Java Message Service (JMS)

UNIVERSIDAD TECNICA ESTATAL DE QUEVEDO

APLICACIONES DISTRIBUIDAS

INTEGRANTES:

BURBANO PARRAGA CRISTHIAN

SINCHIGUANO SALTOS LESLIE

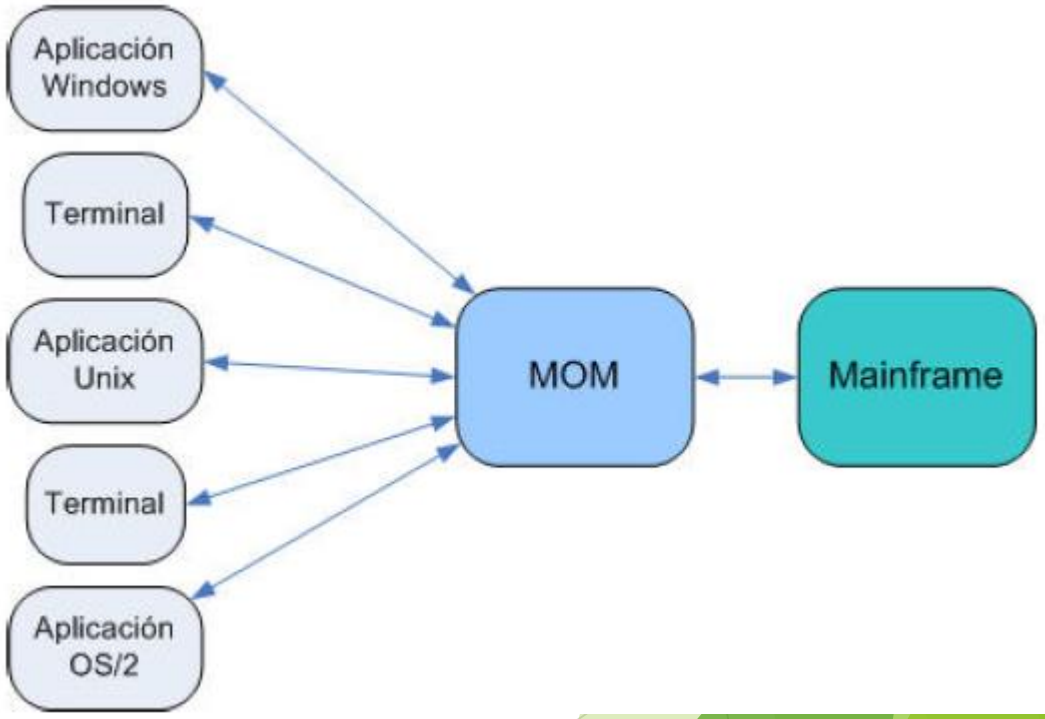
TORRALES PERALTA JANER

2020-2021

Midleware Orientado a Mensajes (MOM)

El MOM es un categoría de software para la intercomunicación de sistemas que ofrece una manera segura, escalable, confiable y con bajo acoplamiento. Los MOMs permiten la comunicación entre aplicaciones mediante un conjunto de APIs ofrecidas por cada proveedor y lenguaje, así pues, tendremos un API propietaria y diferente por cada MOM existente.

La idea principal de un MOM es que actúa como un un mediador entre los emisores y los receptores de mensajes. Esta mediación ofrece un nuevo nivel de desacoplamiento en la mensajería empresarial. Así pues, un MOM se utiliza para mediar en la conectividad y la mensajería, no solo entre las aplicaciones y el mainframe, sino de una aplicación a otra.



¿Qué es JMS?

Una forma común de que los programas Java pueden:

- crear, enviar, recibir
- leer los mensajes empresariales distribuidos

Comunicación acoplada libremente

Mensajes asíncronos

Entrega fiable

- Se garantiza que un mensaje se entrega una vez y sólo una vez.

Una aplicación del JMS

Clientes del JMS

- Los programas Java que envían/reciben mensajes

Mensajes

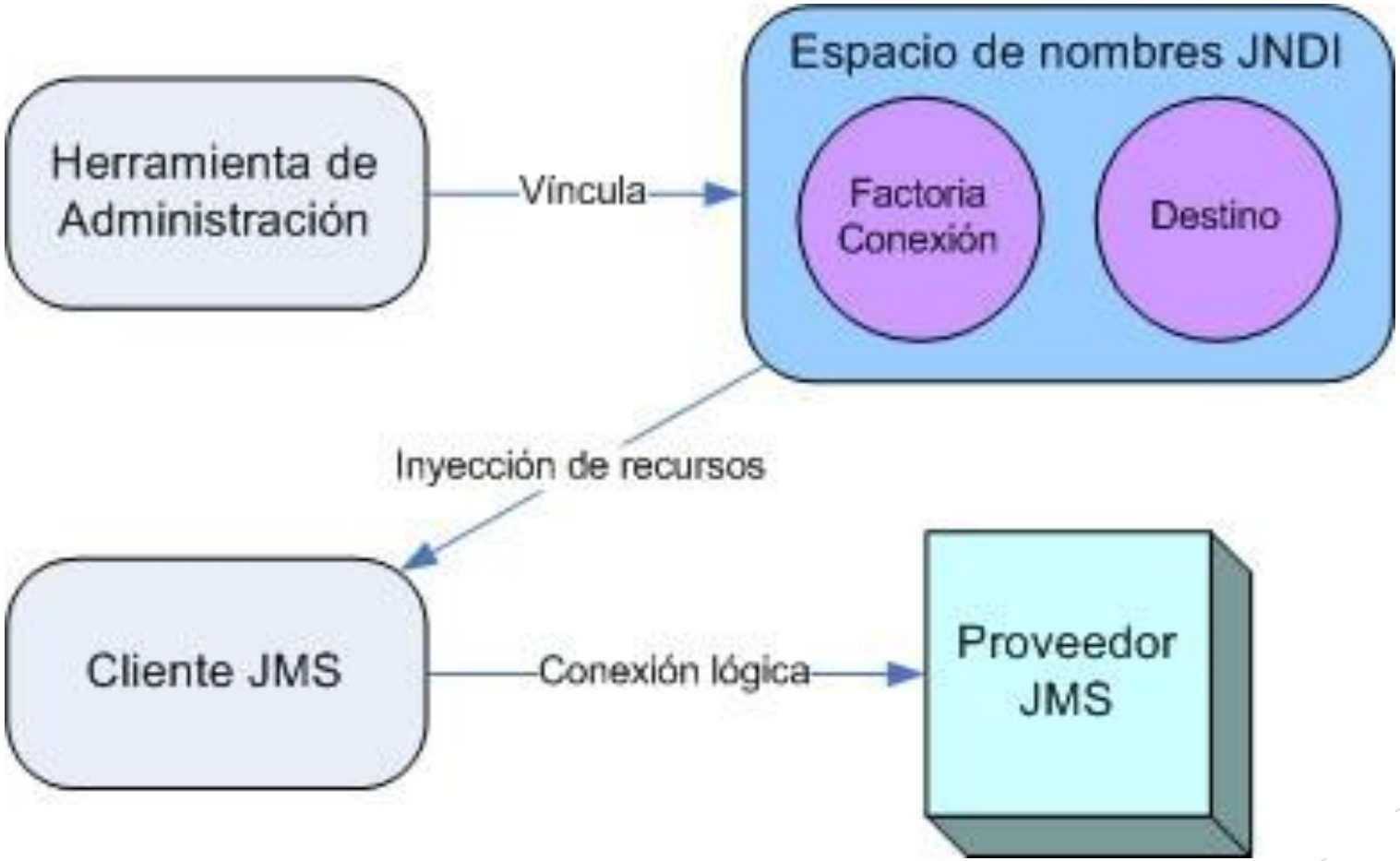
Objetos administrados

- objetos JMS pre configurados creados por un administrador para el uso de los clientes
- ConnectionFactory, Destino (cola o tema)

Proveedor de JMS

- sistema de mensajería que implementa el JMS y funcionalidad administrativa

ADMINISTRACION JMS



Dominios de mensajería JMS

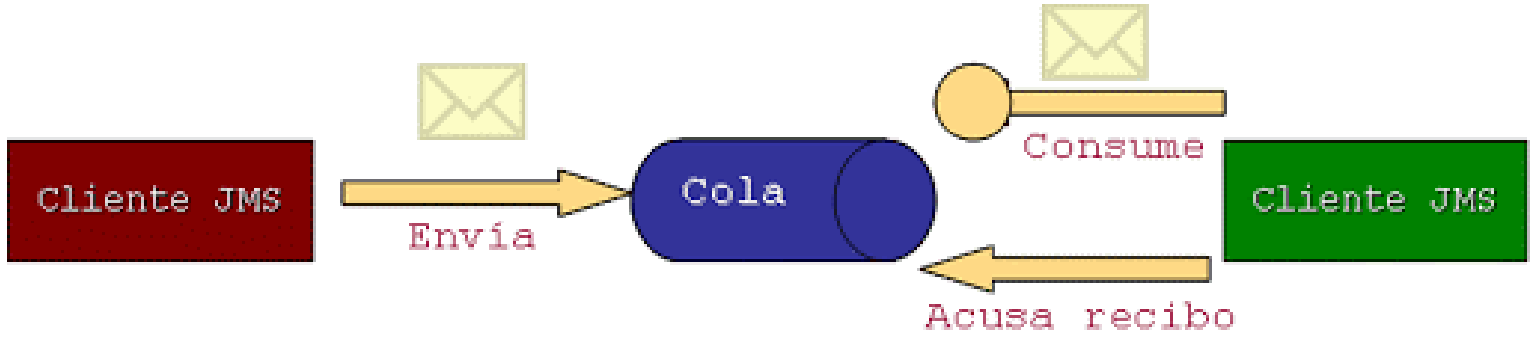
Punto a punto (PTP)

- construido alrededor del concepto de colas de mensajes
- cada mensaje tiene un solo consumidor

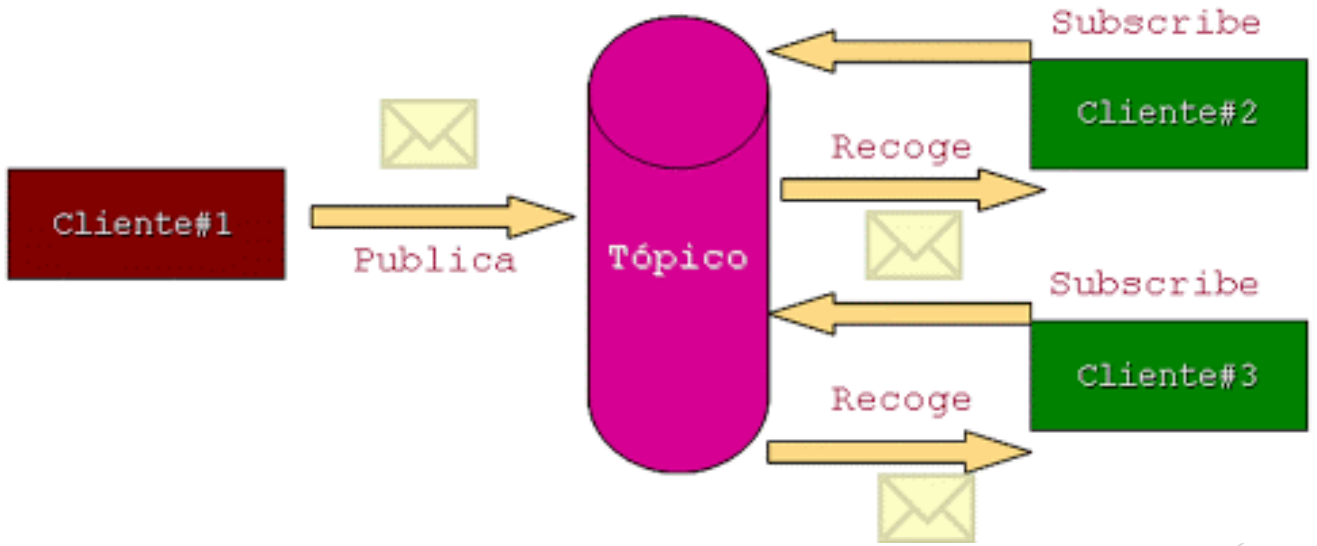
Sistemas de publicación y suscripción

- utiliza un "tema" para enviar y recibir mensajes
- cada mensaje tiene múltiples consumidores

Mensajes punto a punto (PTP)



Publicar/suscribir mensajes



Diferencia entre Connection Factory y TopicConnection

Un cliente utiliza un objeto QueueConnectionFactory para crear objetos QueueConnection con un proveedor JMS punto a punto. QueueConnectionFactory se puede utilizar para crear una QueueConnection, a partir de la cual se pueden crear objetos especializados relacionados con la cola.

Un objeto TopicConnection es una conexión activa a un proveedor JMS de publicación/suscripción. Un cliente utiliza un objeto TopicConnection para crear uno o más objetos TopicSession para producir y consumir mensajes. Un TopicConnection puede ser utilizado para crear un TopicSession, a partir del cual se pueden crear objetos especializados relacionados con el tema.

Consumo de mensajes

Sincrónicamente

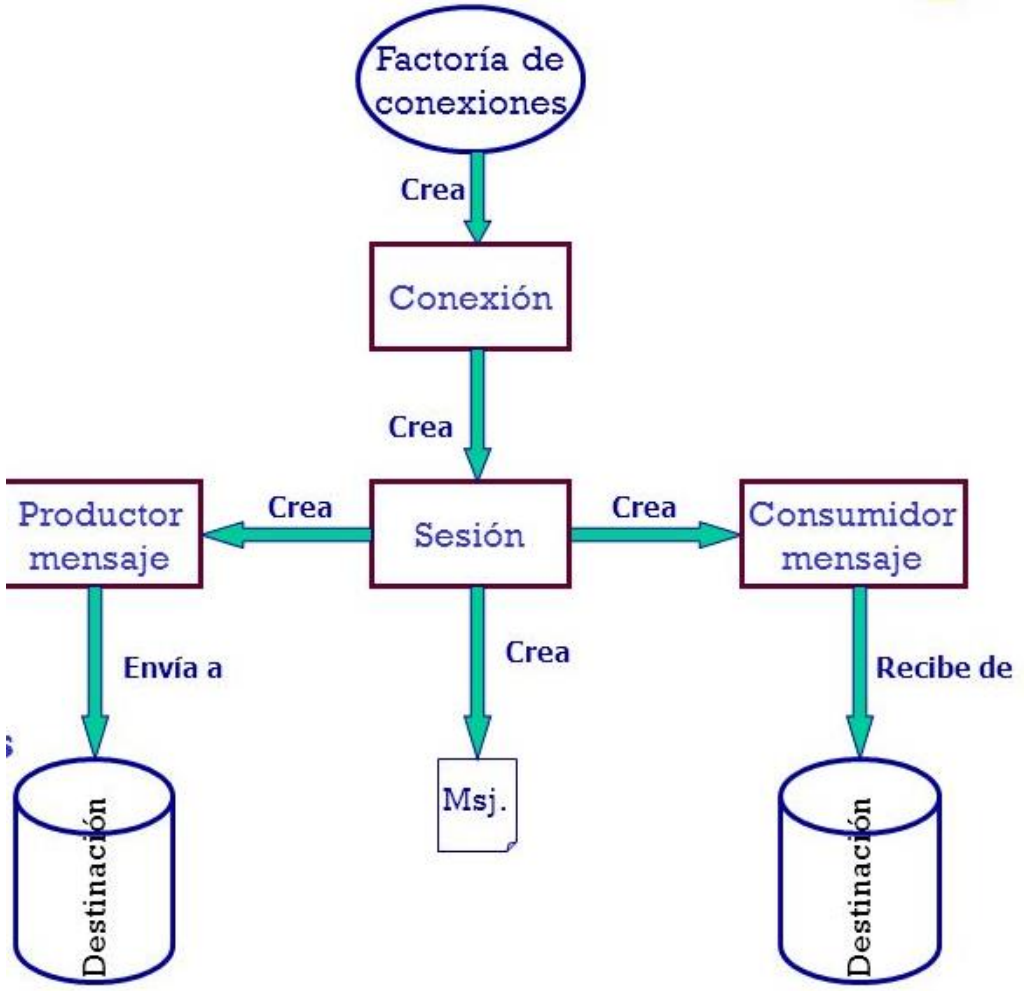
- Un suscriptor o un receptor explícitamente obtiene el mensaje del destino llamando al método de recepción.
- El método de recepción puede bloquearse hasta que llegue un mensaje o puede agotarse si un mensaje no llega en un plazo determinado.

Asincrónicamente

- Un cliente puede registrar a un oyente de mensajes con un consumidor.
- Cada vez que un mensaje llega a su destino, el proveedor del JMS entrega el mensaje llamando al método onMessage() del oyente.

Modelo de programación de la API del JMS

1. Adquirir una factoría de conexión
2. Crear una conexión mediante la factoría de conexión
3. Comenzar la conexión.
4. Crear una sesión a partir de la conexión
5. Adquirir un destino.
6. Dependiendo de si enviamos o recibimos
   - Crear un productor
     1. Crear un productor.
     2. Crear un mensaje y adjuntarlo a su destino.
   - Crear un consumidor.
     1. Crear un consumidor.
     2. Opcionalmente registrar un listener (escucha) de mensajes
7. Enviar/Recibir el/los mensaje/s
8. Cerrar los objetos (consumidor, productor, sesión, conexión)



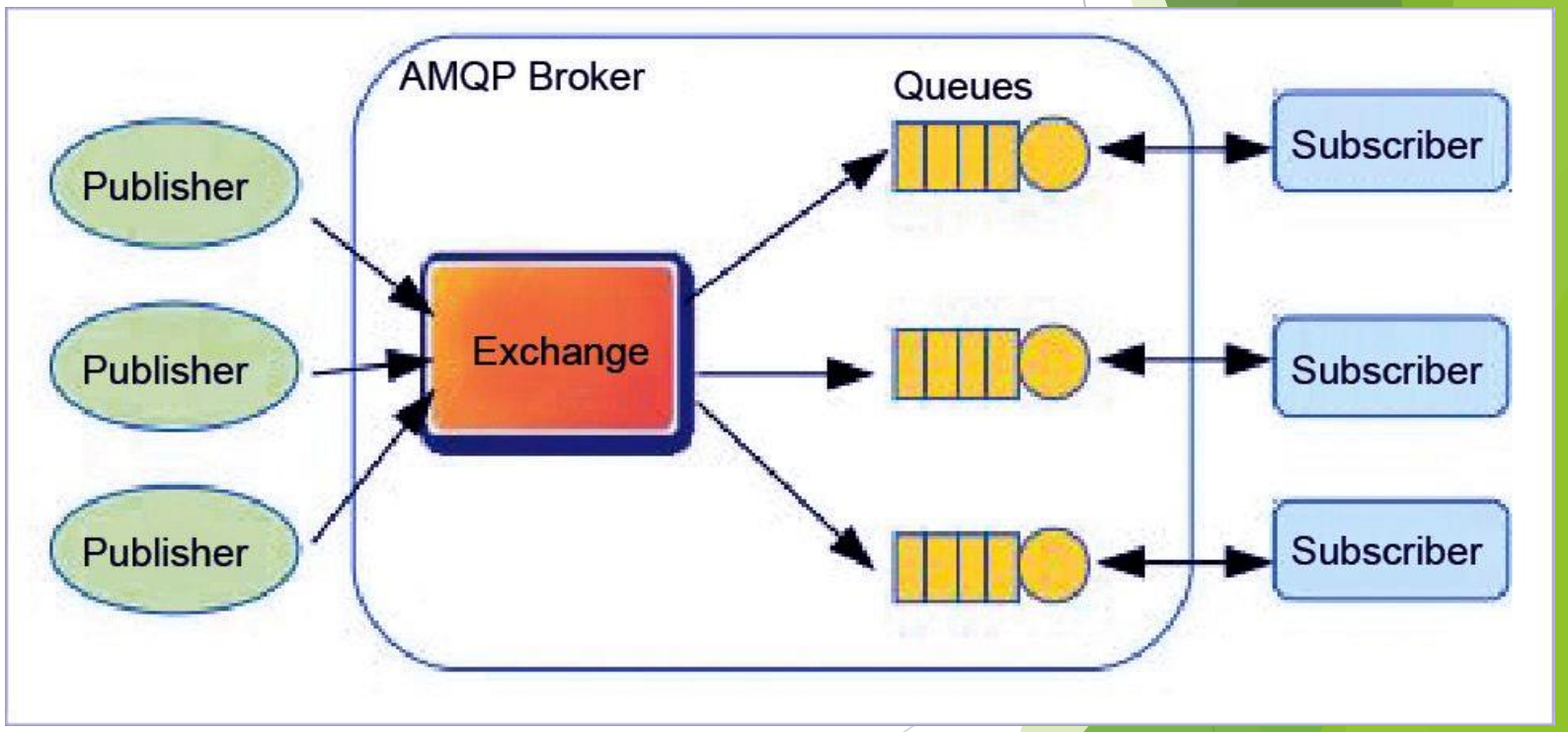
JMS con IOT(Internet de las cosas)

La mensajería ha sido un mercado maduro, y la mayor parte del desarrollo se ha realizado en Java. Ahora que está surgiendo la tecnología de Internet de las cosas (IoT), entran en juego nuevos protocolos como AMQP (Advanced Message Queuing Protocol) y MQTT (Message Queuing Telemetry Transport). Esto puede cambiar la esfera de la mensajería, según Eric Bruno, consultor y escritor sobre desarrollo de aplicaciones empresariales. Los sensores, como las lecturas de temperatura, deberán interactuar con las computadoras y unir diferentes protocolos de mensajería sin problemas, lo que aún incluye el uso de JMS.

Un ejemplo de tecnología de IoT sería una aplicación de red inteligente con una puerta de enlace integrada y sensores en toda la casa que proporcionarían uso de energía en tiempo real a la compañía eléctrica. Los datos de uso de energía se enviarían de forma anónima a los servidores back-end, y los análisis predecirían la demanda para evitar caídas de tensión o generar electricidad innecesaria

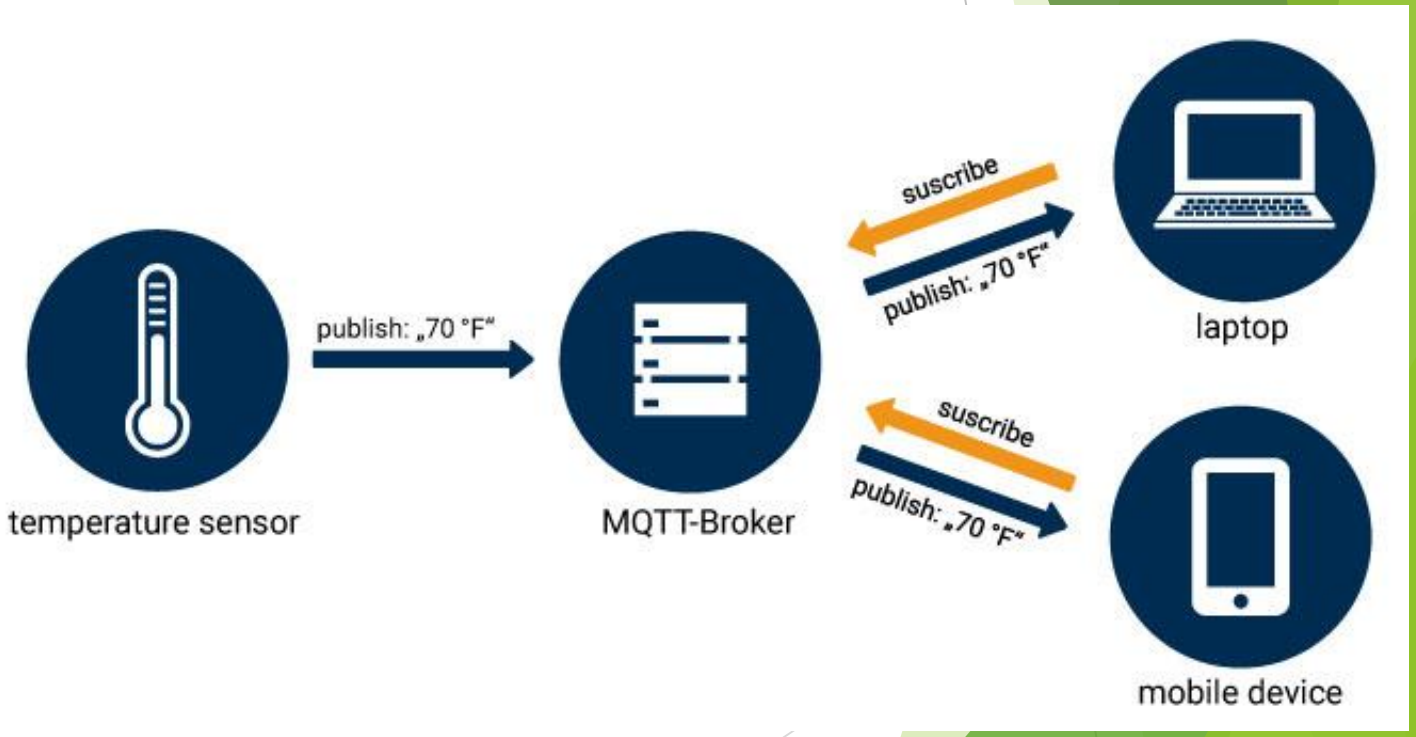
Advanced Message Queuing Protocol (AMQP)
Protocolo avanzado de colas de mensajes (AMQP)

Fue desarrollado por John O'Hara en JP Morgan Chase en Londres. AMQP es un protocolo de capa de software para entornos de middleware orientados a mensajes. Soporta el intercambio verbal fiable a través de primitivas de garantía de transporte de mensajes como "at-most-once", "at least once" y "exactly as soon as shipping".

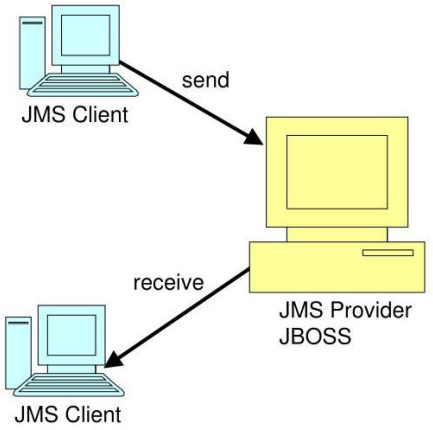


Message Queue Telemetry Transport Protocol (MQTT)
Protocolo de transporte de telemetría de colas de mensajes (MQTT)

Es un protocolo de mensajería, se desarrolló con la ayuda de Andy Stanford-Clark de IBM y Arlen Nipper de Arcom en 1999. Se utiliza normalmente para el seguimiento lejano en IoT. Su principal reto es recoger las estadísticas de muchos gadgets y la entrega de su infraestructura.



Ejemplo de P2P



8/22/2014

Ejemplo de Pub/Sub