**UT 2.1 – Repaso TCP/IP**

1. **Modelo OSI**: El modelo OSI (Open Systems Interconnection) es un estándar de referencia para la comunicación en redes. Consta de 7 capas:
   * **Capa 1 - Física**: Define el medio de transmisión y el flujo de bits.
   * **Capa 2 - Enlace**: Gestiona el direccionamiento físico mediante la dirección MAC y el acceso al medio, con capacidad de detección de errores.
   * **Capa 3 - Red**: Encargada del enrutamiento, con dispositivos como routers y el protocolo IP.
   * **Capa 4 - Transporte**: Permite el envío de datos independientes de la red física, con protocolos como TCP y UDP.
   * **Capa 5 - Sesión**: Mantiene y reanuda las sesiones de comunicación.
   * **Capa 6 - Presentación**: Asegura una representación reconocible y puede cifrar los datos.
   * **Capa 7 - Aplicación**: Interactúa directamente con las aplicaciones y ofrece acceso a servicios de red.
2. **Modelo cliente-servidor**: En este modelo de interacción, un cliente inicia una solicitud y el servidor la procesa. Esto permite centralizar los servicios, facilita el mantenimiento y mejora la seguridad de la red.
3. **Modelo TCP/IP**: Consta de 4 capas principales:
   * **Acceso a red**: Usa direcciones MAC y opera a nivel físico.
   * **Internet**: Usa IP para la identificación y enrutamiento de paquetes.
   * **Transporte**: Asegura la transmisión de datos con protocolos como TCP (fiable) y UDP (sin conexión).
   * **Aplicación**: Proporciona servicios de alto nivel como HTTP, FTP y DNS para las aplicaciones.
4. **Encapsulación en TCP/IP**: Cada capa encapsula la información de la capa superior en estructuras específicas, como tramas, paquetes o datagramas. Este proceso permite la transmisión ordenada y eficiente de datos a través de la red.
5. **Acceso a red**:
   * **Dirección MAC**: Identificación única de los dispositivos de red, en formato hexadecimal de 6 octetos.
   * **Dispositivos**: Los **switches** almacenan direcciones MAC para transmitir datos, mientras que los **hubs** los difunden a todos los dispositivos conectados.
6. **Red / IP / Internet**:
   * **Dirección IP**: Identificación lógica de los dispositivos en redes IP, en formato de 32 bits para IPv4.
   * **Subnetting y Supernetting**: Estas técnicas permiten dividir redes grandes en subredes (Subnetting) y agrupar subredes en redes mayores (Supernetting) para optimizar el uso de direcciones IP y simplificar el enrutamiento.
7. **Routing**:
   * Los routers determinan la ruta óptima para enviar paquetes a través de diferentes redes.
   * Existen tablas de encaminamiento para gestionar el tráfico de red, y estas pueden ser configuradas de manera estática (archivos de configuración) o dinámica (protocolo RIP o OSPF).
8. **Protocolos TCP/UDP**:
   * **TCP** asegura la entrega de los paquetes mediante control de flujo y corrección de errores.
   * **UDP** ofrece un envío de paquetes sin conexión, ideal para aplicaciones que requieren rapidez y tolerancia a pérdida de datos.
9. **Protocolos de aplicación**: Incluyen servicios como:
   * **DHCP** (asignación de direcciones IP),
   * **FTP** (transferencia de archivos),
   * **HTTP/HTTPS** (navegación web),
   * **DNS** (resolución de nombres de dominio),
   * **SMTP, POP3, IMAP** (correo electrónico).

**UT 2.1 – Servidores, Escalabilidad y Arquitecturas**

1. **Arquitectura cliente-servidor**:
   * **Cliente**: Dispositivo o aplicación que solicita recursos, como páginas web o datos.
   * **Servidor**: Gestiona, almacena o genera recursos, y responde a las solicitudes de los clientes.
2. **Tipos de servidores**:
   * **Servidores web**: Proporcionan contenido estático como HTML y CSS. Ejemplos: Apache, Nginx.
   * **Servidores de aplicaciones**: Gestionan la lógica de negocio y pueden interactuar con bases de datos. Ejemplos: Tomcat, GlassFish.
   * **Servidores de bases de datos**: Almacenan y gestionan datos estructurados. Otros servidores comunes son los de correo, archivos y caché.
3. **Servidores on-premise y en la nube**:
   * **On-premise**: Equipos físicos en las instalaciones de la empresa, lo que implica un mayor control, pero también más costes iniciales y necesidad de personal especializado.
   * **Cloud**: Virtualizados en infraestructuras de proveedores de servicios, con flexibilidad y pago por uso. Ejemplos: AWS, Azure.
4. **Escalabilidad**:
   * **Escalabilidad vertical (scale-up)**: Aumenta los recursos de un único servidor.
   * **Escalabilidad horizontal (scale-out)**: Añade más servidores para distribuir la carga de trabajo, lo que mejora la tolerancia a fallos y la disponibilidad.
5. **Balanceo de carga**: Distribuye las solicitudes entre varios servidores, lo que mejora el rendimiento y reduce el tiempo de respuesta. Es clave en arquitecturas escalables.
6. **Replicación**: Mantiene varias copias de una base de datos en diferentes servidores para mejorar la disponibilidad y la redundancia en caso de fallos. Puede ser:
   * **Maestro-esclavo**: Solo el maestro acepta escrituras; los esclavos gestionan las lecturas.
   * **Maestro-maestro**: Todos los nodos pueden leer y escribir, y sincronizan los datos entre ellos.
7. **CDN (Red de Distribución de Contenidos)**: Red de servidores distribuidos geográficamente que almacena contenido estático de las webs para reducir latencia y mejorar la disponibilidad. Ejemplos: Cloudflare, Akamai.
8. **Arquitecturas de servidores**:
   * **Servidor único**: Para aplicaciones pequeñas, aunque limita la escalabilidad y la tolerancia a fallos.
   * **Servidor web y de BBDD separados**: Mejora el rendimiento y seguridad.
   * **Balanceo en web**: Varias instancias para la parte web, pero un único servidor para la base de datos.
   * **Balanceo en web y BBDD**: Múltiples instancias para web y bases de datos, mejorando la tolerancia a fallos en ambas áreas.
9. **Arquitectura n-niveles**:
   * **Niveles**: División física en componentes distribuidos en varias máquinas.
   * **Capas**: División lógica del software, con funciones como presentación, lógica de negocio y acceso a datos.
10. **Microservicios**:
    * Dividen la aplicación en servicios independientes, que pueden ser escalados y gestionados por separado.
    * Cada microservicio es autónomo y se comunica mediante APIs bien definidas, lo que permite usar diferentes lenguajes de programación para cada uno.

**UT 2.1 – Apache**

1. **Apache Software Foundation**:
   * Organización que promueve software libre, con más de 350 proyectos, como Apache Tomcat, Apache Hadoop, y Apache Maven.
   * El servidor web Apache es uno de los proyectos más utilizados en Internet.
2. **Servidor web Apache**:
   * **Código abierto** y disponible para su descarga y modificación.
   * Se actualiza regularmente y es compatible con protocolos como TLS 1.3.
   * Es modular, permitiendo la ampliación de sus capacidades mediante módulos (autenticación, encriptación, soporte de lenguajes, etc.).
3. **Instalación de Apache**:
   * En sistemas como Debian o Ubuntu, se instala con sudo apt install apache2.
   * La configuración principal se encuentra en /etc/apache2/apache2.conf.
4. **Configuración de módulos**:
   * Los módulos disponibles están en /etc/apache2/mods-available, y los activos en /etc/apache2/mods-enabled.
   * Los comandos a2enmod y a2dismod permiten habilitar o deshabilitar módulos.
5. **Configuración de sitios virtuales**:
   * **Directorio /etc/apache2/sites-available**: Contiene los sitios configurados que pueden activarse.
   * **Directorio /etc/apache2/sites-enabled**: Contiene enlaces simbólicos a los sitios activos.
   * Los comandos a2ensite y a2dissite permiten activar o desactivar sitios.
6. **Herramientas de configuración adicionales**:
   * **Para módulos**: a2enmod (activar) y a2dismod (desactivar).
   * **Para sitios**: a2ensite (activar) y a2dissite (desactivar).
   * **Para configuraciones adicionales**: a2enconf y a2disconf permiten gestionar archivos de configuración específicos.