

Apuntes - Redes IIC2333

CLASE 1

7 / 10 / 2024

Preguntas concurrentes

¿Porque las redes de datos? → Permitir que un proceso se comunique con otro

¿Donde se encuentra el otro proceso?

¿Por que medio? → Cable, aire?

Como se transmite? por el aire?

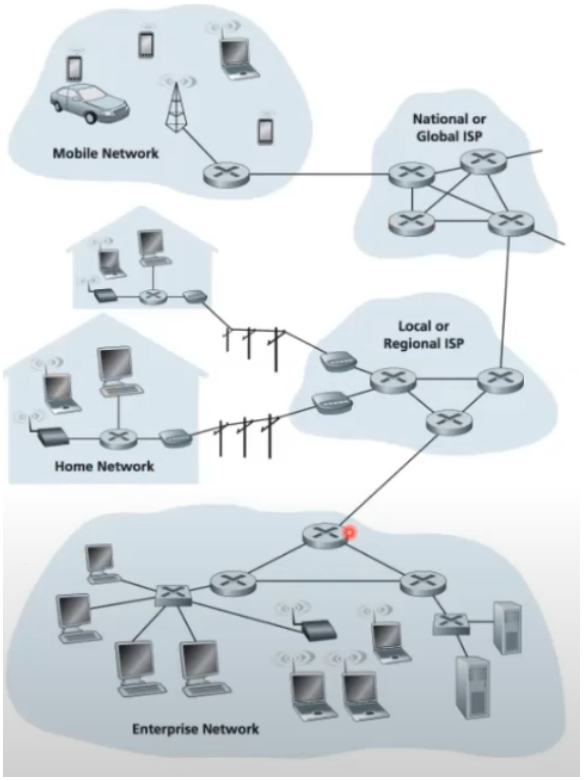
etc etc etc

La red mas grande: Internet

red como interconexion de redes mas pequeñas

hay estructuras de red distintas

Ejemplo de redes distintas:



Elementos de las redes

NODOS (HOST): Elemenots de origen y destinto de mensajes

1. Ej: Puede ser un dispositivo inalambrico, cableado etc (celular computador)

ENLACES: por donde se transmiten, enlaces por cables o inalambricas (datos moviles, wifi)

1. Cada enlace varia en velocidad, calidad etc

DISPOSITIVOS INTERCAMBIOS: Routers, Switches, son el medio donde se deciden encaminar la red, para hacer llegar el mensaje donde deba llegar

1. **Route**r: Para darle conexion a varios dispositivos → Lo que hace es conectarse al Switch
2. **Switch**: Comparte la conexion a varios dispositivos (es como una zapatilla)

PUNTOS ACCESO: Modem, Access point(inalambricos), Cell tower (conexion celular) → Esta ultima, es conectada desde el router para compartir conexion a varios dispositivos

- El modem lo que puede hacer es transformar una señal enviada atravez de los cables (una señal analogica) a una digital → esto Permite ser compartida con los elementos de una red en una casa por ejemplo
- Modem = MOdulador - DEModulador → Dispositivo que permite transforma una señal de un tipo a otro, investigar
- Puede hacer la transformacion inversa, para enviar a los cables la señal (digital → Analoga)

PROVEDORES DE ACCESO: ISP (internet Service Provider**) → Empresa que vende servicio**

ENTONCES?

En una casa, se envia informacion a la ISP (proveedor de internet), el modem transforma la señal digital a analoga para pasarla por los cables (cableado, fibra optica coaxial etc), y en ISP se hace el trabajo inverso, transforma la señal de los cables a digital para poder compartirla entre la estructura de red definida por isp

COMO SE TRANSMITE?

No se transmite la informacion byte a byte → Se transmiten unidades llamadas paquetes

Internet es una **RED DE PAQUETES**

Mediante routers, se envían paquetes a distintas direcciones

Reglas para intercambiar : **PROTOCOLOS**

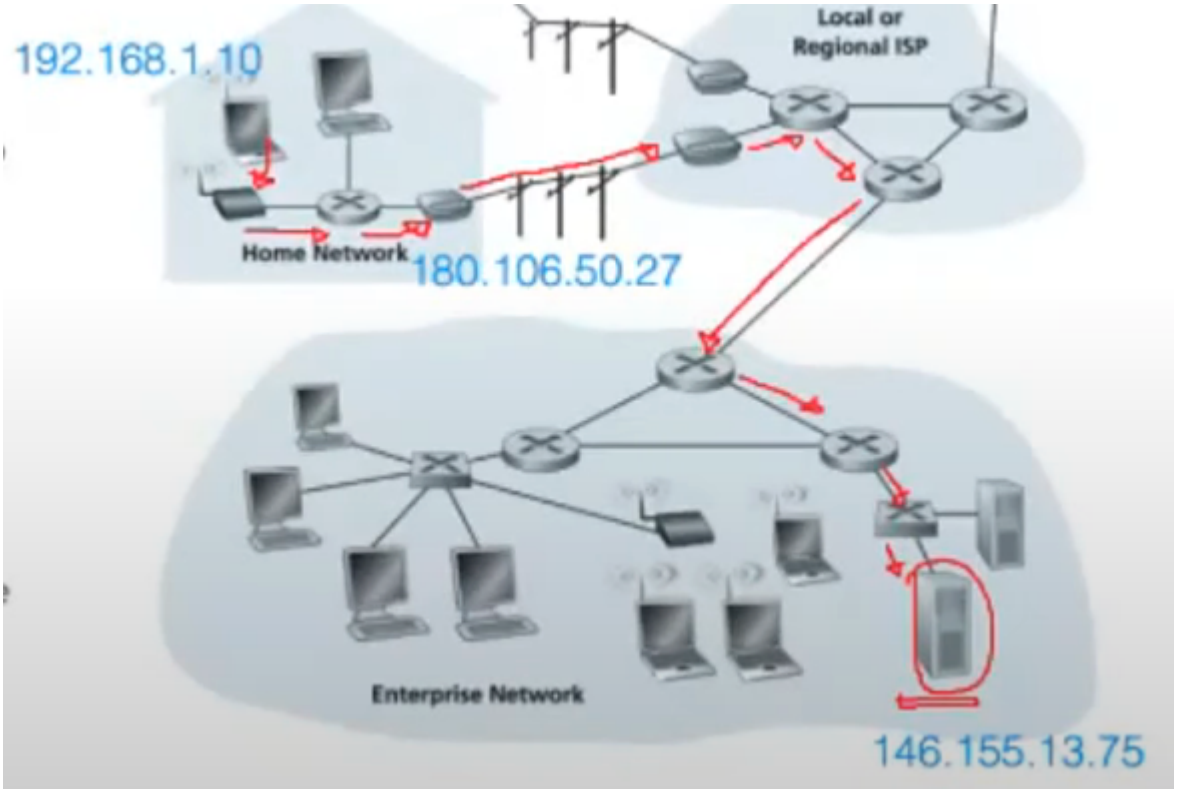
Existen 2:

1. **TCP**: Transport control protocol

2. **IP**: Internet protocol

El protocolo TCP, lo que busca es la integridad del mensaje, que el mensaje enviado llegue correctamente a su destino, ya que como se envian paquetes, quizas un mensaje puede estar siendo enviado particionado, por caminos distintos, afectando el tiempo de llegada a destino

Ejemplo ilustrativo



Finalmente, se busca con una direccion, y por ejemplo el computador con ip: 146.155.13.75 que esta levantando un servidor, (un proceso que espera, esta escuchando), a que le llegue algo, le enviamos a traves de los routers, un paquete

En nuestro caso: 192.168.1.10 seria nuestra ip local, de la conexion en nuestro hogar, y 180.106.50.27 seria la direccion publica con la que se identifica mi dispositivo

Los procesos finalmente, utilizan un **API (**application Programming Interface**) de internet,** que permite que el proceso interactue con una infraestructura (otros procesos en otra parte de la red!!!)

COMO COMUNICARSE?

Al igual que una conversacion humana (+hola -hola +¿tienes la hora? -2.00)

Las computadoras deben interactuar, de tal forma que se avise y se responda, quiero enviar algo, ok solicitud permitida,

+Tcp connection request -Tcp connection reply + get http://awl.com/kurose-ross - file

CLASE 2

9/10/24

Supuesto de que la clase 2 se vio la clase:

https://www.youtube.com/watch?v=_pyWVod9dzE&list=PLVTkEjeczcER2aOMHPIfaHCpTx61iKsoV&index=51

COMUNICACION END-To-END

Comuncacion entre **HOSTS:**

Hosts: Dispositivos que ejecutan programas

- Unos host acutan como clientes
- Otro como servidores

Clientes: Dispositivo liviano que solicita un servicio

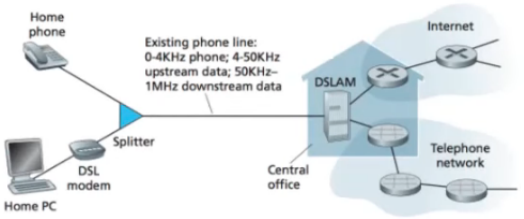
Servidores: En dispositivos poderosos que poseen servicios (data centers)

INFRAESTRUCTURAS DE COMUNCACIONES

- DSL
- CABLE
- FTTH
- ETHERNET
- FIBRA OPTICA
- Wi-Fi
- etc

DSL: utiliza la via telefónica para comunicarse

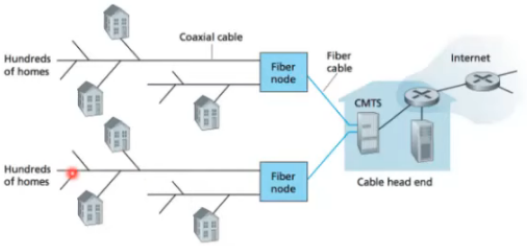
Transforma la señal digital, y la envia mediante via telefónica tradicional, transformando frecuencias de transmision



CABLE: Coaxial por ejemplo, cables de cobre con recubrimiento, esta es un medio compartido, a diferencia de dsl

Ventaja: Instalación mas facil → Extensión de cables

Desventaja: Medio es compartido, si varios usarios se conectan, provoca congestiones



CABLE FIBRA OPTICA (FTTH):

Se basa en cables de fibra optica que tienen una altísima tasa de transmisión

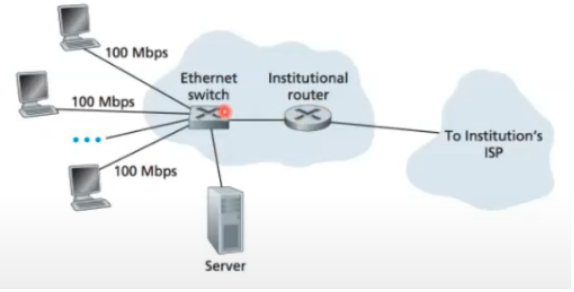
Estándares: desde 1Mbps a 1Gbps

ETHERNET: Cables de red, pares de trenzados de cobre, protocolo IEEE 802.3

Velocidad de transmision: 10Mbs 100Mbps 1Gbps 10 Gbps

Implementan la famosa **LAN (LOCAL AREA NETWORK)**

Router → Switch (comparte conexion) → → → Varios dispositivos (cada uno con su ancho de banda estandarizado)

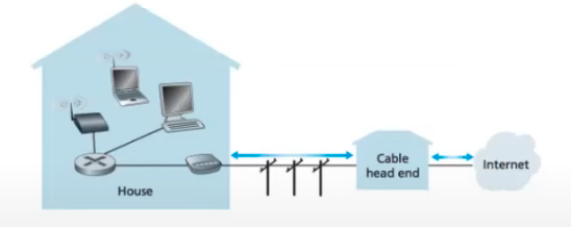


WIFI: simil al ethernet pero inalambcrica, de corto alcance

IEE 802.11 Velocidades: 54Mbps, 150Mbps

Implementan WLAN (wirless LAN)

Similar a esta **BANDA ANCHA MOVIL → Wide Area: 3G, 4G, LTE**



MEDIOS FISICOS

Transmitir bits atravez de un enlace fisico o inalambrico

Los medios pueden ser guiados o cableados: **solidos**

los no guiados son los de tipo broadcast: **inalambricos, satelital**

Solidos:

- **Par trenzado de cobre UTP RECUBIERTO** → (cable que no este recubierto puede tener interferencias y afectar los bits)
- **Cable Coaxial de Cobre** → Un unico cable protegido (asilado del ruido electromagnetico)

- **FIBRA OPTICA →** Usa haz de luz

Medio inalambricos → Al ser emitida, esta expuesto a las interferencias que atenuen

Porpagacion multicamino: Señales que llegan por multiples caminos

FUNCIONAMIENTO DE RED

Si quiero transmitir L bits sobre un enlace que soporta R bits por segundo

Entonces puedo transmitir a una velocidad de L/R

Formula Importante: Demora de trasnmicion : L bits sobre enlace / R (bits/sec)

La red de **INTERNET** es una red de Paquetes, transmite paquetes de un destino a otro, con etiquetas origen y destino a una Infra**Estructura**

Esta estructura es capaz de mirar los packetes, examinarlos y reenviarlos (reenrutar)

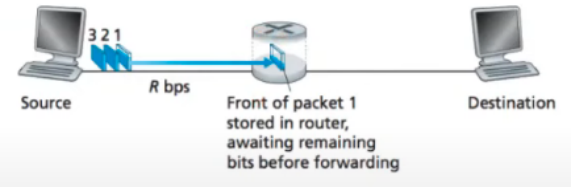
Estan configurados con tablas (mapeo)

EJ : "Todos los paquetes que empiezen con 120.algo ... mandalos por este camino"

Cada dispositivo router es capaz de tomar cualqueir paquete, y rutearlo al camino correcto

PACKET SWITCHING

1. ROUTER: Almacena Paquetes
2. Examina Paquetes y determina punto de envio (**FORWARD)**
3. Usa Tablas de reenvio (**FORWARDING TABLES**



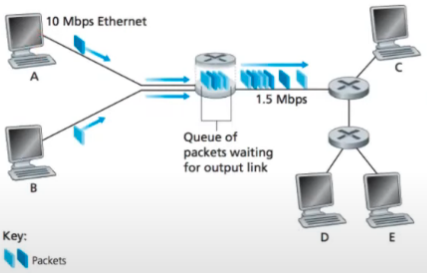
Estos routers tienen una memoria (limitada), si la cola de procesos se llena, decimos que esta congestionado, pero si tenemos varios buffers, podemos enrutar el camino atravez de distintas rutas

Si tengo mas routers, es mas fácil de manejar la congesiton !!!

Tasa de transmisión determinada por el enlace más lento

- *Router* puede sufrir **congestión**
- Si hay N enlaces iguales en la ruta, el tiempo de transmisión de origen a destino (*end-to-end delay*) es:

$$d_{end-to-end} = N \times \frac{L}{R}$$



N Saltos (en cada enlace)

La competencia, es la red (telefonica) o red de **INTERCAMBIO DE CIRCUITOS**

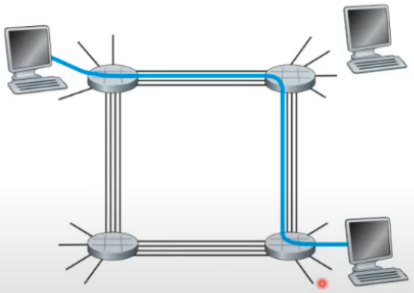
CIRCUIT SWITCHING

CIRCUIT SWITCHING

Establecimiento previo de conexión entre origen-destino

Ancho de banda garantizado durante comunicación

- Enfoque usado por comunicación **telefónica**
- Sensible a saturación ante conexiones simultáneas
- Enlace de 1Mbps, con 4 conexiones simultáneas, cada una recibe hasta 250kbps
- Si hay N enlaces iguales en la ruta, el tiempo de transmisión de origen a destino (*end-to-end delay*) es: $d_{end-to-end} = N \times \frac{L}{R}$



En vez de enviar paquetes, se buscan caminos para elegir rutas

Cada router tiene entradas / salidas

Cada vez que llamo, la conexion no es instantanea, se genera primero el circuito y luego con ese delay, se establece la conexion

- Es muy **SENSIBLE A CONGESTIONES ,** ej: Cuando hay terremotos, se saturan estas vias

Software de Redes

Hardware soluciona mitad de la tarea

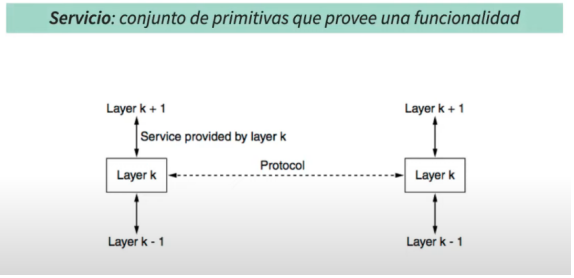
- Redes se organizan en base a un modelo de **CAPAS**
- Las Capas se comunican usando **PROTOCOLOS**

Este modelo de capas se ven a partir de un medio fisico

- Cada capa atiende a un problema especifico
- Y cada capa funciona utilizando un protocolo específico

Los mensajes pueden ser modificados en cada capa → Agregando headers o modificando todo el mensaje

- Cada capa entrega servicios a una Capa Superior
- Y solicita servicios a una capa inferior



Modelo OSI (modelo de referencia)

Define las reponsabilidades para cada capa

No define estandares ni protocolos ni implementaciones

A partir de esto, una implementacion practica es la **STACK TCP/IP**

TCP/IP

Con respecto al modelos OSI: es mas compactada, y tenemos solo 5 capas:

1. Applicacion
2. Transporte
3. Network
4. Link
5. Physical

CAPA APLICACION

Funciona con los procesos, cuando vemos el tema de sockets, estamos hablando de la capa aplicacion, vamos a pedirle servicios a la capa de transporte

Algunos de esto son los protocolos que intercambian **MENSAJES**

ej:

- HTTP
- SMTP
- FTP
- DNS
- IMAP
- POP

Capa presentacion y sesion (solo en modelo OSI) → Son compactadas en Applicacion (para modelo TCP/IP)

Presentacion de mensajes y sesion → Es la logica en el envio

Cada capa aplicacion es encargado de:

- Codificar
- Encriptar
- Autorizar
- etc

CAPA TRANSPORTE

Transporta mensajes entre **ORGIN Y DESTINY**

- TCP: + seguro
  - Sevicion connection-oriented
  - Garantias de entrega
  - Control de flujo
- UDP: + rapido
  - Servicion connection-less
  - no gratiza entrega

CAPA RED

Se ocupa de transmitir **SEGMENTOS DE MENSAJES** entre un par de **HOSTS**

Se preocupa de la **RUTA** de donde se envia el mensaje → (Routing del segmento)

Intermediarios: **ROUTERS**

- IP (internet protocol)
  - Hardware routers
  - ipv4, ipv6
- ICMP: mensajes de control (ping, echo, trace)
- IGMP, OSPF: Protocolos entre routers

CAPA ENLACE

Capa se preocupa del salto entre un dispositivo a otro

Se transmiten en cada par de hosts, de acuerdo a las **Caracteristicas del enlace**

PAQUETES de capa de enlace se conocen como **FRAMES**

- Transmision confiable
- Proveen Control Errores
- Control de acceso al medio (**MAC**)
- Ethernet, IEEE 802.3 → Medio cableado
- Wi-Fi → Medio Inalambrico
- Hardware: **SWITCHS** y hub's

Frames: Unidades de bytes, se envian atraves de un enlace en especifico de un dispositivo a otro dispositivo

CAPA FISICA

Capa mas inferior de todas

Se encarga de transmitir bits

A traves de cables, fibras, inalambrico

Como se traslada, como se transmite una señal inalambrica a traves de una antena, etc

ENCAPSULAMIENTO

Tecnica para envio de mensajes

El mensaje se divide en varios segmentos

La capa de trasnporte les pone headers, info relevante para capa de transporte

La capa de transporte se lo pasa a la capa de red, como **PAQUETE de Red o Datagrama →** Se le agrega un header de la capa de red

La capa de enlace, le agrega su header (link header)

Al final la capa fisica se encarga de transmitirlo

Cuando el paquete fue enviado y llega a un switch, al ser un dispoisitvo de **CAPA DE ENLACE**

SOLO TIENE 2 CAPAS: Enlace y fisica

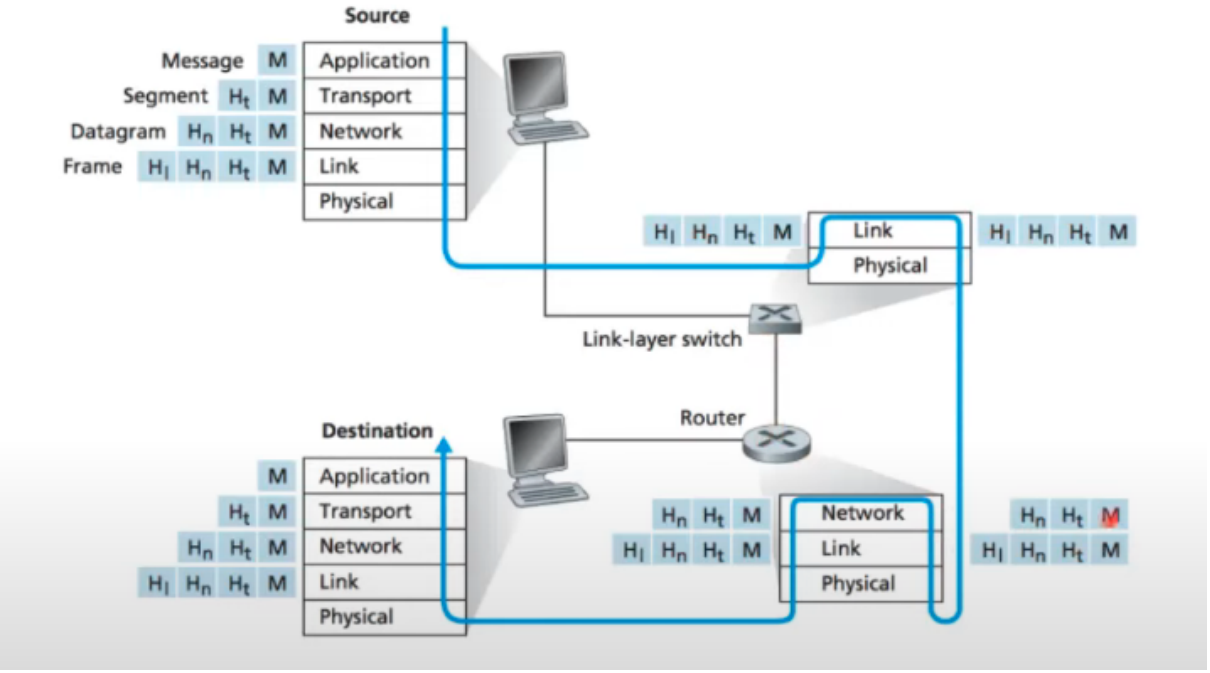
El enlace toma el mensaje, se lo transfiere a la capa fisica y este se lo envia a la siguiente dispositivo

Llega al router, **EL ROUTER Tiene 3 CAPAS: Red , Enlace y Fisica**

Aca llega a la parte fisica, lo pasa a la parte de enlace, en la parte de enlace se le **QUITA EL HEADER,** Y en la capa de red, analiza su **HEADER, y decide que hacer → Borrar modificar etc**

Devuelta al las capas hacia abajo, hasta llegar a enviar al computador de destino, y subir en capas hasta la ultima

EJEMPLO VISUAL



CLASE 3

14/10/24

Resumen clase pasada:

MODELOS TCP/IP:

5. S.Aplicacion → Comunicacion de procesos
4. Transporte → Transporte de msj de origen a destino
3. Red → Routing

2. Enlace → Transporte entre dispositivos y acceso al medio
1. Fisico → Transporte de bits por medio misico

Cuando comunico los procesos, no me refiero a que hayan cables directo entre ellos

En el mensaje envio info adicionl (headers) → Como interpretar el mensaje

Ej: En el header de un http digo estoy intentando hacer un post, un get , un delete , etc...

CAPA APLICACION

Capa únicamente implementada en los **EMISORES Y RECPETORES de Mensajes**

PROTOCOLOS

DNS: domain name system

HTTP: hypertext transfer protocol

FTP: File transfer protocol

SSH: secure shell

etc...

Pregunta Prueba: estoy enviando un archivo html, a que capa pertenece ese archivo de protocolo → **CAPA DE APLICACION**

Si me piden especificamente cierta accion que esta ejec un proceso → CAPA APLICACION

Hay protocolos que se construyen sobre ciuertas arquitecturas

Protocolos:

TIPOS ARQUITECTURA

Cliente/Servidor → ej: HTTP esta construido aqui, FTP, IMAP, etc...

P2P → ej: BitTorrent

La arquitectura cliente/servidor → El servicio es proveido por un solo computador → **FORMA CENTRALIZADA ,** si se cae el servidor se cae todo el servicio, a diferencia de p2p

Generalmente no se hostea en solo una maquina, probablemente se usen muchos mas

(que pasa si se muere un pc, moriria toda la applicacion → Mejor distribuir)

P2P: Ninguno va a tener una jerarquia sobre el otro

Los Dispositivos van a estar conectados entre si

Esto puede traer problemas de seguridad

Mi mensaje va con mi direccion IP → Esto es peligroso, en la RED pueden tener tu ip

- Es mas dificil que se caiga! → Hay que bajar todos los servicios para botarla

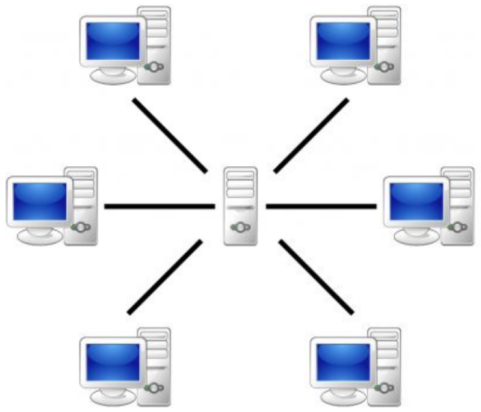
Es ICP Friendly → Por ej: Cuando contratamos servicio internet (nos dice velocidad de bajada)

La velocidad de subida generalmente es mas lenta que la bajada

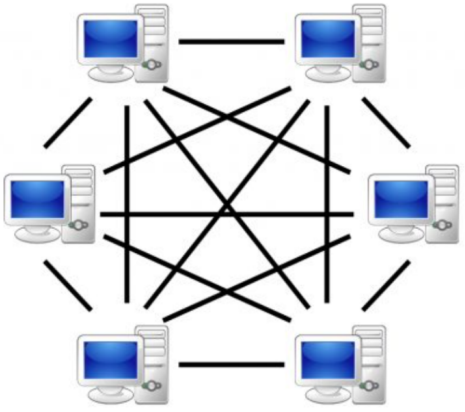
- En una p2p, por ejemplo torrent, hay mas velocidad de bajada que de subida, de cierta manera podria aprovecharme de la velocidad de subida de varios dispositvos en la red.

Sin embargo, otros usuarios pueden fekear ip, y hacerse pasar por computador, haciendo cosas malas

RESUMEN GRAFICO



Server-based



P2P-network

¿Como un proceso logra comunicarse en la RED?

SOCKETS

SOCKETS: Elemento de software que interactúa con la red

- Puerta de entrada/salida al sistema
- **Utiliza servicios de capa Transporte:** TCP o UDP
- Solo existe en hosts de **origen y destino**
- IDentificador conexion → **IP (identificar dispositivo) y PUERTO (identidifica procesos)**

Servicios

Archvios, Emails WebDocuments → Priorizan no perder data (TCP)

Streaming, telefonia, etc → Priorizan velocidad (UDP)

Puertos:

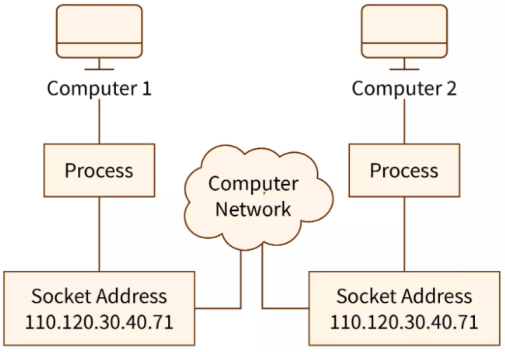
HTTP: Por default 80

SSH: Usa el puerto 22

HTTPS: 443

- Por ejemplo https://www.google.com:443 → Como 443 es el puerto de https → Nos recargara la misma pagina, otro numero no sera identificado porbablemente

Socket: elemento de software que interactúa con la red



- Socket es una puerta de entrada/salida al sistema
- Socket utiliza servicios de transporte (capa inferior)
- Socket **es una interfaz** para entre el programador y la red
- Programador puede controlar parámetros del socket
  - Tipo de servicio de transporte (TCP / UDP)
  - Direccionamiento del host remoto (IP + puerto)

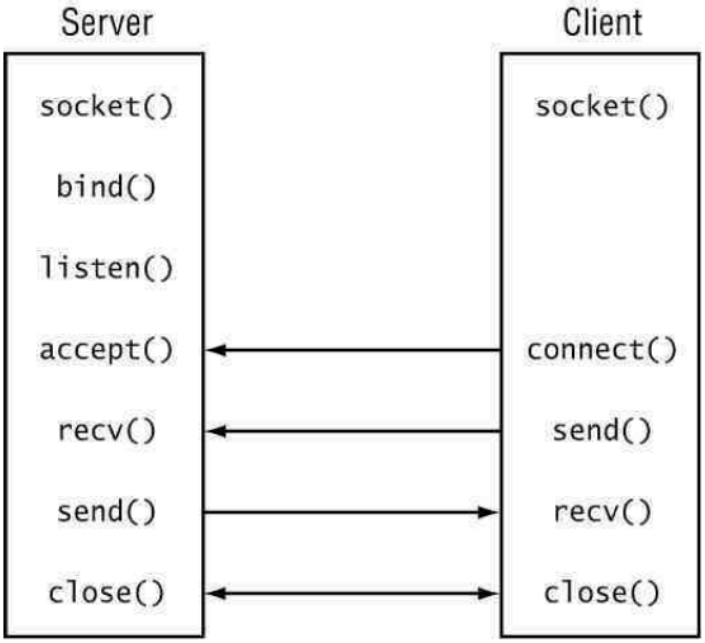
TCP: Protocolo de enviado lento pero seguro (por ej: Cuando estoy enviandome archivos)

UDP: Protocolo de enviado rapido pero no confiable (no asegura llegada completa) (por ej: Streaming de video)

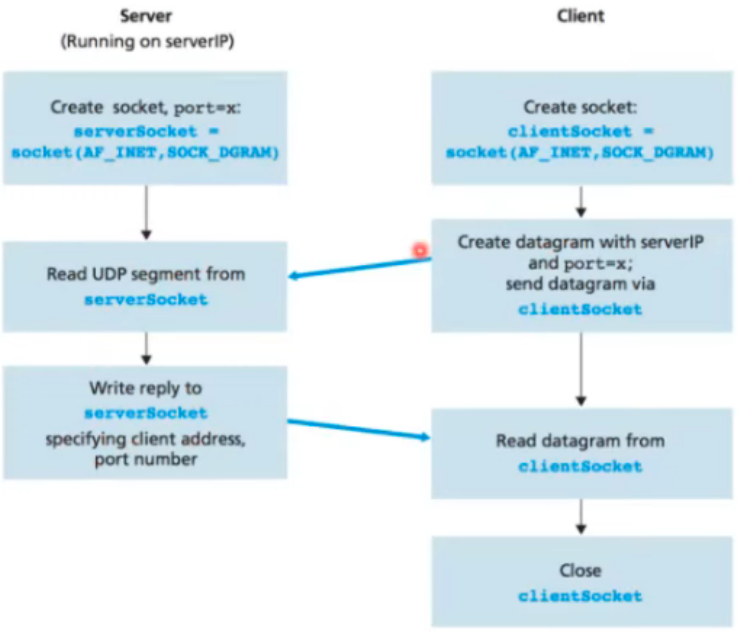
APLICACION CON SOCKETS (ejemplo)

- Cliente lee una línea desde la entrada estándar (stdin)

- Cliente envía línea al servidor
- Servidor recibe línea
- Servidor convierte caracteres a mayúsculas (uppercase)
- Servidor envía línea modificada al cliente2
- Cliente2 recibe línea
- Cliente2 escribe la línea en la salida estándar (stdout)



Conexión UDP



En Resumen: Existe un envio de mensaje, donde se entrega

- el socket,
- struct con datos destino →(Ip destino y puerto),
- buffer con los datos,
- flags

Socket UDP

Si hosteo un servidor con un socket UDP

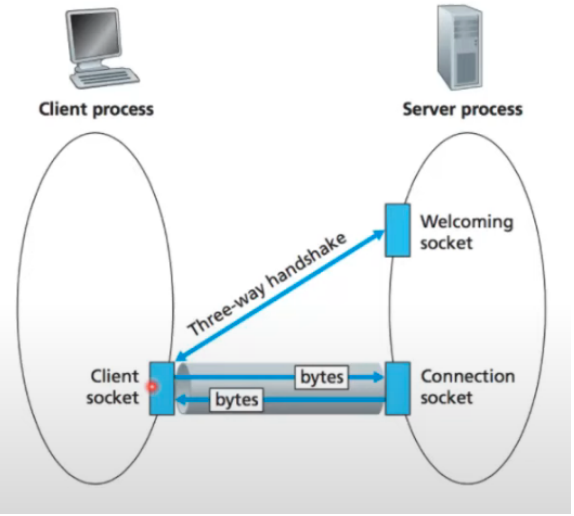
Tendre un **socket comun abierto para todos los clientes** → Esperando la enviada de informacion

- **CONECTION LESS**
- Unico socket para clientes
- Identificador clietnes (IPcliente, PuertoCliente) → Se de donde viene la informacion
- + (IPsrc, Portsrc, IPdestinto, Portdestino) → Saber a donde mandarlos

de sist Op: *Para enviar y recibir informacion necesitaria 2 Threads (porque son bloqueantes), uno para enviar y otro para recibir}*

Conexion TCP

Requiere un proceso anterior → **ESTABLECER CONEXION : HANDSHAKE**



Existe un socket **Welcoming**, el cliente le habla para pedirle conexion, acuerdan la tasa de transferencia de datos y otras cosas, y luego de esto, le entrega un socket especifico para ti

Con esto el socket se transforma en un tubo, por donde el clinete le manda datos y recibe

Una vez que se establece la conexion , los procesos pueden enviarse datos usando

SEND() Y RECV()

Enviar un mensaje atraves de un socket, tiene un buffer de datos a enviar

- **NO NECESITA ESPECIFICAR DIRECCION!** → Dado que ya esta establecida la conexión

Pregunta de prueba: Diferencia entre socket tcp y udp, el tcp no envia la direccion y udp siempre

Socket TCP

- **CONECTION ORIENTED** → TCP HANDSHAKE
- Por cada cliente → **Un puerto** (cada cliente va a ser escuchado por un socket en un servidor)

Pasos de conexion

Establecimiento de conexión: TCP Handshake (saludo de manos)

1. Requiere establecer **conexión** antes de enviar
2. Cliente inicia solicitud de conexión en un puerto conocido del servidor
3. Servidor debe tener un socket en ese puerto
4. Servidor debe **aceptar la conexión**
5. Al aceptar la conexión, servidor **asigna nuevo socket** para la conexión

Al nivel de C, ambos (cliente/servidor) inicializan un Socket, el servidor tendria que aceptar la conexión (TCP: HANDSHAKE)

En resumen (pregunta de prueba)

UDP: Conexion común (un puerto para todos)

TCP: Conexion única (un puerto por cada conexion)

PROTOCOLOS

1. HTTP

Envio de informacion de **texto enriquecido**

- Usa arquitectura **CLIENTE/SERVIDOR**
- Cliente: Solicita y revibe archivos
- Servidor: Recibe solicitudes y envia archivos

- **Es TCP:**
  - (no de capa de transporte), si no de capa de **APLICACION**

Si usamos HTTP: Por debajo estamos usando protocolo TCP

REQUESTS:

HTTP Funciona a base de requests:

- Formato
  - GET /slides/disk.html HTTP/1.1
  - Host: iic2333.ing.puc.cl
  - Connection: close

  Prueba eso en la terminal →

  telnet iic2333.ing.puc.cl 80

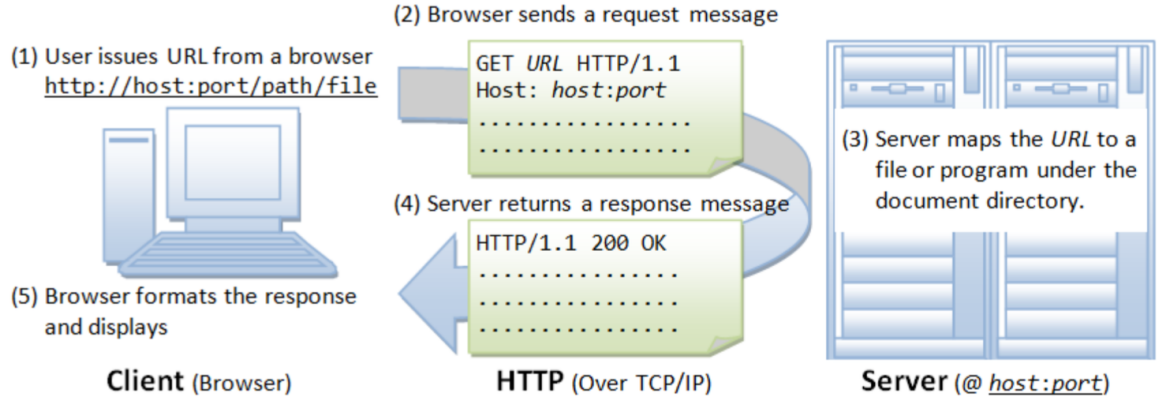
  GET / HTTP/1.1

  Host: iic2333.ing.puc.cl

  Connection: close

  deberia devolver la landing page del servidor del curso (la pagina)

- Siempre que pidas una URL (Uniform Resource Locator) desde el navegador para obtener un recurso web mediante HTTP. El navegador convierte la **URL** en **un mensaje de solicitud** y lo envía al servidor HTTP.



CODIGOS DE RESPUESTA

Códigos

- 1xx. Mensaje de información.
- 2xx. Mensaje de éxito
  - 200 OK
  - 206 Partial Content
- 3xx. Mensaje de redirección
  - 301 Moved Permanently
  - 307 Temporary Redirect
- 4xx. Mensaje de error del cliente
  - 400 Bad Request
  - 403 Forbidden
  - 404 Not Found
  - 405 Method Not Allowed
- 5xx. Mensaje de error del servidor
  - 500 Internal Server Error
  - 501 Not Implemented
  - 502 Bad Gateway
  - 503 Server Not Available

HTTP/1.1 200 OK

Connection: close

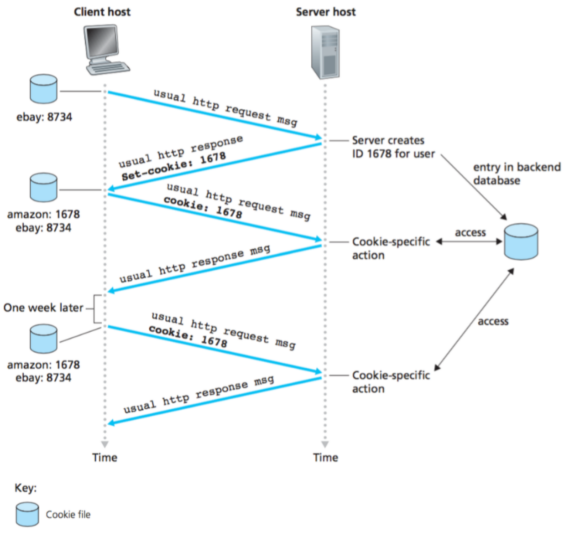
Date: wed Jun 15 01:20:33 CLT

Server: .....data

Es Stateless → Sin **Estado**

No tiene capacidad de recordar solicitudes anteriores, para ellos se usan:

- **COOKIES** (informacion de la accion, se almacena en tu navegador, me compromento con el servidor, cada vez que me conecte, le mando informacion mia al servidor)
- Un ejemplo es por ejemplo: Me da un Id, me comprometo a cuando me conecte, lo haré con esa Id, para que sea mas facil acordarse de mi



Web Caching: Cachear tu informacion, para que llegue mas rapido

- Cuando el contenido no cambia (estatico) → Puede ser almacenado en **Servidores Intermedios → WEB PROXY**
- **Redes de Cache** → CDN: content delivery network): ej: CLOUDFORM: Cachear informacion sobre un servidor web

2. FTP

- **TRANSFERENCIA DE ARCHIVOS** entre sistema **Local y Remoto** (Filezile funciona con FTP)
- Utiliza **TCP**
- Funciona con **Inicio sesion → Contraseña y usuario**
- Usa 2 Conexiones (2 Sockets por 1 Cliente)

- FTP crea 2 conexiones entre las computadoras,
  - una conexión para los comandos y respuestas llamada conexión de control
  - una segunda conexión para transferencias de datos llamada conexión de datos.

SMTP, POP, IMAP → CORREO ELECTRONICO

ENVIO → **SMTP: Servidor de salida**, encargado de enviar los correos y se depositen en el servidor

RECIBO →

- **IMAP: Protocolo Nuevo** , Almacena emails en el servidor (msj y estado) (cualquier dispositivo que pueda iniciar sesión, tenga acceso a **todos** los mensajes)
- **POP3: Descarga los mails** en un **UNICO DISPOSITIVO!**
  - Emails **NO SE ALMACENAN** en el servidor (si no de manera local)

Clase 4

16/10/24

DNS

Sistema que permite Traducir un Dominio con Direcciones

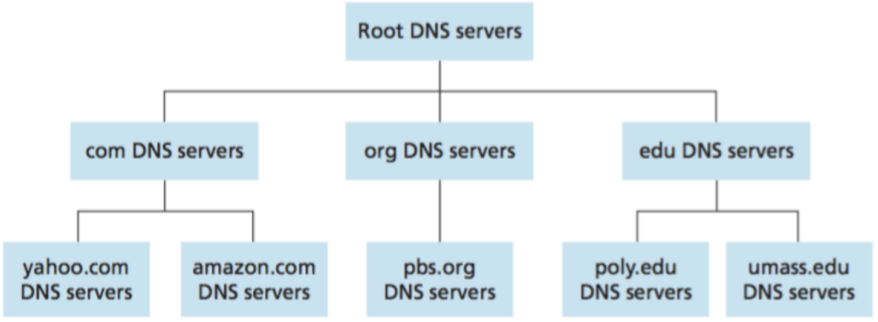
EJ

1. Cliente ejecuta **PROCESO Cliente DNS**
2. Browser extrae la URL: www.google.com y lo pasa al CLIENTE **DNS**
3. Cliente DNS manda **QUERY** por el dominio www.google a **SERVIDOR DNS**

4. Cliente DNS recibe respuesta con la IP del dominio pedido
5. Cliente le pasa respuesta al browser
6. Cliente **HTTP** inicia **CONEXION TCP a la IP www.google.com**

DNS utiliza una estructura de jerarquias y dominios

Base de datos distribuida y jerárquica



Jerarquia:

1. Root Servers
2. TLD Servers
3. Domain Servers → .com .org. edu ...

Servidores **DNS**, almacenan **MAPPINGS**

- Si no conoce mapping:
  - Lo consulta a un servidor de **Mayor jerarquía**
- Eventualmente pueden llegar a la RAIZ

Tipos:

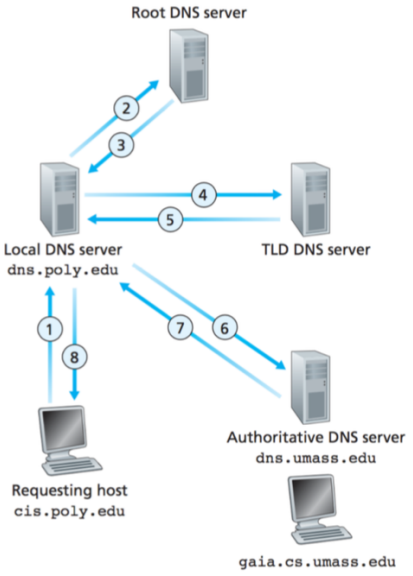
1. **Autoritativos** → administran el mapping → **POSEEN LAS DIRECCIONES**

2. **No autoritativos** → manientienen temporalmente el mapping → **TIENEN QUE BUSCARLAS**

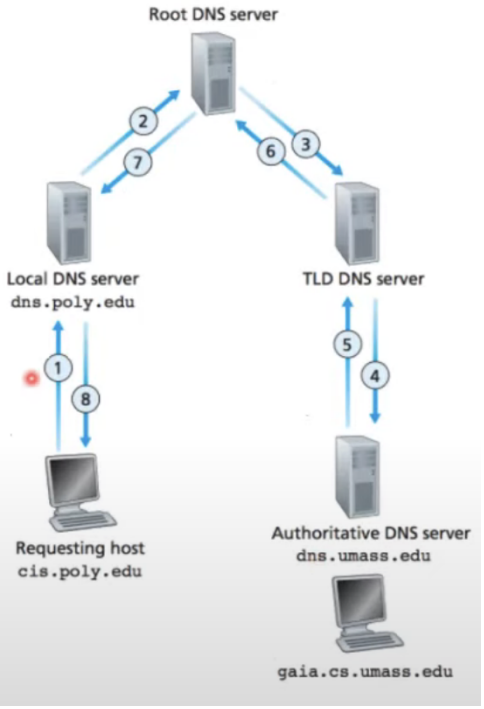
1. **Servidores DNS**: Son los encargados de almacenar y proporcionar la información necesaria para la **traducción de dominios** a direcciones IP.
2. **Resolvidores DNS**: Son los intermediarios que consultan los servidores DNS en nombre del usuario y devuelven la IP correspondiente.
3. **Zonas DNS**: Se refieren a divisiones en el espacio de nombres, cada una administrada por un servidor específico.

TIPOS DE CONSULTAS

- **Consultas iterativas**: retornan la respuesta.
  - Ejemplo el servidor (1,2) de la imagen envían respuesta (completa o parcial) al origen



- **Consulta Recursiva**: no tienen la respuesta
  - Entonces reenvían la consulta.
  - Ejemplo (4,6) reenvían la consulta.



La iterativa no delega completamente el trabajo

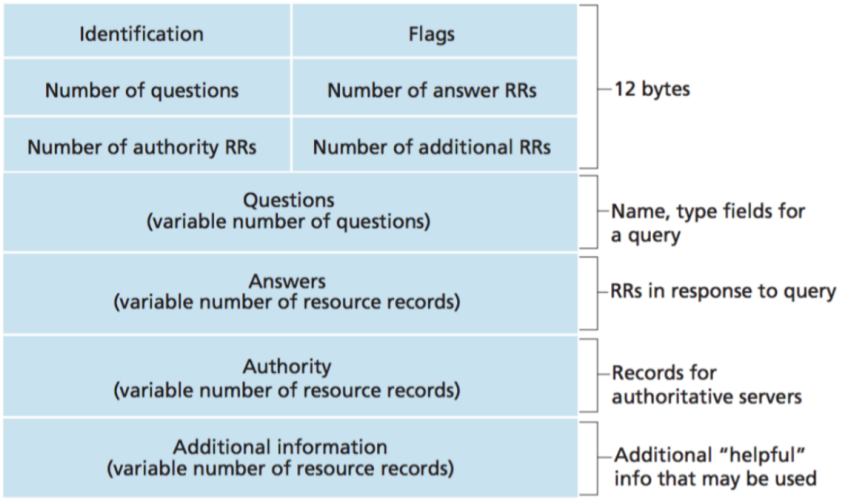
La recursiva si, el servidor local espera que llegue la respuesta completa de una lista de servidores

Ej: COMPUTADOR (Cliente) → LocalServer → RootServer → TLD Server → ....

IMPORTANTE

- Algunas veces preferiremos las **consultas iterativas** por sobre las recursivas
- Los servidores raiz (root), no suelen ser tan consultadas dado que mas abajo, ya hay respuestas cacheadas (en servidores mas abajo en jerarquia)
- Si las respuestas no estan cacheadas - > Se demora el tiempo de respuesta

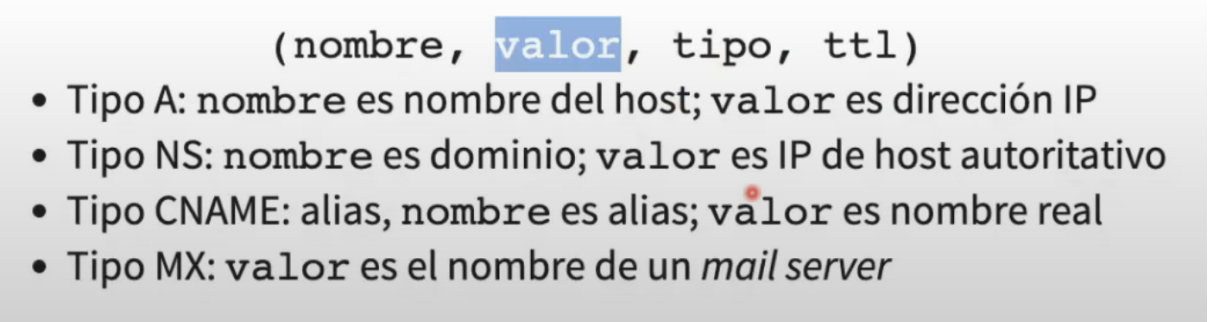
ARGUMENTOS (Records)



LAS RESPUESTAS TIENEN LA SIGUIENTE FORMA

(nombre, valor, tipo, ttl)

- **A:** Domain → IP :
  - ej: (relay1.bar.foo.com , 145.37.93.126, A)
- **NS**: Sirve para buscar servidor **autoritativo**
  - ej: (foo.com, dns.foo.com, NS)
- **CNAME**: Para nombres **alias**
  - ej : (foo.com, relay1.bar.foo.com, CNAME)
- **MX:** para saber a donde va a parar los mail de cierto dominios



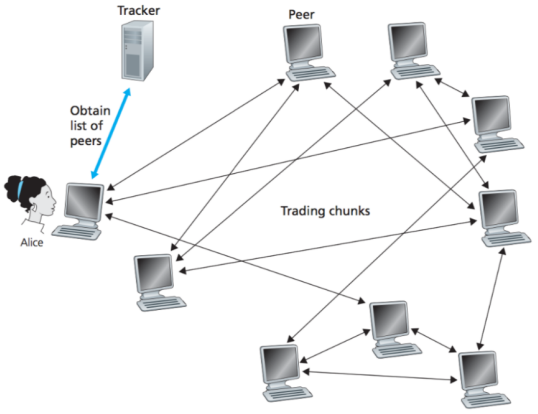
De pregunta en clases → ¿que pasa cuando tengo mas de un servidor (por la ip)?

Si tiene mas de un servior para hostear pagina web → Probablemente nos conectemos a un solo servidor, y ese nos deriva a otros servidores

Tambien generalmente nos conectamos a un **PROXY** → servidor mas cercano (con replicas de la info original de la pagina)

RED P2P

- Redes entre pares de clientes (**NO DEPENDEN DE UN SERVER**)
  - Mejor escalabilidad
  - Deben maneja pertenencia y abandono de miembros



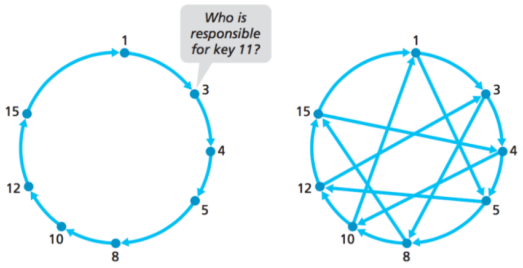
Cuando me quiero conectar a una red p2p: **consulto al servidor un tracker** → Como conectarme a los otros clientes que estan participando de la red

Por ejemplo otro protocolo BitTorrent → Si quiero descargar un juego, cada uno de los host, me puede prestar una parte del juego (archivos por pedazos) para abarcar juegos mas grandes por ejemplo

Se construye una red a base de una **HASH TABLE**

Método de direccionamiento para redes P2P estructuradas
Cada miembro posee una identificación única

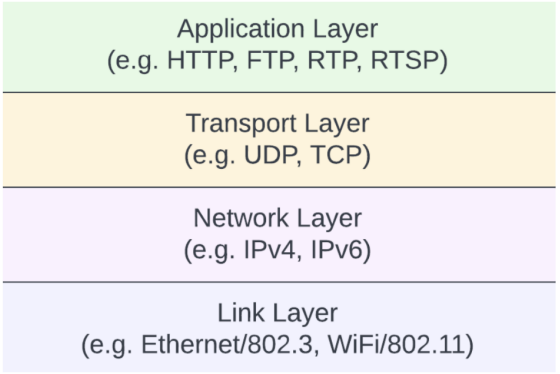
- Se construye una overlay network entre miembros
- Miembros se encarga de enrutar mensajes



CONTENIDO 10 → CAPA DE TRANSPORTE

CAPA TRANSPORTE : Transmision
Origen - Destino

- Se encuentra entre la Capa de APLICACION y la de NETWORK
- NO VIVE en los **SERVIDORES intermedios,** el objetivo es que llegue de un emisor a un receptor



Existen 2 Agentes importantes

1. **Emisor**: Recibe un mensaje de **CAPA APLICACION,**
   - **lo** divide en **SEGMENTOS**
   - **Solicita a capa de RED** que **ENVÍE** los segmentos al **Receptor**
2. **Receptor**: Recibe Segmentos,
   - los Ensambla para formar mensaje

- Lo entrega a la **CAPA APLICACIÓN**

PROTOCOLOS

- **TCP:** Entrega confiable y ordenada. Control de flujo.
  - Establecimiento de conexión.
- **UDP:** Entrega no confiable y desordenada. Más rápida.

- El protocolo solo lo ejecuta origen y destino (NO INTERMEDIARIOS como ROUTERS, etc)
  - Se Transmiten Segmentos
    - (información que quiero enviar desde la capa de aplicacion por ej un Html con cierto header que me va a llevar a ejecutar ciertas reglas que tenga en el protocolo)

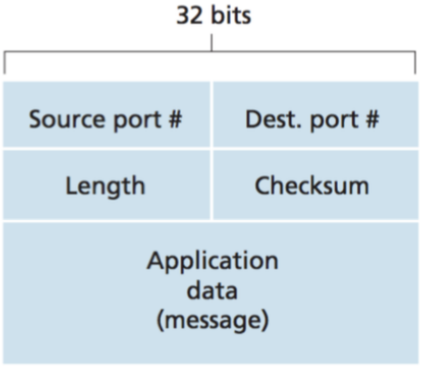
Mensaje se divide en segmentos → Segmentos se transmiten en paquetes IP.
¿Cómo identificar mensajes a distintos procesos?

- **Puerto** permite identificar procesos dentro del nodo emisor o del nodo receptor.
- **Puerto** permite multiplexar mensajes

SEGMENTOS

Segmentos se transmiten en paquetes IP, podemos identificarlos mensajes de sitntos procesos con Puertos

- Puerto → Permite identificar procesos del nodo emisor o receptor
- Multiplexar mensajes



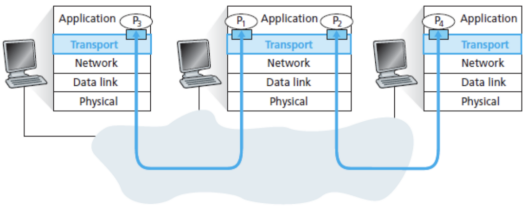
MULTIPLEXION

Los mensajes se envía a traves de **SOCKETS**

Cada socket es asociado a 1 puerto especifico

Un Host pueden tener múltiples puertos abiertos

Un Puerto NO PUEDE estar asignado a multiples sockets

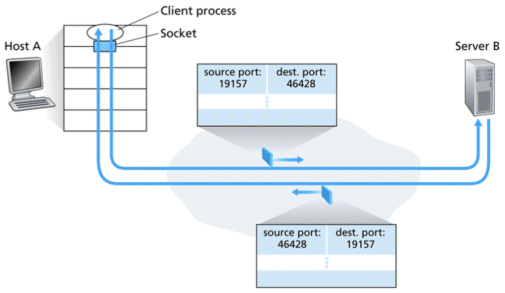


ej: *Si quiero conectarme a una pagina http, necesito abrir un socket, si quiero hacer otro cosa, abro otro socket, tengo que saber que voy a hacer para saber el puerto a abrir*

SIN CONEXION HANDSHAKE → UDP

Sin Conexión
Segmento UDP identificado por: ⟨IP destino, puerto destino⟩

- Receptor observa el puerto de destino
- Pasa el segmento al proceso asociado al puerto de destino



Identificador:

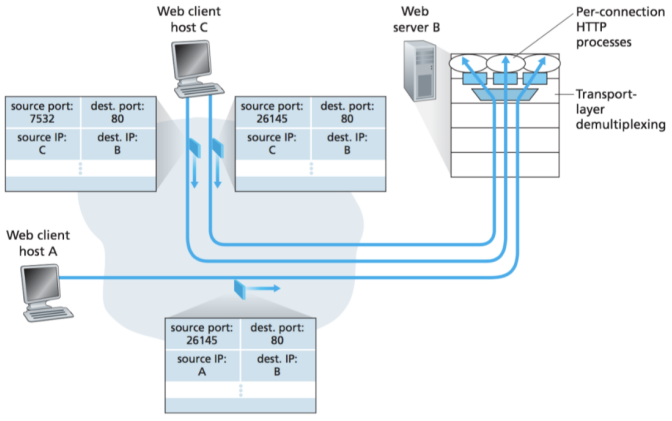
- **<IP-Dest, Port-Dest>**
- Hay un socket que habla con un Proceso
- **Hay 1 solo socket para todos los clientes**

CON CONEXION (HANDSHAKE) → TCP

Con Conexión

Segmento TCP identificado por: ⟨IP origen, puerto origen, IP destino, puerto destino⟩

- Servidores pueden gestionar múltiples conexiones TCP
- Se pueden asociar distintos sockets a distintos clientes

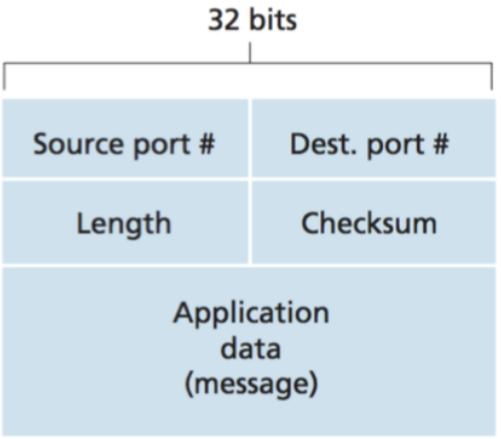


Identificador:

- **<IP-Origen, Port-Origen, IP-Dest, Port-Dest>**
- En Tcp, cada socket esta asociado a un proceso
- **UN SOCKET SIRVE A UN CLIENTE PARTICULAR**

UDP

- Servicio de "mejor esfuerzo" (best-effort).
- Segmentos pueden perderse.
- Segmentos pueden llegar en distinto orden.
- Servicio **no orientado a conexión** (connection-less)
- **No se establece conexión previa**
- SIN estado de conexión
- Cada segmento se gestiona de manera independiente



- Segmento UDP solo tiene 8 byte de overhead

Para identificar necesitamos una Ip, pero esta se maneja de una capa anterior (**Red**)

Cheksum → Para ver simplemente si el mensaje llego correcto, en UDP → Si el cheksum no coincide se ignora

¿Porque UDP? → **VENTAJAS UDP**

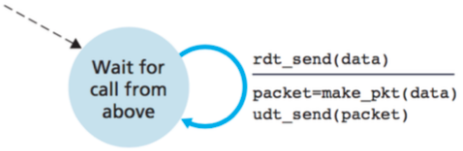
- Capa Aplicación puede controlar el envío
- No hay demora en establecer conexión
  - Menor Overhead de tiempo
- Menor overhead: 8 bytes → TCP USA 20bytes !!

→ Para entender protocolo TCP, estudiaremos primero **PRINCIPIOS de TRANSMISION CONFIABLE**

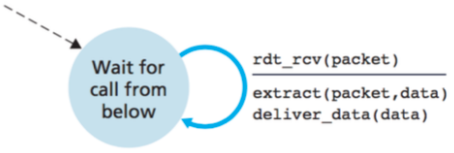
- desafio: Proveer **transferencia confiable** sobre un **MEDIO NO CONFIABLE**

Transmisión Confiable

- **Version 1.0** : Emisor y Receptor de mensajes
  - La comunicacion mas sencilla es:
    - El Emisor cuando la capa superior le pide enviar algo, crea un paquete y lo envia
    - El receptor, recibe un paquete, extrae el mensaje y lo pasa a la aplicacion
    - **Problema**: Paquetes pueden llegar con errores



a. rdt1.0: sending side



b. rdt1.0: receiving side

- Nota: Para leer la maquina de estados arriba del guion esta el evento que se pide, abajo de la barrita horizontal son las acciones que ocurren → En el caso uno, si no piden un send de data, se ejecuta crear el paquete, y mandarlo

CHEKSUMS (ACK)

- **Version 2.0** : Junto con el envio del paquete, agregamos una **SUMA (Cheksum),** si un dato llego corrupto, no va a coincidir con su cheksum
- **ACK y NAK** (usa cheksums) → Comprobar si la informacion llego bien o es la esperada
- **ACK**: Acnolagment: Decirle al emisor que llego el mensaje



Problema → Si llega el ACK o NAK corrupto?

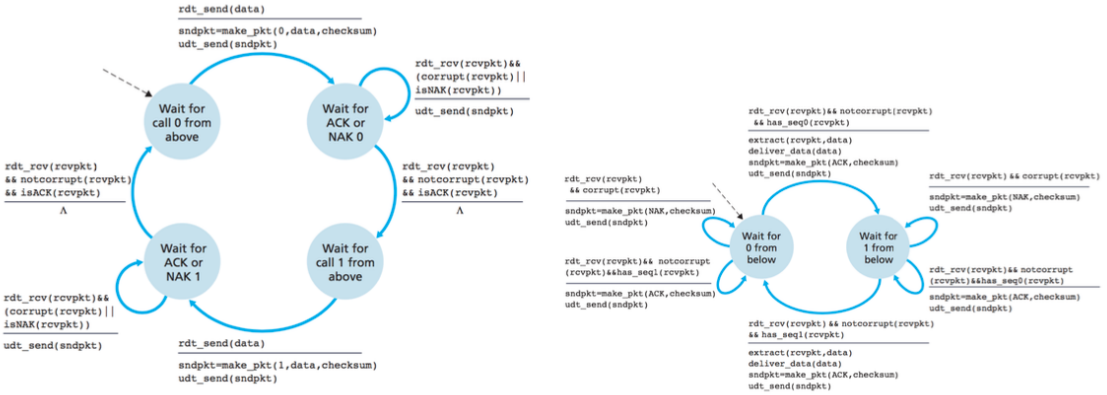
- Retransmitir al recibir un ACK o NAK corrupto → Pueden llegar **duplicados.** **→ corregido: 2.1**

Ejemplo Visual

Computador —— HTML —> Serv: Le envio una peticion html

Computador <—- ACK ——— Serv : Le devuelvo una Ack que me llego bien (Nack seria si no llego bien) → Ahi se entra en el loop visto en la diapositiva

- **Version 2.1** : Ahora los envios de mensajes son con un **NUMERO DE SEQUENCIA** (**identificador**),
  - El recibidor: espera un ACK y ademas un numero de secuencia, para corroborar bien si llego
  - Si me llega mal, llega un nack, con el id indicado (secuencia)



- Si nos llega un **ACK**, con un **numero de sequencia incorrecto**: identificamos un **DUPLICADO** de un mensaje anterior

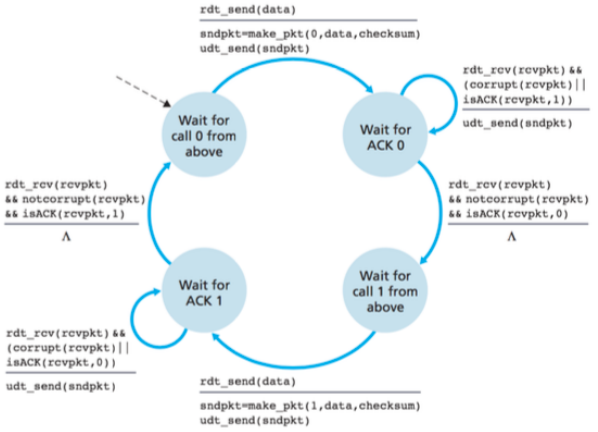
MEJORA → ENVIAR NAK se vuelve innecesario:

- **Version 2.2** :

Si el paquete llega corrupto, lo sabré → No necesito una confirmacion negativa!

El emisor:

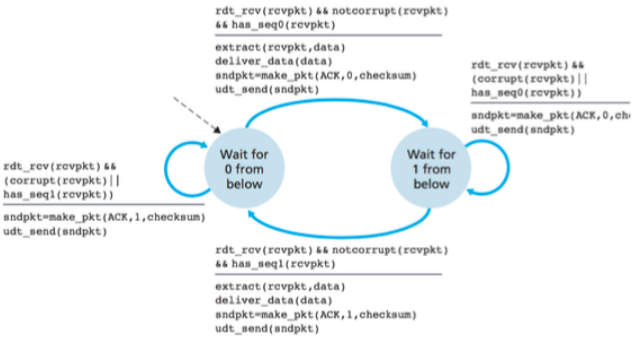
- Le piden enviar un paquete → sndpkt(0, data, cheksum) → El 0 es el numero de seq
- Reenvio paquetes cuando esta corrupto o el Ack con el numero 1, (asi lo identificamos como fallido)



- Reenvio hasta rebir un paquete no corrupto y con Ack 0 → En ese momento cambio de estado

El receptor:

- Lo mismo, si llega corrupto o tiene numero de sequencia no correcto → Se mantiene en estado
- Y si recibe un paquete no corrupto, y con sequencia 0, lo extrae y lo envia a la capa superior

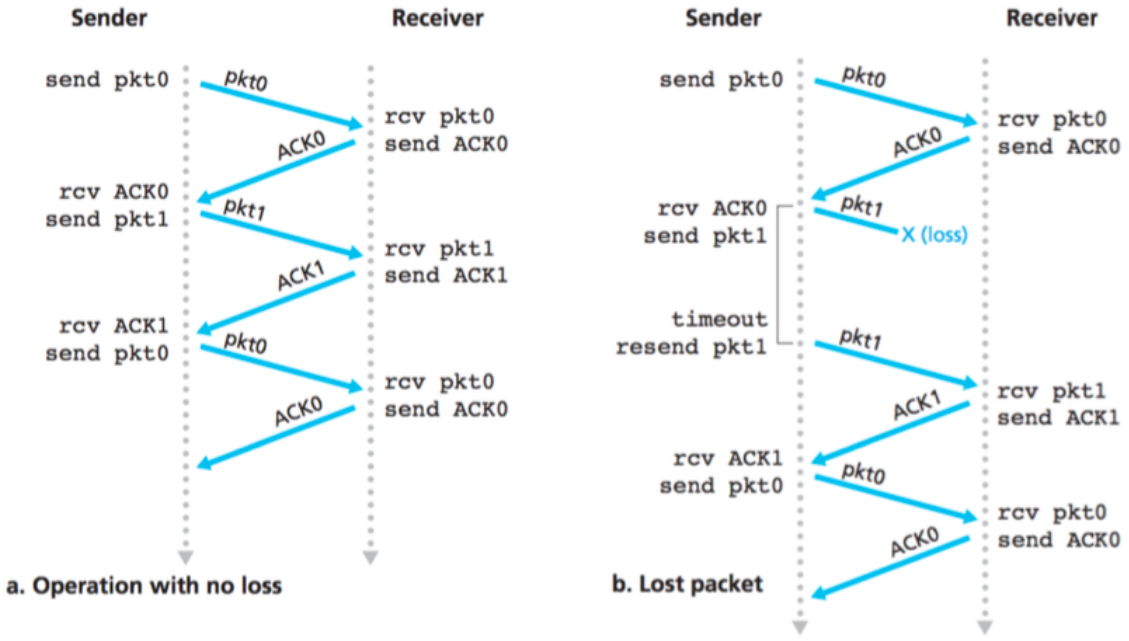


- **Version 3.0**: En el sender se implementa un TimeOut → Solo en el sender
- Asumo con esto, que algo se perdio, con eso reenvio el paquete, sin importar que lleguen 2, esperando a que me llegue el ACK para poder avanzar

RESUMEN

- Versión 1. Tiene los supuestos, pero los paquetes pueden llegar con errores
- Versión 2. Implementa los checksums, pero pueden haber paquetes de feedback duplicados(ACK O NAK).
- Versión 2.1. Agrega un bit que es el sequence number.
- Versión 2.2. Se pueden eliminar los NAK.
- Versión 3.0. Funciona pero el desempeño es malo

FUNCIONAMIENTOS SIN PERDIDA Y CON PERDIDA



- En esta version se ve mejor que procede

En la practica, no se manda solo un paquete, los protocolos funcionan con un modo **STOP-WAIT**

Envia una **SERIE DE PAQUETES, en modo PIPELINED** y ahi ver si uno falla, ver cuanto retrocedo

Usamos buffers, si se pierde un intermedio → Generara problemas

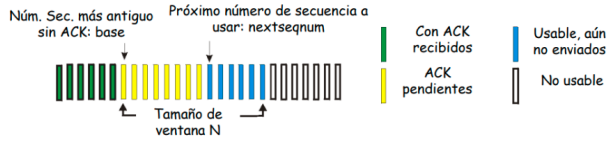
GO-BACK-N

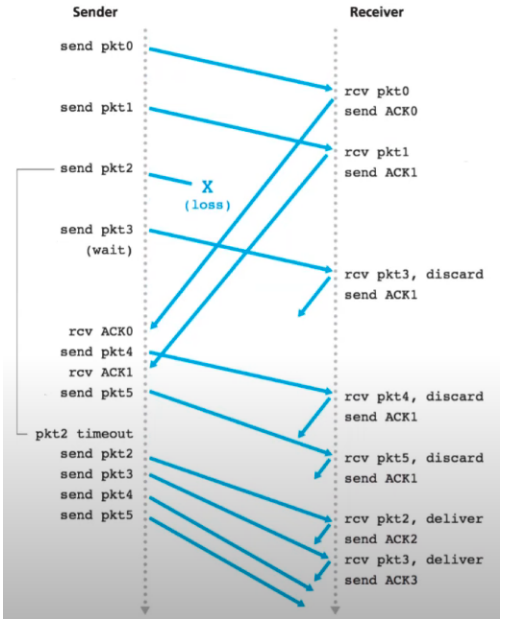
Emisor puede mantener hasta N paquetes sin ACK

Receptor envia ACK por el ultimo paquete recibido CORRECTamente

Emisor usa un **TIMER** para parquete mas ANTIGUO SIN ACK

- Si el timer expira, se **retrasmite** → **TODO EL GRUPO SIN ACK**
- Base : El paquete mas antiguo sin ACK
- nextseqnum: Proximo numero de secuencia a usar
- N es un **WINDOW SIZE**
  - Ejemplo con N = 4



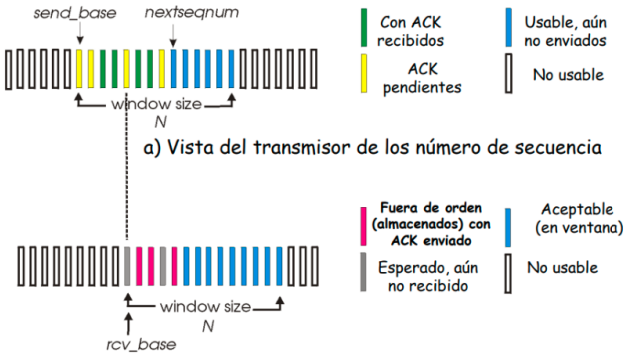


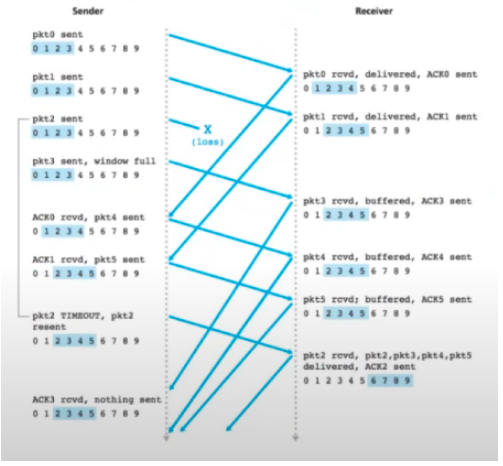
- Fijate que los Acks que devuelven el receptor, es 1 porque todavia no recibe el pkt2 (paquete 2), hasta que no lo reciba, no va a dar otro ACK

SELECTED-REPEAT

Tiene una ventana tanto en el emisor como en el Receptor

Recpetor No va a hacer avanzar la ventana, hasta que haya recibido todos los paquetes enviados del emisor





- Emisor puede mantener hasta N paquetes sin ACK
- Receptor envia ACK para cada paquete Individual!

- Emisor cuando recibe el ACK 0, avanza la ventana hacia el 1
- Receptor no mueve la ventana si no recibe cierto paquete
- Cuando se perdio el 2, el 3 4 y 5 ya habian sido enviados, por ende cuando lo reciba, la ventana avanza 2 3 4 5 → 6 7 8 9

- Emisor usa TIMER para paquete mas antiguo
  - En caso de timer expirado → **SOLO SE RETRANSMITEN PAQUETES SIN ACK**
  - **(NO TODOS)!!!**

Implementando estas cosas → Nace TCP

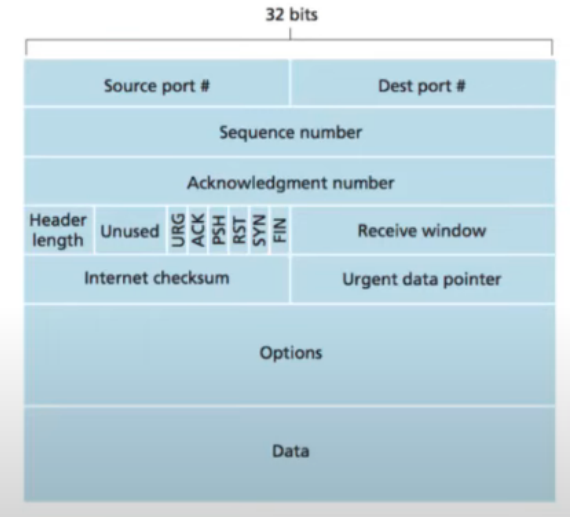
CLASE 5

21/10/24

PROTOCOLO TCP

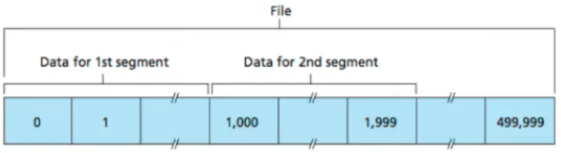
- Protocolo de transmisión **CONFIABLE**

- Encadenada → Secuencia de PAquetes → Hay un orden en que deben guardarse, para mantener el mensaje original enviado
- Hay un buffer en emisor y Receptor
- Orientado a **CONEXION → HANDSHAKE →** (Establencimiento de conexion antes de envio de pktes)
- Posee un overhead mayor a UDP (8): Posee 20 Bits

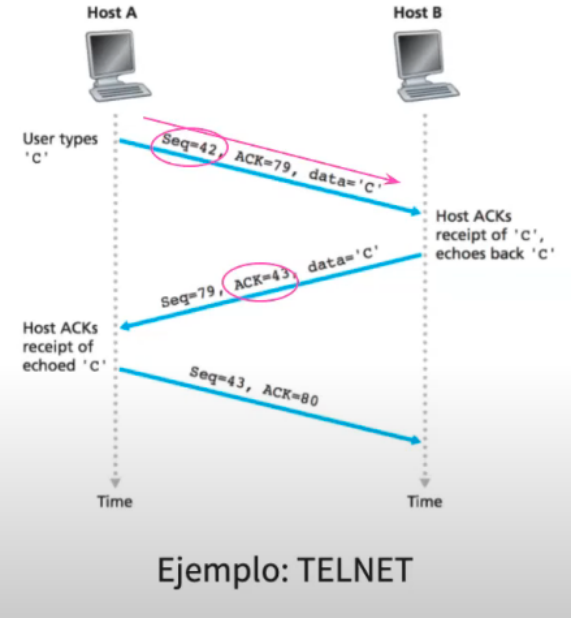


Funciona con **NUMERO DE SEQUENCIA →** No como vimos anteriormente que aumenta de a 1 bit, si no que aumenta segun cantidad de bytes enviados

El mensaje original se divide en varios segmentos:

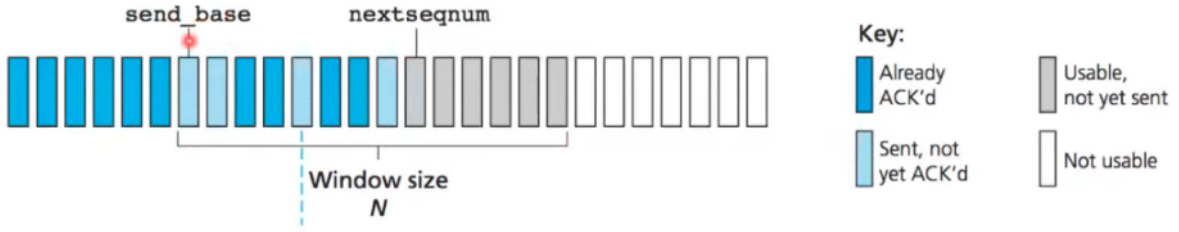


- MSS : Tamaño maximo de Segmento
- Num Seq: Ademas cuentan con bytes en la seq
- ACK contiene el sequence number del siguiente byte que se espera → Cumulative ACK
- Mensaje de respuesta funciona tambien como ACK
    - → Piggybacked ACK: Es como cuando vas a la cocina y hay una taza en el camino, aprovecha de llevarla → OPTIMIZACION



- En el ejemplo, viene un byte, con ‘c’, si la data fuera por ejemplo ‘iic2333’ serian 7 bites, por lo que el ack = 43 → Pasaria a enviarse como ACK = 49

IMPLEMENTACION TCP



SendBase, es la base de la ventana, aqui parte

Nextseqnum : Nuevo numero de seq a enviar : Lo anterior ya fue enviado (algunos recibieron ACK y otros nop)

- se actualiza al crear un segmento nuevo para enviar

Al Expirar el **TIMER → Se RETRASNMITE** segmento con **MENOR SEQ NUMBER** → (funciona como selected-repeat al no retrasnmitir todo)

- Esto porque el ACK pudo haberse perdido o el paquete enviado tmbn

1. Si llega un ACK con valor **ACK ≥ SendBase** → Se a**ctualiza SendBase**

   → Dado que se recibieron paquetes con valor menor a SendBase

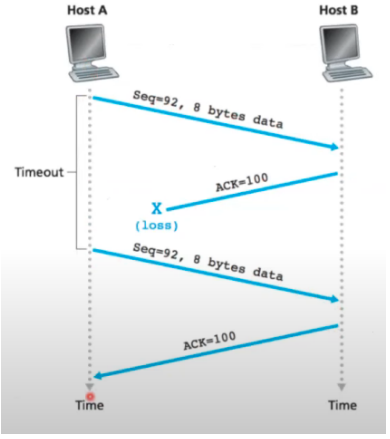
   → Como usamos el concepto de **CUMULATIVE ACK, significa** que si recibo por ejemplo el tercer ACK desde sendbase en adelante, significa que ya recibi los **ANTERIORES!!!**

   por eso actualizo SendBase

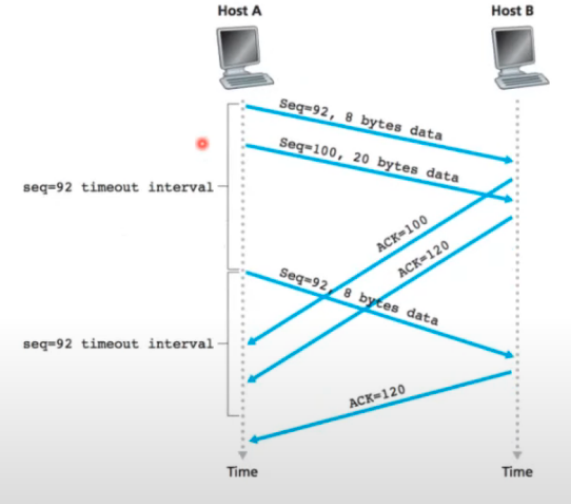
CASOS

1. ACK PERDIDO → Si el ack se pierde,

- se produce Timeout →
  - Reinicia timer y
  - reenvia paquete mas antiguo!
  - B recibira el paquete repetido, pero lo descartamos, reenviando ACK



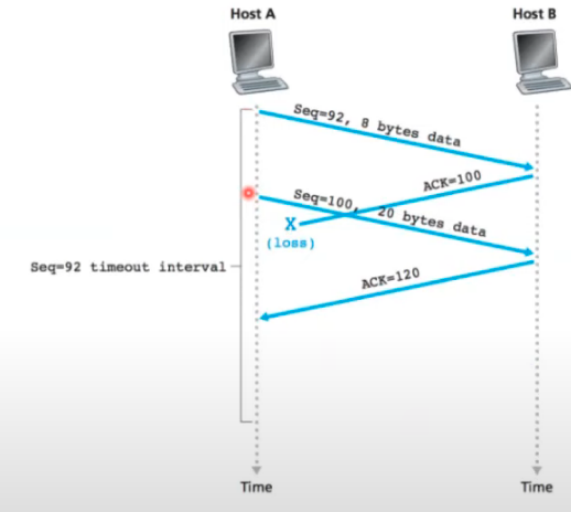
2. TIMEOUT PREMATURO → Si ocurre timeout antes que llegue el ACK, no importa
   a. A reenviara el paquete mas antiguo, y reinicia su timer
   b. A recibe el ACK y reinicia el timer del sgte mensaje
   c. A No reenvia el siguiente mensaje ya que su ACK llega antes del timeout
   d. B recibe el mensaje inicial dubplicado, lo descarta, pero



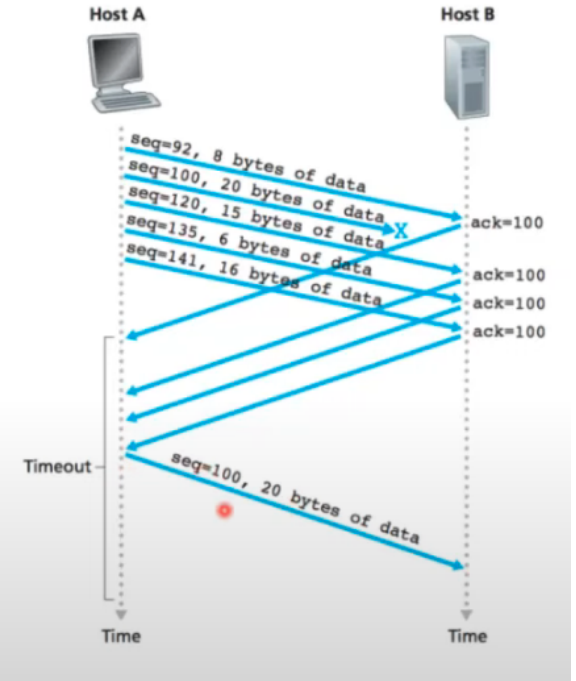
envia **CUMULATIVE ACK**

3. CUMULATIVE ACK → Evita retransmision (porque pesca todos los anteriores que perdieron ACK si recibe un ACK mas grande!)

- Fijate que llega el 120 y listo, es como que hubiera llegado el 100 cuando no llego (se perdio)
- A no reenvia seq=92
- A Incrementa SendBase
- Cuando me llega el 120 → CORTO EL TIMER!!! → Recibi todos los paquetes anteriores correctamente



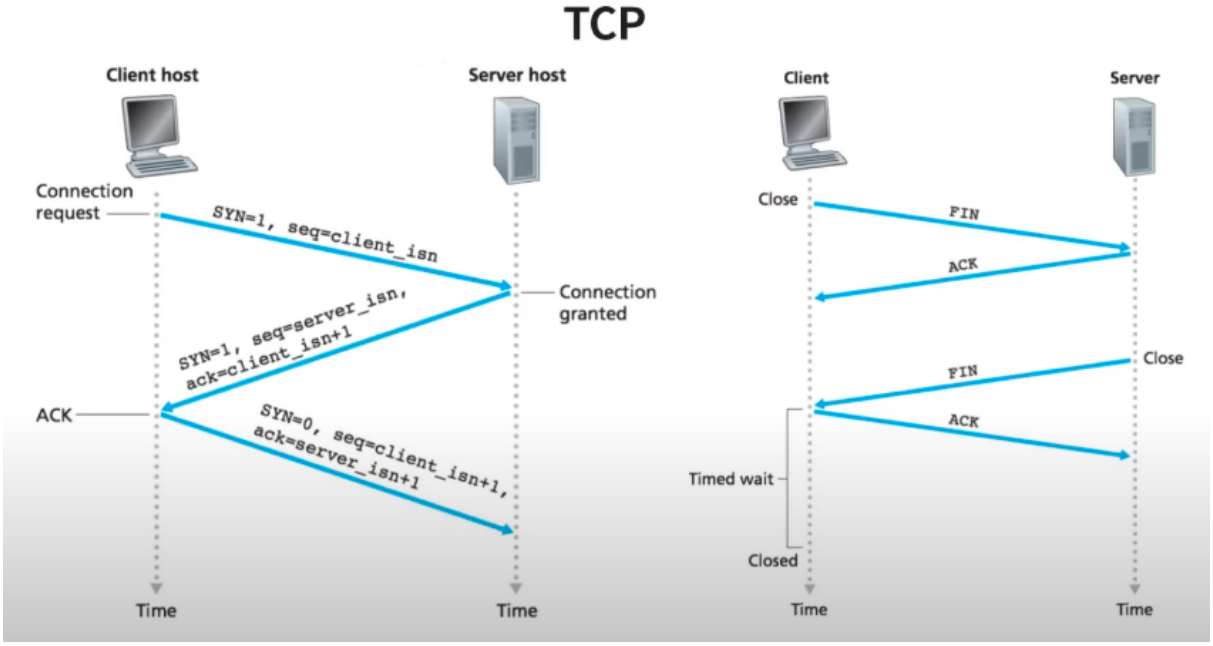
4. RETRANSMISION RAPIDA → Asi es como funciona TCP basicamente
    - Envia proactivamente segmentos perdidos, antes que ocurra un timeout
    - LA vuelta final despues de todo
    - Responde con ACK = 157



TCP HANDSHAKE

Establecimiento de Conexion de 3 pasos

1. Cliente envia segmento con el flag → **SYN:**
    - Incluye numero sequencia incial (del CLIENTE)
2. Servidor Recibe SYN, y responde con SYN y ACK
    - Servidor asigna Buffer
    - Servidor Establece numero sec inicial (Del SERVIDOR)
3. Cliente Recibe SYN Y ACK y responde con ACK
    - Este paquete enviado → YA PUEDE ENVIAR DATOS



A la derecha se encuentra el termino de conexion → Cerrar buffer → Cerrar conexion

CLASE 6

23/10/24

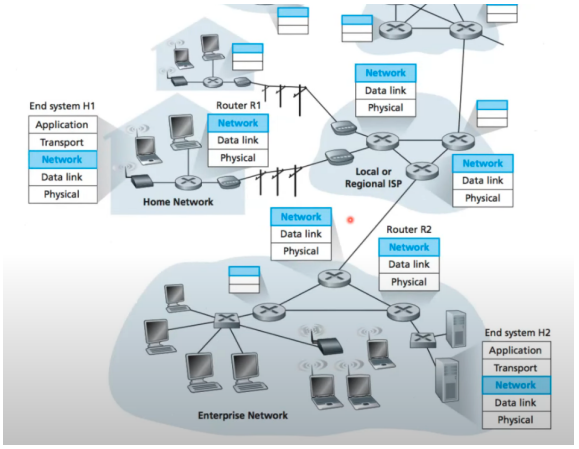
CONTENIDO 11 → CAPA DE RED

https://www.youtube.com/watch?v=syQOSjc-7NY&list=PLVTkEjeczcER2aOMHPIfaHCpTx61iKsoV&index=65

Esta capa esta encargada de:

1. Un Mensaje sea Dirigido → Encuentre al servidor
2. La capa de Red ocurre en cada uno de los dispositivos **INTERMEDIOS**

- Observar en el diagrama, que los dispositivos intermedios (Routers), llegan hasta la capa de NETWORK (RED)
- Va saltando entre elementos intermedios



Capa de red resuelve 2 desafios : **DIRECCIONAMIENTO y ENRUTAMIENTO**

Direccionamiento → Como encontrar la ubicacion del nodo destino en la red

Enrutamiento → Cordinar routers para que el pkte llegue al destino

Para realizar esto → Routers poseen un modelo **STORE - FORWARD**

STORE FORWARD: Almacenamiento y reenvio

- **Forwarding:** Decision local de un router
  - Cada router determina el proximo camino de un paquete mirando su direccion de destino
  - Se usa una tabla de forwarding → Tabla local → Ve headers y ve a donde salir
  - Cada tabla es unica por router

- **Enrutamiento (Routing)**: Decisión distribuida entre routers, ellos ejecutan algoritmo de routing
    - Estos algoritmos, permiten llenar la forwarding tables

MODOS DE CONEXION

VIRTUAL NETWORK: ESQUEMA RED TELEFONICA

Es un modo, que usa la via telefonica por ejemplo, establece un circuito

Antes de establecer una conexion, va construyendo una ruta de a poco el circuito

En construye primero la autopista entre yo y el destino

- Router mantiene estado de la conexión: **Connection-oriented**
- Garantiza ancho de banda durante la conexión
- Sin embargo: **Poco flexible** a **CONGESTIONES**

  Provoca que si un router esta saturado → No va a poder tener mas conexiones

  Por ejemplo → Terremoto, la linea telefonica tiene ancho de banda limite → Satura y la llamada no se establece

DATAGRAM NETWORK: ESQUEMA DE INTERNET

- Asi funciona la internet, mediante segmentos tcp, se encapsulan paquetes, con sus headers, y se envian, y los routers van ruteando ellos, podrian tener distintos caminos
- Los Routers reenvian de acuerdo a forwarding Table
- Routers no saben de caminos: **Connectionless**
- NO GARANTIZA ANCHO DE BANDA!! → El otro si
- Sin embargo:
    - Es **Flexible ante CONGESTION!!**

- Permite ocupar de mejor manera ancho de banda

FORWARDING TABLES

Router debe tener respuesta para cada direccion posible → Usualmente contiene prefijos!

Ejemplos con 32 bits:

Prefix Match	Link Interface
11001000 00010111 00010	0
11001000 00010111 00011000	1
11001000 00010111 00011	2
otherwise	3

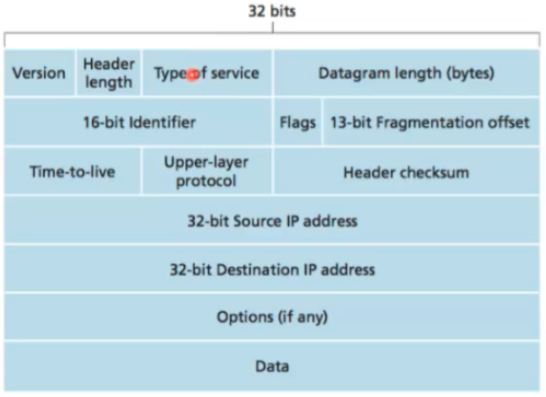
¿Como un mensaje encuentra su destinatario?

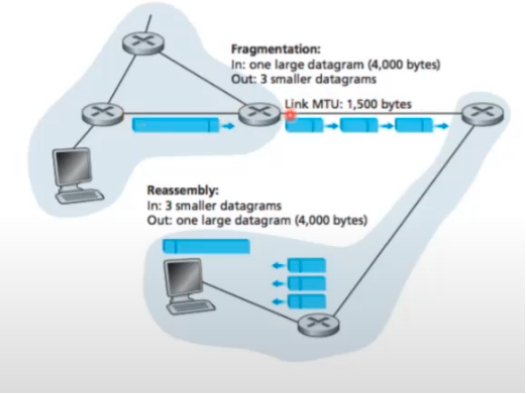
Gracias al **DATAGRAMA (PAQUETE) IP**

Longitud tipica < 1500 bytes

El tamaño va a estar determinado por los **ENLACES INTERMEDIO**

Parametro: MTU : Maximum Transfer Unit





- Paso 1: Fragmentar Datagrama

→ Problemas de la capa de ENLACE

Enlace con **MENOR MTU:**

Como la capa de red, utiliza la capa de enlace para hacer un salto a otro, hay que ver si ¿**puede caber?** en otros enlaces,

si no cabe, hay que fragmentarlo!!

- Enviar fragmentos con mismo ID
- Router del otro extremos lo reensambla

Por ejemplo en 3 paquetes, con el mismo identificador, y con un flag, cuando se levante, ya no quedan fragmentos

DIRECCIONES IP

- **IPV4: 4 Bytes → 32 bits: 2**32 direcciones posibles**
  - Dado que esos numeros de 32 bits son limitados, no existen unicos para todos los dispositivos
  - **DOTTED DECIMAL NOTATION → G**rupo de 8 BITS: (0-255)
    - 146.155.13.15: 10010010.10011011.00001101.00101101

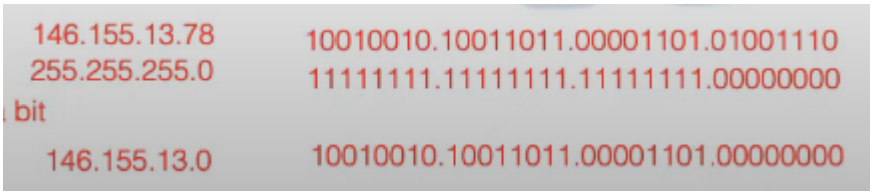
Tenemos subconjuntos llamados Subredes → Se dividen por prefijos de direcciones IP

Classless interdomain Routing → a.b.c.d/x

- ej: 223.1.1.0/24
- 223.1.1 = Subred
- .0/24 = Numero de host → la idea es que no se repite con otro computador
- 24 → Se Refiere a la ctd de bits que requiere la sub RED!!!
- Ademas se le añade una **MASCARA**

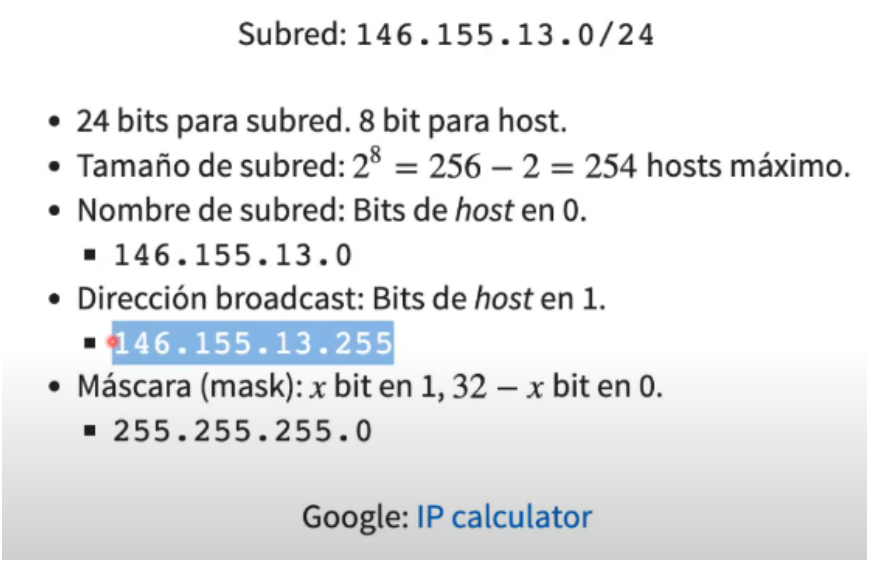
MASCARA: Si quiero conocer a que su **¿red pertence una subred?**, aplico un **AND bit a bit** con una **Mascara**

ej: /24 : 255.255.255.0: equivalente a tener 24 bit 1, 8 bit 0



IP & MASCARA = SUBRED

En este caso: Dio que la **SUBRED ES : 146.155.13.0**



Para este ejemplo: Siempre son 2 las Direcciones que no podemos utilizar de esos 24 bits que son, para la subred hay 2 que no!

1. La que tenga direccion 0 → Es el nombre de la Subred a.b.c.0 → Es la subred
2. La que tiene todos: por ej: 146.155.13.**255** → Esto sera la direccion de **BROADCAST**

Bits del host en 1: BROADCAST → **Envio a todos los miembros de la SUBRED**

→ IPV6: 6 **Bytes → 128 bits: 2**128 direcciones posibles**

Dentro del direccionamiento de REDES, cada Router es un **LIMITANTE → pto limite**

Cada **ROUTER** posee una **direccion IP** en cada salida

→ **Switches** por otro lado, **NO poseen direccion IP**

→ Switches son dispositivos de **CAPA 2 (ENLACE)**

Cada grupo de **HOSTS** conectados a un **ROUTER** → Forma una **SUBRED**

- En el ejemplo, hay 6 subredes!



Direcciones reservadas

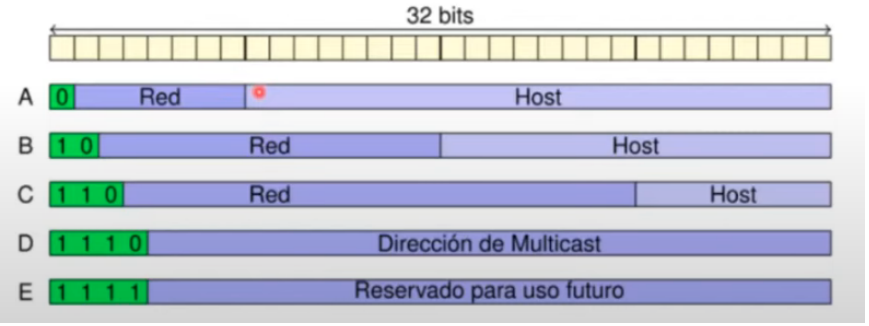
- 0.0.0.0. Forma de referirse a la **Dirección de host actual** (sólo sirve como origen).
  "Todas las IPs locales".
- 127.0.0.1. Dirección loopback (**localhost**)

Subredes reservadas

- Red local: 0.0.0.0/8
- Red loopback: 127.0.0.0/8
- Redes privadas:
  - 10.0.0.0 / **8**
  - 172.16.0.0 / **12**
  - 192.168.0.0 / **16**

Modelo antiguo de clasificacion de redes → CLASES

Clase	bits	Rango de direcciones	Hosts / Red	# redes
A	0	1.0.0.0 → 126.0.0.0	$2^{24} - 2 = 16.777.214$	$2^{7} - 2 = 126$
B	10	128.1.0.0 → 191.254.0.0	$2^{16} - 2 = 65.534$	$2^{14} - 2 = 16,382$
C	110	192.0.1.0 → 223.255.254.0	$2^{8} - 2 = 254$	$2^{21} - 2 = 2,097,150$
D	1110	224.0.0.1 → 239.255.255.254	$2^{28} - 2 = 268.435.454$	-
E	11110	240.0.0.1 → 247.255.255.254	$2^{27} - 2 = 134.217.726$	-



Luego esto se descarto porque se usaban mas direcciones de las que realmente se necesitaban, era mas facil usar las **MASCARAS** para determinar el tamaño de subredes

Asignacion de Direcciones

ISPs (Proveedores de internet) asignan bloques de direcciones a organizaciones, y publican sus bloques

Organizaciones pueden crear nuevas subredes dentro de su bloque

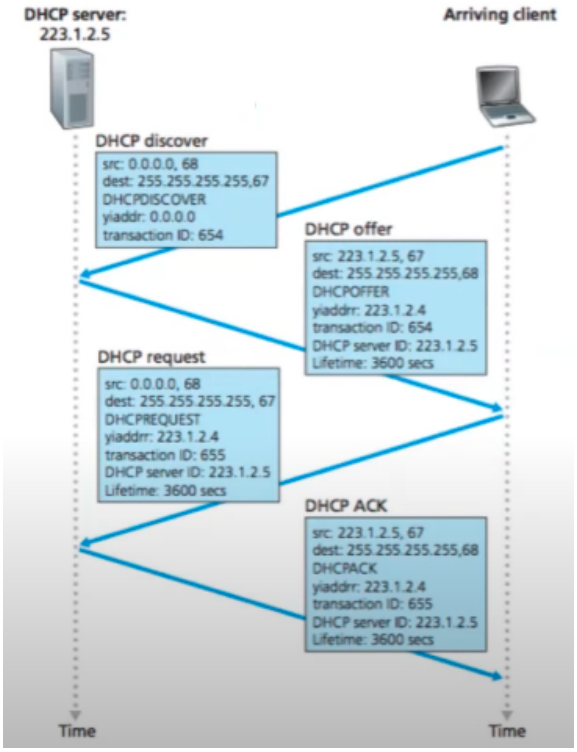
DHCP : DYNAMIC HOST CONFIG PROTOCOL

DHCP es un protocolo plug and play, que nos configura nuestra IP

Cuando detecta que alguien se contecta, le entrega los parametros para la conexion

4 Etapas del protocolo

1. Cuando el cliente llega (enchufa a la red) → **Cliente DHCP discover message**
    - Destino → 255.255.255.255:
2. El Servidor Responde: **DHCP offer Message**
    - Destino → 255.255.255.255 : Sigue siendo braodcast (le llega a todos)
3. Cliente envía un **DHCP Request Message**
4. Server envía **DHCP ACK Message**



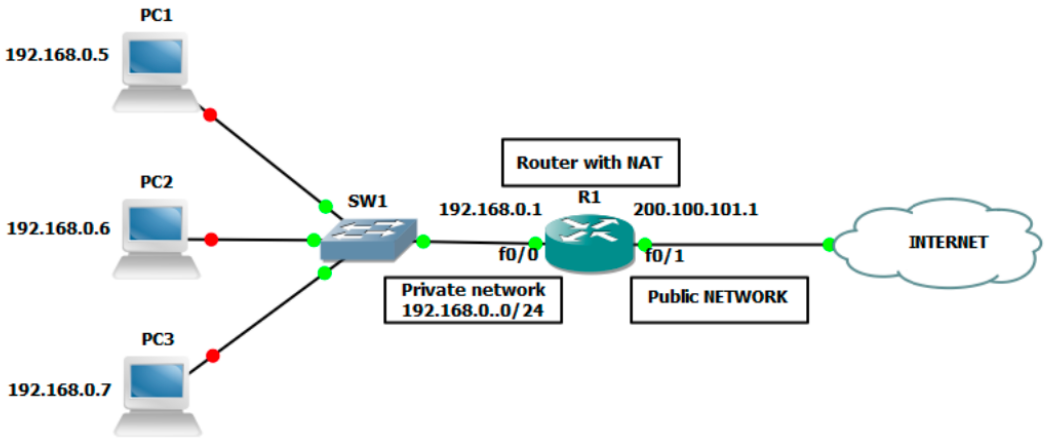
NAT

Network Addres Translation

ISP provee ctd **limitada** de **IPs**, y aveces **solo una** por Casas o oficinas que formen una **LAN**

NAT: Es una tecnología que permite que **varios dispositivos** se conecten a Internet usando **1 sola dirección IP pública**.

- Esto es posible gracias a que la NAT utiliza direcciones IP privadas de forma **interna** en la red doméstica.
- Resumen:
    - 1 Direccion Publica
    - Varias Direcciones Internas → Usando **NAT**



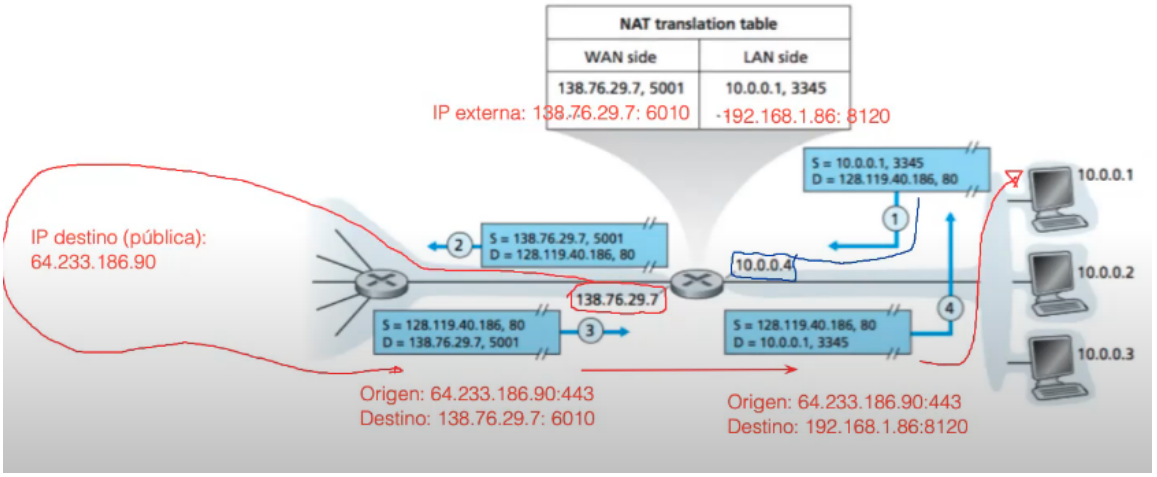
EJEMPLO DE USO COTIDIANO

1. Mi ip interna es 192.168.1.86 y por el puerto 8120 me quiero conectar a Youtube.com, lo hare con ip destinatario: 64.233.186.90 por el puerto 443(https),
    a. La ip destino no vive dentro de la Misma Subred
    b. Consecuencia: Comunicarse con el **ROUTER (gateway)**
        i. Gateway : Puerta de enlace

El router que implementa NAT: va a recibir el paquete, cuando reciba eso, va a ver, que alguien interno quiere conectarse a alguien externo

El Router lo que hace será : **CAMBIAR EL PAQUETE (header)**

- Para cambiarle la direccion de Origen, ya no sera 192.168.1.86, si no sera que la que proveyo el ISP ej: 138.76.29.7 por el puerto 6010
- Luego va el paquete al servidor, se devuelve con origen 64.233.186.90 puerto 443, hacia el destino 128.76.29.7 puerto 6010
- Y el **PROPIO ROUTER NAT,** traduce a quien le llega el paquete → Que es 192.168.1.86



Eso desde el lado del cliente → **COMO FUNCIONA PARA UN SERVIDOR?**

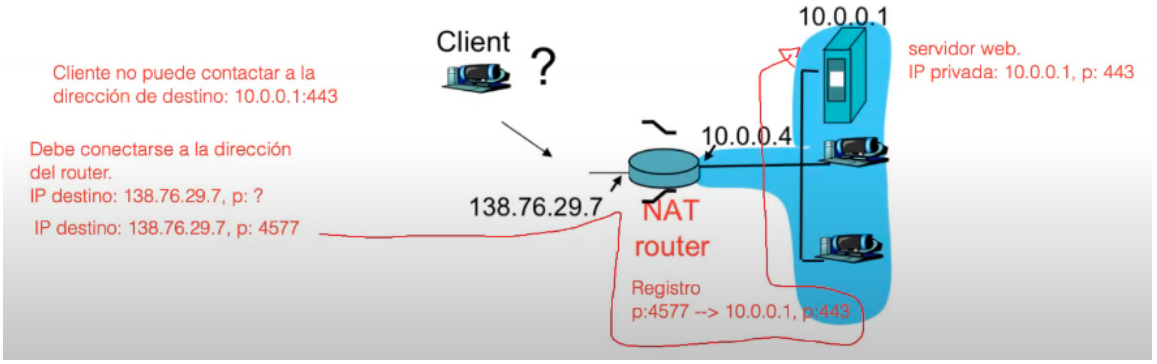
Conectar un Servicio detras de NAT

El Cliente no sabe con quien contactarse (es una **IP PRIVADA**)

El Cliente para conectarse al servidor, tiene que conectarse con la **direccion del Router**

El Router del servidor debe tener un protocolo llamado: **UPNP (Universal plug-play)**

- Esto le permite que los serviodres se registren dentro del **ROUTER NAT**
- Se registra a travez de un **PUERTO**



El Router poseera el registro de que puertos dirigen a que direcciones

Los Clientes deben conocer esto → El servidor debe anunciarse, o en algun DNS , para que el Cliente sepa a que puerto debe conectarse!

ANTI NATS

NAT tiene distractores, dado que Viola el principio de **Extremo-Extremo**

- Router son de Capa 3 → NAT requiere que modifiquen puertos, y **Puertos no son de capa 3**
  - Son de **CAPA DE TRANSPORTE!!!**
- Comunicacion Sin Modificacion Intermedia
- NAT es un parche, con ipv6 ya nunca deberiamos necesitar NAT

ICMP

Protocolo de Control de condiciones de la RED

- Conjunto de **reglas de comunicación** que se utiliza para **diagnosticar problemas de comunicación en una red**.
- Usado para **Mensajes de Diagnostico y Control de la capa de RED**

ICMP Type	Code	Description
0	0	eco reply (to ping)
3	0	destination network unreachable
3	1	destination host unreachable
3	2	destination protocol unreachable
3	7	destination host unknown
8	0	eco request
10	0	router discovery
11	0	TTL expired

Table 1-3 ICMP Message Types

- ICMP contiene dos campos: (ICMP type, ICMP code)

Mensajes Conocidos:

- **(8,0) : Echo Request**
- **(0,0): Echo reply (PING)**
- **Programa traceroute usa mensajes ICMP con distintos TTL.**

IPv6

IPv6 utliza direcciones de 128 bits → Numero tan grande como atomos en el Universo!!!

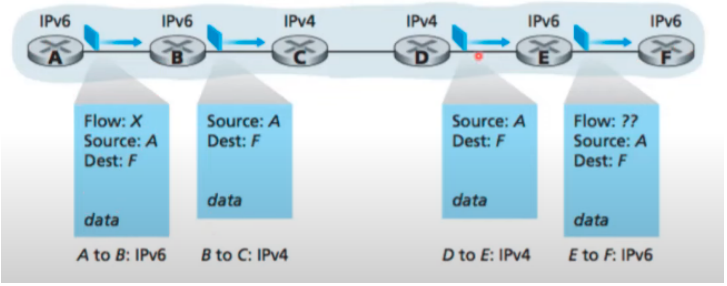
No todo el mundo aun se cambia a Ipv6

HEADER : 40 Bytes: Prioridad es Agilizar el Procedimiento

¿Como hacer la **TRANSICIÓN**?

Router IPV6 puede ser backward compatibles (al 4), pero Router IPV4 **NO!**

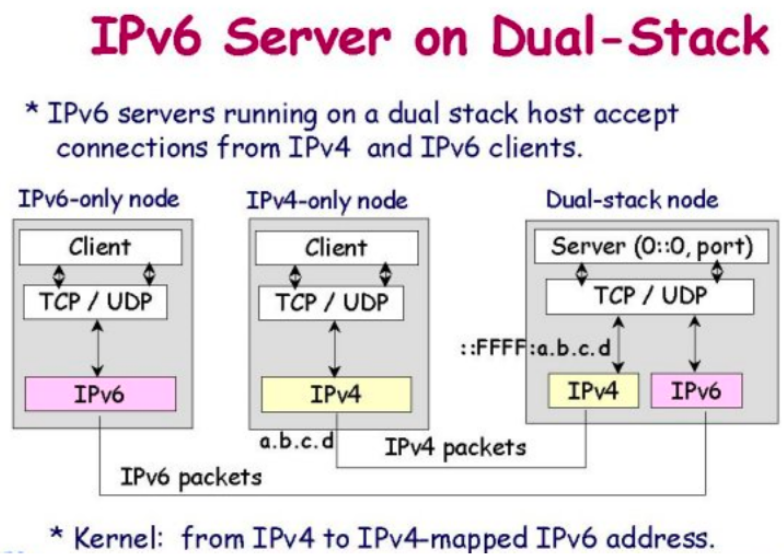
- Notar que en el paso de IPv4 a 6, Flow = ??



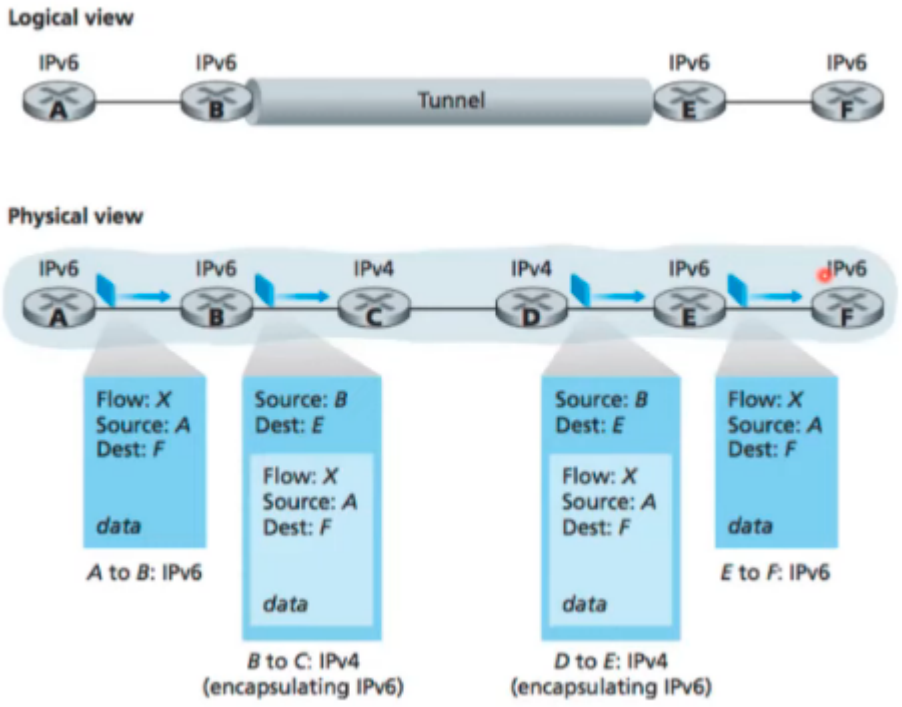
Soluciones

- Establecer un dia de cambio mundial? → YA se intentó

1. Usar Routers **DUAL STACK (IPv6/IPv4)**



2. **TUNNELING:** Encapsular el paquete de un formato dentro de otro → Para darle la forma para que pase por cierta comunicacion especial (tramos ipv4)



ALGORITMOS DE RUTEO

Podemos modelar el conjunto de routers como un **GRAFO G = (V, E)**

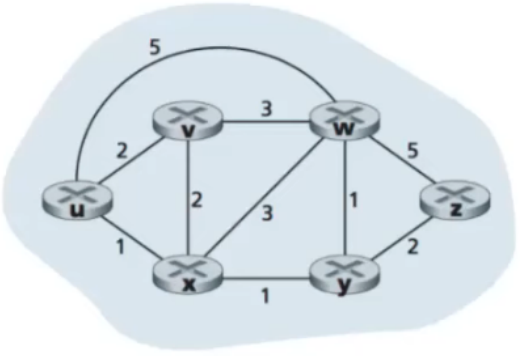
- Costo entre conexiones: Ancho de banda, congestion, distancia
- Enlaces Son las Aristas!

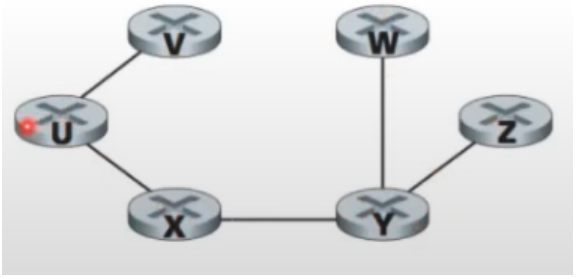
1. **ALGORITMOS CENTRALIZADOS**
    - Nodos conocen la info **COMPLETA** de la RED
    - Algoritmos **LINK-STATE (LS)**
        - Se explora toda la red
        - Y cada nodo determina la ruta mas conveniente

2. **ALGORITMOS DECENTRALIZADOS**
    - Calculo Iterativo y Distribuido entre nodos → Sin Conocer la **TOPOLOGIA COMPLETA**
    - Algoritmos de **Distance-Vector (DS)**
        - Calculo distribuido entre los nodos

LINK-STATE

1. Cada Nodo **ENVIA INFO** de conectividad (Vecinos y Costos), a sus **VECINOS**
2. Se comunican via flooding
3. Cuando todos tienen la info, cada uno Calcula Rutas mas Cortas
4. Implementa por ejemplo : **DIJKSTRA**

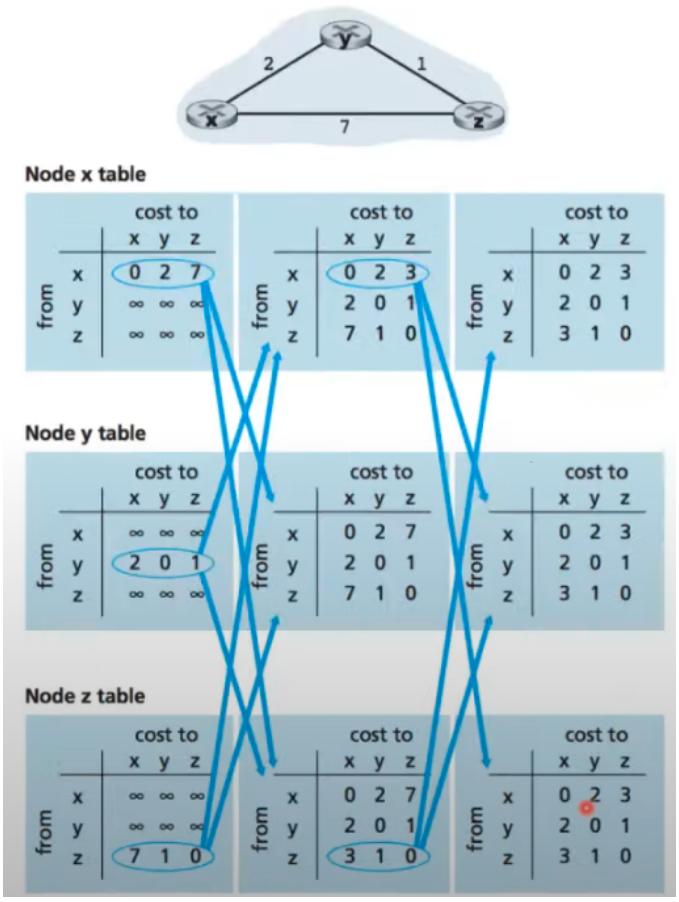




DISTANCE-VECTOR

Utliza la EC de **BELLAMN FORD: Para calc la ruta mas corta entre cada par de nodos es:**
Comparo Ruta directa vs El costo de pasar con cada uno de los costos intermedios

$d(x,y) = \min_v\{c(x, v) + d(v, y)\}$



DIFERENCIA de implement

Link-State: propaga TOPOLOGIAS

Distance-Vector: Ruta mas corta hacia un destinatario

- Distance Vector a diferencia de LS: se propaga información de rutas, en lugar de enlaces

---

Sin embargo, distance-vector tiene un problema → CONTEO HACIA EL INFINITO

Poison Ruta → Si se envenena la ruta ejemplo

```
      y
    /   \
   /     \
4 -> 60   1
```

```
 /         \
x --- 50--- z
```

Si el 4 en (y,x) se actualizace a 60, z no sabe que eso paso, y cuando y quiera irse por z, entonces z le dira, pero oye hay una ruta mas corta que toma 5 (1 + 4), y cuando se devuelve toma 61 → Mas caro (se enveneno)

Lo que hacemos para estos casos es **DECLARAR LA RUTA INVALIDA**

LINK STATE	DISTANCE-VECTOR
N NODOS, E ENLACES: O(NE) Mensajes	Se ejecutan en **paralelo**, y solo se propaga ruta a los vecinos
O(N^2) Complejidad en Cada NODO	Ante Errores, se propaga el error
Ante errores → Se propaga ausencia de enlace local	Puede converger muy lentamente, Tiempo de convergencia Indeterminado

- No es factible almacenar Tablas para todos los Routers de Internet!!!
- Routers pertenecen a distintas organizaciones
- Router pertencientes al mismo dominio administrativo forman un **SISTEMA AUTONOMO**

Finalmente un conjunto de routers, lo veremos como un conjunto de sistemas autonomos

Cada sistema autonomo, esta administrado por una entidad → Por lo tanto habra un algoritmo Interno LiNK STATE → Es mas facil conseguir la topologia

Routers de salida (de Sistemas autonomos) son **GATEWAY ROUTERS**

Algoritmos inter-AS ayudan a determinar **FORWARDING TABLES entre distintos AS**

Algorimtos intra-AS resuelven ruteo interno

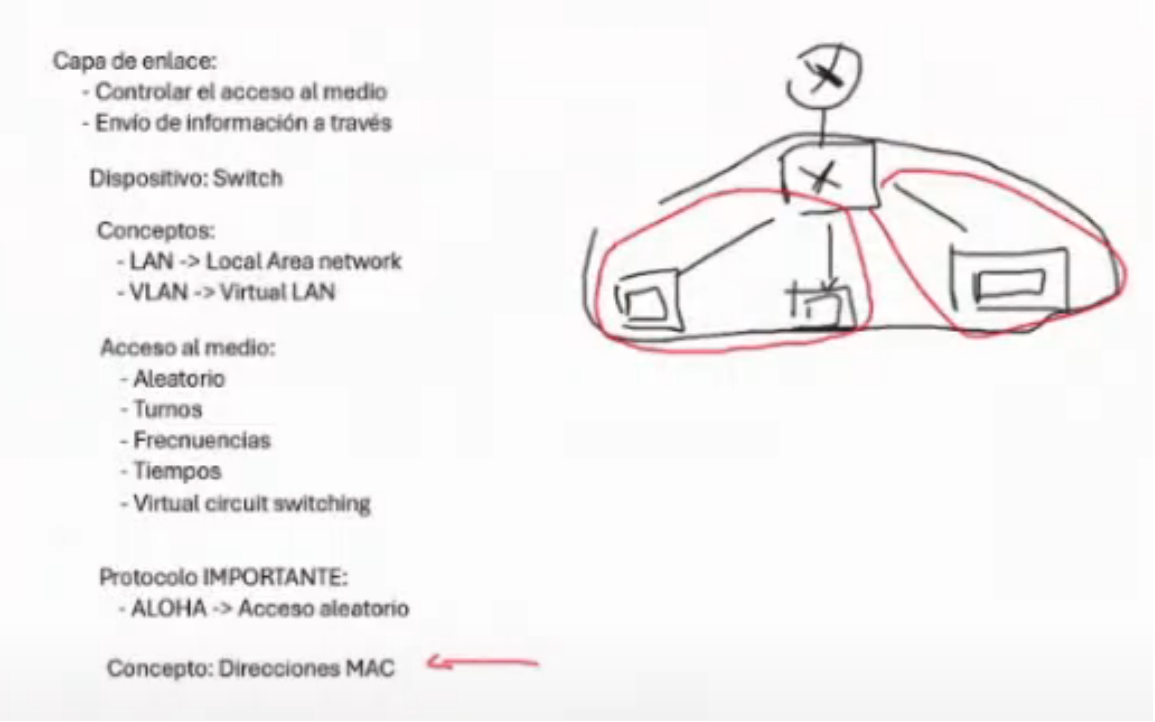
Clase 7

4/11/24

CONTENIDO 12: CAPA ENLACE

SWITCH → XD

Por Estudiar:



CAPA ENLACE:

1. Controla el Acceso al Medio

2. Controla el Envio de informaciones a traves de los medios (Alambricos o Inalambricos)

El Dispositivo es el Switch

LAN : Local Area Network → A partir de un router, una interfaz que puede ser extendida a un switch, la Lan son todos aquellos dispositivos conectados a una misma **SUBRED**

VLAN : Lans Virtuales, para separar trafico de pquetes segun se quiera separar (ej: Domotica, servidores, etc..)

ACCESO AL MEDIO

1. Aleatorio → Importante para el algoritmo **ALOHA**
2. Turnos
3. Frecuencias
4. Virtual Circuit Switching

Protocolo Importante: ALOHA → Todos los dispositivos que estan en una red, le permite a cada uno hablar de forma aleatoria, si hay colisiones, te toca hablar en cierto tiempo despues

Se controla el tiempo de forma aleatoria.

Direcciones **MAC**

Direcciones fisicas que se encuentran dentro de la Tarjeta de Red, Nos sirven para que **EL SWITCH NOS IDENTIFIQUE, y el camino por donde nos llega la INFO**

CHARLA SEGURIDAD

Anotar lo mas importante → No hay videos del profe RUZ , pero la SLIDES tiene lo importante

1. Problemas y Casos tipicos de **FALLAS DE SEGURIDAD**
2. **Estrategias** Sobre **SEGURIDAD**