JAVA MESSAGE SERVICE, UMA ALTERNATIVA ENTRE COMUNICAÇÃO DE SISTEMAS: uma abordagem prática

Lucas Yokowo dos Santos [1]

RESUMO

Mesmo com a consolidação de protocolos de comunicação via rede no mercado, como o File Transfer Protocol (FTP), a comunicação entre sistemas de software sempre foi uma tarefa desafiadora. O Java Message Service (JMS) surgiu com esse mesmo intuito, para que aplicativos escritos na linguagem Java pudessem criar, receber e enviar mensagens de outros aplicativos. A empresa de sistemas de controle viários VELSIS escolheu de começo o FTP como meio de comunicação entre seus equipamentos. Porém, diante as restrições encontradas devido a essa tecnologia a empresa sentiu a necessidade de utilizar alternativas mais estáveis. Este artigo apresenta os resultados e a experiência obtida pela empresa dada a migração do FTP para o JMS como meio de comunicação entre sistemas. Pontos positivos e negativos serão abordados, bem como algumas possíveis soluções para os problemas que vieram a surgir.

Palavras-chave: Interconectividade, FTP, JMS

1 INTRODUÇÃO

A comunicação entre diferentes sistemas de software sempre foi uma tarefa desafiadora. Com o surgimento das redes de computadores, a comunicação entre mainframes abriu caminho para a computação distribuída. Depois com a popularização dos computadores pessoais e o consequente aumento no número de aplicações desenvolvidas em diferentes sistemas operacionais e linguagens de programação, a interconectividade entre aplicações passou a ser ainda mais complexa, envolvendo diferentes plataformas de hardware, software, protocolos e formatos de dados. Diante deste cenário, diferentes modelos arquiteturais, técnicas e tecnologias têm sido propostas para a cooperação e troca de informações entre sistemas distribuídos em rede.

Um dos padrões definidos de transferência de arquivos entre computadores é o FTP (File Transfer Protocol), sendo este um dos primeiros padrões estabelecidos. O principal objetivo na formulação do File Transfer Protocol foi fazer transferências de arquivos simples

[1] Programa de Pós-Graduação em Desenvolvimento Web e Dispositivos Móveis, Faculdade Sant'Ana, Ponta Grossa (PR) – Brasil. E-mail: lucasyokowo@gmail.com

e para aliviar a carga do usuário de aprender os detalhes sobre a forma como a transferência seja efetivamente cumprida.

Nesse mesmo intuito surgiu o JMS (Java Message Service). Projetado pela Sun MicroSystems com o apoio de empresas associadas, e lançado para o mercado em agosto de 1998 o JMS permitiu que os aplicativos escritos na linguagem Java pudessem criar, receber e enviar mensagens destinadas ou oriundas de outros aplicativos. A principal característica deste tipo de processamento, classificado como fracamente acoplado, é que todas as operações que envolvem a troca de mensagens são feitas de forma assíncrona, fazendo com que as aplicações participantes não precisem ficar bloqueadas esperando o término de alguma computação remotamente solicitada.

Os sistemas de controle viário no Brasil passaram por uma profunda mudança quando se intensificou o uso de novas tecnologias que permitiam o controle e fiscalização de veículos de forma automatizada e precisa. E é nesse ambiente que surge a VELSIS - Sistemas e Tecnologia Viária, buscando oferecer ao mercado a melhor solução em mobilidade urbana. Por durante muito tempo a empresa VELSIS usou o FTP como principal meio de comunicação entre seus equipamentos (radares) e seu servidor central. O principal problema encontrado com o uso deste protocolo foi garantir as entregas das informações. Os equipamentos possuem comunicação 3G, por ser uma conexão lenta e muitas vezes instável, tornava pouca confiável a troca de informações via FTP. Diversas operações foram feitas para garantir uma troca confiável de informações muitas vezes tendo de se repetir o processo de comunicação por completo exigindo assim ainda mais da conexão.

A fim de resolver esses problemas a VELSIS estudou e aplicou o JMS como um novo meio de comunicação entre seu servidor central e seus equipamentos. O objetivo deste artigo é apresentar os resultados e a experiência profissional obtida por essa migração de tecnologia tantos os pontos positivos quanto os negativos e apresentar algumas possíveis soluções para os novos problemas que vieram a surgir.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 File Transfer Protocol

O protocolo FTP (File Transfer Protocol) é como o seu nome o indica um protocolo de transferência de arquivos. O FTP Provê serviços de transferência, renomeação e eliminação de arquivos, além da criação, modificação e exclusão de diretórios. Para sua

operação, são mantidas duas conexões: uma de dados e outra de controle. Não implementa segurança, o que deixa para o Transmission Control Protocol (TCP), exceto as requisições de senhas de acesso a determinados arquivos (ou servidores FTP).

As transferências de arquivos podem ser no modo TEXTO (arquivos ASCII), onde há conversões de codificação para o sistema destinatário, e o modo BINÁRIO (arquivos executáveis), onde não há nenhuma conversão e todos os bytes são transferidos como estão.

O protocolo FTP inscreve-se num modelo cliente-servidor, ou seja, uma máquina envia ordens (o cliente) e a outra espera pedidos para efetuar ações (o servidor). Ele permite acesso de outros usuários a um disco rígido ou servidor. Esse tipo de servidor armazena arquivos para dar acesso a eles pela rede.

2.2 Middleware Orientado a Mensagens

Um middleware orientado a mensagens (MOM) pode ser descrito como uma categoria de software cujo objetivo principal é permitir a troca de mensagens entre aplicações distribuídas de maneira assíncrona, escalável, segura e confiável. Os principais benefícios de usar um MOM são poder trocar mensagens em modo assíncrono e obter uma arquitetura desacoplada. Vai bem a muitos casos de uso como, por exemplo, distribuir processamentos demorados em várias máquinas. Um MOM age como um mediador (também chamado de broker) entre aplicações que enviam e recebem mensagens. Internamente estas mensagens são mantidas no que é chamado de destino (destination), uma espécie de repositório para o qual as mensagens são encaminhadas assim que são recebidas pelo MOM, e onde ficam até que sejam entregues à aplicação.

As aplicações que trocam mensagens através do MOM o fazem sempre por meio de destinos específicos; isto é, tanto a aplicação remetente quanto a aplicação destinatária conhecem previamente o destino para onde a mensagem será enviada e de onde será consumida, respectivamente.

Apesar das vantagens de se trabalhar com MOMs ele possui 2 principais desvantagens. Sendo elas:

1) Camada adicional entre enviador e o recebedor de mensagens, podendo causar uma perda de performance no processo.;

2) Centralização em um único ponto introduz risco de falha de todo sistema.

2.3 Java Message Service

O JMS permite que os aplicativos escritos na linguagem Java criem, recebam e enviem mensagens de outros aplicativos. A principal característica deste tipo de processamento é que todas as operações que envolvem a troca de mensagens são feitas de forma assíncrona, através de MOMs, fazendo com que as aplicações participantes não precisem ficar bloqueadas esperando o término de alguma computação remotamente solicitada.

Esta API fornece um conjunto de interfaces que isolam o programador Java (implementando os produtores e consumidores de mensagem) a partir dos provedores MOM.

O JMS utiliza dois modelos básicos de conectividade:

1) Filas ponto-a-ponto (queues), onde mensagens submetidas por uma aplicação "produtora" são entregues a uma única aplicação "consumidora". Pode haver vários consumidores conectados à mesma fila, neste caso somente um deles receberá cada mensagem;

2) Canais publish/subscribe (topics), onde cada mensagem pode ser recebida simultaneamente por diversas aplicações consumidoras.

A comunicação assíncrona facilita criar aplicações com características como:

• Alta concorrência: a comunicação assíncrona implica que a aplicação A não precisa ficar parada enquanto B recebe uma mensagem, há maior potencial de paralelismo.

• Tolerância a falhas: se a aplicação B ficar temporariamente indisponível ou talvez sobrecarregada, isso não impede A de funcionar, nem mesmo deixa A com alguns recursos bloqueados inutilmente enquanto aguarda B – o que reduz a carga e o risco de falhas também no servidor de A.

• Escalabilidade e balanceamento: transações complexas que envolvem vários sistemas são naturalmente divididas em fatias pequenas e independentes, permitindo à conseguir melhor paralelismo.

2.3 Apache ActiveMQ

Para a realização de testes e execução de aplicações JMS o MOM escolhido foi o Apache ActiveMQ. Um open-source que facilita a integração com outras aplicações, pois ele é uma implementação (sendo um Provider) de sucesso da arquitetura message oriented middleware. O exemplo descrito neste artigo demonstra a integração com esse.

O ActiveMQ é rápido e suporta o cruzamento de muitos clientes, linguagens de programação e protocolos. Esta comunicação suporta recursos como computer clustering(múltiplos computadores trabalhando com um único sistema) , capacidade de usar qualquer banco de dados como um provedor de persistência JMS, além de memória virtual e cache.

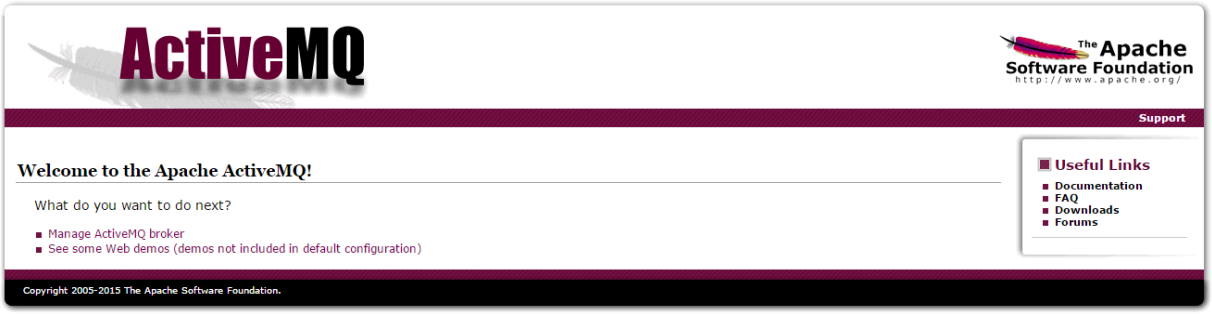

Figura 1 - Interface do Servidor ActiveMQ

3 METODOLOGIA

A comunicação JMS entre os equipamentos e o sistema central seguiu a seguinte estrutura (Figura 2).

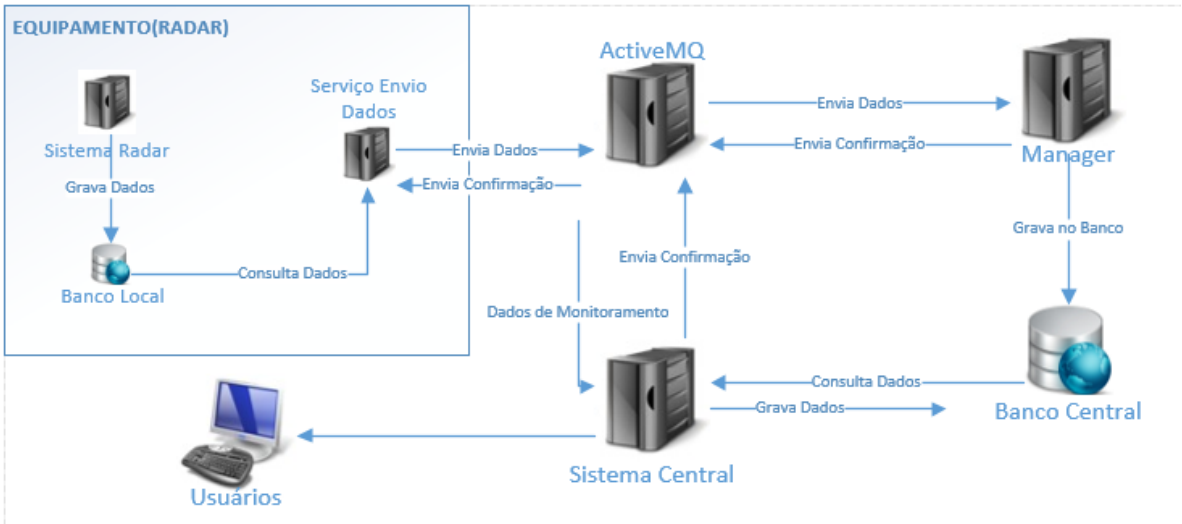

Figura 2 - Arquitetura de Comunicação via JMS

No equipamento temos então o sistema do radar desenvolvido em Pascal/Delphi, esse sistema é responsável pela captação e geração dos dados, este sistema trabalha com um banco que roda localmente. Neste mesmo equipamento roda um serviço em JAVA responsável pelo envio de informações via JMS, informações essas que ficam contidas no banco local que foram geradas pelo sistema do radar. Este serviço também recebe

informações via JMS com confirmações de envio, desse modo garantimos a confiabilidade do envio da informação.

Todas as mensagens enviadas pelos equipamentos passam pelo servidor do ActiveMQ. Este MOM fica responsável por todo gerenciamento de envio destas mensagens. Todas as mensagens antes de serem recebidas passam por este MOM, é ele que garante a comunicação assíncrona e o paralelismo das trocas de informação. Não havendo a disponibilidade do recebimento, ele guarda a mensagem em cache para envio posterior.

Temos então dois sistemas desenvolvidos em JAVA que recebem essas informações. O Manager que recebe os dados a serem gravados no banco e o Sistema Web que é responsável por receber dados relacionados ao monitoramento dos equipamentos (status). Estes dois sistemas recebem dados de todos os equipamentos e envia as confirmações para os mesmos.

Cabe ao Sistema Web interpretar e gerenciar as informações ele fica responsável tanto pela parte de monitoramento dos radares(status de cada um) quanto pela geração dos relatórios e pela validação dos dados no banco.

O principal desafio desse caso de uso é o grande fluxo de dados enviados ao servidor ActiveMQ, visto que podemos ter um numero de equipamentos ilimitados comunicando simultaneamente (Figura 3).

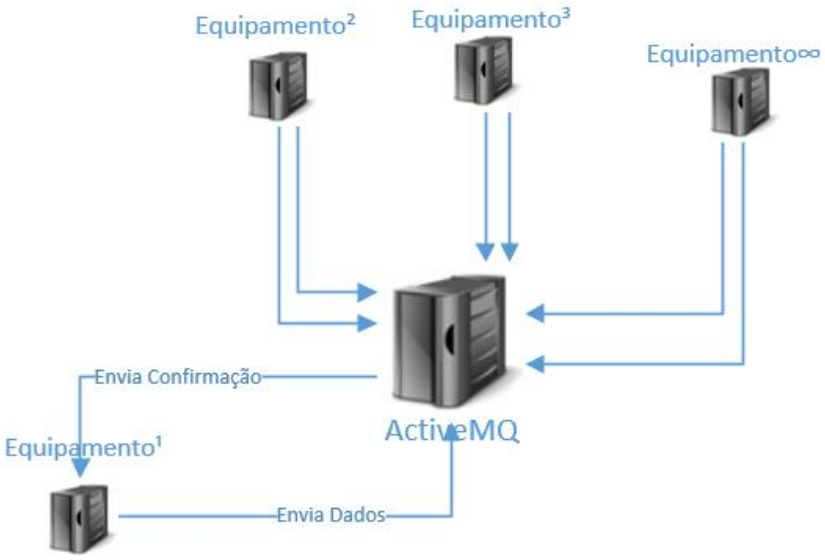

Figura 3 – Quantidade de Equipamentos

O serviço de envio de informações, o sistema web e o manager foram desenvolvidos em JAVA. Para comunicar via JMS foi implementado classes emissora e receptora de mensagens que usam objetos JMS[2] . (Figura 4)

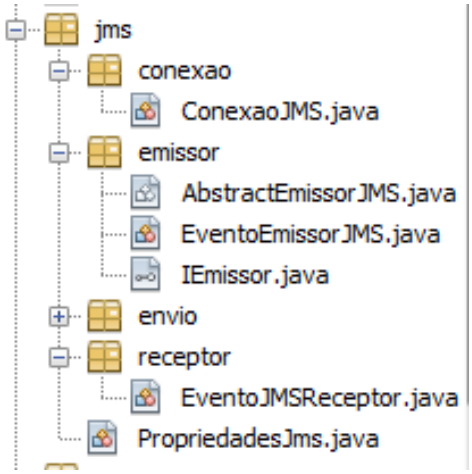

Figura 4 - Estrutura das Classes de comunicação via

Para conexão foi usado a classes do ActiveMQ[3] disponível pela apache. (Figura 5)

```
import org.apache.activemq.ActiveMQConnection;
import org.apache.activemq.ActiveMQConnectionFactory;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

public class ConexaoJMS {

    private static final Logger logger = LoggerFactory.getLogger(ConexaoJMS.class);
    private static ActiveMQConnectionFactory factory;
    private static ActiveMQConnection connection;
```

Figura 5 - Classe que cria conexão com o ActiveMQ.

4 RESULTADOS

Este tipo de comunicação resolveu o problema de confiabilidade na entrega das informações. Esta estrutura também resolveu o problema de quebra da mensagem devido à instabilidade da rede 3G que o equipamento utiliza. Porém não resolveu o problema do alto fluxo de informações via rede que causa lentidão no recebimento das informações. O ActiveMQ demonstrou uma certa incapacidade no gerenciamento, pois por muitas vezes ele ficou congestionado e começou a receber mais mensagens do que é capaz de enviar.

[2] Biblioteca javax.jms
[3] Biblioteca org.apache.activemq

5 DISCUSSÃO

Um dos principais requisitos deste Sistema é a troca de mensagens de maneira rápida para que o monitoramento dos equipamentos possa ser feitos em tempo real e os problemas sejam resolvidos logo quando aparecem. Este alto fluxo de mensagens ocorre devido ao grande numero de equipamentos enviando mensagens ao mesmo tempo.

Temos duas soluções possíveis para esse problema:

1) A clusterização do ActiveMQ. Desse modo aumentamos o processamento e a capacidade troca de mensagens paralelamente.

2) A troca do ActiveMQ. Essa segunda opção não garante uma melhora, fica dependente do desempenho do novo MOM. Algumas opções que temos de troca são RabbitMQ e o Kafka.

Apesar da incapacidade do gerenciamento de mensagens que ActiveMQ demonstrou diante do caso do uso em questão ele apresentou ser bem estável em relação a confiabilidade na entrega da informação.

Como solução inicial a clusterização vem a ser mais aceitável, considerando o fato de que não se necessitaria fazer uma migração de tecnologia. Em caso de troca de MOM, teria de ser feito um estudo de desempenho comparando as duas tecnologias. Se algum MOM apresentar melhores resultados poderia então ser feito a troca.

6 CONCLUSÃO

A comunicação JMS demonstrou grande potencial na confiabilidade para entrega de mensagens. Porem o uso de MOM (ActiveMQ) causou a centralização em um único ponto o que introduziu falha devido ao grande fluxo de dados.

AGRADECIMENTOS

À Faculdade Sant'Ana e ao Instituto Döll de Ponta Grossa, pelos auxílios e conhecimentos oferecidos durante a Pós-Graduação de Desenvolvimento Web e Dispositivos Móveis. E à empresa Velsis que me proporcionou as experiências e as informações necessárias para o desenvolvimento deste artigo.

JAVA MESSAGE SERVICE, AN ALTERNATIVE COMMUNICATION BETWEEN SYSTEMS: a practical approach

Lucas Yokowo dos Santos

ABSTRACT

Even with the consolidation of communication protocols network on the market, such as File Transfer Protocol (FTP), communication between software systems has always been a challenging task. The Java Message Service (JMS) came up with the same order, so that applications written in the Java language could create, send and receive messages from other applications. The company of road control systems VELSIS chose to start with FTP as a protocol communication between their equipment. However, given the constraints encountered due to this technology the company felt the need for more stable alternatives. This article presents the results and the experience gained by the migration of FTP to JMS as a communication between systems. The positives and negatives points will be presented and some possible solutions to the problems that came to

Keywords: Interconnectivity, FTP, JMS.

REFERÊNCIAS

Apache ActiveMQ – Features. Disponível em: <http://activemq.apache.org/features.html>. Acesso em 23 de setembro de 2015.

API Java JMS. Disponivel em: <http://blog.concretesolutions.com.br/2012/01/api-java-jms-parte-1/>. Acesso em: 10 de setembro de 2015.

A Velsis. Disponível em: <http://www.velsis.com.br/empresa.html>. Acesso em 15 de Setembro de 2015.

Conceito: JMS (Java Messaging Service). Disponível em: <http://walderson.com/IBM/RUP7/LargeProjects/tech.j2ee/guidances/concepts/java_messaging_service_jms_84F49452.html>. Acesso em: 08 de setembro de 2015.

Introdução à JMS com Java EE 6 - Java Magazine 80. Disponível em: < http://www.devmedia.com.br/introducao-a-jms-com-java-ee-6-java-magazine-80/17105>. Acesso em 23 de setembro de 2015.

Java Message Service (JMS). Disponível em: <http://www.nce.ufrj.br/conceito/artigos/2005/012p1-2.htm>. Acesso em : 07 de setembro de 2015.

Java Message Message Service – JMS e Wireless Wireless CORBA Specification – WCORBA. Disponível em: <http://www-di.inf.puc-rio.br/~endler/courses/Mobile/transp/jms-wcorba.pdf>. Acesso em 23 de Setembro.

JMS API: comunicação Assíncrona – Parte 1. Disponível em: <http://www.devmedia.com.br/jms-api-comunicacao-assincrona-parte-1/30073>. Acesso em: 05 de setembro de 2015.

O protocolo FTP (File Transfer Protocol). Kioskea.net. Disponível em: <http://br.ccm.net/contents/265-o-protocolo-ftp-file-transfer-protocol>. Acesso em: 07 de setembro de 2015.