Universidad de Talca
Facultad de Ingeniería
Departamento de Ciencias de la Computación

Gestion de Redes

Laboratorio 2

Sockets.

Profesor: José Letelier (`jletelier@utalca.cl`)
Alumno Ayudante: Erik Regla (`eregla09@alumnos.utalca.cl`)

26 de septiembre de 2017

1. Descripción

En este laboratorio se espera que el alumno se familiarize con la implementación y debugging de sockets unix para protocolos de transporte UDP y TCP. Para estos efectos, ha de construir un programa (en C) utilizando sockets UNIX el cual permita trasferir archivos de manera remota.

2. Implementación transferencia de archivos por medio de sockets TCP (1.7 pto.)

Implemente dos programas (`emisor_tcp` y `receptor_tcp`) los cuales enviarán y recibirán respectivamente los archivos por medio de una conexión establecida por medio de TCP.

El programa que debe desarrollar debe cumplir con los siguientes requisitos[1]:

1. El programa que envía el archivo debe ejecutarse de la siguiente manera:
   `./emisor_tcp <archivo_origen><host>:<puerto>`
2. El programa que recibe el archivo debe ejecutarse de la siguiente manera:
   `./emisor_tcp <host>:<puerto>`
3. El programa emisor debe poder cargar archivos en cualquier ruta del sistema, mientras que el receptor al momento de recibirlos deberá escribirlos en su mismo directorio respetando su extensión original.
4. El programa receptor debe poder mostrar el progreso de la transferencia de una manera *comprensible para un usuario sin experiencia.*

[1] Bonus 1: Replique los permisos del archivo original.

3. Implementación transferencia de archivos por medio de sockets UDP (1.7 pto.)

Implemente dos programas (`emisor_udp` y `receptor_tcp`) los cuales enviarán y recibirán respectivamente los archivos por medio de una conexión establecida por medio de UDP. A modo de sugerencia, se recomienda ensamblar el archivo en memoria y dejar el volcado a memoria secundaria como el último paso.

El programa que debe desarrollar debe cumplir con los siguientes requisitos[2]:

1. El programa que envía el archivo debe ejecutarse de la siguiente manera:
   `./emisor_udp <archivo_origen><host>:<puerto>`
2. El programa que recibe el archivo debe ejecutarse de la siguiente manera:
   `./emisor_udp <puerto>`
3. El programa emisor debe poder cargar archivos en cualquier ruta del sistema, mientras que el receptor al momento de recibirlos deberá escribirlos en su mismo directorio respetando su extensión original.
4. El programa receptor debe de ser capaz de *recuperar"* los paquetes perdidos, vale decir, si alguno de estos no llega a destino, debe volver a ser solicitado.
5. El programa receptor debe poder mostrar el progreso de la transferencia de una manera *comprensible para un usuario sin experiencia* (esto incluye la cantidad de paquetes perdidos).

4. Análisis (1.6 pto.)

Para ambas implementaciones del programa anteriormente desarrollado, ejecute las siguientes pruebas:

1. Transfiera archivos de 100B, 512B, 1KB, 500KB, 2MB e identifique los paquetes involucrados en la llamada `send()/sendto()` y `recv()/recvfrom()` y contrástelos.
2. Wireshark para identificar si se produce o no un handshake durante la comunicación para cada una de las dos implementaciones y documente apropiadamente (esto quiere decir, la captura involucrada además de evidencia visual que respalde las afirmaciones. Utilize la interfaz "packet details" para facilitar el proceso).

5. Debugging (1.0 pto.)

Esta[3] es una copia (defectuosa) de un programa desarrollado por el ayudante para el curso "Sistemas Distribuidos" de este semestre. Sin embargo tiene un problema, la transferencia de datos

[2]Bonus 2: Soporte para transferencia de archivos de más de 500MB.

[3]`https://gist.github.com/KukyNekoi/a7e1e56612c403505ddf7e675116dc59`

no se efectua de manera correcta dado que si bien los sockets conectan y la conexión se establece, al momento de recibir los datos solo se obtiene un mensaje vacío.

El objetivo del programa es realizar una cola de tareas. El proceso cliente envía una tarea a una cola de tareas y un worker contínuamente realiza polling para saber si tiene tareas que resolver. En caso de tener tareas, este la resuelve y le envía la respuesta por medio de un mailbox el cual el cliente está activamente esperando. La comunicación entre la cola de tareas y el cliente funciona normalmente, sin embargo al momento de realizar consultas sobre esta solo se obtienen valores inválidos. [4]

1. Diagnostique[5] el defecto. Puede utilizar las herramientas que estime conveniente.
2. Parche el programa.

[4] Puede realizar todas las consultas pertinentes al funcionamiento de este código fuente al ayudante o bien solicitar una demostración de una ejecución correcta.

[5] Por diagnosticar se subentiende analizar el programa sin necesariamente estudiar el código.