1) **¿Qué es un puerto?**

En el modelo OSI quien se preocupa de la administración de los puertos y los establece en el encabezado de los segmentos es la capa de transporte o capa 4, administrando así el envío y re ensamblaje de cada segmento enviado a la red haciendo uso del puerto especificado. Un puerto suele estar numerado para de esta forma poder identificar la aplicación que lo usa. Decidir a qué programa entregará los datos recibidos. Esta asignación de puertos permite a una máquina establecer simultáneamente diversas conexiones con máquinas distintas, ya que todos los segmentos que se reciben tienen la misma dirección, pero van dirigidos a puertos diferentes. Haciendo una analogía un puerto es como un interno en una comunicación telefónica, podemos llamar a la misma empresa con el mismo número, pero dependiendo a quien queremos contactar, marcamos un interno diferente.

Los números de puerto se indican mediante 16 bits (2 bytes), por lo que existen 65536 puertos, numerados del 0 al 65535. Aunque podemos usar cualquiera de ellos para cualquier protocolo, existe una entidad, la IANA, encargada de su asignación, la cual creó tres categorías:

- Puertos bien conocidos: Los puertos inferiores al 1024 son puertos reservados para el sistema operativo y usados por "protocolos bien conocidos" como por ejemplo HTTP (servidor Web), SMTP (servidor de e-mail) y Telnet.
- Puertos registrados: Los comprendidos entre 1024 y 49151 son denominados "registrados" y pueden ser usados por cualquier aplicación. Existe una lista pública en la web del IANA donde se puede ver qué protocolo usa cada uno de ellos.
- Puertos dinámicos: Los comprendidos entre los números 49152 y 65535 son denominados dinámicos o privados, normalmente se asignan en forma dinámica a las aplicaciones de clientes al iniciarse la conexión.

2) **¿Como estan formados los endpoints?**

Un “Endpoint” es la combinación entre la dirección IP y el Puerto, a través de un endpoint un cliente puede acceder a un servidor, o hacer la petición de conexión.

3) **¿Que es un socket?**

Socket designa un concepto abstracto por el cual dos programas (En la misma computadora o no) pueden intercambiar cualquier flujo de datos. Los sockets de Internet constituyen el mecanismo para la entrega de paquetes de datos provenientes de la tarjeta de red a los procesos o hilos apropiados. Un socket queda definido por un par de direcciones IP local y remota, un protocolo de transporte y un par de números de puerto local y remoto.

Para que dos programas puedan comunicarse entre sí es necesario que se cumplan ciertos requisitos:

- Que un programa sea capaz de localizar al otro.

- Que ambos programas sean capaces de intercambiarse datos relevantes a su finalidad.

Para ello son necesarios los dos recursos que originan el concepto de socket:

- Un par de direcciones del protocolo de red (dirección IP, si se utiliza el protocolo TCP/IP), que identifican la computadora de origen y la remota.
- Un par de números de puerto, que identifican a un programa dentro de cada computadora.

Los sockets permiten implementar una arquitectura cliente-servidor.

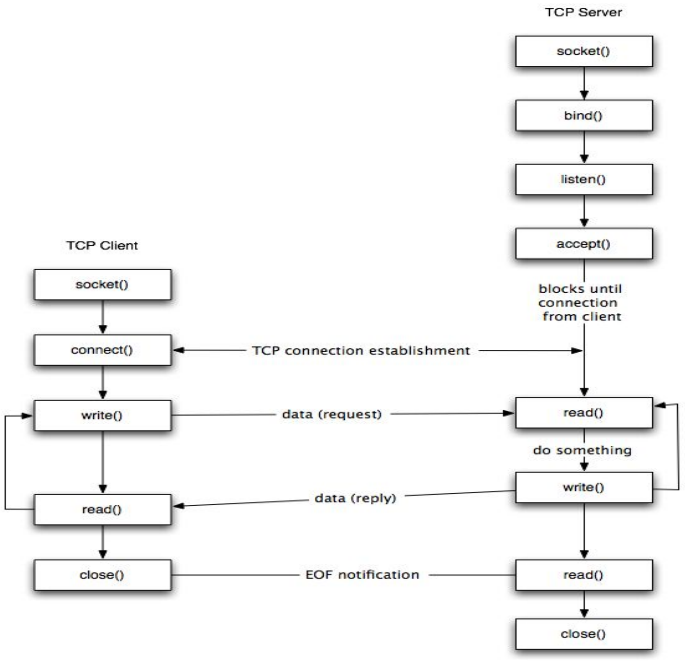
4) **¿A qué capa del modelo TPC/IP pertenecen los sockets? ¿Porque?**

Los sockets pertenecen a la capa de transporte en el modelo TCP/IP, debido a que son los encargados de “establecer las reglas de transporte de la información”, es decir, si serán bajo el protocolo TCP (orientado a la conexión, con un orden especifico de envío de datos, etc. En el cual se utilizan ‘Stream Sockets’) o si serán bajo el protocolo UDP (sin orientación a la conexión ni fiabilidad ‘Datagram Sockets’)

5) **¿Cómo funciona el modelo cliente-servidor con TCP/IP Sockets?**

En el modelo TCP/IP el cliente-servidor funciona de la siguiente manera:

El servidor debe crear un socket (con su dirección IP y el puerto donde va a estar escuchando), se debe configurar y luego quedar en la espera de que un cliente se conecte. Por parte del cliente, debe crear un socket, y posteriormente indicar a qué dirección ip y puerto se desea conectar.
Una vez ya hecho el pedido de coneccion por parte del cliente, y aceptada de parte del servidor, el cliente es quien debe escribir, o hacer la petición al servidor, una vez hecha, el servidor le contestara con un ack, o devolviéndole la información solicitada, y así hasta que el cliente desee cerrar la conexión, cuando esto suceda, el servidor enviará un último mensaje indicando que acepta la solicitud de cerrar la conexión, y se desconectan ambas partes.



6) **¿Cuales son las causas comunes por la que la conexión entre cliente/servidor falle?**

Las fallas más comunes de que una conexión falle son:

- Dirección IP o Puerto inválidos.
- Que la máquina donde se aloja el servidor tenga dicho puerto bloqueado, por ende ningún cliente podrá ingresar a traves de ese puerto.
- Que por alguna razón el servidor se apague, y al restablecerse tenga una ip diferente, y ya no sea existente cuando el cliente quiera re-conectarse.

7) **Diferencias entre sockets UDP y TCP**

Los sockets TCP tienen las siguientes propiedades:

- Son orientados a la conexión.
- Se garantiza la transmisión de todos los octetos sin errores ni omisiones.
- Se garantiza que todo octeto llegará a su destino en el mismo orden en que se ha transmitido.

Estas propiedades son muy importantes para garantizar la corrección de los programas que tratan la información.

Los sockets UDP tienen las siguientes propiedades:

- No son orientados a la conexión.
- No poseen una garantía de entrega (Los mensajes pueden o no llegar en el mismo orden en el que son enviados).

Estos sockets son apropiados cuando se deben entregar grandes flujos de datos con poca latencia y donde los datos no son tan importantes, como por ejemplo, un streaming de audio.

8) **Diferencia entre sync & async sockets?**

Los sockets pueden ser sincrónicos o asincrónicos tanto del lado del cliente, como del servidor.

- Del lado del cliente:
    - Sincrónicos: Un socket de cliente síncrono suspende el programa de aplicación mientras se completa la operación de la red.
    - Asíncrono: Un socket de cliente asíncrono no suspende la aplicación mientras espera a que se completen las operaciones de red. En su lugar, procesa la conexión de red en un hilo mientras la aplicación continúa ejecutándose en el hilo original.

      Es preferible utilizar los sockets asíncronos cuando se hace un uso intensivo de la red.
- Del lado del servidor:
    - Sincrónico: Los sockets de servidor síncronos suspenden la ejecución de la aplicación hasta que se recibe una solicitud de conexión en el socket.
    - Asíncrono: Los sockets asíncronos utilizan subprocesos del sistema para procesar las conexiones entrantes. Un hilo es responsable de aceptar conexiones, otro hilo se utiliza para manejar cada conexión entrante y otro hilo es responsable de recibir datos de la conexión.