

UNIVERSIDADE FEDERAL DO RIO DE JANEIRO

ESCOLA DE ENGENHARIA
DEPARTAMENTO DE ENGENHARIA ELETRÔNICA E DE COMPUTAÇÃO

Transmissão de Voz sobre IP com Taxas Variáveis

Autor:

André Marcos Dias Campos

Orientador:

Prof. Sergio Lima Netto, *Ph.D.*

Examinador:

Prof. Marcelo Luiz Drumond Lanza, *M.Sc.*

Examinador:

Prof. José Ferreira de Rezende, *Ph.D.*

DEL
Agosto/2005

Dedicatória

Dedico este trabalho a minha mãe (Maria Regina Futuro Marcos Dias) e a meus avós (João Marcos Dias e Glória Futuro Marcos Dias) que sempre acreditaram em mim e investiram em minha educação e formação.

Agradecimentos

Agradeço:

- à minha mãe (Maria Regina Futuro Marcos Dias), a minha irmã (Priscilla Marcos Dias Campos) e a minha namorada (Kátia Holz da Silva) por terem sempre me apoiado e me dado força para atingir todos meus objetivos.
- ao avô João e a minha avó Glória que ajudaram a me formar, educar e sempre contribuíram com bons exemplos.
- ao meu orientador Sergio Lima Netto pela orientação, pelas conversas, pelas idéias e pela compreensão.
- aos meus amigos do DEL Rafael Silva Pereira, Gustavo Henrique Fraga de Carvalho, Marcello de Lima Azambuja e Eduardo Brasil por toda a ajuda que me deram no projeto.
- aos amigos da Cisco Emanuel Nigri, Luis Filipe Silva, Julio Gouy e Carlos Mont'Alverne que muito me ensinaram sobre Telefonia IP.
- ao LPS por ter dado infra-estrutura para desenvolver o projeto.
- a todos meus amigos, que sempre me apoiaram e incentivaram.

Resumo:

Neste trabalho será realizada uma apresentação geral de VoIP. Serão mostradas as arquiteturas atuais de VoIP, os protocolos de sinalização mais utilizados atualmente, as atuais arquiteturas de qualidade de serviço (QoS) e os *codecs* de voz mais utilizados.

Além da abordagem teórica de VoIP, será implementado um sistema de transmissão de voz sobre IP com taxas variáveis. Esse consiste em um cliente e um servidor, que trocam pacotes de voz, codificada com um codificador CELP, através de uma rede IP. No sistema proposto, há a possibilidade de se modificar a taxa de compressão de voz no decorrer de uma chamada. Para isto, foi desenvolvido um protocolo entre o cliente e o servidor, que possibilite troca de *codec* durante uma chamada. O objetivo da mudança de *codec* é melhorar a qualidade da chamada, que muda em função das condições da rede. Devido a isto, a decisão de trocar de *codec* será tomada em função dos parâmetros da rede (atraso, taxa de perda de pacotes e *jitter*).

O sistema foi desenvolvido para o sistema operacional *Linux*, utilizando a linguagem de programação C++. Desenvolveu-se também uma interface gráfica para o sistema. Para esta utilizou-se a biblioteca *WX-Windows*, também em C++.

Palavras-chave: VoIP, QoS, TCP/IP, *Codec* e redes.

Índice

SIGLAS E ABREVIATURAS	vi
CAPÍTULO 1 – INTRODUÇÃO	
1.1 Introdução	1
1.2 Organização do Projeto	3
CAPÍTULO 2 – PROTOCOLOS ATUAIS DE VOIP	
2.1 Introdução	5
2.2 Arquitetura VoIP	5
2.3 H.323	8
2.4 SIP	12
2.5 Comparação entre H.323 e SIP	15
2.6 Conclusão	17
CAPÍTULO 3 – QOS	
3.1 Introdução	18
3.2 QoS	18
3.3 DiffServ	21
3.4 IntServ	26
3.4.1 RSVP	26
3.5 Conclusão	28
CAPÍTULO 4 – CODIFICADORES ATUAIS	
4.1 Introdução	30
4.2 Codificadores Atuais	30
4.3 Codificador Utilizado	33
4.4 Análise do Codificador Utilizado	36
4.5 Conclusão	40
CAPÍTULO 5 – APLICAÇÃO DESENVOLVIDA	
5.1 Introdução	41
5.2 Estrutura do Funcionamento da Aplicação	41
5.3 Protocolo para Troca de Modo de Operação	44
5.3.1 Fase de <i>Setup</i>	45
5.3.2 Fase de Conversa	47
5.3.3 Descrição dos Pacotes	49
5.4 Inteligência da Aplicação	53
5.5 Analisador da Rede	54
5.6 Interface Gráfica	56
5.7 Conclusão	60
CAPÍTULO 6 – CONCLUSÃO	
6.1 Conclusão	61
6.2 Trabalhos Futuros	62
REFERÊNCIAS BIBLIOGRÁFICAS	64

Siglas e Abreviaturas:

ADPCM	<i>Adaptive Differential Pulse Code Modulation</i>
AF	<i>Assured Forwarding</i>
ASCII	<i>American Standard Code for Information Interchange</i>
ASN.1	<i>Abstract Syntax Notation number One</i>
ATM	<i>Asynchronous Transfer Mode</i>
BE	<i>Best Effort</i>
CELP	<i>Code Excited Linear Prediction</i>
Diffserv	<i>Differentiated Services</i>
DNS	<i>Domain Name System</i>
DS	<i>Differentiated Services</i>
DSCP	<i>Differentiated Services Code Point</i>
DSP	<i>Digital Signal Processor</i>
EF	<i>Expedited Forwarding</i>
GW	<i>Gateway</i>
HTTP	<i>Hypertext Transfer Protocol</i>
ICMP	<i>Internet Control Message Protocol</i>
IETF	<i>Internet Engineering Task Force</i>
Intserv	<i>Integrated Services</i>
IP	<i>Internet Protocol</i>
ISDN	<i>Integrated Services Digital Network</i>
ITU	<i>International Telecommunication Union</i>
kbps	<i>kilo bits per second</i>
LAN	<i>Local Area Network</i>
LPC	<i>Linear Predictive Coding</i>
LSF	<i>Line Spectral Frequency</i>
Mbps	<i>Mega bits per second</i>
MC	<i>Multipoint Controller</i>
MCU	<i>Multipoint Control Unit</i>
MGCP	<i>Media Gateway Controller Protocol</i>
MIME	<i>Multipurpose Internet Mail Extentions</i>
MOS	<i>Mean Opinion Score</i>
MP	<i>Multipoint Processor</i>
MPLS	<i>Multi Protocol Label Switching</i>
MP-MLQ	<i>Multipulse Multilevel Quantization</i>
Ms	<i>Miliseconds</i>
OSI	<i>Open Systems Interconnection</i>
P2P	<i>Peer-to-Peer</i>
PATH	<i>Path Message</i>
PBX	<i>Private Branch eXchange</i>
PC	<i>Personal Computer</i>

PCM	<i>Pulse Code Modulation</i>
PDA	<i>Personal Digital Assistant</i>
PESQ	<i>Perceptual Evaluation of Speech Quality</i>
PGP	<i>Pretty Good Privacy</i>
PHB	<i>Per-Hop Behavior</i>
POTS	<i>Plain Old Telephony Service</i>
PSTN	<i>Public Switched Telephone Network</i>
QoS	<i>Quality of Service</i>
RAM	<i>Random Access Memory</i>
RESV	<i>Reservation Message</i>
RFC	<i>Request For Comments</i>
RSVP	<i>Resource ReSerVation Protocol</i>
RTCP	<i>Real Time Control Protocol</i>
RTP	<i>Real Time Protocol</i>
SCCP	<i>Signaling Connection Control Point</i>
SCN	<i>Switched Circuit Network</i>
SIP	<i>Session Initiation Protocol</i>
SSL	<i>Secure Socket Layer</i>
TCP	<i>Transmission Control Protocol</i>
ToS	<i>Type of Service</i>
UA	<i>User Agent</i>
UAC	<i>User Agent Client</i>
UAS	<i>User Agent Server</i>
UDP	<i>User Datagram Protocol</i>
UFRJ	<i>Universidade Federal do Rio de Janeiro</i>
URL	<i>Uniform Resource Locator</i>
VAD	<i>Voice Activity Detector</i>
VoIP	<i>Voice over Internet Protocol</i>
WAN	<i>Wide Area Network</i>

Capítulo 1 - Introdução

1.1 Introdução

A evolução das redes de dados baseadas no protocolo IP, e ainda a motivação da diminuição dos custos, desperta o interesse da utilização da rede dados para outros serviços como por exemplo para a transmissão de voz sobre IP (VoIP – *Voice over IP*), motivando ainda mais o crescimento da integração de dados e voz. A idéia básica é aproveitar a mesma estrutura dos meios de comunicação de dados para transportar também a voz. Ao invés de se ter uma rede para dados e outra para voz, pretende-se ter uma rede convergente, na qual dados e voz sejam transmitidos.

Este projeto tem objetivos tanto teóricos como práticos. O tema fundamental do projeto é VoIP, com ênfase em qualidade de serviço (*Quality of Service – QoS*).

Pode-se dizer que o projeto é dividido em duas partes: a parte teórica e a parte prática. O projeto começa com a parte teórica, que tem como objetivo estudar as atuais arquiteturas de VoIP, bem como cada elemento que compõe cada arquitetura. Também são foco de nosso estudo os protocolos de comunicação VoIP atualmente utilizados, H.323 e SIP. Assim, apresentaremos as principais características, vantagens e desvantagens destes protocolos, posteriormente será realizada uma comparação entre ambos. Outro elemento importante de um sistema VoIP é a Qualidade de Serviço (*Quality of Service – QoS*). As atuais arquiteturas utilizadas de QoS são *Diffserv* (*Differentiated Services*) e *Intserv* (*Integrated Services*). Estudaremos as duas arquiteturas bem como o protocolo RSVP (*Resource ReSerVation Protocol*), que é um protocolo que compõe a arquitetura *Intserv*. O último elemento para terminarmos o estudo teórico, são os principais codificadores utilizados atualmente nas implementações de VoIP, como G.711, G.723, G.729, entre outros.

Como os serviços de telefonia públicos são considerados de alta disponibilidade, é necessário que a VoIP mantenha esta mesma qualidade tradicional oferecida pelas redes públicas de telefonia e pelos PBXs (*Private Branch eXchange*). A satisfação do usuário é um fator preponderante de sucesso para a migração da tecnologia. A voz é uma aplicação

que precisa ser tratada de forma adequada pela rede de comunicação para que haja a qualidade de serviço necessária. Devido a isto, entendemos que a QoS é fundamental para a utilização e popularização de VoIP, portanto este projeto tem como principal foco a QoS em VoIP.

No que diz respeito à implementação, desenvolveu-se uma aplicação do tipo cliente-servidor com uma interface gráfica para o usuário, que transmite voz codificada através de uma rede IP. Para esta aplicação desenvolveu-se um protocolo de sinalização entre o cliente e o servidor de modo que o servidor fique monitorando os parâmetros da rede IP que podem influenciar a qualidade da comunicação VoIP: atraso, variação do atraso (*jitter*) e taxa de perda de pacotes. Ao perceber, durante uma chamada, que algum desses parâmetros não está de acordo com o desejado, o servidor toma alguma medida para que a qualidade se restabeleça.

O codificador/decodificador utilizado na aplicação será um CELP desenvolvido na UFRJ [1][2][3]. Ao estudarmos este codificador, constatamos que alterando alguns de seus parâmetros, como por exemplo, o tamanho dos dicionários fixo e adaptativo, podemos modificar características como a complexidade computacional para codificar/decodificar a voz, a banda requerida para transmitir a voz codificada e a qualidade da voz. Explorando este fato, dependendo das condições da rede, o servidor troca algumas características da codificação, ou seja, troca o modo de operação do codificador. Desta forma, insere-se um módulo simples de QoS na aplicação, sem ter que mudar nada na rede, diferente das duas soluções existentes (*Diffserv* e *Intserv*). Porém, é importante ressaltar que, esta proposta de QoS não exclui a utilização tanto de *Diffserv* como de *Intserv*.

Para que tudo isto seja possível, é necessário implementar um protocolo de sinalização para os pontos finais (cliente e servidor) trocarem mensagens de controle, criar um módulo que meça o atraso, o *jitter* e a taxa de perda de pacotes, um módulo que decida quando o codificador deve ser trocado e uma interface gráfica amigável para o usuário, de modo que não somente usuários familiarizados com computador possam utilizar a aplicação.

1.2 Organização do Projeto

No Capítulo 2 deste projeto, abordaremos as arquiteturas e os protocolos atuais de VoIP. No que diz respeito às arquiteturas, mostraremos seus principais componentes e função de cada um em um sistema VoIP. Já com relação aos protocolos, os mais utilizados são H.323 e SIP. Apresentaremos ambos, evidenciando as suas principais características. A partir disto apontaremos as suas principais vantagens e desvantagens, procurando fazer uma comparação direta entre H.323 e SIP.

No Capítulo 3, abordaremos QoS em VoIP. Inicialmente mostraremos a necessidade de QoS em um sistema VoIP. Abordaremos as duas principais arquiteturas atuais de QoS: *Diffserv* e *Intserv*. Mostraremos as suas principais características, e, por fim, faremos uma comparação entre as duas arquiteturas.

O Capítulo 4 é o último capítulo da parte teórica e já expõe aspectos da implementação realizada neste projeto. Nesse capítulo, trataremos dos codificadores de voz. Primeiramente mostraremos a necessidade de usarmos codificadores de voz para transmitir voz por redes IP. Também serão mostrados os codificadores mais utilizados atualmente, como por exemplo, G.711, G.723 e G.729, suas principais características, vantagens e desvantagens, especialmente com relação à qualidade da voz e sua taxa de compressão. Com isto, termina-se a apresentação dos principais componentes de um sistema VoIP e inicia-se a parte prática do projeto. Começaremos, com a apresentação do codificador desenvolvido na UFRJ, como mencionado anteriormente. Com o devido entendimento do funcionamento do codificador CELP, mostraremos os modos de operação do codificador estudados e seus desempenhos no que diz respeito à complexidade computacional, qualidade da voz e taxa de compressão.

No Capítulo 5, veremos todo o detalhamento da aplicação desenvolvida. Inicialmente apresentaremos a arquitetura da aplicação e qual a função de cada parte do sistema. Com isso, apresentaremos o protocolo de comunicação entre o cliente e o servidor, que permite a troca do codificador durante uma chamada, e o critério de escolha adotado pelo servidor, em função dos parâmetros da rede. Nesse capítulo, também

mostramos como foi implementado o módulo que mede os parâmetros de rede (atraso, *jitter* e taxa de perda de pacotes). Por fim, apresentaremos a interface gráfica desenvolvida com a biblioteca Wx-Windows e explicaremos como o usuário deve proceder para utilizar a aplicação através desta interface.

Finalmente no Capítulo 6, concluiremos o projeto. Apontaremos os principais pontos do trabalho, explicitando suas contribuições. Apresentaremos também sugestões de trabalhos futuros, que dêem continuidade a este trabalho.

Capítulo 2 – Protocolos Atuais de VoIP

2.1 Introdução

A evolução das redes de dados baseadas no protocolo IP desperta o interesse para a transmissão de voz sobre IP (VoIP – *Voice over IP*), motivando ainda mais o crescimento da integração de dados e voz. A idéia básica é aproveitar a mesma estrutura dos meios de comunicação de dados para transportar também a voz.

O objetivo deste capítulo é apresentarmos um panorama atual das tecnologias de VoIP, abordando os principais pontos da arquitetura e dos protocolos de comunicação utilizados atualmente. Em capítulos posteriores, abordaremos outros aspectos importantes de comunicação utilizando VoIP como os codificadores e qualidade de serviço (QoS).

Na seção 2.2 apresentaremos a arquitetura VoIP, mostrando seus componentes e funcionamento de modo geral. Na seção 2.3 apresentaremos o protocolo H.323, mostrando seus componentes, funcionamento básico, apontando vantagens e desvantagens. Na seção 2.4 apresentaremos o protocolo SIP, mostrando seus componentes, funcionamento básico, apontando vantagens e desvantagens. Na seção 2.5 faremos uma comparação entre os protocolos H.323 e SIP, explicitando as vantagens e desvantagens de cada um com relação ao outro. Na seção 2.6 concluiremos o capítulo ressaltando os pontos mais importantes abordados.

2.2 Arquitetura VoIP

A definição básica de um sistema VoIP é a conversão das amostras de voz em uma série de pacotes para a transmissão destes através de uma rede IP. A arquitetura de um serviço VoIP é formada por três componentes básicos:

- **Gateway de Voz** - Dispositivo responsável pela digitalização, compressão, demodulação e funções de empacotamento IP sobre o sinal de voz recebido.

- **Agente de Chamada** - Notificado via sinalização enviada pelo *gateway* de que uma chamada está sendo iniciada e analisa como a chamada será gerenciada.
- **Equipamento de usuário** – Dispositivo final usado pelo usuário que solicita o uso de serviço de voz. Podendo ser este analógico ou digital.

A figura 2.1, de [4], apresenta a arquitetura VoIP com os componentes descritos acima. Os telefones que estão ligados aos PBXs são os equipamentos do usuário, os pontos finais. Os *gateways* digitalizam a voz vinda do PBX, empacotam e enviam para o seu destino, ou seja, um outro *gateway*.

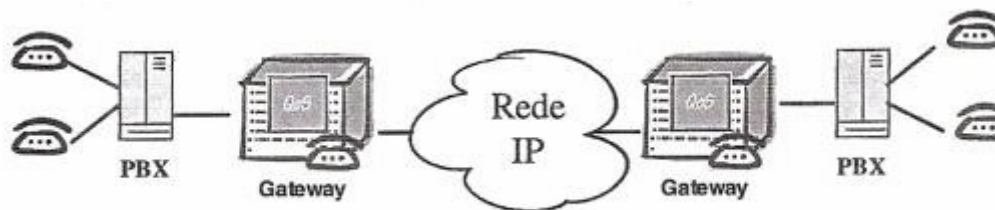


Figura 2.1 – Mostra pontos finais ligados a PBXs que estão interligados através de uma rede IP. [4]

As soluções VoIP seguem um modelo de camadas hierárquico podendo ser comparado ao modelo de referência OSI.

O efeito de cada camada no processo de comunicação VoIP é a adição de um cabeçalho para controle e transmissão da informação. Como por ser visto na figura 2.2.

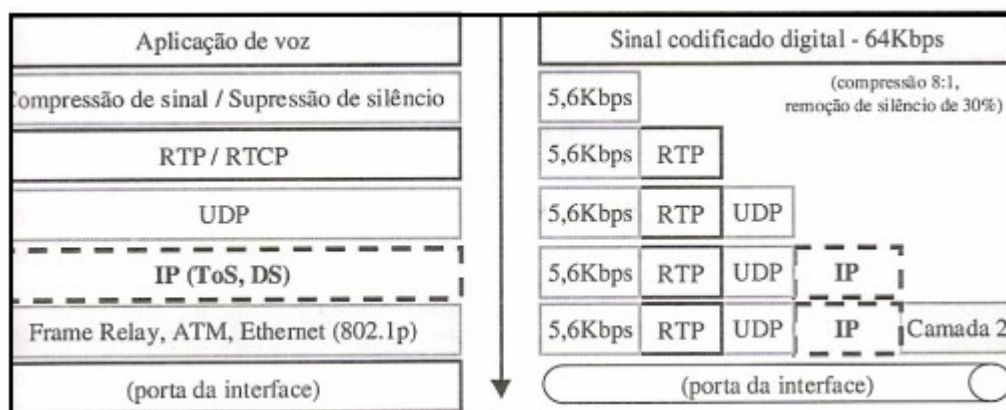


Figura 2.2 – Adição dos cabeçalhos RTP, UDP e IP a amostra de voz codificada. [4]

Inicialmente, a voz é comprimida depois de naturalmente ser digitalizada, deve ser empacotada utilizando o protocolo *Real Time Protocol* (RTP). O RTP transporta os pacotes de voz, incluindo pacotes multimídia interativos, mas sem nenhuma garantia ou qualidade de serviço. Em uma rede IP não se pode garantir qual caminho um pacote percorrerá para chegar ao seu destino. Com isto, os pacotes podem chegar ao destino fora de ordem. Uma das principais funções do protocolo RTP é ordenar os pacotes antes de a voz ser decodificada. Juntamente com RTP, utiliza-se *Real Time Control Protocol* (RTCP). Este é utilizado como protocolo de controle durante a chamada. Este protocolo tem como objetivo medir e informar aos pontos finais as condições da rede. É importante ressaltar, que mesmo detectando condições ruins da rede, o RTCP não toma nenhuma medida.

Devido à necessidade de uma entrega rápida e a falta de tempo para uma retransmissão no caso da perda de um único pacote, utiliza-se UDP para transmitir voz através de RTP. Para que o pacote possa ser transmitido pela rede, é necessário o uso do protocolo IP, que é fundamental para os roteadores conseguirem achar o destino do pacote. No final de todo este processo de adição de cabeçalhos, o pacote de voz tem um *overhead* de 40 bytes (20 bytes IP, 8 bytes UDP e 12 bytes RTP). Os cabeçalhos são enviados seqüencialmente antes das amostras digitalizadas e codificadas de voz como mostra a figura 2.2.

Ao contrário da telefonia convencional, a sinalização IP requer uma tecnologia muita mais complexa devido à natureza distinta dos pontos finais. Cada ponto final pode variar em vários aspectos, como por exemplo, requerimento de banda, áudio, capacidade de dados, etc. Logo antes de qualquer conexão é necessário assegurar que os dois pontos possuam as mesmas capacidades, é necessário implementar uma arquitetura de controle de chamadas ou de sinalização que promova o controle do estado dos recursos da rede, habilitando-os para a correta conexão entre as partes de uma comunicação.

2.3 H.323

O padrão H.323 provê fundamentos para transmissão de áudio, vídeo e comunicação de dados sobre redes baseadas em pacotes, incluindo a Internet. Os usuários podem se comunicar sem a preocupação de compatibilidade, ou seja, produtos multimídia e aplicações de diferentes fornecedores podem interoperar.

Todos os dispositivos que utilizam este padrão devem dar suporte ao tráfego de voz e opcionalmente a vídeo e a dados. Estes dispositivos podem ser integrados a um PC, através de uma solução de *software*, ou implementados em equipamentos do tipo *stand-alone*, como por exemplo, um telefone IP. As conexões podem ser de dois tipos: ponto a ponto ou ponto a multiponto.

O protocolo H.323 é na verdade um protocolo formado por vários outros protocolos. A figura 2.3 mostra a pilha de protocolos implementada pelos pontos finais (terminais e gateways) do H.323 em redes VoIP.

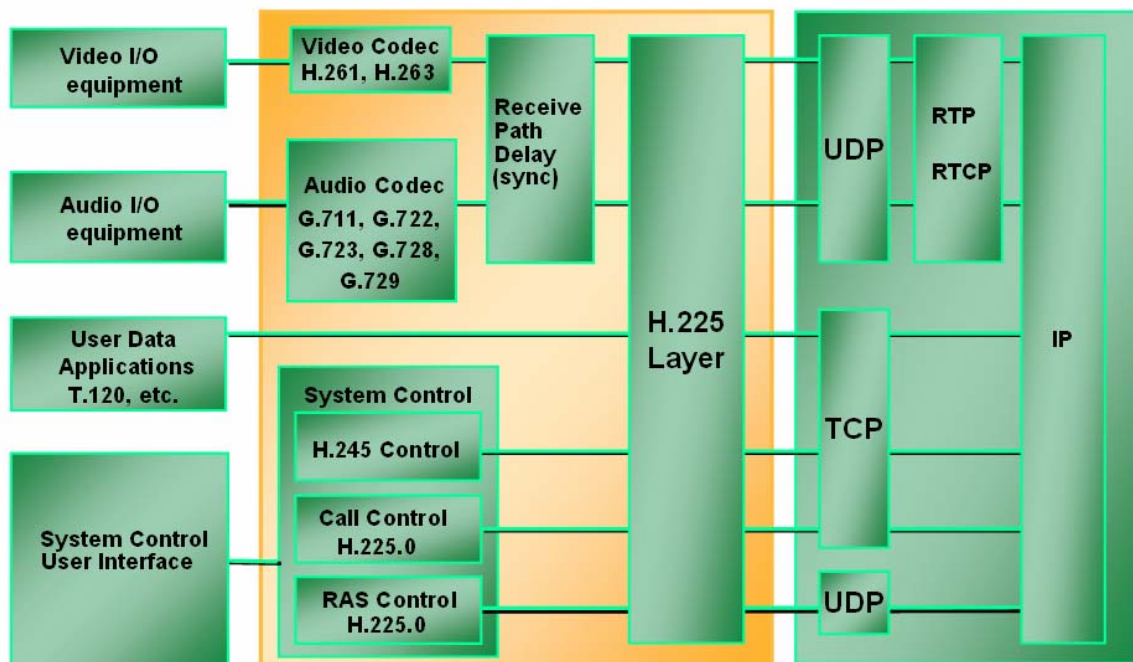


Figura 2.3 – Protocolos formadores do H.323. Os principais são H.245 e H.225[4].

O padrão H.323 foi formalizado em 1996 pela ITU (*International Telecommunication Union*), abaixo descreveremos os demais padrões que o compõe:

- H.225 – especifica mensagens de controle da chamada, incluindo sinalização, registros, admissões, empacotamento sincronização de *streams*.
- H.245 – negocia o *codec* a ser utilizado na chamada, especifica as mensagens para abertura e fechamento de canais de transmissão de *streams* e outros comandos de requerimento e indicação.
- H.261 – codificador de vídeo para serviços audiovisuais a 64 kbps.
- H.263 – especifica um novo codificador de vídeo para vídeos POTS (*Plain Old Telephony Service*).

A arquitetura do padrão H.323 é composta por vários componentes, que serão descritos abaixo:

- Terminal – é um ponto final em uma rede local que prove comunicação em tempo real com um outro terminal H.323, *gateway* ou MCU. Esta comunicação consiste em sinais de controle, indicações, áudio, vídeo e dados entre os dois terminais. Um terminal pode prover somente voz, voz e dados, voz e vídeo, ou ainda voz, dados e vídeo.
- *Gateway* (GW) – é um ponto final em uma rede local que prove comunicação em tempo real entre terminais H.323 em uma LAN (*Local Area Network*) e outros terminais ITU em uma WAN (*Wide-Area Network*), ou outro gateway H.323. Em outros terminais ITU incluem-se dispositivos que suportem H.310 (H.320 em B-ISDN), H.320 (ISDN), H.321 (ATM), H.322 (GQOS-LAN), H.324 (GSTN), H.324M (móvel), e V.70 (DSVD).

- *Gatekeeper* – provê controle de serviço de chamadas para os pontos H.323. Mais um *gatekeeper* pode estar presente e eles podem se comunicar entre si. Um *gatekeeper* é logicamente separado dos pontos finais, mas fisicamente, sua implementação pode coexistir com um terminal, MCU, *gateway*, MC, ou outros dispositivos não H.323.
- Unidade de Controle Multiponto (*Multipoint Control Unit* - MCU) – é um ponto final em uma rede local que permite que três ou mais terminais e *gateways* participem de uma conferência multiponto. Permite também dois terminais se conectarem em uma conferência ponto-a-ponto, que posteriormente pode se tornar uma conferência multiponto.
- Controlador Multiponto (*Multipoint controller* - MC) – é uma entidade H.323 em uma rede local que prove o controle para que três ou mais terminais participem de uma conferência multiponto. Permite também conectar dois terminais em uma conferência ponto-a-ponto que posteriormente pode se tornar uma conferência multiponto.
- Processador Multiponto (*Multipoint Processor* - MP) – é uma entidade H.323 em uma rede local que provê processamento centralizado de *streams* de áudio, vídeo e dados em uma conferência multiponto. O MP prove mistura, chaveamento, ou outros processamentos das *streams* acima sob o controle de um MC.
- Conferência ponto-a-ponto (*Point-to-point Conference*) – uma conferência ponto-a-ponto é uma conferência direta entre dois terminais ou entre um terminal H.323 e um terminal SCN através de um *gateway*.
- *Switched-circuit Network* (SCN) – é uma rede de telecomunicação chaveada pública ou privada como GSTN, N-ISDN ou B-ISDN.

A figura 2.4 mostra uma possível arquitetura com quase todos os componentes do padrão H.323 e com quase todas as modalidades de chamada. Dentre estas podemos destacar: dois terminais fazendo uma chamada ponto-a-ponto dentro da LAN. Dois terminais realizando uma chamada através da Intranet ou da Internet. Dois terminais realizando uma chamada com a ajuda de um *Gatekeeper*. Dois ou mais terminais realizando uma conferência com a ajuda de uma MCU e por último uma chamada de um terminal para a PSTN através de um *gateway*.

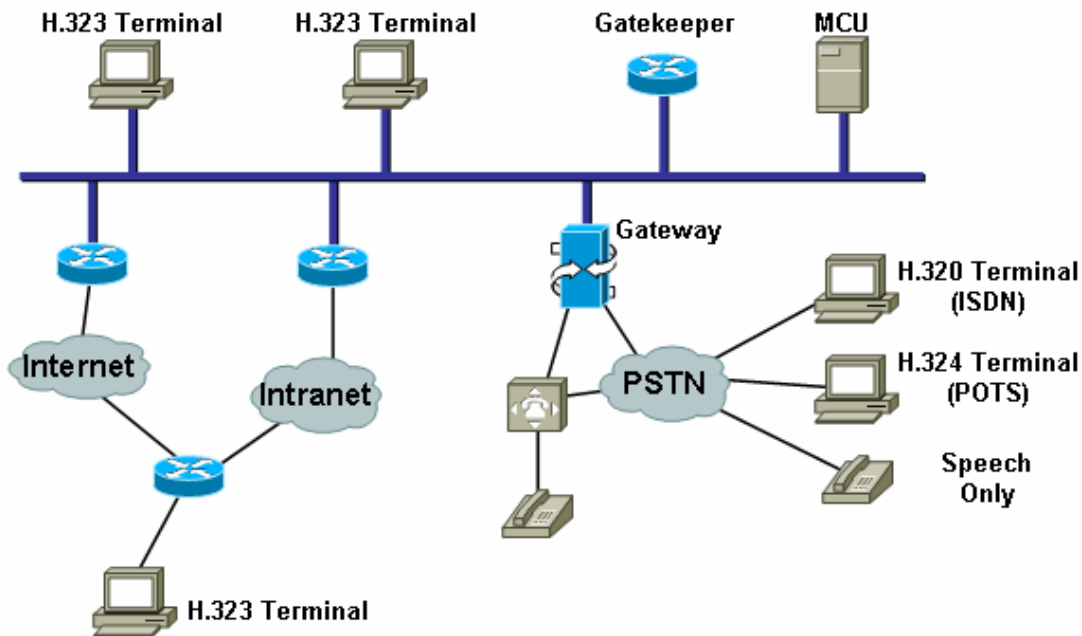


Figura 2.4 - uma possível arquitetura com quase todos os componentes (terminais, gateway, MCU e gatekeeper) do padrão H.323. [4]

O protocolo H.323 tem pontos finais razoavelmente inteligentes, pois são capazes de manter suas próprias chamadas. O protocolo H.323 é um sistema *peer-to-peer* de sinalização. Cada ponto final pode chamar outro diretamente utilizando os procedimentos dados pelo padrão, evidentemente se cada ponto final souber o endereço IP do outro. As mensagens de sinalização do *setup* inicial seguem as do modelo tradicional ISDN Q.931, utilizando o formato ASN.1 encapsulado em pacotes TCP. Após a fase de *setup*, os dois pontos são capazes de negociar qual codificador de áudio vão utilizar, e finalmente escolher quais portas usarão para a transmissão de pacotes RTP, a transmissão de voz em si. É bom notar, que pelo fato das portas RTP serem escolhidas dinamicamente pelos pontos finais em uma grande faixa de possibilidades, há uma relativa dificuldade de operar com a presença de *firewalls*.

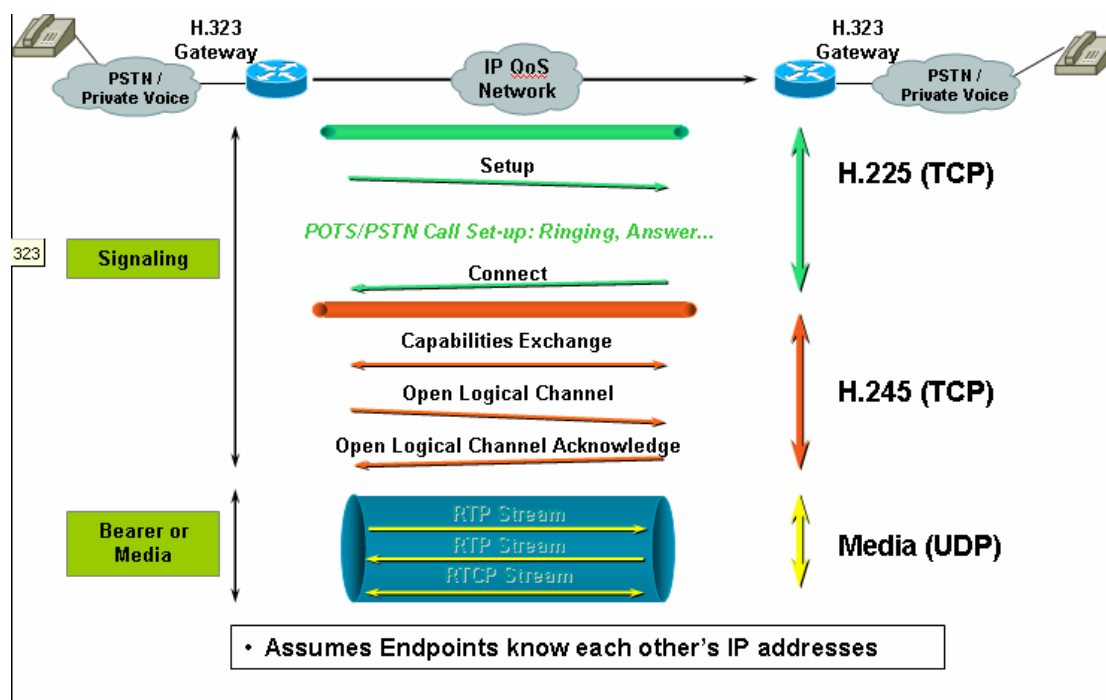


Figura 2.5 – Estabelecimento de uma chamada H.323.

A figura 2.5 mostra o estabelecimento de uma chamada H.323 sem *gatekeeper*. Para realizar o *call-setup*, utiliza-se o protocolo H.225, que utiliza pacotes TCP. Após isto, há a troca capacidade, ou seja, quais codificadores cada ponto final tem disponível, isto se dá através do protocolo H.245. Com isto a chamada H.323 está estabelecida, então se inicia a troca de pacotes de voz através do protocolo RTP.

2.4 SIP

Session Initiation Protocol (SIP) é um novo protocolo de sinalização com uma arquitetura *peer-to-peer* muito parecida com a do H.323. Diferentemente do H.323, SIP é por filosofia e intenção um protocolo direcionado para Internet. Esse está descrito no RFC 2543, que foi desenvolvido pelo *IETF MMUSIC Working Group* em 1999.

O SIP depende de pontos finais relativamente inteligentes, o que requer pouca ou nenhuma interação com servidores. Cada ponto final gerencia sua própria sinalização e ainda há um controle de sessão, o que dá inúmeras vantagens ao SIP.

O protocolo possui uma estrutura simples de mensagens, o que permite fazer o *setup* de uma chamada em menos passos que o H.323, o que acarreta uma melhor performance com relação ao H.323, com o mesmo *hardware*.

SIP também é mais escalável que o H.323 por ter um modelo de chamada distribuído, mas a verdadeira vantagem do SIP é o fato de ser um protocolo concebido para ser usado na Internet. SIP usa mensagens ASCII baseado em HTTP/1.1, o que significa que as mensagens são facilmente decodificadas e depuradas. Devido a isto, outras aplicações *web* podem suportar serviços SIP sem grandes modificações. O SIP suporta nomes de URL (com DNS) e MIME (*Multipurpose Internet Mail Extensions*), o que significa que o modelo permite que o e-mail de um usuário seja, por exemplo, seu número de telefone, fazendo com que diferentes pontos finais possam se comunicar.

Embora o SIP seja filosoficamente um protocolo *peer-to-peer*, é regido pela lógica cliente-servidor. Um típico cliente SIP seria um Telefone IP, um PC ou um PDA (*Personal Digital Assistant*). Para haver comunicação, é necessário haver um UAC (*user agent client*) para originar a requisição SIP e um UAS (*User Agent Server*) para receber a requisição. Também são suportados SIP *Proxy Servers*, SIP *Redirect Servers*, *Registrar* e *Location Servers*. Estes servidores são opcionais, mas normalmente, úteis e valiosos em atuais implementações de SIP. Abaixo apresentaremos esses servidores:

- *Proxy Server* - atua como servidor e cliente, inicializa requisições SIP de acordo com o interesse do UAC.
- Servidor de Redirecionamento – (*Redirect Server – RS*) – recebe requisição SIP, mapeia o destino para um ou mais endereços e responde com aqueles endereços.
- Registrador (*Registrar*) – aceita requisições de registro da atual localização dos UAC. Tipicamente colocado com um *Redirect Server*.

- Servidor de Localização (*Location Server*) – provê informação sobre as possíveis localidades de chamadas. Normalmente, é contatado por um *Redirect Server*. Um *Location Server/Service* pode coexistir com um *SIP Redirect Server*.

Como já foi mencionado, o protocolo SIP é baseado um simples vocabulário de mensagens de requisição e resposta. As mensagens serão descritas abaixo:

- REGISTER – registra localização atual no servidor.
- INVITE – é enviada pelo chamador para iniciar uma chamada.
- ACK – é enviada pelo chamador para confirmar a aceitação da chamada pelo chamado. Esta mensagem não é respondida.
- BYE – é enviado por qualquer lado para terminar a chamada.
- CANCEL – é enviada para terminar uma chamada ainda não conectada.
- OPTIONS – é enviada para requerer capacidades.

Como mencionado anteriormente, o endereçamento SIP é modelado como de URLs. Como por exemplo:

sip: "einstein" aeinstein@smartguy.com; transport=udp

A estrutura do endereço indica os parâmetros como o tipo de transporte e endereços.

Na figura 2.6, é demonstrado o quão mais simples é o *setup* de uma chamada SIP com relação a H.323, mesmo com um servidor *Proxy* envolvido. Sem o *Proxy*, os pontos finais precisam saber o IP um do outro. De qualquer forma, o *setup* de uma chamada, se

inicia com uma simples mensagem INVITE enviada diretamente de um ponto final a outro. Quando há o envolvimento de um servidor de Proxy, em conjunto com os outros servidores descritos anteriormente, são basicamente responsáveis por resolver um endereço SIP em IP, de modo que o UAC consiga enviar a mensagem de INVITE para o UAS.

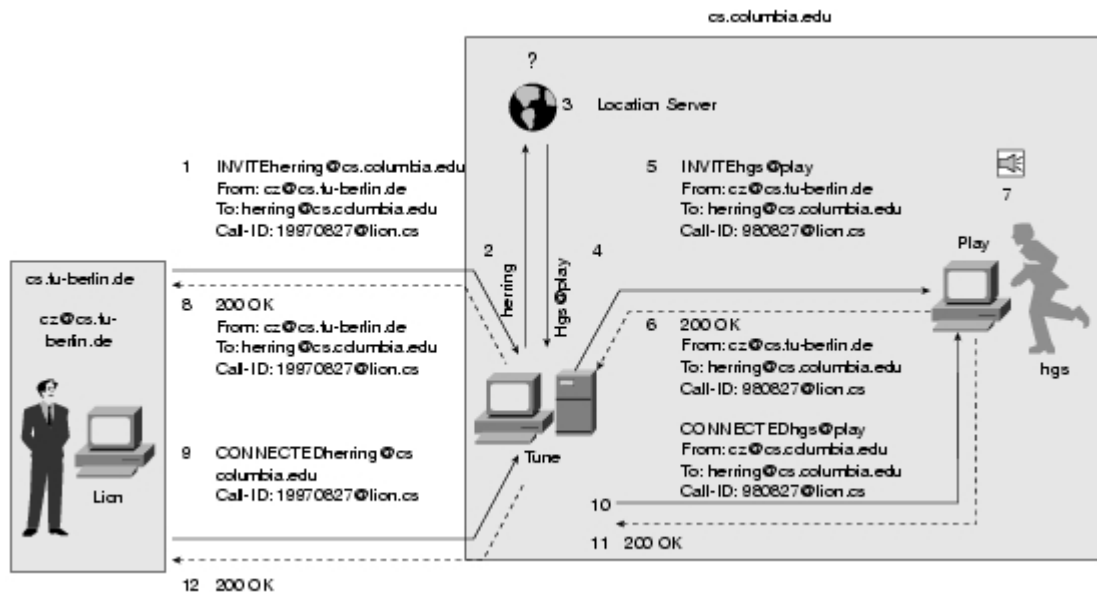


Figura 2.6 – Estabelecimento de uma chamada SIP. [4]

2.5 Comparação entre H.323 e SIP

Ambos os protocolos são usados para configuração e controle de sinalização de chamada, logo os mesmos podem ser usados como protocolos de sinalização em uma rede IP.

Apesar de o SIP ser mais escalável, flexível e possuir uma maior facilidade de implementação. O H.323 vem sendo utilizado por fabricantes como o preferido para soluções VoIP devido a sua maior segurança.

Ainda existe muita discussão sobre qual protocolo será mais usado pela comunidade VoIP. Defensores do ITU-T dizem que o H.323 ganhou suporte de múltiplos fabricantes, em contra partida os defensores do SIP questionam a sua interoperabilidade.

A tabela 2.1 a seguir, sem nenhuma pretensão de apontar qual a melhor implementação, mostra uma comparação entre os dois protocolos.

Tabela 2.1 – Comparação entre H.323 e SIP.

	H.323	SIP
Filosofia	Desenvolvido para gerenciar chamadas de voz, multimídia e serviços suplementares através de recomendações específicas para cada tipo de serviço.	Desenvolvido para estabelecer uma sessão entre dois pontos finais, sem nenhum relacionamento específico com algum tipo de mídia.
Complexidade	Mais Complexo	Relativamente simples
Formato das Mensagens	Representação binária	Representação textual ASCII
Transporte de Mídia	RTP/RTCP	RTP/RTCP
Protocolo de Transporte	UDP	UDP
Endereçamento	Entende URLs e E.164	Somente formato URL
Resolução de endereços	O <i>Gatekeeper</i> pode usar muitos protocolos para descobrir o endereço de destino do usuário chamado, inclusive conversando com outros <i>Gatekeepers</i> .	Um UAC precisa enviar um invite para um servidor <i>Proxy</i> , sendo este o responsável em fazer a resolução de endereço, podendo usar muitos protocolos para descobrir o endereço destino.
Requerimentos para chamada	O <i>Gatekeeper</i> é opcional, uma chamada pode ser estabelecida diretamente entre dois pontos finais.	O servidor <i>Proxy</i> pode não ser usado para o estabelecimento da chamada. Uma chamada pode ser estabelecida diretamente entre dois UA.
Escalabilidade	Não é muito escalável	Altamente escalável

Autenticação	Através de H.235	Via HTTP,SSL,PGP e outros
Encriptação	Via H.235	Via SSL,PGP e outros

2.6 Conclusão

Neste capítulo, apresentamos um panorama atual das tecnologias de VoIP, abordando os principais pontos da arquitetura e dos protocolos mais utilizados atualmente, H.323 e SIP. É interessante observar, que há outros protocolos como MGCP e MEGACO, mas que não serão abordados neste texto.

Atualmente H.323 apresenta mais funcionalidades do que o SIP, mas especialmente pela simplicidade e pela compatibilidade do SIP com a *Web*, há uma tendência do SIP passar a ser mais utilizado que H.323. De qualquer forma, a tecnologia VoIP ainda é muito recente para se apontar um padrão de sinalização definitivo. Ambos ainda têm a evoluir.

No próximo capítulo, será abordada a questão da Qualidade de Serviço (QoS) em sistemas VoIP. Será apresentado à necessidade de se ter QoS em redes que utilizem VoIP e quais são as atuais arquiteturas de QoS.

Capítulo 3 - QoS

3.1 Introdução

Como os serviços de telefonia públicos são considerados de alta disponibilidade, é necessário que a telefonia sobre pacotes mantenha esta mesma qualidade tradicional oferecida pelas redes públicas de telefonia e pelos PBXs. A satisfação do usuário é fator preponderante de sucesso para a migração da tecnologia. Para o entendimento do processo de migração de ambientes distintos de dados e voz para um ambiente integrado, é fundamental a compreensão, de que voz é uma aplicação e como deverá ser tratada de forma adequada pela rede de comunicação, para que haja a qualidade de serviço (QoS – *Quality of Service*) necessária.

Neste capítulo adordaremos as tecnologias e arquiteturas atuais que implementam QoS em redes IP. Na seção 3.2, apresentaremos o conceito de qualidade de serviço, mostrando sua necessidade e importância em um sistema VoIP. Na seção 3.3, apresentaremos a arquitetura de Serviços Diferenciados (*Diffserv*), seu funcionamento, suas vantagens e desvantagens. Na seção 3.4, apresentaremos a arquitetura de Serviços Integrados (*Intserv*), bem como o protocolo RSVP, que implementa a arquitetura, suas vantagens e desvantagens.

3.2 QoS

A convergência de dados, voz e vídeo em um só meio é uma realidade com os sistemas VoIP. Em consequência disto, as redes, e a própria Internet, precisam mudar para acomodar as demandas das novas aplicações, especialmente as multimídia, como voz e vídeo, que tem uma maior exigência de QoS. Quando a voz, dividida e empacotada, compartilha o meio físico com um volume imprevisível de tráfego de dados, a QoS é requisitada para proteger o tráfego de voz. A especificação adequada de QoS que implica no atendimento de um serviço, é um requisito de operação de rede, de seus componentes e dos equipamentos para viabilizar a operação com qualidade para uma aplicação baseada em parâmetros definidos. A quantidade de banda mínima é o parâmetro básico e certamente mais presente na especificação de QoS. Este parâmetro é normalmente

considerado durante a fase de projeto e implantação da rede. A ampliação da largura de banda é necessária, mas não suficiente, pois é necessário que esta seja administrada.

O protocolo IP não tem nenhum compromisso com QoS. É considerado um protocolo que provê um serviço de melhor esforço no qual os recursos da rede são divididos de forma igualitária, não há diferenciação entre os pacotes. A adição de QoS na rede representa uma mudança significativa pois possibilita o tratamento diferenciado de serviços. Entretanto, modifica o princípio fundamental da simplicidade que fez o sucesso da Internet. Entretanto, com o crescimento da quantidade de novos serviços sobre IP (*everything over IP*), estão sendo geradas novas oportunidades e incentivando o suporte a QoS, que é essencial em aplicações de comunicação em tempo real envolvendo voz, áudio e vídeo.

Atualmente, a QoS é definida como a capacidade da rede prover serviço de encaminhamento de dados de forma consistente e previsível. Outra possibilidade é definir QoS como a habilidade de um elemento da rede (seja uma aplicação, *host*, roteador, ou outro dispositivo) ter algum nível de garantia que seu tráfego e exigências de serviço possam ser satisfeitos.

Basicamente existem dois mecanismos básicos para prover QoS em redes IP:

- **Reserva de Recursos:** os recursos da rede são divididos de acordo com os requisitos de QoS da aplicação, e sujeitos à política de administração de largura de banda. O RSVP (*Resource ReSerVation Protocol*), por exemplo, fornece os mecanismos para implementação de Serviços Integrados (*IntServ*) baseado na reserva de recursos. Quando um recurso estiver alocado para uma aplicação, mesmo que esta não o esteja utilizando, ninguém pode utilizá-lo.
- **Priorização:** o tráfego da rede é classificado e os recursos de rede são divididos, entre as classes de tráfego, de acordo com critérios de políticas de administração de largura de banda. Para habilitar QoS os mecanismos de classificação dão tratamento preferencial a aplicações identificadas como tendo requisitos mais

exigentes. A arquitetura de Serviços Diferenciados (*DiffServ*) provê este tipo de serviço.

É importante ressaltar que estas duas arquiteturas de QoS não são mutuamente exclusivas, na verdade são complementares.

Há um conjunto de parâmetros considerados essenciais para redes com QoS. Estes são o atraso fim-a-fim, a variação do atraso (*jitter*), a taxa de perda de pacotes e largura de banda. Cada um destes parâmetros será descrito abaixo:

- Atraso fim-a-fim - é o tempo entre o envio de uma mensagem por um ponto-final e a recepção desta mensagem pelo ponto-final destino. Este atraso ocorre no caminho de transmissão ou em um dispositivo no caminho de transmissão. Para transmissão de voz, a ITU através da norma G.114, recomenda não haver atraso maior que 150 ms.
- Variação do atraso (*jitter*) - é uma distorção ocorrida nos tempos de chegada entre pacotes comparados aos tempos originais de transmissão entre pacotes. É uma distorção que acontece, por exemplo, quando fluxos de voz ou vídeo são transmitidos em uma rede e os pacotes não chegam ao seu destino dentro da ordem sucessiva ou em uma determinada cadência. Ou seja, eles variam em termos de tempo de atraso. Esta distorção é particularmente prejudicial ao tráfego multimídia, fazendo com que o sinal de áudio ou vídeo tenha uma qualidade distorcida ou fragmentada na recepção. Para transmissão de voz recomenda-se, jitter médio de no máximo 30 ms.
- Perda de pacotes - representa o número de pacotes que foram transmitidos na rede, mas não alcançaram seu destino em um determinado período de tempo. Para transmissão de voz recomenda-se, que a perda de pacotes não deve ser maior que 1%.

- Largura de banda - é uma medida de capacidade de transmissão de dados, normalmente expressa em *kilobits* por segundo (kbps) ou *megabits* por segundo (Mbps). Durante a fase de projeto de uma rede, é necessário definir a quantidade de banda que deverá ser utilizada por cada aplicação da rede, de modo que cada aplicação consiga realizar sua função, sem causar prejuízos às demais aplicações.

Nas próximas duas seções, abordaremos os dois modelos (*Diffserv* e *Intserv*) de QoS fim-a-fim que é a habilidade da rede em prover o serviço requerido por um tráfego específico de uma extremidade a outra da rede, mas antes disto é importante apresentarmos o modelo de serviço de melhor esforço (*Best Effort*). Este serviço também é conhecido como sem QoS e, provê simplesmente a conectividade básica sem nenhuma garantia. Atualmente, a Internet é um bom exemplo de serviço de melhor esforço. Embora o serviço de melhor esforço não ofereça QoS, este é adequado para uma grande gama de aplicações de rede, tais como, correio eletrônico e transferência de arquivos, pois estas se adaptam facilmente às condições da rede.

3.3 Diffserv

A qualidade de serviço na solução *DiffServ* se dá através da divisão do tráfego de rede em diferentes classes, chamadas classes de serviço (*Class of Service*), e pela aplicação de parâmetros de QoS nestas classes. A divisão dos pacotes em classes se dá através de um número que fica armazenado no byte de tipo de serviço (*Type of Service – ToS*) no cabeçalho IP. Na verdade, este *byte* não é inteiramente utilizado, usam-se somente os 6 bits mais significativos. Este conjunto de 6 bits é chamado de *Differentiated Services Code Point* (DSCP). Neste texto, abordaremos somente classificação feita com base no valor do DSCP do cabeçalho IP, mas é importante ressaltar que se pode classificar tráfego com base em informações contidas em cabeçalhos 802.1p, MPLS, ATM e *FrameRelay* [4].

Uma vez classificados, os pacotes recebem tratamento específicos para serem encaminhados, isto é formalmente chamado de PHB (*Per-Hop Behavior*), que são aplicados em cada elemento da rede (roteadores ou *switches*), provendo tratamento

apropriado ao pacote no que diz respeito a atraso, *jitter* e banda. Esta combinação de marcação de pacotes e PHBs bem definidos resulta numa solução de QoS bem escalável para qualquer pacote dado, de qualquer aplicação. Com isto, em *Diffserv*, não há sinalização para QoS, o que resulta numa solução altamente escalável.

A descrição da solução *Diffserv* foi terminada pelo IETF no final de 1998. Os objetivos do IETF eram de criar métodos simples de prover diferentes classes de serviço para o tráfego da Internet, para suportar vários tipos de aplicações. Como já mencionado, a marcação de pacotes é feita através dos 6 bits do ToS. Este campo é agora chamado de DS (*Differentiated Services*), que tem dois bits (os menos significativos) que não são utilizados (RFC-2474). Os 6 bits utilizados são chamados de *Differentiated Services Codepoint* (DSCP). Com estes 6 bits é possível classificar ou agregar 64 tipos diferentes de classes. Toda a classificação no modelo *Diffserv* é realizada principalmente através do DSCP.

Cada nó em uma arquitetura *DiffServ* define dois importantes componentes: um de classificação e um de condicionamento de tráfego. O classificador é o componente que divide o fluxo de entrada em um conjunto de fluxos de saída por meio de filtros de tráfego baseados no conteúdo do DSCP. Com a chegada do pacote, o nó se o pacote está de acordo com um perfil de tráfego pré-definido. Dependendo da conformidade três ações podem ser executadas: marcação, formatação (*shaping*) e descarte.

A marcação de um pacote é realizada no campo DSCP, no qual se atribui um valor, este é mapeado para um PHB definido na arquitetura *DiffServ*. Os PHBs definem o comportamento de encaminhamento de um pacote em um nó *DiffServ*.

O DSCP é usado para selecionar o PHB que o pacote terá em cada nó. Este campo é tratado como um índice em uma tabela usada para selecionar o mecanismo de manipulação de pacotes implementado em cada dispositivo. Cada PHB pode especificar que, para determinados pacotes, serão dadas certas prioridades relativas a outros, em termos de banda utilizada ou de preferência para descarte.

PHBs são comportamentos individuais aplicados em cada nó, por isso isoladamente não garantem QoS fim-a-fim. Entretanto, a interligação de roteadores com os mesmos PHBs e a limitação da taxa em que os pacotes são enviados para um PHB, possibilita o uso de PHBs para construir QoS de fim-a-fim. Por exemplo, a concatenação de PHBs ao longo de uma rota pré-estabelecida, com um controle de admissão, pode prover um serviço similar ao de uma linha dedicada, que é satisfatório para voz.

Há três PHB padronizados: PHB EF (*Expedited Forwarding*), AF (*Assured Forwarding*) e PHB *default* que serve para o tráfego BE (*Best Effort*), assegurando compatibilidade com o encaminhamento melhor esforço. No PHB *default*, o valor de DSCP é igual à zero, como mostra a figura 3.1.

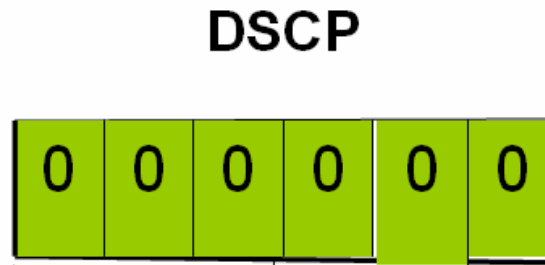


Figura 3.1: O valor do DSCP é igual à zero para PHB *Default*.

O PHB EF pode ser usado para tráfego com requisitos de baixa perda de pacotes, baixo atraso, baixo *jitter* (variação de atraso) e garantia de largura de banda. Estes requisitos são alcançados assegurando-se que os agregados de tráfego encontram nenhum ou pouco enfileiramento. Este PHB é indicado para pacotes de voz. O valor do DSCP para o PHB EF é 46 em decimal, como pode ser visto na figura 3.2.

DSCP

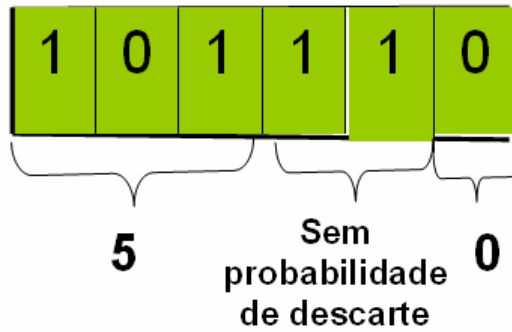


Figura 3.2: O valor do DSCP é igual a 46 para PHB EF.

O PHB AF tem por objetivo fornecer entrega de pacotes IP, com largura de banda assegurada, em quatro classes de transmissão (AF1, AF2, AF3 e AF4), mas não oferece garantias quanto ao atraso. Cada classe tem três precedências de descartes (*Drop Precedence*), que são utilizadas para determinar a importância do pacote. Assim, um nó congestionado dá preferência para serem descartados, entre os pacotes de uma mesma classe, aqueles com maiores valores de precedência de descarte. Este PHB é indicado a pacotes que façam parte de aplicações importantes, no caso de um ambiente empresarial, de aplicações relevantes para o negócio da empresa. O formato do campo DSCP para o PHB AF é “aaadd0”, no qual os três bits mais significativos indicam a classe (AF1 001, AF2 010, AF3 011 e AF4 100) como pode ser visto na tabela 3.1. Já os outros dois bits “dd” indicam a precedência de descarte (Baixa Precedência 01, Média Precedência 10 e Alta Precedência 11), como pode ser visto na tabela 3.2. É interessante ressaltar, que o bit menos significativo do DSCP é sempre igual a zero.

Tabela 3.1: Valores de DSCP para cada classe AF.

Classe	Valor		
AF1	001	dd	0
AF2	010	dd	0
AF3	011	dd	0
AF4	100	dd	0

Tabela 3.2: Valores do DSCP para cada precedência de descarte.

Probabilidade de Descarte	Valor	AF
Baixa	01	AF11
Média	10	AF12
Alta	11	AF13

Após apresentar os PHBs padrões, utilizaremos a tabela 3.3 para exemplificar quais aplicações normalmente compõe cada classe de serviço.

Tabela 3.3: Aplicações que compõe cada classe.

PHB	DSCP	Valor DSCP	Aplicações
EF	EF	101110 / 46	Voz
AF1	AF11	001010 / 10	FTP, Backup
	AF12	001100 / 12	
	AF13	001110 / 14	
AF2	AF21	010010 / 18	Banco de dados, tráfego interativo.
	AF22	010100 / 20	
	AF23	010110 / 22	
AF3	AF31	011010 / 26	Aplicações Críticas para empresa
	AF32	011100 / 28	
	AF33	011110 / 30	
AF4	AF41	100010 / 34	Vídeo, sinalização de voz.
	AF42	100100 / 36	
	AF43	100110 / 38	
Default	Default	000000 / 0	e-mail e Internet

Após a classificação, dependendo da conformidade do pacote com relação ao seu PHB, o pacote, em um caso extremo de fila cheia, pode ser descartado. Normalmente, em

interfaces com *links* pequenos (menores que 2 Mbps), não se deve descartar um pacote, mas armazená-lo em um *buffer*, até que seja possível transmiti-lo (*shapping*).

3.4 IntServ

A solução de Serviços Integrados (*Intserv*) para QoS foi proposta pela IEFT (*Internet Engineering Task Force*). Na solução, cada aplicação que necessite de QoS para seus pacotes deve informar as condições que julgam necessárias ou imprescindíveis para que transmissão tenha qualidade, ou seja, a solução é baseada em reserva de recursos da rede.

Em cada roteador no caminho entre os pontos finais, são necessários mecanismos de controle de admissão e escalonamento de pacotes, de forma que se mantenha uma dinâmica dos recursos mais equilibrada, para que todas as aplicações tenham iguais oportunidades de realizar suas comunicações. O controle de admissão implementa um algoritmo que garante ou nega recursos, mantendo a carga da rede em níveis aceitáveis para todos os usuários. O escalonamento de pacotes utiliza-se de algoritmos que decidem em que ordem devem ser enviados os pacotes que estejam aguardando transmissão, de acordo com a prioridade definida pela aplicação.

Faz-se necessário também a existência de um protocolo de gerência de recursos, de forma a requisitar e reservar em um determinado caminho a QoS requerida pelas aplicações, tanto em ambientes ponto-a-ponto ou *multicast*. Os recursos mais comuns de serem controlados são a banda e o tamanho dos *buffers* dos roteadores.

O RSVP (*ReSerVation Protocol*) é um protocolo que implementa a reserva de recursos da arquitetura *Intserv*. Esse será descrito na próxima subseção.

3.4.1 RSVP

RSVP é um protocolo desenvolvido para permitir que as aplicações requisitem diferentes QoS para seus fluxos de dados. Na implementação de RSVP, os roteadores devem adequar-se aos mecanismos de controle de QoS para garantir a entrega dos

pacotes de dados e cada aplicação deve estar capacitada a fornecer os parâmetros ideais de QoS. Outro fato importante é que RSVP não é um protocolo de roteamento, mas trabalha em conjunto com este. É usado por uma aplicação para requisitar um nível de QoS específico da rede. O protocolo atua tanto em máquinas do usuário quanto em roteadores, responsabilizando-se, nesse caso, a estabelecer e manter as condições para o serviço requisitado. RSVP é um protocolo de sinalização para o controle de reserva de recursos.

Uma série de mensagens devem ser trocadas entre as aplicações e os elementos de rede para corretamente requisitar a qualidade de serviço para uma determinada sessão. Após a definição da sessão, serão trocadas mensagens de controle RSVP. As mensagens RSVP fundamentais são *reservation message* (RESV) e *path message* (PATH).

A mensagem PATH, enviada pelo transmissor, é propagada pelo caminho ponto-a-ponto ou *multicast*, seguindo a rota informada pelos mecanismos de encaminhamento até os receptores. Um elemento no caminho dos dados, ao receber um PATH, criará um estado chamado *PATH state*. As mensagens deste tipo armazenam o estado de cada nó pelos quais a mensagem de PATH transitou. A mensagem RESV, enviada pelo receptor, contém um descritor de fluxo, definindo a QoS aceita pelo receptor em função daquela pedida pela mensagem PATH.

Mediante a troca dessas mensagens, o protocolo toma uma série de decisões, como por exemplo, aceitar ou não um novo fluxo, criando um ambiente para que os recursos sejam reservados. Cada parâmetro utilizado para requisitar a QoS está representado nas mensagens do RSVP.

A seguir, apresentaremos um exemplo em um cenário simples, ponto-a-ponto, em que todos os elementos compreendam o protocolo RSVP. Em cada nó roteador, além do gerenciador de recursos, responsável pela reserva, existem módulos de controle de tráfego e policiamento que auxiliam o RSVP na tarefa de reservar os recursos requisitados.

Exemplo:

Seja T1 uma máquina que contém uma aplicação que tenha requisitos de QoS, R1 outra máquina que receberá dados dessa aplicação e G roteadores intermediários. A aplicação em T1, para iniciar sua transmissão, envia uma mensagem de controle chamada de mensagem de caminho (PATH), que seguirá seu caminho, pelos diversos roteadores até chegar em R1, conforme figura 3.3. A mensagem PATH contém um cabeçalho RSVP e todas as informações sobre o tráfego que a aplicação em T1 espera gerar e valores para os parâmetros de QoS. A cada roteador G existente entre T1 e R1, será criado um estado chamado de *PATH state*.

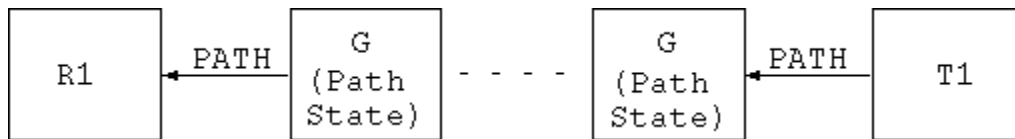


Figura 3.3: Mensagem PATH Seguindo de T1 a R1 [6].

Ao chegar em R1, este analisa as informações contidas em PATH e seleciona os parâmetros de reserva desejados, montando a mensagem de reserva (RESV). Por onde passar, a RESV provocará nos roteadores um estado chamado *SOFT state*, até chegar de volta em T1, informando-o sobre as condições dos parâmetros de QoS da reserva realizada por R1 e propriedades do caminho entre ambos, como é mostrado na figura 3.4.

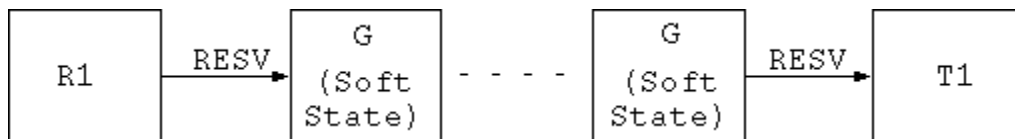


Figura 3.4: Reserva entre R1 e T1, Criando o *Soft State* nos Elementos Intermediários [6].

3.5 Conclusão

Neste capítulo, foi abordada a importância de QoS em redes IP e especialmente para aplicações que transmitam voz e vídeo. Também foram mostradas as duas arquiteturas que implementam atualmente QoS, *Diffserv* e *Intserv*.

Na solução *Diffserv*, QoS é garantida através da divisão do tráfego em classes e após isto uma priorização de cada classe. Devido a esta característica, é uma solução que não requisita sinalização, o que acarreta nenhum *overhead* na rede, garantindo maior escalabilidade, entretanto não há garantia de 100%.

A solução *Intserv* garante QoS através da reserva de recursos, fazendo com que seja necessário um protocolo de sinalização entre os pontos-finais e roteadores (RSVP). Este fato acarreta um *overhead* na rede, o que reduz a escalabilidade da solução. Com *Intserv* há 100% de garantia, pois os recursos que forem reservados para certa aplicação, somente poderão ser utilizados pela mesma.

É importante ressaltarmos que as soluções *Diffserv* e *Intserv* não são excludentes, mas podem e devem atuar conjuntamente para prover a rede melhor QoS.

No próximo capítulo abordaremos outro quesito fundamental para o sucesso de VoIP, os codificadores. Serão apresentados alguns dos codificadores usados atualmente e o codificador utilizado neste trabalho.

Capítulo 4 – Codificadores Atuais

4.1 Introdução

Nos últimos anos, o uso de sistemas de telefonia móvel e VoIP cresceram devido aos avanços nas técnicas de compressão/codificação de voz. Sem estas, a quantidade de banda necessária para que a voz fosse transmitida seria enorme. Isto poderia provocar uma diminuição do número de usuários simultâneos, aumentando os custos por chamada e podendo, assim, inviabilizar a popularização da telefonia móvel e de VoIP.

Neste capítulo, será abordado o último tópico relacionado aos principais componentes de um sistema VoIP; este tópico inclui os codificadores/decodificadores (*codecs*) de voz. Na seção 4.2, serão apresentados os codificadores atuais, explicitando suas vantagens e desvantagens, sempre dando enfoque ao compromisso existente entre qualidade da voz, taxa de compressão e complexidade computacional. Na seção 4.3, será apresentado o *codec* utilizado na implementação deste trabalho, dando ênfase às características mais relevantes para o projeto. Por último, na seção 4.4, será feita uma análise da banda utilizada, da complexidade computacional e da qualidade da voz, de cada modo de operação do *codec* utilizado durante a implementação.

4.2 Codificadores Atuais

A tecnologia de codificação/decodificação (*codec*) de voz avançou muito nos últimos anos graças aos avanços nas arquiteturas dos processadores de sinal digital (*Digital Signal Processor* – DSP) e às pesquisas do funcionamento da fala humana. Nesta seção, serão apresentados os *codecs* de voz mais utilizados atualmente.

Pulse code modulation (PCM) tem duas variações básicas de PCM de 64 kbps: *Mu-law* (utilizado na América do Norte e no Japão) e *A-law* (utilizado no resto do mundo). Ambos os métodos utilizam a compressão logarítmica para atingir a qualidade equivalente a 12 de quantização linear a 8 bits.

Um outro método de codificação também utilizado é o *Adaptive Differential Pulse Code Modulation* (ADPCM), que realiza a codificação utilizando amostras de 4 bits, o que gera uma taxa de transmissão de 32 kbps. Diferentemente de PCM, os 4 bits não codificam diretamente a amplitude do sinal da voz, mas sim as diferenças em amplitude, ou seja, a taxa de mudança da amplitude do sinal da voz, empregando uma predição linear da amostra atual.

PCM e ADPCM são exemplos de codificadores de forma de onda (*waveform codecs*). Essas técnicas de codificação exploram características redundantes na forma de onda do sinal de voz. Novas técnicas de compressão têm sido desenvolvidas nas últimas décadas, com o objetivo de explorar as características de geração da fonte de voz. Elas também utilizam técnicas de processamento de sinais para comprimir a voz, enviando somente informações paramétricas sobre a excitação original de voz e o formato do trato vocal, o que requer menos banda ao transmitir. Os codificadores que utilizam essas técnicas são chamados, de forma geral, de codificadores paramétricos, pois modelam a fonte emissora de voz e transmitem seus parâmetros, não a voz propriamente dita. No conjunto dos codificadores paramétricos, incluem-se variações como *Linear Predictive Coding* (LPC), *Code Excited Linear Prediction* (CELP) e *Multipulse Multilevel Quantization* (MP-MLQ).

Codificadores preditivos avançados utilizam modelos matemáticos do trato vocal humano e, ao invés de enviarem a voz comprimida, enviam representações matemáticas da voz, fazendo com que a voz possa ser gerada novamente no receptor. Inicialmente, os *codecs* só reproduziam bem as vozes de seus desenvolvedores, já que os testes eram realizados com as suas próprias vozes, fazendo com os parâmetros de codificação fossem ajustados para elas. Após testes, descobriu-se que não reproduziam bem vozes femininas e nem de alguns dialetos asiáticos. Para corrigir esses defeitos, alguns *codecs* foram reprojatados, tornando-se mais genéricos em relação à frequência da voz, possibilitando o funcionamento também com outros tipos de vozes humanas [4].

A ITU padronizou os mais populares *codecs* de voz para telefonia e para VoIP, vejamos alguns deles abaixo:

- G.711 descreve a codificação de voz PCM a 64 kbps. A voz codificada com G.711, que já é o formato digital de voz para entrega na rede pública de telefones ou mesmo em PBXs.
- G.726 descreve a codificação ADPCM a 40, 32, 24 e 16 kbps. A voz em ADPCM pode ser transmitida em redes de pacotes, telefonia pública e PBXs.
- G.728 descreve a codificação variação de baixo atraso do CELP a 16 kbps.
- G.729 descreve codificação CELP, permitindo que a voz seja codificada na taxa de 8 kbps. Existem quatro variações deste padrão, G.729, G.729A, G.729B e G.729AB. A diferença entre G.729 e G.729A está na complexidade computacional, G.729A é menos complexo computacionalmente. G.729B é G.729 com supressão de silêncio (VAD-Voice Activity Detector) e finalmente, G.729AB é G.729A com supressão de silêncio. Todos os codecs da família G.729 apresentam uma qualidade tão boa quanto ao ADPCM de 32 kbps.
- G.723.1 descreve a técnica de compressão que consegue comprimir sinais de voz e áudio a uma taxa muito baixa. Faz parte da família de padrões H.324, que possui duas possíveis taxas: 5.3 e 6.3 kbps. A taxa mais alta é baseada na tecnologia MP-MLQ e tem uma ótima qualidade. A taxa mais baixa é baseada na tecnologia CELP, que tem uma boa qualidade.

A maneira utilizada para avaliar a qualidade dos *codecs* é um teste chamado MOS (*Mean Opinion Score*). Como a qualidade da voz e do som é, de maneira geral, muito subjetiva e varia muito em função do ouvinte, é muito importante para o teste de qualidade que uma grande quantidade de ouvintes e de amostras sejam utilizados. No teste MOS, é dada uma amostra de voz para um grupo de ouvintes que atribuem uma nota de 1 (ruim) a 5 (excelente). A nota final do *codec* é uma média de todas as notas. O teste MOS também é utilizado para qualificar cada *codec* em circunstancias variadas, nas quais se incluem ruídos de fundo, múltiplas codificações e decodificações. E, por último, usa-se o teste MOS para comparar a performance entre os *codecs*. A tabela 4.1 mostra as notas obtidas no MOS de alguns *codecs* e as relaciona com a taxa de compressão.

Tabela 4.1 – banda utilizada em kbps, complexidade computacional medida em MIPS por DSPs da família 54xx da Texas Instruments, tamanho de janela e nota no MOS dos principais *codecs*. [4]

Codec	Taxa (kbps)	Complexidade Computacional (MIPS)	Tamanho de Janela	MOS
G.711 PCM	64	0.34	0.125	4.1
G.726 ADPCM	32	14	0.125	3.85
G.728 LD-CELP	16	33	0.625	3.61
G.729 CS-ACELP	8	20	10	3.92
G.729a CS-ACELP	8	10.5	10	3.7
G.723.1 MPMLQ	6.3	16	30	3.9
G.723.1 ACELP	5.3	16	30	3.65

Pela análise da tabela, percebe-se que, quanto menor é a taxa, menor é a qualidade. Também nota-se que há uma correlação entre a taxa e a complexidade computacional; quanto menor a taxa, maior a complexidade computacional. Esta tabela é muito importante, pois demonstra a relação de compromisso no momento da escolha de um *codec* a ser utilizado.

4.3 Codificador Utilizado

Nesta seção apresentaremos o codificador, da família CELP, que será utilizado na implementação deste projeto. O sistema de codificação CELP (*Code Excited Linear Prediction*) é um sistema extremamente complexo. Devido a este fato, somente as características relevantes do sistema CELP que possuem relação com este trabalho serão abordadas nesta seção. Uma profunda análise e apresentação, tanto conceitual, quanto matemática do sistema, podem ser encontradas em [1]. Neste texto, serão abordados somente com os conceitos e características que influenciam na implementação. A base do *codec* que será utilizado foi desenvolvida por [1] e [3].

Como já mencionado na seção anterior, CELP é um sistema de codificação/decodificação híbrido, ou seja, utiliza tanto características dos *codecs* de forma de onda (por exemplo PCM), quanto dos *codecs* paramétricos (por exemplo LPC). Os *codecs* de forma onda têm uma qualidade superior, mas possuem uma taxa de compressão inferior, acarretando em maior uso de banda. Os *codecs* paramétricos conseguem atingir maiores taxas de compressão, entretanto, a qualidade é inferior. Os *codecs* híbridos têm como objetivo uma relação de compromisso entre a qualidade e a banda utilizada, de modo a utilizar características de cada um dos dois tipos de *codec*. O CELP utiliza como base o sistema LPC, que é responsável pela parte de codificação do modelo do trato vocal, e utiliza dicionários fixos e adaptativos para realizar a manipulação das excitações da voz, gerando uma qualidade maior que um *codec* LPC “puro”.

A voz é um sinal altamente não estacionário, então, é preciso utilizar janelas no tempo para que a voz possa ser tratada como um sinal estacionário por partes. O CELP utilizado neste trabalho, utiliza janelas de 20 ms, que, amostradas a 8 kHz (frequência de amostragem usada para voz telefônica), correspondem a 160 amostras por janela. Com isso, em cada pacote de voz enviado pela rede, serão enviados dados referentes a 20 ms de voz ou 160 amostras. Para detalhes sobre o janelamento, ver [1].

O trato vocal é modelado através de um filtro digital de predição linear, que é um filtro no qual uma amostra $x[n]$ pode ser prevista a partir de amostras anteriores. Os coeficientes do filtro são calculados para cada janela, criando-se um modelo preditivo completo.

Depois disso, realiza-se uma série de filtragens preditivas com amostras de excitações (dicionários ou *codebook*), conhecidas tanto pelo transmissor quanto pelo receptor, até que se descubra qual amostra de excitação apresenta melhor resultado ao passar pelo filtro preditivo com um ganho, que também precisa ser calculado, em comparação com a voz original.

Após todo este processo, o transmissor terá o valor dos coeficientes que geram o filtro preditivo que modela o trato vocal durante uma janela de 20 ms ou 160 amostras e índices para os dicionários que indicam quais excitações e ganhos devem ser enviados para o receptor. Este, ao receber esses parâmetros, irá realizar a síntese da voz, passando as amostras apontadas pelos índices dos dicionários, multiplicadas pelos respectivos ganhos, pelo filtro digital, que é implementado com os coeficientes enviados pelo transmissor.

Neste trabalho, serão utilizados 10 coeficientes de predição linear. Este número foi escolhido com base em estudos de [1] e [3]. Devido a conclusões de [3], os 10 coeficientes serão quantizados de modo que todos ocupem juntos 32 bits, sendo que os dois primeiros coeficientes ocupam 4 bits cada um e os oito coeficientes restantes ocupam 3 bits cada, totalizando 32 bits.

Como já foi mencionado anteriormente, cada modelo de predição linear, no que diz respeito aos coeficientes, será utilizado para cada 160 amostras. Entretanto, o cálculo da melhor excitação e do seu ganho será realizado para cada janela de 40 amostras. Como neste projeto são utilizados dois dicionários (um fixo e um adaptativo), ver [1], o transmissor irá enviar ao receptor 10 coeficientes quantizados, 8 índices de dicionário (4 para o fixo e 4 para o adaptativo) e 8 valores de ganho quantizados (um ganho para cada índice de dicionário, cada um ocupando 6 bits) para cada 20 ms de voz ou 160 amostras. Detalhes sobre a quantização podem ser encontrados em [1].

Informações a respeito dos índices dos dicionários serão abordadas na próxima seção, que irá abordar também os modos de operação que serão estudados para a implementação deste projeto. Na tabela 4.2 será mostrado um resumo da composição de um pacote de voz codificada.

Tabela 4.2 – Composição de um pacote de 20 ms de voz codificada.

Ganhos Dicionário Fixo	4*6	24
Ganhos Dicionário Adaptativos	4*6	24
Índices Dicionário Fixo	4*11	44
Índices Dicionário Adaptativo	4*11	44
Coefficientes LSF	2*4 + 8*3	32
Total		168 bits
		21 bytes

4.4 Análise do Codificador Utilizado

Nesta seção, serão abordados os modos de operação do *codec* (descritos na seção 4.3) para uma transmissão VoIP. No que diz respeito à implementação, este trabalho tem como objetivo implementar uma comunicação VoIP de modo que a preocupação com QoS não seja só uma atribuição da rede, como na solução *Diffserv*, descrita no capítulo 3. O objetivo principal é desenvolver uma técnica que consiga melhorar a qualidade da chamada em curso através da mudança de parâmetros da codificação da voz.

Como foi visto no capítulo 3, a banda, o atraso, o *jitter* e a taxa de perda de pacotes são parâmetros fundamentais para o sucesso de uma comunicação que utilize VoIP. Por isso, será elaborada uma técnica que verifique as condições da rede enquanto a voz estiver sendo transmitida e, em função dela, seja capaz de modificar parâmetros do *codec*, de modo que se adeque às condições atuais.

Para a escolha dos modos de operação do *codec*, serão analisadas as seguintes características de cada modo: qualidade da voz taxa utilizada e complexidade computacional. A diferença entre os modos de operação se dará somente nos tamanhos dos dicionários fixo e adaptativo, descritos na seção anterior. Com a variação destes dois parâmetros, avaliaremos as variações na qualidade da voz, na taxa e na complexidade computacional.

Será feito, inicialmente, uma análise com 5 modos distintos, sendo que, em cada modo, os dicionários fixo e adaptativo possuem o mesmo tamanho. No modo número 1 (*codec* 1), cada dicionário tem um tamanho de 128; no modo número 2 (*codec* 2), cada dicionário tem um tamanho de 256; no modo número 3 (*codec* 3), cada dicionário tem um tamanho de 512; no modo número 4 (*codec* 4), cada dicionário tem um tamanho de 1024 e, finalmente, no modo número 5 (*codec* 5), cada dicionário tem um tamanho de 2048.

Como foi mostrado na seção anterior, o transmissor envia os seguintes parâmetros para o receptor: ganhos do dicionário fixo, ganhos do dicionário adaptativo, índices do dicionário fixo, índices do dicionário adaptativo e coeficientes do filtro de predição linear. A variação do tamanho dos dicionários só irá alterar o tamanho dos índices dos dicionários. Pode-se observar que o número de bits utilizado para cada índice de dicionário é igual ao logaritmo na base 2 do tamanho dos dicionários.

Como explicado na seção 4.3, temos 4 (6 bits cada) ganhos para o dicionário fixo, 4 (6 bits cada) ganhos para o dicionário adaptativo, 10 (2 com 4 bits cada e 8 com 3 bits cada) coeficientes do filtro de predição linear, 4 (logaritmo na base 2 do tamanho do dicionário) índices para o dicionário fixo e 4 (logaritmo na base 2 do tamanho do dicionário) índices para o dicionário adaptativo. É importante ressaltar que a janela utilizada é de 20 ms, ou seja, cada pacote de voz é gerado com 20 ms de voz. Para se calcular a taxa requisitada por cada modo do *codec*, basta somar o número de bits utilizados para representar a janela e dividir por 20 ms. Com isso, será obtida a banda utilizada, em kbps. Os dados de cada *codec*, encontrados pela maneira descrita, encontram-se na tabela 4.3.

Tabela 4.3 – A taxa utilizada por cada modo de operação em kbps e kBps, mostrando que com o aumento do tamanho do dicionário, aumenta-se a taxa utilizada.

Codec	Taxa (kbps)	Taxa (kBps)
Codec 1	6.8	0.85
Codec 2	7.2	0.90
Codec 3	7.6	0.95
Codec 4	8.0	1.00
Codec 5	8.4	1.05

Outra análise a ser feita é a da complexidade computacional de cada um dos modos de operação. Para que a complexidade computacional fosse mensurada, um trecho de voz, gravado em um arquivo .wav, foi codificado e o tempo necessário para a codificação foi medido. A comparação do tempo utilizado por cada modo do *codec* permitiu que conclusões a respeito da complexidade computacional fossem obtidas. O tempo medido, em si, não é relevante, o que importa, realmente, é a diferença para cada *codec*.

Para a medição dos tempos, foi utilizado um PC Athlon XP 2000 com 512 MB de memória RAM. A tabela 4.4 mostra os tempos gastos por cada modo de operação.

Tabela 4.4 – Tempo levado por um PC Athlon XP 2000 para codificar um trecho de voz, em segundos.

Codec	Tempo em segundos
Codec 1	0.26
Codec 2	0.45
Codec 3	0.83
Codec 4	1.6
Codec 5	3.13

Com base na tabela 4.4, pode-se constatar um fato que já era esperado. Quanto maior o tamanho do dicionário, maior será o tempo gasto para codificar a voz, ou seja, maior será a complexidade computacional.

Por último será feita a análise dos modos de operação no que diz respeito à qualidade da voz após codificação e decodificação. A maneira ideal para se medir a qualidade de um *codec*, é a realização do teste MOS, descrito na seção 4.2. Esse teste não foi realizado devido ao seu alto custo, principalmente. Para a qualificação dos *codecs*, foi utilizado o teste PESQ (*Perceptual Evaluation of Speech Quality*) através de um *software* de [7] que, através de gravações de voz em arquivos .wav que foram codificados e depois decodificados, dá uma nota que é bem semelhante à nota MOS. Como se deseja que somente nossos *codecs* sejam comprados, isto é, cada modo de operação entre si, o teste PESQ já é suficiente. A realização de um teste que gere uma nota similar ao MOS também é interessante para comparar-se a qualidade do *codec* utilizado com os *codecs* descritos na seção 4.2, mas para este trabalho não é essencial. Para maiores detalhes de como o teste foi feito, ver [7]. A tabela 4.5 mostra os resultados obtidos com cada um dos modos de operação.

Tabela 4.5 – Mostra a nota de cada *codec* no PESQ. Como já era esperado, quanto maior o tamanho do dicionário, melhor a nota.

	PESQ
Codec 1	2.3500
Codec 2	2.6303
Codec 3	2.7029
Codec 4	2.9130
Codec 5	2.9460

Com a análise da tabela 4.5, constata-se aquilo que já era esperado: o modo de operação computacionalmente mais complexo e de pior banda é o que tem a melhor

qualidade. Já o modo de operação com menor complexidade e menor taxa, foi definido como o de menor qualidade.

4.5 Conclusão

Com o término deste capítulo, foi concluída a parte teórica deste trabalho, já tendo abordado todos os principais tópicos que formam um sistema VoIP: os protocolos, QoS e os *codecs*. Neste capítulo, foram apresentados o panorama atual das tecnologias utilizadas para VoIP de codificação e decodificação de VoIP e o *codec* CELP que foi utilizado na implementação deste trabalho. Foi feita também uma análise das características relevantes do *codec* utilizado, tais como qualidade da voz, complexidade computacional e banda utilizada, para sua utilização em VoIP. Com esta análise, chegou-se à conclusão de que não existe um *codec* ideal, ou seja, sempre teremos que analisar a questão do compromisso entre a qualidade da voz, a banda utilizada e a complexidade computacional.

Com a finalização da parte teórica do trabalho, serão abordados, no próximo capítulo, vários aspectos da implementação, especialmente do o protocolo utilizado pelo servidor e cliente para trocar o *codec* durante uma chamada.

Capítulo 5 – Aplicação Desenvolvida

5.1 Introdução

Após termos abordado todos os principais componentes de um sistema VoIP e descrito o *codec* a ser usado no projeto, iremos neste capítulo abordar a implementação deste projeto final. Apresentaremos toda a aplicação desenvolvida, o protocolo para troca de *codec* no decorrer da chamada, o critério de seleção adotado pelo servidor para trocar de *codec*, o módulo que mede os parâmetros da rede (*jitter*, atraso e taxa de perda de pacotes) e a interface gráfica desenvolvida.

É importante ressaltar que toda a aplicação foi desenvolvida em C/C++, utilizando somente bibliotecas *open-source*. A aplicação foi desenvolvida para ser utilizada no sistema operacional *Linux*. Durante a implementação foi utilizada a distribuição *SuSe* versão 9.2.

Na seção 5.2, apresentaremos a estrutura do funcionamento da aplicação, explicando a função de cada módulo e *thread*. Na seção 5.3, apresentaremos o protocolo desenvolvido que permite a troca de *codec* durante uma chamada, o que ocorre nas fases de *setup* e de conversa. Na seção 5.4, explicaremos qual é o critério utilizado pelo servidor para decidir se há necessidade de trocar de *codec* e para qual *codec* deve trocar. Na seção 5.5, descreveremos como foi implementado o módulo para medir os parâmetros da rede, como calculamos o atraso e o *jitter* e como medimos a taxa de perda de pacotes. Finalmente, na seção 5.6, descreveremos a interface gráfica com o usuário desenvolvida e como deve ser sua utilização.

5.2 Estrutura do Funcionamento da Aplicação

Nesta seção, apresentaremos a arquitetura e o funcionamento da aplicação desenvolvida. Pode-se dizer que a aplicação é dividida em duas partes: a interface gráfica e o *core* da aplicação, que será o foco desta seção. A interface gráfica será apresentada em uma seção posterior neste capítulo.

Como já mencionado antes, a aplicação desenvolvida trata-se de uma aplicação cliente-servidor. Ambos, cliente e servidor, estão implementados em um único arquivo binário. O usuário, ao iniciar o programa, decide se quer ser o servidor ou cliente.

No total, foram implementadas seis *threads*. Três para o servidor e três para o cliente. Evidentemente, quando a aplicação está funcionando como cliente, somente as três *threads* do cliente são criadas e vice-versa.

Tanto no cliente como no servidor, há uma *thread* para enviar pacotes de voz e outra *thread* para receber pacotes de voz. A terceira *thread* é uma *thread* de gerência. Pode-se dizer que nesta *thread*, reside toda a inteligência do sistema e a inovação proposta. Nesta *thread* ocorre o monitoramento da rede (medição de atraso, taxa de perda de pacotes e *jitter*) e nela o protocolo de troca de *codec* é executado. Na verdade, a gerência exercida no cliente é diferente da gerência no servidor, mas isso será abordado na seção seguinte.

A comunicação entre servidor e cliente se dá através de três canais distintos. Um canal de sinalização, e dois canais de tráfego de voz (*uplink* e *downlink*). O canal de sinalização foi implementado utilizando pacotes TCP, para haver garantia de entrega dos pacotes de sinalização. Para os canais de voz, foram usados pacotes UDP devido principalmente a sua maior rapidez, tendo em vista que transmissão de voz é muito sensível ao atraso. Na prática, mesmo com a perda de um pacote não há tempo para retransmissão.

Após iniciar o programa, o usuário escolhe se deseja atuar como servidor ou cliente. Se escolher ser cliente, precisa fornecer as seguintes informações: a ordem desejada dos *codecs* (qual *codec* tem maior prioridade), IP do servidor, porta TCP do servidor, portas UDP para *downlink* e *uplink*. Após isto, a *thread* de gerência é iniciada. A primeira coisa a ser feita pela *thread* de gerência é tentar abrir a conexão de sinalização com o servidor. Após a abertura da conexão, há a fase de *setup*. Neste momento há a negociação, entre servidor e cliente, de qual *codec* será utilizado inicialmente. Este

processo será detalhado na seção 5.3. Com o término da fase de *setup*, o codificador e o decodificador são inicializados, com os parâmetros combinados na fase de *setup*. Por último são criadas: uma *thread* para receber pacotes de voz e outra para enviar. A *thread* que recebe os pacotes de voz passa a voz codificada recebida pelo decodificador e o resultado da decodificação é enviado para o dispositivo de som, que reproduz a voz nas caixas de som ou *headphone*. A *thread* que envia os pacotes de voz pega voz no dispositivo de som, voz gerada pelo microfone, a codifica e envia para o outro ponto-final. Após a criação das duas *threads* que transmitem pacotes de voz, a chamada já está estabelecida. Com isso a *thread* de gerência fica parada esperando um pedido de troca de *codec* por parte do servidor, que será enviado através do canal de sinalização. Será apresentado na seção 5.3, como ocorre a troca de *codec*.

Caso o usuário escolha ser servidor, precisa somente fornecer a porta TCP na qual o servidor aguarda pedido de conexão e as portas UDP para *downlink* e *uplink*. O procedimento é bem similar ao descrito para o cliente, com poucas diferenças. A *thread* de gerência ao ser criada fica esperando um pedido de conexão TCP, do canal de sinalização, por parte do cliente. A partir do momento em que estão conectados, inicia-se a fase de *setup*. Após esta fase, como no cliente, o codificador e o decodificador são inicializados. Por fim, inicializam-se as duas *threads* restantes, a que envia e a que recebe voz codificada. Após o estabelecimento da chamada, na *thread* de gerência, o servidor fica analisando a rede periodicamente e decide se deseja ou não trocar o *codec*. Caso sim envia uma requisição ao cliente e o *codec* é trocado, segundo o protocolo que será descrito na seção 5.3. Caso o servidor não julgue necessário trocar o *codec*, o servidor volta a analisar a rede. A figura 5.1 mostra o funcionamento do sistema. Há um canal para sinalização, que utiliza pacotes TCP e dois canais de voz que utilizam pacotes UDP. O diagrama de blocos que representa a arquitetura do sistema descrita acima pode ser visto na figura 5.2.



Figura 5.1 – Funcionamento do sistema, um canal para sinalização e dois canais para voz.

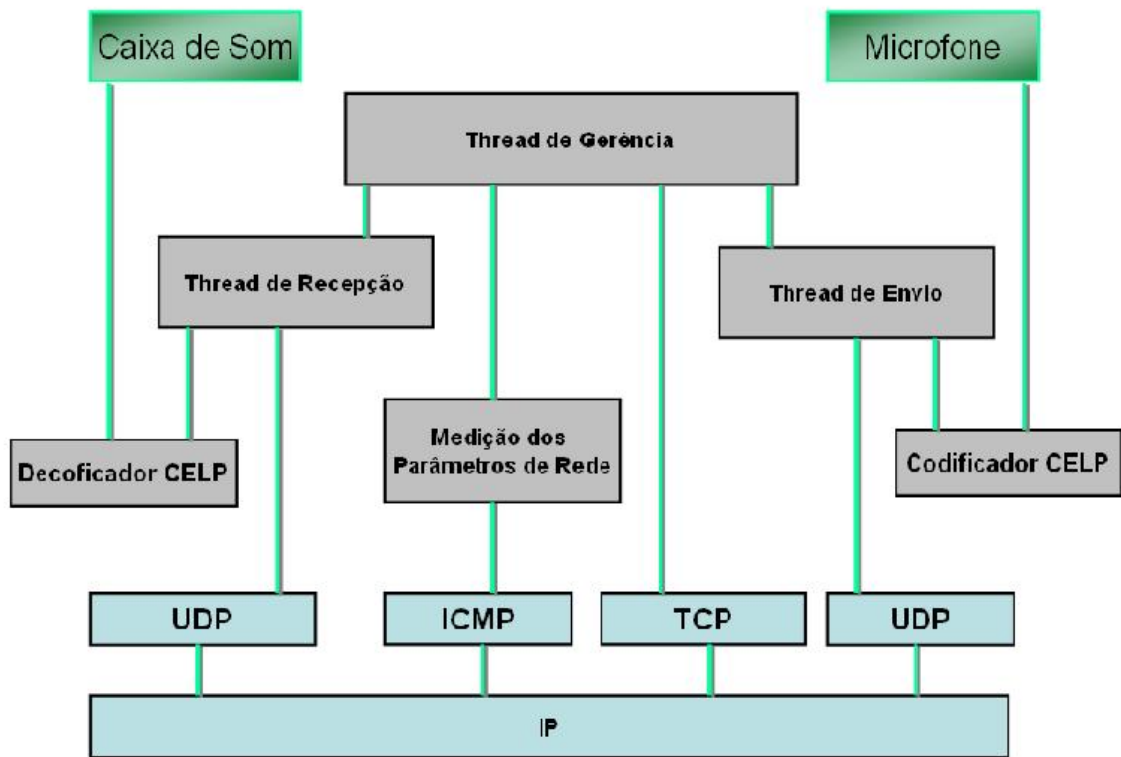


Figura 5.2 – Diagrama de blocos com todos os componentes do sistema.

Após apresentarmos nesta seção o funcionamento geral da aplicação, podemos na próxima seção apresentar o protocolo de comunicação desenvolvido para a troca de *codecs*.

5.3 Protocolo para Troca de Modo de Operação

Nesta seção, abordaremos o protocolo de comunicação entre o cliente e o servidor. Há duas fases de comunicação. A primeira fase é a fase de *setup*. A segunda

fase é a fase de conversa. Nesta fase há troca de pacotes de voz, conversa propriamente dita, e a troca de mensagens de controle, sinalização, pelos módulos de gerência do cliente e do servidor.

Como já mencionado anteriormente, a aplicação desenvolvida trata-se de uma aplicação do tipo cliente-servidor. Foram utilizados três canais de comunicação para estabelecer a chamada: um canal de sinalização e outros dois canais de voz propriamente dita.

O canal de sinalização será utilizado para gerenciar e controlar a chamada, desde seu início até seu término. Pelo fato de ser uma comunicação vital, para a chamada ser bem sucedida, e não há requisito absoluto de desempenho, adotou-se o uso de pacotes TCP para este canal. Pacotes TCP têm a garantia de entrega, com o preço de um desempenho mais baixo, mas atende aos requisitos.

Os outros dois canais são os canais pelos quais a voz codificada, pelo codificador CELP descrito anteriormente, é transmitida. Em um canal, o cliente envia a sua voz codificada e o servidor recebe, no outro o servidor envia a voz codificada e o cliente recebe. Esta comunicação foi implementada através de pacotes UDP, que ao contrário de pacotes TCP, na transmissão de pacotes UDP não há garantia de entrega ou retransmissão. Na transmissão de voz, como já dito anteriormente, atraso é um fator importante, como os pacotes UDP são mais rápidos e mesmo no caso de perda de um pacote, não há tempo para retransmissão, decidimos por utilizar pacotes UDP.

A seguir apresentaremos o protocolo de comunicação desenvolvido nas suas duas fases.

5.3.1 Fase de *Setup*

A fase de *setup* é a fase na qual os dois pontos finais, cliente e servidor, estabelecem a chamada de voz, negociando qual *codec* irão utilizar inicialmente na chamada.

Como toda aplicação cliente-servidor, a comunicação se inicia com o servidor esperando receber pacotes em uma porta. No caso, o servidor aguarda um pedido de conexão TCP, feito pelo cliente, em uma porta definida pelo usuário. Esta é a abertura do canal de sinalização. Após o estabelecimento da conexão TCP, o cliente envia para o servidor uma lista com os *codecs* disponíveis, com uma ordem de preferência, do mais desejado para o menos desejado. O servidor por sua vez verifica quais dos *codecs* requisitados também estão disponíveis. Caso o servidor não tenha nenhum *codec* da lista do cliente, o servidor envia uma mensagem de erro ao cliente, a conexão é terminada e a chamada não é completada.

Ao receber a lista ordenada de *codecs* do cliente, o servidor cria uma lista própria com os codificadores que ele e o cliente tenham, ordenada conforme requisitado pelo cliente. O servidor responde ao cliente ao pedido de conexão com uma mensagem de reconhecimento seguido do codificador escolhido pelo servidor dentre os disponíveis e requisitados pelo cliente. O servidor sempre escolhe como *codec* para iniciar a conversa o primeiro *codec* na preferência do cliente, que ele, servidor, possua, ou seja, o primeiro *codec* da lista do servidor. É importante observar que cada *codec* possui um identificador (*codec_id*). Nas trocas de mensagem entre o servidor e o cliente, no que se refere a *codecs*, somente os *codec_ids* são enviados. A fase de *setup*, descrita acima, está representada na figura 5.3.

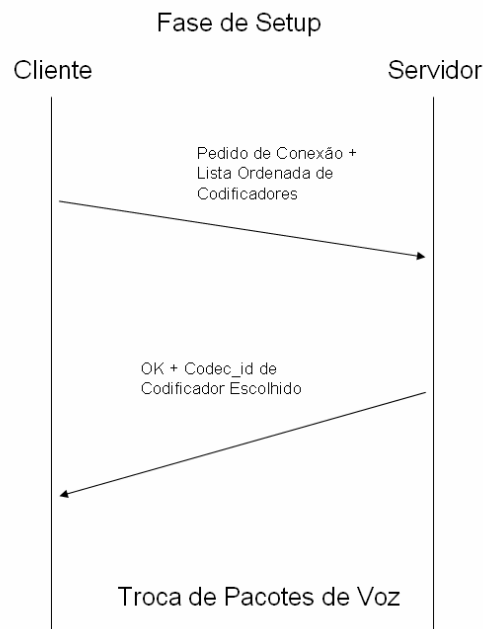


Figura 5.3 – Fase de *setup*, que se dá no canal de sinalização. O cliente faz um pedido de conexão e envia a lista ordenada de *codecs*. O servidor escolhe *codec* qual vai utilizar e responde ao cliente.

Após a negociação do *codec*, os dois canais de voz são abertos, iniciando se a troca de pacotes de voz e fase de conversa que será apresentada na próxima subseção, 5.3.2.

5.3.2 Fase de Conversa

Como já mencionado, a fase de conversa é a fase na qual cliente e servidor já estão trocando pacotes de voz, utilizando pacotes UDP, pelos dois canais de comunicação. Nesta subseção, será descrito o que ocorre no canal de sinalização durante esta fase.

Durante uma conversa, o módulo de gerência do cliente fica aguardando uma mensagem do servidor, no canal de sinalização, requerendo mudança de *codec*. A cada cinco segundos o módulo de gerência do servidor decide se vai requerer ou não uma mudança de *codec*, conforme os critérios que serão apresentados na seção 5.4. Caso se decida não mudar o *codec*, nada acontece, nenhuma mensagem é trocada entre cliente e servidor.

Quando o servidor decide mudar de *codec*, envia pelo canal de sinalização uma mensagem requisitando mudança de *codec* com o *codec_id* do novo *codec*. Após enviar esta mensagem, o servidor aguarda uma mensagem de reconhecimento do cliente, caso o cliente não responda, há um *timeout* (de meio segundos) e o servidor desiste de mudar de *codec* e para de aguardar uma mensagem do cliente. Por sua vez, o cliente ao receber a mensagem de requisição de troca de *codec*, pára de enviar e receber pacotes de voz (bloqueia as *threads* de envio e recebimento de pacotes de voz), isto se faz necessário, pois caso um dos lados envie pacotes de voz codificados com um *codec* não esperado pelo outro lado, acarretará uma queda substancial na qualidade da chamada, então optamos por parar de enviar voz, por um intervalo de tempo imperceptível para o usuário. Após enviar o reconhecimento, o cliente troca de *codec* e fica aguardando um novo reconhecimento do servidor, para reabilitar o envio e o recebimento de pacotes de voz. Caso o servidor não responda dentro de um período de *timeout* de meio segundo, o cliente desiste de trocar de *codec*, retornando para o *codec* anterior, e volta a esperar por uma nova requisição do servidor. Este por sua vez, ao receber o reconhecimento do cliente, muda de *codec* e envia, finalmente, um último reconhecimento para o cliente.

Após uma troca de codificador, o cliente volta ao seu estado de espera por uma mensagem de requisição de troca de *codec* do servidor. A figura 5.4 mostra todo o protocolo de troca de *codec* descrito.

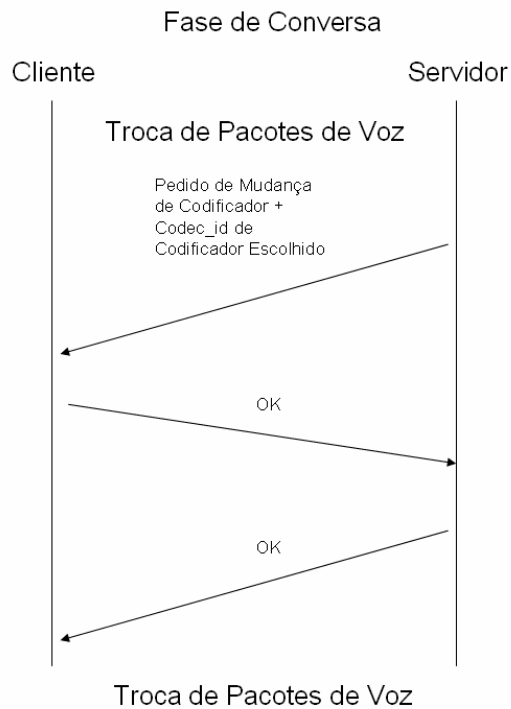


Figura 5.4 – Troca de *codec* durante uma chamada. O servidor decide que é necessário trocar de *codec*, envia uma requisição ao cliente com o novo *codec_id*. O cliente responde com um reconhecimento e aguarda um reconhecimento por parte do servidor.

A chamada pode ser terminada tanto pelo cliente como pelo servidor. Para que isto ocorra, basta o usuário clicar no botão “Parar”, que as *threads*, que transmitem os pacotes de voz e a *thread* são paradas e a chamada é terminada.

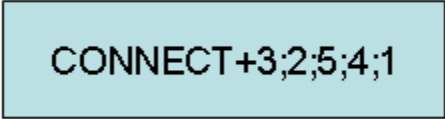
5.3.3 Descrição dos Pacotes

Nesta subseção descreveremos, sucintamente, o conteúdo de cada pacote transmitido pelo canal de sinalização mencionado acima.

Todos os pacotes abaixo estão encapsulados por cabeçalho TCP e cabeçalho IP.

1. Pacote de Pedido de Conexão

Este pacote é enviado pelo cliente para pedir conexão ao servidor. O pacote é composto pela *string* “CONNECT”, mais o caractere ‘+’, seguido da lista de *codec_ids* (cada *codec_id* é constituído por um caractere) separados pelo caractere ‘;’. A figura 5.5 dá um exemplo de como poderia ser o pacote:



CONNECT+3;2;5;4;1

Figura 5.5 – Pacote de pedido de conexão. *String* CONNECT seguida da lista ordenada de *codec_ids* pelo cliente.

Com relação ao tamanho, considerando que o cliente só tem 5 *codecs* disponíveis, e lembrando que estes pacotes utilizam os cabeçalhos TCP e IP teremos em *bytes*: 20 (cabeçalho IP) + 24 (cabeçalho TCP) + 7 (*string* CONNECT) + 1 (caractere '+') + 4 (quatro caracteres '; ') + 5 (um *byte* para cada *codec_id*) = 61 bytes.

2. Pacote de Conexão Aceita e *Codec* Escolhido

Este pacote é enviado pelo servidor para informar ao cliente que a conexão para efetuar a chamada foi aceita e qual foi o codificador escolhido. O pacote é composto pela *string* “CONNECTACK”, mais o caractere '+', seguido do *codec_id* do *codec* escolhido. A figura 5.6 ilustra um exemplo de como poderia ser o pacote:



CONNECTACK+3

Figura 5.6 – Pacote de aceitação de conexão e com o *codec* já escolhido. A *String* CONNECTACK seguida do *codec_id* escolhido pelo servidor que será utilizado inicialmente.

Com relação ao tamanho, teremos em *bytes*: 20 (cabeçalho IP) + 24 (cabeçalho TCP) + 10 (*string* CONNECTACK) + 1 (caractere '+') + 1 (um *byte* para *codec_id*) = 56 bytes.

3. Pacote de Erro

Este pacote é enviado pelo servidor ao cliente quando o servidor não possui nenhum dos codificadores requisitados pelo cliente, ou seja, quando não é

possível completar a chamada. O pacote é composto pela *string* “CONNECTERR”. A figura 5.7 exemplifica como poderia ser o pacote:



CONNECTERR

Figura 5.7 – Pacote de erro. Contém somente a *string* CONNECTERR, que é enviado quando o servidor não possui nenhum dos *codecs* do cliente.

Com relação ao tamanho, teremos em *bytes*: 20 (cabeçalho IP) + 24 (cabeçalho TCP) + 10 (*string* CONNECTERR) = 54 bytes.

4. Pacote de Pedido de Mudança de Codificador

Este pacote é enviado pelo servidor ao cliente quando o servidor decide mudar o codificador. O pacote é composto pela *string* “CHANGECODEC”, mais o caractere ‘+’, seguido do *codec_id* do codificador escolhido. A figura 5.8 dá um exemplo de como poderia ser o pacote:



CHANGECODEC+2

Figura 5.8 – Pacote de pedido de troca de codec, enviado pelo servidor. Contém a *string* CHANGECODEC, seguida pelo *codec_id* do novo *codec*.

Com relação ao tamanho, teremos em *bytes*: 20 (cabeçalho IP) + 24 (cabeçalho TCP) + 11 (*string* CHANGECODEC) + 1 (caractere ‘+’) + 1 (um *byte* para *codec_id*) = 57 bytes.

5. Pacote de reconhecimento

Este pacote é usado tanto pelo servidor como pelo cliente, para informar ao outro lado que um pacote foi recebido e entendido. O pacote é composto pela *string* “ACK”. A figura 5.9 mostra como é o pacote:



Figura 5.9 – Pacote de reconhecimento. Contém a *string* ACK. É utilizado para outro lado que um pacote foi recebido.

Com relação ao tamanho, teremos em *bytes*: 20 (cabeçalho IP) + 24 (cabeçalho TCP) + 3 (*string* ACK) = 47 bytes.

Devido às características do protocolo proposto apresentadas acima, podemos ver que se trata de um protocolo simples e devido ao tamanho dos pacotes e a frequência com a qual são enviadas, não gera tráfego intenso na rede.

Uma questão importante é a compatibilidade do protocolo proposto com os padrões atuais. Como visto no capítulo 2, o H.323 é um protocolo formado por inúmeros protocolos extremamente complexos, além de suportar diversos componentes complexos como *gatekeepers* e *gateways*. Além disto, após o *setup* da chamada, o canal de controle é fechado, ou seja, não há nenhum tipo de preocupação com QoS, isto é um requisito da rede. No protocolo proposto há uma preocupação com QoS e por isto a possibilidade de se trocar de *codec*. Há também outras características do H.323 que tornam o protocolo proposto incompatível com o H.323, como por exemplo, o fato da transmissão dos pacotes de voz usar RTP/RTCP e de haver a possibilidade de criptografia.

Com relação ao SIP, há também problemas de compatibilidade. Da mesma forma que H.323, SIP possui vários componentes complexos como *Proxy Server* e *Redirect Server*, os pacotes, apesar de simples, são totalmente diferentes dos pacotes do protocolo proposto, não há controle dos fluxos no decorrer da chamada. Não está previsto no SIP um protocolo de controle que, por exemplo, permita a troca de *codec* no decorrer de uma chamada.

Na verdade, o objetivo do protocolo proposto não é ser compatível com SIP ou H.323, mas sim demonstrar que em versões futuras destes protocolos pode-se implementar a troca de *codec* no decorrer da chamada. As vantagens seriam, por exemplo, no caso da rede estar em más condições a aplicação poderia tomar alguma medida para melhorar a qualidade da chamada do usuário, ou ainda, em muitos sistemas VoIP, há um limite máximo de chamadas devido a limitações de banda. Um bom exemplo seria uma empresa na qual há um limite de cinco chamadas simultâneas entre duas filiais e o *codec* padrão é G.711, que é um *codec* de ótima qualidade, porém consome muita banda. Caso cinco chamadas já estejam estabelecidas, um novo usuário não conseguirá estabelecer uma sexta chamada, pois não haverá banda disponível. Para resolver esta questão, poderia-se trocar o *codec* de todas as chamadas para, por exemplo, G.729, que tem uma qualidade inferior a G.711, mas requer menos banda. Fazendo isto, novas chamadas poderiam ser estabelecidas. Quando o número total de chamadas voltasse a ser inferior a 5, poderia-se novamente trocar o *codec* das chamadas em curso para um *codec* de maior qualidade.

Levando-se em consideração as vantagens apresentadas acima, a proposta deste projeto com esse protocolo é adicionar tanto ao SIP como ao H.323, em versões futuras, a possibilidade de se trocar o *codec* durante uma chamada. Uma possibilidade de implementação seria, como ambos os protocolos utilizam RTP e RTCP, significa que os pontos-finais envolvidos em uma chamada conhecem a situação atual da rede (atraso, *jitter* e taxa de perda de pacotes) através de RTCP. Com isso, basta modificar H.323 e SIP para extrair estas informações de RTCP e a partir disto tomar a decisão de trocar ou não o *codec*.

5.4 Inteligência da Aplicação

Após termos apresentado o protocolo para troca de *codec*, iremos nesta seção apresentar o critério de seleção adotado pelo servidor para decidir se troca ou não de *codec*. O critério de seleção do servidor para mudar de *codec* será sempre baseado na situação atual da rede, ou seja, no *jitter*, no atraso, na taxa de perda de pacotes e no *codec* atualmente utilizado.

Para este trabalho utilizaremos o valor máximo aceitável de atraso igual a 150 ms (valor do padrão G.114 da ITU [4]), 30 ms para *jitter* e 1% de taxa de perda de pacotes. Se ao medir a rede, todas essas condições forem atendidas, o servidor irá qualificar a rede como em boas condições. Caso uma ou mais condições não seja satisfeita, o servidor irá qualificar a rede como rede em más condições.

Durante a fase de conversa, a cada cinco segundos, o servidor verifica os parâmetros da rede. Se todos os parâmetros abaixo dos limiares estabelecidos, o servidor verifica qual é o *codec* utilizado atualmente. Se já for o melhor *codec* (melhor *codec* na ordenação do cliente), o servidor não faz nada. Se o *codec* atual não for o melhor, o servidor vai à lista de *codecs* ordenada pelo cliente, e troca o *codec* atual por um *codec* melhor, o seguinte na lista. Caso algum dos parâmetros da rede não esteja de acordo com as restrições estabelecidas, o servidor vai à lista de *codecs* ordenada pelo cliente e troca o *codec* atual por um *codec* pior que o atual, pelo *codec* anterior ao atual na lista. Caso o *codec* atual já seja o último da lista, o servidor não faz nada.

Para um total entendimento, é necessário perceber que o conceito de um *codec* ser melhor ou pior, está relacionado à lista de *codecs* ordenada pelo cliente, ou seja, o usuário do cliente que escolhe a ordem dos *codecs* antes de estabelecer cada chamada. É também importante perceber que, como a cada nova chamada o cliente escolhe uma nova lista de *codecs*, um *codec* que em uma chamada era o melhor, pode estar, por exemplo, em quarto lugar na lista em outra chamada. Preferiu-se não arbitrar no *software*, se um *codec* é melhor ou pior que outro esta escolha será feita pelo usuário do cliente antes de efetuar cada chamada, pois na verdade, a percepção de melhor ou pior depende muito do estado atual da rede, do PC que o usuário está utilizando, da percepção e necessidade de qualidade por parte do usuário.

5.5 Analisador da Rede

Nesta seção apresentaremos como foi implementado o módulo que analisa os parâmetros da rede durante uma chamada. Como já mencionado no capítulo 3, os

parâmetros de rede relevantes em transmissões de voz são: atraso, variação do atraso (*jitter*) e taxa de perda de pacotes. Este módulo tem como objetivo medir esses três parâmetros.

Na implementação deste módulo utilizamos a biblioteca de ICMP (*Internet Control Message Protocol*) [8]. Nesta biblioteca há um método que envia um pacote *Echo Request* para um destino especificado e aguarda a resposta.

Para medir a taxa de perda de pacotes, simplesmente dividimos o número de pacotes que foram enviados com sucesso, isto é os pacotes para os quais recebemos resposta do destino, pelo número total de pacotes enviados. Subtraindo este resultado de um, e multiplicando por cem, temos a taxa de perda de pacotes percentualmente.

Para medir o atraso, medimos quanto tempo demora para recebermos a resposta de um pacote enviado. Como normalmente, em uma avaliação da rede, são enviados vários pacotes, medimos o atraso médio, que é a soma dos atrasos divididos pelo número de pacotes. Nesta contagem só são considerados os pacotes que receberam resposta.

Para o cálculo do *jitter*, medimos a diferença entre atrasos consecutivos. Calculamos o módulo, depois somamos todos os módulos das diferenças entre os atrasos e dividimos pelo número de atrasos, assim obtendo o *jitter* médio.

Um detalhe importante é que os pacotes ICMP padrão que foram implementados na biblioteca utilizada são formados por 24 bytes, 20 bytes do cabeçalho IP, mais 4 bytes do cabeçalho ICMP [8]. Já os pacotes de voz transmitidos têm um *overhead* de 49 bytes, 20 bytes do cabeçalho IP, 8 bytes do cabeçalho UDP e 21 bytes (no pior caso) de voz codificada. Medir o atraso, o *jitter* e a taxa de perda de pacotes com pacotes de tamanhos tão diferentes, talvez não nos levasse a uma boa medida. Para resolver isto acrescentamos 25 bytes inócuos no pacote ICMP da biblioteca, somente para que o pacote enviado ficasse do mesmo tamanho do pacote de voz.

5.6 Interface Gráfica

Nesta seção apresentaremos a interface gráfica com usuário desenvolvida. Como toda a aplicação, a interface foi desenvolvida em C/C++, escolheu-se a biblioteca WX-Widgets em C++ para desenvolver a interface gráfica. O objetivo de desenvolver uma interface gráfica é de tornar a aplicação mais amigável para o usuário e mais acessível a usuários de diferentes níveis de conhecimento de computação.

Ao executar o programa, o usuário verá uma janela como mostrado na figura 5.10. O usuário deve clicar em Principal no menu e posteriormente em Configuração. Com isso uma janela de configuração vista na figura 5.11 irá se abrir. Nesta janela o usuário deve escolher se quer atuar como servidor ou como cliente.

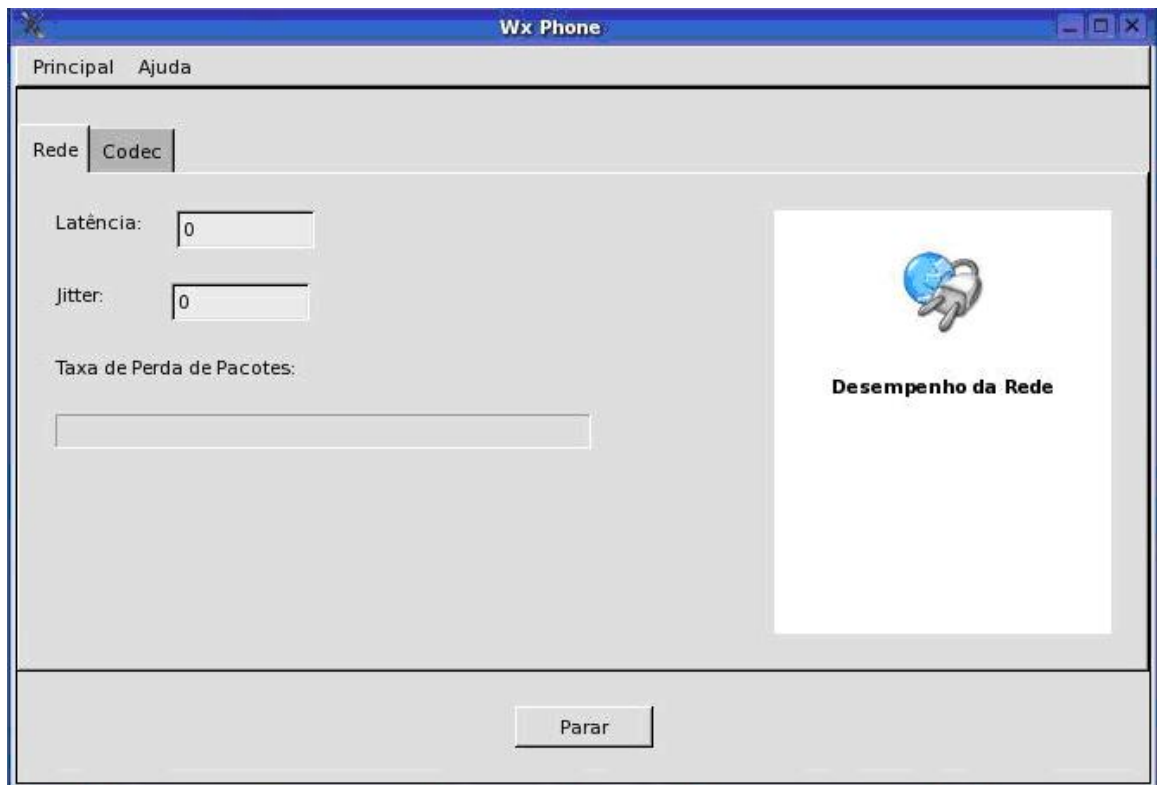


Figura 5.10 – Janela inicial da aplicação.

Caso escolha ser servidor, deve escolher em qual porta TCP deseja que seja estabelecida conexão TCP que implementa o canal de sinalização, depois escolher as

duas portas UDP nas quais se implementam os canais de voz. Após isto, basta clicar em OK. Ao clicar em OK o servidor estará esperando uma conexão do cliente.

The image shows a Windows-style dialog box titled "Start Celp". At the top left, there is a section "Escolha o modo de Operação" with two radio buttons: "Servidor" (which is selected) and "Cliente". To the right of this section is a button labeled "Iniciar". Below this, there are four input fields: "IP:" (with four small boxes for digits), "Porta TCP:", "Porta UDP Local:", and "Porta UDP Remota:". A horizontal line separates this from the bottom section, "Escolha a ordem dos codecs:". This section contains five labels: "Primeiro Codec:", "Segundo Codec:", "Terceiro Codec:", "Quarto Codec:", and "Quinto Codec:". Each label is followed by a dropdown menu, all of which currently display "Primeiro Codec".

Figura 5.11 – Janela de configuração, na qual o usuário escolhe se atuará como servidor ou cliente. Como o padrão é servidor, nesta janela também são configurados os parâmetros do servidor.

Caso o usuário escolha ser cliente, deve fornecer o endereço IP do servidor, a porta TCP no servidor e as duas portas UDP nas quais vão ser implementados os canais de voz. O cliente ainda precisa definir a lista dos *codecs*, ou seja, qual é sua ordem de preferência, como pode ser visto na figura 5.12. Após isto basta clicar em Iniciar, fazendo com o que o cliente se conecte ao servidor.

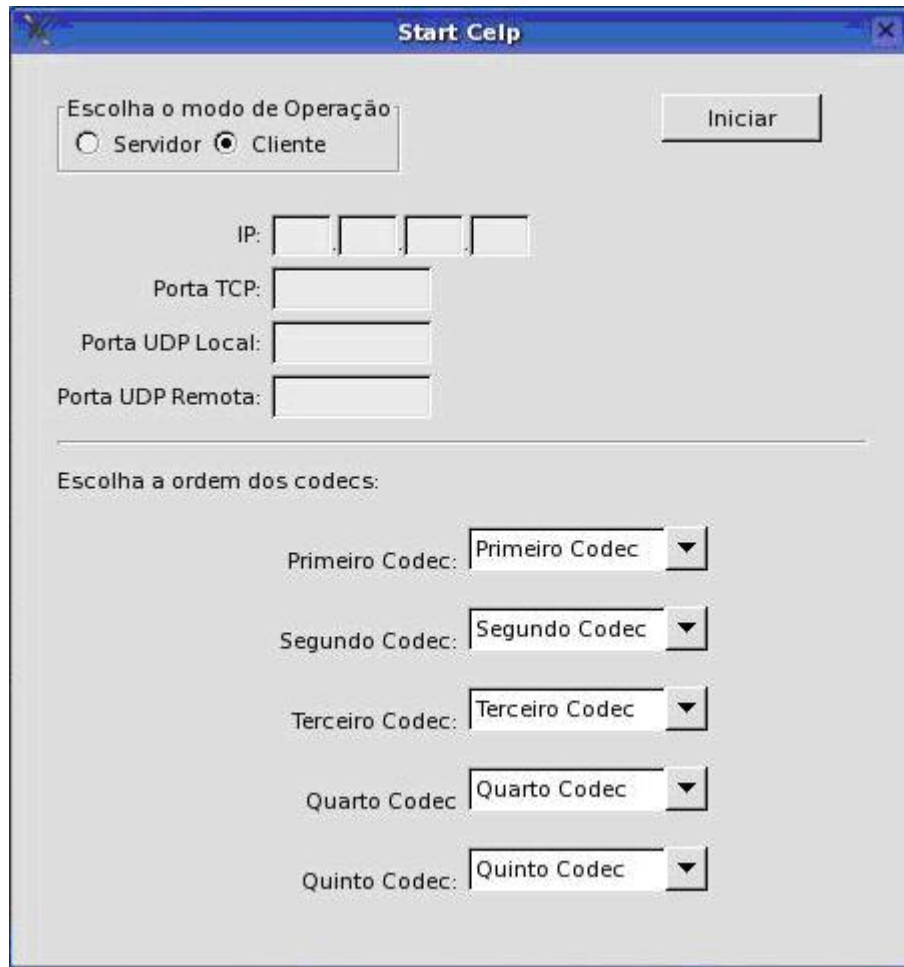


Figura 5.12 – Janela de configuração do cliente. Nesta janela, o cliente insere o endereço IP do servidor, a porta TCP, as portas UDP e a ordem de prioridade dos *codecs*.

Após isto a conversa estará iniciada. O usuário tanto do cliente como do servidor terão duas abas para verificar o *status* da chamada. Uma aba mostra os parâmetros da rede (*jitter*, atraso e taxa de perda de pacotes), como indicado na figura 5.13. A outra aba mostra as características do *codec* atual (taxa do *codec* em bps, qualidade no teste PESQ, tamanho do dicionário fixo, tamanho do dicionário adaptativo, número de coeficientes LSF e gamma), como ilustrado na figura 5.14.

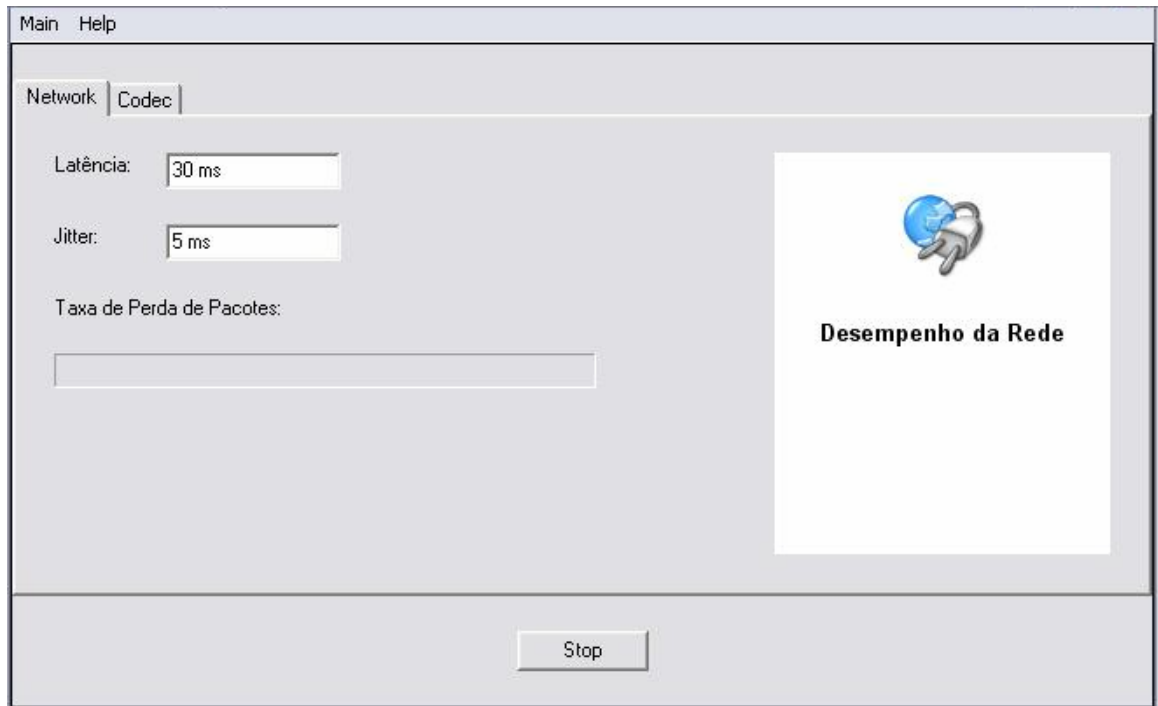


Figura 5.13 - Aba da janela principal da aplicação que mostra, após o início da conversa os parâmetros da rede: latência, jitter e taxa de perda de pacotes.

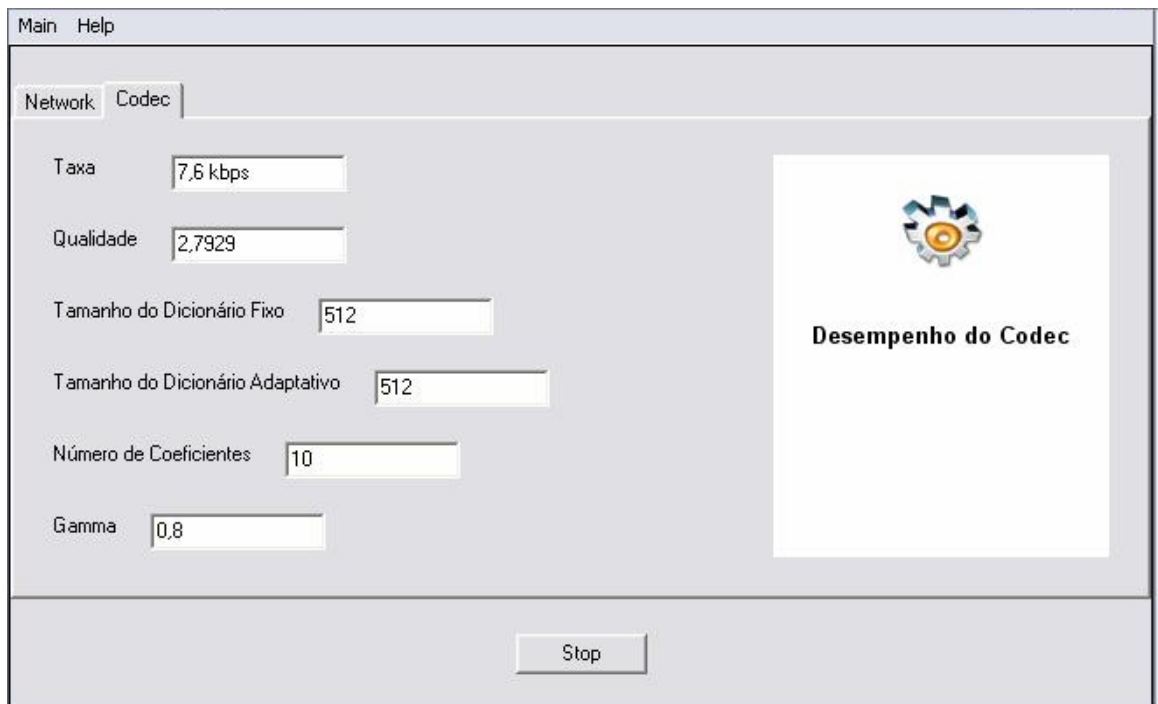


Figura 5.14 - Aba da janela principal da aplicação que mostra, após o início da conversa os parâmetros do codec atual: taxa do *codec* em bps, a qualidade no teste PESQ, o tamanho do dicionário fixo, o tamanho do dicionário adaptativo, o número de coeficientes LSF e o gamma.

Para terminar a chamada e fechar a aplicação basta clicar em parar, que as conexões serão fechadas e a aplicação terminada.

5.7 Conclusão

Neste capítulo descrevemos o sistema implementado neste projeto. Apresentamos o funcionamento e a estrutura da aplicação desenvolvida. Descrevemos o protocolo de comunicação entre os pontos finais, que permite que o *codec* seja trocado no decorrer da chamada. Apresentamos também o critério de seleção utilizado pelo servidor para decidir se há necessidade de trocar o *codec* durante uma chamada, o módulo que mede as características da rede (atraso, *jitter* e taxa de perda de pacotes) e por último, a interface gráfica com usuário.

O módulo de medição dos parâmetros de rede está apresentando imprecisão em seu funcionamento, o que compromete o funcionamento do sistema. Uma solução para esta questão seria utilizar os protocolos RTP/RTCP, pois com RTCP teria-se uma medição mais precisa dos parâmetros da rede.

No próximo capítulo iremos concluir o projeto, destacando os principais pontos do projeto e trabalhos futuros que enriqueceriam o que já foi implementado até agora.

Capítulo 6 - Conclusão

6.1 Conclusão

Neste capítulo, finalmente, concluiremos o projeto. Será feito um breve resumo do que foi abordado no trabalho, dando ênfase aos tópicos mais relevantes. Serão apontados, também, possíveis trabalhos futuros que contribuam para o aperfeiçoamento do sistema desenvolvido.

O projeto teve como objetivo abordar a conjuntura atual dos sistemas VoIP, seus principais componentes e fazer uma proposta que contribua para a resolução do problema de QoS. Assim, faz-se um sistema em que não só a rede tenha a atribuição de prover QoS, mas também a aplicação em cada ponto final.

Para propor alguma melhoria em QoS, foi necessário, primeiramente, estudar como os sistemas VoIP são constituídos. Para isto, dedicou-se o Capítulo 2, que aborda as arquiteturas e os protocolos atuais de VoIP. No que diz respeito às arquiteturas, mostraram-se seus principais componentes com sua respectiva função dentro do sistema VoIP como um todo. Já com relação aos protocolos, os mais utilizados são H.323 e SIP. Ambos os protocolos foram apresentados, evidenciando suas principais características. A partir disto foram apontadas suas principais vantagens e desvantagens.

No Capítulo 3, foi abordado o conceito de QoS em VoIP. Inicialmente, foi mostrada a necessidade de QoS em um sistema VoIP. Foram apresentadas as duas principais arquiteturas atuais de QoS: *Diffserv* e *Intserv*. Foram mostradas suas principais características e foi realizada uma comparação direta entre ambas as arquiteturas.

Para completar todos os elementos que compõem um sistema VoIP, no Capítulo 4, os codificadores de voz foram tratados. Primeiramente, foi mostrada a necessidade de usarmos codificadores de voz para transmitir voz por redes IP. Também foram mostrados os codificadores mais utilizados atualmente, como por exemplo: G.711, G.723 e G.729, suas principais características, vantagens e desvantagens, especialmente com relação à

qualidade da voz e à sua taxa de compressão. Foi feita ainda a apresentação do codificador CELP utilizado na implementação, enfatizando os modos de operação estudados do codificador e seus desempenhos no que diz respeito à complexidade computacional, à qualidade da voz e à taxa de compressão.

Todo o detalhamento de como a aplicação foi desenvolvida foi mostrado no Capítulo 5. Inicialmente, foi apresentada a arquitetura da aplicação e qual a função de cada parte do *software*. Com isso, foi apresentado o protocolo de comunicação entre o cliente e o servidor, que permite a troca do codificador durante uma chamada e o critério de escolha adotado pelo servidor para escolher se troca de codificador ou o mantém, em função dos parâmetros da rede. Nesse capítulo, também foi mostrado como foi implementado o módulo que mede os parâmetros de rede (atraso, *jitter* e taxa de perda de pacotes). Por fim, foi apresentada a interface gráfica desenvolvida com a biblioteca *Wx-Windows* e um guia de como utilizar a aplicação através da interface gráfica.

É importante, mais uma vez, ressaltar que o protocolo desenvolvido não tem a pretensão de resolver todos os problemas de QoS, descartando as arquiteturas atuais de QoS em um sistema VoIP, mas tem como objetivo, ser mais uma alternativa para melhorar QoS, especialmente quando se deseja utilizar VoIP em uma rede que não tenha QoS. Como a troca de mensagens de sinalização não é intensa, não há problema de aumento de tráfego na rede como RSVP e como o protocolo atua na aplicação não há necessidade de se adicionar nada nos roteadores entre os pontos finais. A utilização somente do protocolo proposto não é o bastante para resolver o problema de QoS, mas pode ajudar, diminuindo a banda utilizada, em um momento de congestionamento da rede.

6.2 Trabalhos Futuros

VoIP é um assunto muito vasto e relativamente recente, devido a isto, são inúmeras as possíveis continuações para este trabalho. Dentre elas, podemos destacar as seguintes propostas:

- Utilizar RTP/RTCP no transporte da voz.
- Supressão de silêncio.
- Implementar interpolação dos pacotes de voz no *codec* para o caso de perda de pacote.
- Tornar a aplicação *peer-to-peer* (p2p).
- Fazer com que a aplicação suporte conferências multiponto.

Referências Bibliográficas

- [1] F. C. da C. B. Diniz, *Implementação de um Codificador de Voz CELP em Tempo Real*, Projeto Final, DEL-POLI-UFRJ, Maio, 2003
- [2] R. da S. Maia, *Codificação CELP e Análise Espectral de Voz*, Tese de M.Sc., PEE-COPPE/UFRJ, Março, 2000.
- [3] Eduardo F. Brasil, *Transmissão de Voz pela Internet*, Relatório de Estágio, DEL-POLI-UFRJ, Dezembro, 2004.
- [4] Cisco Systems, Inc. (<http://www.cisco.com>)
- [5] A. J. S. Silva, “Qualidade de Serviço em VoIP – Parte I”, *Boletim Bimestral sobre Tecnologia de Redes da RNP*, volume 4, Maio, 2000.
- [6] A. L. P. Schmidt, “O Protocolo RSVP e o Desempenho de Aplicações Multimídias”, *Boletim Bimestral sobre Tecnologia de Redes da RNP*, volume 4, Maio, 2000.
- [7] Mapping function for transforming P.862 raw result scores to MOS-LQO, ITU-T, P.862-1, Novembro, 2003
- [8] M. de L. Azambuja, “*Biblioteca ICMP em C++*”, Projeto Final da Disciplina Linguagens de Programação, DEL-POLI-UFRJ, Junho, 2003.
- [9] Oliver Hersent, David Guide, Jean-Pierre Petit, *Telefonia IP: Comunicação Multimídia Baseada em Pacotes*, Makron Books, 2002
- [10] W. Richard Stevens, *UNIX Network Programming, Volume 1, Second Edition: Networking APIs: Sockets and XTI*, Prentice Hall, 1998
- [11] W. Richard Stevens, *UNIX Network Programming, Volume 2, Second Edition: Interprocess Communications*, Prentice Hall, 1999
- [12] WxWidgtes (<http://www.wxwindows.org>)