# Comunicações por Computador - TP2 - GR68

Duarte Afonso Freitas Ribeiro :: A100764
Afonso Xavier Cardoso Marques :: A94940
Pedro Duarte Nobre Viana :: A100701

Dezembro 2023

Universidade do Minho
Escola de Engenharia

# 1 Introdução

O presente relatório foi desenvolvido no âmbito da unidade curricular de Comunicações por Computador como proposta de resolução ao Trabalho Prático 2. Os leitores encontram no documento informação detalhada sobre o sistema que foi implementado bem como as conclusões que o grupo retirou face aos resultados obtidos.

O objetivo do trabalho proposto é desenhar, implementar e testar um serviço avançado de partilha de ficheiros numa rede peer-to-peer (P2P) (semelhante a "BitTorrent") com múltiplos servidores, que são também clientes do mesmo serviço e que garanta um bom desempenho e recorra a um protocolo de transferência a correr sobre o protocolo de transporte UDP. Os ficheiros podem estar disponíveis em mais que um peer, e podem ser transferidos de qualquer um deles ou mesmo de vários deles ao mesmo tempo, por partes, com o intuito de melhorar a disponibilidade e o desempenho.

# Contents

# 2 Arquitetura da Solução

De modo a desenvolver o nosso serviço, tivemos à partida de criar um modelo arquitetónico onde estão desenhados a estrutura das funcionalidades requeridas. Para isso, foi utilizada a arquitetura proposta no enunciado do trabalho prático:

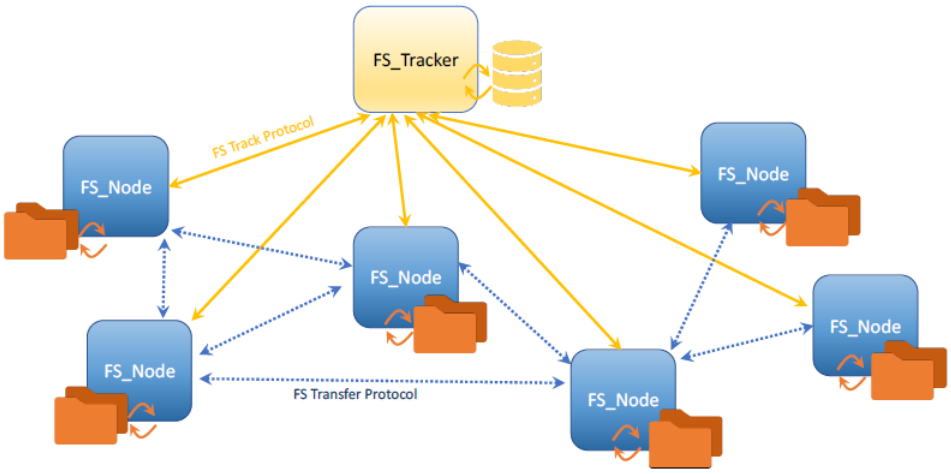


Figure 1: Arquitetura proposta

## 2.1 FS_Tracker

Este é o servidor que fica à escuta de conexões FS Track Protocol na porta TCP 12345 e que manterá uma conexão ativa por cada FS_Node que é registado no serviço. Vários nodos irão estar conectados a este servidor, pelo que é necessário correr em simultâneo vários processos. Sendo assim, para cada conexão que o servidor aceita, é criada uma nova *thread* que lida com todos os pedido que sejam feitos pelo nodo até que este deixa de estar conectado. O primeiro pedido de todos é o de registo.

O servidor *tracker* possui a capacidade de guardar informação sobre os nodos que no momento tem estabelecida uma conexão com ele. Inicialmente esta base de informação interna está vazia, o que significa que o *tracker* não tem conhecimento de nenhum FS_Node na rede. O pedido de registo trata de associar ao nodo que fez o pedido os ficheiros que este contem. A informação contida nesta base de dados está disponível para todos os nodos que estejam conectados ao serviço.

Outros tipos de pedidos suportados pelo *tracker* incluem a adição e remoção de um ficheiro a um nodo, a eliminação de um nodo (ou seja quando este deseja desconectar-se) e o pedido de transferência de um ficheiro.

Toda a comunicação estabelecida com o *tracker* é feita com recurso ao protocolo FS Track Protocol, que é um dos requisitos do serviço e funciona sobre TCP.

### 2.2 FS_Node

Esta é a aplicação que corre para todos os nodos da rede que não sejam o *tracker*. Este serviço, quando inicia a sua execução, começa por se conectar à porta TCP 12345 do FS_Tracker, utilizando de seguida o protocolo FS Track Protocol para informar dos ficheiros ou blocos a seu cargo. Ao mesmo tempo, fica à escuta, em permanência, na porta UDP 9090 por pedidos de blocos segundo o protocolo FS Transfer Protocol. Este efeito é obtido através da criação de uma thread que lida com este tipo de pedidos.

## 3 Especificação de Protocolos

Para o bom funcionamento do serviço e de forma a cumprir com os requisitos do trabalho, foram criados dois protocolos de comunicação entre os nodos da rede: o FS Track Protocol que lida com toda a comunicação feita entre nodos e o servidor *tracker* e FS Transfer Protocol que gere as transferências de ficheiros entre os vários nodos.

### 3.1 FS Track Protocol

Tal como foi referido na secção anterior, foi criado um protocolo de troca de mensagens entre *tracker* e nodos, o FS_Node, que funciona sobre TCP. Para esse efeito, foram criadas várias classes que representam os diferentes tipos de dados que vão ser encapsulados num pacote dependendo do tipo de pedido que foi feito, por exemplo para registar um nodo existe uma classe *Register_Node_Packet* cuja informação serializada (ou seja convertida em bytes) é encapsulada num pacote final, o *tracker packet*. É este pacote final que vai ser enviado entre nodo e servidor.

Os pacotes têm uma estrutura bastante simples, com um campo para indicar o tipo do pacote (representado sob a forma de um número inteiro) e um campo para os dados encapsulados (em formato de bytes). Segue uma ilustração:

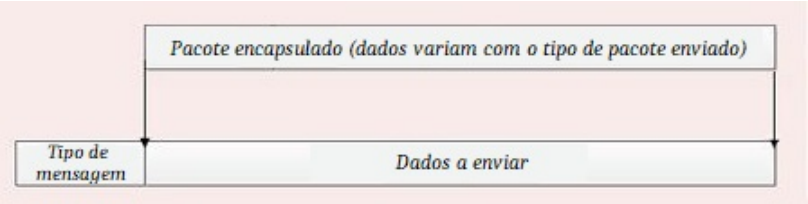


Figure 2: Informação contida no *tracker packet*

### 3.2 FS Transfer Protocol

Relativamente à comunicação que decorre entre os nodos que executam o FS_Node, esta deve ser executada através de um outro protocolo que, contrariamente ao anterior, funciona sobre UDP. Em termos de implementação de funcionalidades, este protocolo é mais focado no processo da transferência propriamente dita de ficheiros entre nodos. Este protocolo faz uso da biblioteca *hashlib* para verificar a integridade dos pacotes durante as transferências.

O protocolo consiste em dividir os dados de um ficheiro em *chunks* e enviar cada um encapsulado num pacote separado, o *transfer packet*. Cada pacote deste tipo possui informação sobre o valor *hash* do *chunk*, o tamanho dos dados, o valor de *hash* do ficheiro, o identificador do *chunk* e os dados do ficheiro em si, tudo em bytes.

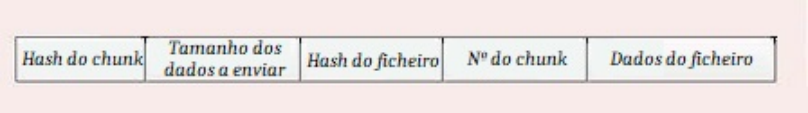


Figure 3: Informação contida no *transfer packet*

Cada pacote que é recebido por um nodo é testado relativamente à sua integridade, isto é calcula-se o valor *hash*, com recurso à função MD5, dos dados recebidos e compara-se esse valor com o *hash* que foi transmitido junto com os dados. Se houver uma diferença entre os dois, isto indica que os dados podem ter sido corrompidos ou adulterados durante a transmissão, e é retornado uma mensagem de erro.

## 4 Implementação

### 4.1 FS Tracker Protocol

O protocolo tracker_protocol utiliza um formato de mensagem composto por três principais partes: o tipo de mensagem (1 *byte*), o tamanho da mensagem (2 *bytes*) e os dados da mensagem.

O protocolo define vários tipos de mensagens, cada uma corresponde a uma operação especifica:

- REGISTER_NODE (0): Mensagem para registar um nó na rede.
- DELETE_NODE (1): Mensagem para remover um nó da rede.
- OKAY (2): Mensagem a indicar que uma operação foi concluída com sucesso.

- FILE_NOT_FOUND (3): Mensagem a indicar que um ficheiro não foi encontrado.
- ADD_FILE((10): Mensagem que indica que um novo ficheiro está no *node*.
- REMOVE_FILE(11): Mensagem que indica que um ficheiro já não está no *node*.
- REQUEST_FILE(12): Mensagem que sinaliza um pedido de um ficheiro.
- REQUEST_FILE_RESPONSE(13): Resposta a um pedido de um ficheiro.

Cada tipo de mensagem é representado por uma classe dedicada, e possui métodos de serialização e desserialização. Os objetos são serializados para facilitar a sua transmissão pela rede.

## 4.2 FS Tranfer Protocol

O protocolo *transfer_protocol* permite a troca de *chunks* de arquivos entre *nodes*, facilitando a construção da base de dados distribuída.

É utilizado um formato de mensagem que inclui um *header* detalhado que garante a integridade dos dados transmitidos, sendo que cada mensagem é composta por:

- 16 bytes - chunk hash: O *hash* do conteúdo do *chunk*, usado para verificar integridade dos dados.
- 2 bytes - message length: O tamanho da mensagem, indicando a quantidade de dados no chunk.
- 40 bytes - file hash: O *hash* do ficheiro relacionado com o *chunk*, com um comprimento fixo.
- 2 bytes - file chunk number: O número do *chunk* no contexto do ficheiro.
- Dados da mensagem: Os dados reais do *chunk*.

A implementação utiliza sockets UDP para comunicação na rede, tal como foi especificado no enunciado.

## 4.3 File Info

A classe *file_info* é usada para organizar as informações dos ficheiros na rede P2P. Gere os detalhes dos ficheiros como o nome, o tamanho, os chunks disponíveis e que *nodes* possuem esses chunks. Permite adicionar os *nodes* que têm o ficheiro e especificar quais os chunks que este têm, caso não tenha o ficheiro completo. Também permite ordenar a lista de *nodes* que têm um ficheiro

através de um *rating* calculado através de penalidades e recompensas com base no sucesso e tempo envolvido numa transferência, estes *ratings* são dinâmicos e são usados para otimizar futuras transferências.

## 4.4 Node

O código do *node* é responsável por gerir o registo do *node* no *tracker*, o envio e receção de pacotes, as solicitações de download de ficheiros, e outras operações.

Inicialmente o *node* cria uma *pool* de *sockets* para transmitir e receber pacotes, também cria um *socket* dedicado apenas a receber pedidos de *chunks*.

O *node* regista-se no *tracker*. Entra de seguida num *loop* onde espera por operações a serem realizadas pelo utilizador, como listar *files* ou fazer download de um *file* disponível no tracker. Em caso de encerramento da conexão, todos os *sockets* e *threads* são adequadamente encerradas e o *tracker* informado.

Para efetuar o download de um ficheiro o node atribui os processos de solicitar chunks a threads. O chunk do ficheiro a ser pedido é escolhido a partir da sua raridade, é usada uma distribuição exponencial para dar prioridade aos chunks mais raros. São então enviadas solicitações aos nodes desejados. Após a receção dos chunks, os ratings dos nodes são atualizados de acordo com o desempenho.

## 4.5 Tracker

O tracker é responsável por gerir e manter a conexão entre *nodes*. Utiliza uma abordagem através de uma *thread* por conexão para gerir múltiplas conexões simultaneamente. O tracker processa os diferentes tipos de mensagens enviadas pelos nodes, como registo de nodes, atualização de ficheiros, pedidos de ficheiros, remoção de nodes, remoção de ficheiros e listagem de ficheiros.

Cada conexão com um node é gerida separadamente por uma thread, nela são recebidas as mensagens e enviadas as respostas apropriadas.

O tracker mantém várias estruturas de dados para armazenar as informações relativas aos *nodes* e aos ficheiros na rede. Os ficheiros na rede são armazenados num mapa em que a chave é a hash do ficheiro e o valor é uma instância de uma classe que contem o nome, o tamanho e uma lista de tuplos, em que cada tuplo possui um chunk e uma lista com os nodes que possuem esse chunk do ficheiro. Outra estrutura de dados do tracker é a lista de nodes registados no tracker. Ambas as estruturas de dados são protegidas por um *lock* de forma a garantir a consistência dos dados no ambiente *multithread*.

# 5 Testes e Resultados

O sistema implementado e os seus protocolos foram testados na seguinte topologia de rede fornecida pela equipa docente. Neste caso de teste, vamos colocar o PC1 a fazer download de um ficheiro que está na posse do Portatil1.

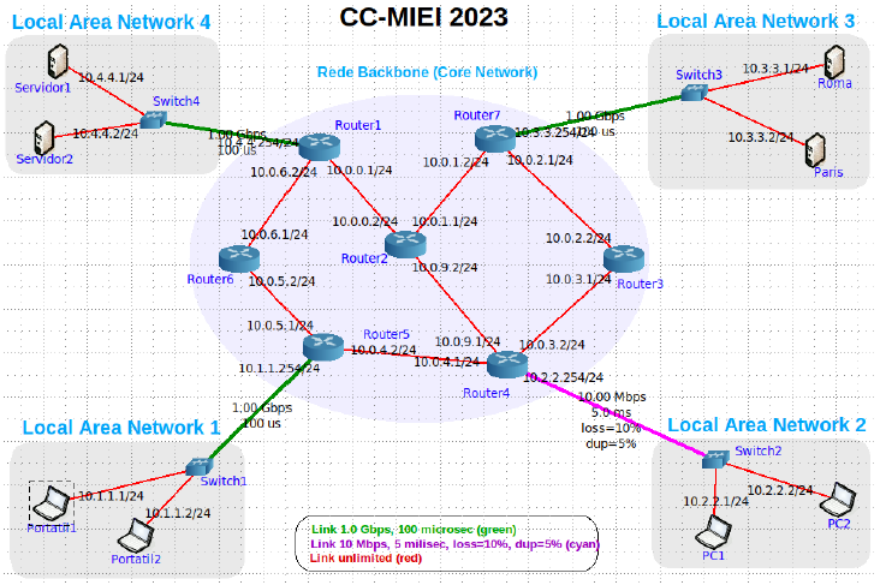


Figure 4: Topologia de rede

Cada um destes nodos tem uma pasta onde devem ser alojados os seus ficheiros. Podemos ver que a pasta do PC1 está vazia enquanto que o ficheiro file1.txt está na pasta respetiva ao Portatil1.

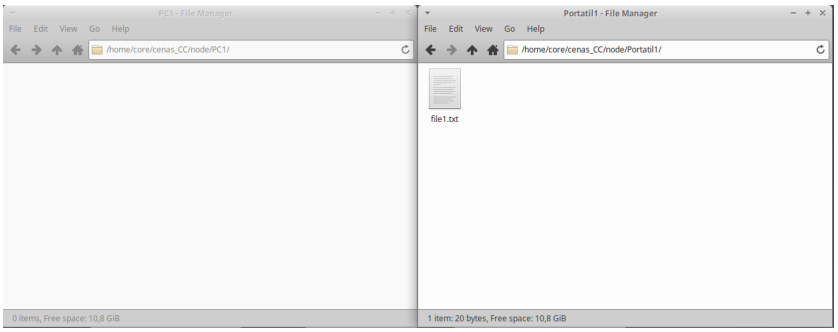


Figure 5: Diretorias dos nodos

Apesar do ficheiro estar na diretoria do nodo Portatil1, é necessário indicar ao serviço que esse ficheiro existe e está naquele nodo. Para isso, após fazer o seu registo, através do comando "add <filename>", adiciona-se à base de dados do *tracker* o novo ficheiro com um valor de *hash* especifico. Podemos comprovar que este ficheiro existe e está acessível aos outros nodos quando, no PC1, executamos o comando "list", que nos vai devolver precisamente uma lista onde a cada ficheiro associa um *hash*. Realçamos que a origem do ficheiro nunca é revelada ao PC1.

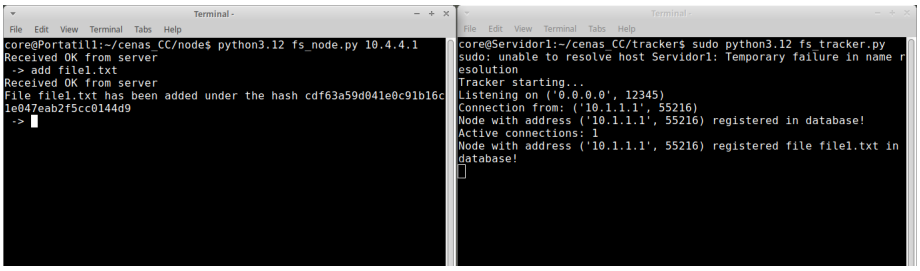


Figure 6: Adição do ficheiro ao serviço (Terminal Portatil1)

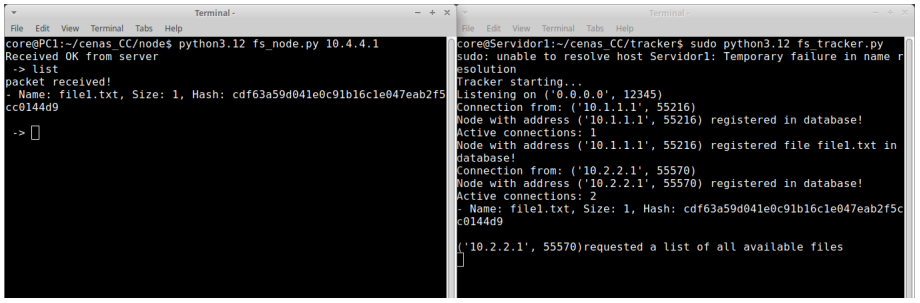


Figure 7: Pedido de listagem (Terminal PC1)

Finalmente a parte do download do ficheiro. O PC1 faz um pedido de "get <file_hash>" ao *tracker* que despoleta o processo de transferência entre o Portatil1 e o PC1 recorrendo ao protocolo FS Transfer Protocol.

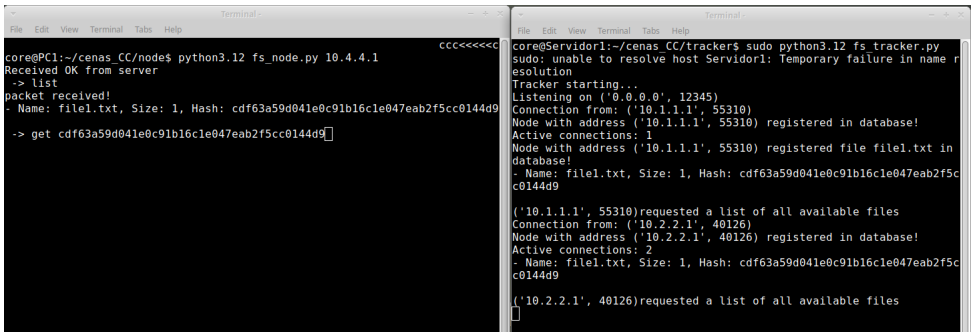


Figure 8: Pedido de download de ficheiro (Terminal PC1)

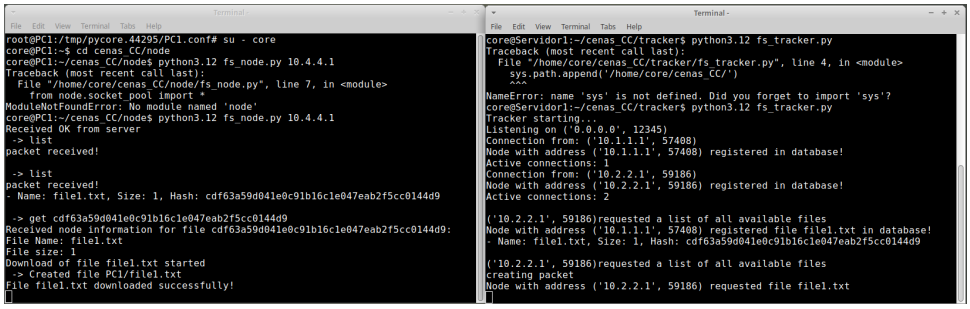


Figure 9: Processo de transferência (Terminal PC1)

O resultado é o esperado com o *file1.txt* a aparecer na diretoria do Portatil1 bem como na do PC1. A integridade do ficheiro não foi comprometida.

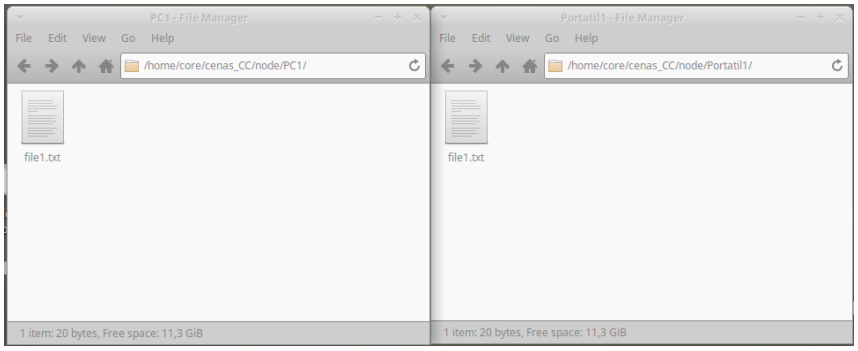


Figure 10: Diretorias após a transferência

# 6 Comentários Finais

Em modo de conclusão, neste trabalho prático aprofundámos o nosso conhecimento relativamente à área de Comunicação por Computadores, pois enquanto este se encontrava em fase de desenvolvimento deparamos-nos com diversos problemas, erros e mesmo novas oportunidades de implementar outras funcionalidades.

Destacamos pela positiva o sucesso na implementação dos dois protocolos, o FS Tracker Protocol e FS Transfer Protocol, bem como as estratégias usadas para a conceção dos mesmos.

# 7 Anexos

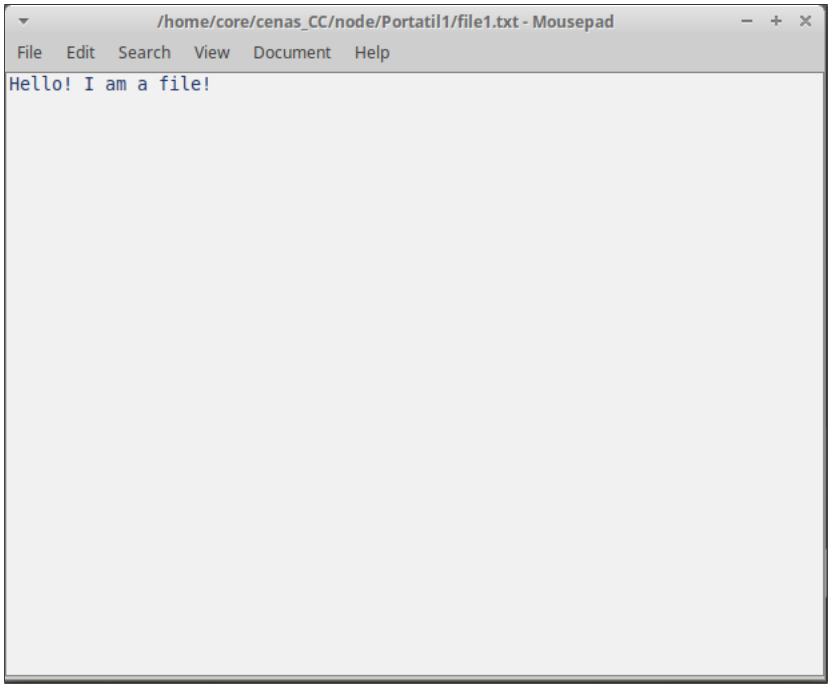


Figure 11: Conteúdo do file1.txt