**2.1. TEMA: Servicios en red: NFS y MQTT**

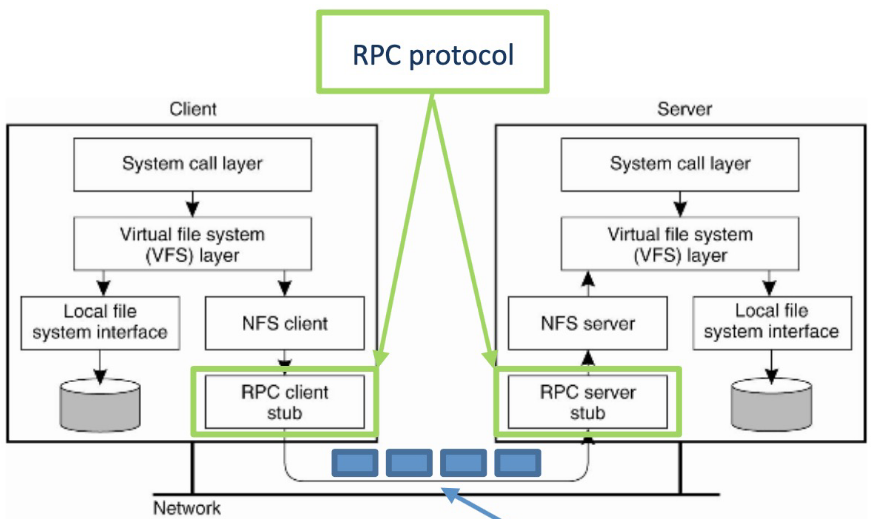
**2.1.1. Introducción:**

Servicios en red:

* Son necesarios para habilitar la comunicación entre aplicaciones que funcionen en:
  + Diferentes equipos.
  + Máquinas virtuales.
* Vamos a enfocarnos en servicios para intercambiar ficheros y mensajes por red.

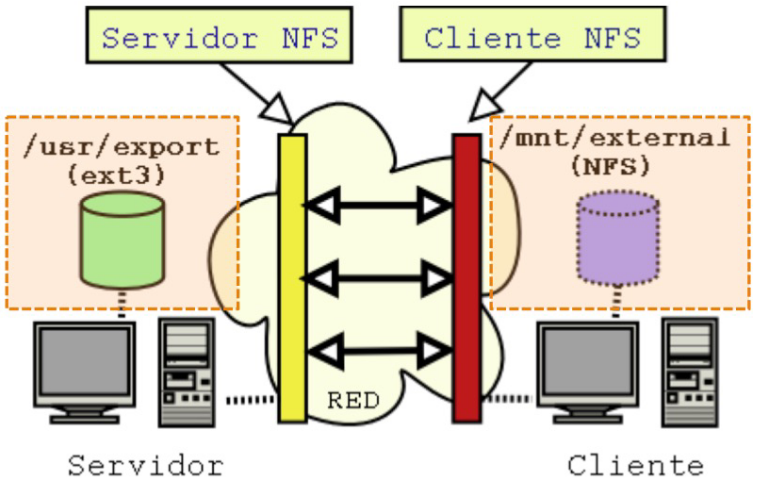
**2.1.2. Intercambio de ficheros en red (NFS):**

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.
* Esquema:
* Define una serie de reglas para 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.



* 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.
* 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 se implementa 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.
* 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 tener abierto el puerto 2049.
* 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, …)

* + - directorio → El directorio del servidor a compartir.
    - cliente → IP o nombre de dominio, se pueden utilizar wildcards (\*, ?)
    - opciones → Listado de 0 o más opciones
  + Ejemplos:
    - /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 #.
  + Opciones:
    - ro → Acceso sólo lectura
    - rw → Acceso de lectura y escritura
    - sync → Contestar a peticiones sólo cuando los cambios se escriban 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).
  + 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>
* 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
* Configuración del servidor:
  + Comprobar los sistemas exportados por NFS → exportfs -v
    - Están definidos en /etc/exports.
  + Aplicar la configuración de NFS → exportfs -ra
  + Reiniciar el servicio NFS → service nfs-kernel-server restart
* Cliente: Crear un punto de montaje
  + Elegir un directorio para el montaje.
    - Si no existe, 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
    - Output:

Filesystem Size Used Avail Use% Mounted on

…

192.168.0.8:/var/nfs/general 25G 1.8G 23G 8% /tmp/nfs

* 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.

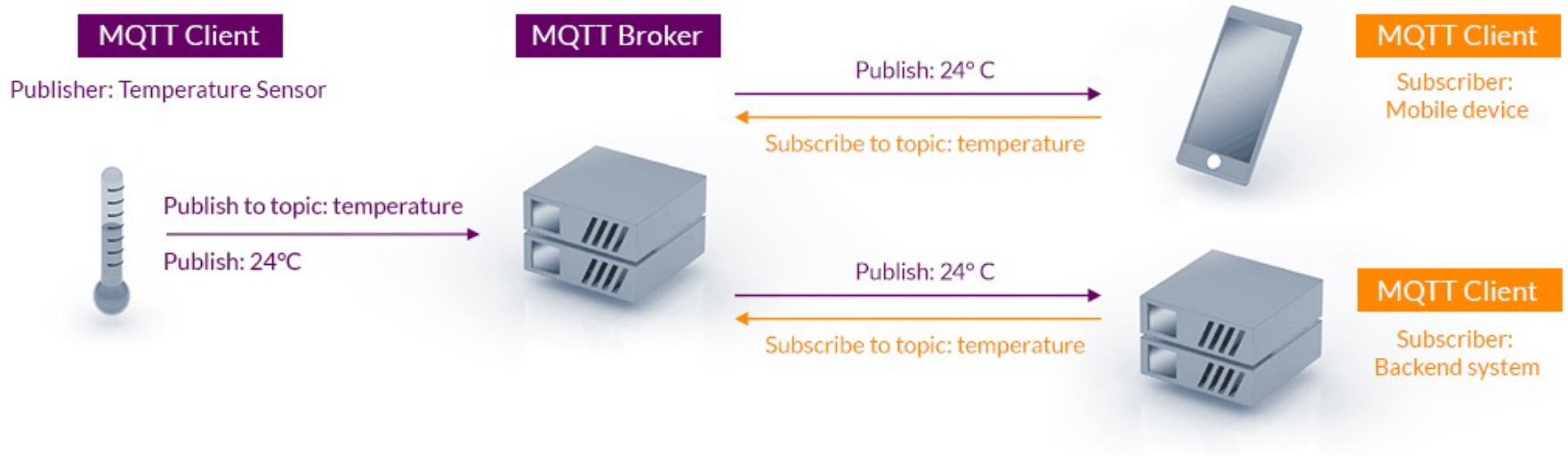
**1. ariketa:**

**2.1.3. Intercambio de mensajes en red (MQTT):**

Intercambiar mensajes es más ágil que intercambiar ficheros.

Message Queue Telemetry Transport (MQTT):

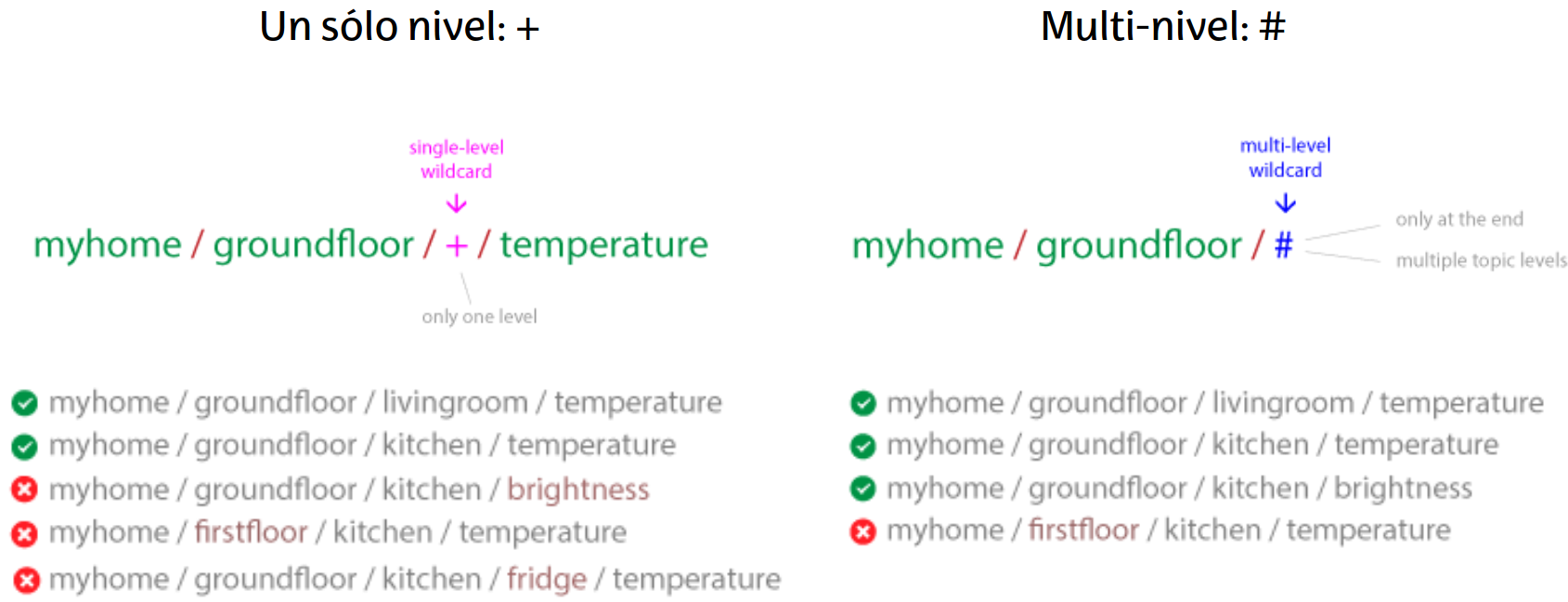
* Un protocolo orientado a paso de mensajes en formato publicar-suscribir.
  + En contraposición al cliente-servidor.
  + Separa al cliente que envía mensajes del que recibe mensajes.
* Web: <https://mqtt.org/>
* Es muy usado en entornos Internet of Things (IoT).
  + Permite gestionar cientos/miles de clientes ligeros.
  + Por ejemplo: Logística, industria, domótica.
* Arquitectura:



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

MQTT topics:

* El proceso Broker filtra los mensajes en base a topics.
  + Topic: Un String de texto que identifica mensajes de una temática concreta.
* Escritura:
  + Pueden ser multi-nivel, se separan mediante /.
  + Son sensibles a mayúsculas-minúsculas y pueden contener espacios.
  + No deben comenzar por $.
* También se pueden filtrar por contenido o tipo, pero no son las formas más usadas.
* Se pueden utilizar wildcards al dirigirse a topics:



Mosquitto:

* Framework open source que implementa MQTT.
* Web: <https://mosquitto.org/>
* Muy utilizado en entornos IoT.
  + Es ligero, configurable y extensible mediante plug-ins.
* Instalación:
  + apt update

apt install mosquitto mosquitto-clients

* Verificar que el servicio está en marcha:
  + service mosquitto status
* Por defecto, utiliza el puerto 1883.