Servicios en red: NFS y MQTT

Administración de Sistemas

Unai Lopez Novoa

unai.lopez@ehu.eus

eman ta zabal zazu

Universidad del País Vasco

Euskal Herriko Unibertsitatea

Contenido

1. Intercambio de ficheros en red
   - NFS
2. Intercambio de mensajes en red
   - MQTT

Introducción

- Los servicios en red son necesarios para habilitar la comunicación entre aplicaciones que funcionen en diferentes equipos o máquinas virtuales.

- En este tema nos vamos a enfocar en servicios para intercambiar ficheros y mensajes por red.

NFS

- Network File System (NFS) es un protocolo utilizado para sistemas de ficheros distribuidos.
  - La primera versión fue desarrollada en 1984.
  - La última versión es la v4.
    - v4.2 publicada en 2016: RFC 7862
- Esquema:

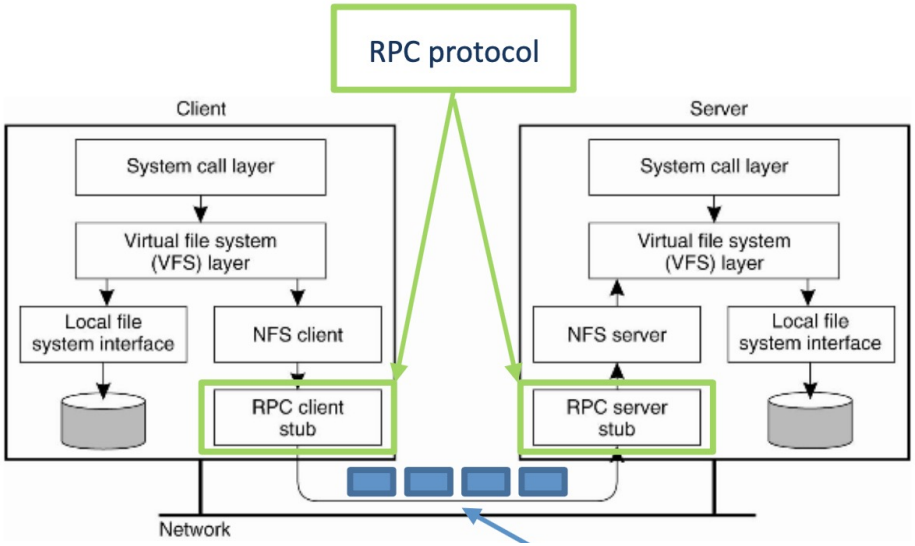


Imagen: José Ángel Herrero Velasco, OCW Unican, 2018

NFS

- El protocolo define una serie de reglas que permiten compartir ficheros sobre TCP/IP de una manera transparente.

- Es un protocolo sin estado.
  - Cada llamada tiene toda la información necesaria para completar la tarea.
  - El servidor no guarda información entre llamadas.

- Interoperabilidad entre NFS v2, v3 y v4.

NFS v4

- Arquitectura interna
  - Cliente: fichero de configuración de montaje
  - Servidor: Exports con el fichero /etc/exportfs

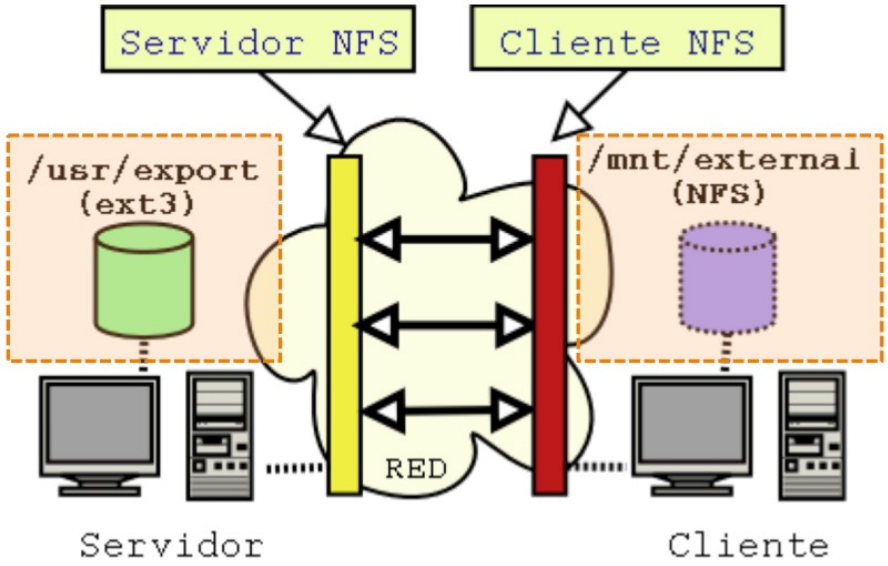
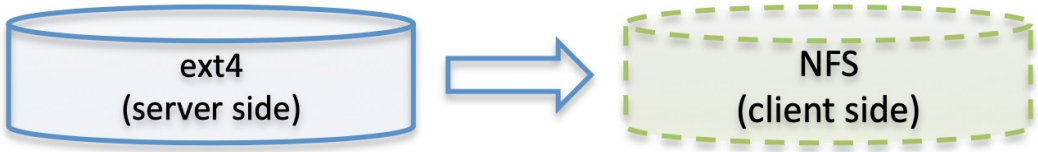


Imagen: José Ángel Herrero Velasco, OCW Unican, 2018

NFS v4

- Punto de vista del usuario:
  - No hay diferencias entre el sistema de ficheros local y remoto.



  - Transparencia en la usabilidad.
    - En los clientes, los datos se acceden a través de un punto de montaje.
  - La diferencia de rendimiento en el acceso dependerá del rendimiento de los discos y de la red
    - También del número de clientes que acceden al servidor NFS.
  - Visión uniforme de los ficheros.
    - Los ficheros del usuario están disponibles desde cualquier cliente de la red.

NFS v4

- Punto de vista del administrador del sistema:
  - Los datos están centralizados.
    - *Pro*: único punto gestión (actualizaciones, backups, ...)
    - *Con*: único punto de fallo
  - La optimización es más sencilla.
    - Único servidor en el que implementar RAID/LVM.
  - Tolerante a fallos (parcialmente).
    - *Pro*: los fallos críticos en los clientes no implican pérdida de datos.
    - *Con*: si el servidor se cae, todos los usuarios pierden acceso a sus ficheros.

NFS v4

- Instalación del servicio.
  - Ejemplos para Debian/Ubuntu.

- En el cliente:

```
apt update
apt install nfs-common
```

- En el servidor:

```
apt update
apt install nfs-kernel-server nfs-common
```

  - Requiere que el puerto TCP 2049 esté abierto.

NFS v4

- Configuración del servidor: fichero /etc/exports
  - Controla qué sistemas de ficheros se exportan a las máquinas remotas (y cómo).
  - Cada línea tiene la siguiente estructura:

    directorio cliente (opción1, opción2, ...)

    donde:

directorio	Es el directorio del servidor a compartir
cliente	IP o nombre de dominio, se pueden utilizar wildcards (*, ?)
opciones	Listado de 0 o más opciones

  - Por ejemplo:

```
/var/nfs/general  client_ip(rw,sync,no_subtree_check)
/home             client_ip(rw,sync,no_root_squash,no_subtree_check)
```

    - Las líneas en blanco se omiten y los comentarios se hacen con #

NFS v4

- Configuración del servidor: fichero /etc/exports
  - Opciones:

ro	Acceso sólo lectura
rw	Acceso de lectura y escritura
sync	Contestar a peticiones sólo cuando los cambios se hayan escrito a disco.
wdelay	Retrasa la escritura a disco si prevé que otra escritura a disco es inminente (opuesto a sync).
no_subtree_check	Impide la lectura de subdirectorios.
root_squash	Impide que usuarios conectados como “root” en los clientes tengan permisos root en el servidor y les asigna el usuario NFS nobody:nogroup
no_root_squash	Permite privilegios “root” (opuesto a root_squash).
acl	Activa el uso de listas de control de acceso (ACLs).

NFS v4

- Configuración del servidor: fichero /etc/exports
  - Si no se proporcionan opciones, NFS toma por defecto:
    - ro, wdelay, root_squash
  - El resto de las opciones se pueden consultar en:
    - https://linux.die.net/man/5/exports

NFS v4

- Configuración del servidor: permisos
  - Hay que tener en cuenta los permisos de las carpetas a compartir
  - Si el UID de un usuario en el servidor coincide con el UID de un usuario en el cliente, los permisos se mantienen.
  - Si root_squash está activo, las operaciones de root en cliente se registrarán como operaciones de nobody:nogroup en el servidor
    - Si hay alguna carpeta compartida en el servidor que pertenezca a root, es conveniente cambiar los permisos a nobody:

```
sudo chown nobody:nogroup /nfs/general
```

NFS v4

- Configuración del servidor
  - Comprobar los sistemas exportados por NFS
    - Definidos en /etc/exports

```
exportfs -v
```

  - Aplicar la configuración de NFS

```
exportfs -ra
```

  - Reiniciar el servicio NFS

```
service nfs-kernel-server restart
```

NFS v4

- Cliente: Crear un punto de montaje
  - Elegir un directorio para el montaje
    - Alternativamente, crear uno:

```
mkdir -p /tmp/nfs
```

  - Montar el directorio remoto usando la IP o nombre de dominio

```
sudo mount -t nfs 192.168.0.8:/var/nfs/general /tmp/nfs
```

  - Verificar que el montaje se ha hecho correctamente

```
df -h

Filesystem                    Size Used Avail Use% Mounted on
...
192.168.0.8:/var/nfs/general 25G  1.8G   23G   8% /tmp/nfs
```

NFS v4

- Cliente: Montaje permanente
  - Fichero /etc/fstab
    - Añadir una entrada para cada export NFS:

‹servidor-NFS›:/‹directorio› ‹directorio-local› nfs (opciones) 0 0

    - Opciones:

ro/rw	Montar como sólo lectura/lectura y escritura
hard	En caso de desconexión del servidor NFS, las aplicaciones usando NFS esperan hasta re-conexión y no se pueden matar.
soft	En caso de desconexión del servidor NFS, las aplicaciones esperan un tiempo determinado y después lanzan un error.
intr	Junto con la opción *hard*, permite que las aplicaciones esperando por la re-conexión del servidor NFS se puedan matar.
timeo	Junto con la opción *soft*, define el tiempo a esperar.
noexec	Impide la ejecución de binarios o scripts en un directorio NFS.

Ejercicio 1

- *Este ejercicio se realizará en parejas*
  - Un miembro de la pareja será **C**, el otro **S**. Cada uno usa su MV.
  - **C** será el cliente NFS, **S** el servidor NFS

1) **S** comparte por NFS una carpeta */compartir*
   - La carpeta debe contener un fichero *texto.tx*t con texto aleatorio.
   - Compartir de forma que **C** sólo pueda leer, no modificar.
2) **C** monta la carpeta en */tmp/compartido*
   - Verificar que el fichero no se puede modificar.
3) **S** modifica las opciones NFS para permitir cambios en los ficheros de la carpeta.
   - **C** realiza algún cambio en el fichero, **S** lo verifica.

MQTT

- Intercambiar ficheros es fundamental para compartir datos y configuraciones en sistemas distribuidos.
- *Pero…*
- En muchas situaciones es mucho más ágil intercambiar mensajes.
  - Evita las complejidades de manipular ficheros.

MQTT

- **M**essage **Q**ueue **T**elemetry **T**ransport

- Es un protocolo orientado a paso de mensajes en formato publicar-suscribir
  - En contraposición al cliente-servidor.
  - Separa al cliente que envía mensajes del que recibe mensajes.

- Web: https://mqtt.org/

MQTT

- Es muy usado en entornos Internet of Things (IoT)
  - Permite gestionar cientos/miles de clientes ligeros.

- Por ejemplo:

Logística

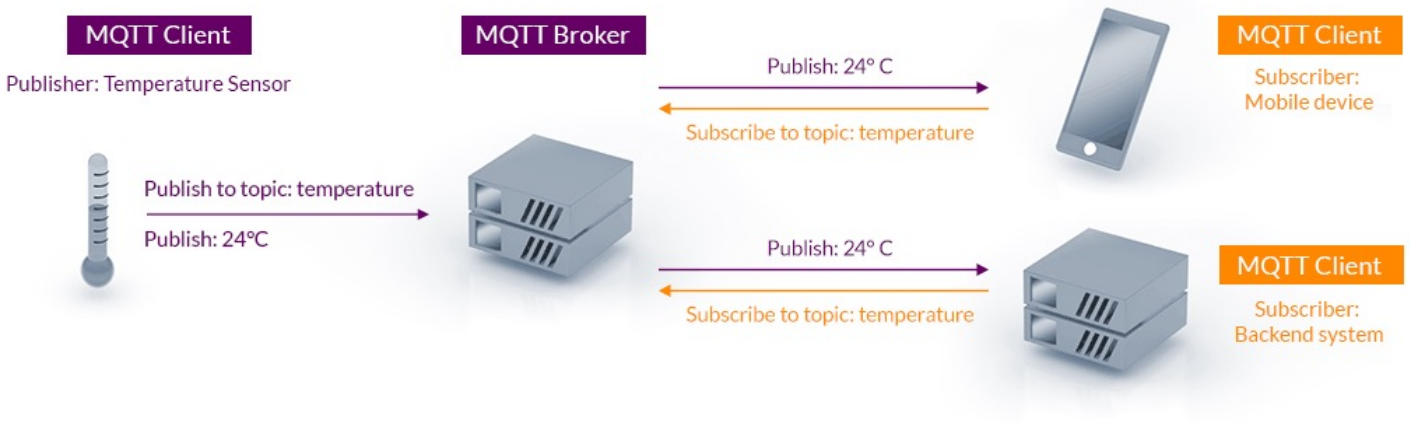
Industria

Domótica

Imágenes: mqtt.org/

MQTT

- Arquitectura:



- Partes:
  - *Publishers*: Envían mensajes.
  - *Broker*: Recibe mensajes y los distribuye a los suscriptores.
  - *Subscribers*: Reciben mensajes enviados por los Publisher.

Imagen: mqtt.org/

MQTT: Topics

- El proceso Broker filtra los mensajes en base a *topics*.
  - Un *topic* es un String de texto que identifica mensajes de una temática concreta, por ejemplo:

    edificio/planta 1/sala7/temperatura

  - Los *topic:*
    - Pueden ser multi-nivel, se separan mediante /
    - Son sensibles a mayúsculas-minusculas y pueden contener espacios
    - No deben comenzar por $

- También se pueden filtrar por contenido o tipo.
  - Pero no son las formas más usadas.

MQTT: Topics

- Se pueden utilizar *wildcards* al dirigirse a *topics*:

Un sólo nivel: +

- myhome / groundfloor / livingroom / temperature
- myhome / groundfloor / kitchen / temperature
- myhome / groundfloor / kitchen / brightness
- myhome / firstfloor / kitchen / temperature
- myhome / groundfloor / kitchen / fridge / temperature

Multi-nivel: #

- myhome / groundfloor / livingroom / temperature
- myhome / groundfloor / kitchen / temperature
- myhome / groundfloor / kitchen / brightness
- myhome / firstfloor / kitchen / temperature

Imágenes: https://www.hivemq.com/blog/mqtt-essentials-part-5-mqtt-topics-best-practices/

Mosquitto

- Framework que implementa MQTT
  - Open source
  - Web: https://mosquitto.org/
- Es muy utilizado en entornos IoT
  - Es ligero, configurable y extensible mediante plug-ins
- En estas diapositivas veremos su uso básico.

Mosquitto

- Instalación
  - En Debian/Ubuntu:

```
apt update
apt install mosquitto mosquitto-clients
```

- Verificar que el servicio está en marcha

```
service mosquitto status
```

- Por defecto, utiliza el puerto 1883.

Mosquitto

- Clientes de línea de comando
  - Publicar un mensaje en un *topic*

```
mosquitto_pub -h <host> -t <topic> -m <mensaje> <opc>
```

    - donde:
      - host: IP del Broker
      - topic: String de texto que indica el topic
      - mensaje: String de texto con el mensaje a publicar
      - opciones: P.e. –d para activar el modo *debug* y mostrar más información

    - Ejemplo:

```
mosquitto_pub -h 127.0.0.1 -t "miTopic" -m "Hola" -d
```

Mosquitto

- Clientes de línea de comando
  - Suscribirse a los mensajes de un *topic*

```
mosquitto_sub -h <host> -t <topic> <opc>
```

    - donde:
      - host: IP del Broker
      - topic: String de texto que indica el topic
      - opciones: P.e. –d activa el modo *debug*, -v activa el modo *verbose*

    - Ejemplo:

```
mosquitto_sub -h 127.0.0.1 -t "miTopic" -v -d
```

Mosquitto

- Configuración del servicio:
  - Fichero /etc/mosquitto/mosquitto.conf

```
# Place your local configuration in /etc/mosquitto/conf.d/
...
log_dest file /var/log/mosquitto/mosquitto.log
include_dir /etc/mosquitto/conf.d
```

  - Añadir las siguientes líneas para:
    - Permitir conexiones al puerto 1883 desde fuera del servidor

```
listener 1883 0.0.0.0
```

    - Permitir conexiones anónimas (sin usar usuario/contraseña)

```
allow_anonymous true
```

Mosquitto

- Por defecto, MQTT sólo entrega mensajes a los suscriptores de un *topic* que estén conectados.
  - Un suscriptor no recibirá mensajes que se publiquen mientras esté desconectado del Broker.

- Se puede hacer que los suscriptores reciban el último mensaje que se envió al *topic*.
  - Se denominan *mensajes retenidos*
  - Se puede publicar un mensaje retenido con el parámetro "-r"
    - Ejemplo:

```
mosquitto_pub -h … -t … -m "Mi nuevo mensaje" -r
```

Ejercicio 2

- *Este ejercicio se realizará en parejas (esta vez, seréis **A** y **B**)*
  - **A** será un Publisher MQTT, **B** será un Subscriber MQTT
  - **B** ejecutará el servidor y Broker Mosquitto

Abrir el puerto TCP 1883 en GCP

- **B** se suscribe a un *topic* "ciudades/bizkaia"
- **A** publica un mensaje "Bilbao" en "ciudades/bizkaia"
  - **B** debe verificar que lo recibe

- **B** se suscribe a un *topic* "ciudades" y todos sus *sub-topics*.
- **A** publica un mensaje "Donostia" en "ciudades/gipuzkoa"
  - **B** debe verificar que lo recibe

- **B** se desconecta del Broker Mosquitto
- **A** envía un mensaje retenido "Gasteiz" en "ciudades/araba"
- **B** se suscribe a "ciudades/#". Debe recibir el mensaje "Gasteiz".

MQTT

- MQTT es muy utilizado en entornos donde hay restricciones de conectividad.

- En función de las necesidades de la aplicación, MQTT permite definir diferentes niveles de tolerancia a la pérdida de mensajes.
  - Esto se define mediante el parámetro Quality of Service (QoS)

MQTT: Casos de uso

- Escenario 1: Agricultura, monitorización de cultivos
  - Posibles restricciones:
    - Dispositivos alimentados por batería
    - Envío de datos por redes móviles

Imagen: https://easternpeak.com/blog/iot-in-agriculture-technology-use-cases-for-smart-farming-and-challenges-to-consider/

MQTT: Casos de uso

- Escenario 2: Industria, monitorización de mecanizados
  - Posibles restricciones:
    - Mantener un orden en los mensajes
    - Evitar mensajes duplicados

Imagen: https://cfaa.eus/instalaciones-e-investigacion/

MQTT: QoS

- Nivel 0
  - Nivel por defecto, el mensaje llegará una vez o no llegará.
  - *Fire and forget*: los datos se envían sin esperar confirmaciones.
  - El más rápido, el que menos ancho de banda consume.

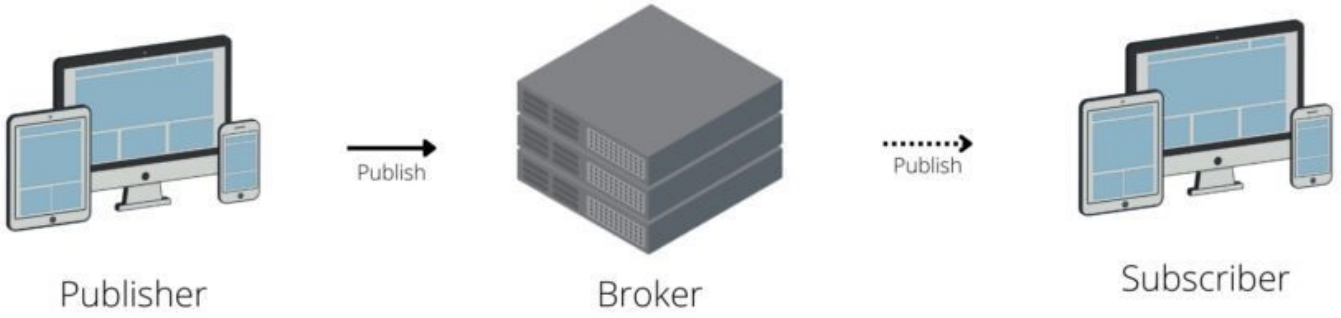


Imagen: https://cedalo.com/blog/understanding-mqtt-qos/

MQTT: QoS

- Nivel 1
  - Asegura que cada mensaje llega al menos una vez.
  - El receptor confirma la recepción con un mensaje *PUBACK*.
    - Si el emisor no recibe *PUBACK* en un tiempo determinado, re-envía el mensaje.
    - Los receptores pueden recibir mensajes duplicados.

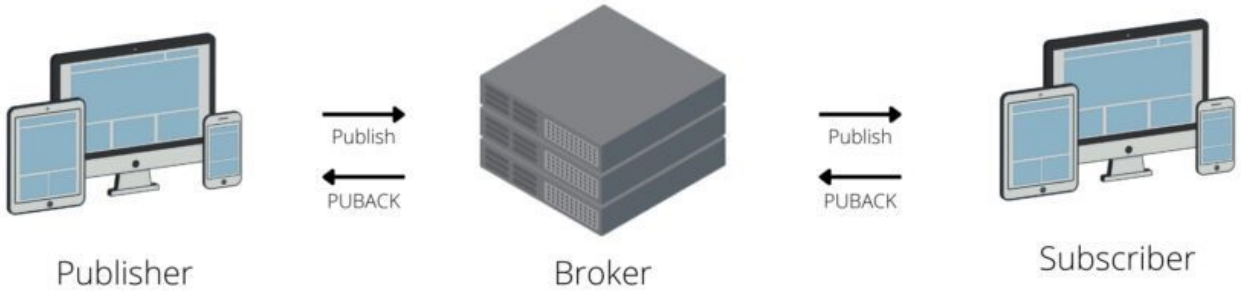


Imagen: https://cedalo.com/blog/understanding-mqtt-qos/

MQTT: QoS

- Nivel 2
  - Asegura que cada mensaje se entrega una vez.
  - El receptor confirma la recepción con un mensaje *PUBREC*.
    - Si el emisor no recibe *PUBREC* en un tiempo determinado, re-envía el mensaje.
  - Tras recibir *PUBREC*, el emisor envía un mensaje *PUBREL* al receptor para confirmar la recepción de *PUBREC*.
    - El receptor confirma la recepción de PUBREC con *PUBCOMP*

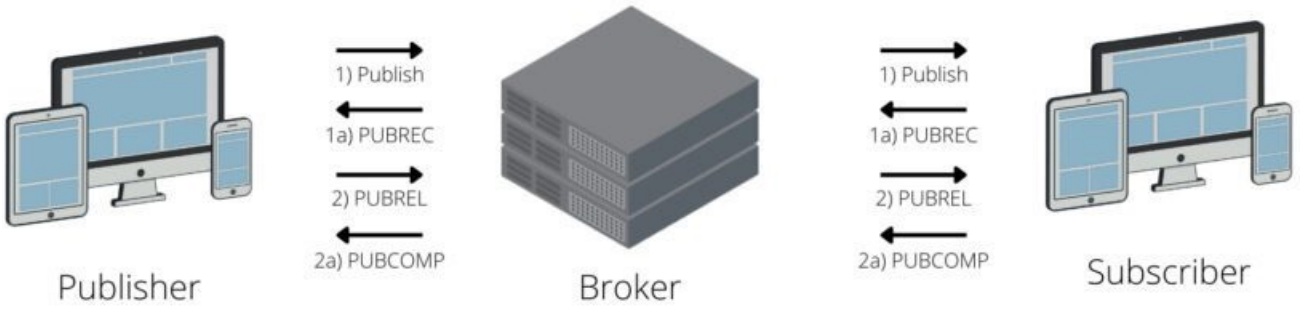


Imagen: https://cedalo.com/blog/understanding-mqtt-qos/

MQTT: QoS

- Nivel 2
  - Cada suscriptor y bróker mantiene una cola de los mensajes recibidos por cada tópico.
    - Comprueba cada mensaje para evitar duplicados.

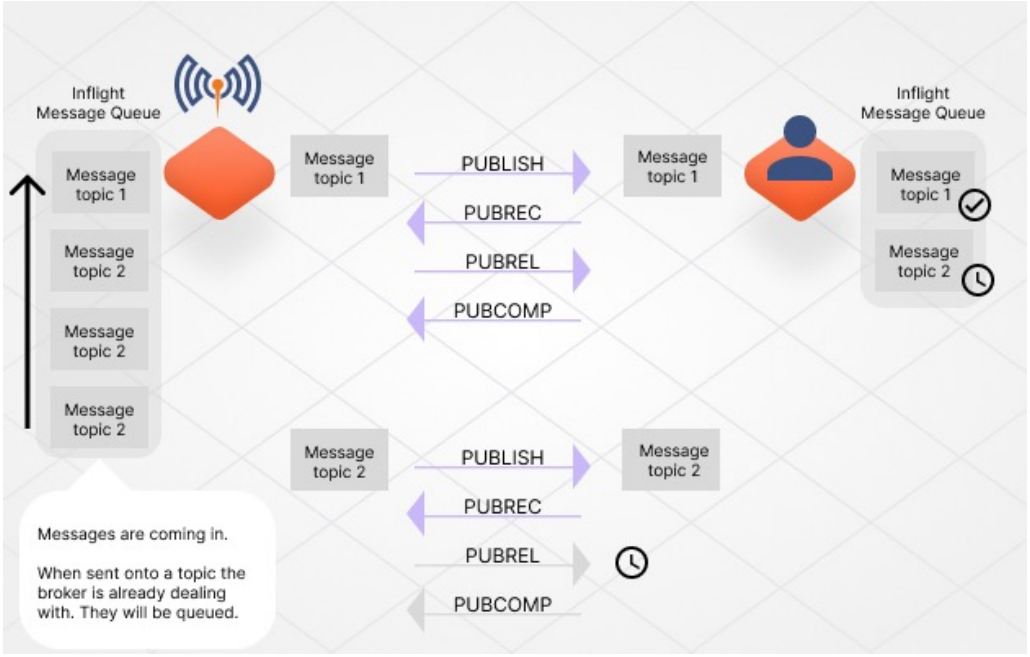


Imagen: https://cedalo.com/blog/understanding-mqtt-qos/

MQTT: QoS

- El emisor y receptor de un mensaje no tienen por qué utilizar el mismo nivel de QoS.
  - En función del nivel de cada uno, la QoS general será diferente:

QoS Emisor-Broker	QoS Broker-Suscriptor	QoS General
0	1 o 2	0
1 o 2	0	0
2	1	1
1	1	1

Tabla: https://cedalo.com/blog/understanding-mqtt-qos/

MQTT: QoS

- Indicar la calidad de servicio en Mosquitto
  - Utilizar el parámetro -q, donde N es el nivel deseado (0, 1 o 2)
    - Para suscribirse a un tópico:

```
mosquitto_sub … -q N
```

    - Para publicar en un tópico:

```
mosquitto_pub … -q N
```

  - El parámetro –d visualiza los mensajes enviados/recibidos.

Mosquitto

- Se puede evitar que cualquier cliente envíe mensajes a un tópico mediante técnicas de autenticación.

- En Mosquitto, dos formas:
  - Usando ficheros:
    - Más simple, la veremos en este tema
  - Usando plug-ins:
    - P.e. https://mosquitto.org/documentation/dynamic-security/

Mosquitto: Cuentas de usuario

- Crear un fichero con los usuarios de Mosquitto
  - Contendrá las cuentas de usuario y sus contraseñas cifradas.
  - Debe cargarse en el Broker.

  - Crear fichero de usuarios con un primer usuario

```
mosquitto_passwd -c /…/misUsuarios mikel
```

  - Añadir un nuevo usuario o modificar contraseña de uno existente

```
mosquitto_passwd /…/misUsuarios mikel
```

  - Eliminar usuarios del fichero de usuarios

```
mosquitto_passwd -D /…/misUsuarios mikel
```

Mosquitto: Cuentas de usuario

- Cargar el fichero de usuarios en la configuración de Mosquitto.
  - En el fichero de configuración de Mosquitto, indicar ruta al fichero de usuarios.
    - Es recomendable desactivar las conexiones anónimas (sin usuario).

```
# Place your local configuration in /etc/mosquitto/conf.d/
...
#allow_anonymous true
password_file /…/misUsuarios
```

  - Reiniciar servicio Mosquitto

Mosquitto: Cuentas de usuario

- Usar autenticación:
  - Para suscribir a un tópico

```
mosquitto_sub … -u <usuario> -P <contraseña>
```

  - Para publicar en un tópico

```
mosquitto_pub … -u <usuario> -P <contraseña>
```

Mosquitto

- Con las cuentas de usuarios gestionamos el acceso.
- *Pero …*
- No indicamos qué permisos tiene cada usuario.
  - Solución: listas de control de acceso (Access Control Lists – ACLs)

Mosquitto: ACLs

- Definir los permisos de cada usuario mediante un fichero de ACLs.
  - Fichero de ejemplo:

```
# Dar acceso de lectura a los mensajes de "ciudades" a usuario1
user usuario1
topic read ciudades/#

# Dar acceso de lectura y escritura al topico ciudades/bizkaia a usuario2
user usuario2
topic readwrite ciudades/bizkaia/#

# Dar acceso de lectura y escritura a todos los topicos a usuario3
user usuario3
topic readwrite #
```

Ejemplo adaptado de: https://medium.com/jungletronics/mosquitto-acls-ac062aea3f9

Mosquitto: ACLs

- Incluir la configuración de ACLs en el Broker.
  - En el fichero de configuración de Mosquitto, indicar la ruta al fichero de ACLs.

```
# Place your local configuration in /etc/mosquitto/conf.d/
...

acl_file /…/miConfiguracionACL
```

  - Reiniciar el servicio de Mosquitto para que entren en funcionamiento.

Ejercicio 3

- *Este ejercicio se realizará en parejas, seréis **A** y **B***
  - **A** será un Publisher MQTT, **B** será un Subscriber MQTT
  - **B** ejecutará el servidor y Broker Mosquitto

- **B** crea dos cuentas de usuario para A y B en el Broker
  - La cuenta de A puede escribir y leer en un *topic* "colores".
  - La cuenta de B puede leer en un *topic* "colores".
- **B** se suscribe al *topic* "colores" y **A** publica un mensaje.
  - **B** verifica que recibe el mensaje.
- **A** se suscribe al *topic* "colores" y **B** publica un mensaje.
  - **A** verifica que no lo recibe.

Bibliografía I

- V. Puente Varona, J.A. Herrero Velasco, P. Abad Fidalgo. "Computer System Design and Administration", OCW UNICAN, 2018:
  - Topic 7: Network file system service: NFSv4.
  - Publicado bajo licencia Creative Commons BY-NC-SA 4.0
  - https://ocw.unican.es/course/view.php?id=245
- Brian Boucheron "How To Set Up an NFS Mount on Ubuntu 20.04", Digital Ocean, 2020:
  - https://www.digitalocean.com/community/tutorials/how-to-set-up-an-nfs-mount-on-ubuntu-20-04

- Consultados en julio 2020

Bibliografía II

- Mosquitto Documentation, 2022.
  - https://mosquitto.org/documentation/
- L. Llamas, “Cómo instalar Mosquitto”, luisllamas.es, 2020.
  - https://www.luisllamas.es/como-instalar-mosquitto-el-broker-mqtt/
- MQTT Essentials: What is MQTT Quality of Service (QoS) 0,1, & 2?, 2023.
  - https://www.hivemq.com/blog/mqtt-essentials-part-6-mqtt-quality-of-service-levels/

- Consultados en septiembre 2022 y 2023.