Laboratorio 1
Sistemas Distribuidos 2020 – María Marta Barja

2.

a. Servidor con estados: son aquellos servidores que guardan información de las sesiones de los clientes de manera que la información que debe transmitir el cliente es mucho menor.

Servidor sin estados: son los servidores que no retienen información de la sesión. El cliente envía los datos de sesión relevantes al receptor de tal manera que cada paquete de información transferido puede entenderse de forma aislada, sin información de contexto de paquetes anteriores en la sesión.

b. Es un servidor con estados. Acumula valores del cliente en la variable acumulador presente en el servidor.

c. En este punto, se creó una variable acumuladora en el cliente, se descarta la misma del lado del servidor y utilizando la misma estructura ya definida, en vez de enviar el id del cliente en el primer campo, se envía el número acumulado guardado en el cliente. De esta manera el servidor solo realiza la suma del acumulado y el valor ingresado y devuelve el nuevo total. El método obtener que indica el valor total acumulado es un método local del cliente.

3.

c. La implementación con gRPC es mucho mas sencilla ya que no debemos ocuparnos de manejar conexiones, ni de serializar/des serializar las estructuras de los mensajes entre cliente y servidor. En la implementación con sockets se debe especificar dónde se aceptan las conexiones a los clientes, y utilizar el canal para enviar los mensajes que además se deben serializar si se trata de una estructura que contiene parámetros, en mi caso utilice la librería pickles de Python para serializar todos los mensajes, aun en los casos del open file y close file, donde las respuestas del servidor son strings. En gRPC las estructuras de los mensajes; y los parámetros y resultados de los métodos que se ejecutan remotamente se definen en el archivo de protocolo, y luego a partir del protocolo definido gRPC genera los stubs del cliente y del servidor, solo hay que implementar los métodos especificados del lado del servidor. En la implementación con sockets todo esto debe realizarse en forma manual. En conclusión es mucho mas sencillo (o menos laborioso) implementar servicios remotos utilizando gRPC.

Para el punto a inicialmente realice una clase servidor y una clase cliente. La clase cliente instanciaba un FSStub, generado por gRPC y se llamaba directamente a los métodos. En la clase Servidor implementé los métodos definidos en el stub generado por gRPC FSServicer. Esta primera implementación funcionaba correctamente. Luego de implementar la solución con sockets utilizando la estructura de base brindada por Pedro, reestructure las clases del punto a, de manera que para ejecutar el punto a y el punto b solo basta con importar los stubs cliente y servidor de los distintos paquetes (p3a y p3b).

4.

a. Se observa que el servidor atrapa una excepción al detectar que la conexión fue terminada por el host remoto pero al lanzar el cliente acepta la conexión y funciona nuevamente.

b. Al detener el servidor se observa el mismo error mencionado anteriormente pero del lado del cliente, detecta que la conexión al servidor ha sido terminada por el host remoto.

c. Es un servidor sin estados. Si bien el servidor almancena en un objeto file_manager, todos los descriptores de los archivos abiertos y el ultimo path del cual listó archivos, el cliente para realizar el read file, debe enviar el nombre del archivo, el offset y la cantidad de bytes, actualizando los últimos 2 valores por cada solicitud que haga al servidor, hasta terminar de transferir el archivo.

6.

a. Se agregó un print(threading.get_ident()) dentro del método run del hilo del servidor y se observó que las solicitudes de distintos clientes son atendidas por distintos hilos. Es decir, al aceptar la conexión del cliente en el servidor, se crea un hilo para atender las solicitudes que llegarán a través de esa conexión, de esta manera cada cliente es atendido por un hilo distinto.

b. A partir de los descripto anteriormente, esta implementación tiene un thread por conexión. Se crea un hilo por cada cliente, el mismo hilo atiende todas las solicitudes y muere al desconectarse el cliente.

c.
Se transfirió un archivo de 878347 KB – con un buffer de 4000 bytes
Con sockets
tiempo de transferencia: 57368 ms

con grpc
tiempo de transferencia: 156755 ms

7. Para este punto implemente un ThreadPoolExecutor en el Stub del cliente, con un limite de 5 workers pero aun esta incompleta la implementación. Igualmente, a mi parecer, es útil esta implementación si se desea limitar el número de clientes conectados en simultáneo.