Practica Nro. 5
Protocolos de transporte: UDP – TCP. Sockets: Concepto y Programación de Sockets.

Cuestionario

Protocolos de transporte

a) Explique las principales diferencias y similitudes entre los protocolos UDP y TCP.
b) Explique qué es Control de Flujo y qué es Congestión. ¿Quién es el responsable de evitarlo en cada caso, el lado transmisor o el lado receptor?
c) Explique como se determinan los Timeouts en el protocolo TCP.
d) Describa los procedimientos utilizados por TCP para realizar control de flujo.
e) Describa los procedimientos utilizados por TCP para evitar la congestión.
f) Describa el procedimiento que se utiliza para realizar una conexión en TCP (3-way handshake). Explique que sucede si hay una falla en alguno de los tres pasos.
g) Explique que diferencia hay entre una apertura de socket TCP pasiva y activa.

Utilitarios de red

a) Explique el funcionamiento del comando *`netstat`* en Linux.
b) Explique el funcionamiento del comando *`nc`* en Linux.
c) Explique el funcionamiento del programa *`telnet`* en Linux. Busque y active la versión de *`telnet`* para Windows.
d) Intente conectarse al port 80 de www.google.com ¿qué observa?

Sockets 1

a) Estudie cuales son las diferentes funciones destinadas a la creación y cierre de sockets. Indique cual es la diferencia entre familia, tipo y protocolo como argumentos de la función `socket()`
b) Estudie la función de la estructura struct sockaddr_in. Indique cuales son sus campos y para que se utilizan.
c) Investigue cuál es la diferencia entre un socket con tipo SOCK_STREAM y SOCK_DGRAM.
d) Describa las funciones `bind()`, `connect()`, `listen()` y `accept()` explicando para que se usa cada una de ellas. Indique cuales se utilizan en TCP, cuales en UDP y cuáles en ambos protocolos.
e) Investigue y describa las funciones `send()`, `sendto()`, `recv()` y `recvfrom()`. Indique cuáles se utilizan para UDP y cuáles se utilizan para TCP.

Sockets 2

a) Analice, compile y ejecute los programas ejemplos ***clienteUDP.c*** y ***serverUDP.c*** presentados en clase.
b) En el caso del programa servidor analice como se inicializa la configuración del mismo: ¿cuál es su IP y el puerto a utilizar? ¿porqué es necesario el uso de la función bind y que significado tiene INADDR_ANY? ¿Es necesario inicializar la estructura client_addr? ¿para qué la utiliza el programa servidor?
c) En el caso del programa cliente analice ¿cual es la información que requiere para comunicarse con el servidor? ¿importa en este caso la IP y el Puerto del cliente? ¿Sería posible ejecutar simultáneamente múltiples instancias del programa cliente?¿Existe una conexión entre el cliente y el servidor?.
d) ¿Qué sucede cuando se ejecuta el programa servidor sin ejecutar el cliente? ¿Qué sucede si el programa cliente se ejecuta primero que el servidor?
e) Intente utilizar el programa cliente con dirección de destino 1.2.3.4 y siga las instrucciones en pantalla, espere al menos 5 minutos y luego comente acerca de lo que observa en la consola.

Sockets 3

a) Analice, compile y ejecute los programas ejemplos ***clienteTCP.c*** y ***serverTCP.c*** presentados en clase.
b) En el caso del programa servidor analice como se inicializa la configuración del mismo: ¿cuál es su IP y el puerto a utilizar? ¿porqué es necesario el uso de la función bind y que significado tiene INADDR_ANY? ¿Es necesario inicializar la estructura client_addr? ¿para qué la utiliza el programa servidor?
c) En el caso del programa cliente analice ¿cual es la información que requiere para comunicarse con el servidor? ¿importa en este caso la IP y el Puerto del cliente? ¿Sería posible ejecutar simultáneamente múltiples instancias del programa cliente?¿Existe una conexión entre el cliente y el servidor?
d) ¿Qué sucede cuando se ejecuta el programa servidor sin ejecutar el cliente? ¿Qué sucede si el programa cliente se ejecuta primero que el servidor?

e) Intente utilizar el programa cliente con dirección de destino 1.2.3.4 y siga las instrucciones en pantalla, espere al menos 5 minutos y luego comente acerca de lo que observa en la consola.

Servidor concurrente

Analice el funcionamiento def programa ***servidorTCPconcurrente.c*** que se encuentra en la pagina de ejemplos de moodle. ¿Cuándo termina cada hijo? ¿Cuándo termina el padre?

TRABAJO PARA ENTREGAR

Queremos un programa ***servidor concurrente*** que permita a un cliente que se conecta utilizando el protocolo TCP enviar un archivo para que el servidor lo almacene.

- Cuando un cliente se conecta, el servidor enviará el mensaje: “listo” y esperará recibir la palabra “archivo”
- A continuación el servidor espera un espacio y luego el nombre con que se quiere guardar el archivo (sólo letras, numeros y el carácter ‘.’) finalizado con un espacio.
- Luego se quedará esperando un número codificado en ascii que indica el tamaño en bytes del archivo, también finalizado con un espacio.
- A continuación comenzará la recepción de los datos, que serán almacenados en un archivo cuyo nombre es el recibido en primer término, hasta completar la cantidad de bytes correspondiente.
- El servidor debe permitir la recepción de archivos binarios.
- Una vez recibida la totalidad de los datos, el servidor contestará con el siguiente mensaje:
  - “Archivo xx completo, tamaño declarado yy bytes, tamaño real zz bytes.”
    - xx: nombre del archivo
    - yy: tamaño enviado en el mensaje del cliente
    - zz: total de bytes recibidos por el servidor.
- Luego el servidor cerrará la conexión con el cliente.
- El servidor debe llevar un archivo de registro de todas las conexiones entrantes, con fecha y hora de inicio y de finalización, tamaño del archivo recibido, cantidad de bytes enviados y cantidad de bytes recibidos en formato csv.
- Defina explicitamente cómo será el manejo de errores.
- Para probar el programa puede utilizar el cliente TCP que se vio como ejemplo, o el programa *nc*.
- Recuerde verificar que no queden procesos zombies cuando finalizan los hijos generados.

Recuerde los pasos necesarios para resolver el problema. Debe entregar todo este material:

1. Interpretación del problema propuesto: ¿qué se debe hacer? ¿qué no se debe hacer? ¿qué datos tengo? ¿qué datos o que información hay que asumir?
2. Resolución del problema: ¿cómo lo va a resolver?. Haga uno o más diagramas o explicaciones que muestren desde un nivel de abstracción mayor a uno menor, la estrategia que adopta para la resolución del problema. Incluya seudo-código y/o diagramas de flujo.
3. Código en C. Recuerde utilizar comentarios para documentar el mismo. El programa deberá coincidir con el seudo-código definido en el paso anterior.

Bibliografía:

- Douglas Comer: “Internetworking with TCP/IP, Vol. I. Principles, protocols and architecture” (disponible en biblioteca)
- Presentaciones y ejemplos en el moodle de la cátedra