iBei: Leilões Invertidos

Sistemas Distribuídos 2016/2017 — Meta 1 — 29 de outubro de 2016 (23:59)

Resumo

Num *leilão invertido* os compradores indicam o que pretendem adquirir e são os vendedores a fazer as licitações. Ganha o vendedor que oferecer o valor mais baixo. Neste tipo de leilões o preço vai descendo à medida que as licitações são feitas, ao contrário do habitual. Este projeto consiste em clonar um sistema típico de leilão (*e.g.*, eBay) por forma a realizar leilões invertidos.

1 Objetivos do projecto

No final do projeto o aluno deverá ter:

- programado um sistema de leilões, seguindo uma arquitetura cliente-servidor;
- aplicado sockets TCP/IP para comunicação entre clientes e servidores;
- seguido um modelo multithreaded para desenhar os servidores;
- criado uma camada de persistência de dados usando Java RMI;
- garantido a disponibilidade da aplicação com failover e balanceamento de carga.

2 Visão geral

Um leilão invertido é iniciado por um comprador que escolhe um artigo, indica o preço máximo que está disposto a pagar, e decide o momento em que o leilão vai terminar (data, hora e minuto). Os vendedores fazem licitações que vão sucessivamente baixando o preço até ao término do leilão. Ganha o vendedor que licitar o valor mais baixo. A descrição apresentada em `https://en.wikipedia.org/wiki/Reverse_auction` é um bom ponto de partida para pesquisar mais sobre o tema do projeto, sendo que a pesquisa não deve ficar por aí.

Para simplificar a escolha do *artigo* que vai a leilão considera-se que cada artigo tem um código que o identifica univocamente (por exemplo, o código EAN de 13 dígitos vulgarmente encontrado junto com o código de barras dos artigos, ou o código ISBN de 10 ou 13 dígitos habitualmente usado para identificar livros). Para iniciar um novo leilão, um utilizador escolhe um artigo, limita o preço máximo que pretende pagar, e indica a data, hora e minuto em que o leilão irá terminar.

3 Funcionalidades a desenvolver

Quando um utilizador se liga à aplicação de leilões deve poder escolher entre **registar uma nova conta** e **entrar com uma conta existente**. Assim que o utilizador tiver entrado na aplicação, poderá realizar as seguintes operações:

- **Criar um novo leilão.** Cria-se um leilão começando por identificar o artigo que se pretende comprar. Para simplificar, considera-se que cada artigo tem um código EAN/ISBN que o identifica univocamente. Cada leilão deve igualmente ter um título, uma descrição e quaisquer detalhes adicionais que o utilizador considere necessários. Para criar o leilão, o comprador indica o preço máximo que está disposto a pagar, bem como a data, hora e minuto em que o leilão termina.
- **Pesquisar leilões existentes por código.** Deve poder-se listar os leilões que estão a decorrer, pesquisando pelo código EAN/ISBN do artigo. Esta listagem apresenta apenas os detalhes suficientes de cada leilão que obedeça ao critério da pesquisa, e pode selecionar-se um desses leilões para consultar os respetivos detalhes.
- **Consultar detalhes de um leilão.** Para qualquer leilão escolhido, deve poder-se obter todos os detalhes relativos à descrição do artigo, ao término do leilão, às mensagens trocadas e ao histórico de licitações efetuadas nesse mesmo leilão.
- **Listar todos os leilões em que o utilizador tenha atividade.** Um utilizador deve poder listar os leilões nos quais tem ou teve alguma atividade, seja como criador do leilão seja como licitador. Esta listagem sumaria os detalhes de cada leilão e pode selecionar-se um desses leilões para consultar os respetivos detalhes.
- **Efetuar uma licitação num leilão.** Um vendedor pode propor um preço mais baixo num determinado leilão, desde que o leilão não tenha terminado e a licitação seja a mais baixa até ao momento e seja inferior ao preço máximo.
- **Editar propriedades de um leilão.** O comprador pode alterar todas as descrições textuais relativas a um leilão, sendo que todas as versões anteriores devem ficar guardadas e poder ser consultadas posteriormente para referência.
- **Escrever mensagem no mural de um leilão.** Cada leilão deve ter um mural onde poderão ser escritos comentários, questões e esclarecimentos relativos ao leilão.
- **Entrega imediata de mensagens a utilizadores que estejam ligados.** Utilizadores que estejam ligados à aplicação recebem imediatamente as mensagens publicadas no mural de um leilão (server push). O criador de um leilão recebe todas as mensagens relativas a esse leilão. Todos os utilizadores que tiverem escrito num mural passam a receber mensagens escritas nesse mesmo mural.
- **Entrega de mensagens a utilizadores desligados assim que estes se liguem.** Qualquer utilizador que devesse ter recebido uma mensagem, mas não se encontrasse ligado à aplicação, recebe a notificação assim que se ligar a próxima vez.
- **Listagem de utilizadores ligados atualmente.** Deve ser possível obter a lista completa de todos os utilizadores que se encontrem ligados à aplicação.

- **Notificação imediata de licitação melhor.** Um vendedor que tenha feito uma licitação num leilão é notificado imediatamente sempre que houver outra licitação melhor do que a sua, caso esteja ligado à aplicação.
- **Término do leilão na data, hora e minuto marcados.** No momento indicado pelo comprador (data, hora e minuto) o leilão termina. Determina-se aí o vencedor e fecha-se a possibilidade de realizar mais licitações. Os detalhes desse leilão são atualizados e podem ser consultados posteriormente.
- **Um administrador pode cancelar um leilão.** Um administrador deve poder cancelar um leilão se tal for necessário. O leilão continua a poder ser consultado pelos utilizadores, mas está dado como encerrado e não podem ser feitas licitações. **Grupos de 3 estudantes.**
- **Um administrador pode banir um utilizador.** Um administrador deve poder banir um utilizador se tal for necessário. Todos os leilões criados por esse utilizador são cancelados. Todas as licitações efetuadas por esse utilizador são apagadas. Note que, ao apagar uma licitação num leilão, quaisquer licitações inferiores a essa devem ser igualmente apagadas exceto a melhor delas, cujo valor se torna igual ao valor da que for apagada. Automaticamente é criada uma mensagem no mural dos leilões afetados lamentando o incómodo. **Grupos de 3 estudantes.**
- **Um administrador pode obter estatísticas de atividade na aplicação.** Um administrador deve poder consultar estatísticas da utilização da aplicação: top 10 utilizadores com mais leilões criados, top 10 utilizadores que mais leilões venceram, número total de leilões nos últimos 10 dias. **Grupos de 3 estudantes.**
- **Um administrador pode realizar testes a um servidor TCP.** Um administrador deve poder identificar um servidor (hostname, port) e realizar automaticamente um conjunto de testes que verifiquem se o servidor está a funcionar corretamente. No mínimo, deverá criar um leilão, licitar nesse leilão, consultar os detalhes desse leilão, e verificar se o resultado é o esperado. **Grupos de 3 estudantes.**

4 Arquitetura

A Figura 1 mostra a arquitetura geral do projeto. Cada grupo deverá programar os servidores RMI, que são idênticos embora um seja inicialmente primário e outro secundário. Cada grupo deverá igualmente programar os servidores TCP, que são idênticos salvo questões de configuração. Grupos de três estudantes têm de programar uma consola de administração, à parte, que se liga aos servidores RMI e aos servidores TCP.

O sistema deverá aceitar qualquer número de clientes TCP, desde que estes sigam o protocolo especificado. Estes poderão ser `telnet`, em ambientes Windows, `netcat` em ambientes Unix ou em qualquer outra linguagem que use sockets.

Deverão existir no mínimo dois clientes TCP que funcionam em federação, isto é, qualquer alteração feita por um cliente é visível em qualquer outro cliente, mesmo que esteja ligado a um servidor TCP diferente. A consolidação do estado é feita através de

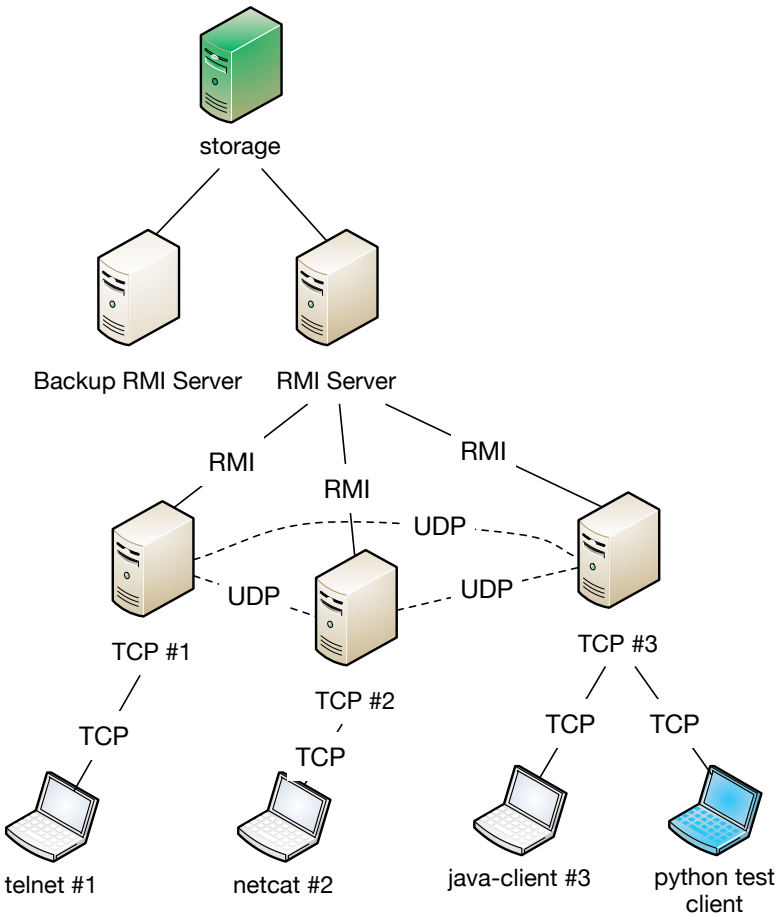

Fig. 1: Arquitectura do projecto

um RMI Server ativo. Como podem existir falhas de rede ou de hardware nesse servidor, deve existir um servidor RMI de backup, pronto a substituir o outro em caso de falha.

Os servidores RMI devem ter os dados persistidos de forma a que nunca se percam, mesmo que os processos sejam terminados. Para tal, poderá ser usado o sistema de ficheiros ou uma base de dados.

Será disponibilizado um cliente (python test client) que terá a capacidade de avaliar as funcionalidades do projecto.

5 Protocolo TCP

De forma a permitir a implementação de clientes iBei para qualquer plataforma (seja Windows, Linux, macos, Android, iOS), o protocolo foi especificado à priori. Este protocolo é extensível, permitindo acrescentar campos em pedidos e respostas com informações que sejam consideradas relevantes.

A estrutura principal deste protocolo é a mensagem que consiste num conjunto não ordenado de pares chave-valor, semelhante a um HashMap em Java ou a um dicionário

em Python, finalizados com uma mudança de linha. Um exemplo seria:

```
chave1: valor1, chave2: valor dois
```

Como tal, não é permitido ter o caracter vírgula nem o caracter mudar de linha no nome da chave nem no valor.

Todas as mensagens têm um campo obrigatório chamado “type”. Este valor permite distinguir o tipo de operação que o cliente pretende fazer no servidor, ou o tipo de resposta que o servidor está a dar. Um exemplo mais realista é portanto:

```
type: login, username: pierre, password: omidyar
```

E a resposta respectiva, sendo que o campo msg é opcional, mas aceite.

```
type: status, logged: on, msg: Welcome to iBei
```

Finalmente, para representar listas de elementos, é usado o tamanho e o contador de elementos. O tamanho é descrito num campo com o sufixo x_count, onde x é o campo com a lista, e cada campo do elemento tem o formato x_i_campo, onde i é o índice da lista, a começar em 0. Um exemplo encontra-se de seguida:

```
type: item_list, item_count: 2, item_0_name: A Book,
  item_0_price: 10, item_1_name: Other Book, item_1_price: 5
```

Este protocolo será usado não só pelos clientes e servidor TCP, mas também por um cliente que fará o teste das funcionalidades. Como tal o cumprimento do protocolo é essencial para a avaliação funcional.

6 Requisitos Funcionais

6.1 Registar conta

Os clientes poderão emitir um pedido de criação de conta, bastando para tal ter o username e a password. Poderão enviar outros parâmetros, mas estes são os únicos obrigatórios. Exemplo de pedido:

```
type: register, username: pierre, password: omidyar
```

A resposta será do mesmo tipo que o pedido e deverá ter o campo "ok"com o valor "true"se a operação foi bem sucedida, ou "false"se não foi. Um exemplo de resposta:

```
type: register, ok: true
```

6.2 Fazer login

Os clientes poderão emitir um pedido de login, bastando para tal ter o username e a password. Exemplo de pedido:

```
type: login, username: pierre, password: omidyar
```

A resposta será do mesmo tipo que o pedido e deverá ter o campo "ok"com o valor "true"se a operação foi bem sucedida, ou "false"se não foi. Um exemplo de resposta:

```
type: login, ok: true
```

6.3 Criar um novo leilão

Os clientes poderão criar um novo leilão a partir de um código identificador de artigo. Exemplo de pedido:

```
type: create_auction, code:9780451524935, title: 1984,
   description: big brother is watching you, deadline:
   2017-01-01 00:01, amount: 10
```

A resposta será do mesmo tipo que o pedido e deverá ter o campo "ok"com o valor "true"se a operação foi bem sucedida, ou "false"se não foi. Um exemplo de resposta:

```
type: create_auction, ok: true
```

6.4 Pesquisar leilão por código

Os clientes poderão pesquisar um novo leilão a partir de um código identificador de artigo. Exemplo de pedido:

```
type: search_auction, code:9780451524935
```

A resposta será do mesmo tipo que o pedido e deverá ter o campo de lista "items". Um exemplo de resposta vazia:

```
type: search_auction, items_count: 0
```

E um exemplo de uma resposta com conteúdo:

```
type: search_auction, items_count: 2, items_0_id: 101,
   items_0_code: 9780451524935, items_0_title: 1984,
   items_1_id: 103, items_1_code: 9780451524935, items_1_title
   : 1984 usado
```

6.5 Consultar detalhes de um leilão

Os clientes poderão consultar um determinado leilão a partir do identificador do leilão. A descrição e a deadline são campos obrigatórios. As mensagens e as bids são apresentadas em formato de lista codificada no protocolo. Exemplo de pedido:

```
type: detail_auction, id:101
```

A resposta será do mesmo tipo que o pedido e deverá ter o campo de lista "items". Um exemplo de resposta :

```
type: detail_auction, title: 1984, description: big brother is
    watching you, deadline: 2017-01-01 00:01, messages_count:
   2, messages_0_user: pierre, messages_0_text: qual a editora
   ?, messages_1_user: pierre, messages_1_text: entretanto vi
   que era a antigona, bids_count: 0
```

6.6 Consultar todos os leilões em que o utilizador tenha actividade

Os clientes poderão pesquisar um novo leilão a partir de um código identificador de artigo. Exemplo de pedido:

```
type: my_auctions
```

A resposta será do mesmo tipo que o pedido e deverá ter o campo de lista "items". Um exemplo de uma resposta com conteúdo:

```
type: my_auctions, items_count: 2, items_0_id: 101,
   items_0_code: 9780451524935, items_0_title: 1984,
   items_1_id: 103, items_1_code: 9780451524935, items_1_title
   : 1984 usado
```

6.7 Efectuar uma licitação num leilão

Os clientes poderão criar uma nova licitação com o identificador do leilão e o valor que estão dispostos a receber. Exemplo de pedido:

```
type: bid, id: 101, amount: 9
```

A resposta será do mesmo tipo que o pedido e deverá ter o campo "ok"com o valor "true"se a operação foi bem sucedida, ou "false"se não foi. Um exemplo de resposta:

```
type: bid, ok: true
```

Sempre que uma licitação é mais baixa que outra nesse mesmo leilão, os vendedores que fizeram licitações anteriores e estejam liados nesse momento devem receber uma notificação, com o formato abaixo:

```
type: notification_bid, id: 101, user: pierre, amount: 5
```

6.8 Editar um leilão

Os clientes poderão editar os diferentes campos de um leilão. Os campos que são passados nesta mensagem serão escritos por cima dos existentes:

```
type: edit_auction, id: 101, deadline: 2017-01-02 00:01
```

A resposta será do mesmo tipo que o pedido e deverá ter o campo "ok"com o valor "true"se a operação foi bem sucedida, ou "false"se não foi. Um exemplo de resposta:

```
type: edit_auction, ok: true
```

6.9 Escrever no mural de um leilão

Os clientes poderão colocar uma mensagem no mural de um leilão:

```
type: message, id: 101, text: alguma editora em especial?
```

A resposta será do mesmo tipo que o pedido e deverá ter o campo "ok"com o valor "true"se a operação foi bem sucedida, ou "false"se não foi. Um exemplo de resposta:

```
type: message, ok: true
```

Para além desta resposta, tanto o criador do leilão, como todos os que já escreveram no mural desse mesmo leilão, deverão receber a seguinte mensagem:

```
type: notification_message, id: 101, user: pierre, text:
   alguma editora em especial?
```

Caso um destes utilizadores não esteja online, receberá esta mensagem assim que volte a fazer login.

6.10 Listar utilizadores online

Os utilizadores poderão pedir uma lista de utilizadores que se encontrem ligados ao sistema usado a mensagem seguinte:

```
type: online_users
```

A resposta será do mesmo tipo que o pedido e deverá ter o campo de lista "users". Um exemplo de uma resposta com conteúdo:

```
type: online_users, users_count: 2, users_0_username: pierre,
   users_1_username: elon
```

6.11 Término do leilão na data, hora e minuto marcados

Assim que chega ao minuto definido para deadline do leilão, é necessário fechar a possibilidade de serem acrescentadas mais licitações.

7 Interface de Administração

Grupos com 3 elementos terão de implementar um conjunto de funcionalidades adicionais, agrupados num cliente TCP para administradores. Este cliente deverá ligar-se ao servidor TCP e disponibilizar ao administrador um conjunto de opções extra.

7.1 Cancelar um leilão

Os utilizadores considerados administradores poderão cancelar um leilão, impossibilitando novas licitações, mas mantendo-o visível.

7.2 Banir um utilizador

Os administradores poderão banir um utilizador, cancelando todos os seus leilões, apagando licitações feitas por ele, e fazendo as correcções necessárias. Ao apagar uma licitação num leilão, todas as licitações inferiores deverão ser apagadas, excepto a mais baixa. O valor da melhor licitação deverá ser alterado para o valor da licitação apagada. Assim, anula-se o efeito do utilizador no leilão. Sempre que isto acontecer, deve ser introduzida uma mensagem no mural da licitação a lamentar o incómodo causado.

7.3 Consultar estatísticas

Os administradores deverão poder consultar várias estatísticas em relação ao sistema.

7.4 Testar servidor TCP

Os administradores poderão realizar um teste a um servidor TCP a partir do hostname e porta. Este teste deverá, no mínimo, um leilão, licitar nesse leilão, consultar os detalhes desse leilão e verificar se o resultado está correcto.

7.5 Os clientes são informados da carga dos servidores TCP

A cada 60 segundos os clientes deverão receber uma mensagem do servidor com a carga dos servidores TCP. O valor usado deverá representar o melhor possível a carga do servidor, tendo em conta que um cliente que esteja num servidor com uma carga elevada poderá mudar para um com uma carga menor.

Um exemplo desta mensagem é:

```
type: notification_load, server_list: 5, server_0_hostname:
  localhost, server_0_port: 3000, server_0_load: 100
```

8 Requisitos Não-Funcionais

8.1 Tratamento de excepções e balanceamento de carga

Como o hardware pode falhar, é necessário que os utilizadores não notem nenhuma falha de serviço. Como tal, no caso do servidor RMI falhar, é preciso garantir que as licitações efectuadas não se percam, nem apareçam duplicadas. No caso das avarias temporárias (inferiores a 30 segundos), os clientes não se devem aperceber desta falha.

Também do lado de cliente é possível que a ligação se perca a meio. É necessário garantir que nenhuma falha do lado do cliente deixe nenhuma operação a meio.

De forma a que os servidores TCP possam dar informação sobre a carga de outros servidores para load balancing, estes devem trocar o número de clientes ligados por UDP.

8.2 Fail-over

De modo a que quando o servidor RMI falhar, o servidor secundário o substitua, é necessário ter alguns cuidados. Em primeiro lugar o servidor secundário deve testar periodicamente o primário para verificar que está a funcionar. Assim quando for detectado que este está em baixo, o servidor secundário liga-se para o substituir. Os servidores TCP também devem detectar quando existe uma falha permanente do servidor RMI e devem tentar ligar-se ao secundário. Dado o uso de persistência do lado do RMI, os dados visíveis pelo TCP devem ser exactamente os mesmos. Do lado dos clientes, todo este processo deve ser transparente. Para eles esta falha nunca deverá acontecer.

Finalmente, se o servidor RMI original recuperar, deverá tomar o papel de secundário e não de primário.

8.3 Relatório

Devem alocar tempo para a escrita do relatório no final do projecto, tendo em conta os passos anteriores. Devem escrever o relatório de modo a que um novo colega se junte ao grupo e perceba a solução criada, as decisões técnicas efectuadas e possa introduzir novos componentes ou modificar os que já existem. O relatório deve incluir:

- Arquitectura de Software detalhadamente descrita. Deverá ser focada a estrutura de processos, threads e sockets usadas, bem como a organização do código.
- Detalhes sobre o funcionamento do servidor TCP. Deve focar extensões feitas ao protocolo fornecido e detalhar a implementação do tratamento de falhas e balanceamento de carga.
- Detalhes sobre o funcionamento do servidor RMI. Deverá explicar o uso da interface e eventuais callbacks usados, bem como a solução usada para fail-over.
- Distribuição de tarefas pelos elementos do grupo.
- Descrição dos testes feitos à plataforma.

8.4 Distribuição de tarefas

De modo a que a avaliação seja justa num trabalho de grupo, é fundamental uma divisão de trabalho justa. Dado que a nota resultante da defesa será individual, são propostas as duas possíveis divisões de trabalho:

- Elemento 1 será responsável pelo Servidor TCP e o elemento 2 pelo Servidor RMI. Esta divisão assume que a interface RMI é negociada inicialmente e as suas alterações serão mínimas. No caso de existir um terceiro elemento, este ficaria responsável pela interface de administração.
- Cada um dos elementos ficará com igual número de funcionalidades a implementar.

Finalmente, poderão ser aceites outras distribuições, desde que previamente acordadas com os docentes.

9 O que irão aprender

Este projecto presupõe que os alunos aprendam competências práticas relativas a:

- Programar sockets em Java.
- Tratamento de Excepções em Java.

- Desenvolvimento de Servidores multi-threaded.
- Java RMI
- Implementar uma solução de Fail-over
- Implementar um servidor com suporte a multi-protocolos.

10 Entrega do projecto.

O projecto deverá ser entregue num ficheiro ZIP. Esse ficheiro deverá conter um ficheiro README com toda a informação necessária para instalar e executar o projecto sem a presença dos alunos. Projectos sem este ficheiro, ou sem informações suficientes **não serão considerados**. Projectos que não executem correctamente também não serão avaliados.

Dentro do ficheiro ZIP deverá também estar um PDF com o relatório. O relatório deve seguir a estrutura fornecida, dado que a avaliação irá incidir sobre cada um dos pontos.

Também no ficheiro ZIP deverão existir dois ficheiros JAR: o server.jar (servidor TCP) e o dataserver.jar (servidor RMI).

Finalmente, o ficheiro ZIP deverá ter também uma pasta com o todo código fonte do projecto. A ausência deste elemento levará à anulação do projecto.

O ficheiro ZIP deverá ser entregue na plataforma inforestudante até ao dia 29 de outubro de 2016 (23:59).

`http://inforestudante.uc.pt`