Práctica 5

Capa de Transporte - Parte I

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

La capa de transporte es responsable de la trasferencia de extremo a extremo de los datos de aplicación. También realiza las siguientes funciones: Permitir múltiples aplicaciones para comunicarse a través de la red al mismo tiempo en un solo dispositivo, asegurar, si se quiere, que los datos sean recibidos de manera confiable y en orden, y emplear mecanismos de manejo de error. También debe cumplir: Seguimiento de la comunicación individual entre aplicaciones en los hosts origen y destino, segmentación de datos y gestión de cada porción, reensamblaje de segmentos en flujos de datos de aplicación, identificación de las diferentes aplicaciones y multiplexación de conversaciones.

Una conversación es cada conjunto de secciones en particular que fluyen desde una aplicación origen a una de destino.
-Segmentación y reensamblaje à Divide los datos de aplicación en bloques de datos de un tamaño adecuado que pueda incluirse en una única PDU (Unidad de datos de protocolo).
-Multiplexación à Permite ejecutar varias aplicaciones o servicios en un único host. Se identifica mediante el puerto.
-Algunos protocolos de transporte también proveen: Conversaciones orientados a la conexión (asegura que la aplicación esta lista para recibir datos), entrega confiable (sino vuelve a reenviar), reconstrucción ordenada de datos (numera y secuencia segmentos) y control de flujo (evita saturación en el host).
-Entrega confiable à Confiabilidad significa asegurar que los datos que envía el origen llega al destino. En trasporte las tres operaciones básicas de confiabilidad son: Seguimiento de datos transmitidos, acuse de recibo de los datos recibidos y retransmisión de datos sin acuse de recibo. El origen debe mantener el seguimiento de todas las porciones de datos sin acuse de recibo del destino.

2. Describa la estructura del segmento TCP y UDP.

TCP
•Puerto origen (16 bits): Identifica el puerto emisor.
•Puerto destino (16 bits): Identifica el puerto receptor.

Estos dos valores identifican la aplicación receptora y la emisora, junto con las direcciones IP del emisor y receptor identifican de forma unívoca cada conexión. La combinación de una dirección IP y un puerto es llamado socket. Es el par de sockets (dirección IP + puerto del emisor y dirección IP+ puerto del receptor) emisor y receptor el que especifica los dos puntos finales que unívocamente se corresponden con cada conexión TCP en internet.

•Número de secuencia (32 bits): Identifica el byte del flujo de datos enviado por el emisor TCP al receptor TCP que representa el primer byte de datos del segmento.

Cuando una conexión está siendo establecida el flag SYN se activa y el campo del número de secuencia contiene el ISN (initial sequence number) elegido por el host para esa conexión. El número de secuencia del primer byte de datos será el ISN+1 ya que el flag SYN consume un número de secuencia.

•Número de acuse de recibo (32 bits): Contiene el valor del siguiente número de secuencia que el emisor del segmento espera recibir.

Una vez que la conexión ha sido establecida, este número se envía siempre y se valida con el flag ACK activado. Enviar ACKs no cuesta nada ya que el campo de acuse de recibo siempre forma parte de la cabecera, al igual que el flag ACK. TCP se puede describir como un protocolo sin asentimientos selectivos o negativos ya que el número de asentimiento en la cabecera TCP significa que se han recibido correctamente los bytes anteriores pero no se incluye ese byte.
No se pueden asentir partes selectivas del flujo de datos (suponiendo que no estamos usando la opción SACK de asentimientos selectivos).

•Longitud de cabecera (4 bits): especifica el tamaño de la cabecera en palabras de 32 bits.
Es requerido porque la longitud del campo “opciones” es variable. Por lo tanto el tamaño máximo de la cabecera está limitado a 60 bytes, mientras que sin “opciones” el tamaño normal será de 20 bytes. A este campo también se le suele llamar “data offset” por el hecho de que es la diferencia en bytes desde el principio del segmento hasta el comienzo de los datos.

•Reservado (3 bits): para uso futuro. Debe estar a 0.
•Flags (9 bits):

- NS (1 bit): ECN-nonce concealment protection. Para proteger frente a paquetes accidentales o maliciosos que se aprovechan del control de congestión para ganar ancho de banda de la red.
- CWR (1bit): Congestion Window Reduced. El flag se activa por el host emisor para indicar que ha recibido un segmento TCP con el flag ECE activado y ha respondido con el mecanismo de control de congestión.
- ECE (1 bit): Para dar indicaciones sobre congestión.
- URG (1 bit): Indica que el campo del puntero urgente es válido.
- ACK (1 bit): Indica que el campo de asentimiento es válido. Todos los paquetes enviados después del paquete SYN inicial deben tener activo este flag.
- PSH (1 bit): Push. El receptor debe pasar los datos a la aplicación tan pronto como sea posible, no teniendo que esperar a recibir más datos.
- RST (1 bit): Reset. Reinicia la conexión, cuando falla un intento de conexión, o al rechazar paquetes no validos.
- SYN (1 bit): Synchronice. Sincroniza los números de secuencia para iniciar la conexión.
- FIN (1 bit): Para que el emisor (del paquete) solicite la liberación de la conexión.

•Tamaño de ventana o ventana de recepción (16 bits): Tamaño de la ventana de recepción que especifica el número máximo de bytes que pueden ser metidos en el buffer de recepción o dicho de otro modo, el número máximo de bytes pendientes de asentimiento. Es un sistema de control de flujo.
•Suma de verificación (16 bits): Checksum utilizado para la comprobación de errores tanto en la cabecera como en los datos.
•Puntero urgente (16 bits): Cantidad de bytes desde el número de secuencia que indica el lugar donde acaban los datos urgentes.
•Opciones: Para poder añadir características no cubiertas por la cabecera fija.
•Relleno: Se utiliza para asegurarse que la cabecera acaba con un tamaño múltiplo de 32 bits.

UDP

+	Bits 0 - 15	16 - 31
0	Puerto origen	Puerto destino
32	Longitud del Mensaje	Suma de verificación
64	Datos	

La cabecera UDP consta de 4 campos de los cuales puerto origen y checksum son opcionales (con fondo rojo en la tabla). Los campos de los puertos origen y destino son campos de 16 bits que identifican el proceso de emisión y recepción. Ya que UDP carece de un servidor de estado y el origen

UDP no solicita respuestas, el puerto origen es opcional. En caso de no ser utilizado, el puerto origen debe ser puesto a cero. A los campos del puerto destino le sigue un campo obligatorio que indica el tamaño en bytes del datagrama UDP incluidos los datos. El valor mínimo es de 8 bytes. El campo de la cabecera restante es una suma de comprobación de 16 bits que abarca una pseudo-cabecera IP (con las IP origen y destino, el protocolo y la longitud del paquete UDP), la cabecera UDP, los datos y 0's hasta completar un múltiplo de 16. El checksum también es opcional en IPv4, aunque generalmente se utiliza en la práctica (en IPv6 su uso es obligatorio). A continuación se muestra los campos para el cálculo del checksum en IPv4, marcada en rojo la pseudo-cabecera IP.

bits	0 – 7	8 – 15	16 – 23	24 – 31
0	Dirección Origen			
32	Dirección Destino			
64	Ceros	Protocolo	Longitud UDP	
96	Puerto Origen		Puerto Destino	
128	Longitud del Mensaje		Suma de verificación	
160	Datos			

El protocolo UDP se utiliza por ejemplo cuando se necesita transmitir voz o vídeo y resulta más importante transmitir con velocidad que garantizar el hecho de que lleguen absolutamente todos los bytes.

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

La combinación de una dirección IP y un puerto es llamado socket. Es el par de sockets (dirección IP + puerto del emisor y dirección IP+ puerto del receptor) emisor y receptor el que especifica los dos puntos finales que unívocamente se corresponden con cada conexión TCP en internet.

La interfaz socket define las reglas que un programa ha de seguir para utilizar los servicios del nivel de transporte en una red TCP/IP. Dado que en un mismo dispositivo/ordenador podemos estar ejecutando de forma simultánea diferentes aplicaciones que utilizan Internet para comunicarse, resulta imprescindible identificar cada socket con una dirección diferente. Un socket se va a identificar por la dirección IP del dispositivo, más un número de puerto (de 16 bits). En Internet se suele asociar a cada aplicación un número de puerto concreto (por ejemplo: 80 para la web, 25 para el correo electrónico o 7 para ECHO). Una conexión está determinada por un par de sockets, uno en cada extremo de la conexión. Existen dos tipos de socket: socket stream y socket datagram.

4. Compare TCP y UDP en cuanto a:

a. Confiabilidad.
UDP no es un sistema de transferencia de datos confiables ya que no garantiza que los datos enviados por un proceso lleguen intactos. Algunas aplicaciones utilizan UDP ya que son tolerantes a

pérdidas de cantidades pequeñas de paquetes. En cambio, TCP si puede garantizar que los datos enviados lleguen intactos ya que es un servicio de transferencia de datos fiable.

b. Multiplexación.
Ambos UDP y TCP realizan tareas de multiplexación ya que es uno de los servicios que deben implementar los protocolos de la capa de transporte. La multiplexación es la única característica que tienen en común TCP y UDP.

c. Orientado a la conexión.
TCP lleva a cabo un proceso de establecimieto de la conexión en tres fases antes de iniciar la transferencia de datos. UDP inicia la transmisión de manera inmediata, sin formalidades preliminares. UDP no añade ningún retardo adicional a causa del establecimiento de una conexión, es por eso que DNS opera sobre UDP y no TCP.

d. Controles de congestión.
TCP si brinda mecanismos de control de congestión, que evitan que cualquier conexión TCP inunde con una cantidad de tráfico excesiva los enlaces y routers existentes entre los hosts que están comunicándose. UDP no proporciona mecanismos de control de congestión, puesto que es más simple y más apropiado a las aplicaciones en tiempo real (como comunicaciones o multimedia) las cuales no responden demasiado bien a estos mecanismos. El tráfico UDP, por el contrario de TCP, no está regulado.

e.Utilización de puertos.
*Ambos protocolos utilizan puertos para poder intercambiar los datos entre distintos procesos.

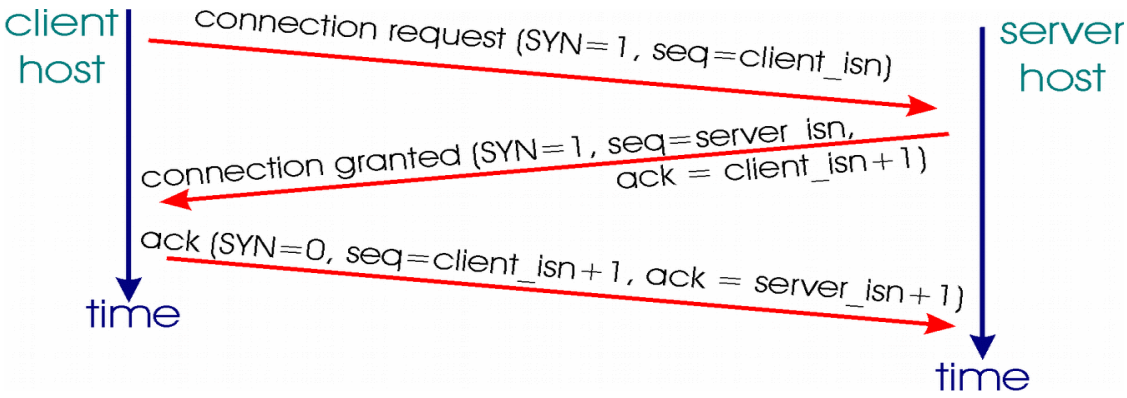
f. ¿Cuál es el campo del datagrama IP y los valores que se utilizan en este para diferenciar que se transporta TCP o UDP? (Ayuda: buscar en /etc/protocols y contrastarlo con una captura de tráfico)
En el datagrama IP el protocolo usado en la capa de transporte se indica en el campo Protocolo que tiene 8 bits de lognitud. Indica el protocolo de las capas superiores al que debe entregarse el paquete.
Para indicar que se envia un segmento TCP se envia el valor 6, y para indicar que es un datagrama UDP se usa el 17.

5. La PDU de la capa de transporte es el segmento. Sin embargo, en algunos contextos suele utilizarse el término datagrama. Indique cuando.

La PDU del UDP se conoce como un datagrama, aunque los términos segmento y datagrama se utilizan algunas veces de forma intercambiable para describir una PDU de la capa de transporte.
En los RFC los paquetes que manejan tanto TCP como UDP son los segmentos. Sin embargo no es poco común encontrar que en algunos textos mencionan como segmentos sólo a los paquetes TCP y datagrama a los paquetes UDP.

6. Describa el saludo de tres vías de TCP.

Para establecer una conexión TCP, el proceso de aplicación cliente informa en primer lugar al cliente TCP que desea establecer una conexión con un proceso servidor. A continuación, el protocolo TCP en el cliente establece una conexión TCP con el protocolo TCP en el servidor de la siguiente manera:
1. TCP del lado del cliente envía un segmento TCP especial al TCP del lado del servidor. Este segmento no contiene datos de la capa de aplicación pero uno de los bits indicadores en la cabecera del segmento, el bit SYN se pone a 1. Además, el cliente selecciona de forma aleatoria un número de secuencia inicial (client_isn) y lo coloca en el campo número de secuencia del segmento TCP inicial SYN.
2. Una vez que el datagrama IP que contiene el segmento SYN TCP llega al host servior (suponiendo que llega), el servidor extrae dicho segmento SYN del datagrama, asigna los buffers y variables TCP a la conexión y envía un segmento de conexión concedida al cliente TCP. Este segmento de conexión concedida se conoce como segmento SYNACK y contiene tres fragmentos importantes de la cabecera del segmento. El primero, el bit SYN se pone a 1. El segundo, el campo reconocimiento (ACK) se hace igual a cliente_isn+1. Por último el servidor elige su propio numero de secuencia inicial (server_isn) que lo coloca en el indicador de secuencia.
3. Al recibir el segmento SYNACK, el cliente también asigna buffers y variables a la conexión. El host cliente envía entonces al servidor un último segmento que confirma el segmento de conexión concedida del servidor (almacena el valor server_isn+1 en el campo de reconocimiento de la cabecera). El bit SYN se pone a cero, ya que laconexión está establecida.



Aunque es posible que un par de entidades finales comiencen una conexión entre ellas simultáneamente, normalmente una de ellas abre un socket en un determinado puerto tcp y se queda a la escucha de nuevas conexiones. Es común referirse a esto como apertura pasiva, y determina el lado servidor de una conexión. El lado cliente de una conexión realiza una apertura activa de un puerto enviando un paquete SYN inicial al servidor como parte de la negociación en tres pasos. En el lado del servidor se comprueba si el puerto está abierto, es decir, si existe algún proceso escuchando en ese puerto. En caso de no estarlo, se envía al cliente un paquete de respuesta con el bit RST activado, lo que significa el rechazo del intento de conexión. En caso de que sí se encuentre abierto el puerto, el lado servidor respondería a la petición SYN válida con un paquete SYN/ACK. Finalmente, el cliente debería responderle al servidor con un ACK, completando así la negociación en tres pasos (SYN, SYN/ACK y ACK) y la fase de establecimiento de conexión.

Estados en TCP
CLOSED : No hay conexión activa ni pendiente.
LISTEN: El servidor espera una llamada.
SYN RCVD: Llegó una solicitud de conexión; espera ACK.
SYN SENT: La aplicación comenzó a abrir una conexión.
ESTABLISHED: Estado normal de transferencia de datos.
FIN WAIT 1: La aplicación dijo que ya terminó.
FIN WAIT 2: El otro lado acordó liberar.
TIMED WAIT: Espera a que todos los paquetes mueran.
CLOSING: Ambos lados intentaron cerrar simultáneamente.
CLOSE WAIT: El otro lado inició una liberación.
LAST ACK: Espera a que todos los paquetes mueran.

7. Investigue qué es el ISN (Initial Sequence Number). Relaciónelo con el saludo de tres vías.

El números de secuencia inicial (ISN) se refieren al número de secuencia único de 32 bits asignado a cada nueva conexión en una comunicación de datos basada en el protocolo de control de transmisión (TCP). Un ISN está diseñado para seleccionar aleatoriamente un número de secuencia para el primer byte de datos transmitidos en una nueva conexión TCP. El ISN puede ser cualquier número de 0 a 4.294.967.295.

Estos números de secuencia son usados para identificar los datos dentro del flujo de bytes, y poder identificar (y contar) los bytes de los datos de la aplicación. Siempre hay un par de números de secuencia incluidos en todo segmento TCP, referidos al número de secuencia y al número de asentimiento. Un emisor TCP se refiere a su propio número de secuencia cuando habla de número de secuencia, mientras que con el número de asentimiento se refiere al número de secuencia del receptor. Para mantener la fiabilidad, un receptor asiente los segmentos TCP indicando que ha recibido una parte del flujo continuo de bytes. Una mejora de TCP, llamada asentimiento selectivo (SACK, Selective Acknowledgement) permite a un receptor TCP asentir los datos que se han recibido de tal forma que el remitente solo retransmita los segmentos de datos que faltan.

A través del uso de números de secuencia y asentimiento, TCP puede pasar los segmentos recibidos en el orden correcto dentro del flujo de bytes a la aplicación receptora. Vuelve a cero tras el siguiente byte después del 232-1.

8. Utilice el comando ss (reemplazo de netstat) para obtener la siguiente información de su PC:

ss is used to dump socket statistics. It allows showing information similar to *netstat*. It can display more TCP and state informations than other tools.

OPTIONS

When no option is used ss displays a list of open non-listening sockets (e.g. TCP/UNIX/UDP) that have established connection.

-H, --no-header
Suppress header line.

```
       -n, --numeric
              Do not try to resolve service names.
       -r, --resolve
              Try to resolve numeric address/ports.
       -a, --all
              Display both listening and non-listening (for TCP this means
              established connections) sockets.
       -l, --listening
              Display only listening sockets (these are omitted by default).
       -p, --processes
              Show process using socket.
       -4, --ipv4
              Display only IP version 4 sockets (alias for -f inet).
       -6, --ipv6
              Display only IP version 6 sockets (alias for -f inet6).
       -t, --tcp
              Display TCP sockets.
       -u, --udp
              Display UDP sockets.
       -e, --extended
              Show detailed socket information. The output format is:

              uid:<uid_number> ino:<inode_number> sk:<cookie>
```

a. Para listar las comunicaciones TCP establecidas.
ss -tn

b. Para listar las comunicaciones UDP establecidas.
ss -un

c. Obtener sólo los servicios TCP que están esperando comunicaciones
ss -tln

d. Obtener sólo los servicios UDP que están esperando comunicaciones.
ss -uln

e. Repetir los anteriores para visualizar el proceso del sistema asociado a la conexión.
a_ ss -tpn b_ ss -upn c_ ss -tlpn d_ ss -ulpn

f. Obtenga la misma información planteada en los items anteriores usando el comando netstat.

```
       netstat  -  Print network connections, routing tables, interface sta-
       tistics, masquerade connections, and multicast memberships

--numeric, -n
       Show numerical addresses instead of trying to determine symbolic
       host, port or user names.
   -e, --extend
       Display additional information.  Use this option twice for maximum
       detail.
   -o, --timers
       Include information related to networking timers.
   -p, --program
       Show the PID and name of the program to which each socket belongs.
   -l, --listening
       Show only listening sockets.  (These are omitted by default.)
   -a, --all
       Show both listening and non-listening sockets.  With the --interfaces
       option, show interfaces that are not up
```

a_ netstat -tpn
b_ netstat s -upn
c_ netstat -tlpn
d_ netstat -ulpn

9. ¿Qué sucede si llega un segmento TCP a un host que no tiene ningún proceso esperando en el puerto destino de dicho segmento (es decir, que dicho puerto no está en estado LISTEN)?

Quien recibe el bit debe enviar un flag para informar que no se puede aceptar la conexión.

hping3 - send (almost) arbitrary TCP/IP packets to network hosts

Default protocol is TCP, by default hping3 will send tcp headers to target host's port 0 with a winsize of 64 without any tcp flag on. Often this is the best way to do an 'hide ping', useful when target is behind a firewall that drop ICMP. Moreover a tcp null-flag to port 0 has a good probability of not being logged.

-2 --udp

UDP mode, by default hping3 will send udp to target host's port 0. UDP header tunable options are the following: **--baseport, --destport, --keep.**

a. Utilice hping3 para enviar paquetes TCP al puerto destino 22 de la máquina virtual con el flag SYN activado.

```
redes@redes:~$ sudo hping3 -S 127.0.0.1 -p 22
HPING 127.0.0.1 (lo 127.0.0.1): S set, 40 headers + 0 data bytes
len=44 ip=127.0.0.1 ttl=64 DF id=0 sport=22 flags=SA seq=0 win=43690 rtt=1.1 ms
len=44 ip=127.0.0.1 ttl=64 DF id=0 sport=22 flags=SA seq=1 win=43690 rtt=2.8 ms
len=44 ip=127.0.0.1 ttl=64 DF id=0 sport=22 flags=SA seq=2 win=43690 rtt=2.1 ms
^C
--- 127.0.0.1 hping statistic ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 1.1/2.0/2.8 ms
```

b. Utilice hping3 para enviar paquetes TCP al puerto destino 40 de la máquina virtual con el flag SYN activado.

```
redes@redes:~$ sudo hping3 -S 127.0.0.1 -p 40
HPING 127.0.0.1 (lo 127.0.0.1): S set, 40 headers + 0 data bytes
len=40 ip=127.0.0.1 ttl=64 DF id=59561 sport=40 flags=RA seq=0 win=0 rtt=3.7 ms
len=40 ip=127.0.0.1 ttl=64 DF id=59741 sport=40 flags=RA seq=1 win=0 rtt=2.6 ms
len=40 ip=127.0.0.1 ttl=64 DF id=59987 sport=40 flags=RA seq=2 win=0 rtt=2.6 ms
^C
--- 127.0.0.1 hping statistic ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 2.6/3.0/3.7 ms
```

c. ¿Qué diferencias nota en las respuestas obtenidas en los dos casos anteriores? ¿Puede explicar a qué se debe? (Ayuda: utilice el comando ss visto anteriormente).

Cuando se envía un segmento TCP para establecer la conexión a un puerto que no está escuchando (40) entonces se recibe una respuesta con el flag RA, que quiere decir conexión rechazada. En cambio, cuando se envía a un puerto que si está escuchando, se recibe un flag SA que significa Syn ACK, que sería el segundo paso del saludo de tres vías para iniciar una conexión TCP.

10. ¿Qué sucede si llega un datagrama UDP a un host que no tiene a ningún proceso esperando en el puerto destino de dicho datagrama (es decir, que dicho puerto no está en estado LISTEN)?
Se va a recibir un mensaje del protocomo ICMP de la capa de transporte informando que el puerto en inalcanzable.

a. Utilice hping3 para enviar datagramas UDP al puerto destino 68 de la máquina virtual.

```
redes@redes:~$ sudo hping3 -2 127.0.0.1 -p 68
HPING 127.0.0.1 (lo 127.0.0.1): udp mode set, 28 headers + 0 data bytes
^C
--- 127.0.0.1 hping statistic ---
7 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
redes@redes:~$
```

b. Utilice hping3 para enviar datagramas UDP al puerto destino 40 de la máquina virtual.

```
redes@redes:~$ sudo hping3 -2 127.0.0.1 -p 40
HPING 127.0.0.1 (lo 127.0.0.1): udp mode set, 28 headers + 0 data bytes
ICMP Port Unreachable from ip=127.0.0.1 name=localhost
status=0 port=2350 seq=0
ICMP Port Unreachable from ip=127.0.0.1 name=localhost
status=0 port=2351 seq=1
ICMP Port Unreachable from ip=127.0.0.1 name=localhost
status=0 port=2352 seq=2
^C
--- 127.0.0.1 hping statistic ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 2.9/3.4/4.4 ms
```

c. ¿Qué diferencias nota en las respuestas obtenidas en los dos casos anteriores? ¿Puede explicar a qué se debe? (Ayuda: utilice el comando ss visto anteriormente).

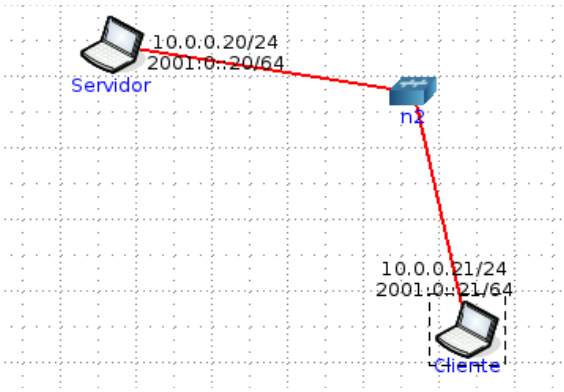
Cuando se envían los datos hacia un puerto que está escuchando entonces no se va a recibir respuesta ya que con el protocolo UDP simplemente se envían y se reciben datos y no hay un protocolo de saludo para establecer una conexión. Cuando se envía un segmento UDP a un puerto que no está escuchando o no es alcanzable, se va a recibir un mensaje desde la capa de red avisando que el puerto es inalcanzable.
Es posible establecer una conexión TCP y recibir datos sobre el mismo número de puerto, puesto que el SO toma un puerto UDP o un puerto TCP como puertos distintos de acuerdo al protocolo. Es decir, que el SO vería el puerto 63UDP como algo distinto a un puerto 63TCP.

11. Investigue qué es multicast. ¿Sobre cuál de los protocolos de capa de transporte funciona? ¿Se podría adaptar para que funcione sobre el otro protocolo de capa de transporte?¿Por qué?

Se denomina multicast a una comunicación grupal donde la información es dirigida hacia un grupo de computadoras de manera simultánea. Puede ser multicast de aplicación o multicast de red, siendo la última la que permita efectivamente enviar de manera simultánea la información. El protocolo de la capa de transporte utilizado para multicast es el protocolo UDP y suele ser utilizado por aplicaciones de estilo transmisión de televisión, o para charlas grupales.

Algunas empresas han desarrollado un sistema de transferencia fiable en la capa de aplicación que usa UDP como protocolo de transporte, para poder utilizar un multicast más fiable.
No se podría implementar con TCP, puesto que multicast sería el envío de un mensaje a muchos hosts, mientras que con TCP se imitaría pero no sería la misma función ya que deberían enviarse un mensaje por cada conexión abierto.

12. Use CORE para armar una topología como la siguiente, sobre la cual deberá realizar:



a. En ambos equipos inspeccionar el estado de las conexiones y mantener abiertas ambas ventanas con el comando corriendo para poder visualizar los cambios a medida que se realiza el ejercicio. Ayuda: watch -n1 'ss -nat'.
watch -n1 'ss -nat' para poder ver los cambios en cada host.

b. En Servidor, utilice la herramienta ncat para levantar un servicio que escuche en el puerto 8001/TCP. Utilice la opcion -k para que el servicio sea persistente. Verifique el estado de las conexiones.
root@Servidor:/tmp/pycore.50137/Servidor.conf# ncat -k -l 8001 para levantar un servicio que escuche en el puerto 8001

c. Desde CLIENTE1 conectarse a dicho servicio utilizando también la herramienta ncat. Inspeccione el estado de las conexiones.
root@Cliente:/tmp/pycore.50137/Cliente.conf# ncat 10.0.0.20 8001 para establecer una conexión.

```
Every 1,0s: ss -nat                                  Wed Apr 11 00:25:47 2018

State      Recv-Q Send-Q        Local Address:Port          Peer Address:Port
LISTEN     0      10                        *:8001                     *:*
ESTAB      0      0                10.0.0.20:8001             10.0.0.21:48680
LISTEN     0      10                       :::8001                    :::*
```

d. Iniciar otra conexión desde CLIENTE1 de la misma manera que la anterior y verificar el estado de las conexiones. ¿De qué manera puede identificar cada conexión?

```
Every 1,0s: ss -nat                                  Wed Apr 11 00:24:17 2018

State      Recv-Q Send-Q        Local Address:Port          Peer Address:Port
LISTEN     0      10                        *:8001                     *:*
ESTAB      0      0                10.0.0.20:8001             10.0.0.21:48680
ESTAB      0      0                10.0.0.20:8001             10.0.0.21:48681
LISTEN     0      10                       :::8001                    :::*
```

Se puede identificar cada conexión por el puerto del cliente.

e. En base a lo observado en el item anterior, ¿es posible iniciar más de una conexión desde el cliente al servidor en el mismo puerto destino? ¿Por qué? ¿Cómo se garantiza que los datos de una conexión no se mezclarán con los de la otra?

Si, como para definir los sokets se usan tanto las ip, como los puertos, se pueden enviar datos en mas de una conexión a la vez, y se entregan a los procesos que les correspondoan por su puerto.
La cuatrupla (ip_origen, port_origen, ip_destino, port_destino) es unica.

f. Cerrar la última conexión establecida desde CLIENTE1 y ver los estados de las conexiones en ambos equipos.

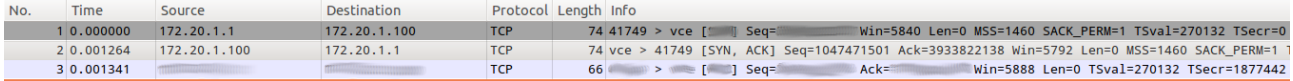
```
Every 1,0s: ss -nat                                     Wed Apr 11 00:25:47 2018

State      Recv-Q Send-Q        Local Address:Port          Peer Address:Port
TIME-WAIT  0      0                10.0.0.21:48681            10.0.0.20:8001
ESTAB      0      0                10.0.0.21:48680            10.0.0.20:8001
```

La conexión queda en estado de TIME-WAIT, esperando que pase un determinado tiempo antes de que se puedan volver a utilizar los mismos parametros.

g. Cortar el servicio de ncat en el servidor (Ctrl+C) y ver los estados de las conexiones en ambos equipos. Luego, cerrar la conexión en el cliente y verificar nuevamente los estados de las conexiones.
Si la conexión se cierra primero desde el servidor, entonces se queda en estado FIN-WAIT y hasta que no se cierra desde el cliente no se deja de listar; desde el cliente se puede ver el estado CLOSE-WAIT.

La conexión levantada estará en estado listen hasta que otra máquina se conecte. Cuando otra máquina se conecte, el puerto pasará a la categoría de established, y se listará con que IP se estableció la conexión. Una vez establecida la conexión ambos hosts pueden intercambiar segmentos. Cada una de las conexiones abiertas se va a identificar por el número de puerto.
Si un servidor establece varias conexiones sobre el mismo puerto entonces lo que distinguirá a las conexiones que estén establecidas en el mismo puerto será el campo IP de origen. Si la conexión se cierra primero desde el servidor, entonces se queda en estado FIN-WAIT y hasta que no se cierra desde el cliente no se deja de listar; desde el cliente se puede ver el estado CLOSE-WAIT.
Si el cliente cierra la conexión, desde el lado del cliente se puede ver el estado TIME-WAIT por unos segundos hasta que desaparece de la lista.

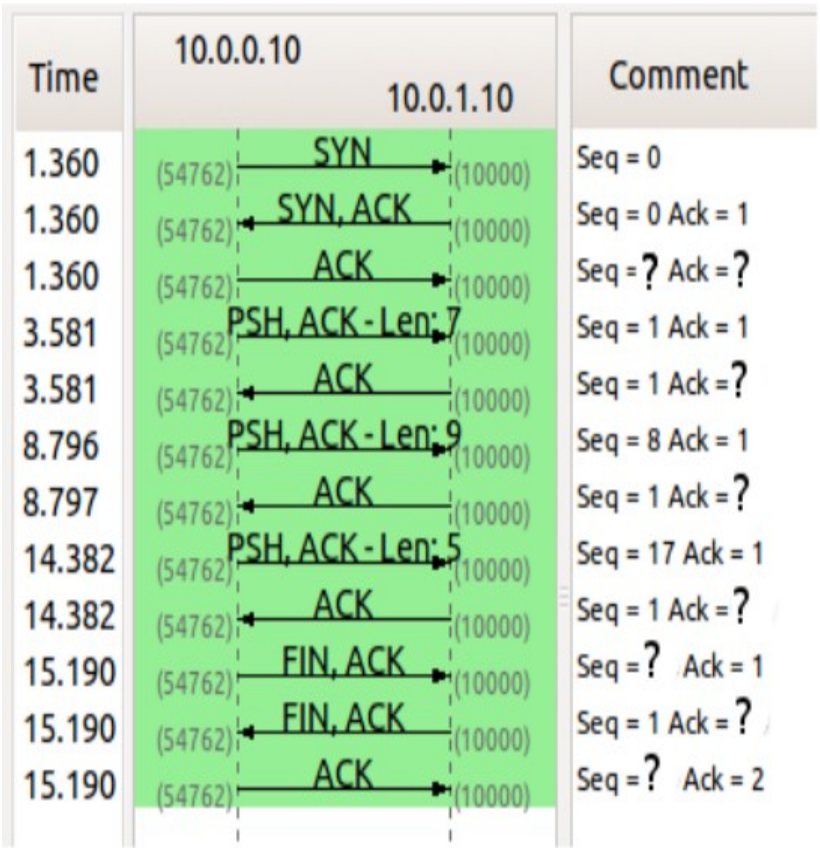
13. De acuerdo a la captura de la siguiente figura, indique los valores de los campos borroneados.



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	172.20.1.1	172.20.1.100	TCP	74	41749 > vce [ ] Seq= Win=5840 Len=0 MSS=1460 SACK_PERM=1 TSval=270132 TSecr=0
2	0.001264	172.20.1.100	172.20.1.1	TCP	74	vce > 41749 [SYN, ACK] Seq=1047471501 Ack=3933822138 Win=5792 Len=0 MSS=1460 SACK_PERM=1
3	0.001341			TCP	66	> [ ] Seq= Ack= Win=5888 Len=0 TSval=270132 TSecr=1877442

Nro.	Source	Destination	Info			
1	172.20.1.1	172.20.1.100	41749 >	vce	[SYN]	Seq=3933822437
2	172.20.1.100	172.20.1.1	vce < 41749		[SYN,ACK]	Seq=1047471501 Ack=3933822438
3	172.20.1.1	172.20.1.100	41749 >	vce	[ACK]	Seq=3933822438 Ack=1047471502

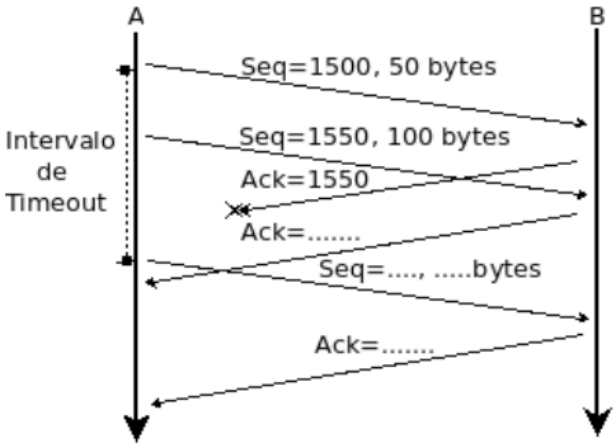
14. Dada la sesión TCP de la figura, completar los valores marcados con un signo de interrogación.



Seq= 1	Ack=1
Seq= 1	Ack=1
Seq=1	Ack=8
Seq=8	Ack=1
Seq=1	Ack=17
Seq=17	Ack=1
Seq=1	Ack=22
Seq=22	Ack=1
Seq=1	Ack=23
Seq=23	Ack=2

15. Completar los datos que faltan en el intercambio de mensajes del siguiente diagrama de flujo TCP:

TCP utiliza el algoritmo Selective Repeat.



1_ Ack=1650

2_ Seq= 1500 50bytes (Como no se recibio confirmacion se reenvian de nuevo los paquetes)

3_ Ack=1550 (confirma el ultimo que recibio)