

**CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA
CELSO SUCKOW DA FONSECA - CEFET/RJ**

**DIRETORIA DE PESQUISA E PÓS-GRADUAÇÃO
COORDENADORIA DO PROGRAMA DE PÓS-GRADUAÇÃO EM TECNOLOGIA**

DISSERTAÇÃO

**GERENCIAMENTO, ANÁLISE E TRANSMISSÃO DE IMAGENS,
ON-LINE, VIA TCP/IP.**

Angelo Márcio de Paula

**DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DO PROGRAMA DE
PÓS-GRADUAÇÃO EM TECNOLOGIA COMO PARTE DOS REQUISITOS
NECESSÁRIOS PARA OBTENÇÃO DO GRAU DE MESTRE EM TECNOLOGIA**

**Carlos Henrique Figueiredo Alves, D.Sc.
Orientador**

**RIO DE JANEIRO, RJ – BRASIL.
DEZEMBRO / 2007**

Livros Grátis

<http://www.livrosgratis.com.br>

Milhares de livros grátis para download.

SUMÁRIO

INTRODUÇÃO	1
I- REVISÃO BIBLIOGRÁFICA	3
Considerações iniciais	3
Ultra-Som	4
Telemedicina no Brasil.....	5
Trabalhos Realizados.....	8
AVET.....	8
CSVTool	10
DynaVideo	11
IMAGESERVER	12
Revisão Tecnológica	13
Desenvolvimento de Software	13
Elementos Computacionais	16
Percepção Multimídia	37
Software Livre	38
II- FUNDAMENTOS TEÓRICOS	41
Arquivos Multimídia	41
Tipos Básicos	41
Áudio.....	44
Imagem Estática	45
Imagem Dinâmica.....	48
Compressão.....	49
Algoritmos de Compressão	51
Compressão de Vídeo	60
Compressão de Áudio	64
Outros Formatos de Compressão	66
Transmissão Multimídia	69
Limitações do Modelo do Melhor Esforço.....	69
Desempenho	71
Aplicações em Tempo Real.....	72
Classificação das Aplicações	72
Qualidade de Serviço (QoS).....	73
Aplicações Multimídia	75
H.323	76
SIP	78

JMF.....	82
JAI.....	85
III- MATERIAIS E MÉTODOS.....	87
Descrição Funcional.....	87
Concepção	88
Identificação dos Casos de Uso	88
Diagrama de Contexto.....	91
Planejamento das Iterações	92
Análise dos casos de uso.....	93
Caso de Uso Manter Laudo.....	93
Caso de Uso Pesquisar Laudo	95
Caso de Uso Realizar Procedimento.....	96
Caso de Uso Gravar Procedimento.....	98
Caso de Uso Abrir Procedimento Local.....	99
Caso de Uso Transmitir Procedimento On-Line	101
Caso de Uso Transmitir Procedimento Off-Line	103
Caso de Uso Abrir Procedimento à Distância	104
Caso de Uso Ajuste Cognitivo	106
Caso de Uso Analisar Laudo	108
Caso de Uso Chat	109
Diagrama de classe de domínio	110
Arquitetura Proposta.....	112
Arquitetura Técnica.....	112
Arquitetura Aplicação.....	116
IV- RESULTADOS.....	126
UC1 - Manter Laudo.....	126
UC2 - Pesquisar Laudo	127
UC3 - Realizar Procedimento.....	128
UC4 - Gravar procedimento	129
UC5 - Abrir Procedimento Local.....	130
UC6 - Transmitir procedimento on-line.....	131
UC7 - Transmitir procedimento off-line	132
UC8 - Abrir procedimento à distância.....	133
UC9-Ajuste cognitivo	134
UC10-Analisar Laudo	137

UC11-Chat	138
Modelo de Compressão	139
V- DISCUSSÃO	143
CONCLUSÃO	146
Trabalhos Futuros	147
BIBLIOGRAFIA.....	149

Ficha Catalográfica elaborada pela biblioteca Central do CEFET-RJ

P324 Paula, Angelo Márcio de.
Gerenciamento, análise e transmissão de imagens, on-line, via TCP/IP /
Angelo Márcio de Paula. – 2007.
xi, 152 f.: il.color., tabs.; enc.

Dissertação (Mestrado) Centro Federal de Educação Tecnológica Celso
Suckow da Fonseca, 2007.
Bibliografia: f. 149-152.

1. Processamento de imagens-Técnicas digitais. 2. Ultra-sonografia-
Qualidade da imagem. 3. Sistemas multimídia. I. Título.

CDD 621.367

Dedicatória

Dedico este trabalho a minha esposa Érica e minhas filhas Mariana e Lara.

Agradecimentos

- Ao Professor Carlos Henrique Figueiredo Alves (D.Sc.), pelo empenho no trabalho de orientação, dedicação e incentivo que muito contribuíram para a elaboração deste trabalho.
- Ao Programa de Mestrado em Sistemas e Computação do IME-RJ, em especial ao Professor Uff Bergman (D.Sc.) pelos conhecimentos passados nas disciplinas complementares.
- Ao professores Antônio Maurício Castanheira das Neves (D.F.), Maria da Glória de Faria Leal (D.H.), Leydervan de Souza Xavier (D.C.), Marina Rodrigues Brochado (D.Sc.), Jorge Carlos Ferreira Jorge (D.Sc.) e Maurício Saldanha Motta (D.Sc.) pelas críticas e sugestões.
- Ao Reitor Professor Jessé de Hollanda Cordeiro Júnior, do Unifoa Volta Redonda, pelo apoio e incentivo para realização deste trabalho.
- Ao Pró-Reitor Professor Alexandre Habibe, do Unifoa Volta Redonda, pelo apoio e incentivo para realização deste trabalho.
- Aos funcionários Abraão Ferreira e Bráulio Tito, pela dedicação e presteza.
- Aos meus pais, Vicente de Paula e Vilma Terezinha de Paula pelo apoio, incentivo e afeto.
- Aos meus colegas do UniFoa pelo companheirismo, incentivo e aconselhamento.
- Aos professores da Universidade Severino Sombra em Vassouras-RJ por minha formação na graduação, sem a qual não teria chegado até aqui.
- E a todos aqueles que direta ou indiretamente contribuíram para a realização deste trabalho.

Resumo da dissertação submetida ao PPTEC/CEFET-RJ como parte dos requisitos necessários para a obtenção do grau de mestre em tecnologia (M.T.).

GERENCIAMENTO, ANÁLISE E TRANSMISSÃO DE IMAGENS,
ON-LINE, VIA TCP/IP.

Angelo Márcio de Paula

Dezembro de 2007

Orientador: Carlos Henrique Figueiredo Alves, D.Sc.

Programa: PPTEC

Aplicações envolvendo áudio, vídeo, imagem estática e texto, são denominadas aplicações multimídia e requerem requisitos especiais para armazenamento e transmissão. Este trabalho tem como proposta gerenciar laudos de ultra-som usando objetos persistentes, analisá-los através de técnicas de processamento de imagem e aplicar trabalhos colaborativos entre especialistas através da transmissão do procedimento em tempo real em conjunto com técnicas de comunicação on-line. A transmissão é dotada de um ajuste cognitivo que permite o médico especialista atuar na qualidade da imagem de modo a eliminar perturbações ocasionadas pela rede de comunicação. Todos requisitos foram integrados em um aplicativo desenvolvido na linguagem de programa Java e resultou em um software eficaz e moderno. Concluindo, o trabalho é um sistema que realiza transmissão de vídeos pela *Internet* de forma gerenciada, que atua na qualidade da geração da imagem, diminuindo o seu tamanho, facilitando a sua entrega, sem sofrer interferências indesejadas pelas variáveis de uma rede de comunicação.

Palavras-chave: Multimídia, Rede, Multicast, Processamento de Imagem

Abstract of dissertation submitted to PPTEC/CEFET/RJ as partial fulfillment of the requirements for the degree of Master in Technology (M.T.).

MANAGEMENT, ANALYSIS AND TRANSMISSION OF IMAGES,
ON LINE, BY TCP / IP.

Angelo Márcio de Paula

December / 2007

Supervisor: Carlos Henrique Figueiredo Alves, D.Sc.

Program: PPTEC

Applications involving audio, video, image and text, are called multimedia applications and require special requirements for storage and transmission. The architectures that dominate this matter are currently in opposition as to the standard to be adopted. This work is intended proposal manage reports of ultrasound using objects persistent, analyse them through techniques of image processing and implement collaborative work between experts through the transmission of the procedure in real time together with techniques of communication online. The transmission is equipped with an adjustment cognitive allowing the expert medical serve in image quality in order to eliminate nuisances caused by the network of communication. All requirements have been incorporated into an application developed in the language of Java program and resulted in effective and modern software. In conclusion, the work is a system that performs transmission of video over the Internet on a managed, which operates in the quality of the generation of the image, reducing its size, facilitating their delivery, without suffering unwanted interference by the variables of a communication network.

Keyword: Multimedia, Network, Multicast, Processing Image

LISTA DE ILUSTRAÇÕES

Figura I.1 – Aplicação em medicina segundo o site RUTE.....	6
Figura I.2 – Teleconsulta e telediagnóstico segundo o site RUTE	7
Figura I.3 – Rede Nacional de Ensino e Pesquisa segundo o site RUTE	7
Figura I.4 – Sistema AVET, encontrado em Serra (2001, p.117).....	9
Figura I.5 – Sistema CSVTOll, encontrado em Lima <i>et al</i> (2005)	10
Figura I.6– Sistema CSVTOll, encontrado em Lima <i>et al</i> (2005)	10
Figura I.7– Diagrama de componentes do DynaVideo segundo Paula (2001)	11
Figura I.8 – Arquitetura IMAGESERVER em Laurenson (2003).....	12
Figura I.9 – Fase de desenvolvimento	13
Figura I.10– O modelo de ciclo de vida iterativo e Incremental	14
Figura I.11 – Ciclo de vida de construção de software em Fowler e Scott (2000)	15
Figura I.12 – Construção de sistema adaptado de Booch, Jacobson e Rumbaugh (2000).....	16
Figura I.13 – Modelo simplificado de banco de dados	17
Figura I.14 – Modelo Java	19
Figura I.15 – Camada TCP/IP	22
Figura I.16 – Endereçamento IP	23
Figura I.17 – Protocolos do modelo TCP/IP	24
Figura I.18 – Exemplo de protocolo do modelo TCP/IP	25
Figura I.19 – Endereço IP e Port	31
Figura I.20 – Remote Method Invocation	32
Figura I.21 – Representação do conhecimento em Castanheira (informação verbal) ¹	38
Figura II.1 – Marshalling de argumentos em Davie e Peterson (2004, p. 392)	41
Figura II.2 – Exemplo de imagem estática (pâncreas)	45
Figura II.3 – Algoritmos de compressão	50
Figura II.4 – Diagrama de blocos JPEG adaptado de Tanenbaum (2003b, p. 349)	55
Figura II.5 – Diagrama de blocos JPEG segundo Davie e Peterson (2004, p. 403)	59
Figura II.6 – Quadros I,P e B do MPEG em Davie e Peterson (2004, p. 407)	62
Figura II.7 – Buffer de reprodução adaptado de Kurose e Ross (2006, p.458).....	69
Figura II.8 – Tipos de atrasos segundo Kurose e Ross (2006, p. 29)	70
Figura II.9 – Latência x Largura de Banda segundo Davie e Peterson (2004, p.33)	71
Figura II.10 – Classificação das aplicações, segundo Davie e Peterson (2004, p.362)	73
Figura II.11 – Arquitetura H.323	76
Figura II.12 – Topologia H.323 segundo Gomes (2003, p.59)	78
Figura II.13 – Topologia SIP	79
Figura II.14 – Topologia SIP em um mesmo domínio, adaptado de Ferreira <i>et al</i> (2006)	81
Figura II.15 – Topologia SIP em domínios diferentes, adaptado de Ferreira <i>et al</i> (2006)	81
Figura II.16 – Modelo de processamento multimídia	82
Figura II.17 – JMF aplicado a ultra-som	83
Figura II.18 – modelo JMF	85
Figura II.19 – Processamento de imagem.....	86
Figura III.1 – Fronteira do sistema.....	88
Figura III.2 – Diagrama de contexto de caso de uso	91
Figura III.3 – Diagrama de Seqüência: Pesquisar laudo	94
Figura III.4 – Diagrama de Seqüência: Pesquisar laudo	96
Figura III.5 – Diagrama de Seqüência: Realizar procedimento	97
Figura III.6 – Diagrama de Seqüência: Gravar procedimento	99
Figura III.7 – Diagrama de Seqüência: Abrir procedimento local	100
Figura III.8 – Diagrama de Seqüência: Transmitir procedimento on-line	102
Figura III.9 – Diagrama de Seqüência: Transmitir procedimento off-line.....	104
Figura III.10 – Diagrama de Seqüência: Abrir procedimento à distância	105
Figura III.11 – Diagrama de Seqüência: Ajuste cognitivo	107
Figura III.12 – Diagrama de Seqüência: Analisar Imagem	109
Figura III.13 – Diagrama de classe domínio	111

Figura III.14 – Arquitetura técnica	113
Figura III.15 – Protocolos	113
Figura III.16 – Modelo Maleta	115
Figura III .17– Modelo – Visão - Controlador	117
Figura III.18 – Sobrecarga de chamadas em objetos remotos	119
Figura III.19 – Aplicação dos padrões Facade e Value Object	120
Figura III.20 – Negócio acessando base de dados	122
Figura III.21 – Exemplo do padrão DAO	123
Figura III.22– Arquitetura da aplicação	125
Figura IV.1– Protótipo do caso de uso Manter Laudo	126
Figura IV.2– Protótipo do caso de uso Pesquisar Laudo	127
Figura IV.3– Protótipo do caso de uso Realizar Procedimento	128
Figura IV.4– Protótipo do caso de uso Gravar Procedimento	129
Figura IV.5– Protótipo do caso de uso Abrir Procedimento Local	130
Figura IV.6– Protótipo do caso de uso Transmitir Procedimento On-Line	131
Figura IV.7– Protótipo do caso de uso Abrir Procedimento a distância	132
Figura IV.8– Protótipo do caso de uso Transmitir Procedimento Off-Line	133
Figura IV.9– Protótipo do caso de uso Ajuste Cognitivo (melhor caso)	135
Figura IV.10– Protótipo do caso de uso Ajuste Cognitivo (pior caso)	136
Figura IV.11– Protótipo do caso de uso Analisar Laudo	137
Figura IV.12– Protótipo do caso de uso Chat	138
Figura IV.13 – Imagem gravada no padrão GIF	141
Figura V.1 – Inovação do produto	145

LISTA DE TABELAS

Tabela I.1 Frequências utilizadas em exames de ultra-som.....	4
Tabela I.2 Imagens médicas em Neto, Oliveira e Valeri (2003).....	5
Tabela I.3 – Padrões J2EE	21
Tabela I.4 – Espaço de endereçamento IP	23
Tabela I.5 – Adaptação da QoS das aplicações segundo Carvalho e Deusdado (2004)	37
Tabela I.6 – Lista de licenças em Hexsel (2002).....	39
Tabela I.7 – Lista de produtos livres e/ou código aberto	40
Tabela II.1- Tipos primitivos em Java em Ernest, Heller, Roberts. (2000, p.8).....	42
Tabela II.2 –Tabela de quantização JPEG em Davie e Peterson(2004, p.404).....	58
Tabela II.3- Resolução de vídeos em Costa (2007, p. 18)	61
Tabela II.4- Exemplos de soluções H.323	77
Tabela II.5- Exemplos de soluções SIP	80
Tabela II.6 – Valores típicos para processar vídeo em Java Media Framework (1999, p.5).....	83
Tabela II.7 – Valores típicos para processar áudio Java Media Framework (1999, p.6).....	84
Tabela III.1 – Definição do negócio e das fronteiras do sistema	87
Tabela III.2- Identificação dos usuários do sistema.....	88
Tabela III.3 – Planejamento das iterações	92
Tabela III.4 – Manter Laudo	93
Tabela III.5 – Pesquisar Laudo.....	95
Tabela III.6 – Realizar Procedimento	97
Tabela III.7 – Gravar Procedimento	98
Tabela III.8 – Abrir Procedimento Local.....	100
Tabela III.9 – Transmitir Procedimento On-Line.....	101
Tabela III.10 – Transmitir Procedimento Off-Line.....	103
Tabela III.11 – Abrir Procedimento à Distância	105
Tabela III.12 – Ajuste Cognitivo	107
Tabela III.13 – Analisar Laudo.....	108
Tabela III.14 – Chat	110
Tabela IV.1- Análise da imagem 640x480 do fígado.....	139
Tabela IV.2- Análise das imagens 640x480 de biomicroscopia	140

INTRODUÇÃO

O cerne deste trabalho é aplicar conceitos de desenvolvimento de software que usam a *Internet* para transmissão de dados multimídia com informações usadas na medicina, obedecendo a principal característica dos dados baseados no tempo, que é a entrega dentro de parâmetros de qualidade sensíveis para percepção da visão humana. A fonte de dados a ser transmitida são imagens provenientes de aparelhos de biomicroscopia, que serão consultados *on-line* por uma rede de médicos caracterizando uma telemedicina.

A utilização de imagens médicas digitais já é uma realidade em centros evoluídos e está em crescimento na maioria dos hospitais e clínicas do mundo, tornando imperativo o desenvolvimento de técnicas que viabilizem o armazenamento e a transmissão destas imagens em redes de computadores.

A pesquisa pretende ser criteriosa quanto às informações multimídia, não pretendemos classificá-la somente como áudio e vídeo, como encontrada em algumas literaturas, mas também fornecer um tratamento amplo quanto às informações tradicionais e imagens estáticas, aplicando inclusive técnicas de processamento de imagens para auxiliar na análise de laudos.

A demanda por software especialista é grande, sendo necessário construí-lo conforme as regras de negócio específico da organização. Logo, a pesquisa levará em conta as aplicações já existentes mostrando a possibilidade em adaptá-las aos requisitos do projeto maior, que é a construção de um aparelho de análise de ultra-som portátil com transmissão do procedimento em tempo real.

O setor de telecomunicação com suas redes ligando o mundo, impulsionou o conceito totalmente difundido hoje em dia chamado *Internet* e acima das redes, estão os *softwares* que representam hoje em dia uma oportunidade de mercado com agregação de valor caracterizando o surgimento de produtos.

Desta forma, o setor de telecomunicações torna-se impulsionador dessa síntese, pois esse trabalho só foi possível devido ao êxito na modernização, ofertas de novos serviços, dentre outros e principalmente abriu uma cadeia de novos valores que interagem com o setor de telecomunicações, podendo até dizer que o sucesso desse setor abriu portas para o surgimento de um tipo de negócio chamado “informação”, empresas puderam se interligar tornando o acesso rápido à informação de forma segura.

Esse trabalho tem o objetivo de criar uma aplicação multimídia para gerenciamento, análise e transmissão de imagens de exames para diagnósticos utilizando ultra-som. A aplicação vai capturar um sinal do sensor, processá-lo, enviar via rede TCP/IP e recebê-lo em um cliente conectado na rede. O cliente atuará na imagem recebida através de técnicas de processamento de imagens permitindo inclusive ajustar a qualidade do sinal recebido diretamente na fonte emissora. Isto será um avanço, principalmente na área médica, pois proporcionará uma segurança no relato do laudo, visto que, a qualidade da imagem será controlada pelo especialista que se encontra à distância, e não pela fonte emissora ou modelos matemáticos. Desta forma, o especialista não será exposto a erros de interpretação causados pela degradação na imagem sem a sua autorização.

A primeira parte do estudo, visa analisar o estado da arte já pesquisado, tais como os padrões *Session Initiation Protocol* (SIP), H.323, os *frameworks Java Media Framework* (JMF) e *Java Advanced Image* (JAI) para construção de uma aplicação multimídia, modelos matemáticos adaptativos para garantir a transmissão em tempo real e modelos que permitem atuar nos *pixels* da imagem através de técnicas de processamento de imagem que ajudarão na análise da mesma.

A segunda parte do estudo visa melhorar pontos ainda não bem definidos gerando uma proposta prática a ser aplicado na camada de alto nível do modelo TCP/IP, denominado camada de aplicação, onde apontará vários outros trabalhos futuros.

O estudo envolverá além de algoritmos complexos, a influência do ser humano na operação do software. O sistema apresentará uma interface cuja função é aproximar usuários que estão fisicamente separados, através de técnicas já conhecidas, tais como, *chat*, transmissão de áudio, transmissão de vídeo, todos em uma única aplicação. Dando ao homem a oportunidade de influenciar no laudo da imagem gerada. Logo, qualquer modelo para ter sucesso deve levar em consideração que está sendo operado por seres humanos.

I- REVISÃO BIBLIOGRÁFICA

Considerações iniciais

Este trabalho nasceu devido à necessidade de uma aplicação confiável para utilização em telemedicina que faça gerenciamento, análise e distribuição de dados multimídia de forma confiável e segura. A equipe além de usar essa ferramenta com o propósito de distribuição de resultados à distância e permitir a interação de pesquisadores de outras instituições, pretende também, obedecendo a uma dinâmica temporal das inovações de processo e de produto, onde segundo Freire (2004, pg 25), tem início a partir da oportunidade de inserção de um novo produto no mercado, oferecê-la comercialmente.

A aplicação em questão tem a necessidade de transmitir informação multimídia (texto, voz, imagem estática e vídeo) *on-line* e *off-line* em tempo real e permitir uma interação entre os participantes usando funcionalidades comuns no mercado de software que permitam uma conversa *on-line* em texto e voz, além de permitir aplicar processamento na imagem com o objetivo de tornar o contato entre os participantes como se estivessem na mesma sala.

Alguns fatores estão contribuindo para a realização deste trabalho:

- Fatores humanos:
 - Região demográfica brasileira extensa.
 - Exames médicos de última geração somente nos grandes centros.
 - Especialistas estão situados nos grandes centros.
 - Necessidade de atender população de regiões distantes.
- Fatores organizacionais:
 - Necessidade de manter em banco de dados os laudos dos procedimentos.
 - Necessidade de acesso controlado aos procedimentos.
 - Necessidade manter acesso somente a pessoas autorizadas no sistema.
- Fatores técnicos:

- Não existe um padrão unânime para realizar transmissão multimídia.
- Os arquivos gerados nos exames são grandes em bytes.
- Dificuldade de transmissão em redes de baixa velocidade.
- Redes de computadores denominadas banda larga é uma realidade no Brasil.
- Não existe comercialmente uma aplicação capaz de realizar todos os requisitos exigidos pelos sistemas.

Os fatores analisados oferecem uma oportunidade para desenvolvimento de pesquisa, através da criação de um produto novo a partir da integração de soluções já oferecidas aplicando melhorias na adaptação no vídeo recebido pela estação cliente.

Ultra-Som

Segundo Alves, Ritter e Shung (2000), o uso de técnicas de ultra-som na medicina vem crescendo nas últimas décadas e tem se destacado como uma das mais promissoras áreas em diagnóstico médico através de imagens. Para aplicação em seres humanos, freqüências da ordem de poucos MHz têm sido utilizadas. A Tabela I.1 mostra alguns exemplos de freqüências utilizadas:

Tabela I.1 Freqüências utilizadas em exames de ultra-som

APLICAÇÃO	FREQUÊNCIA
Abdômen de adultos	3,5 MHz
Fluxo em vasos sanguíneos	entre 7,5 e 10,0 MHz
Oftalmologia e dermatologia	entre 20 e 100 MHz

O ultra-som está contribuindo na construção de diagnósticos essenciais para medicina. A qualidade da imagem é a parte mais importante em se tratando de imagens médicas. A escolha do melhor algoritmo para compressão deve levar em consideração se as perdas não afetarão a qualidade da imagem, de modo a não induzir os especialistas a erros na interpretação da mesma, e conseqüentemente um diagnóstico não preciso resultando prejuízo ao paciente. A qualidade da imagem também está relacionada ao uso correto dos sensores para aquisição dos dados ainda na parte analógica.

A engenharia de transdutores de ultra-som estuda continuamente a construção de sensores com freqüências cada vez maiores, de modo a melhorar a

qualidade da imagem, principalmente em se tratando da quantidade de quadros por segundo que segundo Alves *et al* (2000) na maioria das vezes, necessita-se de sistemas em tempo real com o maior número possível de quadros por segundo, originando sistemas com maior definição de imagem, sabendo-se que 30 quadros por segundo é um valor ideal para percepção humana.

As imagens médicas não são limitadas a ultra-som. Este trabalho está levando em consideração imagem para ultra-som, mas o trabalho pode ser estendido para demais aplicações médicas, tais como: radiografia, mamografia, tomografia computadorizada, ressonância magnética e medicina nuclear. A Tabela I.2 apresenta uma lista típica de imagens provenientes de diversos aparelhos, respectivamente com a resolução utilizada e espaço necessário para armazenamento e transmissão.

Tabela I.2 Imagens médicas em Neto, Oliveira e Valeri (2003)

TIPO DE IMAGEM	RESOLUÇÃO TÍPICA	TAMANHO TÍPICO
Radiografia	2048x2048x12 bits	32 MBytes
Mamografia	4096x5120x12 bits	160 Mbytes
Tomografia	512x512x12x n.º imagens	15 MBytes
Ressonância	256x256x12x50 imagens	6.3 MBytes
Ultra-sons	256x256x8 bits	1.5 MBtes
Medicina nuclear	128x128x8 bits	0.4 MBytes

Telemedicina no Brasil

Existe no Brasil uma iniciativa concreta de telemedicina que pode ser explorada em <www.rute.rnp.br>, segundo esse site, essa é uma iniciativa da Rede Nacional de Ensino e Pesquisa (RNP) e o Ministério da Ciência e Tecnologia (MCT), com apoio da Associação Brasileira de Hospitais Universitários (ABRAHUE), e da Rede Universitária de Telemedicina (RUTE). Uma rede com o objetivo de promover a interconexão e a colaboração entre grupos de pesquisa em saúde em todo o país apoiando o aprimoramento de projetos em telemedicina já existentes e incentivando o surgimento de futuros trabalhos nas universidades.

Financiada pela Finep, a rede interconectará instituições de ensino e pesquisa que já possuem trabalhos em telemedicina. A infra-estrutura de alta capacidade e qualidade adequadas será fornecida pela RNP por meio de sua infra-estrutura de rede. Inicialmente, as vinte instituições participantes de RUTE são:

- Hospital São Paulo e Escola Paulista de Medicina - Unifesp, (São Paulo)
- Hospital das Clínicas - FM USP, (São Paulo)
- Hospital Universitário de São Paulo/LSITEC - USP (São Paulo)
- Hospital das Clínicas da Unicamp (Campinas/SP)
- Instituto Dante Pazzanese de Cardiologia (São Paulo)
- Hospital Universidade de Marília -Unimar (Marília/SP)
- Hospital Universitário Pedro Ernesto - UERJ (Rio de Janeiro)
- Fundação Oswaldo Cruz, (Rio de Janeiro)
- Hospital Universitário Prof. P. E. de São Thiago - UFSC (Florianópolis)
- Hospital das Clínicas Prof. Arnóbio Marques - UFPE (Recife)
- Hospital Universitário Walter Cantídio - UFC (Fortaleza)
- Hospital da Irmandade da Santa Casa de Misericórdia, (Porto Alegre)
- Hospital Universitário Getúlio Vargas - UFAM (Manaus)
- Hospital das Clínicas – UFMG, (Belo Horizonte)
- Hospital Universitário - UFES (Vitória)
- Hospital Universitário Prof. Alberto Antunes - UFAL (Maceió)
- Hospital Universitário Prof. Edgar Santos - UFBA (Salvador)
- Hospital das Clínicas - UFMA (São Luís)
- Hospital Universitário Lauro Wanderley -UFPB (João Pessoa)
- Hospital Universitário - UFPR (Curitiba)

A integração dos hospitais universitários na RUTE viabilizará a troca de informações médicas, estudo de caso, consultas por videoconferência, análise de sinais e imagens médicas, radiologia por imagem, sala de laudo virtual, diagnósticos e cursos de capacitação médica à distância, dentre outros, promovendo a melhoria do atendimento especializado à população, educação e redução de custos com comunicação e deslocamento. As Figuras II.1 e II.2 mostram algumas iniciativas.

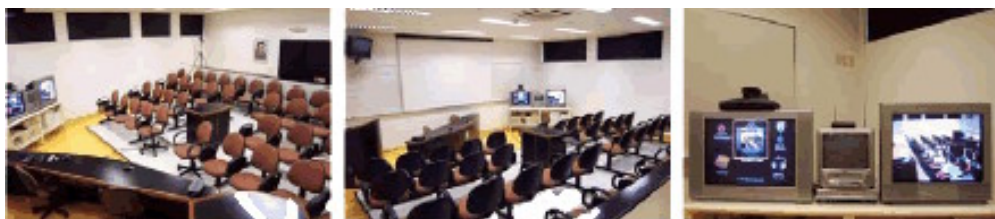


Figura I.1 – Aplicação em medicina segundo o site RUTE



Figura I.2 – Teleconsulta e telediagnóstico segundo o site RUTE

Nesse sentido, a RNP pretende que RUTE promova o desenvolvimento de aplicações de saúde que possam se valer da rede avançada e estimular a integração de hospitais universitários e redes colaborativas de saúde.

Estando as unidades de saúde interconectadas ocorrerá uma diminuição no custo de transporte da população até os grandes centros, ocorrerá uma melhor interação entre os especialistas promovendo a segunda opinião nos procedimentos. Cidades que jamais ocorreram exames, agora será possível realizar com a instalação de uma unidade móvel fora das metrópoles, causando uma inversão na população, antes elas se deslocavam até os grandes centros, agora eles vão até a cidade vizinha ao encontro da unidade móvel. A distância não será mais uma barreira para a medicina. As instituições participantes devem possuir redes de computadores que favoreça a transmissão de dados multimídia. A Figura I.3 mostra o atual panorama da Rede Nacional de Ensino e Pesquisa.

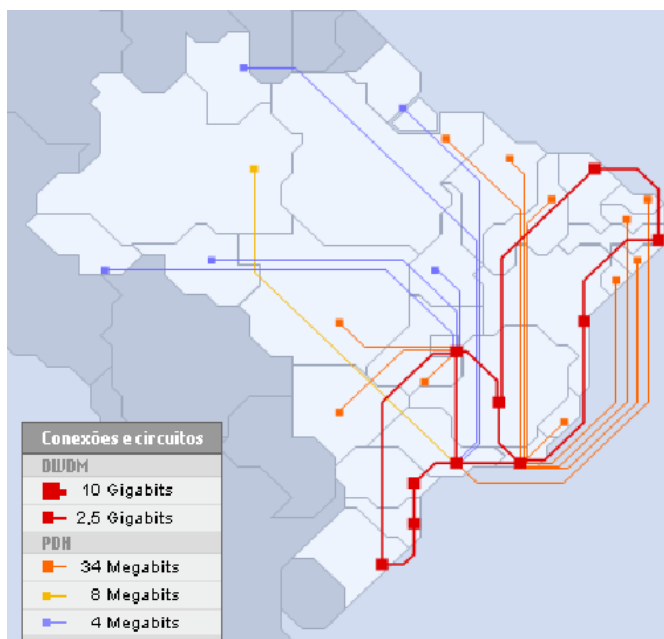


Figura I.3 – Rede Nacional de Ensino e Pesquisa segundo o site RUTE

Trabalhos Realizados

Entre os trabalhos pesquisados no meio acadêmico, destacam-se sob o ponto de vista acadêmico:

- Sistema *AVET*, desenvolvido por Serra (2001), “Uma Solução de Distribuição para Aplicações em Tempo Real no Contexto do Ensino Tecnológico à Distância”, Dissertação de Mestrado em Ciência da Computação da Universidade Federal do Ceará .
- Sistema *CSVTool*, desenvolvido por Lima et al (2005), Grupo de Computação Gráfica da PUC-RIO – Tecgraf, com o título “A Multi-user Videoconference-based Collaboration Tool: Design and Implementation Issues”, foi apresentado no “9th International Conference on Computer Supported Cooperative Work in Design (CSCWiD)”, em Coventry, U.K.
- Sistema *DynaVideo*, desenvolvido por Batista, Filho e Leite (2001), “Um Serviço de Distribuição de Vídeo baseado em Configuração Dinâmica”, apresentado no 19º Simpósio Brasileiro de Redes de Computadores, em Florianópolis, Santa Catarina.
- Sistema *IMAGESERVER*, desenvolvido por Laurenson et al (2003), com o título “*IMAGESERVER: A System For A Distributed Image Processing Application Based On Java Advanced Imaging*”, apresentado no *EFITA 2003 Conference*, na cidade de *Debrecen, Hungary*.

AVET

A solução apresentada por Serra (2001) propõe a concepção de uma solução de distribuição de recursos para aplicações de tempo real em “Ambientes Virtuais de Educação a Distância”. São também apresentadas a modelagem e a implementação de uma “Aplicação de Videoconferência”, no “Contexto da Educação Tecnológica à Distância”, levando em conta sua especificidade e utilizando a solução de distribuição de recursos. O sistema é uma aplicação de

videoconferência em um ambiente distribuído e está apoiada na plataforma *Java* pelos seguintes fatores:

- A existência de uma *API Java (JMF - Java Media Framework)* específica para o tratamento de áudio e vídeo.
- O *Java* é uma linguagem orientada a objetos, facilitando o desenvolvimento e manutenção do código.
- Possui a tecnologia *RMI (Remote Method Invocation)*, que permite a invocação de métodos em objetos distribuídos.
- Pelas perspectivas de ampliação da utilização da linguagem *Java* em áreas diversas.

A Figura I.4 mostra a arquitetura utilizada pelo sistema AVET, a apresentação da mídia está apoiada em um *applet* que roda no navegador, esse *applet* faz requisições para camada de aplicação através do protocolo *http* para troca de informações gerenciais. O controle da apresentação é realizado pelo *RMI* e a reprodução do vídeo é feita pelo protocolo *RTP*. O sistema possui um gerenciador de serviços e um servidor de distribuição de mídia baseado no *JMF*.

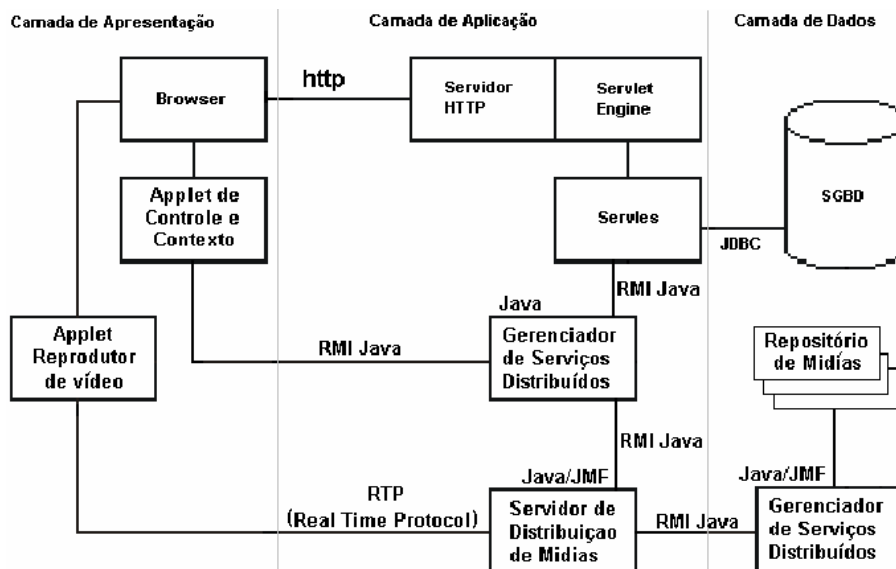


Figura I.4 – Sistema AVET, encontrado em Serra (2001, p.117)

CSVTool

A solução apresentada pelo Grupo de Computação Gráfica da PUC-RIO – Tecgraf é uma ferramenta para trabalho colaborativo multimídia projetada para ser simples, com suporte a múltiplos usuários sobre redes *anycast* e plataforma independente. Tem como destaque a estratégia adotada para controlar os usuários na colaboração usando áudio e vídeo. Essa ferramenta está apoiada nas seguintes tecnologias:

- Linguagem de Programação Java
- *Middleware Corba* para computação distribuída
- JMF (Java Media Framework)

A Figura I.5 mostra a arquitetura utilizada pelo sistema CSVToll:

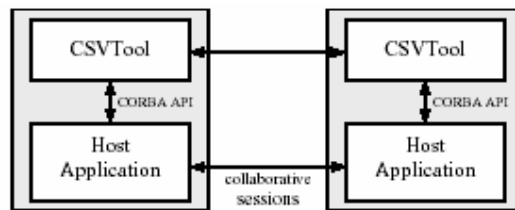


Figura I.5 – Sistema CSVToll, encontrado em Lima *et al* (2005)

A Figura I.6 Apresenta um exemplo de um trabalho colaborativo:



Figura I.6– Sistema CSVToll, encontrado em Lima *et al* (2005)

DynaVideo

O trabalho realizado por Batista, Filho e Leite (2001) no Departamento de Informática e Matemática Aplicada da Universidade Federal do Rio Grande do Norte, denominado *DynaVideo*, é uma ferramenta para apresentação de conteúdo multimídia. Este serviço foi projetado para distribuir vídeo de forma independente de formato e com suporte para interação com diferentes tipos de clientes. O serviço pode ser utilizado para distribuir vídeo em qualquer tipo de rede, entretanto um dos focos do projeto é a Internet. Uma das principais características do *DynaVideo* é a capacidade de configurar dinamicamente o serviço para atender a uma determinada demanda. Paula (2001) apresenta a tecnologia empregada no desenvolvimento e o diagrama de componentes da aplicação é apresentado na Figura I.7 :

- Suporte a banco de dados, utilizando servlets.
- Programação distribuída.
- Manuseio de vídeo e sessões RTP (*Real Time Protocol*), com a API JMF (*Java Media Framework*).
- Aplicação voltada para *Web*, com *applets*.
- Desenvolvida com a linguagem Java.

A Figura I.7 mostra o diagrama de componentes:

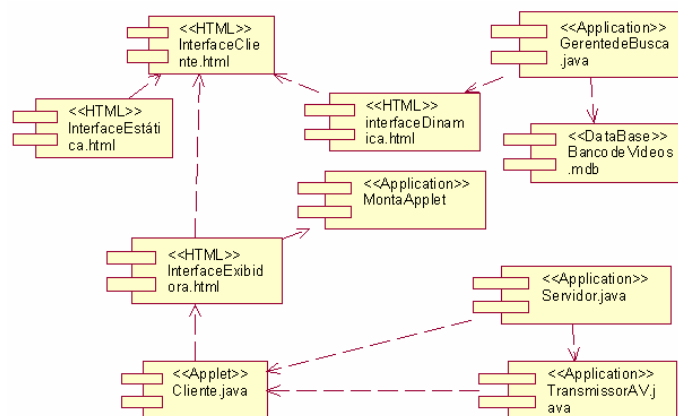


Figura I.7– Diagrama de componentes do DynaVideo segundo Paula (2001)

IMAGESERVER

Trabalho realizado por Laurenson *et al* (2003), uma parceria entre pesquisadores japoneses e chineses, é uma ferramenta para processamento de imagem. Essa ferramenta permite aplicar processamento remoto na imagem estática através de um *applet* que roda no navegador de *internet* no lado do cliente. Está apoiada na plataforma *Java* pelos seguintes fatores:

- Plataforma independente de sistema operacional.
- É possível ser acoplada a um navegador da *Internet* usando-se *applets*.
- Poderosa ferramenta para processamento de imagem usando-se JAI (*Java Advanced Image*).
- Permite operação remota através do RMI.

A Figura I.8, mostra a arquitetura do IMAGESERVER, o cliente comunica com o servidor por RMI onde é realizado o processamento da imagem. Como o RMI possui limitações quando utilizado em ambientes que usam *firewall*, um *Servlet* através do protocolo HTTP na porta 80 é invocado e esse comunicará com o servidor de imagem através do RMI e responderá para a *applet* pela mesma porta.

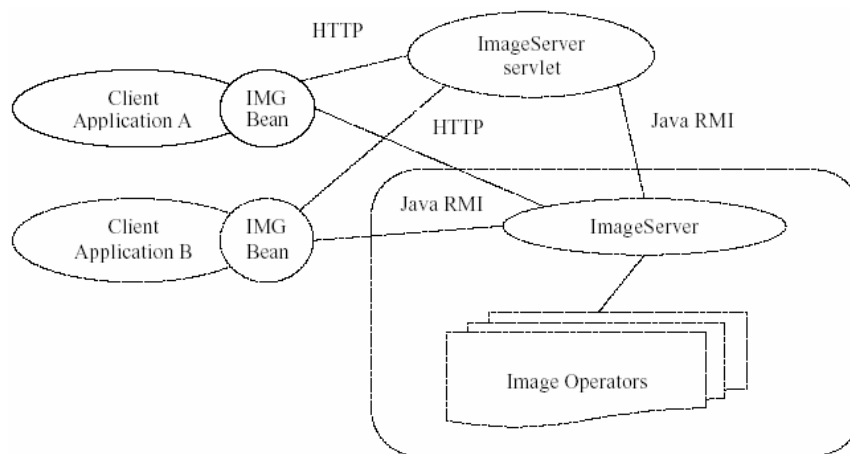


Figura I.8 – Arquitetura IMAGESERVER em Laurenson (2003)

Revisão Tecnológica

Desenvolvimento de Software

Desenvolver um trabalho que envolva a construção de *software* é necessário seguir algum processo de desenvolvimento, pois a demanda por mais serviços, novos produtos e pela melhoria da qualidade e com prazos acirrados é cada vez mais comum. E, além do aspecto puramente técnico, encontramos ainda enorme dificuldade em se gerenciar o processo de desenvolvimento.

Aragão (2002) em sua tese de mestrado apresenta um estudo e reforça a importância de um desenvolvimento planejado com qualidade e previsibilidade usando os recursos necessários para manter a eficiência do projeto em questão, logo, uma empresa de *software* bem-sucedida é aquela que fornece *software* de qualidade e capaz de atender às necessidades dos respectivos usuários. Uma empresa que consiga desenvolver esse *software* de maneira previsível e em determinado período, com utilização eficiente e eficaz de recursos, será uma empresa com um negócio viável, tal premissa vale também para o meio acadêmico.

As fases de desenvolvimento de aplicações são apresentadas na Figura I.9, essas são executadas concomitantemente de forma que problemas detectados numa certa fase modifiquem e melhorem as fases desenvolvidas anteriormente.

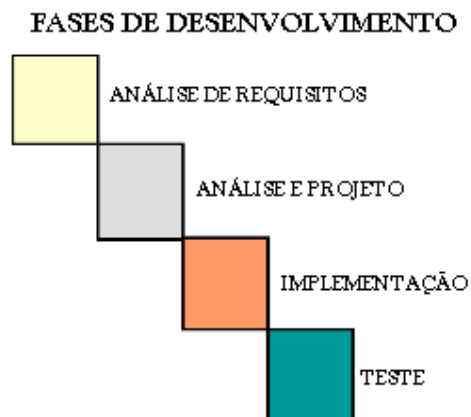


Figura I.9 – Fase de desenvolvimento

A análise de requisitos captura as intenções e necessidades dos usuários do sistema a ser desenvolvido através dos casos de uso. A análise está preocupada com as primeiras abstrações (classes e objetos) e mecanismos que estarão presentes no domínio do problema. As classes são modeladas e ligadas através de relacionamentos com outras classes, e são descritas no diagrama de classe. Na fase de projeto, o resultado da análise é expandido em soluções técnicas. Novas classes serão adicionadas para prover uma infra-estrutura técnica, tais como, a interface do usuário e de periféricos, gerenciamento de banco de dados, comunicação com outros sistemas, etc. As classes do domínio do problema modeladas na fase de análise são mescladas nessa nova infra-estrutura técnica tornando possível alterar tanto o domínio do problema quanto a infra-estrutura. Na fase de programação, as classes provenientes do projeto são convertidas para o código da linguagem orientada a objetos escolhida. Um sistema normalmente é rodado em testes de unidade, integração, e aceitação. Os testes de unidade são para classes individuais ou grupos de classes e são geralmente testados pelo programador. Os testes de integração são aplicados já usando as classes e componentes integrados para se confirmar se as classes estão cooperando uma com as outras como especificado nos modelos.

O desenvolvimento iterativo e incremental apresentado na Figura I.10 é o processo de construção de sistema de software feito em pequenos passos, usando pequenos ciclos de desenvolvimento chamados de “iteração”, suas fases são entrelaçadas, as especificações evoluem junto com o sistema e suportam os requisitos parcialmente definidos.

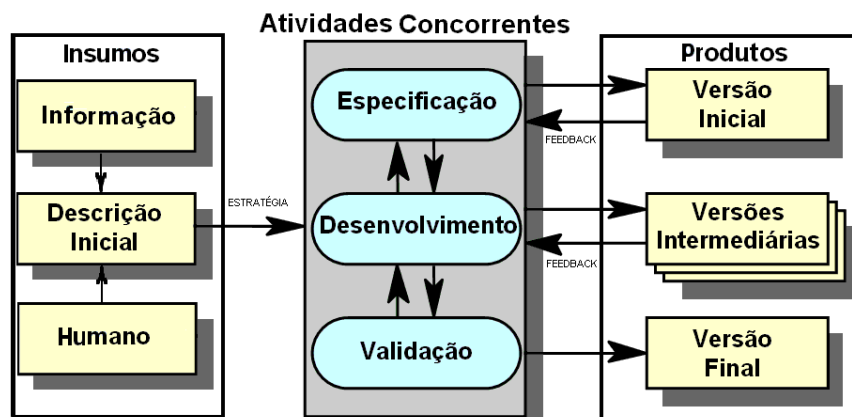


Figura I.10– O modelo de ciclo de vida Iterativo e Incremental

Segundo Ahmed e Umrysh (2001, p.64), e apresentado por Aragão (2002), O ciclo de vida do *software* é dividido em fases onde o resultado de cada fase é a geração de um produto. Cada fase é composta de uma sucessão de fases que são Concepção; Elaboração; Construção e Transição. A Figura I.11 mostra o ciclo de vida de um projeto

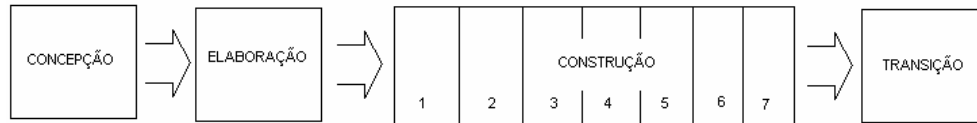


Figura I.11 – Ciclo de vida de construção de software em Fowler e Scott (2000)

O propósito da Concepção é estabelecer qual o negócio para um novo sistema ou para uma atualização de um sistema existente. O propósito da Elaboração é analisar o domínio do problema, estabelecer uma base arquitetural apropriada, identificar os elementos de maior risco do projeto e desenvolver um plano abrangente de como será completado o projeto. O propósito da construção é desenvolver incrementalmente um produto de software completo que está pronto para a transição para a comunidade de usuários. O propósito da transição é promover a transição do produto de software para a comunidade de usuários

O planejamento é o ponto de controle e deve ser realizado pelo gerente de projeto baseado em feedback de toda equipe de desenvolvimento. Alguns pontos não podem ficar de fora desse planejamento tais como:

- Identificar e priorizar os riscos significativos do projeto junto com os clientes
- Selecionar um número pequeno de cenários que endereçam diretamente os riscos prioritários
- Os cenários escolhidos são escolhidos para:
 - Os desenvolvedores identificarem o que deve ser implementado pela iteração
 - Os testadores desenvolverem os planos e procedimentos de teste para a iteração
 - Os clientes realizarem o aceite do produto gerado na iteração
- Ao final da iteração:
 - Determinar que riscos foram reduzidos ou eliminados

- Determinar se novos riscos foram descobertos
- Atualizar o plano para as iterações restantes

A Figura I.12 mostra o modelo de construção de software.

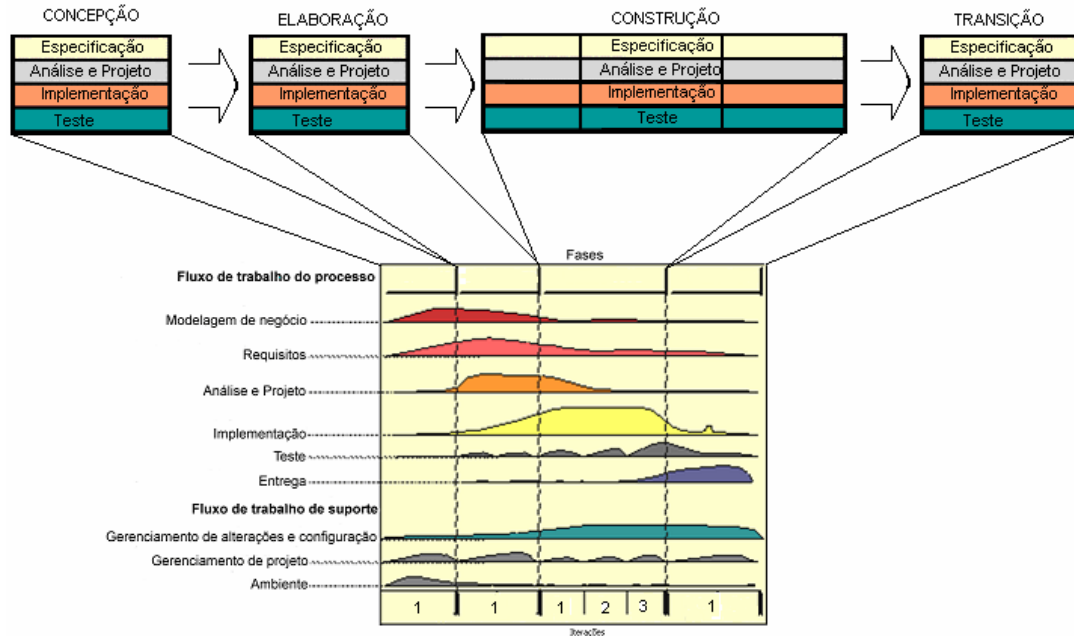


Figura I.12 – Construção de sistema adaptado de Booch, Jacobson e Rumbaugh (2000)

Elementos Computacionais

Armazenamento

“Um banco de dados é uma coleção de dados relacionados”. (ELMASRI E NAVATHE, 2005, p.4). Trata-se de um elemento com a capacidade de armazenar as informações no sistema, é de fundamental importância, pois essa tecnologia vai armazenar os dados que justificam o sistema.

Um banco de dados pode ser qualquer elemento que seja capaz de armazenar a informação que é manipulada no sistema. Atualmente existem vários tipos de armazenadores de informação, entre eles destacam-se o banco de dados relacional, fabricado/construído pela Oracle, IBM (DB2), Microsoft (SQL Server) e outros. Até armazenadores que usam documentos do tipo XML e o próprio

sistema E/S. A Figura I.13 mostra a relação simplificada do sistema e o banco de dados.

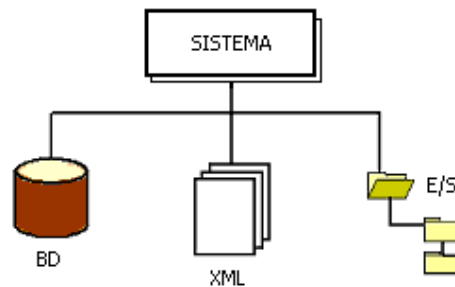


Figura I.13 – Modelo simplificado de banco de dados

Dada uma informação multimídia armazenada, por exemplo um vídeo, deseja-se encontrar uma determinada imagem em alguma parte do mesmo, dada uma imagem padrão. É necessário então que o sistema gerenciador do banco de dados seja especialista para tal propósito. Porém, o que se vê atualmente não contrasta com as exigências dos dados multimídia. Não existem SGBDs projetados exclusivamente para o propósito de gerenciar dados multimídia, não há nenhum que possua o conjunto de funcionalidades exigidas para dar suporte completamente a todas as aplicações de gerenciamento de informação multimídia (ELMASRI E NAVATHE, 2005, p.668).

Tanto Angelo (2007) quanto Camargo (2005), que desenvolveram aplicações multimídias em suas dissertações de doutorado e mestrado respectivamente, utilizaram a base de dados *MySQL* para o armazenamento dos dados multimídia, esse banco de dados tem sido utilizado por vários órgãos, acadêmicos e comerciais devido a sua alta performance, segurança e ser de isento de custos de licença

O modelo objeto relacional realiza persistência de objetos em um modelo relacional. Trabalhar com uma linguagem orientada a objetos como o Java e ter que persistir os objetos em um banco de dados relacional representa uma incompatibilidade de tecnologia. Tentar resolver essa incompatibilidade desenvolvendo soluções própria, levará a um tempo extra de desenvolvimento, tirando o desenvolvedor do seu foco principal para resolver problemas de tecnologia.

Júnior (2006) realizou um estudo de caso e aplicou o *framework Hibernate* na camada de negócio da aplicação AulaNet da PUC-RIO. Esse *framework*

relaciona as classes de entidade Java às tabelas da base de dados do banco de dados relacional, inclusive os tipos de dados de Java aos tipos de dados do SQL, Além de fornecer facilidades na manipulação de *queries* em um modelo voltado para trabalhar com objetos, denominado *Hibernate Query Language* (HQL).

Linguagem de Programação

O Java é uma linguagem de programação com recursos para aplicações *desktop* e *web*. Atualmente, existem inúmeras ferramentas de desenvolvimento completas sem custo para o desenvolvedor, principalmente para os projetos acadêmicos, que devido aos extensos recursos disponibilizados e ausência de custos de licenças, se tornou unanimidade entre os trabalhos que envolvem desenvolvimento de aplicação.

A tecnologia Java é um ambiente interpretado por dois motivos: Primeiro, a velocidade de desenvolvimento, pois reduz o ciclo de compilação, vinculação, carga e teste. Segundo, a Portabilidade de códigos, pois um ambiente interpretado faz chamadas específicas, em nível de sistema operacional, no lugar do ambiente de tempo de execução.

Essa tecnologia também elimina as práticas de codificação que afetam a eficiências dos códigos, tais como, Aritmética de ponteiros; alocação / desalocação de memória; fornece recurso para que os programas executem mais de um thread de atividade; fornece recursos para alterar dinamicamente os programas durante o tempo de execução, permitindo que descarreguem módulos de código e fornece recursos de verificar os módulos de códigos carregados para evitar problemas de segurança. Esse recurso é resumido na máquina virtual Java (JVM), na coleta de lixo e na segurança de códigos.

Várias linguagens de programação permitem a alocação dinâmica de memória no tempo de execução. O processo de alocação de memória varia de acordo com a sintaxe da linguagem, mas sempre envolve o retorno de um ponteiro ao endereço inicial do bloco de memória. Quando a memória alocada não for mais necessária, é aconselhável desalocar a memória para evitar que o programa fique sem memória.

Nas linguagens C e C++ (entre outras), o desenvolvedor do programa é responsável pela desalocação da memória. Às vezes, isso é difícil, especialmente

porque nem sempre se sabe, quando a memória precisa ser liberada. Quando não houver mais memória para ser alocada no sistema, é possível que ocorram falhas em programas que não desalocam a memória.

Ao utilizar o Java, o programador não mais terá a responsabilidade de liberar a memória, pois essa linguagem fornece um *thread* de segundo plano, em nível de sistema, que registra toda a alocação de memória e mantém uma contagem do número de referências feitas a cada ponteiro de memória. Durante os ciclos de inatividade no tempo de execução da JVM, o *thread* de coleta de lixo verifica se há algum ponteiro de memória em que o número de referências tenha sido reduzido para zero. Se houver, a memória marcada, o *thread* de coleta de lixo a limpará. A coleta de lixo ocorre automaticamente durante a vida útil de um programa Java diminuindo a possibilidade de perdas de memória.

Os arquivos da linguagem Java são compilados para que sejam convertidos do formato de texto em que os programadores os criam para um conjunto de códigos de bytes independente de máquina.

No tempo de execução, os códigos de *bytes* que constituem o programa Java são carregados, verificados e, em seguida, executados em um interpretador. O interpretador tem duas funções: executar o código de *bytes* Java e fazer as chamadas apropriadas, através de um aplicativo em tempo de execução do sistema, para o hardware subjacente. A Figura I.14 mostra o modelo de Java de desenvolvimento.

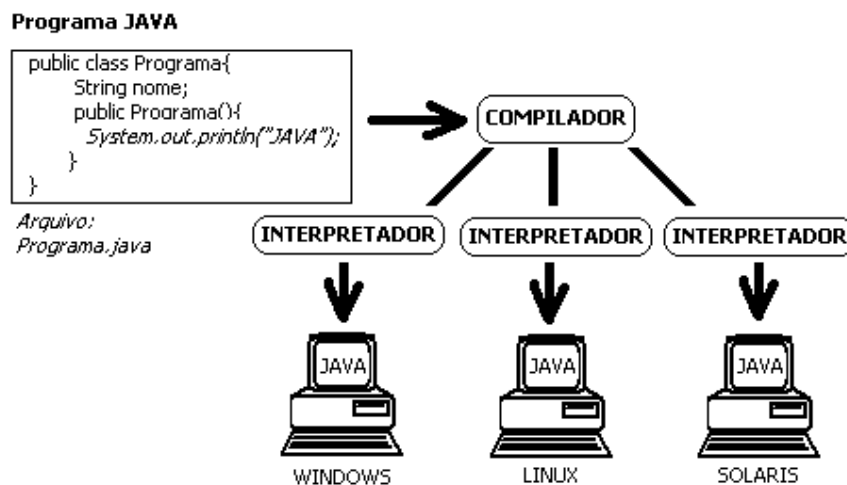


Figura I.14 – Modelo Java

Padrões de Projetos

Júnior (2005) utilizou o J2EE no desenvolvimento de sua tese de mestrado para reduzir o esforço de implementação das funcionalidades típicas de sistemas distribuídos. Para um aproveitamento com sucesso, o mesmo levou em consideração os padrões de projetos desta tecnologia. Onde concluímos que um projeto de sistema de software é agregado de um “*plus*” quando desenvolvido com soluções reutilizáveis de software amplamente disponibilizado pelos *Padrões de Projeto*.

Os padrões de projeto surgiram em 1970 com Christopher Alexander, o qual documentou padrões relacionados à arquitetura e engenharia civil. A comunidade de software baseou-se nestas idéias e padronizaram soluções para problemas comuns enfrentados pelos desenvolvedores, onde cada um fornecia a sua própria solução.

Padrões em software atingiram popularidade com o livro *Design Patterns: Elements of Reusable Object-Oriented Software* por *Erich Gamma, Helm e Johnson, Richard Helm, Ralph Johnson e John Vlissides*, essa obra também é conhecida como *Gang of Four*, ou simplesmente *Gof (Gangue dos quatros)*. Esses autores não inventaram os padrões, eles pesquisaram ao longo do mundo e julgaram as melhores soluções para os problemas comuns enfrentados pelos desenvolvedores. (ALUR, CRUPI e MALKS, 2001, p.7).

No decorrer do desenvolvimento de uma aplicação, é comum nos depararmos com problemas que precisam de conhecimentos estratégicos para ser resolvidos. Cada problema que encontramos, imediatamente é considerado inúmeras formas de resolvê-lo, incluindo soluções de sucesso já vividas ou soluções completamente novas.

O segredo é documentar cada solução aplicada e à medida que reusamos uma solução com sucesso, considera-se que temos uma solução para resolver um determinado problema. Logo os padrões de projeto só surgiram pois algumas pessoas tiveram o cuidado de documentar e compartilhar suas soluções com outros desenvolvedores, os quais são consideradas boas práticas de solução de problema. (DESHMUKH E MALAVIA, 2003, p.346). A Tabela I.3 mostra a relação dos padrões usados na arquitetura J2EE.

Tabela I.3 – Padrões J2EE

CAMADA	NOME DO PADRÃO
APRESENTAÇÃO	Intercepting Filter Front Controller View Helper Composite View Service to Worker Dispatcher View
NEGÓCIO	Business Delegate Value Object Session Facade Composite Entity Value Object Assembler Value List Handler Service Locator
INTEGRAÇÃO	Data Access Object Service Activator

Arquitetura TCP/IP

O TCP/IP é considerado uma arquitetura para *internet* com uma filosofia de interligação de redes de computadores cuja característica mais relevante é a total transparência, aos seus usuários, dos detalhes relativos às tecnologias e à forma com a qual essa interligação é feita e segundo Tanenbaum (2003a, p.60), “uma máquina está na *Internet* quando executa a pilha de protocolos TCP/IP, tem um endereço IP e pode enviar pacotes IP a todas as outras máquinas da *Internet*.”

O modelo TCP/IP é caracterizado por camadas fornecendo serviços específicos através de protocolos dedicados que cooperam entre si permitindo a comunicação entre as redes. A Figura I.15 mostra as camadas de aplicação, transporte, rede e enlace. As definições das camadas seguem segundo Kurose e Ross (2006, p. 37).

A camada de aplicação é composta de protocolos no nível do usuário, ou seja, onde residem as aplicações que fazem requisição e transferência de documentos pela *Web* (protocolo http), transferência de mensagens de correio eletrônico (protocolo SMTP), aplicação multimídia em rede (protocolo RTP) e outros. Essa camada permite a adição dos protocolos desenvolvidos pelos usuários.

A camada de transporte faz o controle de fluxo dos dados entre os equipamentos, ou seja, transporta os dados entre o cliente e o servidor de uma aplicação. Os serviços ocorrem de duas formas: Orientados a conexão (TCP) e não orientados a conexão (UDP). Fazendo uma analogia, o TCP equivale a uma ligação telefônica, onde é necessária uma conexão entre as partes para ocorrer à conversa, ou seja, troca de dados. O UDP equivale a um serviço postal de entrega de cartas, o destinatário não está esperando a mensagem.

A camada de rede é responsável pelo endereçamento e roteamento dos dados. Ou seja, é responsável pela movimentação de uma máquina para outra dos *datagramas*.

A camada de enlace é a interface com os métodos de acesso a vários meios físicos. Essa camada é ligada diretamente ao meio físico e fará o roteamento do *datagrama* por meio de uma série de comutadores de pacotes denominados roteadores entre a origem e o destino.



Figura I.15 – Camada TCP/IP

Classes de Rede

Cada computador ligado à rede possui um identificador que funciona como um endereço de localização. O campo de endereço possui 32 bits, denominado *IPv4* (já existe o estudo para criação do *IPv6*, o que não influenciará o cerne deste trabalho). Esse endereçamento está dividido em endereço de rede e endereço de *host* e são distribuídos em classes A, B, C, D e E. A Figura I.16 mostra os três tipos de endereços do modelo TCP/IP, endereçamento *unicast* (A, B e C), endereçamento *multicast* (D) e endereçamento especial ou reservado (E). A Tabela I.4 resume os endereços IP possíveis para cada classe.

Segundo Serra (2001), a conexão *multicast* explora as capacidades de compartilhamento das redes locais, onde potencialmente, cada estação recebe todos os pacotes distribuídos na rede, logo, não há necessidade de duplicar a informação para envio a um participante, tornando essa comunicação mais econômica. Desta forma, a conexão *multicast* é mais adequada para transmissão

de dados multimídia. E o endereçamento *unicast*, identificado pelas classes A, B e C, é o endereço que identifica a rede e os seus computadores individualmente. Esses endereços representam a grande massa de redes e computadores na *internet*. Sendo assim, são os endereços mais usados no modelo. Esses endereços não se repetem dentro da grande rede. Para redes com poucos equipamentos usa-se o endereçamento classe C, para muitos equipamentos usa-se a classe A e no caso intermediário usa-se a classe B. A classe E, é a faixa reservada para uso futuro, não sendo usado para identificar as redes e computadores.

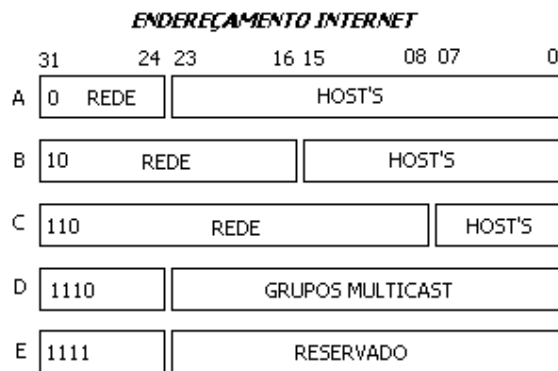


Figura I.16 – Endereçamento IP

Tabela I.4 – Espaço de endereçamento IP

CLASSE	REDES VÁLIDAS	HOSTS VÁLIDOS	NÚMERO DE REDES	NÚMERO DE HOSTS
A	1.0.0.0 A 126.0.0.0	1.0.0.1 A 126.255.255.254	$2^7 - 2$	$2^{24} - 2$
B	128.0.0.0 A 191.255.0.0	128.0.0.1 A 191.255.255.254	2^{14}	$2^{16} - 2$
C	192.0.0.0 A 223.255.255.0	192.0.0.1 A 223.255.255.254	2^{21}	$2^8 - 2$
CLASSE	ENDEREÇOS		GRUPOS DE MULTICASTING	
D	224.0.0.0 A 239.255.255.255		2^{28}	
CLASSE	ENDEREÇOS			
E	240.0.0.0 A 240.255.255.255			

Protocolos, Arquiteturas e Aplicações

Uma rede é formada por camadas conforme apresentado no modelo TCP/IP. As camadas se comunicam através de protocolos, onde cada protocolo tem a função de auxiliar o protocolo da camada imediatamente superior, ou seja, a camada de enlace fornece serviços para camada de rede, esta fornece serviço para camada de transporte que fornece serviço para camada da aplicação. A camada da aplicação fornece serviço para o software do usuário. A Figura I.17 mostra as camadas e seus respectivos protocolos.

Alguns protocolos são tão utilizados que já são considerados verdadeiras aplicações. Alguns surgiram de aplicações de sucesso. Alguns requisitos são tão complexos que somente um protocolo não seria capaz de suportar toda solução, desta forma, surgem às arquiteturas, que especificam um conjunto de protocolos para operar entre si formando uma só solução.

Aplicações multimídia são tão complexas que exigem inclusive que o modelo TCP/IP seja alterado para permitir um modelo de qualidade eficaz para garantir o funcionamento da transmissão.

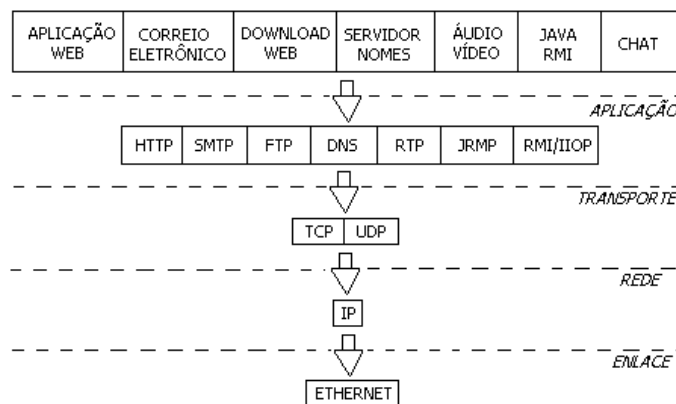


Figura I.17 – Protocolos do modelo TCP/IP

A camada de aplicação é a camada limítrofe entre o humano e a máquina, essa camada suporta os protocolos apresentados na Figura I.17 e os criados pelo próprio programador do sistema. A camada do enlace é a camada limítrofe entre o sistema e o meio físico.

A Figura I.18 exemplifica uma aplicação multimídia usando o protocolo RTP localizado na camada de aplicação que usa o protocolo UDP na camada de

transporte que usa o protocolo IP na camada de rede e o protocolo ETHERNET na camada de enlace. Esses protocolos cooperam entre si de modo fornecer a comunicação entre computadores em rede. Assim como uma aplicação multimídia usa o protocolo UDP da camada de transporte, uma aplicação de correio eletrônico usa o protocolo TCP.

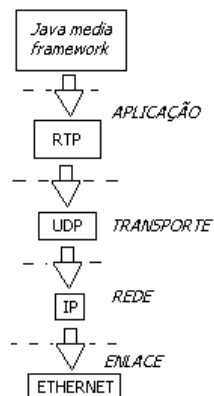


Figura I.18 – Exemplo de protocolo do modelo TCP/IP

A seguir, serão revisados os protocolos que atenderão a especificação proposta deste trabalho:

HTTP

Segundo Berners-Lee *et al* (1999), o *Hypertext Transfer Protocol* (HTTP) é um protocolo no nível da camada de aplicação usado para distribuição e colaboração de informações em sistemas *hipermídia*. É definido na RFC2616.

O protocolo HTTP é fundamental para a comunicação através da *Web*. Esse protocolo simples e sem estado tem por base um paradigma de pedido/resposta, ou seja, um pedido de um cliente HTTP gera uma resposta de servidor que é transmitida de volta para o cliente HTTP. O par de mensagens pedido/resposta é chamado de transação HTTP. As mensagens são transportadas através do TCP/IP, que é o mecanismo de transporte subjacente para a *Internet* e para a maioria das *Intranets*.

Normalmente, um pedido HTTP é enviado digitando-se um URL de pedido na janela do navegador ou clicando-se em um *hyperlink* na página da *Web* que está sendo exibida pelo navegador. Um pedido também poderia ser gerado

através de um programa. O pedido especifica determinado método, que deve ser executado pelo servidor da *Web*, como GET, POST, HEAD, PUT ou DELETE. Esses métodos são definidos na especificação HTTP. Os dois métodos mais freqüentemente usados são GET e POST. Normalmente, o método GET é usado para recuperar uma página do servidor da *Web*. Ele pode incluir parâmetros de pedido em uma *string* de consulta como *parameterName=parameterValue*, anexados no URL de pedido. Por exemplo, digitar o URL a seguir gera um pedido HTTP GET com um parâmetro de pedido chamado *user*, que tem o valor *apaula*:

`http://www.cefet-rj.br/consultaAluno.jsp?user=apaula`

Em contraste o método POST envia parâmetros no corpo do pedido HTTP e eles não aparecem no URL do pedido. O método POST é mais conveniente do que o método GET para carregar grandes volumes de dados, como o conteúdo de um arquivo, ou para enviar informações confidenciais, como a autenticação de um usuário. Esse método é usado quando o pedido modifica dados no servidor.

Uma resposta HTTP é gerada pelo servidor da *Web* como resultado do processamento de um pedido HTTP. Ela consiste em cabeçalhos HTTP opcionais e um corpo. Os cabeçalhos são usados pelo navegador da *Web* para apresentar o conteúdo do corpo. Por exemplo, se o cabeçalho *Content-type* for especificado como *text/html*, que é o tipo MIME mais usado para páginas da *Web*, então o navegador da *Web* interpretará as *tags* HTML no corpo da mensagem, para formatar o conteúdo da página.

O protocolo HTTP trabalha lado a lado com a linguagem de marcação HTML. O HTML usa *tags* de marcação especiais para o conteúdo a ser apresentado em uma página da *Web*. Essas *tags* são interpretadas por um navegador da *Web* de acordo com a semântica definida na especificação HTML. Existem *tags* para diferentes propósitos, por exemplo, para indicar como o texto deve ser formatado, para manipular entrada de dados do usuário e para navegar de uma página para outra. Geralmente, as *tags* são usadas em pares, cada *tag* “de início” tem uma *tag* “de fim” correspondente e o texto incluído entre as *tags* está sujeito ao efeito da *tag*. Por exemplo, o texto que aparece entre as *tags* `` e `` é apresentado com fonte em negrito. Normalmente, as *tags* são aninhadas em uma estrutura hierárquica.

SMTP

O *Simple Mail Transfer Protocol* (SMTP) é um protocolo da camada de aplicação e possui especificação definida na RFC0821. Segundo Postel (1982), tem o objetivo de transferir de forma confiável e eficiente o correio eletrônico, popularmente conhecido como *email*. O projeto do SMTP estabelece um modelo de comunicação que permite a criação de um canal de comunicação entre os participantes, onde o *sender-SMTP* (remetente) estabelece um canal de transmissão com o *receiver-SMTP* (receptor). A transmissão possui uma série de comandos de negociação, característicos do protocolo, para envio da mensagem.

DNS

O *Domain Name System* (DNS) é um protocolo da camada de aplicação e possui especificação definida na RFC1034. Segundo Mockapetris (1987), o DNS é um estilo de nome de domínio que representa os *host* que estão na *Internet*.

Os sites que estão na internet são hospedados por esses *hosts* e, para acessar, digita-se um endereço no campo correspondente do seu navegador, por exemplo, **www.cefet-rj.br**. Na verdade esse endereço sem o sistema de resolução de nomes seria 200.9.149.38. Devido à quantidade de *sites* existentes na *Internet* é impossível lembrar todos esses números sem nenhum significado. É neste ponto que a tecnologia DNS trabalha, ou seja, Trata-se de um recurso usado em redes TCP/IP que permite acessar computadores sem que o usuário ou sem que o próprio computador tenha conhecimento de seu endereço IP.

RTP

O *Real Time Protocol* (RTP) é um caso de protocolo que surgiu de uma aplicação de sucesso. Ele é considerado uma unanimidade entre aplicações e arquiteturas para transmissão de dados multimídia em tempo real na *Internet*.

O RTP possui todas as características para ser um protocolo da camada de transporte, porém ele é considerado por alguns autores como camada de aplicação, e por outros uma espécie de intermediador entre os dados a serem

transmitidos e os meios que efetivam a transmissão. Possui especificação definida na RFC1889.

Segundo Casner *et al.* (1996), o RTP fornece as funções apropriadas de transporte de rede fim a fim para aplicações que transmitem dados em tempo real, tais como áudio e vídeo, sobre redes que fornecem serviços *multicast* e *unicast*. O RTP não faz reserva de recurso e não garante qualidade de serviço (QoS) para transmissão em tempo real. O transporte dos dados é acrescido por um outro protocolo de controle, denominado *Real Time Control Protocol* (RTCP) para permitir a monitoração da entrega dos dados em uma maneira escalável às grandes redes *multicast* e fornecer a funcionalidade mínima do controle e da identificação. RTP e RTCP são projetados para serem independentes das camadas de transporte e de rede.

Dentre os cenários onde se aplica o protocolo RTP destaca-se o uso em aplicações de videoconferência. Se ambas as mídias, áudio e vídeo, forem usadas na transmissão, elas serão transmitidas em sessões RTP diferentes, inclusive portas e endereços *multicast*. Apesar de haver uma distinção entre os sinais de áudio e vídeo, isso é importante no caso de algum participante receber somente um tipo de mídia (áudio), o protocolo RTCP garantirá o sincronismo entre os sinais como se fosse uma sessão única.

O RTP é suportado por linguagens de alto nível como c++ e Java. As classes do pacote *javax.media.rtp*, *javax.media.rtp.event*, e *javax.media.rtp.rtcp* fornecem a sustentação para o RTP, permitindo a transmissão e a recepção multimídia através da rede através do *framework Java Media Framework*.

O RTP necessita dos recursos dos protocolos das camadas inferiores, como o mesmo é usado freqüentemente sobre o UDP, a transmissão fica órfã de mecanismos de garantia de entrega de dados em ordem ou com atrasos constantes, além de não fornecer garantia de qualidade nas aplicações de tempo real. O protocolo RTCP foi especificado para ser um protocolo de controle para auxiliar o RTP na transmissão de dados em tempo real disponibilizando informações de QoS, facilitando o uso de técnicas adaptativas para resolver problemas na rede.

JRMP

O *Java Remote Method Protocol* (JRMP) segundo Lee, Ryan e Seligman (1999), é usado pelo *Remote Method Invocation* (RMI) do Java, esse protocolo faz a comunicação entre objetos remotos específico da tecnologia Java. É responsável pela conexão entre as máquinas virtuais do cliente e do servidor, e pelo transporte das mensagens entre as mesmas. O JRMP roda sobre o TCP/IP para a comunicação entre as máquinas virtuais. JRMP é um protocolo proprietário da *Sun Microsystems* baseado em fluxos de *bytes* (*streams*). O protocolo HTTP também pode ser utilizado para encapsular as mensagens do protocolo JRMP quando existir *firewall* entre o cliente e o servidor.

TCP

O *Transmission Control Protocol* (TCP) é um protocolo da camada de transporte. Tem a sua especificação na RFC0793 (1981) e a sua utilização é quando necessita de um ambiente que precise de garantias quanto a integridade dos pacotes enviados pela rede.

O TCP é um protocolo orientado a conexão, sendo destinado a aplicações multi-rede. O TCP é um meio garantido de transmissão porque os dados danificados ou perdidos enviados na rede possuem uma confirmação pelo receptor da sua chegada. Isto é conseguido atribuindo um número de seqüência a cada octeto transmitido, e requerendo um reconhecimento positivo (ACK) do TCP de recepção. Se o ACK não for recebido dentro de um intervalo do intervalo de tempo, os dados são retransmitidos. No receptor, os números de seqüência são usados requisitar corretamente os segmentos recebidos, permitindo a eliminação de pacotes duplicados. Esse protocolo é indicado para aplicações que não permitem perdas de pacote, tais como uma aplicação de consulta de prontuário remoto de paciente, onde os dados apresentados não podem conter erros.

UDP

O *User Datagram Protocol* (UDP) é o outro protocolo da camada de transporte. Tem a sua especificação definida na RFC 0768, e segundo Postel (1980), o UDP fornece um procedimento para aplicação emitir mensagens a

outras aplicações usando um mecanismo simples comparado ao TCP. O protocolo é sem conexão, ou seja, o receptor não precisa estar conectado ao emissor. Este protocolo não garante a entrega, ficando a cargo da aplicação a correção de possíveis problemas de perda de pacote.

Devido à simplicidade dos mecanismos utilizados na transmissão, esse protocolo é mais rápido que o TCP, sendo indicado para aplicações onde pequenas perdas de pacote não afetarão a qualidade dos dados no receptor. É aplicado em transmissão de áudio e vídeo em conjunto com o protocolo RTP.

Meios de comunicação na camada de aplicação

A exigência de objetos em ambientes distribuídos e heterogêneos, com características de portabilidade, reusabilidade, interoperabilidade vem crescendo junto com o surgimento de novas tecnologias, como as redes de comunicação de alta velocidade. Os padrões para interoperabilidade entre objetos distribuídos permitem objetos em diferentes linguagens se comunicarem via protocolos que oferecem níveis de transparência na utilização de objetos heterogêneos distribuídos.

A tecnologia de comunicação se baseia em *socket*, sendo considerada no conjunto de soluções disponível para comunicação a forma mais elementar de desenvolvimento, onde o programador tem a função de desenvolver toda a comunicação inter-rede, o Java apresenta uma facilidade de programação que permite construir soluções desta natureza escrevendo poucas linhas de código.

Um computador possui uma única conexão física à rede, e os dados destinados a esse computador chegam através desta conexão. Um computador também possui várias aplicações funcionando ao mesmo. Entretanto, os dados podem ser pretendidos para aplicações diferentes que funcionam no computador. Assim um computador sabe a que aplicação enviar os dados pelo o seu *port*. Um port não é um dispositivo físico, mas uma abstração para facilitar a comunicação entre um servidor e um cliente.

A Figura I.19 ilustra os dados sendo transmitidos sobre a *internet* acompanhado pelo endereço IP de 32 *bits* que identifica o computador e o número do *port* de 16 bits que identifica a aplicação, usados pelo tanto pelo TCP quanto UDP para entregar os dados para quem de direito.

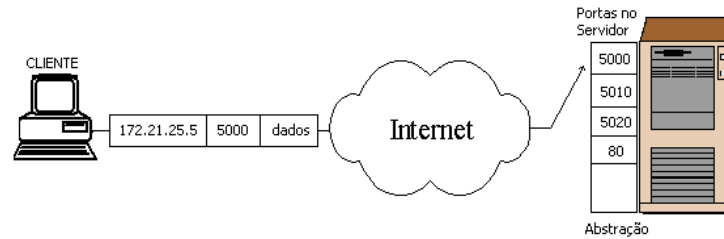


Figura I.19 – Endereço IP e Port

O UDP é um protocolo que envia pacotes de dados, denominados *datagramas*, sem a garantia de entrega e sem conexão com o destinatário, o datagrama chega de surpresa sem uma conexão pré-existente. Os passos são mostrados a seguir:

➤ Criando um *InetAddress*

```
InetAddress IPAddress = InetAddress.getByName("localhost");
```

➤ Criando um *DatagramSocket*

```
DatagramSocket socket=new DatagramSocket ();
```

➤ Criando um *DatagramPacket*

```
String mensagem = "Olá mundo!";
DatagramPacket packet=new DatagramPacket(mensagem.getBytes(),
                                          mensagem.getBytes().length, IPAddress, 5000);
```

➤ Enviando um *DatagramPacket*

```
socket.send(packet);
```

➤ Recebendo um *DatagramPacket*

```
byte dados[] = new byte[257];
DatagramPacket packet=new DatagramPacket(dados,dados.length);
try {
    socket.receive(packet);
    System.out.println(packet.getData());
} catch (IOException e) {
    System.out.println("Erro ao receber dados");
}
```

➤ Fechando a conexão

```
socket.close();
```

A decisão da tecnologia de comunicação foi influenciada por Calabrez (2004) que avaliou em sua tese de mestrado as tecnologias de comunicação RMI,

RMI/IIOP, Corba e JAX/RPC. Calabrez (2004, p.44) observou que RMI e CORBA possuem características que as diferenciam das demais. JAX-RPC (SOAP) não apresenta nenhuma característica não encontrada nas outras. RMI-IIOP tem a vantagem de possibilitar a interoperabilidade entre aplicações RMI e CORBA.

O mercado atualmente está sendo impulsionado pela fabricante *Microsoft* para o uso de *Web Services*, contudo essa tecnologia por ser baseada no protocolo http não é adequada para situações envolvendo chamadas de procedimentos remotos, a mensagem trocada entre clientes e servidores na forma de documentos XML torna necessário o uso de analisadores sintáticos e semânticos em tempo de execução e não provê mecanismos para a passagem de objetos por referência.

Calabrez (2004, pg. 87) concluiu que a tecnologia RMI apresentou o melhor desempenho e as demais tecnologias são mais flexíveis e oferece maior portabilidade.

Segundo Jguru (2000), o *Remote Method Invocation* (RMI) permite um programa rodando em um computador cliente fazer chamadas de métodos remotos localizados em um servidor remoto. Ele capacita o programador a realizar computação distribuída através da rede. A arquitetura RMI fornece três camadas: camada de transporte, camada de referência remota e camada *stubs/skeleton*. A Figura I.20 mostra a arquitetura de comunicação do RMI.

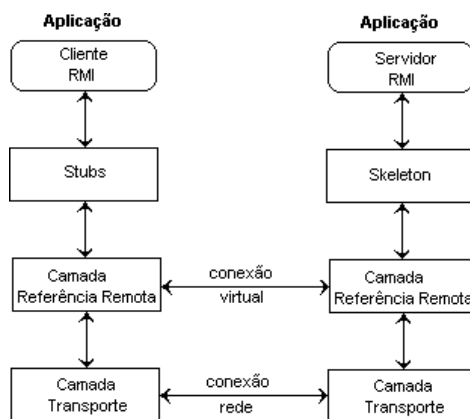


Figura I.20 – Remote Method Invocation

O RMI suporta dois protocolos de comunicação, o *Java Remote Method Protocol* (JRMP) e o *Internet Inter-ORB Protocol* (IIOP). Ambos permitem objetos se comunicarem, a diferença, está no fato do JRMP ser uma especialização da linguagem *Java* permitindo a comunicação somente dentro desta tecnologia.

O *Stubs/Skeletons* é um mecanismo padrão para comunicação entre objetos remotos. O *Stub* é um objeto remoto localizado no cliente que encapsula os mecanismos da comunicação do cliente. O *Skeleton* está localizado no servidor e têm a função de encapsular os mecanismos da comunicação no servidor.

A Camada de referência remota interpreta e gerencia as referências feitas pelos clientes para os objetos remotos.

A camada de transporte é baseada em conexões TCP/IP, mas pode ser substituído pelo protocolo UDP/IP. É responsável por transportar a semântica da invocação ao servidor e escondem da aplicação as dependências e complexidade em enviar uma chamada remota ao servidor. Essa camada ainda inclui uma conexão que é uma entidade abstrata que direciona ao servidor todas conexões de um cliente. Desta forma, é possível afirmar o seguinte quanto ao RMI:

- É um mecanismo que permite a um objeto invocar um método que está em outro espaço de endereçamento, na mesma máquina ou em uma máquina diferente.
- O RMI é um mecanismo de RPC (*Remote Procedure Call*) para programação orientada a objetos em Java.
- O RMI compatível com CORBA, através da implementação RMI/IIOP que usa o protocolo IIOP (*Internet Inter-ORB Protocol*) como *background* na comunicação do RMI.

Quanto às classes envolvidas, é possível afirmar o seguinte:

- Instâncias das classes remotas são acessadas localmente (mesmo espaço de endereçamento) ou remotamente (outro espaço de endereçamento) através da utilização da referência remota do objeto que está em outro espaço de endereçamento.
- Os objetos serializáveis são passados como parâmetros ou como retorno de uma invocação de método.
- O estado do objeto é copiado de um espaço de endereçamento para outro.

Clientes encontram objetos remotos usando um serviço de diretório. Esse serviço tem a função de armazenar previamente os objetos remotos e disponibiliza-los aos clientes.

O RMI usa diferentes serviços de diretório, incluindo o *Java Naming and Directory Interface* (JNDI). O próprio RMI possui um serviço de diretório próprio denominado *RMI Registry*. O *RMI Registry* roda em cada máquina que pretende disponibilizar serviços remotos através dos objetos registrados no *port* 1099.

O computador servidor deverá ter um programa que cria o serviço remoto. Primeiro cria-se o objeto que implementará o serviço remoto, depois ele é exportado para o RMI que cria uma lista de serviço que fica esperando os clientes se conectarem e requisitar serviços. O objeto é registrado no *RMI Registry* com um nome de domínio público.

No lado cliente, o *RMI Registry* é acessado através da classe estática *Naming*. Ela fornece o método *lookup()* para procurar um registro de um objeto remoto. Esse método aceita como parâmetro um URL especificando o computador servidor do serviço. Como retorno, o método devolve a referência do objeto remoto.

Para trabalhar com RMI é necessário criar uma interface remota, implementar o objeto, criar um servidor, gerar o *Stub* e *Skeleton*, criar o objeto remoto e por último criar os clientes RMI que usarão a aplicação remotamente. A seguir são mostrados os passos para trabalhar com RMI:

- Interface Remota: Define quais os métodos do objeto que poderão ser chamados remotamente pelo Cliente.

```
import java.rmi.Remote;
import java.rmi.RemoteException;
public interface HelloRemote extends java.rmi.Remote {
    public String say() throws RemoteException;
}
```

- Implementação do Objeto: Contém a implementação dos métodos definidos na interface remota.

```
import java.rmi.*;
import java.rmi.server.*;
public class Hello extends UnicastRemoteObject implements HelloRemote {
    private String message;
    public Hello (String msg) throws RemoteException {
        message = msg;
    }
    public String say() throws RemoteException {
        return message;
    }
}
```

- Servidor: Responsável pela criação do objeto e pela sua colocação no registro de maneira a que possa ser localizado pelos clientes.

```
import java.rmi.*;
public class Server {
    public static void main (String[] argv) {
        try {
            Naming.rebind ("Hello", new Hello ("Hello, world!"));
            System.out.println ("Hello Servidor está funcionando");
        } catch (Exception e) {
            System.out.println ("Hello Server failed: " + e);
            e.printStackTrace();
        }
    }
}
```

- Geração dos Stubs e Skeletons: Geração das classes Hello_Stub e Hello_Skel.

Esta tarefa é realizada pelo aplicativo *rmic* localizado no diretório *bin* da instalação do Java. *rmic Hello*. Essas classes vão conter todas as complexidades do protocolo de comunicação de rede. Desta forma, o programador se concentrará em desenvolver as regras de negócio da aplicação, e não infra-estrutura.

- Criação do objeto remoto
 - Iniciar o processo que irá gerenciar a busca de objetos distribuídos (rmiregistry)
 - c:/diretorio da aplicacao> rmiregistry
 - Executar o programa servidor
 - c:/diretorio da aplicacao> Java Server

- Clientes RMI: Acesso aos métodos do objeto remoto

```
import java.rmi.*;
import HelloRemote;
public class Client {
    public static void main (String[] argv) {
        try {
            HelloRemote hello = (HelloRemote) Naming.lookup
                ("//localhost/Hello");
            System.out.println (hello.say());
        } catch (Exception e) {
            System.out.println ("HelloClient exception: " + e);
        }
    }
}
```


Qualidade de Serviço

Segundo Elkind (2006), cada tipo de dado em uma transmissão multimídia requer seu próprio requisito de qualidade de serviço, exigindo o desenvolvimento de aplicações cada vez mais complexas para funcionar satisfatoriamente sobre as atuais redes de dados que não garantem QoS (qualidade de serviço).

Elkind (2006) propõe um sistema confiável de distribuição de mídia que mantém QoS dos fluxos de vídeo, atendendo a uma política preemptiva de prioridades definida para os clientes, monitorando e reagindo às ações dos mecanismos de tolerância as falhas existentes na infra-estrutura. Esses dois objetivos são inter-relacionados e complementares.

A garantia da qualidade é realizada com a ajuda do codec H.263 implementado no *framework* JMF (*Java Media Framework*), onde através de emulações, foram definidos valores limites para *jitter* e perdas, então foi programada uma rede *neuro-fuzzy*, que acoplada a um código denominado “Inspetor de Desempenho”, permite a avaliação on-line da qualidade do vídeo no cliente, dadas as condições presentes da rede.

Os valores limites de qualidade fornecido ao “inspetor de qualidade” foram definidos através de testes objetivos e subjetivos. O subjetivo leva em consideração as reações cognitivas ocorridas para diversas situações de degradação na qualidade do vídeo. Os testes subjetivos foram orientados pelo método *Mean Opinion Score (MOS)*, recomendado pelo ITU (ITU-R BT.500-10, 2000). Neste método, cada uma das pessoas em um grupo de avaliadores, assiste a várias apresentações de um mesmo vídeo. Cada apresentação consiste de uma situação nos parâmetros que vão definir a qualidade do vídeo. Ao final, computa-se a média das opiniões para cada situação que gerou a degradação sendo avaliada.

Em Carvalho e Deusdado (2004) é apresentado um exemplo prático de uma aplicação para e Educação à Distância que realiza QoS por técnica adaptativa. Ao iniciar a transmissão para participantes de uma sessão multicast, será executado um algoritmo para determinar as condições ideais para realizar a sessão. O resultado do algoritmo é comparado com a Tabela I.5 onde será adaptado o modo de transmissão ideal para iniciar a transmissão. A partir desse ponto, o sistema atuará para manter o QoS em níveis aceitáveis. Segue a equação para o cálculo da adaptação:

$$M = (\text{int}) (B / (RTT/2) + ML/P) * K$$

Onde:

M = Modo de ajuste selecionado;

B = Largura de banda em Kbps;

RTT = *Round trip time* em ms;

ML = Memória livre em (*MegaByte*);

P = Taxa de ocupação do processador em %;

K = 1/50 – constante para reduzir escala nos modos 1 até 5.

Tabela I.5 – Adaptação da QoS das aplicações segundo Carvalho e Deusdado (2004)

MODO	LARGURA DE BANDA MÁXIMA	TAXA DE FRAMES	CODEC VÍDEO	CORES	CODEC ÁUDIO
5	1 Mbps	30 fps	H.261	SIM	L16
4	512 Kbps	25 fps	H.261	SIM	PCM
3	256 Kbps	20 fps	H.261	SIM	DVI
2	128 Kbps	15 fps	H.263	SIM	GSM
1	64 Kbps	10 fps	H.263	NÃO	LPC

Percepção Multimídia

A percepção do objeto pela visão humana é resumida em duas etapas: A primeira ocorre pela estimulação luminosa do objeto provocando impulsos nervosos nas fibras do nervo óptico. Como resultado, tem-se o uma descrição da cena observada em uma linguagem "elétrica" decodificada pelo cérebro. A segunda etapa, consiste no entendimento de fato da imagem representada pelo objeto. Logo a primeira etapa é representada por um sensor e a segunda é representada cognitivamente, resultado da representação do conhecimento adquirida ao longo dos anos (FERRAZ, 1998, p.67).

A representação do conhecimento é definida como uma imaginação de um objeto que foi extraído do mundo externo, processado nos campos da sensação e percepção, onde ainda o objeto estava presente e devidamente processado no campo da representação caracterizando a imaginação, e nesse caso não temos a presença do objeto e segundo Castanheira (informação verbal)¹ chamamos a representação do conhecimento de “o retorno do percebido”.

¹ Aula ministrada por Maurício Castanheira na disciplina Sistema de Interação Homem-Máquina do mestrado de tecnologia do CEFET-RJ em Março 2005.

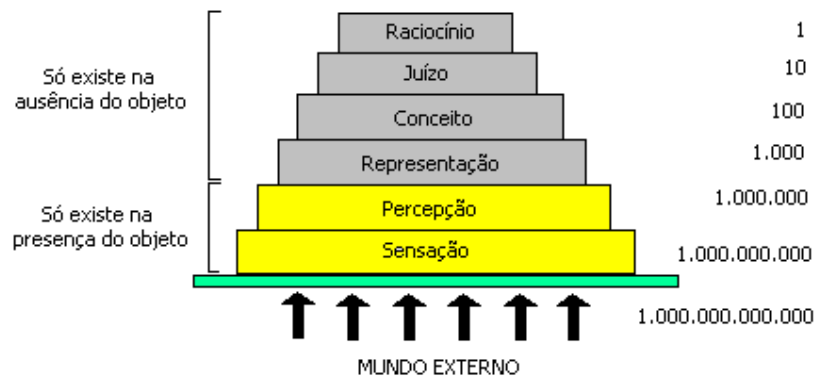


Figura I.21 – Representação do conhecimento em Castanheira (informação verbal)¹.

A Figura I.21 indica que a manifestação inteligente pressupõe aquisição, armazenamento e um raciocínio em cima do conhecimento, para que o conhecimento possa ser armazenado é essencial que se possa representá-lo. Estando em contato com o objeto/informação construímos mentalmente uma representação do conhecimento que sem sua existência não poderíamos manipular as nossas aprendizagens e usá-las posteriormente no decorrer das resoluções diárias de problemas. Quando tratamos de qualquer tipo de inteligência, sempre partimos da inteligência humana, e a representação do conhecimento é modelada através de modernas técnicas e algoritmos para que a computação possa também representar a informação na sua ausência com a mesma eficiência que o cérebro humano.

Software Livre

As licenças caracterizam um produto como livre, parcialmente livre ou proprietário. Em Hexsel (2002) foi feito um estudo completo do uso dessas licenças. A Tabela I.6 a apresenta uma definição resumida das licenças.

Os desenvolvedores de software devem estar atentos quanto ao tipo de licença que eles estarão usando para construir o seu produto, visto que esses conjuntos de licenças podem confundir se tornando uma armadilha. Uma vez entendido o tipo de licença, e usando ferramentas básicas sem custo nenhum para desenvolver o produto sem ficar atrelado a outros produtos proprietários, esse produto também será um produto livre.

A comunidade de software livre defende a construção de produtos que possuam uma redistribuição livre onde trabalhos derivados desses seja permitido sem restrições, não permita distribuições casadas de produtos, e não restrinja o funcionamento de outro software em processamento paralelo.

Existe a necessidade de construir um software para transmissão multimídia de biomicroscopia, pois não existe no mercado uma aplicação que atenda todos os requisitos funcionais e não-funcionais exigidos para esse tipo de aplicação.

Tabela I.6 – Lista de licenças em Hexsel (2002)

GPL	A Licença Pública Geral GNU é a licença que acompanha os pacotes distribuídos pelo Projeto GNU, e mais uma grande variedade de software, incluindo o núcleo do sistema operacional Linux. A formulação da GPL é tal que ao invés de limitar a distribuição do software por ela protegido, ela de fato impede que este software seja integrado em software proprietário. A GPL é baseada na legislação internacional de copyright, o que deve garantir cobertura legal para o software licenciado com a GPL5.
Debian	A licença Debian é parte do contrato social celebrado entre a Debian e a comunidade de usuários de software livre, e é chamada de Debian Free Software Guidelines (DFSG). Em essência, esta licença contém critérios para a distribuição que incluem, além da exigência da publicação do código fonte. Estes critérios são: (a) a redistribuição deve ser livre; (b) o código fonte deve ser incluído e deve poder ser redistribuído; (c) trabalhos derivados devem poder ser redistribuídos sob a mesma licença do original; (d) pode haver restrições quanto à redistribuição do código fonte, se o original foi modificado; (e) a licença não pode discriminar contra qualquer pessoa ou grupo de pessoas, nem quanto a formas de utilização do software; (f) os direitos outorgados não podem depender da distribuição onde o software se encontra; e (g) a licença não pode 'contaminar' outro software.
Open Source	A licença do Open Source Initiative é derivada da Licença Debian, com as menções a Debian removidas.
BSD	A licença BSD cobre as distribuições de software da Berkeley Software Distribution, além de outros programas. Esta é uma licença considerada 'permissiva' porque impõe poucas restrições sobre a forma de uso, alterações e redistribuição do software licenciado. O software pode ser vendido e não há obrigações quanto à inclusão do código fonte, podendo o mesmo ser incluído em software proprietário. Esta licença garante o crédito aos autores do software, mas não tenta garantir que trabalhos derivados permaneçam como software livre.
X.org	O Consórcio X distribui o X Windows System sob uma licença que o faz software livre, mas não adere ao copyleft. Existem distribuições sob a licença da X.org que são software livre, e outras distribuições não o são. Existem algumas versões não-livres do sistema de janelas X11 para estações de trabalho e certos dispositivos do IBM-PC que são as únicas funcionais disponíveis, sem similares distribuídos como software livre.
Software de domínio público	Software em domínio público é software sem copyright. Alguns tipos de cópia, ou versões modificadas, podem não ser livre porque o autor permite que restrições adicionais sejam impostas na redistribuição do original ou de trabalhos derivados.
Software semi-livre	Software semi-livre é software que não é livre, mas é concedida a permissão para que indivíduos o usem, copiem, distribuam e modifiquem, incluindo a distribuição de

	versões modificadas, desde que o façam sem o propósito de auferir lucros. Exemplos de software semi-livre são as primeiras versões do Internet Explorer da Microsoft, algumas versões dos browsers da Netscape, e o StarOffice.
Freeware	O termo freeware não possui uma definição amplamente aceita, mas é usado com programas que permitem a redistribuição, mas não a modificação, e seu código fonte não é disponibilizado. Estes programas não são software livre.
Shareware	Shareware é o software disponibilizado com a permissão para que seja redistribuído, mas a sua utilização implica no pagamento pela sua licença. Geralmente, o código fonte não é disponibilizado e, portanto modificações são impossíveis.
Software proprietário	Software proprietário é aquele cuja cópia, redistribuição ou modificação são em alguma medida proibidos pelo seu proprietário. Para usar, copiar ou redistribuir, deve-se solicitar permissão ao proprietário, ou pagar para poder fazê-lo.

Toda documentação deve ser disponível para uso imediato, sem restrição, deve acompanhar o produto e ser fiel ao mesmo.

O mundo do *software* livre não está limitado em atender o mundo de usuários denominados básicos, ou seja, aqueles que usam computador como ferramentas de escritório e acesso a *Internet*, onde são usados basicamente aplicativos prontos como editores de textos e navegadores, mas também os desenvolvedores e estudantes de sistema que usam ferramentas de programação para construir aplicativos.

Uma série de componentes são envolvidos na construção de um produto *enterprise*, a Tabela I.7 apresenta uma lista com algumas dicas de produtos livres.

Tabela I.7 – Lista de produtos livres e/ou código aberto

COMPONENTE	PRODUTO	SITE
Servidor de aplicação	JBOSS	http://www.jboss.org
	Jonas	http://www.objectweb.org/jonas
	OpenEJB	http://openejb.sourceforge.net
	Apache Tomcat	http://jakarta.apache.org/tomcat/
	JORAM	http://www.objectweb.org/joram
Banco de dados	MySQL	http://www.mysql.com
	Postgres	http://www.postgresql.org
	Hypersonic SQL	http://hsqldb.sourceforge.net
Controladores de versão	CVS	http://www.cvs.com
Ferramentas de desenvolvimento	Eclipse	http://www.eclipse.org
	NetBeans	http://www.netbeans.org
	Argo UML	http://argouml.tigris.org
	Eclipse UML	http://www.eclipseuml.com
Geradores de relatórios	JfreeReport	http://www.object-refinery.com/jfreereport/
	JasperReport	http://jasperreports.sourceforge.net.net/
	Jolt	http://www.planetaprojeto.com.br/jolt
Frameworks	Hibernate	http://www.hibernate.org
	JBanana	http://www.jbanana.org
	Struts	http://jakarta.apache.org/struts
	WebWork	http://www.sourceforge.net/projects/webwork
	Velocity	http://jakarta.apache.org/velocity

II- FUNDAMENTOS TEÓRICOS

Arquivos Multimídia

Tipos Básicos

Os tipos básicos são as formas mais tradicionais na transmissão de dados, sendo as redes projetadas especialmente para esses tipos de informações, a questão aqui é estabelecer em ambos os lados da transmissão a mesmas codificações, tais como inteiros, números de ponto flutuante, *string* de caracteres, arranjos e estruturas.

Os bits para serem enviados pela rede precisam sofrer uma conversão, um empacotamento e uma linearização. (DAVIE E PETERSON, 2004, p. 392). A Figura II.1 exemplifica esta ação. Este processo é denominado *marshalling* de argumentos.

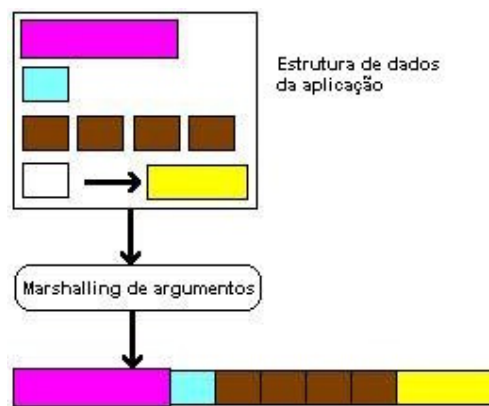


Figura II.1 – Marshalling de argumentos em Davie e Peterson (2004, p. 392)

Segundo Davie e Peterson (2004, p. 391), Os tipos suportados por um mecanismo de *marshaling* de argumentos são classificados em três níveis. Cada nível torna mais difícil a tarefa a ser realizada pelo sistema de *marshalling*.

No primeiro caso, o sistema opera um certo conjunto de tipos básicos. Os tipos básicos incluem inteiros, números de ponto flutuante e caracteres. O sistema deve suportar também os tipos ordinais e booleanos. Esse processo deve ser capaz de perceber que alguns computadores representam números de ponto flutuante no formato IEEE padrão 754, enquanto outras máquinas ainda usam seus formatos não padronizados. Mesmo algo simples como os inteiros,

arquiteturas diferentes utilizam tamanhos diferentes (por exemplo, 16 bits, 32 bits e 64 bits). Os inteiros também são representados na forma big-endian (o bit mais significativo de uma palavra está no byte com o endereço mais alto), enquanto em outras máquinas os inteiros são representados na forma little-endian (o bit mais significativo está no byte com o endereço mais baixo). Além de os aplicativos poderem ser escritos em linguagens de programação diferentes.

No segundo caso, Davie e Peterson (2004, p.392) trata das estruturas e arrays. O problema é que os compiladores utilizados inserem informações denominadas tag's entre os campos que formam a estrutura e o sistema de *marshalling* em geral empacota estruturas de forma a não conter informações extras.

E finalmente, no terceiro caso são tratados os tipos que são construídos com ponteiros. Ou seja, os dados transmitidos podem conter ponteiros apontando para endereços de memória que contenham dados. E, quando a informação chega ao destino, ela não ocupará o mesmo endereço de memória que estava na máquina origem.

Os dados devem ser transmitidos com uma linguagem de alto nível capaz de lidar com todos os problemas apresentados, tirando do programador a complexidade imposta pelo meio, liberando-o para preocupar-se somente com a regra de negócio imposta pela aplicação, e não com tarefas que são específicas de infra-estrutura de sistemas distribuídos.

Segundo Ernest, Heller e Roberts (2000, p.8) Java define oito tipos de dados que são básicos, também conhecidos como tipo primitivo. Eles podem ser utilizados como valores literais ou variáveis. Os tipos podem ser considerados em quatro categorias: lógica, textual, integral e ponto flutuante. A Tabela II.1 mostra a quantidade de bits suportada por cada tipo primitivo separados por categoria.

Tabela II.1- Tipos primitivos em Java em Ernest, Heller, Roberts. (2000, p.8)

Categoria	Tipo	Qtde de Bits	Faixa
Lógica	Boolean	1	-
Integral	byte	8	$-2^7 \dots 2^7 - 1$
	short	16	$-2^{15} \dots 2^{15} - 1$
	int	32	$-2^{31} \dots 2^{31} - 1$
	long	64	$-2^{63} \dots 2^{63} - 1$
Textual	char	16	$0 \dots 2^{16} - 1$
Ponto Flutuante	float	32	$-2^{31} \dots 2^{31} - 1$
	double	64	$-2^{63} \dots 2^{63} - 1$

Os valores lógicos possuem dois estados. Em geral são indicados como ativado e desativado, verdadeiro e falso ou sim e não. Na linguagem Java, esse valor é representado pelo tipo `boolean`.

Há quatro tipos integrais na linguagem Java. Cada tipo é declarado através de uma das palavras-chave `byte`, `short`, `int` ou `long`. Os literais de um tipo integral podem ser representados através e formas decimais, octais ou hexadecimais, o valor *default* é decimal.

Os caracteres simples são representados através do tipo `char`. O `char` representa um caractere unicode que utiliza um número de 16 bits sem sinal, em uma faixa de 0 a $2^{16} - 1$.

Uma variável de ponto flutuante pode ser declarada através da palavra-chave `float` ou `double`. Veja a seguir exemplos de números de ponto flutuante. Um literal numérico será um ponto flutuante se incluir um ponto decimal, uma parte exponencial (a letra E) ou for seguido da letra F ou D. O formato de um número flutuante é definido pela especificação Java como uma representação IEEE 754, usando os tamanhos mostrados acima. Esse formato independe de plataforma, fator útil para aplicações distribuídas, assim os objetos terão mesmo comportamento independente da plataforma de hardware.

Uma cadeia de caracteres é conhecida como *String*. Trata-se de uma classe geral de objetos para representar sequências de caracteres, seus valores não podem se mudados após criação, logo são constantes, devido a essa característica, podem ser compartilhados. Cadeias de caracteres são objetos, instâncias da classe *String* do pacote *java.lang*.

Para determinar se duas cadeias de caracteres possuem o mesmo conteúdo, usa-se o método `equals`. Este método considera maiúsculas e minúsculas como diferentes. Se não desejamos fazer esta distinção, devemos usar o método `equalsIgnoreCase`.

É importante não confundir estas comparações de conteúdo com o uso do operador `==`, que, quando usado entre duas variáveis que são referências a objetos, serve para determinar se as duas variáveis apontam para o mesmo objeto.

Áudio

O sinal de áudio segundo Tanenbaum (2003a, p.718) é uma onda acústica unidimensional, que apresenta uma série de variações de pressão. Quando essa onda entra nos ouvidos os tímpanos vibram enviando impulsos nervosos ao cérebro que são processados e entendidos como sons. Os seres humanos ouvem a uma frequência de 20 até 20.000 Hz (20kHz).

Essa é a definição para áudio analógico, um áudio digital é uma série de valores amostrados que para ser processado, deve ser capturado pelo sensor (microfone) e convertido em código binário através do conversor analógico-digital da placa de som instalada no computador.

Para ouvir o que foi gravado, o dado no formato binário, são transformados em sinais elétricos através do conversor digital analógico da placa de som instalada no computador.

O sinal de áudio apresenta cinco elementos básicos: amplitude, frequência, timbre, duração e dinâmica.

A amplitude é o volume do som. É a quantidade de energia produzida ao longo do tempo. Essa energia é medida em decibéis (dB).

A frequência é o que determina nossa percepção que um som é grave ou agudo. A unidade da frequência é o Hertz (Hz). A frequência é o número de ciclos por segundo, logo uma frequência de 20 Hz possui 20 ciclos por segundo.

O timbre é mais fácil de perceber que explicar. O som é composto por diversas frequências. É o timbre, com o seu conteúdo harmônico, ou seja, conteúdo de frequências distintas, que caracteriza a sonoridade de cada instrumento ou voz. É o timbre que nos permite perceber que instrumentos diferentes tocando a mesma nota possuem sons diferentes, e dá a sensação e percepção que o som do piano é diferente do som do violão.

A duração de um determinado som também irá influenciar na forma que vamos percebê-lo. Trata-se do tempo de vibração de um objeto qualquer em decorrência de uma única excitação.

E finalmente a dinâmica é a variação do volume (amplitude), apresentado por um som, ao longo do tempo.

Quando ouvimos um CD de música temos todas essas percepções reunidas, pois um CD possui uma qualidade que permite um som agradável para o ouvido do ser humano.

Um exemplo tradicional de aplicação usando áudio é a formação de um CD convencional:

- Amostras por segundo (Hz) : 44100
- Quantidade de bits por amostra: 16
- Largura de banda = Amostras por segundo x Quantidade de bits por amostra
- Largura de banda = $44100 \text{ (1/s)} \times 16 \text{ bits}$
- Largura de banda = 705,6 kbps para um som monofônico
- Largura de banda = 1,411 Mbps para um som estéreo

Sendo a largura de banda em um link de comunicações o número de bits por segundo que pode ser transmitido pelo link, podemos dizer que um CD convencional precisa de uma largura de banda de 705,6 kbps para som monofônico e 1,411 Mbps para o som estéreo.

Imagem Estática

Uma imagem digital é uma matriz de valores amostrados denominados pixels. A Figura II.2 apresenta uma imagem.



Figura II.2 – Exemplo de imagem estática (pâncreas)

A imagem tem um grande valor caso ela possa ser representada de forma fidedigna, para tanto ela deve ser transportada até o receptor sem sofrer transformações pelo caminho.

Segundo GONZALEZ E WOODS (2005, p1), os métodos de processamento de imagem é justificável em aplicações de melhoria da informação visual para interpretação humana e processamento de dados de cenas para percepção automática através de máquinas.

Alguns passos são fundamentais para um processamento de imagens, que são:

- Aquisição da imagem
- Pré-processamento
- Segmentação
- Representação e descrição
- Reconhecimento e interpretação

O primeiro passo, aquisição da imagem, significa adquirir uma imagem digital, para isso é necessário sensores e circuitos digitalizadores.

O segundo passo, pré-processamento, significa aplicar técnicas de realce de contrastes, remoção de ruído e isolamento de regiões cuja textura indique a probabilidade de informação alfanumérica, tudo isso objetivando uma melhora da imagem capturada.

O terceiro passo, segmentação, significa dividir a imagem em partes ou objetos constituintes, esse passo se aplicado com algoritmos fracos ou erráticos aumentam a probabilidade de falhas no processamento.

O quarto passo, representação e descrição, os pixels (raw pixel data) devem ser representados como fronteiras ou como regiões completas, dependendo do interesse. Completando esse passo, deve ser especificado um método para descrever os dados de modo que as características de interesse sejam enfatizadas. Ou seja, esse método procura extrair características que resultem em alguma informação quantitativa de interesse ou que sejam básicas para discriminação entre classes e objetos.

O quinto e último passo, reconhecimento e interpretação, começando pelo reconhecimento, significa rotular um objeto, baseando na informação fornecida pelo seu descritor e a interpretação significa dar um significado aos objetos já reconhecidos.

Além dos passos fundamentais para um processamento de imagem com sucesso, alguns elementos desempenham funções distintas que podem influenciar na qualidade do processamento, que são:

- Aquisição
- Armazenamento
- Processamento
- Comunicação
- Exibição da imagem

A aquisição da imagem é feita com sensores apropriados e digitalizadores de modo a converter o sinal elétrico em digital.

O armazenamento tem sido muito importante pois as imagens capturadas tem sido cada vez maiores, tanto pelo fato da quantidade de experimentos, quanto na quantidade de informação necessária para formar uma imagem. Os meios de armazenamento atuais são capazes de suportar as imagens capturadas, uma vez que elas sejam comprimidas a valores aceitáveis.

O processamento da imagem é implementado por software, assim, é dependente dos algoritmos serem bem elaborados de acordo com a solução específica. Obviamente a capacidade de processamento do hardware vai influenciar, mas os algoritmos devem ser criados para atender a escalabilidade do processo.

A comunicação significa uma distribuição da imagem em processos diferentes, ou seja, uma comunicação remota de um ponto ao outro. A comunicação à longa distância, também conhecida como *wan's* ainda é um desafio para os pesquisadores, pois uma imagem possui uma massa excessiva de dados, tornando a transmissão lenta com os atuais meios de transmissão hoje existente. Técnicas de compressão e descompressão de dados são fundamentais atualmente para transmissão de imagem remota.

A exibição é a forma em que o usuário final ou receptor da informação visualizará a imagem. Os monitores, as impressoras, os celulares, etc. são os meios mais comuns utilizados para visualizar imagens digitais.

Uma imagem é representada na memória do computador por métodos de codificação da imagem. O mais conhecido é o RGB, esse nome é formado pelas iniciais *Red*, *Green* e *Blue* respectivamente vermelho, verde e azul. Cada cor é representada por 8 bits (1 byte) e cada pixel é formado pelas três cores, totalizando 24 bits por pixel, permitindo reproduzir até 16,7 milhões de cores. A cor representada em 8 bits permite 256 níveis ou valores por cor. O valor (0, 0, 0) de R, G e B equivale ao preto, e o valor (255, 255, 255) de R, G e B equivale ao

branco. Estas três cores são conhecidas como primárias; o sistema é baseado na combinação da luz emitida por três fontes de luz.

O CMYK é um sistema que usa quatro cores por pixel. Cada cor é representada em 8 bits, totalizando 32 bits, permitindo 256 nível de cor. Como o modo CMYK usa 4 bytes por pixel, ele ocupa 33% mais espaço em memória e em disco do que o modo RGB.

Uma imagem em escala de cinza é um sistema que usa 256 níveis de cinza por pixel, ou 8 bits (1 byte) por pixel. O valor 0 corresponde ao preto, e o valor 255 ao branco. Este modo é o recomendado para armazenar imagens em preto e branco guardando tons contínuos.

O sistema preto e branco cada pixel é representado por um único bit. Neste sistema, cada pixel pode assumir o valor 0 (preto) ou 1 (branco).

O sistema de cor indexada usa de 1 a 8 bits, também conhecido como 256 cores. Neste modo, cada pixel assume um valor presente numa paleta de 256 cores. Programas de manipulação permitem customizar uma paleta e criar uma nova fazendo uma amostragem da imagem a ser convertida. Este modo é útil para aplicações multimídia e para publicar na Web.

Imagem Dinâmica

Quando uma imagem é projetada na retina, ela é retida por um determinado milissegundos antes de apagar. Se uma seqüência de imagem estática for projetada a uma velocidade de trinta ou mais quadros por segundo, o olho não perceberá que está vendo imagens estáticas e sim um vídeo contínuo, ou seja, uma imagem dinâmica, também conhecida como vídeo. (TANENBAUM, 2003b, p.737).

Então, um vídeo digital é uma seqüência de imagens apresentadas numa certa taxa sendo formado por várias imagens estáticas projetadas. Cada imagem é chamada quadro e a quantidade de imagens projetadas por segundo é chamada quadros por segundo ou FPS (*frames per second*). Quanto mais quadros por segundo seu vídeo tiver, melhor, pois mais realista será a imagem. Os Quadros sucessivos apresentam redundâncias, ou seja, apresentam a mesma informação, visto que boa parte da imagem não apresenta movimento. Não sendo necessário enviar mais de uma vez a mesma informação.

Compressão

Existem duas forças opostas que operam na transmissão de dados multimídia. Num sentido, é desejável uma redundância nos dados de modo a garantir a plena recuperação da informação no caso de perda de algum byte na transmissão, mas isso acarretará em mais bytes aumentando o tamanho do arquivo. No outro sentido, gostaríamos de retirar o máximo de redundância dos dados possível, de modo a transmitir o menor número de bytes possível na banda disponível (DAVIE E PETERSON, 2004, p.390).

A máxima taxa de transferência a ser compartilhada tipicamente varia de 10 Mbits/s (tecnologia Ethernet) a 100 Mbits/s (tecnologia fast Ethernet ou FDDI).

Uma das principais razões em comprimir um arquivo está no fato da largura de banda da rede não permitir a transmissão de vídeo em tempo real. Atualmente, com as larguras de bandas disponíveis, reduzir as redundâncias é essencial, esse é o objetivo da compressão de dados.

Os usuários das aplicações multimídia são humanos ou máquinas para reconhecimento de padrão. Humanos podem tolerar alguns erros de informação ou perdas sem afetar a da comunicação. Isto implica que a versão comprimida não necessita representar com fidelidade a informação original. Isto é bem diferente dos dados alfanuméricos que não se tolera qualquer erro ou perda.

A compactação é realizada sempre por dois algoritmos, um na origem realizando a compactação da informação e outro no destino realizando a descompactação.

Os algoritmos de codificação/decodificação são classificados em algoritmos com perdas e algoritmos sem perdas. Os algoritmos com perdas pela característica da remoção da informação alcançam uma taxa de compressão melhor que os algoritmos sem perdas.

A compressão sem perdas de informação assegura que os dados recuperados dos procedimentos de compressão/descompressão são exatamente os mesmos dados originais (DAVIE E PETERSON, 2004, p.400). São aplicados em códigos executáveis, arquivos de texto e dados numéricos, pois os programas que processam tais arquivos de dados não toleram erros nos dados.

A compressão com perdas é quando uma imagem é codificada na origem, transmitidas pela rede e decodificada no destino, ao comparar a imagem no

destino, ela não apresentará a quantidade de bits igual ao que foi gerada, ou seja, ocorreram perdas no processo de compressão.

A compressão com perdas de informações removem dados que não serão restaurados pelo processo de descompressão, porém, as informações perdidas não vão fazer falta no receptor, pois esses dados contêm mais informação do que os olhos ou ouvidos humanos podem perceber. São usados para comprimir imagens estáticas, vídeo, e áudio (DAVIE E PETERSON, 2004, p.400).

O algoritmo com perdas é necessário para dados multimídia devido ao tamanho desses arquivos, visto que, enquanto os algoritmos com perdas comprimem a uma taxa de até 100:1, o algoritmo sem perdas fica a uma taxa de 2:1, valores didáticos. Então, os sistemas com perdas são importantes porque aceitar perda de um pequeno volume pode oferecer uma grande vantagem na hora da transmissão, principalmente se a largura de banda disponível for conhecida. Pois a qualidade da imagem pode ser controlada fazendo uma adaptação do tamanho do arquivo em relação a largura de banda disponível.

Vários algoritmos estão disponíveis para uso para compressão de imagens. A Figura II.3 mostra alguns dos algoritmos mais usados nos modelos de compressão.

O termo algoritmo não é uma palavra única no contexto da compressão, pois o processo de compressão possui princípios, técnicas, modelos e padrões que juntos formam um contexto único para ser aplicado para cada caso de acordo com o tipo de arquivo, a necessidade aplicada e a largura de banda disponível.

Domínio espacial	<ul style="list-style-type: none"> Mod.Delta PCM (Pulse Code Modulation) DPCM (Differential Pulse Code Modulation) Quant. Vetorial Outros 	Métodos baseados em estatística	<ul style="list-style-type: none"> Shannon - Fano Huffman Gilbert Outros
Domínio da frequência	<ul style="list-style-type: none"> Filtro <ul style="list-style-type: none"> Subband Wavelet Outros Transformada <ul style="list-style-type: none"> Fourier KL (Karhunen - Loève) DCT(Discrete Cosine Transform) Outros 	Métodos baseados em dicionário	<ul style="list-style-type: none"> RLE (Run Length Encoder) Aritmético LZW (Lempel - Ziv Welch) Outros

Figura II.3 – Algoritmos de compressão

Cada situação promoverá o uso de uma determinada técnica. O desempenho é um fator a ser levado em consideração na decisão de qual algoritmo a ser utilizado para compressão, logo, a técnica de compressão deve se basear nos requisitos da aplicação. Os parâmetros de desempenho mais usados são: Taxa de compressão, qualidade da mídia reconstituída, complexidade de implementação e velocidade de implementação.

A Taxa de compressão é a razão entre o tamanho do dado original e o tamanho do dado após a compressão.

A qualidade da mídia reconstituída é a medida da razão sinal/ruído. Este parâmetro é aplicável apenas para técnicas com perda. Para a escolha de uma técnica de compressão com perdas, deve-se optar pelo compromisso entre uma alta taxa de compressão e a qualidade desejada para a aplicação.

A complexidade de implementação e velocidade de compressão, são relacionadas, pois, geralmente quanto mais complexa a técnica menor é a velocidade de compressão. No caso de aplicações tempo-real, como videoconferência, estes parâmetros devem ser considerados. Pois a compressão/descompressão devem ser realizadas em tempo-real. No caso de aplicações do tipo obtenção e apresentação de informação a velocidade de compressão não é muito importante, mas a velocidade de descompressão é importante.

Por exemplo, Segundo Davie e Peterson (2004, p.409) o padrão MPEG exige uma computação custosa devido ao envolvimento de cálculos extensos, sendo normalmente feita off-line. Isso permite concluir que não existe uma técnica perfeita.

Algoritmos de Compressão

Código de Huffman

O código de *Huffman* é um algoritmo sem perdas. Sua idéia essencial é obter a probabilidade relativa com que cada símbolo aparece nos dados, então, atribui-se uma quantidade de bits diferente para cada símbolo, de forma a

minimizar o número de bits necessários para codificar um determinado bloco de dados (DAVIE E PETERSON, 2004, p.400).

Neste método de compressão, são atribuídos menos bits a símbolos que aparecem mais freqüentemente e mais bits para símbolos que aparecem menos. Assim, o tamanho em bits dos caracteres codificados serão diferentes.

A Codificação de Huffman é um exemplo de técnica de codificação estatística, que diz respeito ao uso de um código curto para representar símbolos comuns, e códigos longos para representar símbolos pouco freqüentes.

Código Run Length

Run Length Encoding (RLE) é uma técnica simples usada na compressão de repetições de símbolos num “bloco” de dados. A simplicidade do RLE e a facilidade de implementação resulta na sua frequente utilização em compressores, apesar da sua eficácia relativamente pobre.

O RLE é um algoritmo sem perdas. A idéia é substituir ocorrências consecutivas de um determinado símbolo por apenas uma cópia do símbolo, mais a contagem representativa do número de vezes que ele ocorre. Por exemplo, a string AAABBCDDD seria codificada como 3A2B1C3D. Essa técnica é bastante eficaz para imagens que possuam grandes regiões homogêneas resultando em agrupamentos de símbolos iguais. As principais aplicações do método de Run-Length, são em imagens binárias, imagens com grandes espaços envolvendo uma só cor e em imagens geradas por computador, onde os dados estão agrupados de forma mais geometricamente definida (DAVIE E PETERSON, 2004, p.401).

Diferencial Pulse Code Modulation

O código modulação por código de pulso diferencial (DPCM - *Diferencial Pulse Code Modulation*) é um algoritmo sem perdas. Em uma imagem, há redundância de informação entre os pixels vizinhos e, conseqüentemente, o valor de cada pixel pode ser predito por sua vizinhança.

Assim, a codificação preditiva visa eliminar a redundância interpixels presente na informação original, codificando somente a diferença ou resíduo entre o valor do pixel original e o valor predito para este pixel. DPCM é o método que

utiliza a soma do valor do pixel predito com o valor do resíduo para obter o valor do pixel original.

O DPCM gera na saída um símbolo de referência e, então, para cada símbolo nos dados, gerará na saída a diferença entre o símbolo e o símbolo de referência (DAVIE E PETERSON, 2004, p.401). Por exemplo, utilizando o símbolo A de referência, a string AAABBCDDDD seria codificada como A0001123333, já que A é o próprio símbolo de referência, B apresenta uma diferença de 1 em relação ao símbolo de referência, e assim por diante. O ganho está no fato em que a faixa de 0-3 pode ser representada com apenas 2 bits (00, 01, 10, 11), enquanto os caracteres necessitam de oito bits.

Essa técnica é muito útil pois ela aproveita o fato de que os pixels adjacentes nas imagens digitais são usualmente similares.

Codificação Aritmética

A codificação Aritmética é uma técnica que permite que uma sequência de mensagens de uma determinada fonte seja combinada compartilhando os mesmos bits.

Os codificadores aritméticos têm uma maior performance quando a distribuição de probabilidades com que trabalham tem probabilidades muito altas.

Na codificação aritmética não existe uma relação direta entre os símbolos-fonte e as palavras-código. Em vez disso, toda sequência de símbolos-fontes que são as mensagens é atribuída a uma única palavra de código aritmético. Na medida que o número de símbolos na mensagem aumenta, o intervalo usado para representá-la reduz e o número de unidades de informação, requeridos para representar o intervalo, aumenta. Cada símbolo da mensagem reduz o tamanho do intervalo de acordo com sua probabilidade de ocorrência. (GONZALEZ E WOODS, 2005, p.248).

Métodos Baseados em Dicionários

Existem mais de uma codificação baseada em métodos baseados em dicionários. O algoritmo de compressão Lempel-Ziv (LZ ou LZW) é o mais conhecido. foi desenvolvido em 1984 para compactar dados em discos

magnéticos. Hoje em dia, métodos como este tornaram-se populares pelo seu uso em programas de microcomputadores (Doublespace, Zip, etc).

A idéia é construir uma Tabela onde a chave é um número e os dados é a palavra do dicionário. Logo, na ocorrência da palavra no arquivo essa é substituída pelo número correspondente. Por exemplo, a palavra *compression* possui o índice 4978, então na ocorrência desta palavra, será substituído pelo número 4978, significando que a palavra *compression* seria representada por 15 bits em vez dos 77 bits necessários para o código ASCII de 7 bits (DAVIE E PETERSON, 2004, p.402).

Em 1987, a Compuserve criou o formato GIF e passou a usar o LZW como método de compressão. Em 1988, a Aldus lançou a versão 5 do formato TIFF, onde compressão LZW era uma opção. A compressão LZW é obrigatória no formato GIF e uma opção do TIFF. A sua maior característica é não acarretar alguma perda de informação. O LZW funciona muito bem com imagens gráficas, onde a quantidade de cores é discreta e onde existem muitas áreas com tons constantes.

O algoritmo LZW também é encontrado em diversos descompactadores populares. Dentre eles, o PKUNZIP (para o DOS), ARJ, GUNZIP e o Uncompress (para o UNIX).

Compressão de imagem Estática

É possível aplicar compressão em imagens digitais pois as mesmas apresentam redundâncias. Basicamente, existem três tipos de redundâncias: de código, interpixel e psicovisual.

Na redundância de código o processo de codificação atribui à informação códigos com número de bits variáveis de acordo com a probabilidade de ocorrência de determinado tom de cinza ou cor do pixel na cena; ou seja, o nível de cinza ou cor com maior ocorrência, será representado por uma palavra-código de comprimento menor; ao contrário, se um nível de cinza ou cor tem pouca presença na cena, este valor é representado por uma palavra-código maior.

A redundância interpixel em imagens digitais ocorre devido as amostras vizinhas em uma linha de escaneamento e as amostras vizinhas em linhas adjacentes serem similares, chamadas de redundância espacial. A redundância interpixel nos permite prever o valor de um pixel pelos valores de seus pixels

vizinhos, esta correlação espacial está ligada ao relacionamento geométrico entre os objetos na imagem.

A redundância psicovisual está relacionada à informação visual real ou quantificada em uma cena. Portanto, a redução ou a eliminação da redundância psicovisual, leva necessariamente a um processamento com perdas.

O padrão JPEG (*joint photographic experts group* – grupo conjunto de especialistas em fotografia) para compressão de imagens estáticas de tons contínuos (por exemplo, fotografias) foi desenvolvido por especialistas na área conjuntamente patrocinados por outros organismos de padronização como ITU, ISO, IEC.

Embora este padrão ter sido projetado inicialmente para imagens, codificação e decodificação JPEG tempo-real tem sido implementada para vídeo. Esta aplicação é chamada de *Motion JPEG* (MJPEG).

O padrão JPEG apresenta quatro modos de operação: JPEG - DCT, JPEG - DPCM, modo Progressivo e o modo Hierárquico. Normalmente, quando há referência ao padrão JPEG trata-se do JPEG - DCT cujo grau de qualidade e velocidade de descompressão podem ser variados de acordo com os parâmetros da compressão. Isso significa que o tamanho do arquivo da imagem em processamento pode ser regulado de acordo com a qualidade final da imagem desejada.

Segundo TANENBAUM (2003b, p.349) Existem seis passos para a efetivação completa da compressão JPEG conforme a Figura II.4.

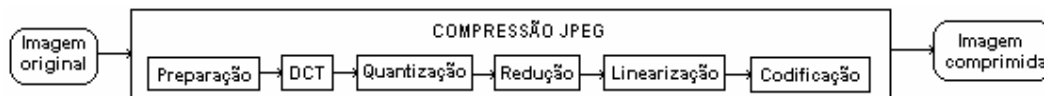


Figura II.4 – Diagrama de blocos JPEG adaptado de Tanenbaum (2003b, p. 349)

Primeiro passo: Preparação do bloco

Segundo Davie e Peterson (2004, p.405 e p.406) imagens com tons de cinza possui um único valor de 8 bits por pixel. No caso de uma imagem colorida existe a representação RGB que representa a cor de cada pixel a partir de três

componentes, logo um pixel será representado por 24 bits (3 x 8). Os componentes são: Vermelho (*red*), verde (*green*), azul (*blue*).

Uma outra representação, chamada YIQ, possui também três componentes: Luminância (Y), e duas cromaticâncias (I e Q).

O sistema visual humano é capaz de distinguir a luminância (brilho) de um pixel muito melhor do que a sua cor.

Cada pixel de uma imagem colorida possui três valores separados. Para comprimir essa imagem, cada uma dessas três componentes é processada de forma independente. Enquanto que na escala de cinza só existe um componente para processar. Logo, uma imagem colorida é processada cada componente como uma imagem de somente nível de cinza.

A preparação do bloco, é calculada a luminância e dois sinais de cromaticância a partir dos valores RGB originais de acordo com a fórmula apresentada em Tanenbaum (2003b, p.742) para o sistema NTSC (*National Television Standards Committee*):

$$\begin{aligned}\text{Luminância} = Y &= 0,30R + 0,59G + 0,11B \\ \text{Cromaticância} = I &= 0,60R - 0,28G - 3,2B \\ \text{Cromaticância} = Q &= 0,21R - 0,52G + 0,31B\end{aligned}$$

Para o sistema PAL, as cromaticâncias são chamadas de U e V e os coeficientes são diferentes, mas a idéia é a mesma (TANENBAUM, 2003b, p.742).

O olho humano é muito mais sensível ao sinal de luminância que aos sinais de cromaticância e, portanto esses últimos não precisam ser transmitidos com a mesma precisão. (TANENBAUM, 2003b, p.739).

Segundo passo: Fase da DCT

Aplicação da DCT (discrete cosine transformation – transformação discreta de cosseno) para todos os blocos de 8x8, um de cada vez.

O formato JPEG consegue alcançar boas taxas de compressão à custa da exploração das limitações da visão humana, a qual apresenta sensibilidades diferentes relativamente às componentes de frequência presentes numa dada imagem. Como a visão humana é menos sensível às componentes de alta frequência do que às de baixa frequência, aquelas podem ser desprezadas sem que daí resultem grandes alterações no conteúdo dessa mesma imagem. O JPEG

é parametrizável neste sentido, quanto maior é a compressão escolhida, menor é o número de componentes de alta frequência desprezados.

Pensando na imagem como um sinal do domínio espacial, então a DCT transforma-o num sinal equivalente no domínio da frequência espacial (DAVIE E PETERSON, 2004, p.403).

A idéia em transformar o sinal para o domínio da frequência é encontrar os pixels de detalhes finos, ou seja, pixels que ocorrem em altas frequências. Os detalhes finos são menos essenciais e, em alguns casos, mal podem ser percebidos pelo olho humano. E as baixas frequências correspondem às características grossas da imagem, se esses elementos forem eliminados o olho humano perceberá com facilidade a alteração da imagem quando for realizado o processo inverso para obter a imagem novamente.

Esta fase será decisiva para decidir quais elementos serão eliminados da imagem. A DCT, juntamente com a sua inversa é definida pelas seguintes fórmulas (DAVIE E PETERSON, 2004, p.403):

$$DCT(i,j) = \frac{1}{\sqrt{2N}} C(i) C(j) \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} pixel(x,y) \cos \left[\frac{(2x+1)i\pi}{2N} \right] \cos \left[\frac{(2y+1)j\pi}{2N} \right]$$

$$pixel(x,y) = \frac{1}{\sqrt{2N}} \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} C(i) C(j) DCT(i,j) \cos \left[\frac{(2x+1)i\pi}{2N} \right] \cos \left[\frac{(2y+1)j\pi}{2N} \right]$$

$$C(x) = \begin{cases} \frac{1}{\sqrt{2}} & \text{se } x=0 \\ 1 & \text{se } x>0 \end{cases}$$

onde,

$pixel(x,y)$ é o valor em tons de cinza do pixel na posição (x,y) no bloco 8x8 que está sendo comprimido;

$N=8$, bloco 8x8 de imagens de tons cinza.

Terceiro passo: Fase da Quantização

Os passos 1 e 2 não sofreram perdas significativas, ou seja, sofreram apenas perdas de arredondamento matemático imperceptível para o olho humano.

O seu objetivo foi transformar a imagem numa forma que torna mais fácil saber que informações remover. Pois, conforme você vai do primeiro coeficiente de frequência em direção ao coeficiente 64, você está movendo das informações de baixa frequência para as informações de alta frequência. Esses coeficientes de frequências mais altas vão se tornando cada vez menos importantes para a qualidade percebida da imagem (DAVIE E PETERSON, 2004, p.404).

Então, a perda resultará após os coeficientes DCT menos importantes contidos no sinal serem eliminados. Essa transformação é realizada dividindo-se cada um dos coeficientes da matriz 8x8 por um peso a partir de uma Tabela de quantização. Cada aplicação deve fornecer sua tabela de quantização particular, possibilitando controlar a perda e a compressão.

Os elementos da Tabela de quantização são denominados *quantum*. Em vez de utilizar o mesmo quantum para todos os 64 coeficientes, o JPEG usa uma Tabela de quantização que fornece o *quantum* a ser usado para cada coeficiente e esse parâmetro é configurado pela aplicação para controlar a quantidade de informação da imagem e a compressão alcançada (DAVIE E PETERSON, 2004, p.404).

Segundo Davie e Peterson (2004, p.405) a equação básica da quantização é:

$$\text{Valor quantizado}(i,j) = \text{Arredondamento Inteiro}(\text{DCT}(i,j) / \text{Quantum}(i,j))$$

Onde,

$$\text{Arredondamento Inteiro}(x) = \begin{cases} [x + 0,5] & \text{se } x \geq 0 \\ [x - 0,5] & \text{se } x < 0 \end{cases}$$

Tabela II.2 –Tabela de quantização JPEG em Davie e Peterson(2004, p.404)

Quantum=	3	5	7	9	11	13	15	17
	5	7	9	11	13	15	17	19
	7	9	11	13	15	17	19	21
	9	11	13	15	17	19	21	23
	11	13	15	17	19	21	23	25
	13	15	17	19	21	23	25	27
	15	17	19	21	23	25	27	29
	17	19	21	23	25	27	29	31

Estando disponível na aplicação a tabela de quantização da Tabela II.2 é possível obter como saída uma matriz 8x8 de coeficientes de frequência quantizados conforme a seguir:

Por exemplo, se o coeficiente DCT (0,0) para um bloco de uma imagem particular for igual a 25, então a quantização desse valor usando a Tabela II.7 resultará em $[25/3 + 0,5] = 8$.

A descompressão obterá o mesmo coeficiente da seguinte forma: $8 \times 3 = 24$.

Quarto passo: Fase da Redução

Redução de valor (0,0) de cada bloco substituindo-o pelo tanto que ele difere do elemento correspondente no bloco anterior. O primeiro coeficiente de frequência na posição (0,0) na matriz de saída é conhecido como coeficiente DC e os outros 63 elementos da matriz de saída são chamados de componentes AC. O coeficiente DC é uma medida do valor médio dos 64 pixels de entrada. O

Quinto Passo: Fase da linearização

Linearizar os elementos aplicando a codificação run-length a essa lista de elementos.

Sexto passo: Fase da codificação

Codificação de Huffman para armazenamento ou transmissão, ou seja, a compressão propriamente dita.

Segundo Davie e Peterson (2004, p.403), as fases de *redução*, *linearização* e *codificação* formam um único bloco que codifica os coeficientes de frequência quantizados em uma forma compacta, formando um bloco de codificação sem perdas, e permite usar a codificação aritmética em vez do código de Huffman. Essa visão é mostrada na Figura II.5:

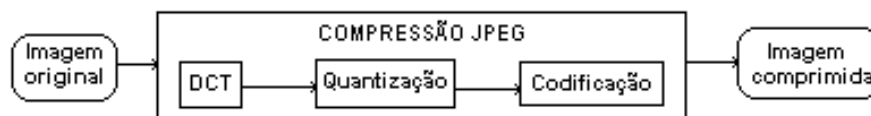


Figura II.5 – Diagrama de blocos JPEG segundo Davie e Peterson (2004, p. 403)

Davie e Peterson (2004) e Tanenbaum (2003b) aplicam os mesmos conceitos quanto ao padrão JPEG, inclusive quando afirmam nas entrelinhas que o JPEG não é apenas um algoritmo de compressão, eles definem um formato para os dados de imagens ou vídeos.

A taxa de compressão do padrão JPEG será menor ou maior conforme a tabela de quantização fornecida ou controlada pela aplicação podendo atingir 2:1 ou até 100:1, conforme a necessidade de qualidade requerida pela aplicação.

Este formato apresenta ótimas taxas de compressão para imagens fotográficas naturais de tons contínuos diminuindo consideravelmente quando aplicado a imagens gráficas com contornos e áreas bem definidas de cor ou a imagens com texto, como é o caso dos logotipos.

Compressão de Vídeo

A transmissão de vídeo não compactado está fora de cogitação nas atuais largura de banda disponíveis atualmente. Por exemplo, uma imagem é gerada para ser apresentada com uma suavidade na sequência de apresentação dos quadros além de não poder apresentar oscilações. Então considere que essa imagem possua uma resolução de 1024x768 com 8 bits por pixel e 25 quadros por segundo, será necessário uma largura de banda de 157,3 Mbps ($1024 \times 768 \times 8 \times 25$). Logo, aplicar técnicas de compactação em imagens dinâmicas é fundamental para tornar a transmissão possível.

Uma maneira de diminuir o tamanho do vídeo é diminuir a quantidade de quadros por segundo. O tamanho do vídeo diminui, e sua qualidade também, pois os quadros são quebrados, isto é, os movimentos no vídeo ficam visíveis pelo processo da retina, perdendo a característica de continuidade.

Uma outra maneira de diminuir o tamanho do vídeo é atuar na sua resolução, gerando quadros com resolução menores conforme a Tabela II.3. Cada *codec* de vídeo opera com uma faixa possível de quadros por segundo e com um conjunto de resoluções que influenciaram diretamente na taxa de transmissão utilizada (COSTA, 2007, p.18).

Apresentar multimídia em tempo real, como videoconferência ou a realização de um exame de ultra-som a distância é necessário que a codificação e a decodificação ocorra em tempo real, ou seja, esses vídeos não são armazenados em discos antes da transmissão, eles são capturados diretamente

da fonte geradora e codificados automaticamente e transmitidos pela rede ao seu destino e decodificado automaticamente para ser apresentado.

Vídeo digital é uma sequência de imagens, portanto ele tem redundância espacial herdada pela imagem estática. A redundância espacial pode ser eliminada codificando cada quadro em separado com o JPEG. Além disso, imagens vizinhas em vídeos são geralmente similares. Esta redundância é chamada de redundância temporal.

Tabela II.3- Resolução de vídeos em Costa (2007, p. 18)

FORMATO	SIGNIFICADO DA SIGLA	RESOLUÇÃO
SVGA	Super VGA	800 x 600 pixels
VGA	Video Graphics Array	640 x 480 pixels
16CIF	16 vezes CIF	1048 x 1152 pixels
4CIF	4 vezes CIF	704 x 576 pixels
CIF	Common Intermediate Format	352 x 288 pixels
QCIF	Quarter CIF	176 x 144 pixels
SQCIF	Sub-quarter CIF	128 x 96 pixels

MPEG

Segundo Tanenbaum (2003b, p.351) os padrões MPEG (*motion picture experts group* – grupo de especialistas em imagens em movimento) é o principal algoritmo usado para comprimir vídeos, sendo padrão internacional desde 1993.

Então, o padrão multimídia para imagens em movimento, MPEG, é apenas a codificação JPEG de cada quadro em separado, mais alguns aspectos adicionais para compressão entre os quadros e de compensação de movimento (TANENBAUM, 2003b, p.349).

O grupo MPEG criou vários padrões MPEG:

- O MPEG1 é usado para Vídeo CD;
- O MPEG2 é usado para DVD e TV Digital;
- O MPEG3 ou simplesmente mp3 é usado para áudio e
- O MPEG4 é o padrão orientado a objetos que em breve dominará a internet.

O padrão MPEG2 recebe uma sequência de quadros de vídeo como entrada e comprime esses quadros em três tipos diferentes, chamados quadros do tipo *I* (*Intracontidos*), quadros do tipo *P* (*preditivo*) e quadros do tipo *B* (*preditivo*).

bidirecional), Esses três tipos de quadros devem ser processados pelo programa de visualização (DAVIE E PETERSON, 2004, p.406).

Os quadros *I* (Intracodificados) são quadros de referência, eles são autocontidos, e não dependem de outros quadros, fazendo uma analogia, esses quadros são simplesmente a versão JPEG do quadro correspondente na fonte de vídeo.

Os quadros *P* (Preditivos) não são autocontidos, ele especifica as diferenças em relação ao quadro *I* anterior. Ou seja, contém as diferenças bloco por bloco com o último quadro *I*.

Os quadros *B* (preditivos Bidirecionais) não são autocontidos, ele especifica as diferenças entre o último e o próximo quadro, realizando uma interpolação entre os quadros anterior e subsequente, sejam eles do tipo *P* ou *I*.

A Figura II.6 é um exemplo de codificação usando a compressão MPEG que mostra a sequência de quadros *I*, *P* e *B*. Os dois quadros *I* são independentes, o quadro *P* depende do quadro *I* anterior e o quadro *B* depende tanto do quadros anteriores (*I* ou *P*) quanto dos subsequentes. Para que o MPEG possa descomprimir a imagem os quadros de referência (*B*) deverão estar presentes (DAVIE E PETERSON, 2004, p.407).

Não existe uma proporção entre quadros *I* em relação aos quadros *P* e *B*. A proporção pode variar dependendo da relação qualidade da imagem e largura de banda disponível. É possível enviar somente quadros *I*, isso seria equivalente a um codificação JPEG para vídeo. Mas quanto mais quadros *I*, menor a compressão, pois as redundâncias temporais não seriam eliminadas, ou seja, as partes que são consideradas fixas na imagem seriam sempre enviadas. O que não ocorre quando se aplica a compressão interquadros (redundância temporal), onde são enviadas apenas as diferenças entre quadros sucessivos.

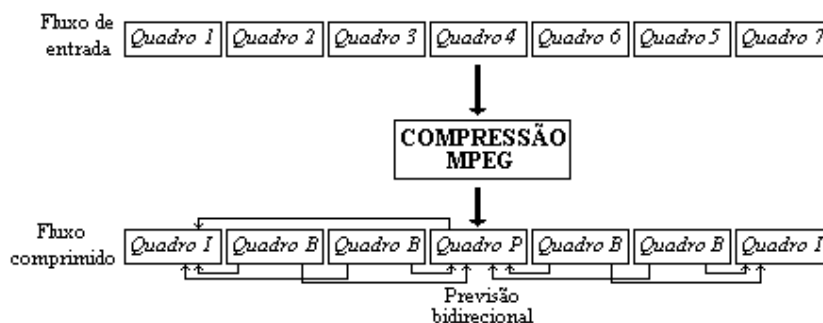


Figura II.6 – Quadros *I*, *P* e *B* do MPEG em Davie e Peterson (2004, p. 407)

Sendo mais fiel, os quadros I são aproximadamente iguais à codificação JPEG. A diferença está no fato que o MPEG trabalha em macroblocos de 16x16 para um vídeo colorido representado em Y I Q, as componentes I e Q de cada macrobloco tem sua taxa de amostragem diminuída, resultando em blocos 8x8. Segundo Davie e Peterson (2004, p.409) o padrão MPEG foi projetado para taxas da ordem de 1,5 Mbps e a compressão é normalmente feita off-line exigindo uma computação bastante custosa (DAVIE E PETERSON, 2004, p.407 à p.409).

Série H

O padrão MPEG não é o único padrão de codificação disponível a ser considerado para transmissão de vídeo. O ITU-T também definiu a série H para codificação em tempo real de dados multimídia. A série H também inclui padrões para áudio e dados em um único fluxo de bits. A primeira geração da série denomina-se H.261 e a segunda geração denomina-se H.263. Enquanto o padrão MPEG trabalha com taxas de 1,5 Mbps, a série H foi projetada para trabalhar com velocidades compatíveis com o ISDN, ou seja, larguras de banda na ordem de 64 Kbps. A série H é similar ao MPEG, também usa a DCT, quantização e compressão interquadro (DAVIE E PETERSON, 2004, p.409).

O padrão H.261, é um algoritmo que combina codificação intraquadro e interquadro (redundância espacial e temporal) para fornecer um processamento para compressão/descompressão em tempo-real para transmissão de vídeo.

A principal diferença entre o MPEG e o H.261 é a maneira pela qual a estimação de movimento é tratada. A padronização H.261 especifica que cada quadro seja comparado apenas com o quadro anterior, enquanto a padronização MPEG não define o número de quadros que são usados no processo de estimação de movimento. (GONZALEZ E WOODS, 2005, p.288).

O H.263 é um padrão de vídeo a baixa taxa de bits para aplicações de teleconferência que opera a taxas abaixo de 64 Kbps. A codificação de vídeo é uma extensão do H.261 e descreve um método de codificação DPCM/DCT.

O H.263, baseado no MPEG, possui um quadro denominado PB. Ele consiste de duas imagens codificadas em uma unidade. Um quadro PB consiste de um quadro P (MPEG) que é produzido a partir do último quadro P decodificado e um quadro B (MPEG) que é produzido a partir do último quadro P decodificado e do quadro P sendo decodificado.

Compressão de Áudio

A telefonia digital já vem trabalhando para aprimorar a qualidade do áudio digital. Vários produtos já dispõem atualmente, como no caso do *Skype*, *MSN* da *Microsoft*. Esses produtos só possuem o inconveniente da propriedade intelectual protegida. O framework da parceria IBM e SUN, *JMF*, é de código aberto e atende as recomendações atuais de compressão.

Amostragens de áudio adjacentes são similares. A amostra futura não é completamente diferente da passada, o próximo valor pode ser previsto baseado no valor atual. A técnica de compressão que se aproveita desta característica do áudio é chamada de codificação preditiva. Técnicas de compressão preditiva armazenam a amostra anterior para ajudar construir a próxima amostra.

Na codificação preditiva, em vez de se transmitir uma amostra, a diferença entre uma previsão do valor da amostra e do valor real é codificada e transmitida. Esta diferença é chamada de erro de predição. Se esta diferença é quantificada e codificada, o esquema de codificação é chamado de PCM diferencial (DPCM). O qual já foi citado.

A modulação *delta* apresenta alguns problemas. O variante mais prático é chamado de DPCM adaptativo (ADPCM). Para manipular sinais que mudam rapidamente tão bem quanto sinais que mudam lentamente, o tamanho do passo de quantificação aumenta com o aumento da variação do sinal. Então se a forma de onda está trocando rapidamente, grandes passos de quantificação são utilizados e vice-versa.

A ITU-TS recomenda uma série de esquemas de compressão de voz, com uma suposição de que a largura de banda do áudio está dentro de 3,4 kHz até 7 kHz.

O G.711 define a representação de voz não compactada, usando modulação PCM e largura de banda de 3,4 KHz.

A recomendação G.721 converte um fluxo de bits de 64 Kbits/s em um fluxo de 32 Kbits/s. G.721 é baseado na técnica ADPCM e sua taxa de amostragem é de 8 kHz e largura de banda de 3,4 KHz.

A recomendação G.722, usa 8 bits para cada amostragem, É baseado no método ADPCM sub-banda, onde o sinal de voz é dividido em duas sub-bandas: bandas de alta e de baixa frequência. O propósito desta subdivisão é que a sub-banda de baixa frequência é mais importante, assim ela precisa ser codificada

com maior precisão. A largura de banda de um sinal compactado G.722 varia de 50 Hz a 7 kHz.

G.723 é outro padrão de compressão sem perdas baseado no ADPCM que opera a 24 Kbits/s e largura de banda de 3,4 KHz.

O G.728 é para baixas taxas de bits. Ele opera a 16 Kbits/s mas sua largura de banda é limitada a 3,4 kHz. Ele usa uma técnica de quantificação vetorial, onde um número de amostragens é agrupado em um vetor e cada vetor é representado por um índice na entrada do dicionário.

Conforme citado anteriormente em áudio, um CD convencional possui os seguintes dados:

- Amostras por segundo (Hz): 44100
- Quantidade de bits por amostra: 16
- Largura de banda=Amostras por segundo x Quantidade de bits por amostra
- Largura de banda = 44100 (1/s) x 16 bits
- Largura de banda = 705,6 kbps para um som monofônico
- Largura de banda = 1,411 Mbps para um som estéreo

Fazendo uma comparação com a voz humana usando os seguintes dados:

- Amostras por segundo: 8 KHz
- Quantidade de bits por amostra: 8
- Largura de banda=Amostras por segundo x Quantidade de bits por amostra
- Largura de banda = 8000 (1/s) x 8 bits
- Largura de banda = 64 Kbps

Um link ISDN possui uma taxa de 64 Kbps, sendo esse, a base instalada na Internet no território brasileiro. Então para transmitir áudio com qualidade de CD por uma rede ISDN também é necessária alguma compressão.

A ITU-TS, conforme já citado, recomenda a série G.xxx para voz humana, outras técnicas são usadas de forma genérica para comprimir sons dentro da faixa dos sons audíveis, ou seja, sons até 20 kHz.

Segundo Davie e Peterson (2004, p.413) o padrão MPEG pode ser usado para comprimir o áudio de um filme intercalado com o vídeo em um único fluxo ou pode ser usado para comprimir apenas áudio com qualidade de CD.

Assim como toda informação multimídia, o áudio precisa ser compactado para ser transmitido pela rede. Segundo Tanenbaum (2003b, p. 720) o mais

popular modelo de compressão de áudio é o MPEG, sendo o MP3 (MPEG audio layers) o mais eficiente e conhecido.

Segundo Davie e Peterson (2004, p.413) o MPEG não define apenas como o vídeo é comprimido, mas também define uma padrão para compressão de áudio.

Ainda em Tanenbaum (2003b, p.720) tem-se duas maneiras de realizar compressão em áudio, a primeira trabalha na codificação de forma de onda, onde o sinal é transformado matematicamente por uma transformação de Fourier em seus componentes de frequência, o objetivo é reproduzir com precisão a forma de onda com menor quantidade de bits possível.

A segunda é a codificação perceptiva, que explora falhas no sistema auditivo humano, baseado na ciência da psicoacústica, que é a forma que as pessoas percebem o som. O MP3 é baseado na codificação perceptiva, que efetua a transformação de Fourier do som para obter a potência em cada frequência, e depois transmite apenas as frequências não-mascaradas, codificando-as na menor quantidade de bits possível.

O princípio de funcionamento básico do mp3 é buscar num sinal de áudio normal todos os sinais redundantes e irrelevantes que não sensibilizam nosso ouvido. O algoritmo de compactação do mp3 corta frequências muito altas, acima dos 20kHz, que não são audíveis pelo ouvido humano. O mp3 diminui o número de bits desse sinal mais fraco e mantém os bits do sinal mais forte, diminuindo o tamanho final do arquivo. Para alcançar a compressão desejada, o MP3 utiliza as seguintes técnicas segundo Davie e Peterson (2004, p.413):

- Divisão do fluxo de áudio em um número de sub-bandas de frequência;
- Cada sub-banda é dividida em uma sequência de blocos de modo similar aos macroblocos do MPEG;
- Cada bloco é transformado usando um algoritmo DCT modificado, quantizando e codificando por meio do dódigo de Huffman, da mesma forma que o MPEG.

Outros Formatos de Compressão

Diferentes fabricantes de equipamentos digitais e programas de computador resultaram em uma grande quantidade de formatos de arquivos. A

seguir será apresentado superficialmente algumas soluções para armazenamento de imagens:

GIF (Graphics Interchange Format)

Criado pela *Compuserve* para a transmissão de imagens do tipo *bitmap* pela *Internet*. A primeira versão do GIF surgiu em 1987. Imagens GIF são sempre comprimidas e codificadas pela especificação LZW. Atualmente, este é um dos formatos de armazenamento de imagens sem perdas que oferece as melhores taxas de compressão.

O formato GIF admite o tratamento de imagens com uma profundidade de cor de até 8 bits/pixel, ou seja, imagens com um máximo de 256 cores. Se possuímos uma imagem true color, com 24 bits/pixel, ao convertermos esta imagem para o formato GIF, estamos perdendo grande parte da informação de cor. Neste caso, verifica-se que a qualidade da imagem obtida é bastante inferior à qualidade da imagem original, devido ao aparecimento de ruído em toda a imagem. Uma imagem GIF pode ser lida e gravada infinitas vezes e sempre será idêntica à original. Essa característica é imposta pela codificação LZW.

JFIF (JPEG File Interchange Format)

Como JPEG especifica apenas um método de compactação de imagens. O padrão JFIF é uma implementação do JPEG e foi criado para que os programas de manipulação possam trocar dados de um modo compatível. Atualmente, todos os programas são capazes de gravar imagens compactadas em JPEG num arquivo no formato JFIF. Esse formato ainda não é largamente usado.

TIFF (Tagged Image File Format)

É um formato de arquivos que praticamente todos os programas de imagem aceitam. Foi desenvolvido em 1986 pela *Aldus* e pela *Microsoft* numa tentativa de criar um padrão. O TIFF é capaz de armazenar imagens em 24 ou 32

bits. Permite que imagens sejam comprimidas usando o método LZW, o que deu a fama de imagens com alta qualidade.

PICT (PCT)

É o formato gráfico nativo do *Macintosh*, permite armazenar imagens *raster* e vetoriais ao mesmo tempo. Na imagem *raster*, também conhecida como *bitmap*, a informação gráfica é descrita como um conjunto de píxeis, cada píxel tem uma localização específica e uma cor atribuída a ele. Exemplos de programas que usam imagens *bitmap*: *Photoshop*, *Photo Paint*, *Microsoft Paint*. Na imagem Vetorial, a informação gráfica é descrita em termos de equações matemáticas ou de objetos discretos, tais como círculos, linha, retângulos, etc. Usado normalmente em programas de CAD e desenho.

BMP (Windows bitmap)

É o formato gráfico nativo do *Windows* da *Microsoft*. É capaz de armazenar cor em até 24 bits, e muito popular em ambiente PC. Esse formato sobrevive devido à popularidade do *Windows*, muitos programas, inclusive em *Macintosh*, suportam o formato BMP.

PNG

O algoritmo PNG surgiu devido ao GIF usar o algoritmo LZW de propriedade da Unisys. No início este algoritmo era de domínio público, no entanto, há relativamente pouco tempo, a Unisys resolveu passar a cobrar uma taxa pela sua utilização. Por este motivo, começou a pensar-se numa alternativa válida ao formato GIF, tendo surgido o formato PNG (*Portable Network Graphics*). O PNG suporta até true color 48 bits por pixel ou 16 bits por pixel para escalas de cinza e não suporta animação. PNG usa os algoritmos de compressão Lempel-Ziv 77 (LZ77) e de Huffman.

Transmissão Multimídia

Limitações do Modelo do Melhor Esforço

A *Internet* em seu formato original possui um modelo de entrega de dados denominado “fim a fim”. Este faz o melhor esforço para entregar o dado até o seu destino. Esse serviço possui limitações que acarretam perdas de pacotes, atraso fim a fim e variação de atraso (*jitter*).

Dada a limitação do serviço do melhor esforço, algumas decisões quanto ao projeto podem ser tomadas para melhorar a qualidade de uma aplicação de rede multimídia percebida pelo usuário (KUROSE E ROSS, 2006, p. 445):

- Enviar áudio e vídeo sobre UDP.
- Retardar a reprodução no receptor, por exemplo 100ms, para reduzir a variação de atraso induzida pela rede.
- Colocar marca de tempo nos pacotes no remetente de modo que o receptor saiba quando ele deve ser reproduzido.

A Figura II.7 mostra o efeito do *buffer* no atraso variável de rede. O receptor enviará os *bits* em uma taxa constante, o cliente também deve reproduzir em taxa de *bit* constante. Sendo assim, o buffer será um condensador que diminuirá a flutuação do atraso gerado pela rede. O tamanho do *buffer* variará conforme a necessidade da aplicação, sendo inclusive adaptado para melhorar a qualidade final do resultado no cliente.

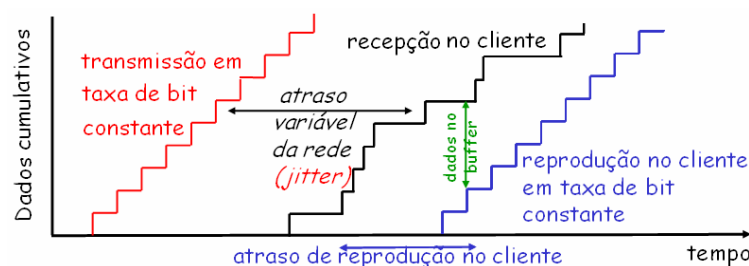


Figura II.7 – Buffer de reprodução adaptado de Kurose e Ross (2006, p.458)

A perda de pacote ocorre por exemplo, em uma transmissão UDP, um datagrama trafegando pela rede que encontre roteadores com *buffers* lotados, nesse caso, o datagrama será descartado e nunca chegará ao destino por uma característica da transmissão UDP.

O atraso fim a fim é apresentado na Figura II.8 e segundo Kurose e Ross (2006, p.28), uma rede é formada por vários nós (sistema final ou roteador). Quando um pacote viaja de um nó para outro em direção ao destino final, sofre um atraso nodal. O atraso nodal é formado pelos atrasos de processamento, atraso de fila, atraso de transmissão e atraso de propagação.

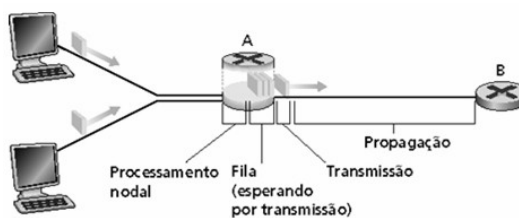


Figura II.8 – Tipos de atrasos segundo Kurose e Ross (2006, p. 29)

O atraso de processamento ocorre devido à necessidade de executar algumas verificações no pacote antes de encaminhá-lo. Entre essas verificações destacam-se os exames do cabeçalho do pacote para direcioná-lo ao seu destino e verificações de erros de bits no pacote.

O atraso de fila depende se o tráfego estiver congestionado ou não. Se o tráfego estiver congestionado, uma fila é criada gerando uma espera para a transmissão do pacote no enlace.

O atraso de transmissão é uma relação entre o tamanho do pacote e a velocidade de transmissão do enlace (*tempo de transmissão = tamanho / largura de banda*).

O atraso de propagação é uma relação da distância entre os roteadores (ou roteador e computador) e a velocidade de propagação do enlace, o qual depende do meio físico do enlace, sendo $3,0 \times 10^8$ m/s no vácuo, $2,3 \times 10^8$ m/s em um cabo e a $2,0 \times 10^8$ m/s na fibra ótica.

A variação de atraso ou *Jitter*, são os atrasos aleatórios que ocorrem nas filas dos roteadores, causando uma variação de atraso nos pacotes dentro do

mesmo fluxo. Essa variação é prejudicial para as mídias contínuas, como o áudio e o vídeo, pois ocasionam perdas de continuidade na reprodução.

Desempenho

“A largura de banda de uma rede é dada pelo número de bits que pode ser transmitido pela rede durante um determinado período de tempo e a latência corresponde ao tempo gasto por uma mensagem para percorrer toda a rede, de um extremo ao outro”, então, a latência é formada pelos atrasos nodais ocorridos ao longo do caminho entre a origem e o destino (DAVIE E PETERSON, 2004, p.29).

. Essas definições servem para medir o desempenho de uma rede de transmissão de dados. Analisando as definições, se uma rede possui uma largura de banda de 10 Mbps, ela é capaz de transmitir 10 milhões de *bits* em um segundo. Uma rede com uma latência de 24 milissegundos(ms) significa que a mensagem vai levar esse tempo para ser transmitida entre a origem e o destino levando em consideração todos os nós existentes pelo caminho.

Então, quantos bytes podem ser transmitidos em uma rede? A quantidade de *bytes* está relacionada ao produto entre a latência e a largura de banda, a Figura II.9 faz uma analogia onde latência corresponde ao comprimento do duto, e a largura de banda ao seu diâmetro, então essa relação é vista como um volume que determina a quantidade de bits suportada entre dois pontos na rede.

Sendo assim, a quantidade de *bits* é dada pela relação entre a largura de banda e a latência.

$$\text{quantidade de bits} = \text{largura de banda} \times \text{latência}.$$

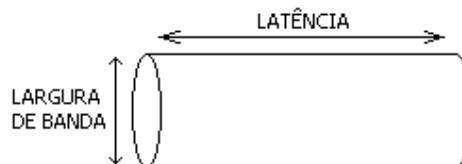


Figura II.9 – Latência x Largura de Banda segundo Davie e Peterson (2004, p.33)

Aplicações em Tempo Real

Segundo Davie e Peterson (2004, p.350 e p.360), uma aplicação de voz é dita em tempo real porque possui características que não suportam atrasos para chegar no receptor a tempo de serem reproduzidas, essa premissa também é válida para transmissão de um vídeo.

Ainda no exemplo da voz, cada amostra possui um tempo de reprodução de 125 μ s depois da amostra anterior. Se os dados chegarem depois do seu tempo de reprodução, eles serão inúteis. Essa inutilidade caracteriza uma aplicação em tempo real.

Devido aos atrasos apresentados em uma rede de comutação de pacotes e principalmente o fato desses atrasos serem variáveis com o tempo, devido às condições de utilização da *Internet (jitter)*, é necessário que os dados no receptor seja depositado em *buffer* antes da sua reprodução, fornecendo uma reserva de pacotes aguardando para serem reproduzidos. Se um pacote atrasar além do tempo do *buffer* ele será descartado.

O tamanho do buffer é alterado para melhorar a relação qualidade da reprodução. Para ser em tempo real tem que gerar a sensação de estar sendo reproduzido ao mesmo tempo em que está sendo gerado, mas se o *buffer* for pequeno a ponto de gerar um número grande de perdas de pacote, a qualidade de reprodução será comprometida. A Figura II.7 ilustra um *buffer* de reprodução. O valor do *buffer* vai depender do tipo de aplicação. No exemplo da voz um tempo acima de 300 ms comprometerá uma conversa com outra pessoa, assim é necessário que a rede garanta que os dados cheguem dentro dos 300 ms.

Classificação das Aplicações

Conforme já mencionado, basicamente existem dois tipos tradicionais de aplicações: aplicações elásticas e em tempo real. As aplicações de interesse nesse caso são as aplicações em tempo real.

Segundo Davie e Peterson (2006, p. 359) e conforme ilustrado na Figura II.10, as aplicações em tempo real são tolerantes a perdas ou não, essas perdas são os pacotes que chegaram atrasados e foram descartados por sua inutilidade

ou os pacotes perdidos durante o caminho, sendo assim, trata-se de um processo natural do serviço de melhor esforço.

A segunda forma de classificar uma aplicação em tempo real é quanto a sua adaptabilidade, ou seja, adaptar a reprodução segundo as características de chegada do pacote, por exemplo, se num determinado momento de pico a maioria dos pacotes só conseguem chegar dentro de 300ms, o ponto de reprodução deve ser ajustado para este valor, porém após a passagem do ponto de pico os pacotes chegarem em sua maioria com 100 ms, a aplicação deve voltar o ponto de execução para o melhor valor.

E por último, uma aplicação é adaptável ao retardo, ou à taxa, Ao retardo são as aplicações que podem ajustar seu ponto de reprodução conforme já explicado, e à taxa, são as que modificam o algoritmo de compressão alterando os seus valores de codificação de acordo com a largura de banda disponível, ou seja, aumentar ou diminuir a qualidade segundo a demanda de largura de banda.

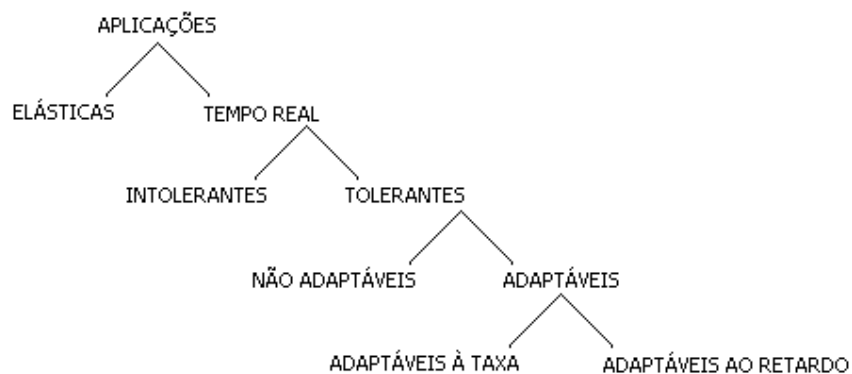


Figura II.10 – Classificação das aplicações, segundo Davie e Peterson (2004, p.362)

Qualidade de Serviço (QoS)

O dado não possui nenhuma prioridade quanto ao seu tipo (*content-type*), ou seja, não existe uma qualificação se o dado é uma mensagem de *e-mail*, uma conversa telefônica ou um vídeo. Os dados serão enfileirados nos roteadores em ordem de chegada para serem transmitidos recebendo o mesmo tratamento. Se as filas estiverem cheias, esses pacotes serão descartados.

A tentativa de retransmitir a informação caso a mesma não chegue ao destinatário não é uma boa opção em se tratando de áudio e vídeo, pois esses são sensíveis ao tempo e a retransmissão aumentaria a latência total.

Fica claro que os *hosts* não são os únicos responsáveis em transmitir a informação com qualidade, os roteadores também possuem esta tarefa. Desta forma, a rede deve tratar alguns pacotes de forma diferente dos outros, o que não é feito no modelo do melhor esforço. Redes que oferecem diferentes níveis de serviços são consideradas como admitindo qualidade de serviço (DAVIE E PETERSON, 2004, p.358 e p.359).

Então, existem aplicações tradicionais que não exigem tratamento especial e o modelo “fim a fim” mesmo com as limitações, atende os requisitos exigidos. São exemplos de aplicações tradicionais: *Telnet*, *FTP*, correio eletrônico, navegação na *web* e outras. Outra definição é dada pelo termo “aplicações elásticas”, pois são capazes de esticar muito bem se houver maiores atrasos.

Existe um outro grupo que somente com o surgimento de técnicas mais eficazes de transmissão e o aumento da banda passante estão se tornando populares, são as denominadas aplicações de tempo real, são consideradas aplicações de tempo real: Videoconferência, telefonia digital, aplicações de robótica e outras. Estas aplicações terão êxito em redes que admitir qualidade de serviço (DAVIE E PETERSON, 2004, p. 359).

Como o modelo de melhor esforço não atende os requisitos de uma aplicação multimídia, há necessidade de um modelo mais rico que atenda às necessidades de qualquer aplicação. Segundo Davie e Peterson (2006, p. 363), as técnicas existentes estão divididas em duas características gerais, as técnicas minuciosas, que oferecem QoS a aplicações ou fluxos individuais e as técnicas grosseiras, que oferecem QoS a grandes classes de dados ou tráfego agregado. Ambas são consideradas técnicas de qualidade de serviço que permite que aplicações com variáveis de retardo, perda e outras tenham esses requisitos atendidos através de mecanismo não existentes no serviço de melhor esforço.

Na primeira categoria se encontra o *Integrated Services*, traduzindo, Serviços Integrados. Uma categoria desenvolvida pelo IETF e normalmente associada ao protocolo de reserva de recursos. Então essa arquitetura trabalha com reserva de recursos a fluxos individuais. Ou seja, permite que fluxos de aplicações individuais especifiquem suas necessidades aos roteadores usando um mecanismo de sinalização explícito através do protocolo RSVP.

Segundo Kurose e Ross (2006, p.491), O *Intserv* tem a função de fornecer garantias de qualidade de serviço específicas às sessões de aplicações individuais e para isso possui duas características fundamentais:

- Reserva de recursos: Um roteador tem de saber que quantidade está reservada para a sessão em andamento.
- Estabelecimento de chamada: Uma sessão que exige QoS deve primeiramente estar habilitada a reservar recursos suficientes em cada roteador da rede em seu trajeto. Exigir a participação de cada roteador no trajeto. Cada roteador deve determinar os recursos locais exigidos, considerando os recursos já comprometidos determinando se o mesmo possui recursos para atender a demanda. Tudo isso sem comprometer a qualidade de serviço.

Na segunda categoria estão os *Differentiated Services*, traduzindo, serviços diferenciados. Este serviço foi padronizado pelo IETF. Esse aloca recursos a um pequeno número de classes de tráfego, ou seja, atribui pacotes a um pequeno número de classes, que recebem tratamento diferenciado nos roteadores. “O *Diffserv* tem o objetivo de prover a capacidade de manipular diferentes classes de tráfego de modos diferentes dentro da *Internet*” (KUROSE E ROSS, 2006, p.491).

Aplicações Multimídia

São as aplicações responsáveis por trabalhar com informações de áudio, vídeo, e dados convencionais e fazem interface diretamente com o usuário final. Essas aplicações normalmente realizam os seguintes requisitos:

- Captura da mídia;
- algoritmos de compressão e descompressão (*codecs*);
- controle da comunicação, também chamado de sinalização;
- Transmissão em tempo real;

H.323

Segundo Kurose e Ross (2006, p.476) o H.323 é um padrão popular para transmissão multimídia entre sistemas finais na *Internet*. Este padrão é uma recomendação do ITU para o controle de conferências multimídia sobre redes TCP/IP. Trata-se de um padrão da série H do ITU-T.

O padrão H.323 não é considerado um protocolo, mas sim uma especificação que estabelece padrões para codificação, transmissão, controle e decodificação de dados multimídia, logo, trata-se de uma solução composta de uma série de protocolos que cooperam entre si para realizar comunicação em tempo de real de dados multimídia.

Em Costa (2007, p.106) é apresentada uma pilha de protocolos do padrão H.323, sendo relatado a seguir e ilustrado na Figura II.11.

- H.225: Abertura de conexões e controle básico das conferências.
- H.235: Utilizado para oferecer certo nível de segurança à comunicação.
- H.245: Abertura de canais lógicos e controle geral das conferências.
- RAS: Interação entre terminais H.323 e *gatekeepers*.
- T.120: Tratamento dos dados

CODECS	RTCP	CONTROLE DE CONFERÊNCIA			DADOS
RTP		RAS	H.225	H.245	T.120
UDP		TCP/SCTP			
IPv4/IPv6					
INTERFACE DE REDE					

Figura II.11 – Arquitetura H.323

Os protocolos RTP e RTCP não fazem parte da especificação do padrão, porém é recomendado. O protocolo trabalha tanto em TCP quanto UDP. Desta forma, a arquitetura ou pilha de protocolos, formam uma solução para transmissão de dados multimídia em tempo real pela *Internet*.

Segundo Costa (2007, p.115), este padrão possui serviços específicos através da implantação opcional de alguns elementos, estes elementos são: MCU (*Multipoint Control unit*), *gatekeepers* e *gateways*. Estes elementos estão ilustrados na Figura II.12.

O MCU é um elemento utilizado para garantir conferências com mais de dois participantes.

O *gatekeeper* age como um ponto central para todas as comunicações dentro de uma determinada área, conhecido como zona H.323. e segundo Costa (2007, p.118) possui as seguintes funções:

- Tradução de endereços *aliases* para endereços IP;
- Controle de admissão de acesso em uma zona H.323;
- Gerenciamento de largura de banda;
- Roteamento de chamadas;
- Controle do número e do tipo de conexões permitidas.

O *gateway* é necessário quando terminais de padrões diferentes precisam comunicar com terminais H.323. Com este elemento é possível comunicação inclusive com a telefonia convencional. Os serviços providos pelos gateways compreendem basicamente a interoperabilidade entre padrões de áudio/vídeo e redes; conversão de protocolos; conversão de formatos de áudio e vídeo.

Qualquer equipamento que se comunicar utilizando a pilha de protocolos da arquitetura é denominado de terminal H.323.

Este padrão tem como elementos negativos a sua utilização os seguintes pontos:

- Demora no estabelecimento de Conexões
- Complexidade de implementação

Em <<http://www.openh323.org>> é encontrado o projeto OpenH323 de código aberto que permite o desenvolvimento de aplicações usando a linguagem C++. A Tabela II.4 lista exemplos de soluções H.323.

Tabela II.4- Exemplos de soluções H.323

Solução	Mais informações
NetMeeting (aplicação)	http://www.microsoft.com
GnomeMeeting (aplicação)	http://www.gnomemeeting.org
GnuGk (gatekeeper)	http://www.gnugk.org
ViewStation (câmera)	http://www.polycom.com
ViaVideo (câmera)	http://www.polycom.com

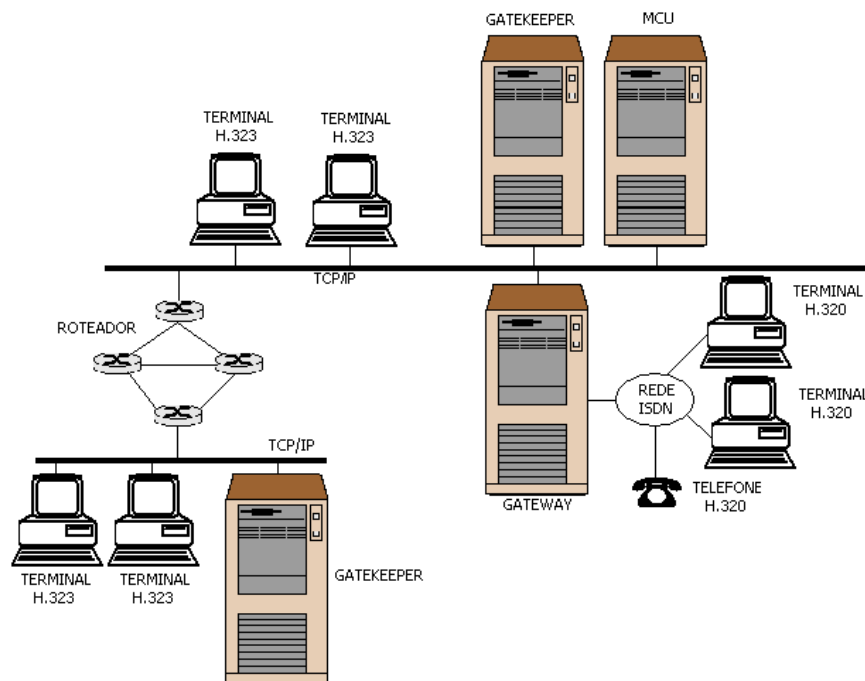


Figura II.12 – Topologia H.323 segundo Gomes (2003, p.59)

SIP

O *Session Initiation Protocol* (SIP) é um protocolo da camada de aplicação e possui especificação definida na RFC 3261. Segundo Camarillo et. al (2002), há aplicações na *Internet* que requerem a criação e a gerência do conceito de sessão, onde a sessão é o estabelecimento de uma comunicação entre participantes. A execução destas aplicações é complicada pelas práticas de locomoção entre os participantes, ou seja, é comum o usuário ter mais de um ponto de conexão com a *Internet*, tais como, PDA, *notebook* e *desktop*. Não importa em qual endereço o usuário esteja ele deve ser encontrado no caso de um convite para participar de uma sessão.

O SIP é um protocolo do controle da camada de aplicação com a função de estabelecer, modificar, e terminar sessões entre comunicações multimídia, tais como chamadas de telefone pela *Internet*. O SIP também realiza sessões *unicast* e *multicast*.

Segundo Kurose e Ross (2006, p. 472) o SIP é um protocolo que estabelece chamadas entre dois interlocutores por uma rede IP, permite o chamador avisar que está sendo convidado, permite a concordância entre a

codificação da mídia e o encerramento da chamada. Além de fazer o gerenciamento das chamadas, tais como adicionar novas correntes de mídia, mudar a codificação e convidar outros participantes.

O SIP também realiza a localização de usuário, ele considera que os host podem ser móveis, implementando um mecanismo para descobrir o endereço IP atual desses *hosts*. Então o SIP é um protocolo com funcionalidades específicas para comunicações multimídia em tempo real na *Internet*.

Segundo Costa (2007, p.128) o SIP é uma arquitetura que combina inúmeros protocolos com objetivos específicos, combinando funcionalidades capaz de oferecer serviços para aplicações multimídia. Os principais protocolos que formam a arquitetura SIP são relatados a seguir e ilustrados na Figura II.13

- SIP: *Session Initiate Protocol*
- SDP: *Session Description Protocol*
- RTP: *Real Time Protocol*
- RTCP: *Real Time Control Protocol*
- TCP: *Transmission Control Protocol*
- UDP: *User Datagram Protocol*

CONTROLE DE CONFERÊNCIAS	CODECS	RTCP
SDP	RTP	
SIP		
TCP/SCTP	UDP	
IP		

Figura II.13 – Topologia SIP

Desta forma, o termo SIP é usado para identificar a arquitetura que é formada por um conjunto de protocolos que devem operar entre si para obter um serviço de comunicação em tempo real satisfatório, contudo há um protocolo SIP. Desta forma o SIP é tanto um protocolo quanto uma arquitetura de comunicação em tempo real (COSTA, 2007, p.128).

O protocolo RTP é quase um padrão nas comunicações multimídia. O protocolo padrão na camada de transporte é o UDP ou TCP, porém não impede que o protocolo SCTP seja usado.

Falando especificamente do protocolo definido na RFC 3261, o SIP foi baseado em dois protocolos consagrados pela *Internet*, O http (*Hypertext Transfer Protocol*) e o SMTP (*Simple Mail Transfer Protocol*). Desta forma o SIP se comunica utilizando endereços URL.

Tendo como base uma estrutura alicerçada na topologia cliente-servidor, o SIP possui dois componentes: *User Agent Client (UAC)* e *User Agent Server (UAS)*. O UAC é o componente cliente que inicia às chamadas e envia as requisições e finaliza a sessão SIP. O UAC pode ser um telefone SIP, um cliente PC (*softphone*) ou até mesmo um gateway SIP. O UAS residente no servidor, é responsável pela resposta das requisições feitas pelo UAC.

Este padrão é composto de vários elementos com funcionalidades específicas que juntas formam uma solução completa para transmissão em tempo real de áudio e vídeo, esses elementos são: Terminais SIP, servidores de presença, *gateway*, *proxy*, servidor de redirecionamento e registro. As Figuras II.14 e II.15 ilustram a topologia SIP, em um mesmo domínio e em domínios diferentes respectivamente e a Tabela II.5 lista exemplos de soluções SIP.

O servidor de presença é responsável em informar se um elemento está *on-line*, para isso ele mantém o estado dos dispositivos em seu poder dos elementos que solicitaram o pedido de registro.

O *gateway* faz a compatibilização do sistema SIP com outras tecnologias.

O *proxy* faz a intermediação da comunicação das mensagens SIP, cabe aos servidores *Proxy* a tarefa de receber as requisições e enviá-las aos outros servidores. O servidor pode ser outro servidor *Proxy*, um servidor de redirecionamento, ou um UAS.

O servidor de redirecionamento é utilizado para localizar um determinado recurso, compreende uma entidade de controle de chamada que provê informação de roteamento ao UAS. Basicamente, um servidor de redirecionamento recebe as requisições e determina um servidor para encaminhamento.

O servidor de registro faz a localização lógica dos dispositivos dentro do domínio, ou seja, realiza a tradução do endereço *alias* em endereço IP, possui um papel importante na localização do usuário quanto a sua mobilidade, os servidores de registro recebem as atualizações sobre a localização corrente de cada usuário e efetuam a resolução dos nomes. Funcionam da mesma forma que os servidores DNS para uma rede *web*.

Tabela II.5- Exemplos de soluções SIP

Soluções	Mais informações
Windows Messenger (aplicação)	http://www.microsoft.com
Gizmo (aplicação)	http://www.gizmoproject.com
SIP Express Router (proxy, redirecionamento e registro)	http://www.iptel.org
OpenSER (proxy, redirecionamento e registro)	http://www.openser.org

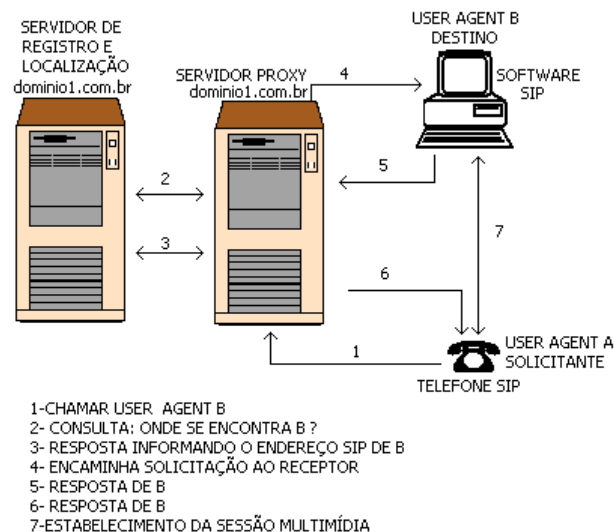


Figura II.14 – Topologia SIP em um mesmo domínio, adaptado de Ferreira *et al* (2006)

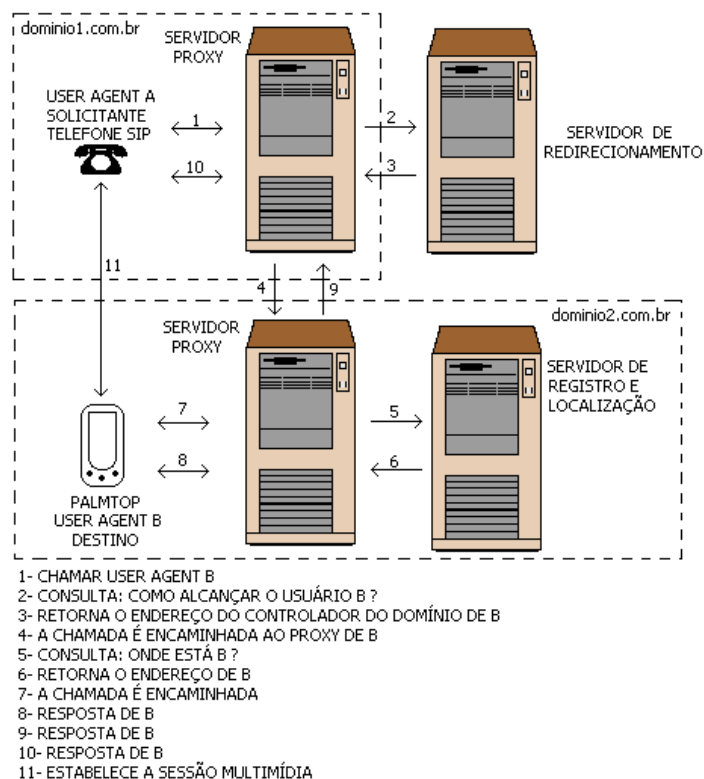


Figura II.15 – Topologia SIP em domínios diferentes, adaptado de Ferreira *et al* (2006)

JMF

A versão 2.0 do *Java Media Framework* (JMF) é resultado dos esforços das empresas Sun Microsystems, Inc. e IBM Corporation. Já o JMF 1.0 foi resultado da união entre a Sun Microsystems Inc., Intel Corporation, and Silicon Graphics, Inc. (JAVA MEDIA FRAMEWORK, 1999).

O JMF é um conjunto de classes, denominadas no jargão da informática de *API (Application Programming Interface)*, orientadas a objetos com o objetivo de serem aplicadas em sistemas que manipulam arquivos de áudio e vídeo.

Os dados multimídia são obtidos por uma variedade de fontes, tais como um microfone, uma câmera de vídeo, um sensor de ultra-som, um arquivo local ou através da *Internet*. Uma vez capturado, o mesmo é modificado através de técnicas específicas de processamento de imagem. E finalmente, está pronto para ser apresentado localmente ou remotamente através de uma comunicação pela *Internet*. A Figura II.16 mostra um processamento característico do modelo JMF.



Figura II.16 – Modelo de processamento multimídia

Esse processamento consiste de captura, processamento e saída, usados por uma aplicação *desktops* ou *applets Java*. O JMF também executa os controles básicos presentes em qualquer programa para processar arquivos multimídia que são tocar, parar, avançar e retroceder os arquivos. A Figura II.17 mostra a exposição de um vídeo proveniente de um ultra-som usando os controles nativos do JMF implementados na aplicação JMStudio.



Figura II.17 – JMF aplicado a ultra-som

O JMF apesar de ser uma framework jovem, já está sendo indicado para desenvolvimento de processos importantes, entre eles a definição dos padrões de desenvolvimento da TV Digital brasileira.

O uso de uma aplicação multimídia está condicionado a largura de banda presente na comunicação. As Tabelas II.6 e II.7 mostram os valores típicos dos codificadores e decodificadores já implementados pelo *framework JMF*.

Tabela II.6 – Valores típicos para processar vídeo em Java Media Framework (1999, p.5)

Formato	Content type	Qualidade	CPU requerida	Largura de Banda requerida
Cinepak	AVI QuickTime	Média	Baixa	Alta
MPEG-1	MPEG	Alta	Alta	Alta
H.261	AVI RTP	Baixa	Média	Média
H.263	QuickTime AVI RTP	Média	Média	Baixa
JPEG	QuickTime AVI RTP	Alta	Alta	Alta
Indeo	Quick Time AVI	Média	Média	Média

Tabela II.7 – Valores típicos para processar áudio Java Media Framework (1999, p.6)

Formato	Content type	Qualidade	CPU requerida	Largura de Banda requerida
PCM	AVI QuickTime WAV	Alta	Baixa	Alta
Mu-Law	AVI QuickTime WAV RTP	Baixa	Baixa	Alta
ADPCM (DVI, IMA4)	AVI QuickTime WAV RTP	Média	Média	Média
MPEG-1	MPEG	Alta	Alta	Alta
MPEG Layer 3	MPEG	Alta	Alta	Média
GSM	WAV RTP	Baixa	Baixa	Baixa
G.723.1	WAV RTP	Média	Média	Baixa

Uma característica importante de dados como áudio e vídeo é que eles são baseados no tempo e requerem um tempo curto e controlado para realizar a captura, processamento, entrega e apresentação. Dados que sofrem todas essas transformações, que são basicamente áudio e vídeo, possuem a denominação especial no JMF de *streaming media*.

Uma *mídia* é categorizada de acordo como o dado chega ao seu destino, recebe a denominação *pull* quando a transferência é iniciada e controlada pelo cliente, o protocolo http é um exemplo. Recebe a denominação *push* quando o servidor inicia a transferência e o controle do fluxo de dados. O protocolo RTP é um exemplo, sendo este usado pelo JMF para realizar *streaming media*, ou seja, realizar transferência de áudio e vídeo pela rede.

A arquitetura do JMF é familiar a um vídeo cassete, onde existe uma câmera para capturar o dados, uma fita que armazena o dado, um aparelho para tocá-la e um televisor com alto falante para a devida apresentação. A câmera representa o dispositivo de captura (*capture device*), a fita é a fonte de dados (*data source*), o aparelho (*player*) vai executar a mídia e finalmente a mídia é apresentada no destino final (*output devices*). A Figura II.18 ilustra a arquitetura.

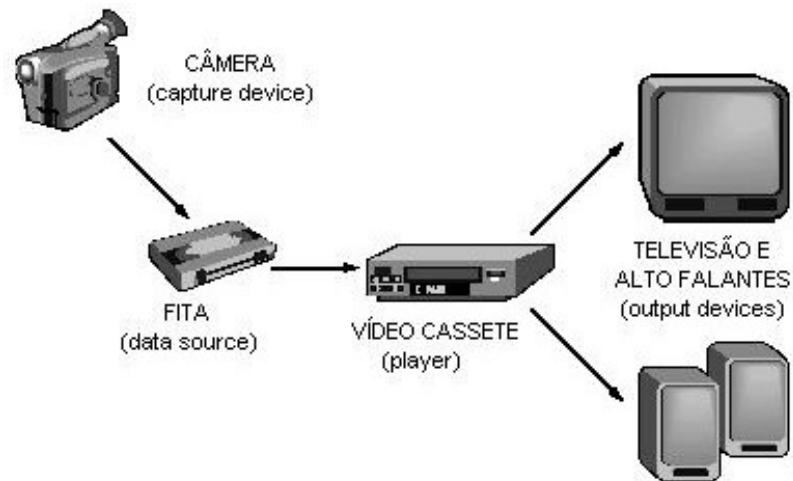


Figura II.18 – modelo JMF

JAI

O *Java Advanced Imaging* (JAI) é um conjunto de classes orientadas a objetos da SUN que fornecem suporte para um modelo avançado de manipulação de imagens.

O JAI é uma extensão da plataforma de Java para processamento de imagem com alto desempenho que podem ser incorporadas em *applet* e aplicações *desktop*. O JAI é um conjunto de classes que fornecem funcionalidades de manipulação da imagem. Implementa um conjunto de capacidades de processamento de imagem que incluem região de contorno e processamento no domínio da frequência. É altamente extensível, permitindo que as operações novas de processamento de imagem sejam adicionadas ao *framework*.

Resumindo, com o JAI, a aplicação é estendida de funcionalidades que permitem extrair informações da imagem, modificá-la geometricamente, fazer operações com os seus *pixels* e realizar compressão com os principais decodificadores atuais (Programming in Java Advanced Imaging, 1999). A Figura II.19 ilustra parte do poder de transformação alcançado por essa API.

Com essa API é possível realizar detecção de bordas, suavização, rotação, aplicar modelos matemáticos como Transformada de *Laplace* e muitos mais.

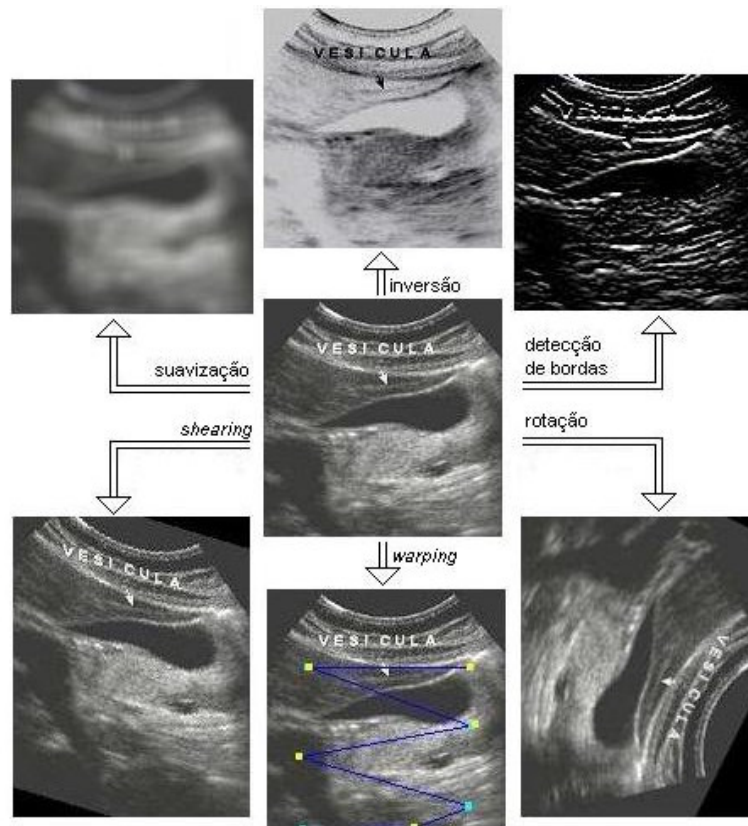


Figura II.19 – Processamento de imagem

III- MATERIAIS E MÉTODOS

Descrição Funcional

Criar um sistema que permita especialistas autorizados a realizar exames de ultra-som e disponibilizá-los para outros especialistas acompanhar o resultado de qualquer lugar que o mesmo esteja, bastando para isso uma conexão privada e segura com portas abertas para trafegar dados multimídia pela Internet. A transmissão da imagem de biomicroscopia deve atender as característica de tempo real permitindo a qualquer usuário que esteja on-line uma ação através de um chat com os outros participantes on-line. O sistema deve ser capaz de:

- Disponibilizar vídeo do procedimento *off-line*.
- Disponibilizar vídeo do procedimento *on-line*.
- Manter laudo do procedimento com a imagem gerada no sistema.
- Aplicar processamento de imagem no laudo.
- Cadastrar usuários.
- Realizar controle de acesso.
- Realizar um *chat on-line* (texto e voz).

A Tabela III.1 define o negócio, a Tabela III.2 define os usuários e a Figura III.1 define as fronteiras do sistema.

Tabela III.1 – Definição do negócio e das fronteiras do sistema

ÁREAS FUNCIONAIS	DESCRIÇÃO
Ultra-som	Manter laudo com a imagem gerada no sistema.
Especialista	Pesquisar laudo; analisar o laudo usando recursos de processamento de imagem.
Multimídia	Realizar, Gravar, abrir, transmitir e receber procedimentos.
Suporte	Mantém os usuários, mantém os endereços IP, mantém o controle de acesso no sistema.

Tabela III.2- Identificação dos usuários do sistema

ATOR	ÁREAS FUNCIONAIS
Radiologista	Ultra-som e Multimídia
Médico	Especialista e Multimídia
Administrador	Suporte

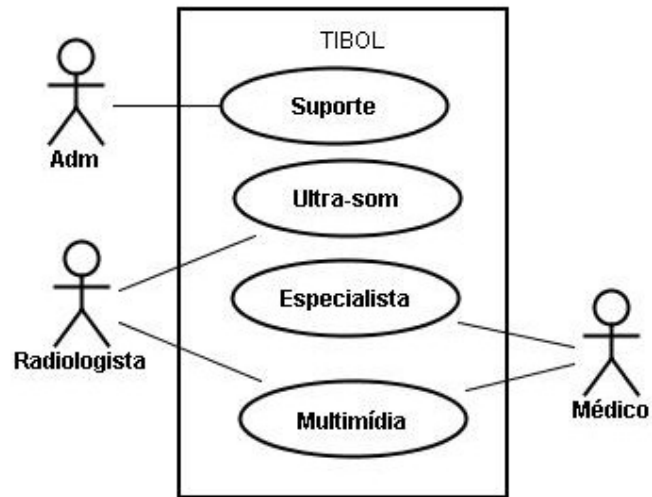


Figura III.1 – Fronteira do sistema

Concepção

Identificação dos Casos de Uso

- Manter laudo
 - Ator: Radiologista
 - Área Funcional: Ultra-som
 - Descrição: Manter laudo da biomicroscopia

- Pesquisar laudo
 - Ator: Médico
 - Área Funcional: Especialista
 - Descrição: Pesquisar o laudo da biomicroscopia

- Realizar procedimento
 - Ator: Radiologista
 - Área Funcional: Multimídia
 - Descrição: Ler a imagem da fonte geradora

- Gravar Procedimento
 - Ator: Radiologista
 - Área Funcional: Multimídia
 - Descrição: Armazenar o procedimento gerado

- Abrir procedimento local
 - Ator: Radiologista e médico
 - Área Funcional: Multimídia
 - Descrição: Visualizar um procedimento armazenado

- Transmitir Procedimento on-line
 - Ator: Radiologista
 - Área Funcional: Multimídia
 - Descrição: Transmitir o procedimento no momento que está sendo gerado

- Transmitir Procedimento off-line
 - Ator: Radiologista
 - Área Funcional: Multimídia
 - Descrição: Transmitir procedimento que já foi realizado e está armazenado na base de dados.

- Abrir procedimento à distância
 - Ator: Médico
 - Área Funcional: Multimídia
 - Descrição: Abrir uma sessão com um procedimento que esteja sendo transmitido, *on-line* ou *off-line*.

- Analisar laudo
 - Ator: Médico
 - Área Funcional: Especialista
 - Descrição: Analisar o laudo usando de processamento de imagem
- Chat escrito
 - Ator: Radiologista e médico
 - Área Funcional: Multimídia
 - Descrição: Interagir com outros usuários usando caracteres ISSO.
- Chat falado
 - Ator: Radiologista e médico
 - Área Funcional: Multimídia
 - Descrição: Permite interagir com outros usuários usando áudio
- Manter usuário
 - Ator: administrador
 - Área Funcional: Suporte
 - Descrição: Manter o cadastro dos usuários que usam o sistema.
- Manter IP
 - Ator: Administrador
 - Área Funcional: Suporte
 - Descrição: Manter atualizados os endereços IP *multicast* e *anycast*
- Controle de acesso
 - Ator: Administrador
 - Área Funcional: Suporte
 - Descrição: Atribuir papéis aos usuários do sistema
- Gerar Programação
 - Ator: Administrador
 - Área Funcional: Suporte
 - Descrição: Gerar programação dos exames, associando os participantes aos procedimentos.

Diagrama de Contexto

O Diagrama de contexto é apresentado na Figura III.2, nele são mostrados as funcionalidades (caso de uso) que os usuários (atores) estarão usando no sistema. Demonstra o que se espera do sistema.

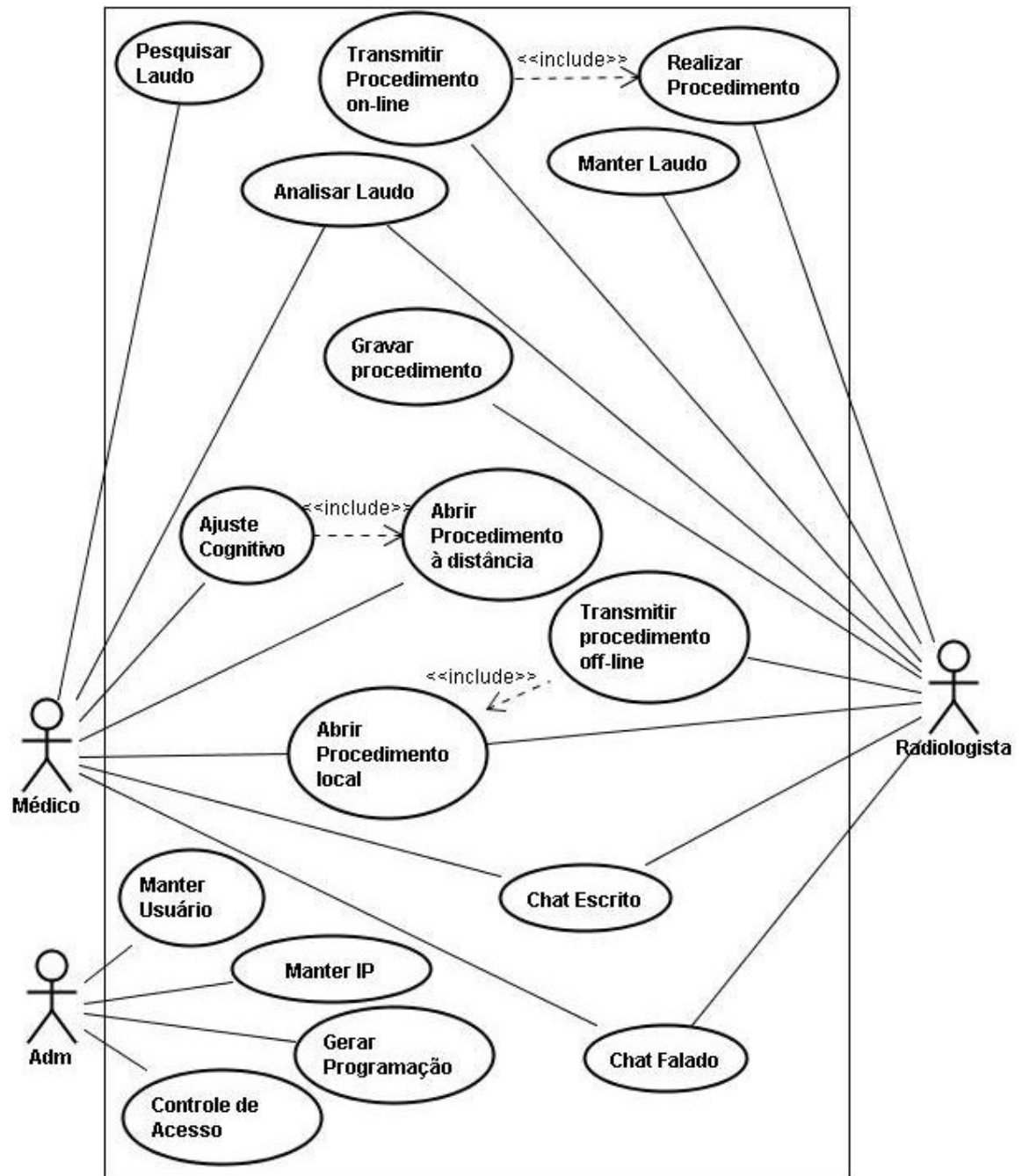


Figura III.2 – Diagrama de contexto de caso de uso

Planejamento das Iterações

A Tabela III.3 apresenta o planejamento da construção dos casos de uso. Os casos de uso que apresentam riscos que levam a atrasos no projeto serão construídos primeiro de modo a eliminar os riscos no início do projeto.

Tabela III.3 – Planejamento das iterações

FASE	NOME	DURAÇÃO
Concepção	Visão Inicial Fatores de Riscos Descrição Funcional Definição do negócio e das fronteiras do sistema Identificação dos usuários do sistema Fronteiras do sistema Identificação dos casos de uso Diagrama de contexto de caso de uso Planejamento das iterações	240 horas
Elaboração	<i>Iteração 1</i> ❖ UC1-Manter laudo ❖ UC2-Pesquisar laudo <i>Iteração 2</i> ❖ UC3-Realizar procedimento ❖ UC4-Gravar procedimento ❖ UC5-Abrir procedimento local <i>Iteração 3</i> ❖ UC6-Transmitir procedimento on-line ❖ UC7-Transmitir procedimento off-line ❖ UC8-Abrir procedimento à distância ❖ UC9-Ajuste cognitivo <i>Iteração 4</i> ❖ UC10-Analisar Laudo <i>Iteração 5</i> ❖ UC11-Chat Escrito ❖	480 horas
Construção	<i>Iteração 1</i> ❖ UC12-Manter usuário ❖ UC13-Manter IP ❖ UC14-Gerar programação ❖ UC15-Chat Falado ❖ UC16-Manter Controle de Acesso	640 horas
Transição	Implantação do projeto Ajuste de performance Teste Integrado Acompanhamento	160 horas
Total		1520 horas

Análise dos casos de uso

Caso de Uso Manter Laudo

Tabela III.4 – Manter Laudo

Nº 1	Manter Laudo	
Informações Características		
Escopo	Ultra-som	
Ator primário	Radiologista	
Objetivo do ator primário	Manter laudo da ultra-sonografia com a imagem gerada no sistema	
Descrição sucinta	O radiologista pesquisa laudos, inclui laudos novos, atualiza os dados do laudo ou exclui um laudo do sistema.	
Pré-requisito	Não há	
Pós-requisitos	Caminho principal	Os laudos são apresentados e um específico é visualizado em detalhes
	Caminho alternativo 1	Um laudo é atualizado
	Caminho alternativo 2	Um laudo é excluído
	Caminho alternativo 3	Um laudo é incluído
	Caminho alternativo 4	Laudo inexistente
Trigger Event	Radiologista solicita manter laudo	
Caminho Principal (cenário de sucesso)		
Ator		Sistema
1. Solicita uma pesquisa pelo título do laudo		
		2. Verifica se há laudos cadastrados com o respectivo título
		3. Mostra uma lista de laudos
4. Solicita visualizar o laudo específico da lista		
		5. Apresenta em detalhes o laudo
		6. Finaliza o caso de uso.
Caminho Alternativo 1		
Passo 5	O radiologista deseja atualizar um laudo já existente	
	5.1 O radiologista entra com novos dados	
	5.2 Os dados são atualizados	
	5.3 Retorna ao passo 6	
Caminho Alternativo 2		
Passo 5	O radiologista deseja excluir um laudo já existente	
	5.1 Os dados são excluídos	
	5.2 Retorna ao passo 6	
Caminho Alternativo 3		
Passo 5	O radiologista deseja incluir um laudo novo	
	5.1 O radiologista entra com os dados	
	5.2 Os dados são incluídos	
	5.3 Retorna ao passo 6	
Caminho Alternativo 4		
Passo 3	Sistema não encontrou laudos na pesquisa solicitada	
	3.1 Apresenta uma lista vazia (indicação de laudo inexistente)	
	3.2 Retorna ao passo 6	

O caso de uso Manter Laudo é descrito na Tabela III.4. O Radiologista é o responsável pela manutenção dos dados do laudo no sistema, desta forma, as funções primárias de um sistema estarão presentes nesta funcionalidade. O caminho principal apresentará um laudo já previamente gravado em detalhes, o sistema apresentará de forma alternativa a possibilidade de atualizar, excluir ou incluir um laudo novo.

O diagrama de sequência da Figura III.3 representa o caminho principal, o qual, os laudos são apresentados e um específico é visualizado pelo radiologista, isso ocorrerá quando o mesmo solicitar o método *pesquisar()* do objeto *form* da classe *FormLaudo* para realizar a tarefa, esse objeto limítrofe enviará uma mensagem de *listarLaudo(string:String)* ao objeto controlador *facade* da classe *LaudoFacade*, o objeto *facade* é o responsável pela distribuição desta tarefa, desta forma, ele criará um *Dao* correspondente a tarefa e delegará essa função ao mesmo, o dado é retornado a camada limítrofe como um *Collection*, que se encarregará de preencher o formulário com as informações solicitadas.

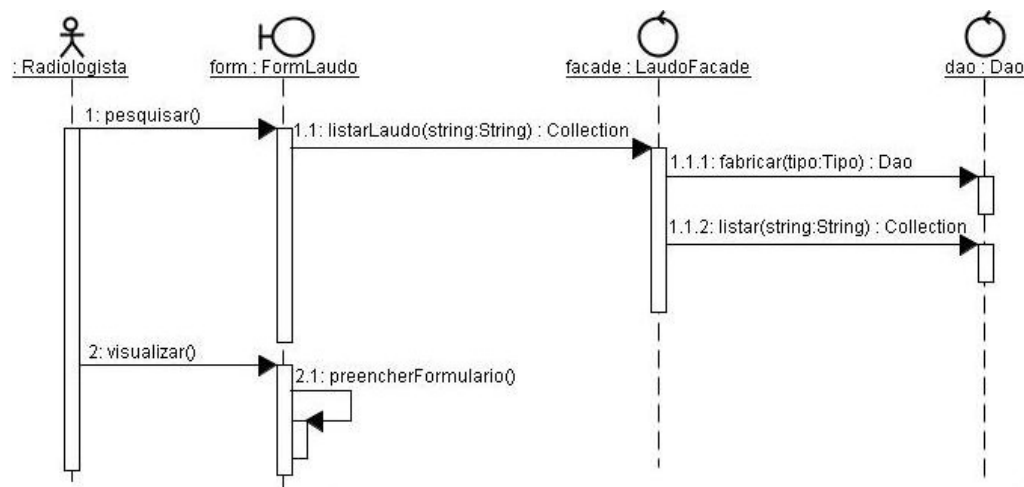


Figura III.3 – Diagrama de Seqüência: Pesquisar laudo

Caso de Uso Pesquisar Laudo

Tabela III.5 – Pesquisar Laudo

Nº 2		Pesquisar Laudo	
Informações Características			
Escopo	Especialista		
Ator primário	Médico		
Objetivo do ator primário	Verificar um laudo remotamente		
Descrição sucinta	O especialista pesquisa laudos e visualiza um laudo em detalhe		
Pré-requisito	Não há		
Pós-requisitos	Caminho principal	Os laudos são apresentados e um específico é visualizado em detalhes	
	Caminho alternativo 1	Laudo inexistente	
Trigger Event	Médico solicita pesquisar laudo		
Caminho Principal (cenário de sucesso)			
Ator		Sistema	
1. Solicita uma pesquisa pelo título do laudo		2. Verifica se há laudos cadastrados com o respectivo título	
		3. Mostra uma lista de laudos	
4. Solicita visualizar o laudo específico da lista		5. Apresenta em detalhes o laudo	
		6. Finaliza o caso de uso.	
Caminho Alternativo 1			
Passo 3	Sistema não encontrou laudos na pesquisa solicitada		
	3.1 Apresenta uma mensagem que não há laudo com a pesquisa desejada		
	3.2 Retorna ao passo 6		

O caso de uso Pesquisar Laudo é descrito na Tabela III.5. O médico usará essa função remotamente para fornecer uma segunda opinião ao laudo, essa funcionalidade permite o médico pesquisar os laudos e visualizar um em detalhe. De forma alternativa, se o laudo não for encontrado, o sistema informa que a pesquisa não pode ser realizada com os parâmetros informados.

A Figura III.4 mostra o diagrama de sequência do caminho principal onde os objetos trocam mensagens para realizar a tarefa. O médico executa o método *pesquisar()* do objeto *form* da classe *FormLaudo* na camada limítrofe, a camada limítrofe acessará o objeto *remote* que é uma implementação remota da classe *LaudoRMI* que se encontra no servidor e possui atribuições específicas para acessar o objeto controlador *facade* da classe *LaudoFacade*, o objeto *facade* é o responsável pela distribuição desta tarefa, desta forma, ele criará um *Dao*

correspondente a tarefa e delegará essa função ao mesmo, o dado é retornado a camada limítrofe como um *Collection*, que se encarregará de preencher o formulário com as informações solicitadas.

A tarefa ocorrerá remotamente pois a interface *LaudoRemote* realiza a tarefa remota com o auxílio do protocolo JRMP, ou seja, ele usa a tecnologia RMI para acessar os métodos do controlador RMI.

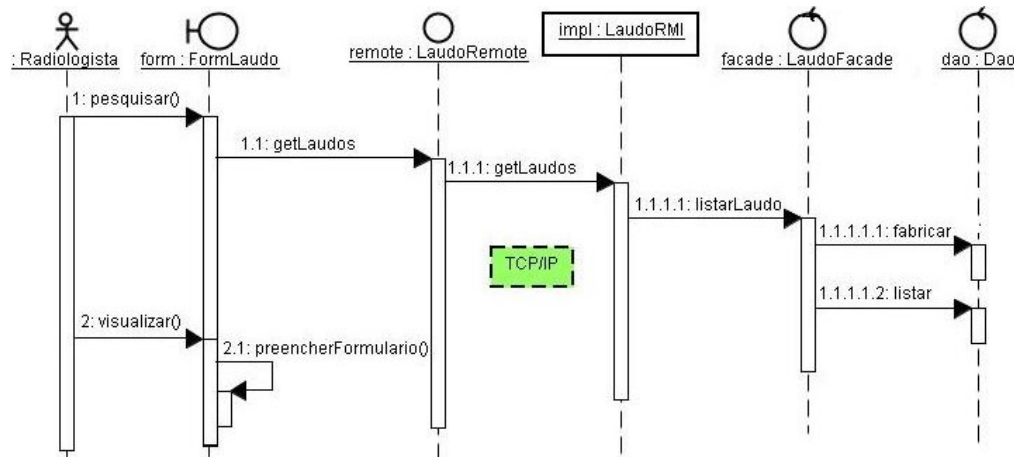


Figura III.4 – Diagrama de Sequência: Pesquisar laudo

Caso de Uso Realizar Procedimento

O caso de uso Realizar Procedimento é descrito na Tabela III.6. Essa funcionalidade vai capturar uma imagem da fonte geradora local e caso haja sucesso, um vídeo será exibido na tela. De forma alternativa, se o dispositivo de captura não for encontrado, o sistema apresentará uma mensagem informando que não foi possível realizar a captura.

A Figura III.5 mostra o diagrama de sequência onde os objetos trocam as mensagens para realizar a tarefa. O radiologista executa o método *iniciarStudio* no objeto *form* da classe *FormLaudo*, esse executará o método *createNewFrame()* do objeto *studio* da classe *JMStudio* que pertence a implementação do JMF. O próprio *studio* se encarregará de capturar o sinal do sensor após a entrada dos parâmetros que informam as características da imagem, tais como: *encoding*, *resolução* e *quadros por segundo*.

Tabela III.6 – Realizar Procedimento

Nº 3		Realizar Procedimento	
Informações Características			
Escopo	Multimídia		
Ator primário	Radiologista		
Objetivo do ator primário	Ler uma imagem da fonte geradora		
Descrição sucinta	O ator captura a imagem da fonte geradora e mostra no vídeo		
Pré-requisito	Possui um laudo		
Pós-requisitos	Caminho principal	Um vídeo é exibido no vídeo	
	Caminho alternativo 1	Laudo inexistente	
Trigger Event	Radiologista solicita realizar procedimento		
Caminho Principal (cenário de sucesso)			
Ator		Sistema	
1. Solicita capturar sensor			
		2. Abre o JMF Studio	
3. Solicita capturar dispositivo			
4. Informa as características do vídeo			
		5. Apresenta o vídeo na tela	
		6. Finaliza o caso de uso.	
Caminho Alternativo 1			
Passo 5	Dispositivo não está instalado		
	3.1 Apresenta uma mensagem que não há dispositivo disponível		
	3.2 Retorna ao passo 6		

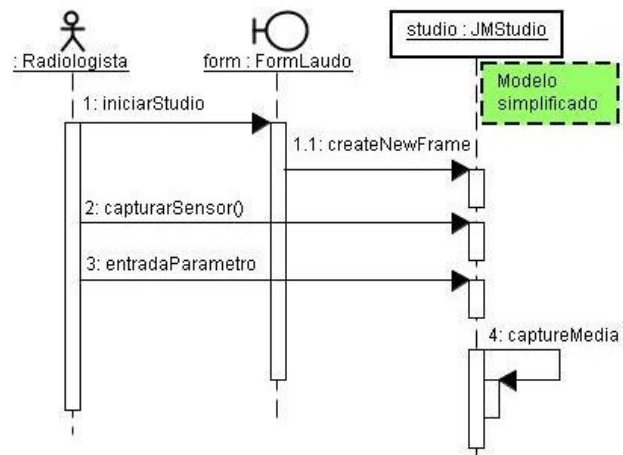


Figura III.5 – Diagrama de Seqüência: Realizar procedimento

Caso de Uso Gravar Procedimento

O caso de uso Gravar Procedimento é descrito na Tabela III.7. Essa funcionalidade permite o radiologista gravar um procedimento e associá-lo a um laudo específico. De forma alternativa, se o dispositivo de captura não estiver instalado, uma mensagem informará que não é possível capturar o procedimento e por consequência o procedimento não será gravado.

A Figura III.6 mostra o diagrama de sequência para realizar o caminho principal da tarefa. O radiologista executa o método *salvar()* do objeto *studio* da classe *JMStudio*, entra com os parâmetro característicos da funcionalidade, o *studio* salvará o procedimento no diretório informado e associará o procedimento ao laudo corrente no momento do procedimento, logo após, executará o método *atualizarTabelaVideo* do objeto *form* da classe *FormLaudo* para atualizar a tela corrente do radiologista.

Tabela III.7 – Gravar Procedimento

Nº 4		Gravar Procedimento	
Informações Características			
Escopo	Multimídia		
Ator primário	Radiologista		
Objetivo do ator primário	Gravar um procedimento associando a um laudo		
Descrição sucinta	O ator salva o vídeo em um laudo		
Pré-requisito	Studio deve estar tocando um vídeo		
Pós-requisitos	Caminho principal	Um vídeo é gravado e associado a um laudo	
	Caminho alternativo 1	Vídeo não é gravado	
Trigger Event	Radiologista solicita gravar procedimento		
Caminho Principal (cenário de sucesso)			
Ator		Sistema	
1. Solicita gravar sensor			
2. Informa as características do vídeo			
3. Solicita capturar dispositivo			
4. Informa as características do vídeo			
		5. Apresenta o vídeo na tela	
		6. Finaliza o caso de uso.	
Caminho Alternativo 1			
Passo 5	Dispositivo não está instalado		
	3.1 Apresenta uma mensagem que não há dispositivo disponível		
	3.2 Retorna ao passo 6		

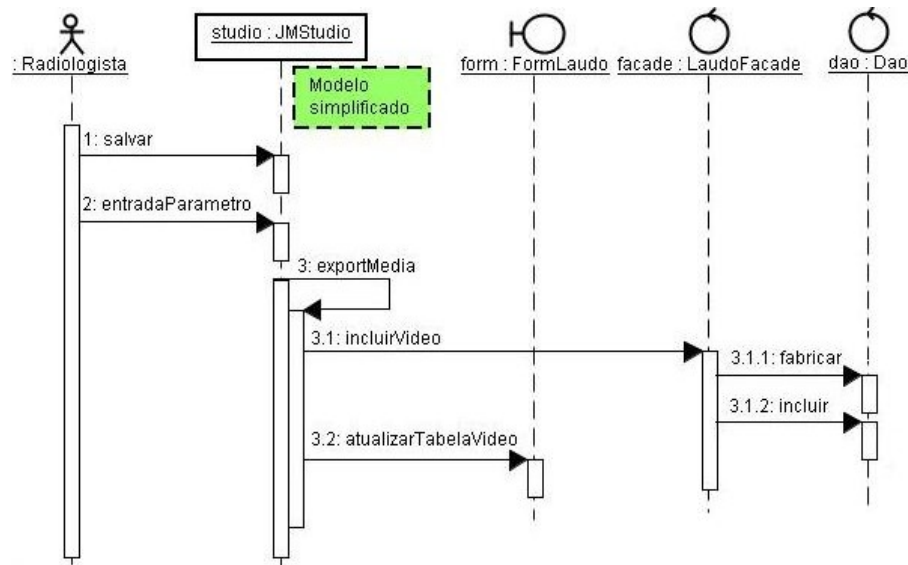


Figura III.6 – Diagrama de Seqüência: Gravar procedimento

Caso de Uso Abrir Procedimento Local

O caso de uso Abrir Procedimento Local é descrito na Tabela III.8. Ambos radiologista e médico podem usar essa funcionalidade para tocar um vídeo local. De forma alternativa, caso o arquivo esteja corrompido, uma mensagem informará que não é possível tocar o vídeo. O radiologista somente executará essa função caso haja laudo com pelo menos um vídeo gravado. Essa execução não é em tempo real, logo, clientes remotos só executaram esta função através de compartilhamento de diretórios ou via um servidor HTTP.

A Figura III.7 mostra o diagrama de sequência onde os objetos trocam as mensagens para realizar a tarefa principal que é tocar um vídeo. O radiologista executa o método *iniciarStudo(arquivo)* passando como parâmetro o nome e caminho do arquivo desejado do objeto *form* da classe *FormLaudo*, essa chamará o método *createNewFrame()* do objeto *studio* da classe *JMStudio* que pertence a aplicação de referência do JMF. O objeto *formLaudo* ainda precisa executar o método *open(nameUrl)* do objeto *studio* para iniciar o procedimento.

Tabela III.8 – Abrir Procedimento Local

Nº 5		Abrir Procedimento	
Informações Características			
Escopo	Multimídia		
Ator primário	Radiologista e Médico		
Objetivo do ator primário	Abrir um procedimento associado a um laudo		
Descrição sucinta	O ator abre um vídeo específico		
Pré-requisito	Ter laudo gravado com pelo menos um vídeo		
Pós-requisitos	Caminho principal	Um vídeo é tocado	
	Caminho alternativo 1	Vídeo não toca	
Trigger Event	Radiologista solicita abrir um vídeo		
Caminho Principal (cenário de sucesso)			
Ator		Sistema	
1. Solicita abrir um procedimento			
		2. Apresenta o vídeo na tela	
		3. Finaliza o caso de uso.	
Caminho Alternativo 1			
Passo 2	Arquivo está corrompido		
	3.1 Apresenta uma mensagem que o arquivo está indisponível		
	3.2 Retorna ao passo 3		

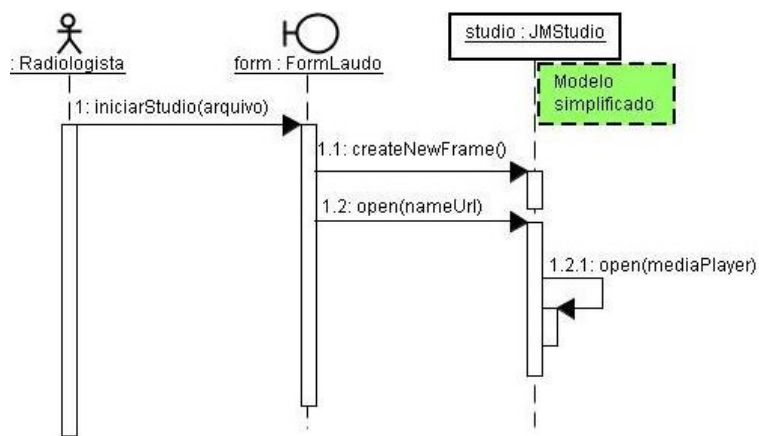


Figura III.7 – Diagrama de Seqüência: Abrir procedimento local

Caso de Uso Transmitir Procedimento On-Line

Tabela III.9 – Transmitir Procedimento On-Line

Nº 6	Transmitir Procedimento On-Line	
Informações Características		
Escopo	Multimídia	
Ator primário	Radiologista	
Objetivo do ator primário	Transmitir um procedimento on-line	
Descrição sucinta	O sistema transmite um procedimento diretamente do sensor	
Pré-requisito	Ter laudo gravado	
Pós-requisitos	Caminho principal	Um vídeo é transmitido
	Caminho alternativo 1	Vídeo não é transmitido
Trigger Event	Radiologista solicita transmitir vídeo on-line	
Caminho Principal (cenário de sucesso)		
Ator		Sistema
1. INCLUIR Realizar Procedimento		
2. Solicitar transmitir		
3. Informar parâmetros do vídeo		
4. Informa sessão de transmissão		
		5. Iniciar transmissão
6. Finalizar transmissão		
		7. Finalizar o caso de uso
Caminho Alternativo 1		
Passo 5	Dispositivo não está instalado	
	3.1 Apresenta uma mensagem que não há dispositivo disponível	
	3.2 Retorna ao passo 6	

O caso de uso Transmitir Procedimento On-Line é descrito na Tabela III.9. Essa funcionalidade é para transmitir um vídeo em tempo real diretamente do sensor, de forma alternativa, caso o sensor apresente problemas, uma mensagem informará que não será possível transmitir o procedimento.

A Figura III.8 mostra em nível simplificado o comportamento do sistema para atender o caminho principal da funcionalidade prevista. Esse caso de uso para ser realizado precisa do caso de uso *Realizar Procedimento*. Desta forma, o vídeo é transmitido para um cliente localizado na rede em uma determinada porta. O radiologista executa o método *iniciarStudio* para que uma janela possa iniciar a captura o vídeo diretamente no sensor, uma vez iniciada a captura, o radiologista solicita a transmissão em tempo real, entra com os parâmetros característico do

vídeo, informa os dados da sessão, ou seja, endereço IP e porta para iniciar a comunicação, é permitido IP *multicast* e *anycast*. Ao final, após entrar com todos parâmetros, confirma a transmissão. Todas essas funcionalidades são executadas pelo objeto *studio* da classe *JMStudio* pertencente a implementação de referência do JMF. O *studio* após iniciar a transmissão instanciará o objeto *facade* da classe *QualidadeFacade*, o qual, conhece as característica do vídeo capturado pois ele possui uma instância do objeto através da interface *QualityControl* pertencente ao JMF, que possui métodos específicos para atuar na qualidade da imagem gerada.

Portanto, A inovação proposta neste ponto, é a criação do objeto *QualidadeFacade* que ficará escutando chamadas remotas através do RMI, com a solicitação de atuar na qualidade do vídeo capturado, de modo a diminuir ou aumentar a qualidade da captura, adequando o tamanho do vídeo para as características da rede de transmissão.

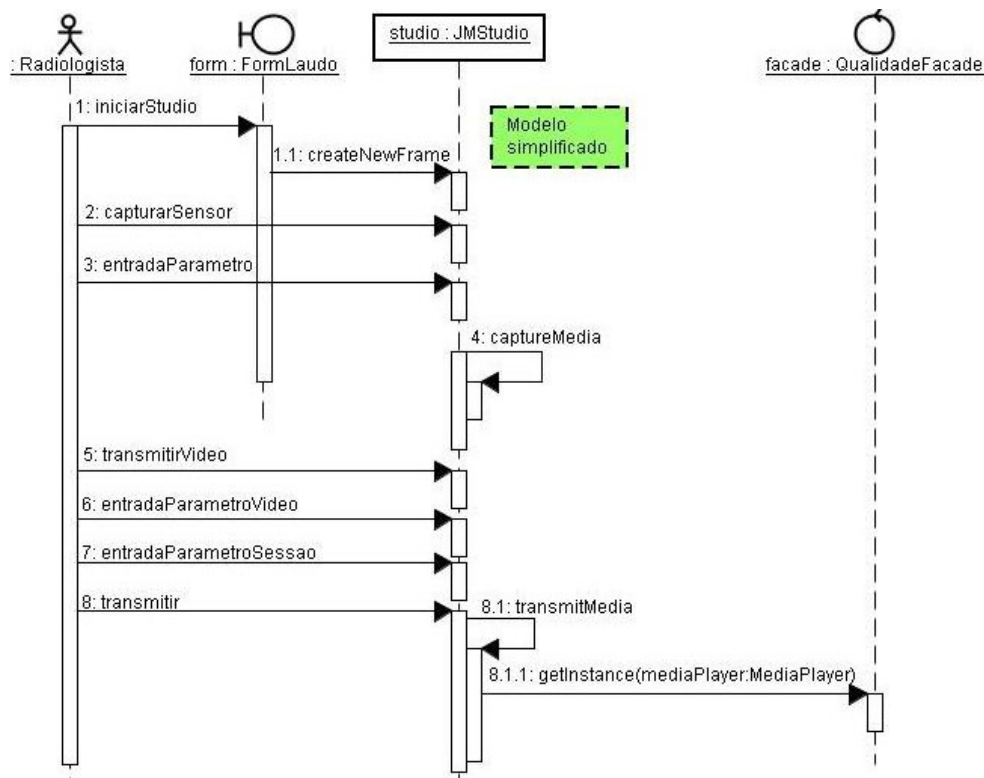


Figura III.8 – Diagrama de Seqüência: Transmitir procedimento on-line

Caso de Uso Transmitir Procedimento Off-Line

O caso de uso Transmitir Procedimento Off-Line é descrito na Tabela III.10. A Figura III.9 mostra em nível simplificado o comportamento do sistema para atender o caminho principal da funcionalidade prevista. Esse caso de uso para ser realizado precisa do caso de uso *Abrir Procedimento Local*. Desta forma, o vídeo é transmitido para um cliente localizado na rede em uma determinada porta.

A inovação proposta neste ponto, é reutilizar a rotina proposta no caso de uso *Transmitir Procedimento On-Line*, através da criação do objeto *QualidadeFacade* que ficará escutando chamadas remotas através do RMI, com a solicitação de atuar na qualidade da leitura do vídeo gravado, de modo a diminuir ou aumentar a qualidade do vídeo, adequando o seu tamanho as características da rede de transmissão.

Tabela III.10 – Transmitir Procedimento Off-Line

Nº 7		Transmitir Procedimento Off-Line	
Informações Características			
Escopo	Multimídia		
Ator primário	Radiologista		
Objetivo do ator primário	Transmitir um procedimento off-line		
Descrição sucinta	O sistema transmite um procedimento gravado em um laudo		
Pré-requisito	Ter laudo gravado		
Pós-requisitos	Caminho principal	Um vídeo é transmitido	
	Caminho alternativo 1	Vídeo não é transmitido	
Trigger Event	Radiologista solicita transmitir vídeo on-line		
Caminho Principal (cenário de sucesso)			
Ator		Sistema	
1. INCLUIR Abrir Procedimento Local			
2. Solicitar transmitir			
3. Informar parâmetros do vídeo			
4. Informa sessão de transmissão			
		5. Iniciar transmissão	
6. Finalizar transmissão			
		7. Finalizar o caso de uso	
Caminho Alternativo 1			
Passo 2	Arquivo está corrompido		
	2.1 Apresenta uma mensagem que o arquivo está indisponível		
	2.2 Retorna ao passo 7		

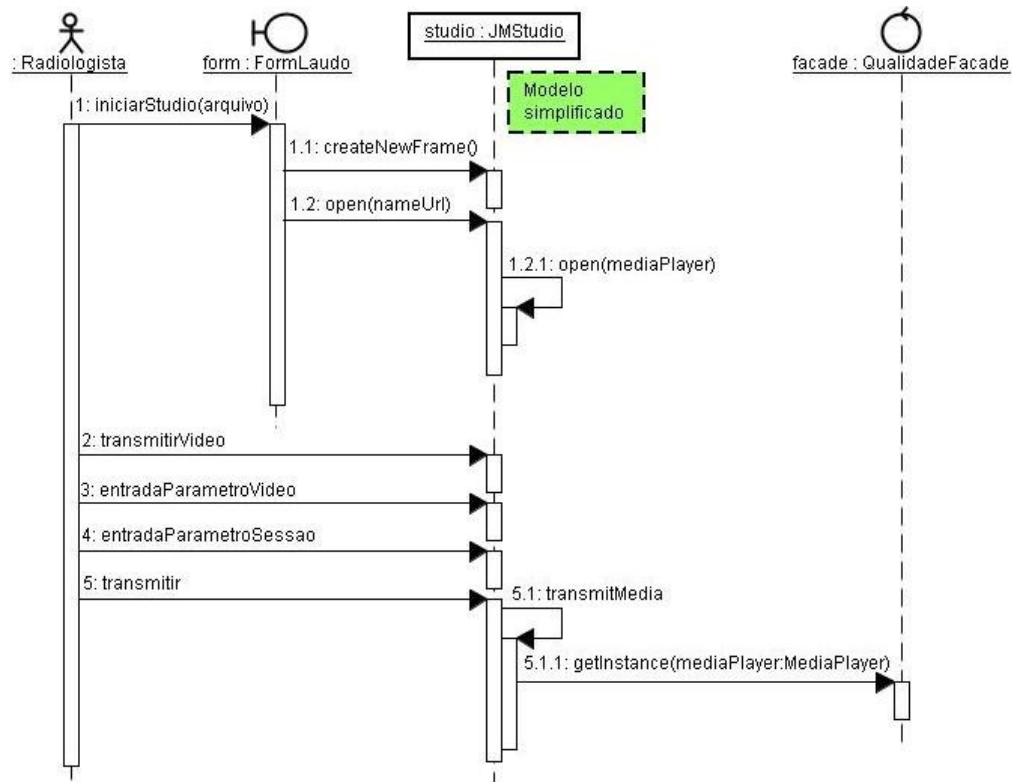


Figura III.9 – Diagrama de Seqüência: Transmitir procedimento off-line

Caso de Uso Abrir Procedimento à Distância

O caso de uso Abrir Procedimento a Distância é descrito na Tabela III.11. O médico visualizará remotamente um vídeo sendo transmitido em tempo real. De forma alternativa, caso não encontre vídeo sendo transmitido com a sessão informada, uma mensagem será exibida informando que não é possível iniciar a sessão.

A Figura III.10 mostra em nível simplificado o comportamento do sistema para atender o caminho principal da funcionalidade prevista. Esse caso de uso para ter sucesso, precisa que um vídeo esteja sendo transmitido em tempo real. O objeto *studio* da classe *JMStudio* se encarregará de iniciar a sessão, o médico precisa apenas solicitar para iniciar o processo e informar os dados da sessão.

Tabela III.11 – Abrir Procedimento à Distância

Nº 8	Abrir Procedimento a Distância	
Informações Características		
Escopo	Multimídia	
Ator primário	Médico	
Objetivo do ator primário	Abrir um vídeo remoto	
Descrição sucinta	O sistema transmite um procedimento e o médico visualiza remotamente	
Pré-requisito	Ter laudo sendo transmitido	
Pós-requisitos	Caminho principal	Um vídeo é visualizado
	Caminho alternativo 1	Vídeo não é visualizado
	Caminho alternativo 2	Vídeo não é visualizado
Trigger Event	Médico solicita abrir vídeo remoto em tempo real	
Caminho Principal (cenário de sucesso)		
Ator		Sistema
1. Solicitar abrir vídeo remoto em tempo real		
2. Informa sessão da transmissão		
		3. Inicia a recepção
4. Finaliza a recepção		
		5. Finalizar o caso de uso
Caminho Alternativo 1		
Passo 3	Inicia a recepção após um tempo	
	3.1 Aguardar até 60 segundos	
	3.2 Retorna ao passo 3	
Caminho Alternativo 2		
Passo 3	Não inicia a recepção	
	3.1 Aguardar até 60 segundos	
	3.2 Retorna ao passo 5	

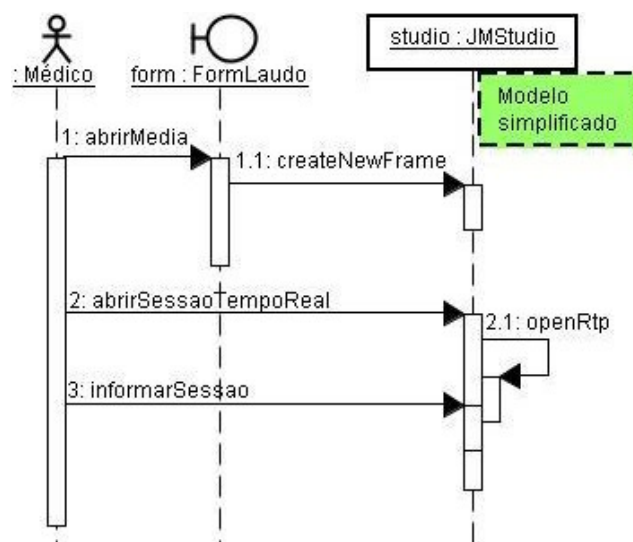


Figura III.10 – Diagrama de Seqüência: Abrir procedimento à distância

Caso de Uso Ajuste Cognitivo

O caso de uso Ajuste Cognitivo é descrito na Tabela III.12. O Médico atuará na qualidade do vídeo as condições da rede, o sistema de captura ajusta a qualidade do vídeo conforme o valor passado pelo médico que se encontra remotamente.

A Figura III.11 mostra em nível simplificado o comportamento do sistema para atender o caminho principal da funcionalidade prevista. Esse caso de uso para ser realizado precisa do caso de uso *Abrir Procedimento a Distância*. Desta forma, o vídeo é recebido por um cliente localizado na rede em uma determinada porta. Uma vez estando recebendo um vídeo em tempo real, o médico executa o método *ajusteCognitivo()* do objeto *form* da classe *FormQualidade*. A faixa de valor do ajuste varia e 0,000 até 1,000, exatamente com 3 casas decimais.

A inovação proposta neste ponto é a criação do objeto *FormQualidade* acoplado ao JMF que faz a exibição do vídeo remoto. O *FormQualidade* possui um componente visual *Slider*, que permite um ajuste que atuará remotamente através do RMI no objeto *QualidadeFacade*, que se encontra na fonte geradora do vídeo escutando chamadas remotas, com a intenção de atuar na qualidade da leitura do vídeo gravado ou na qualidade da captura do sinal, de modo a diminuir ou aumentar a qualidade do vídeo, adequando-o para as características da rede de transmissão.

O ajuste cognitivo é uma inovação pois atua no campo subjetivo da visualização da imagem pelo ser humano, a proposta é permitir que o cliente atue na qualidade da imagem que ele está vendo, partimos do pré-suposto que cada pessoa possui o seu próprio grau de cognitismo possuindo condições de atuar na qualidade da imagem em conjunto com modelos matemáticos com essa função.

Cada *encoding* utilizado obteve resultados diferentes, por exemplo, o *encoding JPEG-RTP* obteve resultado insatisfatório para um ajuste próximo do mínimo quando a imagem apresenta letras, já o *encoding H.263* a imagem praticamente não sofreu alterações. O uso correto do *encoding* está associado a largura de banda disponível, sendo o *JPEG-RTP* indicado para largura de banda alta e o *H.263* indicado no caso de largura baixa.

Tabela III.12 – Ajuste Cognitivo

Nº 9		Ajuste Cognitivo	
Informações Características			
Escopo	Multimídia		
Ator primário	Médico		
Objetivo do ator primário	Adaptar a qualidade do vídeo as condições da rede		
Descrição sucinta	O sistema de captura ajusta a qualidade do vídeo conforme valor passado pelo cliente remoto.		
Pré-requisito	Vídeo sendo transmitido		
Pós-requisitos	Caminho principal	Qualidade é adaptada	
Trigger Event	Médico ajusta o potenciômetro cognitivo		
Caminho Principal (cenário de sucesso)			
Ator		Sistema	
1. INCLUIR <i>Abrir Procedimento à distância</i>			
2. Ajustar potenciômetro cognitivo			
		3. Solicita ao controlador de qualidade remoto o devido ajuste	
		4. Imagem é degradada ou melhorada	
		5. Finaliza o caso de uso	

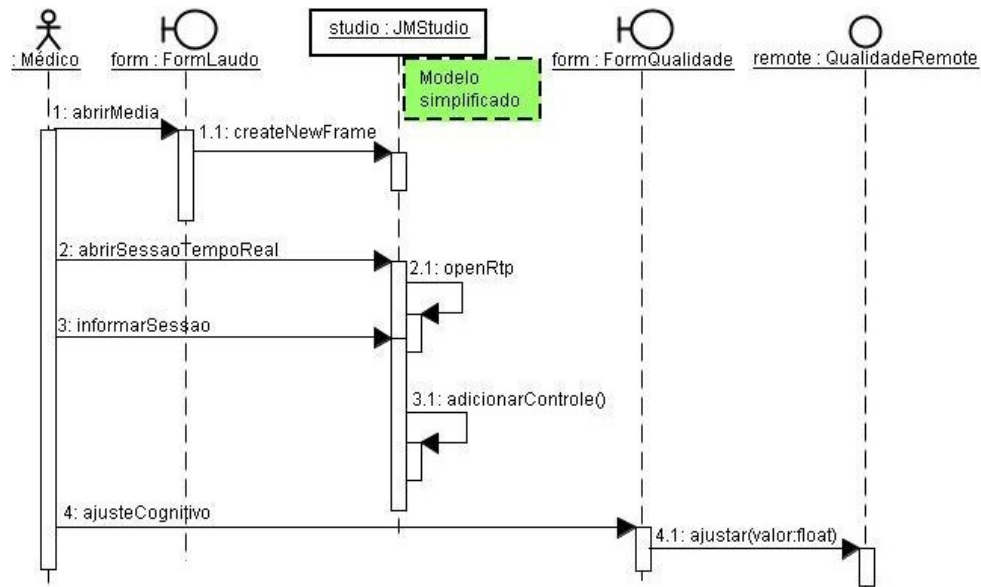


Figura III.11 – Diagrama de Seqüência: Ajuste cognitivo

Caso de Uso Analisar Laudo

O caso de uso Analisar Laudo é descrito na Tabela III.13. Através de recursos de processamento de imagens, onde é possível manipular os *pixels* das imagens, ambos, médicos e radiologistas possuirão uma funcionalidade para realizar análise na imagem. O sistema permite a análise local ou remota, no caso da remota, o cliente (médico) tem a opção de permitir que o servidor (radiologista) acompanhe em tempo real todas manipulações realizadas.

A Figura III.12 mostra em nível simplificado o comportamento do sistema para atender o caminho principal da funcionalidade prevista. Esse caso de uso para ter sucesso, precisa de um laudo para análise.

A inovação proposta neste ponto é o compartilhamento da mesma análise entre os computadores cliente e servidor. Ou seja, uma determinada transformação no *pixel* da imagem feita pelo médico remoto, é visualizada em tempo real pelo radiologista. Essa função ajudará o radiologista a focar o exame em partes específicas solicitadas pelo médico remoto.

Tabela III.13 – Analisar Laudo

Nº 10	Analisar Laudo	
Informações Características		
Escopo	Especialista	
Ator primário	Médico e Radiologista	
Objetivo do ator primário	Analisar o laudo da biomicroscopia usando recursos de processamento de imagem	
Descrição sucinta	Aplicar manipulações na imagem para extrair informações não visíveis na imagem original.	
Pré-requisito	Possuir laudo	
Pós-requisitos	Caminho principal	Imagem é manipulada e apresentada
	Caminho alternativo 1	Imagem não suporta manipulação
Trigger Event	Ator solicitar analisar uma imagem	
Caminho Principal (cenário de sucesso)		
Ator		Sistema
1. Solicita analisar uma imagem		
2. Solicita uma manipulação		
3. Ajusta os valores		
		4. Imagem é manipulada
		5. Finaliza o caso de uso
Caminho Alternativo 1		
Passo 4	Imagem não suporta manipulação	
	4.1 Apresentar mensagem	
	4.2 Retorna ao passo 5	

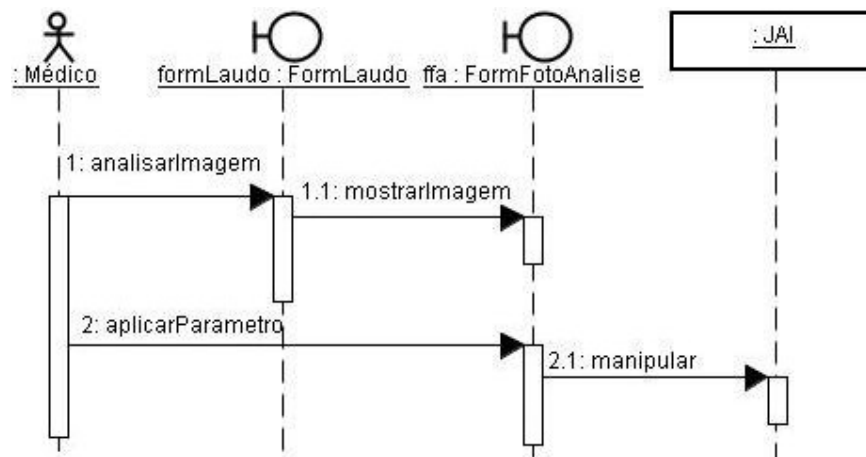


Figura III.12 – Diagrama de Seqüência: Analisar Imagem

Caso de Uso Chat

O caso de uso Chat é descrito na Tabela III.14. Em Deitel e Deitel (2005, pg. 867) é apresentado o núcleo de um sistema bate-papo *on-line* através de textos. Esse núcleo permite clientes se conectarem ao servidor, onde esse tem a função de rotear as mensagens para o endereço *multicast* já configurado.

O núcleo do *bate-papo* é composto de um endereço para datagramas de *multicast*, uma porta para ouvir os datagramas *multicast*, uma porta para enviar datagramas de *multicast* e uma porta para conexões de *Socket* com o núcleo do sistema. As informações que representam as sessões de comunicação serão transparentes para o usuário pela implementação do caso de uso *Manter IP*.

O sistema envia dados pela rede usando conexão de rede baseada em datagramas utilizando UDP e graças ao *multicast*, os dados são enviados a múltiplos clientes com um único comando.

O JMF é a solução utilizada para atuar no *chat on-line* através de áudio. Então, para realizar a funcionalidade do áudio no *chat*, o sistema será dotado das funcionalidades do JMF sem alterações significativas, visto que esse trabalho se concentra em melhorias específicas na qualidade da imagem para melhorar laudos em telemedicina.

Tabela III.14 – Chat

Nº 11		Chat	
Informações Características			
Escopo	Multimídia		
Ator primário	Médico e Radiologista		
Objetivo do ator primário	Realizar contato entre os participantes através de texto (bate-papo)		
Descrição sucinta	Usuário se conecta na sala de bate-papo, fornece uma identificação e troca mensagens		
Pré-requisito	Não há		
Pós-requisitos	Caminho principal	Ocorre uma troca de mensagem	
Trigger Event	Ator solicitar um chat		
Caminho Principal (cenário de sucesso)			
Ator		Sistema	
1. Solicita chat			
2. Fornece uma identificação			
		3. Sistema o adiciona na sala	
4. Escreve e envia uma mensagem			
		5. Mensagem é enviada aos participantes da sala	

Diagrama de classe de domínio

O diagrama apresentado na Figura III.13 tem a função de mostrar como as classes se relacionam para proporcionar as funcionalidades exigidas pelos casos de usos.

As classes com esteriótipo *boundary* farão a interface com os usuários do sistema, as classes com esteriótipo *entity* realizam as operações para propiciar as funcionalidades das regras de negócio, as classes com esteriótipo *control* controlaram o fluxo das mensagens enviadas pelos usuários para as classes *entities* realizar o trabalho.

O sistema armazenará os vídeos em diretórios, onde a classe *Video* armazenará o caminho do diretório no atributo *url*. Os vídeos armazenados no sistema estará relacionado a um *Laudo*.

O sistema armazenará as fotos em banco de dados relacional, onde a classe *Foto* armazenará os *bytes* da mesma no atributo *imagem[]*. As fotos armazenadas no sistema estarão relacionadas a um *Laudo*.

As funcionalidades relacionadas a multimídia serão realizadas através de comunicação direta com os frameworks JMF e JAI acoplados ao sistema.

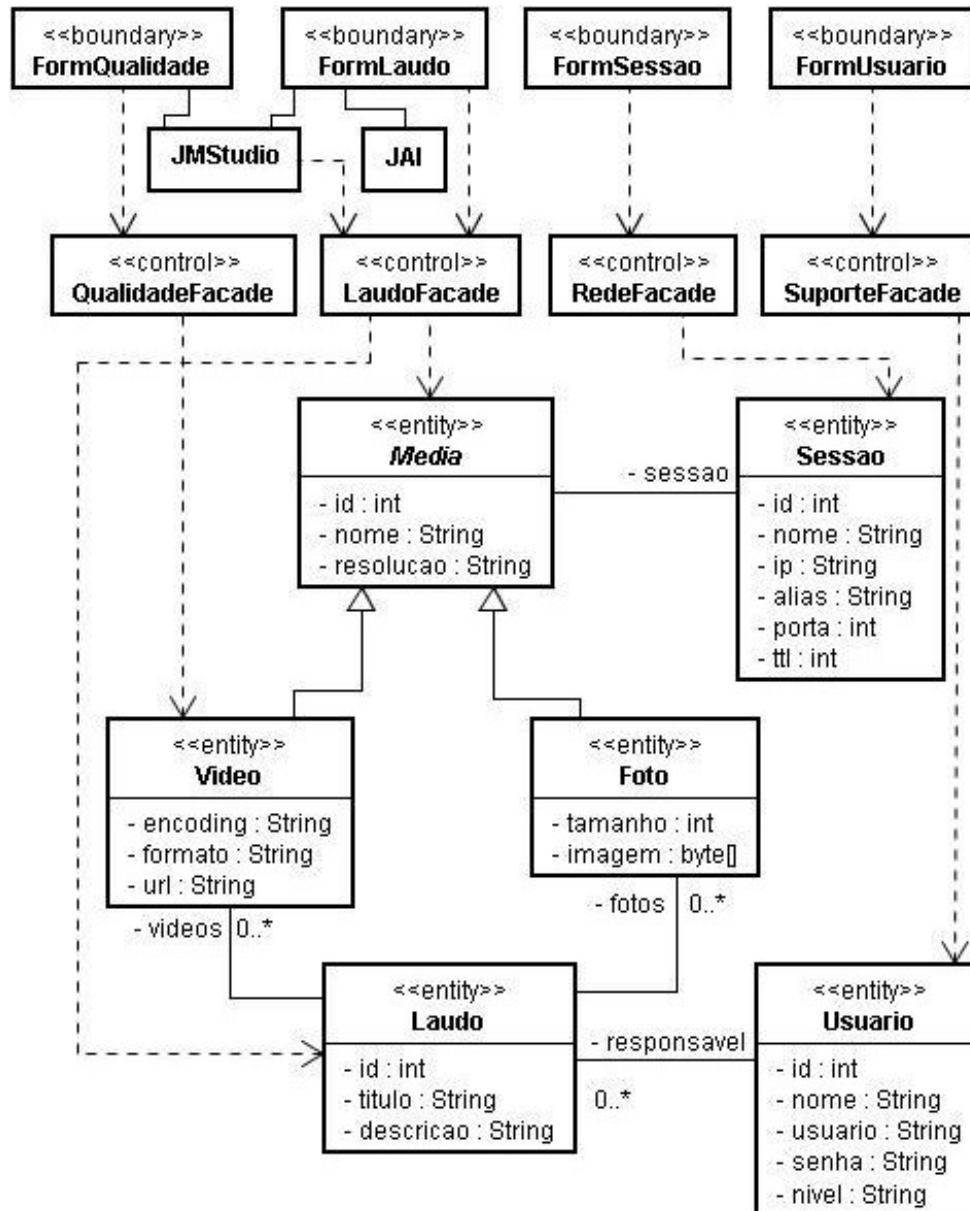


Figura III.13 – Diagrama de classe domínio

Arquitetura Proposta

Uma arquitetura deve ser projetada para atender todas as necessidades da aplicação. A arquitetura deve ser capaz de:

- Permitir o servidor capturar áudio e vídeo dos sensores.
- Permitir o servidor capturar vídeo do sistema de arquivo.
- Permitir o servidor realizar transmissão de vídeo em tempo real.
- Permitir clientes se conectarem as sessões de tempo real.
- Permitir consulta dos laudos usando um navegador padrão da Internet ou uma aplicação *desktop*.
- Permitir rodar uma aplicação de *chat*
- Realizar adaptação na taxa de transmissão para melhorar a qualidade de apresentação
- Realizar adaptação no retardo para iniciar a recepção para melhorar a qualidade de apresentação

Arquitetura Técnica

A Figura III.14 apresenta a arquitetura técnica mostrando a localização de cada software no sistema. A Figura III.15 apresenta os protocolos envolvidos na arquitetura para atender todos os requisitos necessários para aplicação *multimídia*. A linha tracejada representa a plataforma para atendimento *web*, ou seja, permitirá o uso do browser para consultas de laudo e visualização de procedimentos *off-line*.

A escalabilidade representa para um sistema, a capacidade de suportar mais usuários com o tempo sem a necessidade de alterações no software, somente no hardware. Portanto, é necessária uma arquitetura robusta que use partes já consagradas por outros pesquisadores. Visto que, devido aos vários tipos de dados apresentados em um sistema multimídia, vários protocolos são usados para suportar o tráfego dos dados sem interferência, e em certos casos ocorre à necessidade de sincronismo entre os mesmo.

Quando o áudio e transmitido com o vídeo são necessários sincronismos entre eles. Isso é possível com o protocolo RTP. O usuário precisa receber o laudo de um paciente com os dados confiáveis, isso é possível como o RMI.

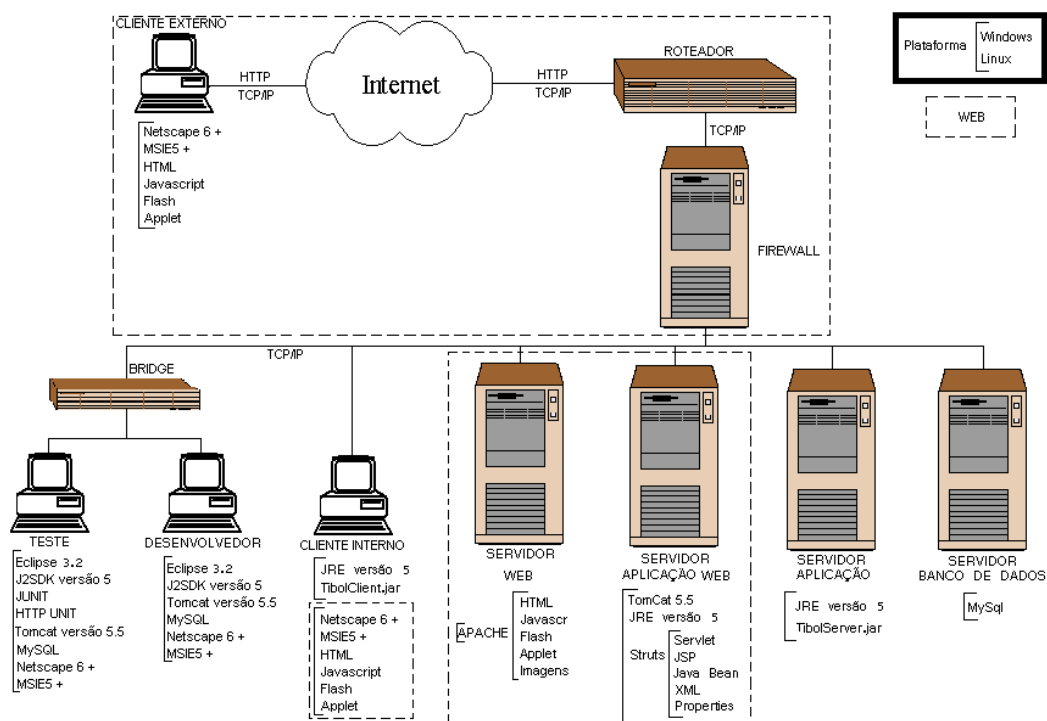


Figura III.14 – Arquitetura técnica

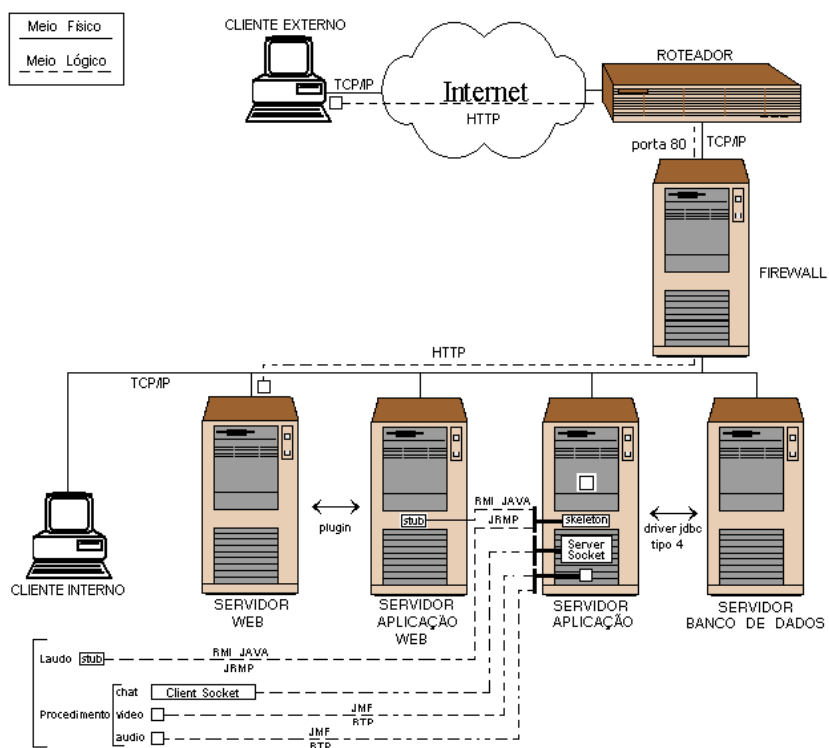


Figura III.15 – Protocolos

O cliente externo usará a aplicação usando um navegador. Esse cliente não terá acesso ao procedimento em tempo real, pois o protocolo http não é apropriado para esse tipo de aplicação. O mesmo terá acesso aos laudos e ao procedimento gravado sob demanda.

O Cliente Interno usará a aplicação através do módulo *TibolClient.jar*. Trata-se da camada limítrofe entre o cliente e o sistema. Toda requisição deste módulo é realizada remotamente. O laudo é acessado no servidor usando RMI/JRMP na camada de aplicação e TCP na camada de transporte.

Na realização do procedimento, serão três aplicações distintas cooperando de forma sincronizada para realização do mesmo.

O *chat* é uma aplicação cliente desenvolvida com *Socket* através do protocolo UDP da camada de transporte através de *datagrama multicast*.

Ambos, vídeo e áudio, são extensões do *framework Java Media Framework* usando o protocolo RTP da camada de aplicação com o protocolo UDP da camada de transporte. Aplicações *unicast* e *multicast* são aceitas para transmissão do procedimento.

O Servidor WEB tem a função de fornecer aplicações que rodam no navegador da *Internet*. Essas aplicações são desenvolvidas na linguagem HTML e funcionam remotamente sobre o protocolo http. Os teste mostraram que o protocolo http não é indicado para transportar áudio e vídeo sob demanda. Logo, as aplicações construídas para rodar no navegador serão limitadas a conteúdo institucional e acesso aos laudos dos procedimentos previamente gravados. Essa aplicação é útil para os usuários fora da rede de acesso que funcionará o sistema *Tibol*. Assim nenhuma porta de comunicação será necessária pois o *port* de comunicação para aplicações http número 80 (oitenta) fica normalmente disponível para essas aplicações.

O Servidor de Aplicação *web* trabalha em conjunto com o servidor *web*. Tem a função de realizar o processamento dinâmico para os usuários da *Internet*. Apesar do *framework Struts* ser utilizado para realizar esta tarefa, uma customização da tecnologia JSP/Servlet será utilizada por ser considerada de simples construção não agregando mais uma complexidade para o sistema.

A aplicação *web* acessará o servidor de aplicação através do protocolo do RMI/JRMP na camada de aplicação e TCP na camada de transporte. Basicamente para consultar o resultado do laudo.

O Servidor de aplicação é composto da aplicação *TibolServer.jar*. Esta aplicação é o núcleo do funcionamento do sistema *Tibol*. É responsável pela execução remota de todos os módulos do sistema, inclusive o servidor de *chat* e servidor de JMF. O servidor acessará a base de dados através do *driver JDBC* tipo 4.

O Servidor de banco de dados é onde os dados serão gravados. O vídeo será gravado no próprio sistema de arquivos e os demais dados serão gravados em um banco de dados relacional.

Quando a aplicação for instalada no modo maleta, conforme Figura III.16, será um simples *notebook* acoplado a um sensor para capturar a imagem, resumindo-se a um cliente e um servidor fornecendo as funcionalidades de servidor da aplicação e servidor de banco de dados.



Figura III.16 – Modelo Maleta

A exigência por uma arquitetura rodando somente no navegador é questionada por questões de segurança da informação dentro da instituição. O sistema *Tibol* exige um conjunto de *port* abertos para o seu pleno funcionamento. O protocolo RMI exige *ports* dinâmicos para garantir performance. Atualmente, é inadmissível qualquer aplicação rodando em uma rede pública sem a proteção de um *firewall* e este dispositivo é um problema para aplicações distribuídas usando o conceito de *port* exigidos pelos protocolos e *frameworks* que rodam no TCP/IP.

Arquitetura Aplicação

A arquitetura da aplicação deve ser projetada de modo que determinadas situações típicas não se tornem problemas que impedirão a aplicação de atender aos requisitos do sistema. São consideradas as seguintes situações como críticas:

- Sistemas envolvendo interfaces com usuário, servidor de aplicação remoto e banco de dados.
- Em aplicações distribuídas.
- Em aplicações que acessam dados em mais de uma fonte de dados.

Sistemas envolvendo interfaces com usuário, servidor de aplicação remoto e banco de dados

- O sistema tem que aceitar dados do usuário, atualizar o banco de dados, e retornar o dado para o usuário em um outro cenário mais tarde.
- Há várias formas em que o dado pode ser aceito e apresentado no sistema do usuário.
- Dados que alimentam um sistema em um formulário deverão ser restauráveis em outro formulário.

Se o sistema desenvolve um simples componente que interage com o usuário e também mantém o banco de dados, então um requerimento para suportar um novo tipo de apresentação necessitará o reprojeto do componente.

A proposta é que o hospital (ou similar) também forneça a facilidade de acompanhamento dos resultados dos exames. Quando o usuário se autenticar no sistema, se ele for o paciente, ele verificará o resultado de um determinado exame; se ele for o administrador, verificará a quantidade de exames realizados em um período de tempo; se ele for o médico, dará uma segunda opinião no resultado do exame previamente realizado. Ou seja, a mesma informação pode ser visualizada de várias formas de acordo com o perfil do solicitante. Aqui, o mesmo dado que representa o laudo é visto em múltiplas formas, mas é controlado por uma única entidade na aplicação. Assim, três tarefas ocorrem ao mesmo tempo na aplicação:

- Gerenciar a interação do usuário com o sistema
- Gerenciar o dado atual
- Formatar o dado em múltiplas formas

Dessa forma, um simples componente que faz todas tarefas, pode ser dividido dentro de três componentes independentes.

Todas três tarefas podem ser tratadas por diferentes componentes. Então a solução segundo Deshmukh e Malavia (2003, p.359) é separar os dados da apresentação dos dados de manutenção e ter um terceiro componente coordenador. Esses três componentes são chamados “Modelo”, “Visão” e “Controlador”. eles formam o básico do padrão MVC. A Figura III.17 mostra o relacionamento entre os componentes do modelo MVC.

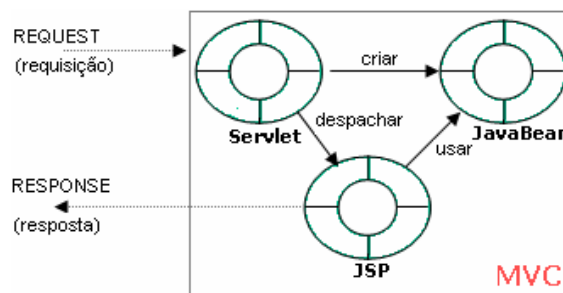


Figura III .17– Modelo – Visão - Controlador

O modelo é responsável por manter o dado ou o estado da aplicação. Ele também gerencia o armazenamento e recuperação do dado armazenado. o componente *JavaBean* em conjunto com o *framework Hibernate* foi aplicado nessa camada.

A visão contém a lógica de apresentação. Ela mostra o dado contido no modelo para o usuário. Ele também permite o usuário interagir com o sistema e notificar o controlador às ações do usuário. Os componentes *web JavaServer Page (JSP)* e o *JSP Standard Tag Library (JSTL)* foram aplicados na construção desta camada.

O Controlador gerencia o modelo e a visão. Ele instancia e associa o modelo e a visão. Dependendo do requerimento da aplicação, ele pode instanciar múltiplas visões e pode associá-las com o mesmo modelo. Ele escuta as ações do

usuário e manipula o modelo de acordo com a regra de negócio. Um *Servlet* foi aplicado nessa camada.

Desenvolver uma aplicação seguindo o padrão *model-view-controller* permite criar múltiplas visões para um mesmo dado. Mudanças ocorrem nos componentes do modelo ou nos componentes de visão de forma independente. A visão permanece a mesma. Isso aumenta a manutenibilidade e a extensabilidade do sistema.

Em aplicações distribuídas

Os componentes do cliente e os componentes do servidor residem em locais remotos e a comunicação ocorre sobre a rede de comunicação. Sendo assim:

- O banco de dados fica no lado do servidor
- O servidor fornece métodos *get's* (*getImage[]()*, por exemplo) para que o mesmo possa chamá-lo um por um para reaver os dados.
- O servidor fornece métodos *set's* (*setImage(byte image[])* por exemplo) para o cliente, para que o mesmo possa chamá-lo um por um para atualizar os dados no banco de dados.

Cada chamada entre o cliente e o servidor é uma chamada de métodos remotos com uma substancial sobrecarga na rede. Se a aplicação cliente chama os métodos *get's* e *set's* para reaver ou atualizar simples valores de atributos, ocorrerá muitas chamadas remotas para esse atributo. Essa chamada remota gera muita sobrecarga na rede e conseqüentemente uma diminuição da performance do sistema.

A camada de negócio acessará o banco de dados diretamente ou via a camada de recursos, e colocará o mecanismo de acesso dentro de um conjunto de entidades e *beans* controladores. Esses componentes expõem o dado via interfaces remotas. As aplicações *desktop's*, os *servlets* e páginas *JSP* na camada de apresentação precisam acessar esses dados de negócio, eles fazem isso chamando os métodos das interfaces remotas implementadas por *javabeans*.

Para exemplificar, vamos considerar que desejamos incluir dados de uma paciente na base de dados e se trata de uma aplicação distribuída. Os dados são: identificador, nome e imagem do ultra-som. O qual está encapsulado em um

javabeans chamado *Laudos*. O acesso para a informação está encapsulado pela aplicação da camada de negócio com a ajuda de um *javabeans* controlador chamado *LaudosFacade*. Esse controlador expõe métodos para o cliente remoto, tais como: *getId()*, *setId(...)*, *getNome()*, *setNome(...)*, *getImagem()*, *setImagem(...)*. Os servlets e páginas JSP então tem que chamar cada método um por um remotamente no servidor. Conforme a Figura III.18.

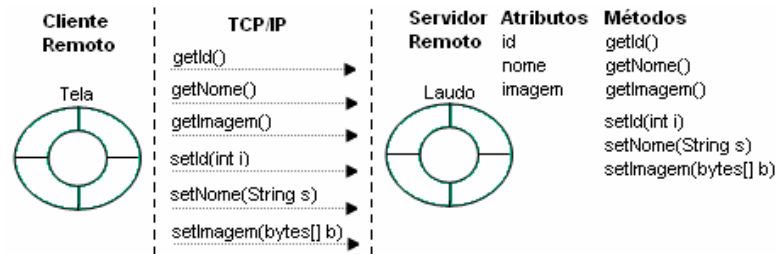


Figura III.18 – Sobrecarga de chamadas em objetos remotos

Na Figura III.18 alguns problemas são identificados:

- Um simples objeto de negócio tem muitos atributos
- Na maioria das vezes, o cliente requer vários atributos ao mesmo tempo e quase nunca apenas um atributo
- A utilização dos métodos *get's* é muito maior que os métodos *set's*

Desta forma, criar um objeto para encapsular todos os atributos que são requeridos pela aplicação cliente. Esse objeto é chamado de *Value Object*. Quando o cliente requer o dado do servidor, o componente do lado do servidor, extrai os valores localmente e constrói o *Value Object*. Esse *Value Object* é então enviado para o cliente por valor e não por referência, o que significa que esse objeto é serializado e transferido pela rede *bit a bit*.

O cliente do outro lado reconstrói o objeto localmente com todos os valores intactos. Uma vez que o objeto está local, o cliente consulta todos os atributos sem gerar sobrecarga na rede.

Agora, considerando o exemplo anterior, ao invés de fazer múltiplas chamadas remotas no objeto *Laudos* para reaver todos atributos para formar um laudo, o cliente chama um simples método *getLaudos* no *LaudosFacade*, este vai extrair os dados do objeto *Laudos*, criar o *LaudosVO* e retorná-lo pela rede com

todos os atributos estruturados em um objeto próprio para trafegar pela rede. Assim, o cliente acessará todas informações em uma única vez localmente. A Figura III.19 está ilustrando a solução.

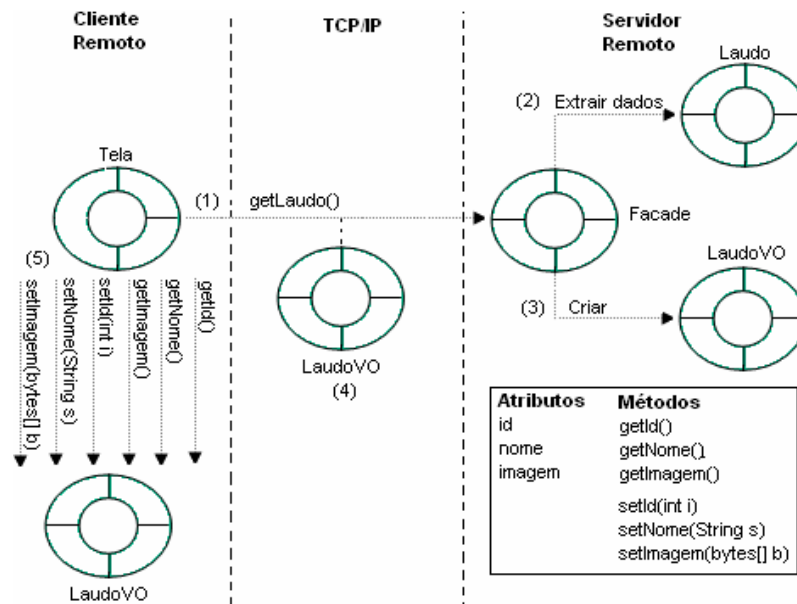


Figura III.19 – Aplicação dos padrões Facade e Value Object

O cliente pode ser uma página web JSP, um servlet, uma applet Java, um JPanel no formato desktop que faz chamadas remotas para o objeto de negócio na camada de negócio no servidor. O objeto de negócio é um componente que cria o *Value Object*. O cliente sempre fará chamadas para o objeto de negócios.

A interface remota é simplificada porque múltiplos métodos retornando um simples valor são simplificados dentro de um simples método retornando um grupo de múltiplos valores. A redução do número de chamadas através da rede melhora o tempo de resposta da aplicação. Se o cliente deseja atualizar o valor do atributo ele primeiro atualiza o valor no *Value Object* local e então o envia para o servidor para que possa ser persistido o novo valor. Tudo isso acontece usando o mecanismo de passagem por valor.

O *Value Object* pode ficar desatualizado, isto é, se o cliente adquiriu um *Value Object* por um longo período de tempo, há possibilidade da informação ter sido atualizada por outro cliente. No caso de *Value Object* mutável, requisições de atualização de dois ou mais clientes podem resultar em conflito de dados.

Resumindo, um *Value Object* é um pequeno objeto Java serializado que é usado para conduzir grupo de dados sobre a rede de um componente residente em outra camada de uma aplicação multicamadas distribuídas. O seu principal propósito é reduzir sobrecarga de comunicação reduzindo o número de chamadas remotas entre os componentes distribuídos.

Em aplicações que acessam dados em mais de uma fonte de dados

As aplicações envolvem atualizações e consultas de fontes de dados diferentes: banco de dados relacionais, sistemas de arquivos, arquivos *XML*, assim por diante. As seguintes situações ocorrem:

- As aplicações envolvem atualizações e consultas de fontes de dados iguais, como um banco de dados, mas bancos de dados são fornecidos por diferentes fabricantes, por exemplo Oracle, SQLServer da Microsoft, DB2 da IBM e outros, onde cada fabricante construiu o seu próprio mecanismo de acesso.
- Alguns bancos de dados permitem manipulação de dados via *store procedure*. O processo de chamada da *store procedure* são dependentes da implementação do sistema do banco de dados.

A camada de negócios encontra dificuldade para acessar dados de diferentes fontes de dados. Visto que cada fonte de dados possui o seu próprio mecanismo de acesso. A Figura III.20 mostra o problema.

Há também, quando a camada de negócios precisa acessar apenas uma fonte de dados, mas se o banco de dados mudar em algum momento, todos os componentes da camada de negócio que acessam a fonte de dados diretamente serão alterados. Então quando a fonte de dados não está definida, se torna um grande problema de manutenção e baixa flexibilidade do sistema.

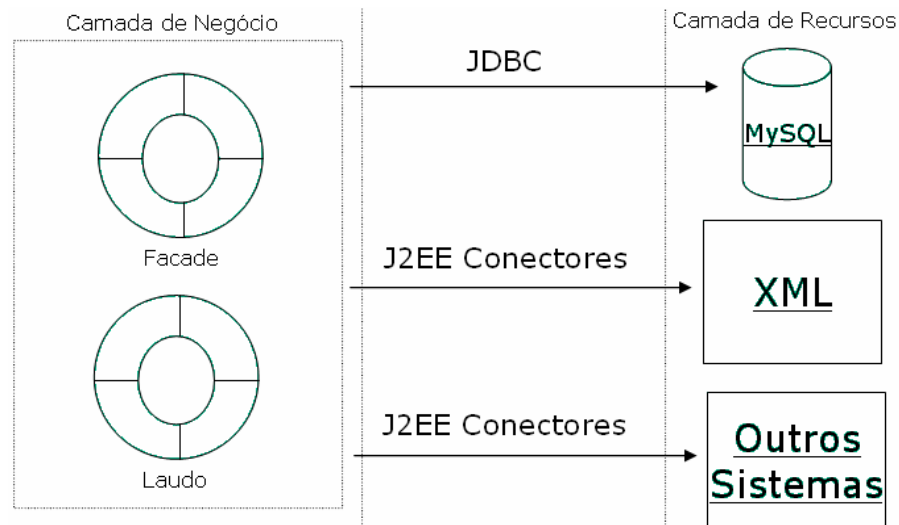


Figura III.20 – Negócio acessando base de dados

Desta forma, a lógica de negócio implementada pelos objetos de negócio não devem depender do tipo do armazenador de dados que é usado. O algoritmo de negócio deve ser separado do código que acessa o banco de dados. A lógica de negócio implementada pelo objeto de negócio não deverá depender da forma em que o dado armazenado é acessado.

A solução apresentada na Figura III.21, cria um objeto especial que se ocupa em acessar os banco de dados. O DAO é uma classe abstrata ou uma interface que tem métodos como *select(..)*, *update(..)*, *delete(...)* e *insert(..)*, as classes derivadas deverão então implementar esses métodos na forma específica que é requerida pelo banco de dados individualmente. Então os *dao's* encapsulam a lógica de acesso aos dados, e o objeto de negócio que usa as classes *dao's* não precisam conhecer sobre algum conjunto de código *SQL* ou as *API's* de baixo nível fornecidas pelo fabricante do banco de dados.

O padrão DAO fornece uma camada de abstração entre as camadas de negócio e a fonte de dados. Objetos da camada de negócio acessam a fonte de dados via *Data Access Object*. O DAO encapsula os detalhes de persistência e fornece um conjunto padrão de interface para acessar o dado. Esse objeto realiza todos as operações relacionadas à fonte de dados, tais como, consultas *SQL*. Isso limita o impacto de mudanças da fonte de dados para apenas as classes que compõe o *DAO*, isso melhora a flexibilidade e manutenibilidade dos objetos de negócio.

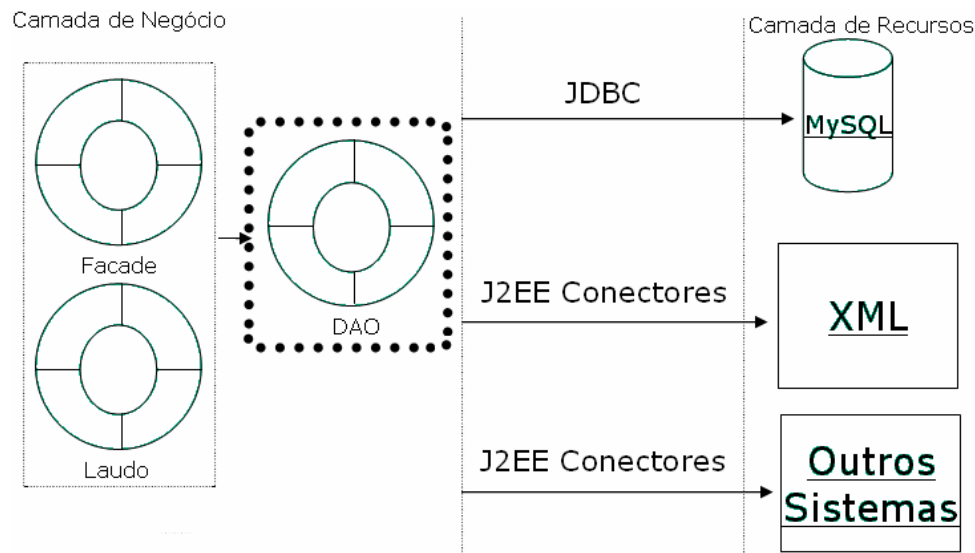


Figura III.21 – Exemplo do padrão DAO

O objeto de negócio, usa o DAO para alcançar o dado armazenado, o próprio DAO abstrai e encapsula as características da fonte de dados que a camada de negócio precisa para acessar os dados.

Desta forma, a fonte de dados será qualquer espécie de banco de dados ou sistema legado e a aplicação vai selecionar dinamicamente a fonte de dados plugando o apropriado DAO.

Arquitetura da Aplicação proposta

A aplicação será construída com componentes para as camadas cliente, apresentação, negócio, integração e dados. A Figura III.22 mostra os padrões cooperando entre si formando a arquitetura.

A camada do cliente permitirá o cliente interagir com a aplicação, essa pode ser uma aplicação *web* rodando em um *browser*, uma aplicação *desktop* ou até uma aplicação *Wirelles* rodando em um *Palm*.

A camada de apresentação define o nível de permissão do usuário, encapsula a lógica da *interface*, gerencia a sessão do usuário, recebe e envia respostas para o usuário.

A camada de negócios é o módulo responsável pela realização das regras de negócio da aplicação.

A camada de integração é responsável pela comunicação com os recursos externos.

A camada de recursos é responsável por manter os dados persistidos.

Desta forma, cliente *web* fará sempre uma requisição para um *Front Controller* que delegará a um *Dispatcher* a função de criar um *action* para atender a solicitação e encaminhar a resposta para um *View* formar a saída para o cliente.

O cliente *Desktop*, acessará o *Facade* diretamente através de uma chamada remota. O padrão *Service Locator*, faz a localização dos objetos remotos.

O *Facade* é um controlador da regra de negócio. Ele conhece o fluxo de atendimento da aplicação, sua principal função é acessar a base de dados através dos *Dão's* criados pelos *Abstracts Factory* e *Factory*, executar a regra de negócio encapsulada pelos objetos de negócios e garantir a resposta para o cliente criando *Value Objects* apropriados para cada solicitação.

O sistema possuirá *Implementation Dao* para cada meio de armazenamento no sistema. os meios mais comuns são XML, Sistema de arquivo e banco de dados relacional.

Pretende-se desta forma uma arquitetura que permita atender os princípios básicos de uma aplicação: Performance; transparência; flexibilidade; confiabilidade e escalabilidade.

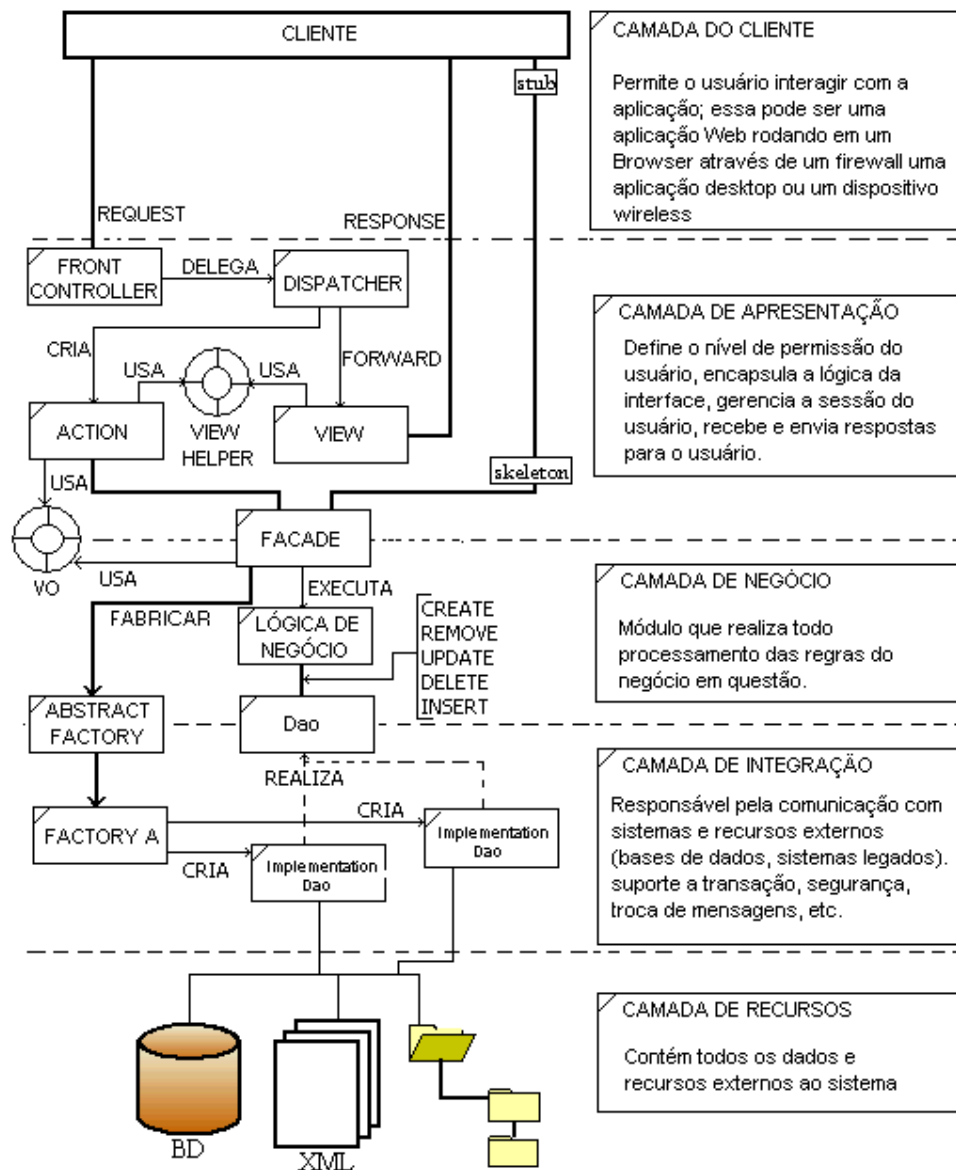


Figura III.22– Arquitetura da aplicação

IV- RESULTADOS

UC1 - Manter Laudo

A Figura IV.1 mostra o gerenciamento do laudo, o sistema *inclui, altera e exclui* um laudo do sistema. Um laudo é composto por informações de negócio, vídeos e fotos. As informações de negócio serão customizadas para atender os clientes segundo as suas próprias regras. Os vídeos são os procedimentos capturados, as fotos são resultados instantâneos durante a realização do procedimento.

O sistema permite uma busca parametrizada pelo título do laudo, essa deverá ser criteriosa quanto ao parâmetro de busca visando trazer somente o laudo desejado não causando sobrecargas desnecessárias na rede com o tráfego de informações que não serão utilizadas.

Esse caso de uso é caracterizado como gerencial, pois os dados são rastreados e encontrados facilmente pelo critério de busca, logo é importante criá-lo com um título sugestivo que caracterize o laudo.

Essa tela concentra todas atividades funcionais do sistema.

Cadastrar Laudo

Título:

Criação:

Descrição:

Vídeos

Nome	Resolução	Encoding	Formato
Lara_Mariana_CIF.avi	352x288	RGB	MSVideo (avi)
Lara_Mariana_Origl...	640x480	RGB	MSVideo (avi)

Fotos

Nome	Resolução	Tamanho
Foto 1	352x288	304544
Foto 2	352x288	304544
Foto 3	352x288	304544

Pesquisa

Título	Data criação	Responsável
Teste Carótida	15/11/2007	TIBOL
Teste Gravidez	16/11/2007	TIBOL

Usuário: TIBOL RMI offline Chat offline

Figura IV.1– Protótipo do caso de uso Manter Laudo

UC2 - Pesquisar Laudo

A Figura IV.2 mostra uma pesquisa sendo realizada remotamente. Essa consulta é realizada sobre o protocolo RMI (JRMP). O usuário remoto faz uma pesquisa por parte do título e clica no botão pesquisar. O sistema realiza uma consulta remota com o título fornecido como parâmetro. Caso encontre laudos com o parâmetro fornecido, a tela do cliente remoto é preenchida com os laudos consultados. Basta clicar em algum laudo para o seu preenchimento no formulário.

Assim como no caso de uso Manter Laudo, o critério de busca deve ser criterioso, de modo a não acarretar em uma busca com informações desnecessária sobrecarregando a rede.

Na versão *desktop* do sistema não será permitido acessar o vídeo sob demanda, somente em tempo real, portanto está previsto um caso de caso que fará um agendamento de quando o vídeo estará disponível para transmissão em tempo real. Desta forma o usuário o acessará através de um *link* nessa tela.

A foto é visualizada e analisada local ou remotamente ao clicar no nome da mesma na tabela de foto.

O cliente remoto somente tem direitos de consulta nos dados do laudo.

Nome	Resolução
Foto 1	352x288
Foto 2	800x564

Título	Data criação	Responsável
Teste Carótida	15/11/2007	TIBOL
Teste Gravidez	16/11/2007	TIBOL

Figura IV.2– Protótipo do caso de uso Pesquisar Laudo

UC3 - Realizar Procedimento

A Figura IV.3 mostra a exibição do sinal diretamente do dispositivo de captura (ultra-som, webcam, etc..). Para capturar um procedimento é necessário um laudo ativo no formulário. Após pressionar o botão Sensor, o JMStudio é ativado para realizar o procedimento. A captura iniciará após a entrada dos parâmetros adequados para realizar o procedimento.

A captura é influenciada pelos parâmetros imposto pelo usuário. O *encoding*, a resolução e a quantidade de *frames* por segundo são os parâmetros que exigem uma tomada de decisão, pois são características impostas pelo negócio e não pelo sistema.

Através de uma ação no menu (Arquivo e *Player*) é possível gravar o vídeo e gerar fotos instantâneas que serão automaticamente associadas ao laudo ativo no formulário. O botão localizado no canto inferior direito quando acionado, exibe um formulário com informações técnicas do vídeo que está sendo exibido. O botão localizado no canto inferior à esquerda quando acionado, para ou reinicia a captura.

O teste realizado na Figura IV.3 foi com uma webcam.

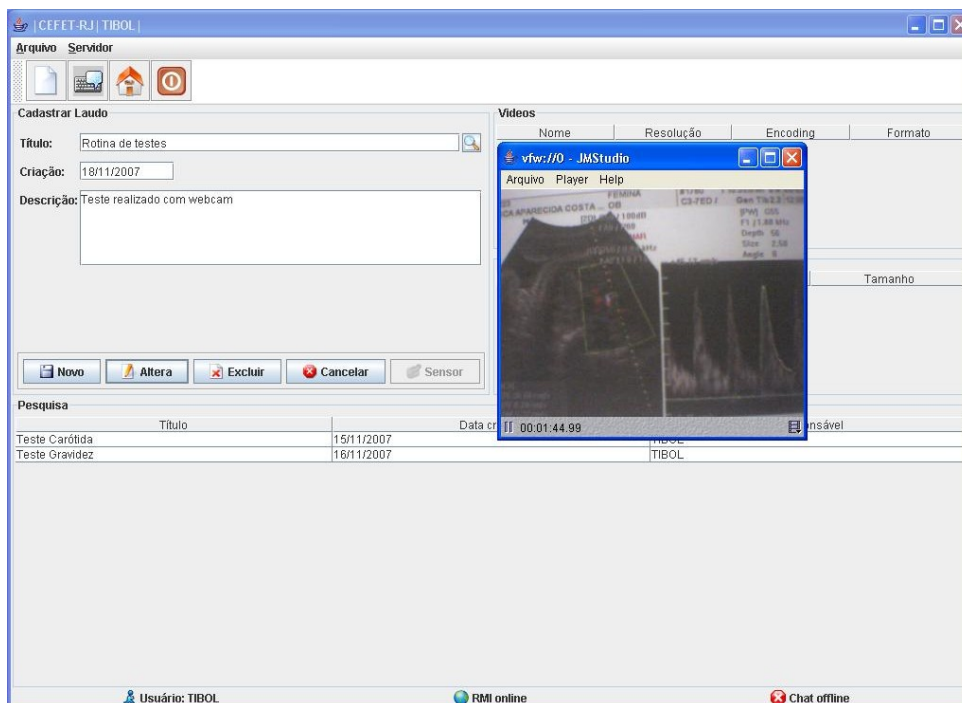


Figura IV.3— Protótipo do caso de uso Realizar Procedimento

UC4 - Gravar procedimento

A Figura IV.4 mostra um vídeo sendo gravado em diretório. O caminho de gravação somado ao nome do arquivo é gravado no campo url da entidade Laudo. Desta forma, o vídeo gravado está associado ao laudo gerado pelo usuário radiologista. O sistema realiza a transação necessária quando um laudo é excluído do sistema.

A ação gravar um vídeo é acionada através de uma ação no menu Arquivo, vários *encoding* estão disponíveis para a gravação. Demais parâmetros como resolução de *frames* por segundo também precisam ser determinados pelo usuário. Essa parametrização também é considerada uma decisão de negócio e não do sistema.

Caso um vídeo seja removido do diretório gravado pelo sistema, uma inconsistência será gerada sendo aconselhável exclusão do mesmo. Portanto, é desaconselhável acessar um vídeo diretamente pelo sistema de diretório do sistema operacional.

Atualmente estão disponíveis os formatos *QuickTime(mov)* e *MSVideo(avi)* para realizar a gravação.

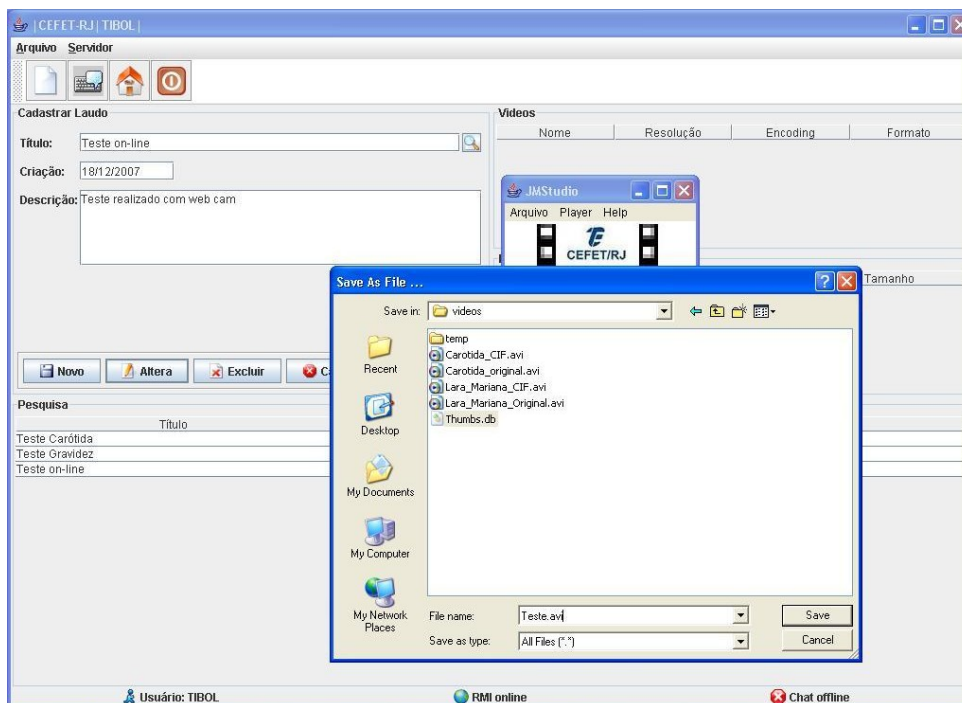


Figura IV.4– Protótipo do caso de uso Gravar Procedimento

UC5 - Abrir Procedimento Local

A Figura IV.5 mostra o vídeo de um laudo sendo exibido. Para iniciar a exibição é necessário clicar no vídeo desejado na tabela de vídeos no formulário que está exibindo os dados de um laudo. Logo a seguir basta confirmar a abertura do mesmo para iniciar a exibição pelo JMStudio.

Os controles *iniciar*, *parar*, *retroceder* e *avançar* estão presentes na exibição de um vídeo gravado, esses controles são necessários pois permite uma interação com o vídeo exibido. O mesmo será exibido com os parâmetros gravados.

O botão localizado no canto inferior à direita foi mencionado anteriormente e também possui a função de apresentar as propriedades do vídeo em exibição.

Devido ao gerenciamento eficaz da ferramenta, o usuário não precisa saber onde está localizado o vídeo, basta um clique no nome do mesmo para iniciar a exibição, isso é possível porque o caminho do vídeo no diretório de gravação está armazenado no campo *url* da entidade Vídeo no banco de dados do sistema.

Essa opção não está disponível para o usuário remoto, uma vez o trabalho possui foco em transmissão em tempo real.

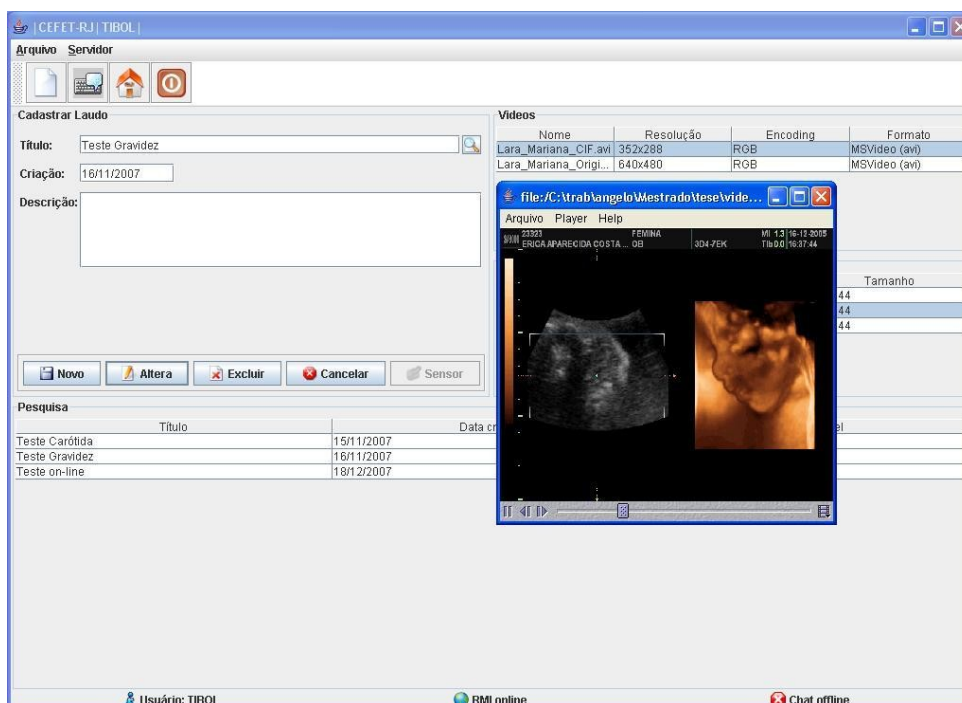


Figura IV.5– Protótipo do caso de uso Abrir Procedimento Local

UC6 - Transmitir procedimento on-line

A Figura IV.6 mostra um procedimento sendo transmitido diretamente do dispositivo de captura (ultra-som, webcam, etc..). Para transmitir um procedimento é necessário um Laudo ativo no formulário. Após pressionar o botão sensor, o JMStudio é ativado para realizar a captura e transmissão do procedimento. A transmissão iniciará após a entrada dos parâmetros adequados para realizar a transmissão.

Um objeto que controla a qualidade é criado na memória para realizar o controle de qualidade solicitado pelo usuário remoto.

Através de uma ação no menu *Player* o sistema apresenta um relatório estatístico da transmissão, essas informações são importantes pois refletem diretamente na qualidade da exibição do vídeo no cliente remoto. Entre as informações destacam-se: Total de pacotes enviados, colisões locais, colisões remotas e falhas na transmissão.

A transmissão é realizada pelo protocolo RTP e a estatística é fornecida pelo protocolo RTCP

A captura e transmissão da Figura IV.6 foram realizadas com uma webcam.

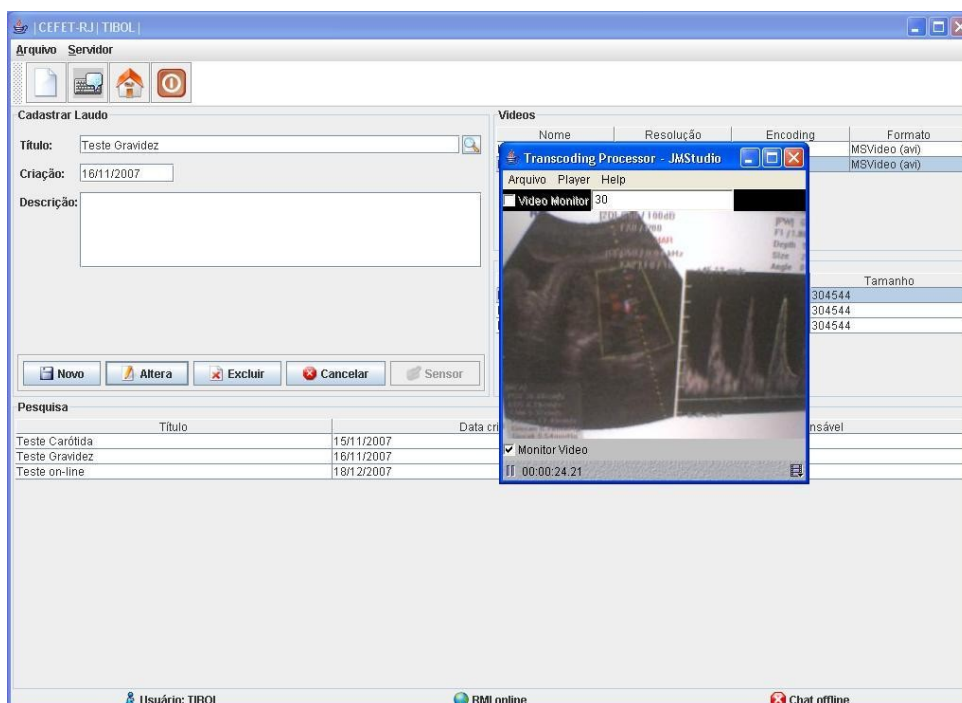


Figura IV.6— Protótipo do caso de uso Transmitir Procedimento On-Line

UC7 - Transmitir procedimento off-line

A Figura IV.7 mostra o vídeo de um Laudo sendo transmitido. Para iniciar a exibição é necessário clicar no vídeo desejado na tabela de vídeos no formulário que está exibindo os dados de um laudo. Logo a seguir basta confirmar a transmissão que o JMStudio é ativado para realizar a captura no diretório e a transmissão do procedimento. A transmissão iniciará após a entrada dos parâmetros adequados para realização do mesmo.

No procedimento off-line é possível controlar a exibição da transmissão através dos controles *iniciar*, *parar*, *retroceder* e *avançar*. No mais, é similar ao procedimento on-line, também é transmitido pelo protocolo RTP, a estatística da transmissão é fornecida pelo protocolo RTCP, e também é criado um objeto para controlar a qualidade de captura e transmissão conforme as solicitações do cliente remoto.

Apesar de ser um vídeo *off-line* é necessário enviá-lo em tempo real, pois o mesmo sofre todas interferências de qualidade de serviço que um vídeo on-line. As opções encontradas na *Internet* são transmissões sob demanda, essa transmissão não é realizada com o protocolo RTP, logo, não possui garantias que chegará ao seu destino.

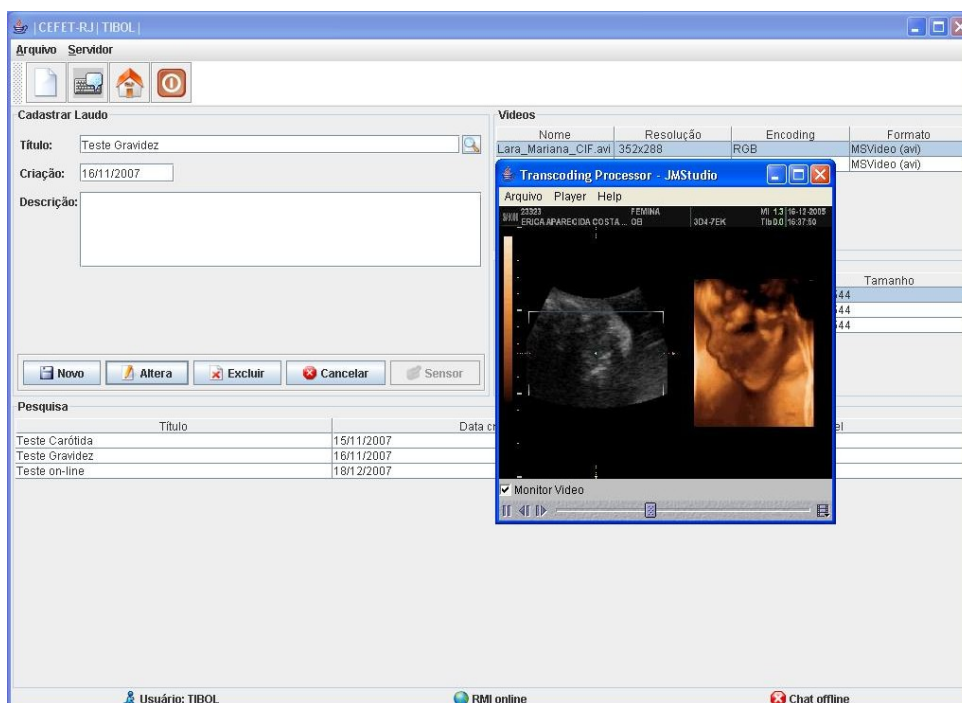


Figura IV.7– Protótipo do caso de uso Abrir Procedimento a distância

UC8 - Abrir procedimento à distância

A Figura IV.8 mostra um vídeo sendo exibido em tempo real em um cliente localizado na rede TCP/IP. Para iniciar esse procedimento deve-se clicar no botão Media. O JMSutio será carregado e a visualização iniciará após a entrada dos parâmetros adequados para iniciar uma sessão em tempo real. Esse procedimento só terá êxito caso o servidor esteja enviando um vídeo em uma sessão RTP.

Esse caso de uso é o cerne deste trabalho, a partir desse ponto, aplica-se um conceito inovador para ajuste da qualidade da imagem através da influencia direta do cognitismo humano. Um objeto visual *Slider* é apresentado para realizar o ajuste cognitivo, o resultado é apresentado no caso de uso *Ajuste Cognitivo*.

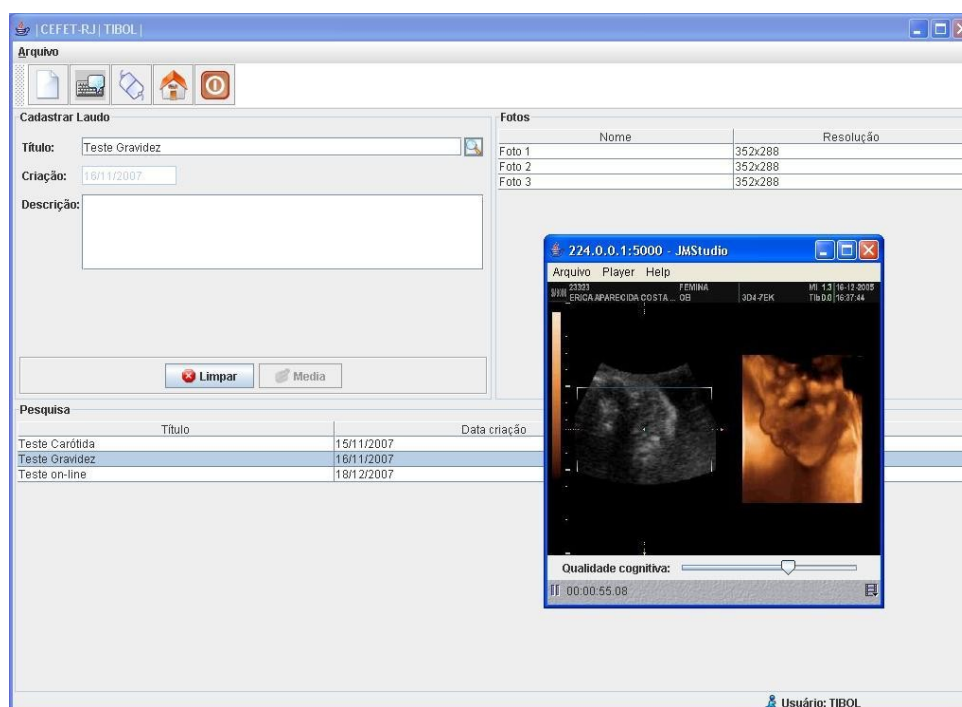


Figura IV.8— Protótipo do caso de uso Transmitir Procedimento Off-Line

UC9-Ajuste cognitivo

As Figuras IV.9 e IV.10 mostram um vídeo sendo exibido em tempo real em um cliente localizado na rede TCP/IP. Para iniciar esse procedimento deve-se clicar no botão *Media*. O JMSutio será carregado e a visualização iniciará após a entrada dos parâmetros adequados para iniciar uma sessão em tempo real. Esse procedimento só terá êxito caso o servidor esteja enviando um vídeo em uma sessão RTP. A partir desse ponto, o cliente possui uma ferramenta caracterizada por um objeto *Slider* que atuará na fonte receptora alterando a qualidade do vídeo capturado e transmitido, a Figura IV.9 mostra o vídeo ajustado pelo *Slider* para obter qualidade máxima e a Figura IV.10 mostra o vídeo ajustado pelo *Slider* para obter qualidade mínima. A posição ideal depende de vários fatores, entre eles destaca-se o *encoding* usado na transmissão.

O *encoding* usado nas Figuras IV.9 e 10 foi o JPEG-RTP, nesse caso, percebe-se que ambas as imagens em cada figura são as mesmas, ou seja, a imagem gerada no servidor é a mesma que aparece no cliente, caracterizando o tempo real. O *Slider* da Figura IV.10 foi propositalmente colocado na posição que representa o pior caso de geração da imagem, as imagens continuaram sendo as mesmas. A região que contém imagem gerada pelo ultra-som não foi alterada com a diminuição da qualidade de captura, somente os caracteres contidos na imagem sofreram degradação, caso essa influência venha gerar desconforto na análise da imagem, o próprio médico deverá ajustar a posição do *Slider* para uma posição que melhor o atenda.

Quanto pior a qualidade da imagem na fonte geradora, menor será o seu tamanho e melhor será a transmissão, logo, o médico localizado remotamente terá que possuir conhecimentos suficientes para decidir a razão “*tamanho x qualidade*” para o momento que está ocorrendo à teleconferência. Não é possível definir uma posição ideal, pois as variáveis que influenciam na transmissão são dependentes de fatores fora do controle da aplicação, tais como *jitter*, atrasos e largura de banda.



Figura IV.9– Protótipo do caso de uso Ajuste Cognitivo (melhor caso)



Figura IV.10— Protótipo do caso de uso Ajuste Cognitivo (pior caso)

UC10-Analisar Laudo

A Figura IV.11 mostra uma foto sendo analisada em um cliente e sendo acompanhada pelo radiologista remotamente. Essa ferramenta permite que uma imagem instantânea de um vídeo seja capturada e tenha os seus *pixels* manipulados por funções especiais. O cliente tem a opção de realizar a análise localmente ou com a participação do servidor, enviando ao mesmo, todas as manipulações efetuadas na imagem original em tempo real.

A Figura IV.11 mostra a função *Lupa* aumentando a imagem do olho do *feto* no útero da gestante. Essa ação está sendo compartilhada entre o médico especialista e o radiologista, ou seja, cliente e servidor.

Outras funções estão disponíveis, porém, ainda é necessário um estudo para mapear as funções disponíveis com a sua aplicabilidade.

A manipulação remota tem como inovação, a garantia que ambos participantes estão vendo a mesma imagem, uma vez que no início do processo é feita uma cópia da imagem e passada por RMI para outro computador participante, já as manipulações são realizadas localmente em cada máquina, apenas sinais de comando via RMI são trafegados pela rede.

A única interferência do radiologista neste processo é fechar a tela ao final do processo.

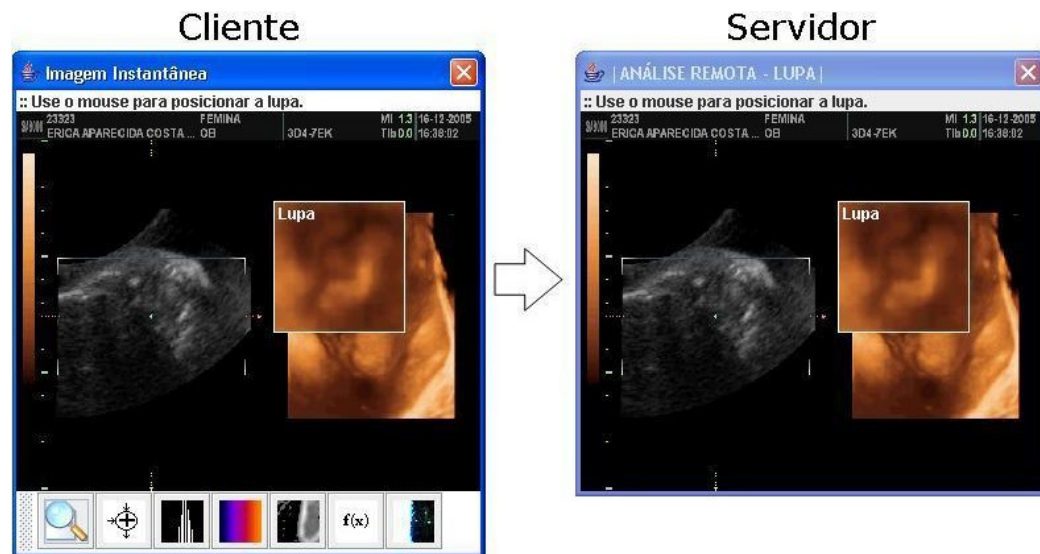


Figura IV.11– Protótipo do caso de uso Analisar Laudo

UC11-Chat

A Figura IV.12 mostra o chat do sistema através de texto. Essa ferramenta tem o objetivo de aproximar os participantes que estão participando da análise do laudo de modo a facilitar a comunicação do mesmo. O radiologista que está gerando o sinal atuará em cima das solicitações do especialista em tempo real.

A inovação está no fato da integração com o sistema de áudio e vídeo na mesma aplicação. No mais, trata-se de uma aplicação já estudada, encontrada em livros didáticos disponíveis no mercado.

O *chat* é soberano em relação à comunicação usando voz devido à quantidade de dados que são enviados pela rede. Uma aplicação de voz possui requisitos de qualidade superiores aos do texto.

O sistema está preparado para enviar áudio através do JMStudio usando os mesmos procedimentos aplicados para a imagem, porém não foi feito nenhum estudo inovador no sentido de melhorias na sua transmissão.

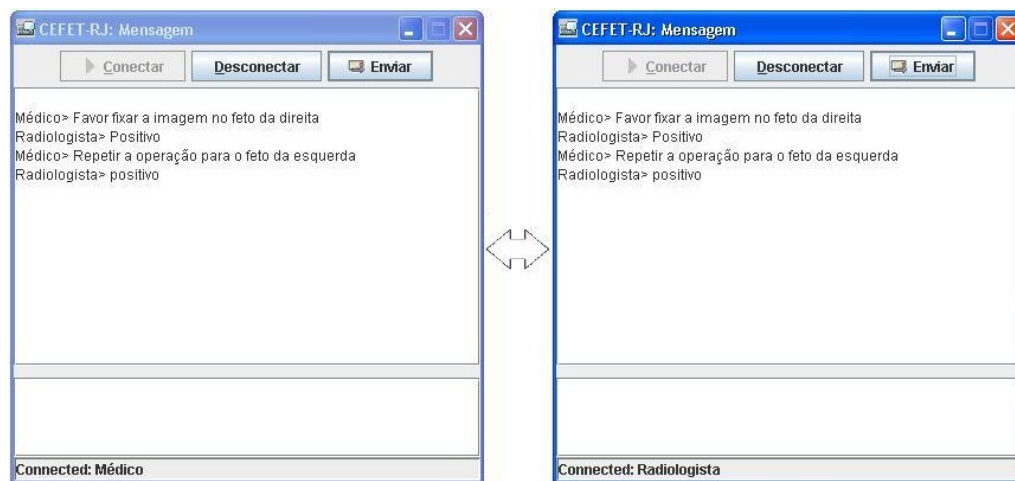
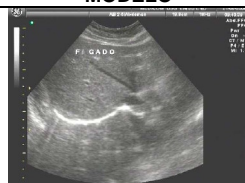






Figura IV.12– Protótipo do caso de uso Chat

Modelo de Compressão








Os tamanhos originais das imagens geradas, tanto para armazenamento, quanto para transmissão precisam ter os seus tamanhos reduzidos para serem coerentes com os meios de armazenamento e transmissões atuais. No modelo de compressão sem perdas, a imagem após o processo de descompressão será exatamente igual à imagem original. Porém, os resultados dos testes apresentados nas Tabelas IV.1 e IV.2 mostram que ao usar um modelo de compressão com perdas, as informações perdidas são redundantes e imperceptíveis para o sistema visual do ser humano. Essa análise tem o objetivo de aplicar os modelos de compressão existente de modo a auxiliar no processo de escolha do algoritmo de compressão a ser empregado para imagens provenientes de sensores ultra-sônicos. A Tabela IV.1 registra as compressões alcançadas por modelos de compressão atualmente disponíveis.




Tabela IV.1- Análise da imagem 640x480 do fígado

MODELO	VALOR DO PIXEL	TAMANHO	OBSERVAÇÃO
	figado.bmp	901 kbytes	
	figado.gif	46 kbytes	Houve perdas sensíveis para o olho humano.
	figado.jpg	33 kbytes	Excelente resultado
	figado.png	238 kbytes	
	figado.tif	341 kbytes	

O padrão de compressão jpeg tem mostrado o porque vem sendo citado nas demais bibliografias como uns dos mais usados sendo inclusive indicado para compressão de vídeo. Para confirmar o resultado foi realizado o mesmo teste em mais dez imagens diferentes mas da mesma origem, sendo o resultado registrado na Tabela IV.2

Tabela IV.2- Análise das imagens 640x480 de biomicroscopia

IMAGEM	IMAGEM	BMP KByte	GIF KByte	JPG KByte	PNG KByte	TIF KByte
	AORTA	901	35	19	174	257
	BACO E PANCREAS	901	51	27	286	427
	BEXIGA	901	35	24	188	255
	COLEDOPO	901	46	31	248	344
	OVÁRIO DIREITO	901	43	26	234	329
	PANCREAS	901	44	29	237	326
	RETRO AUREOLAR DIREITO	901	50	31	265	360

	RIM	901	60	42	340	481
	VESÍCULA	901	35	20	162	237
	VESÍCULA E COLEDOCO	901	48	29	245	355

O modelo JPEG apresentou nos ensaios uma compressão muito superior em relação aos outros modelos de compressão testados. As imagens testadas possuem originalmente 921,6 kBytes $[(640 \times 480 \times 24)/8]$, por exemplo, no caso da imagem aorta, o padrão jpeg atingiu uma compressão de aproximadamente 50:1, ou seja, a imagem ficou cinquenta vezes menor $(921,6/19 = 48,5:1)$. O padrão gif gerou marcas contínuas na imagem que foi percebida visualmente, o resultado é apresentado na Figura IV.13

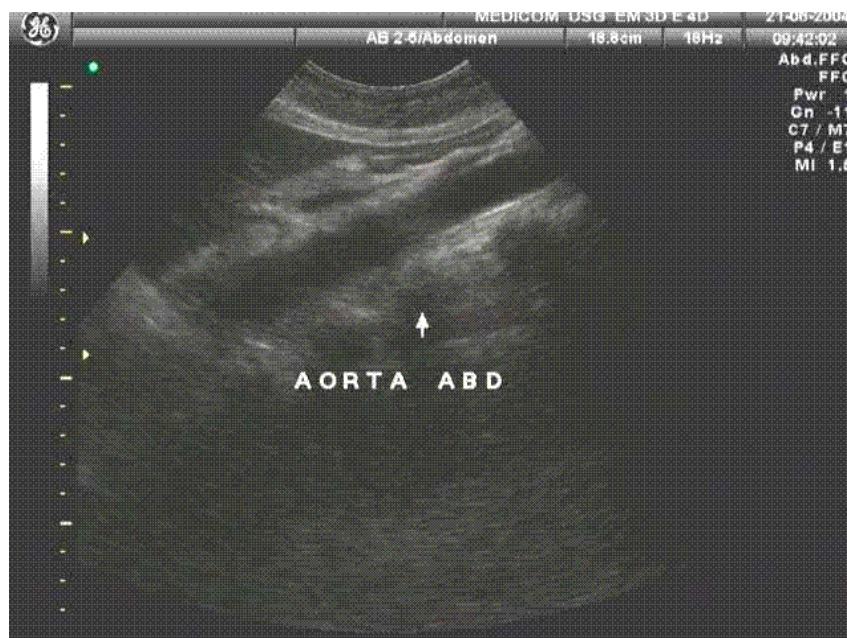


Figura IV.13 – Imagem gravada no padrão GIF.

Uma imagem usando o modelo de compressão/descompressão JPEG apresentará perdas consideráveis, isso é unanimidade entre os autores citados, porém, o quanto essa perda na imagem vai prejudicar a qualidade visual da mesma vai depender do tipo da imagem que está sendo trabalhada. Essa dissertação está fazendo um embasamento empírico simulando várias imagens para afirmar que uma imagem proveniente de biomicroscopia, assim como uma imagem de uma paisagem da natureza, pode ser classificada como um tipo que possui resultado satisfatório quando se aplica o padrão JPEG.

Os resultados mostram que o padrão JPEG é uma técnica de compressão de imagem com qualidade para ser aplicado em imagens de biomicroscopia e que não apresenta perdas visuais percebíveis pelo aparelho visual humano.

V- DISCUSSÃO

O objetivo foi criar uma aplicação que permitisse a troca de informações de vídeo, com qualidade, entre pontos remotos. Um sistema de objetos distribuídos é aquele que permite a operação com objetos entre máquinas diferentes. Dessa forma é possível, a partir de uma aplicação cliente orientada a objetos em uma máquina remota, obter uma referência para um objeto em outra máquina, que oferece o serviço desejado, e através dessa referência, invocar métodos desse objeto. Essencialmente, uma arquitetura orientada a objetos estabelece as regras, diretrizes e convenções definindo como as aplicações podem se comunicar e interoperar. Esse embasamento deu suporte a todo desenvolvimento remoto da aplicação.

Um software para ser implementado precisa levar em consideração a interação com o ser humano que vai operá-lo. Consideramos ser melhor, o próprio homem embasado na sua experiência, ajustar a qualidade da imagem que ele está vendo, do que permitir uma equação rodando em paralelo influenciar sozinha na qualidade e trazer um desconforto para quem está vendo. Uma insatisfação do usuário leva a inutilização do produto.

A dimensão alcançada pela *Internet*, torna-se impraticável esperar que os equipamentos que compõem as redes fiquem unicamente responsáveis para atender as necessidades de uma aplicação multimídia, desta forma, as aplicações também devem implementar técnicas que diminuam o tamanho dos arquivos compatibilizando-os com os atributos das redes atuais para eliminar problemas ocorridos na transmissão.

A melhor forma de transmitir dados multimídia é possuir uma rede de dados com largura de banda e latência compatíveis com as necessidades exigidas para uma dada aplicação. Obviamente, quando se projeta uma rede de dados, todas variáveis são levantadas e a mesma é implementada com os recursos de qualidade de serviço (QoS) compatível para atender os requisitos desejados. Com o passar do tempo, as necessidades vão sendo alteradas, e as redes não possuem escalabilidade, tornando-se incapazes de atender a novas demandas sem alterações no seu formato original, assim, iniciam os problemas para entregar o dado até o seu destino acarretando perdas de pacotes, atraso fim a fim e variação de atraso (*jitter*), elementos que não existiam no projeto inicial. Esse é o retrato da *Internet*, onde inicialmente era uma rede de uso exclusivo das forças

armadas americana que trafegava somente dados estáticos e hoje em dia é base de comunicação para uma infinidade de sistemas por todo mundo.

Uma forma de minimizar os problemas, é não duplicar informações pela rede, logo a conexão *multicast* é uma forma de comunicação que contribui para minimizar os problemas, visto que a comunicação ponto a ponto cada elemento na rede receberá uma cópia do dado.

Entre as técnicas disponíveis para diminuir o tamanho da imagem, a tabela de quantização utilizada no modelo de compressão JPEG-RTP e no H.263 é a que causa menos transtorno para o usuário final, visto que reduzir o número de quadros por segundo gera um desconforto na continuidade do vídeo e reduzir a resolução vai diminuir o tamanho da imagem. Ambos modelos de compressão são utilizados pela aplicação, o formato JPEG-RTP deve ser empregado quando o conjunto possuir CPU de alta capacidade, alta largura de banda e exigir alta qualidade na imagem. O formato H.263 não requer tanta largura de banda, não exige muita CPU, porém a qualidade será menor que o JPEG-RTP. Essa decisão é feita no início da transmissão.

Atualmente, é justo afirmar que as tecnologias de software e protocolos atendem aos requisitos impostos pelas necessidades dos dados multimídia. Ainda é necessário melhorar a velocidades atuais das redes de comunicação bem como a qualidade de serviço fornecida.

Protocolos como RTP e RTCP são unanimidades para aplicações multimídias, porque foram especificados para trabalhar em situações críticas como as exigidas por essas aplicações.

A linguagem Java foi fundamental pois a mesma realiza comunicação remota com RMI, trabalha com áudio e vídeo com JMF e processamento de imagem com JAI. A tecnologia RMI além de ser eficaz, simplificou a comunicação das chamadas de métodos remotos. O JMF fez todo processamento dos vídeos e áudio de acordo com os atuais protocolos e codificadores/decodificadores consagrados no mercado. O JAI é uma poderosa ferramenta para processamento da imagem através da manipulação dos seus *pixels*.

Essa integração foi um desafio que a lista de trabalhos futuros mostra que ainda está longe de ser saturada. A nossa aplicação é um alicerce para implementações futuras. Esse alicerce já é um grande avanço na área de exames que utilizam imagem por ultra-som, pois um simples armazenamento significa um

rastreamento para a vida toda, uma transmissão significa levar os melhores especialistas a lugares jamais alcançados.

A Figura V.1 resume as inovações alcançadas com a criação deste software de gerenciamento, análise e transmissão da imagem.



Figura V.1 – Inovação do produto

A análise remota em tempo real é uma inovação pois permite o radiologista acompanhar a análise realizada pelo médico especialista, desta forma, o radiologista focará o exame exatamente no ponto especificado, no momento certo, ocasionando uma precisão no exame realizado.

O ajuste cognitivo permite o especialista decidir se a qualidade da imagem está afetando o resultado do exame e não parâmetros de qualidade de serviço agregados em modelos matemáticos, desta forma, o usuário remoto vai “sintonizar a imagem” desejada.

A integração multimídia é inovadora pois consideramos que não possui no mercado uma aplicação que reúna todas essas funcionalidades em uma única ferramenta. Estão disponíveis ferramentas que trabalham com áudio e vídeo ou somente com processamento de imagens estáticas.

Esse software foi construído com ferramentas livres. Não teve custo de licenças para produção nem distribuição, é multiplataforma, não exige produtos casados, está em conformidade com a atual política de software do governo federal e possui o seu código aberto para atualizações que contribuam para o crescimento da pesquisa.

CONCLUSÃO

Concluimos que, devido à importância alcançada pelos exames de ultra-som, é necessário um software que realize gerenciamento dos dados, análise através de técnicas de processamento de imagem e trabalhos colaborativos entre especialistas através da transmissão do procedimento em tempo real em conjunto com técnicas de comunicação on-line.

A gerência dos dados obteve sucesso devido à integração dos elementos computacionais usados, a linguagem de programação Java foi o pilar do desenvolvimento tornando a aplicação escalável e transparente.

A análise através de técnicas de processamento de imagem trouxe um ganho extra na conclusão do laudo do ultra-som, pois a manipulação dos *pixel's*, mostrará informações que a imagem original não mostra.

O trabalho colaborativo através de técnicas de comunicação *on-line* permite uma comunicação entre os especialistas, dando ao especialista remoto, a chance de emitir a sua opinião no momento da realização do exame.

A transmissão é dotada de um ajuste cognitivo que permite o médico especialista atuar na qualidade da imagem de modo a eliminar perturbações ocasionadas pela rede de comunicação.

Então, a segunda conclusão, é que sendo a gerência da informação, o processamento dos dados e o trabalho colaborativo funcionalidades básicas quando tratadas individualmente, juntas permitirão os usuários obter melhores resultados, promovendo controle, rapidez e eficiência, requisitos fundamentais em um laudo médico. E o ajuste cognitivo atuará na qualidade do exame, sendo esse fundamental para um bom diagnóstico.

Concluimos também que o especialista que se encontra remotamente é quem deve disparar e controlar a intensidade na degradação da imagem, pois o mesmo é o principal elemento afetado caso a imagem não chegue em perfeitas condições e que o mesmo fará esse ajuste cognitivamente, tanto na degradação da imagem na fonte geradora, quanto no tamanho do *buffer* da sua estação, para diminuir as perdas e atrasos respectivamente.

Desta forma, este sistema está pronto para ser usado sobre a camada de aplicação do modelo TCP/IP e realizar transmissão de vídeos pela *Internet* de forma gerenciada, com ferramentas para processamento de imagem, meios colaborativos entre os participantes e uma ferramenta de apoio para facilitar e controlar a transmissão.

Trabalhos Futuros

As aplicações de teleconferência, vídeo sob demanda, processamento distribuído e troca de mensagens simultâneas terão êxito quando forem implantadas sob redes e softwares com as tecnologias e arquiteturas apresentadas.

Desta forma alguns pontos ainda precisam ser adequados para um pleno atendimento dos requisitos funcionais e não funcionais para a solução completa dos problemas. A seguir serão descritos alguns problemas que ainda não foram solucionados:

- Simulador matemático “cognitivo” para vídeo
- Simulador matemático “cognitivo” para áudio
- Reconhecimento da imagem
- Aplicações para o processamento de imagem
- Definir o melhor padrão para transmissão de vídeo
- Definir o melhor padrão para transmissão de áudio
- *Ergodesigner*

Os simuladores matemáticos “cognitivos” para áudio e vídeo devem ser eficazes trazendo conforto para quem está assistindo, sem a interferência do homem, porém com a mesma eficiência. A idéia é fazer o sistema auto-aprender as condições ideais de transmissão conforme o ajuste efetuado ao longo do tempo da posição do *slider* que ajusta a qualidade com as condições de *jitter*, atrasos e perdas de pacotes, associado a um determinado usuário. O sistema deverá saber quem está usando o sistema e aplicar o ajuste que foi “aprendido com o tempo” para o usuário específico.

Dados empíricos mostram que cada tipo de vídeo possui um padrão de codificação mais eficaz para captura e transmissão. Desta forma é necessário realizar uma abstração e encaixar determinados tipos de imagens com natureza comum para que se possa indicar o melhor padrão de codificação para mesma.

O processamento de imagem através da manipulação dos *pixels* se torna uma ferramenta poderosa, porém é necessário realizar um mapeamento indicando em quais problemas cada transformação pode auxiliar. A partir daí, novas soluções podem ser apontadas.

A melhor codificação para captura e transmissão depende de vários fatores objetivos e subjetivos. Essa aplicação não faz nenhuma indicação, apenas sugestão, quanto ao melhor codificador/decodificar. É necessário levantar quais fatores influenciam para cada tipo de codificador disponível

O *Ergodesigner* vem se tornando cada vez mais necessário para os atuais padrões de exigência de mercado. Fica uma pergunta no ar: Como o radiologista vai realizar o exame, falar em um microfone, escrever no teclado, acompanhar um *chat* e acompanhar a análise remota? É necessário um estudo profundo para que o radiologista consiga usufruir todas as inovações proposta no sistema de forma eficaz.

BIBLIOGRAFIA

ALUR, D.; CRUPI, J.; MALKS, D.; *Core J2EE Patterns: Best Practices and Design Strategies*, Califórnia, Sun Microsystems Press, 2001.

AHMED, K. Z.; UMRYSH, C.E.; *Developing Enterprise Java Applications with J2EE and UML*, 1 ed. Boston, Addison Wesley, 2001.

ALVES, C. H. F.; CHEN, W.; RITTER, T. A.; SHUNG, K. K. ; SNOOK, K. A.; “*Microscopia por Retro-espalhamento de Ultra-som (UBM)*”. In: *XVII Congresso Brasileiro de Engenharia Biomédica*, p. 660-664. Florianópolis, Santa Catarina, 2000

ALVES, C. H. F.; RITTER, T. A.; SHUNG, K. K.; “*Transdutores de Alta Frequência (50 MHz), Utilizando PVDF, para aplicações em Imagens Médicas*”. In: *XVII Congresso Brasileiro de Engenharia Biomédica*, p. 665-668, Florianópolis, Santa Catarina, 2000.

ANGELO, M.F.; *Sistema de Processamento de Imagens Mamográficas e Auxílio ao Diagnóstico via-Internet*, Dissertação de Doutorado, Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, SP, Brasil, 2007

ARAGÃO, A.S.L.; *Aspectos Técnicos e Metodológicos do Processo de Desenvolvimento de Aplicações Baseado em Componentes*, Dissertação de Mestrado, Pós-Graduação em Engenharia de Produção, Universidade Federal de Santa Catarina, Florianópolis, SC, Brasil, 2002.

BATISTA T.V.; FILHO G.L.S.; LEITE L.E.C.; “*DynaVideo – Um Serviço de Distribuição de Vídeo baseado em ConFiguração Dinâmica*”. In: *19º Simpósio Brasileiro de Redes de Computadores*, p. 838-852. Florianópolis, Santa Catarina, Maio 2001 .

BERNERS-LEE, T.; FIELDING, R.; FRYSTYK, H.; GETTYS, J.; IRVINE, U.C.; LEACH, P.; MOGUL, J.; MASINTER, L; **RFC2616** Hypertext Transfer Protocol--HTTP/1.1. Network Working Group, 1999. Disponível em: <<http://www.ietf.org/rfc/rfc2616.txt> >. Acesso em: 15 mai. 2007.

BOOCH, G.; JACOBSON, I.; RUMBAUGH, J.; *UML Guia do Usuário*, 2 ed. Rio de Janeiro, Campus, 2000.

CALABREZ, C.E.; *Uma Comparação entre Diversas Tecnologias de Comunicação de Objetos Distribuídos em Java*, Dissertação de Mestrado Profissional, Instituto de Computação, Universidade Estadual de Campinas, Campinas, SP, Brasil, 2004

CAMARGO, W.P.; *Desenvolvimento de um Ambiente WEB para a Interação Entre Participantes de Projetos de Agricultura de Precisão*, Dissertação de Mestrado, Escola Superior de Agricultura Luiz de Queiroz, Universidade de São Paulo, Piracicaba, SP, Barsil, 2005

CAMARILLO, G.; HANDLEY, M.; JOHNSTON, A.; PETERSON, J.; ROSENBERG, J.; SCHOOLER, E.; SCHULZRINNE, H.; SPARKS, R.; **RFC 3261** SIP: Session Initiation Protocol. Network Working Group, 2002. Disponível em: <<http://www.ietf.org/rfc/rfc1889.txt>>. Acesso em: 15 mai. 2007.

CARVALHO P.; DEUSDADO. S.; *Groupware multicast com QoS pró-ativa integrado num sistema de e-learning*. Escola de Engenharia, Universidade do Minho, Braga, Portugal, 2004

CASNER, S.; FOKUS, G.M.D.; FREDERICK, R; JACOBSON, V.; SCHULZRINNE, H.; **RFC1889** RTP: A Transport Protocol for Real-Time Applications. Network Working Group, 1996. Disponível em: <<http://www.ietf.org/rfc/rfc1889.txt>>. Acesso em: 15 mai. 2007.

COSTA, D.G.; *Comunicações Multimídia na Internet: Da Teoria à Prática*, 1 ed. Rio de Janeiro, Ciência Moderna, 2007.

DAVIE, B.S.; PETERSON, L.L.; *Redes de Computadores, Uma Abordagem Sistêmica*, 3 ed. Rio de Janeiro, Campus, 2004.

DEITEL, H.M.; DEITEL, P.J.; *Java: Como Programar*, 6 ed. São Paulo, Pearson, 2005.

DESHMUKH, H.; MALAVIA, J.; *SCWCD, Exam Study Kit: Java Web component Developer Certification*, Greenwich, 2003.

ELKIND M.; *Um Sistema Confiável de distribuição de Mídia*, Dissertação de M.Sc., COPPE/UFRJ, Rio de Janeiro, RJ, Brasil, 2006

ELMASRI, R.; NAVATHE, S. B.; *Sistemas de Banco de Dados*, 4 ed. São Paulo, Pearson, 2005.

ERNEST, M.; HELLER, P.; ROBERTS, S.; *Complete Java 2 Certification: Study guide*, 2 ed. Alameda, Sybex, 2000.

FERRAZ, M. C.; *Codificação de Imagens*, Dissertação de M.Sc., IMPA, Rio de Janeiro, RJ, Brasil, 1998.

FERREIRA, D.M.; LOPES, P.R.L.; PISA, I.T.; SIGULEM, D. "Benefícios da Utilização do Session Initiation Protocol (SIP) em Aplicações de Comunicação Multimídia para a Saúde". In: *X Congresso Brasileiro de Informática em Saúde - CBIS2006*, Florianópolis, SC, 2006.

FOWLER, M; SCOTT, K.; *UML ESSENCIAL*, 3 ed. São Paulo, Bookman, 2005.

FREIRE, A; *Inovação: Novos Produtos, Serviços e Negócios para Portugal*, Lisboa, Editora Verbo, 2000

GONZALEZ, R. C.;WOODS, R. E.; *Processamento de Imagens Digitais*, 1 ed. São Paulo, Edgard Blucher, 2005.

GOMES, F.L.S.; *VideoConferência: Sistemas e Aplicações*, Florianópolis, Visual Books, 2003.

HEXSEL, R. A.; *Software Livre: Propostas de Ações de Governo para Incentivar o Uso de Software Livre*, Curitiba, 2002. Disponível em: < http://www.inf.ufpr.br/info/techrep/RT_DINF004_2002.pdf>. Acesso em: 6 jun. 2005

Java Media Framework API Guide. Califórnia: Sun Microsystems, 1999. Disponível em: <<http://java.sun.com/products/java-media/jmf/2.1.1/specdownload.html> > Acesso em: 16 Mai. 2007.

JGURU; **Remote Method Invocation**. Califórnia: Sun Microsystem, 2000. Disponível em: < <http://java.sun.com/developer/on-lineTraining/rmi/RMI.html>> Acesso em: 16 Mai. 2007.

JÚNIOR, C.G.B.; *Agregando Frameworks de Infra-Estrutura em uma Arquitetura Baseada em Componentes: Um Estudo de Caso no Ambiente AulaNet*, Dissertação de Mestrado, Pós-Graduação em Informática, PUC-RIO, Rio de Janeiro, RJ, Brasil, 2006.

JÚNIOR, C.N.; *Gerente de Distribuição do Ambiente Xchart em J2EE: Sistemas Reativos Distribuídos na Tecnologia Xchart*, Dissertação de Mestrado, Instituto de Computação, Universidade Estadual de Campinas, Campinas, SP, Brasil, 2005.

LAURENSEN, M.; NINOMIYA, S.; SHEN, Z.; YU, X.; "IMAGESERVER: A System For A Distributed Image Processing Application Based On Java Advanced Imaging". In: *EFITA 2003 Conference*, p. 150-156. Debrecen, Hungary, Jul 2003

LEE, R.; RYAN, V.; SELIGMAN, S.; **RFC2713** Schema for Representing Java (tm) Objects in an LDAP Directory. Sun Microsystems, Inc., 1999. Disponível em: <<http://www.ietf.org/rfc/rfc2713.txt>>. Acesso em: 15 mai. 2007.

LIMA, L.S.; POZZER, C.T.; RAPOSO, A.B.; VIEIRA, C.J.G.; "A Multi-user Videoconference-based Collaboration Tool: Design and Implementation Issues". In: *9th International Conference on Computer Supported Cooperative Work in Design (CSCWiD)*, p. 547-552. Coventry, U.K., May 2005.

KUROSE, J.F.; ROSS, K.W.; *Redes de Computadores e a Internet: Uma Abordagem top-down*, 3 ed. São Paulo, Pearson, 2006.

MOCKAPETRIS, P.; **RFC1034** Domain Names, Concepts and Facilities. ISI, Network Working Group, 1987. Disponível em: <<http://www.ietf.org/rfc/rfc1034.txt>>. Acesso em: 15 mai. 2007.

NETO, G.R.; OLIVEIRA, W.; VALERI, F.V.; "Armazenamento de Imagens Médicas com InterBase". In: *Anais da VI SECICOM*, v. 3.1, pp 13-17, Lavras, 2004

POSTEL, J.; **RFC0768** User Datagram Protocol. ISI, Network Working Group, 1980. Disponível em: <<http://www.ietf.org/rfc/rfc0768.txt>>. Acesso em: 15 mai. 2007.

POSTEL, J.B.; **RFC0821** Simple Mail Transfer Protocol. Califórnia: Information Sciences Institute University of Southern California, 1982. Disponível em: <<http://www.ietf.org/rfc/rfc0821.txt>>. Acesso em: 15 mai. 2007.

POSTEL, J.; REYNOLDS, J.; **RFC0959** File Transfer Protocol. ISI, Network Working Group, 1985. Disponível em: <<http://www.ietf.org/rfc/rfc0959.txt>>. Acesso em: 15 mai. 2007.

Programing in Java Advanced Imaging. Califórnia: Sun Microsystems, 1999. Disponível em: <<http://www.sun.com/products-n-solutions/hardware/docs/pdf/806-5413-10.pdf>> Acesso em: 01 Mai. 2007.

RFC0793; Transmission Control Protocol. Califórnia: Information Sciences Institute University of Southern California, 1981. Disponível em: <<http://www.ietf.org/rfc/rfc0793.txt>>. Acesso em: 15 mai. 2007.

RUTE. Rede Universitária de Telemedicina. Desenvolvido pelo Ministério da Ciência e Tecnolgia, 2007. Apresenta textos e vídeos sobre telemedicina no Brasil. Disponível em <<http://www.rute.rnp.br/videos>>. Acesso em 01 dez. 2006

SERRA, A.B.; *Uma Solução de Distribuição para Aplicações em Tempo Real no Contexto do Ensino Tecnológico à Distância*, Dissertação de Mestrado, Mestrado em Ciência da Computação, Universidade Federal do Ceará, Fortaleza, CE, 2001

TANENBAUM, A.S.; *Redes de Computadores*, 4 ed. Rio de Janeiro, Campus, 2003a.

TANENBAUM, A.S.; *Sistemas Operacionais Modernos*, 2 ed. São Paulo, Pearson, 2003b.

CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA
CELSO SUCKOW DA FONSECA-CEFET/RJ
DIRETORIA DE PESQUISA E PÓS-GRADUAÇÃO
COORDENADORIA DO PROGRAMA DE PÓS-GRADUAÇÃO EM TECNOLOGIA

DISSERTAÇÃO

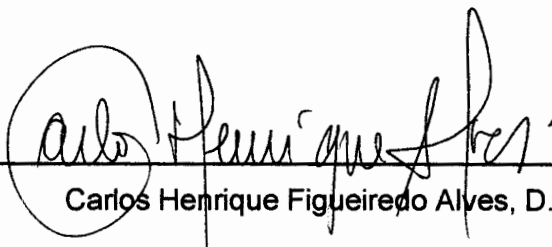
GERENCIAMENTO, ANÁLISE E TRANSMISSÃO DE IMAGENS,
ON-LINE, VIA TCP/IP.

Angelo Márcio de Paula

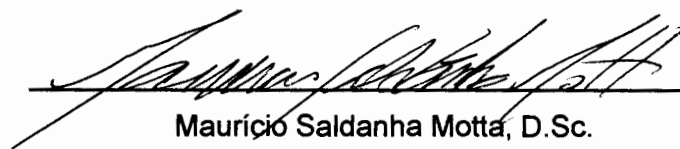
DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DO PROGRAMA DE PÓS-GRADUAÇÃO EM TECNOLOGIA COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM TECNOLOGIA.

Data da defesa: 20/12/2007.

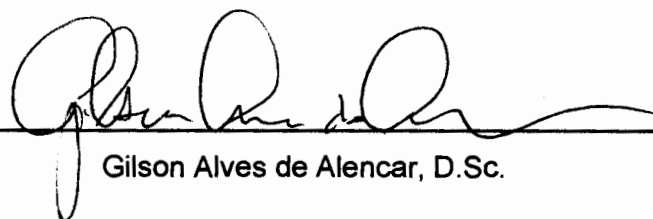
Aprovação:



Carlos Henrique Figueiredo Alves, D.Sc.



Maurício Saldanha Motta, D.Sc.



Gilson Alves de Alencar, D.Sc.

Livros Grátis

(<http://www.livrosgratis.com.br>)

Milhares de Livros para Download:

[Baixar livros de Administração](#)

[Baixar livros de Agronomia](#)

[Baixar livros de Arquitetura](#)

[Baixar livros de Artes](#)

[Baixar livros de Astronomia](#)

[Baixar livros de Biologia Geral](#)

[Baixar livros de Ciência da Computação](#)

[Baixar livros de Ciência da Informação](#)

[Baixar livros de Ciência Política](#)

[Baixar livros de Ciências da Saúde](#)

[Baixar livros de Comunicação](#)

[Baixar livros do Conselho Nacional de Educação - CNE](#)

[Baixar livros de Defesa civil](#)

[Baixar livros de Direito](#)

[Baixar livros de Direitos humanos](#)

[Baixar livros de Economia](#)

[Baixar livros de Economia Doméstica](#)

[Baixar livros de Educação](#)

[Baixar livros de Educação - Trânsito](#)

[Baixar livros de Educação Física](#)

[Baixar livros de Engenharia Aeroespacial](#)

[Baixar livros de Farmácia](#)

[Baixar livros de Filosofia](#)

[Baixar livros de Física](#)

[Baixar livros de Geociências](#)

[Baixar livros de Geografia](#)

[Baixar livros de História](#)

[Baixar livros de Línguas](#)

[Baixar livros de Literatura](#)
[Baixar livros de Literatura de Cordel](#)
[Baixar livros de Literatura Infantil](#)
[Baixar livros de Matemática](#)
[Baixar livros de Medicina](#)
[Baixar livros de Medicina Veterinária](#)
[Baixar livros de Meio Ambiente](#)
[Baixar livros de Meteorologia](#)
[Baixar Monografias e TCC](#)
[Baixar livros Multidisciplinar](#)
[Baixar livros de Música](#)
[Baixar livros de Psicologia](#)
[Baixar livros de Química](#)
[Baixar livros de Saúde Coletiva](#)
[Baixar livros de Serviço Social](#)
[Baixar livros de Sociologia](#)
[Baixar livros de Teologia](#)
[Baixar livros de Trabalho](#)
[Baixar livros de Turismo](#)