MQTT fue creado por el Dr. Andy Stanford-Clark de IBM y Arlen Nipper de Arcom – ahora Eurotech – en 1999. MQTT fue creado como una forma rentable y fiable de conectar los dispositivos de monitorización utilizados en las industrias del petróleo y el gas a servidores empresariales remotos.

Cuando fueron desafiados con encontrar una manera de empujar los datos de los sensores de la tubería en el desierto a los sistemas de control de supervisión y adquisición de datos (SCADA) fuera del sitio, decidieron una topología de publicación/suscripción basada en TCP/IP que sería impulsada por los eventos para mantener bajos los costos de transmisión del enlace satelital.

MQTT es ahora un protocolo abierto que es supervisado por la Organización para el Avance de los Estándares de Información Estructurada (OASIS).

MQTT no es parte de la serie original de IBM MQSeries; sin embargo, a partir de la versión 7.1, está disponible en WebSphere MQ. MQTT fue previamente conocido como el protocolo SCADA, MQ Integrator SCADA Device Protocol (MQIsdp) y WebSphere MQTT (WMQTT), aunque todas estas variaciones han caído en desuso.

Uso del protocolo MQTT

Facebook utiliza MQTT para su aplicación de Messenger, no sólo porque el protocolo conserva la energía de la batería durante la mensajería de teléfono móvil a teléfono, sino también porque el protocolo permite que los mensajes se entreguen de manera eficiente en milisegundos (ms), a pesar de las inconsistentes conexiones de Internet en todo el mundo.

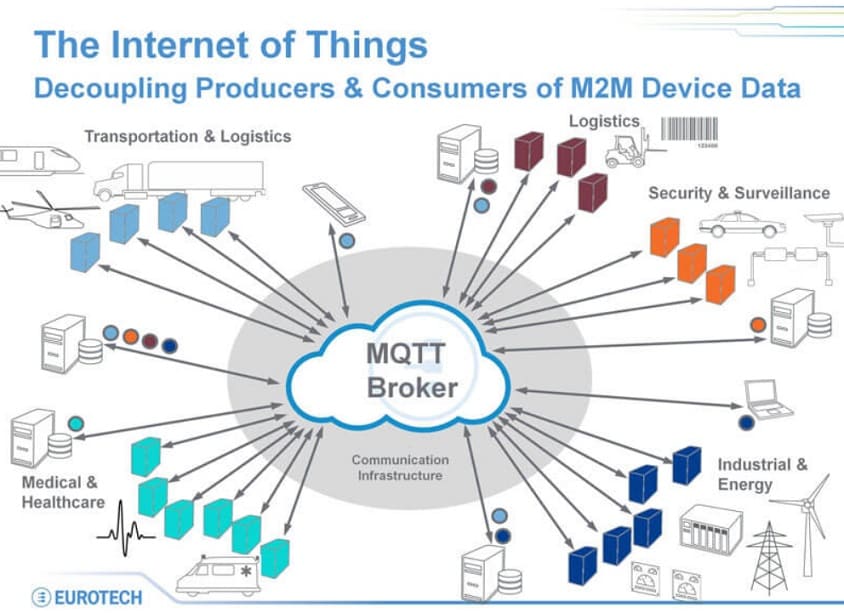
La mayoría de los principales proveedores de servicios en la nube, incluyendo Amazon Web Services (AWS), Google Cloud, IBM Cloud y Microsoft Azure, soportan MQTT.

MQTT es muy adecuado para aplicaciones que utilizan dispositivos M2M e IoT para propósitos tales como análisis en tiempo real, mantenimiento preventivo y monitorización en entornos, incluyendo hogares inteligentes, sanidad, logística, industria y fabricación.

¿Quién utiliza MQTT?

MQTT es usado por muchas compañías importantes, especialmente en los sectores de automoción, industria 4.0, transporte y entretenimiento. MQTT se utiliza para el intercambio de datos entre dispositivos restringidos y aplicaciones de servidor. Mantiene los requerimientos de ancho de banda al mínimo absoluto, maneja redes poco fiables, requiere poco esfuerzo de implementación para los desarrolladores y es, por lo tanto, ideal para la comunicación de máquina a máquina (M2M).

MQTT en IoT



El MQTT es uno de los protocolos más utilizados en relación con la Internet de las Cosas. El MQTT permite que los dispositivos de IoT con recursos limitados envíen o publiquen información sobre un tema determinado a un servidor que funciona como un agente de mensajes MQTT.

El servidor envía la información a los clientes que se han suscrito previamente al tema.

Para un humano, un tema tiene el aspecto de una ruta de archivo jerárquica.

Los clientes pueden suscribirse a un nivel específico de la jerarquía de un tema o utilizar un carácter comodín para suscribirse a varios niveles.

Como ejemplos, las plataformas de IO Carriots, Evrything y ThingWorx soportan el protocolo MQTT.

Protocolos competidores

Otros protocolos de transferencia que compiten con MQTT son los siguientes:

* Constrained Application Protocol (CoAP) es otro protocolo muy adecuado para la IO. El CoAP también utiliza un patrón de comunicación de solicitud/respuesta.
* El Advanced Message Queuing Protocol (AMQP), como el MQTT, utiliza un patrón de comunicación de publicación/suscripción.
* Simple/Streaming Text Oriented Messaging Protocol (STOMP) es un protocolo basado en texto. Sin embargo, STOMP no se ocupa de colas y temas; utiliza una semántica de envío con una cadena de destino.
* Mosquitto es un broker MQTT de código abierto.
* Simple Media Control Protocol (SMCP) es una pila CoAP que se utiliza en entornos embebidos. SMCP también está basado en C.
* SSI (Simple Sensor Interface) es un protocolo de comunicaciones para la transferencia de datos entre una combinación de ordenadores y sensores.
* El Servicio de Distribución de Datos (DDS) para sistemas en tiempo real es un estándar de middleware que puede publicar o suscribir directamente comunicaciones en tiempo real en sistemas embebidos.

Ventajas y desventajas de MQTT

Las ventajas:

1. transmisión de datos eficiente y rápida de implementar debido a que es un protocolo ligero;
2. bajo uso de la red, debido a la minimización de los paquetes de datos;
3. distribución eficiente de los datos;
4. implementación exitosa de la teledetección y el control;
5. entrega de mensajes rápida y eficiente;
6. uso de pequeñas cantidades de energía,
7. reducción del ancho de banda de la red

Las desventajas de MQTT :

* MQTT tiene ciclos de transmisión más lentos en comparación con CoAP.
* El descubrimiento de recursos de MQTT funciona con una suscripción de temas flexible, mientras que CoAP utiliza un sistema estable de descubrimiento de recursos.
* MQTT no está encriptado. En su lugar, utiliza TLS/SSL para el cifrado de seguridad.
* Es difícil crear una red MQTT globalmente escalable.

Desafíos de MQTT: Seguridad, interoperabilidad y autenticación

Debido a que el protocolo MQTT no fue diseñado con la seguridad en mente, el protocolo ha sido tradicionalmente usado en redes de back-end seguras para propósitos específicos de la aplicación. La estructura temática de MQTT puede formar fácilmente un árbol enorme, y no hay una forma clara de dividir un árbol en dominios lógicos más pequeños que puedan ser federados. Esto dificulta la creación de una red MQTT globalmente escalable porque, a medida que el tamaño del árbol de temas crece, la complejidad aumenta.

Otro aspecto negativo de MQTT es su falta de interoperabilidad. Debido a que las cargas útiles de los mensajes son binarias, sin información sobre cómo están codificadas, pueden surgir problemas – especialmente en arquitecturas abiertas donde se supone que diferentes aplicaciones de diferentes fabricantes trabajan sin problemas entre sí.

Como se mencionó anteriormente, MQTT tiene características mínimas de autenticación incorporadas en el protocolo. Los nombres de usuario y las contraseñas se envían en texto claro, y cualquier forma de uso seguro de MQTT debe emplear SSL/TLS, que, desafortunadamente, no es un protocolo ligero.

Autenticar clientes con certificados del lado del cliente no es un proceso simple, y no hay manera en MQTT de controlar quién es el dueño de un tema y quién puede publicar información sobre él, excepto usando medios propietarios fuera de la banda. Esto facilita la inyección de mensajes dañinos en la red, ya sea voluntariamente o por error.

Además, no hay manera de que el receptor del mensaje sepa quién envió el mensaje original, a menos que esa información esté contenida en el mensaje real. Las características de seguridad que tienen que ser implementadas en la parte superior de MQTT en una forma propietaria aumentan la huella del código y hacen que las implementaciones sean más difíciles.

Niveles de calidad de servicio

La calidad de servicio se refiere a un acuerdo entre el remitente de un mensaje y el destinatario del mismo. La QoS definirá la garantía de entrega en referencia a un mensaje específico. La QoS actúa como una característica clave en MQTT, dando al cliente la posibilidad de elegir entre tres niveles de servicio.

Los tres niveles de QoS diferentes determinan cómo se gestiona el contenido mediante el protocolo MQTT. Aunque los niveles más altos de QoS son más fiables, tienen más requisitos de latencia y ancho de banda, por lo que los clientes suscritos pueden especificar el nivel de QoS más alto que les gustaría recibir.

1. El nivel de QoS más simple es el servicio no reconocido. Este nivel de QoS utiliza una secuencia de paquetes PUBLISH; el editor envía un mensaje al agente una vez y el agente pasa el mensaje a los suscriptores una vez. No existe ningún mecanismo que asegure que el mensaje se ha recibido correctamente, y el agente no guarda el mensaje. Este nivel de QoS también puede denominarse como máximo una vez, QoS0 o “fire and forget” (disparar y olvidar).
2. El segundo nivel de QoS es el servicio de acuse de recibo. Este nivel de QoS utiliza una secuencia de paquetes PUBLISH/PUBACK entre el editor y su corredor, así como entre el corredor y los suscriptores. Un paquete de acuse de recibo verifica que se ha recibido el contenido, y un mecanismo de reintento enviará el contenido original de nuevo si no se recibe un acuse de recibo a tiempo. Esto puede dar lugar a que el suscriptor reciba varias copias del mismo mensaje. Este nivel de calidad de servicio también puede denominarse “al menos una vez” o “QoS1”.
3. El tercer nivel de QoS es el servicio asegurado. Este nivel de QoS entrega el mensaje con dos pares de paquetes. El primer par se denomina PUBLISH/PUBREC, y el segundo par se denomina PUBREL/PUBCOMP. Los dos pares aseguran que, independientemente del número de reintentos, el mensaje sólo se entregará una vez. Este nivel de QoS también puede denominarse exactamente una vez o QoS2.

Especificaciones de MQTT

MQTT tiene diferentes especificaciones dependiendo de la versión específica. La versión 5.0 sustituyó a la última versión de MQTT, la versión 3.1.1. Algunas especificaciones más recientes, como las definidas por OASIS, incluyen lo siguiente:

* el uso de patrones de mensajes de publicación/suscripción;
* un mecanismo que puede notificar a los usuarios cuando se producen desconexiones anormales;
* los tres niveles de entrega de mensajes: a lo sumo una vez, a lo menos una vez y exactamente una vez;
* la minimización de la sobrecarga de transporte y los intercambios de protocolos para reducir el tráfico de red; y
* un transporte de mensajería agnóstico que se refiere al contenido de la carga útil.

Más especificaciones se pueden encontrar en el sitio web de OASIS en [este enlace](https://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.html" \o "especificaciones MQTT" \t "_blank).

Actualizaciones de MQTT

El MQTT fue aprobado oficialmente como un estándar OASIS el 28 de octubre de 2015. A finales de enero de 2016, fue aceptado como un estándar de la Organización Internacional de Normalización (ISO). El protocolo está en continua mejora y ahora soporta WebSockets, otro protocolo que permite la comunicación bidireccional entre clientes y corredores en tiempo real. Posteriormente, se han producido versiones destacadas como la v3.1.1 y la v5.0, ambas aprobadas como estándares OASIS. Como ejemplo de algunas de sus actualizaciones, la versión 5.0 incluía un mejor reporte de errores, incluyendo metadatos en los encabezados de los mensajes, suscripciones compartidas, expiraciones de mensajes y sesiones, y alias de temas.

Clientes de MQTT

Debido a que los clientes de MQTT no tienen direcciones como direcciones de correo electrónico, números de teléfono, etc. no es necesario asignar direcciones a los clientes como se hace con la mayoría de los sistemas de mensajería.

Para MQTTv3.1.1 hay software cliente disponible en casi todos los lenguajes de programación y para los principales sistemas operativos Linux, Windows, Mac del proyecto Eclipse Paho.

Para MQTTv5.0 hay un soporte limitado para el software cliente de Eclipse, actualmente sólo hay un cliente C disponible. Aquí hay un enlace a la tabla de comparación de clientes y a la página de descargas.

Actualmente estoy usando un cliente Python de esta página github que soporta v5.

MQTT Brokers o Servidores

El término original era “broker” pero ahora ha sido estandarizado como “Server”. Verás que se utilizan ambos términos. Hay muchos brokers MQTT disponibles que puede utilizar para pruebas y para aplicaciones reales. Hay brokers gratuitos auto alojados, los más populares son Mosquitto y los comerciales como HiveMQ. Mosquitto es un broker MQTT de código abierto gratuito que funciona en Windows y Linux.

Si no deseas instalar y gestionar su propio broker puede utilizar un broker basado en la nube de los proveedores de servicios en la nube como IBM, Microsoft (Azure), etc.

Eclipse tiene un broker público gratuito MQTT y un servidor COAP que también puede utilizar para realizar pruebas. La dirección es iot.eclipse.org y el puerto es 1883 o 8883(SSL).

MQTT sobre WebSockets

Websockets te permite recibir los datos de MQTT directamente en un navegador web. Esto es importante ya que el navegador web puede convertirse en la interfase DE-facto para visualizar los datos MQTT.

El soporte de MQTT websocket para navegadores web es proporcionado por el Javascript MQTT Client.

¿Es seguro el MQTT?

MQTT soporta varias autenticaciones y mecanismos de seguridad de datos.

Es importante señalar que estos mecanismos de seguridad se configuran en el broker MQTT, y depende del cliente cumplir con los mecanismos establecidos.

MQTT está diseñado para permitir una comunicación muy segura. Como un protocolo de capa de aplicación introduce amplias posibilidades de autenticación y autorización de dispositivos. El protocolo de transporte TCP/IP subyacente puede añadir seguridad adicional a través de la encriptación TLS.

¿Es MQTT de código abierto?

MQTT es un protocolo abierto que está estandarizado por OASIS e ISO (ISO/IEC 20922:2016).

¿Cuál es la diferencia entre HTTP y MQTT?

MQTT es un protocolo binario centrado en datos, extremadamente ligero. Debido a su mínima sobrecarga de paquetes, MQTT sobresale en la transferencia de datos a través del cable en comparación con los protocolos centrados en el documento, como HTTP. A diferencia de HTTP, que se basa en un patrón de petición/respuesta, MQTT proporciona una comunicación basada en el push. Este push es posible a través de conexiones TCP permanentes.

¿Cuál es la diferencia entre AMQP y MQTT?

AMQP es un protocolo de mensajería bidireccional, síncrono y de par a par, que tiene altas exigencias de complejidad de implementación y una mayor sobrecarga de la red. Es un protocolo de cable binario, construido como reemplazo del middleware de mensajes con un enfoque principal en la interoperabilidad entre diferentes proveedores a través de características enriquecidas y varios patrones de intercambio.

MQTT es un protocolo binario con la fuerza en la simplicidad de ser ideal para la aplicación móvil de IO y M2M. Proporciona el patrón de mensajería pub/sub y está diseñado para dispositivos con recursos limitados, ancho de banda bajo y redes de alta latencia. MQTT está especificado por el estándar oficial OASIS.

¿Utiliza MQTT WebSockets?

Con la implementación del broker adecuado que soporte WebSockets nativos, MQTT con el 100% de su conjunto de características puede ser utilizado a través de WebSockets. Ya que proveen un canal de comunicación bidireccional, ordenado y sin pérdidas a través de TCP.

¿MQTT utiliza TCP o UDP?

MQTT utiliza TCP. Debido a los requisitos de pedido MQTT sobre UDP no está posible.

¿Requiere MQTT Internet?

Sí, para enviar o recibir mensajes, el cliente MQTT debe establecer una conexión TCP con el broker. Sin embargo, MQTT viene con las características diseñadas específicamente para hacer frente a las conexiones de red inestables, como el buffer de mensajes entrantes del corredor para los clientes desconectados.

¿Funciona MQTT con Apache Kafka?

Sí, MQTT y Kafka pueden ser integrados entre sí. La manera más eficiente de hacerlo es utilizando la Extensión Kafka de HiveMQ Enterprise.

¿Es MQTT independiente?

No. MQTT no es independiente y no proporciona un patrón de solicitud/respuesta. Es un protocolo de mensajería de capa de aplicación de publicación/suscripción que requiere una conexión TCP permanente y transmite mensajes de manera instantánea, por empuje.