# Compressed Network Communication

**Cristiano L. S. Rosa[1], Ilem L. D. Santos[1]**

[1] Departamento de Ciência da Computação – Universidade Federal de Roraima (UFRR)
Avenida Capitão Ene Garcez, 2413, Aeroporto – Boa Vista – RR – Brazil

cristiano.16bits@gmail.com, ilemlima@gmail.com

***Abstract.*** *The purpose of this report is to document what was developed for the final design of operating systems, using the client-server model for sending compressed data in a communication using sockets in tcp mode.*

***Resumo.*** *O objetivo deste relatório é documentar o que foi desenvolvido para o projeto final de sistemas operacionais, utilizando o modelo cliente-servidor para envios de dados compactados numa comunicação, sendo usado sockets no modo tcp.*

## 1. Introdução

O projeto a ser implementado utiliza o modelo cliente-servidor, no qual o cliente solicita ou envia dados ao servidor. Frequentemente são criados agentes tanto do lado cliente quanto do servidor para controlar a comunicação em rede visando a proteção do aplicativo das complexidades dos protocolos de acesso. Os agentes intermediários podem implementar recursos para aumentar a segurança, como criptografar o tráfego, e reduzir o custo de dados para o tráfego assim melhorando o desempenho na rede.

## 2. Metodologia

A conexão é composta pelos processos cliente e servidor que são conectados pelo protocolo de comunicação TCP e o shell que é conectado ao servidor via pipes. A Figura 1 retrata a estrutura projetada. O cliente e o servidor são desenvolvidos usando a linguagem de programação C e para a compressão de dados será utilizado a biblioteca Zlib.

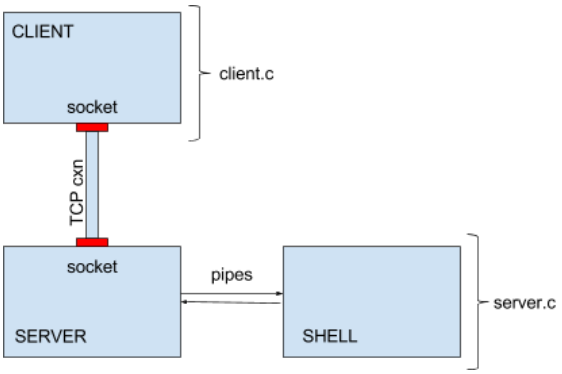


**Figure 1. Diagrama do Projeto**

## 2.1. Cliente

O programa cliente.c recebe algumas informações como parâmetro sendo a porta de conexão, nome do arquivo com sua extensão(.txt) e o endereço que por padrão será localhost para que seja iniciado uma conexão com o servidor. O cliente ecoa a entrada do teclado para o soquete, e a entrada do soquete para o monitor.

parâmetros :

-l, --log, gera um registro .txt dos dados enviados pelos soquetes;

-c, --compress, confirmação para realizar a compactação dos dados;

-p, --port, porta usada para abrir uma conexão com o servidor

-h, --hostname, nome usado pelo host que é sempre usado localhost

## 2.2. Servidor

O programa servidor se conectará ao cliente via conexão TCP que receberá os comandos do cliente pela entrada do soquete de rede (porta especificada com o parâmetro obrigatório --port=linha de comando). Assim que a conexão for feita, o servidor criará um fork do processo filho, que executa um shell para processar os comandos recebidos por meio de pipes. No servidor são criados dois pipes para comunicação entre processos, sendo um utilizado para a entrada e o outro para saída.

O processo do shell tem a sua entrada padrão (stdin) redirecionada para a saída do pipe de entrada, e a saída padrão (stdout) e erro padrão (stderr) são redirecionadas para a entrada do pipe de saída.

Assim que o shell executa o comando e o servidor recebe a saída pelo pipe, o servidor envia ao cliente através da conexão TCP.

## 2.2. Compressão

Para a comunicação comprimida dos dados foi utilizada a biblioteca zlib que é uma biblioteca para executar a compressão e descompressão gzip (com base em uma combinação de codificação LZW e Huffman). Zlib foi escrita por Jean-loup Gailly e Mark Adler tendo sua distribuição open-source (Código aberto).

## 3. Aplicação

O sistema operacional usado foi Ubuntu 20.04 com o compilador gcc (Ubuntu 9.3.0-17ubuntu1~20.04) 9.3.0. Depois de compilados utilize os comandos para execução, primeiro iniciar o servidor e depois o cliente.

./server --port=8080

./client --port=8080 --log=texto.txt

Execução do servidor na porta 8080.

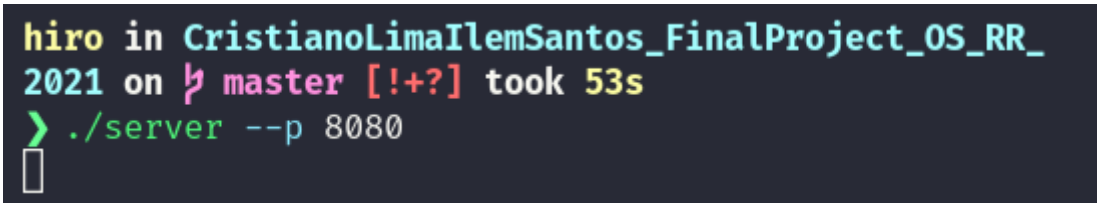


**Figure 2. Inicialização do servidor**

Execução do cliente passando a porta e o nome do arquivo do arquivo para armazenarmos o registro da comunicação. É solicitado ao servidor o comando ls para listar os arquivos no diretório.

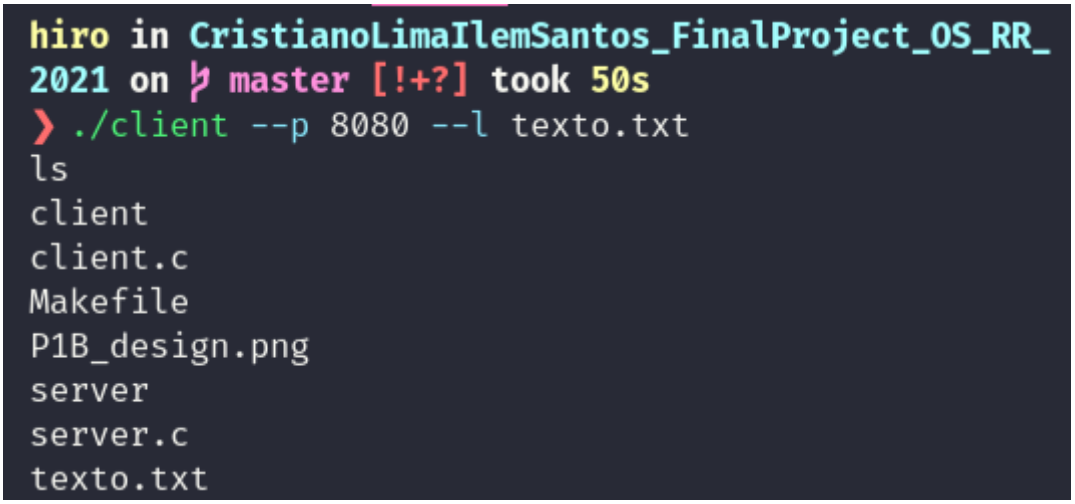


**Figure 3. Inicialização do cliente**

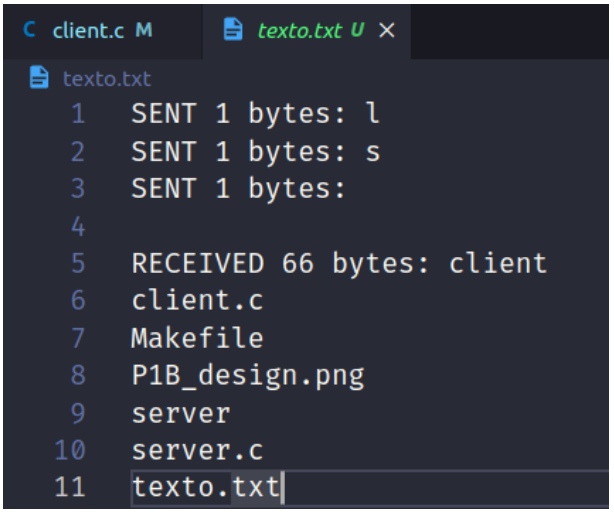


**Figure 4. Registro sem compactação**

Execução do cliente e servidor com a flag de --c

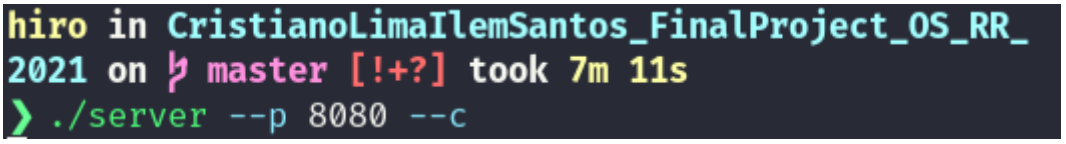


**Figure 5. Inicialização do servidor com método de compressão**

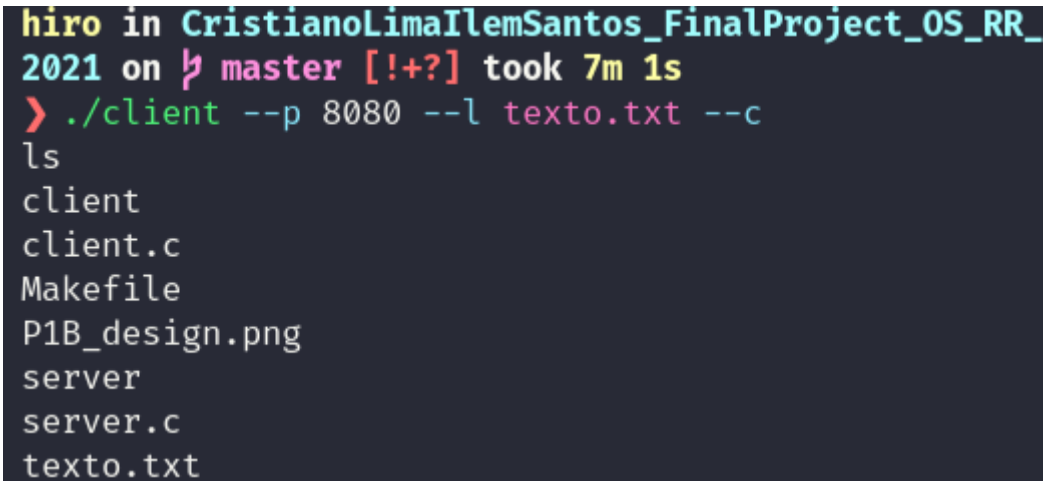


**Figure 6. Inicialização do cliente com método de compressão**

Os registros compactado serão armazenados no arquivo gerado texto.txt

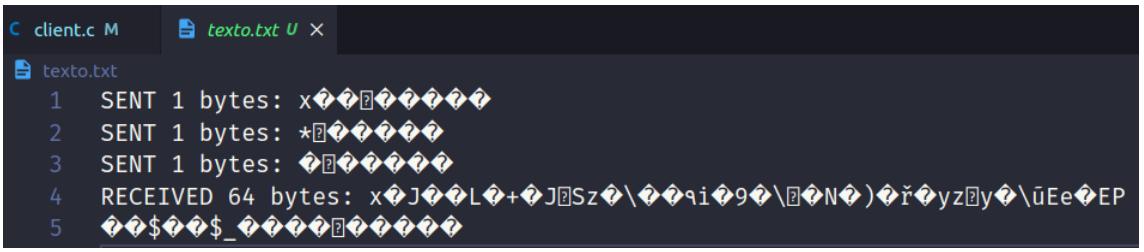


**Figure 7. Registro com compressão**

## 4. Conclusão

O projeto foi dividido em duas etapas, a implementação do cliente foi desenvolvida por Cristiano e o servidor pelo Ilem. Grande parte do projeto já tinha sido implementado por outros desenvolvedores, logo o desafio foi compreender o comportamento e os mecanismos utilizados para pegar os valores de entrada via TCP, para fazer a conexão e a compressão dos dados. Infelizmente não conseguimos fazer a descompactação de dados no lado do cliente que ultrapassar os 1024 bytes, mas todo conhecimento aplicado no projeto como redes, compressão de arquivos e chamadas de sistema nos dão ideia do funcionamento de aplicações de um sistema operacional.

## References


Robert Ingalls (2021) “Sockets Tutorial”, https://www.cs.rpi.edu/~moorthy/Courses/os98/Pgms/socket1.html, Acessado: 25-09-2021.

Shahriar Shovon (2019) “Pipe System Call in C”, https://linuxhint.com/pipe_system_call_c/, Acessado: 20-09-2021.

R. Shanker (2021) “Input-output system calls in C”, https://www.geeksforgeeks.org/input-output-system-calls-c-create-open-close-read-write/, Acessado: 21-09-2021.

Daemonio Labs (2011) “Usando As Funções getopt() e getopt_long() Em C”,

https://daemoniolabs.wordpress.com/2011/10/07/usando-com-as-funcoes-getopt-e-getopt_long-em-c/, Acessado: 20-09-2021.

Michelle Sus (2018) “Compressed Network and Communication”, https://github.com/michfit/Compressed-Network-and-Communication, Acessado: 20-09-2021.