Relatório Trabalho Prático 1

ucDrive

Sistemas Distribuídos

2021/2022

Marco da Cruz Pais, 2019218981
Simão Carvalho Monteiro, 2019215412

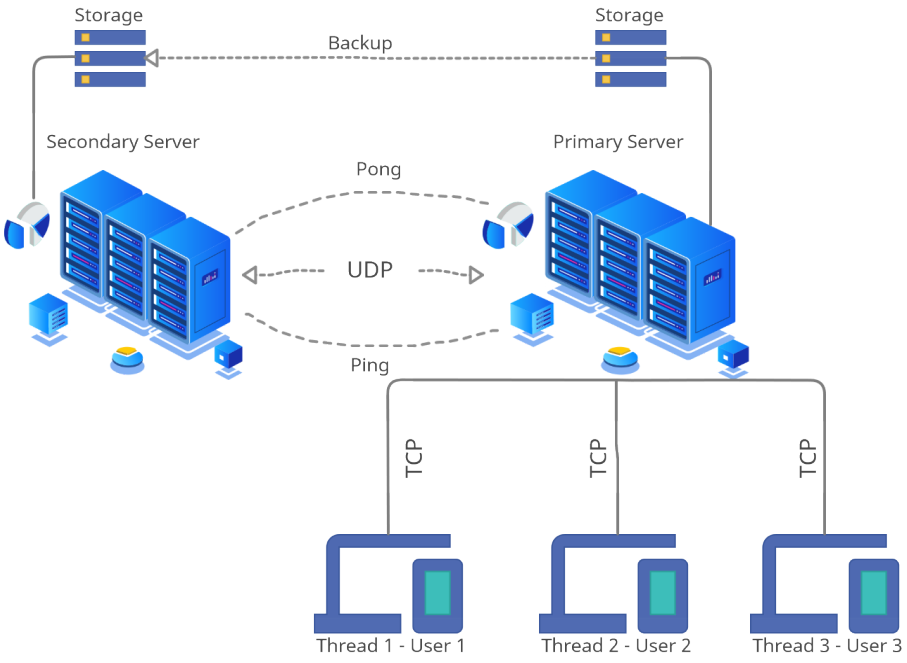
Índice

Introdução

No âmbito da cadeira de Sistemas Distribuídos foi proposta a realização de um trabalho que envolve sockets, streams e comunicação via UDP e TCP. Foi realizado em java e foi necessário implementar os diferentes mecanismos lecionados nas aulas teóricas e práticas.

Arquitetura de software

Na figura em baixo está representada a arquitetura do nosso sistema ucDrive.



O nosso sistema é constituído, principalmente, por:

- **Primary Server** - Este aceita ligações por parte dos utilizadores, com quem comunica via TCP, criando sockets e threads diferentes para os mesmos.
- **Secondary Server** - Este serve como backup e mantém a comunicação via UDP com o servidor principal para a troca de ficheiros. Caso o primário falhe este toma o seu lugar, mantendo a aplicação utilizável.

- **Threads User** - São criadas pelo servidor primário. São responsáveis por responder e processar as mensagens do seu respetivo utilizador, oferecendo assim todas as funcionalidades da drive aos utilizadores.

Funcionamento do servidor ucDrive

Arranque do Servidor

No seu arranque, o servidor ucDrive abre o ficheiro de propriedades ***conf.properties*** e carrega os diferentes endereços e portos, primários e secundários, para uma variável global. Abre, também, o ficheiro ***users.properties*** que contém a informação dos utilizadores da plataforma e carrega para uma variável global.

De seguida o servidor verifica se o endereço primário está ocupado. Caso não esteja, cria um socket com esse endereço e uma thread para responder aos pings de um servidor secundário. Fica, então, à espera de conexões dentro de um ***loop*** e cria ***threads*** para cada utilizador conectado. Caso esteja ocupado, cria um socket com o endereço secundário e envia pings para o servidor primário.

Comunicação via UDP

As funções abaixo descritas irão ser elaboradas com mais detalhe na secção dos mecanismos de failover.

ping() e ***pong()*** são funções usadas para testar se o servidor primário não falha e para fazer a troca de servidores, caso falhe. O servidor secundário envia continuamente ***pings*** via UDP para o servidor principal, que responde com ***pongs*** até que haja algum problema para ser efetuada a troca. Na função de ping também é usado um catch de uma exceção para na inicialização do servidor ser escolhido o endereço primário ou secundário.

sendBackup() e ***receiveBackup()*** permitem ao servidor primário efetuar backup dos seus ficheiros no servidor secundário. O servidor

primário irá chamar esta função sempre que existam ações como o carregamento de um ficheiro, uma mudança de diretoria ou uma mudança na password de utilizador. O servidor secundário terá uma thread aberta para receber esses ficheiros e fazer uma réplica do servidor primário.

Comunicação via TCP

A partir do main são criadas as ***threads Connection*** de cada utilizador, que guardam a informação do ***socket*** a que se deve conectar. Criam-se também as ***streams in*** e ***out*** de dados.

Dentro da classe Connection temos as seguintes funções:

- ***run()*** - É chamada quando a thread é criada e tem como função receber os comandos dos utilizadores e chamar as funções respetivas a esses comandos;
- ***authentication()*** - Usada para autenticar os utilizadores. Recebe nome de utilizador e password e verifica se estes foram inseridos corretamente. Caso seja a primeira vez que um utilizador se está a autenticar, é criada uma diretoria para esse utilizador e um ficheiro ***<utilizador>.properties***. Este contém sempre a última diretoria remota do utilizador e o ficheiro de retry, caso o terminal cliente do utilizador tenha ficado indisponível no decorrer de um download;
- ***changePasswd()*** - Utilizada para alterar a password do utilizador. Recebe a password atual e, caso seja a correta, pede ao utilizador para inserir a nova password e a sua confirmação. Caso estas sejam iguais, muda o ficheiro ***users.properties*** com a nova password do utilizador e remove a autenticação do mesmo do servidor;
- ***listFiles()*** - Percorre o diretório atual e guarda todos os ficheiros e subdiretórios numa string separada por espaço e é enviada para o utilizador. Quando o nome de algum ficheiro

tem espaços, esses são substituídos por ***"%20"***, que do lado do cliente são removidos;

- ***changeDir()*** - Se o diretório de mudança recebido for ***".."***, o diretório remoto vai ser alterado para o diretório que contém o atual. Caso o diretório de mudança for ***"."***, ***"/"*** ou ***"home"***, o diretório remoto irá mudar para o ***"home***" do utilizador. Caso seja apenas um diretório, é verificado se este existe. Se sim, altera-se; se não, fica no atual;
- ***getFile()*** - Tem como função enviar ficheiros para o diretório local do utilizador. Recebe o nome do ficheiro e verifica se este existe. Caso exista é enviado o tamanho do ficheiro para o utilizador e é criada uma thread que envia o ficheiro em pedaços de 1024 bytes para o utilizador. Caso se perca a conexão é guardado no ficheiro ***<utilizador>.properties*** o nome do ficheiro, para se tentar novamente o download quando o utilizador se voltar a conectar;
- ***putFile()*** - Tem como função receber ficheiros e guardá-los na diretoria remota do utilizador. Recebe o nome do ficheiro e o tamanho. Caso o tamanho seja válido, começa a transferência criando uma nova thread que vai receber os pedaços de 1024 bytes e escrever num ficheiro na diretoria remota até receber o ficheiro todo.

Mecanismo de failover

Foram usadas as funções ***ping()*** e ***pong()*** para testar se o servidor primário não falha e para fazer a troca de servidores caso falhe.

Na função de ***ping()*** é sempre testado se o endereço primário está em uso. Se estiver, esse servidor inicia-se como secundário sem aceitar ligações. O servidor secundário envia ***pings*** via UDP para o servidor

principal, que responde com ***pongs***. Se o servidor primário não responder 5 vezes, o servidor secundário passa a aceitar ligações e assume a função de servidor principal.

A mudança de servidor é feita de forma automática por parte do terminal cliente. Quando o cliente faz um catch de uma exceção ao enviar um comando para o endereço do servidor principal, troca de servidor automaticamente e conecta-se ao secundário, sendo pedido para se autenticar novamente. Na nossa implementação pode existir sempre troca de servidores caso haja problemas e sejam reiniciados, não havendo interrupções no uso por parte do utilizador.

Distribuição de tarefas pelos elementos do grupo

Primeiramente, o trabalho foi dividido em 6 grandes tarefas:

- Terminal cliente;
- Servidor ucDrive;
- Failover;
- Tratamento de exceções;
- Qualidade de código;
- Presente relatório.

O primeiro e o segundo ponto foram divididos de forma igual pelos dois elementos, sendo que cada um começou por implementar tanto as funções principais do lado do cliente como do lado do servidor .

O terceiro ponto foi feito em conjunto, pois foi aquele que nos causou uma maior dificuldade. Um de nós fez o ponto quatro e o outro fez o ponto cinco.

Por fim, o presente relatório foi feito por ambos.

Descrição dos testes feitos à plataforma

Nº Teste	Descrição	Resultado
1	Login de um cliente com credenciais válidas	Passou
2	Rejeição de login com credenciais inválidas	Passou
3	Mudar de diretoria para uma pasta existente	Passou
4	Rejeição do comandos sem argumentos	Passou
5	Rejeição de mudança de diretoria para um ficheiro	Passou
6	Rejeição de mudança de diretoria para antes da pasta home	Passou
7	Upload de ficheiros para o server	Passou
8	Download de ficheiros do server	Passou
9	Sincronização entre servidores após upload de ficheiros	Passou
10	Sincronização entre servidores após mudança de diretoria	Passou
11	Funcionamento do servidor para vários utilizadores em simultâneo	Passou
12	Mudança de servidor para o secundário	Passou
13	Servidor principal assumir o papel de secundário após ir abaixo	Passou
14	Upload do mesmo ficheiro duas vezes	Passou
15	Envio sincronizado de ficheiros entre servidores via UDP	Passou
16	Mudança de diretoria local	Passou
17	Listagem dos ficheiros locais	Passou
18	Listagem dos ficheiros do server	Passou
19	Mudança de diretoria local	Passou
20	Retry download quando cliente crasha	Passou

Conclusão

Este trabalho prático ajudou-nos a consolidar o conhecimento obtido nas aulas teóricas e práticas sobre sockets UDP e TCP, threads e streams. A implementação dos diferentes requisitos exigiu de nós uma procura sobre o material relacionado com o que estava a ser desenvolvido e, ainda, expôs de uma forma diferente os vários temas lecionados nas aulas o que, por sua vez, ajudou à melhor compreensão dos mesmos.