Projeto 1
Relatório

UNIVERSIDADE DE COIMBRA

Licenciatura em Engenharia Informática

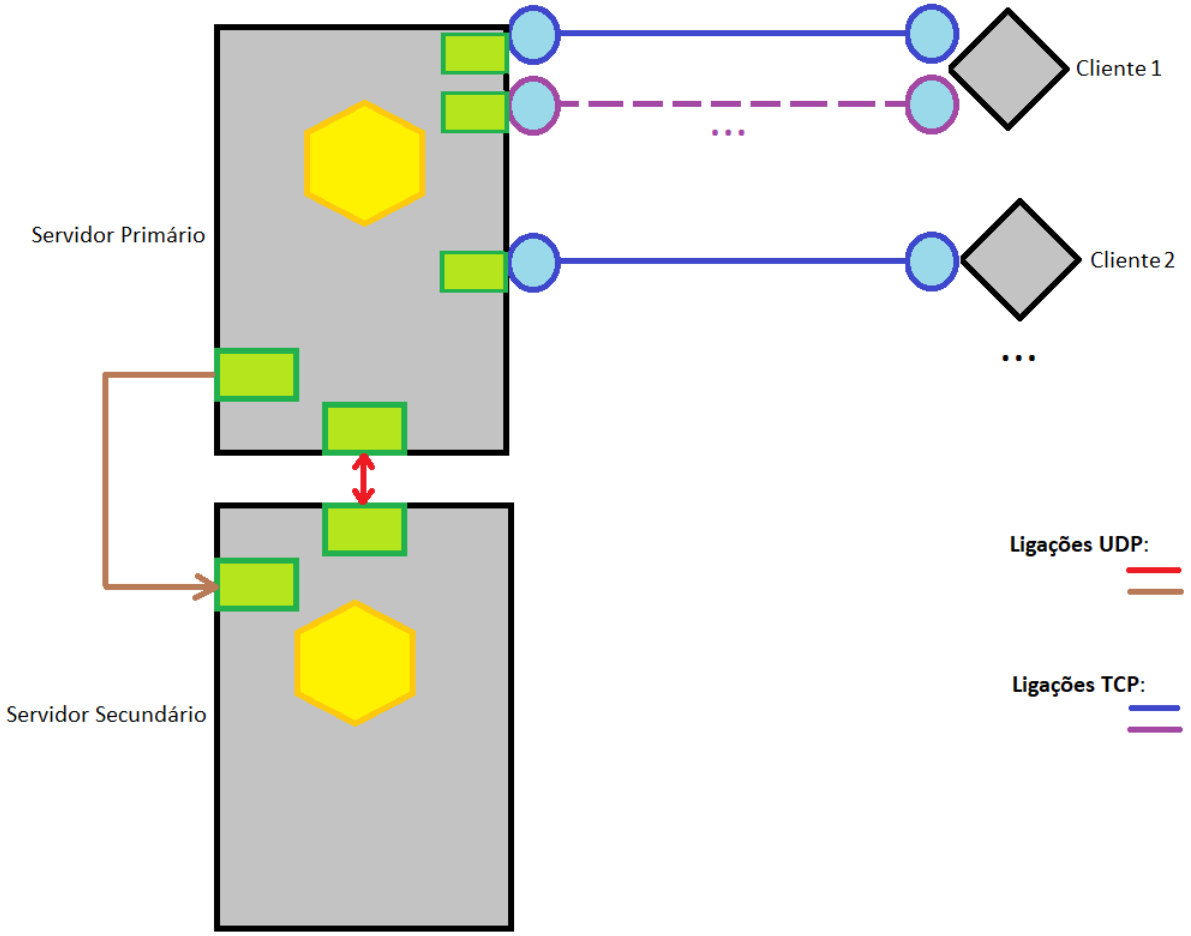
Sistemas Distribuídos

Autores:

João Leite | 2018285542

Nuno Silva | 2018285621

- Arquitetura de software



Considerando a imagem acima que simboliza o cenário implementado:

- ☐ Cada servidor está em execução (*representado pelo retângulo preto*), sendo que aqui assumimos que ambos estão disponíveis;
- ☐ A cada nova ligação de um cliente ao servidor primário - que é por default o servidor que aceita conexões - abrimos uma nova thread

(*representada pelos retângulos verdes posicionados sob o lado direito do retângulo preto*) para tratar do cliente em questão.

- ☐ A conexão estabelecida inicialmente entre o servidor e um cliente é feita através de um socket (*estes sockets estão representados pelos círculos azuis e a conexão pela reta azul*).
- ☐ Sempre que é necessário realizar uma transferência de ficheiros cria uma nova thread e abre-se aí um novo socket entre o cliente e o servidor para tratar especificamente dessa tarefa (*neste caso os sockets estão representados pelos círculos roxos e a conexão pela reta roxa*).
- ☐ O servidor secundário troca pings com o servidor primário por UDP (*representado pela dupla seta vermelha*), sendo que para isto cada servidor tem uma thread focada nesta mesma tarefa (*representada pelos retângulos verdes ligados à dupla seta vermelha*).
- ☐ Para garantir a consistência dos ficheiros de texto presentes em ambos os servidores, existe uma thread no servidor primário (*representado pelo retângulo verde mais à esquerda*) que envia por UDP ao servidor secundário as alterações, atualizando-se assim o conteúdo do servidor secundário de modo a ser igual em ambos os lados (*o envio está representado pela seta castanha*).
- ☐ Cada servidor tem a si associada uma shared memory (*representada pelo hexágono amarelo*) que está disponível para ser usada pelas threads que esse servidor contém permitindo a comunicação sincronizada entre threads.

- <u>Funcionamento do servidor ucDrive</u>

1. **okServer()**: Verifica se estamos conectados ao servidor primário enviando true em caso afirmativo, ou false no caso contrário.

2. **checkUser()**: Assim que é feito o request de autenticação por um determinado utilizador, este método valida os dados enviando true no caso dos dados corresponderem a um utilizador que esteja registado no servidor, ou false se não existir nenhum utilizador que esteja registado com os dados fornecidos. De notar que o ficheiro que contém os dados de todos os utilizadores registados, clients.txt, segue a seguinte estrutura: username|password|/pasta1/pasta2/… (o campo que se segue após o último delimitador | representa a última diretoria em que esse utilizador esteve, antes de se desconectar).

3. **saveLastDir()**: Este método está associado às partes de mudar de password e também para guardar a secção de um utilizador assim que

este se desconecte para que da próxima vez que se volte a conectar seja redirecionado para o sítio onde ficou na última ligação.

3.1. **newPassword()**: Recolhe a nova password que foi enviada pelo cliente (que é atualizada no servidor com o auxílio da função do ponto 3) e envia-lhe true para confirmar que recebeu a password.

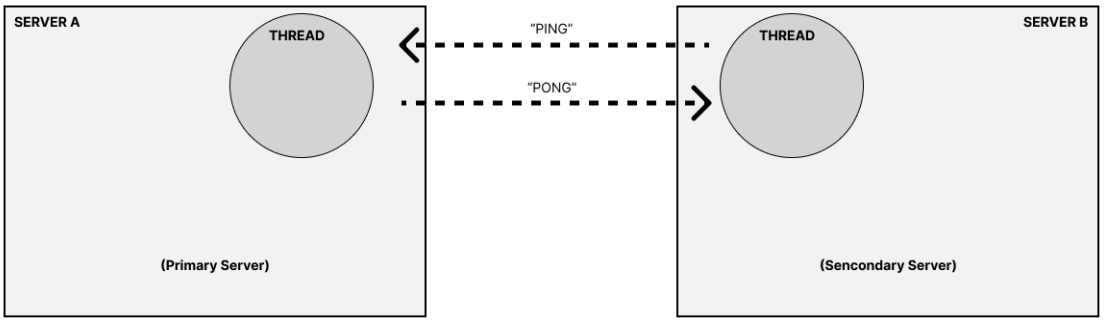
4. **changeDir()**: Constrói o path para a nova diretoria com base no nome que o utilizador passa, guardando a referência para a mesma no lado do servidor.

5. **listServerFiles()**: Com base na referência para a diretoria atual, guarda-se num JSONArray o conteúdo aí presente (pastas e ficheiros de texto), enviando de seguida ao cliente esse mesmo JSONArray que faz a diferenciação entre diretorias e ficheiros.

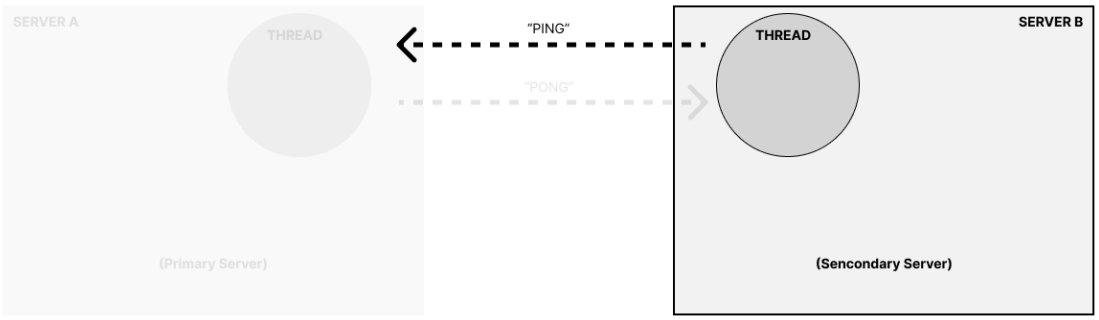
- <u>Mecanismo de Failover</u>

  Como é que o servidor secundário sabe que o primário crashou? Quando é que o servidor secundário pode aceitar clientes?

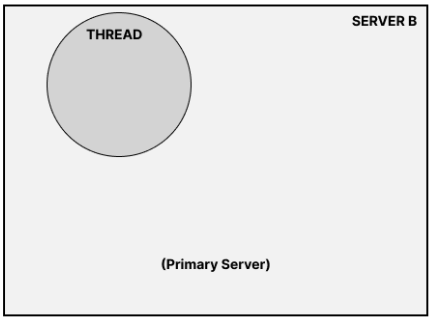
  O servidor A é iniciado primeiro, logo assume o papel de servidor principal e fica à escuta de “pings”, que irá receber assim que o servidor B for iniciado (servidor secundário). Assim que o servidor principal recebe um “ping”, ele devolve um “pong”, provando ao servidor secundário que está vivo.



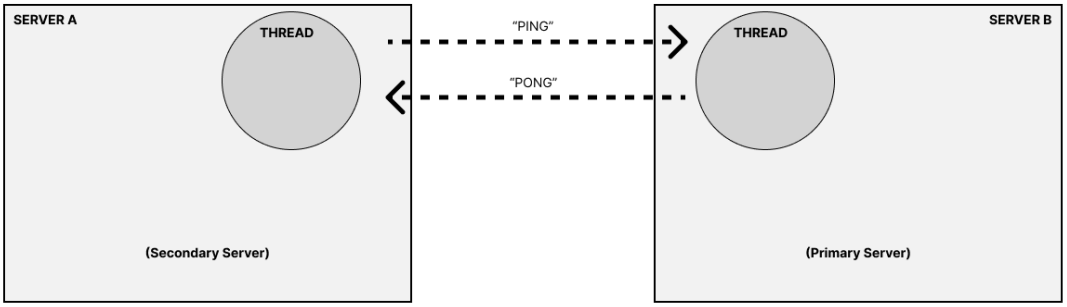
  Caso haja um crash no servidor principal, o secundário nota que não obtém resposta aos seus “pings”.



E após o tempo de timeout (10 segundos), ele considera que o anterior servidor principal morreu e vai assumir o seu papel. As threads que estão responsáveis pelo UDP e a comunicação entre os servidores são interrompidas e reiniciadas com as novas funções de servidor principal; já as threads que trabalham com o TCP, apenas é lhes dada permissão para aceitar e tratar de futuros clientes.



O servidor B, passa assim, a receber clientes e à espera que um novo servidor secundário lhe envie “pings” para este responder, invertendo-se os papéis.



- Mecanismo de sincronização

  Após um upload (cliente -> servidor), o servidor principal precisa de sincronizar esse ficheiro com o servidor secundário. Para isto, existe uma thread em cada servidor competente para tal (o servidor principal envia e o servidor secundário recebe).

Em cada upload e todas as vezes que um utilizador se desconecta, é adicionado a um ArrayList presente na Shared Memory um JSON Object com as seguintes informações: diretoria do ficheiro e nome do ficheiro.
A thread responsável pelo envio destes ficheiros para o servidor secundário vai verificando de 0,5 em 0,5 segundos se o ArrayList da Shared Memory referido anteriormente tem informações de algum ficheiro para seguir com destino ao servidor secundário, que apenas tem de se preocupar em guardar os novos ficheiros.

- Distribuição de tarefas

  O projeto foi realizado sempre com o conhecimento de ambas as partes, na medida em que se alguém implementava algo novo o outro elemento tomava conhecimento dessa ocorrência.
  De destacar que na modulação do projeto, o mecanismo de failover e no desenvolvimento das funcionalidades mais relacionadas com a transferência de ficheiros o Nuno esteve mais em cima, enquanto no que se refere ao layout do output do lado do cliente, nas restantes funcionalidades que envolviam nomeadamente mudanças dos dados do cliente e nos testes gerais à plataforma o João esteve mais por dentro.

- Testes feitos à plataforma

  Todos os testes feitos à plataforma e as respetivas descrições dos mesmos constam no ficheiro testes.xlsx (ficheiro excel que vem em conjunto no .zip desta entrega).