RESUMEN REDES Y COMUNICACIONES

CAPA DE APLICACIÓN:

Cliente-servidor : Siempre existe un host activo denominado **servidor** que da servicio a las solicitudes de muchos otros hosts que son los **clientes**, como es el caso de 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.El servidor tiene una dirección fija y conocida denominada dirección IP.Entre las aplicaciones más conocidas que utilizan la arquitectura cliente-servidor se encuentran las aplicaciones web,FTP , Telnet y de correo electrónico.
En una aplicación cliente-servidor un único host servidor es incapaz de responder a todas las solicitudes de sus clientes,suele utilizarse una agrupación (clusters) de host denominados **centro de datos** para crear un servidor virtual de gran capacidad
En una arquitectura p2p la aplicación explota la comunicación directa entre parejas de hosts conectados de forma intermitente , conocidos como peers,Los peers no son propiedad del proveedor del servicio sino que son las computadoras de escritorio controlados por usuarios,se comunican sin pasar por un servidor dedicado

Características: una de las características más importantes es la **auto-escalabilidad**.Por ejemplo en una aplicación de compartición de archivos P2P aunque cada peer genera una carga de trabajo solicitando archivos , también añade capacidad de servicio al sistema distribuyendo archivos a otros peers.P2P se enfrenta a los siguientes retos:

- Orientadas al ISP: ya que los isp te dan mas descarga que subida y en la arquitectura p2p desplazan el tráfico de carga/subida de los servidores a los ISP residenciales
- Seguridad: al ser tan abierta y distribuido tienen un reto hacia la seguridad
- Incentivos: El éxito de las aplicaciones p2p también depende de convencer a los usuarios para ofrecer voluntariamente a las aplicaciones recursos de ancho de banda de almacenamiento y de computación

Procesos de comunicación

Un proceso envía mensajes a la red y los recibe a la red a través los **“sockets”.**Un socket es la interfaz entre la capa de aplicación y la capa de transporte de un host.

Servicios de capa de aplicación :

- **Transferencia de datos fiable**: si un protocolo proporciona un servicio de entrega de datos garantizado , se dice que proporciona una transferencia de datos fiable.
- **Tasa de transferencia** : Las aplicaciones con requisitos de tasa de transferencia se conocen como aplicaciones sensibles al ancho de banda. Mientras que las aplicaciones sensibles al ancho de banda tienen que cumplir requisitos para la tasa de transferencia, las aplicaciones elásticas pueden hacer uso de la tasa de transferencia mucha o poca que haya disponible
- **Temporización** : Como por ejemplo que cada bit en el socket llegue al receptor en no más de 100 milisegundos.esto es importantes para aplicaciones en tiempo real , juegos multijugador,teleconferencias,etc.
- **Seguridad**: Por ejemplo en el host emisor un protocolo de transporte puede cifrar todos los datos transmitidos por el proceso emisor y en el host receptor puede descifrar los datos antes de entregarlos al proceso receptor.

Introducción tcp y udp

Un protocolo de la capa de aplicación define :

- Los tipos de mensajes intercambiados por ejemplo de solicitud y mensajes de respuesta
- La sintaxis de los diversos tipos de mensajes , como los campos de los que consta el mensaje y cómo se delimitan esos campos
- La semántica de los campos , como el significado de la información contenida en los campos
- Las reglas para determinar cuando y como un proceso envía mensajes y responde a los mismos

WEB Y HTTP:

Protocolo de transferencia de hipertexto, pertenece a la capa de aplicación, se implementa en dos programas cliente y servidor

HTTP utiliza TCPcomo su protocolo de transporte subyacente.El cliente HTTP primero inicia una conexión TCP con el servidor.Una vez que la conexión se ha establecido los procesos de navegador y de servidor acceden a TCP a través de sus interfaces de sockets.El cliente envía mensajes de solicitud HTTP al socket y recibe mensajes de respuestas HTTP procedentes de su interfaz de socket.De la misma funciona del lado del servidor,una vez que el cliente envía un msj a su socket, el mensaje deja de estar “en las manos” del cliente y pasa “a las manos”deTCP.Dado que HTTP no mantiene ninguna información acerca de los clientes se dice que HTTP es un **protocolo sin memoria del estado.**

Conexión HTTP No persistente :

Cuando en cada par solicitud/respuesta se envían las solicitudes y rtas a través de distintas conexiones TCP

1. El proceso cliente HTTP inicia una conexión TCP con el servidorwww.unaEscuela.edu en el puerto 80(defecto para HTTP).Asociados con la conexión TCP, habrá un socket en el cliente y un socket en el servidor
2. El cliente HTTP envía un mensaje de solicitud HTTP al servidor a través de su socket.El mensaje de solicitud incluye el nombre de la ruta /un Departamento/<|
3. El proceso servidor HTTP recibe el mensaje de solicitud a través de su socket, recupera el objeto /un Departamento/home.index de su medio de almacenamiento,encapsula el objeto en un mensaje de respuesta HTTP y lo envía al cliente a través de su socket
4. El proceso servidor HTTP indica a TCP que cierre la conexión TCP
5. El cliente http recibe el mensaje de respuesta.La conexión TCP termina.El mensaje indica que el objeto encapsulado en un archivo HTML.El cliente extrae el archivo del mensaje de rta,examina el archivo HTML y localiza las referencias a los 10 objetos JPEG
6. Los cuatro primeros pasos se repiten para cada objeto JPEG referenciado.
   Este tipo de conexiones presentan inconvenientes como tener que establecer y mantener una conexión nueva para cada objeto solicitado,para cada conexión se debe asignar los buffers TCP y las variables TCP tienen que mantenerse tanto en el cliente como en el servidor.Esto puede sobrecargar el servidor web.Otro problema es el retardo de entrega de dos RTT uno para establecer la conexión TCP y otro para solicitar y recibir el objeto

Conexión HTTP persistentes:Se puede enviar una pagina web completa en la misma conexión TCP ya que el servidor la deja abierta.La conexión se cierra después de un determinado intervalo de tiempo en que no haya actividad

Persistente o no persistente :

no persistente : requiere 2 tiempos de requerimientos por objetos, el So debe trabajar para conexión TCP , el navegador abre conexiones paralelas para los objs referenciados
persistente: el servidor deja las conexiones abiertas después de enviar la respuesta,los mensajes siguientes son enviados por la conexión,

por defecto HTTP utiliza conexiones persistentes y en serie

Formato de mensaje HTTP: se compone de las líneas de solicitud (campo de método get o post ,url y versión http) y las líneas de cabecera **(host en el que reside el objeto, browser y lenguaje)**

Cookies:Permiten identificar al usuario para autorizar o enviar contenido en función de su identidad, también guardan preferencias de los usuarios.
Entre las desventajas se encuentran los problemas de privacidad (las compañías obtienen mucha info a partir de los datos que almacenan los sitios web), y problema de seguridad (robo de sesiones)
Se componen de :

1. línea de encabezado cookie en el msj respuesta HTTP
2. línea de encabezado cookie en el requerimiento HTTP
3. archivo cookie almacenado en la máquina del usuario y administrada por el browser
4. base de datos en sitio web

Mecanismo que permite a las aplicaciones web del servidor “manejar estados”.
El cliente hace un request.
El servidor retorna un recurso (un objeto HTTP, como una página HTML) indicando al cliente que almacene determinados valores por un tiempo.
La Cookie es introducida al cliente mediante el mensaje en el header Set-Cookie: mensaje que indica un par (nombre,valor).
El cliente en cada requerimiento luego de haber almacenado la Cookie se la envía´a al servidor con el header Cookie:.
El servidor puede utilizarlo o no.
El servidor puede borrarlo.
Esta información puede ser usada por client-side scripts

18.Indique las diferencias principales entre HTTP 1.0, HTTP 1.1 y HTTPS.

HTTP 1.0

- Define un formato de mensaje para el Request y otro para el Response.
- Se debe especificar la versión en el requerimiento del cliente.
- Para los Request, define diferentes métodos HTTP.
- Define códigos de respuesta.(200 OK , 404 NOT FOUND)
- Admite repertorio de caracteres, además del ASCII, como: ISO 88591, UTF8, etc.
- Admite MIME (No solo sirve para descargar HTML e imágenes).
- · Por default NO utilizar conexiones persistentes, en HTTP 1.0 para generar una conexión persistente se utiliza “Connection: Keep-Alive” en el header
- GET: obtener el documento requerido. Puede enviar información, pero no demasiada. Es enviada en la URL. Formato ?var1=val1&var2=val2.... Max. info- 256 bytes.
- HEAD: idéntico a GET, pero solo requiere la meta información del documento, por ejemplo, su tamaño. Usado por clientes con cache.
- POST: hace un requerimiento de un documento, pero también envía información en el Body. Generalmente, usado en el fill-in de un formulario HTML (FORM). Puede enviar mucha más información que un GET.
- PUT: usado para reemplazar un documento en el servidor.
- En general, deshabilitada. Utilizado, por ejemplo, por protocolos montados sobre HTTP, como WebDAV [WDV].
- DELETE: usado para borrar un documento en el servidor. En general, deshabilitada. También, puede ser utilizada por WebDAV.
- LINK, UNLINK: establecen/des-establecen relaciones entre documentos

HTTP 1.1

● Nuevos mensajes HTTP 1.1: OPTIONS, TRACE, CONNECT.
● Conexiones persistentes por omisi´on.
● Pipelining.

Pipelining: Sólo disponible en conexiones persistentes, no necesita esperar la respuesta para pedir otro objeto HTTP.
TRACE: método utilizado para debugear, el cliente puede comparar lo que envía con lo que recibe.
CONNECT: utilizada para generar conexiones a otros servicios montadas sobre HTTP.

HTTPS

● En el protocolo HTTP las URLs comienzan con "http://" y utilizan por defecto el puerto 80, Las URLs de HTTPS comienzan con "https://" y utilizan el puerto 443 por defecto.
● HTTP es inseguro y está sujeto a ataques maninthemiddle
y eavesdropping que pueden
permitir al atacante obtener acceso a cuentas de un sitio web e información confidencial. HTTPS está diseñado para resistir esos ataques y ser menos inseguro.
● HTTPS no es un protocolo separado, pero difiere del uso del HTTP ordinario sobre una Capa de Conexión Segura cifrada Secure Sockets Layer (SSL) o una conexión con Seguridad de la Capa de Transporte (TLS).
● Adquisición de certificados.

Almacenamiento en cache web: una cache web también denominada servidor proxy, es una entidad de red que satisface solicitudes HTTP en nombre de un servidor web de origen.La cache web dispone de su propio almacenamiento en disco y mantiene en el copias de los objetos solicitados recientemente.

-El browser envía todos los requerimientos HTTP al cache :
-Si el objeto está en cache : cache retorna objeto

Sino el cache requiere los objetos desde el servidor web y retorna el objeto al cliente
La caché actúa como cliente y servidor , normalmente el caché está instalado por ISP
Ventajas:

- Reduce tiempo de respuesta de las peticiones del cliente
- Reduce tráfico de los enlaces Internet de la institución
- Permite a proveedores con bajos recursos entregar contenido en forma efectiva

Los web browser tienen sus propias cache locales.
Los servidores agregan headers:
Last-Modified: date
ETag: (entity tag) hash
Requerimientos condicionales desde los clientes:
If-Modified-Since: date
If-None-Match: hash
En general corren sobre UDP.

Get condicional: http dispone de este mecanismo que permite a la cache ceritficar que sus objetos estén actualizados.En primer lugar una cache proxy envía un mensaje de solicitud a un servidor web en nombre de un navegador que realiza una solicitud.En segundo lugar el servidor web envía a la cache un mensaje de respuesta con el objeto solicitado.La cache reenvia el objeto al navegador que lo ha solicitado pero también lo almacena localmente.Y lo

que es mas importante, la cache también almacena la fecha de la ultima modificación junto con el objeto.Otro navegador solicita el mismo objeto a través de la cache y el objeto todavía se encuentra almacenado allí.Puesto que este objeto puede haber sido modificado en el servidor web en el transcurso de la semana , la cache realiza una comprobación de actualización ejecutando un GET condicional.Si el valor de cabecera *if-modified-since* es igual a la línea de *last-modified* este get condicional le pide al servidor que envie el objeto solo si ha sido modificado después de la ultima fecha especificada.

FTP transferencia de archivos**:**

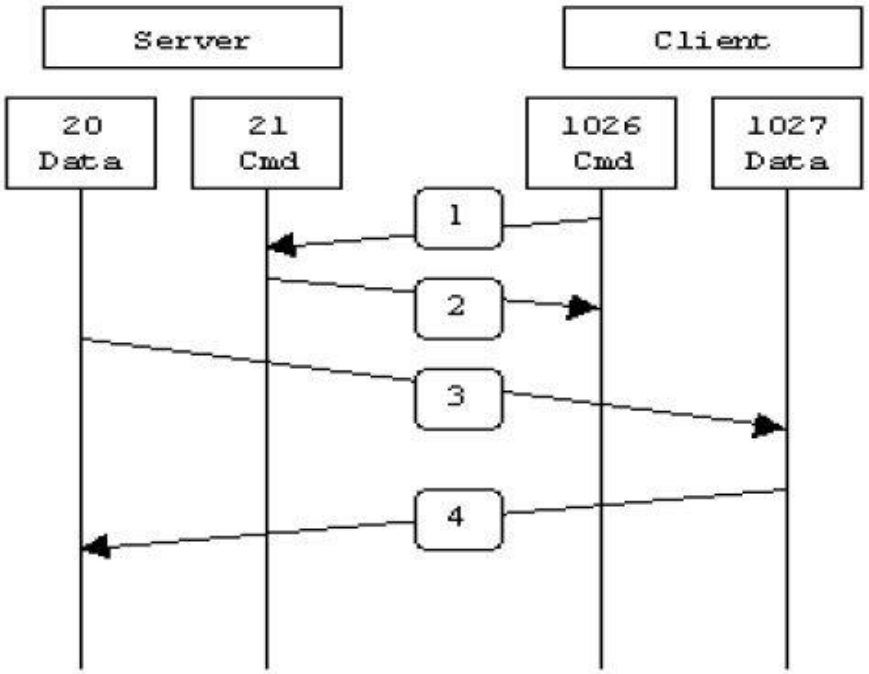
Protocolo de transferencia de archivo que tiene muchas características en común con HTTP , los dos se ejecutan sobre **TCP** , sin embargo a diferencia de http , ftp utiliza 2 conexiones TCP paralelas para transferir un archivo , una **conexión de contro**l y una **conexión de dato**.La primera se usa para enviar información de control entre los dos hosts , como la identificación del usuario , la contraseña , comandos para modificar el directorio remoto y comandos para introducir y extraer archivios.La conexión de datos se utiliza para enviar un archivo.Como FTP utiliza una conexión de control separado , se dice que FTP envía su información de control **fuera de banda**.Mientras HTTP envía **en banda** porque envía las líneas de cabecera de solicitud y rta por la misma conexión TCP que transporta el propio archivo transferido.

Cuando un usuario inicia una sesión FTP con un host remoto , el lado del cliente de FTP(usuario) establece en primer lugar una conexión de control TCP con el lado del servidor(host remoto) en el puerto número 21.El lado del cliente envía la identificación y la contraseña del usuario a través de esta conexión de control.El lado del cliente FTP también envía a través de la conexión de control comandos para modificar el directorio remoto. Cuando el lado del servidor recibe un comando para realizar una transferencia de archivo(modo pasivo o activo) a través de la conexión de control el lado del servidor inicia una conexión de datos TCP con el lado del cliente.

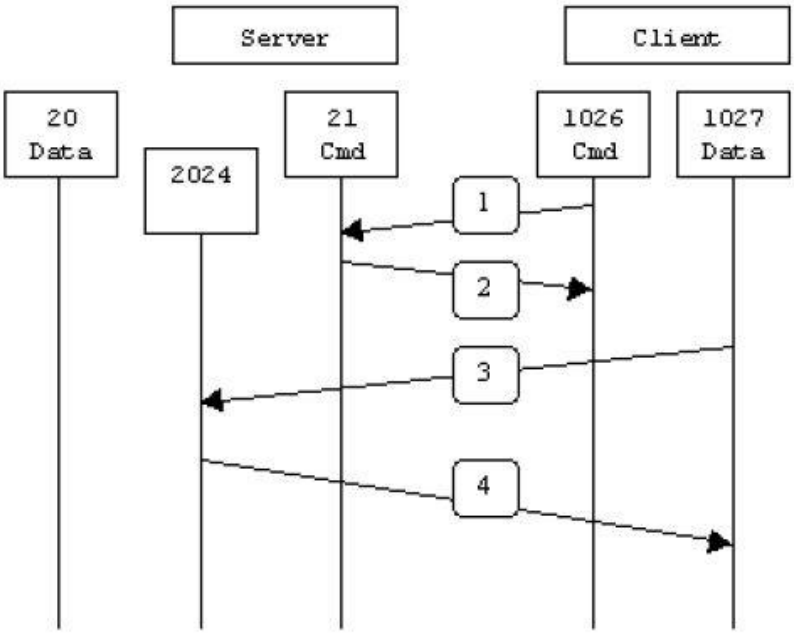
FTP envía un archivo a través de la conexión de datos y cierra la conexión.Si se quiere transmitir otro se abre una nueva conexión de datos.La conexión de control permanece abierta hasta que la sesión termine.Por lo tanto FTP del lado del servidor tiene que mantener un **estado** de usuario, mientras que en HTTP no hacia falta.

es activo o pasivo respecto al servidor

FTP Activo



FTP Pasivo



Comandos FTP :
RETR: Obtener un archivo desde el servidor
STOR envía un archivo al servidor
LIST: listar archivos
DEL: borrar un archivo
STAT: obtener info de un archivo en el servidor

FTP	HTTP
Diseñado para upload	Se puede con PUT, POST
Soporte de download	Soporte de download
Formato ASCII/binary	Meta-data content type
Sin headers	Con headers
FTP command /responde más lento	Pipeline
Más complejo con firewalls y NAT	Más amigable con firewall y NAT
Dos conexiones	Una conexión
Autenticación	Autenticación

CORREO ELECTRÓNICO EN INTERNET:

Existen 3 componentes principales: **agentes de usuario, servidores de correo** y el **protocolo simple de transferencia de correo(SMTP)**.

- **MUA (Mail User Agent):** Interfaz con la que el usuario puede leer , editar o escribir e-mails, utiliza SMTP POP e IMAP y se comunica con MTA y MDA
- **MTA (Mail Transport Agent) :** se encarga de enviar el mensaje al servidor donde esta almacenada la casilla de mensaje destino , usa SMTP
- **MDA (Mail Delivery Agent):** Se encarga de tomar los mensajes recibidos por el MTA y llevarlos al usuario , utiliza POP y/o IMAP

SMTP es el principal protocolo de la capa de aplicación para el correo electrónico, tiene dos lados el lado del cliente, que se ejecuta en el servidor de correo del emisor y el lado del servidor que se ejecuta en el servidor de correo del destinatario.
Ejemplo de Alicia:

1. Alicia invoca a su agente de usuario para correo electrónico , proporciona la dirección de correo electrónico de Benito, y compone el mensaje
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 al cola de mensajes.Abre una conexión TCP con su servidor 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 el mensaje en el buzón de Benito
6. Benito invoca a su agente de usuario para leer el mensaje cuando quiera

SMTP no utiliza servidores de correo intermedios para enviar correo.
En primer lugar el cliente SMTP establece una conexión TCP en el puerto 25 del servidor SMTP.Si el servidor no esta operativo el cliente lo intenta mastarde.Una vez establecida la conexión , el servidor y el cliente llevan a cabo el proceso de negociación de la capa de aplicación.Durante esta fase de negociación el cliente SMTP especifica la dirección de correo del emisor, y la dirección del correo del destinatario.Una vez presentador el cliente envía el mensaje.SMTP cuenta con el servicio de transferencia de datos fiable de TCP para transferir el mensaje al servidor sin errores, el cliente repite este proceso con la misma conexión TCP hasta que no quiera enviar mas mensajes que se le indica que se cierre la conexión.Comandos para el envio de email:MAIL,FROM,RCPT,TO,DATA,QUIT.
HELO: para abrir una sesión con el servidor
MAIL FROM: para indicar quien envía el mensaje
RCPT TO: para indicar el destinatario del mensaje
DATA: para indicar el comienzo del mensaje, éste finalizará cuando haya una línea únicamente con un
punto.
QUIT: para cerrar la sesión

HTTP O SMTP:

Similitudes:

Los dos transfieren archivos , ambos usan conexiones persistentes.

Diferencias:

1. 1)Http es un protocolo **pull**(extracción)osea que los usuarios utilizan HTTP para extraer la información previamente cargada en un servidor web, en cambio SMTP es **push**(inserción) el servidor de correo emisor introduce el archivo en el servidor de correo receptor

2. SMTP requiere que cada mensaje este en ASCII de 7 bits,en HTTP no hay restricción
3. Tiene que ver cómo se maneja un documento con imágenes, en HTTP se encapsula cada objeto en su propio mensaje de respuesta HTTP.El correo incluye todos los objetos del mensaje en un mismo mensaje

POP3 :protocolo de acceso a correo extremadamente simple.POP3 se inicia cuando el agente de usuario(el cliente) abre una conexión TCP en el puerto 110 al servidor de correo(el servidor),utiliza el formato ASCII y corren sobre SSL (Secure Socket Layer) .Una vez establecida la conexión TCP, POP3 pasa a través de tres fases: autorización,transacción y actualización:

1. **Autorización:** El agente usuario introduce el usuario y contraseña en texto legible
2. **Transacción:** El agente usuario recupera (o puede borrar tambien) los mensajes
3. **Actualización;** Se termina la sesión y el servidor borra los mensajes marcados si es que los hay.

El servidor puede devolver dos posibles respuestas antes cada comando

- +OK el comando es correcto
- -ERR hubo algún error en el comando anterior

Los comandos posibles que se envían al servidor son : *list (listar) ,retr (abrir), dele , (deletear), quit (borrar).*
Nota : Si está activo el modo de descargar y borrar, en el caso que el usuario se traslade de pc , este no podrá leer los mensajes.Para ello se utiliza el modo descargar y guardar
Nota 2 : pop3 NO guarda los estados ni ningún tipo de información de la sesión

IMAP: Este protocolo más complejo tiene varias funciones más que el pop3 , entre ellas permite crear carpetas para organizar los correos(por default va a bandeja de entrada),también almacena información durante las distintas sesiones.Guarda estados (nombre de las carpetas) , Permite buscar mensajes por Headers

Correo electrónico web : hoy en día cuando se envía-recibe desde un browser a los correos electrónicos se hace mediante HTTP en vez de pop/imap/smtp, pero internamente el servidor de correo electrónico sigue funcionando con los protocolos pop-imap y smtp

¿Para que se utilizan los encabezados MIME types? Indicar los protocolos de aplicación que lo utilizan.
MIME permite enviar datos binarios.MIME continúa usando la RFC 822, pero agrega una estructura al cuerpo del mensaje y definir reglas de codificación para los mensajes no ASCII, los mensajes MIME puede enviarse y usando los programas y protocolos de correo electrónico existente. Todo lo que tiene que cambiar son los programas emisores y receptores, lo que tiene que cambiarse son los programas emisores y receptores, lo que pueden hacer los usuarios mismo.
MIME define cinco nuevos encabezados de mensajes:
encabezados :
MIME Version: indica al agente de usuario receptor del mensaje que está tratando con un mensaje MIME, así como la versión. Se considera que cualquier mensaje que no contenga un encabezado MIME Version
será procesado como texto normal.
Content Description: es una cadena ASCII que dice lo que está en el mensaje. Permite al

usuario saber qué tipo de contenido sin haberlo codificado aún, para luego decidir si corresponde o no.
ContentId: Identifica al contenido; usa el mismo formato que el encabezado estándar Messageid.
Content Transfer Encoding: indica la manera en que está envuelto el cuerpo para su transmisión a través de una red donde se podría tener problemas con la mayoría de los caracteres distintos de las letras, números y signos de puntuación.
ContentType: El propósito de este campo es describir la información contenida en el cuerpo lo suficiente para que el agente de usuario receptor pueda utilizar el agente o mecanismo apropiado para presentar al usuario la información, o de lo contrario hacerse cargo de la información de la manera más adecuada. Al valor de este campo se le denomina tipo de contenido.

Protocolos que utilizan mime: SMTP, HTTP.

DNS : como sabemos podemos identificar a los host mediante nombre de host o mediante las direcciones IP. lo que este protocolo hace es traducir de nombre de host a la dirección IP, entonces se puede definir el protocolo como :

1. Una base de datos distribuida implementada en una jerarquía de servidores DNS
2. Un protocolo de la capa de aplicación que permite a los host consultar la base de datos distribuida

Estas máquinas suelen ser UNIX que ejecuta el software BIND . **DNS corre sobre UDP y usa el puerto 53**
Cuando yo ingrese una dirección en la URL ¿Como funciona? :

1. Ejecuta del lado cliente de la aplicación DNS
2. El navegador extrae la URl y se la pasa a la aplicación DNS
3. El cliente DNS pasa ese nombre a un servidor DNS
4. El cliente DNS recibe una respuesta con la dirección IP de la consulta
5. El navegador recibe la ip e inicia una conexión TCP en el puerto 80 con esa dirección ip

Otras funciones del protocolo DNS :

- **Alias de host :** Un nombre de host como por ejemplo relay.west1-coast.enterprise.com puede tener alias mas cortos como enterprise.com ya que son mas memorables? .En este caso se dice que relay.west1-coast.enterprise.com es el *nombre de host canonico*
- **Alias del servidor de correo:** idem para correo electrónico
- **Distribución de carga:** Contiene servidores replicados para distribuir la carga.Cuando los clientes realizan una consulta DNS sobre un nombre asignado a un conjunto de direcciones, el servidor responde con el conjunto completo de direcciones IP pero rota el orden en cada respuesta

Funcionamiento DNS: cuando una aplicación necesita traducir un nombre de host a una ip , ejecuta el servicio de DNS (por ejemplo con una función en java),entonces se envía la consulta por UDP puerto 53 ,se esperan unos milisegundos hasta que la respuesta llega, como sabemos no hay un solo servidor DNS por la gran demanda de INTERNET , si fuese un solo servidor dns en todo el mundo los problemas serían :

- Único punto de fallo: En caso que rompa algo
- Volumen de tráfico : ya que se debería gestionar todo el tráfico
- Base de datos centralizada distante:si tenemos el servidor en nueva york , una consulta desde australia debería recorrer mucho
- Mantenimiento: El mantenimiento es complicado al tener toda la DB de los hosts en todo el mundo

Base de datos jerárquica y distribuida : 3 tipos de servidores

- **servidores DNS raíz:** existen nada más que 13 , cada servidores es un cluster de servidores replicados
- **Servidores de dominio de nivel superior (TLD):** responsables de los dominios com .org, .com , .edu, .net , .gov .Se sub-dividen en Genéricas (los responsables de .edu,.org.y en Countrycode (.ar .br)
- **Servidores DNS autoritativos:**cualquier organización que tenga un servidor público tiene que tener otro servidor DNS autoritativo que haga la traducción , si no lo tienen puede pagarle a una empresa para alojar su servidor dns autoritativo

Existe otro tipo de servidores DNS que no entra en la jerarquía y son los servidores DNS LOCALES , cuando un host realiza una consulta cualquiera , primero se conecta al DNS local y este envía la consulta a la jerarquía de DNS

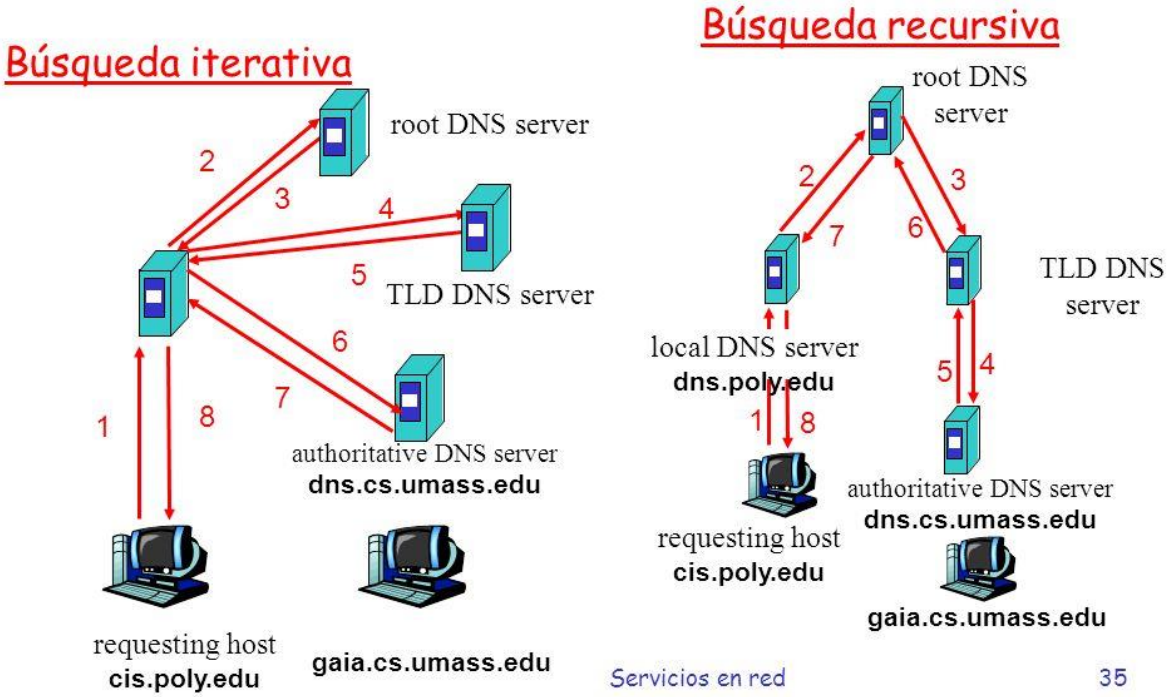
Ejemplo: El host benito.lopez.edu quiere conocer la direcion gaia.cs.umass.edu, tambien que mi dns local es dns.lopez.edu , y el servidor autoritativo de gaia es dns.umass.edu**.**

1. se envia la consulta con la direccion que se necesita resolver al DNS local dns.lopez.edu
2. eL DNS local reenvía la consulta a un servidor RAÍZ ,y este responde con las IP de los servidores TLD que resuelven .edu
3. eL DNS local envia OTRA consulta a uno de los servidores TLDy este responde con las ip de los servidores autoritavisoq eu resuelven el prefijo umass.edu es decir la direcion dns.umass.edu
4. Por ultimo el dns local envia directamente al servidor autoritativo y este responde la IP de la direccion gaia.cs.umass.edu

La primer consulta (paso 1) se la llama **recursiva** ya que le delega al dns local que se haga cargo de todo , mientras que el resto de las consultas son iterativas (aunque en teoría se puede ver todas las consultas DNS como iterativas y recursivas)

Búsqueda iterativa y recursiva

Ejemplo de resolución de nombres DNS



Almacenamiento en caché DNS: una de las características más importantes es que un servidor DNS puede almacenar en caché las consultas hechas , por ejemplo el DNS local puede almacenar por cierto periodo la dirección ip directamente de un host, o así como también guardar ip de los servidores TLD para saltarse la consulta de los RAIZ

Registros y mensajes DNS: los servidores DNS almacenan los **registros de recursos (RR)** , formado por :
(Nombre, valor , tipo, TTL.)
Si tipo = **A** entonces nombre es un dominio (gizmodo.ez) y valor es la dirección IP de ese dominio
Si tipo = **NS** entonces nombre es un dominio (gizmodo.es) y valor es el nombre de host de DNS autoritativo que resuelve esa el dominio gizmodo.es
Si tipo = **CNAME** entonces valor es un nombre de host canónico correspondiente al alias especificado por el nombre
Si tipo = **MX** entonces valor es un nombre de un servidor de correo que tiene un alias dado por nombre, Un cliente dns lo puede utilizar para tener el nombre canónico del servidor de correo

Inserccion de registros a la base de datos: i
Se deberá contactar a una de las posibles entidades de registración de registros, y brindarles los nombres e ip de los servidores DNS autoritativos primario y secundario , así

como el registro MX, una vez hecho todo esto se podrá acceder al sitio web y los empleados de la empresa podrán intercambiar correos
El software de DNS permite asignar roles de Servidor Primario y Servidor(es) Secundario(s). De esta forma, solo se requiere configurar el primario y luego el secundario obtendrá una copia de la base de datos del servidor maestro/primario. Es importante que los servidor estén actualizados, por eso debe existir algún mecanismo para mantenerlos sincronizados. El Servidor
Primario deberıa avisar a todos los Servidores Secundarios cuando se realiza un cambio, as´ı estos re-copiaron la base de datos de nombres desde el servidor maestro. La entre servidores se realiza v´ıa el mismo protocolo de DNS, con la diferencia que se hace sobre TCP en lugar de UDP, esto se debe a que, generalmente, los datos transmitidos ocupan más de 512 bytes (máximo para DNS sobre UDP). Esta operación se llama Transferencia de Zona o, en ingles, Zone Transfer

ARQUITECTURA P2P :

bitTorrent : es el protocolo p2p más popular.Varios pares(peers) participan en la distribución de un torrent/archivo, donde cada peer va descargando fragmentos (gralmente 256 kb) del archivo, una vez descargado puedes ser egoísta y borrarlo o dejar que otros sigan descargandolo
Cada archivo tiene un nodo de infraestructura (Tracker) que se registra cuando un peer se suma a la descarga del archivo
Cuando Benito se une a un torrente, el tracker selecciona una cantidad x de todos los pares que hay en el torrente y se Ías envía a Benito , entonces la pc de Benito va a empezar a tirar conexiones TCP a todas las IP de ese subconjunto x de pares que me tiro el nodo, con los que Benito pueda conectarse los llamamos “pares vecinos”
Benito irá pidiendo los fragmentos faltantes a los vecinos , para ello usa la técnica “primero el menos comun” osea los fragmentos más difíciles pero posibles de conseguir son los que va a pedir.
A la hora de enviar fragmentos a los vecinos que necesitan, torrent utiliza el algoritmo “intercambio inteligente” que le da prioridad a los que tienen suministran datos a mayor velocidad ,gralmente son los primeros 4 , y después se elige uno de forma aleatoria para también enviarle fragmentos.EL resto de estos 5 pares están filtrados

2 - CAPA DE TRANSPORTE

Servicios de transporte y protocolos

Un protocolo de la capa de transporte proporciona una **comunicación lógica** entre procesos de aplicación que se ejecutan en hosts diferentes, es decir los host actúan como si estuvieran conectados directamente aunque puede haber cientos de routers de por medio
En un principio la capa convierte los mensajes de las aplicaciones en **segmentos,** osea divide estos “mensajes” en partes pequeñas añadiendo una cabecera a cada fragmento.La

capa de transporte proporciona una comunicación lógica entre procesos que se ejecutan en distintos host , mientras que la capa de red proporciona una comunicación lógica entre host
 Los protocolos de transporte sólo corren en los sistemas finales
    - Lado emisor: divide el mensaje de la aplicación en segmentos y los pasa a la capa de red
    - Lado receptor: reensambla los segmentos en forma de mensajes y los pasa a la capa de aplicación

TCP :

- proporciona una transferencia de datos fiable : usando técnica de control de flujo , números de secuencia , mensajes de reconocimiento y temporizadores, hace que los datos lleguen bien y en orden
- proporciona mecanismos de control de congestión: evitan que cualquier conexión TCP inunde con una cantidad de tráfico excesiva los enlaces y routers existentes entre los host que están comunicandose, esto se hace regulando la velocidad de transmisión , mientras que en UDP podes enviar a cualquier velocidad

Multiplexación y demultiplexación :

Como sabemos un proceso puede tener uno o más sockets (puertas que pasan los datos de la red al proceso), por lo tanto la capa de transporte entrega los datos al socket intermediario y después van al proceso.Cada socket tiene un identificador único y el formato depende si es TCP O UDP.
Cada segmento contiene un conjunto de campos , la capa de transporte chequea estos campos y dirige este segmento al socket adecuado , esto se llama **multiplexación**,MIentras que la tarea de reunir los fragmentos de datos desde los diferentes sockets para crear los segmentos y pasarlos a la capa de red se llama **multiplexación.**
Dentro de ese conjunto de campos estan : **número de puerto de origen , número de puerto destino** (del 0 al 1023 reservados para protocolos de la capa de aplicación => **números de puertos bien conocidos).**

multiplexaxion y demultiplexacion en UDP: Suponemos que un proceso del host A, con el puerto UDP 19157 envia un fragmento de datos de una aplicacion a un proceso con el puerto UDP 46428 en el host B.La capa de transporte del host A crea un segmento que incluye los datos de la aplicacion , el numero de puerto origen , destino y otros valores mas.La capa de transporte pasa el segmento resultante a la capa de red.La capa de red encapsula el segmento en un datagrama IP y trata de entregar el semento al host receptor.Si llega al host B, la capa de transporte examina el numero de puerto destino especificado en el segmento 46428 y entrega el segmento a su socket identificado por dicho puerto.A medida que los segmentos UDP llegan a la red , el host dirige (demultiplexa) cada segmento al socket apropiado examinando el numero de puerto de destino del segmento.

Nota 1 : Recordar que el socket se lo identifica con la tupla ***(IP destino , nro de puerto destino)***

Nota 2 : El número de puerto origen serviria si el host B quiere responder al host A , entonces transforma ese número de puerto (y la IP) en su tupla destino

multiplexación y demultiplexación en TCP : el socket TCP a diferencia del UDP , utiliza 4 valores (ip origen , puerto origen , ip destino , puerto destino) para identificarse, solo si estos 4 valores se corresponden se podrán enviar segmentos el cliente y el servidor.
Un host servidor puede soportar muchos sockets TCP simultáneos
Servidores web tiene sockets diferentes por cada clientes conectado.HTTP no persistente tendrá diferentes sockets por cada requerimiento

UDP: Se dice que es un protocolo sin conexión ya que no tiene lugar una fase de establecimiento de la conexión entre las entidades de capa emisora y receptora
Ventajas de UDP sobre tcp :

- *Mejor control en el nivel de aplicación sobre qué datos se envían y cuando:* cuando un proceso manda datos a udp estos se empaquetan en un segmento udp e inmediatamente se entrega a la capa de red , sin control de congestión ni nada, va de una el loco
- *Sin establecimiento de la conexión:* no tiene retardo de establecimiento de conexión
- *Sin información del estado de la conexión:* Al no tener tanta información un servidor dedicado a una aplicación concreta puede soportar más clientes activos cuando la app se ejecuta sobre UDP que cuando se hace sobre TCP
- *poca sobrecarga debida a la cabecera de los paquetes*: Las cabeceras de los segmentos UDP ocupan 8 bytes mientras que en TCP 20 bytes

Suma de comprobacion de UDP : la suma de comprobacion de UDP proporciona un mecanismo de deteccion de errores.Se utiliza para determinar si los bits contenidos en los segmentos UDP han sidos alterados segun se desplazaban desde el origen hasta el destino(por ejemplo a causa de la existencia de ruido en los enlaces).Se calcula en el amisor el complemento a 1(invertir nro) de la suma de todas las palabras de 16 bits del segmehnto

protocolo de transferencia de datos fiables: los protocolos fiables basados en retransmisiones (que diga si el paquete llego bien o no) se los conoce como **protocolo ARQ (automatic repeat request, solicitud automática de repetición)**, estos tienen 3 capacidades:

- detección de errores
- realimentación del receptor : las respuestas de acuse de recibo positivo (ACK) y negativo (NAK) que el receptor envía para decir si llegaron bien o no
- retransmision: un paquete que se recibe con errores en el receptor será retransmitido por el emisor

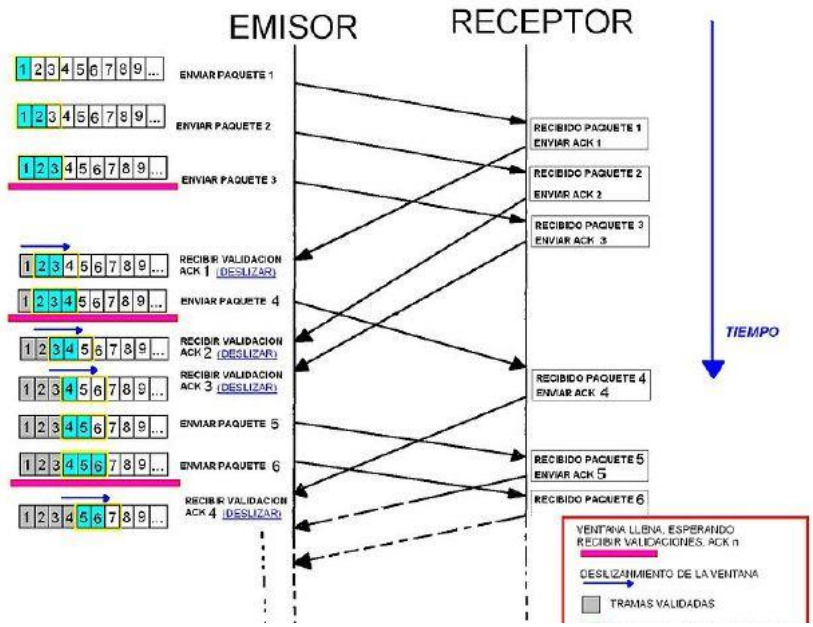
Ahora la pregunta es , cómo sabemos si ese paquete ACK/NAK tiene errores , la solución que se implementa para eso, es enumerar los paquetes añadiendo en el campo el **número de secuencia**, y con esto el receptor va a saber si el paquete se está retransmitiendo o no, los paquetes ACK/NAK no necesitan del número de secuencia ya que suponemos que contamos con un canal que no pierde dato

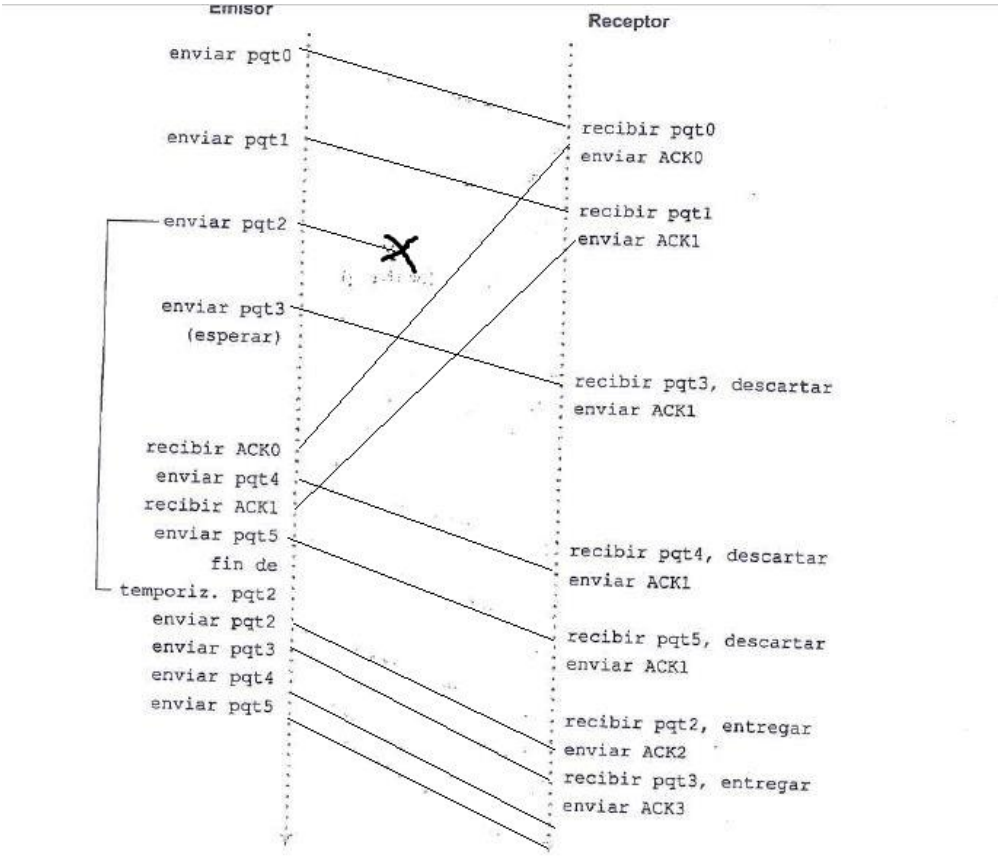
Protocolo fiable con pérdida de datos
Si a nuestro protocolo super complejo lo queres mejorar aún más , vamos a suponer que ahora hay pérdidas en la red, para solventar esto lo que hacemos es que a un x tiempo pasado si no recibí una respuesta entonces re envio , hay que calcular ese tiempo mínimo sabiendo el retardo ida y vuelta y lo que tarda el receptor en procesar el paquete

---

Procesamiento en cadena :

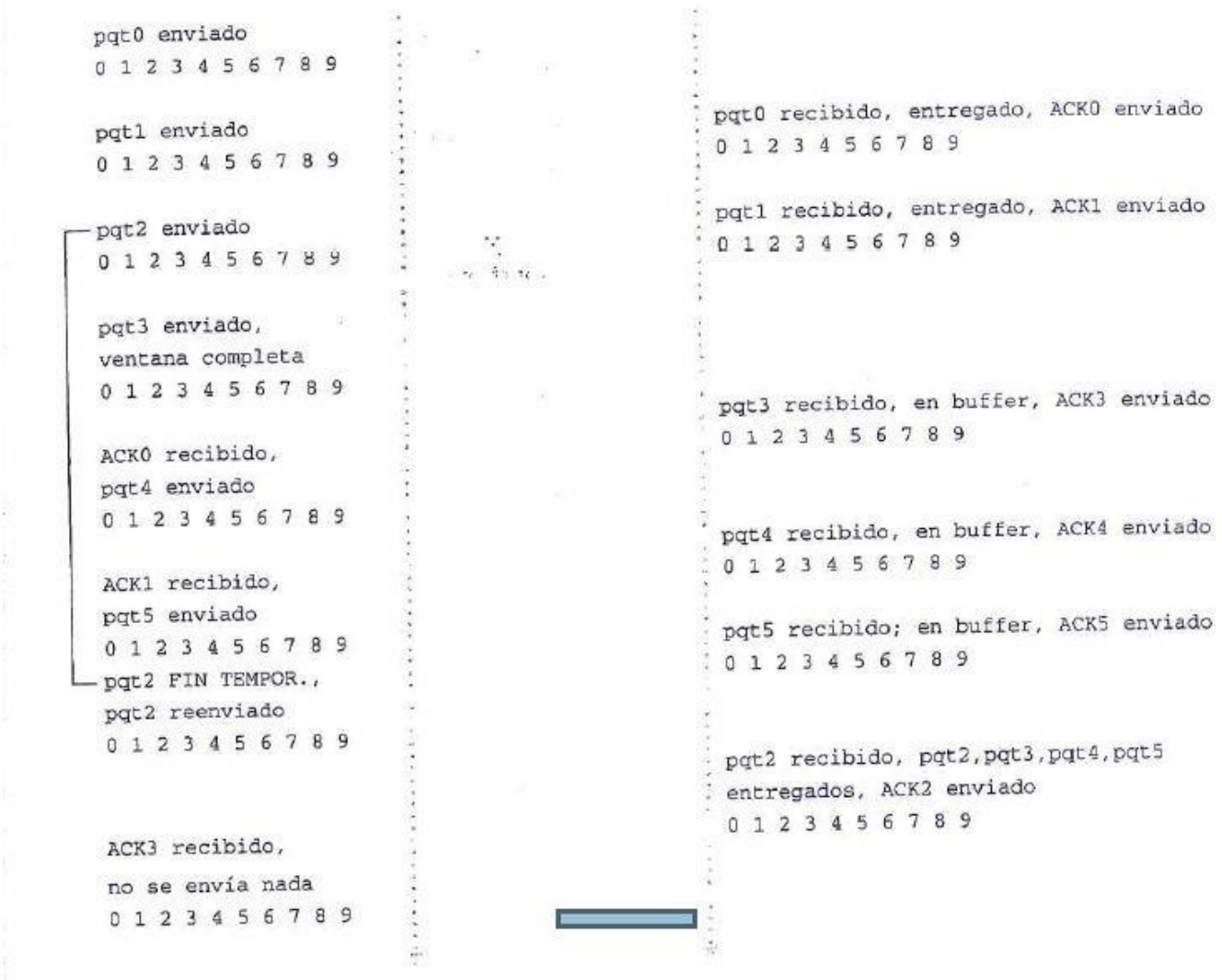
Retroceder N (GBN): En este protocolo el emisor puede transmitir varios paquetes sin tener que esperar a que sean reconocidos, pero esta restringido a no tener más de un número máximo permitido, N de paquetes no reconocidos en el canal
A N se la denomina **tamaño de ventana,** se dice que el protocolo GBN es un **protocolo de ventana deslizante**
El funcionamiento es bastante simple : del lado del receptor si se recibe un paquete con un número de secuencia n correctamente y en orden , el receptor envía el ACK para ese paquete n , y entrega los datos a la capa superior.En los casos restantes se descarta el paquete y se reenvía un mensaje ACK para el último paquete que llego correctamente (el mas nuevito)
Usa un timer por cada paquete en tránsito , En caso de Timeout retransmite el paquete N y todo los paquetes con número de secuencia siguientes
ejemplos:





En el ejemplo de arriba vemos un tamaño de ventana de 4 paquetes, entonces el emisor en un principio puede enviar los paquetes 0 a 3 ,, a medida que se reciben los sucesivos paquetes (ack0 y ack1), se la ventana se desplaza hacia adelante permitiendo al emisor enviar los paquetes 4 y 5.En el lado del receptor se pierde el paquete 2 por lo tanto los paquetes 3 , 4 y 5 no se reciben en orden correcto entonces como dijimos se descartan.

Repetición selectiva (SR): Nace a partir de la incomodidad de GBN de que podríamos estar teniendo varias retransmisiones por culpa de un solo paquete.Entonces hace que se transmiten solamente aquellos paquetes que se sospeche que llegaron al receptor con error.Para ello requiere que el receptor confirme *individualmente* que paquetes recibió correctamente.
Los paquetes no recibidos en orden se almacenarán en el buffer de recepción hasta que se reciban los que faltan(los paquetes con números de secuencia menores)
Por lo tanto solo se reenvían los paquetes sin ACK recibidos.El transmisor usa un timer por cada paquete sin ACK



Se ve fácilmente que el receptor almacena en el buffer los paquetes 3 4 y 5 y luego los entrega cuando llega el paquete 2

Transmisor:

Llega datos desde arriba:

Si el próximo # de sec. está en ventana, enviar paquete

Timeout(n)

:Re-enviar paquete n, reiniciar timer

ACK(n) en [senderbase,send base+N]:

Marcar paquete n como recibido

Si n es el paquete más antiguo sin ACK, avanzar la base de la ventana al próximo # de sec. sin ACK.

Receptor

Llega paquete n en [rcvbase,rcvbase+N-1]

-Enviar ACK(n).

-Si está fuera de orden:
almacenar en buffer

-En-orden: entregar a capa superior (también entregar paquetes en orden del buffer),

avanzar ventana al paquete
próximo aún no recibido
paquete n en [rcvbase-N,rcvbase-1]
ACK(n)
Otro caso:
ignoralo

------(Quedan dos casos raros de ejemplos para ver ***VER EN PPT!)----------

Mecanismo	Uso, comentarios
Suma de comprobación (checksum)	Utilizada para detectar errores de bit en un paquete transmitido.
Temporizador	Se emplea para detectar el fin de temporización y retransmitir un paquete, posiblemente porque el paquete (o su mensaje ACK correspondiente) se ha perdido en el canal. Puesto que se puede producir un fin de temporización si un paquete está retardado pero no perdido (fin de temporización prematura), o si el receptor ha recibido un paquete pero se ha perdido el correspondiente ACK del receptor al emisor, puede ocurrir que el receptor reciba copias duplicadas de un paquete.
Número de secuencia	Se emplea para numerar secuencialmente los paquetes de datos que fluyen del emisor hacia el receptor. Los saltos en los números de secuencia de los paquetes recibidos permiten al receptor detectar que se ha perdido un paquete. Los paquetes con números de secuencia duplicados permiten al receptor detectar copias duplicadas de un paquete.
Reconocimiento (ACK)	El receptor utiliza estos paquetes para indicar al emisor que un paquete o un conjunto de paquetes ha sido recibido correctamente. Los mensajes de reconocimiento suelen contener el número de secuencia del paquete o los paquetes que están confirmando. Dependiendo del protocolo, los mensajes de reconocimiento pueden ser individuales o acumulativos.
Reconocimiento negativo (NAK)	El receptor utiliza estos paquetes para indicar al emisor que un paquete no ha sido recibido correctamente. Normalmente, los mensajes de reconocimiento negativo contienen el número de secuencia de dicho paquete erróneo.
Ventana, procesamiento en cadena	El emisor puede estar restringido para enviar únicamente paquetes cuyo número de secuencia caiga dentro de un rango determinado. Permitiendo que se transmitan varios paquetes aunque no estén todavía reconocidos, se puede incrementar la tasa de utilización del emisor respecto al modo de operación de los protocolos de parada y espera. Veremos brevemente que el tamaño de la ventana se puede establecer basándose en la capacidad del receptor para recibir y almacenar en buffer los mensajes, o en el nivel de congestión de la red, o en ambos parámetros.

TCP : una conexión tcp proporciona un servicio **full-duplex.**

Casi siempre es una conexion **punto a punto**, osea entre un unico emisor y un unico receptor.no existe la posibilidad en tcp de comunicacion entre un emisor y muchos receptores.En un principio se establece el **acuerdo de tres fases** (seguir leyendo), para establecer la conexion, luego los dos procesos de app pueden enviarse datos uno a otro , de la forma que el cliente pasa un flujo de datos a traves del socket,y se encuentran en manos del protocolo TCP que se ejecuta en el cliente, lso datos van al **buffer de emision** de la conexion, la cant maxima de datos que puede ir tomando TCP de ese buffer esta limitada por el **tamaño maximo de segmento** (**MSS).**Cada lado de la conexion (emisor y

receptor)
tiene su propio buffer de emisión/recepción

Estructura del segmento TCP : consta de campos de cabecera y un campo de datos(limitado por MSS).
Al igual que UDP contiene nro de puerto origen y destino para multiplexar y demultiplexar y suma de comprobación.y se le agregan los siguientes campos:

- **Número de secuencia y nro de reconocimiento**: para implementar un servicio de transferencia fiable
- **Ventana de recepción:**número de bytes que un receptor está dispuesto a aceptar
- **Longitud de cabecera:** La cabecera TCP puede tener una longitud fija o variable
- **Opciones:**
- **indicador:** transporte los bits ACK/RST/SYN/FIN/PSH/URG

Número de secuencia : es el número del primer byte del segmento dentro del flujo de bytes,ej: si tenemos un archivo de 500.000 bytes y el tamaño MSS de 1000 bytes , y el primer byte del flujo de datos esta numerado en 0(en la práctica es un numero random), entonces se construyen 500 segmentos en total , y los números de secuencia van a ser : 0 , 1000 , 2000 ,etc
Numero de reconocimiento:*el numero de reconocimiento que el host A incluye en su segmento es el numero de secuencia del siguiente byte que el host A esta esperando de B.*Eh: el host A ha recibido un segmento de B de los bytes 0 a 535 y otro segmento q tiene de los bytes 900 a 1000 , por algun motivo no recibio los bytes de 536 a 899 , el host A va a enviar un segmento a B con el numero de reconocimiento 536.Una vez que lleguen los bytes faltantes estos se ubican/ordenan en el hueco faltante (gralmente)
Ejemplo de un cliente servidor , enviando la letra “C”:

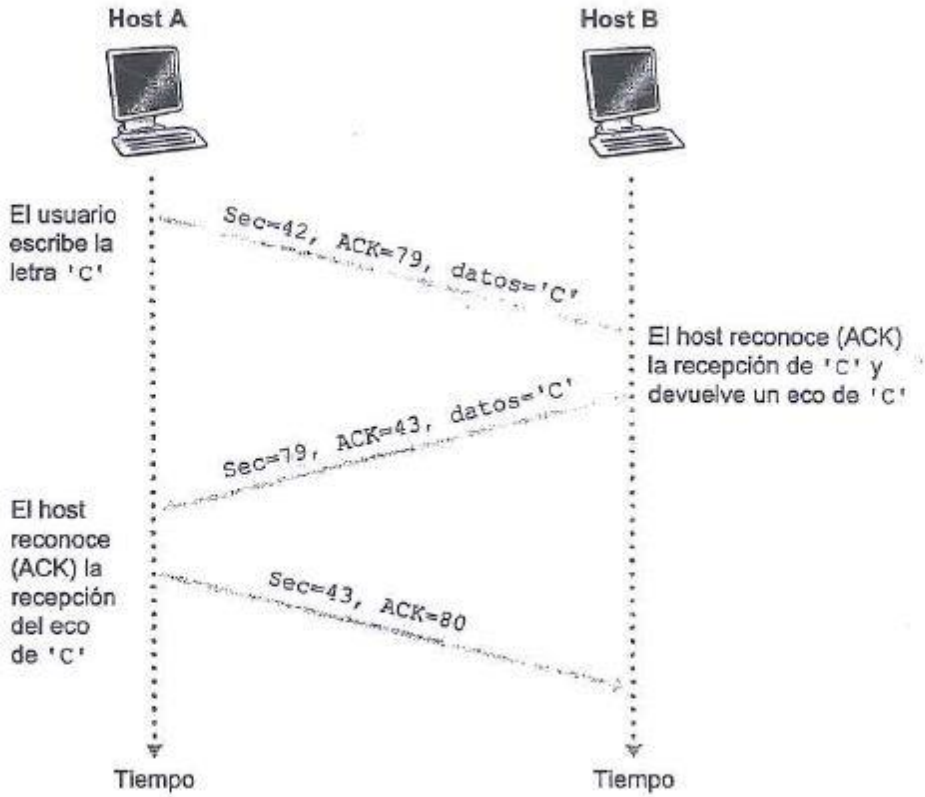
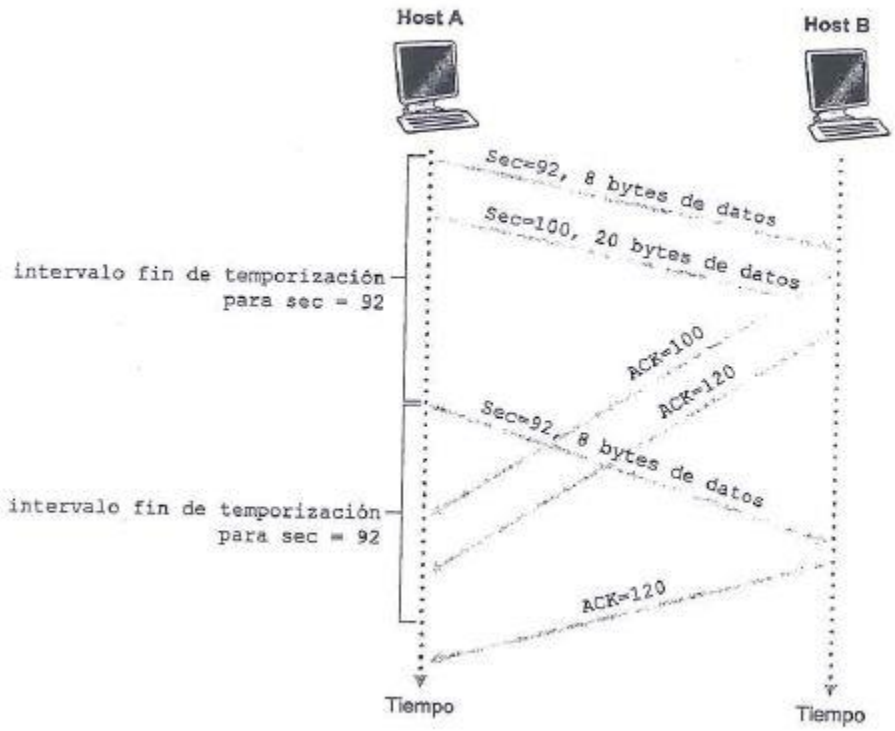


Figura 3.31 • Números de secuencia y de reconocimiento en una aplicación Telnet simple sobre TCP.

Ejemplo con temporizador

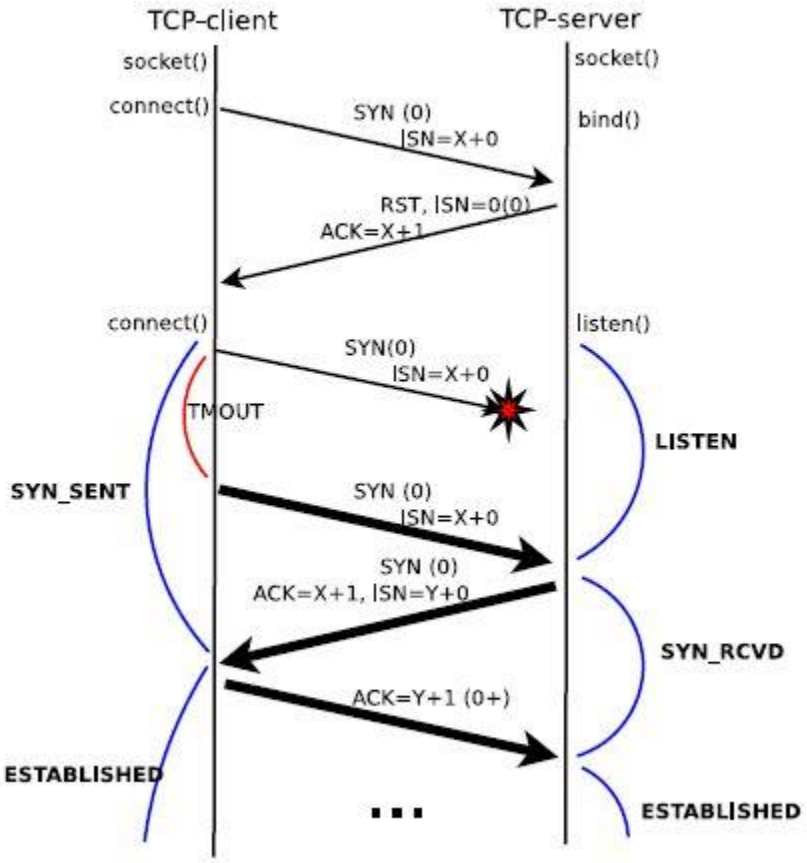


igura 3.35 • Segmento 100 no retransmitido.

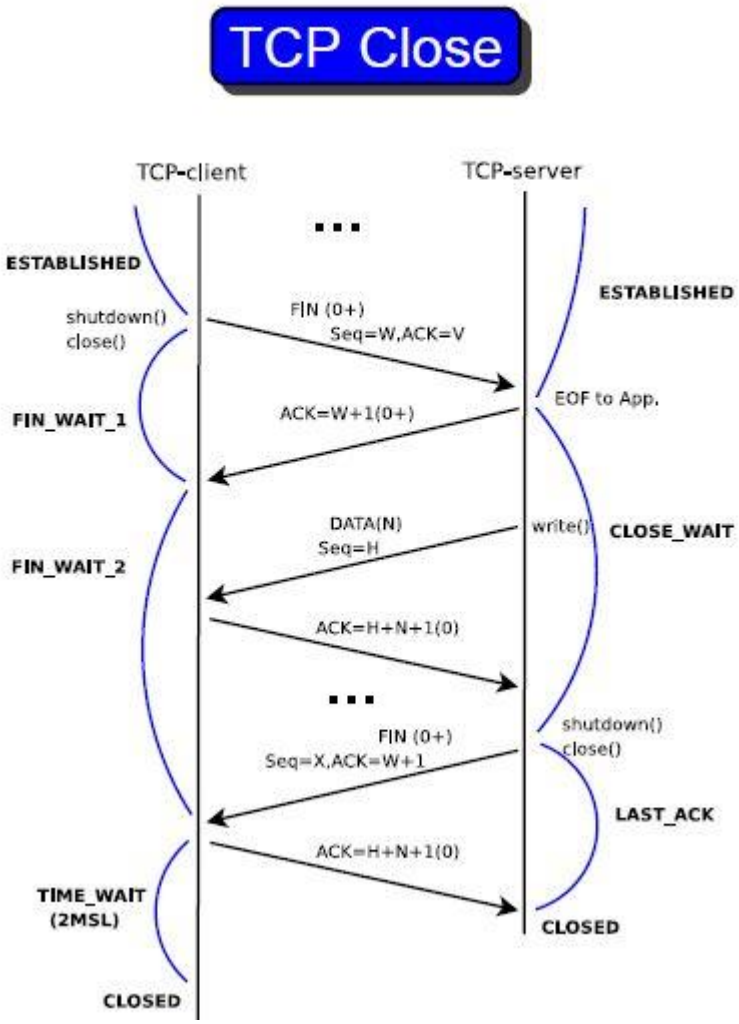
Control de flujo : si la aplicacion receptora esta ocupada, el emisor podria desbordar el buffer, para ello TCP proporciona un servicio de control de flujo, eliminando la posibilidad de desbordar el buffer del receptor.Es un servicio de adaptacion de velocidades (osea la del emisor y receptor),Para realizar esto el emisor mantiene la variable **ventana de recepcion** que guarda el espacio disponible en el receptor (en realiadad es una cuenta del tamaño total del buffer - lo que le queda por leer todavia al receptor)

Triple saludo - saludo de tres vías:

1. TCP del lado del cliente envía un segmento especial al servidor con el flag SYN en 1 , y seleciona un valor random para el número de secuencia
2. Una vez recibido por el servidor , esté asigna los buffers y variables TCP a la conexión y envía un segmento de conexión concedida al cliente TCP,donde pone el bit SYN en 1 , el campo de reconocimiento le asigna el de secuencia que llego+1 , y el servidor elige su propio número de secuencia inicial random.(Este paquete se lo suele llamar SYN ACK)
3. Al recibir el syn ack el cliente , ahora el cliente asigna buffers y variables a la conexión, y envía un último segmento, también pone en el campo de reconocimiento el nro random de secuencia que recibió del serv +1.y finalmente pone el bit SYN en 0



Ejemplo del triplete, con devolución del flag RST ya que no estaba abierto el socket todavía , dsp se pierde el siguiente SYN y dsp si se hace la conexión



Ahora si hablamos de cierre de conexion esto es mas simple , el cliente envía al servidor un segmento especial con el flag FIN en 1 , el servidor le devuelve el ACK , y seguido de esto el servidor envía otro segmento especial al cliente con el FIN en 1 tambien , y por ultimo el cliente responde con un ACK. Cuando los segmentos FIN llevan se liberan los buffers y las variables de la conexión TCP

8. ¿ Qué sucede al intentar establecer la conexión TCP y no existe un proceso asignado al socket(IP:port) ?
El host enviará al origen un segmento especial de reinicio (RST: Reset). Cuando un host envía un segmento reset le está diciendo al emisor: “No tengo un socket para ese segmento. Por favor no reenvíes el segmento.”

Control de Congestión:
-Controlar el tráfico que se envıa

evitando que colapse la red y se descarten los datos, teniendo que re transmitirse.
-A diferencia del control de Flujo: debe tener en cuenta los buffers de la red, no solo el buffer del receptor
-Controlar el tráfico que se envıa evitando que se colapse la red y se descarte teniendo que re transmitirse. Tiene en cuenta el estado de la red, no solo el buffer del receptor.

Mecanismos de control de congestión TCP: Utiliza un control de congestión *terminal a terminal* ya que la capa de red no proporciona soporte explícito a la capa de transporte para propósitos de control de congestión.Lo que hace este mecanismo es reducir o aumentar la velocidad de transferencia dependiendo si hay congestión o no en la red
Para reducir la velocidad de transferencia TCP hace uso de la variable ventana de congestión.
La cantidad de datos no reconocidos en un emisor no puede exceder el mínimo de entre ventana de congestión y ventana de recepción
La velocidad de transmisión del emisor es aprox igual a Ventana COngestión/RTT bytes/segundo.Ajustando el valor de la ventana de congestión el emisor puede ajustar la velocidad a la que transmite los datos a través de su conexión
Se dice que TCP es **auto-temporizado** ya que utiliza los paquetes de reconocimiento para provocar (o temporizar) los incrementos en la ventana de congestión
Para establecer la velocidad de transferencia TCP se basa en 3 cosas:

- Un segmento perdido implica congestión y por tanto la velocidad del emisor TCP debe reducirse cuando se pierde un segmento
- Mientras que un segmento que fue reconocido (llego el ACK) la velocidad de puede incrementarse
- tanteo de ancho de banda

Por último vamo a nombrar que el Algoritmo de control de congestión de TCP tiene 3 etapas : 1) arranque lento , 2)evitación de la congestion 3)recuperacion rapida (preguntas si se estudia o no)

Arranque lento:
Se inicia TCP con valor de la ventana de congestión aprox en 1MSS(tamaño máx de segmento) por lo que la velocidad de transmisión inicial es igual a MSS/RTT.Por ejemplo si MSS= 500bytes y RTT=200 milisegundos , la velocidad inicial es de 20 kbps**.**
EL valor de ventana de congestión se va a ir aumentando de a 1 MSS a medida que llegan los ACK (como que va todo bien) NOTA EL VALOR DE MSS SE VA INCREMENTANDO DE FORMA EXPONENCIAL EJ: SI ES 500KB INICIAL DSP SIGUE 1000KB Y DSP 2000 KB Y DSP 4000 KB.

- Si se pierde un paquete (osea hubo congestión) entonces arranca todo de nuevo y la ventana de congestión vuelve a 1.
- Si la ventana de congestión es igual al umbral (el umbral es igual al valor de la ventana /2 y entonces pasa a **evitación de congestión**
- Si se producen 3 pasa al modo **recuperación rápida**

En el modo de **evitación de la congestión** la ventana de congestión aumenta de forma menos drástica , es decir de forma lineal 1 (MSS/VENTANA de congestión).

Recuperación rápida: en esta fase el valor de la *ventana de congestión* se incrementa en 1 MSS por cada ack duplicado recibido correspondiente al segmento que falta y que ha causado que entre en este estado.Cuando llega un ACK para el segmento que falta, TCP entra de nuevo en el estado de evitación de la congestión luego de disminuir el valor de la ventana de congestión.

Conclusiones : En general la ventana de congestión consiste en un crecimiento lineal (aditivo) de 1MSS por RTT, seguido de un decremento multiplicativo (división entre dos).

(pregunta de APREF).
Pensando en slow start o arranque lento, si la ventana de congestión es 4, luego de que se envían los 4 segmentos y los mismos son reconocidos antes de un evento de pérdida, ¿Qué tamaño pasa a tener la ventana de congestión? ~
Va a tener tamaño 8.
Slow Start
Algoritmo para el cálculo de la ventana de congestión aplicado al principio de la conexión, y hasta que se alcanza el umbral de congestión. Consiste en lo siguiente:
● La ventana de congestión se inicia con el valor de un segmento de tamaño máximo (MSS).
● Cada vez que se recibe un ACK, la ventana de congestión se incrementa en tantos bytes como hayan sido reconocidos en el ACK recibido. En la práctica, esto supone que el tamaño de la ventana de congestión se doble por cada RTT, lo que da lugar a un crecimiento exponencial de la ventana.
n = 4;
Acks 1 2 3 4
Packets 1 2 4 8

Después de 3 ACKs duplicados:
- CongWin se va a la mitad
- posteriormente la ventana crece linealmente

PERO después de alcanzado el timeout(mas drastico):
- CongWin se va a 1 MSS;
- posteriormente la ventan crece exponencialmente
- Hasta llegar a un umbral, donde seguirá creciendo linealmente

3 - CAPA DE RED

Función de la capa de red : transporta paquetes desde un host emisor a un host receptor
Reenvío(forwarding): implica la transferencia de un paquete desde un enlace de entrada a un enlace de salida dentro de un mismo router
Enrutamiento(routing): implica a todos los routers de una red cuyas interacciones mediante los protocolos de enrutamiento determinan las rutas que seguirán los paquetes en sus viajes desde el origen hasta el destino
Todo router tiene una **tabla de reenvío**
conmutador de paquetes : dispositivo de conmutación de paquetes gral que transfiere un paquete desde la interfaz del enlace de entrada a la interfaz del enlace de salida

Entrega garantizada: este servicio garantiza que el paquete va a llegar a su destino
Entrega garantizada con retardo limitado: este servicio no solo garantiza la entrega del paquete sino tendra un limite de retardo especificado de host a host

A un flujo de paquetes entre origen y destino se le puede ofrecer:
entrega de los paquetes en orden
ancho de banda mínimo garantizado
fluctuación máxima garantizada
servicios de seguridad
La capa de red ofrece el servicio de **mejor esfuerzo**
Redes de circuitos virtuales : estas arquitecturas utilizan conexiones en la capa de red , que se denominan circuitos virtuales.Un circuito virtual consta de 1) una ruta entre host origen y destino 2) números de VC, un número para cada enlace a lo largo de la ruta, y 3)entradas en la tabla de reenvío de cada router existente a lo largo de la ruta.Un paquete que pertenece a un circuito virtual transportará un número de VC en su cabecera
En una red de circuitos virtuales , los routers de la red tienen que mantener **información de estado de la conexión** para las conexiones activas.existen 3 etapas identificables :

1. configuración del VC:Durante la fase de configuración, la capa de transporte del emisor contacta con la capa de red y le dice la dirección del receptor para que determine la ruta
2. Transferencia de datos: una vez establecido el circuito los paquetes pueden comenzar a fluir a lo largo del mismo
3. Terminación del VC: cuando el emisor/receptor informa a la capa de red que quiere terminar el circuito virtual

Los mensajes que los sistemas terminales envían a la red para iniciar o terminar un VC y los mensajes pasados entre los routers para configurar el VC se conocen como **mensajes de señalización** y los protocolos empleados para intercambiar estos mensajes se denominan **protocolos de señalización**

Redes de datagramas: Cuando un sistema terminal desea enviar un paquete marca el paquete con la dirección del sistema terminal de destino y luego introduce el paquete en la red

Protocolo de internet IP : reenvío y direccionamiento en internet:

blabla

Formato de los datagramas:

- **Número de versión :** iPV4 O IPV6
- **Longitud de cabecera:** determinan donde comienzan realmente los datos del datagrama
- **Tipo de servicio:** Se incluyeron con el fin de poder diferencias entre distintos tipos de datagramas IP (por ejemplo que requieran más prioridad)
- **Longitud del datagrama:** longitud total del datagrama
- **Identificador, indicadores, desplazamiento de fragmentación:**
- **Tiempo de vida :** ya lo sabes
- **Protocolo:** Solo se usa cuando un datagrama alcanza su destino final para decir si los datos se pasaran por medio de TCP o UDP
- **Suma de comprobación de cabecera**: ayuda a los routers a detectar errores de bit en un datagrama IP recibido
- **Direcciones IP de origen y destino:** Cuando un origen crea un datagrama , inserte su dirección IP en el campo de dirección IP de origen e inserta la dirección del destino final en el campo de dirección IP destino
- **Opciones:**
- **Datos**

Fragmentación del datagrama: Si el enlace (refiriéndose a la capa de enlace)del router tiene un MTU (es la cantidad max de datos que una trama de la capa de enlace puede transportar) MENOR al tamaño del datagrama IP entonces va a ser necesario fragmentar el datagrama IP.

Hay que tener en cuenta que los fragmentos tienen que ser reensamblados antes de llegar a la capa de transporte destino.Los diseñadores decidieron que el trabajo de reensamblar datagramas lo hagan los sistemas terminales en lugar de los routers.Para llevar a cabo el reensamblado se incluyeron en el datagrama los campos *identificador , indicador y desplazamiento de fragmentación* , donde el host emisor les pone un numero fijo en *identificador* cada fragmento saliente) para después armarlos,en el campo indicador pone todos en 1 , salvo el último que lo pone en 0 para saber que termino el datagrama completo La fragmentación no esta permitida en el IPV6 por cuestiones de seguridad ya que permitía a los atacantes realizar ataques DDoS cambiando offsets de los fragmentos, por último el campo desplazamiento para saber en qué posición encaja el fragmento.En el caso que falte alguna parte el datagrama se descarta por completo

IPV4: El límite entre el host y el enlace físico se denomina **Interfaz.**Un router tiene varias interfaces, Cada Interfaz tiene su propia dirección IP.

Una dirección IP tiene una longitud de 32 bits , se expresan en **notación decimal con punto** , en la que cada byte se escribe en formato decimal y separada en punto mediante cada byte.Cada interfaz tiene una IP global única (salvo con NAT).

Características :

- No orientado a conexión

- Mejor esfuerzo , no confiable
- PDU : datagrama/paquete

Funciones :

- DIreccionamiento
- Ruteo/forwarding
- Multiplexación/Multiplexación de protocolos

Tipos :

- Unicast: IP que se utiliza cuando se envía de un único emisor a un único receptor
- Broadcast: IP que se utiliza cuando se envía de un único emisor a todos los receptores
- Multicast: IP que se utiliza cuando se envía de un único emisor a un grupo de receptores
- Unicast :IP que se utiliza cuando la información es enrutada al mejor destino desde el punto de vista de la topología de la red

Introducción al CIDR, clases de los rangos IP , agotamiento de IPV4 , y que para obtener un conjunto de ips debo contactarme con el ISP y el ISP se contacta con la entidad de registro para los números y nombres ICANN

Obtener una dirección de host: Protocolo de configuración dinámica de host:
El protocolo **DHCP** permite a un host que se le asigne una direccion IP automaticamente , asi como tambien tener una dirección IP temporal que será distinta cada vez que el host se conecta a la red, también permite que un host obtenga info adicional como la máscara de subred, la dirección de su serv DNS local , etc.
A medida que los host se unen a la red y salen el servidor DHCP actualiza la lista de direcciones IP disponibles , cada vez que un host se une a la red , el servidor DHCP asigna una dirección arbitraria del conjunto de direcciones.Necesitamos reservar un servidor por subred (en un buen caso) que actúe como DHCP
Funcionamiento :

1. Descubrimiento del sevidor DHCP: el host envia un paquete por UDP de descubrimiento para buscar un servidor DHCP con la direcicion broadcast 255.255.255.255, y con la IP de origen 0.0.0.0 (ya que no tengo ip asignada)
2. Oferta del servidor DHCP: el servidor dhcp que recibe el mensaje anterior , responde al cliente con un **mensaje de oferta DHCP** también a la dirección 255.255.255.255, este mensaje va a tener un ID , la dirección propuesta para el cliente con la máscara de red y el tiempo en que la IP esa será válida
3. Solicitud DHCP: el cliente recién llegado seleccionará entre las ofertas de servidor y responde con un **mensaje de solicitud DHCP** devolviendo los parametros de configuracion

4. ACK DHCP: El servidor contesta al mensaje de solicitud DHCP con un mensaje ACK para confirmar

Traducción de direcciones de red (NAT):
Nat básico : Se tiene una IP pública por cada IP privada, se puede hacer de forma **estática** como de form **dinámica** (se requiere un timer por cada entrada)
NAPT:NETWORK ADDRESS PORT TRANSLATION
La idea es de tener muchos dispositivos internos en una casa con ip privadas para que se comuniquen entre ellos ,pero una sola dirección pública cuando sale del **router-NAT**, esto oculta detalles domésticos al exterior.
Ahora si hay una sola IP pública , cuando llega un paquete como se a que dispositivo mandarlo? para eso existe una **tabla de traducción NAT** almacenada por el router NAT, donde almacena los números de puerto como las direcciones IP en la entrada de la tabla
Ejemplo: Mi celu con ip 10.0.0.1 y puerto 3345 quiere entrar a gizmodo.es , envia la solicitud con su puerto origen, e ip origen y destino al router nat, que le va a asignar una entrada con un puerto random , supongamos 5001.El paquete sale con la unica IP de mi red (asignada por DHCP).Ahora cuando el paquete vuelve, como tiene una sola IP destino (la de mi red) el router nat tiene que mirar el datagrama que va a decir puerto destino :5001 y va a hacer el matcheo con la ip del celu (10.0.0.1) y el puerto original 3345

Protocolo de mensajes de control de internet(ICMP): Genera informes de error acerca de la capa de red, como por ejemplo “red de destino inalcanzable”.ICMP se considera parte de IP pero en el sentido arquitectónico se encuentra justo encima de IP.ICMP se encapsula en IP.
Mensajes posibles : Echo Request/Echo Reply(PING)
Destino inalcanzable
TTL expirado
Redireccion de Ruta
Address mask y timestamp

Destino inalcanzable :

- Host Inalcanzable**:** posible causa: no está encendido el host o no responde ARP
- Red inalcanzable : No tiene el router una ruta en la tabla de ruteo a esa red
- Puerto Inalcanzable: No hay proceso UDP en el puerto

IPV6: formato del datagrama:

IPV4 A IPV6

Algoritmos de enrutamiento: Un algoritmo de enrutamiento global calcula la ruta de coste minimo entre un origen y un destino utilizando el conocimiento global y completo de la red.Se los denomina **algoritmos de estado de enlace (**dijkstra**)**
Algoritmo de enrutamiento descentralizado: El cálculo de la ruta de coste mínimo se realiza de manera iterativa y distribuida (no tiene la info del resto de la red).Un ejemplo es el **algoritmo de vector de distancia** donde cada nodo mantiene un vector de estimaciones de las distancias a los demás nodos de la red.

Caracterısticas de Protocolos de DV

- **Corren un algoritmo distribuido, conocido como Bellman-Ford.**
- **Cada nodo intercambia información con sus vecinos (nodos directamente conectados).**
- **Intercambio de información periódico, aún sin que existan cambios.**
- **Ven la topologıa de la red desde la perspectiva de los vecinos.**
- **Estas derivan en las siguientes consecuencias:**
    - **Convergencia lenta.**
    - **Propensos a lazos (LOOPS).**
    - **Requieren menos memoria y capacidad de procesamiento.**

Caracterısticas de Protocolos de Link State

- **Todos los nodos comparten toda la información con todos:**
- **imagen común.**
- **Una vez que todos tienen la imagen de la red ejecutan un**
- **algoritmo centralizado por ejemplo el de Dijkstra -SFP-.**
- **No hay intercambio de toda la información (tabla de**
- **enrutamiento) de forma periódica**
- **Se informan cambios, actualizaciones y se lanzan paquetes**
- **para testear los enlaces.**
- **Alto consumo de bandwidth al principio, hasta que converge**
- **pero luego es mınimo.**
- **Capacidad de organizar en áreas.**
- **Ven la topologıa de la red completa en cada nodo.**
- **Consecuencias de las Caracter´ısticas:**
    - **Convergencia mas rapida, mas fácil de detectar nuevos caminos y descartar lazos.**
    - **No son propensos a lazos (LOOPLESS).**
    - **Requieren más memoria y capacidad de procesamiento porque deben almacenar la informaci´on de la red completa, y para cada destino, cada nodo, debe corre el algoritmo de forma independiente.**

Algoritmo de estado de enlace vs algoritmo de enrutamiento :
complejidad: El estado de enlace es más complejo , ya que tiene que conocer toda la red
velocidad de convergencia: el vector de distancias puede converger lentamente y puede loopear
robustez: a la hora de romper , el vector de distancia es más dañino que el estado de enlace , ya que este puede informar un camino mínimo mal a toda la red, mientras que el estado de enlace calcula sólo su propia tabla de reenvío
Otra forma de clasificar los algoritmos => **algoritmos de enrutamiento estático y algoritmos de enrutamiento dinámico**

Un **Sistema autónomo** es un grupo de routers que se encuentran bajo el mismo control administrativo , los routers del mismo AS ejecutan el mismo algoritmo de enrutamiento, los routers “pasarela” son los que se encarga de mandar los paquetes fuera del AS

RIP OSPF??

3 - CAPA DE ENLACE

Servicios que puede llegar a proporcionar :

- Entramado : envuelven un datagrama en una trama con su respectiva cabecera
- Acceso al enlace: Un protocolo de control de acceso al medio (MAC) especifica las reglas para transmitir una trama a través del enlace
- Entrega fiable:Asegura que se va a transportar cada datagrama
- Control de flujo: Los nodos situaciones en cada extremo de un enlace tiene una capacidad limitada de almacenamiento en buffer, por lo que hay que regular la velocidad de recibida para no desbordar el buffer
- Detección de errores : Puede decidir si la trama llegó con error o no (atenuación de señal y ruido)
- Corrección de errores: Te dice en qué punto de la tr4ama se produjo el error (y los corrige)
- Semi Duplex y full duplex: ambos extremos de un enlace pueden transmitir paquetes al mismo tiempo

En su mayor parte la capa de enlace se implementa en el **adaptador de red, o también llamado Tarjeta de interfaz de red**

Detección de errores :

- **Comprobación de paridad** : es la forma más simple, en un esquema de paridad par , el emisor incluye un bit adicional y selecciona su valor de modo que el número total

de UNOS sea par.Del lado del receptor tiene que haber una cantidad par de unos , sino significa que hubo un error

- **Suma de comprobación**: consiste en agrupar bytes de datos de 16 bits y sumarlos , luego calcular el complemento a1 y eso tiene que dar todos unos , si tiene algún cero es porque se produjo algún error.**En TCP y UDP la suma de comprobación se calcula sobre todos los campos (incluyendo cabecera y datos) mientras que en IP solo en la cabecera**
- **Comprobación de redundancia cíclica:** Tenemos una secuencia de datos de bits D, que el emisor quiere transmitir al receptor, primero entre ellos acuerdan un patrón de r+1 bits conocido como **GENERADOR (G).**Entonces para una secuencia de datos D , el emisor selecciona r bits adicionales R , y se los añade a D de forma que el patrón d+r sea divisible por G (que no tenga ningún resto), si es distinto de cero hubo un error.

Protocolo de acceso múltiple:

Usan un canal simple de difusión compartido , como más de un nodo puede transmitir información al mismo tiempo , se puede dar interferencias y colisión.
Para solucionar esto se tienen algoritmos distribuidos que indican a cada nodo cuando puede transmitir información.
Existen dos tipos de “enlaces” físicos:

1. Punto a Punto
2. Broadcast

Existen 3 tipos de protocolos MAC:

1. Canal subdividido/particionado: Divide el canal en pequeños pedazos , y los asigna a los nodos para su uso exclusivo
2. Acceso Aleatorio: Canal no dividido , pueden darse colisiones , y hay que recuperarse de estas.
3. Tomando Turnos: Los nodos toman turnos pero nodos con más por enviar pueden tomar turnos más largos

Protocolo MAC del canal subdivido :
Se comparte el canal eficientemente y equitativamente en alta carga
Son eficiente a baja carga : Hay retardo en acceso al canal , 1/N del ancho de banda es asignado aún si hay solo un nodo activo.

- TDMA :Time division multiple access - Acceso a canales en esas “rondas”.Cada estacion obtiene una ranura de largo fijo (largo= tiempo transmision del paquete) en cada ronda.Es la tecnologia que usa la telefonia movil GSM
- FDMA: Frequency division multiple Access, DIvide el enlace en frecuencias , a cada estación se le asigna una frecuencia fija , se pueden mandar paquetes por distintas frecuencias al mismo tiempo sin colisiones.

Protocolo MAC de acceso aleatorio:

Cuando un nodo tiene que enviar paquetes, los envía a la tasa máxima del canal , si dos lo hacen al mismo tiempo , estos colisionan y deben recuperarse.Son eficientes a baja carga , un único canal puede utilizar completamente el canal
Alta carga : ineficiencias por colisiones

- Aloha Ranurado : cuando un nodo obtiene una trama nueva a enviar, este transmite en proxima ranura.Si no hay colision , el nodo puede enviar una nueva trama en proxima ranura, SI hay colision , el nodo retransmite la trama en cada ranura siguiente con probabilidad p hasta transmision exitosa.
    - Ventajas: Es simple si hay un nodo activo , ya que permite transferencia a tasa máxima y el altamente descentralizado.
    - Desventajas : Si hay colisiones se desperdician las ranuras, quedan algunas desocupadas y que necesita sincronización

- Aloha no ranurado: Más simple, sin sincronización.Cuando se quiere enviar una trama se transmite inmediatamente, mayor probabilidad de colisión
- Carrier Sense Multiple Access(CSMA): se fija si el canal esta ocupado , si esta ocupado no manda nada, si esta libre manda,Igual se pueden dar colisiones , ya que el retardo de propagación hace que un nodo pueda creer falsamente que esta libre el enlace.
- CSMA/CD :se detectan las colisiones en corto tiempo, abortando las transmisiones.Es mas facil detectar colisiones en LANs cableadas que en inalambricas.
    - **Funcionamiento :**

1. Un adaptador puede comenzar a transmitir cuando quiera
2. Si detecta que otro está transmitiendo , no hace nada (sondeo de portadora)
3. Si está transmitiendo y detecta que otro esta haciendo lo mismo , entonces para y deja al otro (detección de colisiones).Envía una señal de bloqueo (JAM)
4. Antes de retransmitir el adaptador espera un intervalo de tiempo aleatorio pequeño

Backoff exponencial: las retransmisiones intentan estimar la carga actual de la red, cuantas más colisiones se den , más es el tiempo aleatorio a esperar antes de una nueva retransmisión
Protocolo MAC de “Toma de turnos”

- Busca lo mejor de Canal subdividido y Aleatorio.SE cuenta con un nodo maestro que les avisa a los demas cuando pueden enviar informacion , o los nodos se pasan iun token entre si que quien lo tenga en su poder podra enviar informacion al enlace

DIrección MAC: direcciones de 48 bits grabadas en la ROM de las tarjetas de red que sirven para conducir un datagrama de una interfaz a otra físicamente conectada (en la misma red)
ARP (address resolution protocol): Protocolo de L2 , a veces considerado L3, Protocolo “Helper” de IP, Mapea Dir Lógicas (IP) a direcciones Hardware (MAC), trabaja en forma dinámica , auto-aprendizaje sin configuración

Cómo funciona? (Mirar ppt ARP)
Tener en cuenta que se genera una trama broadcast (INCOMPLETA) cuando se quiere averiguar la MAC de una PC , y por otro lado el paquete ARP que tiene como MAC destino 00:00:00_00:00:00.hasta que alguien responda (ARP REPLY) con su MAC , y crea el paquete ARP con su MAC origen , a la MAC destino que me había preguntado originalmente
Luego terminado el ARP ya conociendo la MAC que quería , se arma la trama COMPLETA

Trama ethernet campos:

Campo de datos, Dirección de destino , Dirección de origen , Campo de tipo (que protocolo de la capa de red se esta usando),, comprobacion de redundancia ciclica y preámbulo

Es un servicio sin conexión no fiable ya que ya que envía de un adaptador a otro adaptador encapsulando el datagrama en una trama Ethernet y la manda de una a la LAN sin establecer nada de saludo triplete

Switch/Conmutador: El conmutador recibe las tramas de la capa de enlace entrante y las reenvía a los enlaces de salida.El conmutador es **transparente** para los nodos, osea un nodo dirige una trama a otro nodo y la envía a la red LAN sin saber q hay un switch en el diome.

Reenvío y filtrado: El filtrado es la función que determina si una trama debe ser reenvía a alguna interfaz o debe ser descartada, mientras que el reenvío es la función que determina las interfaces a la que una traba debe dirigirse y luego envía las trama a esas interfaces.Cumple estas funciones gracias a las **tablas del conmutador,** La tabla guarda por cada entrada :La mac de un nodo , la interfaz /puerto por el que fue incluida esa mac y la hora en la que se agregó
Funcionamiento: llega una trama a la interfaz *x* con la dirección MAC destino DD-DD-DD-DD-DD-DD.El switch va a buscar esa MAC en su tabla y hay 3 posibilidades

1. No encuentra esa MAC, entonces reenvía a todas las salidas/interfaces/puertos SALVO por la interfaz x que es por la que LLEGÓ
2. Encuentra esa mac y está asociada con la interfaz X también , entonces la DESCARTAn
3. Encuentra esa MAC y está asociada por una interfaz Y , entonces reenvía esa trama al buffer de salida de esa interfaz

Auto-aprendizaje de la tabla del switch:

Inicialmente la tabla está vacía , y por cada trama que recibe en una interfaz el conmutador almacena en su tabla

- ***Dirección MAC especificada en el campo dirección de origen de la trama***
- ***La interfaz de la que procede la trama***
- ***la hora actual***

Routers y switch :

- switch es plug and play mientras que el router hay que configurar los host que se conectan
- Switch ofrecen tasas de filtrado y reenvío altas
- switch no ofrece nada frente a tormentas de broadcast , mientras que los routers si
- En un router los paquetes no van a seguir nunca un ciclo , ya que el direccionamiento es jerárquico y si hay una ruta mal configurada esta el TTL
- Los routers a diferencia de un switch no están restringidos a un árbol de recubrimiento entonces utilizan las mejores rutas entre origen y destino ,y también hace que existan muchos enlaces entre Europa y América por ejemplo

En conclusión , para redes pequeñas de cientos de host , con switch alcanza , pero si tenemos miles de host habrá que mechar con routers para proporcionar un aislamiento más robusto de tráfico , controlar las tormentas de broadcast y utilizar rutas más inteligentes

VLAN: switch que permite definir varias redes LAN virtuales , sobre una sola estructura LAN física.Los host de una VLAN se comunican entre sí como si ellos estuvieran conectados al switch.El administrador divide los puertos/interfaces del switch en grupos, donde cada grupo forma una VLAN,
Las VLAN solucionan 3 inconvenientes de las LAN:

1. Falta de aislamiento del tráfico: (cuando se envían los broadcast por ejemplo)
2. Uso ineficiente de los conmutadores:
3. Gestión de los usuarios :

Control de enlaces de datos (DLC) Punto a Punto (Point to Point)
1 Tx, 1 Rx, un enlace: más simple que enlace broadcast:
no hay control de acceso al medio
no se requiere dirección MAC explícita
e.g., enlace telefónico
Características:
no corrección/recuperación de errores
no control de flujo
Entrega fuera de orden es OK
Estas características son de responsabilidad de las capas superiores
Estructura de la trama PPP:
Flag: delimitador (framing)
Address: nada hace (sólo una opción)
Control: hace nada; se pensó para múltiples campos de control futuros.
Protocol: protocolo de capa superior al cual entregar los datos (e.g., IP, etc)
info: datos de la capa superior
check: cyclic redundancy check (CRC) para detección de errores

Wireless LAN :
Características de IEEE 802.11
802.11 es una familia de estándares.
No es un reemplazo de las redes cableadas.
Similar a 802.3 (Ethernet).
Usan radiofrecuencias e infrarrojos (luego eliminados de
implementaciones del estándar).
Funcionan en bandas no licenciadas (depende del país).
Permite la movilidad de los usuarios.
De rápida implementación.
Acceso al Medio DCF - CSMA/CA
Carrier Sense Multiple Access / Collision
Avoidance.
Acceso al medio con contención, no determinístico, múltiples posibles accesos.
Algoritmo Distribuido.
Escucha en el canal,Si está libre, espera DIFS y
vuelve a escuchar.,Si sigue libre, Transmite.

Si está ocupado (en cualquiera de las 2 circunstancias) activa backoff.Luego de Transmitir, debe aplicar backoff también (luego de Ack o tmout).

Mencione 3 ejemplos de protocolos a nivel de enlace e indique si es de LAN o de WAN.
Ethernet (Lan).
Token Ring (Lan).
PPP Point to Point Protocol (Wan).

Hub (repetidor de capa física) :

- Los bits que ingresan por un enlace salen por TODOS los otros
- No hay almacenamiento y reenvío, ni CSMA/CD
- Hub de backbone interconecta segmentos LAN

Nota: para terminar de entender todo ver el ejemplo de los 24 pasos del libro de kurose página 479 (Benito)