

Comunicações por Computador 2023/24

Engenharia Informática

TP2 – Transferência rápida e fiável de múltiplos servidores em simultâneo



Trabalho realizado por:

João Andrade Rodrigues - A100711

Mateus Lemos Martins - A100645

João Machado Gonçalves - A97321

Índice

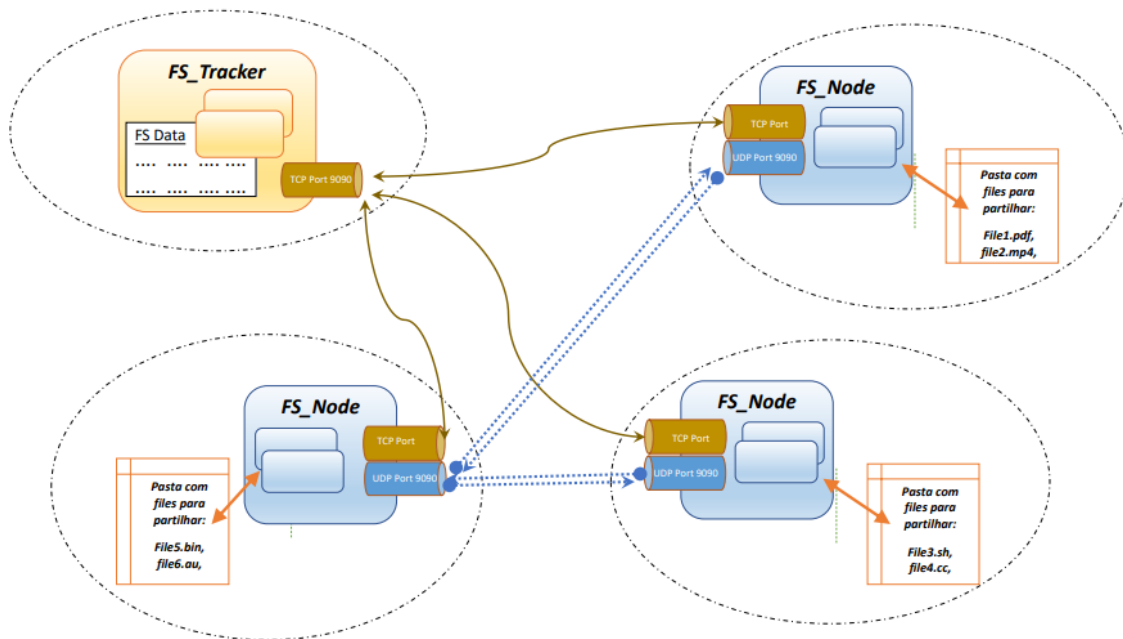
Introdução.....	3
Arquitetura da Solução.....	4
Especificação dos Protocolos Propostos.....	5
Implementação.....	8
Testes e Resultados.....	8
Conclusões e Trabalho Futuro.....	11

Introdução

A partilha de ficheiros é um dos serviços essenciais de uma rede. Permitindo que um programa cliente identifique o ficheiro e o descarregue de um outro cliente que o disponibilize. O serviço tem como requisito base que a transferência seja fiável. Os dados devem ser descarregados e guardados sem erros. A réplica descarregada pelo cliente é uma cópia integral da existente no servidor, cuja integridade pode ser verificada. Os protocolos desenhados para a transferência de ficheiros procuram garantir os requisitos base, mas também assegurar um bom desempenho na transferência. Neste trabalho pretende-se desenhar um serviço avançado de transferência de ficheiros, numa rede peer-to-peer (P2P) com um servidor, que terá a informação de todos os clientes, por este servidor passaram apenas dados, e nunca ficheiros em si. Sendo assim, estes clientes têm que dialogar entre si. Num dado instante, um ficheiro pode estar disponível em mais de um peer, e pode ser transferido de qualquer um deles, com o intuito de melhorar a disponibilidade e o desempenho. Um peer que inicie a descarga de um dado ficheiro pode de imediato disponibilizá-lo a outros peers. O desempenho global melhora substancialmente com o número de réplicas disponíveis na rede. O serviço é levemente inspirado nas redes do tipo “BitTorrent”.

Arquitetura da Solução

Como já foi levemente abordado na Introdução, iremos ter um **FS_Tracker** (servidor) que possui toda a informação necessária de um **FS_Node** (o seu ip, e o nome dos ficheiros que possui). A ligação entre o servidor e os clientes será efetuada através de um socket que suporta mensagens **TCP's**. A comunicação entre clientes será efetuada através de um socket UDP, que irá suportar um protocolo que será mais tarde visto, o **FS Transfer Protocol**.

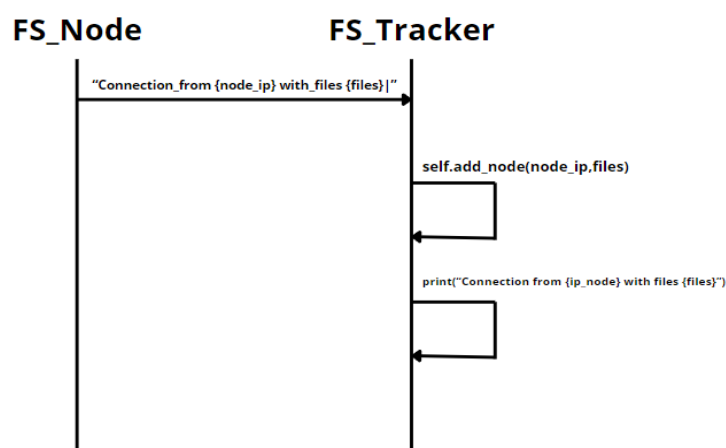


Especificação dos Protocolos Propostos

FS Track Protocol

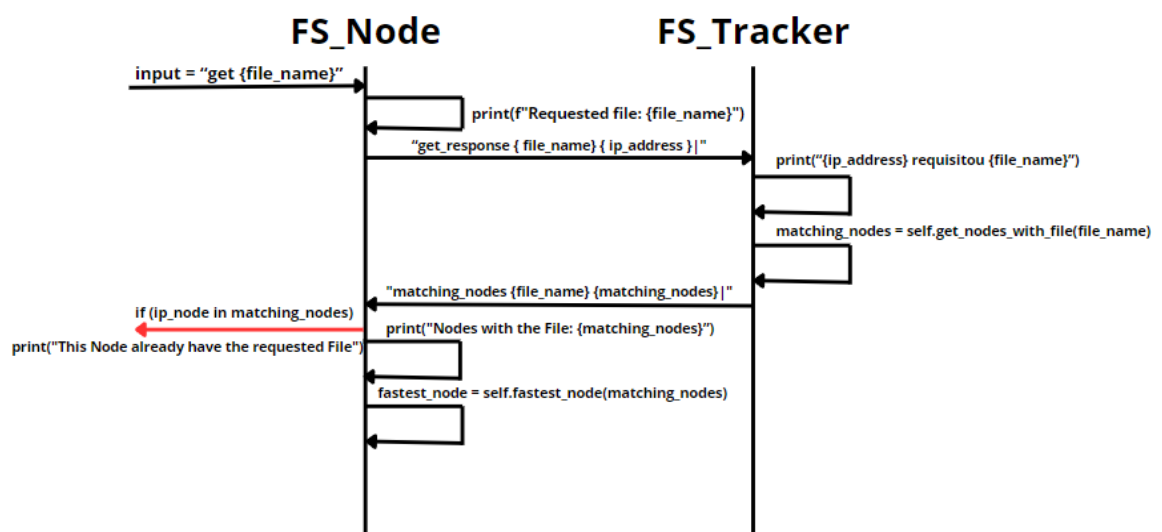
O **FS Track Protocol** suporta o registo de um **FS_Node** na base de dados do **FS_Tracker** e a localização de ficheiros através da troca de mensagens **TCP's**. Estas trocas de mensagens encontram-se representadas abaixo.

Registo de um FS_Node



O **FS_Node** que se quer conectar envia a mensagem "Connection_from {node_ip} with_files {files}|" , de seguida o **FS_Tracker** adiciona as informações sobre o node (guarda num dicionário (ip, files que possui)) e dá print a "Connection from {ip_node} with files {files}" para sabermos que houve efetivamente a conexão do node.

Aquisição um ficheiro

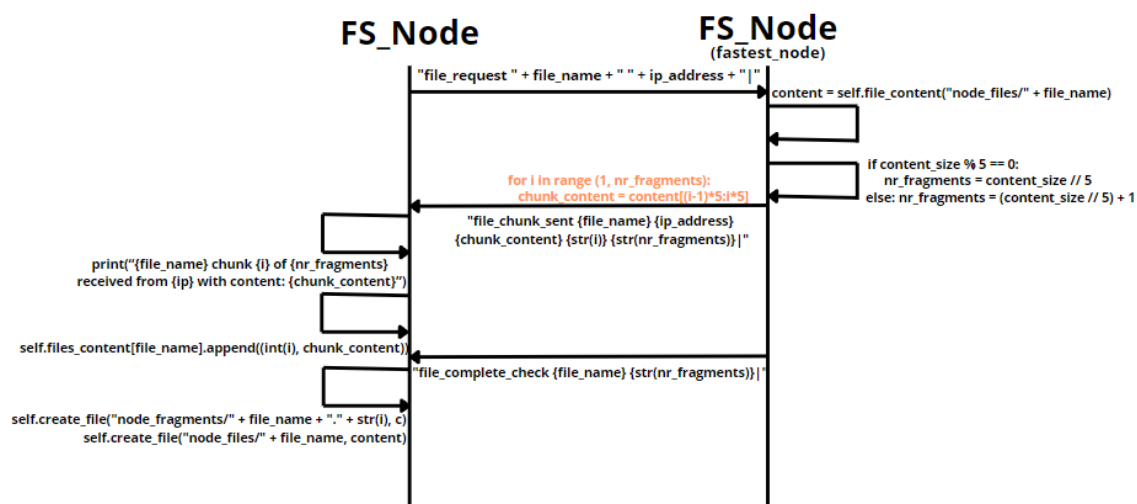


Quando o utilizador escreve no terminal do **FS_Node** “get {file_name}”, o node dá print a “Requested file: {file_name}” e envia uma mensagem ao **FS_Tracker** a dizer “get_response {file_name} {ip_adress}”. O tracker faz então um print alusivo a esta requisição “{ip_adress} requisitou {file_name}” e consultando a sua base de dados de que ficheiros cada node possui guarda na variável “matching_nodes” que nodes possuem o ficheiro requisitado, depois envia uma mensagem ao **FS_Node** com os ip’s desses “matching_nodes”. O Node analisa essa lista de ip’s e se o seu ip constar dessa lista é porque já possui esse ficheiro tornando uma possível transferência inútil, terminando assim a tarefa devolvendo “This Node already have the requested file”. Se não possuir o ficheiro, prossegue e dá print aos nodes que possuem o ficheiro e calcula qual o node mais rápido para a obtenção do mesmo, através de um processo que será demonstrado mais à frente.

FS Transfer Protocol

O **FS Track Protocol** suporta a obtenção de um ficheiro por parte de um **FS_Node** através de mensagens **UDP’s** entre **FS_Node’s**. Faz também a confirmação da receção de um ficheiro através de trocas de mensagens **TCP’s** entre o Node que requisitou o ficheiro e o **FS_Tracker** atualizando assim a base de dados do Tracker em relação a que ficheiros tem cada Node. Abaixo apresentamos diagramas temporais dos comportamentos descritos.

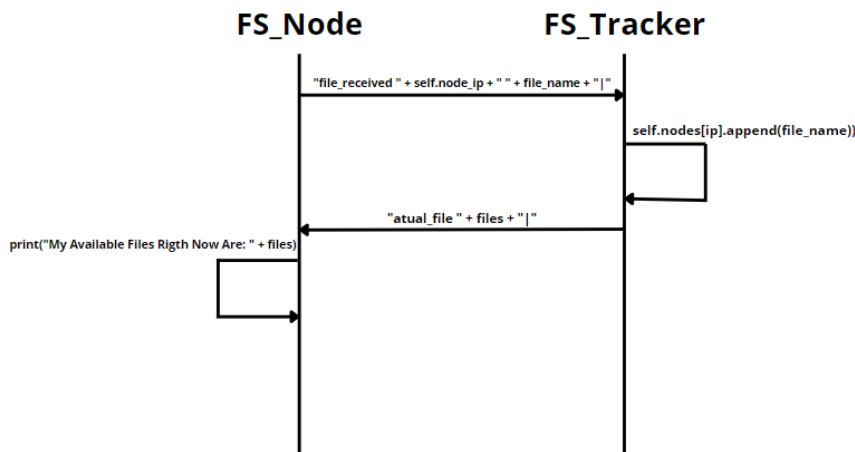
Aquisição um ficheiro



O **FS_Node** que pretende obter o ficheiro envia para o **FS_Node** (“fastest_node”, calculado anteriormente) uma mensagem a requisitar o ficheiro “file_request {file_name} {ip_adress}”, de seguida o “fastest_node” guarda na variável “content” o conteúdo do ficheiro requisitado e calcula quantos fragmentos serão precisos mandar. Em seguida envia quantas mensagens quanto o número de fragmentos a dizer “file_chunk_sent {file_name} {ip_adress} {chunk_content} {nr_fragment} {nr_fragments}”, o **FS_Node** ao receber estas mensagens dá print a “{file_name} chunk {nr_fagment} of {nr_fragments} received from {ip} with content {chunk_content}”, para assim percebermos que efetivamente está a receber um fragmento com certo conteúdo. O **FS_Node** adiciona também a um dicionário que faz corresponder um “file_name” a um tuplo (“nr_fragment”, “chunk_content”) a informação presente em cada mensagem. Esse dicionário só aceita inserção de informação caso o “nr_fragment” não esteja já presente no dicionário, não permitindo assim fragmentos repetidos. Também é recebida a mensagem “file_complete_check {file_name} {nr_fragments}” que faz o **FS_Node** ver se o dicionário dos fragmentos de um ficheiro já tem tantas

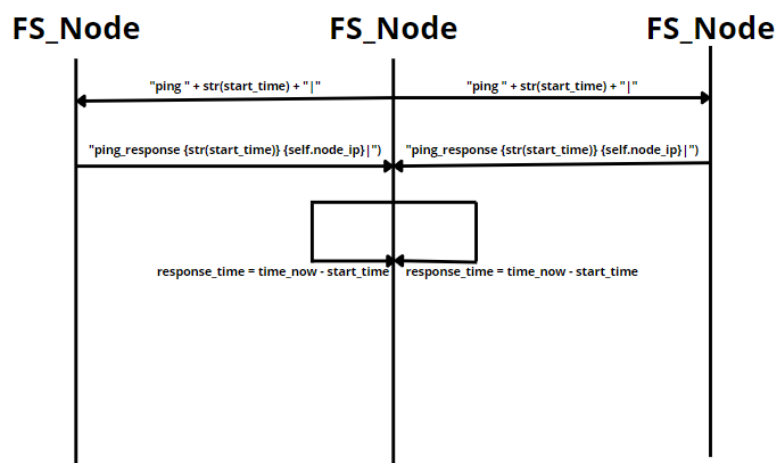
entradas como “nr_fragments”, se já tiver, significa que o ficheiro já foi recebido na sua totalidade, e então, o dicionário é ordenado e é criado o ficheiro associado à transferência (em “node_files/”) e os seus fragmentos(em “node_fragments/”).

Confirmar aquisição de ficheiro



Depois da receção de um ficheiro é necessário comunicar ao **FS_Tracker** o sucesso da operação para que o mesmo mantenha uma base de dados atualizada de que ficheiros possui cada **FS_Node**. Para isso o **FS_Node** envia uma mensagem **TCP** ao Tracker “file_received {self.node_ip} {file_name}” e o Tracker atualiza a sua base de dados. No final envia “atual_file {files}” onde “files” é a variável que representa os ficheiros atualizados desse ficheiro. O **FS_Node** ao receber esta mensagem dá print a “My Available Files Right Now Are: {files}”, o que nos ajuda a saber que a transferência foi efetuada com sucesso e base de dados relativas aquele Node atualizada.

Encontrar Node mais rápido



Para sabermos qual o **FS_Node** mais rápido para o ficheiro ser recebido é utilizado o método “fastest_node” que envia a mensagem **UDP** “ping {start_time}” a todos os nodes presentes em “matching_nodes”, é esperada uma resposta “ping_response {start_time} {node_ip}” quando recebe a mensagem calcula o tempo passado entre o envio e a resposta e quando receber a mensagem de todos os nodes vê qual deles teve o tempo de resposta menor e esse será o node ao qual deve pedir o ficheiro.

Implementação

Para a implementação do código necessário a este projeto estávamos indecisos entre “Python” e “Java”. Por um lado sabíamos que estaríamos mais à vontade a usar “java” uma vez que já tínhamos feito trabalhos com a linguagem, por outro sabíamos que “Python” traria bastantes vantagens na implementação dada a vasta oferta de bibliotecas. Com isto em mente, optamos por utilizar a linguagem de programação “Python”, visto que seria assim mais uma linguagem com a qual ficaríamos familiarizados e também porque achamos que facilitaria bastante a implementação das tarefas requisitadas.

Junto com este documento será disponibilizado o código que suporta o nosso trabalho.

Testes e Resultados

Para verificar o funcionamento do nosso trabalho, utilizamos o core emulado numa máquina virtual, usamos a topologia dada nas aulas práticas e criamos o “file1” no “Portatil1”, e o “file2” tanto no “Portatil2” como em “PC1”. Para fazermos isto temos também de iniciar o servidor “Servidor1”.

Comando para iniciar o FS_Tracker:

```
root@Servidor1:/tmp/pycore.35809/Servidor1.conf# python3 ../../../../home/core/Desktop/CC_TP/FS_Tracker.py
```

Comando para iniciar o FS_Node em “Portatil1” com o “file1”:

```
root@Portatil1:/tmp/pycore.35809/Portatil1.conf# mkdir node_files && mkdir node_fragments && echo "ajjabddb" > node_files/file1
&& python3 ../../../../home/core/Desktop/CC_TP/FS_Node.py node_files 10.1.1.1 9090
```

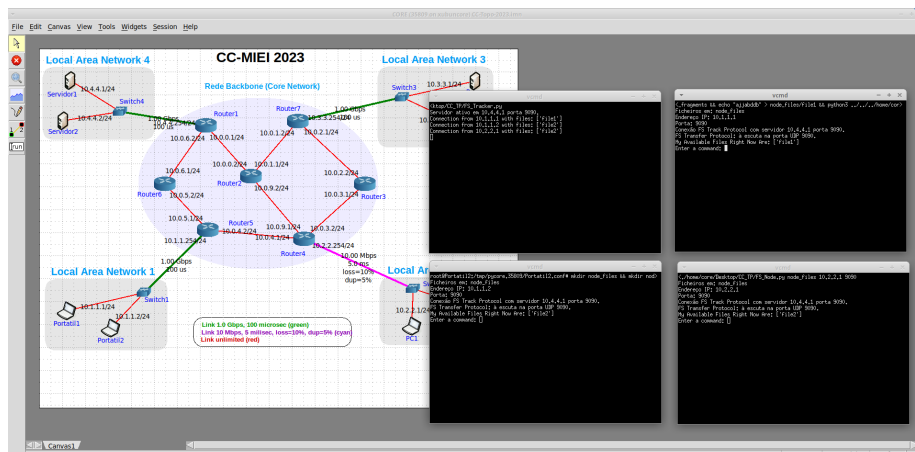
Comando para iniciar o FS_Node em “Portatil2” com o “file2”:

```
root@Portatil2:/tmp/pycore.35809/Portatil2.conf# mkdir node_files mkdir && node_fragments &&
echo "este_aqui_e_o_conteudo_do_file2" > node_files/file2 && python3 ../../../../home/core/Desktop/CC_TP/FS_Node.py node_files 10.1.1.2 9090
```

Comando para iniciar o FS_Node em “PC1” com o “file2”:

```
root@PC1:/tmp/pycore.35809/PC1.conf# mkdir node_files && mkdir node_fragments && echo "este_aqui_e_o_conteudo_do_file2" >
node_files/file2 && python3 ../../../../home/core/Desktop/CC_TP/FS_Node.py node_files 10.2.2.1 9090
```


Ao executarmos estes comandos o ambiente por nós escolhido como ambiente de teste está assim inicializado.



Escreveu-se “get file2” no terminal do Portatil1 e o resultado foi o esperado, ele recebeu os ip's dos nodes que possuíam o ficheiro, calculou o “fastest_node” e recebeu o ficheiro deste node. No terminal do Tracker aparecem também algumas mensagens relativas ao processo. Como se pode ver através do print seguinte.

```

vcmd
C:\Desktop\CC_IP\FS_Tracker.py
Servidor ativo em 10.4.4.1 porta 9090.
Connection from 10.1.1.1 with files: ['file1']
Connection from 10.1.1.2 with files: ['file2']
Connection from 10.2.2.1 with files: ['file2']
10.1.1.1 requisitou file2
file2 está presente em: 10.1.1.2;10.2.2.1
[]

vcmd
FS Transfer Protocol: à escuta na porta UDP 9090.
My Available Files Right Now Are: ['file1']
Enter a command: get file2
Requested file: file2
Nodes with the File: 10.1.1.2;10.2.2.1
The Fastest Node to Receive the File is: 10.1.1.2
file2 chunk 1 of 7 received from 10.1.1.2 with content: este_
file2 chunk 2 of 7 received from 10.1.1.2 with content: aqui_
file2 chunk 3 of 7 received from 10.1.1.2 with content: e_o_c
file2 chunk 4 of 7 received from 10.1.1.2 with content: onteu
file2 chunk 5 of 7 received from 10.1.1.2 with content: do do
file2 chunk 6 of 7 received from 10.1.1.2 with content: file
file2 chunk 7 of 7 received from 10.1.1.2 with content: 2
Ficheiro 'node_fragments/file2.1' criado com sucesso.
Ficheiro 'node_fragments/file2.2' criado com sucesso.
Ficheiro 'node_fragments/file2.3' criado com sucesso.
Ficheiro 'node_fragments/file2.4' criado com sucesso.
Ficheiro 'node_fragments/file2.5' criado com sucesso.
Ficheiro 'node_fragments/file2.6' criado com sucesso.
Ficheiro 'node_fragments/file2.7' criado com sucesso.
Ficheiro 'node_files/file2' criado com sucesso.
My Available Files Right Now Are: file1;file2
Enter a command:

vcmd
C:\Desktop\CC_IP\FS_Node.py node_files 10.1.1.2 9090
Ficheiros em: node_files
Endereço IP: 10.1.1.2
Porta: 9090
Conexão FS Track Protocol com servidor 10.4.4.1 porta 9090.
FS Transfer Protocol: à escuta na porta UDP 9090.
My Available Files Right Now Are: ['file2']
Enter a command: file2 requested from 10.1.1.1
file2 sent to 10.1.1.1
[]

vcmd
C:\Desktop\CC_IP\FS_Node.py node_files 10.2.2.1 9090
Ficheiros em: node_files
Endereço IP: 10.2.2.1
Porta: 9090
Conexão FS Track Protocol com servidor 10.4.4.1 porta 9090.
FS Transfer Protocol: à escuta na porta UDP 9090.
My Available Files Right Now Are: ['file2']
Enter a command:

```

No print seguinte é demonstrado o resultado de ser escrito “get file1” em PC1, um processo muito semelhante ao anterior. Que volta a dar o resultado esperado, onde o PC1 recebe e cria o ficheiro e os fragmentos relativos ao “file1”.

```

vcmd
kktop/CC_TP/FS_Tracker.py
Servidor ativo em 10.4.4.1 porta 9090.
Connection from 10.1.1.1 with files: ['file1']
Connection from 10.1.1.2 with files: ['file2']
Connection from 10.2.2.1 with files: ['file2']
10.1.1.1 requisitou file2
file2 está presente em: 10.1.1.2:10.2.2.1
10.2.2.1 requisitou file1
file1 está presente em: 10.1.1.1
[]

vcmd
Enter a command: get file2
Requested file: file2
Nodes with the File: 10.1.1.2:10.2.2.1
The Fastest Node to Receive the File is: 10.1.1.2
file2 chunk 1 of 7 received from 10.1.1.2 with content: este_
file2 chunk 2 of 7 received from 10.1.1.2 with content: aqui_
file2 chunk 3 of 7 received from 10.1.1.2 with content: e_o_c
file2 chunk 4 of 7 received from 10.1.1.2 with content: onteu
file2 chunk 5 of 7 received from 10.1.1.2 with content: do_do
file2 chunk 6 of 7 received from 10.1.1.2 with content: file
file2 chunk 7 of 7 received from 10.1.1.2 with content: 2
Ficheiro 'node_fragments/file2.1' criado com sucesso.
Ficheiro 'node_fragments/file2.2' criado com sucesso.
Ficheiro 'node_fragments/file2.3' criado com sucesso.
Ficheiro 'node_fragments/file2.4' criado com sucesso.
Ficheiro 'node_fragments/file2.5' criado com sucesso.
Ficheiro 'node_fragments/file2.6' criado com sucesso.
Ficheiro 'node_fragments/file2.7' criado com sucesso.
Ficheiro 'node_files/file2' criado com sucesso.
My Available Files Righ Now Are: file1;file2
Enter a command: file1 requested from 10.2.2.1
file1 sent to 10.2.2.1
[]

vcmd
C:\Desktop\CC_TP\FS_Node.py node_files 10.1.1.2 9090
Ficheiros em: node_files
Endereço IP: 10.1.1.2
Porta: 9090
Conexão FS Track Protocol com servidor 10.4.4.1 porta 9090.
FS Transfer Protocol: à escuta na porta UDP 9090.
My Available Files Right Now Are: ['file2']
Enter a command: file2 requested from 10.1.1.1
file2 sent to 10.1.1.1
[]

vcmd
C:\Desktop\CC_TP\FS_Node.py node_files 10.2.2.1 9090
Ficheiros em: node_files
Endereço IP: 10.2.2.1
Porta: 9090
Conexão FS Track Protocol com servidor 10.4.4.1 porta 9090.
FS Transfer Protocol: à escuta na porta UDP 9090.
My Available Files Right Now Are: ['file2']
Enter a command: get file1
Requested file: file1
Nodes with the File: 10.1.1.1
The Fastest Node to Receive the File is: 10.1.1.1
file1 chunk 1 of 2 received from 10.1.1.1 with content: ajob
file1 chunk 2 of 2 received from 10.1.1.1 with content: ddb
Ficheiro 'node_fragments/file1.1' criado com sucesso.
Ficheiro 'node_fragments/file1.2' criado com sucesso.
Ficheiro 'node_files/file1' criado com sucesso.
My Available Files Righ Now Are: file2;file1
Enter a command:

```

Neste próximo print é evidenciado a criação dos ficheiros em “node_files/” e a criação dos fragmentos em “node_fragments/”. Para isso utilizamos “cd” para ir para as pastas e “ls” para mostrar os ficheiros que cada pasta possui.

```

vcmd
kktop/CC_TP/FS_Tracker.py
Servidor ativo em 10.4.4.1 porta 9090.
Connection from 10.1.1.1 with files: ['file1']
Connection from 10.1.1.2 with files: ['file2']
Connection from 10.2.2.1 with files: ['file2']
10.1.1.1 requisitou file2
file2 está presente em: 10.1.1.2:10.2.2.1
10.2.2.1 requisitou file1
file1 está presente em: 10.1.1.1
10.1.1.2 requisitou file2
file2 está presente em: 10.1.1.1:10.1.1.2:10.2.2.1
[]

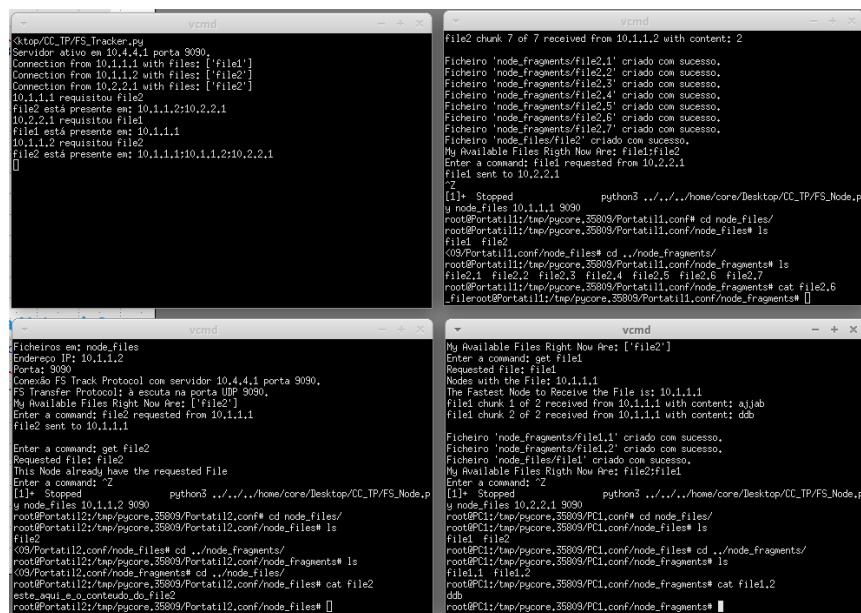
vcmd
file2 chunk 6 of 7 received from 10.1.1.2 with content: _file
file2 chunk 7 of 7 received from 10.1.1.2 with content: 2
Ficheiro 'node_fragments/file2.1' criado com sucesso.
Ficheiro 'node_fragments/file2.2' criado com sucesso.
Ficheiro 'node_fragments/file2.3' criado com sucesso.
Ficheiro 'node_fragments/file2.4' criado com sucesso.
Ficheiro 'node_fragments/file2.5' criado com sucesso.
Ficheiro 'node_fragments/file2.6' criado com sucesso.
Ficheiro 'node_fragments/file2.7' criado com sucesso.
Ficheiro 'node_files/file2' criado com sucesso.
My Available Files Righ Now Are: file1;file2
Enter a command: file1 requested from 10.2.2.1
file1 sent to 10.2.2.1
[]
[]- Stopped python3 ../.../home/core/Desktop/CC_TP/FS_Node.py
y node_files 10.1.1.1 9090
root@Portatil1:/tmp/pycore.35809/Portatil1.conf# cd node_files/
root@Portatil1:/tmp/pycore.35809/Portatil1.conf/node_files# ls
file1 file2
<09/Portatil1.conf/node_files# cd ../node_fragments/
root@Portatil1:/tmp/pycore.35809/Portatil1.conf/node_fragments# ls
file2.1 file2.2 file2.3 file2.4 file2.5 file2.6 file2.7
root@Portatil1:/tmp/pycore.35809/Portatil1.conf/node_fragments#

vcmd
C:\Desktop\CC_TP\FS_Node.py node_files 10.1.1.2 9090
Ficheiros em: node_files
Endereço IP: 10.1.1.2
Porta: 9090
Conexão FS Track Protocol com servidor 10.4.4.1 porta 9090.
FS Transfer Protocol: à escuta na porta UDP 9090.
My Available Files Right Now Are: ['file2']
Enter a command: get file2
Requested file: file2
This Node already have the requested File
Enter a command: Z
[]- Stopped python3 ../.../home/core/Desktop/CC_TP/FS_Node.py
y node_files 10.1.1.2 9090
root@Portatil1:/tmp/pycore.35809/Portatil2.conf# cd node_files/
root@Portatil1:/tmp/pycore.35809/Portatil2.conf/node_files# ls
file2
<09/Portatil2.conf/node_files# cd ../node_fragments/
root@Portatil1:/tmp/pycore.35809/Portatil2.conf/node_fragments# ls
root@Portatil1:/tmp/pycore.35809/Portatil2.conf/node_fragments#

vcmd
Conexão FS Track Protocol com servidor 10.4.4.1 porta 9090.
FS Transfer Protocol: à escuta na porta UDP 9090.
My Available Files Right Now Are: ['file2']
Enter a command: get file1
Requested file: file1
Nodes with the File: 10.1.1.1
The Fastest Node to Receive the File is: 10.1.1.1
file1 chunk 1 of 2 received from 10.1.1.1 with content: ajob
file1 chunk 2 of 2 received from 10.1.1.1 with content: ddb
Ficheiro 'node_fragments/file1.1' criado com sucesso.
Ficheiro 'node_fragments/file1.2' criado com sucesso.
Ficheiro 'node_files/file1' criado com sucesso.
My Available Files Righ Now Are: file2;file1
Enter a command: Z
[]- Stopped python3 ../.../home/core/Desktop/CC_TP/FS_Node.py
y node_files 10.2.2.1 9090
root@PC1:/tmp/pycore.35809/PC1.conf# cd node_files/
root@PC1:/tmp/pycore.35809/PC1.conf/node_files# ls
file1 file2
root@PC1:/tmp/pycore.35809/PC1.conf/node_files# cd ../node_fragments/
root@PC1:/tmp/pycore.35809/PC1.conf/node_fragments# ls
file1.1 file1.2
root@PC1:/tmp/pycore.35809/PC1.conf/node_fragments#

```

Neste print é evidenciado que de facto os ficheiros e os fragmentos têm o escritos em si o que deviam. Para isso recorremos ao comando “cat”.



The image displays three terminal windows from a VMware environment, illustrating the file transfer process. The top-left window shows a server (10.4.4.1) listening on port 9090 and receiving connections from 10.1.1.1 and 10.2.2.1. The top-right window shows the server's internal state, listing available files and fragments, and receiving a request for 'file2' from 10.2.2.1. The bottom-left window shows the server's internal state after receiving a request for 'file2' from 10.1.1.1. The bottom-right window shows the server's internal state after receiving a request for 'file2' from 10.2.2.1, listing available files and fragments, and receiving a request for 'file1' from 10.1.1.1.

```
vmcmd
~/cc/TP/FS_Tracker.py
Servidor ativo em 10.4.4.1 porta 9090.
Connection from 10.1.1.1 with Files: ['file1']
Connection from 10.1.1.2 with Files: ['file2']
Connection from 10.2.2.1 with Files: ['file2']
10.1.1.1 requisitou File2
File2 está presente em: 10.1.1.2:10.2.2.1
10.2.2.1 requisitou File1
File1 está presente em: 10.1.1.1
10.1.1.2 requisitou File2
File2 está presente em: 10.1.1.1:10.1.1.2:10.2.2.1
[]

vmcmd
File2 chunk 7 of 7 received from 10.1.1.2 with content: 2
Ficheiro 'node_fragments/file2.1' criado com sucesso.
Ficheiro 'node_fragments/file2.2' criado com sucesso.
Ficheiro 'node_fragments/file2.3' criado com sucesso.
Ficheiro 'node_fragments/file2.4' criado com sucesso.
Ficheiro 'node_fragments/file2.5' criado com sucesso.
Ficheiro 'node_fragments/file2.6' criado com sucesso.
Ficheiro 'node_fragments/file2.7' criado com sucesso.
My Available Files Right Now Are: file1:file2
Enter a command: file1 requested from 10.2.2.1
file1 sent to 10.2.2.1
[]
[1] Stopped python3 ../.../home/core/Desktop/CC_TP/FS_Node.p
y node_files 10.1.1.1 9090
root@Portatill1:/tmp/pycore.35809/Portatill1.conf# cd node_files/
root@Portatill1:/tmp/pycore.35809/Portatill1.conf/node_files# ls
file1 file2
root@Portatill1:/tmp/pycore.35809/Portatill1.conf/node_files# ls
file2.1 file2.2 file2.3 file2.4 file2.5 file2.6 file2.7
root@Portatill1:/tmp/pycore.35809/Portatill1.conf/node_files# cat file2.6
..file1root@Portatill1:/tmp/pycore.35809/Portatill1.conf/node_files# []

vmcmd
Ficheiros em: node_files
Endereço IP: 10.1.1.2
Porta: 9090
Conexão FS Track Protocol com servidor 10.4.4.1 porta 9090.
FS Transfer Protocol: a escuta na porta UDP 9090.
My Available Files Right Now Are: ['file2']
Enter a command: file2 requested from 10.1.1.1
file2 sent to 10.1.1.1
Enter a command: get file2
Requested file: file2
This Node already have the requested File
Enter a command: 'Z'
[1] Stopped python3 ../.../home/core/Desktop/CC_TP/FS_Node.p
y node_files 10.1.1.2 9090
root@Portatill2:/tmp/pycore.35809/Portatill2.conf# cd node_files/
root@Portatill2:/tmp/pycore.35809/Portatill2.conf/node_files# ls
file2
root@Portatill2:/tmp/pycore.35809/Portatill2.conf/node_files# cd ../node_fragments/
root@Portatill2:/tmp/pycore.35809/Portatill2.conf/node_fragments# ls
root@Portatill2:/tmp/pycore.35809/Portatill2.conf/node_fragments# cd ../node_files/
root@Portatill2:/tmp/pycore.35809/Portatill2.conf/node_files# cat file2
este_equi_e_o conteudo do file2
root@Portatill2:/tmp/pycore.35809/Portatill2.conf/node_files# []

vmcmd
My Available Files Right Now Are: ['file2']
Enter a command: get file1
Requested file: file1
Node with the File: 10.1.1.1
The Fastest Node to Receive the File is: 10.1.1.1
file1 chunk 1 of 2 received from 10.1.1.1 with content: ajjab
file1 chunk 2 of 2 received from 10.1.1.1 with content: ddb
Ficheiro 'node_fragments/file1.1' criado com sucesso.
Ficheiro 'node_fragments/file1.2' criado com sucesso.
Ficheiro 'node_files/file1' criado com sucesso.
My Available Files Right Now Are: file2:file1
Enter a command: 'Z'
[1] Stopped python3 ../.../home/core/Desktop/CC_TP/FS_Node.p
y node_files 10.2.2.1 9090
root@PC1:/tmp/pycore.35809/PC1.conf# cd node_files/
root@PC1:/tmp/pycore.35809/PC1.conf/node_files# ls
file1 file2
root@PC1:/tmp/pycore.35809/PC1.conf/node_files# cd ../node_fragments/
root@PC1:/tmp/pycore.35809/PC1.conf/node_fragments# ls
file1.1 file1.2
root@PC1:/tmp/pycore.35809/PC1.conf/node_fragments# cat file1.2
ddb
root@PC1:/tmp/pycore.35809/PC1.conf/node_fragments# []
```

Conclusões e trabalho futuro

Em suma, consideramos ter um trabalho bastante satisfatório. Pensamos ter conseguido o objetivo principal deste trabalho que era a implementação de uma arquitetura que suportasse o envio e receção de ficheiros. Porém temos noção que não conseguimos implementar todos os objetivos propostos para este trabalho, o que foi em muito devido à falta de tempo provocada pelas realizações de outros projetos e estudo para outras unidades curriculares.

Os pontos principais que gostaríamos de ter implementado são:

- retransmissão das mensagens caso não chegue a resposta esperada (apesar de nunca nos ter dado nenhum erro durante os testes);
- uso da linguagem DNS;
- transferência de um fragmento de um ficheiro em específico (apesar de as nossas transferências de ficheiros serem feitas através de envios de fragmentos não podemos pedir um fragmento em específico, o código não está preparado para isso. Uma implementação simples seria fazer uma triagem quando se recebe um “get file_name”, se o file_name possui-se um “.” seria então um fragmento e iríamos fazer um processo diferente de transferência.