UNIVERSIDADE FEDERAL DE MINAS GERAIS
Instituto de Ciências Exatas
Departamento de Ciência da Computação
DCC023 -- Redes de Computadores
Trabalho Prático 2 - Transmissão de arquivos usando UDP
com controle sobre TCP
Guilherme de Abreu Lima Buitrago Miranda - 2018054788

1 - Introdução

O trabalho prático 2 da disciplina de Redes de Computadores tem como objetivo a criação de um servidor e clientes para transmissão de arquivos usando UDP, com o controle da mesma sendo feito via TCP, bastante semelhante a um sistema de armazenamento na nuvem, como o Dropbox ou o Google Drive.

Nas seções a seguir, encontram-se detalhes relativos à implementação do trabalho, desafios enfrentados, conclusões e referências bibliográficas.

2 - Implementação

A implementação do trabalho encontra-se dividida em três arquivos principais: Servidor.java, Cliente.java e MessagesUtils.java. Contudo, o método main é implementado apenas no servidor e no cliente, ou seja: o arquivo MessagesUtils não é executável. Além disso, é fundamental que o servidor seja executado antes do cliente, para que a conexão possa ser corretamente estabelecida.

Ainda em termos cronológicos, assim que o cliente se conecta ao servidor, o último cria uma thread invocando a classe ClientHandler, implementada no próprio arquivo. A existência da mesma é justificada pela conexão de diversos clientes em um mesmo servidor simultaneamente. Assim, enquanto a thread é criada no servidor, o cliente já trata de enviar a primeira mensagem, cujo código é HELLO (1). Tal mensagem tem como objetivo identificar o cliente para o servidor.

Assim que a mensagem HELLO (1) é recebida no servidor, este trata de responder ao cliente com a mensagem CONNECTION (2), que tem como objetivo enunciar qual o porto UDP deve ser conectado para o envio do arquivo. Ato contínuo, o cliente recebe a mensagem e responde com outra do tipo INFO FILE (3). Nela, há, além do próprio tipo da mensagem, informações a respeito do nome do arquivo e do tamanho do mesmo em bytes. Por fim, tendo o servidor recebendo a mensagem anterior, responde ao cliente com OK (4), sinalizando que a transferência do arquivo pode ser iniciada.

Todas as mensagens acima mencionadas são enviadas por meio do protocolo TCP e, portanto, não é necessário que o programa trate casos de retransmissão. Além disso, toda a conversão de shorts, inteiros e longs para bytes é feita no arquivo MessagesUtils.java, de forma que os métodos possam ser aproveitados, tanto do lado do cliente, quanto do lado do servidor. Assim,

estando as mensagens já convertidas, as classes DataInputStream e DataOutputStream tratam de recebê-las e enviá-las via rede, utilizando os métodos read e write respectivamente.

Dessa forma, inicia-se efetivamente a transferência do arquivo. No cliente, há o envio dos pacotes via protocolo UDP (com identificador FILE (6)) e o controle da janela deslizante do método Go-Back-N via protocolo TCP (com identificador ACK (7)), com o objetivo de verificar a integridade do arquivo (em caso de perdas de pacotes). Após a definição do tamanho da janela e do valor de timeout adequados para os testes (em redes com até 25% de perdas), cria-se também uma vector que armazena as mensagens já enviadas. Assim, caso haja a necessidade de retransmissão, o programa não precisa se preocupar em recriar uma mensagem em particular; basta acessar o vector. O vector foi escolhido em detrimento do arraylist devido ao fato de ser thread safe, conforme pode ser observado em sua documentação.

Isso posto, o cliente tem, portanto, o papel de enviar os pacotes para o servidor, começando do primeiro pacote cuja confirmação de chegada foi recebida (no começo do programa, o primeiro pacote do arquivo) e indo até o fim da janela (ou até o último pacote do arquivo, no fim da execução). Já o servidor fica responsável por enviar ao cliente a confirmação dos pacotes recebidos. Caso o servidor receba um pacote, digamos, i + 1, sem que tenha recebido o pacote i, as informações são armazenadas num HashMap, com o objetivo de serem utilizadas futuramente. Ao receber o pacote i, o servidor escreve o conteúdo no FileOutputStream e, logo em seguida, escreve o conteúdo do pacote i + 1, assim como envia, na ordem adequada, as mensagens de recebimento ACK (7). Dessa forma, fica garantida a integridade do arquivo, o conhecimento dos pacotes que o servidor tem em posse pelo cliente, e a implementação da variação do método Go Back N em que as janelas de ambas as partes tem tamanho maior que 1.

Assim, após o servidor se certificar de que foram recebidos corretamente todos os pacotes que deveriam ser enviados pelo cliente, é enviada uma mensagem do tipo FIM (5), que sinaliza que o arquivo foi corretamente recebido e que a comunicação pode se encerrar. Consequentemente, todos os sockets de comunicação entre cliente e servidor são fechados e o cliente é finalizado com sucesso. Já o servidor continua executando, disposto a receber a conexão de novos clientes.

Os testes de envio do arquivo foram feitos em redes com até 25% de perdas de pacote e funcionaram como o esperado. Para maiores instruções sobre a execução, há disponível, no mesmo .zip que a presente documentação, um arquivo readme.txt. Em caso de maiores taxas de perdas, é provável que seja necessário acrescer o timeout dos pacotes.

3 - Desafios

Em comparação ao trabalho prático 1, os desafios mudaram de forma considerável. Deixei de encarar percalços inerentes à programação na linguagem C e pude focar exclusivamente na aplicação do conhecimento teórico na prática. Isso se deu, sobretudo, pela minha maior experiência com a linguagem Java e ao nível mais alto (em termos de abstração) das bibliotecas de sockets da linguagem.

Dessa forma, a principal dificuldade se deu durante a implementação da versão do Go Back N cujas janelas são maiores que 1, tanto no servidor, quanto no cliente. Após revisitar os conceitos

teóricos do protocolo e elucidar minhas dúvidas no fórum da disciplina no moodle, enfim fui capaz de corretamente transferir o arquivo do cliente para o servidor, mesmo com perdas de pacotes durante o processo.

4 - Conclusão

A implementação do trabalho foi bastante proveitosa para efetivamente fixar os conceitos expostos até o presente momento na disciplina. Dessa forma, o aprendizado se deu, sobretudo, durante o enfrentamento dos desafios supracitados, assim como nas outras dúvidas expostas pelos colegas no fórum de discussões.

Não obstante, o trabalho também me possibilitou aprender, na prática, ainda mais a respeito da programação de redes com linguagem Java, visto que, apesar de programar na linguagem há muito tempo, nunca havia tido a oportunidade de utilizar a java.net. Com isso, considero que o trabalho foi de grande proveito para o processo de aprendizagem.

5 - Referências Bibliográficas

- Introducing Threads in Socket Programming in Java - https://www.geeksforgeeks.org/introducing-threads-socket-programming-java/
- Working with UDP DatagramSockets in Java - https://www.geeksforgeeks.org/working-udp-datagramsockets-java/
- How to check if a String contains only ASCII? - https://stackoverflow.com/questions/3585053/how-to-check-if-a-string-contains-only-ascii