TP FINAL REDES

Integrantes:

Schaab, Juan Manuel.

Viñas, Federico.

Preguntas

1. **¿Qué es un puerto?**

Un puerto es conector o interfaz que sirve para comunicar elementos de hardware o software y permite enviar y recibir datos de un dispositivo a otro.

Puertos lógicos: interfaz que permite establecer comunicaciones entre procesos.

Interfaz: mecanismo o herramienta que permite establecer una comunicación entre programas.

1. **¿Cómo están formados los endpoints?**

Están formados por diversos elementos como sistemas operativos, licencias antivirus o cliente VPN, todos ellos actualizados

**3. ¿Qué es un socket?**

Concepto abstracto por el cual dos procesos pueden intercambiar datos de forma confiable, son orientados a la conexión. Es un mecanismo para la entrega de paquetes de datos provenientes de la tarjeta de red a los procesos o hilos apropiados.

Se define por dos direcciones IP (local y remota), un protocolo de transporte (TCP o UDP) y dos números de puerto (local y remoto) .

De esta forma es posible implementar una arquitectura cliente-servidor, la información será transmitida por las diferentes capas de la red.

Está formado por algunos elementos representados como objetos

1. **¿A qué capa del modelo TPC/IP pertenecen los sockets? ¿Porque?**

Pertenecen a la capa de Transporte, ya que usa protocolos de esta capa para su funcionamiento. Por ejemplo, las propiedades de un socket dependen de las características del protocolo en el que se implementan. El protocolo más utilizado es la Transmisión Control Protocol. Cuando se implementan con el protocolo TCP, los sockets 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.

1. **¿Cómo funciona el modelo cliente-servidor con TCP/IP Sockets?**

Un servidor es una aplicación que ofrece un servicio a usuarios de Internet, un cliente es el que pide ese servicio. Una aplicación consta de una parte de servidor y una de cliente, que se pueden ejecutar en el mismo o en diferentes sistemas.

Los usuarios invocan la parte cliente de la aplicación, que construye una solicitud para ese servicio y se la envía al servidor de la aplicación que usa TCP/IP como transporte.

El servidor es un programa que recibe una solicitud, realiza el servicio requerido y devuelve los resultados en forma de una respuesta. Generalmente un servidor puede tratar múltiples peticiones (múltiples clientes) al mismo tiempo.

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

Las causas comunes por la que la conexión entre cliente/servidor falle son:

Error 2200: El cliente no recibió una respuesta del servidor en un tiempo determinado (timeout), esto sucede solo si un límite de tiempo fue especificado. Error 2300: El servidor cerró la conexión.

Error 2310: El servidor se ha caído mientras intentaba procesar el Handshake Request (conexión con el cliente). La conexión se cerró.

Error 2315: El servidor recibió el Handshake Request (conexión con el cliente) y devolvió una respuesta del tipo non-IIOP que el cliente no puede procesar (El cliente recibe una respuesta en un lenguaje que no entiende).

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

La diferencia principal es que TCP es orientado a la conexión y UDP no.

TCP es un protocolo confiable, UDP es poco confiable.

TCP es más lento ya que establece la conexión antes de transmitir los datos.

Ambos pueden detectar errores pero solo TCP puede corregirlos.

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

**Sync Sockets**: Los sockets asincrónicos esperan por una respuesta, si se usa por ejemplo un read, el programa quedara esperando hasta tener data disponible. La comunicación es parecida a una llamada telefónica, uno habla y luego espera una respuesta del otro lado.

Los sockets asincrónicos realizan operaciones secuenciales:

* Abrir un puerto de comunicación.
* Intercambiar información
* Cerrar el puerto de comunicación

Cada conexión usa un hilo que abre un socket escribe el pedido y muestra el resultado, hay N hilos si hay N conexiones.

**Async Sockets**: De manera asincrónica no se espera una respuesta para continuar, los programas envían información y chequean la respuesta más tarde. Es similar a la comunicación por mensajes de texto. Uno envía un mensaje pero no se queda esperando la respuesta inmediata, ya que esta puede llegar en unas cuantas horas.

La comunicación no es secuencial para nada.

El programa principal abre N sockets y los maneja en un loop principal.