Internet of Things (Internet de las Cosas)

Redes de Ordenadores para Robots y Máquinas Inteligentes

GSyC
Departamento de Teoría de la Señal y Comunicaciones y
Sistemas Telemáticos y Computación

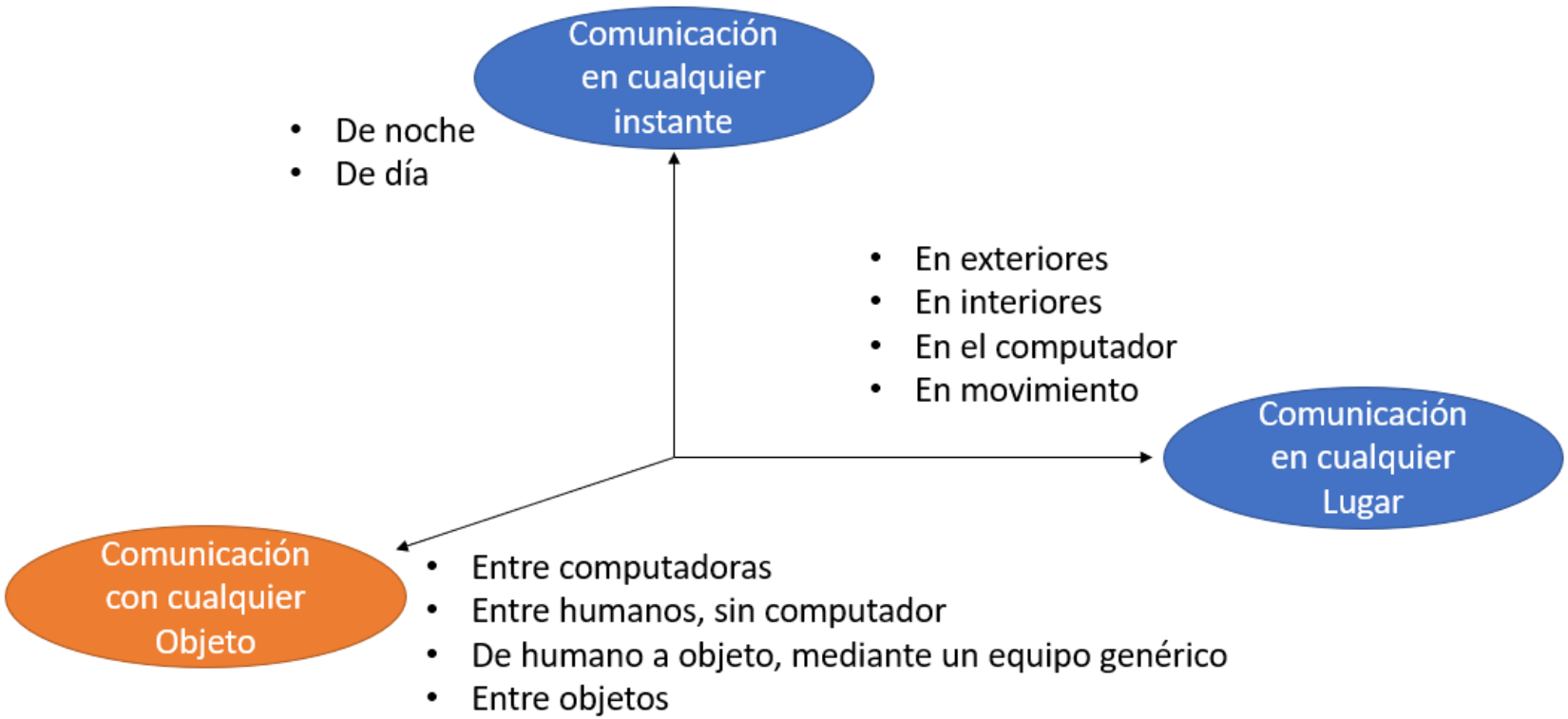
Abril de 2024

Contenidos

Contenidos

Internet de los Objetos

Internet de los objetos (IoT): Infraestructura que propicia la prestación de servicios avanzados mediante la interconexión de objetos (físicos y virtuales) gracias a la interoperatividad de tecnologías de la información y la comunicación.



Qué son los Objetos

En el contexto de Internet de los objetos se trata de un objeto del mundo físico (objetos físicos) o del mundo de la información (objetos virtuales) que se puede identificar e integrar en las redes de comunicaciones.

Componentes del Internet de los objetos

Componentes del Internet de los Objetos

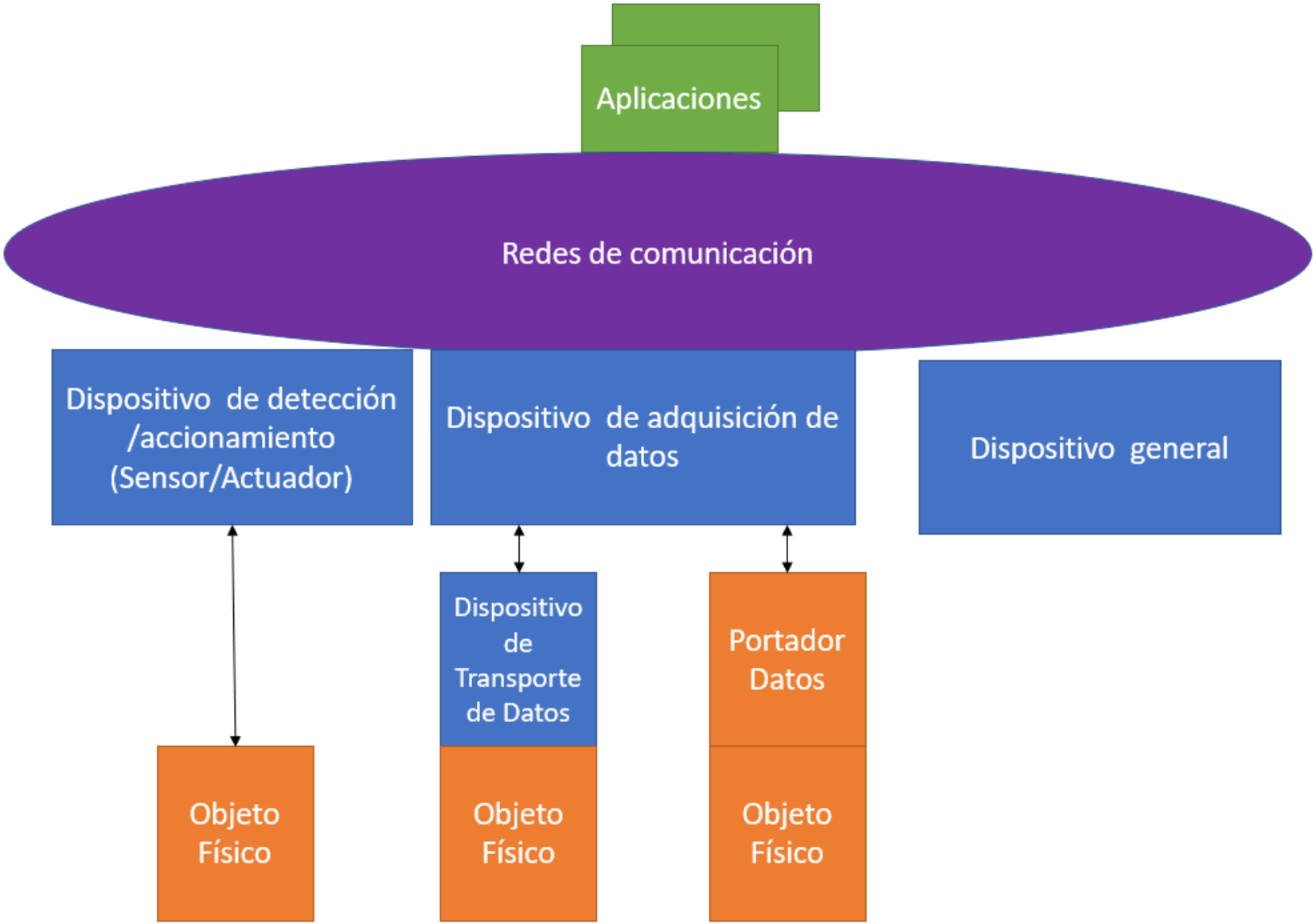
- Sensores
- Actuadores
- Microcontroladores
- Transceptores
- RFID

Componentes del Internet de los objetos

Sensores: Mide parámetros del entorno y entrega una señal electrónica proporcional al valor observado
Actuadores: Recibe una señal electrónica y responde interactuando con su entorno.
Microcontroladores: Proporciona la "inteligencia" de un objeto.
Transceptor: Proporciona funciones de transmisor y receptor en un solo dispositivo.
RFID: Permite identificar objetos, personas o animales; almacenar información acerca de ellos; y transferirlo a otros dispositivos electrónicos. Componentes principales:

- **Etiqueta (Tag)**: pequeño dispositivo programable aplicado a una entidad.
- **Lector:** recopila datos de etiquetas; normalmente conectado a un sistema informático

Arquitectura (I)



Arquitectura (II)

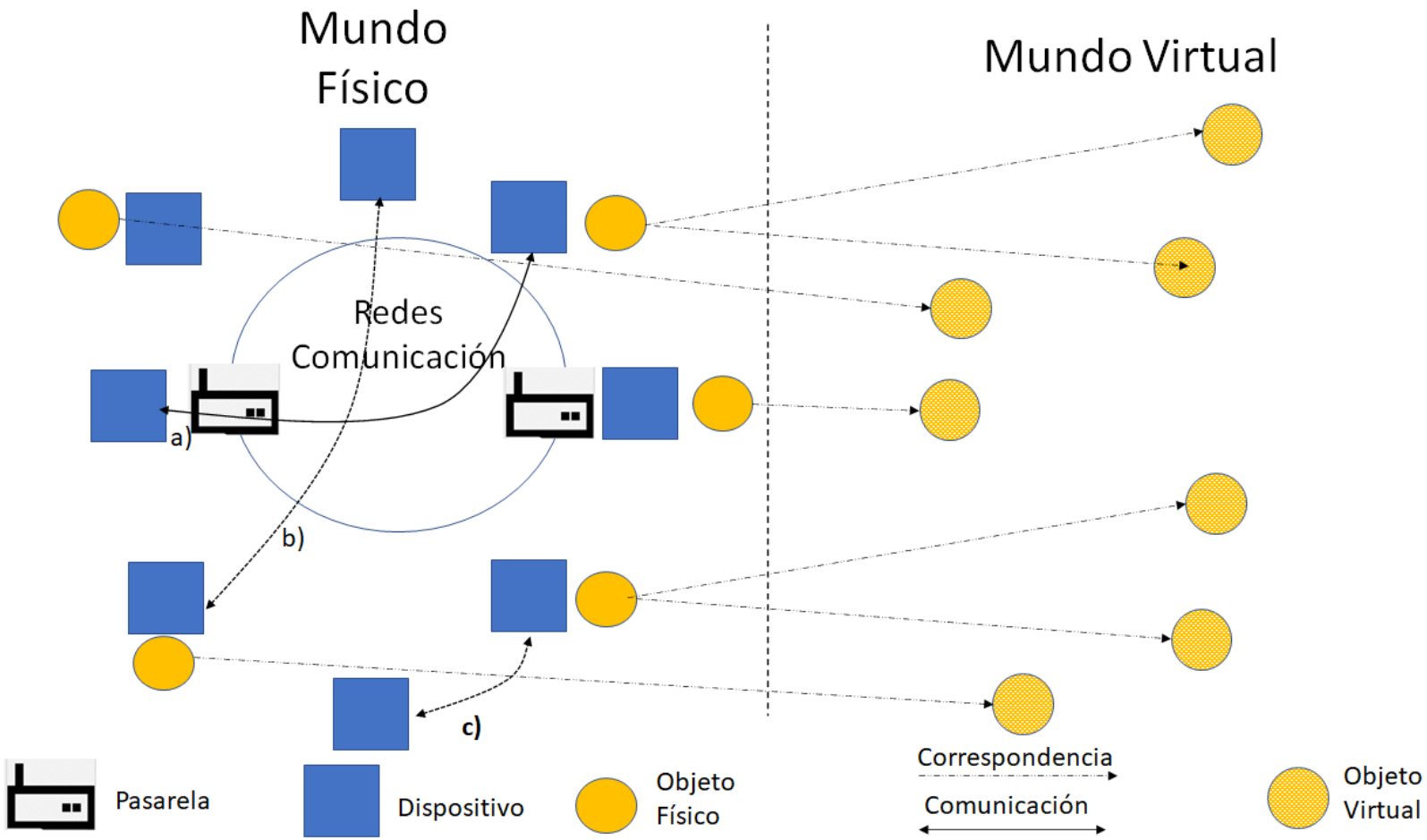
Término	Definición
Aplicación	Interactúa con los objetos para proporcionar la información útil para los usuarios finales
Redes de comunicación	Envía datos capturados por los dispositivos a las aplicaciones e instrucciones de las aplicaciones a los dispositivos
Dispositivo	Equipamiento con la capacidad de comunicación y opcionalmente con la capacidad de actuar, medir, capturar datos, almacenar y procesar datos
Objeto	Objetos del mundo físico (objetos físicos) o del mundo de la información (mundo virtual) que se pueden identificar e integrar en redes de comunicación. Los objetos físicos existen en el mundo físico y es posible detectarlos, actuar sobre ellos y conectarlos. Los objetos virtuales existen en el mundo de la información y se pueden almacenar, procesar y acceder a las mismas.

Arquitectura (III)

Término	Definición
Dispositivo de detección y accionamiento	Detecta o mide información de su entorno y la convierte en señales electrónicas digitales. También puede convertir señales electrónicas digitales procedentes de las redes de información en operaciones. Por lo general, los dispositivos de detección y accionamiento forman redes locales que se comunican entre sí utilizando tecnologías de comunicación alámbricas o inalámbricas y utilizan pasarelas para conectarse con las redes de comunicación.
Dispositivo de transporte de datos	Dispositivo anexo a un objeto físico para conectar indirectamente el objeto físico con las redes de comunicación. (ejemplo RFID tag)
Dispositivo de adquisición de datos	Dispositivo de lectura/escritura con capacidad para interactuar con objetos físicos. La interacción puede suceder indirectamente a través de dispositivos de transporte de datos, o directamente a través de dispositivos de transporte de datos unidos a los objetos físicos.
Portador datos	Un objeto portador de datos sin batería conectado a algo físico (por ejemplo, códigos de barras y códigos QR).
Dispositivo genérico	Dispositivo que cuenta con capacidades de procesamiento y comunicación Los dispositivos generales incluyen equipos y aplicaciones para diferentes dominios de aplicación IoT, tales como máquinas industriales, electrodomésticos y teléfonos inteligentes.

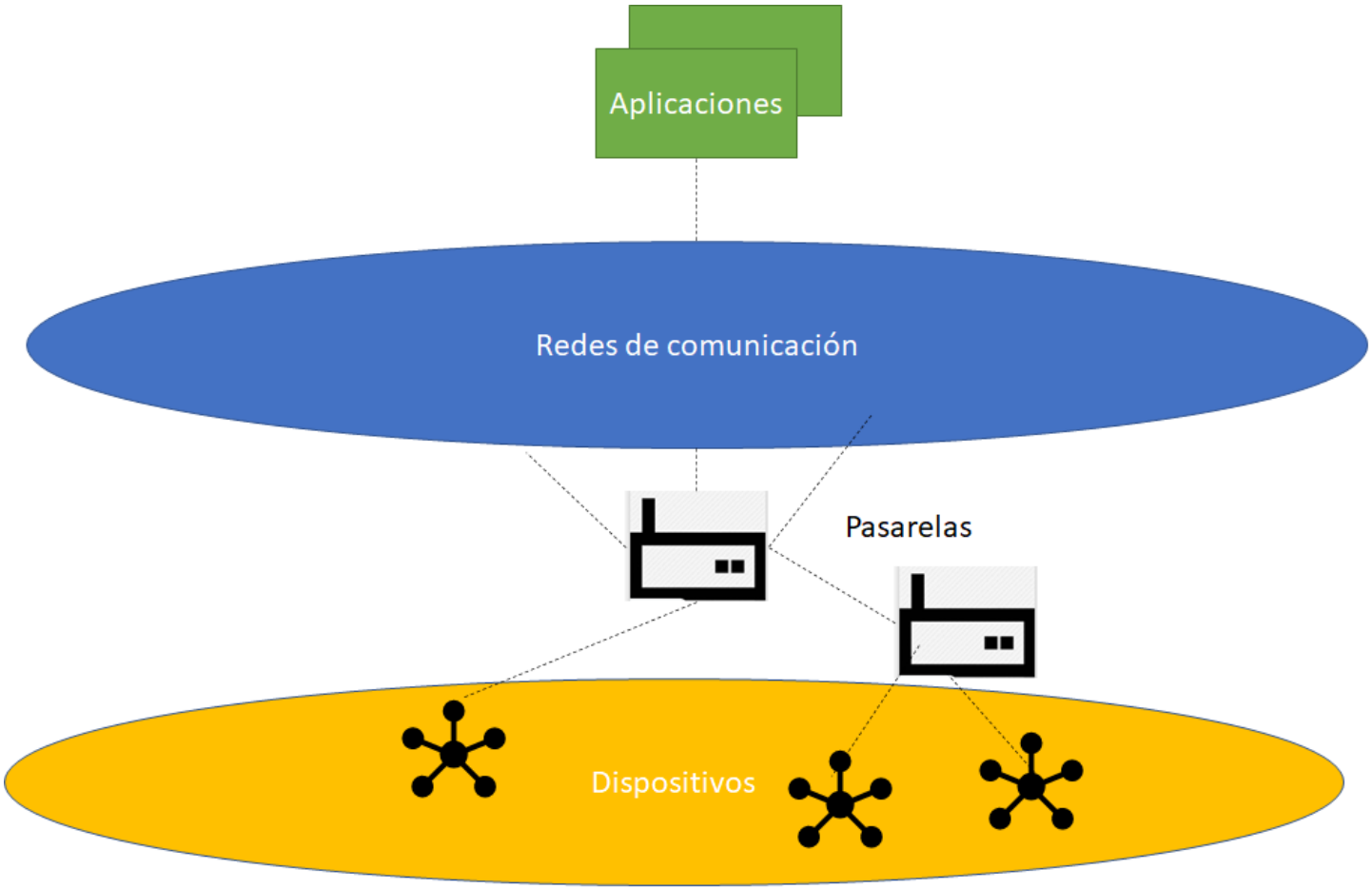
Descripción técnica de Iot

Los dispositivos se comunican con otros dispositivos: a través de la red de comunicaciones por medio de una pasarela, por medio de otros dispositivos o directamente. Un objeto físico puede estar representado en el mundo de la información por una o varios objetos virtuales (correspondencia), pero el objeto virtual también puede existir sin tener asociada ninguna objeto físico.

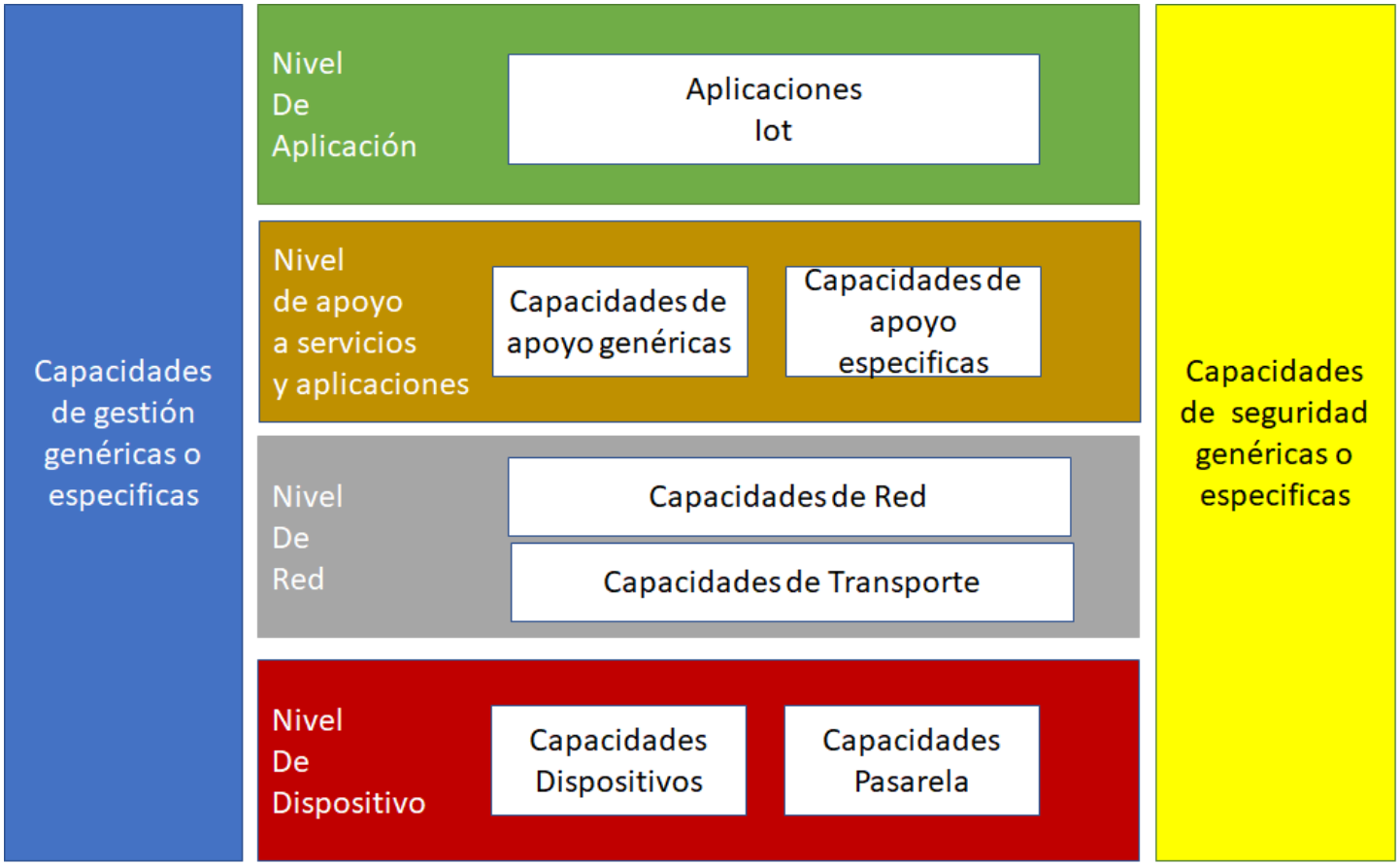


Pasarelas

Interconecta los dispositivos con las redes de comunicación. Realiza la traducción necesaria entre los protocolos empleados en las redes de comunicación y los utilizados por los dispositivos.



Arquitectura IoT



Arquitectura IoT (Niveles superiores y comunes)

- **Nivel de aplicación** El nivel de aplicación contiene las aplicaciones IoT.
- **Nivel de apoyo a servicios de aplicación**: El nivel de soporte de servicios y aplicaciones consiste en necesidades que pueda tener la aplicación ejemplos de este estilo almacenamiento, alta disponibilidad, tolerancia a fallos, etc
- **Capacidades de gestión:** gestión de dispositivos, como activación y desactivación de dispositivos remotos, diagnóstico, actualización del firmware y/o del software, gestión del estado de trabajo del dispositivo;etc Las capacidades especificas son para casos como por ejemplo PLC
- **Capacidades de seguridad:** garantizar en todas las capas el cumplimiento de las reglas de confidencialidad, integridad y disponibilidad

Arquitectura IoT (Nivel de Red)

- **Capacidades de red**: ofrecen funciones de control de la conectividad en red, tales como funciones de control de acceso y de recursos de transporte, gestión de la movilidad y autenticación, autorización y contabilidad (AAA)
- **Capacidades de transporte:** centradas en suministrar conectividad para el transporte de información y datos específicos de servicios y aplicaciones IoT, así como el transporte de información de control y gestión relacionada con IoT.

Arquitectura IoT (Nivel de Dispositivo)

Capacidades de dispositivo:

- **Interacción directa o indirecta con la red de comunicaciones:** Los dispositivos pueden recabar y cargar información directamente (por ejemplo, instrucciones) en la red de comunicación o indirectamente mediante uso de pasarelas.
- **Redes ad-hoc**: Los dispositivos puede construir redes de manera ad-hoc en algunas circunstancias cuando sea necesario para aumentar la capacidad evolutiva y la velocidad de despliegue.
- **Modo reposo y activo:** los dispositivos deben disponer de mecanismos para pasar a los modos reposo y activo con el objetivo de ahorrar energía

Arquitectura IoT (Nivel de Dispositivo (II))

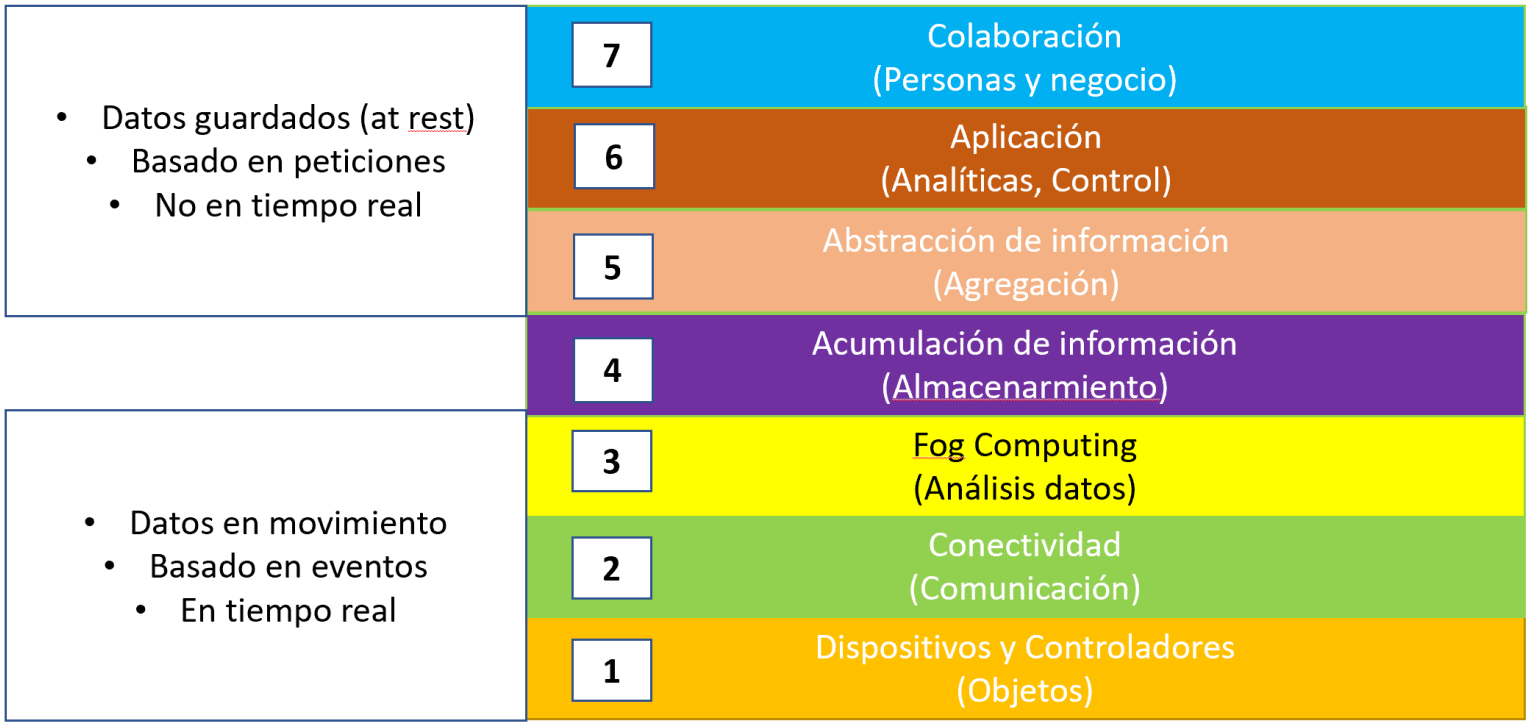
Capacidades de pasarela:

• **Soporte de interfaces múltiples:** En el nivel de dispositivo, las capacidades de pasarela soportan dispositivos conectados mediante diferentes tipos de tecnologías alámbricas e inalámbricas, tales como el bus de red de control de zona (CAN), ZigBee, Bluetooth o Wi-Fi. En el nivel de red, las capacidades de pasarela pueden comunicarse a través de diversas tecnologías

• **Conversión de protocolo:** Hay dos tipos de situaciones en las que se necesitan capacidades de pasarela. Una es cuando las comunicaciones en el nivel de dispositivo utilizan protocolos diferentes, por ejemplo, protocolos de tecnología ZigBee y Bluetooth, y la otra es cuando en la comunicación intervienen el nivel de dispositivo y la de red y se utilizan protocolos diferentes en cada una, por ejemplo, el protocolo de tecnología ZigBee en el nivel de dispositivo y el protocolo de tecnología 3G en el nivel de red.

Modelo de referencia de Iot IWF

.



Nivel 1 Dispositivos y controladores

- Dispositivos fisicos y controladores que controlan diversos dispositivos
- Se corresponde con el nivel de dispositivos del modelo de la ITU-T
- Capacidades de los dispositivos
  - Conversión digital-analógica y al revés
  - Generación de datos

Nivel 2 Conectividad

- Permite comunicaciones confiables y oportunas:
  - Entre dispositivos
  - Entre dispositivos y el nivel superior como las aplicaciones
- Consta de dispositivos de red, por ejemplo, routers, conmutadores y cortafuegos
- Las comunicaciones y el procesamiento en este nivel deben ser proporcionados por redes existentes
  - Es posible que se necesiten pasarelas para admitir dispositivos antiguos(no IP)
- La conectividad también incluye seguridad a nivel de red
- Corresponde al nivel de red del modelo de referencia de ITU-T

Nivel 3 Fog Computing

- Convierte datos de red de gran volumen en información adecuado para almacenamiento y procesamiento de nivel superior
- El procesamiento comienza temprano y cerca del borde del red (computación en la niebla (fog))...
- ... pero es limitado, se realiza paquete por paquete
  - Evaluación: ¿Deberían procesarse los datos a un nivel superior?
  - Reformateo, para un procesamiento de nivel superior consistente
  - Expandir / decodificar, usando información de contexto
  - Reducción, para minimizar el impacto en la red y niveles superiores
  - Evaluación de umbrales y alertas
  - Los elementos de este nivel pueden verse como dispositivos generales del modelo ITU-T

Nivel 4 Acumulación información

- Almacena los datos y los hace utilizables para aplicaciones
  - La generación de datos se basa en eventos y los datos se mueven a través de la red generada por dispositivos
  - Pero la mayoría de las aplicaciones no procesan datos en velocidades de transferencia de red
  - **El nivel 4 convierte datos en movimiento en datos en reposo**, ofreciendo mecanismo basado en consultas para acceder a datos relevantes
- Reduce los datos mediante el filtrado y el almacenamiento selectivo, determinando:
  - Si los datos son relevantes para los niveles superiores
  - Si los datos deben conservarse y cómo (por ejemplo, en una base de datos relacional)
  - Si los datos deben ser recombinados, recalculados o agregados

Nivel 5 Abstracción información

• Agrega, formatea y almacena datos de manera que las aplicaciones puedan acceder de forma eficiente
• El procesamiento en este nivel incluye:
 o Combinar datos de múltiples fuentes, conciliar diferencias en formatos de datos, semántica y acceso protocolos
 o Alerta a las aplicaciones de nivel superior de que los datos están completos
 o Consolidando datos en un solo lugar
 o Protección de datos (autenticación y autorización)
 o Normalizar / desnormalizar e indexar datos para proporcionar un acceso rápido a aplicaciones

Nivel 6 Aplicación

- Incluye aplicaciones que usan datos de IoT o controlan dispositivos IoT
- Generalmente, las aplicaciones interactúan con el nivel 5 y los datos en reposo, aunque pueden tener acceso directo al nivel 3 y 2 (conectividad)
- La definición de aplicaciones está fuera del alcance de la modelo de referencia

Nivel 7 Colaboración

- La definición de este nivel destaca que un sistema de IoT es valioso siempre que produce acciones, lo que normalmente requiere personas y procesos comerciales
- Con frecuencia, las acciones requieren comunicación y colaboración entre personas

Contenidos

Tecnologías de conectividad

Zigbee ZigBee es una tecnología inalámbrica más centrada en aplicaciones domóticas e industriales. Los perfiles ZigBee PRO y ZigBee Remote Control (RF4CE) se basan en el protocolo IEEE 802.15.4, una tecnología de red inalámbrica que opera a 2,4GHz en aplicaciones que requieren comunicaciones con baja tasa de envío de datos dentro de áreas delimitadas con un alcance de 100 metros, como viviendas o edificios

Wifi Ampliamente desplegado, proporciona grandes tasas transmisión pero consume más potencia

Bluetooth Bluetooth de baja energía, también conocido como Bluetooth LE o Bluetooth Smart, es otro protocolo importante para desarrollar aplicaciones IoT. Se caracteriza por ofrecer un alcance similar al de la tecnología Bluetooth normal pero con un consumo de energía significativamente reducido

Tecnologías de conectividad Resumen

Tipo Conectividad	Estándar	Frecuencia	Alcance	Velocidad Transferencia
Zigbee	IEEE 802.15.4	2,4Ghz	10-100m	250kbps
Wifi	Basado en 802.11(a,b,n)	2,4 y 5 GHz	Aprox 50 m	Hasta 600 Mbps
Bluetooth	Bluetooth 4.2	2,4Ghz	50-150m	1Mbps

Tecnologías de conectividad (II) Low-power wide-area (LPWA) y Low-power wireless personal area networks (LoWPAN)

Sigfox muchas aplicaciones M2M que funcionan con una batería pequeña y requieren niveles menores de transferencia de datos, en espacios donde WiFi se queda demasiado corto y la comunicación móvil es muy cara y consume demasiada energía.

LoraWan LoRaWAN está diseñada para implementar redes de área amplia (WAN) con características específicas para soportar comunicaciones móviles, bidireccionales, económicas y seguras para aplicaciones de IoT

6LoWPAN (IPv6 Low-power wireless Personal Area Network) es una tecnología inalámbrica basada en IP . 6LowPAN es un protocolo de red que permite mecanismos de encapsulado y compresión de cabeceras. Ofrece libertad de banda de frecuencia y nivel físico, por lo que se puede utilizar a través de múltiples plataformas de comunicaciones, como Ethernet, Wi-Fi, 802.15.4.

Tecnologías de conectividad (II) Low-power wide-area (LPWA) y Low-power wireless personal area networks (LoWPAN) Resumen

Tipo Conectividad	Estándar	Frecuencia	Alcance	Velocidad Transferencia
Sigfox	Sigfox	900MHz	30-50km (ambientes rurales), 3-10km (ambientes urbanos)	10-1000 bps
LoraWan	Lora	Varias	15km (entorno rural) 2-5km (entorno urbano)	Hasta 600 Mbps
6LoWPAN	RFC 6282	Adaptable a múltiples capas físicas como Bluetooth, ZigBee (2,4 Ghz) o RF de bajo consumo (sub-1GHz)	N/A	N/A

Tecnologías de conectividad móvil

- En Releases anteriores a release 13 algunos conceptos como PSM (Power Saving Mode) el concepto de dispositivos de tipo 0 e Iot como MTC (Machine Traffic Communication)
- 3G 3GPP Release 13 incluyó **eMTC (enhanced MTC) Cat M1 (conocido tambien LTE M o Categoria M)**,
- 4G NB IOT **(NarrowBand IoT)** Está desarrollada para permitir una comunicación eficiente y una larga duración de la batería para dispositivos distribuidos de manera masiva,
- 5G NB IOT **(5G NarrowBand IoT)** Se introducen conceptos de slicing de red con asignación de recursos para los mMTC (massive Machine Traffic Communication)

Contenidos

Protocolos de aplicaciones en Iot

MQTT MQ Telemetry Transport. Lo analizaremos en profundidad más tarde
COAP Constrained Application Protocol
LwM2M LightWeight Machine-to-Machine

COAP

COAP Estandarizado por IETF en RFC 7252

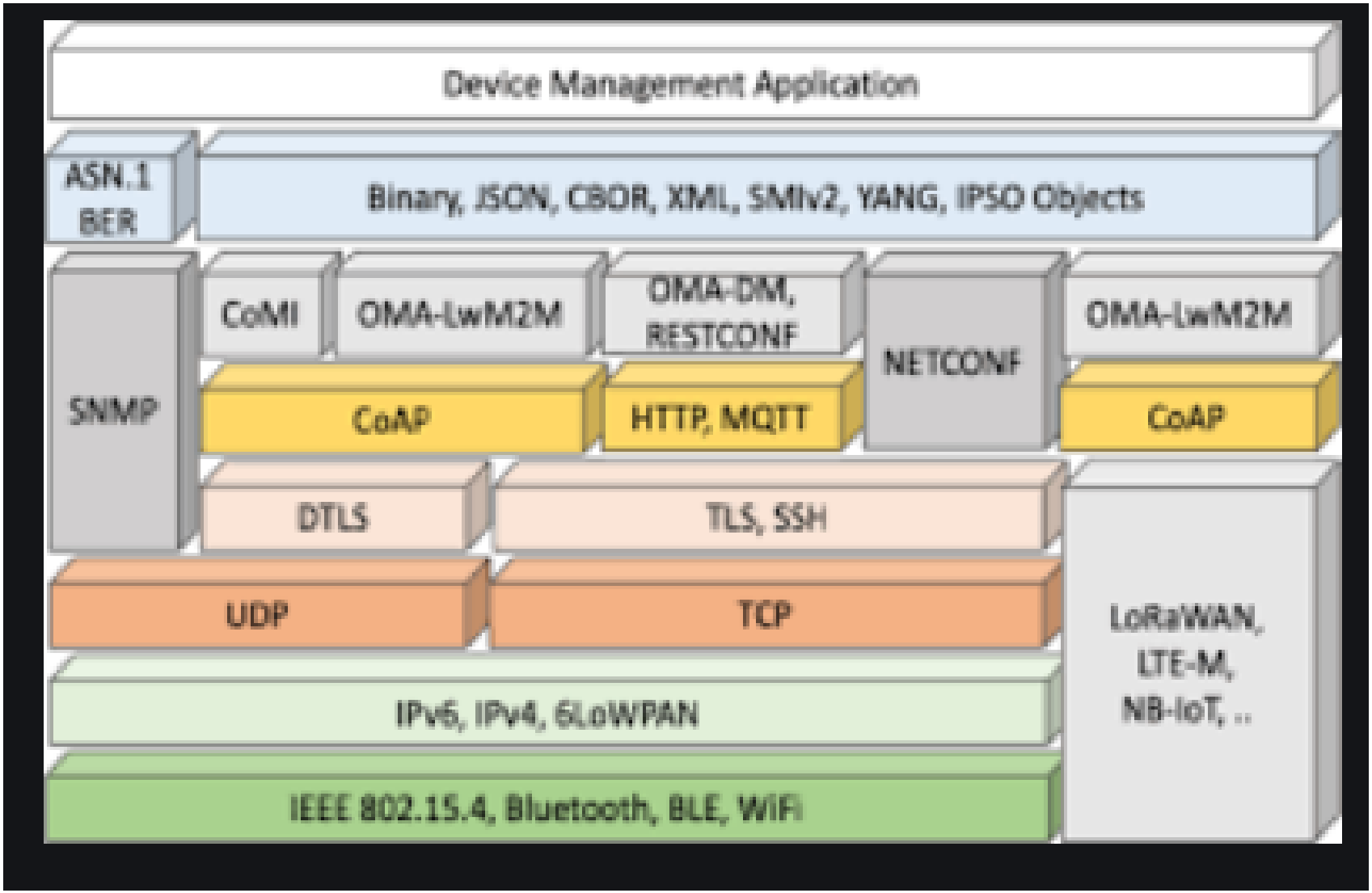
- Un protocolo de transferencia web especializado para usar con nodos y redes restringidas- en términos de capacidad en IoT.
- Diseñado para aplicaciones machine-to-machine (M2M).
- Al igual que HTTP, CoAP esta basado en un modelo REST:
  - Los servidores ponen sus recursos bajo una URL
  - Los clientes acceden a esos recursos usando métodos como GET, PUT, POST, y DELETE.
- CoAP se ha diseñado para trabajar con controladores con poca memoria (10 kib) y espacio de usuario
- CoAP usa UDP o DTLS

LwM2M

- Protocolo de gestión para dispositivos con recursos limitados.
- Especificado por SpecWorks de Open Mobile Alliance (OMA).Propuesto originalmente en 2012.
- LwM2M es un protocolo de gestión de dispositivos diseñado para redes de sensores. Inspirado en la gestión de dispositivos OMA. Un servidor lee / escribe objetos en UE.
- Originalmente, solo usaba CoAP como protocolo de nivel inferior.Hasta la versión 1.1, LwM2M admitía el transporte basado en CoAP sobre UDP, DTLS, TCP, TLS y SMS. LwM2M v1.2 introduce soporte para protocolos de transporte HTTP y MQTT. Aplicable a Cellular, 6LoWPAN, WiFi, ZigBee IP y cualquier otro dispositivo IP.

Caso de uso: mecanismo de habilitación de servicios y administración remota de bajo costo Habilita tanto la gestión como los datos de la aplicación.

Vista de los diferentes protocolos



Obtenido de https://ieeexplore.ieee.org/document/8848791

Contenidos

Caracteristicas MQTT

MQTT es un protocolo de mensajería estándar de OASIS para Internet de las cosas (IoT). Está diseñado como un transporte de mensajería de publicación / suscripción extremadamente liviano que es ideal para conectar dispositivos remotos con una huella de código pequeña y un ancho de banda de red mínimo. MQTT hoy en día se utiliza en una amplia variedad de industrias, la fabricación, las telecomunicaciones, el petróleo y el gas, etc.
El funcionamiento del MQTT es un servicio de mensajería push con patrón publicador/suscriptor (pub-sub).Los clientes se conectan con un servidor central denominado broker.
Para filtrar los mensajes que son enviados a cada cliente los mensajes se disponen en topics organizados **jerárquicamente**. Un cliente puede publicar un mensaje en un determinado topic. Otros clientes pueden suscribirse a este topic, y el broker le hará llegar los mensajes suscritos.

Conectividad Red

MQTT usa TCP. Soporta tambien su uso con TLS.
MQTT emplea el puerto 1883 y el 8883 cuando funciona sobre TLS
Conecta el Cliente al Servidor.
Proporciona los medios para enviar un flujo de bytes ordenado y sin pérdidas en ambas direcciones.

Definiciones: mensaje de aplicación

Mensaje de aplicación: Los datos transportados por el protocolo MQTT a través de la red para la aplicación. MQTT transporta un tipo de mensaje de aplicación, una calidad de servicio (QoS) y un nombre de tema (topic).
Sesión: Una interacción con estado entre un cliente y un servidor. Algunas sesiones duran solo la conexión de red , otras pueden abarcar varias conexiones de red consecutivas entre un cliente y un servidor.

Definiciones: Cliente

Cliente: Un programa o dispositivo que usa MQTT.
Un cliente:

- abre la conexión de red al servidor
- publica mensajes de aplicación que pueden interesar a otros clientes.
- se suscribe para solicitar mensajes de aplicación que le interese recibir.
- cancela la suscripción para eliminar una solicitud de mensajes de aplicación.
- cierra la conexión de red al servidor.

Servidor o Broker

Servidor: Un programa o dispositivo que actúa como intermediario entre los clientes que publican mensajes de aplicación y los clientes que han realizado suscripciones.
Un servidor:

- acepta conexiones de red de clientes.
- acepta mensajes de aplicación publicados por los clientes.
- procesa las solicitudes de suscripción y cancelación de suscripción de los clientes.
- reenvía mensajes de aplicaciones que coinciden con las suscripciones de clientes.
- cierra la conexión de red del cliente.

Topic

Nombre del Topic (tema) La etiqueta adjunta a un mensaje de aplicación que se compara con las suscripciones conocidas por el servidor. Los Topics están formados por uno o más “niveles” separados entre sí por una barra inclinada ’/’. Cada nivel debe está formado por uno o más caracteres.
Ejemplos:
Casa/Cocina/Temperatura
Casa/Salon/Temperatura
Casa/Salon/Persiana
Filtro del Topic (tema) Una expresión contenida en una suscripción para indicar un interés en uno o más temas. Un filtro de tema puede incluir caracteres comodín.

Topic Wildcard (comodín)

Algunos carácteres comodín # +

• # coincide con cualquier número de niveles dentro de un tema.

Tiene que ser el último carácter. Ejemplo: sport/tennis/player1/#,

Es válido para:

· sport/tennis/player1

· sport/tennis/player1/ranking

· sport/tennis/player1/score/wimbledon

• + carácter comodín que solo coincide con un nivel de tema

Ejemplo sport/tennis/+ concuerda con sport/tennis/player1 y "sport/tennis/player2", pero no con sport/tennis/player1/ranking.

Los Wildcards son únicamente para la suscripción. Los mensajes pueden publicarse únicamente contra un Topic.

Subscripción

Subscripción Una suscripción comprende un filtro de tema (topic) y una Qos máxima. Una suscripción está asociada a una única sesión. Una sesión puede contener más de una suscripción. Cada suscripción dentro de una sesión tiene un filtro de tema diferente.
Subscripción compartida Una suscripción compartida comprende un filtro de tema y una Qos máxima. Una suscripción compartida se puede asociar con más de una sesión. Un mensaje de aplicación que coincide con una suscripción compartida solo se envía al cliente asociado con una de estas sesiones. Una sesión puede suscribirse a más de una suscripción compartida y puede contener tanto suscripciones compartidas como suscripciones que no se comparten.
Suscripción comodín: Una suscripción comodín es una suscripción con un filtro de tema que contiene uno o más caracteres comodín. Esto permite que la suscripción coincida con más de un nombre de tema.

MQTT/TCP vs HTTP/TCP

Centrado en los datos MQTT es agnóstico del contenido de los datos. HTTP sin embargo se centra en documento (codificación, lenguaje, etc)

Simplicidad MQTT tiene unos pocos métodos (Publish/(un)Subscribe). Es fácil de aprender. HTTP puede ser complejo. Tiene multitud de métodos y de códigos de respuesta

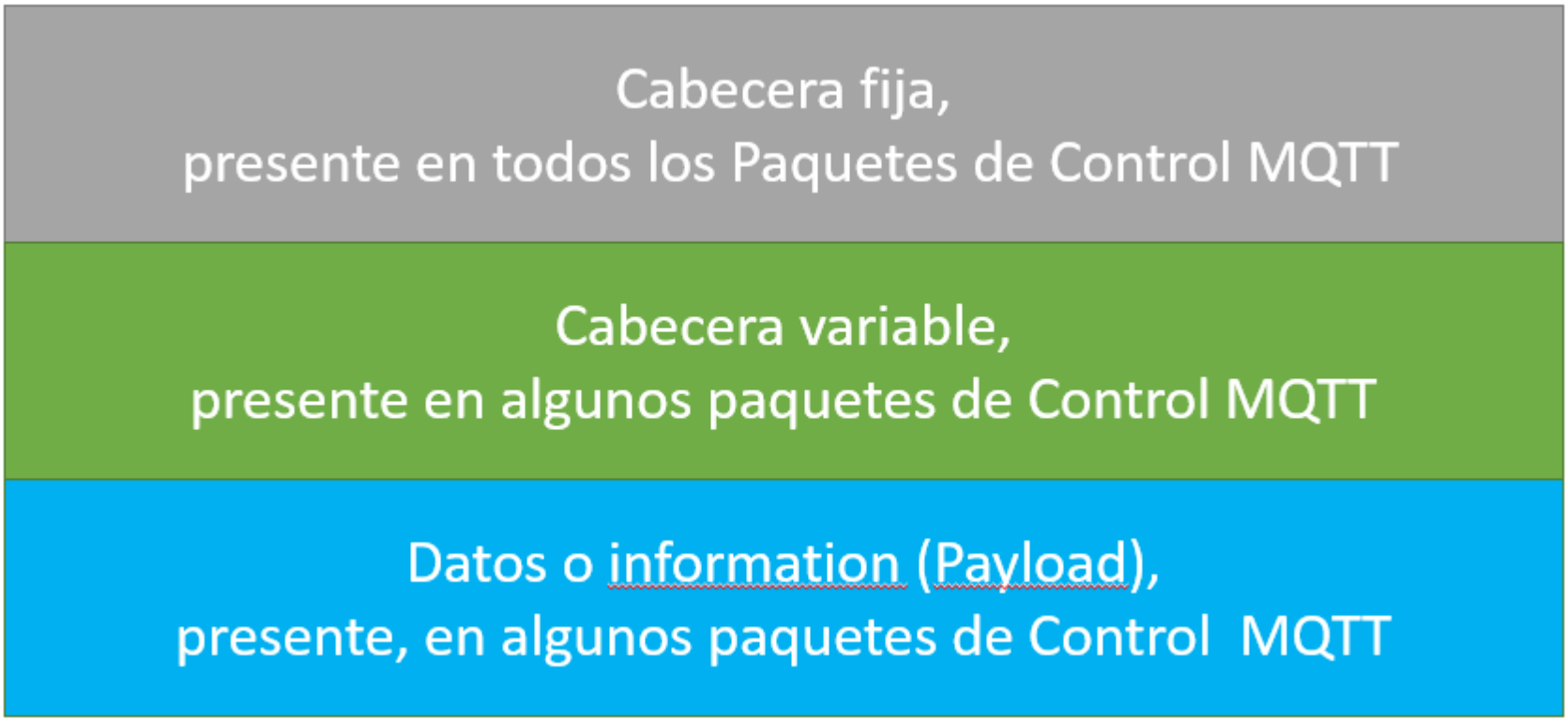
Ligereza y Huella El paquete más pequeño posible de MQTT es de 2 bytes.El protocolo (MQTT) está optimizado para redes no confiable, con poco ancho de banda y con alta latencia HTTP es más detallado - mucha 'charla' en un POST MQTT es muy ligero. HTTP es más pesado

Muchas topologías MQTT distribuye mensajes 1 a 1 y 1 a N con los mecanismos de publicación /suscripción de mensajes. HTTP es punto a punto

Qos MQTT soporta diferentes Qos HTTP no tiene reintento / confirmación. La aplicación debe gestionar los timeout y reintentos

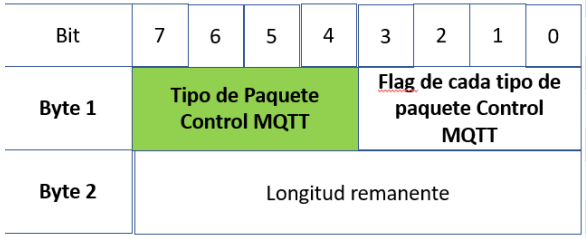
Estructura de trama de control

El protocolo MQTT opera intercambiando una serie de paquetes de control MQTT de una manera definida. Un paquete de control MQTT consta de hasta tres partes, siempre en el siguiente orden como se muestra a continuación.



Cabecera fija

Ocupa 2 a 5 bytes, obligatorio. Consta de un código de control, que identifica el tipo de mensaje enviado, y de la longitud del mensaje. La longitud se codifica en 1 a 4 bytes, de los cuales se emplean los 7 primeros bits, y el último es un bit de continuidad.



Nombre	Valor	Dirección
Reservado	0	Prohibido
CONNECT	1	Cliente->Servidor
CONNACK	2	Servidor-> Cliente
PUBLISH	3	Cliente-> Servidor o Servidor-> Cliente
PUBACK	4	Cliente-> Servidor o Servidor-> Cliente
PUBREC	5	Cliente-> Servidor o Servidor-> Cliente
PUBREL	6	Cliente-> Servidor o Servidor-> Cliente
PUBCOMP	7	Cliente-> Servidor o Servidor-> Cliente
SUBSCRIBE	8	Cliente->Servidor
SUBACK	9	Servidor-> Cliente
UNSUBSCRIBE	10	Cliente->Servidor
UNSUBACK	11	Servidor-> Cliente
PINGREQ	12	Cliente->Servidor
PINGRESP	13	Servidor-> Cliente
DISCONNECT	14	Cliente-> Servidor o Servidor-> Cliente
AUTH	15	Cliente-> Servidor o Servidor-> Cliente

Cabecera fija II

Los flags se usan más en paquete Publish que utiliza 3 campos:
DUP, QOS, RETAIN
La longitud remanente es un entero de bytes variables que representa el número de bytes que quedan dentro del paquete de control actual, incluidos los datos en el encabezado variable y la carga útil (payload). La longitud restante no incluye los bytes utilizados para codificar la longitud restante.
El tamaño del paquete es el número total de bytes en un paquete de control MQTT, esto es igual a la longitud del encabezado fijo más la longitud restante .

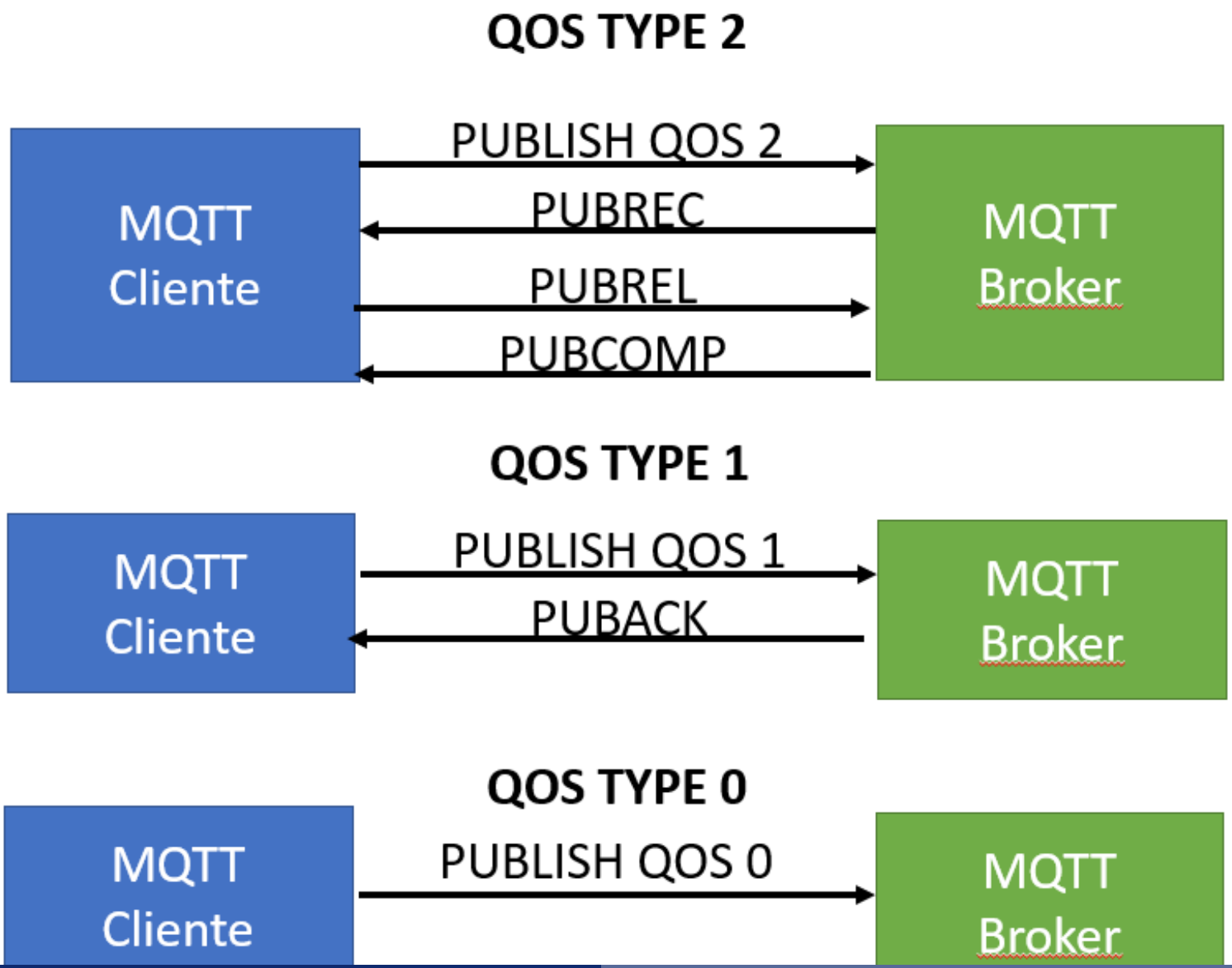
QoS

MQTT dispone de un mecanismo de calidad del servicio o QoS, entendido como la forma de gestionar la robustez del envío de mensajes al cliente ante fallos (por ejemplo, de conectividad).
MQTT tiene tres niveles QoS posibles.

- **QoS 0 unacknowledged (at most one)**: El mensaje se envía una única vez.
- **QoS 1 acknowledged (at least one):** El mensaje se envía hasta que se garantiza la entrega. En caso de fallo, el suscriptor puede recibir algún mensaje duplicados.
- **QoS 2 assured (exactly one).** Se garantiza que cada mensaje se entrega al suscriptor, y únicamente una vez.

Usar un nivel u otro depende de las características y necesidades de fiabilidad de nuestro sistema. Lógicamente, un nivel de QoS superior requiere un mayor intercambio mayor de mensajes de verificación con el cliente y, por tanto, mayor carga al sistema.

Qos (II)


QOS TYPE 2
MQTT Cliente
PUBLISH QOS 2
PUBREC
PUBREL
PUBCOMP
MQTT Broker
QOS TYPE 1
MQTT Cliente
PUBLISH QOS 1
PUBACK
MQTT Broker
QOS TYPE 0
MQTT Cliente
PUBLISH QOS 0
MQTT Broker

Retain message

En MQTT, el cliente que publica un mensaje no tiene garantía de que un cliente suscrito realmente reciba el mensaje. El cliente que publica sólo puede asegurarse de que el mensaje se entregue de forma segura al broker. El cliente que se conecta y se suscribe a temas no tiene garantía de cuándo se publicará un mensaje en uno de sus temas de interés. El editor puede tardar unos segundos, minutos u horas en enviar un nuevo mensaje en uno de los temas suscritos. Hasta que se publique el siguiente mensaje, el cliente que se suscribe no sabe el estado actual del tema.
El broker almacena el último mensaje (solo uno por tema) retenido y junto con el QoS de ese tema. Cada cliente subscrito a un patrón de tema que coincide con el tema del mensaje retenido recibe el mensaje retenido inmediatamente después de suscribirse.
Los mensajes retenidos ayudan a los clientes recién suscritos a obtener una actualización de estado inmediatamente después de suscribirse a un tema.

DUP

El indicador DUP debe establecerse a 1 por el cliente o servidor cuando intenta volver a entregar un paquete PUBLISH El indicador DUP debe establecerse en 0 para todos los mensajes QoS 0.

Cabecera variable

Opcional, contiene información adicional que es necesaria en ciertos mensajes o situaciones.

Identificador de paquete El componente de encabezado variable de muchos de los tipos de paquetes de control MQTT incluye un campo de identificador de paquete entero de dos bytes. Estos paquetes de control MQTT son PUBLISH (donde QoS mayor que 0), PUBACK, PUBREC, PUBREL, PUBCOMP, SUBSCRIBE, SUBACK, UNSUBSCRIBE, UNSUBACK.

Propiedades El último campo en la cabecera variable del paquete CONNECT, CONNACK, PUBLISH, PUBACK, PUBREC, PUBREL, PUBCOMP, SUBSCRIBE, SUBACK, UNSUBSCRIBE, UNSUBACK, DISCONNECT y AUTH es un conjunto de propiedades. En el paquete CONNECT también hay un conjunto opcional de propiedades en el campo Will Properties con el payload (carga útil).El conjunto de Propiedades se compone de una longitud de propiedad seguida de las propiedades.

Payload o datos y Reason Code

Payload: Es el contenido real del mensaje. Puede tener un máximo de 256 Mb aunque en implementaciones reales el máximo es de 2 a 4 kB. Algunos paquetes de control MQTT contienen una carga útil como parte final del paquete (CONNECT, SUBSCRIBE, SUBACK,UNSUBSCRIBE UNSUBACK)

Un código de respuesta (REASON CODE) es un valor sin signo de un byte que indica el resultado de una operación.

- Los REASON CODE inferiores a 0x80 indican la finalización satisfactoria de una operación. **El REASON CODE normal exitoso es 0**.
- Los valores REASON CODE de 0x80 o más indican fallo.

Los paquetes de control CONNACK, PUBACK, PUBREC, PUBREL, PUBCOMP, DISCONNECT y AUTH tienen un solo REASON CODE como parte de la cabecera variable.

Connect

Después de que un cliente establece una conexión de red a un servidor, el primer paquete enviado desde el cliente al servidor debe ser un paquete CONNECT
Un cliente solo puede enviar el paquete CONNECT una vez a través de una conexión de red. El servidor debe procesar un segundo paquete CONNECT enviado desde un cliente como un error de protocolo y cerrar la conexión de red
El identificador de cliente (ClientId) identifica a cada cliente MQTT que se conecta a un broker
Flags del mensaje Connect

- Comienzo limpio (clean start). Valor 1: el cliente y el servidor deben descartar cualquier sesión existente e iniciar una nueva sesión

Connect

En MQTT, se utiliza la función de última voluntad o testamento para notificar a otros clientes acerca de un cliente desconectado. Cada cliente puede especificar su último mensaje cuando se conecta a un broker. El último mensaje es un mensaje MQTT normal con un tema, un indicador de mensaje retenido, QoS y carga útil.
El broker almacena el mensaje hasta que detecta que el cliente se ha desconectado inesperadamente. En respuesta a la desconexión inesperada, el broker envía el mensaje de última voluntad a todos los clientes suscritos del tema del mensaje de última voluntad. Si el cliente se desconecta correctamente con un mensaje DISCONNECT correcto, el intermediario descarta el mensaje almacenado.

- **Will Flag** Si contiene opciones de will Qos, etc
- **Will Qos** Define el Qos
- **Will Retain** Cuando se debe retener los mensajes

Otros parametros: **UserName and Password and Keepalive**

Connect Estructura

MQTT Connect Message Structure

Connect Clean Session True Client ID =PYTON1

Byte	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
Meaning	Header	Remaining Length	Length of prtocol name		Protocol Name +Version					Connect Flags	Keep Alive		Length								
Hex	0x10	0x13	0x0	0x4	0x4d	0x51	0x54	0x54	0x4	0x2	0x0	0x3c	0x0	0x7	0x70	0x79	0x74	0x68	0x6F	6E	0x31
Ascii		19		4	M	Q	T	T	4			60		7	P	Y	T	H	O	N	1

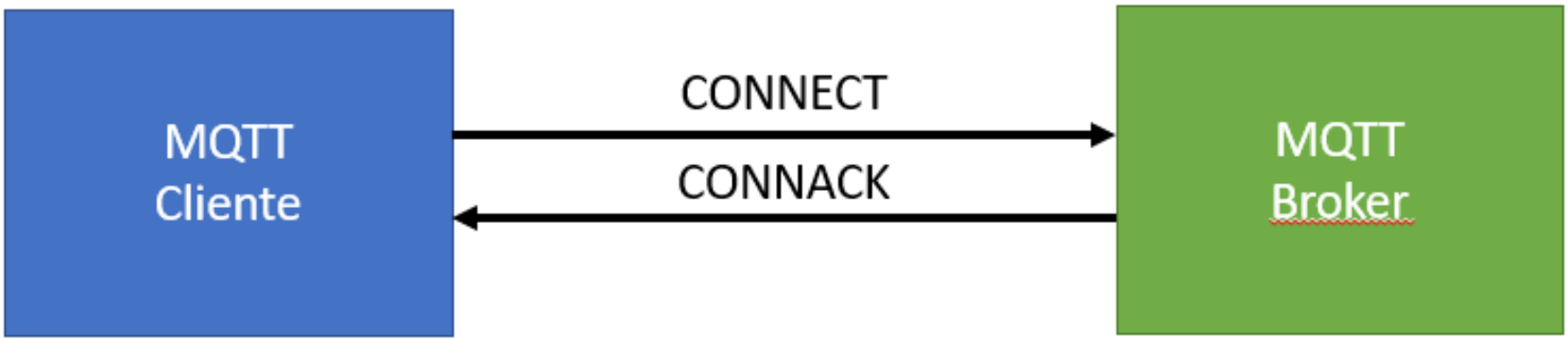
Notes:
Remaining Length = bytes 3 to 21
Length pf protocol name=4 =MQTT
length in bytes 13-14 -payload length =7
Connect Flags show Clean Session =True

Connect Flags

User name flag =	bit 7
Password Flag =	bit 6
Will Retain =	bit 5
Will QOS =	bit 5
Will QOS =	bit 4
Will Flag =	bit 2
Clean Session =	bit 1
Reserved =	bit 0

Connect Connack

La conexión MQTT siempre es entre un cliente y el broker(servidor). Los clientes nunca se conectan entre sí directamente. Para iniciar una conexión, el cliente envía un mensaje CONNECT al broker. El broker responde con un mensaje CONNACK y un código de estado. Una vez que se establece la conexión, el broker la mantiene abierta hasta que el cliente envía un comando de desconexión o se interrumpe la conexión.



Subscribe

El paquete SUBSCRIBE se envía desde el cliente al servidor para crear una o más suscripciones. Cada suscripción registra el interés de un cliente en uno o más temas. El paquete SUBSCRIBE también especifica (para cada Suscripción) la calidad de servicio máxima con la que el servidor puede enviar mensajes de aplicación al cliente

El payload de un paquete SUBSCRIBE contiene una lista de filtros de temas que indican los temas a los que el cliente desea suscribirse. Cada filtro de tema va seguido de un byte de opciones de suscripción. La carga útil (payload) debe contener al menos un par de opciones de suscripción y filtro de tema

Cuando el servidor recibe un paquete SUBSCRIBE de un cliente, el servidor debe responder con un paquete SUBACK El paquete SUBACK debe tener el mismo identificador de paquete que el paquete SUBSCRIBE que está reconociendo

Publish

Cada mensaje debe contener un tema que el broker pueda utilizar para reenviar el mensaje a los clientes interesados.
Cuando un cliente envía un mensaje a un broker MQTT para su publicación, el broker lee el mensaje, reconoce el mensaje (según el nivel de QoS) y procesa el mensaje. El procesamiento por parte del broker incluye determinar qué clientes se han suscrito al tema y enviarles el mensaje.
El cliente que publica inicialmente el mensaje solo se preocupa por entregar el mensaje PUBLISH al broker. Una vez que el broker recibe el mensaje PUBLISH, es responsabilidad del broker entregar el mensaje a todos los suscriptores. El cliente que transmite el PUBLISH (publicador) no recibe ningún comentario sobre si alguien está interesado en el mensaje publicado o cuántos clientes recibieron el mensaje del intermediario.

Publish

En la parte fija de la cabecera, se define:

• **DUP** 0 indica que esta es la primera vez que el Cliente o el Servidor ha intentado enviar este paquete Si el indicador DUP se establece en 1, indica que esto podría ser una nueva entrega de un intento anterior de enviar el paquete.

• **Qos** Define el Qos (ver la slide Qos) En función del tipo Qos, se debe responder con un PUBACK (Qos1) o PUBREC(Qos2)

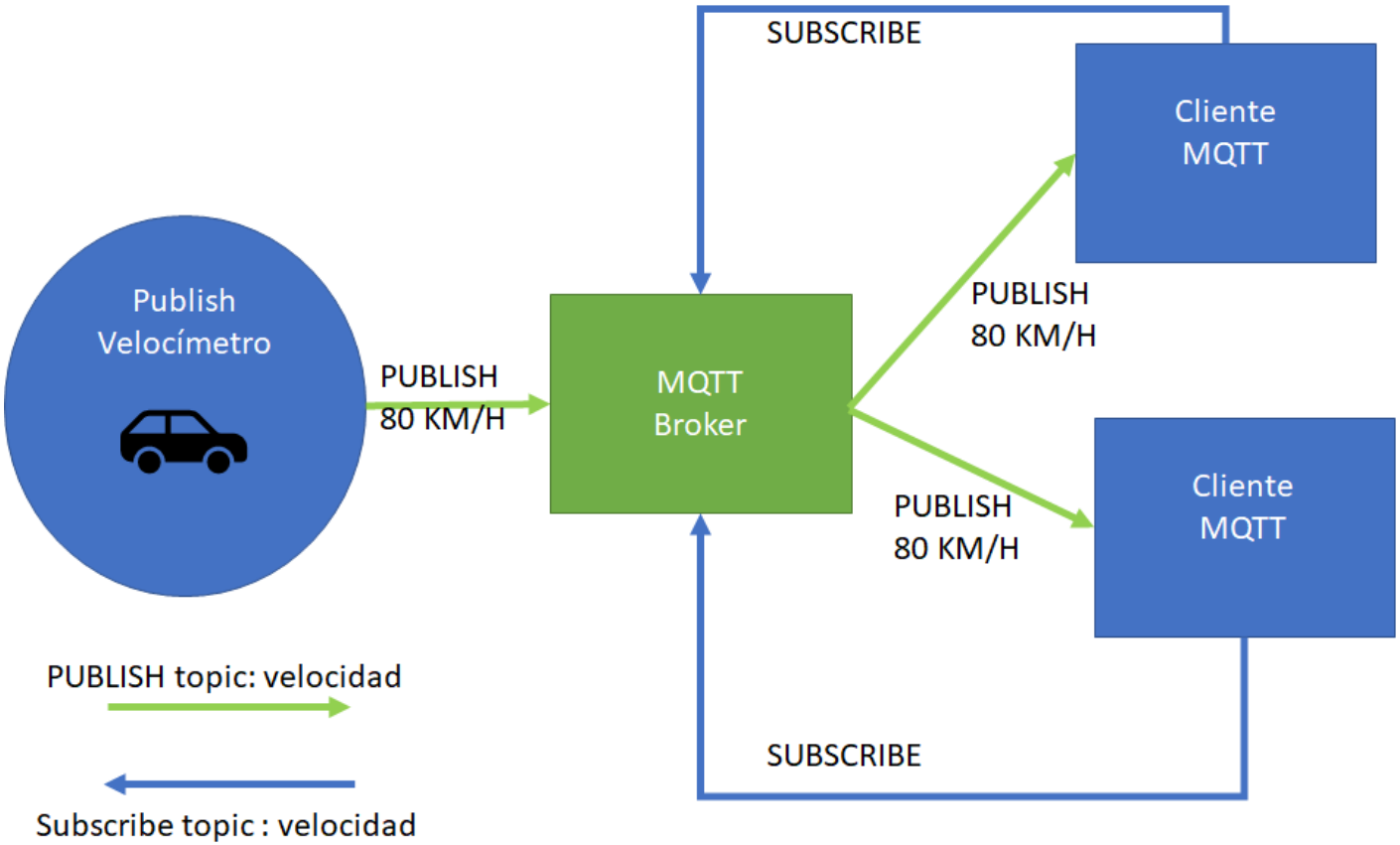
• **Retain** Si el indicador RETAIN se establece en 1 en un paquete PUBLISH enviado por un cliente a un servidor, el servidor debe reemplazar cualquier mensaje retenido existente para este tema y almacenar el mensaje de aplicacións

Además también incluye el nombre del tema topic compuesto por una cadena simple que está estructurada jerárquicamente con barras diagonales como delimitadores

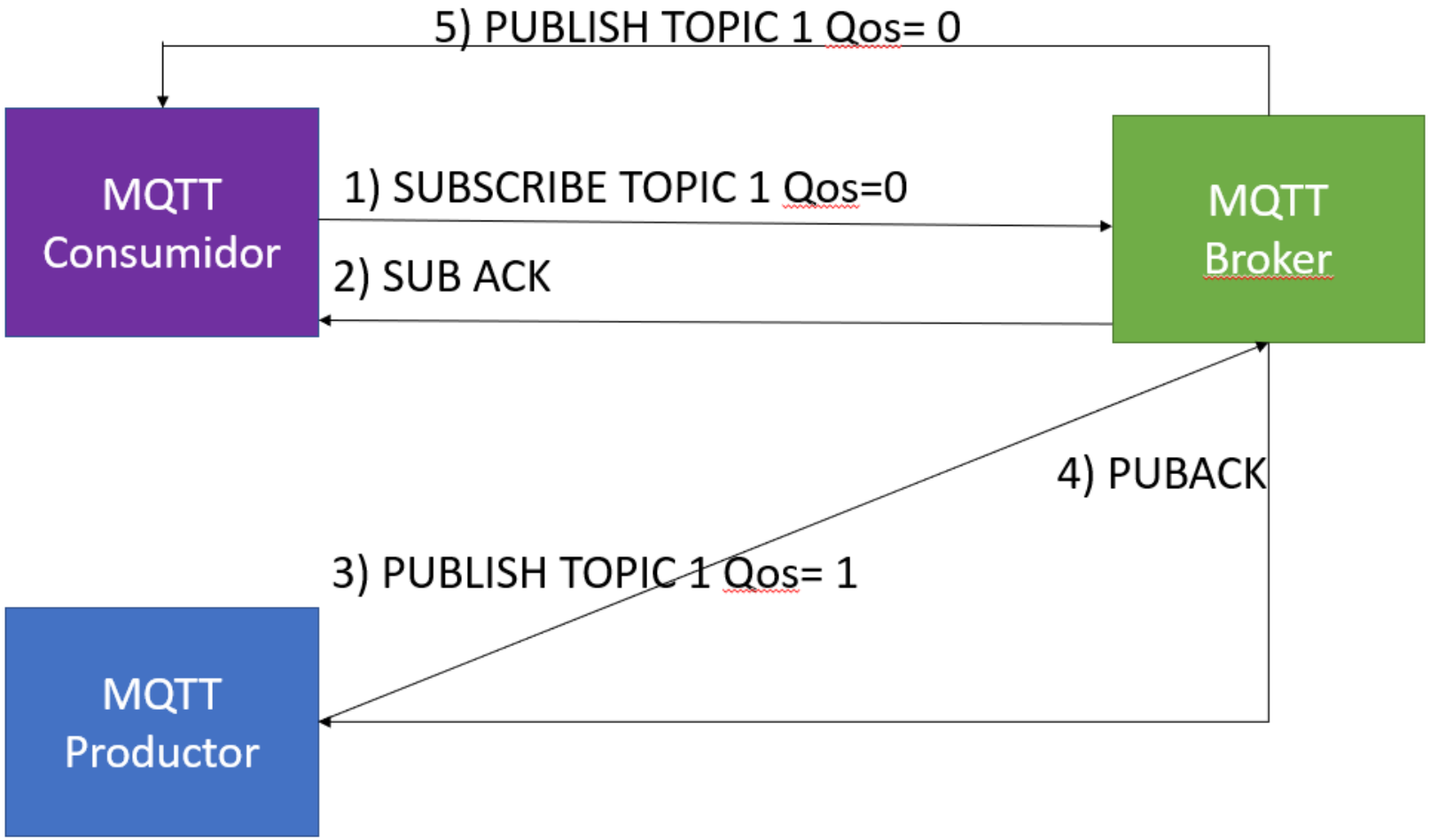
Subscribe-Publish

Publicar un mensaje no tiene sentido si nadie lo recibe. Es decir, si no hay clientes que se suscriban a los temas de los mensajes. Para recibir mensajes sobre temas de interés, el cliente envía un mensaje SUBSCRIBE al broker MQTT. El servidor envía paquetes PUBLISH al cliente para reenviar los mensajes de la aplicación que se publicaron a los temas que coinciden con estas suscripciones.

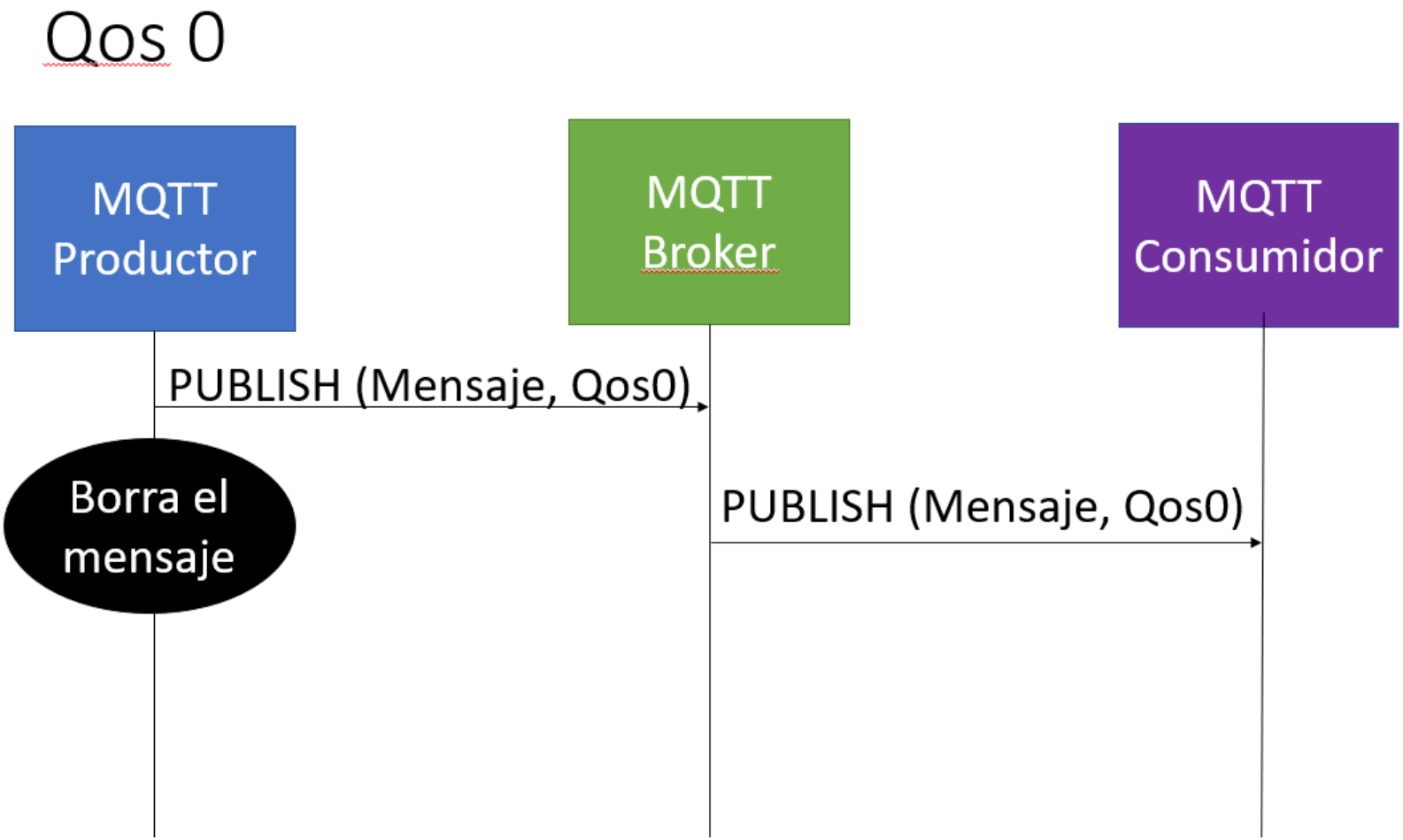
Subscribe-Publish (I)



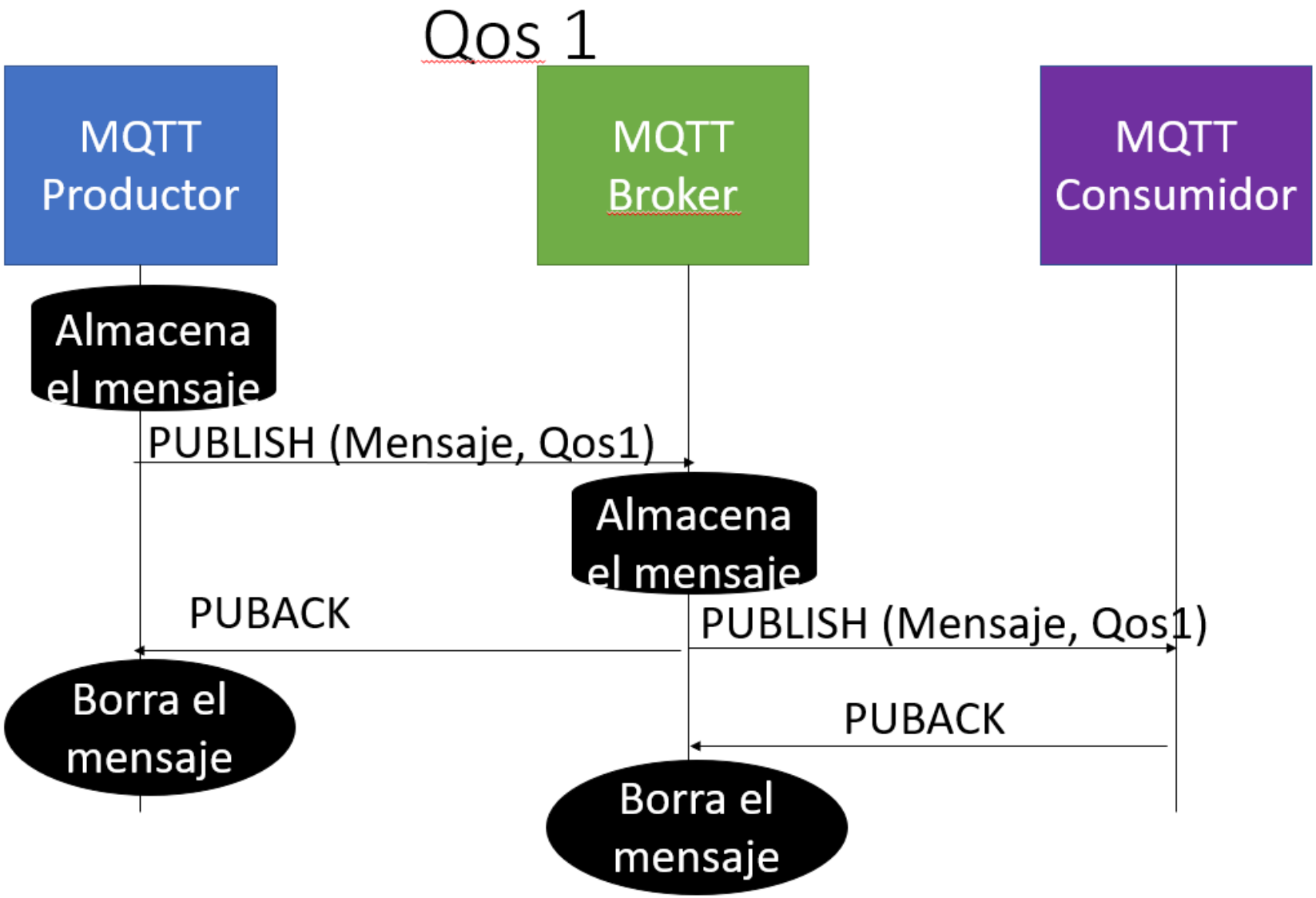
Subscribe Publish (II)



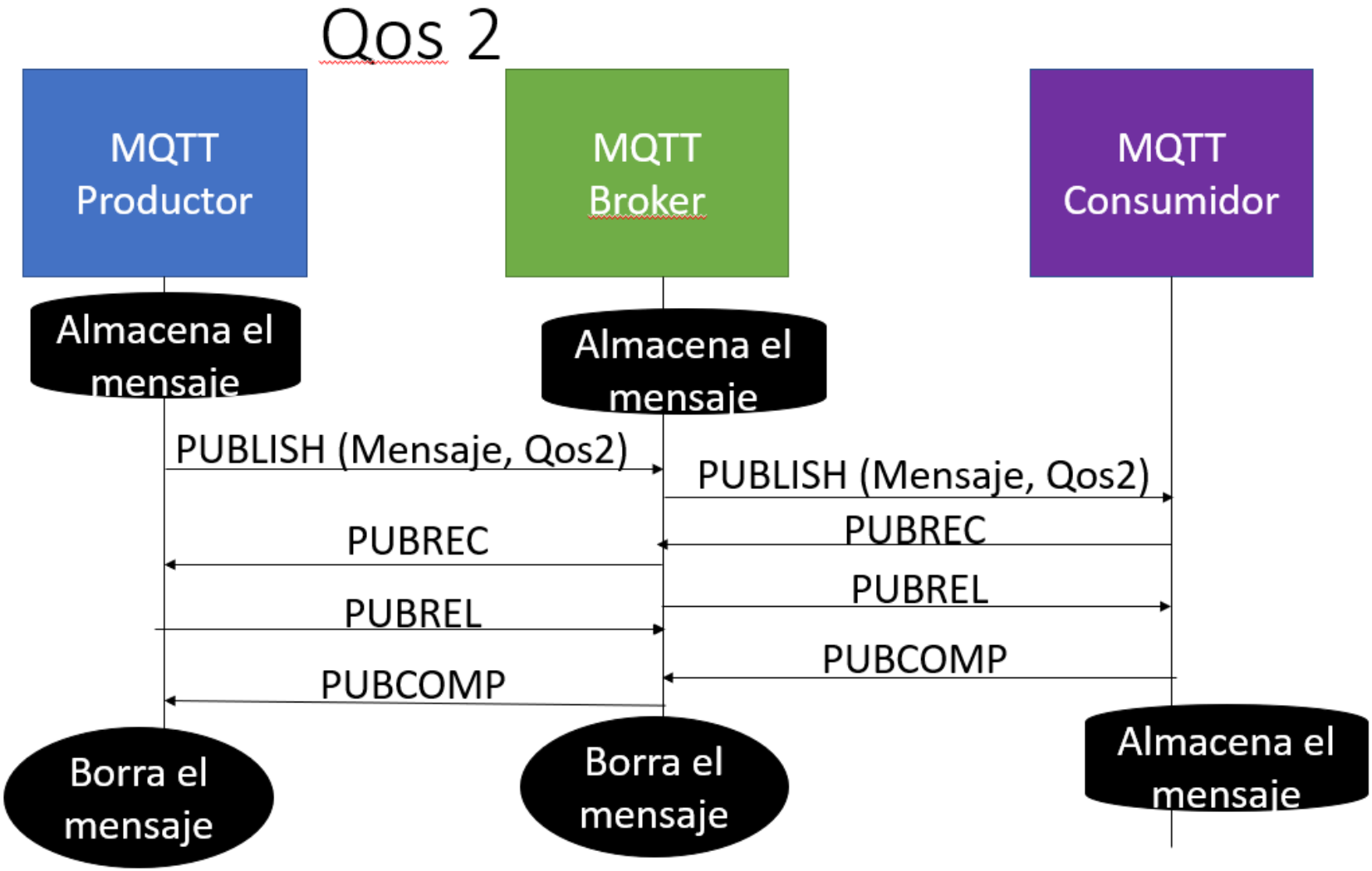
Subscribe Publish (Qos0)



Subscribe Publish (Qos1)



Subscribe Publish (Qos2)



Subscribe Publish (Tabla Qos)

Tabla de Qos:

QoS Publica	QoS Subscriptor	QoS del Mensaje Recibido
0	0	0
0	1	0
0	2	0
1	0	0
1	1	1
1	2	1
2	0	0
2	1	1
2	2	2

Sesiones persistentes

Las sesiones pueden persistir durante diferentes conexiones. Para ello, es necesario enviar el parámetro Clean Session a False.
Las subscripciones que se hagan permanecerán en el broker.
Las publicaciones, cuando coincida con la subscripción, **con un Qos mayor que 0** mientras se está desconectado se recibirán en el consumidor.
En versión 5 se utiliza el campo **Clean Flag** y el atributo Session Expiry establecer Clean Start a 1 y Session Expiry a 0 es equivalente a definir un valor de Clean Session a 1 en MQTT version 3

Almacenamiento de mensajes en el Broker

- Persistencia de sesiones con Clean Flag a false. Se guardan las subscripciones y se reciben aquellos mensajes que se hayan recibido offline con Qos=1 y Qos=2
- Mensajes de tipo Qos1,Qos2 que tengan que ser entregados a los consumidores
- Mensajes con retain activado

Ping -Disconnect

Por otro lado, para asegurar que la conexión está activa los clientes mandan periódicamente un mensaje PINGREQ que es respondido por el servidor con un PINGRESP.
Finalmente, el cliente se desconecta enviando un mensaje de DISCONNECT
En version 5, el servidor (broker) también puede mandar un DISCONNECT

Unsubscribe y Auth

Con el mensaje Unsubscribe se puede quitar la subscripción a un tema

En version 5 Auth son mensajes de autenticación. Este mensaje se utiliza para autenticar al cliente después de que se ha establecido una conexión exitosa entre el cliente y el servidor. En MQTT versión 5, el mensaje AUTH tiene nuevos campos de encabezado para la autenticación, incluyendo el tipo de autenticación y la información de autenticación

Diferencias entre versión 3 y versión 5

1. Características extendidas de sesión: MQTT versión 5 agrega nuevas características de sesión para mejorar la capacidad de los clientes para establecer y mantener conexiones con el servidor. .
2. Admite propiedades personalizadas: En MQTT versión 5, se pueden agregar propiedades personalizadas a los mensajes MQTT. Esto significa que los clientes y servidores pueden intercambiar información adicional y personalizada, lo que permite una mayor flexibilidad y personalización del protocolo.

Diferencias entre versión 3 y versión 5

3. Manejo mejorado de errores: MQTT versión 5 incluye un manejo mejorado de errores. Los clientes y servidores pueden informar y manejar errores con mayor detalle, lo que permite una mejor detección y recuperación de errores.
4. Mejoras en la seguridad: MQTT versión 5 presenta varias mejoras de seguridad, incluido el soporte para autenticación y autorización mejoradas, y el cifrado de mensajes a nivel de paquete.
5. Admite mensajes enriquecidos: MQTT versión 5 admite mensajes enriquecidos con encabezados personalizados y metadatos adicionales. Esto permite una mayor flexibilidad y personalización en la transmisión de mensajes.

Contenidos

Referencias

- T-REC-Y.2060, **Descripción general de Internet de los objetos**: `https://www.itu.int/rec/T-REC-Y.2060-201206-I/es`
- T-REC-Y.4101, **Internet de las cosas y ciudades y comunidades inteligentes**: `https://www.itu.int/rec/T-REC-Y.4101-201710-I/es`
- W. Stallings, Foundations of Modern Networking: SDN, NFV, QoE, IoT, and Cloud, Addison-Wesley Professional, Oct. 2015, ISBN: 0-13- 417539-5
- **MQTT**: `https://mqtt.org/mqtt-specification/` item **MQTT v5** `https://docs.oasis-open.org/mqtt/mqtt/v5.0/os/mqtt-v5.0`