Controlador de versiones

Trabajo práctico N.º 3

1. Introducción

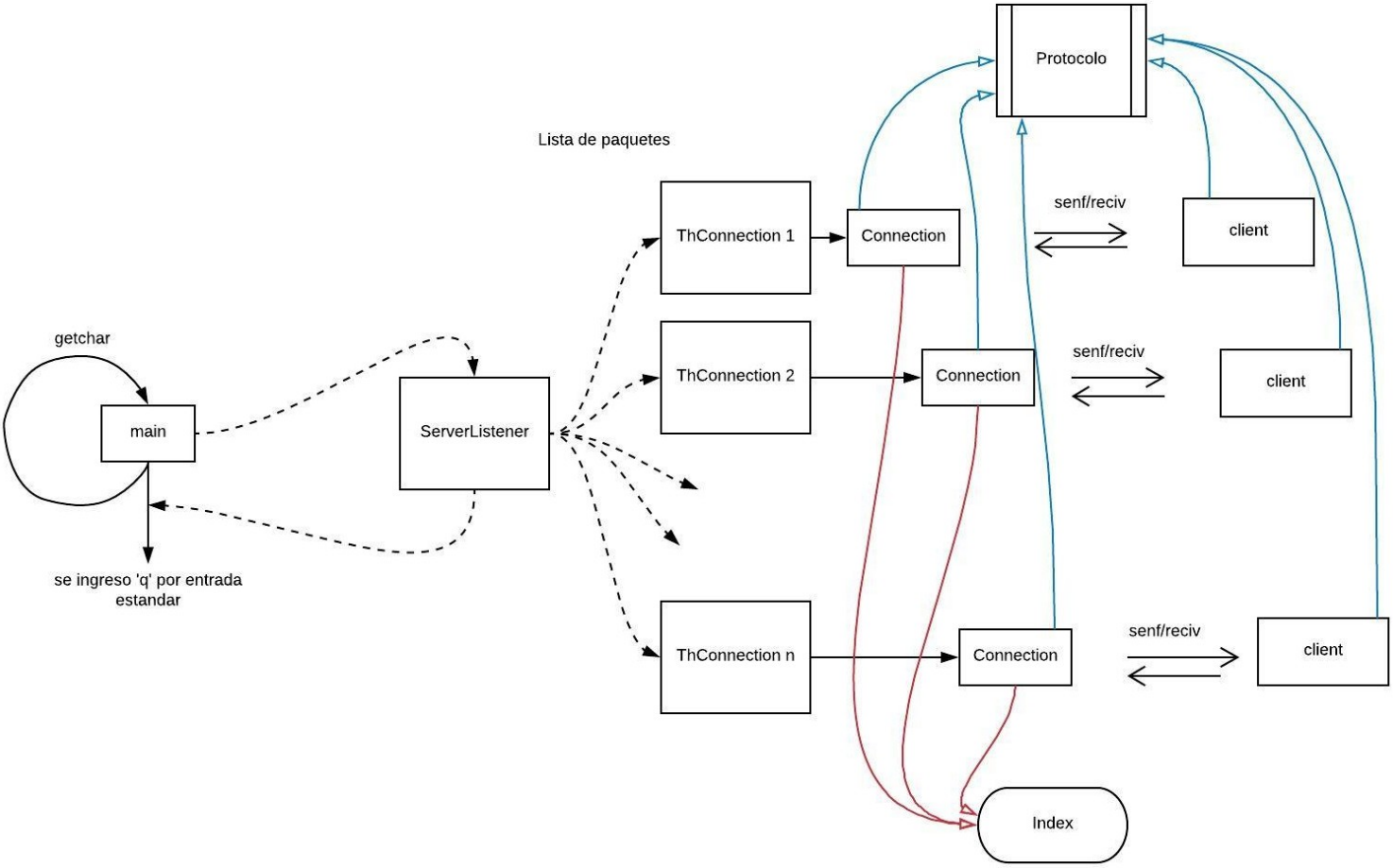
El presente trabajo consiste en un pequeño controlador de versiones. Los clientes pueden conectarse al servidor para subir una nueva versión de un archivo (push), agrupar distintas versiones de archivos (tag) o descargarse todos las versiones de archivos agrupados (pull).

2. Implementación

El programa tiene dos puntos de entrada, uno como servidor y otro como cliente.
La parte del cliente es simple porque unicamente se conecta con el servidor y ejecuta alguno de los comandos y luego termina.
El servidor, en cambio, hace dos cosas a la vez, por un lado espera un carácter para finalizar el programa en caso de que se ingrese una 'q'. Por el otro lado inicia un server_listener en un nuevo hilo (no seria posible en el mismo hilo). EL server_listener es un functor que se inicializa con un socket y cuya funcion principal es quedarse esperando distintos clientes en un loop infinito (hasta que se corte desde el hilo main).
El server_listener también tiene un vector de thConnections, por cada nueva conección con un cliente lanzará un nuevo hilo (thConnection). El hilo contiene dentro un objeto Connection el cual contiene un peer socket. El objeto connection se encargará de la ejecución del comando que le demande el cliente, y cuando termine se marcara como "dead". Asi es como logra tener varias conexiones en simultaneo.
No hay ninguna rest condition en la comunicación de los sockets porque cada uno tiene un a instancia de peer socket diferente.
El problema es cuando se quiere acceder a la información del archivo indice del servidor. El archivo indice es manejado por un objeto llamado serverIndex, el cual, al comienzo del programa lee el archivo y crea dos mapas, uno de archivos otro de tags. Se realizarán los cambios sobre estos mapas y solo al finalizar el server se escribirá nuevamente el archivo index actualizado.
Cada hilo tiene thConnection tiene una referencia a la misma instancia de serverIndex por lo tanto fue necesario lockear todas las funciones que manipulan los mapas.

En cuanto a la comunicación, como el socket del cliente y el peer socket deben comunicarse de acuerdo con el mismo protocolo, se creo una case abstracta protocolo de la cual hereda la clase Cliente y la clase Connection.
El protocolo es el owner del socket pero este esta marcado como protected al igual que todos sus métodos, para que pueda ser manipulado por las clases que heredan de él.

Cuando se ingresa una 'q' por entrada estándar el hilo main le avisara al serverListener. Este primero marcará la variable de condición del ciclo que acepta cliente como false para que no se sigan aceptando nuevos clientes. Luego recorrerá los thConnection del vector de hilos, desconectará los peersockets y hará join de los hilos. Por ultimo se desconectará el socket del servidor.



3. Problemas que surgieron con la implementación

Al desconectar un socket que estaba esperando un cliente surge un error. Para solucionarlo se cambio el tipo de socket a no bloqueante, es decir que no se queda bloqueado esperando a que se conecte un cliente, trata de conectarse y si no existe un cliente rompe.

Por otro lado, las pruebas pasan correctamente de forma local y udando valgrind Pero al querer redireccionar la entrada estándar del servidor surge un core dump. No pude identificar correctamente la causa del core dump. Al ejecutar con Valgrind se identifica que el server termina con un error causado al llamar un destructor de threads.

4. Soluciones para la reentrega

Se quito el flag de socket no bloqueante y para no mostrar el error por pantalla del shotdown del socket aceptador se lo envolvió la función accept en un try catch.

Se logro que pasaran las pruebas en el sercom arreglando la forma en la que se leía de la entrada estándar. El problema estaba en que no se tenia en cuenta la posibilidad de que ingresaran otro carácter que no fuera una q. Solo se tenia un *if(getchar() == 'q')*, por eso al leer desde un archivo un EOF, la condición pasaba a ser falsa, se salteaba el shutdown de los socket y el join de los hilos y no se quedaba bloqueado en el getchar. Se cambio por *while(getchar() != 'q')* y se solucionó.