Distributed Systems Programming

A.Y. 2022/23

Exam Assignment for Exam Call on 17/05/2023

Deadline for submission: 15/05/2023, 12:00 CET

Reusing your solution of Laboratory 4, develop a new java client and a new java server that have the purpose of transferring a file from client to server, using a protocol similar to the one used in Laboratory 4 for file conversion, but with the addition of a recovery mechanism which facilitates the transfer of files over a network characterized by low bandwidth and high failure rate, with frequent network outages. This mechanism enables the recovery of a previously interrupted file transfer. More precisely, the protocol implemented by the new client and server must adhere to the following specifications:

1. After a connection has been established, if the client wants to send a new file from scratch, it sends to the server the id of the file to be transferred, represented as a string of ASCII characters terminated by a zero byte (null ASCII character). The maximum length of the id must be 10 characters.
2. The server, upon correctly receiving the file id, creates the name of a local file where the received file bytes will be stored, by appending a time stamp to the received id, represented as string of ASCII characters. Then the server responds to the client by sending one byte with value 0 if the file name was received correctly or value 1 in case of error. If no error has occurred, the server continues by sending the previously computed local filename to the client as a null terminated string of ASCII characters and opens the file for writing in a local folder, otherwise it starts the procedure for closing the connection gracefully.
3. When the client receives the server response, if the response is positive, it sends the file length and contents to the server. More precisely, it sends (in the specified order):
   - a 4-byte 2's complement integer number in network byte order, representing the length in bytes of the file that must be transferred.
   - the bytes of the file that must be transferred.

   If, instead, the client receives a negative response from the server, it completes the graceful closing of the connection.
4. The server stores the received file bytes into the local file in blocks, as far as they are received. Finally, when the file has been stored completely, the server sends a zero byte to the client signaling that the entire file has been stored

correctly, and it starts the procedure for gracefully closing the connection. In case errors occur during the storage of the file, the server sends a byte with value 1 and it starts the procedure for gracefully closing the connection.

5. If the TCP connection drops during the transfer of a file, the client can recover the transfer by reconnecting to the server. In this case, the client has to send the id of the file, including the appended time stamp, as it was received from the server. If a file named with the specified id exists in the server local folder, the server will respond by sending one byte with value 2, followed by the length of the local file, represented as a 4-byte 2's complement integer number in network byte order. Upon receiving this information, the client will send the missing part of the file to be transferred, by sending:
   - a 4-byte 2's complement integer number in network byte order, representing the length in bytes of the missing part of the file that must be transferred.
   - the bytes of the missing part of the file that must be transferred.

   In this case the server will open the local file in append mode and continue storing the file contents as in the normal case.

The server application must listen on port 2001, as in Laboratory 4. The client application must receive as first argument on the command line the path in the local file system of the file to be transferred. As optional second argument, the client must receive the timestamped id of a file whose transfer has to be restored. The client must write to the standard output the timestamped id of the file, as received from the server or from the command line, so that it can be used later to recover the transfer in case it is interrupted. Afterwards, when the client is ready to start transferring the file, it must output the string "The client is starting the file transfer", while when the transfer has been completed, it must output the string "The client has finished transferring the file" and it must terminate. In case the client detects that the TCP connection has been dropped, the client must output the string "TCP connection interrupted" and it must terminate.

Test your new Java client and server together, including robustness tests with a dropped TCP connection, a crash of the server, and a crash of the client. To cause a TCP dropped connection, you can use the tcpkill Linux command.

Submit the updated solution, including all the following items:

- The code of the new client and server
- README.md files that specify the contents of folders and instructions on how to compile and run the code from scratch

Important:

- The solution must work within the Labinf Linux machines, with the software already installed in those machines.
- The solution must be uploaded to a git repository for which you will get the credentials.