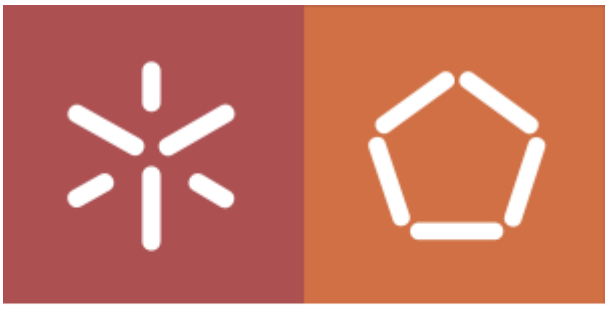


Universidade do Minho
Escola de Engenharia

Licenciatura em Engenharia Informática

Comunicações por Computador

a100604, Afonso Oliveira e Silva
a97455, Henrique Nuno Marinho Malheiro
a97678, Ema Maria Monteiro Martins

Trabalho 2: Transferência rápida e fiável de múltiplos servidores em simultâneo

Dezembro 2023

Índice

Introdução:

Com este trabalho pretende-se que vários Nodes comuniquem entre si para a partilha de ficheiros ou partes dos mesmos através de ligações UDP. Para isso, cada Node informa o Tracker, através de ligações TCP, sobre os ficheiros que possui, disponibilizando-os para todos os Nodes do sistema, quando esse ficheiro for solicitado por outro Node.

DNS:

Define-se a zona de dns “cc” no “named.conf”, para que seja possível traduzir os diversos nomes dos Nodes nos seus respectivos hosts.

Ficheiros Locais:

Cada Node tem uma pasta de ficheiros locais, sendo a mesma constituída por ficheiros (quando tem unicamente o ficheiro completo) e por pastas com o nome do ficheiro (quando tem o ficheiro partido em blocos). Essa pasta com o nome do ficheiro, tem por sua vez uma pasta “blocks” com os diversos blocos, e terá o ficheiro completo quando tiver adquirido todos os blocos e feito a sua reconstituição.

Blocos:

Definiu-se que todos os blocos dos diversos ficheiros teriam 128 bytes (podendo o último ser menor). Esse tamanho permite garantir que as mensagens que transportem os blocos não vão ser divididas nos sockets UDP.

Tracker:

Inicializa-se o Tracker, onde contém duas estruturas de dados que irão guardar a informação sobre cada ficheiro e cada bloco desse ficheiro, quais Nodes o têm localmente e também para cada ficheiro quantos blocos ele tem.

O Tracker vai também criar um socket TCP na sua porta 9090, ficando à escuta de pedidos de conexão dos Nodes, criando uma thread para atender cada Node concorrentemente, mantendo-a ativa até esse Node se desconectar.

Caso o Tracker se desconecte, todas as ligações aos diversos nodes serão terminadas.

Node:

Quando se inicia um Node, o mesmo vai receber a diretoria onde estão guardados os seus ficheiros localmente, bem como o nome e porta do Tracker a que se quer conectar, onde o DNS irá traduzir para o host do Tracker, enviando-lhe um “startConnection”, recebendo e guardando o seu host como resposta.

Após, envia ao Tracker as estruturas com as informações sobre quais ficheiros e blocos tem localmente, bem como quantos blocos os ficheiros têm (só o consegue fazer para todos os ficheiros que contém completos).

De seguida, cria uma thread que irá correr o Transfer, ficando a thread principal a correr o modo interativo. Esse modo é responsável por ficar continuamente à espera de indicações do utilizador de que ficheiro pretende adquirir do sistema ou se pretende terminar a conexão.

Após o pedido do utilizador de um determinado ficheiro, o Node irá solicitar ao Tracker quais os Nodes do sistema que têm blocos deste ficheiro, bem como o número total de blocos por qual o ficheiro é composto.

Recebendo essa informação, vai escolhendo de forma aleatória a quem pedir os determinados blocos que necessita, criando uma thread para cada pedido (pedidos de blocos de maneira concorrente). Passados 5 segundos, caso esse bloco ainda não esteja presente localmente, solicita novamente o bloco, repetindo este processo até ter o bloco localmente.

Caso receba uma solicitação do utilizador para sair, termina a ligação ao socket do Tracker, bem como todas as threads criadas.

Transfer:

O Transfer vai constantemente ler do socket UDP e guarda essas mensagens num buffer. Na leitura desse buffer podemos ter duas situações distintas.

Caso se trate de um pedido de um bloco de outro Node, o mesmo irá criar um thread, que o irá enviar para o Node que o solicitou.

Por outro lado, se for um bloco que tenha pedido previamente, vai similarmente criar um thread para atender esse bloco, que o irá guardar localmente, bem como avisar o Tracker que tem esse bloco, permitindo assim que outros Nodes no futuro possam-no solicitar. Caso este bloco for o último que faltava localmente deste ficheiro, irá proceder à reconstituição do ficheiro, guardando-o.

Definição de mensagens do Tracker:

As mensagens trocadas entre o Tracker e o Node encontram-se no ficheiro tracker_protocol_mensage.py.

Cabeçalho (20bytes):

O cabeçalho é composto por uma codificação de mensagem a realizar (3bytes), um separador, “|” (1 byte) e o tamanho da sequência de dados (16 bytes).

Corpo da mensagem:

O corpo da mensagem é sempre um dicionário que nele pode conter mais dicionários. O seu conteúdo depende do tipo de mensagem que vamos mandar:

000(startConnection): Dicionário que manda a informação relativa ao nome do Node.

001(sendDictsFiles): Dicionário que manda um dicionário com os ficheiros que o node contém em blocos e outro com os ficheiros que contêm completos (contém informação sobre o número de blocos dos seus ficheiros).

010(getFile): Dicionário com o nome do ficheiro que pretendemos pedir.

110(newBlockLocaly): Dicionário com o nome do ficheiro e respetivo bloco que queremos informar ao Tracker que este node recebeu.

100(endConnection): Dicionário com o nome do nó a terminar conexão.

Definição de mensagens do Transfer:

As mensagens trocadas entre o Transfer e o Node encontram-se no ficheiro transfer_protocol_mensage.py.

Cabeçalho (18 bytes):

O cabeçalho é composto por uma codificação de mensagem a realizar (1 byte), um separador, "|" (1 byte) e o tamanho da sequência de dados (16 bytes).

Corpo da mensagem:

O corpo da mensagem é sempre um dicionário que nele pode conter mais dicionários. O seu conteúdo depende do tipo de mensagem que vamos mandar:

0(get_block): Dicionário com o cliente a quem vamos pedir o bloco, o ficheiro que vamos pedir, o bloco deste ficheiro que vamos pedir e o número de blocos desse ficheiro.

1(send_block): Dicionário com o bloco que vamos mandar, o tamanho do bloco, o nome do ficheiro ao qual o bloco pertence, o número de blocos desse ficheiro e o checksum.

Como tratar as mensagens:

Todas as mensagens têm um cabeçalho fixo. Quando recebem uma mensagem, começam por ler o cabeçalho pois este tem o tamanho da mensagem a ler. Esta contém uma sequência inicial de 3 bytes no caso do Tracker e 1 byte no caso do Transfer, que codificam a função a ser executada e o tamanho do bloco de mensagem a ler. Em seguida, leem tantos bytes quantos os especificados no parâmetro de tamanho de mensagem que é passado no cabeçalho.

Tracker:

O Tracker comporta-se de maneiras distintas para cada código de mensagem recebido:

000(startConnection): É adicionado o nome do Node a uma lista, listNodes, que contém todos os Nodes conectados ao Tracker. Em seguida é mandado uma mensagem ao Node a dizer que a conexão foi realizada com sucesso.

001(sendDictsFiles): O Tracker recebe dois dicionários. O primeiro associa os ficheiros que o Node contém a uma lista com os blocos que ele contém desse ficheiro. Faz uma inversão da estrutura de dados pois guarda a informação num dicionário, dict_filename_dictBlockListNode, em que as chaves são os ficheiros e em que os valores são dicionários em que se faz corresponder os blocos a listas de nodos que contém esses blocos.
O segundo dicionário que recebe associa os ficheiros ao número de blocos que este contém. Esse cálculo é determinado, dividindo o ficheiro pelo tamanho de bloco que definimos como sendo 128 bytes. É assim adicionado uma instância ao um dicionário dict_filename_numBlocks, que associa nomes de ficheiros ao número de blocos que o constituem o ficheiro.

010(getFile): Verifica se alguém tem o ficheiro pretendido. Se ninguém o tiver é enviada uma mensagem de erro. Caso contrário, é enviada uma mensagem que diz que Nodes é que têm cada parte do ficheiro.

110(newBlockLocaly): Quando recebe esta mensagem, o Tracker sabe que o nodo que a enviou recebeu um bloco, pelo que tem de atualizar o seu dicionário, dict_filename_dictBlockListNode.

100(endConnection): O Tracker começa por retirar o nodo da listNodes. Em seguida, remova todas as entradas no dicionário dict_filename_dictBlockListNode que sejam relativas ao nó que pretende terminar a conexão. Por último, fecha o socket.

Transfer:

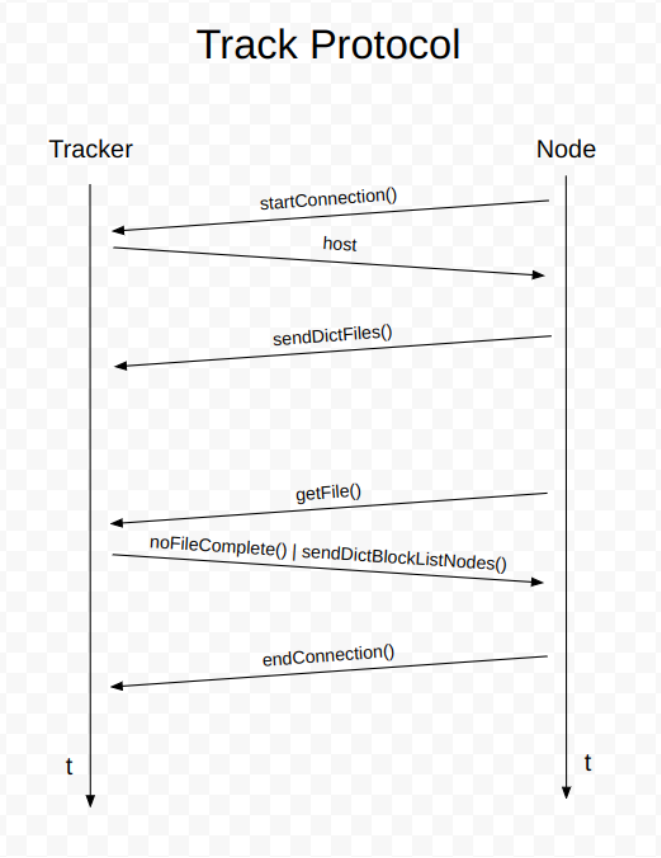
O Transfer comporta-se de maneiras distintas para cada código de mensagem recebido:

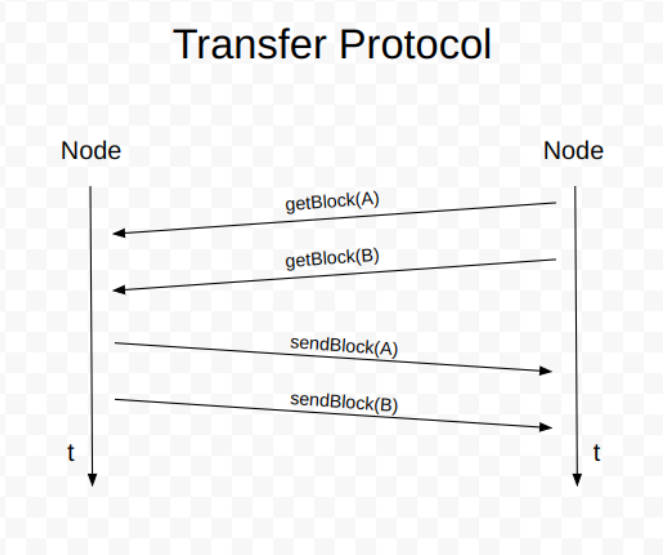
0(get_block): Cria uma thread que vai enviar o bloco de ficheiro pretendido numa mensagem do tipo send_block. Primeiro, verifica se contém o bloco e manda o

bloco em caso afirmativo. Se não o tiver, parte o ficheiro inteiro e manda o bloco pretendido.

1(send_block): Se receber uma mensagem deste tipo, primeiro verifica o checksum. Se este estiver correto, então procede a guardar o bloco. Se este tiver todos os blocos que contém o ficheiro, então procede-se à reconstrução do ficheiro completo.

Diagramas temporais:





Conclusão:

Este trabalho permitiu-nos entender melhor as diferenças dos protocolos TCP e UDP, bem como o funcionamento do DNS. Além disso permitiu-nos distinguir melhor as diferenças entre a camada de transporte e a camada aplicação. Por outro lado, permitiu-nos também desenvolver um melhor conhecimento sobre os cuidados a ter no desenvolvimento de um protocolo, nomeadamente a definição de cabeçalhos, o tamanho das mensagens, controlo de erros, reenvio de pacotes perdidos ou corrompidos.