Comunicações por Computador
Relatório trabalho prático 2
PL5 Grupo 2

Lucas Oliveira, Mike Pinto, and Rafael Gomes

Universidade do Minho, Departamento de Informatica, 4710-057 Braga, Portugal
e-mail: {a98695,a89292,a96208}@alunos.uminho.pt

Resumo Este trabalho consiste no desenvolvimento, implementação e teste de um serviço de partilha de ficheiros *peer-to-peer* (*P2P*) de alto desempenho. Este serviço é composto por uma rede de dispositivos designados de *FS_Nodes* que consistem simultaneamente em serviço cliente/servidor. Cada *FS_Node* conecta-se a um dispositivo de registo de conteúdo, designado por *FS_Tracker* que regista a localização de todos os ficheiros disponíveis nos *FS_Nodes*. As comunicações entre *FS_Node* e *FS_Tracker* são realizadas mediante um protocolo designado por *FS_Tracker_Protocol* que corre sobre *TCP*. Já as comunicações entre *FS_Nodes* são realizadas mediante o *FS_Transfer_Protocol* que corre sobre *UDP*.

1 Introdução

Um dos principais pilares que sustentam a *world wide web* é a troca de informação. Esta é uma ferramenta crucial da sociedade de hoje em dia, impulsionando a disseminação rápida e global de conhecimento, promovendo a conectividade entre indivíduos, organizações e culturas diversas. A facilidade com que a informação é compartilhada na *web* não só encurta as distâncias geográficas, mas também desempenha um papel vital na formação de comunidades virtuais e na aceleração do progresso em diversos setores, moldando assim a dinâmica e a evolução da sociedade.

Deste modo, toda a transferência de informação tem de ser rápida e fiável. O mesmo se aplica na troca de ficheiros, onde um *host* localiza e identifica um ficheiro disponibilizado num *servidor* e o transfere. Neste artigo iremos explorar o conceito de transferência de ficheiros numa rede *peer-to-peer* (*p2p*) de alto nível, onde os ficheiros são partilhados diretamente entre dois *hosts*, mediante o uso de um servidor de registo de conteúdo, que partilhara a localização de todos os ficheiros dos *hosts* conectados.

Esta rede será formada por um servidor de registo de informação designado de *FS Tracker* e de vários clientes designados por *FS Node's* que irão partilhar com o *FS Tracker* toda a informação de ficheiros existentes na sua pasta partilhada(nome do ficheiro, tamanho total, informação sobre os blocos do ficheiro) por um protocolo de comunicação designado de *FS Tracker Protocol* que irá funcionar sobre comunicação *TCP*. Similarmente, um *FS Node* irá servir como um Cliente/Servidor ao nível de troca de ficheiros entre si, onde um *FS Node* cliente irá solicitar blocos de um determinado ficheiro a um *FS Node* servidor. Esta comunicação irá funcionar via um protocolo designado de *FS Transfer Protocol* sobre comunicações *UDP*.

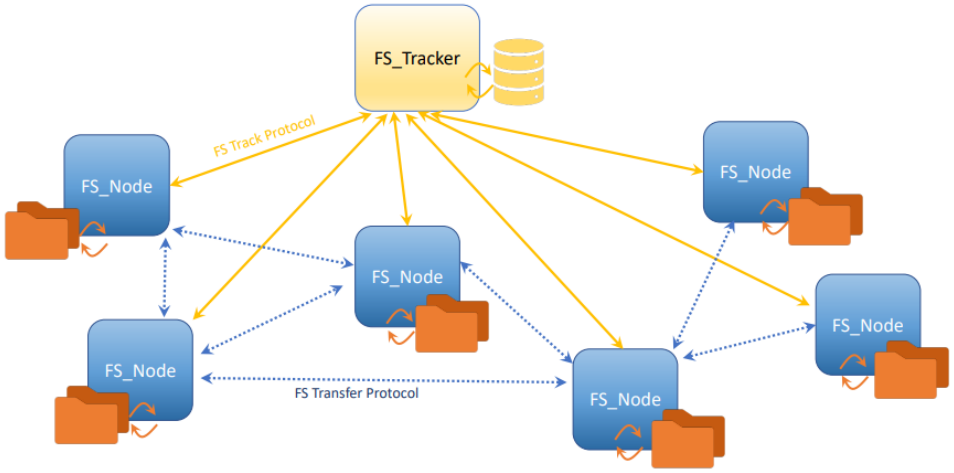

Figura 1. Esquema geral de funcionamento.

2 Arquitetura da Solução

Após a análise e recolha dos requisitos da aplicação, foi sugerido uma solução recorrendo a *Threads* para garantir a multiplexagem de clientes e o paralelismo de funcionamento da aplicação. Foi ainda proposto uma arquitetura modular no qual permitisse a evolução do *"software"*.

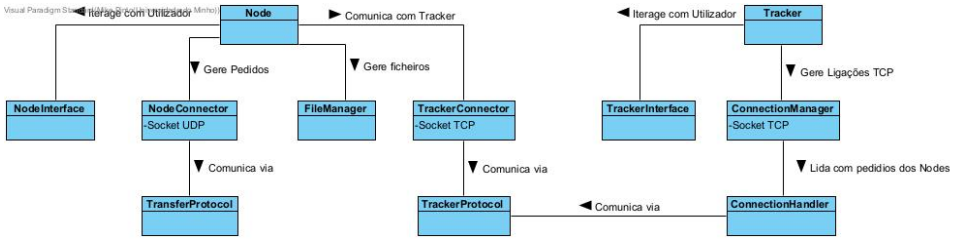

Figura 2. Proposta de Arquitetura

2.1 FS Tracker

O programa *FS_Tracker* tem início na classe *FS_Tracker* que se responsabiliza pela leitura dos argumentos(como a porta a utilizar para a comunicação *TCP*), por todos os métodos necessários para o bom funcionamento da aplicação, pela criação de uma *Thread* para a classe ***FS_Tracker_Connection_Manager*** que irá ficar à escuta de novas ligações TCP provenientes de outros *FS Node's* na porta recebida como argumento. Esta classe está ainda responsável pela chamada da função que inicializa a "interface" em texto do *FS Tracker* da classe ***FS_Tracker_Interface***.

FS_Tracker_Connection_Manager: Esta classe está responsável por ficar à escuta por conexões *TCP* provenientes dos *FS Node's* e de criar uma *Thread* da classe *Tracker_Connection_Handler* por cada conexão recebido.

FS_Tracker_Connection_Handler: Esta classe está responsável pela gestão e satisfação dos pedidos recebidos provenientes dos *FS Node's*.

2.2 FS Node

O programa *FS Node* tem início na classe *FS_Node* e é responsavel pela leitura dos argumentos de inicio(endereço/nome do tracker, porta a utilizar pelos *Sockets* e caminho para a pasta partilhada), por todos os métodos necessários para o bom funcionamento da aplicação, pela criação de *Threads* para as classes, *NodeConnector*, *FileManager*, *Tracker-Connector* e pela chamada da função que inicializa a "interface" de texto do *FS Node*.

NodeConnector: Esta classe é responsável pelo envio e receção de mensagens entre os *FS Node's*.

FileManager: Esta classe é responsável pela leitura, escrita e atualização dos ficheiros presentes na pasta partilhada.

TrackerConnector: Esta classe é responsável pela conexão com o *FS Tracker* bem como o envio e receção de mensagens entre estes.

3 Especificação Protocolocar

Com o intuito de gerir as comunicações entre *FS Tracker* e *FS Node* e entre *FS Node* e *FS Node*, foram desenvolvidos dois protocolos de comunicação, um com o funcionamento sobre *TCP*, o *FS Tracker Protocol* para estruturar mensagens de dados com informações de ficheiros de um *FS Node* e um *FS Tracker* e outro para estrutrar mensagens com blocos de um ficheiro que irá funcionar sobre *UDP* e servirá de recurso a comunicações entre *FS Node* e *FS Node* designado por *FS Transfer Protocol*.

3.1 FS Tracker Protocol

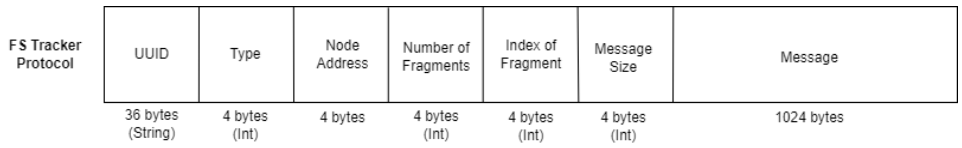

Figura 3. Estrutura do cabeçalho do FS Tracker Protocol

Este protocolo possui 6 campos de cabeçalho, não contando a mensagem. O tamanho do cabeçalho é de 56 bytes + 1024 bytes de mensagem, resultando num total de 1080 bytes. Todas as mensagens são enviadas com um tamanho de 1024 bytes mais o tamanho do cabeçalho, caso a mensagem seja de tamanho superior, a mensagem é fragmentada e criado um cabeçalho novo para cada uma com o mesmo código *UUID*. Semelhantemente, caso a mensagem seja de tamanho inferior então é preenchida a mensagem com bytes de valor '1' até ao tamanho de 1024 bytes.

Foi decidido manter um campo com o endereço do *FS Node* para que este protocolo possa ser utilizado num futuro com diferentes tipos de conexão se necessário.

– **UUID:** Identificador único com 36 bytes, gerado pela classe *UniqueIdGenerator* do *DataUtilities*.
– **Type:** Tipo da mensagens que podem ser:
  - 1 - Envio da lista de informação de ficheiros da pasta partilhada;
  - 2 - Envio de informação de um novo ficheiro;
  - 3 - Envio de informação de um ficheiro para remover;
  - 4 - Pedido da lista de ficheiros para transferir;
  - 5 - Resposta com a lista de ficheiros para transferir;
  - 6 - Pedido da localização de um ficheiro;
  - 7 - Resposta com a localização de um ficheiro;
  - 8 - Termino de comunidação;
  - 9 - Mensagem de Erro;
  - 10 - Atualização da informação de um ficheiro;
– **Number of fragments:** Numero de fragmentos total da mensagem.
– **Message Size:** Tamanho em bytes da mensagem enviada.

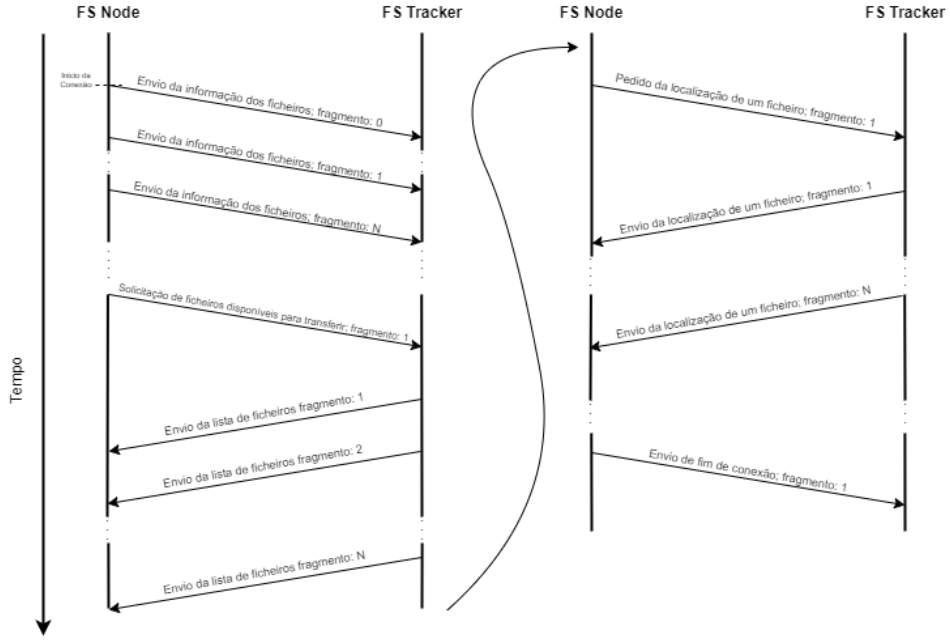

Figura 4. Exemplo de interações entre um *FS Node* e um *FS Tracker* mediante o protocolo *FS Tracker Protocol*.

3.2 FS Transfer Protocol

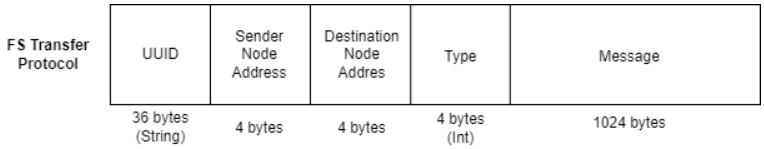

Figura 5. Estrutura do cabeçalho do FS Transfer Protocol

Este protocolo possui 4 campos de cabeçalho, excluindo a mensagem. O cabeçalho possui um tamanho de 48 bytes + 1024 bytes de mensagem. Foi definido previamente que o envio de um ficheiro iria se realizar através de blocos com um tamanho de 1024 bytes, sendo que então o tamanho máximo deste protocolo será de 1072 bytes, não sendo necessário a mensagem ser então fragmentada devido a possuir sempre o mesmo tamanho.

- **UUID:** Identificador único com 36 bytes, gerado pela classe *UniqueIdGenerator* do *DataUtilities*.
- **Sender Node Address**: Endereço do Node que enviou a mensagem.
- **Destination Node Address**: Endereço do Node a qual se destina a mensagem.
- **Type:** Tipo da mensagem, que podem ser:
  - 0 - Mensagem de Acknowledge, onde no campo vai a mensagem recebida para o controlo de erros;
  - 1 - Pedido de envio de um bloco de um ficheiro, o campo da mensagem possui o nome do ficheiro e o índice do bloco em interesse;
  - 2 - Envio do bloco do ficheiro pedido;
  - 10 - Envio de uma mensagem de Erro, campo usado com o intuito de informar um Node de um possível erro de envio da mensagem;

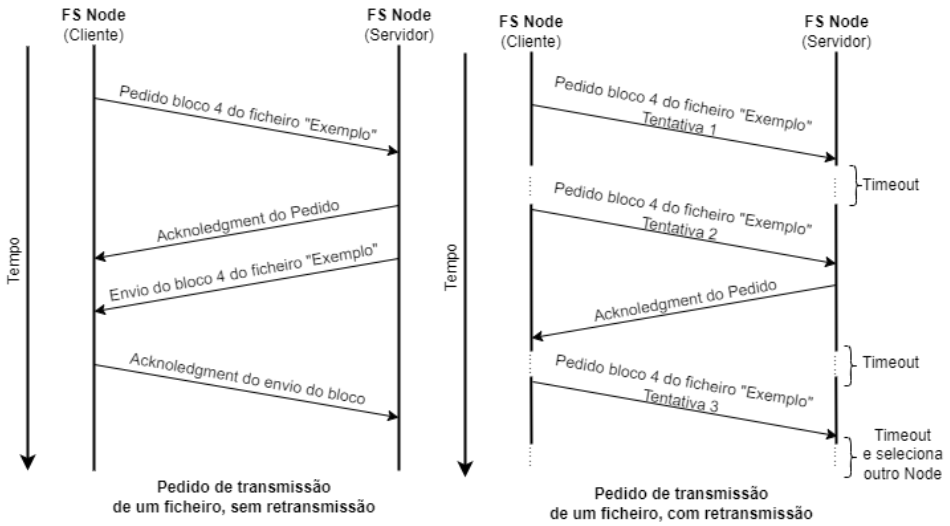

Figura 6. Exemplo de interações entre um *FS Node* cliente e um *FS Node* servidor mediante o protocolo *FS Transfer Protocol*.

4 Implementação

Após a fase de especificação da arquitetura do sistema e da especificação protocolar deu-se o início da fase da implementação do sistema de troca de ficheiros. A linguagem de programação utilizada para o desenvolvimento foi ***Java*** à familiarização com a mesma e por ser uma linguagem orientada aos objetos.

Deste modo foram criados *packages* como *FS_Tracker, FS_Node, FS_Tracker_Protocol, FS_Transfer_Protocol, Files e DataUtilities*.

4.1 FS_Tracker

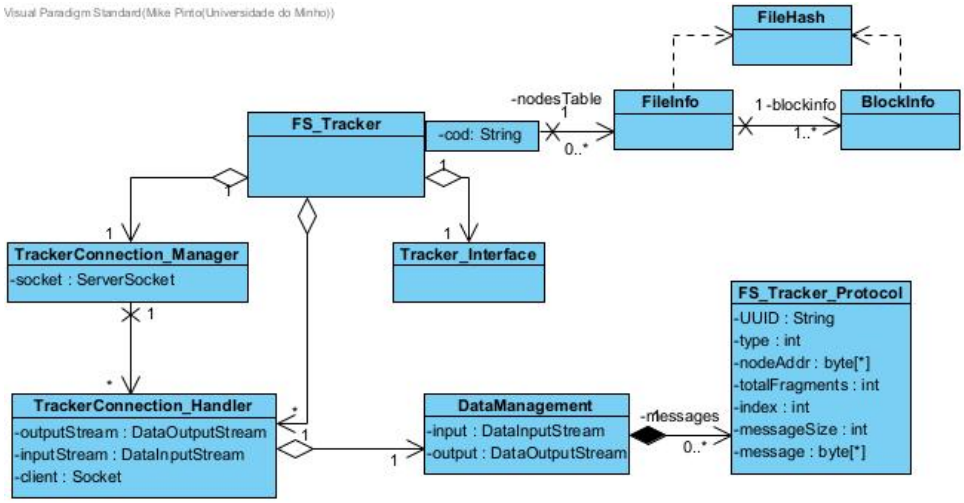

Figura 7. Diagrama de classes simplificado do *FS Tracker*

Este *Package* possui todas as classes e módulos de funcionamento do *FS Tracker* como, *TrackerInterface*, classe responsável pela interface em texto, a classe FS_Tracker_Connection_Manager responsável por ficar à escuta por ligações no socket TCP e criar *Threads* da classe *FS_Tracker_Connection_Handler* responsável pela satisfação dos pedidos de um *FS Node*. Cada *Thread* criada representa uma ligação de um *FS Node* e fica em execução até ao término da conexão. A classe *FS_Tracker* é ainda responsável por guardar "logs"dos diversos acontecimentos do sistema e escrevê-los num ficheiro de texto na pasta do sistema operativo *"/var/Backups/FSTracker/"*.

4.2 FS_Node

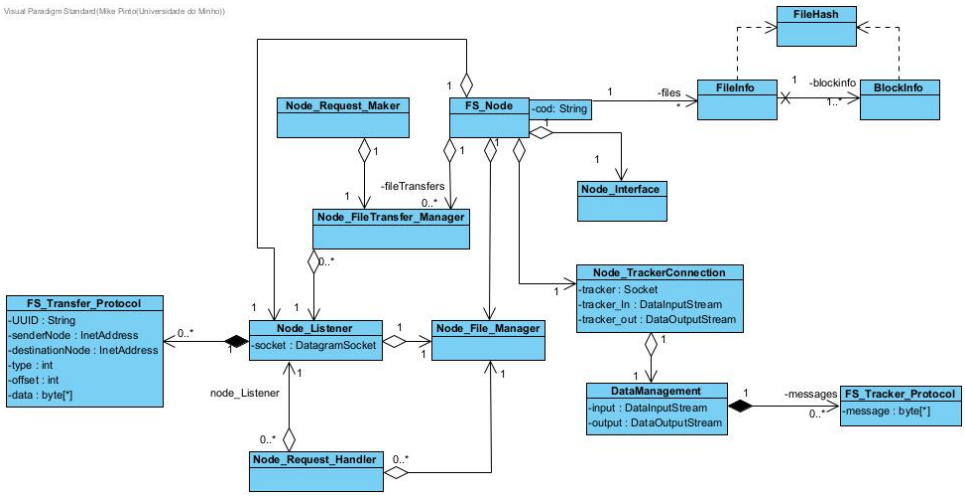

Figura 8. Diagrama de Classes simplificado do *FS Node*

Este *Package* possui todas as classes e módulos de funcionamento do *FS Node* como as classes para fazer e gerir pedidos(*Node_Request_Handler* e *Node_Request_Maker*) no qual recorrem à classe *NodeConnector* no qual possui toda a lógica necessária para o envio e receção de mensagens utilizando o protocolo *FS_Transfer_Protocol*. A classe *Node_File_Manager* responsável por guardar, adicionar, alterar e gerir a informação dos ficheiros pertencentes à pasta partilhada e por ler e escrever os ficheiros. A escrita do ficheiro é feita com recurso à criação de um ficheiro temporário na pasta do sistema operativo *"/tmp/FSCache/<identificação do Node>/*, tendo o seguinte modo de funcionamento:

- Ao receber um novo bloco de um ficheiro, é verificado se o *Node* já possui algum bloco desse ficheiro.
- Em caso negativo é simplesmente escrito esse bloco num novo ficheiro na pasta partilhada.
- Em caso afirmativo, o ficheiro já escrito na pasta partilhada é lido em memória, sendo criado um ficheiro na pasta temporária onde é realizado a escrita dos blocos pela ordem correta. No final da organização dos blocos, é copiado a nova versão do ficheiro para o ficheiro original da pasta partilhada e apagado o ficheiro temporário.

No final da escrita do ficheiro é enviado ao *FS Tracker* a nova informação do ficheiro. Esta classe também é responsável por verificar se a pasta partilhada sofreu alguma alteração durante o *Runtime* da aplicação(verificando se foi adicionado/removido algum ficheiro).

O *Package* do *FS_Node* possui ainda a classe *Node_Tracker_Connector* responsável pela gestão das mensagens enviadas e recebidas do *FS Tracker*. Esta classe utiliza a classe *DataManagement* do *Package FS_Tracker_Protocol* que possui toda a lógica para o envio e receção de mensagens através deste protocolo. Para além desta classe o *Node* possui também a classe *Node_Interface* responsável por toda a "interface" em texto para interação com o utilizador.

Por fim, a classe *FS_Node* ainda é responsável, no final da execução do programa, guardar um ficheiro de configuração e de "logs" do programa na pasta *"/var/Backups/FSNode/Identificação do Node/"*. O ficheiro de configuração não passa por ser uma cópia da informação dos ficheiros que o *Node* possui no final do programa, para ser possível, ao abrir o programa novamente, o utilizador solicitar os restantes blocos de um ficheiro que não tenha sido concluído a transferência e não se perca a ordem dos blocos escritos.

4.3 FS_Tracker_Protocol

Este *Package* possui apenas duas classes, a classe *FS_Tracker_Protocol* onde está estruturado o *Header* do protocolo, bem como as respetivas funções de *Serialização* e *Desserialização* do objeto de/e para um *array de bytes*. Para além desta classe, este *Package* possui ainda a classe *DataManagement* utilizada para enviar e receber mensagens atráves deste protocolo. Esta classe é responsável por fragmentar as mensagens e por "juntar"os fragmentos da mensagem recebido.

4.4 FS_Transfer_Protocol

Este *Package* possui apenas uma classe responsável pela estruturação do cabeçalho das mensagens do protocolo *FS Transfer Protocol* utilizado para a transferencia de ficheiros. Esta classe possui ainda os metodos para serialização e deserialização de/e para um array de bytes.

4.5 Files

Este *Package* possui classes necessárias para a criação de objetos com a informação dos ficheiros. Este pacote possui a classe *File_Info* que tem como variáveis de instância o

nome do ficheiro, o tamanho total do ficheiro, o código *Hash*, gerado pela classe *File_Hash* do mesmo pacote, o numero total de blocos do ficheiro e uma lista de objetos da classe *Block_Info*. A classe *Block_Info* possui informação dos blocos que um ficheiro possui como o tamanho em bytes do bloco, qual o seu indice(offset) e o seu código *Hash*.

4.6 DataUtilities

Por fim, o pacote *DataUtilities* possui classes com métodos estáticos como a classe *Serializer* que possui metodos para serialização de objetos do tipo *Map* e *List* utilizados como mensagens na comunicação. Este pacote possui ainda a classe *UniqueIdGenerator* que possui um método para gerar uma String com um identificador único.

5 Testes e Resultados

Para a realização dos testes e resultados, recorremos ao uso da máquina virtual do ambiente *Core* disponibilizada pela equipa docente e da topologia *Core "CC-Topo-2023-v2.imn"*.

Com o objetivo de realizar pedidos de ficheiros e responder a pedidos de ficheiros, foram criadas três pastas distintas utilizadas como pastas partilhadas dos *FS Node's*("sharedFolder/", "sharedFolder2/"e "sharedFolder3/"). A pasta "sharedFolder/"possui uma imagem com 85560 bytes de tamanho, a pasta "sharedFolder2/"possui um ficheiro de texto com e a pasta "sharedFolder3/"encontra-se vazia.

Inicialmente foi iniciado o servidor *DNS*, criado especificamente para esta topologia (com recurso ao programa *Bind9*), no *Servidor1* (10.4.4.1). Foi então iniciado uma instância do *FS Tracker* no mesmo servidor sem algum argumento(por defeito é utilizado a porta 9090).

A seguir iniciou-se três instâncias diferentes do *FS Node*, uma no host *Portatil1* (10.1.1.1) passando como argumento o caminho para a pasta partilhada "sharedFolder/"e nome do host a correr o *FS Tracker*(por omissão a porta utilizada será a 9090):

– *java FS_Node 'FS_Node/sharedFolder/' Servidor1.cc.pt*

Foi realizado um processo semelhante para os hosts PC1 (10.2.2.1) e para o host Servidor2 (10.4.4.2) onde se passou a cada um o caminho para as pastas partilhadas "sharedFolder2/"e "sharedFolder3/"respetivamente.

No host *Servidor2* solicitou-se ao *Tracker* a lista de ficheiros disponíveis na rede e iniciou-se a transferência do ficheiro de imagem partilhado pelo host *Portatil1*. Paralelamente, no host *Portatil1* solicitou-se a transferência do ficheiro de texto partilhado pelo host "Servidor2"e no host "PC1"solicitou-se a transferência do ficheiro de imagem no qual os blocos no momento são partilhados pelos hosts *"Portatil1"*(Ficheiro total) e *Servidor2*(Ficheiro Parcial). No final verificaram-se os ficheiros de "logs"dos programas, tendo-se obtido os seguintes resultados:

Figura 9. Paralelismo de envio e transferência de ficheiros

Como podemos ver na imagem anterior um *FS Node* enquanto enviava blocos de um ficheiro a outro *FS Node*, solicitou os blocos de outro ficheiro.

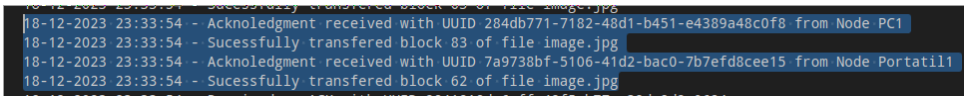

Figura 10. Receção de dois blocos do mesmo ficheiro de Nodes diferentes

Como podemos observar pela imagem anterior, ocorreu paralelismo na receção de blocos de um ficheiro, ao serem recebidos dois *ACK's* de dois *Node's* diferentes pelo *Servidor2*.

6 Conclusão e trabalho futuro

A realização deste trabalho permitiu-nos aplicar os conhecimentos adquiridos pela unidade curricular de *Comunicações por Computador* sobre os protocolos de transporte *UDP* e *TCP*, assim como adquirir conhecimentos sobre *DNS*. Foram desenvolvidas todas as funcionalidades requisitadas da aplicação para a transferência de blocos/ficheiros entre *FS Node's* e de informar o *FS Tracker* dos ficheiros disponíveis na rede. Foi adicionado ao *FS Tracker* e ao *FS Node* "interfaces" em texto para interagir com o utilizador, bem como "logs"de eventos que ocorreram durante a execução dos programas e a criação de um ficheiro de configuração que possui os dados da informação dos ficheiros da pasta partilhado, visando não quebrar a lógica de escrita e lida dos ficheiros.

Como trabalho futuro ficará a adaptação da serialização dos protocolos desenvolvidos para "texto"em vez de usar as classes de serialização de objetos do java que adiciona um overhead desnecessário de dados, visando num futuro, se necessário, conseguir adaptar o *FS Tracker* e *FS Node* para outras linguagens de programação diferentes do java. Além disso, a arquitetura do *FS Node* poderia ser um pouco revista, devida a possuir algumas dependencias desnecessárias entra as diversas classes do programa.