Universidade do Minho
Escola de Engenharia

Licenciatura em Engenharia Informática

CC - Trabalho Prático Nº2 -Transferência rápida e fiável de múltiplos servidores em simultâneo

GRUPO 08

João Magalhães(A100740) Jorge Rodrigues(A101758)
Rodrigo Gomes(A100555)

Ano Letivo de 2023/2023

Universidade do Minho, Braga, Portugal

Abstract. Uma implementação de um serviço de transferência de ficheiros a pedido da Unidade Curricular de Comunicação por Computador.

Keywords: Protocolo de Rastreamento · Protocolo de Transferência · Resolução de Nomes

1 Introdução

A partilha de ficheiros é um elemento fundamental em qualquer rede, permitindo que os utilizadores descarreguem e armazenem dados de forma fiável e eficiente.Neste projeto será desenvolvido um serviço avançado de transferência de ficheiros em uma rede peer-to-peer (P2P) com múltiplos servidores. Tal, envolve a criação de dois protocolos essenciais: o FS Track Protocol, que funciona sobre TCP, garantindo uma maior segurança e confiabilidade dos dados, para a localização de conteúdos, e o FS Transfer Protocol, que opera sobre UDP para a transferência de dados mais rápida e eficiente, garantindo a recuperação de blocos perdidos e priorizando os utilizadores que oferecem as melhores condições de transferência.

2 Arquitetura da solução

O serviço desenvolvido, essencialmente, é construído com base em dois protocolos: O "FS Tracker Protocol" e o "FS Transfer Portocol". A combinação dos dois é o que proporciona a troca de ficheiros. Antes de mais, vale a pena referir que, visto que o serviço exige dois protocolos distintos, foi escolhida uma abordagem em que é feita uma separação clara de funcionalidades, onde existe um ponto de encontro, mais concretamente uma sintaxe que habilita a comunicação entre os dois programas. Posto isto, poderemos agora descrever como os servidores, cliente e "tracker" atuam e servem-se das aplicações desenvolvidas.

2.1 FS-Tracker

O "FS-Tracker" é, de maneira singela, um órgão que possui informações relativas a todos os clientes da rede e que serve os mesmos convenientemente. Um segundo papel, importante para o funcionamento da rede, é manter a integridade das informações, isto é, remover possíveis registos que não estejam atualizados. Por fim, basta perceber como é que este elemento da rede serve os outros servidores. Aquando a chegada de um pedido de transferência (detalhes sobre as mensagens podem ser encontrados nas próximas secções), o "Tracker" limita-se a enviar toda a informação que tem relativamente a um ficheiro. Estas informações consistem em, por ficheiro, saber que blocos existem e quem os tem. Este servidor apenas trabalha sobre o protocolo "FS Tracker Protocol", deixando a lógica por de trás da transferência para os clientes "FS-Node".

2.2 FS-Node

Grande parte da complexidade do projeto existe nos servidores "FS-Nodes", pois não só serve se do "FS Tracker Protocol", pois precisa das informações do "Tracker", mas também recorre ao "FS Transfer Portocol" que permite receber vários segmentos do ficheiro desejado de nodos diferentes. Posto isto, podemos descrever a estrutura de um "FS-Node" que se encontre operacional na rede. Um nodo pode ser dividido em 3 partes que operam em simultâneo.

A primeira parte funciona em segundo plano, contudo não deixa de ser importância extrema. O nome escolhido foi de "checker" que consiste numa "thread" do programa que verifica uma pasta partilhada, comunicando ao servidor os conteúdos da pasta. Isto garante ao serviço que o servidor mantém conhecimento dos ficheiros da rede.

A segunda também trabalha de fundo, contudo já não interage com o "Tracker", fazendo assim parte do "FS Transfer Protocol". Este elemento é um servidor que fica à escuta de pedidos de transferência de segmentos, o que acaba por fazer dos nodos um servidor. A pedido de um outro cliente, é enviado para o endereço e porta especificada o segmento pedido devidamente identificado.

Por fim, a parte com que o utilizador interage, serve de ponte entre protocolos e responde naturalmente aos utilizadores. Nesta parte, um utilizador pode pedir um ficheiro, que resultará em diversas interações, sendo a primeira naturalmente com o "Tracker", que devolve a lista dos ficheiros, e as próximas com outros nodos, que vão escutar o pedido e enviar os dados pedidos.

3 Especificação dos protocolos propostos

Os protocolos desenvolvidos servem para sustentar a arquitetura do serviço. No ponto anterior já foram feitas algumas referências ao seu papel dentro do sistema, contudo resta desenvolver muitos dos aspetos da especificação. Como mencionado no ponto "2", os protocolos são independentes (salvo um pequeno passo que acontece nos nodos de invocação do programa de transferência), sendo possível funcionar um sem o outro, isto é, se o protocolo "FS Tracker Protocol" fosse rescrito e, garantíssemos que os nodos recebessem na mesma as informações dos ficheiros, o "FS Transfer Protocol" funcionaria na mesma.

3.1 Formato das mensagens protocolares

Na conceção de um protocolo, um dos pontos principais é a escolha do formato das mensagens, isto porque é importante que todos os intervenientes se consigam entender, algo que é habilitado quando todos falam sobre o mesmo conjunto de regras. Assim, foram definidas para cada protocolo os seguintes formatos para as mensagens:

1. FS Tracker Protocol As mensagens neste protocolo têm um formato muito especifico que deve ser sempre respeitado.Deste modo, este são as difrentes mensagens trocadas entre nodos e "tracker":

"regn" → é uma mensagem enviada dos nodos para o "tracker" registar a sua existência na sua base de dados.O envio desta mensagem é automático com o arranque do cliente.O servidor não envia uma mensagem de resposta.

"unregn" → é uma mensagem enviada dos nodos para o "tracker" limpar a sua existência. Esta mensagem é escrita pelo utilizador para sair da rede.

"regf 'nomeDoFicheiro' 'Byte inicial' 'Byte final'" → é uma mensagem enviada dos nodos para o "tracker" com argumentos que descrevem o bloco da pasta partilhada. O servidor responde a esta mensagem com uma mensagem de "Sucesso"

"rtt 'estimativa'" → é uma mensagem enviada dos nodos para o "tracker" com um argumento correspondente a uma estimativa de o tempo que um nó demora a comunicar com o servidor.

"getf 'nomeDoFicheiro'" → é uma mensagem enviada dos nodos para o "tracker" com o nome do ficheiro desejado. O servidor responde a esta mensagem com uma lista dos nodos e os blocos que eles possuem do ficheiro

2. FS Transfer Protocol Neste protocolo, só existe uma mensagem, o que facilita bastante a sua implementação:

"sendFileTo 'nomeDoFicheiro' 'Byte inicial' 'Byte final' 'EndereçoResposta' 'PortaUsada'→ Aqui os servidores "FS-Nodes" pedem diretamente os blocos e colocam na mensagem o destino ao qual o bloco pedido deve ser enviado.

3.2 Interações

A troca de mensagens entre membros da rede gera, naturalmente, eventos que possibilitam o serviço apresentado. Estas interações variam entre simples de reconhecimento, até o envio de informações importantes. Desta forma, existem as seguintes ocorrências dentro dos nossos protocolos.

1. FS Tracker Protocol Do ponto de vista de interações, este protocolo é de longe o que possui maior variedade, como se pode inferir pelo número de tipos de mensagem. No que toca a interações, maior parte vem por parte dos clientes, o que logicamente instiga uma operação no servidor. Assim, pegando

nas mensagens obtemos as seguintes interações resultantes:

"Registo de um nó" → esta interação funciona sobre a mensagem "regn" explorada no ponto anterior. O que acontece a nível protocolar é simples: um cliente indica ao servidor que quer pertencer à rede, e consequentemente é aceite pelo servidor. Para esta ação não foi considerado relevante um reconhecimento visto que, como o protocolo é sustentado em "TCP", exige uma conexão que pode ser sentida do lado do cliente.

"Desconectar um nó" → esta interação funciona sobre a mensagem "unregn".Para cada nó que envia a mensagem "unregn" ao servidor, este remove o nó da lista de nós conectados. Do lado do cliente, este não espera nenhum reconhecimento do servidor e segue com um desligar gracioso de todos os seus componentes.

"Registar um Ficheiro"→.esta ação relaciona-se com a mensagem "regf".Em semelhança ao registo de um nó, esta ação não envolve o utilizador, é feita automaticamente pelo sistema. Do lado do servidor, é feita uma atualização na base de dados, e é enviado um reconhecimento ao utilizador.Graças a esse reconhecimento, é calculado uma estimativa da qualidade da conexão que é posteriormente enviada para o servidor com a mensagem "rtt".

"Estado da rede do nó"→ esta funcionalidade age sobre a mensagem "rtt", e envolve envio de mensagens do cliente para o servidor.Servindo-se da comutação utilizada para registar um ficheiro, o cliente faz uma estimativa do "rtt", que depois comunica com o servidor.

"Pedir Ficheiro"→ esta funcionalidade age sobre a mensagem "getf", e envolve envio de mensagens pelas duas partes. Incialmente, após a input do utilizador e o envio da mensagem, o servidor apenas envia os dados relativos ao ficheiro, o que possiblita a segunda fase para receber o ficheiro.

2. FS Transfer Protocol Apesar das interações serem restritas a só um tipo: pedir dados, que resulta no envio do bloco correspondente, este protocolo pode, em casos onde valha a pena, o protocolo possui bons resultados precisamente graças à sua simplicidade. As interações são sempre entre um recetor dos bolcos e de um ou mais servidores que enviam estes segmentos. A comutação começa com o cliente que deseja o ficheiro pedir aos outros através da mensagem

"sendFileTo". De seguida, são enviados por parte dos outros clientes os blocos pedidos.

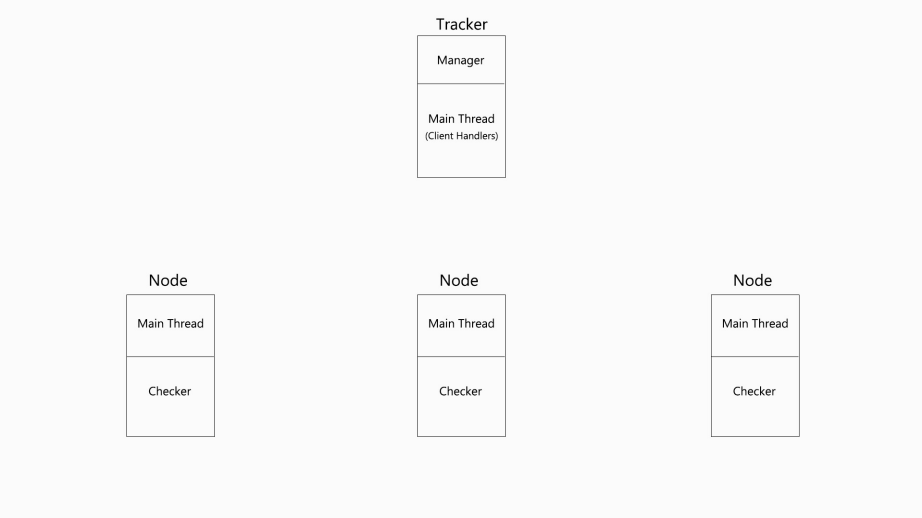

Fig. 1. Animação da implementação

4 Implementação

Com o planeamento completado, podemos proceder para a implementação do serviço na sua integra. Para esta secção vamos abordar as decisões do grupo em relação aos caminhos escolhidos em diversos tópicos, mantendo os princípios definidos nas secções anteriores.

4.1 Servidor Tracker

O servidor central "tracker" foi implementado na sua integra na linguagem "Python",assim, como toda a parte que envolve o protocolo "Fs Tracker Protocol". Tendo em conta os requisitos definidos na arquitetura no ponto anterior, foram tomadas algumas decisões que vão ser abordadas de seguida.

Base de dados: O servidor não usa um motor de Base de dados tradicional. Em vez disso, devido ao facto de nos encontrarmos em um meio académico, isto é, não existir um enorme volume de dados para gerir, decidimos usar dicionários, estruturas nativas da linguagem. Naturalmente, caso houvesse necessidade de mapear um elevado número de dados, teríamos que apostar num motor de base de dados

como o “MySql” entre outros. Esta possível implementação mais completa não iria alterar muito o código desenvolvido, tendo apenas de criar, por exemplo, um modelo relacional que acomoda as necessidades, e também fazer uma ponte entre os dois programas. Isto posto, o tracker conta com dois dicionários: O “mapaFicheiros” que usa o nome dos ficheiros como uma chave e, como pare, possui uma lista com os endereços e os respetivos segmentos que dos membros da rede que têm pelo menos um bloco do ficheiro na rede. Para este último dicionário, o servidor dispõe de uma “thread” que, periodicamente, elimina segmentos que não são renovados dentro do tempo estabelecido. Visto que, novamente, estamos a produzir um trabalho académico, os tempos estão bastante acelerados para ser mais visível o funcionamento do programa.O “conectados”, guarda o endereço e porta como chave e, como valor associado, o respetivo estado da conexão e o “rtt” atual.

Client Handler: O servidor precisa de um mecanismo que suporte as diferentes mensagens recebidas que, tipicamente, visto que chegam de um cliente, é designado por “client handler”. De modo geral, com a receção de uma nova mensagem por parte de um cliente, o servidor inicia uma “thread” que vai atender o respetivo pedido. Antes de mais, é relevante mencionar que grande parte dos pedidos vão alterar o estado da base de dados. Por isso, foram colocados mecanismo de exclusão múltipla. Deste jeito, podemos agora repescar alguns dos pontos da secção número três e explicar como foram implementadas e porquê. Começando pelas funcionalidades de registo, isto é, o “Registo de um nó”, “Registo de um ficheiro” e “Estado da rede do nó”. Estas funcionalidades são bastante semelhantes no que toca a funcionamento. Após ler os dados relevantes da mensagem, o que as “handlers” fazem nestas situações consiste em aceder aos dicionários e colocar a informação. No caso de registar um nó, a “thread” introduz no dicionário “conectados” a informação inicial com o “rtt” a “-1”. No caso de registar um ficheiro, o funcionamento é semelhante ao anterior, a diferença é que o par corresponde a uma lista, por isso apenas adicionamos a nova entrada nessa lista. Além disso criamos um “timestamp”, que é importante para manter a integridade da base de dados. Por fim, o estado da rede é registado no dicionário “conectados”, para isso basta apenas aceder ao par pelo endereço (chave no dicionário) e colocar o novo valor. A última tarefa das “handlers” é a de responder a um pedido de transferência de um ficheiro. O que acontece é simplesmente devolver a lista que existe no “mapaFicheiros” relativa ao ficheiro pedido, e de seguida colocar o valor de ”rtt” de cada Host do ficheiro.

4.2 Cliente/Servidor Node:

Os nodos, ao contrário do ”tracker”, não foram totalmente desenvolvidos em “Python”. Apenas funcionalidades que estão ligadas ao protocolo ”FS Tracker Protocol” foram desenvolvidas nesta linguagem, sendo que toda a restante lógica para acomodar o “FS Transfer Protocol” foi escrita em “C++”.

Programa Cliente: O programa cliente é, como o nome indica, a parte do nodo que é servida pelo "tracker". Nesta componente do nodo encontramos duas "threads". Primeiramente, existe uma thread "checker" que é responsável por informar o servidor dos conteúdos da pasta partilhada do nodo. Alem disso, é guardado num dicionário "MyFiles" os ficheiros para garantir que não há transferências de ficheiros repetidos e funciona em segundo plano. Passando à "thread" principal, esta serve para inicializar todas as "threads" e processos necessários para o funcionamento do nodo. Contudo, a sua função principal, é atender aos pedidos de busca de ficheiros dos utilizadores que, para esse efeito, precisa de comunicar com o "tracker" e executar uma pequena sequência que, essencialmente, consiste em, primeiramente, verificar se o ficheiro já se encontra na máquina e, caso não se encontre, executar um processo responsável pelo protocolo "FS Transfer Protocol" que será abordado num próximo ponto. Quando a transferência termina, o programa apresenta a duração da transferência ao utilizador.Ainda aqui é feito o escalonamento do ficheiro, isto é, que bloco é atribuído a qual Host (isto se for selecionado). Para isso, utilizamos o "Rtt" que é recebido do servidor como métrica para tal. O algoritmo funciona da seguinte forma: Criar um somatório com todos os valores de "Rtt" de cada Host, e de seguida atribuir uma percentagem que ocupa desse total. De seguida fazer uma média e eliminar Hosts que ultrapassem muito essa média(media * 1.85). Este processo é repetido até ninguém ser eliminado. De seguida, as percentagens são trocadas entre Hosts: o maior troca com o menor, o segundo maior com o segundo menor, etc. Com isto, o ficheiro é divido tendo em conta esta percentagem.

Remetente O "remetente" é uma das partes da codificação em C++ e é uma das duas componentes do protocolo "FS Transfer Protocol". Todos os nodos possuem um processo "remetente" sempre à escuta. Esta parte pode ser vista como um servidor bastante simples que apenas cria uma socket "UDP" e que fica à espera de pedidos para enviar ficheiro. Quando recebe um pedido, o programa envia para o destino o segmento pedido (encontram-se nos campos da mensagem como especificado anteriormente). No que toca a enviar o segmento, há uma questão muito relevante que vale a pena ser mencionada, mais concretamente, a etiqueta. Tendo em conta que estamos a trabalhar com "C++", foi decidido resolver esta questão de uma forma muito conveniente e simples. Como a unidade "char" nesta linguagem é vista como um "array" de bytes (um byte corresponde a um "char"), o programa coloca na cabeça do "array" a etiqueta, um número correspondente ao início do segmento, seguido do caracter ":", onde tudo o que esteja para a direita do caracter é dados do ficheiro. Deste modo a etiqueta vai ter sempre um tamanho de "número de dígitos do "start" do segmento", mais um, visto que usamos um byte como delimitador.

Recetor O "recetor" é a segunda parte da codificação em C++ e é uma das duas componentes do protocolo "FS Transfer Protocol". Este processo é invocado apenas se um nodo requisitar uma transferência e necessita de receber argumentos em grupos de "4" assim como a pasta onde vai guardar o ficheiro (o

ficheiro não existe na pasta garantidamente). O primeiro passo feito por este programa é dividir os blocos grandes em blocos mais pequenos, isto tendo em conta a métrica referida anteriormente de "número de digitos" e o tamanho máximo por pacote. De seguida o recetor prepara a socket "UDP" onde vai receber os ficheiros, uma "thread" que escuta desse socket e um "mapa", uma estrutura bastante conhecida que mapeia uma chave (start do segmento) a um valor (bytes do segmento). Deste modo, a tarefa de encontrar falhas e montar o ficheiro (pois não há garantias que chegam pela ordem certa) fica mais fácil. Se a transferência correr bem, o ficheiro é montado e colocado na pasta correta. Caso não seja, implementamos um mecanismo que tenta um número fixo de vezes. Caso ultrapasse esse número, damos a transferência como não completa. O número escolhido não foi muito ponderado, apenas foi uma escolha para demonstrar o código, podendo ser então otimizado.

DNS O "DNS" foi também implementado na nossa simulação. Para isso, recorremos ao "bind9" para construir uma rede nossa. No que toca a codificação, utilizamos duas zonas, sendo que uma faz o mapeamento de o nome para os endereços, e a outra faz o mapeamento ao contrário, dos endereços para o nome. Podemos ainda acrescentar as configurações que, comparativamente às pré-definidas, apenas alteramos a questão da recursão, onde não permitimos que haja na nossa rede local, pois não faria sentido já que a nossa rede é totalmente local. Se fosse aplicado a um ambiente real, poderia fazer sentido ativar a recursão dependendo das necessidades.

5 Testes e Resultados

Por fim, resta apenas aplicar a nossa implementação e perceber se, primeiramente, funciona, e também fundamentar algumas decisões que foram tomadas pelo grupo.

5.1 Testes

A implementação foi aplicada na máquina "core" disponibilizada na unidade curricular, assim como a respetiva topologia. O "DNS" ficava a correr no "Servidor1", o "Tracker" no Servidor2, e os "Nodos" nos Hosts restantes. Para cada Host foram criadas pastas onde foram colocados ficheiros de diferentes formatos, como ".zip", ".txt" entre outros. De seguida fizemos várias transferências entre os mesmos. Com estas transferências obtemos confirmação sobre o bom funcionamento da aplicação. Essencialmente o que pode ser observado foi: Os diferentes ficheiros a chegarem ao Host pretendido; O escalonamento foi igual ao esperado tendo em conta a implementação; O mecanismo de tentativas também foi bem-sucedido; A comunicação entre Nodos e "Tracker" correu também como esperado, isto é, os Nodos informavam o servidor e este era capaz de servir os Nodos. Ainda foram feitos outros testes que, desta vez, serviram para pensar no mecanismo de escalonamento. Estes foram fáceis de executar visto que o nosso

programa consegue funcionar separadamente (sem Tracker), e vão servir para a discussão nos resultados.

5.2 Resultados

Aqui teremos oportunidade de comparar aspetos arquiteturais, e discutir os mesmos resultados. Objetivo desta secção será expor as decisões que levaram o grupo à implementação atual do programa.

Mensagens Nesta componente, achamos que seria vantajoso, tendo em conta as necessidades utilizar mensagens com um formato pré-definido, implementadas de forma "hard coded" no programa. Esta decisão veio do facto de não serem precisas mensagens muito complexas e de ter vantagens do ponto de vista computacional, não ficando dependentes de outras "Apis". Também optamos por manter grande parte da comutação fora do alcance do utilizador, pois facilita a utilização do programa e dá garantias que as mensagens seguem os padrões corretos.

Perdas de blocos Relativamente às perdas, como referido anteriormente, optou-se por uma abordagem de tentativas. De modo geral, conseguimos melhores resultados com esta abordagem em comparação com outras que envolviam temporização, isto porque, deixamos de depender de estimativas, dando mais oportunidades a Hosts com problemas momentâneos.

Escalonamento de blocos A questão que mais gerou dificuldades no grupo é o escalonamento. Que métrica e implementação deve ser utilizada para conseguir bons tempos de transferência? Para esta questão, o primeiro passo foi perceber que o paralelismo é bom, e que, consequentemente, metodologias que oferecem esta condição são mais favoráveis. De seguida, fomos procurar uma métrica. Tendo em conta os recursos disponíveis, optamos por uma métrica que não precisasse de diagnósticos à rede ou qualquer tipo de informação fornecida pelo utilizador. Assim, escolhemos o tempo. Como usar esta noção de tempo gerou um novo problema, que seria como utilizar esta medida corretamente. A primeira opção que o grupo considerou seria a noção de histórico, mas para isso ser aplicável a qualquer rede, precisaríamos de um histórico entre a comunicação cada par de nós em concreto. Em redes maiores seria muito difícil de gerir, já que teríamos um pior caso fatorial. Assim optamos por uma solução que não precisasse de grande gestão, e que limitava apenas ao número de Hosts ligados (linear). Utilizando o servidor como um ponto comum a todos os nós, cada nó vai calculando o "Rtt" entre a comutação com o servidor e vai atualizando este valor no próprio servidor. Com esta opção usamos a nossa arquitetura centralizada de uma forma ótima. Por fim, vamos mostrar algumas comparações feitas durantes os nossos testes para apresentar uma ideia do que ganhamos com a nossa escolha.

Table 1. Tabela de Resultados realizados no "core" tendo em conta o "Portatil1", o "Portatil2" e o "PC1" com um ficheiro ".zip" de 56430 bytes

Escalonamento	Descrição	Resultados	Consideração
Ficheiro Inteiro	Pedido feito a um único Host da rede	Portatil1 - Média:0.01 segundos Portatil2 - Média:0.008 segundos PC1 - Média:36.730 segundos	Bom só se todos os Hosts tiverem uma boa conexão, não é ótimo
Ficheiro Partes iguais	Pedido feito a vários Hosts dividido em partes iguais	Média:15.04 segundos	Melhor que o anterior, mas requer boas conexões para obeter bons resultados, não é ótimo
Nossa Implementação	Pedido com base no algoritmo que definimos	Média:0.006 segundos	Bons resultados garantidos, não perde por maus nós na rede, não é ótimo mas é próximo

6 Conclusão

Em suma, a implementação de um serviço de transferência de ficheiros baseado em dois protocolos - FS Tracker Protocol e FS Transfer Protocol - revelou-se enriquecedora e desafiante. A arquitetura desenvolvida permitiu uma troca eficiente de ficheiros em uma rede peer-to-peer (P2P), demonstrando a importância de uma abordagem bem definida na separação de funcionalidades entre os servidores FS-Tracker e FS-Node. A decisão de utilizar mensagens com formatos pré-definidos, o encapsulamento das funcionalidades e a estratégia de escalonamento baseada no tempo trouxeram resultados notáveis durante os testes.

A análise comparativa dos métodos de escalonamento serviu para mostrar a eficiência da abordagem implementada, mostrando-se resiliente em ambientes com diferentes condições de rede. A flexibilidade do sistema, ao permitir a transferência de ficheiros mesmo com nodos com menor desempenho, demonstra a robustez da solução proposta.

Em resumo, o desenvolvimento deste serviço reforçou a importância de uma arquitetura bem planeada e de protocolos otimizados, além de evidenciar a necessidade de adaptação às condições variáveis de uma rede P2P.