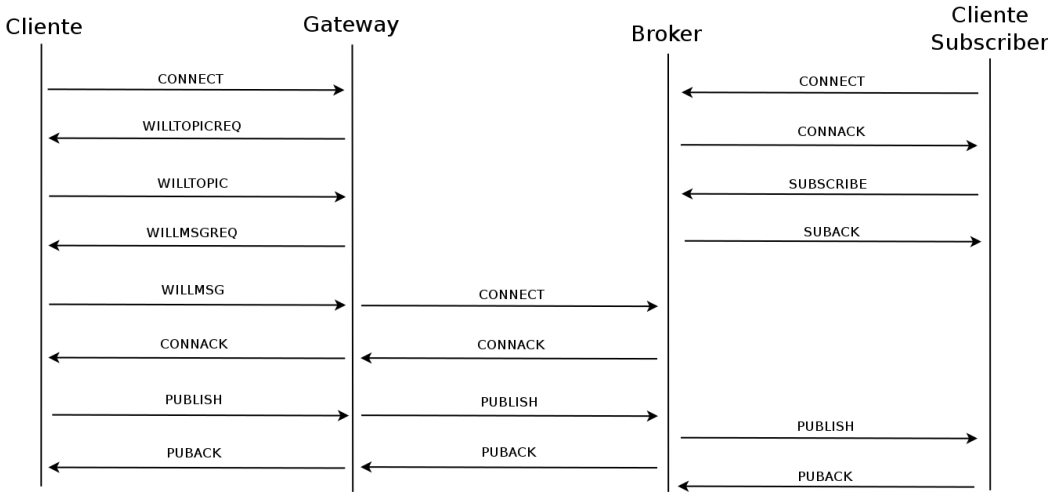


Arquitectura (II)

- Esta arquitectura ha sido diseñada de esta manera como primer paso de ejemplo, para el desarrollo de aplicaciones MQTT en el sistema operativo OpenWSN.
- Ha sido necesario añadir un Gateway de pasarela para transformar la comunicación de MQTT-S a MQTT.
- Se decidió no utilizar ningún portal en la nube para registrar los datos obtenidos en la aplicación, porque es un primer paso para el desarrollo.

Arquitectura (III)

Diagrama de secuencia completo de la arquitectura diseñada:



Aplicaciones (I)

En esta arquitectura se han desarrollado cuatro aplicaciones con el lenguaje C y ha sido necesario configurar tres más de terceros (Broker Mosquitto, Mysql-Server, Grafana). Las aplicaciones desarrolladas son las siguientes:

- **Aplicación Cliente**: Gestiona el interfaz y envía datos.
- **Interfaz openmqtt**: Interfaz transparente que gestionan los clientes.
- **Aplicación Gateway**: Transforma MQTT-S a MQTT
- **Aplicación Suscriptora**: Recibe datos del Broker y los gestiona para su muestra y análisis.

Aplicaciones (II)

Aplicación Cliente "**mrandom**":

- Aplicación simple que mediante un **timer**, envía un valor aleatorio cada cierto tiempo.
- Utiliza el interfaz **openmqtt**.
- Para enviar datos controla cuando tiene radio y cuando es un Mote DagRoot.
- Utiliza el Topic "**Pruebas**" para difundir los datos al broker.

Aplicaciones (III)

Interfaz openmqtt:

- Utiliza la librería **openudp** del sistema operativo.
- Encapsula la información que se quiere enviar en una capa del protocolo MQTT-S y la envía.
- Utiliza el proyecto **paho.embeded.mqtt-sn** adaptándolo al proyecto mediante la librería **openmqtt** creada para este proyecto.

Aplicaciones (IV)

Aplicación Gateway:

- Aplicación monolítica que adapta los protocolos MQTT-S y MQTT.
- Usa la librería **paho.embeded.mqtt-sn** y **paho.mqtt.c**.
- No funciona de forma asíncrona en esta versión.
- Solo funciona con un cliente, debido a ese funcionamiento síncrono de la versión inicial.

Aplicaciones (V)

Aplicación Suscriptora:

- Usa la librería **paho.mqtt.c** y **libmysql**.
- Recibe datos del Broker y los sube a una base de datos **Mysql**.
- Es una aplicación monolítica que funciona de forma asincrona contra el Broker.
- Se suscribe al Topic "**Pruebas**".
- Se añade una marca de tiempo a los datos insertados en la base de datos.

Aplicaciones (VI)

Aplicaciones de terceros:

Aplicación Broker de MQTT, que no ha necesitado configuración especifica. Se han utilizado las especificaciones de origen. **Puerto 1883**.

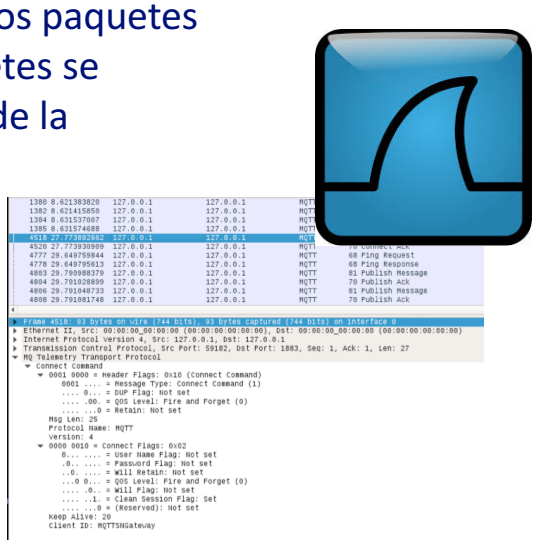
Aplicación servidor de base de datos, que ha necesitado la configuración de la base de datos, mediante un **script sql**.

Aplicación que muestra datos de una fuente de manera amigable, ha habido que configurar un **dashboard** personalizado, con una fuente de datos **Mysql**.

Pruebas (I)

Las pruebas consisten en el análisis de los paquetes enviados a través de la red. Estos paquetes se muestran en el diagrama de secuencia de la arquitectura.

Para lo que se utiliza el analizador de paquetes **Wireshark**, que proporciona la información suficiente para poder analizar los mensajes.



Pruebas (II)

Es necesario tener unas métricas que validen la arquitectura diseñada, las utilizadas han sido:

- Analizar que el Topic “Pruebas” se registra tanto en el Gateway como en el Broker.
- Comprobar que el dato enviado llega y se transmite correctamente a través de la red.
- Comprobar que la secuencia de mensajes se corresponde con los mensajes del protocolo.

Conclusiones

Habiendo verificado todo el protocolo, se puede concluir que como primer paso para el desarrollo, cumple perfectamente con su cometido, siendo el primer ejemplo para OpenWSN. Esta arquitectura ofrece las siguientes ventajas:

- Facilidad de desarrollo de aplicaciones MQTT sobre OpenWSN gracias a la abstracción que ofrece el interfaz.
- Control de errores a través del protocolo, MQTT-S pretende parecerse a un protocolo TCP orientado a conexión.
- Debido a la posibilidad de hacer paquetes de datos de 2 Bytes, permite una gran cantidad de clientes simultaneos multiplexando un canal.