UNIVERSIDADE ESTADUAL DE CAMPINAS
FACULDADE DE ENGENHARIA ELÉTRICA E COMPUTAÇÃO
DEPARTAMENTO DE COMUNICAÇÕES

IMPLEMENTAÇÃO DE UM SISTEMA SIP PARA O SISTEMA OPERACIONAL LINUX

Dissertação apresentada a Faculdade de Engenharia Elétrica e de Computação da Universidade Estadual de Campinas, como parte dos requisitos exigidos para a obtenção do título de Mestre em Engenharia Elétrica.

AUTOR: DAVISON GONZAGA DA SILVA
ORIENTADOR: LEONARDO DE SOUZA MENDES

BANCA EXAMINADORA:

Prof. Dr. Leonardo de Souza Mendes - DECOM/FEEC/UNICAMP
Prof. Dr. Antônio Marcos Alberti – Dr. pela FEEC/UNICAMP
Prof. Dr. Renato Baldini Filho - DECOM/FEEC/UNICAMP
Prof. Dr. Luís Geraldo Pedroso Meloni - DECOM/FEEC/UNICAMP

Campinas, 12 de dezembro de 2003

FICHA CATALOGRÁFICA ELABORADA PELA
BIBLIOTECA DA ÁREA DE ENGENHARIA - BAE - UNICAMP

Silva, Davison Gonzaga da
Si38i Implementação de um sistema SIP para o sistema operacional linux /Davison Gonzaga da Silva. --Campinas, SP: [s.n.], 2003.

Orientador: Leonardo de Souza Mendes.
Dissertação (mestrado) - Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e de Computação.

1. Internet (Redes de computação). 2. TCP/IP (Protocolo de rede de computação). 3. Linux (Sistema operacional de computador). 4. C++ (Linguagem de programação de computador). 5. Trafego telefônico. 6. Programação em tempo-real. I. Mendes, Leonardo de Souza. II. Universidade Estadual de Campinas. Faculdade de Engenharia Elétrica e de Computação. III. Título.

"Uma descoberta consiste em ver aquilo que todos viram e pensar naquilo que ninguém pensou."
Albert Szent-Gyorgi

Aos meus pais Joel e Glória.

Aos meus irmãos David e Joelma.

AGRADECIMENTOS

Eu gostaria de agradecer a Deus por ter me dado a vida e por guiar sempre o meu caminho.

Aos meus pais, pela educação, pelo amor, carinho e compreensão que sempre tiveram comigo.

Aos meus irmãos, que sempre estiveram juntos comigo nesta nova etapa da minha vida.

Ao Professor Leonardo de Souza Mendes, pela oportunidade, orientação e ensinamentos durante o meu mestrado.

Aos meus amigos Ricardo Külzer e Gláucia Külzer, pela amizade sincera que eles sempre demonstraram ter comigo.

Aos meus colegas do LARCOM Henrique, Wildner, Rodrigo, Leonardo Inácio, Marcelo, Cláudio, Ana Luiza, Maurício, Antônio, Ricardo Alberti, Gean, Ernesto Barrientos e Ana Carolina, pelo companheirismo e descontrações durante estes dois anos de mestrado.

À Meire, pela amizade, descontração, ensinamentos e apoio que sempre teve comigo durante o meu mestrado, sem contar pelos deliciosos bolos e outras delícias que a Meire fez durante estes dois anos.

Aos professores da UFES, pela minha formação universitária. Em especial, eu gostaria de agradecer à Professora Jussara Fardin, pelo apoio dado para que eu viesse para a UNICAMP; ao Professor Gilberto Drumond, pelos ensinamentos que eu obtive durante o meu Projeto de Graduação; ao Professor Teodiano Freire, pela oportunidade de aprendizado que me foi dada através da monitoria da disciplina Circuitos Elétricos I; e ao Professor Hans-Jorg, pelo aprendizado que obtive junto ao projeto de iniciação científica.

Aos meus amigos de Vitória; Beatriz, Kátia, Luciano, Leonardo, Martha, Jorge Raphael, Wagner, Gabriel, Juliano, Robson, Roziane, e todos os outros não citados aqui, pela amizade e suporte dado a mim durante esta nova etapa na minha vida.

E a todos que, diretamente ou indiretamente, contribuíram para a conclusão deste trabalho.

RESUMO

Este trabalho apresenta a implementação de um Sistema de VoIP usando o Protocolo SIP. Este Sistema SIP foi desenvolvido para o Linux, usando-se a linguagem C++ em conjunto com a biblioteca QT. O Sistema SIP é composto de três entidades básicas: o Terminal SIP, o Proxy e o Servidor de Registros. O Terminal SIP é a entidade responsável por estabelecer sessões SIP com outros Terminais SIP. Para o Terminal SIP, foi desenvolvida uma biblioteca de acesso à placa de áudio, que permite a modificação dos parâmetros do *driver* do dispositivo. O Proxy é a entidade que processa a mensagem para a enviar ao destino correto. O Servidor de Registros é a entidade que mantém uma lista de ligações de endereços SIP e endereços reais do domínio de sua responsabilidade. Com estes componentes, o Sistema SIP pode ser usado em vários ambientes de rede e para diversas aplicações, como o ensino a distância.

ABSTRACT

This work presents an implementation of a VoIP system using the SIP protocol. This SIP system was developed for Linux, using C++ language using the QT library. The SIP system is composed by three basic entities: the SIP Terminal, the Proxy and the Registers Server. The SIP Terminal is the entity responsible for establishing SIP sessions with others SIP Terminals. For the SIP Terminal, it was developed a library to access the audio card, which allows modifying the device driver parameters. Proxy is the entity that processes the message to send to the correct destination. The Registers Server is the entity that keeps a list of SIP addresses links, and real addresses of the domain under this server responsibility. With these components, the SIP System can be used in several network environments and for diverse applications, as e-learning for example.

ÍNDICE ANALÍTICO

1. Introdução 1
1.1 Voz sobre IP 1
1.2 Protocolo H.323 3
1.3 Protocolo SIP 3
1.4 Motivações para o Desenvolvimento deste Trabalho 4
1.5 O Sistema SIP Implementado 4
1.6 Organização do Trabalho 5
2. Protocolo H.323 7
2.1 Introdução 7
2.2 Recomendação H.323 7
2.3 Componentes de um Sistema H.323 8
2.3.1 Terminais H.323 8
2.3.2 Gateway 10
2.3.3 Gatekeeper 10
2.3.4 Unidade Controle Multiponto 11
2.4 Sinalização H.323 12
2.4.1 Fase A: Inicialização da Chamada 12
2.4.2 Fase B: Comunicação Inicial e Troca de Funcionalidades 13
2.4.3 Fase C: Estabelecimento da Comunicação Audiovisual 15
2.4.4 Fase D: Serviços da Chamada 16
2.4.5 Fase E: Finalização da Chamada 16
2.5 Considerações Finais sobre o Protocolo H.323 17
3. SIP: Session Initiation Protocol 19
3.1 Introdução 19
3.2 Protocolo SIP 19
3.3 Estrutura do Protocolo SIP 20
3.4 Mensagens SIP 22
3.4.1 Mensagens de Requisição 23
3.4.2 Mensagens de Resposta 24
3.5 Entidades SIP 25

3.6 Visão Geral do Funcionamento do Protocolo 25
3.6.1 Registrando um Terminal SIP 26
3.6.2 Estabelecendo uma Sessão SIP 28
3.6.3 Encerrando-se a Sessão 33
3.6.4 Cancelando uma Requisição INVITE 34
3.6.5 Descobrindo as Capacidades dos Terminais SIP 37
4. Protocolo SDP 41
4.1 Introdução 41
4.2 Uso do Protocolo SDP 41
4.2.1 Funcionalidades do Protocolo SDP 42
4.3 Descrição do Protocolo SDP 42
4.3.1 O Campo v 44
4.3.2 O Campo o 44
4.3.3 Os Campos “e” e s 45
4.3.4 O Campo c 45
4.3.5 O Campo t 45
4.3.6 O Campo m 46
4.3.7 Campo a 46
5. Descrição Funcional do Sistema SIP Implementado 49
5.1 Introdução 49
5.2 Terminal SIP 50
5.2.1 Transação do Usuário 51
5.2.2 Camada de Transação 58
5.2.3 Camada de Transporte 67
5.2.4 Camada de Acesso à Placa de Áudio 68
5.3 Servidor de Registros 69
5.4 Proxy 70
5.5 Funcionamento do Sistema SIP Implementado 71
5.6 Terminal SIP – Interface Gráfica 72
6. Exemplos do Funcionamento do Sistema SIP 78
6.1 Sinalização SIP 78

6.1.1 Registro de um Terminal 78
6.1.2 Estabelecimento de uma sessão 80
6.1.3 Finalização de uma Chamada 83
6.2 Configurações da Placa de áudio 83
7. Conclusões 87
7.1 Trabalhos Futuros 88
8. Referências Bibliográficas 89
Apêndice I. Campos das Mensagens SIP 93
A I.1. Accept 93
A I.2. Accept-Encoding 93
A I.3. Accept-Language 93
A I.4. Alert-Info 94
A I.5. Allow 94
A I.6. Authentication-Info 94
A I.7. Authorization 94
A I.8. Call-ID 95
A I.9. Call-Info 95
A I.10. Contact 95
A I.11. Content-Disposition 95
A I.12. Content-Encoding 96
A I.13. Content-Language 96
A I.14. Content-Length 96
A I.15. Content-Type 96
A I.16. CSeq 97
A I.17. Date 97
A I.18. Error-Info 97
A I.19. Expires 97
A I.20. From 98
A I.21. In-Reply-To 98
A I.22. Max-Forwards 98
A I.23. Min-Expires 98

A I.24. MIME-Version 99
A I.25. Organization 99
A I.26. Priority 99
A I.27. Proxy-Authenticate 99
A I.28. Proxy-Authorization 100
A I.29. Proxy-Require 100
A I.30. Record-Route 100
A I.31. Reply-To 100
A I.32. Require 101
A I.33. Retry-After 101
A I.34. Route 101
A I.35. Server 102
A I.36. Subject 102
A I.37. Supported 102
A I.38. Timestamp 102
A I.39. To 102
A I.40. Unsupported 103
A I.41. User-Agent 103
A I.42. Via 103
A I.43. Warning 103
A I.44. WWW-Authenticate 105
Apêndice II. Campos da Mensagem SDP 107
A II.1. Versão do Protocolo (v) 107
A II.2. Origem (o) 107
A II.3. Nome da Sessão (s) 107
A II.4. Informação sobre a Sessão e a Mídia (i) 108
A II.5. URI (u) 108
A II.6. Endereço de e-mail (e) e número do telefone (p) 108
A II.7. Informações sobre a Conexão (c) 108
A II.8. Largura de Banda (b) 109
A II.9. Tempo de Duração da Sessão (t) e de repetição (r) 109

A II.10. Ajuste do Tempo (z) 109
A II.11. Chaves de Criptografia (k) 110
A II.12. Campos Descritores de Mídia (m) e de Atributos (a) 110
Apêndice III. Biblioteca QT 111
A III.1. Introdução 111
A III.2. Uma aplicação em QT 111
A III.3. Sinais e Slots 112
Apêndice IV. Detalhes da Implementação do Sistema SIP 115
A IV I. Telefone SIP 115
A IV I.I. QTSIPPhone 115
A IV I.II. CUac_core 118
A IV I.III. CUas_core 120
A IV I.IV. CInvite_client_trans 121
A IV I.V. Cnon_Invite_Client_trans 122
A IV I.VI. CInvite_server_trans 123
A IV I.VII. Cnon_Invite_Server_trans 123
A IV I.VIII. CMessage 124
A IV I.IX. CMyAudio 125
A IV II. Proxy e o Servidor de Registros 126
A IV II.I. CProxy 126
A IV II.II. CRegistrar 127

ÍNDICE DE FIGURAS

Figura 1. Exemplos de cenários de VoIP 2
Figura 2. Terminal H.323 8
Figura 3. Cenário da sinalização H.323 12
Figura 4. Troca de mensagens da Fase A 13
Figura 5. Troca de mensagens de abertura do canal de controle H.245. 14
Figura 6. Troca de mensagens das funcionalidades do terminal 15
Figura 7. Troca de Mensagens da Fase C – Estabelecimento dos canais de áudio bidirecionais 15
Figura 8. Sinalização para o estabelecimento dos canais de áudio unidirecionais 16
Figura 9. Finalização da chamada H.323 17
Figura 10. Comunicação básica entre as camadas do protocolo SIP 21
Figura 11. Formato das mensagens SIP 22
Figura 12. Cenário da transação REGISTER 26
Figura 13. Mensagem REGISTER 26
Figura 14. Mensagem de resposta à requisição REGISTER 27
Figura 15. Cenário da transação INVITE 29
Figura 16. Mensagem INVITE enviada pelo Terminal SIP 1 30
Figura 17. Resposta 100 Trying enviada pelo Proxy 1 30
Figura 18. Mensagem INVITE enviada pelo Proxy 1 31
Figura 19. Resposta enviada pelo Terminal SIP 2 32
Figura 20. Mensagem ACK enviada pelo Terminal SIP 1 33
Figura 21. Mensagem BYE enviada pelo Terminal SIP 2 33
Figura 22. Resposta 200 Ok enviada pelo Terminal SIP 1 34
Figura 23. Cenário da transação CANCEL 35
Figura 24. Mensagem CANCEL enviada pelo Terminal SIP 1 35
Figura 25. Resposta ao INVITE enviado após o recebimento da requisição CANCEL 36
Figura 26. Mensagem de resposta à transação CANCEL 37
Figura 27. Cenário da transação OPTIONS 37

Figura 28. Mensagem OPTIONS enviada pelo Terminal SIP 1 38
Figura 29. Mensagem 200 Ok gerada pelo Terminal SIP 2. 39
Figura 30. Mensagem SIP carregando a mensagem SDP como um corpo 42
Figura 31. Comunicação entre as camadas do Terminal SIP 50
Figura 32. Subdivisão da camada de Transação do Usuário 52
Figura 33. Máquina de estados do Agente Cliente do Usuário 53
Figura 34. Máquina de estados do Agente Servidor do Usuário 57
Figura 35. Máquina de estados da Transação Cliente INVITE 61
Figura 36. Máquina de estados da Transação Cliente não-INVITE 62
Figura 37. Máquina de Estados da Camada de Transação Servidora INVITE 64
Figura 38. Máquina de estados da Camada de Transação Servidora não-INVITE 66
Figura 39. Sinalização INVITE no Sistema SIP 72
Figura 40. Interface gráfica do terminal SIP 73
Figura 41. Tela de registro do Terminal SIP 74
Figura 42. Tela para o estabelecimento de uma sessão SIP 74
Figura 43. Tela do encerramento de uma sessão SIP 75
Figura 44. Tela da configuração da placa de som do Terminal SIP 75
Figura 45. Tela com as informações dos formatos suportados pelo Terminal SIP 76
Figura 46. Tela de configuração do formato de áudio e da taxa de amostragem 76
Figura 47. Tela de configuração do tamanho do fragmento 77
Figura 48. Sistema SIP montado no laboratório LARCOM 78
Figura 49. Tela de saída do programa Hello Qt! 112

GLOSSÁRIO DE ABREVIATURAS

ASN.1 – Abstract Syntax Notation One

HTTP – Hypertext Transfer Protocol

IETF – Internet Engineering Task Force

IP – Internet Protocol

ISDN – Integrated Services Digital Network

ITU-T – International Telecommunication Union

MCU – Multipoint Controller Unit

MEGACO – Media Gateway Control Protocol

PSTN – Public Switched Telephone Network

RAS – Registration, Admission and Status

RFC – Request for Comment

RTP – Real-time Transport Protocol

RTSP – Real-time Streaming Protocol

SDP – Session Description Protocol

SIP – Session Initiation Protocol

SPX – Sequential Protocol Exchange

TCP – Transport Control Protocol

TU – Transaciton User

UA – User Agent

UAC – User Agent Client

UAS – User Agent Server

UDP – User Datagram Protocol

URI – Uniform Resource Identifier

UTF - UCS Transformation Formats

VoIP – Voz sobre IP

WAN – Wide Area Network

1. Introdução

Uma parte das informações que estão sendo trocadas pela Internet é devido ao tráfego de voz. Atualmente, as redes de comunicações de voz são formadas principalmente pela rede de telefonia pública e pela rede de serviços integrados (ISDN), ambas utilizando a tecnologia de redes baseadas em chaveamento de circuitos. As redes baseadas em chaveamento de circuitos provêem uma largura de banda fixa e uma latência (atraso) controlada, atendendo aos requisitos do tráfego de voz. Contudo, a capacidade dos circuitos não é compartilhada por outros usuários, o que leva a uma baixa taxa de utilização dos recursos.

Por outro lado, uma rede baseada em pacotes, tal como a rede de IP, envia dados através de pacotes na rede. Os pacotes contêm uma identificação do destino e são roteados independentemente uns dos outros, usando uma técnica conhecida como serviço de melhor esforço (*best-effort*). Este serviço pode não atender aos requisitos do tráfego de voz porque o atraso e a largura de banda não são fixos. Apesar desta restrição, existe uma série de soluções para se usar a rede IP como um meio de transporte de tráfego de voz, pois a rede IP pode fornecer vários benefícios, como o baixo custo e uma maior eficiência no uso da banda disponível.

O transporte de pacotes de voz em uma rede IP é comumente chamado de Voz sobre IP (VoIP) ou telefonia IP.

1.1 Voz sobre IP

Voz sobre IP (VoIP) acontece quando se transporta o sinal de voz digitalizado sobre o protocolo IP. Existem vários cenários em que VoIP pode ser aplicado. A Figura 1 mostra três cenários como exemplo. O primeiro cenário apresenta dois telefones, um interligado à rede de telefonia pública, e outro ligado a um *gateway*. As ligações podem ser iniciadas através de qualquer um dos telefones. As chamadas passam através da rede de telefonia pública e através da Internet. O *gateway* é o terminal de rede que converte uma chamada de VoIP em chamada de rede de telefonia pública. O segundo cenário mostra um telefone e um computador que podem estabelecer uma chamada de voz. No

computador deverá haver um aplicativo de VoIP. O terceiro cenário mostra uma aplicação de VoIP mais comum. Neste cenário, existem dois computadores rodando algum software que estabeleça um canal de comunicação entre eles. Os computadores trocam pacotes de voz entre si através da rede IP – *Internet Protocol* (rede WAN – *Wide Area Network*).

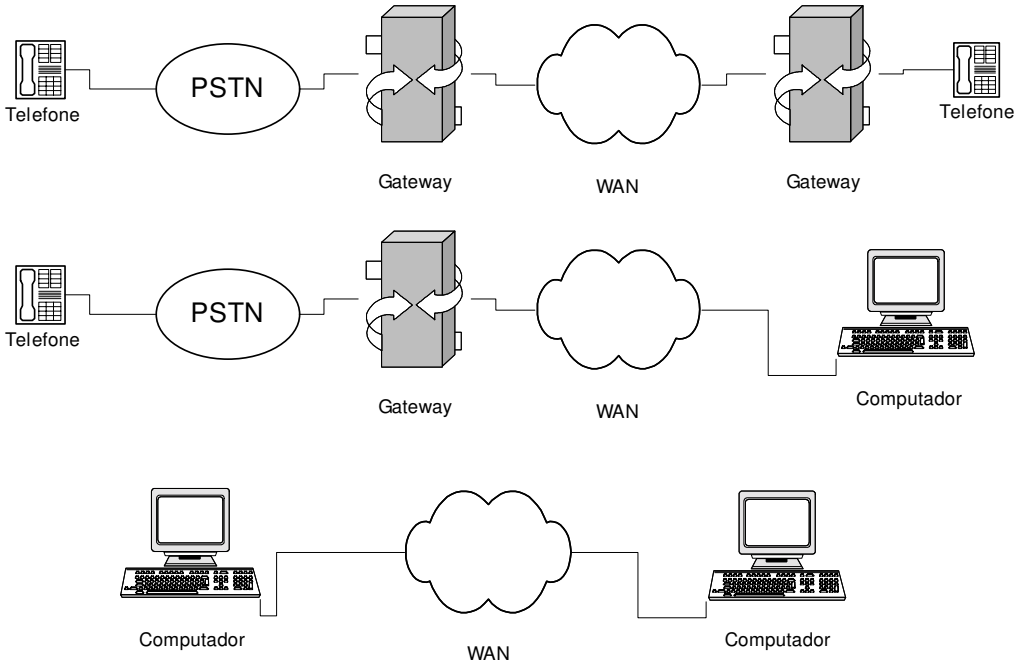

Figura 1. Exemplos de cenários de VoIP

O estabelecimento de uma chamada de VoIP se dá através dos **Protocolos de VoIP**. Uma vez que o usuário deseja estabelecer uma conexão com outro usuário, seja pela discagem de um número no telefone da PSTN, seja por um clique em algum software de computador, uma sinalização é necessária para determinar o *status* da outra parte e para estabelecer a chamada. Os protocolos de VoIP são os que fornecem esta sinalização e gerenciam a chamada. Atualmente, os protocolos de VoIP mais discutidos são o **H.323**[15] e o **SIP[1]**. O H.323 é um protocolo criado pelo ITU-T – *International Telecommunication Union*, que define um conjunto de mensagens para o estabelecimento de canais de voz, vídeo e dados. O protocolo SIP – *Session Initiation Protocol*, foi criado pelo IETF – *Internet Engineering Task Force*, e é um protocolo de sinalização para chamadas de telefonia sobre IP.

1.2 Protocolo H.323

O protocolo H.323, como dito anteriormente, pode negociar o estabelecimento de canais de voz, vídeo e dados sobre uma rede baseada em pacotes. O H.323 é uma pilha de protocolos projetada para operar acima da camada de transporte da rede. Portanto, o H.323 pode ser usado em qualquer rede baseada em pacotes. Um exemplo é a rede baseada na pilha TCP/UDP/IP. O H.323 especifica protocolos para estabelecer uma comunicação em tempo real entre dois terminais em uma rede sem garantias de serviço.

Podemos sumarizar as características gerais da especificação do H.323 em:

- Suporte a conferência ponto-a-ponto e multiponto;
- Interoperabilidade entre redes;
- Suporta diferentes funcionalidades dos terminais clientes;
- Suporta *codecs* de áudio e vídeo;
- Suporta criptografia; e
- Fornece serviços suplementares.

Uma melhor descrição do H.323 será dada no Capítulo 2.

1.3 Protocolo SIP

O SIP é um protocolo de sinalização para o estabelecimento de sessões entre dois ou mais terminais. Um exemplo de sessão é uma chamada de telefonia sobre IP. Diferente do H.323, o protocolo SIP foi projetado para trabalhar com a Internet. Ele explora a gerenciabilidade do protocolo IP e faz o desenvolvimento de uma aplicação de telefonia mais simples do que uma aplicação baseada no protocolo H.323.

O protocolo SIP pode ser empregado para iniciar sessões e convidar membros para participar de sessões já estabelecidas através de outros métodos ou protocolos. O protocolo de sinalização transparentemente suporta mapeamento de nomes e servidores de re-direção. Com isto, a implementação de serviços inteligentes de telefonia se torna mais fácil. Estas facilidades também capacitam ao usuário ter uma mobilidade e uma habilidade de originarem e receberem chamadas. Estas facilidades e outros pontos, que serão destacados na seção 2.5, levaram à escolha do protocolo SIP como o protocolo para o desenvolvimento deste sistema.

Veremos, no Capítulo 3, uma descrição mais detalhada do protocolo SIP.

1.4 Motivações para o Desenvolvimento deste Trabalho

Este trabalho foi motivado por vários fatores. Podemos destacar o interesse em conhecer os protocolos de telefonia IP mais utilizados e os problemas que são encontrados no desenvolvimento de aplicações para a telefonia IP.

Além deste ponto, houve um interesse em conhecer as funcionalidades que o Sistema Operacional Linux oferece para o desenvolvimento de aplicações. Estas facilidades incluem o acesso completo e irrestrito ao *kernel* (núcleo do sistema operacional) e o acesso aos *drivers* dos dispositivos do sistema operacional. Além disto, existia um interesse em desenvolver uma biblioteca SIP que pudesse servir como base para outras aplicações que possam utilizar este protocolo.

1.5 O Sistema SIP Implementado

O Sistema SIP implementado consiste de três entidades: o Terminal SIP, o Servidor de Registros e o Proxy. Todos os três componentes foram implementados para o Sistema Operacional Linux. O Linux foi escolhido, pois ele oferece muitas vantagens sobre o Windows. Pode-se destacar que o Linux é um sistema operacional de baixo custo, às vezes de custo zero. O Linux, também, permite ao desenvolvedor acessar o *kernel* do sistema operacional e os drivers do dispositivo diretamente. Isto é um grande ponto pois facilita o desenvolvimento de tecnologias e possíveis melhoramentos nos drivers dos dispositivos.

O Terminal SIP é a parte do sistema em que o usuário usa das funcionalidades do protocolo SIP para estabelecer uma sessão de voz. As funcionalidades que o Terminal SIP oferece para o usuário são:

- **Registro do usuário:** O Terminal SIP permite que o usuário possa se registrar em um domínio SIP.

- **Estabelecimento e encerramento de sessões de voz:** Através do protocolo SIP, o usuário pode estabelecer e encerrar sessões de voz com outro Terminal SIP.
- **Obtenção das configurações da placa de áudio:** O Terminal SIP possui uma biblioteca de acesso à placa de áudio onde é possível se obter os parâmetros de configuração do dispositivo.
- **Configurações da placa de áudio:** Através da biblioteca de acesso à placa de áudio, o Terminal SIP pode configurar os parâmetros do *driver* do dispositivo.

O Servidor de Registros e o Proxy são as outras duas entidades que compõem o Sistema SIP, e eles foram implementados em um único dispositivo para facilitar o acesso do Proxy às tabelas de registro do Servidor de Registro. As suas funções principais são:

- **Registro dos usuários no domínio SIP**: O Servidor de Registros efetua o registro dos Terminais SIP em sua base de dados.
- **Localização dos usuários na rede**: Através da base de dados do Servidor de Registros, o Proxy pode determinar o endereço IP do terminal de destino da mensagem.
- **Roteamento das mensagens SIP**: Após a consulta no Servidor de Registros, o Proxy escolhe por onde rotear a mensagem. Se o usuário está registrado no seu domínio, o Proxy envia a mensagem diretamente para o usuário. Se o usuário não está registrado no domínio, o Proxy envia a mensagem para outro Proxy.

1.6 Organização do Trabalho

Este trabalho está disposto da seguinte forma:

- **Capítulo 1:** O Capítulo 1 dá uma breve introdução sobre a tecnologia de Voz sobre IP e apresenta um resumo sobre o Sistema SIP desenvolvido.
- **Capítulo 2:** O Capítulo 2 dá uma breve descrição do protocolo H.323, descrevendo as suas características principais, sua sinalização e uma pequena comparação com o protocolo SIP.

- **Capítulo 3:** O Capítulo 3 descreve o protocolo SIP. São descritas as funcionalidades do protocolo, seus componentes, suas mensagens e a sua sinalização.
- **Capítulo 4:** O Capítulo 4 descreve o protocolo SDP e as suas funcionalidades.
- **Capítulo 5:** O Capítulo 5 dá uma descrição funcional do Sistema SIP implementado.
- **Capítulo 6:** O Capítulo 6 descreve os testes e resultados obtidos com o sistema.
- **Capítulo 7:** O Capítulo 7 descreve as conclusões deste trabalho.

Além destes capítulos, o trabalho ainda traz a Bibliografia utilizada e os seguintes apêndices:

- **Apêndice I:** Traz uma descrição de todos os campos da mensagem SIP padronizados pela RFC 3261[1].
- **Apêndice II:** Traz uma descrição de todos os campos da mensagem SDP padronizados pela RFC 2327[2].
- **Apêndice III:** Descreve resumidamente a Biblioteca QT[25].
- **Apêndice IV:** Descreve os detalhes da implementação do Sistema SIP.

2. Protocolo H.323

2.1 Introdução

O Protocolo H.323 é um conjunto de mensagens definidas pelas recomendações do ITU-T, *International Telecommunications Union – Telecommunication Standardization Sector*. Destas recomendações, a mais importante é a H.323 – *Packet-based Multimedia Communications Systems*[15]. As outras recomendações que compõem o protocolo são: H.245 – *Control Protocol for Multimedia Communication*[17], o H.225.0 – *Call Signalling Protocols and Media Stream Packetization for Packet-based Multimedia Communication Systems*[16], e Q.931 - *ISDN User-network Interface Layer 3 Specification for Basic Call Control*[18].

O protocolo H.323 pode ser usado no topo de qualquer rede baseada em pacotes, pois ele não depende do protocolo de transporte. Comumente, os protocolos de transporte TCP[8] e UDP[6], e o protocolo IP são usados para o transporte do protocolo e dos fluxos de mídia associados à família de protocolos H.323.

Neste capítulo será descrito o protocolo H.323, os seus componentes, padrões e a operação do protocolo em um cenário simples.

2.2 Recomendação H.323

A recomendação H.323 surgiu em sua primeira versão em 1996. Depois, o ITU-T lançou as versões de 1997, 1998, 1999, 2000 e 2001 que é, atualmente, a última versão desta recomendação. O H.323 especifica os requerimentos técnicos para sistemas de comunicações multimídia em situações aonde a rede de transporte é uma rede baseada em pacotes. A recomendação descreve, também, os componentes de um sistema H.323. Este sistema inclui terminais, *gateways*, *gatekeepers* e Unidades de Controle Multiponto. Estes componentes serão logo a seguir.

As características principais da recomendação H.323 podem ser resumidas em:

- Suporte a conferência ponto-a-ponto e ponto-multiponto;
- Estabelecimento de canais entre dois terminais de capacidades heterogêneas;

- Padronização de *codecs* de áudio e vídeo.

2.3 Componentes de um Sistema H.323

2.3.1 Terminais H.323

Um terminal H.323 é um terminal na rede que estabelece um canal de comunicação bidirecional, e em tempo real, com outro terminal H.323, com um *gatekeeper*, com um *gateway* ou com uma unidade de controle multiponto. A comunicação entre dois terminais consiste de mensagens de controle, fluxos multimídia e dados. A recomendação não obriga que todos os terminais possuam capacidades de troca de dados e multimídia; a exigência mínima é que os terminais tenham a capacidade de estabelecer canais de áudio.

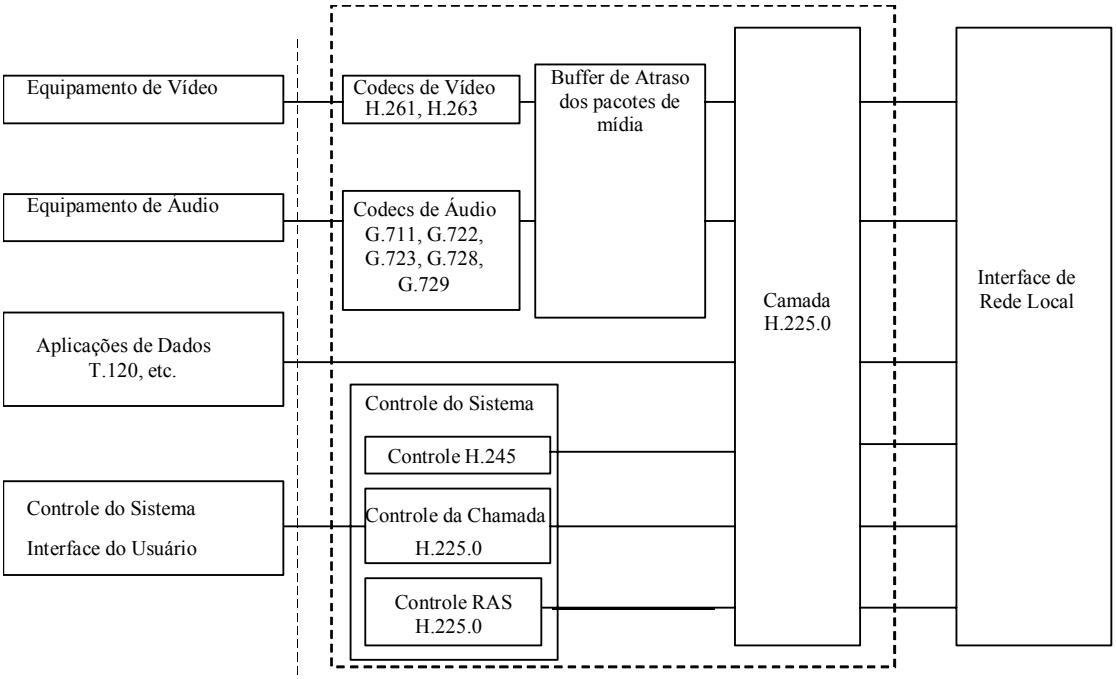

Figura 2. Terminal H.323

A Figura 2 mostra as possíveis funcionalidades de um terminal H.323. Dentre elas, as seguintes funcionalidades são obrigatórias:

- Unidade de Controle do Sistema,
- Camada H.225.0,

- Interface de rede, e
- Unidade dos *Codecs* de Áudio.

As outras funcionalidades são opcionais.

2.3.1.1 Interface de Rede Local

A interface de rede compreende os protocolos de comunicação, como o TCP – *Transport Control Protocol*, SPX - *Sequential Protocol Exchange*, UDP - *User Datagram Protocol*, e outros. A recomendação não dita a implementação da interface. Ela apenas requer que estes protocolos atendam certas exigências da sinalização. Por exemplo, para o canal de controle H.245 aberto entre dois terminais H.323, exige-se confiabilidade na troca de mensagens, obrigando, assim, a utilização de um protocolo com tal característica, como o TCP.

2.3.1.2 Unidade dos Codecs de Áudio

Esta é a parte do terminal de áudio onde estão contidos os *codecs* de áudio que serão usados na comunicação. Todos os terminais H.323 devem possuir um *codec* de áudio, que é o G.711[19]. Os outros *codecs* são opcionais dependendo da capacidade da rede. Por exemplo, o G.711 não é aplicável em redes que usam o modem de 56 kbps como o seu canal de acesso à Internet. A recomendação estabelece que deve ser usado um outro *codec*, como o G.729[23], por exemplo.

2.3.1.3 Unidade de Controle do Sistema

Esta unidade é dividida em três subunidades: Controle H.245, Controle da Chamada H.225.0 e Controle RAS H.225.0.

O Controle H.245 usa o canal de controle H.245, aberto na negociação H.323, para enviar mensagens de controle que governam a operação do terminal H.323. Estas operações incluem trocas de capacidades, abertura e fechamento de canais lógicos,

requisição dos modos de preferência, entre outros. Esta subcamada usa mensagens e procedimentos descritos na recomendação H.245.

O Controle RAS H.225.0 usa mensagens H.225.0 para realizar registros, admissões e procedimentos de desconexão entre terminais e *gatekeepers*. O Controle da Chamada H.225.0 usa a sinalização de chamada H.225.0 para estabelecer uma conexão entre dois terminais H.323. Este é o primeiro canal de sinalização aberto entre dois terminais H.323.

2.3.2 Gateway

O *gateway* é um dispositivo de adaptação utilizado para permitir a comunicação entre terminais H.323 e terminais não H.323. A sua principal função é a translação entre formatos de transmissão (por exemplo: H.225 para H.221), procedimentos de comunicação (por exemplo: H.245 para H.242), e formatos de áudio, vídeo e dados. O *gateway* executa, também, funções de estabelecimento e desconexão de chamadas do lado da rede baseada em pacotes e da rede com comutação de circuitos. Em geral, o objetivo do *gateway* é refletir as características de um terminal da rede local para um outro terminal da rede comutada por circuitos, e vice-versa.

2.3.3 Gatekeeper

O *gatekeeper* provê serviços de controle de chamada para os terminais H.323. Ele é opcional em uma rede. Quando ele está presente, ele provê os seguintes serviços:

- **Tradução de Endereços:** O *gatekeeper* permite o uso local de esquemas de endereçamento proprietários (chamados de apelidos H.323), tais como mnemônicos, *nicknames*, ou endereços de e-mail. O *Gatekeeper* irá traduzir estes endereços em endereços IP necessários para o estabelecimento das comunicações H.323. Este processo pode ser feito através de uma tabela de translação, que é atualizada através de mensagens de registro.
- **Controle de Admissões:** O *gatekeeper* é responsável por controlar o estabelecimento de chamadas entre terminais H.323, *gateways*, e outros

dispositivos não-H.323 (através do *gateway*). A admissão de uma chamada pode ser autorizada ou negada pelo *gatekeeper* baseada em procedimentos de autenticação do usuário, endereços de fonte ou destino, hora do dia, largura de banda disponível, ou qualquer outro critério. O controle da admissão pode, também, ser uma função nula, onde todos os pedidos seriam aceitos.

- **Controle da Largura de Banda:** O *gatekeeper* pode controlar o número de terminais H.323 com acesso simultâneo à rede. Através da sinalização H.225.0, o *gatekeeper* pode rejeitar chamadas de um terminal devido a limitações de largura de banda. Esta função pode, também, operar durante uma chamada ativa se um terminal solicitar banda adicional. O critério usado para determinar se a largura de banda disponível é suficiente para atender uma chamada entrante não é definido pela recomendação H.323.
- **Gerenciamento de Zona de Atuação do Gatekeeper:** O *gatekeeper* pode coordenar as ações associadas às funções descritas acima entre os dispositivos que fazem parte de sua zona de influência (dispositivos que estão sob influência do *gatekeeper*).
- **Sinalização de Chamada:** O *gatekeeper* pode atuar como um *proxy* de sinalização de chamada entre os terminais ou *gateways* sob sua responsabilidade, aliviando-os da necessidade de suportar o protocolo de controle da chamada. De outra forma, o *gatekeeper* pode, simplesmente, servir como um ponto de contato inicial, ou seja, após a admissão da chamada proposta, o *gatekeeper* interconecta os dois terminais para trocar mensagens de sinalização diretamente.

2.3.4 Unidade Controle Multiponto

A unidade de controle multiponto (MCU – *Multipoint Controller Unit*) suporta conferências entre três ou mais terminais e *Gateways*.

Um MCU típico consiste de um controlador multiponto e um processador multiponto de áudio, vídeo e dados. O processador multiponto suporta a negociação de capacidades entre todos os terminais para garantir um nível comum de comunicações. O

controlador multiponto envia um conjunto de capacidades aos participantes da conferência, indicando os modos de operação em que eles devem transmitir. Este conjunto de capacidades pode ser revisto pelo controlador multiponto e reenviado aos terminais, em função da adesão de novos terminais ou desistência de membros da conferência. O processador multiponto é o processador central de voz, vídeo e dados para uma conferência multiponto.

2.4 Sinalização H.323

A sinalização H.323 envolve vários cenários diferentes. Será dado um exemplo de sinalização entre dois terminais diretamente. O cenário é mostrado na Figura 3.

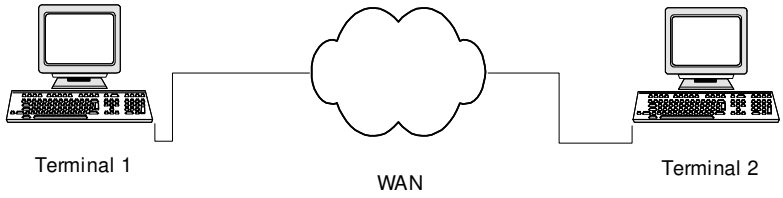

Figura 3. Cenário da sinalização H.323

A sinalização H.323 envolve cinco fases de sinalização:

- Fase A: Inicialização da Chamada;
- Fase B: Comunicação Inicial e Troca de Funcionalidades;
- Fase C: Estabelecimento da Comunicação Audiovisual;
- Fase D: Serviços da Chamada; e
- Fase E: Finalização da Chamada.

Cada fase será descrita nos itens a seguir.

2.4.1 Fase A: Inicialização da Chamada

A inicialização da chamada ocorre através da troca de mensagens de controle de chamada definidas na recomendação H.225.0. Na Figura 3, nenhum dos terminais está registrado em algum *gatekeeper*. Então, o terminal 1 deverá conhecer o endereço IP do terminal 2 e enviar uma mensagem diretamente para ele. A primeira mensagem enviada

pelo terminal 1 é a mensagem **Setup**. Esta mensagem indica, para o terminal 2 que o terminal 1 deseja estabelecer uma conexão com ele. Nesta mensagem, são descritas as informações básicas sobre a conexão a ser estabelecida, como o tipo de sessão, o endereço no qual a mídia deverá ser trocada e a versão do protocolo que o terminal 1 está utilizando. Quando o terminal 2 recebe a mensagem, se ele não puder respondê-la rapidamente, ele envia uma mensagem **Call Proceeding**, indicando que o terminal 2 recebeu a mensagem e a está processando. Quando o terminal 2 termina o processamento da mensagem, ele envia uma resposta; se ele aceitar o pedido de conexão, ele responde com a mensagem **Connect**. Se ele não desejar aceitar a conexão, ele responde com um **Reject**. A troca de mensagens é mostrada na Figura 4.

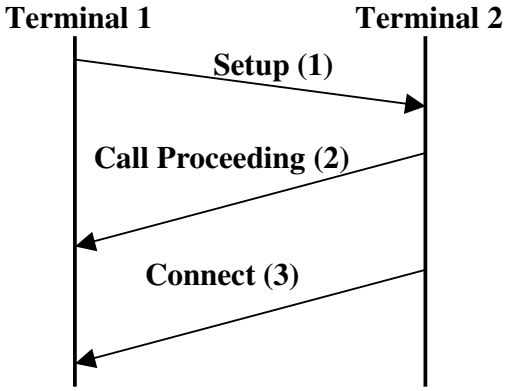

Figura 4. Troca de mensagens da Fase A

Após o recebimento da mensagem **Connect** pelo terminal 1, a fase A é encerrada e o terminal 1 passa para a fase B.

2.4.2 Fase B: Comunicação Inicial e Troca de Funcionalidades

Nesta fase, os terminais devem estabelecer o canal de controle H.245, referido na recomendação H.323 como canal de controle 0, e trocar as suas funcionalidades. A comunicação inicial é a troca de mensagens para o estabelecimento do canal de controle H.245.

Para estabelecer o canal de controle, o terminal 1 envia a mensagem **OpenLogicalChannel**. Esta mensagem descreve totalmente o canal lógico a ser aberto. Por exemplo, a mensagem inclui a descrição do dado a ser trocado, como áudio a uma

taxa de 64kbps. No caso do canal de controle, a mensagem descreve o tipo de dados a ser trocado e dá uma faixa de transmissão de dados, especificando as taxas máximas e mínimas de bits que o terminal 1 pode operar. Quando o terminal 2 receber esta mensagem, ele a processa, verificando se é compatível com os requerimentos da mensagem recebida. Se ele puder operar nas especificações da mensagem, ele responde com a mensagem **OpenLogicalChannelAck**. Se ele não aceitar, responde com um **OpenLogicalChannelReject**. No caso de um recebimento de um **OpenLogicalChannelReject**, o terminal 1 pode tentar mais duas vezes o estabelecimento, alterando os parâmetros do canal. Se estas duas novas tentativas falharem, o terminal 1 aborta a tentativa de estabelecimento da chamada. Esta sinalização é mostrada na Figura 5.

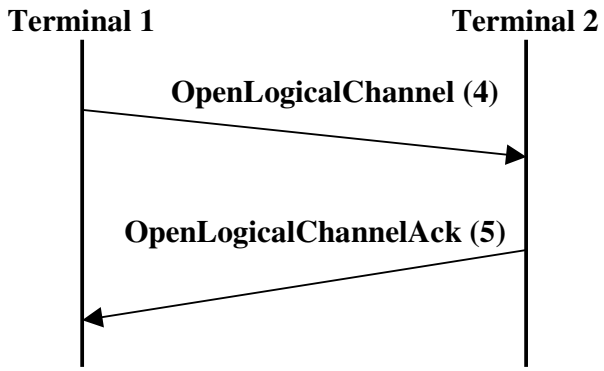

Figura 5. Troca de mensagens de abertura do canal de controle H.245.

Após o estabelecimento do canal de controle, os terminais o usarão para a troca de funcionalidades. O terminal 1 gera a mensagem **TerminalCapabilitySet**, que indica os modos de recebimento e transmissão de dados e mídia. Estes modos compreendem de um agrupamento de descritores de funcionalidades. Cada descritor indica um modo de operação do terminal. Por exemplo, o terminal é capaz de enviar e receber áudio com o *codec* G.723.1[21] e enviar e receber vídeo com o *codec* H.261, ou o terminal envia pacotes de áudio com o *codec* G.723.1 e recebe com o *codec* G.729. O terminal 1, então, envia a mensagem para o terminal 2 através do canal de controle H.245. Quando o terminal 2 recebe a mensagem, ele analisa cada descritor e verifica quais ele pode trabalhar. Se o terminal 2 não desejar, por falta de banda disponível, ou não puder

trabalhar com algum conjunto de descritores, ele envia uma mensagem **TerminalCapabilityReject**, contendo a razão pela rejeição da troca de capacidades. Se o terminal aceitar, ele responde com um **TerminalCapabiliyAck**, indicando que ele aceitou o conjunto de funcionalidades do terminal 1. A negociação é mostrada na Figura 6.

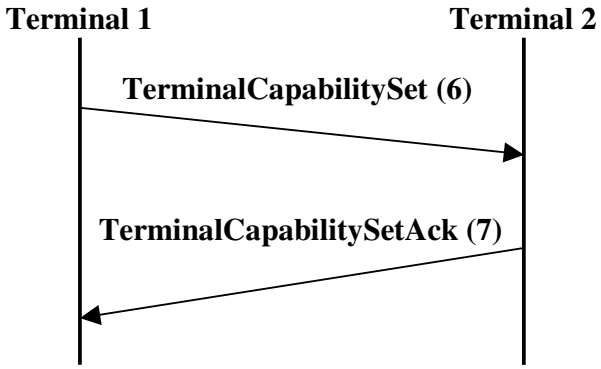

Figura 6. Troca de mensagens das funcionalidades do terminal

2.4.3 Fase C: Estabelecimento da Comunicação Audiovisual

Após a troca de funcionalidades, os terminais devem abrir os canais lógicos para os vários fluxos de mídia. Neste exemplo, os dois terminais vão estabelecer um canal de áudio somente. Os canais de áudio podem ser bidirecionais ou unidirecionais. No caso de canais bidirecionais, o terminal 1 realiza a mesma negociação usada na abertura do canal de controle H.245, para a abertura do canal de áudio. Ele envia uma mensagem **OpenLogicalChannel** e espera o recebimento de uma mensagem **OpenLogicalChannelAck** ou **OpenLogicalChannelReject**. Após a confirmação do estabelecimento dos canais, cada terminal poderá iniciar o fluxo de áudio. A Figura 7 mostra a sinalização.

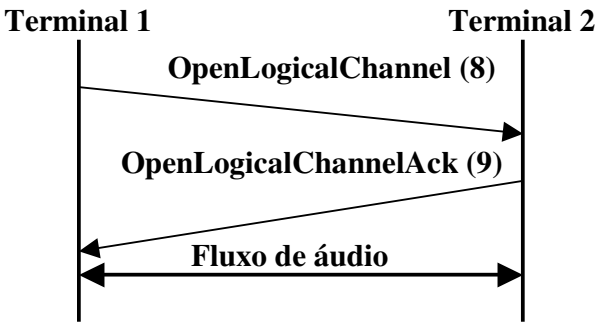

Figura 7. Troca de Mensagens da Fase C – Estabelecimento dos canais de áudio bidirecionais

No caso de abertura de canais unidirecionais, cada terminal deverá fazer a sua negociação de abertura de canal lógico para abrir o seu canal de áudio. Esta negociação é mostrada na Figura 8.

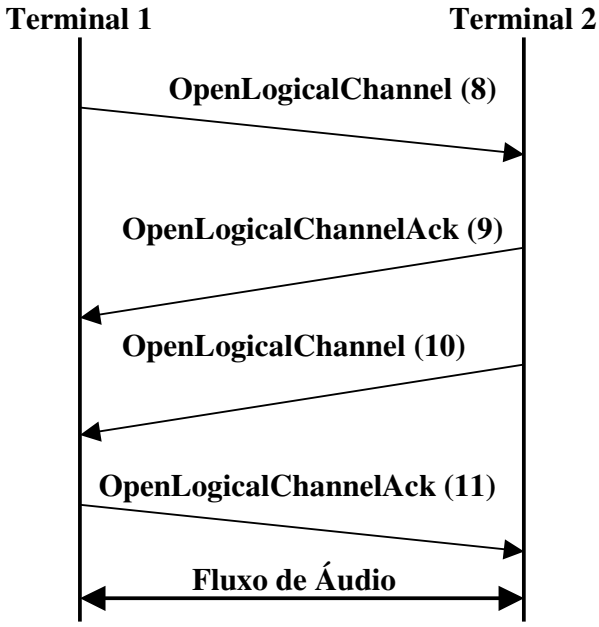

Figura 8. Sinalização para o estabelecimento dos canais de áudio unidirecionais

2.4.4 Fase D: Serviços da Chamada

Nesta fase, os terminais já estabeleceram os canais de áudio e estão com uma chamada em andamento. Esta fase é usada quando um dos terminais deseja alterar algum parâmetro da chamada, como, por exemplo, o *codec* em uso. Isto é acompanhado pela distribuição de um **CloseLogicalChannel**, indicando o fechamento do canal de áudio aberto. Após o recebimento da resposta ao **CloseLogicalChannel**, o terminal envia um **OpenLogicalChannel**, descrevendo o novo canal a ser aberto.

2.4.5 Fase E: Finalização da Chamada

Nesta fase, a última fase, um dos terminais deseja terminar a chamada. Então, o terminal deverá encerrar o fluxo de áudio e fechar os canais lógicos de áudio abertos, enviando a mensagem **CloseLogicalChannel**. O outro terminal responde com um **CloseLogicalChannelAck**. Então, o terminal iniciante do término da chamada envia a

mensagem **endSessionCommand**, indicando o fim da chamada, o fim da troca de mensagens de controle H.245 e o fechamento do canal de controle H.245. O outro terminal responde com um **endSessionCommand** também, indicando que recebeu a mensagem e que fechará o canal de controle H.245. Quando o terminal iniciante da finalização recebe esta mensagem, ele envia a mensagem **Release Complete** para o outro terminal, indicando o fechamento de qualquer outro canal aberto com o terminal iniciante da chamada. A Figura 9 mostra esta sinalização.

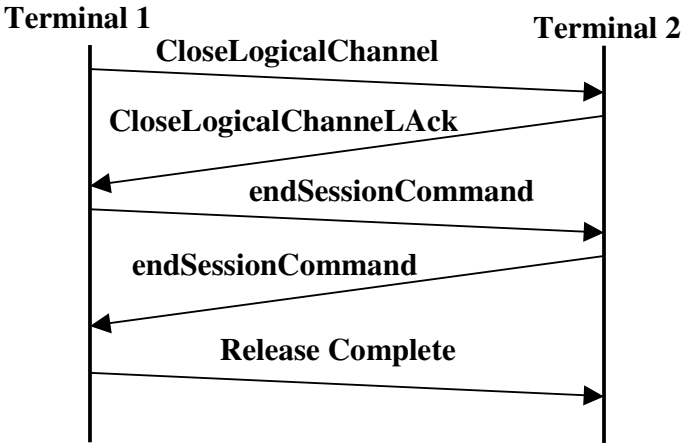

Figura 9. Finalização da chamada H.323

Após esta mensagem, a chamada está finalizada.

2.5 Considerações Finais sobre o Protocolo H.323

O protocolo H.323 está competindo com o protocolo SIP para se tornar o padrão de sinalização da telefonia IP. Cada protocolo tem as suas características, mas ambos estabelecem um canal de comunicação multimídia entre dois ou mais terminais.

O protocolo H.323 é mais complexo que o protocolo SIP. Pode-se, comparar, por exemplo, a especificação SIP com a H.323. O protocolo SIP é baseado em uma única especificação. Já o H.323 é baseado em várias normas do ITU-T. Outro ponto que se pode destacar são as mensagens SIP com relação às mensagens H.323. As mensagens SIP possuem poucos campos, enquanto que as mensagens H.323 possuem centenas de campos, onde cada campo pode possuir vários subcampos.

O protocolo H.323 usa uma representação binária para suas mensagens, baseada no padrão ASN.1, enquanto que as mensagens SIP são mensagens de texto, codificadas em UTF-8[5], como as mensagens do HTTP[13].

O protocolo H.323 também possui alguns problemas quando a questão é a extensibilidade. Os *codecs* de áudio e vídeo usados com o protocolo H.323 devem estar padronizados pelo ITU-T para serem transportados pelas mensagens H.323. Isto é uma barreira para os vários desenvolvimentos que estão ocorrendo nesta área. O protocolo SIP, através do protocolo SDP, possui mecanismos de extensibilidade que suportam a inserção de novos *codecs* de áudio e vídeo à medida que eles estejam sendo desenvolvidos.

Apesar destes pontos, o H.323 tem a vantagem de suportar vários esquemas de endereçamento, através dos *gatekeepers* e *gateways*. Já o SIP trabalha com endereços da forma username@domínio_sip, apesar de suportar algumas outras extensões.

Por causa destas desvantagens, o protocolo SIP tem ganhado a atenção dos desenvolvedores. A simplicidade da sinalização, em relação ao H.323, a grande interação com os protocolos existentes do HTTP e a facilidade de se implementar o protocolo SIP junto com os servidores de e-mail, fazem do SIP uma boa escolha para o desenvolvimento aplicações de Telefonia IP. Foi por isso que desenvolvemos este trabalho usando o Protocolo SIP.

3. SIP: Session Initiation Protocol

3.1 Introdução

O protocolo SIP, definido pelo IETF, é um protocolo de sinalização para chamadas de telefonia sobre IP. Diferente do H.323, o protocolo SIP foi idealizado especificamente para a Internet. O SIP é um protocolo da camada de aplicação que permite a criação, modificação e estabelecimento de sessões com um ou mais participantes. O protocolo pode convidar participantes para iniciar uma sessão ou fazer parte de uma sessão já existente, que pode ter sido criada através de outros protocolos, como o H.323. O protocolo SIP suporta o mapeamento de nomes e serviços de re-direção. Estas facilidades capacitam o usuário a ter uma mobilidade entre terminais na rede.

Neste capítulo, será descrito o protocolo SIP. Inicialmente, serão descritas as características principais do protocolo SIP, isto é, as suas funcionalidades e os protocolos que trabalham em conjunto para estabelecer uma sessão. Na seqüência, serão apresentados a estrutura do protocolo, as suas mensagens e o seu funcionamento.

3.2 Protocolo SIP

O SIP é um protocolo da camada de aplicação que pode estabelecer, modificar e finalizar uma sessão multimídia entre duas ou mais partes. Ele foi criado pelo IETF – *Internet Engineering Task Force*, em 1999, através da RFC 2543 – *SIP: Session Initiation Protocol*. Esta RFC já foi revisada, originando a RFC 3261[1]. O protocolo é baseado em um modelo cliente/servidor do http[13] – *Hipertext Transfer Protocol*. A entidade que gera uma requisição é a entidade cliente, ou, simplesmente, cliente, e a entidade que responde a esta requisição é a entidade servidora, ou, simplesmente, servidor.

O protocolo SIP é um protocolo simples que possui as mesmas funcionalidades básicas que o protocolo H.323 possui, como a criação de um canal multimídia entre duas ou mais partes. Estas funcionalidades básicas são cinco:

- **Localização do Usuário:** É a localização do outro participante para o estabelecimento de uma sessão SIP;

- **Disponibilidade do Usuário:** É a determinação da disponibilidade do outro participante da sessão em se juntar na comunicação;
- **Capacidades do Usuário:** É a determinação dos parâmetros da mídia a ser trocada entre as duas ou mais partes envolvidas na sessão;
- **Estabelecimento/Finalização da Sessão:** É a capacidade de se estabelecer e finalizar uma sessão através da troca de mensagens SIP; e
- **Gerenciamento da Sessão:** É a capacidade de gerenciar uma sessão estabelecida entre as partes envolvidas. Este gerenciamento é feito através da troca de mensagens entre as partes.

Além destas funcionalidades, o protocolo SIP pode ser usado com outros protocolos IETF para prover outros serviços multimídia. O protocolo SIP é usado, geralmente, com o protocolo SDP – *Session Description Protocol*[2], para descrever os parâmetros da mídia a ser trocada. Além do protocolo SDP, o SIP pode ser usado com o protocolo RTP – *Real-time Transport Protocol*[10], para transportar dados em tempo real, RTSP – *Real-time Streaming Protocol*[11], para controlar a entrega dos pacotes da mídia e o MEGACO – *Media Gateway Control Protocol*[12], para controlar *gateways*. Contudo, a operação SIP é independente destes protocolos.

O protocolo SIP sozinho não provê serviços. Ele provê primitivas que podem ser usadas para implementar diferentes tipos de serviços. Por exemplo, o SIP pode localizar um usuário e entregá-lo um objeto. Este objeto pode ser uma mensagem SDP para descrever um canal multimídia. Portanto, o protocolo SIP não trabalha sozinho para estabelecer um canal de comunicação multimídia.

3.3 Estrutura do Protocolo SIP

O protocolo SIP é estruturado em camadas. Estas camadas trabalham em conjunto para prover as funcionalidades SIP. O protocolo é subdividido em três camadas básicas: Camada de Transporte (*Transport Layer*), Camada de Transação (*Transaction Layer*) e Transação do Usuário (*Transaction User – TU)*. A comunicação básica entre as camadas é mostrada na Figura 10.

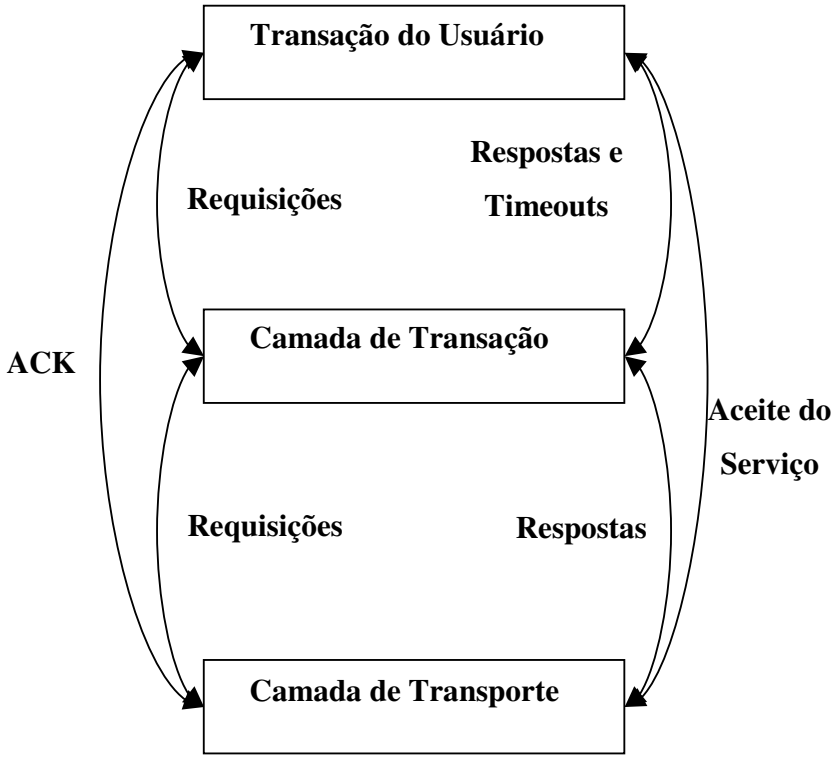

Figura 10. Comunicação básica entre as camadas do protocolo SIP

A camada inferior do protocolo SIP é a camada de transporte. Ela define como o cliente envia as requisições e recebe as respostas e como o servidor recebe as requisições e envia as respostas através de uma rede. Esta camada é que contém os protocolos de rede, como a pilha TCP/IP.

A segunda camada é a camada de transação. Esta camada é fundamental para o protocolo SIP, pois é ela quem gerencia as transações das mensagens SIP. A camada de transação gerencia as retransmissões das mensagens, verifica as respostas recebidas com a requisição enviada e gerencia o estouro do tempo da transação. Qualquer requisição, que a TU quiser enviar, é acompanhada por uma transação. A camada de transação é ainda subdividida em quatro subcamadas: Transação Cliente *Invite*, Transação Cliente não-*Invite*, Transação Servidora *Invite* e Transação Servidora não-*Invite*. Estas subcamadas serão explicadas na seção 5.2.2.

A camada superior do protocolo SIP é a transação do usuário. Ela é a responsável pela interação com o usuário. Esta camada é ainda subdividida em duas subcamadas: Agente Cliente do Usuário – *User Agent Client* e Agente Servidor do Usuário – *User*

Agent Server. O agente cliente do usuário é responsável por gerar requisições e receber as respostas. O agente servidor do usuário é responsável por receber as requisições e respondê-las.

3.4 Mensagens SIP

O Protocolo SIP usa mensagens textuais, codificadas em UTF-8 (*UCS Transformation Formats)*, para fornecer a sua sinalização. Estas mensagens são subdivididas em duas classes:

- **Mensagens de Requisição**: Estas mensagens requisitam alguma ação do destinatário, e esta ação requisitada é chamada de método; e
- **Mensagens de Resposta**: As mensagens de resposta indicam o resultado do processamento da requisição, isto é, indicam a ação tomada pelo destinatário da requisição.

Apesar da classificação das mensagens SIP em dois grupos, elas têm um formato geral. As mensagens SIP contêm uma linha inicial, alguns campos e, às vezes, um corpo. O formato da mensagem genérica é mostrado na Figura 11.

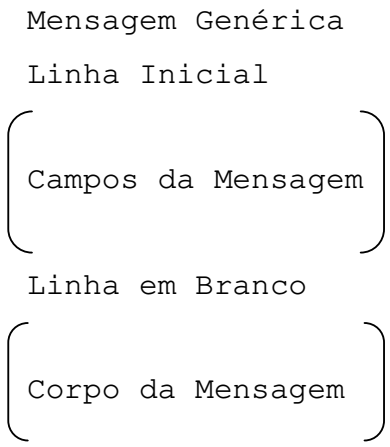

Figura 11. Formato das mensagens SIP

O que diferencia se a mensagem é uma requisição ou uma resposta é a Linha Inicial. As mensagens SIP são compostas de Campos. Estes campos são atributos que

contêm informações sobre a mensagem e sobre quem envia a mensagem. O formato geral de cada campo é:

```
Nome-do-Campo: Valor-do-Campo;parâmetro=valor-parâmetro
```

O nome do campo e o seu valor definem o campo. O parâmetro é alguma informação adicional relativo ao campo. Um exemplo de campo é dado abaixo:

```
To: meireang@hamal.mc21.fee.unicamp.br;tag=0c9enbieyVn
```

Os campos definidos pela RFC – *Request for Comments*, estão descritos no Apêndice I.

O corpo da mensagem é construído de acordo com a especificação do protocolo a ser anexado na mensagem SIP. Por exemplo, quando o corpo da mensagem for relativo ao protocolo SDP, o formato da mensagem deve estar de acordo com a RFC 2327.

3.4.1 Mensagens de Requisição

As mensagens de requisição são distintas das mensagens de resposta pela linha inicial. A linha inicial das mensagens de requisição é chamada de linha de requisição. A linha de requisição tem o seguinte formato:

```
Método URI-Destino Versão-do-Protocolo
```

- **Método:** Atualmente existem seis métodos definidos na RFC 3261: **INVITE** e **ACK**, para estabelecer uma sessão; **BYE**, para finalizar uma sessão; **OPTIONS**, para obter as capacidades do servidor; **REGISTER**, para registrar o terminal SIP em um servidor de registros; e **CANCEL**, para cancelar uma tentativa de estabelecimento de sessão.
- **URI-Destino:** Um URI – *Uniform Resource Identifier*[4], é um identificador de um terminal. Ele indica o usuário ou serviço para o qual esta requisição está sendo enviada. A forma geral de um URI em SIP é *username@domínio_sip*. Somente na mensagem REGISTER, o URI é o endereço IP do servidor de registros.
- **Versão-do-Protocolo:** Ambas as requisições e respostas possuem a versão do protocolo na primeira linha da mensagem. A versão do protocolo mais recente é a 2.0. Portanto, o valor deste campo é SIP/2.0.

Um exemplo da linha inicial de uma requisição é:

```
INVITE sip:davison@hamal.mc21.fee.unicamp.br SIP/2.0
```

3.4.2 Mensagens de Resposta

As mensagens de resposta são diferentes das mensagens de requisição por causa da linha inicial. A linha inicial, nas mensagens de resposta, é chamada linha de *status*. Isto se deve ao fato de que as respostas se referem a um *status* da requisição enviada para um servidor. A linha de *status* consiste de um código de status numérico, uma frase textual associada a este código, e a versão do protocolo SIP.

```
Versão-do-Protocolo Código-de-Status Frase
```

O código de *status* é um número de três dígitos que indica o resultado de uma tentativa de um servidor de entender e processar a requisição. A frase textual é uma descrição curta do código de *status*. O primeiro dígito do código de status define a classe da resposta. Os dois últimos dígitos não têm qualquer regra de categorização. A classificação das respostas é dada por:

- **1xx:** Respostas Provisionais – Esta classe de resposta indica que o servidor recebeu a requisição e a está processando, mas ele não pode responder a requisição imediatamente. Portanto, esta resposta indica para o cliente que ele deve aguardar para que o servidor possa responder a requisição enviada.
- **2xx:** Respostas de Sucesso – Esta classe de resposta indica que o servidor recebeu a requisição, processou e aceitou a requisição.
- **3xx:** Respostas de Redireção – Esta classe indica que a requisição deve ser redirecionada para outro endereço, contido na resposta.
- **4xx:** Respostas de Erro de Cliente – Esta classe de resposta indica que o servidor recebeu a requisição, mas o servidor não entendeu algum campo essencial da mensagem ou algum campo da mensagem não está bem formada.
- **5xx:** Respostas de Erro no Servidor – Esta classe indica que houve alguma falha no processamento da resposta pelo servidor e ele não pode aceitar a requisição.

- **6xx:** Respostas de Erro Global – Esta classe indica erros que podem ocorrer devido a algum requisito da mensagem que o servidor não suporte.

Um exemplo da linha inicial de mensagem de resposta é:

```
SIP/2.0 200 Ok
```

3.5 Entidades SIP

Existem basicamente três entidades SIP: o terminal SIP, o proxy e o servidor de registros.

O terminal SIP é aquele em que um usuário tem acesso aos serviços do protocolo SIP para estabelecer uma sessão. O terminal SIP pode ser um computador ou um telefone SIP.

O proxy é um elemento que é subdividido em dois: proxy com estado e proxy sem estado. Um proxy com estado é aquele que quando recebe uma mensagem, se comporta como se ele mesmo estivesse mandando aquela mensagem. O proxy sem estado é simplesmente um enviador de mensagens. Ele somente verifica se a mensagem é para ele ou para o servidor de registros do domínio daquele proxy e reenvia a mensagem. Ele não se comporta como se fosse o requisitante.

O servidor de registros é a entidade SIP responsável por registrar os endereços SIP e os endereços reais dos usuários do protocolo SIP. Através do seu arquivo de registros, um proxy pode solicitar uma pesquisa para determinar se um determinado contato está registrado naquele servidor.

Todas as entidades SIP, com exceção do proxy sem estado, possuem todas as camadas do protocolo. O proxy sem estado possui somente a camada de transporte e um núcleo de processamento de mensagens do proxy.

3.6 Visão Geral do Funcionamento do Protocolo

Esta seção irá descrever o funcionamento do protocolo SIP. Serão descritos o registro do terminal, um estabelecimento de uma sessão SIP, um encerramento de sessão,

um cancelamento de sessão e o levantamento de capacidades através da mensagem OPTIONS.

3.6.1 Registrando um Terminal SIP

Quando o terminal SIP inicia, ele deve efetuar o registro dos seus endereços IP e SIP em um servidor de registros. A transação é exemplificada na Figura 12.

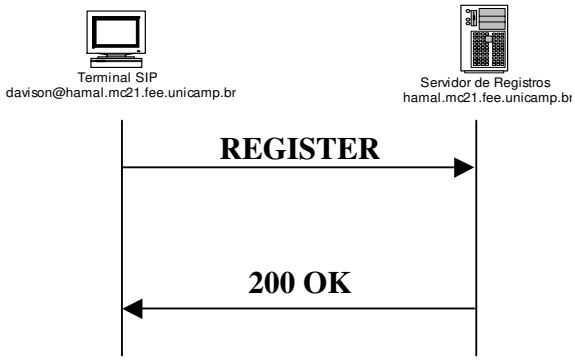

Figura 12. Cenário da transação REGISTER

Na Figura 12, o terminal SIP deseja registrar o seu endereço SIP, davison@hamal.mc21.fee.unicamp.br, no servidor de registros do domínio. O endereço IP do Terminal SIP é regulus.mc21.fee.unicamp.br. Portanto, o Terminal SIP gera uma requisição REGISTER. Ela deverá ser conforme a Figura 13.

```
REGISTER sip:hamal.mc21.fee.unicamp.br SIP/2.0
Via: SIP/2.0/UDP regulus.mc21.fee.unicamp.br:5060;branch=z9hG4bnbie7480nvo4
Max-Forwards: 70
To: Davison <sip:davison@hamal.mc21.fee.unicamp.br>
From: Davison <sip:davison@hamal.mc21.fee.unicamp.br>;tag=4n99b4
Call-ID: 94non40jnb4890hjt
CSeq: 4956 REGISTER
Contact: <sip:davison@regulus.mc21.fee.unicamp.br>
Expires: 3600
Content-Length: 0
```

Figura 13. Mensagem REGISTER

A requisição REGISTER contém vários campos. A linha de requisição da mensagem está conforme explicado na seção 3.4.1, e contém o método, o endereço do servidor de registros e a versão do protocolo. Na segunda linha da mensagem, existe o

campo Via. Este campo é usado para que os proxies possam rotear a mensagem apropriadamente. Quando o Agente Cliente do Usuário cria a mensagem, ele adiciona um campo Via contendo a versão do protocolo SIP, o protocolo utilizado para enviar a mensagem, o endereço IP do Terminal SIP e a porta. Neste campo existe um parâmetro branch que serve para identificar a transação cliente no terminal SIP que está gerenciando a transmissão da requisição. Abaixo do campo Via, existe o campo Max-Forwards. Ele serve para limitar o número de entidades SIP que esta mensagem pode passar. A cada entidade que esta mensagem passa, seu valor é decrementado de 1(um). Em seguida, existe o campo To. Na mensagem REGISTER, o campo To contém o endereço que será registrado no Servidor de Registros. Este endereço de registro será o endereço SIP que o terminal usará para estabelecer sessões. O campo From contém o endereço SIP da pessoa responsável pelo registro. O campo Call-ID contém um identificador da mensagem. Todas as mensagens REGISTER enviadas para o mesmo servidor de registros têm o mesmo valor do campo Call-ID. O campo CSeq garante a ordem das mensagens. A cada mensagem REGISTER enviada para o mesmo servidor de registros, o seu valor é aumentado de 1(um). O campo Contact contém o endereço IP do terminal SIP a ser registrado. Este campo pode conter mais de um endereço. O campo Expires contém o tempo que o registro permanecerá válido no servidor de registros. Este tempo é dado em segundos. O campo Content-Length indica o tamanho do corpo da mensagem SIP. No caso da mensagem REGISTER, o seu valor é zero.

Após a geração da mensagem, o terminal SIP envia a mensagem diretamente para o servidor de registros. O servidor de registros recebe a mensagem, a processa, faz o registro e envia a resposta confirmando o registro. A resposta é 200 Ok.

```
SIP/2.0 200 Ok
Via: SIP/2.0/UDP regulus.mc21.fee.unicamp.br:5060;branch=z9hG4bnbie7480nvo4
;received=143.106.50.80
Max-Forwards: 70
To: Davison <sip:davison@hamal.mc21.fee.unicamp.br>
From: Davison <sip:davison@hamal.mc21.fee.unicamp.br>;tag=4n99b4
Call-ID: 94non40jnb4890hjt
CSeq: 4956 REGISTER
Contact: <sip:davison@regulus.mc21.fee.unicamp.br>
Expires: 3600
Content-Length: 0
```

Figura 14. Mensagem de resposta à requisição REGISTER

A resposta 200 Ok é muito semelhante à requisição REGISTER enviada. A linha inicial contém a versão do protocolo, o código de resposta e a frase descrevendo o código da resposta. O corpo da mensagem é o mesmo da requisição enviada, a não ser pelo campo Via, que possui um parâmetro adicional chamado *received*. Este parâmetro é inserido pelo servidor de registros, indicando qual foi o endereço IP em que ele recebeu a mensagem.

Após a geração da mensagem, o servidor de registros envia a resposta para o Terminal SIP. O terminal SIP, no recebimento desta mensagem, confirma o registro do endereço SIP e a transação (troca de mensagens) termina.

3.6.2 Estabelecendo uma Sessão SIP

A transação mais importante do protocolo SIP é a transação INVITE. A mensagem INVITE é que estabelece as sessões SIP. Nesta seção, será descrito um cenário típico da transação INVITE. Neste cenário, existirão dois terminais SIP, dois proxies e dois servidores de registro, um para cada domínio SIP. A Figura 15 mostra este cenário.

Antes de descrevermos a transação INVITE, vamos supor que os terminais já efetuaram o registro conforme a seção anterior. O Terminal SIP 1 solicitou o registro do endereço SIP davison@hamal.mc21.fee.unicamp.br. O Terminal SIP 2 solicitou o registro do endereço SIP meireang@igniscom.com.br. Ainda neste cenário, nota-se que os proxies e os servidores de registro estão em um mesmo equipamento de rede.

A negociação inicia-se quando o Terminal SIP 1 cria a mensagem INVITE. Esta mensagem pode ser criada por algum estímulo externo, como por exemplo, quando um usuário de um aplicativo clica em algum item que inicia uma sessão de áudio SIP. A mensagem INVITE deverá ser como mostra a Figura 16.

Os campos Via, Max-Forwards, CSeq e o Content-Length têm a mesma função que na mensagem REGISTER. O campo To contém o endereço SIP do destinatário da requisição. Ele também contém uma *string* com o nome do destinatário da mensagem. Da mesma forma, o campo From também contém uma *string* com o nome do usuário do Terminal SIP 1 e o seu endereço SIP. Este campo também contém um parâmetro

chamado tag, cujo valor é uma *string* gerada pseudoaleatoriamente que é adicionada no Terminal SIP 1.

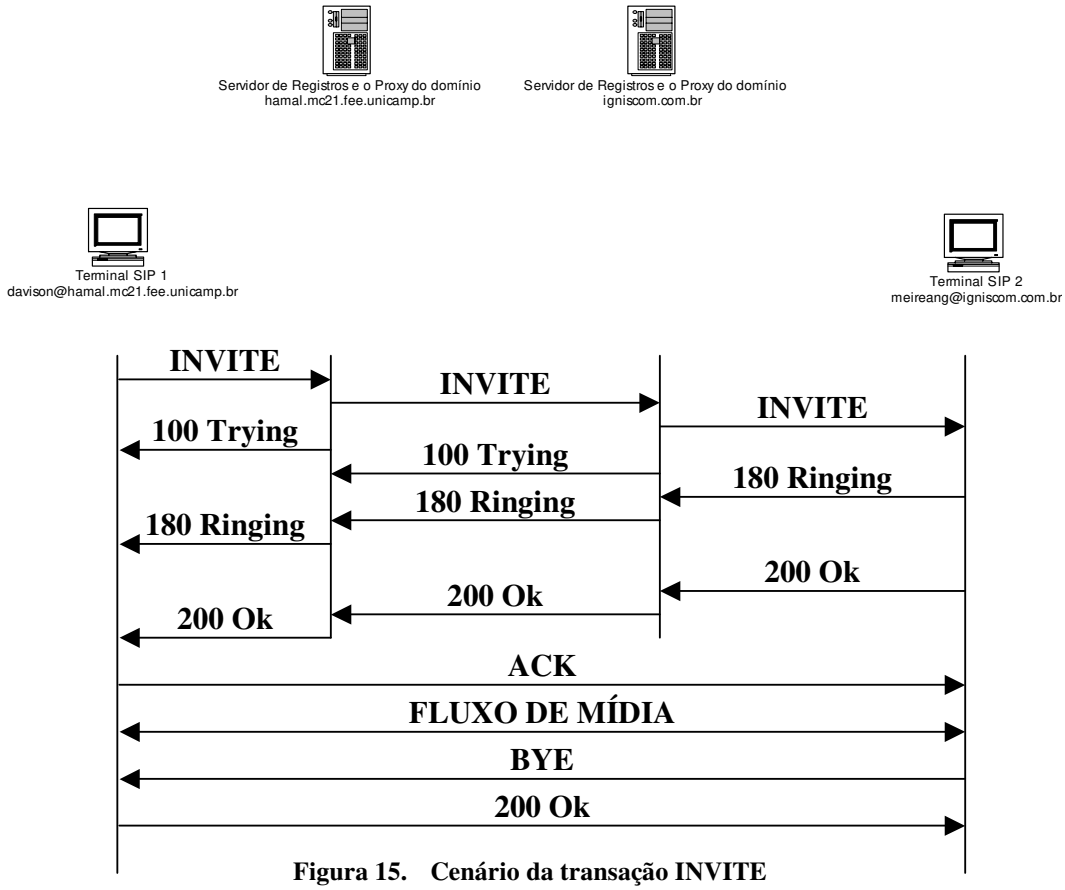

Figura 15. Cenário da transação INVITE

Este tag serve para identificação da chamada. O campo Call-ID contém um identificador para esta chamada. O campo Contact contém um endereço SIP no qual o usuário do Terminal 1 pode ser acessado diretamente. O campo Content-Type contém uma descrição do corpo da mensagem.

```
INVITE sip:meireang@igniscom.com.br SIP/2.0
Via: SIP/2.0/UDP regulus.mc21.fee.unicamp.br:5060;branch=z9hG4knib74u
Max-Forwards: 70
To: Meire Angelica Ferreira <sip:meireang@igniscom.com.br>
From: Davison Gonzaga da Silva <sip:davison@hamal.mc21.fee.unicamp.br>;tag=487jni9
Call-ID: j94nbgh46yu48nbfi
CSeq: 7982 INVITE
Contact: <sip:davison@regulus.mc21.fee.unicamp.br>
Content-Type: application/sdp
Content-Length: 142

v=0
o=davison 165468471 216549871332 IN IP4 143.106.50.80
s=audio call
u=http://www.mc21.fee.unicamp.br
e=davison@decom.fee.unicamp.br
c=IN IP4 143.106.50.80/7000
t=0 0
m=audio 7000 udp 0
```

Figura 16. Mensagem INVITE enviada pelo Terminal SIP 1

Após a criação da mensagem, o Terminal SIP 1 envia a mensagem para o Proxy responsável pelo seu domínio. Uma vez que o Terminal SIP 1 não sabe o endereço final do Terminal SIP 2, ele envia a mensagem para o seu Proxy de domínio, isto é, o Proxy 1. O Proxy 1 recebe a requisição e a envia para o Proxy 2 como se fosse o requisitante e responde para o Terminal SIP 1 com uma mensagem 100 Trying, indicando que o Proxy 1 está tentando localizar o destinatário daquela mensagem. A mensagem 100 Trying é como mostra a Figura 17.

```
SIP/2.0 100 Trying
Via: SIP/2.0/UDP regulus.mc21.fee.unicamp.br:5060;branch=z9hG4knib74u;
received=143.106.50.80
Max-Forwards: 69
To: Meire Angelica Ferreira <sip:meireang@igniscom.com.br>
From: Davison Gonzaga da Silva <sip:davison@hamal.mc21.fee.unicamp.br>;tag=487jni9
Call-ID: j94nbgh46yu48nbfi
CSeq: 7982 INVITE
Content-Length: 0
```

Figura 17. Resposta 100 Trying enviada pelo Proxy 1

O Proxy 1, antes de enviar a requisição para o Proxy 2, adiciona um campo Via na mensagem INVITE. A nova mensagem INVITE fica como mostra a Figura 18.

```
INVITE sip:meireang@igniscom.com.br SIP/2.0
Via: SIP/2.0/UDP hamal.mc21.fee.unicamp.br:5060;branch=z9hG4k70bneoijb94
Via: SIP/2.0/UDP regulus.mc21.fee.unicamp.br:5060;branch=z9hG4knib74u;
received=143.106.50.80
Max-Forwards: 70
To: Meire Angelica Ferreira <sip:meireang@igniscom.com.br>
From: Davison Gonzaga da Silva <sip:davison@hamal.mc21.fee.unicamp.br>;tag=487jni9
Call-ID: j94nbgh46yu48nbfi
CSeq: 7982 INVITE
Contact: <sip:davison@regulus.mc21.fee.unicamp.br>
Content-Type: application/sdp
Content-Length: 142

v=0
o=davison 165468471 216549871332 IN IP4 143.106.50.80
s=audio call
u=http://www.mc21.fee.unicamp.br
e=davison@decom.fee.unicamp.br
c=IN IP4 143.106.50.80/7000
t=0 0
m=audio 7000 udp 0
```

Figura 18. Mensagem INVITE enviada pelo Proxy 1

Isto é feito para que a mensagem de resposta a esta requisição possa passar pelo mesmo caminho que a requisição e os Proxies possam completar as suas transações abertas. O Proxy 2 recebe a mensagem INVITE e responde com uma resposta 100 Trying para o Proxy 1. O Proxy 2, então, consulta o seu Servidor de Registros para tentar descobrir o endereço IP de meireang@igniscom.com.br. O Proxy 2 encontra o endereço IP do Terminal SIP 2, adiciona um campo Via na requisição com o seu endereço, e envia a mensagem para o Terminal SIP 2.

O Terminal SIP 2 recebe a mensagem INVITE e alerta a Meire do recebimento da mensagem, perguntando se ela deseja aceitar ou não a mensagem. Ao mesmo tempo, responde com uma mensagem 180 Ringing, indicando para o Terminal SIP 1 que o Terminal SIP 2 recebeu a mensagem e que está aguardando uma resposta da Meire. Neste cenário, a Meire resolve aceitar a chamada. Então, o Terminal SIP 2 responde com uma resposta 200 Ok, contendo o corpo da mensagem SDP com o parâmetro de mídia que será estabelecido com o Terminal SIP 1, como mostra a Figura 19.

```
SIP/2.0 200 Ok
Via: SIP/2.0/UDP igniscom.com.br:5060;branch=z9hG4knvoin0485604;
received=143.106.56.200
Via: SIP/2.0/UDP hamal.mc21.fee.unicamp.br:5060;branch=z9hG4k70bneoijb94;
received=143.106.50.69
Via: SIP/2.0/UDP regulus.mc21.fee.unicamp.br:5060;branch=z9hG4knib74u;
received=143.106.50.80
Max-Forwards: 70
To: Meire Angelica Ferreira <sip:meireang@igniscom.com.br>;tag=bniehb496cv
From: Davison Gonzaga da Silva <sip:davison@hamal.mc21.fee.unicamp.br>;tag=487jni9
Call-ID: j94nbgh46yu48nbfi
CSeq: 7982 INVITE
Contact: <sip:davison@regulus.mc21.fee.unicamp.br>
Content-Type: application/sdp
Content-Length: 142

v=0
o=meireang 165468471 216549871332 IN IP4 143.106.50.80
s=audio call
u=http://www.mc21.fee.unicamp.br
e=meireang@decom.fee.unicamp.br
c=IN IP4 143.106.56.193/7000
t=0 0
m=audio 7000 udp 0
```

Figura 19. Resposta enviada pelo Terminal SIP 2

Após a criação da resposta, o Terminal SIP 2 envia a mensagem de resposta, que passará pelos dois Proxies até chegar ao Terminal SIP 1. Note que a resposta 200 Ok adiciona um tag ao campo To. Isto é feito para que o Terminal SIP 1 crie as peças de um **Diálogo**. O **Diálogo** é um conjunto de informações que identificam os Terminais SIP envolvidos em uma sessão. Estas informações compreendem os valores dos tag dos campos **To** e **From**, do valor do campo **Call-ID** da requisição, do valor do **CSeq** da requisição, dos valores dos campos **To**, **From** e **Contact**. Após a criação do diálogo por ambas as partes, as mensagens subseqüentes são trocadas diretamente entre os Terminais SIP. Além disto, as informações contidas nas variáveis armazenadas no Diálogo são utilizadas para gerar as mensagens dentro do Diálogo estabelecido. Quando o Terminal SIP 1 recebe a mensagem de resposta, ele responde com uma mensagem ACK, confirmando o recebimento da resposta 200 Ok e dizendo para o Terminal SIP 2 inicia o fluxo de áudio. O recebimento da resposta 200 Ok faz também que o Terminal SIP 1 estabeleça o diálogo e inicialize o fluxo de áudio. A mensagem ACK é como mostra a Figura 20.

```
ACK sip:meireang@ignis5.igniscom.com.br SIP/2.0
Via: SIP/2.0/UDP regulus.mc21.fee.unicamp.br:5060;branch=z9hG4knib74u
Max-Forwards: 70
To: Meire Angelica Ferreira <sip:meireang@igniscom.com.br>;tag=bniehb496cv
From: Davison Gonzaga da Silva <sip:davison@hamal.mc21.fee.unicamp.br>;tag=487jni9
Call-ID: j94nbgh46yu48nbfi
CSeq: 7982 ACK
Contact: <sip:davison@regulus.mc21.fee.unicamp.br>
Content-Length: 0
```

Figura 20. Mensagem ACK enviada pelo Terminal SIP 1

Durante a sessão INVITE, ou o Terminal SIP 1 ou o Terminal SIP 2 podem querer mudar os parâmetros da sessão. Isto é feito enviando-se um INVITE contendo os novos parâmetros que descrevem a mídia. Esta mensagem INVITE contém várias informações da mensagem INVITE anterior. Caso o outro Terminal aceite os novos parâmetros, ele responde com um 200 Ok para aceitar a mudança. O requisitante, então, responde com um ACK. Se a outra parte não desejar aceitar a mudança, ela responde com uma resposta de erro, como a 406 (Not Acceptable), que também recebe um ACK. Mas, uma falha na negociação de novos parâmetros no meio de uma sessão já estabelecida, não causa a finalização dessa sessão; a sessão continua usando os parâmetros negociados anteriormente.

3.6.3 Encerrando-se a Sessão

O encerramento de uma sessão é feito através do envio de uma mensagem BYE para o outro terminal participante da sessão. O cenário é o mesmo da Figura 15.

Neste cenário, a Meire (Terminal SIP 2) deseja encerrar a chamada. O Terminal SIP 2 finaliza o fluxo de áudio e gera uma mensagem BYE. A mensagem BYE é construída com as informações do diálogo, e ela finaliza o diálogo estabelecido entre os dois Terminais SIP. A mensagem BYE é como a mensagem da Figura 21.

```
BYE sip:davison@regulus.mc21.fee.unicamp.br SIP/2.0
Via: SIP/2.0/UDP ignis5.igniscom.com.br:5060;branch=z9hG4khgienhgie
Max-Forwards: 70
To: Davison Gonzaga da Silva <sip:davison@hamal.mc21.fee.unicamp.br>; tag=487jni9
From: Meire Angélica Ferreira <sip:meireang@igniscom.com.br>; tag=bniehb496cv
Call-ID: j94nbgh46yu48nbfi
CSeq: 1587 BYE
Content-Length: 0
```

Figura 21. Mensagem BYE enviada pelo Terminal SIP 2

O Terminal SIP 1, quando receber esta mensagem, finaliza o fluxo de áudio e envia uma resposta 200 Ok, encerrando a sessão.

```
SIP/2.0 200 Ok
Via: SIP/2.0/UDP ignis5.igniscom.com.br:5060;branch=z9hG4khgienhgie
Max-Forwards: 70
To: Davison Gonzaga da Silva <sip:davison@hamal.mc21.fee.unicamp.br>; tag=487jni9
From: Meire Angélica Ferreira <sip:meireang@igniscom.com.br>; tag=bniehb496cv
Call-ID: j94nbgh46yu48nbfi
CSeq: 1587 BYE
Content-Length: 0
```

Figura 22. Resposta 200 Ok enviada pelo Terminal SIP 1

3.6.4 Cancelando uma Requisição INVITE

Às vezes é necessário interromper o início de uma sessão INVITE. Existem vários motivos para isso, como por exemplo, o cliente da sessão não poderá mais estabelecer a conversa porque precisou fazer outra coisa. Para se cancelar uma requisição INVITE, o cliente usa a mensagem CANCEL. Esta mensagem é usada para cancelar uma sessão INVITE antes do recebimento de uma resposta final (2xx – 6xx). Caso o cliente já tenha recebido uma resposta final, ele termina o estabelecimento da sessão e envia um BYE para finalizar a sessão. O cenário é mostrado na Figura 23.

O INVITE inicial é como o da seção 3.6.2. O Terminal SIP 1, desejando cancelar o INVITE enviado, gera a requisição CANCEL. Os campos Call-ID, To, a parte numérica do CSeq e o campo From da mensagem CANCEL são idênticos aos campos da mensagem INVITE enviada. No campo CSeq, a parte do método, ao invés de INVITE, agora é CANCEL.

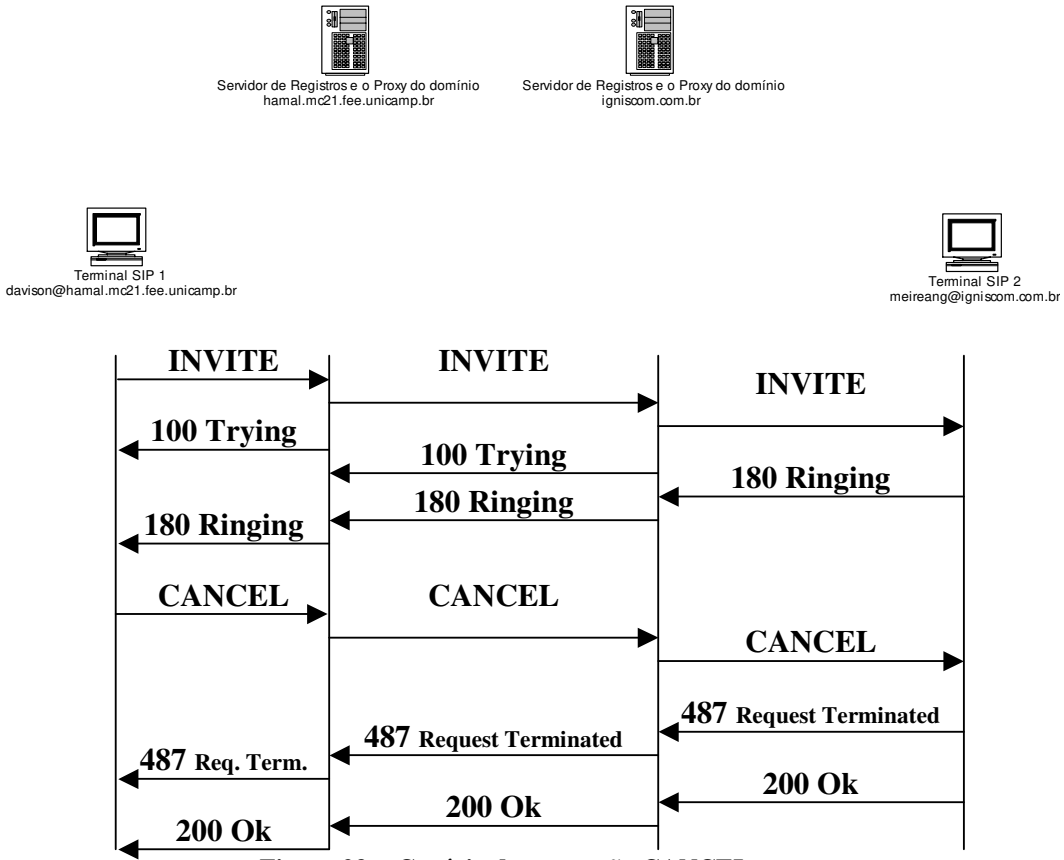

Figura 23. Cenário da transação CANCEL

Portanto, a mensagem final deverá ser como a da Figura 24.

```
CANCEL sip:meireang@igniscom.com.br SIP/2.0
Via: SIP/2.0/UDP regulus.mc21.fee.unicamp.br:5060;branch=z9hG4knib74u
Max-Forwards: 70
To: Meire Angelica Ferreira <sip:meireang@igniscom.com.br>
From: Davison Gonzaga da Silva <sip:davison@hamal.mc21.fee.unicamp.br>;tag=487jni9
Call-ID: j94nbgh46yu48nbfi
CSeq: 7982 CANCEL
Content-Length: 0
```

Figura 24. Mensagem CANCEL enviada pelo Terminal SIP 1

Após a geração da mensagem, o Terminal SIP 1 envia a mensagem para o Proxy 1. O Proxy 1, quando receber a mensagem, pode agir de duas formas diferentes. A primeira é responder a requisição CANCEL com um 487 (Request Terminated), para finalizar a transação INVITE, e, depois, com um 200 Ok, para finalizar a transação CANCEL, e repassar a mensagem para o Proxy 2. Na segunda forma de agir, o Proxy 1

pode repassar a mensagem CANCEL para o Proxy 2 sem respondê-la. Esta forma de agir é que está representada na Figura 23. O Proxy 1, então, repassa a mensagem CANCEL para o Proxy 2, que toma a mesma atitude, repassando a mensagem CANCEL para o terminal SIP 2. O Terminal SIP 2, ao receber a mensagem CANCEL, verifica se o usuário (Meire), já aceitou a chamada. Se o usuário não aceitou a chamada, ele pára a tentativa de conexão e responde a mensagem INVITE com a mensagem 487 (Request Terminated).

```
SIP/2.0 487 Request Terminated
Via: SIP/2.0/UDP igniscom.com.br:5060;branch=z9hG4knvoin0485604;
received=143.106.56.200
Via: SIP/2.0/UDP hamal.mc21.fee.unicamp.br:5060;branch=z9hG4k70bneoijb94;
received=143.106.50.69
Via: SIP/2.0/UDP regulus.mc21.fee.unicamp.br:5060;branch=z9hG4knib74u;
received=143.106.50.80
Max-Forwards: 70
To: Meire Angelica Ferreira <sip:meireang@igniscom.com.br>
From: Davison Gonzaga da Silva <sip:davison@hamal.mc21.fee.unicamp.br>;tag=487jni9
Call-ID: j94nbgh46yu48nbfi
CSeq: 7982 INVITE
Content-Length: 0
```

Figura 25. Resposta ao INVITE enviado após o recebimento da requisição CANCEL

A mensagem de resposta, então, é enviada para o Proxy 2. O Proxy 2, ao receber esta mensagem, cancela a transação INVITE e repassa a mensagem para o Proxy 1. O Proxy 1, ao receber a mensagem, toma a mesma atitude e a repassa para o Terminal SIP 1. O Terminal SIP 1, ao receber a mensagem, finaliza a transação INVITE.

Após o envio da mensagem 487 (Request Terminated), o Terminal SIP 2 gera uma resposta 200 Ok para o CANCEL enviado. Esta resposta é repassada de volta pelo mesmo caminho da resposta CANCEL até o Terminal SIP 1. Ao recebimento desta mensagem, o Terminal SIP 1 termina a transação CANCEL.

```
SIP/2.0 200 Ok
Via: SIP/2.0/UDP igniscom.com.br:5060;branch=z9hG4knvoin0485604;
received=143.106.56.200
Via: SIP/2.0/UDP hamal.mc21.fee.unicamp.br:5060;branch=z9hG4k70bneoijb94;
received=143.106.50.69
Via: SIP/2.0/UDP regulus.mc21.fee.unicamp.br:5060;branch=z9hG4knib74u;
received=143.106.50.80
Max-Forwards: 70
To: Meire Angelica Ferreira <sip:meireang@igniscom.com.br>
From: Davison Gonzaga da Silva <sip:davison@hamal.mc21.fee.unicamp.br>;tag=487jni9
Call-ID: j94nbgh46yu48nbfi
CSeq: 7982 CANCEL
Content-Length: 0
```

Figura 26. Mensagem de resposta à transação CANCEL

3.6.5 Descobrindo as Capacidades dos Terminais SIP

O método OPTIONS permite um terminal questionar outro terminal ou um servidor proxy a respoeito de suas capacidades. Isto permite que o terminal descubra informações sobre os métodos suportados, tipos de conteúdos de seção que o terminal aceita, extensões, *codecs*, etc., sem ter que estabelecer uma sessão com o outro terminal.

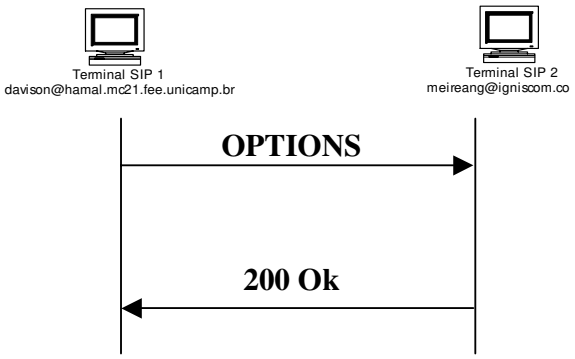

Figura 27. Cenário da transação OPTIONS

No caso do cenário da Figura 27, a mensagem OPTIONS é gerada pelo Terminal SIP 1.

Na mensagem OPTIONS, existe o campo Accept. Este campo indica qual é o tipo do corpo da mensagem que o cliente deseja receber na resposta. Tipicamente, o valor deste campo é fixado para o formato para o qual será usado para descrever as capacidades da mídia de um Terminal SIP.

```
OPTIONS sip:meireang@igniscom.com.br SIP/2.0
Via: SIP/2.0/UDP regulus.mc21.fee.unicamp.br:5060;branch=z9hG4kgheinbubbg004
Max-Forwards: 70
To: Meire Angelica Ferreira <sip:meireang@igniscom.com.br>
From: Davison Gonzaga da Silva <sip:davison@hamal.mc21.fee.unicamp.br>;tag=4nb84n
Call-ID: bhpvdwg9749bnh49h44
CSeq: 2541 OPTIONS
Contact: <sip:davison@regulus.mc21.fee.unicamp.br>
Accept: application/sdp
Content-Length: 0
```

Figura 28. Mensagem OPTIONS enviada pelo Terminal SIP 1

Após a criação da mensagem OPTIONS, ela é enviada para o Proxy 1. No caso da mensagem OPTIONS, os Proxies, ao receberem a mensagem, não agem como se fossem o terminal cliente. Eles agem, simplesmente, como um passador de mensagens. Por isso, eles não são mostrados na Figura 27. Portanto, a mensagem passa de forma transparente pelos dois proxies e chega ao Terminal SIP 2.

O Terminal SIP 2, ao receber a mensagem, responde com uma mensagem 200 Ok (Figura 29). Os campos Allow, Accept, Accept-Enconding, Accept-Language e Supported vão estar presentes na mensagem 200 Ok.

O campo Accept, como dito anteriormente, indica os formatos do corpo da mensagem SIP que são aceitáveis pelo Terminal. SIP 2 O campo Accept-Encoding indica o tipo de codificação que o Terminal SIP 2 aceita receber. Neste cenário, o Terminal SIP 2 aceita receber mensagens compactadas com o padrão gzip. O campo Accept-Language indica a linguagem que será usada nas frases que descreverão o código de resposta. O campo Supported enumera todas as extensões que o Terminal SIP suporta. Estas extensões podem ser específicas às implementações ou podem ser padronizadas em RFC's.

```
SIP/2.0 200 Ok
Via: SIP/2.0/UDP igniscom.com.br:5060;branch=z9hG4knvoin0485604;
received=143.106.56.200
Via: SIP/2.0/UDP hamal.mc21.fee.unicamp.br:5060;branch=z9hG4k70bneoijb94;
received=143.106.50.69
Via: SIP/2.0/UDP regulus.mc21.fee.unicamp.br:5060;branch=z9hG4knib74u;
received=143.106.50.80
To: Meire Angelica Ferreira <sip:meireang@igniscom.com.br>
From: Davison Gonzaga da Silva <sip:davison@hamal.mc21.fee.unicamp.br>;tag=4nb84n
Call-ID: bhpvdwg9749bnh49h44
CSeq: 2541 OPTIONS
Contact: <sip:meireang@ignis5.igniscom.com.br>
Allow: INVITE, ACK, CANCEL, OPTIONS, BYE
Accept: application/sdp
Accept-Encoding: gzip
Accept-Language en
Supported: foo
Content-Type: application/sdp
Content-Length: 242

v=0
o=meireang 215426498 546149841 IN IP4 143.106.56.200
s=audio call
e=meireang@decom.fee.unicamp.br
c=IN IP4 143.106.56.200/7000
t=0 0
m=audio 7000 udp 0 96 97 98
a=wb:96 L8/8000
a=wb:97 U8/8000
a=wb:98 U8/16000
```

Figura 29. Mensagem 200 Ok gerada pelo Terminal SIP 2.

4. Protocolo SDP

4.1 Introdução

Como foi visto no capítulo anterior, o protocolo SIP estabelece e gerencia sessões entre duas ou mais entidades, mas ele não descreve os parâmetros de áudio, vídeo, etc..., de uma sessão. Para a descrição destes parâmetros, o protocolo SIP carrega, como corpo da mensagem, um outro protocolo. No caso de sessões multimídia, o IETF definiu o protocolo SDP – *Session Description Protocol* para descrever os parâmetros da sessão multimídia aberta pelo protocolo SIP. Este protocolo é especificado na RFC 2327 – *Session Description Protocol*[2].

O protocolo SIP, no entanto, não depende do protocolo SDP para descrever uma sessão multimídia. Pode-se usar qualquer outro protocolo, inclusive as mensagens do H.323, para se descrever uma sessão multimídia. A vantagem do SDP é que ele é um protocolo bem simples e compacto, capaz de carregar informações suficientes para a descrição dos parâmetros da mídia a ser trocada. Esta vantagem nos levou a escolher implementar o SDP neste sistema.

Neste capítulo, será descrito o protocolo SDP. Primeiramente, será dada uma visão geral do uso do protocolo SDP. Em seguida, serão descritas as suas características e as suas funcionalidades. E, por último, serão descritos os principais campos que compõem uma mensagem SDP.

4.2 Uso do Protocolo SDP

O protocolo SDP é um protocolo descritor de sessão para sessões multimídias. Um uso comum do protocolo é quando um cliente deseja anunciar uma sessão multimídia para um servidor. Esta sessão multimídia é descrita pelo protocolo SDP e é estabelecida através do protocolo SIP. O protocolo SIP, então, transporta em seu corpo a mensagem SDP, como mostra a Figura 30.

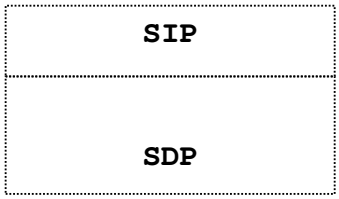

Figura 30. Mensagem SIP carregando a mensagem SDP como um corpo

As mensagens SIP, que geralmente carregam um corpo SDP, são a requisição INVITE e a mensagem de resposta 200 Ok.

4.2.1 Funcionalidades do Protocolo SDP

O propósito básico do SDP é conter informações sobre os fluxos de mídia da sessão para permitir que as partes envolvidas possam estabelecer canais com as configurações apropriadas. Para este propósito, o protocolo SDP inclui em sua mensagem campos que descrevem:

- O nome da sessão e o propósito;
- O tempo em que a sessão está ativa, quando a sessão é limitada em tempo;
- A mídia que será trocada na sessão; e
- Os parâmetros para se receber a mídia, que incluem: o tipo de mídia (áudio, vídeo, etc...), o protocolo de transporte (RTP/UDP/IP, etc.), e o formato da mídia (H.261, G.711, etc...).

O protocolo SDP também pode descrever, em casos onde os recursos da rede são limitados, informações sobre a largura de banda a ser usada na sessão.

4.3 Descrição do Protocolo SDP

O protocolo SDP, assim como o protocolo SIP, é baseado em mensagens puramente textuais codificadas em UTF-8[5]. As mensagens SDP consistem de um número finito de linhas de campos que descrevem os parâmetros da sessão. Estes campos possuem a seguinte forma:

<tipo>=<valor>

O **<tipo>** é um caractere que descreve o campo e **<valor>** é uma *string* estruturada de acordo com o campo. Por exemplo, uma linha do protocolo SDP tem o seguinte formato:

v=0

O tipo é o caractere **v** e o valor é o caractere **0**. O protocolo contém vários campos, que podem ter significados diferentes dependendo da sua posição na mensagem. Por isso, os campos da mensagem SDP são subdivididos da seguinte em:

Descritores da Sessão

v= (versão do protocolo)

o= (identificação do requisitante da sessão)

s= (nome da sessão)

i= (informação sobre a sessão)

u= (página da Internet que contenha a descrição da sessão)

e= (endereço de e-mail do requisitante)

p= (número do telefone do requisitante)

c= (informações sobre a conexão)

b= (largura de banda exigida pela sessão)

z= (ajuste do relógio entre as localidades)

k= (chave criptográfica)

a= (atributos da sessão)

Descrição de Tempo de Sessão

t= (tempo em que a sessão vai estar ativa)

r= (número de repetições do tempo em que a sessão pode estar ativa)

Descrição da Mídia

m= (nome da mídia e endereço de transporte)

i= (título da mídia)

c= (informações sobre a conexão)

b= (informações sobre a largura de banda)

k= (chave criptográfica)

a= (atributos da mídia)

Apesar do protocolo suportar todos estes campos, somente 5 (cinco) são obrigatórios nas mensagens SDP: **v**, **o**, **s**, **t** e **m**. O restante dos campos são opcionais. Além deste conjunto de campos, podem ser adicionados outros campos específicos à implementação, pois a RFC 2327 especifica que campos não reconhecidos em uma mensagem SDP devem ser ignorados.

Um exemplo de uma mensagem SDP é mostrado abaixo.

```
v=0
o=meireang 165468471 216549871332 IN IP4 143.106.56.200
s=audio call
e=meireang@decom.fee.unicamp.br
c=IN IP4 143.106.56.200/7000
t=0 0
m=audio 7000 udp 0 96 97 98
a=wb:96 L8/8000
a=wb:97 U8/8000
a=wb:98 U8/16000
```

A seguir, será apresentada uma descrição dos campos que compõem esta mensagem. Os outros campos da mensagem SDP são descritos no Apêndice II.

4.3.1 O Campo v

O campo **v** identifica a versão do protocolo SDP. Atualmente, existe somente uma única versão do protocolo, e a RFC 2327[2] define o valor deste campo como sendo 0.

4.3.2 O Campo o

O campo **o** contém informações sobre o requisitante da sessão. O seu valor tem o seguinte formato:

`o=<username> <session id> <version> <network type> <address type> <address>`

O valor **username** é o nome do usuário do terminal que originou a mensagem SDP. No caso do uso em conjunto com o protocolo SIP, o nome do usuário SIP é colocado neste valor. O **session id** é uma *string* numérica que serve para identificar a chamada. O valor **version** é um número de versão desta descrição. Ele é gerado aleatoriamente e tem o propósito de manter a seqüência das mensagens SDP enviadas. A

cada descritor enviado para um mesmo destinatário, o seu valor é aumentado de 1 (um). O **network type** é uma *string* que define o tipo de rede em que a mensagem vai atravessar. Para este valor, **IN** significa rede de Internet. O valor **address type** é o tipo de protocolo IP que será usado na troca de mídia. Os tipos definidos para este valor são **IP4**, indicando o protocolo IP versão 4, e o **IP6**, indicando o protocolo IP versão 6. E, o valor **address** é o endereço IP da máquina que está requisitando esta sessão.

4.3.3 Os Campos "e" e s

O campo **s** é um campo que contém uma *string* que descreve a sessão a ser estabelecida. O campo **e** contém o endereço de e-mail do requisitante da sessão multimídia.

4.3.4 O Campo c

O campo **c** descreve a conexão do requisitante. Ele tem o seguinte formato:

```
c=<network type> <address type> <connection address>
```

Os valores **network type** e **address type** têm o mesmo valor que os definidos no campo **o**. O valor **connection address** geralmente é da forma IP/porta, por exemplo, 143.106.50.80/7000.

4.3.5 O Campo t

O campo **t** é o campo que contém o tempo, em segundos, no qual a sessão deverá permanecer ativa. Este campo tem o seguinte formato:

```
t=<start time> <stop time>
```

Os valores **start time** e **stop time** são os valores do tempo de início e fim da sessão. Se a sessão não é limitada por tempo, os seus valores são 0.

4.3.6 O Campo m

O campo **m** é o campo que contém uma descrição das mídias que são permitidas pelo requisitante da sessão. O seu formato geral é:

```
m=<media> <port> <transport> <fmt list>
```

A mensagem SDP pode conter vários campos **m**. Cada campo contém um descritor de uma mídia diferente.

O valor **media** é o tipo de mídia. Os valores definidos pela RFC 2327 são “audio”, “video", “application”, “data” e “control”, embora esta lista possa ser estendida quando novas modalidades de comunicações surgirem. O valor “control” é usado quando é necessária a adição de um canal de controle para a mídia.

O valor **port** contém a porta no qual os pacotes de mídia serão entregues.

O valor **transport** é o protocolo de transporte do fluxo de mídia. Os seus valores podem ser RTP/AVP ou udp.

O valor **fmt list** é uma lista de formatos de mídia. Estes formatos são especificados na RFC 1890 – *RTP Profile for Audio and Video Conferences with Minimal Control*[14]. Esta lista de formatos são números inteiros que variam de 0 até 127, onde os valores de 0 até 95 são valores reservados para a RFC 1890. Os outros valores descrevem implementações específicas. Por exemplo, o número 0 identifica o *codec* G.711 Lei μ. Quando o **fmt list** possui valores de 96 a 127, o campo **m** vem acompanhado por um ou mais campos **a**, cada qual descrevendo um *codec*. Na mensagem da seção 4.3, existem três campos **a** descrevendo os valores 96, 97 e 98.

4.3.7 Campo a

O campo **a** é o campo de atributos. Ele pode ter vários valores, dependendo da aplicação. Para o caso dos descritores de mídia, o campo **a** tem a seguinte forma:

```
a=<attribute>:<payload type> <encoding name>/<clock rate>
```

O valor **attribute** dependerá do protocolo em uso. Se o protocolo for RTP, o seu valor será rtpmap. Se for UDP, o seu valor será wb.

O valor **payload type** indica um número do valor **fmt list** do campo **m**. O valor **encoding name** contém o tipo de codificação da mídia. Por exemplo, L8 significa uma codificação de 8 bits linear. E o valor **clock rate** é a taxa de amostragem da mídia. Por exemplo, 8000 significa que as amostras são colhidas a uma taxa de 8 KHz.

Para cada número da **fmt list**, deverá existir um campo **a**. Uma lista completa dos formatos possíveis para o campo **a** é dada na RFC 2327[2].

5. Descrição Funcional do Sistema SIP Implementado

5.1 Introdução

Neste capítulo apresentaremos o Sistema SIP implementado. Esta implementação é baseada nos protocolos SIP e SDP apresentados nos capítulos anteriores. Como vimos, um Sistema SIP é um conjunto de entidades que usam as funcionalidades do protocolo SIP para prover serviços. Um Sistema SIP é composto de três entidades básicas, conforme apresentado na seção 3.5:

- **Terminal SIP**: É a entidade responsável por estabelecer sessões através do protocolo SIP;
- **Proxy**: É a entidade responsável por receber as mensagens, processá-las e enviá-las para a entidade de destino, onde a entidade de destino pode ser outro Proxy, um Terminal SIP ou um Servidor de Registros; e
- **Servidor de Registros**: É a entidade responsável por efetuar o registro dos usuários no domínio de sua responsabilidade. O Servidor de Registros é uma entidade que só processa requisições REGISTER e OPTIONS.

Nossa implementação é baseada nestas três entidades, sendo que o Proxy foi implementado como um Proxy sem estados. Um Proxy sem estados, ou sem memória, é aquele que processa as mensagens, e as envia sem tomar parte da sua transação. O Terminal SIP foi construído utilizando-se a biblioteca QT[25][28], que é uma biblioteca da linguagem C++ voltada a eventos e sinais (Apêndice III). O Servidor de Registros e o Proxy também foram implementados em C++, visando a sua execução como processos em segundo plano do Sistema Operacional. Todo o Sistema foi implementado para o Sistema Operacional Linux, porque este oferece ferramentas de programação e de acesso de baixo nível ao núcleo do sistema operacional e ao driver do dispositivo.

O restante deste capítulo fará uma descrição do funcionamento do Sistema SIP desenvolvido. A primeira seção tratará do Terminal SIP, descrevendo as suas características e suas funcionalidades. A segunda seção descreverá o Servidor de Registros e o Proxy. A última seção fará uma descrição do sistema como um todo.

5.2 Terminal SIP

O Terminal SIP é a entidade que faz a interação com o usuário final do sistema. Ele utiliza as mensagens SIP para estabelecer e finalizar sessões de áudio e efetuar registros. Como vimos na seção 3.3 o Terminal SIP é dividido em três camadas:

- **Transação do Usuário (TU)**;
- **Camada de Transação**; e
- **Camada de Transporte**.

Estas três camadas trabalham em conjunto para fornecer as funcionalidades SIP. Além destas três camadas SIP, o Terminal implementado possui uma biblioteca de acesso à placa de áudio. Além de prover os serviços de captura e entrega de pacotes de áudio, esta biblioteca permite a configuração dos parâmetros do *driver* da placa de áudio (seção 5.2.4).

O funcionamento do Terminal SIP é mostrado na Figura 31.

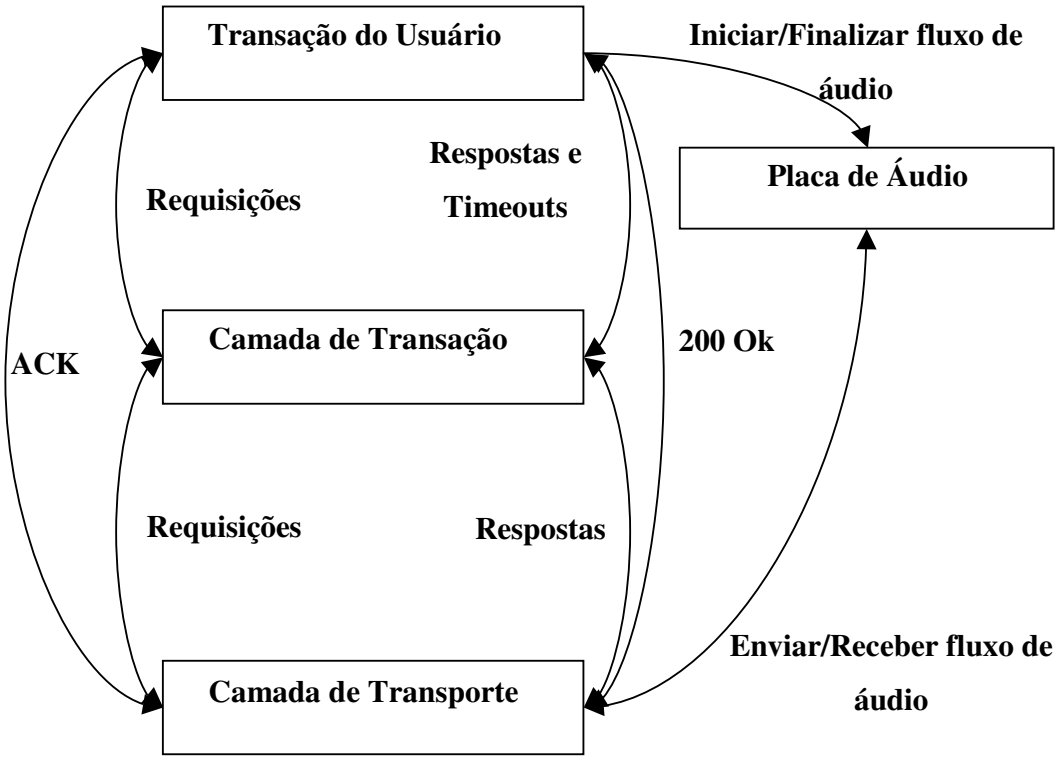

Figura 31. Comunicação entre as camadas do Terminal SIP

Cada camada do Terminal SIP é implementada por uma ou mais classes. Por exemplo, a Camada de Transação foi implementada através de quatro classes: **CInvite_Client_trans**, **Cnon_Invite_Client_trans,** **C_Invite_Server_trans** e **Cnon_Invite_Server_trans**. Os detalhes da implementação são dados no Apêndice IV.

5.2.1 Transação do Usuário

A Transação do Usuário é a camada mais alta do Terminal SIP. Nesta camada é que são feitos a geração e o processamento das requisições e respostas. A geração de requisições e o processamento das respostas a essas requisições são feitos pela parte cliente da Transação do Usuário. O processamento das requisições recebidas e a geração das respostas às requisições recebidas são feitos pela parte servidora da Transação do Usuário. Por isso, a Transação do Usuário é subdividida em duas porções:

- **Agente Cliente do Usuário**: É a parte cliente da Transação do Usuário; e
- **Agente Servidor do Usuário**: É a parte servidora da Transação do Usuário.

A Transação do Usuário pode ser vista na Figura 32.

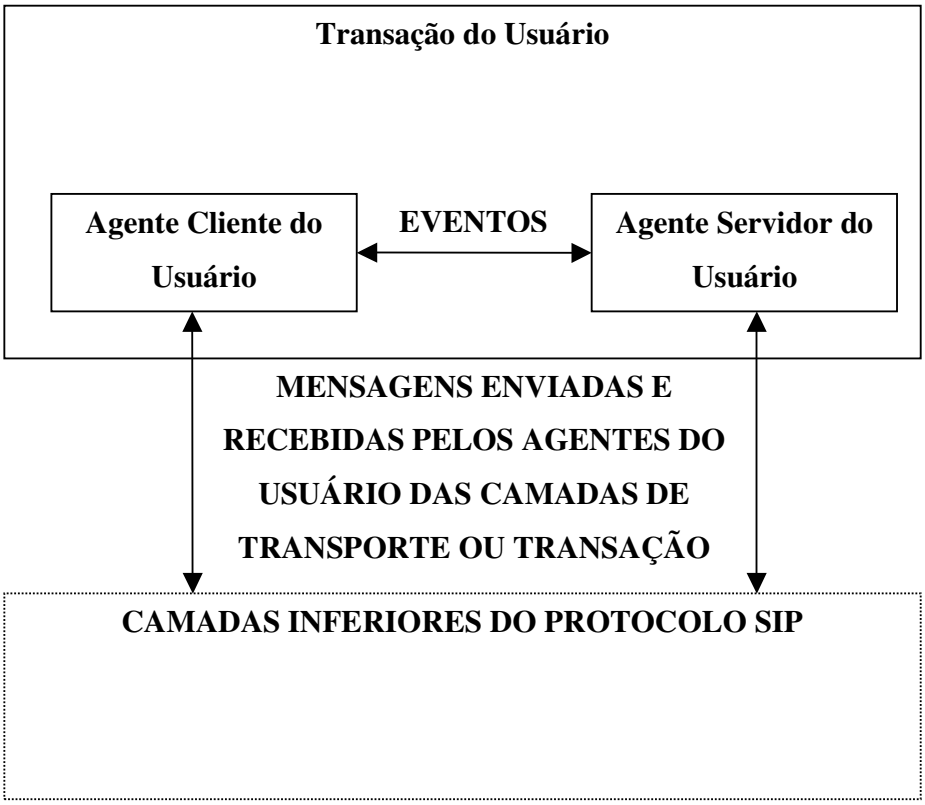

Figura 32. Subdivisão da camada de Transação do Usuário

5.2.1.1 Agente Cliente do Usuário

O Agente Cliente do Usuário é a subcamada da Transação do Usuário responsável por gerar as requisições, criar a transação que gerenciará a transmissão desta requisição e processar as respostas a esta requisição.

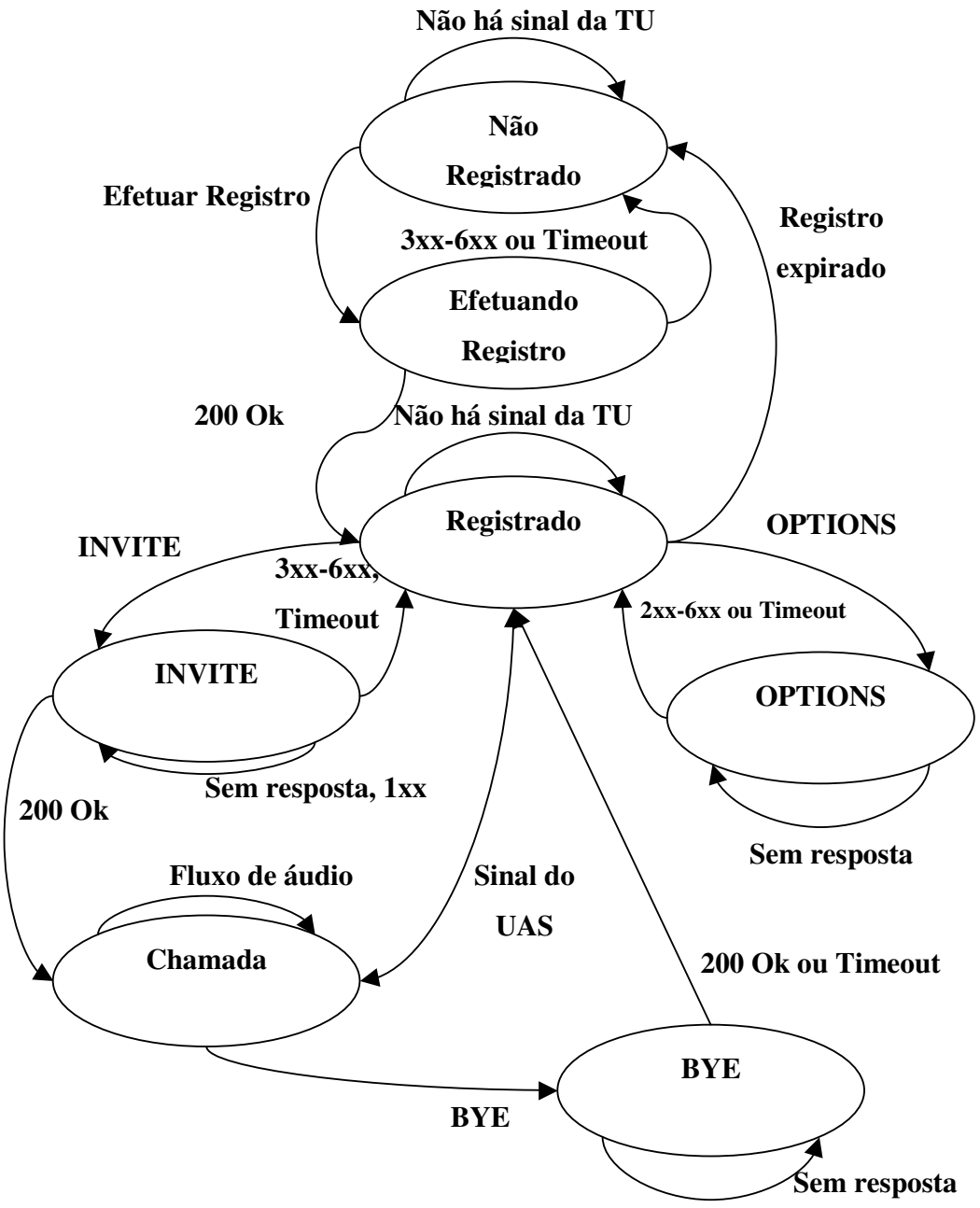

Figura 33. Máquina de estados do Agente Cliente do Usuário

A Figura 33 modela o comportamento implementado para a subcamada Agente Cliente do Usuário. O estado inicial do Agente Cliente é o **não Registrado**. Neste estado, o Agente Cliente fica esperando um sinal para efetuar o registro do terminal. Quando o sinal de Registro chega, o Agente Cliente passa para o estado **Efetuando Registro**. Neste estado, o Agente Cliente gera uma requisição **REGISTER**, cria uma Transação Cliente

não-INVITE na Camada de Transação, e envia a mensagem, o endereço IP de destino da mensagem, a porta e o protocolo a ser utilizado (na implementação atual, o protocolo usado é o UDP) para a Transação criada. Após o envio da mensagem para a Camada de Transação, o Agente Cliente fica aguardando uma resposta da Camada de Transação. Quando a Camada de Transação recebe alguma resposta, ela envia a resposta para o Agente Cliente. O Agente Cliente, então, examina o código de status da mensagem. Se a mensagem for **200 Ok**, ele vai para o estado **Registrado** e sinaliza, para o usuário, que o registro foi efetuado com sucesso. Se a resposta for entre **3xx – 6xx**, o Agente Cliente sinaliza o erro e volta para o estado inicial. A Camada de Transação pode, também, enviar um sinal indicando um **Timeout** da transação, isto é, a Camada de Transação não recebeu nenhuma resposta dentro do tempo máximo da transação, que é de 32 segundos. Ao receber o **Timeout**, o Agente Cliente sinaliza o estouro de tempo e volta para o estado inicial.

No estado **Registrado**, o Agente Cliente pode efetuar uma chamada, quando recebe um sinal para iniciá-la, pode enviar uma mensagem **OPTIONS**, para levantar as capacidades de um outro Terminal SIP, ou pode receber um sinal do Agente Servidor indicando que o Terminal SIP está em uma chamada. Se o Agente Cliente receber um sinal para iniciar uma chamada, ele vai para o estado **INVITE**. Neste estado, o Agente Cliente gera uma requisição **INVITE**, contendo os campos da mensagem e o protocolo SDP, como corpo da mensagem. Após a geração da mensagem, o Agente Cliente cria uma Transação Cliente INVITE, na Camada de Transação, e passa a mensagem, o endereço IP, a porta do destino e o protocolo a ser usado. A partir daí, o Agente Cliente fica aguardando o recebimento de uma resposta. Se a Camada de Transação entregar uma resposta **1xx**, o Agente Cliente permanece neste estado. Se a reposta for entre **3xx – 6xx**, ele retorna para o estado **Registrado** e indica, para o usuário o erro. Se a resposta for **2xx**, o Agente Cliente vai para o estado **Chamada**. O Agente Cliente também pode receber, da Camada de Transação, um sinal de **Timeout**. Neste caso, também ocorre a volta para o estado Registrado.

Após o recebimento da resposta **2xx**, o Agente Cliente vai para o estado **Chamada**. Neste estado, o Agente Cliente cria o **Diálogo** entre os dois terminais. Após a criação do **Diálogo**, o Agente Cliente gera uma confirmação (ACK) e envia a resposta

diretamente para a Camada de Transporte. A partir do envio do ACK, o Agente Cliente envia um sinal, para o núcleo do Terminal SIP, indicando o início do fluxo de áudio. Enquanto ele está neste estado, ele responde a retransmissões da resposta **2xx** que ele venha a receber. Ainda neste estado, o Agente Cliente avisa para o Agente Servidor que ele está em uma chamada. Este aviso consiste de um sinal que é enviado para o Agente Servidor contendo o **Diálogo** estabelecido. O Agente Servidor, ao receber este sinal, muda o seu estado para **Chamada**. Neste estado, o Agente Cliente fica aguardando um pedido de finalização da chamada ou um sinal vindo do Agente Servidor, indicando o fim da chamada. Se o Agente Cliente receber um pedido de finalização de chamada, ele vai para o estado **BYE**.

No estado **BYE**, o Agente Cliente pára o fluxo de áudio e constrói uma mensagem **BYE**. Esta mensagem é construída com os dados do **Diálogo** e ela finalizará o **Diálogo** estabelecido entre os dois terminais. Após a construção da mensagem, o Agente Cliente cria uma Transação Cliente não-INVITE e passa a mensagem. A seguir, ele fica esperando uma resposta da Camada de Transação. Qualquer resposta que o Agente Cliente receber da camada de transação, vai fazer com que ele mude do estado **BYE** para o estado **Registrado**, destruindo o **Diálogo** estabelecido.

De volta ao estado **Registrado**, o Agente Cliente pode receber um sinal **OPTIONS**. O recebimento deste sinal causa a mudança do estado **Registrado** para o estado **OPTIONS**. Neste estado, o Agente Cliente gera uma mensagem **OPTIONS**, cria uma Transação Cliente não-INVITE, e passa a mensagem, o endereço IP, a porta e o protocolo a ser utilizado pela Camada de Transporte para o envio da mensagem. Então, o Agente Cliente fica aguardando uma resposta da Camada de Transação. Qualquer resposta, que o Agente Cliente receber, causará a transição do estado **OPTIONS** de volta para o estado **Registrado**.

A Figura 33, mostra uma mudança do estado **Registrado** para o estado **Chamada**, e vice-versa. Isto ocorre pelo recebimento de um sinal, do Agente Servidor, indicando que o Terminal SIP está em uma chamada estabelecida pelo Agente Servidor, ou que a chamada foi finalizada pelo Agente Servidor.

5.2.1.2 Agente Servidor do Usuário

O Agente Servidor do Usuário é a subcamada da Transação do Usuário responsável por receber as requisições, processá-las e respondê-las. Como para o Agente Cliente do Usuário, todas as requisições são acompanhadas por uma transação na Camada de Transação. O funcionamento do Agente Servidor é mais simples do que o do Agente Cliente, e, também, a Figura 34 modela o comportamento implementado para esta subcamada.

O estado inicial é o **não-REGISTRADO**. Neste estado, o Agente Servidor aguarda o recebimento de um sinal do Agente Cliente para ir para o estado **REGISTRADO.** No estado **REGISTRADO**, o Agente Servidor está pronto para receber uma requisição. Quando uma requisição chega, o Agente Servidor verifica o método da requisição. Se o Agente Servidor reconhece o método, ele continua o processamento da mensagem. Se ele não conhecer o método, o Agente Servidor responde com uma mensagem 405 (*Method Not Allowed*). O Agente Servidor ainda inclui um campo na mensagem indicando os métodos que ele suporta. Após a inspeção do método, o Agente Servidor inspeciona os campos da mensagem. Se o Agente Servidor não entender algum campo que não seja essencial na mensagem, ele ignora este campo. Se o campo não entendido for um campo essencial, o Agente Servidor responde com uma mensagem 400 (*Bad Request*). Os campos essenciais em uma mensagem SIP são: **To, From, Call-ID, CSeq, Via** e **Max-Forwards**. Na inspeção dos campos da mensagem, o Agente Servidor verifica se a mensagem é para ele mesmo, verificando o campo **To** da mensagem. Após o processamento do campo, o Agente Servidor processa o corpo da mensagem, se ela o possuir. Para o corpo de mensagem SDP, se o Agente Servidor não suportar nenhum tipo de mídia especificado na mensagem, ele responde com uma mensagem 415 (*Unsupported Media Type*). Esta resposta conterá o tipo de mídia que o Agente Servidor suporta. Após o processamento da mensagem, o Agente Servidor pode tomar duas ações. Se a mensagem é um **INVITE**, ele muda de estado. Se a mensagem for um **OPTIONS**, ele responde automaticamente a esta mensagem. Se a mensagem for qualquer outra, o Agente Servidor responde com um 400 (*Bad Request*), pois no estado corrente ele só pode aceitar **INVITE** e **OPTIONS**.

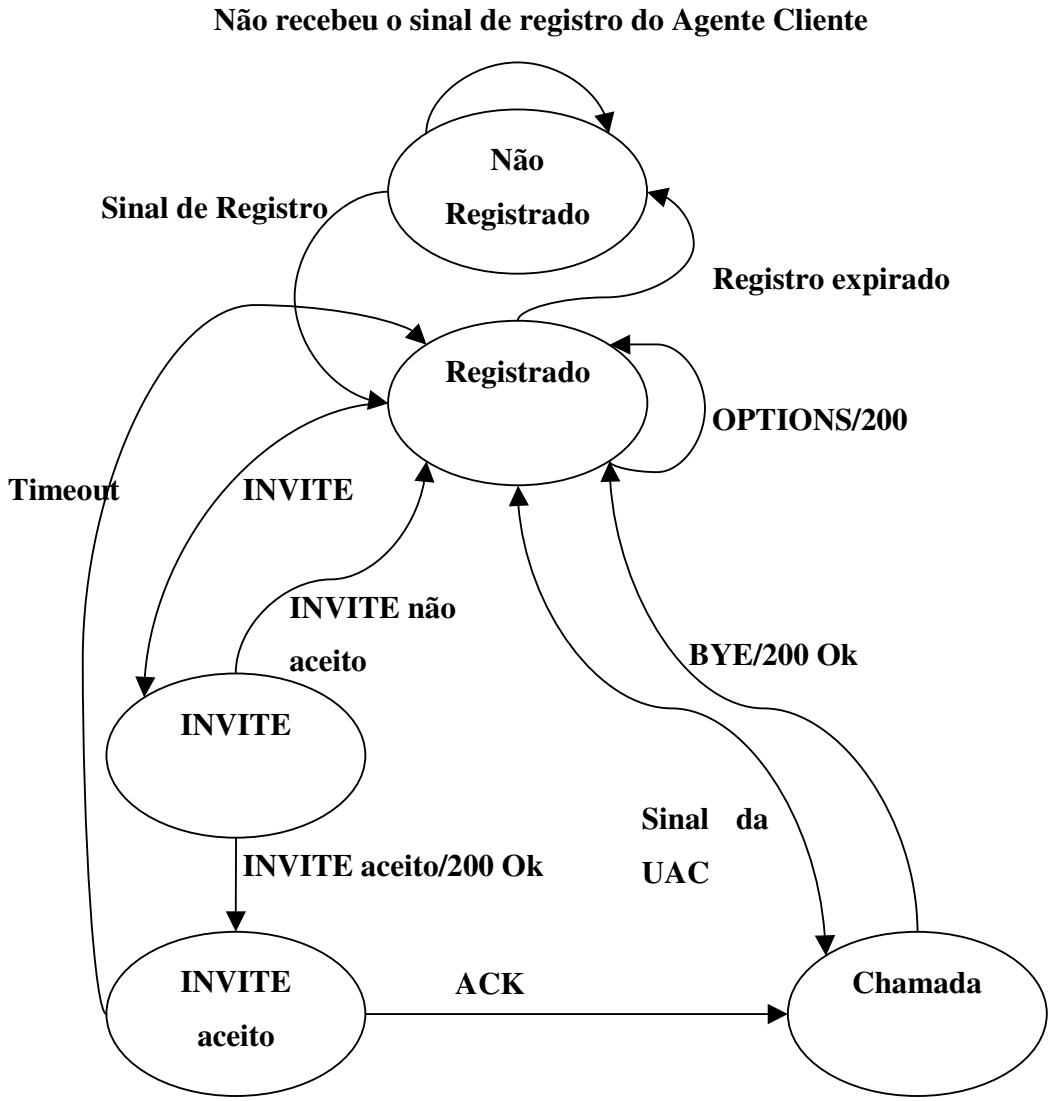

Figura 34. Máquina de estados do Agente Servidor do Usuário

Quando a mensagem é um **INVITE**, o Agente Servidor muda para o estado **INVITE**. Neste estado, o usuário é alertado do recebimento de um pedido de chamada e o Agente Servidor fica esperando o usuário responder. Se o usuário aceitar o pedido, o Agente Servidor vai para o estado **INVITE Aceito**; se não aceitar, o Agente Servidor responde com um 488 (*Not Acceptable Here*).

Quando o **INVITE** é aceito pelo usuário, o Agente Servidor vai para o estado **INVITE Aceito**, onde o Agente Servidor gera a resposta 200 Ok. Esta resposta estabelece um **Diálogo**. O Agente Servidor, então, monta todas as peças do Diálogo, isto é, salva todos os parâmetros da mensagem que identificarão a sessão a ser estabelecida. Após o estabelecimento do **Diálogo**, o Agente Servidor envia a mensagem para a transação que recebeu a mensagem **INVITE**, ou seja, a Transação Servidora INVITE. Após o envio da resposta, o Agente Servidor fica aguardando o recebimento de uma confirmação (ACK). Se o **ACK** não chegar, o Agente Servidor retransmite a resposta em intervalos de tempo de 0.5 seg, 1 seg, 2 seg, 4 seg, 4 seg, ..., até o tempo total de retransmissão se tornar 32 segundos. Se, em 32 segundos, nenhum **ACK** chegar, o Agente Servidor inicia o encerramento da sessão, solicitando, do Agente Cliente, a finalização da sessão, isto é, o envio de uma mensagem **BYE**. O Agente Servidor, então, volta para o estado **Registrado** e destrói o **Diálogo** criado.

Se uma mensagem **ACK** chegar enquanto o Agente Servidor estiver no estado **INVITE Aceito**, o Agente Servidor muda para o estado **Chamada**, inicia o fluxo de áudio e avisa, ao Agente Cliente, que o Terminal SIP está em uma chamada. Neste estado, para qualquer requisição que vier fora do **Diálogo**, o Agente Servidor responde com uma resposta 486 (*Busy Here*). Neste estado, se o Agente Servidor receber um **BYE** do outro Terminal SIP participante da sessão, ele pára o fluxo de áudio, destrói a sessão, responde com um **200 Ok** e volta para o estado inicial, isto é, o estado **não-Registrado**.

A Figura 34 mostra, ainda, uma transação do estado **Registrado** para o estado **Chamada** causada por sinais vindos do Agente Cliente. A mudança do estado **Registrado** para o estado **Chamada** é causada quando o Agente Cliente envia um sinal indicando que o Terminal SIP está em uma sessão efetuada por ele. A mudança do estado **Chamada** para o estado **Registrado** é causada por outro sinal do Agente Cliente indicando que o Terminal SIP deseja encerrar a chamada.

5.2.2 Camada de Transação

A Camada de Transação é a que gerencia as transações SIP. Uma transação SIP consiste de uma única requisição e quaisquer respostas para aquela requisição, onde as

respostas compreendem de zero ou mais respostas provisionais (**1xx**) e uma ou mais respostas finais (**2xx-6xx**). A transação possui duas partes: a parte cliente e a parte servidora. A parte cliente é conhecida como transação cliente e a parte servidora é conhecida como transação servidora. A transação cliente envia a requisição e a transação servidora envia as respostas.

Quando a camada de Transação do Usuário deseja iniciar uma nova transação, ela cria uma transação cliente e passa a requisição SIP, um endereço IP, a porta e o protocolo de transporte que será usado para enviar a requisição. A transação cliente, então, inicia a sua execução. No lado servidor, quando uma requisição é recebida, uma transação servidora é criada. Esta transação servidora é responsável pela entrega da requisição à camada de Transação do Usuário e a transmissão confiável das respostas.

Os dois tipos de transação, cliente e servidora, dependem se a mensagem é uma requisição INVITE ou não. Por causa disto, a Camada de Transação é subdividida em quatro subcamadas:

- Transação Cliente INVITE;
- Transação Cliente não-INVITE;
- Transação Servidora INVITE; e
- Transação Servidora não-INVITE.

5.2.2.1 Transação Cliente INVITE

A transação cliente INVITE é aquela que manuseia a transmissão da mensagem INVITE. A transação cliente envia um INVITE, a transação servidora responde ao INVITE recebido e a transação cliente envia um ACK, se a resposta à transação é uma resposta final de erro (**3xx-6xx**). A Figura 35 modela o comportamento da camada de transação cliente INVITE implementado.

Esta máquina de estados depende do protocolo a ser utilizado. Em nossa implementação, o protocolo utilizado é o UDP para a transmissão das mensagens.

O estado inicial, **CHAMANDO**, é acionado quando a Transação do Usuário inicia uma nova transação cliente INVITE, com uma mensagem INVITE. A transação cliente deverá passar a mensagem para a camada de transporte, que fará a transmissão da

mensagem. A partir deste ponto, a transação cliente inicia dois *timers*, um Timer A, com um valor inicial de 0,5 segundos, e um Timer B, com o valor de 32 seg. Se o Timer A estourar e nenhuma resposta, seja provisional, seja final, não for recebida, então a mensagem será retransmitida. A cada estouro de tempo, o Timer A é reconfigurado com o dobro do seu valor antigo, então, a mensagem deverá ser retransmitida nos seguintes intervalos de tempo: 0.5 seg, 1 seg, 2 seg., 4 seg., 8 seg.,... Se o Timer B estourar e a transação cliente ainda estiver no estado inicial, a transação é finalizada e um sinal de **Timeout** é enviado para a Transação do Usuário (Agente Cliente do Usuário). Se, neste estado, a transação cliente receber uma resposta com o código de status entre **3xx – 6xx**, a transação cliente vai para o estado **COMPLETADO**. Se receber uma resposta **2xx**, vai para o estado **TERMINADO**, e a transação é finalizada. Independente do código de status (**2xx – 6xx)**, a resposta final recebida é enviada para a Transação do Usuário.

Se a transação cliente receber uma resposta provisional no estado **CHAMANDO**, ela muda para o estado **PROCEDENDO**. Neste estado, a transação cliente não retransmite mais a requisição. Ela passa a resposta provisional recebida para a Transação do usuário e espera uma resposta da outra parte. Quaisquer outras respostas provisionais recebidas são passadas para a Transação do Usuário. Como no estado **CHAMANDO**, se a transação cliente receber uma resposta com o código de status entre **3xx – 6xx**, a transação cliente vai para o estado **COMPLETADO**. Se a transação cliente receber uma resposta com código de status **2xx**, a transação vai para o estado **TERMINADO**.

No estado **COMPLETADO**, a transação cliente é quem responde com um **ACK** para a resposta recebida. Neste caso, a mensagem é enviada e a transação inicia um Timer D de 32 segundos. Neste período, a transação retransmite o **ACK** para cada retransmissão da resposta que ela receber. Após o término do Timer D, a transação vai para o estado **TERMINADO**.

No estado **TERMINADO**, a transação cliente é destruída.

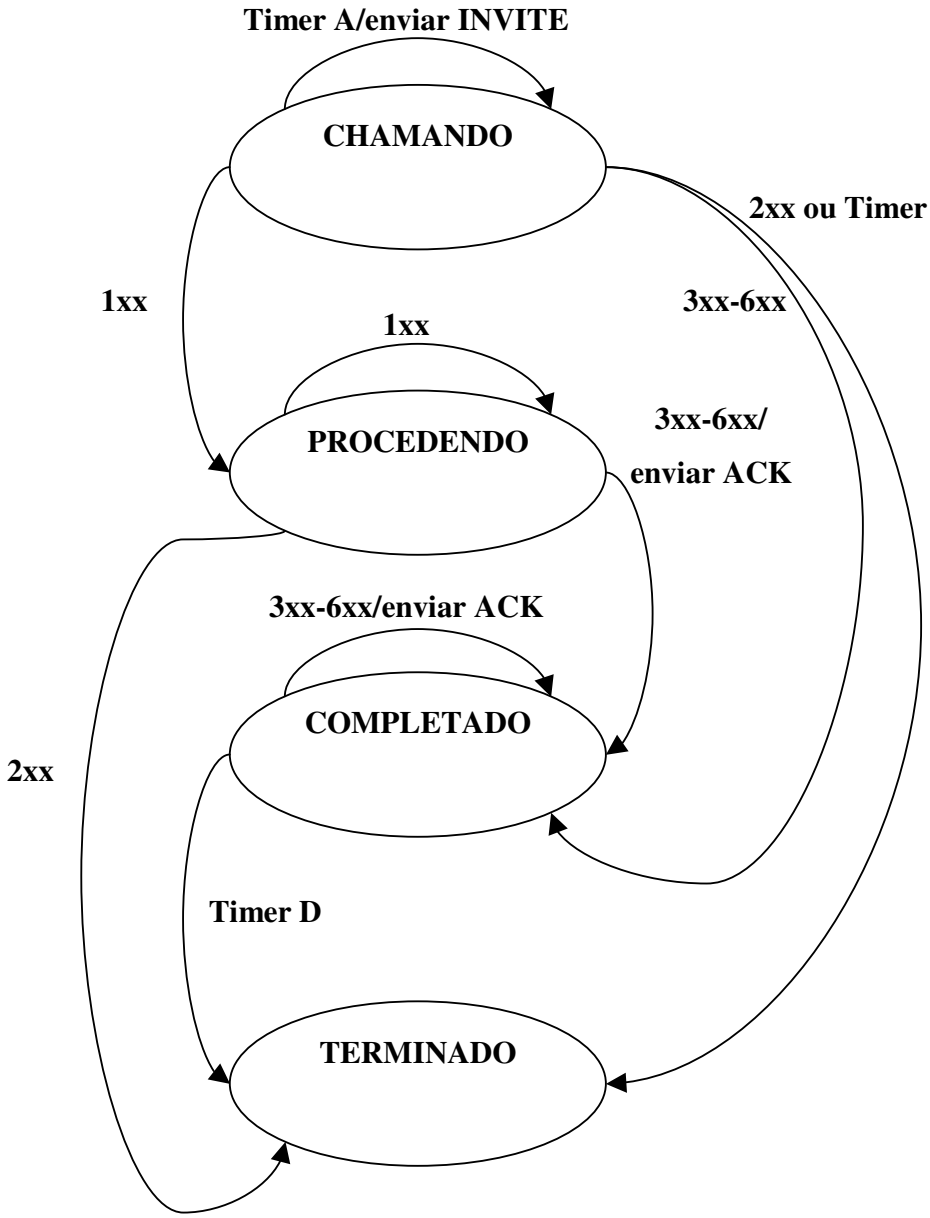

Figura 35. Máquina de estados da Transação Cliente INVITE

5.2.2.2 Transação Cliente não-INVITE

As transações cliente não-INVITE não fazem uso de um **ACK**. Elas são interações simples de requisição e resposta. Para o UDP, as requisições são retransmitidas

em intervalos que se iniciam em 0.5 seg. e dobram de valor até alcançar 4 seg. Se uma resposta provisional for recebida, as retransmissões da mensagem continuam em intervalos de 4 seg. A Figura 36 modela o comportamento implementado da transação cliente não-INVITE.

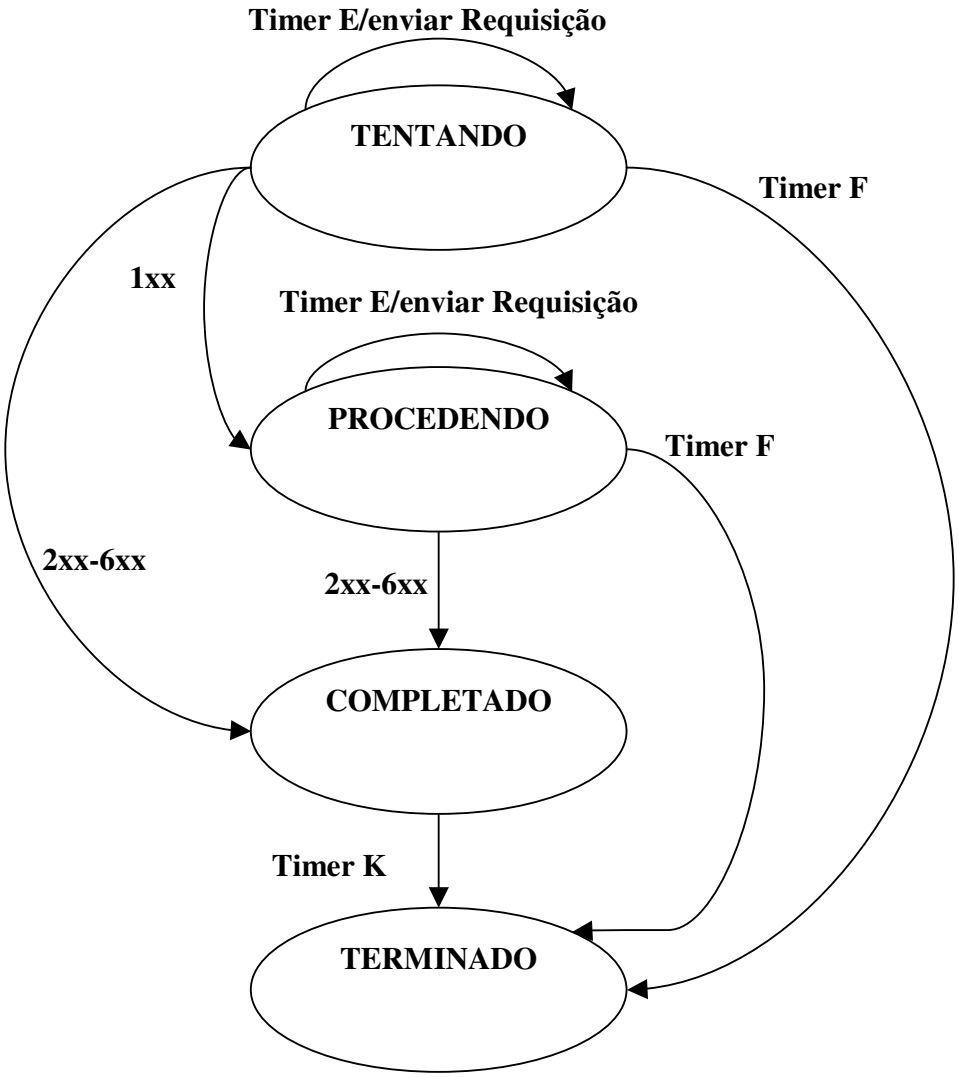

Figura 36. Máquina de estados da Transação Cliente não-INVITE

A máquina de estados para a transação cliente não-INVITE é muito similar a da transação cliente INVITE. Quando a transação cliente não-INVITE é criada, ela entra no estado **TENTANDO**. Neste estado, a transação cliente inicia um *timer*, o Timer F, com o

valor de 32 seg. e envia a requisição para a Camada de Transporte. Como o protocolo utilizado é o UDP, a transação cliente inicia um *timer*, o Timer E, com o valor inicial de 0.5 segundos. Se o Timer E estourar e nenhuma resposta for recebida neste estado, a transação cliente retransmite a mensagem e atualiza o valor do Timer E para o dobro do valor antigo. A cada retransmissão subseqüente, o valor do Timer E é atualizado para o dobro do valor antigo até que o valor do Timer E alcançe 4 segundos. Ao atingir este valor, o Timer E não é mais atualizado, e as retransmissões, a partir deste ponto, são feitas em intervalos de 4 segundos. Com isso, os intervalos de retransmissão resultam em 0.5 seg., 1 seg., 2 seg., 4 seg., 4 seg., ..., e assim por diante.

Se o Timer F estourar enquanto a transação cliente estiver no estado **TENTANDO**, a transação cliente deve informar a Transação do Usuário de um **Timeout** e deverá entrar no estado **TERMINADO**. Se uma resposta provisional é recebida enquanto a transação está no estado **TENTANDO**, a resposta é passada para a Transação do Usuário e a transação cliente vai para o estado **PROCEDENDO**. Se uma resposta final (código de status entre **2xx – 6xx**) é recebida enquanto a transação está no estado **TENTANDO**, a transação cliente passa a resposta para a Transação do Usuário e muda para o estado **COMPLETADO**.

No estado **PROCEDENDO**, a transação cliente não desabilita os *timers*. Se o Timer E estourar, a mensagem é retransmitida e o Timer E é atualizado para o valor de 4 segundos. Se o Timer F estourar neste estado, a transação cliente informa à Transação do Usuário que ocorreu um estouro de tempo e vai para o estado **TERMINADO**. Se uma resposta final chegar enquanto a transação ainda está no estado **PROCEDENDO**, a resposta é passada para a Transação do Usuário e a transação cliente vai para o estado **COMPLETADO**.

No estado **COMPLETADO**, a transação cliente inicia um *timer*, o Timer K, de 5 segundos. Este estado existe para armazenar qualquer retransmissão da resposta que pode ser recebida. Quando o Timer K estourar, a transação cliente vai para o estado **TERMINADO**. No estado **TERMINADO**, a transação cliente não-INVITE é destruída.

5.2.2.3 Transação Servidora INVITE

A transação servidora INVITE é criada quando uma requisição INVITE chega ao Terminal SIP. A Figura 37 modela o comportamento da camada de transação servidora INVITE implementado.

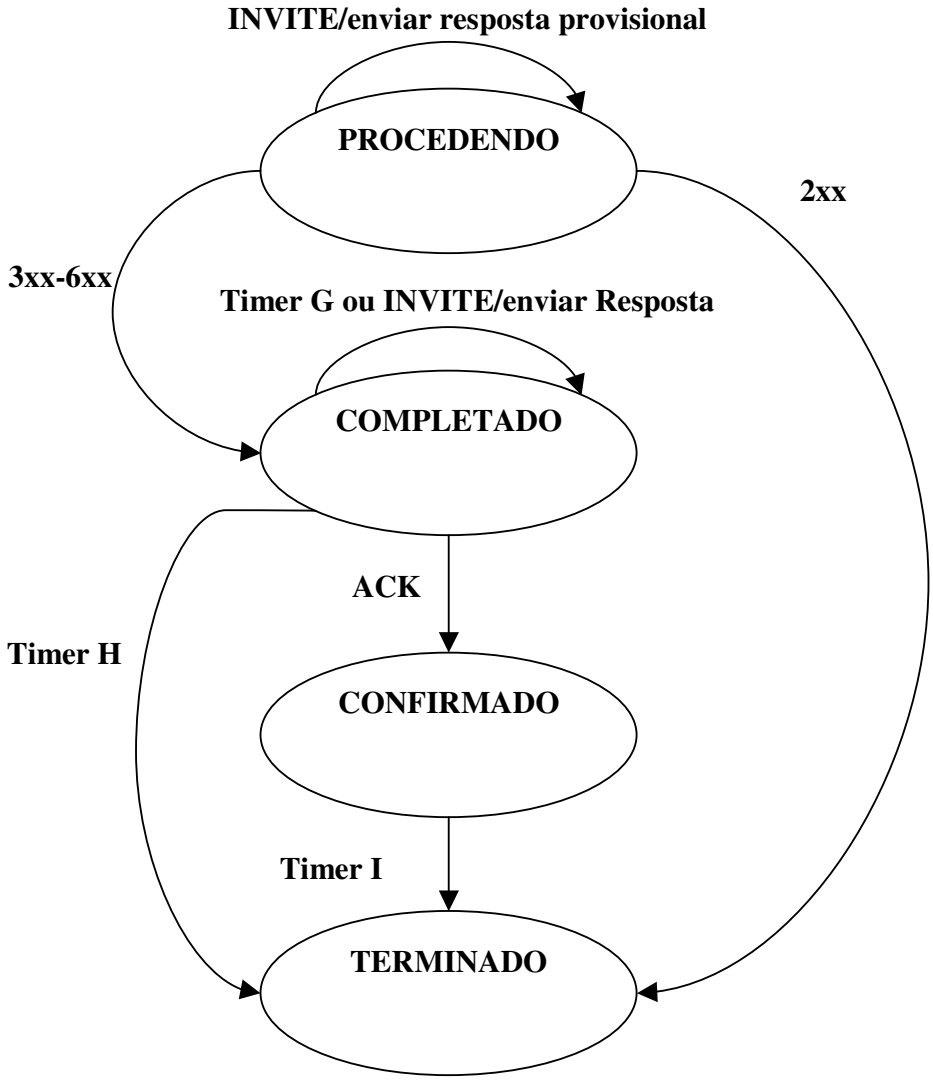

Figura 37. Máquina de Estados da Camada de Transação Servidora INVITE

Quando a transação servidora é construída com a requisição INVITE, ele inicia no estado **PROCEDENDO**. Neste estado, a transação servidora gera uma resposta **100** (*Trying*). Esta resposta provisional é passada para a Camada de Transporte para efetivar a Transmissão. A Transação do Usuário, ao receber a requisição, pode responder com uma

resposta com código de status entre **100 – 199**. A transação servidora transmite a mensagem, mas não muda de estado. Se uma retransmissão da requisição é recebida enquanto a transação servidora está no estado **PROCEDENDO**, a resposta provisional mais recente enviada pela Transação do Usuário ou a resposta provisional gerada pela transação servidora, caso a Transação do Usuário não tenha gerado nenhuma resposta provisional, é enviada. Se, enquanto a transação servidora está no estado **PROCEDENDO**, a Transação do Usuário responde com uma resposta com o código de status **2xx**, a transação servidora envia a resposta e vai para o estado **TERMINADO**. Se, a transação servidora receber da Transação do Usuário uma resposta com o código de status entre **300 – 699**, a resposta deve ser passada para a Camada de Transporte e a transação servidora vai para o estado **COMPLETADO**.

No estado **COMPLETADO**, a transação servidora inicia dois *timers*; o Timer G, com o valor inicial de 0.5 segundos, e o Timer H, com o valor de 32 segundos. Se o Timer G estourar, a resposta é repassada para a Camada de Transporte para a retransmissão e o valor do Timer G é atualizado para o dobro do seu valor antigo e ele poderá crescer até atingir o valor de 4 segundos. As retransmissões aqui são como na transação cliente não-INVITE. Se o Timer H estourar enquanto a transação servidora ainda está neste estado, a transação servidora indica para a Transação do Usuário do **Timeout** e vai para o estado Terminado. Se a transação servidora receber uma confirmação (ACK) enquanto ainda está no estado **COMPLETADO**, a transação muda para o estado **CONFIRMADO**.

No estado **CONFIRMADO**, a transação servidora inicia um *timer*, o Timer I, com o valor de 5 segundos. Este estado é para absorver as possíveis retransmissões da confirmação ACK. Uma vez que o Timer I estourar, a transação servidora vai para o estado **TERMINADO**.

No estado **TERMINADO**, a transação servidora é destruída.

5.2.2.4 Transação Servidora não-INVITE

A transação servidora não-INVITE é criada quando uma requisição, que não seja a INVITE ou a ACK, chega ao Terminal SIP. A máquina de estados desta transação servidora é mostrada na Figura 38.

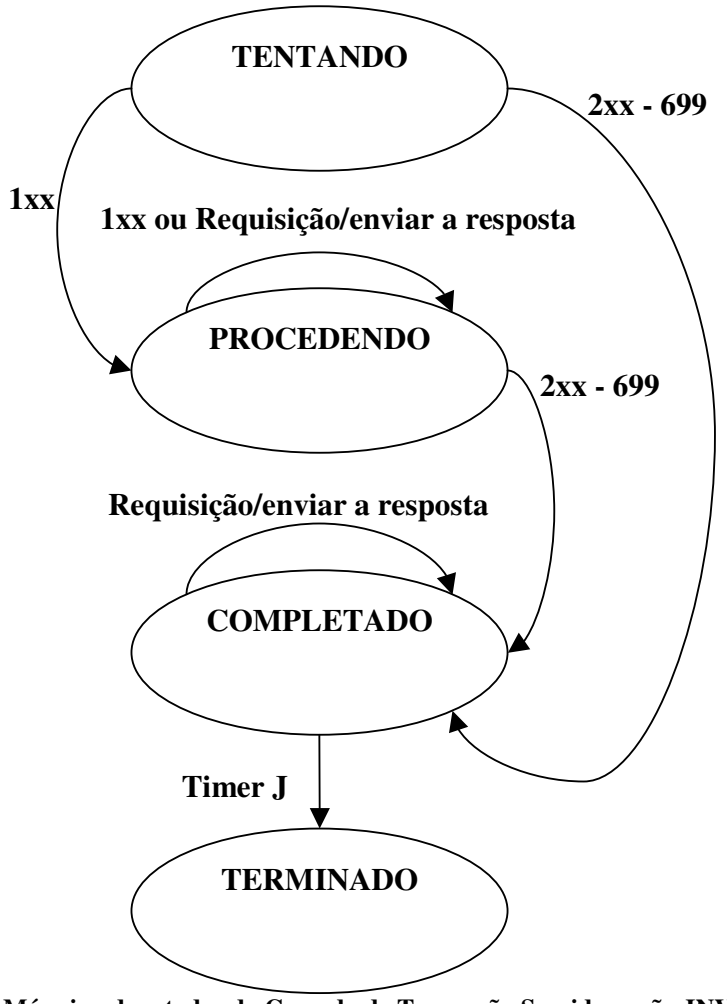

Figura 38. Máquina de estados da Camada de Transação Servidora não-INVITE

A transação servidora não-INVITE é iniciada no estado **TENTANDO**. Neste estado, a transação passa a requisição para a Transação do Usuário e fica aguardando uma resposta dela. As retransmissões das mensagens recebidas, enquanto a transação está neste estado, são descartadas. Se a Transação do Usuário passar uma resposta provisional

para a transação servidora enquanto ela está no estado **TENTANDO**, a transação servidora envia a mensagem e passa para o estado **PROCEDENDO**. Se a transação servidora receber uma mensagem com código de status entre **200 – 699**, a transação vai para o estado **COMPLETADO**.

No estado **PROCEDENDO**, se a Transação do Usuário enviar outras respostas provisionais, a camada de transação retransmite todas as respostas provisionais e não muda de estado. Se a transação servidora receber uma retransmissão da requisição, enquanto ela está no estado **PROCEDENDO**, a transação servidora retransmite a última resposta recebida da Transação do Usuário. Se a Transação do Usuário passar uma resposta com código de status entre **200 – 699**, a transação vai para o estado **COMPLETADO** e envia a resposta recebida pela Transação do Usuário.

No estado **COMPLETADO**, a transação servidora inicia um *timer*, o Timer J, com o valor de 32 segundos. Neste estado, se a transação servidora receber qualquer retransmissão da requisição, ele envia a resposta final recebida da Transação do Usuário. Quando o Timer J estourar, a transação servidora vai para o estado **TERMINADO**.

No estado **TERMINADO**, a transação servidora é destruída.

5.2.3 Camada de Transporte

A Camada de Transporte é responsável pela transmissão das requisições e respostas através da rede. Esta camada é responsável pelo gerenciamento dos protocolos de rede utilizados e pela entrega das mensagens à sua transação apropriada. A Camada de Transporte também é dividida em duas subcamadas.

- **Transporte Cliente**: A parte cliente da Camada de Transporte é responsável por enviar a requisição e receber as respostas. Quando uma Transação Cliente ou uma Transação do Usuário deseja enviar uma mensagem através da Camada de Transporte, eles informam o protocolo de transporte, o endereço IP, a porta e a mensagem a ser enviada.
- **Transporte Servidor**: A parte servidora da Camada de Transporte é responsável por receber as requisições e enviar as respostas repassadas pelas camadas superiores. Quando uma requisição é recebida, a Camada de

Transporte determina, através do método, para qual camada ela deve enviar a requisição.

5.2.4 Camada de Acesso à Placa de Áudio

Esta camada do Terminal SIP é responsável pela comunicação do Terminal SIP com a placa de áudio. Esta comunicação inclui o encaminhamento de um pacote recebido da placa de áudio e o envio do mesmo através da rede, o recebimento de um pacote de áudio através da rede e o encaminhamento do mesmo para a placa de áudio, e o acesso às configurações do dispositivo de áudio. Esta camada não é especificada nas RFCs que descrevem o protocolo SIP.

Com a placa de áudio devidamente iniciada, o Terminal SIP pode obter os seus modos de operação e configurá-los. Pode, também, ler e escrever pacotes de áudio no dispositivo.

O modo de operação da placa de áudio é caracterizado, principalmente, por quatro configurações básicas:

- **Taxa de amostragem**: As placas de áudio mais antigas permitiam, somente, uma taxa de amostragem máxima de 22050 Hz. Hoje, as placas mais novas podem ser configuradas com uma taxa de amostragem de até 96 kHz. Para uma chamada telefônica, o valor padrão da taxa de amostragem é de 8 kHz. O programa permite ao usuário alterar a taxa de amostragem desde 8 kHz até o valor máximo permitido pela placa de áudio.
- **Número de bits por amostra**: A placa de áudio permite duas configurações para o número de bits: 8 bits e 16 bits. Além disso, permite o usuário selecionar o tipo de compressão usada: Lei A ou Lei μ. 8 bits é usado para as conversações telefônicas e jogos com baixa resolução de áudio. 16 bits é usado para música em geral. A configuração padrão do Terminal SIP é de 8 bits.
- **Número de canais**: Esta configuração depende da placa de som. A configuração *default* do Terminal SIP é de apenas um canal.

- **Tamanho do fragmento**: Esta configuração é que determina o tempo de empacotamento do pacote de áudio. Este tempo é dado pela relação: $t_E = \frac{F \times 8}{B} seg.$, onde : F é o tamanho do fragmento em bytes e B é a taxa de transmissão em bps. Como exemplo, pode-se considerar a configuração padrão das placas de áudio: B = 64 kbps (8 bits por amostra com 8 KHz de taxa de amostragem), e F = 2048 bytes. Portanto, o tempo de empacotamento será de 256 mseg. Este tempo, em redes locais, é o comportamento predominante do atraso sofrido pelo pacote entre dois terminais SIP. As configurações possíveis para este parâmetro dependem da placa de áudio. Algumas placas de áudio não permitem que se mude o valor padrão. Outras permitem uma faixa de valores, iniciando-se em 16 bytes até 2048 bytes, em intervalos múltiplos de 2, isto é, 16, 32, 64, 128,..., 2048 bytes.

Cada tipo de configuração define um modo de operação da placa de áudio. Por exemplo, o modo padrão de operação da placa de áudio é com uma taxa de amostragem de 8 kHz, mono canal, amostras de 8 bits e tamanho do fragmento de 2048 bytes.

Tipicamente, as placas de som já vêm com algum *codec* instalado. As placas de som mais antigas da Creative possuíam a família de *codecs* do ITU-T. Se a placa possuir algum *codec* instalado, esta camada permite o acesso, por parte do programa, a este *codec*.

5.3 Servidor de Registros

Um Servidor de Registros é um Agente Servidor do Usuário que responde somente às requisições **REGISTER** e **OPTIONS**, e que mantém uma lista de ligações que são acessíveis para os Proxies dentro do seu domínio de rede.

O Servidor de Registros implementado, quando recebe uma requisição **REGISTER**, segue os seguintes passos de processamento:

1. Inspeciona o domínio ao qual o usuário pertence para ver se ele é responsável por este domínio. Se o Servidor de Registros não for o responsável pelo domínio, ele envia a mensagem para o domínio especificado na requisição.

2. O Servidor de Registros, após verificar que é responsável pelo domínio ao qual o usuário pertence, extrai o endereço de registro contido no campo **To** e verifica se o campo tem um endereço de registro válido para aquele domínio.
3. O Servidor de Registros verifica o campo **Contact** para extrair os endereços para a ligação com o endereço de registro neste servidor. Se o endereço de registro do campo **To** já existe no servidor, ele verifica cada valor do campo **Contact**. Se o campo **Contact** tiver o valor igual a *, significa que o contato deve ser removido do servidor de registros. Se o valor for um valor previamente armazenado, ele atualiza os tempo de expiração do registro. Se o campo **To** contiver um endereço que não existe no servidor, ele extrai o valor do campo **Contact** para efetuar um novo registro.
4. O Servidor de Registros processa o campo **Expires** da mensagem.
5. O Servidor de Registros extrai, também, o valor dos campos **Call-ID** e **CSeq**.
6. O Servidor de Registros, então, responde a requisição com a resposta **200 Ok**. A resposta conterá um campo **Contact**, que inclui os endereços registrados.

5.4 Proxy

A outra entidade SIP desenvolvida foi o Proxy. Os proxies são entidades SIP que roteiam as requisições para os agentes servidores e as respostas SIP para os agentes clientes. As requisições podem atravessar vários proxies até o seu destino. Cada Proxy deverá fazer as suas decisões de roteamento, modificando a requisição antes de enviá-la para o próximo elemento. As respostas seguem o mesmo caminho que as requisições, isto é, passando pelos mesmos proxies que a requisição passou.

O Proxy implementado, quando recebe uma requisição executa algumas funções de processamento:

- **Valida a Requisição**: A requisição, antes de ser passada para um Terminal SIP, deve passar por alguns testes no Proxy. Estes testes verificam se a requisição está bem formada, se o endereço de destino da mensagem é um endereço SIP válido e se o campo **Max-Forwards** é maior que zero,

indicando que a mensagem já atravessou o número máximo de entidades SIP e deve ser descartada.

- **Determinar o Alvo da Requisição**: O destino da requisição é verificado pelo Proxy ou através de consulta no servidor de registros ou através do conteúdo da requisição. Se o Proxy rotear a mensagem através do conteúdo da mensagem, ele escolhe um destino, dos vários que possam conter na mensagem, para enviar.
- **Atualizar a mensagem e enviá-la**: Após determinar o destino da mensagem, o Proxy adiciona o campo **Via**, contendo o seu endereço e a porta, e atualiza o valor do campo **Max-Forwards**, decrementando o seu valor de uma unidade. Após este passo, o Proxy envia a mensagem.

5.5 Funcionamento do Sistema SIP Implementado

O Sistema SIP implementado é composto de entidades que utilizam o protocolo SIP para prover funcionalidades. O Terminal SIP é a parte do Sistema que estabelece sessões, o Proxy é a entidade que envia as mensagens para o destinatário, e o Servidor de Registros é a entidade responsável pelo registro dos usuários no Sistema e por fornecer informação para o envio das mensagens pelo Proxy.

A sinalização, neste Sistema, é um pouco diferente da sinalização SIP tradicional (seção 3.6). A Figura 39 mostra como é feita em nossa implementação.

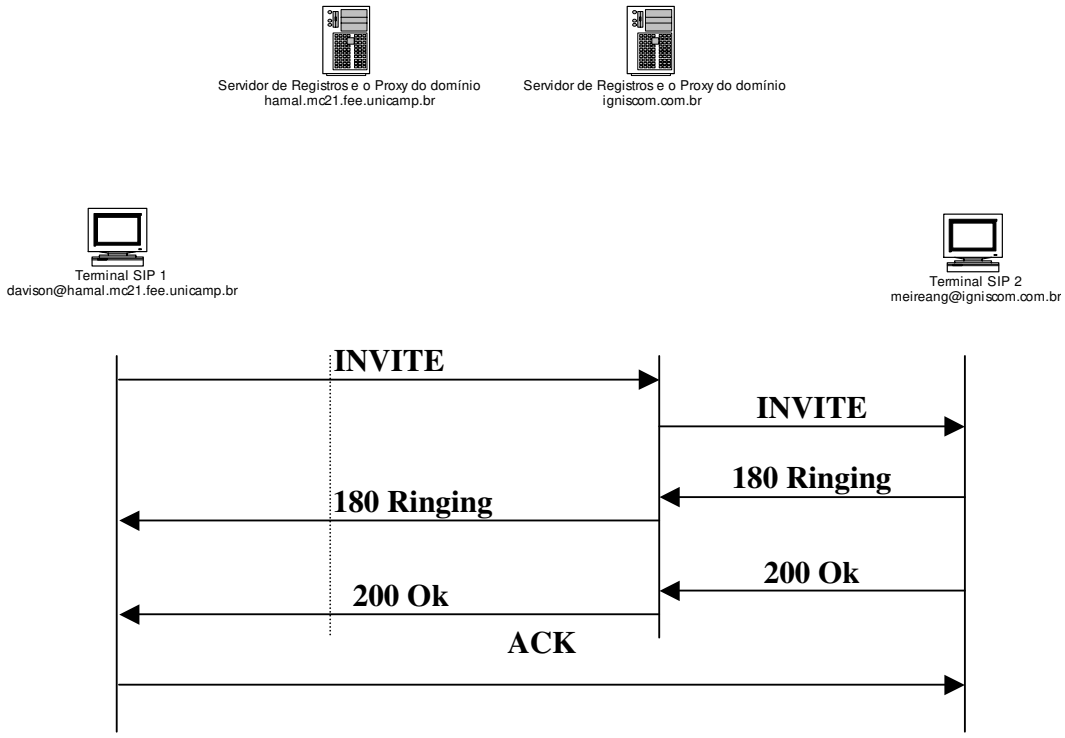

Figura 39. Sinalização INVITE no Sistema SIP

Diferente da sinalização SIP tradicional, os Proxies não tomam parte da negociação, com isso, o caminho das mensagens SIP pode ser diferente daquele apresentado na seção 3.6.2. Nesta implementação, as mensagens SIP sempre são enviadas para o Proxy do domínio do destinatário da mensagem, e não para o Proxy do domínio do remetente da requisição. Este esquema diminui o processamento nos Proxies e o número de entidades SIP que a mensagem atravessa. As respostas, então, atravessam o mesmo caminho que a requisição. Uma vez que uma sessão SIP está estabelecida entre os Terminais, os Proxies não participam mais da negociação entre as entidades.

5.6 Terminal SIP – Interface Gráfica

A Interface gráfica do Terminal SIP é mostrada na Figura 40. Nesta figura, podem ser vistos dois menus: **Principal** e **Opções de Áudio**. No menu principal, estão as funcionalidades do protocolo SIP, isto é, o estabelecimento de uma sessão, o registro, a finalização de uma chamada e o encerramento do programa.

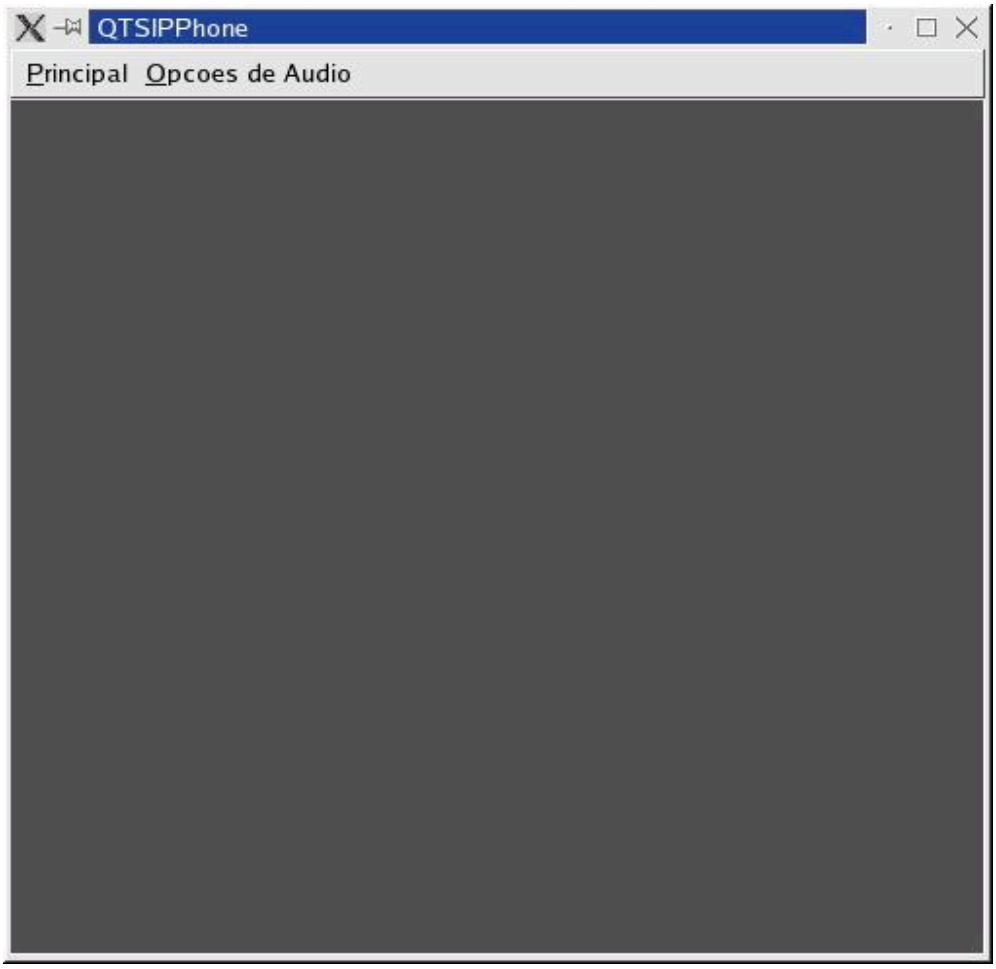

Figura 40. Interface gráfica do terminal SIP

Quando um usuário deseja iniciar um registro, ele escolhe o menu **Principal** e depois a opção **Registro**. O programa pede para o usuário entrar com o endereço do servidor de registros e com o *username* que ele deseja se registrar (Figura 41).

Após o registro, o usuário pode iniciar uma sessão SIP. Para iniciá-la, o usuário seleciona o menu **Principal** e a opção **Conectar**. Será pedido o endereço SIP do terminal de destino, conforme mostra a Figura 42.

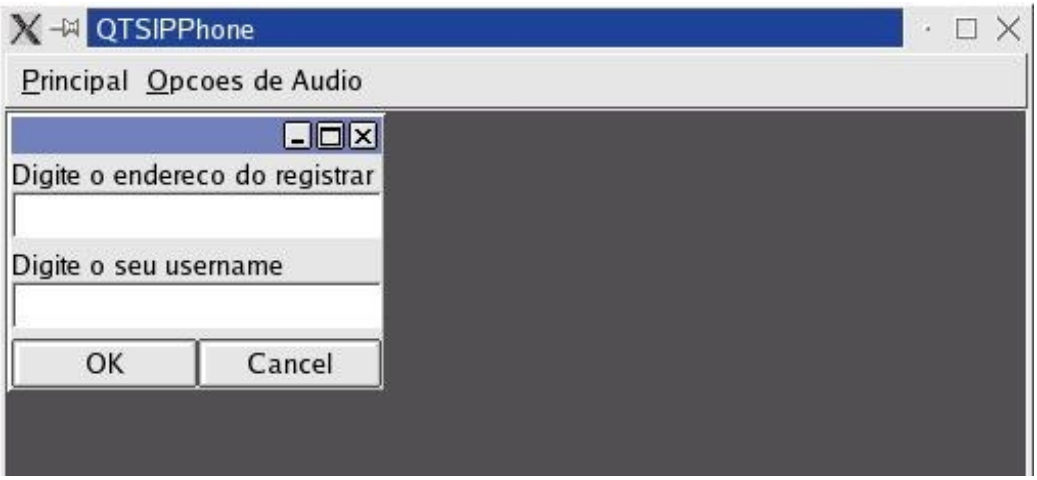

Figura 41. Tela de registro do Terminal SIP

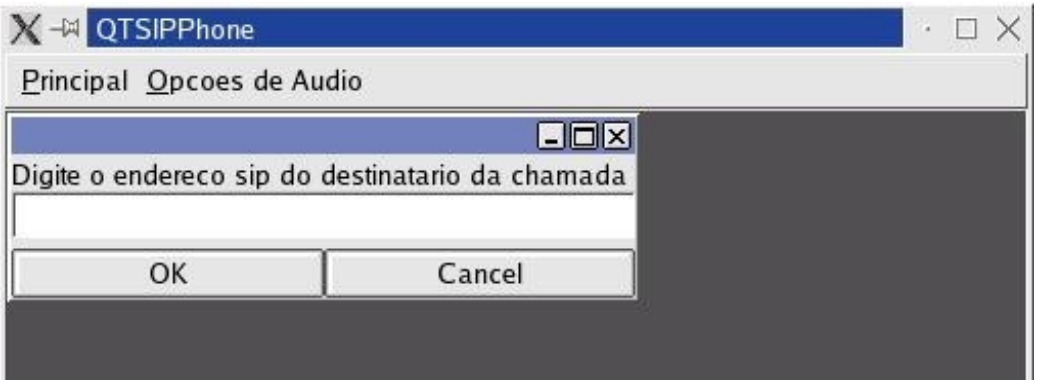

Figura 42. Tela para o estabelecimento de uma sessão SIP

Após o encerramento da sinalização, a sessão SIP está estabelecida e os Terminais SIP podem trocar pacotes de áudio. Para finalizar a sessão, o usuário clica em **Principal** e **Finalizar Chamada**. O programa pergunta se ele deseja finalizar a chamada (Figura 43). Se o usuário clicar em **Sim**, o programa gera a mensagem **BYE** e finaliza o fluxo de áudio.

O outro item presente no menu é **Opções de Áudio**. Neste item é possível obter informações da placa de áudio local e é possível alterar as configurações do dispositivo de áudio. Para obter informações sobre as configurações atuais do dispositivo de áudio, basta clicar em **Opções de Áudio** e **Configurações de Áudio** (Figura 44).

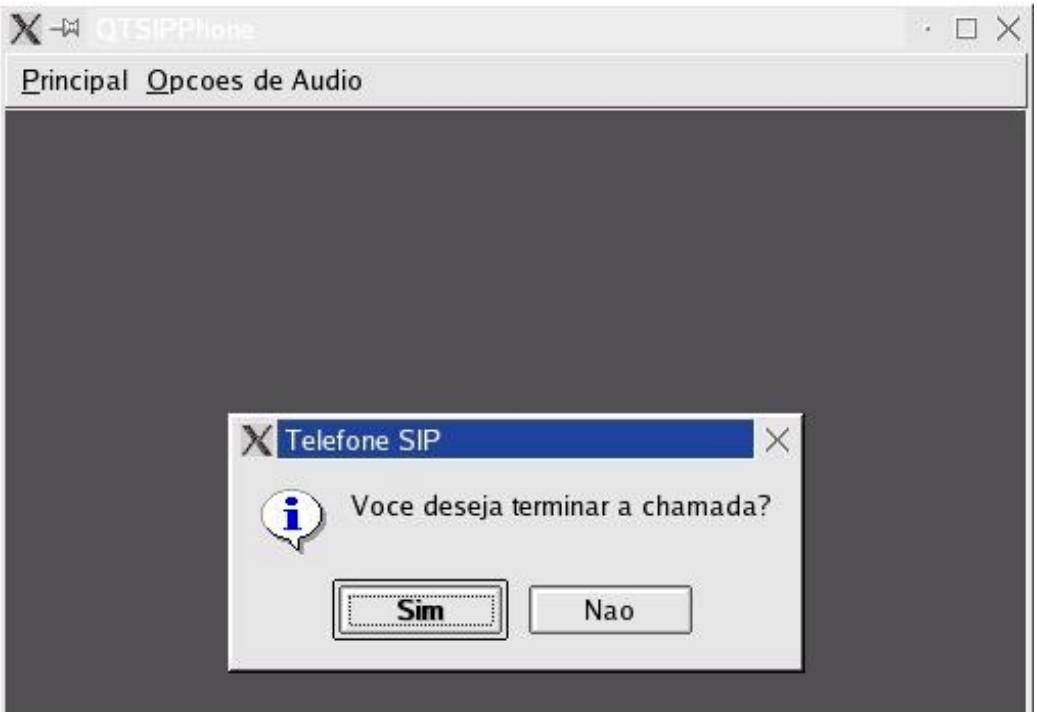

Figura 43. Tela do encerramento de uma sessão SIP

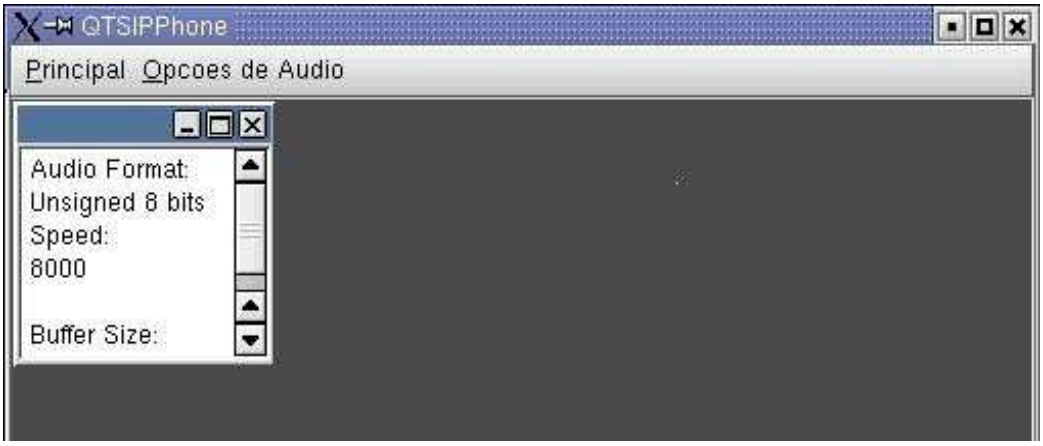

Figura 44. Tela da configuração da placa de som do Terminal SIP

Outro dado possível de se coletar do programa são os formatos suportados pela placa de som. O Terminal SIP disponibiliza, para o usuário, os formatos de áudio que a placa de som suporta. A Figura 45 mostra um exemplo de formatos suportados em um Terminal SIP.

Além da obtenção dos parâmetros do dispositivo de áudio, o Terminal SIP permite, ao usuário, alterá-los. A Figura 46 e a Figura 47 mostram as opções de configuração fornecidas pelo Terminal SIP.

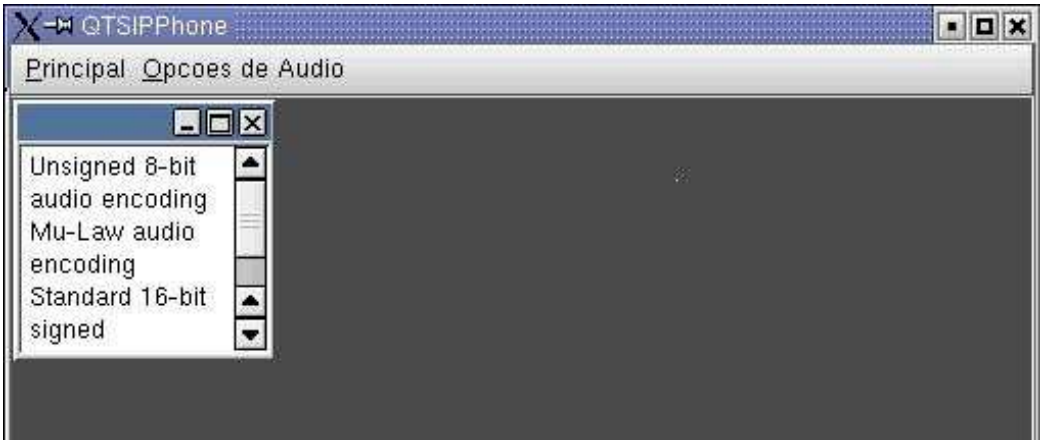

Figura 45. Tela com as informações dos formatos suportados pelo Terminal SIP

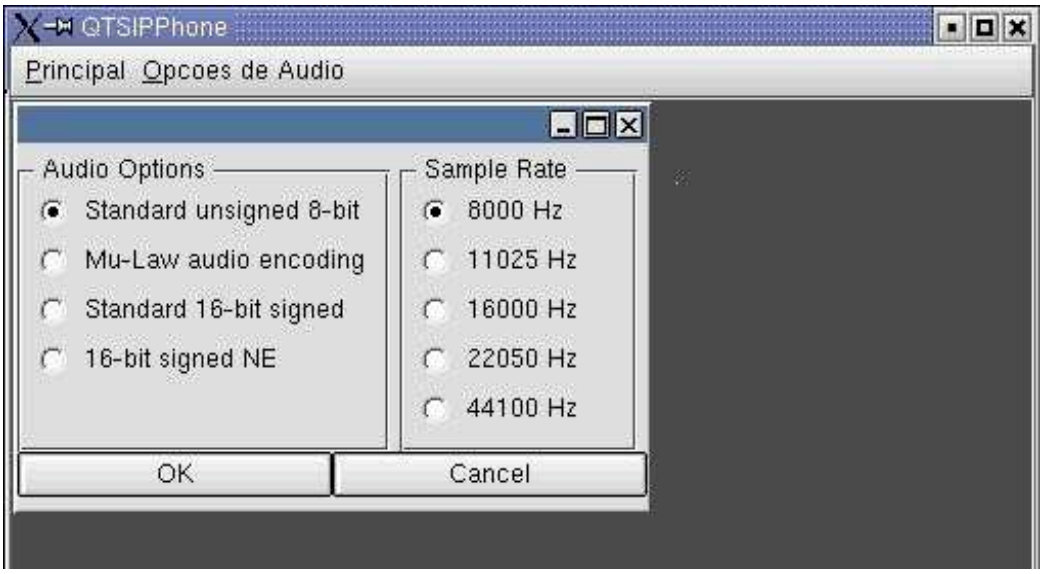

Figura 46. Tela de configuração do formato de áudio e da taxa de amostragem

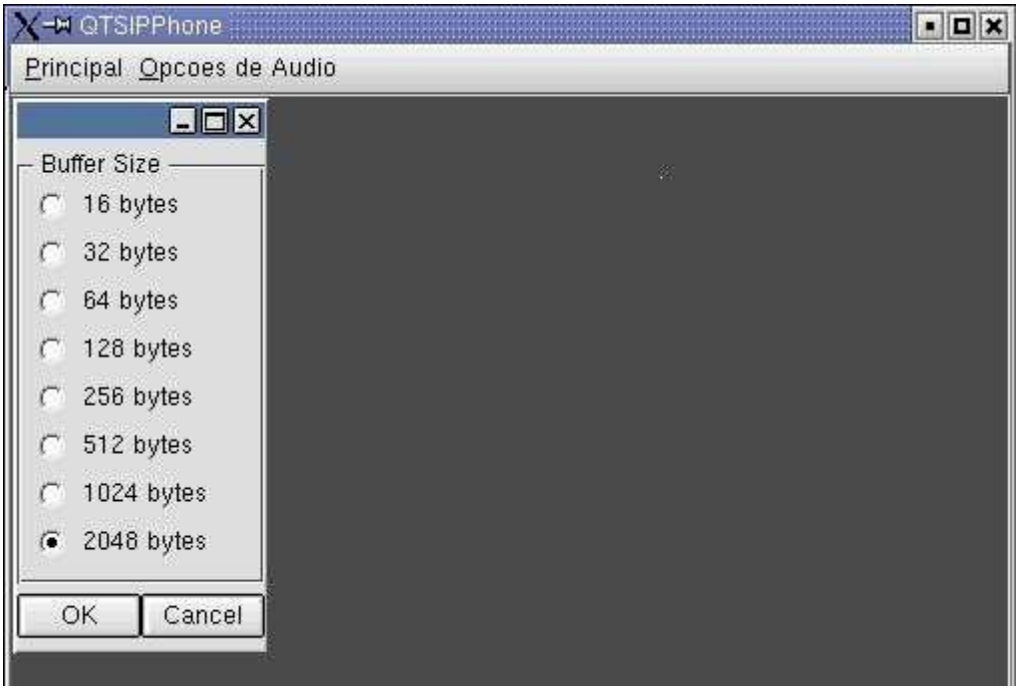

Figura 47. Tela de configuração do tamanho do fragmento

6. Exemplos do Funcionamento do Sistema SIP

6.1 Sinalização SIP

O cenário abaixo (Figura 48) foi montado no laboratório LARCOM para mostrar o funcionamento do Sistema SIP implementado.

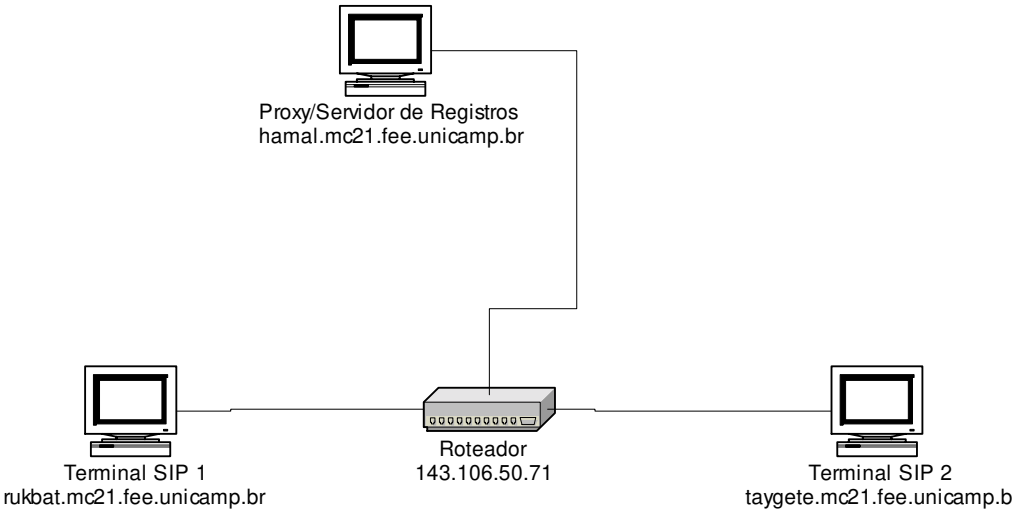

Figura 48. Sistema SIP montado no laboratório LARCOM

Neste cenário, os Terminais SIP 1 e 2 são dois computadores com processador Pentium II, 266 MHz e 196 MBytes de memória RAM. Estes computadores possuem uma distribuição do Linux instalada, o Mandrake 9.0. O Proxy/Servidor de Registros é um AMD 500 MHz, com 196 MBytes de memória RAM, e com o Mandrake 9.0 instalado.

6.1.1 Registro de um Terminal

O Registro do Terminal consiste do envio de uma mensagem **REGISTER** para o Servidor de Registros, e o recebimento de uma resposta **200 Ok**, confirmando o endereço Registrado.

Para o Terminal SIP 1, a sinalização resultou na seguinte troca de mensagens:

- **Terminal SIP 1 → Servidor de Registros**:

```
REGISTER sip:hamal.mc21.fee.unicamp.br SIP/2.0
Via: SIP/2.0/UDP rukbat.mc21.fee.unicamp.br:5060;branch=k9hG4kmmBJIROiHsPhw
Max-Forwards: 70
To: <sip:davison@hamal.mc21.fee.unicamp.br>
From: <sip:davison@hamal.mc21.fee.unicamp.br>;tag=bfvkLn5cYbE6Y
Call-ID:amn6xnL7yhSgQ
CSeq: 5965 REGISTER
Contact: <sip:davison@rukbat.mc21.fee.unicamp.br>
Content-Length: 0
```

- **Servidor de Registros → Terminal SIP 1:**

```
SIP/2.0 200 Ok
Via: SIP/2.0/UDP rukbat.mc21.fee.unicamp.br:5060;branch=k9hG4kmmBJIROiHsPhw;
received=143.106.50.84
Max-Forwards: 70
To: <sip:davison@hamal.mc21.fee.unicamp.br>
From: <sip:davison@hamal.mc21.fee.unicamp.br>;tag=bfvkLn5cYbE6Y
Call-ID: amn6xnL7yhSgQ
CSeq: 5965 REGISTER
Contact: <sip:davison@rukbat.mc21.fee.unicamp.br>
Content-Length: 0
```

Para o Terminal SIP 2, a negociação resultou em:

- **Terminal SIP 2 → Servidor de Registros**:

```
REGISTER sip:hamal.mc21.fee.unicamp.br SIP/2.0
Via: SIP/2.0/UDP taygete.mc21.fee.unicamp.br:5060;branch=k9hG4kbb0pZ1Q7.iaIs
Max-Forwards: 70
To: <sip:david@hamal.mc21.fee.unicamp.br>
From: <sip:david@hamal.mc21.fee.unicamp.br>;tag=rmPvLh1AgcZVc
Call-ID: ab/vwHDvfZIyc
CSeq: 6649 REGISTER
Contatc: <sip:david@taygete.mc21.fee.unicamp.br>
Content-Length: 0
```

- **Servidor de Registros → Terminal SIP 2:**

```
SIP/2.0 200 Ok
Via: SIP/2.0/UDP taygete.mc21.fee.unicamp.br:5060;branch=k9hG4kbb0pZ1Q7.iaIs;
received=143.106.50.74
Max-Forwards: 70
To: <sip:david@hamal.mc21.fee.unicamp.br>
From: <sip:david@hamal.mc21.fee.unicamp.br>;tag=rmPvLh1AgcZVc
Call-ID: ab/vwHDvfZIyc
CSeq: 6649 REGISTER
```

```
Contact: <sip:david@taygete.mc21.fee.unicamp.br>
Content-Length: 0
```

Observe que na mensagem não existe o campo **Expires**. Neste caso, o Servidor de Registros implementado assume que os registros deverão ser válidos por uma hora.

6.1.2 Estabelecimento de uma sessão

O estabelecimento de uma sessão é feito através da troca de mensagens **INVITE/200 Ok/ACK**(seção 5.5). O Terminal SIP 1 envia um **INVITE** para o Proxy do domínio do Terminal SIP 2. A sinalização completa é mostrada na seqüência abaixo:

- **Terminal SIP 1 → Proxy**

```
INVITE sip:david@hamal.mc21.fee.unicamp.br SIP/2.0
Via: SIP/2.0/UDP rukbat.mc21.fee.unicamp.br:5060;branch=k9hG4kbbaWFjvxx5tSI
Max-Forwards: 70
To: <sip:david@hamal.mc21.fee.unicamp.br>
From: <sip:davison@hamal.mc21.fee.unicamp.br>;tag=faygNT1lt/eLY
Call-ID: ..8o7ZxomUexo
CSeq: 6358 INVITE
Contact: <sip:davison@rukbat.mc21.fee.unicamp.br>
Content-Type: application/sdp
Content-Length: 166

v=0
user=davison 1234843216574 1944887216587 IN IP4 143.106.50.84
s=Session SDP
c=IN IP4 143.106.50.84/8000
t=0 0
m=audio 8000 udp/AVP 97
a=wb:97 U8/8000
```

- **Proxy → Terminal SIP 2**

```
INVITE sip:David@hamal.mc21.fee.unicamp.br SIP/2.0
Via: SIP/2.0/UDP hamal.mc21.fee.unicamp.br:5060;branch=k9hG4kwhIJRoa98nmbh
Via: SIP/2.0/UDP rukbat.mc21.fee.unicamp.br:5060;branch=k9hG4kbbaWFjvxx5tSI;
received=143.106.50.84
Max-Forwards: 69
To: <sip:david@hamal.mc21.fee.unicamp.br>
From: <sip:davison@hamal.mc21.fee.unicamp.br>;tag=faygNT1lt/eLY
Call-ID: ..8o7ZxomUexo
CSeq: 6358 INVITE
Contact: <sip:davison@rukbat.mc21.fee.unicamp.br>
Content-Type: application/sdp
```

```
Content-Length: 166

v=0
user=davison 1234843216574 1944887216587 IN IP4 143.106.50.84
s=Session SDP
c=IN IP4 143.106.50.84/8000
t=0 0
m=audio 8000 udp/AVP 97
a=wb:97 U8/8000
```

- **Terminal SIP 2 → Proxy**

```
SIP/2.0 180 Ringing
Via: SIP/2.0/UDP hamal.mc21.fee.unicamp.br:5060;branch=k9hG4kwhIJRoa98nmbh;
received=143.106.50.69
Via: SIP/2.0/UDP rukbat.mc21.fee.unicamp.br:5060;branch=k9hG4kbbaWFjvxx5tSI;
received=143.106.50.84
To: <sip:david@hamal.mc21.fee.unicamp.br>
From: <sip:davison@hamal.mc21.fee.unicamp.br>;tag=faygNT1lt/eLY
Call-ID: ..8o7ZxomUexo
CSeq: 6358 INVITE
Content-Length: 0
```

- **Proxy → Terminal SIP 1**

```
SIP/2.0 180 Ringing
Via: SIP/2.0/UDP rukbat.mc21.fee.unicamp.br:5060;branch=k9hG4kbbaWFjvxx5tSI;
received=143.106.50.84
To: <sip:david@hamal.mc21.fee.unicamp.br>
From: <sip:davison@hamal.mc21.fee.unicamp.br>;tag=faygNT1lt/eLY
Call-ID: ..8o7ZxomUexo
CSeq: 6358 INVITE
Content-Length: 0
```

Após o recebimento da resposta 180 (*Ringing*), o Terminal SIP 1 fica aguardando uma resposta final do Terminal SIP 2. O Terminal SIP 2, aceitando o convite e responde como abaixo:

- **Terminal SIP 2 → Proxy**

```
SIP/2.0 200 Ok
Via: SIP/2.0/UDP hamal.mc21.fee.unicamp.br:5060;branch=k9hG4kwhIJRoa98nmbh;
received=143.106.50.69
Via: SIP/2.0/UDP rukbat.mc21.fee.unicamp.br:5060;branch=k9hG4kbbaWFjvxx5tSI;
received=143.106.50.84
```

```
Max-Forwards: 69
To: <sip:david@hamal.mc21.fee.unicamp.br>;tag=n947n9j87
From: <sip:davison@hamal.mc21.fee.unicamp.br>;tag=faygNT1lt/eLY
Call-ID: ..8o7ZxomUexo
CSeq: 6358 INVITE
Contact: <sip:david@taygete.mc21.fee.unicamp.br>
Content-Type: application/sdp
Content-Length: 166

v=0
user=david 1234843216574 1944887216587 IN IP4 143.106.50.74
s=Session SDP
c=IN IP4 143.106.50.74/8000
t=0 0
m=audio 8000 udp/AVP 97
a=wb:97 U8/8000
```

- **Proxy → Terminal SIP 1**

```
SIP/2.0 200 Ok
Via: SIP/2.0/UDP rukbat.mc21.fee.unicamp.br:5060;branch=k9hG4kbbaWFjvxx5tSI;
received=143.106.50.84
Max-Forwards: 69
To: <sip:david@hamal.mc21.fee.unicamp.br>;tag=n947n9j87
From: <sip:davison@hamal.mc21.fee.unicamp.br>;tag=faygNT1lt/eLY
Call-ID: ..8o7ZxomUexo
CSeq: 6358 INVITE
Contact: <sip:david@taygete.mc21.fee.unicamp.br>
Content-Type: application/sdp
Content-Length: 166

v=0
user=david 1234843216574 1944887216587 IN IP4 143.106.50.74
s=Session SDP
c=IN IP4 143.106.50.74/8000
t=0 0
m=audio 8000 udp/AVP 97
a=wb:97 U8/8000
```

- **Terminal SIP 1 → Terminal SIP 2**

```
ACK sip:david@taygete.mc21.fee.unicamp.br SIP/2.0
Via: SIP/2.0/UDP rukbat.mc21.fee.unicamp.br:5060;branch=k9hG4kbbaWFjvxx5tSI
Max-Forwards: 70
To: <sip:david@hamal.mc21.fee.unicamp.br>;tag=n947n9j87
From: <sip:davison@hamal.mc21.fee.unicamp.br>;tag=faygNT1lt/eLY
```

```
Call-ID: ..8o7ZxomUexo
CSeq: 6358 ACK
Content-Length: 0
```

6.1.3 Finalização de uma Chamada

Em nossa implementação, a finalização de uma chamada é efetuada pelo envio de uma mensagem **BYE** e o recebimento de uma resposta **200 Ok**. A sinalização completa é mostrada abaixo.

- **Terminal SIP 1 → Terminal SIP 2**

```
BYE sip:david@taygete.mc21.fee.unicamp.br SIP/2.0
Via: SIP/2.0/UDP rukbat.mc21.fee.unicamp.br:5060;branch=k9hG4kbbaWFjvxx5tSI
Max-Forwards: 70
To: <sip:david@hamal.mc21.fee.unicamp.br>;tag=n947n9j87
From: <sip:davison@hamal.mc21.fee.unicamp.br>;tag=faygNT1lt/eLY
Call-ID: ..8o7ZxomUexo
CSeq: 6359 BYE
Content-Length: 0
```

- **Terminal SIP 2 → Terminal SIP 1**

```
SIP/2.0 200 Ok
Via: SIP/2.0/UDP rukbat.mc21.fee.unicamp.br:5060;branch=k9hG4kbbaWFjvxx5tSI;
received=143.106.50.84
Max-Forwards: 70
To: <sip:david@hamal.mc21.fee.unicamp.br>;tag=n947n9j87
From: <sip:davison@hamal.mc21.fee.unicamp.br>;tag=faygNT1lt/eLY
Call-ID: ..8o7ZxomUexo
CSeq: 6359 BYE
Content-Length: 0
```

Após do recebimento do 200 Ok, a sessão está finalizada.

6.2 Configurações da Placa de áudio

Em nossa implementação, existem várias configurações de áudio que podem ser alteradas na placa de som. Estas configurações foram descritas na seção 5.2.4 e podem alterar o atraso da conversação.

Para uma chamada de telefonia, o padrão de codificação possui uma taxa de amostragem de 8 kHz, com amostras de 8 bits de tamanho, lei de compressão A (sistema

brasileiro) e a conversa é em monocanal. Se forem considerados estes valores, pode-se estimar o atraso sofrido por um pacote de áudio. Considerando-se os seguintes componentes de atrasos:

- **Tempo de empacotamento:** É o tempo em que o dispositivo de áudio demora em disponibilizar um pacote de áudio para a aplicação.
- **Tempo de transmissão na origem:** Este é o tempo em que a interface de rede demora em transmitir um pacote de áudio.
- **Tempo de propagação na rede:** Este é o tempo que o pacote demora a se propagar na rede.
- **Tempo de recepção no destino:** Este é o tempo que a interface de rede demora em receber um pacote de áudio.
- **Tempo de desempacotamento e reprodução:** É o tempo gasto pelo dispositivo de áudio para desempacotar o pacote e reproduzi-lo.

Os tempos de transmissão na origem e de recepção no destino são valores que dependem da interface de rede utilizada nos Terminais. Normalmente, as interfaces utilizadas são placas de rede e modens. Para o caso dos modens, considerando-se uma taxa de 56 kbps, o tempo de transmissão é dado pela seguinte relação:

$t_{t_0} = \frac{(F + 40) \times 8}{56000}$, onde F é o tamanho do fragmento da placa de som e os 40 Bytes se referem ao tamanho do cabeçalho do pacote.

Considerando-se F = 256 Bytes, o tempo de transmissão será: $t_{t_0} = 42{,}29 mseg$.

Para o caso de redes locais, a interface de rede trabalha, normalmente, a 10 Mbps. Portanto, o tempo de transmissão é dado pela relação:

$t_{t_0} = \frac{(F + 40) \times 8}{10^7}$. Considerando-se F = 256 Bytes, $t_{t_0} = 234{,}8 \mu seg$.

O tempo de transmissão, e conseqüentemente o tempo de recepção, para redes locais é muito menor que o tempo de transmissão na Internet com modens. Quando se está utilizando uma rede local, os tempos de transmissão na origem e recepção no destino podem ser desprezados. Portanto, o atraso de um pacote de áudio pode ser dado pela relação:

$$T = t_E + t_P + t_R, \text{ onde}: \begin{cases} t_E \rightarrow \text{Tempo de empacotamento.} \\ t_P \rightarrow \text{Tempo de propagação.} \\ t_R \rightarrow \text{Tempo de desempacotamento e reprodução} \\ T \rightarrow \text{Atraso total} \end{cases}$$

Pode-se considerar que $t_E = t_R$ [24], então:

$T = 2 \times t_E + t_P$

O tempo de propagação na rede não é um valor constante. Quando a rede está congestionada, este tempo pode se torna da ordem de segundos. Os atrasos nos quais se pode ter um controle são os atrasos devido ao empacotamento e ao desempacotamento, e devido a reprodução do pacote. Este atraso é dado pela seguinte relação:

$t_E = \frac{F \times 8}{B} seg.$, previamente apresentada na seção 5.2.4.

Através desta relação, pode-se fazer uma tabela relacionando o tamanho do fragmento (*F*) e o tempo de empacotamento.

F (Bytes)	T_E (mseg)
64	8
128	16
256	32
512	64
1024	128
2048	256

Tabela 1. Tabela do tamanho do fragmento e o tempo de empacotamento

O ITU-T, através da recomendação G.114 - *One-way transmission time*[24], estabelece que as comunicações de áudio, em redes de pacotes, devem atender aos seguintes requisitos:

- Para redes em tempo real, o atraso máximo sofrido por um pacote de áudio em uma direção deverá ser no máximo de 150 mseg.

- Para redes que não possuam garantias de tempo real, o atraso máximo pode ser de 450 mseg, desde que os usuários do sistema estejam cientes dos atrasos que ocorrerão na conversação.

Pode-se observar que se for considerado que o atraso na rede é muito pequeno em comparação ao tempo de empacotamento (o que geralmente acontece em redes locais), o maior fragmento que se pode usar, para atender o requisito de 150 mseg, será de 512 bytes. Por isso, um parâmetro importante de configuração que deve estar disponível para o usuário é o tamanho do fragmento do dispositivo.

O Terminal SIP implementado permite que o usuário configure o tamanho do fragmento, se o dispositivo de áudio permitir, de um valor inicial de 16 bytes até o máximo permitido pelo dispositivo, que geralmente é de 2048 bytes.

Variando-se o valor do tamanho do fragmento, pode-se observar o atraso sofrido entre os dois Terminais SIP da Figura 48. Para obter tais dados, foram utilizadas as ferramentas de captura e análise de dados **tcpdump** e **tcptrace**. Estas ferramentas acompanham o Linux e servem para coletar e mostrar as estatísticas dos pacotes que saem de uma determinada interface de rede.

F (Bytes)	*T* (mseg)
128	15,98
256	31,98
512	63,98
1024	127,93

Tabela 2. Tabela do tamanho do fragmento e atraso entre dois terminais

Comparando-se a Tabela 1 com a Tabela 2, pode-se observar que o atraso sofrido entre os dois terminais se deve principalmente ao tempo de empacotamento.

7. Conclusões

Neste trabalho, foi desenvolvido um Sistema de Voz sobre IP baseado no protocolo SIP[1]. Este Sistema de Voz sobre IP, o Sistema SIP, é composto pelos componentes básicos apresentados na RFC 3261[1]: o Terminal SIP, o Proxy e o Servidor de Registros. O Terminal SIP é um programa desenvolvido na linguagem C++ e com interface gráfica baseada na biblioteca QT[25]. O Proxy e o Servidor de Registros foram desenvolvidos em um único programa, também em linguagem C++, a fim de facilitar o processamento e o envio das mensagens SIP. Todos estes componentes, desenvolvidos para o Sistema Operacional Linux, podem ser utilizados para o estabelecimento de sessões de áudio. Um exemplo de seu uso é em ambientes de educação à distância.

O foco deste trabalho, o Sistema SIP, é baseado em uma biblioteca SIP, que é uma contribuição deste trabalho. Diferente de outras implementações em desenvolvimento[31][32], esta biblioteca é bem simples e de fácil manipulação. O código é em linguagem C++ e as mensagens são geradas dentro de funções, as quais o programador pode incluir ou extrair campos da forma que desejar.

Além da biblioteca SIP, outra contribuição é a sinalização do sistema que é feita de uma maneira diferente da maneira convencional[1]. As mensagens SIP, com exceção da mensagem REGISTER, são enviadas para o Proxy do domínio do destinatário da mensagem. Isto é vantajoso porque diminui o número de Proxies que a requisição deverá passar e diminui o processamento nos Proxies.

O acesso aos parâmetros da placa de áudio é outra contribuição do trabalho. Normalmente, os programas de Voz sobre IP não permitem, ao usuário, modificar algumas configurações da placa de áudio que podem melhorar a conversação. Um destes parâmetros é o tamanho do fragmento, que reduz o tempo de empacotamento da placa de som.

Já no contexto teórico, a contribuição deste trabalho é uma pesquisa bibliográfica baseada em livros, normas técnicas, RFCs, páginas de Internet e artigos publicados. A partir desta pesquisa, foram trabalhados vários conceitos relacionados com os temas:

- Protocolos de Telefonia IP: SIP e H.323;

- Protocolo TCP/IP;
- Sistema Operacional Linux; e
- Programação de dispositivos, Programação Paralela (Threads) e Programação Gráfica[25].

7.1 Trabalhos Futuros

Este trabalho pode servir de base para a inclusão de novas funcionalidades e novos serviços em Sistemas de Telefonia sobre IP. Por isso, aqui estão algumas sugestões para possíveis trabalhos futuros.

- O Sistema SIP não tem um método de troca de informações entre os Servidores de Registro. Um trabalho possível é a implementação da comunicação entre estes Servidores como um ambiente de Sistemas Distribuídos.
- Recentemente, o IETF publicou a RFC 3313 – *Private Session Initiation Protocol (SIP) Extensions for Media Authorization*. Esta RFC trata da sinalização SIP para negociar os parâmetros de QoS. Um outro trabalho possível é adicionar a negociação de QoS na biblioteca SIP desenvolvida.
- Os chips programáveis são uma boa opção de desenvolvimento de tecnologias. Eles são baseados em sistemas operacionais que permitem copiar, para a sua memória, programas desenvolvidos em outros computadores. A biblioteca SIP pode ser adaptada para trabalhar em um destes chips.
- A inclusão de vídeo neste trabalho seria um trabalho interessante. O protocolo SDP pode negociar os parâmetros de vídeo e de áudio para uma sessão de vídeo-conferência.
- Um outro trabalho futuro é realizar testes na Internet e tentar ajustar os parâmetros da placa de áudio a fim de que o atraso sofrido entre os pacotes de áudio seja reduzido ao máximo.

8. Referências Bibliográficas

[1]. Rosenber, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M. e Schooler, E.: **SIP: Session Initiation Protocol – RFC 3261**, *Internet Engineering Task Force*, Junho 2002.

[2]. Handley, M. e Jacobson, V.: **SDP: Session Description Protocol – RFC 2327**, *Internet Engineering Task Force*, Abril 1998.

[3]. Resnick, P. e Ed.: **Internet Message Format – RFC 2822**, *Internet Engineering Task Force*, Abril 2001.

[4]. Berners-Lee, T., Masinter, L. e MacCahill, M.: **Uniform Resource Locators (URL) – RFC 1738**, *Internet Engineering Task Force*, Janeiro 1998.

[5]. Yergeau, F.: **UTF-8, a Transformation Format of ISO 10646 – RFC 2279**, *Internet Engineering Task Force*, Janeiro 1998.

[6]. Postel, J.: **User Datagram Protocol – RFC 768**, *Internet Engineering Task Force*, Agosto 1980.

[7]. Postel, J.: **DoD Standard Transmission Control Protocol – RFC 761**, *Internet Engineering Task Force*, Janeiro 1980.

[8]. Defense Advanced Research Projects Agency: **Transmission Control Protocol – DARPA Internet Program – Protocol Specification – RFC 793**, *Internet Engineering Task Force*, Setembro 1981.

[9]. Franks, J., Hallam-Baker, P. Hostetler, J., Lawrence, S., Leach, P., Luotonen, A. e Stewart, L.: **HTTP Authentication: Basic and Digest Access Authentication – RFC 2617**, *Internet Engineering Task Force*, Junho 1999.

[10]. Schulzrinne, H., Casner, S., Frederick, R. e Jacobson, V.: **RTP: a transport protocol for real-time applications – RFC 1889**, *Internet Engineering Task Force*, Janeiro 1996.

[11]. Schulzrinne, H., Rao, A. e Lanphier, R.: **Real Time Streaming Protocol (RTSP) – RFC 2326**, *Internet Engineering Task Force*, Abril 1998.

[12]. Cuervo, F., Greene, N., Rayhan, A., Huitema, C., Rosen, B. e Segers, J.: **Megaco protocol version 1.0 – RFC 3015**, *Internet Engineering Task Force*, Novembro 2000.

[13]. Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P. e Bereners-Lee, T.: **Hypertext Transfer Protocol – HTTP/1.1 – RFC 2616**, *Internet Engineering Task Force*, Junho 1999.

[14]. Schulzrinne, H.: **RTP Profile for Audio and Video Conferences with Minimal Control – RFC 1890**, *Internet Engineering Task Force*, Janeiro 1996.

[15]. Recomendação H.323: **Packet-based Multimedia Communications Systems**, *International Telecommunication Union*, Novembro 2001.

[16]. Recomendação H.225.0: **Call Signalling Protocols and Media Stream Packetization for Packet Based Multimedia Communications Systems,** *International Telecommunication Union*, Novembro 2000.

[17]. Recomendação H.245: **Control Protocol for Multimedia Communication**, *International Telecommunication Union*, Março 2000.

[18]. Recomendação Q.931: **ISDN User-network Interface Layer 3 Specification for Basic Call Control,** *International Telecommunication Union*, 1993.

[19]. Recomendação G.711: **Pulse Code Modulation (PCM) of Voice Frequencies**, *International Telecommunication Union*, 1988.

[20]. Recomendação G.722: **7 kHz Audio-Coding within 64 kbit/s**, *International Telecommunication Union*, 1988.

[21]. Recomendação G.723.1: **Speech coders: Dual Rate Speech Coder for Multimedia Communications Transmitting at 5.3 and 6.3 kbit/s**, *International Telecommunication Union*, 1996.

[22]. Recomendação G.728: **Coding of Speech at 16 kbit/s Using Low-Delay Code Excited Linear Prediction**, *International Telecommunication Union*, 1992.

[23]. Recomendação G.729: **Coding of Speech at 8 kbit/s Using Conjugate Structure Algebraic-Code-Excited Linear-Predicition (CS-ACELP)**, *International Telecommunication Union*, 1996.

[24]. Recomendação G.114: **One-way Transmission Time**, *International Telecommunication Union*, 2000.

[25]. Ward, Patrick: **QT Programming for Linux and Windows 2000**, *Editora Hewlett-Packard Professional Books*, 2001.

[26]. Leffler, S. J., Fabry, R. S. e Joy W. N., Lapsley, P.: **An Advanced 4.3BSD Interprocess Communication Tutorial**, *University of Maryland*, 2001

[27]. Mehta, P. e Udani, S.: **Voice over IP – Sounding good on the Internet**, *IEEE Potentials*, 2001.

[28]. **QT 3.1 Whitepaper**, *Trolltech* – http://www.trolltech.com

[29]. Schulzrinne, H. e Rosenberg, J.: **A Comparison of SIP and H.323 for Internet Telephony**. http://citesser.nj.nec.com

[30]. **Open Sound System Programmer's Guide**, Versão 1.1, *4Front Technologies*, 2000. http://www.opensound.com.

[31]. Azevedo, Ernani: **LiboSIP Telephony Library**, Versão 0.9.0. http://www.fsf.org/software/osip/osip.html.

[32]. **Vovida**, *Vovida.org*, http://www.vovida.org.

Apêndice I. Campos das Mensagens SIP

Os campos mais usados nas mensagens SIP, e que foram utilizados neste trabalho, foram descritos nos capítulos deste trabalho. Nesta seção, será dada uma descrição de todos os campos existentes na RFC 3261[1].

A I.1. Accept

O campo **Accept** lista todos os formatos da sessão que são suportados pela entidade SIP em uma sessão. Se este campo não estiver presente em mensagens que estabeleçam sessões, o lado servidor da negociação assume que o seu valor é **application/sdp.**

Um campo **Accept** válido é mostrado no exemplo abaixo:

```
Accept: application/sdp, text/html
```

Note que os formatos são separados por uma vírgula.

A I.2. Accept-Encoding

O campo **Accept-Encoding** restringe a codificação da sessão a ser estabelecida.

Por exemplo:

```
Accept-Encoding: gzip
```

A I.3. Accept-Language

O campo **Accept-Language** é usado nas requisições para indicar a língua desejada nas frases descritivas de um código de status, nos descritores da sessão e nas respostas de status conduzidas como o corpo da mensagem de resposta. O valor padrão deste campo é **en**(inglês).

Por exemplo:

```
Accept-Language: da, pt-br, en-gb, em
```

A I.4. Alert-Info

Este campo pode ser usado na mensagem **INVITE** para indicar ao servidor o tipo de tom de discagem para o lado servidor, isto é, ele especifica uma página de Internet que possua um som para ser tocado quando for dar o tom de chamada.

Por exemplo:

Alert-Info: <http://www.mc21.fee.unicamp.br/tom.wav>

A I.5. Allow

O campo **Allow** lista o conjunto de métodos suportados pelo agente servidor do usuário.

Por exemplo:

Allow: INVITE, ACK, OPTIONS, CANCEL, BYE.

A I.6. Authentication-Info

O campo **Authentication-Info** fornece uma autenticação mútua com o *HTTP Digest*[9], que é um mecanismo de segurança para as páginas de Internet. Um agente servidor pode incluir este campo em uma resposta **2xx** para uma requisição que foi autenticada usando a chave incluída no campo **Authorization** da requisição recebida.

Por exemplo:

Authentication-Info: nextnonce="nv934764nhb94087689b84"

A I.7. Authorization

O campo **Authorization** contém as credenciais de autenticação dos Agentes do Usuário. Este campo é usado com a autenticação http.

Por exemplo:

Authorization: Digest username="Davison",
realm="regulus.mc21.fee.unicamp.br,
nonce="84a4cc6f3082121f32b42a2187831a9e",
response="7587245234b3434cc3412213e5f113a5432"

A I.8. Call-ID

O campo **Call-ID** identifica uma sessão SIP ou todos os registros de um cliente em um Servidor de Registros.

Por exemplo:

```
Call-ID: 948bneiovoeyu59034u@mc21.fee.unicamp.br
```

A I.9. Call-Info

O campo **Call-Info** fornece informações adicionais sobre o iniciante da sessão, quando este campo está numa mensagem **INVITE**, ou sobre o destinatário da mensagem, quando este campo está numa resposta final **2xx**.

Por exemplo:

```
Call-Info: <http://www.mc21.fee.unicamp.br/foto.jpg>;
purpose=icon
```

O parâmetro **purpose** indica que o link é de uma imagem.

A I.10. Contact

O campo **Contact** contém um URI cujo significado depende do tipo da requisição ou resposta em que ele está. Por exemplo, na mensagem **REGISTER**, este campo contém ligações de endereços reais ao qual o endereço SIP estará relacionado. Cada valor pode vir com um parâmetro **expires**, indicando o tempo de registro de cada endereço.

```
Contact: <davison@regulus.mc21.fee.unicamp.br>,
expires=3600
```

A I.11. Content-Disposition

O campo **Content-Disposition** indica como o Agente do Usuário deverá interpretar o corpo da mensagem, isto é, ele indica se o corpo da mensagem deverá ser interpretado como texto, figura, aplicativo, descritor de sessão, etc.

Por exemplo:

```
Content-Disposition: session
```

A I.12. Content-Encoding

O campo **Content-Encoding** indica qual a codificação usada para o corpo da mensagem.

Por exemplo:

```
Content-Encoding: gzip
```

A I.13. Content-Language

O campo **Content-Language** indica qual a linguagem do corpo da mensagem na requisição.

Por exemplo:

```
Content-Language: pt-br
```

A I.14. Content-Length

O campo **Content-Length** indica qual o tamanho do corpo da mensagem SIP, em bytes.

Por exemplo:

```
Content-Lenght: 349
```

A I.15. Content-Type

O campo **Content-Type** indica o tipo do corpo da mensagem.

Por exemplo:

```
Content-Type: application/sdp
```

A I.16. CSeq

O campo **CSeq**, em uma requisição, contém um número de seqüência e o método da mensagem.

Por exemplo:

```
CSeq: 4722 INVITE
```

A I.17. Date

O campo **Date** contém a data e a hora em que a requisição ou a resposta foi enviada.

Por exemplo:

```
Date: Sat, 26 Nov 2003 12:00:00 GMT
```

A I.18. Error-Info

O campo **Error-Info** contém um endereço de Internet no qual podem ser obtidas mais informações sobre o status de erro da resposta.

Por exemplo:

```
Error-Info:<http://www.mc21.fee.unicamp.br/errors.html>
```

A I.19. Expires

O campo **Expires** contém o tempo, em segundos, que a mensagem ou conteúdo da mensagem expira. Este tempo depende do método. No caso do **INVITE**, o campo **Expires** é relativo ao tempo em que a requisição **INVITE** é válida, e não ao tempo da duração da sessão estabelecida. No caso da mensagem **REGISTER,** este campo indica qual é o tempo de duração daquele registro.

Por exemplo:

```
Expires: 3600
```

A I.20. From

O campo **From** indica, para uma mensagem **INVITE**, o endereço SIP do iniciante da sessão. Para a mensagem **REGISTER**, este campo indica o endereço SIP de quem está efetuando o registro.

Por exemplo:

```
From: <sip:davison@hamal.mc21.fee.unicamp.br>,tag=47594
```

A I.21. In-Reply-To

O campo **In-Reply-To** referencia aos valores dos **Call-IDs** que identificam a sessão atual.

Por exemplo:

```
In-Reply-To: noeogu048048@mc21.fee.unicamp.br,
949840809@igniscom.com.br
```

A I.22. Max-Forwards

O campo **Max-Forwards** deve estar presente em todas as mensagens SIP para limitar o número de proxies e gateways pelos quais a mensagem pode passar.

Por exemplo:

```
Max-Forwards: 70
```

A I.23. Min-Expires

O campo **Min-Expires** contém o tempo mínimo de expiração suportado pelo servidor. Para o caso de uma mensagem **REGISTER**, este campo aparece na resposta **423 Registration Too Brief**, indicando que o tempo do campo **Expires** da mensagem **REGISTER** é menor do que o tempo suportado pelo servidor de registros.

Por exemplo:

```
Min-Expires: 300
```

A I.24. MIME-Version

O campo **MIME-Version** indica qual a versão do **MIME** que está sendo usado.

Por exemplo:

```
MIME-Version: 1.0
```

A I.25. Organization

O campo **Organization** contém o nome da organização para o qual o elemento SIP enviando a requisição ou resposta pertence.

Por exemplo:

```
Organization: MC-21
```

A I.26. Priority

O campo **Priority** é usado para indicar se a sessão SIP é uma sessão de emergência ou não. Tipicamente, o campo **Priority** é usado para indicar o campo **Subject**.

Por exemplo:

```
Subject: A tornado is heading our way!
Priority: emergency
```

A I.27. Proxy-Authenticate

O campo **Proxy-Authenticate** contém as chaves de autenticação para o Proxy.

Por exemplo:

```
Proxy-Authenticate: Digest realm="mc21.fee.unicamp.br",
domain="hamal.mc21.fee.unicamp.br",
nonce="f84f1ce41e6cb45aea9c8488d359,opaque="", stale=FALSE,
algorithm=MD5.
```

Onde o valor **Digest** indica a autenticação http e o parâmetro **algorithm** indica o esquema de codificação.

A I.28. Proxy-Authorization

O campo **Proxy-Authorization** permite o cliente se identificar em um Proxy que requeira autenticação.

Por exemplo:

```
Proxy-Authorization: Digest username="Davison",
realm="mc21.fee.unicamp.br,
nonce="c60f3082ee1313b402a21831ae",
response="245f23e15f11432b343441c022"
```

A I.29. Proxy-Require

O campo **Proxy-Require** é usado para indicar opções que o Proxy deve aceitar. O seu valor é uma *string* que descreve os serviços disponibilizados através de novas RFCs.

Por exemplo:

```
Proxy-Require: foo
```

A I.30. Record-Route

O campo **Record-Route** é inserido na requisição pelos proxies para forçar que as futuras requisições em um diálogo sejam roteadas pelo Proxy.

Por exemplo:

```
Record-Route: <sip:hamal.mc21.fee.unicamp.br>,
<sip:sheratan.mc21.fee.unicamp.br>
```

A I.31. Reply-To

O campo **Reply-To** contém um URI para o retorno das mensagens que pode ser diferente daquele fornecido pelo campo **From**. Por exemplo, o endereço contido no **Reply-To** pode ser usado para retornar respostas de erro durante o estabelecimento de uma sessão.

```
Reply-To: Davison <sip:davison@mc21.fee.unicamp.br>
```

A I.32. Require

O campo **Require** é usado pelo agente cliente do usuário para dizer ao agente servidor do usuário quais opções que o agente cliente espera que o agente servidor suporte para processar a requisição.

Por exemplo:

```
Require: 100rel
```

A I.33. Retry-After

O campo **Retry-After** pode ser usado em uma mensagem de resposta **503 (Service Unavailable)** para indicar quanto tempo o serviço vai estar indisponível para o cliente, e pode ser usado nas respostas **404 (Not Found), 413 (Request Entity Too Large), 480 (Temporarily Unavailable), 486 (Busy Here), 600 (Busy)** ou **603 (Decline)** para indicar quando a parte chamada vai estar disponível de novo. Um parâmetro opcional, o **duration**, pode estar presente para indicar o período em que o serviço ou o servidor vai estar disponível após o retorno do serviço.

Por exemplo:

```
Retry-After: 180000; duration=3600
```

A I.34. Route

O campo **Route** é usado para forçar o roteamento de uma mensagem através de um conjunto de proxies.

Por exemplo:

```
Route: <sip:hamal.mc21.fee.unicamp.br>,
<sip:ignis5.igniscom.com.br>,
<sip:directnet8.directnet.com.br>
```

A I.35. Server

O campo **Server** contém o nome do Software usado para gerenciar a requisição pelo servidor.

Por exemplo:

```
Server: QTServer v3
```

A I.36. Subject

O campo **Subject** informa um resumo da sessão ou indica a natureza da sessão.

Por exemplo:

```
Subject: audio call
```

A I.37. Supported

O campo **Suported** enumera todas as extensões suportadas pelo agente cliente ou agente servidor.

Por exemplo:

```
Supported: foo
```

A I.38. Timestamp

O campo **Timestamp** indica quando o agente cliente envia a requisição para o agente servidor.

Por exemplo:

```
Timestamp: 54
```

A I.39. To

O campo **To** indica o destinatário SIP da requisição, exceto nas requisições **REGISTER**, onde este campo contém o endereço de registro do Terminal SIP.

Por exemplo:

To: Davison <sip:davison@mc21.fee.unicamp.br>

A I.40. Unsupported

O campo **Unsupported** indica quais as extensões que o servidor não suporta.

Por exemplo:

Unsupported: foo

A I.41. User-Agent

O campo **User-Agent** possui informações sobre o agente cliente que está gerando a requisição.

Por exemplo:

User-Agent: QTSIPphone v2.0

A I.42. Via

O campo **Via** indica o caminho tomado pela requisição e indica o caminho de volta da resposta. O parâmetro **branch** serve como um identificador da transação e é usado pelos proxies para detectar loops.

Por exemplo:

Via: SIP/2.0/UDP regulus.mc21.fee.unicamp.br:5060; branch=z9hG4bK87asdks7

Via: SIP/2.0/UDP hamal.mc21.fee.unicamp.br:5060; received=143.106.50.69;branch=z9hG4bK77asjd

A I.43. Warning

O campo **Warning** é usado para conduzir informações adicionais sobre o código de *status* da resposta. O campo contém um número de três dígitos, que é o código do *warning*, o nome do terminal e um texto de *warning*.

Os códigos definidos pela RFC 3261 são:

- **300 Incompatible network protocol:** Um ou mais protocolos de rede contidos no descritor de sessão não são suportados pelo servidor.

- **301 Incompatible network address formats:** Um ou mais formatos de endereços de rede contidos no descritor de sessão não são suportados pelo servidor.
- **302 Imcompatible transport protocol:** Um ou mais protocolos de transporte não são suportados pelo servidor.
- **303 Incompatilbe bandwidth units:** A largura de banda requisitada no corpo da mensagem não é suportada pelo servidor.
- **304 Media type not available:** Um ou mais tipos de mídia contidos no corpo da mensagem não são suportados pelo servidor.
- **305 Incompatible media formate:** Um ou mais tipos de formatos de mídia contidos no corpo da mensagem não são suportados pelo servidor.
- **306 Attribute not understood:** Algum parâmetro descritor da mídia não foi entendido pelo servidor.
- **307 Session description parameter not understood:** Algum outro parâmetro do corpo da mensagem que não foi entendido pelo servidor
- **330 Multicast not available:** O terminal onde o usuário está, não suporta multicast.
- **331 Unicast not available:** O terminal onde o usuário está não suporta unicast.
- **370 Insufficient bandwidth:** É quando o terminal não possui mais banda para aceitar uma conexão com a banda requisitada no corpo da mensagem.
- **399 Miscellaneous warning:** É um texto arbitrário de *warning* contendo alguma informação que não está em um dos itens descritos acima.

Por exemplo:

`Warning: 301 unicamp.br "Session parameter foo not understood"`

A I.44. WWW-Authenticate

O campo **WWW-Authenticate** contém informações de autenticação www.

Por exemplo:

WWW-Authenticate: Digest realm="unicamp.br",

domain="sip:mc21.fee.unicamp.br",

nonce="f84f1cec41e6cbe5aea9c8e88d359", opaque="",

stale=FALSE, algorithm=MD5

Apêndice II. Campos da Mensagem SDP

Neste apêndice serão descritos os campos das mensagens SDP definidos na RFC 2327[2].

A II.1. Versão do Protocolo (v)

O campo **v** dá a versão do protocolo SDP em uso. A versão atual é a zero, pois só existe uma única versão desenvolvida.

Por exemplo:

```
v=0
```

A II.2. Origem (o)

O campo **o** tem o seguinte formato:

```
o=<username> <id sessão> <versão> <tipo de rede> <tipo de endereço> <endereço de rede>
```

O campo **o** descreve o cliente da sessão SIP, indicando o seu **username**, um identificador da sessão (**id sessão**), o número da versão deste descritor de sessão (**versão**), o tipo de rede (se é IP ou não), o tipo de endereço (IPv4 ou IPv6), e o endereço de rede do terminal.

Por exemplo:

```
o=davison 2890844526 2980842807 IN IP4 143.106.50.80
```

Onde o **IN** indica uma rede IP e o **IP4** indica que o protocolo em uso é o IP versão 4.

A II.3. Nome da Sessão (s)

O campo **s** indica o nome da sessão SDP a ser estabelecida.

Por exemplo:

```
s=SDP seminar
```

A II.4. Informação sobre a Sessão e a Mídia (i)

O campo **i** é um campo que descreve a sessão a ser estabelecida.

Por exemplo:

```
i=A Seminar on the Session Initiation Protocol
```

A II.5. URI (u)

O campo **u** contém um endereço de internet de uma página que mantenha informações sobre a sessão SDP a ser estabelecida.

Por exemplo:

```
u=http://www.mc21.fee.unicamp.br/sip/
```

A II.6. Endereço de e-mail (e) e número do telefone (p)

Os campos **p** e **e** descrevem o telefone e o endereço de e-mail do requisitante.

Por exemplo:

```
p=+55-019-3788-3829
e=davison@decom.fee.unicamp.br
```

A II.7. Informações sobre a Conexão (c)

O campo **c** contém informações sobre o endereço em que o requisitante espera receber os pacotes da mídia.

Por exemplo:

```
c=IN IP4 143.106.50.80/8000
```

A II.8. Largura de Banda (b)

O campo **b** indica a largura de banda da sessão a ser estabelecida. O seu valor é dado em kilobits por segundo.

Por exemplo:

```
b=64
```

A II.9. Tempo de Duração da Sessão (t) e de repetição (r)

O campo **t** indica o tempo, geralmente em segundos, de duração da sessão. O seu formato geral é:

```
t=<início> <fim>
```

O **início** é a hora de início da troca de mídia, e **fim** é a hora de término do fluxo de mídia. O valor destes campos é em segundos e é dado pela representação decimal do **NTP** – *Network Time Protocol***.** Para converter os valores para a hora do sistema operacional, basta subtrair o valor de 2208988800. Se o valor deste campo for 0 0, indica que a sessão não é limitada por tempo.

Por exemplo:

```
t=3034423619 3042462419
```

O campo **r** indica quantas vezes a sessão será repetida durante a semana. Os seus valores podem ser dados em unidades de dias (d), horas (h), minutos (m) e segundos (s).

Por exemplo:

```
r=6d 1h 0 25h
```

A II.10. Ajuste do Tempo (z)

O campo **z** é usado para ajustar a hora de acordo com a zona de horário do país da pessoa. Por exemplo:

```
z=2882844526 -1h 2898848070 0
```

A II.11. Chaves de Criptografia (k)

O campo **k** indica a chave de criptografia e o tipo de codificação a ser usado.

Por exemplo:

```
k=base64:9n947b74nb049756
```

A II.12. Campos Descritores de Mídia (m) e de Atributos (a)

O campo **m** descreve os parâmetros da conexão da mídia. O seu formato geral é:

```
m=<mídia> <porta> <transporte> <lista de formatos>
```

O parâmetro **mídia** descreve o tipo de mídia a ser trocada na sessão. Os seus valores possíveis são: **audio**, **video**, **application**, **data** e **control**. O parâmetro **porta** indica a porta na qual serão trocados os pacotes da mídia. O parâmetro **transporte** indica o tipo de protocolo de transporte. O parâmetro **lista de formatos** indica quais os formatos suportados pelo cliente. Estes formatos são números inteiros que descrevem o *codec* em uso. Os números entre 0 e 96 são números restritos a RFC 1890 – *RTP Profile for Audio and Video Conferences with Minimal Control*. Por exemplo, o número 0 identifica o *codec* G.711, lei μ. Para os valores entre 97 a 127, o campo **a** descreverá os parâmetros do *codec* em uso.

Por exemplo:

```
m=audio 8000 udp 0 97
a=media:97 L8/8000/2
```

O parâmetro **L8** indica amostras com 8 bits com sinal. O parâmetro **8000** é a taxa de amostragem, e o parâmetro **2** é o número de canais.

Apêndice III. Biblioteca QT

A III.1. Introdução

QT é uma biblioteca da Linguagem C++ e uma ferramenta de programação gráfica que contém um conjunto básico de serviços necessários para maioria do desenvolvimento de aplicações nas plataformas Linux e Windows. O QT foi desenvolvido pela empresa norueguesa Trolltech.

A biblioteca QT fornece para o programador as seguintes funcionalidades:

- Componentes de desenvolvimento de interfaces gráficas.
- Um modelo de interação dos objetos, baseado em sinais e slots.
- Serviços de acesso a arquivos e diretórios independentes da plataforma de trabalho.
- Classes para a manipulação de imagens e texto.
- Uma coleção de classes e funções para as mais diversas aplicações.

A III.2. Uma aplicação em QT

Nesta seção, será apresentada uma aplicação simples construída com a biblioteca QT. Esta aplicação consiste do exemplo básico de todo o livro ou apostila de programação: um programa que mostra na tela a frase “Hello World”.

```
#include <qapplication.h>
#include <qlabel.h>
int main(int argc, char **argv)
{
      QApplication *app = new QApplication(argc, argv);
      QLabel *l = new QLabel(“Hello Qt!”,0,0);
      app->setMainWidget(l);
      l->show();
      app->exec();
      delete app;
      return 0;
}
```

A saída do programa deverá ser como a figura abaixo:

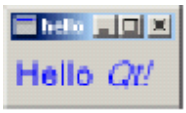

Figura 49. Tela de saída do programa Hello Qt!

Para qualquer programa em QT, existirá um conjunto de arquivos de definição dos objetos do QT. Neste exemplo, os arquivos de definição são **qapplication.h** e **qlabel.h**.

A classe QApplication é requerida para qualquer aplicação que faz uso de eventos ou de interfaces gráficas. Esta classe define o *loop* principal de eventos do programa, aonde todos os eventos do ambiente gráfico e de outras fontes são processados e despachados. Esta classe também manuseia a inicialização e a finalização do programa. O programa de exemplo cria uma instância da classe na pilha e passa os argumentos da linha de comando. Neste exemplo também é criada uma instância da classe QLabel na pilha e ela é inicializada com o texto "Hello World". Na próxima linha, o programa configura que a janela que a classe QApplication deverá gerenciar será a QLabel. Esta classe que é passada para o controle da classe QApplication é chamada de **Widget**. Widgets são elementos visuais que são combinados para criar interfaces gráficas do usuário. Botões, menus, barras de rolagem, caixas de texto e janelas de aplicativos são exemplos de *Widgets*.

Na próxima linha, o programa diz para a classe QLabel mostrar a si mesma, e depois passa o controle da classe para o objeto app.

A III.3. Sinais e Slots

Qualquer aplicação gráfica, que forneça alguma interação com o usuário, é baseada em eventos ou mensagens que deverão executar algum código. Estes eventos devem estar relacionados aos objetos de alguma maneira. O QT define um método de interligar estes eventos com o código; **sinais** e **slots**. Os sinais são enviados pelos objetos QT ou pelos objetos que o programador criou. Este sinal é emitido para notificar algum evento, por exemplo, quando o botão direito do mouse é acionado. Os *slots* são as funções que são interligadas aos sinais para tratá-los.

A definição de sinais e *slots* é fácil. Por exemplo, será considerada a declaração da classe abaixo.

```
class CUas_core: public QObject, public CMessage{
Q_OBJECT
public:
     CUas_core();
     ~CUas_core();
public slots:
     void receiveMessage(struct message_s * message);
     void setDialog(struct dialog_s * dialog);

signals:
     void sendMessage(struct message_s * message);
}
```

Qualquer objeto que implemente sinais e *slots* deverá ser um objeto filho da classe **QObject**. A classe que usar de sinais e *slots* também deve incluir a palavra **Q_OBJECT** no topo da sua declaração. As funções sinais e *slot* sempre deverão retornar um valor **void**.

Neste exemplo, tem-se a definição de um sinal, o **sendMessage**, e dois *slots*, **receiveMessage** e **setDialog**. Os sinais podem entregar alguma informação ao *slot* ao qual o sinal está interligado. Neste exemplo, o sinal sendMessage entrega uma estrutura do tipo **message_s** para o *slot* em que ele está interconectado.

Para se ligar um sinal a um *slot*, utiliza-se a função **connect**.

```
connect(this,SIGNAL(sendMessage(struct message_s
*)),this,SLOT(receiveMessage(struct message_s *)));
```

Quando um objeto deseja emitir algum sinal, ele usa a função **emit**.

```
emit sendMessage(&message);
```

Portanto, quando um evento ocorre e deseja-se chamar alguma função que execute um código específico, basta definir um sinal e um *slot* e interligá-los através da função **connect**.

Apêndice IV. Detalhes da Implementação do Sistema SIP

A IV I. Telefone SIP

O Telefone SIP é um software desenvolvido para o Sistema Operacional Linux. O programa chama-se **QTSIPPhone**. Este programa foi desenvolvido em C++, usando-se a biblioteca QT[25] para a implementação da interface gráfica. A biblioteca QT foi usada devido às vantagens da programação voltada a eventos.

O **QTSIPPhone** possui várias classes que o compõe, sendo que o núcleo do programa foi implementado na classe **QTSIPPhone**, que possui o mesmo nome que o programa. As outras classes do programa implementam o protocolo SIP e o acesso à placa de áudio.

A IV I.I. QTSIPPhone

Esta classe contém o núcleo do Telefone SIP. Nela, as classes que implementam a pilha do protocolo e o acesso à placa de áudio são inicializadas. A Camada de Transporte também foi implementada nesta classe, através da biblioteca de sockets.

Em um programa que usa a biblioteca QT, as classes podem possuir funções declaradas de uma maneira diferente da declaração em C++. Em QT, as funções são classificadas como funções simples, aquelas declaradas como em C++, as funções do tipo **SLOT** e as funções do tipo **SIGNAL**. As funções do tipo **SIGNAL** são funções que não possuem corpo e que definem algum sinal do programa. As funções do tipo **SLOT** são aquelas que são chamadas quando um sinal é enviado.

As principais funções do tipo **SLOT** que compõem o Telefone SIP são:

- **sendMessagefromInviteClient**: Esta função é interligada com a camada de transação através de um sinal. Sempre que a Transação Cliente INVITE quiser enviar uma mensagem, ela envia um sinal, **sendMessage**, contendo a mensagem a ser enviada. Quando esta função é chamada, a mensagem, o endereço IP, a porta e o protocolo a ser usado são passados como argumentos. Com estes dados, a função envia a mensagem.

- **sendMessagefromNonInviteClient**: Esta função envia as mensagens da Transação Cliente não-INVITE. Ela é interligada com a transação através de um sinal, **sendMessage**, definido na Transação Cliente não-INVITE.
- **sendMessagefromInviteServer**: Esta função envia as mensagens da Transação Servidora INVITE, e é interligada com a Camada de Transação através do sinal **sendMessage** definido na Transação Servidora INVITE.
- **sendMessagefromNonInviteServer**: Esta função envia as mensagens da Transação Servidora não-INVITE e é interligada com a Camada de Transação através do sinal **sendMessage**, definido na Transação Servidora não-INVITE.
- **sendMessagefromTU**: Esta função envia as mensagens da Camada de Transação do Usuário. Ela é usada para enviar a mensagem ACK. Esta função é interligada na Camada de Transação do Usuário através do sinal **sendMessage**.
- **initiateAudio**: Esta função é chamada após a finalização da Transação INVITE. O Agente Cliente do Usuário envia um sinal, também chamado **initiateAudio**, contendo o endereço e a porta para o envio dos pacotes de áudio.
- **terminateAudio**: Esta função é chamada para a finalização do fluxo de áudio. Ela pode ser chamada tanto por um sinal vindo do Agente Servidor do Usuário, tanto pelo núcleo do programa.
- **endCall**: Esta função é chamada quando o usuário clica em "Finalizar Chamada". Esta função, então chama a função **terminateAudio**, e envia um sinal para o Agente Cliente do Usuário dizendo para ele gerar a requisição BYE.
- **terminateInviteClientTrans**: Esta função é chamada pelo Agente Cliente para finalizar uma Transação Cliente INVITE, ou é chamada pela própria Transação Cliente para finalizá-la, quando ocorrer um *timeout*.
- **terminateNonInviteClienteTrans**: Esta função é chamada pelo Agente Cliente para finalizar uma Transação Cliente não-INVITE, ou é chamada pela própria Transação Cliente para finalizá-la, quando ocorrer um *timeout*.

- **terminateInviteServerTrans**: Esta função é chamada pelo Agente Servidor ou pela própria Transação Servidora INVITE para finalizá-la.
- **terminateNonInviteServerTrans**: Esta função é chamada pelo Agente Servidor ou pela própria Transação Servidora para finalizá-la.
- **sendAudioPacket**: Esta função é utilizada para enviar os pacotes de áudio através dos *sockets* criados na função **initiateAudio**.
- **receiveAudioPacket**: Esta função é utilizada para receber os pacotes de áudio através dos sockets criados na função **initiateAudio**.

A outra classe de funções que compõe a classe **QTSIPPhone** são as funções do tipo **SIGNAL**.

- **sendMessagetoInviteClient**: Este sinal serve para entregar a mensagem recebida pela Camada de Transporte para a Transação Cliente INVITE. **sendMessagetoInviteServer**: Este sinal serve para entregar a mensagem recebida pela Camada de Transporte para a Transação Servidora INVITE.
- **sendMessagetoNonInviteClient**: Este sinal serve para entregar uma mensagem recebida pela Camada de Transporte para a Transação Cliente não INVITE.
- **sendMessagetoNonInviteServer**: Este sinal serve para entregar uma mensagem recebida pela Camada de Transporte para a Transação Servidora não-INVITE.
- **sendMessagetoClient**: Este sinal é ligado ao Agente Cliente do Usuário e serve para entregar as retransmissões da resposta **200 Ok** diretamente ao Agente Cliente do Usuário.
- **sendMessagetoServer**: Este sinal é ligado ao Agente Servidor do Usuário e serve para receber as retransmissões do **ACK** que o terminal possa a vir receber.
- **sendAudioPackettoAudioCard**: Este sinal envia um pacote de áudio recebido para a classe que trata do acesso à placa de áudio do sistema.

A IV I.II. CUac_core

Esta classe é a que implementa as funcionalidades do Agente Cliente do Usuário. Ela é composta de funções, que se comunicam com as outras camadas do programa, para efetuar as transações SIP.

As funções que geram as mensagens têm o seguinte formato: **generatexxx()**, onde **xxx** é o nome da requisição. Por exemplo, para a requisição **INVITE**, a função se chama **generateInvite()**. Estas funções geram as mensagens e guardam os valores dos campos enquanto durar uma sessão, ou somente para a transação. A exceção é a mensagem **REGISTER**, aonde os *tags* e identificadores da mensagem devem ser os mesmos para cada registro efetuado ou atualizado em um mesmo servidor de registros. Existem, também, as funções em que o agente cliente do usuário interage com o núcleo do programa. Estas funções são do tipo **SLOT**.

- **setReceiverAddress:** Esta função é usada para configurar o endereço do receptor da mensagem SIP.
- **setRegistrarName:** Esta função configura o endereço do servidor de registros deste usuário. A partir desta configuração, o endereço SIP do usuário será o seu *username* mais o endereço do servidor de registros.
- **setUsername:** Esta função configura o nome do usuário que será usado nas transações SIP.
- **terminateSession:** Esta função é interligada ao núcleo do programa e é chamada quando o usuário clica em **Finalizar Chamada**. Quando esta função é chamada, o Agente Cliente do Usuário gera a requisição BYE e finaliza o fluxo de áudio, através do sinal **terminateAudio**.
- **slotInvite:** Esta função é chamada quando o usuário deseja iniciar uma chamada. Ela chama a função **generateInvite**, para gerar a mensagem e alterar o estado do Agente Cliente do Usuário.
- **slotRegister:** Esta função é chamada quando o usuário deseja iniciar um registro. Esta função chama a função **generateRegister** e altera o estado do Agente Cliente do Usuário.

Esta classe também possui funções do tipo **SIGNAL** para interagir com o núcleo do programa. São elas:

- **initiateAudio:** Este sinal indica, para o núcleo do programa, que ele pode iniciar o fluxo de áudio. Este sinal passa, como argumento, o endereço que a camada de Transporte deverá usar para enviar os pacotes de áudio.
- **terminateAudio:** Este sinal indica que o usuário deseja finalizar a chamada e que o fluxo de áudio deve ser finalizado.

Existem, também, as funções de interação com as camadas de Transação e de Transporte. As funções do tipo **SLOT** são:

- **receiveMessageTrans:** Esta função é utilizada pelo agente cliente do usuário para receber as mensagens da Camada de Transação. Quando uma mensagem é recebida, o Agente Cliente verifica o código de status da mensagem e a transação que a mensagem está envolvida para tomar alguma ação. Esta ação vai depender da transação envolvida. Para a transação REGISTER, o agente cliente do usuário, ao receber uma resposta **200 Ok**, muda para o estado **REGISTRADO**. Para o **INVITE**, o recebimento de um **200 Ok** faz o Agente Cliente do Usuário mudar para o estado **CHAMADA**, gerar um **ACK**, criar o **Diálogo**, e ficar aguardando, por 32 segundos, as possíveis retransmissões da resposta **200 Ok**. Para o **BYE**, as sessões e o diálogo são finalizados.
- **receiveMessage:** Esta função é responsável pela interação com a Camada de Transporte diretamente. Após a troca de mensagens **INVITE/200 Ok**, o Agente Cliente do Usuário deverá enviar o **ACK** para confirmar o recebimento de um **200 Ok** e avisar a outra parte para iniciar o fluxo de áudio. Mas, a outra parte pode não receber o **ACK** e retransmitir o **200 Ok**. Esta função recebe as retransmissões do **200 Ok** e retransmite o **ACK**, através da função **send_message**.
- **slotTimeout:** Esta função é responsável por receber o sinal de **Timeout** enviado pela Camada de Transação. Esta função, então, sinaliza para o usuário de um *timeout*, finaliza a transação e altera o estado do Agente Cliente do Usuário.

Existem, também, as funções do tipo **SIGNAL** para interagir com as camadas de Transação e de Transporte.

- **sendMessageTrans:** É a função que envia uma mensagem para a transação. Ela é usada pelo Agente Cliente do Usuário no início da transação.
- **sendMessage:** É a função que envia o **ACK** diretamente para a Camada de Transporte quando o Agente Cliente do Usuário recebe uma retransmissão da mensagem **200 Ok**.

O Agente Cliente também interage com o Agente Servidor através de duas funções:

- **setDialog:** Esta função do tipo **SLOT** é usada para copiar as peças de um diálogo estabelecido pelo Agente Servidor do Usuário.
- **sendDialogtoUas:** Esta função do tipo **SIGNAL** é usada para enviar o diálogo estabelecido pelo Agente Cliente do Usuário.

A IV I.III. CUas_core

A classe **CUas_core** implementa as funções do núcleo do Agente Servidor do Usuário.

Esta classe também é composta por funções que interagem com as Camadas de Transação e de Transporte, com o Agente Cliente do Usuário, com o núcleo do programa e com a Camada de Acesso á Placa de Áudio.

Para a interação com as camadas de Transação e Transporte, as funções do tipo **SLOT** são:

- **receiveMessageTrans:** Esta função recebe a mensagem da Camada de Transação e atualiza o estado do Agente Servidor dependendo da requisição. Nesta função, as requisições são processadas e as mudanças de estado, que poderão ocorrer, são feitas.
- **receiveMessage:** Após a confirmação de um estabelecimento de uma sessão INVITE, o Agente Servidor fica aguardando o recebimento de uma resposta ACK. Esta função é interligada à Camada de Transporte e serve para receber o ACK diretamente.

As funções do tipo **SIGNAL** são:

- **sendMessageTrans:** Esta função é usada para enviar as respostas para a Camada de Transação.

- **sendMessage:** Esta função é usada para retransmitir a resposta 200 Ok diretamente para a Camada de Transporte.

Existem as funções de interação com o núcleo do programa. Estas funções compreendem de cinco sinais e uma função do tipo **SLOT**.

- **inviteAccept:** Esta é a função do tipo **SLOT** que o núcleo do programa usa para avisar ao Agente Servidor se o usuário aceitou a chamada ou não.
- **inviteReceived:** Esta função é do tipo **SIGNAL** e é enviado para o núcleo do programa para avisar ao usuário que um convite de conexão chegou e para questionar ao usuário se ele deseja aceitar ou não.
- **initiateAudio:** Esta função do tipo **SIGNAL** é usada para iniciar o fluxo de áudio. O argumento desta função é o endereço do destinatário dos pacotes de áudio.
- **terminateAudio:** Esta função do tipo **SIGNAL** é usada para finalizar o fluxo de áudio.
- **testAudioFormat:** Esta função do tipo **SIGNAL** é usada para testar um formato de áudio descrito pela mensagem SDP.
- **setAudioFormat:** Esta função do tipo **SIGNAL** é usada para especificar uma configuração de áudio visando o estabelecimento do fluxo de áudio.

Como no Agente Cliente, o Agente Servidor possui funções para interagir como Agente Cliente. Estas funções são:

- **setDialog:** Esta função é do tipo **SLOT** e é usada pelo Agente Cliente para configurar um diálogo estabelecido por ele.
- **sendDialogtoUac:** Esta função é do tipo **SIGNAL** e é usada pelo Agente Servidor para enviar um diálogo estabelecido pelo Agente Servidor.

A IV I.IV. ClInvite_client_trans

Esta é a classe que implementa as funcionalidades da Transação Invite Cliente. Ela possui funções de interação com as outras classes do programa e consigo mesmo. Esta classe foi implementada conforme mostrado na Figura 35.

- **receiveMessage:** Esta função, do tipo **SLOT**, interage com a camada de transporte para o recebimento das respostas ao **INVITE** enviado. Dependendo

da resposta recebida e do estado atual, a transação muda de estado. Após o recebimento da mensagem, nesta mesma função, ela é processada. Aqui é que é gerado o **ACK** para as respostas **300 – 699**.

- **receiveMessageTu:** Esta função é responsável por receber a requisição **INVITE** e inicializar o funcionamento da transação conforme a máquina de estados da Figura 35. Após esta inicialização, a transação envia a mensagem para a Camada de Transporte. Esta função é do tipo **SLOT**.
- **sendMessage:** Esta função, do tipo **SIGNAL**, envia a requisição **INVITE** para a Camada de Transporte.
- **sendMessageTu:** Esta função, do tipo **SIGNAL**, serve para enviar as respostas recebidas pela transação para a transação do usuário.
- **timeout:** Esta função avisa a transação do usuário de um **timeout** na transação. Esta função é do tipo **SIGNAL.**

A IV I.V. Cnon_Invite_Client_trans

Esta classe implementa as funcionalidades da Transação Cliente não-INVITE. Ela foi implementada conforme a máquina de estados mostrada na Figura 36.

Como a Transação Cliente INVITE, esta classe possui as seguintes funções:

- **receiveMessage:** Nesta classe, esta função é quem recebe as respostas às requisições enviadas. Para transações não-INVITE, o recebimento de respostas finais não é acompanhado pelo envio do **ACK**. Esta função é do tipo **SLOT**.
- **receiveMessageTu:** Esta função recebe a requisição da Transação do Usuário e inicia o funcionamento da transação. Esta função é do tipo **SLOT.**
- **sendMessage:** Envia a requisição para a Camada de Transporte. Esta função é do tipo **SIGNAL**.
- **sendMessageTu:** Envia a resposta recebida para a Transação do Usuário. É uma função do tipo **SIGNAL**.
- **timeout:** Avisa à Transação do Usuário quando ocorrer um *timeout* na transação. É uma função do tipo **SIGNAL**.

A IV I.VI. CInvite_server_trans

Esta classe é a que implementa a Transação Servidora INVITE. Esta classe interage com o Agente Servidor do Usuário. Ela foi implementada conforme a máquina de estados mostrada na Figura 37.

As funções principais desta classe são:

- **receiveMessageTu:** Esta função recebe as resposta enviadas pela Transação do Usuário a um **INVITE** recebido. Diferente da implementação das transações cliente, esta função pode receber mais de uma mensagem da Transação do Usuário. Isto se deve ao fato de que a Transação do Usuário pode enviar várias respostas provisionais a um **INVITE** recebido. Dependendo da resposta recebida da Transação do Usuário e do estado atual, a Transação Servidora muda de estado. É uma função do tipo **SLOT.**
- **receiveMessage:** Esta função recebe as requisições da Camada de Transporte e gera a resposta **100 Trying** para conter as retransmissões do **INVITE**. É uma função do tipo **SLOT.**
- **sendMessageTu:** Esta função é responsável por enviar o **INVITE** à Transação do Usuário. É uma função do tipo **SIGNAL**.
- **sendMessage:** Esta função envia as resposta enviadas pela Transação do Usuário para a Camada de Transporte. É uma função do tipo **SIGNAL**.
- **timeout:** Esta função avisa de um *timeout* na transação. É uma função do tipo **SIGNAL**.

A IV I.VII. Cnon_Invite_Server_trans

Esta classe é a que implementa as funcionalidades da Transação Servidora não-INVITE. Para esta classe, as funções são:

- **receiveMessageTu:** Esta função recebe as respostas da Transação do Usuário à requisição recebida e muda de estado. É uma função do tipo **SLOT**.
- **receiveMessage:** Esta função recebe as requisições da Camada de Transporte. Esta função é quem inicializa a transação conforme a máquina de estados mostrada na Figura 38. Esta função é do tipo **SLOT**.

- **sendMessageTu:** Esta função envia a mensagem para a Transação do Usuário. É uma função do tipo **SIGNAL.**
- **sendmessage:** Esta função envia as respostas recebidas pela Transação do Usuário para a Camada de Transporte. É uma função do tipo **SIGNAL.**

A IV I.VIII. CMessage

A classe **CMessage** é responsável pela manipulação das mensagens recebidas. Todas as outras classes usam esta classe para a manipulação de mensagens.

A classe **CMessage** é composta pelas seguintes funções:

- **returnSdpHeader:** Esta classe retorna o valor de um campo da mensagem SDP recebida. O campo, ao qual se quer o valor, e a mensagem são passados como argumentos da função. A função retorna uma *string* contendo o valor do campo.
- **returnAudioPort:** Esta função é usada pelo agente servidor do usuário para obter a porta ao qual o fluxo de áudio será trocado. O valor retornado pela função é uma *string* contendo a porta.
- **returnHeaderField:** Esta função retorna o campo da mensagem SIP. A mensagem e o campo, ao qual se quer, são passados como argumento. A função retorna uma *string* contendo o valor do campo inteiro.
- **returnMethod:** esta função é usada pela parte servidora do protocolo SIP. Ela retorna o método da mensagem. O método é retornado através de uma *string*, e a mensagem, a qual se quer saber o método, é passada como parâmetro de entrada da função.
- **returnParamValue:** Esta função retorna o valor do campo da mensagem SIP sem incluir os *tags*. A mensagem e o campo são passados como argumentos de entrada e a função retorna uma *string* contendo o valor do campo.
- **returnStatusCode:** Esta função retorna o código de *status* da resposta SIP. Ela é muito utilizada no lado cliente do protocolo SIP. A mensagem é passada como argumento e a saída é uma *string* contendo o código de *status*.
- **returnTagValue:** Esta função retorna o valor da *tag* do campo **To** ou **From**. Esta função é utilizada tanto pelo agente servidor do usuário como pelo agente

cliente do usuário para se montar o diálogo. O campo o qual se deseja a *tag* e a mensagem são passados como argumentos e a função retorna uma *string* contendo a *tag* desejada.

- **returnTypeofMedia:** Esta função é usada pelo agente servidor do usuário para se obter os valores do parâmetro **a**, da mensagem SDP, referentes à taxa de amostragem e o tipo de codificação.

A IV I.IX. CMyAudio

A classe **CMyAudio** é a que permite ao Terminal SIP acessar a placa de áudio. O acesso à placa de áudio permite a leitura e a escrita de pacotes de áudio. Também, é possível configurar o dispositivo, conhecendo-se a mascara de *bits* a qual o dispositivo é programável.

Existem vários modelos e marcas de placas de áudio no mercado. Em geral, as placas de áudio possuem algumas configurações que são comuns a todas as placas. Estas configurações foram vistas na seção 5.2.4. Esta classe é composta de funções que configuram estes parâmetros da placa de áudio.

- **openAudioDevice:** Esta função abre o dispositivo de áudio e o configura com os valores padrão de uma chamada telefônica.
- **closeAudioDevice:** Esta função fecha o dispositivo de áudio.
- **setSpeed:** Esta função serve para configurar o valor da taxa de amostragem da placa de áudio. Geralmente, os valores possíveis estão na faixa de 5 KHz a 96 KHz.
- **setChannels:** Esta função configura o número de canais em que a placa de som pode trabalhar. Para chamadas de telefonia, usa-se um único canal (mono).
- **setAudioFormat:** Esta função configura o formato da amostra de áudio. Dependendo da placa de som, os formatos vão de amostras simplesmente binárias a amostras codificadas por algum *codec*.
- **setBufferSize:** Esta função configura o tamanho do fragmento da placa de som. O valor padrão para as placas de som é de 2048 *bytes*. Algumas placas de som não permitem alterar este valor.

- **testFormat:** Esta função é usada pelo Agente Servidor do Usuário para testar um formato recebido em uma mensagem SDP.
- **reset:** Esta função coloca a placa de som na sua configuração inicial.
- **initiateAudio:** Esta função serve para iniciar o fluxo de áudio.
- **returnAudioFormat:** Esta função retorna a configuração atual da placa de som.

Esta classe também possui funções do tipo **SLOT** e **SIGNAL**. São elas:

- **writeData:** Esta função é interligada ao núcleo do programa e serve para escrever um pacote de áudio da rede na placa de som. Esta função é do tipo **SLOT.**
- **sendData:** Esta função é interligada ao núcleo do programa e serve para enviar um pacote de áudio lido da placa de som. Esta função é do tipo **SIGNAL**.

A IV II. Proxy e o Servidor de Registros

O Proxy e o Servidor de Registros foram implementados em um único programa, na Linguagem C++. O Proxy possui uma camada de processamento das mensagens e o acesso ao banco de dados do Servidor de Registros. O Servidor de Registros possui um Agente Servidor que processa somente requisições REGISTER e OPTIONS. As demais requisições são respondidas com uma resposta contendo o código de *status* apropriado.

As classes que compõem o Proxy e o Servidor de Registros estão descritas nos itens a seguir.

A IV II.I. CProxy

A classe **CProxy** implementa um Proxy sem estados. Este Proxy recebe a mensagem, faz todo o processamento da mensagem e a envia para o destinatário. O Proxy sem estados só não monta uma transação para enviar as mensagens, isto é, não se comporta como se fosse o Agente Cliente que gerou a requisição.

Esta classe é composta de duas funções:

- **receiveMessage:** Esta função é responsável por receber as mensagens, processá-las e por tomar as decisões de roteamento. Se a mensagem for para o Servidor de Registros, o Proxy entrega a mensagem para ele. Se não for, ele envia a mensagem para uma outra entidade SIP.
- **sendMessage:** Esta função é responsável por enviar as mensagens processadas pelo Proxy. Nesta função, os campos da mensagem que precisam ser atualizados são alterados.

A IV II.II. CRegistrar

A classe **CRegistrar** implementa as funcionalidades de um Servidor de Registros. Na RFC 3261[1], um Servidor de Registros é chamado de REGISTRAR.

Esta classe é composta pelas seguintes funções:

- **searchContact:** Esta função é usada pelo Servidor de Registro para procurar por um contato no seu banco de dados. Se o contato existir, esta função retorna uma *string* contendo o endereço do contato desejado. Se ele não existir, ela retorna uma *string* nula.
- **sendResponse:** Esta função envia uma resposta a uma requisição recebida. Esta resposta informa o status de um registro efetuado ou alterado, ou indica um erro, por causa de uma requisição que este servidor não aceita.
- **receiveMessage:** Esta função recebe a mensagem do Proxy. Após isto, o Servidor de Registros começa o processamento da requisição.
- **addContact:** Esta função é usada pelo Servidor de Registros para adicionar um contato no seu banco de dados.
- **removeContact:** Esta função é usada pelo Servidor de Registros para remover um contato da lista de contatos.
- **searchContact:** Esta função pesquisa por um contato na lista de contatos do servidor de registros.