

Universidade do Minho

Mestrado Integrado em Engenharia Informática



TP2: Serviço de transferência rápida e fiável de dados sobre UDP

Comunicações por Computador

Grupo 1 – PL5



Carla Cruz A80564



Ana Ribeiro A82474



Jéssica Lemos A82061

1. Introdução

Para a unidade curricular de Comunicações por Computador, foi-nos proposta a realização do trabalho prático número dois – Serviço de transferência rápida e fiável de dados sobre UDP – que tem como objetivo implementar um serviço de transferência de ficheiros sobre uma conexão UDP genérica.

Neste trabalho pretende-se incluir todos os conhecimentos adquiridos ao longo das aulas, nomeadamente, o uso de mecanismos de verificação de integridade para controlo de erros, entre outros.

Neste relatório iremos apresentar a implementação para dar resposta ao problema proposto.

2. Especificação do protocolo

2.1. Formado das mensagens protocolares

Nesta secção será apresentada a estrutura que definimos para o PDU, que representa os dados que serão enviados através de um socket. Assim, todos os pacotes enviados terão a seguinte estrutura:

- **info** - Campo do tipo *String*, que serve para identificar o tipo do pacote, isto é, por exemplo verificar se corresponde a um ack.
- **numSeq** - Campo do tipo *int* que funciona como um identificador do pacote.
- **data** – Campo do tipo array de bytes que possui o conteúdo da mensagem, este tem um tamanho máximo de 1024.

2.2. Interações

O nosso projeto é *single client* baseado no algoritmo Stop-and-Wait, existindo apenas um servidor que irá atender um cliente de cada vez. Sendo assim, um cliente deverá efetuar a sua autenticação de modo a iniciar conexão com o servidor, para de seguida ser efetuada as trocas de pacotes. A comunicação entre o cliente e o servidor é efetuado através de UDP recorrendo aos seguintes componentes:

- **Sender:** Responsável por armazenar o conteúdo de um dado ficheiro em pacotes e por enviar esses mesmos por UDP, dirigida para a porta pretendida que depende de a ação pretendida ser upload ou download – 9999 no caso do servidor e 6973 no caso do cliente - e que fica em escuta pelas respostas garantindo que todos os pacotes são recebidos.
- **Receiver:** Responsável por receber os pacotes de dados por UDP e à medida que os pacotes vão sendo recebidos é efetuada a escrita do seu conteúdo no ficheiro. Tal deve-se a que como o algoritmo usado é o Stop-and-Wait não teremos problemas de ordem de chegada dos pacotes. Este irá enviar resposta aos pacotes recebidos permitindo garantir que todos os pacotes serão recebidos.

3. Implementação

Para a implementação da solução do problema proposto, através do recurso à linguagem Java, elaboramos as seguintes classes:

- **Cliente**
Esta classe é responsável, por invocar os métodos necessários conforme a opção do menu escolhida. Assim encontra-se definido o método *initConnection* que trata do início de conexão, isto é, envia um pacote a indicar que pretende iniciar a conexão,

espera que o servidor aceite e depois envia um *ack*. Encontra-se também definido o *closeConnection* em que o cliente envia um pacote a indicar que pretende terminar a conexão, espera que o servidor indique também que pretende terminar a conexão e depois envia um *ack* a confirmar. Para além destes encontram-se os métodos de iniciar sessão e de registar utilizadores.

- **ClientMain**

Esta classe é responsável por inicializar o menu e o cliente, que permite apresentar os menus e ler as opções pretendidas.

- **ServerMain**

Esta classe é responsável por iniciar o servidor. Nesta também são processados os pacotes recebidos, por exemplo, se é recebido um pacote do tipo *CONNECT* o servidor envia outro a dar resposta ao pedido de conexão. Se o pacote recebido contiver informação para realizar um upload é criada a thread com o *Receiver* e caso seja do download a thread com o *Sender*.

- **Utilizador**

Esta classe contém os dados de cada utilizador, nomeadamente o nome e a password. Esta informação será útil na realização da autenticação e no momento de registar novos utilizadores.

- **Menu**

Classe que tem como funcionalidade apresentar a interface ao utilizador. Assim é possível a comunicação entre o Cliente e o Servidor. Esta classe garante que a opção escolhida pelo utilizador seja válida. Esta contém os diferentes menus que poderão ser apresentados.

- **Packet**

O *Packet* é a classe que irá conter a informação relativa ao que está a ser enviado, o número de sequência do mesmo, de forma a garantir a receção ordenada destes ou caso não seja recebido, reenviá-lo. Este também conterá um Array de bytes, *data*, que terá a mensagem convertida em bytes.

- **Sender**

Classe que será responsável por enviar os pacotes com os dados de um determinado ficheiro. Para tal, nesta é lido o conteúdo do ficheiro pretendido para um *ArrayList*. Desta forma, enquanto todos os dados não forem enviados é criado um *Packet* com os dados a enviar cujo tipo é *DATA*, e contém o número de sequência para além da mensagem em bytes. De realçar que o tamanho máximo da mensagem é de 1024. De seguida, é esperado pelo *ack* cujo tipo do pacote a ser recebido será também *DATA*. Caso este não seja recebido será reenviado o pacote com esses dados. Por fim, de modo a indicar que o conteúdo do ficheiro foi todo enviado, é criado e enviado um pacote cujo tipo será *END*. Sempre que é enviado um pacote é incrementado o número de sequência.

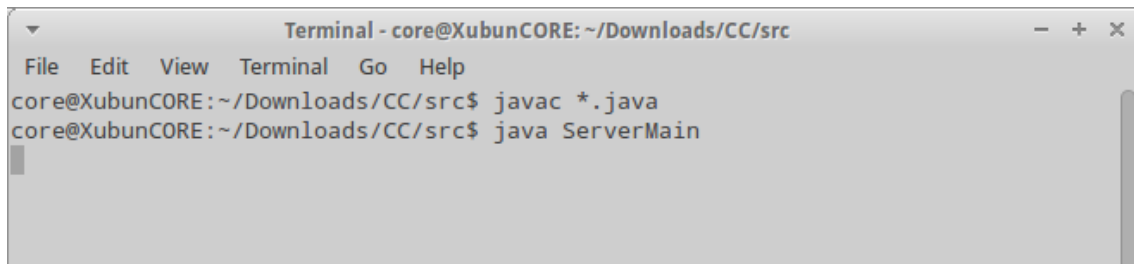
Tendo em conta que os *DatagramPackets* são usados para implementar um serviço de entrega de pacotes sem conexão, os vários pacotes podem ser enviados por diferentes rotas e podem chegar em qualquer ordem não sendo a sua entrega garantida, enviamos e recebemos pacotes através de um *DatagramSocket*.

- **Receiver**

Classe responsável por receber os pacotes de dados enviados na classe apresentada anteriormente. Enquanto o tipo de pacote recebido não for *END* são recebidos os pacotes e escrita a mensagem para o ficheiro, de seguida criamos e enviamos um pacote que será um *ack* ao pacote recebido.

4. Teste e Resultados

Nesta secção será testado o trabalho desenvolvido. Desta forma, primeiro é necessário colocar o servidor a correr, como pode ser observado na Figura 1 e de seguida coloca-se o cliente. Como é possível verificar na Figura 2, é apresentado de imediato um menu ao utilizador. Este menu permite que este se autentique ou registe.

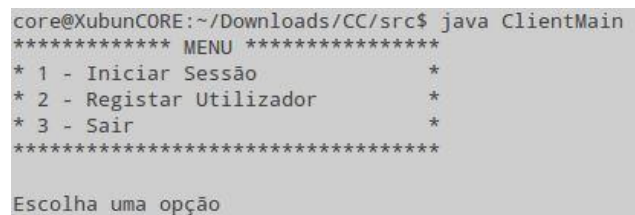


```

Terminal - core@XubunCORE: ~/Downloads/CC/src
File Edit View Terminal Go Help
core@XubunCORE:~/Downloads/CC/src$ javac *.java
core@XubunCORE:~/Downloads/CC/src$ java ServerMain

```

Figura 1 - Servidor a correr



```

core@XubunCORE:~/Downloads/CC/src$ java ClientMain
***** MENU *****
* 1 - Iniciar Sessão *
* 2 - Registar Utilizador *
* 3 - Sair *
*****
Escolha uma opção

```

Figura 2 - Cliente a correr

Neste caso, vamos registar um utilizador para demonstrar o processo. Assim, escolhemos a opção 1 do menu e é-nos solicitado o username e a password que pretendemos utilizar, tal como se pode visualizar na Figura 3. Depois é nos apresentado de novo o menu e vamos autenticar-nos, indicando o username e a password com que estamos registados. Depois de validados os dados é apresentado um novo menu que permite realizar as transferências, como se pode verificar na Figura 4.

```
core@XubunCORE:~/Downloads/CC/src$ java ClientMain
***** MENU *****
* 1 - Iniciar Sessão          *
* 2 - Registar Utilizador     *
* 3 - Sair                    *
*****

Escolha uma opção
2
Vou registar
Username:
Ana Sousa
Password:
AnaSousa
***** MENU *****
* 1 - Iniciar Sessão          *
* 2 - Registar Utilizador     *
* 3 - Sair                    *
*****

Escolha uma opção
1
Username:
Ana Sousa
Password:
AnaSousa
```

Figura 3 - Registo e autenticação

```
***** MENU *****
* 1 - Upload                  *
* 2 - Download                *
* 0 - Terminar conexão     *
*****

Escolha uma opção

```

Figura 4 - Menu

Por exemplo, para fazer um upload é-nos solicitado a diretoria onde se encontra o ficheiro que pretendemos transferir e o seu nome, caso o caminho e o ficheiro esteja correto a transferência inicia, tal como se pode ver na Figura 5. Na eventualidade de pretendemos fazer um download o processo é o mesmo, sendo que a diretoria deve ser onde se pretende guardar e o nome do ficheiro a transferir, como se pode constatar na Figura 6.

```
***** MENU *****
* 1 - Upload          *
* 2 - Download        *
* 0 - Terminar conexão *
*****

Escolha uma opção
1
Digite a diretoria do arquivo.
/home/core/Music/
Digite o nome do arquivo.
CC-Enunciado-TP1-2019.pdf
Uploading...

Upload finished!
Filename: CC-Enunciado-TP1-2019.pdf
```

Figura 5 - Upload

```
***** MENU *****
* 1 - Upload          *
* 2 - Download        *
* 0 - Terminar conexão *
*****

Escolha uma opção
2
Digite a diretoria do arquivo.
/home/core/Music/
Digite o nome do arquivo.
CC-Enunciado-TP1-2019.pdf
Downloading...
Transferencia terminada!
```

Figura 6 - Download

5. Conclusão

Concluído o projeto proposto, com a implementação da maioria das funcionalidades base obrigatórias e também a autenticação e registo de utilizadores, vários dos conceitos lecionados na unidade curricular foram consolidados. Contudo, nem todas as funcionalidades foram realizadas como pretendido, tal como implementar o algoritmo Sliding Window, porém devido a alguns problemas e dificuldades optamos por manter uma versão mais simples do projeto, apresentando o mesmo segundo o algoritmo Stop-and-Wait. Durante a elaboração deste trabalho prático deparamo-nos com diversas dificuldades e problemas na implementação, pelo que foi necessária a tomada de decisões importantes. Para além, disso a escassez de tempo não nos permitiu corrigir problemas com os quais nos deparamos nem implementar determinadas funcionalidades como pretendíamos.