Instituto Superior Técnico

Programação de Sistemas

Relatório do Projeto

João Rodrigues (81843), Sara Vieira (80935)

Grupo 20

supervised by
João Silva

02/06/17

Conteúdo

1 Introdução

O projeto da cadeira de Programação de sistemas visa criar um conjunto de programas que simulem um sistema de partilha de fotos entre *clients* e *peers*, por meio de uma *gateway*. Estes diversos componentes deverão ser capazes de comunicar entre si e implementar um série de funções relativas ao processo de *upload* de fotos e respectivo processamento de informação, sendo que todos os *peers* deverão reter e estar atualizados com toda a informação relativa às fotos adicionadas ao sistemas pelos diversos clientes.

2 Funcionamento e Estrutura Geral

2.1 Arquitectura

O projecto é constituído por três tipos de componentes diferentes: *gateway*(apenas uma), *clients* e *peers*. Estes três componentes interagem entre si e cada um executa um determinado conjunto de tarefas, que explicaremos adiante.

Os componentes comunicam entre si através de sockets, sendo que quando um *client* ou um *peer* é executado, é-lhe fornecido o endereço IP da rede onde a *gateway* é executada. Assim sendo, um *client* começa por se conectar à *gateway* (1), que lhe fornecerá o endereço de um *peer* disponível para se conectar.

Após estabelecida a conecção entre *client* e *peer*(2), o primeiro está apto a enviar fotos para o sistema e organizá-las no mesmo. O *client* informa o *peer* sobre que ação deseja executar e, consoante esta, o peer processa a informação que recebe posteriormente e/ou reencaminha um conjunto de pedidos, ações e informações para a *gateway*(3), nomeadamente quando é necessário replicar/atualizar as informações das fotos actualmente guardadas em todos os *peers* do sistema(4).

A *gateway* é também responsável por verificar o funcionamento de todos os *peers*(A).

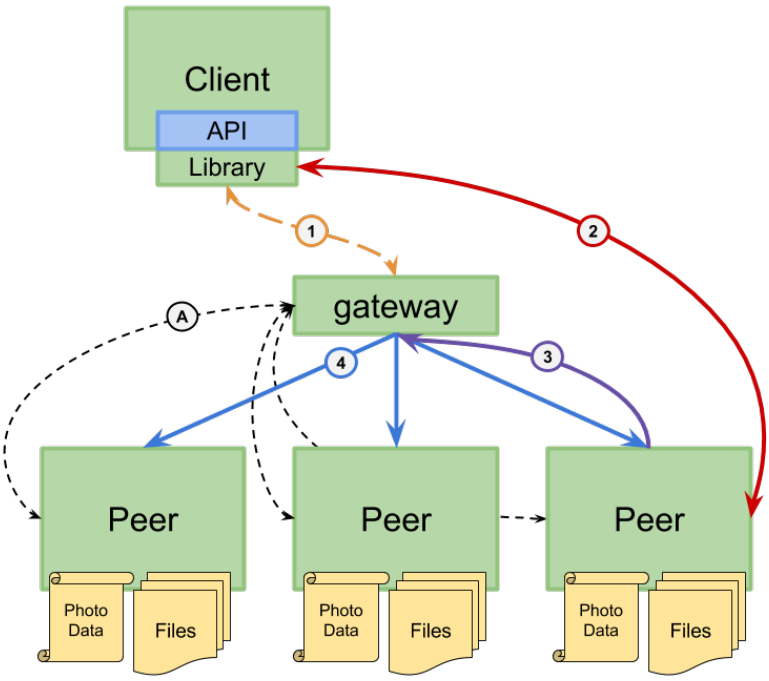

Figura 1: Arquitectura Geral do Projecto

2.2 Componentes

Um dos componentes deste projecto trata da *gateway*, responsável por garantir a gestão de dados do sistema, sendo que a ela está associada uma lista dos *peers* ativos no programa. A partir desta informação a gateway fornece os dados necessários para um *client* se conectar a um dos *peers* (endereço e porto) e gere toda a manipulação de informação relativa às fotos adicionadas ao sistema (garantindo, por exemplo, a replicação de uma foto para as listas de todos os *peers*). Na *gateway* é também verificado o funcionamento de todos os *peers*, através da função *machine_that_goes_ping* e gerado um id para cada nova foto através de um simples contador.

O *client* é a interface através da qual se vai introduzir toda a informação relativa a uma foto. Inicialmente, o *client* contacta com a *gateway* para esta lhe fornecer dados de um *peer* para com ele se conectar. Após estabelecida a conecção, o *client* está apto a enviar as fotos e respectiva informação para o *peer* ao qual está conectado.

Por sua vez, os *peers* tem a função de armazenar cada um a informação relativa às fotos recebidas no total do programa, usufruindo para isso de uma lista a eles associada onde será guardada toda essa informação. Sempre que um *peer* receba uma informação de um dos seus *clients* que implique a manipulação da informação actual sobre as fotos guardadas, este

contacta a *gateway* que tratará de gerir toda a replicação/sincronização dessa informação para os restantes *peers*. Caso o *client* requeira apenas alguma informação sobre fotos já armazenadas no sistema, o *peer* limita-se a devolver-lhe a informação requisitada.

2.3 Threads

2.3.1 gateway

Na *gateway* estão presentes 3 tipos de *threads*.

O **primeiro tipo** é denominado *thrd_server_fnc*, e é responsável pelas comunicações com um denominado *peer*, utilizando uma *socket stream*. Por cada *peer* aceite, uma *thread* deste tipo é inicializada.

O **segundo tipo** corre a função *thrd_client_fnc* e é responsável pela comunicação com o *client* através da única *socket datagram*, disponível para todos os clientes.

O **terceiro tipo**, *machine_that_goes_ping* trata de ver se os *peers* ainda se encontram ativos, e é também criada uma por cada *peer*, desta vez utilizando uma *socket datagram* designada para aquele *peer* em específico.

2.3.2 peers

Em cada peer estão presentes dois tipos de threads, muito parecidas com as utilizadas na gateway.

O **primeiro tipo**, *handle_client*, trata da comunicação com o *client* e determina se tem que enviar informação à *gateway* para replicação ou pode responder diretamente ao *client*. Existem tantas *threads* deste tipo quantos *clients* estiverem designados ao *peer*.

O **segundo tipo**, *gw_connection*, trata da comunicação com a *gateway*, através da *socket stream*. Existe apenas uma *thread* deste tipo por cada *peer*.

O **terceiro tipo**, *sync_fnc*, é responsável por responder ao *ping* da *gateway*, usando para isso uma *socket datagram*. Existe também 1 *thread* deste tipo por cada *peer*.

3 Partilha de Informação entre Componentes

3.1 Estruturas

No total, temos três tipos de estruturas, sendo que duas delas servem para armazenar informação sobre *peers* e fotos.

Primeiramente, temos uma estrutura **message**, que tem uma utilização mais geral no código. Resumidamente, recorremos a esta estrutura para o envio de informações simples,

tal como um inteiro(tipo de ação a realizar), uma *string*(nome de ficheiros ou *keywords*), portos ou pequenas informações para formar uma *socket*.

Na estrutura **server_struct** está guardada toda a informação relativa a um *peer*; logo esta estrutura fará parte da lista de *peers*. Nela estará armazenada informação relativa ao endereço do *peer*, endereço das *sockets* a ele associadas e número de *clients* que está a processar. Esta estrutura é também utilizada para envio de dados relativos a *sockets*.

No caso do **photo_struct**, será armazenada toda a informação relativa a uma foto, nomeadamente o nome do ficheiro, o id atribuido pela *gateway*, vector de *keywords* a ela associadas e a quantidade destas, o tamanho do ficheiro e informação sobre o *client* que a adicionou ou sistema. Mais uma vez, esta estrutura integra a lista de fotos presente em cada *peer* e é por vezes usada para envio de informação relativa a uma foto via *sockets*.

3.2 Sockets

Várias sockets são utilizadas no funcionamento do repositório. Aqui explicamos a sua função.

3.2.1 client-gateway

Entre os vários *clients* e a *gateway* existe uma *socket datagram*, que os *clients* usam para receber o endereço do *peer* que vai ser responsável por cada um deles.

3.2.2 client-peer

Entre cada *client* e o *peer* corresponde existe uma *socket stream*, utilizada para todas as comunicações entre *client* e *peer*, como transferência de ficheiros ou quaisquer outros dados relacionados com as fotos.

3.2.3 peer-gateway

Por cada *peer* em funcionamento existem duas *sockets* que o ligam à *gateway*. A primeira é uma *socket stream*, responsável pela transferência de ficheiros e dados relacionados com as fotos presentes no repositório. A segunda é uma *socket datagram*, responsável pelo *ping* que a *gateway* vai fazer a cada *peer* para determinar se este ainda se encontra ativo.

3.3 Sincronização

Para além do uso de *threads*, a sincronização foi conseguida com a utilização de *mutexes*. Na sua maioria, os *mutexes* foram aplicados em funções do código que envolviam a manipulação/consulta da informação nas listas de *peers* e fotos.

Assim, no inicio destas *threads* foi inicializado *mutex* e colocados *locks* e *unlocks* no inicio e final(respectivamente) de cada função chamada que requerisse a estagnação dos dados dessa mesma lista para recolha ou atualização de informação. No caso das *threads thr_client_fnc* e

thr_server_fnc, os *mutexes* incidem sobre as funções que acedem a lista de *peers* na *gateway*. Nas *threads gw_connection* e *handle_client*, os *mutexes* foram criados para salvaguardar a manipulação das listas de fotos de cada *peer*, nas funções que as acedessem.

Foi também criado um *mutex* para garantir que não haveria interferência de mensagens entre *clients* e *gateway* aquando do envio de uma foto para o sistema.

4 Replicação de Dados

A garantia da replicação correta de dados é dada pela *gateway*, que envia a informação a todos os *peers* sempre pela mesma ordem, de acordo com a sua posição na lista. A informação chega à *gateway* através da *socket stream* do *peer* contactado por um *client*, é processada, se necessário, e depois enviada por ordem a todos os *peers* presentes na lista.

5 *machine_that_goes_ping*

Na *gateway* e em cada *peer* ligado a esta, existe uma *thread* responsável apenas por enviar e receber um sinal, de maneira a que a *gateway* possa determinar quais os *peers* que se encontram ativos.

Tendo em conta o objetivo, a *gateway* irá percorrer a lista de *peers*, enviando um sinal e esperando um certo tempo pela sua resposta (usou-se um recv com a *flag* MSG_DONTWAIT). Caso não receba resposta do *peer*, retira-lhe uma vida (cada *peer* na lista tem à partida três vidas). Quando as vidas de um certo *peer* chegam a 0, este é retirado da lista, e não será atribuído a nenhum *client*.

6 Conclusão

Se chegou até aqui está de parabéns. Gostámos muito de fazer este projeto, foi bastante interessante e desafiante, dada a complexidade da comunicação entre diferentes processos e a necessidade de serem utilizadas diferentes sockets e threads.