Universidade de Coimbra

Sistemas Distribuídos

LEI - DEI - 2021/2022

Relatório do trabalho 1

Joel Oliveira - 2019227468
Tomás Mendes - 2019232272

2 de abril de 2022

1 Introdução

O projeto tinha como principal objetivo, o desenvolvimento do ucDrive, uma plataforma de armazenamento de ficheiros partilhados por alunos e docentes da Universidade. Para o fazer, era pedido para desenvolver um servidor *multithreaded* com os mecanismos de *failover* necessários de modo a garantir a disponibilidade do serviço.

2 Requisitos

- Cliente
  - ✓ Alterar a sua diretoria atual
  - ✓ Listar os ficheiros na diretoria atual
  - ✓ Enviar ficheiros para o servidor
  - ✓ Fazer *download* de ficheiros do servidor
  - ✓ Alterar a sua password
  - ✓ Configurar os endereços e portos dos servidores primários e secundários
  - ✓ Fazer *login* no servidor
  - ✓ Alterar a diretoria no servidor
  - ✓ Listar os ficheiros na diretoria atual no servidor
- Servidor
  - ✓ Suportar *multithread*
  - ✓ *Sockets* TCP para comunicação com cliente
  - ✓ *Sockets* UDP para comunicação entre servidores primários e secundários
  - ✓ Bloqueio de comunicações de cliente com o servidor secundário
  - – Mecanismos de *failover*
    - ✓ *Hearthbeat/pings* do servidor secundário para o primário
    - ✓ Replicação de ficheiros, quando são recebidos

3 Funcionamento

Para correr e testar a aplicação, é necessário executar os ficheiros *ucDrive.jar* e *terminal.jar*. O *ucDrive.jar* inicializa um servidor como primário. Se for inicializado outro, será considerado como secundário. O *terminal.jar* inicializa uma instância de cliente que se conecta ao servidor primário. Como o projeto foi realizado num grupo de dois alunos, não há um consola de administração, mas sim ficheiros de configuração estática do servidor e cliente. Usar os seguintes comandos para correr:

```
$ java -jar ucDrive.jar

$ java -jar terminal.jar
```

Para o funcionamento de ambas as aplicações, é necessário um ficheiro *config.yaml* na pasta *server*, para os servidores, e *client* para os vários clientes. Ambas as pastas *server* e *client* precisam de ter uma pasta *users*, com os vários utilizadores das aplicações. A pasta entregue contém já as pastas e os ficheiros necessários para o funcionamento das aplicações cliente e servidor.

4 Arquitetura

4.1 Sistema de ficheiros

Ao abrir a aplicação cliente, é pedido um *username*. Este é verificado se existe uma diretoria referente a esse utilizador. Apenas é possível aceder aos utilizadores que tenham sido criados anteriormente, não havendo a possibilidade de registar novos utilizadores.

Após escolher o utilizador, é possível usar os comandos disponíveis. Um utilizador pode listar todos os comandos disponíveis com o comando *help*.

Enquanto o utilizador não se autenticar, os comandos *ls*, para listar os ficheiros e *cd*, para mudar a diretoria, funcionam no sistema de ficheiros local. Após a autenticação, estes comandos passam a ter efeito no sistema de ficheiros do servidor.

4.2 Transferência de ficheiros

Os comandos *send*, para enviar um ficheiro para o servidor e *download*, para descarregar um ficheiro do servidor, apenas funcionam se o utilizador estiver autenticado. Ambos os comandos utilizam o protocolo TCP para a transferência de ficheiros por se tratar de uma ligação cliente-servidor.

É criada uma *thread* de ambos os lados para a transferência de um ficheiro, de modo a não interromper a utilização do servidor. Para enviar ou descarregar um ficheiro, o cliente faz o pedido ao servidor. Este abre um *socket* na porta 0, que faz com que seja atribuída uma porta disponível, e envia ao cliente a porta para que este se possa conectar. Após estabelecida a conexão, o ficheiro é divido em N *bytes* para ser enviado. O ficheiro é lido conforme é enviado, de forma a evitar ler o conteúdo do ficheiro todo para memória.

4.3 Mecanismos de *failover*

De modo a garantir a integridade do serviço, existem alguns mecanismos implementados para o efeito. A comunicação e conexão entre servidores é feita através do protocolo UDP.

4.3.1 Hearthbeats

Os servidores implementam um mecanismo de *hearthbeats* de modo a verificar se o servidor primário continua ativo. O servidor secundário envia um *DatagramPacket* com um número, que é incrementado depois de receber o *acknowledge* do pacote. Após um certo número de *hearthbeats* falhar, o servidor secundário assume-se como primário e passa a aceitar conecções de clientes. O

intervalo entre o envio dos pacotes, o número de pacotes que não receberam um *acknowledge* e o porto uasado para este mecanismo, pode ser alterado no ficheiro de configuração do servidor.

4.3.2 Transferência de ficheiros

Como é usado o protocolo UDP para transferência de ficheiros entre os sistemas de ficheiros dos dois servidores, foram implementados alguns mecanismos do protocolo TCP no protocolo UDP. Primeiramente, e tal como na implementação feita para o TCP, é criado um socket UDP na porta 0 e enviado para o *receiver*(o servidor secundário). São criadas *threads* em ambos os servidores para que estes possam continuar as suas operações. O ficheiro é então divido em blocos de 1024 *bytes* e são usados os dois primeiros *byets* para para o número do pacote. O UDP, por *default*, não implementa nenhuma mecanismo de sincronização dos pacotes, e para isso, precisamos de garantir que todos os pacotes foram recebidos. Os *bytes* reservados ajudam, através de um *acknowledge* de receção do pacote, a garantir que não se perde nenhum pacote, e que estes chegam por ordem. A implementação da transferência de ficheiros usando o UDP, foi baseado de [1]

5 Testes

Para garantir a integridade e maximizar o *uptime*, tanto da aplicação cliente como do servidor, foram criados alguns testes funcionais. Os testes efetuados encontram-se descritos na tabela [1], bem como o resultado do mesmo.

Instrução	Validação	Resultado
login	password errada impede conexão	✓
login	password correta concta ao servidor	✓
send	envia ficheiro do cliente para o servidor	✓
download	cliente recebe ficheiro do servidor	✓
passwd	altera a palavra passe do utilizador atual	✓
help	imprime os comandos possiveis de realizar, e noções do que cada um faz	✓
ls	imprime informçãoes de ficheiros e pastas do diretorio atual	✓
cd	altera para o path absoluto especificado ou para /home em caso de caminho vazio	✓
config	altera host e porta ao qual conectar	✓
server	definição automática de primário/secundário	✓
failover	servidor secundario assume-se primario	✓
failover	heartbeat persistente	✓
failover	transparência por parte do cliente	×
sincronização	servidor primário envia ficheiro para servidor secundario	✓
sincronização	alteração de password é replicada no servidor secundario	✓
mecanismo retry	em caso de crash do cliente, retomar operação interrompida	×

Tab. 1: Testes Funcionais

6 Bibliografia

1. Implementação da transferência de ficheiros usando o protocolo UDP
https://gist.github.com/absalomhr/ce11c2e43df517b2571b1dfc9bc9b487