- CAPA DE APLICACIÓN
  - DNS
    - Cómo Funciona el DNS
    - Jerarquía del DNS:
    - Registro de recursos
    - Resumen
      - Resumen del Flujo de Proceso para github.com:
  - SMTP
    - Funcionamiento de SMTP
    - Limitaciones de SMTP
    - Extensiones de SMTP (ESMTP - RFC 5321)
    - Envío y Transferencia de Mensajes
    - Resumen
      - 1. Composición del Correo
      - 2. Envío del Correo
      - 3. Transferencia del Correo al Servidor SMTP
      - 4. Enrutamiento del Correo
      - 5. Recepción del Correo
      - 6. Acceso del Destinatario
  - Resumen del Flujo
  - World Wide Web (WWW)
    - Sistema MIME
    - Servidores Webs
      - Funcionamiento Básico del Servidor Web
      - Optimización del Servidor
      - Procesamiento de Solicitudes
      - Consideraciones Adicionales
    - Cookies
      - Usos
    - HTTP
      - Encabezados HTTP
        - Encabezados de solicitud (del cliente al servidor)
        - Encabezados de respuesta (del servidor al cliente)
        - Encabezados compartidos (usados en ambas direcciones)
      - Almacenamiento en Cache
  - Audio y video de flujo continuo
    - Audio digital

CAPA DE APLICACIÓN

> La capa de aplicación en las redes de computadoras es la capa más cercana al usuario y es responsable de proporcionar servicios de red a las aplicaciones de software. Esta capa actúa como intermediario entre las aplicaciones y los protocolos de red, permitiendo que las aplicaciones utilicen los servicios de red sin tener que preocupar por los detalles de implementación de los protocolos de bajo nivel.

🌍 DNS

El Sistema de Nombres de Dominio (DNS) es fundamental para la navegación en Internet, ya que permite a los usuarios acceder a recursos en la red mediante nombres legibles en lugar de direcciones IP numéricas. Aunque es posible referirse a páginas web y otros recursos usando direcciones IP, recordar estos números es complicado

para las personas. Por ejemplo, si un servidor web cambia su dirección IP, todos tendrían que ser informados sobre la nueva dirección. Corre por el puerto 53(UDP) de protocolo UDP.

> **Cómo Funciona el DNS**
>
> 1. **Proceso de Resolución** : Para convertir un nombre de dominio en una dirección IP, una aplicación utiliza un procedimiento de biblioteca llamado **resolvedor** .
> 2. **Consulta** : El resolvedor envía una consulta al servidor DNS local con el nombre del dominio.
> 3. **Respuesta** : El servidor DNS busca el nombre y devuelve la dirección IP al resolvedor.
> 4. **Conexión** : Una vez que se obtiene la dirección IP, la aplicación puede establecer una conexión TCP o enviar paquetes UDP a ese host.

Los mensajes de solicitud y respuesta entre el resolvedor y el servidor DNS se transmiten utilizando paquetes UDP, lo que permite una comunicación eficiente y rápida.

Jerarquía del DNS:

- **Nivel Superior (Raíz)** : En Internet, la jerarquía comienza en la raíz (sin nombre) y se divide en **dominios de nivel superior (TLDs)** , administrados por ICANN (Internet Corporation for Assigned Names and Numbers). Cada dominio de nivel superior puede contener subdominios.
- **Dominios Genéricos** : Los TLDs genéricos incluyen dominios como `.com`, `.org`, `.edu`, etc. Estos dominios agrupan diferentes tipos de organizaciones y tienen ciertos criterios para su uso (por ejemplo, `.edu` para instituciones educativas).
- **Dominios de País** : Se incluyen en esta categoría los dominios asignados a países según la norma ISO 3166, como `.us` para Estados Unidos, `.jp` para Japón, etc. Además, en 2010 se introdujeron dominios internacionalizados, permitiendo el uso de alfabetos distintos al latín, como árabe o chino.
- **Subdivisión de Dominios** : Los dominios se subdividen jerárquicamente a medida que se desciende en el árbol. Por ejemplo, un dominio como `eng.cisco.com` representa un subdominio dentro de `cisco.com`, que a su vez es parte de `com`.
- **Dominios Absolutos y Relativos** : Un nombre de dominio absoluto siempre termina con un punto (`eng.cisco.com.`), mientras que un dominio relativo no lo

hace y se interpreta en un contexto determinado (por ejemplo, `eng.cisco.com` en un contexto de búsqueda dentro del dominio).

Registro de recursos

Un **registro de recursos** es una **5-tupla** que contiene información clave para resolver una consulta DNS. El formato general es:

```
Nombre_dominio Tiempo_de_vida Clase Tipo Valor
```

- **Nombre_dominio** : El dominio asociado al registro, siendo la clave primaria de búsqueda.
- **Tiempo_de_vida (TTL)** : Tiempo durante el cual el registro se mantiene válido en caché. Este valor se expresa en segundos.
- **Clase** : Casi siempre es **IN** (para Internet), pero puede variar en otros casos.
- **Tipo** : Especifica el tipo de registro DNS (A, MX, NS, etc.).
- **Valor** : Depende del tipo de registro, como una dirección IP o un nombre de dominio.

```
example.com. 86400 IN A 93.184.216.34
```

Resumen

Cuando se escribe una URL como `github.com` en tu navegador , el sistema DNS se encarga de traducir este nombre de dominio a una dirección IP que el navegador necesita para establecer una conexión con el servidor de GitHub. Lo primero que hace el navegador es verificar si ya tiene la dirección IP correspondiente a `github.com` almacenada en su **caché local** . Si no está en la caché del navegador, entonces necesita realizar una consulta DNS para obtenerla.

Si la dirección no está en la caché del navegador, la solicitud se envía a un servidor **DNS recursivo** . Este servidor generalmente es proporcionado por tu ISP (Proveedor de Servicios de Internet) o puede ser un servidor DNS público como los de Google (8.8.8.8) o Cloudflare (1.1.1.1).

El servidor recursivo intenta resolver la consulta de la forma más eficiente posible. Si tiene la dirección IP de `github.com` en su **caché** , la devuelve al navegador

inmediatamente.

Si el servidor DNS local no tiene la dirección IP en su caché, comienza a hacer consultas a otros servidores. Primero, envía una consulta a un **servidor raíz** . Los servidores raíz no conocen la dirección IP de `github.com` directamente, pero saben qué servidores manejarán los dominios de nivel superior, como `.com`.

Por ejemplo, el servidor raíz diría: "No sé dónde está `github.com`, pero sé que los servidores de nombres del TLD `.com` pueden ayudar".

El servidor raíz responde con la dirección de los **servidores de nombres** responsables de los dominios `.com`. El servidor DNS recursivo ahora consulta uno de esos servidores.

El servidor TLD `.com` tampoco tiene la IP directa de `github.com`, pero sabe que el dominio `github.com` tiene servidores específicos responsables de él, es decir, los **servidores autoritativos** de GitHub.

El servidor TLD `.com` responde con la dirección de los **servidores DNS autoritativos de GitHub** . Estos servidores contienen la información precisa para resolver `github.com`.

El servidor DNS recursivo ahora realiza una consulta a los servidores autoritativos de GitHub para obtener la dirección IP de `github.com`.

Los servidores autoritativos de GitHub responden con la dirección IP asociada a `github.com`.

La dirección IP de `github.com` se envía de vuelta al servidor DNS recursivo, que la guarda en su caché para futuras consultas. Luego, el servidor DNS recursivo envía esta dirección IP al navegador.

Una vez que el navegador recibe la dirección IP, puede usar ese valor para realizar una conexión directa con el servidor de GitHub a través de **TCP/IP** . El navegador ahora puede enviar solicitudes HTTP(S) al servidor de GitHub, y recibir la página web que pediste.

Los servidores DNS suelen guardar la dirección IP en su caché durante un tiempo determinado. Este tiempo se controla mediante el campo **TTL** (Time to Live), que indica cuánto tiempo una respuesta DNS debe considerarse válida antes de tener que

consultar nuevamente. El TTL puede variar, pero generalmente puede ser desde unos pocos minutos hasta varias horas.

Resumen del Flujo de Proceso para `github.com`:

1. El navegador consulta si la IP está en su caché.
2. Si no está, envía la consulta al servidor DNS recursivo. (de tu proveedor de red o una publica como la de google (8.8.8.8))
3. El servidor DNS recursivo consulta los servidores raíz.
4. El servidor raíz remite la consulta a los servidores del TLD `.com`.
5. Los servidores del TLD `.com` remiten la consulta a los servidores autoritativos de GitHub.
6. Los servidores autoritativos de GitHub devuelven la dirección IP de `github.com`.
7. El navegador se conecta a GitHub usando la dirección IP y carga el contenido.

📧 SMTP

El **protocolo SMTP (Simple Mail Transfer Protocol)** , el cual opera sobre **TCP en el puerto 25** . Este protocolo es responsable de enviar mensajes entre agentes de usuario y servidores de correo, así como de transferirlos entre servidores de correo. En el RFC 4409.

Funcionamiento de SMTP

1. **Conexión:** La computadora emisora establece una conexión TCP con la receptora, que escucha en el puerto 25.
2. **Intercambio de mensajes:** Se identifican los participantes (comando `HELO` o `EHLO` en ESMTP) y se envían los detalles del correo.
3. **Transferencia del mensaje:** Se envía el mensaje utilizando comandos como `MAIL FROM`, `RCPT TO`, `DATA`, y se confirma la recepción.
4. **Cierre de conexión:** Una vez entregado el mensaje, la conexión se cierra

Limitaciones de SMTP

- **Falta de autenticación:** Permite falsificar remitentes, lo que facilita el envío de spam.
- **Soporte solo para ASCII:** Requiere codificación base64 para archivos adjuntos, lo que es ineficiente en uso de ancho de banda. (transfiere mensajes ASCII , no

datos binarios)
- **Falta de cifrado:** No protege la privacidad del contenido del correo.

Extensiones de SMTP (ESMTP - RFC 5321)

Para mejorar sus funcionalidades, se introdujeron extensiones como:

- **AUTH:** Autenticación de usuarios.
- **BINARYMIME:** Soporte para mensajes binarios.
- **STARTTLS:** Cifrado de la comunicación con TLS.
- **SIZE:** Verificación del tamaño del mensaje antes de enviarlo.

Envío y Transferencia de Mensajes

- **Envío de correos:** Ahora se realiza a través del puerto **587** , usando autenticación (`AUTH`).
- **Transferencia entre servidores:** Se utiliza DNS para determinar el servidor de destino y entregar el correo al agente de transferencia de mensajes receptor.

SMTP sigue siendo el protocolo estándar para la transferencia de correos, pero con mejoras en seguridad y autenticación mediante extensiones como **ESMTP y STARTTLS**

Resumen

El protocolo SMTP (Simple Mail Transfer Protocol) se utiliza para enviar y transferir correos electrónicos entre servidores y desde un cliente de correo a un servidor. Aquí tienes un resumen de cómo funciona el proceso completo de movimiento del correo electrónico utilizando SMTP:

1. Composición del Correo

- Un usuario redacta un mensaje de correo electrónico en un cliente de correo (como Outlook, Thunderbird o una aplicación web).
- El mensaje incluye campos como el destinatario, el asunto y el cuerpo del mensaje.

2. Envío del Correo

- Cuando el usuario hace clic en "Enviar", el cliente de correo se conecta al servidor SMTP configurado.
- El cliente inicia una sesión SMTP y envía comandos al servidor para transferir el mensaje. Estos comandos incluyen:
  - `HELO` o `EHLO`: Identifica al cliente SMTP ante el servidor.
  - `MAIL FROM`: Indica la dirección del remitente.
  - `RCPT TO`: Indica la dirección del destinatario.
  - `DATA`: Comienza el envío del cuerpo del mensaje.

3. Transferencia del Correo al Servidor SMTP

- El servidor SMTP recibe el mensaje y verifica la dirección del remitente y del destinatario.
- Si el destinatario está en el mismo dominio que el remitente, el servidor SMTP lo entrega directamente al servidor de correo del destinatario.
- Si el destinatario está en otro dominio, el servidor SMTP busca el servidor de correo del destinatario utilizando el DNS (Domain Name System) para resolver el nombre de dominio del destinatario a una dirección IP.

4. Enrutamiento del Correo

- El servidor SMTP del remitente se conecta al servidor SMTP del destinatario utilizando el puerto 25 (o 587 para conexiones seguras).
- El proceso de enrutamiento puede implicar varios servidores SMTP intermediarios, dependiendo de la red y la ruta del correo.

5. Recepción del Correo

- Una vez que el servidor SMTP del destinatario recibe el mensaje, lo almacena en el buzón de correo del destinatario.
- Aquí es donde los protocolos IMAP o POP3 entran en juego para que el destinatario pueda acceder a su correo.

6. Acceso del Destinatario

- El destinatario utiliza un cliente de correo (como un cliente de escritorio o una aplicación web) para conectarse a su servidor de correo.
- Dependiendo del protocolo utilizado (IMAP o POP3), el cliente puede descargar el mensaje o simplemente mostrarlo desde el servidor.

Resumen del Flujo

1. El usuario compone el mensaje en un cliente de correo.
2. El cliente se conecta al servidor SMTP y envía el mensaje.
3. El servidor SMTP verifica y enrutará el mensaje hacia el servidor del destinatario.
4. El servidor del destinatario almacena el mensaje en el buzón.
5. El destinatario accede a su correo mediante IMAP o POP3.

Este flujo completo asegura que los correos electrónicos se transmitan de manera efectiva y segura a través de la red

World Wide Web (WWW)

La **World Wide Web (WWW)** es un sistema de acceso a información distribuida en millones de computadoras a través de Internet.La WWW es una plataforma que permite a los usuarios acceder a información a través de **navegadores web** , utilizando el protocolo **HTTP** y el lenguaje **HTML** . Su éxito se debe a su facilidad de uso y a la posibilidad de vincular documentos mediante **hipervínculos** .

La arquitectura de la web se basa en un modelo client-servidor donde los usuarios acceden a paginas web a traves de navegadores. Estas paginas pueden contener hipervinculos que permiten la navegacion entre distintos documentos en cualquer parte del mundo. El proceso funciona mediante **HTTP (HyperText Transfer Protocol)** , un protocolo de solicitud-respuesta que permite al navegador obtener contenido desde servidores web.

Sistema MIME

MIME (Multipurpose Internet Mail Extensions) es un estándar que indica el tipo de contenido que se está transmitiendo en la web. Permite a los navegadores saber cómo interpretar y mostrar diferentes tipos de archivos (por ejemplo, HTML, PDF, JPEG, MP3).

Servidores Webs

Funcionamiento Básico del Servidor Web

1. **Conexión Inicial** :

- Cuando un usuario ingresa un URL o hace clic en un enlace, el navegador envía una solicitud al servidor correspondiente a la dirección IP obtenida mediante el DNS, estableciendo una conexión TCP en el puerto 80.

1. **Proceso de Solicitud** :

- El servidor recibe la solicitud y sigue estos pasos básicos:
    1. Acepta la conexión TCP del cliente.
    2. Obtiene la ruta del archivo solicitado.
    3. Accede al archivo en el disco o ejecuta un programa si el contenido es dinámico.
    4. Envía el contenido al cliente.
    5. Libera la conexión TCP.

Optimización del Servidor

- **Caché** :
- Para mejorar la eficiencia, los servidores modernos mantienen una caché en memoria con los archivos más solicitados, lo que reduce el acceso al disco y acelera la respuesta.
- **Multihilos** :
- En lugar de procesar una solicitud a la vez, los servidores utilizan un diseño multihilos, donde múltiples hilos pueden manejar solicitudes simultáneamente. Esto permite un procesamiento más rápido, ya que otros hilos pueden seguir trabajando mientras un hilo espera una operación de disco.

Procesamiento de Solicitudes

Cada solicitud puede implicar un proceso más complejo que simplemente devolver un archivo. Los pasos adicionales incluyen:

1. **Resolución de Nombre** :

- El servidor puede necesitar traducir nombres de archivo o directorios a nombres de archivos reales, como convertir un URL vacío en `index.html`.

1. **Control de Acceso** :

- Verifica si el cliente tiene permiso para acceder a la página solicitada, utilizando métodos como autenticación o restricciones geográficas.

1. **Verificación de Caché** :

- Determina si el archivo solicitado está en la caché para evitar accesos innecesarios al disco.

1. **Obtención del Archivo** :

- Si no está en la caché, el servidor debe cargar el archivo desde el disco o ejecutar un programa que genere contenido dinámico.

1. **Determinación de Respuesta** :

- Establece detalles adicionales de la respuesta, como el tipo MIME, que indica el formato del contenido.

1. **Envío de Respuesta** :

- Envía el contenido de la página al cliente a través de la red.

1. **Registro de Actividades** :

- Realiza una entrada en el registro para fines administrativos y estadísticas sobre el uso del sitio.

Consideraciones Adicionales

- La ejecución de programas y el manejo de archivos pueden requerir verificaciones adicionales, como si el contenido ha cambiado o si los resultados son dinámicos.
- La reutilización de conexiones TCP para múltiples solicitudes mejora el rendimiento, pero requiere lógica adicional para gestionar las respuestas correctamente.

Cookies

Las cookies son pequeños archivos de texto (máximo 4 KB) que los servidores web envían a los navegadores para almacenar información sobre los usuarios. Se utilizan para recordar información entre sesiones de navegación, ya que el protocolo HTTP es **sin estado** (no recuerda interacciones previas).

> Una cookie contiene:
>
> - **Dominio** : El sitio web que la creó.

> - **Ruta** : Qué partes del sitio pueden usarla.
> - **Contenido** : Datos guardados (por ejemplo, un identificador de usuario o carrito de compras).
> - **Expiración** : Cuándo deja de ser válida.
> - **Seguro** : Si solo puede enviarse por conexiones cifradas (SSL/TLS)

Usos

- **Autenticación y sesiones de usuario:** Permiten que los usuarios permanezcan autenticados sin tener que enviar credenciales en cada solicitud.
- **Carritos de compra en e-commerce**: El sitio guarda el producto en una cookie.Si sales del sitio y vuelves más tarde, el producto sigue en el carrito. Sin cookies, cada vez que cambies de página, el carrito se vaciaría.
- **Preferencias y personalización**: Modos oscuro de los sitios. Cuando vuelves al sitio, sigue en modo oscuro. Permiten que la experiencia del usuario sea más fluida y personalizada.
- **Rastreo y analítica web:** Si visitas varias páginas, la cookie permite saber que eres la misma persona. El sitio analiza qué partes son más visitadas y por cuánto tiempo. Las empresas pueden optimizar sus sitios web basándose en el comportamiento de los usuarios.
- **Publicidad y remarketing**

🌐 HTTP

HTTP es el protocolo que permite la comunicación entre clientes (como navegadores web) y servidores. Funciona sobre TCP y sigue un modelo de solicitud-respuesta. Un cliente envía una solicitud HTTP y el servidor responde con el contenido solicitado.

Inicialmente, con **HTTP 1.0** , cada solicitud requería abrir y cerrar una conexión TCP, lo que resultaba ineficiente cuando una página web tenía múltiples elementos, como imágenes y hojas de estilo.

Con **HTTP 1.1** , se introdujeron **conexiones persistentes** , lo que permite reutilizar la misma conexión TCP para múltiples solicitudes, reduciendo el tiempo de espera y la sobrecarga en la red. Además, se introdujo la **canalización de solicitudes** , lo que permite enviar varias solicitudes seguidas sin esperar la respuesta de la anterior, mejorando aún más la velocidad de carga

Encabezados HTTP

Los encabezados son líneas adicionales en las solicitudes y respuestas HTTP que proporcionan información adicional. Algunos importantes son:

Encabezados de solicitud (del cliente al servidor)

- **User-Agent** : Indica el navegador o cliente que realiza la solicitud.
- **Accept** : Especifica los tipos MIME que el cliente puede procesar.
- **Accept-Charset, Accept-Encoding, Accept-Language** : Indican qué codificación, compresión e idioma prefiere el cliente.
- **Host** : Especifica el nombre del servidor (obligatorio en HTTP/1.1).
- **Authorization** : Envía credenciales para acceder a recursos protegidos.
- **Referer** : Indica la página desde la que se hizo la solicitud.
- **Cookie** : Envía cookies previamente guardadas al servidor.

Encabezados de respuesta (del servidor al cliente)

- **Set-Cookie** : Envía una cookie al cliente.
- **Server** : Indica la versión del servidor web.
- **Content-Encoding, Content-Language, Content-Length, Content-Type** : Describen la codificación, idioma, tamaño y tipo de contenido de la respuesta.
- **Last-Modified, Expires** : Indican la última modificación del recurso y su fecha de expiración.
- **Location** : Redirige a una nueva URL.
- **Accept-Ranges** : Indica si el servidor admite respuestas parciales.

Encabezados compartidos (usados en ambas direcciones)

- **Date** : Indica la fecha y hora de la solicitud o respuesta.
- **Range** : Solicita solo una parte del contenido.
- **Cache-Control** : Controla el almacenamiento en caché.
- **ETag** : Proporciona una etiqueta única para identificar versiones de un recurso.
- **Upgrade** : Sugiere cambiar a otro protocolo más avanzado.

Estos encabezados son esenciales para optimizar la comunicación entre clientes y servidores, mejorar la seguridad y gestionar la caché de contenido.

Almacenamiento en Cache

El almacenamiento en caché es un mecanismo que permite a los navegadores y servidores almacenar copias de páginas web y otros recursos (como imágenes, hojas

de estilo y scripts) para reutilizarlos en el futuro sin necesidad de descargarlos nuevamente. Esto mejora el rendimiento, reduce el tráfico en la red y disminuye la latencia.

El problema principal del almacenamiento en caché es determinar si una copia almacenada sigue siendo válida. HTTP utiliza dos estrategias para resolver esto:

1. **Validación de Caché** : Si el navegador tiene una copia válida de un recurso, puede usarla sin hacer una nueva solicitud al servidor. Se utilizan encabezados como `Expires` y `Last-Modified` para indicar cuándo expira la caché.
2. **GET Condicional** : Si el cliente no está seguro de si la copia en caché sigue siendo válida, envía una solicitud condicional con los encabezados `If-Modified-Since` (basado en `Last-Modified`) o `If-None-Match` (basado en `ETag`). Si el servidor confirma que la copia sigue siendo válida, responde con un código `304 Not Modified`, evitando reenviar el recurso completo.

Adicionalmente, el encabezado `Cache-Control` permite configurar políticas de caché, como `no-cache` para evitar el almacenamiento o `max-age` para definir el tiempo de vida de la caché.

El almacenamiento en caché también puede realizarse en servidores intermedios (proxy caching), lo que beneficia a múltiples usuarios dentro de una red.

Audio y video de flujo continuo

Audio digital

- **Audio Digital** : El audio digital es la representación digital de una onda de audio. Se convierte a formato digital mediante un Convertidor Analógico-Digital (ADC), que toma muestras de la onda de sonido a intervalos regulares (ΔT) y produce números binarios.
- **Convertidores** : El proceso inverso, convertir valores digitales a señales analógicas, se realiza mediante un Convertidor Digital-Analógico (DAC). Luego, un altavoz convierte la señal analógica en ondas acústicas que se pueden escuchar.

📂 FTP (File Transfer Protocol)

FTP (File Transfer Protocol) es un protocolo de red utilizado para transferir archivos entre un cliente y un servidor a través de una red TCP/IP. Se basa en un modelo cliente-servidor y puede operar en dos modos: **activo y pasivo** .

1. Arquitectura de FTP

FTP usa **dos conexiones** entre el cliente y el servidor:

- **Canal de control (Puerto 21):** Se usa para enviar comandos y recibir respuestas.
- **Canal de datos (Puerto 20 o dinámico):** Se usa para la transferencia de archivos.

2. Modos de Funcionamiento

FTP tiene dos modos de operación:

Modo Activo:

1. El cliente inicia la conexión al servidor FTP en el **puerto 21** (canal de control).
2. El cliente informa al servidor qué puerto usará para recibir datos.
3. El servidor inicia la conexión de datos desde su puerto 20 al puerto indicado por el cliente.

💡 *Problema:* Este modo puede ser bloqueado por firewalls, ya que el servidor inicia la conexión de datos hacia el cliente.

Modo Pasivo:

1. El cliente se conecta al servidor en el **puerto 21** (canal de control).
2. El cliente solicita al servidor que abra un puerto dinámico para la transferencia de datos.
3. El cliente se conecta a ese puerto y la transferencia ocurre.

💡 *Ventaja:* Como el cliente inicia ambas conexiones, este modo es más compatible con firewalls y NAT.

3. Tipos de Transferencia

- **ASCII:** Para archivos de texto.
- **Binario:** Para archivos como imágenes, vídeos y programas ejecutables.

---

4. Modos de Autenticación

- **FTP Anónimo:** No requiere credenciales, común en sitios públicos de descarga.
- **FTP con Autenticación:** Se necesita un usuario y contraseña para acceder.

---

5. Alternativas Modernas

Dado que FTP no cifra los datos, existen protocolos más seguros:

- **SFTP (Secure FTP):** Usa SSH para encriptar la transferencia.
- **FTPS (FTP Secure):** Usa SSL/TLS para cifrar la comunicación.

💬 IRC (Internet Relay Chat)

IRC (Internet Relay Chat) es un protocolo de comunicación en tiempo real que permite a múltiples usuarios chatear en canales temáticos dentro de una red de servidores IRC. Puerto 6667 (TCP)

1. Arquitectura de IRC

IRC usa un modelo **cliente-servidor** en el que los usuarios se conectan a un **servidor IRC** y participan en **canales** (salas de chat).

🔹 **Cliente IRC:** Software que permite a los usuarios conectarse a un servidor. Ejemplos: mIRC, HexChat, irssi. 🔹 **Servidor IRC:** Mantiene la conexión de los clientes y distribuye los mensajes. 🔹 **Red IRC:** Conjunto de servidores interconectados que forman una comunidad.

---

2. Funcionamiento Básico

1. El usuario abre un cliente IRC y se conecta a un **servidor** usando un **nickname** .
2. Puede unirse a **canales de chat** (por ejemplo, `#tecnologia`).
3. Puede enviar mensajes públicos o privados.
4. Los servidores intercambian información para mantener la red sincronizada.

💡 *Ejemplo de conexión en consola:*

```
/connect irc.servidor.com
/join # canal
/msg usuario Hola!
/quit
```

---

3. Tipos de Comunicación

- **Mensajes públicos:** Se envían en un canal visible para todos los participantes.
- **Mensajes privados (PMs):** Se envían directamente a un usuario.
- **Acciones y comandos:** Se pueden usar comandos como `/me está escribiendo...` para indicar una acción.

---

4. Seguridad en IRC

IRC no cifra los mensajes por defecto, pero se pueden usar extensiones como:

- **SSL/TLS:** Cifra la conexión entre cliente y servidor.
- **SASL:** Autenticación segura con credenciales encriptadas.

⚠️ *Problemas comunes:*

- **Flooding:** Envío masivo de mensajes para colapsar un canal.
- **Bots maliciosos:** Usuarios automatizados que pueden generar spam o ataques DDoS.

5. Modos de Usuario y Canal

Los canales pueden tener restricciones y permisos:

- `+o` (operador): Puede administrar usuarios y el canal.
- `+v` (voz): Puede hablar en canales moderados.
- `+b` (ban): Bloquea a un usuario.
- `+k` (clave): El canal requiere contraseña.

Ejemplo para asignar un operador:

```
/mode #canal +o usuario
```

---

6. Alternativas Modernas

IRC ha perdido popularidad en favor de plataformas como Discord, Slack o Telegram, pero sigue siendo usado en comunidades técnicas y de código abierto.

Flujo de Conexion IRC

1. **Inicio de la Conexión** : Un cliente IRC se conecta a un servidor IRC especificando la dirección IP o el nombre de dominio del servidor y el puerto (por defecto, el puerto 6667)
2. **Establecimiento de la Conexión TCP** : El cliente y el servidor establecen una conexión TCP. Esto implica un proceso de "handshake" donde el cliente envía una solicitud de conexión al servidor, y el servidor acepta la conexión
3. **Identificación del Cliente** : Una vez establecida la conexión TCP, el cliente envía información de identificación al servidor, que incluye el nombre de usuario, el nombre del host y el nombre de la red.
4. **Registro en el Servidor** : El cliente debe registrarse en el servidor IRC enviando un comando `NICK` para establecer su apodo y un comando `USER` para proporcionar información adicional sobre sí mismo.

    1. ```
       NICK mi_apodo
       USER mi_usuario 0 * :Mi nombre
       ```

🔐 SSH (Secure Shell)

SSH es un protocolo de red que permite acceder de forma segura a sistemas remotos a través de una conexión cifrada.

🔹 **Características principales:**

- Usa **cifrado** para proteger datos transmitidos.
- Permite **acceso remoto** seguro a servidores.
- Soporta **túneles cifrados** para otras conexiones.
- Usa **claves públicas/privadas** para autenticación.

🔹 **Protocolo y puertos:**
       ```

- Funciona sobre **TCP** .
- **Puerto por defecto: 22** .

```
ssh usuario@servidor.com
```

Flujo de Conexion SSH

- **Inicio de la Conexión** : El cliente SSH inicia la conexión al servidor SSH especificando la dirección IP o el nombre de dominio del servidor y el puerto (por defecto, el puerto 22).
- **Establecimiento de la Conexión TCP** : El cliente y el servidor establecen una conexión TCP. Esto implica un proceso de "handshake" donde el cliente envía una solicitud de conexión al servidor, y el servidor acepta la conexión.
- **Intercambio de Versiones** : Una vez que se establece la conexión TCP, el cliente y el servidor intercambian información sobre sus versiones de SSH. Esto asegura que ambas partes puedan comunicarse utilizando un conjunto compatible de protocolos.
- **Autenticación del Servidor** : El servidor envía su clave pública al cliente. El cliente verifica la autenticidad del servidor mediante la comparación de la clave pública recibida con las claves almacenadas localmente. Si es la primera vez que se conecta, el cliente puede recibir un aviso de advertencia y deberá aceptar la clave.
- **Establecimiento del Canal Seguro** : Después de la autenticación del servidor, el cliente y el servidor negocian un conjunto de algoritmos de cifrado y crean un canal seguro mediante el uso de técnicas de cifrado simétrico. Esto asegura que los datos transmitidos estén protegidos durante la sesión.
- **Autenticación del Cliente** : El servidor solicita al cliente que se autentique. Esto puede hacerse mediante diferentes métodos, como:
- - **Contraseña** : El cliente envía su nombre de usuario y contraseña.
  - **Claves públicas/privadas** : El cliente presenta su clave pública, y el servidor verifica si la clave privada correspondiente está autorizada.
- **Acceso Remoto** : Una vez que el cliente ha sido autenticado, se establece una sesión de shell interactiva o se pueden ejecutar comandos de forma remota. Todas las comunicaciones entre el cliente y el servidor están cifradas y son seguras.
- **Túneles Cifrados (Opcional)** : SSH permite crear túneles cifrados para redirigir tráfico de otras aplicaciones (por ejemplo, tráfico HTTP) a través de la conexión

SSH, lo que proporciona una capa adicional de seguridad.
- **Cierre de la Conexión** : Cuando el usuario finaliza la sesión, el cliente envía una solicitud para cerrar la conexión. El servidor confirma el cierre, y la conexión TCP se finaliza.

⏰ NTP (Network Time Protocol)

NTP es un protocolo usado para **sincronizar el reloj** de dispositivos en una red con una fuente de tiempo precisa.

🔷 **Características principales:**

- Usa una jerarquía de servidores de tiempo (estratos).
- Puede corregir retrasos en la red y ajustar la hora gradualmente.
- Es crucial para sistemas donde la sincronización del tiempo es importante (servidores, telecomunicaciones, etc.).

🔷 **Protocolo y puertos:**

- Funciona sobre **UDP** .
- **Puerto por defecto: 123** .

```
ntpdate -q pool.ntp.org
```

Respuesta en la terminal

```
server 198.46.254.130, stratum 2, offset -1.427396, delay 0.17189
server 23.150.40.242, stratum 2, offset -1.424596, delay 0.18794
server 108.181.220.94, stratum 4, offset -1.427943, delay 0.18810
server 74.6.168.73, stratum 2, offset -1.429125, delay 0.25006
26 Feb 13:24:44 ntpdate[226]: step time server 198.46.254.130 offset -1.427396 sec
```

Flujo de Conexion:

- **Inicialización** : El cliente NTP se configura para comunicarse con uno o varios servidores NTP. Esto puede incluir la especificación de la dirección IP o el nombre de dominio del servidor.
- **Solicitud de sincronización** : El cliente envía un paquete de solicitud de sincronización (NTP request) al servidor. Este paquete incluye un timestamp que

marca el momento en que se envió la solicitud.

- **Recepción de la solicitud** : El servidor NTP recibe la solicitud del cliente y registra el tiempo en que se recibió. Este timestamp es importante para calcular el tiempo de red.
- **Cálculo del tiempo** : El servidor responde al cliente enviando un paquete que incluye varios timestamps:
  - El tiempo en que recibió la solicitud.
  - El tiempo en que se generó el paquete de respuesta.
  - El tiempo en que el servidor considera que está (su hora actual).
- **Recepción de la respuesta** : El cliente recibe el paquete de respuesta del servidor. En este punto, tiene cuatro timestamps:
  - T1: el tiempo en que el cliente envió la solicitud.
  - T2: el tiempo en que el servidor recibió la solicitud.
  - T3: el tiempo en que el servidor envió la respuesta.
  - T4: el tiempo en que el cliente recibió la respuesta.

- **Cálculo del desplazamiento y latencia** :
  - **Latencia** (delay) se calcula como: $D = (T4 - T1) - (T3 - T2)$
  - **Desplazamiento** (offset) se calcula como: $O = \frac{(T2-T1)+(T3-T4)}{2}$
- **Sincronización del tiempo** : Usando el desplazamiento calculado, el cliente ajusta su reloj local para sincronizarse con el tiempo del servidor.
- **Ciclo de sincronización** : El cliente puede repetir este proceso periódicamente para mantener su hora actualizada, adaptándose a los cambios en la red o en el reloj del servidor.