2. Network Application Architectures

Las aplicaciones de red son **la razon de ser** de una red de computadoras

La arquitectura de red es fija y proporcional un conjunto de servicios a las aplicaciones.
El desarrollador diseña la arquitectura de la aplicacion y establece cómo debe estructurarse la aplicacion en los sistemas terminales.

Hoy en día hay dos arquitecturas predominantes: **cliente-servidor** y **P2P**(peer-to-peer)

Client-Server:

- siempre hay un host(servidor) activo que atiende las solicititudes de muchos otros hosts llamads clientes.
- Los clientes no se comunican directamente entre ellos.
- El servidor tiene una direccion fija y conocida llamada **direccion IP** en la cual un cliente siempre puede contactar con el servidor mandando un paquete a la direccion IP del servidor.
- Si se utiliza un unico servidor es muy probable que sea incapaz de tratar con todas las solicitudes de los clientes a la vez(se desborda la aplicación). Para ello se utiliza un DATA CENTER (tiene una gran cantidad de host) usado para crear un servidor virtual poderosos.

P2P:

- Existe una minima o nula dependencia de los servidores dedicados a data center.
- Comunicacion directa entre los pares conectados intermitentemente
- Los pares no son propiedad del provedor de servicio sino que son controlados por computadoras de escritorio y portatiles (controladas por los usuarios)
- La arquitectura se llama peer-to-peer porque los pares se comunican entre ellos sin tener q pasar por un servidor-
- Es autoescalable y es economica ya que no se requiere de una arquitectura de servidores
- Enfrentan desafíos de seguridad, rendimiento y confibilidad debido a su estrcutura altamente desentralizada

2.1.2 Comunicacion entre procesos

Los programas NO se comunican sino que son los **procesos.**

Si ambos procesos pertenecen a la misma máquina el S.O se encarga de su comunicacion mientras que si son procesos de distintos sistemas terminales entonces estos se comunican intercambiando **mensajes**.

Procesos cliente servidor

Una aplicacion de red cuenta con parejas de procesos que se envían mensajes entre sí por medio de una red.

En general se denomina a uno de estos procesos como **cliente** y a otro como **servidor**. Ejemplo: en una web un navegador es un proceso cliente y el servidor web es un proceso servidor. En P2P un proceso puede ser tanto cliente como servidor.

Interfaz entre el proceso y la red de computadora:

Un mensaje es enviado y recibido a traves de una interfaz de software llamada socket.
Un socket es la interfaz entre la capa de aplicacion y la capa de transporte de un host. (API)
Del lado de la aplicacion el desarrollador tiene el control sobre todos los elementos pero del lado de transporte tiene muy poco control. El unico poder de control que tiene es la eleccion del protocolo de transporte y el tamaño de los buffers

Direccionamiento de procesos

Para que un proceso pueda mandar un mensaje a otro proceso , el proceso receptor tiene que tener una **direccion IP** y también debe tener un identificador (**puerto**) que especifica que es el host destinatario.

2.1.3 Servicio de Transporte disponible

La aplicacion del lado emisor envía mensajes a traves del socket. El protocolo de la capa de transporte tiene la responsabilidad de llevar los mensajes hasta el socket del proceso receptor.

hay distintos servicios que puede ofrece el protocolo de la capa de transporte:

- Transferencia de datos confiable: Si garantiza que los datos enviados llegan al proceso receptor sin errores. Sin erroresimplica que llegan, en orden y no están corrompidos.

- Tasa de transferencia(Thoughput): Durante el tiempo el thoughtput puede ir variando dado que otras sesiones compartirarn el ancho de banda. Una aplicación podría pedir una garantia minima de thoughput de r buts/seg y si la capa de transporte ofrece este servicio le asegurara que esa sea la tasa de transferencia.

- Temporizador(Timing): un protocolo de transporte garantiza un timing, es decir, que cada cierto tiempo se cumpla la recepcion de un bit

- Seguridad: proveer confidencialidad entre dos procesos.Ej: encriptar los datos. También puede ofrecer mecanismos para garantizar la integridad de los datos y mecanimisto de autentificacion en el punto terminal.

2.1.4 Servicio de transporte proporcionados por internet

El internet permite utilizar dos procolos de transporte: UDP y TCP.

TCP

Los servicios que incluye son:

- Servicio orientado a la conexion:
    - en tcp el cliente y el servidor intercamabin informacion de control en la capa de transporte antes de que empiecen a mandar los mensajes de la aplicacion.
    - Este proceso se denomina handshaking: es una etapa de 3 fases
    - Luego de esta fase, se dice que se establece una conexion TCP entre los sockets de ambos procesos.
    - La conexion puede ser full-duplex: dos procesos pueden enviar mensajes entresi al mismo tiempo.
    - Cuando uno de los dos termina se cierra la conexion.
- Transferencia de datos fiabñe:
    - Los procesos pueden confiar que la informacion va a ser enviada, no va tener errores y va a llegar en el orden correcto.

Tcp tambien incluye un mecanismo de control de congestion donde TCP regula el proceso emisor cuando la red está congestionada

SSL: mejora de TCP donde provee servicios críticos de seguridad como lo es el cifrado, la integridad de los datos y autentificacion en el punto terminal.

UDP

No está orientado a la conexion.
No provee un servicio de transferencia de datos fiable
No incluye un mecanismo de control de congestion.

Ninguno de los dos protocolos puede garanzar los servicios de sincrinizacion y rendimiento.

2.1.5 Protocolos de la capa de aplicacion

Un protocolo de la capa de aplicacion define **como** los procesos de una aplicacion ejecutandose en diferentes sistemas terminales pueden pasar mensajes entre sí
Define:

- Tipos de mensajes intercambiados
- La sitaxis de los mensajes
- La semantica de los campos
- Las reglas que determinan cuándo y cómo un proceso envía y responde mensajes.

2.2 La Web y HTTP

Es el protocolo de la capa de aplicacion de la web

Se implementa mediante dos programas: programa cliente programa servidor
Ambos programas se ejecutan en sistemas terminales y se comunican entre sí intercambiando mensajes HTTP.
Http define la estructura de estos mensajes y cómo el cliente y el servidor intercambian los mensajes

Cuando un usuario realiza una peticion a una web page, el brower envía una HTTP REQUEST al servidor y este responde con una HTTP RESPONSE que contiene los objetos (un arquivo que puede direccionarse mediante un unico URL)

HTTP utiliza TCP como su protocolo de transporte. El cliente HTTP inicia una conexion TCP con el servidor. Una vez establecida la conexion, los procesos de Navegador y de servidor acceden a tcp a traves de sus interfaces de sockets.

HTTP no tiene que preocuparse de la pérdida de datos ya que TCP le garantiza que van a llegar bien.

El servidor envía los archivos solicitados a los clientes sin guardar los estados de estos, es por esto que se lo denomina **stateless protocol**

2.2.2 Conexiones persistentes y no persistentes

Cuando hay interacciones cliente-servidor debe definirse si las request/respuesta van en la misma conexion TCP (persistent connections) o en distintas (non-persistent connections). Por default HTTP usa conexiones persistentes. En conexiones no persistentes puede ocurrir cierto paralelismo.

- No persistentes: cada conexion tcp transporta exactamente una request y una response
- Persistentes: Una conexion es establecida y por esa es donde circulan los objetos y luego es terminada despues de cierto periodo de tiempo

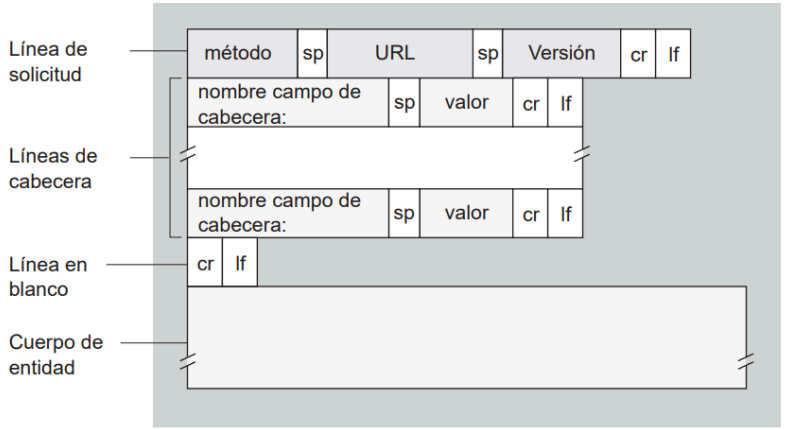
2.2.3 Formato de los mensajes HTTP

Existen dos tipos de mensajes HTTP: de solicitud(request) y de respuesta(response)
Ejemplo:

HTTP formato del mensaje

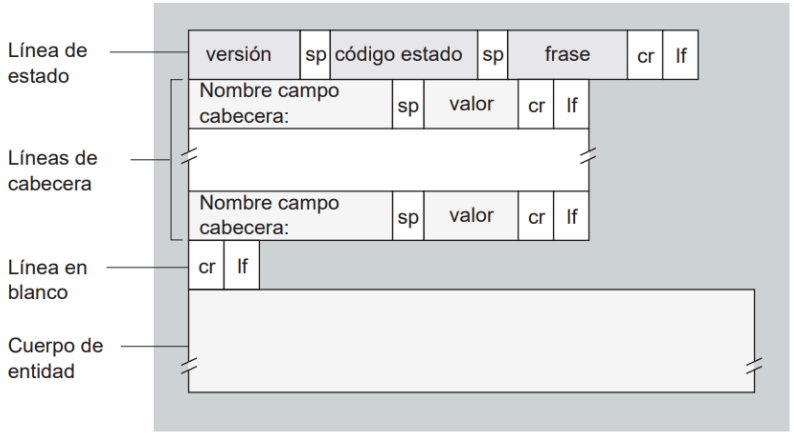
Hay dos tipo de formato:

- Request: primera línea es la request line, tiene 3 campos: metodo (GET, POST, HEAD, PUT, DELETE), URL y version de HTTP. Las líneas que siguen son header lines. La de Host especifica el host donde reside el objeto, en connection va el tipo de conexion, en user-agent va el browser y en accept-language va el lenguaje que



El campo entity va lleno cuando se usa el método POST enviando info. También puede ir en la URL con un método GET. El HEAD se utiliza para debugging. PUT es para subir objetos a servidores web

- Respuesta: tiene una **status line** (3 campos: version del protocolo, código de status y el mensaje de status), seis header lines y un **entity body** (contiene lo que se pidió en el request). Las header lines: date tiene hora y día en que el servidor envía data, Server indica que servidor generó el mensaje, Last modified indica el momento de creacion o última modificacion del objeto, Content-Lenght indica el largo del body y Content-type el tipo del body (HTML por ej). El status code puede ser varios: 200 es OK, 301 es que el objecto fue movido, 400 es bad request, 404 no encontrado, etc



Las conexiones No persistentes presentan algunos inconvenientes en primer lugar tiene que establecerse y gestionarse una conexión completamente nueva para cada objeto solicitado para cada una de estas conexiones deben accionarse buffers tcp y tienen que gestionarse variables tcp tanto en el cliente con el servidor esto puede sobrecargar de forma significativa al servidor web ya que puede estar sirviendo solicitudes de cientos de clientes extintos simultáneamente además cada objeto sufre un retardo de entrega de dos rtt un rtt para establecerse la conexión tcp y otra red para solicitar y recibir un objeto

2.2.4 Interaccion usuario-servidor: **COOKIES**

HTTP es stateless pero usa **cookies para trackear a los usuarios**.Esta tecnologia tiene 4 componentes:

- Cookie Header line en respuesta de HTTP (Set-Cookie:)
- Cookie Header line en request de HTTP (Cookie:)
- archivo cookie guardado en el end system y manejado por el browser

- base de datos en la página de internet.

Cómo funciona:
Cuando una request llega al server, el server crea un número id y crea una entrada en su backend database que será indexada por ese id.
Luego responde al browser incluyendo en el mensaje de http response un set cookie = id
Cuando el browser recibe esa respuesta y ve el header con el set cookie agrega una línea al archivo especial de cookies que incluye el host de servidor y el número de identificacion.

Entonces cada vez que ese proceso cliente manda una request al servidor, esté consulta su archivo de cookies y extrae su número de identificación para ese sitio y lo añana a la solicitud http en la cabecera. y Así el servidor puede seguir la actividad del proceso.

2.2.5 Web Caching:

El **caché de la web** (proxy server) guarda copias de objetos recientemente pedidos. Antes de establecer TCP con servidor, se establece con el Web Cache y se ve si tiene el objeto deseado. Si lo tiene, se envía en una respuesta HTTP. Sino, la Web Cache establece una conexion TCP con el servidor, envía una request y al recibir una respuesta la envía al usuario, previamente almacenando una copia.

Habitualmente es un isp quien adquiere e instala una caché web

El Web cache hace la conexion más rapida y reduce el tráfico hacia el servidor. Gracias al uso de las redes de distribución de contenido - Content Distribution Networks (CDNs). Una compañía CDN instala caches a lo largo de todo internet Localizando la mayor parte del tráfico.

Aparece un problema, el objecto en la caché **puede estar desactualizado**. Aparece el **GET condicional**, que agrega campo If-Modified-Since. El caché envía al server un request con este campo, indicando al servidor que envíe un nuevo objecto si la fecha en que fue modificado es distinta a la del request de la caché (Si el objeto no ha sido modificado el servidor web envía un mensaje de respuesta a la caché con 304 not modified status code pero no incluye el objeto solicitado en el mismo)

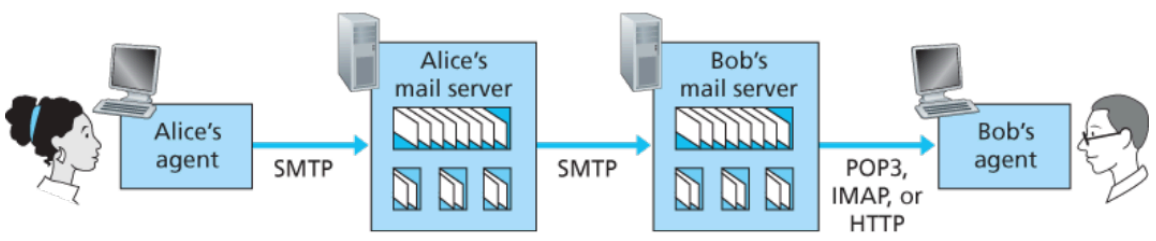
2.3 Correo en internet

El correo electronico es un medio de transporte asincronico. Sus componentes principales son:
- User agents: permite a los usuarios leer, responder, enviar, guardar y escribir mensajes (Outlook)
- Mail servers: Aquí se envían los mensajes, donde son colocados en una cola de salida. Cuando un usuario quiere leer, el agente lo recupera de este servidor. Aquí están las mailbox.
- Simple Mail Transfer Protocol (SMTP)

2.3.1 SMTP

Es el principal protocolo de la capa de aplicaciones para mails. Usa reliable data transfer de TCP.

SMTP transfiere mensajes desde los servidores de correo de los emisores a los servidores de correo de los destinatarios.
Restringe el cuerpo de todos los mensajes a formate ASCII de 7 bits.



1. Alicia invoca a su agente de usuario para correo electrónico, proporciona la dirección de correo electrónico de Benito (por ejemplo, benito@unaescuela.edu), compone un mensaje e indica al agente de usuario que lo envíe.
2. El agente de usuario de Alicia envía el mensaje al servidor de correo de ella, donde es colocado en una cola de mensajes.
3. El lado del cliente de SMTP, que se ejecuta en el servidor de correo de Alicia, ve el mensaje en la cola de mensajes. Abre una conexión TCP con un servidor SMTP, que se ejecuta en el servidor de correo de Benito.
4. Después de la fase de negociación inicial de SMTP, el cliente SMTP envía el mensaje de Alicia a través de la conexión TCP.
5. En el servidor de correo de Benito, el lado del servidor de SMTP recibe el mensaje. El servidor de correo de Benito coloca entonces el mensaje en el buzón de Benito.
6. Benito invoca a su agente de usuario para leer el mensaje cuando le apetezca.

SMTP tiene un handshake donde se indica las direcciones de email de ambos. Comandos del cliente: HELO, MAIL FROM, RCPT TO, DATA, QUIT, CRLF.CRLF (linea vacia). telnet sirve para establecer conexion TCP entre localhost y mail server.

2.3.2 SMTP vs HTTP

Similitudes:

- Ambos protocolos se emplean para **transferir archivos** de un host a otro: HTTP tranfiere objetos desde un web server a un web client mientras que SMTP tranfiere un mail desde un mail server hacia otro mail server
- Ambos utilizan **conexiones persistentes**.

Diferencias:

- **HTTP es un pull protocol**, alguien sube data a la web para que varios pulleen del server cuando quieran. Por el contrario, **SMTP** es más bién **push protocol**, donde el que envía el mensaje lo pushea al mail server del receptor.
- SMTP restringe los mensajes a ASCIIs de 7 bits, HTTP no tiene restricción.
- HTTP encapsula cada objeto en su mensaje de respuesta, SMTP envía todos en un mensaje.

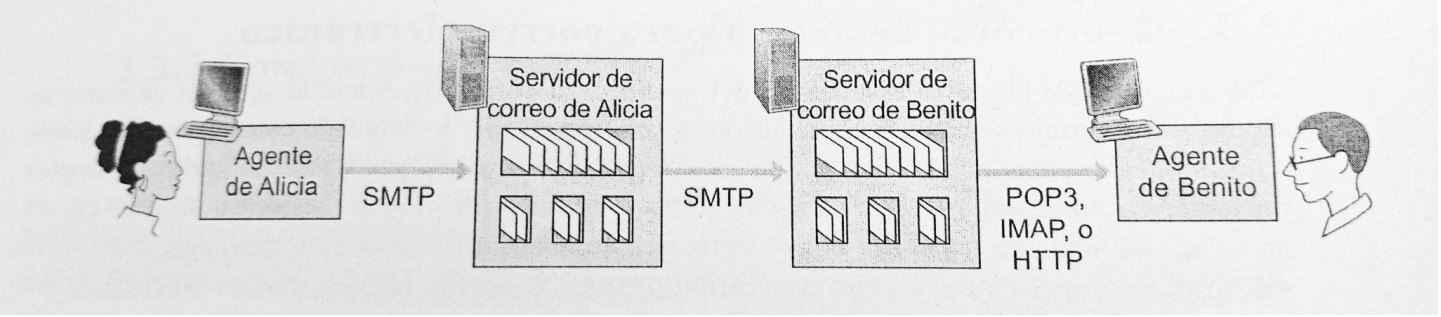
2.3.4 Protocolos de acceso a mail

La idea de hacer que el agente de usuario del emisor tenga que comunicarse primero con su **propio servidor** es debido a que este puede comunicarse con el servidor del receptor e intentar reenviar el mensaje cada cierto tiempo por un periodo de tiempo hasta el arribo o descarte del mismo.

Sin este paso, el mensaje podría no ser enviado, o la pc del emisor tendría que estar siempre encendida para poder ser un servidor en correcto funcionamiento.

Es por esto que SMTP utiliza el servidor del emisor y el servidor del receptor como receptores del envío original para evitar ese problema.

Para que el receptor luego reciba sus mensajes con una operación de PULL, existen protocolos que transfieren mensajes del servidor a la PC local.



POP3

Es un protocolo de acceso a correo simple y su funcionalidad es limitada.
El agente de usuario establece una conexion TCP con el servidor del correo.
una vez establecida la conexion TCP tiene 3 fases:

- En authorization: envía contraseña y usuario (user pass <contraseña>)
- En transaction: recupera el mensaje y otros comandos (list, retr, dele, quit).
- En update: termina la sesión POP3 y el servidor elimina los emails marcados para este fin. POP3 no guarda información entre sesiones.

IMAP

Es más complejo que pop3, permite asociar mensajes a una carpeta y luego moverlo a otra. También permite buscar mensajes segun diferentes criterios.
Este protocolo asocia cada mensaje con una carpeta (INBOX cuando llega el mensaje, luego puede moverse a otra).
También mantiene informacion del usaurio entre sesiones y permite obtener porciones de mensajes (solo el header por ej). Se le puede pedir partes del mensaje, es muy uitl si hay bajo ancho de banda.

Correo electronico web

El mail server se comunica con el mailbox via http aunq de un mail server a otro se siguen comunicnado por smtp

2.4 DNS (es un servicio)

Para identificar un host se pueden usar **hostnames**, pero estos nombres no permiten conocer su locacion en el internet. Entonces se los **identifica con las direcciones IP** (4 bytes), separados por "." donde cada byte en decimal va de 0 a 255. Es jerárquica xq si la miramos de izquierda a derecha podemos obtener mas y mas informacion de donde se encuentra el host en la internet.

Servicios proporcionados por DNS

El DNS es un **servicio** que traduce de hostname a IP. DNS es:

- una base de datos distribuida implementada en una jerarquia de servidores DNS
- un protocolo de la capa de aplicacion que permite a los hosts realizar consultas a la base de datos.

DNS corre sobre UDP.
DNS es usada por otros protocolos como HTTP y SMTP para traducir los nombres de host suministrados por el usuario en direccion IP.

Servicios que provee DNS:

- **Host aliasing**: un host puede tener más de un nombre o alias. El más completo es el canonico. Una aplicación puede invocar DNS para obtener el nombre de host canónico para un determinado alias, así como la dirección IP del host.
- **mail server aliasing**: DNS puede ser invocado por aplicaciones de mail para obtener el hostname canonico para un determinado alias.
- **load distribution**: permite asociar varias IPs a un nombre canónico para distribuir la carga en varios servidores que repliquen lo mismo

2.4.2 Cómo funciona DNS

Una base de datos jerárquica y distribuida

DNS utiliza un gran número de servidores, organizados de forma jerárquica y distribuidos alrededor de todo el mundo.

Distribuido: Ningún servidor DNS dispone de todas las correspondencias de todos los hosts de Internet. En su lugar, las correspondencias están repartidas por los servidores DNS

Jerarquico: Existen 3 clases de servidores.

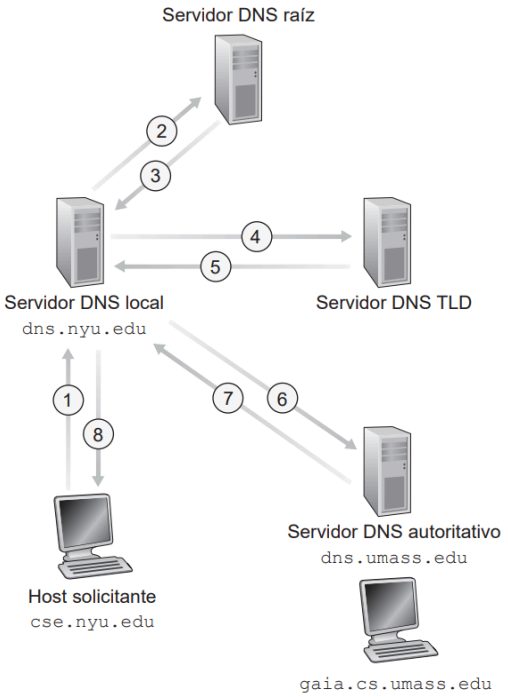
- servidores DNS raiz: proveen la IP de los servidores TLD. (hay aprox 400)
- Top-level domain (TLD) DNS servers: Para todos los dominios de nivel superior (com, net, edu, org) hay un servidor TLD. Devuelven la IP de los servidores DNS Autoritativos
- Authoritative DNS servers: mapea el nombre de los hosts a su IP.

Es necesario que sea distribuida y gerarquica para: 1. evitar un unico punto de fallo 2. Disminuir el volumen del trafico 3. Estar cerca de los hosts 4.Hacer mas facil elmantenimiento. Si fuera centralizada No podria escalar.

Existe otro tipo de DNS denominado servidores DNS locales, no pertenece de manera estricta a la jerarquia pero es central para la arquitectura DNS.
Cada ISP tiene un **servidor DNS loca**l. Este se encuentra cerca del host. Cuando un host realiza una consulta DNS, esta se envia al servidor DNS local que actua como proxy reenviando la consulta a la jerarquia de servidores DNS.

Ejemplo:

- El host envía un mensaje de consulta al servidor DNS local conteniendo el nombre del host (www.fi.uba.ar) que debe traducirse.
- El DNS local reenvia la consulta a un servidor raiz, que devuelve direcciones IP para serividores TLD con cierto dominio (ar).
- Luego el DNS local contacta uno de estos servidores TLD, donde el servidor TLD toma el sufijo uba y devuelve direccion IP de un servidor authorative para cierta direccion (uba.ar).
- Finalmente el DNS local contacta al servidor authorative de la página (fi.uba.ar) que devuelve la IP del host requerido (www.fi.uba.ar).



Las querys son recursivas e iterativas. Recursivas piden que se obtenga el mapeo en su nombre, iterativas porque responden inmediatamente al servidor DNS local. Las querys de DNS pueden ser recursivas e iterativas.

DNS Caching

Para mejorar el delay del sistema DNS y disminuir los mensajes DNS, El servidor local DNS cachea el par hostname/IP
Cuando un servidor DNS recibe una respuesta DNS (ejemplo contenga un correspondencia entre hostname y direccion IP) puede almacenar esta información en su memoria local.

Un servidor DNS local también puede almacenar en caché las direcciones IP de los servidores TLD para así ellos pueden salterase la consulta a los DNS raiz.

Responder: ¿De qué forma una CDN puede utilizar DNS para acercar el contenido a los usuarios?

La idea es que haya múltiples servidores CDN distribuidos por el mundo y que dependiendo la ubicación de los request de los usuarios DNS devuelve la IP del CDN más cercano del usuario, de esta forma beneficiando al usuario de tener menos latencia y acceso al contenido más rápido y también como un beneficio global la red se ve menos saturada ya que los paquetes deben hacer un recorrido menor.

Una CDN tiene múltiples servidores distribuidos por todo el mundo, tal que intenta redirigir las solicitudes de cada cliente al CDN que le provea la mejor experiencia.
Muchas CDNs aprovechan DNS para redirigir solicitudes. Por ejemplo: un proveedor NetCinema distribuye sus videos mediante KingCDN, en el sitio de NetCinema las urls tienen la forma http://video.netcinema.com/6Y7B23V. Lo que sucede:

1. El usuario ingresa a la url y el host envia la solicitud DNS para video.netcinema.com
2. El DNS local del usuario envía la solicitud al servidor de mayor jerarquía de NetCinema. Este último, en vez de devolver la IP, devuelve otro hostname del dominio de KingCDN, podría ser por ej. a1105.kingcdn.com
3. Ahora el DNS local envía una segunda query pero esta vez al servidor DNS de KingCDN y este sí devuelve la IP.

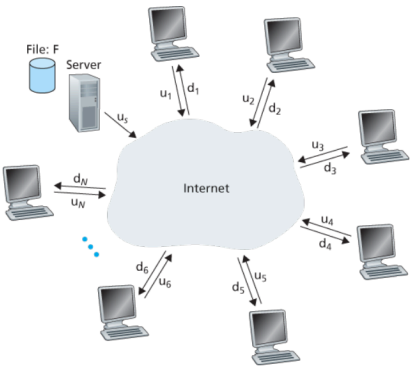
Mensajes DNS

Los servidores DNS almacenan resource records (RRs) incluyendo rrs que proveen correspondencia entre hostname y direccion IP. Cada respuesta DNS lleva uno o más de estos: (Name, Value, Type, TTL).

TTL: time to live, cuando borrarlo de la caché.
Tipo: si es A, name es hostname y value es IP. Si es NS, name es domain y value es hostname de authorative DNS server. Si es CNAME, value es el hostname canonico y name el alias. Si es MX, vale es nombre canonico del servidor de mail con alias en name

Insertar un registro en la base de datos DNS
Registrador es una entidad comercial que verifica la unicidad del nombre del dominimo, lo añade a la base de datos DNS y te cobra una cuota por el trabajo.
Cuando se registra un dominio se debe proveer también un registro con las ips de los servidores autoritativos, primarios y secundarios.

2.5 Distribucion de archivos P2P



La distribución de un archivo de gran tamaño desde un único servidor a muchos otros hosts: P2P.

En la distribución de archivos P2P, cada par puede redistribuir cualquier parte del archivo que ha recibido a cualesquiera otros pares, ayudando de este modo al servidor a llevar a cabo el proceso de distribución.

Con una arquitectura P2P, los pares de host conectados se comunican entre sí. Estos pares no son propiedad del proveedor de servicios sino que son computadoras de escritorio controladas por los usuarios.

Cada peer puede redistribuir una parte de un archivo grande. El tiempo de distribuicion es el tiempo que le lleva al archivo llegar a todos. En una cliente servidor va a ser el que le lleve al cliente con la menor tasa de descarga o lo que le lleve al servidor si tiene menor tasa de envío. Sube linealmente con la cantidad de clientes. En P2P, es más complicado de calcular. Primero el servidor debe enviar aunque sea 1 vez todos los bits del archivo. La tasa de subida es la del servidor más la de los peers

Bittorrent

Es un protocolo P2P para la distribucion de archivos. Los peers de un torrent descargar fragmentos del mismo tamaño del archivo de uno a otro. Cuando un par se une por primera vez al torrent, no tiene fragmentos del archivo. A lo largo del tiempo va acumulando cada vez más fragmentos.
A la vez que descarga fragmentos, actualiza fragmentos en otros pares. Una vez que un par ha adquirido el archivo completo, puede abandonar el torrent o permanecer en el mismo y continuar suministrando fragmentos a otros pares.

Cada torrent tiene un nodo llamado **tracker**. Cuando un peer se suma al torrent, se registra con el tracker, informando periodicamente si sigue en el torrent o no. Al sumarse a un torrent, **se le envían los IPs de los peers** (no todos) al recien llegado. Se intentan generar conexiones TCP con todos. De estos peces es de quien se **obtienen las porciones del archivo**. Primero va a pedir de los que no tiene, los que menos se repitan entre los vecinos (rarest first). Para enviar porciones de archivo, se prioriza a los que estén suministrando datos a la tasa más alta. Se elige a los 4 peers (Unchoked) que más rapido están suministrando data y se les envía. Esto se calcula cada 10 segundos. Además cada 30 segundos se elige un peer al azar (optmistically unchoked).

Flujos de video y distribucion de contenido

Cuanto más alto el bitrate, mejor será la calidad del video. 4K precisa aprox 10Mbps. Se crean varias compresiones y se envían a distintas tasas. El usuario elige cuál ver en base a su ancho de banda.

En HTTP streaming el vídeo se guarda en un servidor HTTP. En el lado del cliente, una vez que se pasa un umbral en el buffer de recepcion se empieza a reproducir el video. La contra es que todos los clientes reciben los mismo auqnue tengan distinto ancho de banda. Surge el Dynamic Adaptive Streamig over HTTP (DASH), el vídeo se encodea en distintas versiones cada una con un bitrate distinto (distinta calidad. El cliente solicita porciones del video dinámicamente (cuando el ancho de banda es alto solicita mejor calidad y viceversa). Cada version del video se guarda en el server HTTP con una URL distinta. En el server HTTP hay un manifest file con las URL a estas versiones.

Content Distribution Network (CDN)

Desventajas de proporcionar servicios de flujo de videos centralizados lo primero es que almacenar todos los videos en un centro de datos Y enviar reflejos directamente del centro de datos a clientes repartidos por todo el mundo Sí el cliente está lejos de centro de datos va a tener Que atravesar muchos isps Y si uno de esos enlaces proporciona una tasa de transferencia inferior a la velocidad a la que se consume el video la tasa de transferencia extremo extremo también será inferior a la tasa de consumo lo que provocará delays para el cliente Segundo si un video es muy popular probablemente se ha enviado muchas veces a través de los mismos de comunicaciones entonces Esto hace que se desperdice el ancho de banda de red y además la empresa estará pagando a su ISP proveedor por enviar los mismos nytes una y otra vez a internet además como tercer problema Tener un único centro de datos representa un único punto de fallo
Es por esto que para distribuir cantidades Masivas de datos Surgen las cdn s siendo estos servidores Situados en múltiples Ubicaciones geográficamente distribuidas, almacenando copias de los videos en sus servidores y tratando de dirigir cada solicitud de usuario a una ubicación de la cdn que proporcione una mejor experiencia de usuario
Casi todas las plataformas de streaming usan CDN. Servidores repartidos geograficamente que almacenan copias multimedia. CDN puede ser privado (Google) o third-party. Politicas de ubicacion:

Enter Deep: colocar servidores en access ISPs. Se está más cerca de los endsystems. Muy distribuido, más dificil de mantener.
Bring Home: servidores en menos lugares (por ej IXPs). Más fácil de mantener a expensas de un mayor delay
Cuando se hace una request de un video, CDN la intercepta y determina el servidor CDN más conveniente y redirecciona la request. La intersepcion se logra haciendo uso de la DNS.

¿Cómo determinar el cluster de CDN más conveniente? cluster selection strategy:

Más cercano geograficamente
real time measurements: toman medidas del delay para determinar el más cercano en cuanto a tiempo y no distancia
También es posible el caso de P2P streaming, muy similar a lo comentado en la sección de P2P