Fabio David

Ferramentas de Monitoração Ativa e Passiva para Avaliação da Qualidade de Redes VoIP

Orientador**:**
Prof. Paulo Henrique de Aguiar Rodrigues, Ph.D.

Universidade Federal do Rio de Janeiro - UFRJ
Instituto de Matemática - IM
Núcleo de Computação Eletrônica – NCE

Rio de Janeiro, Setembro de 2003

FERRAMENTAS DE MONITORAÇÃO ATIVA E PASSIVA PARA AVALIAÇÃO DA QUALIDADE DE REDES VOIP

Fabio David

Orientador: Paulo Henrique de Aguiar Rodrigues

DISSERTAÇÃO DE MESTRADO SUBMETIDA AO PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA DO INSTITUTO DE MATEMÁTICA / NÚCLEO DE COMPUTAÇÃO ELETRÔNICA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO – UFRJ, COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA OBTENÇÃO DO TÍTULO DE MESTRE EM CIÊNCIAS EM INFORMÁTICA.

Aprovada por:

_____
Prof. Paulo Henrique de Aguiar Rodrigues, Ph.D.

_____
Prof[a]. Luci Pirmez, D.Sc.

_____
Prof. José Ferreira de Rezende, Dr.

RIO DE JANEIRO, RJ - BRASIL
SETEMBRO 2003

FICHA CATALOGRÁFICA

David, Fabio.

Ferramentas de Monitoração Ativa e Passiva para Avaliação da Qualidade de Redes VoIP / Fabio David. - Rio de Janeiro, 2003.

xii, 111 f.: il.

Dissertação (Mestrado em Informática) – Universidade Federal do Rio de Janeiro - UFRJ, Instituto de Matemática – IM, Núcleo de Computação Eletrônica – NCE. Prograna de Pós-Graduação em Informática – IM/NCE, 2003.

Orientador: Paulo Henrique de Aguiar Rodrigues

1 Redes. 2. VoIP. 3. Monitoração. 4. Qualidade. I. Rodrigues, Paulo Henrique de Aguiar (Orient.). II. Universidade Federal do Rio de Janeiro. Instituto de Matemática / Núcleo de Computação Eletrônica. III. Título.

AOS MEUS FILHOS, PAULA E EDUARDO

AGRADECIMENTOS

Ao meu orientador Professor Paulo Aguiar, pelas valiosas contribuições e orientação para a execução deste trabalho.

Aos meus pais Hermann e Margot, sempre presentes e que me apóiam em todas as situações.

A minha irmã Silvia, pelo seu incentivo e interesse.

A minha esposa Teca, por ter assumido algumas das minhas obrigações durante o período de elaboração deste trabalho.

Ao meu amigo João Carlos Peixoto, que realizou sua pesquisa concomitantemente à minha e que sempre contribuiu com sugestões importantes.

Aos colegas do Laboratório VoIP do NCE, pela amizade e cooperação.

Aos amigos do NCE, que estimularam e desejaram o meu completo sucesso no decorrer deste trabalho.

Ao Núcleo de Computação Eletrônica, pelas excelentes condições de trabalho fornecidas para esta pesquisa.

RESUMO

FERRAMENTAS DE MONITORAÇÃO ATIVA E PASSIVA PARA AVALIAÇÃO DA QUALIDADE DE REDES VOIP

Fabio David

Orientador: Paulo Henrique de Aguiar Rodrigues

Resumo da Dissertação de Mestrado submetida ao Programa de Pós-graduação em Informática, Instituto de Matemática e Núcleo de Computação Eletrônica, da Universidade Federal do Rio de Janeiro – UFRJ, como parte dos requisitos necessários à obtenção do título de Mestre em Informática.

O serviço de Voz sobre IP tem despertado cada vez mais interesse, seja por motivos econômicos ou pela flexibilidade e facilidades inerentes desta tecnologia. Para se implementar VoIP com sucesso, é necessário que existam ferramentas que permitam uma monitoração da qualidade do serviço sendo oferecida, possibilitando que ações sejam tomadas antecipadamente, de forma pró-ativa. Neste contexto, duas arquiteturas de ferramentas de monitoração e avaliação de qualidade VoIP são apresentadas: uma atuando no modo ativo, onde o tráfego é gerado pelo próprio aplicativo, e outra no modo passivo, ou não-intrusivo, onde o tráfego de ligações reais é capturado para avaliação em tempo real.

Rio de Janeiro, RJ - BRASIL

Setembro 2003

ABSTRACT

ACTIVE AND NON-INTRUSIVE MONITORING TOOLS FOR QUALITY MEASUREMENT IN VOIP NETWORKS

Fabio David

Advisor: Paulo Henrique de Aguiar Rodrigues

Abstract da Dissertação de Mestrado submetida ao Programa de Pós-graduação em Informática, Instituto de Matemática e Núcleo de Computação Eletrônica, da Universidade Federal do Rio de Janeiro – UFRJ, como parte dos requisitos necessários à obtenção do título de Mestre em Informática.

Voice over IP is gaining more acceptance, due to economic reasons and its flexibility and ease of use. To efficiently deploy this technology, it is necessary to use tools that monitor and evaluate the quality of service being offered, making it possible to take proactive actions. In this context, two architectures for VoIP monitoring and performance assessment tools are presented: one that generates its own flows, and another non-intrusive tool, where actual VoIP traffic is captured for real-time evaluation.

Rio de Janeiro, RJ - BRASIL

September 2003

LISTAS DE SIGLAS E ABREVIATURAS

DLSR	*Delay Since Last Report*
DIFFSERV	*Differentiated Services*
ETSI	*European Telecommunications Standards Institute*
FEC	*Front End Clipping*
HOT	*Hold-Over Time*
IETF	*Internet Engineering Task Force*
ITU-T	*International Telecommunication Union Telecom Standardization*
INTSERV	*Integrated Services*
LSR	*Last Sender Report*
MCU	*Multiport Control Unit*
MIB	*Management Information Base*
MMUSIC	*Multiparty Multimedia Session Control*
MNB	*Measuring Normalizing Blocks*
MOS	*Mean Opinion Score*
NTIA	*National Telecommunications and Information Administration*
OWD	*One-Way Delay*
PAMS	*Perceptual Analysis Measurement System*
PESQ	*Perceptual Evaluation of Speech Quality*
PSQM	*Perceptual Speech Quality Measurement*
PSTN	*Public Switched Telephone Network*
QoS	*Quality of Service*
RAQMON	*Real-time Application Quality of Service Monitoring*
RAS	*Registration Admission Status*
RNP	*Rede Nacional de Ensino e Pesquisa*
RR	*Receiver Report*
RSVP	*Resource ReSerVation Protocol*
RTCP	*Real-time Control Protocol*
RTP	*Real-time Transport Protocol*
RTT	*Round-Trip Time*
SDES	*Source Description*
SIP	*Session Initiation Protocol*
SNMP	*Simple Network Management Protocol*
SR	*Sender Report*
TCP	*Transmission Control Protocol*
TDM	*Time-Division Multiplexing*
UDP	*User Datagram Protocol*
VAD	*Voice Activity Detection*
VOIP	*Voice Over IP*

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

Capítulo 1

Introdução

Nos últimos anos, tem havido um avanço na convergência entre as tecnologias de redes de comunicação de dados e de voz. No entanto, a utilização de fluxos multimídia e de tempo real na Internet tem sido um grande desafio. A tecnologia utilizada na Internet é baseada no conjunto de protocolos TCP/IP, que somente provê serviços de entrega de pacotes com o modelo *best-effort*, em sua forma mais convencional. Este modelo de serviço, como o próprio nome sugere, não oferece garantia de entrega de pacotes, nem o seqüenciamento dos mesmos, sendo possível inclusive o surgimento de pacotes duplicados. Além disto, não provê nenhuma facilidade para implementação de serviços baseados em tempo real. Apesar destas dificuldades inerentes à arquitetura adotada na Internet, serviços integrando a telefonia convencional com a rede de dados têm se tornado cada vez mais comuns.

A motivação não é puramente de ordem econômica. Diversos novos serviços, que antes não eram possíveis, passam a ser oferecidos utilizando as redes de telefonia e dados de forma integrada. Contudo, a telefonia convencional é uma tecnologia consolidada e extremamente estável e, portanto, confiável. É um desafio fazer com que os serviços de VoIP (*Voice over IP* ou Voz sobre IP) alcancem o mesmo patamar de qualidade e disponibilidade da telefonia convencional.

A arquitetura da telefonia convencional é baseada na tecnologia TDM (*Time-Division Multiplexing*), totalmente adequada para serviços de comunicação de voz. Tal arquitetura fornece uma latência limitada entre origem e destino, variação desta latência praticamente nula, perda extremamente baixa e uma taxa constante de largura de banda disponível para transmissão. Por outro lado, redes IP foram projetadas para comportar tráfego em rajadas, como transferência de arquivos e correio eletrônico, sem características de tempo real. Qualidade de Serviço, ou simplesmente QoS, tem sido alvo de muitas pesquisas, tanto por parte da comunidade acadêmica como das indústrias, por ser um fator fundamental para o sucesso de implementações de voz sobre

redes de dados. Propostas de implementações como IntServ (*Integrated Services*) [1] e DiffServ (*Differentiated Services*) [2] são discutidas desde meados da década de 90, mas ainda não são empregadas em larga escala até hoje. Estes serviços de QoS introduzem tratamento diferenciado para determinados fluxos da rede IP, reduzindo a latência, o jitter e a taxa de perda de pacotes, fatores críticos para tráfego em tempo real.

Uma vez implementado o serviço de Voz sobre IP, surge a inevitável comparação da qualidade do serviço oferecido com a do sistema de telefonia convencional. Torna-se necessária a existência de ferramentas que permitam uma avaliação quantitativa e comparativa entre as conexões de telefonia convencional e as de voz sobre IP. Essas ferramentas passam a ter um papel fundamental na gerência do serviço VoIP, permitindo não apenas a medição da qualidade do serviço mas também a determinação de pontos de falha ou congestionamento na rede, servindo como fonte de informações para tomada de decisão em intervenções pró-ativas visando resolver o problema, ou mesmo para dimensionamento e planejamento de capacidade da infra-estrutura de rede.

O processo desta avaliação de qualidade pode ter critérios subjetivos, como clareza da voz, volume ou nível de ruídos durante a ligação. É importante frisar que, em telefonia, a fidelidade da reprodução não é tão importante quanto a sua inteligibilidade, ou seja, a capacidade das extremidades se comunicarem de forma adequada e confortável [3, 4].

Diversos fatores influenciam diretamente na qualidade de voz obtida em uma conexão, independente da tecnologia sendo utilizada para efetuar as ligações. Podemos citar como fatores para medir a qualidade da voz [3, 4]:

- volume
- atraso
- eco
- clareza da voz
    - inteligibilidade
    - ruído
    - falhas ou interrupções
    - interferências

Em redes de Voz sobre IP, fatores adicionais, inexistentes na rede de voz convencional, passam a ter influência sobre a qualidade da voz obtida:

- variação do atraso
- clareza da voz
  - perda de pacotes
  - codificação / compressão / supressão de silêncio
  - largura de banda disponível

Alguns fatores externos, mesmo que não estejam diretamente relacionados à tecnologia sendo utilizada, podem gerar degradação na avaliação da qualidade de voz. Ruídos externos, tanto no momento da transmissão quanto no momento da recepção, prejudicam o entendimento das palavras e, por conseqüência, afetam a avaliação da conexão.

O momento da ocorrência das falhas de uma ligação também afeta a percepção do receptor. Falhas que ocorrem no início ou no meio de uma ligação são mais bem toleradas do que as ocorridas no final das mesmas. Este fato, conhecido como efeito memória, é atribuído ao fato do cérebro humano dar maior prioridade a informações mais recentes. Dessa forma, a avaliação de uma ligação tende a ser mais rigorosa caso ocorra uma maior degradação no final da mesma [13, 30]. Um algoritmo de avaliação, cuja estimativa da qualidade de voz seja mais precisa, deve levar em consideração não apenas o percentual de pacotes perdidos, mas também a sua distribuição ao longo do tempo da ligação.

Um interessante estudo realizado pelo laboratório de pesquisas da AT&T comprova o efeito memória [13], mostrando que a mente humana tende a valorizar fatos mais recentes em relação a fatos ocorridos anteriormente. Em outro estudo, feito nos laboratórios da France Telecom em [14], demonstra que melhorias nas condições da ligação não refletem em um aumento imediato da qualidade percebida pelos usuários.

Portanto, um algoritmo de avaliação, cuja estimativa da qualidade de voz seja mais precisa, deve levar em consideração não apenas o percentual de pacotes perdidos, mas também a sua distribuição ao longo do tempo da ligação.

1.1. Métodos para Avaliação da Qualidade de Voz

Com tantas variáveis, a avaliação de qualidade de voz é um processo complexo e subjetivo. Diversos métodos foram desenvolvidos para medir a qualidade da voz sob o ponto de vista do receptor. Alguns destes métodos utilizam uma avaliação subjetiva. Essa avaliação é realizada por um conjunto de usuários, que dão notas a alguns critérios pré-estabelecidos. A média obtida destas avaliações torna-se a métrica que representa a qualidade desta ligação. Outros métodos de avaliação da qualidade de voz calculam esta métrica de forma objetiva. A métrica é calculada a partir de parâmetros obtidos das próprias conexões como, por exemplo, perda de pacotes, latência, *jitter* médio, codificador utilizado e tamanho dos *buffers* de compensação de *jitter*, entre outros. Outra forma de se obter uma avaliação objetiva é através da comparação do sinal de voz da mensagem original com o sinal da mensagem recebida, após ter sofrido a degradação causada pelo processo de codificação e decodificação, e pela sua transmissão na rede.

Métodos subjetivos apresentam resultados mais reais, pois utilizam a interpretação e a avaliação humana como forma de se obter uma métrica representativa da qualidade. No entanto, a implementação é um processo caro e complexo. Este método é tipicamente usado para avaliar a performance de novos codificadores de áudio, ou na certificação de equipamentos que utilizem a tecnologia VoIP [20]. Métodos subjetivos não são adequados para análise de qualidade em tempo real de conexões de usuários em ambientes de produção, por questões de privacidade e escalabilidade.

O primeiro método a ser mais largamente utilizado para avaliar a qualidade de voz foi o *Mean Opinion Score* (MOS) [15], utilizando uma avaliação subjetiva. Nesse método, um grupo de pessoas escuta e avalia individualmente as ligações, gerando valores médios estatísticos que podem quantificar a qualidade da voz. As técnicas de como tais testes devem ser aplicados são descritas pela recomendação ITU P.800 [15].

À medida que novas tecnologias passam a ser utilizadas para a transmissão de voz, cresce a necessidade da criação de métodos de medição e aferição da qualidade de voz de forma automática, estimando como esta qualidade seria percebida por uma avaliação humana. Surgiram diversos modelos de avaliação objetiva como, por exemplo, PSQM (*Perceptual Speech Quality Measurement*)[16], PAMS *(Perceptual*

Analysis Measurement System)[17], PESQ (*Perceptual Evaluation of Speech Quality*)[18] e E-Model[19].

Diversos trabalhos [25, 26, 27] foram realizados visando validar a metodologia de cada modelo, apresentando como eles se relacionam com o MOS. É importante observar que, através dos resultados das comparações, foi constatado que um determinado modelo é mais apropriado para um certo contexto, mas não se pode afirmar que algum dos modelos seja melhor em todos os casos.

Alguns destes modelos são baseados na comparação direta do sinal da mensagem original com o da mensagem recebida, já degradada pelos fatores de sua própria transmissão. Para ferramentas de gerência em tempo real, estes modelos não são adequados, pois dependem do sinal original, conhecido como sinal de referência. Para tal tipo de gerência, é necessário um modelo que utilize somente fatores instantâneos da rede como, por exemplo, taxa de perda de pacotes, atrasos e *jitter*.

O E-Model foi originalmente proposto pelo ETSI (*European Telecommunications Standards Institute*), através do relatório técnico ETR250 [28], e depois transformado em padrão pelo ITU como G.107 [19]. O objetivo do modelo é determinar a qualidade de uma transmissão no esquema emissor-receptor (*mouth to ear*) através da análise de parâmetros coletados como atrasos e perdas.

Embora o E-model seja uma maneira conveniente de se calcular o MOS de forma não-intrusiva em redes de voz sobre IP, não leva em consideração na sua estimativa de qualidade o momento em que os fatores que causam a degradação ocorrem durante a chamada, nem a sua distribuição ao longo do tempo, e nem tampouco as características do comportamento humano, como por exemplo, o efeito memória[65].

Um novo modelo foi proposto por Clark [30] como extensão ao E-model, passando a levar em consideração os fatores que não são atendidos pelo modelo original, acarretando em uma melhor aproximação do MOS.

Desta forma, é muito importante que as informações sobre os fatores que afetam a qualidade durante as ligações sejam coletadas com o maior nível de detalhamento

possível, permitindo a análise não apenas dos seus valores, mas também da distribuição de ocorrência dos mesmos ao longo de cada chamada.

1.2. Ferramentas de Coleta de Dados para Estimativa da Qualidade

Independente do método de avaliação adotado, seja ele subjetivo ou objetivo, torna-se necessário coletar, através de ferramentas específicas, informações referentes às conexões para possibilitar a estimativa da qualidade das mesmas. Tais ferramentas de coleta podem ser classificadas como ativas ou passivas.

Ferramentas de monitoração ativa geram o seu próprio tráfego de voz, simulando as duas extremidades da conexão. Parâmetros como a perda de pacotes, a latência e o *jitter* são obtidos diretamente dos fluxos gerados. Tais ferramentas permitem ainda uma análise subjetiva de como a voz foi recebida, através do armazenamento da recepção da mesma durante a conexão simulada e sua posterior análise por um conjunto de avaliadores.

Ferramentas de monitoração passiva, também chamadas ferramentas não-intrusivas, analisam os fluxos de voz que trafegam pela rede, extraindo as informações para análise da qualidade. Esta monitoração é feita em pontos estratégicos da rede, coletando as informações a medida que elas passam pelo meio físico ou então atuando em pontos de acesso, onde os fluxos passam obrigatoriamente como *proxies* e *firewalls*, por exemplo. Uma vantagem deste tipo de ferramenta é não depender de clientes ou equipamentos específicos, podendo avaliar fluxos gerados por dispositivos de diferentes fabricantes. Além disto, esse tipo de ferramenta utiliza o tráfego de voz real como entrada de dados para a modelagem da qualidade.

Existem diversas ferramentas disponíveis comercialmente como, por exemplo, NetIQ Chariot [66], Agilent Voice Quality Tester (VQT) [67], Spirent Abacus2 [68], Artiza VoIP Analyzer [69], Avaya ExpertNet [70] e Telchemy VQmon [20]. A maioria dessas ferramentas é dependente de hardware específico, e atua de forma ativa.

1.3. Objetivo e Organização do Trabalho

O objetivo deste trabalho é apresentar duas ferramentas de monitoração de qualidade de voz, uma operando em modo ativo e outra em modo passivo.

A implementação dessas ferramentas torna-se essencial como apoio para o processo de implantação do projeto piloto VoIP da RNP (Rede Nacional de Pesquisas) [33]. Através dos dados coletados por tais ferramentas, pode-se avaliar o serviço VoIP de um modo geral, como também o impacto na qualidade do serviço referente a implementação de políticas de QoS no *backbone* da RNP priorizando o tráfego de voz (atualmente em fase de implantação). Também permite correlacionar de forma mais precisa a qualidade de uma ligação VoIP com as características da rede RNP, bem como tomar medidas pró-ativas ao se identificar a degradação gradativa das conexões em determinado segmento do *backbone*. Embora existam ferramentas comerciais para este fim, muitas são voltadas para hardware específico e possuem custo elevado. A inexistência de ferramentas de domínio público, de código fonte aberto e independente de plataforma, são fatores de motivação deste trabalho. O desenvolvimento dessas ferramentas irá proporcionar a criação de uma massa de dados que reflete o perfil de uso de serviços VoIP da comunidade acadêmica brasileira possibilitando, assim, novas pesquisas em algoritmos para modelos de estimativa da qualidade de conexões VoIP, bem como o gerenciamento de redes oferecendo este serviço.

As ferramentas são baseadas na sinalização H.323 do ITU-T [44], e obtêm as informações para avaliação da qualidade de voz a partir dos dados obtidos de pacotes RTP e RTCP [5].

Este trabalho está estruturado em 6 capítulos. O primeiro capítulo apresenta a motivação e algumas noções introdutórias, e os objetivos deste trabalho.

O segundo capítulo apresenta a base teórica necessária para a compreensão do trabalho, mostrando os conceitos básicos de uma conexão VoIP, como o protocolo RTP e a estrutura da sinalização H.323.

O terceiro capítulo detalha os fatores que influenciam na qualidade de voz e os modelos de avaliação de qualidade existentes.

O quarto capítulo descreve a ferramenta de monitoração ativa, suas características e detalhes da sua implementação. São apresentados detalhes como a linguagem de programação utilizada, as bibliotecas necessárias, bem como as dificuldades encontradas. A interface com o usuário é descrita, e alguns exemplos de dados reais coletados através da ferramenta são mostrados. A ferramenta já vem sendo utilizada desde o final de 2002, com seus resultados publicados em [34, 35].

O quinto capítulo propõe a estrutura da ferramenta de monitoração passiva, o ambiente de programação e as bibliotecas a serem utilizadas. A arquitetura adotada permite a sua integração com estações de gerência SNMP, utilizando estruturas sendo propostas pelo RMON MIB *Working Group* do IETF [36]. Esta ferramenta irá compor o ambiente de gerência e diagnóstico do serviço VoIP da RNP, permitindo a análise da qualidade de voz instantânea, inclusive através de equipamentos ou implementações não-abertas como *gateways* entre redes IP e PSTN, que deverão ter uso intensivo no cenário proposto.

E finalmente, as perspectivas e conclusões são apresentadas no sexto capítulo, bem como sugestões de trabalhos futuros.

Capítulo 2

Protocolos e Sinalização

Os protocolos utilizados para VoIP podem ser divididos em duas categorias: protocolos de sinalização e protocolos de transporte. O protocolo RTP [5] é universalmente utilizado para aplicações de tempo real, sendo adotado como o padrão de transporte de voz. Em relação aos protocolos de sinalização, atualmente existem duas fortes correntes: o padrão H.323, proposto pelo ITU-T (setor de padronização de telecomunicações do *International Telecommunication Union*) [37], e o padrão SIP (*Session Initiation Protocol*) [38], proposto originalmente pelo grupo de trabalho de MMUSIC (*Multiparty Multimedia Session Control*) [39] do IETF (*Internet Engineering Task Force*) [40], sendo atualmente mantido pelo grupo de trabalho SIP [41] do IETF. Embora o padrão H.323 ainda tenha uma grande utilização no mercado, com diversos equipamentos, dispositivos e implementações disponíveis, o protocolo SIP aparece cada vez mais como tendência de padronização de protocolo de sinalização. No entanto, em virtude da maior variedade de implementações e por ser uma arquitetura proposta há mais tempo que o SIP, o H.323 permanece como padrão para sinalização de voz. Nos próximos anos, a integração e convivência destes dois padrões será uma constante em redes VoIP.

2.1. Real-time Transport Protocol

O protocolo RTP utiliza o UDP como camada de transporte. A garantia de entrega e integridade dos dados, fornecida pelo protocolo TCP, não é adequada para aplicações do tipo multimídia em tempo real, pois gera *overhead* e atrasos não tolerados por estas categorias de aplicação. O primeiro aspecto a ser observado é o fato de não existirem portas UDP padrão para transmissão de voz pelas sessões RTP. Estas portas são negociadas dinamicamente pelos protocolos de sinalização durante o processo de estabelecimento de chamada.

Normalmente cada pacote transporta entre 10 e 30 ms de amostra da voz [42]. Cada pacote RTP pode transportar uma ou mais amostras de voz codificadas. A Figura 1 mostra um pacote de voz, com o detalhamento do cabeçalho RTP.

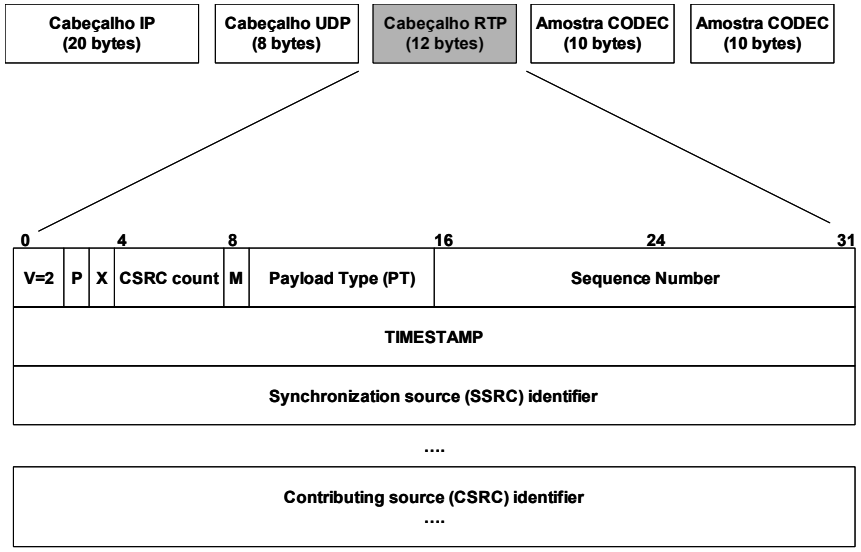


Figura 1. Pacote VoIP com detalhamento do cabeçalho RTP

O campo PT permite a identificação do tipo de pacote sendo transportado, como o formato da codificação utilizada pela mídia. O número de seqüência é iniciado com valor aleatório e incrementado a cada pacote transmitido, permitindo ao receptor identificar a perda de um ou mais pacotes. A descrição detalhada de todos os campos do cabeçalho RTP encontra-se no Apêndice A.

A simples observação do tamanho total do pacote VoIP, quando comparado ao tamanho do *payload* de voz sendo transportado, aponta para uma ineficiência do protocolo, em termos de *overhead* de cabeçalho. Embora não seja definido na RFC, o número de amostras sendo transportadas em um pacote RTP deve ser ponderado entre a redução de *overhead* de cabeçalho e o aumento de retardo gerado pela espera do preenchimento dos *buffers* de transmissão.

Como pode ser observado na Figura 1, o protocolo RTP utiliza o UDP como protocolo de transporte. O UDP não garante a entrega dos pacotes, nem a seqüência dos mesmos, bem como a retransmissão em caso de perda ou erro durante a transmissão.

Algumas destas deficiências são supridas pelo protocolo RTP, que acrescenta os seguintes serviços ao protocolo de transporte UDP:

- identificação entre os diversos transmissores em um fluxo RTP utilizando *multicast*;
- preservação da relação temporal existente entre os pacotes transmitidos;
- sincronização entre fluxos de mídias diferentes;
- seqüenciamento dos pacotes para a identificação de pacotes perdidos;
- identificação do tipo de mídia sendo utilizado no fluxo.

Vale ressaltar que o protocolo RTP não garante a qualidade de serviço associado ao fluxo. Esta funcionalidade de QoS (Qualidade de Serviço) é essencial para o funcionamento satisfatório de VoIP, devendo ser implementada na infra-estrutura de redes que será utilizada durante a conexão.

2.2. Real-Time Control Protocol

O protocolo RTCP (*Real-time Control Protocol*) [5] é especificado na mesma RFC que o protocolo RTP. Considerado parte integrante do protocolo RTP, o RTCP complementa a sua funcionalidade, permitindo ao transmissor receber relatórios dos receptores de como os dados estão sendo recebidos. No caso específico de VoIP, sua função é informar ao transmissor como a conexão está sendo vista pelo receptor em termos de QoS.

Existem cinco tipos diferentes de mensagens RTCP:

- Sender Report (SR)
- Receiver Report (RR)
- Source Description (SDES)
- Disconnection (BYE)
- Application-Specific (APP)

Diferentes tipos de pacotes RTCP podem ser encapsulados em um mesmo pacote UDP, criando um pacote RTCP composto. Desta forma, há uma redução do *overhead* causado pela inserção dos cabeçalhos IP e UDP.

Pacotes RTCP do tipo BYE são importantes para a identificação do término de uma chamada. Em termos de cálculo de QoS, os pacotes RTCP do tipo SR e RR fornecem os dados aos modelos de avaliação objetiva de qualidade. Os relatórios fazem largo uso de uma contabilidade acumulativa dos parâmetros. A justificativa dada na RFC é que esta forma permite realizar a verificação dos valores instantâneos, ou ao longo de toda a sessão. Parâmetros instantâneos de qualidade de serviço são obtidos através da diferença entre dois relatórios consecutivos. Valores acumulativos fornecem confiabilidade aos dados frente à perda de relatórios intermediários em possíveis situações de congestionamento da rede.

Recentemente, novos relatórios RTCP têm sido propostos para melhor representar os parâmetros de qualidade de serviço para voz sobre IP e facilitar as ferramentas de gerência de rede baseadas nestes relatórios. Esta proposta é conhecida como RTCP XR [56], mas ainda permanece em fase de estudo.

2.2.1. Sender Report e Receiver Report

O transmissor de um fluxo RTP envia em intervalos regulares pacotes RTCP do tipo SR (*Sender Report*) para o(s) receptor(es). O receptor, por sua vez, envia pacotes do tipo RR (*Receiver Report*) para os outros participantes da sessão. Desta forma, o transmissor sabe como o seu fluxo enviado está sendo recebido. No caso específico de VoIP, as conexões são normalmente realizadas em *unicast*, fazendo com que existam somente dois participantes por sessão.

Os pacotes RTCP do tipo SR, além de informações sobre o transmissor, podem incluir uma seção com informações equivalentes às fornecidas por um pacote RR, reportando também como está a recepção do transmissor. É importante observar que em uma sessão RTP, os transmissores também podem atuar como receptores de fluxos. Mais especificamente, em conexões VoIP, os participantes sempre atuam como transmissores e receptores ao mesmo tempo.

Uma vez que os participantes de uma sessão VoIP são transmissores e receptores simultaneamente, ambos enviam pacotes SR, independentemente de quem iniciou a transmissão. Cada ponta é informada de como está a qualidade vista pela outra

ponta através da seção de recepção dos pacotes SR recebidos. Mais especificamente, o transmissor obtém informações relevantes sobre aspectos da rede, como o RTT (*Round-Trip Time*), a taxa de perda de pacotes e a variação de atraso (*jitter*).

O formato de um pacote RTCP tipo SR é ilustrado na Figura 2. A função dos campos do cabeçalho de um pacote RTCP SR é descrita mais detalhadamente no apêndice A.

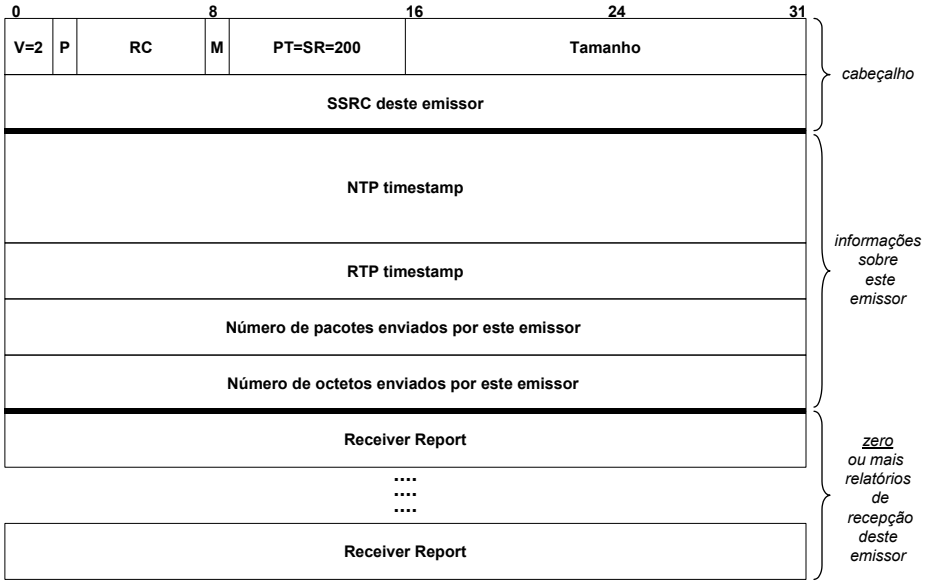


Figura 2. Formato da mensagem RTCP tipo SR

Um pacote RR é semelhante a um pacote SR, exceto pelo pacote RR não possuir a seção de informações sobre o transmissor e o campo *payload type* assumir o valor 201. A informação contida em uma seção de RR de um pacote SR é idêntica a um pacote RR, exceto por não possuir a parte inicial do cabeçalho.

Figura 3. Formato da mensagem RTCP tipo RR

A Figura 3 ilustra o formato de uma mensagem RTCP do tipo RR. A parte inicial do cabeçalho do pacote RR é idêntica a do pacote SR. Os campos específicos de um pacote RR são descritos no apêndice A. O receptor, ao receber um pacote SR, registra o momento exato de recepção. O valor do campo NTP do último pacote SR recebido é copiado para o campo LSR do pacote RR sendo enviado. Imediatamente antes da transmissão do pacote RR é calculado o tempo passado desde a recepção do último SR e colocado este valor no campo DLSR. O campo *Fraction Lost* informa o número de pacotes perdidos dividido pelo número de pacotes enviados pelo transmissor desde a recepção do último pacote SR.

2.2.2. Cálculo do RTT e da Perda de Pacotes

Conforme pode ser observado na Figura 4, o transmissor consegue calcular o RTT (*Round-Trip Time*) utilizando os campos LSR e DLSR (obtidos através de pacotes RR ou da seção de recepção de pacotes SR), e registrando o instante da recepção do pacote, através da seguinte equação [5]:

$$RTT = (LSR - \textit{instante-de-recepção}) - DLSR$$

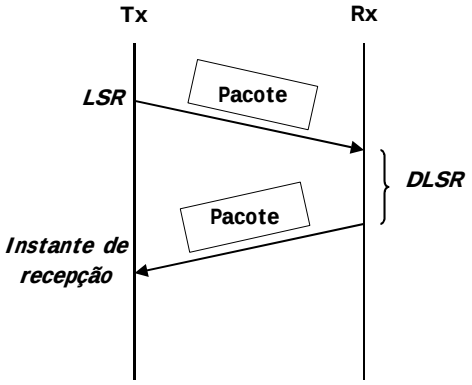


Figura 4. Cálculo do RTT

A perda de pacotes pode ser avaliada através dos campos *Fraction Lost* e *Cumulative Packets Lost*. O campo *Cumulative Packets Lost* representa o número total de pacotes RTP perdidos desde o início da sessão, calculado como a diferença entre o número de pacotes esperado e o número de pacotes realmente recebidos. Perdas de

pacotes acumulativas mostram congestionamentos persistentes, enquanto *jitter* aponta para congestionamentos temporários.

2.2.3. Cálculo de Jitter

O *jitter* é calculado pelo receptor a cada pacote RTP recebido, utilizando o campo *timestamp* do seu cabeçalho, e registrando o instante de chegada destes pacotes. A diferença do tempo em trânsito entre pacotes consecutivos é calculada de acordo com a seguinte fórmula [5]:

$$D(i,j) = (R_j - S_j) - (R_i - S_i)$$

onde $D(i,j)$ é a diferença de tempo em trânsito entre os pacotes adjacentes i e j, R_i é o instante em que o pacote i foi recebido e S_i é o instante em que o pacote i foi enviado, obtido através do campo *timestamp* do pacote RTP. A visualização destes valores pode ser feita através da Figura 5.

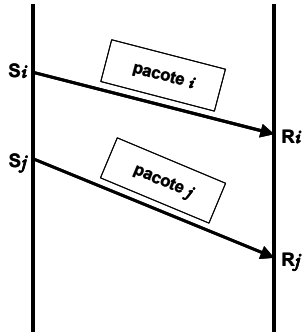
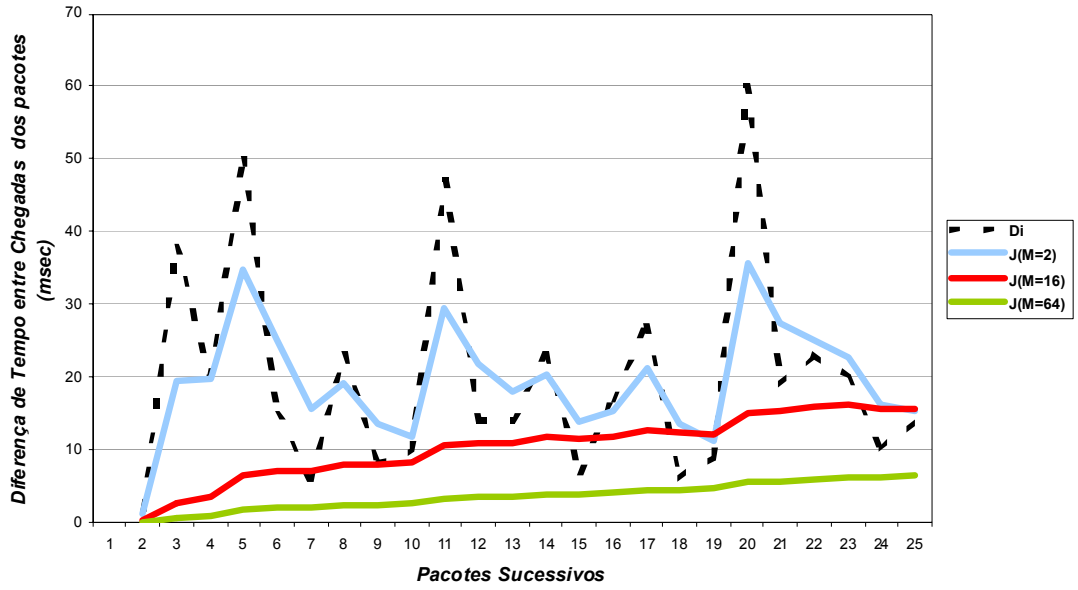


Figura 5. Cálculo de Jitter

A cada pacote RTP recebido, é feito o cálculo de $D(i, i-1)$, ou simplesmente D_i. O cálculo do jitter J_i, relativo ao pacote i inclusive, é feito através de seguinte fórmula [5]:

$$J_i = \frac{M-1}{M} \cdot J_{i-1} + \frac{1}{M} \cdot |D_i|$$

onde M é uma constante adotada na especificação do RTP com valor 16. O valor de M define o grau em que o cálculo do *jitter* será influenciado pelos atrasos anteriores. Para pequenos valores de M, a medição de *jitter* tem forte influência dos valores recentes de D_i, fazendo com que o *jitter* varie constantemente com a chegada de novos dados. Para valores maiores de M, a variação das medidas de *jitter* torna-se mais suave, reduzindo os picos. Para melhor visualizar o cálculo de *jitter* para diferentes valores de M, foi feita uma simulação em uma planilha e posteriormente gerado um gráfico, que pode ser visto na Figura 6. Os valores de atraso foram gerados utilizando a função rand() da planilha MS-Excel, com valores entre 60 e 250 ms, e aplicando a fórmula para valores de M=1,



M=2, M=16 e M=64.

Figura 6. Simulação de cálculo de jitter para diferentes valores de M

Como pode ser observado na simulação, o valor 16 para M, adotado para o cálculo de *jitter* [5], permite um cálculo de *jitter* médio ao longo do tempo com variações mais suaves, mas ainda assim refletindo algum comportamento das condições de rede momentâneas [43].

2.3. O Padrão H.323

A recomendação do ITU-T H.323 (*Packet-Based Multimedia Communication Systems*) [44] é o conjunto de padrões mais amplamente utilizado para comunicações multimídia como vídeo e voz. O padrão H.323 é descrito em um documento que faz referência a protocolos e formatos de mensagens definidos em outros documentos do ITU-T, bem como a forma a qual estes protocolos interagem com os elementos que compõe o ambiente.

As funções de sinalização são definidas na recomendação do ITU-T através das especificações da Tabela 1[44]. A sinalização H.225 é usada para o estabelecimento de chamadas entre pontos H.323. A sinalização H.245 é usada para troca de características dos terminais e usada para abrir e fechar canais de controle.

Tabela 1. Recomendações ITU-T para as funções de sinalização

Recomendação	*Título*
H.225.0	Call Signaling Protocols and Media Stream Packetization for Packet-Based Multimedia Communication Systems
H.235	Security and Encryption for H-Series Multimedia Terminals
H.245	Control Protocol for Multimedia Communication
H.450.x	Supplemental Services for H.323
T.120 series	Data Protocols for Multimedia Conferencing

Além disto, o H.323 incorpora uma variedade de formatos de mídia e estruturas de dados para as aplicações, como os codificadores de áudio G.711, G.722, G.723.1, G.728, G.729, GSM e os codificadores de vídeo H.261, H.262 e H.263.

2.3.1. Componentes do H.323

As comunicações utilizando H.323 ocorrem sempre entre um dos elementos abaixo, conforme ilustrado na Figura 7, sejam eles dispositivos de hardware ou simplesmente componentes de software:

- Terminal
- Multiport Control Unit (MCU)
- Gateway
- Gatekeeper

Apesar de representarem elementos lógicos diferentes dentro da arquitetura H.323, *gatekeepers*, *gateways* e MCUs podem ser implementados em um único dispositivo.

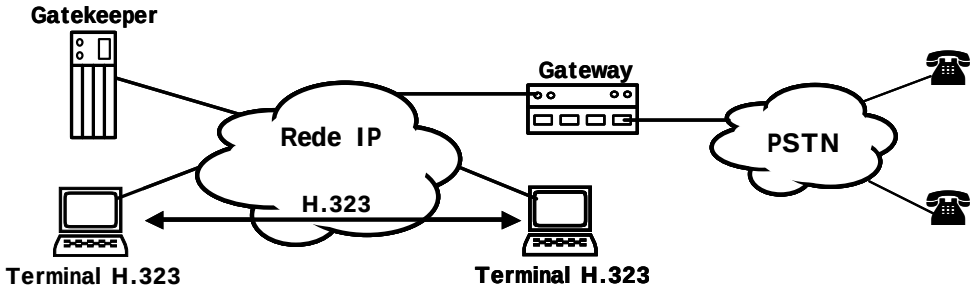


Figura 7. Componentes H.323

2.3.2. Terminal

Um terminal H.323 é a ponta de uma comunicação H.323, que é bidirecional e em tempo real. Possui interfaces baseadas em pacotes, podendo ser tanto um computador pessoal quanto um dispositivo dedicado executando uma aplicação multimídia H.323. Operado pelo usuário final, desempenha função semelhante à do aparelho telefônico em uma estrutura de telefonia convencional.

Sua funcionalidade é montada através de diversos módulos que compõe sua estrutura, representados na Figura 8.

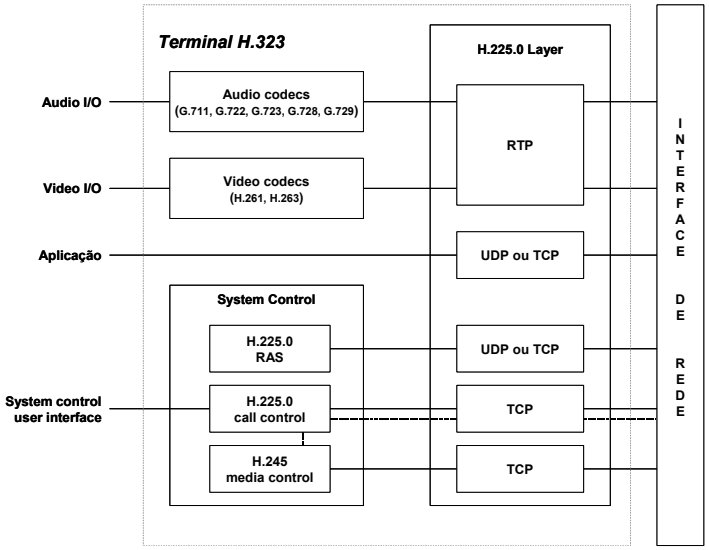


Figura 8. Estrutura funcional de um terminal H.323

A camada H.225.0 provê a funcionalidade de encapsulamento, seqüenciamento e detecção de erros e correção, quando possível, para cada tipo de mídia. É responsável pela formatação de vídeo, áudio, dados e fluxos de controle em pacotes a serem enviados durante a transmissão e pela recepção dos pacotes que chegam através da interface de rede. No caso específico de VoIP, é utilizado o protocolo RTP para o transporte de áudio e tanto TCP quanto UDP para mensagens de controle.

A unidade de controle do sistema (utilizando os protocolos H.225.0 e H.245) fornece a sinalização necessária para o funcionamento do terminal H.323. Provê as funcionalidades de controle de chamada, troca de características específicas de cada terminal, sinalização vinda de comandos e ainda define as mensagens para a abertura e o estabelecimento de canais lógicos de comunicação, além da descrição completa de seu conteúdo.

O módulo H.225.0 RAS (*Registration, Admission, Status*) gerencia o registro, admissão e sinalização de controle entre o terminal e o *gatekeeper*. Quando a comunicação é feita diretamente ponto-a-ponto entre terminais, este módulo não é utilizado.

O módulo H.225.0 *call control* administra o estabelecimento/encerramento de uma chamada entre terminais ou *gateways* e estabelece o canal de comunicação para controle de mídia (H.245).

Através do protocolo H.245, são negociados os codificadores a serem utilizados durante a transmissão, os requisitos de QoS e outras opções específicas de mídia. Também são negociadas as portas das sessões RTP.

Terminais H.323 obrigatoriamente devem suportar os codificadores de áudio, mas codificadores de vídeo e aplicações de dados são opcionais.

2.3.3. Gatekeeper

Embora não seja obrigatório, desempenha um papel fundamental em uma rede H.323. Cada rede H.323 pode ser dividida em zonas, ou seja, um conjunto de dispositivos agrupados logicamente que são controlados por um *gatekeeper*. Cabe ao *gatekeeper* gerenciar quais terminais de sua zona poderão iniciar ou receber chamadas. Também pode controlar como as chamadas serão realizadas: se diretamente entre os terminais, ou passando pelo *gatekeeper*. Caso a chamada seja realizada através do *gatekeeper*, este atua como controlador da sinalização trocada entre as pontas, podendo desempenhar também o papel de *proxy* de mídia. Esta característica de atuação como *proxy* de sinalização e mídia é importante, pois permite concentrar todas as conexões em um único ponto, facilitando a gerência de recursos e segurança. Este aspecto é discutido no capítulo referente a ferramenta de monitoração passiva.

Outros serviços são realizados pelo *gatekeeper* como, por exemplo: tradução de endereço; autorização e autenticação de terminais e *gateways*; controle e gerenciamento de banda de rede sendo utilizada e ainda contabilização das chamadas.

A tradução de endereços feita pelo *gatekeeper* permite que endereços H.323 (com formato semelhante ao utilizado em correio eletrônico) ou E.164 (adotado pela telefonia convencional) [45] sejam convertidos em endereços IP e portas TCP/UDP para que a conexão seja efetivada. Desta forma, é possível realizar chamadas através de identificação convencional de telefonia, permitindo que chamadas feitas via telefones comuns possam se comunicar com terminais H.323 de forma transparente. Também permite que a comunicação entre terminais H.323 passe a ser feita através de nomes associados a terminais, conhecidos como *alias*.

O controle de admissão para terminais, *gateways* e MCUs é feito através do canal de comunicação do H.225.0 RAS, por intermédio de mensagens de *admission request* (ARQ), *admission confirm* (ACF) e *admission reject* (ARJ).

O *gatekeeper* pode controlar a banda sendo utilizada também através de sinalização H.225.0 RAS, com mensagens *bandwidth request* (BRQ), *bandwidth confirm* (BRC) e *bandwidth reject* (BRJ). Por exemplo, se o administrador especificar

um limite de conexões simultâneas em uma zona H.323, o *gatekeeper* pode recusar uma nova conexão após o limite ser atingido.

2.3.4. Multiport Control Unit

MCUs possibilitam conferências entre três ou mais terminais H.323. Todos os terminais H.323 participantes da conferência estabelecem uma conexão com o MCU. O MCU gerencia os recursos utilizados pela conferência e faz a negociação entre os terminais para determinar quais codificadores de áudio e vídeo serão utilizados, podendo ainda administrar o fluxo da mídia.

2.3.5. Gateways

Gateways viabilizam a interconexão de redes H.323 com outras de diferentes tecnologias como, por exemplo, a rede de telefonia pública convencional (PSTN). Desta maneira, um aparelho de telefone convencional pode ser visto como um terminal H.323 através do *gateway*. De forma análoga, o *gateway* possibilita que um terminal H.323 possa efetuar uma chamada para um telefone convencional ligado a um PBX.

É importante ressaltar que o *gateway* tem que fazer a conversão da sinalização das chamadas e de mídia, além da mídia propriamente dita, como ilustrado na Figura 9.

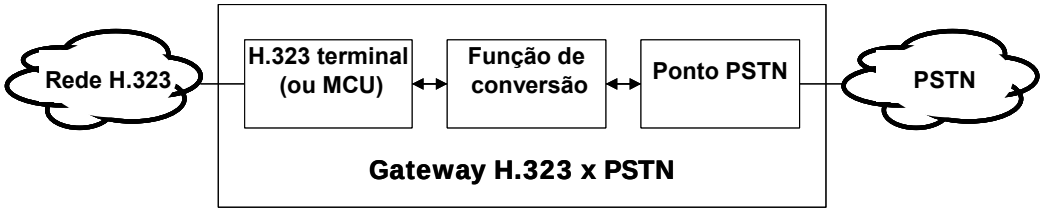


Figura 9. Esquema de funcionamento de um gateway H.323

2.3.6. Protocolos H.323

A camada de rede e de transporte é transparente para a pilha de protocolos H.323, não fazendo parte da sua recomendação [44].

A Figura 10 mostra toda a pilha de protocolos H.323, assumindo uma rede IP.

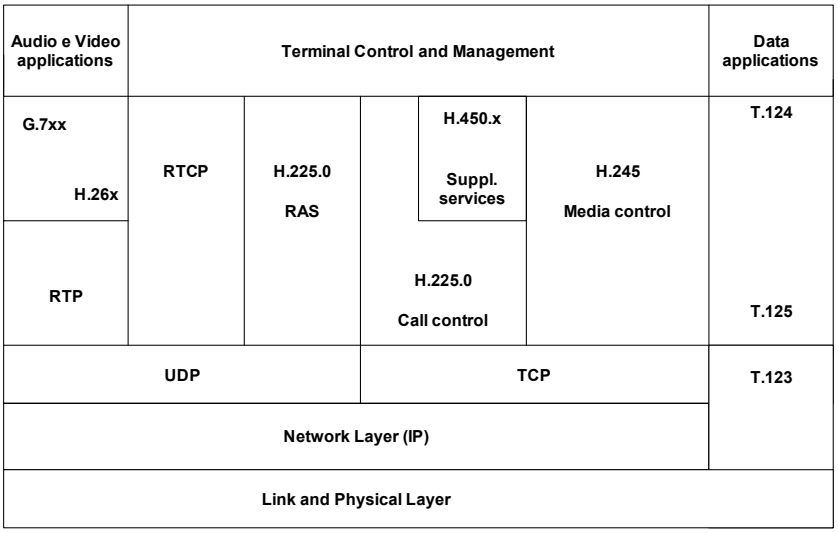


Figura 10. Pilha de Protocolos H.323

2.3.6.1. H.225 RAS

O protocolo de sinalização entre *endpoints* (terminais e *gateways*) e *gatekeepers* é o RAS (*Registration, Admission, Status*). O RAS permite o registro de *endpoints* no *gatekeeper*, o controle de admissão pelo *gatekeeper*, status e a desconexão com o *gatekeeper*. Um canal RAS é usado para a troca de mensagens do tipo RAS. Este canal de sinalização é aberto entre o *endpoint* e o *gatekeeper* antes de qualquer outro canal de comunicação ser estabelecido.

2.3.6.2. H.225 Call Signaling

O protocolo de sinalização H.225 *call signaling* é utilizado para o estabelecimento de sessão entre *endpoints* ou *endpoints* e *gatekeeper*, através da qual os dados de mídia serão transportados. A troca de mensagens H.225 é feita usando um canal de comunicação confiável. No caso de redes IP, o TCP é adotado como protocolo de camada de transporte.

As mensagens H.225 são trocadas diretamente entre os *endpoints*, caso não esteja sendo usado um *gatekeeper*. O *gatekeeper* pode funcionar como interface entre a troca de mensagens dos *endpoints*, neste caso atuando como *proxy*.

2.3.6.3. H.245 Control Signaling

O protocolo de sinalização H.245 é feito entre os *endpoints*. Permite a abertura e o fechamento de canais lógicos de comunicação para o transporte de mídia, além da troca de características dos terminais como *codecs* suportados, por exemplo.

2.3.7. Estabelecimento de Conexões H.323

Os passos de estabelecimento de uma chamada H.323, troca de mídia e término da chamada são descritos a seguir. O exemplo assume o estabelecimento de sessão entre terminais H.323 (T1 e T2) conectados através de um *gatekeeper*.

O estabelecimento de uma chamada H.323, ilustrado na Figura 11, é descrito a seguir. T1 envia uma mensagem RAS ARQ para o canal RAS do *gatekeeper*, solicitando o seu registro (1). Neste exemplo, T1 solicita sinalização direta com o outro *endpoint* (T2). O *gatekeeper* confirma a admissão de T1, enviando uma mensagem ACF para T1 (2). T1 então envia uma mensagem H.225 *call signaling* para estabelecimento de sessão (*setup*) com T2 (3). T2 responde com uma mensagem de estabelecimento de sessão em andamento (4). Em seguida, T2 solicita registro ao *gatekeeper*, através de uma mensagem ARQ (5), sendo confirmado pela mensagem ACF enviada pelo *gatekeeper* (6). T2 alerta T1 sobre o estabelecimento de conexão (7). Finalmente, T2 confirma o estabelecimento da chamada (8).

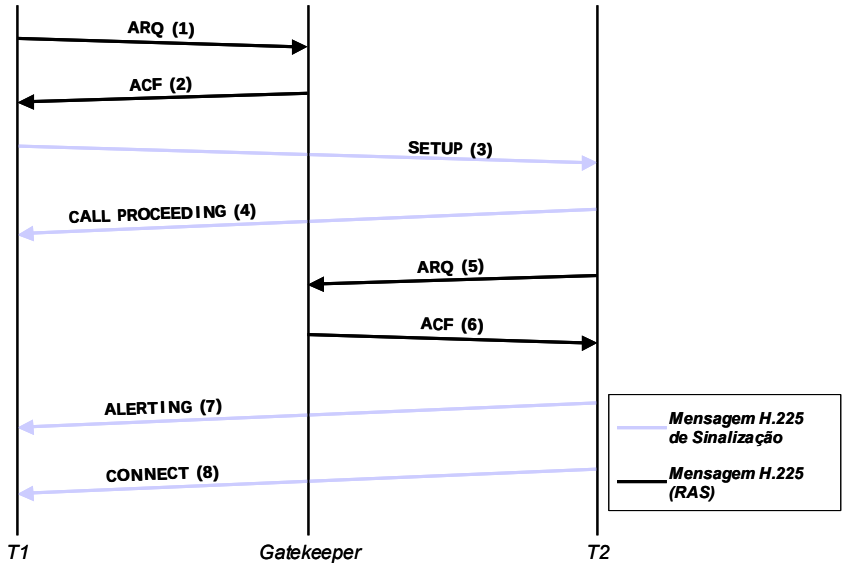


Figura 11. Estabelecimento de Chamada H.323

A Figura 12 mostra um exemplo de sinalização H.245, com troca de características entre *endpoints* e negociação de portas de mídia.

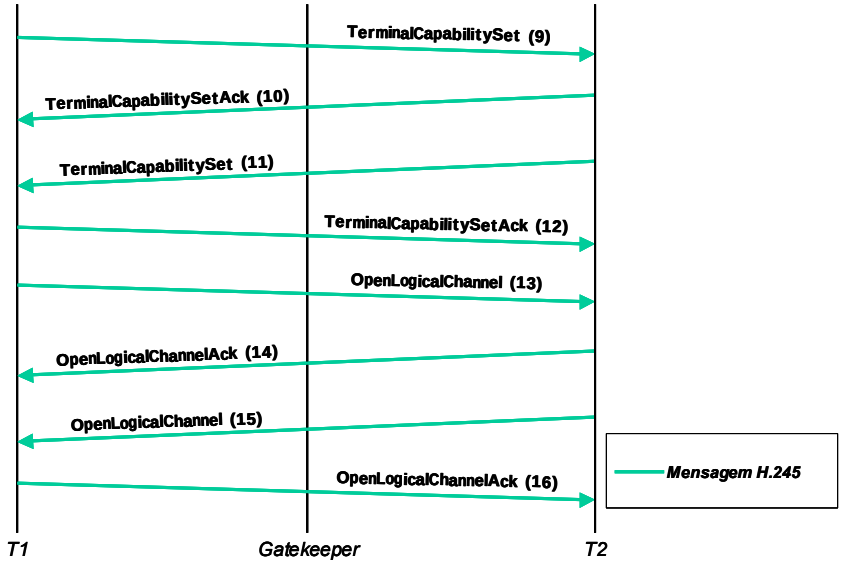


Figura 12. Sinalização de controle H.323

Um canal de controle H.245 é estabelecido entre T1 e T2. T1 envia uma mensagem para T2 informando suas características (9), com confirmação de T2 para T1 em seguida (10). T2 envia para T1 as suas características (11), com confirmação posterior de T1 (12). T1 abre um canal lógico de comunicação de mídia, enviando uma mensagem H.245 *OpenLogicalChannel*, informando também o canal de comunicação do RTCP (13). T2 confirma o estabelecimento do canal, informando qual a porta que T2 estará pronto para receber pacotes RTP de T1 (14). De forma análoga, T2 abre o seu canal de comunicação com T1 (15), informando a sua porta de recebimento de pacotes RTCP. T1 informa em seguida qual a porta estará recebendo pacotes RTP vindos de T2 (16). Após esta troca de mensagens, um canal bidirecional de mídia é estabelecido entre T1 e T2, com pacotes RTP e RTCP passando a trafegar entre os dois, transportando a mídia e informações de controle de QoS.

O término de uma sessão é ilustrado na Figura 13. Quando um dos *endpoints* decide encerrar a comunicação, uma mensagem de encerramento de sessão é enviada. No exemplo, T2 inicia o encerramento da chamada (17). T1 libera a chamada e confirma o seu término através de mensagem H.245 para T2 (18). T2 envia então uma mensagem de sinalização H.225 informando que a conexão foi desfeita (19). Posteriormente, T1 e T2 solicitam ao *gatekeeper* sua desconexão, através de mensagens RAS DRQ (20). O *gatekeeper* confirma finalmente a liberação dos *endpoints*, através de mensagens DCF (21).

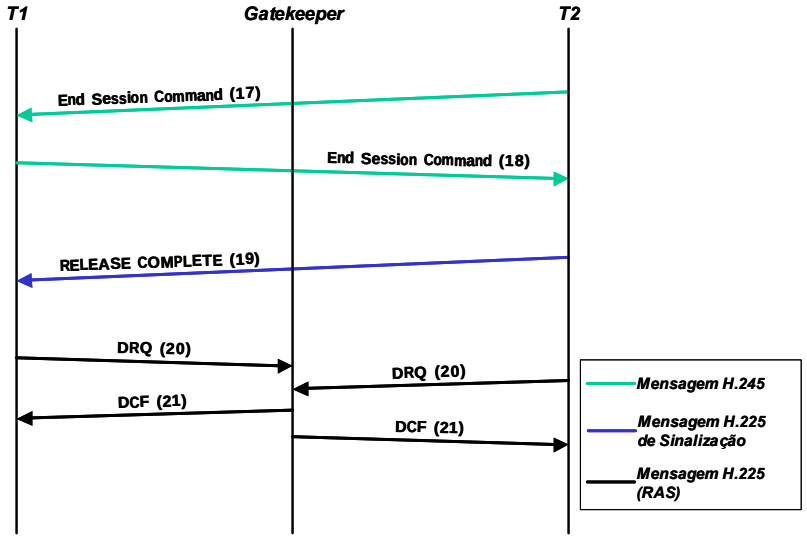


Figura 13. Término de chamada H.323

Todos estes procedimentos de sinalização são importantes para a ferramenta de monitoração passiva, descrita no capítulo 5.

Capítulo 3

Qualidade de voz

Este capítulo descreve os principais fatores que afetam a qualidade de uma ligação utilizando VoIP e em seguida os principais modelos de avaliação da qualidade são detalhados.

3.1. Fatores de Qualidade de Voz

Diversos fatores influenciam diretamente na qualidade de uma conexão de voz. Pode-se observar que certos fatores são independentes da tecnologia sendo adotada como por exemplo, aspectos do comportamento humano ou ruídos externos durante a transmissão ou recepção da ligação.

3.1.1. Perda de Pacotes

Inicialmente, para se efetuar uma transmissão VoIP, é necessário fazer a captura do sinal de voz do transmissor, através de processos de amostragem, digitalização e codificação, gerando pequenos quadros de amostras de voz. Estas tarefas são executadas por codificadores, conhecidos como *codecs*. Estes quadros são então encapsulados em pacotes RTP (*Real-time Transport Protocol*) [5] que por sua vez utilizam o UDP (*User Datagram Protocol*) [6] como protocolo de transporte. A adoção do UDP é importante, pois as facilidades de retransmissão e garantia de entrega oferecidas pelo protocolo de transporte TCP (*Transmission Control Protocol*) [7] não são adequadas para a transmissão de voz em tempo real.

Em relação à perda de pacotes, existem dois enfoques distintos. Sob o ponto de vista da rede de transmissão, uma perda é ocasionada quando um pacote não consegue chegar ao seu destino. Entretanto, para aplicações multimídia em tempo real, pacotes que chegam muito atrasados em relação ao instante de tempo em que deveriam ser reproduzidos no lado receptor tornam-se inúteis, e conseqüentemente, são descartados,

sendo considerados pacotes perdidos sob o ponto de vista do receptor. Portanto, ao se avaliar a qualidade de voz, não basta computar os pacotes perdidos em relação à camada de rede, mas também em relação à camada de aplicação.

3.1.2. Perda de Pacotes em Rajadas

A ocorrência de determinados fatores na infra-estrutura de redes durante um período de tempo longo podem causar a perda de vários pacotes consecutivos. A perda de pacotes em rajada provoca a degradação na voz, de modo que até mesmo *codecs* com algoritmos de correção de perdas não conseguem minimizar seu efeito. Cox e Perkins [8] compararam o impacto da perda de pacotes distribuída randomicamente ao longo de uma ligação em relação à perda em rajadas, utilizando os *codecs* G.711 e G.729A.

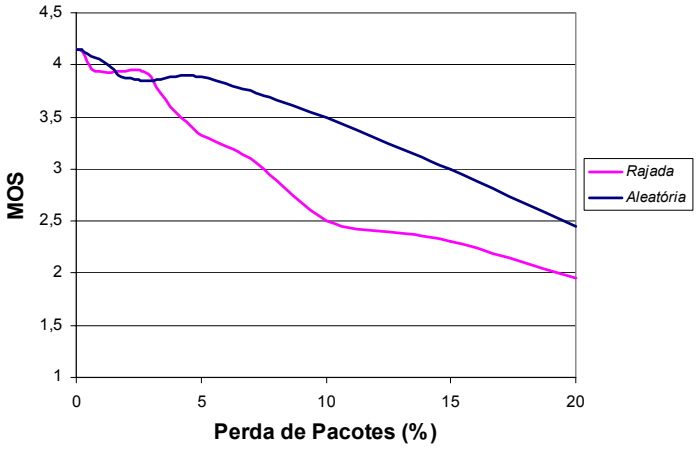


Figura 14. Impacto de perda em rajada X perda randomicamente distribuída

Foi constatado que, para taxas de perda muito baixas, a qualidade das ligações com perda de pacotes em rajadas foi mais bem avaliada subjetivamente do que as ligações com perdas randomicamente distribuídas. Tal efeito ocorre pois uma rajada de curta duração tem um impacto não muito maior do que a perda de um único pacote, além de existirem grandes períodos de transmissão normal entre as perdas. No entanto, quando a perda ultrapassa o limite de 3%, pode-se observar que a perda em rajadas tem um impacto muito superior do que a perda distribuída randomicamente, como pode ser visto na Figura 14.

3.1.3. Atraso ou Latência

Baixos níveis de atraso são imperceptíveis em uma conversação, no entanto valores maiores proporcionam desconforto, tornando as falas desencontradas, dependendo do grau de interatividade do diálogo. O padrão ITU-T G.114 [9] recomenda que o atraso máximo em um sentido (*one-way delay*) seja no máximo de 150 ms. Entretanto, o impacto que o atraso causa na qualidade da voz depende de outros fatores, como o grau de interatividade da própria conversa. Este atraso é formado por um componente fixo e outro variável [10].

3.1.3.1. Atraso Fixo

Exemplos de atraso fixo são: atraso de empacotamento, atraso de enfileiramento dos bits para transmissão e o tempo de propagação no meio físico.

O tempo necessário para amostragem e codificação dos sinais de voz analógicos em pacotes de dados, seguido opcionalmente de compactação, é conhecido como *packetization delay*. Este tempo é fixo e depende diretamente do tipo de codificação e *codec* utilizado, variando conforme sua complexidade, conforme pode ser observado na Tabela 2. O tempo decorrente da descompactação e decodificação no lado receptor é sempre é muito inferior, mas também deve ser contabilizado [11].

Tabela 2. *Packetization Delay* por codec

Algoritmo / Codec	Largura de banda	Payload (bytes)	Packetization delay (ms)
PCM, G.711	64 kbps	160	20
		240	30
ADPCM, G.726	32 kbps	80	20
		120	30
CS-ACELP, G.729	8 kbps	20	20
		30	30
MP-MLQ, G.723.1	6,3 kbps	24	24
		60	48
MP-ACELP, G.723.1	5,3 kbps	20	30
		60	60

O tempo necessário para colocar os bits dos pacotes de dados no meio físico, ou *serialization delay*, depende diretamente da velocidade de transmissão do meio físico utilizado e do tamanho do pacote sendo transmitido [11]. O *overhead* no tamanho do pacote causado pela camada de enlace também deve ser considerado. O atraso de serialização pode ser obtido através da divisão do tamanho do quadro sendo transmitido pela taxa de transmissão do meio, conforme mostra a Tabela 3.

Tabela 3. Tempo de serialização do pacote em bits

Tamanho do quadro (bytes)	Velocidade da canal de transmissão (Kbps)									
	19,2	**56**	**64**	**128**	**256**	**384**	**512**	**768**	**1024**	**2048**
38	15,83	5,43	4,75	2,38	1,19	0,79	0,59	0,40	0,30	0,15
48	20,00	6,86	6,00	3,00	1,50	1,00	0,75	0,50	0,38	0,19
64	26,67	9,14	8,00	4,00	2,00	1,33	1,00	0,67	0,50	0,25
128	53,33	18,29	16,00	8,00	4,00	2,67	2,00	1,33	1,00	0,50
256	106,67	36,57	32,00	16,00	8,00	5,33	4,00	2,67	2,00	1,00
512	213,33	73,14	64,00	32,00	16,00	10,67	8,00	5,33	4,00	2,00
1024	426,67	146,29	128,00	64,00	32,00	21,33	16,00	10,67	8,00	4,00
1500	625,00	214,29	187,50	93,75	46,88	31,25	23,44	15,63	11,72	5,86
2048	853,33	292,57	256,00	128,00	64,00	42,67	32,00	21,33	16,00	8,00

O tempo de propagação no meio físico também constitui um componente fixo de atraso, conhecido como *propagation delay*. Utiliza-se como método para estimar o tempo de propagação os valores 10 microssegundos por milha ou 6 microssegundos por quilômetro [10, 11].

3.1.3.2. Atraso Variável

Componentes variáveis de atraso são relacionados a problemas de congestionamento na rede, ou mesmo a mudanças na sua topologia. Roteadores e comutadores no caminho entre a origem e o destino da conexão geram atrasos variáveis, dependendo da velocidade de comutação de pacotes dos equipamentos, bem como dos retardos decorrentes de enfileiramento e armazenamento dos pacotes durante a sua transmissão. A implementação de um esquema de QoS reduz este fator, dando

prioridade aos pacotes de voz e, por conseqüência, reduzindo o seu tempo de espera em filas de transmissão.

3.1.4. Jitter

Como descrito no item anterior, o atraso de um pacote de voz da origem até o destino não é constante, pois possui componentes fixos e variáveis. Esta variação de atraso, conhecida como *jitter*, geralmente é solucionada através da utilização de *buffers* no lado receptor de compensação de *jitter*. Antes de serem passados para reprodução pelo *codec*, os pacotes são reordenados e seu compasso reajustado de forma a receberem os tempos relativos originais do momento da transmissão. *Buffers* maiores permitem compensar variações maiores, mas aumentam o atraso total da reprodução. A maioria das implementações deve compensar um *jitter* médio de até 30 ms [11, 13].

3.1.5. Eco

Outro fator que influencia a qualidade de uma ligação telefônica é o efeito de eco. O eco não é considerado problema quando ocorre com defasagem de 16 a 20 ms em relação ao som original. Pelo contrário, fornece ao transmissor a sensação que a conexão ainda está estabelecida, e que está sendo recebida pela outra ponta. No entanto, para valores superiores a 32 ms, surge uma sensação de desconforto para o usuário [3, 4].

3.1.6. Problemas Relacionados a Codecs

Codificadores de voz, ou simplesmente *codecs*, são responsáveis no lado transmissor pela conversão de sinais analógicos em sinais digitais, e sua posterior transformação em pacotes de dados. Esses codificadores desempenham função inversa no lado receptor. Cada *codec* possui o seu próprio algoritmo de codificação. Alguns algoritmos de *codecs* reduzem a banda necessária para a transmissão através da supressão de informações redundantes ou menos relevantes. A taxa de compressão obtida é determinada por fatores como, por exemplo, qualidade da voz requerida para a conexão, capacidade computacional de equipamento fazendo a codificação e atraso

máximo tolerado. A compressão de sinal de voz faz uso intensivo da capacidade de processamento do equipamento. Quanto maior a redução de largura de banda obtida, maior o custo computacional do *codec*, conseqüentemente, maior o atraso gerado pelo processo de codificação. Este atraso causado pela complexidade do *codec* pode ser constatado na Tabela 2. Os *codecs* G.729 e G.723.1 obtêm taxas reduzidas de transmissão, pois ao invés de transmitir uma representação do formato da onda do sinal de voz, transportam uma representação subjetiva do som original, que é interpretado e reproduzido no lado receptor. Isto acarreta numa maior sensibilidade destes *codecs* a fatores de degradação como perda de pacotes, por exemplo. Ao se avaliar a qualidade do serviço VoIP, tais fatores devem ser levados em consideração.

Alguns codecs incorporam a característica de supressão de silêncio, também conhecido como VAD (*Voice Activity Detection*). Pacotes contendo períodos de silêncio não são transmitidos, gerando economia na banda consumida durante a transmissão de até 50% [11, 12].

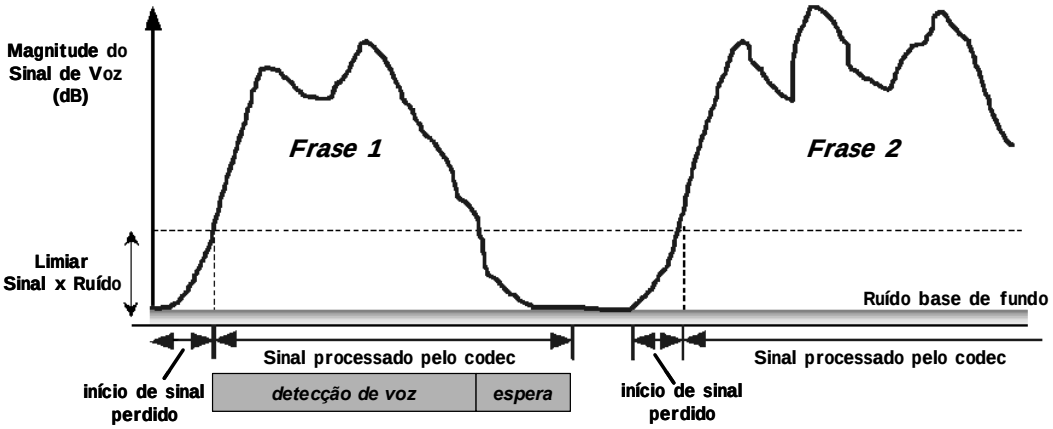


Figura 15. Supressão de Silêncio

O uso de VAD pode acarretar em uma degradação na qualidade obtida na reprodução do sinal, como ilustrado na Figura 15. O *codec*, após detectar um período de silêncio, pode perder o início de uma palavra. Tal efeito é conhecido como FEC (*front end clipping*) [13]. O tempo de espera para se identificar um período de silêncio é conhecido como HOT (*hold-over time*). O HOT, quando é excessivo, reduz a economia de banda obtida. Por outro lado, quando o HOT é muito pequeno, a reprodução gerada é

muito picotada, pois qualquer pausa entre palavras é considerada um período de silêncio [13].

Outro efeito decorrente do emprego de VAD é a sensação de perda de conexão por parte do usuário receptor, já que este não recebe sinal algum de áudio durante longos períodos de silêncio. Para minimizar este efeito, o *codec* do lado receptor gera um ruído de conforto, dando a sensação ao usuário que a conexão permanece ativa [13].

3.1.7. Efeito "Memória Recente"

Um fator importante durante a avaliação da qualidade de voz é a observação não apenas dos fatores que causam a degradação da qualidade de uma conexão, mas também a distribuição da ocorrência destes fatores ao longo da ligação. Ligações onde ocorram falhas mais concentradas no seu final tendem a ser mais mal avaliadas do que outras ligações. Isto é explicado pelo fato do cérebro humano dar maior prioridade a informações mais recentes, conforme Baddeley [65].

Um estudo realizado pelo laboratório de pesquisas da AT&T comprova o efeito memória [13]. Neste trabalho, uma mensagem com 60 segundos de duração foi avaliada de forma subjetiva por três grupos de avaliadores. Em cada grupo foi introduzido um período de ruído durante a transmissão. O resultado obtido destas avaliações pode ser observado na Figura 16.

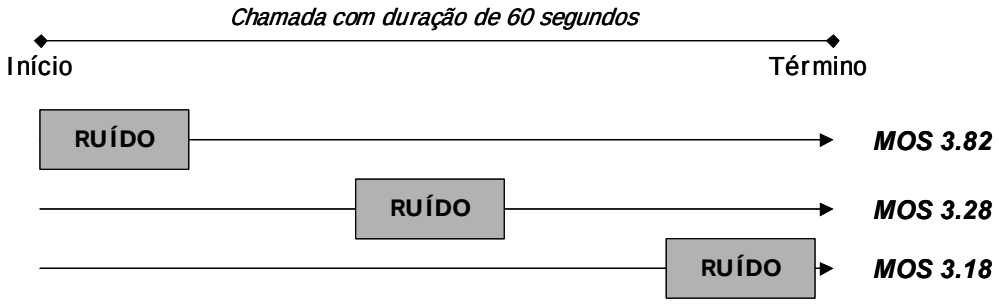


Figura 16. Estudo comprovando o "efeito-memória"

Outro estudo, feito nos laboratórios da France Telecom em [14], demonstra que melhorias nas condições da ligação não refletem em um aumento imediato da qualidade percebida. O trabalho consistia em ligações de 3 minutos de duração, com perda de

pacotes variando de 0 a 25%. Em um dos testes, cujo resultado é mostrado na Figura 17, uma perda de 25% foi mantida durante quase todo o período da ligação, com um período de 30 segundos sem perdas no meio da chamada. Foi solicitado aos avaliadores que, ao longo de toda a ligação, fossem registrando a qualidade instantânea, e que também fosse realizada ao final da ligação uma avaliação geral.

O resultado obtido é uma curva exponencial com um tempo constante de aproximadamente 5 segundos de diferença em relação ao momento de transições da qualidade da ligação de boa para ruim e de 15 segundos em transições de qualidade ruim para boa. Neste mesmo teste, obteve-se um resultado similar ao obtido nos testes da AT&T em relação ao efeito memória [13].

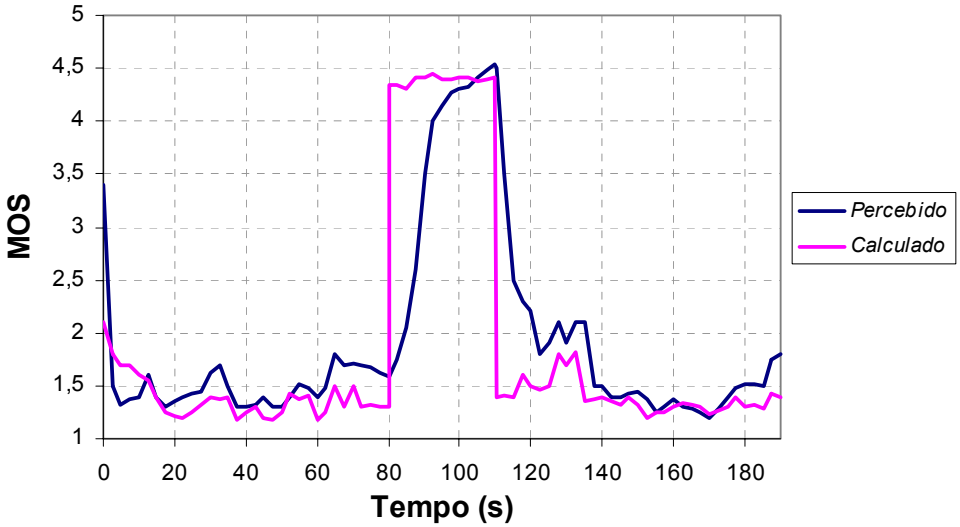


Figura 17. Relação entre qualidade "percebida" X "calculada"

No gráfico apresentado na Figura 17 pode-se observar que uma melhora na qualidade real no sinal demora a ser refletida na qualidade percebida. Por outro lado, quando a qualidade volta ao patamar anterior de qualidade, seus reflexos aparecem quase que imediatamente na qualidade percebida.

O resultado deste e outros testes levam a conclusão [30] representada na Figura 18. Observa-se que a qualidade do sinal, conforme percepção do usuário em relação à qualidade real transmitida, acompanha de forma muito mais próxima nas transições de qualidade boa para ruim do que de ruim para boa.

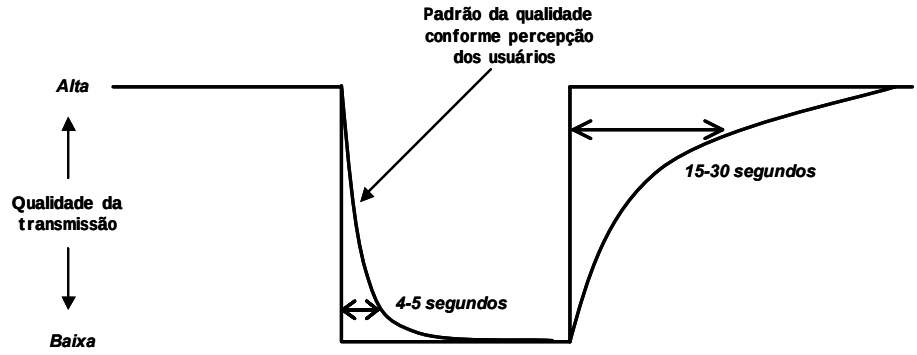


Figura 18. Curva de percepção do usuário

O gráfico da Figura 18 mostra claramente que o ser humano identifica muito mais rapidamente a queda de qualidade do que o retorno ao patamar anretior a queda.

3.2. Métodos para Avaliação da Qualidade de Voz

As metodologias mais importantes são descritas a seguir, segundo ordem cronológica em que foram propostas e empregadas pela indústria [3, 4].

3.2.1. MOS

O MOS é um modelo subjetivo que utiliza um conjunto de avaliadores para escutar individualmente ligações utilizando mensagens pré-gravadas. O valor médio destas avaliações representa quantitativamente a qualidade da ligação. Este modelo é descrito pela recomendação ITU P.800 [16]. A recomendação P.830 [21] descreve os testes mais específicos para uma análise subjetiva de codificadores de áudio.

Nestas recomendações são encontrados os métodos de cada um dos testes, o esquema de pontuação para cada item, as características das amostras de voz a serem utilizadas nos testes, e as condições gerais de como estes testes devem ser aplicados.

Os testes de MOS podem ser aplicados em conexões nas quais as duas pontas são transmissoras e receptoras, portanto ambas avaliam a ligação, ou então feita unicamente em uma das extremidades da conexão, onde o avaliador atua apenas como

receptor. No segundo caso, uma mesma mensagem padrão é aplicada repetidas vezes para apreciação de diferentes avaliadores.

Tabela 4. Classificação de qualidade MOS

Nota	Qualidade de voz
5	*Excelente*
4	*Boa*
3	*Razoável*
2	*Fraca*
1	*Ruim*

A avaliação de qualidade da conexão e da clareza de voz obtida é feita utilizando a escala apresentada na Tabela 4.

Outro critério comumente usado na avaliação de qualidade é o esforço realizado para o entendimento do significado das mensagens, cuja classificação de qualidade é apresentada na Tabela 5.

Tabela 5. Classificação de Qualidade MOS em relação ao entendimento

Nota	Compreensão do significado da mensagem
5	*Completo relaxamento; nenhum esforço exigido.*
4	*Necessário prestar atenção; esforço não significativo exigido.*
3	*Esforço moderado exigido.*
2	*Esforço considerável exigido.*
1	*Impossível entender, independente do esforço realizado.*

Quando avaliações MOS são aplicadas a ligações VoIP, valores considerados satisfatórios variam entre 3,5 e 4,2 [21].

No entanto, os testes de MOS apresentam algumas desvantagens. Por ser completamente subjetivo, existem fatores que nem sempre podem ser controlados como, por exemplo, o próprio estado de espírito do avaliador, o idioma utilizado, o perfil da conversa (diálogo coloquial *versus* diálogo técnico, com números, por exemplo) [21]. Conseqüentemente, o MOS deve ser aplicado a um grande número de avaliadores para

representar valores consistentes, tornando o processo dispendioso e de implementação complexa.

O método MOS não se aplica a um esquema de utilização diária ou freqüente, necessário para se obter avaliação constante da qualidade do serviço sendo oferecido. Porém, somente uma avaliação constante permite fazer uma gerência pró-ativa e um planejamento antecipado de capacidade dos recursos.

As desvantagens encontradas no MOS reforçam a necessidade de se utilizar métodos objetivos, automatizados e repetitivos. Foram concebidos diversos métodos para avaliar de forma objetiva a clareza da voz e sua percepção humana [15, 16, 17, 18, 19].

3.2.2. PSQM

Um dos primeiros métodos a gerar resultados que se aproximavam de avaliações MOS foi o método PSQM (*Perceptual Speech Quality Measurement*), que passou a ser conhecido como a recomendação P.861 do ITU-T, publicada em 1996. Rapidamente foi avaliado como um método de apuração consistente e acurado, que leva em consideração fatores de percepção humana [16, 22].

O PSQM é um processo matemático com capacidade de prever resultados dos testes subjetivos, em particular os obtidos com testes do padrão P.830. No entanto, retornam valores em uma escala diferente da adotada pelo MOS. O PSQM foi projetado para atender à banda utilizada pela telefonia convencional (300-3400 Hz). Assim, sinais nesta faixa são processados pelos codificadores de áudio e as distorções geradas por este processo são medidas de acordo com os fatores da percepção humana.

O esquema de funcionamento do modelo PSQM, conforme ilustrado na Figura 19, envolve a gravação de fala em mensagens utilizando as mesmas características específicas para testes MOS no padrão ITU P.830. Em seguida, esta mensagem é processada por um codificador e tanto o sinal original quanto o codificado são submetidos ao algoritmo de comparação do PSQM. Esse processo recebe ambos os sinais recebidos, e faz as comparações em pequenos segmentos sincronizados, levando

em consideração não apenas o espectro, como também a sensibilidade auditiva humana, como freqüência e *loudness*.

O valor resultante desta comparação varia de zero a infinito, representando a distância entre os sinais de entrada e saída dos codificadores. O valor zero representa sinais idênticos, sem nenhuma degradação. Quanto mais alto é o valor gerado pelo PSQM, maior é o nível de distorção. No entanto, na prática os valores máximos de PSQM variam entre 15 e 20 [4].

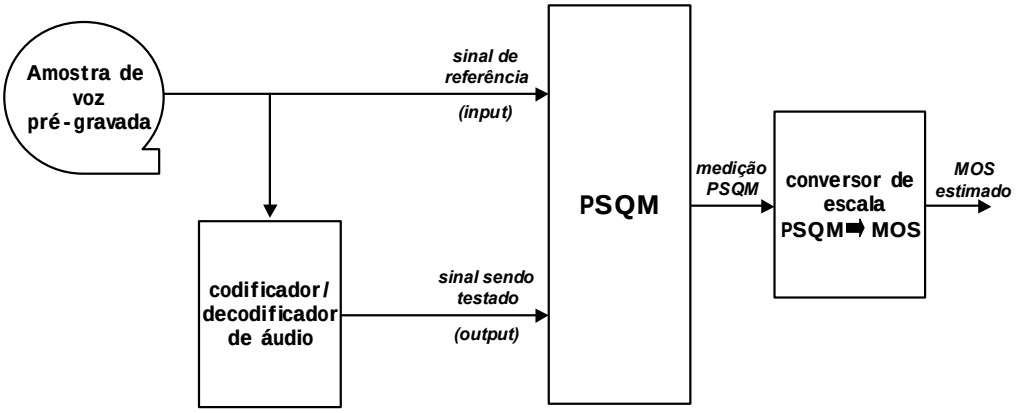


Figura 19. Esquema de funcionamento do modelo PSQM

O algoritmo do PSQM assume determinadas características nos fluxos sendo analisados, apresentadas a seguir.

a. Os sinais (*input* e *output*) são sincronizados. As comparações são feitas em segmentos (256 amostras para sinal amostrado em 8kHz). Caso o sincronismo não seja perfeito, resultados incorretos serão obtidos.

b. Devido ao sincronismo, atrasos não afetam o resultado. No entanto, grandes variações de atraso podem impedir o sincronismo dos sinais, acarretando em um avaliação incorreta.

c. Não podem existir ruídos de fundo no sinal de entrada.

d. Não pode haver degradação nos canais de comunicação, como erros de transmissão e perda de pacotes, pois o PSQM não mede o impacto das degradações causadas por estes fatores. Caso existam, PSQM tende a gerar

valores de qualidade superestimados, superiores aos que seriam obtidos por testes subjetivos.

Resumindo, o método PSQM apresenta bons resultados quando são avaliados fatores como o tipo de *codec* sendo utilizado e propriedades específicas da mensagem, tais como idioma. PSQM não se propõe a medir impacto de parâmetros como atraso, *jitter*, fontes simultâneas de mensagens, problemas de transmissão (perda de pacotes, por exemplo), ruídos de fundo ou música como fonte de entrada.

3.2.3. MNB

Proposto em 1997 e aprovado em 1998, foi adicionado o apêndice II no padrão P.861 (PSQM), baseado no relatório COM 12-24 E [23] do grupo de estudo 12 do ITU-T. O apêndice descreve MNB (*Measuring Normalizing Blocks*), uma técnica alternativa ao PSQM. O método MNB, por possuir um algoritmo diferente ao do PSQM, é mais adequado para medir o impacto na clareza da voz levando em consideração outros fatores como, por exemplo, erros nos canais de comunicação ou *codecs* com taxas de transmissão inferiores a 4 kbps. Tais fatores não são tratados pelo PSQM original.

O MNB pode ser considerado como uma técnica complementar ao algoritmo do PSQM [24].

3.2.4. PSQM+

O modelo PSQM foi rapidamente aceito como uma boa técnica para medir a qualidade e a clareza da voz, inclusive em redes VoIP. Uma das desvantagens do algoritmo PSQM é não levar em consideração fatores que causam distorção do sinal original, como a perda de pacotes, principal fator de degradação em redes VoIP. Conseqüentemente, o PSQM tende a reportar uma qualidade melhor que uma interpretação humana faria. O ITU-T publicou em 1997 o relatório COM 12-20E [25], que passou a ser conhecido como PSQM+.

O PSQM+ foi baseado no algoritmo padrão do PSQM com apenas algumas alterações, que o tornam sensível à falhas de maior porte como, por exemplo, um

aumento significativo no indicador de perda de pacotes ou na da variação de atraso. Desta forma, os resultados do PSQM+ são muito mais próximos aos do MOS obtidos em testes subjetivos [4].

3.2.5. PAMS

Pesquisas realizadas nos laboratórios da *KPN Research*, mesmo centro de pesquisa que desenvolveu originalmente o PSQM, resultaram em uma outra técnica, posteriormente aprimorada pela *British Telecommunications*, que ficou conhecida como PAMS (*Perceptual Analysis Measurement System*). A primeira versão do PAMS foi divulgada em agosto de 1998, sendo que a versão 3.0 foi liberada em fevereiro de 2000 [18]. Seu emprego começou inicialmente na Europa, mas cada vez mais tem se difundido como forma de avaliação [4].

O PAMS propõe um modelo alternativo ao PSQM+, embora ambos tenham o mesmo objetivo: prever o resultado de testes subjetivos através de testes objetivos em redes com distorções, causadas pelo processo de codificação ou deficiências no meio de transmissão.

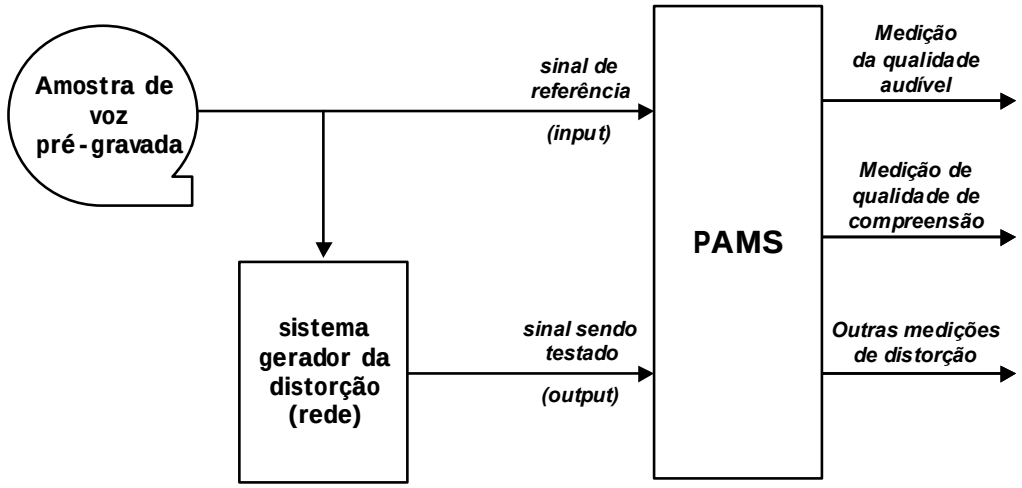


Figura 20. Esquema de funcionamento do modelo PAMS

Embora semelhante em alguns aspectos ao PSQM, o esquema de funcionamento do modelo PAMS, ilustrado na Figura 20, utiliza técnicas diferentes para processamento do sinal e simulação da percepção humana [18]. O sinal de entrada utilizado segue o

especificado no padrão P.830. PAMS é otimizado para sistemas de voz sintetizada, embora voz humana possa ser utilizada como entrada. O sinal de saída é gravado à medida que é recebido, fazendo com que os efeitos gerados pela variação de atraso sejam mantidos e refletidos no resultado final.

Outra diferença em relação ao PSQM é que a escala dos resultados do PAMS é idêntica à adotada pelo MOS, não necessitando de conversão de escalas para que possa ser efetuada uma análise comparativa entre os resultados gerados em testes utilizando PAMS e MOS. Os testes simulados pelo PAMS são equivalentes aos testes propostos pelo padrão MOS.

Basicamente, PAMS leva em consideração a distorção causada pelo codificador, a perda de pacotes e o *jitter*. PAMS assume que não existe ruído de fundo no sinal de referência e este sinal deve ter um nível constante de volume.

PAMS não é adequado para medir o impacto gerado pelo atraso, a variação lenta do atraso, o ruído de fundo no sinal de referência (embora sirva para medir o impacto de ruído de fundo adicionado ao sinal de saída) e utilização de música como sinal de entrada.

3.2.6. PESQ

O modelo PSQM foi novamente aprimorado pela equipe do *KPN Research* em 1999, resultando no método conhecido como PSQM99. No período 1998-2000, o ITU recebeu cinco submissões de propostas para novos modelos de avaliação objetiva de qualidade de voz. Após a avaliação e comparação dos modelos propostos, concluiu-se que o PSQM99 e o PAMS apresentavam resultados equivalentemente eficientes. Foi constatado que cada um dos modelos possuía propriedades importantes e que a indústria, de um modo geral, iria se beneficiar da combinação das melhores características de cada um. Um novo projeto, desenvolvido em colaboração da *British Telecommunications* e da *KPN Research*, foi submetido ao ITU em maio de 2000, dando origem ao modelo PESQ (*Perceptual Evaluation of Speech Quality*), passando a ser o padrão P.862 [18] em 2001. A tendência é que o PESQ passe a ser o padrão de testes adotado internacionalmente [4].

Assim como PSQM e PAMS, PESQ ainda é direcionado para a banda de sinais utilizada pela telefonia. Seu uso é adequado para todos os tipos de *codecs*, bem como avaliar atraso variável, perda de pacotes ou células e erros em canais de transmissão. PESQ combina a consideração dada a variação do atraso, característica do PAMS, com a apurada modelagem da percepção humana do PSQM99 [4].

3.3. Comparação dos Modelos

Diversos trabalhos [25, 26, 27] foram feitos para comparar os resultados obtidos por cada modelo. O relatório COM 12-20-E [25] compara PSQM com PSQM+, onde, de um modo geral, PSQM+ se mostra superior. O relatório COM 12-D80-E [26] compara PAMS, PSQM, PSQM+ e MNB, com 16000 amostras e 30 testes subjetivos realizados, sob diversas implementações e condições distintas. De um modo geral, PAMS apresentou os melhores resultados. Quando ruído de fundo é adicionado, PSQM+ se mostrou superior. O PSQM+ foi mais bem avaliado do que o PSQM em quase todos os testes. Resultado de estudo realizado pelo NTIA (*National Telecommunications and Information Administration*), o relatório 98-347 [27] compara diversas métricas utilizadas em testes com os resultados gerados pelo PSQM e MNB, mostrando a sua proximidade com os resultados obtidos com MOS.

3.4. E-Model

No entanto, todos os modelos mostrados anteriormente não são adequados em uma avaliação em tempo real e normalmente usados em ferramentas de monitoração ativa [64]. Portanto, surge a necessidade de um modelo que utilize somente fatores instantâneos da rede como, por exemplo, taxa de perda de pacotes e *jitter*.

O E-Model foi originalmente proposto pelo ETSI através do relatório ETR250 [28], e depois transformado posteriormente no padrão ITU como G.107 [19]. O objetivo do modelo é determinar a qualidade de uma transmissão através da análise de parâmetros coletados como atrasos e perdas.

O resultado do modelo é o valor R (*R-factor*), que varia em uma escala de 0 a 100. Pode-se relacionar o valor obtido com o *R-factor* e o MOS, como mostra a Figura 21.

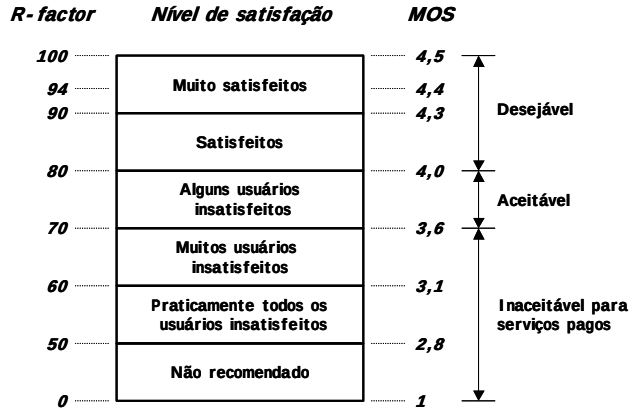


Figura 21. Relação entre *R-factor* e MOS

Valores abaixo de 50 são considerados inaceitáveis. Como as ligações telefônicas típicas não ultrapassam o valor 94, em termos práticos podemos considerar o intervalo real do *R-factor* entre 50-94 [3, 4]. A telefonia convencional obtém um MOS no pior caso de 3.6, sendo que a meta a ser alcançada é um MOS superior a 4,0 [4].

O *R-factor* é calculado a partir da seguinte fórmula [19]:

$$R = (R_O - I_S) - I_d - I_e + A$$

Os valores $\boldsymbol{R_O}$ (relação entre sinal e ruído) e $\boldsymbol{I_S}$ (degradação decorrente da quantização da voz) são intrínsecos do próprio sinal de voz e não dependem do meio de transmissão, conseqüentemente são considerados irrelevantes quando se compara telefonia convencional com voz sobre IP [4]. $\boldsymbol{I_d}$ é o índice de degradação no sinal causado pelo atraso e $\boldsymbol{I_e}$ representa a degradação causada pela distorção no sinal devido aos *codecs*, perda de pacotes, e *buffers* de reprodução e compensação de *jitter*. O valor A (*advantage factor*) representa a expectativa dos usuários em relação ao serviço, ou seja, o quanto o usuário está disposto a aceitar um pouco de degradação da qualidade de voz em contrapartida a facilidades ou vantagens de seu uso. Por exemplo, usuários normalmente toleram uma qualidade de ligação inferior quando utilizam telefonia

celular. Para efeito de comparação com PSTN, este fator também é ignorado [4]. I_e utiliza valores indicativos de qualidade associados a cada *codec* e a uma taxa de perda de pacotes.

Uma boa comparação do comportamento do E-Model com outros modelos pode ser obtida em [29].

A Tabela 6 mostra valores de I_e para alguns codecs, assumindo não haver perda de pacotes [4].

Tabela 6. Valores de Ie para alguns codecs sem perda da de pacotes

Padrão	Tipo do Codec	Taxa de transmissão (Kbps)	I_e (Perda = 0)	*R-factor*
G.711	*PCM*	*64*	*0*	*94,3*
G.729	*CS-ACELP*	*8*	*10*	*84,3*
G.723.1	*ACELP*	*5,3*	*19*	*75,3*
G.723.1	*MP-MLQ*	*6,3*	*15*	*79,3*

O E-Model é o modelo atualmente usado pelos provedores de telefonia como ferramenta de apoio para planejamento de capacidade e monitoração de performance dos seus serviços [4].

3.5. Extended E-Model

O E-Model é uma maneira simples e direta de se calcular o MOS de forma passiva em redes VoIP. No entanto, o E-Model leva em consideração apenas fatores como atraso e perda de pacotes (I_d e I_e). O E-Model não considera a distribuição destes fatores durante a ligação, nem aspectos humanos encontrados em avaliações MOS, como o efeito memória. O Extended E-Model [30] é uma extensão proposta ao modelo E-Model, contemplando também estes fatores.

Os resultados a seguir foram obtidos pela empresa Telchemy [20], comparando o Extended E-Model com outros modelos. Nos testes de comparação do Extended E-

Model com PSQM e PAMS, este apresentou a melhor performance em cinco dos seis testes realizados [31].

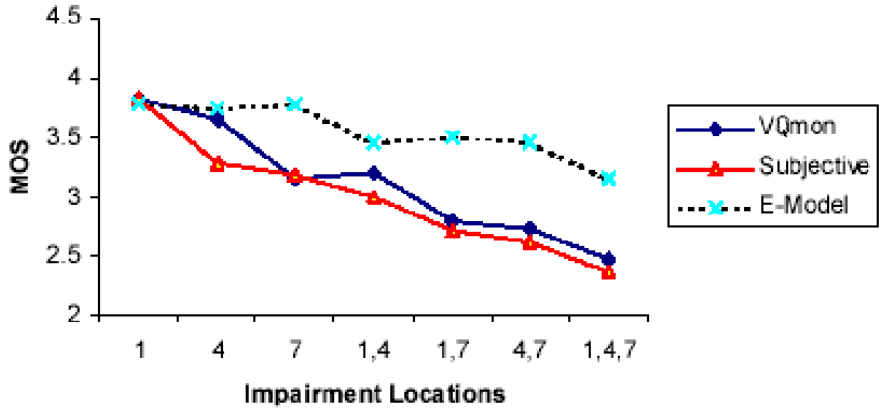


Figura 22. Comparação E-model X Extended E-model X MOS

A Figura 22 ilustra a comparação do Extended E-Model, implementado através do software VQmon, da própria Telchemy, com o E-Model padrão [32]. Como pode ser observado, os valores estimados pelo Extended E-Model são muito mais próximos aos do MOS do que os valores calculados através do E-Model.

Portanto, uma ferramenta de avaliação de qualidade de voz deve levar em consideração não apenas os fatores que causam a degradação, mas também a sua distribuição ao decorrer de toda a chamada. A coleta de todas estas informações é essencial para uma análise mais correta e precisa.

Capítulo 4

Implementação de Ferramenta de Monitoração Ativa

As ferramentas ativas, como já visto, geram o seu próprio tráfego. No caso específico do serviço VoIP no *backbone* da RNP, que é um novo produto a ser oferecido para as suas instituições usuárias, era necessário traçar um perfil do comportamento do *backbone* em relação ao tráfego de voz, apesar da inexistência de tal tráfego real para realizar as medições e avaliação. Para tanto, foi desenvolvida uma ferramenta, que além de traçar o comportamento padrão do *backbone*, permite gerar uma massa de dados representativa e consistente do tráfego VoIP para que outras ferramentas de análise possam avaliar a qualidade das conexões bem como determinar os recursos necessários quando o serviço entrar realmente em produção. Para a geração desta massa de dados, foram coletadas informações de conexões realizadas para diversos pontos do país, utilizando diferentes *codecs*, geradas pela própria ferramenta durante um período de aproximadamente dois meses.

A mesma massa de dados, refletindo as características do *backbone* da RNP atual, será utilizada para uma análise comparativa com os dados obtidos através de um novo processo de coleta na segunda etapa do projeto VoIP da RNP, quando regras de QoS serão aplicadas ao *backbone* priorizando o tráfego de voz. Desta forma, será possível fazer a avaliação da efetividade das regras de priorização do tráfego VoIP no *backbone*.

Também é objetivo desse trabalho coletar a qualidade de voz obtida por um mesmo usuário, apenas alterando determinados parâmetros das conexões como, por exemplo, *codecs*, tamanho de *buffers*, tamanho das amostras de voz, entre outros.

A organização deste capítulo é descrita a seguir. Inicialmente é apresentada a estrutura da ferramenta, seus principais módulos e componentes. Em seguida são apresentados os detalhes de implementação, as estruturas internas e detalhes de programação. Alguns problemas encontrados durante o desenvolvimento e as soluções

adotadas são descritos. A interface *web* é apresentada, e alguns exemplos de dados reais coletados são apresentados.

4.1. Estrutura da Ferramenta

As estatísticas são geradas após a execução de programas em etapas. A primeira etapa consiste na geração de fluxos de mídia, sua transmissão pela rede, sua recepção e a armazenagem dos dados. Na segunda etapa, são agregados todos os dados coletados em um único banco de dados, onde as informações são processadas e consolidadas. A terceira etapa consiste na visualização dos dados já pré-processados, permitindo a análise subjetiva da qualidade das transmissões, bem como a detecção e correlação de prováveis causas dos problemas encontrados. Não faz parte do escopo desta ferramenta o desenvolvimento de modelos de avaliação, como o E-Model, de forma a efetuar a análise de maneira quantitativa. Tal análise deve ser gerada por outros aplicativos específicos para este fim, a partir dos dados coletados pela ferramenta.

4.2. Geração das Chamadas e Armazenamento de Dados

Uma das premissas do projeto era que a ferramenta fosse baseada em software livre, permitindo o seu uso pelas instituições acadêmicas sem ônus. Com intuito de facilitar a compreensão, essa sessão foi divida em três etapas: programa receptor de fluxo, programa gerador de fluxo e armazenamento dos dados.

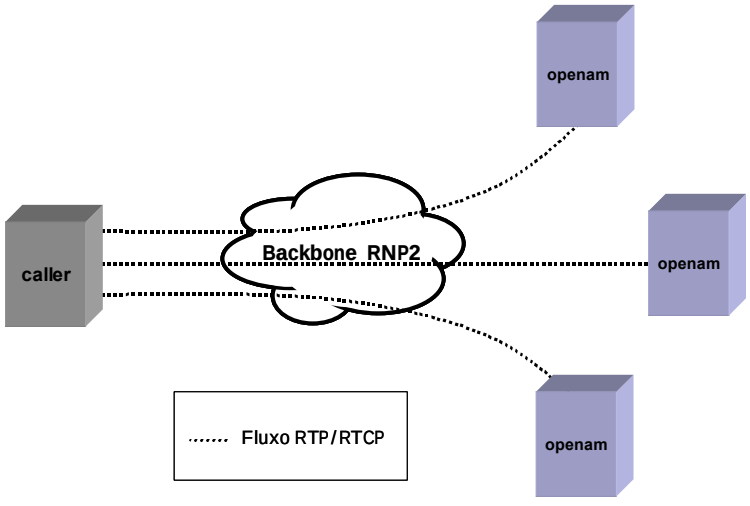


Figura 23. Geração das Chamadas e Armazenamento de dados

4.2.1. Programa Receptor de Fluxo

O software utilizado para a primeira etapa foi baseado em programas já desenvolvidos, utilizando a biblioteca OpenH323, de código aberto, disponíveis no próprio site do projeto OpenH323 [57]. Foram desenvolvidos dois utilitários para o projeto: um receptor de fluxos e outro gerador.

O programa receptor de fluxos, responsável pelo recebimento de conexões VoIP e pelo armazenamento da mensagem de voz recebida, além de guardar toda a sinalização da chamada, foi baseado no aplicativo *openam* (*Open Answering Machine*) que implementa uma secretária eletrônica VoIP. A própria semelhança das funções desempenhadas pelo *openam* com as necessárias para implementar o programa receptor foram determinantes na escolha do software a ser adotado. O programa receptor deveria aguardar por uma conexão VoIP e gravar a mensagem de áudio recebida para análise posterior. Este programa também deveria armazenar em arquivos de *logs* todos os pacotes recebidos de sinalização, para estudo e compreensão do funcionamento do protocolo H.323, além de também armazenar todos os pacotes RTP e RTCP (principalmente do tipo SR e RR), gerados durante o fluxo recebido. A partir dos dados obtidos nos pacotes RTCP, tipo SR e RR, é feita a análise dos parâmetros de qualidade de voz.

Como o próprio programa *openam* já implementa todas estas funções, foi um processo simples adaptá-lo para a função do programa receptor, não sendo necessárias grandes alterações no seu código. A própria estrutura da biblioteca OpenH323 facilita muito a implementação deste tipo de utilitário.

O programa *openam* foi desenvolvido na linguagem de programação C++, chamando objetos oferecidos pela biblioteca, como a própria sinalização do H.323 (SETUP, CONNECT, H.245), a pilha do protocolo RTP/RTCP, além dos principais *codecs* utilizados, como o G723.1, G.711 (*a-law e υ-law*), GSM, MS-GSM e LPC-10.

Os objetos utilizados pelo software *openam* e como estes se relacionam são mostrados na Figura 24.

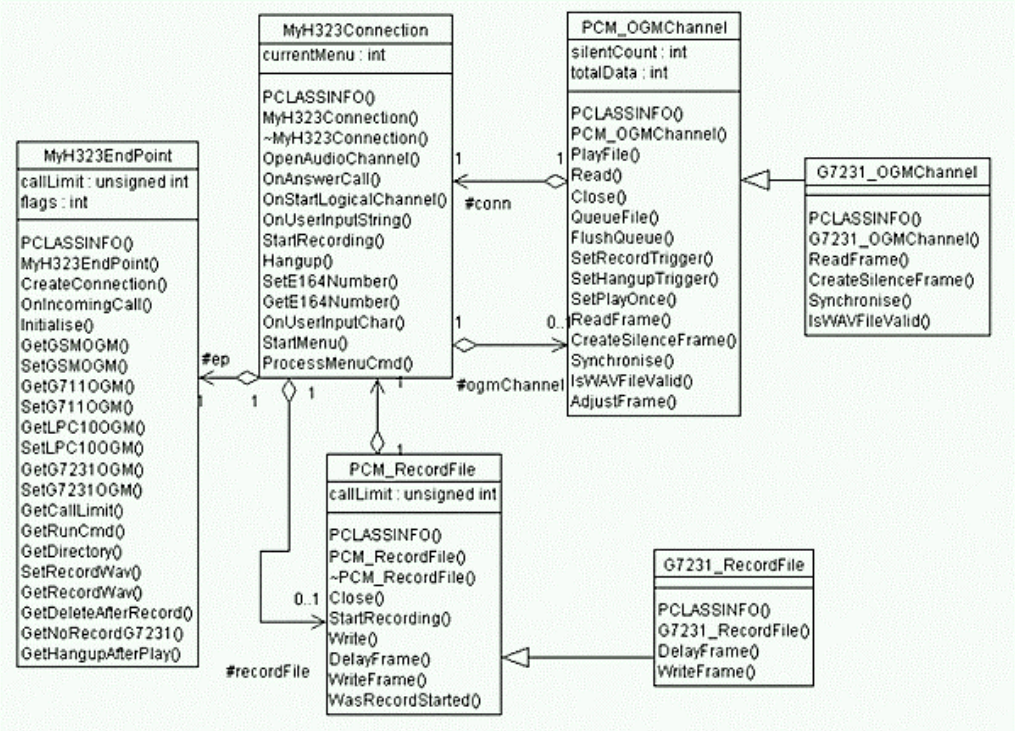


Figura 24. Diagrama de Classes do Openam

O funcionamento do *openam* é descrito a seguir. Inicialmente, o programa fica pronto para receber conexões de SETUP, através do recebimento de pacotes de sinalização do H.225, utilizando a porta 1720 do TCP. Ao receber um SETUP, a secretária eletrônica faz todo o *handshake* H.323 para sincronização e negociação dos *codecs* a serem utilizados, bem como quais portas serão utilizadas para a troca de mensagens RTP e RTCP. Uma vez realizada toda a negociação e inicialização de todas as instâncias dos objetos envolvidos, o *openam* atua como uma secretária eletrônica convencional. Inicialmente, reproduz uma mensagem pré-gravada do tipo "você ligou para... deixe seu recado", e logo após passa a gravar todos os pacotes de voz recebidos, até o término da conexão por parte do chamador ou então até ser atingido um tempo máximo configurado para a gravação da mensagem.

Ao receber a conexão de outro programa H.323, o objeto #ep (instância de MyH323Endpoint) é acionado pelo evento *MyH.323Endpoint::OnIncomingCall*, e que, por sua vez, cria o objeto *MyH323Connection*, relacionado especificamente àquela ligação. Esta capacidade permite ao *openam* receber ligações simultâneas, separando-as, respectivamente, como *threads* e objetos concorrentes para cada ligação. Esta é uma

característica importante, pois permite que sejam feitos testes com múltiplas conexões VOIP simultaneamente. O objeto *#conn*, instância de *MyH323Connection*, cria dois objetos internos, responsáveis por reproduzir e gravar mensagens de áudio. O objeto para reprodução de áudio é o *#ogmchannel*, instância de PCM ou *G7231OGMChannel*. O OGM da sigla dos objetos indica *outgoing message*, ou mensagem de boas vindas, associado a um arquivo de áudio pré-gravado a ser reproduzido após o estabelecimento da chamada. O objeto para gravação do áudio é *#recordFile*, instância de PCM ou *G7231RecordFile*, que gera um arquivo (cujo nome é formado pelo horário de sistema no instante de início da gravação) contendo a mensagem sendo recebida através do fluxo RTP. Ambos os objetos precisam de métodos de sincronização de tempo, como o método *PCM_OGMChannel::Synchronise*, que vai inserindo atrasos caso seja necessário em cada leitura de um frame de áudio, obtido do arquivo pré-gravado e armazenado temporariamente no buffer de envio até o seu encapsulamento RTP. A operação contrária é bastante similar. Na gravação do áudio recebido, o método *PCM_RecordFile::DelayFrame* vai recompondo o compasso original dos *frames* de áudio que estão armazenados no buffer de recepção ou *playout*.

Portanto, devido a esta bufferização e sincronismo, não é possível perceber a degradação na ligação gerada pelo atraso e pela sua variação, ao se reproduzir posteriormente arquivo de áudio gerado pelo *openam*. No entanto, fatores que causam degradação na qualidade como perda de pacotes, descarte de pacotes atrasados, *codecs* utilizados na ligação, podem ser percebidos durante a sua reprodução.

4.2.2. Programa Gerador de Fluxo

O segundo aplicativo utilizado durante a fase de geração e coleta dos dados é o programa gerador das chamadas que serão avaliadas. Este programa, cujo nome adotado foi *caller*, foi baseado na idéia de outra implementação chamada *callgenerator* [46] que, por sua vez, foi baseado na estrutura do código-fonte do próprio *openam*. O programa *caller* pode realizar até 254 chamadas IP simultâneas, através de *threads* concorrentes, sem precisar ter nenhum recurso de hardware especial de áudio como placa de som ou microfone. A principal diferença do *callgenerator* em relação ao *openam* é o fato do primeiro iniciar a conexão, ao invés de permanecer aguardando por uma ligação, como

faz uma secretária eletrônica padrão. No entanto, foi necessário fazer diversas alterações no programa *caller* a partir do código do programa *openam*, para atender as nossas necessidades, descritas a seguir.

- Substituição da chamada inicial ao mecanismo *listenerH323* pelo procedimento de chamada ao método *MakeOutgoingCall*. O endereço IP a ser chamado por este método passou a ser um dos parâmetros de execução do programa *caller*. Outros métodos utilizados para estabelecimento de conexões H.323 foram inseridos, como, *OnConnectionEstablished*, *OnConnectionCleared*, *OnAlerting* e *AwaitTermination*, além do próprio método *MakeOutgoingCall*, que não fazia parte do código original do *openam*.

- Atualização no código do método *OnConnectionEstablished* (tanto do *caller* quanto do *openam*), visando alterar a freqüência em que pacotes RTCP são enviados. Na implementação original, pacotes RTCP são enviados a cada 12 segundos, passando agora a enviar a cada segundo. Este intervalo de tempo não é determinado na RFC do RTP [5], cabendo a cada implementação decidir qual a melhor relação informação X *overhead* e qual a granularidade desejada. Por exemplo, o programa NetMeeting da Microsoft utiliza o valor de 5 segundos para intervalo entre pacotes RTCP. A adoção do período de um segundo foi resultante da granularidade mais adequada necessária para nossos testes, sem haver impacto na performance. Valores inferiores a um segundo causaram inclusive o travamento do programa.

- Foi acrescentado um novo parâmetro de execução (--*dejitter*), tanto no *caller* quanto no *openam*, permitindo especificar o tamanho do *buffer* de compensação de *jitter* dos aplicativos, podendo variar de 20 ms até 10 s.

Foram necessárias algumas alterações feitas diretamente no código fonte da biblioteca H.323.

- No código de implementação da *stack* RTP/RTCP é feito o cálculo da variação de *jitter*. Um dos campos utilizados na fórmula do cálculo é a função D, que representa o intervalo de tempo ocorrido entre dois pacotes RTP consecutivos.

No entanto, este valor somente era utilizado para efeito de cálculo do *jitter* suavizado, sendo desprezado logo em seguida. Como este valor representa o *jitter* instantâneo em um dado momento da conexão, ele é muito importante para a estimativa da qualidade da voz. Foi criada a variável *jitterinstant* para armazenar o valor do *jitter* instantâneo.

- Toda chegada de pacote RTP gera um evento da biblioteca, onde é gravado um registro de *log* com as informações contidas nos pacotes, além de outras informações calculadas no instante de recepção. Foi inserida neste registro de *log* a variável recém-criada *jitterinstant*.

- Inclusão nos registros de *log* de informações sobre o número de pacotes perdidos até o momento, permitindo o conhecimento posterior da relação de perda de pacotes com a sua distribuição no tempo durante a ligação.

Como os dois programas foram baseados na estrutura de implementação de uma secretária eletrônica, ambos tem um comportamento similar. Primeiro reproduzem a mensagem armazenada e depois entram em estado de recepção e gravação. Sua principal diferença é que o *openam* aguarda o estabelecimento de uma conexão, enquanto o *caller* inicia o estabelecimento da mesma. Desta forma, após a conexão realizada entre o *caller* e o *openam*, a máquina de estados dos dois programas estaria na fase de reprodução de mensagem armazenada ou *play OGM*, "falando" ao mesmo tempo. Após a reprodução de todo o arquivo OGM, ambas as máquinas mudariam para o estado *Record file*.

A solução para sincronizar o momento de fala e escuta de cada uma foi utilizando uma característica da biblioteca OpenH323, que é sua capacidade de detectar sinais de DTMF dentro da banda de áudio. Tal característica é fundamental para a implementação de serviços no qual o usuário interage com o aplicativo, através da discagem de números no seu aparelho. Através da observação da documentação e do código fonte do *openam*, foi verificado que o *openam* já utiliza esta característica de detecção de DTMF. Ao receber um DTMF 4, o *openam* interrompe a reprodução da mensagem pré-gravada e entra imediatamente em modo de armazenamento de mensagem. Aproveitando esta peculiaridade, foi inserido um sinal de DTMF 4 no início

da mensagem reproduzida pelo programa *caller*, fazendo com que o programa *openam* do lado receptor mudasse imediatamente do estado *play OGM* para *Record file* ao receber o sinal DTMF 4, passando a armazenar o restante da mensagem sendo recebida.

Após um período também informado ao programa através de parâmetro de execução do *openam*, a conexão é desfeita pelo lado receptor. O programa *caller* não chega a reproduzir toda a mensagem original de forma exaustiva antes de receber a sinalização de término da chamada enviada pelo *openam*, nem chegando, eventualmente, a entrar em modo de gravação de mensagens recebidas, como pode ser observado na Figura 25.

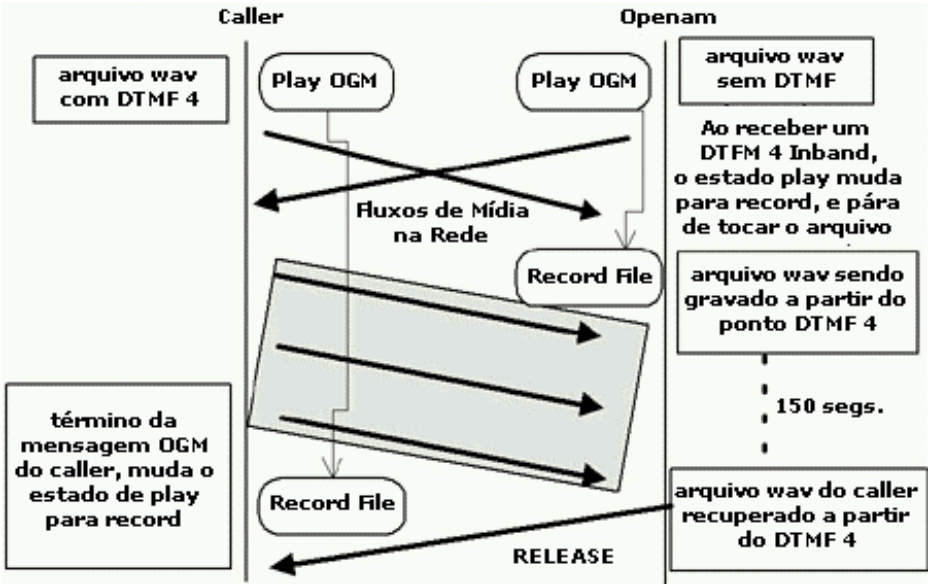


Figura 25. Estado *Play OGM x Record File* nos programas *caller* e *openam*

A gravação da mensagem original enviada pelo *caller* passa a ser um arquivo de referência, que posteriormente pode ser comparada com a mensagem recebida pelo *openam*, após ter sofrido a degradação causada pela própria transmissão na rede e pelas características dos *codecs* utilizados. A mensagem de referência é um arquivo com a extensão ".wav", contendo a voz de um locutor fazendo uma contagem pausada entre os números, de modo a existirem períodos de silêncio bem definidos intercalados com períodos de fala. Outro fator para a escolha desta mensagem é que o intervalo entre os números foi de aproximadamente um segundo, facilitando a rápida localização de um determinado trecho da mensagem.

4.2.3. Armazenamento dos Dados

Os programas *caller* ou *openam* geram a cada ligação um extenso arquivo de texto, contendo os *logs* com todas as medidas realizadas (perdas instantâneas, perdas acumuladas, RTT, diferença entre chegadas, *jitter* suavizado RTCP, entre outras). Além disto, o programa *openam* armazena um arquivo contendo o áudio de como a mensagem foi recebida.

Os nomes dos arquivos gerados são suficientes para identificar a hora e as instituições envolvidas na ligação, através da seguinte regra de formação:

<origem>-<destino>-<L|R>-<ano>-<mês>-<dia>-<hora>-<min>-<segundo>.txt

onde *origem* e *destino* identificam os locais onde estão sendo executados os programas de coleta, e os campos *dia*, *mês*, *ano*, *hora*, *minuto* e *segundo* informam o momento em que foi iniciada a conexão, sendo extraídos do campo NTP do primeiro pacote RTP do fluxo. Por exemplo, um arquivo com o nome `nceufrj-brasilia-R-2003-02-05-11-30-22.txt` teria as informações de um fluxo gerado às 11:30:22 de 05/02/2003, com *caller* sendo executado na UFRJ e o *openam* em Brasília. O "R" indica que o arquivo foi gerado pelo ponto remoto, refletindo a visão da conexão sob o seu ponto de vista, enquanto um "L" indicaria a geração local, feita pelo chamador, em relação à mesma conexão.

As informações contidas nos arquivos de *log* mostram todos os pacotes de sinalização H.323, pacotes RTP e RTCP. Para esta ferramenta específica, determinados registros são fundamentais para a avaliação da qualidade das conexões.

Um pequeno trecho de um arquivo de *log* contendo o registro de pacotes RTP pode ser visualizado a seguir.

0:12.284 RTP Jitter:864de88 RTP Receive statistics: packets=396 octets=95040 **lost=0** jitterInstant=0 jitter=10

0:12.311 RTP Jitter:864de88 RTP Receive statistics: packets=397 octets=95280 lost=0 **jitterInstant=24** jitter=9

0:12.353 RTP Jitter:864de88 RTP Receive statistics: packets=398 octets=95520 lost=0 jitterInstant=120 **jitter=10**

A cada evento de recepção de um pacote RTP, um registro é gravado no *log*. Além do momento de recepção do pacote, medidas instantâneas são obtidas, através das variáveis *lost*, *jitterInstant* e *jitter*. A variável *lost* informa quantos pacotes foram perdidos desde o último pacote RTP. A variável *jitterInstant* mostra o valor D da equação do *jitter*, incluído no log após a alteração feita no código fonte. A variável *jitter* representa o valor suavizado, conforme descrito na RFC RTP [5], mostrando as tendências de congestionamento.

Outros registros fundamentais dos arquivos de *logs* são os que mostram os pacotes RTCP. No caso do *caller* e *openam*, eles são pacotes compostos por um SR, um RR e um SDES. Pode-se observar que o instante de recepção do pacote é o mesmo para os três relatórios, pois fazem parte de um pacote RTCP composto.

```
00:12.374 RTP Jitter:864de88 RTP OnRxSenderReport: ssrc=1803467860
                ntp=2002/10/24-10:12:59.610288 rtp=95520 psent=398 osent=95520

0:12.374 RTP Jitter:864de88 RTP OnRxSenderReport RR: ssrc=2232852623 fraction=0
                lost=0 last_seq=0 jitter=39 lsr=0.000 dlsr=0.000

0:12.374 RTPJitter:864de88 OnSourceDescription: ssrc=1803467860 item[0]:
                type=CNAME data="voip@server3.pop-df.rnp.br" item[1]:
                type=TOOL data="OpenAM"
```

No exemplo de registro de *log* acima, vemos a ocorrência de um pacote RTCP. Dele podem ser extraídos os campos do relatório emissor (SR) como *ssrc* (identificação da fonte), *ntp* (tempo absoluto em formato NTP da máquina emissora), o *timestamp* RTP do pacote SR, número de pacotes enviados pelo emissor até o momento (*psent*), entre outras. Também no mesmo pacote, podemos verificar um relatório de receptor (RR) da outra fonte identificado por outro número de *ssrc*, além dos valores da fração de perda (*fraction*), e do valor acumulado do número de pacotes perdidos (*lost*).

Uma observação importante é que os valores de LSR (*Last Sender Report*) e DLSR (*Delay Since Last Report*), usados para calcular o *Round-Trip Time* (RTT), apresentam o valor 0, inviabilizando o cálculo do RTT pelo processo previsto na RFC [5]. O cálculo do RTT implementado pela ferramenta é descrito na seção 4.3.1.

Após a fase de geração e coleta de dados, com a criação de arquivos contendo as mensagens armazenadas e toda a sinalização utilizada, é necessário fazer a transferência destes arquivos para um servidor para posterior consolidação.

4.3. Consolidação dos Dados

Execuções consecutivas do *caller* para diversos pontos remotos executando *openam* geram arquivos de *log* referentes a cada fluxo criado, contendo todas as informações relativas à qualidade obtida em cada ligação. Além dos arquivos de *log*, o arquivo .wav contendo a própria ligação, como vista pelo receptor, também fica armazenado nos pontos remotos.

Para permitir a consolidação dos dados, conforme ilustrado na Figura 26, todos os arquivos remotos são transferidos para um equipamento central, localizado no laboratório VoIP do NCE/UFRJ.

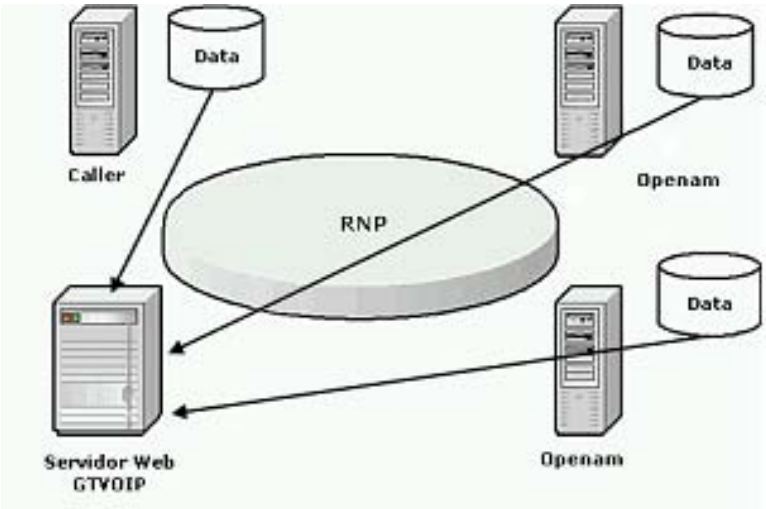


Figura 26. Transferência e consolidação dos dados

Uma grande capacidade de armazenamento é necessária, pois os arquivos de *logs* gerados possuem em média cerca de 1.2 Mb cada (sem compressão) e cada arquivo

de áudio cresce proporcionalmente com a duração de cada conexão. A título de ilustração, um arquivo contendo 150 segundos de gravação ocupa aproximadamente 2,5 Mb. Portanto, cada conexão efetuada necessita aproximadamente 5 Mb (1,2 Mb arquivo de *log* local + 1,2 Mb arquivo *log* remoto + 2,5 Mb do arquivo .wav remoto). Uma vez que sempre era utilizado o mesmo arquivo de referência (.wav local), não foi necessário armazená-lo repetidas vezes.

Como visto, os arquivos de *log* são gerados em formato texto, contendo de forma descritiva todos os pacotes RTP/RTCP referentes à ligação. É importante frisar que para cada ligação existem dois arquivos de *log* (local e remoto), refletindo a qualidade da ligação sob o ponto de vista de cada extremidade.

Com as alterações feitas nos programas *caller* e *openam* e na própria biblioteca OpenH323, os *logs* contêm todas as informações desejadas, com geração de pacotes RTCP a cada segundo. A partir dos registros nos *logs* referentes aos pacotes RTP é possível obter as seguintes informações: perdas instantâneas, perdas acumuladas, *jitter* instantâneo e *jitter* suavizado. Além dessas informações, pode-se extrair outros dados a partir dos pacotes RTCP, como o *fraction lost*, o último número de seqüência recebido, número de pacotes enviados, bem como o tempo absoluto da máquina em formato NTP e o *timestamp* do próprio pacote SR.

Somente as variáveis relevantes são extraídas dos arquivos de *log*, gerando novos arquivos de tamanho reduzido, contendo somente a informação essencial para a análise e visualização dos dados diária. Arquivos com o resumo diário informam os valores mínimo, médio, máximo e desvio padrão das variáveis RTT, *jitter* e perdas relativos a cada ligação. Possuem um tamanho muito reduzido, de aproximadamente 20 Kbytes. Desta forma, o processamento posterior se torna muito mais rápido e eficiente, além de utilizar o espaço em disco de forma mais racional. Os arquivos de *log* originais são então armazenados, para futura reavaliação ou extração de novos dados, se necessário.

4.3.1. Cálculo do RTT

Como as variáveis LSR e DLSR não são preenchidas pela biblioteca OpenH323, o cálculo do RTT deveria ser feito de alguma outra forma. A primeira solução implementada foi disparar um processo *ping* em paralelo ao estabelecimento das chamadas entre *caller* e *openam*, com a sua saída sendo armazenada para posterior consolidação. Esta solução não era a mais adequada, pois não utilizava pacotes do próprio fluxo, além de utilizar um processo externo. O programa *ping*, implementado em praticamente todos sistemas operacionais, utiliza pacotes ICMP do tipo *Echo Request* e *Echo Reply*. Pacotes ICMP podem possuir tratamento diferenciado na rede, por parte dos roteadores intermediários, ou mesmo pelo próprio sistema operacional onde os processos *caller* e *openam* estão sendo executados. Desta forma, uma margem de incerteza era inserida nas medições de RTT. A solução adotada deveria obter o RTT a partir dos próprios arquivos de *logs* para não introduzir erros. O cálculo passou a ser feito a partir dos próprios pacotes RTCP, mesmo sem ter os registros de DLSR e LSR.

O procedimento adotado para cálculo do RTT é descrito em seguida. Após a transferência dos arquivos de *logs* de uma conexão (local e remoto), lê-se o arquivo local seqüencialmente. Ao se encontrar um evento de envio de pacote RTCP, seja referente a um pacote tipo SR ou RR, armazena-se o *timestamp* do instante do envio (T1). O evento de recepção deste pacote é então procurado no arquivo remoto e o momento da recepção armazenado (T2). A partir do registro no *log* remoto referente à T2, é procurado seqüencialmente no próprio arquivo o envio do próximo pacote RTCP do tipo SR ou RR, obtendo-se assim o instante T3. O próximo passo é encontrar a recepção deste pacote enviado no instante T3 no arquivo local, representando o instante T4. O RTT pode ser então calculado como sendo o resultado da seguinte fórmula:

$$RTT = (T4 – T1) – (T3 – T2)$$

Esta fórmula é equivalente ao cálculo que seria feito caso os campos LSR e DLSR fossem preenchidos. Caso a recepção de um pacote RTCP enviado não seja localizado no log de recepção, o pacote é desprezado e volta-se ao primeiro passo. É válido ressaltar que não é necessário haver sincronismo de tempo entre as máquinas, pois os tempos são calculados de forma relativa dentro de uma mesma máquina. T1e T4

utilizam o relógio do emissor e, de forma análoga, T2 e T3 utilizam o relógio do receptor, como ilustrado na Figura 27.

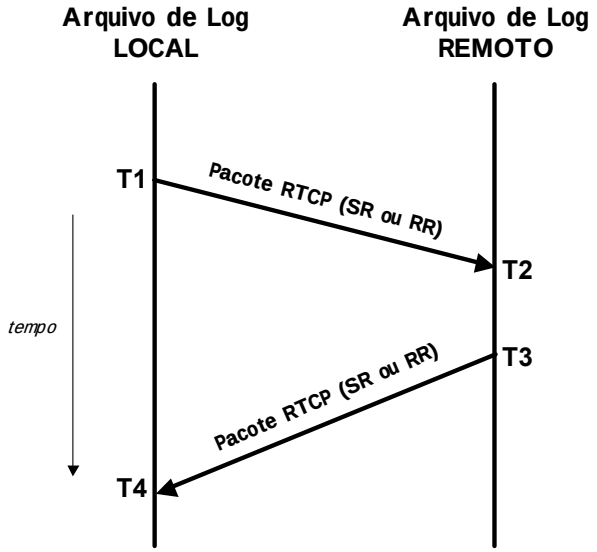


Figura 27. Cálculo de RTT a partir de pacotes RTCP

O valor resultante do cálculo do RTT é associado ao instante T4 da conexão. Todos os RTTs calculados e seus T4 correspondentes são armazenados em um arquivo exclusivo para cada ligação. A regra de formação de nome do arquivo é a mesma utilizada para *logs*, exceto que com a extensão ".rtt".

Durante a fase de execução, alguns resultados inesperados surgiram no cálculo do RTT. Em determinados pontos remotos, esporadicamente o cálculo do RTT resultava em valores negativos. Após a revisão do código implementado para o cálculo do RTT, concluiu-se que o problema deveria estar relacionado a alguma peculiaridade destes pontos remotos. Após um período de pesquisa e investigação, ao se verificar o ambiente operacional no qual o *openam* era executado, foi detectado que o deamon *ntpd* estava instalado, responsável pelo ajuste constante da hora da máquina com algum servidor NTP. O *ntpd*, ao sincronizar o relógio da máquina, gerava distorções que acarretavam no erro de cálculo do RTT. Quando o ajuste de relógio do receptor ocorria exatamente entre os instantes T2 e T3, dependendo do grau da variação do *clock* naquele momento, a diferença entre T3 e T2 era superior a diferença entre T4 e T1, resultando em RTT negativo. Uma vez desativado o *ntpd*, o problema parou de ocorrer. A variação dos

clocks de cada máquina não influencia as medições, pois o intervalo de tempo entre eventos é muito pequeno, tornando desprezível a variação do *clock* neste período.

Na realidade, o cálculo do RTT pode ser feito sob o ponto de vista do chamador e do receptor, mas sempre geram resultados muito semelhantes, como era de se esperar.

4.3.2. Análise e Correlação dos Resultados com Dados Externos

Ao se analisar dados coletados durante a fase de testes, em diversas ligações eram percebidas perdas de pacote ou então, aumento repentino do RTT, o que justificava a degradação provável que ocorria por conseqüência destas falhas. No entanto, não era possível, em uma análise posterior ao momento da conexão, tentar relacionar estas alterações de RTT ou perda de pacotes com algum evento ocorrido rede durante o período da coleta. Visando permitir esta avaliação, um *traceroute* é disparado simultaneamente a ativação de uma chamada pelo *caller* e o seu resultado armazenado em arquivo na máquina chamadora. Com isto, torna-se possível determinar posteriormente qual a rota estava sendo utilizada no instante da chamada.

Em adição a esta informação, foi desenvolvido um outro *script*, utilizando a linguagem de programação PERL, que ao analisar os roteadores e enlaces envolvidos na ligação resultante da execução do *traceroute*, obtém as estatísticas de uso dos mesmos, disponibilizadas pela própria RNP em sua página *web* [47]. A RNP permite que qualquer usuário verifique, através de um *browser* comum, a taxa de utilização dos seus enlaces, tanto em número de *bits* quanto em pacotes trafegados, além do atraso de cada enlace. A atualização desta página é realizada automaticamente a cada cinco minutos, através do software de gerência MRTG, disponível em [48]. O *script* de consulta destas estatísticas, também desenvolvido em PERL, abre uma conexão TCP na porta 80 do servidor *web* da RNP e, através do protocolo HTTP, extrai as informações relevantes em relação ao link desejado, informando os valores mínimos, médios e máximos para cada uma das variáveis citadas acima. As informações coletadas são então armazenadas em outro arquivo, contendo o endereço IP dos roteadores entre a origem e o destino (obtidos através do *traceroute*) e as estatísticas de uso das suas interfaces, quando disponíveis.

Com as informações obtidas pelo *traceroute* e pelo MRTG da RNP, pode-se posteriormente correlacionar alguma ocorrência no *backbone* da RNP com a variação da qualidade de voz sendo percebida. A seguir, um exemplo de arquivo com os dados do *traceroute* e dos valores mínimo, médio e máximo para cada variável obtida na página de estatísticas da RNP.

```
146.164.247.193---- 1 146.164.247.193 0.325 ms
146.164.8.193---- 2 146.164.8.193 1.632 ms
200.20.94.58---- 3 200.20.94.58 2.343 ms
200.143.254.22---- 4 200.143.254.22 1.942 ms
                                          Min.      Medio     Max.
200.143.254.137----(Vazão em bits - Entrada) 10382171   6697466   7117723 |
                   (Vazão em bits - Saída) 23904966   15202608   22127953 |
                   (Vazão em pacotes – Entrada) 3749   2331   3096  |
                   (Vazão em Pacotes – Saída) 4979   3415   4441  |
                    (Delay – Entrada) 66   37   37 |
                    (Delay – Saída) 100   0   0 |
                    5 200.143.254.137 39.598 ms
200.19.119.123---- 6 200.19.119.123 53.111 ms
```

No exemplo acima, vemos o arquivo resultante da integração entre o *traceroute* e o MRTG, com comentário dos campos. O arquivo mostra a conexão realizada entre uma máquina no laboratório VoIP do NCE/UFRJ (146.164.247.196) com uma máquina no POP-DF (200.19.119.123). Os dois primeiros roteadores estão localizados no NCE/UFRJ. O terceiro é o roteador de saída da RedeRio (200.20.94.58), que faz conexão com a RNP. Nesta amostragem, somente estavam disponíveis estatísticas referentes ao roteador 200.143.254.137, localizado em Brasília. À medida que outras estatísticas de uso dos roteadores intermediários sejam disponibilizadas para acesso via *web*, estas serão incorporadas à ferramenta.

4.4. Geração e Visualização das Estatísticas

Uma vez de posse de todos os arquivos, já é possível gerar as estatísticas gerais, ou seja, calcular os valores mínimo, médio, máximo e desvio padrão das variáveis *jitter*, perda de pacotes e RTT de cada ligação. Esta tarefa é realizada por outro script, também

escrito na linguagem PERL, utilizando o módulo estatístico *Statistics::Descriptive* do *STAT.pm*, que possui todas as funções estatísticas necessárias. Este módulo estatístico encontra-se disponível no *site* CPAN [49]. Após os cálculos, o *script* gera um arquivo com o resumo de cada ligação ao longo do dia para um determinado destino. Este arquivo facilita a geração dos gráficos posteriormente, evitando que cálculos estatísticos sejam realizados a cada consulta.

O resultado de todos os dados coletados e compilados podem ser visualizados através de uma página *web* desenvolvida. A interface permite analisar os valores mínimos, médios, máximos e desvio padrão das variáveis de todas as ligações realizadas para um ponto remoto, possibilitando ver o seu comportamento ao longo de um dia, através dos dados obtidos no arquivo estatístico resumido. O usuário pode escolher a origem, destino, dia e *codec* utilizado, como ilustrado na Figura 28.

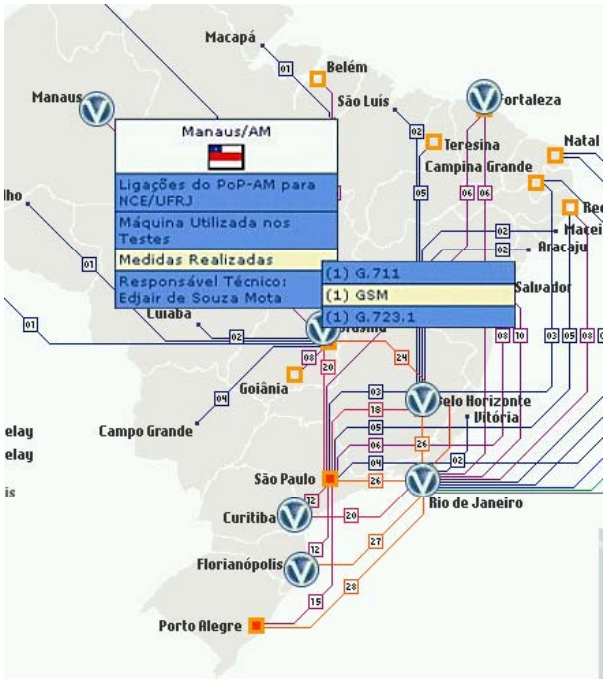


Figura 28. Interface Web em Javascript para visualizar resultados

Ao se clicar no ícone do laboratório VoIP, são oferecidos os *codecs* utilizados pelos testes, através de menus em *javascript*, quando então são mostrados três gráficos, contendo o *jitter*, RTT e perda de pacotes de todas as ligações ocorridas em cada dia.

Após a observação geral do comportamento das chamadas, pode-se selecionar uma chamada específica, conforme ilustrado na Figura 29, bastando selecionar o dia e hora da ligação e qual a origem e destino. Depois de selecionados estes dados, toda a ligação pode ser observada, como foi a distribuição de perda de pacotes ao longo da ligação e o comportamento do *jitter* e do RTT da mesma. Através da própria página, é possível ouvir a ligação de referência e como ela foi recebida pelo usuário remoto. Também é mostrada a rota feita pelos pacotes no momento da coleta e os dados estatísticos de utilização dos enlaces dos roteadores da RNP, extraídos da própria página da RNP.

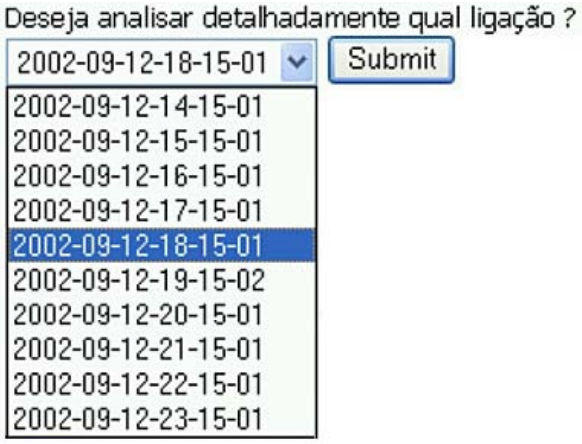


Figura 29. Seleção de chamada específica

A página foi gerada em *html* e *javascript* e os gráficos são gerados dinamicamente por um *cgi* escrito em PERL, utilizando as rotinas gráficas do módulo *GD.pm* e *GDGraph.pm*, também disponíveis através do CPAN [49].

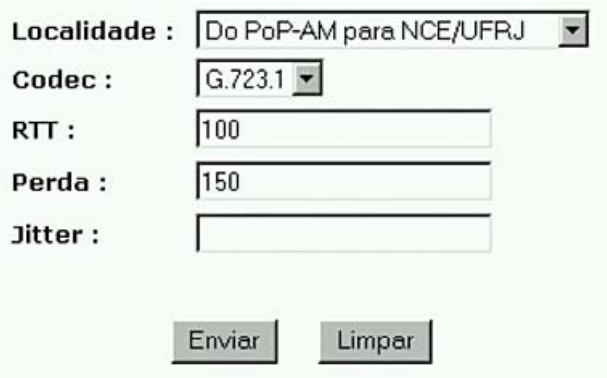


Figura 30. Interface de seleção por parâmetros de QoS

Também é possível selecionar chamadas especificas a partir de critérios de QoS como a taxa de perda de pacotes, *jitter* e RTT. Desta forma, é possível verificar o impacto de degradação gerado pela variação destes parâmetros.

A Figura 30 ilustra um exemplo de seleção de chamadas por parâmetros de QoS, e o seu resultado é ilustrado na Figura 31.

Ligação	RTT	Perdas	Jitter
nceufrj-amazonas-R-2002-11-25-11-35-21	190	191	47
nceufrj-amazonas-R-2002-11-26-08-35-21	244	175	37

Figura 31. Resultado de seleção por critérios de QoS

Neste exemplo, os fluxos obtidos são do POP-AM para o NCE/UFRJ, utilizando o *codec* G.723.1, com RTT igual ou superior a 100 ms e perda de pacotes superior a 150. Ao se selecionar determinada ligação, todos os detalhes da mesma são mostrados.

Em todos os momentos, é possível clicar sobre qualquer gráfico para obtenção de ampliação do mesmo, facilitando a sua visualização.

4.5. Medidas no Backbone RNP Utilizando a Ferramenta

A coordenação da RNP indicou os pontos que poderiam participar desta fase inicial do projeto piloto VoIP, e as instituições que estariam interessadas em colaborar. Cada instituição forneceu um equipamento na sua instalação, configurado com sistema operacional Linux ou FreeBSD, onde foram instalados o software *openam* e os *scripts* de apoio. O critério de escolha das instituições participantes foi baseado na distribuição geográfica das mesmas, objetivando traçar um perfil do comportamento do *backbone* da RNP passando por praticamente todos os seus enlaces, de forma a tornar o experimento mais representativo. As instituições selecionadas para o teste foram:

POP-AM	*Manaus*
POP-CE	*Fortaleza*
POP-DF	*Brasília*
POP-MG	*Belo Horizonte*
POP-PR	*Curitiba*
POP-SC	*Florianópolis*

Um script foi instalado na máquina belem.voip.nce.ufrj.br, localizada no NCE/UFRJ, que, através do *crontab*, disparava a cada hora uma chamada com 150 segundos de duração para cada um destes pontos, durante um período de aproximadamente dois meses. A primeira fase de coletas foi feita utilizando o *codec* G.711, e posteriormente foram feitas coletas com os *codecs* GSM e G.723.

Os dados eram transferidos posteriormente para a máquina gtvoip.nce.ufrj.br, onde eram armazenados. Depois de consolidados, os dados eram pré-processados para acelerar a sua visualização através da interface *web*.

Em alguns casos em que a conexão termina de forma anormal, o processo de *openam* continua ativo, gerando um conflito quando uma nova cópia do *openam* for disparada. A solução de contorno para este problema foi colocar um script no *crontab*, ao invés da execução direta da ferramenta *openam*. Este script encerra a execução de qualquer processo *openam* que esteja ativo, antes de disparar um novo processo do *openam*. Desta forma, garante-se que os arquivos de *log* e de gravação de mensagem recebida estarão vazios e evita que um processo *openam* fique executando indefinidamente.

4.5.1. Exemplos de Visualização

A seguir, são mostrados os gráficos gerados para o dia 21/10/2002, mostrando o comportamento de todas as ligações efetuados em RJ e DF, utilizando o codec G.711.

Os dados mostrados são selecionados após a apresentação do mapa do backbone da RNP e da localização de cada ponto de coleta VoIP instalado no país (Figura 28).

A escolha da ligação pode ser feita através da seleção específica da conexão, informando origem, destino, *codec*, data e hora, ou através da seleção automática por critérios de QoS.

Os gráficos de visualização por dia permitem uma visão geral do comportamento de determinado ponto, facilitando a seleção de pontos suspeitos ou fora da curva.

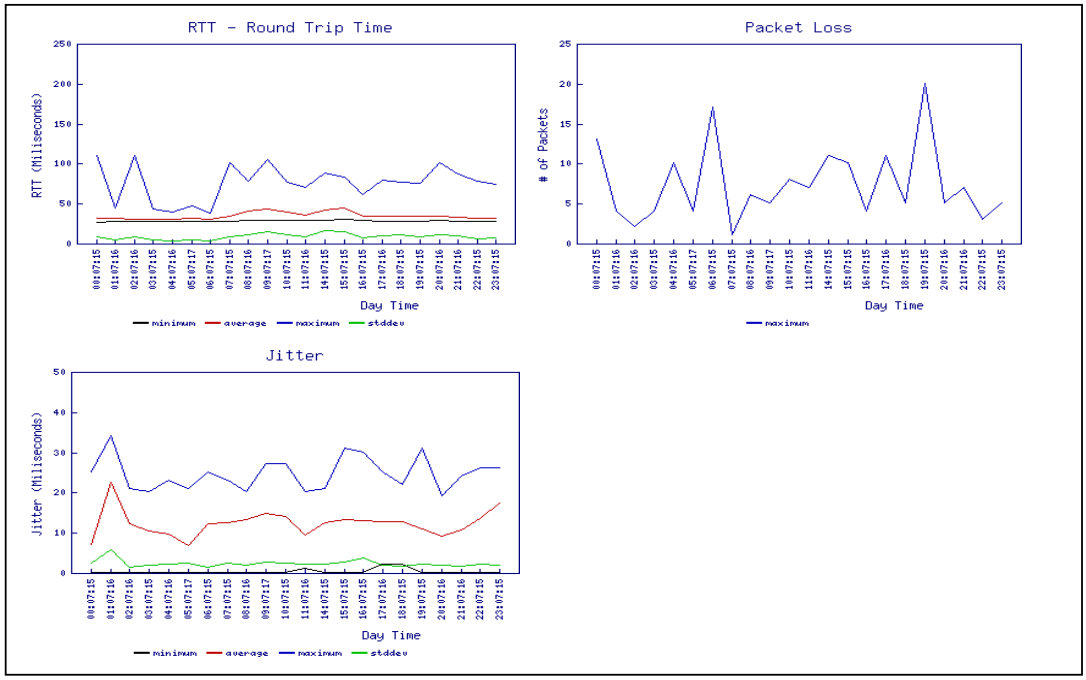


Figura 32. Exemplo de visualização geral diária

Os detalhes de uma conexão específica são mostrados na Figura 33 e Figura 34, conforme visualizados pela interface *web*.

Detalhamento da chamada
De : nceufrj
Para : brasilia
Hora : 10:14:16
Data : 28/10/2002
Codificador : ITU-T G.711
Ligação original => Ouvir esta ligação

Traceroute

IP	Tempo	MRTG					
146.164.247.193	0.239 ms						
146.164.8.193	0.628 ms						
200.20.94.58	2.455 ms						
200.143.254.22	2.338 ms						
200.143.254.137	25.422 ms	In			Out		
	%bps	Maximum	Average	Current	Maximum	Average	Current
		29.42%	17.90%	26.62%	86.54%	20.47%	67.30%
	Pcts/seg	Maximum	Average	Current	Maximum	Average	Current
		2570	1070	2345	4064	1638	3404
	RTT	Maximum	Average	Current			
		29	23	27			
200.19.119.123	26.590 ms						

Figura 33. Visualização de ligação específica

É possível escutar como foi recebida a ligação, bem como o som da ligação original. Esta característica do programa é interessante, pois permite verificar o impacto causado por determinados valores dos fatores de degradação da qualidade da ligação como a perda de pacotes, o RTT e o *jitter*.

Neste gráfico pode-se observar todos os dados referentes à ligação como, por exemplo, origem e destino da chamada, data e hora em que foi realizada e qual o codificador utilizado nesta chamada.

A informação do *traceroute* mostra o caminho utilizado pelos pacotes entre origem e destino. Os dados estatísticos obtidos das estatísticas do MRTG também podem ser verificados.

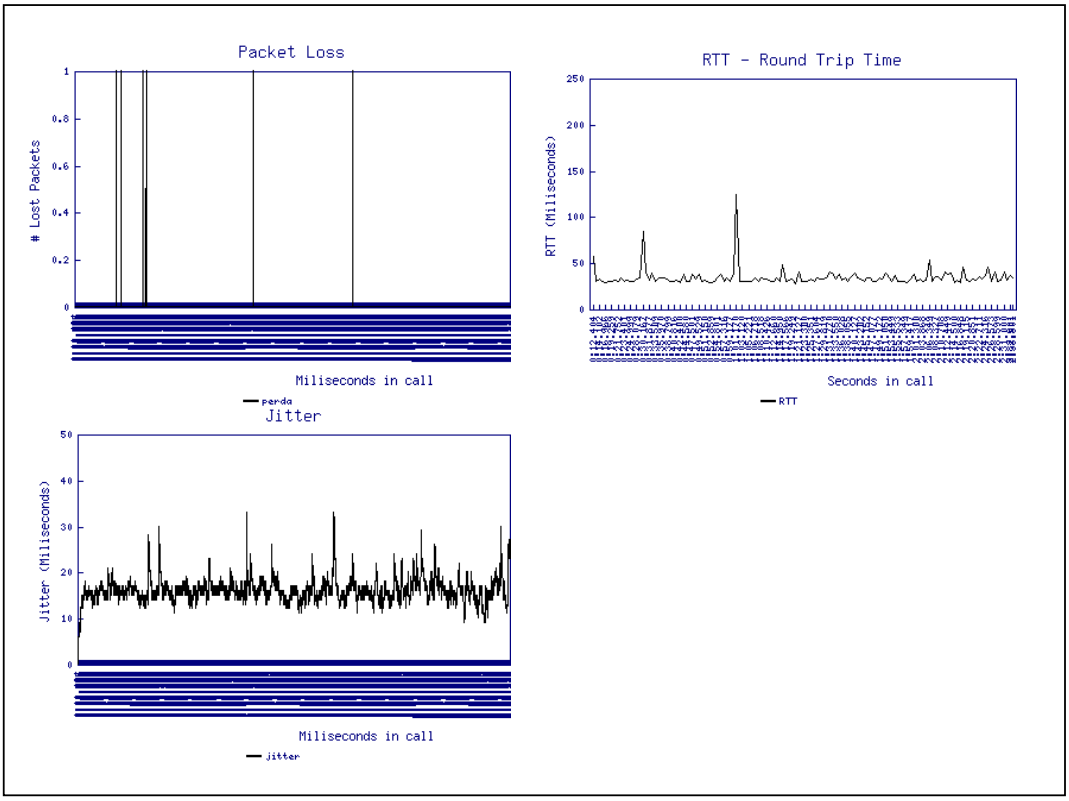


Figura 34. Visualização de ligação específica

Todos os resultados coletados estão disponíveis para visualização e consulta, através da página do laboratório VoIP do NCE/UFRJ.

Atualmente, a ferramenta está sendo utilizada para a análise de comportamento de ligações simultâneas, através da alteração dos scripts que disparam as chamadas.

Com a implantação de políticas de QoS no *backbone* da RNP, uma nova rodada de coletas será necessária para posterior comparação de comportamento das conexões.

Capítulo 5

Ferramenta de Monitoração Passiva

A ferramenta de monitoração ativa descrita no capítulo anterior permite entender melhor o comportamento do *backbone* da RNP em relação a aplicações de tempo real e fazer uma projeção de planejamento de capacidade e expansão da infra-estrutura da rede, servindo também como base para dimensionar os recursos sendo utilizados. No entanto, não é um método para avaliação de qualidade em tempo real sendo obtida instantaneamente. Essa avaliação em tempo real é fundamental para a implementação de uma gerência pró-ativa, na qual a queda de qualidade das ligações é detectada no momento em que está ocorrendo. Outro fator fundamental para a utilização de uma ferramenta não-intrusiva é que a ferramenta de monitoração ativa é baseada em clientes específicos. Desta forma, a ferramenta de monitoração ativa não serve para avaliar a qualidade das conexões efetuadas através de *gateways* ou equipamentos dedicados. Ligações realizadas através de *gateways* e *gatekeepers* atuando em modo *proxy* serão muito utilizadas no cenário proposto para o serviço VoIP na RNP.

A principal motivação para o desenvolvimento de uma ferramenta de monitoração passiva é a inexistência de um aplicativo de domínio público com este perfil. A grande dificuldade de se implementar este tipo de ferramenta é a ausência de portas de comunicação pré-definidas para a comunicação VoIP, visto que essas portas são negociadas dinamicamente pelos protocolos de sinalização. Somente os protocolos de sinalização para estabelecimento de conexões utilizam portas fixas. Um programa de monitoração passiva deve inicialmente monitorar todos os pacotes de sinalização e, ao identificar o início de uma nova conexão, interpretar os pacotes a fim de identificar que portas serão usadas por esta conexão VoIP sendo estabelecida.

A utilização de *sniffers* de uso genérico, como o popular *tcpdump* [50] ou o *ethereal* [51], não é a solução mais indicada. Embora ambos sejam capazes de interpretar pacotes RTP e RTCP, e no caso do *ethereal*, até mesmo pacotes de sinalização H.323 (por intermédio da instalação de um *plugin* específico) [51], nenhum

dos dois é capaz de filtrar dinamicamente as portas negociadas durante a sinalização, nem associar as diversas portas sendo utilizadas de um mesmo fluxo. O uso de *sniffers* praticamente se restringe à depuração de aplicativos e problemas em ambientes VoIP, ou mesmo para fins acadêmicos, onde desempenham papel fundamental para o entendimento e visualização dos pacotes de sinalização H.323.

Um programa que capture os pacotes VoIP que trafegam por uma rede, deve inicialmente analisar os pacotes de sinalização e de inicialização de uma chamada (*setup*). Durante esta conversação, são negociados diversos atributos que serão utilizados pelas pontas da conexão como *codecs*, buffer e o mais importante neste caso, as portas que serão utilizadas pelo tráfego de multimídia propriamente dito. A partir da identificação das portas dos canais de mídia, é possível selecionar e analisar os pacotes RTP e RTCP referentes à conexão. Todas as portas utilizadas (sinalização, mídia) devem ser mantidas relacionadas ao fluxo, possibilitando que ao término do mesmo, estas portas não permaneçam sendo monitoradas indefinidamente.

Existem ferramentas comerciais que se propõe a avaliar, de modo passivo, um tráfego de voz passando pela rede. Um estudo comparativo interessante foi realizado em [52], onde as características de algumas destas ferramentas são descritas e analisadas. O apêndice B mostra uma tabela contendo o resultado de parte deste estudo.

O laboratório de pesquisas da AT&T desenvolveu uma ferramenta com características muito similares as necessárias para o projeto piloto VoIP. O projeto da AT&T, denominado *mmdump* [53], o descreve como uma ferramenta para monitoração de tráfego multimídia. O *mmdump* é baseado no código do *tcpdump*, mas faz uso de módulos de protocolos específicos para determinar as portas dinâmicas que devem ser monitoradas. Inicialmente foi incorporado ao seu código um interpretador para a sinalização H.323 e, numa segunda fase passou a reconhecer também a sinalização SIP. O *mmdump* mantém uma tabela de estados de cada ligação em andamento. Como o objetivo do *mmdump* era medir o tráfego real na rede da AT&T, que possui um volume muito elevado, foi tomado um cuidado especial em relação à performance e escalabilidade [53]. Essa ferramenta tem sido utilizada com sucesso, tendo gerado material para diversos *papers* e dados para pesquisa. A ferramenta, com pequenas adaptações, atenderia perfeitamente as necessidades do projeto VoIP. No entanto, o

mmdump foi desenvolvido apenas para uso interno e conseqüentemente, seu código não está disponível. Pelos motivos acima foi desenvolvida a ferramenta para monitoramento passivo de chamadas de voz.

5.1. Esquema Geral de Funcionamento

A ferramenta é um agente localizado no ponto onde se deseja coletar a informação sobre as conexões. A ferramenta deve ser colocada no mesmo segmento físico por onde passa o tráfego do equipamento a ser monitorado, como um *gateway* ou *gatekeeper*. Deste modo, a ferramenta é capaz de analisar todos os pacotes cujo endereço de origem ou de destino seja o do equipamento sendo monitorado. As informações coletadas são processadas e disponibilizadas via SNMP. A Figura 35 mostra o cenário de aplicação da ferramenta.

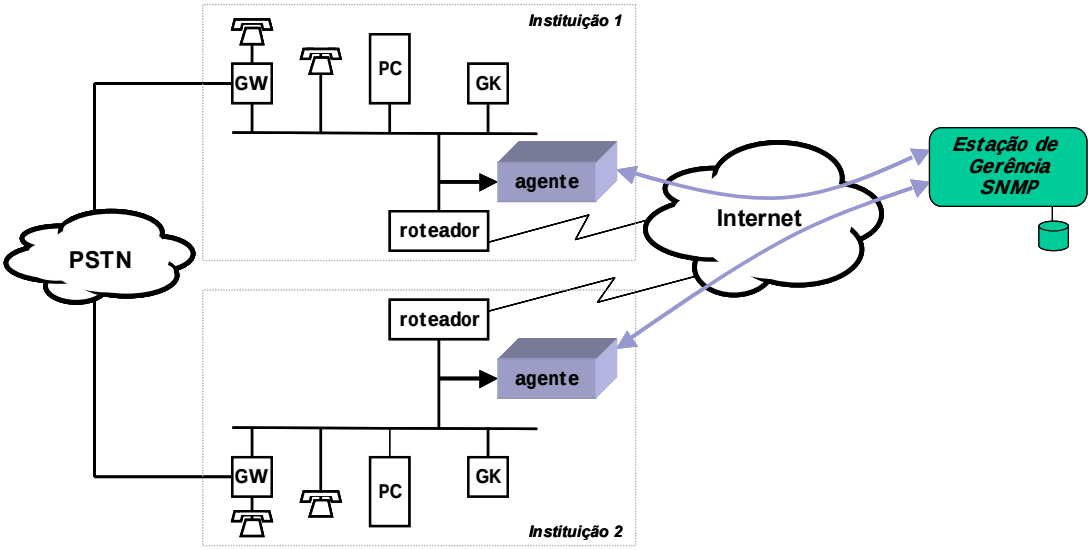


Figura 35. Cenário de Utilização da Ferramenta

O agente captura os pacotes trafegando na rede, interpreta a sinalização, analisa o fluxo de mídia, faz uma estimativa de qualidade de cada fluxo em tempo real e disponibiliza esta informação através de uma MIB SNMP. Também gera *traps* SNMP para uma estação de gerência toda vez que parâmetros de qualidade de uma ligação atinjam limiares pré-definidos.

5.2. Management Information Base

As informações coletadas pela ferramenta são disponibilizadas para consulta por intermédio de estações de gerência SNMP. Os dados ficam armazenados em duas MIB (*Management Information Base*): MIB de conexões ativas e MIB de conexões encerradas. A MIB de conexões ativas armazena as informações de todas as conexões ativas em um dado instante. Todos os dados estatísticos e a qualidade estimada das conexões armazenadas na MIB refletem os valores até o momento da consulta, sendo atualizados periodicamente. Uma vez encerrada a conexão, as informações são movidas para a MIB de conexões encerradas, permanecendo disponíveis para consulta por um período pré-determinado. Nessa MIB, os valores refletem os resultados finais coletados para cada conexão. A MIB de conexões encerradas mantêm o histórico das chamadas monitoradas mais recentemente.

As MIBS utilizam o formato definido pelo *framework* RAQMON (*Real-time Application Quality of Service Monitoring*) [58, 59, 60], que é uma proposta de extensão ao *framework* RMON [61] do IETF. A opção por duas MIBs distintas permite tratamento diferenciado entre conexões ativas e conexões já encerradas. O tempo mínimo de permanência da informação na MIB e número de entradas válidas por MIB são exemplos desta diferença. Além disto, permite consultas mais otimizadas por parte das estações de gerência. Outras MIBs, como a DialControl Mib [71], também fazem esta separação.

Diversas informações referentes a cada conexão ficam armazenadas nas MIBs. Entre elas, podemos citar: endereço IP origem e destino, portas utilizadas pela conexão, *string* de identificação da origem e do destino (usuário@host, identificador E.164 ou identificador SIP), *string* de identificação da aplicação utilizada pela origem e destino, formato de mídia sendo usado pela origem e destino, hora de início da chamada no formato NTP, duração total da chamada, total de pacotes perdidos, *jitter* suavizado conforme RFC do RTP, RTT médio, total de pacotes recebidos e enviados, total de octetos recebidos e enviados, entre outros. A avaliação objetiva da qualidade instantânea da conexão também fica armazenada na MIB.

5.3. Descrição de Funcionamento

A ferramenta é composta por um processo principal e processos auxiliares. O processo principal identifica as conexões H.323 sendo estabelecidas. Ele também é responsável pela manutenção das MIBs, pela comunicação SNMP com as estações de gerência e pela interpretação de comandos de administração remota da ferramenta.

A cada nova conexão identificada, um processo auxiliar é disparado para captura e análise de todos os pacotes relativos à conexão H.323 correspondente, permanecendo em execução enquanto a conexão estiver ativa. Esse processo possui um módulo de avaliação objetiva da qualidade de voz, fazendo uma avaliação instantânea da chamada em andamento.

A Figura 36 ilustra o esquema geral de funcionamento.

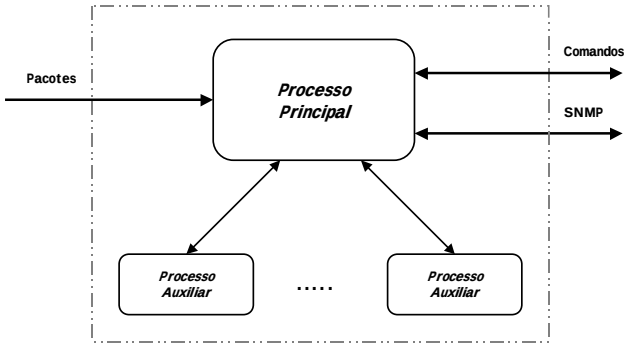


Figura 36. Esquema Geral de Funcionamento da Ferramenta

Os processos que compõem a ferramenta foram concebidos com o uso de módulos visando atender a dois objetivos básicos: possibilitar eficiência na identificação, coleta e processamento dos fluxos e, por conseguinte, garantir a escalabilidade da ferramenta; e também permitir que, sempre que possível, os módulos utilizem bibliotecas de código aberto e rotinas disponibilizadas pelo sistema operacional.

O processo de inicialização da ferramenta consiste em criar em memória as estruturas vazias das MIB e inicializar as variáveis globais de configuração. A função responsável pela comunicação SNMP com estações de gerência é ativada, e passa a

aguardar por comandos ou eventos. Outra função disparada possibilita a administração da ferramenta remotamente, através de comandos vindos pela rede.

Os processos são detalhados a seguir.

5.4. Processo Principal

Este processo é composto dos módulos apresentados na Figura 37.

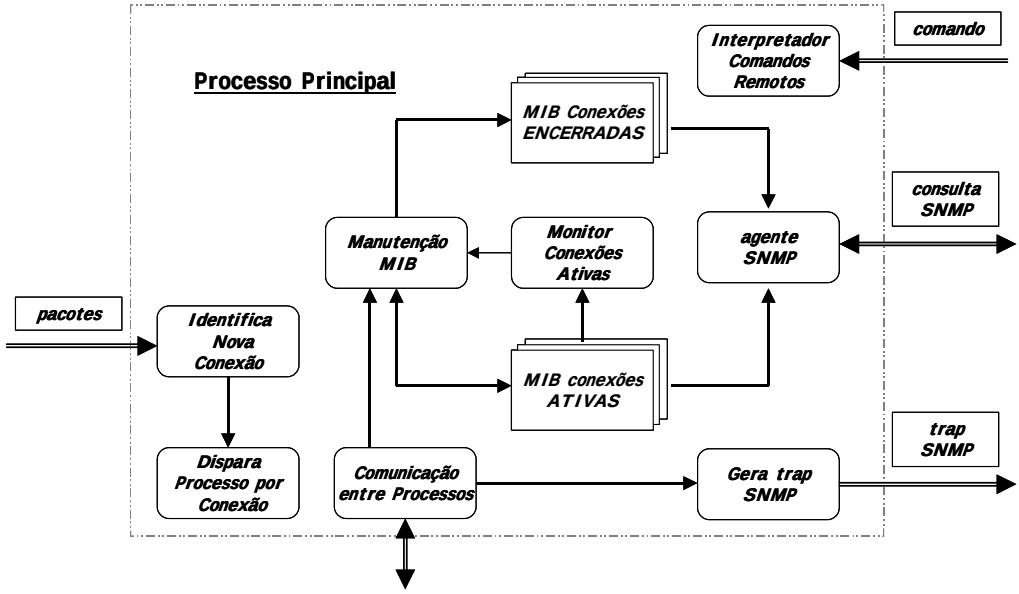


Figura 37. Arquitetura do Processo Principal

A operação básica é iniciada com a identificação de uma nova conexão, através do módulo "*Identifica Nova Conexão*". Ao ser identificada, o módulo "*Dispara Processo Por Conexão*" é acionado e um processo auxiliar é criado para tratar esta nova conexão. Como descrito na seção 5.2, as MIBs mantêm informações separadas para as conexões ativas e as já encerradas. A manutenção destas MIBs é feita pelos módulos "*Manutenção MIB*" e "*Monitor Conexões Ativas*". As transações SNMP são tratadas pelos módulos "*agente SNMP*" e "*Gera trap SNMP*". A comunicação entre o processo principal e os processos auxiliares é feita através do módulo "*Comunicação entre Processos*". A interpretação e execução dos comandos de administração remota é feita pelo módulo "*Interpretador Comandos Remotos*".

O detalhamento de cada módulo do processo principal é descrito a seguir.

5.4.1. Módulo "*Identifica Nova Conexão*"

Este módulo é responsável pela obtenção de todos os pacotes de sinalização que passam pela rede para posterior análise. A captura de pacotes é feita colocando a interface de rede em modo promíscuo, utilizando a biblioteca *pcap*, disponível nos ambientes Unix e Windows. Esta biblioteca permite selecionar que pacotes serão capturados, através da criação de filtros. Neste módulo, o filtro utilizado seleciona somente pacotes TCP com porta destino 1720, que possuam o bit SYN ligado e ACK desligado, caracterizando o início de uma nova conexão TCP. A porta TCP 1720 é o *default* para o estabelecimento de chamadas H.323. Quando um pacote é capturado, são extraídos os endereços IP e portas TCP da origem e do destino da conexão H.323 sendo estabelecida, e estas informações são passadas para o módulo "*Dispara Processo por Conexão*".

5.4.2. Módulo "*Dispara Processo por Conexão*"

Este módulo recebe as informações sobre a nova conexão sendo estabelecida e cria um processo auxiliar, passando estes dados como parâmetro. Estes parâmetros serão usados pelo processo auxiliar para configurar o seu filtro de pacotes, de forma a selecionar somente os pacotes desta conexão específica.

5.4.3. Módulo "*Manutenção MIB*"

Este módulo é responsável por checar periodicamente se existe alguma entrada nas MIBs que já possa ser retirada. O intervalo de tempo desta checagem, bem como o tempo de mínimo de permanência de uma entrada na MIB e o número total de entradas permitidas são configuráveis. A transferência das informações entre MIBs, quando uma conexão é encerrada, também é realizada por este módulo. Este módulo evita que uma MIB cresça de forma infinita. Quando um processo auxiliar detecta o final de uma chamada, este módulo é avisado através do módulo "*Comunicação entre Processos*". Este módulo também interage com o módulo "*Monitor de Conexões Ativas*".

5.4.4. Módulo "*Monitor de Conexões Ativas*"

Este módulo checa periodicamente se existe alguma conexão na MIB ativa sem atualização após um determinado período. Este é um procedimento de segurança para evitar que alguma conexão H.323 permaneça na MIB de conexões ativas indefinidamente. Este tipo de situação poderia ocorrer em caso de término anormal do processo auxiliar da conexão. Caso exista na MIB alguma conexão nesta situação, o módulo "*Manutenção de MIB*" é acionado para transferência destas informações para a MIB de conexões encerradas.

5.4.5. Módulo "*Agente SNMP*"

Este módulo, ao receber as requisições SNMP vindas de uma estação de gerência, consulta a MIB pertinente e retorna a informação solicitada.

5.4.6. Módulo "*Gera trap SNMP*"

Sua função é gerar um *trap* SNMP para uma estação de gerência pré-configurada, após receber a indicação de um processo auxiliar que sua conexão está com qualidade estimada fora dos limites tolerados. Esta comunicação é feita através do módulo "*Comunicação entre Processos*". Estes *traps* permitem que o gerenciamento do serviço determine as tendências do comportamento da qualidade das conexões sendo monitoradas.

5.4.7. Módulo "*Interpretador de Comandos Remotos*"

Este módulo possibilita a administração remota do coletor, permitindo a alteração de diversos parâmetros de configuração. Os comandos para o programa coletor são enviados pelo cliente utilizando uma conexão TCP. A porta utilizada por *default* é a 5261, podendo ser alterada pelo parâmetro de execução *CmdPort* no momento de inicialização do programa coletor. A ferramenta, ao ser iniciada, dispara o módulo servidor, que abre a conexão *socket* e fica aguardando os comandos vindos dos clientes. A restrição dos clientes autorizados a enviar comandos é feita pelo IP. Pode-se

utilizar um arquivo de configuração da própria aplicação cujo conteúdo são os endereços IP dos clientes válidos. Outra forma é no nível do próprio *kernel* do sistema operacional, através de filtro de pacotes como *iptables* para ambientes Linux ou *ipfw* para ambientes FreeBSD.

Este módulo provê uma maior flexibilidade à ferramenta, permitindo que diversos parâmetros sejam alterados durante a execução. Inicialmente são previstos comandos para alteração do endereço IP para onde devem ser enviados os *traps* SNMP, limite tolerado de qualidade estimada, período de varredura para manutenção das MIBs, tempo máximo de inatividade de uma conexão e período de atualização dos dados das MIBs.

5.4.8. Módulo "*Comunicação entre Processos*"

Este módulo possibilita a comunicação entre o processo principal e os processos auxiliares. Através deste módulo, um processo auxiliar pode informar o encerramento de sua conexão, atualizar a informação da MIB ou gerar um *trap* de alerta para uma estação de gerência.

5.5. Processo Auxiliar

Os módulos que compõe este processo são mostrados na Figura 38.

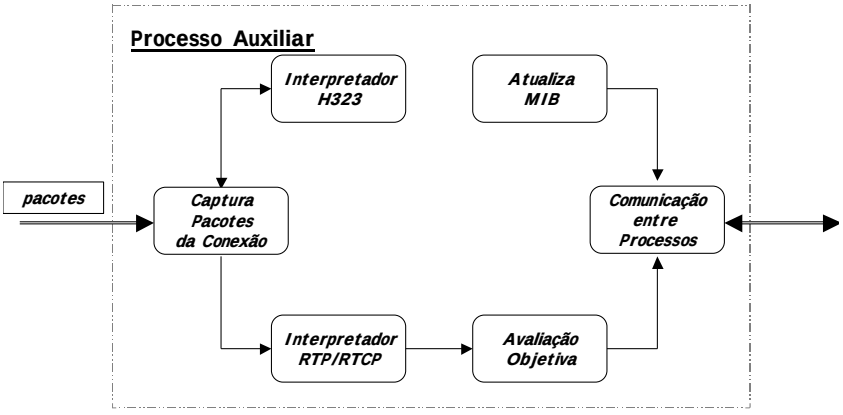


Figura 38. Arquitetura do Processo Auxiliar

O processo auxiliar utiliza os parâmetros passados na sua ativação para configuração do filtro de captura dos pacotes da conexão. Cada processo auxiliar é responsável pela captura dos pacotes da sua conexão, através do módulo "*Captura Pacotes da Conexão*". O módulo "*Interpretador H323*" analisa os pacotes capturados, e interpreta a sinalização H323, identificando a negociação dos canais de comunicação sendo estabelecidos, tanto de mídia quanto de sinalização. O módulo "*Interpretador RTP/RTCP*" extrai as informações necessárias para o cálculo de avaliação objetiva da qualidade da conexão, e as repassa para o módulo "*Avaliação Objetiva*". A atualização da MIB de conexões ativas é feita periodicamente através do módulo "*Atualiza MIB*". Toda comunicação entre o processo auxiliar e o processo principal é feita através do módulo "*Comunicação entre Processos*".

O detalhamento de cada módulo do processo auxiliar é descrito a seguir.

5.5.1. Módulo "*Captura Pacotes da Conexão*"

Este módulo captura os pacotes específicos da conexão. Inicialmente o seu filtro seleciona somente os pacotes de estabelecimento de chamada. À medida que novos canais de comunicação são negociados para esta conexão, o módulo "*Interpretador H.323*" os identifica e atualiza dinamicamente o filtro de seleção de pacotes para incluir as novas portas. São incluídas no filtro as portas dos canais de sinalização H.225 e H.245, além das portas de transporte de mídia do RTP e do RTCP. Os pacotes capturados que se referem à mídia são passados para o módulo "*Interpretador RTP/RTCP*", e os de sinalização para o módulo "*Interpretador H.323*".

A Figura 39 mostra o processo de identificação das portas negociadas dinamicamente pela sinalização H.225 e H.245.

5.5.2. Módulo "*Interpretador H323*"

Este módulo é responsável por monitorar toda a sinalização H.323 desta conexão. Através da análise da negociação realizada pelos protocolos de sinalização de H.225 e H.245, este módulo identifica que portas serão utilizadas para os canais de

mídia entre origem e destino desta chamada. O filtro de pacotes do processo auxiliar é alterado dinamicamente para incluir estas portas, conforme discutido no item 5.5.1. A análise da sinalização é feita durante toda a chamada. Ao detectar a sinalização de encerramento da mesma, informa ao processo principal e encerra a execução do processo auxiliar.

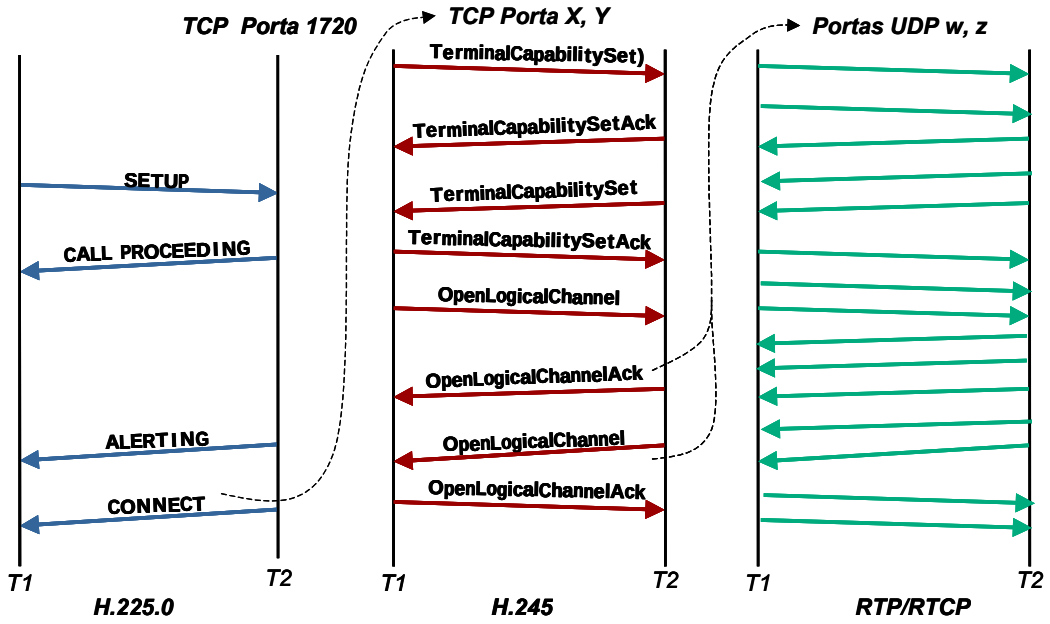


Figura 39. Identificação das portas sendo utilizadas

Este módulo deve ser capaz de interpretar todas as formas de sinalização H.323, inclusive melhorias feitas ao estabelecimento de chamadas, como o *fast-start* [44].

5.5.3. Módulo "*Interpretador RTP/RTCP*"

Este módulo é responsável por interpretar os pacotes RTP e RTCP desta chamada. O *jitter* instantâneo é avaliado através do tempo entre chegadas de pacotes RTP. Informações como *jitter* suavizado, total de pacotes perdidos e RTT são extraídos dos pacotes RTCP. Estas informações são passadas para o módulo "*Avaliação Objetiva*".

5.5.4. Módulo "*Avaliação Objetiva*"

Este módulo faz uma análise objetiva da qualidade da conexão sendo monitorada. Não faz parte do escopo deste trabalho implementar ou desenvolver novos modelos de avaliação objetiva. Este trabalho será fruto de outras pesquisas sendo realizadas no laboratório VoIP do NCE. Uma vez estimada a qualidade instantânea desta conexão, um evento é enviado para o módulo de *traps* SNMP do processo principal, caso a avaliação desta conexão esteja fora dos limites tolerados. Estes limites podem ser ajustados dinamicamente através de comandos ao módulo de administração remota no processo principal, que os repassa ao processo auxiliar correspondente. Cada conexão pode ter limites específicos para a qualidade tolerada.

5.5.5. Módulo "*Atualiza MIB*"

Periodicamente, este módulo pega os valores calculados até o momento e os copia para a MIB de conexões ativas. O intervalo de tempo desta atualização é configurável, podendo ser alterado dinamicamente através do módulo "*Interpretador de Comandos Remotos*". Caso este módulo verifique que a conexão sendo monitorada está inativa por um determinado período, é assumido que a conexão teve um término anormal. Nesse caso, o módulo "*Manutenção MIB*" do processo principal é avisado e o processo auxiliar é encerrado.

5.5.6. Módulo "*Comunicação entre Processos*"

Este módulo permite a comunicação entre o processo auxiliar e o processo principal.

5.6. Implementação da Ferramenta

O programa está sendo desenvolvido utilizando a linguagem C++, voltado para ambiente Unix, embora seja possível convertê-lo para o ambiente MS-WINDOWS sem que seja necessário um grande esforço de programação.

Toda a captura de pacotes é realizada através da biblioteca de captura de pacotes *libpcap*. Esta biblioteca é disponível em todos os sistemas Unix e foi transportada para ou ambiente Windows com o nome de *winpcap* [54].

A *pcap* possui uma interface extremamente simples e direta, tornando a programação de um utilitário tipo *sniffer* uma tarefa relativamente simples. Os passos para colocar uma interface em modo promíscuo são brevemente descritos a seguir.

1. Determinação da interface de rede a ser utilizada para captura. Pode ser descoberta automaticamente por rotinas da própria biblioteca, ou então especificada explicitamente.

2. Inicialização da *pcap*. Neste ponto é que realmente a interface é colocada em modo promíscuo.

3. Opcionalmente, a criação de um filtro para selecionar os pacotes desejados dentre todos os capturados. O processo de criar um filtro é dividido em duas fases, descritas a seguir. Na primeira fase, uma *string* contendo os critérios de seleção do filtro é passada para uma rotina da própria biblioteca. Essa rotina converte o texto recebido em uma estrutura de meta código contendo a mesma informação, porém em um formato que a biblioteca compreende. A segunda fase de criação de um filtro é responsável pela associação do código gerado com a interface propriamente dita. Esta conversão é chamada de compilação de filtro.

4. O último passo é entrar em um processo de leitura de pacotes capturados. Pode ser feito através de leitura individual, pacote a pacote, ou então chamar uma rotina da biblioteca, que associa o evento de captura de novos pacotes com uma rotina definida pelo próprio usuário.

Os filtros são compostos por expressões regulares descritas em modo texto, com possibilidade de incluir operadores booleanos. Por exemplo, o filtro "*udp and (host X and port W) and (host Y and port Z)*", selecionaria o fluxo de pacotes UDP entre os hosts com endereço IP X e Y e portas W e Z, respectivamente. O esquema de filtro de pacotes é extremamente eficiente, sendo otimizado para atuar no nível de *kernel* do sistema operacional. No entanto, a geração do código do filtro através da rotina *pcap-*

compile é um processo que faz um uso intensivo de CPU. Os desenvolvedores do projeto *mmdump* apresentam soluções em [55]. Os autores da *pcap* estão estudando alternativas para a criação de filtros dinâmicos de forma eficiente e com escalabilidade [63].

O módulo de interpretação da sinalização H.323 utiliza a biblioteca OpenH323, que permite a interpretação das definições ASN.1 de cada PDU do H.323. As chamadas para interpretação da sinalização H.323 foram baseadas no programa Dump323, escrito em C++ e disponível no próprio site do projeto OpenH323. Este programa foi escrito pela equipe que desenvolveu a biblioteca OpenH323, com o objetivo de depuração na época do desenvolvimento da mesma.

O módulo de interpretação RTP/RTCP também utilizará a biblioteca OpenH323.

Os módulos relacionados à manipulação das MIBs e gerência SNMP irão utilizar as rotinas da biblioteca Net-SNMP [72]. Esta biblioteca é bastante completa e fornece todas as ferramentas necessárias.

O módulo de avaliação objetiva é comentado a seguir.

5.7. Módulo de Avaliação Objetiva

Como descrito anteriormente, o módulo de avaliação objetiva será resultado de outros trabalhos em andamento do laboratório VoIP do NCE/UFRJ, permitindo a análise e pesquisa de novas metodologias para estimar a qualidade de serviço VoIP.

Numa primeira fase de implementação, a biblioteca VQmon desenvolvida pela empresa Telchemy, estará sendo utilizada para a obtenção da qualidade de voz na ferramenta. Essa biblioteca, que implementa o modelo do Extended E-Model, foi cedida pela empresa desenvolvedora como fruto de uma parceria com o NCE/UFRJ.

Para exemplificar a avaliação de MOS feita pela biblioteca VQmon, foi utilizado o utilitário *VQcapture* que, a partir de um arquivo capturado pelo *tcpdump*, faz uma análise da qualidade obtida da conexão. A execução do *VQcapture* permitiu avaliar o potencial da biblioteca VQmon.

```
[fabio@belem redhat]$ ./vqcapture

Telchemy VQcapture Voice Quality Analyzer - Version 1.10.013
Using Telchemy VQmon/SA version 1.1 - build 34
Copyright (c) 2002 Telchemy, Incorporated. All rights reserved.

usage: vqcapture [-options] <capture file>

where the supported <options> are:
-q                  Quiet mode. Minimize import output.
-v                  Verbose mode. Detailed import output.
-s                  Enhanced statistics on call completion.
-jf                 Use VQmon/SA fixed jitter buffer emulator.
-ja                 Use VQmon/SA adaptive jitter buffer emulator.
-jnom <delay>       VQmon/SA jitter buffer nominal delay.
-jmax <delay>       VQmon/SA jitter buffer maximum delay.
-jmin <delay>       VQmon/SA jitter buffer minimum delay.
-e <export file>    Exports RTP packet fields to the specified file.
-f <filter>         Specifies the filter expression.
-u <frequency>      Specifies the metrics update frequency in ms.
```

A execução simples do utilitário *VQcapture* mostra as opções válidas para execução. No exemplo a seguir, foi utilizado o arquivo "ucla". Este arquivo foi gerado através do programa *ethereal*, que coletou os pacotes de uma sessão feita a partir dois PC, localizados no NCE e na UCLA, Califórnia, executando o software NetMeeting. Pode-se observar que o MOS calculado resulta em uma avaliação muito boa da qualidade obtida, o que realmente ocorreu durante a conexão. A opção "s" foi usada para gerar mais detalhes da conexão.

```
fabio@belem redhat]$ ./vqcapture -s ucla

Telchemy VQcapture Voice Quality Analyzer - Version 1.10.013
Using Telchemy VQmon/SA version 1.1 - build 34
Copyright (c) 2002 Telchemy, Incorporated. All rights reserved.

Capture file name:ucla
Capture start time:Thu Jul 24 18:58:02 2003
Capture duration:00:01:02
Capture packet records:6178
VQmon JB mode:Fixed
VQmon JB nominal delay:40
VQmon JB maximum delay:80

Call Stream Quality Analysis (SSRC=2169468879):
===============================================================
Call source:129.79.51.203:26118
Call destination:146.164.247.202:18642
Call start time:Thu Jul 24 18:58:02 2003
CODEC type:G.729A/G.729AB  (20 msec/pkt)
Quality (R-factors):Rnet=83, Ruser=82
Quality (MOS):CQ=3.93, LQ=3.96, PQ=4.02
                                                    CONTINUA…
```

```
Packet Delay Variation (ms):avg=8.4, max=32.2
Detectable Packet Delay (ms):avg=2.7, min=-10.4, max=23.8
Packets received:3084
Packets lost:0
Packets discarded:0
Packets early:595
Packets late:0
Packets out-of-seq:63
Overrun discards:0
Underrun discards:0
Duplicate discards:0
Delay increases:0
Delay decreases:0
Resynchronizations:0
Resets:0
Minimum delay:40
Maximum delay:40
Average delay:40
Current delay:40

Call Stream Quality Analysis (SSRC=178517962):
================================================================
Call source:146.164.247.202:18642
Call destination:129.79.51.203:26118
Call start time:Thu Jul 24 18:58:03 2003
CODEC type:G.729A/G.729AB  (20 msec/pkt)
Quality (R-factors):Rnet=83, Ruser=82
Quality (MOS):CQ=3.93, LQ=3.96, PQ=4.02
Packet Delay Variation (ms):avg=0.4, max=40.1
Detectable Packet Delay (ms):avg=-1.5, min=-2.9, max=38.2
Packets received:3074
Packets lost:0
Packets discarded:1
Packets early:484
Packets late:1
Packets out-of-seq:0
Overrun discards:0
Underrun discards:1
Duplicate discards:0
Delay increases:0
Delay decreases:0
Resynchronizations:0
Resets:0
Minimum delay:40
Maximum delay:40
Average delay:40
Current delay:40


Total RTP streams:              2
Total RTP packets:              6159
Total filtered packets:         19
```

Os valores obtidos mostram uma ligação de muito boa qualidade, com o MOS aproximadamente 4 e *R-factor* com 82, em uma escala de 0 a 100. *Rnet* é o cálculo do

R-factor simplesmente levando em consideração a degradação relacionada à rede como a perda de pacotes em rajada, o *jitter* e o algoritmo de codificação usado na conexão. Já o *Ruser* leva em consideração os fatores de atraso, a perda de pacotes e o efeito memória. Na avaliação MOS, CQ é a estimativa do VQmon para a "*Conversational Quality*", que leva em consideração atraso e efeito memória. MOS-LQ, ou "*Listening Quality*", já não leva em consideração o atraso e efeito memória, gerando uma avaliação de MOS superior ao calculado em CQ. O valor MOS-LQ basicamente implementa o padrão do ITU P.862 (PESQ) *Listening Quality*. PQ-MOS é o valor obtido pelo padrão P.862 *Raw Quality*.

```
[fabio@belem redhat]$ ./vqcapture -jnom 20 -jmax 20 -s -v  ucla

Telchemy VQcapture Voice Quality Analyzer - Version 1.10.013
Using Telchemy VQmon/SA version 1.1 - build 34
Copyright (c) 2002 Telchemy, Incorporated. All rights reserved.

Capture file name:ucla
Capture start time:Thu Jul 24 18:58:02 2003
Capture duration:00:01:02
Capture packet records:6178
VQmon JB mode:Fixed
VQmon JB nominal delay:20
VQmon JB maximum delay:20

Call Stream Quality Analysis (SSRC=2169468879):
================================================================
Call source:129.79.51.203:26118
Call destination:146.164.247.202:18642
Call start time:Thu Jul 24 18:58:02 2003
CODEC type:G.729A/G.729AB  (20 msec/pkt)
Quality (R-factors):Rnet=54, Ruser=54
Quality (MOS):CQ=2.68, LQ=2.68, PQ=2.74
Packet Delay Variation (ms):avg=13.0, max=32.2
Detectable Packet Delay (ms):avg=2.7, min=-10.4, max=23.8
Packets received:2805
Packets lost:0
Packets discarded:279
Packets early:595
Packets late:232
Packets out-of-seq:63
Overrun discards:47
Underrun discards:232
Duplicate discards:0
Delay increases:0
Delay decreases:0
Resynchronizations:0
Resets:0
Minimum delay:20
Maximum delay:20
Average delay:20
Current delay:20

                                                        CONTINUA…
```

```
Call Stream Quality Analysis (SSRC=178517962):
============================================================
Call source:146.164.247.202:18642
Call destination:129.79.51.203:26118
Call start time:Thu Jul 24 18:58:03 2003
CODEC type:G.729A/G.729AB  (20 msec/pkt)
Quality (R-factors):Rnet=83, Ruser=83
Quality (MOS):CQ=3.96, LQ=3.96, PQ=4.02
Packet Delay Variation (ms):avg=0.4, max=40.1
Detectable Packet Delay (ms):avg=-1.5, min=-2.9, max=38.2
Packets received:3072
Packets lost:0
Packets discarded:3
Packets early:484
Packets late:3
Packets out-of-seq:0
Overrun discards:0
Underrun discards:3
Duplicate discards:0
Delay increases:0
Delay decreases:0
Resynchronizations:0
Resets:0
Minimum delay:20
Maximum delay:20
Average delay:20
Current delay:20

Total RTP streams:              2
Total RTP packets:              6159
Total filtered packets:         19
```

Nesta segunda execução do VQcapture, foi reduzido de 40 para 20 o tamanho do buffer de compensação de jitter, acarretando na queda de qualidade estimada em todos os fatores de medição de qualidade de serviço na simulação de recepção do lado do NCE.

5.8. Desenvolvimento da Ferramenta

Alguns módulos da ferramenta já estão operacionais, como interpretador da sinalização H.323 e coletor de pacotes em modo promíscuo.

A implementação do módulo de interpretador da sinalização é relativamente complexa. O manual da biblioteca OpenH323 explica como invocar os seus objetos para desenvolver um cliente H.323, por exemplo. No entanto, não indica como chamar as

rotinas para interpretar pacotes já capturados. Para elucidar este problema foi necessário analisar o código fonte do programa Dump323. O Dump323 foi escrito pela equipe de desenvolvimento da própria biblioteca OpenH323, com o objetivo de depuração. O Dump323 recebe pacotes capturados pelo *tcpdump*, porém formatados em ASCII com a representação hexadecimal de cada byte. A análise do seu código fonte foi essencial para que a interpretação de pacotes H.323 fosse possível.

A programação de captura de pacotes em modo promíscuo foi mais simples e direta. A documentação da *pcap* é bem didática e adequada, contendo exemplos de programação. A princípio, a idéia era aproveitar o código fonte do *tcpdump*. Porém o *tcpdump* é uma implementação mais completa do que o módulo necessitava. Optou-se por desenvolver um programa mais simples, a partir dos protótipos fornecidos pelos exemplos dos tutoriais.

O entendimento do uso da biblioteca VQmon também consumiu um tempo de pesquisa. A biblioteca possui muitas funções e seus exemplos são muito simples, sem grande riqueza de detalhes.

Os módulos restantes serão desenvolvidos em um curto espaço de tempo, tornando a ferramenta completamente operacional. Cabe ressaltar que a existência de módulos básicos, com a adição da biblioteca VQmon, já permite a criação de um protótipo operacional ferramenta para testes, mesmo que sem todas as características propostas.

Capítulo 6

Conclusões

O presente trabalho apresentou duas ferramentas de monitoração de qualidade de redes VoIP, baseadas no protocolo de sinalização H.323.

No capítulo 2 foram introduzidos os conceitos sobre os protocolos e a arquitetura da recomendação ITU-T H.323. Também foram descritos os protocolos RTP e RTCP, fundamentais para obtenção dos dados necessários para avaliação da qualidade de redes VoIP.

O capítulo 3 descreveu os fatores que influenciam diretamente na qualidade de uma chamada, bem como métodos de avaliação desta qualidade.

A implementação de uma ferramenta de monitoração ativa foi descrita no capítulo 4. Esta ferramenta gera o seu próprio tráfego de voz para avaliar a qualidade sendo obtida. A arquitetura da ferramenta é composta por três módulos principais. Um módulo é responsável pela realização das chamadas e um outro módulo é responsável pelo seu recebimento. Ambos geram arquivos que contém todo o processo de sinalização e de troca de mensagens de mídia que ocorrem durante as ligações simuladas. Um terceiro módulo é responsável pela transferência de todos esses arquivos gerados e sua consolidação. Os programas foram desenvolvidos utilizando a linguagem de programação C++ e PERL e empregando os objetos disponibilizados pela biblioteca OpenH323. A visualização dos dados coletados é feita através da *web* utilizando páginas criadas em *javascript*. Essa interface *web* permite a seleção de ligações feitas por critérios específicos de QoS e a qualidade das conexões pode ser verificada, através da reprodução audível das mensagens armazenadas.

Ainda no capítulo 4 foi descrito o processo de utilização da ferramenta no *backbone* da RNP. A fase de coleta de dados reais utilizando essa ferramenta foi feita durante um período de aproximadamente 2 meses, através da instalação dos módulos receptores em diversos pontos da RNP espalhados geograficamente por todo o território nacional. Esta distribuição permitiu uma visão geral do comportamento do *backbone* da

RNP em relação a ligações VoIP. As ligações foram feitas utilizando os *codecs* G.711, G.723 e GSM em intervalos regulares de uma hora entre cada ligação simulada. A qualidade obtida em ligações com o padrão G.711 se mostraram de boa qualidade, mesmo em condições adversas. Os dados coletados com o uso dessa ferramenta podem ser observados na página do laboratório VoIP do NCE, http://www.voip.nce.ufrj.br.

O capítulo 5 descreve uma ferramenta para monitoração das ligações VoIP de forma passiva, ou não-intrusiva. Diferentemente da ferramenta descrita no capítulo 4, esta ferramenta não gera o seu próprio tráfego. Os dados são coletados a partir de ligações reais por um programa que captura os pacotes enquanto trafegam pela rede. Os pacotes capturados de sinalização H.323 são interpretados, e as portas dinâmicas usadas pelo fluxos de mídia RTP/RTCP de cada conexão são identificadas. Os dados relevantes de cada pacote RTP/RTCP capturado são utilizados para o cálculo em tempo real da qualidade estimada de cada ligação em andamento. Esses dados também são armazenados em uma MIB, permitindo que estações de gerência SNMP possam consultá-las. A MIB adotada é compatível com a proposta da arquitetura RAQMON feita muito recentemente pelo RMON *Working Group* do IETF. Outra característica da ferramenta é a capacidade de enviar *traps* SNMP para uma estação de gerência quando a qualidade estiver estive ultrapassando um limite previamente especificado. A integração dessa ferramenta com módulos de avaliação objetiva permitirá a monitoração da qualidade das ligações em tempo real.

A grande vantagem da implementação da ferramenta de monitoração passiva, proposta no capítulo 5, será permitir a avaliação da qualidade de ligações sendo efetivadas através de *gateways* out *gatekeepers* atuando em modo *proxy*. Esses dois casos serão muito comuns no cenário proposto para o projeto piloto VoIP. Foi testada e integrada ao ambiente de testes da segunda ferramenta a biblioteca VQmon, desenvolvida pela empresa Telchemy, especificamente para gerar estimativas de MOS baseada no Extended E-Model.

As ferramentas descritas nesta dissertação serão utilizadas no projeto piloto VoIP da Rede Nacional de Pesquisa, em âmbito nacional. A primeira ferramenta já vem sendo empregada com sucesso, permitindo traçar uma linha base da qualidade obtida por serviços multimídia no *backbone* da RNP. Uma outra fase de coleta utilizando essa

ferramenta será iniciada brevemente, quando políticas de QoS, essenciais para o funcionamento adequado de conexões VoIP, forem implantadas no *backbone* da RNP2.

Através das ferramentas implementadas, novos trabalhos e estudos poderão ser realizados, como a implantação e a comparação de diferentes modelos de cálculo de MOS, bem como o desenvolvimento de novos modelos de avaliação objetiva.

O presente trabalho tem prosseguimento natural com a implantação da sinalização SIP para as ferramentas desenvolvidas, com particular interesse para a segunda ferramenta, de análise não-intrusiva. Outro trabalho será a integração da ferramenta de monitoração passiva com a ferramenta de gerência sendo proposta em outro trabalho do laboratório VoIP do NCE. Esta ferramenta também poderá ser usada como apoio para o estudo de engenharia de tráfego pró-ativa, sendo alimentada pela avaliação de QoS instantânea de conexões VoIP na rede.

Os dados coletados por estas ferramentas formarão um banco de dados com informações sobre o perfil de utilização do serviço voltado para a realidade brasileira, permitindo que novas pesquisas e estudos sejam realizados a partir dessa massa de dados.

Referências Bibliográficas

[1] BRADEN, R., ZHANG L., BERSON, S., HERZOG S., JAMIN S., *Resource ReSerVation Protocol (RSVP) -- Version 1 Functional Specification*, RFC2205, Setembro 1997

[2] BLAKE, S., BLACK, D., CARLSON M., DAVIES, E.,WANG Z., WEISS, W., *An Architecture for Differentiated Service,* RFC2475, Dezembro 1998

[3] MIRAS, D., *A Survey of Network QoS Needs of Advanced Internet Applications, Internet2 QoS WG*, 2002 , obtido em http://qos.internet2.edu/wg/apps/fellowship/Docs/Internet2AppsQoSNeeds.pdf. Acesso em Agosto 2003

[4] ANDERSON, J., *Methods for measuring perceptual speech quality.* Agilent Technologies – White Paper. 2002, obtido em http://www.onenetworks.com/WhitePapers.asp . Acesso em Agosto 2003

[5] SCHULZRINNE, H., CASNER, S., FREDERICK, R., AND JACOBSON, V., *RTP: A Transport Protocol for Real-Time Applications*, RFC 3550, Julho 2003

[6] POSTEL, J., *User Datagram Protocol*, RFC0768, Agosto 1980

[7] POSTEL, J., *Transmission Control Protocol*, RFC0793, Setembro 1981

[8] COX, R., PERKINS, R., *Results of a Subjective Listening Test for G.711 with Frame Erasure Concealment*, AT&T T1A1,7/99-016, Maio 1999

[9] ITU-T Recommendation G.114, *General Characteristics of International Telephone Connections and International Telephone Circuits*, International Telecommunications Union, 1998

[10] Cisco Systems, *Understanding Delay In Packet Voice Networks*, Cisco Systems White Paper, 2001

[11] BOGER, Y., *Fine-tuning Voice over Packet services*, RADCOM white paper

[12] PRACHT, S., HARDMAN, D., *Voice Quality in Converging Telephony and IP Networks*, Agilent Technologies white paper, Outubro 2001

[13] ROSENBLUTH, J. H., *Testing the Quality of Connections of Connections having Time Varying Impairments*. Committee contribution T1A1.7/98-031

[14] FRANCE TELECOM, *Study of the relationship between instantaneous and overall subjective speech quality for time-varying quality speech sequences: influence of a recency effect*, ITU Study Group 12 D.139, Maio 2000

[15] ITU-T Recommendation P.800, *Methods for Subjective Determination of Transmission Quality*, International Telecommunication Union, Agosto 1996

[16] ITU-T Recommendation P.861, *Objective Quality Measurement of Telephone Band (300-3400 Hz) Speech Codecs*, International Telecommunication Union, Agosto 1996

[17] RIX, A. W., HOLLIER, M. P., *The perceptual analysis measurement system for robust end-to-end speech quality assessment,* IEEE ICASSP, Junho 2000.

[18] BEERENDS, J. G., RIX, A. W., HEKSTRA, A. P., HOLLIER, M. P., *Proposed Draft Recommendation P.862: Perceptual Evaluation of Speech Quality (PESQ), an Objective Method for End-to-end Speech Quality Assessment of Narrowband Telephone Networks and Speech Codecs,* ITU-T, Study Group 12, Maio 2000

[19] ITU-T Recommendation G.107, *The E Model, A Computational Model for use in Transmission Planning*, International Telecommunication Union, Maio 2000

[20] http://www.telchemy.com/references/voice_quality.html. Acesso em Agosto 2003

[21] ITU-T Recommendation P.830, *Subjective Performance Assessment Of Telephone-Band And Wideband Digital Codecs*, International Telecommunication Union, Fevereiro 1996

[22] BEERENDS, J. G., STEMERDINK, J. A., *A perceptual speech quality measure based on a psychoacoustic sound representation.* Journal of Audio Engineering Society, 42:115–123, Março 1994

[23] ATKINSON, D.J., "Proposed Annex A to Recommendation P.861", ITU-T Study Group 12 Contribution 24 (COM 12-24-E), International Telecommunication Union, Dezembro 1997

[24] VORAN, S. *Objective estimation of perceived speech quality, Part I: Development of the measuring normalizing block technique; Part II: Evaluation of the measuring normalizing block technique*. IEEE Trans. on Speech and Audio Processing, 7(4):383–390, Julho 1999.

[25] BEERENDS, J. G., MEIJER, E. J., HEKSTRA, A. P., *Improvement of the P.861 Perceptual Speech Quality Measure*, Contribution COM 12-20 to ITU-T Study Group 12, Dezembro1997

[26] RIX, A. W., HOLLIER, M. P., *Comparison of Speech Quality Assessment Algorithms: BT PAMS, PSQM, PSQM+ and MNB*, ITU-T Study Group 12 Delayed Contribution 80 (COM 12-D80), International Telecommunication Union, Novembro 1998

[27] VORAN, S., *Objective Estimation of Perceived Speech Quality Using Measuring Normalizing Blocks*, NTIA Report 98-347, National Telecommunications and Information Administration, U.S Dept of Commerce, Abril 1998

[28] ETSI, *Speech Communication Quality for Mouth to Ear for 3.1 kHz Handset Telephony across Networks*, Technical Report ETR250, 1996

[29] MARKOPOULOU, A. P., TOBAGI, F. A., KARAM, M. J., *Assessment of VoIP Quality over Internet Backbone*, IEEE, 2002

[30] CLARK, A., *Extensions to the E Model to incorporate the effects of time varying packet loss and recency*, T1A1.1/2001-037, Abril 2001

[31] CLARK, A., *Comparison of extended E Model with PSSQ and PAMS*, T1A1.1/2001-027, Abril 2001

[32] CLARK, A., *Comparison of TS101 329-5 Annex E with E Model*, Tiphom#23 document 062, Julho 2001

[33] http://www.rnp.br. Acesso em Agosto 2003

[34] MARCONDES, C.A.C.; RODRIGUES, P.H.A.; DAVID, F.; COSTA, J.C.P. *Ambiente para Simulação e Monitoração de Ligações Telefônicas IP*. SBRC 2003 Simpósio Brasileiro de Redes de Computadores. Natal, Rio Grande do Norte, Maio 2003

[35] *Boletim Eletrônico NewsGeneration RNP2*. Volume 7, número 1. 14 de abril de 2003 http://www.rnp.br/newsgen/0301/voip.shtml. Acesso em Agosto 2003

[36] RMON MIB WG http://www.ietf.org/html.charters/rmonmib-charter.html. Acesso em Agosto 2003

[37] ITU http://www.itu.int/home/index.html. Acesso em Agosto 2003

[38] ROSENBERG, J., SCHULZRINNE, CAMARILLO, G. et al, *SIP: Session Initiation Protocol*, RFC3261, Junho 2002

[39] MMUSIC WG http://www.ietf.org/html.charters/mmusic-charter.html. Acesso em Agosto 2003

[40] IETF http://www.ietf.org/. Acesso em Agosto 2003

[41] SIP WG http://www.ietf.org/html.charters/sip-charter.html. Acesso em Agosto 2003

[42] DOUSKALIS, B., *IP Telephony: The Integration of Robust VoIP Services*, Prentice Hall, 2000

[43] CADSOW, J., *Foundations of Digital Signal Processing and Data Analysis*, Macmillan, 1987

[44] ITU-T Recommendation H.323, *Packet-based multimedia communications systems*, International Telecommunication Union, Setembro 1999

[45] ITU-T Recommendation E.164, *The international public telecommunication numbering plan*, International Telecommunication Union, Maio 1997

[46] STOYCHEV, M., http://kgb.cnsys.bg/voice. Acesso em Agosto 2003

[47] ESTATISTICAS RNP http://www.rnp.br/operacao/trafego/. Acesso em Agosto 2003

[48] MRTG, *Multi Router Traffic Grapher,* http://people.ee.ethz.ch/~oetiker/webtools/mrtg/. Acesso em Agosto 2003

[49] CPAN, *Comprehensive Perl Archive Network*, http://www.cpan.org/. Acesso em Agosto 2003

[50] TCPDUMP, http://www.tcpdump.org. Acesso em Agosto 2003

[51] ETHEREAL, http://www.ethereal.com. Acesso em Agosto 2003

[52] HERMANN, K., MARTEL, M., MATTEI M., REGINI, S., *Enterprise Network Assessment for VoIP Call Quality - Is it Adequate?*, Dezembro 2002

[53] MMDUMP, http://www.research.att.com/info/Projects/mmdump. Acesso em Agosto 2003

[54] WINPCAP, http://winpcap.polito.it/ . Acesso em Agosto 2003

[55] MERWE, J., CACERES, R., CHU. Y., SREENAN, C., "mmdump: A Tool for Monitoring Internet Multimedia Traffic", ACM SIGCOMM, Outubro 2000

[56] FRIEDMAN, T., CACERES, R., CLARK, A., *RTP Control Protocol Extended Reports (RTCP XR)*, Internet-Draft IETF draft-ietf-avt-rtcp-report-extns-06.txt., Maio 2003

[57] http://www.openh323.org . Acesso em Agosto 2003

[58] SIDDIQUI, A., ROMASCANU, D., GOLOVINSKY E., *Real-time Application Quality of Service Monitoring (RAQMON) Protocol Data Unit (PDU)*, IETF Internet-draft draft-ietf-rmonmib-raqmon-pdu-02.txt, Junho 2003

[59] _____, *Real-time Application Quality of Service Monitoring (RAQMON) Framework*, IETF Internet-draft draft-ietf-rmonmib-raqmon-framework-02.txt, Junho 2003

[60] _____, *Real-time Application Quality of Service Monitoring (RAQMON) MIB*, IETF Internet-draft draft-ietf-rmonmib-raqmon-mib-01.txt, Junho 2003

[61] WALDBUSSER, S., *Remote Network Monitoring Management Information Base*, RFC2819, Maio 2000

[62] SMI Network Management Private Enterprise Code, http://www.iana.org/assignments/enterprise-numbers . Acesso em Agosto 2003

[63] BEGEL, A., MCCANNE, S., GRAHAM, S. L. *BPF+: Exploiting Global Data-flow Optimization in a Generalized Packet Filter Architecture*. Proc. ACM SIGCOMM '99, Agosto 1999

[64] CLARK, A., Improving VoIP call quality with Embedded Monitoring, http://www.commsdesign.com, Setembro 2001

[65] BADDELEY, A. D., Human Memory: Theory and Practice. Psychology Press, Hove Erlbaum Associates, revised ed., 1997

[66] NetIQ Chariot, http://www.netiq.com/products/chr/default.asp. Acesso em Agosto 2003

[67] Agilent Voice Quality Tester (VQT), http://www.agilent.com/. Acesso em Agosto 2003

[68] Spirent Abacus2, http://www.spirentcom.com. Acesso em Agosto 2003

[69] Artiza, http://www.artizanetworks.com/products/voipana_index.html. Acesso em Agosto 2003

[70] ExpertNet, http://www.avaya.com. Acesso em Agosto 2003

[71] ROECK, G., Dial Control Management Information Base using SMIv2, RFC2128, Março 1997

[72] NET-SNMP http://www.net-snmp.org/. Acesso em Agosto 2003

ANEXOS

Apêndice A

Descrição dos campos do cabeçalho RTP e RTCP

Campo do Cabeçalho RTP	Tamanho (bits)	Descrição
Versão (V)	2	Especifica a versão do RTP sendo utilizada, atualmente com o valor 2.
Padding (P)	1	Indica a existência de bytes de preenchimento.
Extension (X)	1	Indica se existe uma extensão de cabeçalho após a parte fixa do cabeçalho (12 bytes).
CSRC Count (CC)	4	Indica o número de campos CSRC existentes após a parte fixa do cabeçalho (12 bytes).
Marker (M)	1	Depende da aplicação; no caso de VoIP, indica o início de uma seqüência de fala.
Payload Type (PT)	7	Indica o formato de codificação da mídia
Sequence Number	16	Número de seqüência do pacote, iniciado com um valor aleatório, este campo é incrementado a cada pacote transmitido. Possibilita ao receptor identificar a perda de um pacote.
Time Stamp	32	Permite a reprodução no lado receptor com o tempo corrigido. Representa o exato instante (formato NTP) da amostragem do primeiro octeto deste pacote.
Sending Source (SSRC)	32	Identificação do emissor, através de número aleatório único em uma dada sessão.
Contributing Sources (CSRC)	32	Emissores de uma sessão; no caso de VoIP somente usado em conferências indicando todos os SSRC participantes. Máximo de 15 itens

Tabela 7. Descrição resumida dos campos do cabeçalho RTP

Campo do Cabeçalho SENDER REPORT	*Tamanho (bits)*	*Descrição*
Versão (V)	2	Especifica a versão do RTP sendo utilizada, atualmente com o valor 2
Padding (P)	1	Indica a existência de bytes de preenchimento
Reception Report Count (RC)	5	Indica o número de blocos de recepção após a informação sobre o transmissor.
Payload Type (PT)	8	Identifica o pacote como RTCP SR (valor = 200)
Length	16	Tamanho do pacote (em múltiplos de 32 bits menos um)
Sending Source (SSRC)	32	Identificação do emissor do pacote RTCP, através de número aleatório, único em uma dada sessão.
NTP Time Stamp	64	Instante real em que o pacote foi transmitido. Captura o horário local de sistema da máquina fonte, no momento do envio da mensagem SR, usando a formatação do NTP (Network Time Protocol)
RTP Time Stamp	32	Permite a correlação dos tempos utilizados pelo RTP com o tempo real. Corresponde ao instante de tempo (normalmente múltiplo da unidade de amostragem de áudio) em que o áudio, pertencente ao último pacote RTP, foi enviado por determinado emissor.
Sender's Packet Count	32	Número total de pacotes RTP enviados pelo emissor desde o início da sessão até o momento em que um pacote SR foi gerado.
Sender's Byte Count	32	Número total de bytes dos pacotes RTP enviados desde o início da sessão, sem contar cabeçalhos e paddings

Tabela 8. Descrição dos campos do cabeçalho RTCP tipo SR

Campo do Cabeçalho RECEIVER REPORT	*Tamanho (bits)*	*Descrição*
SSRC_n	32	Identificação SSRC do emissor a qual este relatório de recepção se refere.
Fraction Lost	8	Número de pacotes RTP perdidos dividido pelo número de pacotes enviados, desde o último SR/RR recebido.
Cumulative Packets Lost	24	Número total de pacotes RTP perdidos desde o início da sessão. Calculado como número de pacotes esperado menos o número de pacotes atualmente recebidos. O número de pacotes esperado é definido pelo campo "extended highest sequence number received".
Extended Sequence Number Received	32	Maior número de seqüência do transmissor recebido pelo receptor em pacotes RTP
Inter-arrival Jitter (J)	32	Jitter calculado pelo receptor
Last SR (LSR) time stamp	32	NTP timestamp do último pacote SR recebido
Delay Since Last SR (DLSR)	32	Tempo passado entre a recepção do último SR recebido e o envio deste RR

Tabela 9. Descrição dos campos do cabeçalho RTCP tipo RR

Apêndice B

Produtos de monitoração comerciais disponíveis

Fabricante	Produto	Características
Agilent	*Voice Quality Tester*	Métodos objetivos usados: PESQ, PSQM+ e PAMS Métricas Analisadas: atraso de voz, (troubleshooting), eco, supressão de silêncio, VAD front-end clipping, hold-over time Análise de tom e ruído. Medição de perda de sinal comparando ruído, tom e voz. Medição da distorção do tom de DTMF
Agilent	*IP Telephony Analyzer*	Utiliza algoritmo proprietário para medição e avaliação de voz. Permite a reprodução do áudio a partir de pacotes VoIP para os codecs G.711, G.723, G.729a e G.729b, capacidade de reprodução de fluxos de voz RTP com ou sem jitter, analisa os valores de jitter e perda de pacotes do RTCP. Análise de CDR (Call Detail Records) de cada chamada. Decodificação em tempo real para interfaces LAN, WAN e ATM.
Avaya	*ExpertNet*	Utiliza algoritmo proprietário para medição e avaliação de voz. Capacidade de injeção de chamadas sintéticas (fluxos RTP), identifica o caminho exato feito por cada chamada. Faz análise de utilização de banda e balanceamento de carga. Monitora dispositivos SNMP.
NetIQ	*Chariot*	Utiliza uma versão modificada do E-model. Gera fluxos RTP para simular tráfego VoIP Medição de one-way delay, perda de pacotes, rajadas, jitter, e efeitos de codec. Permite a simulação de buffers de jitter Possui ferramenta de gerencia e diagnostico integrada. Implementa estratégia para testes periódicos e repetitivos, para análise de utilização de rede em momentos de pico. Possui módulo de certificação de equipamentos. Integração com ferramentas de gerência SNMP como HP Open View

Fabricante	Produto	Características
Spirent	*SmartBits VOIPQOS*	Utiliza PSQM para análise objetiva. Gera pacotes RTP para diversos codecs Analisa IP/DiffServ & IP/TOS precedence por fluxo. Suporte para ITU-T G.711 μ-Law, G.711 A-Law, G.723.1, G.726, and G.729 Codecs. Analisa a qualidade da voz e de dados da rede. Capacidade de gerar e analisar até 500 fluxos de voz por porta.
Spirent	*IP Wave Network Impairment Emulator*	Simula efeitos fatores de rede, como: atraso, duplicação, reordenação, fragmentação, perda, por exemplo**.**
RADCOM	*Performer*	Analise PESQ e PAMS entre redes de diferentes tecnologias interligadas por rede PSTN Possui módulos de simulação para sinalização H323, SIP, MGCP, permitindo emulação de setup de chamadas, milhares de chamadas simultâneas e teste de carga em servidores de autenticação e registro. Possui capacidade de gerar milhões de sessões de sinalização por hora. Analisa dados de sinalização e mídia gerados por H.323/MGCP/SIP.
Telchemy	*VQMon End Point and Stream Analyzers*	Utiliza método proprietário objetivo e não intrusivo em tempo real, baseado no modelo de avaliação do E-Model. Avalia chamadas reais. Leva em consideração perda de pacotes em rajadas, não usando apenas o seu valor médio. Possui módulo para ser incorporado a endpoints por desenvolvedores. Analisa o descarte de buffer de jitter no lado receptor.

Apêndice C

Definição da RAQMON MIB

```
  --
  -- The original version of this MIB was
created by Richard Smith from
  -- Avaya Labs Australia

        RAQMON-MIB DEFINITIONS ::=
BEGIN

          IMPORTS
              OBJECT-GROUP,
NOTIFICATION-GROUP
                  FROM SNMPv2-CONF
              mib-2, enterprises, IpAddress,
Integer32, Unsigned32,
              Gauge32, Counter32, OBJECT-
TYPE,
                    MODULE-IDENTITY,
NOTIFICATION-TYPE
                        FROM SNMPv2-SMI
                rmon          FROM RMON-MIB
                RowStatus, TEXTUAL-
CONVENTION, TruthValue

                  FROM SNMPv2-TC
              Utf8String
                  FROM SYSAPPL-MIB
                Dscp
               FROM DIFFSERV-DSCP-TC;



          raqmon MODULE-IDENTITY
              LAST-UPDATED
"200305260000Z"     -- May 26, 2003
              ORGANIZATION
                  "IETF RMON MIB Working
Group"
              CONTACT-INFO
                  "
         WG Charter:

http://www.ietf.org/html.charters/rmonmib-
charter.html

         Mailing lists:
          General Discussion: rmonmib@ietf.org
          To Subscribe: rmonmib-
requests@ietf.org
          In Body: subscribe your_email_address

         Chair: Andy Bierman
          Cisco Inc.
          Email: abierman@cisco.com

         Editor: Dan Romascanu
          Avaya Israel
          Email:  dromasca@avaya.com
                  "
          DESCRIPTION
            "Real-Time Application QoS
Monitoring MIB"
          REVISION "200305260000Z"     -- May
26, 2003
          DESCRIPTION
            "The MIB module for transferring the
Real-Time
             Application QoS Monitoring
(RAQMON) information
             from a RAQMON collector to a
management application.

             Copyright (c) The Internet Society
(2003). This version of
             this MIB module is part of RFC yyyy;
See the RFC itself
             for full legal notices.
  -- RFC Ed.: replace yyyy with the actual RFC
number & remove this notice.
  "
          ::= { rmon 6889 }  -- need to assign a
'real' branch under RMON



  --
  -- Type definitions
  --

          Duration ::= Unsigned32


  --
  -- Textual conventions
  --

          RaqmonDateAndTime ::= TEXTUAL-
CONVENTION
              DISPLAY-HINT
                  "2d-1d-1d,1d:1d:1d"
              STATUS current
              DESCRIPTION
                  "A date-time specification in
Coordinated Universal
                  Time (UTC).
                  This definition is used rather than
DateAndTime
                  from SNMPv2-TC or
ExtUTCTime from SMIv2
                  since a fixed length field in UTC
was preferred
                  for use as an INDEX.
```

```
                field  octets  contents
range
                -----  ------  --------              -----
                 1      1-2   year
0..65536
                 2       3    month
1..12
                 3       4    day                   1..31
                 4       5    hour                  0..23
                 5       6    minutes
0..59
                 6       7    seconds
0..59

                For example, Tuesday May 26,
1992 at 1:30:15 UTC
                would be displayed as:

                             1992-5-26,13:30:15."
            SYNTAX OCTET STRING (SIZE
(7))


  --
  -- Node definitions
  --

        raqmonEvents OBJECT IDENTIFIER
::= { raqmon 0 }

        raqmonSessionAlarm
NOTIFICATION-TYPE
            OBJECTS { raqmonParticipantAddr,
                raqmonParticipantName,
                raqmonParticipantPeerAddr,
raqmonQosRTT,
                raqmonQosJitter,
                raqmonQosLostPackets,
raqmonQosRcvdPackets }
            STATUS current
            DESCRIPTION
                "A notification generated by an
entry in the
                raqmonSessionExceptionTable."
            ::= { raqmonEvents 1 }


        raqmonMIBObjects OBJECT
IDENTIFIER ::= { raqmon 1 }

        raqmonSession OBJECT IDENTIFIER
::= { raqmonMIBObjects 1 }

        raqmonParticipantTable OBJECT-
TYPE
            SYNTAX SEQUENCE OF
RaqmonParticipantEntry
            MAX-ACCESS not-accessible
            STATUS current
            DESCRIPTION
                "Table of information about
participants in sessions.
                Contains both active and closed
(terminated) sessions.
                "
            ::= { raqmonSession 1 }

        raqmonParticipantEntry OBJECT-
TYPE
            SYNTAX RaqmonParticipantEntry
            MAX-ACCESS not-accessible
            STATUS current
            DESCRIPTION
                "Each entry contains information
from a single session
                participant.
                Rows are removed for inactive
sessions
                when implementation
                specific age or space limits are
reached.
                "
            INDEX {
raqmonParticipantStartDate,
raqmonParticipantIndex }
            ::= { raqmonParticipantTable 1 }

        RaqmonParticipantEntry ::=
            SEQUENCE {
                raqmonParticipantStartDate
                    RaqmonDateAndTime,
                raqmonParticipantIndex
                    Integer32,
                raqmonParticipantAddr
                    IpAddress,
                raqmonParticipantSendPort
                    Unsigned32,
                raqmonParticipantRecvPort
                    Unsigned32,
                raqmonParticipantSetupDelay
                    Unsigned32,
                raqmonParticipantName
                    Utf8String,
                raqmonParticipantTool
                    Utf8String,
                raqmonParticipantQosCount
                    Unsigned32,
                raqmonParticipantEndDate
                    RaqmonDateAndTime,
                raqmonParticipantRcvdPT
                    Unsigned32,
                raqmonParticipantSentPT
                    Unsigned32,
                raqmonParticipantActive
                    TruthValue,
```

```
            raqmonParticipantPeerIndex
                OCTET STRING,
            raqmonParticipantPeerAddr
                IpAddress,
            raqmonParticipantSrcLayer2
                Unsigned32,
            raqmonParticipantDestLayer2
                Unsigned32,
            raqmonParticipantSrcLayer3
                Dscp,
            raqmonParticipantDestLayer3
                Dscp,
            raqmonParticipantCPUMean
                Unsigned32,
            raqmonParticipantCPUMin
                Unsigned32,
            raqmonParticipantCPUMax
                Unsigned32,
            raqmonParticipantMemoryMean
                Unsigned32,
            raqmonParticipantMemoryMin
                Unsigned32,
            raqmonParticipantMemoryMax
                Unsigned32,
            raqmonParticipantRTTMean
                Gauge32,
            raqmonParticipantRTTMin
                Gauge32,
            raqmonParticipantRTTMax
                Gauge32,
            raqmonParticipantJitterMean
                Gauge32,
            raqmonParticipantJitterMin
                Gauge32,
            raqmonParticipantJitterMax
                Gauge32,
            raqmonParticipantOWDMean
                Gauge32,
            raqmonParticipantOWDMin
                Gauge32,
            raqmonParticipantOWDMax
                Gauge32,
            raqmonParticipantPackets
                Counter32,
            raqmonParticipantLostPackets
                Counter32
        }

    raqmonParticipantStartDate OBJECT-
TYPE
        SYNTAX RaqmonDateAndTime
        MAX-ACCESS not-accessible
        STATUS current
        DESCRIPTION
            "The date and time of this entry.
            It will be the date and time
            of the first report received."
        ::= { raqmonParticipantEntry 1 }

    raqmonParticipantIndex OBJECT-
TYPE
        SYNTAX Integer32
(1..2147483647)
        MAX-ACCESS not-accessible
        STATUS current
        DESCRIPTION
            "The index of the conceptual row
which is for
            SNMP purposes
            only and has no relation to any
protocol value.
            There is
            no requirement that these rows
are created or maintained
            sequentially. The index will be
unique for a
            particular date and time."
        ::= { raqmonParticipantEntry 2 }

    raqmonParticipantAddr OBJECT-TYPE
        SYNTAX IpAddress
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
            "IP Address of the participant for
this session."
        ::= { raqmonParticipantEntry 3 }

    raqmonParticipantSendPort OBJECT-
TYPE
        SYNTAX Unsigned32 (0..65535)
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
            "Port from which session data is
sent."
        ::= { raqmonParticipantEntry 4 }

    raqmonParticipantRecvPort OBJECT-
TYPE
        SYNTAX Unsigned32 (0..65535)
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
            "Port on which session data is
received."
        ::= { raqmonParticipantEntry 5 }

    raqmonParticipantSetupDelay
OBJECT-TYPE
        SYNTAX Unsigned32
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
            "Session setup time in
milliseconds."
```

```
            ::= { raqmonParticipantEntry 6 }

        raqmonParticipantName OBJECT-
TYPE
            SYNTAX Utf8String
            MAX-ACCESS read-only
            STATUS current
            DESCRIPTION
                "The data source name for the
participant."
            ::= { raqmonParticipantEntry 7 }

        raqmonParticipantTool OBJECT-TYPE
            SYNTAX Utf8String
            MAX-ACCESS read-only
            STATUS current
            DESCRIPTION
                "A string giving the name and
possibly version
                of the application
                generating the stream, e.g.,
videotool 1.2.
                This information may be
                useful for debugging purposes
and is similar
                to the Mailer or Mail-
                System-Version SMTP headers.
The tool value
                is expected to remain
                constant for the duration of the
session."
            ::= { raqmonParticipantEntry 8 }

        raqmonParticipantQosCount OBJECT-
TYPE
            SYNTAX Unsigned32
            MAX-ACCESS read-only
            STATUS current
            DESCRIPTION
                "Count of entries in the QOS
history table
                for this participant and session."
            ::= { raqmonParticipantEntry 9 }

        raqmonParticipantEndDate OBJECT-
TYPE
            SYNTAX RaqmonDateAndTime
            MAX-ACCESS read-only
            STATUS current
            DESCRIPTION
                "The date and time of the last
report received."
            ::= { raqmonParticipantEntry 10 }

        raqmonParticipantRcvdPT OBJECT-
TYPE
            SYNTAX Unsigned32 (0..127)
            MAX-ACCESS read-only
            STATUS current
            DESCRIPTION
                "Payload type of received
packets."
            ::= { raqmonParticipantEntry 11 }

        raqmonParticipantSentPT OBJECT-
TYPE
            SYNTAX Unsigned32 (0..127)
            MAX-ACCESS read-only
            STATUS current
            DESCRIPTION
                "Payload type of sent packets."
            ::= { raqmonParticipantEntry 12 }

        raqmonParticipantActive OBJECT-
TYPE
            SYNTAX TruthValue
            MAX-ACCESS read-only
            STATUS current
            DESCRIPTION
                "Value 'true' ndicates that the
session
                    for this participant is active
(open).
                Value 'false' indicates that the
session
                is closed (terminated).
                "
            ::= { raqmonParticipantEntry 13 }

        raqmonParticipantPeerIndex OBJECT-
TYPE
            SYNTAX OCTET STRING (SIZE
(0 | 11))
            MAX-ACCESS read-only

            STATUS current
            DESCRIPTION
                "The index of the corresponding
entry in this table for
                the other peer participant. If there
is no
                such entry then the
                value will be a zero length string.
Otherwise it will
                be a string of length 11 consisting
of
                the raqmonParticipantStartDate
                octet string appended with an
octet string of length 4
                containing
raqmonParticipantIndex (most
                significant octet first).
                Note, the entry may no longer
exist even if the index is
                not zero length since the entry
may have
```

```
                been deleted due
                to implementation defined limits
being exceeded.
                "
            ::= { raqmonParticipantEntry 14 }

        raqmonParticipantPeerAddr OBJECT-
TYPE
            SYNTAX IpAddress
            MAX-ACCESS read-only
            STATUS current
            DESCRIPTION
                "IP address of peer endpoint."
            ::= { raqmonParticipantEntry 15 }

        raqmonParticipantSrcLayer2 OBJECT-
TYPE
            SYNTAX Unsigned32 (0..7)
            MAX-ACCESS read-only
            STATUS current
            DESCRIPTION
                "Source Layer 2 priority"
            ::= { raqmonParticipantEntry 16 }

        raqmonParticipantDestLayer2
OBJECT-TYPE
            SYNTAX Unsigned32 (0..7)
            MAX-ACCESS read-only
            STATUS current
            DESCRIPTION
                "Destination Layer 2 priority."
            ::= { raqmonParticipantEntry 17 }

        raqmonParticipantSrcLayer3 OBJECT-
TYPE
            SYNTAX Dscp
            MAX-ACCESS read-only
            STATUS current
            DESCRIPTION
                "Source Layer 3 DSCP value"
            ::= { raqmonParticipantEntry 18 }

        raqmonParticipantDestLayer3
OBJECT-TYPE
            SYNTAX Dscp
            MAX-ACCESS read-only
            STATUS current
            DESCRIPTION
                "Destination Layer 3 DSCP
value"
            ::= { raqmonParticipantEntry 19 }

        raqmonParticipantCPUMean OBJECT-
TYPE
            SYNTAX Unsigned32 (0..100)
            MAX-ACCESS read-only
            STATUS current
            DESCRIPTION
                "Mean CPU utilization as a
percentage."
            ::= { raqmonParticipantEntry 20 }

        raqmonParticipantCPUMin OBJECT-
TYPE
            SYNTAX Unsigned32 (0..100)
            MAX-ACCESS read-only
            STATUS current
            DESCRIPTION
                "Minimum CPU utilization as a
percentage."
            ::= { raqmonParticipantEntry 21 }

        raqmonParticipantCPUMax OBJECT-
TYPE
            SYNTAX Unsigned32 (0..100)
            MAX-ACCESS read-only
            STATUS current
            DESCRIPTION
                "Maximum CPU utilization as a
percentage."
            ::= { raqmonParticipantEntry 22 }

        raqmonParticipantMemoryMean
OBJECT-TYPE
            SYNTAX Unsigned32 (0..100)
            MAX-ACCESS read-only
            STATUS current
            DESCRIPTION
                "Mean memory utilization as a
percentage."
            ::= { raqmonParticipantEntry 23 }

        raqmonParticipantMemoryMin
OBJECT-TYPE
            SYNTAX Unsigned32 (0..100)
            MAX-ACCESS read-only
            STATUS current
            DESCRIPTION
                "Minimum memory utilization as
a percentage."
            ::= { raqmonParticipantEntry 24 }

        raqmonParticipantMemoryMax
OBJECT-TYPE
            SYNTAX Unsigned32 (0..100)
            MAX-ACCESS read-only
            STATUS current
            DESCRIPTION
                "Maximum memory utilization as
a percentage."
            ::= { raqmonParticipantEntry 25 }

        raqmonParticipantRTTMean OBJECT-
TYPE
            SYNTAX Gauge32
            MAX-ACCESS read-only
```

```
            STATUS current
            DESCRIPTION
                "Mean round trip time (RTT)
over the entire session."
            ::= { raqmonParticipantEntry 26 }

        raqmonParticipantRTTMin OBJECT-
TYPE
            SYNTAX Gauge32
            MAX-ACCESS read-only
            STATUS current
            DESCRIPTION
                "Minimum round trip time (RTT)
over the entire session."
            ::= { raqmonParticipantEntry 27 }

        raqmonParticipantRTTMax OBJECT-
TYPE
            SYNTAX Gauge32
            MAX-ACCESS read-only
            STATUS current
            DESCRIPTION
                "Maximum round trip time (RTT)
over the entire session."
            ::= { raqmonParticipantEntry 28 }

        raqmonParticipantJitterMean OBJECT-
TYPE
            SYNTAX Gauge32
            MAX-ACCESS read-only
            STATUS current
            DESCRIPTION
                "Mean jitter over the entire
session."
            ::= { raqmonParticipantEntry 29 }

        raqmonParticipantJitterMin OBJECT-
TYPE
            SYNTAX Gauge32
            MAX-ACCESS read-only
            STATUS current
            DESCRIPTION
                "Minimum jitter over the entire
session."
            ::= { raqmonParticipantEntry 30 }

        raqmonParticipantJitterMax OBJECT-
TYPE
            SYNTAX Gauge32
            MAX-ACCESS read-only
            STATUS current
            DESCRIPTION
                "Maximim jitter over the entire
session."
            ::= { raqmonParticipantEntry 31 }

         raqmonParticipantOWDMean
OBJECT-TYPE
            SYNTAX Gauge32
            MAX-ACCESS read-only
            STATUS current
            DESCRIPTION
                "Mean one way delay (OWD)
over the entire session."
            ::= { raqmonParticipantEntry 32 }

        raqmonParticipantOWDMin OBJECT-
TYPE
            SYNTAX Gauge32
            MAX-ACCESS read-only
            STATUS current
            DESCRIPTION
                "Minimum one way delay
(OWD) over the entire session."
            ::= { raqmonParticipantEntry 33 }

        raqmonParticipantOWDMax OBJECT-
TYPE
            SYNTAX Gauge32
            MAX-ACCESS read-only
            STATUS current
            DESCRIPTION
                "Maximum one way delay
(OWD) over the entire session."
            ::= { raqmonParticipantEntry 34 }

        raqmonParticipantPackets OBJECT-
TYPE
            SYNTAX Counter32
            MAX-ACCESS read-only
            STATUS current
            DESCRIPTION
                "Count of packets received for the
entire session."
            ::= { raqmonParticipantEntry 35 }

        raqmonParticipantLostPackets
OBJECT-TYPE
            SYNTAX Counter32
            MAX-ACCESS read-only
            STATUS current
            DESCRIPTION
                "Count of packets lost by this
                receiver for the entire session."
            ::= { raqmonParticipantEntry 36 }

        raqmonQosTable OBJECT-TYPE
            SYNTAX SEQUENCE OF
RaqmonQosEntry
            MAX-ACCESS not-accessible
            STATUS current
            DESCRIPTION
                "Table of historical information
                about quality of service
                data during sessions.
                "
```

```
            ::= { raqmonSession 2 }

        raqmonQosEntry OBJECT-TYPE
            SYNTAX RaqmonQosEntry
            MAX-ACCESS not-accessible
            STATUS current
            DESCRIPTION
                "Each entry contains information
from
                a single RAQMON packet.
                Rows are removed for
                inactive sessions when
implementation
                specific time or space limits are
reached.
                "
            INDEX {
raqmonParticipantStartDate,
                        raqmonParticipantIndex,
raqmonQosTime }

            ::= { raqmonQosTable 1 }

        RaqmonQosEntry ::=
            SEQUENCE {
                raqmonQosTime
                    Duration,
                raqmonQosRTT
                    Gauge32,
                raqmonQosJitter
                    Gauge32,
                raqmonQosRcvdPackets
                    Integer32,
                raqmonQosRcvdOctets
                    Integer32,
                raqmonQosSentPackets
                    Integer32,
                raqmonQosSentOctets
                    Integer32,
                raqmonQosLostPackets
                    Integer32,
                raqmonQosRsvpStatus
                    INTEGER
             }

        raqmonQosTime OBJECT-TYPE
            SYNTAX Duration
            MAX-ACCESS read-only
            STATUS current
            DESCRIPTION
                "Time of this entry measured
                from the start of the
corresponding
                participant session."
            ::= { raqmonQosEntry 1 }

        raqmonQosRTT OBJECT-TYPE
            SYNTAX Gauge32
            MAX-ACCESS read-only
            STATUS current
            DESCRIPTION
                "The round trip time.
                Will contain the previous value if
there was no report
                for this time (or 2^32 - 1 if value
never reported).
                "
            ::= { raqmonQosEntry 2 }

        raqmonQosJitter OBJECT-TYPE
            SYNTAX Gauge32
            MAX-ACCESS read-only
            STATUS current
            DESCRIPTION
                "An estimate of delay variation
                as observed by this receiver.
                Will contain the previous value if
there was no report
                for this time (or 2^32 - 1 if value
never reported).
                "
            ::= { raqmonQosEntry 3 }

        raqmonQosRcvdPackets OBJECT-
TYPE
            SYNTAX Integer32
            MAX-ACCESS read-only
            STATUS current
            DESCRIPTION
                "Count of packets received by
this receiver
                since the last entry containing a
value for this field.
                This is not a cumulative
                value since the start of the
session.
                Set to -1 if value not reported for
this time.
                "
            ::= { raqmonQosEntry 4 }

        raqmonQosRcvdOctets OBJECT-TYPE
            SYNTAX Integer32
            MAX-ACCESS read-only
            STATUS current
            DESCRIPTION
                "Count of octets received by this
receiver
                since the last entry containing a
value for this field.
                This is not a cumulative value
                since the start of the session.
                Set to -1 if value not reported for
this time.
                "
            ::= { raqmonQosEntry 5 }
```

```
        raqmonQosSentPackets OBJECT-
TYPE
            SYNTAX Integer32
            MAX-ACCESS read-only
            STATUS current
            DESCRIPTION
                "Count of packets sent since the
last entry containing a
                value for this field.
                This is not a cumulative value
since
                the start of the session.
                Set to -1 if value not reported for
this time.
                "
            ::= { raqmonQosEntry 6 }

        raqmonQosSentOctets OBJECT-TYPE
            SYNTAX Integer32
            MAX-ACCESS read-only
            STATUS current
            DESCRIPTION
                "Count of octets sent since the
last entry containing a
                value for this field.
                This is not a cumulative value
                since the start of the session.
                Set to -1 if value not reported for
this time.
                "
            ::= { raqmonQosEntry 7 }

        raqmonQosLostPackets OBJECT-
TYPE
            SYNTAX Integer32
            MAX-ACCESS read-only
            STATUS current
            DESCRIPTION
                "A count of packets lost as
observed by this receiver
                since the last entry containing a
value for this field.
                Set to -1 if value not reported for
this time."
            ::= { raqmonQosEntry 8 }

        raqmonQosRsvpStatus OBJECT-TYPE
            SYNTAX INTEGER
                {
                unknown(-1),
                notInUse(0),
                disabled(1),
                reservationPending(2),
                reservationFailed(3),
                reservationSuccess(4)
                }
            MAX-ACCESS read-only
            STATUS current
            DESCRIPTION
                "The RSVP status.
                Will contain the previous value if
there was no report
                for this time (or <unknown> if no
                value was ever reported)."
            ::= { raqmonQosEntry 9 }

        raqmonParticipantAddrTable OBJECT-
TYPE
            SYNTAX SEQUENCE OF
RaqmonParticipantAddrEntry
            MAX-ACCESS not-accessible
            STATUS current
            DESCRIPTION
                "Maps raqmonParticipantAddr to
the index of the
                raqmonParticipantTable. This
table allows management
                applications to find entries
                sorted by raqmonParticipantAddr
                rather than
raqmonParticipantStartDate."
            ::= { raqmonSession 3 }

        raqmonParticipantAddrEntry OBJECT-
TYPE
            SYNTAX
RaqmonParticipantAddrEntry
            MAX-ACCESS not-accessible
            STATUS current
            DESCRIPTION
                "Each entry corresponds to
exactly one entry in the
                raqmonParticipantEntry - the
entry containing the
                index pair
rraqmonParticipantStartDate,
                 raqmonParticipantIndex."
            INDEX { raqmonParticipantAddr,

raqmonParticipantStartDate,
                        raqmonParticipantIndex }
            ::= { raqmonParticipantAddrTable 1
}


        RaqmonParticipantAddrEntry ::=
            SEQUENCE {
                raqmonParticipantAddrEndDate
                    RaqmonDateAndTime
             }

        raqmonParticipantAddrEndDate
OBJECT-TYPE
            SYNTAX RaqmonDateAndTime
            MAX-ACCESS read-only
```

```
            STATUS current
            DESCRIPTION
                "The value of
                raqmonParticipantEndDate for
the corresponding
                raqmonParticipantEntry."
            ::= { raqmonParticipantAddrEntry 1
}

        raqmonException OBJECT
IDENTIFIER ::= { raqmonMIBObjects 2 }

        raqmonSessionExceptionTable
OBJECT-TYPE
            SYNTAX SEQUENCE OF
RaqmonSessionExceptionEntry
            MAX-ACCESS not-accessible
            STATUS current
            DESCRIPTION
                "This table creates filters so that a
                management station can
                get immediate notification of a
                session that has had poor
                quality of service."
            ::= { raqmonException 2 }

        raqmonSessionExceptionEntry
OBJECT-TYPE
            SYNTAX
RaqmonSessionExceptionEntry
            MAX-ACCESS not-accessible
            STATUS current
            DESCRIPTION
                "A conceptual row in the
raqmonSessionExceptionTable."
            INDEX {
raqmonSessionExceptionIndex }
            ::= { raqmonSessionExceptionTable
1 }

        RaqmonSessionExceptionEntry ::=
            SEQUENCE {
                raqmonSessionExceptionIndex
                    Unsigned32,

raqmonSessionExceptionJitterThreshold
                    Unsigned32,

raqmonSessionExceptionRttThreshold
                    Unsigned32,

raqmonSessionExceptionLostPacketsThreshold
                    Unsigned32,

raqmonSessionExceptionRowStatus
                    RowStatus
             }

        raqmonSessionExceptionIndex
OBJECT-TYPE
            SYNTAX Unsigned32 (1..65535)
            MAX-ACCESS not-accessible
            STATUS current
            DESCRIPTION
                "An index that uniquely identifies
an
                entry in the
raqmonSessionExceptionTable."
            ::= { raqmonSessionExceptionEntry
2 }


raqmonSessionExceptionJitterThreshold
OBJECT-TYPE
            SYNTAX Unsigned32
            MAX-ACCESS read-write
            STATUS current
            DESCRIPTION
                "Threshold for jitter in units of
milliseconds.
                The value during a session must
be greater than or
                equal to this value for an
exception to be created."
            ::= { raqmonSessionExceptionEntry
3 }

        raqmonSessionExceptionRttThreshold
OBJECT-TYPE
            SYNTAX Unsigned32
            MAX-ACCESS read-write
            STATUS current
            DESCRIPTION
                "Threshold for round trip time in
units of milliseconds.
                The value during a session must
be greater than or
                equal to this value for an
exception to be created."
            ::= { raqmonSessionExceptionEntry
4 }


raqmonSessionExceptionLostPacketsThreshold
OBJECT-TYPE
            SYNTAX Unsigned32 (0..1000)
            MAX-ACCESS read-write
            STATUS current
            DESCRIPTION
                "Threshold for lost packets in
                units of tenth of a percent.
                The value during a session must
be greater than or
                equal to this value for an
exception to be created."
```

```
            ::= { raqmonSessionExceptionEntry
5 }

        raqmonSessionExceptionRowStatus
OBJECT-TYPE
            SYNTAX RowStatus
            MAX-ACCESS read-create

            STATUS current
            DESCRIPTION
                "Value of 'active' when
exceptions
                are being monitored by the
system.
                A newly-created conceptual row
must have the all
                read-create objects initialized
                before becoming 'active'.
                A conceptual row that is in
                the 'notReady' or 'notInService'
                state MAY be removed after 5
minutes."
            ::= { raqmonSessionExceptionEntry
7 }

        raqmonConfig OBJECT IDENTIFIER
::= { raqmonMIBObjects 3 }

        raqmonConfigPort OBJECT-TYPE
            SYNTAX Unsigned32 (0..65535)
            MAX-ACCESS read-write
            STATUS current
            DESCRIPTION
                "The UDP port to listen on for
RAQMON reports,
                    running on other transport
protocols than SNMP.
                    If the RAQMON PDU
transport protocol is SNMP,
                    The standard port 162 is
always used"
            ::= { raqmonConfig 1 }

        raqmonConfigPDUTransport
OBJECT-TYPE
            SYNTAX BITS
                {
                other(0),
                   rtcp(1),
                snmp(2)
                }
            MAX-ACCESS read-only
            STATUS current
            DESCRIPTION
                "The PDU transport used by this
collector.
                    If other(0) is set, the collector
supports a
                    transport other than RTCP or
SNMP
                    If rtcp(1) is set, the collector
supports RTCP
                    transport
                    If snmp(2) is set, the collector
supports SNMP
                    transport."
            ::= { raqmonConfig 2 }


        raqmonConfigRaqmonPDUs OBJECT-
TYPE
            SYNTAX Counter32
            MAX-ACCESS read-only

            STATUS current
            DESCRIPTION
                "Count of RAQMON PDUs
received by the Collector."
            ::= { raqmonConfig 3 }


        raqmonConformance OBJECT
IDENTIFIER ::= { raqmon 2 }

        raqmonGroups OBJECT IDENTIFIER
::= { raqmonConformance 1 }

        raqmonCollectorGroup OBJECT-
GROUP
            OBJECTS {
                raqmonParticipantStartDate,
                raqmonParticipantIndex,
                raqmonParticipantAddr,
                raqmonParticipantSendPort,
                raqmonParticipantRecvPort,
                raqmonParticipantSetupDelay,
                raqmonParticipantName,
                raqmonParticipantTool,
                raqmonParticipantQosCount,
                raqmonParticipantEndDate,
                raqmonParticipantRcvdPT,
                raqmonParticipantSentPT,
                raqmonParticipantActive,
                raqmonParticipantPeerIndex,
                raqmonParticipantPeerAddr,
                raqmonParticipantSrcLayer2,
                raqmonParticipantDestLayer2,
                raqmonParticipantSrcLayer3,
                raqmonParticipantDestLayer3,
                raqmonParticipantCPUMean,
                raqmonParticipantCPUMin,
                raqmonParticipantCPUMax,
                raqmonParticipantMemoryMean,
                raqmonParticipantMemoryMin,
                raqmonParticipantMemoryMax,
                raqmonParticipantRTTMean,
```

```
                raqmonParticipantRTTMin,
                raqmonParticipantRTTMax,
                raqmonParticipantJitterMean,
                raqmonParticipantJitterMin,
                raqmonParticipantJitterMax,
                raqmonParticipantOWDMean,
                raqmonParticipantOWDMin,
                raqmonParticipantOWDMax,
                raqmonParticipantPackets,
                raqmonParticipantLostPackets,
                raqmonQosTime,
                raqmonQosRTT,
                raqmonQosJitter,
                raqmonQosRcvdPackets,
                raqmonQosRcvdOctets,
                raqmonQosSentPackets,
                raqmonQosSentOctets,
                raqmonQosLostPackets,
                raqmonQosRsvpStatus,
                raqmonParticipantAddrEndDate,
                raqmonConfigPort,
                raqmonSessionExceptionIndex,

raqmonSessionExceptionJitterThreshold,

raqmonSessionExceptionRttThreshold,

raqmonSessionExceptionLostPacketsThreshold,

raqmonSessionExceptionRowStatus,
                    raqmonConfigPDUTransport,
                    raqmonConfigRaqmonPDUs }
                STATUS current
                DESCRIPTION
                    "Objects used in RAQMON by a
Collector"
                ::= { raqmonGroups 1 }

            raqmonCollectorNotifications
NOTIFICATION-GROUP
                NOTIFICATIONS {
raqmonSessionAlarm }
                STATUS current
                DESCRIPTION
                    "Notifications emitted by a
RAQMON collector."
                ::= { raqmonGroups 2 }


        END
```