Resumen Tema 5

Servicios y protocolos de aplicación en internet

Autor: @BlackTyson

Índice

1. Introducción a las aplicaciones de red

1.1. Interacción cliente-servidor

Servidor

- **Siempre en funcionamiento.**
- **IP permanente y pública.**
- **Agrupados en granjas.**

Cliente

- **Funcionan intermitentemente.**
- **Pueden tener IP dinámica y privada.**
- **Se comunican con el servidor, pero no entre sí.**

1.2. Interfaz socket

- Un proceso envía/recibe mensajes a/desde su ***socket***.
- Para recibir esos mensajes, un proceso debe estar identificado (**IP + puerto**).
- El *socket* es un descriptor de transmisión por el cual la aplicación puede enviar/recibir información desde otro proceso; funciona como **"puerta"** entre la aplicación y el transporte.
- Se implementa como una variable puntero a una estructura en el sistema operativo.

1.3. Retardo en cola

Teoría de colas

- El uso de un servidor se modela mediante un sistema M/M/1.
- El retardo de cola se expresa como:

$$R = \frac{\lambda T_s^2}{1 - \lambda T_s}$$

  - T_s es el valor esperado del tiempo de servicio.
  - λ es el valor esperado de la tasa de llegada.
- La misma expresión puede aplicarse al cálculo de retardo en cola de un **router**.

1.4. Qué definen los protocolos de aplicación

- **Tipo de servicio**:
  - **Orientado o no a conexión.**
  - **Con o sin realimentación.**
- **Tipo de mensaje**:
  - **Request** (*solicitud*).
  - **Response** (*respuesta*).
- **Sintaxis**:
  - **Definición y estructura** de campos en el mensaje.
  - **Uso de HTTP o DNS**.
  - **Uso de** `Type_Length_Value`.
- **Semántica**: significado de los campos.
- **Reglas**: cuándo y para qué las entidades solicitan o responden mensajes.

Tipos de protocolo

- **Dominio público** (**HTTP**) vs **propietarios** (**Skype**).
- **In-Band vs Out-of-Band**.
- *Stateless* vs *state-full*.
- **Persistentes vs no-persistentes**.

Tendencia

Protocolos flexibles con:

- **Cabecera fija.**
- **Chunks** obligatorios y opcionales. Pueden incluir una cabecera específica más una serie de datos:
  - **Parámetros fijos (ordenados).**
  - **Parámetros de longitud variable.**
  - **En formato TLV (Type–Length–Value).**

1.5. Características

- **Tolerancia a pérdidas de datos (errores)**: algunas aplicaciones (audio) pueden tolerar pérdidas ligeras, mientras que otras (FTP) requieren fiabilidad total.
- **Exigencia de requisitos temporales**: algunas aplicaciones requieren retardo acotado; en otras no es necesario.
- **Demanda de ancho de banda**: algunas requieren tasa mínima de envío, otras no.
- **Nivel de seguridad**: muy variable, según la finalidad de la aplicación.

Diferentes aplicaciones tienen distintos requisitos.

1.6. Protocolos de transporte

TCP y **UDP**, al ser usuarios de **IP**, no garantizan calidad de servicio:

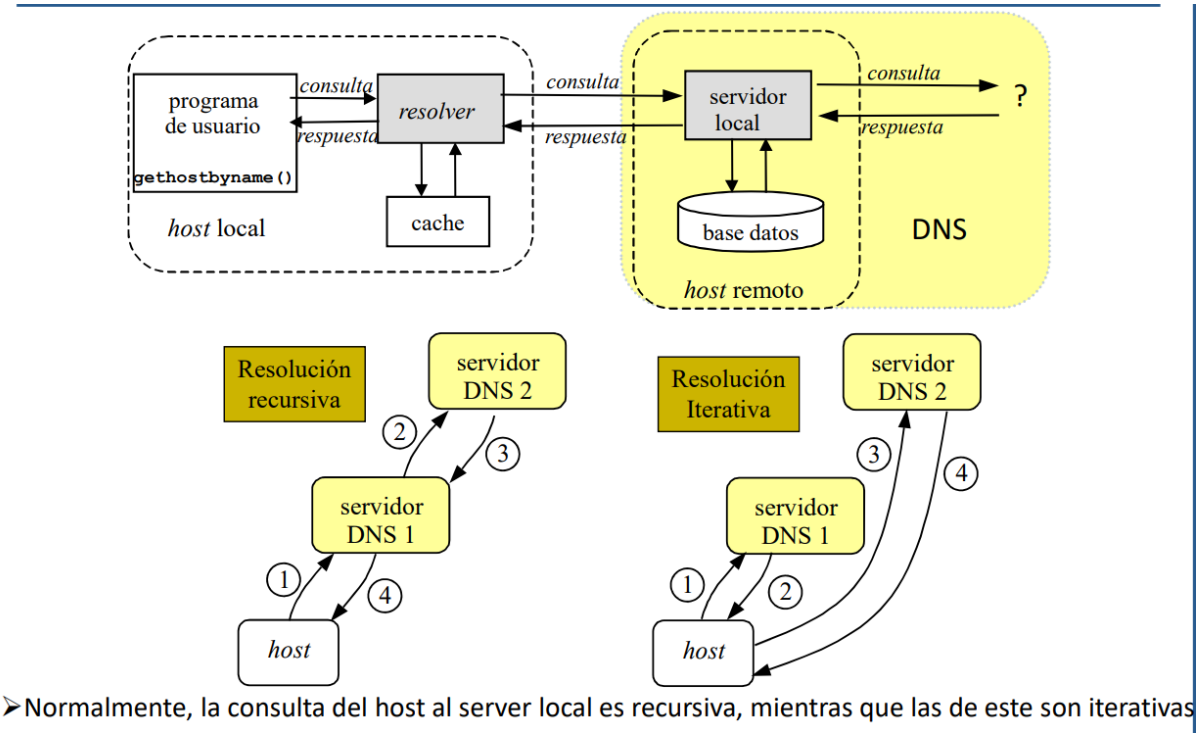
- **Retardo no acotado.**
- **Fluctuaciones en el retardo no acotadas.**
- **No hay velocidad mínima garantizada.**
- **Probabilidad de pérdidas no acotada.**
- **No hay garantías de seguridad.**

2. Servicio de Nombres de Dominio (DNS)

2.1. Fundamentos

- **Los nombres de dominio** simplifican el uso de direcciones IP. El **DNS** traduce nombres a IP (y viceversa). △ Ejemplo: `www.ugr.es` ↔ **150.214.204.25**
- **Estructura jerárquica**: el espacio de nombres es un árbol con raíz gestionado por el **ICANN**.
- **TLDs** (**Top Level Domains**): `.com`, `.es`, etc.
- **DNS** es un protocolo de aplicación para acceder a una **base de datos distribuida**.
- Se utilizan **cachés** para mejorar prestaciones.
- **Puerto 53**, tanto sobre **UDP** como **TCP**.

Ejemplo de resolución de nombres DNS



En la figura se ilustra cómo un ***host*** realiza consultas para **resolver nombres de dominio** a través de **DNS**, mostrando tanto la **resolución recursiva** como la **iterativa**:

- **Host local y `resolver`**:
  - Un **programa de usuario** (que llama a `gethostbyname()`) solicita la resolución de un nombre de dominio.
  - El `resolver` local primero revisa su **caché** para ver si tiene la respuesta.
  - Si no está en caché, el `resolver` envía la consulta al **servidor DNS local**, que puede consultar su propia base de datos.
- **Resolución recursiva**:
  - El **host** confía en el **servidor DNS 1** para encontrar la dirección IP.
  - El **servidor DNS 1** realiza consultas a otros servidores (**DNS 2**, etc.) hasta hallar la respuesta.
  - Flechas (1), (2), (3), (4) en el diagrama: peticiones/respuestas sucesivas, tipo **"pídeme todo y devuélveme el resultado final"**.
- **Resolución iterativa**:
  - El **host** realiza sucesivas consultas a servidores DNS, recibiendo **información parcial** (referencias) hasta encontrar la respuesta.
  - Flechas (1), (2), (3), (4): el **host** pregunta a **DNS 1**, recibe pista, continúa con **DNS 2**, etc.
  - Cada respuesta puede indicar una **redirección** u orientación para la siguiente consulta, hasta obtener la IP final.

En **resumen**, la **resolución recursiva** delega todo el proceso de búsqueda en el primer servidor, mientras que la **iterativa** hace que el **host** realice cada paso. Así, el `resolver` puede seleccionar distintos modos según configuración y política del servidor DNS.

2.2. Conceptos de la base de datos DNS

- Sistema **DNS** formado por servidores cooperativos en árbol, **BIND** (*Berkeley Internet Name Domain*).
- Cada servidor es responsable de una **zona**.
- Cada **zona** incluye nombres de dominio completos sobre los que un servidor tiene autoridad.
- La autoridad puede **delegarse** jerárquicamente a servidores "hijo".

2.3. Cómo es la BBDD DNS

- Se almacenan **zone files** (**ficheros de texto**).
- Cada **zone file** contiene **Resource Records** (**RR**).
- Cada **zone file** define un **TTL** (*Time To Live*) que indica el tiempo de validez de un **RR** en caché.
- Cada **RR** contiene:
  - **Nombre de dominio.**
  - **Clase** (por ejemplo, *IN*).
  - **Tipo de registro.**
  - **Valor** (dependiente del tipo).
- Existe una **base de datos de resolución inversa** para traducir **IP** a nombres de dominio.
- La **resolución inversa** se usa para obtener el **nombre de dominio** asociado a una **IP**.

2.4. Gestión de la BBDD DNS

- **Delegar autoridad** implica ceder gestión y responsabilidad a otro servidor.
- **Servidores primarios**: almacenan una copia **máster** de la BBDD en disco local.
- **Servidores secundarios**: obtienen la copia por **transferencia**.
- Los servidores **mantienen caché**.
- Para escalabilidad, hay **13 servidores raíz** (réplicas con la misma **IP**) repartidos globalmente.
- En **Madrid** existe un **punto neutro**.
- Un servidor puede:
  - **Responder con autoridad**: conoce y **devuelve la IP**.
  - **Responder sin autoridad**: no es autoridad, pero lo tiene **en caché**.
  - **No dar respuesta**: consulta a otros servidores (**recursivo** o **iterativo**); puede llegar al servidor **raíz**.

3. Navegación web (HTTP)

3.1. Fundamentos

- Una **página web** es un fichero con varios **objetos**, cada uno con su **URL** o **URI**.
- Se sirve mediante **HTTP**, en un modelo **cliente-servidor**:
  - **Cliente**: navegador.
  - **Servidor**: envía objetos web a peticiones.
- Las páginas pueden ser **estáticas** o **dinámicas**.
- Las **dinámicas** generan contenido variable, usando:
  - Scripting en **cliente** (*Frontend*).
  - Scripting en **servidor** (*Backend*).

3.2. Características del protocolo HTTP

- Emplea **TCP** en el **puerto 80**.
- Es **stateless**: el servidor no conserva información de estado de las peticiones.
- Usa **cookies** para mantener cierta continuidad con el usuario.
- Es **in-band**, orientado a **texto**.
- Tipos de **servidores HTTP**:
  - **No persistente** (HTTP/1.0): **un objeto por conexión**.
  - **Persistente** (HTTP/1.1): **múltiples objetos por conexión**.

3.3. Navegación

1. El **cliente** (navegador) solicita un objeto por su **URL**.
2. Consulta al **resolver DNS** para obtener la **IP**.
3. **DNS** responde con la IP.
4. El cliente abre una **conexión TCP** al **puerto 80**.
5. El cliente envía una petición `GET`.
6. El **servidor** responde con el fichero `index.html` por la misma conexión (TPC)
7. Si es persistente, se piden más objetos en la misma sesión.
8. Se cierra la **conexión TCP**.
9. El cliente muestra el contenido.

3.4. Métodos HTTP

- `OPTIONS`
- `GET`
- `HEAD`
- `POST`
- `PUT`
- `DELETE`

3.5. Respuestas HTTP

- **1xx**: información
- **2xx**: éxito
- **3xx**: redirección
- **4xx**: error de cliente
- **5xx**: error de servidor

3.6. Cabeceras HTTP

- `Content-Type`: descripción de la información.
- `Content-Length`: longitud de los datos enviados.
- `Content-Encoding`: formato de codificación.
- `Date`: fecha local de la operación.
- `Accept` (**cliente**): lista de tipos aceptados.
- `Authorization` (**cliente**): clave de acceso a recurso protegido.
- `From` (**cliente**): correo del usuario (opcional).
- `If-Modified-Since` (**cliente**): para **GET condicional**.
- `Referer` (**cliente**): URL del documento origen.
- `User-Agent` (**cliente**): indica tipo y versión del navegador.
- `Allow` (**servidor**): métodos HTTP válidos para el objeto.
- `Expires` (**servidor**): fecha de caducidad del objeto.
- `Last-Modified` (**servidor**): fecha de modificación local del objeto.

3.7. Caché

- El objetivo es **reducir tráfico** sin servir contenido **desactualizado**.
- El usuario configura el navegador para cursar solicitudes vía **proxy**.
- La **caché** puede estar en el ordenador del usuario.
- El navegador envía los requerimientos a la **caché**, que responde si tiene la información válida.

3.8. Cookies

Son ficheros de texto que resuelven la **falta de estado** en HTTP:

- La primera vez que se accede a un servidor, éste puede proporcionar una **cookie** para usos posteriores.
- El **cliente** almacena la cookie y la reenvía en accesos sucesivos.
- Esto ocurre **de forma transparente**, sin intervención del usuario (salvo configuraciones específicas).

Variables predefinidas en cookies

- **Domain**: direcciones válidas para la cookie.
- **Path**: subconjunto de **URLs** válidas.
- **Version**: versión del modelo de cookie.
- **Expires**: fecha de caducidad.

3.9. Acceso restringido

HTTP no es seguro, pero incluye cabeceras de **autenticación** y **autorización** para limitar acceso. Sigue siendo vulnerable a **ataques por repetición**.

4. Correo electrónico

4.1. Elementos y protocolos principales

- **Cliente de correo (MUA)**: compone, edita y lee mensajes.
- **Servidor de correo (MTA)**: reenvía salientes y almacena entrantes, desacoplando remitente/destinatario.
- **Protocolo de envío**: **SMTP** (*Simple Mail Transfer Protocol*).
  - Dos programas:
    - **Cliente SMTP** (en MUA o MTA que envía).
    - **Servidor SMTP** (en MTA que recibe).
  - Usa **TCP** en **puerto 25**, orientado a texto.
  - Fases: **handshaking**, transferencia y **cierre**.
  - Comandos **ASCII** y respuestas (códigos de estado).
- **Protocolo de descarga**: **POP3**, **IMAP** o **HTTP**.

4.2. Pasos en el envío/recepción de correo

1. El usuario origen (MUA) compone un mensaje para la dirección destino.
2. Se envía con **SMTP** al **MTA** de origen, que lo pone en cola de **salientes**.
3. El **cliente SMTP** abre conexión con el **MTA** del destino.
4. **SMTP** transfiere el mensaje por **TCP** (puerto 25).
5. El **MTA** destino ubica el mensaje en el **mailbox**.
6. El usuario destino lee el mensaje con **POP3**, **IMAP** o **HTTP**.

4.3. Descarga/lectura de correo

- **POP3**:
  - **Puerto 110 TCP**.
  - Fases:
    - **Autorización** (`user`, `pass`; respuestas `+ok`, `-err`).
    - **Transacción**:
      - `list`: lista mensajes.
      - `retr`: obtiene mensajes.
      - `dele`: borra.
      - `quit`: termina sesión.
    - **Actualización**: tras `quit`.
- **IMAP4**:
  - Organización en carpetas en el **servidor**.
  - Mantiene información entre sesiones.
  - Facilita descargas parciales.
  - Permite acceso simultáneo con varios clientes.
- **HTTP**:
  - Toda la organización reside en el **servidor**.
  - Puede usar **HTTPS** para mayor seguridad.

5. Aplicaciones multimedia

5.1. Conceptos

- **Aplicaciones multimedia**: audio, vídeo, etc.
- **QoS** (Quality of Service): capacidad de ofrecer un **rendimiento** adecuado.
- **IP** trabaja en modo **best-effort** (sin garantías de **QoS**).

5.2. Tipos de aplicaciones

- **Flujo de audio/vídeo almacenado**: *YouTube*, etc.
- **Flujo de audio/vídeo en vivo**: *radio en línea*.
- **Audio/vídeo interactivo**: *videollamadas*, *Skype*, etc.

5.3. Características

- **Elevado ancho de banda.**
- **Tolerancia relativa a pérdidas de datos.**
- **Necesidad de retardo acotado.**
- **Control de fluctuación (jitter).**
- **Posible uso de multicast.**