Comentario técnico: CTC-087
Componente: **MQTT**
Autor: Sergio R. Caprile, Senior R&D Engineer

Revisiones	Fecha	Comentarios
0	03/04/20	

En el CTC-086 charlamos sobre la “Internet de *nuestras* cosas” y propusimos una solución escalable basada en MQTT, a la vez que recomendamos recurrir a los grandes proveedores para una red de envergadura. Vamos a enfocarnos aquí en MQTT para quienes deseen conocer más o emplearlo directamente. Procederemos entonces a describir brevemente qué es y cómo opera este protocolo; cuya versión corriente es MQTT 5 y tal vez la más difundida es MQTT 3.1.1. Nos centraremos en los aspectos comunes a ambas versiones.

Breve descripción

MQTT (MQ[1] Telemetry Transport) es desarrollado a fines de los años '90 con el fin de resolver un problema de reutilización de datos y optimizar su entrega; particularmente en redes y sistemas donde el ancho de banda es escaso y por ende costoso.

El diseño de MQTT responde al paradigma *publish/subscribe*, *publicar/suscribirse*. En este modelo arquitectónico se hace una división entre generadores de contenidos, y consumidores de éstos; se trata sin embargo de una división funcional abstracta, dado que un dispositivo físico puede ser a la vez generador y consumidor.

Con esta idea, los generadores de contenidos publican dicha información en un sitio central denominado *broker*, mientras que los consumidores de información toman dichos contenidos de este broker. No existe conexión directa entre generadores y consumidores sino sólo entre cada uno de éstos y el broker.

Los contenidos se dividen en unidades temáticas llamadas *tópicos;* cada mensaje, cada transacción iniciada por un generador que publica información, lleva asociada el nombre de un tópico en el cual será publicado. En teoría pueden existir infinitos tópicos, pero lo interesante es que existan sólo los necesarios para que la información pueda ser recopilada por los interesados en obtenerla. Éstos, a su vez, se subscriben a los tópicos de su interés y reciben transacciones iniciadas por el broker indicando el tópico del mensaje y proveyendo el contenido publicado.

Los nombres de los tópicos deben entonces ser acordados entre generadores y consumidores, o en nuestro ámbito diseñados junto con la red.

Esta arquitectura, entonces, desacopla a generadores y consumidores, facilitando el diseño de la red. Distribuye además entre los consumidores la tarea de acceso a la información; relegando al sitio central, el *b*roker, el mantenimiento de la información y el control de acceso a ésta.

Existen básicamente dos tipos de información y tres niveles de calidad de servicio.

La información puede ser persistente o no. Siempre se entrega al momento de ser generada a quienes están suscriptos y conectados; pero la información persistente, si el broker se configura para soportarla con una base de datos ad hoc, es además entregada a los suscriptores que no estaban conectados al momento de generarse, cuando se conectan. Es decir, persiste hasta tanto se publique otra en dicho tópico.

Los niveles de calidad de servicio responden al hecho de que diferentes tipos de información tienen distinta importancia y necesidades, y es útil asignarles el uso del ancho de banda de manera acorde, eligiendo entre latencia y confiabilidad. Los describimos en más detalle más adelante.

Los dispositivos, tanto productores como consumidores, permanecen siempre conectados al broker. En algunos casos, es necesario actualizar un estado o simplemente indicar que se ha perdido la conexión. Para este fin,

1 “MQ” en el nombre refiere a la serie MQ de servidores de IBM, una de las empresas involucradas en el desarrollo de este protocolo

existe un mensaje llamado “de última voluntad” (*last will*), el cual el dispositivo envía al broker y sólo es distribuido por este último a los suscriptores si se detecta la interrupción de la conexión con el dispositivo.
Finalmente, MQTT asume que es transportado de una forma que preserva el orden de la información y su integridad, es decir, un protocolo como TCP.

Tópicos

A fin de agrupar los tópicos de manera de minimizar la cantidad de suscripciones maximizando la cantidad de información recibida por suscripción, se utiliza un esquema en árbol con comodines.
Por ejemplo (nombres ficticios):

- */edificio/depto1a/pieza/temperatura* y */edificio/depto1a/pieza/humedad* identifican unívocamente datos de temperatura y humedad respectivamente en la pieza del departamento 1a del edificio (al cual a partir de ahora omitiremos referirnos)
- */edificio/depto1a/+/temperatura* identifica a las mediciones de temperatura de todas las habitaciones
- /*edificio/depto1a/pieza/#* identifica a todas las mediciones de la pieza
- */edificio/depto1a/#* a todas las mediciones de todas las habitaciones

El '+' reemplaza a un nivel de profundidad y el '#' a todos los que le siguen. Los nombres de tópicos son asignados libremente y deben acordarse o preferentemente diseñarse.

Calidad de servicio

Tanto cuando se publica un mensaje como cuando se lo recibe, la entrega se hace de acuerdo a un nivel de calidad de servicio especificado en el mensaje y en la suscripción al tópico.

Nivel 0

Corresponde a la más simple de las opciones, un nivel elemental en el que el mensaje se envía y si llega o no depende del azar. La presunción, dado que por lo general tenemos a TCP como transporte es que va a llegar, pero podría haber una desconexión y no sabríamos si llegó o no. Este nivel envía los mensajes “a lo sumo una vez”.

Nivel 1

Es un nivel medio en el que se incorpora una confirmación de recepción para garantizar que esto ha ocurrido. La confirmación es obviamente del receptor inmediato, es decir, del broker a quien envía o de quien recibe al broker (no existen confirmaciones de extremo a extremo, recordemos el paradigma de diseño). Dado que el extravío de una confirmación podría ocasionar un reenvío y consiguiente duplicación del mensaje, este nivel envía los mensajes “al menos una vez”.

Nivel 2

El nivel más alto, el cual incorpora un doble handshake por el que se confirma la operación en ambos sentidos como cierre del procedimiento de envío, garantizando así que el mensaje se recibe “exactamente una vez”. Es decir, el dispositivo informa al broker que recibió la confirmación de parte de éste y sólo cuando ambos saben que el otro recibió el mensaje, termina el proceso.

Forma de utilización

Este protocolo es orientado a conexión, lo primero que hace un dispositivo (luego de establecida la conexión TCP, nos referimos aquí sólo a MQTT) es establecer la conexión con el broker, identificándose e iniciando una sesión. Luego de esta fase el dispositivo realiza lo que debe hacer de acuerdo a su función de generador o consumidor de contenidos (lo cual puede incluir ambas a la vez, pero aquí las explicamos por separado). Transcurrida la operación, ambos, dispositivo y broker, quedan en silencio hasta tanto ocurra un evento.

Generadores de contenidos

- Cuando es necesario informar algo, se publica un mensaje seleccionando un tópico y un nivel de calidad de servicio.
- Cuando no sucede nada y transcurre un determinado tiempo de inactividad, se confirma la sanidad de la conexión mediante un ping.

Consumidores de contenidos

- Establecida la conexión el dispositivo se suscribe a los tópicos de su interés, solicitando un nivel de calidad de servicio. A continuación el broker entrega los mensajes con retención que estuvieran pendientes en dichos tópicos.
- Cuando al broker ingresa un mensaje nuevo en alguno de los tópicos en los que el dispositivo se haya suscripto, el mensaje es enviado al dispositivo.
- Cuando no sucede nada y transcurre un determinado tiempo de inactividad, se confirma la sanidad de la conexión mediante un ping.

Soluciones disponibles para broker

En general es recomendable y preferible utilizar servicios de la Internet. Sin embargo, existen opciones comerciales y open source que podemos instalar en nuestros servidores. Nombramos aquí sólo algunas, tal vez las más conocidas, al menos las que conocemos y hemos usado.

- Servicios MQTT de la Internet
  - CloudMQTT[2], comercial, opción gratis limitada en cantidad de dispositivos conectados
  - HiveMQ Cloud[3], comercial, infraestructura de servicio
  - Eclipse provee un servidor público[4], que podemos utilizar para pruebas de concepto
- Software
  - Mosquitto[5], open source
  - HiveMQ[6], comercial

Soluciones disponibles para dispositivos

En general, la operatoria que describimos para generadores y consumidores de contenidos se agrupa bajo el concepto de *cliente MQTT*. A su vez, las operaciones básicas se agrupan en *libraries*.
De acuerdo a si empleamos un sistema operativo o no, y las características de nuestro entorno, podemos aprovechar determinados clientes y/o libraries. Por ejemplo:

- En Linux existen clientes propios de Mosquitto provistos con el paquete de código del broker, también han sido compilados para otros entornos.
- lwIP[7] provee un cliente MQTT que no requiere de un sistema operativo.
- Muchos RTOS tienen un stack TCP/IP asociado, el cual a su vez dispone de un cliente MQTT.
- Eclipse Paho[8], incluye una serie de clientes y libraries en diversos lenguajes de programación para variados entornos, entre ellos C++ y C en ambientes embedded[9].[10]
- Muchos entornos de desarrollo ya proveen soporte MQTT, algunos propietario y otros basado en alguno de los mencionados. Entre los que utilizamos se encuentran:

2 https://www.cloudmqtt.com/
3 https://www.hivemq.com/cloud/
4 https://mqtt.eclipse.org/
5 https://mosquitto.org/
6 https://www.hivemq.com/
7 stack TCP/IP open source, funciona tanto bajo un RTOS como directamente en *bare metal*. https://savannah.nongnu.org/projects/lwip/
8 https://www.eclipse.org/paho/
9 https://www.eclipse.org/paho/clients/c/embedded/
10 Los clientes no requieren un sistema operativo aunque puede que asuman ciertas facilidades, en particular hemos utilizado solamente la library

    - Espressif IDF[11] (para ESP32)
    - Mongoose-OS[12] (para ESP32 y muchos otros procesadores ARM Cortex-M)
- La página wiki de MQTT tiene mucha más información al respecto.[13]

Aspectos relativos a la seguridad en MQTT

MQTT es un protocolo más cercano a la aplicación, sin embargo no posee demasiada complejidad ni sofisticación. A este respecto, en principio MQTT 3.1.1 contiene nombre de usuario y contraseña en el mensaje de conexión. Los campos son opcionales y el broker debe configurarse para realizar la autenticación. Existe además un campo *client-id* que suele ser utilizado como factor adicional. MQTT 5 incorpora además un esquema de autenticación extensible mediante el cual es posible negociar un protocolo y mecanismo para esta función, como por ejemplo los soportados por TLS.
Dependiendo del broker utilizado, es posible correr MQTT sobre TLS o Websockets; muchos brokers en Internet utilizan puertos TCP diferentes para cada una de estas opciones.
Finalmente, el broker está alojado en la Internet, y los datos con persistencia resultan almacenados. De hecho nada impide que además puedan almacenarse los no persistentes. El almacenamiento atañe al contenido de los mensajes, por lo que si no confiamos en nuestro proveedor y por algún motivo no podemos cambiar a otro en el que sí confiemos, deberemos encriptar la información por fuera de MQTT, es decir, lo que entregamos a MQTT no debe ser legible por quien no posee la clave o claves o certificados. Esto a su vez, nos genera el problema de la distribución y mantenimiento de los mismos.

11 https://docs.espressif.com/projects/esp-idf/en/latest/esp32/
12 https://mongoose-os.com/
13 https://github.com/mqtt/mqtt.github.io/wiki/libraries