



Gonalo Vasconcelos Correia, n  2019216331, pdcorreia@student.dei.uc.pt

Jo o Garcia n  2019216375, uc2019216375@student.uc.pt

Sistemas Distribu dos

ucDriver: Reposit rios de ficheiros na UC

Projeto 1- Sockets & RMI - Turma PL 4

Coimbra, 2 de abril 2022

Índice

1. Descrição da arquitetura de software	3
2. Detalhes sobre o funcionamento do servidor ucDrive	4
3. Descrição do mecanismo de failover	5
4. Descrição dos testes feitos à plataforma.....	5
4.1 Requisitos	6

1. Descrição da arquitetura de software

Na figura que se segue, é assumido que o *TCPServer* é o servidor primário e o *BackUpServer* é o servidor secundário a título de exemplo.

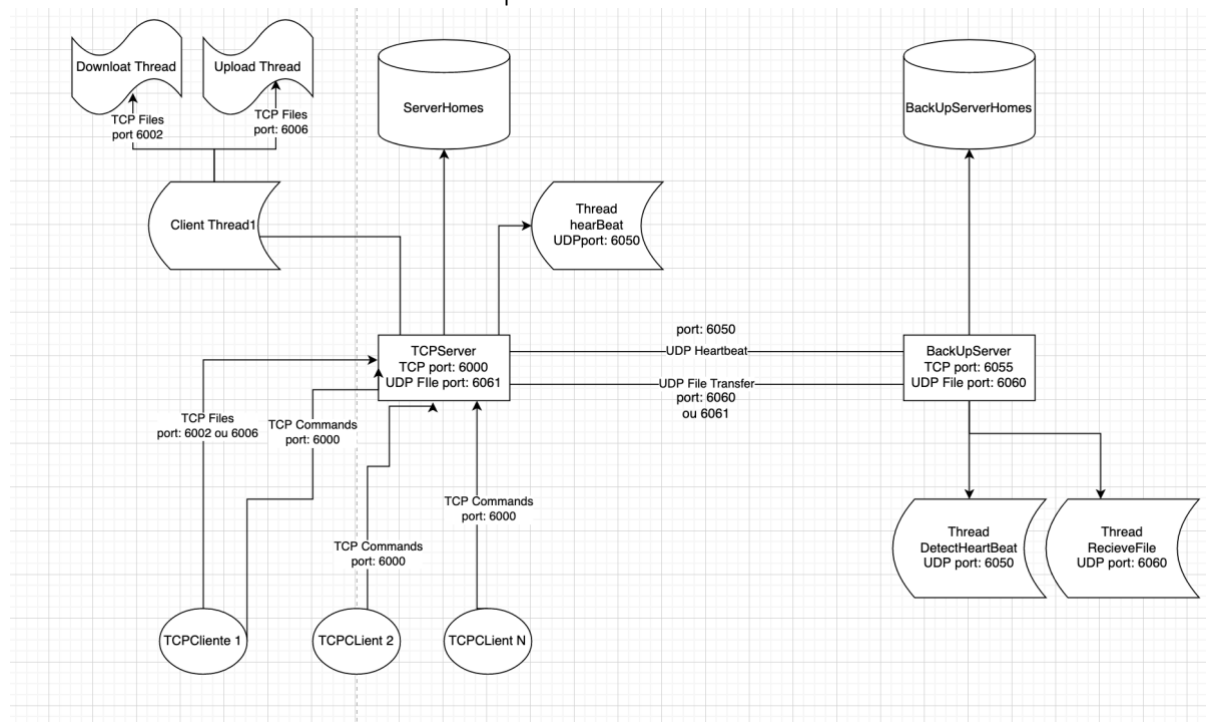


Fig 1- Exemplo do esquema geral do *software*

A partir da imagem podemos ver a arquitetura geral do software. De facto, o *TCPServer* ou *BackUpServer* quando é iniciado, começa por criar um *DatagramSocket* (no porto 6050), criando também uma *thread detectHeartBeat* que usa este socket, de modo a verificar se já existe algum servidor operacional. Paralelamente, também é iniciado outro *DatagramSocket* (no porto 6060 ou 6061), sendo criada uma *thread* que irá fazer uso deste *socket* para receber ficheiros vindos de um servidor Primário. Deste modo, se houver um servidor Primário em funcionamento, quando um outro servidor for iniciado, apenas irá ter o mecanismo de deteção de *heartbeats* e de receção de ficheiro, via UDP, ativos (não aceitado assim ligações de clientes).

Por outro lado, se o servidor chegar ao limite de deteção de *heartbeats* falhados, a *thread detectHeartBeat* acaba, sendo iniciada a *thread heartBeat* e aberto o *socket* TCP (no porto definido para o servidor, no caso da imagem 6000 ou 6055) que irá aceitar ligações de clientes. Deste modo, este servidor assume-se como primário, aceitando ligações via TCP de clientes, e enviando *pings/heartbeats* e ficheiros a um servidor secundário.

Quando um cliente é iniciado, é passado como o argumento o porto do servidor a que se pretende conectar. Assim, é criado um *socket* que se irá tentar conectar a esse porto. Se a ligação for bem-sucedida, o *socket* é criado e a conexão é estabelecida. Do lado do servidor, é criado uma nova *thread Connection* para o cliente, ficando o servidor paralelamente à escuta

de novas conexões de outros clientes. Quando um cliente pretende fazer um download, é criado um novo *socket* no servidor com o porto 6002 que irá ficar à escuta do novo *socket* também criado no lado do Cliente. Quando a conexão é feita, o servidor irá criar uma nova *thread* download que irá realizar as operações de download. Este processo é o mesmo para o *Upload*, apenas muda o porto que é o 6006. Após a efetuação da transferência pretendida, a *thread* termina e os *sockets* são fechados de ambos os lados.

Em suma, o cliente troca os comandos com o servidor Primário através de uma ligação TCP a partir do porto 6000 ou 6055 (dependendo se o servidor primário for o *TCPServer* ou o *BackUpServer*). Quando é iniciado um download ou upload a transferência de ficheiros é feita a partir um novo porto com novos *sockets* que já foram abordados anteriormente. Em paralelo com o uso da *thread heartBeat*, é feito o envio de *pings* para o servidor secundário. O servidor secundário recebe *pings* através da *thread detectHeartbeat* e em paralelo recebe ficheiros através da *thread recieveFiles*.

Observando o exemplo da figura 1, o *TCPServer* é o servidor primário e o *BackUpServer* assume o papel de servidor secundário. O *TCPServer* aceita ligações com vários clientes via TCP a partir do porto 6000. Cada conexão nova é uma nova *thread* no servidor. Em paralelo o TCP Server tem a *thread heartBeat* em funcionamento que envia *pings* para o servidor secundário via UDP a partir do porto 6050. Quando um utilizador pretende fazer um download é criada uma *thread* para esta transferência via TCP a partir do porto 6002. O *upload* tem o mesmo funcionamento, mas a partir do porto 6006. No entanto, após ser feito o *upload*, o ficheiro é enviado para o servidor secundário via UDP através do porto 6060. Neste exemplo, o *BackUpServer*, tem dois canais abertos com ligações UDP: um para receber ficheiros (porto 6060 OU 6061) e outro para receber *pings* (porto 6050) (não aceitando ligações de clientes).

2. Detalhes sobre o funcionamento do servidor ucDrive

O servidor *ucDrive*, no momento antes de abrir o porto TCP para aceitar ligações de clientes, lê o ficheiro de clientes registados, colocando toda a sua informação num *ArrayList* de clientes. Após este procedimento, encontra-se disponível para aceitar ligações de clientes. Quando um cliente inicia sessão envia o comando que escolheu para o servidor. Através deste comando o servidor saberá qual opção o cliente escolheu. O servidor para algumas das operações possui métodos que as solucionam:

- `public void listDirectory (int i)`

Este método recebe como parâmetro o índice do cliente da *thread*. Através deste índice, o servidor sabe a qual das *Homes* terá de aceder, e irá listar todos os ficheiros que encontrem dentro da diretoria atual em que o cliente se encontra no servidor.

- `public void changeDirectory (int i)`

Este método recebe como parâmetro o índice do cliente da *thread*. Através deste índice o servidor saberá a qual cliente terá de aceder. Estando no cliente, este método analisa qual a diretoria pretendida pelo utilizador e muda seja permitida essa operação. Caso o utilizador coloque “..” a diretoria volta uma para trás.

- `public int authenticate (String username, String password)`

Este método recebe como parâmetros o *username* e *password* que o cliente inseriu. O *arrayList* dos clientes irá ser percorrido, e se encontrar um cliente que contenha os dados corretos irá retornar -1 caso não encontre o cliente e "i" (o índice do cliente) caso o encontre.

- `public void send2Secondary (String filePath)`

Este método recebe como parâmetro o caminho no qual foi inserido o ficheiro no servidor primário. Este método permite enviar o ficheiro por UDP para o servidor secundário. Inicialmente, começa por ver o porto para o qual tem de enviar o ficheiro UDP, visto que o *TCPServer* tem o porto de receber ficheiros via UDP como 6061 e o *BackUPServer* como 6060. Após este passo, é enviado para o servidor secundário o caminho onde foi inserido o ficheiro e, posteriormente, é enviado o tamanho do ficheiro inserido. Assim, depois deste procedimento, são enviados pacotes com partes do ficheiro para o server secundário.

3. Descrição do mecanismo de failover



O mecanismo de *failover* foi feito com base nas classes *detectHeartbeats* e *heartBeat*. De facto, quando um server (*TCPServer* ou *BackUpServer*) é iniciado, ele vai começar por detetar se existe algum server Primário ativo. Caso esteja, irá receber os *heartbeats* provenientes deste, ficando a atuar como servidor secundário, ficando apenas apto para receção de ficheiros e *heartbeats* via UDP. Caso o servidor primário vá abaixo, do lado cliente, caso o servidor secundário ainda não se tenha assumido como primário, se este efetuar alguma operação irá a aparecer uma mensagem "Loading...". Enquanto isso, o servidor secundário vai detetando os *heartbeats* falhados, e quando chegar ao limite, irá criar um novo *heartbeat* e abrir uma ligação TCP de modo que os clientes se possam conectar a este. Assim, quando um cliente colocar algum comando, ser-lhe-á pedida uma nova autenticação e irá ser criada uma conexão com o servidor secundário, que passou a atuar como primário. Se o antigo servidor Primário voltar a estar ativo, este começará por tentar detetar *heartbeats*. Se detetar, irá assumir o papel de servidor secundário, estando apenas disponível para receber *pings* e ficheiros via UDP. Caso não detete *heartbeats* irá ter o fazer o mesmo processo descrito anteriormente.

4. Descrição dos testes feitos à plataforma




Tendo em conta os seguintes requisitos detalhámos os seguintes testes para cada um:

4.1 Requisitos



1. Estrutura de comunicação TCP e troca de mensagens


Testes	Desfecho Esperado	Pass/Fail
Envio de Comandos do Cliente para o Servidor	Comando são enviados e recebidos com sucesso	
Envio de Mensagens do Servidor para o Cliente	Mensagens são enviados e recebidos com sucesso	

2. Autenticar o do utilizador no terminal de cliente

Testes	Desfecho Esperado	Pass/Fail
Tentar entrar com Cliente já registado	Autenticação com Sucesso sendo apresentado menu de opções	
Tentar entrar com Cliente ainda não registado	Autenticação falhada, sendo pedido para tentar novamente	
Tentar entrar com Cliente registado, mas com palavra-passe errada	Autenticação falhada, sendo pedido para tentar novamente	

3. Alterar a password do utilizador




Testes	Desfecho Esperado	Pass/Fail
Alterar password para uma diferente da já existente	Password alterada na estrutura cliente e no ficheiro de utilizadores, desconectando o cliente sendo pedida nova autenticação, voltando à diretoria onde estava no Server	
Alterar a palavra-passe e depois mandar o servidor abaixo, sendo que o secundário passa para primário	Palavra-passe mudada com sucesso, e quando é pedida nova autenticação quando o secundário assume, já é a nova palavra-passe que está ativa	

Alterar password para a mesma	Enviar mensagem a informar que password é a mesma apresentado o menu novamente	
-------------------------------	--	---




4. Configurar endereços e portos de servidores primário e secundário

Funcionalidade não foi desenvolvida




5. Listar os ficheiros que existem na diretoria atual do servidor

Testes	Desfecho Esperado	Pass/Fail
Listar ficheiros da diretoria atual do servidor	Listar corretamente os ficheiros da diretoria atual	
Listar ficheiros da diretoria atual do servidor após lhe serem adicionados mais ficheiros	Listar corretamente os ficheiros da diretoria atual (incluindo os novos adicionados)	
Mudar diretoria e listar ficheiros dela	Listar corretamente os ficheiros da nova diretoria	




6. Mudar a diretoria atual do servidor

Testes	Desfecho Esperado	Pass/Fail
Mudar para a diretoria atrás	Muda com sucesso para a diretoria atrás, se chegar à página Home do cliente deve ser enviada mensagem a notificar o cliente de que não é possível ir mais para trás	
Mudar para uma diretoria correta	Deve ser enviada mensagem com nova diretoria atual do cliente no servidor	
Mudar para uma diretoria que não existe	Deve ser enviada mensagem a informar que a diretoria não existe	





7. Listar os ficheiros que existem na diretoria atual do cliente



Testes	Desfecho Esperado	Pass/Fail
Listar ficheiros da diretoria atual do cliente	Listar corretamente os ficheiros da diretoria atual	
Listar ficheiros da diretoria atual do servidor após lhe serem adicionados mais ficheiros	Listar corretamente os ficheiros da diretoria atual (incluindo os novos adicionados)	
Mudar diretoria e listar ficheiros dela	Listar corretamente os ficheiros da nova diretoria	

8. Mudar a diretoria atual do cliente







Testes	Desfecho Esperado	Pass/Fail
Mudar para a diretoria atrás	Muda com sucesso para a diretoria atrás, se chegar à diretoria última do cliente sendo dito que não é possível ir mais para trás	
Mudar para uma diretoria correta	Deve ser enviada mensagem com nova diretoria atual do cliente	
Mudar para uma diretoria que não existe	Deve ser enviada mensagem a informar que a diretoria não existe	

9. Descarregar um ficheiro do servidor




Testes	Desfecho Esperado	Pass/Fail
Descarregar um ficheiro que existe	Descarregar com sucesso e ser armazenado na diretoria atual do cliente	
Descarregar um ficheiro que não existe	Deve ser enviada mensagem a informar que ficheiro pretendido não existe	
Descarregar um ficheiro que já se encontra na diretoria do cliente	Deve ser enviada mensagem a informar que o ficheiro já se encontra descarregado	
Mudar de diretoria no server e descarregar ficheiro	Descarregar com sucesso e ser armazenado na diretoria atual do cliente	

Mudar a diretoria no cliente e descarregar um ficheiro	Descarregar com sucesso e ser armazenado na diretoria mudada cliente	
Mudar de diretoria no cliente e no servidor e descarregar um ficheiro	Descarregar com sucesso e ser armazenado na diretoria mudada do cliente	


10. Carregar um ficheiro para o servidor

Testes	Desfecho Esperado	Pass/Fail
Carregar um ficheiro que existe	Carregar com sucesso e ser armazenado na diretoria atual do cliente no servidor	
Carregar um ficheiro que não existe	Deve ser enviada mensagem a informar que ficheiro pretendido não existe	
Descarregar um ficheiro que já se encontra na diretoria do cliente no servidor	Deve ser enviada mensagem a informar que o ficheiro já se encontra carregado	
Mudar de diretoria no server e descarregar ficheiro	Carregado com sucesso e ser armazenado na diretoria atual do cliente no servidor	
Mudar a diretoria no cliente e descarregar um ficheiro	Carregar com sucesso e ser armazenado na diretoria mudada cliente no servidor	
Mudar de diretoria no cliente e do servidor e descarregar um ficheiro	Carregar com sucesso e ser armazenado na diretoria mudada do cliente no servidor	




11. Tratamento de exceções não tratadas nos outros pontos

Testes	Desfecho Esperado	Pass/Fail
Operação de Download ou Upload falhar a meio por algum motivo	Retry da operação que estava a ser feita e sem aparecer ficheiros duplicados	
Tentar colocar comandos enquanto nenhum server assumiu ser o Primário	Mensagem "Loading"	
Colocar comandos inválidos	Notificar cliente de que o comando é inválido	


12. Failover - Estrutura de comunicação UDP


Testes	Desfecho Esperado	Pass/Fail
Enviar mensagem por UDP de servidor para outro	Mensagem enviada com sucesso	

13. Failover - Heartbeat e deteção de falha no servidor primário


Testes	Desfecho Esperado	Pass/Fail
Servidor secundário recebe heartbeats enquanto o Primário está ativo	Servidor secundário deteta os heartbeats dando print da informação recebida	
Servidor secundário deteta heartbeats quando o Primário vai abaixo	Servidor secundário não deteta heartbeats incrementado o número de heartbeats falhado, assumindo-se como primário	
Enviar comandos para o servidor após o primário ir abaixo	Caso o servidor secundário já seja o primário, é pedido novo login ao cliente, sendo que o cliente se conecta ao novo servidor primário. Caso nenhum servidor esteja ativo deve receber mensagem "Loading"	

14. Ficheiros carregados para o primário são copiados para o secundário

Testes	Desfecho Esperado	Pass/Fail
Dar Upload de um ou mais ficheiros para o servidor primário	Ficheiro guardado com sucesso na diretoria do cliente certa no servidor, sendo que é guardado com sucesso no mesmo local na base de dados do servidor secundário	

Mandar o primário abaixo sendo que o secundário passa para primário e vice-versa	Ficheiro guardado com sucesso na diretoria do cliente certa no servidor, sendo que é guardado com sucesso no mesmo local na base de dados do servidor secundário	
--	--	---

15. Clientes conseguem operar com o secundário quando o primário está em baixo

Testes	Desfecho Esperado	Pass/Fail
Mandar o servidor primário abaixo	Secundário fica ativo sendo que é pedida nova autenticação ao cliente	
Mandar pelo menos o servidor primário (do momento) abaixo	Secundário fica ativo sendo que é pedida nova autenticação ao cliente	