PAULO JORGE CÂMARA PIZARRO

MonitorIP - MONITORAMENTO DE SINAIS VITAIS ATRAVÉS DE UMA REDE IP

FLORIANÓPOLIS
2003

UNIVERSIDADE FEDERAL DE SANTA CATARINA

PROGRAMA DE PÓS-GRADUAÇÃO
EM ENGENHARIA ELÉTRICA

MonitorIP - MONITORAMENTO DE SINAIS VITAIS ATRAVÉS DE UMA REDE IP

Dissertação submetida à
Universidade Federal de Santa Catarina
como parte dos requisitos para a
obtenção do grau de Mestre em Engenharia Elétrica.

PAULO JORGE CÂMARA PIZARRO

FLORIANÓPOLIS, junho de 2003

MonitorIP - MONITORAMENTO DE SINAIS VITAIS ATRAVÉS DE UMA REDE IP

Paulo Jorge Câmara Pizarro

'Esta Dissertação foi julgada adequada para a obtenção do Título de Mestre em Engenharia Elétrica, Área de Concentração em Engenharia Biomédica, e aprovada em sua forma final pelo Programa de Pós-Graduação
em Engenharia Elétrica da Universidade Federal de Santa Catarina'

_____
Jefferson Luiz Brum Marques, PhD.
Orientador

_____
Edson Roberto de Pieri, Dr.
Coordenador do Programa de Pós-Graduação
em Engenharia Elétrica

Banca Examinadora:

_____
Jefferson Luiz Brum Marques, PhD.
Presidente

_____
Fernando Mendes de Azevedo, Dr.

_____
Raimes Moraes, PhD.

_____
Elizabeth Sueli Specialski, Dr.

À minha querida esposa, Mariana, por seu amor, apoio e incentivo constante, os quais foram fundamentais para a conclusão desse trabalho.

AGRADECIMENTOS

Gostaria de agradecer aos professores Raimes Moraes e Fernando Mendes de Azevedo pelo interesse e pelas discussões sobre partes deste trabalho.

Aos caros colegas de trabalho, Pantaleão, Humberto, Mauro, Marcus e Daniel e aos demais amigos do GPEB pelo companheirismo, entusiasmo e colaboração, que tornaram o ambiente de trabalho mais ameno.

Ao IEB/UFSC pela infra-estrutura e apoio e à secretaria da PGEEL, em especial ao Wilson e ao Marcos, pelo apoio.

Em especial, agradeço a orientação do professor Jefferson Luiz Brum Marques.

[1] PIZARRO, Paulo; LOPES, Cássio; MORAES, Raimes; AZEVEDO, Fernando Mendes de; MARQUES, Jefferson Luiz Brum. **Monitoramento Remoto de Sinais Bioelétricos.** In: II Congreso Latinoamericano de Ingeniería Biomédica. Habana, Cuba, May., 2001, v.1.

Resumo da Dissertação apresentada à UFSC como parte dos requisitos necessários para a obtenção do grau de Mestre em Engenharia Elétrica.

MonitorIP - MONITORAMENTO DE SINAIS VITAIS ATRAVÉS DE UMA REDE IP

Paulo Jorge Câmara Pizarro

junho / 2003

Orientador: Jefferson Luiz Brum Marques, PhD.
Área de Concentração: Engenharia Biomédica
Palavras Chave: Monitoramento, RTP, RTCP, IP, telemedicina, telecirurgia, home-care
Número de Páginas: 112

RESUMO: O MonitorIP é um monitor de multi-parâmetros remoto, o qual possibilita o monitoramento em tempo-real de sinais vitais de pacientes através de uma rede IP (*Internet Protocol*), como a Internet. Aplicações típicas do MonitorIP são: telemedicina, telecirurgia, telemonitoramento, home-care, automação de unidades de tratamento intensivo entre outras. Em intervalos regulares, pequenos segmentos de sinais bioelétricos são digitalizados e transmitidos para o destino, onde o sinal será visualizado e analisado. Para uma boa interatividade do monitoramento, o atraso total da transmissão de cada pacote deve ser baixo. A variação do atraso da transmissão (*jitter*) foi eliminada através de técnicas de *buffering* e a perda de pacotes e a quantidade de pacotes com erro devem ser baixas, para que não comprometa a qualidade do sinal. O IP, um dos protocolos da arquitetura TCP/IP, oferece um serviço sem conexão baseado em datagramas, o qual não garante a entrega dos mesmos a tempo, na ordem correta e nem mesmo a entrega ao destino. Além do mais, cada datagrama leva tempos diferentes para chegar ao seu destino. Esse tipo de serviço sem conexão é conhecido como serviço de melhor esforço (*best-effort*). Como solução, as aplicações tradicionais, normalmente, utilizam o protocolo de transporte TCP (*Transmission Control Protocol*), o qual oferece um serviço seguro e confiável de entrega de pacotes. Na transmissão dos sinais vitais, o protocolo TCP não é uma boa escolha, pois ele tem uma série de características desnecessárias no monitoramento em tempo-real, as quais só incrementam o atraso total. Portanto, o sistema MonitorIP utilizou o protocolo UDP (*User Datagram Protocol*), ao invés do TCP pela maior simplicidade e menor *overhead* desse protocolo. O problema de sequenciamento foi resolvido pela utilização do protocolo RTP (*Real-Time Transport Protocol*), o qual fornece informações para sincronização, controle e identificação de fluxo e de congestionamento.

Abstract of Dissertation presented to UFSC as a partial fulfillment of the requirements for the degree of Master in Electrical Engineering.

MonitorIP - MONITORING VITAL SIGNALS THROUGH AN IP NETWORK

Paulo Jorge Câmara Pizarro

June / 2003

Advisor: Jefferson Luiz Brum Marques, PhD.
Area of Concentration: Biomedic Engineering
Keywords: Monitoring, RTP, RTCP, IP, telemedicine, telesurgery, home-care
Number of Pages: 112

ABSTRACT: MonitorIP is a remote multi-parameter monitor, which allows the remote real-time monitoring of a patient vital signals using an IP (Internet Protocol) network, for example the Internet. The typical applications of the MonitorIP are: telemedicine, telesurgery, telemonitoring, home-care, intensive care unit (ICU) automation and others. At regular intervals, short segments of bioelectrical signals are digitalized and transmitted to the destination where they are displayed. For a real-time monitoring, the overall delay should be low. Delay variance or jitter should be eliminated through buffering. The loss of the information packets and the quantity of the corrupted packets should be low, to keep the quality of the signal. The IP is part of the TCP/IP architecture, which offers a service without connection based on datagrams. The IP does not guarantee the delivery on time, in the correct order and neither the delivery at all. Moreover, each packet takes a different amount of time to reach its destination. This type of service without connection is called best-effort service. The classical applications normally do not use the IP itself, but the higher-level protocols: TCP, which offers a reliable *byte* stream service, and UDP which offers a similar service to IP. To transmit the vital data, TCP is not a good choice since it has a lot of features which are unnecessary for real-time monitoring, increasing the overall delay and does not allow synchronization. The MonitorIP uses the Real-Time Transport Protocol (RTP), which provides information for synchronization, flow and traffic control and identification. The UDP protocol is used as a transport protocol for the RTP protocol.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

1. INTRODUÇÃO

Neste capítulo será feita uma introdução do sistema MonitorIP, além de uma descrição de como está organizada esta dissertação.

Com o avanço da tecnologia de comunicações, novas possibilidades têm surgido para a medicina, no sentido de se monitorar pacientes à distância. Pacientes em recuperação pós-operatória são monitorados por monitores de multi-parâmetros em unidades de tratamento intensivo (UTI); nesta situação, eles ficam sujeitos a infecção hospitalar, além dos altos custos financeiros de sua permanência no hospital. A proposta desta dissertação está inserida na utilização das tecnologias de redes de computadores existentes, para que pacientes recuperando-se em suas casas possam ser monitorados remotamente pela equipe médica situada em hospitais ou em clínicas. Com isso, os riscos de infecções hospitalares diminuiriam e os pacientes teriam uma melhor recuperação por estarem pertos de seus familiares. O objetivo é realizar pesquisa e desenvolvimento para possibilitar o monitoramento em tempo-real dos sinais vitais de pacientes através de uma rede IP (Internet Protocol). Parâmetros como o eletrocardiograma (ECG), pressão arterial não invasiva (EBP), oximetria (SPO_2), capnografia (CO_2), respiração, freqüência cardíaca e temperatura estarão disponíveis remotamente no computador da unidade de monitoramento e/ou *notebook* do médico.

Os aspectos de tempo-real são importantes, principalmente em situações como em uma telecirurgia, onde o paciente precisa de uma intervenção imediata do médico, caso alguns de seus dados fisiológicos apresentem uma anormalidade. O atraso máximo do envio de tais dados deve ser tão pequeno quanto possível. Como veremos durante esta dissertação, para atender os requisitos temporais, realizar um controle e identificação do fluxo e do congestionamento, e ainda sincronizar os diversos sinais entre si, o sistema MonitorIP utilizou os protocolos RTP (Real-Time Transport Protocol) e RTCP (Real-Time Control Protocol) de rede de computadores, os quais fornecem características favoráveis para os requisitos citados acima.

1.1 Objetivos

Os principais objetivos deste trabalho são:

- Definir um perfil de utilização do protocolo RTP (*Real-Time Transport*

Protocol) para o transporte de sinais bioelétricos através de uma rede de computadores, viabilizando o monitoramento *on-line* de pacientes.

- Desenvolver um software responsável pela aquisição de sinais vitais e pela transmissão dos mesmos, através de uma rede IP, de acordo com o perfil especificado para o protocolo RTP.
- Desenvolver um software responsável pela recepção e exibição dos sinais vitais transmitidos pelo software citado acima através de uma rede IP.
- Implementar técnicas de bufferização de pacotes RTP para minimizar os efeitos de *jitter* de uma rede IP.
- Implementar o esquema de endereçamento *multicasting* para minimizar a quantidade de largura de banda necessária no monitoramento.
- Implementar algoritmos de sincronização durante a exibição dos sinais bioelétricos de um mesmo paciente, utilizando para tal, dados fornecidos pelo protocolo de controle RTCP (*RTP Control Protocol*).
- Desenvolver um software para avaliar a qualidade de transmissão de pacotes RTP através da rede, utilizando para tal, dados fornecidos por pacotes RTCP.

1.2 Uma visão Darwiniana do monitoramento de pacientes

Os monitores de multi-parâmetros são destinados à ambientes hospitalares para a monitorização de sinais vitais de pacientes adultos, pediátricos e neonatais. Na Figura 1.1 podemos visualizar um monitor de multi-parâmetros sendo utilizado durante uma cirurgia.

Figura 1.1 Monitor de Multi-Parâmetros sendo utilizado em uma cirurgia.

Os sinais bioelétricos são apresentados no vídeo colorido em formato de curvas de forma de onda e em valores numéricos. Os principais parâmetros monitorados pelos monitores de multi-parâmetros são: eletrocardiograma (ECG), pressão arterial não invasiva (EBP), oximetria (SPO2), capnografia (CO2), respiração, freqüência cardíaca, temperatura e gases anestésicos. Durante uma cirurgia o médico observa no monitor de multi-parâmetros os sinais vitais do paciente com a finalidade de verificar de forma imediata seu estado geral.

A Figura 1.2 mostra o esquema de montagem tradicional de uma central de monitoramento de pacientes internados em uma unidade de terapia intensiva (UTI). Cada monitor de multi-pârametros, junto à cama do paciente, conecta-se a central através de um cabo. Essa montagem, apesar de ser uma tecnologia antiga, tem muitas vantagens: ela é simples e o atraso fim a fim na transmissão dos dados do monitor até a central é muito pequeno. É também uma tecnologia de baixo custo quando existe apenas uma única central de monitoramento e a distância entre os monitores e a central é pequena.

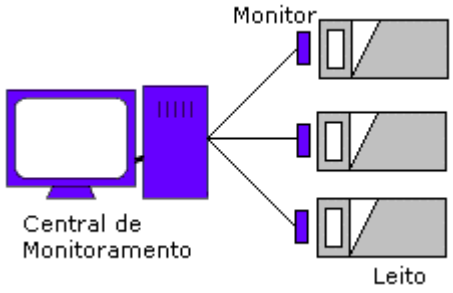

Figura 1.2 Esquema de montagem tradicional de uma central de monitoramento.

Essa tecnologia requer um cabo para cada conexão monitor-central, tornando-a rapidamente impraticável e cara com o aumento do número de centrais (pontos de monitoramento). No esquema representado na Figura 1.3, adicionou-se um novo ponto de monitoramento, por exemplo na sala de médicos. Pode-se observar o aumento significativo de cabos de conexão, agora dois para cada monitor, o que obriga a este ter duas interfaces seriais.

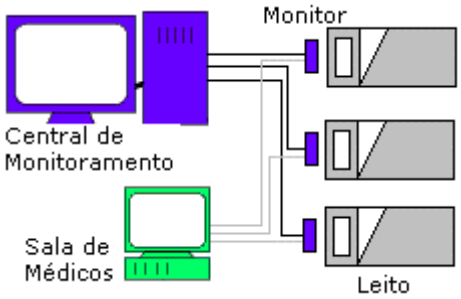

Figura 1.3 Esquema de montagem tradicional com dois pontos de monitoramento.

Com a popularização das redes de computadores, tornou-se comum encontrar redes de computadores espalhadas pelas empresas, o que não é diferente dentro do ambiente hospitalar. O sistema MonitorIP, desenvolvido nesta dissertação, aproveita essa estrutura de rede de computadores disponível dentro dos hospitais e torna viável o monitoramento remoto por mais de uma central ou unidade de monitoramento remota.

No cenário apresentado na Figura 1.4, todos os monitores de multi-parâmetros estão conectados a um mesmo computador, que está próximo ao leito dos pacientes. Esse computador conectado à rede local do hospital faz o papel de um servidor em uma arquitetura cliente/servidor. Ele é responsável por receber os dados dos monitores de multi-parâmetros e de transmiti-los para cada unidade de monitoramento (cliente) conectada ao servidor. O que será apresentado nesta dissertação é uma descrição da arquitetura do sistema MonitorIP e os protocolos de

redes de computadores utilizados, com o objetivo de monitorar pacientes em tempo-real através de redes de computadores, que utiliza o protocolo Internet Protocol (IP), como por exemplo a Internet.

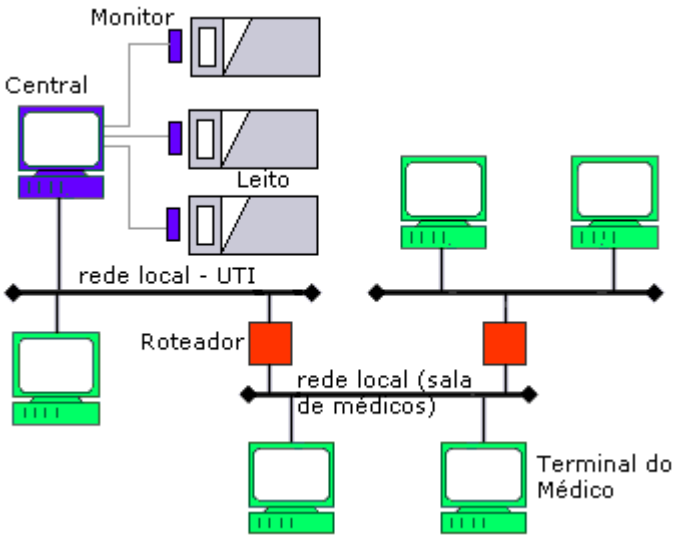

Figura 1.4 Esquema de montagem do sistema MonitorIP com vários pontos de monitoramento.

O avanço da tecnologia de sistemas embutidos com processadores cada vez mais poderosos, com maior capacidade de memória, de tamanhos cada vez menores, capazes de executar sistemas operacionais abertos, como o Linux, e com suporte à rede, facilitou aos fabricantes de monitores de multi-parâmetros a incorporação de tais processadores em seus equipamentos, além do fornecimento de conectividade à rede de computadores.

Nesse novo cenário, como se pode observar na Figura 1.5, cada monitor de multi-parâmetros está conectado à rede local do hospital e atua como um servidor independente. Cada monitor apresenta um sistema embutido e possui um sistema MonitorIP implantado, o qual é responsável por transmitir os sinais para cada uma das unidades de monitoramento conectadas a ele. A diferença básica para o esquema anterior, é que antes só existia um único servidor para todos os monitores, o que obrigava a necessidade de um mecanismo com o qual as unidades de monitoramento informassem ao servidor qual monitor elas gostariam de monitorar. Nesse novo esquema não existe a necessidade de tal mecanismo, pois cada servidor representa um único monitor de multi-parâmetros.

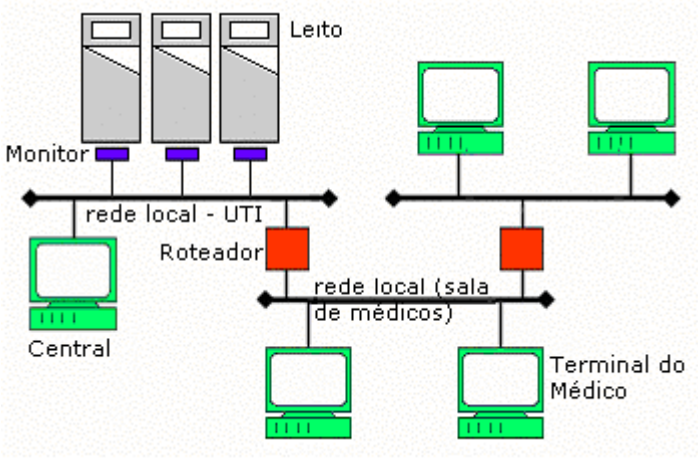

Figura 1.5 Esquema de montagem mostrando cada monitor conectado diretamente à rede

Com as tecnologias emergentes de redes de computadores sem fio (*Ethernet* 802.11h, *Bluetooth*), juntamente com a de sistemas embutidos, adicionou-se um grau de mobilidade ao monitoramento. Pacientes carregando módulos portáteis agora são monitorados enquanto caminham pelos corredores do hospital. Cada módulo portátil possui um sistema embutido, microprocessado rodando Linux e executando o sistema MonitorIP. A transmissão dos dados é feita através de um dispositivo de rede sem fio, o qual utiliza rádio freqüência com técnicas de modulação por espalhamento espectral. As vantagens são minimização da interferência em outros equipamentos, bem como imunidade às interferências externas. Tais dispositivos são portanto recomendados para o uso dentro de ambientes hospitalares. A Figura 1.6, apresenta um esquema de montagem utilizando redes sem fio.

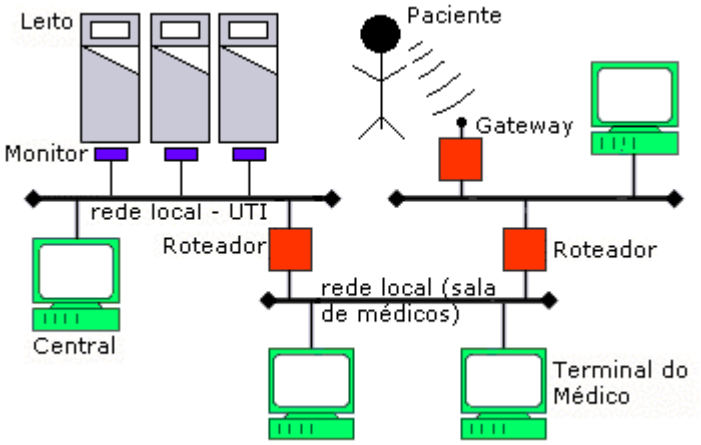

Figura 1.6 Esquema de montagem mostrando o monitoramento de pacientes utilizando conexões sem fio.

Hoje em dia o sistema de rede está muito bem estruturado. Ele utiliza uma

abordagem em camadas, onde cada uma fornece uma certa funcionalidade, a qual será usada pela camada imediatamente acima. Umas das vantagens dessa abordagem é a portabilidade. O sistema MonitorIP não foi implementado para um tipo de hardware e meio físico específicos por onde os dados são transmitidos, ou seja, como o software de rede é implementado em camadas, simplesmente troca-se o software da camada de acesso à rede para suportar um novo hardware e meio físico de transmissão. Para tal, basta apenas instalar o módulo (*driver*) no sistema operacional referente ao novo hardware e configurá-lo, sem a necessidade de se fazer qualquer alteração no software.

O MonitorIP utiliza como protocolo da camada de Rede o protocolo Internet Protocol (IP), de onde surgiu a sigla IP da palavra MonitoIP. Dessa forma, pode-se utilizar o sistema em qualquer rede que utilize o protocolo IP. Na Figura 1.7 é apresentado um esquema de montagem do monitoramento através da Internet. Assim, se o hospital está conectado à Internet, ou pertence a uma rede coorporativa interligando diversos hospitais, um especialista localizado em outro hospital analisaria remotamente os sinais bioelétricos de um paciente e auxiliaria no diagnóstico.

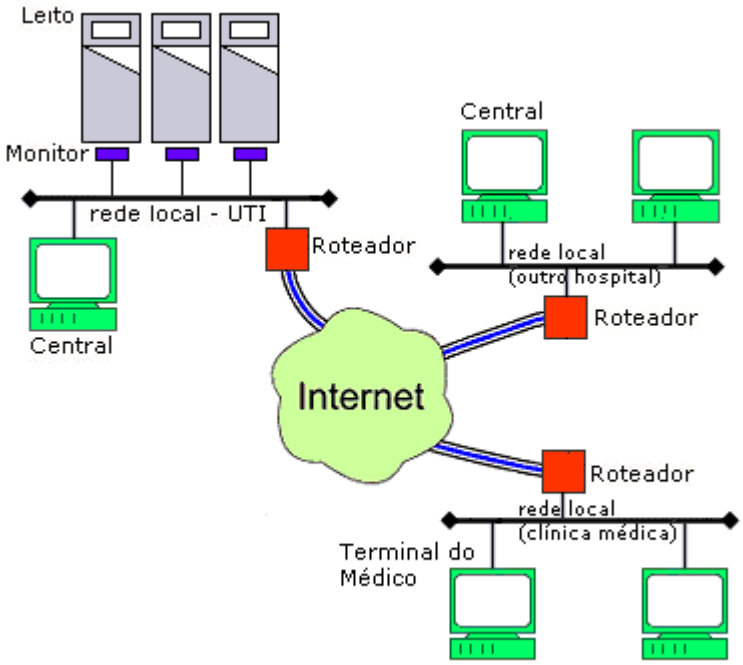

Figura 1.7 Esquema de montagem do monitoramento através da Internet.

Da mesma forma, durante uma telecirurgia o médico monitoraria remotamente e em tempo-real os sinais vitais. Pacientes recuperando-se de cirurgias em suas casas

podem ser monitorados remotamente por médicos situados em hospitais e clínicas. Indo mais longe, no ápice da aplicação do sistema MonitorIP, teríamos o monitoramento remoto de um paciente dentro de uma ambulância a caminho do hospital, através da rede de comunicação celular e da tecnologia *Global System for Mobile Communications* (GSM).

1.3 Componentes do Sistema MonitorIP

Nesta seção serão apresentados os componentes principais do sistema MonitorIP. Esses componentes são os que trabalharão durante o monitoramento, uma vez que já se tenha estabelecido uma conexão.

A Figura 1.8 mostra os principais componentes do núcleo do sistema MonitorIP. As flechas indicam a seqüência do processamento desde a captura dos sinais junto ao paciente até a visualização remota pelo médico. Todos esses componentes serão descritos detalhadamente nos próximos capítulos desta dissertação, mas a seguir, será feita uma descrição geral de cada componente do sistema. Tendo em mente uma visão geral do sistema MonitorIP, ficará mais clara a explicação em detalhes de cada componente.

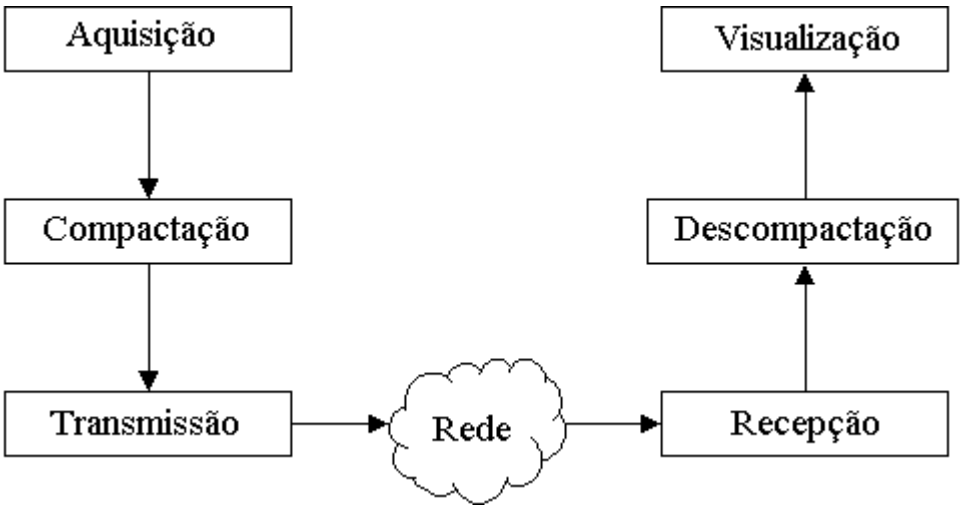

Figura 1.8 Componentes do sistema MonitorIP.

1.3.1 Aquisição e Visualização

Para tornar possível o envio de informações fisiológicas de pacientes através de uma rede de computadores, os sinais bioelétricos têm que ser codificados para uma representação digital (conversão analógica para digital). Esses sinais são amostrados

por eletrodos e convertidos para a forma digital através de um dispositivo especial. Esse processo é chamado de aquisição ou digitalização.

A fim de manter os aspectos de tempo-real da monitorização, é necessário que o receptor comece a receber os sinais assim que possível, depois que a transmissão tenha iniciado. Desta forma, pequenos blocos de informação são enviados em intervalos regulares através da rede e prontos para serem processados pelo receptor.

Quando um bloco é recebido, ele é mostrado na tela do computador do médico como um gráfico de forma de onda. Assim, o médico poderá ver todas as informações fisiológicas do paciente na tela de seu computador, como se ele estivesse visualizando um monitor de multi-parâmetros ao lado do paciente.

Os vários sinais fisiológicos serão digitalizados e enviados separadamente através da rede, ou seja, cada bloco de dados enviado corresponde a um sinal em específico. Com isso, consegue-se aproveitar melhor a característica de cada sinal para aplicar-se técnicas particulares de compressão, e como o médico tem a opção de escolher quais informações ele quer visualizar, isto facilita a implementação da transmissão e permite uma escalabilidade do sistema para novos dados.

Quando os blocos de dados são enviados pela rede, existem pequenas variações no tempo que cada bloco leva para chegar ao destino. De acordo com o congestionamento que ocorrer na rede, essas variações podem ser grandes. A importância dada a essas variações é a seguinte: suponha que sejam apresentados na tela do computador receptor do médico os dados fisiológicos do primeiro bloco de dados assim que o mesmo seja recebido. Por causa do *jitter* (variação do atraso), é possível que o próximo bloco ainda não tenha chegado no instante em que seja terminada a apresentação do primeiro. Para que esse problema não ocorra, técnicas de *buffering* serão utilizadas, de forma que ao final da apresentação das informações de um bloco, o próximo já estará disponível. Porém, esse *buffering* introduzirá um certo atraso, portanto cuidados devem ser tomados para que o atraso total não seja excessivamente grande.

1.3.2 Compactação e Descompactação

A informação digitalizada requer uma certa quantidade de largura de banda disponível para a conexão. Esquemas de compressão podem ser utilizados para reduzir essa largura de banda requerida para o monitoramento remoto.

Existem vários tipos de compressão, desde técnicas gerais utilizadas para

outros tipos de dados, até tipos mais específicos que tentam explorar as características dos sinais fisiológicos, como as apresentadas em [1] [2] [3] [4] [5].

Uma vez que os blocos compactados com os dados fisiológicos cheguem ao destino, eles têm que ser descompactados. O protocolo RTP, que realiza a transferência dos dados, possui um campo em seu cabeçalho (*payload*) identificando qual é o tipo de dado que esse bloco está transportando. Assim, pode-se aplicar o algoritmo exato de descompactação diretamente relacionado ao algoritmo de compactação utilizado. O sinal descompactado deverá ser tão igual quanto possível ao sinal original antes da compactação. Esse campo, identificando o tipo de dado que o protocolo RTP está transportando, traz uma flexibilidade no sentido de permitir uma alteração imediata do algoritmo de compactação utilizado, uma vez que o receptor saberá exatamente qual algoritmo de descompactação deverá utilizar. Desta forma, pode-se implementar mecanismos de controle adaptativo no transmissor, com a finalidade de adequar a taxa de *bits* às variações de carga da rede, utilizando diferentes algoritmos de compactação para cada caso como apresentado em [6] [7] [8] [9] [10] [11].

A compactação é muito importante quando a conexão tem uma largura de banda pequena, como, por exemplo, em acessos discados (*dial-up*). Além disso, o médico pode escolher mais dados a serem monitorados, o que aumenta a largura de banda necessária, tornando a compactação uma opção importante para a redução desse requisito.

1.3.3 Transmissão e Recepção

Finalmente, os blocos têm que ser enviados da origem para o destino através de uma rede de computadores. Algumas informações devem ser adicionadas aos dados, de forma a possibilitar a reconstrução da ordem exata dos blocos pelo receptor, bem como identificar o tipo de dado que ele está transportando. Isto é necessário pois os blocos podem ser perdidos, atrasados ou duplicados durante a transferência. Existem maneiras de se garantir uma certa qualidade da comunicação e realizar uma transferência mais eficiente quando trabalhando com múltiplos destinos, mas elas serão explicadas mais tarde.

1.4 Organização deste Documento

Esta dissertação está organizada em duas partes principais. A primeira parte

são alguns capítulos relacionados à fundamentação teórica do trabalho proposto. E a segunda parte, são alguns capítulos que discutem o desenvolvimento do sistema, os resultados e as conclusões gerais. Abaixo segue um resumo dos capítulos encontrados nesta dissertação.

1.4.1 Fundamentação Teórica

A parte relacionada à fundamentação teórica inicia com o capítulo 2, descrevendo o que é um monitor de multi-parâmetros e quais os dados fisiológicos monitorados, bem como suas características. No capítulo 3, será apresentado o protocolo IP (*Internet Protocol*), uma vez que a proposta do MonitorIP é a transferência de dados através de uma rede IP, destacando-se a importância da descrição de algumas de suas características. No capítulo 4 será mostrado o que vem a ser qualidade de serviço (QoS). As características da transmissão de sinais vitais bem como as restrições temporais para o monitoramento em tempo-real serão apresentados no capítulo 5. O *Real-Time Transport Protocol* (RTP) será descrito no capítulo 6, finalizando a primeira parte desta dissertação.

1.4.2 MonitorIP

O capítulo 7 descreve a arquitetura do software desenvolvido e detalha cada processo, mostrando as particularidades de sua implementação e seus diagramas de classes. Durante o desenvolvimento do MonitorIP vários aspectos foram observados e as decisões de projeto tomadas serão discutidas neste capítulo. O capítulo 8 descreve os resultados obtidos com o sistema MonitorIP. Discussões, conclusões e trabalhos futuros serão descritos no capítulo 9 e por último as referências bibliográficas.

2. MONITOR DE MULTI-PARÂMETROS

O Monitor de multi-parâmetros é um aparelho eletrônico que incorpora diversas funções de medição de sinais vitais do paciente, além de um monitor de gases respiratórios. O campo de aplicações do monitor de multi-parâmetros é bastante amplo, incluindo anestesia e terapia intensiva.

2.1 Descrição Geral

O monitor de sinais vitais pode ser configurado em diversas composições diferentes, com distintas opções de parâmetros monitorados. A Tabela 2.1 relaciona os parâmetros básicos disponíveis em todas as versões de monitores, bem como os parâmetros opcionais, que podem estar presentes conforme a necessidade do médico.

Tabela 2.1 Parâmetros básicos e opcionais de um monitor de multi-parâmetros.

PARÂMETRO	UNIDADE DE MEDIDA
Parâmetros Básicos	
Saturação de Oxigênio (SpO_2)	$\%O_2$
Eletrocardiograma (ECG)	V (Volts)
Freqüência Cardíaca (FC)	bpm (batimentos/min)
Pressão Arterial Não-Invasiva (PANI)	mmHg
Temperatura (T)	ºC
Parâmetros Opcionais	
Gases Respiratórios (O_2, CO_2, N_2O, Agente anestésico)	CO_2 em mmHg ou %
Freqüência Respiratória (FR)	rpm
Pressão Arterial Invasiva (PAI)	mmHg

Alarmes sonoros e visuais, ajustáveis para todos os parâmetros, sinalizam alguma anormalidade para os médicos e enfermeiros.

A seguir será apresentada uma breve descrição dos principais parâmetros fisiológicos monitorados pelos monitores de multi-parâmetros.

2.2 Oxímetro de Pulso (SpO_2)

O oxímetro de pulso realiza a medição da porcentagem de hemoglobina saturada com oxigênio no sangue (SpO_2). Este valor de saturação de oxigênio é medido através de um sensor não invasivo. A monitorização do paciente com o oxímetro de pulso aumenta a segurança em todos os procedimentos que envolvem a administração

de oxigênio ao paciente, indicando de uma forma prática, rápida e precisa as alterações na oxigenação sangüínea.

2.2.1 Princípio de Funcionamento

A saturação de oxigênio no sangue arterial (SpO_2) é definida como sendo a relação entre a hemoglobina oxigenada e a soma da hemoglobina oxigenada com a hemoglobina livre para ligação com oxigênio. A hemoglobina é encontrada no sangue sob diferentes formas, incluindo a hemoglobina oxigenada (Oxihemoglobina, ou O_2Hgb), a hemoglobina reduzida (Desoxihemoglobina, ou Hgb) e outras que não são medidas diretamente pelo oxímetro de pulso, e portanto, não entram no cálculo.

$$\%saturaçãodeOxigênio = \frac{Oxihemoglobina}{Oxihemoglobina + Desoxihemoglobina} \times 100$$

O oxímetro de pulso mede a saturação de oxigênio (em porcentagem) de forma não invasiva, pelos princípios da pletismografia e da espectrofotometria. Neste método, dois comprimentos de onda de luz atravessam uma região rica em vasos sanguíneos, e os sinais resultantes são medidos por um fotodetector. Sendo os comprimentos de onda sensíveis à hemoglobina e a desoxihemoglobina, o cálculo da porcentagem de saturação de oxigênio pode então ser efetuado. Estes sinais eletrônicos também fornecem ao sistema uma indicação de quando as medições de SpO_2 devem ser feitas, permitindo assim a determinação da pulsação do paciente e a compensação dos efeitos do sangue venoso, da pigmentação da pele e de outros tecidos. Um dos comprimentos de onda é utilizado para se obter a pletismografia (curva baseada na medição de mudanças de volume), a qual pode ser visualizada na tela do monitor de multi-parâmetros. O valor da SpO_2 é atualizado na tela a cada pulsação, sendo calculado como a média dos últimos 16 valores obtidos de saturação de oxigênio.

2.3 Eletrocardiograma (ECG)

O monitor cardíaco possui eletrodos posicionados externamente na superfície do corpo do paciente, para a monitorização da atividade elétrica do coração. A monitorização do eletrocardiograma (ECG) é um item fundamental em anestesia e em diversas outras situações clínicas. É feita também a medição da freqüência cardíaca (FC).

2.3.1 Princípio de Funcionamento

O coração é um órgão formado por quatro câmaras, situado entre os dois pulmões, cujos movimentos de contração (sístole) e relaxamento (diástole) impulsionam o sangue pelos vasos sanguíneos. Está dividido em dois compartimentos superiores, denominados átrios, e dois inferiores, denominados ventrículos. Os átrios se comunicam com os ventrículos através de válvulas que deixam o sangue passar, mas impede o seu retorno. As paredes do coração são formadas por um tecido muscular denominado miocárdio [12].

O ECG é o registro da atividade elétrica do coração relacionada à ação do músculo cardíaco. A atividade mecânica da função cardíaca está intimamente relacionada à atividade elétrica, tornando o eletrocardiograma uma ferramenta importante para monitorar o funcionamento do coração.

Um sinal típico de ECG é mostrado na Figura 2.1. O sistema de condução elétrica do coração inicia-se com um impulso no nó sinoatrial, localizado na junção da veia cava superior com o átrio direito. Uma onda de excitação espalha-se pelo átrio produzindo a onda P e causando a sua contração. A excitação é então atrasada no nó atrioventricular, resultando no intervalo P-R. Posteriormente a onda de excitação se espalha através dos ventrículos, causando uma contração e produzindo o complexo QRS. Recuperando-se, a repolarização ventricular produz a onda T [12].

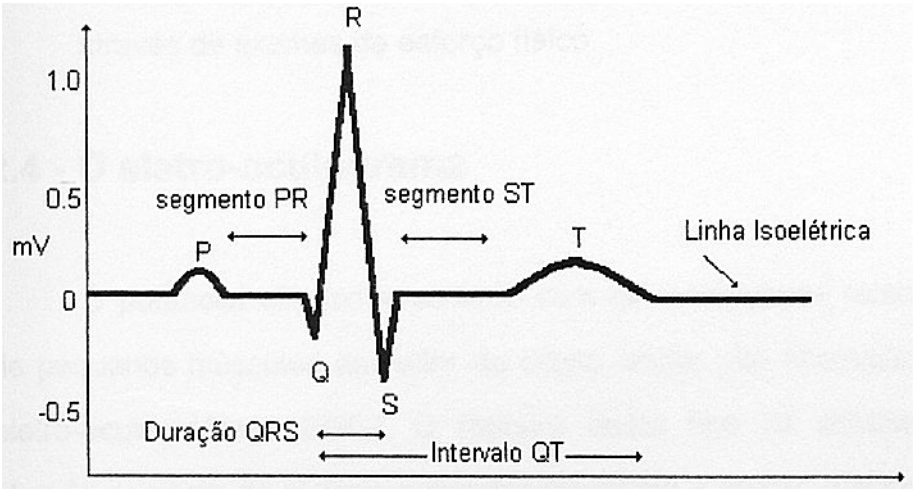

Figura 2.1 Onda típica do complexo QRS [12].

A faixa de amplitude do ECG varia de 50μV a 5mV e em freqüência de 0,1 a 100Hz. Porém, para estudos de eletrocardiografia de alta resolução, torna-se necessário considerar sinais com valores de amplitude mais baixos. Isto se deve ao

fato que manifestações eletrofisiológicas destas condições dão origem a sinais de amplitude menor que 10μV, e que normalmente apresentam superposição de ruído, além de se apresentarem em uma faixa de freqüência mais alta (entre 25 e 250Hz) que as do ECG convencional. Estes potenciais são denominados Potenciais Ventriculares Tardios (PTV) [13][14], e consistem na despolarização tardia do miocárdio. Os PTV's podem ser observados na região terminal do complexo QRS e/ou no começo do segmento ST do ECG, na presença de doenças cardiovasculares [13] .

Entre as diversas aplicações do ECG pode-se citar:

a) *Cardiologia* – diagnosticar doenças, ou arritmias relativas ao mau funcionamento do coração, por exemplo: taquicardia, bradicardia, infartos do miocárdio, entre outras.
b) *Fisiologia do exercício* – analisar a funcionalidade do músculo cardíaco através de exames de esforço físico.

2.4 Freqüência Cardíaca

A freqüência cardíaca (FC) pode ser medida através do oxímetro de pulso, ECG ou através da PANI (Pressão Arterial Não Invasiva), sendo esta escolha feita automaticamente pelo sistema em função do sinal disponível com melhor qualidade. Os monitores de multi-parâmetros possuem indicação audiovisual de pulso e alarmes de baixa e alta freqüência cardíaca.

2.5 Pressão Arterial Não Invasiva (PANI)

Os monitores de multi-parâmetros realizam a medição da pressão não invasiva (PANI) em pacientes de todas as idades, desde neonatais até idosos. É utilizado o método oscilométrico de medição, com o enchimento e o esvaziamento automático de um manguito colocado no braço do paciente.

2.5.1 Princípio de Funcionamento

O método oscilométrico detecta a pulsação arterial para medir a pressão sanguínea, através da variação de pressão no interior do manguito. O manguito posicionado ao redor do braço do paciente é inflado e pressurizado, até interromper o fluxo do sangue nas artérias. Enquanto o manguito é gradualmente esvaziado, a magnitude da oscilação da pressão detectada no interior do manguito aumenta até atingir um pico, e começa a diminuir. A pressão arterial é calculada a partir da

relação entre as variações da pressão no interior do manguito e da oscilação de pressão. A pressão do manguito é medida por um transdutor de pressão com semicondutor, interno ao sistema.

A pressão sistólica é a pressão do manguito quando a oscilação começa a ser perceptível. A pressão diastólica é a pressão do manguito quando a oscilação deixa de ser perceptível. Pressão média é a pressão do manguito quando a oscilação é máxima.

2.6 Temperatura (T)

O monitor de temperatura integrado aos monitores de multi-parâmetros possui geralmente dois canais independentes, para dois sensores do tipo termistor. Termistor é um componente eletrônico cuja impedância é variável com a temperatura, resultando numa relação da temperatura com a corrente ou tensão elétrica aplicada.

2.7 Gases Respiratórios

Os monitores de multi-parâmetros podem vir acompanhados de funções de monitoramento de gases respiratórios, para a medição da concentração de cada gás na mistura inspirada e expirada pelo paciente. Concentrações de oxigênio, gás carbônico, agente anestésico e óxido nitroso (N_2O), na inspiração e na expiração, bem como agentes anestésicos voláteis Halothane, Enflurane, Isoflurane, Sevoflurane e Desflurane, podem ser medidos.

2.7.1 Princípio de Funcionamento

O sistema mede as concentrações dos gases CO_2, N_2O e agente anestésico através do princípio da absorção não dispersiva de infravermelho. A concentração de O_2 é medida através de um sensor galvânico ou paramagnético.

Uma pequena amostra de gás é recolhida do sistema respiratório pelos monitores, através de um tubo fino apropriado. Esta amostra é submetida então aos seguintes subsistemas de medição:

- Analisador da concentração de gás carbônico e gases anestésicos.
- Analisador da concentração de oxigênio identificador do tipo de agente anestésico.

A análise da composição da amostra por absorção não dispersiva de infravermelho realiza-se quando esta passa pelo interior de uma câmara de medição. A tecnologia do infravermelho é largamente utilizada na análise de gases, utilizando

uma luz que não pode ser vista pelo olho humano. A luz infravermelha é gerada por uma fonte, atravessando a mistura de gases encontrada dentro da câmara. Esta mistura absorve diferentes quantidades de energia em diferentes comprimentos de onda, dependendo da sua composição. O feixe de luz atravessa então alternadamente três filtros que permitem somente a passagem do comprimento de onda da luz em que o CO_2, N_2O e agente anestésico absorvem energia, fazendo com que a intensidade da luz seja relacionada com a concentração destes gases. Um detector converte o feixe de luz infravermelho em sinais elétricos, que são processados por um circuito eletrônico para fornecer a leitura das concentrações. Os resultados são apresentados sob a forma gráfica e numérica na tela do monitor de multi-parâmetros.

O sensor de O_2 constitui-se em uma célula galvânica com um sistema de eletrodos que, por intermédio de reações eletroquímicas, fornece ao circuito eletrônico do monitor um sinal de tensão elétrica proporcional à concentração de oxigênio na mistura. Este sinal é processado e convertido em um valor digital de porcentagem volumétrica de oxigênio.

2.8 Freqüência Respiratória (FR)

A freqüência respiratória (FR) é medida através do sinal da curva de capnografia. A faixa de medida é de 0 a 90 respirações por minuto (rpm), a resolução é de 1 rpm, a precisão é de ±2 rpm e a fonte de medição é o CO_2.

2.9 Pressão Arterial Invasiva (PAI)

Os monitores de multi-parâmetros medem a pressão arterial invasiva através de um circuito acoplado a um transdutor de pressão e a um cateter introduzido no sistema circulatório do paciente. A pressão é transmitida desde o local de medição até o transdutor através de um fluído fisiológico que preenche o circuito. Alguns dos locais de medição da pressão invasiva:

- ABP – Pressão Arterial
- LVP – Pressão no Ventrículo Esquerdo
- RVP – Pressão no Ventrículo Direito
- LAP – Pressão no Átrio Esquerdo
- RAP – Pressão no Átrio Direito
- CVP – Pressão Venosa Central
- OP – Pressão de Oclusão ou Wedge

- ICP – Pressão Intracraniana
- PAP – Pressão na Artéria Pulmonar
- AOP – Pressão na Aorta

Geralmente os monitores de multi-parâmetros possuem dois canais de PAI independentes, com apresentação gráfica das curvas de pressão em tempo real e apresentação numérica dos valores mínimos, médio e máximo. São utilizados transdutores de pressão de alta sensibilidade e baixo tempo de resposta.

2.9.1 Princípio de Funcionamento

A medição da pressão arterial invasiva é realizada através de um circuito adequado, acoplado a um transdutor de pressão com baixo tempo de resposta e a um cateter introduzido no sistema vascular do paciente. A pressão é transmitida do local de medição até o transdutor através de um fluido fisiológico que preenche o circuito.

O sistema de lavagem (*flushing*) é um dos pontos fundamentais do circuito de medição. Ele consiste de um soro heparinizado mantido sob pressão constante dentro de uma bolsa, e é conectado ao paciente através de um conjunto de tubos e torneiras (*taps*). A bolsa pressurizadora é preenchida até uma pressão de 300 mmHg, e travada, de forma a manter uma pressão constante. Montado na linha de pressão, existe um dispositivo que, sob pressão constante, assegura um fluxo contínuo de 3 a 5 ml/hora de soro heparinizado através do sistema. Isto evita a formação de coágulos e o retorno do fluxo do cateter inserido no sistema vascular. Este dispositivo de fluxo constante também permite a lavagem manual do sistema, se necessário.

A linha de pressão consiste de pequenos tubos que devem ser curtos e rígidos, para minimizar o efeito de amortecimento dos sinais dinâmicos de pressão. Por este mesmo motivo, não deve haver bolhas de ar no sistema.

3. O *INTERNET PROTOCOL* (IP)

Antes que se possa falar sobre o MonitorIP, será preciso explicar o que significa a sigla IP da palavra MonitorIP. A abreviação corresponde ao Internet Protocol (IP), o qual será descrito neste capítulo. Inicia-se com uma discussão sobre a arquitetura TCP/IP, seguida por uma descrição do protocolo IP. Serão vistas também algumas características das redes IP e descritos os protocolos mais usados que rodam sobre IP. E, termina-se dando algumas razões para se usar o IP para a transmissão dos dados fisiológicos.

3.1 Arquitetura do Software de Rede

Hoje em dia, o software de rede está muito bem estruturado. Esta seção mostra como ele está organizado. Contém também uma discussão sobre o modelo de referência *Open System Interconnection* (OSI), o qual é um excelente exemplo de projeto estruturado e o modelo de referência TCP/IP, onde o protocolo IP tem um papel importante.

3.1.1 Projeto em Camadas

Para facilitar o projeto do software de rede, geralmente se utiliza uma abordagem em camadas. Nesta abordagem, cada camada fornece uma certa funcionalidade, a qual será usada pela camada imediatamente acima. Existem várias vantagens de se utilizar essa abordagem.

Usando-se essa abordagem estruturada, facilita-se o desenvolvimento do projeto do software, bem como a realização de buscas por falhas e uma correção mais eficiente. Seria muito difícil e provavelmente resultaria em uma série de falhas, tentar implementar toda a funcionalidade desejada de uma vez. Além disso, essas falhas seriam muito difíceis de se localizar e corrigir. Dividindo-se o programa em camadas, tem-se apenas a preocupação com a implementação de cada camada em separado.

Outra vantagem é a capacidade de adaptação. Se algumas mudanças são necessárias, como por exemplo, corrigir uma falha ou sofisticar algum algoritmo, somente será realizada a alteração na camada a que diz respeito, se a interface com a camada diretamente acima permanecer a mesma.

A portabilidade está intimamente relacionada à essa abordagem. Se as

camadas foram bem projetadas, somente algumas delas terão de ser alteradas para serem capazes de utilizar o software com outro hardware de rede ou outro sistema operacional.

E finalmente, uma vez que várias camadas provavelmente são implementadas como parte do sistema operacional, o aplicativo do usuário final não terá de implementar todas as camadas. Desta maneira, o tamanho da aplicação será reduzido e simplificará a implementação.

Para tornar possível a comunicação entre dois *hosts* (computadores), eles terão de ser conectados através de algum tipo de meio físico. Todos os dados serão enviados através deste meio, mas somente a camada inferior terá acesso diretamente a eles. Conceitualmente, portanto, duas camadas em máquinas diferentes, mas com o mesmo nível hierárquico, podem se comunicar diretamente. As regras e convenções usadas nesta comunicação são definidas no protocolo usado para cada nível. O conjunto completo de protocolos de todas as camadas é freqüentemente referenciado como pilha de protocolos. A Figura 3.1 ilustra um exemplo de projeto em camadas.

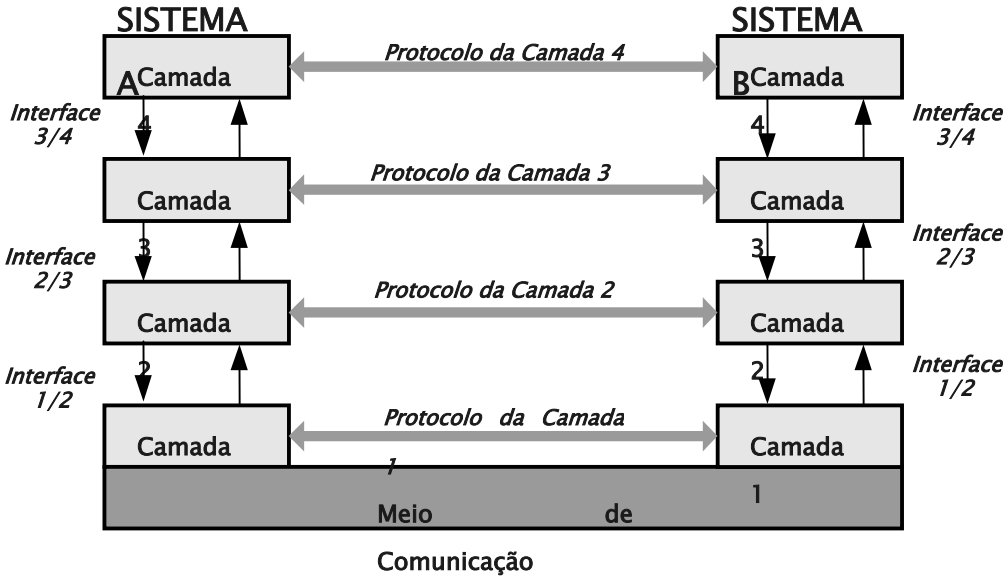

Figura 3.1 Abordagem em camadas da estrutura de projeto de um sistema de comunicação [15].

Quando uma camada quer transmitir algum dado para a camada correspondente à sua em outra máquina, ela usa os serviços da camada imediatamente abaixo para fazer isso. Esta camada então, adiciona aos dados algumas informações de controle, geralmente na forma de um cabeçalho, e utiliza a

camada de baixo para transmiti-los. O processo inteiro se repete, até que os dados são finalmente enviados através do meio de físico.

Assim que os dados chegam ao receptor, a primeira camada processa as informações de controle e passa os dados para a camada imediatamente acima. Repete-se então, o processo para cada camada.

3.2 Modelo de Referência OSI

O modelo de referência *Open System Interconnection* (OSI) é um modelo com sete camadas, as quais foram desenvolvidas pelo *International Standards Organization* (ISO) [20]. Conceitualmente, o modelo é extremamente refinado e é um bom exemplo de um projeto em camadas.

A forma como os dados são transmitidos ao longo do modelo OSI é ilustrada na Figura 3.2.

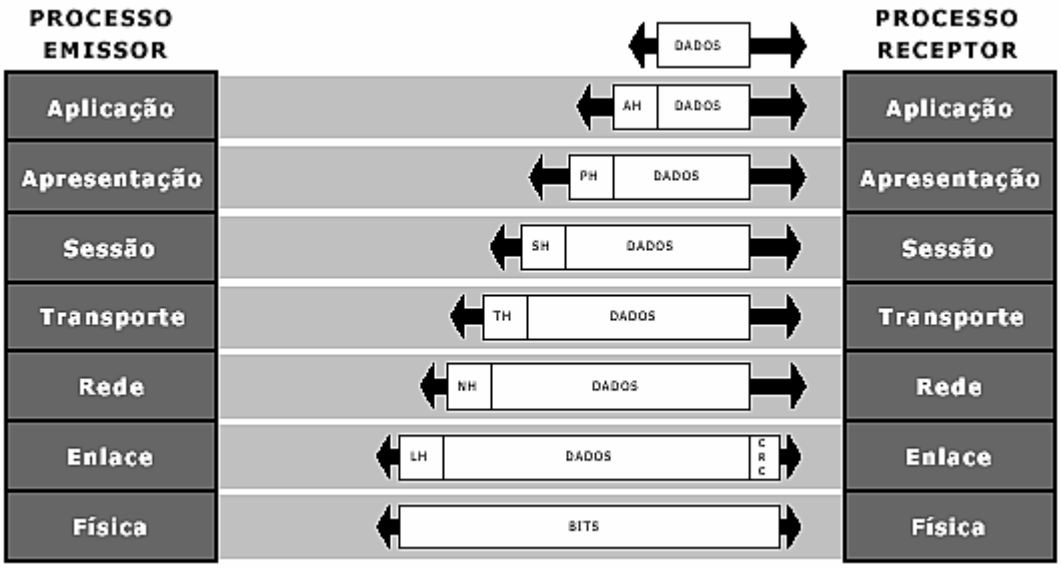

Figura 3.2 Ilustração da comunicação no modelo OSI [15].

Como se pode ver, o processo emissor vai enviar uma certa quantidade de dados ao processo receptor. Ele envia, então, os dados à camada de Aplicação que introduz a estes um cabeçalho de aplicação (AH – *application header*), e envia a mensagem resultante à camada de Apresentação. Esta camada, por sua vez, introduz a mensagem recebida um cabeçalho de apresentação (PH – *presentation header*), enviando a mensagem em seguida à camada inferior. É importante ressaltar aqui que esta camada não toma conhecimento da existência e significado do cabeçalho de aplicação, considerando este como parte dos dados compondo a mensagem. Este

processo de transferência de camada a camada vai se repetindo até o nível físico, quando os dados serão, enfim, transmitidos ao sistema destino. Neste sistema, os diversos cabeçalhos introduzidos nas camadas de rede do sistema fonte vão sendo interpretados e eliminados nas camadas correspondentes até que os dados cheguem ao processo receptor. O conceito fundamental da transferência de dados é que cada camada foi projetada como se ela fosse realmente horizontal, quando na verdade a transmissão se dá de modo vertical [15]. Vamos descrever, a seguir, as principais funções realizadas por cada uma das camadas definidas no modelo OSI.

3.2.1 As Camadas do Modelo de Referência OSI

A **Camada Física** é responsável pela transferência de *bits* num circuito de comunicação. De modo geral, a sua função é garantir que cada *bit* enviado de um lado será recebido do outro lado sem alteração de valor; ou seja, se o *bit* enviado está em 1, ele será recebido em 1 e não em 0 [15]. Para isto, as questões a serem resolvidas neste nível são do tipo:

- os modos de representação dos *bits* 0 e 1 de maneira a evitar ambigüidades ou confusões (valor da tensão em volts para a representação dos valores 0 e 1 dos *bits*, duração de cada sinal representando um *bit*, a codificação dos sinais, etc.);
- os tipos de conectores a serem utilizados nas ligações (número de pinos utilizados, as funções associadas a cada pino, etc.);
- a maneira como as conexões são estabelecidas para a iniciação de um diálogo e como é feita a desconexão ao final deste;
- modo de transmissão adotado (unidirecional, bidirecional, etc.);
- modo de conexão adotado (ponto-a-ponto, multiponto, etc.);
- modo de tratamento dos erros (detecção, tratamento, etc.).

A concepção desta camada deve se relacionar à definição das interfaces elétricas e mecânicas, seus modos de funcionamento, o suporte de comunicação adotado, etc.

A **Camada de Enlace de Dados** tem por função principal a transformação do meio de comunicação "bruto" em uma linha livre de erros de transmissão para a camada de Rede. Ela efetua esta função através do fracionamento das mensagens recebidas do emissor em unidades de dados denominadas **quadros**, que correspondem a algumas centenas de *bytes*. Estes quadros são transmitidos seqüencialmente e vão

gerar quadros de reconhecimento enviados pelo receptor. Nesta camada, as unidades de dados são enriquecidas com um conjunto de *bits* adicional (no início e fim de cada quadro) de modo a permitir o reconhecimento destes [15].

Um problema típico deste nível é o da ocorrência de uma perturbação sobre a linha de transmissão que provoque a destruição (perda) do quadro enviado. Neste caso, o quadro deve ser retransmitido para garantir a integridade da informação transferida. Por outro lado, deve-se também evitar múltiplas retransmissões de um mesmo quadro, o que pode provocar a sua duplicação, por exemplo, se o quadro de reconhecimento é perdido. Uma outra função desta camada é evitar uma alta taxa de envio de dados da parte do emissor no caso do sistema receptor não ter capacidade de absorver a informação à mesma taxa. Este mecanismo deve permitir informar ao emissor a necessidade de armazenamento dos dados a transmitir (controle de fluxo).

A **Camada de Rede** é responsável da gestão de sub-redes; ela define a forma como os pacotes de dados serão encaminhados do emissor ao receptor. Os caminhos a serem utilizados podem ser definidos em função de tabelas estáticas ou determinados dinamicamente no momento de cada diálogo em função das condições de tráfego da rede [15]. Esta camada deve ainda efetuar a gestão dos problemas de congestionamento provocados pela presença de uma quantidade excessiva de pacotes de dados na rede. Ela deve, finalmente, resolver todos os problemas relacionados à interconexão de redes heterogêneas, particularmente:

- incompatibilidades no endereçamento;
- incoerências em relação aos tamanhos das mensagens;
- etc.

A **Camada de Transporte** recebe os dados enviados da camada de Sessão, decompõe-os, se for o caso, em unidades de dados menores e garante que todas as partes da mensagem serão transmitidas corretamente à outra extremidade. Esta função deve ser suprida de maneira eficiente, inclusive, sem que a camada de Sessão tome conhecimento de possíveis alterações na tecnologia da parte física da rede [15]. Esta camada cria, normalmente, uma conexão de rede para cada conexão de transporte requerida pela camada de Sessão, embora, caso as necessidades de velocidade de transmissão justifiquem, ela possa estabelecer diversas conexões de rede para uma mesma conexão de transporte. Por outro lado, se o custo da manutenção de uma conexão de rede é considerado elevado, esta camada pode efetuar a função inversa, ou seja, a multiplexação de várias conexões de transporte sobre uma

mesma conexão de rede, esta tarefa sendo feita de modo transparente para a camada de Sessão. Ela deve determinar, também, o tipo de serviço oferecido à camada de Sessão e, por conseqüência, aos usuários da rede. Uma conexão de transporte típica é aquela de um canal ponto-a-ponto, livre de erros de transmissão, transmitindo as mensagens na mesma ordem em que elas foram enviadas. Por outro lado, outras classes de serviços podem fornecer uma conexão capaz de enviar as mensagens de modo isolado, mas sem a garantia de uma ordem correta na transmissão. O tipo do serviço a ser fornecido é definido no momento do estabelecimento da conexão.

Uma característica desta camada é que ela implementa um verdadeiro diálogo *fim-a-fim*, ou seja, o programa executado no sistema fonte dialoga com o programa executando na máquina destino através dos cabeçalhos e informações de controle contidas nas mensagens deste nível. Já nas camadas mais baixas, os protocolos operam entre máquinas vizinhas e não entre os sistemas fonte e destino [15].

Dado que esta camada é responsável pelo estabelecimento e término das conexões de rede, ela deve definir um mecanismo de endereçamento que permita a um sistema indicar com qual sistema ele deseja dialogar. Finalmente, ela deve implementar um mecanismo de controle de fluxo fim-a-fim para evitar que o sistema fonte envie mensagens numa taxa superior àquela com a qual o sistema destino pode consumi-las.

A **Camada de Sessão** é responsável pelos estabelecimentos de sessões de diálogo para os usuários da rede. Uma sessão deve permitir o transporte de dados, da mesma forma que os serviços oferecidos pela camada de Transporte, mas ela oferece serviços mais sofisticados de comunicação que podem ser úteis a determinadas aplicações [15]. Um exemplo disto é a possibilidade de envio, através de uma sessão, de um arquivo de dados (ou programa) de um sistema a outro. Outro serviço da camada de Sessão é efetuar a gestão do diálogo, ou seja, definir, por exemplo, se o diálogo vai ser efetuado em modo uni ou bidirecional.

Um serviço também importante é aquele da sincronização do diálogo. Por exemplo, se um arquivo deve ser transferido através de uma sessão e esta deve durar duas horas. Se, por uma razão qualquer, o tempo médio entre duas panes for de uma hora, após uma primeira interrupção por pane a transferência deverá reiniciar, podendo ocasionar em erros de transmissão. Uma forma de evitar isto é a inserção de pontos de teste junto aos dados fazendo com que, após uma interrupção de transferência, os dados sejam retomados apenas a partir do último ponto de teste.

A **Camada de Apresentação** utiliza algumas funções freqüentemente necessárias de modo a poupar o usuário deste trabalho. Esta camada assume particularmente as funções associadas à sintaxe e à semântica dos dados transmitidos [15]. Um exemplo típico das funções efetuadas por esta camada é a codificação da informação num padrão bem definido (ASCII, EBCDIC, etc.).

Esta camada pode ainda suprir outras funções associadas à compreensão dos dados, utilizando-se do conhecimento do significado da informação para reduzir a quantidade de informação enviada, inclusive para implementar funções de confidencialidade e de autenticação.

A **Camada de Aplicação** implementa um conjunto de protocolos bastante diversificado e orientado a aplicações bem definidas. Um exemplo disto é o protocolo de terminal virtual, que permite gerar a utilização de um determinado programa (por exemplo, um editor de textos) de forma independente do tipo de terminal conectado à rede. Outro serviço importante é o de transferência de arquivos, que permite adaptar o tipo do arquivo transferido à forma implementada pelo sistema de arquivamento do sistema considerado. Na parte dedicada a esta camada veremos, além destas, outras classes de serviços implementados a este nível [15].

3.3 Modelo de Referência TCP/IP

O termo "protocolo TCP/IP" é utilizado como designação comum para uma família de protocolos de comunicação de dados, sendo que o *Transmission Control Protocol* (TCP) e o *Internet Protocol* (IP) são apenas dois deles. Esta família de protocolos teve origem na rede Arpanet. A Internet cresceu muito além do que se podia imaginar. A Internet é hoje uma coleção de redes acadêmicas, militares e comerciais espalhadas pelo mundo, interconectadas através do protocolo TCP/IP. Entre elas, pode-se citar a própria Rede Nacional de Pacotes (RNP) no Brasil [15].

Uma vez que toda a rede conectada à Internet deve "falar" o protocolo TCP/IP, é natural que o interesse comercial por este protocolo tenha crescido muito, a ponto de hoje estar disponível em quase todas as plataformas. Além disso, é comum encontrarmos TCP/IP sendo utilizado em redes locais que não estão conectadas à Internet. O sucesso e a popularidade do protocolo TCP/IP não se deve apenas à imposição das agências militares americanas, mas também ao fato der ter sido o primeiro protocolo a atingir a importante meta da comunicação de dados com abrangência mundial. Isto somente foi possível graças a algumas de suas

características, a saber [15]:

- TCP/IP é um protocolo aberto, público e completamente independente de equipamentos e de sistemas operacionais;
- TCP/IP não define protocolos para o nível físico, possibilitando sua implementação sobre uma grande variedade de protocolos já existentes, tais como: *Ethernet*, *Token Ring* e X.25;
- O esquema de endereçamento do TCP/IP permite designar univocamente qualquer máquina, mesmo em redes globais como a Internet;
- TCP/IP inclui protocolos do nível de aplicação que atendem muito bem à demanda de serviços imposta pelos usuários.

Uma vez que a padronização foi essencial para a definição do TCP/IP como o protocolo mais utilizado no mundo, é importante que se conheça como ele foi, e continua sendo, padronizado. Originalmente, os protocolos básicos do TCP/IP foram padronizados através de *Military Standards* (MIL STD) e de *Internet Engineering Notes* (IEN). Atualmente, a maior parte da padronização do TCP/IP é feita através de *Requests For Comments* (RFC), que, além da especificação formal dos protocolos, inclui informações importantes sobre seu funcionamento e uso [15].

3.3.1 As Camadas do TCP/IP

A descrição da arquitetura do protocolo TCP/IP em camadas como as do modelo de referência OSI é uma tarefa difícil e, certamente, controversa. As camadas OSI foram definidas por pesquisadores ao longo de anos, sempre com o compromisso acadêmico de ser um modelo de referência, enquanto que o protocolo TCP/IP não teve qualquer compromisso que não a funcionalidade. Assim sendo, tentar estabelecer uma relação precisa entre as camadas OSI e TCP/IP é algo praticamente impossível [15].

O modelo de referência TCP/IP, é composto de quatro camadas: Acesso à Rede, Internet, Transporte e Aplicação. Este modelo é apresentado na Figura 3.3, em comparação ao modelo de referência OSI [15].

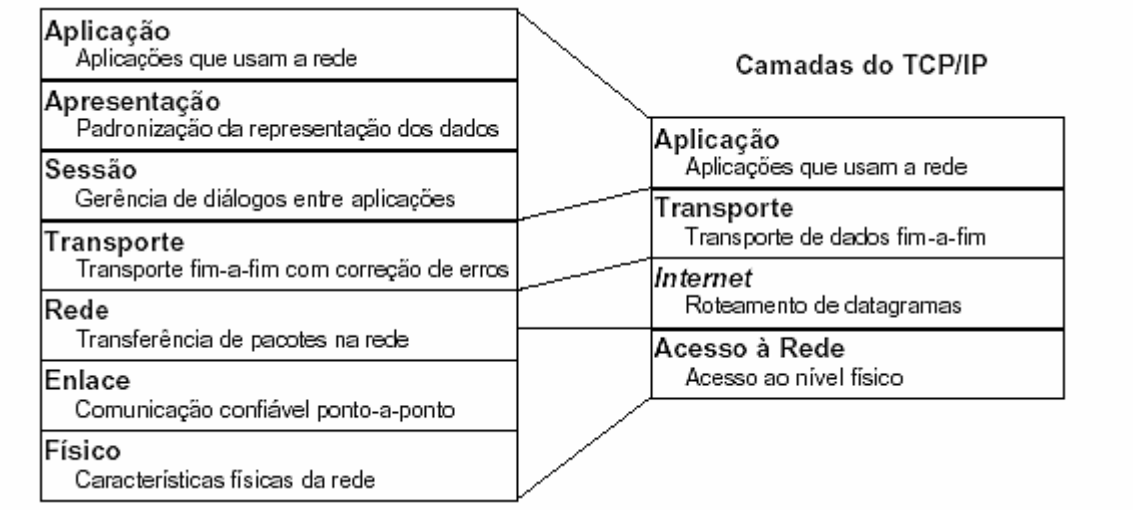

Figura 3.3 Correspondência entre as camadas da arquitetura TCP/IP e as do OSI [15].

Da mesma forma que no modelo de referência OSI, os dados descem a pilha de protocolos para chegar à rede e sobem para chegar às aplicações. Cada camada da pilha de protocolos adiciona um cabeçalho com informações de controle e trata o que recebe da camada superior como dados. Esta adição de informações de controle em cada nível é denominada encapsulamento (Figura 3.4). O processo reverso acontece quando uma camada passa dados às superiores, ou seja, o cabeçalho é removido e o restante do pacote é passado para cima como dados.

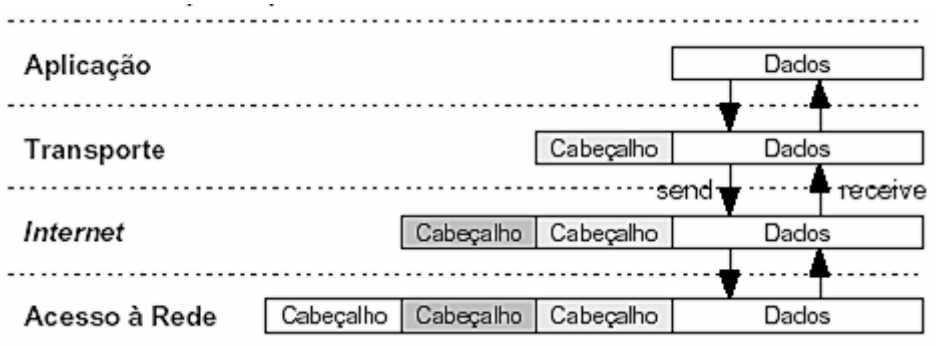

Figura 3.4 Encapsulamento de dados na pilha TCP/IP [15].

Cada camada da pilha possui estruturas de dados próprias e independentes. Assim sendo, cada protocolo faz referência aos dados de forma específica. Por exemplo, aplicações que fazem uso do protocolo TCP referem-se aos dados como *streams*, ao passo que aplicações que fazem uso do protocolo *User Datagram Protocol* (UDP) referem-se aos dados como mensagens. O protocolo TCP, por sua vez, refere-se aos dados como segmentos, enquanto que o UDP se refere aos dados como pacotes. O protocolo IP sempre refere-se aos dados como datagramas, enquanto que os dados

passados à camada de acesso à rede são referidos como *frames* ou quadros. A Figura 3.5 ilustra esta terminologia.

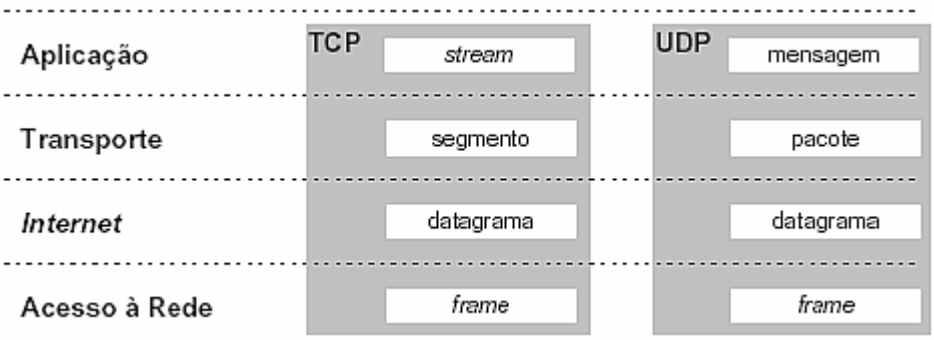

Figura 3.5 Comparação entre terminologia usadas pelo TCP e UDP [15].

A **Camada de Acesso à Rede** é a mais baixa na hierarquia de protocolos TCP/IP. Os protocolos nesta camada provêem meios para que os dados sejam transmitidos a outros computadores na mesma rede física. Esta camada pode abranger as três primeiras camadas do modelo de referência OSI: física, de enlace e de rede. Entretanto, a camada de acesso à rede do TCP/IP não define propriamente os protocolos para estes três níveis, mas sim como utilizar os protocolos já existentes para suportar a transmissão de um datagrama IP [15]. À medida que novas tecnologias de rede vão surgindo, novos protocolos são acrescidos à camada de acesso à rede. As principais funções da camada de acesso à rede são: o encapsulamento de datagramas IP em *frames* para transmissão e a tradução de endereços IP em endereços físicos de rede. Estas duas funções apresentam implementações específicas para cada tipo de rede.

A **Camada Internet** fica exatamente sobre a camada de acesso à rede. O *Internet Protocol* (IP), é o coração desta camada. Ele provê um serviço básico de datagrama sobre o qual as redes TCP/IP são implementadas. Ela tem o trabalho de levar os datagramas da origem para o destino, através de tipos diferentes de redes se necessário. Todos os protocolos das camadas superiores a esta fazem uso do protocolo IP [15]. As principais funções do protocolo IP são:

- definir o datagrama IP, que é a unidade básica de transmissão de dados da arquitetura TCP/IP;
- definir o esquema de endereçamento IP;
- passar dados da camada de acesso à rede à camada de transporte;
- rotear datagramas IP;
- fragmentar e remontar datagramas IP.

O IP é um protocolo não orientado à conexão, ou seja, não existe negociação prévia de uma conexão para a transmissão de dados. Isto não impede a existência de protocolos orientados à conexão nas camadas superiores, mas eles deverão negociar o estabelecimento de conexões por si próprios. Além de ser não orientado à conexão, o protocolo IP também não é confiável, uma vez que não suporta mecanismos de detecção e recuperação de erros. Em outras palavras, o protocolo IP não verifica se um datagrama foi recebido corretamente, deixando esta responsabilidade para os protocolos das camadas superiores. O serviço, o qual está camada oferece, é freqüentemente chamado de serviço de melhor esforço (*best-effort*).

A **Camada de Transporte** fim-a-fim está localizada exatamente sobre a camada Internet na hierarquia TCP/IP. Os principais protocolos desta camada são: *Transmission Control Protocol* (TCP) e *User Datagram Protocol* (UDP). O TCP é um protocolo orientado à conexão com detecção e correção de erros fim-a-fim. UDP é um protocolo não orientado à conexão e não confiável, sendo portanto muito leve. Ambos os protocolos passam dados entre as camadas de aplicação e Internet. Cada aplicação é livre para escolher o protocolo que melhor se adapta à sua natureza [15].

Para possibilitar que múltiplas aplicações utilizem as facilidades da rede simultaneamente em um mesmo computador, alguns mecanismos extras são necessários. A camada Internet possui um mecanismo de endereçamento para identificar diferentes computadores (*hosts*), mas falta ainda uma maneira de diferenciar quais processos estão usando a rede. Isto é feito através da camada de transporte pelo uso de um número de porta (*port*), como ilustra a Figura 3.6.

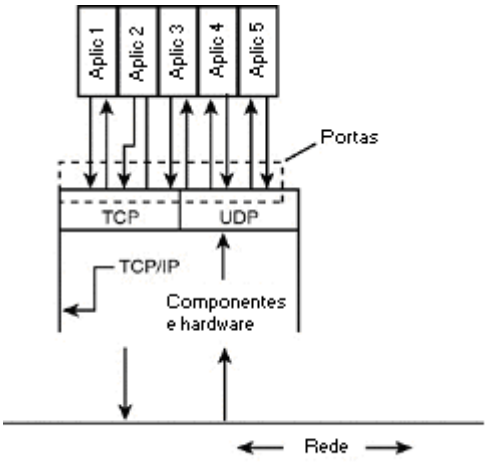

Figura 3.6 Aplicações acessando a rede através de endereços de porta.

O *Transmission Control Protocol* (TCP) é um protocolo orientado à conexão e

confiável. A transmissão de dados através de uma conexão, ou *stream*, se dá através de segmentos. Ele é um protocolo muito importante, uma vez que ele torna possível uma comunicação confiável e por isso, o modelo de referência TCP/IP leva o seu nome. Da mesma forma que o UDP, o TCP carrega informações sobre as aplicações, como as portas de origem e destino.

O *User Datagram Protocol* (UDP) provê meios para que aplicações tenham acesso direto ao serviço de datagrama IP. Aplicações que usam este protocolo inserem pouco *overhead* na rede. Como o próprio IP, o protocolo UDP é não orientado à conexão e não confiável. Note que a expressão não confiável implica apenas na inexistência de mecanismos de confirmação do correto recebimento do datagrama. O protocolo UDP é utilizado principalmente por aplicações que transmitem dados em pequenas quantidades, de tal forma que o *overhead* de uma conexão é maior do que o da retransmissão dos dados em caso de erro. Ele é apenas uma extensão para o protocolo IP, contendo informações do número da porta utilizado na multiplexação e um campo opcional para *checksum* dos dados.

A **Camada de Aplicação** fica no topo da pilha TCP/IP e inclui todos os processos que utilizam serviços das camadas inferiores para transmitir dados através da rede [15]. Alguns protocolos desta camada são citados abaixo, e a Figura 3.7 ilustra dois exemplos de pilha TCP/IP.

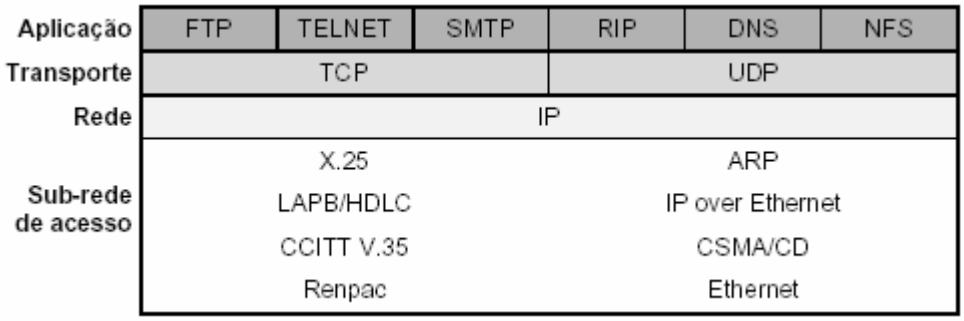

Figura 3.7 Exemplos de protocolos da pilha TCP/IP [15].

- *Telnet*: serviço de terminal virtual;
- *File Transfer Protocol* (FTP): serviço de transferência de arquivos;
- *Simple Mail Transfer Protocol* (SMTP): serviço de correio eletrônico;
- *Domain Name Service* (DNS): serviço de tradução de nomes de *hosts* em endereços IP;
- *Routing Information Protocol* (RIP): suporta a troca de informações de roteamento entre *gateways*;

- *Network File System* (NFS): sistema de arquivos remotamente acessíveis.

3.4 Como o IP Trabalha

Agora o protocolo IP será discutido em mais detalhes, e como ele torna possível a comunicação entre dois *hosts*. Primeiro, será descrito o cabeçalho do datagrama IP, em seguida, o mecanismo de endereçamento. Uma análise de como os datagramas são roteados da origem ao destino também será apresentada. Finalizando, será discutida a técnica de *multicasting*, a qual permite uma economia de largura de banda quando os mesmos dados são enviados para múltiplos destinos.

3.4.1 Formato do Cabeçalho IP

Todo datagrama enviado pela camada Internet é formado por um cabeçalho IP mais os dados. O formato do cabeçalho IP é mostrado na Figura 3.8.

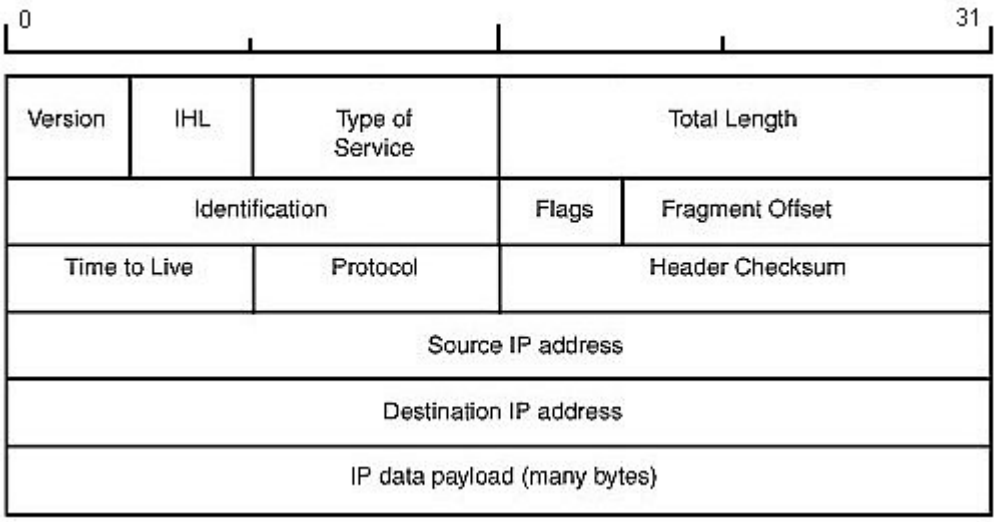

Figura 3.8 Estrutura do cabeçalho IP.

O *bit* mais significativo é o *bit* mais à esquerda, numerado por 0. O *bit* menos significativo é o *bit* mais à direita, numerado por 31. A transmissão é feita usando a ordem de *byte* de rede, que está no formato *big endian*. Isto significa que a cada palavra de 32-*bit* enviada, o *byte* mais significativo é enviado primeiro e o *byte* menos significativo é enviado por último.

O campo ***Version*** deve conter o valor quatro para a versão atual do protocolo IP. Este campo tem a finalidade de permitir que diferentes versões coexistam, tornando assim mais fácil a utilização de novas versões.

O campo **IHL** contém o *Internet Header Length*, que especifica o comprimento do cabeçalho em palavras de 32-*bit*. Sendo este campo de 4-*bit*, o comprimento máximo para o cabeçalho será de 64 *bytes*. Como a parte obrigatória do cabeçalho consiste de cinco palavras de 32-*bit*, o menor valor para este campo será cinco. A especificação em palavras de 32-*bit*, obriga o cabeçalho IP a ter um comprimento múltiplo de 32 *bits*, assim se o campo de opções (*IP Data Payload*) for utilizado, é possível que *bits* sem significados (*padding bits*) sejam requeridos.

O próximo campo é o ***Type of Service*** (ToS). Este campo foi idealizado para fornecer um mecanismo de qualidade de serviço (QoS), mas na prática ele é raramente utilizado. Porém, como os sinais fisiológicos a serem transmitidos têm aspectos de tempo-real, o mesmo receberá uma atenção especial, com a tentativa de se manter um baixo atraso na comunicação fim a fim. O campo ToS é composto por 8 *bits*, os quais têm significados especiais conforme mostrado na Figura 3.9.

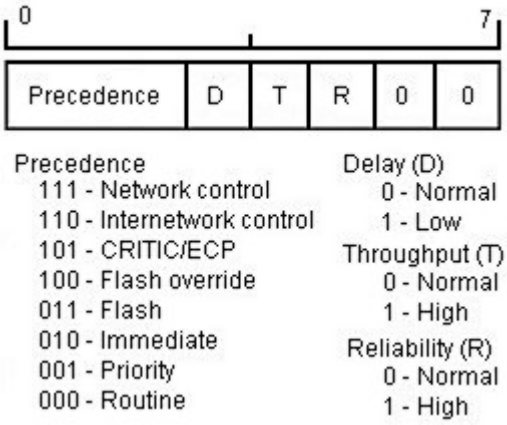

Figura 3.9 Campo *Type of Service* (ToS).

O campo ***precedence*** formado pelos primeiros três *bits* do campo ToS, especifica a prioridade do datagrama. Um valor igual a zero indica uma prioridade normal e um valor igual a sete indica a mais alta prioridade. Os próximos três *bits* são respectivamente *Delay*, *Throughput* e *Reliability*. Somente um destes *bits* poderá ter o valor igual a 1 ao mesmo tempo. Os próximos dois *bits* do campo ToS não são utilizados e devem ter o valor igual a zero.

O tamanho do datagrama IP é indicado pelo campo ***Total Length***. Ele é um campo de 16-*bit*, dessa forma o máximo comprimento de um datagrama será 65535 *bytes*. Muitas redes não podem tratar esse tamanho, por isso os valores utilizados são bem menores, evitando-se assim a fragmentação, explicada mais abaixo.

Durante a transmissão do datagrama, é possível que ele passe por tipos

diferentes de redes. Cada rede tem seu próprio MTU (*Maximum Transfer Unit*), que especifica o tamanho máximo do frame que ela pode tratar, incluindo o cabeçalho da camada de acesso à rede. Isto significa que existe sempre a possibilidade de que o datagrama que passe por diferentes redes possa ser transmitido por uma e não possa ser transmitido por outra rede com menor MTU. Dessa forma, o datagrama terá de ser fragmentado para que cada porção possa ser enviado através dessa rede com menor MTU.

O campo ***Identification*** é um campo de ajuda na reconstrução de datagramas fragmentados. Cada fragmento de um mesmo datagrama terá o mesmo valor para este campo. No envio de um datagrama IP, o *host* incrementa este campo para cada datagrama enviado. Assim, cada fragmento de um mesmo datagrama terá o mesmo valor, mas fragmentos de datagramas diferentes terão valores diferentes para esse campo.

O próximo campo de comprimento de três *bits* é o ***Flag***. O primeiro *bit* desse campo é reservado e deverá ser zero. O próximo *bit* identificado por DF (*Don't Fragment*) determina se o datagrama pode ser fragmentado ou não. Se ele for igual a 1 e o datagrama não puder ser transmitido pela rede, por ser superior ao seu MTU, um erro será enviado para o transmissor. O último *bit* desse campo, MF (*More Fragments*), informa se o datagrama é o último fragmento ou não. Todos os fragmentos do datagrama original, menos o último, devem ter o *bit* MF com o valor igual a 1.

A camada Internet utiliza o próximo campo ***Fragment Offset*** para poder remontar os datagramas fragmentados. Este valor, de 13-*bit*, especifica a posição do fragmento dentro do datagrama original. O valor desse deslocamento é dado em múltiplos de palavras de 64-*bit*.

O campo ***Time to Live*** (TTL) é usado para limitar o tempo de vida de um datagrama. Na teoria o valor especifica o número de segundos que o datagrama é permitido existir. Cada roteador, ao tratar um datagrama, deve decrementar esse campo de pelo menos um segundo. Se o datagrama permanece muito tempo na fila do roteador, o valor do campo TTL deve ser decrementado do número de segundos que o datagrama gastou na fila. Quando esse valor chegar a zero, o datagrama deverá ser descartado. Na prática, o valor é apenas decrementado a cada roteador, dessa forma esse campo funciona como uma indicação do número de roteadores que o datagrama pode passar.

O campo ***Protocol*** é usado para especificar qual protocolo pertencem os dados no datagrama. O valor pode indicar um protocolo da camada de transporte, bem como um protocolo de controle da camada Internet.

O campo ***Header Checksum*** é utilizado para checar a validade do datagrama. Observe que essa verificação (*checksum*) opera apenas para o cabeçalho, por isso os protocolos de maior nível é que deverão fornecer seus próprios esquemas de verificação, se eles querem garantir que seus dados sejam validados.

Finalmente, terminando os campos obrigatórios do cabeçalho IP, temos os ***Source IP Address*** e ***Destination IP Address***. Eles indicam o endereço IP de origem e de destino da comunicação. Esses endereços devem ser incluídos em cada datagrama, uma vez que a camada de Internet opera sem conexão. Cada datagrama é enviado separadamente e, portanto, cada datagrama deve conter não somente seu destino, mas também sua origem, para o caso de um erro ter de ser reportado. O formato do endereço IP será discutido mais tarde.

O campo ***Options*** (***IP Data Payload***) pode ser usado para definir uma rota por onde os datagramas devem passar, através da opção Source Routing, para registrar todo o caminho por onde o datagrama passou, entre outras opções.

3.4.2 Endereçamento

Cada *host* que utiliza o IP para uma interligação em rede, como a internet, deve ter um endereço IP único. O endereço IP é um valor de 32-*bit* e o espaço de endereçamento é dividido em cinco classes, nomeadas da classe A até a classe E. O modo como essas classes são representadas é mostrado na Figura 3.10.

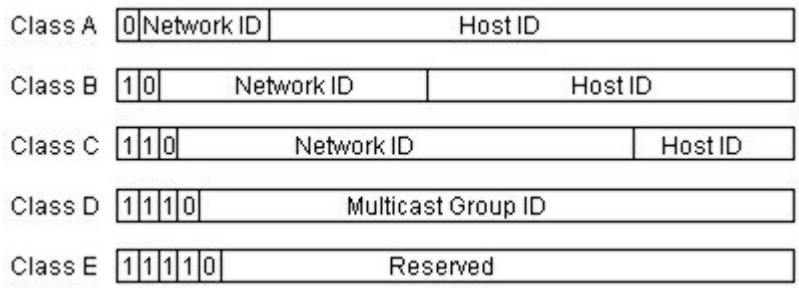

Figura 3.10 Classes de endereços IP.

Geralmente o endereço IP é escrito na forma de notação decimal com ponto. Os endereços IP são escritos com quatro números inteiros decimais separados por pontos decimais, no qual cada número inteiro fornece o valor de um octeto de endereço IP. Assim, o endereço IP de 32-*bit*, 10000000 00001010 00000010 00011110 é

representado por 128.10.2.30.

As três primeiras classes contêm os endereços, os quais se pode associar a um *host*. Nem todas as possibilidades são permitidas, existem alguns endereços reservados. O primeiro endereço, com o campo HostID com valor igual a zero, não especifica um *host*, mas sim uma rede onde os *hosts* com o NetworkID especificado são localizados.

Se o HostID é o maior valor possível dentro de sua classe (todos os *bits* em 1 no formato binário), o endereço é um endereço de difusão (*broadcast*) para uma certa rede. Isto significa que se for enviado datagramas IP para esse endereço, eles serão entregues a todos os *hosts* daquela rede. Como exemplo de endereços pertencente a uma mesma rede tem-se: o endereço de rede 192.168.0.0 (classe C), o endereço de *host* 192.168.0.1 e o endereço de difusão 192.168.0.255.

Quando o NetworkID de um endereço é zero, ele especifica a rede local. Esse tipo de endereço somente é usado em procedimentos de inicialização, quando o NetworkID local ainda não é conhecido.

Outros endereços reservados são 0.0.0.0 e 255.255.255.255. O primeiro identifica o *host* local em uma rede local. Ele também é usado em procedimentos de inicialização. O segundo é o endereço chamado de endereço de difusão limitado. Ele identifica um endereço de difusão em uma rede local.

Das duas classes remanescentes, somente a classe D está atualmente sendo utilizada. A classe E foi especificada para uso futuro. A classe D especifica um endereço de *multicast*. *Multicasting* permite que dados sejam enviados para um grupo de *hosts*. Isto significa que quando se envia um datagrama IP para essa classe de endereço IP, o datagrama será enviado para todos os *hosts* correspondentes ao grupo *multicast*. *Multicasting* será explicado mais tarde, em maiores detalhes.

3.4.3 Roteamento

A camada Internet utiliza a camada de Acesso à rede para transmitir seus dados. A camada de Acesso à rede, porém, pode somente fornecer os dados aos *hosts* que estão conectados para o mesmo meio físico. Então, para que se seja capaz de enviar dados através de várias redes, roteadores são utilizados. Esses dispositivos conectam várias redes e garantem que o datagrama IP que ele está recebendo será repassado para a rede apropriada. Em seguida será explicado o processo básico do mecanismo de roteamento.

Quando a camada Internet, de um *host* que está enviando, tem que transmitir um datagrama para um certo destino, ela primeiro examina o endereço IP de destino. Isto é necessário porque a camada de Internet tem que informar à camada de Acesso à rede para qual máquina os dados tem que ser enviados.

Se o endereço IP de destino pertence à mesma rede, a máquina que receberá o datagrama será simplesmente o destino da transmissão. Se o endereço não identifica um *host* pertencente à rede local, a camada Internet examina sua tabela de roteamento. Os valores armazenados nessa tabela de roteamento podem ser vistos com pares de um endereço de destino e um endereço de roteador. O endereço de destino pode ser um endereço de um *host* ou um endereço de uma rede.

A camada Internet, então, inicia a procura de um roteador, ao qual ela possa enviar os datagramas. Para fazer isto, ela compara o endereço de destino do datagrama com os endereços de destino da tabela de roteamento. Se o endereço de destino não pode ser encontrado, ela então parte pela procura da rede de destino do datagrama nessa tabela. Se mesmo assim, nem o endereço de *host* e nem o endereço de rede de destino do datagrama puderam ser encontrados, ela então utilizará uma opção padrão, que poderá ser um roteador padrão (*gateway*) para aquela rede local. Se um endereço de destino foi encontrado na tabela de roteamento, então a camada de Internet pega o endereço de roteador correspondente e informa à camada de Acesso à rede para enviar o datagrama para o roteador encontrado.

Por exemplo, suponha-se que um *host* com endereço IP 192.168.0.1, quer enviar um pacote para o endereço IP 192.168.10.100. O endereço do *host* de destino não pertence à mesma rede local, assim a camada de Internet do *host* transmissor, terá que consultar sua tabela de roteamento. A Tabela 3.1 apresenta uma representação simplificada da tabela de roteamento. As tabelas atuais de roteamento possuem mais informações do que as apresentadas aqui.

Tabela 3.1 Tabela simplificada de roteamento.

Destino	**Gateway**
192.168.5.3	192.168.0.252
192.168.10.0	192.168.0.253
Padrão (Default)	192.168.0.254

A camada Internet primeiro procura pelo endereço completo de destino 192.168.10.100. Como ela não encontra nada, então ela parte para a procura da rede

no qual o *host* de destino está localizado – 192.168.10.0. Neste caso, ela encontra a rede na segunda linha da tabela de roteamento. A camada Internet então pega o endereço do roteador (*gateway*) correspondente – endereço 192.168.0.253 – e envia o pacote para esse roteador.

Quando o datagrama chega a esse roteador, ele é passado da camada de Acesso à rede para a camada Internet. A camada Internet desse roteador, então segue os mesmos procedimentos pela procura de uma máquina de destino para encaminhar o datagrama adiante. A única diferença é que os roteadores geralmente estão conectados a várias redes, o que significa que a interface de rede apropriada para transmitir os dados, também terá que ser escolhida. O processo inteiro é repetido até que o datagrama chegue ao seu destino final. A Figura 3.11 ilustra esse roteamento.

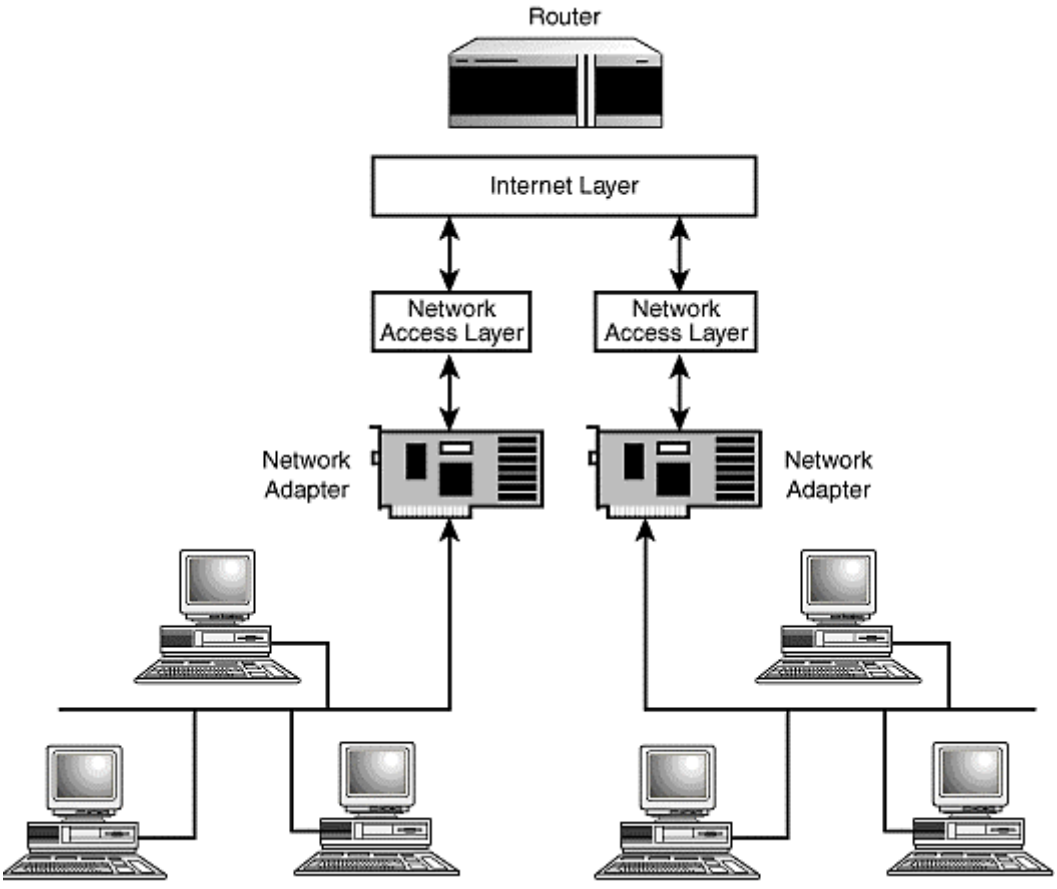

Figura 3.11 Roteador interligando duas redes.

Para garantir que uma boa rota seja escolhida, alguns roteadores comunicam-se um com os outros. Eles trocam informações de roteamento e baseado nelas cada roteador atualiza sua tabela de roteamento, assim ela conterá a melhor rota conhecida para cada destino. O tipo de informação e o modo como essa troca é feita são

determinados pelos protocolos de roteamento, que estão sendo usados. Exemplos de protocolos de roteamento são o *Open Shortest Path First Protocol* (OSPF) e o *Border Gateway Protocol* (BGP) [20]. A Figura 3.12 ilustra roteadores interligando dois *hosts*.

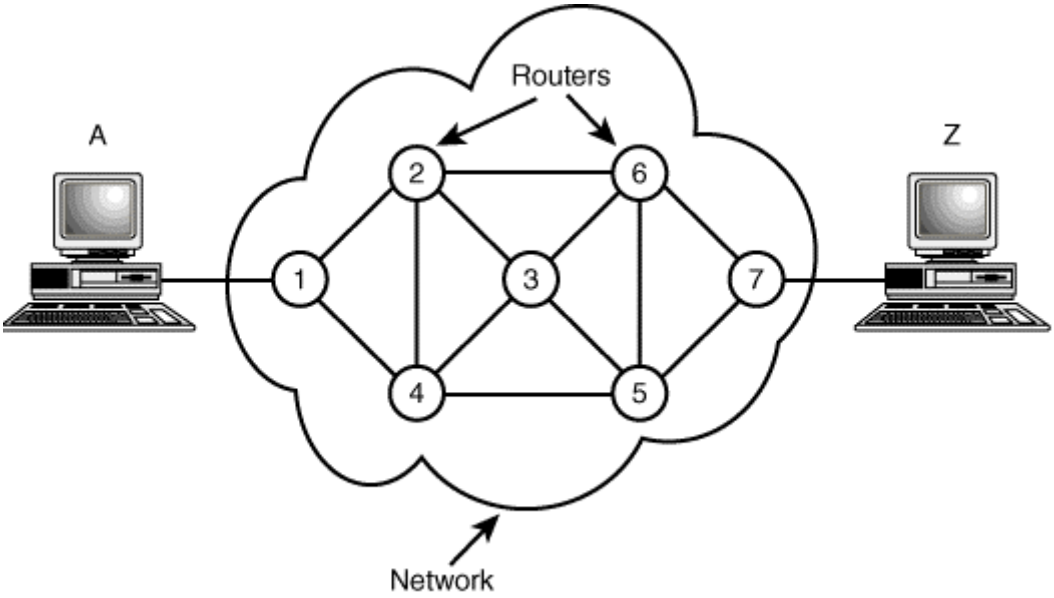

Figura 3.12 Roteadores interligando dois *hosts*.

3.4.4 *Multicasting*

Basicamente existem três modos de transmissão, que podem ser usados para enviar um datagrama IP. Esses modos são chamados de *unicast*, *multicast* e *broadcast* [20].

Unicasting é quando simplesmente se envia um datagrama de uma origem para um destino, como se observa na Figura 3.13. O termo *broadcasting* é utilizado quando se quer enviar um datagrama para todos os *hosts* de uma rede específica.

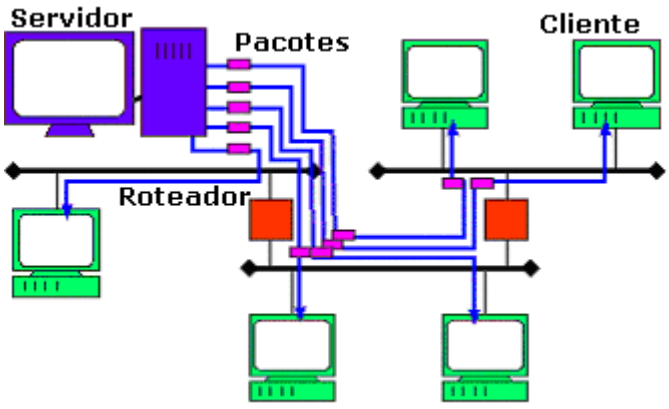

Figura 3.13 Modo de transmissão *unicast*, onde é enviado um datagrama para cada cliente.

Quando se pretende enviar um datagrama para um conjunto arbitrário de *hosts*, então esse modo é chamado de *multicasting*, como se observa na Figura 3.14.

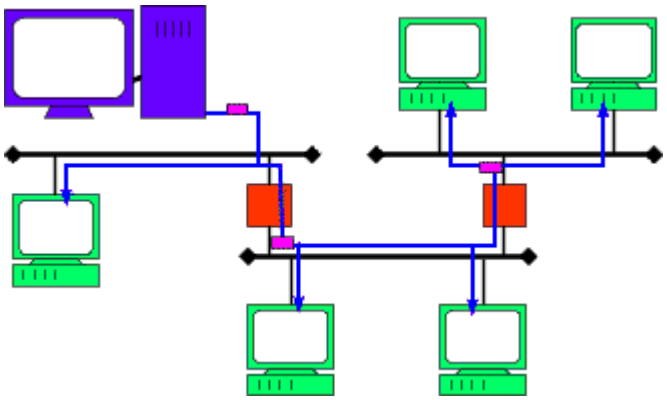

Figura 3.14 Modo de transmissão *multicast*, onde um único pacote é enviado para todos os clientes de um mesmo grupo *multicast*.

A maneira mais simples de se implementar uma transmissão *multicast* seria enviando de modo *unicast* uma cópia do datagrama para cada destino. Esse método, obviamente, consome vários recursos. Uma maneira melhor, seria transmitir um único datagrama que somente será duplicado nos pontos onde ele necessite seguir rotas diferentes para chegar aos seus destinos. Um *host*, para ser capaz de receber datagramas direcionados para um certo endereço *multicast*, deve primeiro juntar-se a um grupo *multicast*. O gerenciamento desse grupo é feito pelo *Internet Group Management Protocol* (IGMP) [20].

De forma geral, o protocolo IGMP trabalha da seguinte maneira: cada *host* mantêm uma lista dos grupos *multicast* dos quais ele quer receber datagramas. Roteadores *multicast* periodicamente enviam consultas IGMP usando *broadcasting* nas redes nas quais eles estão conectados. Os *hosts* então analisam essas consultas e enviam respostas IGMP contendo os grupos nos quais eles estão interessados.

Uma vez que essas respostas já tenham sido processadas, os roteadores *multicast* trocam informações entre si utilizando o IGMP e com todas essas informações atualizam suas tabelas de roteamento. Quando eles recebem um datagrama *multicast*, eles podem determinar para quais *hosts* e roteadores *multicast* os datagramas devem ser enviados.

3.5 Características das Redes IP

Quando datagramas têm de viajar através de várias redes, eles necessitam passar por diversos roteadores. Cada roteador tem de examinar todos os datagramas

entrantes e isto introduzirá um certo atraso na comunicação. Estudos têm mostrado que o tempo levado por um pacote para chegar ao seu destino é muito mais afetado pelo número de roteadores que ele tem de passar, do que pela distância geográfica percorrida.

Quando um roteador está extremamente carregado, alguns datagramas podem ser descartados. Essa perda de pacotes geralmente ocorre por um curto período de tempo, sendo que datagramas consecutivos são perdidos.

Roteadores comunicam-se um com os outros e, dinamicamente, adaptam suas tabelas de roteamento para o estado atual da rede. Dessa forma, datagramas indo para um mesmo destino, podem algumas vezes seguir caminhos diferentes. Mesmo que mudanças nas rotas não ocorram freqüentemente durante uma transmissão, datagramas podem seguir caminhos diferentes. Dessa forma, datagramas podem chegar ao destino fora de ordem.

Além da perda de pacotes e a entrega fora de ordem no destino, pode acontecer que pacotes sejam duplicados durante a transmissão. Isto causará que dois ou mais pacotes idênticos cheguem ao destino, possivelmente com algum atraso entre eles.

Finalmente, um outro aspecto importante das redes IP é o fato que quando a origem envia datagramas para um certo destino, o intervalo de tempo para chegar ao destino difere para cada datagrama. Isto é geralmente chamado de variação do atraso ou simplesmente *jitter*.

3.6 Protocolos de Transporte

Os dois protocolos mais comuns da camada de transporte da arquitetura TCP/IP são o *Transmission Control Protocol* (TCP) e o *User Datagram Protocol* (UDP). Cada um deles oferece um tipo específico de serviço que as aplicações podem utilizar para a comunicação através da rede.

3.6.1 Protocolo TCP – *Transmission Control Protocol*

Atualmente, sem dúvida nenhuma, o TCP é o protocolo mais utilizado dos dois. Este protocolo transforma o serviço baseado em datagramas não seguro (sem garantia de entrega dos dados) e sem conexão da camada Internet em um serviço orientado a conexão e confiável. O protocolo foi projetado para a comunicação entre dois *hosts* e portanto somente suporta a comunicação *unicasting*.

Para oferecer este tipo de serviço, o módulo TCP tem que realizar uma série de

tarefas. Primeiramente, uma conexão terá que ser estabelecida, e isto será feito de uma maneira segura. O módulo tem que garantir que a conexão não será estabelecida acidentalmente, como por exemplo, devido a pacotes duplicados.

O conjunto de *bytes* (*stream*) a serem transmitidos, devem ser segmentados antes da transmissão. No lado do receptor, este conjunto terá de ser reconstruído. Alguns cuidados como o descarte de datagramas duplicados e a ordenação de datagramas fora de ordem devem ser implementados. Algum tipo de mecanismo será utilizado para terminar com a perda de pacotes.

Tudo isso é realizado muito efetivamente pelo TCP. Para estabelecer uma conexão o módulo do TCP utiliza um mecanismo de negociação (*handshake*), chamado de *three-way handshake* [16]. Os problemas de datagramas duplicados e fora de ordem são solucionados utilizando-se números de seqüência. A perda de pacotes é tratada por um mecanismo de confirmação de entrega (*acknowledgement*). Assim, todos os *bytes* de uma série (*stream*) devem ter o seu recebimento confirmado pelo destino. Se a origem não recebe essa confirmação depois de um certo intervalo de tempo, ela envia os dados necessários novamente. O protocolo especifica também um mecanismo de controle de fluxo, o qual previne o transbordamento de um receptor mais lento, e um mecanismo de controle para evitar congestionamento.

O modo como o módulo do TCP trabalha é muito mais completo do que esse breve resumo exposto. Para uma especificação mais completa consulte [16] [34] [35] [36] .

3.6.2 Protocolo UDP – *User Datagram Protocol*

Aplicações que não requerem a funcionalidade do TCP, podem utilizar o UDP. Para transmitir dados, o módulo UDP simplesmente entrega um cabeçalho UDP seguido dos dados para a camada Internet, a qual envia os datagramas à sua maneira. Isto significa, que como o IP, o UDP é um serviço de melhor esforço (*best-effort*). Nenhuma garantia sobre a entrega é dada e datagramas ainda podem chegar fora de ordem e serem duplicados. Uma especificação completa do UDP pode ser encontrada em [16] [34] [38] [39] [40].

O cabeçalho UDP é mostrado na Figura 3.15. Ele possui as portas de origem e de destino, as quais identificam as aplicações que estão enviando e recebendo. Depois, ele informa o número de *bytes* que devem ser enviados e finalmente contém um espaço para uma verificação (*checksum*) opcional.

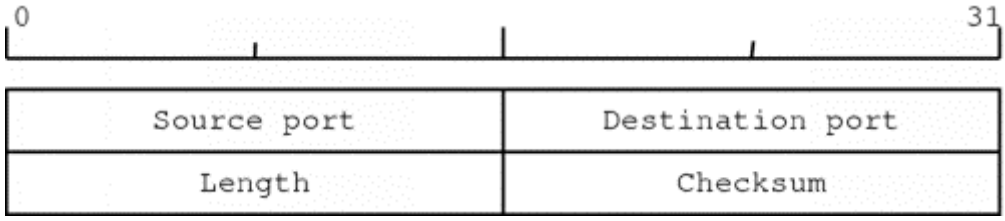

Figura 3.15 Cabeçalho UDP.

Uma vez que o serviço oferecido pelo o UDP é muito idêntico ao serviço do IP, é possível para aplicações enviarem pacotes UDP para endereços *multicast* e receberem pacotes UDP de um grupo *multicast*.

3.7 Por Que Usar IP?

Este protocolo foi projetado simplesmente para o transporte de dados e portanto, tem um limitado suporte para qualidade de serviços (QoS). Então, por que utilizá-lo? A principal razão é por causa da sua onipresença. A arquitetura tem mostrado ser muito popular e é largamente utilizada pelo mundo. Este fato dá ao IP uma enorme vantagem sobre outros protocolos.

Alternativas para a transmissão de pacotes com informações fisiológicas incluiriam FR (*Frame Relay*) e ATM (*Asynchronous Transfer Mode*), assim o nome seria mudado para MonitorFR ou MonitorATM. Ambos permitem um melhor suporte para o tráfego de tempo-real que uma rede IP, porém essas tecnologias não são largamente utilizadas como o IP.

4. QUALIDADE DE SERVIÇO (QoS)

No capítulo anterior, apresentou-se o *Internet Protocol* (IP) de uma maneira geral, sem se preocupar com a qualidade de serviços nas redes IP. Genericamente, a maioria das novas aplicações, como voz sobre IP (VoIP), vídeo-conferência, educação à distância, aplicações de tempo-real, telemedicina e outras, são normalmente conhecidas como aplicações multimídia à medida que envolvem a transferência de múltiplos tipos de mídia (dados, voz, vídeos, sinais vitais) com requisitos de tempo e sincronização para a sua operação com qualidade.

O protocolo IP oferece um serviço sem conexão baseado em datagramas, que não garante a entrega dos mesmos a tempo, não garante que eles cheguem ao destino na ordem correta e nem mesmo garante que cheguem ao destino. Os roteadores fazem o melhor que podem, esforçam-se ao máximo, mas não podem garantiar a entrega dos pacotes. Esse tipo de serviço, sem conexão, é conhecido como serviço de melhor esforço (*Best Effort*). No serviço de melhor esforço, a rede tenta encaminhar todos os pacotes o mais rápido possível, mas não pode fazer qualquer tipo de garantia quantitativa sobre a Qualidade de Serviço (QoS) [17].

De uma maneira geral, QoS especifica o grau de satisfação ou visão do usuário com relação à prestação de um serviço e pode ser definida quantitativamente em termos de parâmetros como, por exemplo, taxa de perda, atraso fim a fim, e *jitter* (variação do atraso). A provisão de QoS é fundamental para os negócios e aplicações de tempo real, tais como: telemedicina, videoconferência e educação à distância. Portanto, uma explosão considerável de atividades de pesquisa vem sendo conduzida no desenvolvimento de protocolos e arquiteturas para suportar tanto o tráfego tradicional, quanto o tráfego multimídia e de tempo real na Internet.

4.1 Qualidade de Serviço (QoS)

A obtenção de uma QoS adequada é um requisito de operação da rede e de seus componentes para viabilizar a operação de uma aplicação com qualidade.

Qualidade de Serviço (QoS) é um requisito da(s) aplicação(ões) para a qual exige-se que determinados parâmetros (atrasos, vazão, perdas) estejam dentro de limites bem definidos (valor mínimo, valor máximo).

A QoS é garantida pela rede, seus componentes e equipamentos utilizados. Do ponto de vista dos programas de aplicação, a QoS é tipicamente expressa e solicitada

em termos de uma "Solicitação de Serviço" ou "Contrato de Serviço". A solicitação de QoS da aplicação é denominada tipicamente de SLA (*Service Level Agreement*) [18] [19]. Um exemplo típico de SLA para uma aplicação de voz sobre IP (VoIP - *Voice over IP*) com algumas centenas de canais de voz simultâneos numa rede IP WAN poderia ser:

- Vazão ≥ 2 Mbps;
- Atraso ≤ 250 ms;
- Disponibilidade ≥ 99,5%.

Uma vez que a rede garanta esta SLA, tem-se como resultado que a aplicação VoIP em questão poderá executar garantindo a qualidade de voz prevista para os seus usuários se comunicando simultaneamente através da rede IP.

4.2 QoS - Parâmetros

Como definido anteriormente, a QoS necessária às aplicações é definida em termos de uma SLA. Na especificação das SLAs são definidos os parâmetros de qualidade de serviço:

- Vazão (Banda);
- Atraso (Latência);
- *Jitter* (Variação do Atraso);
- Taxa de Perdas, Taxa de Erros;
- Disponibilidade;
- Outros.

4.2.1 Quais Aplicações Necessitam de QoS?

Inicialmente, é necessário considerar que não são todas as aplicações que realmente necessitam de garantias fortes e rígidas de qualidade de serviço (QoS) para que seu desempenho seja satisfatório. Dentre as novas aplicações, as aplicações de VoIP (Voz sobre IP), multimídia, telemedicina são, normalmente, aquelas que têm uma maior exigência de QoS.

No mínimo, as aplicações sempre precisam de vazão (banda) e, assim sendo, este é o parâmetro mais básico e certamente mais presente nas especificações de QoS. Este parâmetro da qualidade de serviço é normalmente considerado durante a fase de projeto e implantação da rede e corresponde a um domínio de conhecimento bem discutido e relatado na literatura técnica.

As considerações que seguem tentam identificar as exigências em termos de QoS das aplicações multimídia ilustrando algumas situações práticas.

Uma aplicação multimídia *offline* envolvendo, por exemplo, dados, gráficos e arquivos com animação (vídeo) não necessita de sincronização e, assim sendo, não necessita de cuidados especiais (QoS) da rede. Observe que, se há dados correspondente a uma animação que, em termos práticos, necessita de uma determinada vazão, eventualmente carrega a rede, mas não exige atrasos máximos, sincronização ou tempo de resposta. Este é um caso típico onde a necessidade de QoS reduz-se a uma necessidade de vazão, normalmente atendida pelo próprio projeto da rede.

Por outro lado, para o monitoramento em tempo-real dos sinais vitais de um paciente, garantir apenas a vazão não é suficiente. Neste caso específico, os atrasos de comunicação e as perdas de pacotes influenciam na interatividade com os médicos e na qualidade da aplicação. Considerando números, se o MonitorIP gera uma vazão (fluxo de dados) de 16 kbps, mesmo a utilização de uma LP (Linha Privada) em redes de longas distâncias (WAN - *Wide Área Network*) de 256 kbps pode não ser suficiente. Neste caso, os atrasos e perdas decorrentes da operação podem prejudicar a qualidade da aplicação. Diz-se então que a aplicação exige uma qualidade de serviço da rede.

4.2.2 Vazão

A vazão (banda) é o parâmetro mais básico de QoS e é necessário para a operação adequada de qualquer aplicação.

Em termos práticos, as aplicações geram vazões que devem ser atendidas pela rede. A Tabela 4.1 em seguida ilustra a vazão típica de algumas aplicações.

Como discutido, o atendimento do requisito vazão para a qualidade de serviço é um dos aspectos levados em conta no projeto da rede.

Tabela 4.1 Vazão típica de aplicações de rede.

Aplicação	**Vazão (Típica)**

Aplicações Transacionais	1 kbps a 50 kbps
Sinais Vitais (Bioelétricos)	4 kbps a 160 kbps
Quadro Branco (*Whiteboard*)	10 kbps a 100 kbps
Voz	10 kbps a 120 kbps
Aplicações Web (WWW)	10 kbps a 500 kbps
Transferência de Arquivos (Grandes)	10 kbps a 1 Mbps
Vídeo (*Streaming*)	100 kbps a 1 Mbps
Aplicação Conferência	500 kbps a 1 Mbps
Vídeo MPEG	1 Mbps a 10 Mbps
Aplicação Imagens Médicas	10 Mbps a 100 Mbps
Aplicação Realidade Virtual	80 Mbps a 150 Mbps

4.2.3 Latência (Atraso)

A latência e o atraso são parâmetros importantes para a qualidade de serviço das aplicações. Ambos os termos podem ser utilizados na especificação de QoS, embora o termo latência seja convencionalmente mais utilizado para equipamentos e o termo atraso seja mais utilizado com as transmissões de dados (e.g.: atrasos de transmissão, atrasos de propagação).

De maneira geral, a latência da rede pode ser entendida como a somatória dos atrasos impostos pela rede e equipamentos utilizados na comunicação. Do ponto de vista da aplicação, a latência (atrasos) resulta em um tempo de resposta (tempo de entrega da informação - pacotes) para a aplicação.

Os principais fatores que influenciam na latência de uma rede são os seguintes:

- Atraso de propagação (*Propagation Delay*);
- Velocidade de transmissão;
- Processamento nos equipamentos.

O atraso de propagação corresponde ao tempo necessário para a propagação do sinal elétrico ou propagação do sinal óptico no meio sendo utilizado (fibras ópticas, satélite, coaxial) e é um parâmetro imutável onde o gerente de rede não tem nenhuma influência. A Tabela 4.2 ilustra, a título de exemplo, alguns valores para o atraso de propagação entre cidades numa rede WAN utilizando fibras ópticas como meio físico de comunicação.

Tabela 4.2 Exemplos de atrasos de propagação – Fibras Ópticas.

Trecho *(Round Trip Delay)*	Atraso de Propagação
Miami a São Paulo	100 mseg
New York a Los Angeles	50 mseg
Los Angeles a Hong Kong	170 mseg

A velocidade de transmissão é um parâmetro controlado pelo gerente visando normalmente a adequação da rede à qualidade de serviço solicitada. Em se tratando de redes locais (LANs – Local Área Network) [20], as velocidades de transmissão são normalmente bastante elevadas, tendendo a ser tipicamente superior a 10 Mbps dedicada por usuário (e.g.: utilizando LAN *Switches* [21]).

Em se tratando de redes de longa distância (redes corporativas estaduais e nacionais, redes metropolitanas, intranets metropolitanas) as velocidades de transmissão são dependentes da escolha de tecnologia de rede WAN (linhas privadas, *frame relay*, satélite, ATM). Embora exista obviamente a possibilidade de escolha da velocidade adequada para garantia da qualidade de serviço, observam-se neste caso restrições e/ou limitações nas velocidades utilizadas, tipicamente devido aos custos mensais envolvidos na operação da rede. Além desse fator, observa-se também alguma restrição quanto à disponibilidade tanto da tecnologia quanto da velocidade de transmissão desejada. Em termos práticos, trabalha-se em WAN tipicamente com vazões da ordem de alguns *megabits* por segundo (Mbps) para grupos de usuários.

O resultado das considerações discutidas é que a garantia de QoS é certamente mais crítica em redes MAN (*Metropolitam Área Network*) e WAN pelo somatório de dois fatores, ambos negativos:

- Trabalha-se com velocidades (vazão) mais baixas;
- A latência (atrasos) é muito maior quando se compara com o cenário das redes locais.

O terceiro fator que contribui para a latência da rede é a contribuição de atraso referente ao processamento realizado nos equipamentos. A título de exemplo, numa rede IP os pacotes são processados ao longo do percurso entre origem e destino por:

- Roteadores (comutação de pacotes);
- LAN *Switches* (comutação de quadros);
- Servidores de Acesso Remoto (RAS) (comutação de pacotes);
- *Firewalls* (processamento no nível de pacotes ou no nível de aplicação);
- Outros.

Considerando que a latência é um parâmetro fim-a-fim, os equipamentos finais (*hosts*) também têm sua parcela de contribuição para o atraso. No caso dos *hosts*, o atraso depende de uma série de fatores, a saber:

- Capacidade de processamento do processador;
- Disponibilidade de memória;
- Mecanismos de *cache;*
- Processamento nas camadas de nível superior da rede (programa de aplicação, camadas acima da camada IP);
- Outros.

Em resumo, observe-se que os *hosts* são também um fator importante para a qualidade de serviço e, em determinados casos, podem ser um ponto crítico na garantia de QoS. Esta consideração é particularmente válida para equipamentos servidores (*Servers*) que têm a tarefa de atender solicitações simultâneas de clientes em rede.

4.2.4 Jitter

O *jitter* é um outro parâmetro importante para a qualidade de serviço. No caso, o *jitter* é importante para as aplicações executando em rede cuja operação adequada depende de alguma forma da garantia de que as informações (pacotes) devam ser processadas em períodos de tempo bem definidos. Este é o caso, por exemplo, de aplicações de voz sobre IP (VoIP), aplicações de tempo real, telemedicina, etc.

Do ponto de vista de uma rede de computador, o *jitter* pode ser entendido como a variação no tempo e na seqüência de entrega das informações (por exemplo: pacotes) (*Packet Delay Variation*) devido à variação na latência (atrasos) da rede.

Conforme discutido, no item anterior, a rede e seus equipamentos impõem um atraso à informação (e.g.: pacotes) e este atraso é variável devido a uma série de fatores, a saber:

- Tempos de processamento diferentes nos equipamentos intermediários (roteadores, *switches*);
- Tempos de retenção diferentes impostos pelas redes públicas (*Frame relay*, ATM, X.25, IP);
- Outros fatores ligados à operação da rede.

A Figura 4.1 ilustra o efeito do *jitter* entre a entrega de pacotes na origem e o seu processamento no destino. Observe que o *jitter* causa não somente uma entrega

com periodicidade variável (*Packet Delay Variation*) como também a entrega de pacotes fora de ordem.

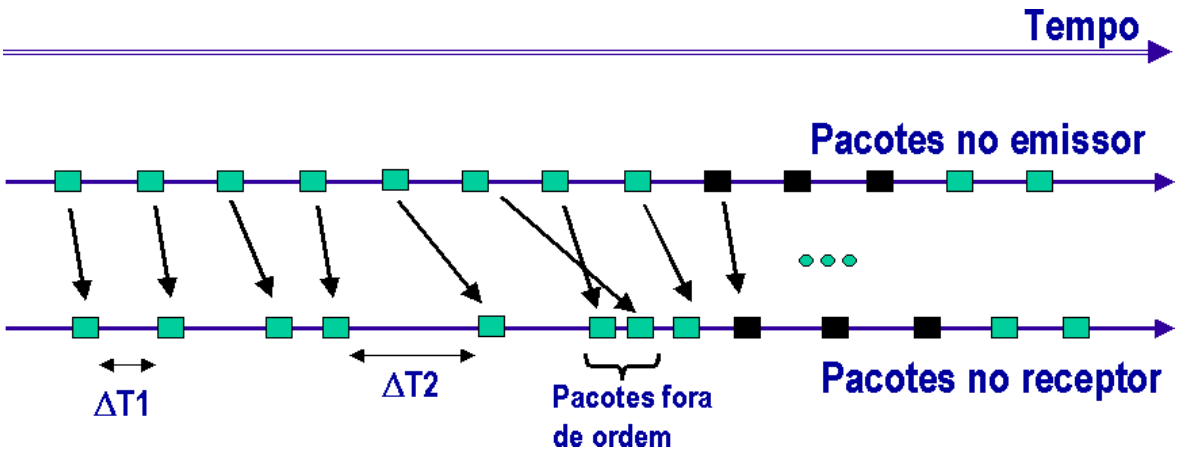

Figura 4.1 Efeito do *jitter* para aplicações com restrições temporais.

Em princípio, o problema dos pacotes fora de ordem poderia ser resolvido com o auxílio de um protocolo de transporte como o TCP (*Transmission Control Protocol*) [34] que verifica o sequenciamento das mensagens e faz as devidas correções. Entretanto, na prática tem-se que a grande maioria das aplicações com restrições temporais optam por utilizar o UDP (*User Datagram Protocol*) [21] ao invés do TCP pela maior simplicidade e menor *overhead* deste protocolo. Nestes casos, o problema de sequenciamento deve ser resolvido por protocolos de mais alto nível normalmente incorporados à aplicação como, por exemplo, o RTP (*Real Time Transfer Protocol*) [22][24].

O *jitter* introduz distorção no processamento da informação na recepção e deve ter mecanismos específicos de compensação e controle que dependem da aplicação em questão. Genericamente, uma das soluções mais comuns para o problema consiste na utilização de *buffers* (técnica de *buffering*).

4.2.5 Perdas

As perdas de pacotes em redes IP ocorrem principalmente em função de fatores tais como:

- Descarte de pacotes nos roteadores e *switch routers* (erros, congestionamento);
- Perda de pacotes devido a erros ocorridos na camada 2 (PPP - *Point-to-Point Protocol*, *ethernet*, F*rame Relay*, ATM) durante o transporte dos mesmos.

De maneira geral, as perdas de pacotes em redes IP são um problema sério para determinadas aplicações como, por exemplo, o MonitorIP. Neste caso específico, as perdas de pacotes, com trechos de sinais vitais digitalizados, implicam numa perda de qualidade eventualmente não aceitável para a aplicação. O que fazer em caso de perdas de pacotes é uma questão específica de cada aplicação em particular.

Do ponto de vista da qualidade de serviço da rede (QoS) a preocupação é normalmente no sentido de especificar e garantir limites razoáveis (Taxas de Perdas) que permitam uma operação adequada da aplicação.

4.2.6 Disponibilidade

A disponibilidade é um aspecto da qualidade de serviço abordada normalmente na fase de projeto da rede.

Em termos práticos, a disponibilidade é uma medida da garantia de execução da aplicação ao longo do tempo e depende de fatores tais como:

- Disponibilidade dos equipamentos utilizados na rede proprietária (rede do cliente) (LAN, MAN ou WAN);
- Disponibilidade da rede pública, quando a mesma é utilizada (operadoras de telecomunicações, *carriers*, ISPs - *Internet Service Providers*).

As empresas dependem cada vez mais das redes de computadores para a viabilização de seus negócios (comércio eletrônico, *home-banking*, atendimento *online*, transações *online*) e, neste sentido, a disponibilidade é um requisito bastante rígido. A título de exemplo, requisitos de disponibilidade acima de 99% do tempo são comuns para a QoS de aplicações WEB, aplicações cliente/servidor, aplicações de forte interação com o público, e aplicações de telemedicina.

5. TRANSMISSÃO DE SINAIS VITAIS

No capítulo anterior, os parâmetros básicos necessários para a garantia da qualidade de serviço (QoS) de uma aplicação foram apresentados de uma forma geral, sem dar-se ênfase ao monitoramento em tempo-real dos sinais vitais. Neste capítulo serão especificados tais parâmetros para garantir a transmissão de sinais vitais sobre IP com qualidade.

5.1 Aquisição e Apresentação

Para tornar possível o envio de informações fisiológicas de pacientes através de uma rede de computadores, os sinais bioelétricos têm de ser codificados para uma representação digital. Esses sinais são captados por eletrodos, amplificados e digitalizados.

A digitalização é feita através de equipamentos de aquisição de sinais bioelétricos. Não é o objetivo deste trabalho mostrar o funcionamento de tal sistema de aquisição, mas sim apresentar como monitorar sinais vitais remotamente através de uma rede IP em tempo-real. Em [23] apresentou-se um dispositivo de aquisição, o qual possui 16 canais independentes com parâmetros configuráveis via software. A Figura 5.1 apresenta um diagrama de blocos simplificado do hardware de aquisição. Observe que esse sistema é composto por um amplificador de instrumentação (AI) com ganho ajustável (G1=1, 10, 100 ou 1000), um filtro passa alta (FPA) de 1ª ordem para eliminação de artefatos com freqüência de corte de 0,05 Hz, um filtro passa baixa (FPB) Bessel de 8ª ordem com freqüência de corte f_c ajustável (0 – 50 kHz), um bloco de ajuste fino de ganho (G2=256 níveis), freqüência de amostragem de 100 Hz até 100 kHz e um conversor A/D de 12 *bits*.

Os sinais digitalizados por um dispositivo de aquisição agora podem ser empacotados e transmitidos através de uma rede IP. A fim de manter os aspectos de tempo-real do monitoramento, é necessário que o receptor comece a receber os sinais assim que possível, depois que a transmissão tenha iniciado. Desta forma, pequenos pacotes com os sinais digitalizados são enviados em intervalos regulares através da rede, e prontos para serem processados pelo receptor.

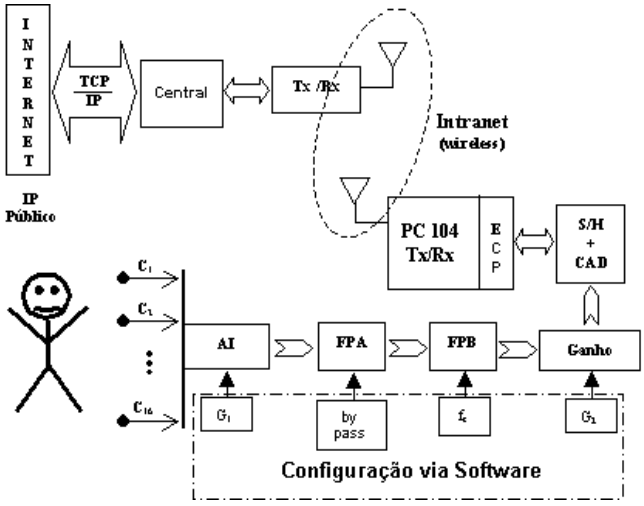

Figura 5.1 Diagrama em blocos simplificado do hardware de aquisição dos sinais.

Quando um pacote é recebido, ele é mostrado na tela do médico como um gráfico de forma de onda. Assim, o médico verá todas as informações fisiológicas do paciente na tela de seu computador, como se ele estivesse visualizando em um monitor de multi-parâmetros ao lado do paciente. Os vários sinais fisiológicos são digitalizados e enviados separadamente através da rede, ou seja, cada pacote de dados enviado corresponde a um sinal em específico. Com isso, conseguimos aproveitar melhor as características de cada sinal para aplicarmos técnicas particulares de compressão, e o médico tem a opção de escolher quais informações ele quer visualizar; isto facilita a implementação da transmissão e permite uma escalabilidade do sistema para novos sinais. Dessa forma, o receptor ao receber um pacote, identifica pelo seu cabeçalho a qual dos sinais pertence e, então, os apresenta em uma determinada região da tela do computador destinada para o sinal identificado.

Cada pacote possui também informações do instante em que as amostras foram adquiridas e, com isso, permite ao receptor sincronizar os diversos sinais. Dessa forma, como se observa na Figura 5.2, os dois sinais vitais estão sincronizados segundo uma mesma linha de tempo, no eixo horizontal.

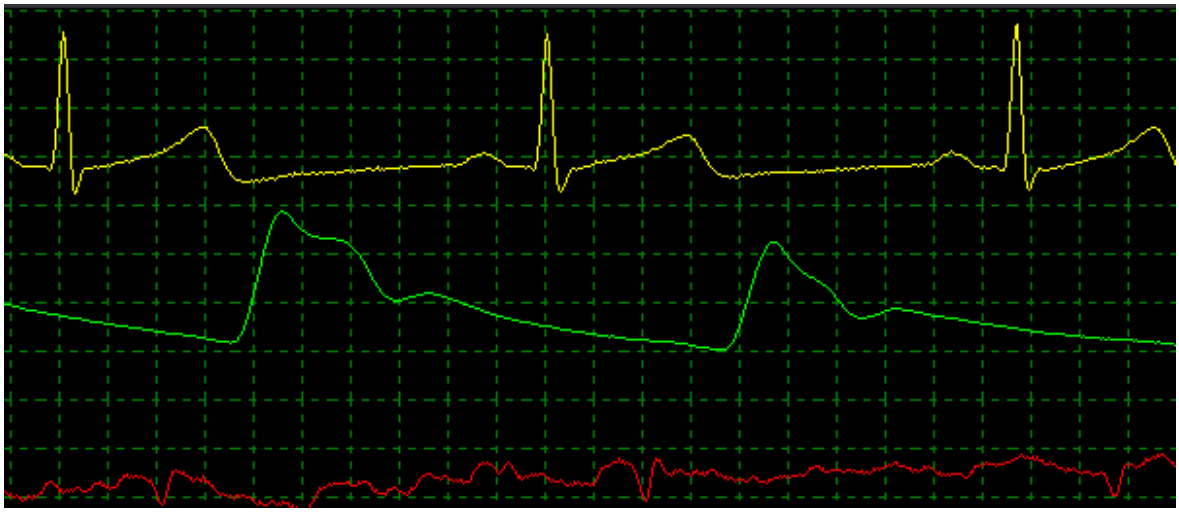

Figura 5.2 Tela do MonitorIP apresentando em tempo-real os sinais vitais de um paciente.

5.2 Requerimentos da Comunicação

Quando o médico está diante do monitor de multi-parâmetros, analisando os sinais vitais de um paciente, tipicamente ocorrem muito poucos erros e o atraso no monitoramento é muito baixo. Esses erros e atrasos são quase que exclusivamente gerados pelo sistema de aquisição em si, o qual após digitalizar e quantitizar o sinal, quase que instantaneamente apresenta-o na tela do monitor. Também nesse caso não ocorrem problemas como a variação do atraso no monitoramento. Com o monitoramento remoto, cada pacote tipicamente chegará com uma pequena diferença na quantidade de atraso, resultando no *jitter*. Também não existe nenhuma garantia sobre o atraso causado pela rede. Além disso, alguns pacotes poderão chegar com erros ou nem mesmo todos os pacotes poderão chegar.

Nesta sessão, serão discutidos os requerimentos necessários para realizar o monitoramento com qualidade dos sinais vitais de pacientes. Entende-se por monitoramento com qualidade, uma forma de monitoramento na qual o médico possa realizar o diagnóstico sem receio sobre a qualidade dos dados recebidos.

5.2.1 Vazão

Sabe-se que quanto maior a taxa de amostragem e maior a precisão do conversor analógico-digital (A/D), melhor será a representação digital do sinal original. Mas isto também significa que uma maior quantidade de informação terá de ser transmitida e, portanto, uma maior vazão (largura de banda) será requerida. Assim, será preciso determinar quanto de informação é necessária para realizar o

monitoramento com qualidade.

Pelo critério de Nyquist, sabe-se que a mínima freqüência de amostragem deve ser maior que duas vezes a maior freqüência do sinal analógico. Os sinais vitais típicos encontrados podem ser digitalizados utilizando-se uma freqüência de amostragem de 250 Hz e uma resolução de 16-*bits* para o conversor A/D, o que será suficiente para permitir aos médicos um perfeito monitoramento.

Agora, pode-se calcular a largura de banda requerida para o monitoramento dos sinais vitais. Utilizando-se uma freqüência de amostragem de 250 Hz e um valor de 16 *bits* para representar cada amostra, em um segundo serão transmitido 250 amostras de 16 *bits*. Isto requer uma vazão de pelo menos 4000 bps (*bits per second*) ou 4 kbps por canal transmitido. No sistema MonitorIP, até quatro canais (i.e, sinais) podem ser monitorados ao mesmo tempo o que solicitará uma vazão mínima de 16 kbps (sem levar em consideração o tamanho dos cabeçalhos dos protocolos envolvidos, apenas os dados).

5.2.2 Atraso

Em aplicações típicas, como acessar uma página na web, o atraso levado por um pacote em ir do servidor até o seu computador não é relevante. Porém, em aplicações multimídia, como por exemplo uma tele-cirurgia, o atraso total é extremamente importante. O tempo levado por um cirurgião observar remotamente, uma nova alteração ocorrida em um dos sinais vitais do paciente, deverá ser o menor possível.

O atraso total pode ser categorizado em dois tipos. O primeiro tipo é o atraso fixo. Este é o atraso que estará sempre presente durante todo o caminho da transmissão, influenciado principalmente pela velocidade de propagação do sinal elétrico no meio de transmissão. O segundo tipo é o atraso variável. Este é o atraso causado principalmente pelo enfileiramento dos pacotes nos roteadores e, conseqüentemente, pelo congestionamento na rede. Um outro atraso importante e que será adicionado artificialmente, é o atraso devido aos pacotes deixados em um *buffer* no receptor para compensar o *jitter*.

5.2.3 *Jitter*

Quando pacotes de dados são enviados pela rede, existem pequenas variações no tempo que cada pacote leva para chegar ao destino. De acordo com o

congestionamento que ocorrer na rede, essas variações podem ser grandes. A importância dada a essas variações é a seguinte: suponha-se que seja apresentado na tela de um médico os dados fisiológicos de um pacote, assim que o mesmo seja recebido. Por causa do *jitter* (variação do atraso), é possível que o próximo pacote a ser exibido não tenha chegado no instante de ser apresentado, assim uma pequena paralisação no monitoramento ocorreria até a chegada desse pacote, ou ainda, um pacote poderia chegar antecipadamente a um outro, o qual deveria ser apresentado posteriormente a esse. Uma vez que o *jitter* não ocorre por um curto período de tempo, mas continuamente, ele é muito prejudicial para manter uma boa interatividade do sinal monitorado.

Para evitar esse problema, uma técnica simples de *buffering* será utilizada. Ao invés de apresentar na tela os dados de um pacote assim que este é recebido, coloca-se esse pacote em um *buffer* e uma pequena quantidade de atraso é introduzida. Uma vez, que isto é feito para todos os pacotes, existe uma alta probabilidade que ao final da apresentação das informações de um pacote, o próximo já estará disponível no *buffer*. Porém, esse *buffering* introduzirá uma certa quantidade de atraso. Portanto cuidados devem ser tomados para que o atraso total não seja muito grande. Normalmente, somente uma pequena quantidade de *buffer* será necessária.

5.2.4 Perdas

Ao contrário do que ocorre na comunicação de dados, onde um pequeno erro pode causar resultados desagradáveis, a comunicação de sinais vitais é mais tolerante à presença de erros. Um erro ocasional não prejudicará seriamente o monitoramento (diagnóstico) remoto realizado pelo médico, desde que o erro não afete uma porção relativamente grande do sinal.

Durante a transmissão dos sinais vitais através de uma rede IP, pacotes podem ser corrompidos ou mesmo se perderem. Para reduzir a quantidade de informação perdida, um pacote deveria conter somente uma pequena quantidade de informação de sinal vital. Desta maneira, se um pacote fosse perdido, somente uma pequena faixa do sinal vital seria perdida, assim essa pequena perda não seria suficiente para atrapalhar o monitoramento ou diagnóstico pelo médico.

5.2.5 Disponibilidade

Em uma aplicação de telemedicina existe uma dependência das redes de

computadores para a viabilização de, por exemplo, uma cirurgia remota. Neste sentido, a disponibilidade é um requisito bastante rígido. Idealmente, a disponibilidade deveria ser de 100%, mas na prática isso não ocorre. Como se trata de uma aplicação crítica, uma disponibilidade maior que 99,99% deve ser negociada com os ISPs (*Internet Service Providers*).

5.3 Protocolos de Transmissão

Ao se transmitir pacotes contendo os dados digitalizados de um sinal vital, um mecanismo que preserve a sincronização do sinal vital faz-se necessário. Ou seja, pacotes consecutivos devem ser apresentados no tempo certo e na ordem correta. Esse tipo de sincronização é chamado de sincronização intramídia. Da mesma forma, um mecanismo para sincronizar entre si os diversos sinais vitais sendo monitorados também faz-se necessário. Esse tipo de sincronização é chamado de sincronização intermídia.

Se uma aplicação quer transmitir dados, ela utiliza para tal, um determinado protocolo. Como observou-se na Seção 3.6, a arquitetura TCP/IP disponibiliza dois protocolos de transporte, o TCP e o UDP, os quais podem ser utilizados pelas aplicações para transmitir seus dados.

Será mostrado que os protocolos TCP e UDP não são suficientes para o monitoramento remoto de sinais vitais em tempo-real. O protocolo RTP (*Real-time Transport Protocol*) será proposto como solução para esse problema. O protocolo RTP é largamente utilizado em aplicações multimídia de tempo-real, como videoconferência e VoIP.

5.3.1 Por Que nem TCP e nem UDP?

No início do projeto MonitorIP, pensava-se que o protocolo TCP seria um bom candidato para se transmitir os sinais vitais. Ele oferece um serviço orientado a conexão e confiável. O TCP garante que todos os dados cheguem ao receptor corretamente e na mesma ordem em que foram transmitidos, o que seria muito bom para o monitoramento remoto de sinais vitais. Além do mais, ele ainda oferece mecanismos de controle de fluxo e congestionamento, os quais permitem uma boa proteção contra a sobrecarga da rede.

Existem, porém, várias desvantagens de se utilizar o TCP. Um dos problemas básicos é que para oferecer um serviço confiável de envio de dados, o protocolo realiza

indeterminadamente a retransmissão de pacotes perdidos ou corrompidos. Enquanto isto oferece um serviço confiável, no qual a ordem é preservada, a espera pela retransmissão de pacotes adiciona um atraso extra na comunicação. No monitoramento em tempo-real é melhor contar com a perda ocasional de pacotes ou com pacotes corrompidos eventualmente, do que se adicionar uma grande quantidade de atraso extra, para se garantir uma transmissão segura.

Uma conseqüência disso é que um pacote perdido ou corrompido efetivamente proíbe a aplicação de receber quaisquer pacotes que cheguem depois deste, uma vez que o TCP preserva a ordem dos pacotes. A aplicação tem que desenhar no vídeo os sinais vitais em intervalos regulares, assim se um pacote permanece perdido por um longo intervalo de tempo, isto impedirá que outros pacotes sejam apresentados no vídeo, mesmo quando eles já tenham chegado.

Os recursos de controle de fluxo e de congestionamento do TCP podem parecer muito úteis, mas uma aplicação tem muito pouco controle sobre isto. O TCP pode facilmente decidir por si só decrementar a taxa nos quais os dados são enviados, e isto novamente incrementaria o atraso total.

O ponto chave a ser apresentado aqui é que o protocolo TCP tem uma série de propriedades e de complexidades, as quais não são muito úteis ao monitoramento em tempo-real e que só colaboram para o incremento do atraso total. Pode-se obter uma igual ou melhor performance utilizando-se um protocolo menos elaborado (com um menor *overhead*).

Quando os sinais vitais de um paciente têm que ser enviados para mais de um médico ao mesmo tempo, o TCP tem outra desvantagem significativa. Enquanto o protocolo IP oferece uma eficiente distribuição de dados utilizando *multicasting*, o TCP não tem suporte para isto. Se um dado tem que ser transmitido para vários destinos usando o TCP, uma conexão para cada destino terá que ser realizada, e esse dado enviado repetidamente para cada uma delas. Dessa forma, é claro, uma maior largura de banda (vazão) será necessária quando se utiliza o TCP.

Ao eliminar-se a utilização do TCP do projeto do MonitorIP, o UDP seria outro protocolo básico de transporte o qual se poderia utilizar. Este protocolo simplesmente é uma extensão do protocolo IP, com nenhuma complexidade adicionada, assim ele só oferece um serviço de melhor-esforço (*best-effort*).

O protocolo UDP tem a vantagem de não ter que esperar pela retransmissão de pacotes perdidos. Além disso, como ele é apenas uma extensão do protocolo IP, ele

pode utilizar o recurso de *multicasting* do IP e economizar largura de banda quando o dado precisa ser transmitido para vários destinos. Tão bom quanto isto possa parecer, existem também algumas desvantagens: o UDP não fornece absolutamente nenhum mecanismo para sincronização e não existe também nenhum controle de fluxo e congestionamento.

A solução para esse problema é estender o protocolo UDP para algo que se possa adicionar informação extra sobre os dados de sinais vitais contidos em cada pacote e utilizar o UDP para distribuir essas informações de controle e os dados de sinais vitais. Esse é o papel que o protocolo RTP (*Real-time Transport Protocol*) realiza na arquitetura TCP/IP e será o protocolo utilizado para a transmissão dos sinais vitais no projeto MonitorIP. A seguir, no Capítulo 6, será discutido o protocolo RTP.

6. *REAL-TIME TRANSPORT PROTOCOL* (RTP)

O *Real-time Transport Protocol* é formalmente especificado em [22] [24]. Ele é definido como um protocolo que fornece um serviço de entrega fim-a-fim para dados com características de tempo-real, tais como áudio e vídeo interativos.

A especificação RTP atualmente define dois protocolos. O primeiro deles é o *Real-time Transport Protocol* (RTP). O segundo é o *RTP Control Protocol* (RTCP). A função do RTP é transferir dados de tempo-real. O protocolo de controle RTCP fornece informações sobre a qualidade da transmissão dos dados e informações sobre os participantes na sessão. Os protocolos são definidos de tal maneira que eles podem ser utilizados em várias arquiteturas de redes e não somente em redes TCP/IP. No entanto, quando o RTP e o RTCP são usados em uma rede TCP/IP, eles são tipicamente utilizados sobre o protocolo UDP, uma vez que o esquema de transmissão do TCP não é adaptado para dados que precisam ser transportados com uma latência muito baixa, como no caso de uma telecirurgia. Nesse caso, o RTP é tradicionalmente associado a uma porta UDP de número par e o RTCP, à próxima porta UDP de número ímpar.

Os protocolos RTP e RTCP em si não fornecem nenhum mecanismo de garantia do fornecimento dos dados a tempo, da entrega dos mesmos e nem previnem a entrega fora de ordem. Eles de maneira nenhuma controlam a qualidade de serviço (QoS). Esses requisitos têm de ser fornecidos por alguns outros mecanismos, como por exemplo o protocolo de reserva de recursos, o RSVP (*Resource reSerVation Protocol*). Porém, os protocolos fornecem os dados necessários para que a aplicação possa colocar os pacotes na ordem correta. Da mesma forma, o protocolo RTCP provê informações sobre a qualidade da recepção, com a qual a aplicação pode usar para fazer ajustes locais. Por exemplo, se um congestionamento está se formando, a aplicação pode decidir diminuir a taxa de dados. O RTP e o RTCP simplesmente permitem aos receptores compensar o *jitter* da rede, por meio do controle de *buffer* e sequenciamento, e ter mais informações a respeito da rede de maneira que medidas corretivas apropriadas possam ser adotadas (redundância, codecs a taxas mais baixas, etc.).

Na sessão seguinte, será dada uma introdução para o protocolo RTP e RTCP. Essas sessões são baseadas na especificação em [22], a qual é uma apropriada fonte de consulta para informações mais detalhadas sobre esses itens.

6.1 *Real-time Transport Protocol* (RTP)

O RTP permite o transporte de dados isócronos através de uma rede de pacotes que introduz *jitter* e que pode tirar a seqüência dos pacotes. Normalmente é usado em cima do UDP, que fornece a noção de porta e a detecção de erro. Quando usa o UDP, o RTP pode ser transportado por pacotes IP *multicast*, ou seja, pacotes com endereços de destino *multicast* (e.g.: 224.34.54.23). Dessa forma, um fluxo RTP gerado por uma única fonte pode atingir vários destinos.

6.1.1 Alguns Usos do RTP

Número de seqüência e *timestamp*: cada pacote RTP transporta um número de seqüência e um *timestamp*. Dependendo da aplicação, eles podem ser usados de várias formas. Uma aplicação de vídeo, por exemplo, pode deduzir imediatamente a partir do *timestamp*, qual parte da tela é descrita pelo pacote IP. Mesmo que a aplicação ainda não tenha recebido nenhum pacote, devido a problemas de deseqüenciamento ou perdas, ela ainda pode usar o pacote para construir a parte da imagem que ele descreve.

Uma aplicação de áudio não pode fazer o mesmo (porque não entenderíamos a fala corrompida) e usa o número de seqüência e o *timestamp* para gerenciar um *buffer* de recepção. Uma aplicação pode determinar, por exemplo, que vai armazenar 100 milisegundos de fala no *buffer* antes de iniciar sua reprodução. Cada vez que um novo pacote RTP chega, ele é colocado no *buffer* na posição apropriada, dependendo do seu número de seqüência. Se um pacote não chegar a tempo e ainda estiver faltando no instante da reprodução, a aplicação poderá optar por copiar o último quadro do pacote que acabou de ser reproduzido e repeti-lo até chegar a vez do próximo pacote recebido ou usar algum esquema de interpolação definido pelo codec de áudio que estiver sendo usado.

Tipo de *payload* (***Payload Type*** – **PT**): o *payload* de cada pacote RTP é a informação em tempo real contida em cada pacote. O seu formato é completamente livre e deve ser definido pela aplicação ou pelo perfil do RTP em uso. Com o objetivo de distinguir um formato em particular dos demais, sem que seja necessário analisar o conteúdo do *payload*, o cabeçalho de cada pacote RTP contém um identificador de tipo de *payload*. Eles são chamados de tipos de *payload* estáticos, designados pela *Internet Assigned Numbers Authority* (IANA). Os tipos de *payload* de número 96 a 127 são reservados para PTs dinâmicos, ou seja, o codec usado com esses PTs é negociado

dinamicamente por um protocolo de controle de conferência.

Como o RTP não define, ele próprio, o formato da seção de *payload*, cada aplicação deve definir ou referir-se a um perfil.

6.1.2 Algumas Definições

Sessão RTP: uma sessão RTP é uma associação de participantes que se comunicam no RTP. Cada participante usa dois endereços de transporte (no caso do IP, por exemplo, duas portas UDP na máquina local) para cada sessão: uma porta para o fluxo RTP e uma para as mensagens RTCP. Quando uma transmissão *multicast* é usada, todos os participantes usam o mesmo par de endereços de transporte *multicast*. Fluxos de dados na mesma sessão devem compartilhar um canal RTCP comum.

Fonte de sincronização SSRC (*Synchronisation Source Identifier*): fonte de um fluxo RTP, identificada por 32 *bits* no cabeçalho RTP. Todos os pacotes RTP com um SSRC comum possuem uma mesma referência de tempo e de sequênciamento.

Fonte contribuinte CSRC: quando um fluxo RTP é o resultado de uma combinação de vários fluxos contribuintes feita por um misturador (*mixer*) RTP, a lista, com os SSRCs de cada um dos fluxos contribuintes, é adicionada ao cabeçalho RTP do fluxo resultante como uma lista de CSRCs. O fluxo resultante tem o seu próprio SSRC. Esse recurso não é utilizado no monitoramento de sinais vitais de pacientes.

Formato NTP (*Network Time Protocol*): uma maneira padrão de formatar um *timestamp*, escrevendo-se o número de segundos passados desde 01/01/1900 às 00:00 com 32 *bits* para a parte inteira e 32 *bits* para a parte decimal (esta expressa como o número de $1/2^{32}$ de segundo – por exemplo, 0x80000000 significa 0,5 segundo). Existe também uma maneira compacta com apenas 16 *bits* para a parte inteira e 16 *bits* para a parte decimal. Os primeiros 16 dígitos da parte inteira geralmente podem ser obtidos a partir do dia corrente e a parte fracionária simplesmente é truncada para os 16 dígitos mais significativos.

6.2 O Pacote RTP

Um pacote RTP consiste de um cabeçalho RTP, seguido pelos dados a enviar. Na especificação, esses dados são referenciados como *payload*. Todos os campos que

precedem a lista CSRC estão sempre presentes em um pacote RTP (veja a Figura 6.1). A lista CSRC está presente apenas depois que o pacote passa por um misturador (*mixer*), e portanto, não estará presente no contexto do monitoramento de sinais vitais.

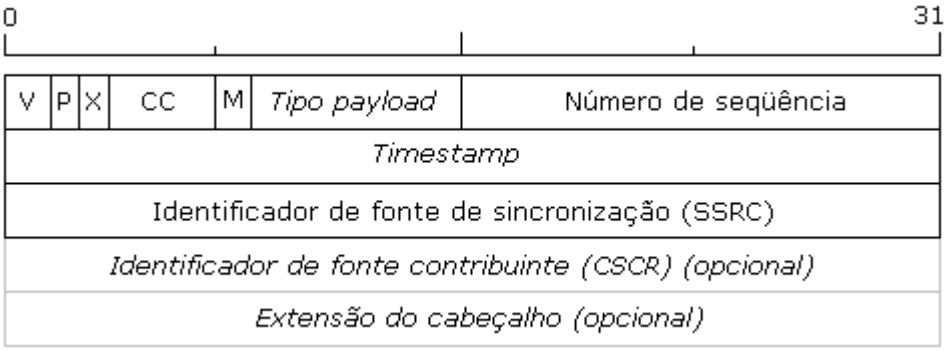

Figura 6.1 Cabeçalho de um pacote RTP.

***Version* (V)**: Os primeiros 2 *bits* do cabeçalho são reservados para a versão do RTP. A versão atual é a 2.

***Padding* (P)**: 1 *bit* de *padding* indica se o *payload* sofreu enchimento para fins de alinhamento. Se tiver havido enchimento (P=1), então o último octeto do campo *payload* indica precisamente quantos octetos de enchimento foram acrescentados ao *payload* original.

***Extension* (X)**: 1 *bit* de extensão X indica a presença de extensões após eventuais CSRCs do cabeçalho fixo.

CSRC *Count* (CC): 4 *bits* e informa quantos identificadores CSRC vêm após o cabeçalho fixo. O valor de CC pode ir de zero (o mais usual) a 15.

***Marker* (M)**: 1 *bit* e o seu uso é definido pelo perfil RTP, ou seja, a exata interpretação não é definida na especificação RTP, isto é deixado para a aplicação definir seu uso. Por exemplo, codificações de áudio que suportam supressão de silêncio, ele deve ser colocado em 1 no primeiro pacote de cada período de fala (*talkspurt*) subseqüente a um período de silêncio.

***Payload Type* (PT)**: 7 *bits* e identifica o tipo de dado transportado pelo pacote. Alguns tipos de *payload* estáticos são definidos no RFC 1889 e no RFC '*Assigned Numbers*' (RFC 1700).

Sequence Number: 16 *bits* e começa com um valor aleatório e é incrementado a cada pacote RTP. Pode ser usado por uma aplicação para colocar os pacotes recebidos na ordem correta.

Timestamp: 32 *bits* e contém informação de sincronismo para uma seqüência de *bytes* de pacotes. Este valor especifica quando o primeiro *byte* do *payload* foi amostrado. Por exemplo, para áudio *timestamp* é tipicamente incrementado com a quantidade de amostras no pacote. Baseado neste valor, a aplicação receptora pode então reproduzir os dados de áudio exatamente no tempo certo. Da mesma forma que o número de seqüência, o valor inicial do *timestamp* é aleatório. Observe que vários pacotes podem ter o mesmo valor para o *timestamp*: com vídeo digitalizado, por exemplo, uma imagem geralmente será enviada em vários pedaços. Esses pedaços todos terão um número de seqüência diferente, mas o valor de seu *timestamp* será o mesmo.

***Synchronisation Source Identifier* (SSRC)**: é o número identificador da aplicação que enviou o pacote. Se uma aplicação tiver de enviar diferentes mídias ao mesmo tempo, por exemplo áudio e vídeo, ela deverá ter sessões separadas para cada uma das mídias. Desta maneira, uma aplicação pode agrupar dados entrantes de acordo com o valor de SSRC. O identificador é escolhido aleatoriamente e a chance de duas partes comunicando-se acidentalmente terem o mesmo SSRC valor é extremamente pequena. Em casos raros em que isto acontecer, a especificação fornece um apropriado curso das ações que devem ser tomadas para resolver esse problema.

Observe que o cabeçalho não contém um campo com o tamanho da área de *payload*. O protocolo, da camada inferior da pilha de protocolos, é que deve determinar o tamanho do *payload*. Por exemplo, na arquitetura TCP/IP, RTP é usado sobre o UDP, o qual contém informação do comprimento. Assim, uma aplicação pode determinar o tamanho do pacote RTP inteiro e depois que seu cabeçalho estiver sido processado, ela automaticamente saberá a quantidade de dados em sua área de *payload*.

6.3 RTP *Control Protocol* (RTCP)

O protocolo RTP é acompanhado por um protocolo de controle, o RTCP. Cada participante de uma sessão RTP, periodicamente, envia pacotes RTCP para todos os outros participantes na sessão. Como definido em [22], RTCP tem quatro funções:

1) A função principal é a de fornecer um *feedback* na qualidade da distribuição dos dados. Tais informações podem ser usadas pelas aplicações com o intuito de criar funções de controle de fluxo e de congestionamento. A informação também pode ser usada para

propósitos de diagnóstico da transmissão.

2) O RTCP distribui um identificador o qual pode ser usado para agrupar diferentes *streams* (seqüência de *bytes*) – por exemplo áudio, vídeo e sinais vitais pertencentes a um mesmo participante.
3) Enviando pacotes RTCP periodicamente, cada sessão pode determinar o número de participantes. O RTP não pode ser usado para isto, uma vez que podem existir participantes exclusivamente passivos, ou seja, não enviam nenhum pacote RTP, como por exemplo, em uma palestra *on-line*. Esses participantes passivos enviam somente pacotes RTCP.
4) Uma função opcional é a distribuição das informações sobre os participantes. Essas informações podem ser usadas em uma interface com o usuário, por exemplo, mostrando os dados dos participantes.

Existem vários tipos de pacotes RTCP para suportar as funcionalidades citadas acima. O ***sender report*** (SR) é um tipo de pacote RTCP usado por participantes ativos (transmissores), e tem a finalidade de fornecer estatísticas sobre a transmissão e recepção de pacotes RTP. Um participante passivo (receptor), que recebe apenas pacotes RTP, também fornece estatísticas da recepção de pacotes RTP, através de pacotes RTCP do tipo ***receiver report*** (RR). Informações as quais descrevem um participante são transmitidas por pacotes RTCP do tipo ***source description*** (SDES). Existem também pacotes RTCP do tipo ***application*** (APP), que fornecem informações definidas pela aplicação. Finalmente, quando um participante está para deixar uma sessão RTP, ele envia um pacote RTCP do tipo ***goodbye*** (BYE), com a finalidade de avisar a todos os participantes de sua saída.

As estatísticas de transmissão fornecidas por um transmissor (participante ativo), incluem o número de *bytes* enviados e o número de pacotes enviados, além de dois *timestamps*: um *Network Time Protocol* (NTP) *timestamp*, o qual fornece o horário da criação desse relatório e um RTP *timestamp*, que descreve o mesmo horário, mas no mesmo formato dos *timestamp* dos pacotes RTP, usando o mesmo valor aleatório utilizados nos pacotes RTP dessa sessão.

Isto é particularmente útil quando várias *streams* têm que ser relacionadas umas com as outras. Por exemplo, no MonitorIP, cada sinal vital é enviado em uma sessão RTP. Cada sessão RTP tem um diferente *timestamp* uma da outra, pois no início da sessão é feito um sorteio para definir esse *timestamp* e os seus pacotes subseqüentes são apenas incrementos desse valor. Como existe a necessidade de

sincronizar os diversos sinais vitais apresentados na tela do computador, precisa-se descobrir a relação dos vários valores de *timestamp* de cada sinal, e isto é fornecido pelo pacote RTCP, quando informa em um pacote do tipo *sender report* o valor do *timestamp* da sessão e seu valor equivalente ao NTP (*Network Time Protocol*) *timestamp*, dessa forma as aplicações receptoras podem fazer os cálculos necessários para sincronizar as várias *streams*.

Um participante em uma sessão RTP, distribui estatísticas da recepção de pacotes RTP de cada participante ativo (*sender*) na sessão. Para cada transmissor (*sender*) é fornecido um relatório específico da recepção que contém as seguintes informações:

- Uma fração (pacotes recebidos/pacotes esperados*256) da perda de pacotes desde o último relatório enviado. Um aumento neste valor, pode ser usado como uma indicação de congestionamento;
- A quantidade total de perda de pacotes desde o início da sessão;
- Uma estimativa da variância do tempo entre chegadas (*interarrival jitter*) subseqüentes de pacotes RTP, medida nas mesmas unidades que o *timestamp* RTP. Quando esse valor de *jitter* aumenta, pode também representar uma possível indicação de congestionamento. Esse valor pode ser utilizado pelas aplicações para determinar dinamicamente o tamanho do *buffer* necessário para eliminação do *jitter*;
- Informações que podem ser usadas pelos transmissores para calcular o tempo de ida-e-volta (*round trip time*) para o seu receptor. O valor do *round trip time* é o tempo que um pacote leva para viajar até o seu receptor e voltar.

Os pacotes RTCP do tipo *source description* (SDES) fornecem informações gerais sobre os participantes, como nome e *e-mail*. Mas também incluem um nome que identifica, de forma única, o transmissor dos pacotes RTP, e denominado de *canonical name* (CNAME). Cada participante pode ter várias sessões RTP, uma para cada tipo de *stream*, como é o caso do MonitorIP. Em cada sessão, o participante tem um identificador SSRC para identificar os seus pacotes RTP, mas o valor desse identificador é diferente para as outras sessões que ele participa. Ou seja, o sinal de ECG de um paciente enviado em uma sessão RTP (definida para a transmissão dos sinais de ECG de todos os pacientes) possui um identificador, o qual identifica os pacotes RTP desse paciente dentro dessa sessão. Um outro valor identificador é usado

para a transmissão da saturação de oxigênio desse paciente em uma sessão RTP específica para a transmissão de saturação de oxigênio de todos os pacientes. Portanto, a única forma de agrupar esses diferentes identificadores de cada sinal é através do CNAME, o qual é único para cada paciente.

Uma vez que esses pacotes RTCP são enviados periodicamente por cada participante para todos os destinos, tem-se que tomar o cuidado de não utilizar demasiadamente a largura de banda disponível. O intervalo de envio de pacotes RTCP é calculado de acordo com o número de participantes e a quantidade de largura de banda disponível que esses pacotes RTCP podem utilizar. Para evitar que todos os participantes enviem pacotes RTCP ao mesmo tempo e saturem a largura de banda disponível, é somado ao cálculo desse intervalo um valor aleatório.

7. ARQUITETURA DO SISTEMA MONITORIP

Neste capítulo será apresentado o funcionamento do sistema MonitorIP e a disposição e configuração de seus componentes pela rede de computadores. Em seguida, diagramas de classes e de sequenciamento serão apresentados, com a finalidade de mostrar como cada um dos componentes do sistema foram implementados. O sistema MonitorIP é dividido em dois subsistemas: PacienteIP e MonitorIP.

7.1 PacienteIP

O subsistema PacienteIP é conectado ao paciente e é responsável por realizar a aquisição dos sinais vitais e de transmití-los através de uma rede IP para o MonitorIP.

Na Figura 7.1, observa-se o fluxo dos dados e as funções básicas do PacienteIP. A função Aquisição é responsável por controlar a aquisição dos sinais vitais de interesse e a função RTP por empacotar esses dados e por colocá-los na rede.

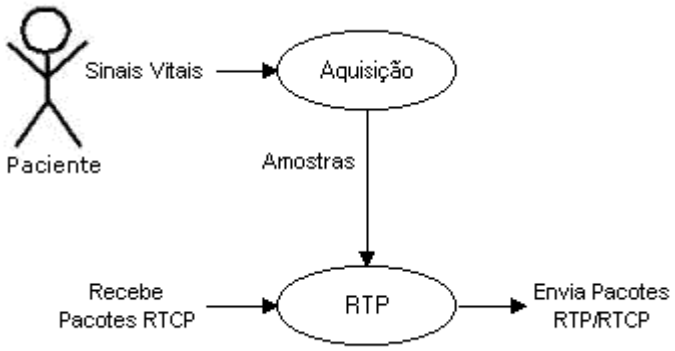

Figura 7.1 Diagrama de sequênciamento do PacienteIP.

A Tabela 7.1, apresenta os arquivos fontes utilizados na implementação do subsistema PacienteIP. Para exemplificar o funcionamento do sistema, será analisada cada etapa do processo.

Tabela 7.1 Arquivos referenciados no PacienteIP.

Arquivo	Função
MonitorIP/PacienteIP/PacienteIP.cpp	*Arquivo principal responsável por gerenciar as sessões RTP e utilizá-las para enviar as amostras capturadas pelo sistema de aquisição.*
MonitorIP/jRTPlib/rtpsession.cpp	*Arquivo com a implementação da classe RTPSession.*
MonitorIP/jRTPlib/rtpsession.h	*Arquivo com a definição da classe RTPSession.*

7.1.1 Aquisição dos Sinais Vitais

A primeira etapa refere-se à aquisição dos sinais vitais. O subsistema PacienteIP é composto por uma placa mãe PC/104 (*embedded system*), apresentado na Figura 7.2, de baixo consumo e de dimensões 10x10cm, com processador MachZ x86 PC de 100MHz, 32M*bytes* de memória RAM, e por um flash IDE driver de 8M*bytes*.

Figura 7.2 Placa mãe PC104 do subsistema PacienteIP, com processador MachZ.

O módulo de aquisição é conectado à placa mãe através da porta paralela e possui 4 canais independentes, permitindo que até 4 sinais vitais sejam monitorados ao mesmo tempo, porém o módulo de aquisição foi projetado para suportar até 16 canais independentes. Os sinais são adquiridos pelo módulo a uma freqüência de amostragem de 250Hz por um conversor A/D com uma resolução de 16 *bits*. Utilizou-

se como sistema operacional o Linux e foi desenvolvido para tal um processo (*daemon*) responsável por controlar a aquisição dos sinais, empacotar os dados adquiridos e transmití-los para o subsistema MonitorIP.

7.1.2 Empacotamento dos Sinais Vitais

Após a aquisição dos sinais vitais, o subsistema PacienteIP empacota os dados e os envia através da rede ao subsistema MonitorIP. A fim de manter os aspectos de tempo-real do monitoramento, é necessário que o subsistema MonitorIP comece a receber os sinais assim que possível. Para tal, pequenos pacotes com sinais vitais são enviados a intervalos regulares através da rede. O tamanho desses pacotes foi escolhido levando-se em conta o atraso máximo aceitável para o envio dos dados após a primeira aquisição e a quantidade de informação perdida com a perda de um pacote.

Cada pacote transporta 50 amostras (100 *bytes*), o que equivale a 200ms do sinal amostrado (fs=250Hz). Dessa forma, após 200ms da aquisição da primeira amostra é que o pacote será enviado (tempo necessário para a aquisição das 50 amostras que o pacote transporta). O número de amostras por pacote não pode ser muito grande para que não afete a visualização do sinal pelo médico no caso de perda do pacote. Para o sinal de ECG, cada pacote (200ms) corresponde em torno de 1/5 do período do sinal. Para diminuir esse atraso (200ms) e o efeito da perda de um pacote poderíamos diminuir o número de amostras do pacote para 5, o que equivaleria a 20ms do sinal ou 1/50 de um período do sinal de ECG. Dessa forma o médico perderia pouca informação a respeito do sinal no caso de perda de algum pacote. No entanto, na transmissão de cada pacote tem-se um *overhead* causado pelas informações nos cabeçalhos dos protocolos envolvidos.

Como se observa na Figura 7.3, primeiramente tem-se o cabeçalho do protocolo RTP, o qual possui um tamanho mínimo de 20 *bytes*. Na camada de transporte tem-se o protocolo UDP, o qual possui um cabeçalho de 8 *bytes*. E na camada de rede tem-se o protocolo IP com um tamanho mínimo de 20 *bytes* para seu cabeçalho, totalizando assim um adicional de 48 *bytes* por pacote transmitido. Dessa forma o tamanho mínimo do datagrama necessário para transmitirmos uma única amostra (2 *bytes*), que seria o caso ideal para diminuir o efeito da perda de um pacote e obter uma maior interatividade, seria de 50 *bytes* (2 *bytes* da amostra + 20 *bytes* do RTP + 8 *bytes* do UDP + 20 *bytes* do IP). Ou seja, um *overhead* de 96%.

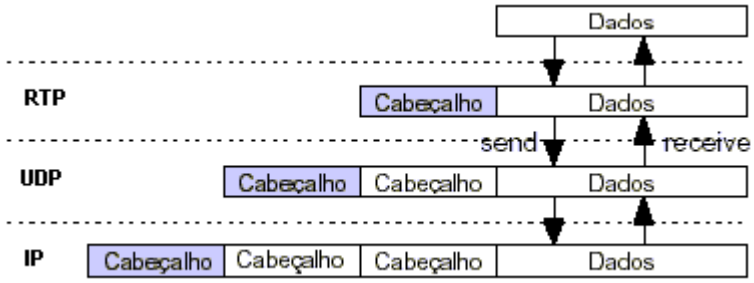

Figura 7.3 Encapsulamento de dados na pilha de protocolos TCP/IP do sistema MonitorIP.

Com uma freqüência de amostragem de 250Hz, tem-se a cada 4ms uma nova amostra a ser enviada, o que nos obriga a ter uma largura de banda de aproximadamente 97,65625 Kbps (250 amostras/s * 50 *bytes*/1 amostra * 8 *bits*) por sinal.

Para diminuir o *overhead* causado pelos cabeçalhos dos protocolos e com isso ter um melhor aproveitamento da largura de banda, deve-se aumentar o número de amostras por pacote. Aumentando-se o número de amostras do pacote para 500, resultaria em um tamanho de 1048 *bytes* e um *overhead* de 4,58%. Como cada amostra é adquirida a cada 4ms e o pacote possui 500 amostras, então cada pacote equivale a 2s do sinal amostrado e a largura de banda necessária seria de aproximadamente 4,09375 Kbps (250 amostras/s * 1048 *bytes*/500 amostras * 8 *bits*), tendo-se assim um melhor aproveitamento da largura de banda.

Contudo, a perda de um pacote corresponde a 2s do sinal monitorado e a interatividade do sistema é muito prejudicada. Qualquer alteração no sinal só seria verificada 2s depois mais o tempo de transmissão e de *jitter*. Com os testes realizados chegamos a conclusão que 50 amostras por pacote seria o melhor custo/beneficio para o nosso sistema. Assim a largura necessária por sinal monitorado seria de aproximadamente 5,78125 Kbps (250 amostras/s * 148 *bytes*/50 amostras * 8 *bits*) e o *overhead* resultante é de 32,43%. Cada pacote corresponde a 200ms do sinal amostrado, o qual mantém ainda uma boa interatividade do sistema.

Na Tabela 7.2, observa-se um comparativo do número de amostras por pacote, com a quantidade de tempo de sinal amostrado correspondente a cada pacote, com a largura de banda necessária e o *overhead* resultante.

Tabela 7.2 Amostras/Pacote X Largura de Banda.

Amostras/Pacote	$t_{pacote\ (ms)}$	Vazão (Kbps)	*Overhead* (%)
1	4	100	96

5	20	24	82
25	100	7,8	48
50	**200**	**5,8**	**32**
100	400	4,8	15
250	1000	4,3	8,7
500	2000	4	4,5
1000	4000	4	2,3

***** A área em destaque indica o número de amostras contidas em cada pacote RTP utilizado pelo subsistema PacienteIP.

7.1.3 Transmissão dos Sinais Vitais

Cada pacote enviado pela rede conterá apenas informações de um único sinal vital, ou seja, os vários sinais sendo monitorados serão enviados em pacotes separados. No sistema MonitorIP, quatro sinais vitais podem ser monitorados ao mesmo tempo, assim, em intervalos regulares quatro pacotes, um para cada sinal, são colocados na rede.

Para cada sinal é criada uma sessão RTP (*Real-time Transport Protocol*), com uma dada porta base. Os pacotes RTP de uma sessão serão enviados por essa porta base (geralmente um valor par) e os pacotes RTCP (*RTP Control Protocol*) serão enviados na porta seguinte (geralmente um valor ímpar). Como foi mostrado no Capítulo 6, os pacotes RTP são os resposáveis por transportarem os dados do sinal vital e os pacotes RTCP fornecem informações sobre a qualidade da transmissão dos dados e informações sobre os participantes na sessão.

No subsistema PacienteIP serão criadas quatros sessões RTP, uma para cada sinal, e os valores das portas dessas sessões são mostrados na Tabela 7.3.

Tabela 7.3 Portas utilizadas em cada sessão/sinal.

	Sessão RTP	
	RTP	RTCP
Sessão/Sinal 1	5000	5001
Sessão/Sinal 2	5002	5003

Sessão/Sinal 3	5004	5005
Sessão/Sinal 4	5006	5007

O endereço de destino desses pacotes, onde estará sendo executado o subsistema MonitorIP, pode ser um endereço *unicast* ou um endereço *multicast*. A opção *multicast* é a escolha preferencial, pois economiza recursos de largura de banda da rede. No endereçamento *multicasting*, se mais que um subsistema MonitorIP estiver monitorando o mesmo paciente, apenas um único pacote para cada sinal terá que ser transmitido (todos os subsistemas MonitorIP configurados para aquele endereço *multicast*, receberam o mesmo pacote através da rede). Se a utilização do endereçamento *multicasting* não for possível, a utilização do endereçamento *unicasting* implicará no cadastramento dos endereços IP de todos os *hosts* que irão monitorar esse paciente, assim várias cópias do mesmo pacote serão enviadas, uma para cada um dos subsistemas MonitorIP, aumentando a utilização dos recursos de largura de banda da rede. O endereço padrão de destino, utilizado pelo sistema MonitorIP, para o envio dos pacotes RTP e RTCP, é o endereço *multicast* 224.0.0.1.

No subsistema MonitorIP são criadas também quatro sessões RTP, uma para cada sinal. Os pacotes RTP/RTCP do sinal 1 serão recebidos pela porta 5000/5001, os do sinal 2 pela porta 5002/5003, os do sinal 3 pela porta 5004/5005 e os do sinal 4 pela porta 5006/5007.

Na Figura 7.4, tem-se dois subsistemas MonitorIP A e B monitorando o mesmo paciente A. Os pacotes enviados pelo paciente A são capturados tanto pelo MonitorIP A como pelo B. Dessa forma, os pacotes RTP com informações do sinal 1 do paciente A são enviados pela porta 5000 do subsistema PacienteIP A e colocados na rede. Os dois subsistemas MonitorIP A e B capturam esses pacotes do sinal 1 através da porta 5000 de cada um de seus *hosts*. O mesmo acontece para os sinais 2, 3 e 4 respectivamente nas portas 5002, 5004 e 5006. O paciente B também envia seus pacotes para as mesmas portas e para o mesmo endereço de destino do paciente A.

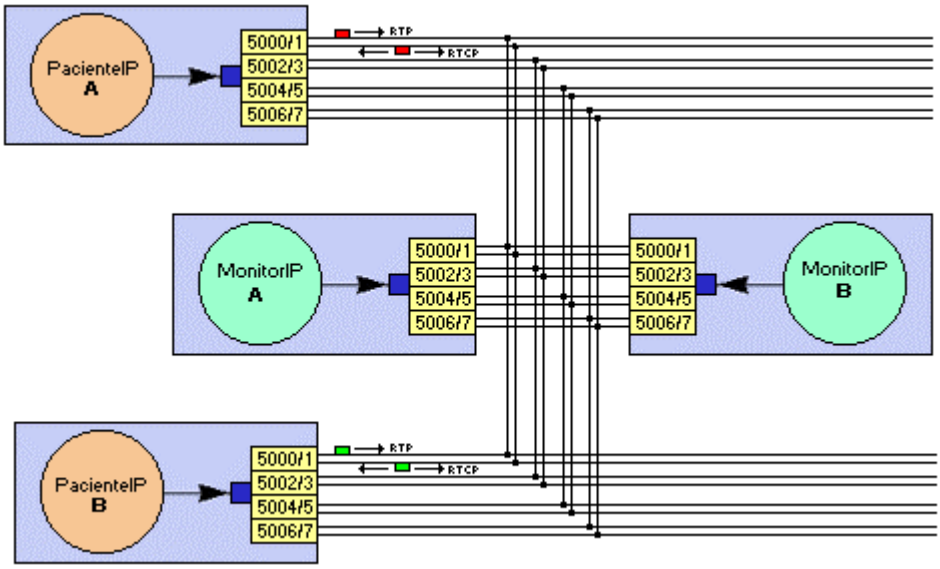

Figura 7.4 Diagrama de comportamento do sistema MonitorIP.

Ao ser criada uma sessão RTP no subsistema PacienteIP para um determinado sinal, um identificador SSRC de 32 *bits* é criado para identificar essa fonte de fluxos RTP (pacotes com sinal vital). O algoritmo para a criação desse identificador foi proposto em [22], de forma que não existirá nenhum outro sinal de outro paciente na porta da sessão com o mesmo identificador. Todos os pacotes RTP com um SSRC comum possuem uma mesma referência de tempo e de sequenciamento e se referem ao mesmo sinal.

Através desse identificador SSRC, o subsistema MonitorIP identifica a qual paciente pertence o pacote que está sendo recebido em uma determinada porta. Como exemplo, o sinal 1 do paciente A e o sinal 1 do paciente B são ambos enviados através da porta 5000, porém, os pacotes referentes ao paciente A possuem um identificador diferente dos pacotes pertencente ao paciente B.

7.1.4 Diagrama de Classe do Subsistema PacienteIP

O subsistema PacienteIP é formado por três classes principais: PacienteIP, RTPSession e RTPSourceData, como pode ser observado na Figura 7.5. A classe PacienteIP é a classe responsável por toda a execução do subsistema PacienteIP. Ela realiza a aquisição dos sinais vitais, empacota as amostras e as transmite para o subsistema MonitorIP. Para realizar essas tarefas ela instância um objeto da classe RTPSession para cada sessão/sinal. A classe RTPSession fornece métodos para a criação de uma sessão RTP, métodos para o envio de pacotes RTP e métodos para o envio de pacotes RTCP com informações a respeito da transmissão.

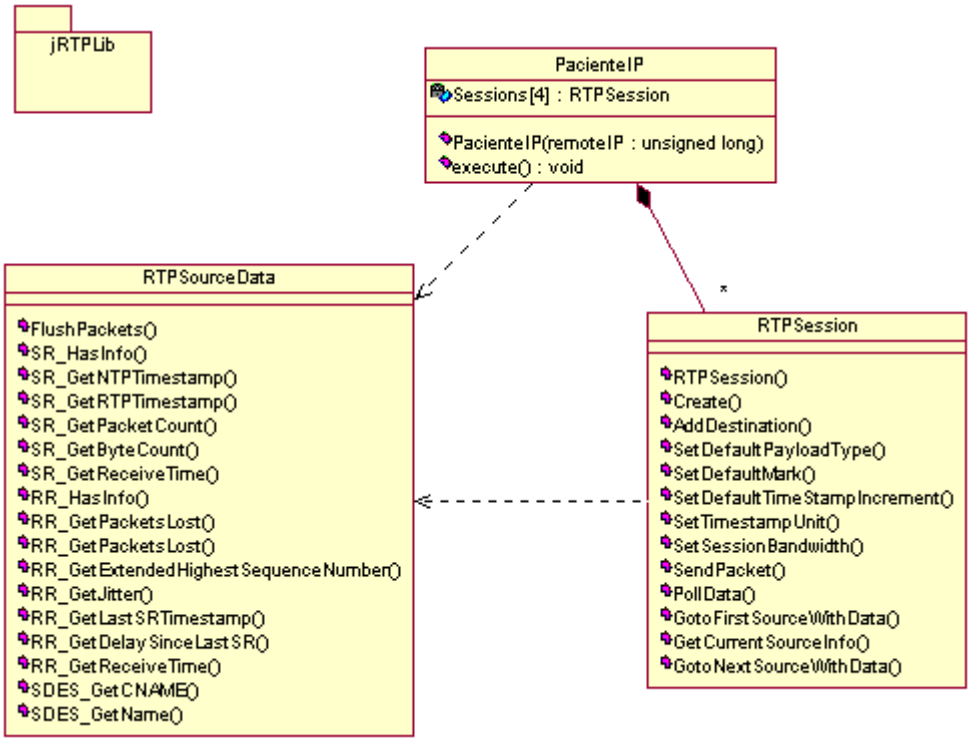

Figura 7.5 Diagrama de classe do subsistema PacienteIP.

7.2 MonitorIP

O subsistema MonitorIP é responsável por receber os sinais vitais enviados pelo PacienteIP e apresentá-los na tela do computador do médico, de forma que este possa monitorar o paciente remotamente e em tempo-real, como se estivesse diante de um monitor de multi-parâmetros.

Na Figura 7.6, observa-se que as funções básicas do subsistema MonitorIP são: Interface, RTP e Sincronismo.

A função interface é responsável pela apresentação dos sinais na tela do computador, bem como pelas configurações de cada sinal. Através da interface, pode-se ajustar a escala horizontal e o deslocamento no eixo vertical, além de se alterar a cor com que o sinal será apresentado e a opção de ligar e desligar o *grid*. A interface apresenta também uma lista dos pacientes que estão enviando pacotes RTP e que podem ser monitorados, e através dessa lista é possível escolher qual desses pacientes se deseja monitorar.

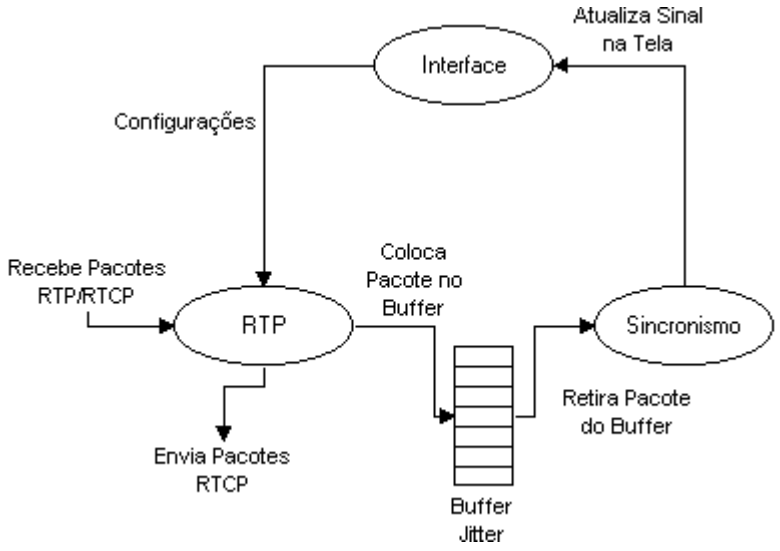

Figura 7.6 Diagrama de sequenciamento do subsistema MonitorIP.

A função RTP cria as sessões RTP para cada um dos sinais e é responsável por receber os pacotes RTP e RTPC. Para cada pacote RTP recebido, a função identifica a qual paciente pertence aquele sinal. Caso o paciente identificado não esteja sendo monitorado, o pacote é descartado, porém, se o paciente está sendo monitorado, o pacote é colocado em um *buffer* específico. Para cada sinal é criado um *buffer*, para compensar o efeito *jitter* causado pela rede. A função RTP calcula constantemente a quantidade de *jitter*, analisando os pacotes RTCP enviados pelo subsistema PacienteIP e, dessa forma, determina o tamanho máximo do *buffer*. Os pacotes RTP possuem um campo em seu cabeçalho que identifica a que instante aquele pacote pertence. Assim, a função ao colocar o pacote no *buffer*, primeiro verifica se aquele instante já ocorreu e caso isso seja verdadeiro, o pacote será descartado. Em seguida o pacote será colocado no *buffer* em ordem cronológica, assim corrigindo a entrega fora de ordem causada pelo *jitter* da rede.

A função sincronismo, como o nome propõe, é responsável por sincronizar os diversos sinais monitorados. Para realizar o sincronismo, a função implementa um relógio, o qual deverá gerar eventos a cada 200 ms (tempo entre a chegada de pacotes). Para cada evento gerado pelo relógio a função retira do *buffer* de cada sinal, os dados a serem apresentados na tela, referentes a aquele intervalo. Se o *buffer* de um dos sinais estiver vazio, ou o primeiro pacote no *buffer* não corresponder ao instante a ser apresentado, um espaço proporcional a esse intervalo é deixado em branco na tela do computador. Isso poderá ocorrer se o pacote se perder na rede ou chegar com erro e for descartado, ou o tamanho do *buffer* não foi suficientemente

grande para evitar o efeito do *jitter*.

Na Figura 7.7, observa-se a interface do subsistema MonitorIP.

Figura 7.7 Interface do subsistema MonitorIP mostrando os sinais vitais de um paciente.

A Tabela 7.4, apresenta os arquivos fontes utilizados na implementação do subsistema MonitorIP. Para exemplificar o funcionamento do sistema, será analisada cada etapa do processo.

Tabela 7.4 Arquivos referenciados no subsistema MonitorIP.

Arquivo	**Função**
MonitorIP/MonitorIP/MonitorIP.cpp	*Arquivo principal responsável por criar o formulário TfrmMain.*
MonitorIP/MonitorIP/Main.cpp	*Arquivo com a implementação da classe TfrmMain.*
MonitorIP/MonitorIP/Main.h	*Arquivo com as definições da classe TfrmMain.*
MonitorIP/MonitorIP/RTPThread.cpp	*Arquivo com a implementação da classe TRTPThread.*
MonitorIP/MonitorIP/RTPThread.h	*Arquivo com as definições da classe TRTPThread.*

MonitorIP/MonitorIP/SyncView.cpp	*Arquivo com a implementação da classe TSyncThread.*
MonitorIP/MonitorIP/SyncView.h	*Arquivo com as definições da classe TSyncThread.*
MonitorIP/MonitorIP/Session.cpp	*Arquivo com a implementação da classe TRTPSession.*
MonitorIP/MonitorIP/Session.h	*Arquivo com as definições da classe TRTPSession.*
MonitorIP/MonitorIP/Patient.cpp	*Arquivo com a implementação da classe TPatient.*
MonitorIP/MonitorIP/Patient.h	*Arquivo com as definições da classe TPatient.*
MonitorIP/MonitorIP/Signal.cpp	*Arquivo com a implementações da classe TSignal.*
MonitorIP/MonitorIP/Signal.h	*Arquivo com as definições da classe TSignal.*
MonitorIP/Package/SignalView.cpp	*Arquivo com a implementação da classe TSignalView.*
MonitorIP/Package/SignalView.h	*Arquivo com as definições da classe TSignalView.*

7.2.1 Diagrama de Classe do Subsistema MonitorIP

O subsistema MonitorIP é formado por três classes principais: TfrmMain, TRTPThread e TSyncThread, como pode ser observado na Figura 7.8.

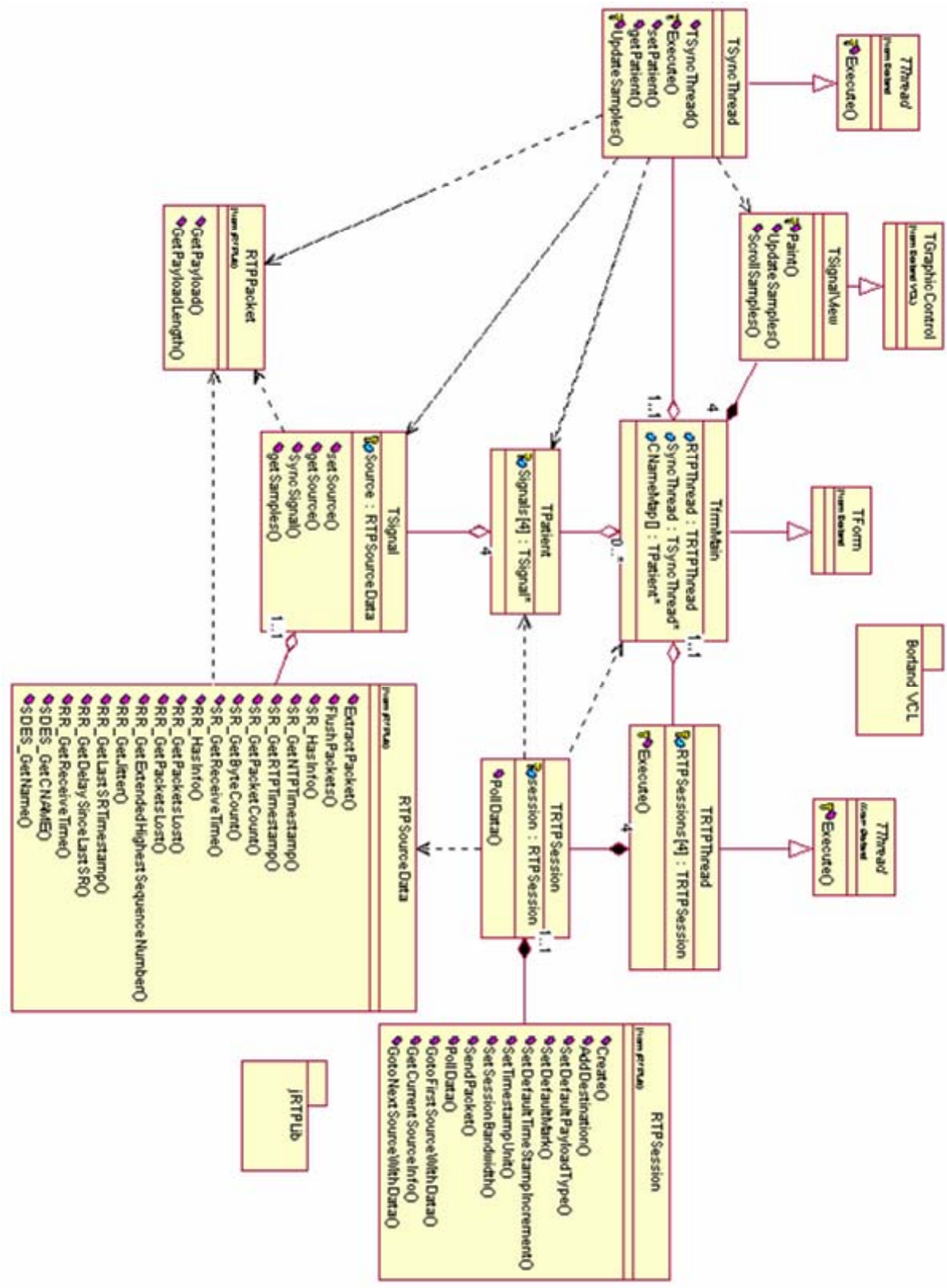

Figura 7.8 Diagrama de classe do subsistema MonitorIP.

8. RESULTADOS PRELIMINARES

Durante os testes foram utilizados os dois subsistemas apresentados no capítulo anterior: PacienteIP e MonitorIP. Além deles, foi desenvolvido um terceiro *software*, responsável por analizar os pacotes RTCP e apresentar informações a respeito da transmissão dos sinais. Esse *software* foi denominado RTPMonitor.

8.1 Testes

Realizaram-se vários testes em dois cenários diferentes de utilização. O primeiro cenário simulava a arquitetura de uma rede local (ethernet) dentro de um hospital. Utilizou-se para tal, a rede local do Instituto de Engenharia Biomédica da Universidade Federal de Santa Catarina. Dois pacientes virtuais, conectados a essa rede, transmitiam seus sinais vitais para outros computadores espalhados por essa rede, onde em cada um deles podia-se monitorar o estado de cada paciente, além de acompanhar a qualidade da transmissão.

O segundo cenário simulava o monitoramento de um paciente localizado em sua residência, através de uma conexão dial-up entre o hospital e sua casa. Dessa forma, pode-se observar a qualidade da transmissão de sinais vitais através de uma linha discada, com pequena largura de banda.

8.2 Cenário 1 – Rede Local

Para a elaboração dos testes deste cenário foram montados dois pacientes virtuais, os quais forneciam os sinais vitais para a transmissão através da rede do laboratório.

O hardware do primeiro paciente virtual formava-se de um sistema embarcado com as seguintes descrição:

- Placa mãe PC104 Tri-m 10x10cm;
- Processador x86 MatchZ 100 MHz;
- Memória RAM de 32 MBytes;
- Flash DiskOnChip de 8 MBytes;
- Placa de rede ethernet PC104 10/100 Mbps;
- Módulo de alimentação PC104.

Utilizou-se o Linux, versão de kernel 2.2.18, como sistema operacional, e a

distribuição utilizada foi a fornecida pela empresa Red Hat, versão 6.2.

Para tornar possível o armazenamento do sistema operacional na *flash* com capacidade de apenas 8 MBytes, realizou-se algumas alterações na distribuição fornecida pela Red Hat. Primeiramente, removeu-se todos os *softwares* e arquivos desnecessários para a aplicação em questão. E, em seguida, compactou-se os *softwares* e arquivos restantes, com o aplicativo "gzip", repeitando-se a estrutura de diretórios. Essa imagem de aproximadamente 6 M*Bytes* foi armazenada na memória *flash* do equipamento. Compilou-se o kernel do Linux, alterando-se algumas opções para que este reconhecesse a memória *flash* (DiskOnChip 2000) e descompactasse essa imagem em um RAM *disk,* montado na memória RAM do PC104, durante a inicialização (*boot*) do equipamento. A quantidade de memória RAM, utilizada pelo RAM *disk,* foi de 14 MBytes, restando assim um total de 18 MBytes de memória RAM para a execução do sistema operacional e dos aplicativos.

Implementou-se o subsistema PacienteIP para o sistema operacional Linux em questão. Uma pequena modificação foi realizada no software do PacienteIP, para que os sinais não fossem mais adquiridos de um hardware de aquisição, e sim lidos de um arquivo armazenado na memória *flash*. Esse arquivo possui alguns minutos de amostras de quatro tipos de sinais vitais de um determinado paciente e foi obtido através do site http://www.physionet.org. O *software* PacienteIP realizava leituras periódicas de amostras desse arquivos, encapsulava-as em um pacote RTP e as transmitia pela rede.

O segundo paciente virtual formava-se por um PC desktop K6-2 com uma interface USB. A essa interface foi conectado um equipamento de aquisição, o qual amostrava sinais fornecidos por um emulador de sinais bioelétricos. As amostras eram lidas da interface USB, encapsuladas em pacotes RTP e transmitidas pela rede.

Os dois pacientes virtuais transmitiam seus pacotes RTP para o endereço multicast 224.0.0.1, onde quatro outros computadores conectados a rede local recebiam esses sinais, como pode ser observado na Figura 8.1.

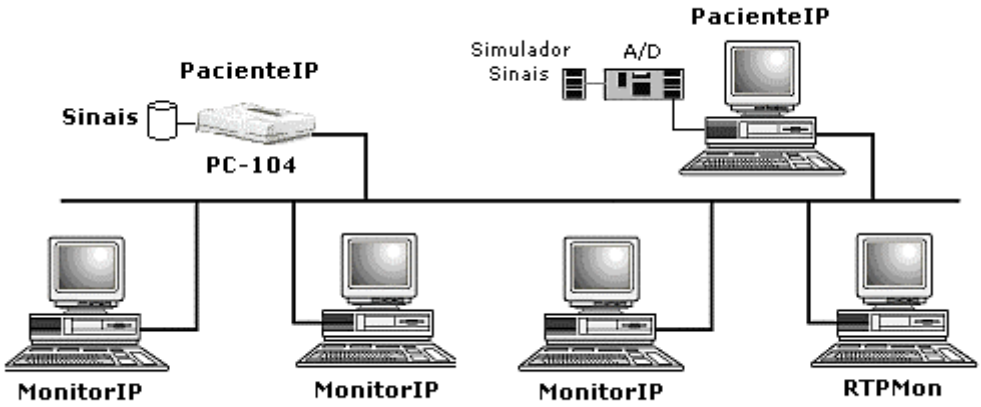

Figura 8.1 Cenário 1 – Rede Local

Três desses computadores executavam o *software* MonitorIP, com o qual podia-se monitorar os sinais vitais dos dois pacientes em tempo-real. Em um outro computador, executava-se o software RTPMonitor, com o qual podia-se avaliar a qualidade da transmissão e gerar as estatísticas necessárias para os testes.

8.2.1 Resultados Cenário 1

O tipo de endereçamento utilizado neste cenário foi o endereçamento *multicasting*. Dessa forma, um pacote RTP contendo sinais vitais de um paciente só precisa ser transmitido uma única vez, economizando assim largura de banda, pois não existe a necessidade de se transmitir o mesmo pacote RTP para cada um dos computadores que estiverem monitorando esse paciente.

A largura de banda necessária para transmitir um único sinal é de 5,8 Kbps (250 amostras/s * 148 *bytes* / 50 amostras * 8 *bits*). Como está se monitorando quatro sinais por paciente, a largura de banda utilizada por paciente é de 23,12 Kbps. E portanto, a largura de banda utilizada no monitoramento desses dois pacientes virtuais é de 46,25 Kbps. Como a máxima largura de banda disponível para a rede local do laboratório é de 10 Mbps, está se ocupando apenas 0,45% (46,25 Kbps/10 Mbps) do total.

A Figura 8.2 apresenta a tela do *software* RTPMonitor, com o qual monitora-se a transmissão dos sinais na rede local para um determinado paciente. Escolhe-se o paciente a ser monitorado na caixa de seleção no canto superior direito da tela. Logo abaixo, no canto direito, apresenta-se os resultados da transmissão de cada sinal desse paciente. Como se observa, para o primeiro sinal foram transmitidos 599

pacotes e nenhum pacote foi perdido, assim a taxa de perda foi de 0%, o que representou um ótimo resultado. Esse resultado já era esperado, uma vez que a ocupação da largura de banda da rede local é de apenas 0,45% e esses testes foram realizados sem gerar tráfego de outros dados, ou seja a largura de banda estava praticamente toda disponível para o monitoramento dos sinais.

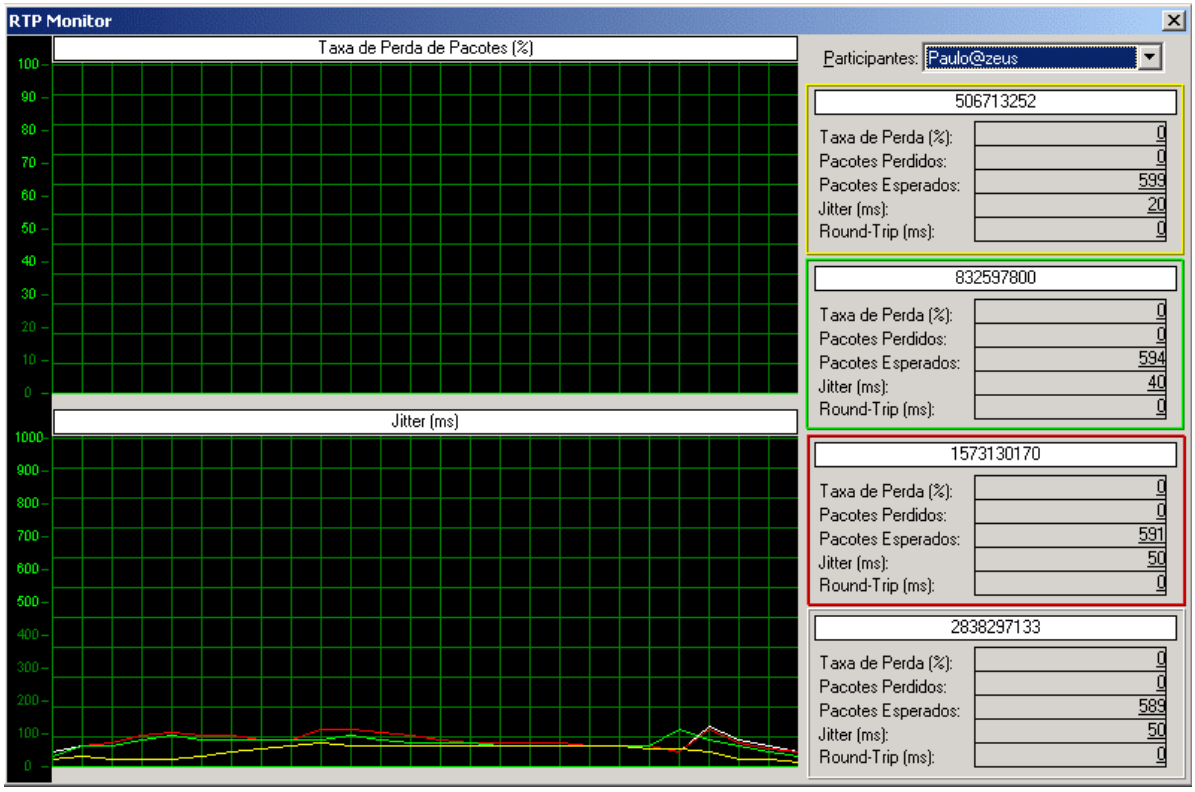

Figura 8.2 Tela do RTPMonitor – Rede local sem tráfego

Na tela mostrada na Figura 8.2, ainda pode-se observar dois gráficos. O primeiro (no topo da tela), mostra a taxa de perda de pacotes ao longo da transmissão. Como pode-se observar, a taxa de perda permaneceu em 0% durante toda a transmissão. O segundo gráfico (na parte inferior da tela), apresenta o comportamento do *jitter* durante toda a transmissão. O *jitter* mede as variações no tempo, que cada pacote leva para chegar ao destino. De acordo com o congestionamento que ocorrer na rede, essas variações podem ser grandes. Observou-se que tais variações no *jitter* foram pequenas, o que demonstra que a rede não estava congestionada. O *jitter* mede o congestionamento transiente, enquanto que a taxa de perda mede o congestionamento persistente. À medida do *jitter* pode indicar congestionamento antes de ocorrer perda de pacotes.

Em um determinado instante, gerou-se tráfego na rede, realizando-se

download de arquivos (FTP), enviando e recebendo *e-mails* e navegando por páginas da *web*. A Figura 8.3 apresenta os resultados da transmissão dos sinais, no instante em que foi gerado esse congestionamento. Pode-se observar do segundo gráfico (na parte inferior da tela), que as variações das medidas do *jitter* aumentaram. O valor de pico (valor máximo) do *jitter*, no teste anterior sem tráfego na rede, foi de 150 ms e agora com o tráfego passou para 300 ms, um aumento de 100%. Esse resultado já era esperado, pois esse aumento na variação do *jitter* representa o aumento do congestionamento da rede. Mas, mesmo com esse aumento, pode-se observar nos dados da transmissão de cada sinal, que nenhum deles perdeu qualquer pacote.

O MonitorIP calcula constantemente o *jitter* da rede e determina o tamanho do *buffer* necessário. Uma vez que a quantidade de *jitter* é tipicamente não muito alta e portanto somente uma pequena quantidade de pacotes precisa permanecer no *buffer*, esta técnica apresentou bons resultados. Como cada pacote representa 200 ms do sinal, o tamanho do *buffer* necessário para esse teste passou de 1 pacote sem tráfego, para 2 pacotes com tráfego.

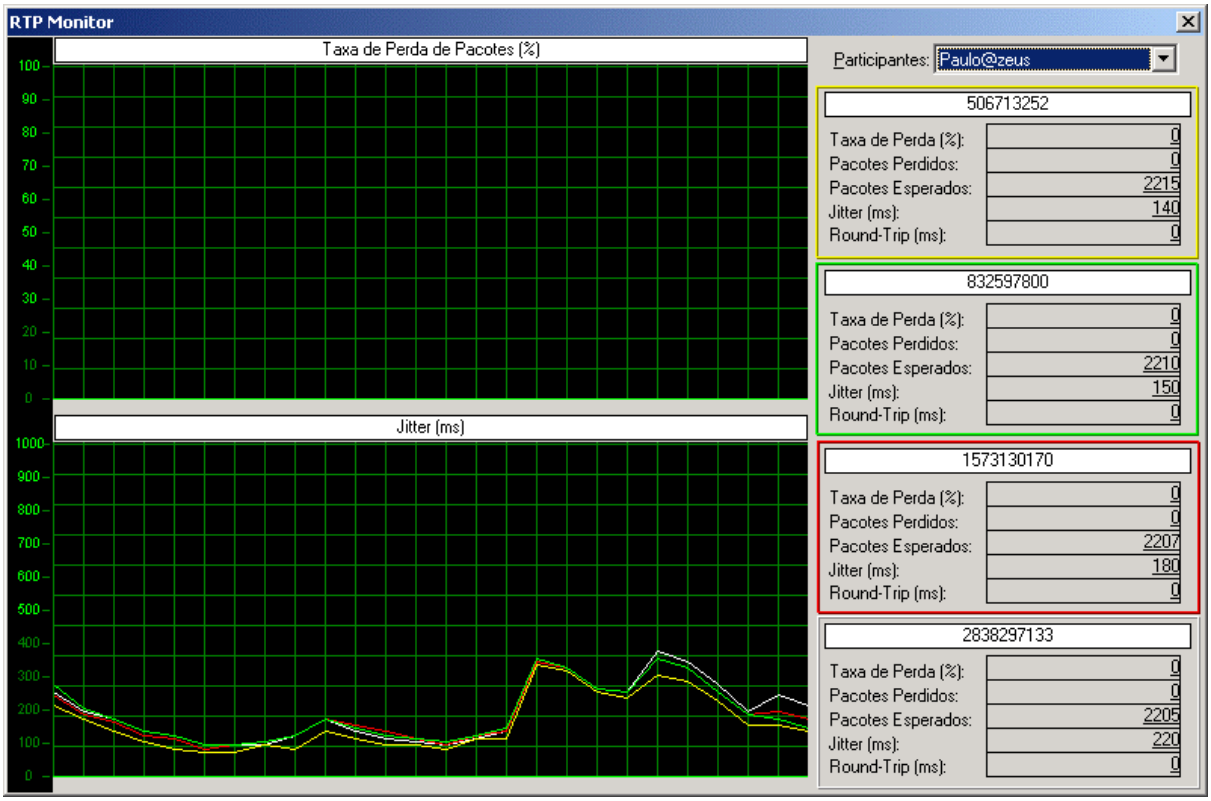

Figura 8.3 Tela do RTPMonitor – Rede local com tráfego

8.3 Cenário 2 – Conexão Dial-Up

Para a elaboração do cenário 2 utilizou-se o paciente virtual (com arquivo de sinais – PC104) do cenário anterior, colocando-se no lugar da placa de rede ethernet um *modem* de 56Kbps. A esse *modem* foi conectado uma linha discada.

Configurou-se o Linux para realizar uma conexão dial-up com o Núcleo de Processamento de Dados (NPD) da UFSC. Uma vez conectado na rede do NPD, o paciente virtual enviava pacotes RTP com os sinais vitais, para um computador localizado dentro do laboratório do Instituto de Engenharia Biomédica. O endereço de destino desses pacotes RTP, utilizado pelo software PacienteIP, foi um endereço *unicast*, ou seja, identificava apenas um determinado computador do laboratório, dessa forma apenas esse computador podia monitorar os sinais do paciente virtual. Não foi possível utilizar-se um endereçamento *multicasting*, o qual permitiria que qualquer computador do laboratório monitorasse o paciente, porque o paciente virtual conectava-se na rede do NPD e não à rede local do laboratório. Pois, os roteadores, que roteavam os pacotes até a rede local do laboratório, não estavam configurados para rotear pacotes com endereçamento *multicasting*. O diagrama da rede pode ser observado na Figura 8.4. Mediu-se a qualidade da transmissão dos pacotes RTP, executando-se o software RTPMonitor nesse mesmo computador do laboratório.

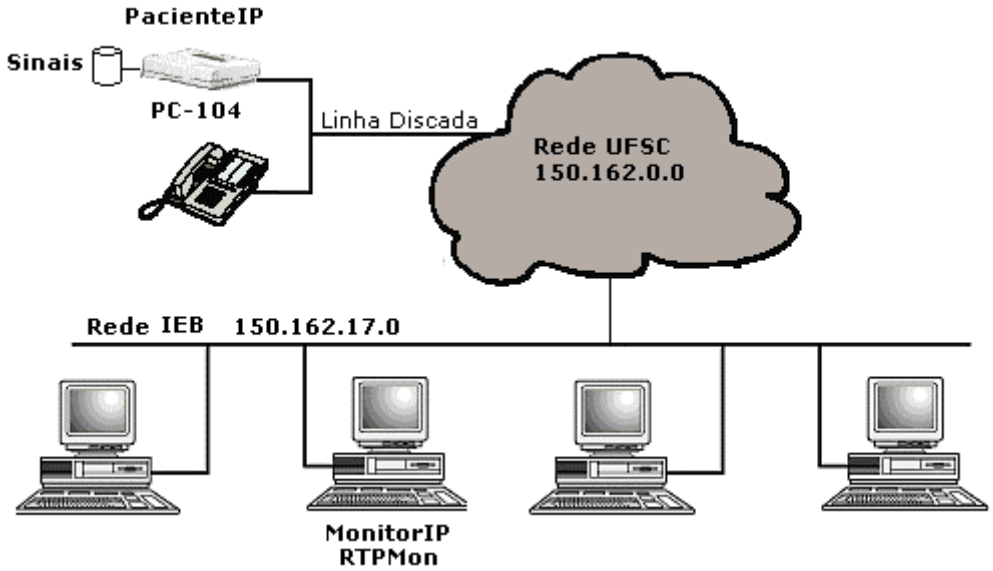

Figura 8.4 Cenário 2 – Conexão Dial-Up

8.3.1 Resultados Cenário 2

A largura de banda necessária para transmitir os quatro sinais do paciente virtual é de 23,12 Kbps. Como a máxima largura de banda disponível para a conexão dial-up é de 56 Kbps, ocupou-se então 41,29% (23,12 Kbps/56 Kbps) do total para a transmissão dos sinais vitais.

A Figura 8.5 apresenta a tela do *software* RTPMonitor, com o qual monitorou-se a transmissão dos sinais através dessa conexão dial-up. No canto direito, apresenta-se os resultados da transmissão de cada sinal desse paciente. Como se observa, para o primeiro sinal foram transmitidos 832 pacotes e 2 pacotes foram perdidos, assim a taxa de perda foi de 0,24%, o que representou um bom resultado.

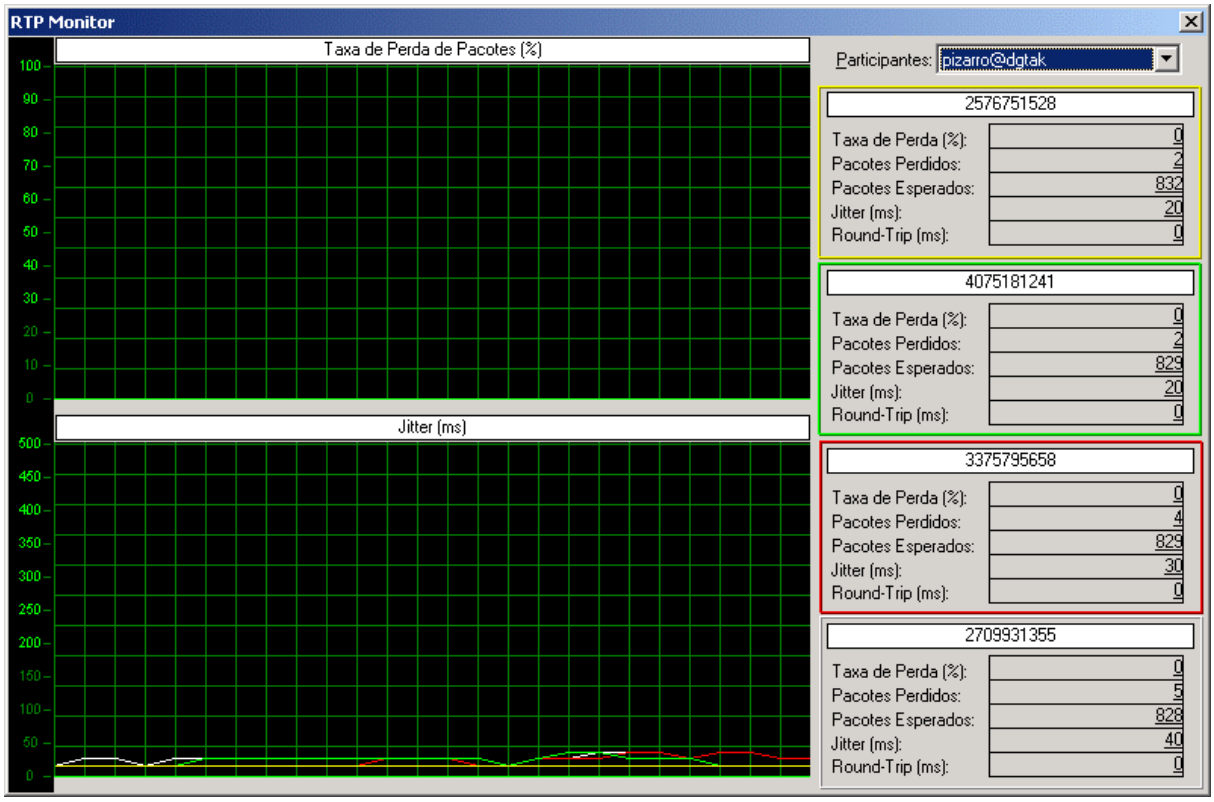

Figura 8.5 Tela do RTPMonitor – Conexão Dial-Up sem tráfego.

Na tela mostrada na Figura 8.5, ainda pode-se observar dois gráficos. O primeiro (no topo da tela), mostra a taxa de perda de pacotes ao longo da transmissão. Como pode-se observar, a taxa de perda permaneceu menor que 1% durante toda a transmissão. O segundo gráfico (na parte inferior da tela), apresenta o comportamento do *jitter* durante a transmissão. Observou-se, que as variações no *jitter* foram pequenas, o que demonstra que a rede não estava congestionada.

Em um determinado instante, gerou-se tráfego na rede, realizando-se

download de arquivos (FTP), enviando e recebendo *e-mails* e navegando por páginas da *web*. A Figura 8.6 apresenta os resultados da transmissão dos sinais, no instante desse congestionamento. Pode-se observar do segundo gráfico (na parte inferior da tela), que as variações das medidas do *jitter* aumentaram consideravelmente. O valor de pico (valor máximo) do *jitter*, no teste anterior sem tráfego na rede, foi de 50 ms e com o tráfego passou para 600 ms, um aumento de 1.200%. Esse aumento na variação do *jitter*, sinalizava um aumento signifivativo do congestionamento da rede, de tal forma que alguns pacotes foram perdidos. Pode-se observar nos dados do sinal 4, que 62 pacotes foram perdidos de um total de 3048 transmitidos, representando uma taxa de perda de 2,03%. Mesmo com essa taxa de perda, o monitoramento na tela do MonitorIP foi muito pouco afetado, não prejudicando a determinação do estado do paciente.

Como cada pacote representa 200 ms do sinal, o tamanho do *buffer* necessário para esse teste passou de 1 pacote sem tráfego, para 3 pacotes com tráfego.

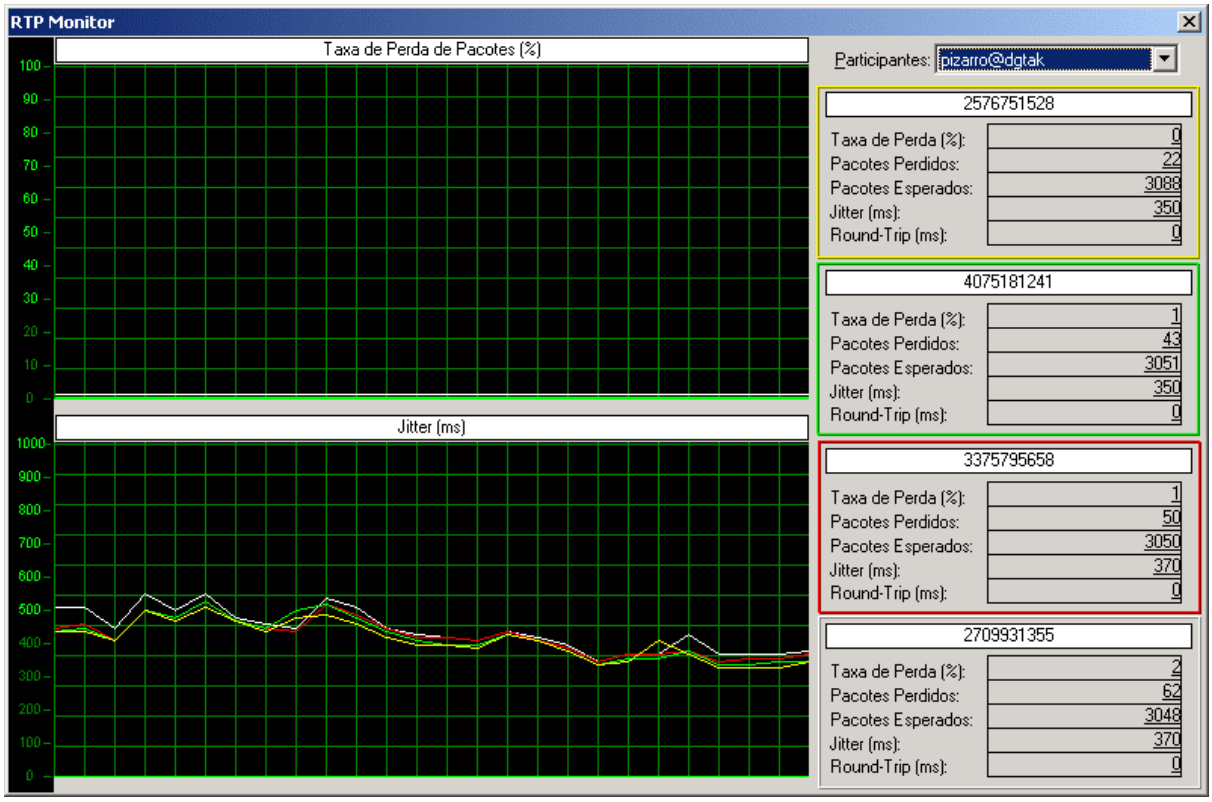

Figura 8.6 Tela do RTPMonitor – Conexão Dial-Up com tráfego.

8.4 Resultados Gerais

Utilizou-se uma freqüência de amostragem de 250Hz para a aquisição dos

sinais e cada pacote enviado transportava 50 amostras, o que equivalia a 200ms do sinal amostrado. O número de amostras por pacote não pode ser muito grande para que não afete a visualização do sinal pelo médico no caso de perda do pacote. A resolução do A/D utilizado na aquisição foi de 16 *bits*, assim o tamanho dos dados referentes às amostras foi de 100 *bytes*. Nos testes realizados, foi monitorado o sinal de ECG capturado através de um emulador de ECG. Cada pacote (200ms) corresponde em torno de 1/5 do período do sinal de ECG. Para diminuir o efeito da perda de um pacote poderia-se diminuir o número de amostras por pacote para 5, o que equivaleria a 20ms do sinal ou 1/50 do período desse sinal e dessa forma poderia-se pensar que o médico perderia pouca informação a respeito do sinal, mas isso não ocorre. O problema é que para a transmissão de cada pacote, tem-se um *overhead* causado pelas informações nos cabeçalhos dos protocolos envolvidos.

Primeiramente tem-se o cabeçalho do protocolo RTP, o qual possui um tamanho mínimo de 20 *bytes*. Na camada de transporte temos o protocolo UDP, o qual possui um cabeçalho de 8 *bytes*. E na camada de rede temos o protocolo IP com um tamanho mínimo de 20 *bytes* para seu cabeçalho, totalizando assim um adicional de 48 *bytes* por pacote transmitido. Dessa forma o tamanho mínimo do datagrama necessário para transmitirmos uma única amostra (2 *bytes*), que seria o caso ideal para diminuir o efeito da perda de um pacote, seria de 50 *bytes* (2 *bytes* da amostra + 20 *bytes* do RTP + 8 *bytes* do UDP + 20 *bytes* do IP). Ou seja, um *overhead* de 96%. Como a freqüência de amostragem é de 250Hz, a cada 4ms teríamos uma nova amostra a ser enviada, o que nos obriga a ter uma largura de banda de aproximadamente 100 Kbps (250 amostras/s * 50 *bytes* / 1 amostra * 8 *bits*). Para diminuir o *overhead* causado pelos cabeçalhos dos protocolos e com isso ter um melhor aproveitamento da largura de banda, devemos aumentar o número de amostras por pacote. Poderia-se definir que cada pacote transportaria 500 amostras, o que resultaria em um tamanho de 1048 *bytes* por datagrama e um *overhead* de 4,58%. Como cada amostra é adquirida a cada 4ms e o pacote possui 500 amostras, então cada pacote equivale a 2s do sinal amostrado e a largura de banda necessária seria de aproximadamente 4 Kbps (250 amostras/s * 1048 *bytes* / 500 amostras * 8 *bits*), tendo-se assim um melhor aproveitamento da largura de banda. O problema é que a perda de um pacote corresponde a 2s do sinal monitorado o que afeta muito a interatividade do sistema. Qualquer alteração no sinal só seria verificada 2s depois mais o tempo de transmissão e mais o tempo do *buffer* para compensar o efeito do *jitter*. Com os testes

realizados chegamos a conclusão que 50 amostras por pacote seria o melhor custo/beneficio para o nosso sistema. Assim a largura necessária por sinal monitorado seria de aproximadamente 5,8 Kbps (250 amostras/s * 148 *bytes* / 50 amostras * 8 *bits*) e o *overhead* resultante seria de 32,43%. Cada pacote corresponde a 200ms do sinal amostrado, o qual mantém ainda uma boa interatividade do sistema.

Na Tabela 7.2, pode-se observar um comparativo do número de amostras por pacote, com a quantidade de tempo de sinal amostrado correspondente a cada pacote, com a largura de banda necessária e o *overhead* resultante.

9. DISCUSSÃO E CONCLUSÕES

Uma das chaves para a sobrevivência econômica das organizações da área da saúde será utilizar a tecnologia da informação como uma arma competitiva. Com o avanço da tecnologia de comunicações, novas possibilidades têm surgido para a medicina e, como foi abordado neste trabalho, o monitoramento remoto e em tempo-real de pacientes é uma delas. Pacientes podem ser monitorados, não apenas dentro do hospital (LAN), mas também através da Internet. Acomodados em suas casas, eles podem ser monitorados remotamente por médicos situados em hospitais ou em clínicas. Com isso, diminuem-se os riscos de infecções hospitalares, como também, os pacientes têm uma melhor recuperação por estarem perto de seus familiares.

9.1 Discussão

A seguir apresenta-se uma discussão das vantagens e das desvantagens das soluções adotadas no projeto do sistema MonitorIP.

A utilização do protocolo IP proporcionou uma maior flexibilidade em conectividade. Além da vantagem das redes IP serem largamente utilizadas e possuírem um baixo custo de implantação em relação a redes *frame-relay* e ATM. A desvantagem causada pela escolha do protocolo IP, foi a deste não oferecer recursos para a garantia de QoS de aplicações de tempo-real. Outros protocolos como *frame-relay* e ATM oferecem melhor suporte, porém não oferecem flexibilidade em conectividade e os custos de implantação dessas redes são bem mais elevados do que as redes IP. A solução para resolver esta desvantagem será a utilização de protocolos como o RSVP (*Resource Reservation Protocol*), para alocação de recursos em uma rede IP e, dessa forma, garantir a QoS. Porém esses protocolos ainda não são largamente utilizados, principalmente na Internet, limitando sua utilização em uma estrutura de rede proprietária.

Outra opção de projeto foi a utilização do protocolo UDP como protocolo de transporte. Esse protocolo é simplesmente uma extensão do protocolo IP, com nenhuma complexidade adicional, e não realiza a retransmissão de pacotes perdidos (menor *overhead* que o protocolo TCP). Outra vantagem da sua utilização é a de suportar recursos de *multicasting*, economizando assim largura de banda no monitoramento de um paciente por mais de um terminal.

Uma vantagem conquistada pela utilização do protocolo RTP/RTCP foi a de

fornecer informações necessárias para o monitoramento em tempo-real, além de criar mecanismo para sincronização entre os diversos sinais, possibilitar o controle de fluxo e identificação de congestionamentos. Porém, não existe um esquema de gerenciamento da largura de banda ao se detectar um congestionamento na rede. Uma solução seria alterar dinamicamente a natureza dos dados transmitidos, por exemplo, diminuindo-se a frequência de amostragem do sinal ou deixando-se de transmitir um determinado sinal não tão prioritário em relação a um outro.

Suprimiu-se a necessidade de compactação das amostras durante a transmissão do sinal, pois a largura de banda necessária apresentava-se bastante reduzida em relação ao que as redes IP oferecem atualmente. Dessa forma diminui-se a complexidade do hardware necessário, não existindo a necessidade de CODECs (DSPs), além de diminuir também o atraso total, pois os algoritmos de compactação utilizam técnicas de *lookahead*. Esses algoritmos precisam manter um buffer com alguns quadros de amostras para realizarem a compactação ao invés de transmití-los imediatamente.

A opção de projeto no qual cada pacote RTP transporta somente os dados de um único sinal, melhorou a escalabilidade do sistema. Facilmente pode–se escalar o sistema para outros sinais. Bastando para isso criar uma nova sessão RTP para cada sinal adicionado. Não existe a necessidade de se alterar o protocolo e nem a forma como os dados são armazenados dentro do pacote RTP.

Outra vantagem foi a flexibilidade do sistema em monitorar apenas os sinais que forem de interesse, proporcionada pelo fato de que em cada sessão RTP, transmite-se apenas as amostras de um determinado sinal. Dessa forma, pode-se desenvolver aplicações que analisem ou apresentem apenas determinados sinais, não havendo a necessidade de criarem sessões para os outros sinais que não sejam de seu interesse, mesmo que o sistema PacienteIP conectado ao paciente, esteja transmitindo todos os sinais do paciente.

Observou-se a falta de um esquema de autenticação de usuários. Como opção de projeto, qualquer usuário tem o direito de monitorar quaisquer paciente.

A falta de privacidade do paciente, que tem seus sinais sendo transmitidos através da Internet aberta, também foi prejudicada pela ausência de um esquema de segurança dos dados transmitidos pela rede. Poderia-se utilizar algum algoritmo de criptografia no nível do RTP. O RFC 1889 apresenta uma maneira de se fazer isso se usando a criptografia DES/CBC (*Data Encryption Standard/Cipher Block Chaining*).

Não existe nenhum esquema para diminuição da perda de pacotes. Uma forma seria utilizar a redundância dos dados enviados. Assim, cada pacote RTP estaria transportando não apenas um quadro referente a um segmento do sinal amostrado, mas sim mais de um quadro. O pacote RTP seguinte possuiria um quadro novo e mais alguns do pacote anterior. Minimiza-se assim o efeito da perda de pacotes, porém aumenta-se a largura de banda necessária.

A falta de um esquema de sinalização para o início e término da transmissão dos sinais vitais de um paciente, compromete a largura de banda da rede. Atualmente todos os sistemas PacienteIP em execução ficam interruptamente transmitindo os sinais do paciente ao qual ele está conectado, mesmo que nenhum sistema MonitorIP esteja monitorando tal paciente. Dessa forma, o sistema não é escalável no número de pacientes que se pode monitorar, pois rapidamente a largura de banda estaria saturada. Uma solução seria a utilização de protocolos como o SIP (*Session Initiation Protocol*) para gerenciar o início e o término de uma sessão RTP, transmitindo-se assim apenas os sinais que estão sendo monitorados naquele instante e, portanto, obtendo-se uma melhor utilização da largura de banda da rede.

9.2 Conclusões

Durante uma telecirurgia, os aspectos de tempo-real são muito importantes, pois se trata do monitoramento de pacientes que estão sobre intervenção médica e, caso algum de seus sinais vitais sofra alguma alteração, o cirurgião precisa receber essa sinalização o mais rápido possível. Dessa forma, em uma telecirurgia, o atraso máximo do envio dos sinais vitais deve ser tão pequeno quanto possível.

Quando o monitoramento é realizado através da rede local do hospital, onde existem recursos de largura de banda mais do que suficientes, o sistema MonitorIP pode ser utilizado com muito pouco ou sem nenhum problema. Atualmente, o principal problema para a utilização do MonitorIP através da Internet, é satisfazer as garantias de QoS que o sistema exige. Quando protocolos suportando QoS, como o RSVP (*Resource Reservation Protocol*), forem usados em larga escala pela Internet, certamente fará com que sistemas como o MonitorIP tornem-se mais populares na área da saúde, possibilitando aos médicos a realização de telecirurgias com a qualidade que se faz necessária.

Com a rede de telefonia móvel instalada nos grandes centros, torna-se possível através da tecnologia GSM (*Global System for Mobile Communications*) o

monitoramento de pacientes localizados dentro de ambulâncias a caminho do hospital, por médicos localizados no hospital, os quais podem passar instruções para os primeiros atendimentos ao paciente.

9.3 Trabalhos Futuros

A largura de banda necessária para a transmissão de todos os sinais bioelétricos amostrados pelo monitor de multi-parâmetros é da ordem de 24 Kbps. Como trabalho futuro, algumas técnicas de compressão [1][2][3][4][5] poderiam ser utilizadas para reduzir essa quantidade.

Os quatro sinais monitorados, como apresentado no trabalho, utilizavam uma freqüência de amostragem de 250 Hz. Como trabalho futuro, poder-se-ia criar um perfil para a transmissão dos sinais vitais. Nesse perfil, definiria-se o tipo de sinal que o pacote RTP estaria transportando. Para tal definição utilizaria-se o campo PT (*payload type*) do cabeçalho do protocolo RTP. Com essa identificação, pode-se explorar melhor as propriedades de cada sinal. Como por exemplo, a utilização de uma freqüência de amostragem e algoritmos de compressão (CODECs) específicos para o sinal a ser monitorado. Com isso reduziria-se a largura de banda necessária para o monitoramento dos diversos sinais vitais.

Uma outra proposta para trabalhos futuros seria a utilização do protocolo SIP (*Session Initiation Protocol*), como alternativa aos pacotes RTCP do tipo SDES, para a identificação dos pacientes que estariam disponíveis para o monitoramento e para a configuração automática das sessões RTP. Dessa forma, não seria preciso configurar manualmente o endereço de destino no subsistema PacienteIP e nem no subsistema MonitorIP.

Uma última proposta refere-se a confidencialidade dos sinais transportados, ou seja, somente um determinado receptor poderia monitorar os sinais de um paciente. Um estudo de como se criptografaria os dados dos pacotes RTP e para quais dados seria necessária se fazer essa confidencialidade, necessária principalmente quando transmitindo tais informações pela Internet.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] KYOSO, M., UCHIYAMA, A., **ECG Data Reduction Meted for Medical Telemetry System**, Congresso Internacional de Eng. Biomédica. Chicago, 2000.

[2] COX, J.R., NOLLE F.M., FOZZARD H.A., OLIVER G.C., **AZTEC, a preprocessing program for real-time ECG rhythm analysis**. IEEE Trans. Biomed. Eng. vol. 15, pp. 128-129, Apr. 1968.

[3] ISHIJIMA, M., SHIN S.B., HOSTETTER G.H., SKLANSKY J., **Scan along polygonal approximation for data compression of electrograms**. IEEE Trans. Biomed. Eng. vol. 30, pp. 723-729, Nov. 1983.

[4] ABENSTEIN, J.P., TOMPKINS, W.J., **A new data-reduction algorithm for real-time ECG analysis.** IEEE Trans. Biomed. Eng. vol. 29, pp. 43-48, Jan. 1982.

[5] STEWART, D., DOWER, G.E., SURANYI, O., **An ECG compression code. Journal of Electrocardiogram**, vol. 6, pp. 175-176, 1973.

[6] KOLIVER, C., FARINES, J.M., **Um Controlador Nebuloso para Adaptação de QoS**. In: 19o. Simpósio Brasileiro de Redes de Computadores. Florianópolis, Santa Catarina, Maio 2001.

[7] BEZDEK, J. C. **Pattern Recognition with Fuzzy Objective Function Algorithms**. Plenum Press, 1981.

[8] BOGATINOVSKI, M., TRAJKOVSKI, G., SPASENOVSKI, B. **Fuzzy Controller for Video Conference Traffic in B-ISDN**. In IEEE 6th International Workshop on Computer-Aided Modeling, Analysis and Design of Communication Links and Networks (CAMAD'98) (São Paulo, Brasil, 1998), pp. 55-59.

[9] BUSSE, I., DEFFNER, B., SCHULZRINNE, II. **Dynamic QoS of Multimedia Applications Based on RTP**. In 1st International Workshop on High Speed Networks and Open Distributed Platforms (St. Petesburg, Russia, May 1995).

[10] GECSEI, J. **Adaptation in Distributed Multimedia System**. IEEE Multimedia (April-June 1997), 58-95.

[11] GHINEA, G., THOMAS, J. **QoS Impact on User Perception and Understanding of Multimedia Vídeo Clips**. In 6st ACM International Conference on Multimedia'98 (Bristol, England, September 1998), pp. 49-54.

[12] GUYTON, Arthur C.; HALL, John E. **Textbook of Medical Physiology**, 9th ed. USA : W. B. Sauders Company, 1996.

[13] MARQUES, Jefferson Luiz Brum. **Eletrocardiografia de Alta Resolução: Metodologia e Aplicação Clínica**. Florianópolis, 1996. Universidade Federal de Santa Catarina.

[14] BARBOSA, Paulo R.; PERREIRA, Wagner C. A.; NADAL, Jurandir. **Análise de Potenciais Tardios Ventriculares Baseada em Intervalos RR Modais**. In: III Fórum Nacional de Ciência e Tecnologia em Saúde. Campos de Jordão – SP, Anais p.143, 1996. v.1.

[15] SPECIALSKI, Elizabeth. **Arquiteturas de Redes de Computadores**.Curso de Pós-graduação em Ciência da Computação. Florianópolis – SC, 2000. Universidade Federal de Santa Catarina.

[16] STEVENS, W. Richard. **TCP/IP Illustrated – The Protocols**. Vol. 1, Addison-Wesley, 1994.

[17] KAMIENSKI, C. A.; SADOK, D. H. F. **Qualidade de Serviço na Internet**. Mini-curso apresentado no XVIII Simpósio Brasileiro de Redes de Computadores. Maio de 2000. Disponível em http://www.cin.ufpe.br/~cak/publications.html

[18] MARTINS, Joberto. **Redes Corporativas MultiServiço – Caracterização das Aplicações e Parâmetros Básicos de Operação**. In: http://www.jsmnet.com/slides/AnaliseRequisitos/index.htm.

[19] McCABE, James. **Practical Computer Network Analysis and Design**. Morgan Kaufmann Series in Networking, 1998.

[20] TANEMBAUM, Andrew. **Computer Networks**. 3rd edition, Prentice-Hall, 1996.

[21] HEIN, Mathias; GRIFFITHS, David. **Switching Technology in the Local Network – From LAN to Switched LAN to Virtual LAN**, Thomson Computer Press, 1997.

[22] H. Schulzrinne, S. Casner, R. Frederick, V. Jacobson, RFC 1889: **RTP: A Transport Protocol for Real-Time Applications**, 01/25/1996.

[23] PIZARRO, Paulo; LOPES, Cássio; MORAES, Raimes; AZEVEDO, Fernando Mendes de; MARQUES, Jefferson Luiz Brum. **Monitoramento Remoto de Sinais Bioelétricos**.In: II Congreso Latinoamericano de Ingeniería Biomédica. Habana, Cuba, May., 2001, v.1.

[24] H. Schulzrinne, RFC 1890: **RTP Profile for Áudio and Vídeo Conferences with Minimal Control**; 01/25/1996.

[25] WEBSTER, John G. **Medical Instrumentation Application and Design**. 2 ed., Boston : Houghton Mifflin, 1992.

[26] COIMBRA, Alexandre José Fernández. **Análise Computadorizada de Sinais Bioelétricos**. Florianópolis, 1994. Dissertação (Mestrado em Engenharia Elétrica) Centro Tecnológico, Universidade Federal de Santa Catarina.

[27] RODRIGUES, Marco Aurélio Benedetti. **Desenvolvimento de um Instrumento Virtual para Aquisição e Análise de Sinais Bioelétricos**. Florianópolis, 1997. Dissertação (Mestrado em Engenharia Elétrica) Centro Tecnológico, Universidade Federal de Santa Catarina.

[28] KANDEL, Eric R.; SCHWARTZ, James H.; JESSEL, Thomasm. **Principles of Neural Science**. 3rd ed. London : Prentice – Hall International Inc., 1991.

[29] TYNER, Fay S.; KNOTT, John R.; MAYER Jr., W. Brem. **Fundamental of EEG Technology**. s.ed. New York : Raven Press, 1989. V.1: Basic Concepts and Methods.

[30] NORMANN, **Richard A. Principles of Bioinstrumentation and Design**. New York : Wiley, 1988.

[31] ROCHA, Paulo Henrique; ZANCHIN, Carlos Inácio; LIMA, Walter Celso de. **Detorflex – Amplificador de Biopotencial Eletromiográfico Flexível**. In: III Fórum Nacional de Ciência e tecnologia em Saúde. Campos do Jordão – SP, Anais p.21, 1996. v. 1.

[32] KOHN, André F.; MIQUELETI, Sandro **A. Sistema de Captação de Eletromiografia de Superfície com Elétrodo Múltiplo**. In: III Fórum Nacional de Ciência e Tecnologia em Saúde. Campos do Jordão – SP, Anais p.523, 1996, v. 2.

[33] BRADEN, R., RFC 1122, **Requirements for Internet *Hosts*--Communication Layers**, Internet Engineering Task Force, 1989.

[34] COMER, D., **Internetworking with TCP/IP Volume I: Principles, Protocols, and Architecture**, Prentice Hall, Englewood Cliffs, New Jersey, 1991.

[35] COMER, D. and Stevens, D., **Internetworking with TCP/IP Volume II: Design, Implementation, and Internals**, Prentice Hall, Englewood Cliffs, New Jersey, 1991.

[36] COMER, D. and Stevens, D., **Internetworking with TCP/IP Volume III: Client-Server Programming and Applications**, Prentice Hall, Englewood Cliffs, New Jersey, 1991.

[37] LEFFLER, S. et al., **An Advanced 4.3 BSD Interprocess Communication Tutorial**.

[38] STEVENS, W.R., **Unix Network Programming**, Prentice Hall, Englewood Cliffs, New Jersey, 1990.

[39] WRIGHT, G.R. and STEVENS, W.R., **TCP/IP Illustrated Volume 2: The Implementation**, Addison-Wesley., Massachusetts, 1995.

[40] BONNER, P., **Network Programming with Windows Sockets**, ISBN: 0-13-230152-0, Prentice Hall, Englewood Cliffs, New Jersey, 1995.

[41] DUMAS, A., **Programming Windows Sockets**, ISBN: 0-672-30594-1, Sams Publishing, Indianapolis, Indiana, 1995.

[42] QUINN, B. and Shute, D., **Windows Sockets Network Programming**, ISBN: 0-201-63372-8, Addison-Wesley Publishing Company, Reading, Massachusetts, 1995.

[43] ROBERTS, D., **Developing for the Internet with Winsock**, ISBN 1-883577-42-X, Coriolis Group Books, 1995.

[44] STONES R., MATTHEW N., **Beginning Linux Programming**, Wrox Press., 2nd ed., 1999.

[45] RUBINI, A., **Linux Device Drivers**, O'Reilly, 1st ed., 1999.