Problema 4 - Ferrovia compartilhada

Diego J. Leite, Ênio M. Costa e João Victor M. do Nascimento

[1]Departamento de Tecnologia – Universidade Estadual de Feira de Santana (UEFS)
Caixa Postal 44.036-900 – Feira de Santana – BA – Brazil

diegojleite@gmail.com, eniiocosta@gmail.com, victormota15@gmail.com

Abstract. *This article is related to the fourth problem of discipline MI-Competition and Connectivity Course of Computer Engineering, of the Universidade Estadual de Feira de Santana, and it is a the product of a proposed problem in tutorials sessions that aims to understand the functioning and synthesize a system of control for several shared rails between tree trains. The solution for this problem will be presented through of utilization of distribuited systems, where the messages that are exchanged over the network, among stakeholders, have additional layers of security provided by techniques of encryption and the remote method invocation system (RMI).*

Resumo. *Este artigo é referente ao quarto problema da disciplina MI-Concorrência e Conectividade do curso de Engenharia de Computação, da Universidade Estadual de Feira de Santana, e trata-se do produto de um problema proposto em sessões tutoriais que tem como finalidade compreender o funcionamento e sintetizar um sistema de controle para vários trilhos compartilhados entre três trens. A solução para este problema será apresentada através da utilização de sistemas distribuídos, onde as mensagens que são trocadas pela rede, entre os envolvidos, contam com camadas adicionais de segurança providas por técnicas de criptografia e com o sistema de invocação remota de métodos (RMI).*

1. Introdução

Projetos bem planejados assumem possibilidades de ajustes futuros e melhorias relacionadas ao produto final. Modificações podem alterar a estrutura final de um projeto concluído e adicionar características ou comportamentos ao mesmo. Em ferrovias, mudanças podem estar relacionadas com a aquisição de novas linhas de trem ou apenas com a inclusão de novos trilhos em percursos existentes.

Depois que a linha de metrô compartilhada foi construída, e colocada em funcionamento [Leite and Costa 2013], o serviço a possou a ser muito requisitado pela população da cidade. Para atender a todos os adeptos deste novo meio de locomoção, o prefeito resolveu ampliar a capacidade de atendimento da ferrovia da cidade, através da adição de uma nova linha de metrô, incorporando-a com as já existentes. Esta nova linha utilizará as outras linhas existentes para trafegar os seus veículos, através do compartilhamento dos trilhos da ferrovia. Os trens percorrerão o seu percusso no sentido horário, proporcionando o ocorrimento de colisões onde os trilhos que são compartilhados.

As colisões deverão ser controladas através de mecanismos de controle de velocidade durante a utilização dos trilhos compartilhados. Para isto, o sistema contará com

regras de prioridade entre os trens, onde o trem prioritário possui a capacidade de alterar a velocidade dos trens que estão circulando na ferrovia da cidade. A comunicação continuará a ser não centralizada, ou seja, não existirá um servidor central que responderá as requisições de cada trem em circulação nas linhas de metrô.

As informações que serão trocadas entre os trens envolverão um alto gral de periculosidade, exigindo a necessidade de implantação de técnicas de segurança durante a comunicação entre os metrôs. A mudança de velocidade realizada pelo trem prioritário, nos outros trens, será efetivada com o advento de técnicas de criptografia, onde as mensagens que serão trocadas entre os envolvidos necessitarão de camadas de segurança para proteger essas informações.

O sistema de metrô será controlado distribuidamente entre os trens, ou seja, cada trem disponibilizará suas informações para outro trem existente na linha de metrô, descartando a necessidade de implantação de um servidor ou controlador central. As informações enviadas e coletadas na linha de metrô serão determinantes para que apenas um trem tenha acesso ao trilho compartilhado, bloqueando o acesso ao trilho até que ele seja inteiramente desocupado.

Este relatório fora subdividido em quatro seções, são elas: Fundamentação teórica (2): noções teóricas que servem ao trabalho experimental realizado; Metodologia (4): o que foi feito para resolver o problema; Resultados e discussões (5): soluções encontradas e desenvolvidas para a resolver o problema; Conclusão (6): ideias conclusivas e possíveis melhorias que podem ser desenvolvidas para implementações futuras.

2. Fundamentação teórica

Esta seção está subdividida em quatro subseções, todas elas contendo informações relevantes para o acompanhamento da leitura do relatório: Seção 3, *Rivest-Shamir-Adleman cryptosystem*; Seção 3.1, RC4; Seção 3.2, *Remote Method Invocation*.

3. *Rivest-Shamir-Adleman cryptosystem*

O sistema de criptografia *Rivest-Shamir-Adleman* (RSA) é conhecido como o primeiro modelo de encriptação de dados utilizando o conceito de chave pública, sendo largamento utilizado na segurança de informações a serem transmitidas.

Em um sistema de criptografia assimétrica, os interessados em compartilhar informações de forma segura geram, através de cálculos matemáticos [Milanov 2009], uma chave privada e outra pública (Figura 1) para que os dados possam ser encriptados e decriptados respectivamente.

Caso Alice (Figura 2), detentora dos dados a serem compartilhados, queira enviar informações encriptadas para Bob, é necessário que ela, através do algoritmo RSA, gere uma chave privada e outra pública. A chave privada servirá para que os dados de Alice sejam devidamente criptografados. Os dados chegarão para Bob, mas sem sentido, pois estão criptografados. Então, é necessário Alice envie também a chave pública que foi gerada juntamente com a chave privada que foi utilizada para encriptar os dados. Em posse da chave pública, Bob poderá decriptar a informação recebida e fazer a leitura do conteúdo que foi originalmente gerado por Alice.

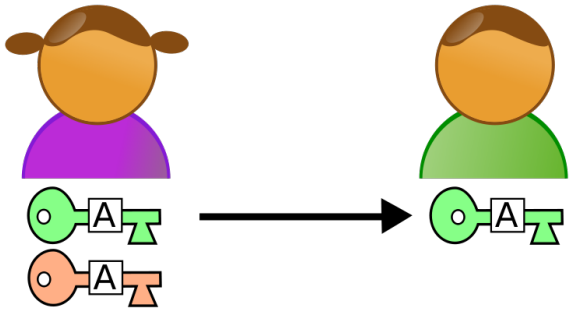

Figura 1. Alice gera duas chaves, uma pública e outra privada, mas envia apenas a chave pública para Bob [Kozlowski 2007a].

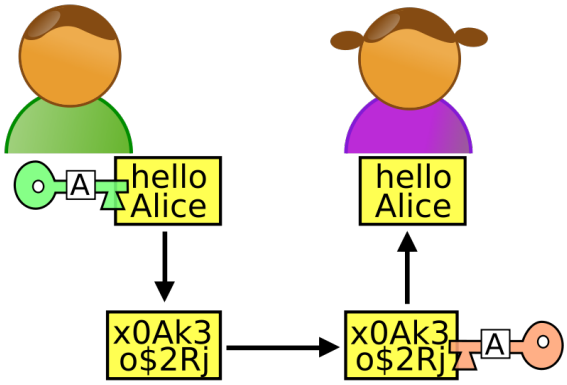

Figura 2. Alice envia os dados e a chave pública para Bob, para que ele possa ler o conteúdo original da informação enviada [Kozlowski 2007b].

3.1. RC4

O RC4 é um algoritmo de encriptação simétrica, onde cada *bit* do dado é sequencialmente encriptado através de um bit da chave gerada [Mousa and Hamad 2006]. Este algoritmo é utilizado também para decriptar *steams* de dados através de uma função XOR entre a chave simétrica gerada e o dado e ser decriptado.

3.2. *Remote Method Invocation*

O *Remote Method Invocation* (RMI) é uma abstração utilizada em *Java*[1] para a comunicação em rede. Com o advento do RMI é possível que um objeto, em estado de execução, em uma máquina virtual Java invoque métodos de outra máquina virtual Java [Oracle 1999] .

Um sistema RMI é constituído por três elementos: cliente RMI, servidor RMI e um servidor de registro [Gouveia 2008] . Onde, o cliente RMI acessa os métodos, disponibilizados em uma interface de acesso no servidor de registro, de outro objeto Java através da rede, representando o servidor RMI (Figura 3).

[1]Referência para a linguagem Java (pacote java.lang): http://docs.oracle.com/javase/7/docs/api/java/lang/package-summary.html

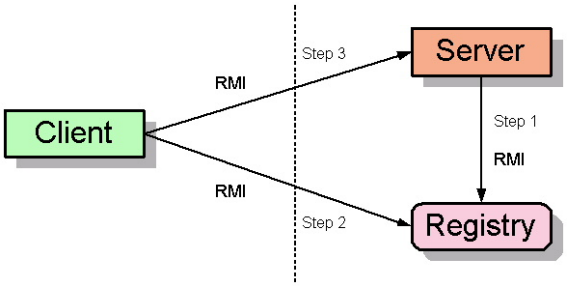

Figura 3. Cliente acessa os métodos do servidor através do servidor de registro [Zheng 1999].

4. Metodologia

O projeto foi desenvolvido utilizando o ambiente integrado de desenvolvimento *Eclipse*[2]. A linguagem de programação escolhida foi *Java SE*[3]. O versionamento do código foi feito através do sistema de controle de versões *Git*[4], possibilitando o trabalho em equipe no mesmo código, sem que limitações de disponibilidade dos envolvidos afetassem no processo de desenvolvimento do projeto.

A implementação da proposta de solução foi realizada utilizando conceitos de sistemas distribuídos *peer-to-peer* [Leite and Costa 2013], pois neste problema seria necessário que cada instância da aplicação desempenhasse o papel de Cliente e de Servidor. A comunicação entre os trens conta com criptografia entre as mensagens que são trocadas.

Foi necessário representar graficamente a solução através de ferramentas disponibilizadas pelo *Java2D* (Seção 2.4), melhorando a percepção de quem analisasse a proposta de solução desenvolvida para o problema.

5. Resultados e discussões

Para definir o trem com prioridade no sistema o usuário deverá informar uma senha no momento de inicialização (Figura 4), esta senha é única e está salva de maneira criptografada (Seção 2.2) no código fonte do programa. Caso o usuário não possua senha deve-se entrar em contato com a empresa responsável pelo sistema já que não existe a possibilidade de efetuar o cadastro de uma nova.

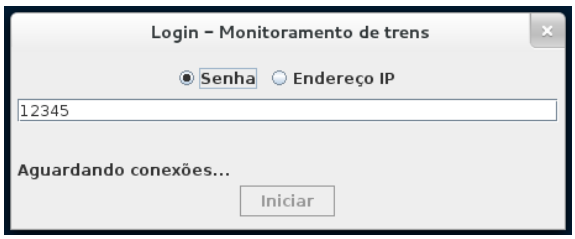

Figura 4. Painel de conexão com os outros trens da ferrovia.

[2]Site do projeto: http://eclipse.org/

[3]Ducumentação da linguagem: http://www.oracle.com/technetwork/java/javase/documentation/index.html

[4]Site do projeto: http://git-scm.com/

Após efetuar o login o usuário recebe a mensagem de que o sistema esta aguardando as conexões (Figura 4). Neste momento o *host*, onde o usuário iniciou o programa, cria dois *sockets* para aceitar a conexão dos outros trens. Assim que um trem se conecta, é feita uma verificação para identificar se existem outros trens conectados ao trem prioritário, caso exista o endereço desses trens é enviado pelo trem prioritário para que o novo trem possa também se comunicar com os demais, caso não exista nenhuma outra conexão o novo trem se conecta ao primeiro e abre um *socket* para aceitar a conexão do próximo Cliente, que neste caso é o terceiro trem. Depois que a comunicação entre os três trens é estabelecida, a chave simétrica (Seção 2.2) é gerada (Figura 5) e enviada para todos.

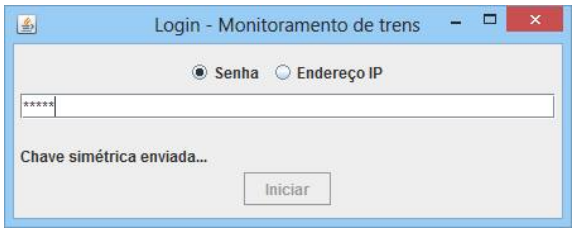

Figura 5. A chave simétrica é gerada e enviada para o trens.

Durante a inicialização do sistema a interface mostra ao usuário através de mensagens a situação atual da conexão. Após estabelecida a conexão dos três trens uma nova janela é aberta (Figura 6), nesta janela é desenhado o trajeto dos trens e sua posição atual, sendo essa posição atualizada a todo instante.

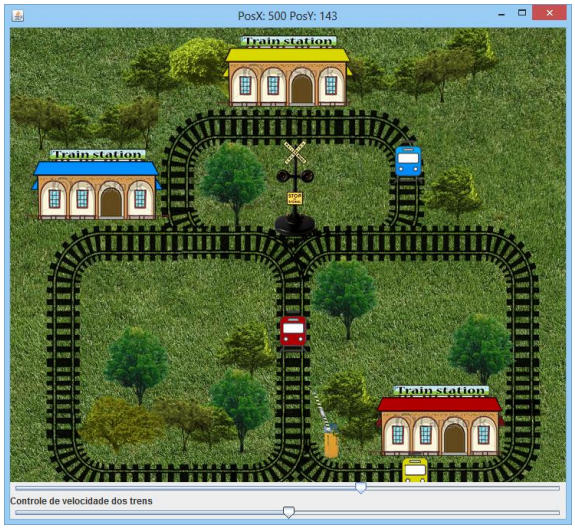

Figura 6. Janela que apresenta a posição inicial de todos os trens existentes na ferrovia.

6. Conclusão

Utilizando *peer-to-peer* [Leite and Costa 2013], foi possível conectar diversos *hosts* sem que um conhecesse o endereço de todos os outros. A arquitetura desenvolvida se mostrou eficiente neste quesito, já que o terceiro trem estabelece uma conexão com o segundo mesmo sem saber o seu endereço.

Todas as funcionalidades foram implementadas exceto a possibilidade de alteração da velocidade de um determinado trem pelo metrô prioritário através do RMI (Seção 2.3). A comunicação entre os trens ocorre de maneira segura através da criptografia das mensagens utilizando chaves assimétricas (Seção 2.1) e simétricas (Seção 2.2).

Algumas melhorias poderiam ser implementadas no sistema, visando uma melhor comunicação entre os trens. Poderia ser implementado um chat, para que cada maquinista de trem pudesse enviar mensagens informativas para outros trens em circulação na ferrovia. A mudança de curso, por parte do trem, seria uma possibilidade interessante de ser implementada em versões futuras deste produto, pois esta é uma situação presente no mundo real.

Referências

Gouveia, D. (2008). *Java em Rede - Programação Distribuída na Internet*. Brasport.

Kozlowski, W. (2007a). Asymetric cryptography - Step 1. Disponível em: <http://pt.wikipedia.org/wiki/Ficheiro:Asymetric_cryptography_-_step_1.svg>. Acesso em: 12 dez. 2013.

Kozlowski, W. (2007b). Asymetric cryptography - Step 2. Disponível em: <http://pt.wikipedia.org/wiki/Ficheiro:Asymetric_cryptography_-_step_2.svg>. Acesso em: 12 dez. 2013.

Leite, D. and Costa, E. (2013). Problema 3 - Metrôs compartilhados (Relatório).

Milanov, E. (2009). The RSA Algorithm. Disponível em: <http://www.math.washington.edu/m̃orrow/336_09/papers/Yevgeny.pdf>. Acesso em: 12 dez. 2013.

Mousa, A. and Hamad, A. (2006). Evaluation of the RC4 Algorithm for Data Encryption. Disponível em: <http://pt.wikipedia.org/wiki/Ficheiro:Asymetric_cryptography_-_step_2.svg>. Acesso em: 12 dez. 2013.

Oracle (1999). Trail: RMI. Disponível em: <http://docs.oracle.com/javase/tutorial/rmi/>. Acesso em: 12 dez. 2013.

Zheng, P. (1999). An Brief Untroduction to RMI. Disponível em: <http://www.cs.cmu.edu/afs/cs.cmu.edu/user/pzheng/www/KDI-IIM/RMI_intro.html>. Acesso em: 12 dez. 2013.