UNIVERSIDADE FEDERAL DE SÃO CARLOS

CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

ARQUITETURA ESCALÁVEL DE ALTO DESEMPENHO PARA ATUALIZAÇÃO, ACESSO E RECUPERAÇÃO DE INFORMAÇÕES EM BANCOS DE DADOS DE APLICAÇÕES EMBARCADAS

DANIEL MEZZALIRA

ORIENTADOR: PROF. DR. LUÍS CARLOS TREVELIN

São Carlos - SP
Agosto/2012

UNIVERSIDADE FEDERAL DE SÃO CARLOS

CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

ARQUITETURA ESCALÁVEL DE ALTO DESEMPENHO PARA ATUALIZAÇÃO, ACESSO E RECUPERAÇÃO DE INFORMAÇÕES EM BANCOS DE DADOS DE APLICAÇÕES EMBARCADAS

DANIEL MEZZALIRA

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de São Carlos, como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação, área de concentração: Sistemas Distribuídos e Redes
Orientador: Prof. Dr. Luis Carlos Trevelin

São Carlos - SP
Agosto/2012

Ficha catalográfica elaborada pelo DePT da Biblioteca Comunitária da UFSCar

M617ae

Mezzalira, Daniel.
Arquitetura escalável de alto desempenho para atualização, acesso e recuperação de informações em bancos de dados de aplicações embarcadas / Daniel Mezzalira. -- São Carlos : UFSCar, 2012.
110 f.

Dissertação (Mestrado) -- Universidade Federal de São Carlos, 2012.

1. Sistemas distribuídos. 2. Arquiteturas distribuídas. 3. Teoria das filas. 4. Escalabilidade. 5. Alto desempenho. I. Título.

CDD: 005.43 (20ª)

Universidade Federal de São Carlos

Centro de Ciências Exatas e de Tecnologia

Programa de Pós-Graduação em Ciência da Computação

"Arquitetura Escalável de Alto Desempenho para Atualização, Acesso e Recuperação de Informações em Bancos de Dados de Aplicações Embarcadas"

Daniel Mezzalira

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de São Carlos, como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação

Membros da Banca:

Prof. Dr. Luis Carlos Trevelin
(Orientador - DC/UFSCar)

Prof. Dr. Cesar Augusto Cavalheiro Marcondes
(DC/UFSCar)

Prof. Dr. Edmundo Roberto Mauro Madeira
(UNICAMP)

São Carlos
Agosto/2012

À minha família, sempre presente na minha vida, me amando incondicionalmente. Ao meu pai Francisco pelo exemplo de bondade, superação e competência. À minha mãe Márcia, por seu amor, força de espírito e devoção. Ao meu irmão Samuel, prova de dedicação, genialidade e seriedade sem perder o espírito brincalhão. Ao meu avô João, homem moldado pela religião, honestidade, integridade e trabalho árduo. À minha avó Ornélia, pelo exemplo de elegância e sutileza. Ao meu avó Éder, que infelizmente nos deixou antes de eu nascer, mas tenho certeza que me ajudou de onde estava. À minha avó Cida, pelo exemplo de alegria mediante as dificuldades e carinho pelos netos.

À minha namorada Camila, minha companheira! Sempre ao meu lado, me apoiando e me ajudando em todas as dificuldades, zelando pela minha saúde, bem estar e acreditando no meu sucesso. Te amo!

AGRADECIMENTO

Agradeço ao meu orientador Prof. Dr. Luís Carlos Trevelin pela oportunidade, pelo tempo que dedicou a me guiar. Obrigado por compartilhar sua experiência, pela paciência e amizade.

Aos professores do departamento de computação da Universidade Federal de São Carlos, em especial aos Profs. Drs. Antônio Francisco Prado, Célio Estevan Moron, César Augusto Cavalheiro Marcondes, Emerson Carlos Pedrino, Estevam Rafael Hruschka Junior, Hélio Crestana Guardia, Jander Moreira, Lúcia Helena Machado Rino, Luís Carlos Trevelin, Maurício Figueiredo, Marilde Terezinha Prado dos Santos, Roberto Ferrari, Sandra Abib, Sandra Camargo Pinto Ferraz Fabbri, Sérgio Donizetti Zorzo pelos ensinamentos dentro e fora das salas de aula e constante apoio para solucionar minhas dúvidas. Obrigado por irem além das obrigações de um professor e realmente contribuírem para minha formação.

A paciência e ajuda da Maria Cristina Carreira Trevelin, que sempre me ajudou a tirar as dúvidas dos procedimentos burocráticos da pós-graduação.

Ao Eduardo Martins Lopes, pela contribuição nos testes de desempenho preliminares nos computadores do Laboratório de Química Teórica da UFSCar.

Ao Cleber R. Manzoni e Ivan R. Bizari, da empresa Enalta Inovações Tecnológicas e Adriano R. de Magalhães da Onixsat, por contribuições fundamentais para o sucesso do projeto.

Aos amigos de laboratório, que sempre ajudam uns aos outros.

A Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) pelo auxílio financeiro durante esta pesquisa.

"É melhor tentar e falhar,

que preocupar-se e ver a vida passar;

é melhor tentar, ainda que em vão,

que sentar-se fazendo nada até o final.

Eu prefiro na chuva caminhar,

que em dias tristes em casa me esconder.

Prefiro ser feliz, embora louco,

que em conformidade viver ..."

Martin Luther King

RESUMO

O gerenciamento remoto de múltiplos sistemas tais como máquinas operatrizes, veículos, aviões, dentre outros, demanda um fluxo bastante intenso de dados entre eles e o sistema gerenciador. Pesquisas têm sido desenvolvidas na concepção e implementação de arquiteturas escaláveis que atendam essas demandas levando a questões interessantes de desempenho. O objetivo deste trabalho é propor uma arquitetura escalável de baixo custo para aplicações embarcadas, utilizando *pools* de computadores pessoais para obter alto desempenho no armazenamento, recuperação e tratamento da informação. É motivado pela grande demanda de rastreamento e monitoramento de máquinas e veículos, contemplando conceitos de redes móveis com tecnologia de satélites e GPRS, juntamente com o requisito de confiabilidade e desempenho no envio da informação. Propõe a definição de uma estrutura de servidor, cuja distribuição é transparente para a aplicação, à qual compete o recebimento das mensagens dos equipamentos embarcados através de tecnologia de radio frequência, decodificação e inserção das informações num banco de dados e posterior recuperação destas informações. Através da simulação de diferentes estratégias modeladas, utilizando a teoria das filas, para determinação da arquitetura e a utilização de métodos matemáticos preditivos para estimação da carga futura para a aplicação servidora, foi possível obter uma solução que atendeu satisfatoriamente às premissas da pesquisa. Dessa forma, conclui-se que é possível estimar tendências de picos de processamento de informação para aplicações de telemetria de frotas.

Palavras-chave: Arquitetura de software distribuída, alto desempenho, escalabilidade, sistema de filas, simulação, sistemas embarcados, telemetria, redes DTN, tolerância a falhas.

ABSTRACT

Managing multiple systems such as machine tools, vehicles, aircraft, among others, demand a very intense flow of data between them and the system manager. Researches have been developed in the design and implementation of scalable architectures that meet these demands leading to interesting questions of performance. The objective of this work is to propose a low cost scalable architecture for embedded applications, using pools of personal computers for high performance storage, retrieval and processing of information. It is driven by strong demand for tracking and monitoring of machines and vehicles, covering concepts of mobile networks with satellites and GPRS technology together with the requirement of reliability and performance in the sending of information. Proposes the definition of a server structure, whose distribution is transparent to the application, which is responsible for the receipt of messages from embedded devices via radio frequency technology, decoding and integration of information in the database and subsequent recovery of these information's. Through simulation of different modeled strategies using queuing theory to determine the architecture and the use of predictive mathematical methods for estimating the future burden for the server application, it was possible to obtain a solution that satisfactorily met the assumptions of the research. Thus, it is concluded that it is possible to estimate trends peaks processing information for telemetry applications fleet.

Keywords: Software Distributed Architecture, high performance, scalability, queuing systems, simulation, embedded systems, telemetry, DTN, fault tolerance.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

2.5G - *Second Generation (com recursos adicionais)*

3G - *Third Generation*

ACK - *Acknowledgement*

BGAN - *Broadband Global Area Network*

BI - Business intelligence

CPU - *Central Processing Unit*

D-AMPS - *Digital-Advanced Mobile Phone Service*

DTN - *Delay-Tolerant Network*

ERP - *Enterprise Resource Planning*

FTP- *File Transfer Protocol*

GEO - *Geostationary Earth Orbit*

GPRS - *General packet radio service*

GSM - *Global System for Mobile Communications (originalmente, Group Special Mobile)*

HTTP- *Hypertext Transfer Protocol*

IDL- *Interface Definition Language*

IEEE - *Institute of Electrical and Electronics Engineers*

JMT- *Java Modelling Tools*

LEO - *Low-Earth Orbit*

MAPL- *Maximum Allowed Path Loss*

MEO - *Medium-Earth Orbit*

NCK - *Non-Acknowledgement*

ORB- *Object Request Broker*

OMG- *Object Management Group*

P2P - *Peer to Peer*

POA- *Portable Object Adaptor*

RPC- *Remote Procedure Calls*

SOAP- *Simple Object Access Protocol*

TCP - *Transmission Control Protocol*

UDP - *User Datagram Protocol*

UHF - *Ultra High Frequency*

URL- *Uniform Resource Locator*

VHF - *Very high frequency*

XML- *Extensible Markup Language*

WI-FI - *Wireless Fidelity*

WIMAX - *Worldwide Interoperability for Microwave Access*

ZIGBEE - *O nome da marca é originado com referência ao "comportamento das abelhas após seu retorno à colmeia."*

SUMÁRIO

Capítulo 1

INTRODUÇÃO

Devido ao destaque brasileiro quanto ao tamanho da frota de veículos e à necessidade de otimização e redução de custos, o monitoramento de ativos motomecanizados ganha destaque para atingir estas metas. Para suprir esta demanda de processamento, poderosos e caros servidores são alocados para suportar os picos de processamento que, muitas vezes, ocorrem em períodos curtos. O projeto apresenta a proposta de um servidor distribuído que, através da medição da capacidade e da sua utilização, realiza a gestão de alocação e desalocação dos nós, com o intuito de apresentar uma arquitetura distribuída, de baixo custo e alta escalabilidade para suprir esta demanda.

1.1 Contexto

O Brasil destaca-se mundialmente pela frota de veículos, tanto leves quantos pesados e pela constante automação e mecanização de diversos setores econômicos. O processo de otimização de recursos destes setores criou uma demanda de sensoriamento e telemetria destes veículos, visando obter informações valiosas para tomadas de decisões e principalmente para redução de gastos, como combustível, custos esses que representam grande parcela da operação (ARAÚJO, 2002).

Esta necessidade gera um grande volume de dados, dentre os quais alguns devem ser registrados e visualizados prontamente. O contexto brasileiro impõe uma barreira infraestrutural de comunicação, na qual sombras de sinal forçam os equipamentos a armazenar informações e descarregá-las abruptamente quando retornam às áreas com conectividade. Para suprir estes picos, poderosos e caros servidores devem ser alocados, para muitas vezes serem exigidos em poucos

períodos bem definidos do mês como, por exemplo, nos dias que antecedem a geração de relatórios de balanço mensais.

Este projeto propôs um estudo baseado na teoria de filas, analisando dados reais fornecidos por empresas do ramo de telemetria veicular, com o intuito de modelar o problema, definir melhor o cenário e, a partir de análises da capacidade do sistema, definir uma arquitetura de um servidor que atenda a demanda de processamento, com alto grau de escalabilidade, baixo custo, utilizando computadores comuns de forma a alocar nós de processamento de acordo com análises de capacidade e usabilidade do sistema.

1.2 Motivação e Objetivos

Investimentos e inovações tecnológicas para qualquer setor são motivados de acordo com a rentabilidade dos mesmos. Alguns desses setores demandam automação de processos, utilizando recursos de motomecanização e utilização de frotas veiculares de diferentes portes, com o objetivo de otimização da produção e redução de custos, agregando competitividade e lucratividade ao proprietário (FAVARETTO, 2001).

Tomando o exemplo do setor sucroalcooleiro, o cenário da operação impõe grandes desafios para a informatização dos equipamentos, como ambiente com grande concentração de partículas suspensas (poeira), umidade, impacto, altas temperaturas e a falta de infraestrutura de comunicação, que será explorada neste trabalho. Os embarcados deste setor comumente transmitem informações de telemetria, rendimentos de máquinas e informações operacionais, com o intuito de rastrear e fornecer informações suficientes para melhor compreensão do que ocorre em campo, a fim de otimizar e reduzir custos da operação.

O envio da informação muitas vezes é custoso, exigindo técnicas de tolerância a falhas, como as redes tolerantes a atrasos (Redes DTN) (WARTHMAN, 2003), que implicam armazenamento da informação e a transmissão de todo o conteúdo quando há conectividade. Esta particularidade gera grandes picos de processamento ao servidor, que muitas vezes deve ser superdimensionado para suprir esta demanda, que ocorre de forma espaçada no tempo.

O objetivo da pesquisa foi propor o estudo e modelagem deste sistema de monitoramento de frotas através da teoria das filas, com o intuito de encontrar um modelo que possua o melhor comportamento médio a estes eventos, utilizando *traces* reais e simulações para representar esta interação entre os embarcados e o servidor. Este sistema de filas deve contemplar o recebimento, processamento e armazenamento da informação, além da interação do usuário às informações como, por exemplo, utilizando uma aplicação de consultas de frota.

1.3 Metodologia

Uma vez que o sistema alvo desta pesquisa possui alto grau de complexidade, envolvendo grande número de nós, computadores embarcados, clientes e servidores, além de toda a topologia de rede dos provedores dos meios de comunicação (empresas de telefonia celular para a utilização de tecnologia GPRS, empresas de comunicação via satélite ou até mesmo as empresas provedoras de internet banda larga que os servidores estarão conectados), torna-se interessante explorar mais de uma abordagem para resolução do problema proposto.

O estudo foi iniciado com o mapeamento do fluxo da informação no cenário agrícola e as particularidades da mesma, como prioridades, tipo e formas de envio das mensagens, ou seja, a topologia do sistema. Após a compreensão do comportamento, arquiteturas de servidores em potencial serão propostas e validadas através de simulações, para garantir que atendam às expectativas.

O trabalho foi desenvolvido através da modelagem e estudo analítico do problema utilizando a teria das filas, visando validar a estratégia utilizada na proposta da arquitetura. Estas hipóteses de gestão e processamento da informação foram colocadas à prova através de simulações, com o objetivo de encontrar a melhor solução que posteriormente foi implementada e testada em ambiente real.

Para esta modelagem, as seguintes etapas foram concluídas:

1. Estudo bibliográfico sobre teoria de filas, sistemas distribuídos de alto desempenho, redes DTN, arquitetura de software distribuída, simulação e análise de tráfego;

2. Obtenção e análise de dados de servidores de empresas que atuam no ramo de telemetria de frotas e ativos motomecanizados. Este item contemplou servidores de grande e pequeno volume de informações;
3. Processo interativo de simulações para compreender o comportamento de diferentes estratégias, previamente definidas através de modelagem dirigida sobre a teoria das filas;
4. Implementação da arquitetura da aplicação servidora que obteve melhor comportamento médio sobre os *traces* adquiridos na etapa 2;
5. Testes para obtenção de resultados da arquitetura da aplicação servidora implementada na etapa 4.

1.4 Organização do Trabalho

O capítulo introdutório apresenta a necessidade de monitoramento de frotas e de ativos motomecanizados para a realização da gestão do setor que utiliza estes recursos. Esta necessidade gera um grande volume de dados que, dependendo de suas características, devem ser disponibilizados prontamente para o usuário. Para isso, poderosos e caros servidores são utilizados para gestão de grandes frotas, sendo exigidos muitas vezes para processamento de picos espaçados no tempo.

O capítulo dois descreve as implicações de um monitoramento remoto. Cita os usos e peculiaridades de cada tecnologia de comunicação sem fio e demonstra as dificuldades de garantir uma área de cobertura total para uma determinada operação. Estas falhas de sinais acarretam no armazenamento da informação pelos sistemas embarcados e envio abrupto quando encontra uma área com sinal, gerando os picos de processamento no servidor.

Para proporcionar e nortear a concepção do projeto, o capítulo três apresenta fundamentos básicos para o estudo de uma arquitetura distribuída de uma aplicação servidora de monitoramento de frota.

Através de parcerias com empresas do ramo de telemetria de frotas, *traces* reais foram estudados no capítulo quatro, que identificou a necessidade de um levantamento de infraestrutura de comunicação para o cenário rural brasileiro, resultando em uma base de dados espacial com a área de cobertura de sinal.

Buscando conceber uma arquitetura para a aplicação servidora, o capítulo cinco traz uma progressão de simulações utilizando e quantificando o desempenho de diferentes modelagens de sistemas de filas.

Assim que a arquitetura foi definida no capítulo anterior, o capítulo seis mostra o processo de implementação desta arquitetura em Java, descrevendo os módulos e definições do projeto.

Finalmente, os resultados são apresentados no capítulo sete, junto com os trabalhos futuros que este estudo fomentou.

Capítulo 2

ASPECTOS TECNOLÓGICOS DA COMUNICAÇÃO NO CAMPO

O monitoramento remoto de sistemas embarcados em campo apresenta diversos desafios tecnológicos para garantir uma cobertura de sinal aceitável. Cada tecnologia disponível apresenta prós e contras, sugerindo a utilização conjunta de mais de uma tecnologia. Estas falhas de comunicação, abordada pelas redes DTN, são evidenciadas e demonstram os picos de processamento que ocorrem no servidor.

2.1 Operação no Campo

Conforme apresentado no capítulo anterior, o agronegócio sob o contexto brasileiro impõe uma série de dificuldades multidisciplinares, entre elas a necessidade de comunicação em ambientes rurais, que são carentes de infraestrutura de comunicação. Este capítulo descreve detalhadamente o problema de monitoramento de frotas e componentes motomecanizados, assim como as propostas de soluções que foram modeladas e estudadas neste trabalho.

É importante definirmos que uma operação no campo não implica necessariamente ambiente agrícola. Uma aplicação em campo pode ser de qualquer aspecto, como agrícola, florestal, marítima, inclusive aeroespacial. Entenderemos neste trabalho que uma operação no campo significa que o sistema está operando em condições reais.

A justificativa de realizar a telemetria dos ativos motomecanizados é fornecer informações suficientes para conceber o conhecimento do processo de negócio e o

cotidiano de sua operação, alimentando constantemente os Sistemas Integrados de Gestão Empresarial (conhecidos por ERPs, ou Enterprise Resource Planning), permitindo tomadas corretas de decisões e planejamentos futuros. Esta necessidade de monitoramento de ativos motomecanizados evidencia a heterogeneidade dos equipamentos, que necessariamente geram tipos de mensagens distintas com prioridades diferentes.

Os tópicos a seguir explicam os resultados desejados com esta coleta de informação, assim como os diferentes tipos de dados e formas de envio. Para melhor compreensão, seguem três tópicos que diferem com relação aos objetivos dos dados coletados, assim como os tipos de mensagens enviadas pelo sistema embarcado em aplicações destinadas ao monitoramento de frotas.

2.1.1 Motomecanização

Motomecanização é o termo que concebe qualquer operação dotada de máquinas motrizes e meios mecânicos. Motomecanizar também define a mecanização de uma atividade, como a substituição do corte manual da cana-de-açúcar pela utilização de colhedoras (adoção da colheita mecanizada). Novamente, é importante fundamentar que a motomecanização não se restringe à operação agrícola, e sim é aplicável a qualquer setor que utilize métodos mecânicos e motorizados.

Exemplificando, a motomecanização de uma usina é também responsável pela colheita e transporte do insumo, além da manutenção dos ativos mecanizados. A gestão deste setor tem por prerrogativa a otimização de recursos desta operação, como consumo de combustível, adequação de frota, logística de apoio, manutenção e outros, com o intuito de otimizar a operação e atingir ganhos de produtividade (ARAÚJO, 2002).

Para esta função, a telemetria dos equipamentos em campo fornece informações georeferenciadas com o objetivo de monitorar a:

- Condição mecânica: informa dados referentes à condição da máquina, como temperatura do motor, pressão do óleo, RPM, etc.
- Produtividade: como todo maquinário desempenha uma função, dados como consumo de combustível e informações específicas, como

tonelada de cana colhida, número de árvores colhidas, valor da carga transportada, entre outros, são monitorados pelo sistema.

Quanto à condição mecânica, o constante monitoramento dos sensores fornece informações a central de monitoramento para garantir o funcionamento correto dos ativos, evitando desgaste e acidentes. Devido à tendência de incorporação de recursos computacionais em diversos setores econômicos, uma vasta diversidade de máquinas está incorporando computadores de bordo, máquinas estas de diferentes portes, inclusive grandes linhas de produção e máquinas pesadas.

Os ramos siderúrgico, florestal e de construção civil são bons exemplos de necessidade de telemetria dos equipamentos. A exigência de máquinas cada vez maiores eleva exponencialmente o risco de acidentes graves (figuras 2.1 e 2.2), pois passamos a lidar com máquinas que transportam cargas de até 880 toneladas, no caso da Kirow Multi Mover 880 C (KIROW, 2011) (figura 2.3), ou até 80 toneladas de escória de alto forno a mil e quinhentos graus centígrados, no caso da Kirow Slag Taurus, utilizada no setor siderúrgico (KIROW, 2011) (figura 2.4), segundo as especificações fornecidas pelo fabricante.

Figura 2.1 – Acidente com gerador de 64MW no Irã (TANDLER, 2011).

Figura 2.2 – Explosão de uma turbina na usina de Sayano Shushenskaya (MAESTRI, 2011).

Figura 2.3 – Kirow Multi Mover 880 C (KIROW, 2011).

Figura 2.4 – Kirow Slag Taurus (KIROW, 2011).

2.1.2 Logística

Corresponde aos benefícios da obtenção e do processamento das informações oriundas do monitoramento de equipamentos em campo. A partir dos dados obtidos é possível extrapolar o comportamento dos ativos motomecanizados, possibilitando uma correta previsão e contenção de eventos. Possibilita também ao empreendimento adotar recursos da chamada “Inteligência Empresarial”, conhecida por BI (Business Intelligence, termo concebido pelo Gartner Group), que aprofunda o conhecimento do negócio e fornece mais recursos para tomadas de decisões (ROSSETTI & MORALES, 2007).

A partir do momento que a latência da transmissão da informação tende à de sistemas em tempo real, um novo tipo de abordagem torna-se viável, como a automação e adequação da linha de produção do negócio. A velocidade da moenda de uma usina de cana-de-açúcar pode ajustar-se automaticamente de acordo com o fluxo de cana que está previsto chegar, alocar sem intervenção humana a frota transportadora de uma empresa, de acordo com a previsão de chegada dos veículos, entre outras possibilidades.

É, portanto, uma funcionalidade do sistema que agrega muito valor à solução, otimiza recursos e evita não conformidades, como, por exemplo, a que se mostra na figura 2.5. Não conformidades geralmente acarretam em retrabalhos ou contenções, gerando maior custo para operação, acarretando em prejuízo e/ou perda de competitividade.

Figura 2.5 - Exemplo de problema de logística: Soja estocada a céu aberto devido à superlotação dos silos (SORRISO, 2010).

2.1.3 Tipos de Mensagens

Para habilitar a aplicabilidade da motomecanização e de análise logística, possibilitando a utilização de recursos de BI, tipos de mensagens distintos são exigidos. Entende-se por mensagem todo pacote de dado enviado pelo sistema embarcado, uma vez que as tecnologias de comunicação disponíveis no mercado trabalham com um número limitado de bytes por envio (as tecnologias de envio serão discutidas na seção 2.2).

É importante haver pelo menos dois tipos de mensagens no sistema: um referente ao acompanhamento instantâneo do ativo motomecanizado, e outro detalhado, para obtenção de relatórios completos e concisos. Geralmente são escritos protocolos de comunicação proprietários, onde cada fornecedor de tecnologia de telemetria encapsula a informação compactada utilizando uma estratégia, a fim de obter o maior número de informações com o mínimo de banda, que pode variar muito de acordo com a tecnologia de comunicação utilizada (o envio por satélite dispende mais recursos financeiros do que uma GPRS, por exemplo).

Além disso, sistemas distintos ou módulos do sistema ativos priorizam informações diferentes. Para cada aplicação, uma determinada informação pode ser mais valiosa do que para outra como, por exemplo, o monitoramento de uma carga de valores por via marítima e o acompanhamento de uma caldeira. Para o primeiro, a telemetria do sensor do lacre da carga deve ser georeferenciada, com envio na ordem de dezenas de minutos; por outro lado, no segundo caso o georeferenciamento é secundário, focando todos os recursos computacionais para os sensores de temperatura e pressão.

2.2 Tecnologias de Comunicação

Os diferentes cenários de comunicação no monitoramento de ativos motomecanizados inserem desafios para obtenção da informação com a menor latência possível, com qualidade e consistência. A falta de infraestrutura em áreas rurais é comum, havendo constantemente sombras e bordas (fronteiras de

qualidade, havendo pouco sinal para a transferência, ocasionando interrupções abruptas) de conectividade.

As distâncias entre equipamentos e a estrutura física da rede implicam criação de uma difícil e onerosa logística para coleta, atualização e manutenção das soluções, tornando as tecnologias de comunicação sem fio atraentes.

Segue nas seções seguintes uma breve descrição e contextualização das tecnologias de comunicação sem fio mais utilizadas pelo mercado. O objetivo é detalhar as tecnologias, seus problemas e exemplificar a utilização de cada uma nos diferentes setores em que a proposta se insere.

2.2.1 Satélite

A utilização de comunicação via satélite garante uma ampla cobertura global, podendo ser de três tipos: *Geostationary Earth Orbit* (GEO), *Medium-Earth Orbit* (MEO) e *Low-Earth Orbit* (LEO), os quais diferem, basicamente, na altitude de órbita que a tecnologia utiliza, como demonstra a figura 2.6:

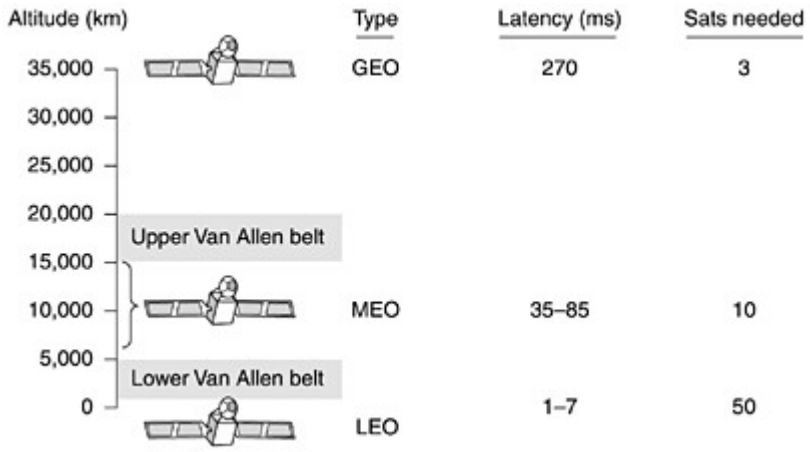

Figura 2.6 – Tipos de Órbitas utilizadas pelos satélites de telecomunicação (TANENBAUM, 2011).

Os três tipos possuem especificações tecnológicas distintas, tais como o número de satélites necessários para cobertura global de sinal e a latência da informação. Estas peculiaridades acarretam em anuências dos requisitos da solução, que devem seguir as características tecnológicas que cada tipo de órbita impõe, como latência, protocolo da camada de transporte (UDP ou TCP) e

imposições dos poucos prestadores destes serviços disponíveis no mercado, como a Iridium (LEO) e InmarSat (GEO).

Cada solução possui vantagens e desvantagens, porém o fator preponderante no uso de cada uma é o custo. Soluções UDP da Iridium (IRIDIUM, 2012) possuem baixo custo de instalação (modem + antena), mas possuem pouco volume de dados (p.e. plano mensal de 10 Kbits mais um valor para cada mensagem excedente); em contrapartida, o uso das soluções TCP da InmarSat (INMARSAT, 2012) requerem elevados valores de hardware e fornecem preços atraentes para grandes pacotes de dados (na ordem de $100,00 por 25 Mbits, utilizando a rede BGAN).

Outra dificuldade imposta pelo contexto brasileiro de comunicação diz respeito às antenas utilizadas pela solução escolhida. Por utilizar baixas órbitas, as antenas utilizadas para comunicação com a rede Iridium não precisam ser potentes. Apesar de serem omnidirecionais, a maioria é sensível à inclinação, conforme a mostra a figura 2.7, que exemplifica as características da antena Satcom S67-1575-168:

SPECIFICATIONS	FOR REFERENCE ONLY
MODEL	S67-1575-168
ELECTRICAL	
Frequency	J1 :L1 1565–1585 MHz, L2 1217–1237 MHz J2: 1610–1626 MHz
VSWR	≤ 2.0:1
Polarization	RHCP
Impedance	50 Ohms Nominal
Power Handling	J1: 1 Watt J2: 60 Watts
Gain	-1.0 dBic 0° ≤ θ ≤ 75° -2.5 dBic 75° < θ ≤ 80° -4.5 dBic 80° < θ ≤ 85° -7.5 dBic θ = 90° @ Horizon
Gain (Preamplifier)	26.5 ±3 dB
Voltage	3.3 TO +5 VDC @ 65 mA max
Lightning Protection	DC Grounded
MECHANICAL	
Weight	16 oz.
Height	.92 in.
Length	7.86 in.
Width	3.00 in.
Material	6061-T6 aluminum
Finish	Skydrol-resistant enamel
Connectors	Type "TNC" Female (2)
ENVIRONMENTAL	
Temperature	-54°C (-65°F) TO +110°C (+230°F)
Vibration	10 Gs
Altitude	-1500 to 70,000 ft

Figura 2.7 – Características de uma antena utilizada pela rede Iridium (SATCOM, 2011).

Levando em consideração a posição relativa do equipamento com o satélite, somada à inclinação predominante do estado de São Paulo, por exemplo, é comum trabalharmos com o ganho da antena reduzido. Este fator adicionado às condições climáticas, como nuvens *cumulus nimbus* entre o equipamento e o satélite, implica

redução de capacidade de transmissão do modem Iridium, o qual recomenda transmitir a informação apenas quando a qualidade do sinal for maior ou igual a três, numa escala de zero a cinco. Portanto, mesmo utilizando esta tecnologia, não é possível garantir efetivamente conectividade constante.

As antenas utilizadas para a comunicação com a rede BGAN (GEO) possuem complexos mecanismos de posicionamento e direcionamento, para garantir correta visada ao satélite. São extremamente caras em comparação aos equipamentos de baixa órbita e, por possuírem muitas partes móveis, não são adequadas à operação móvel suscetível a impacto, como a operação agrícola.

2.2.2 GPRS

Tecnologia de comunicação disponibilizada pelas companhias de telecomunicação celular, amplamente utilizada por frotas de veículos em áreas urbanas. Possui custo atrativo, tanto para o hardware quanto para os pacotes de serviço, possibilitando o manejo de arquivos pelos sistemas embarcados, como atualização do próprio computador de bordo remotamente, envio de fotos, dentre outros.

Definida como tecnologia 2.5G (uma medida paliativa até a chegada da tecnologia 3G), permite o envio de pacotes IPs através de células que executam o sistema de voz, fazendo uma superposição aos sistemas D-AMPS ou GSM (TANENBAUM, 2011). Tecnologias superiores não serão descritas devido à baixa disponibilidade em áreas rurais.

Infelizmente, também não garante conectividade em tempo integral, pois está sujeita a áreas de sombra de duas naturezas: ocasionadas por obstáculos (naturais, como montanhas, ou artificiais, como estruturas metálicas) e devido a concessões das antenas realizadas pelas operadoras de celular (um caminhão saindo de Porto Alegre, Rio Grande do Sul, com destino a Porto Velho, Rondônia, equipado com um modem GPRS vinculado a operadora A, pode não conseguir transmitir as informações por falta de prestação de serviços desta operadora nos estados do centro-oeste. Para atenuar este tipo de problema, são comuns modens com suporte a mais de um SIM-Card).

2.2.3 Radiofrequência

Apesar das tecnologias supracitadas também utilizarem-se de radiofrequência, são vinculadas a operadoras de serviços, limitando a variedade de recursos. Este capítulo aborda as tecnologias sem fio que possibilitam maior liberdade ao provedor da solução, que é capaz de implementar livremente aplicações utilizando estas tecnologias, como a Motorola, que desenvolveu radio-modens a fim de prover uma solução de grande alcance de transmissão de dados.

2.2.3.1 Wi-Fi e WiMax

Tecnologias padronizadas pelo instituto IEEE como 802.11 (Part 11, 2011) e 802.16 (Part 16, 2007) respectivamente. Apesar de possuírem tecnologias e protocolos distintos, têm objetivos de utilização semelhantes, diferenciando-se apenas no alcance da área de cobertura.

Garantindo cobertura de sinal sobre a área de atuação dos equipamentos embarcados, é estabelecida conectividade suficiente para monitoramento em tempo real com ótima largura de banda, sem custo de utilização.

O padrão 802.11 é comumente utilizado em áreas restritas, tais como galpões, parques industriais, instalação em guaritas (para que assim que o sistema embarcado passe por ela, descarregue suas informações), dentre outros. Atualmente, o padrão 802.16 é pouco difundido no Brasil, devido à demora de homologação por parte da Agência Nacional de telecomunicações (ANATEL), que determinou a utilização apenas da faixa de 2.5GHz (ANATEL, 2009). Porém, essa tecnologia possibilita a criação de amplas áreas de cobertura, garantindo conectividade para utilizações no agronegócio e em áreas metropolitanas.

2.2.3.2 ZigBee e Bluetooth

Tecnologias padronizadas pelo instituto IEEE 802.15.4 (Part 15.4, 2009) e 802.15.1 (Part 15.1, 2005) respectivamente. Disponibilizam recursos para implantação de redes pessoais de curto alcance, fornecendo taxas razoáveis de transmissão.

Possibilitam criar zonas de interação, tais como pontos de controle (que disparam eventos quando o equipamento se aproxima e estabelece comunicação),

de embarcado para embarcado, dentre outros. Amplamente utilizados para aplicações de natureza interventiva, como reconfiguração do embarcado via *tablet*, transferência de informações, entre outras.

2.2.3.3 Radio-Modem

Devido às características físicas das ondas de radiofrequência que as tecnologias discutidas neste capítulo utilizam, quanto mais alta a frequência, mais suscetíveis as ondas estão de refletir em obstáculos ou interagir com moléculas de água. Aplicações que atuam em ambientes com muitos obstáculos, tais como as do ramo florestal, onde os equipamentos estão sob a copa de árvores, não atingem áreas de conectividade satisfatórias.

Para viabilizar estas aplicações, modens com frequência VHF ou UHF (MOTOROLA, 2012) criam ondas com tamanhos que transpõem obstáculos com menor perda energética, implementadas nos chamados radio-modens, devido ao fato de estarem sob a infraestrutura dos rádios de comunicação. Essas frequências criam grandes áreas de conexão, através da construção de antenas repetidoras, muitas vezes já instaladas para comunicação entre colaboradores.

2.2.4 Armazenamento Massivo

Outra possibilidade para telemetria e monitoramento de sistemas motomecanizados; consiste na simples coleta manual das informações armazenadas pelos sistemas embarcados.

Por motivos infraestruturais ou econômicos (o hardware é barato e não insere custos mensais), pode ser criada uma operação de coleta da informação periódica, de acordo com a necessidade (diária, mensal, por exemplo).

Este é um sistema que não contribui para tomadas rápidas de decisões, contudo permite maior resolução da informação, pré-processamento da mesma pelo sistema embarcado, diminuindo os custos computacionais do servidor, além de baratear a solução.

2.3 *Delay-Tolerant Networks* - Redes DTN

Como exemplificado no capítulo anterior, toda tecnologia de transferência de informação está sujeita a períodos sem comunicação, criando a necessidade de uma nova abordagem para mitigar os problemas ocasionados por esta instabilidade da rede. Vários estudos e propostas são disponibilizados pela comunidade, abordando técnicas de roteamento, confirmação de entrega e tolerância a falhas, redes estas chamadas de redes DTN (WARTHMAN, 2003). Esta rede possui o desafio de comutar pacotes sobre redes distintas, considerando altas taxas de falhas e perda de conectividade (ABRAMOV, 2007), com o objetivo de entregar o pacote de informações para o destino com sucesso e desempenho.

A arquitetura de uma rede DTN define o conceito de regiões, que determinam a área de atuação de redes com comunicação semelhantes. É implementada como uma camada sobreposta a estas redes regionais, que podem possuir características distintas entre si. Adicionando uma nova camada à pilha de protocolo, a de empacotamento, torna possível o envio de mensagens através de redes heterogêneas. Esta nova camada fica entre as camadas de aplicação e transporte, permitindo que a aplicação utilize a informação proveniente das redes heterogêneas pertencentes à rede DTN. A figura 2.8 demostra a camada de empacotamento e a figura 2.9 mostra um exemplo de transmissão de pacotes sobre a internet comum e a possibilidade deste mesmo transporte sobre uma rede DTN entre regiões com diferentes características de camadas do protocolo:

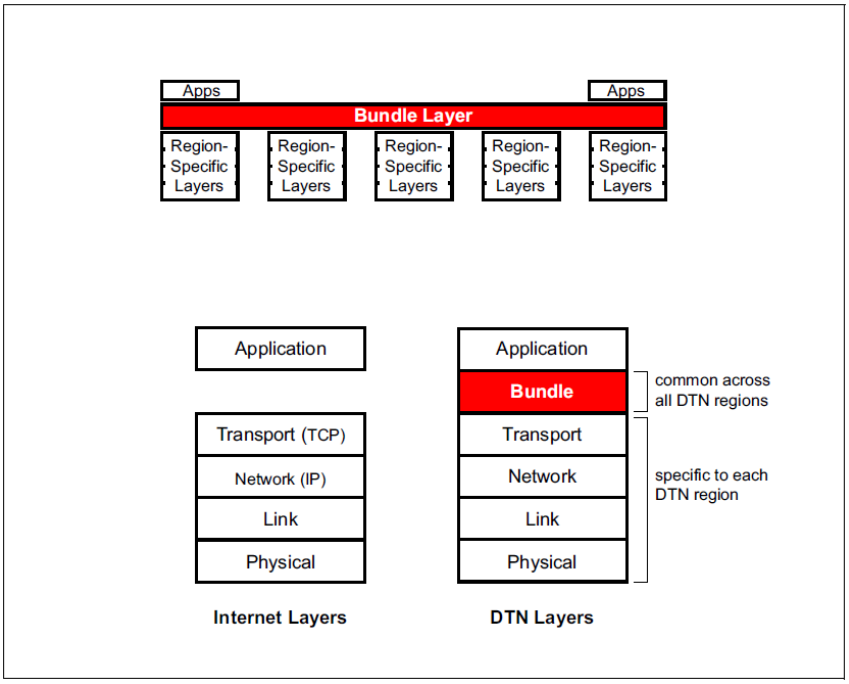

Figura 2.8 – Camada de empacotamento presente em redes DTN (WARTHMAN, 2003).

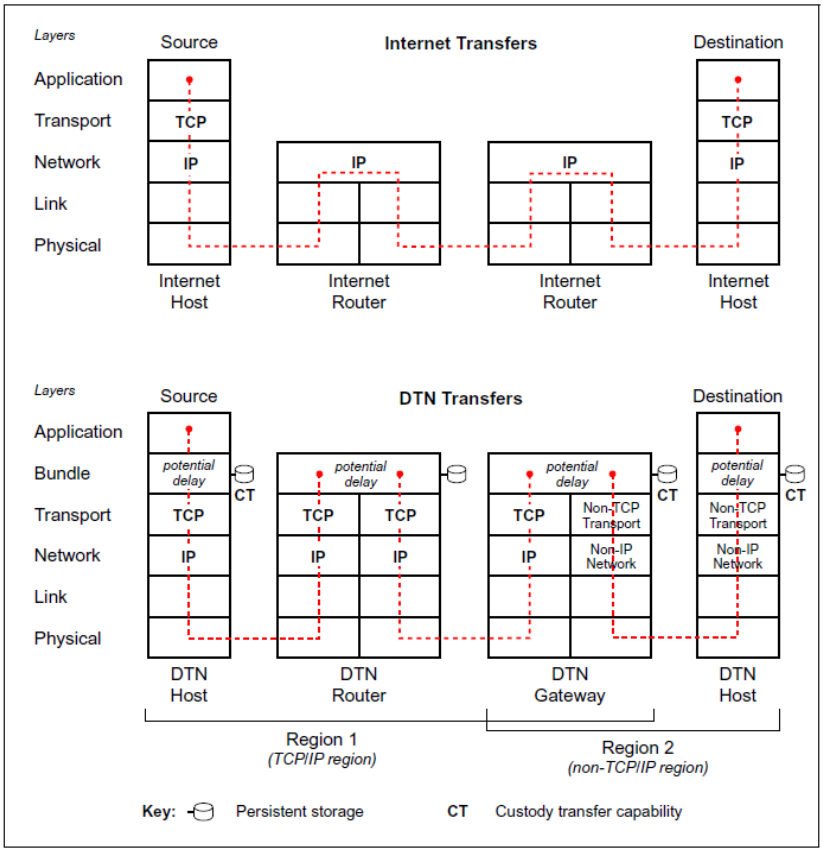

Figura 2.9 – Transmissão de pacotes sobre a internet convencional e sobre uma arquitetura DTN (WARTHMAN, 2003).

2.3.1 Latência

A latência da informação é pertinente de acordo com as prioridades do sistema. O sensoriamento da temperatura e pressão de uma caldeira notoriamente preza pelo acompanhamento em tempo real dos valores, visando à segurança, valores de temperatura e pressão corretas. Já monitorar uma carga lacrada transportada por navio permite intervalos na ordem de dezenas de minutos, uma vez que a baixa velocidade não traz grandes diferenças geográficas e possivelmente apenas o rompimento do lacre do contêiner geraria um registro para o sistema.

2.3.2 O Problema da Latência

A demanda por monitoramento em tempo real e interativo com sistemas embarcados muitas vezes é freada por causa da latência da transmissão e recepção dos dados do campo, que deve ser minimizada a fim de prover um sistema de tomadas de decisões eficiente. Redes DTN criam mecanismos interessantes para tratar deste problema, uma vez que possibilitam comunicação entre redes heterogêneas, permitem estratégias como comunicação oportunística (ALMEIDA, MOSCHETTO & GUARDIA, 2009) e soluções inteligentes, tais como utilização da transferência “sob custódia”, na tentativa de comutação da informação entre os nós presentes no campo até o destino da informação.

A instabilidade que a infraestrutura muitas vezes proporciona acarreta no armazenamento da informação por parte dos equipamentos embarcados que, por sua vez, transmitirão todo o buffer acumulado de forma oportunística (assim que houver conexão), eventos estes que geram picos de processamento e utilização do servidor. A figura 2.10 mostra uma situação típica que ocorre no agronegócio:

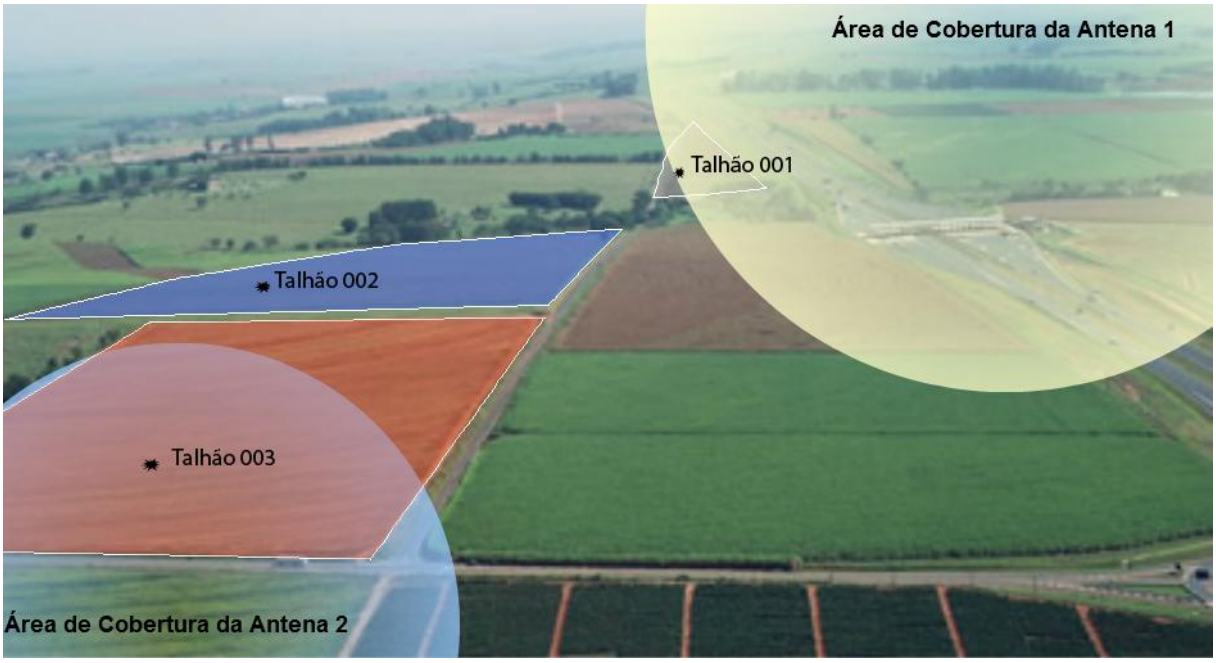

Figura 2.10 – Exemplo de áreas de conectividade sobre uma fazenda de cana-de-açúcar.

Para a concepção do problema, tomaremos o cenário onde uma frota agrícola composta por vinte máquinas realizou o seguinte itinerário: trabalhou por três horas no talhão 001, sete horas no talhão 002 e dez horas no talhão 003. Notem que praticamente não houve perda de conectividade durante a operação no talhão 001, significando carga constante para o servidor do sistema, uma vez que toda a mensagem gerada obteve disponibilidade de entrega. Para o talhão 002, o servidor ficou ocioso, o que é evidenciado pela falta de cobertura de sinal para a área trabalhada durante o período de tempo. É importante destacar o momento em que a frota passou a operar no talhão 003, que possui área parcial de cobertura; assim que o equipamento embarcado adquire conectividade, descarrega toda a informação armazenada em *buffer* de forma abrupta, gerando um grande pico de processamento e utilização de recursos do servidor. Toda a operação transcorrida sobre o talhão 003 será caracterizada por períodos com e sem conectividade, acarretando uma carga inconstante no servidor.

Transpondo este exemplo para a realidade, grupos que monitoram milhares de máquinas, onde muitas vezes almejam o monitoramento em tempo real (aumentando ainda mais o volume de informação que deve ser processada e disponibilizada prontamente para o sistema) necessitam de poderosos e caros servidores, somente para comportar estes picos.

2.4 Considerações Finais

O mercado atual disponibiliza uma ampla diversidade de soluções para comunicação sem fio, agregando vantagens e desvantagens para cada tipo de aplicação. Mesmo assim, ainda não é possível garantir uma comunicação persistente, com baixa latência a custos plausíveis.

Falhas na rede acarretam o armazenamento da informação, que de forma oportunística é transferida abruptamente para o servidor, o qual se encarrega de conferir, processar, disponibilizar e armazenar a informação. Para isso, poderosos e caros servidores são alocados para suportar o sistema de monitoramento, utilizados muitas vezes em períodos dispersos no tempo (picos).

Através de uma modelagem analítica e simulações, foi testada uma arquitetura distribuída de baixo custo a fim de otimizar os recursos computacionais para o monitoramento de equipamentos embarcados.

Capítulo 3

FUNDAMENTOS

Para o sucesso do projeto, os conceitos básicos para a concepção de uma arquitetura distribuída para a aplicação servidora devem ser discutidos. Através do estudo das linguagens de programação distribuídas e sistemas de filas, foi possível a modelagem do sistema que foi submetido a simulações e posteriormente implementado em Java.

Este capítulo faz uma breve descrição dos fundamentos utilizados para alcançar os objetivos previamente propostos. Aborda basicamente três frentes: chamadas de procedimentos, tecnologias que facilitam a criação da aplicação de forma distribuída e a teoria das filas, fundamental para a modelagem da aplicação a fim de prover alto desempenho.

3.1 Computação Distribuída

Segundo Tanenbaum e Steen (TANENBAUM & STEEN, 2002), “um sistema distribuído é um conjunto de computadores independentes que aparenta, ao usuário do sistema, ser um único computador”. Distingue-se de sistemas centralizados quanto à comunicação entre processos, uma vez que não compartilham a mesma memória, carecendo de recursos de rede para esta comunicação. Esta seção mostra que estas comunicações podem ser realizadas de forma transparente, utilizando soluções como Remote Procedure Calls (RPC), onde a ideia é realizar chamadas de procedimentos remotas o mais semelhante possível com chamadas locais.

3.1.1 RPC

O RPC não é um protocolo, e sim um modelo que tem por objetivo permitir a comunicação interprocessos entre máquinas heterogêneas, utilizando o modelo cliente-servidor, de forma transparente ao usuário. Foi projetado de forma que uma chamada de processo local é realizada da mesma forma que uma chamada de processo remota, inserindo mais um nível de abstração, escondendo todos os detalhes de comunicação, tais como localização, montagem e desmontagem da mensagem, entre outros.

Apesar das chamadas entre processos locais serem realizadas de forma idêntica às chamadas remotas, a execução de uma RPC é distinta, contemplando problemas referentes à comunicação em rede, tais como uma latência maior de comunicação e possíveis falhas entre os computadores que constituem efetivamente a chamada (um servidor pode sofrer uma falha crítica e não responder mais às chamadas de seus clientes, por exemplo).

Para compreender melhor o funcionamento de uma chamada RPC, é importante entender primeiro o funcionamento de uma chamada entre processos que ocorre na mesma máquina (*localhost*). Considere a função implementada na linguagem de programação C:

$$\text{count} = \text{read}\ (\text{fd}, \text{buff}, \text{nbytes});$$

Onde fd indica o arquivo a ser processado, buff é um *array* de caracteres e nbytes é um inteiro que indica quantos bytes deverão ser lidos. Se esta função for chamada através da função principal, a pilha de processos será a representada na figura 3.1(a). Para realizar esta chamada, quem a chama deve colocar os parâmetros na pilha conforme mostra a figura 3.1(b), lembrando que esta ordem inversa de parâmetros é em razão do critério utilizado pelo compilador C para utilizar esta função. Após executar este procedimento, o valor de retorno é colocado em um registrador, remove o endereço de retorno e transfere o controle novamente para quem a chamou. Finalmente, quem chamou este procedimento remove os parâmetros da pilha, que retorna ao seu estado original antes desta chamada.

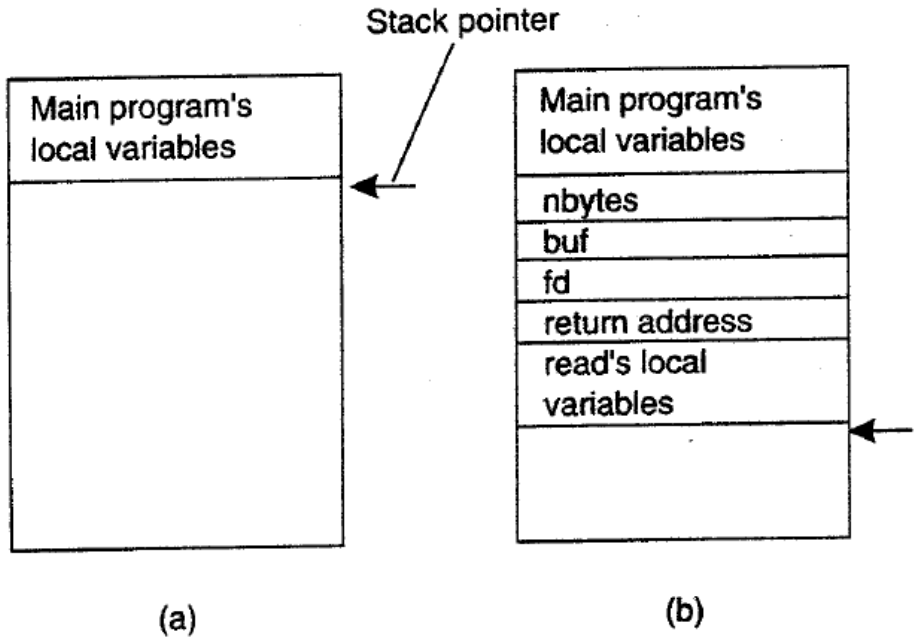

Figura 3.1 – (a) Parâmetros sendo passados em um procedimento local: Pilha antes da chamada. (b) A pilha enquanto o procedimento está ativo (TANENBAUM & STEEN, 2002).

Este exemplo mostra que uma chamada realizada de forma convencional não é viável porque não é possível passar parâmetros (por cópia ou por referência) para a comunicação remota entre processos, e é justamente aí que entra o modelo RPC. O modelo RPC encapsula esta chamada de forma que seja possível obter o resultado esperado, enviando-a para o destino e recuperando o resultado de forma correta e consistente, procedendo de duas formas, conforme mostra a figura 3.2:

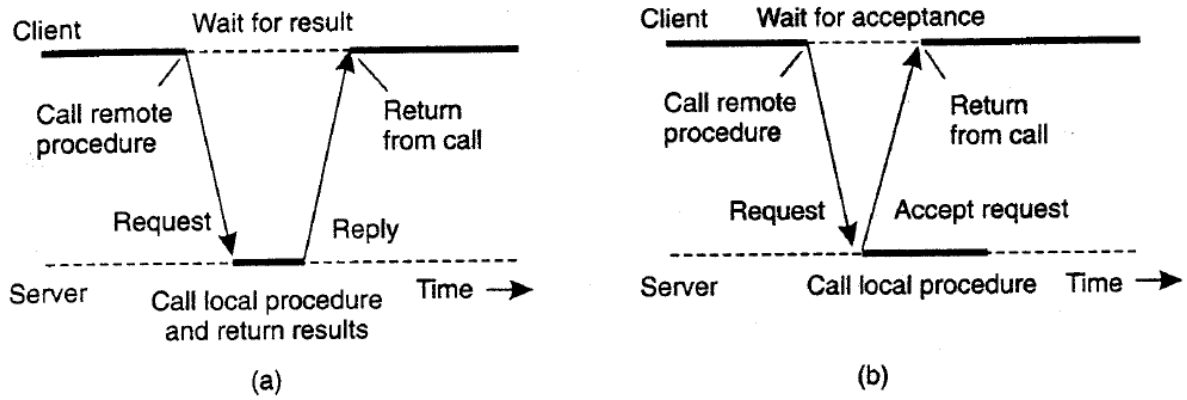

Figura 3.2 – (a) Chamada RPC Síncrona. (b) Chamada RPC Assíncrona (TANENBAUM & STEEN, 2002).

O encapsulamento consiste na criação de um Stub, que nada mais é do que uma mensagem contendo os valores dos parâmetros necessários para executar o

procedimento remotamente, além de contemplar o fato de que trafegará pela rede, necessitando de todas as informações pertinentes para encontrar o local de processamento.

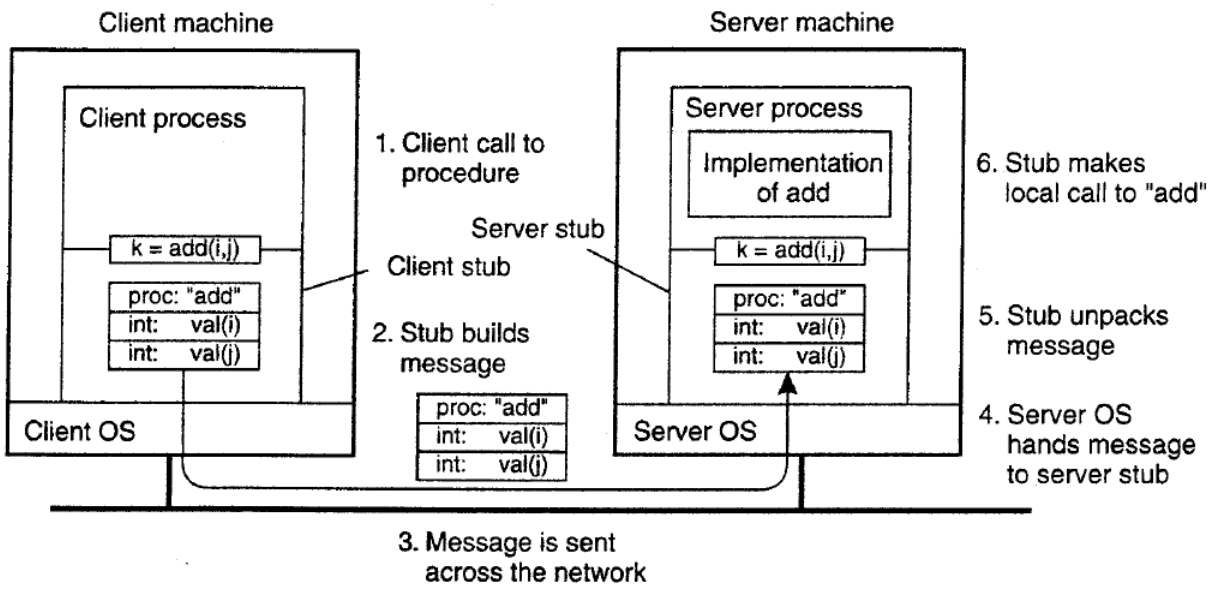

Figura 3.3 – Criação de um Stub, onde o cliente faz uma chamada remota para uma máquina servidora (TANENBAUM & STEEN, 2002).

No qual:

1. O procedimento cliente chama o stub cliente da forma normal;
2. O stub cliente constrói uma mensagem e chama o sistema operacional local;
3. O Sistema Operacional (SO) do cliente envia a mensagem para o SO remoto;
4. O SO remoto entrega a mensagem ao stub servidor;
5. O stub servidor desempacota os parâmetros e chama o servidor;
6. O servidor faz o trabalho e retorna o resultado para o stub;
7. O stub servidor empacota-o em uma mensagem e chama o SO local;
8. O SO do servidor envia a mensagem para o SO do cliente;
9. O SO do cliente entrega a mensagem para o stub cliente;
10. O stub desempacota o resultado e retorna para o cliente.

Desta forma, o modelo RPC permite que várias barreiras sejam vencidas, tais como localidade, diferenças de SO entre as máquinas cliente-servidor e o problema de passagem de parâmetros. Este conceito de modelo é fundamental para o

cumprimento da premissa de que máquinas heterogêneas podem compor o pool de máquinas servidoras que a aplicação irá utilizar.

3.1.2 SOAP

SOAP provém do acrônimo inglês Simple Object Access Protocol (Protocolo Simples de Acesso a Objetos). Consiste em um protocolo para troca de informações, estruturadas em uma plataforma descentralizada e distribuída. Baseia-se na Linguagem de Marcação Extensível (XML) para compor o formato de sua mensagem e normalmente baseia-se em outros protocolos da Camada de aplicação, mais notavelmente em Remote Procedure Calls (RPC) e The Hypertext Transfer Protocol (HTTP), para negociação e transmissão de mensagens. SOAP pode formar a camada base de uma pilha de protocolos de *web services*, fornecendo um framework de mensagens básico sob o qual os serviços web podem ser construídos. Este protocolo baseado em XML consiste de três partes, conforme mostra a figura 3.4: um envelope, que define o que está na mensagem e como processá-la, um conjunto de regras codificadas para expressar instâncias do tipo de dados definidos na aplicação e uma convenção para representar chamadas de procedimentos e respostas.

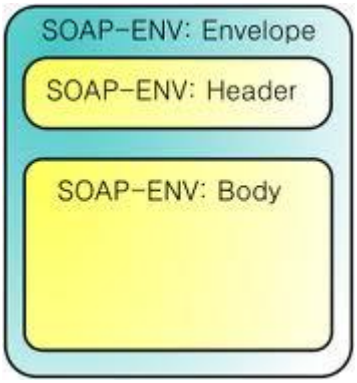

Figura 3.4 – Encapsulamento de uma mensagem SOAP(WIKIPEDIA, 2012).

Sua especificação define um framework que provê maneiras para se construir mensagens que podem trafegar através de diversos protocolos e que foi especificado de forma a ser independente de qualquer modelo de programação ou outra implementação específica. Por não se tratar de um protocolo de acesso a objetos, o acrônimo não é mais utilizado.

Geralmente servidores SOAP são implementados utilizando-se servidores HTTP, embora isto não seja uma restrição para funcionamento do protocolo. As mensagens SOAP são documentos XML que aderem a uma especificação fornecida pelo órgão W3C.

3.1.3 RMI

Java Remote Method Invocation (Java RMI) permite aos programadores criar aplicações distribuídas baseadas em tecnologias Java, onde os métodos do objeto remoto Java podem ser invocados de outras Java Virtual Machine (JVM), que pode estar localizada em outro host. RMI utiliza a serialização dos objetos para inserir e extrair os parâmetros do Stub, sem truncar os diferentes tipos, suportando verdadeiramente o polimorfismo orientado a objetos.

Basicamente, RMI consiste em um mecanismo de RPC em Java, porém contém grandes vantagens sobre uma chamada RPC comum porque é orientado a objetos. Sistemas de RPC tradicionais possuem linguagem de programação neutra, o que não proporciona funcionalidade plena para diferentes plataformas de sistemas operacionais. Como o RMI é executado pela JVM que, através da conversão do código em bytecodes, permite sua execução em diferentes sistemas operacionais, proporciona uma plataforma de desenvolvimento de aplicações distribuídas utilizando plenamente os recursos naturais da linguagem Java, permitindo inclusive passar objetos como parâmetro durante a invocação de um método (GRITZALIS, ILIADIS & OIKONOMOPOULOS, 2000).

De acordo com o documento fornecido pela Oracle (ORACLE 2012), usando Java RMI, os aplicativos podem criar objetos remotos que aceitam chamadas de método de clientes em outras JVM. Para que um cliente possa chamar métodos em um objeto remoto, o cliente deve ter uma maneira de se comunicar com o objeto remoto. Ao invés de ter que programar o cliente para que seja possível falar com o protocolo do objeto remoto, Java RMI usa classes especiais chamados stubs que podem ser enviados para o cliente e que são usados para comunicar-se e realizar chamadas com o objeto remoto. Para esta comunicação, o valor predeterminado na propriedade java.rmi.server.codebase representa um ou mais locais de URL a partir do qual esses stubs (e quaisquer classes necessárias pelos stubs) podem ser baixados. Esta operação acrescenta uma maior sobrecarga, uma vez que os

bytecodes podem ser baixados para locais diferentes, contudo, torna o processo de desenvolvimento da aplicação mais fácil, pois não necessita das chamadas “Interface Definition Language” (IDL) usadas, por exemplo, em CORBA, que possuem apenas tipos de dados primitivos.

Da mesma forma que *applets*, as classes necessárias para executar as chamadas de método remoto podem ser baixadas a partir de endereços URL (file:///). Entretanto, como *applets*, um endereço URL geralmente requer que o cliente e o servidor residam no mesmo hospedeiro físico, a menos que o sistema de arquivos referido pela URL seja disponibilizado usando algum outro protocolo, como o NFS portanto, geralmente as classes necessárias para executar chamadas de métodos remotos devem ser acessíveis a partir de um recurso de rede, como um servidor HTTP ou FTP.

Para definir os objetos remotos que, como citado anteriormente, não necessitam de linguagens IDL, basta que o programador crie uma interface que estenda a classe Remote, presente na biblioteca java.rmi.Remote. Em seguida, é interessante criar uma classe abstrata, contendo as funções que o objeto remoto deve fornecer (apenas para manter as boas práticas de programação) no qual, finalmente, pode criar a classe que estende a classe abstrata, sobrescrevendo e efetivamente implementando as funcionalidades das funções declaradas na classe abstrata. Seguindo este *template*, o reuso de código torna-se fácil e eficaz.

Após definido em linhas de código de uma aplicação quais serão os objetos remotos, uma utilização usual do Java RMI segue os seguintes passos que descrevem a figura 3.5:

1. O Codebase do objeto remoto é especificado pelo servidor do objeto remoto, definindo a propriedade java.rmi.server.codebase. O Java RMI servidor registra um objeto remoto, ligado a um nome, com o Java RMI Registry. O código base definido na JVM do servidor é anotado para a referência do objeto remoto no Java RMI Registry.
2. O cliente Java RMI solicita uma referência a um objeto remoto já registrado (nomeado). Esta referência (instância do objeto remoto stub) é que o cliente vai usar para fazer chamadas de método remoto para o objeto remoto.
3. O Java RMI Registry retorna uma referência (a instância do stub) para a classe solicitada. Se a definição de classe para a instância do stub

pode ser encontrada localmente em *CLASSPATH* do cliente, que é sempre procurado antes do codebase, o cliente irá carregar a classe localmente. No entanto, se a definição para o stub não é encontrado no *CLASSPATH* do cliente, o cliente irá tentar recuperar a definição da classe do código de base do objeto remoto.

4. O cliente solicita a definição de classe a partir da base de código. A base de código que o cliente usa é a URL que foi anotado para a instância stub quando a classe stub foi carregada pelo Registry. De volta para a etapa 1, o stub anotado para o objeto exportado foi então registrado no Java RMI Registry, vinculado a um nome.
5. A definição de classe para o stub (e qualquer outra classe (s) de que necessita) é baixado para o cliente. Note que os passos 4 e 5 são os mesmos passos que o Registry teve de carregar a classe do objeto remoto, quando o objeto remoto foi vinculado a um nome (ou seja, já foi registrado) no Java RMI Registry. Quando o registro tentou carregar a classe do objeto remoto stub, ele solicitou a definição de classe a partir da base de código associado a esse objeto remoto.
6. Agora o cliente tem todas as informações que precisa para invocar métodos remotos do objeto remoto. A instância stub age como um *proxy* para o objeto remoto que existe no servidor, por isso, ao contrário de um *applet* que utiliza uma base de código para executar código em sua JVM local, o Java RMI cliente usa o código de base do objeto remoto para executar código em outra (possivelmente remota) JVM, como ilustrado na Figura 3.6:

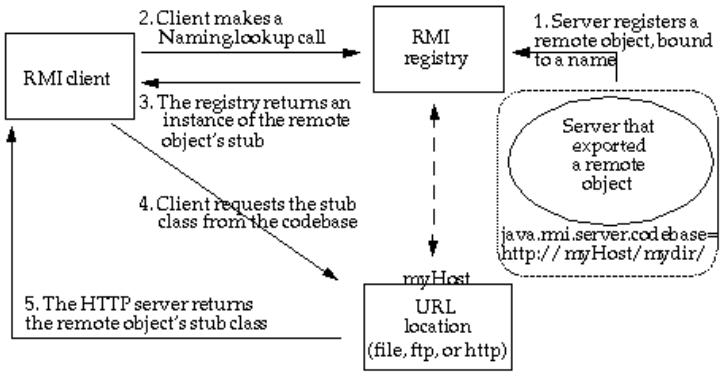

Figura 3.5 – Utilização usual do Java (ORACLE, 2012).

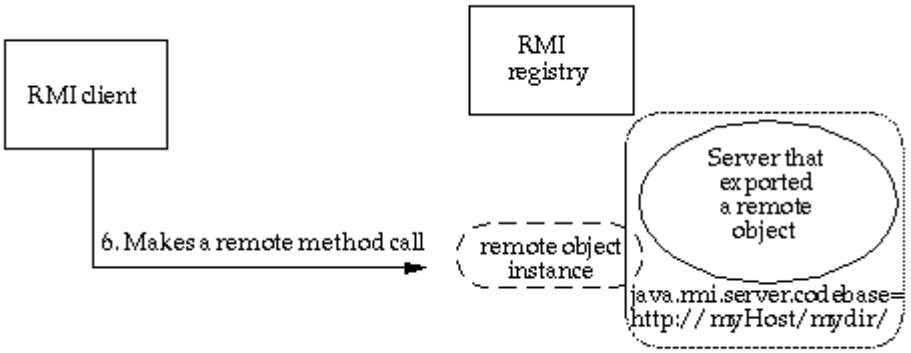

Figura 3.6 – Cliente RMI realizando uma chamada remota (ORACLE, 2012).

Concluindo, utilizar Java RMI fornece uma solução distribuída com todos os benefícios de utilizar a tecnologia Java (segurança, interoperabilidade entre plataformas, dentre outras), utilizando objetos remotos ao invés de solicitar as chamadas remotas, permitindo inclusive a interação entre aplicações baseadas e não baseadas em Java, utilizando a tecnologia Java IDL (ORACLE, 2012).

3.1.4 CORBA

A Common Object Request Broker Arquitecture (CORBA) é um padrão que foi desenvolvido pelo Object Management Group (OMG) (OMG, 2012) e consiste em uma arquitetura para computação distribuída orientada a objetos (arquitetura denominada OMA – Object Management Architecture). O núcleo da arquitetura CORBA é o Object Request Broker (ORB) que atua como um barramento sobre o qual o cliente interage com objetos locais ou remotos, de forma transparente, conforme mostra a figura 3.7, que descreve cada parte do ORB resumidamente. Esta figura representa o caminho de uma requisição da esquerda para a direita, abrangendo apenas as peças ORB que interagem com o cliente e a implementação do objeto, deixando de fora as partes que interagem com o sistema operacional e da rede.

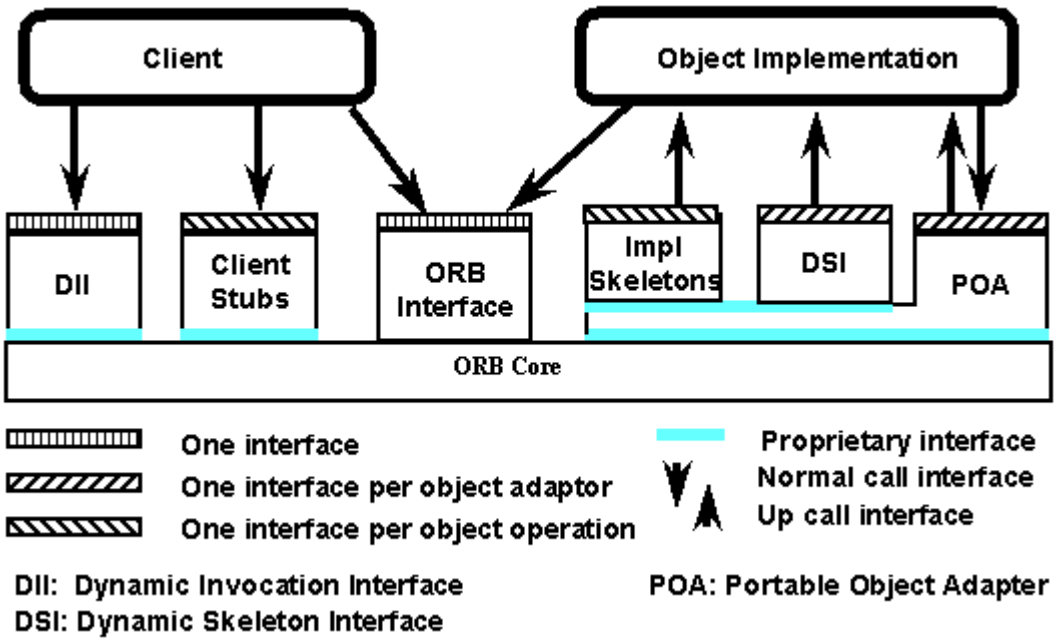

Figura 3.7 – Diagrama ORB (OMG, 2012).

O ORB fornece todos os mecanismos necessários para encontrar a implementação do objeto, realizar uma comunicação para uma requisição e carregar o resultado até o cliente, ou seja, realiza toda a comunicação necessária de forma a tornar a interação possível e transparente. A implementação do objeto interage com o ORB através do Portable Object Adaptor (POA) e através da interface ORB. A interface é definida através da Interface Definition Language (IDL).

Esta arquitetura também especifica um protocolo para comunicação entre ORBs, chamado Internet Inter-ORB Protocol (IIOP). É, portanto, uma especificação que pode ser usado em diversas plataformas de sistema operacionais de grande porte (por exemplo, UNIX e Windows), contanto que haja uma implementação ORB para aquela plataforma.

No CORBA, a implementação do ORB não precisa apresentar um único componente. Ao invés disso, basta que o ORB seja definido por suas interfaces, tornando qualquer implementação adequada às interfaces aceitável. É possível ainda que haja a coexistência de diferentes implementações de ORB, com representações distintas para a referência para objetos e modos distintos de invocação de operações. Para que haja tal coexistência, onde há a possibilidade de um cliente ter acesso simultâneo a objetos gerenciados por ORBs diferentes, é

importante que os próprios ORBs sejam capazes de distinguir as suas referências a objetos, o que deve ser feito de forma transparente para o cliente.

Enfim, para compreender melhor a arquitetura CORBA, o resumo de Marcelo de Paiva Guimarães (GUIMARÃES, 2012) define as seguintes entidades:

1. Cliente: É a entidade que efetivamente faz as requisições para um determinado objeto. Conforme visto nas RPC, esta requisição é transparente ao cliente, sendo realizada como se fosse uma chamada local. Ainda seguindo este conceito, o cliente tem acesso à referência do objeto e, de acordo com os métodos públicos declarados na sua interface, tem acesso à estrutura lógica do objeto sem acessar diretamente os mecanismos de comunicação utilizados (endereço, protocolo, dispositivo de rede, dentre outros), uma vez que este processo é realizado de forma transparente, lembrando-se da portabilidade, onde devem ser capazes de trabalhar em qualquer ORB que suporte o mapeamento de linguagem existente sem qualquer alteração no código fonte.
2. Objeto: Disponibiliza a semântica do objeto, implementando os métodos e tipos de dados descritos nas interfaces, suportando praticamente todas as possibilidades de codificação. Da mesma forma que o cliente, a portabilidade é essencial e devem ser portáveis para quaisquer ORBs que suportem o mapeamento de linguagem adequado. Esta portabilidade e não dependência das implementações dos objetos em relação aos ORBs é fornecida pelos Adaptadores de Objetos.
3. Referências a objetos: São dependentes do mapeamento da linguagem e da implementação do ORB. Consiste nas informações necessárias para especificar um objeto em um ORB.
4. Interface Definition Language (IDL): Pelas definições IDL, é possível mapear objetos CORBA em diferentes linguagens de programação. Define os tipos dos objetos, especificando suas interfaces, que compõe o conjunto de operações e parâmetros das possíveis operações.
5. Interface ORB: Presentes em todas as implementações de ORB, contém operações (úteis tanto aos clientes quanto às implementações) que são comuns a todos os objetos, independentemente da interface

do objeto ou do adaptador de objetos. Em função dessa independência, as operações disponíveis na interface ORB são escassas.

6. Stubs de Cliente (client stubs): Realizam as chamadas para o ORB utilizando interfaces privadas ao núcleo do ORB que está sendo utilizado e são responsáveis pelo acesso às operações definidas na IDL para um objeto, com o intuito de facilitar a programação.
7. Esqueleto da Implementação (Skeletons): Estruturas que permitem o ORB fazer as chamadas às rotinas existentes na interface para os métodos que implementam cada tipo de objeto. Contudo, esta estrutura não é obrigatória, pois, apesar de não ser recomendado, é possível escrever um adaptador de objetos que não se utiliza deste esqueleto para invocar os métodos de implementação.
8. Adaptadores de Objetos: Devido a grande variedade entre cada tipo de objeto (estilo de implementação, tempo de vida, políticas, dentre outros), os adaptadores de objetos possibilita que o ORB acesse rotinas com interfaces comuns. Consiste em um caminho básico pelo qual a implementação do objeto acessa os serviços do ORB, tais como: geração e implementação das referências para objetos, invocação de métodos, ativação e desativação dos objetos e suas implementações e registro das implementações.
9. Mapeamento para Linguagens de Programação: O mapeamento do CORBA para uma linguagem de programação em particular deve ser independente da implementação de ORB utilizada, uma vez que o acesso aos objetos do CORBA pode ser diferente para cada linguagem de programação (uma LPOO pode querer visualizar os objetos do CORBA como objetos da própria linguagem). A especificação do CORBA para o mapeamento de uma linguagem de programação inclui todos os 12 itens aqui descritos, porém até o momento, a especificação CORBA possui apenas o mapeamento para a linguagem C. Outro requisito que também é definido no mapeamento para linguagem é a interação entre invocação dos objetos e o fluxo de controle tanto no cliente quanto na implementação (com chamadas síncronas ou assíncronas).

10. Interface de Invocação Dinâmica: Realiza a construção dinâmica para invocação de objetos. O cliente pode especificar (através de uma sequência de chamadas) o objeto a ser invocado, a operação que será executada e o conjunto de parâmetros para esta operação. Através do repositório de interfaces ou de outra fonte em tempo de execução, informações sobre a operação e os parâmetros necessários podem ser obtidas para realizar a invocação.
11. Repositório de Interfaces: É um serviço que pode ser usado pelo ORB para execução dos pedidos e provê objetos persistentes que representam a informação IDL em forma disponível em tempo de execução possibilitando a localização de um objeto cuja interface é desconhecida, e determinando que operações estarão para estes objetos.
12. Repositório de Implementações: Contém informações necessárias para que o ORB localize e utilize as implementações dos objetos.

Em resumo, O ORB é responsável por todo o trabalho de baixo nível, provendo a transparência tanto na comunicação quanto na transferência dos dados entre o cliente e o objeto. Para invocar um objeto, o cliente deve acessar stubs específicos de um objeto qualquer, como se fosse uma chamada padrão de qualquer rotina. Enquanto que para o cliente esta chamada de rotina é usual, o stub interage com o ORB para efetivamente realizar todas as operações necessárias para esta chamada.

3.2 Teoria das Filas

O processo evolutivo do ser humano que culminou na atual sociedade moderna faz com que em diversas ocasiões esperemos para sermos atendidos. Isto ocorre em supermercados, filas de banco, congestionamentos e em outras infinidades de ocasiões. Leonard Kleinrock (KLEINROCK, 1975) publicou um estudo aprofundado sobre como representar estas filas de forma matemática, desde o

processo de chegada de novos clientes a tipos distintos de sistemas que se utilizam de filas.

Em suma, a formação de filas ocorre quando a procura pelo serviço é maior que a capacidade do sistema para atender esta demanda. Desta forma, a Teoria das Filas tenta através das análises matemáticas mapear o processo a fim de mensurar informações valiosas, tais como capacidade, utilização e se possível, pontos de equilíbrio para o sistema.

As seções a seguir (3.2.1 a 3.2.3) foram baseadas no livro “Avaliação de Desempenho de Sistemas Computacionais” (JOHNSON & MARGALHO, 2011).

Um sistema de fila é caracterizado através de uma linha de entrada, o sistema de fila de fato (numero de filas, servidores em paralelos ou em série, ou seja, efetivamente a modelagem do sistema, conforme mostra a figura 3.8) e uma linha de saída. Na linha de chegada os clientes (Jobs, pacotes, clientes, etc) chegam com tempo arbitrário A(t), que se encaminham para o sistema de filas, onde são servidos em um tempo arbitrário B(x), e depois são liberados em uma linha de saída.

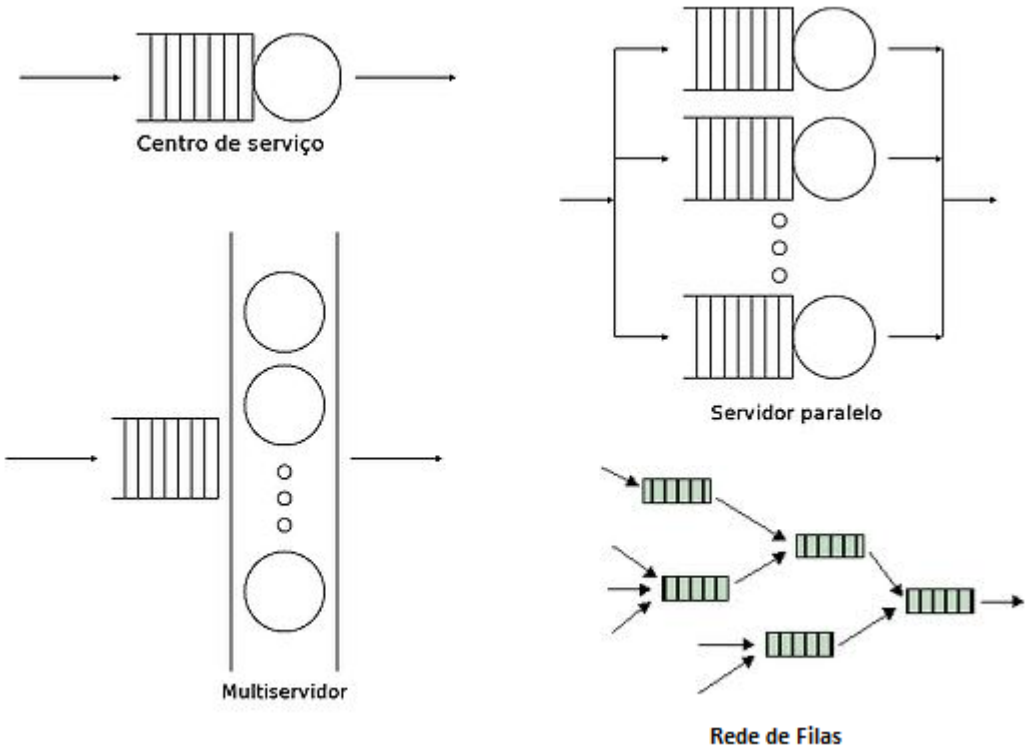

Figura 3.8 – Representação de Sistemas de Filas (WIKIPEDIA, 2012)

Esta seção trata exclusivamente de sistemas computacionais providos de filas, fazendo um breve resumo dos principais conceitos utilizados no projeto.

3.2.1 Variáveis relacionadas a uma fila

Após identificar o fluxo de processamento que será responsável por atender um cliente, algumas variáveis são importantes para mensurar as características deste sistema de filas e serão caracterizadas nesta seção.

Número de Clientes

Variável cujo nome é intuitivo, representa o número de clientes em diferentes estágios no sistema de fila:

- C_n denota o n-ésimo cliente que entra no sistema;
- n_w representa o número de usuários aguardando atendimento. É importante notar que este número é sempre menor que *n*, uma vez que não contempla os usuários que estão sendo atendidos;
- n_S é o número de usuários que estão sendo atendidos dentro do servidor;
- *n* é o número total de clientes no sistema em um determinado instante de tempo, incluindo os clientes em atendimento e os que estão aguardando na fila.

Portanto:

$$E[n] = E[n_w] + E[n_S]$$

Taxa de Chegadas

A taxa de chegadas ($\boldsymbol{\lambda_n}$) é o tempo de chegada de C_n clientes no sistema. Representa também o número médio de chegadas de clientes por unidade de tempo, sendo o inverso do tempo de chegadas:

$$\lambda = \frac{1}{A(t)}$$

Facilmente podemos verificar que se 10 clientes chegaram em um intervalo de 2 minutos, temos 10/2 = 5 clientes por minuto.

Taxa de Serviço

A taxa de serviço (**μ**) é a taxa média por cliente que cada servidor gasta para servi-lo. Este valor refere-se a quantidade de clientes que o servidor consegue atender por unidade de tempo:

$$\mu = \frac{1}{B(x)}$$

É importante definir que para *m* servidores conectados a uma mesma fila, a taxa total de serviço é *mμ.*

Tempo de Espera

O tempo de espera (**w**) é o tempo que o cliente espera na fila antes de ser atendido pelo servidor. Consiste simplesmente em um valor descrito em uma unidade de tempo (por exemplo, 5 segundos).

Tempo de Resposta

O tempo de resposta (**s**) é o tempo médio total gasto por um cliente, desde a entrada no sistema, tempo médio na fila, tempo médio de atendimento até sua saída:

$$E[s] = E[w] + E[x]$$

Portanto, para um sistema de filas onde o servidor demora em média 2 segundos para atender a uma requisição, o tempo médio na fila para cada requisição é 3 segundos, o tempo de resposta será 5 segundos.

Lei de Little (KEILSON & SERVI, 1988)

Se o sistema não descarta os clientes (pode ocorrer com buffers insuficientes, por exemplo), o número médio de usuários no sistema será a taxa de chegadas multiplicada pelo tempo médio de resposta:

$$E[n] = \lambda * E[s]$$

De forma análoga, o número médio de usuários na fila é calculado a partir da taxa de chegadas multiplicada pelo tempo médio de espera:

$$E[n_w] = \lambda * E[w]$$

Utilização

A utilização do sistema **U(t)** denota a fração de tempo na qual o servidor está ocupado. O valor de U(t) varia entre 0 e 1, onde 0 representa que o sistema está vazio (o servidor não está em uso) e 1 representa que o sistema está 100% do tempo ocupado. A fórmula de obtenção da utilização varia de acordo com o tipo de

fila estudado (KLEINROCK, 1975), mas em suma é o cálculo do tempo que o servidor ficou ocupado dividido pelo período de observação.

Intensidade de Tráfego

A intensidade de tráfego (***p***) mede o congestionamento do sistema de fila. Valores próximos de 0 indicam pouca fila e em geral, quando o valor da intensidade de tráfego aumenta, o tamanho da fila também aumenta.

$$p = \frac{tempo\_médio\_serviço}{tempo\_médio\_chegadas} = \frac{taxa\_chegadas}{taxa\_serviço} = \frac{\lambda}{\mu}$$

Através do valor de *p* é possível formular a condição de estabilidade, que deve ser $p < 1$, ou seja, a média dos números de requisições que chega deve ser menor que o número de requisições que pode ser servido, em uma unidade de tempo. Em geral, para *m* servidores:

$$p = \frac{\lambda}{m\mu}$$

Vazão

Para filas, vazão é a média das requisições processadas em uma unidade de tempo, ou seja, é a taxa de saída do sistema. Para sistemas em equilíbrio, a taxa de saída é igual à taxa de entrada, então podemos definir vazão como:

$$\lambda = mp\mu$$

3.2.2 Notação de Kendall (KENDALL, 1953)

Criada em 1953 por D. G. Kendall, esta notação é composta por três parâmetros funcionais que caracterizam os sistemas de filas. Atualmente, a notação possui seis parâmetros e segue o formato A/S/m/B/K/SD, onde:

- **A** é a distribuição do processo de chegadas;
- **S** é a distribuição do tempo de serviço;
- **m** é o número de servidores;
- **B** é a capacidade do sistema;
- **K** é o tamanho da população;
- **SD** é a disciplina de atendimento da fila.

Distribuição de Chegadas (A) e Tempo de Serviço (S)

Define o tipo de distribuição estatística para representar a chegada ou processamento do serviço. Algumas das distribuições mais usadas são:

- **M** – Processo de Poisson (ou randômico);
- $\mathbf{E_k}$ – Distribuição de Erlang de parâmetro k;
- **D** – Tempo determinístico ou fixo;
- **G** – Distribuição genérica;
- **GI** – Distribuição genérica com tempo entre chegadas independentes.

Número de Servidores (m)

Representa o número de servidores disponíveis no sistema, onde obrigatoriamente este valor deve ser maior que um.

Capacidade do Sistema (B)

Denota o número máximo de clientes permitidos dentro do sistema em certo momento. Quando o valor máximo é alcançado, clientes novos que chegam serão descartados. Se este número for omitido na notação, assume-se que a capacidade é ilimitada.

Tamanho da população (K)

É o tamanho da população de clientes interessados em usar o sistema. Por exemplo, um servidor de e-mails possui 1000 usuários, mas não quer dizer que todos estão utilizando o sistema neste instante. Se este valor for omitido na notação, considera-se que o tamanho da população é infinito.

Disciplina de Atendimento da Fila (SD)

Denota a ordem de atendimento dos clientes que estão presentes na fila. Algumas das principais disciplinas são:

- **FIFO/FCFS** – First In First Out / First Come First Server (o primeiro que chega é o primeiro a ser servido);
- **LIFO/LCFS** – Last In First Out / Last Come First Server (o último a chegar será o primeiro a ser servido, como ocorre em uma pilha);
- **SIRO** – Service In Random Order (serviço em ordem randômica);

- **RR** – Round Robin (cada cliente tem uma fatia de tempo em que é servido, onde assim que seu tempo acaba, outro processo ganha o servidor, até chegar novamente sua vez ou acabar);
- **PS** – Processor Sharing (semelhante ao RR, porém com um tempo infinitesimal, aparentando que todos os processos estão sendo servidos simultaneamente com tempo de serviço aumentado);
- **IS** – Infinite Server (servidores infinitos onde nunca se formam fila).
- **Priority** – Prioridades de classes de usuários diferentes são definidas, fazendo com que um cliente seja atendido de acordo com sua prioridade.

3.2.3 Redes de Filas

Uma rede de filas consiste em uma coleção de centros de serviços e clientes, ou seja, caracterizam-se por ser um sistema que contém duas ou mais filas conectadas ou não entre si, conforme mostra a figura 3.9:

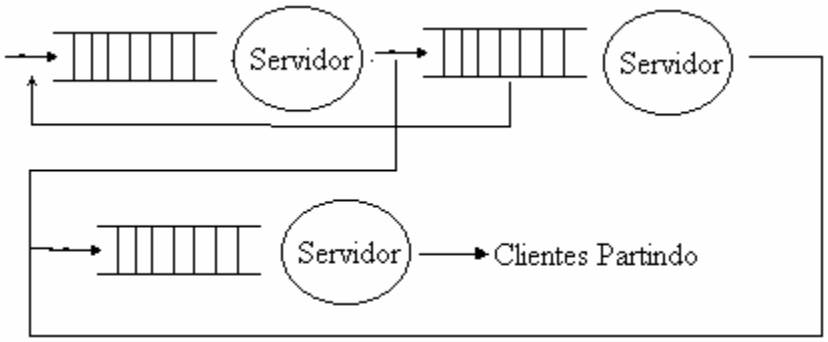

Figura 3.9 – Representação de Redes de Filas (WIKIPEDIA, 2012)

Para estas redes, podemos classifica-las como:

- Redes de Filas Abertas: as que permitem chegadas e saídas de requerimento do e para o exterior;
- Redes de Filas Fechadas: as que não permitem chegadas nem saídas de requerimentos do e para o exterior, havendo um número de requisições constante;
- Rede de Filas Mistas: Possuem partes abertas e parte fechadas simultaneamente.

Onde as variáveis operacionais para Redes de Filas são:

- A_i é o número de chegadas de transações ao servidor *i*;
- C_{ij} é o número de transações que saem do servidor *i* e vão para o servidor *j*;
- C_{i0} é o número de transações que saem do servidor *i* e deixam o sistema;
- C_0 é o número de transações que deixam o sistema;
- V_i é a taxa relativa de visitas ao servidor *i*. Equivale ao número médio de visitas ao dispositivo por requisição:

$$V_i = \frac{C_i}{C_0}$$

- D_i representa a demanda do serviço no servidor *i.* Demanda representa o tempo de serviço total para todas as visitas da mesma requisição ao servidor *i.*
- *X ou* X_0 é a taxa de processamento ou vazão do sistema:

$$X = \frac{C_0}{T}$$

- B_i é o tempo ocupado do servidor *i*;
- S_i é o tempo de serviço do servidor *i*;
- R_i é o tempo de resposta do servidor *i.*

A partir desta nova definição, podemos calcular valores pertinentes a algumas leis operacionais para redes de filas:

Lei do Fluxo Forçado

Esta lei mostra que o fluxo (vazão) em todas as partes do sistema deve ser proporcional:

$$X_i = \frac{C_i}{T} = \frac{C_i}{C_0} = V_i X_0$$

Lei do Tempo de Resposta

O tempo de resposta de uma rede de filas depende do tempo de resposta de cada fila versus o número de visitas que a mesma requisição faz ao servidor daquela fila. Portanto, para um sistema com *K* filas:

$$R = \sum_{i=1}^{K} V_i * R_i$$

Juntando a Lei do Tempo de Resposta com o Teorema do Tempo de Resposta tem-se:

$$R = \sum_{i=1}^{K} \frac{D_i}{1 - U_i}$$

Lei da Demanda de Serviço

Pode ser relacionada à vazão do sistema e sua utilização:

$$D_i = V_i S_i = \frac{X_i}{X_0} * \frac{U_i}{X_i} = \frac{U_i}{X_0}$$

3.3 Fundamentos aplicados na arquitetura

O sistema de telemetria de frotas compõe uma rede singular e heterogênea de sistemas embarcados no campo, infraestrutura de comunicação diversificada e o *pool* de servidores proposto para o processamento deste montante de informações.

Para viabilizar as prerrogativas iniciais, as estratégias tecnológicas utilizadas para a implementação da solução foram baseadas na praticidade de desenvolvimento e alto desempenho. Como o *pool* de servidores compõe um sistema distribuído, as chamadas RPC propiciam grande flexibilidade para o nó mestre transferir uma requisição para outro nó do *pool*, podendo utilizar CORBA ou RMI para tal. Contempla também o protocola SOAP, que orienta as chamadas dos usuários de uma forma portável para os nós heterogêneos.

Finalmente, através dos fundamentos da teoria das filas, o arranjo interno da arquitetura pôde ser concebido e testado previamente, a fim de propiciar alto desempenho para a solução, atendendo as diferentes prioridades das classes distintas existentes no sistema.

Capítulo 4

ESTUDO DE *TRACES* REAIS

Para suprir as necessidades de processamento, o estudo de uma arquitetura distribuída de alto desempenho, baixo custo e alta escalabilidade, é preciso primeiramente estabelecer alguns conceitos para viabilizar este estudo, contemplando as diferentes disciplinas necessárias para o processo investigativo.

Para garantir o sucesso do projeto, este capítulo aborda todo o caminho investigativo que culminou em uma arquitetura para o servidor de aplicação de telemetria de ativos motomecanizados de alto desempenho, escalável, preditivo e de baixo custo.

Através da análise de dados reais fornecidos por empresas do ramo de telemetria de frota veicular, foi possível identificar as classes de informações que a aplicação deverá processar, assim como suas taxas de chegadas. Outras informações também foram obtidas, as quais dizem respeito à telemetria (local onde foram geradas as mensagens) e as condições da infraestrutura local para a transmissão.

4.1 Estudo de viabilidade

Sistemas distribuídos são eficientes principalmente quando o processamento da informação é independente. Cada mensagem transmitida por um sistema embarcado é única e pode ser descompactada de forma livre de contexto, por qualquer unidade computacional.

Esta seção apresenta uma simulação que preliminarmente demonstra um potencial significativo de desempenho realizando a distribuição do processamento para esta aplicação. Foi desenvolvido um gerador de mensagens e um pequeno servidor, para simular a interação entre um equipamento embarcado em campo utilizando-se dos recursos servidores da aplicação. Ambos possuem estrutura *multithread*, buscando otimização dos processos para equipararem-se com implementações convencionais de alto desempenho, seguindo os respectivos fluxogramas de representação demonstrados na figura 4.1:

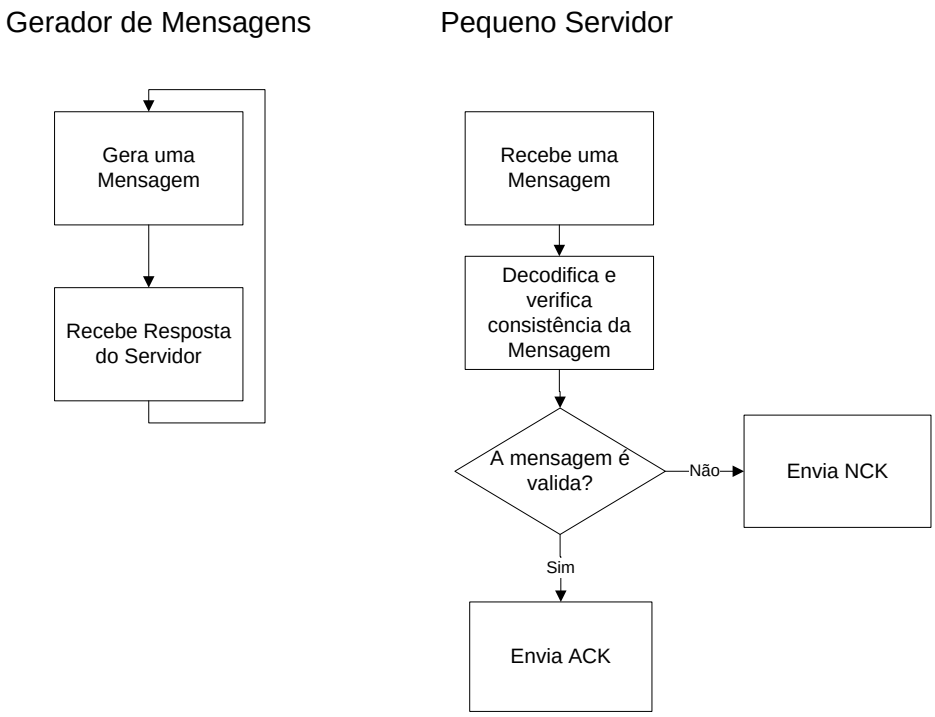

Figura 4.1 – Fluxograma do gerador e servidor de mensagens.

O objetivo do teste é verificar se o poder de processamento de mensagens aumenta proporcionalmente ao investimento em hardware do servidor, a fim de validar a premissa de distribuição do processamento para redução de custos. Foram realizados testes em quatro máquinas, cujas configurações estão descritas na tabela 4.1, repetindo-os cinco vezes por máquina para consolidar a média de desempenho. O tráfego de informações foi conduzido em *Localhost*, para desconsiderar o atraso da rede, alocando dezesseis *threads* para a aplicação servidora (encarregada de receber, decodificar, validar e retornar o resultado) e dezesseis *threads* para a aplicação cliente (geradora das mensagens). Durante dez minutos, a média de

mensagens decodificadas por segundo foi coletada a cada quinhentos milissegundos, resultando no gráfico da figura 4.2.

Tabela 4.1 –Especificação dos computadores testados.

	Quantica	Dirac	Quad	PC Home
Processador	Xeon 5620	Xeon 5410	Intel 2-Quad	Atlon XP 4200
Frequencia (MHz)	2.400	2.333	2.333	2.200
Cores	8 (Hiper Threads)	8	4	2
RAM (GB)	16	8	8	1,5
Sistema Operacional	Open Suse Linux	Arch Linux	Mandriva Linux	Arch Linux
Valor (R$)	7.300,00	6.000,00	2.000,00	300,00

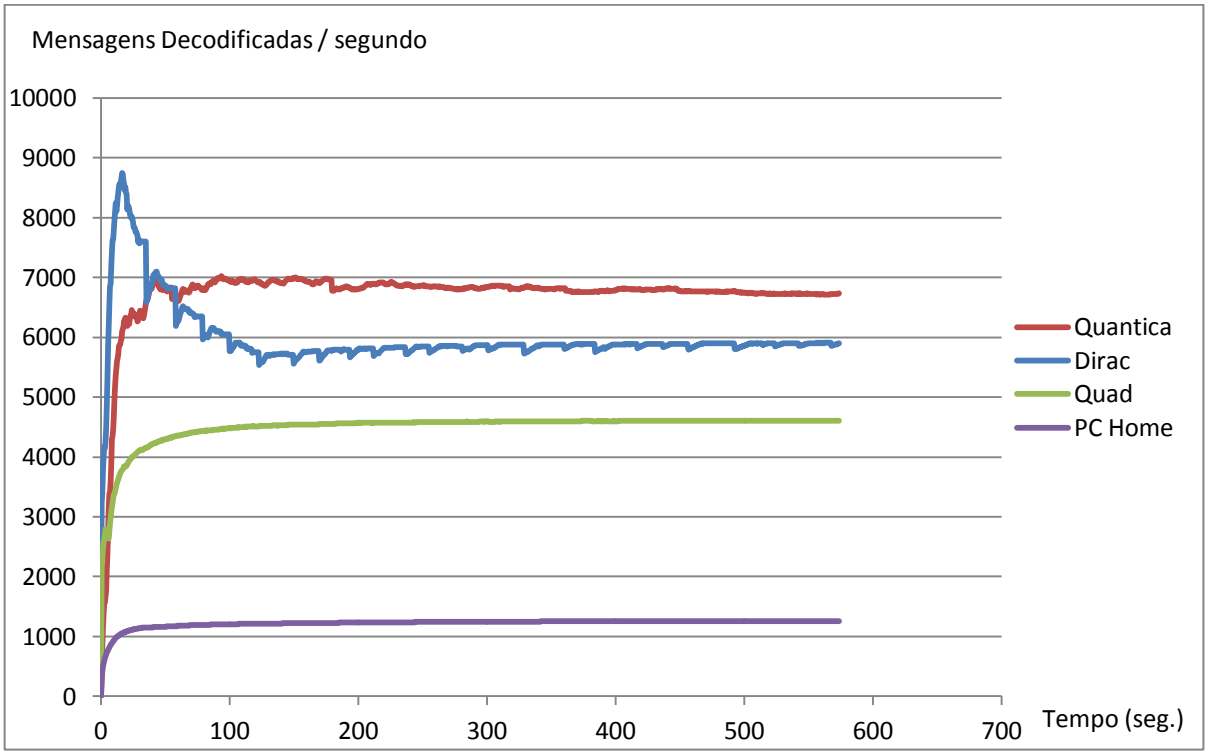

Figura 4.2 – Resultado do processamento de mensagens.

O resultado preliminar sugere que grandes investimentos em hardware não significam um aumento de desempenho proporcional, fato definido pela lei de

Amdahl (EYERMAN & EECKHOUT, 2010) e mostrado um caso semelhante por PERKINSON *et al* (PERKINSON et al, 1994). Trabalhando com o processamento em paralelo das mensagens, a hipótese foi baseada na utilização de mais de um servidor com custos reduzidos para atingir o resultado esperado.

4.2 Estudo de *traces* reais

Para melhor compreensão, tanto do cenário quanto do fluxo de informações que efetivamente um servidor de monitoramento de ativos motomecanizados deve ser capaz de processar, buscou-se parcerias com empresas do ramo. Esta busca resultou no fornecimento de informações de duas empresas: um *dump* do banco de dados de uma empresa de telemetria de ativos motomecanizados para o agronegócio (contendo instante de geração da mensagem, tipo e instante de processamento da mesma pela aplicação servidora), monitorando frotas de uma média de cem ativos e um log contendo o número de bytes que chega por segundo de um servidor de grande porte, de monitoramento de frota veicular em âmbito nacional (ordem de milhares de veículos).

Segue uma discussão sobre os tipos de mensagens geradas, o fluxo de entrada nos diferentes portes de servidores e um estudo de infraestrutura do território brasileiro, pertinente a melhor compreensão do fluxo que chega para os servidores de monitoramento de ativos motomecanizados.

4.2.1 Mensagens geradas pelos sistemas embarcados

O *dump* fornecido por uma empresa especializada em telemetria de ativos motomecanizados para o agronegócio, contendo 12.743.145 mensagens, contemplou o período entre 09.07.2011-11h03min42s a 14.09.2011- 08h43min26s. As mensagens foram fornecidas na íntegra, possuindo informações pertinentes ao posicionamento global, valores de sensores e interações entre o operador do veículo com o sistema. Através destas informações e contribuições da empresa que forneceu o *dump*, foi possível discriminar três tipos distintos de mensagens geradas pelos sistemas embarcados quanto à finalidade e conteúdo da informação. A tabela

4.2 evidencia os tipos de mensagens e a taxa de geração média para cada sistema embarcado.

Tabela 4.2 –Tabela descritiva dos tipos de mensagens existentes no sistema.

Tipo de Mensagem	Prioridade[1]	Taxa de Geração (λ) / segundo
Controle	4	0,075148122
Produtividade	3	1
Operacional	2	1

[1] Quanto maior o valor, maior é a prioridade (valores de 4 a 0).

Na qual:

1. Mensagem de Controle: São mensagens transmitidas ou recebidas pelos sistemas embarcados referentes a ajustes de configurações, parametrização de valores (por exemplo, um ajuste de configuração que determina a faixa ideal de valores de um sensor de monitoramento de temperatura, indicação de uma troca de operador, etc.) e envio de alarmes de não conformidade, informações de maior prioridade para este sistema, devendo ser decodificada e processada prontamente pela aplicação servidora (rever seção 2.1.1);
2. Mensagem de Produtividade: Contém valores referentes à produtividade da operação que o ativo motomecanizado realiza. Pode ser o número de árvores colhidas para atividades do ramo florestal, tonelada de cana-de-açúcar para o setor sucroalcooleiro, área trabalhada, dentre outros. É uma informação muito valiosa para uma boa gestão da operação, uma vez que possibilita acompanhar toda a frota e verificar o cumprimento ou não da meta diária. Se ao longo da operação a produtividade não seguiu o planejado, planos de ação podem ser tomados para garantir o cumprimento da meta. Outro ponto interessante é que a produtividade de uma máquina é diretamente proporcional ao lucro da operação, carecendo de estudo e acompanhamento constante para minimizar os custos e aumentar a produtividade (ARAÚJO, 2002).

3. Mensagem Operacional: Enriquece as informações supracitadas, fornecendo informações detalhadas sobre a operação, com o intuito de rastrear eventos, tais como nome do operador, tipo de operação (colhendo, plantando, arando, dentre outras), informes de motivos de paradas (manutenção, almoço, troca de turno, etc) e os valores dos sensores presentes no ativo, como medição de temperaturas, RPM, velocidade, dentre outros. De fato, quando uma baixa produtividade for detectada, com as mensagens operacionais é possível identificar se houve erros do operador, falhas mecânicas ou erros logísticos, onde uma colhedora ficou muito tempo parado por não haver transbordos suficientes no talhão.

A tabela 4.2 mostra de forma sutil uma informação interessante. Enquanto que as mensagens de controle são geradas por eventos, como um alarme de temperatura da água do radiador indicando superaquecimento do motor ou uma interação humana, as mensagens operacionais e de produtividade são geradas automaticamente, de forma sazonal (no caso, de segundo a segundo, podendo ser configurado de acordo com a necessidade).

Como a natureza das mensagens são distintas, é importante ressaltar que elas serão decodificadas e processadas por processos diferentes e especializados, e que a prioridade de uma mensagem de produtividade pode ser invertida com a operacional, de acordo com a necessidade do usuário. Contudo, este fato não altera em nada o processamento das informações, conforme será demonstrado na seção 4.3.

4.2.2 Fluxo de mensagens para o servidor da aplicação

Após identificar os diferentes tipos de mensagens e suas taxas de geração, com o mesmo *dump* do banco de dados, foi possível analisar o quanto de informação chega para a aplicação servidora a cada segundo, evidenciando as rajadas de informações que as sombras de conectividade acarretam. Para padronizar os gráficos das diferentes empresas, será mostrada a quantia de Bytes por segundo que chega a cada servidor, conforme mostra as figuras 4.3 e 4.4:

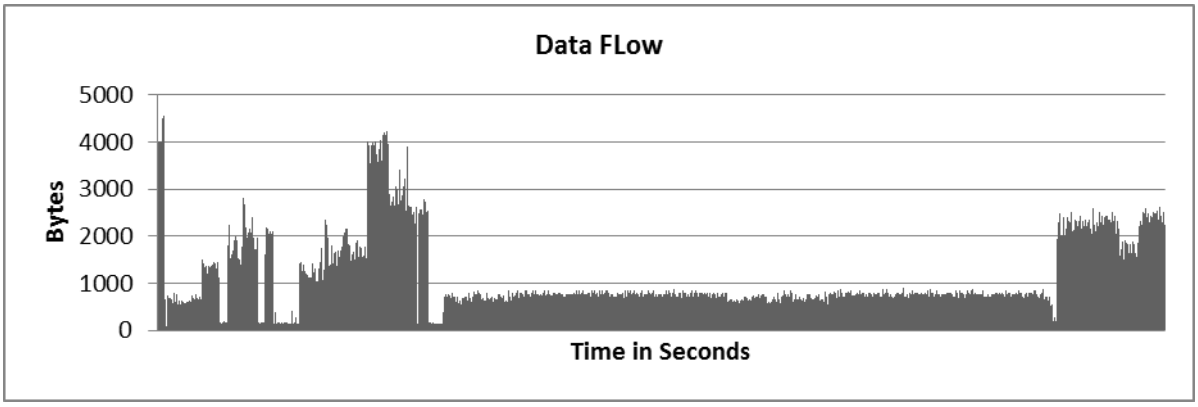

Figura 4.3 – Fluxo de informação que chega por segundo no servidor de pequeno porte.

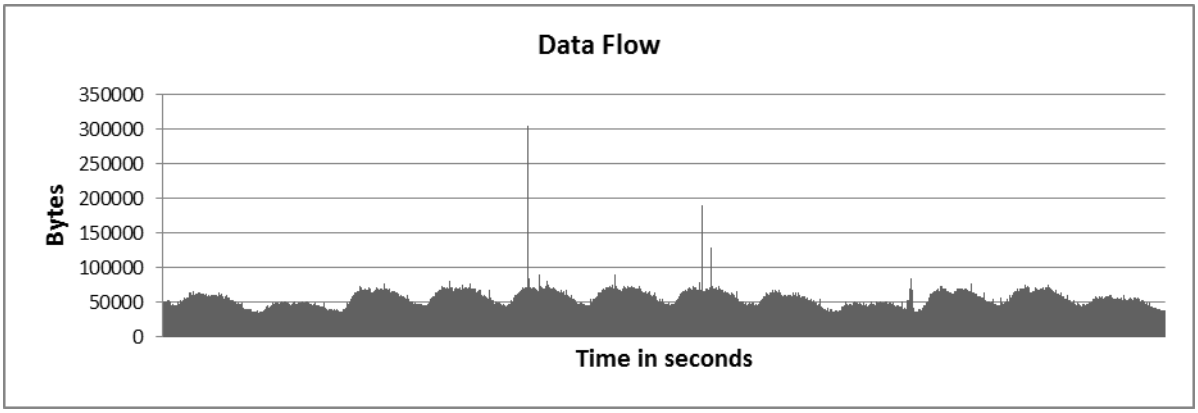

Figura 4.4 – Fluxo de informação que chega por segundo no servidor de grande porte.

É interessante destacar o volume para cada servidor, assim como os picos que cada um foi sujeitado dentro do período de observação. Estas anuências são explicadas estatisticamente, onde um servidor recebendo informações de frotas pequenas, como o demonstrado na figura 4.3, está sujeito a períodos onde praticamente toda a frota atua em áreas sem cobertura de sinal, ficando ocioso. Assim que a frota encontra conectividade, os dados são enviados de forma abrupta, gerando os picos de processamento facilmente observados nos primeiros instantes da figura 4.3.

Em contrapartida, em um servidor de monitoramento de grandes frotas, como o da figura 4.4, que monitora veículos por todo o território brasileiro, a probabilidade de haver pelo menos uma parte da frota transmitindo é maior, alternando um número de veículos entrando em área de sinal quando outros saem de uma área com cobertura. Este fato é facilmente observado na figura 4.4, que mostra variações de fluxo de uma forma mais uniforme. Conforme o porte do servidor aumenta, nota-se também que a relevância dos picos é proporcional. O gráfico mostra poucos picos, porém de intensidade muito maior do que os encontrados na figura 4.3.

A tabela 4.3 faz uma análise estatística do fluxo de dados para cada servidor. É interessante verificar os valores do Quartil 0.25 e 0.75 para cada caso, que contempla os valores que compõe grande parte dos dados processados ao longo do período observado. Através destes valores da tabela, é possível notar que apesar de um grande servidor estar sujeito a picos mais significativos, fato responsável pela obtenção de um desvio padrão tão grande, se relacionarmos o quartil 0.75 com o 0.25 (Quartil (0.75) / Quartil (0.25)), obtemos a razão de 1,85318 para o pequeno servidor e 1,29365 para o grande servidor, mostrando novamente que o fluxo tende a ser mais constante em grande parte do tempo para grandes frotas.

Tabela 4.3 –Análise estatística do fluxo analisado em servidores de diferentes portes.

Variável	Pequeno Servidor (Bytes)	Grande Servidor (Bytes)
Quartil (0.00)	1	10.000
Quartil (0.25)	722	31.500
Quartil (0.50)	1.064	36.250
Quartil (0.75)	1.338	40.750
Quartil (1.00)	5.254	304.250
Desvio Padrão	480,1128	7,0662e+03
Média	1.079	36.514

4.3 Levantamento de Infraestrutura de Comunicação

Esta seção aborda o cenário de atuação de sistemas embarcados de monitoramento de ativos motomecanizados em áreas rurais. Conforme mencionado anteriormente, o *trace* fornecido pela empresa que atua no agronegócio contém mensagens georeferenciadas, contendo o instante de geração da mensagem. Com esta simples informação, foi possível verificar se esta mensagem foi gerada em uma área coberta por sinal ou não.

A seção 2.2.2 abordou a tecnologia fornecida pelas empresas de telefonia celular (GPRS), que segundo as empresas do ramo, são as mais utilizadas para

monitoramento de frotas devido ao baixo custo tanto para aquisição quanto para os contratos mensais de dados.

Munido destas duas informações e do fato de que frotas operando em áreas sem cobertura de sinal irão armazenar os dados e transmiti-los de forma abrupta e oportunística, gerando rajadas e consequentemente picos de processamento para a aplicação servidora, mostrou-se importante explorar de forma mais aprofundada o cenário rural sob o contexto brasileiro.

4.3.1 Levantamento de Antenas de Telefonia Celular

A Agência Nacional de Telecomunicações (ANATEL) (ANATEL, 2012) é o órgão nacional que herdou, do Ministério das Comunicações, os poderes de outorga, regulamentação e fiscalização para a infraestrutura de telecomunicação para o território brasileiro, ou seja, toda e qualquer antena para fins de telecomunicação devem ser registradas e aprovadas por este órgão. É politica deste órgão fornecer informações sobre as antenas de telefonia celular, tais como posição espacial (fornece a latitude e longitude de cada antena), operadora, município, data da licença, entre outras informações.

Através do site da Anatel, é possível adquirir listas das antenas de acordo com o estado brasileiro desejado. Estas listas podem ser obtidas em formato XLS (planilha), possibilitando uma fácil manipulação das informações e obtenção das coordenadas de cada antena.

Este projeto não aborda as questões políticas referentes às operadoras de telefonia celular existentes no país pelo seguinte fato: a impossibilidade de transmitir informações pela falta de rede da operadora é facilmente contornada por modens que suportam mais de um SIM-Card (existem modens com mais de 4 slots), ou seja, o mesmo modem pode verificar para qual operadora irá transmitir, mitigando falta de conectividade por questões políticas.

4.3.2 Área de Cobertura de uma Antena de Celular

Como mostrado na seção 2.2.2, o envio de dados utilizando o protocolo IP é possível a partir da chama tecnologia 2.5G (não é a terceira geração efetivamente). Utilizando o tutorial divulgado pela empresa Teleco (TELECO, 2012), escrito por

Fábio Cunha Oliveira (OLIVEIRA, 2008), através da fórmula de Okumura-Hata COST 231 é possível encontrar a distância máxima entre TX-RX (transmissão e recepção), obtendo uma distância média de alcance para as antenas de acordo com a tecnologia de transmissão.

Para determinar este alcance, é calculado o link budget. Link Budget é o cálculo final de potência por todo o percurso entre transmissão e recepção (TX e RX), levando em consideração: potência de transmissão, as diversas perdas em equipamentos (da estação base e do usuário), ganhos de antena, de amplificadores, e efeitos da propagação. Esse cálculo tem o objetivo de determinar a perda máxima permitida em percurso, determinada Maximum Allowed Path Loss (MAPL), que permite avaliar o alcance efetivo de uma antena celular.

De uma forma geral, a perda máxima em percurso (TX-RF) pode ser expressa pela equação utilizada por Fábio Cunha Oliveira (OLIVEIRA, 2008) :

MAPL = Pot_TX – Perdas_TX – Perdas_RX – Sens_RX – Perdas_Propag + G_Ant_ERB

Na qual:

- Pot_TX: Potência de Transmissão;
- Perdas_TX: Perdas na Transmissão;
- Perdas_RX: Perdas na Recepção;
- Sens_RX: Sensibilidade de Recepção;
- Perdas_Propag: Perdas de Propagação;
- G_Ant_ERB: Ganho da Antena da ERB.

Tanto no padrão GSM quanto no UMTS, o sentido de *uplink* é o fator limitante para o estabelecimento de comunicação entre o aparelho e o equipamento de RF, ou seja, se o *uplink* não pode ser estabelecido o sistema embarcado não consegue transmitir informações.

Cálculo do GSM 900 *Uplink*:

A tabela 4.4 apresenta o *link budget* para *uplink* no sistema GSM 900 MHz.

Tabela 4.4 –Link Budget GSM 900 em Uplink (OLIVEIRA, 2008).

Sentido	Descrição	Valor	Unidade
TX	Potência máxima de transmissão	30	dBm
TX	Atenuações por cabo em TX	0	dB
TX	Atenuações por conectores em TX	0	dB
TX	Ganho da antena em TX	0	dB
TX	Total de perdas e ganhos em TX	0	dB
TX	ERP máximo do aparelho	30	dBm
TX	Temperatura	290	K
TX	Densidade de ruído térmico (*background noise*)	-174	dBm/Hz
TX	Taxa de informação	270	Kbit/s
TX	Taxa de informação [dB] (10 x log(B))	54,4	dB
RX	Figura de ruído do receptor	7	dB
	Carga relativa ao tráfego (50% para serviços de voz)	0,5	percentual (%)
	Margem de interferência (ou da carga)	-3	dB
RX	Relação sinal-ruído requerido (Eb/No)	8	dB
RX	Sensibilidade da BTS	-107,6	dBm
RX	Atenuações por cabos e conectores em RX	-3	dB
RX	Ganho da antena em RX	15	dBi
RX	Total de perdas e ganhos em RX	12	dB
RX	Potência recebida pela BTS	-119,6	dBm
RX	*Cell edge probability* (ε)	0,9	percentual (%)
	Desvio padrão (σ)	7	dB
	Desvanecimento Log-Normal	-9	dB
TX	Ganho de diversidade	0	dB
TX	Perdas de penetração (edificações/	-10	dB

	veicular)		
TX	Perda corporal (relativa à cabeça)	-3	dB
TX	Total dos componentes de propagação	-22	dB
	Perda máxima permitida em percurso (MAPL)	**127,6**	**dB**

Cálculo do UMTS 2100 em *Uplink*

A tabela 4.5 apresenta o *link budget* em *uplink* no padrão UMTS em 2100 MHz.

Tabela 4.5 –*Link Budget* UMTS 2100 em *Uplink* (OLIVEIRA, 2008).

Sentido	Descrição	Valor	Unidade
TX	Máxima potência de transmissão	21	dBm
TX	Perdas físicas em TX	0	dB
TX	Ganho da antena em TX	0	dBi
TX	ERP máximo do aparelho	21	dBm
TX	Temperatura	290	K
TX	Densidade de ruído térmico	-174	dBm/Hz
TX	Taxa de informação (AMR)	12,2	kbit/s
TX	Taxa de informação [dB] (10 x Log(B))	40,9	dB
TX	Figura de ruído do receptor	7	dB
TX	Carga relativa ao tráfego (50% para serviço de voz)	0,5	percentual (%)
TX	Margem de interferência (ou da carga)	-3	dB
TX	Relação sinal-ruído requerido (Eb/Nt)	7,2	dB
TX	Sensibilidade do Node B	-115,9	dBm
RX	Ganho da antena em RX	15	dBi
RX	Perdas nos cabos *feeder*	-3	dBm/100m
RX	Comprimento do cabo *feeder*	60	m
RX	Perdas nos conectores e *Jumper*	-0,6	dB
RX	Total das perdas e ganhos em RX	13,5	dB
RX	*Cell edge probability* (ε)	0,9	percentual

			(%)
	Desvio padrão (σ)	8	dB
	Desvanecimento Log-Normal	-10,3	dB
TX	Ganho de *Handover* (*softhandover*)	4,1	dB
TX	Ganho de diversidade	0	dB
TX	Atenuação por penetração (edificações/ veicular)	-20	dB
TX	Atenuação corporal (nível da cabeça)	-3	dB
TX	Total dos componentes de propagação	-29,2	dB
	Perda máxima permitida em percurso (MAPL)	**120,3**	**dB**

Cálculo da Cobertura

Utilizando a fórmula de Okumura-Hata COST 231, é possível calcular o alcance máximo da cobertura de uma célula através da perda máxima permitida. A fórmula matemática é apresentada a seguir na tabela 4.6.

Tabela 4.6 – Fórmula de Okumura-Hata COST 231 (OLIVEIRA, 2008).

Modelo Okumura-Hata COST 231
L = 46,3+33,9*LOG(f)-13,82*LOG(ht)-Ch+ (44,9-6,55*LOG(ht))*LOG(d*0,001)+C
Ch = (1,1 log f –0,7) hr – (1,56 log f –0,8)

O alcance máximo da cobertura das células GSM é mostrado na tabela 4.7, utilizando o fator C (em dB) igual a 0, considerando o caso de cidades médias e áreas suburbanas.

Tabela 4.7 – Parâmetros de entrada e cálculo da distância máxima do *link* GSM. (OLIVEIRA, 2008).

Parâmetro	**Valor**
Frequência de Transmissão	900 MHz
Altura Efetiva da Estação Base	50 metros
Altura da Antena do Aparelho Celular	1,6 metros

MAPL Mínimo	127,6 dB
Distância de Atuação da BTS para o Mínimo MAPL	1399,7 metros

A tabela 4.8 mostra o cálculo do alcance máximo da cobertura de uma célula UMTS, considerando os mesmos valores da altura da estação base e altura do aparelho:

Tabela 4.8 – Parâmetros de entrada e cálculo da distância máxima do *link* UMTS. (OLIVEIRA, 2008).

Parâmetro	Valor
Frequência de Transmissão	2100 MHz
MAPL Mínimo	120,3 dB
Distância de Atuação do Node B para o Mínimo MAPL	364,6 metros

É importante deixar claro que este valor calculado representa apenas a cobertura máxima de uma célula e não estão sendo considerados fatores que reduzem este alcance, tais como relevo, obstáculos ou até mesmo a rede celular em áreas urbanas, que são limitados pela capacidade de suportar a demanda de tráfego. No caso do WCDMA, o alcance pode ser reduzido pela interferência resultante, podendo ser utilizado artifícios como o *downtilt* elétrico e mecânico das antenas e o controle de potência, para equilibrar estes parâmetros, e garantir a qualidade de serviço.

4.3.3 Mapa de Cobertura de sinal GPRS

Após os cálculos de área máxima para cada antena, utilizamos o valor encontrado para a tecnologia GPRS, que opera sobre a GSM, ou seja, um alcance de aproximadamente 1400 metros da antena até o sistema embarcado.

Com as informações obtidas nas seções 4.3.1 e 4.3.2, foi possível criar um mapa espacial, utilizando o complemento PostGIS do banco de dados Postgree (POSTGIS, 2012), contendo todas as antenas e suas respectivas áreas de cobertura para todo o território brasileiro. A figura 4.5 mostra a localização das antenas existentes no estado de São Paulo, a figura 4.6 as antenas e suas respectivas áreas de cobertura para a cidade de São Carlos – SP e, finalmente, a figura 4.7, a mais

interessante para este projeto, evidencia a falta de infraestrutura de comunicação para o cenário rural.

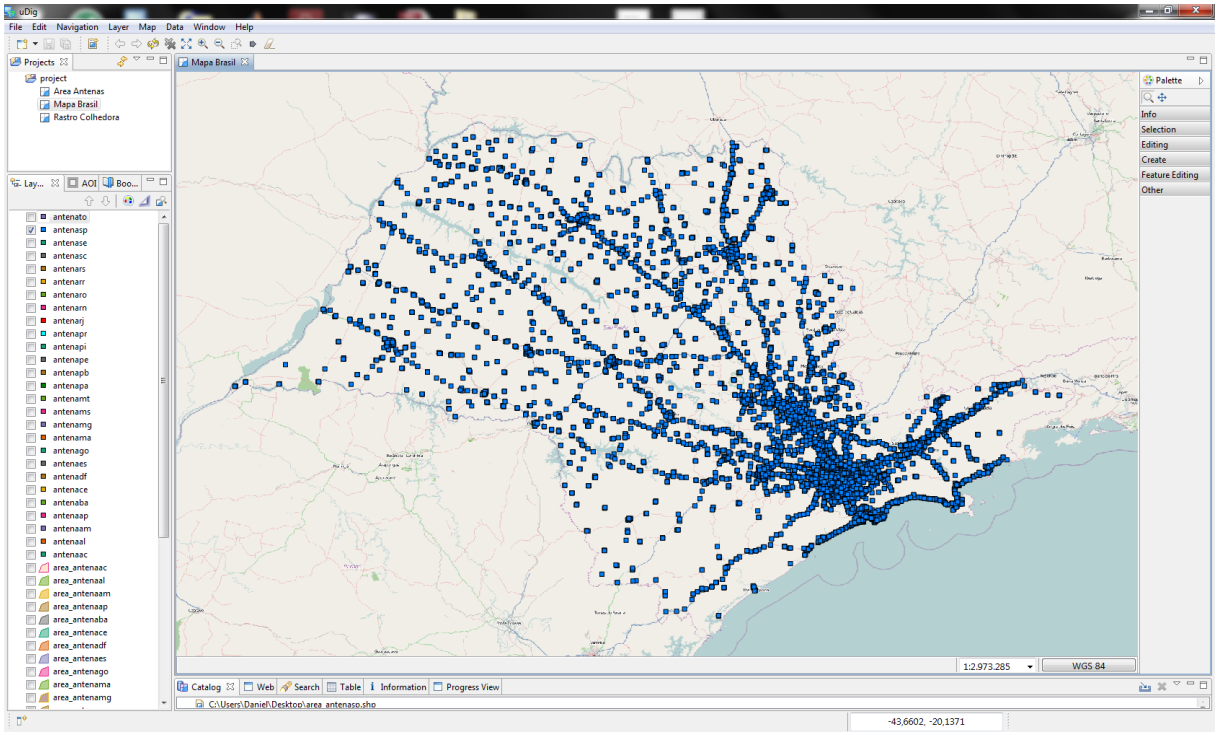

Figura 4.5 – Mapa das antenas de telefonia celular no estado de São Paulo.

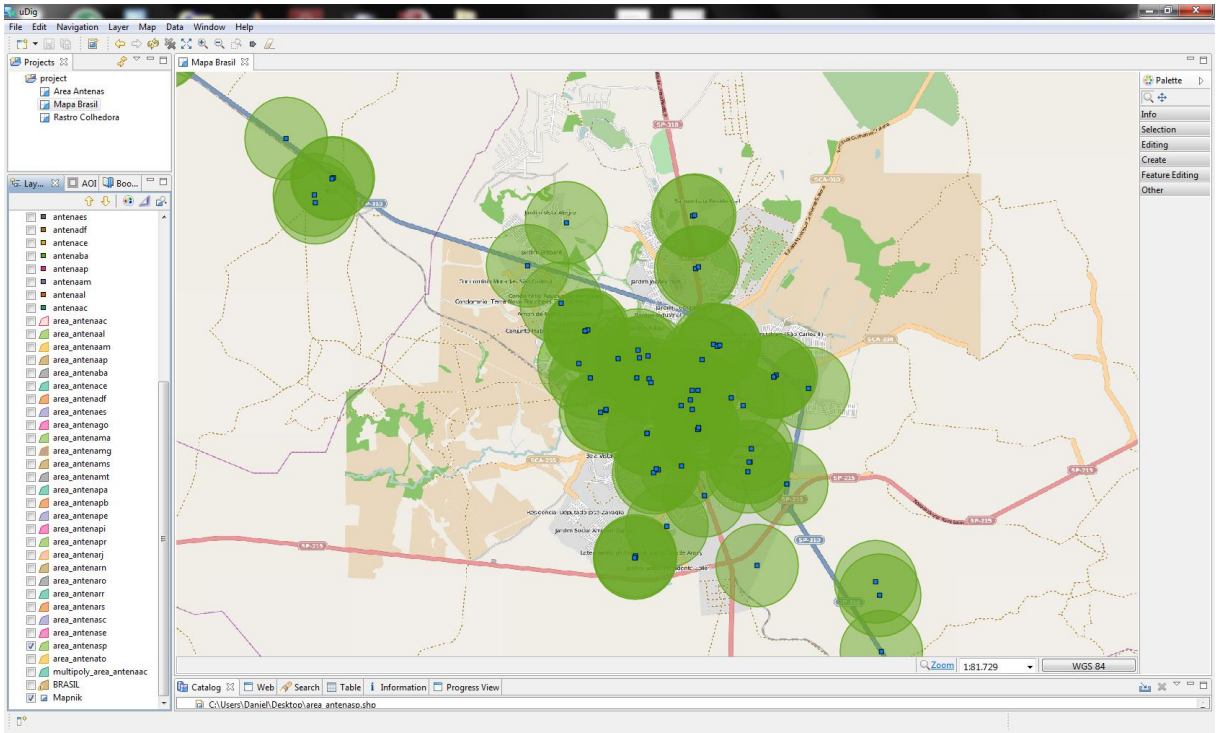

Figura 4.6 – Mapa das áreas de cobertura das antenas de telefonia celular presentes na cidade de São Carlos -SP.

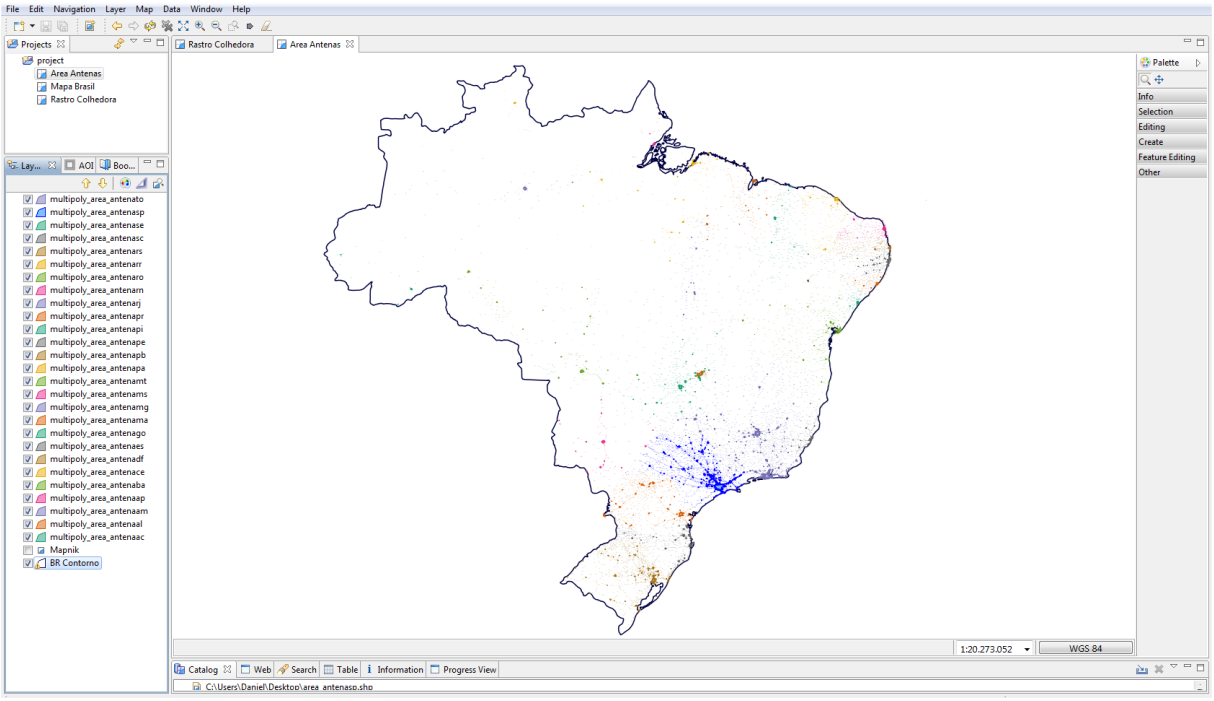

Figura 4.7 – Mapa de cobertura GPRS do território brasileiro.

As empresas de telefonia celular devem cumprir uma determinação de prover sinal para localidades. Toda localidade (ou seja, as residências não podem ter mais de 50 metros de distância uma das outras) com mais de 300 habitantes (EMBRATEL, 2012) deve possuir telefonia. Isto exclui a obrigação de cobrir 100% do território nacional, fazendo com que a empresa de telefonia invista em áreas com potencial de lucro. Para melhor atender seus clientes, é comum também a instalação de antenas às margens de rodovias, conforme a figura 4.5 sugere.

Em posse desta base de dados, contendo a área de cobertura de cada estado, foi possível construir a tabela 4.9, contendo informações sobre os estados brasileiros (IBGE, 2012) e suas respectivas áreas de cobertura.

Tabela 4.9 – Relação entre território e área de cobertura de sinal GPRS.

Estado	Território (Km²)	Área de Sinal (Km²)	Porcentagem de Área Coberta (%)
AC	152.581,388	282,871853	0,001853908
AL	27.767,661	1.189,463768	0,042836297
AP	142.814,585	265,897134	0,001861835
AM	1.570.745,68	823,285326	0,000524137
BA	564.692,669	4.582,270278	0,008114627
CE	148.825,602	2.597,065660	0,017450396
DF	5.801,937	1.094,629974	0,188666298
ES	46.077,519	1.969,014150	0,042732643
GO	340.086,698	3.549,164460	0,010436058

MA	331.983,293	1.958,240923	0,005898613
MT	903.357,908	1.843,154658	0,002040337
MS	357.124,962	1.355,961124	0,003796881
MG	586.528,293	10.983,724669	0,018726675
PA	1.247.689,515	2.202,807697	0,00176551
PB	56.439,838	1.902,223504	0,033703561
PR	199.314,85	5.229,733599	0,026238555
PE	98.311,616	2.697,861316	0,027441938
PI	251.529,186	1.759,594323	0,006995587
RJ	43.696,054	5.230,736810	0,119707304
RN	52.796,791	1.658,708319	0,03141684
RS	281.748,538	7.098,377223	0,025194016
RO	237.576,167	631,653052	0,002658739
RR	224.298,98	191,535414	0,000853929
SC	95.346,181	3.804,065005	0,039897403
SP	248.209,426	17.228,508983	0,069411179
SE	21.910,348	826,421488	0,037718319
TO	277.620,914	1.116,221134	0,004020667
TOTAL	8.514.876,599	84.073,191844	0,009873683

Estas informações foram cruciais para as simulações realizadas na concepção da arquitetura, pois com estes valores foram parametrizadas as simulações.

4.3.4 Georeferenciamento das mensagens

O processo de análise do *trace* real mostrou a exata proporção entre mensagens geradas em áreas com e sem conectividade, quantificando o problema de armazenamento da mensagem, seguido do evento de transmissão de forma oportunística, resultando nos picos de processamento para a aplicação servidora.

A tabela 4.10 mostra esta proporção para 12.743.145 mensagens geradas por uma colhedora de cana-de-açúcar operando no estado do Mato Grosso no cenário rural. Praticamente todas as mensagens foram geradas efetivamente em áreas sem conectividade (aproximadamente 99,95%).

Tabela 4.10 – Análise espacial do instante de geração da mensagem pelo sistema embarcado.

Mensagem gerada:	Quantidade	Porcentagem (%)
Em área com conectividade	6.544	0.000513531
Em área sem conectividade	12.736.601	0.999486469
Pontos com valor de GPS inválidos	451.061	0.035396364

Outra constatação interessante foi o recebimento de valores inválidos de GPS, incluindo valores iguais à zero (ou seja, todos os valores da *string* no formato NMEA (LANGLEY, 1995) iguais à zero). Este valor não é referente a uma possível perda de precisão, ocasionada por fatores naturais como o clima, e sim sugere que sistemas embarcados munidos de GPS estão também sujeitos a outros tipos de falhas ocasionadas por falta de infraestrutura, que no caso, corresponde a uma possível falta de capacidade da constelação de satélites em suprir a demanda de equipamentos em áreas remotas.

4.4 Estudo da Aplicação Servidora

Outra etapa fundamental para o projeto foi um estudo do comportamento da aplicação servidora, responsável pelo tratamento dos dados oriundos dos sistemas embarcados de monitoramento de ativos motomecanizados, presente no portfólio da mesma empresa que forneceu o *dump* do banco de dados, contendo o *trace* real da frota monitorada.

Além da decodificação dos três tipos de mensagens (descritas na seção 4.2.1), a aplicação servidora é também responsável pelo processamento de interações, tanto do usuário (através da solicitação de um relatório, por exemplo), quanto dos sistemas de monitoramento (atualização de estados da frota de um SIG). Estas interações foram classificadas como “consultas” neste trabalho. Através do monitoramento destas consultas foi possível classificá-las em três classes distintas:

- Consultas Simples: Representam em grande parte as atualizações automáticas dos SIGs em utilização (isto é, em execução e acessando

as informações da aplicação servidora) e algumas requisições de usuários. São requisições que realizam consultas simples no banco de dados (isto é, não utilizam operações de grande custo computacional para o banco de dados, como operações *join* (THEODORIDIS, STEFANAKIS & SELLIS, 1998)) e não exige muito processamento por parte da aplicação servidora;

- Consultas Medianas: Compõem as solicitações realizadas na maioria dos casos por usuários, tais como relatórios de ativos motomecanizados (individuais), consulta de rastro, quando se trata de monitoramento de frotas, entre outras que demandam a recuperação de um volume considerável de informações do banco de dados, seguido de processamento significativo por parte da aplicação servidora (organização visual e tratamento da informação recuperada para a composição de relatórios em formato PDF, por exemplo);
- Consultas Complexas: Novamente, são requisições que em grande parte são realizadas pelos usuários do sistema, contendo o processamento e organização de grandes volumes de informações, tais como fechamento mensal de rendimentos de uma frota, comparativos entre ativos de mesma categoria (discrepância entre o de menor com o de maior rendimento, por exemplo), ou seja, são consultas com grande custo computacional tanto para recuperação da informação quanto para o processamento das mesmas.

Em resumo, o fator decisivo para a criação destas três classes foi o custo computacional, medido através do tempo de resposta para cada caso. Os tempos médios foram aferidos através de alterações no código fonte da aplicação, calculando o tempo de resposta do banco de dados e do processamento da informação conforme o pseudocódigo da figura 4.8:

```
18
19          //Retorna o instante em nanosegundos
20          long instanteInicial = System.nanoTime();
21
22          //Decodifica a mensagem gerada pelo sistema embarcado
23          DecodificaMensagem(msg);
24
25          //Retorna o instante referente ao fim da decodificação
26          long instanteFinal = System.nanoTime();
27          //Calcula o tempo de decodificação
28          long TempoResposta = instanteFinal - instanteInicial;
29
30          //Calcula a média do tempo de decodificação de uma mensagem
31          CalculaMediaTempoResposta(TempoResposta);
32
33          //...
34
```

Figura 4.8 – Representação do método utilizado para obtenção de tempo de resposta para cada classe de consulta

Este procedimento foi inserido em todas as funções responsáveis pela decodificação das mensagens geradas pelos sistemas embarcados e pelas consultas, automáticas ou não. Esta investigação foi realizada no mesmo período citado na seção 4.2.1, resultando na tabela 4.11, que contém o tempo de resposta (tempo de processamento) por parte da aplicação servidora e do banco de dados, separadamente:

Tabela 4.11 – Tempo médio de resposta em segundos

Tipo	Processamento CPU	Processamento BD
Decodificação de Mensagem	$6{,}8452 \times 10^{-5}$	$3{,}234099 \times 10^{-3}$
Consulta Simples	0,00813	0,03461
Consulta Mediana	0,1821	1,132
Consulta Complexa	3,767	55,857

Além dos tempos de resposta, este procedimento também permitiu descobrir as taxas de chegada das consultas para a aplicação servidora para cada sistema de monitoramento (SIG) em uso, conforme mostra a tabela 4.12, onde, novamente, maiores valores de prioridade determinam maior grau de importância, devendo ser atendida primeiramente. Esta tabela também evidencia que praticamente todos os

processos das soluções de monitoramento são automatizados, havendo intervenções dos usuários quando ocorrem não conformidades.

Tabela 4.12 – Taxas aproximadas das chegadas de consultas para aplicação servidora para cada SIG em uso

Tipo de Consulta	Prioridade	Taxa Aproximada (λ)
Simples	1	1 / segundo
Mediana	0	0.1 / segundo
Complexa	0	0.25 / hora

Através destas informações foi possível calibrar as simulações para testar possíveis arquiteturas para a aplicação servidora.

Capítulo 5

SIMULAÇÕES DE DESEMPENHO

Através de simulações, diferentes abordagens de sistemas de filas foram colocadas à prova, com o intuito de obter a que melhor se adequa às características do sistema e de seu cenário de atuação.

Para determinar a melhor estratégia de implementação buscando uma boa relação entre *throughput* e tempo de resposta (JIN & MIN, 2007), o projeto contemplou simulações de diferentes arquiteturas para avaliar suas características (NICOLA & ZABURNENKO, 2006).

Utilizando a ferramenta Java Modelling Tools (JMT) (SERAZZI, 2008), foi possível explorar em tempo hábil o comportamento de uma solução com arquitetura convencional com a distribuída, a fim de quantificar o ganho de *throughput* e o tempo de resposta para o processamento das mensagens de controle, as quais devem ser processadas no menor intervalo de tempo possível (a seção 2.1.1 mostra que mensagens desta classe podem inclusive prevenir acidentes).

Foram realizadas simulações prevendo o comportamento médio da aplicação servidora para o monitoramento de frotas com e sem descarregamento da informação abrupta, ou seja, com e sem rajadas (LUO & WILLIAMSON, 2008). Esta abordagem mostrou-se interessante, pois o intuito do projeto é justamente prover uma solução distribuída de baixo custo que tenha um comportamento médio aceitável, ou seja, que seja capaz de processar as mensagens de controle rapidamente e as demais solicitações com tempo suficiente para que as tomadas de decisões, pertinentes a cada tipo de informação ou consulta, ocorram também em tempo hábil para cada área de interesse (por exemplo, um relatório mensal de

desempenho pode demorar minutos, enquanto que consultas de atualizações automáticas dos SIGs não devem ultrapassar um segundo).

5.1 Ferramenta Java Modelling Tools (JMT)

Antes da proposta e resultados das simulações realizadas, esta seção apresenta a poderosa ferramenta Java Modelling Tools. Desenvolvida pelo Departamento de Eletrônica e Informação da Universidade Politécnica de Milão, contém seis módulos interessantes, desenvolvidos em Java, provendo um *framework* completo para medição de *performance*, ajuste do sistema, planejamento de capacidade e estudos de caracterização da carga de trabalho.

Cada módulo desempenha uma função específica, permitindo uma análise exata ou aproximada tanto do desempenho quanto do fluxo a ser processado:

1. JSIMgraph - Simulador com interface gráfica de modelos de redes de filas;
2. JSIMwiz – Simulador de modelos de redes de filas baseado em *interface* “Wizard”, ou seja, passo a passo;
3. JMVA – Análise de Valor Médio para modelos de redes de filas;
4. JABA - Análise assintótica de modelos de rede de filas;
5. JAWAT – Realiza a análise de carga de arquivos de logs;
6. JMCH – Simulador de cadeias de Markov (Ferramenta didática).

Outra vantagem da ferramenta é sua fácil utilização, que permite a criação e execução de simulações de forma rápida e simples. Para ilustrar esta facilidade, vamos utilizar de exemplo o sistema descrito nos capítulos anteriores, que possui três classes de mensagens e três classes de consultas, com prioridades distintas, conforme a tabela 5.1, onde a prioridade mais alta é definida pelo maior valor:

Tabela 5.1 – Tabela de classes e prioridades para o sistema

Classe	Prioridade
Mensagem de Controle	4
Mensagem de Produtividade	3
Mensagem Operacional	2
Consulta Simples	1
Consulta Mediana	0
Consulta Complexa	0

Ao invés de criar cinco filas distintas para tratar de forma correta as diferentes prioridades, esta simulação é facilmente implementada definindo no *framework* as classes e suas prioridades utilizando o módulo gráfico JSIMgraph. O *framework* trata as prioridades exatamente como fora demonstrado na tabela 5.1, quanto maior o valor, maior será sua prioridade. Veja a figura 5.1 como a definição das classes e suas prioridades são realizadas:

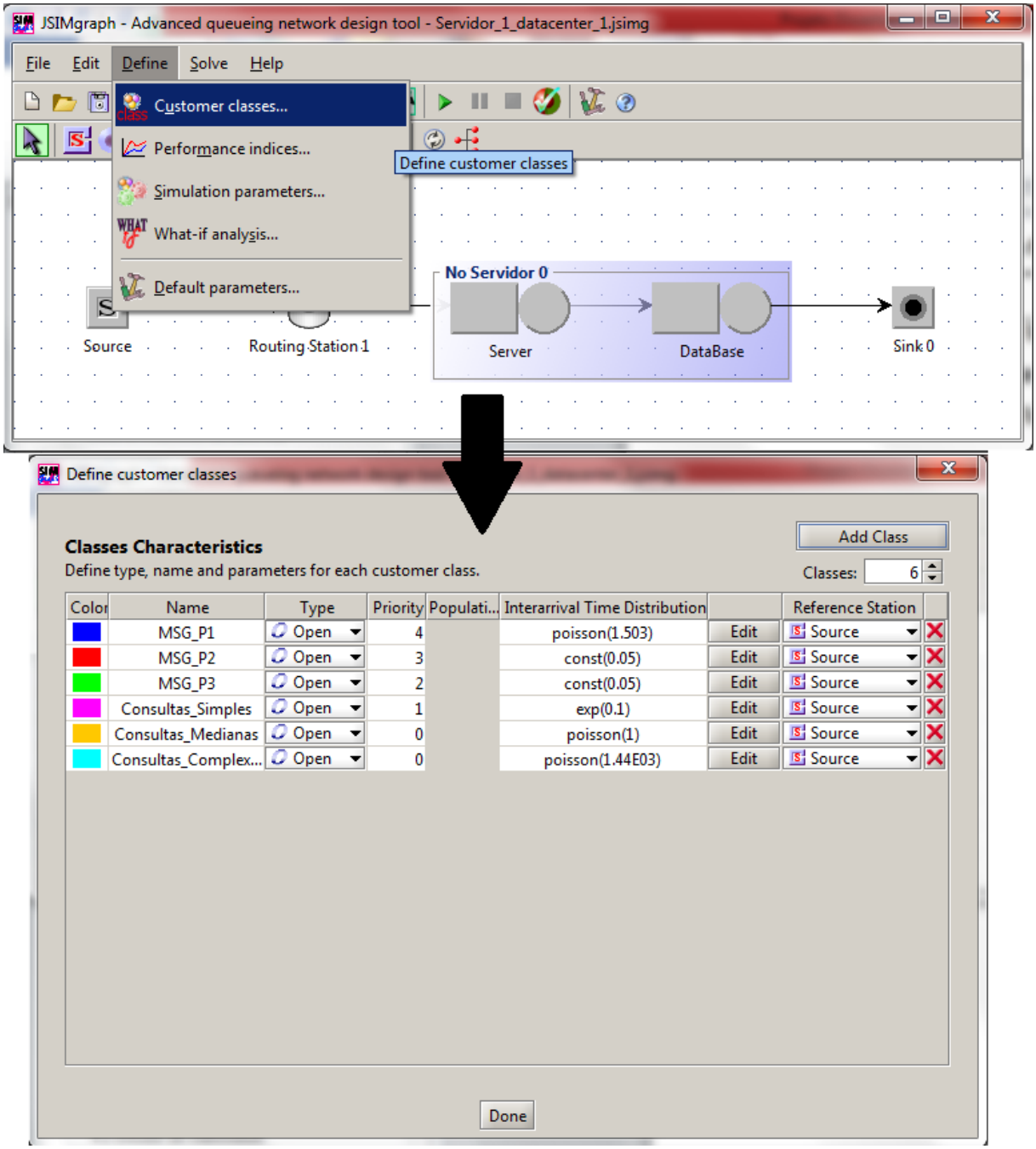

Figura 5.1 – Definindo Classes e Prioridades no *Framework* JMT

O mesmo processo de definição de classes e prioridades define a taxa de chegada para cada classe, permitindo a utilização de diferentes distribuições estatísticas. A coluna “Interarrival Time Distribution” mostra a utilização de distribuições diferentes para cada tipo de classe.

Em seguida, basta configurar cada entidade servidora para utilizar o critério de prioridades para processar as diferentes classes, junto com seu tempo de serviço médio, conforme mostra a figura 5.2:

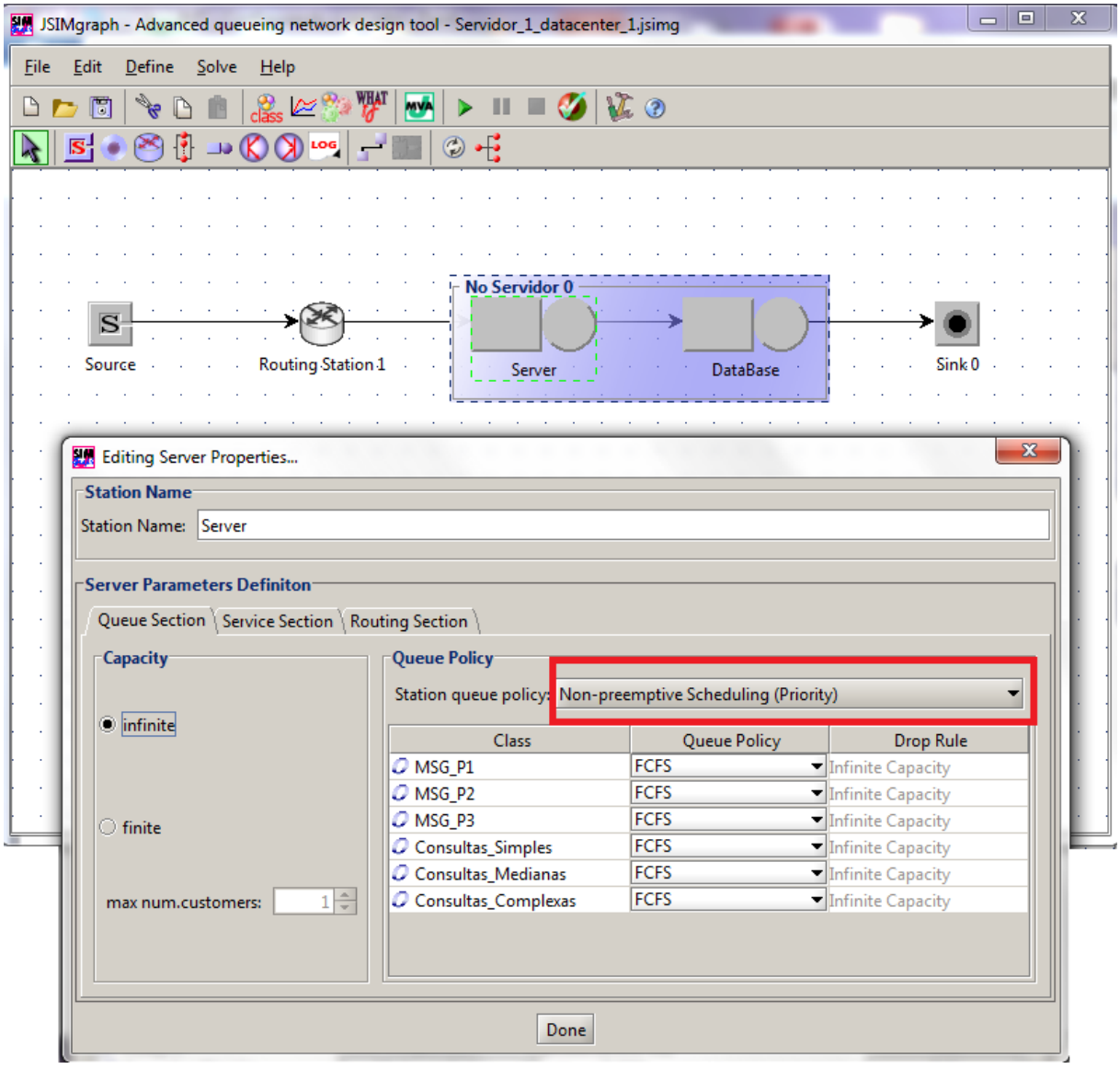

Figura 5.2 – Configurando o servidor para utilizar a política de prioridades

A figura mostra em destaque a opção de utilizar as prioridades determinadas na etapa anterior (ver figura 5.1). A aba “Service Section” permite determinar o tempo de processamento para cada classe, de forma análoga a definição dos tempos de chegadas.

Finalmente, a figura 5.3 mostra uma execução gráfica da simulação. É possível calcular vários parâmetros para todas as classes que definimos, tais como tempo de resposta, tempo de fila, utilização de cada estrutura definida, *throughput*, entre outros.

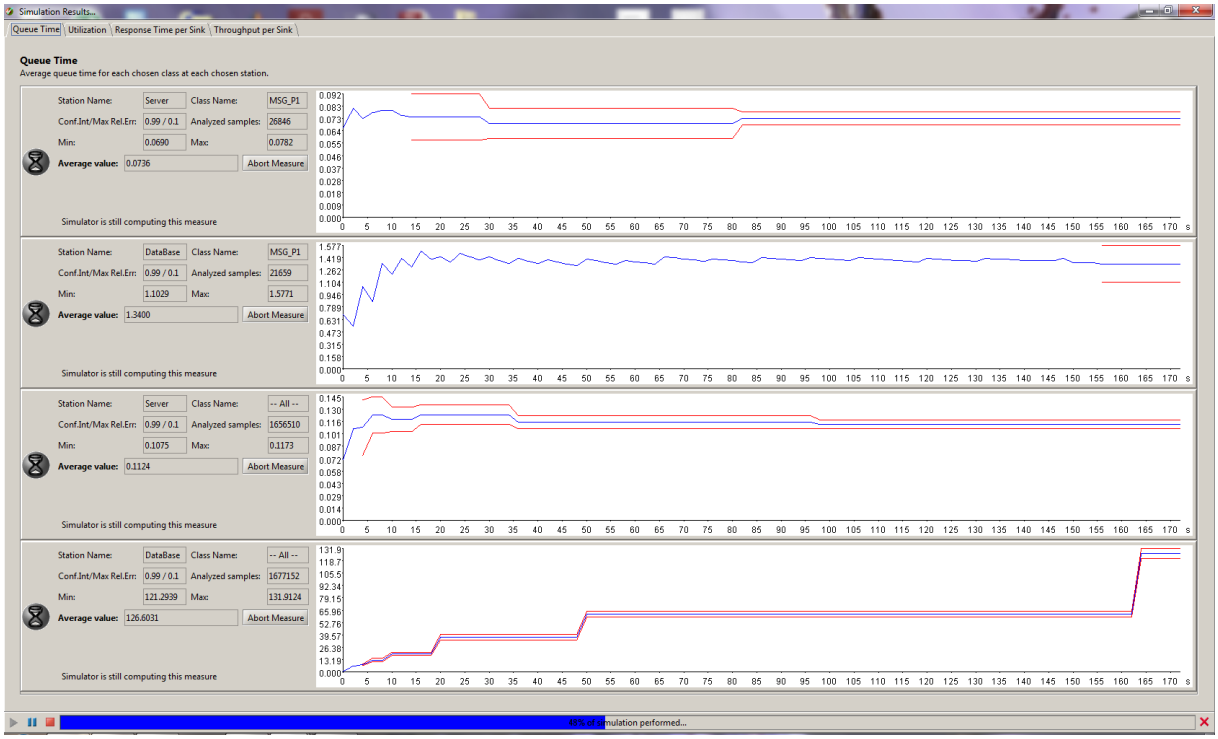

Figura 5.3 – Executando uma simulação utilizando o framework JMT

Para maiores informações, o *framework* fornece um manual, tutoriais e os trabalhos publicados no site http://jmt.sourceforge.net/Documentation.html, onde é possível fazer o download de várias versões do programa também.

5.2 Arquitetura Convencional de Controle

Para mensurar as vantagens e ganhos com diferentes abordagens de arquitetura para a aplicação servidora, esta seção determina uma simulação utilizando o JMT parametrizado com os tempos de serviços do servidor que processou o *trace* real estudado (DIJK, 1990), presente no cliente (tempos presentes na tabela 4.11).

A aplicação servidora presente recebe todas as mensagens e solicitações e armazena em uma fila simples e o processamento é realizado por uma única CPU, ou seja, a aplicação servidora atua em conjunto com o banco de dados em um único computador, conforme mostrado na figura 5.4:

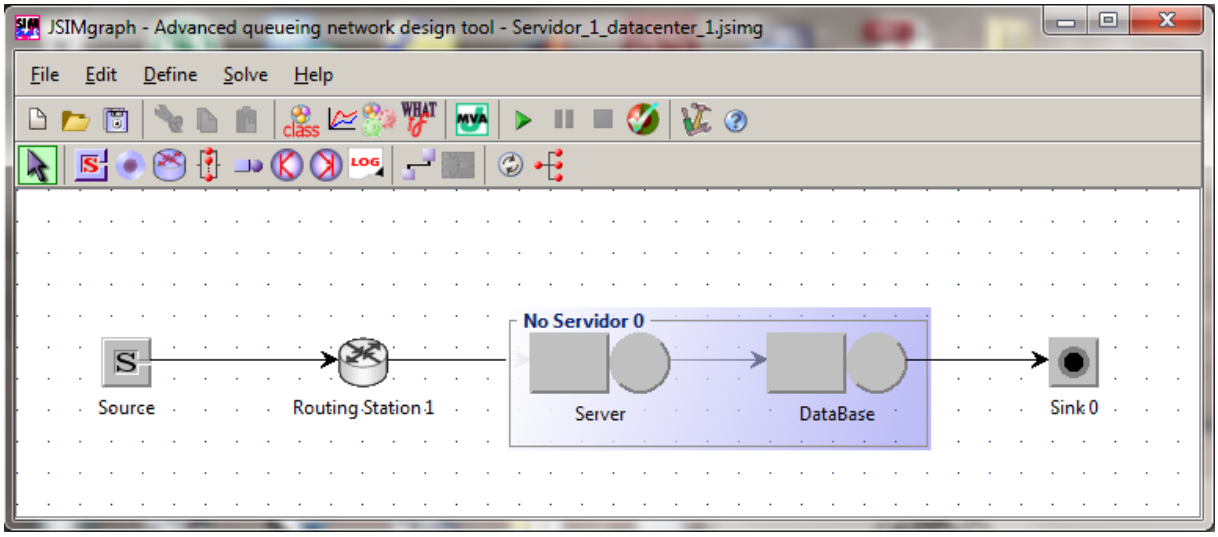

Figura 5.4 – Diagrama da arquitetura convencional da Aplicação Servidora

Calibrando esta simulação com os parâmetros das tabelas 4.2, 4.11 e 4.12, foi possível obter os valores referentes ao monitoramento de uma frota contendo 20 veículos monitorados e 10 usuários utilizando simultaneamente programas SIG. Os resultados mostram que a aplicação servidora não consegue tratar todas as requisições, fazendo com que o tempo de resposta cresça indefinidamente, conforme mostra a figura 5.5.

Esta simulação também evidencia que o grande gargalo é o banco de dados. Todo processamento é seguido de uma consulta ou inserção no banco, somando a discrepância de tempo de processamento de uma consulta complexa com as demais, fica evidente que quando esta consulta entra no sistema, o nó fica ocupado por um grande intervalo (o servidor não é preemptivo), prejudicando todo o resto do processamento.

A simulação com a arquitetura de controle forneceu os valores da tabela 5.2, contendo os tempos de fila para cada classe definida na tabela 5.1 e os valores de utilização e *throughput*, referentes à CPU, banco de dados e para o sistema.

Tabela 5.2 – Resultados da Simulação da Arquitetura de Controle

Tipo	Tempo de Fila (CPU; BD)	Utilização	Tempo de Resposta	Throughput
Mensagem de Controle	0,0772; 1,3389	-	-	-
Mensagem de Produtividade	0,1185; 1,8635	-	-	-
Mensagem Operacional	0,1160; 1,9853	-	-	-
Consulta Simples	0,1148; 2,3685	-	-	-
Consulta Mediana	0,1602; 3,494E4	-	-	-
Consulta Complexa	0,0636; 4,012E4	-	-	-
CPU	-	0,1951	-	-
Banco de Dados	-	1,0000	-	-
Sistema	-	-	>224,2509	41,4792

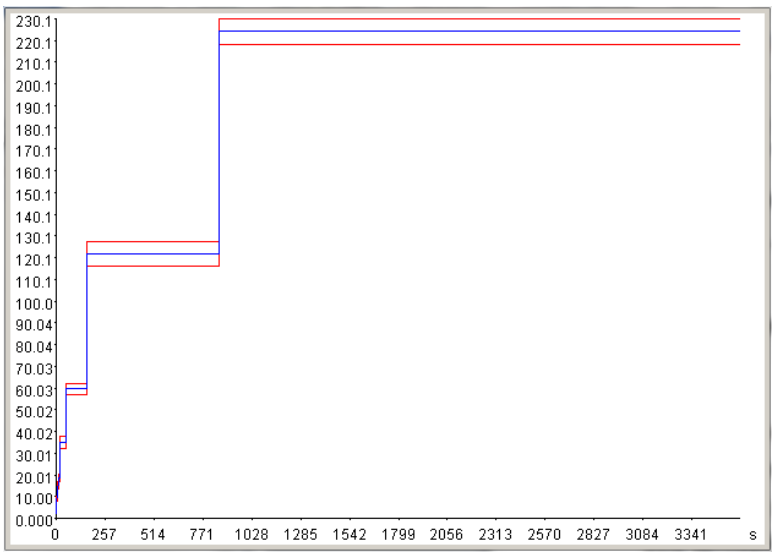

Figura 5.5 – Tempo de resposta para a aplicação servidora convencional

5.3 Arquitetura Convencional com *Speed up*

Ainda estudando uma arquitetura convencional para esta aplicação servidora, a mesma simulação foi executada provendo um *speed up* de 500% (ou seja, cinco vezes mais rápida) para o banco de dados, identificado previamente como o gargalo para esta aplicação.

O intuito desta abordagem é efetivamente prover um comparativo entre a arquitetura convencional com a distribuída, a fim de atender as premissas descritas anteriormente embasando as decisões com dados numéricos. A simulação segue o mesmo modelo da figura 5.4, alterando somente os valores dos tempos de serviços para cada classe na entidade *DataBase* (Banco de Dados).

A tabela 5.3 mostra os valores obtidos na simulação com a mesma taxa de chegada utilizada para a arquitetura de controle.

Tabela 5.3 – Resultados da Simulação da Arquitetura Convencional com *speed up* de 500%

Tipo	Tempo de Fila (CPU; BD)	Utilização	Tempo de Resposta	Throughput
Mensagem de Controle	0,0765; 0,1838	-	-	-
Mensagem de Produtividade	0,1169; 0,2349	-	-	-
Mensagem Operacional	0,1172; 0,2443	-	-	-
Consulta Simples	0,1154; 0,2378	-	-	-
Consulta Mediana	0,1580; 0,3333	-	-	-
Consulta Complexa	0,1687; 0,4532	-	-	-
CPU	-	0,1934	-	-

Banco de Dados	-	0,2658	-	-
Sistema	-	-	0,3691	41,7649

Esta tabela traz resultados interessantes. As tabelas 4.2 e 4.12 mostram que o sistema deve processar aproximadamente 41 requisições por segundo, que torna coerente o *throughput* apresentado pela simulação. Contudo, com o *speed up* do gargalo da aplicação, o sistema consegue tratar todas as requisições com um tempo de resposta baixo.

Para um *speed up* de 500% o sistema passa a atender todas as requisições de forma satisfatória. Contudo, para prover um hardware cinco vezes mais rápido que um servidor convencional requer investimentos na ordem de centenas de milhares a milhões de reais. Além disso, obter o hardware cinco vezes mais rápido não quer dizer que ele futuramente atenda a premissa de escalabilidade, tornando necessária sua total substituição do mesmo quando atingir algum limite de *speed up*, contrastando com as premissas do projeto.

5.4 Arquitetura Distribuída

Esta seção apresenta um teste muito simples. Ao invés de prover um *speed up* de 500% para o hardware em que a aplicação servidora é hospedada, dispor de cinco servidores idênticos ao hardware inicial e distribuir as requisições de forma balanceada (KAZEM, AHMED & SHIRMOHAMMADI, 2007), conforme mostra a figura 5.5:

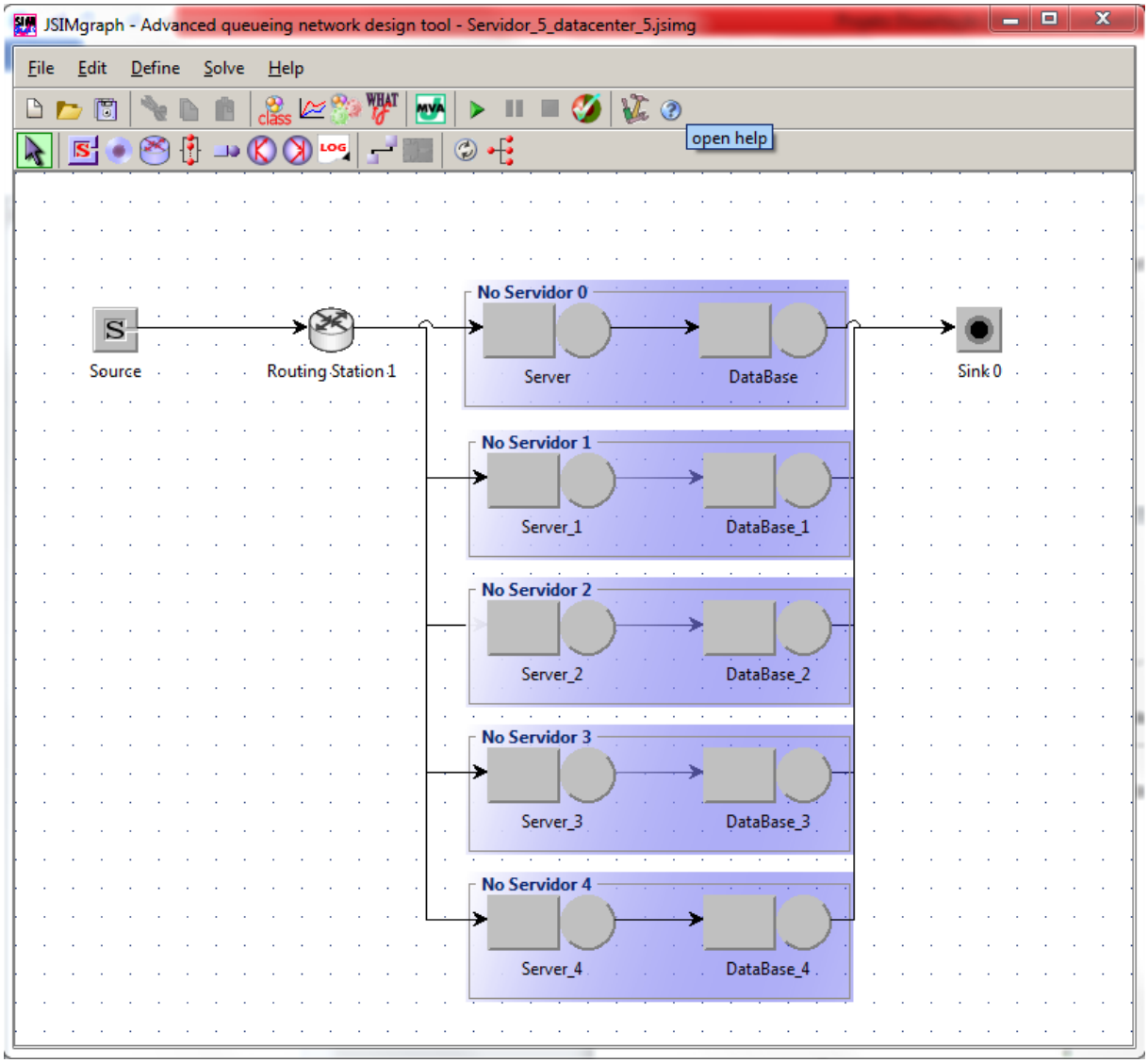

Figura 5.6 – Diagrama da arquitetura distribuída da Aplicação Servidora com cinco nós

A simulação faz-se necessária para mensurar os valores médios desta abordagem distribuída, a fim de comprovar sobre outra perspectiva que o estudo de viabilidade da seção 4.1 fez. Como sabemos que o *speed up* é caro e não necessariamente provê escalabilidade futura, o teste da solução distribuída deverá mostrar seu verdadeiro potencial.

Conforme o esperado, a tabela 5.4 mostra que a abordagem distribuída tem grande potencial para atender as premissas do projeto, apresentando resultados semelhantes ao visto na tabela 5.3:

Tabela 5.4 – Resultados da Simulação da Arquitetura Distribuída com cinco nós

Tipo	Tempo de Fila (CPU; BD)	Utilização	Tempo de Resposta	Throughput
Mensagem de Controle	0,0408; 0,5681	-	-	-
Mensagem de Produtividade	0,0519; 0,7240	-	-	-
Mensagem Operacional	0,0521; 0,7737	-	-	-
Consulta Simples	0,0527; 0,7876	-	-	-
Consulta Mediana	0,0623; 1,6139	-	-	-
Consulta Complexa	0,0719; 1,8265	-	-	-
CPU	-	0,0388	-	-
Banco de Dados	-	0,2656	-	-
Sistema	-	-	1,8717	41,7734

Um fato interessante que ocorre nesta arquitetura é que apesar do tempo de resposta ser maior, o *throughput* também é maior, o que vai contra o conceito de que o *throughput* é inversamente proporcional ao tempo de resposta. Isto ocorre justamente porque o valor é a média entre todas as entidades servidoras (CPU e banco de dados). Sempre que uma consulta complexa é requisitada, esta ocupa um nó de processamento por um longo intervalo de tempo, elevando drasticamente seu valor médio de tempo de resposta. Em contrapartida, como a taxa de chegadas desta consulta complexa é baixa, enquanto um nó está alocado, os demais estão livres para processar as outras requisições.

5.5 Arquitetura Convencional, com *Speed up* e Rajadas

Apesar das seções anteriores trazerem resultados suficientes para uma tomada de decisão do projeto com base em suas premissas, o grande responsável pelos picos de processamento são as rajadas de informações provenientes dos sistemas embarcados, que são transmitidos de forma oportunística.

Para verificar como a arquitetura convencional comporta-se com o processamento de mensagens simulando estes picos, esta simulação foi baseada de acordo com *trace* da figura 4.3 (BANGA & DRUSCHEL, 1999), que representa bem estes picos. A mesma arquitetura da seção 5.3 foi calibrada com a taxa de chegadas em modo *burst* que o JMT fornece, inserindo os valores do quartil 0,25 e 0,25 da tabela 4.3 (722 e 1.338) para a distribuição de Poisson mínima e máxima destes *bursts*.

A tabela 5.5 mostra os resultados obtidos para esta simulação:

Tabela 5.5 – Resultados da Simulação da Arquitetura Convencional com *speed up* de 500%, submetida a rajadas

Tipo	Tempo de Fila	Utilização	Tempo de Resposta	Throughput
Mensagem de Controle	0,1545; 0,2296	-	-	-
Mensagem de Produtividade	0,1800; 0,4781	-	-	-
Mensagem Operacional	0,1813; 0,5024	-	-	-
Consulta Simples	0,1823; 1,4754	-	-	-
Consulta Mediana	0,1868; 2,1180	-	-	-
Consulta Complexa	0,2289; 3,0591	-	-	-
CPU	-	0,2324	-	-

Banco de Dados	-	0,6462	-	-
Sistema	-	-	1,0576	637,8965

O desempenho da arquitetura convencional foi afetado quando submetido a rajadas, uma vez que quando a demanda aumenta, mesmo com o banco de dados 500% mais rápido, não consegue suprir a demanda, fazendo com que o *throughput* seja inferior, mesmo com o tempo de resposta médio bom.

Este evento é facilmente notado pelo tempo de espera na fila das mensagens, que crescem significativamente.

5.6 Arquitetura Distribuída com Rajadas

Esta simulação é a mais significativa para a validação da concepção do projeto até este ponto, onde apresenta os valores pertinentes para comparar de forma analítica as vantagens de adotar a distribuição da aplicação servidora.

Calibrando as rajadas com os mesmos valores da seção 5.5, a simulação do modelo descrito na seção 5.4 foi submetida a testes, resultando na tabela 5.6:

Tabela 5.6 – Resultados da Simulação da Arquitetura Distribuída com cinco nós, submetida a rajadas

Tipo	Tempo de Fila	Utilização	Tempo de Resposta	Throughput
Mensagem de Controle	0,0897; 0,5159	-	-	-
Mensagem de Produtividade	0,0908; 0,5374	-	-	-
Mensagem Operacional	0,1032; 0,5487	-	-	-
Consulta Simples	0,1124; 0,5713	-	-	-

Consulta Mediana	0,1203; 0,8732	-	-	-
Consulta Complexa	0,1365; 1,7254	-	-	-
CPU	-	0,0491	-	-
Banco de Dados	-	0,7293	-	-
Sistema	-	-	6,8421	762,1793

Os resultados de tempo de espera na fila demostram o quanto esta abordagem traz de benefícios para a solução. Mesmo submetido às rajadas, os tempos de espera das filas não alteram tão significativamente quanto as da solução convencional com *speed up*, novamente pelo fato de que quando um nó está alocado, existem outros quatro que podem estar livres, principalmente quando as consultas complexas entram no sistema.

5.7 Conclusão dos resultados encontrados nas Simulações

Observar as tabelas que contém os resultados das simulações deixa claros os benefícios de prover redundância do banco de dados para cada nó do pool de servidores.

O grande gargalo do sistema é o banco de dados, e este problema é agravado quando uma consulta complexa é requisitada, pois é fato que o nó ficará alocado por um longo período de tempo. Como a taxa de entrada destas consultas complexas não é tão grande (se comparado com as demais), o fato de ter mais servidores traz muitos benefícios para a solução, pois mesmo se o nó alocado não tiver recursos para processar as consultas complexas em um período curto de tempo, significa que para esta requisição o sistema vai ter um tempo de resposta ruim; contudo, para as demais, que encontrarão nós livres, terão um tempo de resposta bom.

Este fato garante que na média, a solução possuirá tempos de resposta aceitáveis, semelhantes inclusive às soluções caras.

Capítulo 6

PROPOSTA DE UMA APLICAÇÃO SERVIDORA DISTRIBUÍDA

Através dos fundamentos obtidos no capítulo três, levantamento e estudo de dados reais do capítulo quatro e os resultados das simulações de diferentes sistemas de filas do capítulo cinco, foi possível criar a arquitetura da aplicação servidora distribuída. Cada entidade da solução é identificada e detalhada, evidenciando todas as tomadas de decisões do projeto.

A arquitetura foi concebida com base no modelo de arquitetura de alto desempenho para tráfego em rajadas estudada pelo grupo de pesquisa do GSDR/DC-UFSCar, proposta por Nobile (NOBILE, 2007a, 200 b) e posteriormente ampliada por Juliano (MARCELLO, MORON & TREVELIN, 2010), estendendo seu trabalho para uma arquitetura distribuída, na qual cada nó que irá compor o *pool* é independente e possui seu próprio banco de dados. Prover a redundância da informação em vários locais distintos é fundamental, uma vez que o capítulo anterior identificou que para este tipo de aplicação, o grande gargalo é o banco de dados. A figura 6.1 mostra de forma detalhada a arquitetura baseada nos resultados obtidos no capítulo 5, contemplando um *pool* de computadores para ampliar a capacidade do sistema, de forma análoga à modelagem da seção 5.4.

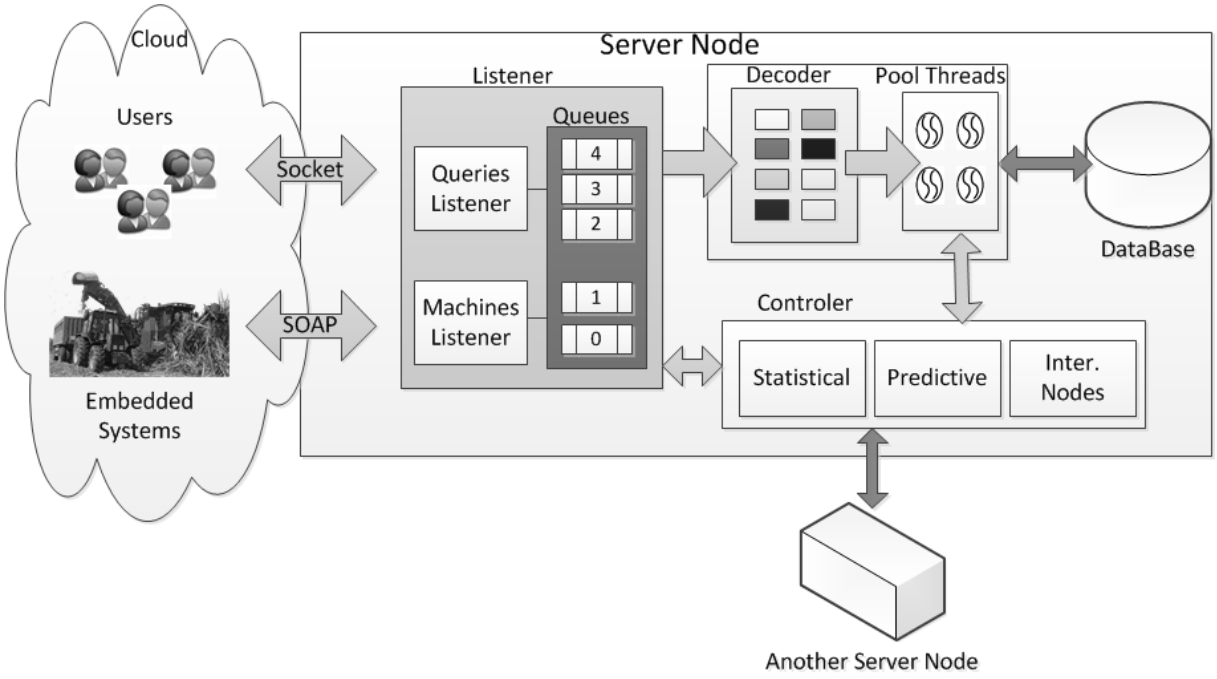

Figura 6.1 – Diagrama da arquitetura da Aplicação Servidora

Esta abordagem mostrou-se interessante sobre o ponto de vista de criar a concepção de um Nó Servidor, tanto de forma conceitual quanto no seu desenvolvimento em linhas de código. A aplicação foi escrita em Java e faz esta representação através da classe NoServidor.java, que contém informações referentes ao computador que está hospedando esta aplicação servidora, como: tempo de serviço para cada classe, IP e porta dos serviços, nome, MAC, número de requisições sendo processadas, entre outras.

Conforme mostra o fluxograma da figura 6.2, quando um nó é adicionado ao *pool*, ele realiza um teste e verifica se já existe um nó mestre atuando. Se existir um nó mestre, ele primeiramente gera uma instância de seu próprio nó, adiciona em sua tabela *Hash* esta instância e envia este mesmo nó para o servidor mestre e inicializa o serviço de intercomunicação de nós, responsável por toda a comunicação e repasse de requisições (CHEN & MUNTZ, 2006). Senão, ele assume o papel de nó mestre, cria sua instância, armazena na tabela *Hash*, inicializa os servidores SOAP e de Mensagens, para fazer o front com as requisições oriundas da internet (usuários e máquinas em campo), inicializa o serviço de intercomunicação entre nós e passa a reger todas as tomadas de decisões para esta aplicação.

O serviço de intercomunicação de nós é responsável para que todos os nós deste *pool* possuam instâncias de cada computador, ou seja, cada nó tem as

informações de todos os nós do *pool*, possibilitando a eleição de um novo nó mestre em caso de falha e a fácil inserção e remoção de nós da aplicação. Outro detalhe importante são os tempos de serviços e quantidades de requisições sendo atendidas contidas nestas instâncias. Estas informações permitem ao nó mestre tomadas de decisões mais conscientes, uma vez que pode calcular o tempo médio que uma requisição levará se fosse calculada por cada nó e decidir enviar para o que responderá em menor tempo, fazendo com que o tempo de resposta médio da solução seja menor.

A transmissão destes objetos é realizada utilizando as primitivas *Socket*, a fim de obter desempenho superior às tecnologias Java RMI, que apesar de facilitar o desenvolvimento da solução insere um *overhead* maior.

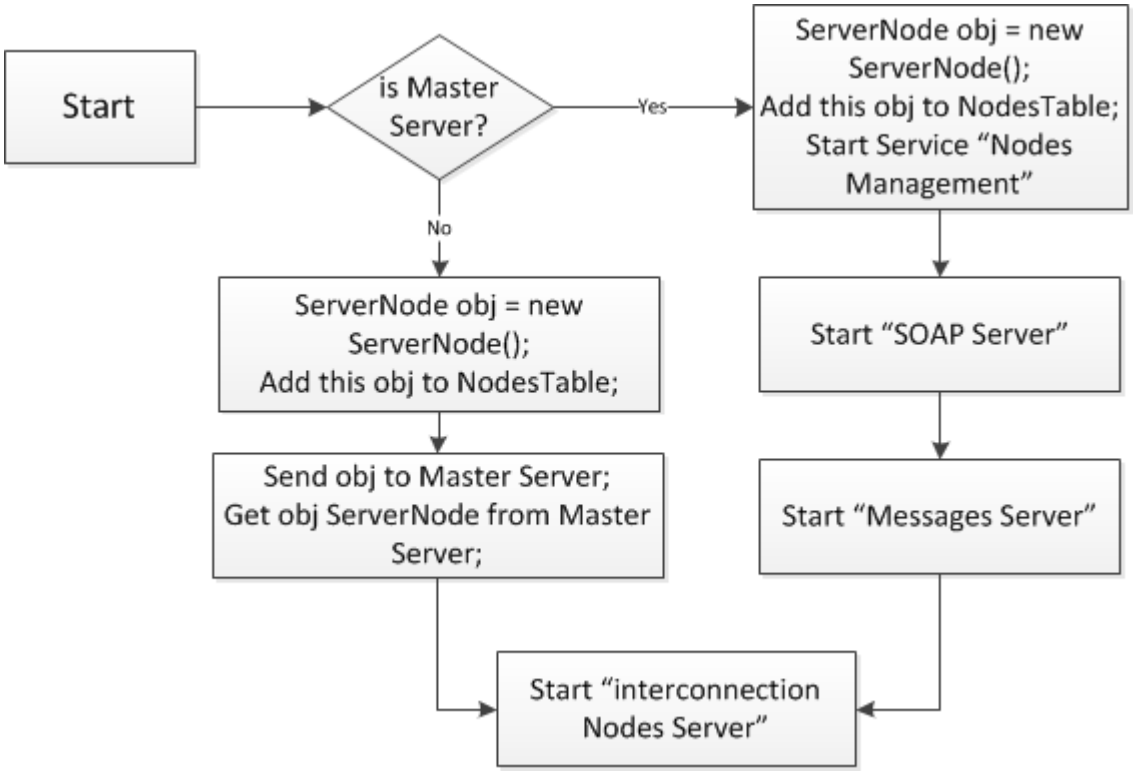

Figura 6.2 – Diagrama de fluxo da arquitetura da Aplicação Servidora

Para melhor compreensão, cada estrutura contida na figura 6.1 e os conceitos envolvidos serão explicados e contextualizados, demonstrando o potencial de expansão benefícios para a solução.

6.1 Listener

Responsável pelo recebimento das mensagens e requisições oriundas do campo e dos usuários. Centraliza todas as entradas com o intuito de facilitar a coleta de métricas importantes (pelo Controlador), como taxa de requisições por segundo. Primeiramente, esta entidade analisa o tipo de requisição, se vier através de um socket TCP ou UDP, quer dizer que é uma máquina, portanto encaminha a requisição para o *listener* especializado em tratar mensagens, senão, se for uma estrutura SOAP, o *listener* de consultas receberá esta requisição. Esta estratégia foi adotada para permitir possíveis expansões, na qual outros protocolos podem ser integrados da mesma forma.

O primeiro passo que ambos os *listeners* realizam é verificar a classe da requisição, analisando sua prioridade e inserindo cada requisição na sua respectiva fila.

6.2 Controlador

Entidade mais importante desta solução, responsável pela coleta de informações, cálculos de previsão de demanda, e tomadas de decisões. Abriga três classes que trabalham em conjunto, mas em instâncias diferentes:

- Estatístico: Possui um *thread* que capta amostras em intervalos de tempos definidos pelo Controlador. Estas amostras são utilizadas para previsões futuras de demanda e log de desempenho;
- Preditivo (MARCELLO, MORON & TREVELIN, 2010): Responsável pelo fornecimento de previsões futuras de demanda, analisando o histórico fornecido pela classe Estatístico em intervalos também definidos pelo Controlador. As previsões ocorrem de acordo com a taxa de cálculo, ou seja, se o Preditivo obtém valores a cada cinco segundos, a predição é realizada para os cinco segundos futuros;

- Interconexão dos Nós: Serviço que coordena a atualização de todas as tabelas de nós presentes em todos os servidores do *pool*. Este módulo é fundamental para que o Controlador tome decisões corretas, pois antes de antecipar uma demanda futura, ele deve tratar as novas requisições com as informações do instante em que elas chegam.

Trabalhando em conjunto com estas três classes, o Controlador é capaz de fornecer recursos para atender às premissas de prioridade de cada classe, buscando o menor tempo de espera e o maior *throughput* para o sistema. Novamente, esta separação lógica permite uma fácil expansão de funcionalidades para a solução, além de permitir total controle de recursos. O Controlador é capaz de alterar o número de *threads* que atenderá cada fila, assim como o número de nós que efetivamente participarão do processamento da demanda. A alocação de recursos deve ser feita de forma rápida para suprir a demanda, e a desalocação destes recursos deve ser mais lenta, para responder melhor a oscilações de demanda conforme demonstra a figura 6.3:

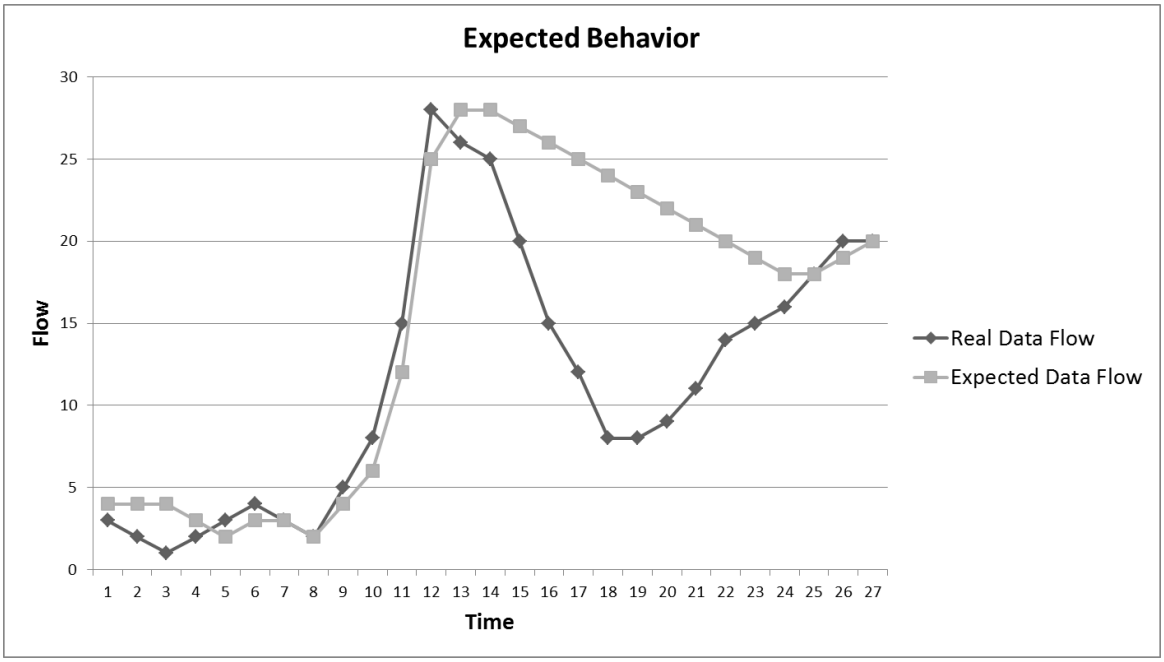

Figura 6.3 – Gráfico representativo do comportamento esperado da solução

Esta estratégia de desalocar recursos de forma lenta foi inspirada no protocolo TCP. Assim que inicia uma conexão, o protocolo aumenta o tamanho da janela utilizando uma progressão geométrica até atingir a linha de *threshold*, denominada partida lenta. Assim que esta linha é ultrapassada, o tamanho da janela

do protocolo passa a aumentar de forma lenta e gradativa. Portanto, o processo de desalocação de recursos é baseado nesta ideia, onde a alocação de recursos é semelhante à partida lenta e desaloca os recursos de forma gradativa e lenta, correspondendo ao valor de 10% do que realmente seria desalocado pelo resultado do módulo preditivo.

Nobile (NOBILE, 2007) propôs uma fórmula que calcula o quanto de recurso é necessário (QP) para uma classe baseando-se no tamanho atual da fila (Taf), do tamanho futuro previsto (Tfp), tempo de serviço (Ts) e o Deadline (D) desta classe, conforme mostra a fórmula 1:

$$QP = ((Taf + Tfp) * Ts) / D$$

Se a frota estiver sob uma área de sombra, o Controlador do nó mestre pode enviar um comando de desligamento para alguns nós ociosos, reativando-os quando necessário, atendendo também aos pré-requisitos do *Green Computing* (LO & QIAN, 2010), pois oferece uma solução atraente sob o ponto de vista computacional, na qual a adição de nós heterogêneos aumenta o poder computacional para esta aplicação e do ponto de vista financeiro, propõe uma solução mais barata que realiza também a gestão de recursos (e.g. Recursos calóricos e energéticos).

6.3 *Pool* de *Threads*

Determina efetivamente o *pool* de *threads* disponível para a solução localmente. A definição deste *pool* é importante para que o Controlador consiga redirecionar recursos para atender as filas de maiores prioridades em um pico, por exemplo. Os recursos alocados encarregam-se de consumir todas as requisições armazenadas nas filas.

Outro fator interessante é a suspensão de *threads* sobressalentes, diminuindo o *overhead* do sistema operacional para fazer o controle e gestão destes *threads*. O Controlador é capaz de alocar ou desalocar *threads* de acordo com a demanda também.

6.4 Decodificadores

A divisão de classes distintas para o manejo e decodificação é fundamental para garantir as boas práticas de programação para esta solução. Quando se trabalha com gestão de frotas, é comum deparar-se com veículos completamente distintos, como um carro convencional e uma colhedora de cana-de-açúcar. Cada mensagem deve ser decodificada por decodificadores especializados, que serão capazes de processar as *tags* e gerar o *script* de inserção no banco de dados corretamente. Outra vantagem é que permite inserir novas classes de veículos de forma transparente a solução.

6.5 Banco de Dados

Uma vez que já foi citado que o processamento de uma mensagem oriunda do campo é na ordem de micro ou nano segundos, e o tratamento de consultas na ordem de segundos ou minutos, fica evidente que o gargalo desta aplicação é o banco de dados. Para minimizar este problema, é conveniente distribuir também este recurso. Cada mensagem decodificada gera um script de inserção no banco que é replicado para todos os nós. Para este trabalho, todos os bancos serão idênticos, permitindo que uma requisição de consulta possa ser realizada em qualquer nó com integridade de dados.

6.6 Predição utilizando Box-Jenkins

Descrito na década de 70, estipula que para uma série temporal cada valor pode ser explicado por valores anteriores, a partir da utilização da estrutura de correlação em série ou autocorrelação, na qual geralmente estão entre os próprios valores da série. Os modelos Box-Jenkis, também conhecidos como ARIMA (Auto Regressivos Integrados Médias Móveis) (BOX & JENKINS, 1976), são modelos matemáticos que

capturam a correlação ou a autocorrelação entre os valores da série temporal e com base nestas informações observar o comportamento da série e realizar predições. Esta abordagem pode observar eventos sazonais, sendo conhecidos como modelos SARIMA, podendo conter uma porção não sazonal, com os parâmetros (p, d, q), e uma porção sazonal, com os parâmetros (P, D, Q).

A predição de eventos futuros é importante para a classe Controlador funcionar corretamente, conforme foi descrito anteriormente. Para realizar os cálculos da classe Predictive, foi utilizado o rJava, que faz uma interface em *runtime* com as bibliotecas do software R-*Statistics* (R, 2012).

6.7 Qualidade de Serviço (QoS)

Geralmente, os pacotes transmitidos por uma rede compartilham a mesma banda e possuem critérios bem definidos de QoS. Uma transmissão de vídeo com atrasos significativos pode caracterizar uma baixa QoS.

Como a infraestrutura de comunicação para frotas veiculares pode ser precária (no caso do agronegócio), é difícil estipular índices e tempos ótimos para a qualidade de serviço de uma mensagem oriunda do campo. A informação deve chegar para o usuário de forma que a operação não seja comprometida, compartilhando responsabilidades tanto do sistema embarcado quanto da aplicação servidora. Por exemplo, uma mensagem de alarme de temperatura do radiador deve acionar um evento que alerta o operador do veículo e que este alarme seja transmitido remotamente, para os gestores da operação tomar as devidas providências (como enviar uma equipe de manutenção para o local).

Este trabalho contemplou a aplicação servidora, portanto preocupou-se em processar a informação com o menor tempo de espera possível. Concluindo, uma mensagem chegar com algumas horas de atraso até o servidor não é necessariamente uma falha crítica, pois esta mensagem poderá ser necessária somente no final do mês para relatórios de produtividade.

Capítulo 7

RESULTADOS E CONCLUSÕES

Assim que a aplicação servidora distribuída foi implementada, testes reais foram realizados para comprovar que as estratégias proveram uma solução de baixo custo, pois utiliza servidores comuns, escalável e de alto desempenho, respeitando todas as premissas do projeto.

Quatro computadores compuseram o *pool* servidor da solução, cabendo a outro computador dedicar-se a gerar o fluxo de informações, baseando-se no *trace* real demonstrado na figura 4.3. A solução foi programada para executar durante um intervalo de tempo, gerando arquivos de *logs* que capturaram os tempos de fila, os vetores utilizados pela classe Predictive, o numero de nós que participaram da decodificação, os tempos de espera, taxa de utilização de processador e memória RAM de cada nó servidor e os valores previstos pelo Box-Jenkis. A figura 7.1 mostra o número de requisições que efetivamente foram realizadas por segundo e as previsões consideradas a partir destas informações, lembrando que propositalmente o controlador regula uma queda mais lenta, para melhor reação a variação entre picos. Mostra também a alocação de nós utilizados para processar esta demanda.

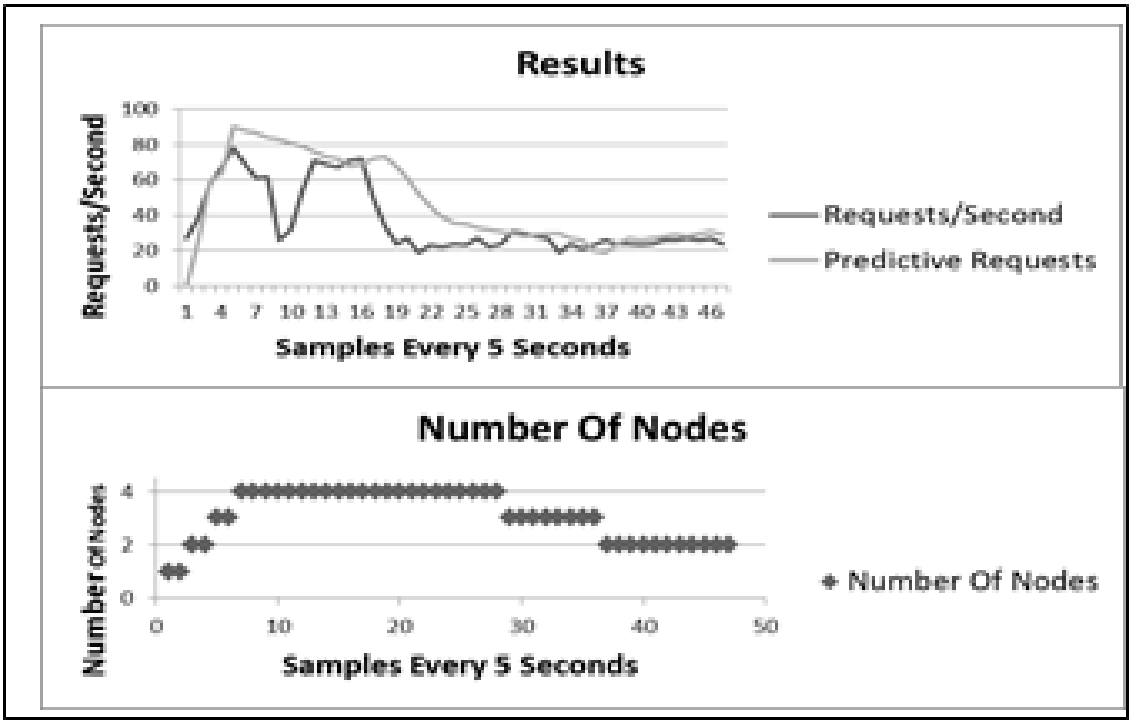

Figura 7.1 – Resultados da arquitetura da Aplicação Servidora

A tabela 7.1 mostra as médias obtidas de processamento e utilização de memória RAM para cada nó.

Tabela 7.1 – Resultados médios de uso de memória RAM e processador para cada nó.

Nó	RAM (MB)	Processador
Mestre	138	0.78
1	101	0.66
2	86	0.47
3	71	0.34

Estas informações mostram que o uso de memória RAM não é significativo para uma gestão de nós para uma aplicação desta natureza. Apesar de trabalhar com picos, a utilização dos processadores não foram tão elevadas, uma vez que houve períodos de ociosidade para alguns nós e conforme já havia sido dito o maior gargalo é o banco de dados.

A tabela 7.2 demostra a média de tempo de espera para cada fila. É interessante verificar que mesmo apesar do tempo de processamento das consultas complexas serem altas, a média geral foi aceitável, uma vez que esta consulta aloca um único nó, deixando o resto para processar as demais classes.

Tabela 7.2 – Medições dos tempos de aplicação servidora.

Fila	Tempo de Espera	Tipo de Mensagem	Média do tempo de Serviço
0	0.102	Controle	< 0.00001
1	0.308	Produtividade	< 0.00001
2	0.309	Operacional	< 0.00001
3	0.341	Consulta Simples	1.321
4	15.13	Consulta Mediana	12.893
4	15.13	Consulta Complexa	50.361

7.1 Conclusões

Os resultados obtidos demostram que é possível utilizar modelos matemáticos preditores, como o Box-Jenkis sem comprometer o desempenho da aplicação (ROBERT & ROBERTS, 2010) e obter bons resultados com alguns cuidados. Para viabilizar estes resultados, a classe Controlador teve que restringir a queda de recursos para as classes de serviços de maior prioridade, uma vez que a predição retornou valores baixos durante as quedas acentuadas entre os picos de processamento.

Este fato não quer dizer que a abordagem seja ruim, apenas que a análise do *trace* real mostrou que, na prática, a variância entre os valores máximos e mínimos durante os picos de fluxo são grandes, e que esta abordagem adequa-se de forma mais satisfatória.

Novamente, a classe Controlador permite ajustes em tempo de execução possibilitando, por exemplo, alterar a taxa de coleta para cálculos de acordo com a demanda, dentre outras possibilidades.

Embora o trabalho baseou-se no contexto agrícola do cenário brasileiro, rajadas de informações são eventos comuns a diversos tipos de aplicação. Como a classe Controlador centraliza as tomadas de decisões e a classe Predictive realiza as predições de cargas futuras para suprir as necessidades de recurso, a arquitetura

proposta permite sua aplicação a diversos outros domínios onde o perfil de comunicação possuir comportamento semelhante aos apresentados neste trabalho.

7.2 Trabalhos Futuros

A pesquisa forneceu resultados interessantes quanto à utilização de métodos preditivos para parametrização dos recursos da aplicação servidora com o objetivo de melhor atender as diferentes classes de mensagens e consultas.

Porém, devido ao curto tempo do mestrado, os trabalhos futuros contemplam:

1. Efetuar novas simulações visando avaliar mais exaustivamente o comportamento da arquitetura;
2. Estender as simulações para explorar outros comportamentos, através de outras distribuições;
3. Explorar melhor os aspectos das comunicações e seus impactos no problema, com vistas a melhorias na arquitetura no que diz respeito à comunicação;
4. Comparativos desta arquitetura da aplicação servidora com outros modelos clássicos de arquiteturas de alto desempenho, visando obter outras soluções correlacionando o desempenho versus custo;
5. Inserção de prioridades dinâmicas para verificar o comportamento e possível redução de tempo de fila de classes com menor prioridade;
6. Produção de mais artigos sobre o assunto.

7.3 Publicações

Durante o desenvolvimento do projeto, duas publicações foram realizadas contemplando o levantamento de infraestrutura e simulações, além de uma terceira que se refere aos resultados obtidos da aplicação servidora já implementada:

1. “Study and modeling of the server application for monitoring embedded systems of vehicle fleet in agribusiness”. *In:* II Brazilian Conference on Critical Embedded Systems (May 21 - 25, 2012, Campinas - SP, Brazil);
2. “Impact of intermittent connectivity for telemetry server applications of vehicular embedded systems in agribusiness: study and modeling of hi-performance architecture”. *In: 2012 International Symposium on Performance Evaluation of Computer and Telecommunication Systems (July 8-11, 2012, Genoa, Italy);*
3. “A low cost scalable predictive server architecture for embedded systems applications”. *In:* 2012 IEEE International Conference on Systems, Man, and Cybernetics (IEEE SMC 2012).

REFERÊNCIAS

ABRAMOV, V.M. Multiserver Queueing Systems with Retrials and Losses. **Anziam J**, v 48, p.297-314, 2007.

ALMEIDA, R.A.P.; MOSCHETTO, D.A.; GUARDIA, H.C. Modelo de Disseminação e Compartilhamento de Conteúdo com suporte à Comunicação Oportunística. **In: Webmedia'09 - 15th Brazilian Symposium on Multimedia and the Web,** p. 5–7, 2009.

ANATEL, Agência Nacional de Telecomunicações. Homologação de equipamentos transceptores digitais que empregam a tecnologia WiMAX na faixa de 2,5 GHz. Análise N° 24/2009 GCER, de 29 de Janeiro de 2009. Disponível em: <http://www.anatel.gov.br/Portal/verificaDocumentos/documento.asp?numeroPublicanum=248292&assuntoPublicacao=null&caminhoRel=null&filtro=1&documentoPath=224829.pdf>. Acesso em 25 de Maio de 2012.

ARAÚJO, A. Indicadores da função motomecanização aplicados em usina de açúcar e álcool em um ambiente gerenciado por processos: um estudo de caso. 2002. 110f. Dissertação (Mestrado em Engenharia de Produção) – Universidade Federal de Santa Catarina, Florianópolis. 2002.

BANGA, G.; DRUSCHEL, P. Measuring the capacity of a Web server under realistic loads. **Journal World Wide Web archive**, v.2, n.1-2, p.69-83,1999.

BOX, G. E. P., JENKINS, G. M., Time series analysis. Holden Day, 1976.

CHEN, A.; MUNTZ, R. R. Peer Clustering: A Hybrid Approach to Distributed Virtual Environments. **In: NetGames '06 Proceedings of 5th ACM SIGCOMM workshop on Network and system support for games Netgames'06**, artigo n 11, 2006.

DIJK, N.M.V. Analytic Error Bounds for Approximations of Queueing Networks with an Application to Alternate Routing. **J. Austral. Math. Soc. Ser. B**, v.31, p.241-258, 1990.

EMBRATEL, Definições de Localidade segundo o IBGE. Disponível em: http://www.embratel.com.br/Embratel02/files/secao/15/14/11262/definicao_localidade.pdf. Acesso em 08 de Março de 2012.

EYERMAN, S.; EECKHOUT, L. Modeling Critical Sections in Amdahl's Law and its Implications for Multicore Design. **In: 37th Annual International Symposium on Computer Architecture (ISCA 2010),** junho, p.19–23, 2010.

FAVARETTO, F. Uma contribuição ao processo de gestão da produção pelo uso da coleta automática de dados de chão de fábrica. 2001. 222f. Tese (Doutorado em Engenharia Mecânica) – Universidade de São Paulo, São Carlos. 2001.

GUIMARÃES, M.P. Tutorial sobre CORBA. Disponível em: http://www.lsi.usp.br/~paiva/sd/CORBA.doc. Acesso em 16 de Novembro de 2011.

GRITZALIS, S.; ILIADIS, J.; OIKONOMOPOULOS, S. Distributed component software security issues on deploying a secure eletronic marketplace. Information Management & Computer Security, v. 8, n. 1, p. 5-13, 2000.

IBGE, Instituto Brasileiro de Geografia e Estatística. Disponível em: < http://www.ibge.gov.br/home/geociencias/cartografia/default_territ_area.shtm>. Acesso em 10 de junho de 2012.

IEEE, Standards Association. Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications. Amendment 8: IEEE 802.11 Wireless Network Management. 2011. 433f. New York. 2011.

IEEE, Standards Association. Part 15.1: Wireless medium access control (MAC) and physical layer (PHY) specifications for wireless personal area networks (WPANs). 2005. 600f. New York. 2005.

IEEE, Standards Association. Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs). Amendment 2: Alternative Physical Layer Extension to support one or more of the Chinese 314–316 MHz, 430–434 MHz, and 779–787 MHz bands. 2009. 33f. New York. 2009.

IEEE, Standards Association. Part 16: Air Interface for Fixed and Mobile Broadband Wireless Access Systems. Amendment 3: Management Plane Procedures and Services. 2007. 202f. New York. 2007.

INMARSAT, Company. Marketplace. Disponível em: <http://www.inmarsat.com/Services/Land/Marketplace/default.aspx?language=EN&textexto=False>. Acesso em 10 de junho de 2012.

IRIDIUM, Company. How to buy an Asset Track. Disponível em: <http://www.iridium.com/Contact/HowToBuy.aspx?productID=142>. Acesso em 10 de junho de 2012.

JIN, X.; MIN, G. Modelling Priority Queueing Systems with Multi-Class Self-Similar Network Traffic. **In: IEEE International Conference on Communications 2007 – ICC,** 2007.

JOHNSON, T; MARGALHO, M. Avaliação de Desempenho de Sistemas Computacionais. Editora LTC. Rio de Janeiro, 2011.

KAZEM, I.; AHMED, D. T.; SHIRMOHAMMADI, S. A Visibility-Driven Approach to Managing Interest in Distributed Simulations with Dynamic Load Balancing. **In: ACM Digital Library of 11th IEEE Symposium on Distributed Simulation and Real-Time Applications,** 2007.

KEILSON, J.; SERVI, L.D. A distributional form of Little's Law. **Operations Research Letters**, v.7, n.5, p. 223-227, 1988.

KENDALL, D.G. Stochastic Processes Occurring in the Theory of Queues and their Analysis by the Method of the Imbedded Markov Chain. **The Annals of Mathematical Statistics**, v.24, n.3, p. 338-354, 1953.

KIROW. Características técnicas do produto Kirow Multi Mover 880 C. Leipzig, Alemanha. Kranunion Companies Group. Disponível em: <http://www.kranunion.de/fileadmin/Downloads/Multi_Mover_C_880.pdf>. Acesso em 17 de maio de 2011.

KIROW. Características técnicas do produto Kirow Slag Taurus. Leipzig, Alemanha. Kranunion Companies Group. Disponível em: <http://www.kranunion.de/index.php?id=52&L=1> Acesso em 17 de Mai 2011.

KLEINROCK, L. Queueing Systems: Theory, v1. Editora John Wiley & Sons, Inc., 1975.

LANGLEY R.B.NMEA 0183: A GPS Receiver Interface Standard. **GPS World**, p. 54-57, 1995.

LO, C.D.; QIAN K.; , "Green Computing Methodology for Next Generation Computing Scientists," **Computer Software and Applications Conference (COMPSAC)**, 2010 IEEE 34th Annual , vol., no., pp.250-251, 19-23 July 2010

LUO, J.; WILLIAMSON, C. Performance implications of fluctuating server capacity. **Journal Computer Communications archive,** v.31, n.16, p.3760-3770, 2008.

MAESTRI, R. Mais de um ano após o acidente com a Usina de Sayano Shushenskaya (75 mortes) ainda nada de concreto. Disponível em: <http://engenheiro.blogspot.com/2011/01/mais-de-um-ano-apos-o-acidente-com.html>. Acesso em 17 de maio de 2011.

MARCELLO, J.; MORON, C.E.; TREVELIN, L.C. A gateway architecture for QoS management considering time constraint application. **In: Systems Man and Cybernetics (SMC), 2010 IEEE International Conference,** p. 1300-1305, 2010.

MOTOROLA. Produtos e Serviços para Empresas. Disponível em: http://www.motorola.com/Business/XL-PT/Produtos+e+Servicos+para+Empresas/Radios+Bidirecionais+Seguranca+Publica/Radios+Portateis/APX_5500. Acesso em 10 de Junho de 2012.

NICOLA, V.F.; ZABURNENKO, T.S. Efficient Heuristics for the Simulation of Population Overflow in Series and Parallel Queues. **In: ACM Digital Library of First International Conference on Performance Evaluation Methodologies and Tools,** outubro, p.11-13, 2006.

NOBILE, P. N. Uma Arquitetura de Proxy com Prioridades de Serviços para Chamadas Remotas de Procedimentos de Tempo Real. 2007. Dissertação (Mestrado em Ciência da Computação), UFSCar, São Carlos. 2007.

NOBILE, P. N. ; LOPES, R. R. F. ; MORON, C. E. ; TREVELIN, L. C. . QoS Proxy Architecture for Real Time RPC with Traffic Prediction. **In: IEEE International Symposium on Distributed Simulation and Real Time Applications**, 2007, Chania. Proceedings of 11th IEEE International Symposium on Distributed Simulation and Real Time Applications, 2007.

OLIVEIRA, F.E. Rede Celular: Avaliação da Transição de GSM/GPRS para 3G/UMTS. III Concurso Teleco de Trabalhos de Conclusão de Curso (TCC) 2007.

(OMG, 2012). CORBA. Disponível em: http://www.omg.org/corba/. Acesso em 14 de Março de 2012.

ORACLE, Docs. Disponível em: http://docs.oracle.com/javase/1.5.0/docs/guide/rmi/codebase.html. Acesso em 07 de Fevereiro de 2012.

ORACLE, Java RMI. Disponível em: http://www.oracle.com/technetwork/java/index-jsp-139188.html. Acesso em 7 de Fevereiro de 2012.

PERKINSON, T.L.; MCLARTY, P.K.; GYURCSIK, R.S.; CAVIN, R.K., III; , "Single-wafer cluster tool performance: an analysis of throughput ," **Semiconductor Manufacturing, IEEE Transactions** on , vol.7, no.3, pp.369-373, Aug 1994

POSTGIS, Adds support for geographic objects to the PostgreSQL object-relational database. Disponível em: http://postgis.refractions.net/. Acesso em 23 de Abril de 2012.

R. The R Project for Statistical Computing, a language and environment for statistical computing and graphics. http://www.r-project.org/ (current Mai. 13, 2012).

ROBERT, P.; ROBERTS, J. A mean field approximation for the capacity of server-limited, gate-limited multi-server polling systems. **ACM SIGMETRICS Performance Evaluation Review,** v.38, n.2, p.24-26, 2010.

ROSSETTI, A.G.; MORALES, A.B.T. O papel da tecnologia da informação na gestão do conhecimento. **Ci Inf**, v.36, n.1, p.124-135, 2007.

SATCOM, 2011. Disponível em: <http://www.sensorantennas.com/antenna_pdf/GPS/S67-1575-168.pdf>. Acesso em 15 de junho de 2011.

SERAZZI, G. Performance Evaluation Modelling with JMT: learning by examples Politecnico di Milano - DEI, TR 2008.09, 366pp., 2008

SORRISO, R. Cenário em municípios produtores é de muito milho estocado a céu aberto. Disponível em: <www.aguaboanews.com.br/portal/index.php?option=com_content&view=article&id=9378:cenario-em-municipios-produtores-e-de-muito-milho-estocado-a-ceu-aberto&catid=29:agricultura&Itemid=202> Acesso em 17 de Mai 2011.

TANDLER. Gerador de 64MW explodiu no Irã. Disponível em: <http://eletricaesuasduvidas.blogspot.com/2011/04/gerador-de-64-mw-explodiu-no-ira.html>. Acesso em 17 de maio de 2011.

TANENBAUM, A.S. Redes de Computadores, 5ª edição, São Paulo, Editora PEARSON EDUCATION - BR, 2011.

TANENBAUM, A.S.; STEEN, M.V. Distributed Systems: Principles and Paradigms. Englewood Cliffs: Prenticle Hall, 2002.

TELECO, Tutoriais em telefonia celular. Disponível em: http://www.teleco.com.br/tutoriais/tutorialavaltrans/default.asp. Acesso em 11 de Outubro de 2011.

THEODORIDIS, Y.; STEFANAKIS, E.; SELLIS, T.; , "Cost models for join queries in spatial databases," Data Engineering, 1998. Proceedings., **14th International Conference on**, vol., no., pp.476-483, 23-27 Feb 1998

WARTHMAN, F. Delay-Tolerant Netkors (DTNs) – A Tutorial. Based on Vinton Cerf, Scott Burleigh, Adrian Hooke, Leigh Torgerson, Robert Durst, Keith Scott, Kevin Fall, Howard Weiss, Delay-Tolerant Network Architecture, DTN Research Group Internet Draft, 2003. Disponível em: <http://www.dtnrg.org/docs/tutorials/warthman-1.1.pdf>. Acesso em: 10 de fevereiro de 2011.

WIKIPEDIA, Simple Object Access Protocol. Disponível em: http://es.wikipedia.org/wiki/Simple_Object_Access_Protocol. Acesso em: 20 de Abril de 2012.

WIKIPEDIA, Teoria das Filas. Disponível em: http://es.wikipedia.org/wiki/Simple_Object_Access_Protocol. Acesso em: 20 de Abril de 2012.