

# RESUMEN REDES Y COMUNICACIONES

## Clase 1 | Introducción a las Redes

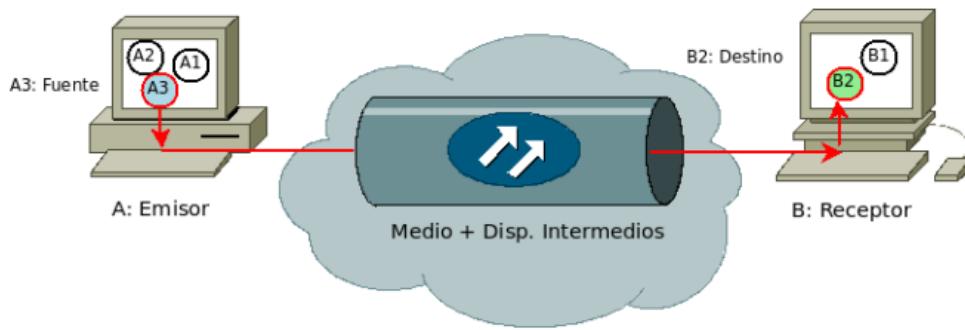
### **RED**

- . Una Red de Computadoras es un grupo de computadoras/dispositivos interconectados. El objetivo es compartir recursos como dispositivos, información y servicios.

### **COMPONENTES DE UNA RED**

- . Fuente (Software).
- . Emisor/Transmisor (Hardware).
- . Medio de transmisión y dispositivos intermedios (Hardware).
- . Procesos intermedios que tratan la información (Software y Hardware).
- . Receptor (Hardware).
- . Destino (Software).
- . Otros: Protocolos (Software), Información, mensaje transmitido (Software).
- . Señal de Información, materialización del mensaje sobre el medio (Hardware?).

*Las componentes de la red deben interactuar y combinarse a través de reglas -> protocolos.*



### **PROTOCOLOS**

- . Un protocolo es un conjunto de conductas y normas a conocer, respetar y cumplir para comunicarse no solo en el medio oficial ya establecido, sino también en el medio social, laboral, etc.  
En términos de redes, un protocolo define el formato, el orden de los mensajes intercambiados y las acciones que se llevan a cabo en la transmisión y/o recepción de un mensaje u otro evento.
- . Un Protocolo de Red es un conjunto de reglas que especifican el intercambio de datos u órdenes durante la comunicación entre las entidades que forman parte de una red. Permiten la comunicación y están implementados en las componentes.

### **REDES CONMUTADAS**

- . Existen redes conmutadas de paquetes y redes conmutadas de circuitos.
- . En una red conmutada de paquetes, los datos se dividen en paquetes y cada uno viaja de forma independiente a través de la red, tomando rutas diferentes si es necesario. Esto

permite un uso eficiente del ancho de banda y una mayor flexibilidad, pero puede causar variabilidad en el tiempo de entrega (latencia).

. En cambio, en una red conmutada de circuitos, se establece una conexión fija y dedicada entre dos puntos durante toda la comunicación. Esto garantiza un ancho de banda constante y un retraso mínimo, pero puede ser menos eficiente en el uso de recursos si la conexión está ociosa.

. Ejemplo:

- Una red de telefonía, tradicionalmente usa conmutación de circuitos. En una llamada telefónica, se establece un circuito dedicado entre el emisor y el receptor durante toda la conversación, garantizando una calidad constante y baja latencia.
- Internet, por su parte, utiliza conmutación de paquetes. Los datos se dividen en paquetes que viajan de forma independiente a través de la red, permitiendo un uso más eficiente del ancho de banda y adaptándose a la variabilidad en el tráfico de la red.

## ***REDES PROPIETARIAS***

. A principios de los 80' las compañías empezaron a implementar redes propias que eran privadas y cerradas. Esto dio comienzo a las primeras Redes Propietarias.

Esto provocó que cada red tenga sus propias especificaciones (protocolos), generando una incompatibilidad entre distintas redes haciendo que la comunicación entre ellas se volviera muy difícil, evolucionara más lentamente y carenciara de estándares.

## ***MODELOS DE ORGANIZACIÓN***

. Nació entonces la necesidad de combinar los distintos protocolos, lo cual dio inicio a la aparición de modelos de organización.

. El Modelo en Capas (Layering) divide la complejidad en componentes reusables, reduciendo la complejidad en componentes más pequeños. Las capas de abajo ocultan la complejidad a las de arriba → abstracción. Cada capa, entonces, utiliza servicios de las de abajo y ofrece servicios a las de arriba.

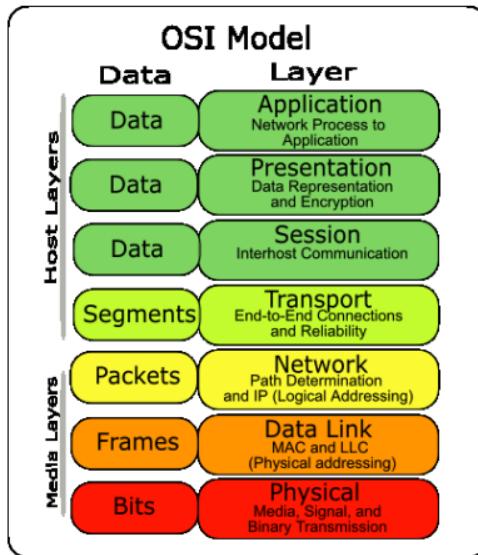
Los cambios en una capa no deberían afectar a las demás si la interfaz se mantiene.

. Las ventajas que ofrece la organización en capas son la modularidad, interoperabilidad, flexibilidad, seguridad y gestión, y simplificación.

## ***MODELO OSI (Open System Interconnection)***

. Ante la necesidad de desarrollar componentes *estándares* de red, La ISO (International Standard Org.) crea el modelo OSI en 1984, basado en los modelos de red en capas.

Este nuevo modelo sería abierto y estándar, estaría dividido en 7 capas, y serviría como modelos de referencia. Es un modelo más teórico.



- . Capas de Host (Host layers): 7,6,5,4, proveen envío de datos de forma confiable.
- . Capas de Medio (Media layers): 3,2,1, controlan el envío físico de los mensajes sobre la red.

#### **. Funcionalidad de las capas:**

Aplicación (7): servicios de red a los usuarios y a procesos, aplicaciones.

Presentación/Representación (6): formato de los datos.

Sesión (5): mantener track de sesiones de la aplicación.

Transporte (4): establecer y mantener canal “seguro” end-to-end (applic-to-applic).

Asegura la entrega correcta y en orden de los datos entre aplicaciones en diferentes sistemas.

Red (3): direccionar y rutear los mensajes host-to-host. Comunicar varias redes.

Enlace de Datos (2): comunicación entre entes directamente conectados. Comunicar una misma red. Acceso al Medio.

Física (1): transportar la información como señal por el medio físico. Características físicas. Información binaria, digital.

## **MODELO TCP/IP**

- . El modelo TCP/IP es la base de Internet y la mayoría de las redes modernas, proporcionando un marco estándar para la comunicación en redes distribuidas.
- . Modelo que se convirtió en estándar. Consta de 5 capas:

Capa de Aplicación (Process/Application).

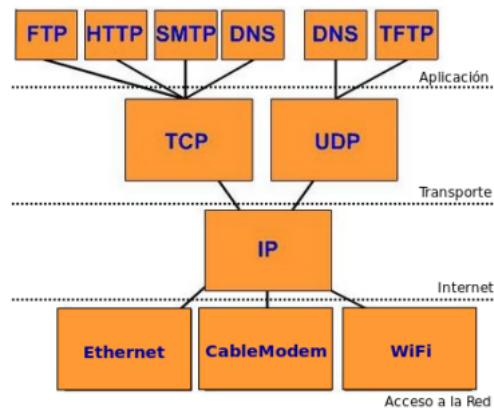
Capa de Transporte o Host-to-Host.

Capa de Internet o Internetworking.

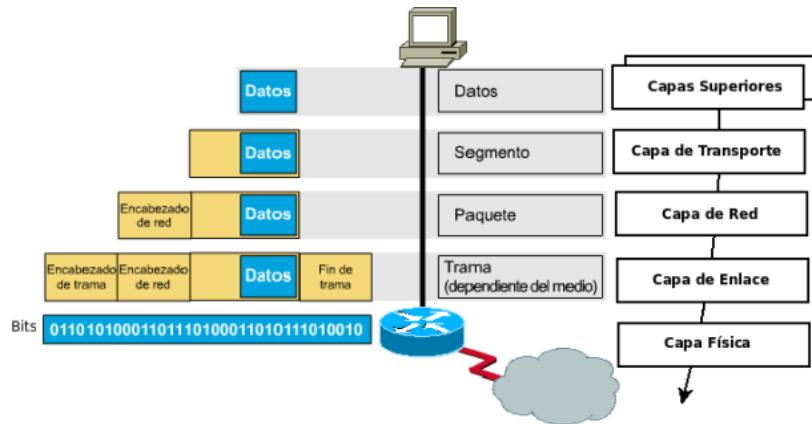
Capa de Enlace(Link Layer).

Capa de Física.

*Puede hablarse de 4 capas, agrupando la Capa de Enlace y la Capa Física dentro de una única capa, la Capa de acceso a la Red.*



. Cada capa define su PDU (Protocol Data Unit):

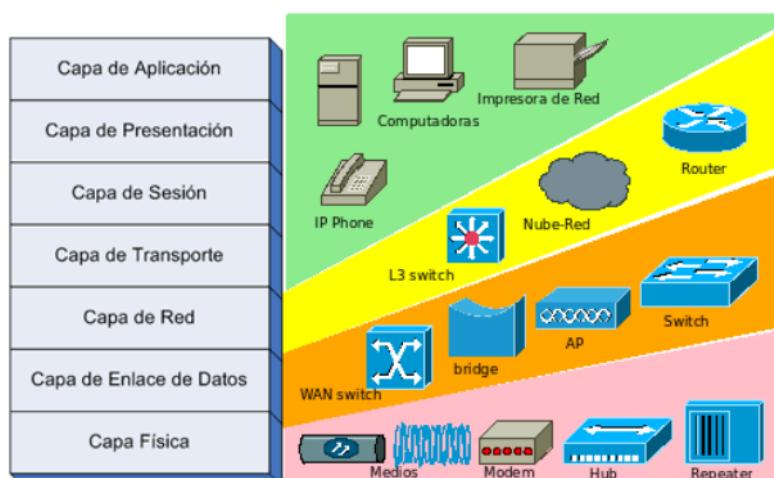


. Encapsulación:

La encapsulación en una capa de red es el proceso mediante el cual los datos son empaquetados con la información de control necesaria para que puedan ser transmitidos correctamente a través de una red.

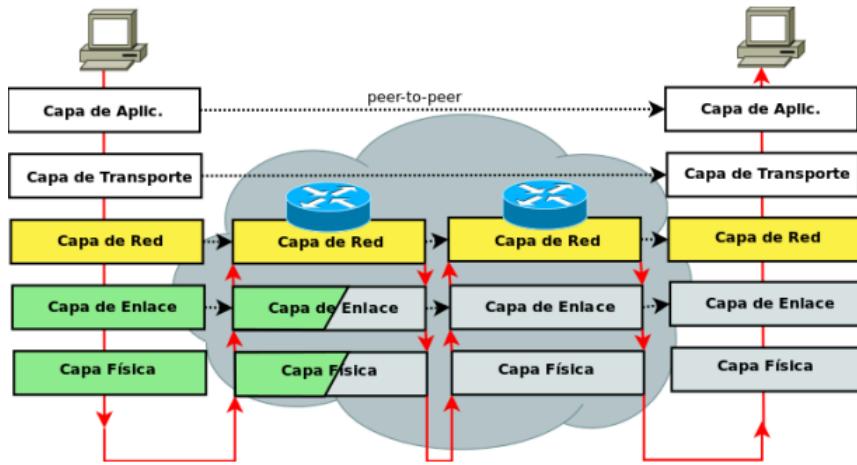
Cada capa en el nodo receptor realiza la decapsulación de la PDU de la capa inferior para recuperar los datos originales, procesándolos en la secuencia inversa a la encapsulación que se llevó a cabo en el nodo emisor.

. Dispositivos y Capas:



. *Comunicación entre capas Peer to Peer:*

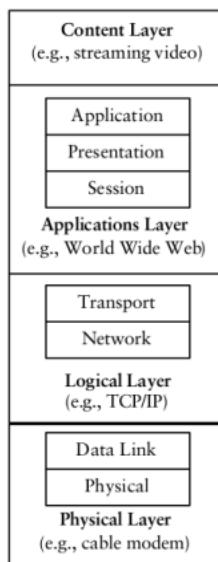
Cada capa usa el servicio de la de abajo y además se comunica con su capa par del otro extremo.



## **INTERNET**

. Internet es una red global de redes interconectadas que permite la comunicación y el intercambio de información entre dispositivos en todo el mundo. Es una red de redes de computadoras, descentralizada, pública, que ejecutan el conjunto abierto de protocolos (suite) TCP/IP. Integra diferentes protocolos de un nivel más bajo: INTERNETWORKING.

. Modelo simplificado de internet:



## **ESTRUCTURA DE INTERNET**

- . Estructura en Jerarquía, en Tiers.
- . Capa de Acceso (Edge): Acceso Residenciales, Acceso de Organizaciones.
- . Capa de núcleo (Core): dividida en diferentes niveles:  
Proveedores Regionales (Regional ISPs).

- Proveedores Nacionales.
- Proveedores Internacionales.
- Proveedores Internacionales en el Tier 1.

## ***ESTÁNDARES DE INTERNET RFCs***

- . Las RFC (Request For Comments) son documentos que establecen la definición de los protocolos.

## **Clase 1.2 | Introducción a la Capa de Aplicación**

### ***CAPA DE APLICACIÓN***

- . **Funciones:** Provee servicios de comunicación a los usuarios y a las aplicaciones, incluye las aplicaciones mismas. Define el formato y la semántica de los mensajes. Y también define cómo debe ser el código y qué mensajes se deben intercambiar.  
Las aplicaciones que usan la red pertenecen a esta capa, así como también los protocolos que implementan las mismas. Aunque también existen aplicaciones que NO son de red que deben trabajar con aplicaciones/servicios para lograr acceso a la red.
- . **Componentes:** Programas que corren en diferentes plataformas y se comunican entre sí, y los protocolos que implementan.
- . Las aplicaciones en la mayoría de los casos corren en los nodos finales(end-systems), no en el núcleo de la red → más fácil el desarrollo y uso, siguen principio end-2-end.
- . En el enfoque orientado a Internet la capa de Aplicación integra: Capa de Aplicación propiamente dicha del modelo OSI, Capa de Presentación del modelo OSI, Capa de Sesión del modelo OSI.

### ***CAPA DE SESIÓN***

- . **Función:** Administra las conversaciones/diálogos entre las aplicaciones. Podría proveer mecanismos transaccionales o de sincronización: COMMIT, CHECKPOINT, ROLLBACK. También maneja actividades e informa excepciones.
- . Se encuentra integrada en las aplicaciones de red mismas.
- . La Capa de Sesión podría verse como un invento de la ISO ya que ninguna de las redes existentes la tenían.
- . La Capa de Sesión es muy delgada, con relativamente pocas características, y en general no se utiliza.

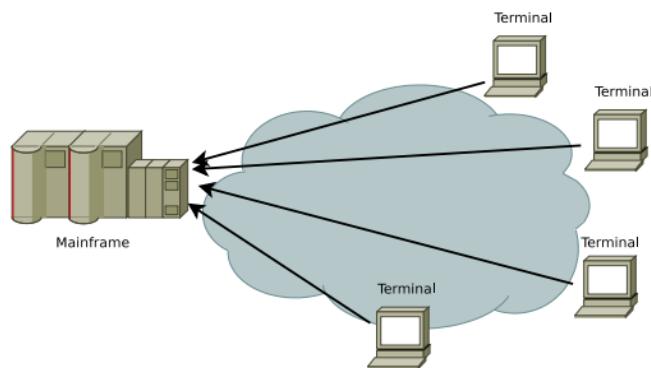
### ***CAPA DE PRESENTACIÓN***

- . **Función:** Conversión y codificación de datos a codificaciones comunes, como por ej: ASCII, EBDIC, charset ISO-8859-1, UTF-8, Unicode16. También compresión y descompresión de datos, cifrado y descifrado de datos. Y define formatos y algoritmos para esto: JPEG, MPEG, LZW, AES, DES, IDEA.

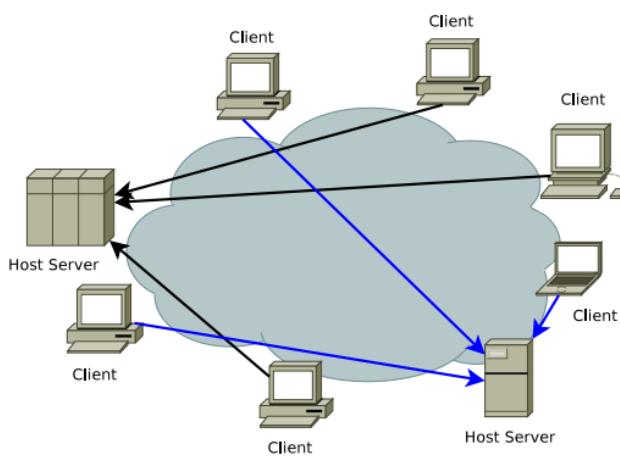
. Se encuentra integrada en las aplicaciones de red mismas.

## **MODELOS DE COMUNICACIÓN DE APLICACIONES**

. **Modelo Mainframe Centralizado (dumb client):** Es un modelo de carga centralizada donde el cliente es “tonto” (dumb) solo corre la comunicación y la interfaz física con el usuario (ej. terminal). El servidor pone todo el procesamiento. El mainframe es el que decide cuando le da el control al cliente y maneja el diálogo de las comunicaciones. El cliente ejecuta en el mainframe.

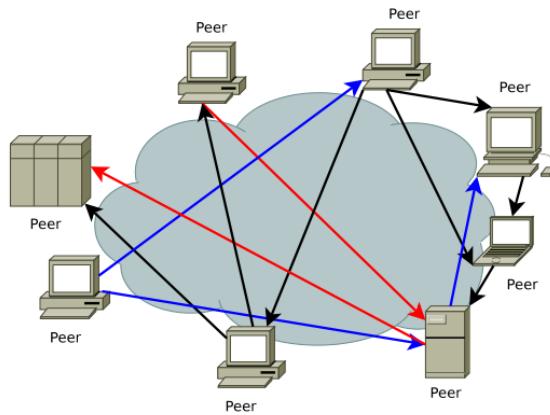


. **Modelo de Cliente/Servidor:** Es un modelo de carga compartida donde la idea inicial es que el cliente pone el procesamiento de interfaz. Luego el servidor pone el resto del procesamiento. Existen modelos en varios tiers (2 tiers, 3 tier o multi-tier). El servidor corre el servicio esperando de forma pasiva la conexión. Los clientes se conectan al servidor y se comunican a través de este. El modelo es asimétrico 1 a N, M a N (donde  $M < N$ ).



. **Modelo de Peer-to-Peer:** Es un modelo de carga completamente compartida y distribuida, donde los peers (participantes) pueden cumplir el rol de cliente, servidor o ambos en un instante. El sistema es escalable en cuanto a rendimiento, pero no es escalable en cuanto a administración.

Modelo asimétrico N a N.

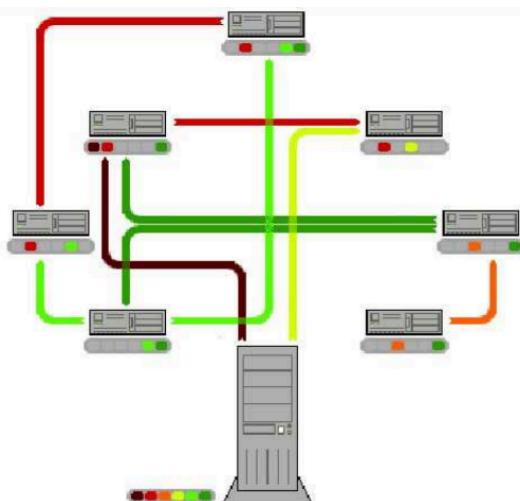


. **Modelo Peer-to-Peer Híbrido:** Es un modelo de carga compartida y distribuida, donde los peers (participantes) pueden cumplir el rol de cliente, servidor o ambos en un instante. Existen diferentes tipos de nodos con diferentes roles. Hay nodos centrales donde se registra la información y al resto de los nodos. El sistema es escalable en cuanto a rendimiento.

Modelo asimétrico M a N.

. **Bit Torrent:** Es un protocolo para el intercambio de archivos grandes de forma masiva como peer-to-peer (P2P). Cuando alguien quiere compartir un archivo genera un archivo *.TORRENT*.

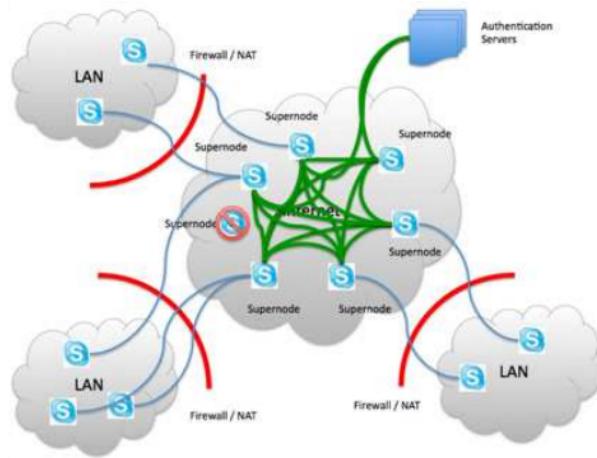
*.TORRENT* Dirección del TRACKER para unirse al grupo de PEERS (+ información del archivo). Desde el TRACKER se obtienen PEERS: las SEEDs contienen el archivo completo, y las LEECHERS contienen archivo parcial. El TRACKER se actualiza con el nuevo PEER. La comunicación con el TRACKER habitualmente se hace sobre HTTP, o podría ser sobre FTP. Los archivos se dividen en chunks, cada uno puede ser descargado de un PEER diferente. Luego se conecta con otros PEERS y comienza la descarga. Cada chunk se comparte con otros PEERS. Utiliza el port 6881 y escanea hacia arriba.



. **Skype**: Es un protocolo propietario que implementa VoIP, donde hay nodos clientes que pueden ser solo nodos o Super-nodos. Estos Súper-nodos actúan como directorios, mientras que los nodos hacen de STUN, permitiendo la conexión de computadoras detrás de NAT.

Los Super-nodos crean P2P Overlay networks (Una red sobre otra red) y corren el software tradicional de los clientes, pero sobre la Internet pública. Luego, estos Super-nodos se conectan entre sí utilizando servicios externos de autenticación.

Existen Mega Super-nodos que pertenecen a Skype.



## Clase 2 | Protocolo HTTP

### PROTOCOLO HTTP

. HTTP es un protocolo de comunicación utilizado para la transferencia de datos entre un cliente (como un navegador web) y un servidor web. HTTP define cómo se envían y reciben los mensajes en la web.

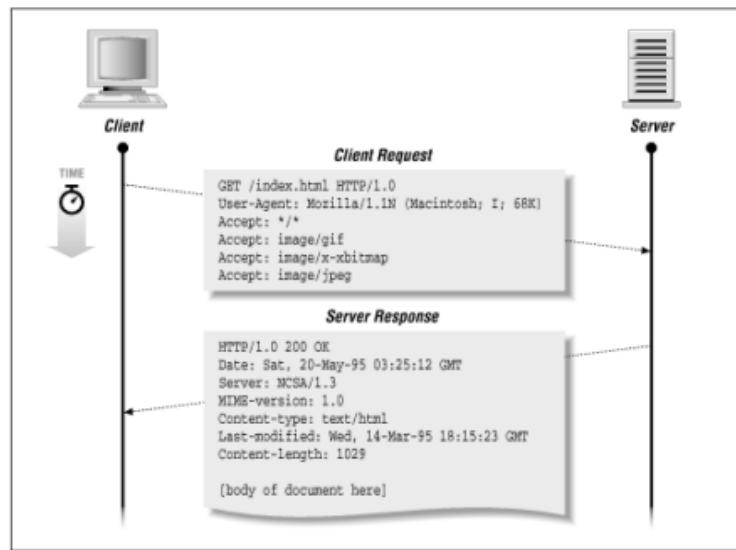
### ELEMENTOS WEB

- . **Recurso u Objeto HTTP**: ej. web page, imágenes JPEG, PNG, GIF, Java Applet, archivos de multimedia: MP3, AVI, etc.
- . **Referencia**: URI (Uniform Resource Id): URL (Uniform Resource Location) o URN (Name).  
Formato de URL: protocol://[user:pass@[host:[port]/[path]]. Ejemplo:  
<http://www.NN.unlp.edu.ar:8080/dir/index.html>.
- . **Páginas web**: archivo HTML que incluye vínculos o directamente otros objetos.

### FUNCIONAMIENTO HTTP

- . Modelo cliente/servidor, Request/Response (sin estados - stateless).
- . Protocolo que corre sobre TCP (requiere protocolo de transporte confiable), usa el puerto 80 por default.
- . El cliente escoge cualquier puerto no privilegiado. Trabaja sobre texto ASCII, y envía información binaria con encabezados MIME.

- . Clientes (llamados browsers o navegadores): Firefox, IE, Opera, Safari, Chrome.
- . Servidores: Apache Server, MS IIS, NGINX , Google GWS, Tomcat.



### **HTTP 0.9**

- . Solo una forma de Requerimiento.
- . Solo una forma de respuesta.
- . Request/Response sin estado.
- . Pasos:
  1. Establecer la conexión TCP
  2. HTTP Request via comando GET.
  3. HTTP Response enviando la página requerida.
  4. Cerrar la conexión TCP por parte del servidor.
  5. Si no existe el documento o hay un error directamente se cierra la conexión.

### **HTTP 1.0**

- . Se debe especificar la versión en el requerimiento del cliente.
- . Para los Request, define diferentes métodos HTTP.
- . Define códigos de respuesta.
- . Admite repertorio de caracteres, además del ASCII, como: ISO-8859-1, UTF-8, etc. Admite MIME (No solo sirve para descargar HTML e imágenes).
- . Por default NO utiliza conexiones persistentes.

. Request:

```
<Method> <URI> <Version>
[<Headers Opcionales>]
<Blank>
[<Entity Body Opcional>]
<Blank>

<Method HTTP 1.0> ::= GET, POST, HEAD, PUT,
DELETE, LINK, UNLINK
```

*Si tiene método, URL y versión es un mensaje de requerimiento.*

. Response:

```
<HTTP Version> <Status Code> <Reason Phrase>
[<Headers Opcionales>]
<Blank>
[<Entity Body Opcional>]
```

*Si indica la versión HTTP, el código de estado y la frase de razón, corresponde a una respuesta.*

El cliente se da cuenta de que ha recibido todo el objeto solicitado cuando el servidor cierra la conexión después de enviar la respuesta.

. Métodos:

**GET:** obtener el documento requerido. Puede enviar información, pero no demasiada. Es enviada en la URL. Formato ?var1=val1&var2=val2.... Limitación de tamaño de URL por parte de las implementaciones. NO espera recibir datos en body.

**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.

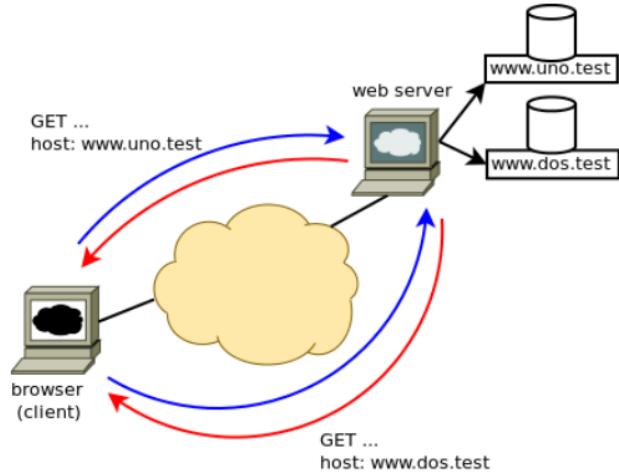
**LINK, UNLINK:** establecen/des-establecen relaciones entre documentos.

. Hosts Virtuales:

Permiten tener distintos sitios en un mismo host. Estos distintos sitios se pueden acceder en la misma dirección, ya que están en el mismo servidor.

Cuando al servidor le llega el requerimiento, puede enviar al cliente a uno o a otro de los hosts virtuales.

Ambos sitios pueden estar en la misma IP.

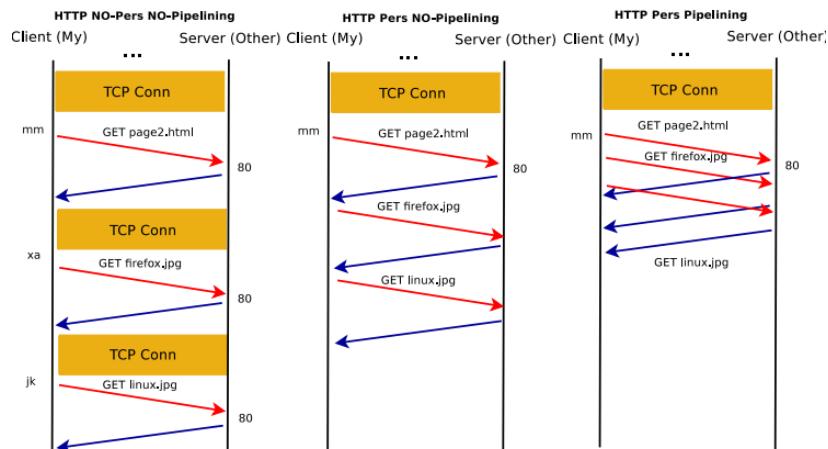


. Autenticación:

- . Encabezados, el cliente y el servidor intercambiar información auth.
- . El servidor, ante un requerimiento de un documento que requiere autenticación, enviará un mensaje 401 indicando la necesidad de autenticación y un Dominio/Realm.
- . El navegador solicitará al usuario los datos de user/password (si es que no los tiene cacheados) y los enviará en texto claro al servidor.
- . El servidor dará o no acceso en base a esos valores.

## HTTP 1.1

- . Adopta nuevos mensajes HTTP 1.1: OPTIONS, TRACE, CONNECT.
- . Implementa conexiones persistentes por omisión.
- . Implementa 'Pipelining', lo cual mejora el tiempo de respuestas:  
No necesita esperar la respuesta para pedir otro objeto HTTP.  
Solo se utiliza con conexiones persistentes.  
Sobre la misma conexión debe mantener el orden de los objetos que se devuelven.  
Se pueden utilizar varios threads para cada conexión.



El cliente se da cuenta de que ha recibido todo el objeto solicitado a partir del encabezado "ContentLength" que indica al cliente la longitud en bytes del objeto en la respuesta, permitiéndole saber cuántos datos esperar. El encabezado "Transfer-Encoding" con valor

"chunked" divide la respuesta en trozos, y el cliente detecta el final de la respuesta cuando recibe un trozo de tamaño 0.

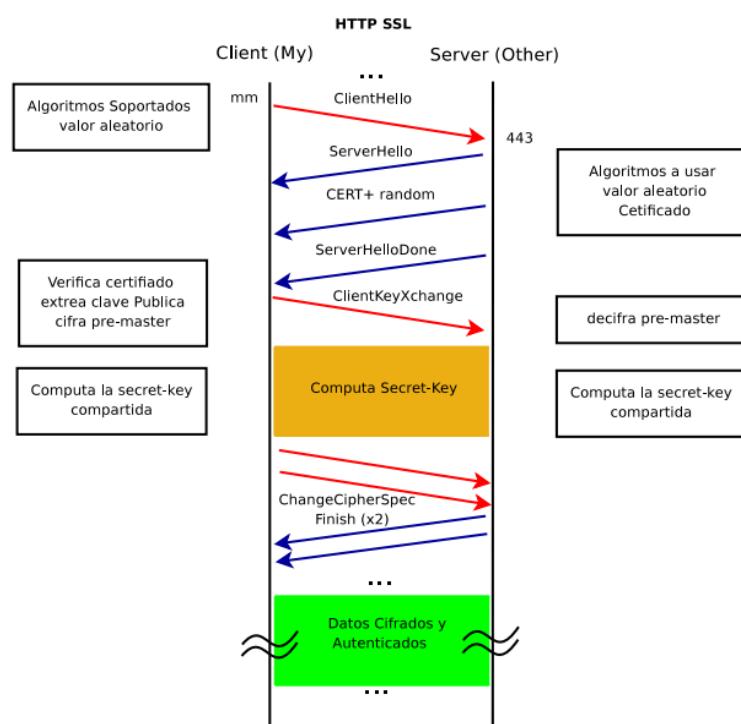
## COOKIES

- . Es un mecanismo que permite a las aplicaciones web del servidor "manejar estados", ya que HTTP no tiene estado.
- . El cliente hace un request, el servidor retorna un recurso indicando al cliente que almacene determinados valores por un tiempo. La Cookie es introducida al cliente mediante el mensaje en el header `Set-Cookie`, el cual indica un par (nombre,valor). El cliente en cada requerimiento luego de haber almacenado la Cookie se la enviará al servidor con el header `Cookie`.

El servidor puede utilizarlo o no, y también puede borrarlo.

## HTTPS

- . Utiliza el port 443 por default.
- . Tiene una etapa de negociación previa. Luego se cifra y autentica todo el mensaje HTTP (incluso el header).



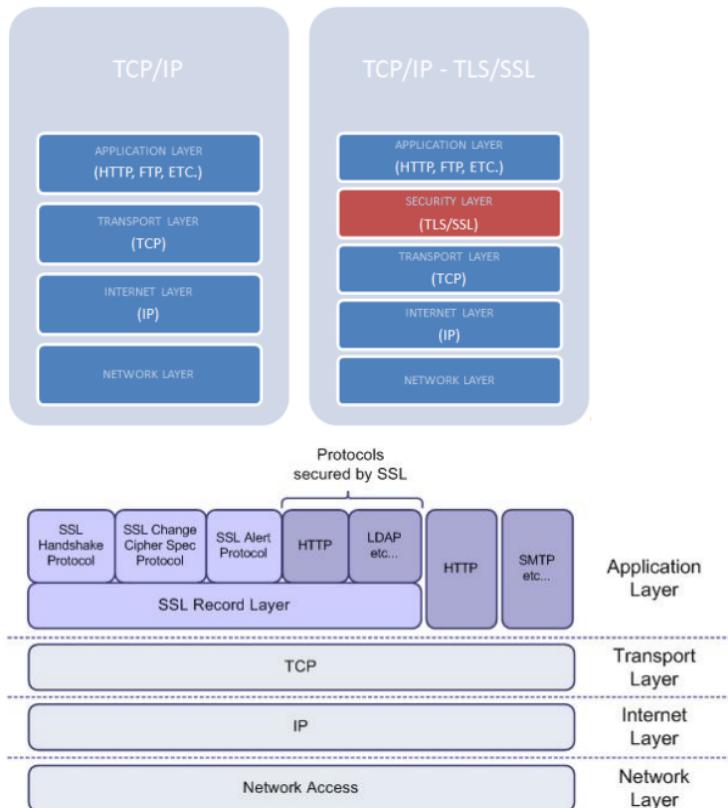
## WEB-CACHE

- . "Proxiar" y Cachear recursos HTTP.
- . Su objetivo es mejorar tiempo de respuesta (reducir retardo en descarga), ahorrar BW (recursos de la red), y mantener un balance de carga y atender a todos los clientes.

- . Funcionamiento: Se solicita el objeto, si está en cache y está “fresco” se retorna desde allí (HIT). Si el objeto no está o es viejo se solicita al destino y se cachea. Se puede realizar control de acceso.
- . La cache del lado del cliente es privada.
- . Los web browser tienen sus propias cache locales.

## **HTTP/2**

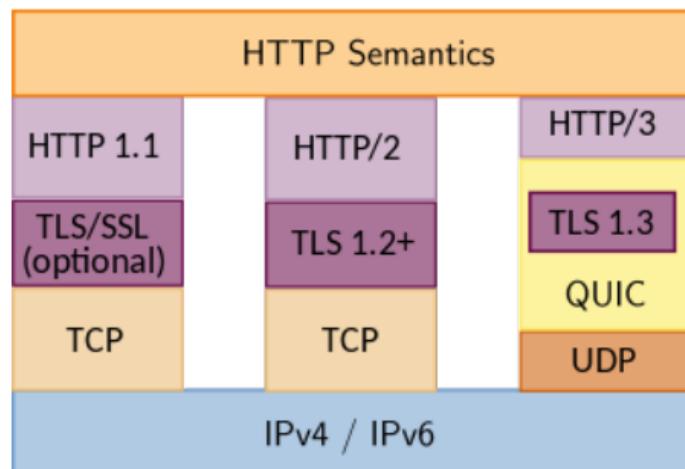
- . Reemplazo de cómo HTTP se transporta.
- . Se conservan métodos y semántica.
- . *Problemas con HTTP/1.0, HTTP/1.1:*
  - Un request por conexión, por vez, muy lento.
  - Pipelining requiere que los responses sean enviados en el orden solicitado, HOL posible.
  - POST no siempre pueden ser enviados en pipelining.
  - Demasiadas conexiones generan problemas, control de congestión, mal uso de la red.
  - Muchos requests, muchos datos duplicados (headers).
- . *Diferencias principales con HTTP/1.1:*
  - Protocolo binario en lugar de textual(ASCII), binary framing: (más eficiente).
  - Multiplexa varios request en una petición en lugar de ser una secuencia ordenada y bloqueante.
  - Usa compresión de encabezado.
  - La mayoría de las implementaciones requieren TLS/SSL, no el estándar.
- . *HTTP/2 mux stream, framing:*
  - Puede generar una o más conexiones TCP. Trata de aprovechar las que tiene establecidas.
  - Un stream es como una sub-conexión (una “conexión” http2 dentro de una conexión TCP) que transporta mensajes. Cada uno tiene un ID y una prioridad(alternativa) y son bidireccionales.
  - Sobre una conexión TCP multiplexa uno o más streams (“conexiones http2”).
  - Los mensajes http2 son divididos en frames dentro del mismo stream. Cada uno de esos frames es una porción de mensaje: header fijo+payload variable (unidad mínima).
  - El mismo stream puede ser usado para llevar diferentes msj.
  - Todos los streams van en una misma conexión, para luego ser identificados y divididos en frames.
- . *Headers:*
  - Se mantienen casi todos los headers de HTTP/1.1, pero no se codifican más en ASCII.
  - Por ejemplo: HEAD /algo HTTP/1.1 se reemplaza con http2:
    - :method: head
    - :path: /algo
    - :scheme: https o http
    - :authority: www.site.com reemplaza al headerHost:..
- . *TLS/SSL:*
  - Agrega una capa de seguridad al protocolo TCP/IP, y se encuentra en la capa de aplicación.



## HTTP/3

- . Transporta HTTP sobre un nuevo protocolo: QUIC, que corre sobre UDP, default UDP:443.
- . Al igual que HTTP/2 conserva métodos y semántica de HTTP/1.1.
- . *Diferencias principales de Http/3-Quic:*
  - Reducción de latencia en conexiones (2 msj. de handshake).
  - No depende del manejo de la conexión y controles de TCP (evita HOL).
  - QUIC genera nuevos números de secuencia y re-cifra las retransmisiones, permitiendo mejor detección de pérdidas y medición de RTT.
  - QUIC utiliza paquetes cifrados de forma individual, no por conexión/bytestream.
  - QUIC facilita movilidad, tiene ID de conexión independiente de IPs y ports.
  - UDP facilita algunos ataques de seguridad.

*En http/2 y http/3 los encabezados se mantienen, al igual que los métodos y semántica.  
Solo cambia la forma de transporte.*



### **PROTOCOLOS BINARIOS VS BASADOS EN TEXTO**

- . Un protocolo binario transmite datos en forma de patrones de bits que las máquinas pueden entender directamente. Suele ser más eficiente en términos de velocidad y uso de ancho de banda, ya que no se desperdicia espacio en caracteres de formato o en la interpretación de texto legible por humanos.
- . Un protocolo basado en texto envía datos como cadenas legibles utilizando caracteres como letras, números y símbolos. Estos caracteres suelen estar codificados en ASCII o UTF-8. Aunque esto hace que la comunicación entre sistemas sea más comprensible para los humanos, puede ser menos eficiente en términos de velocidad y uso de ancho de banda, ya que se requiere más información para representar los mismos datos que en formato binario.
- . HTTP/1.0, HTTP/1.1 son protocolos basados en textos mientras que HTTP/2 es un protocolo binario

### **Clase 3 | Protocolo DNS**

#### **PROTOCOLO DNS**

- . El **DNS** (Domain Name System o Sistema de Nombres de Dominio) es un sistema que traduce nombres de dominio legibles para los humanos (como [www.ejemplo.com](http://www.ejemplo.com)) en direcciones IP numéricas (como [192.168.1.1](http://192.168.1.1)), que son necesarias para que los dispositivos en Internet puedan localizarse entre sí.
- . Entonces, ante la necesidad de utilizar nombres en lugar de direcciones IP, el protocolo DNS implementa un mecanismo para mapear nombres de internet (dominios) a direcciones IP.
- . Este servicio DNS funciona como un sistema distribuido de forma jerárquica, a través de dominios, sub-dominios y nombres finales, con un conjunto de servidores a lo largo del mundo, donde cada servidor tiene la responsabilidad de mantener una parte dentro de la jerarquía de nombres.
- . Posee y utiliza *jerarquías de dominios* distribuidos entre servidores.
- . En principio las consultas y búsquedas se realizan por IP.

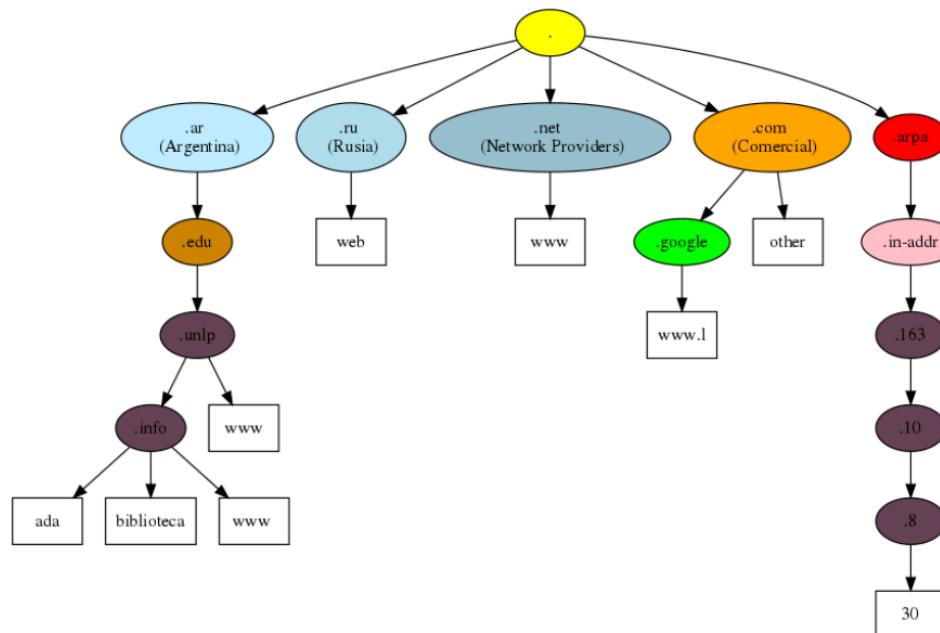
. Aspectos y elementos de DNS:

- Espacio de nombres (sintaxis y zonas, dominios).
- Procedimiento de Delegación y Arquitectura.
- Base de datos distribuida y Servidores.
- Define las componentes y el protocolo para su comunicación.
- Procedimiento de búsqueda/resolución (Protocolos).

. Nombre de dominio FQDN (Nombre de Dominio Completo):

- Lista de etiquetas (labels) separadas por puntos.
- Los TLDs (Top Level Domain), que son la última parte de un nombre de dominio, es decir, lo que sigue inmediatamente después del último punto en un dominio, se pueden clasificar en 3 grupos:
  - . Los dominios con propósitos particulares, **Generic TLDs (gTLDs)**. Estos propósitos van de acuerdo a diferentes actividades políticas definidas por el ICANN (Unsponsored TLD) o definidas por otra organización (Sponsored TLD). Son los dominios que básicamente se utilizan para identificar categorías amplias y distintos tipos de sitios web. Algunos ejemplos de gTLDs incluyen .com, .org, .net y .info.
  - . Los dominios asignados a los distintos países, **Country-Code TLDs (ccTLDs)**.
  - . Dominio especial utilizado internamente para resolución de reversos, .ARPA TLD.

. Esquema de nombres DNS:



. Ejemplo - Delegación de autoridad:

Dominio: `ada.info.unlp.edu.ar`.

“Ada” fue registrada por la administración de la red de la Facultad de Informática. El administrador de la Facultad obtuvo previamente la autoridad sobre el dominio “`info.unlp.edu.ar`”. a partir de la administración de la universidad UNLP.

La Universidad obtuvo autoridad sobre el dominio “`unlp.edu.ar`” a partir de la administración de “`edu.ar`”, RIU (Red Inter-universitaria).

La RIU obtuvo autoridad sobre “`edu.ar`” a partir de la delegación de NIC.AR que depende de alguna organización del gobierno, como la Sec. Legal y Técnica u otro ente a cargo de “`.AR`” (Argentina).

La administración de nombres en la Argentina, sea la Secretaría Legal y Técnica u otro ente obtuvo la autoridad delegada a partir del IANA o ICANN.

. *Funcionamiento de DNS:*

Modelo cliente/servidor, Request/Response.

También hay diálogo entre los servidores.

El protocolo corre sobre UDP y TCP, puerto 53.

El cliente escoge cualquier puerto no privilegiado.

No trabaja sobre texto ASCII.

Si el mensaje supera los 512 bytes se utiliza TCP, e.g. zone transfer (EDNS permite mayor cantidad de datos).

. *Consultas:*

Hay un servidor local contra quien (yo) realizo las consultas. Este servidor local cachea la información, la cual puede quedar desactualizada.

Entonces, si el servidor local tiene la información requerida en la caché, me la devuelve automáticamente. Si no es así, propaga la consulta con otros servidores externos.

. *Tipos de consultas:*

Cuando el cliente realiza una *consulta recursiva*, la realiza esperando el resultado final.

En este tipo de respuesta, el servidor DNS al que se le solicita la resolución de un dominio se encarga de hacer todo el trabajo. Si no conoce la respuesta, contacta con otros servidores DNS en nombre del cliente, obteniendo la respuesta final. El cliente solo recibe la respuesta completa y no interactúa con otros servidores DNS.

Mientras que cuando se realiza una *consulta iterativa*, se consulta por IPs para ir acercándose a la respuesta final. El cliente va recibiendo referencias parciales.

El cliente DNS interactúa con múltiples servidores DNS, haciendo varias solicitudes hasta obtener la respuesta final. El primer servidor consultado no resuelve completamente la consulta, sino que da una pista de a dónde debe ir el cliente para continuar la búsqueda (es decir, le proporciona la dirección de un servidor DNS más específico).

\* Ver slides 27 a 36 de clase 3 | DNS

. *Respuestas:*

Una **respuesta autoritativa** es aquella dada por el servidor que tiene la autoridad sobre el nombre que se está consultando. Este responde directamente desde su base de datos de nombres, sin subdelegaciones ni cacheo de direcciones.

Caso contrario, si se realiza esto último, se trata de una **Respuesta NO Autoritativa**.

. *Tipos de Servidores:*

**Servidor Raíz:** servidor que delega a todos TLD (Top Level Domains). No debería permitir recursivas.

**Servidor Autoritativo:** servidor con una zona o sub-dominio de nombres a cargo. Podría sub-delegar.

**Servidor Local/Resolver Recursivo:** es un servidor que es consultado dentro de una red. Mantiene cache. Puede ser Servidor Autoritativo. Permite recursivas “internas”. También llamado Caching Name Server.

**Open Name Servers:** servidores de DNS que funcionan como locales para cualquier cliente, desde cualquier lugar del mundo.

**Forwarder Name Server:** interactúan directamente con el sistema de DNS exterior. Son DNS proxies de otros DNS internos.

**Servidor Primario y Secundario:** solo una cuestión de implementación. Es donde se modifican los datos realmente.

Los **root servers** son los encargados de proporcionar las direcciones IP de los Top Level Domains (la parte más alta de la jerarquía luego de la raíz).

. *Resolver:*

El Resolver se lo podría considerar como un agente encargado de resolver los nombres a solicitud del cliente. Se puede tener un Stub/Dumb Resolver que no realiza ninguna forma de caching y deja que el encargado de esto sea el Servidor Local o un resolver activo, llamado Smart Resolver, que funciona en cada equipo como si fuese un Servidor Local, realizando caching u ofreciendo funcionalidades extras. Este suele hacer consultas recursivas.

. *Servidores primarios y secundarios:*

El servidor primario es quien tiene toda la configuración, y el secundario funciona como una copia del primario. Esta sincronización se da por un mecanismo definido por el protocolo DNS.

El servidor secundario solo admite consultas sobre un registro en particular ya que tiene que solicitarle la información al primario. Es decir, no se podría consultar por toda la información.

## Clase 4 | Mensajería en Internet (E-Mail) - Protocolo FTP

### **PROTOCOLO FTP (*File Transfer Protocol*)**

. Es un protocolo para copiar archivos completos, a diferencia de otros protocolos que brindan acceso a archivos: NFS, CIFS.

Es un protocolo que corre sobre TCP (requiere protocolo de transporte confiable).

. *Funcionamiento FTP:*

Usa 2 (dos) conexiones TCP:

Conexión de Control (Out-Of-Band Control) port 21.

Conexión para la transferencia de datos.

Cada conexión requiere servicios diferentes:

Conexión de Control: min delay.

Conexión de Datos: max throughput.

Se trabaja con sesiones que se mantienen.

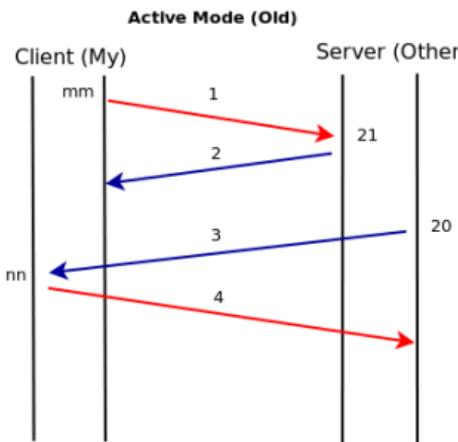
El cliente escoge cualquier puerto no privilegiado, ( $n > 1023$ ) y genera conexión de control contra el puerto 21 del servidor.

El servidor recibe los comandos por dicha conexión y responde/recibe por la conexión de datos aquellos que lo requieran.

La conexión de datos se crea y se cierra bajo demanda.

El estado de cada operación se transmite por el canal de control.

Funcionamiento en modo activo:



#### *. Comandos FTP:*

**RETR**: obtener un archivo desde el servidor. A nivel de interfaz de usuario el comando que lo inicia es el get.

**STOR**: envía un archivo al servidor. A nivel de interfaz de usuario el comando que lo inicia es el put.

**LIST**: envía una petición de listar los archivos del directorio actual en el servidor. A nivel de interfaz de usuario el comando que lo inicia es el ls o dir.

**DELE**: comando para borrar un archivo en el servidor.

**SIZE, STAT**: obtiene información de un archivo en el servidor.

**CD, PWD, RMD, MKD**: cambia de directorio, obtiene el dir. actual, borra y crea dir.

#### *. Modalidades FTP:*

##### **FTP Activo** (modalidad vieja):

Conexión de control: port 21.

Conexión de datos: port 20.

Se diferencia cómo maneja la conexión de datos.

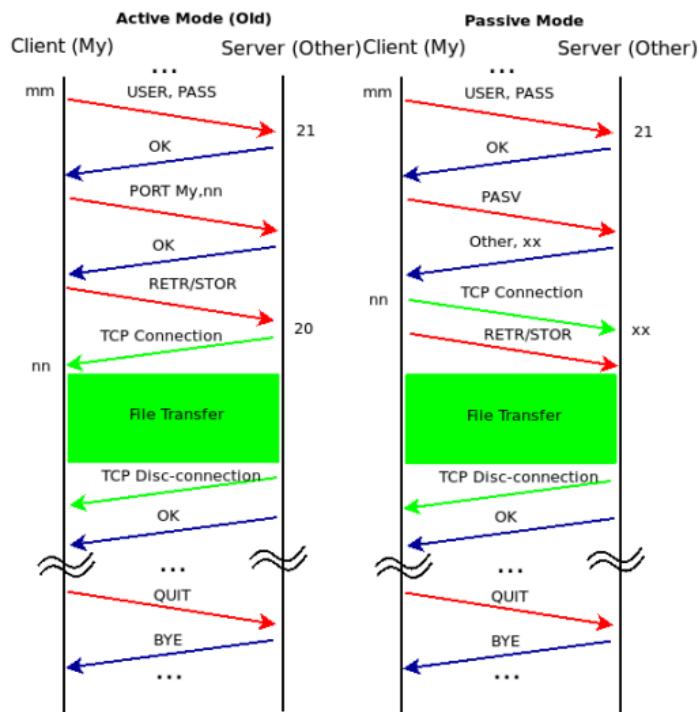
El servidor de forma activa se conecta al cliente para generar la conexión de datos.

##### **FTP Pasivo**:

Conexión de control: port 21.

Conexión de datos: port no privilegiado.

El servidor de forma pasiva indica al cliente a qué nuevo puerto debe conectarse.



#### . Formato de Datos FTP:

FTP tiene funcionalidad de la capa ISO L6( representación).

Debido a los diferentes tipos de plataformas, los archivos pueden ser convertidos a diferentes representaciones.

Es responsabilidad del cliente indicarle al servidor el tipo/formato, sino el default es ASCII, aunque hoy es más común encontrar image.

#### . Formato de Archivo FTP:

Las plataformas (OS) pueden almacenar los archivos en diferentes estructuras.

FTP define estructuras para transportar datos.

Formato default File (F). Se especifica el formato para transferencia con el comando STRU.

#### . FTP vs. HTTP:

FTP	HTTP
Diseñado para Upload	Se puede con PUT, POST
Soporte de Download	Soporte de Download
Formato ASCII/binary cliente selecciona	meta-data with files, Content-Type, más flexible
No maneja Headers	Manjea Headers, mas información
FTP Command/Response, archivos pequeños más lento	Pipelining
Más complejo con Firewalls y NAT	Más amigable con Firewalls y NATs
Dos conexiones, y modo Activo o Pasivo	Una conexión, más sencillo
Soportan Ranges/resume	Range/resume HTTP opciones más avanzadas
Soporte de Autenticación	Soporte de Autenticación
Soporte de Cifrado (problemas fwall)	Soporte de Cifrado
Soporte de Compresión RLE	Soporte de Compresión Deflate: LZ77+Huffman

## **Email**

### *. Arquitectura:*

1. El cliente de correo (MUA) contacta a su servidor de correo depositando su Mail User Agent, el cual se identifica con un nombre. El cliente tiene configurado cuál es su servidor de correo saliente (su servidor local).

El servidor hace un cambio de rol y se conecta vía SMTP al agente remoto.

2. Entonces, esta información (el mail) es enviada a alguna cuenta que está en otro servidor, un servidor remoto. Este envío lo hace el servidor de correo (no el cliente). Previo al envío, el servidor local mapea el nombre a la IP correspondiente y se contacta con el servidor remoto utilizando el protocolo SMTP.

3. Luego, una vez que el mail llega al MailBox, se utiliza el protocolo POP o IMAP para recuperar esa información y poder consumirla.

### *. Protocolos:*

Entre el cliente y su servidor de correo, y entre servidores de correo se comunican utilizando el protocolo SMTP.

Por otro lado, para la recepción de email se cuenta con 2 opciones, POP3 e IMAP:

#### **• POP3 (Post Office Protocol versión 3):**

- **Simplicidad:** POP3 es un protocolo de acceso a correo extremadamente simple. Su simplicidad lo hace fácil de implementar y utilizar.

- **Descarga y Borrado:** POP3 generalmente se configura para descargar los correos electrónicos desde el servidor a la máquina local del usuario. En este modo, los correos se eliminan del servidor después de la descarga (aunque se pueden configurar para mantener una copia en el servidor).

- **No Mantiene Estado:** POP3 no mantiene información de estado entre sesiones. Esto significa que no guarda información sobre carpetas, mensajes marcados o cualquier otra información relacionada con el estado de la cuenta del usuario en el servidor.

- **Limitado para Usuarios Nómadas:** Para usuarios que desean acceder a sus correos electrónicos desde múltiples dispositivos, POP3 puede ser limitante ya que no ofrece una forma sencilla de sincronizar carpetas y correos entre dispositivos.

#### **• IMAP (Internet Message Access Protocol):**

- **Funcionalidad Avanzada:** IMAP es más avanzado que POP3 y ofrece una amplia gama de funcionalidades. Permite a los usuarios organizar correos electrónicos en carpetas remotas, buscar mensajes, mover mensajes entre carpetas y realizar otras acciones avanzadas.

- **Mantiene Estado:** IMAP mantiene información de estado en el servidor. Esto significa que las carpetas, los mensajes marcados como leídos/no leídos, y otras acciones realizadas en un dispositivo se reflejan en todos los dispositivos conectados, lo que lo hace ideal para usuarios nómadas.

- **Acceso a Partes Componentes de los Mensajes:** IMAP permite a los usuarios acceder a partes específicas de los mensajes, como la cabecera o partes de un mensaje MIME. Esto es útil cuando se necesita descargar solo partes específicas de un mensaje para ahorrar ancho de banda.

- **Complejidad Adicional:** Debido a su mayor funcionalidad, IMAP puede ser más complejo de implementar tanto en el lado del cliente como en el lado del servidor en comparación con POP3.

*Componentes:*

Componentes Principales:

- **MUA (Mail User Agent):** Es el cliente de correo electrónico (Gmail, Outlook, Thunderbird), y mantiene una Interfaz con el usuario. Es , además, Lector, Editor y Emisor local de correos (e-mails), y posee integrado un local MTA para comunicarse con el Servidor de Mail Saliente (MTA que hace relay).

El MTA que hace relay para el usuario pre-procesa el e-mail recibido desde el MUA con el agente MSA (Mail Submission Agent). Este, agrega la mayoría de los campos del header, y posee integrado un MRA para comunicarse con el Servidor de Mail Entrante, MAA (Mail Access Agent).

Utilizan protocolos SMTP o ESMTP, POP o IMAP, habla con MTA y con MAA propio.

- **MSA (Mail Submission Agent):** Actúa como cliente y servidor. Es un agente que habitualmente se encuentra integrado en el MTA, y que recibe el mensaje del MUA y lo pre-procesa antes de pasarlo al MTA para que haga el transporte.

En ese pre-procesamiento agrega campos que pueden faltar y termina de dar formato al header del e-mail.

Utiliza protocolo SMTP, ESMTP. Habitualmente usaba el port 25, se recomienda usar 587(submission).

- **MTA (Mail Transport Agent):** Actúa como cliente y servidor. Toma el e-mail desde el MSA o directamente desde el MUA (MSA integrado) y se encarga de enviar el mensaje de e-mail al servidor donde está la casilla de mensaje destino, utilizando comunicación de MTA a MTA.

Almacena temporalmente el correo saliente, encargándose de recibir y almacenar temporalmente los mensajes para las casillas que sirve desde el MTA remoto.

Utiliza protocolo SMTP o ESMTP, entre servidores MTA.

- **MDA (Mail Delivery Agent) o LDA (Local Delivery Agent):** Actúa como servicio interno o servidor. Se encarga de tomar los e-mails recibidos por el MTA (acepta mensajes del MTA) y de llevarlos al mailbox del usuario local.

Habitualmente define el formato del mailbox, ya que el Servicio de MDA puede hacer delivery remoto, cumpliendo el rol de MAA.

Se integra con los protocolos POP y/o IMAP dejando los recursos disponibles al MAA o directamente al usuario.

- **MAA (Mail Access Agent):** Actúa como Servidor. Puede estar integrado o separado del MDA, y su función es autenticar al MUA/usuario y leer los e-mails del mailbox local dejados por el MDA/LDA. Además, se encarga de transportar los e-mails hacia el MUA o los hace accesibles a este.

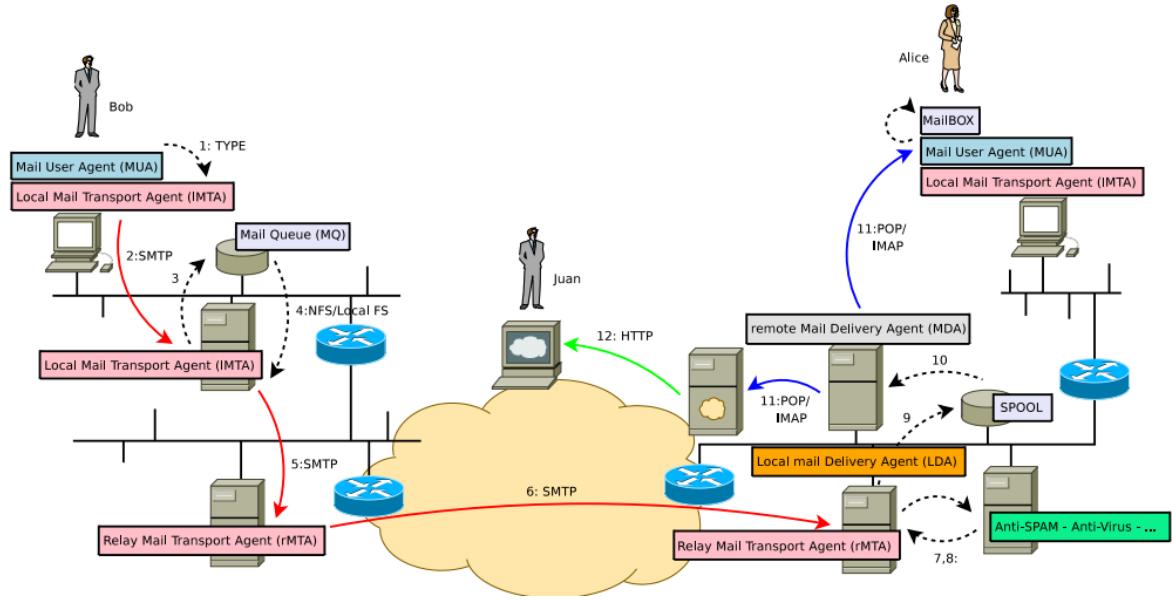
Se integra con el MDA.

Implementa los protocolos POP y/o IMAP dejando acceder a los recursos y dialogando con el MUA.

- **MRA (Mail Retrieval Agent).**

Componentes Secundarias:

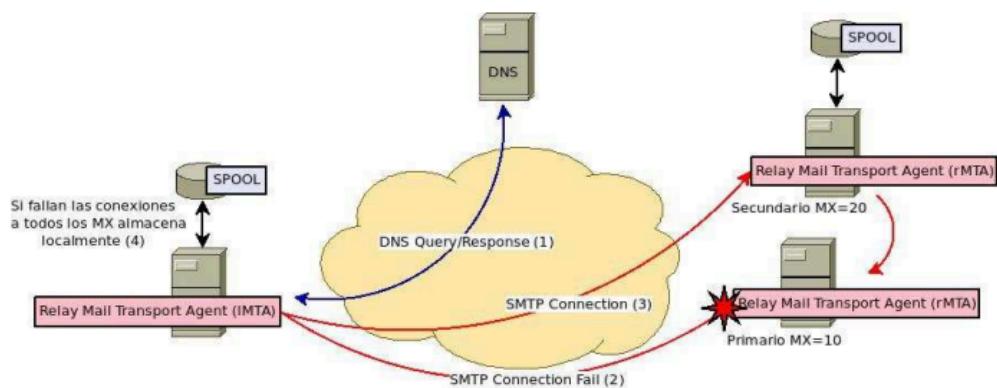
- Servidor de Autenticación externo.
- Web Mail (Front-End WWW).
- Servidor de Anti-Virus, Anti-SPAM, Servidores de Listas, etc.



### SMTP (Simple Mail Transport Protocol)

- . Es un protocolo Cliente/Servidor, que utiliza formato ASCII 7 bits en 8 NVT. Y que usa TCP puerto servidor: 25.
- . Los LMTA de los MUA hablan SMTP con su servidor SMTP saliente. Los servidores SMTP hablan entre sí este protocolo.
- . Usa conexiones persistentes.
- . Trabaja de forma Interactiva (Requerimiento/Respuesta) o Pipeline.
- . Puede o no requerir autenticación.
- . Puede o no trabajar de forma segura: SSL/TLS.

### SMTP y DNS



### PROTOCOLOS DE ACCESO A CORREO

- . POP: Post Office Protocol, RFC-1939: POPv3.
- . IMAP: Internet Mail Access Protocol, RFC-1730: IMAPv4.
- . Ambos requieren autenticación, y utilizan formato ASCII 7 bits en 8 NVT.
- . Usan TCP puertos servidor: 110 y 143.

- . Permiten correr de forma segura sobre SSL/TLS.
- . IMAP es más flexible ya que permite el uso de carpetas y la manipulación de mensajes en el servidor.

## **Clase 5 | Capa de Transporte**

### **CAPA DE TRANSPORTE**

. Es la 4ta capa del modelo OSI y del TCP/IP. Esta capa se encarga de gestionar la transmisión de mensajes o datos a través de una red, asegurándose de que lleguen de manera correcta y en el orden adecuado. Es decir, garantiza la entrega confiable de datos entre sistemas finales.

Proporciona, entonces, una comunicación lógica entre procesos de aplicación que se ejecutan en dispositivos diferentes dentro de una red. Esta comunicación lógica permite que los procesos de aplicación se envíen mensajes entre sí sin preocuparse por los detalles de la infraestructura física subyacente.

- . Brinda servicio a la capa de aplicación y usa servicios de la capa de internet.
- . Realiza la encapsulación, donde define PDU donde se envían los mensajes de la aplicación.
- . Soporta datos de tamaños arbitrarios.
- . Realiza control de flujo y de congestión.
- . Puede utilizar 2 modelos:
  - TCP (confiable).
  - UDP (no confiable): más rápido y con menos overhead.

Una aplicación puede elegir cuál de los 2 modelos utilizar.

### **PROTOCOLO DE TRANSPORTE TCP**

- . Transport Control Protocol.
- . Es un protocolo confiable, ordenado, con buffering, control de errores, de flujo y de congestión.
- . Está orientado a Streams, por lo que su PDU es segmento, una porción del stream de bytes.
- . Provee MUX/DEMUX.
- . Incrementa Overhead end-to-end para ofrecer confiabilidad.
- . Requiere establecimiento de conexión (y cierre).
- . Sus aplicaciones son transferencia de archivos, FTP/HTTP/SMTP/acceso remoto(SSH, telnet,...)/Unicast.
- . Este protocolo corre solo en los end-points (hosts).
- . La comunicación lógica es HOST-TO-HOST (END-TO-END), comunica procesos.
- . Transporte (TCP,UDP, u otro) transporta al mensaje de aplicación hasta el proceso (puerto), donde los paquetes (segmentos) son llevados por la red.
- . El encabezado IP provee: Ruteo, Fragmentación, Detección de algunos errores, pero no hace controles.
- . El encabezado TCP provee: MUX/DEMUX de aplicación, Detección de errores (obligatorio) y controles para hacerlo confiable.
- . IP indica que lleva TCP con el código de protocolo 6.

- . El protocolo TCP/IP usa IP como servicio.
- . Utiliza puertos para distinguir las aplicaciones (y, por lo tanto, protocolos) que están enviando/recibiendo datos. Los puertos actúan como puntos finales en una comunicación y permiten que múltiples aplicaciones en una misma computadora o dispositivo se comuniquen simultáneamente a través de la red.

. *Protocolo IP:*

Es utilizado en la capa de red, es un protocolo débil ya que no tiene la capacidad de resolver algunos problemas que pueden suceder como pérdida de paquetes, información duplicada, etc.

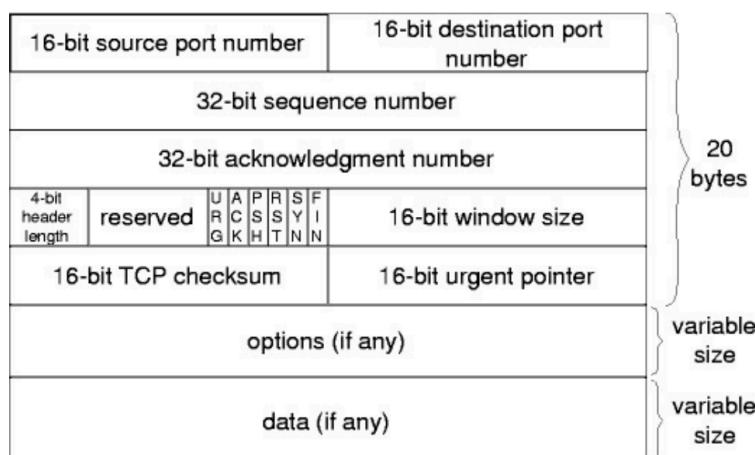
En IP, por ejemplo, los paquetes/datagramas pueden ser descartados, desordenados, retardados, duplicados o corrompidos.

Este protocolo corre en “todos” los nodos de la red (internet) (routers y host).

La comunicación lógica es HOP-BY-HOP, comunica hosts.

Entonces, la Red (IP), dirige hosts, transporta al mensaje del transporte hasta el host (dir IP).

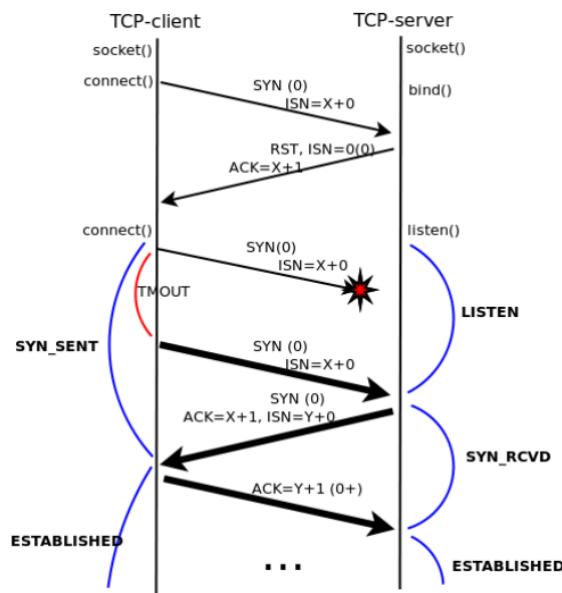
. *Segmento TCP:*



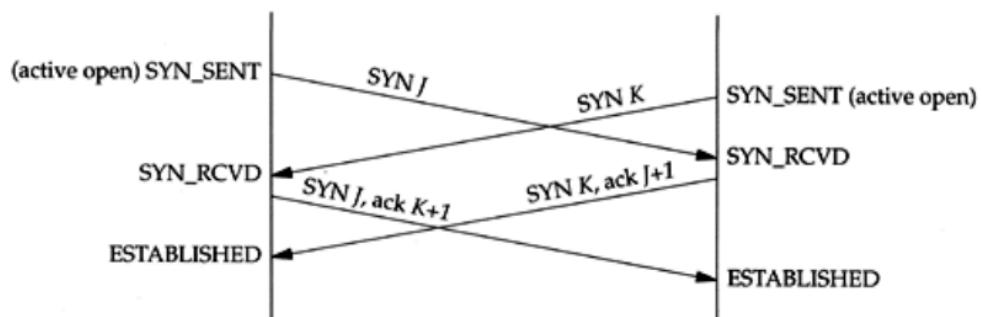
- Puertos: MUX/DEMUX.
- No tiene Longitud total, si de HDR LEN (variable, max 60B Unit=4B).
- Total LEN se computa para PseudoHDR, no viaja en el segmento.
- El tamaño del segmento se calcula dentro del datagrama IP.
- Checksum:
  - Cálculo Ca1. Obligatorio, calculado, igual que UDP.
  - Si tiene error podría pedir retransmisión, implementación de TCP descarta y espera RTO (Retransmisión Timer).
- Necesidad de manejar Timers, RTO (tmout. por cada segmento). (implementaciones lo manejan más eficientemente).
- Campos de Sesiones: Flags: SYN(Synchronize), FIN(Finish), RST (Reset).
- Campo de Detección de Errores: Checksum.
- Campos de Control de Errores: ACK, Num. Sec (#Seq), Num. Ack (#Ack).
- Campo de Control de Flujo: a los de control de errores se agrega, Win.
- Campos de Control de Congestión: se agregan flags si participa la red.

. Funcionamiento:

- En primer lugar y antes de transmitir datos, TCP establece una conexión confiable entre el emisor y el receptor mediante un proceso de tres pasos, conocido como el handshake de tres vías:
  - SYN (Synchronize): El cliente envía un mensaje SYN al servidor para iniciar la conexión.
  - SYN-ACK (Synchronize-Acknowledge): El servidor responde con un mensaje SYN-ACK para confirmar que recibió la solicitud y está listo para establecer la conexión.
  - ACK (Acknowledge): El cliente responde con un mensaje ACK para confirmar que ha recibido el mensaje del servidor. Ahora, la conexión está establecida y los datos pueden empezar a transmitirse.

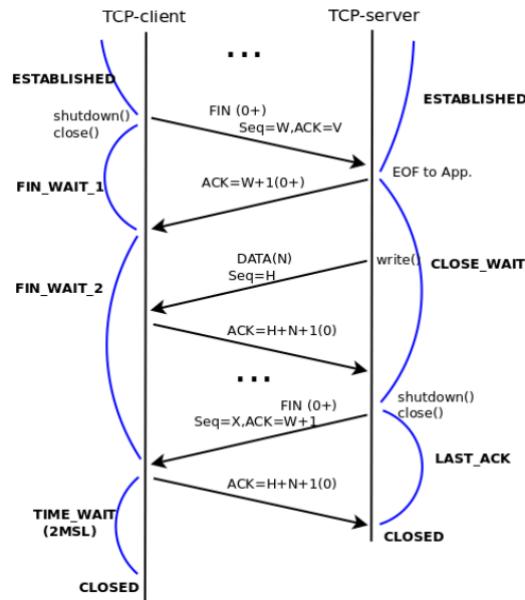


Open simultáneo:



- Luego, una vez que la conexión está establecida, TCP garantiza que los datos se envíen de manera confiable mediante estos mecanismos:
  - Segmentación: Los datos a enviar se dividen en segmentos más pequeños (paquetes). Cada segmento tiene un número de secuencia que permite reconstruir los datos en el orden correcto en el lado del receptor.
  - Confirmación (Acknowledgment): El receptor envía un ACK (acuse de recibo) de cada segmento que recibe correctamente. Si no recibe un ACK en un tiempo determinado, el emisor asume que el segmento se perdió y lo retransmite.

- Control de flujo: TCP utiliza la ventana deslizante (sliding window), donde el receptor puede indicar cuántos datos puede manejar a la vez, ayudando a evitar la congestión en la red o el sobrecargado del receptor.
  - Control de errores: TCP incluye un mecanismo para detectar y corregir errores en la transmisión, utilizando sumas de verificación (checksums) en cada segmento.
  - Posteriormente, se realiza un control de congestión de la red, ajustando dinámicamente la cantidad de datos que se pueden enviar en función de las condiciones de la red.
- Algunos algoritmos son:
- Slow start: Inicialmente, TCP envía una pequeña cantidad de datos y va aumentando la velocidad de transmisión conforme recibe ACKs.
  - Congestion avoidance: Si se detecta pérdida de paquetes, TCP reduce el ritmo de transmisión y ajusta la ventana de envío.
  - Para finalizar, cuando los datos han sido transmitidos, TCP cierra la conexión de manera ordenada usando un proceso llamado *fin de conexión en cuatro pasos*:
    - FIN: El cliente o servidor que desea cerrar la conexión envía un segmento FIN.
    - ACK: El otro extremo responde con un ACK para reconocer la solicitud de cierre.
    - FIN: El receptor luego envía su propio FIN para indicar que también ha terminado de enviar datos.
    - ACK: El primer emisor responde con un último ACK, y la conexión se cierra.



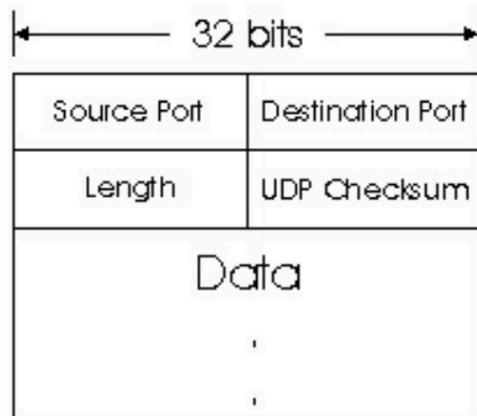
- TCP entrega y envía los datos agrupados o separados de forma dis-asociada de la aplicación:
  - La aplicación puede enviar 3000 bytes en un write y TCP lo podría enviar en 3 segmentos separados de 1000 bytes c/u.
  - La aplicación puede enviar 100 bytes y luego otros 200 y TCP esperar para enviarlos todos juntos en un segmento de 300 bytes.
  - La aplicación puede intentar leer 200 bytes del buffer y TCP solo entregar 150 bytes y luego el resto.
- Requiere mantener “recursos” en cada extremo para los controles:
  - Buffers de Rx y Tx.

- Timer(s) RTO.
- Variables: datos enviados, no confirmados, retransmisiones, umbrales, RTT, etc.
- Estado de la conexión.

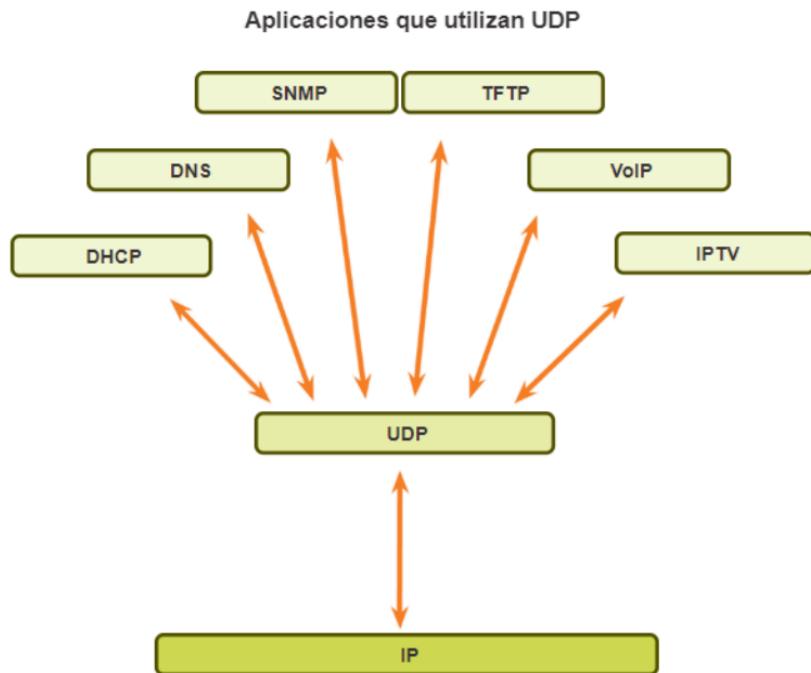
## **PROTOCOLO DE TRANSPORTE UDP**

- . User Datagram Protocol.
- . Es un protocolo de comunicaciones utilizado principalmente en aplicaciones relacionadas con el envío de mensajes SMS (Short Message Service) entre sistemas de telecomunicaciones, como servidores de mensajería y centros de servicios de mensajes cortos.
- . Se conoce como un protocolo **minimalista** ya que está diseñado para cumplir una función específica, que es la transmisión de mensajes de manera rápida y sin muchas características adicionales. Esto resulta en un protocolo con **poco overhead**, es decir, con poca información extra que se añade a los datos principales para gestionar su transmisión, como encabezados, verificaciones de errores, o controles de flujo.
- . El tráfico UDP no está regulado. Una aplicación que emplee el protocolo de transporte UDP puede enviar los datos a la velocidad que le parezca, durante todo el tiempo que quiera.
- . UDP está orientado a paquetes/datagramas (mensajes auto-contenidos). Por lo que su PDU es Datagramas (Por coherencia con nivel Transporte se suele llamar Segmento).
- . Solo provee MUX/DEMUX (multiplexación) y detección de algunos errores.
- . No requiere establecimiento de conexión.
- . El encabezado IP provee: Ruteo, Fragmentación, Detección de algunos errores.
- . El encabezado UDP provee: MUX/DEMUX de aplicación, Detección de errores.
- . IP indica que lleva UDP con el código de protocolo 17.

. *Datagrama UDP:*



- Puertos: MUX/DEMUX.
- Longitud: UDP HDR + Payload.
- Checksum:
  - Cálculo Ca1, Opcional. 0 = Sin checksum.
  - Calculado HDR + PseudoHDR + Payload.
  - PseudoHDR: IP.SRC + IP.DST + Zero + IP.PROTO + UDP.LENGTH.
  - PseudoHDR: protección contra paquetes mal enrutados.
  - Aplicaciones de LAN por eficiencia lo podrían deshabilitar.
  - Si tiene error se descarta silenciosamente.



### **CONTROL DE ERRORES EN TCP**

. El control de errores en TCP es esencial debido a que se ejecuta sobre IP (Internet Protocol), un protocolo no confiable que solo garantiza la entrega de los paquetes según el principio de "best-effort". TCP incorpora mecanismos para garantizar que los datos lleguen de manera correcta, en orden, y sin errores.

Es un mecanismo protocolar, algoritmo, que permite ordenar los segmentos que llegan fuera de orden y recuperarse mediante solicitudes y/o retransmisiones de aquellos segmentos perdidos o con errores. Objetivo: recuperarse de los efectos del reordenamiento, la pérdida o la corrupción de los paquetes en la red.

. TCP utiliza un mecanismo llamado ARQ (Automatic Repeat reQuest). Este mecanismo se basa en el principio de que el receptor debe confirmar la correcta recepción de los datos enviando un mensaje de confirmación (ACK). Si el remitente no recibe esta confirmación en un tiempo determinado, volverá a transmitir el segmento de datos.

Existen diferentes tipos de ARQ que TCP puede usar, como:

- Stop-and-Wait ARQ: El remitente espera la confirmación de cada paquete antes de enviar el siguiente.
- Go-Back-N ARQ: El remitente puede enviar múltiples paquetes sin esperar confirmación, pero si se detecta un error, retransmite todos los paquetes desde el paquete con error en adelante.
- Selective Repeat ARQ: Similar a Go-Back-N, pero solo retransmite los paquetes que no se recibieron correctamente.

. Números de secuencia y confirmaciones: cada byte de datos que se transmite a través de TCP tiene un número de secuencia, que indica su posición en la secuencia de bytes que se está enviando. El receptor utiliza estos números de secuencia para reordenar los datos si llegan fuera de orden y para detectar si falta algún byte.

. Confirmación de datos recibidos: cuando el receptor recibe un segmento de datos, envía un ACK al remitente. Si el remitente no recibe el ACK antes de que expire un temporizador,

asumirá que el segmento se perdió y lo retransmitirá. Este sistema asegura que los datos sean recibidos correctamente o retransmitidos si es necesario.

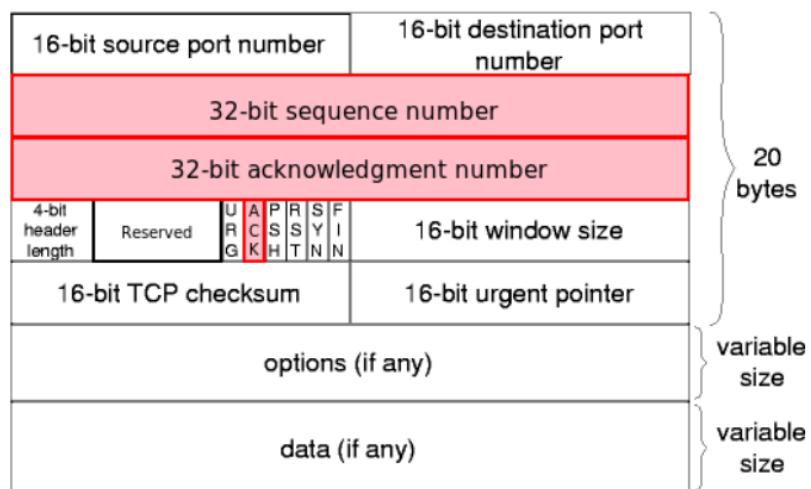
. Timer - RTO (Retransmission Timeout): TCP utiliza un temporizador llamado RTO (Retransmission Timeout) para definir el tiempo máximo que el remitente debe esperar por un ACK antes de retransmitir un segmento. El valor de RTO se calcula dinámicamente basándose en el tiempo de ida y vuelta (RTT - Round Trip Time) de los paquetes entre el remitente y el receptor.

. Buffers de transmisión (TxBuf) y recepción (RxBuf): TCP requiere buffers en ambos extremos de la conexión:

- TxBuf (Transmission Buffer): Almacena los segmentos que han sido transmitidos pero que aún no han sido confirmados (ACK). Si el ACK no llega en el tiempo esperado, los segmentos en este buffer pueden ser retransmitidos.
- RxBuf (Reception Buffer): Almacena los segmentos recibidos que aún no se han procesado o que han llegado fuera de orden. TCP reorganiza estos segmentos en el buffer para asegurarse de que los datos lleguen en el orden correcto a la aplicación.

. Checksum/CRC para detección de errores: TCP utiliza un checksum en cada segmento para verificar si los datos han sido corrompidos durante la transmisión. Este checksum cubre tanto los datos como el encabezado TCP. Si el receptor detecta una discrepancia en el checksum, descarta el segmento y no envía un ACK, lo que eventualmente llevará a la retransmisión del segmento por parte del remitente.

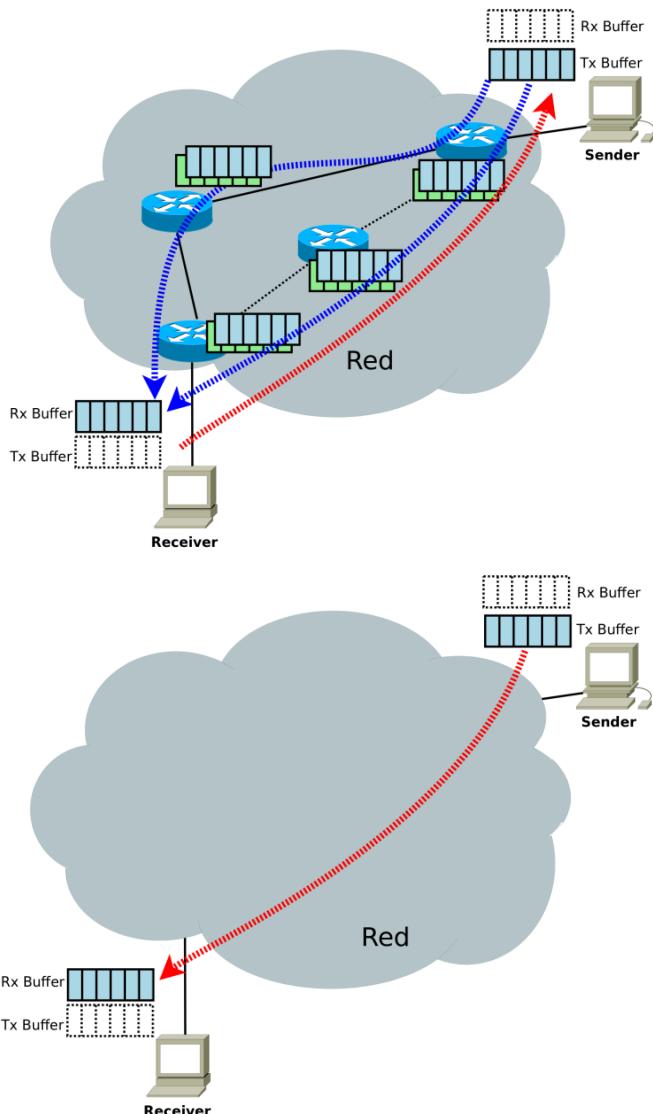
. Segmento TCP en control de errores:



. Capa de Red:

Capa de Red

Capa de Red para TCP



Se pueden generar errores en los extremos o en la red, ya que pueden perderse datos y/o ACKs por descartes o errores en la red, y también pueden duplicarse datos y/o ACK, por retransmisiones o por errores en equipos extremos o intermedios, red.

Además, pueden desordenarse por usar caminos múltiples o errores de procesamiento en equipos. O bien, los datos y/o ACKs podrían corromperse.

Existe delay (retardos) mayores a 1 RTT, llevando a que el RTT podría ser variable.

. El RTT (Round-Trip Time) es el tiempo que tarda un paquete en viajar desde un origen hasta un destino y de vuelta al origen. Se mide desde el momento en que se envía un paquete de datos hasta que el remitente recibe una respuesta. RTT incluye tanto el tiempo de transmisión (de ida y vuelta) como los tiempos de procesamiento en los dispositivos intermedios, como routers o servidores.

#### *. Control de Flujo:*

El control de flujo es activado por el receptor enviando ventanas más chicas. Esto deja en evidencia que el receptor tiene poco espacio (o no tiene más lugar) para seguir recibiendo datos. Esto se realiza a través del campo de tamaño de ventana en los encabezados de los segmentos TCP.

Resuelve el problema de la posible saturación o congestión de los buffers en los endpoints. Al indicar al emisor que reduzca la cantidad de datos que está enviando, evita que el receptor se sobrecargue.

Cuanto tiempo dura activo depende del receptor (más que nada la velocidad en que lee la aplicación). El control de flujo está activo mientras el receptor envíe ventanas más pequeñas (indicando capacidad limitada). Durará activo hasta que el receptor envíe ventanas más grandes, lo que indica que tiene más capacidad para recibir datos.

En todo momento ambos extremos están actualizando su propia ventana.

#### *.Control de Congestión:*

El control de congestión lo activa el emisor. El emisor limita la velocidad de transmisión de tráfico a través de su conexión en función de la congestión de red percibida. Este proceso es dinámico y adaptativo, y el emisor ajusta su velocidad de transmisión en respuesta a las condiciones cambiantes de la red.

Los posibles disparadores son:

- Fin de Temporización: La expiración del temporizador asociado con el envío de un segmento TCP puede ser interpretada como una señal de pérdida, indicando posiblemente congestión en la ruta.
- Recepción de TRES ACK Duplicados: La recepción de paquetes ACK duplicados procedentes del receptor también se interpreta como un suceso de pérdida. Este evento puede sugerir la pérdida de un paquete en la red debido a la congestión.

El objetivo es que no se desborde la propia red. Esto ocurre cuando hay más tráfico de red del que la red puede manejar eficientemente, lo que puede resultar en la pérdida de paquetes, retrasos elevados y un rendimiento de red deficiente. El control de congestión busca evitar que la red se sobrecargue ajustando la tasa de transmisión de datos del emisor para que sea compatible con la capacidad de la red.

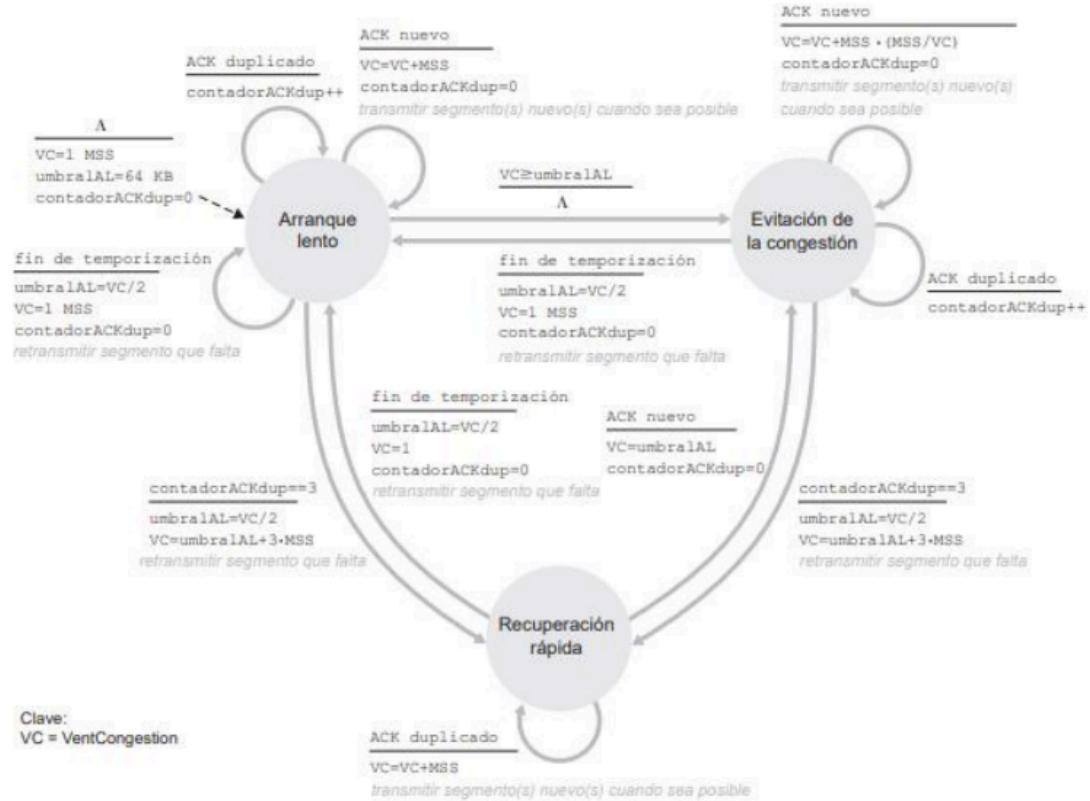
Arranque Lento (Slow Start):

- Inicio de la Conexión: Se utiliza al inicio de una conexión TCP.
- Tamaño de la Ventana de Congestión (VentCongestion): Inicializado con un valor pequeño (1 MSS, tamaño máximo de segmento).
- Crecimiento Exponencial: La ventana de congestión se duplica en cada periodo RTT.
- Finalización del Crecimiento Exponencial:
  - o Al detectarse un suceso de pérdida (fin de temporización).
  - o Cuando el valor de VentCongestion alcanza o sobrepasa el umbral de arranque lento (umbralAL).

Evitación de la Congestión (Congestion Avoidance):

- Transición desde Slow Start: Inicia cuando se detecta congestión y se sale del arranque lento.
- Tamaño de Ventana de Congestión (VentCongestion): Aproximadamente la mitad del valor cuando se detectó congestión por última vez.
- Crecimiento Lineal: Se incrementa en un MSS por RTT, más conservador que el crecimiento exponencial.
- Finalización del Crecimiento Lineal:
  - o Al detectarse un suceso de pérdida (fin de temporización o tres ACK duplicados).
  - o El valor de VentCongestion se fija en 1 MSS y se actualiza el umbral de arranque lento (umbralAL).

o En el caso de pérdida detectada por tres ACK duplicados, se realiza un ajuste menos drástico del valor de VentCongestion y umbralAL, entrando en el estado de recuperación rápida.



#### . ARQ - Stop & Wait:

El emisor manda un segmento numerado y espera confirmación. Al enviar el segmento arranca un RTO, y si no recibe ACK (confirmación) lo reenvía.

El receptor, cada vez que recibe un segmento, confirma indicando (Num. por segmentos o bytes. Se puede confirmar el actual o el que se espera. Si tiene errores, lo descarta o podría confirmar indicando que Num.(#) espera. Funcionan como NAK (No Acknowledge). ACK recupera los problemas en la red.

Luego, si el emisor recibe confirmación envía un nuevo segmento, si tiene en el TxBuf, e inicia un nuevo RTO (en caso que la capa superior haya dejado datos para enviar en el buffer). Si tenía confirmaciones fuera de secuencia (out-of-order), e.g. atrasadas, las descarta. Si recibe confirmación del anterior (similar NAK) podría re-enviar, en caso de ACKs > N, N >= 2.

Para analizar el rendimiento del S&W podemos ver que si RTT aumenta y BW aumenta se hace peor.

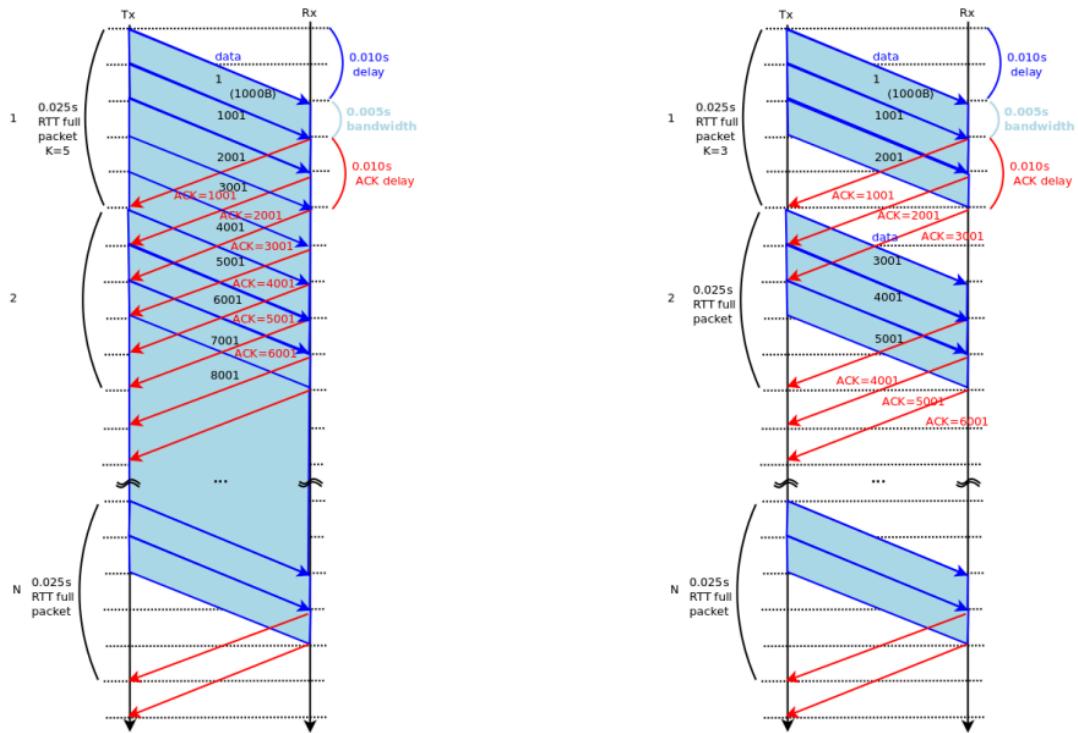
El sistema es ineficiente, ya que envía un dato por vez. No se envía el próximo mensaje hasta que no se confirma el que se envió. Además, cada vez que envía un segmento requiere arrancar un timer: RTO.

Ver gráficos S&W, diapositivas 9 a 16, clase 5 - Control de Errores TCP.

#### . Pipelining/Sliding Window:

Es un protocolo de control de flujo y manejo de errores que solo retransmite los paquetes que no se confirmaron.

- . Permite enviar múltiples segmentos o paquetes por RTT (rafaga) sin aun haber recibido confirmaciones, paquetes "in-flight".
- . El emisor debe saber la cantidad de segmentos o bytes que se pueden enviar sin aún recibir confirmación. A esta cantidad se la llama 'Ventana', notado como K o W , W = n, donde n > 1.
- . Requiere buffering de Tx, TxBuf (lo que deja la capa superior y aún no se confirmó) y de Rx, RxBuf (lo que se va recibiendo hasta entregar a la capa superior).
- . Por cada mensaje enviado se podría iniciar un timer de retransmisión, RTO (no escala), y además, mantener un RTO por ráfaga, para el segmento más viejo no confirmado.
- . El receptor debe confirmar los segmentos recibidos (Se confirma indicando lo que se espera). Estas confirmaciones deslizan la ventana sobre TxBuf habilitando nuevos segmentos a transmitir.
- . Podrían generarse Confirmaciones Negativas: NAK (NO-Acknowledge), implícitas o explícitas. (TCP usa implícitas con varios ACK dups).



- . ARQ - Go-Back-N:
- . Go-Back-N es un protocolo de control de flujo y manejo de errores que permite el envío de múltiples paquetes sin necesidad de esperar a recibir una confirmación por parte del receptor, donde cada paquete tiene un número de secuencia único que lo identifica, y donde el receptor envía un ACK solo para el último paquete recibido en orden. Si un paquete llega fuera de orden, el receptor simplemente lo descarta y no envía un ACK hasta que reciba el paquete que esperaba.
- Esto genera que el emisor, si no recibe un ACK para un paquete en un tiempo determinado (debido a pérdida o error), debe retransmitir **todos los paquetes a partir de ese paquete no reconocido**, incluso si algunos de los paquetes más adelante se habían recibido correctamente por el receptor. Este es el concepto de "Go-Back-N", ya que el emisor vuelve y retransmite desde el paquete N en adelante.

- . Entonces, no se admiten segmentos fuera de orden, ni confirmaciones fuera de orden. Solo se confirman por la positiva los segmentos que se pudieron colocar en el buffer en orden.
- Además, solo se puede confirmar desde N hacia atrás (ACK acumulativos). No necesariamente se confirma cada segmento individualmente.
- . El emisor transmite tantos segmentos o bytes que admite W (tamaño de la ventana) de acuerdo a lo que tiene en el buffer dejados por la capa superior. Y se inicia RTO al enviar segmento.
- . Utiliza piggy-backing, que es una técnica que busca aprovechar los mensajes o paquetes de datos que ya se están enviando para agregar (o "montar") otra información útil, como un ACK (acknowledgment), en lugar de enviar esos datos por separado. Esto ayuda a reducir la cantidad de mensajes que deben ser enviados, mejorando la eficiencia y minimizando el uso del ancho de banda.

#### **Emisor (Tx):**

- Si tiene datos en el buffer TxBuf y la ventana lo permite,  $W >$  in flight segments, los transmite.
- Si no tiene RTO activo arranca un nuevo RTO para el primero que manda.
- Si recibe un ACK en orden, desliza/actualiza la ventana, si hay segmentos "in-flight" re-arranca nuevo RTO, sino lo descarta.
- Si vence RTO, re-envía todo a partir del segmento más viejo aún no confirmado.

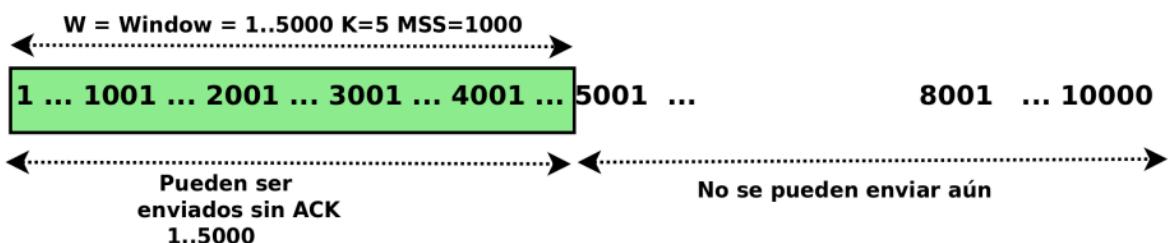
#### **Receptor (Rx):**

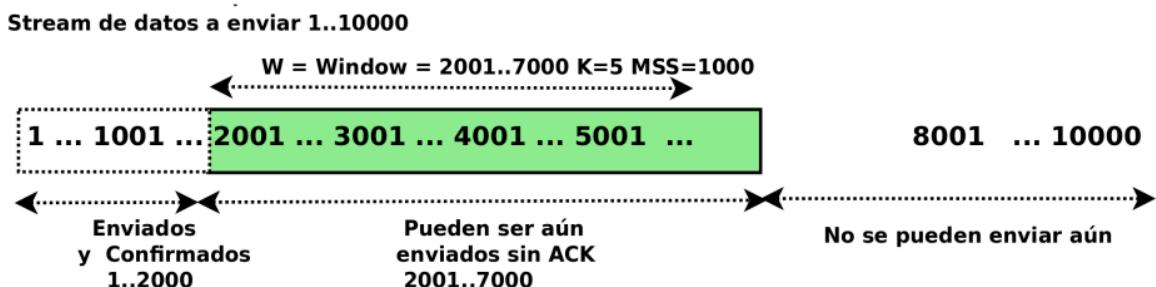
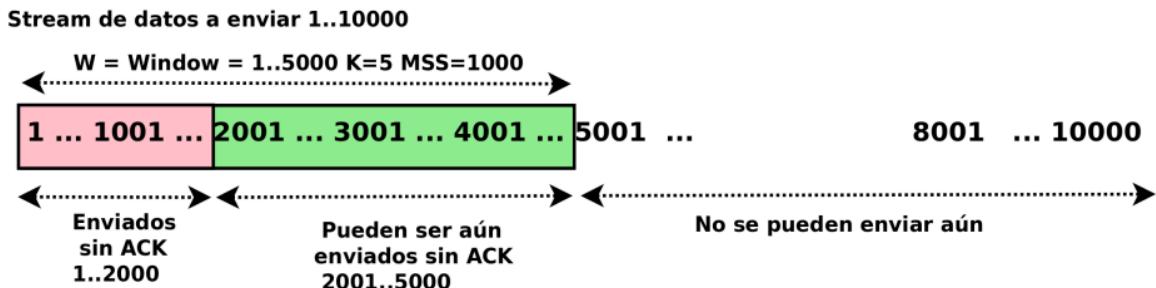
- Si recibe segmento en orden confirma por la positiva indicando el próximo que espera.
- Si el segmento recibido está corrupto, lo descarta como si se hubiese perdido, espera retransmisión.
- Si recibe un segmento fuera de orden confirma indicando que espera uno anterior (NAK):

- Puede descartar el segmento fuera de orden, ya que será retransmitido.
- Puede bufferear el fuera de orden, pero no entregar a capa superior, esperando que llegue el/los segmento/s que llena/n el hueco y luego confirmarlo.

- . Evolución de la ventana Inicial:

#### **Stream de datos a enviar 1..10000**





. Segmentos fuera de orden:

Las confirmaciones por la negativa pueden generar ACK duplicados (NAK).

Mensajes fuera de orden:

Buffering hasta recibir los que llenan los huecos. (requiere buffering de Rx antes de pasarlos a la capa usuaria).

Descartarlos y esperar retransmisiones. (no requiere buffering del receptor, solo recordar la secuencia que se espera).

Los NAK por sí solos no generan retransmisiones, requiere varios duplicados para retransmitir. Un NAK solo no necesariamente indica pérdida, puede ser re-ordering en la red.

. ARQ - Selective Repeat:

. Selective Repeat es un protocolo de control de flujo y manejo de errores que permite el envío de múltiples paquetes sin necesidad de esperar a recibir una confirmación por parte del receptor, donde cada paquete tiene un número de secuencia único que lo identifica, usados por el receptor para saber si los paquetes han llegado y en qué orden.

A diferencia de Go-Back-N, el receptor envía un ACK independiente para cada paquete recibido, incluso si los paquetes llegan fuera de orden. Esto significa que puede aceptar y almacenar temporalmente paquetes fuera de orden (en un búfer), mientras espera los paquetes que faltan.

El emisor sólo retransmite los paquetes específicos que no fueron reconocidos (aquellos que no recibieron un ACK), en lugar de retransmitir todos los paquetes desde el último no reconocido. Esto lo hace más eficiente que Go-Back-N.

. No se deben confundir los segmentos de diferentes ráfagas. Por lo que no se deben reusar #ID/SEQ hasta asegurarse que tiene todos los mensajes previos o estos no están en la red. La ventana se desliza sin dejar huecos, desde los confirmados más viejos.

**Emisor (Tx):**

- Si tiene datos en el buffer y la ventana lo permite los transmite igual que GBN.

- A diferencia que GBN cada segmento requiere su RTO, se puede simular con un solo timer.
- Si recibe un ACK en orden, desliza/actualiza la ventana, detiene el RTO para el segmento confirmado.
- Si recibe un ACK fuera de orden, no desliza/actualiza la ventana, detiene el RTO para el segmento confirmado y lo marca como ya recibido.
- Si vence RTO, re-envía el segmento asociado.

#### **Receptor (Rx):**

- Si recibe un segmento dentro de la ventana esperada confirma el segmento particular (grupo de bytes).
- Si recibe segmento dentro de la ventana esperada y no lo tiene lo almacena, si ya lo tiene lo descarta. Siempre confirma. Requiere buffear los segmentos RxBuf.
- El receptor actualiza su ventana conforme recibe los segmentos en orden.

#### *. Cálculo del RTO:*

Para calcular RTO se estima RTT (Round Trip Time). RTT inicial RFC-2988(2000), 3seg - RFC-6298(2011), 1 seg. Cambio en las redes.

$$RTO = SRTT + (4 * DevRTT) \text{ (RFC-6298).}$$

$$SRTTi = (1 - \alpha) * SRTTi-1 + \alpha * RTT, \alpha = \frac{1}{4}$$

La influencia de las muestras pasadas decrece exponencialmente.

$$DevRTTi = (1 - \beta) * DevRTTi-1 + \beta * |RTT - SRTTi|, \beta = \frac{1}{4}$$

Si hay gran variación en SRTTi se usa un mayor margen.

RTO < 1seg : redondeado a 1 seg (RFC-6298).

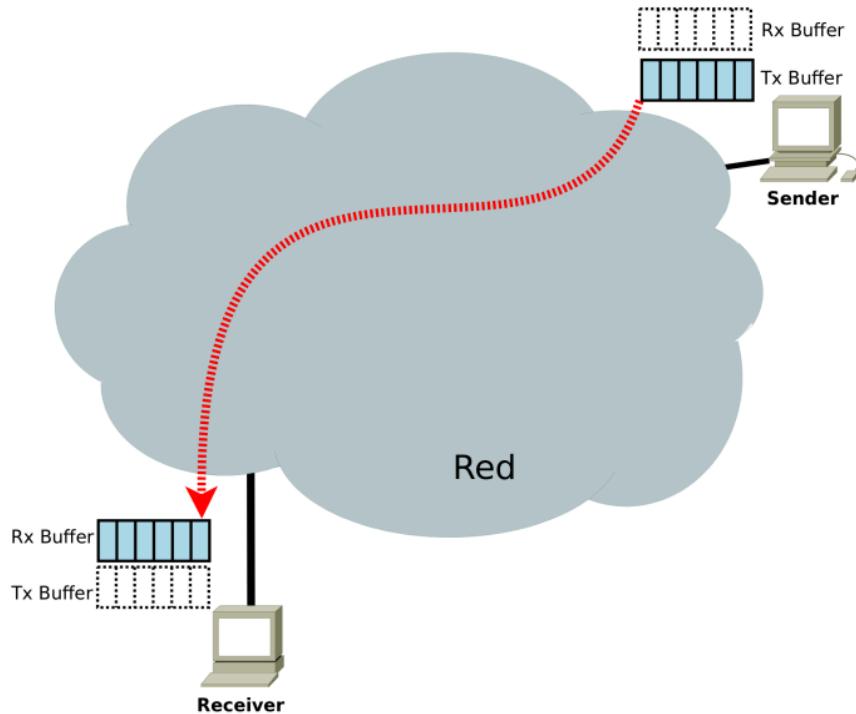
Se mide por cada RTT. Se puede utilizar la opción TimeStamp.

## **CONTROL DE FLUJO**

. Mecanismo protocolar, algoritmo, que permite al receptor controlar la tasa a la que le envía datos el transmisor. Controla cuánto puede enviar una aplicación sabiendo que la receptora tiene capacidad de recibirlo y procesarlo. Objetivo: prevenir que el emisor sobrecargue al receptor con datos evitando un mal uso de la red.

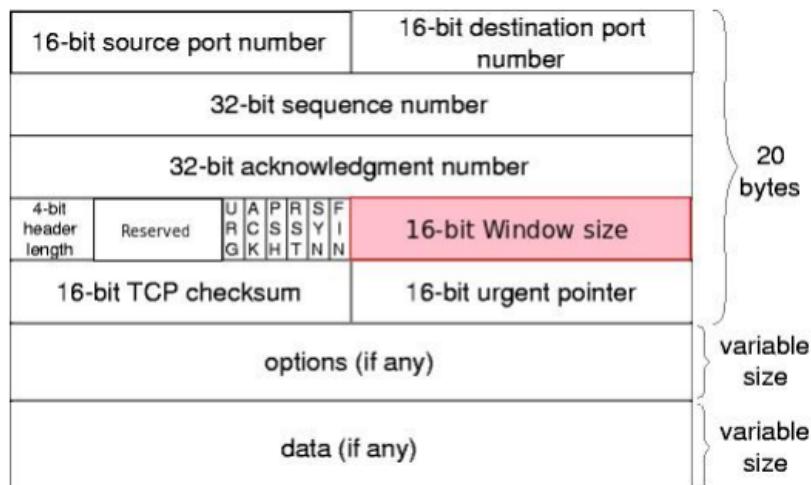
. Utiliza técnicas de ARQ para transferencia de datos fiables. Pero ARQ solo no hace control de flujo, requiere de otros mecanismos como RNR (Receive-Not-Ready), o Dynamic Window (Ventana Dinámica). TCP usa Ventana Dinámica.

## De Extremo a Extremo, principio end-to-end.



### . Funcionamiento:

- El receptor (cada extremo puede recibir, es FDX) indica el espacio del buffer de recepción, Rx Buffer, en el campo del segmento: Window (de datos o ACK) Advertised Window (Ventana Anunciada).



- Por cada segmento que envía indica el tamaño del buffer de recepción Rx Buffer (mbufs). (Cada conexión mantiene su propio buffer) en espacio del kernel (TCP).
- Window (Ventana) indica la cantidad de datos en bytes que el emisor le puede enviar sin esperar confirmación (mejora notablemente contra Stop & Wait). La ventana de recepción de cada extremo es independiente.
- Cada vez que llega un segmento nuevo en orden es puesto por TCP en el Rx Buffer, TCP lo debe confirmar.
- Cada vez que la aplicación lee se hace espacio en el Rx Buffer. Se va modificando el tamaño de la ventana. Se comunica con los segmentos de ACK (y de datos).

- Cada vez que llega un ACK en orden se mueve la ventana en el Transmisor, se descartan segmentos confirmados del Tx Buffer.

*. Ventana Deslizante:*

Gestiona el envío y la recepción de segmentos de datos, asegurando que el receptor no se vea abrumado por demasiada información y que el transmisor no envíe datos más rápido de lo que el receptor puede procesar.

**Ventana de envío (send window):**

- Es el rango de bytes que el emisor puede enviar antes de necesitar una confirmación (ACK) del receptor.
- Determina la cantidad máxima de datos que pueden estar "en tránsito" sin haber sido confirmados aún.

**Ventana de recepción (receive window):**

- Es el tamaño de buffer que el receptor tiene disponible para recibir datos. Le indica al emisor cuántos bytes puede seguir enviando sin desbordar la capacidad del receptor.

**Control de flujo (flow control):**

- TCP ajusta dinámicamente el tamaño de la ventana según las capacidades del receptor para prevenir la congestión.
- Si el receptor tiene un buffer pequeño o está recibiendo los datos lentamente, la ventana de recepción será pequeña, limitando la cantidad de datos que el emisor puede enviar.

**Confirmaciones (ACKs):**

- Cuando el receptor recibe datos, envía una confirmación al emisor. Este puede continuar enviando más datos siempre que estén dentro del tamaño permitido por la ventana.

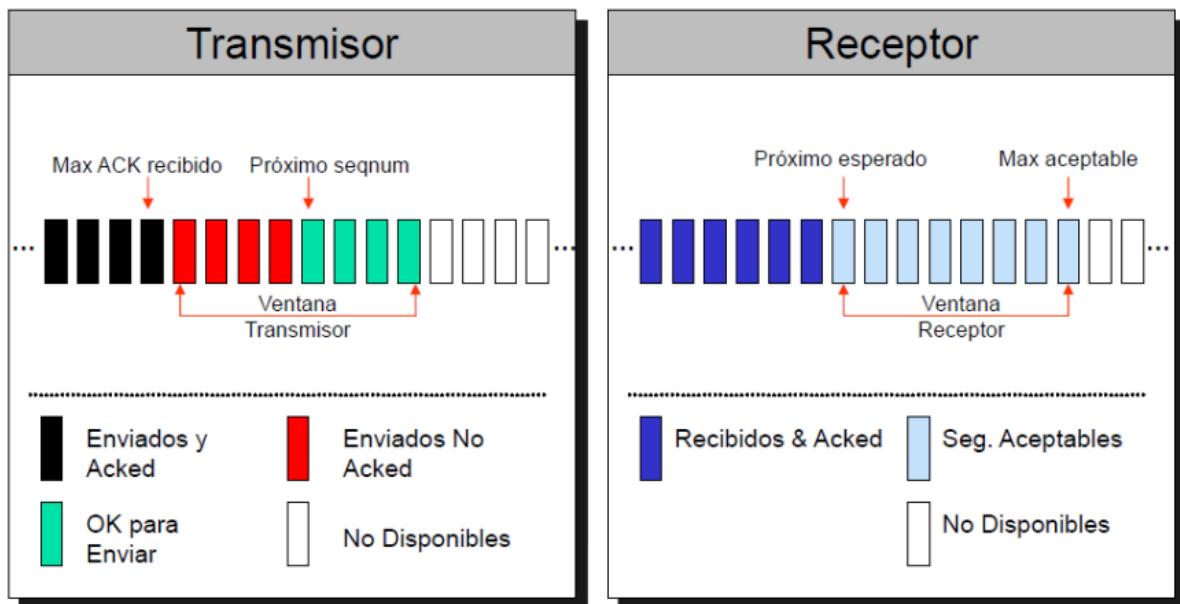
**Deslizamiento de la ventana:**

- La ventana "se desliza" conforme se reciben las confirmaciones de los paquetes. Es decir, a medida que el emisor recibe ACKs por datos ya recibidos, puede enviar nuevos datos que "avanzan" la ventana hacia adelante.

**Ejemplo de funcionamiento:**

Supongamos que la ventana de envío tiene capacidad para 4 segmentos:

- El emisor envía los primeros 4 segmentos y espera las confirmaciones.
- Una vez que el receptor confirma el primer segmento (ACK), la ventana se desliza, permitiendo al emisor enviar un nuevo segmento en lugar del que fue confirmado.
- Este proceso continúa, con la ventana deslizándose a medida que se confirman los segmentos.



### Segmento enviado

Source Port	Dest. Port
Sequence Number	
Acknowledgment	
HL/Flags	Window
D. Checksum	Urgent Pointer
Options..	

### Segmento recibido

Source Port	Dest. Port
Sequence Number	
Acknowledgment	
HL/Flags	Window
D. Checksum	Urgent Pointer
Options..	

acknowledged      enviados      a enviar

El receptor “ofrece/publica” la ventana Win en los segmentos TCP.

El transmisor no puede enviar más de la cantidad de bytes en:

Win – Sent.No.ACKed

Effective\_Win = Win – (LastByteSent – LastByteAcked) si no se tiene en cuenta la congestión.

- Al recibir ACKs de TCP (Aplic. no lee aún) se cierra la ventana.
- Al recibir ACKs y Win fijo desliza ventana (Aplic. lee a tasa(rate) fija).
- Al achicarse Win se reduce la ventana (Aplic. no lee).
- Al agrandarse Win tiene posibilidad de enviar más (Aplic. lee más rápido).
- Tamaño de ventana seleccionado por el kernel o por aplicación setsockopt().

. *Enfoques TCP:*

**TCP Bulk:**

- Este enfoque implica la transmisión de grandes cantidades de datos de una vez, en un solo flujo continuo.

**TCP Iterativo:**

- En este enfoque, los datos se envían en bloques más pequeños o porciones iterativas, y cada bloque enviado requiere una confirmación o respuesta antes de enviar el siguiente.

En estos enfoques se utilizan las siguientes técnicas:

- Delayed ACKs: reduce la cantidad de ACKs enviados al esperar un poco antes de confirmar la recepción de un paquete, con la esperanza de que el receptor también tenga datos que enviar y pueda "adjuntar" el ACK junto con esos datos (*piggybacking*).
- Algoritmo Nagle: evitar que TCP envíe múltiples pequeños paquetes (conocidos como **tinygrams**) cuando no hay confirmaciones pendientes. El algoritmo intenta combinar varios pequeños fragmentos de datos en un solo paquete antes de enviarlos, lo que optimiza el uso de la red al evitar la sobrecarga de tener muchos encabezados TCP/IP pequeños.
- Silly Window: El **Silly Window Syndrome** (SWS) ocurre cuando la ventana de recepción TCP (es decir, el buffer del receptor) está casi llena, y el receptor solo puede aceptar pequeñas cantidades de datos adicionales. Si el receptor sigue anunciando pequeños incrementos en la ventana de recepción, el emisor puede enviar tinygrams, lo que reduce la eficiencia.

**TCP Escalado de Ventana:**

- Esta técnica permite al protocolo TCP utilizar ventanas de recepción más grandes, lo cual es crucial para aprovechar al máximo el ancho de banda disponible en estas redes.

## **CONTROL DE CONGESTIÓN**

. Se realiza por cada conexión (End-to-End, App-to-App), permitiendo que las aplicaciones no saturen la capacidad de la red.

Tiene en cuenta el estado de la red, a diferencia del control de flujo que solo ve el receptor.

. Su objetivo es controlar el tráfico que se envía evitando que se colapse la red y se descarten los envíos teniendo estos que retransmitirse.

. La congestión son los problemas de delay en los routers, y problemas de overflow y descarte.

. Usa control extremo a extremo, sin asistencia de la red.

. *CongWin* es dinámica y en función de la congestión percibida de la red.

. *RcvWindow* es el número de bytes que el Rx puede recibir en su buffer, aquí lo suponemos grande y no limita la tasa de envío.

. *Causas de congestión de la red:*

- Límite de la capacidad de la red:

    Velocidad de los Routers/Switches (CPU).

    Capacidad de los Buffers de los Routers/Switches (Memoria).

    Velocidad de los enlaces (Interfaces).

- Utilización de la red:

    Demasiado tráfico en la red (modelo de red compartida).

Se detecta por los nodos intermedios (router/switches) por ejemplo: cuando las colas sobrepasan un umbral. Se utiliza simple umbral o doble umbral (min,max).

#### . Modelo End-to-End:

- Modelo en el cual no participa la red (más implementado).
- Se utilizan nuevos parámetros a los de Control de Flujo (variables locales):
  - cwnd* Ventana de congestión. Tiene en cuenta el estado de la red.
  - ssthresh* Slow Start Threshold (Umbral).
- Se calcula:  $\text{MaxWin} = \text{Min}(\text{rwnd}, \text{cwnd})$ . rwnd era la ventana de recepción, usada para el control de flujo.
- $\text{FlightSize} = (\text{LastByteSent} - \text{LastByteACKed})$  segmentos en vuelo, enviados y aún no confirmados (Sent.No.ACKed).
- Puede enviar:
  - $\text{Effective\_Win} = \text{MaxWin} - \text{FlightSize}$ .
  - $\text{MaxWin} = \text{Min}(\text{rwnd}, \text{cwnd})$  se calcula en base a quien está más cargado, el receptor o la red.
- Diferentes Fases:
  - Si  $\text{cwnd} < \text{ssthresh}$ : Fase de crecimiento inicial: SS (Slow Start).
  - Si  $\text{cwnd} \geq \text{ssthresh}$  Fase de mantenimiento: CA (Congestion Avoidance).

#### . Slow Start:

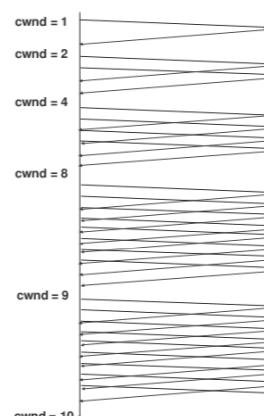
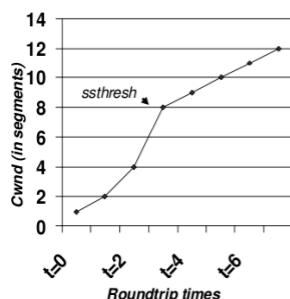
- Crece exponencialmente, de forma rápida, no es slow (lento).
- Se le llama Slow Start porque comienza a probar con pocos paquetes, menos agresivo que el enfoque de enviar tanto como la ventana de recepción permita.
- Inicia  $\text{cwnd} = \text{IW} = 1 * \text{MSS}$  (a veces se usa 2 o 3). Transmite y espera ACK.
- ACK recibido,  $\text{cwnd} + +$ :  $\text{cwnd} = 2 * \text{MSS}$ , nuevos ACKs:  $\text{cwnd} = 4 * \text{MSS}$  ...
- Si el destino retarda ACK no se cumple.
- Incrementa  $\text{cwnd} + +$  (en MSS) por cada ACK (varios por RTT, rafaga).

#### . Congestion Avoidance:

- Ante el primer evento de congestión se calcula el *ssthresh*, primer rafaga SS puro.
- Una vez que  $\text{cwnd} \geq \text{ssthresh}$  crece de forma lineal.
- Incrementa  $\text{cwnd} + +$  por cada RTT.

### Fases Control de Congestión TCP (SS y CA)

Assume that  $\text{ssthresh} = 8$



#### . Enfoque Reno:

- Actualmente se utilizan variantes de este enfoque.

- Se implementan de forma correcta fast Retransmit y Fast Recovery:
  - Slow Start (SS).
  - Congestion Avoidance (CA).
  - Fast Retransmit (FRT).
  - Fast Recovery (FR).
- $FlightSize = cwnd$  si siempre tuvo datos para enviar, en general  $cwnd > FlightSize$ ,  $cwnd$  crece sin enviar realmente datos, no reflejaría el límite de la red.

- *Fast Retransmit:*

Intenta recuperarse más rápido que un timeout (expira RTO).

El receptor inmediatamente al recibir un segmento fuera de orden no debe esperar RTO del emisor sino generar un ACK (DUP ACK). No se debe retardar. Debido a que el emisor no sabe si se perdió o llegó fuera de secuencia: espera por más ACK duplicados. El emisor, si recibe 3 ACK duplicados (4 ACK para el mismo segmento) considera que se perdió (no es re-ordenamiento) y re-envía el segmento solicitado sin esperar RTO.

- *Fast Recovery:*

Si recibe 3 ACK duplicados (4 ACK para el mismo segmento) entró en Fast Retransmit y se reenvió el segmento. Luego de Fast Retransmit entra en fase Fast Recovery, crece linealmente.

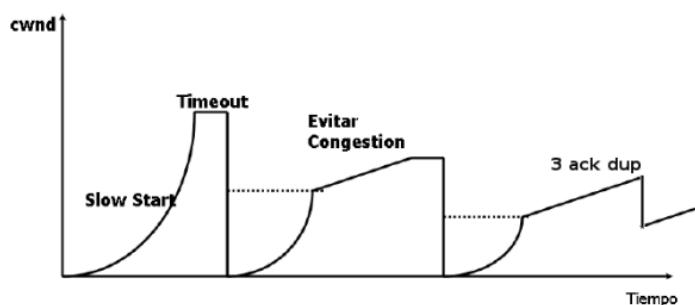
Incrementa de forma lineal la ventana por cada ACK recibido, considera que es un espacio nuevo en la red (incremento inicialmente en 3, por los 3 ACK duplicados). Luego de Fast Recovery (se ACKed el segmento perdido) se realiza CA(Reno), no SS(Tahoe).

- *RTO:*

Si da timeout, el RTO expira.

1. Retransmite el segmento perdido.
2.  $ssthresh = \min(cwnd/2, 2) * MSS$ , en RFC  $ssthresh = \min(FlightSize/2, 2 * SMSS)$
3.  $cwnd = LW = 1 * MSS$ , en RFC  $LW = 1 * SMSS$ .
4. Inicia con Slow Start como en los algoritmos anteriores.
5. Comienza a retransmitir como Go-Back-N a partir del segmento que dio timeout de acuerdo a lo que le permite la ventana.

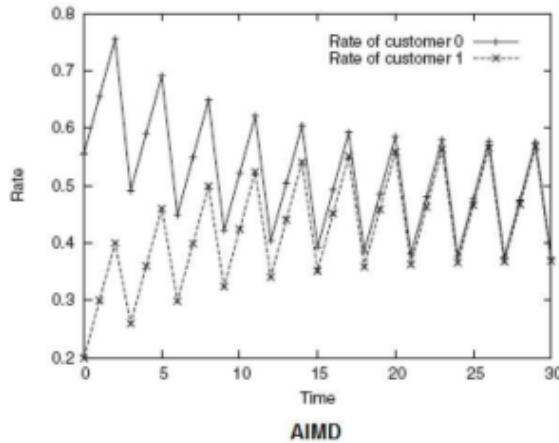
**Ejemplo: Evolución Ventana de Congestión**



- *AIMD (Additive Increase/Multiplicative Decrease):*

Ajusta dinámicamente la ventana de congestión (congestion window o **cwnd**), que define cuántos segmentos puede enviar el emisor antes de recibir un acuse de recibo (ACK).

Crece de forma aditiva (aumenta *CongWin* en 1 MSS cada RTT en ausencia de pérdida) con ACK positivos y decrece de forma multiplicativa ante 3 ACK DUP (reduce *CongWin* a la mitad luego de perdida). Se tratan de autorregular entre flujos.



#### *. Enfoque New Reno:*

- Es una variante utilizada hoy en día.
- TCP Reno ante la pérdida de múltiples segmentos probablemente termine en timeout (RTO expira) y vuelve a Slow Start.
- Por el contrario, TCP New Reno, una vez que está en Fast Recovery, permite enviar varios segmentos perdidos y recuperarse de ACK parciales. New Reno obtiene buen rendimiento con pérdida en varios segmentos y sin SACK.

## Clase 6 | Capa de Red

### CAPA DE RED

. La capa de red proporciona los servicios de enrutamiento y reenvío de paquetes (su PDU) entre distintos hosts. El dispositivo principal de esta capa es el router.

Utiliza el protocolo IP, al cual se lo considera un protocolo de mejor esfuerzo ya que se trata de un protocolo poco confiable, esto quiere decir que hace todo lo posible para entregar los datos, pero no garantiza que todos los paquetes llegarán al destino, ni de que lo harán en el orden correcto.

. Los prefijos de longitud fija por clase provocan un uso inefficiente en el espacio de direcciones y muchos equipos, produce escasez de direcciones.

Esto supone la aparición de las subredes, que básicamente permite que haya subgrupos en las redes, se utiliza para generar redes dentro de la red. Para ello toma una parte del hostid. La división en subredes plantea que si una red de clase desperdicia muchas direcciones IP entonces la misma sea dividida en N subredes más pequeñas que aprovechen mejor el espacio de direccionamiento.

Las máscaras se utilizan para saber en una dirección IP qué bits son de red y qué bits son de host, siendo 1 los bits de **red** y 0 los de **host**: **1111111.1111111.1111100.00000000**.

### INTERNET

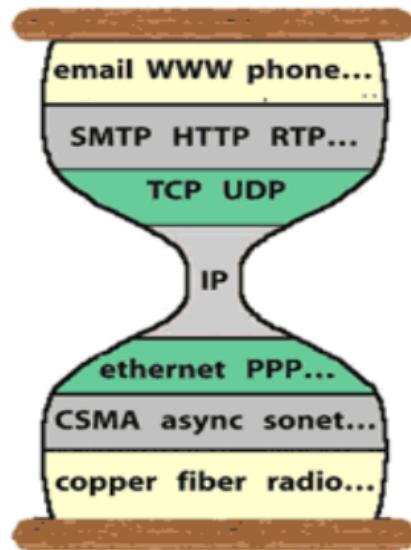
#### *. Estructura de Internet:*

Internet es un conjunto de redes interconectadas y agregadas de forma “jerárquica”.

Puede haber una *Net Neutrality* (Neutralidad de la red), que es el tratamiento de todo el tráfico de forma equivalente por redes de ISP y carriers.

. *Modelo de Internet:*

Internet posee un modelo de “reloj de arena”, ya que todo debe pasar por el protocolo IP:



## **PROTOCOLO IP**

- . El protocolo IP es un protocolo que trabaja End-to-End (Extremo-a-extremo), donde el ruteo se produce hop-by-hop (salto-a-salto), y donde cada nodo debe implementar IP.
- . Los protocolos IP actuales brindan servicios a la capa de transporte y usa servicios de la capa de enlace.

. *Características de IPv4:*

Es un Protocolo de Red no orientado a conexión. Es llamado “Protocolo de Mejor Esfuerzo” (best-effort) que no es confiable, ya que no asegura el arribo de los mensajes. Su PDU es el datagrama o paquete.

Funcionalidad: Direccionamiento. Ruteo/Forwarding/Switching L3. Mux/Demux de protocolos superiores. Accesorias (Solucionar deficiencias del protocolo). Fragmentación. Otras: como evitar loops (TTL), detección de errores.

Es un Protocolo de Red no orientado a conexión a diferencia de TCP u otros protocolos considerados de red X.25, ATM.

. *Direccionamiento IP:*

Una Dirección IP identifica únicamente un punto de acceso (interfaz) a la red.

Por su lado, un router o un host multi-homed tienen varias IPs. Cada interfaz tiene un valor único.

Tienen un significado global en la Internet o privado (local).

Globales: asignadas por autoridad central.

*Direccionamiento Fijo:*

4 Redes físicas, requieren 4 redes IP:

Si cada red tiene menos de 254 hosts, por ejemplo 25 c/red. se pueden utilizar 4 clases C:

Red A: 193.168.1.0\*

Red B: 193.168.2.0

Red C: 193.168.3.0

Red D: 193.168.4.0

#### Problemas con Direcccionamiento Fijo:

Prefijos de longitud fija por clase, provoca un uso ineficiente en el espacio de direcciones.

Muchos equipos, produce escasez de direcciones.

El crecimiento acelerado de la Internet, evidencia la falta de escalabilidad del esquema.

Crecimiento de tablas de ruteo en el núcleo de la red.

Codificar la red en la dirección IP implica que si un host cambia de red, cambiará su dirección (IP Mobility). Problema atacado en IPv4, mejor resuelto en IPv6.

Soluciones IPv4: subnetting, CIDR, NAT, DHCP.

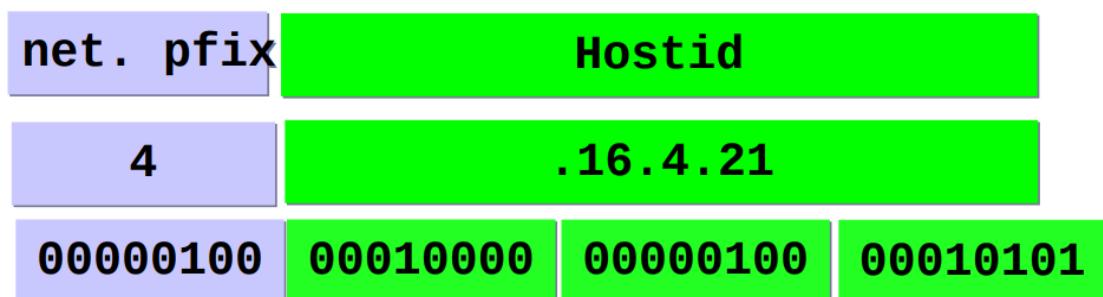
Definitivamente solucionados en IPv6.

#### . Dirección IP:

Son números de 32 bits, expresados en notación decimal delimitada por puntos byte a byte (e.g. 163.10.45.77). Para facilidad de los usuarios, mapping con nombres de dominio (DNS - Domain Name Server).

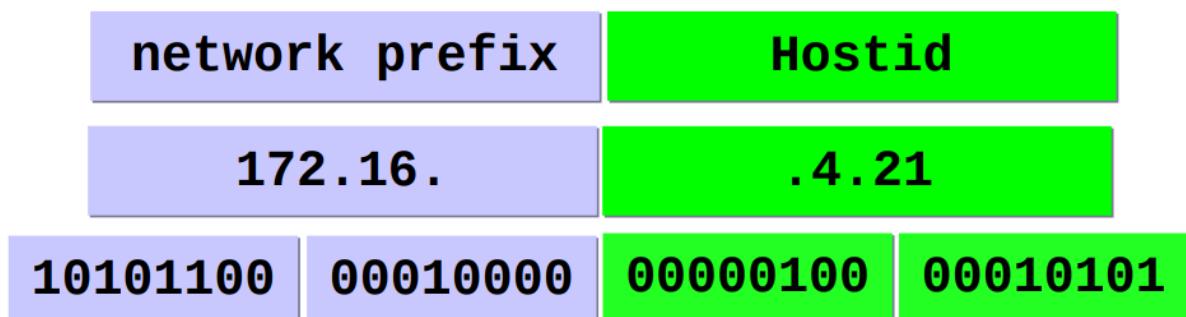
Son necesarias para rutear la información por la Internet. Y son direcciones lógicas.

Las direcciones están codificadas en dos partes: Red (Net) y Anfitrión (Host).



#### Clases:

- Clases A, pocas redes grandes.
- Clases B, más redes medianas.
- Clases C, muchas redes chicas.



Class	First Octet Range	Max Hosts	Format
A	1-126	16M	 1 Octet                            3 Octets
B	128-191	64K	 2 Octets                            2 Octets
C	192-223	254	 3 Octets                            1 Octet
D	224-239	N/A	 Multicast Address
E	240-255	N/A	 Experimental

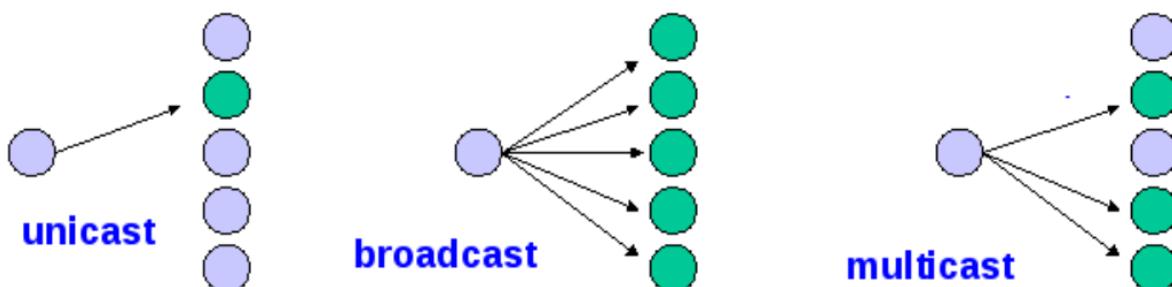
*Tipos:*

**Unicast:** destino a un host/interfaz en particular, son las más comunes. e.g: 172.16.4.21.

**Broadcast:** destino a todos los hosts en una red.

**Multicast:** destinada a un grupo de hosts en una red o varias redes. Clase D.

**Anycast:** destinada al primero que resuelva. IPv4 no hay casos especiales.



*Direcciones IP Especiales:*

- Loopback: unicast, red clase A. 127.0.0.1. La más utilizada: 127.0.0.1, localhost. Aunque podría ser cualquier otra.
- Dirección de red: la primera (zero). e.g. 172.16.0.0, 192.168.1.0.
- Dirección de broadcast: Directed Broadcast: la última (ones). e.g. 172.16.255.255, 192.168.1.255. Limited Broadcast: (all ones). 255.255.255.255.
- “Este host”, cuando aún no tiene asignada una dirección: 0.0.0.0 (Utilizada en BOOTP/DHCP).

*Direcciones Privadas:*

No tienen significado global, no son únicas. Se utilizan en Intranets. Redes autónomas sin conexión a Internet. Para conectarse a Internet requieren un proceso de transformación: NAT, RFC-1631. No deberían pasar a la Internet. Filtradas por routers de borde.

10.0.0.0 – 10.255.255.255, 1 Clase A.

172.16.0.0 – 172.31.255.255, 16 Clases B.

192.168.0.0 – 192.168.255.255, 256 Clases C.

### *. Subnetting IP:*

Proceso de dividir una red más grande en redes más pequeñas llamadas subredes. Esto se hace modificando la máscara de subred (subnet mask) para crear múltiples segmentos dentro de una red IP, lo cual mejora la organización, la seguridad y la eficiencia en el uso de direcciones IP.

Se toma una parte del hostid. Se utiliza para generar redes dentro de la red. Se agrega una "máscara" de bits. Para saber la subred se aplica un "AND" lógico.

Agregar un nivel más en la estructura: Red, Subred, Host.

Ejemplo usar un bloque clase B como 256 clases C:

network prefix	Subnet	Hostid
172.16 .	.4	.21
10101100	00010000	00000100
11111111	11111111	11111111
172.16.4.		.0

Las máscaras defaults:

Clase A: 255.0.0.0.

Clase B: 255.255.0.0.

Clase C: 255.255.255.0.

Ejemplo Subnetting Fijo, 4 Redes físicas, pueden direccionarse con una red IP: 4 redes requieren 2 bits  $2^2 = 4$ . Si fuesen 6, se requieren  $2^3 = 8 \sim 6$ . Siempre potencias de 2.

193.168.4.	0
193.168.4.	--
1100001	10101000
11111111	11111111
193.168.4.	
	00
	01
	..
	11

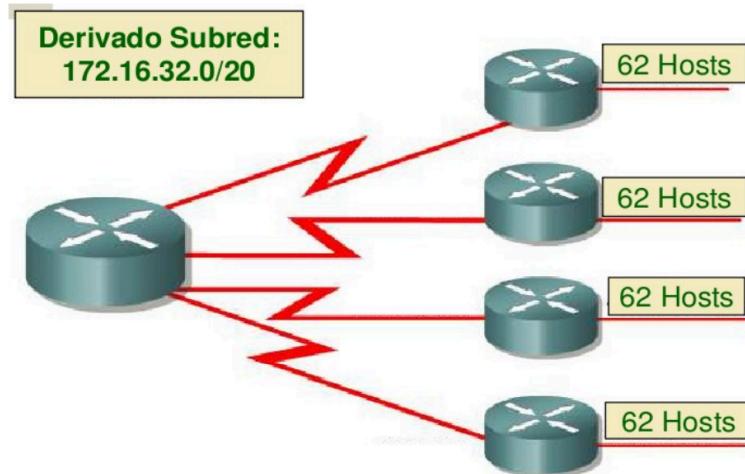
### *VLSM Subnetting:*

Variable Length Subnet Mask.

Es una técnica de subnetting que permite crear subredes de diferentes tamaños dentro de una misma red de clase. Esto es útil para optimizar el uso de direcciones IP, ya que permite asignar máscaras de subred de longitud variable según la cantidad de hosts que necesita cada subred.

La longitud de la máscara no tiene necesidad de ser para todas las subredes igual.

Ejemplo:



Paso 1: Calcular la máscara para la/s subred/es de mayor cantidad de hosts:

En el ejemplo se requieren 6 bits para hosts → la máscara es 255.255.255.192 o /26

Paso 2: Escribir 172.16.32.0 en forma binaria.

Paso 3: Dibujar una línea vertical por el límite original de la subred (/20).

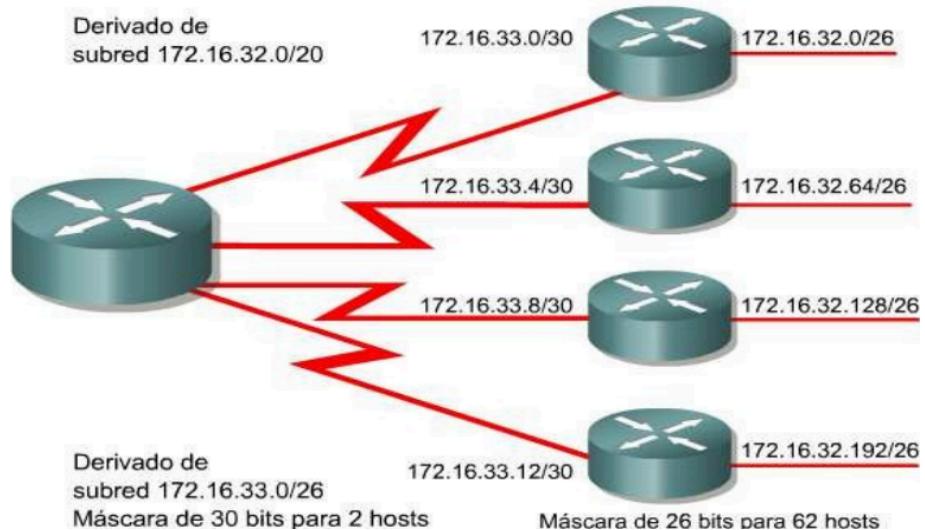
Paso 4: Dibujar una línea vertical por el segundo límite de subred (/26).

Paso 5: Calcular las 64 direcciones de subred usando los bits entre las 2 líneas . La figura muestra las primeras 5 subredes disponibles.

VLSM Address: 172.16.32.0/26			
In Binary 10101100.00010000.0010		0000.00	000000
1st subnet:	172 • 16	.0010	0000.00 000000 = 172.16.32.0/26
2nd subnet:	172 • 16	.0010	0000.01 000000 = 172.16.32.64/26
3rd subnet:	172 • 16	.0010	0000.10 000000 = 172.16.32.128/26
4th subnet:	172 • 16	.0010	0000.11 000000 = 172.16.32.192/26
5th subnet:	172 • 16	.0010	0001.00 000000 = 172.16.33.0/26

Las primeras 4 subredes se pueden usar para las LANs.

Siguientes pasos: a la 5ta subred la podemos dividir para obtener las direcciones de los enlaces WAN, usando una máscara /30, siguiendo el mismo procedimiento:

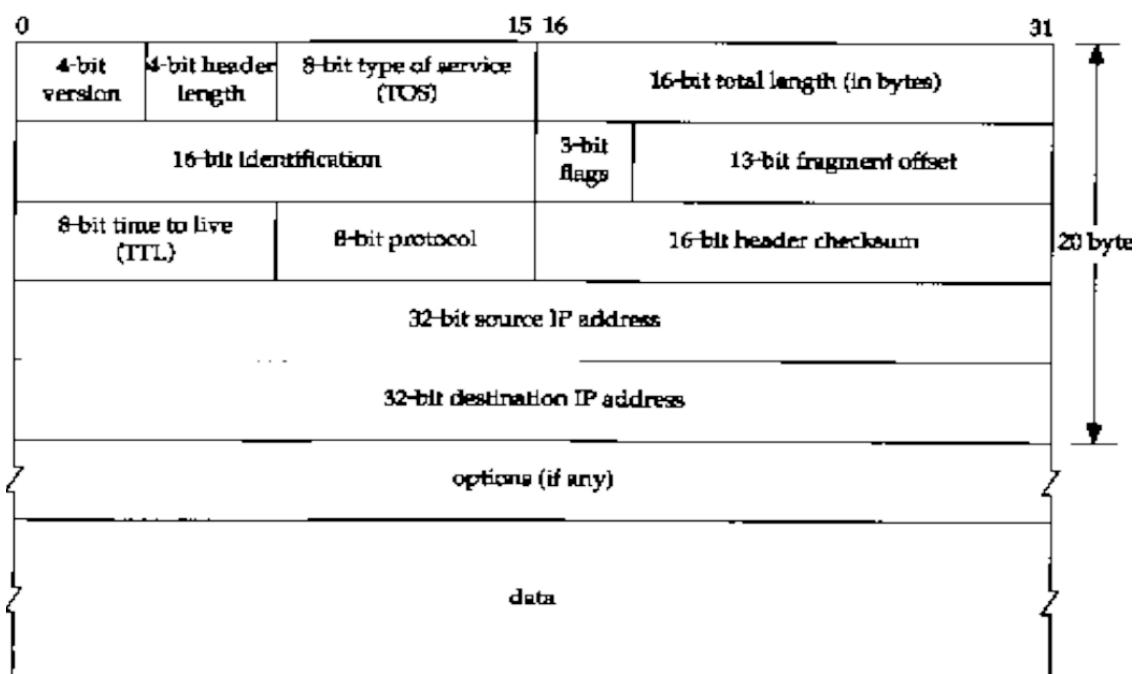


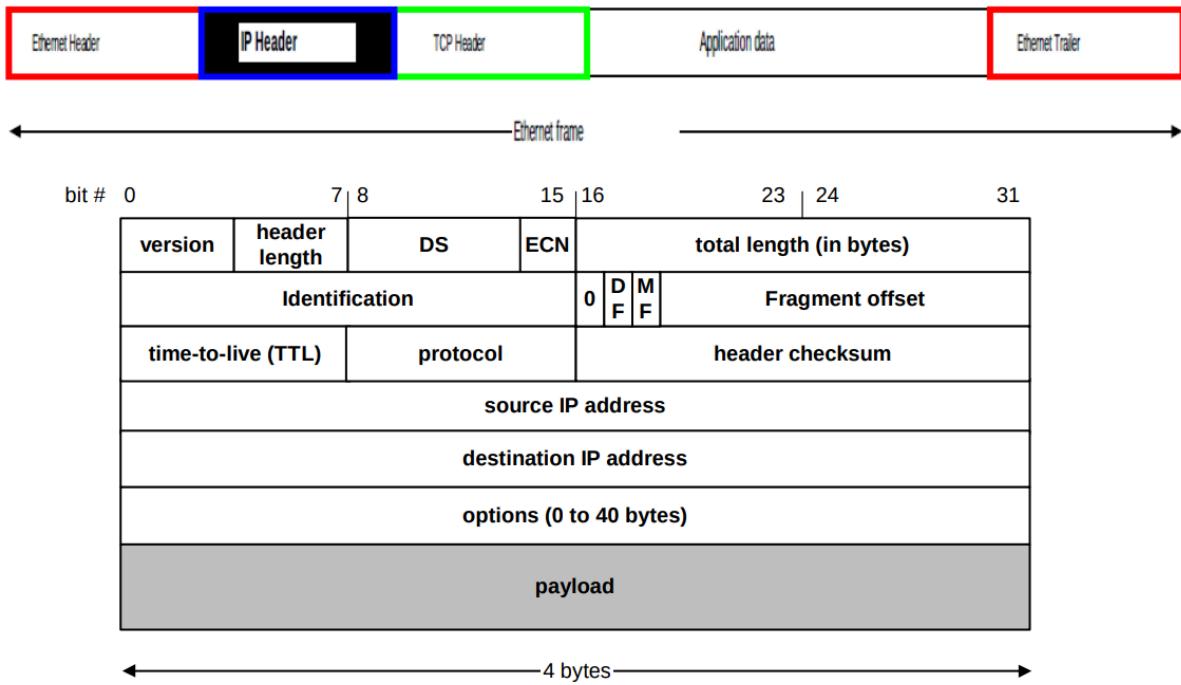
### CIDR - Supernetting:

Classless Inter Domain Routing.

Es una técnica que combina varias redes IP contiguas en una sola red mayor o "supernet". Esta técnica ayuda a simplificar las rutas y a optimizar el uso de direcciones IP, especialmente en el contexto de los enruteadores, donde puede reducir la complejidad de las tablas de enruteamiento.

### . Datagrama IPV4:





Version (4 bits): versión actual 4, la nueva 6.

Header length (4 bits): longitud en múltiplos de 4B.

DS/ECN field (1 byte): TOS (Type of Service), DSCP DiffService Codepoint.

Differentiated Service (DS) (6 bits): Usado para marcar QoS.

Explicit Congestion Notification (ECN) (2 bits): Usado en control de congestión con TCP.

Identification (16 bits): identificador único. Utilizado para la fragmentación.

Flags (3 bits):

Primero es 0.

DF bit (Do not fragment).

MF bit (More fragments).

Utilizados para la fragmentación.

Time To Live (TTL) (1 byte):

Cuantos saltos puede dar el datagrama.

Evita loops.

Emisor lo pone a un valor, e.g. 128 o 64.

Cada router por el que pasa lo decrementa en 1.

Si está más de un segundo también.

Si llega a un router que no está en la red destino y TTL=0, se descarta.

Protocol (1 byte): Para mux/demux.

Header checksum (2 bytes): 16 bit checksum del header solamente.

Options:

Security restrictions.

Record Route

Timestamp.

(loose) Source Routing

(strict) Source Routing.

Padding: agregado para ser múltiplo de 4B.

**. Ruteo:**

El ruteo consiste en seleccionar la interfaz de salida y el próximo salto. Involucra a los routers y hosts. Es necesario para que un paquete vaya de un extremo a otro.

**Routing:** Decisiones de “forwarding” en IP se llevan a cabo localmente. Deriva en conectividad entre los diferentes puntos de la red.

Se requiere recolectar y procesar un estado global, el cual se mantiene localmente en cada router. Los estados locales deben ser consistentes, si son inconsistentes la red no habrá convergido a un estado estable, se generan loops.

Se requiere: Consistencia, Completitud y Escalabilidad.

Se desea: Camino óptimo, Balanceo y Adaptabilidad.

**Tipos de Routing:** Todos los equipos en la red corren el protocolo ENRUTADO, por ejemplo IP (IPv4 o IPv6), pero los host no requieren correr protocolos de ENRUTAMIENTO/RUTEO. Por su parte, los routers requieren hacer el ENRUTAMIENTO, podrían trabajar de dos formas/ tipos de Routing: Ruteo Estático o Ruteo Dinámico. Una red compleja: muchos routers y enlaces requiere un protocolo de ruteo dinámico. Los routers pueden participar de forma activa en el routing: reciben, generan y propagan información, los hosts lo hacen de forma pasiva (no envían ni propagan información).

**Ruteo Estático:** Las rutas son establecidas por el administrador manualmente.

Es más propenso a errores. Pero ofrece mayor control.

Si se cambia la topología requiere cambios manuales en los routers.

Sirve cuando se tiene una red sencilla.

No tiene problemas de seguridad ni de incompatibilidad, y no implica costo de procesamiento extra.

Esquema NO escalable y NO tolerante a fallos.

**Ruteo Dinámico:** Requiere una configuración inicial por el administrador. Si se cambia la topología se adapta de forma automática.

Facilita, además, el mantenimiento cuando se tiene una red compleja, e implica costo de procesamiento extra.

Esquema escalable y tolerante a fallos.

Resolución de Problemas/Debugging, más complejo.

Caminos “óptimos” de acuerdo a la información manejada por el protocolo (métrica, costo).

**Routing Domain:** seleccionamos el/los protocolo/s de Ruteo en un Routing Domain, conjunto de routers con Routing Protocols comunes. Uno o más de estos incluidos en un AS.

Un Routing Domain o dominio de enrutamiento se refiere a un grupo de routers y redes que comparten una política común de enrutamiento.

**Sistema Autónomo (Autonomous System AS):** Es un conjunto de redes que se encuentra bajo la misma administración (podría ser gestionada por más de un operador de red), y utilizando un protocolo de ruteo o combinaciones para rutear internamente, independientemente de la red de su proveedor.

Posee una clara y única política de ruteo. Cada AS (Sistema Autónomo) en Internet (necesidad de intercambiar tráfico con otros dominios) debe tener un número identificador: ASN (AS Number).

**Clasificación de Protocolos de Ruteo Dinámico:**

- IGP (Interior Gateway Protocols), trabajan en el mismo AS:

RIP (Routing Internet Protocol) , v1, v2 (estándar IETF).

IGRP e EIGRP (-Enhanced- Interior Gateway Routing Protocol) (propietarios de cisco).

OSPF (Open Shortest Path First) , v2 , v3 (estándar IETF).

IS-IS (Intermediate System to Intermediate System) (estándar de la ISO)

- EGP (Exterior Gateway Protocols), trabajan entre diferentes AS:

GGP (Gateway to Gateway Protocol) (antecesor).

EGP (Exterior Gateway Protocol) (estándar IETF, en desuso).

BGP (Border Gateway Protocol) (estándar IETF).

**Tabla de ruteo:** estructura en hosts y routers (gateways) que indica cómo despachar un mensaje. Perspectiva del vecino, siguiente salto.

Estructura de tabla de ruteo:

Red Destino (Net/Mask).

Next Hop (Próximo salto).

Interfaz de salida.

**Host:** no despacha mensajes que recibe que no son para él. Despacha solo sus mensajes mirando su tabla de ruteo.

**Router:** Nodos intermedios, más de una interfaz, despacha mensajes mirando tabla de ruteo, desde cualquier interfaz.

**Host multihome:** tiene varias interfaces, no rutea.

**Ruteo:** seleccionar la interfaz de salida y el próximo salto. Routers y Hosts.

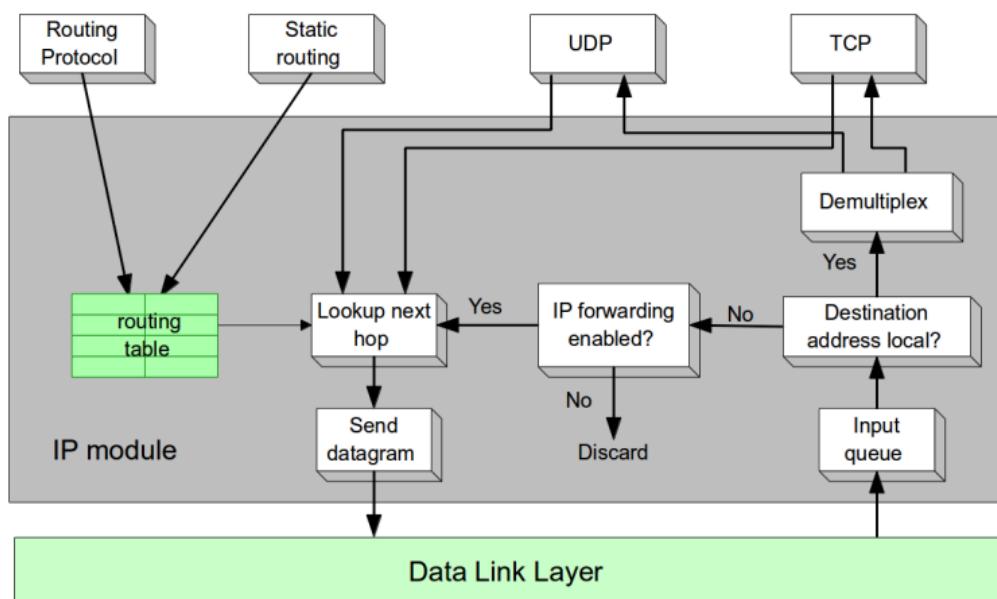
**Forwarding/Despacho:** pasar el paquete desde una interfaz de entrada hacia una de salida. Solo routers.

Se selecciona un port de salida en función de la dirección de destino y tabla de ruteo.

El ruteo es de control, alimentado por protocolos de enrutamiento (routing).

El forwarding es de datos, envía protocolos enrutados (routed).

**FIB:** Forwarding Information Base / Forwarding Table, el proceso de forwarding que se hace a partir de la RIB se optimiza generando una tabla más eficiente, FIB, por ejemplo implementada como TCAM en multilayer-switching.



**Tareas de Ruteo:**

Validación de datagrama: IP header.

Calcula checksum (solo header).

Leer IP destino.

Buscar en tabla de ruteo, seleccionar prefijo más largo ("best match").

- Decrementar TTL.
- Fragmentar (alternativo).
- Transmitir o Descartar.
- Generar ICMP (alternativo).

*Fragmentación:*

Debido a que hay diferentes capas de enlaces con diferentes MTUs.

Fragmentos múltiples de 8 bytes. Offset en unidades de 8 bytes.

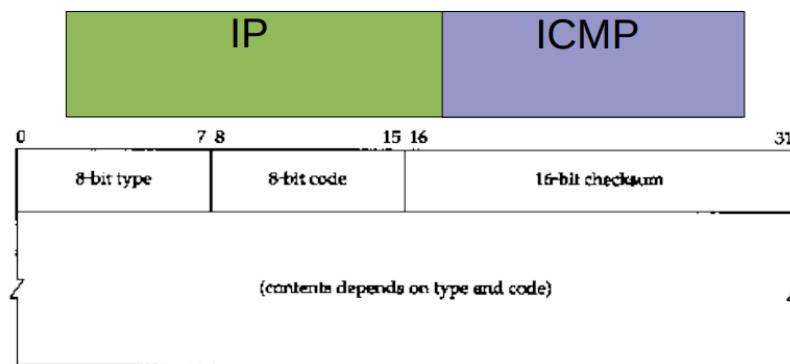
Se deben agregar los headers.

version	header length	DS	ECN	total length (in bytes)
Identification		0	D M F F	Fragment offset
time-to-live (TTL)	protocol		header checksum	

## ICMP

- . Internet control Message Protocol.
- . Es un protocolo “Helper” de IP, ya que IP carece de control, el cual es dado por un protocolo auxiliar.
- . ICMP no le agrega confiabilidad a IP, solo brinda un “feedback” para poder resolver problemas en la red.
- . ICMP se encapsula en IP:

### Formato de Mensaje:



. Mensajes ICMP:

**ICMP Ping (Echo):** Pensado para probar conectividad IP entre dos hosts. Sirve para medir el RTT min/avg/max/dev y loss, de esta forma poder diagnosticar problemas. Si un nodo recibe un ICMP Echo Request, debe responder copiando el contenido con un Echo Reply (PONG).

**Destino Inalcanzable:** Sirve para indicar que una red, un host o un puerto es inalcanzable, por diferentes causas.

Host Inalcanzable (Host Unreachable): Puede darse por no estar encendido el host, no responder ARP, etc.

Red Inalcanzable (Network Unreachable): No tiene el router una ruta en la tabla de ruteo a esta red. Puerto Inalcanzable (Port Unreachable): No hay un proceso UDP en el puerto.

Los mensajes requieren fragmentación.

**TTL Expirado:** Significa que el tiempo de vida ha expirado.

En realidad es el hop count con el cual salió el mensaje que ha expirado.

Time Exceeded:

Tiempo Excedido en viaje.

Tiempo Excedido en re-ensamblado.

## **DYNAMIC HOST CONFIGURATION PROTOCOL (DHCP)**

Un host para conectarse a una red IP requiere 3 parámetros + 1: Dirección IP - Máscara de Subred - Gateway (Puerta de enlace predeterminada) - Servidor DNS (opcional).

Para conectarse a la red local requiere de una Dirección IP + una Máscara de red.

Para conectarse a otras redes requiere del Router por default (Default Gateway).

Para usar servicios requiere de Servidor(es) de DNS.

La forma de obtener los parámetros es a través de:

Configuración manual/estática: Difícil de mantener, no escalable (recolección, re-assign.), no sirve para movilidad.

Configuración dinámica: RARP | ICMP | BOOTP | DHCP.

. *Definición y Funcionamiento:*

DHCP es un protocolo “Helper” de IP, de L3. Que como está montado sobre UDP se lo suele considerar protocolo de nivel de aplicación, y sirve tanto para IPv4 o IPv6.

Este protocolo permite la configuración dinámica de los parámetros de red de los hosts.

Los hosts al arrancar solo tienen acceso a su red local de forma broadcast. En esta red existen uno o más servidores de auto-configuración llamados ‘DHCP servers’.

Los host sin parámetros de red envían requerimientos y los servidores los atienden asignando los valores que brindan conectividad.

El parámetro se reserva por un tiempo.

1. Descubrimiento (DHCP Discover):

Cuando un dispositivo (cliente DHCP) se conecta a la red, envía un mensaje de difusión para localizar un servidor DHCP.

2. Oferta (DHCP Offer):

Un servidor DHCP responde con una oferta que incluye una dirección IP disponible y otros parámetros de configuración.

3. Solicitud (DHCP Request):

El cliente elige una oferta (en caso de recibir varias) y envía una solicitud para confirmar el uso de esa dirección IP.

4. Confirmación (DHCP Acknowledgment):

El servidor DHCP confirma la asignación enviando un mensaje de reconocimiento con la dirección IP y las configuraciones finales.

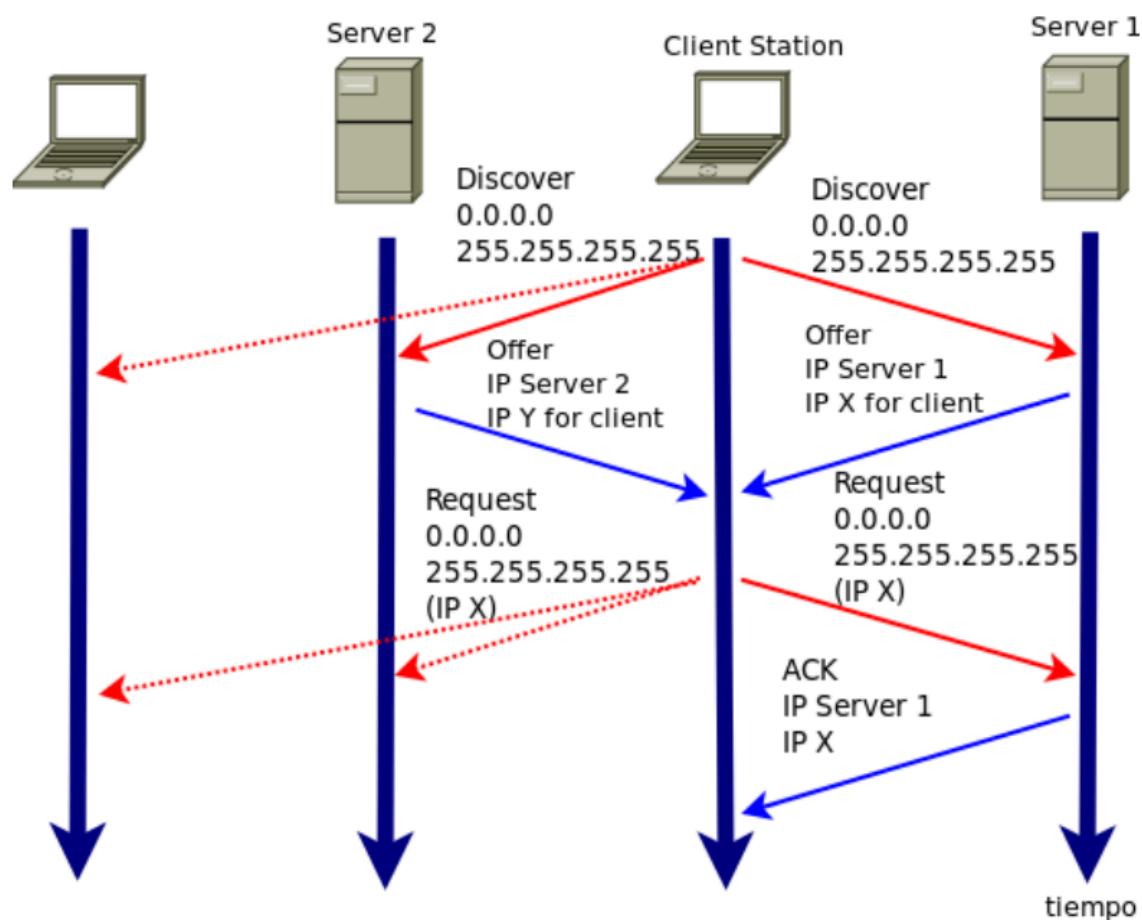
. *Escenarios comunes de DHCP:*

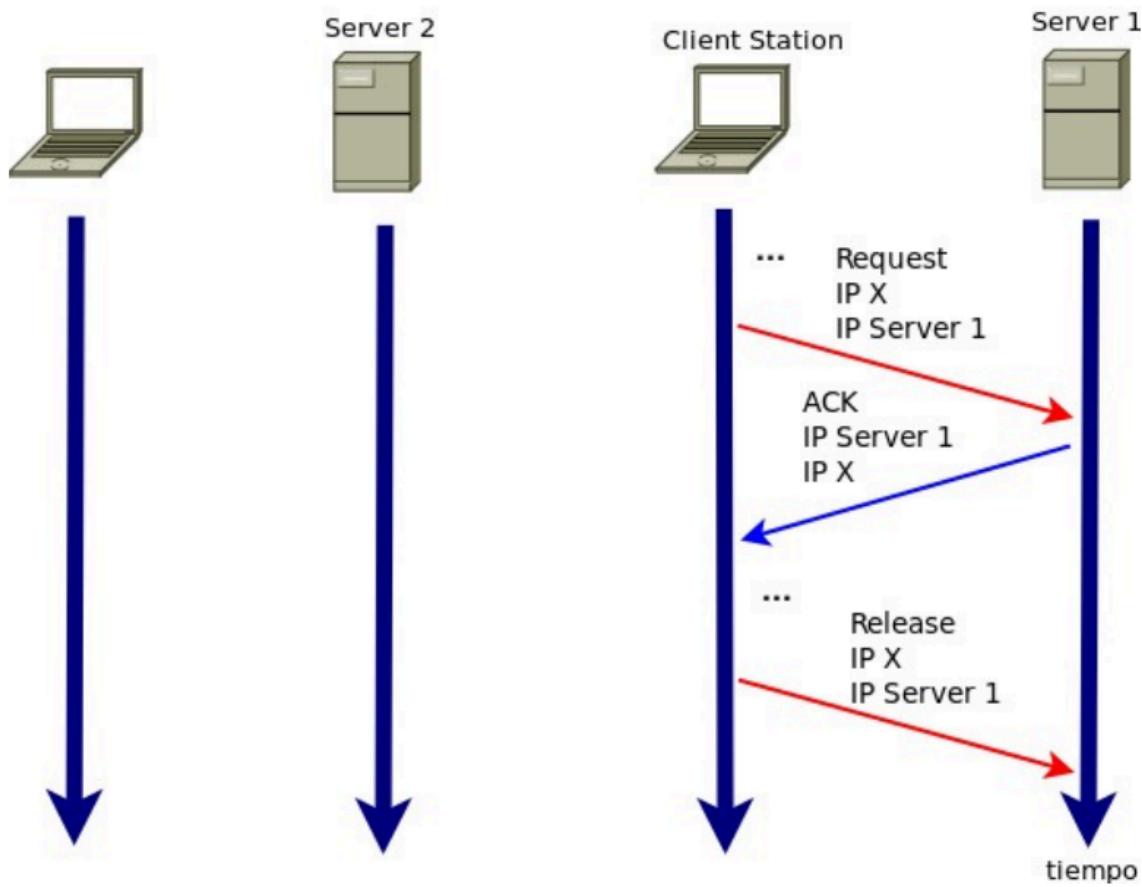
**Red doméstica:** El router actúa como servidor DHCP, asignando direcciones IP a dispositivos como laptops, teléfonos o televisores inteligentes.

**Red empresarial:** Servidores dedicados administran DHCP en redes más complejas con múltiples subredes.

**Red pública:** En lugares como cafeterías o aeropuertos, DHCP se utiliza para asignar direcciones IP temporales a visitantes.

. Ejemplo:





#### Etapa 1: Descubrimiento (DHCP Discover)

El cliente envía un mensaje de difusión (Discover) para localizar servidores DHCP disponibles. Este mensaje tiene:

Dirección IP de origen: 0.0.0.0 (porque el cliente no tiene asignada una IP aún).

Dirección IP de destino: 255.255.255.255 (difusión a toda la red local).

#### Etapa 2: Oferta (DHCP Offer)

Ambos servidores DHCP (Server 1 y Server 2) responden al cliente con una oferta:

Server 1 ofrece una IP (IP X) para el cliente.

Server 2 ofrece otra IP (IP Y) para el cliente.

Cada oferta incluye la dirección IP propuesta, máscara de subred, puerta de enlace y tiempo de arrendamiento.

#### Etapa 3: Solicitud (DHCP Request)

El cliente elige una de las ofertas (en este caso, la de Server 1) y envía un mensaje de Request a toda la red:

En este mensaje, el cliente solicita específicamente la IP X ofrecida por Server 1.

Esto indica a Server 2 que su oferta ha sido rechazada.

#### Etapa 4: Confirmación (DHCP Acknowledgment - ACK)

Server 1 envía un mensaje de ACK (reconocimiento), confirmando la asignación de la dirección IP X al cliente.

A partir de este momento, el cliente puede usar la IP X junto con los demás parámetros proporcionados (máscara, puerta de enlace, DNS).

#### Etapa 5: Solicitud y confirmación de la IP asignada

Request (Solicitud):

El cliente (Client Station) solicita formalmente la dirección IP X que le fue ofrecida por Server 1 (esto ocurre como parte de la etapa anterior ya descrita).

Este mensaje reafirma que el cliente desea utilizar IP X.

ACK (Acknowledgment - Reconocimiento):

Server 1 confirma la asignación de IP X al cliente mediante un mensaje ACK.

En este punto, el cliente ya tiene configurada la dirección IP junto con los demás parámetros de red proporcionados.

#### **Etapa 6: Liberación de la dirección IP**

Release (Liberación):

Posteriormente, el cliente envía un mensaje Release a Server 1 para liberar la dirección IP X.

Esto ocurre, por ejemplo, cuando:

El cliente se desconecta de la red.

El cliente ya no necesita la dirección IP.

Con este mensaje, el servidor puede marcar la IP X como disponible para asignarla a otro cliente en la red.

Los mensajes DISCOVER y REQUEST se envían de forma broadcast, mientras que OFFER se envía de manera unicast aunque podría ser broadcast.

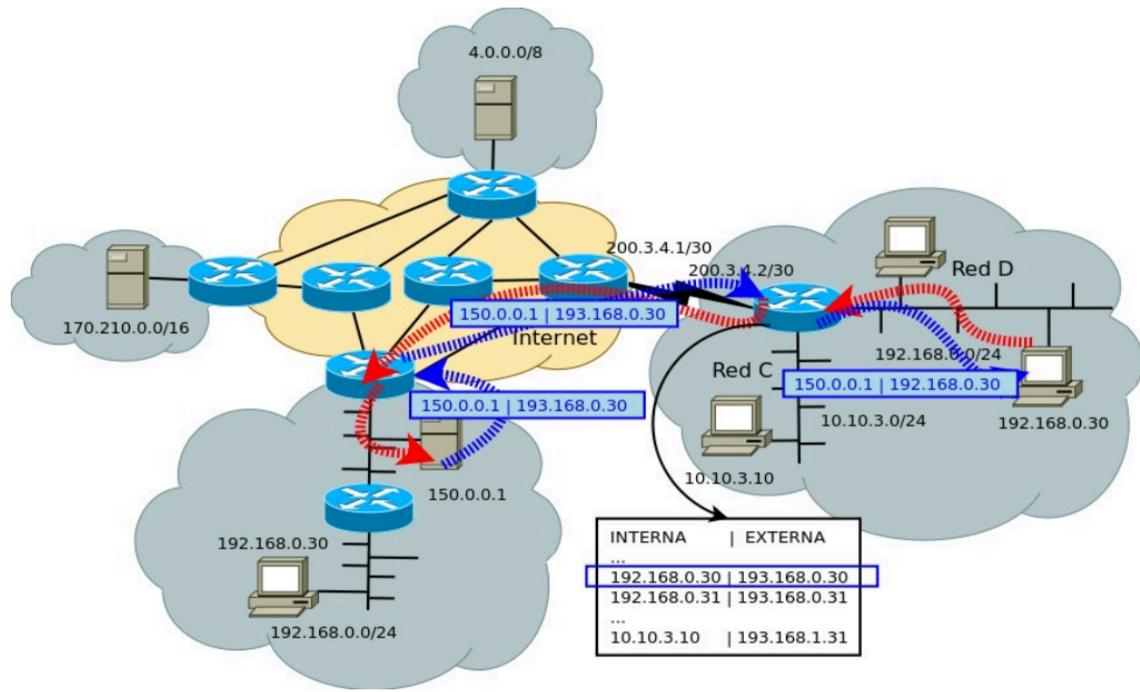
#### **NAT**

. Network Address Translation es un proceso de traducción de direcciones. Se utiliza para traducir direcciones privadas dentro de un espacio no privado (no “enrutable” en Internet) a direcciones públicas para un espacio público. Permite que múltiples dispositivos en una red compartan una única dirección IP pública. Por ejemplo, en una red doméstica donde hay varios dispositivos que desean acceder a Internet a través de un router. NAT traduce las direcciones IP privadas de estos dispositivos en una única dirección IP pública, lo que permite que se comuniquen con Internet. El router lleva un registro de estas traducciones para dirigir correctamente los datos a los dispositivos locales. Esto permite que haya un ahorro de direcciones públicas IPv4 debido a que las privadas se pueden repetir en los diferentes espacios privados (pero no dentro de uno mismo) y esto permite que varios dispositivos que están en una misma red privada puedan usar la misma dirección pública o que necesariamente no haya una dirección pública para cada dirección privada. NAPT tiene en cuenta el puerto y toca también la capa de transporte.

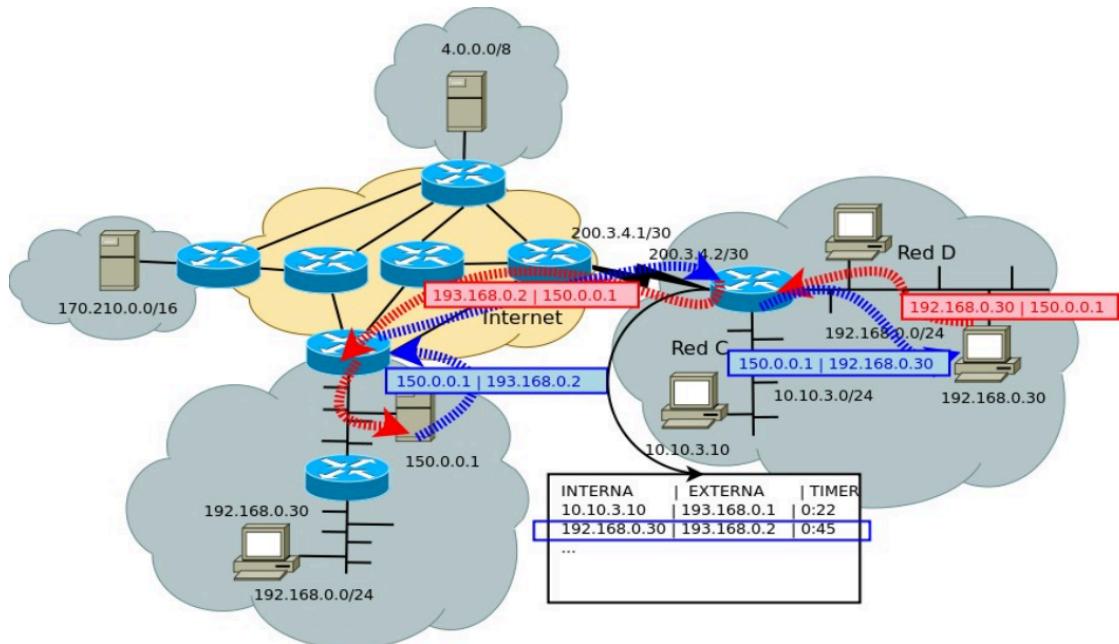
. *NAT Básico:*

Se mapea una dirección IPv4 privada a una dirección IPv4 pública.

**Estático:** Si se hace de forma estática requiere tantas direcciones públicas como privadas. Permite acceso en ambas direcciones.



**Dinámico:** Si se hace de forma dinámica no es necesario, pero sí se requiere un timer por cada entrada. Limitando el acceso simultáneo de acuerdo al pool pub.

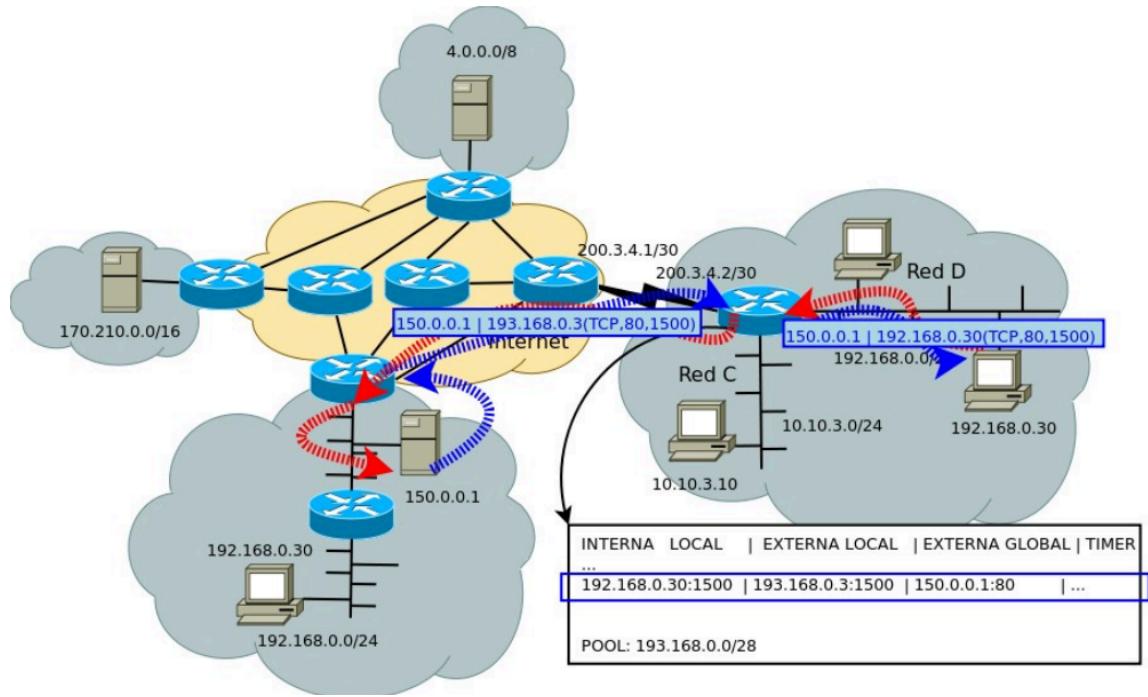


#### . NAPT:

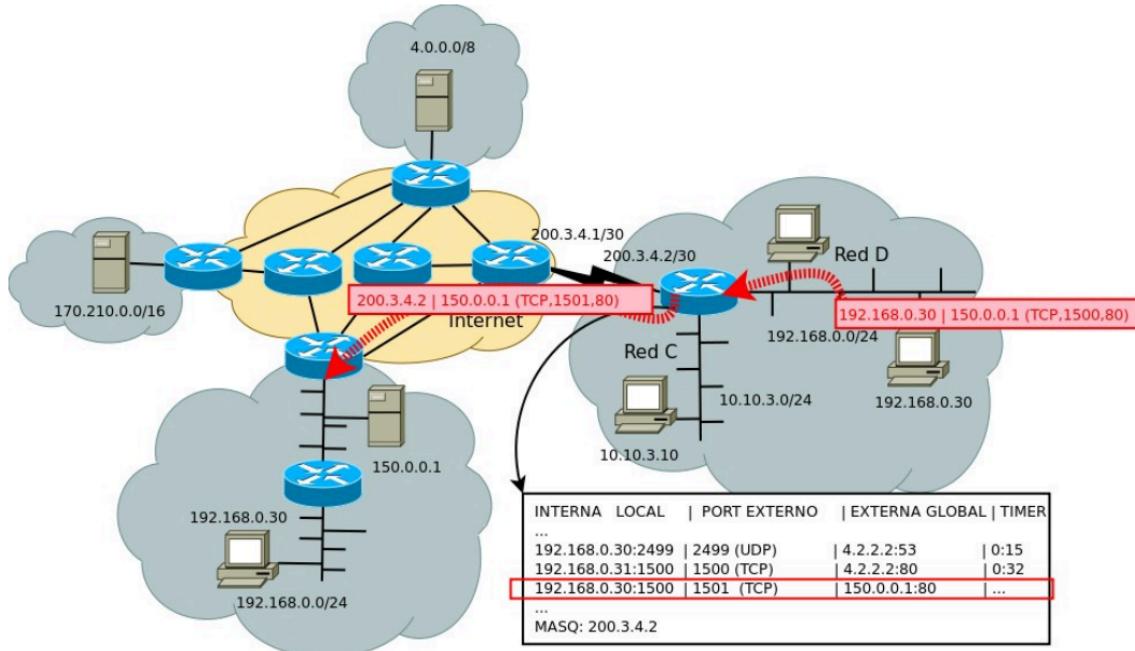
NAT no es implementable cuando se tiene un pool chico de direcciones o no se poseen direcciones públicas asignadas. En ese caso se debe trabajar con campos de la capa de transporte o del payload. NAPT es conocido como PAT (Port Address Translation): "one-to-many". Se utilizan los puertos de los protocolos u otros valores como ICMP Identifier para resolver el mapeo. Se pueden usar timers y sesión del protocolo.

En la tabla de traslaciones se mantienen el protocolo y los puertos origen y destino. Se intenta conservar el puerto origen, pero si está "ocupado" se debe reemplazar por otro. El dispositivo debe "violar" los límites impuestos por la división en capas.

**Dinámico sobre pool:** Utiliza un pool y hace PAT sobre este.

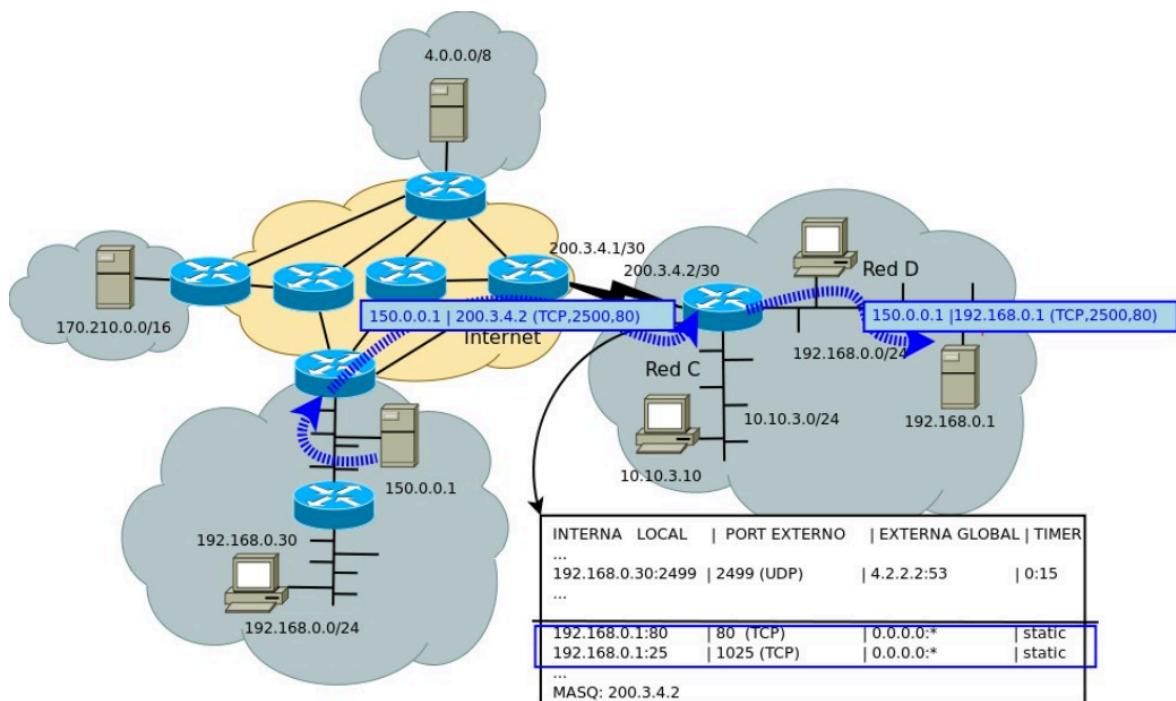


**Dinámico sobre dir:** Utiliza la dir. IP externa y hace overloading/masquerading sobre esta.



#### *. Port Forwarding:*

Overloading/Masq no permiten acceso desde “afuera” hacia “adentro”. Solo se permite entrar tráfico de conexiones generadas internamente. Mediante Port Forwarding (Re-envío de puerto) se permite poder tener servicios en una red privada accesibles desde “afuera”. No se requiere NAT estático, se implementa con NAPT y mapeo reverso estático de puertos.



## IPv6

- . Es la versión más nueva del protocolo IP. Proporciona mayor espacio de direcciones (128 bits), un formato de cabecera simplificado y menor overhead de procesamiento.
- . Es necesaria su implementación para solucionar las limitaciones (en cuanto a direcciones IP) que traía IPv4. Garantiza que haya suficientes direcciones IP únicas para todos los dispositivos. Además, IPv6 proporciona mejoras en seguridad y eficiencia en comparación con IPv4.
- . En IPv4 había un campo opcional cuyo tamaño podría variar, haciendo que varíe la longitud del encabezado. En cambio, en IPv6 todos los campos tienen un tamaño fijo, haciendo que el encabezado tenga una longitud constante. Y además se eliminan las opciones.
- . En IPv6, el campo de checksum es eliminado para simplificar el procesamiento de los paquetes en los routers y dispositivos, dejando esta verificación a las capas superiores (TCP y UDP). En cuanto a estas, su obligatoriedad no ha cambiado.
- . ICMP en IPv6 es obligatorio, ya que cumple funciones similares a las que desempeña en IPv4, aunque con algunas diferencias:
  1. Descubrimiento de vecinos: ICMPv6 ND reemplaza ARP en IPv4, mapea direcciones IPv6 a direcciones de enlace para envío de paquetes.
  2. Gestión de errores: ICMPv6 informa sobre errores de entrega, como paquetes demasiado grandes, tiempo de vida agotado y destino inalcanzable.
  3. Redirección de rutas: ICMPv6 informa a los hosts sobre rutas más eficientes en la red.
  4. Pruebas de conectividad: ICMPv6 incluye mensajes Echo Request y Echo Reply para pruebas de conectividad, similar a ping en IPv4.
  5. MLD (Multicast Listener Discovery): ICMPv6 se utiliza para el descubrimiento de escuchadores de multidifusión en redes IPv6.

. *Neighbour Discovery*:

Las funciones del protocolo Neighbour Discovery son las de mapear direcciones lógicas (IPv6) a direcciones de Hardware (MAC, EUI-48, EUI-64):

- . Descubrimiento de vecinos: Permite que los nodos en una red determinen la dirección de enlace local (dirección MAC) de otros nodos dentro de la misma red, similar a lo que hace el protocolo ARP (Address Resolution Protocol) en IPv4.  
Se utiliza para resolver direcciones IP en direcciones de capa de enlace.
- . Descubrimiento de vecinos inalcanzables: Facilita la detección de nodos vecinos que ya no están disponibles o accesibles en la red. Esto asegura que un nodo no intente comunicarse con un vecino no funcional, optimizando el uso de recursos.
- . Descubrimiento de routers: Permite a los nodos identificar la presencia de routers en la red local, obteniendo información sobre los routers disponibles y seleccionando uno para la comunicación con redes externas.
- . Determinación de prefijos de red y configuración sin estado: A través de mensajes de "Router Advertisement" (RA), un nodo puede aprender el prefijo de red y otra información esencial para auto-configurarse sin requerir un servidor DHCP.
- . Redirección de tráfico: Los routers utilizan NDP para informar a un nodo que hay una mejor ruta para llegar a un destino específico. Esto es útil para optimizar el enrutamiento en una red local.
- . Duplicated Address Detection (DAD): Garantiza que una dirección IPv6 no esté duplicada dentro de la red antes de asignársela a un nodo. Esto evita conflictos de direcciones.

## Clase 7 | Capa de Enlace

. La función de la capa de enlace es mover un datagrama desde un nodo hasta otro adyacente a través de un único enlace de comunicaciones. Su función principal es proporcionar una interfaz entre la capa de red (capa 3) y el medio físico subyacente, como cables, fibra óptica o enlaces inalámbricos. En RED LAN.

. Servicios que presta:

1. Entramado (framing):

- Encapsulado del datagrama en la trama, agregando encabezado (header) y cola (trailer).

2. Acceso al enlace:

- Acceso al canal si es un medio compartido.
- Direcciones "MAC" utilizadas en los encabezados de las tramas para identificar el origen y el destino.

3. Entrega confiable:

- Entre nodos adyacentes.
- Rara vez utilizados en enlaces de pocos errores (fibra óptica).

4. Control de flujo:

- Acuerdo entre nodos emisor y receptor (adyacentes).

5. Detección de errores

- Errores causados por atenuación de señal.
- El receptor detecta presencia de errores.

6. Corrección de errores.

7. Half-duplex y full-duplex:

- Half-duplex puedo recibir y transmitir pero no al mismo tiempo.

. Los servicios de detección y corrección de errores y control de flujo son ofrecidos también por la capa de transporte. La diferencia entre ambos radica en que la capa de enlace se enfoca en aspectos locales del enlace, mientras que la capa de transporte aborda la transferencia extremo a extremo a través de redes más amplias. Además, el transporte proporciona una fiabilidad y garantías que el enlace no.

. *Direccionamiento:*

Las máquinas en una red Ethernet se identifican mediante una dirección MAC.

Estas direcciones también se conocen como direcciones de capa de enlace. Son direcciones físicas de 48 bits (6 bytes) y se expresan en hexadecimal. Los primeros 24 bits (3 bytes) identifican al fabricante de la tarjeta (OUI) y los siguientes 24 bits son únicos para la interfaz de red.

FF:FF:FF:FF:FF:FF es la dirección de broadcast, y su función principal es enviar información a todos los dispositivos en la red sin la necesidad de conocer sus direcciones MAC individuales.

. *Dispositivos:*

**HUB:** Actúa como un repetidor y simplemente repite las señales a todos los puertos. No divide dominios de colisión ni de broadcast.

**Switch:** Examina las direcciones MAC para enviar tramas solo al puerto específico donde se encuentra el destinatario. Divide dominios de colisión. No divide dominios de broadcast.

**Bridge:** Conecta dos segmentos de red, examina las direcciones MAC y aprende las ubicaciones de las direcciones MAC en ambos lados. Puede dividir dominios de colisión, no divide dominios de broadcast (es como el switch solo que tiene menos puertos).

## **CSMA/CD**

. El algoritmo de acceso al medio en Ethernet es fundamental para regular cómo los dispositivos conectados a una red Ethernet comparten y acceden al medio físico (como un cable) para transmitir datos, evitando colisiones y garantizando una comunicación eficiente. Este algoritmo, conocido como CSMA/CD (Carrier Sense Multiple Access with Collision Detection), tiene varias funciones clave:

**Evitar colisiones:** Garantiza que dos dispositivos no transmitan simultáneamente en el mismo medio, lo que provocaría una colisión de datos.

**Coordinar el acceso:** Permite que múltiples dispositivos comparten un mismo medio de comunicación de manera ordenada y equitativa.

**Maximizar la eficiencia:** Reduce el tiempo perdido en retransmisiones innecesarias, mejorando el rendimiento de la red.

**Recuperarse de colisiones:** En caso de que ocurra una colisión, el algoritmo establece cómo los dispositivos deben detectarla y retransmitir sus datos.

. *Funcionamiento:*

**Carrier Sense (CS):** Antes de transmitir, un dispositivo "escucha" el medio para verificar si está ocupado.

Si detecta actividad, espera hasta que el medio esté libre.

**Multiple Access (MA):** Como múltiples dispositivos comparten el medio, cada uno debe esperar su turno para transmitir, basado en la disponibilidad del medio.

**Collision Detection (CD):** Si dos dispositivos transmiten al mismo tiempo, el algoritmo detecta la colisión porque la señal resultante es diferente de la esperada.

Ambos dispositivos interrumpen la transmisión y emiten una señal de jam (ruido) para indicar la colisión.

**Backoff (retroceso):** Cada dispositivo espera un tiempo aleatorio antes de intentar retransmitir, reduciendo la probabilidad de una nueva colisión.