Práctica 5

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

La función principal de la capa de transporte es proporcionar una comunicación lógica (desde la perspectiva de la aplicación, es como si los hosts que ejecutan los procesos estuvieran conectados directamente) entre procesos de aplicación que se ejecutan en dispositivos diferentes dentro de una red. Esta comunicación lógica permite que los procesos de aplicación se envíen mensajes entre sí sin preocuparse por los detalles de la infraestructura física subyacente.

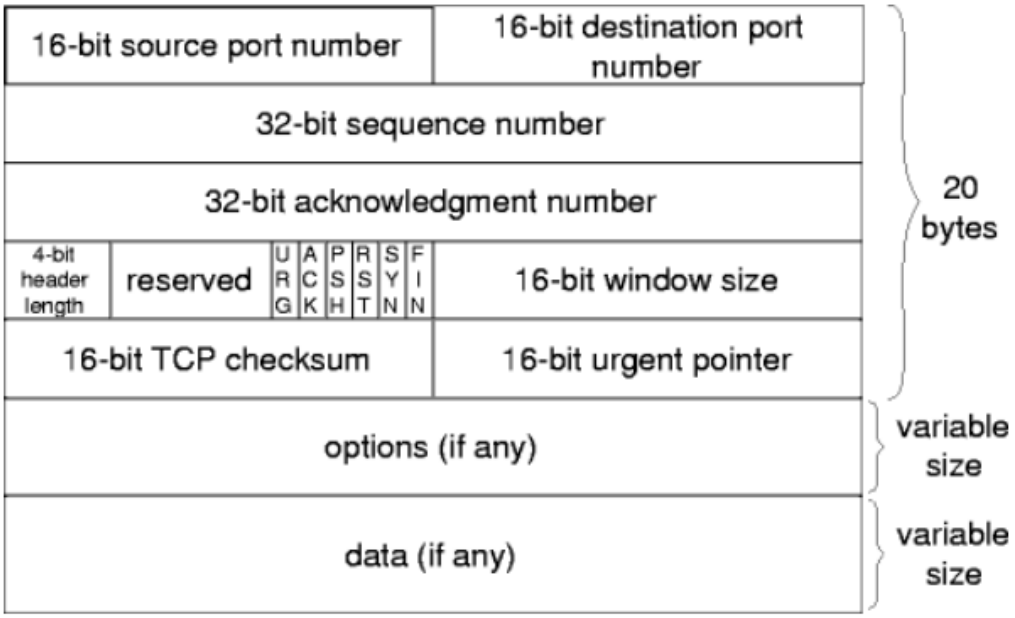
Los protocolos de la capa de transporte se implementan en los sistemas terminales (hosts), y no en los routers de la red. En el lado emisor, la capa de transporte convierte los mensajes de la aplicación en segmentos de la capa de transporte, que luego se encapsulan en paquetes de la capa de red y se envían al destino. Los routers de la red solo actúan sobre los campos correspondientes a la capa de red del paquete, sin examinar los campos del segmento de la capa de transporte encapsulado.

En el lado receptor, la capa de transporte extrae el segmento de la capa de transporte del paquete y lo entrega a la aplicación receptora. Para las aplicaciones de red, existen varios protocolos de la capa de transporte disponibles, como TCP y UDP, cada uno ofreciendo un conjunto diferente de servicios a las aplicaciones que los utilizan.

2. Describa la estructura del segmento TCP y UDP.

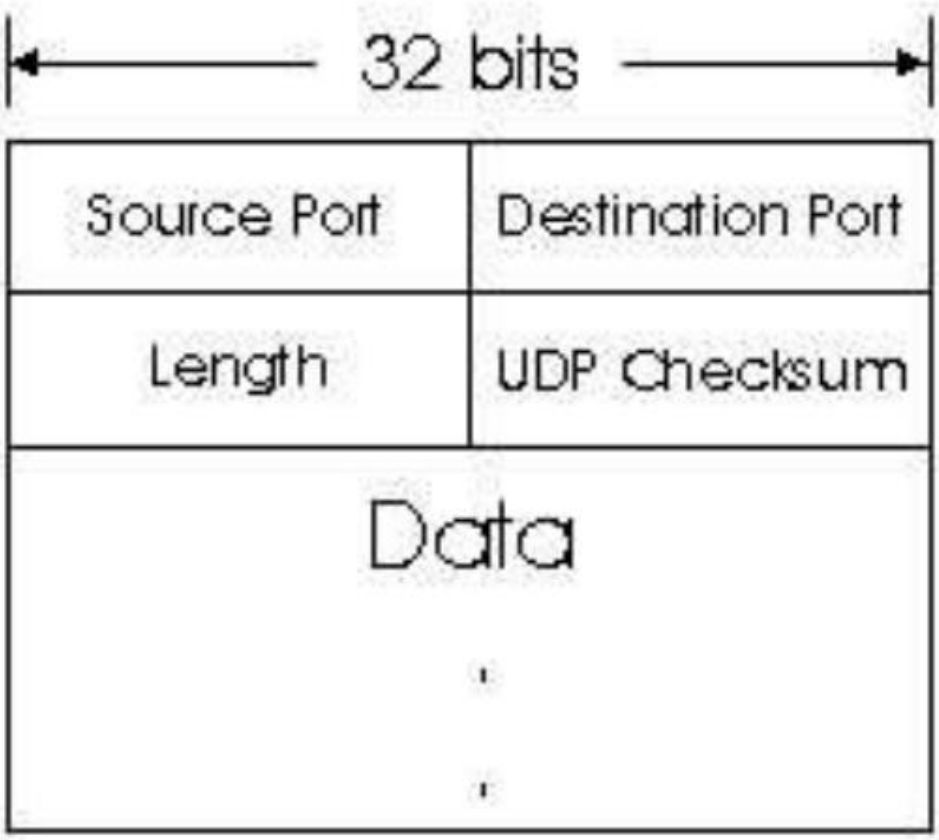
TCP	
16-bit source port number	Indica el número de puerto de origen de la aplicación que está enviando los datos.
16-bit destination port number	Especifica el número de puerto de destino de la aplicación receptora.
32-bit sequence number	Este campo se utiliza para mantener un seguimiento del orden de los segmentos TCP en una comunicación. Cada segmento TCP se etiqueta con un número de secuencia único.
32-bit acknowledgment number	Indica el número de secuencia que espera recibir el emisor del siguiente segmento. Ayuda a establecer que los datos se han recibido de manera confiable.
4-bit header length	Este campo especifica la longitud del encabezado TCP en palabras de 32 bits. Se utiliza para identificar dónde comienza la carga útil de datos en el segmento.

reserved	Este campo se reserva para uso futuro y debe establecerse en cero.
16-bit widow size	Indica el tamaño de la ventana de recepción que el receptor tiene disponible para aceptar datos. Ayuda a controlar el flujo de datos en la conexión.
16-bit TCP checksum	Proporciona una suma de verificación para verificar la integridad de los datos en el segmento TCP y detectar errores de transmisión.
16-bit urgent pointer	Se utiliza en la comunicación para indicar la posición de datos urgentes dentro del segmento, si es necesario.
options	Este campo opcional permite la inclusión de información adicional en el encabezado TCP, como máximas segment size (MSS), ventana de escala, timestamp, entre otros. Las opciones se utilizan para ajustar y optimizar la comunicación según las necesidades de la aplicación.



UDP	
Source Port	Especifica el número de puerto del proceso que envía el datagrama UDP
Destination Port	Este campo indica el número de puerto del proceso de destino al que se debe entregar el datagrama UDP
Length	Este campo especifica la longitud total del datagrama UDP, incluyendo tanto el encabezado como los datos. La longitud se mide en bytes y permite al receptor conocer la cantidad de información que debe procesar en el datagrama.

UDP Checksum	La suma de verificación es un valor calculado que se utiliza para detectar errores en el datagrama UDP durante la transmisión. Se calcula en función del contenido del datagrama, incluyendo el encabezado y los datos. El receptor verifica esta suma de verificación para determinar si el datagrama UDP ha llegado intacto o si ha sufrido algún tipo de corrupción durante la transmisión.



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

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

4. Compare TCP y UDP en cuanto a:

a. Confiabilidad.

TCP	UDP
Es un modelo confiable ya que garantiza que los datos se entreguen en el orden correcto y sin errores a través de técnicas	UDP es menos confiable en comparación con TCP. No garantiza la entrega de datos ni el orden de entrega. Los

como la retransmisión de datos perdidos y la detección y corrección de errores utilizando sumas de verificación.	segmentos UDP pueden perderse o llegar desordenados sin corrección automática.

b. Multiplexación.

Un proceso puede tener uno o varios sockets, por tanto la capa de transporte entrega los datos al socket (y no directamente a la aplicación), para identificar los sockets, éstos tienen un identificador único. Cada segmento de la capa de transporte contiene un campo para poder entregar los datos al socket adecuado. En el receptor, la capa de transporte examina estos campos para identificar el socket receptor y lo envía (demultiplexación).

Denominamos multiplexación al trabajo de reunir los datos en el host origen desde diferentes sockets, encapsulando los fragmentos de datos con la información de cabecera (que se usará en la demultiplexación).

TCP	UDP
Multiplexación y demultiplexación orientada a la conexion	Multiplexación y demultiplexación sin conexión
El socket TCP queda identificado por una tupla de cuatro elementos: dirección IP de origen, número de puerto de origen, dirección IP de destino, número de puerto de destino. Por lo tanto, cuando un segmento TCP llega a un host procedente de la red, el host emplea los cuatro valores para dirigir (demultiplexar) el segmento al socket apropiado.	El socket UDP queda completamente identificado por una tupla que consta de una dirección IP de destino y un número de puerto de destino. En consecuencia, si dos segmentos UDP tienen diferentes direcciones IP y/o números de puerto de origen, pero la misma dirección IP de destino y el mismo número puerto de destino, entonces los dos segmentos se enviarán al mismo proceso de destino a través del mismo socket de destino.

c. Orientado a la conexión.

TCP	UDP
Establece una conexión antes de la transmisión de datos y asegura que ambas partes estén sincronizadas en términos de secuencia de datos y control de flujo.	No se necesita conexión para iniciar y finalizar una transferencia de datos

d. Controles de congestión.

TCP	UDP

Proporciona mecanismos de control de congestión. Los mecanismos de control de congestión de TCP evitan que cualquier conexión TCP inunde con una cantidad de tráfico excesiva los enlaces y routers existentes entre los hosts que están comunicándose. Esto se consigue regulando la velocidad a la que los lados emisores de las conexiones TCP pueden enviar tráfico a la red.	El tráfico UDP no está regulado. Una aplicación que emplee el protocolo de transporte UDP puede enviar los datos a la velocidad que le parezca, durante todo el tiempo que quiera.
Posee un mecanismo que indica al emisor cuánto espacio libre hay en el búfer de almacenamiento del receptor (ventana de recepción). Ayuda a controlar el flujo de datos para evitar la congestión y garantizar una comunicación eficiente, permitiendo que el emisor ajuste la cantidad de datos enviados en función de la capacidad disponible en el receptor.	

e. Utilización de puertos.

TCP	UDP
Como está orientado a la conexión, establece una conexión punto a punto entre dos dispositivos, por lo que cada conexión está limitada a dos procesos que intercambian datos.	Permite que muchos clientes o procesos envíen datos por el mismo socket
Utiliza números de puerto para identificar aplicaciones específicas.	Utiliza números de puerto para identificar aplicaciones específicas.

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

Cuando se trata del protocolo UDP, el termino datagrama se utiliza para su PDU.

6. Describa el saludo de tres vías de TCP. ¿Se utiliza algo similar en UDP?

También conocido como protocolo de enlace de TCP, es un método utilizado por TCP para establecer una conexión confiable entre dos dispositivos en una red. Es un método de tres pasos que requiere que tanto el cliente como el servidor intercambien segmentos SYN y ACK antes de que comience la comunicación de datos real.

Paso 1 (SYN) – El cliente inicia el proceso enviando un segmento al servidor con el bit SYN establecido y un número de secuencia inicial (ISN) generado de manera pseudoaleatoria. Este es importante para identificar y ordenar los datos en la conexión.

Paso 2 (SYN/ACK) – El servidor recibe el segmento del cliente, reconoce el bit SYN y responde enviando un segmento de respuesta con los bits SYN y ACK establecidos. En este, el servidor incluye su propio número de secuencia inicial (ISN), que también es elegido de manera pseudoaleatoria.
El servidor también reconoce el ISN del cliente, lo que indica que ha recibido correctamente el paquete de solicitud de conexión.

Paso 3 (ACK) – El cliente recibe la respuesta del servidor, reconociendo el ISN del servidor. El cliente responde enviando un segmento de confirmación con el bit ACK establecido, confirmando que ha recibido correctamente la respuesta del servidor. Ambos establecen una conexión confiable con la cual iniciarán la transferencia de datos real.

En UPD no se utiliza nada similar ya no se establece ninguna conexion

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

   Initial Sequence Number (ISN) se refieren al número de secuencia único de 32 bits asignado a cada nueva conexión en una comunicación TCP. Ayuda a que no entren en conflicto los bytes de datos transmitidos a través de una conexión TCP. Un ISN es único para cada conexión y está separado por cada dispositivo. Para el ISN se utiliza un contador que se incrementa cada 4 mseg.

   El ISN ayudar a identificar, controlar el origen y mantener el orden de los segmentos de datos transmitidos entre el cliente y el servidor.

8. **Investigue qué es el MSS. ¿Cuándo y cómo se negocia?**

   El tamaño de la ventana de recepción TCP es la cantidad de datos de recepción (en bytes) que se pueden almacenar en búfer durante una conexión.

   En lugar de usar un tamaño de ventana de recepción predeterminado codificado de forma rígida, TCP se ajusta a incrementos pares del tamaño máximo de segmento (MSS).

   Maximum Segment Size es un campo de los encabezados que indica el tamaño más grande de datos que puede tener un segmento sin ser fragmentado. El MSS mide la parte de un paquete que no tiene encabezado, lo que se conoce como carga útil. El MSS está determinado por otra métrica que tiene que ver con el tamaño de los paquetes: MTU, o la unidad máxima de transmisión, que sí incluye los encabezados TCP e IP (Protocolo de Internet).

   El MSS es igual a la MTU menos el tamaño de un encabezado TCP y un encabezado IP:

MTU - (encabezado TCP + encabezado IP) = MSS

Una de las principales diferencias entre la MTU y el MSS es que si un paquete supera la MTU de un dispositivo, se divide en trozos más pequeños, o "se fragmenta." En cambio, si un paquete supera el MSS, se descarta y no se entrega.

El MSS se negocia durante la configuración de la conexión, es decir, durante el saludo de tres vías.

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

   a. **Para listar las comunicaciones TCP establecidas.**

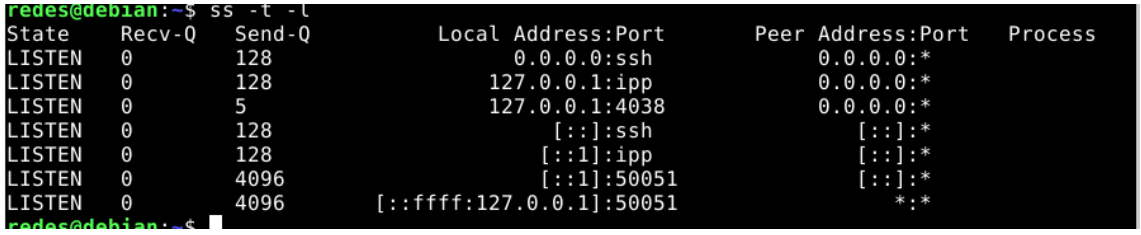
   ss −t

   b. **Para listar las comunicaciones UDP establecidas.**

   ss −u

   c. **Obtener sólo los servicios TCP que están esperando comunicaciones**

   ss −t −l



   d. **Obtener sólo los servicios UDP que están esperando comunicaciones.**

   ss −u −l

```
redes@debian:~$ ss -u -l
State      Recv-Q     Send-Q         Local Address:Port          Peer Address:Port    Process
UNCONN     0          0                    0.0.0.0:48109              0.0.0.0:*
UNCONN     0          0                    0.0.0.0:mdns               0.0.0.0:*
UNCONN     0          0                  127.0.0.1:4038               0.0.0.0:*
UNCONN     0          0                    0.0.0.0:631                0.0.0.0:*
UNCONN     0          0                       [::]:mdns                  [::]:*
UNCONN     0          0                       [::]:48940                 [::]:*
```

e. **Repetir los anteriores para visualizar el proceso del sistema asociado a la conexión.**

ss -t -p
ss -u -p
ss -t -l -p
ss -u -l -p

(hacerlo desde root)

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

netstat -t -p
netstat -u -p
netstat -t -l -p
netstat -u -l -p

10. **¿Qué sucede si llega un segmento TCP con el flag SYN activo 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)?**

Si no hay proceso en estado LISTEN en el puerto destino, el host enviara un segmente con la flag RST (reset) activado para indicar que la conexión no se puede establecer en dicho puerto. De esta manera informará al remitente que no hay no se puede establecer la conexión en ese momento para que no siga enviando segmentos.

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

hping3 -S -p 22 localhost

```
len=44 ip=127.0.0.1 ttl=64 DF id=0 sport=22 flags=SA seq=0 win=65495 rtt=2.6 ms
len=44 ip=127.0.0.1 ttl=64 DF id=0 sport=22 flags=SA seq=1 win=65495 rtt=2.2 ms
len=44 ip=127.0.0.1 ttl=64 DF id=0 sport=22 flags=SA seq=2 win=65495 rtt=4.9 ms
len=44 ip=127.0.0.1 ttl=64 DF id=0 sport=22 flags=SA seq=3 win=65495 rtt=12.7 ms
len=44 ip=127.0.0.1 ttl=64 DF id=0 sport=22 flags=SA seq=4 win=65495 rtt=0.1 ms
len=44 ip=127.0.0.1 ttl=64 DF id=0 sport=22 flags=SA seq=5 win=65495 rtt=2.0 ms
len=44 ip=127.0.0.1 ttl=64 DF id=0 sport=22 flags=SA seq=6 win=65495 rtt=0.5 ms
len=44 ip=127.0.0.1 ttl=64 DF id=0 sport=22 flags=SA seq=7 win=65495 rtt=12.8 ms
len=44 ip=127.0.0.1 ttl=64 DF id=0 sport=22 flags=SA seq=8 win=65495 rtt=3.3 ms
```

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

hping3 -S -p 40 localhost

```
len=40 ip=127.0.0.1 ttl=64 DF id=0 sport=40 flags=RA seq=1 win=0 rtt=6.9 ms
len=40 ip=127.0.0.1 ttl=64 DF id=0 sport=40 flags=RA seq=2 win=0 rtt=1.9 ms
len=40 ip=127.0.0.1 ttl=64 DF id=0 sport=40 flags=RA seq=3 win=0 rtt=5.9 ms
len=40 ip=127.0.0.1 ttl=64 DF id=0 sport=40 flags=RA seq=4 win=0 rtt=5.1 ms
len=40 ip=127.0.0.1 ttl=64 DF id=0 sport=40 flags=RA seq=5 win=0 rtt=169.3 ms
len=40 ip=127.0.0.1 ttl=64 DF id=0 sport=40 flags=RA seq=6 win=0 rtt=3.6 ms
len=40 ip=127.0.0.1 ttl=64 DF id=0 sport=40 flags=RA seq=7 win=0 rtt=4.5 ms
len=40 ip=127.0.0.1 ttl=64 DF id=0 sport=40 flags=RA seq=8 win=0 rtt=17.3 ms
len=40 ip=127.0.0.1 ttl=64 DF id=0 sport=40 flags=RA seq=9 win=0 rtt=4.6 ms
len=40 ip=127.0.0.1 ttl=64 DF id=0 sport=40 flags=RA seq=10 win=0 rtt=6.6 ms
len=40 ip=127.0.0.1 ttl=64 DF id=0 sport=40 flags=RA seq=11 win=0 rtt=8.3 ms
len=40 ip=127.0.0.1 ttl=64 DF id=0 sport=40 flags=RA seq=12 win=0 rtt=4.8 ms
len=40 ip=127.0.0.1 ttl=64 DF id=0 sport=40 flags=RA seq=13 win=0 rtt=4.6 ms
len=40 ip=127.0.0.1 ttl=64 DF id=0 sport=40 flags=RA seq=14 win=0 rtt=5.5 ms
len=40 ip=127.0.0.1 ttl=64 DF id=0 sport=40 flags=RA seq=15 win=0 rtt=8.6 ms
len=40 ip=127.0.0.1 ttl=64 DF id=0 sport=40 flags=RA seq=16 win=0 rtt=3.3 ms
len=40 ip=127.0.0.1 ttl=64 DF id=0 sport=40 flags=RA seq=17 win=0 rtt=3.6 ms
len=40 ip=127.0.0.1 ttl=64 DF id=0 sport=40 flags=RA seq=18 win=0 rtt=7.5 ms
```

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

   En el puerto 22 se tiene el flag SA , que corresponde a cuando el puerto destino está abierto, es decir, el flag SYN/ACK esta activado y se pudo establecer la comunicación. Por su parte, en el 40 se tiene el flag RA, que corresponde a cuando el puerto destino está cerrado, es decir RST/ACK esta activado y no se pudo establecer la comunicación.

11. **¿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)?**

    Si no hay proceso en estado LISTEN en el puerto destino, la respuesta dependerá de si la aplicación que escucha en ese puerto ha sido configurada para proporcionar feedback. En algunos casos, la aplicación puede estar diseñada para enviar un mensaje de error o una respuesta personalizada, como un paquete ICMP “Destination Unreachable”, si recibe un datagrama en un puerto no esperado o no deseado. Este mensaje ICMP indica que el puerto o el host destino no están disponibles.
    Sin embargo, si la aplicación en el puerto destino no está configurada para proporcionar este tipo de feedback, UDP no generará ninguna notificación al remitente ya que al ser un protocolo sin conexión y no confiable, no proporciona mecanismos integrados para notificar al remitente si un datagrama no se entregó o si el puerto no está en uso.

    a. **Utilice hping3 para enviar datagramas UDP al puerto destino 5353 de la máquina virtual**

```
HPING localhost (lo 127.0.0.1): udp mode set, 28 headers + 0 data bytes
^C
--- localhost hping statistic ---
22 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
```

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

```
ICMP Port Unreachable from ip=127.0.0.1 name=localhost
status=0 port=2133 seq=2
ICMP Port Unreachable from ip=127.0.0.1 name=localhost
status=0 port=2134 seq=3
ICMP Port Unreachable from ip=127.0.0.1 name=localhost
status=0 port=2135 seq=4
ICMP Port Unreachable from ip=127.0.0.1 name=localhost
status=0 port=2136 seq=5
ICMP Port Unreachable from ip=127.0.0.1 name=localhost
status=0 port=2137 seq=6
ICMP Port Unreachable from ip=127.0.0.1 name=localhost
status=0 port=2138 seq=7
ICMP Port Unreachable from ip=127.0.0.1 name=localhost
status=0 port=2139 seq=8
```

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).**

La diferencia en este caso es que en el puerto 5353 hay un proceso escuchando (igualmente no se indica que el datagrama llego y eso es razonable ya que se trata del procolo UDP) y en el 40 no. En el caso del puerto 40, hay un feedback (ICMP) que indica que el puerto es inalcanzable.

Se puede ver mediante este comando que en el puerto 5353 hay un proceso activo.

```
root@debian:~# ss -u -ln
State   Recv-Q   Send-Q     Local Address:Port      Peer Address:Port  Process
UNCONN  0        0              127.0.0.1:4038           0.0.0.0:*
UNCONN  0        0                0.0.0.0:49424          0.0.0.0:*
UNCONN  0        0                0.0.0.0:631            0.0.0.0:*
UNCONN  0        0                0.0.0.0:5353           0.0.0.0:*
UNCONN  0        0                   [::]:42014             [::]:*
UNCONN  0        0                   [::]:5353              [::]:*
```

12. **Investigue los distintos tipos de estado que puede tener una conexión TCP. Ver https://users.cs.northwestern.edu/~agupta/cs340/project2/TCPIP_State_Transition_Diagram.pdf**

Una conexión atraviesa una serie de estados a lo largo de su ciclo de vida. Estos estados incluyen: LISTEN, SYN-SENT, SYN-RECEIVED, ESTABLISHED, FIN-WAIT-1, FIN-WAIT-2, CLOSE-WAIT, CLOSING, LAST-ACK, TIME-WAIT y el estado ficticio CLOSED. El estado CLOSED es ficticio porque representa la situación en la que no existe un Bloque de Control de Transmisión (TCB), lo que significa que no hay conexión alguna. Para resumir, estos estados tienen los siguientes significados:

- ✧ LISTEN representa la espera de una solicitud de conexión procedente de cualquier TCP remoto y puerto.

- ✧ SYN-SENT representa la espera de una solicitud de conexión coincidente después de haber enviado una solicitud de conexión.

- ✧ SYN-RECEIVED representa la espera de una confirmación de solicitud de conexión después de haber recibido y enviado una solicitud de conexión.

- ✧ ESTABLISHED representa una conexión abierta, lo que permite que los datos recibidos sean entregados al usuario. Es el estado normal durante la fase de transferencia de datos de la conexión.

- ✧ FIN-WAIT-1 representa la espera de una solicitud de terminación de conexión procedente del TCP remoto, o la confirmación de la solicitud de terminación de conexión previamente enviada.

- ✧ FIN-WAIT-2 representa la espera de una solicitud de terminación de conexión procedente del TCP remoto.

- ✧ CLOSE-WAIT representa la espera de una solicitud de terminación de conexión procedente del usuario local.

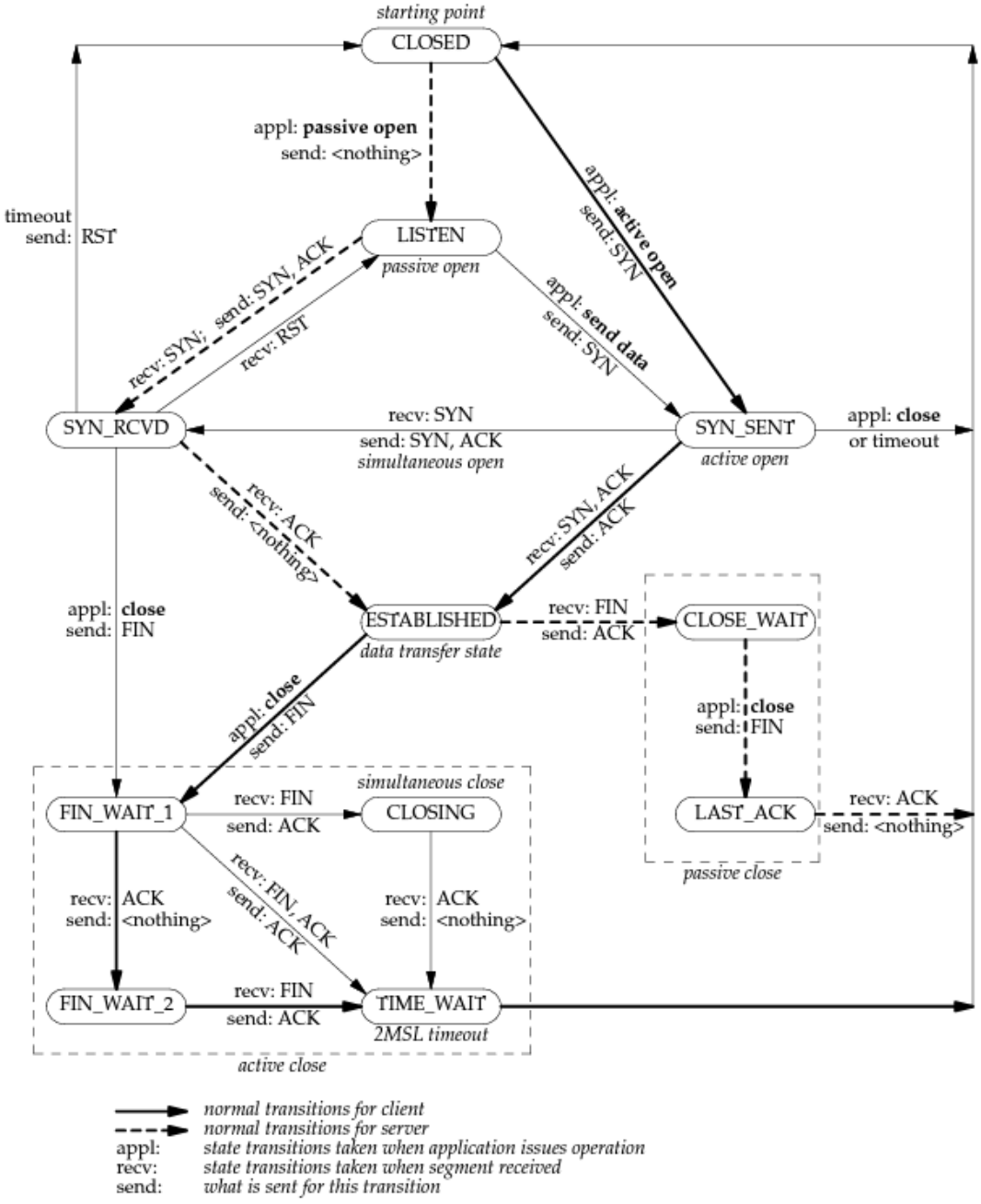
- ✧ CLOSING representa la espera de una confirmación de la solicitud de terminación de conexión procedente del TCP remoto.

- ✧ LAST-ACK representa la espera de una confirmación de la solicitud de terminación de conexión previamente enviada al TCP remoto (la cual incluye una confirmación de su solicitud de terminación de conexión).

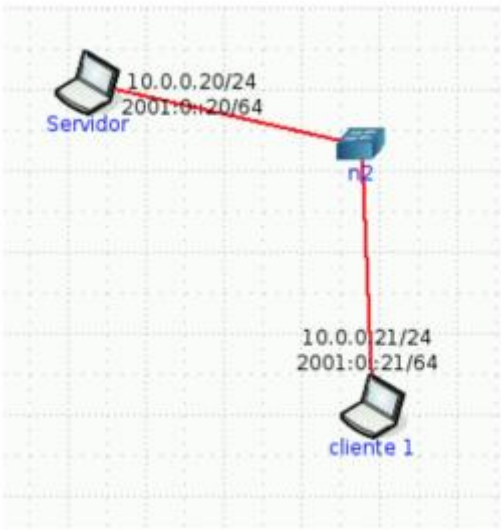
- ✧ TIME-WAIT representa la espera de suficiente tiempo para asegurar que el TCP remoto ha recibido la confirmación de su solicitud de terminación de conexión.

- ✧ CLOSED representa la ausencia total de una conexión activa.

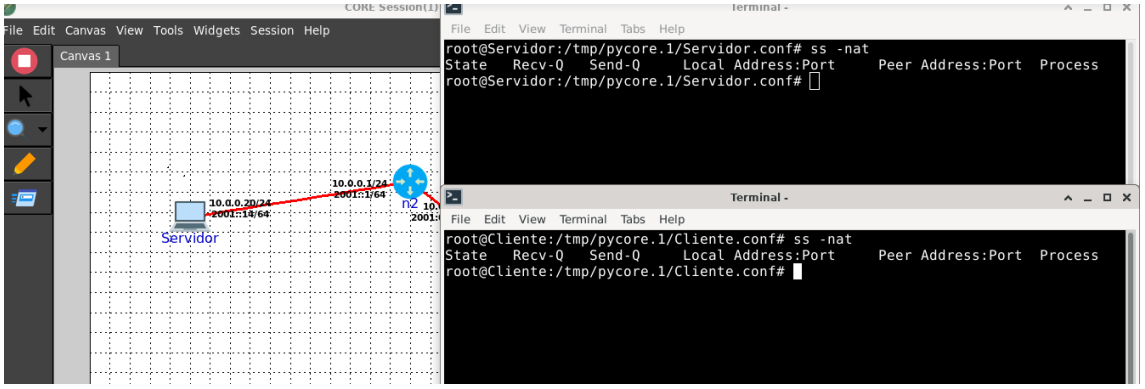
Una conexión TCP avanza de un estado a otro en respuesta a diversos eventos, tales como llamadas del usuario (OPEN, SEND, RECEIVE, CLOSE, ABORT y STATUS), segmentos entrantes, especialmente aquellos que contienen las banderas SYN, ACK, RST y FIN, así como también mediante el uso de temporizadores."



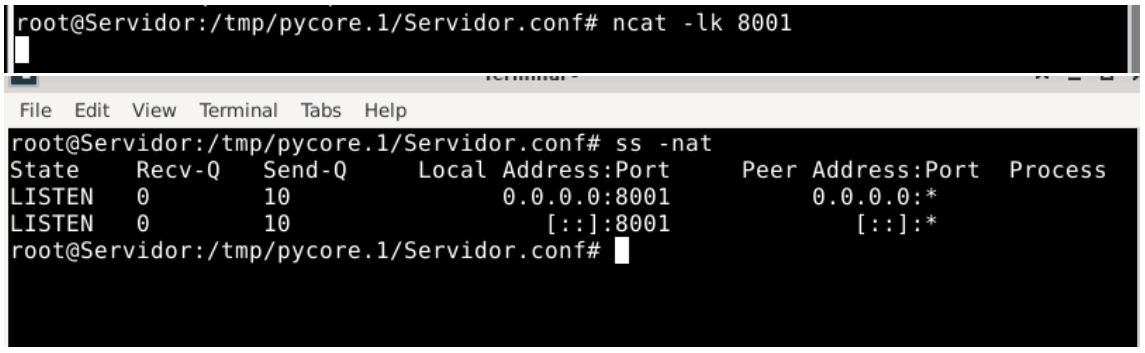
13. Use CORE para armar una topología como la siguiente, sobre la cual deberá realizar:



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'.**



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.**



c. **Desde CLIENTE1 conectarse a dicho servicio utilizando también la herramienta ncat. Inspeccione el estado de las conexiones.**

```
root@Cliente:/tmp/pycore.1/Cliente.conf# ncat 10.0.0.20 8001
```

```
root@Cliente:/tmp/pycore.1/Cliente.conf# ss -nat
State    Recv-Q    Send-Q        Local Address:Port        Peer Address:Port   Process
ESTAB    0         0               10.0.1.20:42496          10.0.0.20:8001
```

```
root@Servidor:/tmp/pycore.1/Servidor.conf# ss -nat
State    Recv-Q    Send-Q        Local Address:Port        Peer Address:Port   Process
LISTEN   0         10                0.0.0.0:8001             0.0.0.0:*
ESTAB    0         0               10.0.0.20:8001           10.0.1.20:42496
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?

```
State    Recv-Q    Send-Q        Local Address:Port        Peer Address:Port   Process
ESTAB    0         0               10.0.1.20:42496          10.0.0.20:8001
ESTAB    0         0               10.0.1.20:42756          10.0.0.20:8001
```

```
root@Servidor:/tmp/pycore.1/Servidor.conf# ss -nat
State    Recv-Q    Send-Q        Local Address:Port        Peer Address:Port   Process
LISTEN   0         10                0.0.0.0:8001             0.0.0.0:*
ESTAB    0         0               10.0.0.20:8001           10.0.1.20:42496
ESTAB    0         0               10.0.0.20:8001           10.0.1.20:42756
LISTEN   0         10                   [::]:8001                [::]:*
```

Se pueden identificar gracias a que los clientes están en puertos distintos.

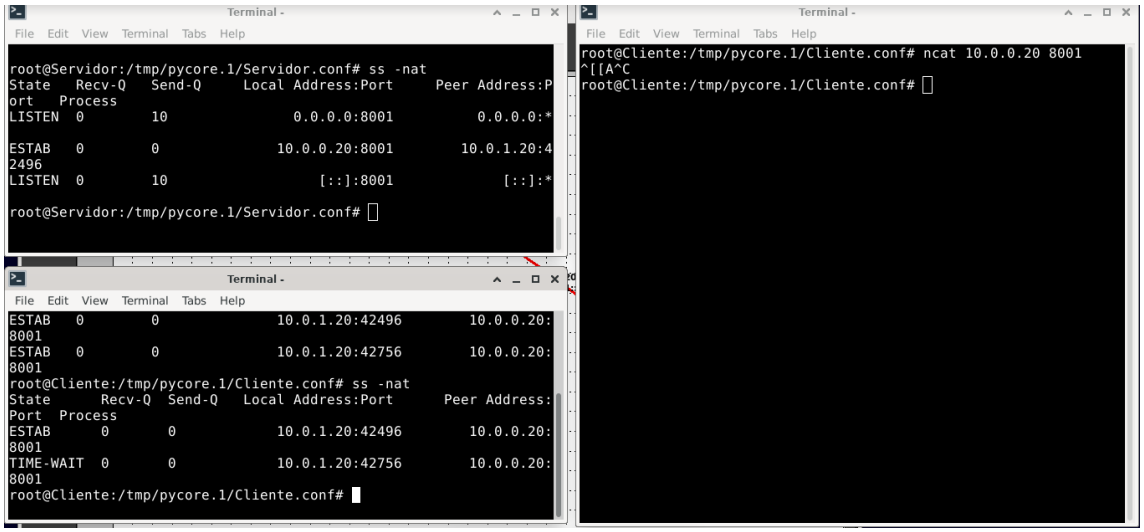
e. En base a lo observado en el item anterior, ¿es posible iniciar más de una conexión desde el cliente al servidor en el mismo puerto destino? ¿Por qué? ¿Cómo se garantiza que los datos de una conexión no se mezclarán con los de la otra?

Si, es posible ya que los distintos clientes tienen distintos puertos origines y eso garantiza que sea posible que se inicie más de una conexión desde el cliente al servidor en el mismo puerto destino ya que cada conexión se identifica de manera única por la combinación de la dirección IP y el número de puerto local y remoto.

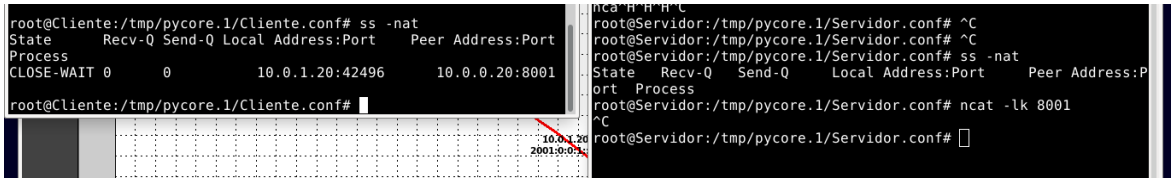
Cada conexión se gestiona de manera individual y se mantiene separada de las demás gracias a la combinación única de direcciones IP y puertos.

f. Analice en el tráfico de red, los flags de los segmentos TCP que ocurren cuando:

i. Cierra la última conexión establecida desde CLIENTE1. Evalúe los estados de las conexiones en ambos equipos.



ii. Corta el servicio de ncat en el servidor (Ctrl+C). Evalúe los estados de las conexiones en ambos equipos.



iii. Cierra la conexión en el cliente. Evalúe nuevamente los estados de las conexiones.

14. Dada la siguiente salida del comando ss, responda:

```
Netid  State      Recv-Q Send-Q   Local Address:Port       Peer Address:Port
tcp    LISTEN     0      128                *:22                    *:*      users:(("sshd",pid=468,fd=29))
tcp    LISTEN     0      128                *:80                    *:*      users:(("apache2",pid=991,fd=95))
udp    LISTEN     0      128       163.10.5.222:53                  *:*      users:(("named",pid=452,fd=10))
tcp    ESTAB      0      0         163.10.5.222:59736   64.233.163.120:443    users:(("x-www-browser",pid=1079,fd=51))
tcp    CLOSE-WAIT 0      0         163.10.5.222:41654    200.115.89.30:443    users:(("x-www-browser",pid=1079,fd=50))
tcp    ESTAB      0      0         163.10.5.222:59737   64.233.163.120:443    users:(("x-www-browser",pid=1079,fd=55))
tcp    ESTAB      0      0         163.10.5.222:33583    200.115.89.15:443    users:(("x-www-browser",pid=1079,fd=53))
tcp    ESTAB      0      0         163.10.5.222:45293    64.233.190.99:443    users:(("x-www-browser",pid=1079,fd=59))
tcp    LISTEN     0      128                *:25                    *:*      users:(("postfix",pid=627,fd=3))
tcp    ESTAB      0      0           127.0.0.1:22          127.0.0.1:41220    users:(("sshd",pid=1418,fd=3),
                                                                               ("sshd",pid=1416,fd=3))
tcp    ESTAB      0      0         163.10.5.222:52952    64.233.190.94:443    users:(("x-www-browser",pid=1079,fd=29))
tcp    TIME-WAIT  0      0         163.10.5.222:36676    54.149.207.17:443    users:(("x-www-browser",pid=1079,fd=3))
tcp    ESTAB      0      0         163.10.5.222:52960    64.233.190.94:443    users:(("x-www-browser",pid=1079,fd=67))
tcp    ESTAB      0      0         163.10.5.222:50521    200.115.89.57:443    users:(("x-www-browser",pid=1079,fd=69))
tcp    SYN-SENT   0      0         163.10.5.222:52132      43.232.2.2:9500    users:(("x-www-browser",pid=1079,fd=70))
tcp    ESTAB      0      0           127.0.0.1:41220       127.0.0.1:22       users:(("ssh",pid=1415,fd=3))
udp    LISTEN     0      128         127.0.0.1:53                  *:*      users:(("named",pid=452,fd=9))
```

a. **¿Cuántas conexiones hay establecidas?**

   9 (ESTAB)

b. **¿Cuántos puertos hay abiertos a la espera de posibles nuevas conexiones?**

   4 (LISTEN)

c. **El cliente y el servidor de las comunicaciones HTTPS (puerto 443), ¿residen en la misma máquina?**

   No, deberían coincidir las IPS.

d. **El cliente y el servidor de la comunicación SSH (puerto 22), ¿residen en la misma máquina?**

   Si, tanto la IP Local como la Destino son la misma. Una esta en el puerto 22 y la otra en el 41220. En el puerto 22 está el proceso SSHD, y en el 41220 está corriendo el proceso SSH.

e. **Liste los nombres de todos los procesos asociados con cada comunicación. Indique para cada uno si se trata de un proceso cliente o uno servidor.**

   Los que están en estado LISTEN y/o en puertos del 0 al 1023 son procesos servidores.

   - SSHD: Servidor
   - APACHE2: Servidor
   - NAMED: Servidor
   - X-WWW-BROWSER: Cliente
   - POSTFIX: Servidor
   - SSH: Cliente

f. **¿Cuáles conexiones tuvieron el cierre iniciado por el host local y cuáles por el remoto?**

   TIME-WAIT Por el local, CLOSE-WAIT por el remoto

g. **¿Cuántas conexiones están aún pendientes por establecerse?**

   1 (SYN-SENT)

15. **Dadas las salidas de los siguientes comandos ejecutados en el cliente y el servidor, responder:**

```
servidor# ss -natu | grep 110
tcp    LISTEN     0      0             *:110              *:*
tcp    SYN-RECV   0      0     157.0.0.1:110     157.0.11.1:52843

cliente# ss -natu | grep 110
tcp    SYN-SENT   0      1    157.0.11.1:52843    157.0.0.1:110
```

a. ¿Qué segmentos llegaron y cuáles se están perdiendo en la red?

SYN llego al servidor, pero SYNACK no llego al cliente.

b. ¿A qué protocolo de capa de aplicación y de transporte se está intentando conectar el cliente?

Al protocolo POP3 de la capa de aplicación con el protocolo TCP

c. ¿Qué flags tendría seteado el segmento perdido?

SYN: 1
ACK: 1