Flower 1, 2, 3 Project: Compressed Network Communication
Projeto Final de Sistemas Operacionais

Débora Bianca Taveira de Moura
Ciência da Computação
Universidade Federal de Roraima
Boa Vista, Roraima - BR
Email: deborabiancatm@gmail.com

Ewelly Fabiane Cunha de Sousa
Ciência da Computação
Universidade Federal de Roraima
Boa Vista, Roraima - BR
Email: ewelly.fab@gmail.com

***Abstract*—Communication is the main purpose of technology development, and the necessity of reduce the quantity of data is an consequence of it. Currently we use differents applications to connect with anyone and anything, this consumes lot of space in databases or in cloud to save our data. Ocasionally the idea of compress data has created. On this project our purpose is create a compressed network communication betwen a server and a client using a TCP socket.**

1. Introduction

O método utilizado para realizar a comunicação em nosso projeto é modelo cliente e servidor, onde o cliente é um processo que se conecta ao servidor para fazer uma requisição ou passar informações. Usualmente são criados agentes tanto do lado do cliente quanto do servidor para manipular a comunicação em rede com o intuito de proteger o aplicativo das complexidades dos protocolos de acesso remoto. Os agentes intermediários podem implementar recursos para aumentar a segurança, como criptografar o tráfego, e de redução do custo da comunicação, ao compactar o tráfego para melhorar o desempenho, de forma totalmente transparente para o usuário e o aplicativo, em nosso projeto realizaremos a compressão.

2. Metodologia

A network será composta pelos processos cliente, servidor, que são conectados via conexão TCP, e o shell que é conectado ao servidor via pipes. A figura 1 representa o esquema definido anteriormente. O servidor e o cliente serão desenvolvidos em C, a compressão será realizada com o auxílio da biblioteca Zlib, e as outras funcionalidades utilizaram as próprias bibliotecas do C.

2.1. Servidor

O programa do servidor executa em um soquete de rede com a porta especificada com o parâmetro port, ao se conectar com o cliente, recebe os comandos dele, atende os comando e os envia para o shell. Ele sempre aceitará uma

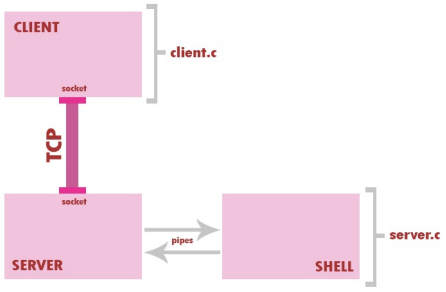

Figure 1. BigPicture

conexão e quando ela for feita é criado um fork do processo filho que executará um shell para processar os comandos recebidos. O stdin/stdout/stderr é redirecionado do processo shell para as extremidades apropriadas do pipe.

2.2. Cliente

No programa cliente será aberta uma conexão com um servidor. O cliente ecoa a entrada do teclado para o soquete, e a entrada do soquete para o monitor; Possui também uma opção –log=filename, que mantém um registro dos dados enviados pelo soquete, salvando o histórico em um .txt, além da opção –compress ao cliente, para realizar a compressão dos dados;

O programa cliente abrirá uma conexão com um servidor passando a porta como parâmetro em vez de enviá-lo diretamente para um shell. O cliente deve então enviar a entrada do teclado para o soquete (enquanto ecoa para o monitor), e a entrada do soquete para o monitor. Dependendo dos parâmetros passados na conexão do do cliente ao servidor ele poderá executar algumas funções, como por exemplo:

- -l, –log, que gera um registro .txt dos dados enviados pelo soquete;
- -c, –compress, para realizar a compactação dos dados;
- -p, –port abre uma conexão com o servidor na porta especificada;

- -h, –hostname, define o nome do host que sempre é usado o localhost;
- -a, –ajuda, para exibir todas a opções anteriores;

2.3. Compressão

Para a compressão dos dados foi utilizada a biblioteca zlib. O método de compactação da zlib é o Deflate, originado de uma combinação do algoritmo LZ77 e Huffman. Atualmente é utilizado em imagens em formato PNG e ZIP.

3. Aplicação

Figura 2. Demonstra a execução do servidor que passa como parâmetro a porta disponível para conexão e define que a compressão dos dados está ativa.

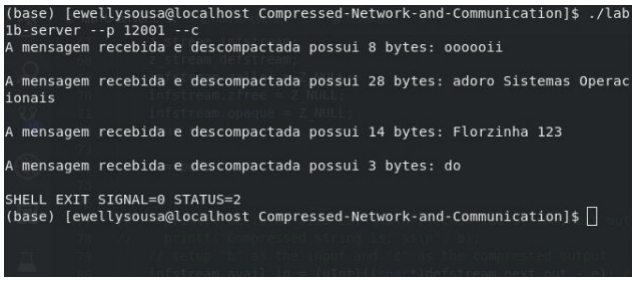

Figure 2. Comunicação pelo lado do servidor

Figura 3. Representa o lado do cliente, que ao executar a aplicação passa por parâmetro a porta, solicita o log, define o host e comprime as mensagens.

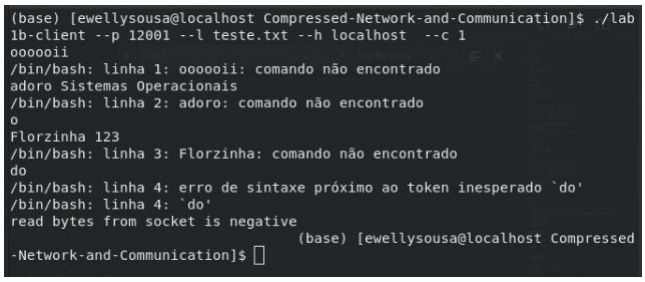

Figure 3. Comunicação pelo lado do cliente

Figura 4. Apresenta o uso do comando de ajuda solicitado pelo usuário, que imprime as possíveis funcionalidades da aplicação.

Figura 5. Exibe o arquivo .txt que representa o logfile. Infelizmente quando o cliente solicita o logfile os dados passados ao arquivo estão comprimidos, este foi um problema encontrado em nossa aplicação que não conseguimos solucionar.

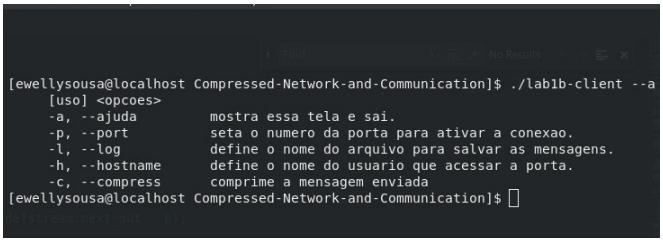

Figure 4. Acesso ao comando do cliente –ajuda

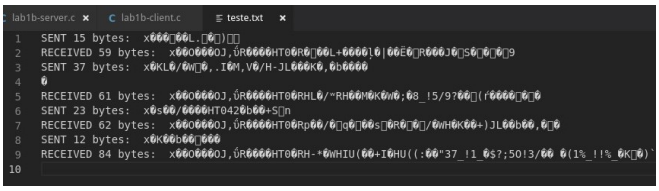

Figure 5. Arquivo .txt do logfile com o texto compactado

4. Conclusão

Grande parte do desenvolvimento do código do projeto foi realizado com o auxílio de pessoas que já desenvolveram o sistema de comunicação cliente e servidor utilizando TCP via socket e do shell com pipes. Além de grandes contribuições dos usuários da biblioteca zlib, pois a existência de várias aplicações utilizando-a nos orientou, servindo como modelo ou referência para nosso projeto.

Agradecimentos

Nós gostaríamos de agradecer ao Jean-Loupe Gailly e ao Mark Adler pela criação da biblioteca zlib que é distribuída de forma gratuita; Robert Ingalls, criador do tutorial de Sockets utilizado pra embasamento do projeto; Herbert Rocha por proporcionar a experiência para realização deste projeto.

References

[1] R. Ingalls *Sockets Tutorial* www.cs.rpi.edu

[2] J. Gailly e M. Adler *Zlib Library* www.zlib.net

[3] M. Sus *Compressed Network and Communication* github.com/michfit

[4] P. Deutsch *DEFLATE Compressed Data Format* www.w3.org

[5] A. Quinlan *Compress and then Decompress a string with zlib* gist.github.com

[6] Die.Net *tcsetattr(3) - Linux man page* linux.die.net

[7] Mks Software *Struct Termios* www.mkssoftware.com

[8] Programiz *C Programming Files I/O* www.programiz.com

[9] R. Shanker *Input-output system calls in C* www.geeksforgeeks.org

[10] Daemonio Labs *Funções getopt() e* $getopt_l ong()EmC$ *daemoniolabs.wordpress.com*