

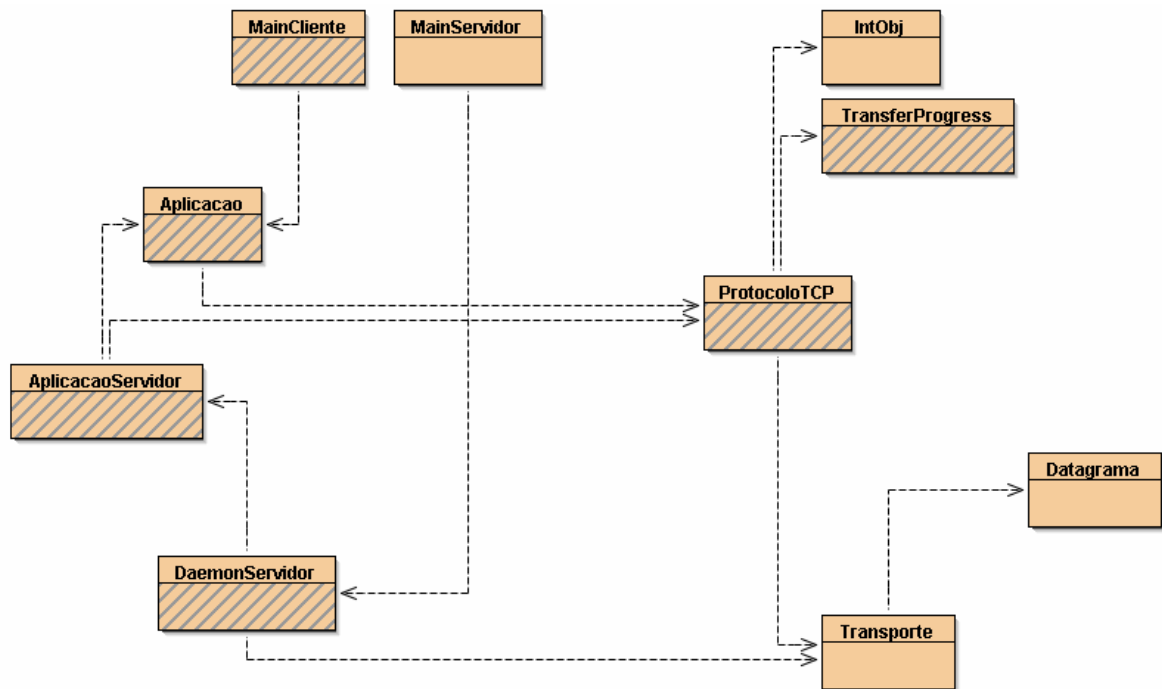
COMUNICAÇÕES POR COMPUTADOR EI 2007/2008  
Mini Pilha Protocolar para Partilha de Ficheiros (MP3F)

Fase 3  
Camada 2 e Aplicação

**Grupo 30**  
Tiago Mendonça Coutinho Correia nº47108  
Miguel dos Santos Esteves nº43165  
Nuno Miguel Mendonça Coutinho Correia nº43179

2008-04-17

## Diagrama de Classe



### Classe Datagrama:

A camada 1 é tratada aqui. Nomeadamente todos os encargos de envio e recepção, controlo de erros e simulação de erros de transmissão. Alberga métodos *sendDatagrama* que são usados em camadas acima para enviar ao mais baixo nível um pacote de bytes.

### Classe Transporte:

Os campos de um pacote de TCP são aqui alocados num array de bytes. Usam-se ainda métodos *sendTransporte* onde se enviam pacotes de camada 2 usando *sendDatagrama* da camada inferior.

### Classe ProtocoloTCP:

Aqui é feito o controlo de fluxo e congestão de toda a camada 2. Métodos como *enviarBytesSEND* controlam o mecanismo de envio de pacotes de transporte e *enviarBytesRECEIVE* controlam o mecanismo de recepção de pacotes com o respectivo envio das confirmações. Nesta camada é implementado o sistema de retransmissão *Go-Back-N* onde o terminal que está a enviar, se não receber uma confirmação, ou se receber uma atrasada, vai voltar a transmitir todos os fragmentos a partir da confirmação recebida. Existe um mecanismo de *sliding window* onde o tamanho de janela aumenta ou diminui conforme a recepção de confirmações positivas. O *Round Trip Time* é decrementado a cada retransmissão do mesmo pacote e reiniciada se for enviado outro diferente. Está condicionado com as confirmações que recebe.

### Classe DaemonServidor:

Esta classe faz a gestão de um *daemon* que está sempre à escuta de pedidos de conexão de clientes. Sempre que recebe um vai encaminhar esse pedido para a *AplicacaoServidor* e continua à escuta de novos pedidos de conexão.

**Classe AplicacaoServidor:**

Aqui é tratada a aplicação do servidor. Vai receber comandos do cliente e enviar respostas. Existe uma directoria raiz que para manusear com os ficheiros nela existentes usa a métodos da classe *Aplicacao*.

**Classe Aplicacao:**

Esta classe, além de possuir os métodos para manuseamento em pastas, também faz a gestão de uma pequena linha de comandos acessível apenas na aplicação do cliente.

## Aplicação

### *Servidor*

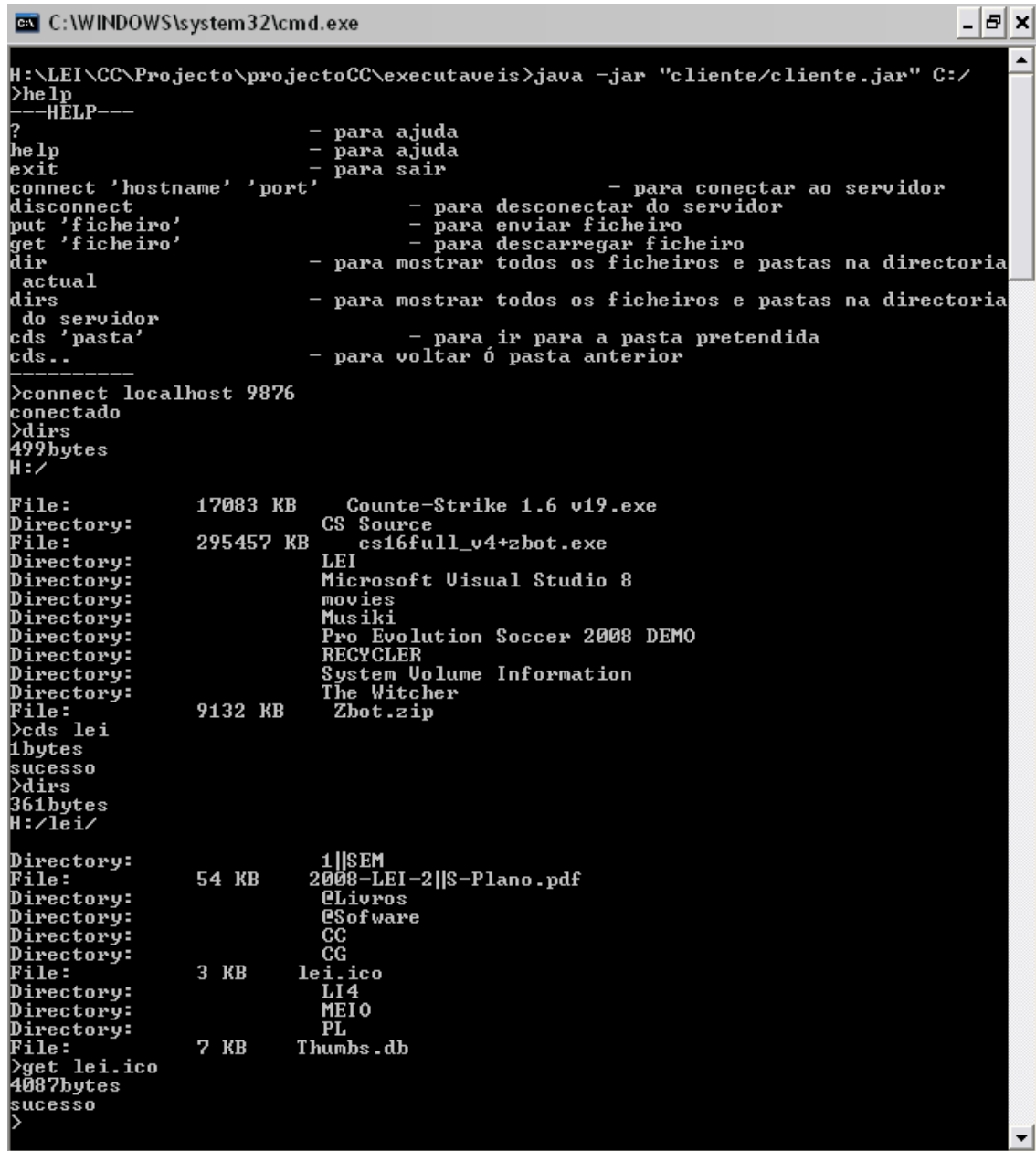
Para executar a aplicação do servidor é preciso ir à linha de comandos do sistema operativo e executar o comando *java -jar "ficheiro.jar" 9876 H:/* O último argumento refere-se à directoria de raiz onde queremos que a aplicação comece. O número 9876 refere-se à porta onde queremos por o servidor à escuta.

Como a aplicação do servidor não é interactiva, não precisa de mais instruções.

## Cliente

Para executar este programa é preciso ir à linha de comandos do sistema operativo e executar o comando `java -jar "ficheiro.jar" C:/`. O último argumento refere-se à directoria de raiz onde queremos que a aplicação comece.

Considere-se a seguinte sequência de acções na aplicação do cliente.



```
C:\WINDOWS\system32\cmd.exe
H:\LEI\CC\Projecto\projectoCC\executaveis>java -jar "cliente/cliente.jar" C:/
>help
---HELP---
?                - para ajuda
help             - para ajuda
exit             - para sair
connect 'hostname' 'port' - para conectar ao servidor
disconnect       - para desconectar do servidor
put 'ficheiro'   - para enviar ficheiro
get 'ficheiro'   - para descarregar ficheiro
dir             - para mostrar todos os ficheiros e pastas na directoria
actual
dirs            - para mostrar todos os ficheiros e pastas na directoria
do servidor
cds 'pasta'      - para ir para a pasta pretendida
cds..           - para voltar ó pasta anterior
-----
>connect localhost 9876
conectado
>dirs
499bytes
H:/
File:           17083 KB    Counte-Strike 1.6 v19.exe
Directory:      CS Source
File:           295457 KB   cs16full_v4+zbot.exe
Directory:      LEI
Directory:      Microsoft Visual Studio 8
Directory:      movies
Directory:      Musiki
Directory:      Pro Evolution Soccer 2008 DEMO
Directory:      RECYCLER
Directory:      System Volume Information
Directory:      The Witcher
File:           9132 KB    Zbot.zip
>cds lei
1bytes
sucesso
>dirs
361bytes
H:/lei/
Directory:      1||SEM
File:           54 KB     2008-LEI-2||S-Plano.pdf
Directory:      @Livros
Directory:      @Software
Directory:      CC
Directory:      CG
File:           3 KB     lei.ico
Directory:      LI4
Directory:      MEIO
Directory:      PL
File:           7 KB     Thumbs.db
>get lei.ico
4087bytes
sucesso
>
```

Com o comando `help` pode-se visualizar todos os comandos reconhecíveis pela linha de comandos.

Para começar executou-se o comando `connect localhost 9876`. Este comando devolve o endereço e a porta da máquina com quem queremos trocar informação à aplicação. O primeiro argumento deste comando é o endereço, o segundo é a porta em que o servidor foi aberto.

Ao executar o comando *dirs* estamos a pedir ao servidor que nos envie a sua lista de ficheiros na directoria actual. Se tivéssemos feito *dir* a aplicação não ia comunicar com o servidor pois ia buscar a lista de ficheiros na directoria raiz do cliente.

O comando *cds lei* está a enviar um pedido ao servidor para que este mude a sua directoria raiz actual para *H:/lei/*. Ao enviar o comando *dirs* agora vamos receber uma nova lista de ficheiros.

O comando *get lei.ico* vai pedir o ficheiro à directoria *H:/lei/* pois é a actual no servidor. O servidor vai receber o pedido e enviar o ficheiro para o cliente.

```
sucesso
>cds..
1bytes
sucesso
>dirs
499bytes
H:/
File:          17083 KB    Counte-Strike 1.6 v19.exe
Directory:      CS Source
File:          295457 KB   cs16full_v4+zbot.exe
Directory:      LEI
Directory:      Microsoft Visual Studio 8
Directory:      movies
Directory:      Musiki
Directory:      Pro Evolution Soccer 2008 DEMO
Directory:      RECYCLER
Directory:      System Volume Information
Directory:      The Witcher
File:           9132 KB    Zbot.zip
```

O comando *cds..* pede ao servidor para voltar à directoria pai.

```

>dir
C:/
File:            16 KB      .cd_catalog.xml.swp
Directory:       .Trash-miguel
File:            0 KB      AUTOEXEC.BAT
File:            2 KB      books.xml
File:            0 KB      boot.ini
File:            4 KB      Bootfont.bin
File:            4 KB      cd_catalog.xml
File:            0 KB      CONFIG.SYS
Directory:       Dev-Cpp
Directory:       Documents and Settings
File:           60 KB      estrutura.png
File:            0 KB      IO.SYS
File:            3 KB      lei.ico
File:            0 KB      MSDOS.SYS
File:           46 KB      NTDETECT.COM
File:          244 KB      ntlldr
File:          786432 KB      pagefile.sys
File:           12 KB      PDOXUSRS.NET
Directory:       Programas
Directory:       RECYCLER
File:           66 KB      resultPage2.png
File:           18 KB      socialnet2.xml
Directory:       System Volume Information
Directory:       vbroker
Directory:       WINDOWS
Directory:       xampp
Directory:       xml
>put estrutura.png
61501bytes
sucesso
>dirs
537bytes
H:/
File:          17083 KB      Counte-Strike 1.6 v19.exe
Directory:       CS Source
File:          295457 KB      cs16full_v4+zbot.exe
File:           60 KB      estrutura.png
Directory:       LEI
Directory:       Microsoft Visual Studio 8
Directory:       movies
Directory:       Musiki
Directory:       Pro Evolution Soccer 2008 DEMO
Directory:       RECYCLER
Directory:       System Volume Information
Directory:       The Witcher
File:           9132 KB      Zbot.zip
>

```

O comando *dir* mostra a lista dos ficheiros na directoria do cliente.

O comando *put estrutura.png* envia um pedido de envio ao servidor. O servidor recebe esse comando e então vai esperar por receber um ficheiro. O ficheiro é transferido e depois é executado um comando *dirs* para confirmar que o ficheiro foi mesmo transferido para o servidor.