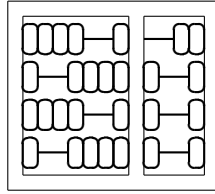


**UNIVERSIDADE ESTADUAL DE CAMPINAS**

**INSTITUTO DE COMPUTAÇÃO**



**SERVIDOR DE AGENDA BASEADO EM SOCKET UDP**

*Relatório do segundo laboratório de MC823*

**Aluno:** Marcelo Keith Matsumoto    **RA:** 085937

**Aluno:** Tiago Chedraoui Silva    **RA:** 082941

### **Resumo**

The Java Remote Method Invocation (RMI) system allows an object running in one Java virtual machine to invoke methods on an object running in another Java virtual machine. RMI provides for remote communication between programs written in the Java programming language.

# Sumário

<b>1</b>	<b>Objetivo</b>	<b>2</b>
1.1	Teoria . . . . .	2
<b>2</b>	<b>Servidor de agenda</b>	<b>2</b>
2.1	Menu inicial . . . . .	3
2.1.1	Login . . . . .	3
2.1.2	Novo usuário . . . . .	3
2.2	Menu usuário . . . . .	3
2.2.1	Inserção de compromisso . . . . .	4
2.2.2	Remoção de compromisso . . . . .	4
2.2.3	Pesquisas . . . . .	4
<b>3</b>	<b>Ambiente de implementação</b>	<b>4</b>
<b>4</b>	<b>Tempos de comunicação e total</b>	<b>4</b>
4.1	Comparação de tecnologias . . . . .	5
<b>5</b>	<b>Conclusão</b>	<b>5</b>
<b>6</b>	<b>Anexo</b>	<b>6</b>

# 1 Objetivo

O objetivoterceiro projeto de laboratório de teleprocessamento e redes é comparar duas implementações distintas do modelo cliente-servidor: Java RMI (Remote Method Invocation) e socket TCP. É de suma importância que utilizando a tecnologia Java RMI, cria-se uma agenda, para possibilitar uma comparação com a mesma agenda em socket TCP, criada anteriormente no projeto 1.

## 1.1 Teoria

Java RMI é uma das abordagens da tecnologia Java, construída para prover as funcionalidade de uma plataforma de objetos distribuídos e com sua API (Application Programming Interface) especificada pelo pacote java.rmi e seus subpacotes. A arquitetura RMI viabiliza a interação de um objeto ativo em uma máquina virtual Java com objetos de outras máquinas virtuais Java.

Aplicações que utilizam objetos distribuídos precisam das realizar as seguintes ações:

**Localização de objetos remotos** aplicações podem usar vários métodos para obter referências a objetos remotos. Ex: utilizar RMI registry

**Comunicação com objetos remotos** Detalhes da comunicação entre objeto remotos são gerenciados pelo RMI, ou seja para o programador chamadas remotas são similares a chamadas de métodos.

**Carregamento de definições de classes para objetos móveis** RMI prove mecanismos para carregar a definição de classes para um objeto assim como para transmitir seus dados

Para o desenvolvimento de uma aplicação cliente-servidor em Java RMI, são necessários um cliente e um para o servidor e a execução do serviço de registro de RMI (RMI registry)(Ver figura figura 1). Um servidor, em geral, instancia objetos remotos, referencia estes objetos e liga-os em uma determinada porta através de um bind, aguardando nesta porta os clientes invocarem os métodos destes objetos. Um cliente, em geral, referencia remotamente um ou mais objetos remotos de um servidor, e invoca os métodos destes objetos. Os mecanismos pelos quais o cliente e o servidor se comunicam e trocam dados são fornecidos pelo Java RMI. O serviço de registro de RMI é uma implementação de um serviço de nomes para RMI, no qual cada serviço disponibilizado na plataforma é registrado através de um nome de serviço, ou seja, uma string única para cada objeto o qual implementa serviços em RMI.

## 2 Servidor de agenda

Para criar uma aplicação distribuída usando a tecnologia RMI deve-se projetar e implementar as componentes da aplicação. Primeiro define as interfaces,depois, baseado nas interfaces, implementa-se os objetos e posteriormente o cliente.

O sistema implementado, uma agenda distribuída, se baseia numa comunicação cliente-servidor. Nele o servidor possui todas as informações da agenda que estão armazenadas em um banco de dados, assim como as opções de interações com os dados que são apresentadas aos clientes em formas de um menu. O cliente só escolhe alguma opção de interação com os dados de acordo com menu.

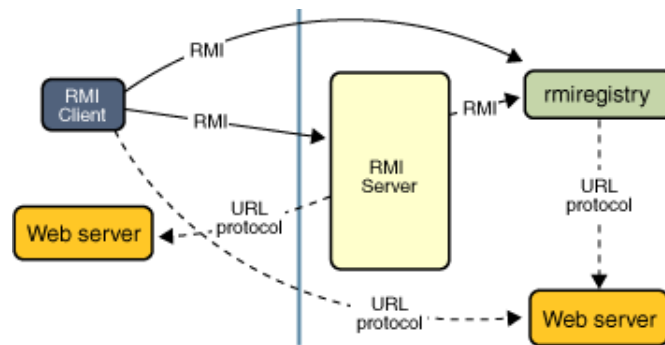


Figura 1: Aplicação distribuída RMI que usa o RMI registry para obter a referência para um objeto remoto.

## 2.1 Menu inicial

No menu inicial pode-se:

- Logar
- Criar novo usuário
- Sair

### 2.1.1 Login

O servidor pede ao usuário o nome de usuário, caso o nome estiver no banco de dados ele pede uma senha que é comparada ao valor do banco de dados, se o usuário não existir é avisado sobre a inexistência, se a senha não conferir é avisado que a senha não confere, caso contrário o usuário consegue logar no sistema, e o servidor recupera sua agenda (cada usuário possui sua agenda).

### 2.1.2 Novo usuário

O servidor pede um nome de usuário, o servidor verifica se o nome já não existe, se não existir pede a senha e armazena o usuário no sistema, assim como cria uma agenda vazia para o mesmo.

## 2.2 Menu usuário

Dentre as possibilidades de interações para um usuário logado tem-se:

- Inserção de um compromisso que possui um nome, dia, hora, e minuto.
- Remoção de um compromisso através de seu nome
- Pesquisa de compromisso por dia
- Pesquisa de compromisso por dia e hora
- Ver todos os compromisso de mês de abril

### 2.2.1 Inserção de compromisso

O usuário deve fornecer o nome do compromisso, o dia, a hora e o minutos em que ele ocorrerá. Caso o compromisso seja possível de ser alocado o servidor avisa com um “OK”, se não for possível também é avisado de tal impossibilidade. Um compromisso é inserido ordenado na agenda se não existir um compromisso com mesmo horário.

### 2.2.2 Remoção de compromisso

O usuário deve fornecer o nome do compromisso que deve ser removido. Caso o compromisso seja encontrado ele é removido, caso contrário é dito que tal compromisso não existe. Se existirem dois compromissos de mesmo nome, o primeiro é removido. Logo é esperado que compromissos possuam nomes diferentes.

### 2.2.3 Pesquisas

O servidor faz um requerimento interativo, ou seja, se for selecionado a pesquisa por dia e hora, o servidor pergunta primeiramente o dia e depois a hora. Logo, é uma pesquisa em etapas no qual o servidor interage com nosso usuário.

## 3 Ambiente de implementação

O sistema de agenda foi implementado e executado nos seguintes sistemas operacionais :

- FC14 - Fedora Laughlin Linux 2.6.35.11
- Mac OS X 10.6.7

O sistema de agenda foi implementado na linguagem Java, utilizando a tecnologia RMI. Para o armazenamento dos dados, utilizou-se arquivos. Cada usuário possui um arquivo, a sua agenda, no qual armazena-se o nome do compromisso, o dia, a hora e o minuto do mesmo. O sistema lê a agenda a cada função chamada o servidor atualiza as informações dos arquivos.

O nosso sistema, além disso, apresenta transparência ao usuário. Os tipos de transparência a serem destacados são:

**Acesso:** Esconde as diferenças nas representações de dados e na invocação de funções ou métodos para facilitar a comunicação entre objetos da rede.

**Localização:** Esconde o local em que o objeto se encontra.

**Concorrência:** Esconde como as atividades são coordenadas entre os objetos para obter consistência em um nível mais alto.

## 4 Tempos de comunicação e total

Aplicamos o cálculo de tempo ao programa principal de forma a obtermos o tempo total, tempo de comunicação e os tempos da execução de cada função. Para o tempo total, no cliente pega-se o tempo antes da chamada da função do servidor e após o retorno dessa função.

Para o tempo de comunicação, pega-se o tempo total e subtrai-se o tempo de processamento do servidor, que é depois da chamada da função e antes do retorno da função. Para o tempo total das funções obteve-se o tempo de inserir um compromisso, remover o compromisso, ver a agenda do mês, ver a agenda de um dia, ver a agenda de uma hora. Os dados e os testes estão exemplificados nas tabelas seguintes:

#### 4.1 Comparação de tecnologias

O RMI utiliza o protocolo TCP, que uma das suas características é a transferência garantida, assim não foi necessário uma análise de erros na entrega dos pacotes. O que possibilitou uma diminuição do código se comparado a utilização do protocolo UDP em C.

Ao utilizar a tecnologia RMI conseguiu-se uma grande abstração em relação a comunicação entre cliente e servidor, já que a após estabelecida a comunicação o servidor é chamado através de funções como se não fossem distribuídas. Da mesma maneira os arquivos são vistos como se fossem locais, o que é uma característica de transparência de localização, um dos objetivos de um sistema distribuído.

### 5 Conclusão

Utilizar a tecnologia Java RMI facilitou o desenvolvimento de aplicações distribuídas, no qual existe a interação entre um cliente e um servidor, devido a inclusão da implementação do protocolo TCP. Além disso, Java proporciona a funcionalidade garbage collector que nos exime de se preocupar com a limpeza de memória, diferentemente do que ocorreu desenvolvendo a agenda na linguagem C.

Por outro lado, existe a necessidade de uma largura de banda consideravelmente maior em relação ao Socket TCP. Entretanto, como a tecnologia Java RMI tem como objetivo fornecer uma transparência de localização e não a eficiência no transporte de dados, o que permitindo um maior nível de abstração e de transparência, auxiliando o programador; a utilização de Java RMI tem uma relação custo-benefício muito boa.

Apesar dos benefícios, escrever código em Java requer um maior conhecimento de orientação a objetos, e o seu desempenho é pior se comparado à linguagem C.

### Referências

- [1] Tanenbaum, Andrew S. e Maarten Van Steen Distributed Systems: Principles and Paradigms. Prentice Hall.
- [2] Brian "Beej" Jorgensen "Hall" Beej's Guide to Network Programming - Using Internet Sockets . Disponível em <http://beej.us/guide/bgnet/>, [Último acesso: 07/04/2011].
- [3] Tutorial RMI Oracle. Disponível em <http://download.oracle.com/javase/tutorial/rmi/index.html>, [Último acesso: 12/05/2011].
- [4] J. Kurose e K. Ross. *Computer Networking: A Top-Down Approach Featuring the Internet*. Pearson Addison Wesley, 3 ed., 2005.

**6 Anexo**