Redes II

Resumen de Cuestionario

Emiliano Salvatori

Agosto 2019

1. Cuestionario clase nº 2

1.1. Definición de Socket

Los procesos envían mensajes desde la capa de aplicación a la capa de transporte mediante una API que se denomina Socket, el cual depende del tipo de protocolo que se vaya a emplear (TCP o UPD), se utiliza un socket u otro. El desarrollador de aplicaciones controla totalmente el proceso, pero a la hora de decidir cómo se enviará el mensaje a la capa de transporte, si será TCP o UDP, esta capa le deja configurar unos pocos parámetros, como el tamaño del buffer o del segmento.

Un socket es también una dirección de Internet, combinando una dirección IP (la dirección numérica única de cuatro partes que identifica a un host o terminal en Internet) y un número de puerto (el cual identifica a la aplicación en particular).

1.2. Describa como es el funcionamiento de las Cookies y para qué sirve. ¿Riesgos asociados?

HTTP es un protocolo SIN memoria, por lo que si se quiere tener un control del historial de sus usuarios, hay que pedir que se registren y tener todas sus peticiones y transferencias en una base de datos y leerlas cada vez que ingresen a la página. El uso de COOKIES evita este procedimiento, almacenando en el navegador del lado del cliente, determinados datos.

Una cookie consiste en:

- Un par nombre-valor con datos almacenados.
- Una fecha de expiración a partir de la cual la cookie será inválida y eliminada.
- El dominio y el directorio del servidor a cual se enviará dicho dato.

El uso general de la cookie es:

- Cuando se quiere conectar por primera vez desde un navegador web a una página, por ejemplo Amazon, se envía un típico mensaje HTTP de solicitud.
- El Servidor de Amazon nos contestará el mensaje de petición enviado PERO con una solicitud de cookies que se guardará en nuestro navegador, de la siguiente forma: set-cookie: 1234. Por otro lado, el servidor de Amazon, generará en su base de dato la entrada 1234 para guardar información sobre mis conexiones.
- Cada vez que nos conectemos a Amazon, lo que hará nuestro navegador es enviarle la cookie almacenada con el número 1234 para que el servidor pueda buscar en su base datos toda la información nuestra y así poder tener disponibles todo el historial de navegación.

El problema que tiene el uso de las cookies es la privacidad:

- Se puede conocer información personal: nombre, email, etc.
- Pueden guardar información a lo largo de internet, como por ejemplo los anuncios que se muestran.
- Se puede hacer un seguimiento del historial de la navegación del usuario.
- Muchas empresas venden sus cookies a terceros, eso generó una controversia por lo que en la Unión Europea se decretaron leyes que atentan contra el desconocimiento de los navegantes en páginas que guardan cookies.

1.3. Explique y describa la Arquitectura Cliente-Servidor

En una arquitectura cliente-servidor siempre existe un host activo, denominado servidor, que da servicio a las solicitudes de muchos otros hosts, que son los clientes. Los hosts clientes pueden estar activos siempre o de forma intermitente. Un ejemplo clásico es la Web en la que un servidor web siempre activo sirve las solicitudes de los navegadores que se ejecutan en los hosts clientes. Cuando un servidor web recibe una solicitud de un objeto de un host cliente, responde enviándole el objeto solicitado.

Con la arquitectura cliente-servidor, los Host clientes no se comunican directamente entre sí; por ejemplo, en la aplicación web, dos navegadores no se comunican entre sí. Otra característica de la arquitectura cliente-servidor es que el servidor tiene una dirección fija y conocida, denominada dirección IP. Puesto que el servidor tiene una dirección fija y conocida, y siempre está activo, un cliente siempre puede contactar con él enviando un paquete a su dirección. Entre las aplicaciones más conocidas que utilizan la arquitectura cliente-servidor se encuentran las aplicaciones web, FTP, Telnet y de correo electrónico.

1.4. ¿Qué tipos de protocolos existen y que definen los protocolos de la capa de Aplicación?

Los protocolos de dominio público son los definidos por los RFC's: Por ejemplo HTTP o SMTP. Los protocolos de propietarios son por ejemplo aquellos que utiliza KaZaA o Skype.

Definen lo siguiente:

- Definen los tipos de mensajes intercambiados, por ejemplo: mensajes de requerimiento y respuesta.
- Definen la sintaxis de los tipos de mensajes: los campos en los mensajes que utilizan & son delimitados.
- Definen la semántica de los campos, por ejemplo el significado de la información en los campos.
- Por último definen las reglas para cuándo y cómo los procesos deben enviar y responder a mensajes.

1.5. Diferencias entre Web Cache y Proxy

A medida que las comunicaciones fueron haciéndose más veloces, también se implementaron mejoras en la navegación. La idea es que haya un servidor intermediario, gestionando las peticiones que los clientes hacen hacia los servidores, para que en caso de que haya una segunda solicitud a una misma página ya visitada, no tener que remitir la solicitud al servidor original, sino que sea el servidor Proxy quien conteste primero esa petición, cuya página estará guardad en su memoria caché (no hay que confundir, ya que no es la memoria caché del navegador)

El Servidor Proxy será por el que pasarán todas las peticiones de un cliente, si tiene lo solicitado, este se lo devolverá, en caso contrario, redirigirá la petición al Servidor original y cuando el objeto pase por el servidor Proxy, se guardará en él en caso de que se vuelva a consultar.

Por lo tanto el Servidor Proxy actuará como servidor para mi navegador web, como de cliente para otros servidores webs. Estos son instalados por el ISP para acelerar la navegación y evitar tráfico innecesario.

1.6. Explique las Arquitecturas P2P e Hibridas. Ejemplos de c/u

Arquitectura Peer To Peer

También denominada **arquitectura entre pares o P2P**

1. A diferencia de la Arquitectura Cliente-Servidor, todos los Host pueden actuar como clientes o servidores de forma simultánea.
2. Por ello, los servidores NO siempre estarán activos.
3. Los host clientes se podrán comunicar entre sí, pudiendo también cambiar su dirección IP.
4. Esta arquitectura es altamente escalable pero muy difíciles de gestionar.

Inconvenientes:

- Uno de los inconvenientes que tiene esta arquitectura es que las infraestructuras de red que han desplegado los ISP están orientadas a arquitecturas Cliente-Servidor donde es mayor el tráfico de bajada que el de subida (Aunque el uso de fibra óptica está re-dimensionando las líneas, haciendo un uso simétrico del ancho de banda).

- Otro de los problemas es la seguridad, dada la naturaleza extremadamente distribuida y abierta, es como poner puertas al campo.
- Con estas arquitecturas se han de crear políticas de incentivos para convencer a los usuarios y que ofrezcan voluntariamente ancho de banda, almacenamiento y computación.

Ejemplo de programas que utilizan la Arquitectura P2P:

- BitTorrent.Net
- G3 Torrent
- mlMac
- MXIE
- QTorrent
- Shareaza
- MuTorrent
- Vuze

Arquitecturas Híbridas

Hay arquitecturas p2p donde no todos los host son iguales, por lo que no son p2p puras. Son híbridas.

- Se tienen pares comunes (peers)
- Pero también se tienen super-peers por lo que se genera una jerarquía en la red.
- Se tiene un servidor centralizado de login, pero el índice que asigna los nombres de usuario a las direcciones IP está distribuido entre los super-peers, probablemente con una tabla hash distribuida.
- Tiene un protocolo privado por lo que no se sabe a ciencia cierta cómo trabaja.

Una de las aplicaciones más conocidas y utilizada que emplea la **Arquitectura P2P** es Skype.

1.7. Para que se usa el GET Condicional

Para actualizar el contenido de los servidores proxy se utiliza el comando GET condicional, mediante el cual se puede introducir en la cabecera, ciertas condiciones para que se traiga el objeto del servidor sólo en caso de que se cumplan. Por ejemplo la cláusula: *if-modified-since* es decir, si ha cambiado desde la última fecha que tiene constancia. Esta fecha se obtuvo en el proxy cuando le llegó el objeto por última vez en su campo last-modified. En este caso, si el servidor proxy tiene el objeto actualizado el servidor le devolverá un 304 not-modified evitando enviar un objeto de nuevo. En caso contrario, si el objeto ha cambiado, le devolverá un 200 OK con el objeto solicitado.

1.8. Compare los protocolos HTTP persistente con y sin pipeline. Como serían los tiempos si se quiere bajar con 1 archivo HTML con 5 objetos

Otra mejora es la denominada *pipelining* que consta de NO esperar a recibir el objeto completo para pedir el siguiente, sino hacer varias peticiones, simultáneas al servidor y que este entregue los objetos según los vaya enviando, o recibiendo las solicitudes de objetos. Esto puede llevar a problemas en algunos navegadores, porque algunos objetos pueden bloquear a otros objetos más importantes. Y es difícil implementarlo correctamente para que la mejora sea apreciable, esto hace que todos los navegadores tengan deshabilitado el *pipelining* por defecto. Se emplea un algoritmo mejor que se denomina *multiplexación*.

1.9. Que servicios proveen los Protocolos de transporte TCP y UDP

Servicio TCP:

- Es Orientado a la conexión establecer conexión (setup) requerido entre procesos cliente y servidor antes de transferencia
- Ofrece Transporte confiable entre proceso Transmisor (Tx) y Receptor (Rx)
- Tiene Control de flujo: Tx no sobrecargará al Rx
- Tiene Control de congestión: el Tx se frena cuando la red está sobrecargada
- No provee: garantías de retardo ni ancho de banda mínimos

Servicio UDP:

- Transferencia de datos no confiable entre proceso Tx y Rx.
- No provee: establecimiento conexión, confiabilidad, control de flujo, control de congestión, garantías de retardo o ancho de banda

¿Por qué existe UDP? UDP es más rápido para enviar rápido un paquete, se utiliza por eso. Porque no abre una conexión de forma previa.

1.10. Que tipos de servicios requieren las aplicaciones de la capa de transporte

La capa de Aplicación necesita:

- **Confiabilidad en la entrega (sin pérdida de datos)**: Algunas aplicaciones requieren transferencia 100 % confiable. Algunas otras si pueden tolerar pérdida (video y audio)
- **Retardo**: Algunas aplicaciones (como los video juegos) requieren bajo retardo para ser *efectivas*.
- **Ancho de Banda**: Algunas aplicaciones requieren una cantidad mínima de ancho de banda para ser *efectivas*. Y otras aplicaciones (*Aplicaciones elásticas*) hacen uso de la banda ancha que obtengan.

1.11. Compare los protocolos HTTP persistente y HTTP no persistente. Como serían los tiempos si se quiere bajar con 1 archivo HTML con 5 objetos

Los tipos de conexiones HTTP pueden ser:

- **Las conexiones no persistentes**: son aquellas donde se cierra la conexión TCP luego de haber transferido cada objeto solicitado.
- **Las conexiones persistentes**: son aquellas en las que se pueden enviar varios objetos antes de cerrar la conexión.

Para evitar este problema al cerrar la conexión luego de enviado el archivo (y con esto generar empobrecimiento en la transferencia de múltiples objetos), se utilizan conexiones persistentes.

Lo que se hace es mantener una conexión abierta entre cliente y servidor aún luego de haber transferido el objeto solicitado por el primero, así en caso de que se soliciten más objetos, se envián por la misma conexión abierta que se tiene. Por lo que el cliente emplea un solo RTT (Round Trip Time) para abrir la conexión y luego un RTT por cada objeto que sea transferido.

Round trip time: tiempo que tarda un paquete en ir del cliente al servidor y volver, sin tener en cuenta su tamaño. En una conexión TCP normal, el cliente envia una solicitud de conexión, y cuando el servidor le contesta que si, estableciendo una conexión, ese es el tiempo RTT.

En el caso práctico se tiene una página HTML con 5 objetos dentro. En las conexiones **HTTP NO persistentes** se tendrán 2 RTT por cada objeto solicitado, en este caso se solicitan 6 objetos (la página HTML con sus sucesivos cinco objetos), por lo que se tienen 12 RTT totales.

En cambio, en las conexiones **HTTP persistentes**, se utiliza 1 RTT para establecer la conexión con el Servidor y luego 1 RTT por cada uno de los objetos que se solicitan; por lo que en caso de pedir 6 objetos (la página HTML con sus sucesivos 5 objetos) se tendrán 7 RTT totales.

2. Questionario Clase nº 3

2.1. Explique el funcionamiento del protocolo FTP, mencionando para que se usa el puerto 20 y el puerto 21 y sus dos modos de trabajo: Activo y Pasivo

Su nombre significa *File Transfer Protocol* sirve para transferir ficheros entre un cliente y un servidor, el cual permite navegar por los directorios de ambos. Una de las cosas curiosas que tiene FTP es que crea dos conexiones, canales; por un canal se envían las señales de control y por el otro canal, los datos. Primero el cliente establece la conexión de control del puerto 21 del servidor, proporcionando clave y contraseña en caso de ser necesario. Una vez establecida, el cliente puede enviar comandos para navegar por los directorios del servidor, buscando la ubicación que le interese.

A la hora de conectarse con un servidor FTP, el cliente podría utilizar

- El modo ACTIVO (o modo PORT): Eligiendo un puerto aleatorio, por encima del 1024 y conectándose con el puerto 20 del servidor para enviar esos datos. Por ejemplo: el cliente elije el puerto 2000, envía un comando PORT al servidor indicándole el puerto que ha elegido, de manera que el servidor pueda abrirle una conexión de datos donde se transferirán los archivos y los listados en ese puerto. Esto tiene un **PROBLEMA DE SEGURIDAD** y es que la máquina cliente debe estar dispuesta a aceptar cualquier conexión por sobre el puerto 1023, con los problemas que ello implica en el firewall.
- Para solucionar el problema de seguridad que acarrea el Modo Activo, se desarrolló el Modo Pasivo (o modo PASV): cuando el cliente envía un comando PASV por el canal de control, el servidor FTP le indica su puerto (el puerto del servidor, el cual debe ser mayor que el 1023), por ejemplo 2040, al que debe conectarse el cliente. Por lo que el cliente es el que inicia la conexión desde el puerto siguiente al puerto que tenga el control (recordar que son dos puertos los que abre el cliente, uno para control (por ejemplo 1034) y otro para datos (por ejemplo 1035)). Y como es el cliente el que inicia la conexión, no hay problemas de seguridad. En caso de tener que abrir otro fichero, se debe abrir otra conexión.

Al final el protocolo FTP, sirve para lo mismo que el protocolo HTTP, traer ficheros de un servidor y ambos se ejecutan sobre TCP; de hecho los navegadores también interpretan el protocolo FTP, por lo que también sirven como clientes de este protocolo.

2.1.1. Diferencia entre FTP y HTTP

- La principal diferencia que existe entre FTP y HTTP es el uso que hace FTP de las dos conexiones (una parte para control, donde envía los comandos y otra para los datos). Mientras que en HTTP tanto comandos como datos, viajan sobre la misma conexión, mediante los mensajes HTTP.
- HTTP envía la información de cabecera de solicitud y respuesta por la misma conexión TCP que transfiere el archivo (en banda), mientras que FTP envía la información de control *fuera de banda* (lo mismo que realiza el protocolo RTSP).
- El servidor FTP tiene que mantener el estado, ya que debe recordar la carpeta en la que se encuentra el usuario, el modo de transferencia de ficheros que le han sido solicitados, etc.

2.2. Enumere los componentes del correo electrónico. Explique el protocolo SMTP (Simple Mail Transfer Protocol), quienes utilizan este protocolo y para que se usa.

Como elementos fundamentales para un servicio de Correo Electrónico se encuentran:

1. Los clientes de Correo: que pueden ser aplicaciones específicas como Thunderbird u Outlook, o directamente desde el navegador.
   - Compone, edita, lee y envía mensajes de correo.
   - Los correos entrantes y salientes se almacenan en el servidor.
2. Servidores de correo que tendrán:
   - Buzón (o mail box) que contiene los mensajes entrantes de un usuario.
   - Cola de mensajes de los correos salientes que tienen que ser enviados.
3. Protocolo SMTP (Simple Mail Transfer Protocol), el cual comunicará los servidores de correo con otros.
   - Utiliza TCP para enviar mails entre servidores de correo.
   - Lado cliente en el servidor de correo del emisor y lado servidor en el servidor de correo del destinatario.

2.2.1. Ejemplo de envío de correo electrónico

Si Alicia quiere enviar un correo a Bob, entonces lo que sucede es lo siguiente:

1. Alicia utiliza un Cliente de Correo (A), en el cual escribe o edita el correo deseado, y luego lo envía.
2. El Agente/Cliente de Correo de Alicia (B) envía el mensaje a su servidor de corro (un servidor aparte del Cliente que utiliza Alicia) que lo pone en su cola de mails.
3. El Servidor de Correo utiliza SMTP para contactarse con el otro Servidor de Correo (C) que utilizará el Cliente/Agente de Bob (D), y abre una conexión TCP.
4. El cliente SMTP envía el mensaje de Alicia por la conexión TCP (de B a C).
5. EL Servidor de Correo de Bob (C) almacena el mansaje en su buzón (mailbox).
6. Bob invoca a su Agente de Correo (D) para leerlo cuando quiera.

2.3. El agente Usuario de correo, que protocolo usa para enviar y recibir correos. Explique cada uno como interacciona con el servidor, ventajas y desventajas entre los protocolos POP3 e IMAP.

¿Cómo hace para acceder desde un Agente a un correo? Para ello existe los protocolos de acceso al correo. Entre ellos se encuentran:

- POP3: Post Office Protocol versión nº 3 [RFC 1939] el cual determina una autorización entre Agente y Servidor, la descarga y actualización.
- IMAP: Internet Mail Access Protocol [RFC 3501] el cual tiene muchísima más funcionalidad pero es más complejo; también permite la manipulación de mensajes almacenados en el servidor.
- HTTP: También puede ser utilizado como los que usan los ya conocidos: Gmail, Hotmail, Yahoo! Mail, etc.

IMAP: Es mucho más potente que POP3, ya que permite gestionar el buzón de correo creando carpetas para poder organizar los correos en el propio servidor por lo que tiene memoria de estado entre sesiones.
POP3: Nos permitirá acreditarnos a nuestro Servidor para poder acceder a nuestro buzón, y descargar así los correos que tengamos almacenados en él. Por defecto, POP3 descarga los correos al Agente de Mail y los borra del Servidor (aunque esto sse puede configurar para que los deje almacenados).

- Al conectarse, lo que hace primero es acreditarse para poder acceder al buzón.
- Luego, pide un listado de los correos y los va descargando y marcando para ser borrados.
- Al final, se envía la orden para que el Servidor borre los correos que han sido marcados para borrar.

2.4. ¿Cuándo me conecto por un browser que tipo de protocolo uso para enviar y recibir el correo en pantalla?

Cuando se quiere chequear el correo electrónico desde la web (como por ejemplo gmail o hotmail) se utiliza el protocolo HTTP, pero sólo entre los servidores utilizan en SMTP. Protocolos de acceso de correo siempre lo usan los Agentes de correo, solo ellos (outlook, thunderbird, etc). IMAP o POP3 son los protocolos para poder descargar los correos: POP3 e IMAP.

2.5. Para agregar seguridad a los protocolos de correo que se usa? ¿Los puertos son los mismos?

Para agregar seguridad a los protocolos de corre se utiliza el protocolo TLS, generando de esta manera una encriptación de punto a punto.
Los nuevos puertos que se usan por cuestiones de seguridad para correo (los cuales implementan TLS) son:

- 465
- 993
- 995 POP3

3. Questionario Clase nº 4

3.1. ¿Qué es un Domain Name System (DNS)?

En internet las máquinas tienen direcciones IP, pero para evitar recordar estos números se emplean nombres de dominio. El DNS o el Domain Name System, es el sistema que permite almacenar las correspondencias entre direcciones IP y nombres de dominio. También se emplea el nombre DNS para referirse al protocolo que se emplea para consultar esa base de datos distribuida. El servidor DNS suele ser máquinas de tipo UNIX que ejecutan BIND sobre UDP utilizando el puerto nº 53.

3.2. ¿Por qué no conviene centralizar el servicio de DNS?

Los servicios que provee DNS son los siguientes:

- Traducción del nombre del host a direcciones IP.
- También proporciona alias para esas máquinas.
- Alias para servidores de correo.
- También ayuda en la distribución de carga haciendo corresponder un único nombre con varias direcciones IP.

Todo esto lo hace de una forma transparente al usuario, además se hace de forma decentralizada. Esto es así ya que:

- Un único DNS sería un único punto de fallo.
- Con un gran volumen de tráfico
- Gran distancia a la Base de datos centralizada, al único nodo.
- Provocaría grandes problemas de mantenimiento.
- Como también contribuiría a tener problemas de escalado .

3.3. ¿Por qué se dice que es una base de datos distribuida y jerárquica (Explique cada una de sus partes)?

Por lo tanto se creó una base de datos, distribuida y jerárquica en la que los servidores se distribuyen principalmente en tres niveles:

- Raíz
- Top Level Domain
- Autorizativos.

Por ejemplo si queremos solicitar la dirección www.unaj.edu.ar lo que se hace es solicitarla al servidor Raíz, este devuelve la dirección del servidor TL Domain .ar, a este le pedimos la IP y nos devuelve la del servidor autorizativo unaj.edu.ar y a este servidor se la pedimos nuevamente, y nos devuelve la IP de la máquina www.unaj.edu.ar.

- Los DNS Raíz son 13 y están replicados y configurados en clusters por todo el mundo.
- Los DNS TLD : Son los responsables de los dominios de nivel superior y de países: .com, .org, .ar, etc.
- Los DNS autorizativos son los que tendrán los registros DNS de los host que están accesibles al público, como los servidores webs o el correo.
- Por último encontramos a los servidores DNS locales o servidores de nombres predeterminados, que son implementados por cada ISP y NO pertenecen a la jerarquía antes descripta, son a los que primero pedimos a una traducción IP y son los encargados de buscarla, por la jerarquía DNS. Tienen una caché por lo que de cada búsqueda que hacen, se quedan una copia, por lo que si se vuelve a pedir, no tienen que volver a buscarla. Los registros DNS almacenados en estas caches expiran al cabo de un tiempo y hay mecanimsmos de actualización y notificación para mantener la información al día.

Es habitual que la información sobre los servidores TLD esté cacheada en nuestro DNS local y no se hagan muchas consultas a los servidores raíz preguntando por los TLDs.

3.4. Enumero al menos 4 tipos de registros DNS, explicando cada uno para qué sirve

Los registros DNS o Research Récords serán los que se almacenen en los servidores DNS y no son más que una tupla con un nombre un valor un tipo y un tiempo de vida o TTL indicando el tiempo que se debe almacenar este registro en una caché.

Hay varios tipos de registros DNS pero los más comunes son los del tipo A en el que el campo nombre tendremos el nombre del host y en el campo valor su dirección IP. Los de tipo NS los cuales en el nombre estará el nombre de dominio y en el valor el nombre de un servidor DNS autoritativo para ese dominio. O los de tipo CNAME en el nombre está el alias para un host cuyo nombre canónico es el que figura en el campo valor.

El de tipo MX que es como un CNAME me pero para servidores de correo electrónico

Si un servidor es autoritativo de un host tendrá un registro tipo A para ese host indicando su IP, si no lo es lo podría tener en su caché o tener un registro DNS con el nombre del servidor DNS que tenga el dominio del host y otro registro tipo A con la IP de ese servidor.

Tenemos entonces que DNS es una base de datos distribuida que consultan los equipos empleando un protocolo. Veamos ahora cómo son los mensajes de ese protocolo DNS.

3.5. Tabla de Hash Distribuida (DHT). ¿Qué es y cómo funciona?

Las tablas de hash distribuidas, conocidas por las siglas DHT, (del inglés, Distributed Hash Tables) son un tipo de tablas de hash, almacenan pares de clave-valor y permiten consultar el valor asociado a una clave, en las que los datos se almacenan de forma distribuida en una serie de nodos (sistemas distribuidos) y proveen un servicio eficiente de búsqueda que permite encontrar el valor asociado a una clave. Para esto último usan un sistema de enrutado que permite encontrar de forma eficiente el nodo en el cual está almacenada la información que se necesita.

La responsabilidad de mantener el mapeo de las claves a los valores está distribuida entre los nodos, de forma que un cambio en el conjunto de participantes causa una cantidad mínima de interrupción. Esto permite que las DHTs puedan escalar a cantidades de nodos extremadamente grandes, y que puedan manejar constantes errores, llegadas y caídas de nodos.

Por ejemplo, el programa Bittorrent no utiliza un tracker centralizado sino que utiliza lo que denominamos tablas hash distribuidas, es decir, que son una especie de bases de datos, que van a contener pares clave-valor, y que en este caso van a contener el nombre del fichero y la dirección IP que tiene y así, el tracker va a poder saber que un fichero determinado, lo tiene el ordenador por ejemplo,210.4.7.2. Entonces los pares van a consultar a esta base de datos distribuida consuntándole por este esta clave y el valor e irá obteniendo los datos de cada torrente.

3.6. DHT. Explique cómo reacciona cuando quiere ingresar un peer y cuando se produce el abandono de un peer

Para realizar esto, se siguen unos pasos preliminares:

- Se deberá asignar los datos parejas (clave, valor) a los pares.
- Se asignará cada pareja al peer cuyo ID sea el más próximo a la clave.
- Por convención el peer más próximo es el inmediato sucesor de la clave.

Entonces, con lo anterior establecido, se tiene que cada peer sólo es consciente de su inmediato sucesor y predecesor (disposición llamada ¿red solapada¿)

Dentro de esta Red Solapada se tiene que:

- Cada par tiene conocimiento de la dirección IP de su sucesor, predecesor y algunos peers de atajo.
- Hay una reducción de 6 a 2 mensajes.
- Es posible que una DHT se puede diseñar con un número de vecinos por par, como número de consultas, del orden de O(log N)

Para el abandono de peers dentro de la red Solapada se deben tener las siguientes consideraciones:

- Gestionar el abandono de peers requiere que cada peer conozca las IP de su primer y segundo sucesor
- Cada par verificará periódicamente que sus dos sucesores están activos.

Entonces,por ejemplo:

- El par 5 abandona de repente la red.
- El par 4 lo detecta; 8 pasa a ser el inmediato sucesor; pregunta por el inmediato sucesor de 8; hace que el inmediato sucesor de 8 sea su segundo sucesor.

3.7. DHT circular con atajos. ¿Se puede tener una DHT totalmente conexa?

De tenerla, perdería eficiencia, ya que la disposición distribuida es lo que permite tanto la calidad de búsqueda como el tiempo que se tarda en ella. Si se tuviera una DHT circular completamente conexa sería como tener una base de datos secuencial, por lo que se perdería

4. Questionario Clase nº 5

4.1. ¿Qué es un socket?

Sockets, son interfaces por los cuales los procesos envían o reciben mensajes a o desde otro proceso. Cuando se construye un socket hay que especificar si la conexión será mediante el protocolo de transporte TCP o UDP.

La aplicación que utilizará el sockets, deberá de utilizar la API de sockets para enviar y recibir mensajes.

El servidor estará ejecutando el proceso que tendrá un socket para recibir y enviar mensajes por un determinado puerto. Este puerto junto con la dirección IP del servidor los debe conocer el cliente para poder enviarle mensajes desde su propio socket.

El servidor debe estar ejecutando el proceso antes de que el cliente envíe algo. Este también debe tener un socket a través del cual recibe y envía segmentos. Por su parte, el cliente necesita conocer la IP y el puerto del servidor. El socket se identifica localmente con un puerto.

Los lenguajes comúnmente utilizados son:

- C.
- C++.
- Java.
- Python.

Un flujo o stream es una cadena de caracteres que entran (flujo de entrada) por ejemplo desde el teclado; o salen (un flujo de salida) por ejemplo al monitor desde un proceso.

Un flujo de entrada está asociado a una fuente de entrada para el proceso, por ejemplo un socket o un teclado.

Un flujo de salida está asociado con una fuente de salida, por ejemplo un socket o monitor.

4.2. Explique ¿cómo es la programación de socket en TCP?

Se debe recordar que en las comunicaciones bajo TCP, es el cliente el que debe iniciar el contacto con el servidor, quien debe estar ejecutando un socket esperando ser contactado.

- El proceso servidor debe estar ejecutándose.
- El servidor debe tener un socket que permita el contracto de acogida.

El cliente contacta al servidor:

- Creando un Socket TCP
- Especificando la IP y el perto del proceso servidor.
- Acuerdo en 3 fases: transparente para procesos clientes y servidor.
- Cuando el cliente se contacta con el servidor, se crea un socket TCP para comunicarse con el cliente. Esto permite al servidor comunicarse con varios clientes, porque utilizarán distintos sockets.
- Una vez establecida la conexión, ambos disponen de una tubería donde podrán transferirse datos para leer y escribir. Esta transferencia es fiable y se garantiza que llega en orden.

4.3. Explique ¿cómo es la programación de socket en UDP?

Hay que recordar que en UDP no hay conexión entre cliente y servidor, por lo que:

- No hay una fase inicial de negociación.
- El emisor explícitamente asocia la IP y el puerto de destino a cada paquete. El servidor recibirá un datagrama UDP llamado también "paquete UDP") del que sacará la IP y el puerto del emisor para poder contestarle.
- Modelo de servicio no fiable en el que se hace el máximo esfuerzo por suministrar el lote de bits al destino.
- El servidor extrae la IP y el puerto del emisor del datagrama recibido.

5. Questionario Clase nº 9

5.1. ¿Cuales son las estrategias para el control de Congestion? ¿Cual usa TCP?

Se entiende por congestión, cuando se tienen muchas fuentes enviando muy rápido demasiado tráfico a se manejado por la red. Es importante *diferenciarlo* del Control de Flujo, donde se intenta en este último, *no saturar el buffer del receptor*.

Se tienen dos opciones para poder controlar la congestión en la red:

- Es aquella en donde la red no proporciona ninguna ayuda, y todo lo tienen que hacer los sistemas terminales de origen y destino. Por lo que se deberá de inferir la congestión por los terminales observando los paquetes que se pierden o se retrasan. Es el método aplicado por TCP.
- Otra opción es que los routers de la red nos ayuden, proporcionando ayuda a los terminales por ejemplo:
  - Mediante un bit que pueda indicar la congestión (SNA, DECbit, TCP/IP, ECN, ATM).
  - Velocidad de transmisión que el router es capaz de soportar.
  - Puede ser directa (choke packet, los cuales son los paquetes de congestión) o a través del receptor.

5.2. ¿Cuáles son los mecanismos de Control De Congestión?

El objetivo en el control de la congestión es intentar transmitir a la máxima velocidad posible, sin congestionar la red, para perder el menor número de paquetes posible y que no haya que retransmitir. Se debe encontrar la tasa justo por debajo del nivel de congestión. Hay que hacerlo de forma descentralizada, lo que significa que cada emisor elige su propia tasa basándose en una realimentación implícita (basándose en la información que recibe mediante los ACK):

- ACK segmento recibido (buena noticia) -¿red no congestionada -¿se puede aumentar la tasa de envío.
- Segmento perdido: se asume que la perdida es debida a la congestión de la red -¿decrecer la tasa de envío.

TCP utiliza varios mecanismos para este control de congestión, alguno de ellos son:

- Arranque Lento: el cual permite obtener de forma rápida el umbral de saturación de la red, aumentando la velocidad de forma exponencial.
- Evitación de la congestión: Trata de acercarse a la congestión pero de una forma más lenta que el Arranque lento.
- Recuperación rápida (implementado en versiones posteriores de TCP, como la RENO): Esta fase diferencia la pérdida de paquetes por fin de temporización (lo que indica que la red está muy mal y que en este caso siempre pasaremos a Arranque Lento disminuyendo la velocidad al mínimo). Esto diferencia de la pérdida por Triple ACK duplicado, lo que indica que la red no está tan mal, porque llegan ACK por lo tanto se puede disminuir la velocidad menos, en concreto, a la mitad.

5.3. ¿Cuál es la diferencia entre TCP Tahoe y TCP RENO con respecto al manejo de la congestión?

TCP ha ido mejorando con el tiempo, por lo que la versión *TCP RENO* sumó una nueva fase, denominada *Recuperación Rápida*. Esta fase diferencia la pérdida de paquetes por fin de temporización (lo que indica que la red está muy mal y que en este caso siempre pasaremos a Arranque Lento disminuyendo la velocidad al mínimo). Esto diferencia de la pérdida por Triple ACK duplicado, lo que indica que la red no está tan mal, porque llegan ACK por lo tanto se puede disminuir la velocidad menos, en concreto, a la mitad.

Es lo que implica esta nueva fase de recuperación rápida:

- Si estamos en Evitación de la Congestión, se detecta 3 ACK duplicado o estamos en Arranque Lento y detectamos 3 ACK duplicado se pasa a la fase de *Recuperación Rápida* disminuyendo la velocidad a la mitad.
- Si recibimos un ACK nuevo, pasamos nuevamente a *Evitación de la congestión* aumentando la velocidad de una forma lineal.

La diferencia entre Taho y RENO es que con la primera se baja la velocidad al mínimo, en cambio con RENO se parte desde el umbral (entrando en la fase de Evitación de la Congestión) por lo que se recupera más rápido de la pérdida de paquetes.

5.4. ¿Qué es la Equidad en TCP? ¿En UDP se peude dar?

El objetivo de la equidad un mecanismo de control es equitativo si cuando N sesiones TCP compartiendo el mismo enlace, el cual tiene un ancho de banda R, y cada una de esas sesiones tiene una velocidad media de conexión de R/N (se distribuye el ancho de banda disponible de forma equitativa entre las N sesiones).

¿Por qué se dice que TCP es equitativo?

Se dice que un mecanismo de control de congestión es equitativo si la velocidad media de transmisión de cada conexión es aproximadamente igual a R/N; es decir, cada conexión obtiene la misma cuota del ancho de banda del enlace. Dado los métodos que emplea TCP para poder gestionar la pérdida de paquetes (vistos anteriormente), la congestión de los enlaces también se ve beneficiada en tanto que TCP permite que dos conexiones que compiten entre sí por el ancho de banda, puedan transmitir tendiendo a emplear el mismo ancho de banda.

Equidad en TCP

Las aplicaciones multimedia, a menudo utilizan UDP para transportar datos de audio y video. Esto es así ya que no se quiere que la velocidad de transferencia quede limitado por el control de congestión que establece TCP. Se opta por UDP para que la entrega de datos multimedia sea a una velocidad constante ya que dadas las características del streaming es tolerante a la pérdida de paquetes.

El problema que se tiene a nivel red, es que dado que la velocidad aumenta para UDP, por no tener mecanismos de control de congestión, el tráfico UDP podría llegar a expulsar al tráfico TCP. Como no se controla la velocidad de emisión, un emisor UDP dadas las ráfagas constantes de paquetes, podría llegar a .ªdueñarse"del canal de comunicación, quedando reducido el tráfico TCP a un mínimo. Para este problema, hay varias investigaciones llevadas a cabo que tratan de solucionar esto.

6. Cuestionario clase nº 9

6.1. ¿Cuales son las estrategias para el control de Congestion? ¿Cual usa TCP?

Las estrategias son: buscamos no saturar los bufers de los router que hay en el camino. la saturación viene dada por múltiples host enviando flujos de datos demasiado rapido a la red. Se encolan paquetes en los buffers, y se generan perdidas (overflow en los routers.)

Hay dos tipos de estrategias:

1. Terminal a terminal: los terminales determinan la congestion, observando que los paquetes se pierden o se retrasan. Es lo que aplica TCP.
2. El núcleo de la red (routers) retroalimentan a los terminales, con un bit indicando si hay congestión. indicando que velocidad soporta, o mediante paquetes de congestión que pueden enviar a los terminales, o a través del receptor cuando le le envía los datos al terminar transmisor.

6.2. ¿Cuáles son los mecanismos de control de congestión?

El objetivo del control de congestión es transmitir a la máxima velocidad posible sin congestionar la red. Para perder el menor numero de paquetes posibles y asi evitar las retransmisiones.

Los mecanismos son: Encontrar la tasa por debajo del nivel de congestión, hay que hacerlo de forma descentralizada, cada emisor tomara las decisiones basándose en la información que recibe con los akcs. Si recibo acks es que todo va bien y puedo aumentar la velocidad.

Con los segmentos perdidos, disminuiremos la velocidad.

Probando el ancho de banda disponible, enviaremos aumentando la velocidad hasta perder paquetes, bajaremos y comenzaremos nuevamente el proceso aumentando y bajando.

Variación de velocidad de transmisión: Podremos variar, mediante la ventana de emisión, se le llama ventana de congestión. Si aumento el numero de paquetes que enviamos sin esperar el reconocimiento aumentaremos la velocidad.

El emisor tiene que mirar la ventana de congestión y la ventana de recepción. Y estará limitado por la menor de ellas.

Perdidas de paquetes: se reduce la ventana de congestión por fin de temporizador (sin rta del receptor) o porque recibamos 3 acks duplicados (retransmisión rápida) se reduce la ventana a la mitad. ACK recibido: Aumentar la ventana de congestión

Fase de arranque lento: aumentar exponencialmente al inicio de la conexión o tras un fin de temporización. Para poder encontrar el punto de saturación de la red rápidamente. Se aumenta en un segmento la ventana por cada ack recibido correctamente.

Cuando detectemos un perdida de paquete ya sea por fin de temporización o triple ACKS duplicado, se establecerá un valor de $umbral = \frac{ventanadecongestion}{2}$ para evitar de llegar a esa velocidad para que no sature la red. A Este valor se lo multiplicara por 2. Cuando lleguemos de nuevo a la saturación en ves de volver a aplicar el mismo método, realizaremos la evitación de la congestión.

Evitación de la congestión: Aumentar la velocidad linealmente, para acercarnos al limite mas gradualmente. En ambos casos si detectamos una pérdida de paquetes volvemos a la fase de arranque lento. Y disminuimos la velocidad al mínimo; intentamos acércanos a al limite de congestión de forma mas lenta. Aumentando en la ventana de congestión en un segmento por ventana.

6.3. ¿Cual es la diferencia entre TCP tahoe y TCP reno con respecto al manejo de la congestion?

6.3.1. TCP Tahoe

Inicia la transmisión en arranque lento, aumentando exponencialmente la velocidad, y ante la perdida de paquete por fin de temporizador o triple ACKS duplicado, el umbral de la ventana de congestion lo ponemos a la mitad, reducimos la velocidad al mínimo, y arrancamos la transmisión en arranque lento hasta el nuevo umbral, y luego continuamos con evitación de la congestion aumentando la velocidad linealmente hasta encontrar nuevamente perdida de paquetes por ACKS duplicados volvemos a repetir el proceso.

6.3.2. TCP Reno

Reno incluye la recuperación rapida, para difertenciar la perdida de paquetes por temporización que indica que la red esta mal, y en ese caso iremos a arranque lento. Disminuyendo la velocidad al mínimo. Esto se diferencia de la perdida por triple ack duplicado indicando que la red no esta tan mal, porque están llegando paquetes y por lo tanto podemos bajar la velocidad a la mitad y no al mínimo pasando a la recuperación rapida. Y al recibir un ACK nuevo pasamos a la evitacaion de la congestion aumentando la velocidad linealmente.

En resumen TCP reno se recupera antes que Tahoe de la perdida de paquetes si esta se ha detectado por un triple ACK duplicado, reno baja a la mitad la velocidad y Tahoe al mínimo, y continua con evitación de la congestión

6.4. ¿Que es Equidad en TCP, en UDP se puede dar la equidad?

Equidad es dividir la capcidad de transmicion en los routers de la red entre los terminales que estan transmitiendo.udp no tiene control de congestion con lo cual no se podria.