STREAMING DE IMAGENS MÉDICAS

Joana Torturella Valadão

TESE SUBMETIDA AO CORPO DOCENTE DA COORDENAÇÃO DOS PROGRAMAS DE PÓS-GRADUAÇÃO DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Aprovada por:

Prof. Antônio Alberto Fernandes de Oliveira, D.Sc.

Prof. Edilberto Strauss, Ph. D

Profa. Aura Conci, D.Sc.

Profa. Fabiana Rodrigues Leta, D.Sc.

RIO DE JANEIRO, RJ – BRASIL

JANEIRO DE 2004

VALADÃO, JOANA TORTURELLA

Streaming de Imagens Médicas [Rio de Janeiro] 2004

X, 65 p. 29,7 cm (COPPE/UFRJ, M.Sc., Engenharia de Sistemas e Computação, 2004)

Tese – Universidade Federal do Rio de Janeiro, COPPE

1. Streaming de Imagens Médicas

2. Padrões Internacionais da Área Médica (HL7, HIS, PACS e DICOM)

I. COPPE/UFRJ II. Título (série)

Dedico esta conquista aos meus pais, irmão e meu marido, por todo o apoio, paciência, e por estarem sempre ao meu lado, sem me deixar desanimar em um só momento. E a Deus, por me guiar em minha jornada.

Agradecimentos

Aos meus orientadores, Edilberto Strauss e Antonio Oliveira, pela orientação, apoio e incentivo.

Ao Marcos pelas nossas conversas, e por acompanhar o desenvolvimento deste trabalho.

Às professoras Aura Conci e Fabiana Rodrigues Leta por aceitarem participar da banca.

Aos professores e alunos do Laboratório de Computação Gráfica (LCG).

À equipe do Hospital Quinta D'or por terem permitido que eu conhecesse um ambiente hospitalar de perto, para desenvolver o meu trabalho.

Ao pessoal administrativo da COPPE/Sistemas.

Ao CNPQ pelo apoio financeiro.

Resumo da Tese apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

STREAMING DE IMAGENS MÉDICAS

Joana Torturella Valadão

Janeiro/2004

Orientador: Antonio Alberto Fernandes de Oliveira

Programa: Engenharia de Sistemas e Computação

A medicina moderna encontra-se baseada em uma gama significativa de modalidades de exames integrados à característica digital. A imagem médica digital transformou-se em elemento essencial na obtenção de um diagnóstico e prognóstico médico mais seguro e padronizado da tecnologia digital sobre os aspectos de armazenamento, transmissão e manipulação das imagens digitais. Para isso, surgem os sistemas PACS (Picture *Archive Communication System*) sobre o padrão internacional DICOM (*Digital Imaging and Communication in Medicine*), normalizados pela ANSI-HL7 (*Health Level Seven*).

Essa dissertação descreve um conjunto de técnicas implementadas para a otimização do armazenamento, do processamento e da visualização de imagens médicas sobre o padrão *web*. O Sistema WebMedViewer (WMV) foi desenvolvido em Java, sobre uma arquitetura em quatro camadas, para melhor atender aos padrões de comunicação Internet/Intranet. A arquitetura implementada garante, a um baixo custo computacional, um acesso remoto a exames médicos, com alta complexidade de informação associada. O Sistema WMV é integrado a um sistema PACS-Hospitalar, seguindo os padrões DICOM (versão 3) descrito da norma HL7, garantindo um processo eficiente de telemedicina.

Abstract of Thesis presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.).

STREAMING OF MEDICAL IMAGES

Joana Torturella Valadão

January/2004

Advisor: Antonio Alberto Fernandes de Oliveira

Department: System and Computing Engineering

Modern Medicine is based on a variety of integrated digital exams. The medical digital image transform itself in an essencial element to obtain a precision and safe medical diagnosis and prognosis. These exams, which are more complex, need a continuous and standardized advance in many technologies, like: storage, transmission and manipulation of digital images.

For that, there are systems like PACS (Picture Archival Communication System), which use an internacional standard named DICOM (Digital Imaging and Communication in Medicine), standardized by ANSI-HL7 (Health Level Seven).

This dissertation describes a set of optimation techniques implemented by storage, processing, and visualizing medical images in *web* form. The system WebMedViewer (WMV) were developed in Java, based on four layers arquitecture, for better attend the internacional intranet and internet standars. The arquitecture implemented, guarantees, a low computacional cost, remote exams acess with complex infomation assotiated. The WMV is integrated with System PACS-Hospital, folowing internacional standars DICOM(version 3) described in HL7, garanting an efficient telemedicine process.

Índice Analítico

Índice de Figuras e Tabelas

Capítulo 1:

Introdução

No mundo atual, as instituições médicas vêm enfrentando uma série de desafios focando principalmente a melhoria na qualidade dos serviços clínicos e hospitalares, sem com isso aumentar demasiadamente os já altos custos operacionais. Para isso, é importante a absorção dos avanços tecnológicos requeridos pela medicina moderna, permitindo o gerenciamento das informações médicas e administrativas de maneira eficiente, tornando o acesso às informações mais rápido e seguro.

Um marco no avanço tecnológico da medicina foi a descoberta do Raio-X, em 1895, que iniciou uma nova fase: o diagnóstico por imagens. Depois do Raio-X, surgiram novas modalidades de obtenção de imagens, como a cintilografia, a ultra-sonografia, a tomografia computadorizada, a ressonância magnética, etc, sendo que a própria imagem originada nos exames evoluiu das imagens em filmes radiológicos para as digitais (REZENDE, 2002). Além dos avanços na área de exames, com a difusão dos computadores, diversos sistemas foram criados a fim de facilitar o gerenciamento das informações médico-administrativas.

Foi neste contexto que surgiram diferentes sistemas computacionais de apoio. Dentre eles o armazenamento de imagens médicas no formato digital (*Picture Archival Communication System* – PACS), o padrão DICOM *(Digital Imaging and Communication in Medicine*) que fornece maior suporte ao PACS, através da padronização da comunicação entre os diversos equipamentos hospitalares, culminando em complexos sistemas de informação (*Hospital Information System* – HIS), responsáveis pela integração de informações de natureza financeira até prontuários médicos. Regendo este complexo sistema hospitalar (o HIS), há ainda o *Health Level Seven* (HL7), que é um padrão que tem como objetivo regulamentar toda a troca de informação entre os diversos sistemas hospitalares.

O armazenamento das imagens em formato digital trouxe uma série de vantagens para o setor médico-hospitalar. Essas vantagens não se resumem apenas na redução de custo dos filmes radiológicos. Mas também, na facilidade de armazenamento, recuperação e manipulação das imagens. Os filmes são "impressos"

apenas no momento em que o exame vai ser entregue ao paciente, evitando desperdícios.

Uma outra importante vantagem da tecnologia digital é facilitar o acesso às imagens, disponibilizando-as na Intranet, bem como na Internet, através de sites seguros, e com acesso restrito, otimizando assim os procedimentos operacionais de médicos e técnicos radiológicos. Porém, a disponibilização destas informações em rede, traz algumas complicações como, por exemplo, a relação entre o tamanho das imagens e sua transferência. Ou seja, para imagens muito grandes (como por exemplo, uma imagem RX digital de mão que possui um tamanho médio de 2500 x 2048 pixels) o tempo de espera associado ao *download* e a visualização é demasiadamente longo.

O problema de arquivos grandes que devem sofrer *download* antes de serem executados é algo comum na Internet. Arquivos de áudio e de vídeo são um bom exemplo. Assim como as imagens médicas, eles devem ser carregados completamente para serem visualizados. O ideal seria disponibilizar a visualização da imagem, ou do vídeo, ou do áudio, durante o *download.* Este objetivo foi atingido com a criação das técnicas de *streaming*[1].

A fim de permitir a visualização do arquivo durante seu *download,* esta técnica promove a segmentação do arquivo, facilitando sua transmissão na rede. Além disso, os segmentos enviados sofrem uma transformação no computador cliente, permitindo a visualização do arquivo. O resultado é a possibilidade de assistir ao vídeo ou ouvir a música, por exemplo, sem longos tempos de espera. Quando se trata de imagens digitais, as partes mais significativas são enviadas primeiro, fornecendo ao usuário uma primeira percepção da imagem, reduzindo a sensação de espera.

Um cuidado que se deve ter ao utilizar técnicas de *streaming* é referente a compressão de dados. A maior parte dessas técnicas realiza a compressão, em geral com perda de informação, a fim de que o arquivo fique menor, e seja mais fácil de transmitir pela rede. Porém, devido à própria natureza das imagens médicas, não é aconselhável o uso de compressões com perda, pois a perda de informações pode prejudicar a análise dos exames.

Existem diversas técnicas de *streaming* para arquivos de áudio e vídeo, porém muito pouco para imagens, e menos ainda para imagens médicas. As que existem, em sua grande maioria, são proprietárias.

---

[1] Streaming é a denominação da técnica que promove a segregação de um arquivo grande, para que ele seja transmitido e seus fragmentos sejam "executados" no computador cliente, em tempo real.

Devido aos altos custos dos sistemas comerciais destinados ao *streaming* de imagens médicas e, em alguns casos, a não concordância com o padrão DICOM, desenvolveu-se um sistema em *tempo real* para a transmissão de imagens médicas através de uma rede de comunicação DICOM (*Digital Imaging and Communications in Medicine*), permitindo que os usuários, utilizando uma rede Intranet/Internet, tenham acesso rápido e seguro aos exames radiológicos (imagens digitais), independente de onde estejam localizados fisicamente, sem a necessidade da instalação de algum *software* especial.

O sistema implementado, baseado em técnicas de *streaming*, também se destina a possibilitar a análise de exames. Desta forma, foram disponibilizadas funções de manipulação para que o usuário consiga simular o que é possível realizar com as imagens em filmes radiológicos.

Esta dissertação está dividida em 5 capítulos, incluindo a presente introdução. Os próximos capítulos abordarão os seguintes assuntos:

- **Capítulo 2 – *Streaming*:** este capítulo apresenta o processo geral envolvido nas técnicas de *streaming*, ressaltando as particularidades necessárias para o *streaming* de imagens médicas, os formatos de arquivo existentes, e algumas técnicas de *streaming* de imagem disponíveis.
- **Capítulo 3 – O Sistema Hospitalar:** apresenta uma visão geral do sistema hospitalar, apresentando as tecnologias envolvidas, como os padrões *Health Level Seven* (HL7), *Digital Imaging and Communication in Medicine* (DICOM), os sistemas *Picture Archival Communication System* (PACS) e o *Hospital Information System* (HIS).
- **Capítulo 4 – O *Software*:** apresenta uma revisão das arquiteturas física e lógica existentes, e a escolhida na implementação do *software*. Descreve a linguagem de programação utilizada, a interface com o usuário desenvolvida, e a técnica de *streaming* implementada.
- **Capítulo 5 – Considerações Finais**: conclui a dissertação fazendo considerações finais a respeito do trabalho, e sugerindo trabalhos futuros.

Capítulo 2:

Streaming

2.1. Introdução

Inicialmente proposta em 1995, a tecnologia de *streaming* vem sendo cada vez mais utilizada na Internet para a disponibilização de arquivos, principalmente, de áudio e vídeo. Porém, para entender a necessidade desta tecnologia, é necessário compreender como funciona o mundo da *World Wide Web* (WWW - *Web*), e suas limitações (BATTISTI, 2002b) (KENNEDY, 1999).

No mundo *web*, quando um servidor recebe uma requisição de informação, ele procura obtê-la e enviá-la ao cliente que a solicitou o mais rápido possível. Após o envio, quando a transação foi completada, o mesmo se "desconecta", e fica esperando por uma nova solicitação. No cliente, o navegador recebe a informação, a exibe na tela, e ignora o servidor *web* até que o usuário acesse um outro *link* (BATTISTI, 2002b) (KENNEDY, 1999).

Esse procedimento é muito eficiente para textos e gráficos, que são informações pequenas e estáticas. Porém, para a transmissão de vídeo e áudio, por exemplo, esta técnica não é eficiente, por exigir o completo *download* do arquivo para que o mesmo seja acessado. Desta forma, para arquivos grandes, é requerido muito tempo de *download* gerando um desconforto para o usuário (BATTISTI, 2002b) (KENNEDY, 1999).

É neste contexto que surge a necessidade de uma nova técnica que seja capaz de exibir o conteúdo solicitado para o usuário, de forma mais rápida. Uma alternativa é iniciar a exibição enquanto o arquivo está sendo carregado. Esta técnica é denominada *streaming* (BATTISTI, 2002b) (KENNEDY, 1999).

Este capítulo apresenta na sua segunda sessão, uma descrição genérica das técnicas de *streaming* e suas classificações. A terceira sessão apresenta os formatos de arquivos disponíveis para imagens digitais e suas principais características. A quarta sessão traz as técnicas de *streaming* para imagens, e um resumo da técnica desenvolvida nesta dissertação.

2.2. A Técnica de *Streaming*

Streaming é um processo em que áudio, vídeo, ou qualquer outro arquivo multimídia, pode ser acessado em tempo real através da Internet ou Intranet. Essa tecnologia é composta de componentes de *hardware* e *software* que trabalham juntos para criar, armazenar e distribuir os arquivos pela *web*, permitindo que múltiplos usuários acessem um arquivo ao mesmo tempo (STREAMINGMEDIA, 2002).

O processo de *streaming* ocorre de forma similar à execução de um arquivo de vídeo ou áudio que se encontra em um CD-ROM (STREAMINGMEDIA, 2002). O arquivo a ser executado é escolhido, e então se inicia a transferência de partes do arquivo que serão executadas. Essas "partes" vão sendo exibidas, enquanto as seguintes são transferidas, até que todo o arquivo tenha sido exibido.

É importante observar que existem diferenças entre os dois processos. Em geral, durante o *streaming*, o arquivo sofre uma compressão, sendo convertido para um formato especial, o que não ocorre no caso de CD-ROM's. Além disso, ele fica armazenado em um servidor remoto, sendo acessado através da Internet/Intranet.

No *streaming*, estão envolvidos diversos passos, que vão desde a aquisição do arquivo, até a exibição do mesmo no computador cliente. Durante a aquisição do arquivo, seja ele áudio, vídeo, ou ambos, a qualidade é um ponto fundamental, pois como tem-se compressão com perda, a mesma é reduzida. Logo, se o arquivo a ser transmitido estiver ruim, ele chegará ainda pior para o usuário. Caso o arquivo não esteja no formato digital, é necessário ainda convertê-lo, para que possa ser transmitido.

Após a aquisição do que se deseja transmitir pela Internet/Intranet, é necessário promover a compressão do arquivo, o que, em geral, já o converte para um formato especial destinado a transmissão em rede. A partir daí, ele é transmitido, de forma segmentada. O computador cliente se encarrega de receber o arquivo segmentado e convertê-lo, transformando os segmentos, para um formato que permita sua exibição.

De uma forma geral, este é o processo envolvido na tecnologia *streaming*. Porém, algumas diferenças e particularidades, envolvidas neste processo, permitem a classificação das técnicas em dois grupos, que são:

- ***The True Streaming***: onde pequenas porções do arquivo são enviadas continuamente pela rede, enquanto ele é exibido;

- ***The Pseudo Streaming*** ou ***The Progressive Download***: realiza o *download* de uma parte significativa do arquivo, antes que a visualização inicie.

Além dessa divisão, as técnicas também podem ser classificadas por utilizarem, ou não, um servidor de *Streaming* (HUNTER et all, 1997). As que não possuem um servidor utilizam o próprio protocolo HTTP, e seu servidor, para transmitir o arquivo. As vantagens de não haver um servidor de *Streaming* são que há um *software* a menos para se aprender, e um servidor a menos para se gerenciar, o que acaba tornando a solução economicamente mais barata.

Por outro lado, a existência do servidor torna mais eficiente a utilização da rede, além de: oferecer um arquivo de melhor qualidade ao cliente; suportar simultaneamente um maior número de usuários; etc. *Softwares* comerciais como o *Real Player*, o *StreamWorks* e *VDOnet's VDOLive* são exemplos de produtos de *streaming* de vídeo que utilizam servidor. Já o *Shockwave* e o *VivoActive* são exemplos dos que adotam a solução sem servidor (HUNTER et all, 1997).

A técnica de *streaming* tem sido amplamente utilizada na atualidade, e não só para assistir "filmes", ou ouvir música na Internet. Muitas empresas estão utilizando esta técnica para a comunicação *Business-to-Business* (B2B), ensino a distância, etc.

O fato de a técnica de *streaming* permitir a visualização de um arquivo mesmo antes de seu *download* ter sido concluído, a torna muito interessante e atrativa para ser utilizada na área médica.

As imagens dos exames médicos são grandes, algumas chegando a ter mais de 10MBytes. Isso torna sua visualização através da rede, seja Intranet ou Internet, algo muito custoso e demorado. O objetivo da técnica de *streaming*, aplicada à área médica, é tornar esta visualização mais rápida para seus usuários.

Porém, como descrito anteriormente, o processo de *streaming*, em geral, envolve a compressão com perda do arquivo para reduzir seu tamanho, e tornar mais fácil e rápida a sua transmissão. Sendo assim, há uma perda na qualidade do arquivo (BATTISTI, 2002b). Este é um grande problema, quando se trata de imagens de exames médicos, pois a perda de qualidade pode resultar em uma análise errada do exame, ou até mesmo impedi-la. Ou seja, é necessário o desenvolvimento de uma técnica de *streaming* sem compressão com perda para a utilização na área médica.

2.3. Formatos de Arquivos

Por trabalhar com o arquivo, segmentando-o para transmiti-lo na rede, é necessário conhecer o arquivo, seu formato, e seus metadados, para se criar uma eficiente técnica de *streaming*.

Durante os últimos anos, diversos formatos de imagens foram criados, em sua maioria por fabricantes de *hardware* e *software*. Desta forma, gerou-se um problema: não é possível visualizar a imagem em qualquer computador e/ou plataforma que se deseje (KODAK, 2003).

Com a grande demanda de aplicações, os fabricantes passaram a criar formatos mais compatíveis com as diversas plataformas, e com os diferentes fornecedores de produtos. Com isso, é possível salvar ou abrir imagens, não só no formato proprietário, mas também em formatos mais genéricos, como GIF, JPEG, dentre outros.

2.3.1. *Tag Image File Format* – TIFF

Desenvolvido por Aldus Corporation em 1986, e comprado pela Acrobat, este formato de arquivo foi criado especialmente para salvar imagens de *scanners*, programas de retoque, etc. O **TIFF** talvez seja um dos mais versáteis e confiáveis formatos, sendo suportado em diversas plataformas (KODAK, 2003)(WAYNE, 2003).

Ele escreve em arquivos grandes, utilizando esquemas de compressão com ou sem perda. A desvantagem é que o arquivo se torna grande, sendo assim, não é muito apropriado para a transmissão através da Internet (WAYNE, 2003).

O TIFF é um padrão quase universal, sendo utilizado por diversos fabricantes de *scanners*, por exemplo. Porém, ele possui uma série de variações, como o utilizado para o envio de documentos através do fax. Isso compromete sua portabilidade, pois é possível que um computador não seja capaz de abrir todas as variações deste formato de arquivo (KODAK, 2003) (WAYNE, 2003).

2.3.2. PICT

Este formato, nativo do Macintosh, surgiu em 1984 e é muito utilizado em páginas de *layout*, e programas gráficos. Assim como o TIFF, ele apresenta compressão

sem perda, porém não pode ser visualizado por programas "Windows" (KODAK, 2003) (WAYNE, 2003), comprometendo sua portabilidade.

2.3.3. *Encapsulated PostScript* – EPS

Criado na década de 80, este é o formato utilizado para armazenar imagens de alta resolução em arquivos *PostScript*, podendo ser utilizado por usuários de Windows e Macintosh (KODAK, 2003).

O EPS possui, em geral, duas partes: texto, que descreve como a imagem deve ser gerada; e, opcionalmente, uma imagem no formato PICT para visualização antes da impressão.

2.3.4. *Graphics Interchange Format* – GIF

Desenvolvido em 1987 pela *CompuServe*, é um dos formatos de arquivo de maior sucesso, pela compressão alcançada (KODAK, 2003)(WAYNE, 2003). O GIF define um protocolo para transmissão e visualização, tornado-o mais independente da plataforma (KODAK, 2003).

O arquivo GIF é definido em termos de blocos e sub-blocos, que possuem as informações necessárias para a montagem da imagem. Ele possui uma paleta de 256 cores (que é o número máximo de cores que uma imagem neste formato pode ter), sendo que essas 256 cores podem ser quaisquer cores, não havendo necessidade de um padrão, ou algo semelhante, para a escolha delas (KODAK, 2003) (WAYNE, 2003). A compressão usada é a LZW, que é uma técnica de compressão sem perda.

2.3.5. *Joint Photographic Experts Group* – JPEG

O formato de arquivo JPEG é um dos mais eficientes no que diz respeito à redução do tamanho da imagem. Porém, isso tem um preço. Para conseguir esta redução, é usada compressão com perda, o que sacrifica a qualidade, que não tem como ser recuperada. Essa perda ocorre toda vez que se salva o arquivo, ocasionando a perda de bits (dados), piorando a qualidade da imagem. Por isso, não é aconselhável armazenar os arquivos originais em JPEG (WAYNE, 2003).

É possível utilizar o JPEG em imagens coloridas, ou em tons de cinza, e até mesmo para imagens reais, como fotografias, por exemplo. Ele é muito utilizado na transmissão de imagens, e em sites *web*, devido a seu tamanho reduzido (KODAK, 2003).

Existe também o JPEG2000 que é uma evolução do formato JPEG. Ele pode ser encarado como um mecanismo padronizado de compressão de imagem. O JPEG2000 pode reduzir um arquivo, utilizando compressão sem perda, de 69Mb para 11.8Mb, por exemplo. Além disso, ele preserva a transparência da imagem, e é um formato eficiente para armazenar imagens de alta qualidade (MILBURN, 2003).

2.3.6. *Windows Bitmap* – BMP

Geralmente, este formato não possui perda de informação. Com isso, seus arquivos se tornam muito grandes. Além disso, eles só podem ser visualizados nos computadores que possuem os sistema operacional Windows, prejudicando sua portabilidade (WAYNE, 2003).

2.3.7. *Digital Imaging and Communication in Medicine* – DICOM

O DICOM não é um formato de arquivo propriamente dito. Usado na área médica, ele é um padrão que especifica como as informações da imagem devem estar, e como elas devem ser transmitidas entre os diversos equipamentos hospitalares (HORII, 2002).

Este formato foi desenvolvido diante da necessidade de se ter um padrão internacional que fosse aceito por todos os fabricantes de equipamentos hospitalares, para a unificação da comunicação da informação médica sem a dependência de programas de interface. O capítulo 3 apresentará maiores detalhes deste padrão.

É importante ser observado que o formato de imagem utilizado na implementação deste trabalho é o DICOM, por estar em conformidade com o padrão internacional estabelecido pelo HL7 (*Health Level Seven*). É sobre ele que foi implementada a técnica de *streaming* apresentada nesta dissertação.

2.4. *Streaming* de Imagens Médicas

As técnicas de *streaming* de imagens, disponíveis hoje são proprietárias. Sendo assim, não se tem muita informação sobre como o processo de *streaming* é realizado.

Algumas empresas, como a *Zoomify* e *RealTime Image*, possuem famílias de *softwares* que permitem a transmissão de imagens grandes, em tempo real, e com pequeno tempo de espera.

2.4.1. Zoomifyer

A empresa *Zoomify* possui uma série de *softwares* que realizam o *streaming* de imagens. Alguns deles, servindo como "*plug-ins*" para outros *softwares* (como por exemplo o *Flash* da *Macromedia*), permitem acoplar imagens grandes a filmes e animações (ZOOMIFY, 2003).

Para a realização do *streaming*, a imagem é copiada diversas vezes, com resoluções diferentes. Depois, as imagens são "cortadas" em pedaços menores, que são guardados em um arquivo juntamente com um índice, que indica a localização exata de cada pedaço. Desta forma, tem-se uma pirâmide de resolução, como exemplifica a figura 2.1(ZOOMIFY, 2003).

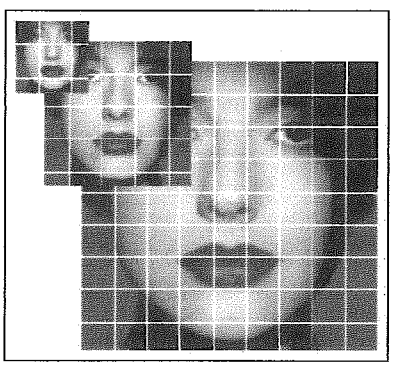

Figura 2.1: Arquivo de imagem do Zoomify

Quando a imagem é visualizada pela primeira vez, a imagem de menor resolução na pirâmide é envidada para o usuário. À medida que ele solicita o zoom, apenas os pedaços necessários para completar a imagem são enviados, reduzindo o tempo. O envio da imagem é feito utilizando-se o protocolo HTTP (ZOOMIFY, 2003).

A figura 2.2 (ZOOMIFY, 2003) exemplifica o resultado de um zoom em uma imagem.

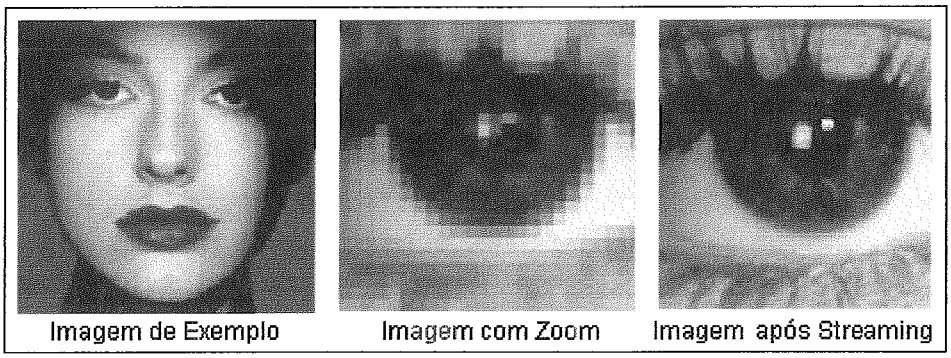

Figura 2.2: Resultado do *Streaming*

2.4.2. iPACS

Desenvolvido pela *RealTime Image* (que tem como parceira a empresa *Wuestec*), o iPACS realiza a transmissão de imagens médicas, através de técnicas de *streaming*. Utilizando-se de um PC padrão, uma conexão a Internet e um navegador (no lado do cliente), ele possibilita o acesso do usuário a imagens de exames médicos, em tempo real, sem a necessidade de grandes tempos de *download* (REALTIME IMAGE, 2003).

A técnica de *streaming* utilizada pelo iPACS é denominada *Pixels-on-Demand®*. Ela permite a transmissão da imagem em tempo real, sem perda de qualidade e sem necessidade de duplicação dos arquivos de imagem. Ela foi desenvolvida pensando na forma como o olho humano processa a informação. As partes mais importantes e relevantes da imagem são enviadas primeiro, permitindo a visualização de uma parte significativa da imagem em pouco tempo (REALTIME IMAGE, 2003).

Além da possibilidade de o usuário observar a imagem enquanto o *download* está sendo realizado, é possível utilizar o modo "*Oncall*". Esta outra opção de visualização disponibiliza as imagens para visualização e manipulação somente quando o *download* foi completamente realizado (REALTIME IMAGE, 2003).

A integração do *iPACS* com o sistema hospitalar já existente é fácil, já que ele pode ser integrado facilmente a rede DICOM, ou mesmo diretamente a uma rede LAN (*Local Area Network*) de conexão rápida (REALTIME IMAGE, 2003).

2.4.3. Técnica de *Streaming* Implementada

A técnica de *streaming* implementada, de acordo com a classificação das técnicas apresentada anteriormente, é sem servidor, pois utiliza o próprio protocolo HTTP para transmitir a informação, e é *True Streaming*, já que a imagem é exibida a partir do primeiro "pacote" de informação que chega no cliente.

Para o *streaming* das imagens, utilizou-se a técnica de histograma, combinada com técnicas de endereço, subtração e redução.

O histograma representa, para a imagem, quantos pixels cada nível de intensidade (tons de cinza) a imagem possui (STREAMINGMEDIA, 2002). Servindo como instrumento que determina as cores mais relevantes da imagem, o histograma fornece subsídios para a ordenação do envio das cores, da mais incidente para a menos, permitindo uma percepção visual mais rápida da imagem.

A técnica de endereço é uma compressão sem perda. Se houver uma seqüência de pixels com a mesma intensidade, apenas o primeiro e o último pixels (seus endereços) serão enviados, reduzindo a quantidade de informação a ser transmitida. Essa técnica é uma variação da técnica de compressão RLE (*Run Length Encoding*).

A técnica de subtração consiste em uma comparação entre duas imagens consecutivas de uma série, para que sejam enviados apenas os pixels diferentes. Esse processo ocorre da seguinte forma: a partir da segunda imagem da série a ser transmitida, verifica-se o que esta imagem tem de diferente da anterior, e transmite-se apenas as diferenças, evitando o reenvio das partes iguais.

A técnica de redução consiste em diminuir o tamanho da imagem (tanto em largura quanto em altura) por um determinado fator, e enviar apenas parte dos pixels, de forma que a imagem não fique deformada, e não aparente perda de qualidade. Essa técnica é utilizada apenas na lista de imagens que serve para o usuário escolher com qual efetivamente irá trabalhar. Após essa escolha, a imagem é completada, não havendo nenhuma perda de informação.

Ao passar pelas técnicas apresentadas anteriormente, a imagem é discretizada em seu espaço de cor (tons de cinza), para que as partes mais significativas sejam enviadas primeiro. Assim, as cores que estão em maior número (pixels) são primeiramente enviadas. Ao chegar no cliente, a imagem é facilmente "remontada" e exibida para o usuário, sem a necessidade de filtros específicos.

O processo de *streaming* desenvolvido será detalhado no capítulo 4, onde serão apresentadas a arquitetura do *software*, e toda a parte técnica envolvida.

Para compreender melhor o processo de *streaming* das imagens médicas, e as dificuldades envolvidas, é necessário conhecer o ambiente onde isso ocorre. O próximo capítulo visa fornecer uma visão geral do ambiente hospitalar.

Capítulo 3

O Sistema Hospitalar

3.1. Introdução

A área médica vem enfrentando um grande desafio: tornar o atendimento ao paciente cada vez melhor e mais eficiente, sem aumentar demais seus custos (BATTISTI, 2002a). Com isso, aumentou-se o foco no controle de custos, tornando cada vez mais importante o controle das informações hospitalares.

A informação de um hospital envolve as especialidades médicas, histórico dos pacientes, dados laboratoriais, dados financeiros, etc, tornando a quantidade de documentos e informações grande demais para se trabalhar e gerenciar. Por isso, um diferencial para clínicas e hospitais é a capacidade de trabalhar de forma mais rápida e eficiente com estas informações (BATTISTI, 2002a).

Diante de todos estes desafios, os hospitais e clínicas passaram a utilizar a tecnologia para armazenar seus dados, no computador em formato digital, sejam eles administrativos ou prontuários de pacientes. Sendo assim, surgiram vários sistemas para controlar todo este conjunto de informações. Estes sistemas auxiliam muito a tarefa na área em que atuam, mas não se comunicam (BATTISTI, 2002a), e o ideal é que as informações estejam integradas, compondo um sistema único, regido por um padrão internacional que regulamente esta integração.

Para suprir esta necessidade, começaram a surgir sistemas capazes de integrar as informações hospitalares, *Hospital Information System* (HIS – Sistema de Informação Hospitalar), exigindo o surgimento de um padrão internacional para integração da informação médica, o *Health Level Seven* (HL7).

Neste capítulo, a sessão 3.2 apresenta o padrão *Health Level Seven* (HL7) que regulamenta toda a troca de informação entre os sistemas médicos. A sessão 3.3 apresenta os sistemas HIS, que são responsáveis pela integração das informações hospitalares. A sessão seguinte apresenta o sistema *Picture Archival Communication System* (PACS), que é responsável pela aquisição, visualização e armazenamento das imagens médicas digitais. Em seguida, na sessão 3.5, é apresentado o padrão internacional *Digital Imaging and Communication in Medicine* (DICOM), que é

responsável pela comunicação das imagens propriamente dita. É este padrão que dá suporte ao PACS.

3.2. *Health Level Seven* – HL7

O *Health Level Seven* (HL7) é uma organização voluntária e sem fins lucrativos da *American National Standards Institute* (ANSI) que desenvolve normas e padrões para a área de saúde. Sua missão é fornecer padrões para a troca, gerenciamento e integração de dados clínicos e administrativos do sistema hospitalar (HL7 ORG, 2003) (GUALBERTO et al., 2003).

A organização HL7 é composta por provedores, vendedores, grupos governamentais, e pessoas que têm interesse no desenvolvimento de padrões para a área médica. Os membros formam os chamados Grupos de Trabalho, e estão organizados em comitês técnicos, e grupos de interesses especiais. Os comitês técnicos são responsáveis pelo padrão em si. Os grupos de interesses especiais, são responsáveis por explorar novas áreas que devem ser abrangidas pelo HL7 (HL7 ORG, 2003) (GUALBERTO et al., 2003).

O "Nível Sete" do nome desta organização vêm da camada de mais alto nível do modelo ISO-OSI[1] de comunicação de uma rede: a camada de aplicação. O sétimo nível suporta funções como segurança, identificação, mecanismos de negociação e estrutura de troca de dados (HL7 ORG, 2003).

Existem diversos padrões para a área de saúde, porém o mais utilizado e difundido é o HL7, pois, ao contrário de outros padrões, trata de todas as informações presentes na área médica, e não de um departamento em particular (HL7 ORG, 2003).

O padrão HL7 possui uma série de trabalhos e iniciativas, como, por exemplo, (HL7 ORG, 2003):

- o ***Reference Information Model*** **(RIM)**: que é um ponto fundamental do padrão. Este modelo é uma representação dos dados clínicos (domínios), identificando os ciclos de vida dos eventos que as mensagens, ou os grupos de mensagens relacionadas, irão transportar.

[1] Este modelo será melhor explicado na sessão 3.5.2, que trata da forma de Comunicação do padrão DICOM.

- os ***Templates***: que são uma estrutura de dados, baseada no RIM, que determina os dados necessários em uma área clínica específica, ou em um contexto administrativo.
- o **Vocabulário**: que é um padrão que determina o significado dos dados codificados. Muitas vezes, os dados transmitidos são codificados, o que os torna inúteis se não houver um padrão bem definido que permita que ambos os lados (transmissor e receptor) conheçam o significado exato, sem ambigüidade dos dados. O Vocabulário permite a troca de informações clínicas e administrativas entre sistemas diferentes.
- o **XML**: a tecnologia XML (Extensible Markup Language) foi desenvolvida para descrever dados. Esta linguagem é muito semelhante ao HTML, porém suas tags não são pré-definidas. O XML é utilizado no HL7 para toda troca de informação entre sistemas (HL7 ORG, 2003).

O sucesso das versões 2.x do HL7 é muito atribuído à sua flexibilidade. Essas versões contêm muitos elementos e segmentos de dados opcionais, tornando-o muito adaptável. Porém, essa adaptabilidade traz um problema, pois torna quase impossível a criação de um grupo de testes único para todas as implementações, forçando os implementadores a gastar mais tempo analisando, planejando e testando as interfaces entre os sistemas, para garantir a comunicação (HL7 ORG, 2003).

A versão 3 deste padrão traz uma metodologia bem definida, baseada no *Reference Information Model* (RIM), ou seja, em dados. Usando uma rigorosa análise e técnicas de construção de mensagens, e incorporando mais eventos de *trigger* e formatos de mensagens com poucos pontos opcionais, esta versão traz a vantagem de fornecer um padrão mais fácil de testar, além de permitir a certificação a fornecedores. Esta versão utiliza a metodologia de desenvolvimento orientado a objeto (HL7 ORG, 2003).

3.3. *Hospital Information System* (HIS)

Em uma clínica ou hospital, é comum a existência de sistemas para tratar de departamentos, como radiologia (RIS – *Radiology Information System*) e farmácia (PIS - *Pharmacy Information System*); sistemas clínicos (CIS – *Clinical Information System*) que são centrados nos pacientes e nos procedimentos médicos; além de sistemas financeiros (FIS – *Financial Information System*) (GANNOT, 2003).

Porém, seria muito mais útil se todos esses sistemas fossem integrados, permitindo a troca de informação entre eles. Esse é um sistema HIS ideal, que integra informações tanto financeiras como clínicas, auxiliando na administração das transações diárias do hospital (GANNOT, 2003).

O HIS compõe o núcleo central de toda informação hospitalar, abrangendo desde a área administrativa, como contabilidade e recursos humanos, por exemplo, até a área de prontuário de pacientes, o setor laboratorial, radiologia, e demais áreas do hospital. A figura 3.1 ilustra este sistema.

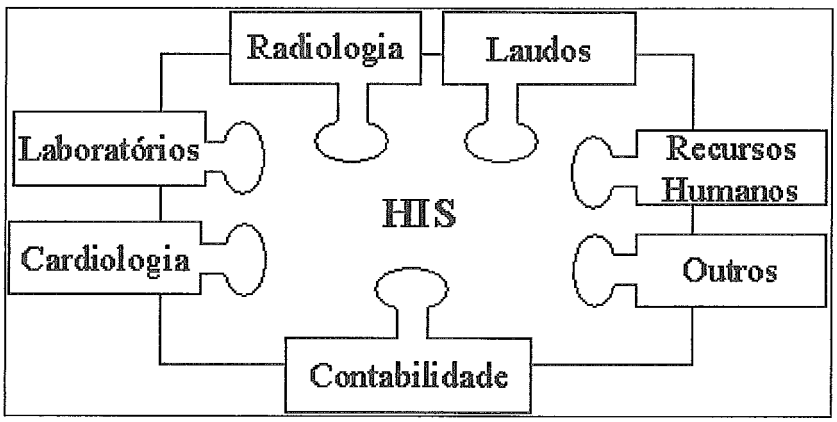

Figura 3.1: O Sistema de Informação Hospitalar (HIS)

Uma das informações geridas por esse sistema são as informações digitais. Dentre elas, destacam-se as imagens digitais geradas em exames de Raio-X, Ressonância Magnética, etc.

A grande quantidade de imagens digitais produzidas para diagnóstico tornou complicada a sua manipulação e seu armazenamento. Por isso, tornou-se necessária a criação de um mecanismo de manipulação e armazenamento dessas imagens em dispositivos conectados em rede, levando em conta a facilidade, rapidez, segurança e o acesso a imagens de qualidade.

3.4. *Picture Archival Communication System* – PACS

Para atender a essa necessidade da área médica, foi criado o PACS, que possui as funcionalidades de gerenciar: a aquisição de imagens; o armazenamento de informação; a distribuição e visualização de imagens (consulta, interpretação ou diagnóstico); o registro de resultados (laudos); a interface com outros sistemas; a comunicação remota e a segurança do sistema (MARTÍNEZ, et all, 2003).

Algumas das vantagens do PACS estão relacionadas a uma percepção de centralização das imagens, pelo usuário, mesmo que elas estejam em bancos de dados distribuídos (*Mini PACS*). Além disso, o sistema garante a segurança e preservação da informação médica, por possuir uma camada de *software* controlando o acesso de múltiplos usuários às imagens.

Para permitir que o PACS tenha todas estas funcionalidades, é necessária uma série de equipamentos trabalhando em conjunto, cada um com uma funcionalidade específica. A figura 3.2 ilustra os componentes de um sistema PACS (MARTÍNEZ, et all, 2003).

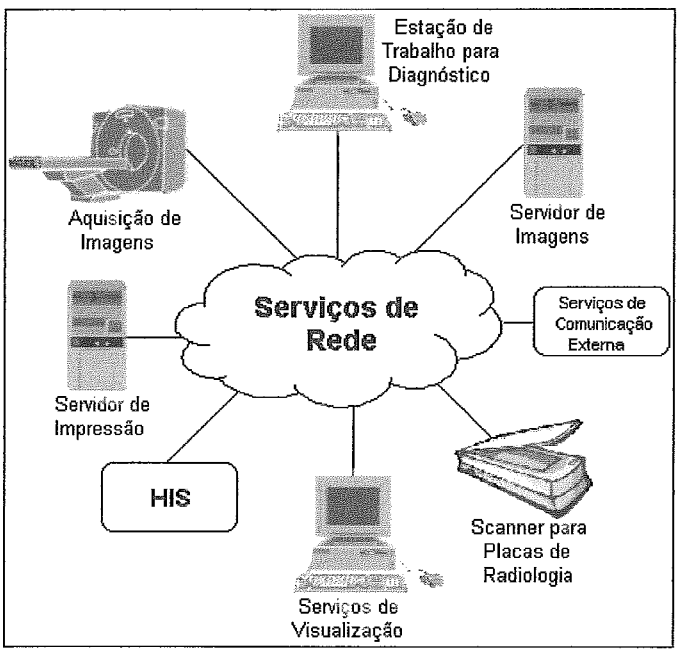

Figura 3.2: Componentes de um sistema PACS

Porém, há um outro problema a ser resolvido. Existem diversos fabricantes de dispositivos de aquisição de imagens, de componentes de rede, além de diversos formatos de imagem. Desta forma, tornou-se necessária a criação de um padrão de comunicação e armazenamento destas imagens, para que os dispositivos dos diversos fabricantes pudessem trabalhar em conjunto, sem a necessidade de programas de interface (API – Application Program Interface) entre eles (RICKARDS, 2003). Este padrão, conhecido como DICOM, faz parte do PACS, e fornece todo o suporte de rede às suas atividades.

3.5. *Digital Imaging and Communication in Medicine* – DICOM

O primeiro padrão internacional criado para a comunicação de imagens foi o *American College of Radiology – National Electrical Manufacturers' Association* (ACR-NEMA), para uma rede ponto-a-ponto. Porém, com a rápida evolução das redes de computadores, uma comunicação ponto-a-ponto tornou-se limitada. Por isso, o padrão ACR-NEMA foi redesenhado, de forma a abranger as novas tecnologias de rede. O resultado deste trabalho foi o *Digital Imaging and Communication in Medicine* (DICOM) (HORII, 2002).

O padrão DICOM contém uma especificação detalhada que descreve: o "formato" da imagem; suas informações agregadas; e como ela deve ser transmitida entre os equipamentos hospitalares (RICKARDS, 2003). Desta forma, o DICOM facilita a interoperabilidade dos equipamentos de imagens médicas ao especificar um conjunto de protocolos que devem ser seguidos, a sintaxe e a semântica dos comandos, e as informações que devem ser fornecidas nas implementações dos equipamentos que estão em conformidade com o padrão (LEAD TECHNOLOGIES, 2003b). Além disso, ele é flexível na definição de novos serviços (MARTÍNEZ, et all, 2003).

Por ser um padrão extremamente adaptável, o DICOM deixou de ser utilizado apenas em radiologia, passando a ser adotado em diversas áreas que geram imagens, como patologia, endoscopia, etc (HORII, 2002). A figura 3.3 abaixo exibe o escopo do DICOM (LEAD TECHNOLOGIES, 2003b):

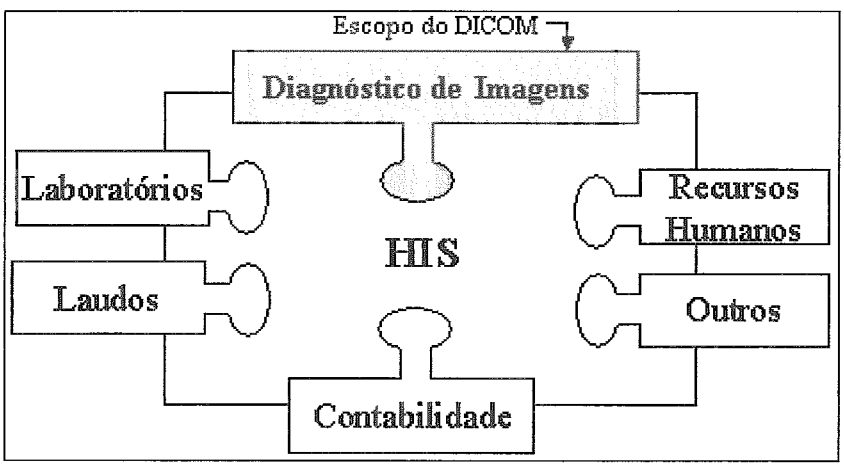

Figura 3.3: Escopo do padrão DICOM

Atualmente a maior parte dos equipamentos médicos é compatível com este padrão sendo ele utilizado em diversos países. Na Europa, a organização de padrões

(Comitâ Europâen de Normalisation) o utiliza como base para o MEDICOM. No Japão, a "*Japanese Industry Association of Radiation Apparatus*" e o Centro de Desenvolvimento de Sistemas de Informação Médica estão adotando o DICOM nos seus padrões de processamento de imagens médicas (HORII, 2002) (RICKARDS, 2003).

3.5.1. Imagens DICOM

Para que a imagem seja transferida, é necessário que ela tenha um formato predeterminado, descrito pelo padrão. Com isso, todas as imagens que seguem esse padrão são denominadas como sendo do formato DICOM (RORDEN, 2002).

Uma imagem DICOM é composta por um cabeçalho, que contém informações sobre o paciente e sobre o exame, pela imagem em si (que pode conter informações em três dimensões, ou seja imagens tri-dimensionais) (RORDEN, 2002)(LEAD TECHONOLOGIES, 2003a), e por um prefixo de 4 bytes "DICM" (LEAD TECHONOLOGIES, 2003a).

A figura 3.4 ilustra como seria um arquivo de imagem DICOM:

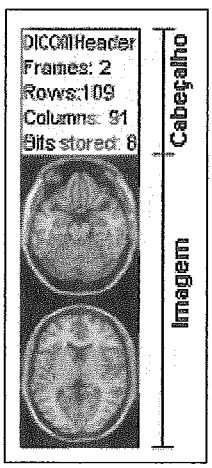

Figura 3.4: Ilustrando um arquivo DICOM

Como ilustrado acima, o cabeçalho armazena informações sobre a imagem, como as dimensões, por exemplo. Além disso, pode armazenar informações sobre o exame, como a data da realização, o dispositivo utilizado, o médico que solicitou, etc. O cabeçalho armazena também, informações sobre o paciente, como nome, data de nascimento, sexo, etc.

A figura 3.5 abaixo exibe um exemplo de um cabeçalho e sua imagem DICOM:

```
bitsAllocated .......16
bitsStored .......12
highBit .......11
signed .......0
 ImageType :::          ORIGINAL\PRIMARY\AXIAL\VOLUME
 studyDate :::          20011115
 modality :::           CT
 manufacturer :::       Philips Medical Systems
 Institution :::        Hospital
 physician  :::
 patientName :::        paciente
 patientID :::          5013979
 patientBirthdate ::: 19310128
 sex              :::   M
 samplesPerPixel :::   1
 h :::                 512
 w :::                 512
 bitsAllocated :::     16
 bitsStored :::        12
 highBit :::           11
 signed :::            0
 size1 :::             524288
```

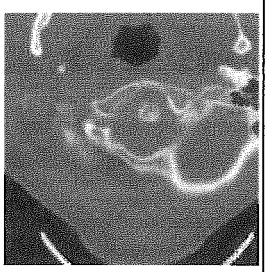

Figura 3.5: Detalhando um arquivo DICOM

Não é obrigatório que o cabeçalho, que possui o tamanho de 128 bytes, esteja preenchido por informações. Ele pode estar "vazio", preenchido apenas pelo valor "00H", que indica que ele não possui informação (LEAD TECHONOLOGIES, 2003a).

A definição do formato das imagens é apenas um dos pontos do padrão DICOM. Ele descreve também como deve ocorrer a transmissão destas imagens.

3.5.2. Comunicação

Como já foi dito, o objetivo do padrão DICOM é descrever um conjunto de regras uniformes e bem compreendidas para a comunicação de imagens médicas (HORII, 2002).

Em geral, quando se fala de troca de informação em computação, logo se imagina um modelo em camadas, cada uma desenvolvendo uma tarefa específica. Na verdade, este modelo é parte de um padrão internacional de comunicação denominado *International Standards Organization Open System Interconnection* (ISO-OSI) *Reference Model*. A figura 3.6 ilustra esse modelo. Como se observa, a camada mais inferior (Física) é a que realiza a conexão com o meio físico, enquanto a superior (Aplicação) é a que realiza a interface com o usuário. As camadas intermediárias são responsáveis, por exemplo, pelos protocolos de comunicação, pela correção de erros, pela codificação/decodificação da mensagem.

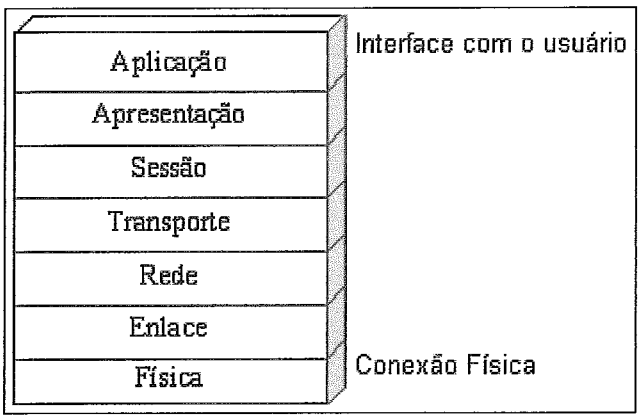

Figura 3.6: Modelo de comunicação

Cada camada recebe uma entrada da camada superior, executa uma determinada tarefa, e fornece o resultado para a camada inferior. O fluxo pode ocorrer também no sentido oposto, já que a comunicação é bidirecional.

Os protocolos das camadas superiores, definidos pelo DICOM, são os próprios protocolos do modelo ISO-OSI, o que garante a comunicação entre aplicações DICOM de forma eficiente e coordenada (LEAD TECHNOLOGIES, 2003b).

A vantagem do modelo de camadas é que uma camada pode ser substituída sem afetar as demais (HORII, 2002), desde que as informações de entradas e saídas sejam respeitadas.

O padrão DICOM utiliza este modelo em camadas para a comunicação e transmissão das imagens, ao invés de criar um modelo próprio. O modelo OSI é amplamente utilizado no mundo da computação, o que traz vantagens significativas no uso do DICOM, pois se torna desnecessária a criação ou adaptação de equipamentos de rede já existentes, bem como a alteração da infra-estrutura de rede hospitalar para transmissão destas imagens.

Além do modelo OSI, o padrão DICOM suporta a comunicação através do protocolo TCP/IP, facilitando mais ainda sua utilização (LEAD TECHNOLOGIES, 2003b). Essa possibilidade de se ter comunicação via ISO-OSI e TCP/IP, permite a migração de um tipo de comunicação para outro, sem impacto na aplicação. A figura 3.7 ilustra a comunicação do DICOM através do protocolo TCP/IP, pelo modelo ISO-OSI, e pela comunicação ponto-a-ponto, a primeira implementada pelo DICOM.

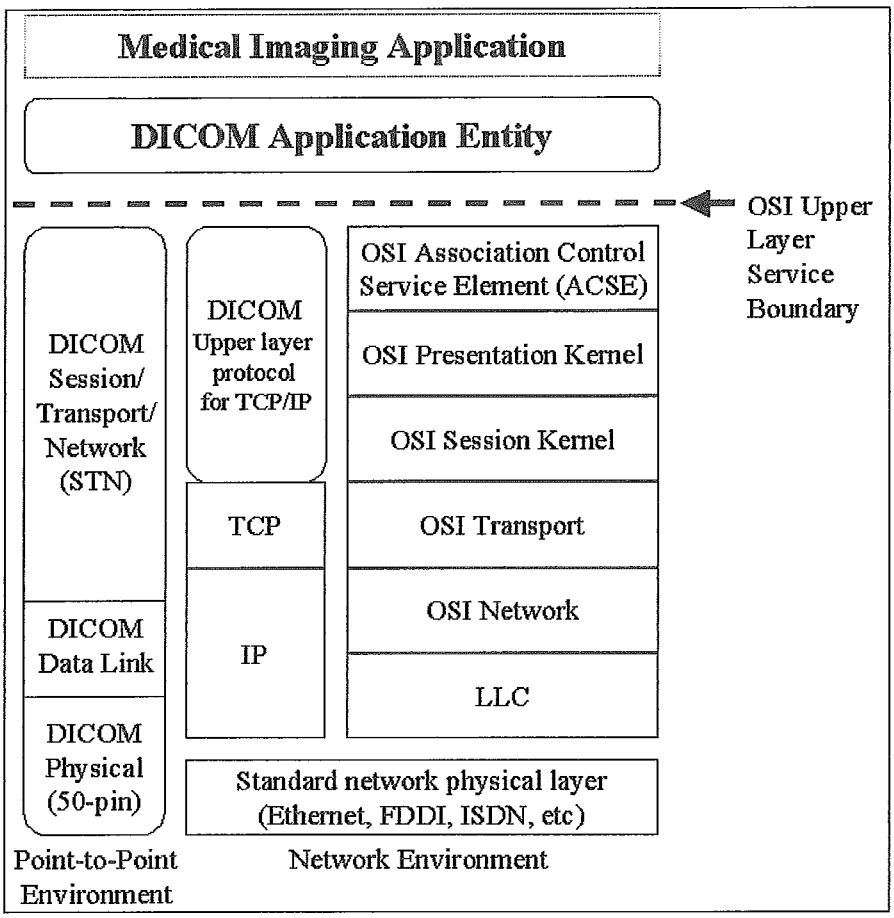

Figura 3.7: Arquitetura de Comunicação do DICOM

A forma como o padrão DICOM especifica os protocolos e a interface do meio físico, permite que um componente que possui uma interface de rede ponto-a-ponto se comunique com uma rede que utiliza o modelo OSI, sendo necessário apenas uma unidade de interface de rede (*Network Interface Unit* – NIU). A figura 3.8 ilustra como ocorre esta conexão.

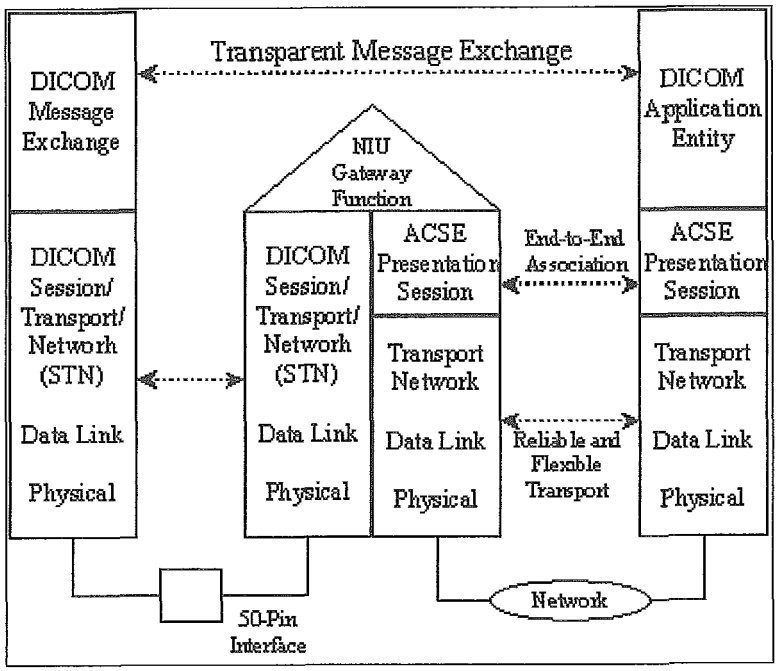

Figura 3.8: Comunicação Ponto-a-Ponto com OSI

Porém, o DICOM não determina somente o formato da imagem, e como enviá-la em uma rede de computadores. Este padrão especifica também os objetos que estão envolvidos em todo o processo, bem como a relação entre eles.

3.5.3. O Modelo de Dados

O modelo de dados fornecido pelo DICOM não é composto apenas de tabelas. Ele explicita todo o relacionamento, a interação e a hierarquia entre os objetos. A figura 3.9 apresenta o modelo de dados completo. É importante observar que este modelo segue o paradigma da Orientação a Objetos.

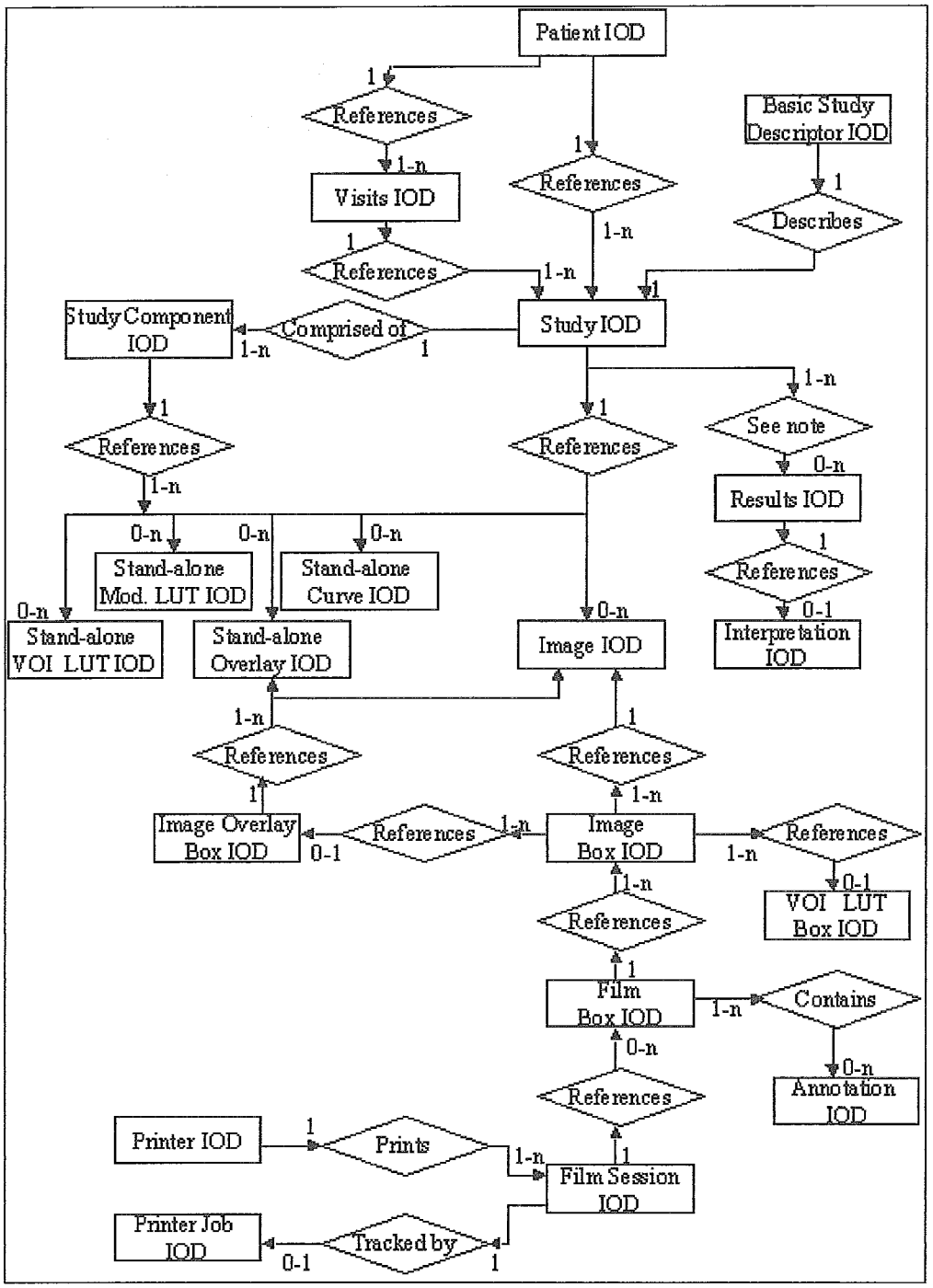

Figura 3.9: Modelo de dados do DICOM

Na figura acima, os retângulos representam as entidades, os losangos os relacionamentos. A sigla IOD significa *Information Object Definition*, VOI *Value of Interest*, e a LUT significa *Lookup Table Modality*.

Neste trabalho foi utilizado um modelo simplificado, que trata da relação entre o paciente, seus estudos, séries e imagens. Ele está representado na figura 3.10:

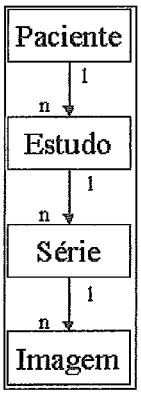

Figura 3.10: Modelo DICOM simplificado

Como mostrado neste capítulo, o DICOM, o PACS e o HIS trabalham em conjunto, formando todo o ambiente de informações digitais da área médica. O PACS é um dos sistemas que compõem o HIS, e o DICOM é o suporte para toda a rede PACS. A figura 3.11 ilustra esta situação, apresentando a interligação entre eles.

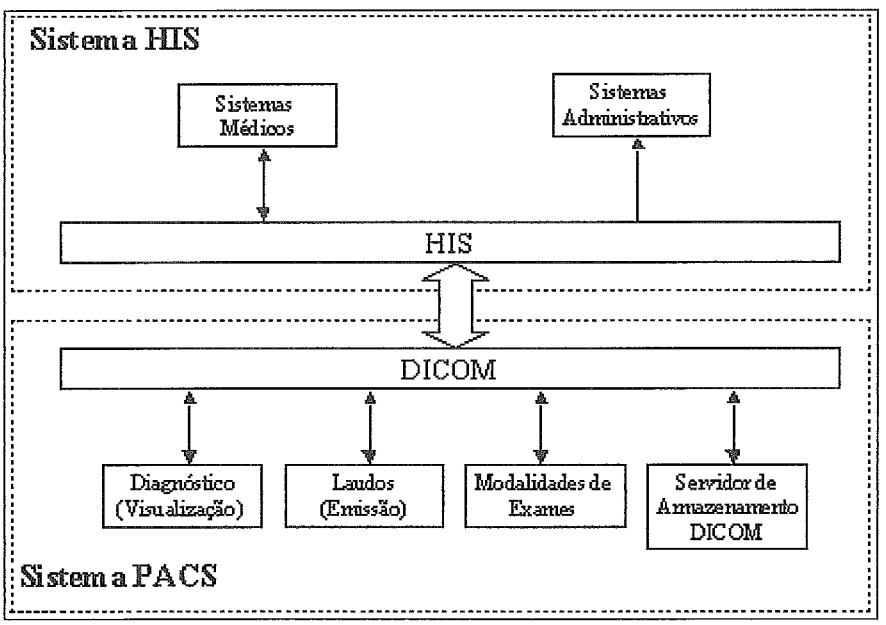

Figura 3.11: Integração HIS/PACS/DICOM

No próximo capítulo, serão apresentados a arquitetura do *software*, a linguagem utilizada, a interface com o usuário e o processo de *streaming*.

Capítulo 4:

O Sistema WebMedViewer

4.1. Introdução

No desenvolvimento de *software*, várias questões estão envolvidas, como: as funcionalidades desejadas; a arquitetura a ser utilizada; tanto do ponto de vista lógico como físico; a linguagem de programação mais adequada; e o ambiente onde o *software* estará inserido.

O *software* desenvolvido nesta dissertação, denominado WebMedViewer, visa disponibilizar, em tempo real, imagens médicas através de uma rede Intranet/Internet, possibilitando que o usuário médico realize a análise de exames das diversas modalidades.

Para que isso seja possível, vários pontos devem ser considerados, além da interface com o usuário a ser criada, e a técnica de *streaming* para a transmissão da imagem. É necessário, também, conhecer e escolher a arquitetura, assim como os *softwares* de "infra-estrutura" que darão suporte ao sistema, e a linguagem de programação utilizada.

Neste capítulo são apresentadas, na sessão 4.2, as diversas arquiteturas físicas existentes, e aquela escolhida para implementação desta dissertação. A sessão 4.3 descreve a linguagem de programação utilizada, sua estrutura e suas vantagens. Em seguida, na sessão 4.4, é detalhada a interface com o usuário. Por último, é apresentada a técnica de *streaming* implementada.

4.2. Arquitetura

Em uma empresa, os sistemas de informação se apoiam na arquitetura de informação (INEI, 2002). Inicialmente, a arquitetura dos sistemas de informação eram centralizadas. Os *mainframes* eram responsáveis por tudo, gerenciamento dos dados, consulta, interface com o usuário, etc. Com o passar do tempo, este tipo de arquitetura passou a ser ineficiente, não satisfazendo mais às necessidades das empresas, gerando a necessidade de uma nova arquitetura. Assim, iniciou-se a descentralização de dados e

recursos de processamento. Foi neste ambiente que surgiu a arquitetura cliente/servidor (BATTISTI, 2002a), que foi tão utilizada, e ainda é, por diversas empresas em todo o mundo.

A arquitetura cliente/servidor em duas camadas consiste em uma ou mais máquinas que atuam como servidores, disponibilizando recursos para os demais computadores, que atuam como clientes. Com isso, tem-se servidor de arquivos, de banco de dados, de impressão, etc (BATTISTI, 2002a). A figura 4.1 ilustra esta arquitetura.

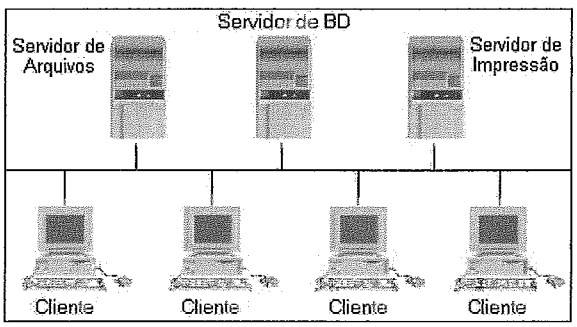

Figura 4.1: Arquitetura Cliente-Servidor.

É importante observar, que não é obrigatório haver uma máquina para cada serviço. Pode-se ter o servidor de banco de dados e de arquivos em um mesmo computador.

No modelo em duas camadas, tem-se um programa instalado em cada máquina cliente, que é responsável pelo acesso aos dados armazenados no servidor. Neste tipo de arquitetura, o servidor de banco de dados, por exemplo, é responsável apenas por armazenar e gerenciar as informações. Já o *software* cliente é responsável pela interface com o usuário (porção gráfica do *software*), e pela lógica do negócio (regras que definem o processamento da informação) (BATTISTI, 2002a). A lógica do negócio engloba desde a consulta e inserção de dados no banco de dados, até, por exemplo, o cálculo de tarifas e descontos em *softwares* de comércio.

A primeira versão desenvolvida nesta dissertação foi em duas camadas. Sendo assim, o *software*, um aplicativo Java, era responsável pela interface com o usuário e pela consulta ao banco de dados, localizado em um servidor. A figura 4.2 ilustra a arquitetura desta primeira versão.

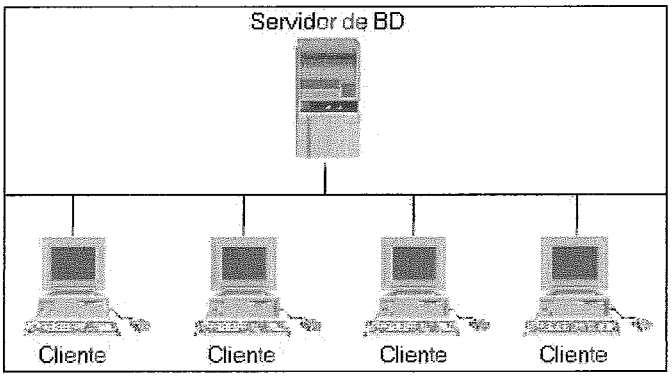

Figura 4.2: Arquitetura em duas camadas

Esta implementação não contém nenhuma parte de *streaming* de imagem, pois, por ser em duas camadas, o servidor de banco de dados não processa a imagem antes de enviá-la, e o aplicativo (no cliente), não tem como fazê-lo, pois a imagem não é local.

O modelo de duas camadas é muito utilizado, porém, possui uma série de desvantagens. Por exemplo, se for realizada uma alteração na interface ou nas regras de negócio, o *software* terá que ser reinstalado em todas as máquinas clientes, mesmo que a alteração tenha sido mínima. Isso gera um grande problema de controle de versão, já que é extremamente complicado gerenciar qual versão está instalada em cada cliente (BATTISTI, 2002a).

Há também o conhecido "inferno das DLL's", onde os aplicativos instalam diferentes versões de uma mesma DLL, prejudicando o funcionamento de outros programas (BATTISTI, 2002a). Como o *software* desenvolvido nesta dissertação não implementa nenhuma DLL, não há o risco de ocorrer este tipo de problema.

A evolução do modelo de duas camadas veio com o crescimento da Internet, surgindo assim, o modelo em três camadas. Este modelo consiste em retirar a lógica do negócio do cliente, e colocá-lo em um servidor de aplicações. Desta forma, a atualização da lógica do negócio não requer a reinstalação do *software* em todos os clientes (BATTISTI, 2002a). A figura 4.3 ilustra esta arquitetura.

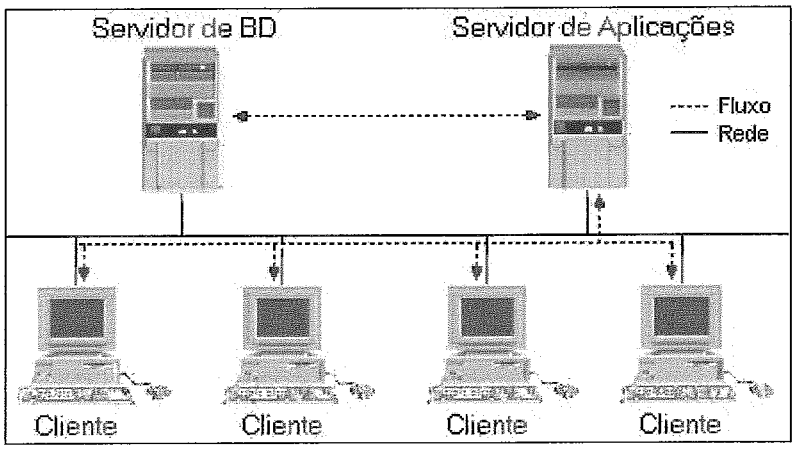

Figura 4.3: Arquitetura em três camadas

Como se pode observar na figura acima, o cliente acessa o banco de dados de acordo com as regras de negócio presentes no servidor de aplicações. Desta forma, tem-se as três camadas, que são (BATTISTI, 2002a):

- **Apresentação:** que é a interface com o usuário. Esta camada continua no *software* instalado no cliente, ou seja, qualquer alteração na interface com o usuário gera a necessidade de reinstalação do *software*.
- **Lógica do negócio:** que são as regras do negócio, que determinam como os dados serão acessados, utilizados, etc. Ao separar a lógica do negócio em um servidor, qualquer alteração realizada nas regras será automaticamente enxergada pelos clientes, sem a necessidade de uma reinstalação do programa.
- **Dados:** que são os dados armazenados no servidor de banco de dados, necessários para o funcionamento da aplicação. Como já foi dito anteriormente, os dados são acessados apenas pelo servidor de aplicação, e não mais pelo cliente.

Esta arquitetura resolveu um problema da arquitetura em duas camadas: a instalação de uma nova versão da aplicação cliente toda vez que uma regra de negócio fosse alterada. Porém, o problema da alteração na interface continua. Se algo for mudado, por menor e mais insignificante que seja, será necessário reinstalar o *software* em todos os clientes.

Para resolver este problema, surgiu o modelo de quatro camadas. Esta arquitetura retira a camada de apresentação do cliente, centralizando-a em um servidor *web*. Desta forma, o cliente passa a "buscar" a interface, através de um navegador, em um servidor. Não há mais uma aplicação instalada no cliente que necessite ser

atualizada. Ao se alterar a interface, basta atualizá-la no servidor *web*, que todos os clientes terão acesso (BATTISTI, 2002b). A figura 4.4 exibe esta arquitetura:

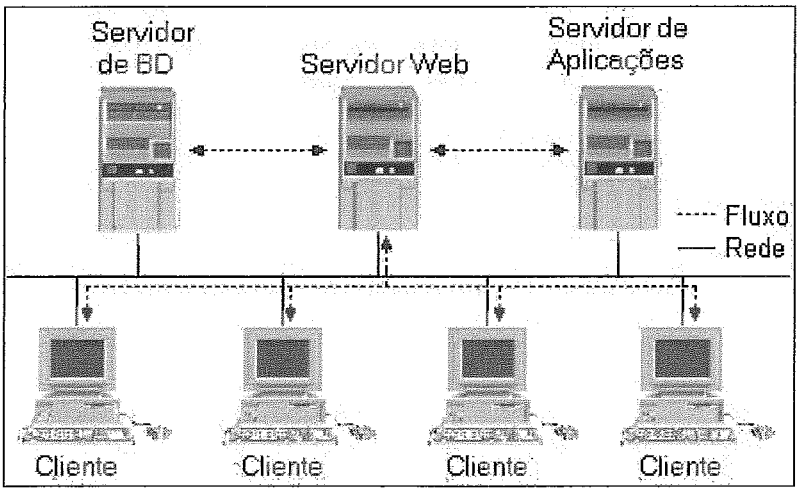

Figura 4.4: Arquitetura em quatro camadas

Da mesma forma que na arquitetura de três camadas, todo acesso ao banco de dados é regulamentado pelo servidor de aplicação. As camadas do modelo são (BATTISTI, 2002b):

- **Cliente:** que é o navegador. Para acessar a aplicação, o cliente precisa apenas digitar um endereço *web* em seu navegador.
- **Apresentação:** que é o servidor *web*. A interface pode ser feita utilizando-se qualquer tecnologia capaz de gerar páginas no navegador. Vale reforçar, que qualquer atualização da interface necessita ser feita apenas no servidor *web*. Desta forma, todos os clientes possuem acesso a versão mais atualizada do *software*.
- **Lógica:** são as regras que determinam como os dados serão utilizados. Assim como no modelo em três camadas, por estar em um servidor próprio (o servidor de aplicações) toda atualização necessita ser realizada apenas no servidor de aplicações, garantindo que todos os clientes tenham automaticamente acesso, de forma indireta, através do servidor *web*.
- **Dados:** é o servidor de banco de dados, com todas as informações necessárias para a aplicação.

A aplicação desenvolvida, que antes estava em duas camadas, foi alterada passando para o modelo em quatro camadas. Como se pode observar na figura 4.5, tem-se o cliente com um navegador, o servidor *web* com a interface com o usuário, o servidor de aplicação com toda a lógica do negócio, e o servidor de banco de dados.

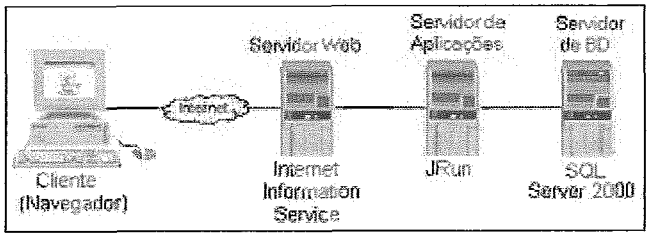

Figura 4.5: Arquitetura com os componentes utilizados

É importante observar que não há a necessidade de os servidores estarem fisicamente em máquinas separadas. Pode-se ter o servidor de banco de dados, o servidor de aplicações e o servidor *web* em apenas um computador.

O cliente, como dito anteriormente, precisa ter instalado apenas um navegador que suporte programas escritos na linguagem de programação escolhida para o desenvolvimento, o Java.

O servidor *web* utilizado foi *Internet Information Service* (IIS) da Microsoft.

O servidor de aplicações, foi o *JRun*, da *MacroMedia*. Ele é responsável por fornecer ao servidor *web* todas as requisições feitas pelo cliente. O *JRun* contém o programa que realiza a consulta ao banco de dados, e busca as imagens para serem exibidas, sendo o responsável pelo *streaming* da imagem.

O servidor de banco de dados utilizado foi o *SQL Server 2000* da Microsoft. Porém, como dito anteriormente, a imagem está no formato DICOM, ou seja, ela é um metadado contendo todas as informações que devem ser armazenadas no banco de dados. O *SQL Server* não é capaz de extrair da imagem as informações necessárias para criar e povoar o banco de dados. Por isso, tem-se o *Coquest* trabalhando em conjunto com o *SQL Server*.

O *Conquest* é um servidor DICOM que extrai as informações do cabeçalho da imagem, e insere essas informações nas tabelas corretas do *SQL Server*. Para a aplicação, o *Conquest* é transparente, ou seja, ela trabalha diretamente com o servidor de banco de Dados *SQL Server*. Porém, sem o *Conquest* as informações teriam que ser inseridas no *SQL Server* manualmente. A figura 4.6 ilustra o trabalho conjunto dos dois.

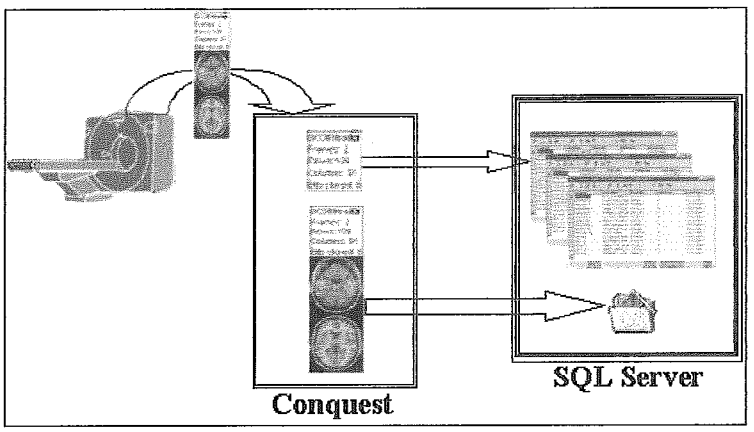

Figura 4.6: *Conquest* e *SQL Server*

Como se pode observar, novas imagens geradas nos aparelhos de exame (ressonância magnética, Raio-X, Ultra-Som, etc) são enviadas para o *Conquest*, que extrai as informações e as envia para as tabelas do SQL Server.

É importante observar que tanto o servidor *web*, quanto o de aplicação e o de banco de dados podem ser alterados. O *software* não utiliza nenhuma particularidade de nenhum deles.

Tão importante quanto a arquitetura e a infra-estrutura para o *software*, é a linguagem de programação. É ela que possibilita a implementação de todas as camadas do *software*, incluindo-se a interface com o usuário.

4.3. Linguagem de Programação

A linguagem de programação utilizada para desenvolver este trabalho foi a linguagem Java.

Esta é uma linguagem de programação orientada a objetos desenvolvida pela Sun Microsystems, em 1991, como parte de um projeto de pesquisa destinado à criação de *software* para dispositivos eletrônicos (equipamentos para televisão, videocassetes, torradeiras, etc) (CHAN, 1999) (CURSO IBM DE PROGRAMAÇÃO, 1999) (MORRISON, 1999).

Ela foi concebida a partir do C++ para ser compacta, simples, rápida e eficiente, sendo capaz de realizar as mesmas tarefas e resolver o mesmo tipo de problema que outras linguagens de programação. Além disso, a linguagem Java foi feita para ser portável ao longo de diferentes plataformas e dispositivos operacionais, sendo

essa uma de suas vantagens mais características sobre outras linguagens (CHAN, 1999) (CURSO IBM DE PROGRAMAÇÃO, 1999).

A portabilidade do Java só é possível por dois motivos principais: Java é uma linguagem interpretada e suas bibliotecas são construídas de forma a não haver necessidade de se alterar o código inter-plataforma (CURSO IBM DE PROGRAMAÇÃO, 1999).

No primeiro caso, o código fonte é passado para o formato denominado *bytecode* que é interpretado pela Máquina Virtual Java (*Java Virtual Machine*, também conhecida pelas siglas JVM e MVJ). A JVM garante que um mesmo código em *bytecode* será executado em qualquer plataforma, pois funciona como um processador Java, sendo uma camada entre o arquivo *bytecode* e a CPU. Na figura 4.7, pode-se observar que existe apenas um arquivo *bytecode* e várias JVMs, uma para cada plataforma, não existindo um arquivo executável.

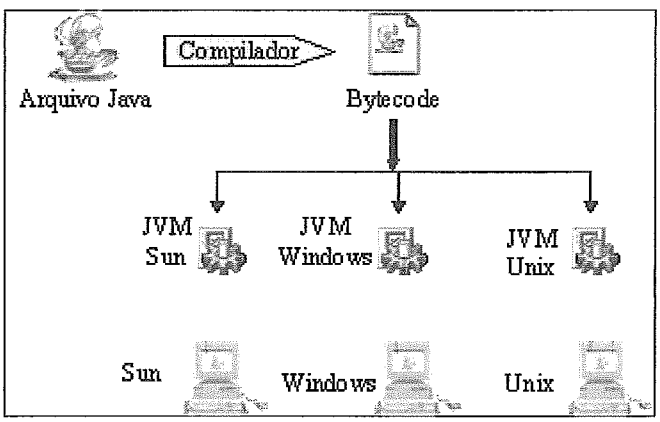

Figura 4.7: Máquina Virtual Java para diversas plataformas

No segundo caso, a portabilidade se dá no nível de código fonte. Suas bibliotecas são escritas de forma que o código não necessita ser reescrito ao se mudar de plataforma. As chamadas dos métodos são sempre as mesmas. Um exemplo claro é o pacote *Abstract Window Toolkit* (AWT), que fornece um conjunto de classes independentes de plataforma que cuidam das operações gráficas. Com isso, ao se criar um botão, ou qualquer outro componente de interface, seu aspecto é diferente em cada plataforma, pois é utilizado o botão padrão daquela plataforma. Porém, o código fonte Java é o mesmo.

Tem-se também a biblioteca *Swing*, incorporada posteriormente, que é uma extensão do AWT. Esta agregou novos componentes e funcionalidades, como por exemplo, a não necessidade de se utilizar componentes do sistema operacional no qual o programa está sendo executado. Com isso, o aplicativo passa a ter a mesma aparência

em qualquer plataforma. Esta é a grande diferença entre AWT e *Swing*: o primeiro se baseia no sistema em que o aplicativo está rodando, enquanto o segundo não utiliza nenhum código nativo. No *software* WebMedViewer implementado, foram utilizadas as duas classes de interface com o usuário.

A portabilidade do Java o torna ideal para a distribuição de programas executáveis através da Internet, onde há grande variedade de plataformas. Esse foi um ponto determinante na escolha desta linguagem, já que o objetivo do trabalho é permitir a visualização de imagens pela Internet.

Com o Java, pode-se criar dois tipos de programa: *applets* e *aplicativos.* Uma *applet* é um programa dinâmico e interativo que pode ser executado em um navegador *web* ou em um visualizador de *applets* (*AppletViewer*). Isso lhe traz a vantagem de já ter uma janela de programa e a capacidade de responder aos eventos da interface com o usuário por meio do *browser*. A *applet* é mais segura que o aplicativo, já que possui acesso restrito ao sistema de arquivos, tanto na máquina local, onde está sendo executada, quanto nas máquinas em rede, fato este de responsabilidade da própria linguagem Java, por encarar os programas de forma diferente (CHAN, 1999).

Um aplicativo, por sua vez, é um programa que roda independente do *browser*, podendo funcionar isoladamente. O aplicativo pode ser executado através de linhas de comando, ou pode conter interface gráfica, com janelas, botões, etc.

No *software* implementado, tem-se uma *applet* de pesquisa chamando aplicativos de visualização. Porém, como o aplicativo está sendo executado a partir da *applet*, ele possui as mesmas restrições de acesso, aumentando a segurança do *software*. Este esquema de segurança nativo da linguagem Java, foi outro ponto importante na sua escolha.

Como dito anteriormente, uma *applet* não pode acessar arquivos da máquina local, nem arquivos ou banco de dados, que não estejam em seu diretório na máquina de origem. Isso gera um problema, já que é necessário realizar consultas a banco de dados, e leitura/processamento dos arquivos de imagem. Para resolver este problema, foram utilizados os *servlets*.

Os *servlets* Java são programas Java utilizados para expandir, ou aperfeiçoar as funcionalidades do servidor. Eles permitem que o desenvolvedor estenda e personalize o servidor com portabilidade, flexibilidade e facilidade (OLIVEIRA, 2003). O *servlet* está para o servidor, assim como a *applet* está para o navegador.

Por exemplo, quando a *applet* necessita acessar o banco de dados ou ler um arquivo ela envia uma requisição para o *servlet*. Ele, por sua vez, realiza a consulta, e retorna o resultado para a *applet*.

No sistema WebMedViewer, o *servlet* realiza o *streaming* da imagem, além de promover o acesso ao banco de dados.

Com relação ao tratamento da imagem, o Java tem uma particularidade que vale ressaltar. Há uma função no Java, a *PixelGrabber* da classe de mesmo nome, que permite transformar uma imagem (que é uma matriz de *pixels*) em um *array*. A figura 4.8 ilustra o resultado desta função.

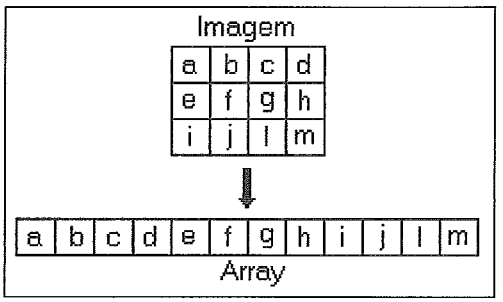

Figura 4.8: Transformando uma imagem em um *array*

A relação entre um ponto na imagem (matriz) e no *array* se dá pela seguinte fórmula: um ponto (m, n) está na posição (n*scan + m + off), onde **scan** é o tamanho de uma linha de dados de *pixels* na imagem, e **off** é o deslocamento dos dados dentro do *array* (CHAN, 1999). É possível, também, converter um *array* em uma imagem. O relacionamento, neste caso, se dá pelas fórmulas: m = parte inteira da divisão do índice pelo *scan*, e n = índice – m (foi considerado o ponto (m, n) de uma imagem de largura *scan*).

Estas funções são muito importantes para alguns processamentos da imagem, como rotação (espelhamento), e para o próprio envio da imagem do servidor para o aplicativo.

O sistema WebMedViewer tem por objetivo permitir a visualização de imagens médicas em rede, Intranet ou Internet, além de permitir ao usuário manipular as imagens alterando características tais como: brilho, contraste, inversão de cores, rotação, zoom, etc. Desta forma o usuário pode melhor visualizar e analisar as imagens dos exames.

Para que o *software* possa ser disponibilizado de forma rápida e simples, a arquitetura utilizada foi a de quatro camadas. Sendo assim, o gerenciamento é facilitado, já que os dados estão em um servidor de banco de dados, a lógica do negócio no servidor de aplicação e toda a interface com o usuário no servidor *web*. Essa arquitetura,

como dito anteriormente, simplifica a atualização de qualquer parte do *software*, pois com as camadas centralizadas, a alteração da interface é feita apenas no servidor *web*, ficando disponível para todos os usuários de forma automática e transparente.

4.4. A Interface com o Usuário

A interface com o usuário é feita através de *applet* e aplicativos Java. O navegador *web*, único *software* que deve ser instalado no cliente, é responsável pela exibição da *applet* que realiza a pesquisa dos estudos[1] disponíveis para visualização. O aplicativo exibido após a *applet* fornece ao usuário a primeira imagem de cada série do estudo desejado, para a seleção de qual série será visualizada. O aplicativo seguinte é o visualizador de séries. Nele, todas as imagens da série são disponibilizadas para que o usuário as visualize e manipule para uma melhor análise.

4.4.1. Pesquisa

A figura 4.9 ilustra a primeira tela exibida para o usuário: a tela *applet* de pesquisa.

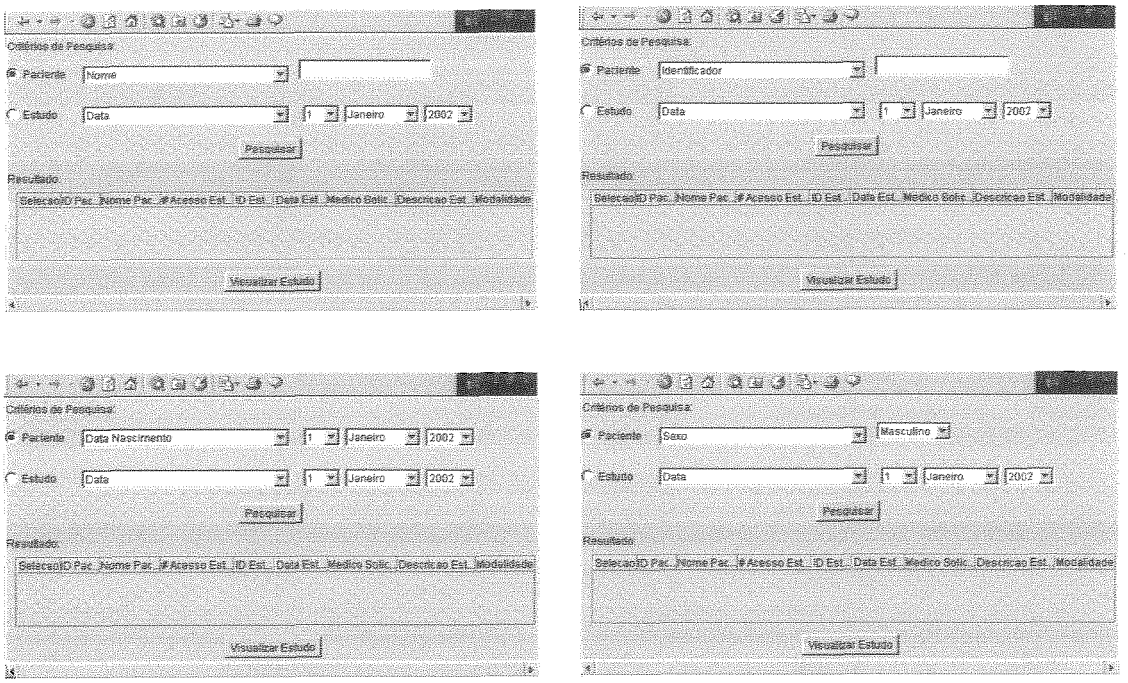

Figura 4.9: *Applet* de Pesquisa

[1] Um exame é composto por um ou mais estudos. Cada estudo é formado por uma ou mais séries, que contém as imagens.

Como se pode observar na figura 4.9, a pesquisa pode ser realizada por paciente, ou por estudo. Após escolher a pesquisa por paciente, o usuário deve selecionar qual o critério de seleção será utilizado. Para paciente, pode-se pesquisar por nome, identificador do paciente, data de nascimento ou sexo. Para os dois primeiros, nome e identificador, deve-se informar qual nome ou identificador deseja-se pesquisar. Não é necessário fornecer o nome, ou identificador, completo. Como nas páginas de pesquisa *web*, pode-se fornecer como critério de pesquisa para nome, por exemplo, o texto "a*". O resultado fornecido será todos os pacientes cujo nome começa com "a". Para a pesquisa por data de nascimento, o usuário deve fornecer uma data nos campos dia, mês e ano. Para a pesquisa por sexo, deve-se escolher o desejado no "menu suspenso" fornecido.

Se a pesquisa for feita por estudo, como mostra a figura 4.10, ao invés de paciente, as opções de critério de pesquisa são: data de realização do estudo, número de acesso, identificador, período, médico solicitante ou descrição do estudo. Para a pesquisa por data de realização do estudo, é solicitada a data desejada. Para período, deve-se informar a data inicial e final. O resultado da pesquisa será todos os estudos realizados entre as datas fornecidas, incluindo-as. Para a pesquisa pelos demais critérios (número de acesso, identificador, médico solicitante e descrição), o usuário deve fornecer um "texto" para a pesquisa. Assim como na pesquisa por nome do paciente, é permitido o uso do asterisco no texto de busca.

Figura 4.10: Pesquisa por Estudo

Há um tratamento de erro para as pesquisas realizadas por data, seja na categoria de Estudos ou Pacientes. Se, por exemplo, o usuário escolher a data 30 de fevereiro de 2003 uma mensagem é exibida, informando que este mês não possui dia 30, e a data é convertida para o último dia válido do mês, neste caso 28 de fevereiro de 2003.

Após determinar o critério de pesquisa, o usuário deve acionar o botão "Pesquisar". Após esta ação, é exibido, na parte inferior da janela, o critério de pesquisa, e o resultado, em forma de tabela. Na tabela de resultados estão disponíveis dados do paciente (nome e identificador) e do estudo (número de acesso, data de realização, médico solicitante, descrição e modalidade). A figura 4.11 ilustra um resultado de uma pesquisa:

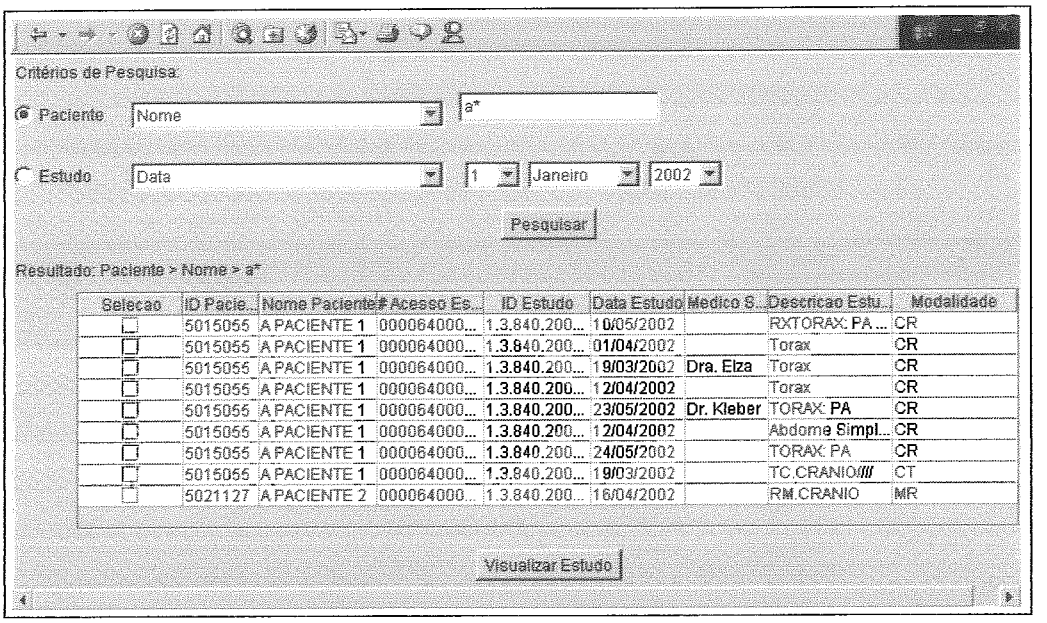

Figura 4.11: Resultado da Pesquisa

Como se pode observar a pesquisa foi realizada por PACIENTE/NOME, iniciado com a letra "A".

A primeira coluna da tabela, como se observa na figura 4.11, possui um botão de seleção (*checkbox*) para cada linha. É através destes botões que o usuário seleciona o estudo que deseja visualizar. Pode-se escolher até três estudos para visualização.

Após determinar os estudos que deseja visualizar, o usuário deve acionar o botão "Visualizar Estudos", que o disponibilizará para o aplicativo de visualização dos estudos.

4.4.2. Visualização de Estudos

Este aplicativo tem como objetivo exibir para o usuário a primeira imagem de cada série do estudo que ele selecionou para visualizar. A figura 4.12 ilustra este aplicativo.

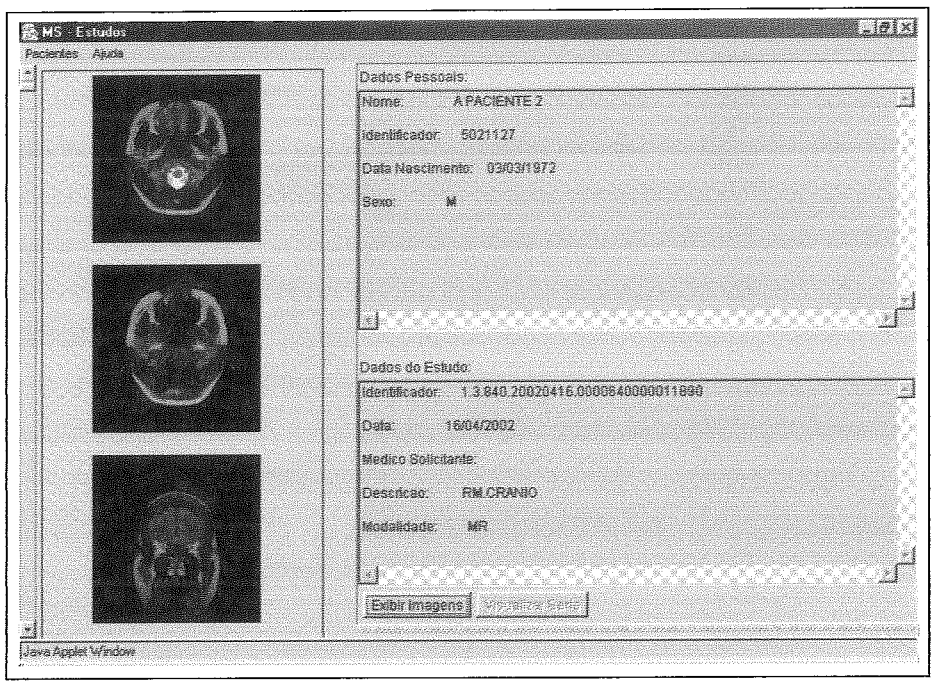

Figura 4.12: Visualizador de Estudo

Como se pode observar, na parte esquerda da janela estão as imagens, e na parte da direita, as informações sobre o paciente e sobre o estudo escolhido.

É importante observar que o usuário pode alterar o estudo que está sendo visualizado nesta janela, caso ele tenha escolhido mais de um na *applet* de pesquisa, através do menu que se encontra na parte superior da janela.

Ao selecionar uma das imagens exibidas, e acionar o botão "Visualizar", o aplicativo para visualização da série será exibido.

4.4.3. Visualizador de Série

Após a seleção da série que se deseja analisar, o conjunto de imagens é exibido para o usuário. A figura 4.13 ilustra um exemplo da visualização de uma série de imagens médicas.

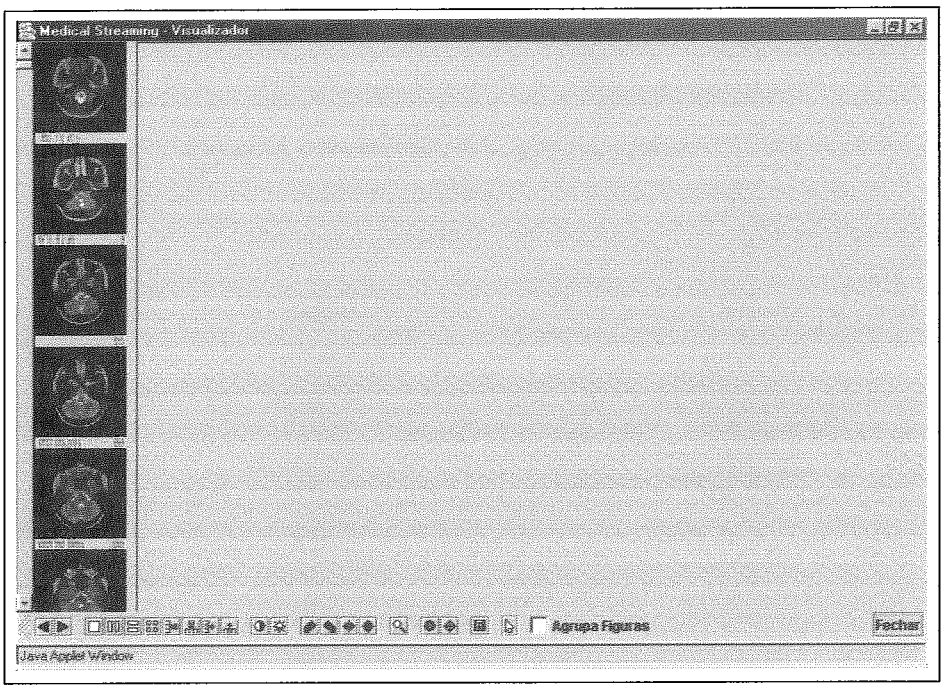

Figura 4.13: Visualizador de Série

No canto esquerdo da janela encontram-se as imagens reduzidas da série selecionada. Para ampliar a imagem, basta selecioná-la com o mouse e, mantendo o botão pressionado, arrastá-la para a área de visualização. A área de visualização é a maior porção localizada no centro da janela.

Na parte inferior, tem-se a barra de ferramentas. Ela contém diversas funcionalidades para manipular as imagens da área de visualização, e sua distribuição na porção gráfica da tela. Essas funcionalidades foram levantadas junto aos médicos, para verificar suas necessidades que, basicamente, é simular digitalmente o que se pode fazer com os filmes radiológicos. A figura 4.14 exibe a barra de ferramentas em detalhes. Como se pode observar, os botões estão agrupados de acordo com sua funcionalidade.

Figura 4.14: Barra de Tarefas do Visualizador

Os dois primeiros botões auxiliam na "navegação" das imagens reduzidas na série. Eles são responsáveis por avançar ou voltar na seqüência de imagens.

Além da facilidade de navegação nas imagens, é possível manipular a área de visualização, ou seja, determinar o número de imagens que podem ser visualizadas simultaneamente. Esta é a função do grupo de botões, exibidos na figura 4.15 que mostra a funcionalidade de cada um.

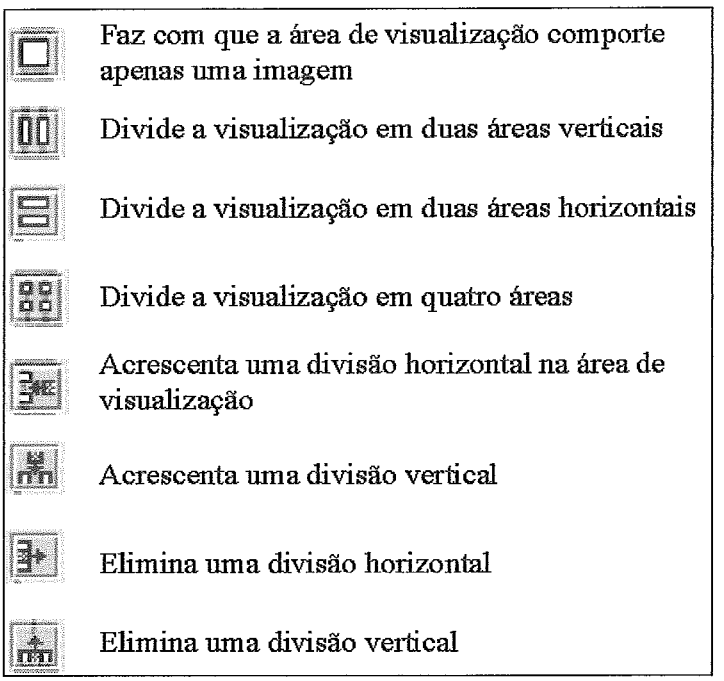

Figura 4.15: Botões de manipulação da área de visualização

Porém, apenas navegar na seqüência de imagens e alterar o número de imagens simultaneamente visualizadas pode não ser suficiente para uma boa análise dos resultados dos exames. É necessário poder alterar o brilho e o contraste, rotacionar a imagem, dentre outras funcionalidades, para facilitar os usuários simular o que podem fazer com os exames em filmes. Para isso existem os botões específicos para manipulação das imagens.

O botão da esquerda, ilustrado na figura 4.16, promove a inversão das cores da imagem, enquanto o botão da direita é responsável por alterar o brilho e o contraste. Para se alterar essas duas últimas propriedades da imagem, o usuário deve, após acionar o botão, arrastar o mouse sobre a imagem. Arrastando-o na horizontal, altera-se o contraste. Ao arrastá-lo na vertical, altera-se o brilho da imagem.

Figura 4.16: Botões de inversão de cor, e brilho/contraste

Para a realização destas três funcionalidades, realiza-se um processamento da imagem, aplicando-se filtros que promovem a inversão das cores, e a alteração do brilho e do contraste. Estes filtros são feitos utilizando-se a classe RGBImageFilter do Java. Eles se baseiam apenas no valor do *pixel* e na sua posição (quando for o caso) para se

calcular o novo valor de cor. A figura 4.17 exibe o resultado da utilização da inversão de cores, do brilho e do contraste em uma imagem digital.

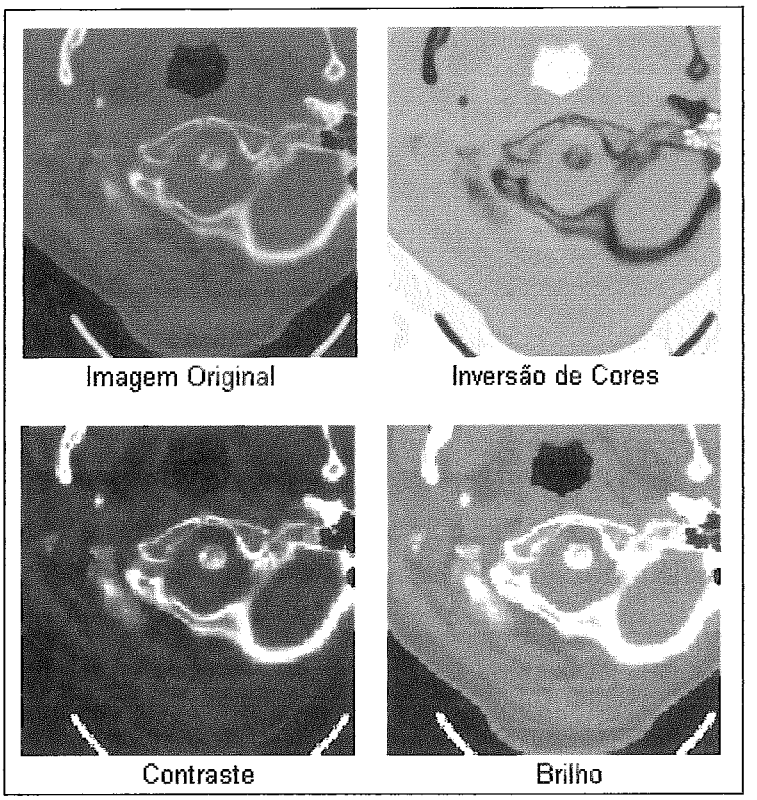

Figura 4.17: Funções de Inversão de Cores, Brillho e Contraste

A rotação da imagem é realizada pelo conjunto de botões ilustrados na figura 4.18. Como os próprios botões já sugerem, é possível rotacionar a imagem para a direita, para a esquerda (rotação de 90° ao redor do eixo perpendicular ao plano da tela), além de realizar uma rotação horizontal e vertical (rotação de 180° ao redor do eixo longitudinal da tela), também conhecida como espelhamento.

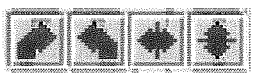

Figura 4.18: Botões de rotação

Para as funções de rotação foram criados algoritmos e funções que promovem, no *array* da imagem, o deslocamento correto dos *pixels* de forma a promover a rotação desejada. A figura 4.19 ilustra o resultado das rotações em imagem.

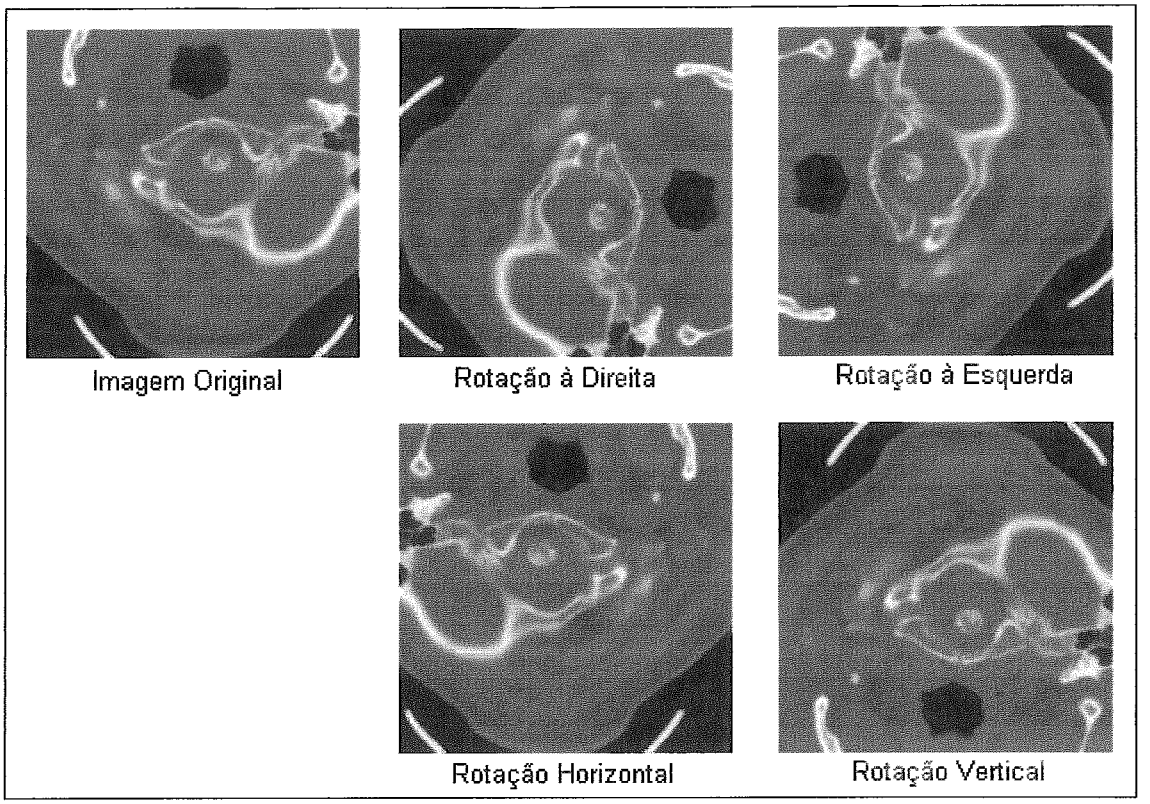

Figura 4.19: Rotações

Há também a possibilidade de se realizar o zoom na imagem, assim como centralizá-la e movê-la. Como mostra a figura 4.20.

Figura 4.20: Mais funcionalidades.

Para promover o zoom, o usuário deve selecionar esta funcionalidade, e arrastar o *mouse* sobre a imagem. Já a funcionalidade de centralizar funciona da seguinte forma: o usuário aciona o botão de centralizar, e clica sobre a imagem. Ao fazer isso, a imagem é deslocada de forma que o ponto selecionado pelo usuário se localize no centro da área de visualização desta imagem.

Para mover a imagem, é necessário apenas selecionar a funcionalidade, pressionar o botão do *mouse* sobre a imagem e, mantendo-o pressionado, arrastá-lo. A figura 4.21 apresenta o resultado da aplicação de cada uma dessas funções em uma imagem. Na imagem que ilustra a centralização, o ponto vermelho representa o ponto utilizado como referência.

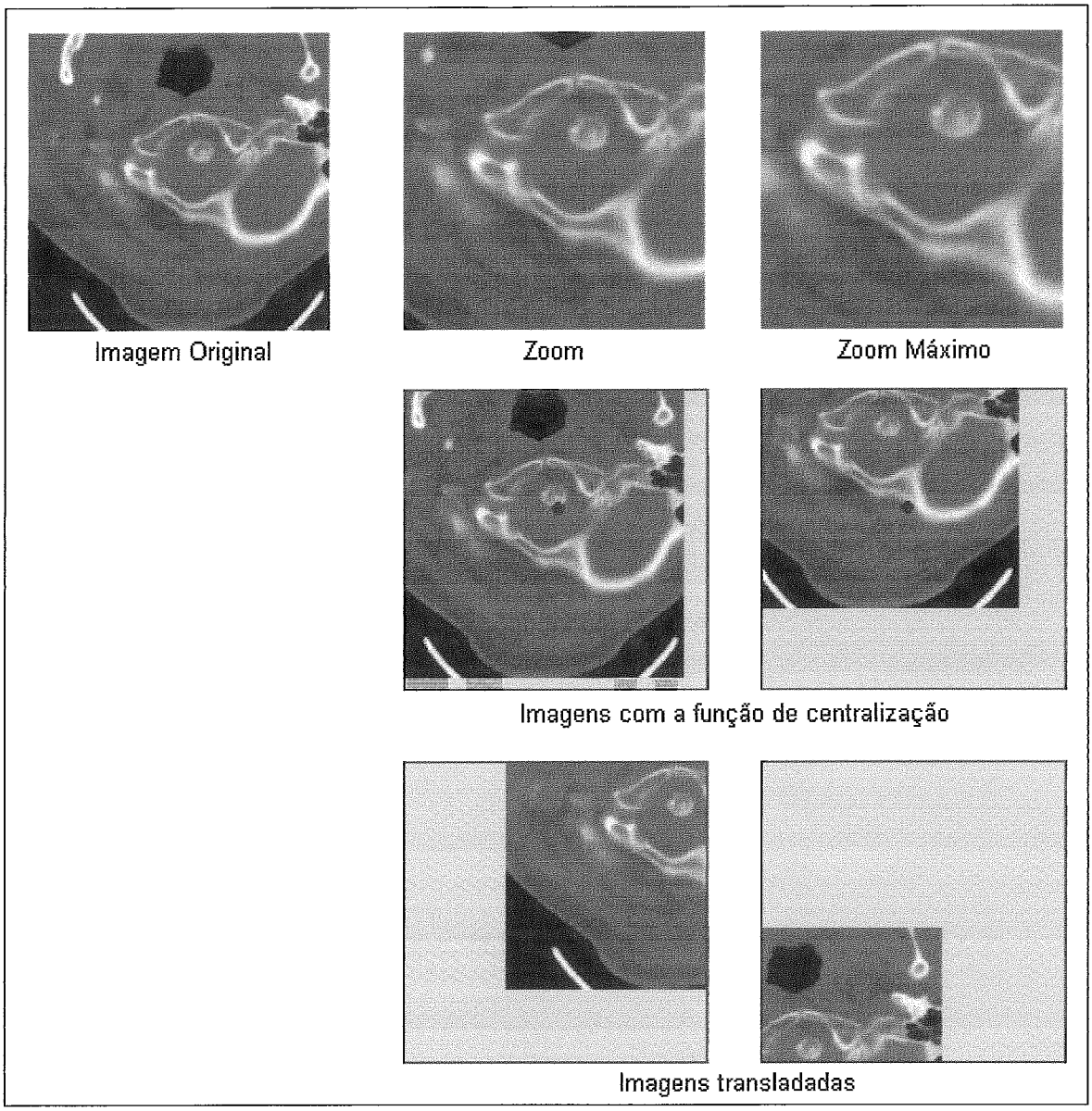

Figura 4.21: Zoom, Centralização e Translação

As funções de zoom e de translação são realizadas através da alteração dos parâmetros da imagem (pontos iniciais e finais) que são passados para a função de impressão da imagem. Já a função de centralização exige um cálculo de conversão de sistema, pois existem dois sistemas na janela. Um, é o sistema de coordenadas utilizado pelo Java, que determina a posição (x, y) onde o *mouse* foi acionado. O outro, é o sistema de coordenadas da imagem. A figura 4.22 exibe os dois sistemas.

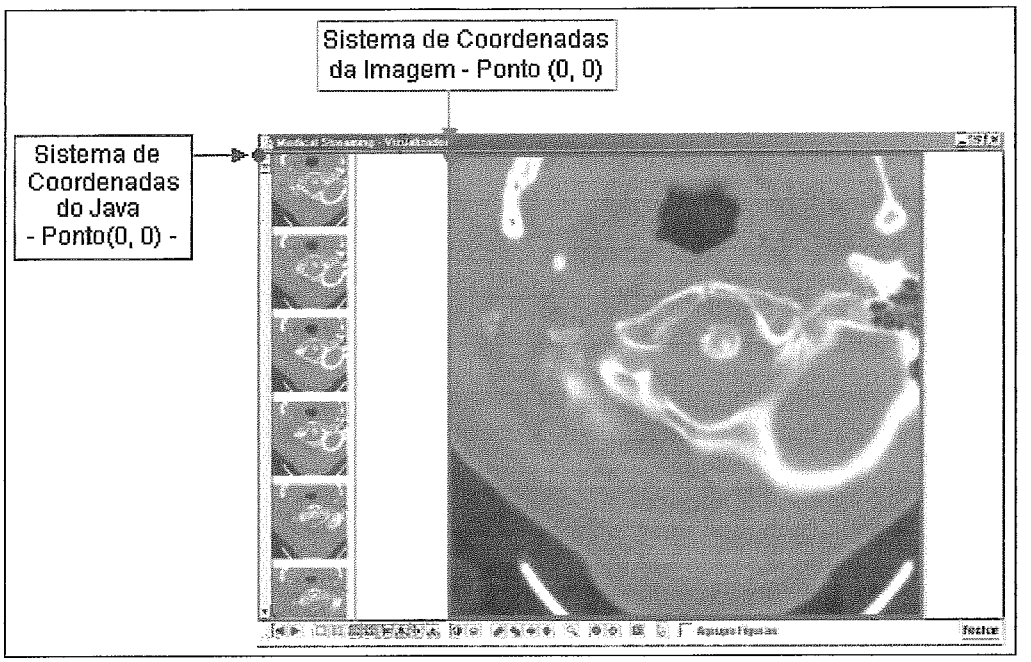

Figura 4.22: Sistemas de Coordenadas

Ao escolher o ponto que deve ser o centro da imagem, o ponto (x, y) do *mouse* que é passado para o programa está no sistema de coordenadas do Java. Ele deve ser convertido para o sistema de coordenadas da imagem, para depois serem realizados os cálculos de limite para mover a imagem de forma que o ponto selecionado se localize no centro da área de visualização.

Após realizar diversas alterações na imagem, pode ser necessário restaurá-la, ou seja, recuperar a imagem original. Para isso, há um botão específico para facilitar o trabalho do usuário. É o quarto botão, da esquerda para a direita, exibido na figura 4.20. A figura 4.23 ilustra esta função.

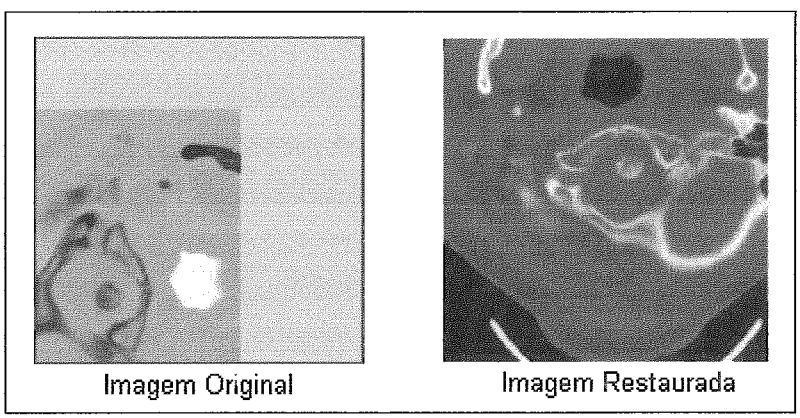

Figura 4.23: Restaurando uma imagem

O último botão da barra de tarefas, exibido na figura 4.20, serve para desmarcar a função ativa.

Como visto, existem inúmeras funções para a manipulação da imagem. Pode-se inverter suas cores, rotacionar, alterar brilho e contraste, transladar, etc. Porém todas estas funções estão sendo aplicadas a apenas uma imagem. Para permitir que uma mesma função seja aplicada a todas as imagens exibidas na área de visualização existe o botão de verificação "Agrupar Figuras". Ao marcar este botão, toda e qualquer função, exceto a de centralizar, que for utilizada será aplicada para todas as imagens exibidas na janela.

A razão de o botão "Agrupar Figuras" não ser válido para a centralização advém do fato de um ponto que serve de referência para uma imagem não ser válido para as demais imagens.

Nos aplicativos de visualização de estudo e de série, o carregamento das imagens é feito utilizando a técnica de *streaming*. Ele é realizado pelos *servlets* Java, localizados no servidor de aplicativo, que processam a imagem, e a enviam ao computador cliente da melhor forma possível.

4.5. *Streaming*

Como mencionado anteriormente, ao se realizar o *streaming* de imagens médicas, deve-se garantir a qualidade da imagem, não sendo possível utilizar, por exemplo, as compressões com perda, presentes na maioria das técnicas existentes.

O processo de *streaming* implementado utiliza as técnicas histograma, quicksort, endereço, subtração e redução, em uma metodologia unificada.

4.5.1. Histograma

A primeira proposta foi a utilização de histogramas, e que acabou se tornando o centro da técnica de *streaming* apresentada nesta dissertação.

O histograma consiste na quantificação do número de pixels referentes à cor (tons de cinza) na imagem. Desta forma, pode-se determinar as cores mais significativas na imagem, determinando a ordem em que devem ser transmitidas ao computador cliente. Assim, consegue-se ter uma rápida percepção do conteúdo da imagem, reduzindo a sensação de espera do usuário. A figura 4.24, abaixo, exibe a imagem, e seu histograma representado graficamente.

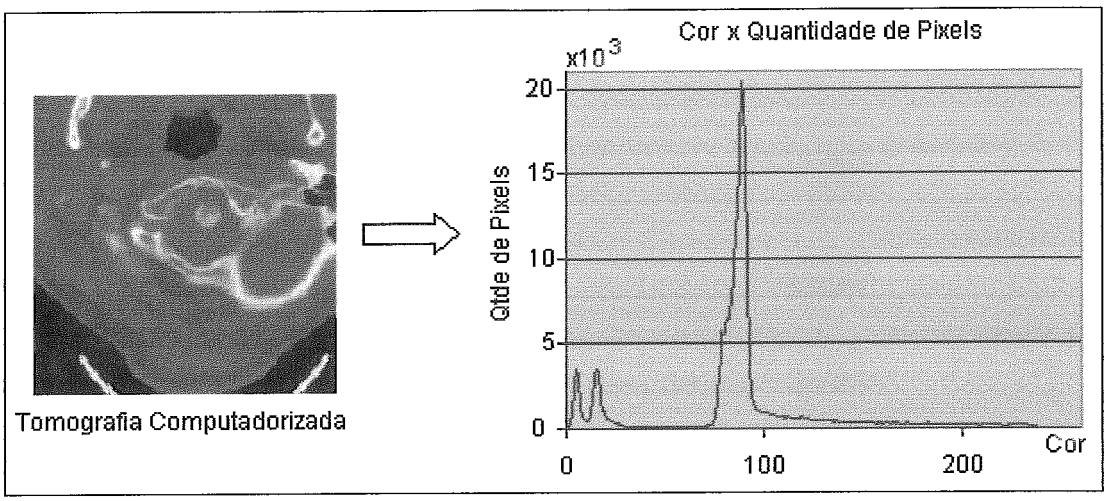

Figura 4.24: Imagem e seu histograma

Como pode ser observado na figura 4.24, as cores das imagens médicas variam de zero a 255. Isso ocorre porque elas são em tons de cinza. Para se calcular o histograma, a imagem é percorrida *pixel* a *pixel*, sendo eles armazenados em um vetor, separados pela cor que possuem. A figura 4.25 ilustra a estrutura formada.

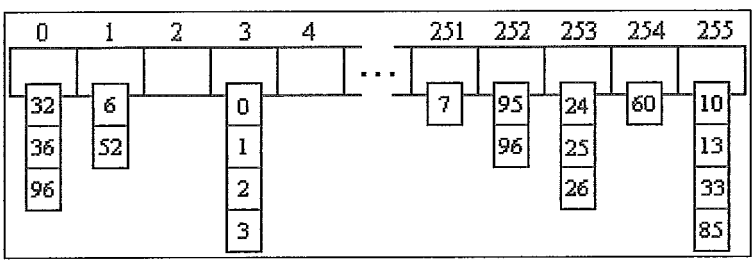

Figura 4.25: Estrutura para armazenar o histograma

O endereço do *pixel* é composto por apenas um número. Isso é possível devido a uma função do Java que transforma uma imagem 2D, com **w** de largura e **h** de altura, em um *array* de dimensão **w*h**. Isso traz uma série de facilidades. Uma delas é a própria forma de se referenciar e armazenar o endereço. A outra é que a partir do *array*, a linguagem é capaz de remontar a imagem, facilitando sua montagem no cliente. Desta forma, à medida que os *pixels* são enviados, eles são armazenados na posição correta do *array* para que a imagem seja remontada.

Sendo o histograma armazenado na estruturada figura 4.25, facilmente pode-se determinar a cor que mais está presente na imagem, pela quantidade de *pixels* (endereços) que cada uma possui, assim priorizando seu envio para a *applet*. A transmissão da informação é feita utilizando-se o protocolo HTTP. Sendo assim, cada cor do histograma, juntamente com seus endereços, são transformados em *string* para serem transmitidos.

O formato da *string*, mostrado na figura 4.26, descreve primeiramente a cor (antecedida pela letra 'c'), seguida de seus endereços de ocorrência na imagem original.

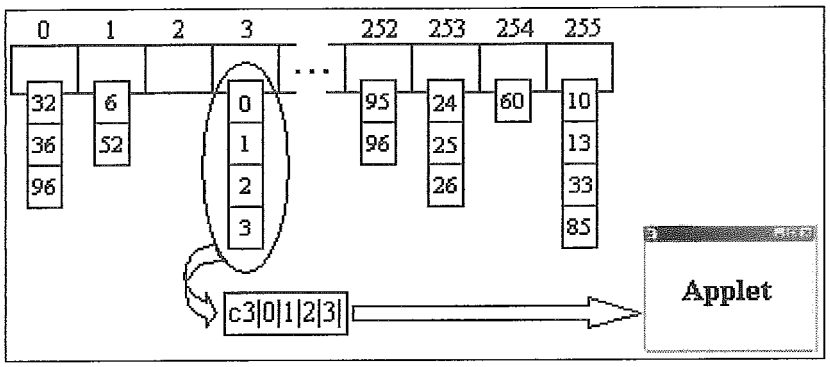

Figura 4.26: Envio de *pixels* para o usuário

Inicialmente, a *string* era montada com a cor e todos os seus endereços de *pixel*, para então ser transmitida à *applet*. Porém, observou-se que o processo ficava muito lento para as cores com um maior número de *pixels*. Essa demora deve-se a montagem da *string*, pois ela se torna muito extensa e sua manipulação computacional, muito custosa. Para solucionar este problema, dividiu-se a *string*. Apenas **k** endereços são enviados de cada vez, sendo o restante transmitido em seguida.

Uma das tentativas para aumentar a velocidade de envio da imagem, foi a de "pré-determinar" a ordem de envio das cores. Essa ordem seria fixa, de acordo com a avaliação das imagens, e observação da ordem em que as cores costumam aparecem. Porém isso não foi possível, já que em uma mesma série, a ordem das cores das imagens diversificam muito. Logo, a forma utilizada foi a de se determinar dinamicamente, após a montagem do vetor de cores e *pixels*, qual cor deve ser enviada de cada vez.

O histograma resolve o principal objetivo: permitir a visualização da imagem, mesmo antes de ela ter sido completamente carregada no computador cliente. Porém, ela não traz nenhuma vantagem com relação a tempo. A imagem continua demorando para ser totalmente carregada. Para melhorar este tempo, utilizou-se a técnica de endereço.

4.5.2. Endereço

O endereço consiste em uma compressão sem perda, onde *pixels* consecutivos, que possuem a mesma cor, são "agrupados" para serem enviados, semelhante ao que ocorre na compressão RLE (*Run Length Encoding*).

Após calcular o histograma, tem-se todos os *pixels* de uma cor em ordem crescente, já que o *array* da imagem é percorrido em apenas um sentido. Desta forma, pode-se perceber e agrupar os *pixels* consecutivos, transmitindo apenas o primeiro e o último *pixel* da seqüência. Isso, além de reduzir a quantidade de *pixels* a serem transmitidos, agiliza o envio da imagem, pois mais *pixels* são transmitidos de uma só vez.

Porém, existem *pixels* que não possuem nenhum "vizinho" com a mesma cor. Nestes casos, o *pixel* é enviado duas vezes, tornando-se o início e o fim da seqüência. A figura 4.27 ilustra como fica a estrutura do histograma após a análise do endereço.

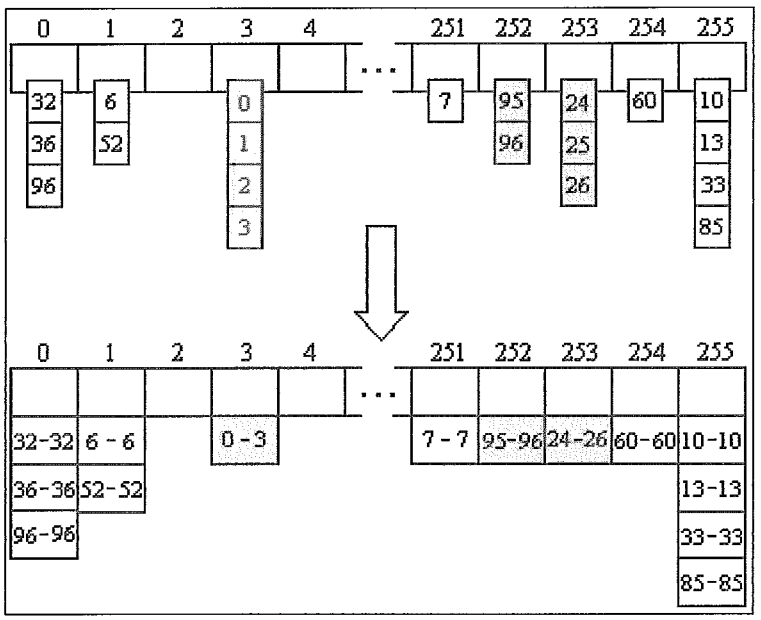

Figura 4.27: Histograma e endereço

A utilização do endereço é útil para reduzir o tempo de envio da imagem, porém trouxe um problema: tornou difícil a determinação da cor de maior ocorrência, já que não se tem mais precisão, na estrutura do histograma, de quantos *pixels* cada cor possui. Uma forma de se resolver este problema, é o uso de um algoritmo de ordenação. Por ser mais eficiente, utilizou-se o QuickSort.

4.5.3. QuickSort

Esta técnica de ordenação é uma das mais eficientes, principalmente quando há muitos elementos para serem ordenados. O QuickSort considera o elemento central de um *array*, e o subdivide em dois *arrays* menores. Um contém os valores menores que o elemento central, e o outro, os maiores. O processo é repetido, com cada novo *array*,

recursivamente, até que o novo *array* contenha apenas um elemento. Neste momento, os *arrays* de único elemento já estão ordenados corretamente (CHAN, 1999).

Para resolver o problema da determinação das cores principais, criou-se um *array* multidimensional com 256 colunas (de zero a 255) e duas linhas. A primeira linha armazena a cor, e a segunda quantos endereços aquela cor possui.

A ordenação ocorre pela segunda linha. Desta forma, após se ter o *array* ordenado, tem-se a ordem em que as cores devem ser transmitidas.

A utilização do algoritmo de QuickSort, além de resolver o problema da determinação correta da ordem de envio das cores, trouxe um ganho na velocidade, pois ordenar um *array* é muito mais rápido do que ficar buscando em toda a estrutura de vetores qual a cor que possui o maior número de *pixels*.

A técnica de endereço, aliada ao qicksort, trouxe um ganho no tempo. Porém o ideal seria que o tempo fosse reduzido ainda mais. Para diminuir a espera do usuário durante a carga da imagem, utilizou-se a subtração.

4.5.4. Subtração

A subtração consiste na determinação das diferenças entre duas imagens, e o envio apenas do que há de diferente entre elas.

O processo de subtração é feito através da comparação, *pixel* a *pixel*, das duas imagens. Os *pixels* que são iguais recebem zero. Os que são diferentes, são mantidos. Colocar zero nos *pixels* da imagem não traz nenhum problema para a montagem do vetor de histograma, pois as cores da imagem estão em um modelo padrão de cor do Java, possibilitando que o zero seja descartado.

Após realizada a subtração, determina-se o histograma, desconsiderando os *pixels* com zero. A partir daí, todo o processo ocorre como para a primeira imagem.

Em uma série (seqüência de imagens), a primeira imagem não sofre subtração, pois não há nenhuma imagem de referência para realizá-la. A segunda é comparada com a primeira, a terceira com a segunda, e assim sucessivamente até o final da série.

Esta técnica trouxe um ganho de velocidade considerável, por transmitir apenas as diferenças. Os *pixels* enviados são sobrepostos a uma cópia da referência, formando a nova imagem da série.

Apesar de todas as técnicas apresentadas, ainda há um problema: o usuário precisa esperar que todas as imagens sejam totalmente carregadas para depois selecionar

a série (no caso da tela de estudos), ou a imagem com que deseja trabalhar, no caso do visualizador. Isso é uma desvantagem, pois o tempo de espera para trabalhar com uma imagem torna-se muito grande.

Para reduzir este problema, realizou-se a redução da imagem.

4.5.5. Redução

A redução consiste na diminuição da quantidade de *pixels* que é enviado para a visualização. Isso gera perda de qualidade, já que há menos *pixels* sendo enviados, porém isso não é um problema. A partir do momento que o usuário escolhe a imagem que vai analisar, todos os seus *pixels* são enviados, não havendo mais a perda de informação. Além disso, a imagem "incompleta" só é visualizada com seu tamanho reduzido, não evidenciando a falta dos pixels.

O processo de redução ocorre da seguinte forma: a imagem original possui **w** de largura e **h** de altura. Essas duas dimensões são divididas por um fator **f**, potência de 2. Logo, a imagem original, que possui dimensões **wxh**, passará a ter **(w/f)x(h/f)**.

Por causa da redução das dimensões da imagem, seus *pixels* também devem ser escolhidos, de forma a respeitar esta redução. Essa escolha é feita de acordo com o endereço (x, y) do *pixel* na imagem. Se o módulo de **x** e de **y** pelo fator for zero, o *pixel* é levado para a imagem reduzida. A figura 4.28 ilustra a imagem antes e depois da redução.

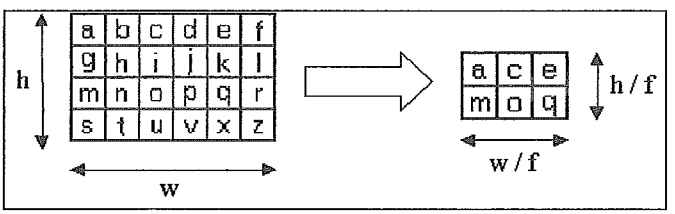

Figura 4.28: Redução da imagem com fator de redução 2

A imagem que será enviada para a Tela de Estudos e para as imagens reduzidas do Visualizador será a imagem da direita, com menos *pixels*. Como dito anteriormente, por exibirem a imagem com um tamanho menor, a perda da qualidade não é perceptível ao usuário. A figura 4.29 exibe a imagem completa, a imagem reduzida exibida na tela de estudos e a imagem do visualizador.

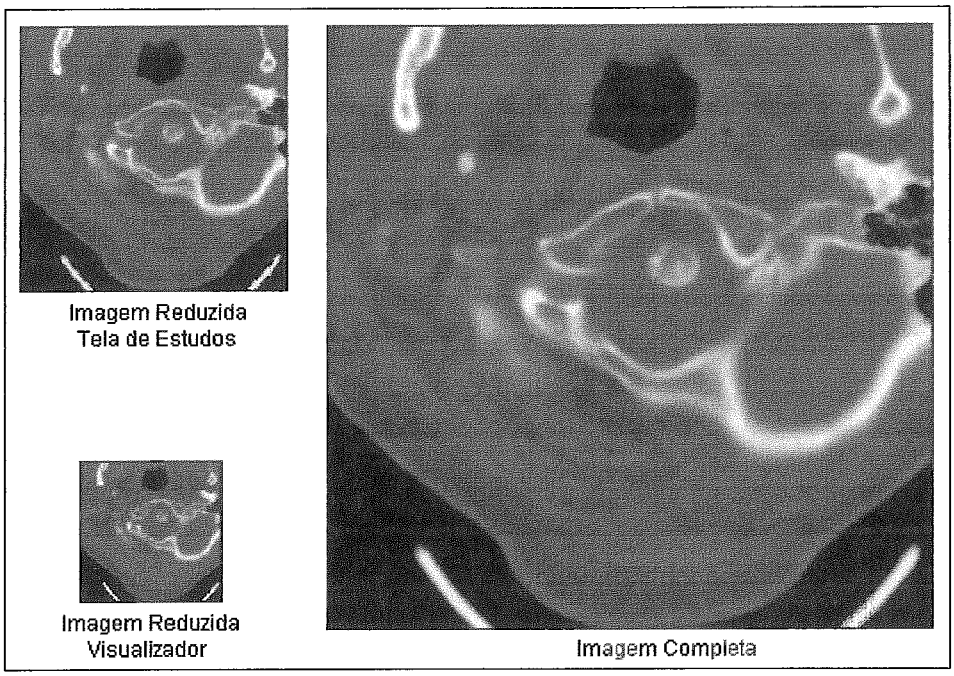

Figura 4.29: Comparação entre a imagem completa e as imagens reduzidas.

A utilização da redução traz uma desvantagem. Quando o usuário arrasta a imagem que deseja estudar para a área de visualização, ela deve ser completada, para que não possua nenhuma perda de qualidade. Essa necessidade de completar a imagem faz com que o usuário tenha que esperar um certo tempo antes de poder trabalhar com a imagem escolhida. Porém, essa desvantagem desaparece, se forem comparados o tempo de espera para que todas as imagens exibidas no lado esquerdo da janela sejam totalmente carregadas, com o tempo de realizar a carga do que falta da imagem com que se vai trabalhar.

Para completar a imagem, devem ser enviados apenas os *pixels* que não estão presentes na imagem reduzida. Na verdade, o processo é inverso ao da redução. Deve-se transformar a imagem com menos *pixels* na imagem completa, localizando corretamente os *pixels* que já haviam sido enviados.

O primeiro passo para completar a imagem é realocar os *pixels* da imagem reduzida em seus lugares corretos no *array* da imagem ampliada. A figura 4.30 ilustra qual deve ser o resultado desse processo.

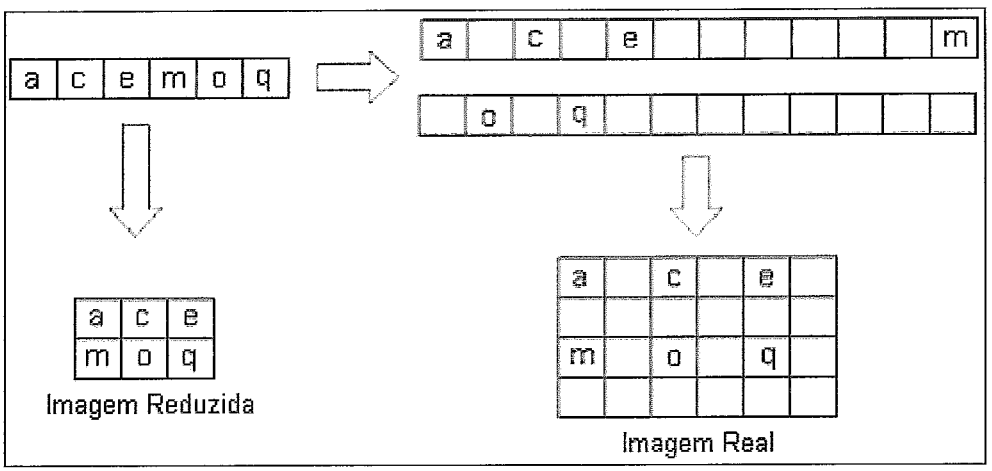

Figura 4.30: Realocando os *pixels* na imagem

A determinação da posição correta dos *pixels* é feita de acordo com sua posição no *array* da imagem que foi reduzida. O primeiro *pixel*, sempre ocupará a primeira posição do *array* da imagem de tamanho real, já que ele se localiza no ponto (0,0) de ambas as imagens. Os *pixels* seguintes são dispostos de acordo com o fator de redução[2], tendo as dimensões de altura e largura da imagem real como limite. A posição do próximo *pixel* é calculada de acordo com a posição do que foi colocado por último. Se ao somar a posição do *pixel* colocado com o fator de redução for menor que a largura da imagem real, o *pixel* é colocado nesta posição calculada. Se for maior, soma-se o valor da posição do último *pixel* colocado, com o fator e a largura da imagem real, colocando-se o novo *pixel* nesta posição. Repetindo-se o processo para todos os *pixels* do *array* da imagem reduzida, tem-se a realocação correta dos *pixels* na imagem. O pseudo código a seguir ilustra o processo.

```
contador = 0;
posição = 0;
enquanto (contador < tamanho_array_imagem_reduzida)
{
      array_img_real[posição] = array_img_reduzida[contador];
      se (posição + fator) < largura_img_real
      então posição = posição + fator
      senão posição = posição + fator + largura_img_real;
      contador = contador + 1;
}
```

Ao realocar os *pixels* no *array* da imagem de tamanho normal, e exibi-lo na janela para o usuário, a imagem apresentada fica "falhada" sendo possível perceber os

[2] O fator de redução, usado no processo de "ampliação" da imagem, é o mesmo do processo de redução. A diferença, é que quando a imagem está sendo reduzida, seus parâmetros são divididos pelo fator, e quando ela está sendo restaurada, ela é multiplicada pelo mesmo fator f.

pontos preenchidos e os espaços em branco. Para que isso não ocorra, os *pixels* que foram colocados no *array* são "propagados", de forma a preencher a imagem tornando-a um pouco mais contínua.

As cores são propagadas pelos (fator-1) *pixels*, no sentido horizontal, e pelas (fator-1) linhas (sentido vertical), formando um quadrado de mesma cor. A figura 4.31 ilustra o resultado. Nela, o fator é 2, fazendo com que os *pixels* sejam propagados uma unidade na horizontal, e uma na vertical. As cores mais escuras representam os *pixels* originais. As mais claras, os que foram propagados.

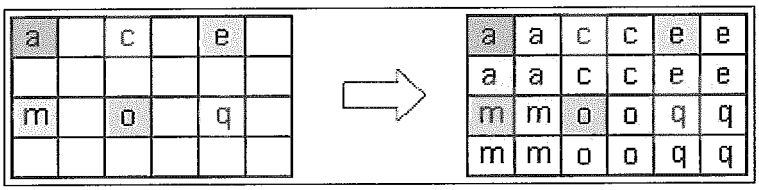

Figura 4.31: Completando a imagem

Desta forma, apesar de não se ter uma boa definição, tem-se a imagem sem espaços em branco, permitindo que o usuário já tenha uma noção maior de como a imagem é. Na figura 4.32, tem-se a imagem com os espaços em branco (à esquerda) e sem os espaços em branco (à direita).

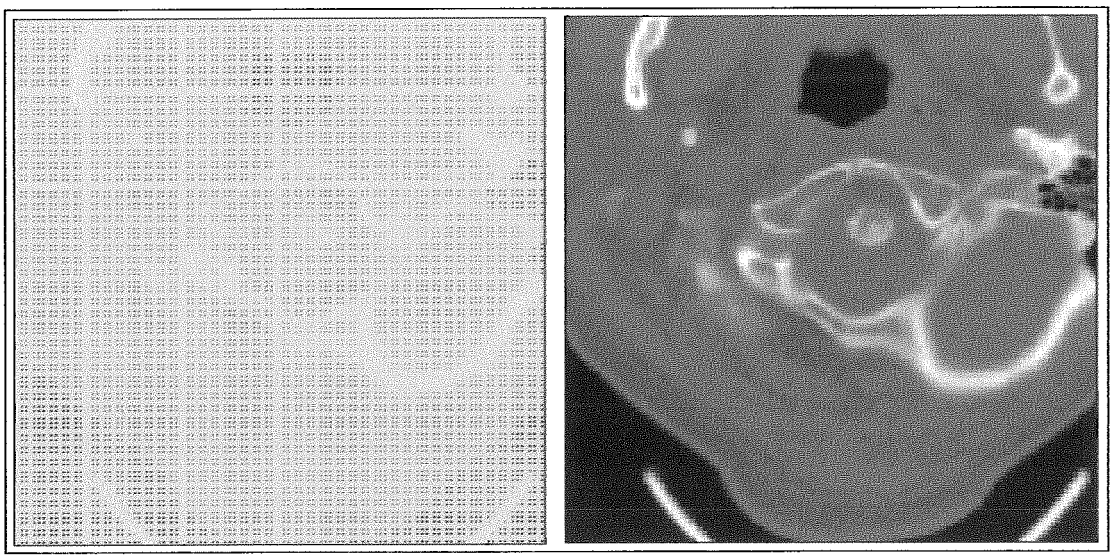

Figura 4.32: Resultado da "propagação"

Após propagar os *pixels*, é necessário que o *servlet* envie para a *applet* aqueles que não foram carregados para que a imagem seja completada. O processo para selecionar os *pixels* que devem ser enviados é o contrário do realizado para reduzir a imagem.

O *servlet* enviará para a *applet* todos os *pixels* cujo módulo de suas coordenadas x e y pelo fator não são zero. À medida que os *pixels* são enviados, eles são colocados nas posições corretas, substituindo aqueles que eram brancos e receberam valores durante a propagação de *pixels*. Assim, após todos os que faltavam terem sido enviados, a imagem fica completa, não havendo mais nenhuma perda de qualidade. A figura 4.33 exibe o resultado final da carga da imagem.

Figura 4.33: Imagem totalmente carregada

Esta imagem completa é a que será utilizada pelo usuário para análise. Todas as funcionalidades apresentadas anteriormente (como brilho, contraste, rotação, etc) podem ser aplicadas sobre esta imagem, sem ocasionar perda de definição (qualidade), ou outro tipo de problema.

Como visto neste capítulo, a técnica de *streaming* implementada neste trabalho é composta pelo histograma, endereço, quicksort, subtração e redução. Essas técnicas são aplicadas na seguinte ordem: redução, subtração, histograma, quicksort e endereço. Vale lembrar, que as imagens exibidas na tela de estudos e a primeira imagem de cada série (na tela de visualização), não sofrem subtração.

Capítulo 5

Considerações Finais

Atualmente, a grande quantidade de imagens que é gerada em exames de diagnóstico por imagem (como Raio-X e ressonância magnética) tornou complicada a manipulação e o armazenamento destas imagens. A alternativa encontrada, com o desenvolvimento tecnológico, foi a utilização de imagens digitais, por permitirem uma manipulação mais eficiente através de dispositivos conectados em rede. Porém, para que esta tecnologia seja bem aceita no meio médico-hospitalar, é necessário que sejam considerados a facilidade de uso, a velocidade e a segurança de acesso às imagens, e a qualidade durante a apresentação das mesmas.

Para atender a estes requisitos, surgiram os sistemas denominados *Picture Archiving Communication System* (PACS), responsáveis pela comunicação, distribuição e armazenamento das imagens digitais. Porém, existem diversos fabricantes de equipamentos hospitalares, e em uma mesma clínica ou hospital é comum a presença de equipamentos de mais de um fabricante. Por isso, para permitir a comunicação entre os diversos equipamentos, o *National Electrical American College of Radiology* (ACR) em conjunto com *o National Electrical Manufacturers Association* (NEMA) padronizou os sistemas PACS.

Esta padronização levou à criação de um novo padrão internacional, específico para a parte de rede: o *Digital Imaging and Communications in Medicine* (DICOM). Inicialmente, este padrão especificava a comunicação em uma rede ponto-a-ponto. Porém, com o avanço da tecnologia de rede, tornou-se necessária a atualização deste padrão. Sua última versão especifica a comunicação em uma rede ponto-a-ponto, em redes que utilizam o protocolo TCP/IP, além de redes que utilizam o padrão ISO/OSI (*International Standards Organization / Open Systems Interconnection*). Isso torna o DICOM simples e utilizável.

Porém, não existem apenas os sistemas PACS em um hospital. A necessidade de melhor gerenciamento das mais diversas áreas hospitalares levou ao surgimento de diversos sistemas independentes. Para integrar estes sistemas, permitindo a troca de informações entre eles, surgiram os *Hospital Information System* (HIS). Estes sistemas integram todas as informações de uma clínica ou hospital. Para padronizar o HIS foi

criado o padrão internacional *Heath Level Seven* (HL7). O HL7 especifica a forma de comunicação e de troca de informações entre os mais diversos sistemas hospitalares.

A utilização de informações digitais trouxe um grande avanço para a área médica. Porém, principalmente no caso das imagens, trouxe alguns problemas também. Em geral, as imagens digitais são muito grandes, tornando muito custosa sua transmissão via rede. Além disso, o tempo necessário para realizar o *download* da imagem, e visualizá-la, torna-se muito grande.

Essa situação não ocorre apenas na área médica. Na Internet, é comum haver este problema para ouvir uma música, ou assistir a um vídeo, por exemplo. A solução utilizada, na Internet, foi a criação de uma técnica, denominada *streaming*, que permite a execução do arquivo desejado, antes mesmo que seu *download* tenha sido completado. Isso reduz a sensação do tempo de espera do usuário.

A técnica de *streaming* promove a segregação e compressão do arquivo, para que seja enviado aos pedaços. No computador cliente, o arquivo sofre uma convolução, e sua exibição e/ou execução é iniciada antes que todo o arquivo esteja no computador cliente.

O objetivo desta dissertação foi desenvolver uma técnica de *streaming* para ser aplicada às imagens médicas digitais sobre o padrão DICOM. A técnica foi implementada utilizando-se o histograma da imagem, além das técnicas de endereço, quicksort, subtração e redução, em uma metodologia unificada[1].

O histograma foi utilizado para determinar as partes mais significativas das imagens, para que estas fossem enviadas ao usuário primeiro. Desta forma, uma imagem significativa é fornecida mais rapidamente.

Outra técnica utilizada foi o endereço, que constitui uma compressão sem perda. O endereço reduz a quantidade de pixels a ser enviada, ao transmitir apenas o primeiro e o último pixels quando há uma seqüência de pixels consecutivos. Não foi utilizada nenhuma técnica de compressão com perda, pois as imagens médicas, devido à sua própria natureza, não podem sofrer a perda de nenhuma informação, pois isso significa redução na qualidade, prejudicando a análise da imagem.

[1] A ordem de aplicação dessas técnicas é: redução, subtração, histograma, quicksort e endereço lembrando que a subtração é aplicada apenas nas imagens que aparecem no visualizador, exceto a primeira imagem da série

Para tornar a escolha da ordem de envio das cores mais rápida, foi utilizada a técnica de ordenação por quicksort. Ele ordena as cores a serem enviadas de acordo com a quantidade de pixels que cada cor possui.

A técnica de subtração foi utilizada visando uma melhor utilização da rede. Em uma série[2], as imagens são muito semelhantes. Por isso, a partir da segunda imagem, é transmitido apenas o que há de diferente entre a imagem que vai ser enviada e a anterior. Isso é possível porque a técnica de subtração identifica os pixels diferentes, para serem enviados, e, no cliente, a imagem anterior é "copiada", e os pixels diferentes são sobrepostos.

A metodologia redução foi utilizada para diminuir ainda mais a sensação de espera do usuário. Na interface com o usuário, há uma parte onde as imagens da série são exibidas em um tamanho menor, para a escolha a imagem a ser visualizada. Essas imagens são reduzidas de forma proporcional, no momento do envio, visando diminuir a quantidade de pixels a serem transmitidos. Sendo essas imagens exibidas em tamanho menor, a perda de informação não é percebida pelo usuário. No momento em que vai ser visualizada, os pixels que "faltam" são enviados, eliminando a perda de pixels, e garantindo a qualidade da imagem que será visualizada.

O ganho de tempo[3] com a técnica de *streaming* implementada pode ser visualizado na tabela 5.1. O computador utilizado como servidor foi um Pentium III, 1GHz, com 256Mb de memória RAM. Este computador trabalhou como servidor *web*, servidor de banco de dados e servidor de aplicação.

Tabela 5.1: Comparação dos tempos de carga

Tipo de Exame	Tempo Médio de Carga	Tempo Médio com Streaming MedWeb	Ganho
Tomografia Computadorizada	36 seg	6 seg	83%
Raio X	105 seg	37 seg	65%
Ressonância Magnética	30 seg	7 seg	77%

[2] Ao realizar um exame, é criado um estudo para o paciente. O estudo possui uma ou mais séries. Cada série possui uma ou várias imagens, que são as imagens digitais.

[3] Para a análise de ganho no tempo, foram utilizadas imagens DICOM de exames de raio X, ressonância magnética, e tomografia computadorizada. No total, foram utilizadas cerca de 20 imagens.

Além da técnica de *streaming*, que permite uma visualização mais rápida da imagem, foi desenvolvido um ambiente para a manipulação das mesmas. Este ambiente procurou implementar funcionalidades que simulam a manipulação da imagem em filmes radiológicos. Com isso, foram criadas, por exemplo, funções de rotação de 90° (à direita e à esquerda) de 180° (horizontal e vertical). Além disso, foram criadas outras funcionalidades, que não podem ser simuladas nos filmes radiológicos. Por exemplo, a inversão de cores e o zoom da imagem.

Para que fosse possível a implementação da técnica de *streaming* e de todas as funcionalidades de manipulação de imagens, fornecendo facilidade, rapidez e segurança ao usuário, foi necessária a utilização de uma arquitetura que atendesse a todos esses requisitos. A arquitetura escolhida foi a de quatro camadas. Nela, a interface com o usuário foi retirada do cliente, e centralizada em um servidor *web*. Assim, o cliente passou a necessitar de apenas um navegador para ter acesso ao programa. A grande vantagem desta arquitetura é a facilidade de gerenciamento das versões do *software*, e a centralização das funcionalidades: o servidor de aplicação contém a lógica do negócio; o servidor de banco de dados trata a persistência dos dados; o servidor *web* centraliza a interface com o usuário; e o cliente tem a garantia de estar acessando sempre a versão mais atual do *software*. A desvantagem desta arquitetura é a quantidade de servidores que se tem para gerenciar.

Alguns trabalhos futuros podem ser realizados a partir do que foi apresentado nesta dissertação. Por exemplo, podem-se aplicar outras técnicas de compressão sem perda, para reduzir a quantidade de pixels a serem enviados ao cliente, reduzindo o tempo de espera. Outra possibilidade é desenvolver mais a região de interesse (ROI – *Region of Interest*). O ROI consiste no envio apenas dos pixels necessários para que a imagem seja visualizada sem perda de qualidade. Por exemplo, se é realizado um *zoom* em apenas uma parte da imagem, apenas os pixels deste pedaço seriam enviados. Isto é feito, atualmente, apenas com as imagens reduzidas da série que são fornecidas ao usuário para que ele escolha com qual deseja trabalhar.

Ainda na técnica de *streaming*, pode-se testar se o envio da imagem é mais eficiente se for passada para um formato intermediário, como o GIF, por exemplo. O formato de imagem GIF trabalha com 256 cores, não necessariamente contínuas, comprimidas pela técnica LZW. Como as imagens médicas possuem tons de cinza que variam de 0 a 255, tem-se as 256 cores que podem ser utilizadas no GIF sem perda de informação.

Com relação à interface com o usuário, podem-se acrescentar mais funcionalidades, como, por exemplo, inserir a dinâmica temporal, como em um filme. Esta funcionalidade exibe as imagens de séries temporais de uma forma seqüencial, com um intervalo de tempo pré-determinado entre elas. Isso proporciona, ao usuário, a sensação de estar assistindo a um filme.

Pode-se criar mais classes de acesso a outros bancos de dados, como o PostGres por exemplo, tornando a aplicação mais genérica.

Um outro trabalho que pode ser desenvolvido é o trabalho colaborativo, onde, por exemplo, dois usuários acessam uma mesma imagem, um podendo visualizar, em tempo real, as ações do outro.

Referências Bibliográficas

BATTISTI, JÚLIO, 2002a. "WebMaster - Em Primeiro Lugar: O Inferno". *Developers'*, ano 6, nº 72 (Agosto) – pp.41-41.

BATTISTI, JÚLIO, 2002b. "WebMaster - Rumo ao Paraíso". *Developers'*, ano 6, nº 73 (Setembro), p.44-45.

CHAN, MARK C., GRIFFITH, STEVEN W., IASI, ANTHONY F, 1999. *Java – 1001 Dicas de Programação*, 1ed, São Paulo, Makron Books.

CURSO IBM de Programação, 1999, n 44 a 51, Editora Planeta.

GANNOT, ISRAEL, 2003. *Hospital Information System* – Disponível na Internet: http://www.eng.tau.ac.il/~gannot/MI/HISHL7.ppt [capturado em 27 abr 2003]

GUALBERTO, E. F., SATO, J., PEIXOTO, M. F., et al. *Iniciativas Nacionais e Internacionais de Integração de Sistemas de Informação em Saúde – Health Level Seven* – Disponível na Internet: http://www.virtual.epm.br/material/tis/curr-med/temas/med5/med5t12000/pe/hl7.htm [capturado em 23 de março de 2003]

HL7 ORG, 2003 – Disponível na Internet: http://www.hl7.org/about/ [capturado em 23 de março de 2003]

HORII, STEVEN C, 2002. *A Nontechnical Introduction to DICOM* – Disponível na Internet: www.rsna.org/REG/practiceres/dicom/nontechintro.html [capturado em 08 maio 2002]

HUNTER, JANE; WITANA, VARUNI; ANTONIADES, MARK, 1997. *A Review of Video Streaming over the Internet.*

HUNTER, JASON, CRAWFORD, WILLIAM, 2002. **Java** *Servlet: Programação* –, 1ed, Rio de Janeiro, Ciência Moderna

INEI, 2002 – *Instituto Nacional de Estadística e Informática – Perú.* Disponível na Internet: http://www.inei.gob.pe/cpi-mapa/bancopub/libfree/lib616/INDEX.HTM [capturado em 18 set. 2002]

KENNEDY, TIM, 1999. *What Is Streaming Media?* – Disponível na Internet: http://www.streamingmediaworld.com/gen/tutor/whatis/ [capturado em 05 jan 2003]

KODAK, 2003. *Digital Learning Center* – Disponível na Internet: http://www.kodak.com/US/en/digital/dlc/book2/chapter4/formatp1.shtml [capturado em 11 jan. 2003]

LEAD TECHONOLGIES, 2003a. *Overview: Basic DICOM File Structure* – Disponível na Internet: http://www.leadtools.com/SDK/Medical/DICOM/ltdc1.htm – [capturado em 12 jan 2003]

LEAD TECHNOLOGIES, 2003b. The *DICOM Medical Imaging Standard* – Disponível em: http://www.leadtools.com/SDK/Medical/DICOM/dicomstnd.htm – [capturado em 12 jan 2003]

MARTÍNEZ, ALFONSO M.; JIMÉNEZ, JUAN R.; MEDINA, VERÓNICA; LEEHAN, JOAQUÍN A, 2003. *Los Sistemas PACS* – Disponível na Internet: http://itzamna.uam.mx/alfonso/pacs.html [capturado em 27 abr 2003]

MILBURN, KEN, 2003. *JPEG2000: the Killer Image File Format for Lossless Storage* – Disponível na Internet: http://www.oreillynet.com/pub/a/javascript/2003/11/14/digphoto_ckbk.html [capturado em 05 de fev 2004]

OLIVEIRA, KEPLER S., 2003 – *Universidade Federal do Rio Grande do Sul: Fundamentos de Radiodiagnóstico por Imagem.* Disponível na Internet: http://www.if.ufrgs.br/ast/med/imagens/node40.htm [capturado em 05 jan. 2003]

REALTIME IMAGE, 2003. *iPACS Enterprise Product Description.* Disponível na Internet: http://www.realtimeimage.com/medical/technology/ [capturado em 01 fev. 2003]

REZENDE, JOFFRE M., 2002. *Caminhos da Medicina – O Uso da Tecnologia no Diagnóstico Médico e suas Conseqüências.* Disponível na Internet: http://usuarios.cultura.com.br/jmrezende/tecnologia.htm [capturado em 06 de abril de 2003]

RICKARDS, TONY, 2003. *Dicom 96 Tutorial* – Disponível na Internet: www.uni-mainz.de/Cardio/dicom/tutorial.html [capturado em 13 fev. 2003]

RORDEN, CHRIS, 2002. *The DICOM Standard* – Disponível na Internet: http://www.psychology.nottingham.ac.uk/staff/cr1/dicom.html [capturado em 05 set. 2002]

STREAMINGMEDIA, 2002. *University of Wisconsin – Understanding Streaming Media – Introduction to Streaming Tutorials* - Disponível na Internet: http://streaming.doit.wisc.edu/tutorial/tutorial1.htm

WAYNE, FULTON, 2003. *A Few Scanning Tips.*. Disponível na Internet: http://www.scantips.com/basics09.html [capturado em 11 jan. 2003]

ZOOMIFY, 2003. *Zoomifyer Authoring Program – User's Guide.* Disponível na Internet: http://www.zoomify.com/downloads/ [capturado em 01 fev. 2003]