

## Capa De Aplicación

### 1. ¿Cuál es la función de la capa de aplicación?

La capa de aplicación da soporte a las aplicaciones de red; incluye los protocolos HTTP (Web), SMTP (email) y FTP (transferencias de archivos) entre otros.

### 2. ¿Qué diferencia un programa de un proceso?

Un proceso es un programa en ejecución en un terminal. Una aplicación de red se compone de procesos en distintos terminales -por ejemplo, un software tipo browser en un terminal y un servidor Web en otro equipo, para el caso de la aplicación Web.

### 3. Si dos procesos deben comunicarse:

a. **En diferentes máquinas:** Utilizan en intercambio de mensajes a través de la red. El proceso emisor crea y envía los mensajes, y el receptor los recibe y responde si es apropiado hacerlo. Los protocolos de la capa de aplicación definirán el formato y orden en que deben enviarse esos mensajes, y las acciones a tomar en consecuencia a la llegada de un mensaje.

b. **En la misma máquina:** Utilizan las facilidades de comunicación inter-proceso que soporte el SO huésped (uso de señales, memoria compartida...).

### 4. Explique brevemente cómo es el modelo Cliente/Servidor. Dé un ejemplo de un sistema Cliente/Servidor en la "vida cotidiana" y un ejemplo de un sistema informático que siga el modelo Cliente/Servidor. ¿Conoce algún otro modelo de comunicación?

En la arquitectura Cliente-Servidor un host **cliente** se comunica con otro, **servidor**, enviando solicitudes. Los clientes se comunican sólo con el servidor -no entre ellos-, y posiblemente desde *IP dinámica* -al contrario del servidor, con *IP fija*-. Es común que un host actúe como cliente y servidor de un servicio, o que haya solicitudes y envío de información en ambas direcciones; en este caso, se denomina *cliente* al host que comienza la sesión. Ejemplo: Un servidor Web -i.e. Apache- y un browser -i.e. Mozilla Firefox-.

Otros modelos de comunicación son: P2P (Peer to Peer, los clientes se comunican entre ellos directamente y posiblemente con un servidor que administra las IP de los clientes conectados) y sistemas híbridos (un servidor coordina algunas tareas y administra las IP cliente, y los clientes se comunican entre ellos tras interactuar con el servidor).

### 5. Describa la funcionalidad de la entidad genérica User agent.

Un **user agent** es la interfaz entre el usuario final y una aplicación de red (por ejemplo, el browser en el caso de la WEB). Implementa la parte cliente de algunos protocolos de aplicación.

**6. Direccionamiento. ¿Es suficiente la dirección de red o nombre de host para que dos máquinas intercambien datos? Justifique (suponga el caso de un servidor WEB con N clientes)**

Para que dos procesos en dos equipos distintos intercambien información se precisa la *dirección IP o nombre* de cada equipo, así como el "*número de puerta*" que identifique al proceso destino (necesario por si el host tiene varios procesos que utilicen la red). Esta combinación se denomina **socket**.

Por ejemplo, en el caso de un servidor WEB, el protocolo HTTP utiliza por defecto el port 80. Múltiples clientes pueden enviar solicitudes al servidor conociendo este port estándar, y en su solicitud envían tanto su IP como el port en que esperarán la respuesta. Los números de port inferiores a 1023 se mantienen reservados para distintos protocolos y servicios, mientras que las aplicaciones pueden utilizar números mayores hasta el 65535.

**DNS** Requerimientos: Debe usar el Live CD de Lihuen GNU/Linux provisto

**7. Investigue y describa cómo funciona el DNS. ¿Cuál es su objetivo?**

Un DNS es una *base de datos distribuida* implementada en una jerarquía de servidores de nombres, y un *protocolo de nivel de aplicación* que permite a los hosts y dichos servidores comunicarse para proveer el servicio de traducción entre la dirección mnemónica y la dirección IP.

Cuando una aplicación precisa traducir una dirección mnemónica a IP, el host invocará un mensaje hacia el cliente DNS con el nombre a traducir. El cliente envía un mensaje de petición a través de la red, y luego de un retardo recibe la dirección de IP enviada por el servidor DNS.

**8. ¿Qué es un root server? ¿Qué es un generic top-level domain (gtld)?**

Los servidores DNS se organizan en forma jerárquica. Un host hace su solicitud a un **servidor local de nombres** que, si no posee la traducción, se transforma en un cliente consultando a un **servidor raíz de nombres -root server-**. Si el root server conoce al **servidor autorizado de nombres** para el dominio que se intenta resolver, envía la solicitud terminando la cadena; si lo desconoce, consulta a servidores intermedios hasta que eventualmente alguno lo conozca. En cada paso, un servidor podría tener registrado el IP del dominio consultado o mantenerlo en cache por una consulta anterior, ofreciendo una respuesta directa.

Para un nombre de dominio, se denomina *Top Level Domain* (TLD) a la etiqueta más a la derecha, desde el último punto. Un **generic top-level domain** es una clasificación de los

dominios de Internet, -un subconjunto de los TLD's- que actualmente incluye sólo las terminaciones: .com, .info, .net y .org. Otras clasificaciones mantienen por ejemplo códigos por país o administrados por distintas organizaciones.

## 9. ¿Qué es una respuesta del tipo authoritative?

Una respuesta authoritative es aquella entregada por el servidor oficial para el nombre de dominio consultado -es decir, que ningún servidor local, raíz ni intermedio respondió con información de caché-.

## 10. ¿Qué diferencia una consulta DNS recursiva de una iterativa?

- **Consulta DNS Recursiva:** Un host realiza una consulta a un servidor, éste se encarga de obtener la resolución de IP y devuelve la respuesta al host.

- **Consulta DNS Iterativa:** Un name server realiza una consulta a un servidor; si este no tiene la respuesta le devuelve la IP del próximo servidor. El name server original se encarga de realizar nuevas solicitudes hasta obtener la respuesta.

Típicamente, un **local name server** realiza **consultas iterativas**, y el resto recursivas -ya que tienen mayor volumen de consultas que el local name server-.

## 11. ¿Qué es el resolver?

Se denomina **resolver** a la parte cliente del DNS, encargada de iniciar y secuenciar las consultas. Usualmente trabajan de modo iterativo, aunque algunos sólo se conectan a un name server único y trabajan con consultas recursivas.

## 12. Describa cada uno de los siguientes tipos de registros de DNS: PTR, A, NS, MX, SOA y CNAME.

Los name servers mantienen **registros de recursos** (RR) con el mapeo hostname <=> IP, en una cuádrupla (name, value, type, TTL). Las respuestas a consultas incluyen uno o varios RR de distintos tipos. El campo TTL -Time To Live- indica cuánto tiempo mantener el RR en caché. Los distintos *tipos* son:

- **A.** El campo name es un **hostname**, value su **IP**. Es el único que traduce hostname<=>IP, y representa al *nombre canónico* (cualquier otro nombre para el hosts será un alias con RR tipo CNAME). Puede haber más de un registro tipo A con igual hostname y distintas IP en el caso de *servidores replicados* para realizar un *balance de carga*. Las respuestas DNS incluirán la lista de asociaciones pero el servidor irá alternando las posiciones para distribuir los accesos de los clientes.
- **PTR (Pointer Record).** Exactamente al revés que un registro tipo A, indica en name el **IP** de un host y en value su **hostname**. Permite la resolución reversa.

- **NS (Name Server Record).** El campo name es un **hostname**, value el **hostname de un name server** a quien seguir consultando. Permite continuar las consultas hasta obtener la traducción definitiva.
- **MX (Mail eXchange Record).** El campo name es un **alias para un mail server**, value incluye un valor numérico que indica la **prioridad y el hostname del mail server**. Permite que los hosts del mail server tengan una dirección mas simple a través del alias. Si un programa desea enviar un correo a un host de ese dominio, debe intentar primero con el de menor valor de prioridad -prioridad más alta- y sólo si falla intentar con el resto.
- **CNAME (Canonical Name Record).** El campo name es un **alias para un hostname**, value el **nombre canónico** o real del mismo (por ejemplo, el uso de alias permite que un sitio sea accedido como `www.sitio.com` y como `sitio.com - alias-`).

Otro tipo de registro es:

- **SOA.** Por *Start of Authority*, indica que los registros siguientes de la base de datos corresponden a información autoritativa. Indica: nombre canónico del NS, usuario responsable -dirección de email con un punto en vez del arroba-, número de versión -serial- del archivo de configuración del NS, refresh o tiempo de espera de servidores secundarios para pedir el SOA al primario, expire o tiempo en segundos que tardarán los servidores secundarios en descartar los datos si no logran contactar al primario, y el mínimo valor TTL para los registros de recursos siguientes.

13.Utilizando el Live CD, utilice alguno de los siguientes comandos: `nslookup`, `host` o `dig`, para obtener:

**\$ nslookup www.redes.unlp.edu.ar**

Server: 127.0.0.1

Address: 127.0.0.1#53

Name: `www.redes.unlp.edu.ar`

Address: `127.0.0.1`

**\$ host www.redes.unlp.edu.ar**

`www.redes.unlp.edu.ar` has address `127.0.0.1`

**lihuen@lihuen:~\$ dig www.redes.unlp.edu.ar**

```
; <<>> DiG 9.5.1-P1 <<>> www.redes.unlp.edu.ar
```

```
:: global options: printcmd
```

```
:: Got answer:
```

```
:: ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 2158
```

```
:: flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 1
```

```
:: QUESTION SECTION:
```

```
;www.redes.unlp.edu.ar.      IN      A
```

```

;; ANSWER SECTION:
www.redes.unlp.edu.ar. 604800 IN A 127.0.0.1

;; AUTHORITY SECTION:
redes.unlp.edu.ar. 604800 IN NS ns.redes.unlp.edu.ar.

;; ADDITIONAL SECTION:
ns.redes.unlp.edu.ar. 604800 IN A 127.0.0.1

;; Query time: 0 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: Thu Apr 16 09:27:48 2009
;; MSG SIZE rcvd: 88

```

a. La IP del host `www.redes.unlp.edu.ar` 127.0.0.1

b. La IP del servidor de nombres para el dominio `redes.unlp.edu.ar`  
127.0.0.1

c. El servidor de correo para `redes.unlp.edu.ar`

```

$ host -t MX redes.unlp.edu.ar
redes.unlp.edu.ar mail is handled by 5 mail.redes.unlp.edu.ar.

```

Los mismos resultados con:

```

$ nslookup -type=MX redes.unlp.edu.ar
$ dig -t MX redes.unlp.edu.ar +nocomments +nostats

```

14. ¿Qué función cumple el archivo `/etc/hosts` (Linux/Unix) y el `\WINDOWS\system32\drivers\etc\hosts` (Windows)?

a. Pruebe editar el archivo (en cualquier Windows/Linux) agregando la línea: `127.0.0.1 www.prueba.com`

b. Luego pruebe ejecutar desde una consola de línea de comandos:  
`ping www.prueba.com`

El archivo `/etc/hosts` es una tabla de búsqueda estática de hostnames, con la siguiente estructura:

```

IP_address canonical_hostname [aliases...]

```

La utilidad de esta tabla está superada por los servidores DNS -como el *BIND*, pero aún se suele utilizar para mantener información sobre los hosts más importantes de la red local (por si el DNS queda fuera de servicio temporalmente) y para redes pequeñas aisladas de Internet o que no varían en el tiempo. Otra opción es relacionar la IP local o una errónea a sitios que se deseen bloquear (por ejemplo, de publicidad o monitorización).

```
$ ping www.prueba.com
```

```
PING www.prueba.com (127.0.0.1) 56(84) bytes of data.  
64 bytes from lihuen (127.0.0.1): icmp_seq=1 ttl=64 time=0.019 ms  
64 bytes from lihuen (127.0.0.1): icmp_seq=2 ttl=64 time=0.004 ms ...
```

**15. Abra el programa Wireshark para comenzar a capturar tráfico, una vez abierto realice una consulta DNS por el registro MX de redes.unlp.edu.ar y luego otra preguntando por los registros NS correspondiente al dominio redes.unlp.edu.ar, utilice dig. Analice la información proporcionada por dig y compárelo con la captura.**

```
lihuen@lihuen: ~$ dig -t NS redes.unlp.edu.ar +nocomments
```

```
; <<>> DiG 9.5.1-P1 <<>> -t NS redes.unlp.edu.ar +nocomments  
;; global options: printcmd  
;redes.unlp.edu.ar.          IN      NS  
redes.unlp.edu.ar.          604800 IN      NS      ns.redes.unlp.edu.ar.  
ns.redes.unlp.edu.ar.       604800 IN      A        127.0.0.1 alu  
;; Query time: 0 msec  
;; SERVER: 127.0.0.1#53(127.0.0.1)  
;; WHEN: Thu Apr 23 00:51:43 2009  
;; MSG SIZE rcvd: 68
```

```
lihuen@lihuen: ~$ dig -t MX redes.unlp.edu.ar +nocomments
```

```
; <<>> DiG 9.5.1-P1 <<>> -t MX redes.unlp.edu.ar +nocomments  
;; global options: printcmd  
;redes.unlp.edu.ar.          IN      MX  
redes.unlp.edu.ar.          604800 IN      MX      5 mail.redes.unlp.edu.ar.  
redes.unlp.edu.ar.          604800 IN      NS      ns.redes.unlp.edu.ar.  
mail.redes.unlp.edu.ar.     604800 IN      A        127.0.0.1  
ns.redes.unlp.edu.ar.       604800 IN      A        127.0.0.1  
;; Query time: 0 msec  
;; SERVER: 127.0.0.1#53(127.0.0.1)  
;; WHEN: Thu Apr 23 00:51:49 2009  
;; MSG SIZE rcvd: 105
```

La información de la captura es equivalente, aunque más detallada, incluyendo todos los campos del mensaje, y los pertenecientes a los protocolos de capas inferiores.

**HTTP** Requerimientos: Debe usar el Live CD de Lihuen GNU/Linux provisto

**16. Defina cada una de las siguientes entidades: Navegador, Servidor WEB, Página WEB, HTTP y URL. ¿Cómo participa cada uno de ellas en la comunicación cliente WEB – servidor WEB?**

- **Navegador:** User agent, cliente de la Web. Realiza solicitudes a los servidores, despliega los resultados y permite la interacción del usuario.

- **Servidor WEB:** Implementa la parte servidor de HTTP y almacena objetos Web, referenciables con una URL. Recibe solicitudes de clientes y devuelve los objetos (archivos) adecuados.

- **Página WEB:** Documento formado por uno o varios objetos (archivos); mínimamente el documento HTML base. Cada objeto se puede referenciar por su URL.

- **HTTP:** HyperText Transfer Protocol, el protocolo de capa de aplicación de soporte de la Web; define los mensajes -solicitudes y respuestas- que pueden enviarse, su formato y orden. Se implementa en dos partes: la cliente -navegador- y el servidor.

- **URL:** Uniform Resource Locator. Una secuencia de caracteres que permite identificar un recurso, como un documento o una imagen en Internet, permitiendo localizar los objetos de una página Web y la definición de enlaces entre páginas. En su forma básica se compone del **nombre del host** que posee el recurso y el **path del recurso** en dicho host. En HTTP, un URL completo puede alcanzar la forma:

`http_o_https://nombre_o_IP_del_dominio:puerto/ruta_al_objeto?param1=valor1&param2=valor2#enlace`

## 17.¿Qué son y en qué se diferencian HTML y HTTP?

**HTTP** es el protocolo que define cómo los clientes Web solicitan páginas Web a los servidores -HTTP request-, y cómo esos servidores transfieren las páginas a los clientes - HTTP response-.

**HTML** (HyperText Markup Language) es el lenguaje usado para la definición de páginas Web.

18.Clasifique las siguientes aplicaciones WEB según sean clientes o servidores. Luego, para cada uno indique sus características principales:

- a. Microsoft Internet Explorer. Cliente.
- b. MS Internet Information Server. Servidor.
- c. Safari. Cliente.
- d. Tomcat. Servidor.
- e. FireFox. Cliente.
- f. Netscape fasttrac. Servidor.

g. Netscape Communicator. Cliente.

h. Apache. Servidor.

i. Konqueror. Cliente.

j. Opera. Cliente.

## 19. Explique la diferencia entre la versión HTTP 1.0 y la versión HTTP 1.1

Ambas versiones utilizan TCP para la capa de transporte y son *stateless*. La diferencia es que HTTP 1.0 utiliza conexiones no persistentes, y HTTP 1.1 puede manejar tanto persistentes como no persistentes.

En una **conexión no persistente**, el server cierra la conexión tras enviar el objeto solicitado (en realidad la conexión se cierra una vez que TCP asegura la transmisión). Así, cada conexión establecida sólo permite solicitar un objeto. Suelen utilizarse varias conexiones paralelas para mejorar los tiempos.

En una **conexión persistente**, el server mantiene la conexión abierta a la espera de nuevas solicitudes. Típicamente el server cierra la conexión al permanecer inactiva durante un **timeout** configurable. A su vez, las conexiones persistentes pueden ser **sin pipelining** (el cliente envía un request sólo si el anterior fue satisfecho) o **con pipelining** (por defecto en HTTP/1.1, el cliente envía un request ni bien encuentra la referencia al objeto, sin importar el estado de los requests previos).

## 20. Utilizando el Live CD, abra un navegador (Iceweasel) e ingrese a la URL: [www.redes.unlp.edu.ar/](http://www.redes.unlp.edu.ar/)

a. Ingrese al link en la sección "Capa de Aplicación" llamado "Protocolos HTTP". En la página mostrada se visualizan dos nuevos links llamados: Protocolo HTTP/1.1 y Protocolo HTTP/1.0. Antes de ingresar a estos links continúe con el siguiente punto.

b. Utilizando el analizador de paquetes Wireshark capture los paquetes enviados y recibidos al presionar sobre el link.

- Nota 1: Capture los paquetes utilizando la interfaz de loopback lo.

- Nota 2: Si el analizador no captura paquetes, utilice Ctrl+F5 en el navegador para forzar la petición HTTP evitando el uso de cache del navegador.

- Nota 3: Ya realizada la captura, la lectura de la captura de paquetes se simplifica utilizando botón derecho sobre un paquete HTTP



perteneciente al flujo capturado y seleccionando la opción Follow TCP Stream.

c. ¿Qué diferencias observa de ambos protocolos?

**HTTP 1.0** - El cliente hace un request con HTTP 1.1, el servidor responde con HTTP 1.0. En los encabezados del HTTP response figura *Connection: close*. Hay una conexión para la página y otra para la imagen. Primer conexión:

**GET /http/HTTP-1.0/ HTTP/1.1**

Host: www.redes.unlp.edu.ar

User-Agent: Mozilla/5.0 (X11; U; Linux i686; es-AR; rv:1.9.0.6) Gecko/2009020409

Iceweasel/3.0.6 (Debian-3.0.6-1)

Accept: text/html,application/xhtml+xml,application/xml;q=0.9,\*/\*;q=0.8

Accept-Language: es-ar,es;q=0.8,en-us;q=0.5,en;q=0.3

Accept-Encoding: gzip,deflate

Accept-Charset: ISO-8859-1,utf-8;q=0.7,\*;q=0.7

**Keep-Alive: 300**

**Connection: keep-alive**

Referer: http://www.redes.unlp.edu.ar/http/protocolos.html

**HTTP/1.0 200 OK**

Date: Thu, 23 Apr 2009 03:33:21 GMT

Server: Apache/2.2.9 (Debian) PHP/5.2.6-1+lenny2 with Suhosin-Patch

Last-Modified: Thu, 09 Apr 2009 00:05:31 GMT

ETag: "6b23-265-46714019e48c0"-gzip

Accept-Ranges: bytes

Vary: Accept-Encoding

Content-Encoding: gzip

Content-Length: 357

**Connection: close**

Content-Type: text/html

.....eRKn.0..s.Q.4.P..

.TH...=.....c.;...3.....yo..O..N

...

**HTTP 1.1** - El cliente hace un request con HTTP 1.1, el servidor responde también con HTTP 1.1. En los encabezados del HTTP response figura *Connection: keep-alive*. Hay una única conexión para todo el contenido:

**GET /http/HTTP-1.1/ HTTP/1.1**

Host: www.redes.unlp.edu.ar

User-Agent: Mozilla/5.0 (X11; U; Linux i686; es-AR; rv:1.9.0.6) Gecko/2009020409

Iceweasel/3.0.6 (Debian-3.0.6-1)

Accept: text/html,application/xhtml+xml,application/xml;q=0.9,\*/\*;q=0.8

Accept-Language: es-ar,es;q=0.8,en-us;q=0.5,en;q=0.3

Accept-Encoding: gzip,deflate

Accept-Charset: ISO-8859-1,utf-8;q=0.7,\*;q=0.7

**Keep-Alive: 300**

**Connection: keep-alive**

Referer: http://www.redes.unlp.edu.ar/http/protocolos.html

**HTTP/1.1 200 OK**

Date: Thu, 23 Apr 2009 03:35:17 GMT

Server: Apache/2.2.9 (Debian) PHP/5.2.6-1+lenny2 with Suhosin-Patch

Last-Modified: Thu, 09 Apr 2009 00:05:31 GMT

ETag: "6b30-1d8-46714019e48c0"-gzip

Accept-Ranges: bytes

Vary: Accept-Encoding

Content-Encoding: gzip

Content-Length: 300

**Keep-Alive: timeout=15, max=100**

**Connection: Keep-Alive**

Content-Type: text/html

.....eQKN.0...).WH.u...v...`..f&MB>e..s..@.TT.o.;~.~v;.Qv.v  
.....^xL.j.....N..#Ld...[...][\$P.....p..D.'....Ty.A.Zk.....<...B!8.L..R|!./f.....=...,rmcK0.....T.....k.o  
2..[.g.....g.....i..8....(.H.9o.h'..^B'...'2O2.U.....y.....z..c...}V5.M}...".\${6.%R.Q}....a.=...f.|.t....  
.....

**GET /http/HTTP-1.1/13532-tuxkiller03green.png HTTP/1.1**

Host: www.redes.unlp.edu.ar

User-Agent: Mozilla/5.0 (X11; U; Linux i686; es-AR; rv:1.9.0.6) Gecko/2009020409

Iceweasel/3.0.6 (Debian-3.0.6-1)

Accept: image/png,image/\*;q=0.8,\*/\*;q=0.5

Accept-Language: es-ar,es;q=0.8,en-us;q=0.5,en;q=0.3

Accept-Encoding: gzip,deflate

Accept-Charset: ISO-8859-1,utf-8;q=0.7,\*;q=0.7

**Keep-Alive: 300**

**Connection: keep-alive**

Referer: http://www.redes.unlp.edu.ar/http/HTTP-1.1/

**HTTP/1.1 200 OK**

Date: Thu, 23 Apr 2009 03:35:17 GMT

Server: Apache/2.2.9 (Debian) PHP/5.2.6-1+lenny2 with Suhosin-Patch

Last-Modified: Thu, 09 Apr 2009 00:05:31 GMT

ETag: "6b31-6028c-46714019e48c0"

Accept-Ranges: bytes

Content-Length: 393868

**Keep-Alive: timeout=15, max=99**

**Connection: Keep-Alive**

Content-Type: image/png

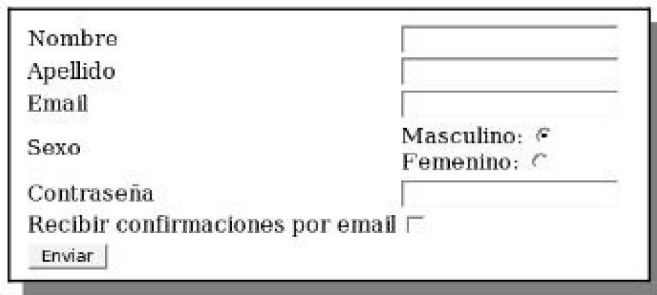
.PNG

.

... IHDR.....

21.Utilizando el Live CD, abra un navegador en ingrese a la URL:  
[www.redes.unlp.edu.ar/](http://www.redes.unlp.edu.ar/)

a. Ingrese al link en la sección "Capa de Aplicación" llamado "Métodos HTTP". En la página mostrada se visualizan dos nuevos links llamados: Método GET y Método POST. Ambos muestran un formulario como el siguiente:



Nombre   
 Apellido   
 Email   
 Sexo: Masculino: ☒ Femenino: ☐  
 Contraseña   
 Recibir confirmaciones por email ☐  
 Enviar



## b. Analice el código HTML

Las diferencias encontradas fueron:

GET: `<form method="GET" action="metodos-lectura-valores.php">`

POST: `<form method="POST" action="metodos-lectura-valores.php">`

## c. Utilizando el analizador de paquetes Wireshark capture los paquetes enviados y recibidos al presionar el botón Enviar.

- Nota 1: capture los paquetes utilizando la interfaz de loopback lo.
- Nota 2: si el analizador no captura paquetes, utilice Ctrl+F5 en el navegador para forzar la petición HTTP evitando el uso de cache del navegador
- Nota 3: ya realizada la captura, la lectura de la misma se simplifica utilizando botón derecho sobre un paquete HTTP perteneciente al flujo capturado y seleccionando la opción Follow TCP Stream.

## d. ¿Qué diferencias detectó en los mensajes enviados por el cliente? ¿Observó alguna diferencia en el browser si se utiliza un mensaje u otro?

GET --> Pasa los datos por URL -visibles en el browser-

POST --> Solo pasa el URL de la pagina y los datos van por Line-based text data: application/x-www-form-urlencoded -cuerpo del mensaje-

## 22. Relacione DNS con HTTP

**SMTP, POP e IMAP** Requerimientos: Debe usar el Live CD de Lihuen GNU/Linux provisto

**23. ¿Qué protocolos se utilizan para el envío y la recepción de mails. Enumere y explique sus características.**

**SMTP:** Envío de emails.

**POP:** Recepción de emails. Por defecto, borra los emails tras bajarlos del servidor siendo imposible consultarlos desde otro equipo.

**IMAP:** Recepción de emails. Mantiene copias de los emails en el servidor, permitiendo organizarlos en carpetas.

**24. Clasifique las siguientes aplicaciones de correo:**

a. Microsoft Exchange

b. Microsoft Outlook

c. Postfix

d. Thunderbird

e. Qmail

f. Eudora

g. Courier

h. Cyrus

**25. Utilizando el Live CD, abra el cliente de correo (Icedove) y configure:**

a. Una cuenta de correo POP

1. Cuenta de correo: `alumnopop@redes.unlp.edu.ar`

2. Nombre de usuario: `alumnopop`

3. Contraseña: `alumnopoppass`

4. Servidor de correo POP: `mail.redes.unlp.edu.ar`

5. Servidor de correo saliente (SMTP): `mail.redes.unlp.edu.ar`

b. Una cuenta de correo IMAP

1. Cuenta de correo: `alumnoimap@redes.unlp.edu.ar`

2. Nombre de usuario: alumnoimap

3. Contraseña: alumnoimappass

4. Servidor de correo IMAP: mail.redes.unlp.edu.ar

c. Envíe un email desde el cliente de una cuenta a la otra y luego chequee el correo de ambas cuentas.

Enviando mails (Analizando SMTP):

d. Reitere el proceso de envío, esta vez capturando los paquetes de protocolo SMTP utilizando Wireshark. Analice el intercambio del protocolo entre el cliente y el servidor, identificando cada comando y su correspondiente respuesta, realice un gráfico que muestre este intercambio.

e. Desde una terminal, utilice los comandos del protocolo SMTP observados en el punto anterior, para enviar un mail al servidor en forma manual.

- Nota: Para conectarse al servidor deberá utilizar el comando: **telnet mail.redes.unlp.edu.ar 25** .

- Nota2: Verifique que haya recibido el correo en la cuenta a la que haya enviado el correo desde la consola.

**\$ telnet mail.redes.unlp.edu.ar 25**

Trying 127.0.0.1...

Connected to mail.redes.unlp.edu.ar.

Escape character is '^['.

220 mail.redes.unlp.edu.ar ESMTP Postfix (Lihuen-3/GNU)

**EHLO 127.0.0.1**

250-mail.redes.unlp.edu.ar

250-PIPELINING

250-SIZE 10240000

250-VERFY

250-ETRN

250-ENHANCEDSTATUSCODES

250-8BITMIME

250 DSN

**MAIL FROM:<alumnoimap@redes.unlp.edu.ar>**

250 2.1.0 Ok

**RCPT TO:<alumnopop@redes.unlp.edu.ar>**

250 2.1.5 Ok

**DATA**

354 End data with <CR><LF>.<CR><LF>

From: lucas@lucas.com

To: ivan@ivan.com

Subject: hola!

Hola como vas???

Lo que esta en verde es del punto f. Te veo!

.

250 2.0.0 Ok: queued as 2AC863429

QUIT

221 2.0.0 Bye

Connection closed by foreign host.

f. Repita este procedimiento utilizando una cuenta diferente de mail para el campo From:, luego verifique que el correo recibido por el destinatario tenga la cuenta ficticia.

Se recibe el mail, y tanto en el remitente como en el destinatario aparecen las direcciones falsas -el *user agent* utiliza la información del contenido del mensaje-.

Recibiendo mails (Analizando POP e IMAP):

g. Vuelva a enviar un correo a `alumnopop@redes.unlp.edu.ar` utilizando el cliente de correo configurado. Comience la captura con Wireshark y chequee la cuenta de correo de `alumnopop` para capturar tráfico del protocolo POP. Analice el intercambio del protocolo entre el cliente y el servidor, identificando cada comando y su correspondiente respuesta, realice un gráfico que muestre este intercambio.

h. Vuelva a enviar un mensaje a `alumnopop@redes.unlp.edu.ar` y sin chequear los mensajes con el cliente, Comience la captura y desde una terminal, utilice los comandos del protocolo POP observados en el punto anterior para consultar los mails del usuario `alumnopop`. Lea el contenido del primer mail desde la consola utilizando telnet.

- Nota: Para conectarse al servidor deberá utilizar el comando: **telnet mail.redes.unlp.edu.ar 110**.

```
$ telnet mail.redes.unlp.edu.ar 110
Trying 127.0.0.1...
Connected to mail.redes.unlp.edu.ar.
Escape character is '^]'.
+OK Hello there.
CAPA
+OK Here's what I can do:
TOP
USER
LOGIN-DELAY 10
PIPELINING
UIDL
IMPLEMENTATION Courier Mail Server
.
USER alumnopop
+OK Password required.
```

**PASS alumnopoppass**

+OK logged in.

**LIST**

+OK POP3 clients that break here, they violate STD53.

1 722

.

**RETR 1**

+OK 722 octets follow.

Return-Path: <alumnoimap@redes.unlp.edu.ar>

X-Original-To: alumnopop@redes.unlp.edu.ar

Delivered-To: alumnopop@redes.unlp.edu.ar

Received: from [127.0.0.1] (lihuen [127.0.0.1])

by mail.redes.unlp.edu.ar (Postfix) with ESMTP id 73CB6344E

for <alumnopop@redes.unlp.edu.ar>; Mon, 27 Apr 2009

18:35:40 -0300 (ART)

Message-ID: <49F6252C.2050506@redes.unlp.edu.ar>

Date: Mon, 27 Apr 2009 18:35:40 -0300

From: alumnoimap <alumnoimap@redes.unlp.edu.ar>

User-Agent: Mozilla-Thunderbird 2.0.0.19 (X11/20090103)

MIME-Version: 1.0

To: alumnopop@redes.unlp.edu.ar

Subject: asuntooooooooooooooooooooo

Content-Type: text/plain; charset=ISO-8859-1; format=flowed

Content-Transfer-Encoding: 7bit

hola

nos vemos!

.

**DELE 1**

+OK Deleted.

**QUIT**

+OK Bye-bye.

Connection closed by foreign host.

**i. Cierre el telnet y comience una nueva captura. Chequee nuevamente el mail con el cliente configurado. ¿Qué diferencia encuentra entre ambas capturas?**

No baja los mails --ya se bajaron y borraron por telnet.

**j. Con el rol de administrador del sistema (root), ejecute el cliente de correos. Para esto, abra una consola de comandos y ejecute: sudo icedove cuando pregunte por la password introduzca "lihuen"**

De esta forma, ud. iniciará el cliente de correo con el perfil del superusuario (diferente del usuario con el que configuró las cuentas antes mencionadas). Recuerde que la contraseña del usuario root es lihuen.

Luego configure las cuentas pop e imap como se describió anteriormente. ¿Qué diferencias observa entre el servicio ofrecido por POP vs el ofrecido por IMAP?

Al enviar recibir un mail con una sesión el user agent lo baja. Al pasar a la otra sesión y revisar el mail, si se utilizó POP no se puede ver el mensaje de nuevo -se borró del servidor- pero si se utilizó IMAP sí -se mantienen los mails en el servidor aunque ya está marcado como leído. Sin embargo, si utilizando IMAP se baja un mail y se lo elimina, al contactar al servidor se elimina también la copia y el otro user agent no puede verlo.

## 26. Relacione DNS con SMTP. Describa el proceso completo en el envío de un correo desde pepe@yahoo.com a jose@hotmail.com.

DNS básicamente sirve de soporte para el resto de los protocolos de aplicación. Al escribir un mail, el user agent busca la IP del servidor de correos del dominio que ofrece la cuenta (servidor SMTP de yahoo.com) para realizar la conexión SMTP y enviar el correo. El servidor de yahoo, utiliza luego DNS para averiguar la IP del servidor de correo de hotmail.com, establece la conexión y envía el email. El destinatario, eventualmente revisa su correo con su user agent, que utilizará DNS para averiguar la IP del servidor POP3 o IMAP.

**FTP** Requerimientos: Debe usar el Live CD de Lihuen GNU/Linux provisto

## 27.¿Cuál es el objetivo del protocolo FTP?

FTP (File Transfer Protocol) permite transferir archivos desde y hasta un host remoto, con autenticación por medio de usuario y contraseña.

## 28.Explique los siguientes comandos FTP: open, user, ls, cd, prompt, mget, mput, get, put, hash, bye, lcd y por último el uso del símbolo !.

<b>open [dir_host]</b>	abrir conexión con el sistema dir_host
<b>user username</b>	identificación en máquina distante
<b>ls [dir]</b>	listado del contenido de un directorio -el actual si no se especifica-
<b>cd [dir]</b>	cambio de directorio en la máquina distante
<b>prompt</b>	cambia de interactivo a no interactivo, y viceversa (para que solicite confirmación para ejecutar los comandos o no)
<b>put [ref_local [ref_distante]]</b>	el archivo ref_local se transmite con el nombre ref_distante. Si se omite ref_distante, se transmite con el mismo nombre
<b>mput ref_local ...</b>	similar al anterior, pero para varios ficheros
<b>get [ref_distante [ref_local ]]</b>	el archivo ref_distante se copia con el nombre ref_local a la máquina local. Si se omite ref_local, se utiliza el mismo nombre.
<b>mget ref_distante ...</b>	similar al anterior, pero para varios ficheros
<b>hash</b>	activa la impresión de caracteres # a medida que se transfieren archivos, a modo de barra de progreso
<b>bye</b>	(o quit) finaliza una conexión FTP y la sesión de trabajo con el programa cliente
<b>lcd [dir]</b>	cambio de directorio o vuelta al directorio base en la máquina local
<b>!comando</b>	se ejecuta el comando en la máquina local
<b>!</b>	sale a la línea de comandos temporalmente sin cerrar la conexión; puede utilizarse exit para volver a ftp



29.Utilizando el Live CD conéctese al servidor ftp utilizando el comando `ftp` <ftp.redes.unlp.edu.ar> utilizando los siguientes datos:

Nombre de usuario: lihuen

Password: lihuen

a. Transfiera un archivo cualquiera al servidor.

b. Transfiera un archivo desde el servidor a la máquina local

```
$ ftp ftp.redes.unlp.edu.ar
Connected to www.redes.unlp.edu.ar.
220----- Welcome to Pure-FTPd [privsep] [TLS] -----
220-You are user number 1 of 50 allowed.
220-Local time is now 18:44. Server port: 21.
220-This is a private system - No anonymous login
220-IPv6 connections are also welcome on this server.
220 You will be disconnected after 15 minutes of inactivity.
Name (ftp.redes.unlp.edu.ar:lihuen): lihuen
331 User lihuen OK. Password required
Password: *****
230-User lihuen has group access to:  lihuen  umlnetwo bind
vnuml
230- uml-net  ssh          netdev  powerdev users  plugdev  video
src
230- audio    sudo        floppy   cdrom    dialout  lp
230 OK. Current directory is /home/lihuen
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> ls
200 PORT command successful
150 Connecting to port 42229
drwxr-xr-x   2 lihuen  lihuen           80 Apr 27 18:30 Desktop
drwx-----   5 lihuen  lihuen           36 Mar 20 20:51 Maildir
-rw-r--r--   1 lihuen  lihuen       12353 Apr 14 20:14 captura.pcap
226-Options: -l
226 3 matches total
ftp> cd ..
250 OK. Current directory is /home
ftp> ls
200 PORT command successful
150 Connecting to port 45942
drwxr-x---   4 alumnoim alumnoim       78 Mar 20 20:51 alumnoimap
drwxr-x---   5 alumnopo alumnopo       90 Apr  8 21:08 alumnopop
drwxr-x---  20 lihuen  lihuen        480 Apr 27 18:13 lihuen
226-Options: -l
226 3 matches total
ftp> cd ~
250 OK. Current directory is /home/lihuen
ftp> get captura.pcap
local: captura.pcap remote: captura.pcap
200 PORT command successful
150-Connecting to port 33683
```

```

150-The computer is your friend. Trust the computer
150 12.1 kbytes to download
226-File successfully transferred
226 0.004 seconds (measured here), 3.30 Mbytes per second
12353 bytes received in 0.00 secs (113806.4 kB/s)
ftp> captura.pcap1get captura.pcap
local: captura.pcap1 remote: captura.pcap
200 PORT command successful
150-Connecting to port 42255
150 12.1 kbytes to download
226-File successfully transferred
226 0.000 seconds (measured here), 161.48 Mbytes per second
12353 bytes received in 0.00 secs (163019.9 kB/s)
ftp> cd Desktop/
250 OK. Current directory is /home/lihuen/Desktop
ftp> ls
200 PORT command successful
150 Connecting to port 34856
-rw-r--r--      1 lihuen  lihuen           318 Apr 14 20:50
Practical_Captura-Redes.desktop
-rw-r--r--      1 lihuen  lihuen          3298 Apr 27 18:13
captura_mail_envio
lrwxrwxrwx      1 lihuen  lihuen           19 Apr 13 20:35 index.html -
> /var/www/index.html
-rw-r--r--      1 lihuen  lihuen          3351 Apr 27 18:30
mail_recupero_pop
226-Options: -l
226 4 matches total
ftp> put captura.pcap copiaremota.pcap
local: captura.pcap remote: copiaremota.pcap
200 PORT command successful
150 Connecting to port 53047
226-File successfully transferred
226 0.000 seconds (measured here), 62.00 Mbytes per second
12353 bytes sent in 0.00 secs (101373.8 kB/s)
ftp> quit
221-Goodbye. You uploaded 13 and downloaded 13 kbytes.
221 Logout.

```

## Ejercicios Evaluables

1- Utilizando la herramienta dig o nslookup simule con consultas iterativas la consulta recursiva por el nombre de dominio WWW.XXX.YYY.ZZZ Debe poder describir la rama del árbol de servidores de DNS dentro del esquema de servidores que tuvo que consultar para llegar a la respuesta del tipo authoritative para ese dominio.

- Nota: El dominio será proporcionado por el ayudante que tenga asignado.

- Nota2: Va a necesitar que la máquina que utilice tenga acceso a Internet.

2- Analizar la captura de tráfico llamada evaluable2.pcap que se encuentra en el directorio personal del usuario lihuen. La misma fué generada en un servidor de correos. El alumno deberá poder responder:

¿Quién es el usuario que revisa su casilla de correos?

--> Usuario lihuen, contraseña lihuen.

¿A quién le escribe un correo?

--> Primero se envía un mail a sí mismo (lihuen@redes.unlp.edu.ar), y despues de consultar su correo por POP le escribe a webmaster@linti.unlp.edu.ar.

¿A qué se debe la aparición de los distintos protocolos de capa de aplicación hallados en la misma? Es necesario que explique porque aparecen en cada caso.

Primeramente utiliza SMTP para enviarse un correo a si mismo. Luego, POP para revisar los correos que tiene por leer. Utiliza nuevamente SMTP para escribir a webmaster@linti.unlp.edu.ar, y la última parte SMTP corresponde al mail server de redes.unlp.edu.ar enviando el email al mail server de linti.unlp.edu.ar -luego de consultar el MX de ese dominio para obtener su IP-.

El alumno deberá explicar estas cuestiones al ayudante evaluador.

## Capa De Transporte

### 1. ¿Cuál es la función de la capa de transporte?

La función de la capa de **Transporte** es proveer de una comunicación lógica entre *procesos de aplicación corriendo en diferentes hosts*, permitiendo el envío de *segmentos* de datos como si estuviesen conectados directamente, sin intermediarios ni preocupaciones por la infraestructura de envío de mensajes. Se implementa en los hosts terminales, y se ubica por sobre la capa de **Red** (que se ocupa de la comunicación lógica, pero entre hosts remotos en vez de entre procesos).

### 2. Comparación TCP vs UDP.

La capa de Transporte ofrece dos protocolos, cada uno con distintos servicios: **TCP** y **UDP**.

Por estar contruídos sobre el único protocolo que ofrece la capa de Red, *IP*, algunos servicios no pueden ser ofrecidos por ninguno de ellos: garantías de retardo ni de ancho de banda -básicamente porque IP utiliza *conmutación de paquetes* y no de *circuitos*. Sin embargo, otras características no soportadas por *IP*, como la transmisión confiable, sí pueden ser ofrecidas por la capa de Transporte.

Característica	Definición	TCP	UDP
<b>Confiabilidad</b>	Asegurar la entrega de todos los paquetes enviados, permitiendo mantener en el receptor el orden en que le fueron enviados (utilizando confirmación de recepción, timeout, reenvío de paquetes y números de secuencia) y detectar errores en la recepción (mediante <i>checksum</i> ).	<b>SI</b> (utiliza establecimiento de conexión de 3 pasos)	<b>NO</b> (sin conexión, sólo ofrece el <i>best effort</i> de IP) (*)
<b>Multiplexación</b>	Extender la comunicación host-host que ofrece IP permitiendo diferenciar los procesos en el receptor y en el emisor, de modo que varios procesos en un host se comuniquen con varios procesos en otros hosts, sin confundir los paquetes (utilizando direcciones IP y números de puerto que identifican a los prarocesos).	<b>SI</b> Socket: ( <i>IP Orig, Port Orig, IP Dest, Port Dest</i> )	<b>SI</b> Socket: ( <i>IP Dest, Port Dest</i> ) (**)
<b>Control de Congestión</b>	Controlar la velocidad de transmisión del emisor para no sobrecargar la red.	<b>SI</b>	<b>NO</b>
<b>Control de Flujo</b>	Controlar la velocidad de transmisión del emisor para no sobrecargar al receptor (según el tamaño de su buffer)	<b>SI</b>	<b>NO</b>
<b>Utilización de puertos</b>	Los ports destino y origen son precisos tanto en TCP como en UDP por fines de multiplexación/demultiplexación, aunque UDP	<b>Port Origen y Port Destino</b>	<b>Port Origen y Port</b>

sólo precisa el port destino para definir un socket (al ser sin conexión se envían segmentos individuales, mientras que TCP divide un mensaje en múltiples segmentos y precisa reconocer el proceso emisor para recomponer el mensaje original)

## Destino

(\*) Es posible un **Transporte Confiable Sobre UDP** -TCsU-, pero dependiente de la aplicación -será esta quien determine cómo recuperarse ante fallos-. Del mismo modo que TCP ofrece transmisión confiable sobre IP, una aplicación puede lograr transmisión confiable sobre UDP. Si bien podría utilizarse TCP en estos casos y simplificar la aplicación, el uso de TCsU ofrece a la aplicación una ventaja: UDP no posee **control de congestión**, de modo que puede transmitir a cualquier tasa de transferencia aunque perjudique el desempeño total de la red -con TCP, se comienza enviando datos a una tasa baja que aumenta gradualmente; por cada segmento no confirmado se baja la tasa por detección de congestión y en conjunto con IP se logra un control de congestión preventivo.

(\*\*) UDP: Segmentos con distinto IP y port de origen pueden ser dirigidos a un mismo socket, dado que sólo lo determina el IP y port destino. El port origen provee una *dirección de retorno*, por si el receptor desea responder de algún modo al emisor, pero esto dependerá del protocolo de aplicación, no es propio de UDP -TCP en cambio, envía siempre un paquete de acuse de recibo o *ACK*-.

### 3. ¿Cuál es el objetivo del uso de puertos en el modelo TCP/IP?

El uso de puertos permite la **multiplexación** de los paquetes que Transporte recibe de los distintos procesos y pasa a la capa de Red en el emisor, y la **demultiplexación** de los segmentos recibidos desde la capa de Red en paquetes que se envían a los procesos adecuados en el receptor.

Si un terminal tuviese dos procesos haciendo solicitudes a un mismo servidor Web, ambos se comunicarían al port 80 -utilizado por el servidor Web en el host destino-. El port destino permite a la capa de Transporte del receptor entregar los paquetes al servidor Web y no a otro proceso que esté corriendo en ese host. Del mismo modo, cada proceso del emisor tendrá asociado un port de origen diferente -comúnmente aleatorio-, permitiendo al servidor Web encaminar sus respuestas en dirección contraria a un proceso específico del host que realizó las solicitudes.

En el caso que dos clientes escojan un mismo port origen, el servidor podrá diferenciar a cada proceso utilizando la IP de origen, en el datagrama de capa de Red. Así, la tripla (*port origen, port destino, IP origen*) permite al host receptor encaminar los datos al socket correspondiente.

### 4. Describa la estructura del segmento TCP y UDP

- **Segmento TCP:** #port fuente, #port destino, # de secuencia de los datos enviados, # de confirmación de la última secuencia recibida, tamaño del header, un campo actualmente no utilizado, flags, tamaño de la ventana en el receptor, checksum, puntero a la posición donde comienzan los datos urgentes, campo de opciones de

longitud variable y los datos en sí que se transmiten -opcionales, por si sólo se envía información de control-.

- Los # de secuencia y confirmación se refieren a la posición de los bytes de los datos enviados, no a la numeración de los segmentos. Un mismo segmento TCP puede indicar cuáles son los datos que se envían en él, y cuáles fueron los últimos datos recibidos -por ser la conexión *full duplex*-, permitiendo ordenar los segmentos que se reciben.
- El tamaño del header es variable debido al campo *opciones*, por esta razón se precisa indicar el tamaño en un campo especial.
- El campo flags incluye 6 flags distintos: URGent data -indica que se envían datos urgentes-, ACKnowledge -confirmación-, PUSH -entregar datos al proceso en forma inmediata-, RST -negación de pedido de conexión-, SYN -utilizado para establecimiento de la conexión en el **three-way handshake**- y FIN -utilizado para el cierre de conexión-. En la práctica, URG y PUSH no se utilizan.
- El tamaño de la ventana del receptor se refiere a la cantidad de bytes que el receptor está dispuesto a aceptar, permitiendo el control de flujo.
- El checksum se calcula sobre los datos y el encabezado.
- El puntero a los datos urgentes no se utiliza en la práctica.
- El campo de opciones permite enviar *timestamps*, información sobre la implementación específica de TCP, negociar el tamaño máximo de segmento...
- **Segmento UDP:** #port fuente, #port destino, largo de los datos, checksum y los datos en sí que se transmiten.

## **5. La PDU de la capa de transporte es el segmento. Sin embargo, en algunos contextos suele utilizarse el término Datagrama, indique cuándo.**

Un datagrama es una cabecera + datos, tal que la cabecera ofrece la información necesaria para que los datos puedan ser encaminados y alcancen un destino.

Se suele hablar de *Datagrama* en referencia a los segmentos en el protocolo *UDP* -*Protocolo de Datagrama de Usuario*-. Sin embargo, como *datagrama* es el nombre formal de los *PDU* de la capa de *Red*-protocolo IP- es preferible evitar esta denominación.

## **6. Utilizando el Live CD. Use el analizador de paquetes Wireshark para capturar los paquetes enviados y recibidos en cada uno de los siguientes casos. Arranque la captura antes de realizar cada una de las acciones indicadas:**

- A) Abra un navegador e ingrese a la URL: [www.redes.unlp.edu.ar](http://www.redes.unlp.edu.ar)
  - i. Analice la secuencia de mensajes que ocurren antes de que se intercambien mensajes HTTP. ¿Cuál es el objetivo? ¿Qué flags se utilizan en cada caso? ¿Qué indica cada uno?

Tras las consultas DNS para obtener la IP de [www.redes.unlp.edu.ar](http://www.redes.unlp.edu.ar), se envían tres segmentos TCP para establecer la conexión -**three-way handshake**-.

- . Ciente#35150 a Servidor#80, flag SYN: Es el pedido de conexión del cliente al servidor. El cliente indica su *sequence number* inicial (0), su *tamaño de ventana*, y a través de *options* algunos parámetros para la conexión (como el *maximum segment size* (16K) y el *window scale* (6)).
- . Servidor#80 a Ciente#35150, flag ACK, SYN: Es la aceptación de la conexión del servidor al cliente. El servidor confirma la conexión -flag *ACK* y *acknowledgement number* en 1 (0+1)- y le pide al cliente que él reserve también recursos para la conexión -flag *SYN* y *sequence number* inicial del servidor en 0-.
- . Ciente#35150 a Servidor#80, flag ACK: Es el acuse de recibo del cliente por el segmento ACKSYN que envió el servidor - *acknowledgement number* en 1 (0+1), confirma al segmento con número de secuencia 0 del servidor-.

## **B) Cierre el navegador:**

### **i. Analice la secuencia de mensajes que ocurren al hacerlo ¿Cuál es el objetivo? ¿Qué flags se utilizan en cada caso? ¿Qué indica cada uno?**

Al cerrar el navegador se produce el cierre de la conexión TCP, en cuatro pasos (dado que se cierra por separado desde cada host -*half close*, pudiendo quedar "medio abierta" si sólo uno de ellos la cierra). En el caso de la captura, el Servidor había iniciado el cierre de conexión cuando se cerró el browser:

- . Servidor#80 a Ciente#35150, flag FIN, ACK: El servidor solicita el fin de la conexión. La presencia de ACK se debe a que en el mismo segmento, confirmó la recepción de un segmento anterior del cliente, pero nada tiene que ver con el cierre de conexión.
- . Ciente#35150 a Servidor#80, flag ACK: El cliente da acuse de recibo del pedido de cierre de conexión del servidor. El servidor queda esperando el pedido de cierre desde el cliente.
- . Ciente#35150 a Servidor#80, flag FIN: El cliente solicita el fin de la conexión al servidor.
- . Servidor#80 a Ciente#35150, flag ACK: El servidor da acuse de recibo del pedido de cierre de conexión del cliente.

En este caso, el servidor hizo un *cierre activo*-fue quien inicio el cierre- y el cliente un *cierre pasivo*, bien pudo ser al revés, o pudieron realizar cierres simultáneos.

## **C) Abra un navegador e ingrese a la URL: [www.redes.unlp.edu.ar/](http://www.redes.unlp.edu.ar/) y abra otra instancia o solapa del navegador en ingrese a la URL: [www.redes.unlp.edu.ar/](http://www.redes.unlp.edu.ar/)**

### **i. Observe: IP origen, IP destino, Puerto Origen, Puerto destino, Protocolo de transporte de los paquetes pertenecientes a cada una de las conexiones ¿Cómo es posible conectarse 2 veces en forma simultánea al mismo lugar? ¿Qué distingue una conexión de otra?**

Los protocolos, IPs y port del Servidor coinciden. La diferencia está en el port que el cliente asigna a cada conexión, lo que le permite multiplexarlas.

**D) Desde la consola de root use el servicio tftp:**

**i. Ejecute tftp localhost y copie un archivo cualquiera desde su PC al servidor, a través de la opción put.**

**ii. Borre el archivo de su PC y obtengalo ahora del servidor a través de la opción get.**

**iii. ¿Qué diferencias encuentra con los puntos A y B en cuanto a mensajes intercambiados?**

**iv. ¿Qué diferencias y similitudes encuentra con los puntos A y B respecto a datos incluidos en los campos de control: IP Origen, IP Destino, Puerto Origen, Puerto destino, Protocolo de Transporte?**

- **TFTP: Trivial File Transfer Protocol.** Es un protocolo de transferencia de archivos muy simple, que funciona sobre UDP puerto #69 en vez de TCP #20 y #21 como FTP. Al no ser orientado a conexión, no tiene autenticación de usuario, encriptación, y no puede listar contenidos de directorios. Utiliza un segmento inicial indicando el nombre del archivo, el tipo de operación y el modo de transferencia. Los archivos se transmiten en segmentos numerados de 512 bytes, excepto el último que necesariamente debe ser inferior para indicar el fin de la transmisión (si el archivo tiene un tamaño múltiplo de 512, se envía un segmento con 0 bytes de datos para indicar el fin).

```
~$ ls -l >listado.txt #genero un archivo para probar
```

```
~$ tftp localhost #me conecto -sin usuario ni nada en TFTP
```

```
tftp> ?
```

Commands may be abbreviated. Commands are:

connect	connect to remote tftp
mode	set file transfer mode
put	send file
get	receive file
quit	exit tftp
verbose	toggle verbose mode
trace	toggle packet tracing
status	show current status
binary	set mode to octet
ascii	set mode to netascii
rexmt	set per-packet retransmission timeout
timeout	set total retransmission timeout
?	print help information

```
tftp> put listado.txt listadoEnServidor.txt
```

```
Sent 239 bytes in 0.0 seconds
```

```
tftp> get listadoEnServidor.txt list2.txt
```

```
Received 239 bytes in 0.0 seconds
```

```
tftp> quit
```

```
~$ ls
```

```
captura.pcap Desktop list2.txt listado.txt Maildir
```

```
~$ find / -name listadoEnServidor.txt 2>/dev/null
```

#busco donde

```
guardó el servidor el archivo que subí por TFTP
```

```
/redes/tftpserver/listadoEnServidor.txt
```

```
/fll/cow/redes/tftpserver/listadoEnServidor.txt
```

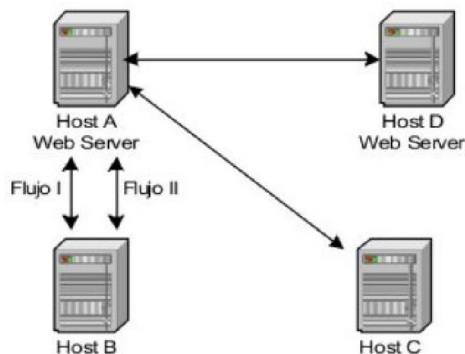


```
/fll/aufs/redes/tftpserver/listadoEnServidor.txt
~$ ls /fll/aufs/redes/tftpserver/listadoEnServidor.txt -l #me fijo con
qué usuario queda
-rw-rw-rw- 1 nobody nogroup 234 may 18 17:11
/fll/aufs/redes/tftpserver/listadoEnServidor.txt
```

**En cuanto a los mensajes intercambiados:** Con TFTP no hay establecimiento de conexión ni confirmaciones. Los mensajes fueron: pedido de escritura del cliente, acuse de recibo del servidor, envío del archivo desde el cliente en bloque #1, acuse de recibo del bloque #1 desde el servidor, pedido de lectura, bloque de datos #1 desde el servidor, acuse de recibo del bloque #1 desde el cliente.

**En cuanto a las IP, puertos y protocolos:** Las IPs corresponden a cliente y servidor. El protocolo de transporte usado es UDP en vez de TCP. Los puertos no se mantienen como en los casos anteriores: varían para cada archivo a transmitir; el cliente se conecta al puerto #69 del servidor, pero recibe las respuestas inmediatamente desde un puerto no reservado (>1023), que se continúa utilizando para la transmisión del primer archivo. Para descargar el segundo archivo, el cliente utiliza un nuevo puerto no reservado y el servidor le responde también desde otro puerto. Sólo se utiliza el puerto #69 para el primer segmento de cada transmisión.

**7. Dado el siguiente gráfico, complete los siguientes cuadros:**



**Tener en cuenta:**

- El Servicio Web corre en el puerto 80 en ambos Hosts.
- B y C acceden al servicio WEB del Host A.
- A accede al servicio WEB del host D.

Flujo I entre A y B				
Origen es A	IP Origen	Puerto Origen	IP Destino	Puerto Destino
	A	80	B	W
Origen es B	IP Origen	Puerto Origen	IP Destino	Puerto Destino
	B	W	A	80
Flujo II entre A y B				
Origen es A	IP Origen	Puerto Origen	IP Destino	Puerto Destino
	A	80	B	X
Origen es B	IP Origen	Puerto Origen	IP Destino	Puerto Destino

**B                      X                      A                      80**

#### Flujo entre A y C

Origen es A	IP Origen <b>A</b>	Puerto Origen <b>80</b>	IP Destino Puerto Destino <b>C                      Y</b>
Origen es C	IP Origen <b>C</b>	Puerto Origen <b>Y</b>	IP Destino Puerto Destino <b>A                      80</b>

#### Flujo entre A y D

Origen es A	IP Origen <b>A</b>	Puerto Origen <b>Z</b>	IP Destino Puerto Destino <b>D                      80</b>
Origen es D	IP Origen <b>D</b>	Puerto Origen <b>80</b>	IP Destino Puerto Destino <b>A                      Z</b>

**En todos los casos indique qué condiciones deben cumplir entre W,X,Y y Z**

Todos mayores a 1023. Las conexiones se inician por B desde W, B desde X, C desde Y y A desde Z -si bien A es Web Server, se comunica con D como cliente, por ello debe iniciar A la conexión desde un #puerto no reservado (>1023).

### 8. ¿Qué es ARQ (Automatic Repeat Request)? ¿Qué capacidades requieren ser implementadas en los protocolos ARQ para detectar la presencia de errores en los datos?

Los protocolos ARQ permiten la transferencia confiable de datos sobre medios no confiables, basándose en la confirmación de recepción de paquetes, la detección de errores y solicitudes de retransmisión.

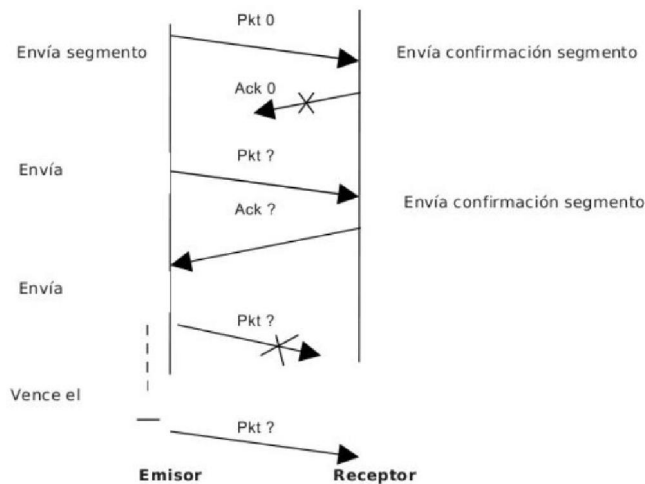
Se debe implementar alguna técnica de detección de errores -campo *checksum*-. A su vez, el receptor debe responder a cada paquete con una señal *ACK*-recepción correcta- o *NACK*-recepción corrupta-. Si se devuelve un NACK o el paquete de respuesta está corrupto, el emisor retransmite el paquete. Si se trataba de un ACK corrupto, los datos habían llegado bien originalmente pero aún así el emisor *duplica el paquete*, para que el receptor pueda identificar paquetes nuevos de *retransmisiones*, se le agrega a los paquetes un *número de secuencia*.

Otra posibilidad, es que el paquete de ACK o NACK nunca llegue al emisor, o que el paquete de datos original nunca llegue al receptor. Para solucionar estos dos inconvenientes, el emisor implementa un **timer**. Si se produce un *timeout* para un paquete enviado y no se ha recibido respuesta ACK ni NACK se asume que nunca llegó el paquete o nunca llegó al respuesta, y se retransmite.

Algunos protocolos ARQ son: **Stop-And-Wait, Go-Back-N y Selective Repeat.**

La misma funcionalidad que se logra con ACK y NACK, puede lograrse sólo con ACK: si se recibe mal un paquete se envía un ACK con el número de secuencia del último paquete bien recibido, en lugar de un NACK. Cuando el emisor recibe un ACK duplicado, reenvía el segmento actual en Stop-and-Wait, o el segmento siguiente en Go-back-N y Selective Repeat.

### 9. Complete los (?) de la siguiente secuencia Stop and Wait:



Por utilizarse **stop and wait**, sólo hay en cada momento un segmento enviado esperando confirmación. El rango de los números de secuencia es sólo [0;1] (es un protocolo sin *pipelining*).

La secuencia completa es: Pkt 0, Ack 0, **Pkt 0, Ack 0, Pkt 1, Pkt 1.**

### 10. Explique la lógica de Go Back – N

Un protocolo **Go Back N** es un protocolo **ARQ** con *pipelining* (es decir, soporta múltiples segmentos en tránsito pendientes de confirmación, implementando buffers en el emisor; el rango de los números de secuencia debe aumentarse).

El emisor maneja una **ventana** de hasta N segmentos (no tendrá nunca más de N segmentos esperando su ACK) y un **timer** para controlar el timeout. La recepción del ACK de un segmento, confirma a todos los anteriores pendientes -**acuse de recibo acumulado**-. El vencimiento del timer para un segmento implica la retransmisión de todos los segmentos siguientes en la ventana. La ventana se desplaza a medida que se reciben ACK's.

El receptor envía el ACK sólo del segmento recibido correctamente de mayor número de secuencia en orden -es decir, que no le falte ninguno-. Puede generar acuses duplicados. No precisa buffers, ya que descarta los segmentos fuera de orden y duplicados -al descartar reenvía el ACK del mayor número de secuencia que recibió-.

## 11.¿Qué restricción existe sobre el tamaño de ventanas en el protocolo Selective Repeat?

FALTA LA RESPUESTA

### Sobre Selective Repeat:

En GBN, si el delay de transmisión y el tamaño de la ventana son grandes, el emisor puede estar mucho tiempo esperando los ACK's, y un solo acuse NACK puede producir la retransmisión de muchos segmentos.

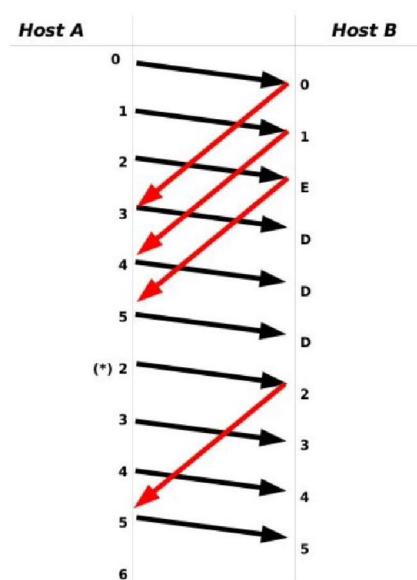
**Selective Repeat** es otro protocolo ARQ con *pipelining*. Tanto el emisor como el receptor manejan una ventana, enviando el receptor ACK's individuales para cada segmento. Esto evita retransmisiones innecesarias, dado que el receptor no descarta tantos segmentos y puede pedir la retransmisión de uno específico.

La llegada de un segmento del comienzo de la ventana del receptor correctamente hace que se envíen a la capa aplicación todos los datos disponibles en orden, desplazando la ventana.

En el emisor, la ventana puede contener segmentos con y sin ACK's -desplazará la ventana cuando los primeros estén con ACK. Deberá manejar timers individuales para cada paquete sin ACK.

Si la ventana del receptor se desliza completa y recibe un segmento anterior a la ventana -retransmisión-, el receptor debe reenviar el ACK correspondiente -o el emisor seguirá reenviándolo cada vez que venza su timer, trabando al desplazamiento de su ventana-.

## 12.Suponiendo Go Back – N y sabiendo que E indica error, D descartar. Indique en el siguiente gráfico:



- **Indicar cada uno de los números de secuencia correspondientes a los ACK de Host B a Host A**

# Secuencia de ACK's: 0, 1, 1, 2

- **¿Qué sucede si en vez de recibir un ACK en (\*) vencería el timeout de este paquete sin haber obtenido confirmación?**

Si en (\*) vence el timer de 2 en vez de recibir un ACK de 1 sucede lo mismo, se retransmite el pkt 2.

- **Considere ambos casos pero con Selective Repeat**

# Secuencia de ACK's: 0, 1, 1, 2 --si D descarta también en SR no se enviarían los ACKs. Sino, pasan a buffer y habrá ACK de 3 4 y 5.

Si en (\*) vence el timer de 2 en vez de recibir un ACK de 1 se reenviaría también el pkt 2. Si se recibieron ACKs de 3 4 y 5, no se retransmitirían, sólo avanzaría la ventana del emisor hasta 6.

### **13.Describa el saludo de tres vías de TCP**

Ver Ejercicio 6.

Si cliente envía SYN y el servidor le niega el servicio -por ejemplo, cliente intenta acceder al puerto #80 y no hay servidor web corriendo-, el servidor responde con RESET.

### **14.¿Es posible usar TCP en comunicaciones multicast?**

No es posible establecer comunicaciones multicast sobre TCP, las mismas se realizan sobre UDP debido a que no son comunicaciones punto a punto. Con TCP antes de comenzar el envío de datos, debería establecerse una comunicación mediante el *handshake* de tres vías. Ese establecimiento de conexión es con una cardinalidad uno a uno, lo que no resultaría práctico para el establecimiento de conexión uno a muchos (lo que es, en forma sencilla, la idea de multicast). Por esto, se utiliza UDP para multicasting, debido a que no es necesario un establecimiento de conexión previo al envío de datos.

### **15.Describa para qué sirve el comando netstat.**

El comando **netstat** imprime información sobre las conexiones de red, tablas de routeo, estadísticas sobre las interfaces de red y membresía multicast.

Por defecto, muestra una lista de los sockets abiertos, indicando protocolo en uso, IP, port local y remoto, y estado de la conexión.

Opciones:

- n** Muestra direcciones numéricas en lugar de intentar determinar nombres simbólicos para hosts, puertos y usuarios.
- a** Muestra todos los sockets, no sólo los que esten en estado *listening*. Con **--interfaces**, muestra incluso las interfaces que no estén iniciadas.
- t** Muestra sólo los puertos que utilicen el protocolo TCP. Igual a **--tcp**.
- u** Muestra sólo los puertos que utilicen el protocolo UDP. Igual a **--udp**.
- p** Muestra el PID del proceso al que pertenece el socket. Igual a **--program**. Sólo mostrará los programas que tiene permiso el usuario; conviene ejecutarlo como *root* si hay servidores funcionando, para ver todos los programas.
- l** Muestra sólo los sockets en estado de *listening* -ocultos por defecto. Igual a **--listening**.
- c** Continuo; se mantiene en funcionamiento ofreciendo su salida cada cierto intervalo. Puede terminarse luego con *Ctrl+C*.

**Considere utilizarlo con los siguientes argumentos y determine que hace en cada caso:**

- **netstat -nat** Direcciones numéricas, todos los sockets, sólo TCP.
- **netstat -natp** Idem anterior, mostrando los PIDs de los procesos que abrieron cada socket.
- **netstat -nlt** Direcciones numéricas, sólo sockets en estado *listening*, sólo TCP.
- **netstat -nau** Direcciones numéricas, todos los sockets, sólo UDP.
- **netstat -naup** Idem anterior, mostrando los PIDs de los procesos que abrieron cada socket.

## Ejercicios Evaluables

### 1. Protocolo FTP:

**a- Explique en qué se diferencia FTP respecto de los canales de comunicación con los demás protocolos de aplicación.**

FTP utiliza dos conexiones TCP paralelas para la transmisión de archivos, una de control/comandos y otra para datos -puertos #21 y #20 del lado servidor respectivamente, aunque el puerto de datos puede variar en conexiones en modo pasivo. Lo habitual es que comandos y datos se envíen por una misma conexión, aunque FTP no es el único que trabaja *out-of-band*.

**b- En el protocolo FTP, el comando *passive* cambia la forma en que el protocolo funciona. Utilizando el capturador de paquetes en el Live CD, capture el intercambio de archivos usando primero el modo activo y luego el modo pasivo. ¿Qué diferencias nota? Represente el comportamiento de cada caso en una tabla como la del ejercicio 7.**

Conexión FTP al *localhost*, se utilizan los comandos ls, get y put en modo activo; se cambia a modo pasivo y se repite la operación. Se sale con quit.

#### ***MODO ACTIVO.***

Conexión inicial, envío de usuario y password, comandos ls, get y put y pasv.

Origen: Cliente#58716, Destino: Servidor#21 -puerto de control-

Respuestas de recepción de comandos:

Origen: Servidor#21, Destino: Cliente#58716

Antes de enviar el comando *ls*, se envía el comando *PORT* indicando el puerto activo del cliente (#41429). Tras el envío de este comando, el servidor abre una segunda conexión para los datos, y envía el resultado del *ls*. Enviados los datos, se cierra la conexión.

Origen: Servidor#20, Destino: Cliente#41429

De igual modo, para el comando *get* se envía antes el comando *PORT* indicando el puerto activo del cliente (#33966). El servidor abre la conexión y envía los datos.

Origen: Servidor#20, Destino: Cliente#33966

Para el comando put:

Origen: Servidor#20, Destino: Cliente#46978

#### ***MODO PASIVO.***

Al pasar a modo pasivo, el user agent envía *PASV* antes de cada comando que precise un canal de datos; el servidor responde código *227 Entering passive mode*, e indica su port pasivo (#46738 en el primer comando usado: ls). El cliente inicia una conexión desde un puerto no privilegiado al pasivo del servidor:

Origen: Cliente#36335, Destino: Servidor#46736 -puerto pasivo indicado como respuesta de *PASV*

Tras cada envío, se realiza el cierre del canal de datos (en ls y get, el servidor es quien realiza primero el FIN; en put, el cliente).

**2. Analice durante la ejecución del ejercicio 6 (A,C y D) de esta práctica la salida del comando netstat. En cada caso, debe ser capaz de distinguir:**

- **Comando utilizado (debe incluir los argumentos)**
- **Puerto del servidor y nombre de la aplicación.**
- **Puerto del o los clientes y nombre de la aplicación.**

### 6. A)

```
~$ sudo netstat -tp
Active Internet connections (w/o servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 lihuen:46522           lihuen:www              ESTABLISHED 6940/firefox-bin
tcp6       0      0 lihuen:www             lihuen:46522            ESTABLISHED 5955/apache2
```

- \* **Comando utilizado:** firefox-bin
- \* **Servidor:** port www (80), aplicación apache2
- \* **Cliente:** port 46522, aplicación firefox-bin

### 6.C)

VER con sudo netstat -natpc >salida.txt así lo miro bien despues....

Del chat:

Alejandro: tenemos la solución para el tema de [www.redes.unlp.edu.ar](http://www.redes.unlp.edu.ar)  
abran una consola

editen con vim /etc/apache2/apache2.conf

busquen Keepalive On, y reemplacen On por Off

y luego reinicien el apache con /etc/init.d/apache2 restart

### 6.D)

Como súper usuario, en una consola se deja el **netstat** trabajando en forma continua mostrando todo lo referente a UDP, con nombre de procesos y valores numéricos. Al estar como súper usuario se puede ver cuál es el servidor TFTP (*in.tftpd*). Las líneas irrelevantes de la salida de **netstat** se resumen con "..."-corresponden a otros servicios en ejecución. En otra consola, se abre el user agent **tftp**, se copia un archivo al servidor con *put* y luego se lo descarga con *get*. Tras salir del user agent, se cierra el netstat con *Ctrl+C*. Se muestran todas las operaciones (los paquetes relevantes detectados por el *sniffer* se muestran en *itálica* junto a *Consola #2*):

#### Consola #1

```
~# netstat -aupcn
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
```

#### Consola #2



```

...
udp      0      0 0.0.0.0:69
0.0.0.0:*                5603/in.tftpd
...
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address   Foreign Address
State          PID/Program name
...
udp      0      0 0.0.0.0:37304
0.0.0.0:*                7406/tftpd
...
udp      0      0 0.0.0.0:69
0.0.0.0:*                5603/in.tftpd
...
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address   Foreign Address
State          PID/Program name
...
udp      0      0 0.0.0.0:69
0.0.0.0:*                5603/in.tftpd
udp      0      0 0.0.0.0:35920
0.0.0.0:*                7406/tftpd
udp      0      0 127.0.0.1:52954 127.0.0.1:37304
ESTABLISHED 7407/in.tftpd
...
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address   Foreign Address
State          PID/Program name
...
udp      0      0 0.0.0.0:69
0.0.0.0:*                5603/in.tftpd
udp      0      0 0.0.0.0:35920
0.0.0.0:*                7406/tftpd
...
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address   Foreign Address
State          PID/Program name
...
udp      0      0 0.0.0.0:55819
0.0.0.0:*                7406/tftpd
...
udp      0      0 0.0.0.0:69
0.0.0.0:*                5603/in.tftpd
udp      0      0 127.0.0.1:53483
127.0.0.1:35920          ESTABLISHED 7410/in.tftpd
...
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address   Foreign Address
State          PID/Program name
...
udp      0      0 0.0.0.0:69
0.0.0.0:*                5603/in.tftpd
...
^C

```

**~\$ tftp localhost**

**tftp> put  
cap1.pcap**

*pkt C#37304 ->  
S#69, write  
request cap1.pcap  
pkt S#52954 ->  
C#37304, ACK  
26 pkts C#37304 -  
> S#52954, data  
packets  
26 pkts S#52954 -  
> C#37304,  
ACKs*

Sent 13211 bytes  
in 0.0 seconds

**tftp> get  
cap1.pcap  
cap77.pcap**

*pkt C#35920 ->  
S#69, read  
request cap1.pcap  
26 pkts S#53483 -  
> C#35920, data  
packets  
26 pkts C#35920 -  
> S#53483, ACKs*

Received 13211  
bytes in 0.0  
seconds

**tftp> quit  
~\$**

## Capa De Red

### 1. ¿Qué servicios presta la capa de red? ¿Cuál es la PDU en esta capa?

La capa de Red permite la conexión desde un host origen a un host destino. En TCP/IP está implementada en el protocolo **IP**, e interviene en cada host y encaminador intermedio **-router-**.

El PDU es el **datagrama**, que encapsula los segmentos de **Transporte** agregándole las direcciones IP origen y destino.

Ofrece dos tipos de servicios:

- **Retransmisión (forwarding):** Los datagramas que llegan a un router son dirigidos a la interfaz de salida apropiada.
- **Encaminamiento (routing):** Determina el camino que toman los paquetes desde el origen al destino (según diferentes algoritmos de ruteo).

Otro tipo de servicio, aunque ausente en Internet, es el **establecimiento de conexión** (en redes tipo ATM -de conmutación de circuitos-): los routers intermedios del camino establecen la conexión virtual, reservando recursos -bandwidth, buffers...- y "*recordando*" el camino elegido antes que los paquetes se comiencen a transmitir (agregan entrada a tabla de ruteo).

### 2. Compare los siguientes modelos de servicios de red:

	Datagrama	Circuitos Virtuales
¿Todos los paquetes siguen el mismo camino?	NO	SI
¿Cuenta con una fase de establecimiento y otra de cierre de circuito?	NO	SI
¿Usa mensajes de señalización?	NO	SI -para establecimiento/mantenimiento/cierre de conexión-
¿Usa tablas de enrutamiento?	SI (rango/prefijo IPs destino, interface OUT)	SI (interface IN, #CV IN, interface OUT, #CV OUT)

3. ¿Qué dispositivo es considerado sólo de esta capa? Explique las dos funciones principales que debe realizar.

El dispositivo de capa de Red es el router. Acorde a los dos servicios de la capa de Red, este dispositivo debe:

- Ejecutar algoritmos/protocolos de enrutamiento que seleccionen hacia dónde reenviar un datagrama recibido.
- Encaminar/conmutar los datagramas que llegan a una interfaz o puerto de entrada, a la interfaz o puerto de salida seleccionada por el algoritmo. La conmutación puede hacerse vía memoria -control directo de una CPU-, vía bus compartido en el router o vía *crossbar* red de interconexión.

El puerto de salida puede realizar *buffering* si su tasa de transmisión es inferior a la de llegada de datos desde el entramado de conmutación -produciendo un *retraso*-; si el buffer se llena, pueden *perderse* paquetes. Del mismo modo si la tasa de llegada de datagramas al puerto de entrada es superior a la velocidad de conmutación, el puerto de entrada utiliza *buffering*.

**4. En las redes IP el ruteo puede hacerse en forma tanto estática como dinámica. Describa conceptualmente como funciona cada uno de ellos e indique ventajas y desventajas de cada método.**

- **Ruteo estático:** En redes en las que las rutas cambian muy lentamente en el tiempo, por ejemplo porque un administrador edita manualmente la tabla de ruteo de un router. Se basan en tablas de ruteo.
  - **V:** Algoritmos de ruteo más simples. Apropiado si el tráfico de red es predecible. Los cambios se reflejan ni bien el administrador modifica las tablas. No hay sobrecarga de la red -no hay intercambio de información entre routers-. No hay problemas de seguridad ni compatibilidad. No hay procesamiento extra.
  - **D:** Precisa configuración manual. Se vuelve inmanejable en redes grandes y cambiantes. Propensa a errores -por ejemplo, si no se actualiza una tabla un router puede enviar información por un camino que no llegará nunca al destino; eventualmente el datagrama será descartado por vencer su *TTL* o se perderá por *buffer overflow*.
- **Ruteo dinámico:** Las rutas cambian frecuentemente de acuerdo a la carga de la red y las esperas que se producen, o bien por cambios en la topología. Puede ejecutarse el algoritmo de ruteo dinámico periódicamente o por cambios en la topología o en los costos de los enlaces. Si un router detecta un cambio, recalcula su tabla y propaga el cambio a los vecinos para que se recalculen.
  - **V:** Reacciona automáticamente a cambios en la red. Precisa poca configuración.
  - **D:** Susceptible a *ciclos* y a *oscilaciones* entre routers. Sobrecarga de red por intercambio de información entre routers. Procesamiento extra por el algoritmo. Los cambios detectados por un router pueden tardar en informarse al resto.

**5. Los algoritmos de ruteo dinámico se dividen en Estado Enlace y Vector Distancia. Dado el siguiente cuadro compare:**

	Vector distancia	Estado de enlace
<b>¿Cada router conoce la topología completa?</b>	<b>NO</b> (decentralizado)	<b>SI</b> (global)
<b>¿Converge rápidamente?</b> Tiempo de convergencia: Velocidad con la que los routers comparten información. La convergencia ocurre cuando todos los routers del dominio están de acuerdo en las rutas que se encuentran disponibles.	<b>NO</b>	<b>SI</b>
<b>Protocolos que lo implementan</b>	RIP v1 y v2 <b>IGRP</b> <b>GGP</b>	<b>OSPF</b> <b>IS-IS</b> (de ISO)

**Vector distancia:** Algoritmo descentralizado, iterativo, asíncrono y distribuido. Cada nodo conoce sólo a sus vecinos, y ninguno conoce el camino completo para un paquete. Cada nodo realiza sus cálculos para mantener su tabla de distancias (con distancia por vecino/destino) e informa los resultados **completos** a sus vecinos directos; el proceso se repite hasta que ningún nodo intercambia información. **Propenso a bucles.**

Protocolos:

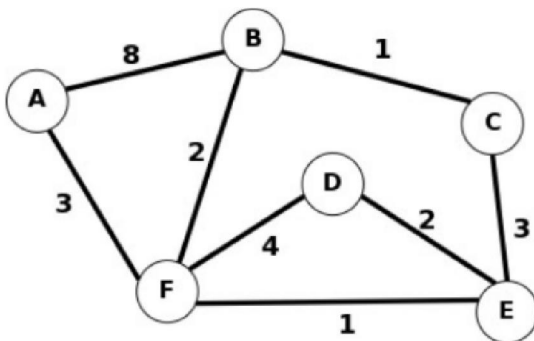
- **RIP:** comparte toda la tabla cada 30 segundos. La versión 1 considera sólo clases completas transmitiendo por *broadcast*, la 2 envía en los paquetes la máscara de subred, por *multicast*. Métrica: cantidad de "saltos" (limitada hasta 15, valores mayores se ignoran para evitar bucles).

- **IGRP:** comparte toda la tabla por broadcast cada 90 segundos, y las actualizaciones al ser detectadas. Métrica compuesta por ancho de banda, retardo, carga, fiabilidad y MTU (Maximum Transfer Unit, la cantidad máxima de información que pueda transportar un paquete de la capa de enlace con el protocolo que se use en ella).

**Estado de Enlace:** Cada nodo conoce la imagen completa del "grafo" que representa la red y costo **sólo de los routers vecinos**, pudiendo todos conocer así la estructura completa). Aplican **Dijkstra** para calcular el camino de costo mínimo desde el origen al destino, obteniendo el camino completo para el paquete. **Libre de bucles.**

Lo primero que hace un router es detectar a sus vecinos enviando un paquete **HELO** por *broadcast*, si alguien responde le envía un **ECHO** para medir el tiempo de respuesta. Luego construye un paquete con la información recogida -puede hacerlo periódicamente o sólo al detectar cambios-, y lo informa a la red por medio de *inundación o flooding*. Cada paquete tiene un #secuencia, de modo que sólo se procesan y retransmiten los de #secuencia mayor al último recibido. Estos paquetes a su vez, mantienen su *edad-TTL* -que se decrementa en cada router, eliminándose los paquetes de edad 0 (evitando problemas por bucles).

**6. Dado el apunte que explica en detalle los algoritmos de ruteo dinámicos de estado enlace y vector distancia, y dado el siguiente grafo, indique el procedimiento para calcular el camino de costo mínimo a partir del nodo B según los siguientes cuadros:**



ESTADO DE ENLACE					VECTOR DISTANCIA				
Iteración	N	D(A),p(A)	D(C),p(C)	D(D),p(D)	D <sup>B</sup> (,)	A	F	C (< vecinos)	
D(E),p(E)	D(F),p(F)								
0	B	8,B	<u>1,B</u>		2,B	A	8	<u>5</u>	7 conviene ir a A pasando por F
1	BC	8,B		4,C	<u>2,B</u>	C	11	5	<u>1</u> conviene ir a C directamente
2	BCF	5,F	6,F	<u>3,F</u>		D	11	<u>5</u>	6 conviene ir a D pasando por F
3	BCFE	<u>5,F</u>	5,E			E	9	<u>3</u>	4 conviene ir a E pasando por F
4	BCFEA		<u>5,E</u>			F	8	<u>2</u>	4 conviene ir a F directamente
5	BCFEAD				( ^ destinos )				

**7. ¿Qué son los sistemas autónomos y por qué resultan necesarios?**

Un **Sistema Autónomo (Autonomous System, AS)** es un conjunto de redes bajo la misma administración (podría ser gestionada por más de un operador de red), y utilizando uno o varios protocolos de enrutamiento internamente, independientemente de la red de su proveedor. Cada SA mantiene una clara y única política de ruteo. Cada AS en Internet debe tener un número identificador: **ASN** (AS Number).

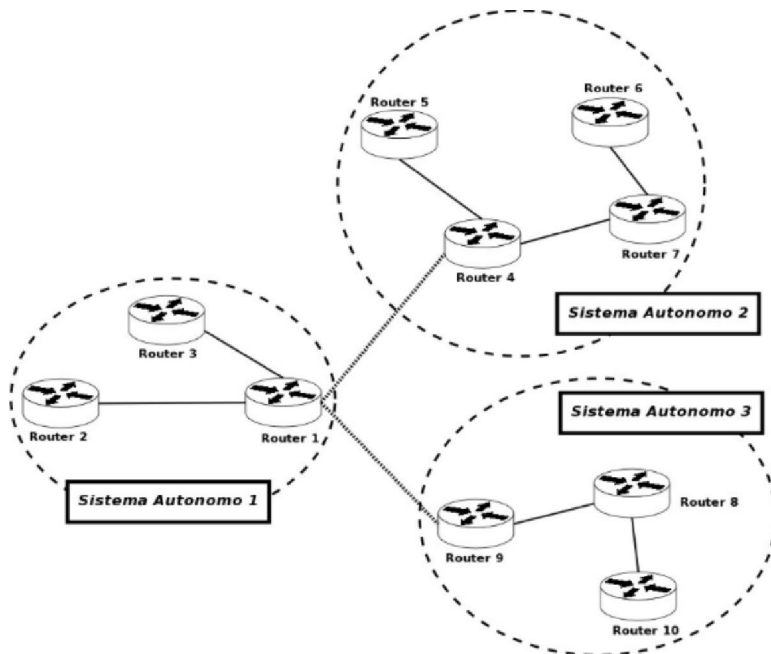
Los protocolos de ruteo que utilizan internamente se denominan **IGP** (Interior Gateway Protocol), y pueden ser: RIP, IGRP, EIGRP, OSPF, IS-IS...

Los SA permiten el **ruteo jerárquico**. Cada SA se conecta a un *router de borde o gateway*, que lo conecta a otros. Se denominan **EGP** (Exterior Gateway Protocols) a los protocolos entre distintos AS (GGP, EGP, BGP).

El **ruteo jerárquico** permite salvar los problemas que supondrían utilizar los mismos protocolos entre diferentes subredes:

- **Escala:** El enorme **tamaño** que alcanzarían las tablas de ruteo, el **overhead** de intercambiar información entre los routers para una red de grandes dimensiones y el enorme **tiempo de convergencia** que se tendría.
- **Autonomía administrativa:** La imposibilidad de elegir un protocolo específico por parte de los administradores de las redes.

## 8. A partir del siguiente gráfico indique



¿Qué tipo de algoritmo se utiliza para compartir información entre los routers 4 y 7?  
IGP

¿Qué tipo de algoritmo se utiliza para compartir información entre los routers 1 y 4?  
EGP

¿Qué tipo de algoritmos alimentan las tablas de ruteo de los routers 3 y 10?  
IGP

¿Qué tipo de algoritmos alimentan las tablas de ruteo de los routers 1 y 4?  
EGP e IGP -para ruteo entre SA y dentro del SA al que pertenece cada uno-

## 9. Dadas las siguientes direcciones IP, indique su clase y si las mismas corresponden a direcciones de máquina, de red o broadcast

Dirección IP	Clase	Tipo
203.15.6.87	C	Host/Interface
67.154.0.0	A	Host/Interface
171.58.0.0	B	Red
127.0.0.1	A --Loopback en realidad--	Loopback / Host/Interface
24.130.56.0	A	Host/Interface
197.54.66.255	C	Broadcast

Clase	Rango	Primer Byte	Nº de Redes	Nº de Host	Máscara de Red	Broadcast
A	1.0.0.0 - 127.0.0.0	0xxx xxxx	126	16.777.214	255.0.0.0	x.255.255.255
B	128.0.0.0 - 191.255.0.0	10xx xxxx	16.384	65.534	255.255.0.0	x.x.255.255
C	192.0.0.0 - 223.255.255.0	110x xxxx	2.097.152	254	255.255.255.0	x.x.x.255
D - Multicast	224.0.0.0 - 239.255.255.255	1110 xxxx				

E -  
*Reservada*- 240.0.0.0 - 1111 xxxx  
255.255.255.255

- Una dirección IP no identifica un equipo (computadora/router) específico, sino que identifica una **interface** de un host en una red. Un equipo con acceso a múltiples redes (por ejemplo, un router) debe tener asignada una dirección IP por cada una de éstas.
- La dirección 0.0.0.0 es utilizada por las máquinas cuando están arrancando o no se les ha asignado dirección.
- La dirección que tiene su parte de host a cero sirve para definir la red en la que se ubica. Se denomina **dirección de red**.
- La dirección que tiene su parte de host a unos sirve para comunicar con todos los hosts de la red indicada. Se denomina **dirección de broadcast directo**.
- La dirección que tiene su parte de red y de host a unos sirve para comunicarse con todos los equipos de la red local (IP 255.255.255.255). Se denomina **dirección de broadcast limitado**.
- Las direcciones 127.x.x.x se reservan para pruebas de retroalimentación. Se denomina **dirección de bucle local** o **loopback**.
- Hay ciertas direcciones en cada clase de dirección IP que no están asignadas y que se denominan **direcciones privadas**; pueden ser utilizadas por los hosts que usan *traducción de dirección de red* (NAT) para conectarse a una red pública o por los hosts que no se conectan a Internet. En una misma red no puede existir dos direcciones iguales, pero sí se pueden repetir en dos redes privadas que no tengan conexión entre sí o que sea a través de NAT. Las direcciones privadas son:
  - Clase A: 10.x.x.x (8 bits red, 24 bits hosts)
  - Clase B: 172.16.x.x a 172.31.x.x (12 bits red, 20 bits hosts)
  - Clase C: 192.168.x.x (16 bits red, 16 bits hosts)

## 10. ¿Qué son las subredes?

La división de subredes es la obtención de otras direcciones de red basadas en una sola dirección con el uso de la **máscara de subred**, "*pidiendo bits prestados*" a la parte del host/interface (dependiendo de la cantidad de bits que se pidan, será la cantidad de subredes que se creen a partir de la original).

Una subred es un **rango de direcciones lógicas**. Cuando una red se vuelve muy grande, conviene dividirla en subredes, para reducir el tamaño de los dominios de broadcast, y hacer la red más manejable.

Típicamente los routers constituyen los límites entre las subredes. La comunicación desde/hasta otras subredes es hecha mediante un router específico. Sin embargo, las subredes permiten dividir lógicamente una red a pesar de su diseño físico, pudiéndose dividir en varias subredes configurando diferentes hosts que utilicen diferentes routers. La dirección de todos los nodos en una subred comienzan con la misma secuencia binaria, que es su ID de red e ID de subred (en IPv4, las subredes deben ser identificadas por la **base** de la dirección y una **máscara de subred**).

Las subredes simplifican el enrutamiento, representándose típicamente cada una como una fila en las tablas de ruteo en cada router conectado. Fueron usadas antes de IPv4 para permitir a una red grande tener un número importante de redes más pequeñas dentro, controladas por varios routers.

Permiten el Enrutamiento Interdominio Sin Clases (**CIDR**).

Los últimos dos bits del último octeto (los menos significativos) nunca se asignan a la subred, sea cual sea la clase de dirección IP. Como consecuencia, una dirección de subred jamás terminará en un número impar. Por otra parte, el uso de todos los bits disponibles para crear subredes dará como resultado subredes con sólo dos Hosts utilizables (un método práctico de conservación de direcciones para el direccionamiento de enlace *punto a punto*, donde no existe otro direccionamiento más que los dos enlaces conectado entre sí).

**11. Dada la red 192.200.45.0. Se necesitan definir 9 subredes. Indique la máscara utilizada y las nueve primeras subredes. Luego tome una de ellas e indique el rango de direcciones asignables en esa subred, dirección de red y broadcast.**

IP: 192.200.45.0, en binario: **110**00000 11001000 00101101 00000000

Como empieza con 110 entonces es clase C. La máscara de red es 11111111 11111111 11111111 00000000 o /24.

Para obtener 9 subredes, se piden prestados 4 bytes:  $2^4 = 16 > 9$  (si se pedían 3 se tendrían  $2^3=8$  subredes solamente, así que se crearán más subredes de las necesarias)

Se usará la máscara de subred /28 o 255.255.255.240.

Las direcciones de las subredes son:

192.200.45.0000 0000 = 0	subred #1
.0001 0000 = 16	subred #2
.0010 0000 = 32	subred #3
.0011 0000 = 48	subred #4
.0100 0000 = 64	subred #5
.0101 0000 = 80	subred #6
.0110 0000 = 96	subred #7
.0111 0000 = 112	subred #8
.1000 0000 = 128	subred #9
.1001 0000 = 144	subred #10 (no se utilizará)
.1010 0000 = 160	subred #11 (no se utilizará)
.1011 0000 = 176	subred #12 (no se utilizará)
.1100 0000 = 192	subred #13 (no se utilizará)
.1101 0000 = 208	subred #14 (no se utilizará)
.1110 0000 = 224	subred #15 (no se utilizará)
.1111 0000 = 240	subred #16 (no se utilizará)

Para la subred #2:

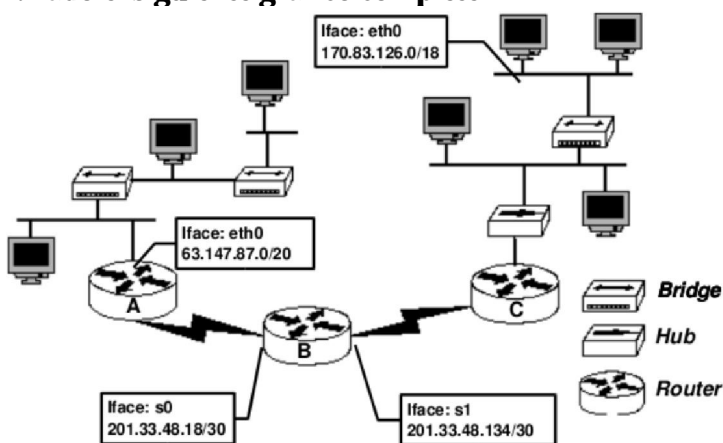
Dirección de Subred: 192.200.45.16/28 (0001 0000)

Dirección de broadcast: 192.200.45.31/28 (0001 1111)

Rango de direcciones asignables: 192.200.45.17/28 - 192.200.45.30/28

Cada subred tendrá  $2^4 - 2 = 14$  direcciones IP disponibles (14 interfaces).

**12. Dado el siguiente gráfico complete**



Interface	IP	Dir. Red	Clase	Dir. Subred	Máscara	Broadcast	Cant. Dir. Utilizables
eth0	170.83.126.0	170.83.0.0	B	#2: 170.83.64.0	/18 = 255.255.192.0	170.83.127.255	16382
eth0'	63.147.87.0	63.0.0.0	A	#2357: 63.147.80.0	/20 = 255.255.240.0	63.147.95.255	4094
s0	201.33.48.18	201.33.48.0	C	#4: 201.33.48.16	/30 = 255.255.255.252	201.33.48.19	2 -conexión punto a punto-
s1	201.33.48.134	201.33.48.0	C	#33: 201.33.48.132	/30 = 255.255.255.252	201.33.48.135	2 -conexión punto a punto-

## 1. Con los datos dados y para cada segmento de red indique:

Hasta 127 A; hasta 191 B; hasta 223 C; hasta 239 Multicast.

Subred de eth0:

170.83.126.0 en 170.83.0.0/18

$2^{(18-16)} = 2^2 = 4$  subredes de  $2^{(32-18)} = 2^{14} = 16384$  interfaces cada una (16382 útiles).

126 = 01111110 subred #2 dirección: 170.83.01000000.0 = subred 170.83.64.0

broadcast: 170.83.01 111111.255 = broadcast 170.83.127.255

Subred eth0:

63.147.87.0 en 63.0.0.0/20

$2^{(20-8)} = 2^{12} = 4096$  subredes de  $2^{(32-20)} = 2^{12} = 4096$  interfaces cada una (4094 útiles)

63.10010011.01010111 subred #2357 dirección: 63.147.01010000.0 = subred 63.147.80.0

broadcast: 63.147.0101 1111.255 = broadcast 63.147.95.255

Subred de s0:

201.33.48.18 en 201.33.48.0/30

$2^{(30-24)} = 2^6 = 64$  subredes de  $2^{(32-30)} = 4$  interfaces cada una (2 útiles)

18 = 00010010 subred #4 dirección: 201.33.48.16

broadcast: 201.33.48.000100 11 = broadcast 201.33.48.19

Subred de s1:

201.33.48.134 en 201.33.48.0/30

$2^{(30-24)} = 2^6 = 64$  subredes de  $2^{(32-30)} = 4$  interfaces cada una (2 útiles)

134 = 10000110 subred #33 dirección: 201.33.48.132

broadcast: 201.33.48.100001 11 = broadcast 201.33.48.135

## 2. Indique la dirección IP en cada una de las interfaces de cada uno de los routers.

**Router A interfaz a B es: 201.33.48.17**

Como tiene máscara /30 es para sólo dos IP disponibles  $2^{(32-30)} - 2 = 2$ . La interfaz de B tiene 201.33.48.18 así que la restante es la de A.

201.33.48.16 (subred), 201.33.48.17 (A), 201.33.48.18 (B), 201.33.48.19 (broadcast)

**Router C interfaz a B: 201.33.48.133**

201.33.48.132 (subred), 201.33.48.133 (C), 201.33.48.134 (B), 201.33.48.135 (broadcast)

La interfaz de C hacia 170.83.0.0 no se puede determinar con la información disponible

## 3. Defina una tabla de ruteo para cada router en el gráfico, de forma tal de que todos los dispositivos en la red puedan comunicarse

Tabla Router A			Tabla Router B			Tabla Router C		
Red Destino	Próximo Router*	Cant Saltos**	Red Destino	Próximo Router*	Cant Saltos**	Red Destino	Próximo Router*	Cant Saltos**
63.0.0.0	-	1	63.0.0.0	201.33.48.17 (A)	2	63.0.0.0	201.33.48.134 (B)	3
170.83.0.0	201.33.48.18 (B)	3	170.83.0.0	201.33.48.133 (C)	2	170.83.0.0	-	1
201.33.48.0	-	1	201.33.48.0	-	1	201.33.48.0	-	1

\* gateway, \*\* métrica



**13. Dada la IP 65.0.56.34. Se necesitan definir 934 subredes. Indique cuál es la máscara utilizada, y luego describa la subred 817 indicando el rango de direcciones asignables, dirección de red y broadcast.**

IP 65.0.56.34 es **Clase A** ( $65 < 127$ ); se usaran **8 bits para dirección de red**. Para lograr 934 subredes se precisan:  $\log_2(934) = 9.86$  es decir **10 bits para indicar la subred**.

Así, la máscara completa es: 255.1111 1111.1100 0000.0 = **máscara de subred 255.255.192.0** (se definirán  $2^{10} = 1024$  redes, sólo se utilizarán las primeras 934).

Se tendrá disponible  $2^{(32-18)} = 2^{14} = \mathbf{16384}$  direcciones IP. Serán utilizables:  $16384 - 934 \times 2 = \mathbf{14516}$  útiles (descontando subred y broadcast de cada subred).

Para la subred #817 ( $817_{10} = 1100110001_2$ , + 1 por el 0:  $1100110010_2$ ):

Dirección de red: 65.1100 1100.1000 0000.0 = **65.204.128.0 (subred)**

Broadcast: 65.1100 1100.1011 1111.255 = **65.204.191.255 (broadcast)**

Rango asignable: **desde 65.204.128.1 hasta 65.204.191.254.**

**14. Dada la dirección IP 172.16.58.223/26. ¿Cuál es la dirección de subred? ¿Y la de broadcast?**

La IP 172.16.58.223 sería de **clase B** ( $127 < 172 < 192$ ). Máscara de Red: 255.255.0.0. Dirección de RED: 172.16.0.0. Dirección de broadcast de RED: 172.16.255.255

**/26 = 255.255.255.192** es la máscara de subred

$2^{(26 \text{ bits de máscara subred} - 16 \text{ bits por ser clase B})} = 2^{10} = \mathbf{1024}$  subredes de  $2^{(32-26)} - 2 = 2^6 - 2 = \mathbf{64 - 2 = 62}$  interfaces útiles en cada subred

**172.16.58.223** = 172.16.00111010.11011111

Subred: 172.16.58.11000000 = 172.16.58.192 dir. de subred

Broadcast: 172.16.58.11111111 = 172.16.58.255 dir de broadcast

**15. Realizar la máxima agregación CIDR (Class Interdomain routing) posible del siguiente conjunto de 4 redes clase C.**

**Classless Inter-Domain Routing (CIDR)** Encaminamiento Inter-Dominios sin Clases) es una mejora en el modo como se interpretan las direcciones IP. Su introducción permitió una mayor flexibilidad al dividir rangos de direcciones IP en redes separadas, reemplazando la sintaxis previa para nombrar direcciones IP, las **clases** de redes.

CIDR usa la técnica **VLSM (Variable-Length Subnet Masking** - Máscara de Subred de Longitud Variable), para hacer posible la asignación de prefijos de longitud arbitraria (la división red/host puede ocurrir en cualquier bit de los 32 que componen la dirección IP).

Un gran beneficio de CIDR es la posibilidad de agregar prefijos de encaminamiento **-supernetting-**. Por ejemplo, dieciséis redes /24 contiguas pueden ser agregadas y publicadas en los routers como una sola ruta /20 (si los primeros 20 bits de sus respectivas redes coinciden). Dos redes /20 contiguas pueden ser agregadas en una /19, etc. Esto permite reducir significativamente el número de rutas que los routers tienen que conocer (reduciendo memoria, recursos, etc.) y previene una explosión de tablas de encaminamiento.

Decimos que una dirección IP **está incluida** en un **bloque CIDR**, y que **encaja** con el prefijo CIDR, si los N bits iniciales de la dirección y el prefijo son iguales. Por tanto, para entender CIDR es necesario visualizar la dirección IP en binario:

**212.56.132.0/24** : 212.56.1000 0100.0

**212.56.133.0/24** : 212.56.1000 0101.0

**212.56.134.0/24** : 212.56.1000 0110.0

**212.56.135.0/24** : 212.56.1000 0111.0

$8 + 8 + 6 = 22$  bits iguales. Los 2 bits restantes del 3er octeto toman las  $2^2=4$  combinaciones posibles, así que es posible reducirlas a un solo bloque.

Las cuatro redes clase C pueden agregarse en el bloque CIDR: **212.56.132/22**

## 16. Listar las redes que involucra el bloque CIDR 200.56.168.0/21.

Se trata de redes clase C ( $192 < 200 < 223$ ).

$200.56.168.0 = \underline{200.56.1010\ 1000.0}$

$8 + 8 + 5 = 21$  bits iguales en el bloque.

El tercer byte queda con 3 bits fuera del bloque: hay  $2^3 = 8$  redes contenidas en el bloque

#1  $200.56.168.0 = \underline{200.56.1010\ 1000.0}$

#2  $200.56.169.0$

#3  $200.56.170.0$

#4  $200.56.171.0$

#5  $200.56.172.0$

#6  $200.56.173.0$

#7  $200.56.174.0$

#8  $200.56.175.0 = \underline{200.56.1010\ 1111.0}$

## 17. Listar las redes que involucra el bloque CIDR 195.24/13.

Redes clase C ( $192 < 195 < 223$ ).

$195.24 = \underline{1100\ 0011.0001\ 1000}$ . 13 bits iniciales iguales en redes de 24 bits de parte de red.  $24 - 13 = 11$

bits restantes "resumidos", hacen a un total de  $2^{11} = \mathbf{2048}$  redes tipo C resumidas.

Las redes en el bloque van **desde la 195.24.0.0 hasta la 195.31.255.0**.

## 18. Una máquina que se conecta a Internet, ¿tiene tabla de ruteo?

Los host también mantienen tablas de ruteo, en el caso más simple conteniendo la red local (la del segmento al que se conecta su interfaz si tiene sólo una tarjeta de red). Si el equipo está conectado a internet, tendrá también una entrada para su gateway por default.

## 19. El comando netstat presentado en la práctica anterior, al igual que el comando route permite visualizar las tablas de ruteo. Analice su salida.

# route

Kernel IP routing table

Destination	Gateway	Genmask	Flags	Metric	Ref	Use
Iface						
10.0.2.0	*	255.255.255.0	U	0	0	0
eth0						
default	10.0.2.2	0.0.0.0	UG	0	0	0
eth0						

La primera columna indica *dirección de red* (debe interpretarse de acuerdo a la *máscara* de la tercera columna) y la segunda columna indica el *gateway*-router- a utilizar (0.0.0.0 o \* se utiliza para indicar que hay conexión directa a la red en cuestión).

La columna *Flags* indica las características de la ruta:

U (route is **u**p)

H (target is a **h**ost)

G (use **g**ateway)

R (**r**einstall route for dynamic routing)

D (**d**ynamically installed by daemon or redirect)

M (**m**odified from routing daemon or redirect)

A (installed by **a**ddrconf)

C (**c**ache entry)

! (reject route)

*Metric*: Valor utilizado para cuantificar la ruta. IP utiliza este valor para seleccionar la mejor de dos o más rutas alternativas a la misma red.

*Ref*: Cantidad de veces que esta ruta fue utilizada para establecer una conexión.

*Use*: Cantidad de paquetes transmitidos a través de esa ruta.

*Iface*: Interfaz de salida a la ruta.

**\$ netstat -r**

```
Kernel IP routing table
Destination      Gateway          Genmask         Flags   MSS Window  irtt  Iface
10.0.2.0         *               255.255.255.0   U        0  0        0  eth0
default          10.0.2.2        0.0.0.0         UG       0  0        0  eth0
```

La salida de netstat agrega las columnas: *MSS*-Maximum Segment Size-, *Window*-tamaño de la ventana de TCP para las conexiones hacia esa ruta- e *irtt*-initial round trip time para conexiones TCP, en milisegundos.

## 20.Describa qué hacen los comandos ping y traceroute (tracert en windows).

**ping -bfr -c count-i interval-I interface/address -s packetsize-t ttl -w deadline-W timeout-APDestino**

Utiliza el protocolo ICMP para solicitar a un host/gateway una respuesta de *echo*. Envía un datagrama ECHO\_REQUEST al IP indicado. Útil para controlar que una interfaz este funcionando o que hay conectividad. Al terminar, muestra algunas estadísticas como cantidad de paquetes perdidos, TTL mínimo/medio/máximo...

- b Permite hacer ping a una dirección de broadcast.
- c *count* Se detiene luego de enviar *count*ECHO\_REQUEST.
- f Imprime un . por cada ECHO\_REQUEST y un backspace (borra el punto) por cada ECHO\_REPLY recibido, mostrando cuantos paquetes se pierden
- i *interval* Permite indicar el intervalo a esperar, en segundos, entre cada paquete.
- I *interface/address* Permite indicar la interfaz/IP que se usará como dirección de origen.
- R Incluye la opción Record route en el paquete ECHO\_REQUEST y muestra el buffer de ruta de los paquetes (el encabezado IP tiene tamaño suficiente para sólo 9 rutas).
- n Mostrar direcciones numéricas, no intentar obtener nombres de host.
- r Permite saltar las tablas de ruteo normal y enviar el paquete a un equipo en una interfaz adjunta.
- s *packetsize* Cantidad de bytes de datos a transmitir.
- t *ttl* Permite indicar el TTL de IP -time to live-.
- w *deadline* Permite indicar un tiempo máximo de funcionamiento.Ping terminará pasado ese tiempo sin importar cuántos paquetes haya enviado o recibido. Si se usa junto a -c, ping terminará al cumplirse el tiempo o al enviar la cantidad indicada de paquetes, lo que ocurra primero.
- W *timeout* Cantidad de segundos queping esperará una respuesta. Por defecto, dos RTTs.

**traceroute -4|6 -ITFn -f firstTTL-m maxTTL-N squeries-p port-q nqueries host**

Muestra la ruta que siguen los paquetes por la red hasta llegar a un destino.

Utiliza el campo TTL de IP, enviando paquetes con valores incrementales desde TTL=1, y recibiendo las respuestas ICMP de tipo TIME\_EXCEEDED de cada gateway/router en el camino hasta el destino.

Para cada valor de TTL envía tres paquetes, luego imprime el gateway que respondió y el RTT obtenido. Si no obtiene respuesta en 5 segundos (por defecto) para algún paquete, imprime un asterisco. Los paquetes se envían a un puerto UDP que difícilmente se utilice, de modo que al llegar al destino, éste responda un ICMP PORT\_UNREACHABLE o un segmento TCP de tipo RESET, denegando la conexión.

- 4|6 Forzar el uso de IPv4 o IPv6
- I Utilizar ICMP ECHO para los paquetes de sondeo. Si no se indica -I ni -T, utiliza UDP.
- T Utilizar TCP SYN para los paquetes de sondeo. Si no se indica -I ni -T, utiliza UDP.

- F Marca el bit de "Don't Fragment" para que los routers intermedios no fragmenten paquetes mayores al MTU -Maximun Transfer Unit- del enlace.
- f *firstTTL* Indica con qué valor de TTL comenzar (por defecto, 1)
- i *interface* Permite indicar la interfaz a utilizar para los paquetes.
- m *maxTTL* Permite indicar la cantidad maxima de "saltos" que se intentará (por defecto, 30).
- N *squeries* Cantidad de paquetes a enviar simultáneamente (por defecto, 16).
- n Mostrar direcciones numéricas, no intentar obtener nombres de host.
- p *port* Para tracing con UDP/TCP indica el número de puerto destino a utilizar. Para ICMP, el número de secuencia inicial.
- q *nqueries* Cantidad de paquetes para cada "salto" posible (cada valor TTL). Por defecto, 3.

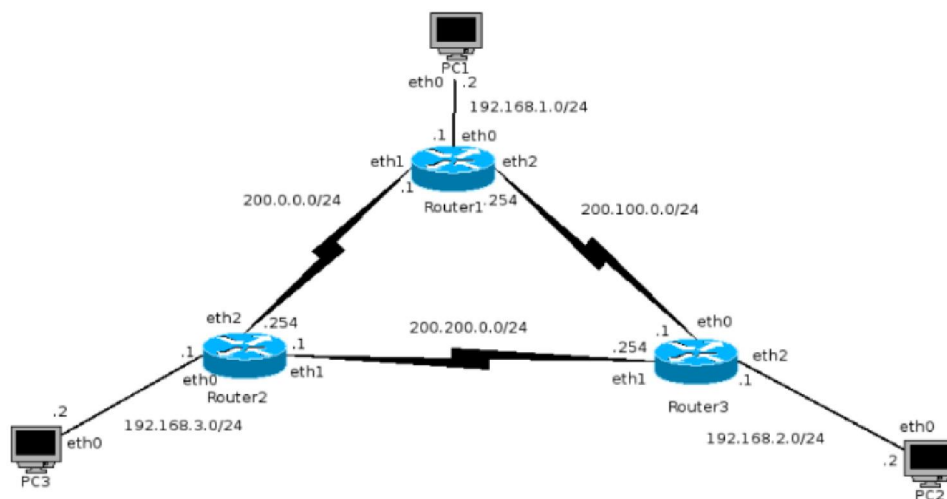
## 21.¿Qué es y para qué sirve la dirección 127.0.0.1? ¿quién responde al siguiente comando: ping 127.0.0.1?

**127.0.0.1/8** es la dirección de **retroalimentación** o **loopback** de todo equipo con placa *Ethernet*. Hace referencia al mismo host que envía el paquete, de modo que éste es procesado por los protocolos del stack TCP/IP pero no llega a salir nunca por la interfaz, sólo se maneja en memoria. Si bien lo más frecuente es utilizar 127.0.0.1, toda la gama 127.x.x.x tiene el mismo efecto, ya que se encuentra reservada para igual propósito.

Se puede usar para probar el funcionamiento de TCP/IP: al hacer **ping 127.0.0.1**, los paquetes se manejan localmente, por lo que responde el mismo host. Así puede asumirse que los componentes asociados al protocolo están bien -sería un primer paso para aislar problemas de red, por ejemplo-.

El equivalente en IPv6 es **::1/128**.

## 22.Utilizando el LiveCD provisto por la cátedra, se simulará la red que muestra el siguiente gráfico:



1. Abra una consola como usuario *lihuery* ejecute el comando:  
`topologia capa-red-estatico start`
2. Aguarde a que aparezcan cada una de las máquinas involucradas en el gráfico. Cada máquina se representa por una ventana xterminal cuyo título se corresponde con el nombre que muestra el gráfico: PC1, PC2, PC3, Router1, Router2 y Router3
3. Configure cada uno de los equipos considerando
  1. Para iniciar sesión en cada equipo, debe utilizar como nombre de usuario *root* y contraseña *xxxx*
  2. Utilice el comando *ifconfig* para configurar las direcciones IP de equipo según las interfaces indicadas en el gráfico. Por ejemplo, en PC3 debe configurar la interfaz eth0 con la IP 192.168.3.2, en Router2 eth0 con la IP 192.168.3.1, eth1 con 200.200.0.1 y eth2 con 200.0.0.254
  3. Cada vez que configure los extremos de un enlace, por ejemplo la interfaz eth0 de PC3 y la interfaz eth0 de Router2, compruebe conectividad utilizando el comando *ping*

## Comando ifconfig.

```
ifconfig [-v] [-a] [-s] [interface]
ifconfig [-v] interface [aftype] options | address ...
```

Permite configurar una tarjeta de red. Si no se le proporcionan parámetros, muestra el estado de las interfaces actualmente *activas*. Si sólo interesa una interfaz, puede indicarse su nombre. Puede indicarse la familia de protocolos a utilizar para una interfaz con el parámetro **aftype** (**inet**: TCP/IP -default-, **inet6**: IPv6, **ipx**: NOVEL IPX...).

**-a** Muestra información de todas las interfaces -incluso inactivas-.

**-s** Lista reducida (igual a **netstat -i**).

**-v** Verbose. Incrementa el nivel de detalle de la salida del programa.

**interface** Nombre de una interfaz, usualmente el driver seguido del número de unidad, como eth0. Si el kernel soporta alias se indican como eth0:0 para el primer alias de eth0 (esto permite asignar una segunda dirección). Puede eliminarse un alias con: **ifconfig eth0:0 down** (si se eliminan el primario, se eliminan también todos los alias).

**up|down** Activa/desactiva la interfaz

**[-]arp** Habilita/Deshabilita el uso del protocolo ARP en la interfaz

**[-]promisc** Habilita/Deshabilita el modo *promiscuo* para la interfaz (permite que la interfaz reciba todos los paquetes en la red, no sólo los que la tienen por destino)

**[-]allmulti** Habilita/Deshabilita el modo all-multicast (permite que la interfaz reciba todos los paquetes multicast en la red)

**metric N** Setea la métrica utilizada por la interfaz

**mtu N** Setea el Maximum Transfer Unit (MTU) de la interfaz

**netmask addr** Setea la máscara de red (por defecto utiliza máscara A, B o C según la dirección IP, pero puede indicarse cualquier valor)

**add|del addr/prefixlen** Agrega/elimina una dirección IPv6 a/de la interfaz

**tunnel aa.bb.cc.dd** Crea un nuevo dispositivo SIT (IPv6-in-IPv4) hacia el destino indicado.

**[-]broadcast [addr]** Setea la dirección de broadcast para la interfaz. Si no se indica la dirección setea/quita el flag IFF\_BROADCAST para la interfaz.

**[-]pointopoint [addr]** Habilita/Deshabilita el modo point-to-point para la interfaz (es un link directo entre dos máquinas sin nadie más escuchando entre ellas). Si se indica la dirección considera que es la que está al otro lado, sino setea/quita el flag IFF\_POINTTOPOINT para la interfaz.

**hw class address** Setea la dirección de hardware para la interfaz. Se debe indicar el nombre de la clase (ether para Ethernet) de hardware y el equivalente ASCII de la dirección.

**multicast** Marca el flag *multicast* en la interfaz. Normalmente no se usa -lo maneja el driver-.

**address** La dirección IP a asignar a la interfaz.

```
pc1: ~# ifconfig eth0 192.168.1.2 netmask 255.255.255.0      #no es precisa la mascara
porque 192... ya es C
```

```
router1:~# ifconfig eth0 192.168.1.1 netmask 255.255.255.0
router1:~# ping 192.168.1.2                                #devuelve 0% packet loss
```

```
pc2: ~# ifconfig eth0 192.168.2.2
router3:~# ifconfig eth2 192.168.2.1
router3:~# ping 192.168.2.2                                #devuelve 0% packet loss
```

```
pc3: ~# ifconfig eth0 192.168.3.2
router2:~# ifconfig eth0 192.168.3.1
```

```
router1:~# ifconfig eth1 200.0.0.1
router1:~# ifconfig eth2 200.100.0.254
router2:~# ifconfig eth1 200.200.0.1
router2:~# ifconfig eth2 200.0.0.254
router3:~# ifconfig eth0 200.100.0.1
router3:~# ifconfig eth1 200.200.0.254
```

**4. Utilice el comando *route* para configurar las rutas estáticas necesarias en cada equipo. En el caso de los routers debe considerar:**

- 1. Router1 envía todo el tráfico desconocido a Router2**
- 2. Router2 envía todo el tráfico desconocido a Router3**
- 3. Router3 envía todo el tráfico desconocido a Router1**

#### **Comando route.**

Manipula/muestra las tablas IP de ruteo estático, según se utilice o no con las opciones add/del.

<b>-n</b>	Utilizar direcciones numéricas (no intenta convertirlas a nombres simbólicos)
<b>-e ee</b>	Controla nivel de detalle de la salida, con un formato como el de netstat
<b>del add</b>	Elimina/Agrega una ruta
<b>target</b>	Red/Host destino (dirección IP o nombre del host)
<b>-net host</b>	Indica que el target es un/a red/host
<b>netmask <i>MM</i></b>	Indica la máscara de red a utilizar si se agrega una ruta de red
<b>gw <i>GW</i></b>	Permite rutear los paquetes vía el gateway indicado. Debe ser alcanzable (por ejemplo por otra ruta estática). Si se indica el nombre de una interfaz local, se la usará para decidir hacia dónde se deberán rutear los paquetes.
<b>reject</b>	Instala una ruta de bloqueo, por lo que fallarán los búsquedas.
<b>dev <i>IF</i></b>	Fuerza que la ruta se asocie a una interfaz, ya que de otro modo el kernel intentará determinar solo por qué interfaz enviar los paquetes.
<b>[If] <i>dev</i></b>	Indica la interfaz que se configura. Es la última opción.

```
router1:~# route add default gw 200.0.0.254 eth1
router2:~# route add default gw 200.200.0.254 eth1
router3:~# route add default gw 200.100.0.254 eth0
pc1:~# route add default gw 192.168.1.1 eth0
pc2:~# route add default gw 192.168.2.1 eth0
pc3:~# route add default gw 192.168.3.1 eth0
```

**5. Si un router no intercambia paquetes entre placas, verifique que los siguientes valores del kernel sean los siguientes:**

- 1. Verificar IP\_FORWARD, este parámetro admite el intercambio de paquetes entre placas:**

- Para obtener el valor:

```
cat /proc/sys/net/ipv4/ip_forward
```

**El valor en 0 deshabilita su funcionalidad. Un 1 lo habilita.**

- Para cambiar el valor:

```
echo 1 > /proc/sys/net/ipv4/ip_forward
```

**2. Verificar RP\_FILTER, este parámetro es de seguridad y evita la recepción de paquetes por una interfaz que tengan una IP de una red diferente a la IP configurada en esa interfaz. Este valor debe deshabilitarse en routers:**

- Para obtener el valor:

```
cat /proc/sys/net/ipv4/conf/all/rp_filter
```

**El valor en 0 deshabilita su funcionalidad. Un 1 lo habilita.**

- Para cambiar el valor:

```
echo 0 > /proc/sys/net/ipv4/conf/all/rp_filter
```

```
router1:~# cat /proc/sys/net/ipv4/ip_forward
0
router1:~# echo 1 > /proc/sys/net/ipv4/ip_forward
router1:~# cat /proc/sys/net/ipv4/conf/all/rp_filter
0
```

```
router2:~# cat /proc/sys/net/ipv4/ip_forward
0
router2:~# echo 1 > /proc/sys/net/ipv4/ip_forward
router2:~# cat /proc/sys/net/ipv4/conf/all/rp_filter
```

0

```
router3: ~# cat /proc/sys/net/ipv4/ip_forward
0
router3: ~# echo 1 > /proc/sys/net/ipv4/ip_forward
router3: ~# cat /proc/sys/net/ipv4/conf/all/rp_filter
0
```

## 6. Verifique conectividad entre PC1, PC2 y PC3:

### 1. Utilizando el comando *ping*

```
pc1: ~# ping 192.168.2.2
PING 192.168.2.2 (192.168.2.2) 56(84) bytes of data.
64 bytes from 192.168.2.2: icmp_seq=1 ttl=62 time=117 ms
64 bytes from 192.168.2.2: icmp_seq=2 ttl=62 time=0.336 ms
...
```

### 2. Utilizando el comando *traceroute*

```
pc1: ~# traceroute 192.168.2.2
traceroute to 192.168.2.2 (192.168.2.2), 30 hops max, 40 byte packets
 1 192.168.1.1 (192.168.1.1) 0.160 ms 0.445 ms 0.115 ms
 2 200.200.0.1 (200.200.0.1) 0.263 ms 0.405 ms 0.200 ms
 3 200.100.0.1 (200.100.0.1) 0.205 ms 0.409 ms 0.181 ms
 4 192.168.2.2 (192.168.2.2) 0.342 ms 0.229 ms 0.200 ms
```

### 3. Utilizando el comando *ping -nR*

```
pc1: ~# ping -nR 192.168.2.2
PING 192.168.2.2 (192.168.2.2) 56(124) bytes of data.
64 bytes from 192.168.2.2: icmp_seq=1 ttl=62 time=0.425 ms
RR:  192.168.1.2
      200.0.0.1
      200.200.0.1
      192.168.2.1
      192.168.2.2
      192.168.2.2
      200.100.0.1
      192.168.1.1
      192.168.2.2
64 bytes from 192.168.2.2: icmp_seq=2 ttl=62 time=0.291 ms    (same
route)
...
```

**4. Mientras realiza ping desde una PC, capture paquetes en un router intermedio y verifique qué paquetes pasan por la interfaz. Por ejemplo, mientras PC1 corre el comando *ping* a la IP de PC2, analice los paquetes que se visualizan en *eth0* y en *eth1* de Router3. La captura de paquetes puede hacerse con el comando *tcpdump -i interfaz* Por ejemplo:**

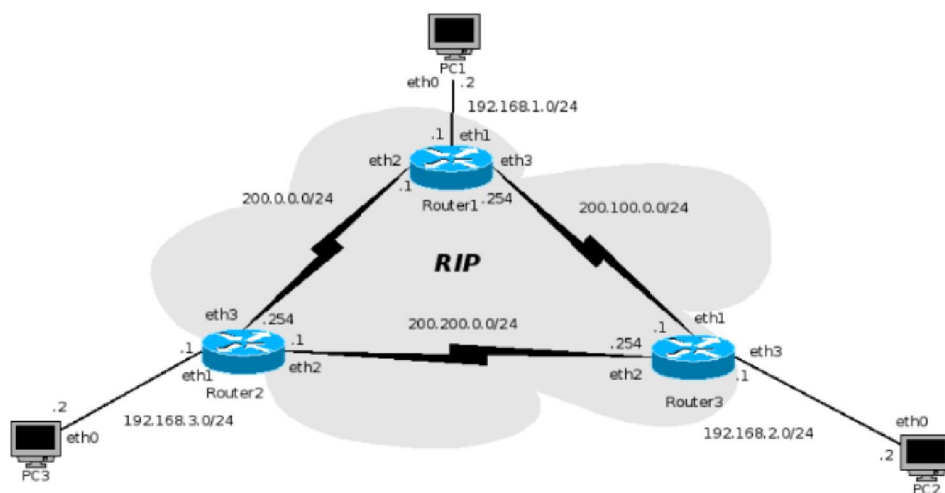
```
tcpdump -i eth0
```

Router 3	PC1
<pre># (tcpdump -vv -i eth0 &gt; eth0.pcap) &amp; [1] 1223 # (tcpdump -vv -i eth1 &gt; eth1.pcap) &amp; [2] 1226  # kill 1223 # kill 1226 # cat eth1.pcap .... 192.168.1.2 &gt; 192.168.2.2: ICMP echo request... # cat eth0.pcap .... 192.168.2.2 &gt; 192.168.1.2: ICMP echo reply...</pre>	<pre># ping 192.168.2.2 PING 192.168.2.2 (192.168.2.2) 56(84) bytes of data. 64 bytes from 192.168.2.2: icmp_seq=1 ttl=62 time=117 ms ... ^C</pre>

5. En base al punto anterior, ¿qué puede deducir?



23. Utilizando el LiveCD provisto por la cátedra, se simulará la red que muestra el siguiente gráfico:



1. Abra una consola como usuario *lihueny* ejecute el comando:  
`topologi a capa-red-di nami co start`
2. Espere a que la consola devuelva el prompt y que aparezcan cada una de las máquinas involucradas en el gráfico. Cada máquina se representa por una ventana *xterminal*/cuyo título se corresponden con los nombres que muestra el gráfico: PC1, PC2, PC3, Router1, Router2 y Router3
3. Cada equipo de la red ya se encuentra configurado y el ruteo es dinámico utilizando RIP. El nombre de usuario de las máquinas es *root* su contraseña *xxxx*.
4. Desde una de las PC compruebe la ruta que siguen los paquetes al resto de las PC. Para verificar las rutas, puede utilizar el comando `ping` o `traceroute`.



```
pc1: ~# ping -nR 192.168.2.2
... RR: 192.168.1.2 / 200.100.0.254 / 192.168.2.1 / 192.168.2.2...
pc1: ~# ping -nR 192.168.3.2
... RR: 192.168.1.2 / 200.0.0.1 / 192.168.3.1 / 192.168.3.2...
```

La ruta de PC1->PC2 cambió desde el ejercicio anterior, usando ahora un camino más directo (PC1->Router1->Router3->PC2).

**5. Dada la ruta obtenida en el punto anterior, daremos de baja uno de los enlaces verificando el funcionamiento del ruteo dinámico. Para ello, debe utilizar el comando `ifconfig INTERFACE down`. Por ejemplo, supongamos que la ruta desde PC1 a PC3 pasa por la red 200.0.0.0/24. Luego, debe dar de baja la interfaz eth2 de Router1 (`ifconfig eth1 down`) y la interfaz eth3 de Router2 (`ifconfig eth3 down`).**

```
router1: ~# ifconfig eth3 down
router3: ~# ifconfig eth1 down
```

**6. Desde la misma PC del punto 4, vuelva comprobar la ruta que siguen los paquetes al resto de las PC.**

```
pc1: ~# ping -nR 192.168.2.2
... RR: 192.168.1.2 / 200.0.0.1 / 200.200.0.1 / 192.168.2.1 / 192.168.2.2...
pc1: ~# ping -nR 192.168.3.2
... RR: 192.168.1.2 / 200.0.0.1 / 192.168.3.1 / 192.168.3.2...
```

## Ejercicios Evaluables

**1. Dada una dirección IP y la máscara correspondiente, deberá saber responder:**

**De qué tipo de dirección se trata (A, B o C).**

**Cuál es la dirección de subred.**

**Cuál es la cantidad máxima de host que pueden estar en esa red.**

**Cuál es la dirección de broadcast.**

**Cuál es el rango de hosts válidos dentro de la red.**

**2. Con el ejercicio 22 resuelto (ruteo estático), copie el gráfico de la topología de red e indique la tabla de ruteo de cada dispositivo de red, tanto routers como PCs de usuarios. Para la confección de la tabla de enrutamiento puede usar el formato del comando “route -n” o “netstat -nr”, aunque alcanza con especificar los siguientes campos: Destino, Gateway, Mascara, Interface.**

**Sobre este gráfico se le realizarán preguntas de comprensión sobre el tema ruteo, como por ejemplo:**

**Si la PC1 le envía un ping a la estación PC2, ¿cuál es el camino por el que viaja el requerimiento?, ¿cuál es el camino por el que viaja la respuesta?**

**El ayudante cambiará alguna de las tablas de ruteo y usted tendrá que evaluar como afecta el cambio realizado.**

## Capa De Enlace

### 1. ¿Qué función cumple la capa de enlace? Indique qué servicios presta esta capa y luego compárelos con los servicios prestados por la capa de transporte.

La Capa de *Enlace* tiene la responsabilidad de transferir datagramas desde un nodo al nodo adyacente a través de un enlace individual (direccionamiento físico), contrastando con la capa de *Transporte* que ofrece comunicación entre hosts remotos.

Define el formato de los "frames" que se intercambian entre los nodos a los extremos del enlace y las acciones que llevan a cabo para transmitir y recibir frames. Ofrece servicios de:

- **Entramado y acceso al medio.** Encapsula al datagrama en el frame (fragmentando de ser necesario). Incluye direcciones MAC -físicas, según el NIC (placa de red)-. Controla el acceso al medio si es compartido, coordinando las transmisiones de los nodos.
- **Transferencia confiable.** Basado en confirmaciones y retransmisiones como TCP. Usado principalmente si el enlace no es confiable para evitar retransmisiones a nivel de red o aplicación - se utiliza en wireless por ejemplo, pero no con fibra óptica, coaxial ni algunas versiones de par trenzado para no sobrecargar duplicando funciones-.
- **Control de flujo.**
- **Detección de Errores.** Producidos por atenuación de la señal o ruido electromagnético. Muy común; se implementa en hardware y es más sofisticado que el realizado por capas superiores.
- **Corrección de Errores** (algunos protocolos; ATM permite la detección sólo en el header).
- **Conexiones Full-Duplex y Half-Duplex.** En las primeras un nodo puede transmitir y recibir a la vez; en las segundas no.

Si bien algunas de estas funciones se "solapan" con las de Transporte, la capa de Enlace las implementa entre nodos adyacentes, no entre hosts remotos.

### 2. Nombre cinco protocolos de capa de enlace. ¿Todos los protocolos en esta capa proveen los mismos servicios?

Ethernet, Token Ring, FDDI, PPP, 802.11 -wireless-... ATM y Frame Relay pueden considerarse protocolos de enlace en determinados contextos.

No todos ofrecen los mismos servicios; la transmisión confiable y la corrección de errores por ejemplo no están presentes en todos.

### 3. Calcule los códigos de detección de error para las siguientes cadenas de bits utilizando paridad par e impar:

- (a) 11010110101001111 Paridad Par: 1 Paridad Impar: 0  
 (b) 01011101011000010 Paridad Par: 0 Paridad Impar: 1  
 (c) 00100010001000111 Paridad Par: 0 Paridad Impar: 1

### 4. Se desea enviar la secuencia de bits 1100000111. Calcular la secuencia completa (datos+FCS) a transmitir considerando que el polinomio generador a utilizar es: $G(x) = x^5 + x^4 + 1$

$G = 110001$  son 6 bits:  $5=r+1$

Se agrega a la secuencia  $6-1 = r=5$  bits en 0

Secuencia: D:1100000111, mas espacio para el resto: 1100000111 00000

Se divide la secuencia por G, y se concatena el resto a la secuencia para la transmisión (se pone en reemplazo de los ceros, lo que equivale a "restar lo que sobró" al hacer la división para forzar que de cero... como se resta con XOR y se habían agregado todos 00000, 00000 XOR RESTO = RESTO).

```

_ 110000011100000    ( - es XOR)
 110001
---
   101110
   110001
---
    111110
    110001
---
     111100

```

$$- \frac{110001}{11010} = R$$

Se transmite la cadena 1100000111 **11010**.

Verificación del CRC: La división de la cadena a transmitir por G debe tener resto 0  
**1100000111 11010**

```

110001
-----
0000010111 1
0000011000 1
-----
0000001111 01
0000001100 01
-----
0000000011 0001
0000000011 0001
-----
0000000000 00000 --> y el resto ES cero!

```

### 5. Encontrar el FCS si se utiliza la función generadora G=110011 y el mensaje M=11100011

```

G = 110011 son 6 bits = r+1      r = 5
M = 11100011 dejando espacio para el resto: 11100011 00000
M' mod G = 1110001100000 mod 110011 = R:
- 1110001100000
  110011
  -----
  101111
- 110011
  -----
  111000
- 110011
  -----
  101100
- 110011
  -----
  111110
- 110011
  -----
  11010 = R

```

Se transmite la cadena 11100011 **11010**.

### 6. Indicar si es verdadero o falso. Justifique su respuesta

(a) Si se utiliza paridad par y se invierte el valor de 2 bits a causa de errores en la transmisión, el receptor detectará el error. FALSO. La inversión de una cantidad par de bits no es detectable bajo ningún esquema de paridad

(b) CRC realiza detección de errores sólo en un bit. FALSO. CRC puede detectar secuencias de errores menores que r+1 bits, y cualquier cantidad impar de bits erróneos.

(c) CRC nunca falla. FALSO. Ráfagas pares de más de r+1 bits no son pares.

(d) 101011 es un valor válido para el polinomio generador. VERDADERO. La única condición necesaria es que comience con 1. Equivale a  $x^5 + x^3 + x + 1$ .

(e) 011111 es un valor válido para el polinomio generador. FALSO. Debe comenzar con 1.

### 7. ¿De qué forma se identifican dos máquinas en una red Ethernet? ¿Qué características poseen estas direcciones?

Las máquinas en una LAN se identifican a nivel de *Enlace* por la dirección LAN/Ethernet/Física/MAC asociada a su adaptador/NIC/placa de red -normalmente grabada en su memoria ROM-. Para Ethernet, utilizan 6 bytes (48 bits), normalmente expresados en hexadecimal.

Estas direcciones son únicas (las administra la IEEE, utilizando 3 bytes para codificar el fabricante y 3 para el número de serie de la placa). Conforman un espacio de direcciones **plano** y no varía por más que la placa se conecte desde distintos lugares (a diferencia de las direcciones IP que forman una jerarquía y varían según la red a la que se conecte el equipo). Existe también una dirección de broadcast: FF-FF-FF-

FF-FF-FF.

Pese a existir las direcciones IP, las MAC son necesarias porque:

- Las LANs no sólo se diseñan para el protocolo de red IP.
- Usar sólo IP requeriría almacenarla en la RAM de la placa y configurarla cada vez que se conecte.
- Si no se usaran direcciones y la capa de Enlace pasara todos los datagramas a la capa superior para que ella filtre, la interrumpiría muy seguido.

## 8. Describa el algoritmo de acceso al medio en Ethernet. ¿Es Ethernet orientado a la conexión?

En Ethernet se utiliza un canal simple de difusión compartida, con acceso aleatorio: si dos nodos transmiten a la vez hay una colisión de la que deben recuperarse. Deben coordinarse utilizando el mismo canal que para transmitir (es *in-line*).

El algoritmo de acceso al medio es **CSMA/CD** (Carrier Sense Multiple Access with Collision Detect). Para transmitir, el adaptador sensa el medio compartido (no utiliza ningún "slot"). Si está ocupado, espera hasta que se libere; si está libre, comienza a transmitir la trama. Durante la transmisión se compara la señal en el medio con la transmitida -otro nodo puede haber sentido el canal libre y comenzado a transmitir, detectándose después de un intervalo por el tiempo de propagación del medio-.

Si se transmite toda la trama sin detectar colisión: se considera transmisión exitosa.

Si se detecta colisión: se aborta la transmisión de la trama, se envía al medio una señal de "taco" (patrón de 48 bits, asegura que todos los nodos detecten la colisión) y se espera antes de volver a intentar retransmitir la señal un tiempo aleatorio que puede ser mayor según la cantidad de colisiones detectadas -**backoff exponencial**-.

Ethernet NO es orientado a conexión (no hay handshaking) y NO es confiable (no hay acks/nacks).

Eficiente con baja carga -habrá pocas colisiones-.

## 9. ¿Qué es la IEEE 802.3? ¿Existen diferencias con Ethernet?

Ethernet es una implementación que se convirtió en estándar *de facto* para LANs con acceso al medio tipo CSMA/CD, definiendo las características de cableado y señalización de nivel físico y los formatos de tramas de datos del nivel de enlace de datos del modelo OSI. Las primeras versiones surgieron en los '70. Se tomó como base, en el '83, para la redacción del estándar formal IEEE 802.3, manejándose usualmente como sinónimos, aunque se diferencian en uno de los campos de la trama de datos (de todos modos, las tramas Ethernet y 802.3 pueden coexistir en la misma red).

Las tecnologías más comunes de tipo Ethernet están agrupadas en el estándar 802.3 (10Base2, 10BaseT, 100BaseT, Gigabit Ethernet...). Se suele hacer referencia a Ethernet como una LAN 802.3.

La diferencia más significativa entre la tecnología Ethernet original y el estándar IEEE 802.3 es la diferencia entre los formatos de sus tramas. Esta diferencia es lo suficientemente significativa como para hacer a las dos versiones incompatibles.

Una de las diferencias entre el formato de las dos tramas está en el preámbulo. El propósito del preámbulo es anunciar la trama y permitir a todos los receptores en la red sincronizarse a sí mismos a la trama entrante. El preámbulo en Ethernet tiene una longitud de 8 bytes pero en IEEE 802.3 la longitud del mismo es de 7 bytes, en este último el octavo byte se convierte en el comienzo del delimitador de la trama.

La segunda diferencia entre el formato de las tramas es en el campo tipo de trama que se encuentra en la trama Ethernet. Un campo tipo es usado para especificar al protocolo que es transportado en la trama (IP, IPX...). Esto posibilita que muchos protocolos puedan ser transportados en la trama. El campo **tipo** fue reemplazado en el estándar IEEE 802.3 por un campo **longitud** de trama, el cual es utilizado para indicar el número de bytes que se encuentran en el campo de datos (puede diferenciarse una trama Ethernet de una 802.3 porque *tipo* toma valores superiores a 1500 y *longitud* valores menores o iguales).

La tercera diferencia entre los formatos de ambas tramas se encuentra en los campos de dirección, tanto de destino como de origen. Mientras que el formato de IEEE 802.3 permite el uso tanto de direcciones de 2 como de 6 bytes, el estándar Ethernet permite solo direcciones de 6 Bytes.

El formato de trama que predomina actualmente en los ambientes Ethernet es el de IEEE 802.3, pero la tecnología de red continua siendo referenciada como Ethernet.

### 10. Describa el algoritmo de acceso al medio empleado por 802.5

En redes **token ring** (estándar 802.5) los N nodos de la LAN se conectan por un enlace directo a un anillo.

Se transmite un frame pequeño llamado "token" por la red. El orden en que pasa el *token* queda definido por la topología. Cuando un nodo obtiene el token vacío puede enviar su frame, que se propaga por el anillo. Al volver el token al nodo emisor, este no lo retransmite, y lo pasa, vacío, al siguiente nodo.

El token se mueve en un solo sentido, así que si un nodo quiere transmitir al anterior, deberá recorrer toda la vuelta.

### 11. ¿Qué es un bridge (o puente)? ¿Y un switch? ¿En qué se diferencian?

Un switch/bridge es un dispositivo de **capa de enlace**, que permite interconectar segmentos de LAN formando **dominios de colisión separados**, pero un mismo dominio de broadcast.

Al recibir una trama por una interfaz, consulta una tabla interna que construye automáticamente con MAC/interfaz, y reenvía la trama por la interfaz adecuada (si es la misma interfaz de entrada, no hace nada; si no estaba en la tabla, hace *flooding* reenviando la trama a todas las interfaces excepto por la que entró).

El **bridge conmuta por software** y está pensado para conmutar entre redes, por lo que tienen pocas interfaces (2). Normalmente manejan de a una trama por vez. Utilizan *store-and-forward* (mantienen buffers internos en los que almacenan las tramas de entrada, retransmitiéndolas recién cuando ingresaron por completo).

El **switch conmuta por hardware** -más rápido- y se pensó para conectarse a hosts directamente. Pueden configurarse en modo *store-and-forward* o *cut-through* (comienza a enviar el frame ni bien reconoce la MAC destino). Pueden manejar múltiples tramas simultáneamente y mantener caminos paralelos.

### 12. ¿Cuál es la función de un HUB? ¿Qué lo diferencia de un switch/bridge?

Un Hub es un dispositivo de la capa física: ingresa una señal por una interfaz, y la distribuye por el resto incrementando normalmente la potencia -actúa como un repetidor-. No interpreta la señal que repite. No realiza almacenamiento y reenvío (no tiene buffers). No implementa CSMA/CD. Permite interconectar segmentos de LAN siempre que utilicen la misma tecnología Ethernet (no puede interconectar 10Base2 con 100BaseT por ejemplo), pero forma un **único dominio de colisión**.

Un switch/bridge trabaja en capa de enlace, tiene buffers y reenvía las tramas basándose en la MAC y una tabla interna, evitando retransmitir por todas las interfaces -salvo que no posea la MAC destino-. Utiliza CSMA/CD y permite interconectar distintas tecnologías ethernet.

### 13. Dado el siguiente esquema de red, responda:

#### i. ¿Quién escucha el mensaje si:

(a) La estación 1 envía una trama al servidor 1

Estaciones 1, 2, 3, 4 y 5, Servidor 1 y Bridge.

(b) La estación 1 envía una trama a la estación 11

Las de (a) más: Estación 11, 12 y 13

(c) La estación 1 envía una trama a la estación 9

Las de (a) más: Switch y Estaciones 8, 9 y 10.

(d) La estación 6 envía una trama a la estación 7

La Estación 7 y el Switch.

(e) La estación 6 envía una trama a la estación 10

Las Estaciones 8, 9 y 10 y el Switch.

#### ii. ¿En qué situaciones se pueden producir colisiones?

Si hosts en un mismo segmento de red transmiten simultáneamente:

- Estaciones 1 a 5 o Servidor 1 transmiten simultáneamente, o mientras llega a su segmento una trama desde el bridge.

- Estaciones 8 a 10 transmiten simultáneamente, o mientras llega a su segmento una trama desde el

switch.

- Estaciones 11 a 13 transmiten simultáneamente, o mientras llega a su segmento una trama desde el bridge.
- Estación 6 o 7 transmiten mientras les llega un mensaje desde el switch.
- Si el bridge envía una trama al switch y éste le envía otra al mismo tiempo.

### iii. Si la estación 5 transmite un broadcast, ¿quiénes escuchan esta trama?

Todos. Como se utilizaron hubs, switches y bridges se trata de un *único dominio de broadcast*. Se precisaría un router para separarlo. {<--IMPORTANTE}

## 14. ¿Qué dispositivos dividen dominios de broadcast? ¿y dominios de colisión?

División de Dominios de Broadcast: router

- División de Dominios de Colisión: router, switch/bridge

Los HUBs no dividen ni dominios de colisión, ni dominios de broadcast por ser de capa física.

### EJEMPLO

<b>7 Dominios de Colisión</b>	<b>3 Dominios de Broadcast</b>

### 15. ¿Cuál es la finalidad del protocolo ARP?

El Protocolo de Resolución de Direcciones (**ARP**) permite la traducción entre los direccionamientos IP de capa de *red* y MAC de capa de *enlace*, **dentro de una misma LAN**. Se implementa con una tabla MAC->IP en cada host, incluyendo un TTL.

Para enviar un datagrama IP de un host a otro, es preciso conocer tanto su dirección IP como su MAC. ARP permite obtener el último, consultando en la tabla. Si no se encuentra en la tabla, hay dos opciones:

- Que el destino sea del mismo segmento/subred: Se averigua la MAC con una consulta ARP al broadcast físico indicando la IP, y el adaptador que reconoce la IP destino responde con su MAC, que es agregada a la tabla del emisor; ó
- Que esté en un segmento de red diferente (al aplicar la máscara de (sub)red, se obtiene una dirección distinta a la del origen): En este caso se envía el frame al gateway, que luego continúa retransmitiéndolo. El datagrama en el interior del frame contiene la IP del destino, pero el frame contiene la MAC del gateway (si no la conoce, utiliza ARP para solicitarla). Al enviarse el frame, es recibido por el router, sube a su capa de *red* y al ver una IP destino distinta de la suya, vuelve a bajar a *enlace* para ser retransmitida al próximo router que corresponda según la tabla de ruteo. Eventualmente un router conocerá al equipo destino, y le enviará el datagrama con la MAC correcta -si no la conoce, utiliza ARP-.

Al utilizar la dirección de broadcast para obtener la información que precisa, ARP es un protocolo *plug-and-play* (no precisa configuración).

## 16. Un host A debe conectarse a un host B. Si A desconoce la dirección MAC de B, qué debería hacer suponiendo las siguientes situaciones:

(a) Las direcciones IP de A y B son 192.23.1.4/24 y 192.23.1.2/24, respectivamente

Host A aplica a IP de B la máscara y obtiene el mismo valor que su red (192.23.1): Los dos hosts están en la misma subred. A prepara un *ARP Request* para obtener la MAC de B, con:

- IP Origen: 192.23.1.4/24, MAC Origen: MAC\_A
- IP Destino: 192.23.1.2/24, MAC Destino: 00:00:00:00:00:00

Luego envía el *Arp Request* en un frame de broadcast:

- MAC Origen: MAC\_A
- MAC Destino: MAC Broadcast (FF:FF:FF:FF:FF:FF).

Una vez que B responda, A agrega la MAC en su tabla ARP y prepara el paquete de datos que debía

enviar.

**(b) Las direcciones IP de A y B son [192.23.1.4/24](#) y [192.23.2.2/24](#), respectivamente**

Host A aplica a IP de B la máscara y obtiene un valor distinto de red (192.23.2): Los dos hosts están en distintas subredes. A no intentará obtener la MAC

1q|+A precisa la MAC del router, consulta la tabla ARP y si no está prepara un *ARP Request* para el router (igual que en el caso anterior, pero con la IP del router que tenga configurada). Una vez con la MAC del router, A prepara un *datagrama IP* con:

- IP Origen: [192.23.1.4/24](#)

- IP Destino: [192.23.1.2/24](#)

a nivel de Enlace, utilizará:

- MAC Origen: MAC\_A

- MAC Destino: MAC\_Router

**17. ¿Qué tipo de servicio ofrece Frame Relay? ¿Que ventaja tiene respecto de un enlace dedicado?**

Frame Relay ofrece **retransmisión de tramas** para redes de circuito virtual, con SVC (circuitos virtuales conmutados) o PVC (circuitos virtuales permanentes) respecto a un enlace dedicado este no ofrece ancho de banda compartido mientras que Frame Relay sí con sus circuitos virtuales.

**18. ¿Con FR las tramas llegan en orden al destinatario?**

Si usan todas la misma trama del circuito virtual, a menos que este se rompa todas van a llegar en el mismo orden.

**19. ¿Qué son los ace?**

*Data Link Connection Identifier* (DLCI) es el identificador de canal del circuito establecido en Frame Relay. Este identificador se aloja en la trama e indica el camino a seguir por los datos, es decir, el circuito virtual establecido.

El DLCI puede valer normalmente entre 0 y 1023 (10 bits), los valores del 0 al 15 y del 992 en adelante están reservados para funciones especiales.

El DLCI tiene significado local, es decir, en el circuito virtual cada extremo puede tener un identificador de circuito diferente para identificar el mismo circuito.

**20. ¿Qué es el CIR? ¿para qué se utiliza el bit DE?**

El CIR (Committed Information Rate) es la *velocidad media de transmisión* de cada Circuito Virtual. En frame relay se dedica la tasa de transmisión especificada en CIR.

DE es un bit de prioridad 0 = alta; 1 = baja. Este bit "de desecho" permite seleccionar, cuando la red se "satura", a los que tienen este bit en 1 para descartarlos primero.

El volumen de tráfico alcanzable transmitiendo a la velocidad media CIR se denomina Bc; si se transmite más de este volumen, las tramas *extras* son marcadas con DE en 1 para que se consideren desechables en caso de congestión.

**21. ¿Qué funcionalidad el estándar de PPP requiere que este implemente, y qué funcionalidad requiere que no se implemente?**

**PPP (Point-to-Point Protocol)** es un protocolo para enlaces *punto a punto*, con un host en cada extremo (a diferencia de los *enlaces de broadcast* donde el medio puede ser compartido por múltiples hosts, y los marcos que cada uno envíe son visibles por todos, presentando el *problema del acceso múltiple*).

El estándar de PPP requiere:

- Packet framing: debe crear los frames encapsulando a los paquetes de capa de red
- Transparencia: no debe imponer límites al contenido de los paquetes de capa de red (limita por ejemplo el uso indiscriminado de patrones de bits para identificar inicio/fin de frames; PPP utiliza un patrón para este fin, pero utiliza también un patrón de "escape" que antepone al primero si aparece en el contenido del frame -por ejemplo, en el datagrama encapsulado-. Si el patrón de escape debe aparecer en el contenido, se lo duplica)

- Permitir múltiples protocolos de capa de red al mismo tiempo (esto hace necesario un campo de "tipo de protocolo" en la cabecera PPP para poder multiplexar).
- Múltiples tipos de tecnologías de enlaces -seriales/paralelas, síncronas/asíncronas, de alta/baja velocidad...)
- Detección de errores en los frames.
- Informar a la capa de red de fallas a nivel enlace
- Mecanismo para la negociación de direcciones de red (se utiliza IPCP -IP Control Protocol- en el caso de IP).
- Simplicidad.

Las características que NO requiere (apuntando a la simplicidad):

- Corrección de errores.
- Control de flujo.
- Secuenciamiento (entrega de frames en el mismo orden que se enviaron).
- Enlaces multipunto (sólo se precisa soportar un único emisor y un único receptor).

## 22. ¿Qué son PAP y CHAP?

CHAP (*Challenge Handshake Authentication Protocol*) es un método de autenticación usado por servidores accesibles vía PPP. Verifica periódicamente la identidad del cliente remoto usando un intercambio de información de tres etapas:

1. Después del establecimiento del enlace, el agente autenticador manda un mensaje que "desafía" al usuario.
2. El usuario responde con un valor calculado usando una función hash de un sólo sentido, como la suma de comprobación MD5.
3. El autenticador verifica la respuesta con el resultado de su propio cálculo de la función hash. Si el valor coincide, el autenticador informa de la autenticación, de lo contrario terminaría la conexión.

A intervalos aleatorios el autenticador manda un nuevo "desafío" con lo que se repite el proceso.

PAP (Password Authentication Protocol) transmite contraseñas o password en ASCII sin cifrar, por lo que se considera inseguro. Se usa como último recurso cuando el servidor de acceso remoto no soporta un protocolo de autenticación más fuerte. Los identificadores de protocolo están especificados en el RFC 1661. Los más importantes son:

- 0x0021 para IP.
- 0xc021 para LCP.
- 0xc023 para PAP.
- 0xc223 para CHAP.

## 23. Utilizando el LiveCD provisto por la cátedra, se simulará la red que muestra el siguiente gráfico:

**i. Abra una consola de usuario y ejecute el comando:**

**topología capa-enlace start**

**ii. Aguarde a que la consola devuelva el prompt y que aparezcan cada una de las máquinas involucradas en el gráfico. Cada máquina se representa por una ventana xterminal cuyo título se corresponde con los nombres que muestra el gráfico: PC1\_HUB, PC2\_HUB, PC3\_HUB, PC1\_SW, PC2\_SW, PC3\_SW y Router.**

**iii. Cada equipo de la de red ya se encuentra configurado con sus respectivas direcciones IP y el nombre de usuario de las máquinas y del router es root y su contraseña xxxx**

**iv. Para observar cómo se comportan el hub y el switch realice las siguientes tareas:**

**(a) Envíe un ping desde la PC1\_HUB a la PC2\_HUB y monitoree el tráfico desde la PC3\_HUB**



**utilizando el siguiente comando `tcpdump -i eth0 -p icmp`. Vea los resultados en la consola de PC3\_HUB. ¿Qué pudo observar?**

Al estar conectadas a través de un HUB, PC3\_hub puede ver el echo request de PC1\_hub a PC2\_hub:

```
pc3_hub:~# tcpdump -i eth0 -p icmp
pc1_hub:~# ping 192.168.100.3
pc3_hub:
16:39:43.059902 IP 192.168.100.2 > 192.168.100.3: ICMP echo request,
id 47876, seq 2, length 64
```

**(b) Envíe un ping desde la PC1\_SW a la PC2\_SW y monitoree el tráfico desde la PC3\_SW utilizando el siguiente comando `tcpdump -i eth0 -p icmp`. Vea los resultados en la consola de PC3\_HUB. ¿Qué pudo observar? ¿Cuáles son las diferencias respecto a lo observado en el punto (a)?**

Al estar conectadas a través de un SWITCH, PC3\_sw NO puede ver el echo request de PC1\_sw a PC2\_sw. Esto es porque el hub trabaja en capa física y hace broadcast de todo lo que recibe por todas las interfaces, mientras que el switch utiliza la tabla ARP para sólo reenviar por la interfaz adecuada.

```
pc3_sw:~# tcpdump -i eth0 -p icmp
pc1_sw:~# ping 192.168.200.3
pc3_sw:0 packets captured
```

**v. Para analizar los paquetes del protocolo ARP realice las siguientes tareas:**  
**(a) Ejecute el comando `ifconfig -a` en la PC1\_HUB.**

`ifconfig -a` muestra todas las interfaces de red, estén activas o no.

**(b) Luego ejecute el comando `arp -n` en la PC1\_HUB para ver su tabla ARP.**

`arp -n` muestra la tabla ARP del host. En PC1\_hub está vacía.

**(c) Monitoree el tráfico arp desde la PC3\_HUB ejecutando `tcpdump -i eth0 -p arp`.**

`tcpdump -i eth0 -p arp` captura desde PC3\_hub todo el tráfico de paquetes del protocolo ARP.

**(d) Envíe un ping desde la PC1\_HUB a la PC2\_HUB y vuelva a observar la tabla ARP de la PC1\_HUB.**

```
pc1_hub:~# ping 192.168.100.3
pc1_hub:~# arp -n
Address                HWtype  HWaddress           Flags Mask  Iface
192.168.100.3          ether    FE:FD:00:00:03:00  C                   eth0
```

**(e) Vea los resultados en la consola de PC3\_HUB a fin de observar las características de los paquetes arp (MAC Origen, MAC Destino, etc).**

Durante el ping de PC1\_hub a PC2\_hub, PC3\_hub capturó:

```
16:57:53.717759 arp who-has 192.168.100.3 tell 192.168.100.2
16:57:53.717770 arp reply 192.168.100.3 is-at fe:fd:00:00:03:00 (oui
```

```
Unknown)
16:57:53.750098 arp who-has 192.168.100.1 tell pc3_hub
16:57:53.750308 arp reply 192.168.100.1 is-at fe:fd:00:00:01:00 (oui
Unknown)
16:57:53.726469 arp who-has 192.168.100.2 tell 192.168.100.3
16:57:53.726472 arp reply 192.168.100.2 is-at fe:fd:00:00:02:00 (oui
Unknown)
...
```

**(f) Monitoree el tráfico arp desde la PC3\_SW ejecutando tcpdump -i eth0 -p arp.**

```
pc1_sw:~# arp -n #la tabla está vacía
pc3_sw:~# tcpdump -i eth0 -p arp
```

**(g) Haga un ping a la PC2\_SW y vuelva a observar la tabla ARP de la PC1\_SW.**

```
pc1_sw:~# ping 192.168.200.3
pc1_sw:~# arp -n
Address                HWtype  HWaddress           Flags Mask  Iface
192.168.200.3          ether    FE:FD:00:00:06:00  C           eth0
pc2_sw:~# arp -n
Address                HWtype  HWaddress           Flags Mask  Iface
192.168.200.2          ether    FE:FD:00:00:05:00  C           eth0
```

**(h) Vea los resultados en la consola de PC3\_HUB a fin de observar cuáles son las diferencias respecto a lo observado en el punto (e) en cuanto a cuáles son los paquetes que se ven en este caso.**

Durante el ping de PC1\_sw a PC2\_sw, PC3\_sw capturó:

```
17:08:25.077775 arp who-has 192.168.200.3 tell 192.168.200.2
17:08:25.108919 arp who-has 192.168.200.1 tell pc3_sw
17:08:25.109121 arp reply 192.168.200.1 is-at fe:fd:00:00:01:01 (oui
Unknown)
...
```

como puede observarse, PC3\_sw puede ver el "who-has" o arp-request de PC1\_sw a PC2\_sw, pero no puede ver la respuesta

**vi. Para analizar el encapsulamiento a nivel de capa 2 y 3 realice las siguientes tareas:**

**(a) Ejecute el comando ifconfig -a en la Router y en la PC1\_SW.**

ifconfig -a en router y PC1\_sw. Puede verse en la interface eth0 del router su MAC: FE:FD:00:00:01:00 y su IP: 192.168.100.1. PC1\_sw tiene eth0 con MAC FE:FD:00:00:05:00 e IP: 192.160.200.2.

**(b) Monitoree el tráfico desde la PC3\_HUB ejecutando tcpdump -i eth0.**

```
pc3_hub:~# tcpdump -i eth0 # no se filtra por ningún protocolo en
particular
```

**(c) Luego ejecute el comando arp -n en la PC1\_HUB para ver su tabla ARP.**

```
pc1_hub:~# arp-n #muestra que la tabla está vacía (si bien
cargó datos anteriormente, se borraron al vencer su TTL)
```

**(d) Envíe un ping desde PC1\_HUB a la PC1\_SW y vuelva a observar la tabla ARP de la PC1\_HUB. ¿A qué dispositivo corresponde la asociación IP-MAC agregada en la tabla?**

```
pc1_hub:~# ping 192.168.200.2
```

```
pc1_hub:~# arp -n
```

Address	HWtype	HWaddress	Flags	Mask	Iface
192.168.100.1	ether	FE:FD:00:00:01:00	C		eth0

Como puede verse, PC1\_hub agregó a su tabla ARP la asociación IP-MAC correspondiente al router -no incluyó datos de PC1\_sw porque forma parte de otra red, de modo que se comunicará con ella a través de su default gateway: el router-. Puede comprobarse que el router está correctamente configurado como default gateway en PC1\_hub con:

```
pc1_hub:~# route
```

Kernel IP routing table					
	Destination	Gateway	Genmask	Flags	Metric
Ref	Use	Iface			
	192.168.100.0	*	255.255.255.0	U	0
0	0	eth0			
	default	192.168.100.1	0.0.0.0	UG	0
0	0	eth0			

**(e) Vea los resultados en la consola de PC3\_HUB a fin de observar tanto los paquetes arp como los paquetes icmp teniendo en cuenta MAC origen, MAC destino, IP origen e IP destino según corresponda.**

En la consola de PC3\_hub el tcpdump capturó:

```
17:23:09.850797 arp who-has 192.168.100.1 tell 192.168.100.2
```

```
17:23:49.972334 arp reply 192.168.100.1 is-at fe:fd:00:00:01:00
```

```
(oui Unknown)
```

```
17:23:850858 IP 192.168.100.2 > 192.168.200.2: ICMP echo request,  
id 55812, seq 1, length 64
```

```
17:23:871518 IP 192.168.200.2 > 192.168.100.2: ICMP echo reply, id  
55812, seq 1, length 64
```

```
...
```

Puede verse que: PC1\_hub solicita la MAC de su default gateway y éste le responde. Luego PC1\_hub envía el echo request a PC1\_sw y esta le responde

**Nota: para finalizar la captura de paquetes resultante del comando tcpdump ejecute ctrl.+C.**

## Ejercicios Entregables

**1. Para la siguiente topología de red indique:**

**1. ¿Cuántos dominios de colisión hay?**

Hay **5** dominios de colisión:

- PC A-switch 2,
- PC B-switch 2,
- PC E-switch 2,
- switch 2-switch 1, y
- switch 1-Hub-PC C-PC D

**2. ¿Cuántos dominios de broadcast hay?**

Sólo **uno** (no se utilizaron routers).

**3. Indique cómo se va llenando la tabla de asociaciones MAC PORT del switch SW1 y SW2 durante el siguiente caso:**

**(a) A envía una solicitud ARP consultando la MAC de C. SW2 agrega a MAC PC A en port #1.**

SW1 agrega a MAC PC A en port #1.

**(b) C responde esta solicitud ARP.** SW1 agrega a MAC PC C en port #2. SW 2 agrega a MAC PC C en port #8.

**(c) A envía una solicitud ARP consultando la MAC de B.** (no se alteran las tablas, excepto que pase un tiempo entre esta solicitud y la anterior y se haya quitado la entrada de PC A de las tablas por vencerse su TTL, en cuyo caso se agrega la misma información que en el paso (a))

**(d) B responde esta solicitud ARP.** SW2 agrega a MAC PC B en port #2.

**4. Si la PC E y la PC D hubiesen estado realizando un tcpdump para escuchar todo lo que pasa por su interfaz de red, ¿Cuáles de los requerimientos/respuestas anteriores hubiesen escuchado cada una?**

PC E hubiese escuchado sólo las solicitudes ARP que hizo PC A -(a) y (c)-.

PC D hubiese escuchado la solicitud ARP de PC A por la MAC de PC C (a), y la respuesta de PC C (b). También hubiese escuchado la solicitud ARP por la MAC de PC B (c), pero no hubiese escuchado la respuesta.

## **2. En la siguiente topología:**

**Suponiendo que todas las tablas ARP están vacías, tanto de PCs como de Routers. Si la PC\_A le hace un ping a la PC\_C, indique:**

**¿Qué protocolos de capa 2 aparecen? ¿En qué dominios de broadcast?**

Hay 4 dominios de Broadcast:

- 1 - del router 1 hacia el switch1/PC\_A/PC\_B
- 2 - del router 1 al router 2
- 3 - del router 2 al router 3
- 4 - del router 3 hacia el switch2/PC\_C

Ethernet y ARP -capa 2- aparece en todos.

**¿Qué protocolos de capa 3 aparecen? ¿En qué dominios de broadcast?**

ICMP e IP -capa 3 ambos (ICMP utiliza IP como si fuese de nivel superior aunque se considera también de nivel 3)- aparece en todos.

**¿Cuál es la secuencia correcta en la que se suceden los anteriores?**

- PC A hace ARP Request para obtener la MAC de router 1.
- Router 1 envía su MAC con ARP response.
- PC A envía datagrama ICMP con IP destino [190.26.12.65/24](#) con MAC destino 7777:7777:7777.
- Router 1 hace ARP Request para obtener MAC de router 2.
- Router 2 envía su MAC en ARP Response.
- Router 1 envía el datagrama ICMP cambiando la MAC origen por la suya propia y la MAC destino a cccc:cccc:cccc
- Router 2 hace un ARP Request para obtener la MAC de Router 3.
- Router 3 envía su MAC en ARP Response.
- Router 2 envía el datagrama ICMP cambiando la MAC origen por la suya propia y la MAC destino a 4444:4444:4444
- Router 3 detecta que la IP destino está en su misma subred. Hace un ARP request para obtener la MAC de PC\_C.
- PC\_C responde con su MAC en un ARP response.
- Router 3 envía el datagrama modificando la MAC origen por la suya propia y la MAC destino a 5555:5555:5555

**Para los paquetes del protocolos de capa 3 que haya identificado:**

**Especifique las direcciones (origen/destino) de capa 2 en los distintos dominios de broadcast.**

**Especifique las direcciones (origen/destino) de capa 3 en los distintos dominios de broadcast.**

Protocolo	Tipo	Dominio de Broadcast	MAC Origen	MAC Destino	IP Origen
ICMP	ECHO request	1	8888:8888:8888	7777:7777:7777	<a href="#">190.26.11.2</a>
ICMP	ECHO request	2	bbbb:bbbb:bbbb	cccc:cccc:cccc	<a href="#">190.26.11.2</a>
ICMP	ECHO request	3	eeee:eeee:eeee	dddd:dddd:dddd	<a href="#">190.26.11.2</a>
ICMP	ECHO request	4	4444:4444:4444	5555:5555:5555	<a href="#">190.26.11.2</a>
ICMP	ECHO reply	4	5555:5555:5555	4444:4444:4444	<a href="#">190.26.12.0</a>
ICMP	ECHO reply	3	dddd:dddd:dddd	eeee:eeee:eeee	<a href="#">190.26.12.0</a>
ICMP	ECHO reply	2	cccc:cccc:cccc	bbbb:bbbb:bbbb	<a href="#">190.26.12.0</a>
ICMP	ECHO reply	1	7777:7777:7777	8888:8888:8888	<a href="#">190.26.12.0</a>

3. Utilice la calculadora para convertir su número de DNI a binario. Por ejemplo para el DNI: 12345678 se corresponde el número binario: 10111000110000101001110. Suponiendo que este es el mensaje a enviar, calcular la secuencia completa (datos+FCS) a transmitir considerando que el polinomio generador a utilizar es:  $G(x) = x^5 + x^4 + 1$ .

Capalbo Lucas (9196/2) DNI: 30344365 <sub>(10)</sub> = 1110011110000010010101101 <sub>(2)</sub>

G = 110001

r = 5

1110011110000010010101101 00000

110001

100011

110001

100101

110001

101000

110001

110010

110001

110001

110001

00101011

110001

110100

110001

1011 00

1100 01

111 010

110 001

1 01100

1 10001

11101 = R

Se transmitirá: 1110011110000010010101101 11101 (por Capalbo Lucas, DNI 30344365)

Ivan Muller 8871/0 DNI : 32791812 <sub>(10)</sub> = 1100100000101111000100 <sub>(2)</sub>

G = 110001

r = 5

1111101000101110100000100 00000

110001

00111110

110001

- 00111100

110001

00110110

110001

000111111

110001

00111001

110001

00100000

110001

0100010

110001

0100110

110001

0101110

110001

0111111

110001

00111000file:///Users/Ivan/Downloads/mbu+xiQH.dmg.part

110001

001001 00

1100 01

0101 010

110 001

011 0110

11 0001

00 01110 = R

Se transmitirá: 1111101000101110100000100 01110 (por Ivan Muller DNI 32791812)

**4. Cuando una PC que esta en una red, se quiere comunicar con otra que no esta en la misma red, esta se da cuenta observando su tabla de rutas. Por ende, para comunicarse debe usar el default gateway de la misma. Si la tabla ARP de la PC esta vacía, cuando la PC realiza un ARP para obtener la MAC del router, ¿que dirección IP destino tiene el requerimiento ARP? ¿es la dirección IP del default gateway o la dirección IP de la maquina destino?**

**Adjunte una captura de tráfico que corrobore su respuesta. Esta captura debe acompañar a la entrega del presente informe.**

El requerimiento ARP en este caso tiene como IP destino a la IP del router (que es el dispositivo del que se precisa obtener la MAC).

Esto se realizó en la práctica, en el ejercicio 23, punto vi, inciso e. Se realizó un ping desde *PC1\_hub* hasta *PC1\_sw*, cuyas redes se encontraban comunicadas por un router con IP 192.168.100.1; durante el ping se realizó la captura del tráfico con el comando tcpdump -i eth0 desde PC3\_hub, obteniendo:

17:23:09.850797 arp who-has 192.168.100.1 tell 192.168.100.2

17:23:49.972334 arp reply 192.168.100.1 is-at fe:fd:00:00:01:00

(oui Unknown)

17:23:850858 IP 192.168.100.2 > [192.168.200.2](#): ICMP echo request, id 55812, seq 1, length 64

17:23:871518 IP 192.168.200.2 > [192.168.100.2](#): ICMP echo reply, id 55812, seq 1, length 64

...

Puede verse que *PC1\_hub* solicita la MAC de su *default gateway* y éste le responde. Obtenida la MAC, *PC1\_hub* envía el *echo request* a *PC1\_sw* y esta le responde. La MAC de *PC1\_sw* nunca llega a *PC1\_hub* (ya que cambian las MAC's de segmento a segmento; tampoco llega la MAC de *PC1\_hub* a *PC1\_sw*).