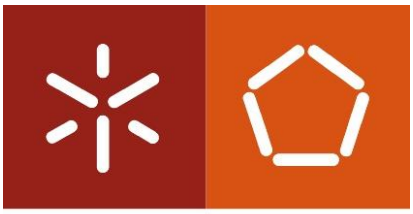

Universidade do Minho

Licenciatura em Engenharia Informática

Comunicação por Computadores - TP2

PL6 - Grupo 4

Gonçalo Ferreira (A93218) Alexandre Martins (A93315)
Diogo Rebelo (A93180)

Ano Lectivo 2021/2022

Capítulo 1

Introdução

Serve o presente relatório, como documentação de uma aplicação de sincronização de pastas, arquitectada e desenvolvida no âmbito da Unidade Curricular de Comunicações por Computador, como forma de avaliação da segunda fase da componente prática da UC.

Enquadrando-se no estudo dos protocolos de transporte, foi pedido ao grupo de trabalho, o desenvolvimento de uma aplicação capaz de sincronizar pastas entre computadores, e o desenho de um protocolo de transferência, que permita de forma fiável, a comunicação de dados entre ambos os nodos.

As principais funcionalidades implementadas na aplicação foram as seguintes:

-Sincronização de pastas e subpastas de ficheiros entre dois nodos, na mesma rede (sincroniza os ficheiros, cuja data da ultima modificação for a mais recente, de ambas as pastas)

-Envio de ficheiros concorrentemente e de forma limitada

-Envio de ficheiros sem restrições ao seu tamanho

-Criação de "logs" que documentam o processo de transferência bem como o tempo e velocidade da mesma, para cada ficheiro transferido

-Mecanismo de autenticação mútua

-Garantia da Integridade da transferência

-Servidor HTTP capaz de responder a pedidos GET, indicando o estado da aplicação

Capítulo 2

Arquitetura da solução

Para solucionar o problema de sincronização de pastas, entre dois sistemas diferentes, o grupo decidiu focar-se na utilização de pedidos de escrita. Para isto, foi necessária a produção de um protocolo, **FTRapid**, que é acompanhado pelo uso de **Threads**, de forma a promover a concorrência entre as transferências de ficheiros.

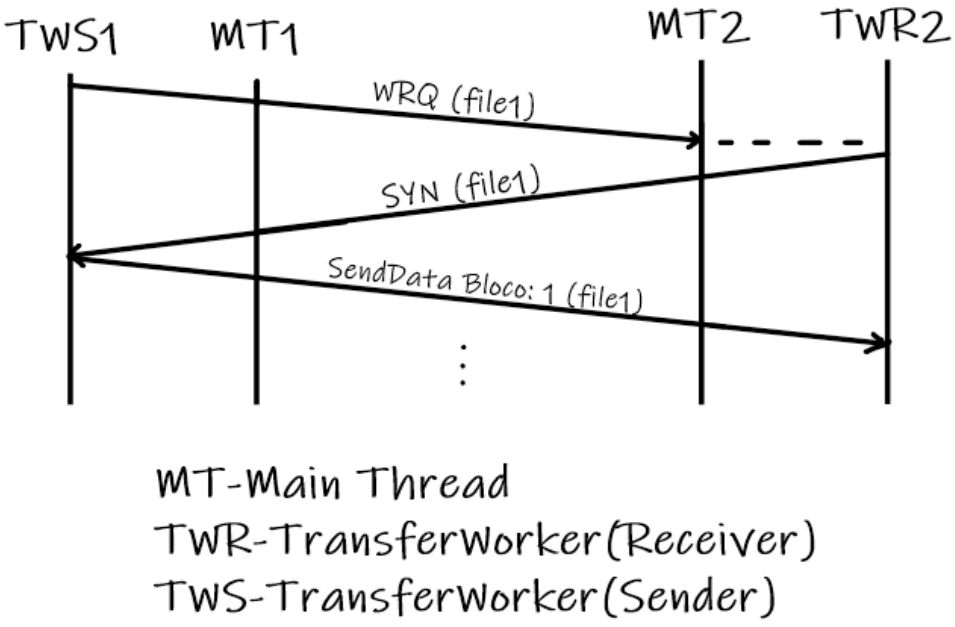

Figura 2.1: Diagrama de Sequência: Pedido de escrita

2.1 Threads

Para desenvolver a aplicação foram desenvolvidos quatro tipos de **Threads**:

2.1.1 FFSync - Main Thread

A Main Thread é responsável por executar diversas tarefas necessárias para o normal funcionamento da aplicação, sendo as seguintes dignas de serem mencionadas individualmente:

- **Teste de conexão:** Como o próprio nome indica, este fio de execução efetua um teste de conexão, verificando se é possível ou não, alcançar o sistema a contactar;
- **Autenticação:** Recorrendo ao protocolo desenvolvido, é executada uma simples autenticação. Ambos os clientes pedem a introdução de uma palavra-passe, que será enviada para o cliente a que se pretende conectar. Após serem recebidas, a autenticação é um sucesso caso correspondam às palavras-passe inseridas localmente;
- **Verificação dos ficheiros a enviar:** Para executar esta tarefa, os clientes começam por recolher o nome de todos os ficheiros, acompanhado da data da sua última modificação, presentes nas diretorias fornecidas. De seguida, é feita a troca das listas de ficheiros entre os clientes, que serão comparadas com as listas locais, de forma a definir os ficheiros que cada cliente tem de enviar.
- **Criação de Threads:** Após ser realizada a verificação dos ficheiros a enviar, são criadas as threads responsáveis por receber os pedidos de escrita (**ConnectionWorker**) e de estado (**HTTPRequestsAnswerer**), assim como as threads responsáveis por enviar os pedidos de escrita e posteriormente os ficheiros, caso o pedido seja aceite (**TransferWorker**).

2.1.2 ConnectionWorker

Este "trabalhador" utiliza a porta 9999 e é responsável por receber todos os pedidos de escrita. Numa versão mais elaborada, seria também encarregue de receber outros tipos de pedidos, assim como as respostas (negativas) a estes, e eventuais erros.

Após receber um pedido de escrita, ignora-o se for um duplicado de um pedido recebido anteriormente. No caso de não ser, cria uma *thread* (**TransferWorker**), que será incumbida de informar a aceitação do pedido, assim como de receber o ficheiro mencionado neste.

2.1.3 TransferWorker

Esta Thread pode executar duas funções diferentes:

- **Emissor:** Responsável por enviar o pedido de escrita de um ficheiro para a porta reservada ao envio destes. Após receber confirmação, passará a transmitir o ficheiro.
- **Recetor:** Responsável por informar que o pedido de escrita foi aceite, e posteriormente, é lhe confiada a recepção e construção do ficheiro.

2.1.4 HTTPRequestsAnswerer

Este "trabalhador" utiliza a porta 8000 e é responsável por atender pedidos HTTP do tipo "GET", devolvendo assim o estado em que o cliente se encontra, mais concretamente, os ficheiros que estão à espera de confirmação para ser enviados, os que estão a ser transmitidos, os que foram transferidos com sucesso e aqueles que possam ter sido interrompidos, seja por um erro de conexão ou por um erro da própria aplicação.

2.2 Classes Auxiliares

Nesta secção, serão apresentadas as classes que auxiliam as diversas **Threads**.

2.2.1 Status

Esta classe, como o próprio nome indica, é utilizada para gerir e representar o estado da aplicação. Dado que se trata de uma aplicação que gere a sincronização de ficheiros entre duas pastas, esta armazena diversas informações relativas às transferências, tais como: o nome dos ficheiros que precisam de ser enviados, e os pedidos que foram enviados e recebido.

2.2.2 SharedInfo

A presente classe serve como armazenamento dos diferentes parâmetros partilhados pelas *Threads* que constituem a aplicação, sendo estes:

- O estado de transferência dos ficheiros
- O caminho para a diretoria que se pretende sincronizar
- O endereço IP do cliente a sincronizar os ficheiros
- A porta utilizada para troca de pedidos
- Diversos locks, condições e variáveis(contadores) utilizados para controlar e limitar a concorrência[1]
- O objeto que permite a escrita para um ficheiro log

[1] A aplicação limita o número de *TransferWorkers*. Por predefinição, este limite é de 30 Threads para cada função(emissor/recetor). O limite pode ser alterado ao iniciar, fornecendo o número de Threads pretendidos para cada função, como o terceiro argumento, tendo em conta, que este número deve ser igual em ambos os clientes.

Capítulo 3

Especificação do protocolo

O protocolo rege todas as transferências de dados entre nodos da nossa aplicação. Este torna-se assim o pilar de toda a nossa implementação, sendo essencial que este se apresente como um protocolo resiliente a erros e que permita transferências, rápidas e eficazes, no seu uso do sistema.

Para conseguir atingir este objectivo, o grupo começou por definir diversos pacotes relevantes para as diferentes etapas de transferência, alguns dos quais acabaram por não ser utilizados na implementação final, mas o grupo considerou importante a sua inclusão, dado que, tornam o nosso protocolo mais completo, e resiliente a implementações futuras.

O protocolo de transferência foi baseado no método "Sliding Window: Selective Repeat", devido aos seus benefícios em termos de performance e eficiência de transferência. Todo este método será explicado, em mais detalhe, na secção **Interações: Transferência de Dados**.

3.1 Formato das mensagens protocolares

Essencial ao processo de transmissão protocolar é a definição de diferentes tipos de pacotes e as suas funções no contexto do protocolo. Na tabela abaixo, são apresentados os pacotes definidos pelo grupo, acompanhados pelos seus respetivos códigos de operação. De notar que todos os pacotes, possuem um hashcode próprio criado para garantir a integridade dos pacotes na chegada ao recetor.

Nas próximas secções serão desenvolvidas as funcionalidades de cada pacote em específico.

OPCODE	Operação
1	Pedido de leitura
2	Pedido de escrita
3	Pacote de dados
4	Pacote de Confirmação
5	Pacote de Erro
6	Pacote de Sincronização
7	Pacote de Autenticação

3.1.1 Pacotes Read Request e Write Request

Inicialmente o grupo, estabeleceu que seriam necessários pacotes de read request e write request, mas com a continuação do desenvolvimento a ideia de pacotes de read request foi abandonada. Ainda assim foi mantida a possibilidade de enviar read requests, devido a sua possível utilidade, em futuras expansões do projeto.

Quanto aos pacotes Write Request, estes são enviados pela porta principal da aplicação para o nodo externo, e contêm as informações relativas a um ficheiro que poderá vir a ser transferido, do nodo local para o externo. Assim, este pacote é formado pela porta que será usada pelo nodo local para comunicar os dados, pela data da ultima alteração do ficheiro, pelo nome do ficheiro a ser transferido, e por fim o código de hash gerado pela combinação dos campos anteriores. Quando o nodo externo receber este pacote, a informação nele contida, será analisada de forma, a compreender se a transferência será mesmo necessária.

1B	2B	8B	NB	1B	4B
opcode	porta	data da última mudança	file name	0	hash code

3.1.2 Pacotes Data

Os pacotes de Data, são os pacotes responsáveis pela transmissão de dados no protocolo FT-Rapid. Estes são constituídos pelo número do bloco, o tamanho deste bloco em bytes, os dados e o hashcode.

1B	2B	2B	NB<1015B	4B
opcode	nº de bloco	tamanho	dados	hash code

Com estes campos, podemos esperar que cada transmissão, de um conjunto de pacotes que esgote o número de blocos (são possíveis 32,768 blocos), e com cada pacote transportando no máximo 1015 bytes de dados, podemos esperar que um ciclo de transferência será capaz de transportar aproximadamente 32MB de dados.

3.1.3 Pacotes ACK

Os pacotes ACK, representam sinais de reconhecimento de mensagem, ou seja, servem de resposta ao envio de pacotes de dados. São apenas constituídos pelo seu "opcode", pelo nº do bloco recebido e pelo hashcode.

1B	2B	4B
opcode	nº do bloco	hash code

3.1.4 Pacotes Error

Os pacotes Error servem para indicar que algum erro ocorreu na ligação. Muitas vezes se o erro for de conexão, servem apenas de cortesia.

Estes apenas contêm o seu "opcode", "código de erro"e "hashcode".

1B	2B	NB	1B	4B
opcode	error code	file name	0	hashcode

3.1.5 Pacotes SYN

Os pacotes SYN servem de resposta a pedidos quer de escrita e leitura, indicando para onde deve o cliente que deu request esperar ligação com este cliente. Para além do seu opcode, contêm a porta para onde comunicar e o hashcode.

1B	2B	NB	1B	4B
opcode	nº da porta	file name	0	hash code

3.1.6 Pacotes AUT

O pacote de Authentication serve como uma forma simples de autenticar uma sessão de transferência, enviando para ambos os nodos, a palavra-passe introduzida pelo utilizador, em ambos os nodos, de forma a comparar se esta é igual. Apenas contêm o opcode, a palavra-passe e o hashcode.

1B	NB	1B	4B
opcode	password	0	hashcode

3.2 Interações

3.2.1 Transferência de Dados

Como já mencionamos, o modelo para o protocolo escolhido, foi o "*Sliding Window Protocol: Selective Repeat*". Inicialmente o grupo, implementou uma versão do modelo "Stop and Wait", mas depois de alguns testes, decidimos redesenhar o protocolo de forma a tornar a transferência mais rápida e eficiente. Assim, acabamos por escolher o protocolo "Selective Repeat", devido a velocidade e fiabilidade por este fornecida.

Para melhor demonstrar o funcionamento do nosso protocolo, o grupo decidiu incluir o diagrama abaixo:

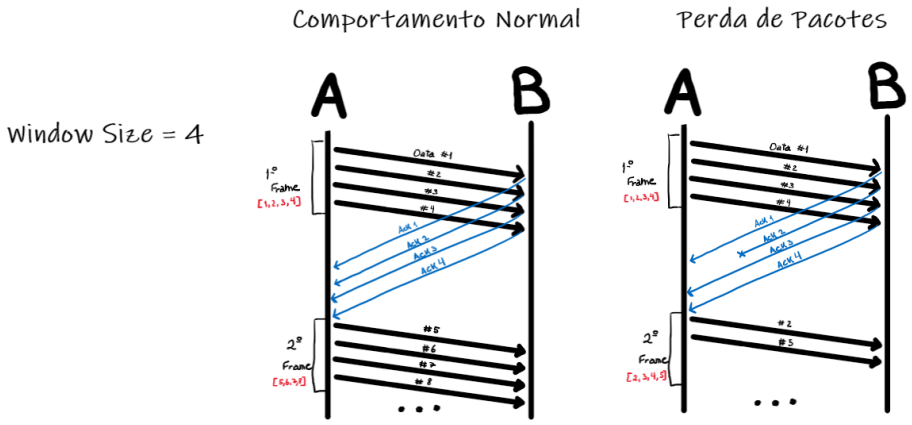

Figura 3.1: Sliding Window Protocol: Selective Repeat

Alguns aspetos relevantes sobre esta solução:

- O "WindowSize" é predefinido em ambos os nodos para 25, ou seja, são enviados a cada ciclo 25 pacotes de dados.
- Cada pacote de dados enviado deve ser respondido por um ack com o mesmo número do bloco.
- Caso o envio de um bloco de dados, não tenha sido confirmado, quer por falha no envio dos dados, quer por falha no envio da confirmação, na próxima iteração, esses dados serão reenviados na próxima iteração. Isto é garantido pela janela, que irá mover um pacote de dados (cuja confirmação ainda não tenha sido recebida) para a primeira posição da janela.
- Apenas serão enviados todos os pacotes de um frame, cuja confirmação ainda não tenha sido recebida
- Caso aconteça duplicação de pacotes, o protocolo irá ignorar ACKs, já recebidos, e irá reenviar ACKs no caso de receber um pacote de dados repetido.
- A transferência de pacotes termina, do lado do emissor, assim que é enviado um pacote com tamanho inferior ao tamanho máximo estabelecido ou que o limite máximo de pacotes seja atingido. Para além disto, o emissor deverá garantir que por todos os pacotes enviados, foram recebidos os ACKs correspondentes.
- Do lado do recetor, quando é recebido um pacote de dados com o tamanho inferior ao máximo é sinalizado o final. Antes de terminar, o recetor deverá ainda: verificar se todos os pacotes de dados anteriores ao último foram recebidos e esperar por eventuais pacotes de dados repetidos, que podem sinalizar a perda de um ACK.

Capítulo 4

Testes e resultados

Nesta secção apresentaremos os resultados, a nível de performance, da nossa aplicação de acordo com os testes

- Serão usados os ficheiros fornecidos pelos docentes
- Os testes serão realizados na máquina virtual utilizada nas aulas práticas da UC
- Os testes serão realizados na topologia fornecida pelos docentes e serão transferências realizadas do nodo "Portatil1"para o nodo "Orca"(com uma ligação normal) e do nodo "Portatil1"para o nodo "Grilo"
- Os tempos apresentados são medidos pela aplicação, e correspondem ao tempo decorrido desde o inicio da transferência até ao final da execução da transferência.
- Os tempos e velocidades serão medidos 5 vezes e os resultados apresentados serão a média destes tempos, Tm (tempo médio) em s (segundos) e Vm (velocidade média) em bps (bits por segundo)
- O tamanho da janela do protocolo será de 25

Ficheiros	Portatil1->Orca	Portatil1->Grilo
TP2-folder1.zip	Tm = 0.1987 s	Tm = 0.2744 s
	Vm = 9484017 bps	Vm = 6883117 bps
TP2-folder2.zip	Tm = 3.7842 s	Tm = 5.5467 s
	Vm =12230545 bps	Vm = 8318843 bps
TP2-foolder3.zip	Tm =19.1317s	Tm = 32.4499 s
	Vm = 16214434 bps	Vm = 9587365 bps

De realçar também que todos os cenários de teste sugeridos pela equipa docente, foram realizados com sucesso:

Teste1 : 0.1941s e 9696357 bps

Teste2 : 0.2574 e 7304736 bps

Teste3: 3.9949s

Teste4: 16.5000s

Capítulo 5

Conclusões e trabalho futuro

Este projeto foi encarado pelo grupo, como o desenvolvimento de uma aplicação que poderia vir a ser genuinamente útil, para membros do nosso grupo, graças a facilidade de sincronização de pastas permitida por este projeto, principalmente para quem tem dois computadores e quer trabalhar nos dois.

Tendo isto em mente, concluímos ter alcançado todos os objetivos propostos e estar presentes, diante de uma aplicação, que acima de tudo é resiliente contra erros, rápida e eficiente nas suas transferências.