Servicios en red: NFS y MQTT

INTRODUCCIÓN

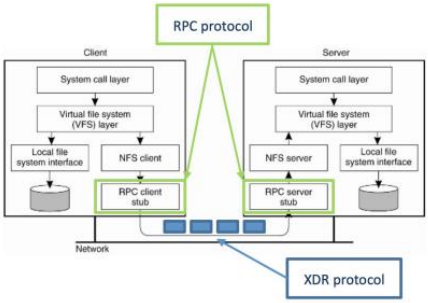
- Servicios en red → Habilitan la comunicación entre aplicaciones que funcionen en diferentes equipos o VM

INTERCAMBIO DE FICHEROS EN RED

SISTEMAS DE FICHEROS DISTRIBUIDOS

- Acceder a datos de equipos remotos de forma transparente → Almacenar, recuperar, listar, compartir y asegurar la seguridad de los datos
- Clasificación:
  - DFS y sistemas Cluster: GlusterFS, HDFS, …
  - Sistemas en la nube: Dropbox, Oracle Cloud, …
  - Sistemas de ficheros paralelos: PVFS2, orangeFS, …
  - **Sistemas de ficheros en red**: NFS, pNFS, …

NFS



Network File System (NFS) → Protocolo para sistemas de ficheros distribuidos

- V1 (1984) → V4 (2016)
- Interoperabilidad entre NFS v2, v3 y v4

Serie de reglas para compartir ficheros sobre TCP/IP de forma transparente.

Protocolo sin estado:

- Cada llamada la información necesaria para si misma
- No se guarda información entre llamadas en el server

NFS4

Usuario:

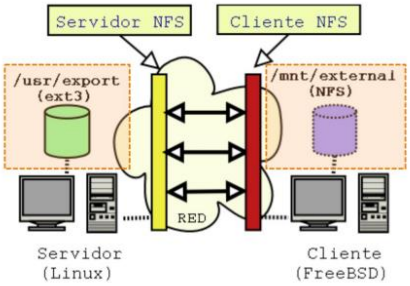
- No ve diferencias entre el sistema de ficheros local y remoto

- Transparencia → Los datos se acceden a través de punto de montaje (clientes)
- El rendimiento en el acceso depende del rendimiento de discos y red + Número de clientes accediendo
- Visión uniforme de los ficheros → Los ficheros del usuario estan disponibles desde cualquier cliente en la red

Administrador:

- Datos centralizados
  - Pro: único punto de gestión
  - Con: único punto de fallo
- Optimización más sencilla → Único servidor para RAID/LVM

- Aislamiento → Discos en servidores no accesibles para los usuarios
- Tolerante a fallos (parcialmente)
    - Pro: fallos críticos en los clientes no acarrean pérdidas de datos
    - Con: si el servidor cae, el acceso a los ficheros también.
- Único puerto: TCP 2049



Arquitectura interna:

- Cliente → Fichero de configuración de montaje
- Servidor → Exports con el fichero /etc/exportsfs

NFS V4: COMANDOS

Instalación

- Instalación en cliente:

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

- Instalación en servidor:

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

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

    Directorio cliente (opción1, opción2)

    Donde:

Directorio	Directorio del servidor a compartir
Cliente	IP o nombre del dominio (puede tener wildcards (*,?))
Opciones	Listado de 0+ opciones

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

- Opciones:
    - ro Read only
    - rw Read/Write
    - sync Contestar a peticiones sólo cuando los cambios se hayan escrito a disco
    - wdelay Retrasa la escritura a disco si prevé otra escritura 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)
- Si no se proporcionan opciones, NFS toma por defecto: ro, wdelay, root_squash

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
- 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
```

- Configuracion 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

Cliente: crear punto de montaje

- 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)

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.

INTERCAMBIO DE MENSAJES EN RED

Message Queue Telemetry Transport. Mucho mas ágil que intercambiar ficheros (mensajes cortos), evita la complejidad de manipular ficheros. Muy usado en entorno IoT.

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.

- Publisher: Envia mensajes
- Broker: Recibe mensajes y los distribuye a los subscriptores
- Subscribers: Reciben mensajes de los publishers

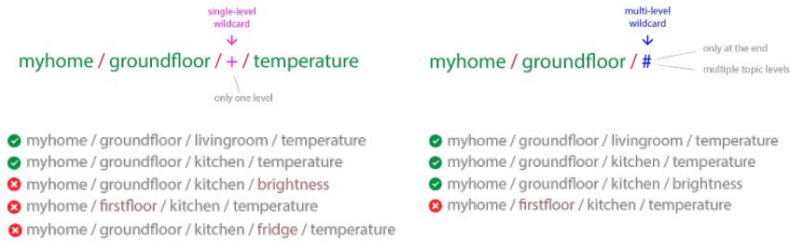
TOPPICS

El proceso Broker filtra los mensajes en base a topics → String de texto que identifica mensajes de una temática concreta, por ejemplo:

edificio/planta 1/sala7/temperatura

- 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 → No son las formas más usadas.

Se pueden usar wildcards:



MOSQUITTO

Instalación:

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

```
service mosquitto status
```

Clientes:

- Publicar 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
```

- Subscribirse a 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
```

Configuracion del servicio (/etc/mosquitto/mosquitto.conf)

- Añadir las siguientes líneas:

```
listener 1883 0.0.0.0        # Permitir conexiones al puerto 1883 desde fuera
allow_anonymous true         # Permite conexiones anónimas
```

Por defecto, MQTT sólo entrega mensajes a los suscriptores de un topic que estén conectados (no recibirá mensajes que se publiquen mientras esté desconectado del Broker). Pero se puede hacer que los suscriptores reciban el último mensaje que se envió al topic → Mensajes retenidos ("-r")

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

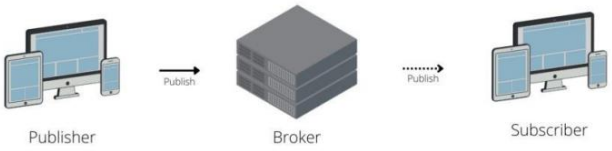
Muy utilizado en entornos donde hay restricciones de conectividad. En función de las necesidades de la aplicación, MQTT permite definir diferentes niveles de tolerancia a la pérdida de mensajes mediante el parámetro Quality of Service (QoS)

CASOS DE USO

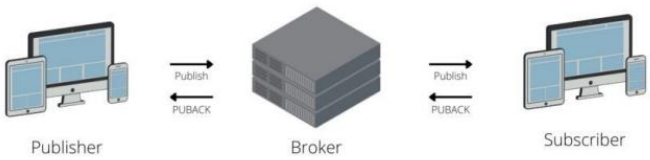
- Escenario 1: Agricultura, monitorización de cultivos
  - Posibles restricciones:
    - Dispositivos alimentados por batería
    - Envío de datos por redes móviles
- Escenario 2: Industria, monitorización de mecanizados
  - Posibles restricciones:
    - Mantener un orden en los mensajes
    - Evitar mensajes duplicados

QoS

- Nivel 0 → 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



- 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



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

El emisor y el receptor no tienen por que usar el mismo QoS → La QoS cambia si hay diferentes

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

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

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

- Ficheros
- Plug-ins

Crear un fichero con los usuarios de Mosquitto (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
```

Cargar el fichero de usuarios en la configuración de Mosquitto → Indicar ruta al fichero de usuarios (y añadir permisos). Es recomendable desactivar las conexiones anónimas y luego reiniciar servicio

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

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

acl_file /…/miConfiguracionACL
```

Usar autentificación:

- Subscribirse:

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

- Publicar:

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

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 #
```