GABRIEL MAURICIO OSWALD VIEIRA

**FERRAMENTA PARA MAPEAR A QUALIDADE PERCEBIDA PELO
USUÁRIO EM REQUISITOS DE QoS EM AMBIENTE CONVERGENTE
E HETEROGÊNEO**

São Paulo

2010

GABRIEL MAURICIO OSWALD VIEIRA

**FERRAMENTA PARA MAPEAR A QUALIDADE PERCEBIDA PELO
USUÁRIO EM REQUISITOS DE QoS EM AMBIENTE CONVERGENTE
E HETEROGÊNEO**

Dissertação apresentada
à Escola Politécnica da
Universidade de São Paulo como
requisito para obtenção do título de
Mestre em Engenharia.

Área de Concentração:
**Engenharia da Computação e
Sistemas Digitais**

Orientador:
Prof. Dr. Moacyr Martucci Junior

São Paulo

2010

Este exemplar foi revisado e alterado em relação à versão original, sob responsabilidade única do autor e com a anuência de seu orientador.

São Paulo, 20 de dezembro de 2010.

Assinatura do autor _____

Assinatura do orientador _____

FICHA CATALOGRÁFICA

Vieira, Gabriel Mauricio Oswald

Ferramenta para mapear a qualidade percebida pelo usuário em requisitos de QoS em ambiente convergente e heterogêneo /G.M.O. Vieira. -- ed.rev. -- São Paulo, 2010.

93p.

Dissertação (Mestrado) - Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia de Computação e Sistemas Digitais.

1. Serviços (Qualidade) 2. Multimídia interativa I. Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia de Computação e Sistemas Digitais II. t.

DEDICATÓRIA

A meus pais Roberto e Loty e a meus filhos Roberto e Nina.

AGRADECIMENTOS

Ao meu Orientador, Prof. Dr. Moacyr Martucci Júnior, pela orientação, confiança, profissionalismo e, principalmente, pela qualidade das opiniões em todos os momentos.

À Marinha do Brasil pela oportunidade e confiança.

A todos os professores, colegas e funcionários da Escola Politécnica da USP que colaboraram direta ou indiretamente na realização deste trabalho.

E a todos, sem exceção, que torceram por mim. Em especial, aos amigos CC(EN) Marcos Cesar M. Pinto e CF(EN) Marcio Moreira da Silva.

RESUMO

Uma das principais características dos sistemas de informação e de comunicação é permitir que os usuários finais utilizem e disponibilizem serviços interativos com conteúdo multimídia a qualquer hora, em qualquer lugar, com qualquer dispositivo e em movimento, independentemente da tecnologia de rede utilizada.

Um desafio que surge face a essa nova característica dos serviços é como identificar os requisitos de QoS (Qualidade de Serviço) fim-a-fim sob a ótica do usuário final em um ambiente convergente e heterogêneo com o intuito de evoluir as relações entre o usuário final e o Provedor de Serviços para o estabelecimento de um contrato com níveis de serviço personalizado.

Este trabalho apresenta uma pesquisa para a elaboração de uma ferramenta genérica e aberta para mapear a qualidade percebida pelo usuário em requisitos técnicos de QoS através da captura do fluxo de pacotes gerado entre as aplicações que compõem um serviço e simula atrasos, *jitter*, perdas e banda percebida com o objetivo de permitir ao usuário final identificar e expressar os valores desses parâmetros tornando transparente os aspectos tecnológicos que envolvem esta identificação.

A definição de um ambiente de simulação que hospeda o serviço, o desenvolvimento de um simulador de QoS específico e uma nova relação entre o usuário e o Provedor de Serviços também são abordados.

Palavras-chave: Simulador de QoS, QoS fim-a-fim, Qualidade Percebida, Arquitetura de Serviços Convergentes, Acordo de Níveis de Serviço Personalizado, Relacionamento entre Requisitos de QoS.

ABSTRACT

One of the main characteristics of communication and information systems is to allow final users to use and make available multimedia content interactive services at any time, anywhere, in motion and using any device regardless of network technology.

Under this perspective a challenge arises on how to identify the QoS (Quality of Service) requirements in a convergent and heterogeneous environment end-to-end user centric way, in order to evolve the relationship between users and service providers aiming the establishment of personalized demands through personalized agreements.

This work presents a study for the development of an open and generic tool for mapping the user perceived quality in QoS requirements through the capture of the packet flow generated between the applications of a service and simulates delays, *jitter*, loss of packets and bandwidth in order to allow the end user to identify and express the values of these parameters making transparent all the technological aspects involved in this identification.

The definition of a service simulation environment, the development of a QoS simulator and a new relationship between users and services providers are also addressed.

Keywords: QoS Simulator, End-to-end QoS, Perceived Quality, Convergent Service Architecture, User Service Level Agreement, QoS Requirements Relationship.

LISTA DE ACRÔNIMOS

CGI – Common Gateway Interface
DCU – Diagrama de Casos de Uso
DDS – Documento de Descrição do Serviço
DiffServ – Differentiated Services
DNS – Domain Name System
DVB-H – Digital Video Broadcasting - Handheld
ETSI – European Telecommunications Standards Institute
GPRS – General Packed Radio Service
IETF – Internet Engineering Task Force
IntServ – Integrated Services
IP – Internet Protocol
ITU-T – International Telecommunication Union – Telecommunication
JIT – Just-in-Time
JRE – Java Runtime Environment
JVM – Java Virtual Machine
ODP – Open Distributed Processing
OPS – Organização Provedora de Serviços
OSI – Open Systems Interconnection
QoE – Quality of Experience
QoP – Quality of Perception
QoS – Quality of Service
RSVP – Reservation Protocol
RTP – Real-Time Protocol
SLA – Service Level Agreement
SNMP – Simple Network Management Protocol
TCP – Transfer Control Protocol
UML – Unified Modeling Language
UDP – User Datagram Protocol
USLA – User Service Level Agreement

LISTA DE FIGURAS

Figura 1 - <i>E-infrastructure</i> para o desenvolvimento de aplicações.....	15
Figura 2 - Visão do Negócio.....	16
Figura 3 - Visão do Usuário do Processo de Simulação do Serviço	42
Figura 4 - Diagrama de Classes de Domínio.....	43
Figura 5 - Diagrama de Atividades	44
Figura 6 - Topologia da Rede Isolada do Ambiente de Simulação.....	49
Figura 7 - Modelo Arquitetural do Ambiente de Simulação	50
Figura 8 - Controles Deslizantes da Interface do QoSS.....	54
Figura 9 - Simulação de Entrega de Pacotes fora de ordem para a Aplicação	56
Figura 10 - Diagrama de Seqüência.....	56
Figura 11 - Redistribuição de vídeo (1→N).....	67
Figura 12 - Ambiente real do Xadrez Online e VLC Media Player.....	68
Figura 13 - Transmissão Remota via VLC MEDIA PLAYER	69
Figura 14 - Topologia do Ambiente Simulado do Xadrez Online	70
Figura 15 - “Sala” de jogos por pontuação do XadrezOnline (platéia).....	71
Figura 16 - Ambiente de Simulação do Estudo de Caso.....	72
Figura 17 - Controles Deslizantes para “n” Mídias	77
Figura 18 - Controles Deslizantes para “n” Mídias (2).....	77
Figura 19 - Perda de Pacotes por atrasos prolongados do <i>clip.avi</i>	78
Figura 20 – Disposição dos Selos de Tempo	80

LISTA DE TABELAS

Tabela 1 – <i>Lay-out</i> dos Requisitos de Simulação do Serviço.....	74
Tabela 2 – <i>Lay-out</i> para Aplicações do Tipo não-Principal	76

SUMÁRIO

1 INTRODUÇÃO	13
1.1 Objetivo	17
1.2 Justificativa	19
1.3 Metodologia	20
1.4 Estrutura do Trabalho	21
2 ASPECTOS GERAIS DE QUALIDADE DE SERVIÇO	22
2.1 Conceitos Relativos a Qualidade de Serviço	22
2.2 A Percepção e a Qualidade Percebida pelo Usuário	26
2.2.1 Técnicas de Simulação	28
2.2.2 Distância de Simulação	30
2.3 Conceitos Aplicados a Serviços Convergentes	30
2.3.1 Multimídia.	31
2.3.2 Interatividade	32
2.3.3 Mobilidade	33
3 A FERRAMENTA DE MAPEAMENTO	34
3.1 O documento de requisitos	35
3.1.1 Requisitos Funcionais	37
3.1.2 Requisitos não-Funcionais	39
3.1.3 Requisitos Normativos	39
3.2 Análise do Domínio	39
3.2.1 Aspectos Gerais sobre o Domínio do Problema	40
3.2.2 Diagrama de Casos de Uso (DCU)	41
3.2.3 Diagrama de Classes de Domínio	42
3.2.4 Diagrama de Atividades	44
3.3 Análise da Aplicação e Decisões de Projeto	46
3.3.1 Modelo Arquitetural do QoSS	46
3.3.2 O Ambiente de Simulação	52
3.3.3 O Oculista	53
3.3.4 A Interface do QoSS	53
3.3.5 Diagrama de Sequência	55
3.3.6 Ambiente Operacional	57

3.3.6.1 A Escolha do Ambiente.....	58
3.4 Implementação - Principais Aspectos do Código Fonte	61
3.4.1 Captura, Simula, Incrementa Banda Percebida e Re-envia Pacote	61
3.4.2 Controles deslizantes – Eventos e Métodos.....	66
4 ESTUDO DE CASO	67
4.1 Ambiente Real do Serviço	68
4.1.1 VLC Media Player.....	68
4.1.2 Xadrez Online	69
4.2 Topologia do Ambiente de Simulação	71
4.2.1 Parametrização dos Controles Deslizantes	73
4.2.2 Simulação dos Requisitos de QoS do Serviço.....	74
4.2.2.1 Perdas	77
4.2.2.2 Atraso e <i>Jitter</i>	78
4.2.2.3 Preço	79
4.3 Análise dos Resultados	80
5 CONSIDERAÇÕES FINAIS	84
REFERÊNCIAS	87

1 INTRODUÇÃO

Com a proliferação dos dispositivos de acesso (principalmente os móveis), das tecnologias de rede de acesso (principalmente sem fio) e dos serviços convergentes com conteúdo multimídia (Vídeo sob demanda, TV interativa, Telefonia móvel, Serviços Internet entre outros) surge a necessidade de se estabelecer níveis de qualidade de serviço personalizados, visto que os usuários tendem a ter demandas diferenciadas e, naturalmente, só desejam contratar serviços com as características e com a qualidade que efetivamente necessitam, o que pressupõe um SLA personalizado. Hyun-Jong Kim.,et. al. (2008) acrescenta que muitas aplicações exigem requisitos mínimos de QoS e este aspecto deve ser considerado no estabelecimento de um SLA. WARD (2005) observa que “Um SLA (*Service Level Agreement*) é um contrato de Serviço que inclui a avaliação criteriosa dos padrões da qualidade dos serviços acordados.”

O Mapeamento de aspectos subjetivos em requisitos tecnológicos com o intuito de se estabelecer níveis de qualidade percebida, tem sido objeto de muitas pesquisas, no entanto, estas pesquisas têm se norteado na identificação de novos padrões de qualidade. Ghinea G. (2005), por exemplo, ao tentar identificar novos padrões de qualidade, trata esses aspectos subjetivos como QoP (*Quality of Perception*) e define QoP como sendo a representação pelo lado do usuário dos aspectos mais técnicos de QoS. Ela estabelece que QoP não engloba somente a satisfação do usuário com a qualidade de uma apresentação multimídia, mas também a habilidade de analisar, sintetizar e assimilar o conteúdo informacional da apresentação e adota uma abordagem empírica para mapear QoP em QoS com o intuito de apresentar resultados reutilizáveis baseados em análise estatística do comportamento do usuário.

SERRA (2007) estabelece um método para determinar conjuntos de parâmetros de QoS sob a ótica do usuário para o estabelecimento de USLAs dentro de determinadas classes de serviços móveis, interativos, com conteúdo multimídia em ambientes de convergência tecnológica. O método consiste em quatro passos: 1. identificação do serviço a ser oferecido ao usuário final e elaboração de um grafo em árvore, que tem como objetivo identificar outros serviços que podem compor o

serviço a ser oferecido ao usuário final; 2. mapeamento dos parâmetros de QoS em um metamodelo; 3. validação do metamodelo, para que se possa extrair do metamodelo os parâmetros de QoS; 4. elaboração do USLA.

Hyun-Jong Kim.,et. al. (2008) propõe um método para avaliar a QoE (*Quality of Experience*) do usuário através de parâmetros de QoS e se fundamenta na hipótese de que o provisionamento tradicional de QoS (recursos de rede) é intimamente relacionado com a QoE do usuário e apresenta um modelo que co-relaciona essas duas abstrações. A experiência do usuário, no que concerne a QoE, já vem sendo investigada a algum tempo, conforme BROWN (1992), e é mais relacionada ao conhecimento e expectativa do usuário do que propriamente à qualidade percebida de uma transmissão multimídia. Estes três aspectos, no entanto, são intimamente ligados e são abordados mais adiante.

Não faz parte deste trabalho estabelecer padrões de qualidade e nem criar mecanismos que facilitem a identificação de novos padrões. Este trabalho visa à criação de um mecanismo que facilite ao usuário mensurar o nível de qualidade de serviço que o atenda em função de critérios pessoais.

De acordo com SERRA (2007), um serviço é um conjunto de aplicações devidamente estruturadas e concatenadas oferecidas aos usuários finais por organizações provedoras de serviço (provedor de serviços) que são entidades responsáveis por disponibilizar o serviço contratado pelo usuário final.

Segundo KARAM (2006), um serviço é um conjunto de aplicações oferecidas aos usuários por organizações e definidos segundo as regras de negócio da empresa visando atender ao usuário.

Assim, considera-se que um serviço tenha embutido aspectos como plano de negócio, infra-estrutura de hardware, software, rede e informação e todos estes aspectos concorrendo para atender ao usuário. O usuário ao contratar os serviços de um Provedor de Serviço espera que este tenha sob si o controle de todos estes aspectos.

A Figura-1 apresenta o contexto que o método proposto neste trabalho está inserido, onde o provedor de serviços propõe novos serviços baseado na expectativa do usuário e estes serviços são então analisados sob o ponto de vista do negócio.

KARAM (2006) observa que um Modelo de Negócio é uma ferramenta conceitual que contém um conjunto de objetos, conceitos e suas relações, com o objetivo de expressar o conceito do negócio que é uma descrição simplificada de uma idéia de negócio. A um modelo de negócio compete representar o entendimento dos fluxos dos processos dentro de um ambiente e, esse modelo deve estar suportado por processos que sejam orientados ao usuário.

Figura-1 E-infraestrutura¹ para o desenvolvimento de aplicações

¹ O framework alia objetivos sociais, econômicos e de meio ambiente à informação e tecnologias de comunicação. MARTUCCI JUNIOR, M., HIRAKAWA, A. R., 2008

Baseado na descrição do modelo de negócio (KARAM, 2006) e na composição do serviço (SERRA, 2007), é gerado um ambiente de simulação que hospeda o serviço com o objetivo de simular e identificar os requisitos de QoS baseado na percepção do usuário final.

Os requisitos de QoS identificados no processo de simulação são analisados sob o ponto de vista tecnológico (hardware, software, rede e informação) e submetidos a um plano de negócio (KARAM, 2006) para análise de viabilidade. O negócio viável pressupõe o estabelecimento de duas classes de contratos: SLA e USLA (*User Service Level Agreement*). (MARTUCCI, et al., 2008), (SERRA, 2007), (MAIA et. al., 2008), (KARAM, 2006).

Como mostra a Figura-2, sob o ponto de vista do negócio existem quatro entidades: Provedor de Serviço, Provedor de Conteúdo, Provedor de Infra-Estrutura e o Usuário Final. Desta forma, os relacionamentos entre Provedores de Serviço, Provedores de Conteúdo e Provedores de Infra-Estrutura são regidos por contratos do Tipo SLA e o relacionamento entre o Provedor de Serviço e o Usuário Final é regido por USLAs personalizados.

Figura-2 Visão do Negócio

Os Provedores de Conteúdo e os Provedores de Infra-Estrutura são entidades contratadas pelo Provedor de Serviço e são responsáveis pelo provimento, em ambiente real, do conteúdo e infra-estrutura tecnológica necessária para operar a composição do serviço selecionada pelo usuário final.

A hierarquia apresentada na Figura-2 entre os provedores de serviço e os provedores de infra-estrutura significa também que não há uma única entidade no ambiente corporativo que o gerencie e o controle por si só, mas sim diversas entidades autônomas (domínios administrativos) cuja cooperação resulta em um efetivo controle da qualidade de serviço de todo o ambiente corporativo.

Na Figura-1, os processos “Gerador do Ambiente de Simulação” e “Simulação do Serviço” são disponibilizados pelo Provedor de Serviço para operacionalizar a simulação do serviço pelo usuário final e, baseado nos resultados da simulação, verificar se eles estão em conformidade com o plano de negócio. Esta verificação se deve ao fato de que diferentes níveis de QoS percebida demandam diferentes recursos para serem disponibilizados pelos provedores de conteúdo e de infra-estrutura o que pode impactar o lucro do negócio.

A Figura-2 apresenta a visão do negócio na qual o usuário seleciona uma composição de serviços cujos componentes podem estar hospedados em mais de um provedor de conteúdo e utilizar mais de um provedor de infra-estrutura.

1.1 OBJETIVO

O objetivo deste trabalho é apresentar uma pesquisa sobre o desenvolvimento de uma técnica que visa estabelecer requisitos de QoS para serviços convergentes, interativos com conteúdo multimídia baseado na qualidade percebida pelo usuário final para o estabelecimento de USLAs personalizados.

Conforme estabelecido por SERRA (2007), o SLA será chamado de USLA (*User Service Level Agreements* - Acordo do Nível de Serviço do Usuário), pois o enfoque do contrato do nível de serviço estabelecido será sempre entre um determinado

usuário, que expressa seus anseios pessoais de qualidade, e o Provedor de Serviço. Um mecanismo para mapear os aspectos subjetivos em requisitos tecnológicos foi desenvolvido pelo autor para ser aplicado no Estudo de Caso deste trabalho e é denominado QoSS (QoS Simulator). Este trabalho também reconhece o conjunto de definições estabelecidas em SERRA (2007) relativas a convergência tecnológica, tais como serviços convergentes, multimídia, *handover*, interatividade, mobilidade entre outros.

A hipótese a ser confirmada na pesquisa é sobre a pertinência ou não da técnica utilizada para se estabelecer os valores dos parâmetros de QoS que são considerados baseado na qualidade percebida pelo usuário final.

O trabalho dá continuidade à pesquisa na área de Arquitetura de Sistemas Convergentes e Aspectos de Qualidade em Serviços, do LSA², e, mais especificamente a continuação da tese de doutorado de SERRA (2007) (Tese apresentada à Escola Politécnica da USP sob o título “Método para Identificação de Parâmetros de Qualidade de Serviços Aplicados a Serviços Móveis e Interativos”) no que tange a métodos de definição de QoS percebida e técnicas de medida dos parâmetros de QoS e ao *framework* de desenvolvimento aplicações do LSA (Figura-1) no que diz respeito aos processo de simulação e parametrização de serviços.

² Laboratório de Sistemas Abertos do Departamento de Engenharia da Computação e Sistemas Digitais da Escola Politécnica da Universidade de São Paulo.

1.2 JUSTIFICATIVA

Com a crescente demanda por serviços convergentes com conteúdo multimídia, a qualidade percebida pelo usuário passa a ser um requisito não-funcional importante para as aplicações nas quais se exige que determinados parâmetros de QoS estejam entre limites mínimos e máximos bem definidos e a QoS fim a fim deve considerar os valores destes parâmetros sob a ótica do usuário final.

Além disso, para serviços com restrição de tempo de resposta operando em ambiente heterogêneo, onde dispositivos com tecnologias e capacidades diferentes se comunicam através de tecnologias de redes diferentes, deve haver alguma adaptação dinâmica de QoS na disponibilização destes serviços (BELLAVISTA, 2003 e MAIA et. al., 2008) e algum mecanismo que trate esta adaptação de forma transparente com o intuito de subsidiá-la com valores de referência definidos através de um processo de identificação de requisitos de QoS sob a ótica do usuário final.

O estabelecimento de contratos de níveis de serviço personalizados entre um provedor de serviços e o usuário final ainda não é muito praticado e com isso, as partes não têm um entendimento claro sobre o nível de QoS adequado e qual a QoS contratada.

Um contrato personalizado entre o usuário final e o Provedor de Serviços pressupõe que aspectos individuais de QoS devam ser considerados na prestação de serviços e assim, o usuário passa a lidar com questões técnicas, como a avaliação de valores mínimos e máximos de requisitos de QoS, de modo a estabelecer seus anseios sobre qualidade. É real e observável, no entanto, que na maioria dos casos, o usuário prefere lidar com parâmetros de qualidade não-técnicos para se expressar como clareza e nitidez.

Infelizmente, os valores destes parâmetros subjetivos serão de difícil medição caso o usuário não disponha de um mecanismo que o auxilie na identificação destes valores.

1.3 METODOLOGIA

O mapeamento da qualidade percebida pelo usuário proposto neste trabalho é discutido e construído em três etapas:

A primeira etapa é composta pela contextualização da pesquisa na área do conhecimento, pela identificação e colocação precisa do problema a ser tratado no trabalho e pela procura de conhecimentos ou instrumentos relevantes ao problema;

A segunda etapa discute questões relativas à qualidade de serviço em ambiente convergente e heterogêneo de forma a estabelecer as bases para a proposta de uma solução genérica e aberta de mapeamento da qualidade percebida pelo usuário em requisitos técnicos de QoS.

A terceira etapa apresenta o processo de desenvolvimento da solução do problema baseado nos preceitos da Engenharia de Software utilizando a UML. Esta etapa engloba também os resultados da utilização da prova de conceito através de um estudo de caso.

A elaboração da pesquisa em sua forma discute os seguintes aspectos relativos à simulação de requisitos de QoS sob a ótica do usuário final, aplicáveis a serviços convergentes operando em uma infra-estrutura heterogênea:

- Uma mudança de paradigma visto que QoS normalmente é definido em função da capacidade de recursos da rede e não em função da satisfação do usuário.
- Evolução das relações entre o usuário final e o Provedor de Serviços ao se discutir uma visão personalizada de QoS para o estabelecimento de USLAs em consonância com as pesquisas de KARAM (2006), SERRA (2007), MARTUCCI, HIRAKAWA (2008) e MAIA (2009).
- A técnica utilizada para se mapear a qualidade percebida pelo usuário em requisitos técnicos de QoS em ambiente convergente e heterogêneo.

1.4 ESTRUTURA DO TRABALHO

O capítulo 2 da dissertação aborda conceitos necessários para o desenvolvimento do trabalho, ou seja, conceitos relativos a parâmetros de qualidade de serviço e conceitos aplicados a serviços convergentes operando em uma infra-estrutura heterogênea.

O capítulo 3 apresenta as fases do desenvolvimento de uma ferramenta de mapeamento baseado nos preceitos da engenharia de software.

O capítulo 4 apresenta um estudo de caso de uma aplicação que disponibiliza uma mídia contínua (VLC Media Player) simultaneamente a um serviço para se jogar xadrez que disponibiliza seu conteúdo através de aplicações transacionais.

O capítulo 5 apresenta considerações finais, ressaltando suas contribuições e sugerindo futuras pesquisas que possam complementar a abordagem discutida neste trabalho.

2 ASPECTOS GERAIS DE QUALIDADE DE SERVIÇO

A grande competitividade do mercado, praticamente, impõe a satisfação do usuário como o principal objetivo a ser alcançado por provedores de serviços. Entretanto, os métodos existentes para se medir essa satisfação tem se norteado pela análise de pesquisas de opinião, cujos resultados são subjetivos e efetuados através de processos lentos e onerosos (Hyun-Jong Kim., et al., 2008). A medição e o provisionamento da qualidade de serviço são geralmente definidos pela capacidade da rede e não em termos de satisfação do usuário. A ferramenta para mapear a qualidade percebida pelo usuário em requisitos de QoS se propõe a considerar a satisfação do usuário no estabelecimento de requisitos QoS específicos de rede para serviços convergentes e interativos com conteúdo multimídia.

A Qualidade de Serviço é um aspecto importante na operação e implantação de serviços em redes por estas atenderem a milhões de usuários por todo o mundo e a quantidade de usuários e dispositivos por usuário continua se expandindo. Neste contexto, a Qualidade de Serviço passa a ser um requisito não-funcional importante para as aplicações para o qual se exige que determinados parâmetros (atrasos, perdas, banda, etc.) estejam em limites mínimos e máximos bem definidos.

A Qualidade de Serviço deve ser garantida pelo serviço sendo oferecido e esta garantia estabelecida em um contrato personalizado.

2.1 CONCEITOS RELATIVOS À QUALIDADE DE SERVIÇO

De acordo com a ITU-T Rec. E.800,1993 apud GOZDECKI, J, 2003 que expressou a primeira definição de QoS como sendo “o efeito coletivo da performance de um serviço, efeito este, que determina o grau de satisfação do usuário do serviço.”

Segundo MENDES (2006) a qualidade de um serviço está em considerá-lo sob o ponto de vista dos consumidores.

Qualidade de Serviço é um conceito pelo qual aplicações podem indicar requisitos específicos para a rede antes destas aplicações começarem a transmitir suas informações. FLUCKIGER (1995).

De acordo com PUTMAN (2001), QoS é um conjunto de requisitos que se dedicam a quantificar uma propriedade da qualidade, definida, independentemente do significado com o qual ela é representada ou controlada e que afeta o comportamento de um ou mais objetos. QoS é uma abstração do comportamento não-funcional de um sistema. Uma propriedade de QoS é um valor mensurável, pode ser especificada em um contrato e medida após a ocorrência de algum evento.

Qualidade de Serviço fim a fim é aquela percebida pelo usuário em toda a cadeia de fornecimento, desde o início do provimento até a efetiva utilização pelo usuário final KARAM (2006).

SERRA (2007) estabelece que os serviços interativos com conteúdo multimídia “têm maior exigência de Qualidade de Serviço (QoS), pois além dos parâmetros de redes (atrasos, perdas, etc.), possuem parâmetros específicos para áudio, imagem e vídeo além de exigirem sincronização de áudio e vídeo, e a QoS fim a fim deve considerar os valores dos parâmetros de QoS sob a ótica do usuário final.”

MAIA (2009) observa que “embora existam diversas abordagens sobre o que significa qualidade de serviço, há pelo menos duas abordagens para se definir qualidade de serviço: a abordagem qualitativa cuja definição aborda requisitos não funcionais de sistema e que afeta a percepção do usuário; e a abordagem quantitativa comumente relacionada aos mecanismos de gerenciamento e controle de qualidade de serviço.”

Considerando a abordagem qualitativa, ou seja, uma abordagem que oferece a possibilidade de se avaliar aplicações reais sob o ponto de vista qualitativo (MINDLIN, 2006), o universo que se deseja investigar neste trabalho são sessões nas quais ocorrem a simulação de requisitos de QoS gerados por um processo genérico e aberto centrado na qualidade percebida pelo usuário final.

Uma sessão é um conjunto de interações entre um usuário final e um serviço com o intuito de se atender a um propósito específico. Por exemplo: Uma sessão entre o usuário e um serviço de home banking com o intuito de se conhecer o saldo da conta-corrente e investimentos em poupança.

Os requisitos de QoS que serão considerados na simulação dessas sessões tem que atender aos seguintes critérios:

- Influência direta na qualidade percebida de áudio e vídeo e na qualidade percebida de sincronização de áudio e vídeo de um serviço convergente interativo com conteúdo multimídia.
- Influência direta no tempo de resposta de aplicações transacionais.

Os requisitos de confiabilidade e disponibilidade são relevantes sob o ponto de vista de simulação de falhas, visto que

a **confiabilidade** é relativa a capacidade de um sistema de se manter funcionando mesmo em condições adversas ou inesperadas e a **disponibilidade** é relativa ao tempo que o sistema fica disponível para uso.

Entretanto, o usuário não precisa de nenhum mecanismo adicional para expressar seus anseios sobre estes dois aspectos da qualidade, mas pode considerar relevante observar as conseqüências de tais adversidades.

Para simular estes requisitos, poder-se-ia interromper o fluxo de rede entre o dispositivo do usuário e o serviço, ou simplesmente desconectar um cabo de rede o que representaria a simulação de uma queda de energia do Provedor de Serviços. No entanto, ficar sem energia durante uma sessão de confirmação dos dados de uma compra via cartão de crédito não é algo cujas conseqüências sejam imprevisíveis para a maioria dos usuários.

No caso destes requisitos, e em detrimento da técnica de simulação, a técnica de experimentação é mais adequada para a avaliação do impacto destes requisitos em uma transmissão multimídia como veremos mais adiante. Desta forma, os requisitos de QoS que serão considerados na simulação das sessões, independentemente do nível de abstração e classificação (qualitativo ou quantitativo) são: atraso, *jitter*,

perdas, banda percebida, tempo de processamento, tempo de resposta, qualidade percebida e preço. Estes requisitos são definidos a seguir.

- **Atraso:** tempo gasto para um pacote ser enviado da origem até ser processado no destino, considerando: (i) atraso de processamento: decodificação e tratamento do cabeçalho do pacote de dados; (ii) atraso de fila; (iii) atraso de transmissão: tempo de processamento para “retirar os bits do meio e processá-los”; (iv) atraso de propagação: tempo de propagação de sinal no meio MAIA (2009).

Segundo COMER (2007), atraso se refere à diferença de tempo que um bit é injetado na rede até o momento de sua saída.

- **Jitter:** diferença entre o atraso experimentado por um pacote e pelo pacote subsequente. MAIA (2009).

Segundo COMER (2007), o *jitter* é um termo utilizado para descrever variâncias indesejadas no atraso, causadas quando em uma seqüência de pacotes, um pacote experimenta um atraso maior que outro. O *jitter* pode ocorrer quando roteadores em uma rede orientada a pacote se tornam congestionados.

FLUCKIGER (1995) define *jitter* como sendo a variação do tempo do “*transit delay*” em uma rede e define “*transit delay*” como sendo o tempo gasto na emissão do primeiro bit de um “*data block*” pela origem e seu recebimento pelo destino. TANENBAUM (2003) acrescenta que o *jitter* é a variação indesejada no tempo de chegada dos pacotes.

O *jitter* ocorre principalmente, pela utilização do hardware ou por congestionamentos e pode ter seu efeito atenuado através do aumento da largura de banda SUBBIAH (2008).

- **Porcentagem de perda de pacotes:** porcentagem de pacotes que não atingem o destino ou são corrompidos durante uma

transmissão ou período de tempo (MAIA, 2009). Segundo TANENBAUM (2003), perdas podem ocorrer por erros de transmissão por falha de hardware ou indisponibilidade de recursos nos roteadores devido a congestionamentos.

- **Banda percebida:** quantidade de pacotes injetados na rede pelo serviço em um determinado período de tempo durante uma sessão do usuário.
- **Tempo de processamento:** é o tempo que uma aplicação leva para processar uma determinada requisição. Este tempo não inclui o tempo que a mensagem leva para trafegar de sua origem até o destino.
- **Tempo de resposta:** Considera o tempo decorrido entre uma ação do usuário no dispositivo até o reconhecimento da resposta da ação pelo usuário.
- **Preço:** valor cobrado ao usuário pela utilização dos serviços.

2.2 A PERCEPÇÃO E A QUALIDADE PERCEBIDA PELO USUÁRIO

As redes de computadores como LANs, WANs ou a Internet tem comportamento imprevisível e muitas pesquisas são feitas para se garantir a QoS de um serviço (TANENBAUM, 2003). No entanto, este não é um problema concernente ao usuário. O usuário espera que ao acionar um serviço, este esteja disponível e que se comporte com a qualidade que atenda as suas expectativas. Cada usuário tem uma expectativa diferente em função da sua experiência (Hyun-Jong Kim, 2008). A análise destas diferenças, no entanto, se torna complexa ao se observar o indivíduo sob os pontos de vista biológico, fisiológico, social, psíquico, intelectual etc. Portanto, será considerado que todos estes aspectos, ao longo do tempo, compõem a experiência e o conhecimento do usuário.

A experiência e conhecimento do usuário influenciam sua percepção em muitos aspectos. Um exemplo seria sob o ponto de vista fisiológico, ou seja, o grau de funcionamento dos sentidos de visão e audição. No entanto, todos estes aspectos são de foro íntimo do usuário e relativos à sua individualidade e não serão

investigados separadamente. Esta pesquisa se dedica a mapear a resultante destes aspectos a qual chamamos de percepção ou mais especificamente de qualidade percebida pelo usuário com relação a serviços convergentes e interativos com conteúdo multimídia.

- **Qualidade percebida** é o julgamento qualitativo do resultado da transmissão do conteúdo em termos de mídia e quaisquer outros aspectos relacionados com o serviço em questão. MAIA (2009).

Segundo GOZDECKI (2003), a qualidade percebida reflete a experiência do usuário no uso de um serviço em particular.

São alguns aspectos de usabilidade dos serviços interativos com conteúdo multimídia que estimulam os sentidos do usuário, geram reações e como consequência determinam uma demanda personalizada de recursos, visto que cada usuário pode reagir de maneira diferente. Tamanho e resolução de vídeo, *frame-rate*, quantidade de cores, dinamicidade de vídeo, controles visuais da aplicação entre outros são alguns exemplos desses aspectos.

Neste trabalho, os aspectos subjetivos da percepção do usuário são representados pela qualidade percebida pelo usuário em apresentações multimídias e os requisitos tecnológicos são representados pelos parâmetros de QoS (atraso, *jitter*, perdas, etc.).

Os aspectos concernentes ao mapeamento da qualidade percebida em requisitos de QoS considerados são o conjunto de relacionamentos entre a visão e audição (sentidos) com os recursos de interação de um serviço.

O mecanismo de simulação de requisitos de QoS foi projetado para ser sensível à vontade e satisfação do usuário no que se refere a serviços interativos com conteúdo multimídia. Assim, seja a audição e a visão do usuário um conjunto de sentidos e *frame-rate*, sincronismo de áudio e vídeo, voz e controles visuais um conjunto de recursos de interação, e se este fosse o universo considerado, ter-se-ia o seguinte conjunto de relacionamentos:

Visão - - - - - *frame-rate*

Visão e Audição - - - - - sincronismo de áudio e vídeo

Visão - - - - - controles visuais (reação do usuário)

Audição - - - - - voz

Os parâmetros de QoS considerados neste trabalho, atrasos, *jitter*, perdas, banda percebida, tempo de processamento e tempo de resposta, influenciam de alguma forma em todos os itens deste conjunto de relacionamentos. Assim,

O mecanismo de simulação se dedica a valorar os parâmetros de QoS na medida em que o usuário navega conscientemente ou não pelos itens do conjunto de relacionamentos;

e o termo mapeamento representa a avaliação conjunta ou não da qualidade percebida dos itens deste conjunto.

2.2.1 Técnicas de Simulação

A um conjunto de processos que de alguma forma concorrem para atingir uma determinada meta ou finalidade normalmente pode ser chamado de sistema. KARAM (2006) acrescenta que um processo é uma seqüência lógico-temporal de atividades. O WEBSTER'S (2006) define um sistema como sendo um conjunto de coisas de tal forma relacionadas, que constituem uma unidade ou um todo orgânico. PRESSMAN (2005), estabelece que um sistema é um conjunto ou arranjo de elementos que são organizados para atingir um meta predefinida através do processamento de informações.

Conforme JERUCHIM (2000), para entendermos melhor os processos de um sistema, fazemos uma série de suposições de como ele deveria funcionar e estas suposições constituem um modelo que é utilizado para adquirirmos um melhor entendimento sobre como o sistema correspondente se comportará. Entenda-se como suposição a análise dos comportamentos ou estados possíveis dos processos

considerados. Neste contexto, uma suposição é o ato de se investigar um determinado estado de um processo.

Se os relacionamentos que compõe este modelo forem suficientemente simples, usam-se métodos formais (álgebra, teoria de filas etc.) para se obter a informação exata sobre as questões de interesse, entretanto, a maioria dos sistemas do mundo real são muito complexos para que modelos realísticos sejam avaliados analiticamente. E nestes casos, estes modelos devem ser construídos utilizando outras técnicas como, por exemplo, a construção de protótipos.

A seguir, uma breve descrição sobre as principais técnicas de simulação.

Basicamente, de acordo com SANTANA (1997), as técnicas de simulação podem ser agrupadas em duas linhas: experimentação e modelagem.

A técnica de experimentação é indicada quando o sistema a ser avaliado já existe, caso contrário, deve-se optar pela elaboração de um protótipo, que é uma simplificação do sistema original, com o qual serão realizados os experimentos. Por outro lado, a técnica de modelagem requer uma abstração a partir das características essenciais de um sistema real e é representada de acordo com um modelo escrito com um método formal.

Ao se escolher a técnica de modelagem, tem-se que decidir de que maneira o modelo será construído. Existem duas alternativas: método analítico ou simulação. O método analítico (solução analítica) produz resultados mais rápidos, mas nem sempre é aplicável, pois, para se encontrar uma solução analítica, muitas simplificações do modelo são necessárias. Essas simplificações impõem restrições que via de regra distorcem a realidade do sistema. A simulação, por outro lado, permite a inclusão de muitos aspectos do sistema sem a imposição de restrições. O problema é com o tempo demandado pelo software que simula o processo. Esse tempo normalmente cresce com a complexidade do modelo.

Na técnica de modelagem, e ao se descartar a solução analítica, utiliza-se a técnica de simulação quando as técnicas de experimentação ou protótipo são inviáveis ou

onerosas. No caso deste trabalho, a experimentação é inviável, pois apesar do sistema existir (rede real do provedor de serviços) ele não nos permite abstrair os aspectos tecnológicos da rede, sendo que esta abstração é um requisito essencial para que apenas os aspectos da qualidade percebida pelo usuário prevaleçam na identificação dos requisitos de QoS. O desenvolvimento de um protótipo também é inviável pelo mesmo motivo visto que a proposta deste trabalho não é simular os componentes de uma rede e sim abstraí-los ao simular os valores de parâmetros de QoS sob a ótica do usuário final.

A solução analítica nos permite conhecer o comportamento da rede que será simulada neste trabalho, visto que todos os pacotes injetados na rede tem seu comportamento conhecido e controlado pelo QoS Simulator (simulador apresentado no próximo capítulo), entretanto, os valores dos parâmetros de QoS estabelecidos por esta técnica não teriam utilidade pois o que se deseja investigar é o reconhecimento desses valores estabelecidos via qualidade percebida pelo usuário final através de um processo de simulação destes valores.

2.2.2 Distância de Simulação

O objetivo do Simulador de QoS, que é apresentado nas próximas seções, é simular requisitos de QoS de um serviço como se os mesmos estivessem ocorrendo em um ambiente real. Entretanto, o tempo para se processar a simulação de cada pacote, incluindo o tempo de reação dos atores envolvidos, influencia de maneira relevante no tempo de tráfego simulado de cada pacote.

A este “custo” de processamento na simulação de cada pacote chamamos de Distância de Simulação. A identificação e o tratamento dos valores deste “custo” são apresentados no estudo de caso deste trabalho.

2.3 CONCEITOS APLICADOS A SERVIÇOS CONVERGENTES

A contínua evolução das infra-estruturas de telecomunicações, dos serviços ligados à computação e das tecnologias de geração de conteúdo levam à existência de serviços globais que poderão ser acessados por quaisquer redes de comunicação,

em qualquer lugar, e com conteúdo de qualquer tipo, ou seja, heterogeneidade da rede e convergência dos serviços MAIA (2009).

De uma maneira geral, pode-se dizer que a convergência deve tratar de pelo menos três aspectos:

- Convergência de redes;
- Convergência de conteúdo;
- Convergência de terminal (dispositivo do usuário).

No cenário da convergência tecnológica há diversos aspectos que ainda precisam de uma solução para essa nova realidade, sendo que um deles é a definição da Qualidade de Serviço sob a ótica do usuário final.

Segundo MENDES (2006) para diversos observadores e teóricos, entre eles Rosnay e Lojkine, hoje se presencia a chamada revolução informacional, que, partindo da retroalimentação e da sinergia de uma série de tecnologias, foi construído o que Castells chama de Era da Informação e do Conhecimento. A convergência de meios traz novos desafios exatamente quando a informação, principal produto da sociedade do conhecimento, passa a ter maior facilidade de transitar e menor possibilidade de controle.

2.3.1 Multimídia

Multimídia é a combinação, controlada por computador, de pelo menos um tipo de mídia estática (texto, fotografia, gráfico), com pelo menos um tipo de mídia dinâmica (video, audio e animação) FLUCKIGER (1995). FLUCKINGER ainda estabelece que Multimídia Digital é a integração controlada por computador de textos, gráficos, imagens estáticas e em movimento, animação, sons e qualquer outro meio, onde cada tipo de informação pode ser representada, armazenada, transmitida e processada digitalmente.

Para FLUCKIGER, os sistemas que tratam conteúdo multimedia possuem pelo menos quatro características:

- Serem controlados por computador;
- Serem integrados;
- A informação que eles manuseiam tem que ser representadas digitalmente;
- A interface do usuário final tem que permitir interatividade.

Segundo SERRA (2007), multimídia é a combinação de elementos de texto, gráficos e imagens com ou sem movimento para apresentação de informações de forma agradável ao usuário. Para TANENBAUM (2003), quando as pessoas se referem à multimídia, em geral elas querem se referir à combinação de dois ou mais meios contínuos, ou seja, meios que têm de ser reproduzidos durante um intervalo de tempo bem definido, e em geral com alguma interação do usuário. Na prática, os dois meios normalmente são áudio e vídeo, ou seja, sons e imagens em movimento.

Este trabalho considera que uma transmissão multimídia envolve pelo menos duas mídias dinâmicas como voz e vídeo e que fazem mais sentido quando sincronizadas ou que tenham alguma relação de dependência.

2.3.2 Interatividade

Sob um ponto de vista genérico, interatividade pode ser a relação entre duas ou mais pessoas (ou coisas) que, em determinada situação, adaptam seus comportamentos e ações umas as outras BELLONI (2008).

Sob o ponto de vista de um serviço com conteúdo multimedia, JENSEN (1998) coloca que interatividade é uma medida do potencial de habilidade que uma mídia tem de permitir que o usuário exerça influência sobre o conteúdo ou a forma da comunicação mediada.

Para FLUCKIGER (1995), a interatividade provê graus de customização da maneira pela qual a informação é pesquisada e apresentada. O tempo ou duração, a ordem ou arranjo, rapidez, e a forma são os quatro graus possíveis.

Conforme estabelecido por SERRA (2007), Interatividade é uma forma de o usuário se comunicar ativamente com um serviço digital, com tempo de resposta adequado

para o serviço oferecido, buscando as informações, recuperando-as, integrando-as, processando-as e **construindo novos conhecimentos**. Um serviço cujo conteúdo usa diferentes mídias digitais e interatividade, pode ser classificado como sendo um serviço interativo com conteúdo multimídia.

2.3.3 Mobilidade

Mobilidade é a habilidade de usuários ou outras entidades em movimento se comunicarem e acessarem serviços independentemente de mudança de localização ou ambiente SERRA (2007).

Segundo de KARAM (2006), mobilidade é a capacidade de uso de diferentes tecnologias de acesso, desprezadas as diferentes localizações, enquanto o usuário se movimenta por si só, permitindo a esse usuário comunicação, uso e gerenciamento consistente de sua aplicação ou serviço através das fronteiras das redes existentes. Ainda segundo a pesquisa de KARAM (2006), mobilidade inclui a capacidade de usar uma determinada aplicação ou serviço, sem a interrupção de sua atividade, através de várias tecnologias, permitindo o movimento entre pontos de acesso com ou sem fio ITU (2004).

3 A FERRAMENTA DE MAPEAMENTO

*“Freqüentemente expectativas falham e estas falhas são mais
freqüentes nas expectativas mais prometidas”*

WILLIAM SHAKESPEARE

A atividade de elicitación de requisitos corresponde à etapa de compreensão do problema aplicada ao desenvolvimento de software BEZERRA (2007). O principal objetivo desta etapa é que usuários e desenvolvedores e demais interessados tenham um entendimento comum do problema a ser resolvido.

Durante esta etapa, a equipe de desenvolvedores tenta conhecer e entender o domínio do problema que deve ser considerado no sistema, suas prioridades e o que eventualmente deve ser automatizado.

Formalmente, um requisito é uma condição ou capacidade que deve ser alcançada ou possuída por um sistema ou componente deste, para satisfazer um contrato ou outros documentos formalmente impostos MACKIASZEK (2000).

Um requisito é uma condição ou capacidade que o sistema deve alcançar ou possuir KRUCHTEN (2000).

A fase de elicitación de requisitos compreende um estudo investigatório minucioso das necessidades dos usuários e da situação de um eventual sistema existente. Conforme a norma IEEE-830 (1998) existem várias técnicas utilizadas para se efetuar este levantamento e alcançar um entendimento entre usuários e desenvolvedores: leitura de obras de referência, observação do ambiente do usuário, entrevistas com especialistas no domínio entre outras. Desta forma, baseado na imprecisão dos resultados das tarefas que podem compor a fase de elicitación de requisitos, pode-se dizer também que:

Um requisito é uma expectativa ou demanda, declarada ou não, que precisa ser identificada, entendida e atendida.

3.1 O DOCUMENTO DE REQUISITOS

O produto da fase de eliciação de requisitos é o documento de requisitos. A norma IEEE-830 se refere a este documento como sendo um *Software Requirements Specifications* (SRS) ou Especificação de Exigências de Software. Segundo BEZERRA (2007), esse documento deve ser de fácil compreensão tanto para desenvolvedores quanto para usuários e demais interessados.

A IEEE-830 (1998) propõe dezessete normas complementares para se produzir um SRS, tais como: guia para desenvolvimento de protótipos, terminologia, plano de garantia de qualidade de software, plano de gerência de configuração de software, dicionário de métricas para confiabilidade de software, modelo de verificação e validação de software, entre outros. As seções de um SRS ou segundo BEZERRA (2007), Documento de Requisitos são descritas a seguir:

- **Requisitos funcionais:** definem as funcionalidades do sistema. Exemplo:
 - a) “O sistema deve permitir que um aluno realize sua matrícula nas disciplinas oferecidas em um semestre letivo.”
 - b) O sistema deve considerar todos os encargos salariais dos funcionários, terceirizados ou não, no preço final dos produtos.
- **Requisitos não-funcionais:** declara as características de qualidade que o sistema deve possuir e que estão relacionadas às suas funcionalidades. Exemplo:
 - c) Confiabilidade: Corresponde a medidas quantitativas da confiabilidade do sistema como tempo médio entre falhas, recuperação de falhas ou quantidade de erros por milhares de linhas de código-fonte.
 - d) Desempenho: requisitos que definem tempos de resposta esperados para as funcionalidades do sistema.

- e) Portabilidade: restrições sobre as plataformas de hardware e software nas quais o sistema será implantado.e também é relativo ao grau de facilidade para transportar o sistema entre plataformas.
 - f) Segurança: Limitações de segurança do sistema relativas a acessos não-autorizados.
 - g) Usabilidade: requisitos que afetam ou que se relacionam com a Usabilidade do sistema facilidade de uso ou necessidade de treinamento dos usuários.
- Requisitos normativos: Restrições relativas a custos e prazos, plataforma tecnológica, aspectos legais, interoperabilidade com sistemas legados, etc.
Exemplo:
 - a) Uma tela do tipo *touch-screen* que reconheça toques simultâneos.

A fase de eliciação de requisitos, no contexto da engenharia de software é concernente às atividades que contribuem para a produção de um SRS e sua manutenção ao longo do tempo, visto que requisitos são voláteis ao longo do tempo. PRESSMAN (2005) considera esta fase para os seguintes aspectos: Ajuda engenheiros de software, stakeholders e usuários a terem um melhor entendimento do problema, tenta garantir que usuários e desenvolvedores tenham o mesmo entendimento do problema e como produto final apresenta por escrito todas as partes do problema como lista de cenários, características e funcionalidades. Em PRESSMAN (2005) é apresentado um roteiro sobre aspectos pontuais de validação de requisitos, que são descritos a seguir:

- Cada requisito é consistente com os objetivos globais do projeto?
- Todos os requisitos foram especificados no nível apropriado de abstração?
- O requisito é realmente necessário ou representa característica que pode não ser essencial?
- Todos os requisitos são não-ambíguos?
- Algum requisito é conflitante?
- O requisito é compatível com o ambiente tecnológico?

- Se implementado, é possível ser testado?
- O modelo de requisitos reflete a informação, funcionalidade e comportamento do sistema?
- O requisito foi particionado de forma a expor mais informações detalhadas do sistema?

Uma das formas de se medir a qualidade de um sistema é pela sua utilidade. Um sistema é útil quando atende as expectativas do usuário e demais interessados.

A seguir são apresentados os requisitos que são considerados no desenvolvimento do simulador de QoS (QoS Simulator - QoSS)

3.1.1 Requisitos Funcionais

Em um primeiro nível de abstração, os seguintes requisitos funcionais foram identificados seguindo os preceitos do *framework* da Figura-1:

- A ferramenta deve ser sensível à percepção do usuário no que concerne à identificação dos requisitos de QoS de um serviço, e permitir ao usuário final interagir de forma transparente com um processo de simulação destes requisitos.
- A ferramenta deve apresentar um relatório contendo os valores dos parâmetros técnicos de QoS identificados no processo de simulação para o estabelecimento do USLA e parametrização personalizada do serviço.
- A visão do usuário sobre o processo de simulação deve ser apenas sobre a escolha do serviço, escolha do dispositivo, qualidade percebida e aspectos contratuais.

A seguir são listados requisitos funcionais que estão em um nível de detalhamento maior e que derivam dos requisitos apresentados acima:

- O usuário pode avaliar mais de um dispositivo de acesso em uma mesma sessão.

- O processo de simulação deve ter disponível dados concernentes aos valores mínimos e máximos dos parâmetros de QoS estabelecidos pelo Provedor de Serviço através de modelo oferecido pelo sistema.
- O usuário tem que saber quando a simulação inicia e quando termina.
- O usuário inexperiente deve expressar seus anseios de qualidade através de controle visual simples e intuitivo e estabelecer os valores dos requisitos de QoS em conjunto.
- O usuário experiente deve poder simular os requisitos de QoS individualmente.
- O simulador deve ser capaz de analisar o fluxo do serviço e reconhecer tecnologias de rede diferentes e instanciar as classes de tratamento correspondentes.
- O simulador tem que ser capaz de gerar atrasos, *jitter* e perdas com precisão baseada na granularidade desses requisitos.
- A sessão do usuário com o serviço deve ser registrada para fins contratuais.
- O simulador tem que ser flexível no que tange às tecnologias que devem ser abstraídos, ou seja, determinada tecnologia pode ser avaliada de forma independente. Os níveis de flexibilidade devem ser informados pelo provedor de serviço.
- Os valores dos requisitos sendo simulados têm que embutir uma eventual distância de simulação ao serem apresentados.
- A simulação de pacotes chegando por caminhos diferentes deve ser tratada no processo de simulação.

3.1.2 Requisitos não-Funcionais

- O simulador deve ser desenvolvido em linguagem capaz de manusear processos simultâneos e APIs relativas às diversas tecnologias de rede demandadas pelo serviço.
- O simulador, ao efetuar suas tarefas, não deve prejudicar o desempenho do serviço sendo avaliado.

3.1.3 Requisitos Normativos

- A interface do simulador deve ser capaz de gerar os valores dos parâmetros sendo simulados de forma simultânea e assíncrona.
- O simulador deve ser compatível e portátil para qualquer plataforma operacional fixa, móvel ou embarcada.

As próximas seções apresentam aspectos concernentes as fases de análise do domínio e análise da aplicação conforme preceituado em MELNIKOFF (2007).

3.2 ANÁLISE DO DOMÍNIO

A fase de análise do domínio também é conhecida como fase de Modelagem do Negócio ou Modelagem dos processos do Negócio. A etapa de Análise de Domínio visa capturar o conhecimento geral sobre uma aplicação. Este conhecimento é advindo de especialistas no domínio e é relativo a conceitos, relacionamentos e processos ou regras do negócio BEZERRA (2007). Através de um enunciado do problema, os aspectos relativos ao mundo real do sistema precisam ser entendidos e suas principais características abstraídas em um modelo MELNIKOFF (2007).

Um domínio é uma área do conhecimento ou de atividade específica caracterizada por conceitos e terminologia compreendidos por especialista na área BEZERRA (2007).

3.2.1 Aspectos Gerais sobre o Domínio do Problema

O estabelecimento de contratos de níveis de serviço personalizados entre uma organização provedora de serviços e um usuário final ainda não é muito praticado. Com isso, as partes não têm um entendimento claro sobre responsabilidades e do nível de QoS adequado e contratado SERRA (2007).

Um contrato personalizado entre o usuário final e a Organização Provedora de Serviços pressupõe que aspectos individuais de Qualidade de Serviço devam ser considerados na prestação dos serviços. Desta forma, o usuário passa a lidar com questões técnicas como avaliação de valores mínimos e máximos de parâmetros de qualidade de serviço de forma a expressar seus anseios sobre qualidade baseado na sua experiência e percepção. É real e observável, no entanto, que na maioria dos casos o usuário prefere lidar com parâmetros de qualidade de serviço menos técnicos para se expressar como clareza e nitidez.

Infelizmente, os valores destes parâmetros subjetivos serão de difícil medição caso o usuário não disponha de um mecanismo que o auxilie na identificação destes valores.

SERRA (2007) coloca a seguinte observação na cláusula 2.1 do USLA que a usuária hipotética Maria deseja firmar com um provedor de serviços:

“Qualidade de áudio, vídeo e imagem clara e nítida”.

Sob o ponto de vista da Maria, esta observação, feita por ela mesma, apresenta o que ela deseja sobre qualidade de áudio, vídeo e imagem, e, provavelmente, é tudo que ela é capaz de dizer sobre a qualidade que deseja.

Se o eventual grau de vista cansada da Maria piorar, ela vai telefonar para o Provedor de Serviços alegando que eles não estão cumprindo o contrato e vai exigir a multa, pois, para ela, a imagem não está mais nítida.

A forma como foi tratado o cenário da Maria serve para definir um dos papéis do Simulador de QoS proposto neste trabalho.

A Maria não vai precisar saber mais nada sobre a sua própria percepção de áudio, vídeo e imagem. A diferença agora é que a clareza e nitidez da Maria passam a ter valores mensuráveis e passíveis de serem aplicados no USLA.

O Simulador de QoS também transforma aspectos
subjetivos em aspectos mensuráveis aplicáveis ao USLA..

Desta forma, o usuário e o provedor de serviços passam a ter um entendimento claro sobre responsabilidades, sobre a QoS adequada e qual a QoS contratada.

3.2.2 Diagrama de Casos de Uso (DCU)

Como dito anteriormente, a visão do usuário sobre o processo de simulação do serviço deve ser apenas para aspectos como escolha do serviço, escolha do Provedor de Serviços, escolha do dispositivo de acesso, qualidade percebida do serviço e questões contratuais.

A Figura-3 mostra, através de um DCU (Diagrama de Casos de Uso), a visão que o usuário tem dos processos relativos a simulação do serviço, junto a um determinado Provedor de Serviços.

Note que todos os casos de uso do DCU compõem a ferramenta de mapeamento e que apenas os casos de uso “Avaliar Qualidade de Serviço e Preço” e “Gerar Requisitos USLA” compõem o simulador de QoS (QoS Simulator - QoSS) desenvolvido pelo autor para ser aplicado no Estudo de Caso deste trabalho.

Figura-3 Visão do usuário do Processo de Simulação do Serviço

3.2.3 Diagrama de Classes de Domínio

Em um processo de desenvolvimento baseado no paradigma da orientação a objeto, um dos resultados da análise é o modelo de classes que evolui e recebe mais detalhes na medida em que as interações acontecem e o processo de desenvolvimento avança (KRUCHTEN, 2000). Existem três estágios sucessivos de abstração pelos quais o modelo de classes se submete: classes de análise ou de domínio, classes de especificação ou de projeto e classes de implementação.

O modelo de classes de domínio mostra a estrutura estática do sistema do mundo real e a organiza em partes utilizáveis. Desta forma, o modelo de domínio descreve as classes do mundo real e suas relações MELNIKOFF (2007).

O domínio do conhecimento ou de atividade a ser considerado neste trabalho é apresentado no diagrama de classes de análise da Figura-4 (ou classes de domínio). O diagrama apresenta o domínio da ferramenta de mapeamento no contexto do *framework* da Figura-1.

DIAGRAMA DE CLASSES DO DOMÍNIO

Figura-4 Diagrama de Classes de Domínio ou Classes de Análise

Note que o DCU da Figura-3 está em um nível diferente de abstração em relação ao Diagrama de Classes de Domínio. O DCU foi construído considerando uma instância da classe Provedor de Serviços e dá ênfase as interações do usuário e do provedor de serviços com os processos que compõem a ferramenta de mapeamento. Vale

observar que as classes concernentes ao simulador de QoS (QoS Simulator - QoSS) estão num nível de detalhamento maior e podem ser consideradas como classes de especificação.

3.2.4 Diagrama de Atividades

A Figura-5 apresenta o diagrama de atividades da Ferramenta de Mapeamento cujo domínio engloba os processos de “Geração do Ambiente de Simulação” e “Simulação do Serviço” da Figura-1.

Figura-5 Diagrama de Atividades

Note que as atividades contidos no diagrama são os mesmos processos com os quais o usuário final interage no DCU da Figura-3, sendo que o processo “Avaliar Qualidade de Serviço e Preço”, foi desmembrados em quatro atividades.

A descrição de cada procedimento é apresentada a seguir:

Selecionar Serviço: O usuário seleciona a composição de serviços que deseja avaliar e é estimulado a tecer alguns comentários relativos a disponibilidade e confiabilidade do serviço.

Selecionar Provedor de Serviços: O usuário seleciona um provedor de serviço habilitado a oferecer a composição de serviços selecionada.

Selecionar Dispositivo: Esta seleção é feita pelo usuário mas deve ser, a princípio, uma sugestão do Provedor de Serviços baseada nos Serviços que o usuário deseja utilizar.

Avaliar QoS: É um processo transparente de simulação de requisitos técnicos de QoS que são definidos sob a ótica do usuário final e aplicados artificialmente no fluxo do serviço para avaliação da qualidade percebida.

Neste caso, o sistema define os requisitos técnicos de QoS que serão estabelecidos no USLA e que vão nortear o estabelecimento de SLAs entre organizações provedoras de serviço sem que o usuário final tome conhecimento desses fatos.

Avaliar \$\$\$: Este procedimento é uma decisão do usuário concernente ao preço do serviço cuja qualidade percebida está sendo avaliada.

Piorar QoS: Este procedimento representa uma piora na qualidade e conduz o processo de volta ao procedimento de **Avaliar QoS**. Vale lembrar que piorar o nível de QoS do serviço nem sempre acarreta uma diminuição do preço a ser pago pelo serviço.

Customizar: Orientado pelo Provedor de Serviço, o usuário pode customizar aspectos como resolução de vídeo ou se utilizar de alguma outra configuração disponível no Dispositivo de forma a adequar o Serviço sendo avaliado ao Dispositivo existente.

Melhor Esforço: Este procedimento representa o caso do usuário estar satisfeito com a política *Best Effort* do Provedor de Serviços e nenhuma Qualidade personalizada é requerida.

3.3 ANÁLISE DA APLICAÇÃO E DECISÕES DE PROJETO

A fase de análise da aplicação tem como objetivo identificar objetos de análise que, normalmente, não fazem sentido para os especialistas do domínio, mas que são necessários para suprir as funcionalidades do sistema em questão BEZERRA (2007).

O principal produto desta fase é se identificar a porção do domínio que fará parte do sistema a ser desenvolvido ou automatizado.

Pelo fato dos artefatos da fase de análise da aplicação não fazerem muito sentido para os especialistas no domínio, esta fase se torna intimamente ligada à fase de Projeto ou mais precisamente às Decisões de Projeto, e por este motivo, estas fases são discutidas na mesma seção.

A fase de Projeto se caracteriza por determinar como o sistema funcionará para atender aos requisitos de acordo com os recursos tecnológicos existentes ou disponíveis.

3.3.1 MODELO ARQUITETURAL DO QoS SIMULATOR - QoSS

“Arquitetura é tudo aquilo que sobra quando você descarta tudo que puder descartar e ainda assim consegue manter um bom entendimento do sistema”

Kruchten (2000)

A simulação do atraso ou do *jitter* de um pacote implica em interferir no tráfego deste pacote impedindo que este trafegue por um determinado período de tempo, e este tempo, deve ser definido baseado na percepção do usuário final. Neste contexto, surge a seguinte questão:

Quais critérios adotar para se identificar o melhor momento e a melhor técnica para capturar o pacote para fins de simulação do atraso considerando todo o caminho que deve ser percorrido desde sua saída na origem até sua chegada ao destino.

Wireshark:

O Wireshark é um *lan analyser* poderoso (disponibilizado via GNU *General Public License*) que pode capturar pacotes de muitos tipos de tráfegos diferentes, e a despeito do seu nome, redes sem fio também.

Os tipos de mídia suportados pelo Wireshark dependem de uma serie de fatores conforme <http://wiki.wireshark.org/CaptureSetup/NetworkMedia> (RAMIREZ, 2010). Para ser aplicado neste trabalho, esta capacidade de reconhecer o tráfego da rede, no entanto não é suficiente visto que o produto não se propõe a interferir no tráfego de pacotes como descartar ou gerar atrasos artificialmente. No entanto, sob o ponto de vista de seus principais usuários (administradores de rede, engenheiros de segurança, etc.) esta conduta é, a princípio, a mais adequada.

Por ser um produto de código aberto e por sua capacidade de entender qualquer bit que tenha alguma semântica no tráfego da rede, uma eventual continuação desta pesquisa deve considerar dotar o WireShark com os requisitos de simulação identificados para o QoS. Entretanto, interferir artificialmente no tráfego de um pacote é muito diferente de apenas atuar como um *sniffer*. Muitos aspectos inerentes a segurança interna em sistemas operacionais teriam que ser abordados.

Um aspecto importante é a possibilidade (não confirmada) de que um mesmo processo poder capturar pacotes oriundos de tecnologias diferentes e tratá-los como um fluxo único, posto que, é sobre o fluxo do serviço (fluxo heterogêneo) que os requisitos de QoS são definidos sob a ótica do usuário final.

Ns2:

Outro produto conhecido que é disponibilizado via GNU (GPL) é o Ns-2. O Ns-2 é um simulador de eventos focado em pesquisa na área de redes de computadores.

Não foram encontrados, entretanto, na sua documentação (<http://nsnam.isi.edu/>) os aspectos discutidos aqui, ou seja, simulação de requisitos de QoS baseado na percepção do usuário final. E por mais que uma adaptação seja possível, estar-se-ia trazendo embutidos no Ns-2 uma série de funcionalidades e um grau de complexidade desnecessários posto que o que se quer discutir é a adequabilidade do que é proposto aqui, e como resultado da discussão, espera-se encontrar tão somente os requisitos específicos para um simulador de QoS específico, ainda que, posteriormente, estas funcionalidades sejam incorporadas por algum simulador genérico.

QoS Simulator - QoSS:

Uma outra alternativa seria desenvolver um simulador para capturar os pacotes na origem e/ou no destino das conexões, posto que, sob a ótica do usuário final, só interessa capturar os pacotes cujos *pay-loads* sejam endereçados às aplicações. Isto porque são estes pacotes (ou a ausência deles) que influenciam a qualidade percebida pelo usuário.

Assim, a captura e simulação poderiam ser efetuadas o mais próximo possível das aplicações que geram e que consomem os pacotes. Desta forma, uma instância do simulador deveria ser instalada em mais de um dispositivo gerando processamento adicional o que aumenta o “custo” de simulação dos requisitos de QoS do serviço.

Considerando a pluralidade de dispositivos disponíveis no mercado e o ambiente heterogêneo proposto na dissertação, surgem outras questões com relação à topologia do modelo arquitetural:

- Não há garantias de que sempre haverá recursos no dispositivo do usuário para hospedar o simulador de QoS;
- Algumas aplicações que compõem o serviço podem ser proprietárias e possuírem um ambiente operacional exclusivo dificultando a instalação do simulador nos dispositivos que participam do serviço;

- O ambiente de simulação deve considerar a pluralidade de dispositivos visto que, além do dispositivo do usuário, alguns componentes do serviço também podem estar hospedados em dispositivos móveis com pouca capacidade e com ambiente operacional proprietário;

Neste novo contexto, os seguintes aspectos arquiteturais foram adotados:

- O ambiente de simulação passa a considerar esta pluralidade e tanto os dispositivos quanto as aplicações passam a operar nos seus ambientes originais que são disponibilizados em rede isolada como mostra a Figura-6.
- Uma única instância do simulador de QoS captura e simula todo o fluxo de pacotes do serviço independente de tecnologia de rede.
- O simulador opera em uma máquina exclusiva na rede do ambiente de simulação conforme Figura-6.

Figura-6 Topologia da Rede Isolada do Ambiente de Simulação

O modelo arquitetural da Figura-7 dá ênfase a um paralelo entre os aspectos arquiteturais de um serviço genérico operando em ambiente heterogêneo, e seu correspondente ambiente de simulação dentro do QoSS, tornando transparente para o usuário final todas as tecnologias de rede. Essencialmente, o QoSS intercepta, simula, apresenta e re-roteia o fluxo de pacotes de um serviço, diferentemente da versão 1.0.2 (SVN Rev 25698) do Wireshark que apenas intercepta e apresenta.

Tendo em consideração todo o caminho percorrido por um fluxo de dados, o terminal do usuário também deve ser considerado como um dos elementos da rede (MAIA, 2009). Assim, isto também sugere que todo processamento dentro da máquina do usuário anterior as camadas orientadas à aplicação do modelo de referência OSI, também deve ser considerado como atraso de rede.

Hillebrand et. al (2004) sugere que cada tecnologia de rede que compõe uma infraestrutura heterogênea possui melhores condições de manipular um determinado tipo de tráfego, e então, seria interessante para um terminal com múltiplos serviços trabalhar com diversas redes simultaneamente, sendo que seus mecanismos de controle deveriam utilizar para cada serviço, a rede mais apropriada. Isto sugere que todo o fluxo heterogêneo do serviço deva convergir para algum ponto onde possa ser tratado em conjunto, ou seja, em alguma das camadas (OSI) orientadas a aplicação.

Figura-7 Modelo Arquitetural do Ambiente de Simulação

Com isso, estabelece-se as bases para um mecanismo que subsidia a adaptação dinâmica de QoS entre as diferentes tecnologias de rede de forma transparente para

o usuário, com valores de referência definidos através de um processo de identificação de requisitos de QoS sob a ótica do usuário final.

Uma descrição de cada conexão é enviada para o QoSS por um arquivo do tipo XML chamado DDS (Documento e Descrição do Serviço) que é gerado pelo “Gerador do Ambiente de Simulação” apresentado na Figura-1.

Entre outras informações, o QoSS precisa saber a origem e o destino das conexões de modo a poder interceptá-las. Desta forma, o DDS inclui o tipo de tecnologia de cada conexão (por exemplo: IP, GPRS, DVB-H, etc.), sua origem e destino, protocolo de comunicação, propriedades da API da conexão, o tipo do dispositivo do usuário, a granularidade dos valores dos requisitos de QoS entre outros.

Com as informações do DDS e o conseqüente controle das conexões o QoSS pode manipular os pacotes e simular atrasos, *jitter*, perdas e calcular e determinar a banda percebida.

Assim, o QoSS garante que apenas os pacotes das conexões definidas no DDS podem trafegar pela rede isolada do ambiente de simulação cuja topologia genérica é apresentada na Figura-6.

A Figura-7 também mostra os principais elementos de software que implementam os processos de captura, simulação e re-roteamento dos pacotes como *Threads*, *APIs*, *Datagrams*, *Stream*, *Buffers* entre outros.

Como já abordado anteriormente, o ponto de simulação da Figura-7 representa o instante que os pacotes são manuseados pelos métodos de simulação.

```
// Método de simulação SetDelay
public static void setDelay (long t) throws IOException {

    long Delay = System.nanoTime() + t;

    // Atrasa t nano segundos
    while(System.nanoTime() < Delay) { }
}
```

Note na Figura-7 que o pacote corrente permanece armazenado no *buffer* durante o processo de simulação e a mesma estrutura de *buffer* é utilizada para re-enviá-lo.

O processo de simulação é composto por quatro atividades distintas: 1) Captura³ dos fluxos entre o dispositivo do usuário e o serviço; 2) Simulação dos requisitos de QoS atraso, *Jitter* e perdas via controle deslizante e cálculo da banda percebida; 3) Entrega dos fluxos simulados ao destinatário; 4) Apresentação dos requisitos de QoS sendo simulados.

O principal aspecto considerado no processo de simulação é a necessidade de se efetuar estas quatro atividades simultaneamente e de forma assíncrona entre si, para cada pacote injetado na rede pelas aplicações envolvidas no serviço. Estes procedimentos são inseridos num laço “eterno” e disparados sempre que QoSS percebe a chegada de um novo pacote. Sob o ponto de vista do dispositivo do usuário, o QoSS opera como um “*proxy reverso*” por onde o serviço é disponibilizado e para onde convergem todas as conexões.

3.3.2 O AMBIENTE DE SIMULAÇÃO

O modelo da ferramenta preceitua que várias aplicações se utilizando de tecnologias de rede diferentes podem estar sendo executadas simultaneamente e de forma assíncrona através do dispositivo do usuário, e gerando tráfego nos dois sentidos. O fluxo de todas as aplicações em conjunto representa o fluxo do serviço e é sobre o fluxo do serviço que os requisitos de QoS são simulados como mostram a Figura-6 e a Figura-7.

Basicamente, o ambiente de simulação é composto por sua descrição via DDS, que é aplicada pelo QoSS, e uma infra-estrutura heterogênea isolada.

³ O QoSS captura os pacotes considerando o tamanho do *buffer* sugerido pelo sistema operacional.

```
byte[] buf = new byte[socket.getReceiveBufferSize()];
packet = new DatagramPacket(buf, buf.length);
socket.receive(packet);
```

Dentro do ambiente de simulação, os dispositivos, as aplicações e as tecnologias de rede operam juntas e harmonicamente, de forma a oferecer ao usuário a possibilidade de perceber suas preferências no mundo real.

3.3.3 O OCULISTA

Com a finalidade de oferecer maior transparência para o usuário e escalabilidade ao ambiente de simulação, a sessão de simulação segue o mesmo processo tradicional usado pelos oculistas na prescrição de lentes, onde o oculista é quem opera o equipamento ótico (o QoSS neste trabalho) para identificar o grau adequado das lentes (neste trabalho representadas pelos controles deslizantes atraso, *jitter* e perdas de pacotes como mostra a Figura-8) e o paciente (neste trabalho representado pelo usuário final) somente informa aspectos como clareza, nitidez, etc. (neste trabalho representado pela qualidade percebida do serviço sendo simulado).

Desta forma, o oculista é o verdadeiro operador do QoSS como mostram as Figuras 6 e 7, e é considerado como um elemento de interface.

3.3.4 A INTERFACE DO QoS SIMULATOR (QoSS)

MANDEL (1997) aponta três regras básicas para um projeto de interface e se refere a estas regras como sendo *The Golden Rules*.

- Manter o usuário no controle das ações.
- Reduzir o esforço de memória e a carga de trabalho do usuário.
- Organizar de forma consistente os componentes da interface.

Na Figura-8, é apresentado a interface gráfica do usuário na qual um técnico do provedor de serviços (oculista) executa a simulação dos requisitos de QoS com um simples mover de mouse sobre os controles deslizantes A, J e P.

Os controles deslizantes A, J e P (lentes) simulam respectivamente atrasos, *jitter* e perda de pacotes individualmente e os valores gerados pelos controles deslizantes

(quando movidos para cima ou para baixo) são aplicados de forma assíncrona e simultânea no fluxo de pacotes do serviço.

Figura-8 Controles deslizantes da interface do QoSS

Internamente, o QoSS simula atrasos, *jitter* e perdas com uma granularidade pré-definida pelo provedor de serviços e aplicada nos controles deslizantes. Entenda-se como granularidade de um parâmetro, o menor valor que o parâmetro pode variar.

Desta forma, o “oculista” pode aplicar qualquer combinação de requisitos de QoS no fluxo de pacotes do serviço baseado na sua experiência no manuseio das “lentes” (controles deslizantes). O controle deslizante AJP tem a finalidade de reprisar o

comportamento do oculista ao manusear os controles deslizantes durante uma sessão do usuário.

3.3.5 DIAGRAMA DE SEQÜÊNCIA

Por conta da imprevisibilidade do comportamento do usuário, o QoSS efetua de forma assíncrona e simultânea as atividades de captura do fluxo real de pacotes do serviço, simulação dos requisitos de QoS via controles deslizantes A, J e P, entrega dos fluxos de pacotes simulados para a aplicação e registro e apresentação do novo comportamento da rede em função dos requisitos de QoS sendo simulados. Estes procedimentos por serem assíncronos e simultâneos entre si são implementados via *threads* (WONG, 1999) e inseridos num laço “eterno” tipo *while true* e disparados sempre que um pacote é injetado na rede pelo serviço.

Um datagrama é uma mensagem independente enviada através da rede cuja chegada, ordem de chegada, tempo de chegada e conteúdo não são garantidos. (TANEMBAUM, 2003). O *DatagramPacket* e *DatagramSocket* são classes do pacote java.net que implementam a comunicação de datagramas em sistemas independentes usando UDP.

Por exemplo, para simular pacotes ou datagramas chegando fora de ordem tem-se que instanciar uma *thread* para reenviar para a aplicação cada *DatagramPacket* e gerar atrasos diferenciados para cada *thread*. O mesmo se aplica às perdas. Ficará a critério da aplicação aguardar por estes pacotes ou não. Na Figura-9, a *thread* que trata o pacote 5 foi encerrada para simular uma perda e os atrasos gerados para as *threads* que tratam os pacotes 2,3 e 4 foram diferenciados no sentido de simular chegadas por caminhos diferentes. O pacote 2 será provavelmente descartado pela aplicação por espera prolongada, no caso o VLC MEDIA PLAYER.

Figura-9 Simulação de Entrega de Pacotes fora de ordem para a Aplicação

O Diagrama de Seqüência é apresentado na Figura-10. No diagrama, na linha da vida do oculista, o laço dos processos A, J e P representa seus respectivos controles deslizantes. Note que somente as chamadas para q:QoSS, T1:Presentation and DDS_Connections não são assíncronas. Observe no diagrama, que os métodos de simulação T3:setDelay(A) e T3:setLoss(P) são executados antes do pacote ser re-enviado para a aplicação (App).

Figura-10 Diagrama de Seqüência

O atraso (A) é alterado de forma assíncrona pelo Oculista através do controle deslizante A e pela thread T3:Connection ao aplicar o *jitter* (J) no atraso ao longo do tempo. O valor do *jitter* só é definido e alterado pelo controle deslizante J.

```
// Simula atraso e jitter.
    setDelay(this.atraso);
    this.atraso = this.atraso + this.jitter;
```

Note, no trecho de código acima, que o primeiro pacote sempre recebe o valor corrente do controle deslizante A e o *jitter* só é aplicado a partir do segundo pacote em diante. Se o atraso atinge valores muito altos pela aplicação constante do *jitter*, o oculista pode redimensioná-lo através do controle deslizante A ou diminuir o *jitter* através do controle deslizante J. Todas estas operações são assíncronas com relação a thread T3:Connection e efetuadas em tempo de simulação.

3.3.6 AMBIENTE OPERACIONAL

O modelo do QoSS preceitua que várias aplicações se utilizando de tecnologias de rede diferentes podem estar sendo executadas simultaneamente e de forma assíncrona através do dispositivo do usuário e assim, gerando conexões que precisam ser “desviadas” para serem capturadas pelo QoSS.

Existem muitas maneiras ou mecanismos para se estabelecer o destino das conexões de um serviço. O mais conhecido deles é utilizar um serviço do tipo DNS (*Domain Name System*) onde através do nome do domínio o serviço informa o endereço IP. Existem também muitos outros mecanismos como, por exemplo, definir rota default em roteadores, serviço de *proxy* ou *proxy* reverso, ou até mesmo, manualmente, para citar apenas alguns. Estes mecanismos, no entanto, não são investigados separadamente visto que qualquer um deles é capaz de fazer o fluxo de rede do ambiente de simulação convergir para o QoSS. O único aspecto relativo ao ambiente operacional que é investigado com mais detalhes neste trabalho é um comparativo entre a plataforma de construção de software JAVA e a Linguagem C++ no que tange à implementação da prova de conceito e de uma eventual versão corporativa do QoSS. Os aspectos que são abordados no comparativo são simplicidade, desempenho, recursos das APIs, portabilidade e *multi-threading*. O

comparativo considera as publicações de REILLY (2002), WONG (1999) e C++ (2010).

3.3.6.1 A Escolha do Ambiente

A seguir, algumas considerações que levaram a escolha do ambiente.

“O C++ é uma linguagem de programação multiparadigma e de uso geral. A linguagem é considerada de médio nível, pois combina características de linguagens de alto e baixo níveis. Desde os anos 1990 é uma das linguagens comerciais mais populares, sendo bastante usada também na academia por seu grande desempenho e base de utilizadores”. Para exemplificar a completude da linguagem, a seguir é apresentado alguns aplicativos desenvolvidos em C++.

- Adobe Acrobat;
- Microsoft Office;
- MAC OS X;
- Microsoft Windows;
- Wireshark

A principal característica da plataforma JAVA hoje é que seu sistema é capaz de funcionar em uma ampla variedade de plataformas sem a necessidade de re-compilação do software ou modificação do código-fonte.

Para programadores intencionados em escrever software para uma única plataforma isto não parece ser muito útil. No entanto, para desenvolvedores de software comercial isto representa uma grande diminuição de custos e economia de tempo visto que o software pode ser escrito para um único ambiente.

Para aplicações que se utilizam da rede, esta característica é ainda mais atraente. Embora o C++ possa ser utilizado nas plataformas UNIX e Wintel, existem muitas diferenças para as chamadas de sistema para estes recursos inclusive diferenças entre variantes do sistema operacional UNIX.

Naturalmente, esta portabilidade tem um “custo”. O código-fonte em Java é compilado em *bytecode* que é executado pela JVM (Java Virtual Machine). A JVM é uma emulação de um dispositivo de hardware e é freqüentemente usada para se recriar um ambiente operacional em outro. Enquanto que a JVM é capaz de executar o Java *bytecode*, ela não é capaz de executar a si própria. Usualmente, a JVM é hospedada dentro de um Java Runtime Environment (JRE) que é responsável por executar a JVM. Existem muitos tipos de JREs disponibilizadas por fabricantes diferentes. Suas principais categorias são: *Java 2 Platform, Standar Edition (J2SE)*, *Java 2 Platform, Enterprise Edition (J2EE)*, *Java 2 Platform, Micro Edition (J2ME)* entre outras.

Tudo isto significa que o código Java não tem o mesmo desempenho que um código nativo que é compilado em linguagem de máquina como o C++. Algumas tentativas foram feitas para melhorar desempenho de um software em Java como, por exemplo, ***just-in-time*** (JIT) que converte Java *bytecode* em código nativo. Entretanto, este código não é tão rápido quando comparado com C++ além de consumir mais espaço em memória.

Linguagens orientadas a objeto podem fazer do desenvolvimento de software uma tarefa mais fácil, entretanto, a orientação a objeto por si só não é suficiente. Algumas linguagens orientadas a objeto, como o C++, são conhecidas por sua complexidade. Características como acesso direto a memória através de ponteiros, necessidade dos programadores de explicitamente alocar e desalocar regiões de memória e herança múltipla fazem do C++ uma linguagem complexa.

Embora o JAVA compartilhe da mesma herança do C++, ele é de longe uma linguagem mais simples de aprender. Em Java, por exemplo, não existem ponteiros para se ter acesso à memória. Somente através da referencia ao objeto, um programador pode acessar outro objeto. Além disso, herança múltipla não é permitida em Java, ou seja, enquanto a classe mantiver um vínculo de herança com uma classe ela não pode se vincular a outra. Tudo isso oferece simplicidade ao código o que é importante para qualquer aplicação e em particular para aplicações que se envolvam em arquitetura em redes.

Simplicidade é uma vantagem relativa. Java é mais simples do que a maioria das linguagens orientadas a objeto e, no entanto, muitos programadores ainda vão achá-la complexa por causa de seus recursos poderosos.

Programadores oriundos de linguagens como C ou Perl podem ter se envolvido com o conceito de múltiplos processos. Em sistemas operacionais como o Unix, os processos eram bastante utilizados. Naturalmente, isso implicava em sobrecarga de recursos, como a memória para a aplicação do código e variáveis que eram duplicadas em cada processo. Uma alternativa bem mais eficiente é a programação *multi-threaded*. A linguagem que suporta *multi-thread* suporta processamento simultâneo e assíncrono, mas com memória compartilhada para o código e para os dados. Isso permite que as *threads* conservem a memória e interajam umas com as outras e trabalhem de forma colaborativa, se necessário.

Embora seja possível escrever aplicativos triviais com arquitetura cliente-servidor sem utilizar recursos de *multi-threading*, mesmo um servidor moderadamente complexo normalmente se utiliza da técnica de *multi-threading*.

Outras linguagens também dão suporte aos recursos de *multi-threading* mas Java foi projetada desde o início para este tipo de programação e provê um conjunto de palavras-chave para simplificar a escrita de processos *multi-threaded*.

Uma aplicação que precisa atender e se adaptar ao comportamento imprevisível do usuário têm que estar preparada para instanciar processos simultâneos e assíncronos, ou seja, processos *multi-threaded*.

Assim, por possuir ferramental adequado para sistemas com arquitetura em redes, oferecer portabilidade na construção do software, contar com recursos em *multi-threading* e principalmente oferecer simplicidade na escrita do código, a Plataforma JAVA foi escolhida para a implementação da versão prova-de-conceito do QoSS.

3.4 IMPLEMENTAÇÃO - Principais Aspectos do Código-Fonte

Na fase de implementação, ocorre a tradução da descrição computacional obtida na fase de projeto em código executável mediante uma ou mais linguagens de programação BEZERRA (2007).

O Modelo de Classes de Implementação é um detalhamento do modelo de classes de especificação. Esse modelo corresponde à implementação das classes em alguma linguagem de programação. Segundo BEZERRA (2007), o próprio código-fonte do software se constitui no modelo de classes de implementação, ou seja, o modelo de classes evolui desde a fase de levantamento do domínio do problema até o código-fonte da aplicação.

A seguir, os aspectos principais do código-fonte do QoSS.

3.4.1 Captura, simula, incrementa banda percebida e re-envia pacote

A classe QoSS_UDP trata de todo o processamento relativo a um fluxo UDP. Seu método principal é executado através de uma *thread* que é instanciada ao se iniciar o processo de captura do fluxo. Este processo de captura do fluxo é iniciado através da tecla “Iniciar” da interface do QoSS conforme código abaixo. Vale lembrar que este método também inicia a thread do tipo T1 da Figura-7.

```
private void Iniciar(java.awt.event.MouseEvent evt) {

    jTextField1.setText("R$ 100,00");

    Thread t = new Thread() {
        String host = "127.0.0.1";
        int localport = 88;
        int remoteport = 1234;

        public void run() {
            try{
                // QoSS_UDP udp = new QoSs_UDP();
                udp.runQoSs(host, remoteport, localport);
                jButton1.setForeground(java.awt.Color.red);
                jButton1.setEnabled(false);
            } catch (Exception e) { System.err.println(e); }
        }
    };
    this.monitorar(evt); // Thread do Tipo T1
    t.start();
}
```

```

    }

    private void monitorar(java.awt.event.MouseEvent evt) {

        Thread t = new Thread() {
            String x = "";

            public void run() {
                jButton2.setEnabled(false);

                long Delay = System.nanoTime() + 10000000000;

                while(true) {

                    Delay = System.nanoTime() + 10000000000;
                    while(System.nanoTime() < Delay) { }

                    // atraso
                    x = String.format(Locale.GERMANY, "%,8d%n", udp.atraso);
                    jTextField3.setText(x);

                    // Pacotes
                    x = String.format(Locale.GERMANY, "%,8d%n", udp.pkt);
                    jTextField8.setText(x);
                    x = String.valueOf(udp.pkt1);
                    jTextField12.setText(x);

                    // Jitter
                    x = String.format(Locale.GERMANY, "%,8d%n", udp.Jitter);
                    jTextField10.setText(x);

                    // pJitter
                    if (udp.atraso==0) { x="0"; } else {
                        x = String.format(Locale.GERMANY, "%,5f%n", (float)
udp.Jitter/udp.atraso*100); }
                    jTextField4.setText(x);

                    // Banda Percebida
                    x = String.format(Locale.GERMANY, "%,8d%n", udp.bd);
                    jTextField2.setText(x);

                    // Tempo
                    x = String.format(Locale.GERMANY, "%,8d%n", udp.bd/8);
                    jTextField6.setText(udp.hora());

                }
            }
        };

        jButton2.setEnabled(false);
        t.start();
    }

```

O método `get_Banda_Percebida()` executa todo o tratamento de um datagrama dentro do `QoSS_UDP` conforme abaixo:

- Capturar pacote e torná-lo disponível para simulação;
- Simular atrasos, *jitter* e perdas;
- Incrementar a banda percebida;
- Re-enviar o pacote para o(s) destino(s).
- Registrar selos de tempo T4-T3 da Figura-20 para cálculo da média.

A seguir o código em Java da classe `QoSS_UDP`.

```
package qossinterface;

import java.io.*;
import java.net.*;
import java.util.*;
import java.util.concurrent.*;

/**
 *
 * @author: Gabriel M. O. Vieira
 *         USP-POLI-ELE-PCS-LSA
 */

// Classe QoSS_UDP
public class QoSS_UDP {

    public static long Jitter = 0;
    long bd;
    long pkt=0;
    int pkt1=0;

    public static float p=0;
    public static long atraso = 0;
    float pJitter = 0;

    public void runQoSS (String host, int remoteport, int localport)
        throws IOException {

        DatagramSocket socket = null;
        socket = new DatagramSocket(localport);

        System.out.println("Buffer S.O.: " +
socket.getReceiveBufferSize());
        System.out.println( "Simulação iniciada ...");

        try {
            byte[] buf = new byte[socket.getReceiveBufferSize()];
```

```

        DatagramPacket packet = new DatagramPacket(buf, buf.length);
        DatagramSocket socketService = new DatagramSocket();

        while (true) {

            get_Banda_Percebida(host, remoteport, localport, socket,
socketService, Jitter);

        }
        } catch (IOException e) { e.printStackTrace(); }
    }

// Método setDelay que simula atrasos de t nanosegundos.
    public void setDelay (long t) throws IOException {

        long wDelay = System.nanoTime()+ t;

        // Atrasa t nanosegundos
        while(System.nanoTime() < wDelay) { }
    }

// Método setPerdas
    public boolean setPerdas (int pacote) throws IOException {

        if ((pacote % (int)(100 / this.p)) == 0) { return true; };
        return false;
    }

    public void get_Banda_Percebida (String host, int remoteport, int
localport, DatagramSocket socket, DatagramSocket socketService, long
Jitter) throws IOException {

        long t;

        try {
            byte[] buf = new byte[socket.getReceiveBufferSize()];

            DatagramPacket packet = new DatagramPacket(buf, buf.length);
            atraso = 0;

            while (true) {

                int pkt2 = 0; // pacotes por segundo
                long wDelay = System.nanoTime()+ 10000000000; // 1 segundo

                while(System.nanoTime() < wDelay) {

                    t = System.nanoTime();

                    // Captura pacote
                    packet = new DatagramPacket(buf, buf.length);
                    socket.receive(packet);

```



```

        // Simula Perdas
        if (this.p > 0 && pkt2 > 0) {if (setPerdas(pkt2))
socket.receive(packet); };

        // Simula atraso e jitter.
        setDelay(this.atraso);
        this.atraso=this.atraso + this.Jitter;

        // Jitter %
        if (this.atraso > 0) {pJitter = (this.Jitter) / this.atraso
* 100; }

        // Envia pacote para address/remoteport
        // Pode-se re-enviar (através de um laço)
        // este pacote para n destinos diferentes
        // apenas alterando os parâmetros address e remoteport

        InetAddress address = packet.getAddress();
        packet = new DatagramPacket(buf, buf.length, address, remoteport);
        socketService.send(packet);

        // Calcula T4-T3 da Figura-20
        System.out.println(System.nanoTime()-t);
        System.out.println(" ");
        pkt2++;

    } // 1 segundo

        pkt = pkt + pkt2;
        pkt1 = pkt2;
        bd = pkt1 * buf.length * 8;

    }

    } catch (IOException e) {
        e.printStackTrace();
    }

    }

// Método getTimeStamp
public void getTimeStamp() {
    System.out.println(System.nanoTime());
}

// Método setTimeStamp
public long setTimeStamp() {
    return System.nanoTime();
}
}
}

```

3.4.2 Controles deslizantes – Eventos e Métodos

O método abaixo é disparado sempre que o “oculista” altera a posição do controle deslizante “A” (atraso). Este método altera de forma assíncrona o valor do atraso sendo simulado durante uma sessão do usuário com o serviço. Vale lembrar que o objeto udp é instanciado sempre que um novo fluxo UDP é percebido pelo QoSS.

Os controles deslizantes relativos ao *jitter* e as perdas tem tratamento semelhante.

```
private void sAtraso(java.awt.event.MouseEvent evt) {

    // posição do controle deslizante
    int v = jSlider2.getValue();

    // Atualiza valor do índice do controle deslizante
    // na interface do QoSS
    String x = String.valueOf(100-v);
    jTextField7.setText(x);

    // atribui ao atraso (de forma assíncrona) a posição do
    // controle deslizantes multiplicado por sua granularidade
    udp.atraso=(100-v)*1000000; // granularidade 1ms

    // atualiza valor do atraso na interface do QoS
    x = String.format(Locale.GERMANY, "%,1d%n", udp.atraso);
    jTextField3.setText(x);

}
```

4 ESTUDO DE CASO

O serviço cujos requisitos de QoS serão simulados, é uma sessão entre um usuário final de uma Central de Transmissão de Vídeo em redes IP utilizando o aplicativo VLC MEDIA PLAYER e simultaneamente, o mesmo usuário joga uma partida de xadrez em um serviço para se jogar xadrez *on line* instalado no ambiente de simulação conforme estabelecido no DDS do serviço.

Não houve neste estudo, a preocupação de simular a técnica de *multicast* que normalmente é utilizada em serviços deste tipo (SILVEIRA, 2007). Entretanto, o QoSS se comportou de maneira satisfatória, ou seja, sem apresentar queda de desempenho significativa ao atuar como uma central de redistribuição de vídeos ao redistribuir um mesmo fluxo de vídeo para três usuários (sockets) diferentes e simultaneamente. A Figura-11 mostra um fluxo UDP sendo redistribuído para três usuários diferentes após ter sido submetido a um eventual processo de simulação. A seção 3.4.1 apresenta o código-fonte em Java relativo a esta redistribuição.

Figura-11 Redistribuição de vídeo (1→3)

4.1 AMBIENTE REAL DO SERVIÇO

Esta seção apresenta as funcionalidades do VLC MEDIA PLAYER e do XadrezOnline operando em seus ambientes originais e reais.

A Figura-12 apresenta a topologia do ambiente real do serviço que é disponibilizado onde dispositivos clientes do XadrezOnline e do VLC MEDIA PLAYER se conectam a seus servidores por suas respectivas redes de acesso e *backbones*. Vale lembrar que o vídeo sendo transmitido pelo VLC poderia estar sendo redistribuído para “n” usuários e que também “n” usuários poderiam estar na “platéia” do jogo de xadrez entre o Jogador 1 e o Jogador 2.

Figura-12: Ambiente real do XadrezOnline e VLC Media Player

4.1.1 VLC MEDIA PLAYER

O VLC MEDIA PLAYER (www.videolan.org) é um aplicativo *freeware*, que trata conteúdo multimídia como vídeo e som. Entre outros recursos, o VLC permite que uma mídia contínua de vídeo ou voz sejam capturados e enviados ou reenviados para serem apresentados remotamente. A escolha desse aplicativo para ser

utilizado no estudo de caso, deveu-se também ao fato de o VLC permitir que seus fluxos de entrada e saída sejam capturados e manuseados.

Por exemplo: Pode-se escolher para que *socket* se deseja enviar as requisições ou as transmissões. Alguns aplicativos que tratam vídeo e voz mais populares como o Skype não oferecem tal facilidade. A Figura-13 mostra a interface do VLC MEDIA PLAYER onde se pode definir e configurar as características de uma transmissão e recepção remota de vídeo e som pela Internet.

Figura-13 Transmissão remota via VLC MEDIA PLAYER

4.1.2 Xadrez Online

O Xadrez Online é um serviço para se jogar xadrez pela Internet. O site foi desenvolvido em Visual Basic 6.0 e utiliza tecnologia CGI (*Common Gateway Interface*) e *webserver* WEBSITE PRO 2.5.4 e está disponível para qualquer usuário que tenha um *browser* para acessá-lo.

O XadrezOnline possui “salas” para se jogar por pontuação, amistosos e salas com tecnologias experimentais. O serviço também oferece salas de “Batepapo” e tutoriais para iniciantes e para jogadores com alguma experiência. Também possui recursos para simultâneas com jogadores renomados.

O XadrezOnline foi escolhido para compor o serviço deste estudo de caso, por ser uma aplicação transacional em todas as suas interações e também, pelo fato de que a performance das transações (lances) influenciam diretamente na qualidade e no resultado do jogo. O principal e mais utilizado recurso do serviço foi instalado no ambiente de simulação cuja topologia utilizada é apresentada na Figura-14. Note que o fluxo nos dois sentidos para as peças brancas é capturado pelo QoS. As peças pretas apenas fazem o papel do oponente e não teve seu fluxo definido no DDS. Note também, que o XadrezOnline, diferentemente da transmissão de vídeo do VLC MEDIA PLAYER, utiliza o protocolo HTTP sobre TCP/IP.

Figura-14: Topologia do Ambiente Simulado do Xadrez Online

O serviço que foi utilizado neste estudo é o tabuleiro da “sala” onde os jogos valem pontuação. Todos os lances são validados no servidor e só depois de validados são enviados simultaneamente para os tabuleiros (respectivamente para quem efetuou o lance e para quem espera pelo lance). Quanto mais rápido um determinado usuário receber os lances, mais tempo terá para pensar e jogar. Muitos jogos no XadrezOnline são decididos em função da qualidade dos serviços contratados. Exemplo: Quem joga por linha-discada geralmente tem menos tempo para pensar e jogar do que quem joga através de um serviço de banda larga.

Seria útil para um internauta que deseja jogar xadrez no XadrezOnline simular um jogo via QoS antes de se decidir pelo tipo de assinatura do serviço, pois corre o risco de perder muitos jogos pelo tempo.

A Figura-15 mostra os recursos que um usuário possui para assistir aos jogos e receber os lances simultaneamente aos participantes do jogo. O usuário tem acesso a uma lista de jogos *on line* e pode assistir a vários jogos simultaneamente.

Figura-15 “Sala” de jogos por pontuação do XadrezOnline (platêia)

4.2 TOPOLOGIA DO AMBIENTE DE SIMULAÇÃO

Neste estudo de caso foi experimentada uma composição de serviços operando em redes IP onde o usuário através do VLC MEDIA PLAYER SERVIDOR deseja receber um vídeo para ser apresentado no VLC MEDIA PLAYER instalado no seu dispositivo e, ao mesmo tempo, que através de um *browser*, joga uma partida de xadrez em um site de xadrez instalado na rede isolada do ambiente de simulação.

Vale lembrar que o fluxo gerado entre o servidor de xadrez e o dispositivo do oponente não estavam definidos no DDS do serviço. A Figura-16 apresenta o ambiente de simulação utilizado no estudo de caso onde apenas os fluxos de D4 para D5, entre D5 e D1, e entre D5 e D3 foram definidos no DDS.

Todos os lances do jogo de xadrez são efetuados no servidor, e se validados, são enviados para o oponente pelo servidor não havendo conexão direta entre os dispositivos dos jogadores.

Figura-16 Ambiente de Simulação do Estudo de Caso

Todo o fluxo do serviço hospedado no ambiente de simulação converge para o QoSS para ser simulado. Por exemplo, o dispositivo D4 envia um vídeo para o dispositivo D5 da seguinte forma: Ao invés de enviar o vídeo para o IP/Porta de D5,

D4 envia para a porta 88 do dispositivo do oculista. Após uma eventual simulação dos requisitos de QoS, o vídeo é re-enviado para D5 pelo QoSS. E assim é para todas as conexões definidas no DDS.

As conexões definidas entre D5 e D3, são relativas a uma consulta a um serviço de resolução de nomes para se determinar o endereço do XadrezOnline. Vale acrescentar que o serviço de DNS retorna o IP do QoSS e através do DDS, o QoSS conhece o endereço real do XadrezOnline na rede isolada do ambiente de simulação.

4.2.1 Parametrização dos Controles Deslizantes

Os parâmetros de Qualidade de Serviço que são simulados neste trabalho são atrasos, *jitter*, perdas e banda percebida e todos concorrendo para espelharem a qualidade percebida pelo usuário final.

O QoSS é uma ferramenta projetada para simular o comportamento conjunto destes parâmetros de QoS em função de uma granularidade e de níveis de serviço mínimos e máximos que são oferecidos pelos Provedores de Serviços via DDS conforme Tabela-1.

A Tabela-1 apresenta dados contidos do DDS relativos ao sentido das conexões e parametrização dos controles deslizantes. O campo origem (quando marcado) representa o fluxo das respostas, e o campo destino (quando marcado) representa o fluxo das requisições. Por exemplo: no caso da central de vídeo, o campo destino é marcado para representar a solicitação de um vídeo e o campo origem é marcado para representar a transmissão do vídeo solicitado.

Ao se aplicar nos controles deslizantes os limites mínimos e máximos dos parâmetros da tabela, isto também significa dizer que os controles deslizantes não geram valores fora destes limites e este controle, naturalmente, é exercido pelo QoSS.

Assim, é necessário uma adaptação no mecanismo que aplica estes requisitos no fluxo de pacotes visto que um único controle deslizante (exemplo: controle deslizante A) terá que gerar atrasos diferenciados para cada tipo de mídia que compõe o serviço.

HILLEBRAND et.al. (2004) afirma que, no futuro, as redes sem fio serão todas IP. Isto, no entanto, não é suficiente para tornar homogênea e uniforme a definição dos requisitos de QoS de um serviço cujas aplicações demandem tráfego heterogêneo entre si como uma transmissão de TV digital ao vivo e uma sessão via *browser* com uma aplicação transacional de *home banking* ou como uma simples transmissão de um vídeo e uma aplicação transacional para se jogar xadrez via WEB. Tomando o atraso como exemplo, uma solução possível para este problema seria considerar o atraso das aplicações que compõem o serviço como uma função do atraso de uma aplicação principal definida como tal pelo usuário final.

Assim, teríamos: $ApP \rightarrow A(m,M,G)$

Onde “A” é o atraso da aplicação principal (ApP);

“m” é o atraso mínimo da ApP definido na Tabela-1;

“M” é o atraso Máximo da ApP definido na Tabela-1 e

“G” é a granularidade definida para o atraso na Tabela-1.

Seja então ApP a aplicação principal do serviço que tem definida o atraso mínimo, Máximo e granularidade conforme Tabela-1 e as demais aplicações seriam uma função do atraso da aplicação principal. Por exemplo: O atraso da Ap1 seria uma função do atraso de ApP, o atraso da Ap2, outra função de ApP e assim sucessivamente conforme fórmula abaixo e Tabela-2:

$Ap1 \rightarrow f_1A(m,M,G)$, $Ap2 \rightarrow f_2A(m,M,G)$, $Apn \rightarrow f_nA(m,M,G)$

Assim, o atraso das “n” aplicações estaria indexado a um único controle deslizante que geraria atrasos simultaneamente para todas as aplicações (fluxos) do serviço.

A Figura-17 apresenta uma alternativa para serviços com “n” mídias que precisam ter os requisitos de QoS gerados de forma independente.

Figura-17 Janela de controles deslizantes para Serviço com “n” Mídias

Figura-18 Janela de controles deslizantes para Serviço com “n” Mídias (2)

A Figura-18 apresenta uma alternativa em relação à Figura-17 onde apenas uma aplicação (tipo de mídia ou tecnologia de rede) estaria ativa de cada vez, e assim, apenas um conjunto de controles deslizantes A, J e P seria necessário, entretanto, esta solução não permite que mais de um tipo de mídia ou tecnologia tenha seus requisitos de QoS simulados simultaneamente.

4.2.2.1 Perda de Pacotes

O oculista ao se utilizar dos controles deslizantes, está navegando entre os valores mínimos e máximos do *jitter* e do atraso, e as perdas são aplicadas em função de

uma porcentagem ou distribuição informadas pelo Provedor Serviços. A banda percebida é recalculada a cada segundo e é apresentada via interface do QoS.

Apenas como exemplo, visto que a distribuição deve ser informada pelo provedor de serviços via DDS, poder-se-ia aplicar uma função randômica para descartar a porcentagem de pacotes estabelecida no DDS.

Considera-se perdas de pacote como sendo a porcentagem de pacotes que não atingem o destino ou são corrompidos durante uma transmissão ou período de tempo, entretanto, os pacotes descartados pela aplicação por atrasos prolongados não são considerados na porcentagem de perdas informada pelo Provedor de Serviços.

4.2.2.2 O atraso e o *jitter*

Note na Figura-19 que o clip.avi apresenta uma aparência típica de uma apresentação com perda de pacotes mas nenhuma perda foi simulada neste trecho do video. Isto se deve ao fato de que o VLC passou a descartar pacotes por espera prolongada quando os atrasos simulados atingiram a faixa de 7.5 ms.

Vale lembrar que os pacotes trafegaram utilizando RTP (*Real-Time Protocol*) sobre UDP (*User Datagram Protocol*).

Figura-19 Perda de Pacotes por atrasos prolongados do clip.avi

O *Jitter* é a relação entre o atraso de um pacote com o atraso do pacote anterior. Assim, a fórmula genérica para o cálculo do *Jitter* de um determinado pacote p seria:

$$Jitter = \frac{a}{p-1} \quad (2)$$

onde **a** (acrécimo) representa a diferença entre o atraso de um pacote com seu antecessor.

$$\text{Para } p = 1, Jitter = 0 \quad (2)$$

$$\text{Para } p > 1, Jitter = 100.a / (p-1) (\%) \quad (2)$$

4.2.2.3 Preço

A relação entre os valores dos parâmetros A, J e P com o preço do serviço deve ser definida (no DDS) de acordo com os valores a serem pagos pelo Provedor de Serviço para os outros provedores envolvidos no provimento do serviço (provedor de conteúdo e provedor de infra-estrutura) mais uma taxa de administração (modelo do negócio). Esta relação deve ser conhecida e entendida pelo oculista, a fim de ser aplicada nas sessões de simulação.

No início da sessão de simulação, o atraso é aplicado somente no primeiro pacote de uma determinada transmissão. Para os pacotes seguintes, o custo para determinar o preço do serviço deve considerar apenas o *jitter* e a perda de pacotes, levando-se em conta que a influência do atraso já foi aplicada no primeiro pacote.

4.3 ANÁLISE DOS RESULTADOS

A necessidade do QoS é produzir, sob o ponto de vista qualitativo, o mesmo resultado que uma transmissão em uma rede real, no entanto, é importante sabermos até que ponto isso é possível e o quanto estes resultados podem ser confiáveis.

Para se conhecer o custo de processamento de um pacote pelo QoS, foi estabelecido quatro selos de tempo, T1, T2, T3 e T4 conforme Figura-20.

Figura-20 Disposição dos Selos de Tempo

$$\text{Distância de Simulação}_{\text{pacote}} = (T4 - T3) - (T2 - T1) \quad (2)$$

Onde $T2 - T1$ é o tempo gasto para um pacote trafegar sem interferências desde a sua origem até o destino e;

$T4 - T3$ é o tempo gasto para um pacote trafegar desde a sua origem até o destino passando pelo dispositivo do QoS considerando $A=0$, $J=0$ e $P=0$.

```
// Método setTimeStamp ex. long t = setTimeStamp();
public static long setTimeStamp() {
    return System.nanoTime(); }
```


Vários experimentos foram efetuados, conforme método acima, estabelecendo-se os selos de tempo $T_4 - T_3$ da Figura-20. Os valores encontrados variaram em média 8 milissegundos.

A seguir, é apresentado o ambiente operacional do experimento:

Hardware: HP Core 2 DUO/4GB.

Software: Product Version: NetBeans IDE 6.0.1 (Build 200801291616)

Java: 1.6.0_04; Java HotSpot(TM) Client VM 10.0-b19

System: Windows Vista version 6.0 running on x86; Cp1252; en_US (nb)

Rede: IP

Para $A=5\text{ms}$, $J=0$ e $P=0$, o fluxo do serviço se comportou com a mesma qualidade que uma transmissão direta sem utilizar o QoSS. A simples captura e reenvio dos pacotes, mais a aplicação do atraso de até 5ms, não gerou perda de qualidade significativa na apresentação do serviço. Isto ocorre por que os recursos de *buffer* do VLC cliente que apresentava o vídeo estava configurado para suportar atrasos superiores a treze mili segundos (ms), conforme discriminado abaixo:

- 5.0 ms → atrasos simulados pelo QoSS para pacotes de 8K.
- 8.0 ms → tempo médio para capturar e re-enviar cada pacote pelo QoSS;
- outros atrasos, como por exemplo, o processamento das aplicações que compõem o serviço.

Os experimentos efetuados neste estudo de caso mostram que:

- o tempo para se capturar e re-enviar um pacote pelo QoSS precisa diminuir;
- a plataforma Java não é a mais adequada visto que a simplicidade de código e portabilidade não são requisitos considerados prioritários em relação ao comprometimento do desempenho.

Uma mudança de paradigma no desenvolvimento e implementação do QoS se faz necessária visto que, como mostra este estudo de caso, um requisito ainda não pode ser atendido:

- O simulador, ao efetuar suas tarefas, não deve prejudicar o desempenho do serviço sendo avaliado.

A princípio, este requisito foi classificado como não-funcional, no entanto, um requisito não atendido, por conta de alguma restrição tecnológica, deve evoluir para um requisito do tipo normativo.

Outros experimentos efetuados na prova de conceito suscitaram as seguintes questões:

- Foi observado que, para um *codec* gerando um *bit-rate* variável, um vídeo pode apresentar uma demanda diferenciada por recursos a cada segundo. O vídeo observado variou de 60 a 303 pacotes de 8k por segundo.
- O modelo estabelece que o oculista deve interagir com o usuário final atuando como um elemento de interface.

A seguir, um exemplo desta interação:

Usuário: “A qualidade agora ficou muito boa! Agora está mais nítido, mas o preço subiu muito. Pode diminuir um pouco a qualidade, por favor!”.

Oculista: “OK !!”.

Considerando que o “oculista” demora um ou mais segundos para reagir, todas as ações estarão atuando no presente, mas são relativas a um estado passado do vídeo ou serviço. Esta questão poderia ser contornada com a inversão dos procedimentos, ou seja, em vez de aguardar por um comando de voz do usuário e reagir, o oculista deve agir e aguardar por comentários.

Mas como agir considerando um tráfego tão dinâmico? Os mesmos requisitos de QoS que valem para 60 pacotes por segundo também valem para 303? E esta diferença ainda pode aumentar muito considerando uma cenografia mais dinâmica e orquestrada seguida de, por exemplo, um horizonte sempre estático, vazio e silencioso.

O “oculista” deve conhecer e entender bem as informações contidas no DDS. Assim, usando sua experiência no manuseio das lentes (controles deslizantes), o “oculista” saberá como conduzir uma sessão de simulação.

5 CONSIDERAÇÕES FINAIS

A simulação de um processo através de um software permite a inclusão de muitos aspectos do problema sem a imposição de restrições. O problema passa a ser com o tempo demandado pelo software que simula o processo. Esse tempo normalmente cresce com a complexidade do modelo.

Mesmo para um engenheiro de software profissional, é difícil aceitar que o processo de elicitação de requisitos de um sistema de software o conduziu a uma solução na qual o software por si só não é suficiente para atender a todos os requisitos identificados. Os controles deslizantes do “oculista”, por exemplo, devem poder ser manuseados de forma a não prejudicar o desempenho do serviço sendo avaliado. Um exemplo seria a utilização de um *joystick* (manche) específico.

A adaptação dinâmica dos controles deslizantes com relação à função que deve ser aplicada para se definir os atrasos, *jitter*, etc. das aplicações do tipo não-principal pode não ser possível para todos os casos.

A solução da Figura-17 propõe instanciar um conjunto de controles deslizantes (A, J e P) para cada tipo de mídia componente do serviço. Isto pressupõe que vários controles deslizantes teriam que estar disponíveis para o “oculista” e serem manuseados de forma simultânea e assíncrona. Além disso, as soluções apresentadas nas Figuras 17 e 18 apresentam restrições importantes concernentes a escalabilidade.

Os experimentos efetuados na prova de conceito (QoSS), como por exemplo, a percepção do tempo decorrido entre uma ação do “oculista” no controle deslizante e o reconhecimento desta ação pelo serviço, mostram que a simples mudança da plataforma Java para uma linguagem nativa como o C++ pode não ser suficiente para garantir o desempenho do assincronismo entre os controles deslizantes e demais demandas de processamento paralelo do QoSS. O mesmo se aplica ao tempo demandado para processar a captura e entrega de cada pacote injetado na rede pelo serviço.

Outro aspecto importante é relativo à transparência do processo de captura dos pacotes visto que a versão atual ainda necessita de algum artifício para fazer o fluxo do serviço convergir para o QoSS.

Face ao exposto acima, sugere-se como trabalhos futuros o que se segue:

- Desenvolvimento de um dispositivo que possa atuar como um elemento de rede (*switch* ou roteador) e que agregue todos os requisitos identificados para o QoSS tais como assincronismo entre processos aliado a desempenho, elementos de interface que substituam o “oculista” e escalabilidade no tratamento das aplicações componentes do serviço;
- Identificação de outros critérios para se definir o melhor momento e a melhor técnica de se capturar e interferir no tráfego de um pacote para fins de simulação dos requisitos de QoS sob a ótica do usuário final sob o ponto de vista de desempenho observando os preceitos da arquitetura de referência OSI (*Open Systems Interconnection*).

Dada a quantidade de variáveis que compõem o domínio do problema discutido aqui, ou seja:

- Continuação da pesquisa de SERRA (2007) no que tange à métodos de definição de QoS percebida e técnicas de medida dos parâmetros de QoS;
- Subsídio ao *framework* de desenvolvimento de aplicações do LSA (Figura-1);
- Identificação de valores de referência sob a ótica do usuário final aplicáveis à adaptação dinâmica de requisitos técnicos de QoS para serviços convergentes e interativos com conteúdo multimídia operando em ambiente heterogêneo;

- E todos estes aspectos estando em consonância com as pesquisas de KARAM (2006), SERRA (2007), MAIA (2009) e MARTUCCI et. al. (2008, 2010).

este trabalho dá um passo no sentido de:

- Estabelecer uma mudança de paradigma visto que QoS normalmente é definido em função da capacidade de recursos da rede e não em função da satisfação do usuário.
- Identificar as principais dificuldades encontradas para o efetivo estabelecimento de requisitos de QoS centrado no usuário final, com o intuito de subsidiar pesquisas futuras intencionadas em definir requisitos de QoS através deste novo paradigma;
- Conseguir definir um domínio que integre todas as pesquisas citadas acima e situar as inúmeras alternativas de solução no contexto das principais ferramentas de código aberto, que se propõem a tratar o estabelecimento de requisitos de QoS em ambiente convergente e heterogêneo.
- Evoluir as relações entre o usuário final e o Provedor de Serviços ao se discutir uma visão personalizada de QoS para o estabelecimento de USLAs.

Finalmente, este trabalho mostra que:

- A simulação do comportamento conjunto dos requisitos de QoS de um serviço segundo a vontade e percepção do usuário final pode ser feita e;
- Este mecanismo pode ser considerado como um mapeamento direto dos aspectos subjetivos que envolvem a percepção do usuário final em requisitos técnicos de QoS.

REFERÊNCIAS

- (BELLAVISTA, 2003) Bellavista, P.; Corradi, A.; Stefanelli, C. Application level QoS Control for Video-on-Demand. In: IEEE Internet Computing, November, 2003. pp. 16-24.
- (BELLONI, 2008) BELLONI, Maria Luiza, Educação a Distância, 5 ed. Campinas, SP. Editora Autores Associados. 2008.
- (BEZERRA, 2007) Bezerra, E. Princípios de Análise e Projeto de Sistemas com UML. Ed. Campus. Rio de Janeiro, Brasil. 2007.
- (BIRGITTA, 2009) BIRGITTA Bergvall-Kåreborn et. Al., Concept Design with a Living Lab Approach, Proceedings of the 42nd Hawaii International Conference on System Sciences – 2009
- (C++, 2010) C++, <http://pt.wikipedia.org/wiki/C%2B%2B>, acessado em 2010.
- (CAMPONOVO, 2003) CAMPONOVO, G.; PIGNEUR, Y. Business Model Analysis Applied to Mobile Business. Proceedings of the 5th International Conference on Enterprise Information Systems – ICEIS 2003, Angers, France, apr/2003. pp 173-183.
- (CHUNG, 2007) CHUNG, L. Non-Functional Requirements in Software Engineering, 2007.

- (COMER, 2006) COMER, D. E. Internetworking with TCP/IP. Fifth Edition, 2006.
- (FLUCKIGER, 1995) FLUCKIGER, François, Understanding Networked Multimedia: applications and Tecnology. Prentice Hall,1995
- (GABOS, 2007) GABOS, D., Notas de Aula, Curso: Gerenciamento de Redes. PCS-USP, 2007
- (GHINEA, 2005) GHINEA, G. Quality of Perception: User Quality of Service in Multimedia Presentations. Member of IEEE. 2005.
- (GUINEA, 1999) GHINEA, G., Thomas J.P. An Approach towards Mapping Quality of Perception to Quality of Service in Multimedia Communications. Multimedia Signal Processing, 1999 IEEE 3rd Workshop on, 1999.
- (GOZDECKI, 2003) GOZDECKI, J.; Jajszczyk, A. Quality of Service Terminology in IP Network. IEEE Communications Magazine, 2003.
- (HAROLD, 2004) HAROLD, E.R. Java Network Programming. O'Reilly, 2004.
- (HILLEBRAND, 2004) HILLEBRAND, J. et. al. Quality of Service signaling for Next-Generation IP-Based Mobile Networks. In: IEEE Communications Magazine, jun. 2004. pp. 72-79.

- (HYUN-JONG KIM, 2008) HYUN-JONG KIM, Dong-Hyeon Lee, Jong-Min Lee, Kyoung-Hee Lee, The QoE Evaluation Method through the QoS-QoE Correlation Model, Fourth International Conference on Networked Computing and Advanced Information Management, IEEE Computer Society, 2008
- (HORSTMANN, 2000) HORSTMANN, C.S., Cornell G., Core Java™ 2: Volume I–II, Prentice Hall PTR, Fifth Edition December 01, 2000
- (HOUAISS, A) HOUAISS, A., Dicionário Houaiss da Língua Portuguesa, Ed. Objetiva. 2001
- (IEEE-830, 1998) IEEE Standards Association, IEEE Std 830 IEEE Recommended Practice for Software Requirements Specifications, 1998.
- (ITU, 2004) ITU – T INTERNATIONAL TELECOMMUNICATION UNION Y.2001 General Principles and general reference model for Next Generation Networks. 2004.
- (JENSEN, 1998) JENSEN, J. F. Interactivity: Tracing a new concept in media and communication studies. vol. 19. Nordicom Review. 1998. pp. 185–204.
- (JERUCHIM, 2005) JERUCHIM, Michel C.(Author). Simulation of Communication Systems, Second Edition: Modeling, Methodology and Techniques. Hingham, MA, USA: Kluwer Academic Publishers, 2000.
- (KARAM, 2006) KARAM, D. J.; Business Model for Interactivity and Mobility in a Convergent and Heterogeneous Environment. Tese de Doutorado, EPUSP, Universidade de São Paulo, São Paulo, Brasil, 2006.

- (MAIA, 2008) MAIA, R. F, Martucci Jr, M. Organization of Mobile Agents for Heterogeneous Network QoS Control Architecture, Communications and Information Technologies, 2008. ISCIT 2008. International Symposium on 21-23 oct. 2008 pages: 664-669.
- (MAIA et. al., 2008) MAIA, R. F, GABOS, D., Martucci Jr, M. A distributed QoS control architecture for heterogeneous networks applied to PLC network. Power Line Communications and Its Applications, 2008. ISPLC 2008. IEEE International Symposium on 2-4 april 2008 pages: 52-57.
- (MAIA, 2009) MAIA, R, F. Arquitetura Genérica e Aberta de Controle de Qualidade de Serviço para Ambientes Convergentes e Heterogêneos. Tese (Doutorado) - Escola Politécnica da Universidade de São Paulo, São Paulo, em elaboração. 2009
- (MARTUCCI , 2008) MARTUCCI M, J.; HIRAKAWA, A. R. e-Infrastructure for Critical Applications, <http://www.paradiso-fp7.eu/>, 2008.
- (MARTUCCI et. al. 2010) MARTUCCI, M, J. et. al., Grupo de discussões dos pesquisadores do LSA (Laboratório de Sistemas Abertos). Departamento de Sistemas Digitais e Computação. Escola Politécnica da Universidade de São Paulo. 2010.
- (MARTIN, 2002) MARTIN, R. C. UML for JAVA Programmers. Prentice-Hall Inc. 2002.
- (MENDEL, 1997) MENDEL T., Elements of User Interface Design, John Wiley & Sons, Inc., 1997
- (MENDES, 2006) MENDES, Joyce Martins, Avaliação de serviços convergentes em ambientes heterogêneos. São Paulo, Tese (doutorado) – Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia de Computação e Sistemas Digitais. 2006.

- (MELNIKOFF, 2007) MELNIKOFF, S. Notas de Aula, Curso: Avaliação de Métodos de Análise e Projeto Orientado a Objeto. PCS-USP, 2007
- (MINDLIN, 2005) MINDLIN, P.A.L, BARTH, J.F, SPINA, E., GABOS, D., Evaluation of Real Applications and Services Using the Heterogeneous Network QoS Evaluation Tool. WSEAS TRANSACTIONS ON INFORMATION SCIENCE AND APPLICATIONS. Issue 6, Volume 2, June 2005.
- (OAKS, 1999) OAKS, S.; Wong, H. Java Threads, O'Reilly, 2nd Edition, 1999.
- (PRESSMAN, 2005) PRESSMAN, Roger S., Software Engineering, 6th ed. 2005
- (PUTMAN, 2001) PUTMAN, J. Architecting with RM-ODP. Prentice Hall PTR, 2001.
- (KRUCHTEN, 2000) KRUCHTEN, P. The Rational Unified Process: An introduction. Addison Wesley Professional; 2 edition (2000).
- (RAMIREZ, 2010) RAMIREZ, G., et. al. <http://www.wireshark.org/docs/>, acessado em 2010.
- (REILLY D, et. al, 2002) REILLY, D., REILLY M, **Java™ Network Programming and Distributed Computing**, Addison Wesley, 2002, Pages: 496

- (SANTANA, 1997) SANTANA, M. J. ; SANTANA, R. H. C. ; FRANCÊS, C. R. L. ; ORLANDI, R. C. G. . Tools and Methodologies for Performance Evaluation of Distributed Computing Systems - A Comparison Study. In: Summer Computer Simulation, 1997. Proceedings of 1997 Summer Computer Simulation, 1997.
- (SERRA, 2006) SERRA, A.P.; Martucci Jr, M. A method to elicit the quality of service parameters to multimedia interactive and mobile services in a technological convergent environment, Proceedings of the 6th WSEAS International Conference on Multimedia, Internet & Video Technologies, 2006 pages:30-35.
- (SERRA, 2007) SERRA, P. Método para Identificação de Parâmetros de Qualidade de Serviços Aplicados a Serviços Móveis e Interativos. Tese de Doutorado, EPUSP, Universidade de São Paulo, São Paulo, Brasil, 2007.
- (SILVEIRA, 1998) SILVEIRA, R. M., Notas de Aula, Curso: Tecnologias para Aplicações de Redes Avançadas, LARC-USP, 2007
- (SUBBIAH, 2008) SUBBIAH, P. Bit-Error Rate for High-Speed Serial Data Communications, <http://www.cypress.com>, NOV, 2008.
- (SUN, 2009) SUN MICROSYSTEMS, <http://java.sun.com/javase/7/docs/api/> , Java™ Platform, Standard Edition 7 API Documentation, DRAFT ea-b66. Acessado em 2009
- (SUN, 2008) SUN MICROSYSTEMS, <http://java.sun.com/docs/books/tutorial/networking/>, acessado em SET/2008
- (TANENBAUM, 2003) TANENBAUM, A. S. **Computer Networks**. 4 ed. Prentice Hall, 2003.

(WARD, 2005)

WARD, C. Fresco: A Web Services Based Framework For Configuring Extensible SLA Management Systems. IBM Thomas J. Watson Research Center. 2005. Proceedings of the IEEE International Conference on Web Services (ICWS'05).

(WEBSTER' 2006)

WEBSTER`S DICTIONARY, 4ed. 2006.