1.º Trabalho Laboratorial de Sistemas Distribuídos

Autores;
Liliana Borges Vilela, 080509137
Ricardo Jorge de Sousa Teixeira, 080509040

Março 2011

Introdução

O principal objectivo deste projecto consiste na implementação de uma aplicação de partilha *peer-to-peer*, tirando partido das funcionalidades multicast IP presentes em qualquer rede local. Esta aplicação deverá ser capaz de efectuar pesquisas por ficheiros presentes nos outros *peers* ligados, e poder optar por fazer o download de um ou vários deles para o computador do utilizador.

Esta transferência de ficheiros será efectuada dividindo os ficheiro em partes (chunks) de 1024 bytes, e usando o algoritmo hash SHA-256 por defeito. Qualquer dos computadores ligados poderá estar a receber um ficheiro e a enviar outro em simultâneo, devendo ao mesmo tempo evitar o envio de informação repetida.

Além de implementar a aplicação descrita acima com sucesso, espera-se também com este projecto ficar a compreender melhor os mecanismos de transmissão de dados em redes de computadores, com especial ênfase no que diz respeito à implementação e manutenção de sistemas distribuídos. Espera-se também fomentar a evolução das técnicas de trabalho em equipa, assim como aquelas relativas a programação em Java para aplicações em rede

Este relatório encontra-se estruturado em seis partes principais. Estas consistem nesta introdução ao documento, na Arquitectura da Aplicação, onde se descreve a arquitectura e estruturas usadas no desenvolvimento deste programa, na Implementação do Protocolo, onde se explica como o protocolo foi efectivamente implementado, na secção onde são referidas as especificações adicionais acrescentadas ao projecto, no exemplo da execução da aplicação, e na conclusão deste relatório, onde são feitas reflexões sobre o projecto realizado.

Arquitectura da Aplicação

Esta aplicação foi dividida por vários módulos de modo a facilitar a sua implementação. Destes, o principal é o módulo MulticastP2P. É aqui que se encontram as funções principais que constituem o "núcleo" da aplicação. Existem também os módulos DownloadingFile, UploadingFile e Chunk, que são responsáveis pelas funcionalidades mais directamente relacionadas com o download do ficheiro, enquanto que fileStruct é usado para representar um ficheiro e os atributos apenas directamente relacionados com este. Há ainda o módulo MP2P, que contém a implementação a interface gráfica do programa.

Em MulticastP2P, podem ser encontrados diversos tipos de estruturas de dados. Um destes é a estrutura do tipo SearchResult, que armazena um resultado de uma pesquisa registando as informações obtidas relativamente ao ficheiro encontrado. Estas incluem nome, id, tamanho do ficheiro e o número de *peers* que se encontra em posse do mesmo. Cada resultado da pesquisa armazenado como objecto SearchResult é então guardado numa segunda estrutura, SearchResults, que se torna assim a responsável pelas funcionalidades de gestão da informação encontrada durante a pesquisa.

No que se refere aos ficheiros lidos do computador local, estes são armazenados num vector de fileStructs. A estrutura fileStruct representa, como o próprio nome indica, a estrutura de um ficheiro lido localmente. Nesta são registados todos os atributos que importam saber do ficheiro para esta aplicação, nomeadamente, o id do ficheiro (valor sha), o nome, tamanho, localização no disco e tamanho total de chunks ocupados por este.

De modo a facilitar a monitorização de um ficheiro cujo download está a ser efectuado em dado momento, criou-se a variável currentDownload, que remete para a estrutura implementada através do módulo DownloadingFile. Além de serem aqui registadas todas as informações disponíveis sobre o ficheiro em questão, também é guardado um vector com os chunks recebidos até ao momento, e outro que contém os números daqueles que ainda faltam obter, assim como registo de quando foi recebido o último chunk, de modo a poder ser efectuado um *timeout* após um determinado período de tempo decorrido sem resposta.

Relativamente ao download de ficheiros, foi criado igualmente o uploadingFile, de modo a guardar os ficheiros a serem transmitidos. Cada vez que é recebido um pedido por um ficheiro, é criada uma nova entrada do tipo uploadingFile, onde são armazenados o identificador do ficheiro requerido e um vector com informação sobre os chunks pedidos. Consegue-se assim facilitar a gestão da transmissão de vários ficheiros em simultâneo.

Por fim, tem-se ainda a estrutura Chunk, usada para tornar a informação sobre os chunks recebidos mais fácil de gerir. Assim, cada objecto Chunk divide os dados pelos vários atributos, sendo estes relativos ao número do chunk, ao hash, e aos dados transmitidos referentes ao conteúdo do ficheiro, sendo este último armazenado sob a forma de um array de bytes.

Implementação do Protocolo

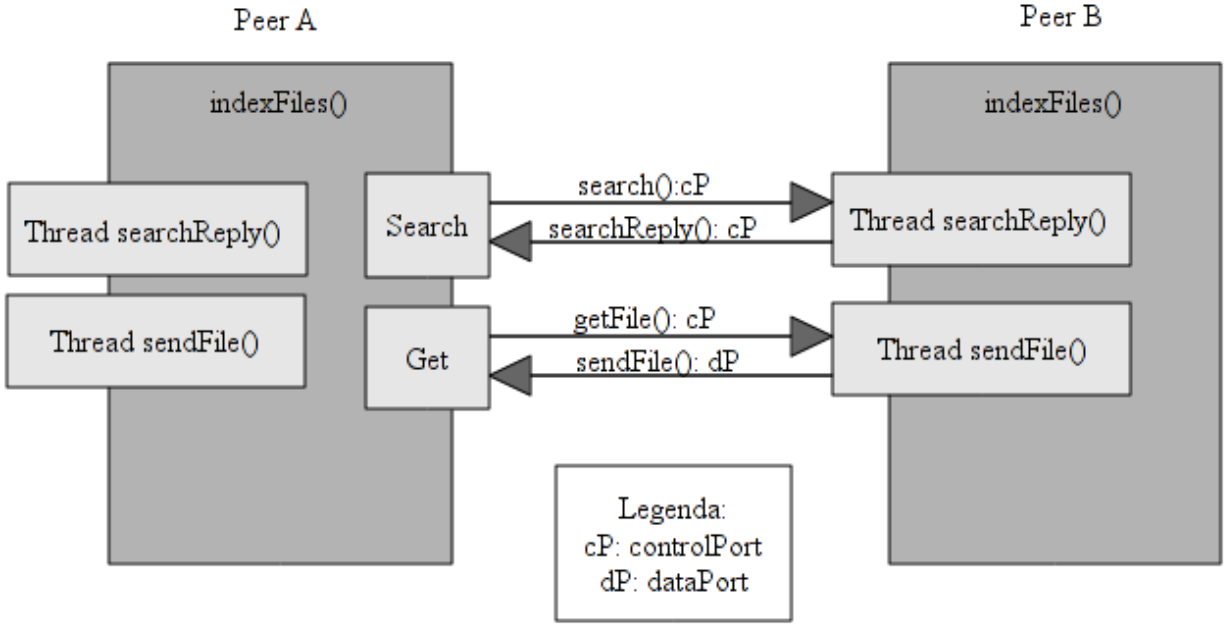

Figura 1. Esquema do funcionamento do protocolo.

Ao iniciar a aplicação, esta começa por registar os ficheiros encontrados na pasta definida para partilha como objectos FileStruct, que são então guardados num array de nome fileArray. De seguida é estabelecido contacto com o grupo multicast. Esta ligação pode ser efectuada através de duas portas, uma porta de controlo, por onde são transmitidas mensagens do tipo search, found e get, ou uma porta de dados, por onde são transmitidos apenas dados (chunks) do ficheiro.

Ambas as threads monitorizam constantemente os dados que circulam na rede na porta de controlo (dataPort). A thread searchReply() responde apenas às mensagens de pesquisa, enquanto que sendFile está à escuta de mensagens do tipo get, que traduzem pedidos de download de um ficheiro.

O peer A, ao efectuar uma pesquisa por um ficheiro, lança a função search() numa thread à parte, onde o pacote search é construído e enviado para o grupo multicast. A informação sobre a pesquisa em curso fica guardada numa variável (currentSearchID), e a thread search() fica então à espera de todas as respostas que surjam e que correspondam ao identificador da pesquisa que lançou.

Entretanto, os outros peers ligados (na figura acima exemplificados como peer B) recebem o pacote search de A. A thread searchReply(), lançada no início da aplicação, detecta o pacote em questão, e pesquisa então por um ficheiro que possua as palavras-chave pelas quais A inquiriu. Depois de passada uma verificação que assegura que um peer não responde a si mesmo, a mensagem found é construída e enviada de volta a A contendo o identificador da pesquisa à qual se responde. É retornada uma mensagem found por cada ficheiro encontrado, não havendo resposta à pesquisa no caso de nenhum conter as palavras procuradas.

Obtém-se então em A uma lista de respostas, aqui processada e guardada numa segunda variável (currentSearchResults), até que se deixem de receber respostas relevantes para a pesquisa em causa. Da lista de ficheiros encontrados apresentada ao utilizador, este pode seleccionar um e escolher efectuar o download do mesmo.

Ao iniciar um download, é usada a função getFile(). Esta envia uma mensagem get através da porta de controlo, especificando o ficheiro a obter e respectivos chunks necessários, ficando de seguida à escuta por respostas.

Em B, a thread sendFile() está à espera de receber quaisquer mensagens do tipo get. Esta vai registando esses eventos em currentUploads, uma vector do tipo UploadingFile. Ao receber o get de A, vai ser efectuada uma verificação em currentUploads para garantir que esse ficheiro não se encontra já a ser transmitido. No caso de esta ser de facto uma nova instância de um pedido, este é adicionado à variável referida e uma nova thread é criada para enviar os chunks. Caso contrário, são acrescentados à informação de currentUploads os novos chunks do ficheiro pedidos.

No que diz respeito ao envio do ficheiro propriamente dito, a função sendChunks(), lançada anteriormente, começa a enviar os chunks requisitados por A. De modo a evitar que sejam enviados pacotes repetidos, esta função cria e executa uma outra thread que fica à escuta na porta de dados pelos pacotes a serem transmitidos na rede. Sempre que detecta um que corresponda a um dos chunks pedidos originalmente e que ainda não tenha sido processado, retira-o da lista de chunks a enviar, evitando assim redundância e fluxo de dados desnecessário na rede local.

Enquanto isso, A está a processar as respostas que chegam pela porta de dados que contêm os chunks pedidos. Ainda na thread getFile(), os chunks recebidos são convertidos em objectos do tipo Chunk, de modo a poderem ser armazenados numa estrutura DownloadingFile. Esta classe contém alguns métodos referentes à verificação do download, incluindo uma função que detecta quando todos os chunks necessários foram obtidos. Uma vez esta condição verificada, o ficheiro é então escrito para o disco, e o download é terminado.

Especificações Adicionais

Além das funcionalidades requeridas, implementou-se também algumas especificações extra julgadas importantes para o melhoramento da aplicação proposta.

Uma das adições feitas diz respeito à possibilidade de passar argumentos adicionais através da linha de comandos. Estes parâmetros consistem na possibilidade de escolher a localização dos ficheiros a disponibilizar para uso na aplicação (-p <Path>), a escolha do IP a ser usado (-i <IP>), a definição da porta de controlo e daquela por onde são transmitidos os dados (-c <ControlPort>, -d <DataPort>, e ainda o tipo de hash a ser usado (-h <HashType>. Considera-se esta adição importante ao ter em conta que as necessidades dos utilizadores variam consideravelmente entre si, não devendo estes ser obrigados a adaptarem-se à aplicação, mas antes o inverso. Assim, se por algum motivo as portas predefinidas estiverem a ser usadas para outro fim, o utilizador é livre de escolher as mais adequadas à sua situação. O mesmo acontece com os outros parâmetros disponibilizados. Os parâmetros não definidos pelo utilizador tomarão o valor por defeito pré-estabelecido para a aplicação.

Outra adição implementada é a interface gráfica. Sendo necessário no contexto desta aplicação efectuar pesquisas por um ficheiro, e poder seleccionar um destes para download à medida que novos resultados vão sendo encontrados, torna-se difícil efectuar esta operação com apenas um terminal de texto. Com a implementação de uma interface gráfica foi possível dividir a informação apresentada em duas subsecções principais: uma onde são mostrados os resultados decorrentes da pesquisa feita, e uma outra onde informações relativas ao download do ficheiro pedido são disponibilizadas. Recorrendo agora ao uso de "botões", elimina-se também a necessidade de permitir escrita por parte do utilizador ao mesmo tempo que o texto relativo à execução do programa é apresentado no terminal de texto. Embora tal fosse possível de realizar, tal solução torna-se muito pouco intuitiva, sendo a interface gráfica uma melhor alternativa também neste aspecto.

Exemplo de Execução da Aplicação

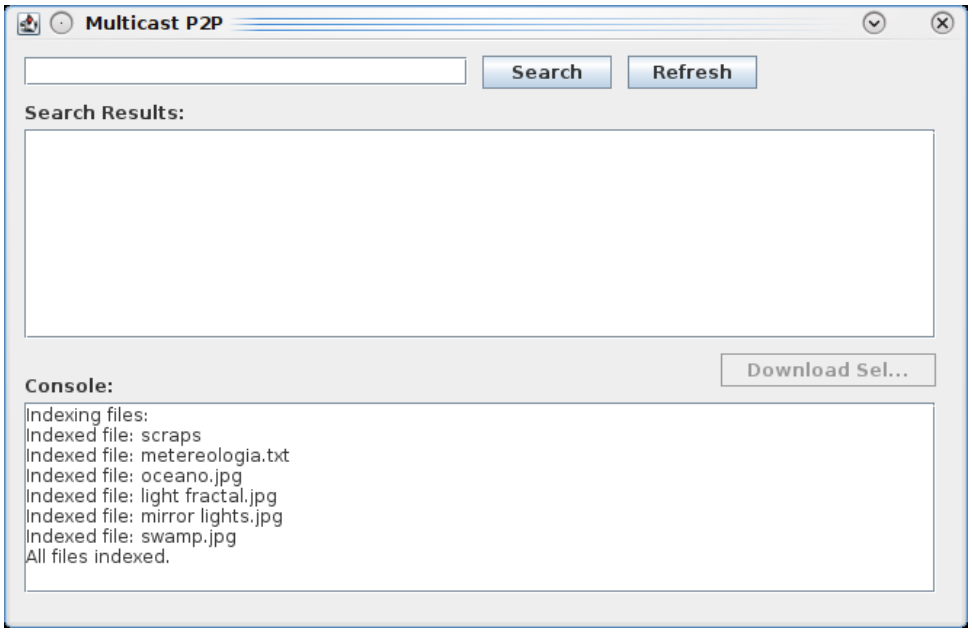

Figura 2. Instância da aplicação iniciada.

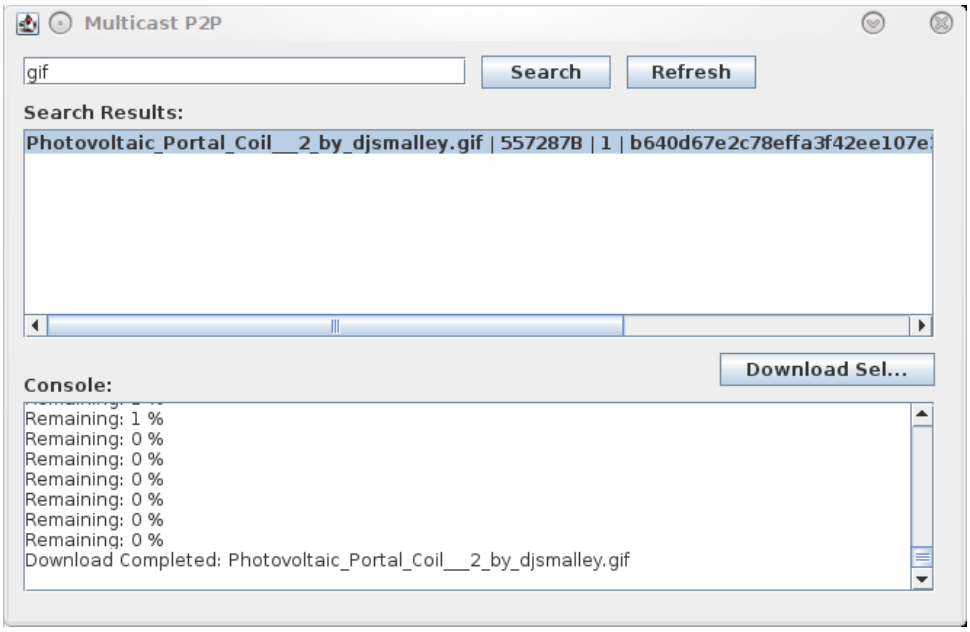

Figura 3. Exemplo de execução do download de um ficheiro.

Conclusão

Considera-se que se conseguiu atingir os objectivos propostos para este projecto, conseguindo-se implementar com sucesso os mecanismos de pesquisa e download de ficheiros de vários tamanhos de forma fiável através de uma rede local, fazendo para tal uso do protocolo multicast IP.

Para além das funcionalidades propostas, foi também conseguida a implementação de algumas funcionalidades extra, sobretudo relacionadas com a interacção da aplicação com o utilizador. Assim, este pode interagir com a aplicação através de uma interface gráfica mais intuitiva, e ter a opção de escolher alguns dos parâmetros usados no programa de acordo com aquilo que lhe for mais conveniente. Uma possível melhoria que se pensou ainda em fazer no futuro relaciona-se com a escolha dos parâmetros do programa através da própria interface gráfica, assim como a implementação de mecanismos extra para verificação da fiabilidade dos dados transmitidos.

Apesar de terem surgido várias dificuldades, sobretudo relacionadas com o manuseamento de dados de ficheiros e a execução de threads concorrentes, considera-se que estas foram ultrapassadas com sucesso, tendo contribuindo no final para o crescimento dos membros da equipa, quer em termos de compreensão dos mecanismos de comunicação em rede, quer no que se refere ao conhecimento adquirido da linguagem Java.

Conclui-se assim que este projecto foi fundamental para a melhor compreensão do funcionamento de sistemas distribuídos, área cada vez mais em destaque hoje em dia devido ao papel central que tem vindo a desempenhar em várias áreas, desde os jogos de computador e aplicações *peer-to-peer*, até à aplicação em redes de telecomunicações e investigação científica. Conseguiu-se também ganhar alguma familiarização com a programação desses mesmo sistemas, tendo sendo assim possível apreender de forma mais eficaz os conceitos inerentes à implementação de aplicações de rede, com especial destaque para o protocolo multicast e para o funcionamento deste.