Neste projeto pretende-se desenvolver um sistema de descarga de ficheiros binários (neste caso imagens) distribuído. Devem existir vários utilizadores na rede P2P e todos podem fazer pedidos de ficheiros aos outros utilizadores ou responder aos pedidos recebidos enviando partes (blocos) dos ficheiros pretendidos. O foco do trabalho é na aplicação de programação concorrente e distribuída, sendo que a parte gráfica das aplicações dos utilizadores deverá ser considerado um aspeto secundário.

Arquitetura

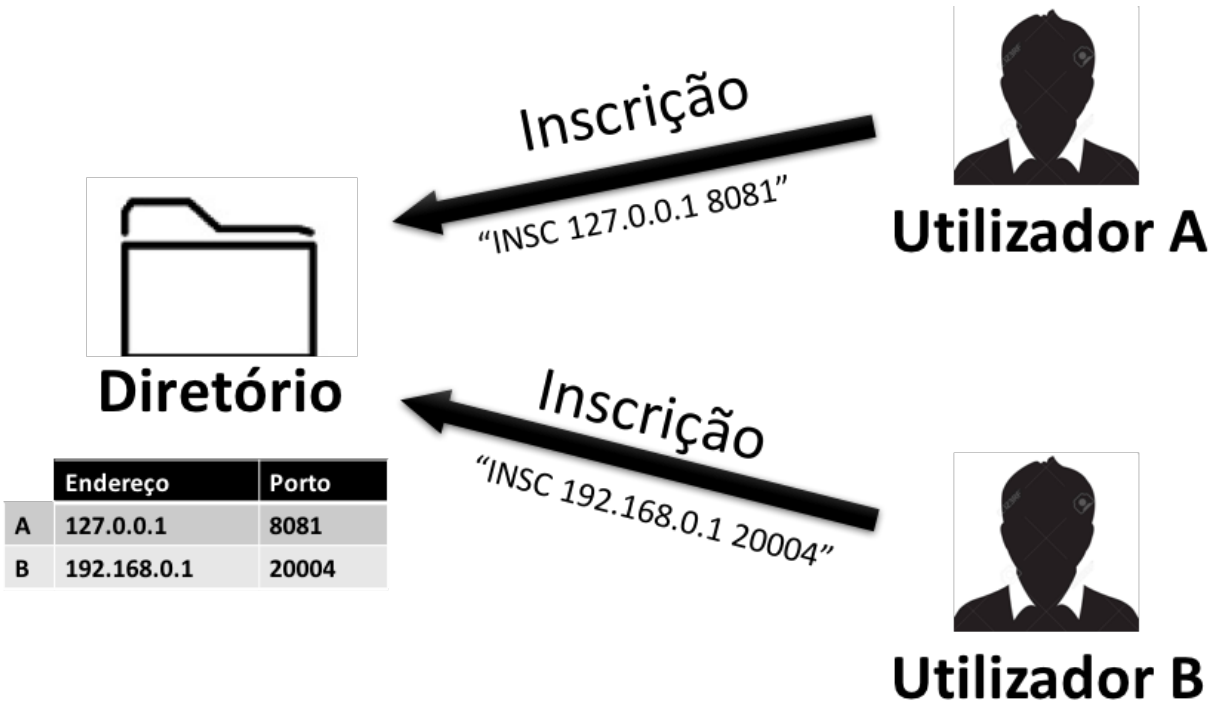
Em termos gerais, a solução deve apresentar uma arquitetura P2P (Peer-to-peer) onde as aplicações dos utilizadores trocam mensagens directamente entre si sempre que pretendem saber quem tem determinados ficheiros ou pretende pedir um ficheiro. A solução a implementar não deve recorrer a um servidor central que garante a ligação entre as aplicações dos diferentes utilizadores. Deve no entanto implementar um servidor que faz o papel de directório, onde os utilizadores devem registar a sua entrada ou saída da rede e poder perguntar a qualquer momento quais os outros utilizadores que se encontram na rede.

As Figuras 1 a 4 ilustram a arquitetura do sistema tendo em conta os principais intervenientes e as funcionalidades previstas.

Diretório

Deverá existir uma aplicação *diretório* na qual os diversos utilizadores se inscrevem logo que arrancam. Este diretório funcionará segundo um modelo cliente-servidor. A Figura 1 ilustra um cenário de funcionamento com utilizadores *A* e *B*, que se inscrevem junto do diretório. As mensagens trocadas com o *diretório* serão sempre textuais. O utilizador deve, para se

inscrever, enviar uma mensagem com o conteúdo `INSC <NOME> <ENDEREÇO> <PORTO>`. Após recebida esta mensagem de inscrição, o utilizador passa a constar da base de dados do diretório.



	Endereço	Porto
A	127.0.0.1	8081
B	192.168.0.1	20004

Figura 1. Inscrição dos utilizadores no diretório.

Aplicações Utilizadores

Os utilizadores podem fazer duas ações diferentes, interagindo com os seus pares: a primeira é fazer pesquisas por palavras chave, que serão enviadas a todos os utilizadores conhecidos, representado na Figura 3. A segunda é pedir a descarga de um ficheiro que exista em pelo menos um utilizador encontrado na pesquisa, representado na Figura 4.

Quando o *Utilizador A* pretender descarregar um ficheiro, terá de começar por saber quem está a disponibilizar esse ficheiro. Para isso deve pedir aos outros utilizadores a lista dos ficheiros que estão a disponibilizar que possuem uma determinada palavra chave.

No entanto para se poder ligar aos restantes utilizadores da rede P2P deve contactar o *diretório*, perguntando-lhe quais os utilizadores actualmente inscritos. Tal é feito através do envio de uma *Consulta utilizadores*, com o formato `CLT`. A resposta *Lista utilizadores* será devolvida no formato `CLT <ENDEREÇO> <PORTO>`, em tantas mensagens quanto o número de utilizadores. Para sinalizar a inexistência de mais utilizadores inscritos, será enviada uma mensagem `END`. Esta interação entre o utilizador e o directório está representada esquematicamente na Figura 2.

Já na posse dos endereços dos utilizadores em funcionamento, será feita uma *Procura* junto dos utilizadores existentes. No exemplo representado na Figura 3 apenas os designados *B* e *C*. A procura deve ser feita através de um canal de objetos, usando uma classe específica com este fim, designada *WordSearchMessage*. A resposta deve incluir uma lista com os identificadores dos ficheiros que correspondem à palavra procurada. O utilizador *B*, não dispondo de nenhum ficheiro correspondente à palavra chave, responderá com uma lista vazia. Já o utilizador *C*, dispondo de ficheiros correspondentes à palavra chave procurada, enviará uma mensagens com a lista dos dados dos ficheiros em questão, representados na classe *FileDetails*, que especifica o nome e tamanho do ficheiro. Note-se que um utilizador apenas deve assinalar que pode fornecer um ficheiro se o detiver na sua totalidade: se apenas tiver partes deste por ainda estar a descarregá-lo, não deve responder positivamente a uma procura.

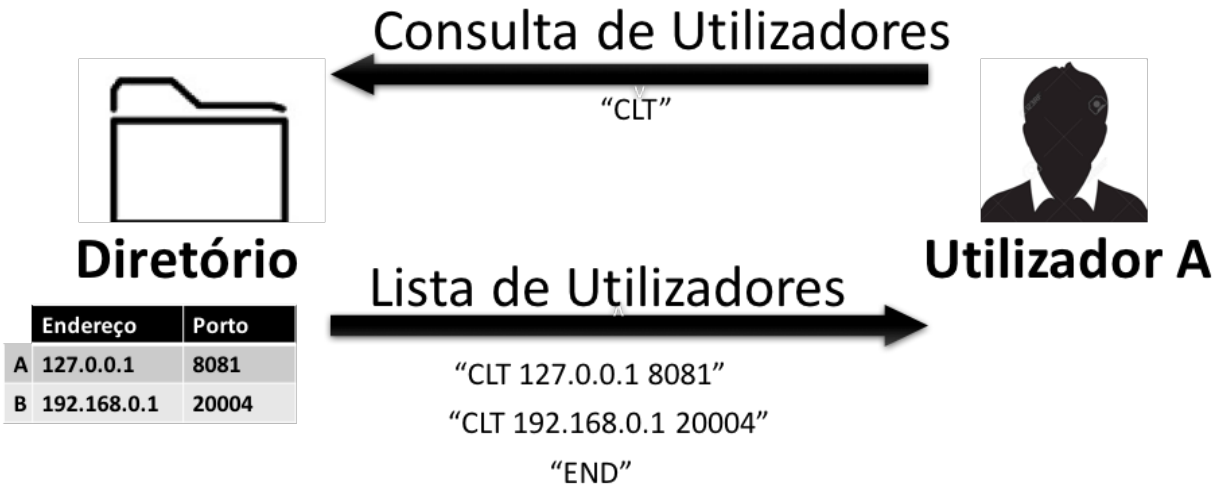


Figura 2. Consulta ao diretório dos utilizadores ligados num determinado momento.

Finalmente, proceder-se-á à descarga propriamente dita, ver Figura 4. Esta deve ser feita em partes, para não sobrecarregar os outros utilizadores e para poder descarregar a partir de vários utilizadores. O tamanho de cada parte a descarregar deve ser definida globalmente através de uma constante, e sugere-se que tenha o valor de 1024 bytes, sujeito a alteração. As partes devem ser sucessivamente pedidas aos diferentes utilizadores que detenham o ficheiro através de um *Pedido parte*, que será feito através de uma mensagem do tipo *FileBlockRequestMessage*, que deverá identificar o ficheiro, o "offset" - índice do byte onde começar; e "length" - número de bytes a ler. Tem de se ter atenção ao facto de a última parte a descarregar poder ter um tamanho mais pequeno.
Antes de se iniciar a descarga deve ser criada uma lista com todos os blocos a serem descarregados. Um novo pedido deve apenas ser enviado a um utilizador quando este terminar o envio do ou bloco anterior. Assim garante-se que os utilizadores mais eficientes vão receber um maior número de pedidos. Esta descarga deve ser feita por várias *threads*, uma para cada utilizador disponível para fornecer o ficheiro.

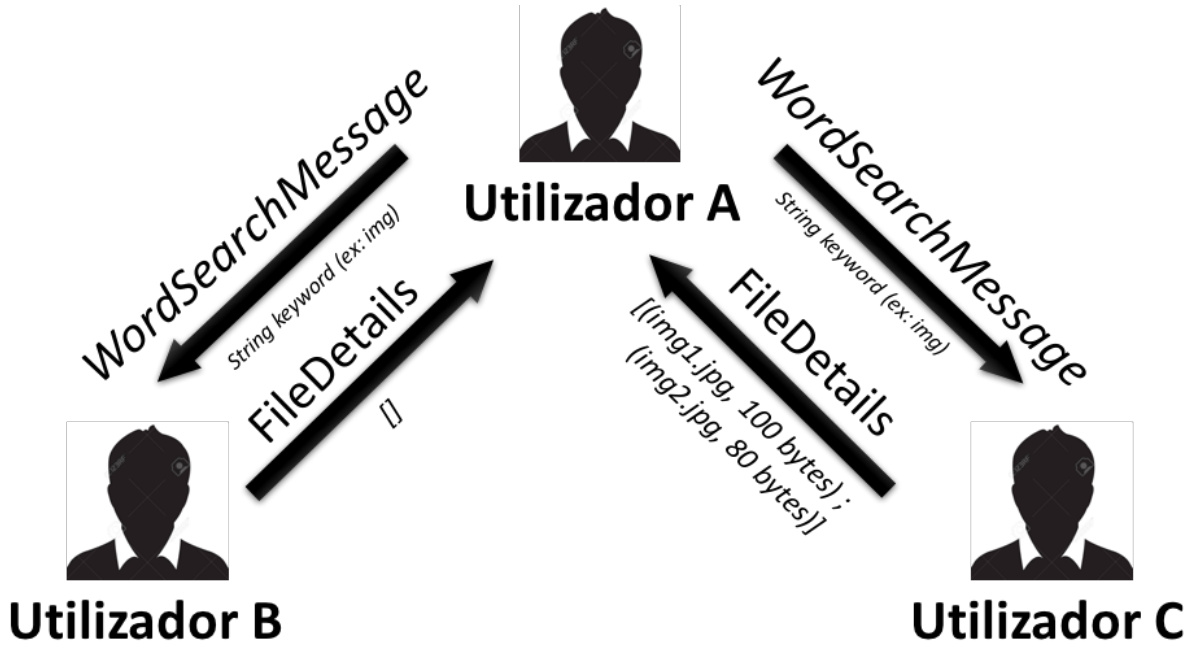


Figura 3. Consulta ao diretório dos utilizadores ligados num determinado momento.

Figura 4. Ações entre utilizadores, após consulta do diretório.

Detalhes de execução

Diretório

O diretório deverá aceitar pedidos de inscrição por parte de utilizador, através de uma ligação TCP/IP num porto com um número bem definido (e do conhecimento das aplicações dos utilizadores). Ao ser estabelecida uma ligação, a aplicação do utilizador envia a mensagem de inscrição. A aplicação do utilizador podem posteriormente enviar pedidos de consulta de todos os utilizadores inscritos. Estas mensagens terão o formato acima indicado.
O porto em que o diretório vai funcionar deve ser definido como o único argumento de execução (através do argumento String[] do *main*).

Aplicações utilizadores

A aplicação do utilizador (*The ISCTE Bay*) deverá poder ser lançado num processo Java normal, por exemplo executando:

```
java TheISCTEBay 127.0.0.1 8080 8081 dl1
```

Os argumentos de execução representam, por ordem, o endereço e porto do diretório, o porto em que este utilizador vai receber pedidos de outros utilizadores e o nome da pasta onde se encontram, os ficheiros partilhados, bem como onde serão guardados os ficheiros descarregados. Para simplificar, sugere-se que esta pasta seja criada dentro do próprio projeto *Eclipse*, sem ser dentro da pasta *src*.
Quando um ficheiro estiver totalmente descarregado, deve ser indicado quantas partes foram descarregadas de cada outro utilizador, bem como o tempo consumido, conforme exemplo na Figura 5. A escrita do ficheiro em disco deve ser efectuado por uma thread específica que deve esperar (ficar em *wait*) até que todos as partes do ficheiro estiverem descarregadas.

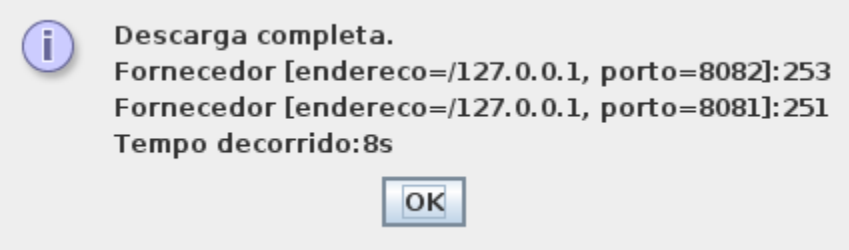


Figura 5: diálogo a assinalar o fim de uma descarga.

Para não sobrecarregar as aplicações dos utilizadores com pedidos, sempre que um utilizador recebe um pedido de uma parte de ficheiro este deve ser colocado numa lista de tarefas. As tarefas da lista devem ser tratadas por um número pré-definido de threads (por exemplo 5). Assim garantimos que se existirem por exemplo 20 utilizadores a pedirem ficheiros, a aplicação vai responder no máximo a 5 de cada vez, isto é, serão enviados no máximo 5 ficheiros simultaneamente. Deve implementar todas as estruturas necessárias para coordenar estas threads trabalhadoras bem como as threads que recebem os pedidos.

Ligações

Neste projecto a aplicação do utilizador vai desempenhar vários papéis:

- Como cliente na relação com o directório quer para se registar como para perguntar quais os outros utilizadores actualmente disponíveis;
- Como servidor para receber ligações de outros utilizadores que lhe podem pedir quais os ficheiros que tem com uma determinada palavra ou para lhe pedir partes de um determinado ficheiro;
- Como cliente para ligar a outros utilizadores para fazer os pedidos anteriormente identificados.

Todos os intervenientes neste sistema deverão funcionar em *multi-threading*, assegurando a máxima disponibilidade para receber novas ligações, bem como reagir a pedidos das ligações já existentes.

Interface gráfica (IG)

Deve criar uma IG à imagem do exibido na Figura 6, no topo existe uma caixa de texto para inserir o texto a procurar e um botão ("*Procurar*") para lançar essa procura. Em baixo, serão exibidos, à esquerda, os resultados da procura, numa `JList`. Do lado direito haverá o botão para lançar a descarga do ficheiro selecionado à esquerda, e, por baixo deste, uma `JProgressBar` para indicar o progresso da descarga. As classes `JList` e `JProgressBar` serão abordados no turno prático da semana de publicação do enunciado. Esta IG não precisa de exibir o conteúdo das imagens.

Figura 6: Aspecto geral da janela principal da Interface Gráfica.

Fases, Avaliação, e Entrega

São propostas as seguintes fases de desenvolvimento como metas para a avaliação, e de forma a que seja mais fácil abordar o problema:

Fase 1: Desenvolver uma versão básica da IG, procedendo à inscrição dos utilizadores no diretório e procedendo à consulta simples dos utilizadores inscritos.

Fase 2: Desenvolver a aplicação do utilizador para efetuarem a procura de ficheiros por palavra chave, e a respetiva exibição na IG.

Fase 3: Desenvolver as funções de descarga de ficheiros.

Fase 4: Evoluir a solução para lidar com utilizador que deixem de responder a pedidos ou demorem demasiado a fazê-lo.

Devem ser usados mecanismos de coordenação que garantam o correto funcionamento de todas as aplicações. Implemente todos estes mecanismos que necessitar, como por exemplo listas bloqueantes, barreiras ou semáforos.

As notas do projeto serão atribuídas de acordo com a realização das fases propostas:

A: completar todas as fases

B: completar as fases (1, 2, 3) de uma forma completa

C: completar as fases (1, 2, 3) com algumas falhas que não comprometam, porém, o funcionamento geral do sistema.

D: não são cumpridos os requisitos mínimos (reprovação à UC)

Grupos: Cada grupo de trabalho é composto por dois alunos, preferencialmente da mesma turma prática.

Entrega Intercalar: Para a entrega intercalar é necessário terminar a fase 1. A demonstração será feita na aula prática a partir de um JAR.

Entrega Final: O projeto desenvolvido deve ser entregue sob a forma de um projeto arquivado usando a funcionalidade de *Export/Archive File* do Eclipse. As entregas serão feitas no e-learning até às 12h de dia 13 de Dezembro. Os grupos deverão comparecer na última semana à aula prática onde estão inscritos para realizarem a discussão do trabalho.

Dicas

Os ficheiros podem ser lidos de um diretório da seguinte forma:

```
File[] files = new File(path).listFiles();
```

Para simplificar as tarefas de leitura e escrita de ficheiros, vamos considerar que estes são lidos de e escritos a partir de `arrays` de `byte`. Tal impede a utilização de ficheiros muito grandes, devido à limitação de tamanho dos arrays, mas no presente caso tal não tem importância em termos de conceitos de PCD.
Para fazer a leitura de um ficheiro, pode utilizar-se o seguinte código, que faz uso das classes `Files` e `Paths` do pacote `java.nio.file`:

```
byte[] fileContents= Files.readAllBytes(files[0].toPath());
```

Já para fazer a escrita do `array bytes`, pode usar-se o seguinte:

```
Files.write(Paths.get(pasta+"/"+nome_ficheiro), bytes);
```