Universidade do Minho
Mestrado Integrado em Engenharia Informática
Sistemas Distribuídos
2015/2016

Grupo 77

João Bernardo Machado Quintas Dias da Costa A70430

Rafael Alexandre Antunes Barbosa A71580

Rui Jorge Cerqueira Soares A71240

Tiago Alexandre Fernandes Lima A70672

Introdução

Neste trabalho foi-nos proposta a criação de um serviço de gestão de deslocações baseado em conecções entre um servidor e diversos clientes. Em todo o processo de desenvolvimento do dito serviço, esteve sempre na mente dos elementos do grupo os conceitos apreendidos nas aulas teóricas e práticas de Sistemas Distríbuidos, onde utilizamos mecanismos para regular o acesso a memória partilhada por processos e linhas de execução concorrentes.

Apesar do conhecimento destes mecanismos, no entanto, foi sempre objectivo do grupo minimizar a utilização dos mesmos ao máximo para, sem deixar de garantir o correto funcionamento do serviço, não desacelerar a sua latência de resposta a um ponto que se torne frustrante para o utilizador.

O modelo adoptado para a comunicação entre clientes e as situações onde o controlo de concorrência foi imposto serão explicados neste relatório, sendo que o código fonte também estará disponível para consulta na diretoria em anexo.

Para iniciar o servidor, executar 'java Uber'.

Para iniciar uma sessão de cliente, executar 'java Client'.

Desenvolvimento

- **Estrutura do Cliente**

O cliente é um pequeno processo com duas threads, uma responsável por enviar todo o input do utilizador para o servidor, e outra responsável por ler as mensagens do servidor e as apresentar ao utilizador através do terminal. O comando 'help' está disponível em qualquer altura para consultar os comandos disponíveis. Foi considerado, para simplificar o processo de parsing, que o modelo do condutor será constituido por apenas uma palavra que corresponde à marca/modelo da sua viatura.

- **Estrutura do Servidor**

O servidor também tem uma estrutura muito simples, sendo apenas constituido por uma thread inicial que está a dar "listen" à porta de comunicação escolhida para comunicar com os clientes e a criar uma nova thread para cada linha de comunicação com um cliente. Também faz parte do servidor uma instância da classe *Database*, responsável pelo armazenamento de todos os dados necessários ao funcionamento do serviço, nomeadamente os clientes e condutores registados, e os condutores actualmente disponíveis para transportar algum cliente.

- **Database**

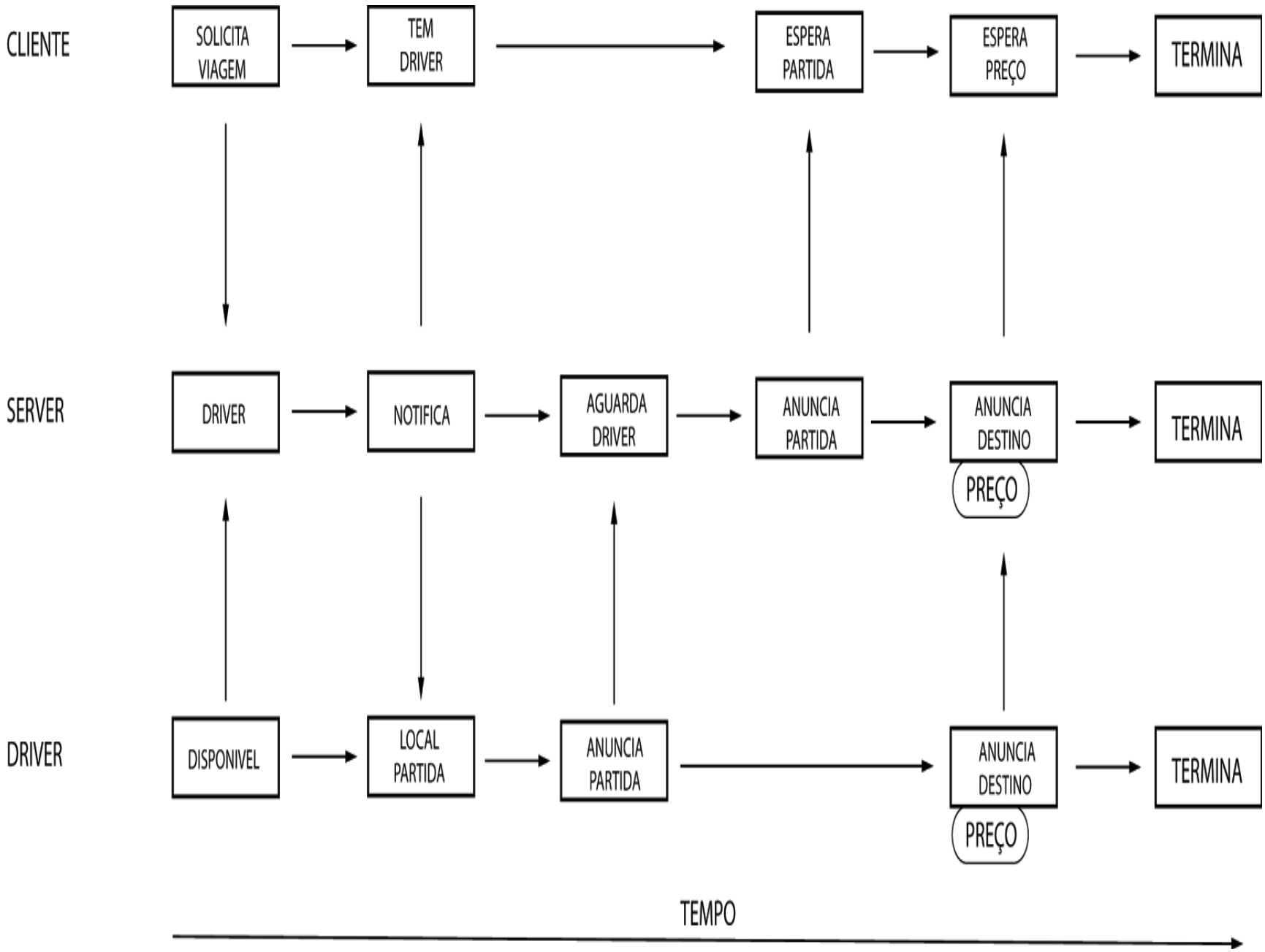
Na database foi usada uma estrutura conhecida do Java para armazenar os clientes, um *Map<String, ArrayList<Cliente>>*, onde a *String* representa a password e o *ArrayList<Cliente>* o conjunto de clientes com a dita password. Estamos a considerar que não existem clientes com o mesmo username, logo é possível fazer uma distinção entre clientes com a mesma password sem ser necessário o armazenamento explicito da mesma.

Faz também parte da database um *ArrayList<Avail>*, correspondente a uma "queue" de condutores disponíveis. O acesso a esta "queue", quer para consulta como para escrita, tem controlo de concorrência. A classe *Avail* guarda um condutor e a posição onde este se encontra disponível, posição essa que depois será usada por cada cliente para escolher o condutor que se encontra mais proxímo do local de partida no seu pedido. Também faz parte da classe *Avail* uma instância de *Pedido*, que inicialmente se encontra a *null*, mas é "preenchida" assim que for atribuido um pedido a esse condutor, sendo o condutor notificado dessa atribuição. Como o *Avail* é removido da queue antes dessa atribuição de um pedido lhe ser feita, e visto que após essa remoção este nunca volta à "queue", não existem problemas de concorrência, no entanto os locks são usados para permitir a notificação da parte do cliente ao condutor de que este já tem um pedido. Esta abordagem não tem impacto no funcionamento do programa visto que é garantido que uma instância de *Avail* é acedida e alterada por um cliente só.

- **Comunicação Cliente - Condutor**

Tal como é sugerido no enunciado, a comunicação entre cliente e condutor é feita tendo o servidor como intermediário. Dados os requisitos, também no enunciado, sobre o que cada condutor e cliente devem fazer, o esquema que se encontra na página seguinte foi elaborado para simplificar a compreensão do funcionamento da comunicação entre cliente e condutor.

Este esquema foi seguido à risca em todos os passos da comunicação entre condutor e servidor, e entre servidor e cliente. De salientar a decisão do grupo de bloquear tanto o cliente como o condutor quando estão em viagens até ao fim da mesma, sendo as únicas interações disponíveis as de questionar ao condutor se já se encontra no local de partida (e posterior notificação ao cliente dessa informação) e o preço da viagem no fim da mesma (sendo esta informação também transmitida ao cliente, como se pode observar no diagrama). No fim, os dois participantes na viagem voltam ao estado em que estavam, loggados nas suas respectivas contas.


CLIENTE
SOLICITA VIAGEM
TEM DRIVER
ESPERA PARTIDA
ESPERA PREÇO
TERMINA
SERVER
DRIVER
NOTIFICA
AGUARDA DRIVER
ANUNCIA PARTIDA
ANUNCIA DESTINO
PREÇO
TERMINA
DRIVER
DISPONIVEL
LOCAL PARTIDA
ANUNCIA PARTIDA
ANUNCIA DESTINO
PREÇO
TERMINA
TEMPO

Conclusão

Foi nos apresentado um projecto para aplicar os conceitos de controlo de concorrência e de comunicação cliente-servidor dados nas aulas de SD. O grupo acredita que conseguiu fazer isso mesmo com sucesso, atingindo os objectivos propostos e cumprindo os requisitos sugeridos no enunciado, produzindo um gestor de deslocações funcional, capaz de processar pedidos de diversos clientes e condutores.

Em suma, o grupo ficou satisfeito com o resultado final e aproveitou o tempo dedicado ao desenvolvimento da aplicação para aprofundar alguns conhecimentos que serão posteriormente avaliados no exame de fim de semestre.