Apartado de Investigación y documentación de M01 UF2

1. ¿Qué es un protocolo? Explica las características típicas de un protocolo de red vistas en clase.

Un **protocolo** es un conjunto de reglas y estándares de comunicaciones que definen cómo se debe realizar la comunicación entre diferentes dispositivos dentro de una red. Los protocolos garantizan que los datos sean enviados y recibidos de manera correcta y eficiente, permitiendo la comunicación entre equipos y servicios.

Características comunes entre los protocolos de red:

- **Sintaxis**: Define el formato de los mensajes, cómo se deben estructurar los datos.
- **Semántica**: Especifica el significado de los mensajes y cómo se deben interpretar.
- **Sincronización**: Asegura que las comunicaciones se lleven a cabo de manera sincronizada, gestionando los tiempos de envío y recepción.
- **Control de errores**: Los protocolos suelen incluir métodos para detectar y corregir errores en la transmisión de datos.
- **Seguridad**: Algunos protocolos incorporan mecanismos de cifrado y autenticación para proteger la información transmitida.

Fuente: https://autmix.com/blog/que-es-protocolo-red

https://openwebinars.net/blog/protocolo-de-red-que-es-tipos-y-caracteristicas/

2. ¿Qué protocolos de red conoces? Comenta al menos 5. Además, investiga al menos 3 protocolos de red que creas interesantes. De todos, debes explicar para qué sirven, qué características tienen, los puertos con los que trabajan, etc.

Protocolos Conocidos:

1. **HTTP (HyperText Transfer Protocol):**

    - **Propósito**: Usado para la transferencia de páginas web.
    - **Puerto**: 80
    - **Características**: Es un protocolo sin estado (stateless), lo que significa que no guarda información entre peticiones.

      Fuente: https://loscreativos.co/marketing-digital/que-es-http-definicion-caracteristicas/

2. **HTTPS (HyperText Transfer Protocol Secure):**

    - **Propósito**: Versión segura de HTTP, utiliza SSL/TLS para cifrar la comunicación.
    - **Puerto**: 443
    - **Características**: HTTPS utiliza el cifrado SSL/TLS, lo que asegura que los datos intercambiados entre el servidor y el usuario estén protegidos y no puedan ser leídos por terceros. Esto es especialmente importante en transacciones sensibles como pagos en línea o el envío de información personal.

      Fuente: https://www.tecnoinver.cl/definicion-protocolo-https/

3. **FTP (File Transfer Protocol):**

    - **Propósito**: Usado para transferir archivos entre sistemas en una red.
    - **Puerto**: 21
    - **Características**: El servicio FTP es un servicio utilizado para el envío y obtención de archivos entre dos equipos remotos. Los casos más usuales son transferencias entre el equipo local de un cliente y el servidor del proveedor.

      Fuente:https://blog.infranetworking.com/servidor-ftp/#Para_que_sirve

4. **SMTP (Simple Mail Transfer Protocol):**

    - **Propósito**: Usado para el envío de correos electrónicos.
    - **Puerto**: 25
    - **Características**: Define cómo se envían los correos electrónicos entre servidores de correo. Seguridad y autenticación, fiabilidad en la entrega de correos electrónicos y eficiencia en la transferencia de mensajes

      Fuente: https://www.arsys.es/blog/que-es-smtp-y-para-que-sirve#tree-3

5. **DNS (Domain Name System):**

    - **Propósito**: Traducir nombres de dominio legibles por humanos (como www.ejemplo.com) a direcciones IP.
    - **Puerto**: 53
    - **Características**: Se usa en la navegación web, ya que convierte las URL en direcciones IP que las computadoras entienden y viceversa.

      Fuente: https://www.ibm.com/es-es/topics/dns-protocol

Protocolos investigados para el ejercicio:

1. **SSH (Secure Shell):**

- **Propósito**: Protocolo seguro para acceder a dispositivos de forma remota y ejecutar comandos.
- **Puerto**: 22
- **Características**: SSH o Secure Shell, es un protocolo de administración remota que le permite a los usuarios controlar y modificar sus servidores remotos a través de Internet a través de un mecanismo de autenticación..

  Fuente:https://www.hostinger.es/tutoriales/que-es-ssh

2. **SNMP (Simple Network Management Protocol):**

   - **Propósito**: Usado para gestionar y monitorear dispositivos en una red.
   - **Puerto**: 161
   - **Características**: Permite supervisar redes, recolectar información y realizar configuraciones de dispositivos en tiempo real.

     Fuente: https://www.whatsupgold.com/es/snmp

3. **DHCP (Dynamic Host Configuration Protocol):**

   - **Propósito**: Asigna direcciones IP de forma dinámica a dispositivos en una red.
   - **Puerto**: 67 (servidor) / 68 (cliente)
   - **Características**: SNMP se puede definir como un protocolo de nivel de aplicación diseñado para supervisar la infraestructura de red y proporciona a los administradores visibilidad centrada en el dispositivo.

     Fuente:https://www.whatsupgold.com/es/snmp

3. ¿Qué servidor HTTP hemos visto en clase? ¿Cómo lo instalamos? ¿Dónde se encuentran sus archivos?

El servidor HTTP que hemos visto es **Apache HTTP Server** (comúnmente conocido como **Apache**).

Cómo instalar Apache:

En Ubuntu/Debian:

```
 sudo apt update
sudo apt install apache2
```

En CentOS/Fedora:

```
 sudo yum install httpd
```

Ubicación de los archivos:

- **Archivos de configuración:** /etc/apache2/ (en Debian/Ubuntu) o /etc/httpd/ (en CentOS/Fedora).
- **Raíz de los documentos:** /var/www/html/ (donde se almacenan las páginas web servidas).

Fuente**:** Apache HTTP Server Official Documentation

4. ¿Por qué, como usuarios normales, no podemos escuchar el puerto correspondiente a HTTP ni otros similares?

En sistemas basados en Unix, los puertos **por debajo de 1024** (como el puerto 80 para HTTP y 443 para HTTPS) están **reservados** para el uso de procesos del sistema con privilegios privados. Los usuarios normales no tienen permisos para escuchar en estos puertos para evitar que aplicaciones no autorizadas interfieran en el sistema. Se requiere privilegios de **root** o **administrador** para que un servicio pueda escuchar en esos puertos.

Fuente**:**
https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml

https://www.reddit.com/r/selfhosted/comments/rzot4s/which_ports_can_i_use_only_below_1024/?tl=es-es&rdt=41067

5. ¿Quién inventó el protocolo HTTP? ¿Para qué sirve? ¿Por qué es considerado uno de los inventos más importantes de la historia?

El protocolo HTTP fue propuesto por Tim Berners-Lee en 1989. Tim Berners-Lee también es el creador de la WWW (World Wide Web).

HTTP se usa para transmitir documentos, imágenes, videos y otros recursos en la web, facilitando la comunicación entre navegadores y servidores.

HTTP es fundamental para el desarrollo de la web moderna. Su facilidad de implementación y uso ha permitido la interconexión global de dispositivos, fomentando el acceso universal a la información.

Fuente**:** https://www.ionos.es/digitalguide/hosting/cuestiones-tecnicas/protocolo-http/

https://es.wikipedia.org/wiki/Tim_Berners-Lee

6. ¿Cómo se diseñan los protocolos, cifrados o sin cifrar? ¿Por qué? ¿Cómo se cifran?

Los protocolos inicialmente son diseñados sin cifrar para agilizar su uso, después si es necesario se les aplica otro protocolo para cifrarlos.

¿Por qué ciframos?

Los protocolos se cifran para mejorar la **privacidad** y garantizar que los datos no sean interceptados por terceros malintencionados. El cifrado también asegura que los datos no sean alterados en su trayecto a nivel de la integridad del propio mensaje.

¿Cómo se cifran los protocolos?

Se usan algoritmos de cifrado como **AES** (Advanced Encryption Standard) o **RSA** para proteger los datos. Por ejemplo, HTTPS utiliza **SSL/TLS** para establecer un canal seguro entre cliente y servidor.

> Fuente**:** https://www.cloudflare.com/en-gb/learning/ssl/what-is-ssl/

7. ¿Por qué, como futuras personas que van a tener un computador (del tipo que sea) es importante conocer cómo funciona la máquina y los sistemas de comunicación que se van a utilizar?

> Lo principal es entender cómo funcionan los sistemas permite protegerlos contra amenazas, como malware o ataques de red. Además de estar protegido el conocimiento te puede ayudar a mejorar el rendimiento del hardware y los sistemas operativos según tus necesidades. Trabajando con ordenadores te encontrarás con multitud de de problemas y conocer la arquitectura y los protocolos permite diagnosticar y resolver estos de manera eficiente.
>
> Una persona que entienda cómo funciona su sistema operativo podrá configurar mejor su red doméstica, asegurándose de que sea segura y eficiente.

8. ¿Por qué todo esto que vemos es abierto y libre y aún así lo utilizan las macrocorporaciones? ¿Por qué no hay patentes de software en Europa ni en casi todo el planeta?

El software es abierto y libre ya que el concepto de patentarlo es muy abstracto y no se ajusta a la realidad del código, es decir: Tu puedes crear una fórmula mediante calcular la velocidad de la tierra alrededor del Sol pero eso no te da derecho a atribuirlo como tuyo, si que está bien reconocer los méritos pero no es algo patentable. En principio. Las macro corporaciones siguen utilizando software libre ya que a ellos también les costaría dinero obtener las licencias y es muy útil.

> Fuente**:** https://fsfe.org/

9. Explica los movimientos vistos en VIM en clase e investiga algún movimiento más que te parezca interesante. ¿Cómo buscamos textos en VIM? ¿Cómo reemplazamos palabras en VIM?

Movimientos vistos:

- **h, j, k, l**: Moverse hacia la izquierda, abajo, arriba y derecha respectivamente.
- **gg**: Ir al principio del archivo.
- **G**: Ir al final del archivo.
- **:wq** Sirve para guardar y salir del proyecto.
- **:q** Sirve para salir sin guardar.

Para buscar, usamos ：

/ seguido del texto:
/palabra_a_buscar

Para reemplazar una palabra en todo el archivo:
:%s/palabra_antigua/palabra_nueva/g

He encontrado este movimiento para tratar con codigos grandes:

> **ctrl+d y ctrl+u**: Permiten moverse hacia abajo y arriba de manera rápida por bloques de texto.
>
> Fuente**:** https://www.vim.org/docs.php

Comentando los Scripts

Script del Servidor

Introducción

El script del servidor es el encargado de escuchar las conexiones que llegan desde el cliente. Una vez que el cliente se conecta, el servidor sigue una serie de pasos para recibir los datos, validarlos y responder correctamente. A lo largo del proceso, el servidor comprueba cosas como el nombre del archivo, su contenido y los hashes **MD5** enviados por el cliente. Si algo sale mal, el servidor envía un mensaje de error distinto para cada uno.

Pasos del Servidor

1. Iniciar el servidor y escuchar conexiones

El servidor se pone en modo de "espera"a escuchar en un puerto específico (en este caso, el puerto 2022). Este paso asegura que esté listo para recibir una conexión del cliente.

2. Recibir la cabecera del cliente

Cuando el cliente se conecta, envía una cabecera ("**DMAM**") para indicar que desea establecer comunicación. El servidor comprueba que esta cabecera sea correcta:

- Si la cabecera es incorrecta, responde con **KO_HEADER** y termina el programa.
- Si es correcta, responde con **OK_HEADER** y espera el siguiente paso.

3. Recibir el nombre del archivo y su MD5

El cliente envía el nombre del archivo y su hash **MD5**. Entonces el Servidor realiza una serie de pasos y comprobaciones:

- Separa el nombre del archivo y el **MD5** recibido para saber como llamar al archivo.
- Calcula el **MD5** del nombre del archivo recibido.
- Compara ambos hashes.

Si los MD5 no coinciden, responde con **KO_FILE_NAME_MD5** y termina. Si todo está bien, responde con **OK_FILE_NAME**.

4. Recibir y almacenar los datos del archivo

El cliente envía el contenido del archivo. Entonces el servidor:

- Comprueba que los datos no estén vacíos.
- Almacena el contenido recibido en un archivo local en el directorio del servidor.

Si algo falla en este paso, envía **KO_DATA** y termina. Si todo es correcto, responde con **OK_DATA**.

5. Recibir y comprobar el MD5 del contenido

El cliente también envía un **MD5** del contenido del archivo. El servidor:

- Calcula el **MD5** del archivo que ha recibido.
- Compara el **MD5** calculado con el que envió el cliente.

Si los hashes no coinciden, responde con **KO_FILE_MD5**. Si coinciden, responde con **OK_FILE_MD5** y termina con éxito.

Resumen

El servidor es como un "portero" que controla todo lo que le llega del cliente. Su objetivo es asegurarse de que los datos recibidos sean válidos antes de almacenarlos y responder al cliente. Si detecta algún problema, no sigue con el proceso y envía un mensaje de error para que el cliente lo sepa.

Script del Cliente

Introducción

El script del cliente es el encargado de conectarse al servidor, enviar los datos y esperar las respuestas del servidor en cada paso. El script está diseñado para que los datos se puedan transmitir de manera correcta sin que se corrompan y hay cierta parte de código que sirve para asegurarse de que aquello que se manda es lo deseado.

Pasos del Cliente

1. Iniciar conexión con el servidor

El cliente comienza enviando una cabecera llamada "**DMAM**" al servidor para indicar que desea establecer comunicación. Luego, espera una respuesta del servidor:
- Si el servidor responde con **OK_HEADER**, el cliente pasa al siguiente paso.
- Si el servidor responde con **KO_HEADER**, el cliente termina el programa, ya que significa que algo salió mal.

2. Enviar el nombre del archivo y su MD5

El cliente toma el nombre del archivo que desea transferir y calcula su hash **MD5** (una especie de huella digital del nombre). Luego, envía ambos datos al servidor en el siguiente formato:

FILE_NAME nombre_del_archivo **MD5SUM**

Después, espera una respuesta del servidor:
OK_FILE_NAME: Significa que el servidor aceptó el nombre del archivo.
KO_FILE_NAME_MD5: Significa que el hash MD5 no coincidió y el cliente debe terminar el programa.

3. Enviar el contenido del archivo

El cliente lee el contenido del archivo y lo envía al servidor. Luego, espera la respuesta del servidor:
OK_DATA: Indica que el servidor recibió y almacenó correctamente los datos.
KO_DATA: Significa que hubo un problema al recibir los datos y el cliente debe terminar el proceso.

4. Enviar el MD5 del contenido del archivo

El cliente calcula el hash **MD5** del contenido del archivo que envió previamente. Luego, envía el **MD5** al servidor con el siguiente formato:

FILE_MD5 MD5SUM

Después de esto, espera la última respuesta del servidor:
OK_FILE_MD5: Indica que el contenido es válido y coincide con lo que el cliente envió.
KO_FILE_MD5: Significa que el contenido no coincidió y el cliente debe terminar.

Resumen:

El cliente es el que envia las cosas y elo que inicia la “transacción” se encarga de enviar la información al servidor siguiendo un protocolo. En cada paso, verifica que el servidor acepte los datos antes de continuar. Si en algún momento el servidor encuentra un problema, el cliente detiene la comunicación para evitar errores mayores.

Esquema del protocolo en general

Esquema del Protocolo

1. Cuando se activa un servidor se le conoce como 'levantar'.
2. Una vez activo, el cliente envía un header.
3. El servidor responde con **OK_HEADER** o **KO_HEADER** dependiendo de si el header es correcto.
4. El cliente envía el mensaje al servidor.
5. El servidor recibe los datos y comprueba el nombre del archivo (**FILENAME**).
6. Si es correcto, el servidor almacena la información y envía una respuesta.
8. Enviar contenido al archivo.
9. Recibir datos.
10. Verificar respuesta.
11. Recibir confirmación (**OK/KO**).
12. Enviar datos adicionales.
13. Recibir y verificar **MD5**.
14. Confirmar **MD5** y finalizar.