1. La **capa de transporte** es la **encargada de proporcionar comunicación lógica entre procesos** que se ejecutan en diferentes sistemas finales (hosts). Esto quiere decir que va más allá de simplemente mover bits entre computadoras (como hace la capa de red); su trabajo es **asegurar que un programa en un host pueda comunicarse correctamente con otro programa en otro host**. Se encarga de ampliar el servicio de entrega de la capa de red entre dos sistemas terminales a un servicio de entrega entre dos procesos de la capa de aplicación que se ejecutan en los sistemas terminales. Los protocolos de la capa de transporte residen en los sistemas terminales. Dentro de un sistema terminal, el protocolo de transporte lleva los mensajes desde los procesos de la aplicación a la frontera de la red y viceversa.
2. Datagrama UDP
    - Puerto Origen (16bits)
    - Puerto Destino (16 bits)
    - Longitud
    - UDP Checksum
    - Data
      Cada número de puerto es un número de 16 bits comprendido en el rango de 0 a 65535. Los puertos pertenecientes al rango 0 a 1023 se conocen como números de puerto bien conocidos y son restringidos. Los números de puerto permiten al host de destino pasar los datos de la aplicación al proceso apropiado que está ejecutándose en el sistema terminal de destino (demultiplexación). El host receptor utiliza la suma de comprobación para detectar si se han introducido errores en el segmento (el campo checksum almacena el resultado). Se utiliza para determinar si los bits contenidos en el segmento UDP han sido alterados según se desplazaban desde el origen hasta el destino. Todos los campos tienen una longitud de 2 bytes.
      Segmento TCP
    - Puerto Origen (16 bits)
    - Puerto Destino (16 bits)
    - Nro de Secuencia (32 bits)
    - Nro de Conocimiento (32 bits)
    - Longitud de encabezado (4 bits)
    - Flags
    - Tamaño de Ventana (16 bits)
    - TCP Checksum (16 bits)
    - Puntero Urgente (16 bits)
    - Opciones

- Data
  El campo de datos contiene un fragmento de los datos de la aplicación. Al igual que UDP tiene número de puerto de origen y de destino (se usar para multiplexar y demultiplexar datos). También incluye un campo de suma de comprobación. El campo de número de secuencia de 32 bits y el campo de número de reconocimiento también de 32 bits son utilizados por el emisor y el receptor de TCP para implementar un servicio de transferencia de datos fiable. La ventana de recepción de 16 bits se utiliza para el control de flujo. El campo longitud de cabecera de 4 bits especifica la longitud de la cabecera TCP en palabras de 32 bits. El campo opciones es opcional y de longitud variable. Se utiliza cuando un emisor y un receptor negocian el tamaño máximo de segmento (MSS) o como un factor de escala de la ventana en las redes de alta velocidad. El campo indicador tiene 6 bits. El bit ACK se utiliza para indicar que el valor transportado en el campo de reconocimiento es válido. Los bits RST, SYN y FIN se utilizan para establecimiento y cierre de conexiones. El URG se utiliza para indicar que hay datos en este segmento que la entidad de la capa superior del lado emisor ha marcado como "urgentes".

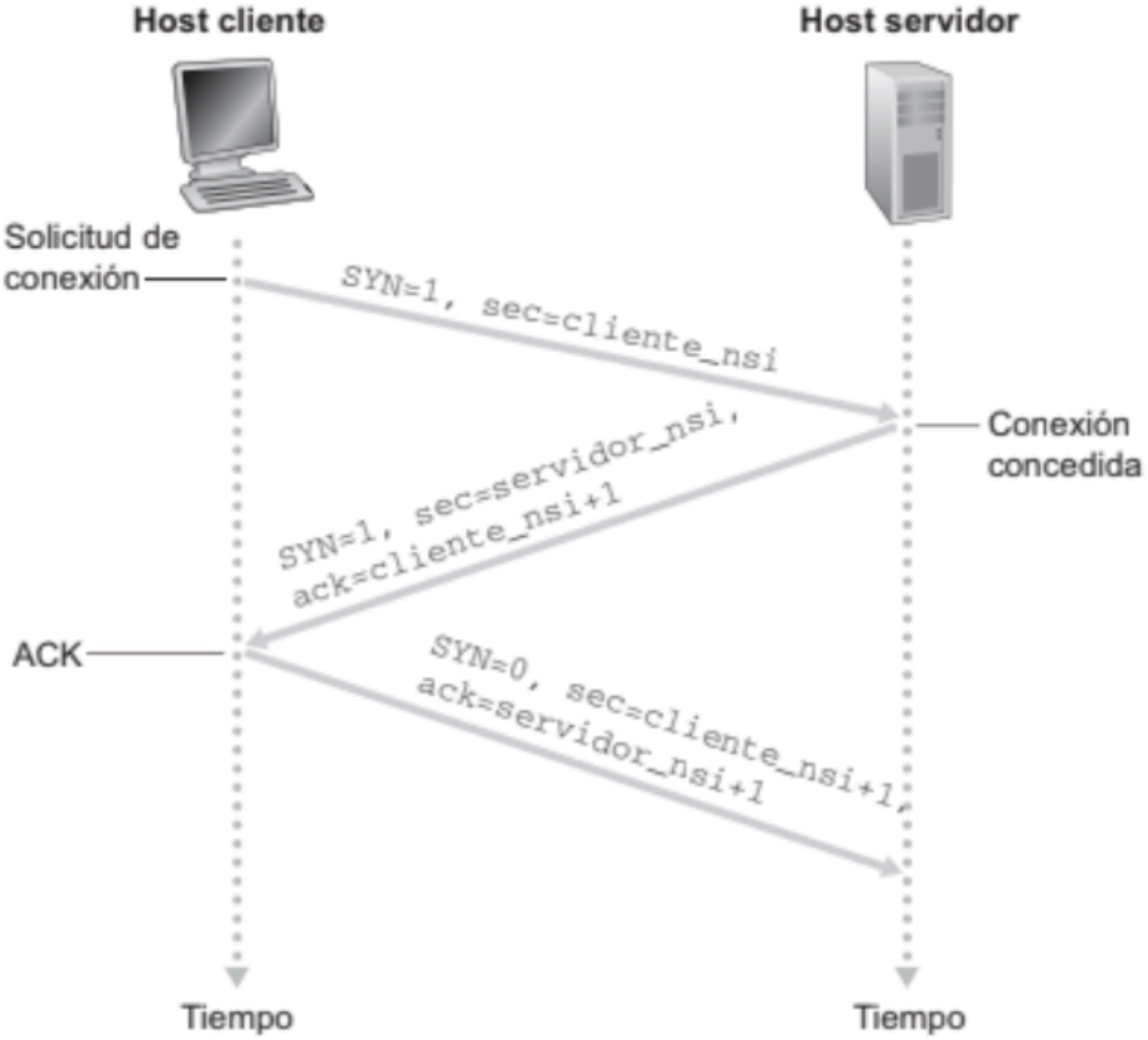
3. El principal objetivo del uso de puertos es identificar a qué proceso (programa) dentro de un host deben entregarse los datos que llegan a través de la red. Los puertos permiten que múltiples aplicaciones puedan usar la red simultáneamente sin confundirse entre sí.
   Ej: Imagina que en tu computadora estás:
   - Navegando por Internet (usando un navegador).
   - Escuchando música desde Spotify.
   - Haciendo una videollamada por Zoom.
     Todos estos programas están **enviando y recibiendo datos por la red al mismo tiempo**, a través de la **misma dirección IP** de tu computadora. Sin puertos, no habría forma de saber a qué aplicación debe entregarse cada segmento de datos que llega.
4.

Característica	**TCP (Transmission Control Protocol)**	**UDP (User Datagram Protocol**
Confiabilidad	✅ **Sí.** TCP garantiza la entrega de datos sin errores, en orden y sin duplicados (mediante ACKs, retransmisiones y números de secuencia).	❌ **No.** UDP no garantiza entrega, orden ni control de errores. Los paquetes pueden perderse o llegar duplicados.
Multiplexación	✅ Sí. TCP usa **números de puerto** (origen/destino) para distinguir múltiples conexiones en un mismo host.	✅ También. UDP también utiliza **números de puerto** para distinguir procesos. Ambos protocolos usan esta técnica.

Característica	TCP (Transmission Control Protocol)	UDP (User Datagram Protocol
Orientado a la conexión	✅ **Sí.** TCP es **orientado a la conexión**, lo que significa que se establece una conexión (handshake de 3 vías) antes de transferir datos.	❌ **No.** UDP es **no orientado a la conexión**. Los datos se envían sin establecer conexión previa.
Controles de congestión	✅ **Sí.** TCP implementa **control de congestión** (como el algoritmo de ventana deslizante, control lento, etc.).	❌ **No.** UDP **no implementa ningún control de congestión**. Envía datos tan rápido como la aplicación quiera.
Utilización de puertos	✅ **Sí.** Usa puertos para identificar procesos en el host. Los puertos son esenciales para TCP.	✅ **Sí.** También usa puertos. Ambos protocolos dependen del sistema de puertos del modelo TCP/IP.

5. Una PDU (Unidad de Datos de Protocolo) es la forma en la que una capa maneja los datos para transmitirlos o recibirlos. El término segmento se utiliza cuando hablamos de la capa de transporte con el protocolo TCP. Esto se debe a que TCP divide los datos de la aplicación en segmentos, les añade encabezados con números de secuencia, puertos, etc., y los gestiona de forma confiable. El término datagrama se usa para referirse a la PDU de la capa de transporte cuando se utiliza el protocolo UDP. A diferencia de TCP, UDP es no orientado a la conexión y sus unidades de datos se llaman datagramas UDP.
6. El saludo de tres vías es el proceso que TCP utiliza para establecer una conexión confiable entre dos hosts antes de enviar datos. Este mecanismo garantiza que ambos extremos estén preparados y sincronizados para la comunicación. El protocolo **UDP (User Datagram Protocol) no tiene saludo de tres vías** ni ningún tipo de establecimiento de conexión (ya que es un protocolo no orientado a la conexión).
   SYN (Synchronize):
   - El cliente envía un segmento con el **bit SYN** activado y un número de secuencia inicial `x`.
   - Esto indica que desea iniciar una conexión.
     SYN-ACK (Synchronize + Acknowledge):
   - El servidor responde con un segmento que tiene activados los bits **SYN y ACK**.
   - El servidor reconoce (`ACK = x + 1`) el número de secuencia del cliente y envía su propio número de secuencia `y`.
     ACK (Acknowledge):
   - El cliente responde con un segmento que **confirma el número de secuencia del servidor** (`ACK = y + 1`).

- A partir de aquí, la conexión TCP está establecida y lista para transmitir datos.



7. El ISN es el número de secuencia inicial que cada host TCP elige al comenzar una nueva conexión.
   Su propósito es:

   - Numerar los bytes que se transmitirán a partir de ese punto.
   - Identificar de forma única los segmentos dentro de una conexión.
   - Prevenir la reutilización de números de secuencia que podrían causar confusión si hay segmentos antiguos en la red (protección contra ataques y errores).
     TCP no utiliza siempre el mismo número inicial. El ISN se elige de manera pseudoaleatoria, para proteger la conexión contra ciertos tipos de ataques, como la suplantación de direcciones (spoofing).

   Pasos del saludo de tres vías con ISN
   Cliente → Servidor: SYN
   - El cliente envía un segmento con el bit SYN = 1 y su ISN, por ejemplo ISN_C = 1000.
   - No contiene datos todavía.

- Indica que quiere iniciar la conexión.
  Servidor → Cliente: SYN-ACK
- El servidor responde con SYN = 1, ACK = ISN_C + 1 = 1001, y envía su propio ISN, por ejemplo ISN_S = 5000.
  Cliente → Servidor: ACK
- El cliente envía un ACK con ACK = ISN_S + 1 = 5001, confirmando la recepción del ISN del servidor.

8. **MSS (Maximum Segment Size)** es el **tamaño máximo de datos (en bytes)** que un host está dispuesto a recibir en un **segmento TCP**, **excluyendo el encabezado TCP e IP**.
   - **No incluye encabezado TCP (20 bytes mínimo)** ni **encabezado IP (20 bytes mínimo)**.
   - Se refiere **solo a la parte útil de datos**, también llamada **carga útil (payload)**.
     Se negocia durante el saludo de tres vías (three-way handshake) de TCP:
     👷 Cliente → Servidor (SYN):
   - El cliente incluye en el segmento SYN una opción TCP indicando su MSS (por ejemplo, 1460 bytes).
     🖥️ Servidor → Cliente (SYN-ACK):
   - El servidor también puede incluir su propio MSS como opción TCP en el SYN-ACK.
     ✅ Luego de esto:
   - Ambos lados conocen el MSS del otro.
   - Cada uno ajustará sus segmentos de datos al MSS del receptor.
9. A. Listar las comunicaciones TCP establecidas

```
redes@debian:~$ ss -t -a state established
Recv-Q    Send-Q       Local Address:Port       Peer Address:Port    Process
redes@debian:~$ netstat -tn
Active Internet connections (w/o servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
```

B. Listar las comunicaciones UDP establecidas

```
redes@debian:~$ ss -u -a
State   Recv-Q  Send-Q      Local Address:Port     Peer Address:Port   Process
UNCONN  0       0                 0.0.0.0:631         0.0.0.0:*
UNCONN  0       0                 0.0.0.0:51999       0.0.0.0:*
UNCONN  0       0               127.0.0.1:4038        0.0.0.0:*
ESTAB   0       0       10.0.2.15%enp0s3:bootpc      10.0.2.2:bootps
UNCONN  0       0                 0.0.0.0:mdns        0.0.0.0:*
UNCONN  0       0                    [::]:59939          [::]:*
UNCONN  0       0                    [::]:mdns           [::]:*
redes@debian:~$ netstat -un
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
```

C. Servicios TCP que están esperando comunicaciones (escuchando)

```
redes@debian:~$ ss -t -l
State   Recv-Q  Send-Q        Local Address:Port    Peer Address:Port  Process
LISTEN  0       5                 127.0.0.1:4038         0.0.0.0:*
LISTEN  0       128                 0.0.0.0:ssh          0.0.0.0:*
LISTEN  0       128               127.0.0.1:ipp          0.0.0.0:*
LISTEN  0       4096                 [::1]:50051            [::]:*
LISTEN  0       4096     [::ffff:127.0.0.1]:50051               *:*
LISTEN  0       128                  [::]:ssh             [::]:*
LISTEN  0       128                 [::1]:ipp             [::]:*
redes@debian:~$ netstat -tnl
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 127.0.0.1:4038          0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:631           0.0.0.0:*               LISTEN
tcp6       0      0 ::1:50051               :::*                    LISTEN
tcp6       0      0 127.0.0.1:50051         :::*                    LISTEN
tcp6       0      0 :::22                   :::*                    LISTEN
tcp6       0      0 ::1:631                 :::*                    LISTEN
```

D. Servicios UDP que están esperando comunicaciones (escuchando)

```
redes@debian:~$ ss -u -l
State   Recv-Q   Send-Q     Local Address:Port      Peer Address:Port  Process
UNCONN  0        0               0.0.0.0:631            0.0.0.0:*
UNCONN  0        0               0.0.0.0:51999          0.0.0.0:*
UNCONN  0        0             127.0.0.1:4038           0.0.0.0:*
UNCONN  0        0               0.0.0.0:mdns           0.0.0.0:*
UNCONN  0        0                  [::]:59939             [::]:*
UNCONN  0        0                  [::]:mdns              [::]:*
redes@debian:~$ netstat -unl
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
udp        0      0 0.0.0.0:631             0.0.0.0:*
udp        0      0 0.0.0.0:51999           0.0.0.0:*
udp        0      0 127.0.0.1:4038          0.0.0.0:*
udp        0      0 0.0.0.0:5353            0.0.0.0:*
udp6       0      0 :::59939                :::*
udp6       0      0 :::5353                 :::*
```

10. Cuando un host recibe un segmento TCP con SYN activado en un puerto donde no hay ningún proceso escuchando (no está en estado LISTEN), el sistema responde con un segmento TCP con el flag RST (Reset) activado. Esto se debe a que el host quiere indicar que no hay ninguna aplicación interesada en esa conexión, y por tanto la conexión debe cerrarse inmediatamente.

    a. Enviar un paquete TCP con `SYN` al puerto 22 (SSH)

```
redes@debian:~$ sudo hping3 -S -p 22 127.0.0.1
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=65495 rtt=10.4 ms
len=44 ip=127.0.0.1 ttl=64 DF id=0 sport=22 flags=SA seq=1 win=65495 rtt=1.2 ms
len=44 ip=127.0.0.1 ttl=64 DF id=0 sport=22 flags=SA seq=2 win=65495 rtt=8.2 ms
len=44 ip=127.0.0.1 ttl=64 DF id=0 sport=22 flags=SA seq=3 win=65495 rtt=4.0 ms
len=44 ip=127.0.0.1 ttl=64 DF id=0 sport=22 flags=SA seq=4 win=65495 rtt=8.5 ms
len=44 ip=127.0.0.1 ttl=64 DF id=0 sport=22 flags=SA seq=5 win=65495 rtt=4.9 ms
len=44 ip=127.0.0.1 ttl=64 DF id=0 sport=22 flags=SA seq=6 win=65495 rtt=7.7 ms
len=44 ip=127.0.0.1 ttl=64 DF id=0 sport=22 flags=SA seq=7 win=65495 rtt=3.1 ms
len=44 ip=127.0.0.1 ttl=64 DF id=0 sport=22 flags=SA seq=8 win=65495 rtt=6.4 ms
len=44 ip=127.0.0.1 ttl=64 DF id=0 sport=22 flags=SA seq=9 win=65495 rtt=6.9 ms
```

    Se recibe respuesta con flag SYN-ACK.

b. Enviar un paquete TCP con `SYN` al puerto 40 (habitualmente cerrado)

```
redes@debian:~$ sudo hping3 -S -p 40 127.0.0.1
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=0 sport=40 flags=RA seq=0 win=0 rtt=7.5 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=7.0 ms
len=40 ip=127.0.0.1 ttl=64 DF id=0 sport=40 flags=RA seq=3 win=0 rtt=1.9 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=4.2 ms
len=40 ip=127.0.0.1 ttl=64 DF id=0 sport=40 flags=RA seq=6 win=0 rtt=3.3 ms
len=40 ip=127.0.0.1 ttl=64 DF id=0 sport=40 flags=RA seq=7 win=0 rtt=5.9 ms
len=40 ip=127.0.0.1 ttl=64 DF id=0 sport=40 flags=RA seq=8 win=0 rtt=2.0 ms
len=40 ip=127.0.0.1 ttl=64 DF id=0 sport=40 flags=RA seq=9 win=0 rtt=1.0 ms
```

Se recibe respuesta con flag RST.

11. - UDP es **no orientado a conexión**, por lo tanto el host **no envía automáticamente un mensaje como TCP (no hay RST)**. Sin embargo, si el **puerto destino está cerrado** y el sistema recibe el datagrama, **puede responder con un mensaje ICMP del tipo "Port Unreachable"**. Este comportamiento depende del sistema operativo y configuración del firewall. Si el **puerto está abierto y hay un servicio escuchando**, el datagrama será entregado y **no se genera ninguna respuesta por defecto** (UDP no hace ACKs).

    a. Usar `hping3` para enviar datagramas UDP al puerto 5353

    ```
    redes@debian:~$ sudo hping3 --udp -p 5353 127.0.0.1
    HPING 127.0.0.1 (lo 127.0.0.1): udp mode set, 28 headers + 0 data bytes
    ```

    mDNS está activo → **no se ve respuesta** (UDP no responde si el puerto está abierto).

    b. Enviar datagramas UDP al puerto 40 (probablemente cerrado)

    ```
    redes@debian:~$ sudo hping3 --udp -p 40 127.0.0.1
    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=2476 seq=0
    ICMP Port Unreachable from ip=127.0.0.1 name=localhost
    status=0 port=2477 seq=1
    ICMP Port Unreachable from ip=127.0.0.1 name=localhost
    status=0 port=2478 seq=2
    ```

    Como **nada suele escuchar en el puerto 40**, obtenemos una respuesta ICMP indicando:**"Destination unreachable – Port unreachable"**

12. A connection progresses through a series of states during its lifetime. The states are: LISTEN, SYN-SENT, SYNRECEIVED, ESTABLISHED, FIN-WAIT-1, FIN-WAIT-2, CLOSE-WAIT, CLOSING, LAST-ACK, TIME-WAIT, and the fictional state CLOSED. CLOSED is fictional because it represents the state when there is no TCB, and therefore, no

connection. Briefly the meanings of the states are:

LISTEN	represents waiting for a connection request from any remote TCP and port.
SYN-SENT	represents waiting for a matching connection request after having sent a connection request.
SYN-RECEIVED	represents waiting for a confirming connection request acknowledgment after having both received and sent a connection request.
ESTABLISHED	represents an open connection, data received can be delivered to the user. The normal state for the data transfer phase of the connection.
FIN-WAIT-1	represents waiting for a connection termination request from the remote TCP, or an acknowledgment of the connection termination request previously sent.
FIN-WAIT-2	represents waiting for a connection termination request from the remote TCP.
CLOSE-WAIT	represents waiting for a connection termination request from the local user.
CLOSING	represents waiting for a connection termination request acknowledgment from the remote TCP.
LAST-ACK	represents waiting for an acknowledgment of the connection termination request previously sent to the remote TCP (which includes an acknowledgment of its connection termination request).
TIME-WAIT	represents waiting for enough time to pass to be sure the remote TCP received the acknowledgment of its connection termination request.
CLOSED	represents no connection state at all.

13. A-Las conexiones establecidas tienen el estado (State) ESTAB --> Hay 9 conexiones establecidas.
    B-Los puertos abiertos a la espera de posibles nuevas conexiones tienen el estado (State) LISTEN --> Hay 5 puertos en listen.
    C-No están en la misma máquina, el cliente tiene la IP 163.10.5.222 y se conecta a servidores remotos en el puerto 443.
    D-Sí, poseen la misma IP 127.0.0.1, residen en la misma máquina.
    E-

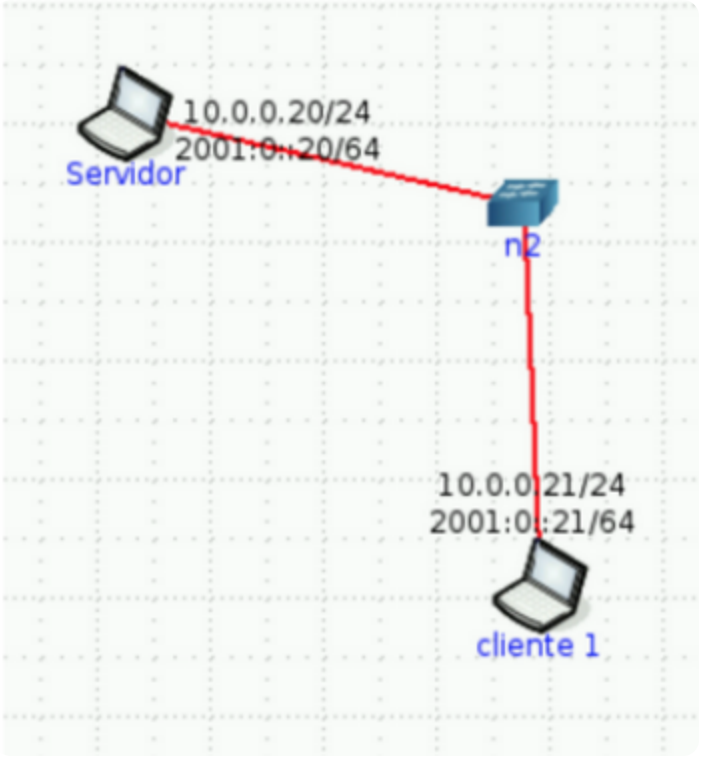
Puerto	Proceso	Rol
22	sshd	Servidor SSH
80	apache2	Servidor HTTP
25	postfix	Servidor SMTP
53	named	Servidor DNS
443	x-www-browser	Cliente HTTPS

Puerto	Proceso	Rol
22	ssh	Cliente SSH

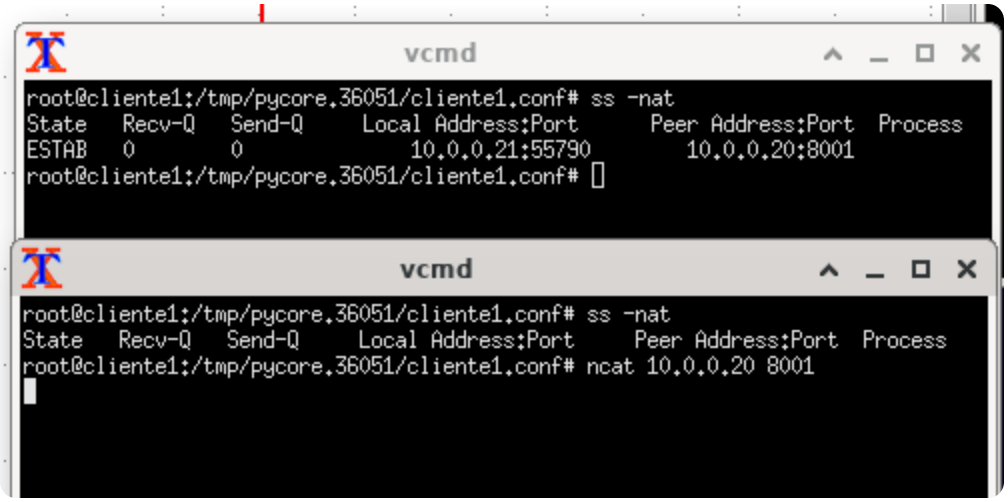
```
F-`TIME-WAIT`: indica que el **host local** inició el cierre. --> Hay una.
`CLOSE-WAIT`: indica que el **host remoto** inició el cierre. --> Hay una.
G- Las conexiones pendientes tienen el estado en SYN-SENT --> Hay una conexión aún pendiente.
```

14. A- El cliente está en estado SYN-SENT, lo que indica que envió un segmento con el flag SYN para iniciar la conexión. El servidor está en estado SYN-RECV, lo que indica que recibió correctamente el SYN del cliente y respondió con un segmento SYN-ACK, pero no ha recibido aún el ACK final del cliente.
    El segmento SYN del cliente llegó correctamente al servidor. El segmento SYN-ACK del servidor aparentemente se perdió en la red o no fue recibido por el cliente. Por eso el cliente permanece en SYN-SENT y el servidor en SYN-RECV.
    B- **Puerto 110** es estándar para **POP3** (Post Office Protocol v3), un protocolo de **capa de aplicación** usado para recibir correos electrónicos.
    - **Capa de transporte**: TCP
    - **Capa de aplicación**: POP3
      C-El segmento perdido es el que envió el servidor al cliente, en respuesta al SYN. Ese segmento es el segundo paso del saludo de tres vías. 📌 Por lo tanto, el segmento perdido tendría los siguientes flags:
    - SYN (para continuar el saludo)
    - ACK (para confirmar recepción del SYN del cliente)

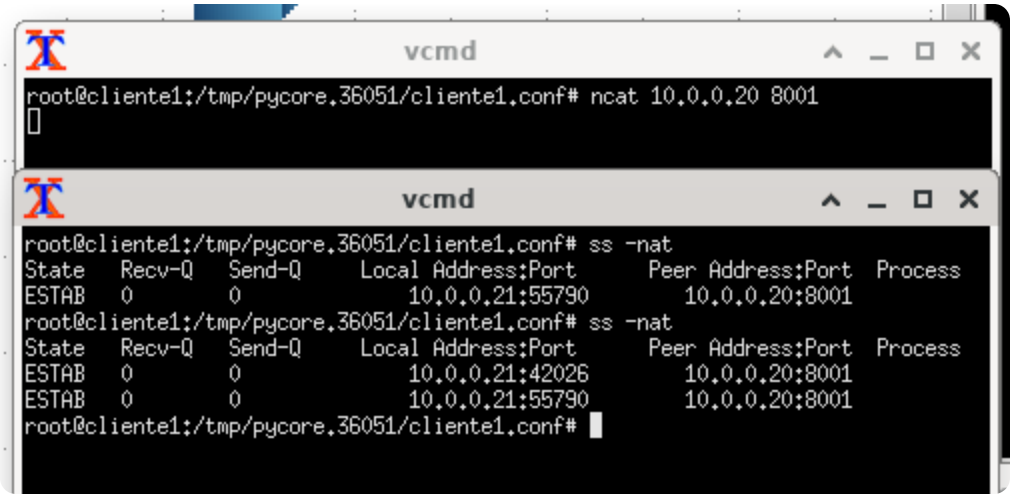
15. A/B-



C-



D-



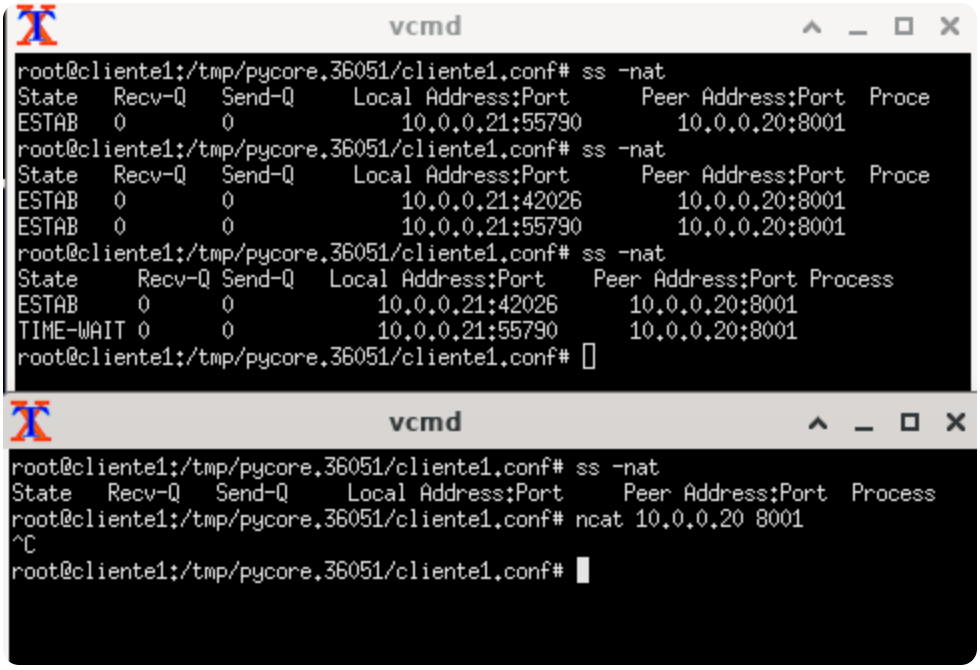
Los puertos son distintos.

E- Se puede a través de distintos puertos orígenes, por lo tanto, las identificaciones en base a las IP´s de origen y de destino y los números de puerto origen y destino siguen

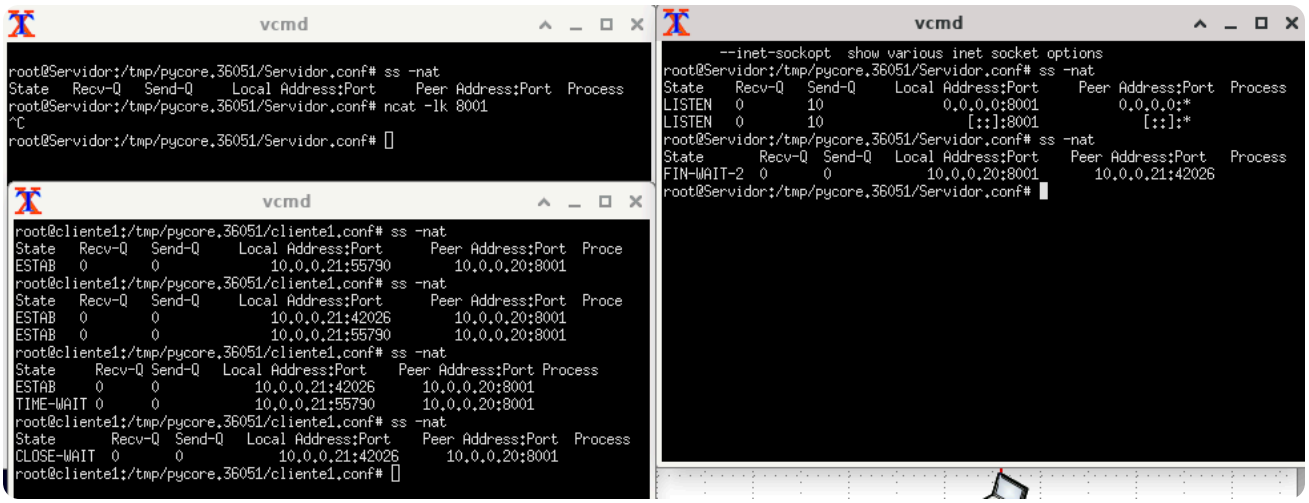
siendo únicas.

F-

I.



II.



III.