Servidor RMI

Leandro Kümmel Tria Mendes RA033910
Fernando Teixeira RA085858

24 de maio de 2013

Sumário

Lista de Figuras

Lista de Tabelas

1 Introdução

O objetivo desse projeto é implementar um sistema cliente/servidor, com operações para o gerenciamento de livros em uma livraria. Na comunicação entre cliente e servidor utilizou-se a interface de programação de chamadas remotas RMI[1] e a partir da execução de testes podemos avaliar alguns aspectos desse protocolo e posteriormente compará-lo com os projetos anteriores, no qual foi utilizado o TCP como protocolo.

2 Desenvolvimento

Linux foi o sistema operacional utilizado para o desenvolvimento (distribuição 2.6.43.8-1.fc15.i686). Igualmente, para os testes utilizou-se duas máquinas com linux, porém com distribuições diferentes.

2.1 RMI - Remote Method Invocation

O nível de abstração do RMI, para o programador, é maior em comparação aos projetos anteriores. Esse sistema possui um conjunto de operações e recursos que estão envolvidos no processo de invocação do método remoto. A implementação do RMI é essencialmente feita em três camadas, que por ordem de ṗroximidade do programador; são:

- *Camada Stub/Skeleton*: Podemos classificar o Stub, que reside no lado do cliente, como um proxie para um objeto remoto, ou seja, o Stub é responsável por receber os parâmetros dos métodos e encaminha-los para o Skeleton. Esse no entando, residente do servidor, recebe os parâmetros enviados pelo primeiro e executa as chamadas no objeto remoto. O Skeleton também é responsável por entregar ao Stub o retorno dos métodos remoto, e esse por fim, lê os resultados e retorna os valores ao objeto que inicialmente executou a chamada.

- *Camada de referência remota - RRL*: Logo abaixo da camada Stub/Skeleton[2.1] a RRL gerência e interpreta as refências remotas. Essa camada estabelece a semântica da ligação RMI, e as refêrencias entre o cliente e servidor (objetos remotos) é unicast, em outras palavras, um Stub para um Skeleton.

- *Camada de transporte TCP*: Essa última camada do RMI assegura a ligação entre as VM[2] através do protocolo TCP/IP, já especificado no projeto anterior.

2.2 Implementação

O projeto conta apenas com quatro diretórios, bin, src, relatorio e estat e um README.md[3] para instruções adicionais. Há também um Makefile, por di-

[1] Mais informações sobre RMI http://docs.oracle.com/javase/tutorial/rmi/overview.html

[2] Virtual Machine: http://en.wikipedia.org/wiki/VirtualMachine

[3] Leia esse arquivo antes de executar o sistema

retório e subdiretório, o qual compila e executa o sistema.
Os diretório são:

I. *bin*: Contém os binários do projeto, incluindo o stub/skeletons[2.1].

II. *src*: Contém os pacotes, e classes, do projeto. Importante lembrar que há o mysql-connector.jar, necessário para a conexão com o banco de dados para buscar resultados da biblioteca.

III. *estat*: Medidas de tempo efetuadas pelo teste.

IV. *relatorio*

O diretório src[II] contém os pacotes do projeto, que são:

I. *common*: Contém algumas variaveis/classes comum tanto ao cliente, quanto ao servidor. Há também a classe que marca e calcula o tempo e as estatísticas.

II. *biblioteca*: Contém as classes que manipulam a biblioteca, incluindo os métodos distribuidos, em outras palavras, os métodos que são acessados pelo cliente, ou seja, esse pacote pertence ao servidor.

III. *client*: Pacote exclusivo do cliente, que contém apenas as classes de execução do cliente e dos testes.

IV. *server*: Pacote exclusivo do servidor, que contém apenas a classes responsável por controle do servidor, ou seja, faz o r̈ebinddo mesmo esperando por conexões dos clientes.

2.2.1 Manipulação de dados

Diferentemente dos projetos anteriores, a manipulação dos dados é feita em conjunto com um banco de dados (MySQL[4]) portanto, ganha-se em simplicidade de implementação.
Arquivos presentes nos pacotes *common*[I] e *biblioteca*[II] :
Common:

I. *CommonVars.java*: Contém as variáveis estáticas utilizadas por ambos os lados do projetos, ou seja, tanto pelo cliente quanto pelo servidor.

II. *InOut.java*: Controla a parte de entrada/saída do programa, assim como a escrita, das estatísticas, em arquivos.

III. *Tempo.java*: Classe que contém os métodos que controla o tempo em nanosegundos (esse futuramente é transformado em milisegundos, para carater de comparação.

Bibliotecas:

I. *Biblioteca.java*: Interface para a BibliotecaImpl, citada abaixo.

II. *BibliotecaImpl.java*: Implementa a interface Biblioteca e contém os métodos acessados pelo cliente. Há também a conexão e consulta junto ao banco de dados.

[4] http://www.mysql.com/

III. *Livro.java*: Objeto Livro, contém os atributos do mesmo, tal como, isbn, quantidade, título.

IV. *Menu.java*: Gerencia o menu básico do sistema.

V. *Usuario.java*: Gerencia o login necessário para editar a quantidade de um livro.

2.2.2 Conexão Servidor/Cliente

Compreende dois pacotes *server*[IV] e *client*[III]
Servidor:

I. *Server.java*: Responsável por fazer o r̈ebinddo servidor, ou seja, nomea-lo e abrir a porta especificada em CommonVars.java[I], além de atribuir o skeleton. A inicialização do RMI é feito durante o tempo de execução.

Cliente:

I. *Cliente.java*: Faz o l̈ookupp̈elo servidor, ou seja, pesquisa pelo endereço do servidor, dado em CommonVars.java[I].

II. *Tester.java*: Mesma estrutura que o Cliente.java[I], porém executa os testes comparativos com os projetos anteriores.

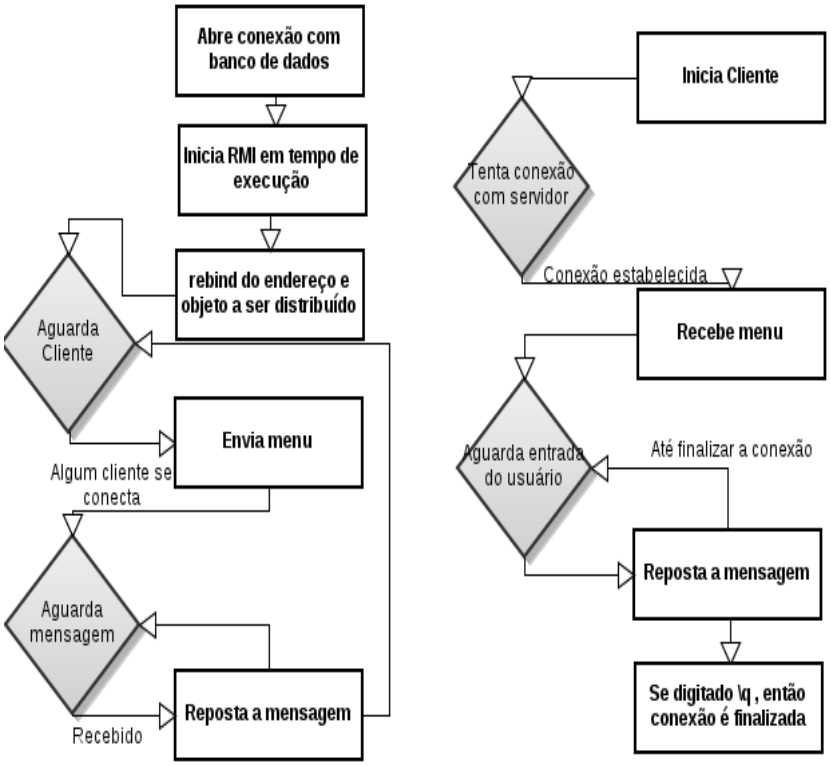
2.3 Coleta e gerência de dados para testes

Para realizar os testes implementou-se alguns arquivos adicionas[III] [II]. Uma constante, denominada $TOTAL$[5] , contém o número de testes a serem realizados, ou seja, cada opção do *menu*[IV] é executada $TOTAL$ vezes. Todos os dados são salvos no diretório *estat*[III].

2.4 Vantagens da implementação

O sistema de livraria é um sistema robusto e com baixa complexidade de tempo. A escolha do banco de dados mysql, possibilitou em boa performance em questão de tempo de processamento e também um maior nível de abstração, uma vez que, essa estrutura mostrou-se eficaz para o problema e tem melhor complexidade de tempo com relação a outras estruturas, como a simples leitura em arquivos, além de ser da implementação e manuntenção serem simples.
Com relação a comunicação entre cliente/servidor, o nível de abstração é maior em comparação com os projetos anteriores, por exemplo, a não preocupação com a leitura do buffer, como havia no caso do TCP e UDP, também não é necessário se preocupar com os z̈umbisp̈resentes no protocolo TCP.

[5]Nesse sistema consideramos TOTAL igual a 100



(a) Fluxograma Servidor (b) Fluxograma Cliente

Figura 1: Fluxogramas

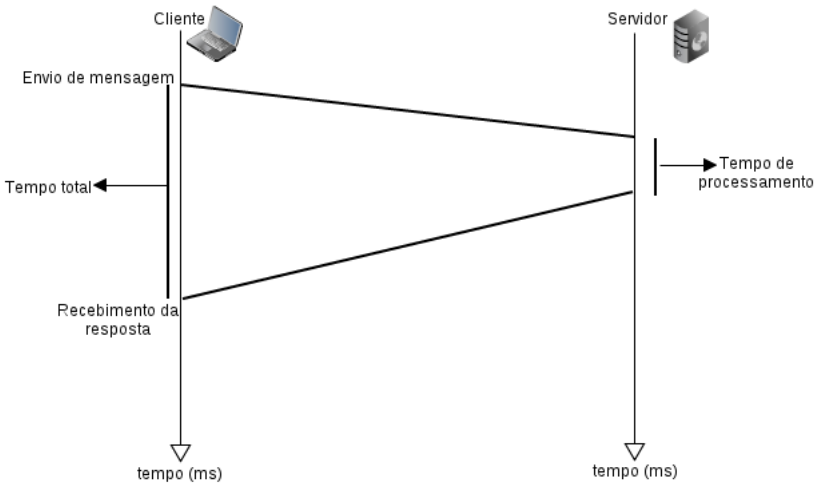


Figura 2: Definição do cálculo dos tempos

3 Resultados e discussões

Os testes foram efetuados em duas máquinas, ambas conectadas à rede porém, não localmente. Denominaremos a máquina servidor como [S] e a cliente como [C]. [S] e [C] estão em continentes diferentes.
Foram efetuadas 100 medições para cada opção do *menu*[IV], ao todo foram 600 medições de tempo. Dividiu-se o tempo em tempo de processamento e tempo de comunicação, sendo o último a diferença entre o tempo total e o tempo de processamento.
Abaixo demostramos o *traceroute*[6], o qual mostra a rota percorrida por um pequeno pacote, marcando assim os roteadores por qual passa. Vale ressaltar que os teste foram realizados em uma rede sem fio, sendo assim há uma perda maior de pacotes, e quanto maior a rota a ser percorrida, também espera-se uma perda maior, porém como descrito anteriormente[**??**], o RMI utilizada na camada de transporte o protocolo TCP, o qual garante a entrega, em ordem, da mensagem.

```
traceroute to 64.90.43.176 (64.90.43.176), 30 hops max, 60 byte packets
 1  c9528601.virtua.com.br (201.82.134.1)  26.304 ms  26.319 ms
30.156 ms
 2  c952005e.virtua.com.br (201.82.0.94)  20.162 ms  20.168 ms
20.187 ms
 3  embratel-T0-4-0-0-uacc01.cas.embratel.net.br (200.213.139.1)
19.337 ms  20.442 ms  25.724 ms
 4  ebt-T0-8-0-0-tcore01.cas.embratel.net.br (200.230.162.22)
38.070 ms ebt-T0-8-0-1-tcore02.cas.embratel.net.br (200.230.162.42)
34.997 ms ebt-T0-8-0-3-tcore02.cas.embratel.net.br (200.230.162.54)
35.304 ms
 5  ebt-Bundle-POS11932-intl03.mianap.embratel.net.br (200.230.220.54)
146.580 ms ebt-Bundle-POS11931-intl03.mianap.embratel.net.br (200.230.220.2)
149.048 ms  156.461 ms
 6  ae99.edge4.Miami1.Level3.net (4.59.90.9)  153.509 ms
134.693 ms  134.205 ms
 7  ae-2-52.edge2.Miami2.Level3.net (4.69.138.109)  140.293 ms ae-1-51.edge2.M
140.607 ms  142.033 ms
 8  ge-2-0-0.mpr1.mia1.us.above.net (64.125.14.89)  125.945 ms
125.854 ms  128.335 ms
 9  ge-1-0-0.mpr2.mia1.us.above.net (64.125.30.194)  135.120 ms
135.575 ms  135.136 ms
10  xe-4-0-0.cr2.iah1.us.above.net (64.125.30.202)  167.099 ms
161.208 ms  161.525 ms
11  xe-2-0-0.cr2.lax112.us.above.net (64.125.25.18)  190.919 ms
197.641 ms  197.113 ms
12  xe-3-2-0.mpr1.lax12.us.above.net (64.125.21.126)  200.062 ms
200.292 ms  200.553 ms
13  xe-0-1-0.mpr1.lax103.us.above.net (64.125.30.45)  204.362 ms
225.417 ms  225.202 ms
14  64.125.187.174 (64.125.187.174)  196.712 ms  200.319 ms
200.532 ms
```

[6] http://en.wikipedia.org/wiki/Traceroute

```
15  ip-66-33-201-126.dreamhost.com (66.33.201.126)  199.449 ms
194.863 ms  195.811 ms
16  ps59582.dreamhost.com (64.90.43.176)  194.074 ms  192.464 ms
193.108 ms
```

3.1 Tabelas e gráficos

3.1.1 Tempo total

[2.4] Representaremos todas as 100 medidas das 6 opções do *menu*[IV]

Opção 1 [ms]	Opção 2 [ms]	Opção 3 [ms]	Opção 4 [ms]	Opção 5 [ms]	Opção 6 [ms]
446302.637001	201859.0	203455.275001	441809.727996	425446.511001	206021.424995
236042.733001	202306.817001	211206.041999	405908.689994	396727.661003	221135.761001
214507.054000	199848.501998	197908.959999	430860.555000	398155.347000	194634.906005
196749.331001	201010.160995	202196.333999	406152.580001	393111.185997	208775.874000
200487.545005	199215.401000	198711.988998	426349.351005	415380.824996	227648.604003
223135.228996	208501.749000	200473.372001	413043.543998	458681.337997	201682.800003
203187.793998	201476.560997	208630.888000	405682.493995	395610.229003	209009.982002
198288.551994	198648.371002	207026.652000	425892.812995	399892.295997	196474.446006
201174.503997	198738.604995	218691.422004	417837.966995	397310.435005	210193.853996
193997.341995	198555.204002	207706.132995	413218.369003	392521.400001	210716.194000
220354.050003	202348.875	222586.570999	417722.527999	407561.427001	196513.631004
207720.732002	204144.968002	197973.020004	418044.424003	404513.660003	198629.799995
195790.853004	223860.324996	211164.223999	417092.571998	395318.034996	208428.291000
202062.186004	202700.039993	225928.515998	411775.334999	418987.489997	212321.134994
213788.694000	193195.859001	198874.804000	417100.540000	401456.188995	200682.558006
200667.402999	199188.389999	198840.096000	404977.132003	397993.916000	201131.635002
196644.170997	203041.983001	205587.041999	425757.493003	426088.117996	222372.802001
251828.336006	214173.309997	207217.750999	426328.093994	418352.233001	199312.712005
197087.450996	216054.957000	208344.699996	419791.510002	428960.597000	196317.111999
198321.185005	197666.222000	202186.023002	405747.554000	399394.164001	196014.212997
209232.793998	196235.888999	196298.675003	422134.178001	392696.665000	201738.277999
209839.222999	207876.913002	199305.034004	401249.620002	407497.417999	198154.059997
198125.916999	199879.262001	195543.421997	469197.847000	405469.250999	207953.881996
198868.113998	198508.363998	211049.842002	404092.696998	421988.010002	197331.064002
197693.325996	207005.726997	201241.436996	409509.182998	419669.027999	198204.980003
199470.491996	210798.0	196029.867996	425125.114997	404504.341995	197909.973999
199530.764999	210775.443000	213466.261001	421729.608993	406095.174003	194491.701995
197973.742004	203899.921005	197755.699996	407172.396003	409951.423004	211310.005996
196282.413002	204159.311996	215781.550003	403737.5	416969.706001	196021.000999
198807.290000	200577.259994	202633.296997	444801.518000	413920.049003	211372.725997
209231.414993	195458.647994	207692.336997	427478.967000	429514.962005	228070.555999
199297.365005	221667.504005	199085.402000	402230.140998	396105.823997	195635.835998
196606.131004	209850.762001	209131.826995	417132.933006	404127.153999	197955.312004
196799.031997	198213.652999	199962.319999	409082.344001	428814.888000	196819.007003
225448.887001	211319.792999	200131.684997	398049.914001	393696.682998	194700.744995
195960.597999	209023.158004	211305.824996	415452.229995	395682.882003	196085.821998
200525.659004	210316.257003	199024.714996	430344.474998	399720.016998	202845.762001
201950.268997	216141.311004	199656.143997	677981.160003	410939.068000	209316.425003
250301.308006	205385.445999	200330.039993	409912.624000	404815.451995	230620.589004
205800.371002	209093.493995	235347.397994	403669.999000	430063.796997	201132.019004
202470.584999	225615.328002	314829.455001	406067.347000	391834.332000	225474.807998
199788.290000	198674.750999	197507.366004	412308.724006	394028.166999	195188.798995
198251.448997	195472.144004	195592.687995	418162.483001	396696.986999	198957.187004
199937.750999	205143.945999	198117.076995	412755.916999	427949.805000	205883.418998
202281.951004	208333.218002	218314.411003	615691.930000	405715.784004	194414.484001
198340.430999	195988.058998	197601.516998	406002.971000	392262.327995	238461.251998
212394.795997	198926.396003	696983.492996	628332.664001	404471.303001	206435.582000
198419.703002	202343.136001	201467.684997	408965.473999	398840.128997	199757.081001
203214.748001	203655.444000	222263.551002	422627.759002	404122.521995	213866.972000
212493.556999	209972.173004	308753.834999	424518.625	413667.358001	225913.565002
207811.866004	200545.443000	199199.266006	404993.372001	434743.774993	196546.166000
197384.250999	197275.022003	200401.573997	415032.046005	407181.969001	213463.575996
207283.457000	200417.637001	913252.808998	604721.551994	396139.897003	195959.438003
203530.086997	219973.697998	245864.714996	402699.257995	429268.846000	206839.480995
204920.613998	200157.276000	196003.160995	395604.644996	450219.206001	220540.803001
207413.364997	220715.338005	197564.656997	444964.260002	412679.062995	209264.978004
211108.770996	207236.249000	211258.578002	404524.757995	414900.902999	197839.481994
197705.387001	196315.908996	203892.424995	432457.637001	392877.727996	392216.760002
195537.539001	197564.450996	211127.557998	415656.097999	397343.270004	199549.319000
196213.248001	200287.746002	204189.597999	401389.765998	392582.660995	197799.188003
203793.743003	212831.952003	209724.394996	567177.739997	403024.106002	203430.643997
200995.234001	307414.807998	210252.112998	431654.763999	430273.665000	208214.865997
209900.257995	240003.151000	211643.968002	464116.106002	411890.513000	213407.930999
196539.827003	208634.329002	201642.031997	418717.779998	414595.512001	202921.930000
200889.497993	200791.722999	197560.337997	414570.232002	454914.128005	215385.958999
219099.312004	216155.674003	213216.848999	443978.635002	419215.959999	220633.361999
198309.452003	201785.068000	212245.299995	419594.986000	408828.225006	197105.193000
198715.155998	195292.268997	197395.237998	409040.403999	414271.880996	197583.671005
198308.546997	195221.592002	196868.642997	403051.783996	398663.841995	192261.052001
199406.020996	200974.921005	200111.273994	416122.310997	438794.867004	194151.289001
195576.106002	212591.645996	219745.275001	400978.642997	393959.314002	215815.277999
226744.072998	195282.427001	212442.816001	398239.986000	409571.060997	194891.460998
244538.752998	198348.010993	212264.862998	393691.483001	394479.981994	201080.104003
196429.0	197866.603004	196730.083000	406578.050003	398891.884994	198913.440002
223037.777000	213719.018005	226795.617004	416645.779998	405211.271995	202933.259002
211193.302001	202181.004997	199681.481994	401386.454994	395328.972000	198982.377998
195414.570999	199644.747001	196140.077003	397678.620002	395871.075996	195121.241996
197315.144004	197390.852996	215272.972999	422361.122993	394279.063995	202507.513999
203538.639999	201281.739997	213744.094001	406289.062004	419013.244995	197895.453002
202906.235000	202089.450004	212239.170997	404218.359001	406165.375999	199605.223999
203950.980003	211811.261001	211371.820999	414535.143997	391801.475997	210042.898002
198990.695999	196298.339996	201663.841995	421251.575004	395017.076995	197364.318000
210292.089996	220099.944000	208548.171997	410738.893005	394524.633995	198122.929000
196241.492004	197679.093002	199326.139999	415191.374000	396468.231002	195653.922996
197758.851997	211163.772003	208015.150001	412997.472000	410256.791000	210346.429000
198670.272003	213207.867996	197828.065002	398801.888000	400727.393997	215784.079002
202353.873001	219191.463996	226185.018997	424688.513000	404806.015998	210709.113998
200304.542999	211865.255996	226055.114997	404203.083999	391936.984001	288172.521995
227480.841995	200830.794006	199736.810997	479177.166999	390264.652999	207316.75
207521.103996	210114.912002	202952.624000	544943.394004	512947.466995	204968.500999
216806.901000	198390.705993	198478.914001	441520.315002	402989.191001	196863.434997
207464.187004	205560.611999	199884.455001	886675.911003	641486.348999	200143.079002
197211.592002	205478.828994	198851.379997	428087.112998	439523.719001	217198.010002
199326.365997	203360.357002	198868.838996	430131.348999	400248.727005	208987.342994
404968.137001	199114.817001	218355.901000	413299.072006	420348.991996	207090.613998
227293.263000	201293.845001	197110.048004	425516.706001	397855.782005	194524.899002
195759.189994	207830.158004	195512.235000	447086.277000	406781.484001	199193.183006
209992.914001	197698.594001	224791.856994	403995.054000	423991.953002	221173.065002
215499.915000	203239.609001	200772.836997	421554.938995	408238.154998	195255.782997
197328.230995	197611.225997	199104.977996	405322.312004	398008.883003	195518.802001

Tabela 1: Tabela de tempo total

3.1.2 Tempo de processamento

[2.4] Representaremos todas as 100 medidas das 6 opções do *menu*[IV]

Opção 1 [ms]	Opção 2 [ms]	Opção 3 [ms]	Opção 4 [ms]	Opção 5 [ms]	Opção 6 [ms]
23499.42675	2588.69042	3674.17187	30315.66699	1988.348632	206021.424995
2741.711914	2166.27148	2809.39257	13130.375	1142.457031	221135.761001
2426.067382	2984.88476	2807.36425	17865.32031	1818.938476	194634.906005
2293.096679	2306.31054	2515.47070	12472.86621	1333.329101	208775.874000
2483.584960	2287.46679	2921.58496	16650.48925	988.6542968	227648.604003
2732.129882	3056.28710	3067.58593	12843.54296	6347.203125	201682.800003
2496.462890	2742.53710	2541.833984	11509.48535	1595.634765	209009.982002
2700.336914	2058.77929	2312.737304	10345.47851	1443.948242	196474.446006
2766.775390	2033.60839	2513.616210	10689.29199	1268.629882	210193.853996
2358.172851	2092.90527	2345.302734	13948.58105	1564.884765	210716.194000
2197.652343	2037.83398	2515.03125	10655.56347	4969.820312	196513.631004
2110.012695	2029.88183	2288.631835	12393.81542	1603.518554	198629.799995
1998.833007	2053.88671	2425.683593	9628.1640625	2353.079101	208428.291000
2033.600585	1992.09375	2267.762695	11944.89550	1618.004882	212321.134994
15533.80859	1939.20410	2315.449218	9182.381835	2624.336914	200682.558006
3161.451171	1906.79589	2256.185546	12086.03417	1620.867187	201131.635002
2274.647460	1929.55371	2201.988281	8954.654296	1142.541992	222372.802001
2100.997070	1996.64062	2189.830078	8639.394531	5510.153320	199312.712005
2092.358398	2156.50292	2962.610351	11984.80468	1543.340820	196317.111999
2376.628906	1977.66015	2055.752929	9506.116210	1564.905274	196014.212997
2383.828125	2045.03320	3069.470703	11532.54687	1212.748046	201738.277999
2089.916992	2102.73828	2143.847656	8912.556640	1645.018554	198154.059997
2098.202148	2213.47363	2408.840820	11315.48242	1459.604492	207953.881996
2298.115234	1944.46582	2326.465820	8787.368164	1286.202148	197331.064002
2955.897460	2150.33398	2477.500976	10796.84179	1261.020507	198204.980003
2044.863281	1784.53515	2418.101562	9116.516601	2306.648437	197909.973999
1960.921875	2122.51562	4408.851562	7822.693359	1258.483398	194491.701995
2085.931640	2024.09765	2209.492187	8474.766601	1519.058593	211310.005996
1999.516601	1937.89843	2048.783203	8002.017578	1768.298828	196021.000999
1985.002929	2206.93847	3774.604492	22611.29589	1475.580078	211372.725997
2314.264648	2116.26074	2004.117187	8703.311523	1681.860351	228070.555999
3211.170898	2156.47656	2419.379882	10909.97851	1524.775390	195635.835998
2230.250976	1826.13378	2079.896484	7674.668945	1320.524414	197955.312004
2409.091796	1969.08007	6134.481445	10327.28906	1701.374023	196819.007003
2142.537109	1745.32519	1894.795898	7567.179687	2158.863281	194700.744995
2088.868164	1863.08691	2232.594726	10223.21582	2145.129882	196085.821998
2257.673828	1610.82519	2940.666992	7426.370117	1564.366210	202845.762001
2641.204101	2063.41601	1885.816403	7706.202148	6250.834960	209316.425003
2255.297851	2386.09082	1824.250976	7812.797851	1527.680664	230620.589004
1825.700195	1649.90136	2054.471679	7009.142578	2639.392578	201132.019004
2079.362304	2705.86132	1824.631835	11831.16894	2115.976562	225474.807998
3437.389648	1729.25097	2103.592773	7020.528320	1355.232421	195188.798995
1926.607421	1865.64257	2128.505859	11155.11425	1257.493164	198957.187004
1862.973632	1663.25292	1940.451171	7306.949218	1106.516601	205883.418998
1717.3359375	1903.49804	2138.087890	215883.2871	2212.407226	194414.484001
2198.416015	2441.68457	1994.018554	7124.931640	2012.960937	238461.251998
2468.734375	1757.96777	2152.935546	10479.54394	1560.958007	206435.582000
2300.391601	1627.75585	2364.742187	7441.275390	1964.243164	199757.081001
2601.793945	1736.50781	4888.248046	8433.822265	1630.072265	213866.972000
1718.525390	2048.62695	2240.315429	8332.396484	1305.174804	225913.565002
1825.108398	2008.40332	2052.795898	6672.342773	1382.259765	196546.166000
1918.760742	1573.51953	3819.780273	10360.20703	1668.455078	213463.575996
1763.375	2080.39160	2134.796875	6950.464843	1457.456054	195959.438003
1587.811523	1725.84277	1930.450195	9369.930664	1066.200195	206839.480995
4070.6875	1640.64648	1781.019531	6570.716796	1402.186523	220540.803001
1531.560546	1443.74121	1852.238281	9488.590820	1357.510742	209264.978004
1527.393554	1933.15820	1853.941406	6379.066406	1201.644531	197839.481994
1513.079101	2116.26660	1934.556640	9937.797851	1608.972656	392216.760002
1525.045898	1625.32910	1688.001953	6878.621093	1385.870117	199549.319000
1734.748046	1714.30273	4222.914062	6077.128906	1447.272460	197799.188003
1660.745117	1802.28710	2009.717773	6263.658203	1498.202148	203430.643997
1561.499023	1583.49414	2002.026367	6037.053710	2530.360351	208214.865997
1903.579101	1719.99902	1904.137695	11610.35253	1251.470703	213407.930999
1511.046875	1610.22070	1703.552734	5989.808593	1574.523437	202921.930000
2468.661132	1849.58886	1850.693359	8498.420898	1199.282226	215385.958999
1588.126953	1530.31738	1758.201171	6212.533203	1823.911132	220633.361999
1541.253906	2223.86230	1747.850585	8067.129882	1696.613281	197105.193000
1734.319335	1616.85156	1685.582031	5460.998046	6037.231445	197583.671005
1748.565429	1421.30468	2361.884765	9296.408203	2382.949218	192261.052001
1281.558593	2223.39160	1656.470703	5133.78125	1688.726562	194151.289001
1318.267578	1423.40820	1749.207031	8326.278320	1549.931640	215815.277999
1490.784179	1587.52441	1746.084960	5560.356445	1597.565429	194891.460998
1541.271484	3861.52050	2101.828125	5255.871093	1859.307617	201080.104003
1308.565429	1734.07714	1789.326171	8345.045898	1418.059570	198913.440002
1313.391601	1808.31738	2061.145507	5448.684570	2052.705078	202933.259002
1450.140625	1615.26269	1625.257812	7759.170898	1481.342773	198982.377998
1223.756835	2609.58300	1787.131835	6312.569335	1589.511718	195121.241996
1944.65625	1638.81835	1682.167968	7794.291015	1837.879882	202507.513999
1305.014648	1663.13281	1763.418945	5210.501953	1653.555664	197895.453002
1267.692382	1699.91308	1989.556640	8614.524414	1834.074218	199605.223999
1641.869140	2133.82617	2044.693359	6811.107421	1938.327148	210042.898002
2623.924804	1397.0	2691.419921	8731.954101	1834.883789	197364.318000
1224.431640	1598.37109	1602.276367	5050.072265	1689.766601	198122.929000
1249.188476	1326.52441	1600.141601	5571.171875	1351.783203	195653.922996
1685.751953	1423.82812	1648.862304	4951.886718	1635.865234	210346.429000
1371.079101	1680.19140	1390.353515	5380.662109	1328.663085	215784.079002
1380.0625	1429.39355	1708.804687	8532.798828	1240.590820	210709.113998
2513.627929	1819.98437	1545.240234	6154.500976	1706.546875	288172.521995
1259.575195	1721.93847	2186.770507	86058.62988	1631.536132	207316.75
1453.826171	1437.08105	1739.717773	5067.154296	1303.712890	204968.500999
1326.1875	1474.50097	1959.007812	5327.113281	1531.705078	196863.434997
1325.004882	1431.42285	1788.939453	10347.16992	1472.869140	200143.079002
2211.475585	1599.43457	1715.9375	5077.920898	1471.658203	217198.010002
1201.508789	1927.44531	2120.757812	5038.077148	1426.590820	208987.342994
1698.415039	1943.12304	1583.235351	8413.541992	1280.854492	207090.613998
1963.515625	1764.84277	1576.397460	5397.816406	1986.184570	194524.899002
1187.314453	1490.30273	1949.586914	5848.684570	1970.21875	199193.183006
1281.374023	1784.49218	1911.683593	5785.825195	6355.679687	221173.065002
1325.030273	1686.62794	1786.504882	5122.362304	1274.379882	195255.782997
1306.115234	1413.28906	2769.505859	6345.514648	8054.580078	195518.802001

Tabela 2: Tabela de tempo de processamento

3.2 Médias e gráficos

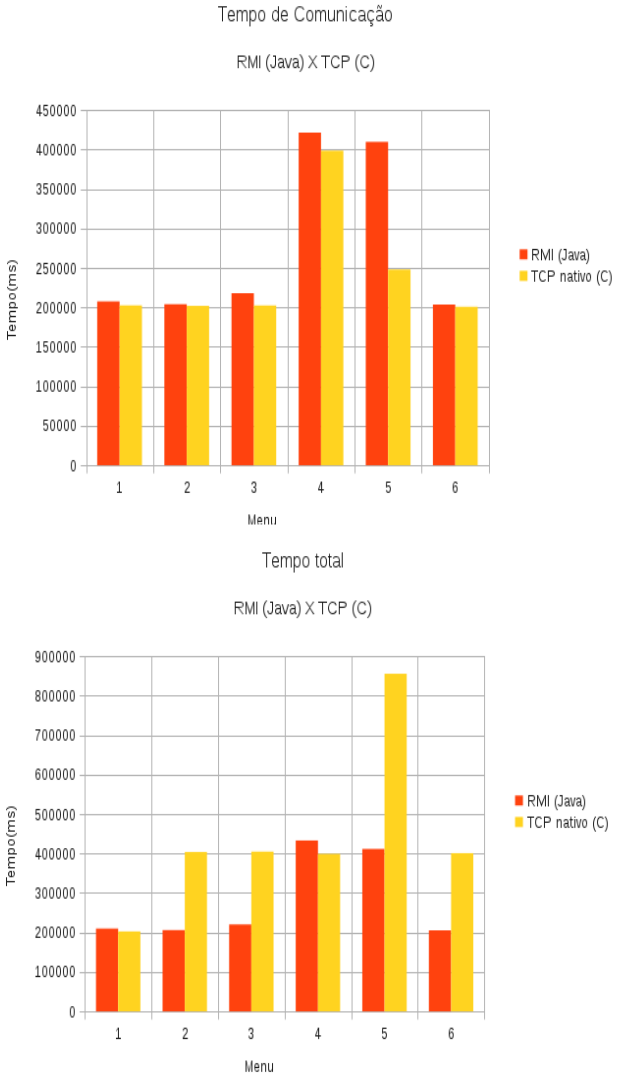
Devido ao grande número de dados decidimos por representar graficamente apenas as médias e desvios de cada opção do *menu*[IV].

3.2.1 Cálculos

A média calculada entre os pontos foi a média simples, em outras palavras, denominamos a média como $\boxed{\mu}$ então, $\boxed{\mu = \frac{1}{N}\sum_{i=1}^{N}(x_i)}$, onde N é o número de pontos, no caso desse projeto temos N=100 medidas para cada uma das opções de menu[**??**].

Já o desvio padrão amostral, denominado $\boxed{\sigma}$ é calculado como $\boxed{\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2}}$

Seja $\boxed{\mu_t}$, $\boxed{\mu_p}$, $\boxed{\mu_c}$, a média do tempo total, média do tempo de processamento e média do tempo de comunicação, respectivamente, e o desvio padrão do tempo de comunicação, de nominado $\boxed{\sigma_c}$.



(a) Gráficos Tempo de comunicação e total

Menu	$\boxed{\mu_t}$	$\boxed{\mu_p}$	$\boxed{\mu_c}$	$\boxed{\pm\sigma_c}$
1	209802.51	2313.57	207488.94	30464.18 (14,68%)
2	205886.77	1913.04	203973.73	12627.63 (6,19%)
3	220287.99	2250.85	218037.14	86018.67 (3,95%)
4	432944.65	11673.84	421270.81	44415.33 (10,54%)
5	411566.99	1932.92	409634.07	28004.75 (6,38%)
6	205504.94	1867.76	203637.18	22187.58 (28,89%)

Tabela 3: Tabela com médias e desvios para RMI

Menu	$\boxed{\mu_t}$	$\boxed{\mu_p}$	$\boxed{\mu_c}$	$\boxed{\pm\sigma_c}$
1	202560.05	60.41	202499.64	10252.64 (5,06%)
2	403655.22	201822.13	201833.09	7159.37 (3,54%)
3	404796.77	202373.30	202423.47	4133.69 (2,04%)
4	399065.79	563.89	398501.90	100786.75 (25,29%)
5	855097.01	607144.72	247952.29	51358.97 (20,71%)
6	400975.64	200207.89	200767.75	5832.36 (2,91%)

Tabela 4: Tabela com médias e desvios para protocolo TCP

3.3 Discussão

Vamos definir três componentes no cálculo comparativo entre o TCP, implementado em C, o qual chamaremos de TCP nativo, e o RMI, que utiliza TCP como protocolo da camada de transporte. São eles:

I. *Simplicidade - [S]*: Relativo ao tempo para escrever o código fonte do projeto, a quantidade de linhas escritas e complexidade de implementação.

II. *Tempo de comunicacao - [TC]*: Tempo de comunicação, já calculado.

III. *Tempo de processamento - [TP]*: Tempo de processamento, já calculado.

Para o cálculo comparativo entre os dois sistemas colocamos peso maior nos tempos de comunicação pois, em nosso projeto queremos focar essa variável porém, há casos em que o importante para o projeto seja uma implementação rápida, de baixo complexidade de organização entre outros fatores que favorecem a simplicidade [S]. O tempo de processamento recebeu o menor peso pois, consideramos que essa componente varia muito entre aplicações e algoritmos utilizados por elas. Visto que o sistema escrito em *Java*, ou seja, o RMI é muito mais simples do que o escrito em C, o projeto de TCP nativo. Também, há opções do menu na qual o RMI obtem menor tempo total devido ao tempo de processamento, nota-se que o mesmo foi superior, ou seja, foi na média 279.68% mais rápido, isso explica-se ao uso de banco de dados ao invéz de um simples leitura e escrita de arquivos, utilizando árvore AVL[7] como estrutura. Já, para a componente [TC] (como citado acima a com maior peso no projeto), temos que na média o TCP nativo foi superior em 13.92% ao RMI, chegando a 65.20% no caso do menu 5, onde há, uma maior troca de mensagens entre o cliente e servidor antes de parar o cronômetro.

[7] Vide projeto 1 e 2

4 Conclusão

Como escolhemos a componente tempo de comunicação, temos que o sistema TCP nativo, escrito na linguagem C, foi mais rápido do que o RMI, escrito em JAVA. Porém, é importante notar que a diferença é pequena em comparação ao ganho em termos das componentes de simplicidade e tempo de processamento do RMI. Logo, conclui-se que dependendo da aplicação que será desenvolvida pode-se escolher o RMI como solução, já que provê uma abstração maior ao programador, todavia se o projeto requer uma menor abstração e um tempo de comunicação menor, então a escolha fica com o TCP nativo.

5 Código fonte *(.java)*

5.1 Pacote *common*

5.1.1 CommonVars.java

```
1  package common;
2
3  public class CommonVars {
4    /*servidor rmi*/
5    public static String addr = "64.90.43.176";
6    public static int port = 2920;
7    public static String appName = "rmi_biblioteca";
8    /*servidor de dados mysql*/
9    public static String Host = "mysql.dirigente.com.br";/*host do
         banco*/
10   public static String Port = "3306";/*porta padrao mysql*/
11   public static String DataBase = "mc823_biblioteca";/*database*/
12   public static String User = "dirigente";/*user mysql*/
13   public static String Password = "diri2010gente";/*senha para o
         user*/
14   public static String Table = "livros";/*tabela com livros*/
15 }
```

5.1.2 InOut.java

```
1  package common;
2
3  import java.io.BufferedReader;
4  import java.io.BufferedWriter;
5  import java.io.File;
6  import java.io.IOException;
7  import java.io.FileWriter;
8  import java.io.InputStreamReader;
9  import java.io.PrintStream;
10
11 public class InOut {
12
13   public static String Isbn="Isbn:";/*requisicao de isbn*/
14   public static String Senha="Senha:";/*requisicao de senha*/
15   public static String Quantidade="Quantidade:";/*requisicao de
         quantidade*/
16
17   public static String requestIn(String t){
18     if(t.equals(InOut.Isbn)){
```

```
19        BufferedReader buff = new BufferedReader(new
              InputStreamReader(System.in));
20        System.out.print(InOut.Isbn);
21        try{
22          return buff.readLine();
23        }
24        catch(IOException ioe){
25          System.out.print("Error: terminal reader");
26          ioe.printStackTrace();
27        }
28      }
29      if(t.equals(InOut.Senha)){
30        BufferedReader buff = new BufferedReader(new
              InputStreamReader(System.in));
31        System.out.print(InOut.Senha);
32        try{
33          return buff.readLine();
34        }
35        catch(IOException ioe){
36          System.out.print("Error: terminal reader");
37          ioe.printStackTrace();
38        }
39      }
40      if(t.equals(InOut.Quantidade)){
41        BufferedReader buff = new BufferedReader(new
              InputStreamReader(System.in));
42        System.out.print(InOut.Quantidade);
43        try{
44          return buff.readLine();
45        }
46        catch(IOException ioe){
47          System.out.print("Error: terminal reader");
48          ioe.printStackTrace();
49        }
50      }
51      return null;
52    }
53    /**
54     * Escreve string em um arquivo
55     * @param String f : caminho do arquivo
56     * @param String d : dado
57     */
58    public static void writeOut(String f, String d){
59      try {
60        File arq = new File(f);
61        if (!arq.exists()) {
62          arq.createNewFile();
63        }
64            BufferedWriter out = new BufferedWriter(new FileWriter(
                  arq.getAbsoluteFile(),true));
65            out.write(d);
66            out.close();
67        } catch (IOException e) {
68          System.out.print("Error: writeOut, erro ao escrever em
                arquivo"+f);
69          e.printStackTrace();
70        }
71    }
72    /**
73     * Escreve int em um arquivo
74     * @param String f : caminho do arquivo
75     * @param int d : dado
```

```
76     */
77    public static void writeOut(String f, int d){
78      try {
79        File arq = new File(f);
80        if (!arq.exists()) {
81          arq.createNewFile();
82        }
83            BufferedWriter out = new BufferedWriter(new FileWriter(
                  arq.getAbsoluteFile(),true));
84            out.write(d);
85            out.close();
86        } catch (IOException e) {
87          System.out.print("Error: writeOut, erro ao escrever em
                arquivo"+f);
88          e.printStackTrace();
89        }
90    }
91    /**
92     * Escreve string no bash
93     * @param PrintStream o : bash output
94     * @param String s : dado
95     */
96    public static void writeOut(PrintStream o, String s){
97      o.println(s);
98    }
99  }
```

5.1.3 Tempo.java

```
1  package common;
2
3  public class Tempo {
4    private double inicio;/*marca inicio*/
5    private double fim;/*marca fim*/
6    public Tempo(long i, long f){
7      this.inicio=i;
8      this.fim=f;
9    }
10   /*Construtor alternativo*/
11   public Tempo(){
12     this.inicio=0;
13     this.fim=0;
14   }
15   /*marca inicio do cronometro*/
16   public void start(){
17     Long l = new Long(System.nanoTime());
18     this.inicio=l.doubleValue()/1000.00;
19   }
20   /*para cronometro*/
21   public void stop(){
22     Long l = new Long(System.nanoTime());
23     this.fim=l.doubleValue()/1000.00;
24   }
25   /**
26    * retorna tempo inicial
27    * @return long inicio
28    */
29   public double getInicio(){
30     return this.inicio;
31   }
32   /**
```

```
33      * retorna tempo final
34      * @return long fim
35      */
36     public double getFim(){
37       return this.fim;
38     }
39     public double getTempo(){
40       return this.fim-this.inicio;
41     }
42     /**
43      * Escreve o tempo decorrido em um arquivo dado
44      * @param String f : caminho do arquivo
45      */
46     public void write(String f){
47       InOut.writeOut(f, String.valueOf(this.getTempo())+"\n");
48     }
49 }
```

5.2 Pacote *biblioteca*

5.2.1 Biblioteca.java

```
1  /**
2   * Interface da classe Server
3   */
4  package biblioteca;
5
6  import java.rmi.Remote;
7  import java.rmi.RemoteException;
8
9  public interface Biblioteca extends Remote {
10   public String getAllISBN() throws RemoteException;
11   public String getDescByIsbn(String isbn) throws RemoteException;
12   public String getBookByIsbn(String isbn) throws RemoteException;
13   public String getBooks() throws RemoteException;
14   public String setQuantidade(String isbn, int qnt) throws
         RemoteException;
15   public String getQuantidade(String isbn) throws RemoteException;
16   public String getMenu() throws RemoteException;
17   public boolean doLogin(String senha) throws RemoteException;
18 }
```

5.2.2 BibliotecaImpl.java

```
1  package biblioteca;
2  import java.rmi.RemoteException;
3  import java.rmi.server.UnicastRemoteObject;
4  import java.sql.DriverManager;
5  import java.sql.Connection;
6  import java.sql.SQLException;
7  import java.sql.Statement;
8  import java.sql.ResultSet;
9  import java.util.Vector;
10 import common.*;
11
12 public class BibliotecaImpl extends UnicastRemoteObject implements
       Biblioteca {
13   private Tempo tp;/*tempo de processamento*/
```

```
14    private Connection conn=null;/*conexao com o banco*/
15
16    public BibliotecaImpl() throws RemoteException {
17      super();
18      this.tp=new Tempo();
19      try{
20        Class.forName("com.mysql.jdbc.Driver");/*driver ok?*/
21      }catch(ClassNotFoundException e){
22        System.out.println("Error: JDBC connector missing");/*falta o
              driver .jar*/
23        e.printStackTrace();
24        return;
25      }
26      try{
27        /*registrando a conex o com o banco*/
28        this.conn = DriverManager.getConnection("jdbc:mysql://"+
              CommonVars.Host+":"+CommonVars.Port+"/"+CommonVars.
              DataBase,CommonVars.User,CommonVars.Password);
29      }catch(SQLException e){
30        System.out.println("Error: Falha ao tentar conex o com o
              banco de dados");/*algum problema ao conectar com o banco
              */
31        e.printStackTrace();
32        return;
33      }
34      if(this.conn==null){/*Essa conex o realmente existe?*/
35        System.out.println("Error: Falta conex o com o banco de
              dados");
36      }
37    }
38    /**
39     * M todo que lista todos os ISBN presentes na Biblioteca.
40     * @return String
41     */
42    public String getAllISBN() throws RemoteException {
43      this.tp.start();/*inicio do tempo de processamento*/
44      String s = new String();/*string de retorno*/
45      String qr = "SELECT isbn FROM "+CommonVars.Table+" ORDER BY
            isbn ASC";/*query*/
46      Statement st = null;/*statement*/
47      ResultSet r = null;/*resultado da query*/
48      Vector<Livro> livros = new Vector<Livro>();/*livros*/
49      /*tentativa de criar o statement*/
50      try{
51        st = this.conn.createStatement();
52      }catch(SQLException e){/*algum problema?*/
53        System.out.print("Error: getAllISBN, falha na ao criar
              statement");/*falha*/
54        e.printStackTrace();
55        return null;/*fim*/
56      }
57      /*tentativa de executar a query*/
58      try{
59         r = st.executeQuery(qr);
60         while(r.next()){/*vamos ver os resultados*/
61           livros.add(new Livro(r.getString("isbn")));/*livro com
                 apenas o isbn*/
62         }
63         for(Livro l : livros)/*percorrer os livros e concaternar
               string s*/
64           s=s.concat(l.toString()+"\n");
65      }catch(SQLException e){/*algum problema?*/
```

```
66        System.out.print("Error: getAllISBN, falha ao executar query"
              );/*erro na query*/
67        e.printStackTrace();
68        return null;/*fim*/
69      }
70      this.tp.stop();/*fim cronometro*/
71      this.tp.write("./estat/tp_1");/*escreve estatistica*/
72      if(s.length()==0)/*n o existe nenhum livro*/
73        return Menu.BookNotFound;/*imprime livro nao encontrado*/
74      return s;/*retorna uma string com os isbn*/
75    }
76
77    /**
78     * M todo que lista a descricao de um livro dado um ISBN
79     * @param String isbn;
80     * @return String s;
81     */
82    public String getDescByIsbn(String isbn) throws RemoteException {
83      this.tp.start();/*inicio do tempo de processamento*/
84      String s = new String();/*string de retorno*/
85      String qr = "SELECT * FROM "+CommonVars.Table+" WHERE isbn LIKE
              '"+isbn+"'";/*query*/
86      Statement st = null;/*statement*/
87      ResultSet r = null;/*resultado da query*/
88      Vector<Livro> livros = new Vector<Livro>();/*livros*/
89      /*tentativa de criar o statement*/
90      try{
91        st = this.conn.createStatement();
92      }catch(SQLException e){/*algum problema?*/
93        System.out.print("Error: getBookByIsbn, falha na ao criar
              statement");/*falha*/
94        e.printStackTrace();
95        return null;/*fim*/
96      }
97      /*tentativa de executar a query*/
98      try{
99         r = st.executeQuery(qr);
100        while(r.next()){/*vamos ver os resultados*/
101          livros.add(new Livro(isbn,
102                     r.getString("desc")));/*livro completo*/
103        }
104        for(Livro l : livros)/*percorrer os livros e concaternar
               string s*/
105          s=s.concat(l.toString()+"\n");
106     }catch(SQLException e){/*algum problema?*/
107       System.out.print("Error: getBookByIsbn, falha ao executar
              query");/*erro na query*/
108       e.printStackTrace();
109       return null;/*fim*/
110     }
111     this.tp.stop();/*fim cronometro*/
112     this.tp.write("./estat/tp_2");/*escreve estatistica*/
113     if(s.length()==0)/*n o existe nenhum livro*/
114       return Menu.BookNotFound;/*imprime livro nao encontrado*/
115     return s;/*retorna uma string com os isbn*/
116   }
117
118   /**
119    * M todo que lista um livro dado um ISBN
120    * @param String isbn;
121    * @return String s;
122    */
```

```
123   public String getBookByIsbn(String isbn) throws RemoteException {
124     this.tp.start();/*inicio do tempo de processamento*/
125     String s = new String();/*string de retorno*/
126     String qr = "SELECT * FROM "+CommonVars.Table+" WHERE isbn LIKE
            '"+isbn+"'";/*query*/
127     Statement st = null;/*statement*/
128     ResultSet r = null;/*resultado da query*/
129     Vector<Livro> livros = new Vector<Livro>();/*livros*/
130     /*tentativa de criar o statement*/
131     try{
132       st = this.conn.createStatement();
133     }catch(SQLException e){/*algum problema?*/
134       System.out.print("Error: getBookByIsbn, falha na ao criar
              statement");/*falha*/
135       e.printStackTrace();
136       return null;/*fim*/
137     }
138     /*tentativa de executar a query*/
139     try{
140        r = st.executeQuery(qr);
141        while(r.next())/*vamos ver os resultados*/
142          livros.add(new Livro(isbn,
143                       r.getString("titulo"),
144                       r.getString("autores").split(";"),
145                       r.getString("desc"),
146                       r.getString("editora"),
147                       r.getString("ano"),
148                       r.getInt("quantidade")));/*livro completo*/
149        for(Livro l : livros)/*percorrer os livros e concaternar
              string s*/
150          s=s.concat(l.toString()+"\n");
151     }catch(SQLException e){/*algum problema?*/
152       System.out.print("Error: getBookByIsbn, falha ao executar
              query");/*erro na query*/
153       e.printStackTrace();
154       return null;/*fim*/
155     }
156     this.tp.stop();/*fim cronometro*/
157     this.tp.write("./estat/tp_3");/*escreve estatistica*/
158     if(s.length()==0)/*n o existe nenhum livro*/
159       return Menu.BookNotFound;/*imprime livro nao encontrado*/
160     return s;/*retorna uma string com os isbn*/
161   }
162
163   /**
164    * M todo que lista todos os livros
165    * @param String isbn;
166    * @return String s;
167    */
168   public String getBooks() throws RemoteException {
169     this.tp.start();/*inicio do tempo de processamento*/
170     String s = new String();/*string de retorno*/
171     String qr = "SELECT * FROM "+CommonVars.Table+" ORDER BY isbn
            ASC";/*query*/
172     Statement st = null;/*statement*/
173     ResultSet r = null;/*resultado da query*/
174     Vector<Livro> livros = new Vector<Livro>();/*livros*/
175     /*tentativa de criar o statement*/
176     try{
177       st = this.conn.createStatement();
178     }catch(SQLException e){/*algum problema?*/
```

```
179         System.out.print("Error: getBooks, falha na ao criar
                statement");/*falha*/
180         e.printStackTrace();
181         return null;/*fim*/
182       }
183       /*tentativa de executar a query*/
184       try{
185           r = st.executeQuery(qr);
186           while(r.next())/*vamos ver os resultados*/
187             livros.add(new Livro(r.getString("isbn"),
188                         r.getString("titulo"),
189                         r.getString("autores").split(";"),
190                         r.getString("desc"),
191                         r.getString("editora"),
192                         r.getString("ano"),
193                         r.getInt("quantidade")));/*livro completo*/
194           for(Livro l : livros)/*percorrer os livros e concaternar
                string s*/
195             s=s.concat(l.toString()+"\n");
196       }catch(SQLException e){/*algum problema?*/
197         System.out.print("Error: getBooks, falha ao executar query");
              /*erro na query*/
198         e.printStackTrace();
199         return null;/*fim*/
200       }
201       this.tp.stop();/*fim cronometro*/
202       this.tp.write("./estat/tp_4");/*escreve estatistica*/
203       if(s.length()==0)/*n o existe nenhum livro*/
204         return Menu.BookNotFound;/*imprime livro nao encontrado*/
205       return s;/*retorna uma string com os isbn */
206     }
207
208     /**
209      * M todo que edita a quantidade em estoque de um livro
210      * @param String isbn;
211      * @return String s;
212      */
213     public String setQuantidade(String isbn, int qnt) throws
            RemoteException {
214       this.tp.start();/*inicio do tempo de processamento*/
215       String s = new String("Opera  o realizada com sucesso.\n");/*
              string de retorno*/
216       if(qnt<0)
217         return "Erro: A quantidade deve ser um n mero natural.\n";
218       String qr = "UPDATE "+CommonVars.Table+" SET quantidade = "+qnt
              +" WHERE isbn LIKE '"+isbn+"'";/*query*/
219       Statement st = null;/*statement*/
220       /*tentativa de criar o statement*/
221       try{
222         st = this.conn.createStatement();
223       }catch(SQLException e){/*algum problema?*/
224         System.out.print("Error: setQuantidade, falha na ao criar
                statement");/*falha*/
225         e.printStackTrace();
226         return null;/*fim*/
227       }
228       /*tentativa de executar a query*/
229       try{
230          st.execute(qr);
231       }catch(SQLException e){/*algum problema?*/
232         System.out.print("Error: setQuantidade, falha ao executar
                query");/*erro na query*/
```

```
233         e.printStackTrace();
234         return null;/*fim*/
235       }
236       this.tp.stop();/*fim cronometro*/
237       this.tp.write("./estat/tp_5");/*escreve estatistica*/
238       if(s.length()==0)/*n o existe nenhum livro*/
239         return Menu.BookNotFound;/*imprime livro nao encontrado*/
240       return s;/*retorna uma string com os isbn*/
241     }
242
243     /**
244      * M todo que lista a quantidade em estoque de um livro
245      * @param String isbn;
246      * @return String s;
247      */
248     public String getQuantidade(String isbn) throws RemoteException {
249       this.tp.start();/*inicio do tempo de processamento*/
250       /*Atencao para carater de comparacao do protocolo TCP, presente
              no projeto_1
251        *o start do tempo de processamento da opcao 5 encontra-se no
              metodo doLogin(*);
252        * */
253       String s = new String();/*string de retorno*/
254       String qr = "SELECT quantidade FROM "+CommonVars.Table+" WHERE
            isbn = '"+isbn+"'";/*query*/
255       Statement st = null;/*statement*/
256       ResultSet r = null;/*resultado da query*/
257       Vector<Livro> livros = new Vector<Livro>();/*livros*/
258       /*tentativa de criar o statement*/
259       try{
260         st = this.conn.createStatement();
261       }catch(SQLException e){/*algum problema?*/
262         System.out.print("Error: getBooks, falha na ao criar
              statement");/*falha*/
263         e.printStackTrace();
264         return null;/*fim*/
265       }
266       /*tentativa de executar a query*/
267       try{
268          r = st.executeQuery(qr);
269          while(r.next())/*vamos ver os resultados*/
270            livros.add(new Livro(isbn,r.getInt("quantidade")));/*livro
                  com isbn e quantidade*/
271          for(Livro l : livros)/*percorrer os livros e concaternar
                string s*/
272            s=s.concat(l.toString()+"\n");
273       }catch(SQLException e){/*algum problema?*/
274         System.out.print("Error: getBooks, falha ao executar query");
              /*erro na query*/
275         e.printStackTrace();
276         return null;/*fim*/
277       }
278       this.tp.stop();/*fim cronometro*/
279       this.tp.write("./estat/tp_6");/*escreve estatistica*/
280       if(s.length()==0)/*n o existe nenhum livro*/
281         return Menu.BookNotFound;/*imprime livro nao encontrado*/
282       return s;/*retorna uma string com os isbn*/
283     }
284     public boolean doLogin(String senha) throws RemoteException {
285       this.tp.start();/*inicio do tempo de processamento PARA A OPCAO
              5 DO MENU*/
286       Usuario u = new Usuario(this.conn);
```

```
287       return u.login(senha);
288     }
289     /*m todo que imprime o menu*/
290     public String getMenu() throws RemoteException {
291       return Menu.getPrint();
292     }
293   }
```

5.2.3 Livro.java

```
1   /**
2    * Classe de manipula  o de um livro
3    */
4
5   package biblioteca;
6
7   import java.util.Arrays;
8
9   public class Livro {
10    private String isbn="";
11    private String titulo="";
12    private String[] autores={};
13    private String desc="";
14    private String editora="";
15    private String ano="";
16    private int quantidade=-1;
17
18    public Livro(String isbn, String titulo, String[] autores, String
          desc, String editora, String ano, int qnt){
19      /*atributos*/
20      this.isbn=isbn;
21      this.titulo=titulo;
22      this.autores=autores;
23      this.desc=desc;
24      this.editora=editora;
25      this.ano=ano;
26      this.quantidade = qnt;
27    }
28    /*apenas o isbn*/
29    public Livro(String isbn){
30      this.isbn=isbn;
31      this.quantidade=-1;
32    }
33    public Livro(String isbn,String desc){
34      this.isbn="";
35      this.desc=desc;
36      this.quantidade=-1;
37    }
38    public Livro(String isbn,int qnt){
39      this.isbn="";
40      this.quantidade=qnt;
41    }
42    public String toString(){
43      String s = new String();
44      if(this.isbn.length()>0)
45        s=s.concat("ISBN: "+this.isbn+"\n");
46      if(this.titulo.length()>0)
47        s=s.concat("T tulo: "+this.titulo+"\n");
48      if(this.autores.length>0)
49        s=s.concat("Autores: "+Arrays.toString(this.autores)+"\n");
50      if(this.desc.length()>0)
```

```
51        s=s.concat("Descri   o: "+this.desc+"\n");
52      if(this.editora.length()>0)
53        s=s.concat("Editora: "+this.editora+"\n");
54      if(this.ano.length()>0)
55        s=s.concat("Ano: "+this.ano+"\n");
56      if(this.quantidade>0)
57        s=s.concat("Quantidade: "+this.quantidade+"\n");
58      return s;
59    }
60  }
```

5.2.4 Menu.java

```
1  package biblioteca;
2  import java.rmi.Naming;
3  import common.*;
4  public class Menu {
5    /*opcoes de menu*/
6    public static String ListIsbn = "1";
7    public static String DescByIsbn = "2";
8    public static String BookByIsbn = "3";
9    public static String ListBooks = "4";
10   public static String EditBook = "5";
11   public static String AmountByIsbn = "6";
12   public static String EndProgram="\\q";
13   public static String RepeatMenu="\\m";
14   public static String BookNotFound = "Livro n o encontrado";
15
16   /*tempo de comunicacao*/
17   private Tempo tc;
18   public Menu(){
19     this.tc=new Tempo();/*inicia objeto tempo*/
20   }
21   /*dirige a escolha do menu*/
22   public void chooseMenu(String opt, Biblioteca b){
23     String out = new String();
24     try {
25       if(opt.equals(Menu.ListIsbn)){
26         this.tc.start();/*inicia cronometro*/
27         out=b.getAllISBN();/*contato com servidor*/
28         this.tc.stop();/*finaliza cronometro*/
29         this.tc.write("./estat/tc_1");/*escreve estatistica*/
30       }
31       else if(opt.equals(Menu.DescByIsbn)){
32         this.tc.start();/*inicia cronometro*/
33         out=b.getDescByIsbn(InOut.requestIn(InOut.Isbn));/*contato
               com servidor*/
34         this.tc.stop();/*finaliza cronometro*/
35         this.tc.write("./estat/tc_2");/*escreve estatistica*/
36       }
37       else if(opt.equals(Menu.BookByIsbn)){
38         this.tc.start();/*inicia cronometro*/
39         out=b.getBookByIsbn(InOut.requestIn(InOut.Isbn));/*contato
               com servidor*/
40         this.tc.stop();/*finaliza cronometro*/
41         this.tc.write("./estat/tc_3");/*escreve estatistica*/
42       }
43       else if(opt.equals(Menu.ListBooks)){
44         this.tc.start();/*inicia cronometro*/
45         out=b.getBooks();/*contato com servidor*/
46         this.tc.stop();/*finaliza cronometro*/
```

```
47          this.tc.write("./estat/tc_4");/*escreve estatistica*/
48        }
49        else if(opt.equals(Menu.EditBook)){
50          this.tc.start();/*inicia cronometro*/
51          if(b.doLogin(InOut.requestIn(InOut.Senha))){/*contato com
                servidor*/
52            String is = InOut.requestIn(InOut.Isbn);/*requisicao de
                input*/
53            int iq = Integer.parseInt(InOut.requestIn(InOut.
                Quantidade));/*requisicao de input*/
54            out=b.setQuantidade(is, iq);/*contato com servidor*/
55          }
56          else{
57            out="Senha inv lida!\n";
58          }
59          this.tc.stop();/*finaliza cronometro*/
60          this.tc.write("./estat/tc_5");/*escreve estatistica*/
61        }
62        else if(opt.equals(Menu.AmountByIsbn)){
63          this.tc.start();/*inicia cronometro*/
64          out=b.getQuantidade(InOut.requestIn(InOut.Isbn));/*contato
                com servidor*/
65          this.tc.stop();/*finaliza cronometro*/
66          this.tc.write("./estat/tc_6");/*escreve estatistica*/
67        }
68        else if(opt.equals(Menu.RepeatMenu)){
69          out=Menu.getPrint();
70        }
71        else{
72          out="Op   o inv lida!\n";/*oops, opcao invalida*/
73          out=out.concat(Menu.getPrint());
74        }
75      }
76      catch (Exception e) {/*algum erro?*/
77        System.out.println("Error: Lookup server");
78        e.printStackTrace();
79      }
80      System.out.print(out);/*imprime resultado*/
81    }
82
83    public static String getPrint(){
84      String s = "*** MENU (tecle \\m para visualizar o menu)
            *********\n\0";
85      s = s.concat("*** (1) Listar ISBN
                                              *\n\0");
86      s = s.concat("*** (2) Ver descricacao por ISBN (entrada: ISBN)
              *\n\0");
87      s = s.concat("*** (3) Ver info. completa (entrada: ISBN)
                    *\n\0");
88      s = s.concat("*** (4) Ver info. completa (todos os livros)
                  *\n\0");
89      s = s.concat("*** (5) Alterar estoque (apenas adm)
                         *\n\0");
90      s = s.concat("*** (6) Ver quantidade em estoque (entrada: ISBN)
              *\n\0");
91      s = s.concat("*** (\\q) Fechar conexao e sair
                               *\n\0");
92      s = s.concat("
            ************************************************\n\0");
93      return s;
94    }
95  }
```

5.2.5 Usuario.java

```
1  package biblioteca;
2
3  import java.sql.Connection;
4  import java.sql.ResultSet;
5  import java.sql.SQLException;
6  import java.sql.Statement;
7
8
9  public class Usuario {
10   private static String Table="usuario";
11   private Connection conn=null;/*conexao com o banco*/
12   /*necessita de uma conexao com o banco*/
13   public Usuario(Connection c){
14     this.conn=c;
15   }
16   /**
17    * Metodo que realiza o login
18    * @param String s : senha
19    * @return boolean : true em caso de sucesso do login
20    */
21   public boolean login(String s){
22     String qr = "SELECT * FROM "+Usuario.Table+" WHERE senha LIKE '
           "+s+"'";/*query*/
23     Statement st = null;/*statement*/
24     ResultSet r = null;/*resultado da query*/
25     /*tentativa de criar o statement*/
26     try{
27       st = this.conn.createStatement();
28     }catch(SQLException e){/*algum problema?*/
29       System.out.print("Error: getBooks, falha na ao criar
             statement");/*falha*/
30       e.printStackTrace();
31       return false;/*fim*/
32     }
33     /*tentativa de executar a query*/
34     try{
35        r = st.executeQuery(qr);
36        if(!r.next())/*senha nao encontrada*/
37          return false;/*login nao realizado*/
38        return true;/*login realizado*/
39     }catch(SQLException e){/*algum problema?*/
40       System.out.print("Error: getBooks, falha ao executar query");
             /*erro na query*/
41       e.printStackTrace();
42       return false;/*fim*/
43     }
44   }
45 }
```

5.3 Pacote *server*

5.3.1 Server.java

```
1  package server;
2  import java.rmi.Naming;
3  import java.rmi.registry.LocateRegistry;
4  import biblioteca.*;
5  import common.*;
```

```
6  public class Server {
7    public Server(){
8      try {
9        LocateRegistry.createRegistry(CommonVars.port);
10       BibliotecaImpl b = new BibliotecaImpl();
11       Naming.rebind("//"+CommonVars.addr+":"+Integer.toString(
             CommonVars.port)+"/"+CommonVars.appName, b);
12     } catch (Exception e) {
13       System.out.print("Error: start server");/*erro na query*/
14       e.printStackTrace();
15     }
16   }
17   public static void main(String args[]) {
18         new Server();
19     }
20
21   public static String getAddr(){
22     return CommonVars.addr;
23   }
24
25   public static int getPort(){
26     return CommonVars.port;
27   }
28
29   public static String getAppName(){
30     return CommonVars.appName;
31   }
32 }
```

5.4 Pacote *client*

5.4.1 Client.java

```
1  package client;
2  import java.io.IOException;
3  import java.rmi.Naming;
4  import java.io.InputStreamReader;
5  import java.io.BufferedReader;
6  import common.*;
7  import biblioteca.*;
8  public class Client {
9     public static void main(String[] args) {
10     Biblioteca b=null;
11     String str = new String();
12     Menu m = new Menu();
13     try {
14       b = (Biblioteca)Naming.lookup("rmi://"+CommonVars.addr+":"+
             CommonVars.port+"/"+CommonVars.appName);
15     }
16     catch (Exception e) {
17       System.out.println("Error: Lookup server");
18     }
19     try{
20       InOut.writeOut(System.out, b.getMenu());
21       do{
22         BufferedReader buff = new BufferedReader(new
               InputStreamReader(System.in));
23         str = buff.readLine();
24         m.chooseMenu(str, b);
25       }while(!str.equals(Menu.EndProgram));
```

```
26      }
27      catch(IOException ioe){
28        System.out.print("Error: terminal reader");
29        ioe.printStackTrace();
30      }
31    }
32  }
```

5.4.2 Tester.java

```
1  package client;
2  import java.io.IOException;
3  import java.rmi.Naming;
4  import java.io.InputStreamReader;
5  import java.io.BufferedReader;
6  import java.util.Vector;
7  import java.util.Random;
8  import common.*;
9  import biblioteca.*;
10 public class Tester {
11
12   private static Vector<String> opt;/*opcoes do menu*/
13   private static Vector<String> isbn;/*lista com alguns isbn*/
14   private static String senha="mc823";/*senha, sempre correta*/
15   private static int Total = 100;/*mesmo numero de testes para TCP
         e UDP*/
16   /**
17    * Metodo que inicializa as variasveis do teste
18    */
19   private static void init(){
20     Tester.opt = new Vector<String>();
21     Tester.isbn = new Vector<String>();
22     Tester.opt.add(Menu.ListIsbn);
23     Tester.opt.add(Menu.DescByIsbn);
24     Tester.opt.add(Menu.BookByIsbn);
25     Tester.opt.add(Menu.ListBooks);
26     Tester.opt.add(Menu.EditBook);
27     Tester.opt.add(Menu.AmountByIsbn);
28     Tester.isbn.add("081297215-5");
29     Tester.isbn.add("207036002-4");
30     Tester.isbn.add("843760494-X");
31     Tester.isbn.add("850109104-9");
32     Tester.isbn.add("857302773-8");
33     Tester.isbn.add("972011011-3");
34     Tester.isbn.add("978052007-4");
35     Tester.isbn.add("978853650-2");
36     Tester.isbn.add("978857522-3");
37     Tester.isbn.add("978857608-5");
38     Tester.isbn.add("978857679-9");
39     Tester.isbn.add("978858041-1");
40   }
41
42   public static void main(String[] args) {
43     Tester.init();
44     Biblioteca b=null;
45     Random random = new Random();
46     try {
47       b = (Biblioteca)Naming.lookup("rmi://"+CommonVars.addr+":"+
           CommonVars.port+"/"+CommonVars.appName);
48     }
49     catch (Exception e) {
```

```
50        System.out.println("Error: Lookup server");
51      }
52      /*executa testes*/
53      try{
54        Tempo tc = new Tempo();
55        int i;
56        for(i=0;i<Tester.Total;i++){
57          for(String o : Tester.opt){
58            if(o.equals(Menu.ListIsbn)){
59              tc.start();/*inicia cronometro*/
60              System.out.print(b.getAllISBN());
61              tc.stop();/*finaliza cronometro*/
62              tc.write("./estat/tc_1");/*escreve estatistica*/
63            }
64            if(o.equals(Menu.DescByIsbn)){
65              tc.start();/*inicia cronometro*/
66              System.out.print(b.getDescByIsbn(Tester.isbn.get(random
                    .nextInt(Tester.isbn.size()))));
67              tc.stop();/*finaliza cronometro*/
68              tc.write("./estat/tc_2");/*escreve estatistica*/
69            }
70            if(o.equals(Menu.BookByIsbn)){
71              tc.start();/*inicia cronometro*/
72              System.out.print(b.getBookByIsbn(Tester.isbn.get(random
                    .nextInt(Tester.isbn.size()))));
73              tc.stop();/*finaliza cronometro*/
74              tc.write("./estat/tc_3");/*escreve estatistica*/
75            }
76            if(o.equals(Menu.ListBooks)){
77              tc.start();/*inicia cronometro*/
78              System.out.print(b.getBooks());
79              tc.stop();/*finaliza cronometro*/
80              tc.write("./estat/tc_4");/*escreve estatistica*/
81            }
82            if(o.equals(Menu.EditBook)){
83              tc.start();/*inicia cronometro*/
84              b.doLogin(Tester.senha);
85              System.out.print(b.setQuantidade(Tester.isbn.get(random
                    .nextInt(Tester.isbn.size())), random.nextInt(1000)
                    ));
86              tc.stop();/*finaliza cronometro*/
87              tc.write("./estat/tc_5");/*escreve estatistica*/
88            }
89            if(o.equals(Menu.AmountByIsbn)){
90              tc.start();/*inicia cronometro*/
91              System.out.print(b.getQuantidade(Tester.isbn.get(random
                    .nextInt(Tester.isbn.size()))));
92              tc.stop();/*finaliza cronometro*/
93              tc.write("./estat/tc_6");/*escreve estatistica*/
94            }
95          }
96        }
97      }
98      catch(IOException ioe){
99        System.out.print("Error: terminal reader");
100       ioe.printStackTrace();
101     }
102   }
103 }
```