Práctica 5

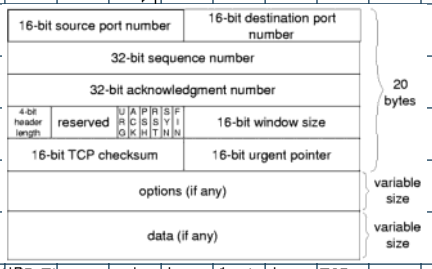
martes, 29 de octubre de 2024 14:18

1. ¿Cuál es la función de la capa de transporte?
Gestiona la comunicación de extremo a extremo entre dispositivos en una red. Identificar cada aplicación. Hacia qué aplicacion de las que está corriendo el otro dispositivo va la data?

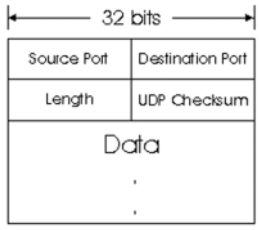
- Red (IP) direcciona hosts, transporta al mensaje del transporte hasta el host (dir IP).
- Transporte (TCP,UDP, u otro) transporta al mensaje de aplicación hasta el proceso (puerto).

2. Describa la estructura del segmento TCP y UDP.

- TCP: Incluye campos como puerto de origen, puerto de destino, número de secuencia, número de acuse de recibo (ACK), tamaño de ventana, banderas (SYN, ACK, FIN, etc.), checksum para control de errores, y datos.



- UDP: Tiene una estructura más simple que TCP, con campos de puerto de origen, puerto de destino, longitud, checksum, y datos.



3. ¿Cuál es el objetivo del uso de puertos en el modelo TCP/IP?
Los puertos permiten que múltiples aplicaciones o servicios en un mismo dispositivo se comuniquen a través de la red. Ayudan a identificar aplicaciones específicas que deben recibir y procesar los datos entrantes, asegurando que las comunicaciones lleguen a la aplicación o servicio correcto en el dispositivo de destino.

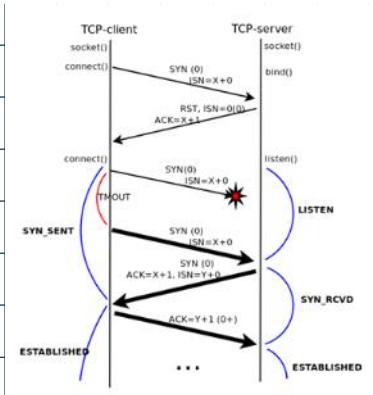
4. Compare TCP y UDP en cuanto a:
a. Confiabilidad
b. Multiplexación.
c. Orientado a la conexión.
d. Controles de congestión.
e. Utilización de puertos

Característica	TCP	UDP
Confiabilidad	Proporciona confiabilidad con acuses de recibo (ACK) y retransmisión de paquetes perdidos.	No ofrece confiabilidad; los datos se envían sin confirmación.
Multiplexación	Usa puertos para multiplexar conexiones, permitiendo múltiples aplicaciones simultáneamente.	También usa puertos para multiplexar, aunque de forma más simple.
Orientado a conexión	Sí, requiere el establecimiento de una conexión (saludo de tres vías) antes de transmitir.	No orientado a la conexión; envía datos sin establecer una conexión previa.
Control de congestión	Implementa controles de congestión para evitar sobrecargar la red y mejorar la estabilidad.	No tiene control de congestión; envía datos independientemente del estado de la red.
Utilización de puertos	Utiliza puertos para identificar las aplicaciones o servicios con comunicación confiable.	También utiliza puertos para identificar aplicaciones, sin proporcionar confiabilidad.

5. La PDU de la capa de transporte es el segmento. Sin embargo, en algunos contextos suele utilizarse el término datagrama. Indique cuando.
El término "datagrama" se utiliza frecuentemente para referirse a la PDU de la capa de transporte cuando se habla de UDP, debido a su naturaleza de transmisión sin conexión. En cambio, en TCP se prefiere "segmento" debido a su estructura orientada a la conexión.

UDP-> datagrama
TCP-> segmento

6. Describa el saludo de tres vías de TCP. ¿UDP tiene esta característica?
El saludo de tres vías de TCP, o *three-way handshake*, consiste en tres pasos: (1) el cliente envía un paquete SYN al servidor, (2) el servidor responde con un SYN-ACK, y (3) el cliente confirma con un ACK. Esto establece una conexión confiable entre ambos extremos. UDP no tiene esta característica, ya que es un protocolo sin conexión y no requiere la configuración previa de una conexión.



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

El **ISN** es el número de secuencia inicial que TCP asigna al inicio de una conexión. Durante el saludo de tres vías, el cliente y el servidor intercambian sus ISN, lo cual permite mantener la confiabilidad en la secuencia de datos. Cada extremo utiliza este número para ordenar correctamente los segmentos de datos que recibirá.

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

El **Maximum Segment Size (MSS)** es el tamaño máximo de un segmento de datos que una conexión TCP puede manejar. El MSS se negocia durante el saludo de tres vías cuando ambos extremos comunican su tamaño máximo de segmento. Esto permite que ambos dispositivos acuerden un tamaño de segmento que evitará la fragmentación en la red y optimizará la transmisión de datos.

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

a. Para listar las comunicaciones TCP establecidas.

```
redes@debian:~$ ss -t state established
Recv-Q   Send-Q     Local Address:Port         Peer Address:Port    Process
0        0              10.0.2.15:43332          31.13.94.24:https
0        0              10.0.2.15:56104       34.107.221.82:http
0        0              10.0.2.15:42662       34.107.243.93:https
0        0              10.0.2.15:48052      200.125.79.225:http
0        0              10.0.2.15:48078      200.125.79.225:http
0        0              10.0.2.15:58956      34.149.100.209:https
0        0              10.0.2.15:49904        192.16.49.85:http
0        0              10.0.2.15:48070      200.125.79.225:http
0        0              10.0.2.15:56106       34.107.221.82:http
0        0              10.0.2.15:54158       34.117.121.53:https
0        0              10.0.2.15:48060      200.125.79.225:http
0        0              10.0.2.15:53678      172.217.172.67:http
0        0              10.0.2.15:53676      172.217.172.67:http
0        0              10.0.2.15:38538      34.160.144.191:https
```

- **Recv-Q**: Muestra la cantidad de datos en la cola de recepción que aún no han sido procesados por la aplicación local.
- **Send-Q**: Muestra la cantidad de datos en la cola de envío que esperan ser enviados al destino.
- **Local Address**
  Indica la dirección IP y el puerto local en tu máquina que está utilizando la conexión.
- **Peer Address**
  Muestra la dirección IP y el puerto del dispositivo remoto al que tu PC está conectada.
- **Process**: Muestra el proceso (o aplicación) asociado a esa conexión, si se usa el parámetro -p al ejecutar el comando.
-

b. Para listar las comunicaciones UDP establecidas.

```
redes@debian:~$ ss -u
Recv-Q     Send-Q              Local Address:Port              Peer Address:Port       Process
0          0                10.0.2.15%enp0s3:bootpc                 10.0.2.2:bootps
```

c. Obtener sólo los servicios TCP que están esperando comunicaciones

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

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

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

Netstat:

a. Para listar las comunicaciones TCP establecidas.

```
redes@debian:~$ netstat -t -n
Active Internet connections (w/o servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 10.0.2.15:43332         31.13.94.24:443         ESTABLISHED
tcp        0      0 10.0.2.15:56104         34.107.221.82:80        ESTABLISHED
tcp        0      0 10.0.2.15:42662         34.107.243.93:443       ESTABLISHED
tcp        0      0 10.0.2.15:48052         200.125.79.225:80       ESTABLISHED
tcp        0      0 10.0.2.15:48078         200.125.79.225:80       ESTABLISHED
tcp        0      0 10.0.2.15:58956         34.149.100.209:443      ESTABLISHED
tcp        0      0 10.0.2.15:36034         20.201.28.144:443       TIME_WAIT
tcp        0      0 10.0.2.15:49904         192.16.49.85:80         ESTABLISHED
tcp        0      0 10.0.2.15:36022         20.201.28.144:443       TIME_WAIT
tcp        0      0 10.0.2.15:48070         200.125.79.225:80       ESTABLISHED
tcp        0      0 10.0.2.15:56106         34.107.221.82:80        ESTABLISHED
tcp        0      0 10.0.2.15:54158         34.117.121.53:443       ESTABLISHED
tcp        0      0 10.0.2.15:48060         200.125.79.225:80       ESTABLISHED
tcp        0      0 10.0.2.15:53678         172.217.172.67:80       TIME_WAIT
tcp        0      0 10.0.2.15:36050         20.201.28.144:443       TIME_WAIT
tcp        0      0 10.0.2.15:53676         172.217.172.67:80       TIME_WAIT
tcp        0      0 10.0.2.15:38538         34.160.144.191:443      ESTABLISHED
tcp        0      0 10.0.2.15:36392         35.201.103.21:443       ESTABLISHED
tcp        0      0 10.0.2.15:46910         34.98.75.36:443         ESTABLISHED
```

b. Para listar las comunicaciones UDP establecidas.

```
redes@debian:~$ netstat -u -n
Active Internet connections (w/o servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
udp        0      0 10.0.2.15:68            10.0.2.2:67             ESTABLISHED
```

Me da paja ver el resto :)

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, el puerto destino no está en estado LISTEN)?

El host **responde con un paquete TCP con el flag RST**, lo que indica que no hay ningún servicio escuchando en ese puerto.

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

```
redes@debian:~$ sudo hping3 -S -p 22 localhost
HPING localhost (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=65495 rtt=3.4 ms
len=44 ip=127.0.0.1 ttl=64 DF id=0 sport=22 flags=SA seq=1 win=65495 rtt=6.1 ms
len=44 ip=127.0.0.1 ttl=64 DF id=0 sport=22 flags=SA seq=2 win=65495 rtt=6.1 ms
len=44 ip=127.0.0.1 ttl=64 DF id=0 sport=22 flags=SA seq=3 win=65495 rtt=1.0 ms
len=44 ip=127.0.0.1 ttl=64 DF id=0 sport=22 flags=SA seq=4 win=65495 rtt=8.2 ms
len=44 ip=127.0.0.1 ttl=64 DF id=0 sport=22 flags=SA seq=5 win=65495 rtt=4.1 ms
len=44 ip=127.0.0.1 ttl=64 DF id=0 sport=22 flags=SA seq=6 win=65495 rtt=3.9 ms
len=44 ip=127.0.0.1 ttl=64 DF id=0 sport=22 flags=SA seq=7 win=65495 rtt=5.6 ms
len=44 ip=127.0.0.1 ttl=64 DF id=0 sport=22 flags=SA seq=8 win=65495 rtt=7.1 ms
len=44 ip=127.0.0.1 ttl=64 DF id=0 sport=22 flags=SA seq=9 win=65495 rtt=6.1 ms
len=44 ip=127.0.0.1 ttl=64 DF id=0 sport=22 flags=SA seq=10 win=65495 rtt=3.6 ms
```

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

```
redes@debian:~$ sudo hping3 -S -p 40 localhost
HPING localhost (lo 127.0.0.1): S set, 40 headers + 0 data bytes
len=40 ip=127.0.0.1 ttl=64 DF id=0 sport=40 flags=RA seq=0 win=0 rtt=8.0 ms
len=40 ip=127.0.0.1 ttl=64 DF id=0 sport=40 flags=RA seq=1 win=0 rtt=2.9 ms
len=40 ip=127.0.0.1 ttl=64 DF id=0 sport=40 flags=RA seq=2 win=0 rtt=6.7 ms
len=40 ip=127.0.0.1 ttl=64 DF id=0 sport=40 flags=RA seq=3 win=0 rtt=14.9 ms
len=40 ip=127.0.0.1 ttl=64 DF id=0 sport=40 flags=RA seq=4 win=0 rtt=7.1 ms
len=40 ip=127.0.0.1 ttl=64 DF id=0 sport=40 flags=RA seq=5 win=0 rtt=5.6 ms
len=40 ip=127.0.0.1 ttl=64 DF id=0 sport=40 flags=RA seq=6 win=0 rtt=8.6 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=8.2 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)

Puerto 22:
El host respondió con un paquete **SYN-ACK** (indicado por flags=SA), lo que significa que hay un servicio escuchando en el puerto 22 (probablemente el servicio SSH). Esto implica que el proceso que está a cargo de manejar las conexiones en este puerto está activo y aceptando conexiones.

Puerto 40:
En este caso, el host respondió con un paquete **RST-ACK** (indicado por flags=RA), lo que significa que el puerto 40 no tiene ningún servicio escuchando, y el sistema rechazó la conexión. Esto es un comportamiento normal para puertos cerrados o no utilizados.

11.¿Qué sucede si llega un datagrama UDP a un host que no tiene ningún proceso esperando en el puerto destino de dicho datagrama (es decir, que dicho puerto no está en estado LISTEN)?
Si no hay un proceso escuchando en el puerto destino, el host generalmente no responderá al datagrama UDP. Esto es diferente de lo que ocurre con TCP, donde se envían respuestas RST para indicar que el puerto está cerrado.
En algunos casos, el sistema operativo puede enviar un mensaje ICMP (Internet Control Message Protocol) de "Puerto Inalcanzable" al remitente para informar que el puerto está cerrado o no disponible. Sin embargo, esto no es garantizado y puede depender de la configuración de la pila de red del sistema operativo.

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

```
redes@debian:~$ sudo hping3 -2 -p 5353 localhost
HPING localhost (lo 127.0.0.1): udp mode set, 28 headers + 0 data bytes
^C
--- localhost hping statistic ---
39 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.

```
redes@debian:~$ sudo hping3 -2 -p 40 localhost
HPING localhost (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=1178 seq=0
ICMP Port Unreachable from ip=127.0.0.1 name=localhost
status=0 port=1179 seq=1
ICMP Port Unreachable from ip=127.0.0.1 name=localhost
status=0 port=1180 seq=2
ICMP Port Unreachable from ip=127.0.0.1 name=localhost
status=0 port=1181 seq=3
ICMP Port Unreachable from ip=127.0.0.1 name=localhost
status=0 port=1182 seq=4
ICMP Port Unreachable from ip=127.0.0.1 name=localhost
status=0 port=1183 seq=5
```

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).

En el primer caso se mandaron paquetes pero no se recibió ninguna respuesta. No hay servicios escuchando en ese puerto. Cuando mandé al 40 me respondió ICMP que como expliqué antes es puerto inalcanzable. No hay ningún servicio escuchando en él y el puerto está cerrado.

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
    1. **CLOSED (Cerrado)**:
        - Estado inicial de la conexión. La conexión TCP no está establecida ni está en uso.
    2. **LISTEN (Escuchando)**:
        - Estado en el que un servidor espera conexiones entrantes. El socket está preparado para aceptar conexiones desde clientes.
    3. **SYN-SENT (SYN Enviado)**:
        - Estado en el que el cliente ha enviado un segmento SYN para iniciar una conexión y está esperando un segmento SYN-ACK a cambio.
    4. **SYN-RECEIVED (SYN Recibido)**:
        - Estado en el que el servidor ha recibido un segmento SYN de un cliente y ha respondido con un SYN-ACK. Está esperando el segmento ACK de vuelta del cliente.
    5. **ESTABLISHED (Establecido)**:
        - Estado en el que la conexión está completamente establecida y ambos extremos pueden enviar y recibir datos. Es el estado activo de la conexión TCP.
    6. **FIN-WAIT-1 (Esperando FIN-1)**:
        - Estado en el que el host ha enviado un segmento FIN (finalizar) y está esperando la respuesta ACK del otro extremo.
    7. **FIN-WAIT-2 (Esperando FIN-2)**:
        - Estado en el que el host ha recibido un ACK para su FIN y está esperando un segmento FIN del otro extremo.
    8. **CLOSING (Cerrando)**:
        - Estado en el que ambos extremos han enviado segmentos FIN y están esperando la confirmación ACK.
    9. **TIME-WAIT (Tiempo de espera)**:
        - Estado en el que el host espera para asegurarse de que el otro extremo haya recibido su último ACK. Este estado se mantiene durante un tiempo específico (generalmente dos veces el tiempo de vida máximo de un segmento).
    10. **CLOSE-WAIT (Espera de cierre)**:
        - Estado en el que el host ha recibido un segmento FIN del otro extremo y está esperando que la aplicación cierre la conexión.
    11. **LAST-ACK (Último ACK)**:
        - Estado en el que el host ha enviado un segmento FIN y está esperando el ACK correspondiente del otro extremo.

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

```
Netid State      Recv-Q Send-Q Local Address:Port         Peer Address:Port       Process
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-WAI  T 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 están establecidas actualmente? 9

b. ¿Cuántos puertos están abiertos a la espera de posibles nuevas conexiones? 5 (22,80,53,25)

c. ¿El cliente y el servidor de las comunicaciones HTTPS (puerto 443) residen en la misma máquina? No, no lo hacen. Uno en 200.115.89.30, 64.223.190.99,200.115.89.15, etc.

d. ¿El cliente y el servidor de la comunicación SSH (puerto 22) residen en la misma máquina? Si. hay dos conexiones SSH (127.0.0.1:22) entre procesos sshd y ssh, que indican que ambos residen en la misma máquina (localhost).

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

- **Proceso Servidor**:
    - sshd (Puerto 22, PID 468)
    - apache2 (Puerto 80, PID 991)
    - named (Puerto 53, PID 452)
    - postfix (Puerto 25, PID 627)

  Me doy cuenta que es sv pq escucha las solicitudes de un puerto específico y espera a que los clientes se conecten. Puede manejar varias conexiones en simultáneo
- **Proceso Cliente**:
    - x-www-browser (Varios puertos 443, PID 1079)
    - ssh (Puerto 22, PID 1415)

  Me doy cuenta que es cliente pq inicia la conexión a un sv y solicita servicios. En el ejemplo sshd en el puerto 41220 creo que está actuando como cliente y se conecta al servidor sshd de la misma máquina

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

- **CLOSE-WAIT**: Este estado indica que el lado que ha recibido el cierre de la conexión está esperando que el socket se cierre (es decir, el host local está en proceso de cerrar la conexión). Esto significa que el cierre fue iniciado por el remoto.
- **TIME-WAIT**: Este estado es un período de espera después de que una conexión ha sido cerrada para asegurarse de que cualquier paquete retrazado no cause confusión. Una conexión que entra en este estado normalmente ha sido cerrada por el host local, y el host remoto ha enviado un segmento FIN (finalización).

Así que
Cierre iniciado por el host local
- Conexión en estado TIME-WAIT: 163.10.5.222:36676

- Cierre iniciado por el remoto
  - Conexión en estado CLOSE-WAIT: 163.10.5.222:41654

La identificación de quién inició el cierre de la conexión se basa en el estado de la conexión y la interpretación de esos estados según la definición de los protocolos TCP. Esto te permite entender cómo fluyen los datos y cómo se cierran las conexiones en tu sistema.

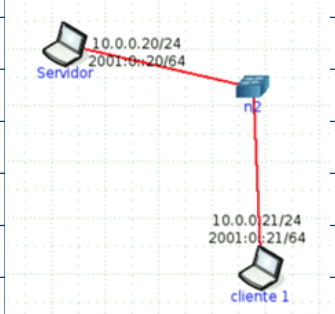
No entendí muy bien esto je
g. ¿Cuántas conexiones están aún pendientes por establecerse? 1. La que está en SYN-SENT (163.10.5.222:52132)

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

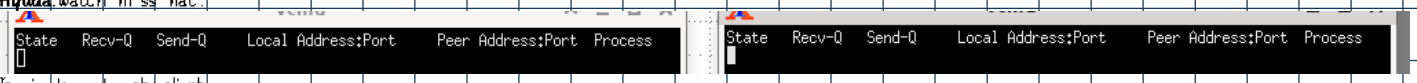
```
servidor# ss -natu | grep 110
tcp    LISTEN 0 0                  *:110              *:*
tcp    SYN-RECV       0 0        157.0.0.1:110  157.0.11.1:52843

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?
El cliente mandó la solicitud de synt al servidor, el servidor la recibe. No se llega a establecer la conexión. Falta la 3er parte del 3 way handshake
b. ¿A qué protocolo de capa de aplicación y de transporte se está intentando conectar el cliente? Se está intentando conectar a tcp
c. ¿Qué flags tendría seteado el segmento perdido? Tendría SYN y ACK prendidos para confirmar la recepción del syn del servidor.

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

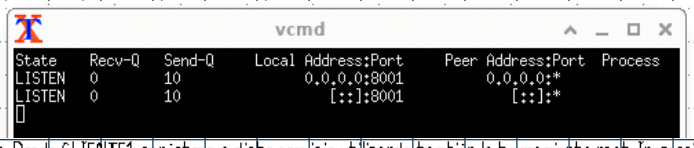


a. En ambos equipos, inspeccionar el estado de las conexiones y mantener abiertas ambas ventanas con el comando corriendo para poder visualizar los cambios a medida que se realiza el ejercicio.
Ayuda: watch -n1 'ss -nat'.

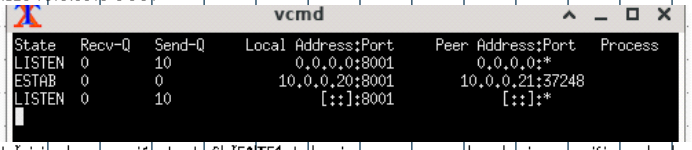


Izquierda sv, derecha cliente
Watch -n1 hace que se repita muchas veces lo de la derecha.
N muestra las ip y puertos en formato numérico
A muestra todas las conexiones
T muestra solo tcp
b. En el servidor, utilice la herramienta ncat para levantar un servicio que escuche en el puerto 8001/TCP. Utilice la opción -k para que el servicio sea persistente. Verifique el estado de las conexiones.
sudo ncat -l -k -p 8001

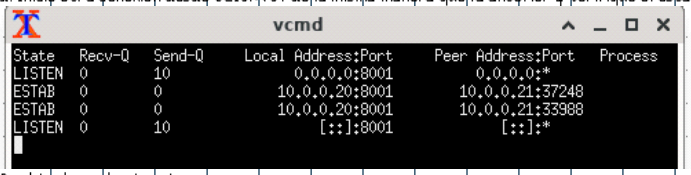
- -l indica que ncat debe escuchar (listen) en el puerto especificado.
- -k permite que el servicio sea persistente, es decir, que permanezca abierto para múltiples conexiones sucesivas.
- -p 8001 especifica el puerto 8001 para el servicio.



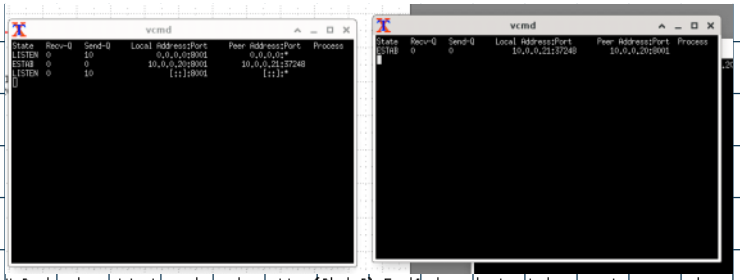
c. Desde CLIENTE1, conéctese a dicho servicio utilizando también la herramienta ncat. Inspeccione el estado de las conexiones.
ncat 10.0.0.20 8001



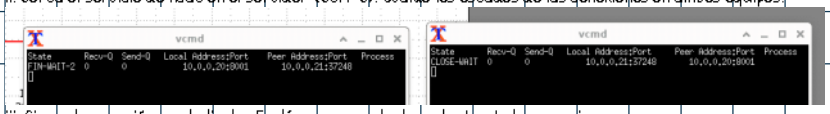
d. Inicie otra conexión desde CLIENTE1 de la misma manera que la anterior y verifique el estado de las conexiones. ¿De qué manera puede identificar cada conexión?



Cambia el puerto de origen
e. En base a lo observado en el ítem 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?
Sí, se puede. Como tienen puertos de origen distintos se puede tener más de una conexión desde el mismo cliente y al mismo puerto destino. Se identifican unicocamente usando la combinación de la ip + nro de puerto local
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.