

UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA
CIÊNCIA DA COMPUTAÇÃO

Artur Luiz Rizzato Toru Soda
Davi Menegaz Junkes
Felipe Elton Pavini Savi

Trabalho 1 - Sistema Peer-to-Peer

Florianópolis
2024

1. INTRODUÇÃO

Este relatório tem como propósito demonstrar as principais decisões e estratégias tomadas durante o primeiro trabalho da disciplina de computação distribuída (INE5418), cujo objetivo é simular uma rede P2P não estruturada de compartilhamento de arquivos. Nesse sentido, cada peer possui arquivos armazenados localmente disponíveis para serem transferidos para outros peers, além disso, devem estar sempre preparados para requisitar ou receber requisições de recursos a qualquer momento.

Sendo assim, para fazer a busca deve-se utilizar o método de procura flooding, e é preciso que a transferência de recursos seja feita em chunks, sendo possível obter cada chunk de peers diferentes.

2. ESTRUTURA DO CÓDIGO

Este trabalho foi dividido nos seguintes arquivos: `main.cpp`, `parse_files.cpp`, `utils.cpp`, `parse_files.hpp`, `utils.hpp`, `structs.hpp`.

Os arquivos `parse_files.hpp` e `parse_files.cpp` possuem funções para ler os arquivos `topologia.txt` e `config.txt`.

Os arquivos `utils.hpp` e `utils.cpp` possuem funções utilitárias, como por exemplo funções para imprimir estruturas de dados na tela, serializar e desserializar os pacotes e funções auxiliares na interface com a API do Linux.

O arquivo `struct.hpp` possui estruturas fundamentais à lógica do programa, como as estruturas dos pacotes de requisição e resposta UDP, a estrutura `File` que armazena os dados de arquivo p2p e a estrutura `Data`, que armazena os dados utilizados ao longo do programa.

O arquivo `main.cpp`, por sua vez, possui a estrutura geral do programa, chamando as funções do módulo `parse_files`, salvando seus retornos na estrutura `Data`, utiliza as funções do módulo `utils` para auxiliar na lógica do programa.

Logo no início do programa, são criadas cinco threads principais, responsáveis pela funcionamento geral do programa:

2.1 RECEIVE_UDP_PACKETS_THREAD

Esta thread tem como objetivo receber todos os pacotes UDPs encaminhados ao peer, e, dependendo do cabeçalho do pacote, o qual pode indicar um pacote de requisição ou de resposta, ela encaminha o pacote ou para a thread `request_file_thread`, ou para a thread `respond_discovery_thread`. Além disso, todos os pacotes são encaminhados também à thread `retransmit_udp_packets_thread`.

2.2 RETRANSMIT_UDP_PACKETS_THREAD

Esta thread tem como objetivo retransmitir os pacotes UDP recebidos pela thread `receive_udp_packets_thread` para seus vizinhos um segundo após ter sido recebido.

2.3 REQUEST_FILE_THREAD

Esta thread tem como objetivo fazer a requisição de um arquivo, consistindo nas etapas:

1. Pedir ao usuário que digite o arquivo p2p que possui a informação de qual arquivo deve ser requisitado

2. Enviar pacotes UDP de requisição do arquivo
3. Esperar notificação de recebimento de resposta da thread `receive_udp_packets_thread`, criando a respectiva thread `receive_chunk_thread` para receber o chunk
4. Após esperar a finalização de todas as threads `receive_chunk_thread`, juntar os chunks em um único arquivo.

2.4 RESPOND_DISCOVERY_THREAD

Esta thread consiste nas etapas:

1. Esperar a notificação de chegada de requisição da thread `receive_udp_packets_thread`
2. Criar thread `send_chunk_thread` para cada chunk que possuir do arquivo requisitado
3. Retornar respostas UDP para cada chunk que possui do arquivo requisitado.

O.B.S.: Os chunks enviados serão aqueles presentes na pasta do nó, excluindo os quais estão sendo recebidos de outros nós no dado momento.

2.5 MANAGE_BYTES_TO_SEND_THREAD

Esta thread, a cada um segundo, atribui à capacidade de bytes de envio o valor da taxa de transmissão.

2.6 RECEIVE_CHUNK_THREAD

Esta thread será criada pela thread `request_file_thread`. Ela inicialmente se conectando ao servidor TCP que irá enviar o chunk. Após conectado, espera em um loop os bytes serem enviados pelo servidor e salva no respectivo arquivo de chunk. Após todos os bytes terem sido recebidos, finaliza sua execução.

2.7 SEND_CHUNK_THREAD

Esta thread será criada pela thread `respond_discovery_thread`. Ela inicialmente irá aguardar por uma conexão do cliente, até que um timeout seja alcançado, utilizando a função “poll”. Após conectar com o cliente, irá, em um loop, esperar a notificação da thread `manage_bytes_to_send_thread`, indicando que houve uma atualização na variável que representa a capacidade de bytes que podem ser enviados no momento, acessando-a assim por meio de um lock. Em seguida, envia os bytes do arquivo. Depois de todos os bytes terem sido enviados, finaliza sua execução.

3. EXECUÇÃO DO CÓDIGO

Para compilar o código, é necessário seguir as seguintes etapas:

1. Criar as pastas dos nós (0, 1, 2, 3, 4) e popula-las com seus chunks

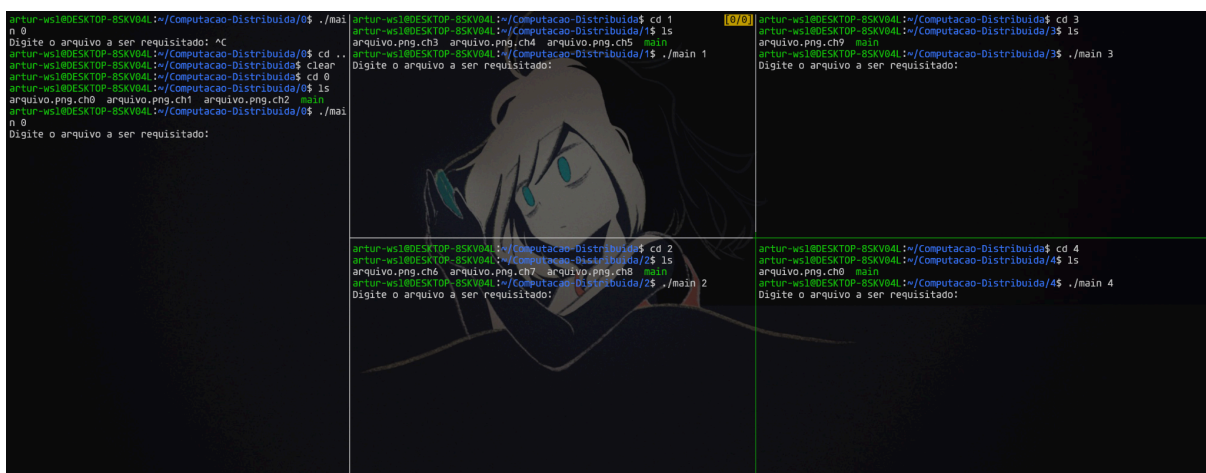
```
artur-wsl@DESKTOP-8SKV04L:~/Computacao-Distribuida$ mkdir 0 1 2 3 4
```

```

artur-wsl@DESKTOP-8SKV04L:~/Computacao-Distribuida/0$ nvim arquivo.png.ch0
artur-wsl@DESKTOP-8SKV04L:~/Computacao-Distribuida/0$ nvim arquivo.png.ch1
artur-wsl@DESKTOP-8SKV04L:~/Computacao-Distribuida/0$ nvim arquivo.png.ch2
artur-wsl@DESKTOP-8SKV04L:~/Computacao-Distribuida/0$ cd ../1
artur-wsl@DESKTOP-8SKV04L:~/Computacao-Distribuida/1$ nvim arquivo.png.ch3
artur-wsl@DESKTOP-8SKV04L:~/Computacao-Distribuida/1$ nvim arquivo.png.ch4
artur-wsl@DESKTOP-8SKV04L:~/Computacao-Distribuida/1$ nvim arquivo.png.ch5
artur-wsl@DESKTOP-8SKV04L:~/Computacao-Distribuida/1$ cd ../2
artur-wsl@DESKTOP-8SKV04L:~/Computacao-Distribuida/2$ nvim arquivo.png.ch6
artur-wsl@DESKTOP-8SKV04L:~/Computacao-Distribuida/2$ nvim arquivo.png.ch7
artur-wsl@DESKTOP-8SKV04L:~/Computacao-Distribuida/2$ nvim arquivo.png.ch8
artur-wsl@DESKTOP-8SKV04L:~/Computacao-Distribuida/2$ cd ../3
artur-wsl@DESKTOP-8SKV04L:~/Computacao-Distribuida/3$ nvim arquivo.png.ch9
artur-wsl@DESKTOP-8SKV04L:~/Computacao-Distribuida/3$ cd ../4
artur-wsl@DESKTOP-8SKV04L:~/Computacao-Distribuida/4$ cp ../1/arquivo.png.ch0 ./arquivo.png.ch0
cp: cannot stat '../1/arquivo.png.ch0': No such file or directory
artur-wsl@DESKTOP-8SKV04L:~/Computacao-Distribuida/4$ cp ../0/arquivo.png.ch0 ./arquivo.png.ch0
artur-wsl@DESKTOP-8SKV04L:~/Computacao-Distribuida/4$

```

2. Rodar o Makefile com o comando “make all” na pasta base. Isso criará a pasta build, com os arquivos parse_files.o, utils.o e main. O arquivo main vai ser copiado para as respectivas pastas dos nós (0, 1, 2, 3, 4)
3. Se for rodar localmente, abrir um terminal separado e rodar o comando “cd” para a pasta do respectivo nó e rodar o comando “main”, passando como argumento o nó em questão. Por exemplo, para o nó 0, rodar o comando “cd 0” seguido de “./main 0”.



```

artur-wsl@DESKTOP-8SKV04L:~/Computacao-Distribuida/0$ ./mai
n 0
Digite o arquivo a ser requisitado: ^C
artur-wsl@DESKTOP-8SKV04L:~/Computacao-Distribuida/0$ cd ..
artur-wsl@DESKTOP-8SKV04L:~/Computacao-Distribuida$ clear
artur-wsl@DESKTOP-8SKV04L:~/Computacao-Distribuida$ cd 0
artur-wsl@DESKTOP-8SKV04L:~/Computacao-Distribuida/0$ ls
arquivo.png.ch0 arquivo.png.ch1 arquivo.png.ch2 main
artur-wsl@DESKTOP-8SKV04L:~/Computacao-Distribuida/0$ ./mai
n 0
Digite o arquivo a ser requisitado:

artur-wsl@DESKTOP-8SKV04L:~/Computacao-Distribuida$ cd 1
artur-wsl@DESKTOP-8SKV04L:~/Computacao-Distribuida/1$ ls
arquivo.png.ch3 arquivo.png.ch4 arquivo.png.ch5 main
artur-wsl@DESKTOP-8SKV04L:~/Computacao-Distribuida/1$ ./main 1
Digite o arquivo a ser requisitado:

artur-wsl@DESKTOP-8SKV04L:~/Computacao-Distribuida$ cd 3
artur-wsl@DESKTOP-8SKV04L:~/Computacao-Distribuida/3$ ls
arquivo.png.ch9 main
artur-wsl@DESKTOP-8SKV04L:~/Computacao-Distribuida/3$ ./main 3
Digite o arquivo a ser requisitado:

artur-wsl@DESKTOP-8SKV04L:~/Computacao-Distribuida$ cd 2
artur-wsl@DESKTOP-8SKV04L:~/Computacao-Distribuida/2$ ls
arquivo.png.ch6 arquivo.png.ch7 arquivo.png.ch8 main
artur-wsl@DESKTOP-8SKV04L:~/Computacao-Distribuida/2$ ./main 2
Digite o arquivo a ser requisitado:

artur-wsl@DESKTOP-8SKV04L:~/Computacao-Distribuida$ cd 4
artur-wsl@DESKTOP-8SKV04L:~/Computacao-Distribuida/4$ ls
arquivo.png.ch0 main
artur-wsl@DESKTOP-8SKV04L:~/Computacao-Distribuida/4$ ./main 4
Digite o arquivo a ser requisitado:

```

4. O programa imprimirá a mensagem “Digite o arquivo a ser requisitado: “, pedindo para digitar o arquivo p2p. Quando respondido com o respectivo arquivo, ele iniciará a requisição.

