

# [Sistemas Distribuídos]

ucDrive: Repositório de ficheiros na UC

## [Relatório do projeto ucDrive]

## Relatório realizado por:

## Alexandre Gameiro Leopoldo – 2019219929

## Luís Miguel Gomes Batista – 2019214869

# Estrutura da comunicação entre aplicações de Servidor

Servidor Primário

Servidor Secundário

Envia heartbeats para verificar se está ativo

Socket UDP para receção de heartbeats

Socket UDP para envio de ficheiro

Socket UDP para gerir portos de comunicação failover

Socket UDP para envio de heartbeats

Socket UDP para receção de ficheiro

Envia acknowledgement do heartbeat recebido

Socket UDP para receber portos de comunicação failover

Envia intenção de replicar ficheiro

Envia porta disponível para replicar ficheiro

Envia ficheiro a ser replicado

Envia acknowledgement de receção do ficheiro

# Estrutura de comunicação entre Cliente e Servidor (Primário)

Envia ficheiro para servidor

Socket TCP para upload de ficheiros para o servidor

Responde consoante o pedido recebido

Envia comandos para interagir com o servidor

Socket TCP para download de ficheiros do cliente

Socket TCP para upload de ficheiros para o cliente

Socket TCP para comunicação com clientes

Socket TCP para download de ficheiros do servidor

Socket TCP para comunicação com o servidor

Servidor Primário

Cliente

Solicita ficheiro para download

Envia ficheiro para cliente

Solicita ficheiro para upload

Arquitetura de software detalhadamente descrita. Deverá ser focada a estrutura

de threads e sockets usadas, bem como a organização do código.

# Arquitetura do Servidor

No planeamento da arquitetura da aplicação do Servidor foi optado o uso de uma thread por conexão a cada cliente de modo a constituir uma “Thread-per-connection architecture”. A thread principal, após um conjunto de etapas de configuração, fica em loop a aceitar ligações TCP para comunicação com utilizadores. Estas comunicações são posteriormente geridas por uma thread da classe “ClientHandler” responsável por receber e responder a comandos do cliente.

Ao iniciar, a aplicação servidor lê um ficheiro de dados de clientes e dá opção de alterá-lo. Após isto pede endereços e portos relativos ao seu próprio servidor e a outro que faz parte do mecanismo de failover.

# Arquitetura do mecanismo de failover

Para a arquitetura do mecanismo de failover da aplicação Servidor, é essencial a distinção entre servidor primário e secundário. Esta diferenciação é realizada através de heartbeats no início do programa em que, caso haja um “acknowledgement”, a aplicação saberá que está outro servidor ligado e por isso passa a ser o servidor secundário. Caso contrário, ao fim de um dado número de heartbeats sem resposta, a aplicação assumirá que é o servidor primário.

A receção de heartbeats no servidor principal é realizada por uma thread da classe “UDPHeartbeat” que se encontra a correr em paralelo à escuta numa socket UDP. No servidor secundário o envio de heartbeats é feito por uma função na thread principal.

Estando estabelecida esta comunicação UDP, o servidor primário sabe que terá de replicar dados para o servidor secundário, sendo este passo fundamental para que não haja perdas de informação. De modo que possa haver um envio de ficheiros, é utilizada a classe “UDPPortManager” responsável pela troca de portos para comunicações UDP e envio/receção de diretorias para manter os dados atualizados.

# Arquitetura do Cliente

Detalhes sobre o funcionamento do servidor ucDrive. Deverá explicar detalhadamente

o funcionamento dos métodos remotos disponibilizados e eventuais callbacks

usados.

Detalhes do mecanismo de failover.

Distribuição de tarefas pelos elementos do grupo.

Descrição dos testes feitos à plataforma (tabela com descrição e pass/fail de cada

teste).