FACULDADE DE
CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE D
COIMBRA

Departamento de Engenharia Informática

Trabalho Prático Nº1

ucDrive - Repositório de ficheiros na UC

Sistemas Distribuídos (SD)

José Aguiar Nº 2019224624

Tiago Oliveira Nº 2019219068

2021/2022

Professores: Filipe Araújo
Hugo Amaro

Índice

Arquitetura de Software e Funcionamento da Plataforma

Nesta implementação da plataforma ucDrive foi desenvolvida uma arquitetura bastante semelhante com a fornecida no enunciado do projeto, como se pode verificar na Figura 1.

No entanto, ao contrário do fornecido no enunciado, não foi implementada nenhuma *Admin Console*, não tendo sido aplicado em prática funcionalidades que usassem RMI.

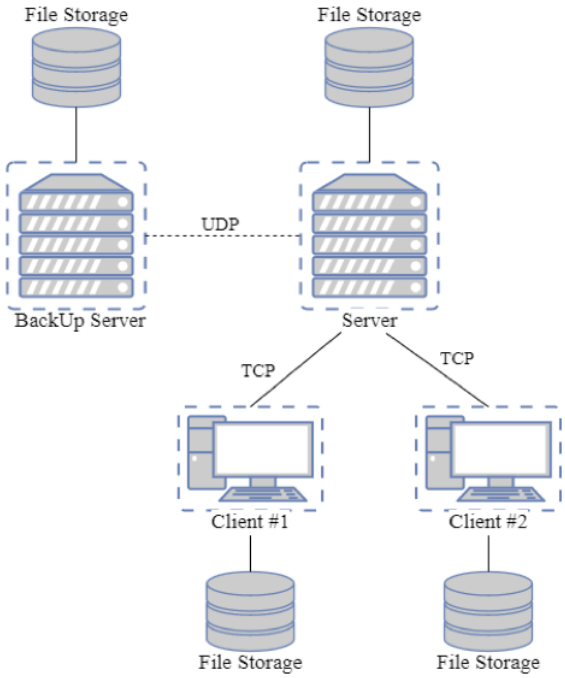

Figura 1 - Arquitetura da plataforma ucDrive

Para o bom funcionamento da plataforma foi necessário recorrer ao uso de *threads* e do tratamento de concorrência das mesmas. Existem 12 ficheiros .java, contendo cada um deles uma classe correspondente. Os ficheiros são os seguintes:

Server.java

Neste ficheiro existe todo o código responsável pelo funcionamento geral do servidor, sendo criadas todas as diretorias respectivas aos clientes registados no ficheiro usersData.txt. Após a leitura deste ficheiro é criada uma *HashSet* que guarda objetos do tipo *User*, objetos esses referentes ao ficheiro User.java.

No caso do cliente já ter estabelecido uma ligação anterior irá ser verificado o ficheiro que contém a última diretoria visitada dos clientes (lastDirs.txt). É também aqui que é verificada a presença de um servidor principal, sendo que se tentar ligar em novo servidor ele irá tentar criar um socket no porto definido para o servidor principal. No caso desse porto já estar a ser usado, o novo servidor irá ser automaticamente ligado ao porto definido para o servidor secundário.

Connection.java

Neste ficheiro existe uma classe que herda todas as propriedades de uma *thread*, ou seja, de cada vez que um objeto do tipo *Connection* é criado é criada uma *thread*, de modo a ser possível ter uma *thread* para tratar cada canal de comunicação com um cliente.

É através deste ficheiro que a comunicação dos clientes com o servidor é estabelecida, sendo tratados, de forma adequada, todos as ações, disponíveis, que o cliente pretenda realizar. É ainda feita uma autenticação do cliente de maneira a poder garantir uma ligação sem interferência de outros utilizadores nos ficheiros e directorias deste cliente autenticado.

Client.java

O ficheiro que equivale ao terminal de utilizador, pois é aqui que é criado o *socket* para a ligação ao servidor e é através deste ficheiro que o cliente pode mandar qualquer comando de ação. Inicialmente o cliente precisa de indicar o seu *username* e a *password* correspondente de forma a ficar autenticado. Quando o cliente muda a sua palavra passe é obrigado a estabelecer uma nova ligação ao servidor, sendo este processo feito de forma automática.

Caso o terminal do cliente seja executado antes do servidor o cliente recebe uma mensagem de erro a dizer que nenhum dos servidores está disponível. Isso é possível devido à implementação feita no ficheiro Client.java e no ficheiro Connection.java.

UDPPingServer.java e UDPPingClient.java

Nestes ficheiros são criados um novo *socket* e uma nova *thread* para enviar pings do servidor de backup para o servidor principal e verificar se são recebidos com sucesso. No caso de não se obter resposta do servidor principal, no espaço de 10 segundos, é considerado que o servidor principal teve uma avaria e o secundário irá passar a funcionar como servidor principal, aceitando assim ligações dos clientes.

Upload.java e Download.java

Tal como os nomes sugerem, neste dois ficheiros existem as classes referentes ao envio e receção de ficheiros, funcionando apenas para a ligação TCP entre os clientes e o servidor. Ambas as classes herdam propriedades de uma *thread*, sendo assim, é criada uma nova *thread* quando se cria um novo objeto destes tipos. Assim, é possível ter diversos ficheiros a ser enviados/recebidos ao mesmo tempo, sem existir a necessidade de esperar pelo fim da transferência anterior. Os ficheiros são enviados/recebidos em “pacotes” de 8 kb, podendo ser alterado para outro valor a qualquer momento. No caso do ficheiro ter um tamanho inferior é enviado na totalidade num só “pacote”.

ReceiveFile.java e SendFile.java

Tal como nos ficheiros acima, é possível enviar e receber ficheiros, divididos em "pacotes". No entanto, neste caso os ficheiros são enviados por UDP e são enviados na ligação entre o servidor principal e o servidor de backup, para evitar perdas de ficheiros em casos de avaria do servidor principal. Caso o servidor secundário ainda não tenha sido ligado, os ficheiros criados na diretoria do servidor principal são guardados numa *queue*, de modo a serem enviados assim que o servidor de secundário seja iniciado.

User.java

Este ficheiro serve apenas de auxílio para todos os outros, contendo uma classe User que guarda todas as informações relativas a um cliente.

Data.java

Este ficheiro foi criado para assegurar que as datas introduzidas no ficheiro de configuração dos utilizadores (usersData.txt)

RandomString.java

Aqui existe uma classe que apenas gera um *token*, que poderia ser usado para a confirmação da autenticação do cliente. No entanto, essa funcionalidade não está devidamente implementada, tornando assim este ficheiro irrelevante para o funcionamento da plataforma.

Descrição do mecanismo de Failover

Foi necessário implementar mecanismos de *failover* para existir sempre um servidor principal pronto a estabelecer ligação com os clientes. Inicialmente, o servidor irá tentar ligar-se como servidor primário, no caso de não conseguir, liga-se como servidor secundário. Após o servidor secundário estar ligado, este vai enviar blocos de 10 pings de 10 em 10 segundos. No caso de 8 pings de um bloco de pings falhar, o servidor secundário assume que o primário teve uma falha e deixou de estar ativo. De seguida, o servidor secundário substitui o servidor primário e assume as ligações com os clientes. Assim que o servidor que teve uma falha voltar a estar ativo vai passar as funcionalidades de servidor de backup.

Se um cliente enviar um ficheiro para o servidor, este mesmo ficheiro é enviado do servidor principal para o secundário, via UDP. No caso do servidor secundário ainda não estar ativo, os ficheiros são guardados numa *queue* para serem enviados imediatamente a seguir à ligação do servidor de backup.

Comunicação Terminal - Servidor

De seguida serão apresentados todos os comandos ao alcance do utilizador, indicando a sua funcionalidade, bem como alguns aspetos importantes de notar.

- **passwd ->** altera a password do cliente, verificando se é diferente da atual.

- **config -main [ip] [port] ->** altera a configuração da ligação do servidor principal (ip e porto).
- **config -backup [ip] [port] ->** altera a configuração da ligação do servidor secundário (ip e porto).

- **ls -server ->** lista todos os ficheiros e pastas da diretoria do servidor, criando uma "hierarquia" consoante o nível das subpastas e dos seus ficheiros.
- **ls -client ->** lista todos os ficheiros e pastas da diretoria do cliente, criando uma "hierarquia" consoante o nível das subpastas e dos seus ficheiros.

- **cd -server "directory" ->** muda a diretoria atual do servidor.
- **cd -client "directory" ->** muda a diretoria atual do cliente.
- **cd [-server / -client] ..** -> muda para a diretoria "pai" da diretoria atual.

Nota: nos 3 comandos acima é necessário colocar a diretoria dentro de aspas no caso de existirem caracteres "especiais", isto é, espaços, acentos…

- **push [filename] [destination] ->** carrega um ficheiro para o servidor.
- **pull [filename] [destination] ->** descarrega um ficheiro do servidor.

Nota: nos comandos acima o cliente tem de ir para a diretoria onde se encontra o ficheiro a ser carregado e para a diretoria onde quer guardar o ficheiro, indicando o nome do novo ficheiro a ser criado.

- **mkdir -server [folderName] ->** cria uma nova diretoria no servidor, dentro da diretoria em que o cliente se encontra.

- **exit ->** termina a ligação do cliente.

Distribuição de tarefas pelos elementos do grupo

Pela análise do enunciado foram definidas tarefas, de forma a repartir o projeto em subproblemas. Essas tarefas estão listadas abaixo.

1. Autenticação do utilizador no terminal de cliente. Os utilizadores devem poder autenticar-se num terminal de cliente através do nome de utilizador e respectiva password. Após a autenticação o servidor deve colocar o utilizador na directoria Home, ou na directoria da última sessão caso este já se tenha autenticado no sistema no passado.
2. Alterar a password do utilizador. O utilizador deve poder alterar a sua password. Para que isto seja possível, estes utilizadores devem estar autenticados através do terminal de cliente. Após a alteração, o terminal deve desconectar o cliente e pedir uma nova autenticação de modo a manter a segurança na alteração da password.
3. Configurar endereços e portos de servidores primário e secundário Na aplicação cliente deve ser possível configurar os endereços IP e portos dos dois ucDrive Servers, o primário e o secundário.
4. Listar os ficheiros que existem na diretoria atual do servidor. Os utilizadores devem poder listar ficheiros existentes numa directoria do servidor. Esta funcionalidade tem como objectivo poder listar os ficheiros de uma directoria remota.
5. Mudar a diretoria atual do servidor. Os utilizadores devem poder mudar a diretoria corrente no servidor para poderem navegar entre as várias diretorias a que têm acesso, estando sempre limitados às diretorias dentro da sua home.
6. Listar os ficheiros que existem na diretoria atual do cliente. Os utilizadores que estejam a aceder ao sistema através de um terminal de cliente devem poder listar os ficheiros existentes na directoria corrente do cliente. Esta funcionalidade tem como objectivo poder listar os ficheiros de uma directoria local.
7. Mudar a diretoria atual do cliente. Os utilizadores devem poder mudar a diretoria corrente no seu sistema de ficheiros local para poderem navegar entre as várias diretorias existentes no sistema local.
8. Descarregar um ficheiro do servidor. Os utilizadores devem poder descarregar um ficheiro do servidor. Para que o possam fazer, devem estar autenticados como tal e devem ter permissões de leitura no ficheiro que pretendem descarregar. Ao descarregar um ficheiro do servidor, o cliente deve ficar com uma cópia do ficheiro armazenado localmente. A descarga do ficheiro deve ser efetuada através de um socket independente conectado com o servidor atual mas num porto diferente.
9. Carregar um ficheiro para o servidor. Os utilizadores devem poder carregar um ficheiro para o servidor. Para que o possam fazer, devem estar autenticados como tal e devem ter permissões de escrita na directoria para onde pretendem carregar o ficheiro. Ao carregar um ficheiro para o servidor, o servidor deve ficar com uma cópia do ficheiro armazenado na directoria. O carregamento do ficheiro deve ser efetuado através de um socket independente conectado com o servidor atual mas num porto diferente.
10. A troca de servidor conectado por parte do cliente, que deve passar a ligação do primário para o secundário no caso de falha. Para a implementação deste mecanismo, o servidor secundário, enquanto o primário está ativo não aceita

ligações dos clientes, passando a aceitar apenas quando detecta que o primário tem um problema e assume a sua função.

11. Réplicas dos ficheiros no ucDrive Server através do protocolo UDP, para questões de replicação de dados e mecanismos de failover.
12. Se ou quando o servidor ucDrive original recuperar, deverá tomar o papel de secundário e não de primário.
13. O servidor secundário deve trocar periodicamente, via UDP, pings ou heartbeats com o servidor primário. Se algumas destas mensagens se perderem, o secundário assume que o primário avariou e assume essa função.

A distribuição das tarefas foi realizada da seguinte maneira:

- **Tiago Oliveira**
  - Tasks - 2, 3, 4, 5, 6, 7, 8, 9
- **José Pedro Aguiar**
  - Tasks - 1, 9, 10, 11, 12, 13

As tarefas propostas para cada um foram cumpridas por ambas as partes. De forma simplificada, o Tiago Oliveira esteve responsável por todo o processo que envolvia diretorias no Servidor Principal bem como da troca de ficheiros entre clientes e servidores. Já o José Aguiar ficou responsável pelo desenvolvimento do Servidor Secundário e Principal, Pings, Replicação de Ficheiros e a Funcionalidade do Login. No entanto, o projeto foi desenvolvido em conjunto e com participação/intervenção de ambas as partes em todas as tarefas.

Testes efetuados à plataforma

Testes	Resultado
Autenticar Clientes (válidos e inválidos) ao serviços ucDrive.	
Alterar a password do utilizador e essa alteração ser guardada em ambos os servidores.	
Configurar endereços e portos de servidores primário e secundário na aplicação cliente.	
Listar os ficheiros que existem na diretoria atual do servidor.	
Mudar a diretoria atual do servidor. (utilizadores usados: tmatos, filipa93)	
Tentar e não mudar de diretoria do servidor para pastas sem acesso por parte do utilizador	
Criar diretorias,através de um utilizador, na diretoria atual do servidor.	
Listar os ficheiros que existem na diretoria atual do cliente.(utilizadores usados: tmatos, filipa93)	
Descarregar um ficheiro do servidor, via TCP. (utilizadores usados: tmatos, filipa93)	
Carregar um ficheiro para o servidor, via TCP. (utilizadores usados: tmatos, filipa93)	
Carregar um ficheiro com mesmo nome,via TCP (levando a substituição do ficheiro e não a duplicação do mesmo)	
A troca de servidor conectado por parte do cliente, que deve passar a ligação do primário para o secundário no caso de falha (8 pings perdidos em 10 ocorre a troca).	
Em caso de falha do servidor Principal , servidor secundário assume	
O servidor secundário deve trocar periodicamente, via UDP, pings ou heartbeats com o servidor primário.	
Réplicas dos ficheiros no ucDrive Server através do protocolo UDP.	
Quando “down” o servidor ucDrive original e este recuperar, este vai tomar o papel de secundário e não de primário.	
Projeto corre em duas máquinas	