Instituto Politécnico de Leiria
Escola Superior de Tecnologia e Gestão

Projecto II
Eng. Informática e Comunicações 5º ano 1º semestre

IPv6@ESTG-Leiria: VoIP em IPv6

Relatório

Aluno: Hugo Alexandre de Oliveira
Nº Aluno: 08705
Orientadores: António Pereira, Mário Antunes, e Nuno Fonseca

15 de Fevereiro de 2006

Identificação

Aluno

Nome: Hugo Alexandre de Oliveira

E-Mail: eic08705@student.estg.ipleiria.pt

Orientador1

Nome: António Manuel de Jesus Pereira

E-Mail: apereira@estg.ipleiria.pt

Orientador2

Nome: Mário João Gonçalves Antunes

E-Mail: mario.antunes@estg.ipleiria.pt

Orientador3

Nome: Nuno Miguel da Costa Santos Fonseca

E-Mail: nfonseca@estg.ipleiria.pt

Área temática

Redes e serviços

Agradecimentos

Aos meus colegas da cadeira de Projecto II, Tiago Amado e ao Rui Bernardo que deu o seu apoio durante todo o projecto e a nível da aplicação MGEN.

Ao professor António Pereira pela ajuda e apoio a nível de QoS, e no relatório.

Aos Orientadores.

Ao professor Paulo Loureiro na ajuda a nível da aplicação do Chariot, e configuração QoS.

Ao Vítor dos Santos que ajudou na configuração de DNSv6.

Ao Faycal Hadj da *Cisco Systems* pela disponibilidade imediata no esclarecimento de dúvidas por e-mail.

Aos meus pais e ao director de curso Luis Távora pelo apoio e motivação.

A todos os que contribuíram directa ou indirectamente para a realização deste projecto.

Resumo

Recorrendo às potencialidades da Internet, actualmente é possível usar o protocolo IP para transportar os serviços tradicionais de comunicação, tais como o telefone e o fax num único canal. Efectivamente, nos últimos anos, tem-se verificado um aumento significativo da telefonia IP, sendo que, um dos factores que mais tem contribuído para este crescimento é a possibilidade de uma forte redução do custo das comunicações, quer ao nível global, quer até mesmo ao nível interno de uma organização.

O protocolo *Session Initiation Protocol* (SIP), definido recentemente, é um dos grandes responsáveis pelo crescimento das comunicações multimédia em redes IP, dado que fornece os mecanismos necessários para que diferentes terminais consigam iniciar as sessões na rede. O VoIP e o *instant messaging* são alguns exemplos de aplicações que usam o referido protocolo para início de secção.

A Empresa *Cisco Systems*[1] utiliza um outro protocolo de sinalização proprietário chamado *Skinny Client Control Protocol* (SCCP), no entanto também tem soluções para a integração do protocolo SIP e do H.323. Este último, é um protocolo bastante relevante, uma vez que permite a interoperabilidade com as redes do tipo Rede Digital com Integração de Serviço (RDIS) e analógicas ou *Plain Old Telephone Service* (POTS).

Actualmente a voz sobre IPv6, ainda está numa fase muito imatura. Existem poucas soluções de voz em IPv6 comerciais sendo que presentemente, por exemplo, a Cisco ainda não suporta IPv6 nas suas soluções de voz. Na comunidade *Open source*, já existem algumas soluções que implementam o protocolo SIP, destacando-se implementação de servidores IPv6 IPTEL[2] e VOCAL[3]. A nível de terminais *softphone,* existem poucas aplicações funcionais, sendo que as que existem possuem alguns *bugs* ou problemas a nível da camada da aplicação do modelo OSI.

No decorrer deste trabalho verificou-se que as soluções disponibilizadas pela *Cisco Systems* não suportam IPv6, assim procurou-se encontrar soluções para implementar voz IPv4 sobre uma rede IPv6, recorrendo a mecanismos de transição IPv4 para IPv6. Dado que a voz sobre IP é sensível ao tráfego existente na rede, procurou-se formas de oferecer QoS (*Quality of*

[1] Cisco Systems URL: http://www.cisco.com

[2] SIP Express Router (SER) URL: http://www.iptel.org/

[3] VOCAL URL: www.vovida.org/

Service) ou Qualidade de Serviço em IPv6. Também foi realizado um cenário com a implementação de VoIP em IPv6 nativo, usando soluções *Open Source*.

Palavras-chave: Telefonia sobre IP, voz sobre IP, Protocolos, QoS, Mecanismos de transição IPv4 para IPv6

Abstract

IP telephony has came a long way, and in these last few years, there has been a great boom in its utilization, because it's cost effective, on both *Local Area Network* (LAN) and *Wide Area Network* (WAN) (especially for external phone calls). Thanks to broadband, applications like audio and video (bandwidth on demand) are now possible, and have better quality. Traditional services like fax and Plain Old Telephone Service (POTS) telephony can be integrated over the IP network. The Session Initiation Protocol (SIP) can easily be integrated for the Internet, and is responsible for session initiations of audio (especially VoIP), video, instant messaging, and other formats.

The Cisco Systems company has its own session control protocol, called the Skinny Client Control Protocol or SCCP, but they also support SIP, and other protocols.

Another very important protocol is the H.323, because of its interoperability with other telephony more traditional technologies like Integrated Services Digital Network (ISDN) and POTS.

Implementing IP telephony is easy because it uses the existing IP infrastructure of the institution or company.

Nowadays, voice over IPv6 still has a long way to go, Cisco Systems for example, doesn't support IPv6 on any of its VoIP solutions. There are no or very few commercial solutions for VoIPv6. However there are one or two Open Source servers (IPTEL and VOCAL) that use the SIPv6 protocol. There are very few SIPv6 terminals most of them are difficult to install, and are full of application bugs.

This project has the objective of integrating a Cisco Systems VoIPv4 solution over an IPv6 backbone, using translation mechanisms IPv4 to IPv6. Quality of service also had to be tested on the IPv6 backbone, because voice over IP is very sensitive to traffic that passes through the IPv6 network.

Another scenario, is the implementation of a native Voice over IPv6 (VoIPv6) solution.

Keywords: IP telephony, Protocols; Voice over IP; QoS; IPv4 to IPv6 translation mechanisms

Índice

Identificação ..... **III**
Agradecimentos ..... **V**
Resumo ..... **VII**
Abstract ..... **X**
Índice ..... **XII**
Índice de figuras ..... **XVI**
Índice de tabelas ..... **XX**
Lista de Siglas/Acrónimos ..... **XXII**
1. Introdução **1**
1.1. Objectivo do Trabalho 3
1.2. Planeamento 4
1.3. Estrutura do relatório 5
2. Telefonia IP **6**
2.1. Componentes básicos 6
2.2. Arquitectura protocolar 10
2.2.1. Sinalização *10*
2.2.2. Codificadores *17*
2.2.3. Transporte *17*
2.2.4. Formatos Payload *19*
2.2.5. Largura de banda utilizada pelo protocolo RTP e de sinalização *19*
3. Métodos de translação **26**
3.1.1. Introdução *27*
3.1.2. Dual Stack *27*
3.1.3. Tunnelling IPv6 sobre IPv4 ..... *28*
3.1.4. DSTM *34*
3.1.5. NAT-PT *35*
3.1.6. Application Layer Gateways *36*
3.1.7. Proxy *38*
4. Mecanismos de Qos **39**
4.1. Introdução 39
4.1.1. Enquadramento *39*
4.2. Parâmetros de QoS 40
4.3. A solução ToS/IP precedence (antecessor) 42
4.4. Intserv 44
4.4.1. RSVP *44*
4.5. DiffServ 46
4.5.1. Arquitectura DiffServ *47*
4.5.2. Marcação de pacotes *51*
4.5.3. Comportamentos por salto *51*
4.5.4. PHB por defeito (definido no RFC 2474) *51*
4.5.5. Class-Selector PHBs (definido no RFC 2474) *52*
4.6. Mecanismos de QoS no IPv6 52
4.6.1. Campos relevantes do IPv6 para QoS *53*
4.6.2. Problemas com DiffServ e IntServ *54*
4.6.3. Os campos Traffic Class e Flow Label *56*
4.6.4. Observações sobre o QoS no IPv6 ..... *58*
4.6.5. QoS suportado no IOS da Cisco para IPv6 [23] *59*
4.6.6. Conclusões sobre QoS *60*
5. VoIP em IPv6 nativo **61**

5.1. Introdução ... 61
5.2. Arquitectura SIP ... 61
5.2.1. Universal Resource Identifier (URI) ... *63*
5.2.2. Mensagens SIP ... *64*
5.3. Soluções de Integração ... 65
5.3.1. Métodos de translação IPv4/IPv6 e vice-versa no SER ... *66*
5.3.2. Integração do telefone IP 7960 da Cisco no SER ou VOCAL ... *67*
5.3.3. Gateway de SIP para POTS ... *68*
5.3.4. Gateway do SIP para PSTN/ISDN ... *68*
5.3.5. Gateway do SIP para H.323 ... *69*
5.3.6. Outros servidores e serviços ... *69*
6. Testes ... 73
6.1. Verificação prática dos protocolos utilizados ... 73
6.1.1. Mensagens no registo do telefone ... *74*
6.1.2. Telefonema entre telefones IP ... *75*
6.1.3. Chamada do telefone IP 7960 para o telefone analógico ... *76*
6.1.4. Chamada realizado num Cluster de CallManagers ... *77*
6.2. Implementação prática do túnel "4 to 6" GRE ... 78
6.2.1. Overhead introduzido pelo o túnel GRE ... *78*
6.3. Testes de QoS no cenário ... 81
6.3.1. Prioritização do tráfego ... *82*
6.3.2. Classificação do tráfego ... *83*
6.3.3. Controlo de admissão ... *85*
6.3.4. Testes Iniciais de QoS ... *86*
6.3.5. Testes finais com tráfego IPv4 e IPv6 no cenário ... *88*
6.3.6. Alternativas para o QoS para IPv6 ... *102*
6.4. Implementação dum cenário VoIP em IPv6 nativo ... 103
6.4.1. DNSv6 ... *103*
6.4.2. Verificação das mensagens SIPv6 ... *104*
7. Conclusão ... 109
8. Anexos ... 111
8.1. Calendarização do projecto ... 111
8.2. Material utilizado ... 112
8.3. Telefonia IP *hardware* e *software* para IPv6 ... 116
8.3.1. Softphones ... *116*
8.3.2. Hardphones ... *124*
8.3.3. Servidores ... *126*
8.3.4. Gateways ... *133*
8.3.5. Testes ... *134*
8.4. Configurações no CallManager 3.1 ... 135
8.4.1. Subscrição de um User e configuração do telefone ... *136*
8.4.2. Adicionar um telefone IP Phone 7960 ... *136*
8.4.3. Criação de Users ou utilizadores ... *137*
8.4.4. Associar Users aos telefones 7960, 7910 ... *138*
8.4.5. Actualizar o telefone ... *138*
8.4.6. Configuração de speed dials ... *140*
8.4.7. Configurações de Rede nos telefones IP da Cisco ... *141*
8.5. Nova versão do CallManager ... 143
8.6. Listagem dos tipos de mensagens Skinny [9] ... 144
8.7. Lista de Codecs para o telefone analógico ... 146
8.8. Descrição do Protocolo GRE Cabeçalho ... 147
8.9. Cabeçalhos do protocolo RTP ... 150
8.10. Mensagens Skinny ... 152
8.10.1. Processo de registo do telefone ... *152*

8.10.2. Processo de registo ... *152*
8.10.2.1. Implementação do DHCP ... *153*
8.10.2.2. Trivial File Transfer Protocol (TFTP) ... *154*
8.10.2.3. Se o 7960 liga e o 7910 não está disponível ... *157*
8.10.3. Telefonema entre telephones IP ... *159*
8.10.4. Do telefone IP para o analógico ... *162*
8.10.5. Do telefone analógico para o telefone IP 7960 ... *164*
8.11. Configurações para o segundo cenário túnel GRE ... 167
8.12. Configurações para os Teste QoS ... 174
8.12.1. Instalação do Chariot da NetIQ ... *174*
8.12.2. MGEN ... *175*
8.12.3. Atribuição do valor DSCP no Chariot para marcar os Pacotes ... *177*
8.12.4. Alteração da stream RTP do Chariot ... *178*
8.12.5. Valor possíveis para o dscp no Router ... *180*
8.13. MGEN 4.2 [40] ... 181
8.13.1. Características da ferramenta ... *182*
8.13.2. Opções da linha de comandos ... *184*
8.13.3. Exemplo de utilização ... *185*
8.13.4. Exemplo de utilização ... *188*
8.13.5. Formato dos ficheiros de log criados pelo MGEN ... *188*
8.14. Configurações dos Routers cenário QoS ... 190
8.14.1. No Router1 ... *190*
8.14.2. No Router2 ... *193*
8.15. Mensagens SIPv6 ... 196
8.16. Manual de instalação do SER da IPTEL ... 198
8.17. Ficheiros para o DNSv6 ... 200
8.18. Manual de instalação do Linphone ... 203
8.19. Registo do softphone no SER ... 207
8.20. Terminação da sessão com o SER ... 208
8.20.1. Problemas anteriores com o Linphone versão 1.1 ... *208*
8.20.2. Mensagens DNSv6 ... *210*
9. Bibliografia ... 213

Índice de figuras

Figura 1 Cenário VoIP em IPv6 nativo 3
Figura 2 Cenário a implementar 4
Figura 3 Media Gateway Controller 8
Figura 4 Arquitectura de telefonia [60] 10
Figura 5 H.323 sinalização 13
Figura 6 Protocolo MGCP 14
Figura 7 Protocolo SIP arquitectura 15
Figura 8 Protocolos de transporte e de sinalização 18
Figura 9 O cabeçalho IP forma uma parte significativa em pacotes (com *payload* pequeno) . 21
Figura 10 Cenário a implementar 26
Figura 11 *Dual stack* 28
Figura 12 Encapsulamento dos dados IPv6 sobre IPv4 29
Figura 13 Cenário da utilização de um túnel 29
Figura 14 Esquema de endereçamento 6to4 30
Figura 15 tunnel 6to4 (imagem da Cisco) 30
Figura 16 Forma de encapsulamento GRE, cabeçalhos 31
Figura 17 Arquitectura DSTM 34
Figura 18 Diagrama lógico do NAT-PT 35
Figura 19 NAT-PT tradução de endereços 36
Figura 20 Selector ALG 37
Figura 21 Cálculo do *Jitter* 41
Figura 22 O byte original *ToS* no *IPv4* 43
Figura 23 O campo *Codepoint* do *Diffserv* [25] 43
Figura 24 Arquitectura DiffServ 48
Figura 25 Traffic Conditioner Block (TCB) do DiffServ 49
Figura 26 Cabeçalhos IPv4 e IPv6 50
Figura 27 Atribuição dos prefixos 52
Figura 28 Cabeçalho dos pacotes IPv6 53
Figura 29 Violação de camada por parte dos *routers* no IPv4. 55
Figura 30 Encriptação ao nível da camada 3 55
Figura 31 Exemplo de fragmentação de um pacote. 56
Figura 32 Arquitectura SIP em diferentes aplicações e ambientes 66
Figura 33 Implementação do cenário explicado no Cookbook (ver nota de rodapé 19) 69
Figura 34 Exemplo da aplicação do ENUM 70
Figura 35 Cenário de teste IPv4, análise dos protocolos 73
Figura 36 Resumo das mensagens no processo de registo do telefone 74
Figura 37 Mensagens trocadas na chamada entre telefones IP 75
Figura 38 Chamada do telefone IP para o analógico 76
Figura 39 Comunicação entre os equipamentos VoIP 77
Figura 40 Implementação prática de um túnel GRE 78
Figura 41 Pacote RTP em IPv4 fora do túnel GRE 79
Figura 42 Pacote RTP dentro do túnel GRE portanto IPv6 80
Figura 43 Cenário de teste completo 81
Figura 44 Fila para VoIP sobre uma WAN (Router) 82
Figura 45 O valor de DSCP tem o valor de 1A para a sinalização *skinny* 84
Figura 46 O valor de DSCP tem o valor de 1A para a sinalização TPKT, H.245 84
Figura 47 Configuração da admissão ao controlo no CCM 86

Figura 48 Cenário de teste inicial 86
Figura 49 Gráfico das perdas 87
Figura 50 Quando existe congestionamento o *throughput* diminui (7,074, 7,166,7,606) 87
Figura 51 Método utilizado Class-based queuing 90
Figura 52 *Jitter* dos fluxos VoIP (4 a 6ms) na presença de tráfego BE de 64kbps 92
Figura 53 *Jitter* dos fluxos VoIP(8 a 10ms) na presença de tráfego BE de 128kbps 92
Figura 54 *Jitter* dos fluxos VoIP (14 a 16ms) na presença de tráfego BE de 256kbps 93
Figura 55 *Jitter* dos fluxos VoIP (14 a 16ms) na presença de tráfego BE de 512kbps 93
Figura 56 *Jitter* dos fluxos VoIP na presença de tráfego BE de 64kbps 94
Figura 57 *Jitter* dos fluxos VoIP na presença de tráfego BE de 128kbps 94
Figura 58 *Jitter* dos fluxos VoIP na presença de tráfego BE de 256kbps 94
Figura 59 *Jitter* dos fluxos VoIP na presença de tráfego BE de 512kbps 95
Figura 60 *Jitter* dos fluxos VoIP na presença de tráfego BE de 64kbps 96
Figura 61 *Jitter* dos fluxos VoIP na presença de tráfego BE de 128kbps 97
Figura 62 *Jitter* dos fluxos VoIP na presença de tráfego BE de 256kbps 97
Figura 63 *Jitter* dos fluxos VoIP na presença de tráfego BE de 512kbps 97
Figura 64 *Jitter* dos fluxos VoIP na presença de tráfego BE de 64kbps 98
Figura 65 *Jitter* dos fluxos VoIP na presença de tráfego BE de 128kbps 98
Figura 66 *Jitter* dos fluxos VoIP na presença de tráfego BE de 256kbps 99
Figura 67 *Jitter* dos fluxos VoIP na presença de tráfego BE de 512kbps 99
Figura 68 Cenário implementado para SIPv6 103
Figura 69 Registo de um telefone no Servidor SIP 105
Figura 70 Estabelecimento, RTP, terminação 106
Figura 71 SIP com *Instante messaging* 107
Figura 72 Mensagem “ola” em texto enviado 107
Figura 73 Registo de um terminal no Servidor 107
Figura 74 Calendarização do projecto 111
Figura 75 Telefones IP da Cisco modelos 7910 (à esquerda) e o 7960 (à direita) 112
Figura 76 Cisco MCS 7800 *Series Server*, utilizado como Call Manager 113
Figura 77 Interface FXS da Cisco com 2 portas para voz 114
Figura 78 Interface FXS hardware *slot* 114
Figura 79 Identificação do tipo de Interface serial este é um WIC2T 114
Figura 80 Identificação do tipo de Interface serial este é um WIC2A/S 114
Figura 81 Router da Cisco modelo 2611 115
Figura 82 Interface do Kphone 117
Figura 83 Interface do Linphone 118
Figura 84 Interface BonePhone v0.1 119
Figura 85 Interface do GnomeMeeting 120
Figura 86 Interface GUI do SIP Communicator 121
Figura 87 Interface do SIPCommunicator nova versão 121
Figura 88 Excerto de código do SIP-Communicator para IPv6 na versão antiga 122
Figura 89 Configuração so SIP-Communicator para IPv6 122
Figura 90 Interface do VOCAL SIPSET 123
Figura 91 Telefones IPv4/IPv6 Freebit 124
Figura 92 Telefones IPv4/IPv6 Stonehenge 125
Figura 93 SER da IPTEL com o módulo SER WEB 127
Figura 94 Ver contactos *online* 127
Figura 95 Página principal de administração do CallManager 3.1 135
Figura 96 página public onde os *Users* ou utilizadores podem aceder 136
Figura 97 Adicionar um novo utilizador 137
Figura 98 O User com o ID ID deve está associado ao telefone 7960. 138

Figura 99 Painel de procura e listagem dos telefones 139
Figura 100 Página principal onde também é possível adicionar speed dials 140
Figura 101 Adição de um speed dial para o telefone IP 7910 141
Figura 102 *Speed dial* criado no telefone 7960 141
Figura 103 CallManager 5.0 (alguém que consegui instalar) 143
Figura 104 Frame GRE 147
Figura 105 Cabeçalho do GRE 147
Figura 106 Route Entry 148
Figura 107 campos do GRE 149
Figura 108 Estrutura do protocolo RTP 150
Figura 109 Registo do telefone IP no CCM 155
Figura 110 Registo do telefone IP no CCM (cont.1) 156
Figura 111 Registo do telefone IP no CCM (cont.2) 157
Figura 112 Se o 7960 liga e o 7910 não está disponível 158
Figura 113 captura no Cisco CallManager 159
Figura 114 *Ethereal* no PC10 160
Figura 115 *Ethereal* no PC11 161
Figura 116 Chamada do telefone IP 7960 → analógico (capturado no Ethereal PC) 162
Figura 117 Chamada do telefone IP 7960 → analógico (cont. 1) 163
Figura 118 Chamada do analógico para o telefone IP 164
Figura 119 Chamada do analógico para o telefone IP (cont.) 165
Figura 120 Pacote RTP 166
Figura 121 O CM é o intermediário para o controlo das chamadas.[11] 166
Figura 122 DHCP no CallManager (duas pools 10.0.0.2/8 e 192.168.0.0/24) 168
Figura 123 Configuração do *gateway* H.323 com o endereço do Rotuer2 192.168.0.1 170
Figura 124 ICMPv6 Request capturado no Túnel GRE 171
Figura 125 Pacote ICMP IPv6 reply, capturado a meio do túnel GRE 171
Figura 126 Mensagem StimulusMessage IPv4 para a inicialização da chamada 172
Figura 127 pacote Skinny IPv6 172
Figura 128 Mensagem com a informação do *Called Party* e *Calling party* em IPv4 173
Figura 129 Pacote RTP IPv4, verifica-se que é sobre UDP 173
Figura 130 Pacote RTP no túnel IPv6 174
Figura 131 Novo DWORD chamado DisableUserTOSSetting 175
Figura 132 MGEN a gerar tráfego ipv6 176
Figura 133 Pacote UDP gerado do MGEN 177
Figura 134 Marcação dos pacotes com o valor 46 (DEC) mesmo dos telefones 178
Figura 135 Pacote gerado pelo o Chariot 178
Figura 136 Alteração do script no Chariot para dar uma frame de 74 bytes para 29.6kbps .. 179
Figura 137 *Stream* RTP gerado pelo o Chariot (32kbps) 179
Figura 138 *Stream* RTP dos telefones da Cisco (27 kbps) 179
Figura 139 formato do pacote MGEN 183
Figura 140 Todas as Mensagens capturadas no servidor SER da IPTEL 197
Figura 141 Indicação para utilizar o protocolo IPv6 205
Figura 142 Configuração do endereço do servidor SIP e o SIP URL 205
Figura 143 Configuração do URL do utilizado e do Servidor SIP 205
Figura 144 Realização de uma chamada para utilizador com o URL win3@ser.sipv6.teste. 206
Figura 145 Janela de *Chat* 206
Figura 146 Pedido de registo ao servidor 207
Figura 147 Resposta do servidor 207
Figura 148 Mensagem REGISTER, para termina a sessão com o SER 208
Figura 149 Mensagem SIP para aceitar a terminação da sessão do win3 208

Figura 150 *Stream* RTP apenas num sentido na versão Linphone 1.1 209
Figura 151 ICMPv6 Port unreachable 209
Figura 152 *Streams* RTP em ambos sentidos WIN2 na versão Linphone 1.2 210
Figura 153 Pedido DNSv6 do nome ser.sipv6.teste 210
Figura 154 Resposta com o endereço IPv6 2000::1 211
Figura 155 Mensagem SIP/SDP 211
Figura 156 Pacote RTP capturado no Ethreal 212

Índice de tabelas

Tabela 1 Cenário VoIP, arquitecturas centralizadas e distribuídas 12
Tabela 2 Comparação dos protocolos VoIP 16
Tabela 3 Variações do Codec G.711[10] 22
Tabela 4 Largura de banda necessário para voz e cabeçalho IP[40] 23
Tabela 5 Largura de banda com os cabeçalhos da camada 2 incluídos [41] 23
Tabela 6 Largura de banda recomendada em kbps[10] 24
Tabela 7 Priority Values for Application Categories 57
Tabela 8 Traffic Classification Guidelines for Various Types of Network Traffic[10] 85
Tabela 9 24 *stream* criada no Chariot 91
Tabela 10 1º teste 1 fluxo de voz gerado no Chariot 91
Tabela 11 2º teste 2 fluxo de voz gerado no *Chariot* 93
Tabela 12 3º teste 3 fluxo de voz gerado no Chariot 95
Tabela 13 4 º teste 4 fluxo de voz gerado no Chariot 98
Tabela 14 Listagem dos tipos de mensagens Skinny 145
Tabela 15 Tabela com algumas mensagens Skinny relevantes 158

Lista de Siglas/Acrónimos

ACF →Admission Confirmation Message
ALG→Aplication Layer Gateways
API→ Application Programming Interface
ARQ →Admission Request Message
CAR →Committed Access Rate
CBWFQ →Class-Based Weighted Fair Queuing
CCM→Cisco Call Manager
CoPS →Common Open Policy Server
CoS →Class of Service
CQ→Custom Queuing
CRTP →Compressed Real-Time Protocol
DCE→ Data Communications Equipment
DHCP→Differentiated Services Code Point
DNS→ Domain Name System
DRAM→ Dynamic random-access memory
DSP →Digital Signal Processor
ENUM→Eletronic Number ou Telephone Number Mapping
FCCN→Fundação para a Computação Científica Nacional
FCP→Firewall Communication Protocol,
FQDN→Fully-Qualified Domain Name
FXS→ Foreign Exchange Station
GRE→Generic routing encapsulation
GSM→ Global System for Mobile Communications
GUI→Graphical User Interface.
HTTP→ Hypertext Transfer Protocol
IETF→Internet Engineering Task Force
IOS→ Internetwork Operating System
IP→Internet Protocol
IPv4→Internet Protocol version 4
IPv6→Internet Protocol version 6
ISATAP→Intra-Site Automatic Tunnel Addressing Protocol
ISDN→Integrated Services Digital Network
ISP→Internet service provider
IT→Information Technology
ITU→Internacional Telecommunication Union,
LAN→Local Área Network
LCF→Location Confirm Message
LLQ→Low-Latency Queuing
LRQ→Location Request Message
MEGACO→Media Gateway Control
MGCP→Media Gateway Control Protocol
MSP→Mini SIP Proxy.
NBAR→Network-Based Application Recognition
PBX→ Private Branch eXchange
PC→Personal Computer

PHB→Per-Hop Behavior
POTS→ Post Office Telephone Service or Post Office Telephone System
PPS →Pacotes por segundo
PQ→Priority Queuing
PSTN →Public Switched Telephone Networks
QoS →Quality of Service
RDIS→Rede Digital com Integração de Serviço
RFC→Request for Comments
RIR→Regional Internet Registry
RRQ→Registration Request Message
RSVP→Resource Reservation Protocol
RTCP→Real-time Transport Control Protocol
RTP→Real-Time Transport Protocol
SCCP→Skinny Client Control Protocol
SDP→Session Description Protocol
SEMS→SIP Express Media Server
SIP →Session Initiation Protocol
SIPRG→SIP Residential Gateway
SLA→Service Level Agreement
SMS→ Short Message Service
SMTP→Simple Mail Transfer Protocol
SS7→Signaling System 7
TCP→Transmission Control Protocol
TFTP→Trivial File Transfer Protocol
ToS→Type of Service
TRIP→Telephony Routing Over IP
UAC→User Agent Client
UAS→User Agent Server
UDP→ User Datagram Protocol
UTP→ Unshielded Twisted Pair
VIC→Voice Interface Card
VoIP→ Voice over Internet Protocol
VoIPv6→ Voice over Internet Protocol sobre IPv6
WIC-2T→ WAN interface card 2port T1

1. Introdução

Hoje em dia fala-se muito sobre VoIP, dado que permite a realização de chamadas a menor custo. Outra vantagem para além dos custos reduzidos, é a facilidade de implementa-lo num rede IP existente. Ao utilizar a rede IP é possível integrar de uma forma distribuida os serviços de *Voice Mail*, video-conferência, *instant messaging*, serviços *Web* no telefone IP, e muitas outras funcionalidades para além da voz, que a telefonia tradicional não tem. Toda a informação passa a estar disponível num único meio de accesso que é a rede IP. Usando equipamentos intermédios é possível integrar diferentes tecnologias, por exemplo a realização de uma chamada analógica POTS ou RDIS para um telefone IP.

Com o aparecimento do protocolo IPv6, houve a necessidade de desenvolver equipamentos (servidores e routers) ou soluções que conseguessem converter do protocolo IPv4 para o protocolo IPv6 e vice versa. Este processo chama-se mecanismo de transição. Para o mecanismo de transição implementada neste projecto foi testado a qualidade de serviço ou Quality of Service QoS para a solução implementada.

História

O conceito de voz sobre *Internet Protocol* (IP) ou *Voice over Internet Protocol* (VoIP) começou em 1995 e foi desenvolvido por uns indivíduos que reconheceram as vantagens de enviar voz em pacotes IP, em vez de utilizar o serviço telefónico comutado. A principal vantagem é um utilizador poder comunicar à distância sem custos a partir dum computador. Os telefones dividem-se em dois tipos *hardphones* e *softphones*. Os *hardphones* funcionam de forma independente e os softphones necessitam de ser instalados num computador. O primeiro *softphone* apareceu em 1995 para a Internet. Inicialmente as aplicações VoIP não tinham muito boa qualidade e conectividade, no entanto verificou-se uma grande utilidade nesta tecnologia.

A partir de 1995 até 1998 o VoIP começou a evoluir significativamente e haviam pequenas empresas que ofereciam um serviço de telefonia a partir dum computador. Seguiu-se depois o serviço telefone a telefone, mas normalmente era necessário utilizar como intermediário um computador para estabelecer a ligação.

Tal como outras aplicações da Internet, nos finais dos anos 1990s, os serviços VoIP dependiam de patrocinadores e publicidade, em vez de serem cobrados aos clientes ou utilizadores.

Com o aumento gradual da largura de banda as chamadas começaram a ter uma melhor qualidade e com menos atraso. No entanto as dificuldades na ligação entre a Internet e o *Public Switched Telephone Networks* (PSTN) continuaram a persistir.

Contudo, as empresas de VoIP, conseguiam oferecer chamadas grátis para os clientes em várias localidades.

Apenas quando os fabricantes como a Cisco Systems e Nortel começaram a produzir equipamentos VoIP dedicados, foi quando o VoIP começou a expandir-se rapidamente.

Significava que as funcionalidades que costumavam ser tratados no computador agora podiam ser tratados por esses equipamentos evitando o sobrecarregamento do computador.

Assim que o *hardware* se tornou a um preço acessível, grandes empresas podiam implementar VoIP nas suas redes LAN, e os *service providers* até começava a encaminhar as chamadas pelas suas redes até à Internet. Desde do ano 2000, o VoIP tem expandido exponencialmente. Enquanto que as empresas transit para VoIP para poupar nas despesas em chamadas de longa distância e de infra-estrutura, o serviço VoIP tem expandido para os utilizadores residenciais. [1]

Na passada década, a industria das telecomunicações tem testemunhado um avanço na forma como as pessoas se organizam e comunicam. Muitos destas mudanças ajudaram na expansão da Internet e de aplicações baseadas no protocolo *Internet Protocol* IP.

A tecnologia de redes baseada em pacotes tem passado a de voz tradicional ou redes comutados como PSTN (DataQuest, 1998) para segundo plano. Esta tecnologia que transporta voz sobre pacotes IP, é chamada voz sobre IP ou *voice over* IP (VoIP).

Os benefícios mais importantes são:

1. **Baixo Custo:** Ao transportar voz em redes IP, as empresas podem reduzir ou eliminar os custos nas chamadas associadas ao transporte de chamadas pela a rede comutada ou *Public Switched Telephone Network* (PSTN).

2. **Interoperabilidade e normas não proprietárias:** Ao optar por normas standard, ou seja, normas não proprietárias é possível comprar equipamentos de vários fabricantes e não ter o problema de depender de soluções proprietárias.
3. **Redes de voz e dados:** As empresas podem ter uma rede integrada de voz e de dados, oferecendo uma rede com qualidade e fiabilidade.

1.1. Objectivo do Trabalho

Um dos objectivos do projecto foi implementar uma solução de VoIP em IPv6 nativo. Assim foi implementado o cenário da figura seguinte. Verificou-se que estas soluções apenas existem em soluções *Open Source*.

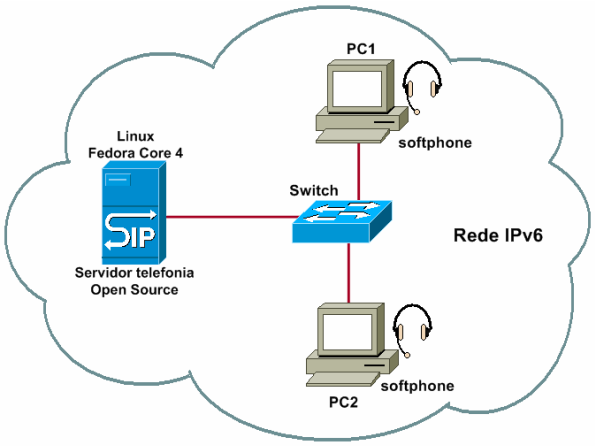


Figura 1 Cenário VoIP em IPv6 nativo

Outro objectivo deste projecto foi a integração de uma solução de voz em IPv4 sobre um *backbone* IPv6. Na figura seguinte é possível verificar três redes, duas ilhas IPv4 e um *backbone* IPv6. O objectivo deste cenário é conseguir estabelecer uma chamada dum telefone para o outro, e do telefone analógico para ambos os telefones IP.

Assim, foi necessário recorrer a um mecanismo de transição IPv4 para IPv6, para poder transportar o tráfego de voz em IPv4 sobre IPv6, com o respectivo análise. Também será implementada e testada uma solução de Qualidade de Serviço (QoS) para o mesmo cenário.

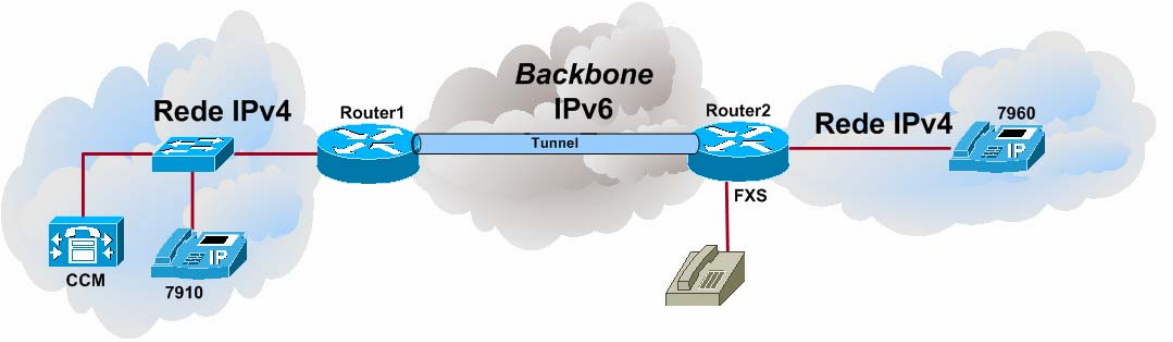


Figura 2 Cenário a implementar

1.2. Planeamento

Este projecto divide-se nas seguintes fases:

1. **Inicial**
    - Fase dedicada à investigação e aprendizagem de conceitos
2. **Pré-requisitos**
    - Fase para a definição de cenários, e dos requisitos para a implementação. Assim esta fase necessita de muita investigação para encontrar uma solução para os cenários definidos (Figura 1 e Figura 2).
3. **Implementação**
    - Uma fase essencialmente prática, e de testes.
4. **Definição de uma solução de qualidade de serviço ou QoS**
    - Deve ser investigado uma solução de QoS, devido ao facto do tráfego de voz ser vulnerável quando a rede está congestionada. Nesta será aplicado um mecanismo de QoS, permitindo que a comunicação por voz tenha qualidade no serviço oferecido.
5. **Fase final**
    - Conclusão do projecto. Constituído pela entrega do relatório final, artigo, poster e a respectiva realização da apresentação.

O gráfico de Gantt encontra-se em anexo (ver Figura 74 Calendarização do projecto).

1.3. Estrutura do relatório

De seguida, são apresentados os conteúdos de cada capítulo.

Cap. 1: Introdução – Ao longo deste capítulo é feita uma introdução sobre história do VoIP e sua utilização na Internet. Além disso, são também apresentados os objectivos gerais do projecto, e o gráfico de *Gantt*.

Cap. 2: Introdução sobre VoIP – Neste capítulo é feita uma introdução sobre a arquitectura VoIP, os seus componentes, e protocolos. No fim do capítulo é realizado a implementação dum cenário VoIP para analisar os protocolos envolvidos.

Cap. 3: Métodos de translação – Neste capítulo são mencionados os mecanismos de translação IPv4 para IPv6, para a implementação do cenário definido. Depois é realizado a explicação da implementação de um túnel GRE.

Cap. 4: Mecanismos de Qualidade de Serviço ou Quality of Service QoS – Estudo aos mecanismos de QoS que existem para IPv6, conceitos a ter em conta para o teste de QoS, para o cenário definido. No fim do capítulo é realizado uma explicação da implementação duma solução QoS ao cenário definido, com os devidos resultados.

Cap. 5: VoIPv6 nativo – Realização de um cenário utilizando o protocolo SIP para IPv6, ou seja, uma solução para VoIP em IPv6 nativo.

Cap. 6:Implementação e Testes – Realização da implementação prática relacionados com cada capítulo descrito.

2. Telefonia IP

Este capítulo irá descrever os componentes básicos relacionados com a telefonia IP, e os seus protocolos. Existem dois tipos de protocolos na telefonia IP, os protocolos de sinalização e os protocolos de transporte. No fim deste capítulo também serão explicados os cálculos para a determinação da quantidade de largura de banda que estes protocolos ocupam. Estes conhecimentos foram importantes para a realização de testes de QoS, referente ao capítulo 6 ou o capítulo dos testes.

2.1. Componentes básicos

A telefonia IP tem uma infra-estrutura que é constituída normalmente por diferentes componentes. Nesta secção serão descritos alguns deles.

Terminal

O terminal é o ponto de inicialização e terminação de chamadas e de *streams* associados. Normalmente são telefones que podem ser feitos de *hardware* (funciona autonomamente) e *software* (funciona no computador). Estes telefones, para além de suportarem voz, também poderão transmitir dados ou vídeo. Os telefones IP necessitam de pelo menos um endereço IP. Vários terminais podem ter o mesmo endereço IP, mas são tratados separadamente.

Servidor

Para estabelecer um chamada em Voz sobre IP (VoIP), é necessário os terminais terem um endereço IP e um porto. Os terminais registam o seu endereço IP no servidor. O servidor armazena os endereços IP destes telefones e os números de telefone associados. Quando um telefone liga para um número, o servidor irá tentar resolver esse número a um endereço IP. Para conseguir obter esta informação, o servidor poderá comunicar com outros servidores e serviços. O servidor de telefonia IP é responsável pela autenticação dos registos, autorização de chamadas e a gestão de contas.

Gateway

Os *Gateways* permitem a comunicação entre terminais que não são inter operáveis. Assim o *Gateway* irá converter uma sinalização protocolar em outro (por exemplo a sinalização SIP para ISDN e vice versa), também poderá traduzir endereços de redes (IPv4/IPv6) ou *codecs.* O VoIP utiliza vários tipos de normas e protocolos. De modo a entender as diferenças entre os protocolos VoIP, o designer deve entender a terminologia. Um equipamento suporta VoIP consegue transportar tráfego de voz sobre a rede IP.

Em VoIP, o *Digital Signal Processor* (DSP) é um microprocessador que consegue digitalizar o sinal de voz e guarda em pacotes de voz. Estes pacotes de voz são transportados utilizando pacotes IP. Uma vez que as aplicações multimédia são muito sensíveis ao atraso. Assim é necessário que a rede ofereça uma boa qualidade de serviço. O *traffic shaping* deve ser utilizado de modo a ter em conta a garantia e fiabilidade das ligações VoIP.

Principais componentes numa rede VoIP são as seguintes:

- *Media gateways*
- *Media gateways controllers*
- A rede IP

Media Gateway

São responsáveis para determinar a origem da chamada, detecção de chamada, conversão analógica para digital, a criação de pacotes de voz, ou compressão e descompressão (codec). Outras funcionalidades, são a compressão de voz, *echo cancellation, silence suppression*, e junção de dados estatísticos. O *media gateway* serve de interface de voz para a rede IP. Numa chamada apenas é utilizado um endereço IP para o transporte usando o protocolo *Real-Time Transport Protocol* RTP (sobre UDP *User Datagram Protocol). Media gateways* podem ser equipamentos dedicados ou um PC que ocorre o software VoIP. Existem outras funcionalidades como *trunking* para o PSTN, oferecer uma interface analógica ou digital (*Private Branch Exchange* PBX), integração de um soft PBX. O *media gateway* pode também ser utilizado como um telefone IP. O *media gateway* também é chamado de *Voice Agent.*

Media Gateway Controller

Os controladores *media gateway* oferece sinalização e controlo de serviços que coordenam as funções de *media gateway.*

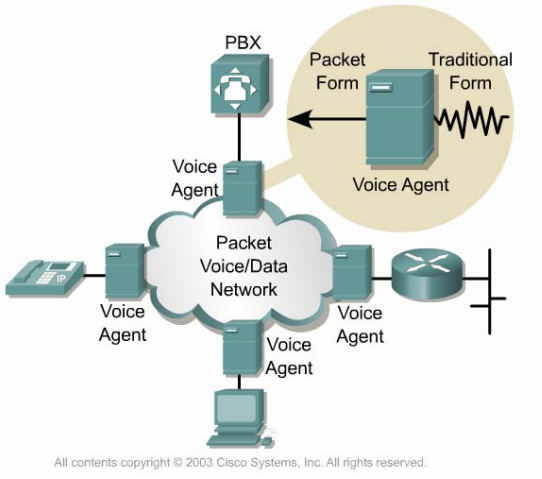


Figura 3 Media Gateway Controller

O *media gatweway controller* é responsável pela a sinalização, tradução de números de telefone, *host lookup*, gestão de recursos, e servir de interface com os protocolos PSTN e *Signaling System* 7 (SS7). Numa rede VoIP escalável, dois equipamentos podem executar as funções de controlador. O *signaling controller* também é referido como o *call agent* na arquitectura centralizada. Também é conhecido como *gatekeeper* numa rede H.323 e o *proxy* ou *redirect server* na rede SIP.

A rede IP

É possível de ver a rede VoIP como um comutador lógico. Contudo é um sistema distribuído, em vez de ser um único comutador. O *backbone* IP oferece conectividade entre os elementos distribuídos.

Endereçamento

Um utilizador para utilizar um serviço de comunicação tem de se identificar. A pessoa que atende a chamada telefónica é chamada *called party*. A pessoa que inicia uma chamada telefónica é chamada *calling party*. Idealmente, este identificador deve ser independente do

local físico. Depois, a rede deve ser responsável por encontrar a localização corrente do *called party*. Um utilizador pode especificar quais os identificadores que ele deseja ser contactado. Um sistema telefónico utiliza os números definidos no E.164 (*the international public telecommunication numbering plan*). [2] ou na Anacom no caso de Portugal [3].

Um identificador é composto por 15 dígitos com o sinal +, por exemplo +123456789123.

Quando se liga com o sinal + é normalmente substituído (normalmente por 00) pelo o código internacional, Seguido pelo o código do país e o número do cliente (*subscriber*)

Os primeiros sistemas de telefonia IP, utilizavam os endereços IP como identificadores dos terminais. Por vezes, ainda é utilizado. Contudo, os endereços IP não são independentes da localização (mesmo em IPv6) e são difíceis de lembrar (especialmente no IPv6), assim não são bons identificadores.

Actualmente a telefonia IP utiliza dois tipos de identificadores:

- URIs **[RFC2396]**
- Números (E.164).

O *Universal Resource Identifier* (URI) identifica um recurso pelo um nome independentemente da localização. Normalmente o nome tem a forma ou sintaxe de um e-mail. Estes identificadores têm várias vantagens. São fáceis de lembrar, e quase todo o utilizador da Internet tem um endereço de e-mail, e um outro serviço que pode ser atribuído ao mesmo identificador. Um utilizador pode estar num domínio ou *Domain Name System* (DNS). Uma desvantagem dos URIs, é quando se pretende ligar para um utilizador com telefone através do número dele. Os números de telefone não são adequados para a Internet. Assim o sistema ENUM foi criado, com o objectivo de adaptar os números de telefone a nomes de domínio.

2.2. Arquitectura protocolar

Existem dois tipos de protocolos em telefonia IP, os protocolos de sinalização e os protocolos de transporte. Em 1995, os primeiros produtos *VoIP* chegaram ao mercado, com os objectivos de reduzir custos. O VoIP tem vários protocolos associados para controlar as chamadas, como:

1. H.323
2. *Media Gateway Control Protocol* (MGCP)
3. *Session Initiation Protocol* (SIP)
4. *H.248/Media Gateway Control* (MEGACO)

Existem outros protocolos que funcionam com estes protocolos de sinalização e controlo, como o *Real-Time Transport Protocol (RTP), Real-time Transport Control Protocol (RTCP), e Resource Reservation Protocol (RSVP).*

2.2.1. Sinalização

A seguir é mostrado em que camada pertence cada protocolo de telefonia, como o RTP que pertence à camada de transporte do modelo OSI.

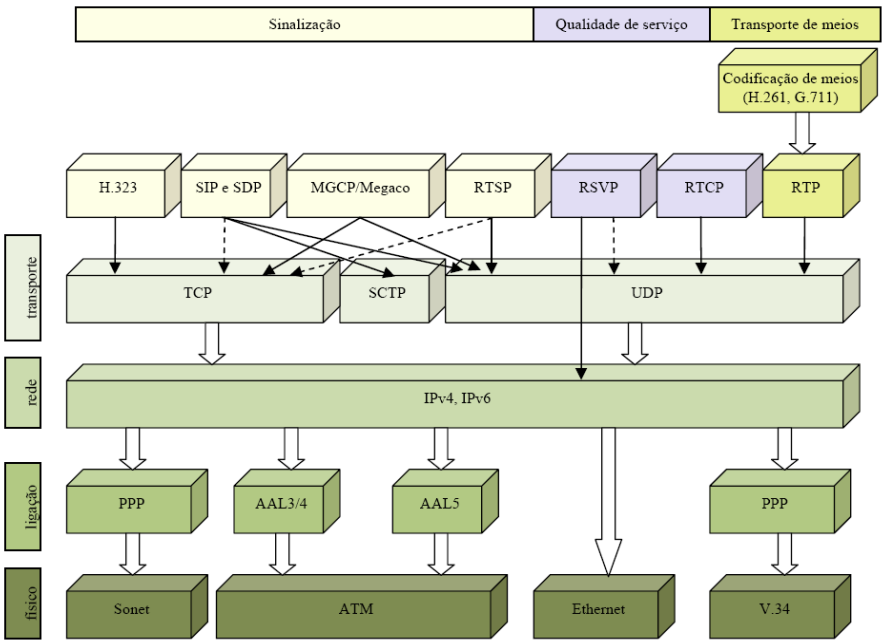


Figura 4 Arquitectura de telefonia [60]

No passado, todas as redes de voz eram construídas por uma arquitectura centralizada com terminais não inteligentes (telefones) e que eram controlados por comutadores centralizados.

Com a tecnologia VoIP, é possível construir redes com uma arquitectura centralizada ou distribuída. Usando esta arquitectura, a rede torna-se flexível, fácil de gerir, e ter inovação nos terminais, dependendo do protocolo utilizado. Em geral, as arquitecturas centralizadas estão associadas aos protocolos MGCP e H.248/MEGACO. Em arquitecturas centralizadas, a parte inteligente da rede é centralizado e os terminais estão limitados às funções nativas. Apesar da maior parte das arquitecturas utilizarem os protocolos MGCP ou H.248/MEGACO, é também possível de construir redes com SIP ou H.323 de forma centralizada utilizando *back-to-back user agents*[4] (B2BUAs) ou *gatekeeper* que fazem o encaminhamento da sinalização das chamadas (GKRCS), respectivamente.

A arquitectura centralizada é a preferida porque permite uma gestão centralizada no controlo das chamadas. Também simplifica as chamadas para equipamentos antecessores ou *legacy.*

As arquitecturas distribuídas são associados aos protocolos H.323 e SIP. Estes protocolos distribuem a inteligência da rede entre terminais e equipamentos de controlo de chamadas. Inteligência é a instância que refere ao estado das chamadas, funcionalidade na chamada, encaminhamento da chamada, facturação, e no controlo das chamadas. Os terminais podem ser *VoIP gateways,* telefones IP, servidores multimédia, ou qualquer outro equipamento que consegue iniciar ou terminar uma chamada VoIP. O equipamento que serve para o controlo das chamadas são chamados de *gatekeeper* numa rede H.323, e o *proxy* ou o *redirect server* na rede SIP.

A arquitectura distribuída é mais flexível porque permite que as aplicações VoIP sejam tratados como qualquer outra aplicação distribuída, esta flexibilidade provem do facto dos terminais serem inteligentes.

[4] A VOCAL tem uma solução B2BUA, http://www.vovida.org/

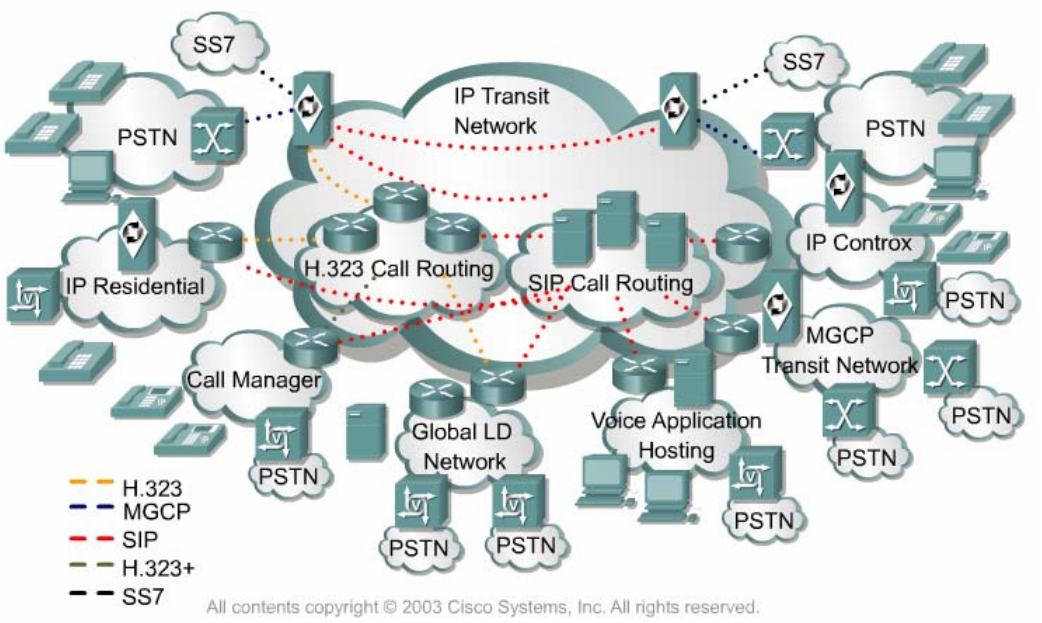


Tabela 1 Cenário VoIP, arquitecturas centralizadas e distribuídas

H.323

Originalmente o H.323 (ITU-T) foi desenvolvido para ser um protocolo de transporte para aplicações multimédia na *Local Area Network* (LAN). Apesar do H.323 ainda ser utilizado por vários fabricantes para aplicações de videoconferência, está também envolvido em redes VoIP.

O protocolo H.323 é o mais utilizado para controlar chamadas e sinalização VoIP. O H.323 define o registo de chamadas, admissão de chamadas, e o estado (RAS). O protocolo H.225 serve para a iniciar chamadas, e o protocolo H.245 serve para a troca de mensagens. O H.323 é baseado no protocolo Q.931 *Integrated Services Digital Network* (ISDN), o que permite ser inter operável com redes de voz antecessores (*legacy*) como PSTN ou *Signaling System* 7 (SS7). Como o protocolo é utilizado na arquitectura distribuída, o H.323 permite às empresas construírem redes escaláveis, redundantes, e flexíveis. Ele oferece mecanismos para a interligação a outras redes VoIP e suporta redes inteligentes nos terminais ou nos *gatekeepers.*

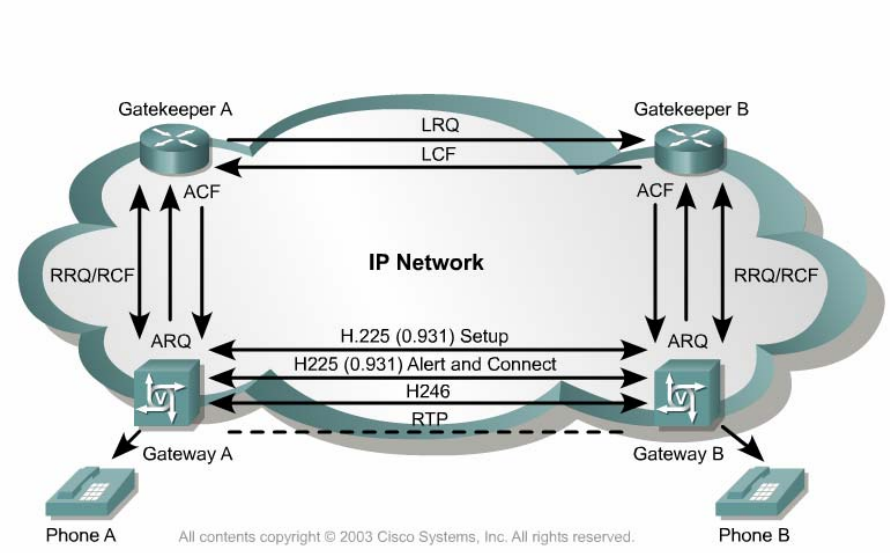


Figura 5 H.323 sinalização

Na figura mostra um exemplo de uma rede H.323. O H.323 utiliza os seguintes tipos de mensagens para facilitar a comunicação, como mostrado na figura anterior:

- *Location Request Message* (LRQ)
- *Location Confirm Message* (LCF)
- *Admission Confirmation Message* (ACF)
- *Admission Request Message* (ARQ)
- *Registration Request Message* (RRQ)

H.245

O H.245 é um protocolo de sinalização na arquitectura de comunicação multimédia H.323,

É utilizado para trocar mensagens H.245 ponto a ponto entre terminais H.323. As mensagens H.245 são transportadas sobre canais de controlo H.245. O canal de controlo H.245 é um canal lógico 0 e está permanentemente aberto, ao contrário que os outros canais. As mensagens trocam a informação sobre a capacidade dos terminais e para abrir e fechar canais lógicos. Depois da ligação ser estabelecida através do *call signalling procedure*, o protocolo *call control* H.245, é utilizado para resolver o *call media type* e estabelecer o fluxo de multimédia, antes da chamada ser estabelecida. Ele também gere as chamadas depois de estarem estabelecidas.

A Recomendação H.245 [H.245, 1998] define o protocolo H.245, o qual permite a troca de informação sobre capacidades, negociação de canais e comutação entre diferentes tipos de meios. Especifica a sintaxe e a semântica das mensagens, assim como procedimentos para o uso delas na negociação de canais no início e durante a comunicação.

Uma vez estabelecida a chamada, a transmissão dos meios é iniciada. O transporte de áudio e vídeo é realizado recorrendo ao protocolo RTP, não esquecendo o RTCP para o controlo e monitorização da entrega de dados do RTP. No entanto também aqui a Recomendação H.225.0 define procedimentos para a formatação e transporte dos pacotes de áudio e vídeo.

Por fim a codificação e descodificação de áudio e vídeo, segundo a Recomendação H.323, são feitas utilizando codificadores definidos nas Recomendações G.711, G.722, G.728, G.729 e G.723.1 para áudio e H.261 e H.263 para vídeo.[60]

MGCP/MEGACO

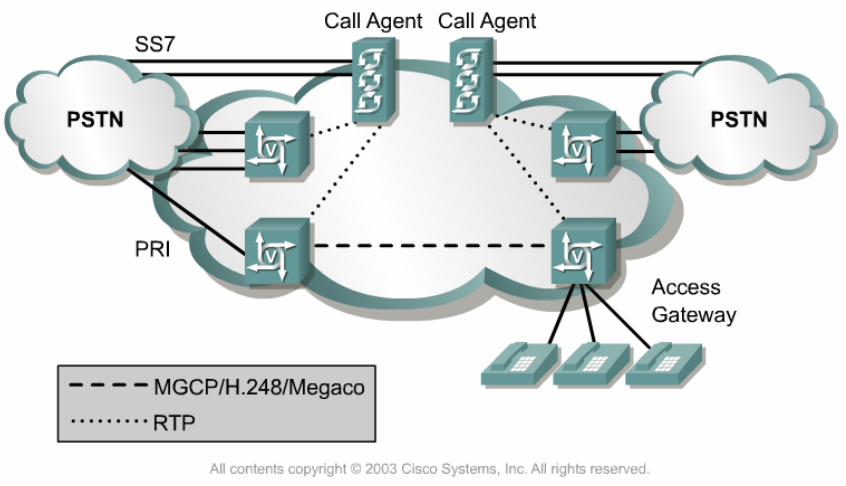


Figura 6 Protocolo MGCP

Media Gateway Control Protocol (MGCP), também conhecido como IETF [RFC 3435], define uma arquitectura centralizada para a criação de uma rede de aplicações multimédia, incluindo VoIP. O H.248 é o resultado da colaboração entre o ITU-T e o IETF. O H.248 é também referido como o IETF [RFC 3525] e como *Multimedia Gateway Control Protocol* (Megaco). O H.248 também define uma arquitectura centralizada para a criação de aplicações multimédia. Em muitos casos o H.248 estende o MGCP. O MGCP e H.248/Megaco foram criados para oferecer uma arquitectura em que o controlo de chamadas e serviços pudessem ser adicionados de forma centralizada à rede VoIP.

Esta arquitectura que utiliza estes protocolos, é muito semelhante à arquitectura PSTN.

Os protocolos MGCP e H.248/Megaco definem muito aspectos da sinalização. O SDP também é utilizado para a troca das mensagens. Numa arquitectura centralizada, MGCP e

H.248/Megaco permitem empresas de construir redes grandes de forma escalonável, com redundância.

SIP

O protocolo SIP projectado para ser um protocolo de multimédia, que podia terá partido das mensagens e arquitecturas encontradas nas aplicações da Internet. Ao utilizar uma arquitectura distribuída com o *universal resource locators* (URLs) para o esquema de nomes ou endereçamento e mensagens de texto. O SIP tira partido do modelo da Internet para as redes e aplicações VoIP. Para além do VoIP, o SIP é utilizado para videoconferência e *instant messaging.* O SIP apenas define como as sessões são iniciadas e terminadas. Ele utiliza outros protocolos do IETF para definir outras sessões de multimédia e de VoIP, como compatibilidade como SDP, endereçamento através de URLs, *Domain Name Systems* (DNSs) para a localização do serviço, e *Telephony Routing Over* IP (TRIP) para o encaminhamento das chamadas.

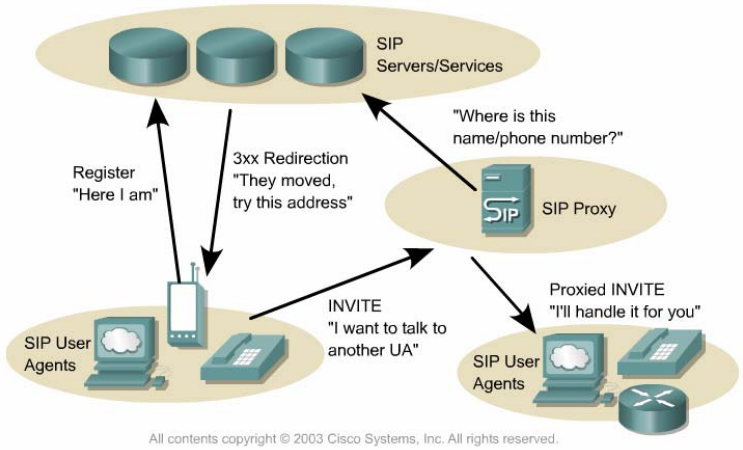


Figura 7 Protocolo SIP arquitectura

Apesar do IETF ter conseguido definir as extensões que permitem que o SIP seja compatível com redes de voz *legacy* ou antecessoras ao VoIP, o principal objectivo deste protocolo é criar um ambiente que suporta os modelos de comunicação da próxima geração que irá utilizar as aplicações da Internet.

Nota: A arquitectura do SIP é tratada com mais detalhe no Capítulo 5 deste relatório.

Interligação de protocolos de VoIP

	H.323	SIP	MGCP/H.248/MEGACO
Norma	ITU	IETF	MGCP/MEGACO-IETF,H.248-ITU
Arquitectura	Distribuída	Distribuída	Centralizada
Versão actual	H.323v4	RFC2543	MGCP1.0, Megaco, H.248
Controlo de chamada	*Gatekeeper*	*Proxy/redirect server*	*Call agente/media gateway controller*
Terminais	*Gateway, terminal*	*User agente*	*Media gateway*
Sinalização (transporte)	Transmission control(TCP) or User datagram Protocol (UDP)	TCP ou UDP	MGCP-UDP, Megaco/H.248-ambos
Suporte multimédia	Sim	Sim	Sim
Serviços suplementares	Oferecido pelos terminais ou controlador de chamadas	Oferecido pelos terminais ou controlador de chamadas	Oferecido pelo controlador de chamadas

Tabela 2 Comparação dos protocolos VoIP

Signaling System 7 (SS7)

Outro protocolo é o *Signaling System* 7 (SS7). O SS7 é um sistema *Common Channel Signaling* (CCS) que utiliza redes de circuitos comutados, como o *Integrated Services Digital Network* (ISDN) e o PSTN. A empresa Bellcore desenvolveu o SS7. Ele separa a informação de sinalização da dos dados do utilizador. Um canal específico, chamado de canal D, é utilizado exclusivamente para transportar a informação de sinalização para todos os outros canais do sistema. Este tipo de sinalização é chamado called *out-of-band* porque ele não necessita da largura de banda do utilizador. Para VoIP conseguir utilizar *route calls* para o PSTN, e deve fazer a interface com o SS7, que o PSTN utiliza.

O protocolo Skinny da Cisco

O Skinny Client Control Protocol (SCCP) é um protocolo proprietário da Cisco. As mensagens deste protocolo são trocadas entre o Cisco *Call Manager* e os telefones IP da Cisco. Também é suportado por outros fabricantes. É possível ter um proxy H.323 para comunicar com um cliente *Skinny* utilizando o protocolo SCCP. Por exemplo, no caso do telefone IP ser um cliente *Skinny* a ligar para um cliente H.323. O proxy é utilizado para a sinalização H.225 e H.245. O cliente *Skinny* (i.e. um telefone *Ethernet*) utiliza TCP/IP para transmitir e receber chamadas de/para o cliente *Skinny* ou terminal H.323. A maior parte do

poder de processamento H.323 reside no H.323 *proxy* também conhecido como o *Call Manager*. O terminal (telefone) que é chamado de Skinny Cliente, consome menos processamento. As mensagens *Skinny* são transportadas sobre TCP e utilizam o porto 2000, enquanto que as mensagens RTP são transportadas sobre UDP.

2.2.2. Codificadores

Os codificadores e descodificadores, são denominados pelos codecs, e eles são dispositivos que permitem reduzir a largura de banda para a transmissão de dados utilizando técnicas de compressão. Estas técnicas de compressão devem para isso operar em tempo real, devido às características do próprio serviço, como a comunicação interactiva. A compressão de sinais é baseada em técnicas de processamento que eliminam informação redundante, ou mesmo desnecessária. Na compressão pode haver ou não perda de informação dependendo do método utilizado. Existem várias entidades responsáveis por normalizar codificadores de áudio e vídeo, tais como a *International Telecommunication Union* (ITU), *Telecommunication Industries Association* (TIA) e *United States Federal Standards* (USFS). Para codificar o sinal áudio em tempo real alguns dos codificadores mais conhecidos são ITU-T G.711, ITU-T G.722, ITU-T G.726, ITU-T G.723, ITU-T G.728, ITU-T G.729, CELP, GSM e MPEG-Audio e para a codificação de vídeo em tempo real os codificadores H.261, H.262, H.263 e JPEG.[60]

2.2.3. Transporte

O transporte de meios difere substancialmente do transporte de dados como a transferência de ficheiros, em que propriedades como a congestão e perda de pacotes, *jitter* e atraso de pacotes são fundamentais para a comunicação entre utilizadores. Protocolos como o HTTP e o *File Transfer Protocol* (FTP) são utilizados sobre o protocolo TCP que tem a particularidade de ser um protocolo com confirmação de entrega de dados, permitindo, quando um pacote é perdido ou corrompido, a sua retransmissão. No entanto o atraso introduzido pela funcionalidade de reenvio não é favorável à transmissão do *stream* e é facilmente dispensada a confirmação, sendo necessário o recurso a outros protocolos. O

protocolo UDP é uma das soluções mais utilizadas pois não utiliza mecanismos de retransmissão mas não garante a entrega de pacotes nem que estes vão chegar ao destino pela ordem de saída. Para a entrega de meios em tempo real é usado o *Real-Time Transport Protocol (*RTP) definido no RFC 1889. Este protocolo é normalmente utilizado sobre o UDP e permite serviços de entrega ponto a ponto para a transmissão de dados em tempo real.[60]

O SIP funciona em conjunto com o *Resource Reservation Protocol (*RSVP*), Real-time Transport Protocol (*RTP*), Real-time Control Protocol (*RTCP*), Real-time Streaming Protocol* (RTSP*), Session Announcement Protocol (*SAP*),* e *Session Description Protocol (SDP).* O H.323 funciona em conjunto com o RTP/RTCP. Os *gateways* modernos de voz normalmente têm duas partes. A primeira é o *gateway* de sinalização e a segunda é o *media gateway*. O *gateway* de sinalização comunica com o *media gateway* utilizando o protocolo MGCP. O MGCP pode interoperar com o SIP e H.323.

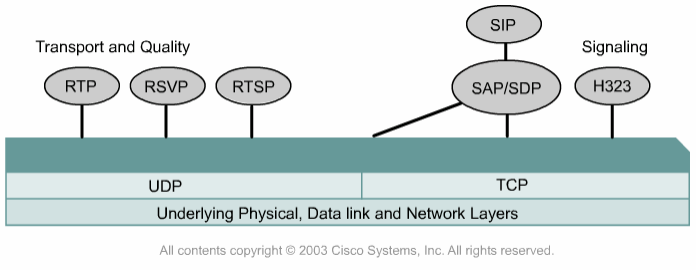


Figura 8 Protocolos de transporte e de sinalização

***Real Time Transport Protocol* (RTP)** é utilizado para transportar os dados em tempo real. É utilizando tanto no protocolo SIP como no protocolo H.323. O RTP suporta a transferência de áudio e vídeo em tempo real, em redes de pacotes comutados. O cabeçalho RTP contém a informação que ajuda o receptor de reconstruir os dados multimédia. Ele contém a informação que especifica como o codec deve partir a *stream* em pacotes. O RTP não reserva recursos na rede, no entanto oferece a informação que permite que o receptor recuperar a presença de perdas ou de *jitter.*

No Anexo 8.9 está a descrição dos campos do cabeçalho do protocolo RTP.

2.2.4. Formatos *Payload*

Já foi referido que o protocolo RTP serve para o transporte de áudio, vídeo e dados em tempo real. De modo a ser flexível, este protocolo permite a utilização de codificadores particulares, utilizando a definição de formatos de dados específicos para esses codificadores. Estes formatos descrevem a sintaxe e a semântica dos dados RTP. A associação entre codificadores e formatos, é através do registo de nomes/Payload Type (PT) no *Internet Assigned Numbers Authority* (IANA).

No [RFC3551,2003], é definido um conjunto de codificadores normalizados e os seus nomes quando são usados dentro do RTP. As especificações de codificação são definidas

independentemente do mecanismo de transporte usado.

Alguns formatos de *payload* foram já definidos para RTP, como o caso dos codificadores de vídeo H.261 [RFC 2032, 1996], H.263 [RFC 2190, 1997], MPEG [RFC 2250, 1998] e

codificadores de áudio G.722.1 [RFC 3047, 2001]. Existem também outros formatos de *payload* criados para permitir serviços genéricos, como por exemplo o [RFC 2198,

1997] que define um formato *payload* RTP para a transmissão de dados codificados de uma maneira redundante que permite recuperar informação perdida.[60]

2.2.5. Largura de banda utilizada pelo protocolo RTP e de sinalização

Dado que na implementação de QoS, foi necessário ter em conta a largura de banda que a *stream* RTP ocupa. A seguir são explicados os conceitos e depois os cálculos.

A largura de banda necessária para transportar Voz sobre IP depende de vários factores:

1. O cabeçalho IP
2. O Codec (coder/decoder) e o período de amostragem
3. O meio de transmissão
4. *Silence suppression*

O codec determina a largura de banda que os dados de voz irão ocupar. O cabeçalho IP/UDP/RTP tem geralmente um *overhead* de 40 bytes por pacote nas ligações ponto a ponto. Utilizando o RTP *compression* é possível reduzir para 2 a 4 bytes [RFC 2508]. A transmissão no meio como *ethernet,* irá adicionar os cabeçalhos, *checksum*, e *spacers* do pacote. Alguns *codecs* implementam *silence suppression*, que reduzem a largura de banda necessária até 50 porcento.

Cabeçalho IP

O termo cabeçalho IP é referente à informação é colocada no pacote, como os cabeçalhos IP, UDP, e RTP. O *payload* gerado pelo o codec é encapsulado em sucessivas camadas de informação de modo a chegar ao destino.

Estas três camadas são:

- IP – *Internet Protocol*
- UDP – *User Datagram Protocol*
- RTP – *Real-time Transport Protocol* - Existem vários tipos de *codecs* os telefones Cisco utilizam o codec G729A, que tem um *sampling rate* de 20msec, um *voice payload* n bytes de 20bytes, o número de pacotes por segundo é 50 e o consumo de largura de banda é de 24kbps. Incluindo os *header* da camada 2 do modelo OSI temos o seguinte:

Latência e *Overhead* do pacote

A escolha do número de frames por pacote afecta tanto a latência como o *overhead* do pacote. Um período de amostragem longo produz uma maior latência afectando a qualidade da chamada. Um período de amostragem curto torna o cabeçalho IP significativamente maior que o *payload*, assim mais de metade da largura de banda é utilizado para os cabeçalhos, o que não é desejável. Assim o período de amostragem deve ser adequado nem muito longo nem muito curto.

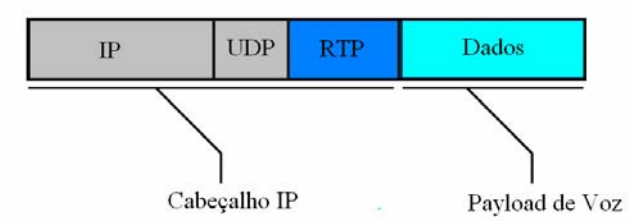


Figura 9 O cabeçalho IP forma uma parte significativa em pacotes (com *payload* pequeno)

O Codec

A conversão do sinal analógico para digital é realizada pelo o codec. O codec tira amostragens ao sinal ou onda com intervalos regulares e gera um valor por amostra. As amostras são tiradas tipicamente 8000 vezes num segundo. Estes valores são acumulados num período de tempo fixo de modo a criar uma *frame* ou quadro de dados. Normalmente o período de amostragem é de 20 ms. Alguns codecs como o G.723.1 utilizam períodos de 30 ms ou maiores. Outros têm períodos mais curtos, como 10 ms implementando pelo o G.729a. Características importantes dos *codecs* são, o número de bits produzidos por segundo. O período de amostragem, define a frequência com que são transmitidas as amostras. Juntos dão o tamanho da *frame*. Por exemplo o codec G.711 tem uma taxa de amostragem de 20 ms. Assim ele gera 50 *frames* de dados por segundo. O codec G.711 transmite 64000 bits por segundo (8kbps) assim cada *frame* irá conter no *payload* 64000/50=1.280 bits ou 160 bytes.

Cálculo da largura de banda[42]

Formulas:

O tamanho total do pacote = (cabeçalho da camada 2) + (cabeçalho IP/UDP/RTP) + (tamanho dos dados ou *payload size*)

PPS = (bit rate do codec)/(tamanho do *payload*)

Largura de banda = tamanho total do pacote*PPS

PPS (Pacotes por segundo) representa o número de pacotes que são necessários para transmitir a cada segundo de modo a transportar o bit rate do codec.

Exemplo para a Norma G.711:

Por exemplo, qual será a largura de banda necessária para o codec G.729 numa *stream* RTP, para o caso dos telefones IP da Cisco utilizados (ver Figura 41).

O tamanho total do pacote ou *frame* (bytes) = 14 bytes (Cabeçalho Ethernet) + (cabeçalhos: IP de 20 bytes + UDP de 8 + RTP de 12) + 160 bytes (nº de bytes dados de voz) = 214 bytes

O tamanho total do pacote (bits) = (214 bytes) * 8 bits per byte = 1712 bits

PPS = (8Kbps codec bit rate) / (160) = 50 pps

Largura de banda por *stream* = tamanho da *frame* (1712 bits)*50 pps = 85,6 Kbps

A seguir na Tabela 3 são mostrados variantes da norma G.711.

Norma de codificação	Taxa de amostragem (ms)	Payload em bytes de Voz	Pacotes por segundo	Largura de banda por stream
G.711	20	160	50.0	80.0 kbps
G.711 (SRTP)	20	164	50.0	81.6 kbps
G.711	30	240	33.3	74.7 kbps
G.711(SRTP)	30	244	33.3	75.8 kbps

Tabela 3 Variações do Codec G.711[10]

Exemplo para a Norma G.729:

Por exemplo, qual será a largura de banda necessária para o codec G.729 numa stream RTP, para o caso dos telefones IP da Cisco utilizados (ver Figura 41).

O tamanho total do pacote (bytes) = (Cabeçalho Ethernet de14 bytes) + (cabeçalhos: IP de 20 bytes + UDP de 8 + RTP de 32) + (nº de bytes dados de voz 20 bytes) = 74 bytes (ver Figura 41)

O tamanho total do pacote (bits) = (74 bytes) * 8 bits por byte = 592 bits

PPS = (8 Kbps codec bit rate) / (160 bits) = 50 pps

Nota: 160 bits = 20 bytes (tamanho do *payload*) * 8 bits por byte

Largura de banda por stream = tamanho do pacote de voz (592 bits) * 50 pps = 29,6 Kbps

É possível resumir a formula da LB = 74 (tamanho da *frame* em bytes)*8 bits por byte*50 = 29.6 Kbps

Largura de banda para *streaming* de voz

CODEC	Taxa de amostragem (msec)	*Payload*	Pacotes/segundo	Largura de banda por chamada
G.711	20	160	50	80
G.711	30	240	33	53
G.729 A	20	20	50	24
G.729 A	30	30	33	16

Tabela 4 Largura de banda necessário para voz e cabeçalho IP[40]

No entanto, é necessário ter em conta os cabeçalhos da camada 2. Por exemplo o codec G.729A a 50pps em *ethernet* 29.6 kbps, em PPP 26.4 kbps em ATM é 42.4 kbps, em *Frame-relay* é de 25.6 kbps, em MPLS é de 25.6 e em WLAN é de 33.6 kbps, ou seja varia entre 25.6 a 42.4 kbps.

CODEC	Ethernet	PPP	ATM	*Frame-relay*
Tamanho do cabeçalho	12	6	53	4
G.711 a 50 pps	85.6	82.4	106	81.6
G.711 a 33 pps	56.5	54.4	70	54
G.729 A a 50pps	**29.6**	**26.4**	**42.4**	**25.6**
G.729 A a 55pps	10.5	17.4	28	17

Tabela 5 Largura de banda com os cabeçalhos da camada 2 incluídos [41]

Largura de banda para sinalização[10]

É necessário reservar largura de banda para o protocolo *skinny*. Verificou-se na prática, que às vezes não é possível estabelecer uma chamada devido ao congestionamento da rede. Existem vários tipos de protocolos de sinalização, o SCCP (*Skinny Client Control Protocol*) (cifrado e não cifrado), H.323, MGCP.

Equação 1: A Cisco recomenda a largura de banda para o tráfego de controlo sem cifragem.

Largura de banda (bps) = 265 * (número de telefones IP e *gateways*)

Na **equação 1** tem em conta os factores como no caso do tráfego do tipo *bursty*, ou com picos de tráfego seguido de períodos de baixa actividade. Por esta razão, não se deve atribuir um valor mínimo uma vez pode resultar em efeitos indesejáveis, devido a atraso em *buffers*, os pacotes são descartados durante períodos de muita actividade.

Por defeito uma fila do *Class-Based Weighted Fair Queuing* (CBWFQ) no IOS Cisco é igual a 64 pacotes. A largura de banda atribuída a esta fila determina o nível ou taxa de serviço desta fila. No caso desta fila chegar ao limite do número de pacotes, então os restantes irão para a fila de *best-effort* ou descartados. Assim é necessário ter um factor de segurança para evitar o *buffer overrun.* No caso de ter cifragem configurada, a largura de banda é afectada porque a cifragem aumenta o tamanho dos pacotes trocados entre o *CallManager* da Cisco e os terminais. É necessário utilizar a seguinte formula:

Equação 2: A Cisco recomenda a largura de banda para o tráfego de controlo com cifragem.

Largura de banda com cifragem (bps) = 415 * (número de telefones IP e *gateways*)

A tabela seguinte resume os valores possíveis para a largura de banda.

Nº de telefones/GW	**LB (sem cifragem)**	**LB (com cifragem)**
1 a 10	8	8
20	8	9
30	8	13
40	11	17
50	14	21
60	16	25
70	19	29
80	21	33

Tabela 6 Largura de banda recomendada em kbps[10]

Para a listagem completo ver no Anexo 8.6.

3. Métodos de translação

De modo a implementar o cenário definido pela a figura seguinte, foi necessário realizar um estudo de mecanismos de transição IPv4 para IPv6.

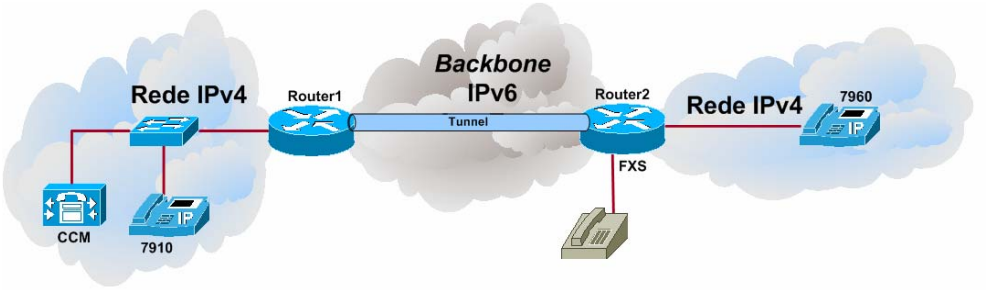


Figura 10 Cenário a implementar

A seguir é apresentado a lista de possíveis soluções, muitos não serão explicados neste relatório. A solução que foi escolha foi o túnel GRE, uma vez que os routers Cisco suportam e portanto torna-se mais fácil de implementar. Outros métodos ainda não têm implementações, ou existem em *Open Source*.

- **Dual Stack**
- **Tunnelling IPv6 sobre IPv4**
  - Automatic tunnelling 6to4
  - Tunnel broker com TSP
  - ISATAP (Intra-Site Automatic Tunnel Addressing Protocol)
  - Teredo
  - Automatic IPv4-Compatible tunnels
  - 6over4
- **Tunnelling IPv4 sobre IPv6**
  - DSTM (com DHCP, RPC, TSP)
  - GRE (Cisco)→implementado neste projecto
- **Translação na camada IP ou de transporte**
  - SIIT (Stateless IP/ICMP Translation)
  - NAT-PT
  - TCP/UDP Relay
  - Bump-in-the-stack
  - Bump-in-the-api
  - Transport-layer Translator
- **Application-layer Gateway**
  - Socks64
  - Application server dual-stack
    - IPv4->IPv6
    - IPv6->IPv4
    - Exemplos:
      - SIP proxy

3.1.1. Introdução

O protocolo IPv6 foi desenvolvimento há alguns anos na *Internet Engineering Task Force* (IETF). O protocolo IPv6 está agora a ter alguma maturidade nas normas *core* ou utilizadas no *backbone*, e os fabricantes agora começam a ter implementações. Um grande desafio é conseguir integrar os serviços de IPv6, de modo a coexistirem em redes IPv4. Existem muitos métodos de integração, no entanto muitos são ainda teóricos e não foram testadas. É necessário disponibilizar os serviços IPv6 (um exemplo é o suporte para mobilidade), é importante manter a qualidade de serviço e não piorá-lo. Os mecanismos de transição definem as técnicas de modo a tornar os *hosts* e *routers* compatíveis com IPv4. Por exemplo os mecanismos de *tunnelling* e mecanismos de *dual stack*. O RFC 4213 (de Outubro de 2005) do IETF com o título "*Basic Transition Mechanisms for IPv6 Hosts and Routers*". Este documento especifica os mecanismos de compatibilidade com o IPv4, que podem ser implementados em *hosts* e *routers IPv6*. Os dois mecanismos especificados são o *dual stack* e o *tunnelling*. O RFC 2185 especifica as questões de transição nas infra-estruturas de encaminhamento. O RFC 2473 investiga sobre as técnicas de *tunnelling*, e o RFC 2529 especifica sem o recurso ao *tunneling*.

3.1.2. **Dual Stack**

O termo "*Dual Stack*" significa um dispositivo de rede, como um *proxy* ou telefone, conseguem comunicar tanto com o protocolo de rede IPv4 ou o protocolo IPv6. A vantagem é que alguns dos protocolos na camada de transporte são iguais em IPv4 e IPv6. Os telefones VoIP modernos irão ser compatíveis com o *dual stack* de modo a suportarem ambos os protocolos. A figura seguinte mostra as entidades do protocolo no sistema operativo.

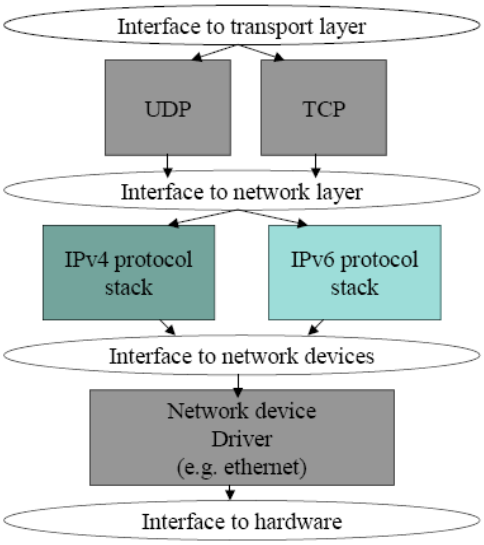


Figura 11 *Dual stack*

Uma máquina *dual stacked* tem uma desvantagem, que é o facto dele necessitar de dois endereços de rede um para IPv4 e outro para IPv6. Para cada telefone na Internet são necessários dois endereços, então o número de endereços IPv4, o que abandona o grande benefício do IPv6. Assim não faz sentido ter um *dual stack* permanentemente no telefone SIP. Faz sentido quando o telefone consegue desactivar a metade da pilha ou *stack.* Outro ponto vista é não ter *dual stack* nos telefones mas sim nos *proxies* SIP. Mesmo assim não resolve o problema porque os terminais têm de comunicar directamente um com o outro, tornando o *dual stack* obrigatório para os terminais, com as consequências bem conhecidas. Assim o *dual stack* não resolve o problema.

3.1.3. *Tunnelling* IPv6 sobre IPv4

O *Tunnelling* [RFC 4213] significa que os pacotes são encapsulados na camada de rede dentro de pacotes de outra camada de rede de um protocolo diferente. Neste caso significa que o pacote IPv6 fica na parte do *payload* do pacote IPv4 que é depois transmitido e encaminhado pela a Internet. Esta técnica permite que ilhas IPv6 sejam compatíveis com outras ilhas IPv6 remotos, tendo um *backbone* IPv4. A figura seguinte mostra esta técnica de encapsulamento.

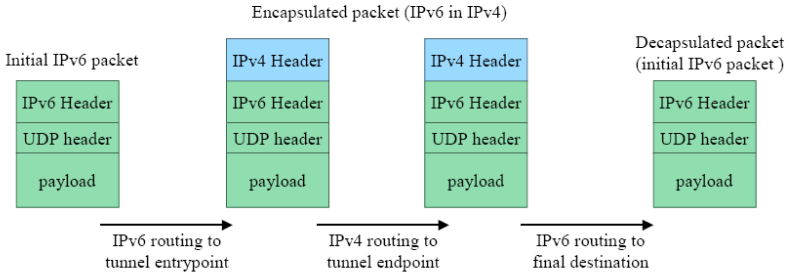


Figura 12 Encapsulamento dos dados IPv6 sobre IPv4

O pacote IPv6 original é transmitido para um router IPv6 que é *dual stacked* e que tem um ponto de entrada IPv6 sobre IPv4. Este router encapsula o pacote com o cabeçalho IPv4 e envia o pacote IPv4 pela a Internet até o término do túnel. Depois é realizado o processo inverso onde o pacote IPv6 original é revelado. Depois é encaminhado para o endereço de destino. A figura seguinte ilustra ligação de rede envolvendo um túnel.

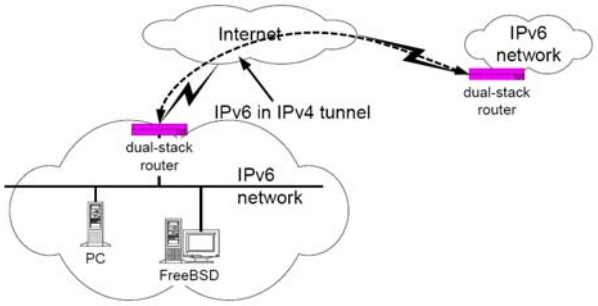


Figura 13 Cenário da utilização de um túnel

É possível ter vários túneis num caminho até ao destino. O túnel é completamente transparente ao utilizador e aos equipamentos terminais que trocam os pacotes IPv6. A técnica de *tunnelling* é utilizada na comunicação IPv6 onde não é possível tê-lo na forma nativa. Este tipo de túnel é chamado "túnel configurado". Isto significa que o administrador de sistemas especifica o túnel com um ponto de saída e de entrada e vice-versa. O método *6-to-4 tunnelling* [RFC3056] da entrada de um túnel selecciona uma saída correcta para o endereço de destino IPv6. A ideia básica é ter uma ilha IPv6 com pelo menos um endereço IPv4, que é atribuído no fim do túnel. Os endereços IPv6 de todos os *hosts* na ilha são construídos no seguinte esquema ou forma (ver figura seguinte):

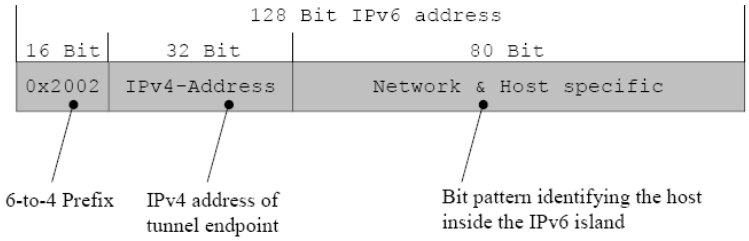


Figura 14 Esquema de endereçamento 6to4

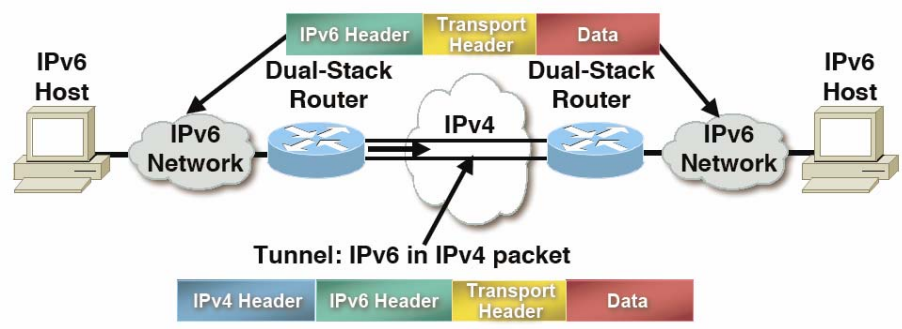


Figura 15 tunnel 6to4 (imagem da Cisco)

O router de fronteira que constrói a entrada do túnel, e recebe o pacote do endereço de partida com o prefixo 6-to-4 (0x2002), ele sabe que a outra ponta do túnel tem o endereço IPv4, porque está indicado depois do prefixo. Assim o pacote é enviado para o túnel correcto, e onde os hosts IPv6 de destino se encontram. Em cada ponta do túnel é necessário que seja implementado o *dual stack* para conseguir utilizar os endereços IPv4 e IPv6.

Tunnelling não permite que um telefone IPv4 comunique com um telefone IPv6, excepto no caso do telefone IPv4 serve de término do túnel ao mesmo tempo. Mesmo neste caso, o *host* IPv6 necessita de saber qual o ponto de entrada para túnel de modo a chegar a um nó especifico, porque ambos os lados do túnel estão associados firmemente.

Mesmo se o endereçamento fosse o problema, o fim do túnel deve lidar com os pacotes IPv6, estes são encapsulados no protocolo de *tunnelling*, que é IPv4.

3.1.3.1. Generic Routing Encapsulation

O GRE é um protocolo de *tunneling* que foi desenvolvido pela Cisco, e consegue fazer mais algumas coisas para além de *tunneling IP-em-IP*. Por exemplo, é possível transportar tráfego *multicast* IPv6 através de um túnel GRE. *Generic Routing Encapsulation* é um protocolo para o encapsulamento de qualquer protocolo da camada de rede sobre outro protocolo da camada de rede. Normalmente o sistema tem um pacote que necessita de ser encapsulado e transportado para o mesmo destino, que é chamado o *payload*.

Primeiro o *payload* é encapsulado no pacote GRE. O resultado é um pacote GRE pode ser encapsulado em outro protocolo e depois reencaminhado. O protocolo exterior é chamado *delivery protocol*. Quando IPv4 está a ser transportado no *payload* do GRE, o campo *Protocol Type* deve ter o valor 0x800. O outro lado do túnel desencapsula o pacote GRE que tem o pacote IPv4 no seu *payload*, o endereço de destino no *header* do pacote (IPv4 *payload*) deve ser utilizado para encaminhar o pacote e o TTL do p*ayload* deve ser decrementado.

Em caso geral, o pacote da camada de rede, é chamado de ***payload packet***, é encapsulado num pacote GRE, que pode incluir a informação da rota de origem. O pacote GRE resultante é encapsulado noutro protocolo da camada de rede, chamado de ***delivery protocol***, e depois encaminhado. Este protocolo descrito no RFC 2784, o pacote GRE é encapsulado da seguinte forma:

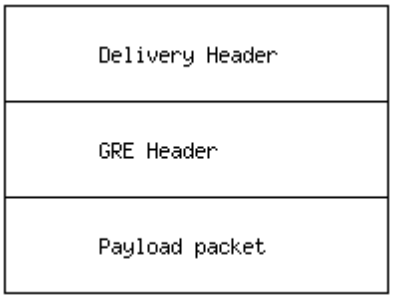


Figura 16 Forma de encapsulamento GRE, cabeçalhos

No anexo 8.8 são descritos os campos sobre o protocolo GRE.

É necessário ter cuidados ao encaminhar o pacote, uma vez que o endereço de destino do *payload* do pacote é o *encapsulator* do pacote (i.e. o outro lado do túnel), o *looping* pode ocorrer. Neste caso, o pacote deve ser descartado. O protocolo é utilizado quando os pacotes GRE são encapsulados em IPv4.

Segurança

A segurança de uma rede que utiliza um túnel GRE deve ser semelhante a uma rede IPv4 normal, uma vez que o mecanismo de encaminhamento é o mesmo que IPv4 nativa. A utilização de *route filtering* não irá mudar. Contudo é necessário ter uma *firewall* para analisar o dentro do pacote GRE ou a filtragem seja realizado nas pontas do túnel GRE (entrada e saída). Neste ambiente é aconselhado que o túnel termine numa *firewall.* Existem muitos artigos que investigam (*Hacker*s) as vulnerabilidades do protocolo GRE.[5]

Túneis que a Cisco implementa

A Cisco implementa vários mecanismos de transição. É necessário ter em conta a versão do IOS. É possível consultá-lo no seguinte *link:* (http://www.cisco.com/en/US/products/sw/iosswrel/ps5187/wp1027171)

Existem cinco métodos de *tunneling* para o tráfego IPv6, implementados pela a Cisco:

- *Manual IPv6 tunnels*
- *Automatic IPv4-Compatible tunnels*
- GRE
- *Automatic 6to4 tunnels*
- *Intra-Site Automatic Tunnel Addressing Protocol* (ISATAP) Tunnels

Maior parte destes túneis converter IPv6 para IPv4. O único tipo de túnel que consegue converter de IPv4 para IPv6, é o GRE.

Cabeçalhos de Extensão (*Extension Headers*)

No processo de encapsulamento GRE IPv4 em IPv6, o protocolo GRE fica encapsulado no IPv6, através dos Cabeçalhos de Extensão ou *Extension Headers*. Os Cabeçalhos de Extensão (*Extension Headers*) [53] são os responsáveis por grande parte da eficiência do IPv6. Os cabeçalhos de extensão podem ser vistos numa localização intermédia, existente entre a camada de rede e a camada de transporte, não fazendo parte do normal cabeçalho IPv6, apesar de o complementarem. Utilizam níveis hierárquicos como forma de atingir a

[5] www.darklab.net/resources/irpas/gre.pdf ou www.phenoelit.de/irpas/gre.html

máxima eficiência, e esse nível é baseado no número de dispositivos que o precisam de processar. Quanto maior o número de dispositivos, maior a prioridade do cabeçalho de extensão. Sendo assim, a razão da hierarquia está associada à frequência com que é necessário o processamento do cabeçalho. A Figura 3.4 mostra a ordem hierárquica que determina a ordem dos cabeçalhos de extensão.

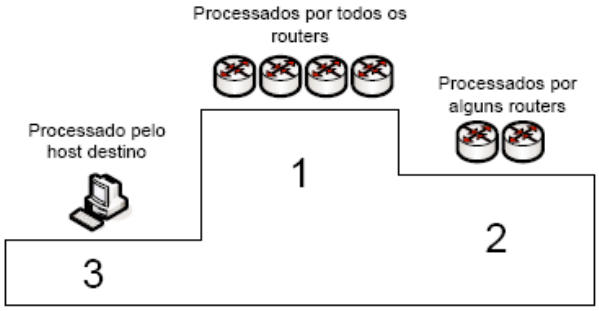


Figura 3.4 – Ordem hierárquica determina a ordem dos cabeçalhos de extensão.

Podem existir vários cabeçalhos de extensão (CE), ou pode não existir nenhum. No caso de existirem vários, interligam-se todos uns com os outros através do campo *Next Header*. Estes cabeçalhos formam, assim, uma corrente, até que o último cabeçalho de extensão aponte, por exemplo para um protocolo da camada acima. A Figura 3.5 exemplifica este processo de interligação.

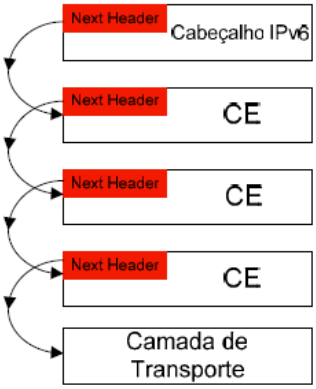


Figura 3.5 – Forma de como os cabeçalhos de extensão se interligam.

Os cabeçalhos de extensão não são examinados nem processados por todos os nós, a não ser pelo nó que contém o endereço do campo *Destination Address* do pacote. A única excepção é o cabeçalho *Hop-by-Hop Options*.

Alguns cabeçalhos de extensão são bastantes parecidos e completamente novos (*Hop-by-Hop Options header*, *Destination Options header*), outros não trazem novidades (*Authentication header*, *Encapsulating Security Payload*), e outros são um aproveitamento de alguns conceitos anteriores, misturados com algumas novidades (*Routing header*, *Fragment header*).[19]

3.1.4. DSTM

Não existem soluções únicas para o problema de translação IPv4 para IPv6. Diferentes mecanismos têm sido propostos, cada um considerando um processo de translação. DSTM é apenas para redes IPv6 que têm *hosts* que necessitam de trocar informação com os *hosts* IPv4 ou aplicações IPv4. Esta solução não foi utilizada no cenário uma vez que necessita de um servidor DSTM, e também existem pouco as implementações deste mecanismo.

Uma ideia como funciona o DSTM. Na figura é possível identificar três tipos de equipamentos:

1. Um *host Dual-stack* que está numa rede IPv6 e quer comunicar utilizando um endereço IPv4.
2. Um servidor DSTM que administra a *pool* de endereços IPv4.
3. Um *gateway* DSTM (ou TEP) que tem a função de encapsular e desencapsular os pacotes IPv4 sobre IPv6. A máquina C tem de *ter* conectividade IPv4 e ter um endereço IPv4 permanente.

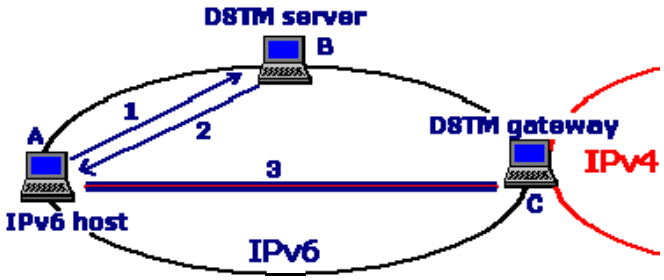


Figura 17 Arquitectura DSTM

Quando um *host* num domínio IPv6 (máquina A na figura) necessita de comunicar com IPv4, o primeiro passo consiste em perguntar ao servidor DSTM o endereço IPv4 temporário. (1) O servidor DSTM (B) reserva um endereço IPv4 para A da pool de endereços e envia-o na resposta. A mensagem de resposta (2) também contém um tempo de validade associado ao endereço alocado e a informação sobre o *gateway* DSTM, o TEP. Depois da mensagem ter sido trocada (o que pode ser realizado utilizando DHCPv6, RPC, ou um formato proprietário) a máquina A configura o *stack* IPv4 com o endereço alocado. Do ponto (3), todos os pacotes que chegam de A vão pelo o túnel (IPv4 sobre IPv6) até ao TEP (C). Para executar o encapsulamento / desencapsulamento dos pacotes IPv4, o *gateway* DSTM (C) guarda uma tabela contendo endereço IPv4 e IPv6 dos *hosts* intranet.

De modo a garantir uma comunicação bidireccional, o encaminhamento IPv4 tem de estar garantido, assim qualquer pacote com o destino para A passar por C. [12]

3.1.5. NAT-PT

Este método baseia-se em ter uma instância *dual stacked* na rede que consegue fazer a translação dos pacotes IPv6 em IPv4 e vice-versa. Ao contrário que o *tunnelling*, não há encapsulamento, o que significa o nó de origem do pacote pode ser apenas IPv4 ou IPv6 (*single stack).* Quando o pacote IPv6 chega ao nó NAT-PT, ele remove o cabeçalho do IPv6 e substitui com o cabeçalho do IPv4, depois copia os campos IPv4 para os campos do cabeçalho do IPv6. Depois o pacote é encaminhado normalmente para o endereço de destino com o novo cabeçalho do pacote. O nó NAT-PT administra a *pool* de endereços IPv4 para dinamicamente atribuir aos nós IPv6 através do nó NAT-PT.

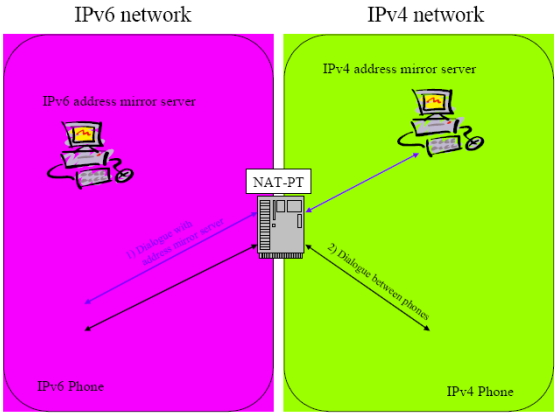


Figura 18 Diagrama lógico do NAT-PT

Cada nó NAT-PT tem uma lista de *Application Layer Gateway* (ALG), que olha para dentro o pacote e se necessário fazer modificações nos dados. Um exemplo de ALG para o *Domain Name System* (DNS). A figura seguinte descreve o mecanismo de NAT-PT, com dois modos de operação. O primeiro modo de translação de pacotes IPv6 para IPv4 e o segundo de IPv4 para IPv6. Exemplo de uma translação NAT-PT a nível das mensagens, usando o protocolo SIP versão 2, de acordo com a figura seguinte:

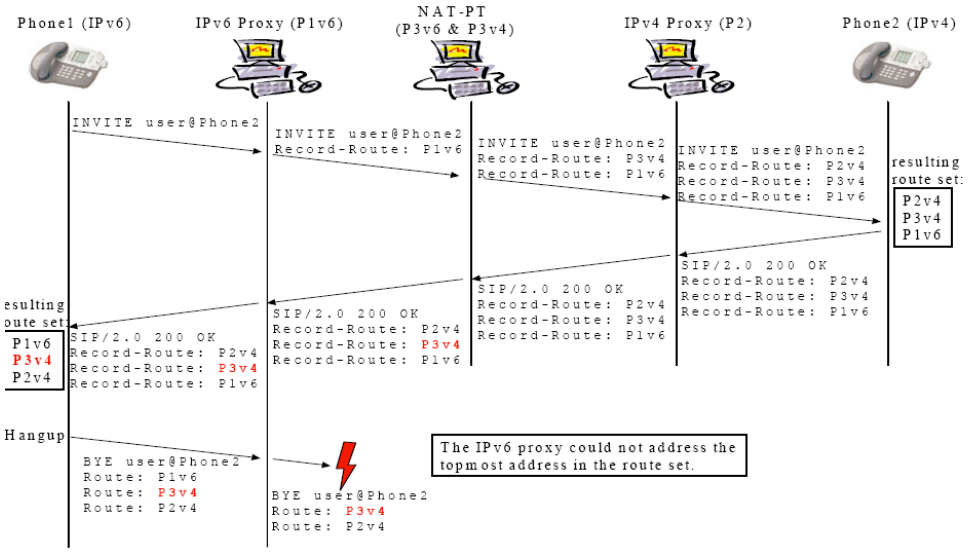


Figura 19 NAT-PT tradução de endereços

3.1.6. Application Layer Gateways

Os Application Layer Gateways (ALGs) são geralmente equipamentos *dual-stack*, estes equipamentos conseguem comunicar em IPv4 ou IPv6. Os ALGs não são dispositivos de translação, em vez disso, eles funcionam como *relays* na camada de aplicação. Em vez de enviar um pedido transaccional para o servidor de aplicação, a aplicação cliente envia-o para o ALG. O ALG depois envia o pedido ao servidor, para o cliente. O ALG é responsável para enviar todos os dados recebidos do servidor novamente para o cliente.

Um exemplo típico é o *web cache* ou proxy. O servidor SMTP também é visto como ALG.

DNS-ALG

Quando um pacote chega ao nó NAT-PT, o pacote chega um selector ALG, que decide qual o ALG que será aplicado ao pacote. A figura seguinte ilustra o selector ALG.

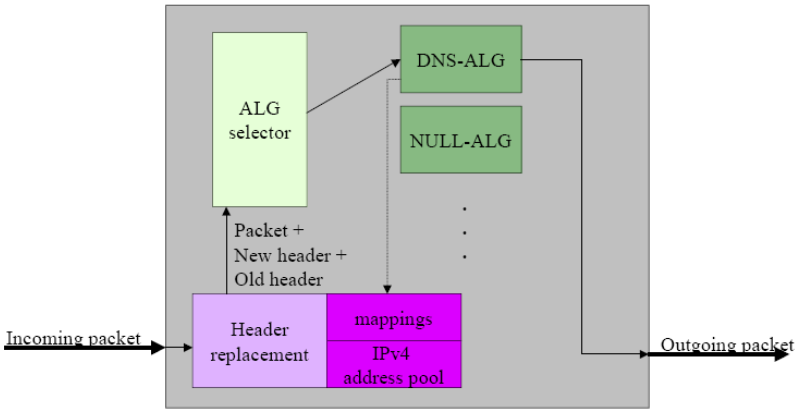


Figura 20 Selector ALG

Apesar do fluxo de dados e as instâncias internas ao NAT-PT. O objectivo do DNS ALG reconhecer se o pacote é um pedido ou resposta DNS, e aplicar as alterações necessárias. O DNS-ALG reconhece os pacotes relacionados com o DNS através do número do porto que é o 53. Depois será substituído o endereço IPv6 irá ser substituído por um endereço IPv4 e vice-versa.

Translação IPv6 para IPv4

Como foi definido no *Internet Protocol Version 6*, IPv6, Existe sempre um endereço IPv6 (sub-endereço) para cada endereço IPv4. Este endereço pode ser obtido usando o prefixo de 96 bits (::ffff). Contudo o *host* IPv6 permite enviar dados para o *host* IPv4, ele ou sabe o nome, ou o seu endereço IPv4. Em primeiro lugar ele não irá saber se o sistema é IPv4 ou não, por causa do DNS-ALG, o *lookup* irá devolver um endereço IPv4 mapeado do endereço IPv6, que pode ser utilizado na comunicação. No segundo caso, ele apenas necessita de estender o endereço IPv4 por um prefixo de 96 bit ::ffff, para ir buscar o endereço IPv4 mapeado no endereço IPv6 do destino. Na comunicação VoIP utilizando SIP, existe a necessidade de ter um conhecimento especial do telefone IPv6, de modo a interpretar os endereços IPv4. O telefone pode enviar o pacote, que irá chegar ao nó NAT-PT. O *host* NAT-PT irá remover os 96 bits de prefixo e irá depois remover o cabeçalho IPv6 e colocar o cabeçalho IPv4 no seu lugar.

Translação IPv4 para IPv6

Este tipo de translação é mais complexa. Para o envio do pacote IPv4 para um sistema IPv6 é necessário utilizar o nome lógico. Esta é a única maneira de estabelecer um mapeamento no NAT-PT. Assim o DNS-ALG é essencial para o NAT-PT funcionar. No caso do pacote IPv4 chegar ao nó NAT-PT. No caso de não encontrar a entrada no DNS então ele irá enviar uma mensagem de erro e enviar os pacotes de voltar à origem. Se encontrar então irá substituir o cabeçalho IPv4 com o cabeçalho IPv6.

3.1.7. Proxy

No caso de não existir NAT-PT, e o protocolo SIP necessita de estabelecer as sessões, através dos *proxies* para o processamento de mensagens SIP. Estes proxies têm de ser de suporta *dual stack.* Também é necessário existir um *relay* (RTP *relay*) ou encaminhador multimédia.

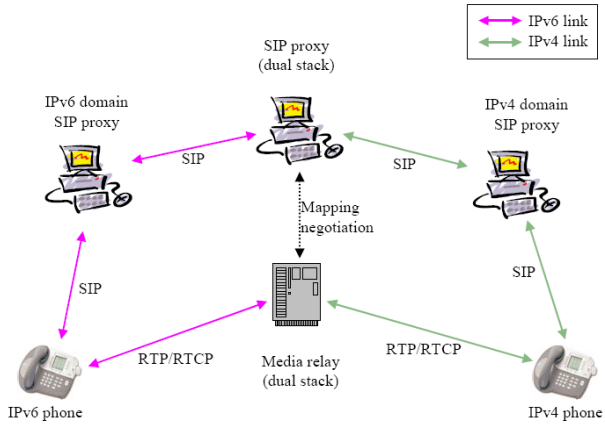


Figure 25 Cenário com a utilização do Proxy

Estes telefones devem poder especificar o proxy, em cada pedido que ele envia, ou seja, e deve saber quem é o proxy. Esta característica é chamada "*outbound proxy*", ou "d*efault route*", porque este proxy pode ser considerado como uma rota no telefone. Cada telefone SIP deve ter estas características, para que o ISP poder facturar as chamadas realizadas [36].

4. Mecanismos de Qos

4.1. Introdução

O protocolo IP (*Internet Protocol*) é responsável pelo o sucesso da Internet. O protocolo IP oferece um serviço de *best-effort* para o transporte dos pacotes e funciona em praticamente qualquer meio de transmissão e plataforma. O aumento da popularidade do IP é o facto de muitas aplicações funcionarem sobre IP. De modo a suportar muitas aplicações como *streaming* de vídeo, voz sobre IP (VoIP), comércio electrónico, e outros, a rede necessita de Qualidade de Serviço ou *Quality of Service* (QoS) para além do serviço do *best-effort*. Diferentes aplicações variam em termos de atraso, variação do atraso (*jitter*), largura de banda, perda de pacotes, e disponibilidade. Estes parâmetros são a base do QoS. Uma rede IP tem de ter um determinado QoS, para satisfazer as exigências das aplicações.

4.1.1. Enquadramento

A investigação na área da qualidade de serviço na Internet tem sido efectuada, sob a égide do IETF, em torno de dois modelos arquitecturais: o modelo de Serviços Diferenciados (DiffServ) e o modelo de Serviços Integrados (IntServ). A arquitectura IntServ surgiu no âmbito do grupo INTSERV-WG do IETF. Esta arquitectura caracteriza-se, fundamentalmente, pelo fornecimento de garantias de Qualidade de Serviço a cada fluxo, individualmente, através de uma reserva de recursos ao longo da rede.

Devido aos problemas de escalabilidade desta arquitectura nos *backbones* das grandes redes, foi criado o grupo DIFFSERV-WG do IETF. Este grupo de trabalho desenvolveu a arquitectura DiffServ onde, em vez de um tratamento individualizado dos fluxos, propôs um agrupamento num conjunto pequeno, mas bem definido de Classes de Serviço (CoS), passando assim a fornecer um tratamento diferenciado ao tráfego sem requerer mecanismos de sinalização e reserva de recursos.

Para a implementação do QoS (ver capítulo dos testes) foram realizados cálculos relacionados com largura de banda (ver 2.2.5) e parâmetros QoS. Os parâmetros ou métricas QoS são a seguir descritos.

4.2. Parâmetros de QoS

Com as novas aplicações nas redes de dados, é absolutamente necessário prever qual o impacto que certas aplicações podem ter na rede. As medições activas são geradas pelo administrador de rede para determinar qual o impacto que essas aplicações podem ter na rede, e se deve negá-lo ou não. Nas medições passivas, apenas é medido o tráfego gerado pelos os utilizadores, no seu funcionamento normal da rede.

Há alguns anos atrás, era possível aumentar a largura de banda e no caso de *bursts* ou picos de tráfego, os fluxos têm sempre os mecanismos de retransmissão e de *timeout*. No entanto, com as novas aplicações, como voz e vídeo, estão mais susceptíveis a estas mudanças na rede. No caso da voz sobre IP é muito susceptível ao tráfego da rede, e as métricas como o atraso ou *jitter* se ficarem muito afectados, podem degradar a aplicação de voz ao ponto de não estar aceitável ao utilizador. É muito importante que o administrador, não faça apenas medições a nível das ligações e dos equipamentos, mas também a nível dos terminais e saber se os utilizadores da rede estão satisfeitos com a qualidade do serviço.

Throughput ou débito

O *throughput* é a taxa de transmissão dum computador ou rede quando envia e recebe dados. Esta é uma boa medida da capacidade da ligação.

Largura de Banda

A largura de banda é a medida que expressa a capacidade de transmissão de dados, normalmente em bits por segundo (kbps). A largura de banda indica a capacidade máxima teórica de uma ligação. Uma analogia pode ser realizada entre a largura de banda e o diâmetro de um tubo, deste modo, quanto maior é o diâmetro mais dados podem ser enviados, simultaneamente, através dele.

Latência ou atraso

Latência é a medida do tempo de resposta (de um único pacote) desde do cliente até à sua resposta (*acknowledgment*).

Jitter (variação no atraso)

Quando as *frames* são transmitidas pela a rede IP, a quantidade de atraso de cada *frame* pode variar. Devido ao atraso nas filas e no tempo de processamento e se existe carga na rede. Apesar dos *gateways* de voz enviarem *frames* em intervalos regulares (por exemplo cada 20 ms), o destinatário não irá receber as *frames* em intervalos regulares, o que provoca *jitter* (ver a Figura 21 Cálculo do *Jitter*). A forma de combater o problema é armazenar as *frames* num *buffer* o tempo suficiente para as *frames* que chegam mais atrasadas conseguirem chegar. Depois são todas enviadas na sequência correcta e em tempos regulares.

Quanto maior o *jitter*, mais tempo as *frames* irão ser guardados no *buffer*, o que introduz atraso. Para minimizar o atraso devido ao *buffering,* muitas implementações utilizam o *adaptive jitter buffer*. Por outras palavras, a quantidade de *jitter* na rede é pequena, e o tamanho do *buffer* irá também ser. No caso do *jitter* aumentar devido ao aumento da carga na rede, o tamanho do *buffer* irá automaticamente aumentar para compensar. Às vezes o *jitter* é tão grande, o *buffer* pode descartar alguma *frame* para manter o atraso.

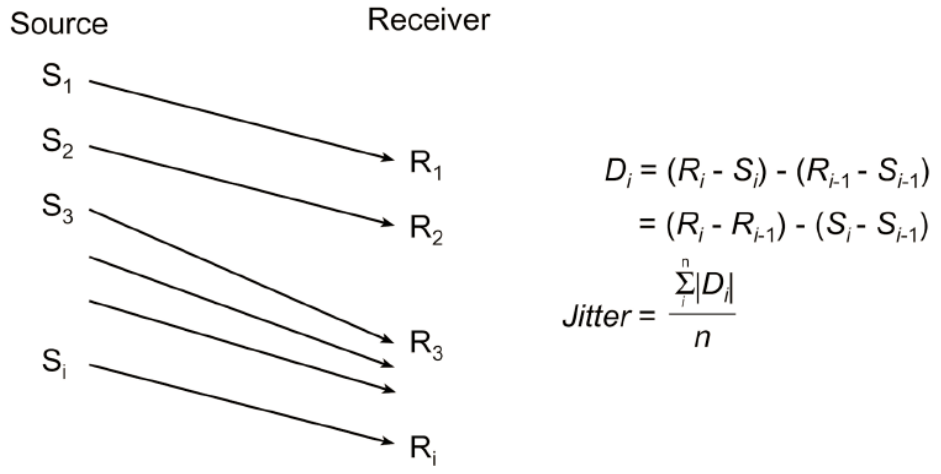


Figura 21 Cálculo do *Jitter*

Perda de pacotes

A perda de pacotes ao longo do caminho, pode degradar a qualidade na voz. Antes de implementar aplicações VoIP, é importante determinar o atraso, *jitter* e perdas de pacotes na rede de modo a determinar se estas aplicações e conseguirem funcionar correctamente.

Limites ou requisitos para transporte de voz em tempo real[39]

O atraso ponto a ponto, não deve ser superior a 150 ms para chamadas de alta qualidade (ITU-T G.114) . Na prática um atraso de até 200 ms é tolerável.

A variação do atraso (*jitter*), variação máxima tolerável entre 20 a 50 ms e depende dos *buffers* adaptativos de compensação de *jitter* e dos outros atrasos que compõem o atraso ponto a ponto.

As perdas de pacotes, as perdas na rede e descartados no *buffer* de compensação de *jitter* não devem ultrapassar 1% para chamadas de alta qualidade.

4.3. A solução ToS/IP precedence (antecessor)

No cabeçalho IPv4 o campo TOS é mostrado na Figura 22. Os três bits *precedence* ou precedência, são utilizados para classificar os pacotes no extremo da rede entre oito categorias, que estão listadas na Figura 23. Os pacotes com a precedência mais baixa, são descartados em favor de pacotes com precedência mais elevada, quando existe congestionamento na rede. Cada pacote pode ser marcado, num dos seguintes níveis atraso, *throughput*, e fiabilidade (bits DTS) no seu processo de *forwarding*

Problemas e limitações:

- O esquema de precedência permite especificar a prioridade relativa do pacote. Não permite, por exemplo, o administrador de rede especificar HTTP e *telnet* como pacotes de alta prioridade.
- Os bits IP *precedence* e DTS não são implementados consistentemente pelos fabricantes. O RFC 1349 redefine o campo TOS, utilizando os bits 3,4,5 e 6, e eliminam o conceito de DTS.

Na prática, apesar de existir em algum software, que utilizam estes bits, os bits TOS têm sido pouco utilizados e geralmente ignoram o *IP precedence* e também tem pouca utilização. As implementações têm sido limitadas, a um pequeno número de domínios, principalmente em implementações *ad hoc* normalmente experimentais. A utilização mais conhecida é a nível de *routing updates* com a utilização do *high precedence* em alguns domínios.[16]

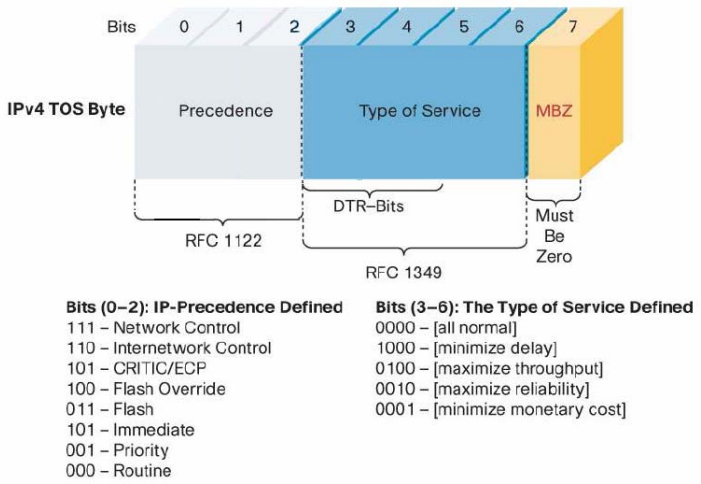


Figura 22 O byte original *ToS* no *IPv4*

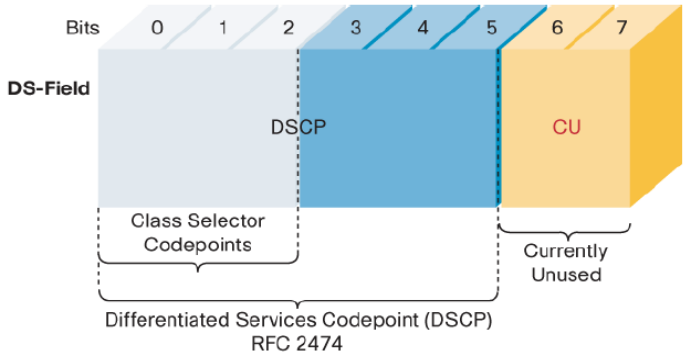


Figura 23 O campo *Codepoint* do *Diffserv* [25]

Modelos de Qualidade de serviço QoS

No caso da VoIP necessita de um *jitter* muito baixo, um atraso de 150 ms num sentido, e largura de banda na ordem dos 8 kbps a 64 kbps, dependendo do codec utilizado. Outro exemplo, é a transferência de ficheiros, baseada em ftp, não existe *jitter*, no entanto a perda de pacotes poderá afectar o *throughput* ou débito.

O Internet Engineering Task Force (IETF) definem dois modelos de QoS: *Integrated Services* (IntServ) e *Differentiated Services* (DiffServ).

4.4. Intserv

A arquitectura *Integrated Services* (IntServ) foi definida com o objectivo de fornecer um conjunto de extensões ao serviço *best effort* existente na Internet. Estas extensões permitem o suporte de aplicações em tempo real na Internet que necessitem de garantias de QoS como, por exemplo, telefonia, videoconferência e distribuição de vídeo. Qualidade de serviço no contexto do IntServ refere-se ao serviço de entrega de pacotes fornecido pela rede, sendo caracterizada pela largura de banda, atraso por pacote e perdas de pacotes.

A arquitectura IntServ caracteriza-se pela reserva de recursos entre os dois extremos de uma comunicação ao longo da rede, capaz de satisfazer a QoS pretendida pelos fluxos de dados das aplicações. Neste contexto, um fluxo identifica um conjunto de pacotes resultantes de determinada actividade de um utilizador na rede, por exemplo, uma sessão de FTP. Pode ser identificado por um conjunto de parâmetros como, por exemplo, no IPv6, o endereço IP origem e o *flow label* e, no IPv4, o protocolo, o porto origem e destino e os endereços IP origem e destino.

Nesta secção é abordada a arquitectura bem como os módulos que a constituem. Depois, são caracterizadas as aplicações existentes, de acordo com os requisitos de atraso, perda de pacotes e variação de atraso. Por último, são abordadas as classes de serviço definidas pelo IntServ, destinadas a suportar estas aplicações.

4.4.1. RSVP

O protocolo de reservas de recursos é utilizado para transportar o pedido de QoS a cada elemento de rede ao longo de todo o percurso dos dados ou, no caso de *multicast*, a cada *router* pertencente aos ramos da árvore de encaminhamento. O RSVP foi desenvolvido para funcionar como protocolo de estabelecimento e manutenção de reserva de recursos numa rede. Como o RSVP é um protocolo *simplex*, as reservas de QoS são efectuadas

apenas numa direcção, do receptor para o emissor. Para ligações *duplex*, tais como conferências de áudio e vídeo onde o emissor é também receptor, é necessário estabelecer duas sessões RSVP.

Na arquitectura IntServ, as mensagens RSVP são enviadas para cada elemento da rede através da camada de rede. Estas mensagens são encapsuladas dentro do campo de dados do protocolo IP com o número de protocolo 46. O atraso da rede e QoS são os factores mais complicados de gerir numa rede com dados e voz. Uma solução para este problema é o *Resource Reservation Protocol,* que foi desenvolvido pelo o IETF. O RSVP consegue prioritizar e reservar recursos para garantir uma latência suficientemente baixa para *streams* IP específicos. Ao longo do caminho o RSVP faz pedidos para reservar de nós ao longo deste caminho. O pedido RSVP apenas é enviado numa direcção. Uma outra reserva deve ser realizado no sentido contrário. O RSVP não é um protocolo de encaminhamento, mas poderá operar com os protocolos actuais e futuros de *unicast* e de *multicast.* Para eficiência o RSVP tem receptores responsáveis que pedem um determinado QoS. O pedido QoS passa da aplicação *host* para o processo RSVP local. O protocolo RSVP transporta o pedido para todos os nós, no caminho inverso da origem dos dados. O RSVP é útil em pelo menos dois ambientes, para o controlo à admissão em telefonia IP, e em MPLS *traffic engineering*, no caso de ter MPLS com IPv6 nativo.

Nota Importante: A Cisco não suporta actualmente RSVP para IPv6, segundo o Project Manager da Cisco[6]. Também verificou-se no IOS 12.4 da Cisco, que não existe por exemplo o comando "**ipv6 rsvp bandwidth 1158 1158**". E no *command reference* deste IOS 12.4 não existe numa referência para o RSVP IPv6[7].

O modelo IntServ, depende do protocolo *Resource Reservation Protocol* (RSVP) para sinalizar e reservar o QoS desejado para cada fluxo da rede. Uma *stream* é um fluxo de dados, unidireccional entre duas aplicações, e é unicamente identificado, pelo o conjunto de 5 elementos (endereço IP de origem, número do porto de origem, endereço IP de

[6] Segundo o Product Manager da Cisco do IOS da Cisco para IPv6, Patrick Grossetete <pgrosset@cisco.com>

[7] http://www.cisco.com/univercd/cc/td/doc/product/software/ios124/124cr/hqos_r/qos_i1h.htm

destino, número do porto de destino, e protocolo de transporte). Dois tipos de serviços podem ser requisitadas pelo o RSVP (assumindo que todos os dispositivos da rede suportam RSVP, ao longo do caminho desde da origem ao destino).

O primeiro tipo é mais rigoroso, o que permite ter um atraso e largura de banda, de acordo com as especificações de reserva.

O segundo tipo é um serviço de carga controlada, que oferece um melhor *best effort* e um atraso baixo. É possível (pelo menos teóricamente) oferece QoS requisitada, para cada fluxo da rede, se sinalizar o RSVP e os recursos disponíveis.

Desvantagens:

- Ao longo do caminho percorrido pelos pacotes, cada equipamento, incluindo servidores e PCs, têm de suportar RSVP e serem capazes de sinalizar o QoS necessário.

- As reservas realizadas em cada equipamento, têm de ser periodicamente actualizadas, assim irá adicionar mais tráfego à rede.

- Manter o estado em cada router, combinado com o controlo de admissão em cada salto e aumento nos requisitos de memória para suportar todas estas reservas, adiciona à complexidade de cada nó da rede ao longo do percurso.

- A informação sobre o estado da reserva, necessita de estar em cada router e ao longo do caminho, o que torna pouco escalonável no caso de ter centenas ou milhares de fluxos pela rede *core*. Assim deu origem ao "RSVP *Refresh Reduction and Reliable Messaging*," "RSVP *scalability enhancements,"*, Proxy RSVP e outras características torna o RSVP mais escalonável e implementável.

4.5. DiffServ

O modelo IntServ apresenta diversas limitações relacionadas fundamentalmente com a escalabilidade devido ao tratamento individualizado dos diferentes fluxos e ao elevado número de mensagens necessárias ao funcionamento do RSVP.

Em alternativa ao tratamento individualizado de fluxos, surgiram propostas com o objectivo de dar um tratamento de tráfego diferenciado, sem, no entanto, requerer

mecanismos de sinalização e reserva de recursos. Inicialmente, nesta secção, serão abordadas as contribuições que estiveram na origem da arquitectura DiffServ. Posteriormente, será abordada a arquitectura *Differentiated Services* (DiffServ) que foi objecto de estudo do grupo DIFFSERV-WG da IETF.

Com base nas propostas do 2*bit*, foi criado um grupo de trabalho com o objectivo de criar uma arquitectura de QoS, sem recurso a mecanismos de sinalização e sem a necessidade de manutenção do estado da comunicação em todos os nós da rede.

A arquitectura tem como principal objectivo a disponibilização de uma gama de serviços para o tráfego na Internet sem necessidade de efectuar sinalização nem reserva de recursos, por fluxo em cada *router*. O tráfego é dividido num número pequeno de classes com requisitos de QoS distintos, tendo cada uma um tratamento diferenciado.

A arquitectura DiffServ transfere a complexidade do núcleo das redes para a periferia, onde o volume de tráfego e os fluxos são menores, passando a oferecer serviços por agregados de tráfego em vez de fluxos de tráfego. Desta forma, é garantida a escalabilidade nas redes DiffServ. O termo serviço, nesta arquitectura, refere-se às características mais importantes na transmissão de pacotes ao longo da rede. Estas características podem ser especificadas quantitativamente, através do débito, atraso, *jitter* e perdas, ou em termos de prioridade relativa no acesso aos recursos da rede.

4.5.1. Arquitectura DiffServ

O DiffServ tem uma região *Differentiated Services* DS, composto por um ou mais domínios DS, possivelmente sob várias autoridades administrativas. Cada domínio DS está preparado para utilizar o DSCP em diferentes PHBs. Na Figura 24 dá uma ideia geral da arquitectura ponto a ponto. Para existir QoS, é necessário que o caminho IP que os pacotes seguirão deverá ter o Diffserv activado. Por exemplo o *service policy* – EF fica com 10 %, *gold* 40%, *silver* 30%, e *bronze* 10%, e o tráfego *best effort* (class por defeito /PHB) fica com os 10% de largura de banda restante. *Gold*, *silver*, e *bronze* pode ser mapeados em classes AF *Assured Forwarding* PHB (Defined in RFC-2597) AF1 e AF2, e AF3, por exemplo. Este pode ser forçada a qualquer parte da nuvem, incluindo ponto a ponto.

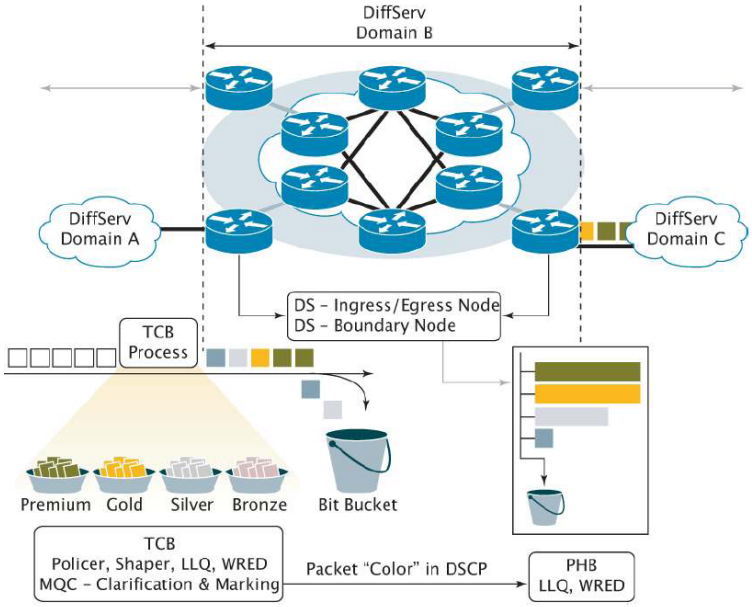


Figura 24 Arquitectura DiffServ

Um domínio DS é constituído por nós DS de entrada (*ingress*), nós DS interiores (no core), nós DS à saída (*egress*). Os nós à entrada e à saída podem estar na fronteira do nó, ligando dois domínios DS. Funcionalmente, todos os nós à entrada e à saída pode ser categorizadas como nós de fronteira, mas como servem de ponto de demarcação entre o domínio DS e num rede que não suporta DS (L2 LAN, etc.). Tipicamente, os nós de fronteira DS realizam condicionamento de tráfego.

Um condicionador de tráfego, tipicamente classifica os pacotes que estão a entrar em agregados predefinidos, o *Meter* ou Medidor depois determina a necessidade através dos parâmetros (e determina se o pacote está no perfil, ou não), marca-os apropriadamente escrevendo/e reescrevendo o DSCP, e afina os *buffers* para conseguir um fluxo mais baixo ou descarta o pacote em caso de congestionamento.

A Figura 25 ilustra o condicionado de tráfego típico no extremo ou fronteira do domínio DS. O nó DS interno reforça o PHB apropriado ao implementar técnicas de políticas ou *shaping*, e às vezes tornar a marcar pacotes dependendo da política ou SLA. [17]

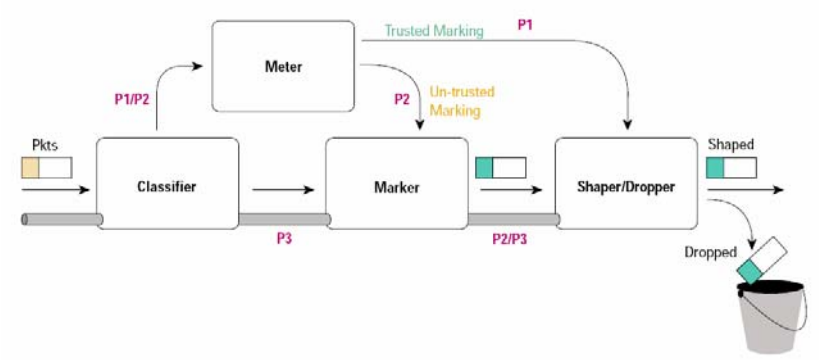


Figura 25 Traffic Conditioner Block (TCB) do DiffServ

Módulo que implementam o serviço:

1. **Classificador/*Classifier***
   - Distribui os pacotes pelas classes em função do *codepoint*
   - Selecciona o pacote da *stream* de tráfego baseado no seu conteúdo na porção do cabeçalho do pacote.
2. **Medidor/*Meter***
   - Mede o tráfego e assegura a sua conformidade com o *profile* ou perfil.
3. **Marcador/*Marker***
   - Policia o tráfego e remarca-o se necessário. Escreve e reescreve o valor de DSCP
4. ***Shaper***
   - Realiza ajustes temporais ao tráfego, como atrasar pacotes.
5. **Descartador/*Dropper***
   - Descarta pacotes quando necessário.

IntServ segue o modelo de sinalização, onde os terminais (*end-hosts*) dizem à rede qual o QoS que precisa, enquanto que o DiffServ funciona de acordo, com um modelo de QoS de aprovisionamento, onde os elementos irão criar várias classe de tráfego conforme varia as necessidades de QoS. Ambos os modelos podem ser construídos a partir de uma politica base, utilizando o protocolo CoPS (*Common Open Policy Server).* IntServ oferece uma boa solução de QoS ponto a ponto, utilizando sinalização ponto a ponto, manutenção do estado (para cada fluxo e reserva RSVP) e admissão de controlo em cada elemento da rede.

O DiffServ, por outro lado, utiliza métodos para classificar o tráfego em diferentes classes, chamadas *Class of Service* (CoS), e aplica os parâmetros QoS a estas classes. Cada pacote

é primeiro dividido em classes através da marcação no campo *Type of Service* (ToS) (1 byte) no cabeçalho dos pacotes IP.

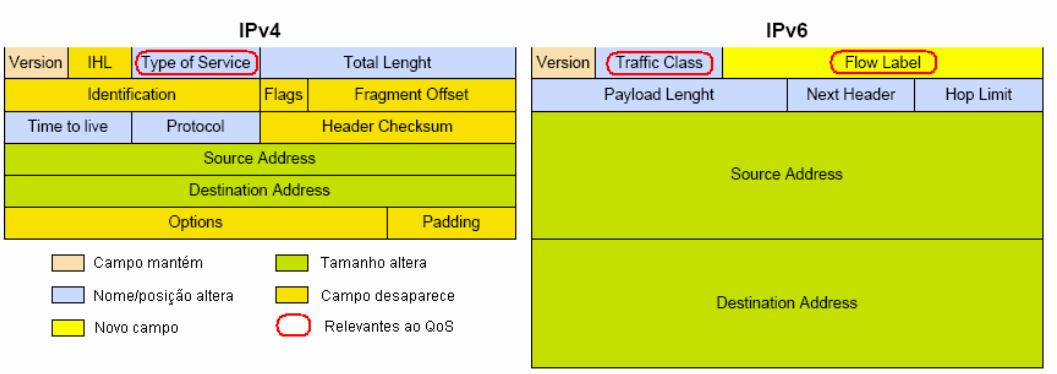


Figura 26 Cabeçalhos IPv4 e IPv6

Um padrão de 6 bits (chamada de *Differentiated Services Code Point* [DSCP])) no octeto TOS do IPv4, ou no octeto *Traffic Class* no IPv6 são mostrados na figura anterior (faz sentido dentro do diffserv).

No extremo da rede, os pacotes sofrem um tratamento de encaminhamento ou *forwarding*, que chama-se *Per-Hop Behavior* (PHB), é aplicado a cada elemento de rede, dado ao pacote um valor de atraso, *jitter*, e largura de banda, etc. A combinação da marcação de pacotes e um PHB bem definido, numa solução QoS escalonável para qualquer tipo de pacotes, qualquer tipo de aplicação. No DiffServ, a sinalização para QoS é eliminado, tornando-o mais escalonável.

O IETF completou o RFC para DiffServ no fim do ano 1998. Um pequeno padrão de bits em cada pacote, no octeto ToS do IPv4 ou no octeto *Traffic Class* no IPv6, é utilizado para marcar os pacotes para depois receber um determinado tratamento de *forwarding* em cada nó da rede. Para que estes bits sejam utilizados e interpretados, é necessário para a sua utilização entre domínios seja inter operável, e ter saber os comportamentos esperados nas redes agregadas.

Assim, foi normalizado o *layout* deste campo de 6 bits para ambos os octetos, chamado DS. O RFC 2474 e RFC 2475 definem a arquitectura, em geral a utilização dos bits no campo DS (sobre as definições do TOS em IPv4 do RFC 1349). De modo a transportar QoS ponto a ponto, esta arquitectura (RFC-2475) tem dois componentes, a marcação de pacotes utilizando o byte do IPv4 TOS e as PHBs.

4.5.2. Marcação de pacotes

O byte ToS foi redefinido (ver a Figura 23). Existem seis bits utilizados para classificar os pacotes. O campo é agora chamado *Differentiated Services* (DS), com dois bits que não estão a ser utilizados (RFC-2474). Estes seis bits substituem os três bits *IP Precedences*, e é chamado *Differentiated Services Codepoint* (DSCP). Com o DSCP, em qualquer nó, podem ser suportados até 64 diferentes agregados/classes (2^6). A classificação e QoS funcionam através do DSCP num modelo DiffServ.

4.5.3. Comportamentos por salto

Ao conjunto de pacotes que atravessam num direcção, e têm o mesmo valor DSCP (chamado de codepoint) neles, é chamado *Behavior Aggregate* (BA). Pacotes de várias aplicações/origens podem pertencer ao mesmo BA. Um PHB refere-se ao comportamento da marcação de pacotes ou *packet scheduling*, filas de espera ou *queuing*, políticas, *shaping*, quando num no nó passam pacotes que pertence ao BA, e configurado por um *Service Level Agreement* (SLA) ou política. [RFC 2475]. Existem quarto PHBs disponíveis para construir uma rede Diffserv, ponto a ponto CoS e QoS.

4.5.4. PHB por defeito (definido no RFC 2474)

O PHB por defeito específica a marcação do pacote com o valor do DSCP (recomendado) de ‘000000’ irá buscar o serviço tradicional de *best-effort* do nó DS subordinado ou *DS-compliant node* (um nó que responde às necessidades do DiffServ). No caso, do pacote chegar ao *DS-compliant node* e o valor do DSCP não for mapeado a qualquer PHB, irá será mapeado ao PHB por defeito.

4.5.5. Class-Selector PHBs (definido no RFC 2474)

Para preservar a compatibilidade com o esquema IP *precedence*, Os valores DSCP da forma ‘xxx000’, onde x é 0 ou 1. Estes *codepoints* são chamados *class-selector codepoints.* Nota por defeito o *codepoint* também é chamado *class-selector codepoint* (‘000000’). O PHB associado com a *class-selector codepoint* é o *class-selector* PHB. Este PHBs têm quase o mesmo comportamento de forwarding que os nós que são implementados no *IP precedence* baseada na classificação e *forwarding.* Por exemplo, pacotes com o valor de DSCP de ‘110000’ (*IP precedence* 110) têm preferencialmente um tratamento de *forwarding* (*scheduling, queuing*, etc.) comparado aos pacotes com o valor de DSCP de ‘100000’ (*IP precedence* 100). Os PHBs garantem que os nós *DS-compliant* podem coexistir com os nós que suportam ou atentos ao *IP precendence*, com excepção dos bits DTS.

4.6. Mecanismos de QoS no IPv6

O IPv6 (*Internet Protocol version* 6), também conhecida por IPng ou IP *Next Generation*, versão 6 do protocolo IP, está a ter cada vez mais importância, e chamar a atenção de fabricantes como a Cisco, e a Nokia [RFC 2460]. Começam existir várias implementações deste protocolo, e instituições ou entidade de distribuição de endereços (*Regional Internet Registry* (RIR), já alocam endereços IPv6 [18]. A IANA, entidade responsável pela correcta atribuição dos endereços IP, atribui endereços aos RIRs, que por sua vez atribuem regionalmente os endereços IP aos respectivos ISPs [19]. Os RIRs por sua vez irão utilizar, com base nos prefixos atribuídos pela IANA, novos prefixos para os ISPs.

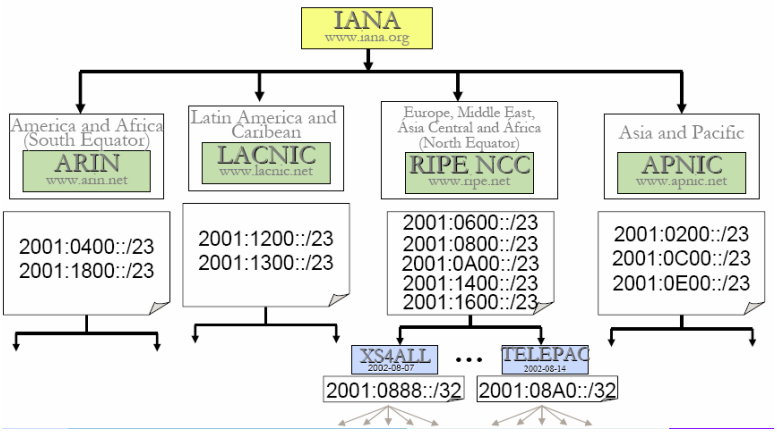


Figura 27 Atribuição dos prefixos

O objectivo principal no desenvolvimento do protocolo IPv6 era de introduzir campos de endereços (origem e destino) maiores, na Internet *Protocol* (IP).

O limite teórico de endereços IPv4 é de 4 mil milhões, mas na prática apenas cerca de 250 milhões podem ser alocados por utilizadores.3,4 × 10^38 é o número teórico de endereços associado ao IPv6 [21]. IPv6 consegue resolver o problema dos endereços, aumentando o tamanho para 128 bits, dos campos de endereço de origem e de destino. O formato básico do IPv6 está na figura seguinte.

Figura 28 Cabeçalho dos pacotes IPv6

Apesar da estrutura do pacote IPv6 ser significativamente diferente da versão 4, a funcionalidade básica do protocolo em termos de Qos não é algo inovador.

4.6.1. Campos relevantes do IPv6 para QoS

No IPv6 a QoS divide-se em dois campos: o campo *Traffic Class* e o campo *Flow Label*; o primeiro é direccionado ao catálogo de tráfego (*DiffServ*) e o segundo à geração de fluxos prioritários (*IntServ*). O campo ToS era o responsável por algum QoS que pudesse existir no IPv4, no entanto nunca foi adoptado pelos *routers*, principalmente por ser muito pequeno [19]. A qualidade de serviço é muito importante, na medida em que as tecnologias hoje em dia requerem que cada vez mais, e são chamadas tecnologias do futuro. Como multi conferência, VoIP, os fluxos de banda larga de áudio/vídeo e a mobilidade.

Existem várias formas de melhorar a qualidade de serviço, e curiosamente já foram descritas algumas delas, tais como:

- Suporte apenas de *multicast*, eliminando o *broadcast*
- Retirar o campo *checksum*
- *Extension Headers*
- Menos campos no cabeçalho IPv6. Por outro lado, alguns dos problemas do IPv4 são:
    - A fragmentação é efectuada nos *routers* com o pacote em trânsito, o que produz congestão, consumo de largura de banda e processador;
    - O ICMP tem demasiadas opções que o tornam por vezes pouco eficiente;
    - O suporte para QoS (*Quality of Service*) 12 é mínimo. O campo ToS (*Type of Service*) é demasiado curto e fixo e por isso não foi adoptado pelos *routers*. Apenas mais tarde foi adoptado o campo ToS como DS (*Differentiated Services*) , esse sim já adoptado pelos *routers*.

4.6.2. Problemas com DiffServ e IntServ

A nível de diferenciação de serviços, o campo responsável pela marcação dos pacotes (*Traffic Class*) mudou de nome, mas o funcionamento é basicamente o mesmo. Em relação à integração de serviços, o objectivo é tentar aumentar a eficiência dos fluxos prioritários, nomeadamente ao nível de anular o problema de violação de camada. A violação de camada, ilustrada na Figura 29, acontece quando algum dispositivo opera numa camada que não seja a sua, o que se traduz num decréscimo a nível de desempenho. O que se passa neste caso, é que o *router*, para identificar os fluxos, utiliza a camada de rede e a camada de transporte, apesar de estar especificado na camada de rede lhe diz respeito. A identificação de fluxos no IPv4 é conseguida através de 5 campos: *Source Address*, *Destination Address*, *Source Port*, *Destination Port* e *Protocol*. A identificação no IPv6 é feita com 3 campos: *Source Address*, *Destination Address* e *Flow Label*.

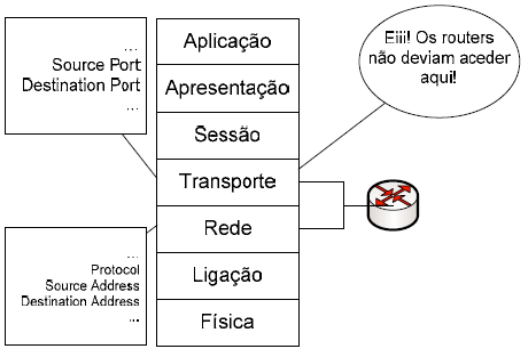


Figura 29 Violação de camada por parte dos *routers* no IPv4.

Existem outros problemas adicionais associados à violação de camada. Esses problemas verificam-se quando existe encriptação na camada de rede (caso do IPsec), ou quando existe fragmentação. No primeiro caso, o que acontece é que existindo encriptação na camada de rede, todas as camadas acima da de rede estarão também encriptadas. Desta forma não é possível ao *router* verificar os portos e, consequentemente, identificar os fluxos. Existem formas de resolver este problema, mas não são muito simples. Este problema é ilustrado na Figura 30 QoS, apesar de as siglas significarem qualidade de serviço, diz respeito a uma parte mais específica dela, onde é abordado a diferenciação de tráfego (DiffServ) e a integração de serviços (IntServ).

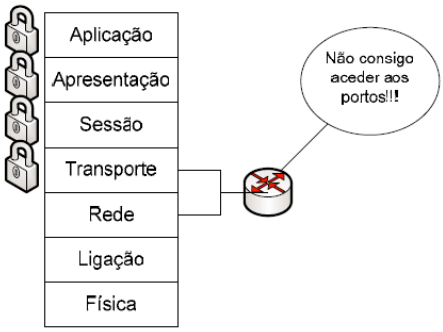


Figura 30 Encriptação ao nível da camada 3

No caso da fragmentação, o caso é bastante simples. É que nem sempre nos fragmentos de pacotes fragmentados existe o cabeçalho TCP, logo existem pacotes em que o fluxo não conseguirá ser identificado. Na Figura 31 pode ver-se o exemplo de um pacote fragmentado em dois fragmentos, em que o cabeçalho TCP só aparece num deles.[19]

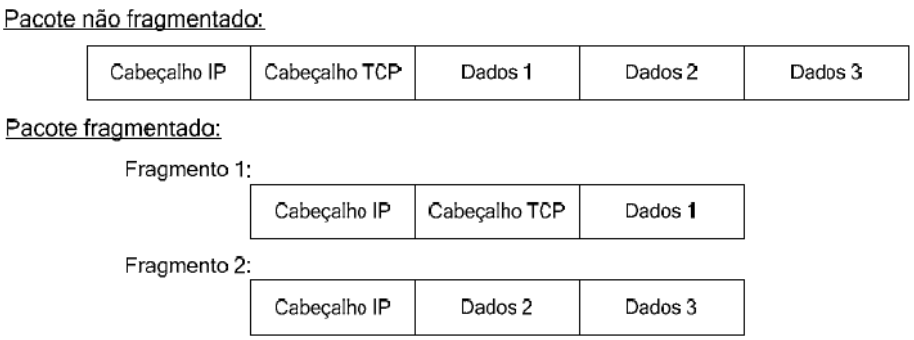


Figura 31 Exemplo de fragmentação de um pacote.

4.6.3. Os campos Traffic Class e *Flow Label*

Os representantes da Qualidade de Serviço no cabeçalho IPv6. Tudo o que seja nestes dois campos diferente de 0 indicará uma diferenciação de tráfego (campo *Traffic Class*), ou diferenciação de fluxo (campo *Flow Label*). Na captura não foi utilizada qualquer tipo de diferenciação. O uso do campo *Flow Label* é uma das maiores novidades do IPv6 e, talvez por isso, tem uma especificação própria [22].

O campo *Traffic Class* do IPv6

O campo *Traffic Class* tem uma história interessante. A versão de 1995 na especificação [RFC1883] era identificada, por um campo de 4 bits chamada *Priority Field* ou Prioridade. Este campo foi criada para permitir que o tráfego de origem, ou outro agente externo, identificar a a prioridade desejada dos pacotes (ao contrário que ter uma rede que impõe a prioridade baseada numa política configurada pelo o administrador).

O campo *Priority* foi dividido em duas gamas:

Os valores de 0 a 7 para o controlo da congestão na origem, i.e. o tráfego, como TCP, responde à perda pacotes e congestão. Os valores de 8 a 15 especificam prioridade do tráfego que não respondeu às situações de congestão, como o tráfego em tempo real enviado a uma taxa constante.

Um aspecto interessante da proposta, é o esquema de prioridades conseguir reconhecer-se

No existia uma prioridade “por defeito” para tráfego na especificação, o que significa que a origem tinha de atribuir tráfego normal com prioridade 0, para o controlo de congestão não caracterizado, e 8 para origens não caracterizados e não controlados.

Valor	Categoria
0	Uncharacterized traffic
1	Filler traffic (e.g., NNTP)
2	Unattended data transfer (e.g., SMTP)
3	Reserved
4	Attended bulk transfer (e.g., FTP, NFS)
5	Reserved
6	Interactive traffic (e.g., Telnet, X)
7	Internet control traffic (e.g., routing protocols, SNMP)

Tabela 7 Priority Values for Application Categories

O bit mais significativo do campo representava a identificação do bit de "congestion control", permitindo, que o sistema intermédio gerir a sua resposta, de acordo com o mecanismo de congestão. Apesar do campo prioridade despertar os designers de protocolos, depois reconheceu-se que era inadequada à performance e gestão de vários tipos de protocolos. Assim a versão actual da norma, torna obsoleto o RFC 1883. O campo prioridade ou *Priority* de 4 bits é substituído por um campo de 8 bits (diminuindo o tamanho do *flow label* de 24 bits a 20 bits em comprimento). De acordo com a arquitectura de serviços diferenciados, o documento não diz nada sobre as propostas para a estrutura interna o campo de 8 bits (*Traffic Flow*), nem associa performance a qualquer bit. Em vez disso, o documento aponta para os modelos de serviços diferenciados que existem no IPv4 e diz "*Detailed definitions of the syntax and semantics of all or some of the IPv6 Traffic Class bits, whether experimental or intended for eventual standardization, are to be provided in separate documents*" [RFC2460].

A vantagem desta estratégia é permitir uma utilização consistente dos campos *Type of Service* (ToS) do IPv4 e o *Traffic Class* do IPv6.

Actualmente a semântica para o campo *Traffic Class* descrito no [RFC2474] "*Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers*", e é uma norma comum para ambos os protocolos versão 4 e versão 6. Esta arquitectura de, apenas permite serviços diferenciados num direcção do fluxo de tráfego, e assim assimétrico. O desenvolvimento de uma arquitectura simétrica é um tópico para outros documentos. Está previsto que os administradores de rede irá querer gestão de congestionamento (*Priority queuing*), tolerância ao atraso (*delay sensitivity*), e *discard preference* neste campo. Quando o tráfego entra na rede, ele é classificado de acordo com um tipo de filtro de

admissão, e é atribuído um valor ao campo, para o processo de classificação. O serviço oferecido ao pacote pela a rede é determinado por este campo.

O Campo *Flow Label* do IPv6

O *Flow label* é um campo do IPv6 de 20 bits, a sua função é identificar, qualquer fluxo de tráfego, para que os nós intermédios possam utilizar esta *label* ou etiqueta para identificar os fluxos para determinados tratamentos. Por exemplo, com o *Resource Reservation Protocol* (RSVP), possivelmente o RSVPv6 ou protocolo similar. Nesta fase, a especificação, diz que o *Flow Label,* uma atribuído ainda experimental do protocolo, e poderá alterar-se devido à experiência prática, i.e. após a sua implementação. (na altura que foi escrito o documento). O nó de origem donde o fluxo parte, irá fazer a atribuição ao *Flow Label,* com a sua identificação. Na especificação IPv6, [RFC2460], necessita que os *hosts* ou *routers,* que não suportam, as funções para o campo *Flow Label,* colocam o campo a zero quando um pacote, quando estão a originar o pacote, não mudam o valor quando estão a fazer *forward* do pacote, e ignoram que estão a receber o pacote. Também está especificado que as *Flow labels* devem ser único e escolhidos aleatoriamente, numa gama hexadecimal 0x000001 a 0xFFFFF combinado com o endereço de origem. Adicionalmente, todos os pacotes pertencem ao mesmo fluxo, deve ser enviados com o mesmo endereço de origem, endereço de destino, e *Flow Label*. Actualmente, para além da sugestão mencionado na especificação do protocolo, Para além do RSVP, não é há métodos para a utilização do *Flow Label*. No entanto, existe pelo menos uma recomendação publicada que oferece uma proposta generalizada para o *Flow Label* no cabeçalho do IPv6. Todos os datagramas com o campo *Flow Label* diferente de zero têm de ter o mesmo endereço de destino, endereço de origem, *Header de Routing,* e a opção do *Hop-by-Hop.* O objectivo é que ao olhar para o *Flow Label* na tabela, o router consegue decidir para onde deve encaminhar e fazer *forward* do datagrama sem ter que olhar para o resto dos pacotes (*Headers*). [RFC1809].

4.6.4. Observações sobre o QoS no IPv6

O campo *Traffic Class* no IPv6, não oferece uma melhoria substancial em relação ao campo *IP Precedence* no cabeçalho IPv4. Actualmente, a comunidade que está a planear a arquitectura dos serviços diferenciados, querem tratar igualmente o campo *Type of Service*

do IPv4 e *Traffic Class* do IPv6 da mesma forma, assim não há uma diferença efectiva. Por enquanto, o *Flow Label* no *IPv6*, o *Flow Label* poderá ser utilizado para o protocolo RSVP (*Resource Reservation Protocol*) para *IPv6*. O *Flow Label*, poderá ser utilizado para associar a um fluxo, com uma reserva específica. A presença do *Flow Label* nos pacotes, poderá ajudar a acelerar o tráfego pelos nós intermédios, cujo caminho foi previamente estabelecido. Contudo, apesar das especulações, para além de utilizar o *Flow Label* com RSVP, ainda não se determinou os benefícios. É possível construir um *Flow Label* do pacote TCP ou UDP IPv4 através da combinação dos endereços de origem e destino com os portos de origem e destino do fluxo.

É conclusão inevitável na examinação do IPv6 é que não oferece qualquer QoS substancial sobre aquele que é conseguido no IPv4.[14]

4.6.5. QoS suportado no IOS[8] da Cisco para IPv6 [23]

A seguir são indicados os métodos QoS que a Cisco suporta no seu IOS(nas versões indicas em rodapé 8) .

Suporta:

- Classificação dos pacotes
- Filas de espera (Queuing) suportam LLQ excluindo PQ/CQ
- Traffic shaping
- WRED
- Class-based pacotes marking
- Policy-based packet marking

Não suporta:

- Compressed Real-Time Protocol (CRTP)
- Network-Based Application Recognition (NBAR)
- Committed Access Rate (CAR)
- Priority Queuing (PQ)
- Custom Queuing (CQ)

[8] Nas versões 12.2(13)T, 12.3, 12.3(2)T, 12.0(28)S , 12.4, 12.4(2)T

- RSVP
- IP RTP priority

4.6.6. Conclusões sobre QoS

Na implementação QoS, o router consegue identificar os pacotes que pertencem a fluxos individuais QoS. Assim os routers alocam a largura de banda necessária para estes pacotes. A informação para o QoS está no cabeçalho do pacote IPv6, isto significa que mesmo que corpo da mensagem for cifrada, o QoS continuará a funcionar porque as instruções QoS não ficam cifradas.

Assim é possível enviar *streaming* de áudio e vídeo pela Internet com a cifragem IPSec, por forma a garantir uma largura de banda adequada para *real-time playback.[24]*

Outra preocupação do IPv6 é oferecer um conjunto de QoS diferenciado, onde IPv4 não consegue. A justificação é do campo chamado *flow label* no cabeçalho IPv6, no entanto este campo ainda não tem fins práticos em ambientes complexos ou em grande escala. Ambos os protocolos IPv4 e IPv6 suportam um 8 bits no IPv4 é chamado ToS e no IPv6 é chamado *Traffic Class*. Ambos contêm um campo de 6 bits para serviços diferenciados (*differentiated service code points*), e ambos oferecem os mesmos campos para um classificador de pacotes para serviços integrados. Assim, a implementação de QoS, não se alterou.[27].

5. VoIP em IPv6 nativo

5.1. Introdução

Actualmente a Cisco Systems não suporta IPv6 ou VoIPv6 nas suas soluções de voz[9], apenas a nível do IOS. Foi realizado um levantamento sobre as soluções (servidores, *softphone* e *hardphones*) para VoIP IPv6 nativa (ver o anexo 8.1), estas soluções são todas *Open source.* Duma notícia encontrada na Internet[10], verifica-se que o governo japonês está apostar em VoIPv6, De acordo com esta notícia já de 2004, existem cerca de 20 000 telefones em cerca de 300 localidades implementados no Japão.

Dado que a Cisco não suporta VoIPv6 nas suas soluções de voz, foi necessário recorrer a soluções *Open Source*, para a implementação do cenário da Figura 68. Dado que estas soluções *Open Source* utilizam o protocolo SIP, será realizado uma breve descrição do protocolo SIP e implementação de um cenário simples de SIPv6 (ver capítulo 6 ou teste). Dado que este capítulo foi definido como extra âmbito, não será muito aprofundada, nem será explorado as arquitecturas SIP em ambientes IPv4/IPv6. No entanto, será realizado uma referência a possíveis soluções para a integração de outras tecnologias de telefonia na arquitectura SIP.

5.2. Arquitectura SIP

O SIP está definido em muitos RFCs, mas o principal é o [RFC 3261] que define o core do protocolo SIP. No entanto existem outros como o [RFC3665] que dá exemplos de fluxos ou mensagens trocadas do protocolo SIP. O SIP é um protocolo de sinalização que funciona a nível da camada da aplicação para iniciar, modificar e terminar sessões com um ou mais participantes. O protocolo SIP utiliza o *Session Description Protocol* (SDP), este protocolo permite negociar um conjunto de parâmetros para comunicação multimédia, por exemplo negociar qual o codec. Como o protocolo SIP não é um protocolo de transporte, é

[9] Segundo o Product Manager da Cisco do IOS da Cisco para IPv6, Patrick Grossetete <pgrosset@cisco.com>

[10] http://www.freebit.com/english/press/pr2004/20040609.html

necessário utilizar o protocolo RTP para transportar os dados. O SIP pode funcionar tanto em redes IPv4 como em IPv6.

O protocolo SIP está especificado no [RFC 3261] (tornado obseleto o RFC 2543) da IETF, o SIP define uma arquitectura distribuída para aplicações de multimédia na rede, como VoIP. H.323 é uma recomendação da ITU-T e define um sistema comunicação multimédia baseado em pacotes. O H.323 também define uma arquitectura distribuída para a criação de aplicações multimédia, incluindo VoIP. H.323 é uma extensão da norma da ITU-T, que permite videoconferência sobre LANs e outras redes comutado por circuitos, assim como vídeo sobre a Internet.

Mensagens SIP:

- Registo do utilizador no sistema
- Convida utilizadores para se juntarem à sessão
- Negocia os termos e condições da sessão
- Terminação das sessões

Filosofia

O SIP foi baseado no protocolo HTTP. O SIP também herdou o modelo cliente/servidor, em que cada participante pode ser o cliente e/ou o servidor. A reutilização de protocolos e arquitecturas IP existentes, faz parte do seu design, da eficiência e da expansibilidade da arquitectura SIP.

Características e benefícios

- Simplicidade
- Escalabilidade
- Funcionalidade distribuída
- Compatível com a Internet
- Mobilidade

Elementos

O SIP inclui os seguintes elementos *User Agent Cliente* (UAC) e *User Agent Servers* (UAS), *proxies stateless e stateful* e servidores de registo. Um *proxy* pode encaminhar a sinalização tanto como cliente como servidor. O *redirect server* ou um servidor de redireccionamentos é um UAS, que transfere as chamadas para outro servidor.

5.2.1. Universal Resource Identifier (URI)

O *Universal Resource Identifier* (URI) utiliza um sistema de nomes que descrevem a localização do recurso de forma independentemente. O esquema de nomes e métodos de acesso incluem endereços de e-mail (*mailto*), identificadores SIP, identificadores H.323 [RFC3508] ou números de telefone[11].

Sintaxe do endereço SIP

O [RFC 3986] tornou obsoletos as seguintes RFCs 2732, 2396, 1808. O RFC 2732 introduziu a sintaxe do endereço IPv6 (*Format for Literal IPv6 Addresses in URL's).*

O SIP utiliza o URI (*Uniform Resource Identifier*) como esquema de endereçamento. O URI tem a seguinte sintaxe **SIP(S):user:password@host:port;uri-parameters?headers**

O endereço e-mail poderá ser parte do SIP URI e a indicação do porto é opcional. No URI é especificado o nome completo ou *fully-qualified domain name* (FQDN) ou o endereço IPv4 ou IPv6. Contudo, ao utilizar o endereço IP seguido pelo o número do porto no URL, formato para IPv6 é um pouco diferente do IPv4.

user@ipv4-address:port; ou **user@[ipv6-address]:port**

No [RFC 3263] diz que o *host* destino é seguido do número do porto, e deve executar um DNS A ou AAAA *record lookup* para o nome do domínio.

[11] http://www.ietf.org/internet-drafts/draft-ietf-iptel-rfc2806bis-02.txt

5.2.2. Mensagens SIP

No [RFC 3261] estão definidos seis métodos:

- **REGISTER para registar a informação do contacto**;
- **INVITE, ACK, CANCEL para iniciar as sessões**
- **BYE para terminar as sessões**; e
- **OPTIONS para determinar as capacidade dos servidores**.

Existem outros RFC que definem outros métodos como o [RFC2976] para o método INFO.

e [RFC 3311] para o método UPDATE.

A seguir serão descritas as mensagens dos pedidos e das respostas mais frequentes:

Pedidos

A seguir são listados os pedidos mais comuns no protocolo SIP:

- **INVITE**→Indica ao utilizador ou ao serviço que pretende iniciar ou participar uma sessão
- **ACK**→Confirma que o cliente recebeu a resposta do pedido INVITE
- **BYE** →Indica que o utilizador pretende terminar a sessão
- **CANCEL**→Pretende cancelar o pedido
- **REGISTER**→Registar o endereço no servidor SIP

Respostas

A seguir é mostrado a lista dos 6 tipos de respostas, com exemplos:

Respostas 1xx : respostas informativas, exemplo: **108 Ringing**

Respostas 2xx : respostas sobre sucesso, exemplo: **200 OK**

Respostas 3xx :respostas de redireccionamento, exemplo: **302 Moved temporarily**

Respostas 4xx :respostas sobre falhas, exemplo: **404 Not Found**

Respostas 5xx :respostas sobre falhas no servidor, exemplo: **503 Service Unavailable**

Respostas 6xx :respostas sobre falhas globais, exemplo: **600 Busy Everywhere**

Nota: O [RFC 3261] tem a lista completa

5.3. Soluções de Integração

Nesta secção será descrito uma breve descrição de soluções para a integração de telefonia IP na arquitectura SIP. Na figura seguinte apresenta um cenário com várias tecnologias de telefonia. Como PSTN, telefonia em IPv4/IPv6, H.323, MSGP, GSM, e VoIP over WiFi, POTS. Todas estas tecnologias podem ser integradas com o protocolo SIP, por meio de servidores, ou *gateways*. No manual do utilizador da VOCAL explica estas arquitecturas e também em alguns artigos da 6net.

Nota importante: A seguir são abordadas as soluções para a integração de outras tecnologias na rede SIP, no entanto não foram implementadas ou testadas neste projecto, poderá ficar para trabalho futuro.

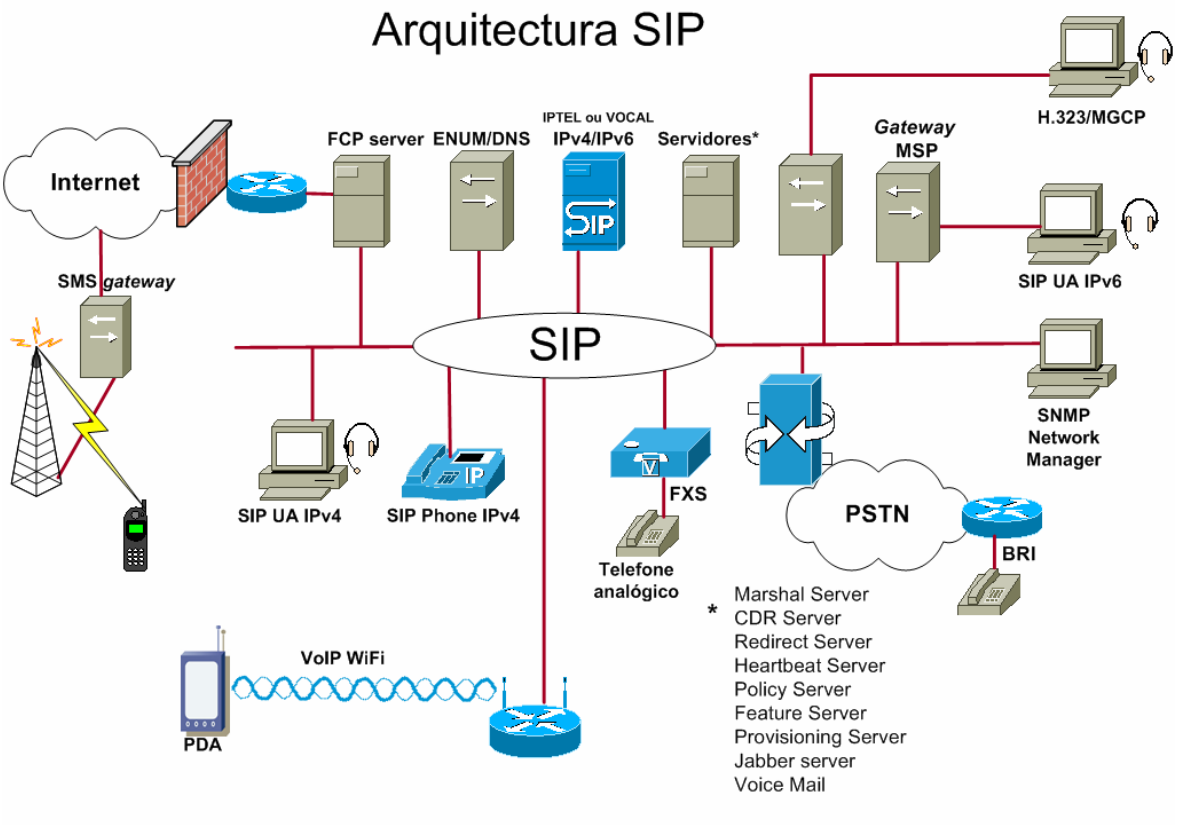


Figura 32 Arquitectura SIP em diferentes aplicações e ambientes

5.3.1. Métodos de translação IPv4/IPv6 e vice-versa no SER

Neste relatório foram explicados os conceitos sobre tradutores IPv4/IPv6, onde foram referidos o SIP *proxy*, RTP *proxy* e NAT-PT (**Nota:** A Cisco suporta NAT-PT no entanto não suportado SIP-ALG). Uma vez que o protocolo SIP funciona a nível da camada de aplicação, são necessários os *Aplication Layer Gateways* (como DNS-ALG e SIP-ALG) para fazer a tradução. O método utilizado no cenário definido foi um túnel GRE, em que os pacotes são empacotados e os campos mantêm-se, de facto é um método bastante diferente, e mais fácil de implementar.

Segundo o documento "*Interoperability between IPv4 and IPv6 SIP User Agents*"[12] Para realização da translação de IPv6/IPv4 e vice-versa é necessário um *Gateway* MSP ou Mini SIP *Proxy*. Para além de um SIP *proxy* é necessário um RTP *proxy*, para a tradução do protocolo de transporte. O servidor da SER da IPTEL também permite fazer de RTPproxy, basta substituir a linha no ficheiro de configuração na directoria **/usr/local/etc/ser/ser.cfg.**

[12] http://www.id.ethz.ch/people/allid_list/armin/SIP-IPv4-IPv6.pdf

modparam("nathelper", "rtpproxy_sock", "/var/run/rtpproxy.sock") **com**
modparam("nathelper", "rtpproxy_sock", "udp:ipv4-address:port") ou
modparam("nathelper", "rtpproxy_sock", "udp6:ipv6-address:port")

Onde o parâmetro ipv4-address ipv6-address é o endereço do computador onde está a correr o rtpproxy. O porto é opcional, por defeito é 22222.

5.3.2. Integração do telefone IP 7960 da Cisco no SER ou VOCAL

Por defeito os telefones IP da Cisco utilizam o protocolo Skinny, no entanto é possível converter o telefone IP 7960 para tornar-se um SIP Phone. Assim o telefone consegue comunicar com outros dispositivos que suportam SIP.

Turoriais que existem e provam o facto

Segundo o documento "*Converting a Cisco 7940/7960 CallManager Phone to a SIP Phone and the Reverse Proces*s" [13]

Para integrar o telefone IP da Cisco com o servidor SIP SER da IPTEL, é necessário seguir o seguinte tutorial "*Tutorial on SIP and SIP SER server Configs for Cisco 7960 with SER*"[14].

Para a integração do telefone IP da Cisco com o servidor Asterisk PBX[15], existe o tutorial "*Cisco 7960 IP Phone - SIP configuration*". Existem tutoriais na Internet para integrar do telefone IP da Cisco no servidor Asterisk por exemplo "*Configuring IP Phones for use with Asterisk*"[16]. No entanto o servidor Asterisk também suporta o protocolo Skinny ou SCCP da Cisco.

A VOCAL tem uma aplicação Web-based, chamada SIPTiger que é muito útil quando se implementa os telefones IP da Cisco7960/7940. Estes telefones e os servidores SIP proxy da Cisco dependem de um conjunto de ficheiros de configuração, o SIPTiger consegue

13 http://www.cisco.com/warp/public/788/voip/handset_to_sip.html

14 http://www.aarnet.edu.au/events/conferences/2004/apan-questnet/sipworkshop/uas/cisco7960/cisco7960.html

15 http://www.asteriskguru.org/tutorials/cisco_7960_ip_phone_configuration.html

16 http://www.asteriskguru.org/tutorials/asterisk_voip_ipphone.html

tratá-los com uma web-based *Graphical User Interface* (GUI). Depois de editados os ficheiros, os telefones SIP podem de reiniciadas remotamente para tomar as devidas alterações.

5.3.3. Gateway de SIP para POTS

Para integrar o telefone analógico na rede SIP, é possível através do comando ***session target*** no Router Cisco, este comando permite encaminhar as chamadas para o servidor SIP:

Exemplo:

```
dial-peer voice 1 voip
destination-pattern …
session target sip-server
```

Também é possível enviar pedidos para o servidor SIP e outros como DNS e ENUM, pela a ajuda no IOS do Router

```
Router(config-dial-peer)#session target
Must be of the form ^((loopback:rtp)|(dns:.*)|(ipv4:[0-9]+\.[0-9]+\.[0-9]+\.
[0-9]+(:[0-9]+)?)|(enum: ( [1-9]|1[0-5]))|(ras)|(sip-server))$
Or ^((settlement)|(settlement:[0-0]+))$
```

5.3.4. Gateway do SIP para PSTN/ISDN

No seguinte livro "*IP Telephony Cookbook*"[17] tem exemplos como integração telefone da rede PSTN para o servidor SIP utilizando um router Cisco, também é referido o *gateway* OpenISDN. A VOCAL tem o SIP Residential Gateway (SIPRG). O SIPRG é um gateway de telefonia IP que permite um SIP User Agent realizar chamadas entre uma rede Public Switched Telephone Network (PSTN) e uma rede SIP-based como a VOCAL.

[17] http://www.informatik.uni-bremen.de/~prelle/terena/cookbook/main/ch05.html#d0e6350

5.3.5. Gateway do SIP para H.323[18]

No seguinte livro "*IP Telephony Cookbook*"[19] também tem exemplos como integração telefone H.323 para o servidor SIP e para o servidor H.323 utilizando um router Cisco a funcionar de gateway.

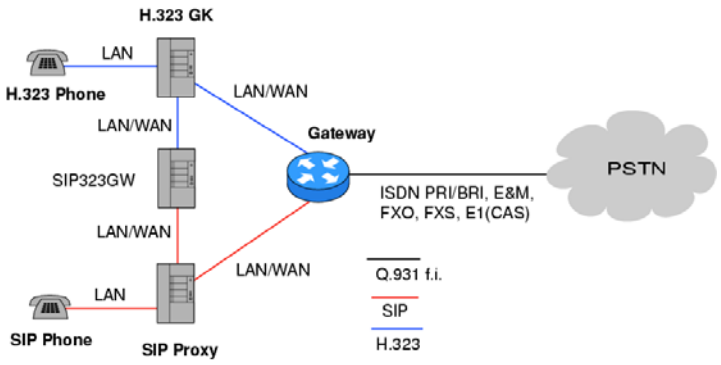


Figura 33 Implementação do cenário explicado no Cookbook (ver nota de rodapé 19**)**

5.3.6. Outros servidores e serviços

ENUM

O IETF Telephone Number Resolution working group, conhecido como ENUM e é um protocolo definido no [RFC2916], é um esquema para mapear os números de telefone E.164 (explicado no primeiro capítulo) em endereços IP utilizando o DNS da Internet. O objectivo é permitir que qualquer aplicação, incluindo as aplicações SIP, descobrir recursos associados a um único número de telefone. Os recursos DNS poderão assim oferecer um endereço SIP correspondente ao número que foi chamado.

Por exemplo:

Todos os números com o prefixo +42012345678 irão ser traduzidos para sip:123456[sufixo]@dominio.org

[18] http://www.informatik.uni-bremen.de/~prelle/terena/cookbook/main/ch05.html#sec-siph323gw

[19] http://www.informatik.uni-bremen.de/~prelle/terena/cookbook/main/ch05.html#d0e6350

No ficheiros de DNS:

```
$ORIGIN 7.6.5.4.3.2.1.0.2.4.e164.arpa

* IN NAPTR 10 100 “u” “E2U+sip” “!^\+420(.*)$!sip:$\1@dominio.org!”
```

O Servidor SER da IPTEL tem um módulo para o efeito[20]. Na Figura 34 mostra um exemplo prático do servidor ENUM.

VoiceMail

Outra aplicação da Internet é a integração de telefonia no e-mail. O VoiceMail funciona como um atendedor de chamadas, no entanto, em vez de ser gravado numa cassete, é gravado num ficheiro e enviado por e-mail. Tradicionalmente o sistema PSTN tinha o IVR (Interactive Voice Responder), quem chamava tinha de navegar pelos os menus, e digitar o número correspondente para a opção pretendida, como mostra a figura com o fluxo de dados num cenário voicemail-2-email, numa aplicação *Open source* chamada SEMS.

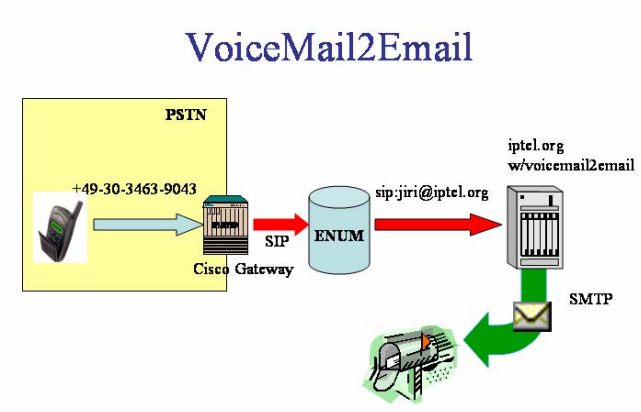


Figura 34 Exemplo da aplicação do ENUM

SEMS[21] significa SIP Express Media Server, existem outros sistemas de *voicemail*, como o OpenAM disponível no site OpenH323, no entanto são difíceis de implementar e ainda não estão preparados para um ambiente empresarial. [64]

[20] http://www.iptel.org/ser/doc/modules/html/enum.html

[21] http://sems.berlios.de/ Actualmente indisponível

Servidor FCP da IPTEL

O protocolo *Firewall Communication Protocol*, permite a interoperabilidade com os servidores SIP, para conseguir realizar chamadas através de firewalls e NATs em linux. Torna-se difícil as chamadas sairem para a Internet e passarem pelos os firewalls e NATs, assim o FCP oferece uma solução para o problema.

STUN

O STUN (**S**imple **T**raversal of **U**ser Datagram Protocol (UDP) Through **N**etwork Address Translators (NATs)), ou seja permite passar UDP (RTP funciona sobre UDP) através de NATs, e está descrito no [RFC3489]. A VOCAL também tem uma aplicação destes.

SMS gateway

O servidor SER também tem um módulo para SMS[22] gateway. Fala-se também muito sobre a implementação de SIP em redes wireless da 3ª geração ou 3GPP.

Para ver outros products ver a lista nos seguintes sites:

http://www.iptel.org/info/products/
http://developer.berlios.de/softwaremap/trove_list.php?form_cat=130&discrim=93
http://www.vovida.org/

Para saber mais sobre o SIP, aconselha-se a vista ao site http://www1.cs.columbia.edu/sip/

Em anexo (8.3.3 e 8.3.4) mostra uma lista de servidores e *gateways* de telefonia.

[22] http://www.iptel.org/ser/doc/modules/html/sms.html

6. Testes

Neste projecto foram realizados três cenários, o primeiro cenário tive como objectivo integração uma solução VoIP em IPv4 para o estudo ou análise os protocolos envolvidos na comunicação entre os equipamentos.

O segundo cenário teve como objectivo de colocar esta solução VoIPv4 sobre um *backbone* IPv6. Depois foi realizado o teste de QoS deste cenário, injectando tráfego IPv4 e IPv6.

O terceiro cenário foi realizado um cenário VoIP mas em IPv6 nativo, ou seja, todos os terminais a suportarem IPv6.

6.1. Verificação prática dos protocolos utilizados

A seguir serão descritos os protocolos envolvidos na arquitectura VoIP da Cisco, utilizando o CallManager Cisco 3.1 com o endereço IP 10.0.0.2, dois telefones IP da Cisco modelos 7960 com o número de telefone 160 e o modelo 7910 com o número de telefone 110. Os endereços IP dos telefones são atribuídos automaticamente pelo o servidor DCHP no CallManager. Foi necessário recorrer ao Ethereal e hubs de modo a apanhar os pacotes na rede. O *default gateway* tem o endereço 10.0.0.1 e está ligado a um router com uma interface FXS ligado a um telefone analógico.

Foi colocado o programa Ethereal, um analisador de protocolos nos pontos indicado na figura seguinte.

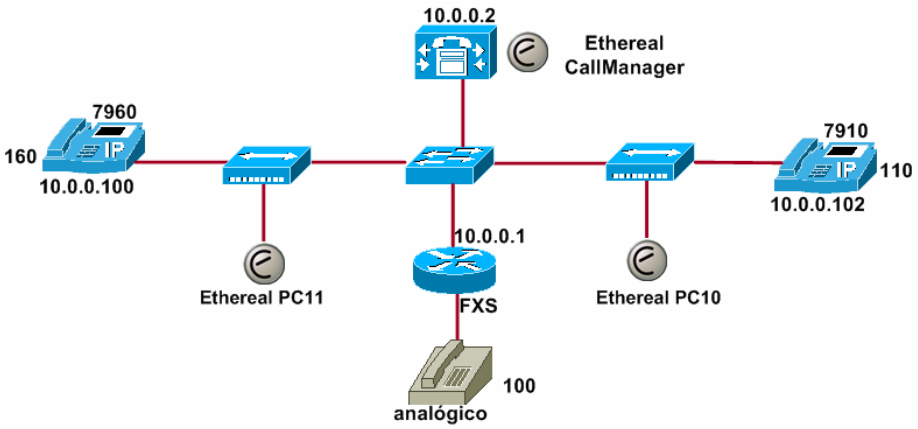


Figura 35 Cenário de teste IPv4, análise dos protocolos

No Cisco Call Manager, foram configurados os números de telefone 160, 110, e para o telefone analógico o 100 atribuído no Router.

Nota importante:

No anexo 8.4 são explicados as configurações necessárias realizar no CallManager. No anexo 8.10 estão todas as mensagens que foram capturadas e configurações realizadas no Router.

6.1.1. Mensagens no registo do telefone

Na prática verificou-se no Ethereal as mensagens de DHCP, TFTP e SKINNY.

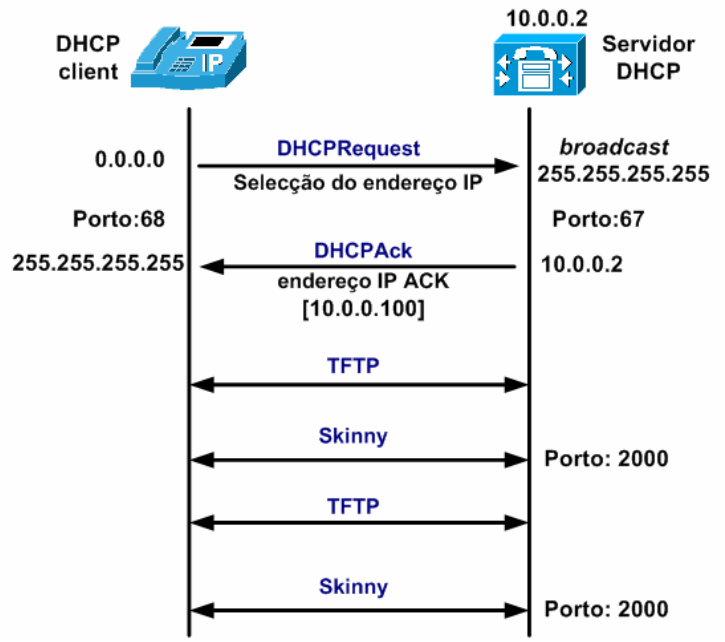


Figura 36 Resumo das mensagens no processo de registo do telefone

Para ver todas as mensagens que foram capturas no Ethereal, ver em Anexo 8.10.1. Verifica-se o protocolo *skinny* utiliza o porto 2000, para DHCP são utilizados os portos 68 e 67.

Nota: Para saber todas as mensagens com informação mais detalhada ver Figura 109 Registo do telefone IP no CCM e Figura 110 Registo do telefone IP no CCM (cont.1) em Anexo.

6.1.2. Telefonema entre telefones IP

A Mensagem stimulusMessage é quando o telefone inicia a chamada. A Mensagem onhookMessage é quando o telefone termina a chamada. Na figura seguinte mostra um resumo das mensagens capturadas no Ethereal no CallManager.

Nota: Para ver todas as mensagens do Ethereal ver em Anexo as Figura 113 captura no Cisco CallManager, Figura 114 Ethereal no PC10 e Figura 115 Ethereal no PC11.

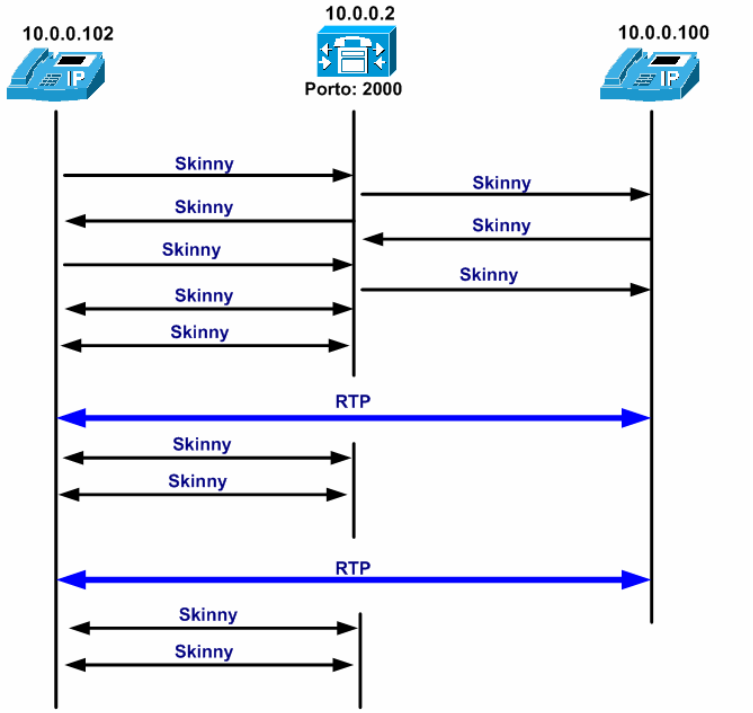


Figura 37 Mensagens trocadas na chamada entre telefones IP

Como foi referido anteriormente o protocolo RTP funciona é ponto a ponto (entre telefones), enquanto que as mensagens Skinny são entre o telefone e o CCM.

6.1.3. Chamada do telefone IP 7960 para o telefone analógico

A figura seguinte mostra o resumo das mensagens trocadas, duma chamada do telefone IP para o analógico.

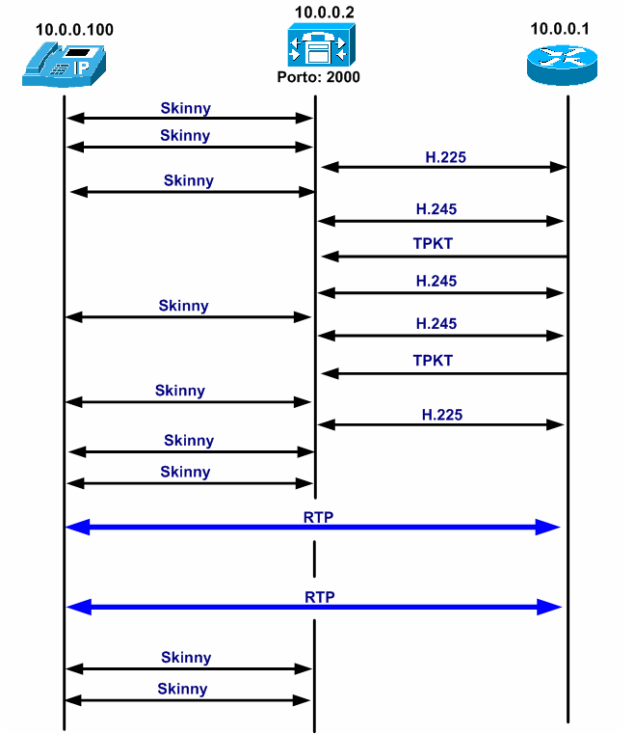


Figura 38 Chamada do telefone IP para o analógico

Verifica-se que as mensagens H.323 são trocadas entre o router e o CCM, as mensagens skinny são trocadas entre o CCM e o telefone IP. As mensagens RTP são trocadas entre o router e o telefone IP. Para uma chamada do telefone analógico para o telefone IP é igual, só por dizer que é o H.225 inicia a sessão a partir do router 10.0.0.1 ou telefone analógico. Verifica-se que as mensagens RTP utilizam o codec G.729. [RFC 1889] e a comunicação e é ponto a ponto.

6.1.4. Chamada realizado num *Cluster* de CallManagers

Agora é considerado um cenário com dois CCM, de acordo com a figura seguinte.

Nota: Dado que a escola apenas tem um CCM, esta solução não foi testada.

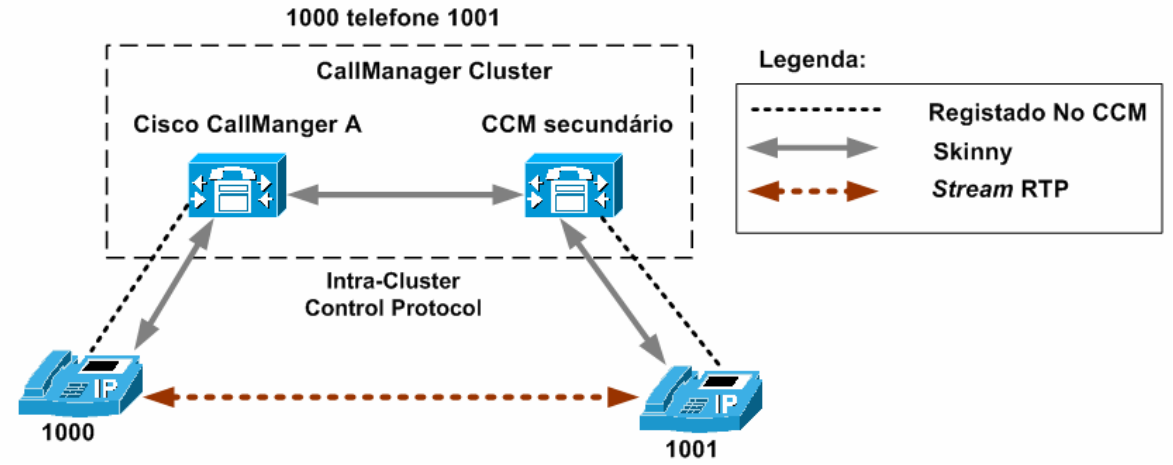


Figura 39 Comunicação entre os equipamentos VoIP

O telefone IP da Cisco (*directory number* ou nº de telefone 1000) quer chamar outro telefone IP da Cisco (*directory number* ou nº de telefone 1001) no mesmo *cluster*. O *cluster* é um grupo de Cisco CallManager tendo em comum a base de dados SQL e muitos base de dados *subscriber* SQL. Na topologia da Figura 39 o CCM1 é o Publisher e o CCM2 é o *subscriber*. Os dois telefones IP (1000 e 1001) estão registados no CCM1 e CCM2, respectivamente. O fluxo da chamada é mostrado no diagrama da Figura 121**.**

Depois para iniciar uma chamada é necessário que o telefone comunique ao CCM através do protocolo Skinny. O CCM depois estabelece a ligação entre os telefones, e a partir daí os telefones comunicam ponto a ponto entre eles, com o protocolo RTP, para trocar dados de áudio. Depois para terminar a ligação o telefone fala novamente ao CCM a dizer que quer terminar a ligação. Os dois CCM comunicam entre eles, através do protocolo *Intra-Cluster Control Protocol ICCP*.

Em Anexo na Figura 121 mostra um resumo das mensagens trocadas entre duas estações Skinny (Station) ou telefone IP Cisco. O telefone IP inicia a ligação ao CCM e depois o CCM analisa o número digitado antes de abrir a sessão para o telefone IP destino. O seguinte diagrama indica, indica as mensagens de uma forma simples.[11]

6.2. Implementação prática do túnel “4 to 6” GRE

Objectivo deste cenário foi a implementação do túnel GRE, e verificação do processo de encapsulamento GRE, e o *overhead* introduzido.

De todos os mecanismos de transição explicados anteriormente, muitos apenas têm suporte em soluções *Open source*. A solução escolhida foi implementação de um túnel GRE “4to6”, o IOS da Cisco suporta GRE ipv4 over ipv6, nas seguintes versões de IOS 12.3(7)T, 12.4 ou 12.4(2)T.

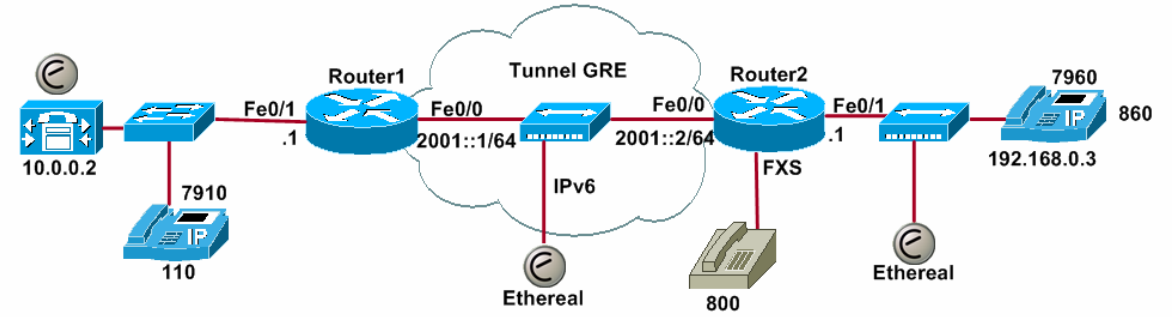


Figura 40 Implementação prática de um túnel GRE

Para verificar o processo de encapsulamento GRE, foi colocado o *Ethereal* para apanhar os pacotes entre os *routers*, ou seja, a meio do túnel, no CCM e entre o telefone IP 7960 e o Router2. Foram utilizados *hubs* para o ligar um computador com o Ethereal. Todas as configurações realizadas estão em Anexo 8.11

6.2.1. *Overhead* introduzido pelo o túnel GRE

Verifica-se que no processo de encapsulamento GRE o valor do campo *Differentiated Services* é copiado para o campo *Traffic Class.* Como é demonstrado nas figuras seguintes.

Na figura seguinte é mostrado o pacote RTP caputado fora do túnel ou seja em IPv4.

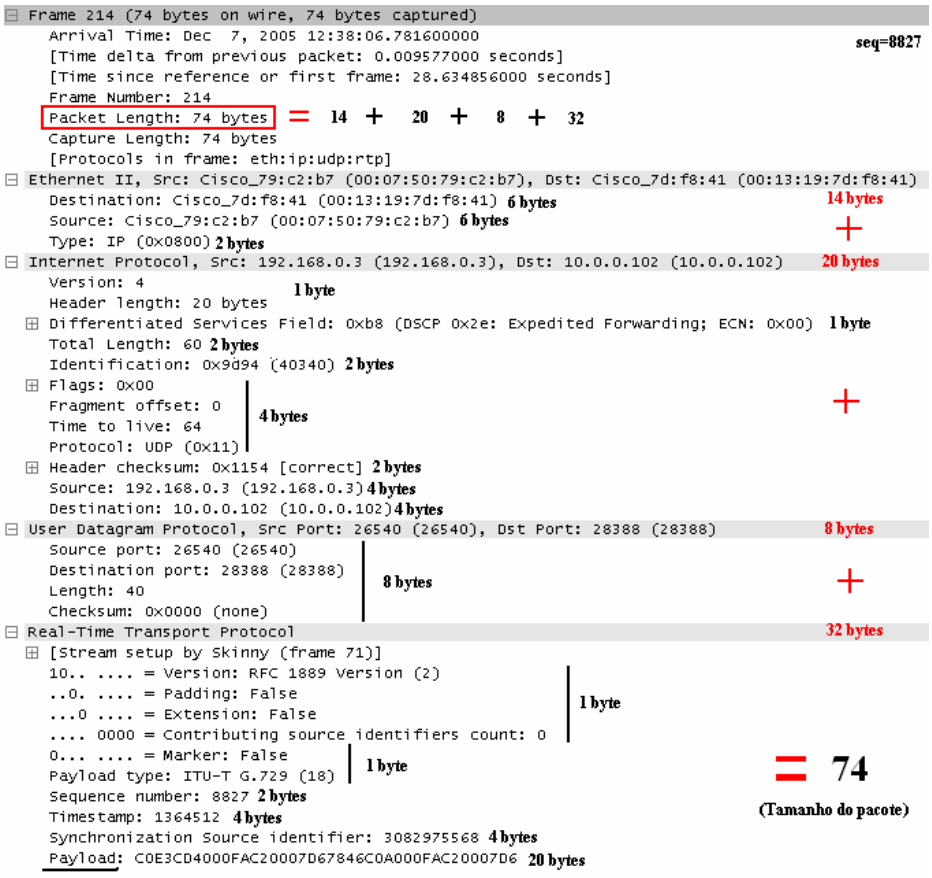


Figura 41 Pacote RTP em IPv4 fora do túnel GRE

Para o cabeçalho da Ethernet ocupa 14 bytes, para o cabeçalho do IPv4 ocupa 20 bytes para o cabeçalho UDP ocupa 8 bytes e finalmente para o cabeçalho RTP ocupa 32 bytes. O que dá uma frame com o tamanho de 74 bytes.

A figura seguinte mostra o pacote RTP capturado dentro do túnel, ou seja encapsulado dentro do IPv6. Verifica-se que IPv4 está dentro do GRE e o GRE está dentro da *extension header* do IPv6.

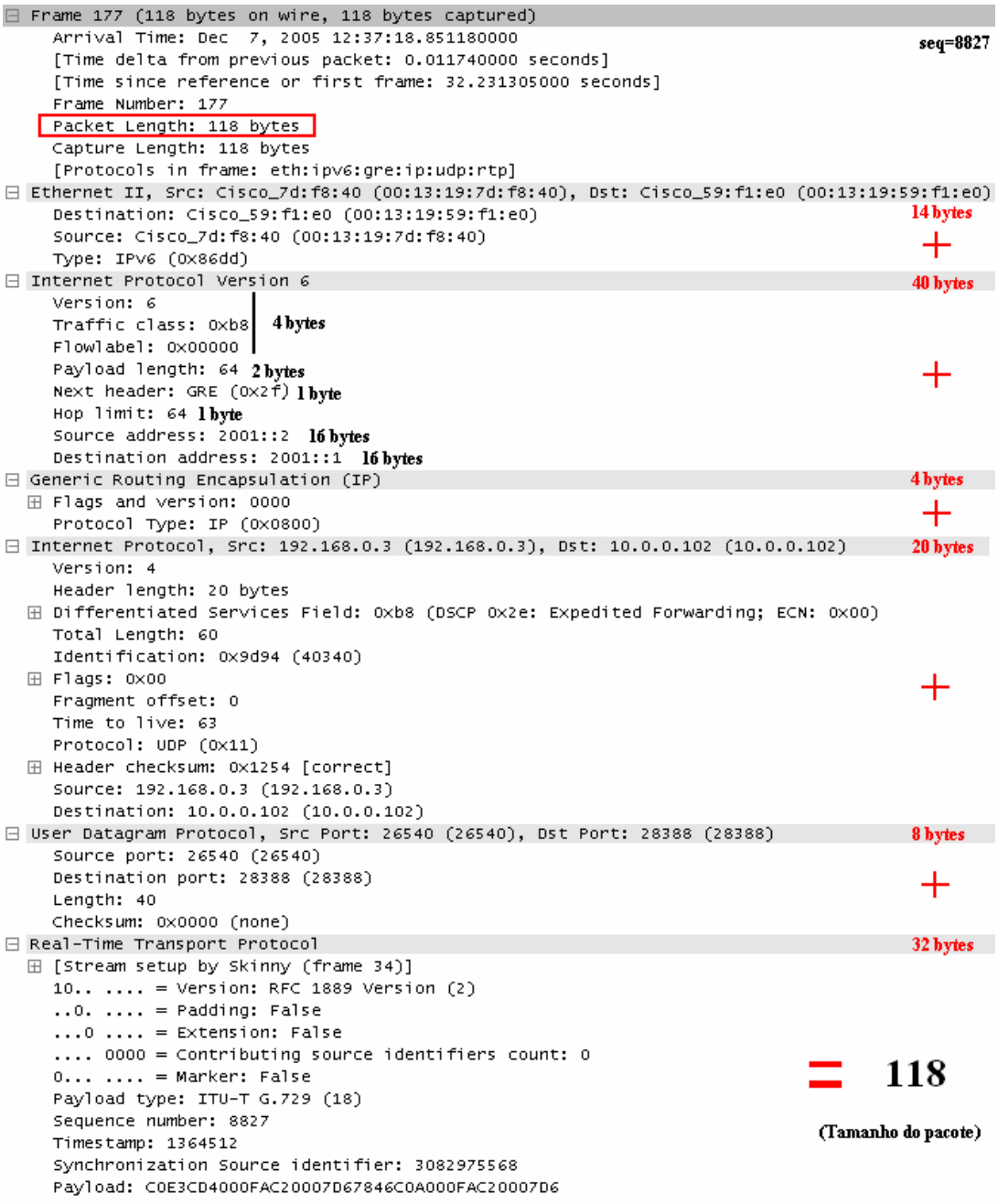


Figura 42 Pacote RTP dentro do túnel GRE portanto IPv6

Agora devido ao processo de encapsulamento existem mais dois cabeçalhos do IPv6 e GRE. O cabeçalho do IPv6 necessita neste caso de 40 bytes e do protocolo GRE necessita de 4 bytes. Portanto o *overhead* é sempre o tamanho da *frame* em IPv4, adicionado 44 bytes devido aos cabeçalhos do protocolo IPv6 (40 bytes, dependendo do número de *extension headers*) e o cabeçalho do protocolo GRE (4 bytes).

6.3. Testes de QoS no cenário

Para testar QoS no cenário definido (ver Figura 2), foi implementado o cenário da figura seguinte. O Chariot foi utilizado para gerar as *streams* de voz, no entanto, o Chariot não suporta IPv6 na versão 4.1. Assim foi necessário recorrer ao MGEN para gerar o tráfego IPv6. Todo o tráfego que é IPv4 (*streams* de voz) irá pelo o tunnel 0, e o tráfego IPv6 irá pela ligação serial. O tráfego IPv6 não irá sofrer o processo de encapsulamento. A interface tunnel é uma interface lógica, que utiliza a ligação serial. Assim as políticas são atribuídas à interface serial. Verificou-se na prática que não é possível atribuir QoS na interface tunnel.

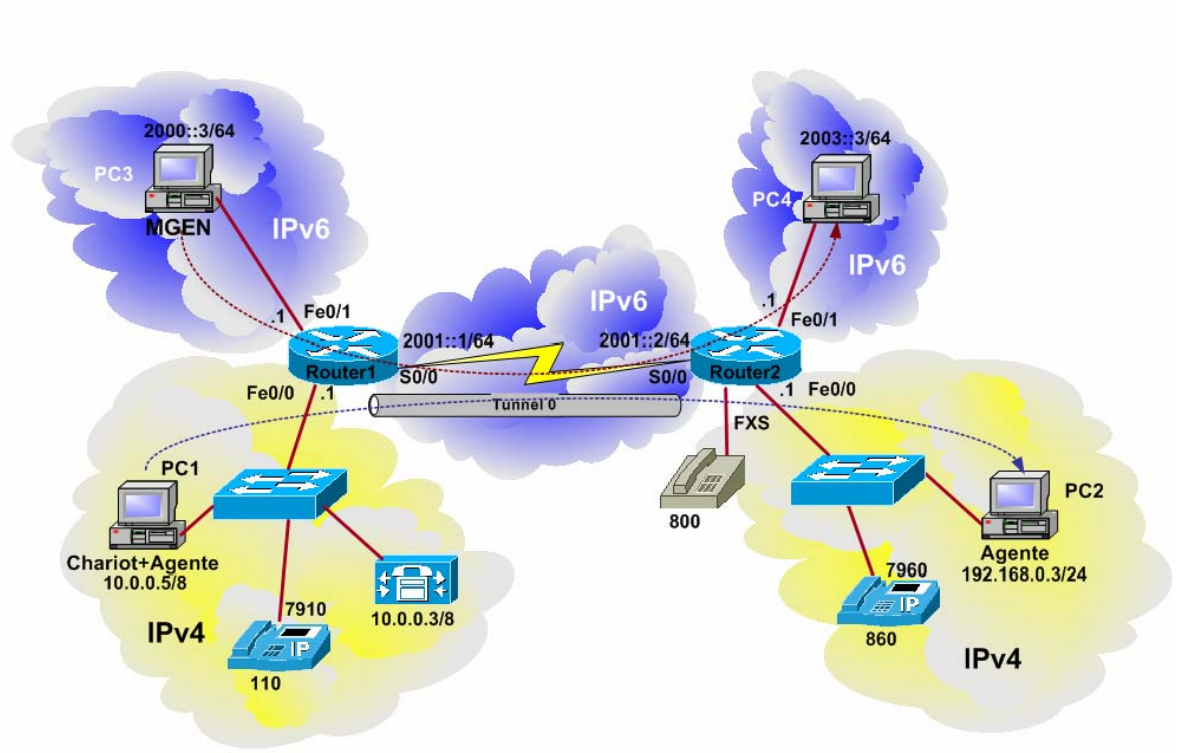


Figura 43 Cenário de teste completo

Todo o tráfego gerado irá no sentido do Router1 para o Router2, assim o Router1 irá ser o *bottleneck* e irá descartar os pacotes em caso de congestionamento.

Nota: Todas as configurações do Chariot e MGEN estão no Anexo 8.12, as configurações dos Routers estão no anexo 8.14

Para a implementação de QoS no cenário definido, foi necessário ter em conta alguns conceitos relacionados com cálculos de largura de banda, *throughput*, *codecs* e outros. A seguir são explicados alguns destes conceitos, que são necessários ter em conta para testar a qualidade de serviço QoS em VoIP. A ferramenta utilizada para gerar e analisar o tráfego, foi o *Chariot da NetIQ versão 4.1*. Devido ao facto do *Chariot* não suportar IPv6, foi necessário recorrer também ao MGEN para gerar tráfego IPv6 apenas. Outra aplicação foi o *Ethereal* utilizado durante todo o projecto.

6.3.1. Prioritização do tráfego

Em ligações com débitos reduzidos é necessário recorrer a técnicas de fragmentação de pacotes de vídeo (pacotes grandes), de modo a que estes não interfiram com os pacotes de voz, que têm um tamanho substancialmente mais pequeno. Mais concretamente, se se enviassem os pacotes de vídeo por ligações de baixo débito, mesmo recorrendo a mecanismos de prioritarização de tráfego, não se iriam obter resultados aceitáveis dado que, os pacotes de voz necessitavam que todo um pacote de vídeo fosse enviado. Se se dividir um pacote grande em pacotes menores, o problema anterior fica resolvido, uma vez que já não é necessário enviar um pacote completo mas sim, parte deste.

A Cisco recomenda em ligações WAN a técnica de *low-latency queuing* (LLQ). Este método suporta até 64 classes de tráfego e permite, por exemplo, criar uma fila prioritária (ou *priority queuing*) para o tráfego de vídeo e outras para tráfego menos sensível ao atraso (*best-effort)*.

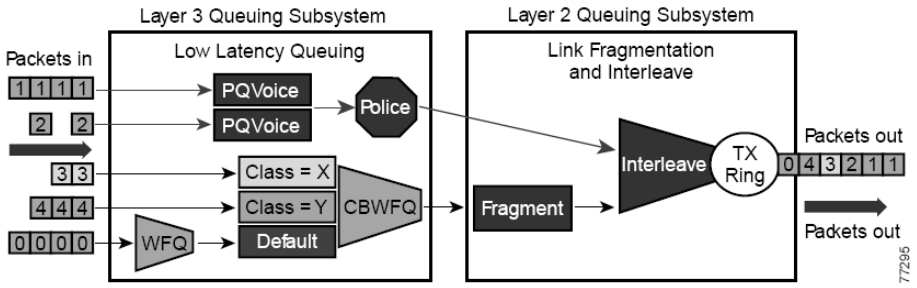


Figura 44 Fila para VoIP sobre uma WAN (Router)

Para se usar LLQ, a Cisco recomenda que a voz seja colocada numa fila de prioridade com o valor do *differentiated services code point* (DSCP) a 46, ou um *per-hop behavior* (PHB)

com o valor de EF. Para tráfego de vídeo-conferência recomenda o valor de 34 para DSCP e de é AF41 para PHB.

Dado que os pacotes de vídeo são bastante grandes, a Cisco recomenda que estes devem ser colocados num fila prioritária, apenas em ligações WAN superiores a 768 Kbps. Para ligações inferiores, o tráfego de vídeo-conferência deve ser colocado numa fila *class-based weighted fair queue* (CBWFQ) separada.

Nota: O tráfego de vídeo unidireccional, como video-on-demand, deve utilizar sempre o CBWFQ porque este tipo de tráfego tem mais tolerância que vídeo bi-direccional.

Com as ligações WAN congestionadas, é possível que os protocolos de sinalização não consigam passar a ligação. Desta forma, passa a ser impossível o estabelecimento da ligação dos telefones IP. Assim, os protocolos de controlo da voz, como o H.323, MGCP, e *Skinny Client Control Protocol* (SCCP), necessitam da sua fila *class-based weighted fair queue* CBWFQ. O critério de entrada é o valor do DSCP de 24 ou valor de PHB a CS3.

6.3.2. Classificação do tráfego

No cenário de VoIP implementado com o túnel “4to6” GRE, foram capturados os pacotes dentro do túnel, onde foi realizado a análise. Utilizando ligações *ethernet* de modo a ter a colocação de um *sniffer* a meio do túnel.

Verifica-se que os pacotes que passam dentro do túnel têm uma *frame* com o tamanho de 118 bytes enquanto que a *frame* que está nos extremos do túnel têm uma *frame* com o tamanho de 74 bytes. Como é possível verificar nas figuras (Figura 41 e Figura 42).

Quanto à qualidade de serviço QoS, o valor do campo *Differentiated Services* do IPv4 para QoS tem o valor de 0x2e (HEX), 46 (DEC) ou 101110 (BIN) para o DSCP para RTP. Assim os pacotes já vêm marcados pelos os telefones ou seja para *call routing*. Não é o *Call Manager* uma vez que as mensagens RTP são ponto a ponto. Para sinalização é utilizado outro valor o 18(HEX), 26(DEC) e 11010 (BIN). Como mostra a figura seguinte:

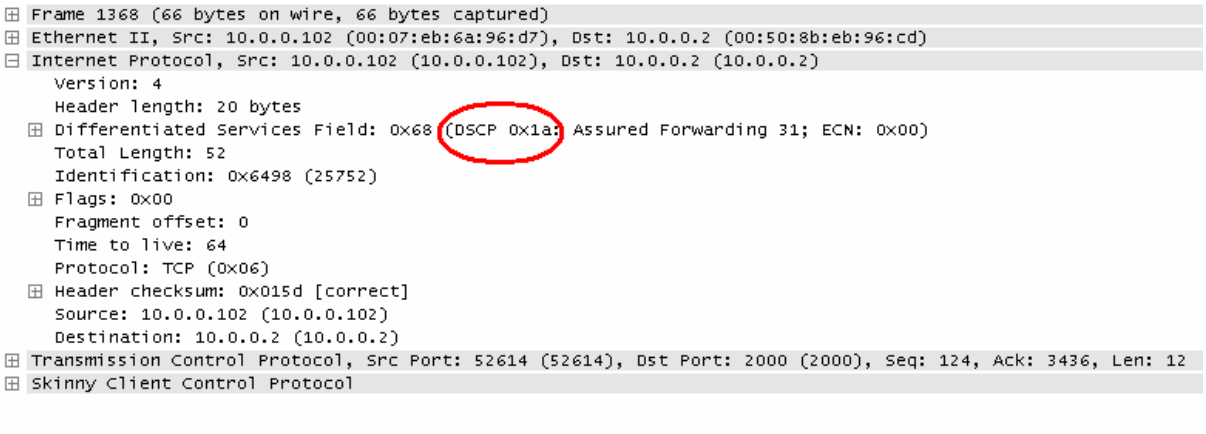


Figura 45 O valor de DSCP tem o valor de 1A para a sinalização *skinny*

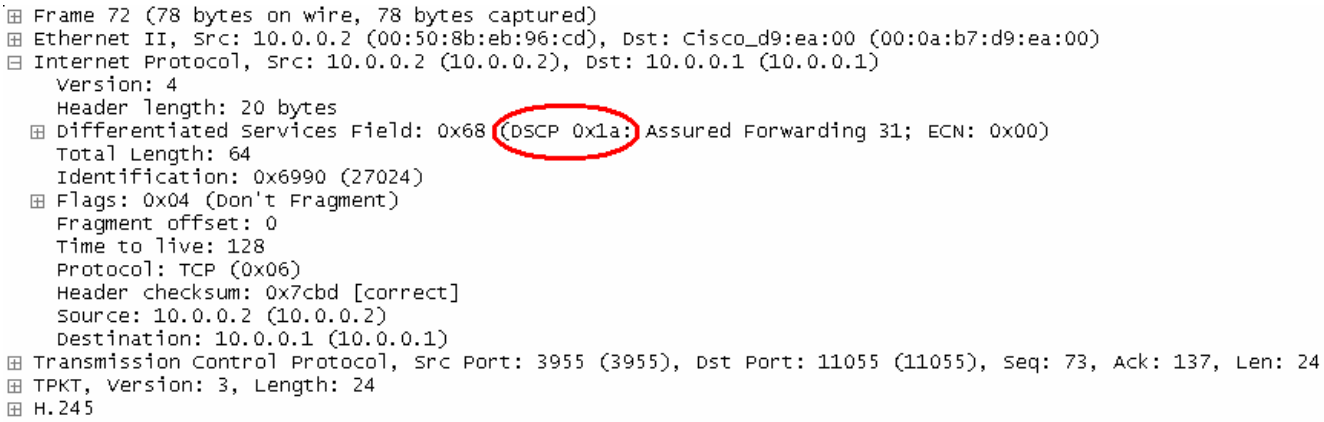


Figura 46 O valor de DSCP tem o valor de 1A para a sinalização TPKT, H.245

Verifica-se que a Cisco preocupa-se em classificar e marcar os pacotes, os telefones IP da Cisco marcam os pacotes para a sinalização de voz e para as *streams* RTP. Sendo assim na implementação não é necessário configurar access-lists ACLs para classificar o tráfego.

Por exemplo:

Capturar os pacotes através de ACLs:

Uma vez que os pacotes não estão marcados é necessário utilizar ACL (Access control lists) que servem para ver qual o protocolo que entra na interface do router, através do porto utilizado. No caso do pacote ser de voz irá para a classe EF.

```
class-map match-all EF
   match access-group 101
class-map match-all  CS3.
   match access-group 102
access-list 101 permit udp any any range 16384 16383[23]
access-list 102 permit tcp any any eq 2000
```

Marcar os pacotes:

Depois de selecciondo o pacote ele irá ser marcada através do comando SET.

```
policy-map SETDSCP
class EF
  set ip dscp 46
policy-map CS3.
  set ip dscp 24
```

Atribuir à interface:

É necessário indicar a interface de entrada, pela qual os pacotes irão ser marcados.

```
interface fastEthernet0/0
ip address 10.0.0.1 255.0.0.0
service-policy input SETDSCP
```

Tipo de tráfego	L2 CoS	L3 IP Precedende	L3 DSCP	L3 PHB
Voice RTP	5	5	46	EF
Voice control signalling	3	3	24	CS3
Video conferência	4	4	34	AF41
Dados	0,1,2	0,1,2	10 a 22	BE a AF23

Tabela 8 Traffic Classification Guidelines for Various Types of Network Traffic[10]

Nota: A marcação do DSCP/PHB para a sinalização e controlo de voz seja alterado de 26/AF31 para 24/CS3. A Cisco está a alterar este valor nos novos produtos, no entanto para o caso destes telefones IP utilizados ainda utilizam o valor 26. Assim recomenda-se ter ambos os valores na configuração dos parâmetros de QoS.

6.3.3. Controlo de admissão

Dado que no comando *priority* (para a configuração do LLQ) é especificado a largura para as N *streams. Por isso* no caso de existir congestionamento irá causar perdas porque não é possível terr N+1 *streams.* Assim para que não hajam mais do que N chamadas deve-se

[23] http://www.cisco.com/en/US/tech/tk652/tk698/technologies_tech_note09186a0080094660.shtml

configurar no CCM, *Call Admission Control*. Assim no CCM deve ser configurado no Location configuration a largura de banda alocada para as N *streams*, a indicar é de acordo com a Tabela 4.

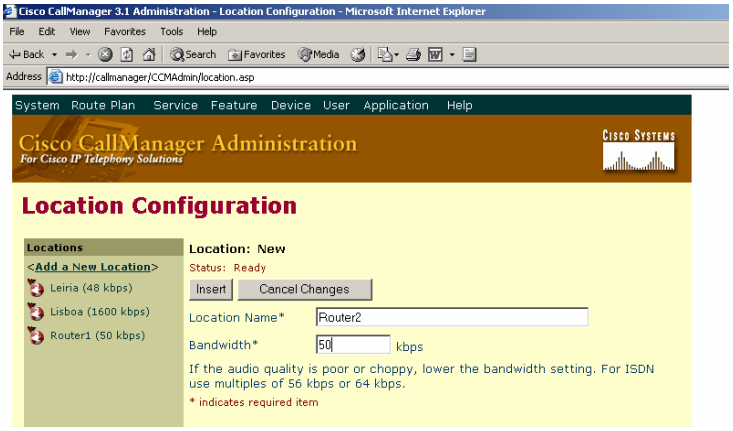


Figura 47 Configuração da admissão ao controlo no CCM

Nota: No caso de indicar o valor zero de largura de banda, está a alocar largura de banda iliminitada e permitir um número ilimitado de chamadas activas na ligação WAN para esse sítio ou location.

6.3.4. Testes Iniciais de QoS

Antes foi testado o seguinte cenário para aprender a lidar com o Chariot e os conceitos de QoS.

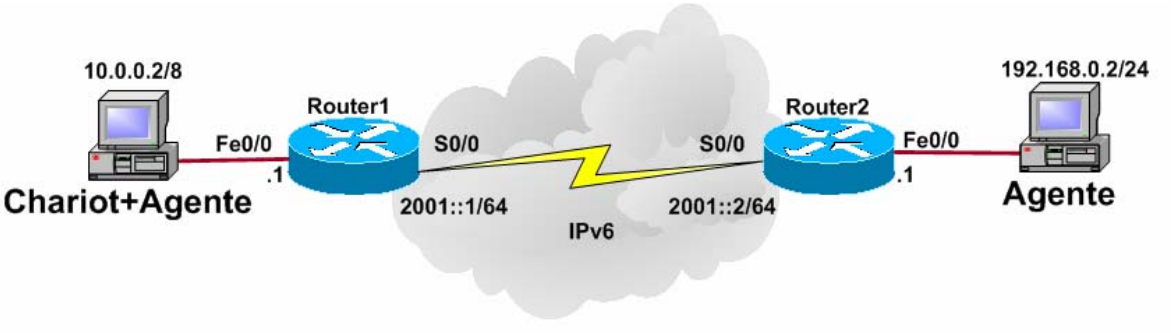


Figura 48 Cenário de teste inicial

Para calcular o *throughput* na rede Ethernet é necessário o seguinte calculo:

Temos 74 bytes (tamanho da *frame*)*8 bits por byte * 50 PPS =**29.6kbps**

Pensou-se que eram possíveis 3 ligações de voz:

114(throughput da ligação)/29.6=**3.851** (3 ligações sem perdas) sem tráfego adicional

No entanto devido ao *overhead* apenas são possíveis 2 ligações:

118 bytes (tamanho da *frame*)*8 bits por byte *50 PPS=**47.2 kbps** **114(throughput da ligação)/47**=2.4 **(2 ligações sem perdas) sem tráfego adicional.**

Inicialmente foi esquecida o *overhead* e portanto os resultados não estavam de acordo com as configurações. No router foram reservados 80kbps para voz, no Chariot foram gerados 3 *streams*. A seguir é mostrado um gráfico das perdas Pair1 (9,081%) e 2 (7,932%) são de voz e o 3(2,278%) voz marcados.

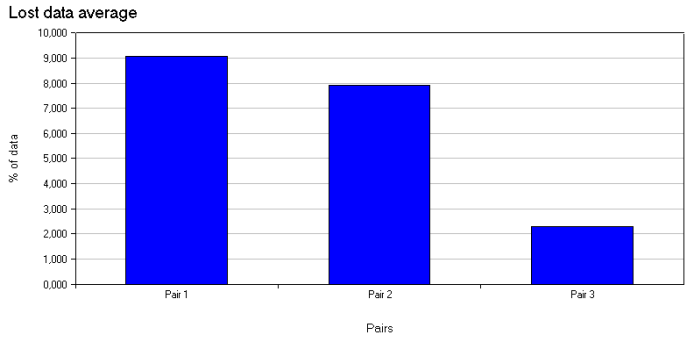


Figura 49 Gráfico das perdas

Verificou-se que havia perdas, deve facto 80 kbps não chegava para as três *streams* de voz (29,6 kbps porque era de 47,2kbps), ou seja existia um *bottleneck* na fila configurada para QoS. Na figura seguinte mostra as *streams* com menores perdas tem maior *throughput.*

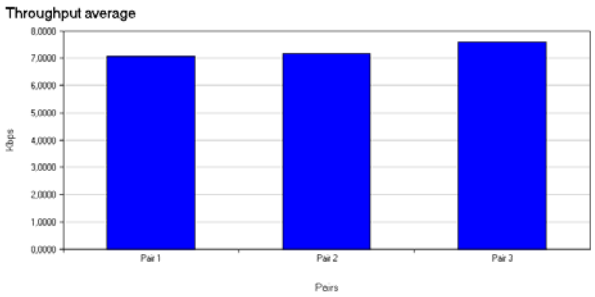


Figura 50 Quando existe congestionamento o *throughput* diminui (7,074, 7,166,7,606)

6.3.5. Testes finais com tráfego IPv4 e IPv6 no cenário

Neste cenário é gerado tráfego IPv4 e IPv6. Os dois routers têm as mesmas configurações em termos de QoS. Foram criadas duas classes uma para transporte (RTP) e outra para sinalização (skinny). No caso de sinalização é verificada se o pacote tem o valor para o DSCP igual a cs3 ou af31.

Por exemplo:

Criou-se uma classe para o protocolo de sinalização:

```
class-map match-all skinny
 match dscp cs3 af31
 match protocol ipv6
```

Criou-se uma classe para o protocolo de transporte:

```
class-map match-all voz
 match  dscp ef
!verifica-se que o pacote de voz já está encapsulado em IPv6 no match
 match protocol ipv6
```

Cria-se um policy-map e atribui-se uma largura de banda de **190kbp**s para a stream de transporte, e **8kbps** para a classes de sinalização (configuração dos parâmetros de QoS):

```
policy-map pm
 class voz
!largura de banda em kbps para as streams de voz
  priority 190
 class skinny
!largura de banda em kbps para o protocolo de sinalização marcado com o valor de
!DSCP=24 ou 26
  bandwidth 8
```

Nota: É importante reservar largura de banda para o protocolo de sinalização, porque quando há congestionamento existem problemas no estabelecimento da chamada. De facto foi verificado na prática (por exemplo estabelecia-se uma chamada no entanto ao atender a chamada o telefone continuava a tocar).

À interface de saída atribui-se esta política de QoS:

```
interface Serial0/0
!atribui-se a politica à interface serial
service-policy output pm
```

Para verificar se a política foi atribuída é executado o comando sh policy-map:

```
Router1#sh policy-map
  Policy Map pm
    Class voz
      Strict Priority
      Bandwidth 190 (kbps) Burst 4750 (Bytes)
```

Class sinalizacao
Bandwidth **8 (kbps)** Max Threshold 64 (packets)

Explicação dos parâmetros

O comando **bandwidth** [46], especifica a largura mínima garantida para a classe em períodos de congestionamento. A largura de banda garantida pode ser especificado em kbps ou em numa percentagem da largura disponível.

O comando **priority** [46] é utilizado para o priority queuing ou LLQ low latency queuing também chamado **CBWFQ (class-based WFQ**), strict queuing é utilizado para tráfego em tempo real como VoIP.

LLQ Low Latency Queueing (antigamente referido como PQCBWFQ) [46], [47] e [48]

Na utilização do comando ***priority*** é necessário ter em conta [49]:

- Quando não há congestionamento, a *classe priority* permite exceder a largura de banda reservada. Quando o equipamento está congestionado, a *class priority* irá descartar pacotes acima da largura de banda reservada.
- Deve-se ter em conta a largura de banda que é ocupado com o encapsulamento da camada 2. No entanto deve-se configurar a largura de banda de modo a deixar lugar para o possível *jitter* introduzido pelos os routers.
- O comando *priority* pode ser configurado em várias classes, mas deve ser apenas para tráfego como voz, com tráfego do tipo CBR (constante bit rate). Se o tráfego não for CBR, é necessário aumentar o parâmetro *bandwidth* para absorver os *data bursts*.

O LLQ cria uma fila prioritária (*strict priority queue*) que está acima de todas as filas. Esta fila é a primeira a ser esvaziada antes de qualquer fila. Apenas quando esta fila está vazia é que as outras são atendidas. É muito importante saber alocar largura para o *priority*, porque se configurar muito o tráfego como prioritário, então não haverá prioridade nenhuma. Por exemplo se todas as pessoas voarem em primeira classe, será que se pode chamar primeira classe? É aconselhado que sejam apenas para tráfego de voz. Os pacotes de voz são pequenos e são transmitidos a um bit rate constante (CBR). O administrador deve saber

exactamente o número de *streams* de voz, para saber alocar a largura de banda necessária. Também deve ser configurado o controlo à admissão, porque o administrador reservou 10 *streams* de voz (5 chamadas), então apenas devem ser permitidas 5 chamadas.

```
Router1#sh queue serial 0/0
 Input queue: 0/75/0/0 (size/max/drops/flushes); Total output drops: 0
 Queueing strategy: Class-based queueing
 Output queue: 0/1000/64/0 (size/max total/threshold/drops)
    Conversations  0/4/256 (active/max active/max total)
    Reserved Conversations 1/1 (allocated/max allocated)
    Available Bandwidth 960 kilobits/sec
```

Figura 51 Método utilizado Class-based queuing

Valor do DSCP que podem ser configurados no router ver anexo 8.12.5

Para a determinação da largura de banda necessário de configurar no comando ***priority***, é necessário realizar a seguinte cálculo:

Para cada *stream* de VoIP necessita de:

118(tamanho da *frame* em bytes) *50pps*8kbps=47.2kbps

Para *Skinny* a largura de banda é calculada da seguinte forma segundo a Cisco:

Largura de banda (bps) = 265 * (número de telefones IP)

Nota: No *router* não dá para especificar valores inferiores a 8kbps, então será sempre utilizado o valor de 8kbps. Como é mostrado na ajuda do router.

```
Router2(config-pmap-c)#  bandwidth ?
  <8-2000000>  Kilo Bits per second
  percent      % of total Bandwidth
  remaining    % of the remaining bandwidth
```

Streams VoIP IPv4 geradas no Chariot

Foram sempre criadas *streams* no sentido do Router1 para o Router2.

Group/ Pair	Endpoint 1	Endpoint 2	Network Protocol	Service Quality	Script Name
Pair 1	10.0.0.5	192.168.0.3	RTP	Cisco	NetMtga.scr

Tabela 9 24 *stream* criada no Chariot

Todos os testes efectuados foram realizados sobre um de *link* de 256 kbps. A Cisco também recomenda deixar largura de banda para o router introduzir *jitter*. Foram realizados quatro testes em cada teste foi criado mais uma *stream*.

Stream IPv6 gerada no MGEN

Para cada teste alterou-se a stream MGEN de 64, 128,256, e para 512kbps. Assim o tráfego BE era criado apenas pelo o MGEN.

1º Teste

Foi criada 1 fluxo VoIP, e atribuiu-se o parâmetro do **Priority** com o valor de **50 kbps** (47.2)

Perdas De voz	*Jitter* médio em ms	MGEN (*thoughput* em kbps)	*Drop* rate BE Router1 (bps)
0	5,040	64	0
0	8,400	128	0
0	14.730	256	**59000**
0	14,970	512	204000

Tabela 10 1º teste 1 fluxo de voz gerado no Chariot

Os valores são os esperados por exemplo para a *stream* de com o tráfego BE ou o MGEN a gerar 256kbps:

Largura de banda disponível BE (sem tráfego BE gerado) = Largura de banda – *priority*+LB para sinalização = 256-58 =198

LB efectiva = largura de banda disponível - Tráfego do MGEN =198-128=**70kbps** (valor positivo por isso não há perdas e ainda estão disponíveis 70kbps)

LB efectiva = largura de banda disponível - Tráfego do MGEN =198-256=**58kbps (perdas,** drop rate**)**

Gráficos do jitter em cada stream

O Chariot gera os gráficos sobre jitter e não do atraso. Nas figuras seguintes são apresentados os resultados, em cada experiência ou teste.

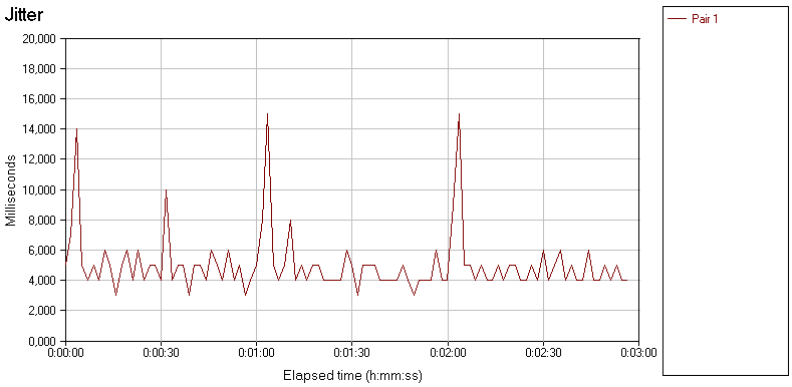


Figura 52 *Jitter* dos fluxos VoIP (4 a 6ms) na presença de tráfego BE de 64kbps

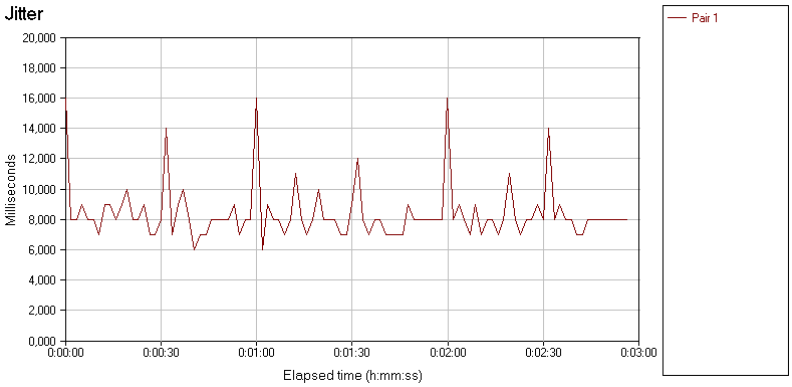


Figura 53 *Jitter* dos fluxos VoIP(8 a 10ms) na presença de tráfego BE de 128kbps

Nos dois primeiros gráficos (ver Figura 52 e Figura 53) é possível verificar que o jitter está com valores abaixos, uma vez que não existe congestionamento. Pelo os gráficos seguinte já existe congestionamento o que implica um aumento do jitter, No entanto nunca há perdas de voz.

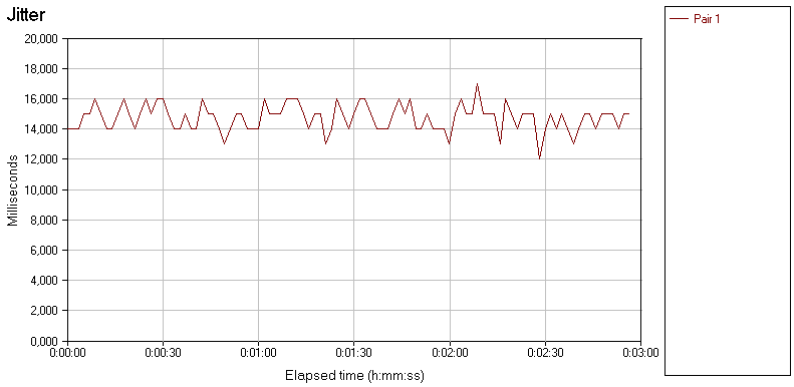


Figura 54 *Jitter* dos fluxos VoIP (14 a 16ms) na presença de tráfego BE de 256kbps

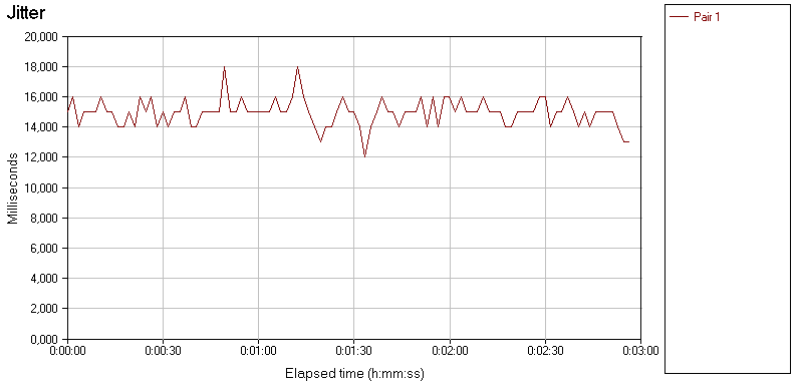


Figura 55 *Jitter* dos fluxos VoIP (14 a 16ms) na presença de tráfego BE de 512kbps

2º Teste

Foi criada 2 fluxos de voz, e atribuiu-se o parâmetro do **Priority** com o valor de **100 kbps** (94,4)

Perdas De voz	***Jitter* médio em ms**	**MGEN (*thoughput* em kbps)**	***Drop rate BE no router1*(bps)**
0	6,524	64	0
0	8,241	128	0
0	14,310	256	78000
0	14,171	512	82000

Tabela 11 2º teste 2 fluxo de voz gerado no *Chariot*

Gráficos do jitter em cada stream

É possível verificar que os resultados são muito semelhantes ao 1º teste. Os valores de jitter podem variar na realização do mesmo teste. O importante é que o valor seja, mais baixo que 20ms.

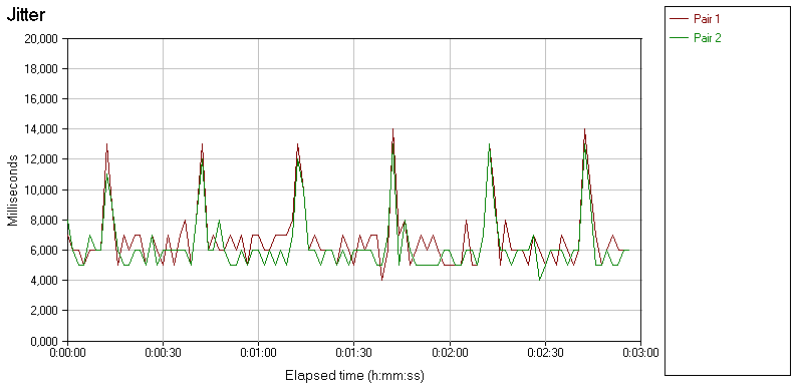


Figura 56 *Jitter* dos fluxos VoIP na presença de tráfego BE de 64kbps

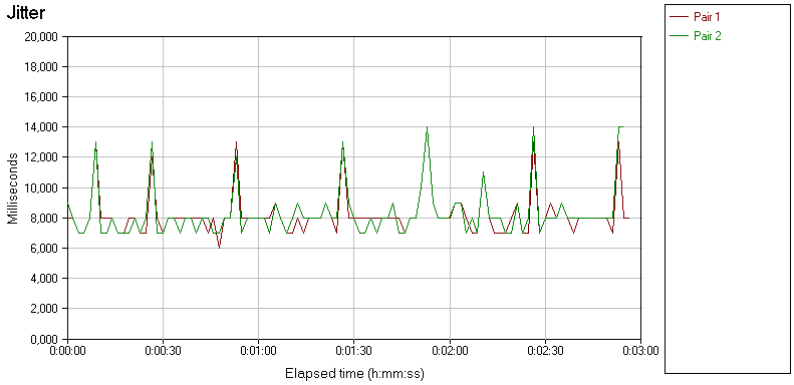


Figura 57 *Jitter* dos fluxos VoIP na presença de tráfego BE de 128kbps

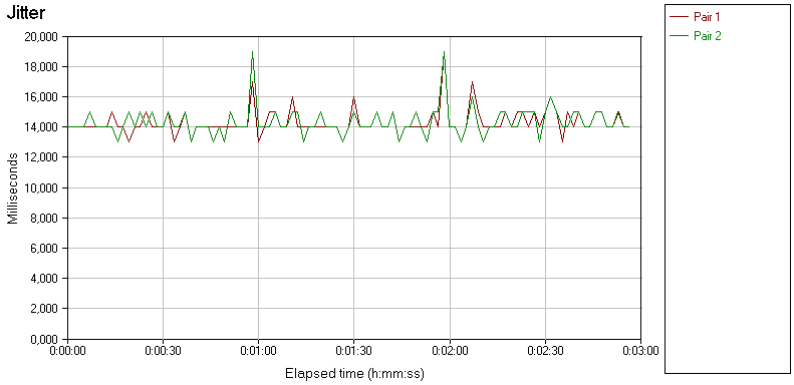


Figura 58 *Jitter* dos fluxos VoIP na presença de tráfego BE de 256kbps

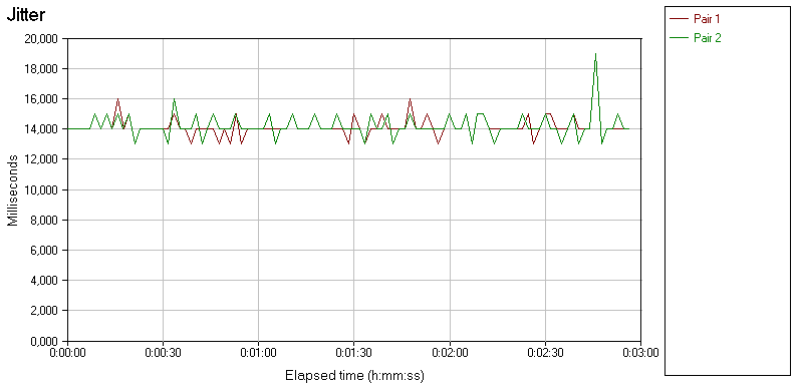


Figura 59 *Jitter* dos fluxos VoIP na presença de tráfego BE de 512kbps

3º Teste

Foi criada 3 fluxos de voz, e atribuiu-se o parâmetro do Priority com o valor de 142 kbps (141,6)

Perdas De voz	***Jitter* médio em ms**	**MGEN (*thoughput* em kbps)**	***Drop rateBE no router1* (bps)**
0	7.078	64	0
0	10,675	128	31000
0	10,536	256	100000
0	11,530	512	260000

Tabela 12 3º teste 3 fluxo de voz gerado no Chariot

Verifica neste teste que de facto começam aparecer perdas já no 2º caso com o MGEN a gerar 128kbps, o que é esperado.

Largura de banda disponível BE (sem tráfego BE gerado) = Largura de banda - *priority* 142 kbps + LB para sinalização 8 kbps = 256 - 150=**106 kbps**

LB efectiva = largura de banda disponível **-** Tráfego do MGEN =106-64=**96 kbps** (valor positivo então não há perdas, pela a tabela anterior está de acordo com o esperado)

LB efectiva = largura de banda disponível **-** Tráfego do MGEN =106-128=**-22 kbps** (valor negativo então há perdas, está de acordo com o esperado)

Verificou-se que o *jitter* tem um valor mais baixo, eventualemente devido ao facto ter sido alocado mais largura de banda, no entanto não é largura banda o factor para diminuir o *jitter*. O valor teórico é 47.2 kbps no entanto a efectivo é à volta dos 43 kbps. A Cisco recomenda alocar um pouco mais, por isso deve-se utilizar o valor teórico, porque o Router pode utilizar alguma largura de banda e introduzir jitter e atraso.

"*It is always safest to allocate to the priority queue slightly more than the known required amount of bandwidth*. (…) *However, because the network and the router or switch can use some of the bandwidth and introduce jitter and delay, allocating slightly more than the required amount of bandwidth (such as 25 kbps) ensures constancy and availability.*"[24]

No entanto o jitter tem origem nos seguintes factores [65]:

- Atraso no processo de codificação do sinal
- Packetization Delay, é o atraso do tempo que levam a colocar dos dados no payload do pacote.
- Serialization Delay, é um atraso fixo relacionado com o tempo para inserir dados de frames para a interface de rede.
- Queuing/Buffering Delay, atraso devido às filas de espera e aos buffers.
- Network Switching Delay, atraso relacionados com o processo de comutação

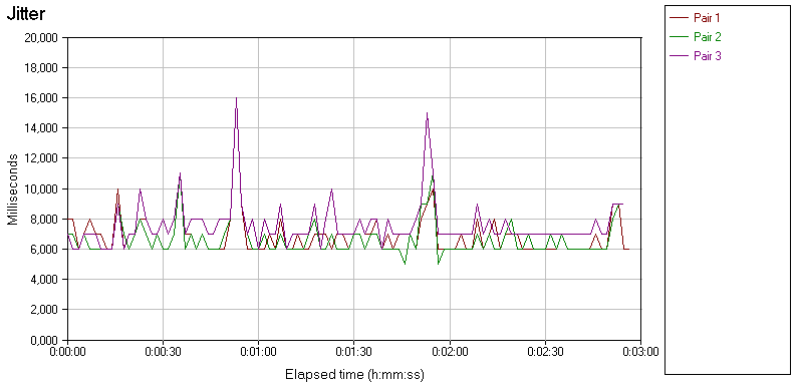


Figura 60 *Jitter* dos fluxos VoIP na presença de tráfego BE de 64kbps

[24] http://www.cisco.com/univercd/cc/td/doc/product/software/ios121/121cgcr/qos_c/qcprt2/qcdconmg.htm

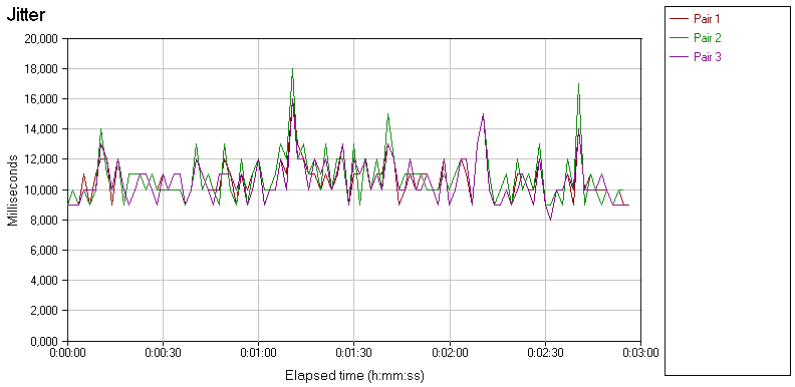


Figura 61 *Jitter* dos fluxos VoIP na presença de tráfego BE de 128kbps

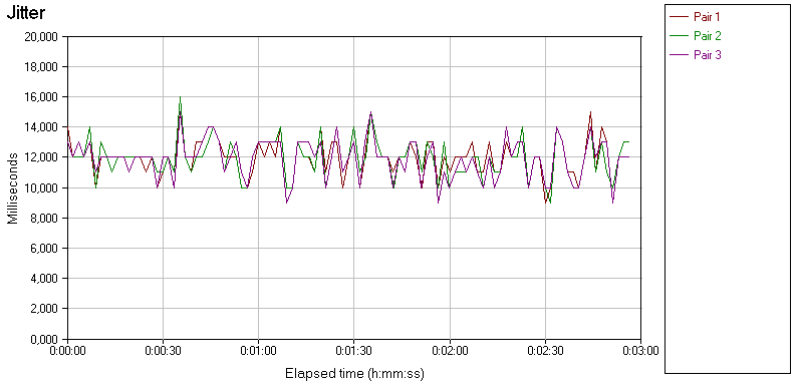


Figura 62 *Jitter* dos fluxos VoIP na presença de tráfego BE de 256kbps

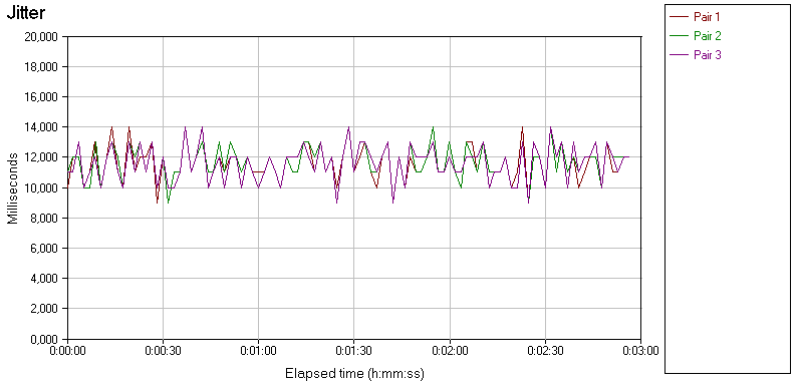


Figura 63 *Jitter* dos fluxos VoIP na presença de tráfego BE de 512kbps

4º Teste

Foi criada 4 fluxos de voz, e atribuiu-se o parâmetro do **Priority** com o valor de **190 kbps** (188,8).

Nota importante: A Cisco [25] recomenda que a soma de todas as alocações de largura de banda não deve exceder 75 porcento da largura total disponível. (neste caso 190+8=198/256=77%), por isso na prática este número de fluxos VoIP não deve ser utilizado.

Perdas De voz	***Jitter* médio em ms**	**MGEN (*thoughput* em kbps)**	***Drop rateBE no router1* (bps)**
0	8,328	64	0
0	7,904	128	48000
0	7,758	256	124000
0	8,040	512	265000

Tabela 13 4 º teste 4 fluxo de voz gerado no Chariot

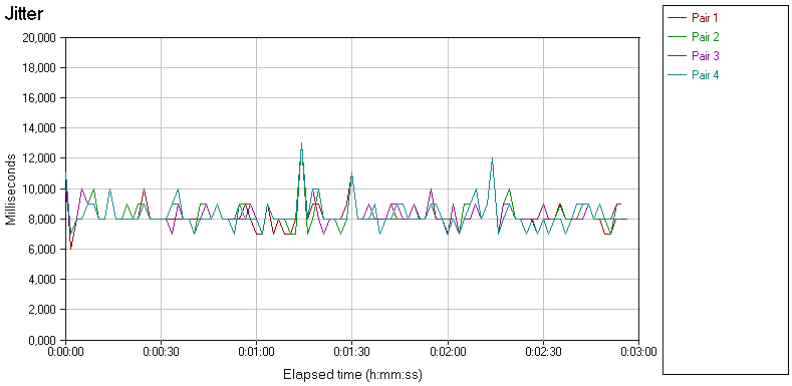


Figura 64 *Jitter* dos fluxos VoIP na presença de tráfego BE de 64kbps

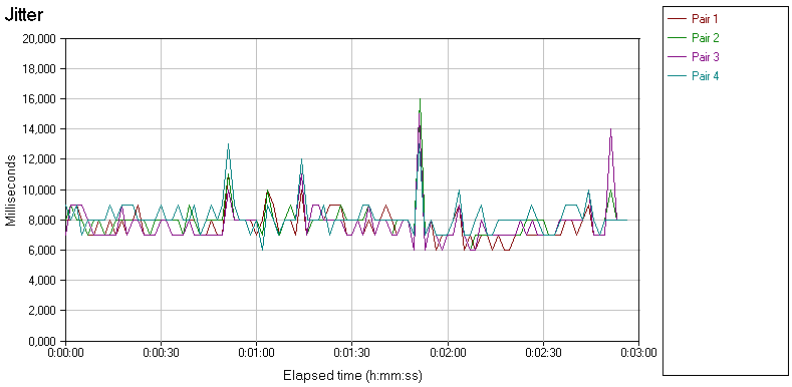


Figura 65 *Jitter* dos fluxos VoIP na presença de tráfego BE de 128kbps

[25] Cisco IOS Quality of Service Solutions Configuration Guide,(capítulo Congestion Management Overview)

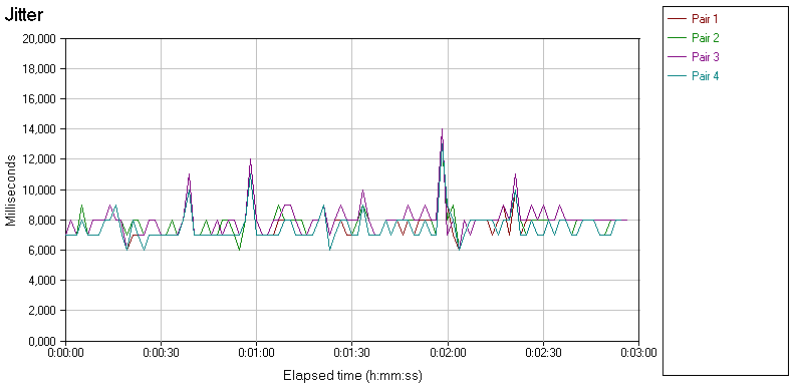


Figura 66 *Jitter* dos fluxos VoIP na presença de tráfego BE de 256kbps

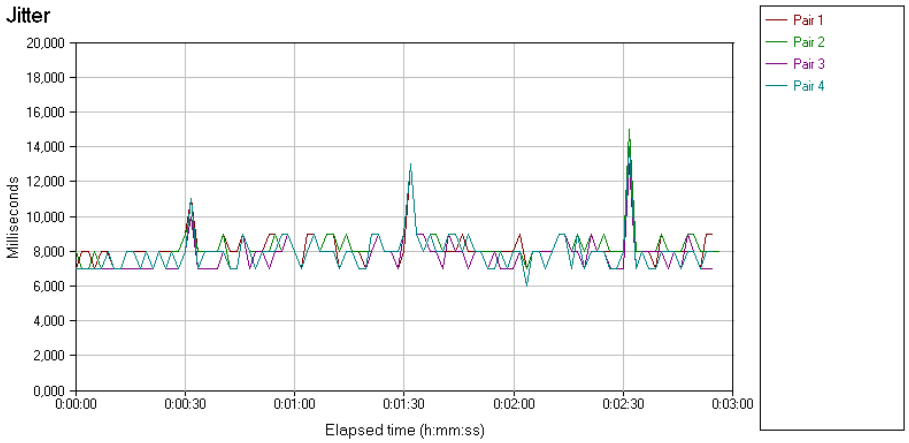


Figura 67 *Jitter* dos fluxos VoIP na presença de tráfego BE de 512kbps

Resultado no Router1 para 1º teste

A seguir é mostrado o output do comando **sh policy-map int ser0/0**, no 1º teste realizado com o tráfego BE a 64kbps, este foi o comando utilizado para verificar o drop rate.

Router1#sh policy-map interface serial 0/0

```
 Service-policy output: pm
   Class-map: voz (match-all)
     8806 packets, 950958 bytes
     5 minute offered rate 36000 bps, drop rate 0 bps
     Match:  dscp ef (46)
     Match: protocol ipv6
     Queueing
       Strict Priority
       Output Queue: Conversation 264
       Bandwidth 50 (kbps) Burst 1250 (Bytes)
       (pkts matched/bytes matched) 8806/950958
       (total drops/bytes drops) 0/0
```

Em todos os testes realizados não houve perdas no Router1 e no Chariot.

Router1#sh policy-map interface serial 0/0

Class-map: sinalizacao (match-all)
0 packets, 0 bytes
5 minute offered rate 0 bps, drop rate 0 bps
Match: dscp cs3 (24) af31 (26)
Match: protocol ipv6
Queueing
Output Queue: Conversation 265
Bandwidth 8 (kbps) Max Threshold 64 (packets)
(pkts matched/bytes matched) 0/0→não foi gerado tráfego deste tipo
(depth/total drops/no-buffer drops) 0/0/0

Uma vez que foi gerado streams de voz e não de skinny não existem pacotes encontrados ou matched, ou não passaram pelo o router pacotes skinny.

Router1#sh policy-map interface serial 0/0

Class-map: class-default (match-any)
3091 packets, 1577610 bytes
5 minute offered rate 45000 bps, **drop rate 0 bps**
Match: any

A classe Best Effort BE também não perdas no primeiro teste. A seguir é mostrado o output do comando sh policy-map int ser0/0, no **4º teste** ou seja quatro streams de voz gerado no Chariot, e no MGEN a gerar o tráfego a 512kbps de BE.

Router1#sh policy-map interface serial 0/0

Class-map: voz (match-all)
34317 packets, 3706236 bytes
5 minute offered rate 98000 bps, drop rate 0 bps
Match: dscp ef (46)
Match: protocol ipv6
Queueing
Strict Priority
Output Queue: Conversation 264
Bandwidth 190 (kbps) Burst 4750 (Bytes)
(pkts matched/bytes matched) 34317/3706236
(total drops/bytes drops) 0/0

Router1#sh policy-map interface serial

Class-map: sinalizacao (match-all)
0 packets, 0 bytes
5 minute offered rate 0 bps, drop rate 0 bps
Match: dscp cs3 (24) af31 (26)
Match: protocol ipv6
Queueing
Output Queue: Conversation 265
Bandwidth 8 (kbps) Max Threshold 64 (packets)

(pkts matched/bytes matched) 0/0
(depth/total drops/no-buffer drops) 0/0/0→não foi gerado tráfego deste tipo

Router1#sh policy-map interface serial

Class-map: class-default (match-any)
23365 packets, 12775697 bytes
5 minute offered rate 333000 bps, **drop rate 265000 bps**
Match: any

De facto verifica-se que está de acordo com o previsto. Como são quatro vezes mais fluxos de voz, passam quatro vezes mais pacotes. O RTP/UDP envia o mesmo número de pacotes por segundo, já o protocolo TCP não o faz.

No 1º teste: (pkts matched/bytes matched) **34317**/3706236

No 4º teste: (pkts matched/bytes matched) **8806**/950958

8806(para uma *stream* VoIP)*4 = **35224** pacotes (durante 3 minutos de simulação)

muito idêntico ao valor dado no router 34317

Conclusões

Verificou-se que *jitter* nunca passa dos 15ms, no entanto há picos de 18 ms máximos. variação máxima tolerável do *jitter* é entre 20 a 50 ms [39]. O Chariot não dá os valores do atraso, mas o que interessa é a variação do atraso ou *jitter*. Verificou-se que não houve perdas nas *streams* de voz no Chariot. Ao fim de cada teste executou-se o comando no Router1 **sh policy-map interface serial 0/0**. Em todos os testes, verificou-se o valor do **drop rate** da classe BE, também verificou-se que não houve perdas na classe de voz no Router1.

6.3.6. Alternativas para o QoS para IPv6

De acordo o descrito no ponto 4.6.5, Não existem muitas alternativas, porque é necessário que o IOS da Cisco suporte os mecanismos de QoS em IPv6.

No caso de QoS em IPv4, seria possível utilizar o Compressed Real-Time Protocol (CRTP), o comando IP RTP priority. Em ligações de baixa largura de banda, é aconselhado utilizar o ppp multilink[26], no entanto não é suportado o comando ip rtp priority para IPv6. Uma configuração em IPv4 seria a seguinte:

```
router(config)# interface multilink 1
router(config-if)# ip address 10.0.0.8 255.0.0.0
router(config-if)# no ip directed-broadcast
router(config-if)# ip rtp priority 16384 16383 200→Não suportado
router(config-if)# no ip mroute-cache
router(config-if)# fair-queue 64 256 0
router(config-if)# ppp multilink
router(config-if)# ppp multilink fragment-delay 20
router(config-if)# ppp multilink interleave
router(config-if)# max-reserved-bandwidth 80
```

A única alteranativa seria *Policy-based packet marking*, no entanto para tráfego de voz é recomendado pela Cisco o LLQ. O método *Policy-based packet marking* não foi testada, por isso não há confirmação se seja uma solução ou não.

[26] http://www.cisco.com/en/US/products/ps6350/products_configuration_guide_chapter09186a008044677d.html ou "Reducing Latency and Jitter for Real-Time Traffic Using Multilink PPP Roadmap"

6.4. Implementação dum cenário VoIP em IPv6 nativo

A arquitectura SIP permite funcionar ponto a ponto, num entanto existem muitas vantagens (por exemplo autenticação) em ter como intermediário um servidor, na arquitectura distribuída do SIP. Assim foi implementado um cenário utilizando o SIP Express Router (SER) da IPTEL, e terminais Linphone, usando o sistema operativo Linux (Fedora Core 4). Foi necessário implementar DNSv6, experimentou-se primeiro utilizar os parênteses rectos **[2000::1]**, no entanto a aplicação Linphone não permitia resultando num erro fatal. No anexo 8.16 e 8.18 estão os manuais de instalação deste cenário.

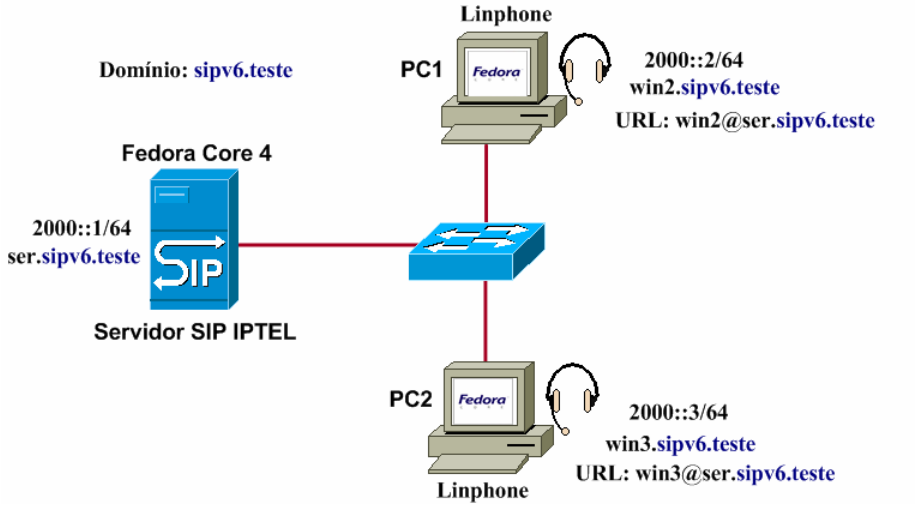


Figura 68 Cenário implementado para SIPv6

O Linphone[27] apenas suporta VoIP, e *instant messaging*, já o SIP Communicator suporta também vídeo. Na versão antiga do SIP Comunicator está disponível no seguinte site[28] A nova versão está em desenvolvimento[29]. No anexo 8.3.1 estão descritos estes softphones.

6.4.1. DNSv6

Foi criado um domínio com o seguinte nome **sipv6.teste**, e os terminais com os nomes **win1** e **win2**, uma vez que foi testado primeiro com SIP Communicator em Windows, assim optou-se por não estar a alterar os nomes na configuração DNS, tomou-se a opção de mantê-los.

27 http://www.linphone.org/

28 https://sip-communicator.dev.java.net/index_old.html

29 http://sip-communicator.org/

Em Anexo 8.17 é mostrado um pedido DNSv6 com nome **ser.sipv6.teste** que é o nome do servidor SER e outras mensagens DNSv6. Nos ficheiros de configuração (ver em Anexo 8.17) existem o tipo endereço (address record type) AAAA que é para IPv6, e o tipo A é para o tipo de endereço IPv4. O protocolo DNS utiliza o porto 53.

Nota: Para o caso de IPv4 não é necessário a configuração de DNS, é possível utilizar directamente os endereços IPv4 tanto nos terminais como no servidor.

6.4.2. Verificação das mensagens SIPv6

O sistema SIP consiste em SIP *User Agents* UA e SIP *Proxies*. Os UAs que são os terminais e os *proxies* são os servidores ou *routers*. O protocolo de sinalização SIP, como qualquer outro protocolo é baseado em pedidos e respostas. As respostas mais importantes são o "INVITE" para iniciar a sessão e "BYE" para terminar a sessão. As mensagens "INVITE" e "OK" servem para estabelecer a sessão de multimédia (chamada telefónica ou de vídeo) e têm uma mensagem SDP para descrever o descodificador que o UA conhece, e o porto onde deve ficar à escuta para receber o UDP do terminal remoto. Anteriormente foi utilizado a versão 1.1 do Linphone, dado os problemas indicados no Anexo página 208.

A experiência foi realizada na sequência indicada:

1. win2 regista-se no ser
2. win3 regista-se no ser
3. win2 telefona ao win3
4. win3 atende fala um pouco e desliga
5. win2 manda um olá para win3
6. win3 responde um olá ao win2
7. win3 desliga o *softphone*
8. win2 desliga o *softphone*

No anexo 8.15 estão todas mensagens SIP capturadas.

Processo de registo dum telefone no servidor

Primeiro o terminal tem de se registar no servidor SER, para que outros os utilizadores no domínio possam ligar a ele e vice-versa. O SER também permite autenticar os utilizadores, no entanto essa opção ou módulo não foi utilizada, uma vez que o objectivo era obter as mensagens SIP essenciais.

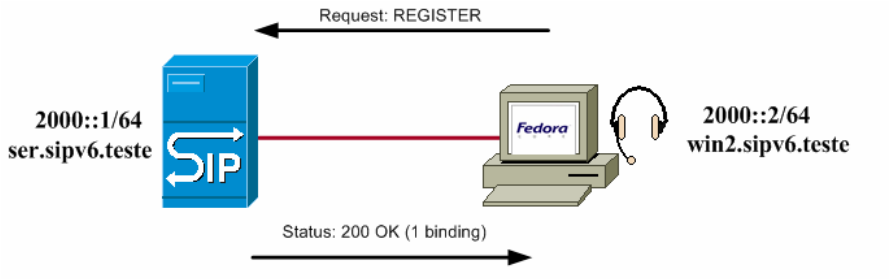


Figura 69 Registo de um telefone no Servidor SIP

Verifica-se que o servidor SIP utiliza o porto 5060, enquanto o protocolo Skinny utiliza o porto 2000. Para registar o terminal é enviado o request-Line REGISTER. O servidor depois envia uma resposta, sobre o estado dizendo "*Status-Line: SIP/2.0 OK*", e se ficou registado com sucesso. Com o seguinte SIP URI: **win@ser.sipv6.teste.**

Em Anexo 8.19 estão as duas mensagens SIP capturadas com mais promenor.

Estabelecimento da chamada, RTP, e Terminação

O terminal para realizar uma chamada, depois de estar registado, envia o pedido **INVITE** seguido do destino. Para o **win2@ser.sipv6.teste** realizar uma chamada para o **win3@ser.sipv6.teste.**

INVITE sip:win3@ser.sipv6.teste SIP/2.0

Na mensagem **INVITE** é também enviado os cabeçalhos SDP. O protocolo SDP é o *Session Description Protocol* (SDP, [RFC 2327]) é utilizado para negociar os componentes de comunicação do canal, por exemplo os codecs, portos, e protocolo de *streaming* a utilizar.

Na figura seguinte é mostrado o terminal com o endereço IPv6 **2000::2/64** e o URI SIP com o nome **win2@ser.sipv6.teste,** a enviar o pedido ao servidor, o servidor depois encaminha o pedido **INVITE** para o destino com o URI SIP com o nome **win3@ser.sipv6.teste**. Depois o terminal destino responde ao servidor a dizer que recebeu o **INVITE** com a mensagem **100 trying**, e um **Dialog Establishment.** Depois enviar um mensagem **180 Ringing** a indicar que está a tocar, mas ainda não atendeu. Ao atender a chamada, é enviado uma mensagem **200 OK SIP/SDP**, a dizer que aceitou a chamada, esta mensagem ao chegar ao outro lado é respondido com o **ACK.**

Nesta fase é realizado a comunicação áudio ponto a ponto entre eles, por meio do protocolo RTP. O terminal win3 (**2000::3/64**) depois pretende terminar a conversa, irá terminar a ligação com uma mensagem **BYE**. O termina win2 depois envia uma mensagem **OK** para o **win3**.

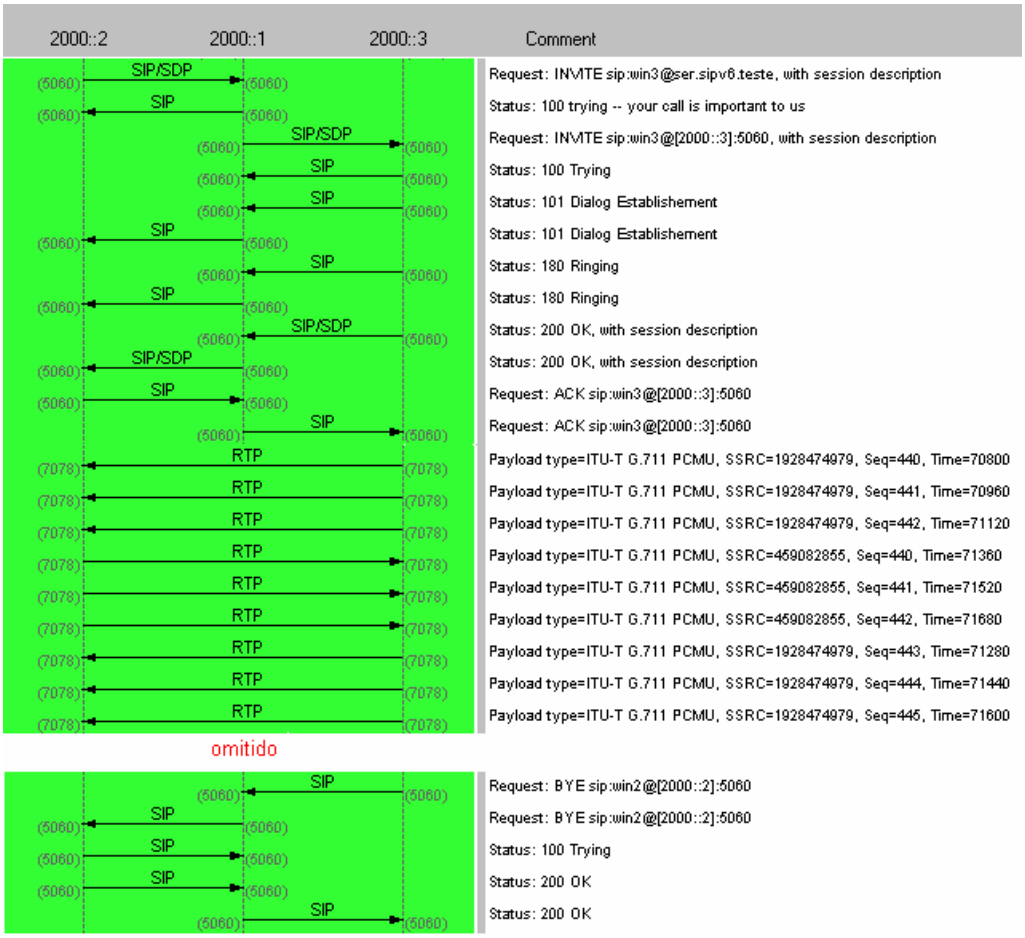


Figura 70 Estabelecimento, RTP, terminação

O servidor reencaminha sempre os pedidos SIP para o destino. No Anexo na Figura 155 Mensagem SIP/SDP é mostrado a mensagem **INVITE SIP/SDP** enviado para o SER. Verifica-se que os campos sobre QoS para os pacotes RTP não estão marcados, e que utiliza o codec G.711 (ver em anexo a Figura 156 Pacote RTP capturado no Ethreal).

Mensagem de texto para *chat*

Como o Linphone também suporta *Instant Messaging*, foi testado e analisado as mensagens envolvidas. Estas mensagens são enviados são do tipo MESSAGE.

MESSAGE sip:win3@ser.sipv6.teste (text/plain)

Estas mensagens SIP não estabelecem numa ligação primeiro, assim passam sempre pelos proxies. A mensagem de texto é transportada no corpo dos pedidos SIP.

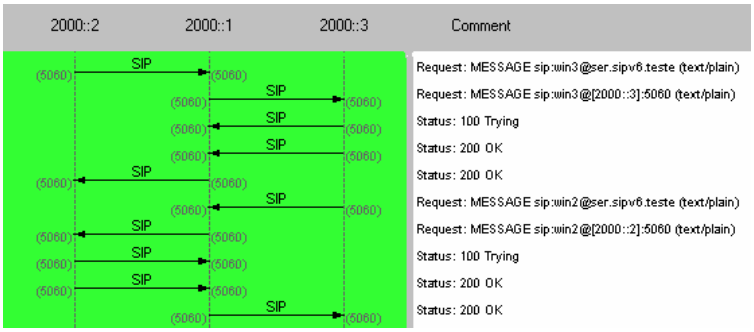


Figura 71 SIP com *Instante messaging*

O servidor apenas reencaminha estas mensagens para o destino. Ao chegar ao destino é enviado uma mensagem tipo **200 OK**. A seguir é mostrado o corpo da mensagem e o seu conteúdo neste caso "ola". Pelo que é possível verificar a mensagem não vai cifrada.

```
⊞ Frame 5396 (407 bytes on wire, 407 bytes captured)
⊞ Ethernet II, Src: 192.168.0.2 (00:11:11:24:67:f6), Dst: 192.168.0.1 (00:11:11:59:15:c1)
⊞ Internet Protocol Version 6
⊞ User Datagram Protocol, Src Port: 5060 (5060), Dst Port: 5060 (5060)
⊟ Session Initiation Protocol
  ⊞ Request-Line: MESSAGE sip:win3@ser.sipv6.teste SIP/2.0
  ⊟ Message Header
      Via: SIP/2.0/UDP [2000::2]:5060;branch=z9hG4bK428459431
    ⊞ From: <sip:win2@ser.sipv6.teste>;tag=371488425
    ⊞ To: <sip:win3@ser.sipv6.teste>
      Call-ID: 364721561@2000::2
      CSeq: 20 MESSAGE
      Max-Forwards: 5
      User-Agent: Linphone-1.2.0/eXosip
      Expires: 120
      Content-Type: text/plain
      Content-Length:     3
  ⊟ Message body
    ⊟ Line-based text data: text/plain
      ola
```

Figura 72 Mensagem "ola" em texto enviado

Desligar a sessão com o servidor

O utilizar **win3@ser.sipv6.teste** pretende terminar a ligação com o servidor SIP,

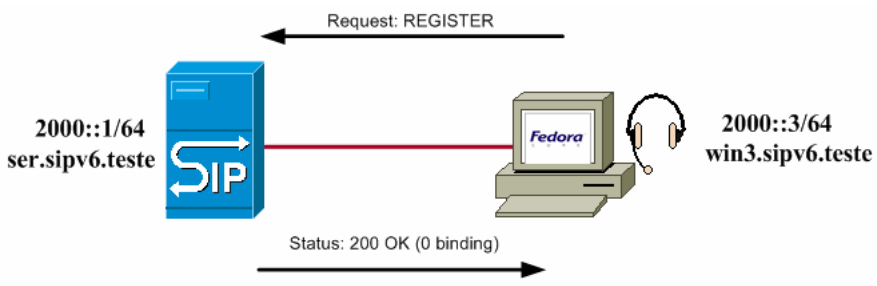


Figura 73 Registo de um terminal no Servidor

Na figura seguinte é mostrado o pedido de **REGISTER,** em que o terminal pretende desligar a sessão com o utilizador. O servidor depois responde com a seguinte mensagem Status: **200 SIP OK (0 binding)**.

Conclusão

O cenário também foi testado em IPv4. Verificou-se de facto a única diferença entre SIPv4 e SIPv6 são as mensagens que têm o endereço IPv6 ou no URI.

7. Conclusão

Em anexo (Figura 74 Calendarização do projecto), são apresentadas as várias fases do projecto, a sequência das tarefas realizadas, as reuniões, e deliverables realizados ao longo do projecto.

Neste projecto, numa fase inicial foi realizado um estudo sobre as arquitecturas VoIP, mecanismos de QoS, e o seu suporte para IPv6. Foi realizado um levantamento sobre as soluções para VoIPv6 em *Open source*, uma vez que a Cisco apenas suporta IPv4, nas suas soluções de voz. Foi realizada uma análise dos protocolos envolvidos na arquitectura Cisco com a realização de um cenário. Após a concretização deste, foram investigadas as soluções para integração IPv6-IPv4. Para o efeito, foi realizado um estudo sobre os mecanismos de transição IPv4 para IPv6, e escolhido o túnel GRE para ser utilizado nos testes. Efectuou-se uma análise do processo de encapsulamento e o *overhead* introduzido por este mecanismo. Dado que a voz é sensível ao tráfego existente na rede, foi necessário investigar uma solução de qualidade de serviço para o cenário.

Para testar a qualidade de serviço recorreu-se a duas aplicações, o Chariot e o MGEN. O Chariot foi utilizado para gerar tráfego de voz IPv4 ao passo que o MGEN foi utilizaso para tráfego IPv6. Foi também necessário ter em conta os fluxos de voz dos telefones Cisco e efectuar um estudo sobre os conceitos associados aos codecs, métricas, débitos e outros.

O VoIPv6 ainda está numa fase imatura, no entanto já existem implementações em *Open Source* e existem projectos[30] que analisam os mecanismos de transição IPv4 para IPv6 e vice-versa (SIP proxies). O protocolo SIP[31] é muito utilizado, uma vez que este é bastante simples e permite o início de sessões de aúdio (nomeadamente VoIP), vídeo, *instante messaging,* e outros formatos. Mesmo a nível de IPv6, o protocolo SIP poderá ser utilizado em equipamentos *wireless* (telefone VoIP usando o WiFi) ou móveis (UMTS).

Não se sabe quando a Cisco irá ter soluções de voz sobre IPv6 nos próximos anos. Os mecanismos de transição podem solucionar o problema, mas no entanto não são uma solução ideal. Convém referir também que o governo japonês está a investir grandemente em soluções VoIPv6.

[30] http://www.6net.org/
[31] RFC 3261 mais recente torna obsoleto a versão RFC 2543

Verificou-se que os *softphones* VoIP IPv4 são bastante mais estáveis quando comparadas com as existentes para IPv6. Existem muitos serviços que funcionam em IPv4, mas para que estes se tornem estáveis em IPv6, ainda irá demorar algum tempo.

Neste projecto, foi também efectuado um levantamento na Internet de soluções para VoIPv6 tanto a nível de software como de hardware. De seguida, foi implementado um cenário prático recorrendo a um servidor IPTEL e vários *softphones* IPv6. No entanto, verificou-se que era necessário recorrer ao DNS, dado que as aplicações não permitiam a introdução estática de endereços IPv6. Alguns deste *softphones* são extremamente difíceis de instalar, no entanto apesar disso, conseguiu-se implementar um cenário utilizando o SER da IPTEL.

Os Servidores *Open Source* SIP como a SER da IPTEL e VOCAL, têm grande potencial, porque oferecem uma solução VoIP de uma forma gratuita. Como é evidente, estes são mais difícil de configurar, quando comparados com o CallManager da Cisco, uma vez que são necessários conhecimentos de Linux e eventualmente de programação em C. Uma outra vantagem dos servidores *Open Source* deve-se ao facto destes serem compatíveis com equipamentos de vários fabricantes. O Instituto Politécnico de Bragança[32] já implementou este servidor na sua rede IPv4.

No cenário Figura 43 seria possível integrar a solução implementada na Figura 68, no entanto o CallManager utiliza o protocolo Skinny e o SER utiliza o SIP. Seria possível estabelecer uma ligação apenas entre IPv6 para IPv6 ou de IPv4 para IPv4 ou IPv4 para analógico e vice-versa. Para estabelecer uma chamada do IPv6 para IPv4, seria difícil uma vez que os routers Cisco (nos router por exemplo não existe o comando *session target IPv6:2001::1*) e o CallManager da Cisco não suporta SIPv6. Seria necesário ou passar os telefones Cisco para SIP e integrar com o servidor SER, uma vez que o SER suporta IPv6/IPv4. Outra opção seria utilizar servidores como o Asterisk que suportam ambos os protocolos SIP, e Skinny ou SCCP em IPv4/IPv6. Poderão existir outras opções.

De facto é um caso a estudar, aqui não é dada nenhuma solução para este problema uma vez que não foi implementado, e será uma boa questão para um futuro projecto.

[32] http://www.ccom.ipb.pt/voip/modules/voip/show_users.php

8. Anexos

8.1. Calendarização do projecto

O projecto divide-se em cinco fases a fase inicial onde é realizado um estudo sobre as arquitecturas VoIP, e QoS em IPv4/IPV6. A fase de pré-requesitos onde é estudado solução para um mecanismo de translação IPv4 para IPv6. A fase da implementação onde é implementado o túnel GRE, e cenário em linux. A fase de QoS onde foi implementado um política de QoS, com respectivos testes. A fase final para a conclusão do relatório. Em todas as fases foram criados *deliverables* ou documentos para apresentar nas reuniões.

	Task Name	Duration	Start	Finish
1	**Fase Inicial**	**15,13 days**	**Tue 27-09-05**	**Tue 18-10-05**
2	Reunião	0,13 days	Tue 27-09-05	Tue 27-09-05
3	Estudo IPv6 relatório 04/05	1 day	Wed 28-09-05	Wed 28-09-05
4	Estudo sobre conceitos QoS	1,62 days	Thu 29-09-05	Fri 30-09-05
5	Elaboração do deliverable sobre conceitos QoS IPv4 versus IPv6	5 days	Mon 03-10-05	Fri 07-10-05
6	Elaboração do deliverable sobre arquitecturas VoIP	5 days	Mon 10-10-05	Fri 14-10-05
7	Reunião (entrega das deliverables) -18 out	0,13 days	Tue 18-10-05	Tue 18-10-05
8	**Fase de pré-requesitos**	**14,13 days**	**Wed 19-10-05**	**Tue 08-11-05**
9	Estudo das soluções (mecanismo de transição) para o cenário definid	7 days	Wed 19-10-05	Thu 27-10-05
10	Estudo de soluções VoIP em IPv6 nativa e elaboração do deliverable	7 days	Fri 28-10-05	Mon 07-11-05
11	Reunião - 8 Nov (Entrega de deliverables)	0,13 days	Tue 08-11-05	Tue 08-11-05
12	**Implementação**	**63 days**	**Wed 09-11-05**	**Fri 03-02-06**
13	Testes dum cenário VoIP em IPv4	4 days	Wed 09-11-05	Mon 14-11-05
14	Implementação do cenário com a solução encontrada	4 days	Tue 15-11-05	Fri 18-11-05
15	(estudar para GIRS)	3 days	Mon 21-11-05	Wed 23-11-05
16	Testar o cenário completo	4 days	Thu 24-11-05	Tue 29-11-05
17	Deliverable Protocolos VoIP dos pacotes capturados (montar cenário,	10 days	Wed 30-11-05	Tue 13-12-05
18	Testar um cenário VoIP IPv6 em linux (menos prioritário)	8 days	Thu 01-12-05	Mon 12-12-05
19	Deliverable VoIP IPv6 em linux (caso terminado) dos pacotes capturad	7 days	Wed 14-12-05	Thu 22-12-05
20	(Terminar deliverable VoIP IPv6 em linux dos pacotes capturados)	7 days	Thu 26-01-06	Fri 03-02-06
21	Reunião (entrega de deliverables) - 13 de Dez	0,13 days	Tue 13-12-05	Tue 13-12-05
22	**QoS (Qualidade de Serviço)**	**41,88 days**	**Tue 13-12-05**	**Wed 08-02-06**
23	Preparação para QoS	2 days	Tue 13-12-05	Thu 15-12-05
24	Testes num ceánário IPv4	2 days	Thu 15-12-05	Fri 16-12-05
25	Configuração de QoS IPv6 no cenário pretendido ou definido.	4 days	Tue 20-12-05	Fri 23-12-05
26	(Férias) documentação QoS e trabalho de GPC	5 days	Mon 26-12-05	Fri 30-12-05
27	(Terminar os trabalhos para as cadeiras GPC e APR)	6 days	Wed 18-01-06	Wed 25-01-06
28	Testar QoS no cenário completo (eventuais altreações e optimização)	2 days	Mon 09-01-06	Tue 10-01-06
29	Estudo de outros métodos que podem ser implementados	3 days	Mon 30-01-06	Wed 01-02-06
30	Deliverable sobre QoS a parte de implementação e análise dos resulta	18 days	Mon 16-01-06	Wed 08-02-06
31	Reunião	1 day	Tue 17-01-06	Tue 17-01-06
32	Reunião	0,13 days	Tue 31-01-06	Tue 31-01-06
33	**Fase Final**	**18 days**	**Wed 01-02-06**	**Fri 24-02-06**
34	Preparação	4 days	Wed 01-02-06	Mon 06-02-06
35	Entrega versão preliminar (marcado 8 de Fev) se possível entregar an	0 days	Wed 08-02-06	Wed 08-02-06
36	Correcção por parte dos orientadores	3 days	Wed 08-02-06	Fri 10-02-06
37	Reunião para as alterações necessárias	0,13 days	Fri 10-02-06	Fri 10-02-06
38	Fazer as devidas alterações indicadas	4 days	Fri 10-02-06	Wed 15-02-06
39	Entrega da versão final do relatório (15 de Fev)	0 days	Wed 15-02-06	Wed 15-02-06
40	Preparação da Apresentação.	5 days	Thu 16-02-06	Wed 22-02-06
41	Apresentação (22 de Fev)	1 day	Wed 22-02-06	Wed 22-02-06
42	Elaboração do Artigo	5 days	Thu 16-02-06	Wed 22-02-06
43	Entrega artigo versão preliminar	0 days	Thu 23-02-06	Thu 23-02-06
44	Correções do artigo por parte dos orientadores	0,38 days	Fri 24-02-06	Fri 24-02-06
45	Alterar artigo	2 days	Thu 23-02-06	Fri 24-02-06
46	Entrega artigo	0 days	Fri 24-02-06	Fri 24-02-06
47	Conclusão do projecto	0 days	Fri 24-02-06	Fri 24-02-06

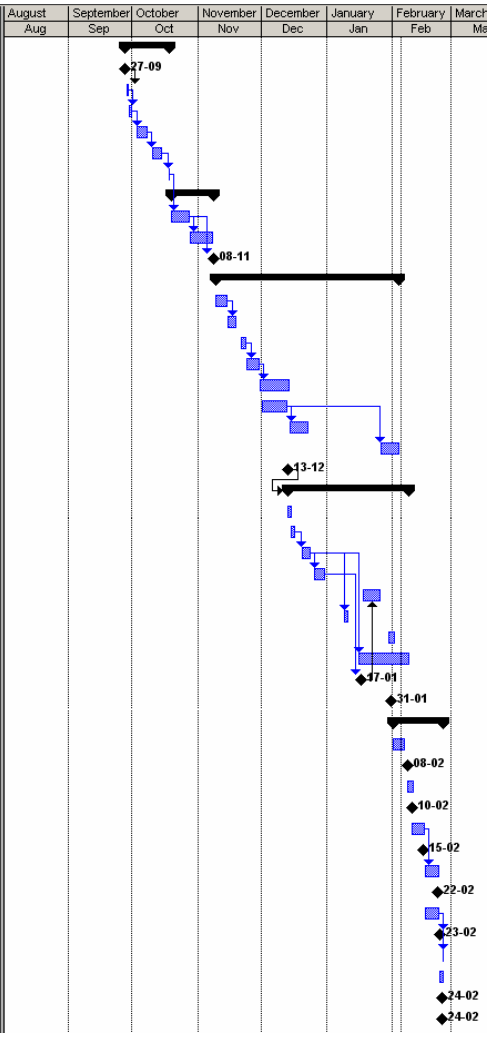


Figura 74 Calendarização do projecto

8.2. Material utilizado

A escola tem dois telefones IP da Cisco modelos 7960 e 7910, estes telefones utilizam o CallManager da Cisco.

Figura 75 Telefones IP da Cisco modelos 7910 (à esquerda) e o 7960 (à direita)

Nome do produto: CISCO 7960

URL do produto: http://www.cisco.com/warp/public/cc/pd/tlhw/prodlit/7960_ds.htm

Fabricante: Cisco

Protocolos suportados: SIP, SCCP

Descrição:

Ele suporta H.323 assumindo que existe um Call Manager, que é um software da CISCO para gerir os *hardphones.* A comunicação entre o Call Manager e o Cisco 7960 é realizado pelo o protocolo proprietário da *Cisco Skinny Protocol* (não é um norma H.323!), assim a comunicação com a norma H.323 é apenas executado externamente ao cliente e apenas executado pelo o Call Manager e não o telefone. Ele suporta o protocolo SIP mas no ponto de vista da norma. O *firmware* (Application LoadID: P0S30203; BootLoadID: PC03A300) é perfeitamente inter operável com o sistema VOCAL, IPTEL e com MSN Messenger. Este telefone é um pouco caro, no entanto é um bom telefone mas não é muito bom com o H.323. Funciona bem com o protocolo SIP, o que é muito bom. É muito confortável de ser utilizado, mesmo em conversas longas. O *display* ou ecrã é grande o suficiente de ser visto a uma grande distância. Este telefone tem 6 linhas. A imagem SIP deve ser carregada para o telefone, antes de funcionar como telefone SIP. O telefone é muito inter operável com o protocolo SIP, e pode ter até 6 contas SIP podem ser registadas simultaneamente.

Infelizmente não é possível de recusar uma chamada de entrada, o telefone pode ser colocado no modo "*Do not distrub*".

Nota: Actualmente não existem quais quer soluções de voz da Cisco (CallManager, e telefones IP) VoIP, que suportam IPv6.

Call Manager da Cisco

O Call Manager da Cisco utilizado no projecto a versão 3.1. O *Call Manager* é um Cisco *certified* Windows 2000 *server* que contém *software* de *Call Manager* e serve para a gestão dos telefones e sinalização.

De seguida, são apresentadas algumas das funcionalidades desempenhadas pelo o servidor:

Processamento de Chamadas,
Sinalização e Controlo dos equipamentos,
Administração centralizada dos equipamentos telefónicos,
Serviços Extra e
APIs para os programadores.

O CallManager usado neste trabalho foi o Cisco ICS-7825-1133.

Figura 76 Cisco MCS 7800 *Series Server*, utilizado como Call Manager

Interface FXS da Cisco

A interface Foreign Exchange Station (FXS) liga directamente para um telefone analógico, máquina fax, ou outro dispositivo similiar e oferece toque, voltagem, e *dial tone*. A interface FXS da Cisco utiliza tomadas RJ-11, que permite ligações de telefonia básica, *keysets*, e Private Branch Exchanges (PBXs).

Nota: A interface Foreign exchange office (FXO), não é o mesmo que Foreign Exchange Station (FXS) e assim não oferece *dial tone*. Não se deve ligar o telefone a uma interface FXO. A desvantagem é quando se está a chamar para o PBX, o número de extensão pode ser chamado apenas como um sufixo Dual-Tone Multifrequency.

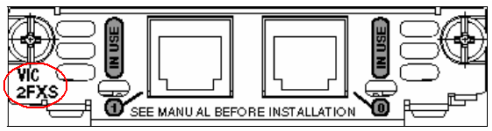


Figura 77 Interface FXS da Cisco com 2 portas para voz

Figura 78 Interface FXS hardware *slot*

A interface FXS é instalado no Network module *slot* do Router.

Interface WIC serial

Existem dois tipos de interface serial a WIC2T e WIC2A/S. A WIX2A/S apenas permite ligações até 128kbps. Para ter uma ligação serial com mais de 128kbps é necessário utilizar interfaces WIC2T. É possível ter uma interface WIC2T como DCE ligado a uma interface WICA/S DTE e funcionar a mais de 128kbps, no entanto ao atribuir a política de QoS à interface já não permite. A seguir é mostrado como se identifica cada um deles.

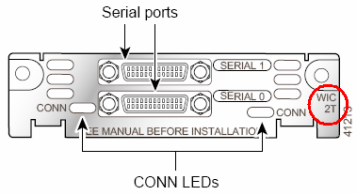


Figura 79 Identificação do tipo de Interface serial este é um WIC2T

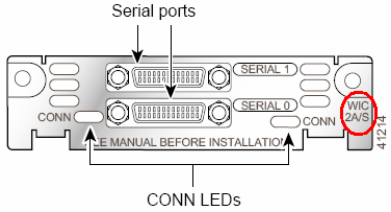


Figura 80 Identificação do tipo de Interface serial este é um WIC2A/S

Routers da Cisco modelo 2611XM

Uma vez que neste projecto foi necessário utilizar o IOS 12.4 que suporta IPv6, foi necessário utilizar os routers da Cisco do modelo 2611XM, com a capacidade de memória flash e DRAM superior a 32M. Assim é possível o IOS 12.4 ser armazenado na flash e descomprimida para a DRAM.

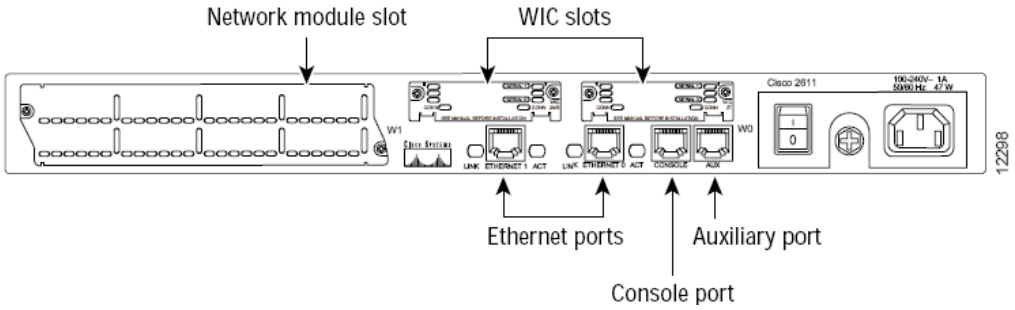


Figura 81 Router da Cisco modelo 2611

8.3. Telefonia IP *hardware* e *software* para IPv6

A seguir são apresentados alguns equipamentos que suportam VoIP em IPv6. Algumas soluções foram encontrada no livro do cookbook de IP telefonia[59].

8.3.1. Softphones

Foram instalados praticamente todos os *softphones* encontrados que suportam IPv6, são difíceis de instalar. Não se chegou a experimentar o *kphone, 6voice*. Para instalar o Kphone é necessário instalar também o KDE 3.1, no entanto apenas está disponível o *source*, por isso é necessário recompila-lo), existem artigos dizem que o kphone tem muitos *bugs*. O linphone 1.2 foi o único que funcionou para IPv6 e para IPv4. O SIP comunicator eventualmente seria necessário corrigir o problema através da recompilação do código em Java, foi possível estabelecer a ligação e a chamada, no entanto deu um erro relacionado com RTP. Este erro era muito semelhante ao dado no linphone na versão 1.1. No entanto uma nova versão so SIP Communicator está sair em breve.

Nome do produto: Windows Messenger
URL do produto: http://www.microsoft.com/windows/messenger
Fabricante: Microsoft
Protocolos suportados: SIP
Plataforma: Windows
Descrição: Um *softphone* SIP da Microsoft bem conhecido. Existem diferentes versões deste software, nem todos suportam SIP. Ele suporta áudio, vídeo, e *instante messaging*. Ele suporta as técnicas de NAT transversais mas pode ser utilizado atrás de NAT através do servidor SIP na Internet publica porque ele implementa sinalização simétrica no meio.

Nome do produto: kphone

URL do produto: http://www.iptel.org/products/kphone/

Fabricante: Wirlab (Open Source)

Protocolos suportados: SIP

Plataforma: Unix/Linux

Screenshot:

Descrição: Um *User agent* para linux. As versões até 4.0 são ligados ao KDE, as versões acima não necessitam das livrarias KDE e utiliza QT apenas. O telefone suporta presence e instante *messaging* baseada em normas SIMPLE. A conversão de vídeo é suportada por VIC. Este telefone é muito estável no entanto vai abaixo algumas vezes. Ele suporta os codecs G.711, GSM, e iLBC.

Installação:

O *kphone* para *linux* necessita de kde3.1.(http://www.kde.org/info/3.1.php). Este *softphone* é um tal como todos os outros *softphones*, tem muito *bugs.* Entanto é aquele que é utilizado em vário *deliverables* que foram encontrados na Internet.

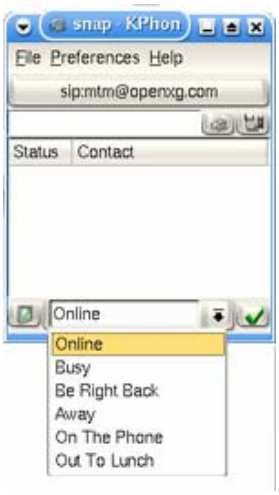


Figura 82 Interface do Kphone

Nome do produto: 6voice

URL do produto: http://www.telscom.ch/6voice/index.htm

Fabricante: telscom

Protocolos suportados: SIP

Plataforma: Unix/Linux

Screenshot: Não disponível

Descrição: Consegue transmitir voz sobre uma rede IPv6, implementando SIP e RTP. Funciona em linux e necessita do JAVA (Java Runtime Environment e Java Media Framework).

Nome do produto: Linphone

URL do produto: http://www.linphone.org

Fabricante: Simon Morlat (Open Source)

Protocolos suportados: SIP

Plataforma: Unix/Linux

Screenshot:

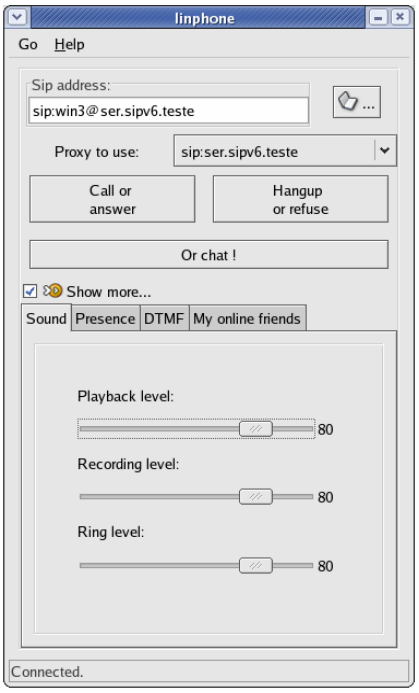


Figura 83 Interface do Linphone

Descrição: Um user agent para linux simples, utiliza frameworks GNOME. O telefone é muito instável e vai abaixo às vezes. O Linphone, é um SIP phone que permite a realização de chamadas VoIP, e *instant messaging*.

Permite também ter uma linha de comando, programa chamado linphonec.

Suporta muitos codecs (G711-ulaw, G711-alaw, LPC10-15, GSM, SPEEX and iLBC), graças ao codec Speex. Suporta toques DTMF pelo o suporte do [RFC2833] e ENUM, ter números SIP em vez de URI. Linhphone é free software, sobre o General Public Licence

Suporte para IPV6 : Versão que funciona para IPv6 Linphone 1.2.0 - Friday December 16, 2005,

Nome do produto: BonePhone
URL do produto: http://www.iptel.org/products/bonephone/
Fabricante: iptel (Open Source)
Protocolos suportados: SIP
Plataforma: linux
Screenshot:

Figura 84 Interface BonePhone v0.1

Descrição: O BonePhone pode suportar várias chamadas ao mesmo tempo. É possível fazer mute/unmute das chamadas que pretende por sinalização SIP. A sinalização SIP e RTP funcionam sobre IPv4, mas também IPv6. O softphone suporta vários perfis de 33.6 Kbit até 1Mbit da largura de disponível. Utilizadores podem definir os seus perfis. O utilizadores podem utilizar endereços manualmente ou de um livro telefónico ou phonebook.

Nome do produto: GomeMeeting

URL do produto: http://www.gnomemeeting.org/

Fabricante: http://www.ingi.ucl.ac.be/

Protocolos suportados: H.323, SIP (mais recentemente) e muitos codecs

Plataforma: linux e windows

Screenshot:

Figura 85 Interface do GnomeMeeting

Descrição: GnomeMeeting é compatível com H.323 e é uma aplicação para videoconferência e VoIP/telefonia IP, para fazer chamadas de voz e vídeo para utilizadores remotos com H.323. Suporta todas as características de videoconferência, como registar numa directoria ILS, suporta para gatekeeper, realizar chamadas multi-utilizador utilizando um MCU externo, e chamadas de PC a PC. Compatível com outros telefones IP como *MyPhone*, *Netmeeting*, *CuSeeMe*, *OpenPhone*, *SJPhone*, *SwissVoice*. Este software irá suportar SIP. Não compatível com Skype, e nunca será uma vez que é proprietário. A versão GnomeMeeting 1.2.1 está disponível e suporta IPv6. A versão SIP, não existem pacotes apenas para o Mandrake 10.1, mas o source já está disponível.

Nome do produto: SIP COMMUNICATOR

URL do produto: https://sip-communicator.dev.java.net/

Fabricante: http://www.ingi.ucl.ac.be/

Protocolos suportados: SIP

Plataforma: Windows XP e RedHat 8.0 Kernel-2.5.65

Screenshot:

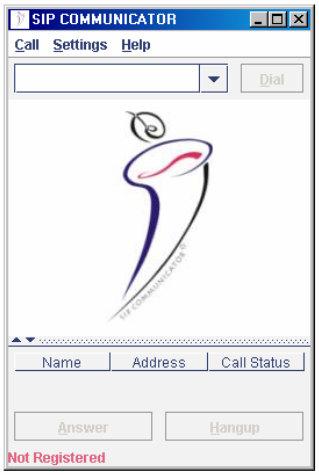


Figura 86 Interface GUI do SIP Communicator

A versão antiga está disponível no seguinte link:

https://sip-communicator.dev.java.net/index_old.html

A nova versão irá sair em breve irá poderá ter a seguinte interface gráfica da figura seguinte, o código fonte está disponível em http://sip-communicator.org/

Figura 87 Interface do SIPCommunicator nova versão

Descrição: O SIP COMMUNICATOR é um SIP User Agent com capacidade de áudio/vídeo escrito em baseado em JAVA, sobre a API JAIN-SIP-1.1 e Java Media Framework. Esta aplicação foi testado com sucesso com Microsoft Messenger, Ubiquity's Helmsman User Agent, e NIST's SIP proxy, e registrar server. Existe um problema com RTP com os kernels do linux 2.4.x e sessões caiem.

Aplicação suporta Vídeo, áudio e instant messaging (na nova versão suporta SIP/SIMPLE e ICQ/AIM). A seguir é mostrado um excerto de código que prova que o Sip-communicator suporta IPv6.

```
/**
 * Determines whether the address could be used in a VoIP session. Attention,
 * routable address as determined by this method are not only globally routable
 * addresses in the general sense of the term. Link local addresses such as
 * 192.168.x.x or fe80::xxxx are also considered usable.
 * @param address the address to test.
 * @return true if the address could be used in a VoIP session.
 */
public static boolean isRoutable(InetAddress address)
{
    if(address instanceof Inet6Address)
    {
        return !address.isLoopbackAddress();
    }
    else
    {
        return (!address.isLoopbackAddress())
                && (!isWindowsAutoConfiguredIPv4Address(address));
    }
}
```

Figura 88 Excerto de código do SIP-Communicator para IPv6 na versão antiga

No seguinte site japonês (http://www.free-sip.com/) fala sobre a implementação do SIP e a utilização do SIP-Communicator para Vídeo em IPv6. Para além de ser necessário alterar os parâmetros como a indicação do SER e outros parâmetros no wizard do Sip-communicator, é necessário alterar no fim do ficheiro XML, sip-communicator.xml, desactivar o IPv4 e activar o IPv6 com as seguintes linhas de código.

```
<java>
    <net>
        <preferIPv4Stack system="false" value="false"/>
        <preferIPv4Addresses system="false" value="false"/>
        <preferIPv6Stack system="true" value="true"/>
        <preferIPv6Addresses system="true" value="true"/>
    </net>
</java>
```

Figura 89 Configuração so SIP-Communicator para IPv6

Nome do produto: SIPSET

URL do produto: http://www.vovida.org/applications/downloads/sipset/

Fabricante: Vovida.org (Open Source)

Protocolos suportados: SIP

Plataforma:

Screenshot:

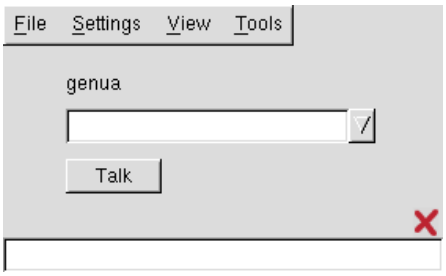


Figura 90 Interface do VOCAL SIPSET

Descrição: SIPSet is a SIP User Agent with a GUI front end that works with the Vovida SIP stack. You can use the SIPSet as a soft phone, to make and receives phone calls from your Linux PC.

This release implements these new features and functionality:

- SIPSet can make calls through a SIP proxy.
- SIPSet can register to receive calls through a SIP proxy.
- SIPSet can make and receive calls directly with another User Agent.

Configuração: http://www.vovida.org/document/man/sipset.html

8.3.2. Hardphones

Nome do produto: Freebit IPv6/v4 hardware phone

URL do produto: http://www.freebit.com/english/officeone/

Fabricante: FreeBit Co., Ltd.

Protocolos suportados: SIP

Imagem do telefone:

Figura 91 Telefones IPv4/IPv6 Freebit

Descrição:

É um hardphone que suporta SIP e IPv6, com capacidade de funcionar como um terminal IP-PBX como call transfer / forwarding / park / pickup, remote update e muito mais.

Este telefone é produzido por uma empresa Japonesa. O Japão está aposta fortemente no IPv6 o governo investiu no IPv6, e tem projecto como IPv6style http://www.ipv6style.jp/en/index.shtml. Existem cerca de 20 000 telefone em cerca de 300 localidades implementados. (ver notícia de 2004, http://www.freebit.com/english/press/pr2004/20040609.html)

O governo impôs que todos os sectores na área da tecnologia da informação corressem em IPv6 até 2005 (http://www.the3gportal.com/members/archives/003209.php).

Nome do produto: Stonehenge IP250

URL do produto: http://www.moimstone.com/Product/product.html

Fabricante: Moimstone

Protocolos suportados: SIP

Imagem do telefone:

Figura 92 Telefones IPv4/IPv6 Stonehenge

Descrição: Stonehenge IP250 are fully featured business IP Phone that coverged communications capability of data and voice over a single wiring infrastucture.

Network Features

- IP Version - IPv4, IPv6
- Network Parameters - DHCP, Static IP address
- VLAN - IEEE802.1Q
- QoS - IEEE802.1p
- Dos - DiffServ(ToS), Denial Service
- Software Up-grade - TFTP Server

Interfaces

- Ethernet
- Dual switched 10/100 Base-T through RJ-45 interfaces
- Auto detect MDI/MDIX cabling
- Power over Ethernet(PoE)/IEEE802.3af(optional)

Physical Specifications

- Dimensions(HxWxD)-218x165x86mm
  Weight-550g

Referências para outros softphones no entanto são na maior parte para IPv4 :

http://www.iptel.org/info/products/sipphones.php

http://www.voip-info.org/wiki/view/VOIP+Phones

http://www.iptelephony.org/GIP/vendors/client-phones/

8.3.3. Servidores

Nome do produto: SIP Express Router
URL do produto: http://iptel.org/ser
Fabricante: iptel.org (Open Source)
Protocolos suportados: SIP
Plataforma: POSIX

Descrição: O SER ou SIP Expresso Router é um servidor proxy SIP [RFC3621] muito rápido e flexível. Escrito a linguagem C, e consegue 1000 de chamadas por segundo até hardware barato. Um *script* permite controlar o servidor. Tem uma arquitectura modular, dando-o mais funcionalidades. O servidor pode ser optimizado para melhor performance, utilizando vários servidores SIP. Um problema será a configuração, é necessário ter bons conhecimentos do protocolo SIP. O servidor consegue localizar os utilizar, e inicia as sessões, e fazer o encaminhamento das mensagens de instante messaging. Ele é bastante inter operável e garante a compatibilidade com outros fabricantes. Tem conseguido com sucesso, ser inter operável com outros produtos SIP, de outros fabricantes. O SER está disponível no site BerliOS ou da IPTEL. O SER é modular, ou seja permite adicionar novas funcionalidades, conforme necessário. Um módulo é o Ser Web, que é uma interface para o administrador e utilizador, inclui funcionalidades como fazer controlo à admissão, alterar o perfil, ver contactos online, enviar mensagens chat. Poder funcionar como um servidor de registo, proxy ou redirect. Ele suporta SMS gateway, SIMPLE2Jabber gateway, contas e autenticação RADIUS/syslog, monitorização do estado do servidor, segurança FCP, e outros.

Nota: Este servidor suporta IPV6 na versão 0.9.4. (utilizada na implementação)

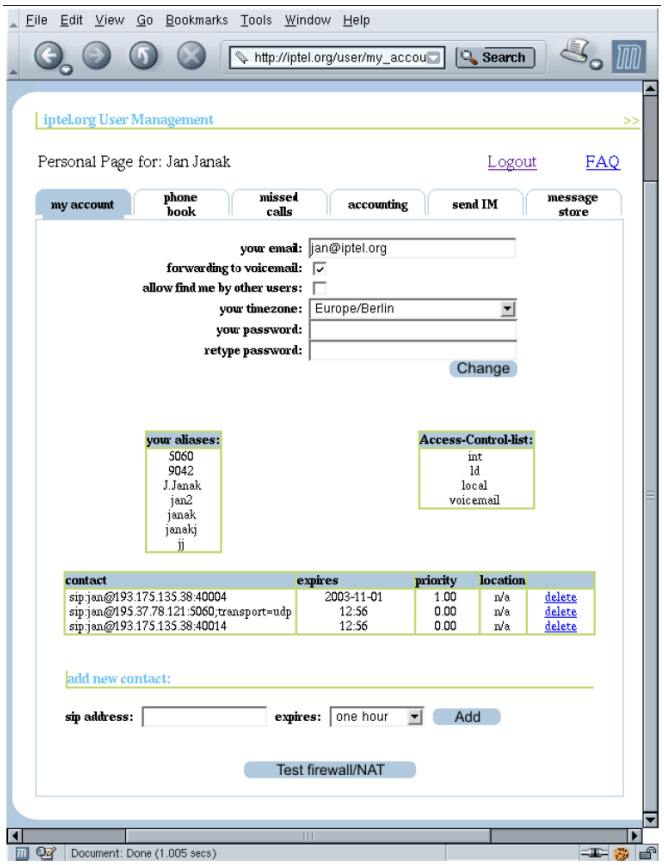


Figura 93 SER da IPTEL com o módulo SER WEB

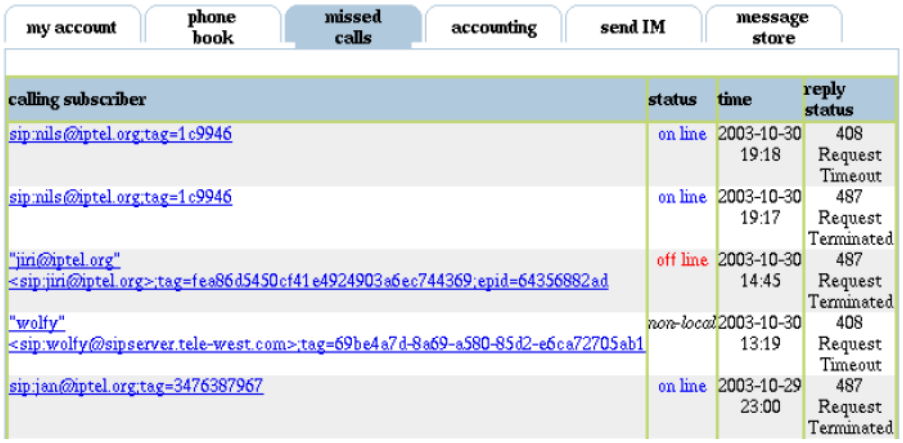


Figura 94 Ver contactos *online*

Exemplo encontrados na implementação do SER WEB:

Existem já muitos servidores SERs implementados, por exemplo o instituto politécnico de Bragança. No projecto "VoIP@IPB-Piloto de telefonia IP numa Instituição de Ensino Superior" Realizado por Nuno Rodrigues, Eduardo Costa, Sérgio do Vale[33]. Outro exemplo é o site japonês free sip[34] que utiliza o SER WEB

[33] http://www.ccom.ipb.pt/voip/modules/voip/show_users.php

Nome do produto: VOCAL (Vovida Open Communication Application Library)
URL do produto: http://www.vovida.org/
Fabricante: Open Source
Protocolos suportados: H.323, SIP, MGCP
Plataforma: Linux

Descrição: O Vovida Open Communication Application Library (VOCAL) é um projecto open source que permite facilitar a integração do VoIP no mercado, de forma gratita.

O software VOCAL inclui o *Session Initiation Procotol* (SIP) *baseado Redirect Server (*RS*), Feature Server (*FS*), Provisioning Server (*PS*), Marshal Server (*MS*) e Voice Mail Server (vmserver).*

Outras aplicações incluídas:

- SIP to MGCP translator 2. Policy server 3. Inet/Conference proxy server. 4. SNMP/NetMgnt
- SIP to H323 Translator It has IPv6 Support and a lot of subsidiary application (from the site http://www.vovida.org). Unfortunately, provisioning currently requires valid IPv4 addresses.

É uma solução para uma empresa média. Necessita de soluções complementares por forma, a ter um sistema distribuído (a gestão deve continuar ser centralizada mas ter a possibilidade de comunicar com outros domínios). O VOCAL oferece aos programadores um software e ferramentas para ajudar o desenvolvimento de serviços e aplicações VoIP.

O sistema VOCAL é uma arquitectura distribuída, consistia por servidores que oferecem serviço de telefonia VoIP. O VOCAL suporta equipamentos que comunicam utilizando o SIP. O VOCAL também suporta telefones analógicos através de *gateways. O VOCAL* suporta chamadas *on-network* e *off-network.* Chamadas do tipo *off-network* permitem os

---

[34] http://free-sip.com/user/index.php

clientes ligarem a dispositivos remotos pela Internet ou pelo o *Public Switched Telephone Network* (PSTN). O software básico VOCAL inclui um conjunto de servidores (*Redirect Server, Feature Server, Provisioning Server and Marshal Proxy Server*). Mesmo que todas as funcionalidades sejam incluídas na versão 1.5.0, o código fonte está disponível no CVS.

Outras funcionalidades:

- SIP to MGCP translator
- Policy server
- Conference proxy server
- SIP to H.323 translator
- JTAPI feature server
- SIP User Agent (substituído pelo o SIPset)
- SNMP/NetMgnt

Suporte para IPv6:

O VOCAL suporta IPv6 na versão VOCAL 1.5.0 alpha2, que é *dual stack* e IPv6-only

Nome do produto: YXA
URL do produto: http://www.stacken.kth.se/projekt/yxa/
Fabricante: stacken
Protocolos suportados: SIP
Plataforma: Unix/Linux
Logótipo:

Descrição:

É um servidor SIP compatível com o [RFC3261], com as seguintes características:

Registar dos utilizadores

Controlar os pedidos SIP para o domínio

Faz encaminhamento dos pedidos dos utilizadores para utilizadores em domínios remotos.

Suporta TCP, UDP e TLS (incluindo SIPS)

Automaticamente mapea endereços de e-mail para endereços SIP, no caso de ter endereços e-mail em LDAP.

Controlar vários domínios num único servidor

Suporta ENUM para PSTN bypass

Suporta IPv6

Forking em paralelo e sequencial

CPL [RFC3880] para controlar eventos dos utilizadore (chamadas a entrar)

Base de dados modular, actualmente com LDA, Mnesia, MySQL, e ficheiros de texto

Controlo de acesso com o destino para PSTN

Nome do produto: OpenH323 Gatekeeper - GnuGK
URL do produto: http://www.gnugk.org
Fabricante: Open Source
Protocolos suportados: H.323
Plataforma: Qualquer, é possível compilar a livaria OpenH323 (Linux,Windows, FreeBSD,Solaris, etc.)
Descrição: Existem alguns problemas com o Netmeeting. O Netmeeting não suporta o Gatekeeper Discovery, assim deve-se configurar directamente o endereço nas opções avançadas opção Calling. O Netmeeting pede uma largura de banda incorrecta, deve-se fazer o disable da largura de banda para evitar problemas com o GnuGK.

Este servidor suporta Radius, MySQL, e LDAP. O manual está escrito em Inglês, chinês, e português. É utilizado em muitas aplicações comerciais e tem uma boa interface gráfica para configurá-lo, monitorizar os registos, e definir grupos de terminais para a gestão de chamadas, etc. É o melhor Gatekeeper em Open source disponível.

Nome do produto: OpenMCU (H.323 *Conferencing Server*)
URL do produto: http://www.openh323.org/code.html
Fabricante: Open Source
Protocolos suportados: H.323
Plataforma: qualquer platforma pode compilar a biblioteca OpenH323 Library (Linux,Windows, FreeBSD, Solaris, etc.)

Descrição: Ele aceita múltiplas ligações em simultânea. Depois dos primeiros quarto clientes ligados, ele aceita aplicações áudio e vídeo, depoi apenas existem para áudio. Ele inicia a chamada do MCU até aos terminais remotos e suporta áudio. O software MCU necessita de uma infra-estrutura de hardware nos termos de partilha de memória utilizada. É possível alterar os parâmetros de compressão de vídeo e qualidade faz com que seja mais escalonável em termos necessidades computacionais. É muito útil para conferência multi-ponto, no entanto não utiliza *multicast*, mas várias ligações *unicasts.*

Nome do produto: AppEngine
URL do produto: http://www.dynamicsoft.com/prod_sol/AppEngine/appenginev4.php
Fabricante: Dynamicsoft
Protocolos suportados: SIP
Plataforma: Sun Solaris
Descrição: Esta aplicação SIP, é escrito em Java. O servidor implementa os Java SIP *servlets* que permitem criar aplicações SIP complexas chamadas *servlets.* Estes *servlets* podem interagir com os *servlets* http também. Tem uma boa documentação e muitos exemplos.

Nome do produto: Cisco IP/VC 3510 MCU
URL do produto: http://www.cisco.com/univercd/cc/td/doc/product/ipvc/ipvc2 2/2 2mcurn.htm
http://www.cisco.com/univercd/cc/td/doc/product/ipvc/
Fabricante: Cisco
Protocolos suportados: H.323
Plataforma: N/A

Descrição: This MCU is an older product, identical to the Radvision OnLan MCU, but OEMed by Cisco and currently not supported any more, as it has been replaced by the 3511 MCU. The 3510 is capable of connecting 12 participants at 384Kbps each, or a combination of conferences at different rates and different numbers of participants.

The default hardware does not provide audio/vídeo transcoding between participants, so conference settings must be matched by all. Continuous presence is supported, but with asymmetric video rates, i.e. each participant sends 384Kbps video but receives 4x384Kbps back from the MCU, for viewing 4 participating sites simultaneously.

Check the IP/VC Products page at
http://www.cisco.com/univercd/cc/td/doc/product/ipvc/ipvc2_2/2_2mcurn.htm

Nome do produto: Cisco MCU gatekeeper

URL do produto: http://www.cisco.com/univercd/cc/td/doc/product/software/ios122/122cgcr/fvvfax_c/vvf323gk.htm
Fabricante: Cisco
Protocolos suportados: H.323
Plataforma: Cisco router - IOS based
Descrição: O Cisco Multimedia Conference Manager (MCM) é o nome do gatekeeper que funciona no IOS Cisco. O MCM suporta o protocolo H.323. O MCM pode funcionar como proxy para cliente H.323 atrás de firewalls. O API que foi desenvolvido pela Cisco, e permite controlar os eventos do gatekeeper por uma aplicação externa. (Cisco Gatekeeper External Interface)

Referência:
http://www.cisco.com/univercd/cc/td/doc/product/software/ios122/rel_docs/gktmp4_1/index.htm

8.3.4. Gateways

Nome do produto: OpenISDN (H.323 Call Generator)
URL do produto: http://www.gae.ucm.es/~openisdngw/home_en.php
122cgcr/fvvfax_c/vvf323gk.htm
Fabricante: Open Source
Protocolos suportados: H.323
Plataforma: Qualquer plataforma onde é possível compilar a biblioteca OpenH323 (Linux,Windows, FreeBSD, Solaris, etc.)
Descrição: Experiência operacional: As placas ISDN têm de ser instalados correctamente e configurados na máquina local de modo a fazer as ligações com o ISDN.

Funciona apenas com linhas ISDN (PSTN não é suportado) para a gestão de chamadas em simultâneo, para que os canais ISDN estejam disponíveis. O Gatekeeper pode ser um endereço IP bem conhecido ou pode descobrir pela a rede fazendo o broadcast RAS. É possível ter uma configuração dinâmica, uma vez que o programa começa no cliente e é configurado utilizando as opções da linha de comando.

É um Gateway H.323/ISDN simples. É necessário mais investigação e desenvolvimento para ser completamente compatível com o H.320 para conferências por ISDN. Neste momento apenas é um gateways de áudio.

Nome do produto: Asterisk Open Source PBX
URL do produto: http://www.asterisk.org
Fabricante: Digium
Protocolos suportados: SIP, H.323, SCCP, MGCP
Plataforma: N/A
Descrição: Asterisk é open source, é uma plataforma que suporta TDM e VoIP PBX e IVR

Permite integrar as tecnologias TDM (T1, PRI, FXS, FXO) e VoIP (IAX, SIP, H.323), num único PBX enquanto que permite a funcionalidade IVR através de uma linguagem de scripting disponível no linux. Existe um servidor chamado Asterisk que permite ter serviços de Voicemail e suporta chamadas de conferência, Resposta interactivas de voz, chamadas em espera. Ele suporta chamadas three-way, serviços de caller ID, ADSI, IAX, SIP, H.323 (como cliente e *gateway*), MGCP (apenas como gestor) e SCCP/Skinny.

8.3.5. Testes

Nome do produto: CallGen323 (H.323 Call Generator)
URL do produto: http://www.openh323.org/code.html
Fabricante: Open Source
Protocolos suportados: H.323
Plataforma: Qualquer (necessário compilar a biblioteca OpenH323)
Descrição: Não é muito útil uma vez que os servidores têm de ser testados sob stress. Uma das vantagens é a configuração estática, não tem gestão dinâmica e tem suporte limitado.

Nome do produto: Sipsak
URL do produto: http://sipsak.berlios.de
Fabricante: iptel.org (Open Source)
Protocolos suportados: SIP
Plataforma: N/A
Descrição: Uma aplicação de diagnóstico e *Stress Utility* (gerado de tráfego), o Sipsak é uma aplicação simples utilizado para testar as várias funções do servidor SIP. Ele inclui testes de proxy, registos, e autenticação *digest.* Ele também consegue gerar mensagens SIP para testar o servidor.

Nome do produto: SIPStone
URL do produto: http://www.sipstone.org
Fabricante: Columbia University and Ubiquity
Protocolos suportados: SIP
Plataforma: N/A
Descrição: Actualmente esta aplicação é apenas na fase daft, e serve para medir o performance do SIP. Esta ferramenta de medição está disponível da universidade de Columbia (ver o Lins seguinte)

http://www.cs.columbia.edu/IRT/cinema/sipstone/
outro:
http://www.softfront.co.jp/en/spp/test/testserver.html
http://www.softfront.co.jp/en/spp/sip/ua/index.html

8.4. Configurações no CallManager 3.1[35]

No arranque do CallManager é importante que tenha rede, senão ele não terá um funcionamento correcto, basta ligá-lo a um switch. O CallManager 3.1 se estiver ligado à rede poderá ser acedido pelo URL: http://CALLMANAGER/admin

O CallManager tem duas placas de rede, verificar qual está na rede 10.0.0.0/8-se Para verificar utiliza. O comando ipconfig, qual o ip do CallManager. A página principal do CallManager tem o seguinte aspecto:

Figura 95 Página principal de administração do CallManager 3.1

Para o callManager 3.1, a documentação existe no link:

www.cisco.com/univercd/cc/td/doc/product/voice/c_callmg/3_1/sys_ad/3_1_2/ccmcfg/bccm.pdf

e a versão online é:

http://www.cisco.com/univercd/cc/td/doc/product/voice/c_callmg/3_1/index.htm

Na própria página de administração do CallManager também tem a ajuda ou Help.

Nota: Todas as configurações no callManager são armazenadas na base de dados (Microsoft SQL Server 7.0) do callManager.

[35] Para outras configurações consultar http://www.cisco.com/univercd/cc/td/doc/product/voice/c_callmg/3_1/qs_31.htm
Ou a ajuda do CallManager

8.4.1. Subscrição de um User e configuração do telefone

Até agora o serviço apenas fica disponível para utilizadores subscritos. O administrador do callManager depois deve fornecer o URL para os utilizadores a modos de poder configurar os serviços que pretendem. http://10.0.0.2/CCMUser/logon.asp ou http://CALLMANAGER/CCMUser/logon.asp

Figura 96 página public onde os *Users* ou utilizadores podem aceder

Os utilizadores que entram nesta página (Cisco IP Phone User Options) são autenticados através da directory LDAP.

NOTA: Se não saber o *login* e *password* de um User, cria um novo utilizador em Add new user. Este utilizador tem que estar associado ao IP Phone 7960 ou 7910.

8.4.2. Adicionar um telefone IP Phone 7960

Procedimento

- Escolher no menu **Device** a opção **Add** a **Device**.
    - Aparece a página Add a New Device
- Escolher a opção Phone da drop-down list box, e carregar em Next

- Escolher o Phone type que se pretende (7960, 7960 ou softphone(CTI Port))
    - Nota uma vez escolhida o tipo não é possível modificá-la
- Aparece depois a página chamada Phone Configuration.
- Preencher os atributos endereço MAC o telefone 7960 (00075079C2B7) e o 7910 (0007EB6A96D7) carregar **Insert**.
- Aparece depois uma mensagem a dizer que o telefone foi adicionado na base de dados.
- Coloca-se depois o *directory number* ou seja o numero de telefone.

8.4.3. Criação de *Users* ou utilizadores

Procedimento

- Escolher na página de admin. do CM User > Add a New User.
- Preenchido estes campos.
- Guardar as modificações e adicionar o utilizador carregando Insert.

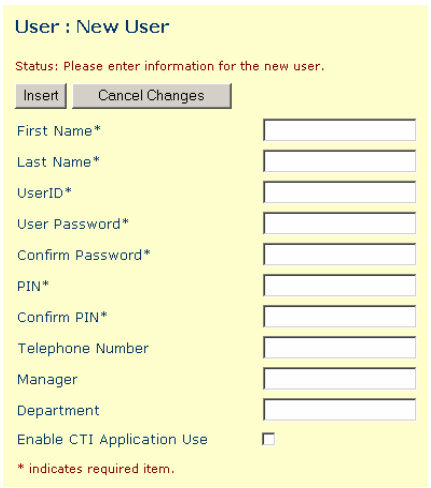


Figura 97 Adicionar um novo utilizador

8.4.4. Associar *Users* aos telefones 7960, 7910

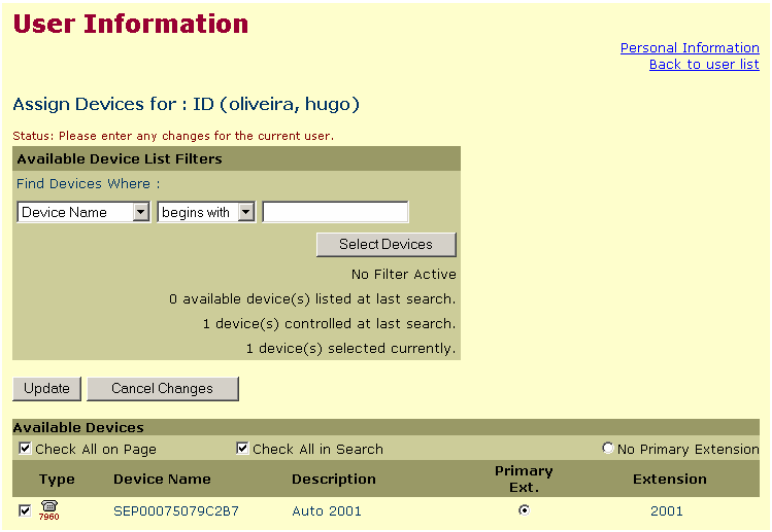


Figura 98 O User com o ID ID deve está associado ao telefone 7960.

8.4.5. Actualizar o telefone

Sempre que for feita alguma alteração no callManager que altere as configurações do telefone é necessário actualiza-lo.

Para efectuar a actualização do telefone segue-se o seguinte Procedimento

- Escolher na página admin do CM **Device > Phone**.O painel Find and List Phones aparece.
- Inserir o nome de determinado telefone e pressionar em **Find**.A lista de telefones que vão ao encontro do nome da pesquisa anterior aparece.
- Da lista, escolher o nome do telefone que quer fazer a actualização.O painel das configurações do telefone aparece.
- Actualize os campos apropriadas
- Carregue em **Update**.
- Carregue em **Reset Phone** para reiniciar ou fazer o novamente o registo ao CM do telefone aplicando as novas definições. Carregando em “Restart” volta a registar o telefone com o Cisco CallManager sem desligar o telefone. Fazer o “reset”(reiniciar) ao telefone desliga e volta a iniciar o telefone fazendo o registo ao CallManager.

- Reiniciar o telefone

É necessário efectuar o "reset" do telefone IP da Cisco depois de introduzir um directory number ou depois de actualizar as definições para que as alterações tenham efeito. Para reiniciar o telefone seguir o seguinte procedimento.

Nota: Se uma chamada está em curso o telefone não faz reinicia enquanto a chamada não termina.

Procedimento

- Escolha **Device > Phone**. O painel Find and List Phones aparece.
- Inserir o nome de determinado telefone e pressionar em **Find**. A lista de telefones que vão ao encontro do nome da pesquisa anterior aparece.

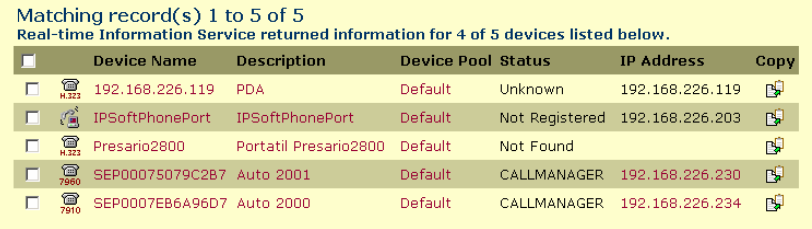


Figura 99 Painel de procura e listagem dos telefones

- Seleccionar as caixas de escolha dos telefones que é para reiniciar. Para seleccionar todos os telefones do painel selecciona a caixa que está na barra dos títulos dos campos.
- Carregar em Reset Selected. O painel de reiniciar o telefone aparece.
- Carregue nos seguintes campos.
    - **Restart Device**—Volta a registar o telefone ao CM com as novas alterações (não desliga o telefone).
    - **Reset Device**—Reinicia o telefone desligando-o e registando novamente ao CM .

8.4.6. Configuração de speed dials

Nota: Esta configuração não é necessária para correr os serviços. É apenas uma curiosidade. É possível configurar speed dials. Os speed dials são números de telefone que um utilizador utiliza mais vezes. Eles permitem ligar rapidamente para o número de destino com o toque de um botão.

- Basta entrar na página

http://10.0.0.2/CCMuser/logon.asp

- Fazer o logon com o login e a password do utilizador.

- Depois carrega-se no link **Add/Update your Speed dials** como mostra a figura seguinte

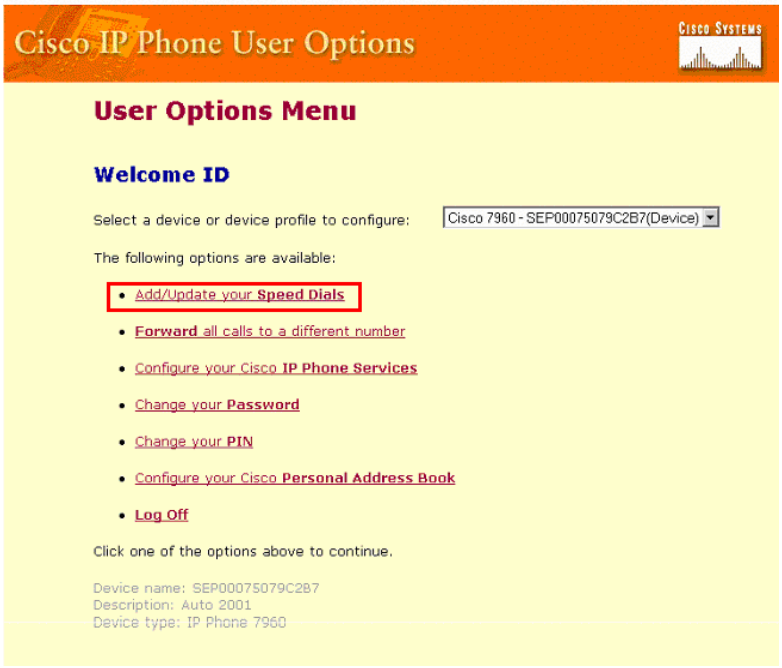


Figura 100 Página principal onde também é possível adicionar speed dials

- Preencher os campos **Speed dial 1** por exemplo e fazer Update

Figura 101 Adição de um speed dial para o telefone IP 7910

No telefone o *speed dial* aparece da seguinte forma como indicado na figura seguinte. O utilizador só tem que carregar no botão para estabelecer a chamada.

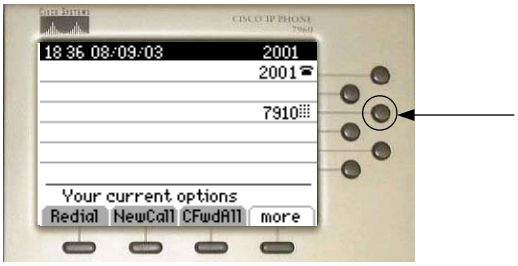


Figura 102 *Speed dial* criado no telefone 7960

8.4.7. Configurações de Rede nos telefones IP da Cisco

No telefone não há configurações a fazer, normalmente o IP é atribuído automática pelo o callManager através de DHCP. Ao carregar o botão **Settings** e depois o menuItem **Network configuration** é possível ver os atributos de configuração por exemplo o MAC, o IP, e outros. Normalmente aparece um pequeno cadeado fechado ao lado do título onde diz Network Configuration indicando que está bloqueado para evitar alterações incorrectas ou acidentais.

Para desbloquear é necessário carregar a sequência **# no *keypad*. Depois das alterações realizadas é necessário carregar no softkey Save, depois Exit. Para reiniciar o telefone basta escrever a sequência no keypad **#**

Sempre que existem problemas é sempre possível carregar as configurações por defeito "factory defaults" no menuItem Network configuration.

8.5. Nova versão do CallManager

Já existe o CallManager versão 5.0, mas ainda não está oficialmente lançado, mas está para breve segundo o Product Manager da Cisco. A versão Beta do Call Manager saiu por um período de tempo http://www.cisco.com/cgi-bin/tablebuild.pl/callmgr-beta mas já não está disponível.

Figura 103 CallManager 5.0 (alguém que consegui instalar)

A documentação irá fica no seguinte URL:
http://www.cisco.com/univercd/cc/td/doc/product/voice/c_callmg/5_0/
ainda não existe documentação sobre o CallManager 5.0, mas não irá suportar IPv6, segundo o Product Manager da Cisco, e que irá estar brevemente disponível, segundo o que eles dizem…

8.6. Listagem dos tipos de mensagens Skinny [9]

Code	Station Message ID Message
0x0000	Keep Alive Message
0x0001	Station Register Message
0x0002	Station IP Port Message
0x0003	Station Key Pad Button Message
0x0004	Station Enbloc Call Message
0x0005	Station Stimulus Message
0x0006	Station Off Hook Message
0x0007	Station On Hook Message
0x0008	Station Hook Flash Message
0x0009	Station Forward Status Request Message
0x11	Station Media Port List Message
0x000A	Station Speed Dial Status Request Message
0x000B	Station Line Status Request Message
0x000C	Station Configuration Status Request Message
0x000D	Station Time Date Request Message
0x000E	Station Button Template Request Message
0x000F	Station Version Request Message
0x0010	Station Capabilities Response Message
0x0012	Station Server Request Message
0x0020	Station Alarm Message
0x0021	Station Multicast Media Reception Ack Message
0x0024	Station Off Hook With Calling Party Number Message
0x22	Station Open Receive Channel Ack Message
0x23	Station Connection Statistics Response Message
0x25	Station Soft Key Template Request Message
0x26	Station Soft Key Set Request Message
0x27	Station Soft Key Event Message
0x28	Station Unregister Message
0x0081	Station Keep Alive Message
0x0082	Station Start Tone Message
0x0083	Station Stop Tone Message
0x0085	Station Set Ringer Message
0x0086	Station Set Lamp Message
0x0087	Station Set Hook Flash Detect Message
0x0088	Station Set Speaker Mode Message
0x0089	Station Set Microphone Mode Message
0x008A	Station Start Media Transmission
0x008B	Station Stop Media Transmission
0x008F	Station Call Information Message
0x009D	Station Register Reject Message
0x009F	Station Reset Message

0x0090	Station Forward Status Message
0x0091	Station Speed Dial Status Message
0x0092	Station Line Status Message
0x0093	Station Configuration Status Message
0x0094	Station Define Time & Date Message
0x0095	Station Start Session Transmission Message
0x0096	Station Stop Session Transmission Message
0x0097	Station Button Template Message
0x0098	Station Version Message
0x0099	Station Display Text Message
0x009A	Station Clear Display Message
0x009B	Station Capabilities Request Message
0x009C	Station Enunciator Command Message
0x009E	Station Server Respond Message
0x0101	Station Start Multicast Media Reception Message
0x0102	Station Start Multicast Media Transmission Message
0x0103	Station Stop Multicast Media Reception Message
0x0104	Station Stop Multicast Media Transmission Message
0x105	Station Open Receive Channel Message
0x0106	Station Close Receive Channel Message
0x107	Station Connection Statistics Request Message
0x0108	Station Soft Key Template Respond Message
0x109	Station Soft Key Set Respond Message
0x0110	Station Select Soft Keys Message
0x0111	Station Call State Message
0x0112	Station Display Prompt Message
0x0113	Station Clear Prompt Message
0x0114	Station Display Notify Message
0x0115	Station Clear Notify Message
0x0116	Station Activate Call Plane Message
0x0117	Station Deactivate Call Plane Message
0x118	Station Unregister Ack Message

Tabela 14 Listagem dos tipos de mensagens Skinny

8.7. Lista de Codecs para o telefone analógico

g711alaw—G.711 A Law 64,000 bps

g711ulaw—G.711 u Law 64,000 bps

g723ar53—G.723.1 ANNEX-A 5,300 bps

g723ar63—G.723.1 ANNEX-A 6,300 bps

g723r53—G.723.1 5,300 bps

g723r63—G.723.1 6,300 bps

g726r16—G.726 16,000 bps

g726r24—G.726 24,000 bps

g726r32—G.726 32,000 bps

g728—G.728 16,000 bps

g729abr8—G.729 ANNEX-A & B 8,000 bps

g729br8—G.729 ANNEX-B 8,000 bps

g729r8—G.729 8000 bps

gsmefr—Global System for Mobile Communications Enhanced Full Rate (GSMEFR) 12,200 bps

gsmfr—Global System for Mobile Communications (GSM) Full Rate (GSMFR) 13,200 bps

transparent—Enables codec capabilities to be passed transparently between endpoints.

8.8. Descrição do Protocolo GRE Cabeçalho

O protocolo GRE tem duas versões. A versão 0 e a ver

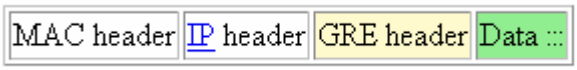


Figura 104 Frame GRE

Cabeçalho GRE, versão 0

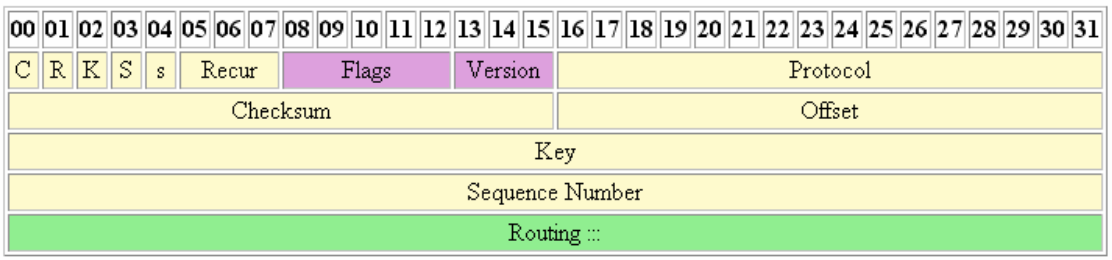


Figura 105 Cabeçalho do GRE

C, Checksum Present. 1 bit.

O campo *Checksum* está presente e contém informação válida (se a 1). Se o bit *Checksum Present ou o bit Routing Present estão a 1, os campos Checksum e Offset estão ambos presents.*

R, Routing Present. 1 bit.
Se o campo Offset está presente e contém informação válida (se a 1). Se o bit *Checksum Present* ou o bit *Routing Present* estão a 1, os campos *Checksum* e Offset estão ambos presentes.

K, Key Present. 1 bit.

Se o campo *Key* está presente e contém informação válida.

S, Sequence Number present. 1 bit.

Se está a true o campo *Sequence Number* está presente e contém informação válida.

s, Strict Source Route. 1 bit.
Este bit server está definido em outros documentos (RFCs). É recomendado que este bit atribuído quando toda a informação de *Routing* consistem em rotas *Strict Source Routes.*

Recur, Recursion Control. 3 bits, unsigned integer.
Contém o número de encapsulamentos adicionais que são permitidos. Zero é o valor por defeito.

Flags. 5 bits.
Estes bits estão reservados se devem ser transmitidos com o valor zero.

Version. 3 bits.
GRE protocol version. Deve ter o valor de zero.

Protocol. 16 bits.
Contém o tipo de protocolo que está no *payload* do pacote. Em geral, o valor irá ser o campo tipo de protocolo *Ethernet.*

Checksum. 16 bits.
Opcional. Tem *checksum* o cabeçalho GRE e o *payload* do pacote.

Offset. 16 bits.
Opcional. Indica o offset do byte desde do início do campo *Routing* até o primeiro byte do *Source Route Entry.*

Key. 32 bits.
Opcional. Contém o número que irá ser inserido pelo o *encapsulator*. Ele pode ser utilizado pelo o receptor para autenticar a origem do pacote.

Sequence Number. 32 bits, unsigned integer.
Opcional. Contém o número que é inserido pelo o *encapsulator.* Ele pode ser utilizado pelo receptor e pode estabelecer a ordem pela a qual os pacotes são transmitidos desde do *encapsulator* até ao receptor.

Routing. Variable length.
Opcional. Este campo está na lista do SREs.

SRE, O pacote Source Route Entry

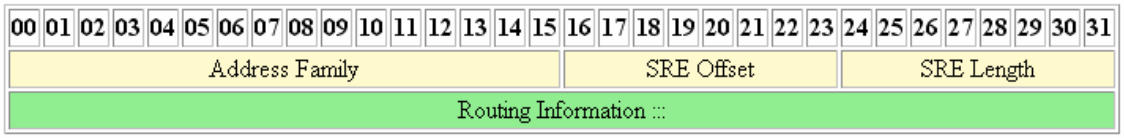


Figura 106 Route Entry

Address Family. 16 bits.
Indica a sintaxe e a semântica do campo *Routing Information*.

SRE Offset. 8 bits.

Indica offset do byte desde do começo do campo *Routing Information* até o primeiro bytes da entrada activa.

SRE Length. 8 bits.
O número de bytes no SRE. Se for zero, indica que é o último SRE.

Routing Information. Variável

Contém dados que podem ser utilizados, no routing deste pacote.

O GRE *header* ou cabeçalho tem o seguinte formato.

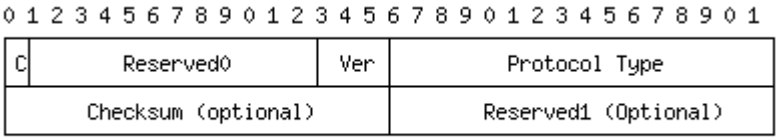


Figura 107 campos do GRE

Campos:

Checksum Present (bit 0)

Este campo indica se há ou não *Checksum*, se sim, então os campos *Checksum* e Reserved1 têm de conter informação válida.

Reserved0 (bits 1-12)

O receptor deve descartar o pacote onde qualquer dos bits 1-5 são diferentes de zero, a não ser que o receptor implementa o [RFC1701]. Os bits 6-12 estão reservados para o futuro.

Version Number (bits 13-15)

O campo sobre o número da versão.

Protocol Type (2 octets)

Tipo de Protocolo que existe no payload do pacote. Estes tipos de protocolos estão definidos no [RFC1700] como os "ETHER TYPES" ou [ETYPES]. An implementation

receiving a packet containing a Protocol Type which is not listed in [RFC1700] or [ETYPES] SHOULD discard the packet.

Checksum (2 octets)

O campo checksum contém o IP, e o checksum com a soma de todas as palavras de16 bits no cabeçalho GRE e do payload do pacote.

Reserved1 (2 octets)

O campo Reserved1 está reservado para o futuro, e no caso de estar presented eve ser transmitido a zero.

IPv4 as a Payload

Quando IPv4 é transportado no payload do GRE, o campo Protocol Type deve ter o valor 0x800.

Implementação em linux vistar o seguinte *url*:
http://www.tldp.org/HOWTO/Adv-Routing-HOWTO/lartc.tunnel.gre.html

8.9. Cabeçalhos do protocolo RTP

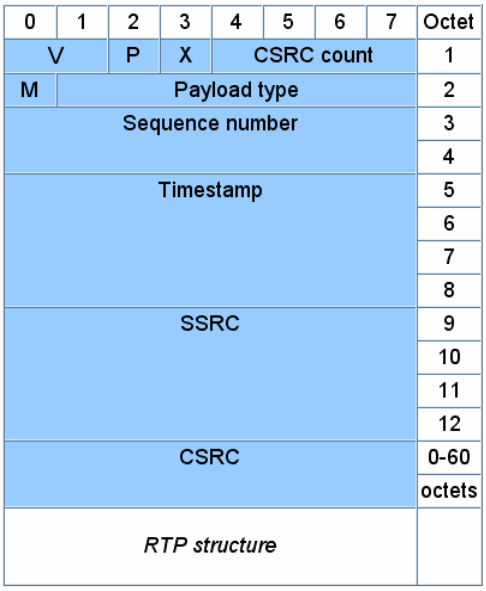


Figura 108 Estrutura do protocolo RTP

V→Version. Identifica a versão do RTP.

P→Padding. Quando atribuído, o pacote contém mais do que um octeto de *padding* que não pertence ao *payload*.

X→Extension bit. Quando atribuído, o cabeçalho é fixo e é seguido de apenas um *extension header* com um formato bem definido.

CSRC→Count, Contém o número de identificadores CSRC que seguem o cabeçalho fixo.

M→Marker. A interpretação do marcador é definida por um perfil. Tem como intenção permitir que eventos significativos como os limites da *frame*, para serem marcados na *stream* de pacotes.

Payload type→Identifica o formato do *payload* do RTP para a aplicação interpretar.

Sequence number→Incrementa por um para cada pacote de dados RTP enviada, e pode ser utilizado pelo o receptor para detectar a perda de pacote e para restaurar a sequência.

Timestamp→Reflecte o instante de amostragem do primeiro octeto do pacote de dados RTP. O instante de amostragem pode ser derivado do relógio que incrementa metodicamente e linearmente a tempo de permitir sincronização e os cálculos de *jitter*.

A resolução do relógio deve ser o suficiente para obter com rigor a sincronização e para medir o *jitter* do pacote à chegada.

SSRC→Identifica o sincronizador. Este identificador é escolhido aleatoriamente, com a intenção de sincronizar duas origens, a mesma sessão RTP irá ter o mesmo identificador SSRC.

CSRC→Contributing source identifiers list. Identifica os contribuidores para o *payload* contido no pacote.

8.10. Mensagens Skinny

8.10.1. Processo de registo do telefone

O telefone irá buscar os seguintes ficheiros:

- **OS79XX.TXT**
- **SEP00075079C2B7.cnf.xml**
- **Classic1.raw**

Configuração no router do cenário da

```
Interface fastethernet0/0
ip address 10.0.0.1
no shudown

dial-peer voice 100 pots
destination-pattern 100
translate-outgoing called 100
port 1/0/1
```

Nota: o comando **translate-outgoing called 100 (translation profiles)** serve para mostrar no telefone IP aparecer o CallerID, ou o número de telefone de origem[36]. Como o CallManager está directamente ligado ao router, não é necessário o comando **session target ipv4:10.0.0.2**

8.10.2. Processo de registo

O telefone primeiro necessita de um endereço IP, portanto irá fazer o pedido DHCP Request depois ter o endereço atribuído irá buscar ficheiros de configuração que necessita por meio de TFTP.

Entre as mensagens TFTP são trocadas também mensagens de Skinny para configurar o telefone como por exemplo os *speedials.*

Dynamic Host Configuration Protocol (DHCP)

[36] http://www.cisco.com/en/US/tech/tk652/tk90/technologies_configuration_example09186a00803f818a.shtml

O DHCP é utilizado pelos os *hosts* da rede para buscar a informação inicial, incluindo o endereço IP, a mascara, o *default gateway*, e o endereço IP do servidor TFTP. Os terminais da telefonia IP utilizam a opção 150 (DHCP option 150) para identificar a origem da informação de configuração, disponível no servidor TFTP (*Trivial File Transfer Protocol)*

contact the secondary TFTP server. A Cisco recomenda utilizar directamente o endereço IP (sem ser necessário recorrer ao serviço DNS) para a opção 150 (comando Option 150), assim elimina a dependência do serviço DNS, que o telefone inicia o processo de registo

8.10.2.1. Implementação do DHCP

No caso deste cenário não é necessário uma configuração nos routers de DHCP, apenas no cenário da Figura 40 Implementação prática de um túnel GRE.

Existem duas opções para implementar DHCP numa rede de telefonia.

- **Um servidor de DHCP centralizado**→Tipicamente deve ser instalado um servidor DHCP num local centralizado na rede. Deve existir servidores redundantes. Este tipo de implementação necessita da configuração do comando **ip helper-address** na *fastethernet* do router onde está ligado o terminal ou DHCP *client.* Por defeito o comando **service dhcp** no IOS da Cisco e não aparece na configuração. Não se deve desactivar o serviço, uma vez que o comando ip helper-address não irá funcionar.
- **Servidor DHCP centralizado e um router remoto com o servidor DHCP no IOS da Cisco**→Este tipo de implementação garantem que os serviços de DHCP ficam disponíveis, mesmo que a WAN (neste caso o *backbone* IPv6)

Exemplo da configuração do servidor DHCP no IOS da Cisco:

```
service dhcp
! Activar o service DHCP no IOS
ip dhcp excluded-address <ip-address>|<ip-address-low> <ip-address-high>
! Especifica a gama de endereços a serem excluídos do DHCP pool
ip dhcp pool <dhcp-pool name>
! Especifica a nome da pool DHCP
network <ip-subnet> <mask>
```

```
! Especifica a pool DHCP a subrede e a mascara
default-router <default-gateway-ip>
! específica o Default-gateway
option 150 ip <tftp-server-ip-1> .
! Especifica o servidor TFTP até quatro
```

8.10.2.2. Trivial File Transfer Protocol (TFTP)

Com o Cisco CallManager CCM, os terminais (como os telefones que utilizam o protocolo SCCP) dependem do processo de TFTP para obter a informação de configuração. Os terminais fazem o pedido do ficheiro de configuração, cujo nome é baseado no endereço MAC de quem faz o pedido (por exemplo, para o telefone IP com o endereço MAC **ABCDEF123456**, o ficheiro irá ter o nome **SEPABCDEF123456.cnf.xml**). O ficheiro de configuração inclui a versão de software que o telefone deve ocorrer e uma lista de servidores CCM que o telefone pode registar. Se o ficheiro de configuração indica ao telefone para utilizar outro ficheiro de software então o telefone irá pedir a nova versão de software do servidor TFTP. O telefone passa então pelo o processo de actualização do software.

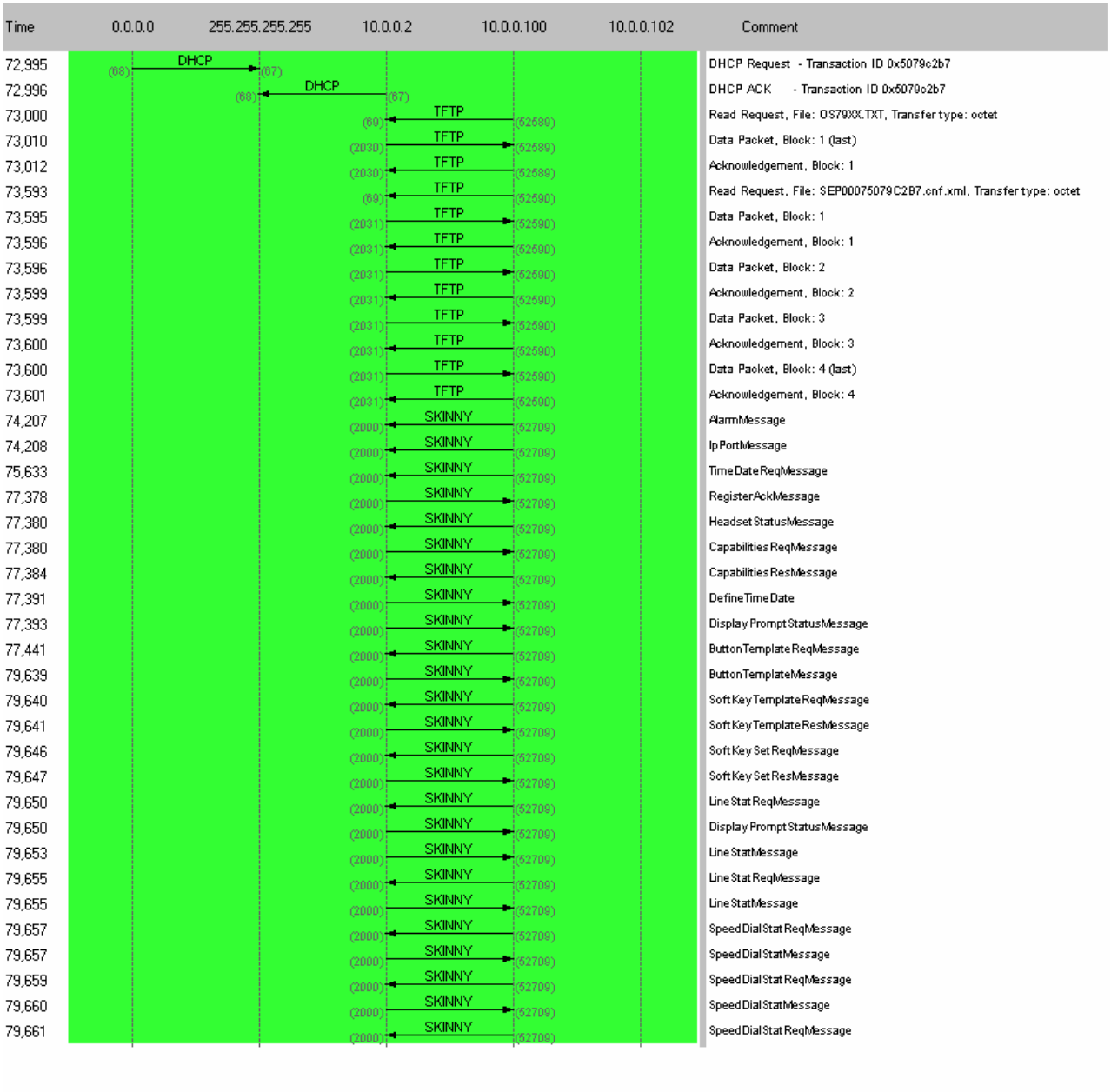


Figura 109 Registo do telefone IP no CCM

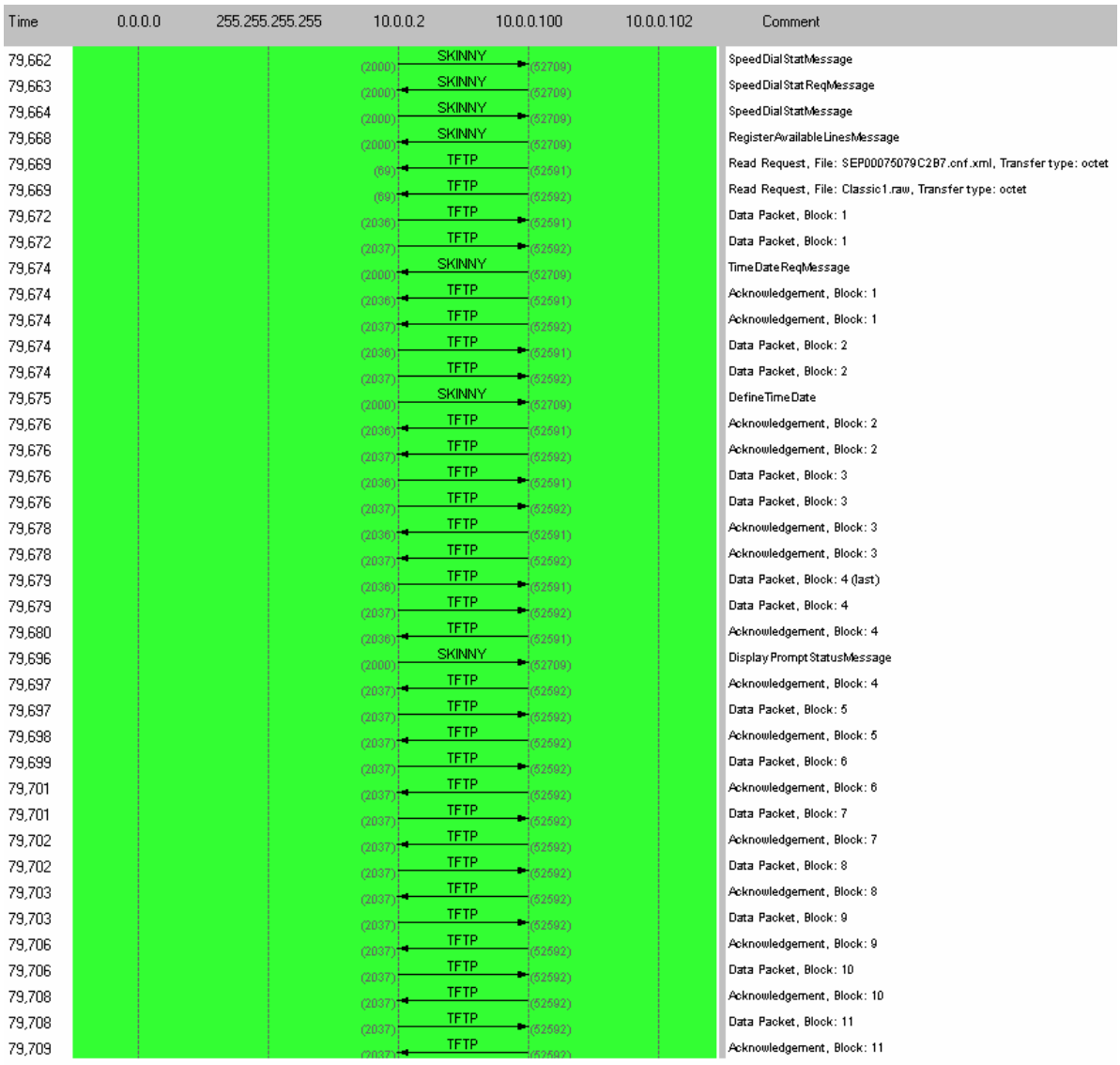


Figura 110 Registo do telefone IP no CCM (cont.1)

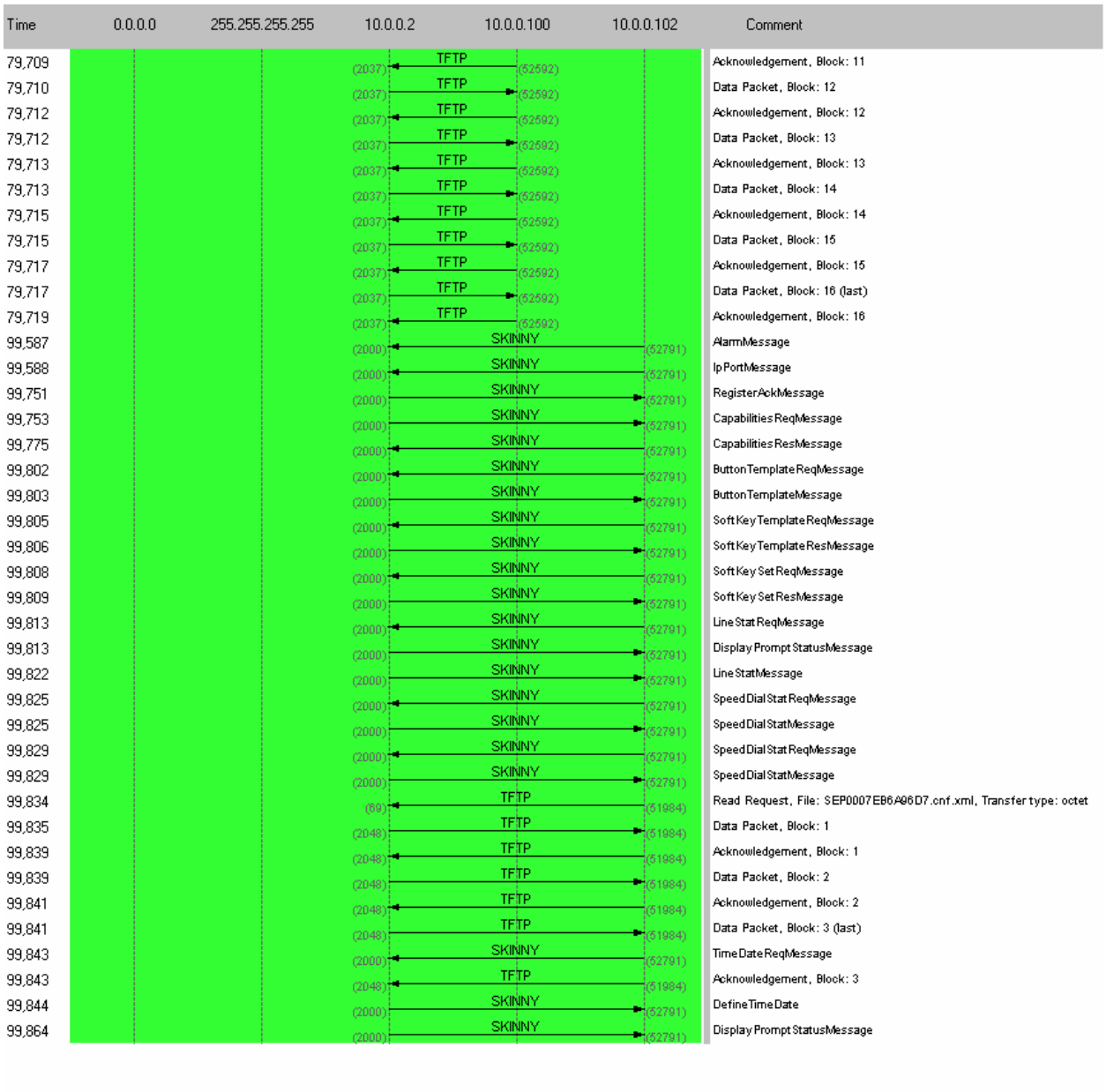


Figura 111 Registo do telefone IP no CCM (cont.2)

(tftp or skinny or bootp or arp and ip.dst==10.0.0.100 or ip.dst==10.0.0.2 or ip.src==10.0.0.100 or ip.src==10.0.0.2 or ip.src==0.0.0.0) filtragem utilizada no Ethereal.

8.10.2.3. Se o 7960 liga e o 7910 não está disponível

Depois desligou-se o cabo do telefone IP 7910 (10.0.0.102), e realizou-se uma chamada do 7960 (10.0.0.100). Dando no Ethereal do CCM com o endereço 10.0.0.2, as seguintes mensagens.

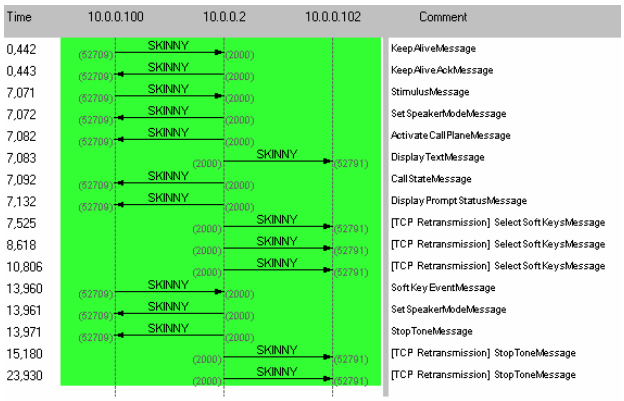


Figura 112 Se o 7960 liga e o 7910 não está disponível

Na tabela seguinte tem uma lista de mensagens que são necessários para abrir e fechar as sessões.

Skinny Data Session Messages	
Skinny Inspection Message	**Descrição**
StationOpenReceiveChannelAckMessage	Contém o endereço IP e a informação do porto do cliente Skinny que está a enviar a mensagem. Esta mensagem também contem o estado, no caso do cliente irá receber tráfego de voz.
StationStartMediaTransmissionMessage	Contém o endereço IP e a informação do porto do cliente Skinny remoto
StationCloseReceiveChannelMessage	O CM manda o cliente Skinny (dependendo na informação da mensagem) para fechar o canal receptor.
StationStopMediaTransmissionMessage	O CM manda o cliente Skinny (dependendo na informação da mensagem) para parar de transmitir tráfego de voz.
StationStopSessionTransmissionMessage	O CM manda o cliente Skinny (dependendo na informação da mensagem) para terminar uma determinada sessão.

Tabela 15 Tabela com algumas mensagens Skinny relevantes

8.10.3. Telefonema entre telephones IP

Experiência realizada: 7960 para o 7910 e o 7910 atende e desliga, 7910 para o 7960 o 7960 atende e o 7910 desliga

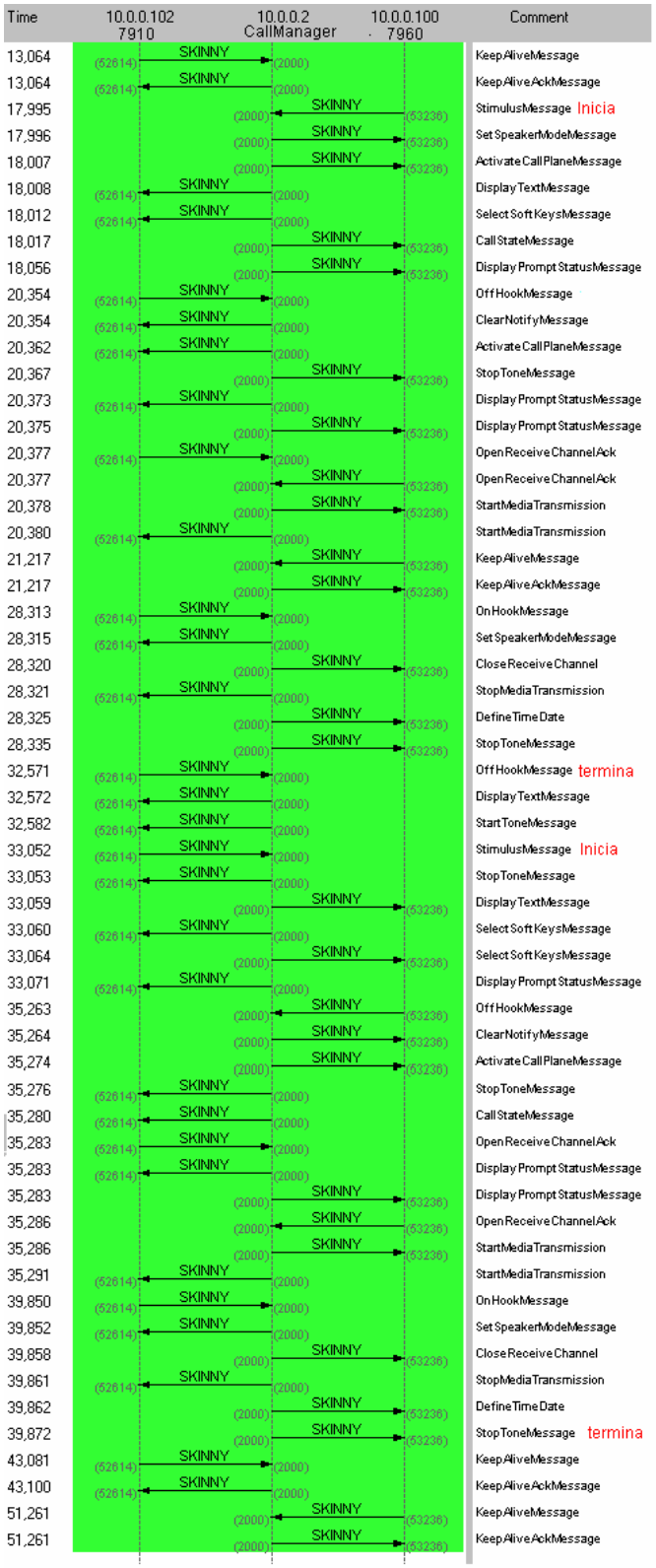


Figura 113 captura no Cisco CallManager

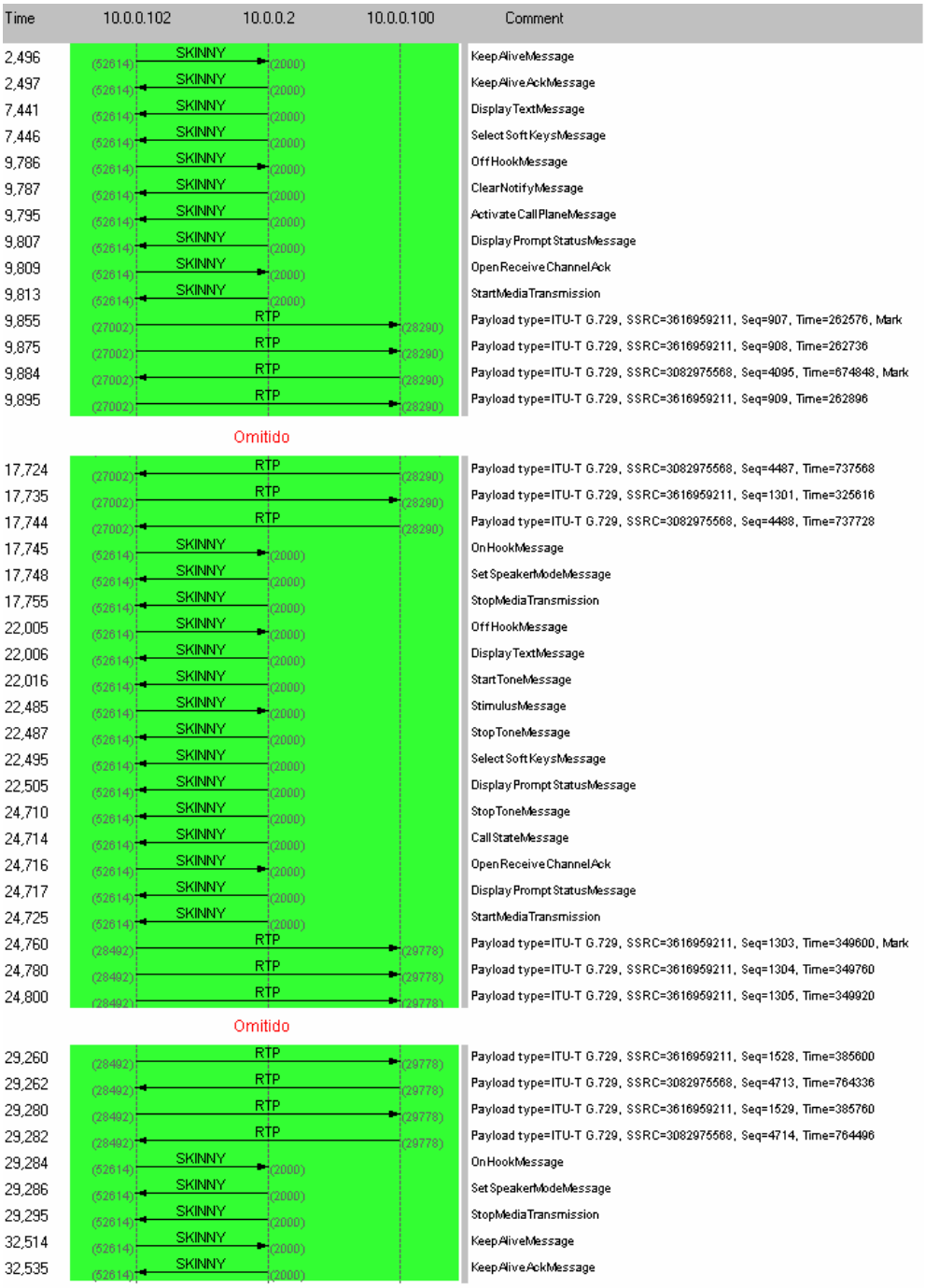


Figura 114 *Ethereal* no PC10

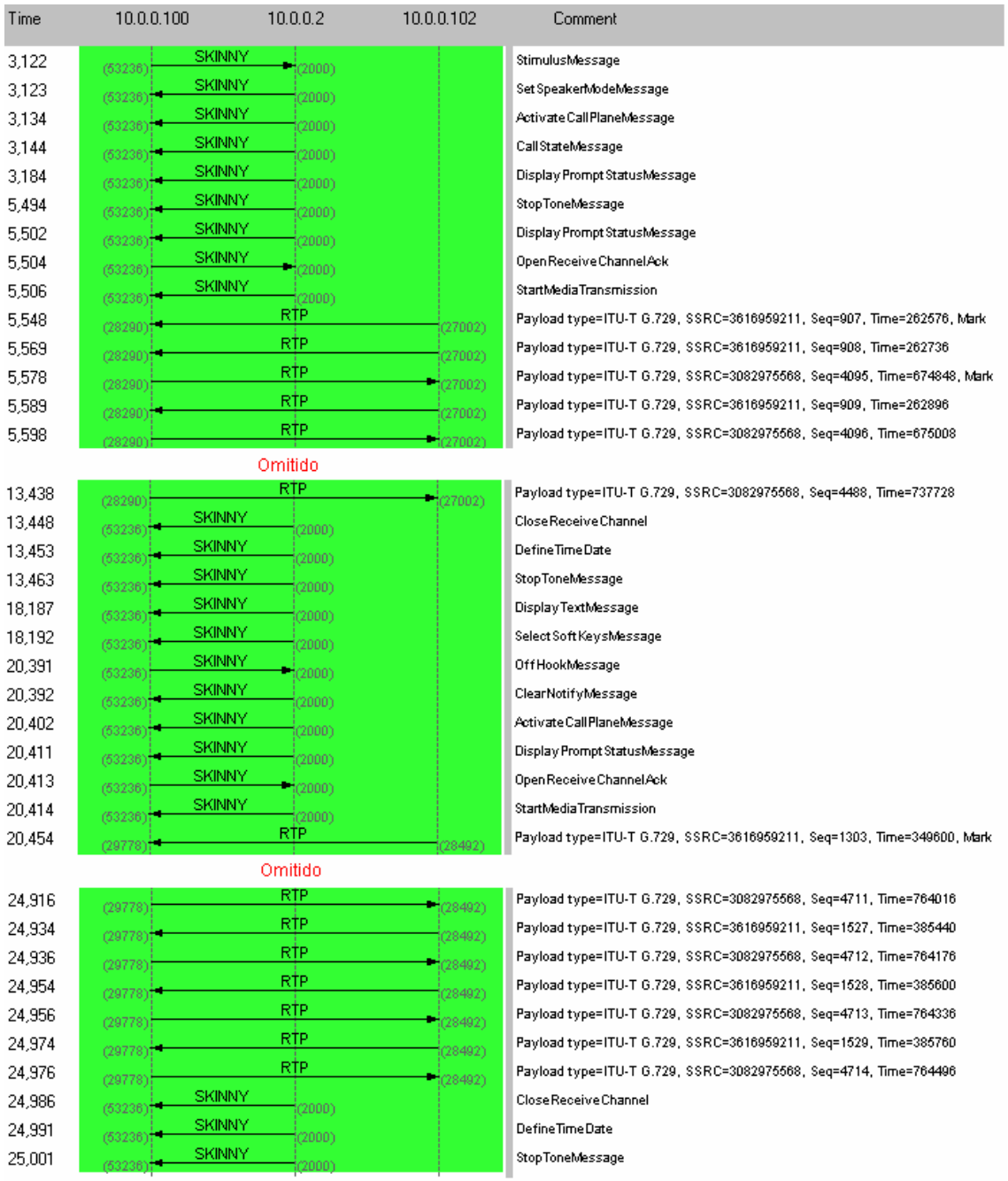


Figura 115 *Ethereal* no PC11

8.10.4. Do telefone IP para o analógico

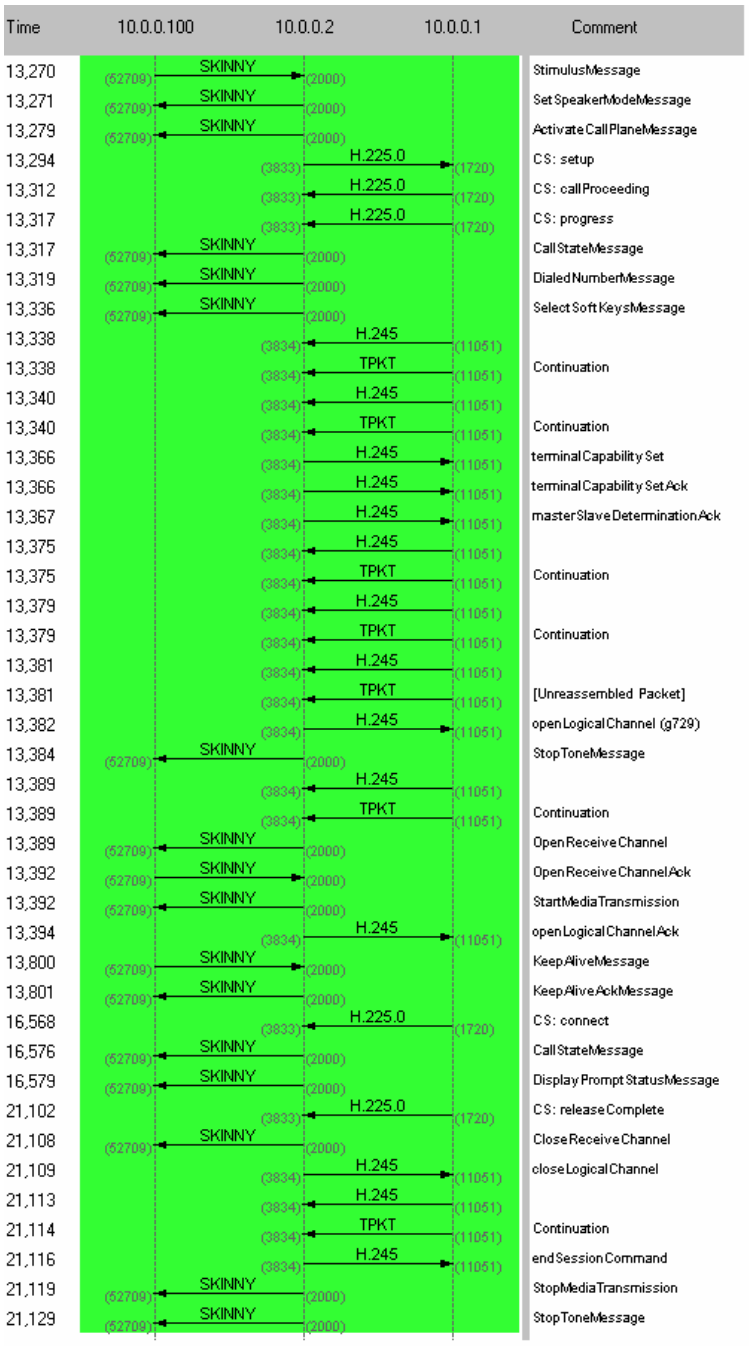


Figura 116 Chamada do telefone IP 7960 → analógico (capturado no Ethereal PC)

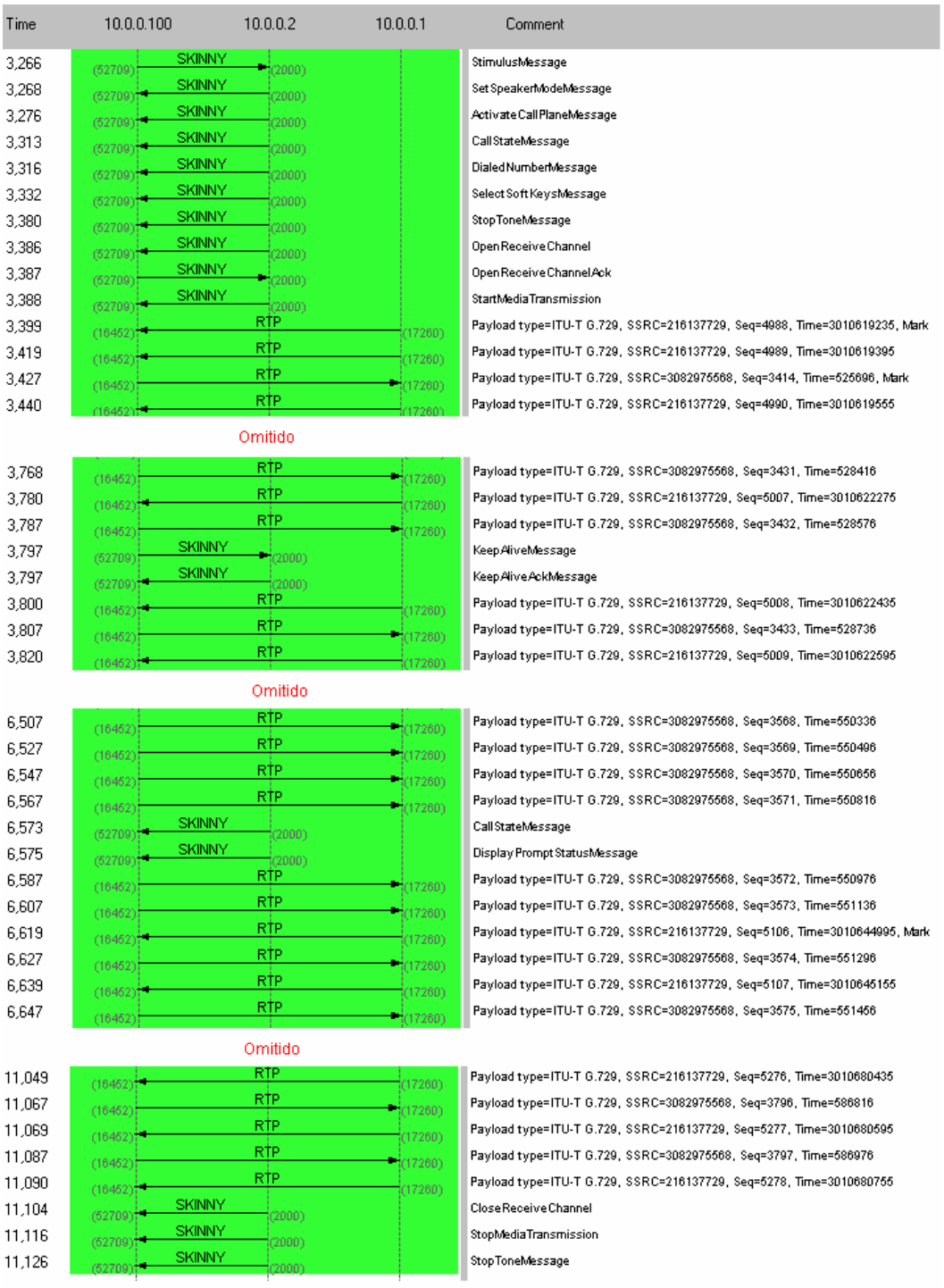


Figura 117 Chamada do telefone IP 7960 → analógico (cont. 1)

8.10.5. Do telefone analógico para o telefone IP 7960

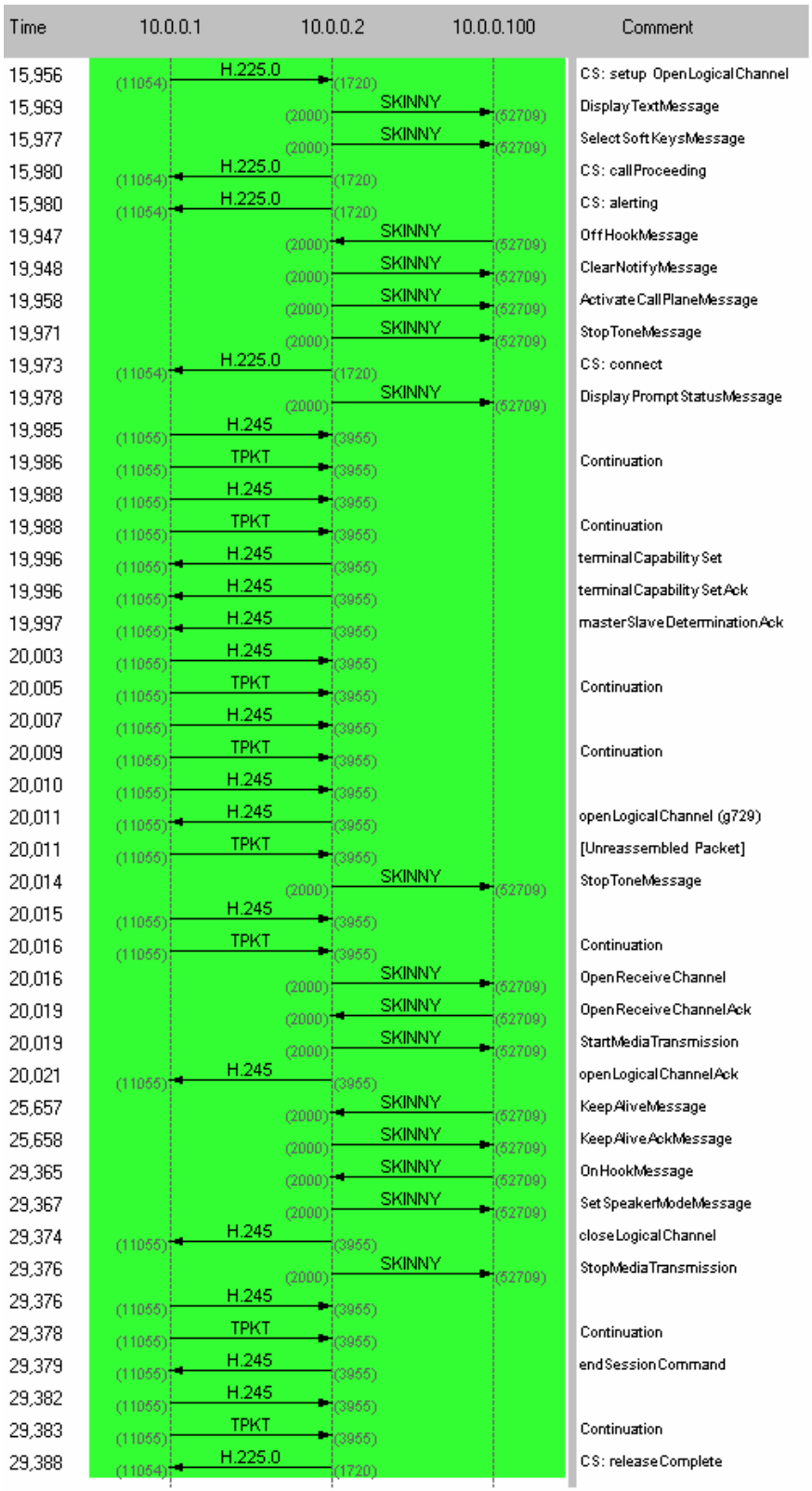


Figura 118 Chamada do analógico para o telefone IP

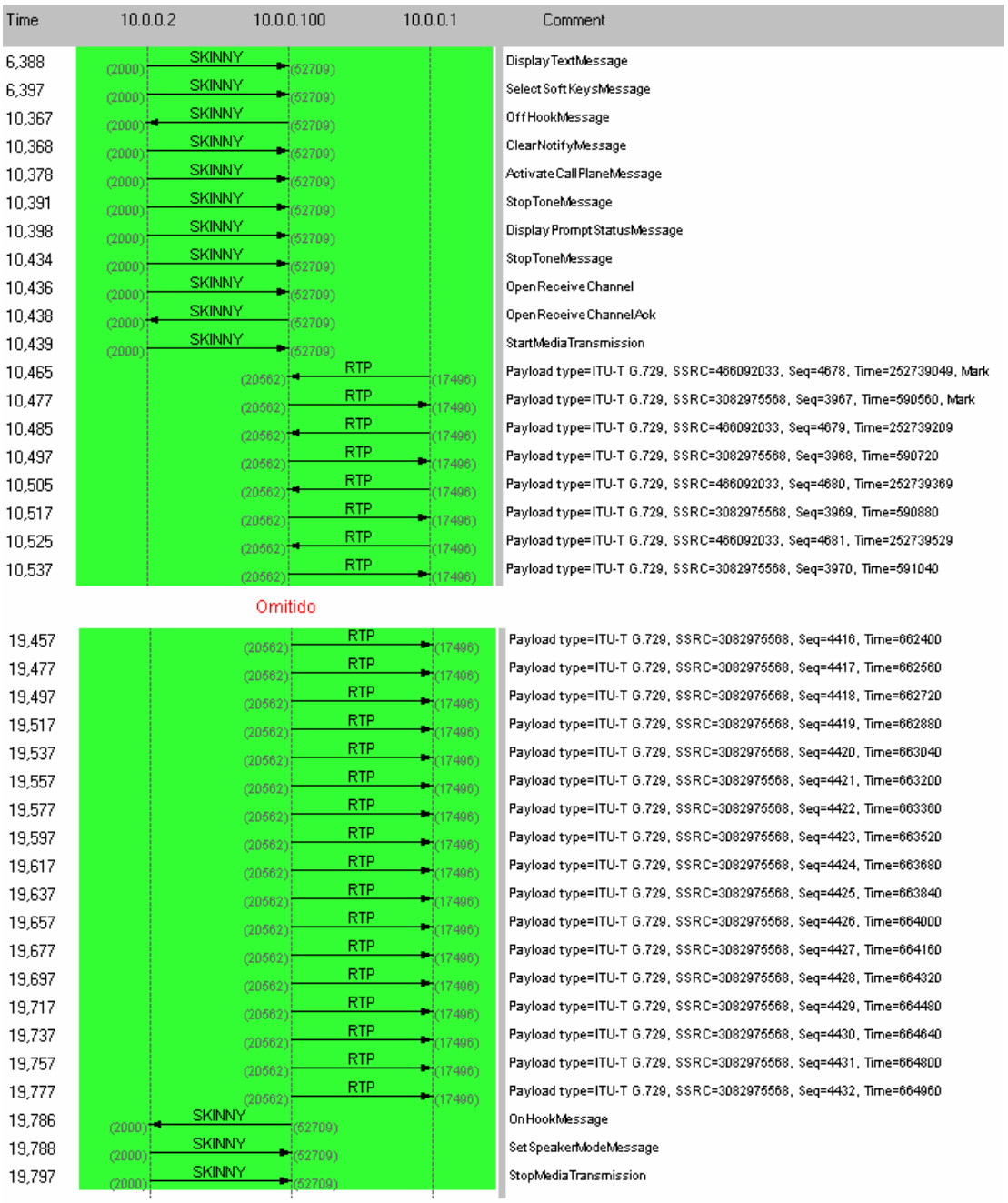


Figura 119 Chamada do analógico para o telefone IP (cont.)

A mensagem RTP capturada, verifica-se que utiliza o codec G.729.

```
⊞ Frame 553 (74 bytes on wire, 74 bytes captured)
⊞ Ethernet II, Src: Cisco_d9:ea:00 (00:0a:b7:d9:ea:00), Dst: 10.0.0.100 (00:07:50:79:c2:b7)
⊞ Internet Protocol, Src: 10.0.0.1 (10.0.0.1), Dst: 10.0.0.100 (10.0.0.100)
⊞ User Datagram Protocol, Src Port: 19260 (19260), Dst Port: 31940 (31940)
⊟ Real-Time Transport Protocol
  ⊞ [Stream setup by Skinny (frame 82)]
    10.. .... = Version: RFC 1889 Version (2)
    ..0. .... = Padding: False
    ...0 .... = Extension: False
    .... 0000 = Contributing source identifiers count: 0
    0... .... = Marker: False
    Payload type: ITU-T G.729 (18)
    Sequence number: 1036
    Timestamp: 2045817449
    Synchronization Source identifier: 556531713
    Payload: 30BE3A6DC0AFCE2AB140F8F2BD048C0E9375B258
```

Figura 120 Pacote RTP

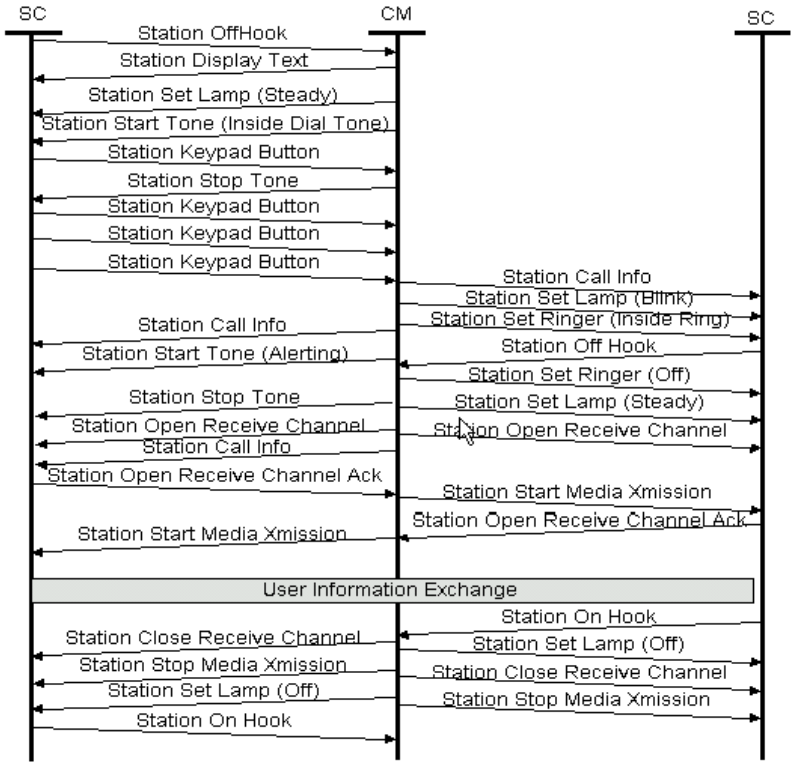


Figura 121 O CM é o intermediário para o controlo das chamadas.[11]

8.11. Configurações para o segundo cenário túnel GRE

Configuração necessário para o funcionamento do túnel GRE:

hostname **Router1**	hostname **Router2**
ipv6 unicast-routing	ipv6 unicast-routing
interface Tunnel0	interface Tunnel0
ip unnumbered FastEthernet0/1	ip unnumbered FastEthernet0/1
tunnel source FastEthernet0/0	tunnel source FastEthernet0/0
tunnel destination 2001::2	tunnel destination 2001::1
tunnel mode gre ipv6	tunnel mode gre ipv6
no shut	no shut
exit	exit
interface FastEthernet0/0	interface FastEthernet0/0
ipv6 address 2001::1/64	ipv6 address 2001::2/64
no shut	no shut
exit	exit
interface FastEthernet0/1	interface FastEthernet0/1
ip address 10.0.0.1 255.0.0.0	ip address 192.168.0.1 255.255.255.0
no shut	ip helper-address 10.0.0.2
exit	no shut
	exit
ip route 0.0.0.0 0.0.0.0 Tunnel0	ip route 0.0.0.0 0.0.0.0 Tunnel0

Como foi referido anteriormente, os telefones IP da Cisco necessitam de DHCP e TFTP para funcionarem. Configuração das *pools* DHCP no CallManager, a figura mostra as duas *pools* que foram criadas no CCM.

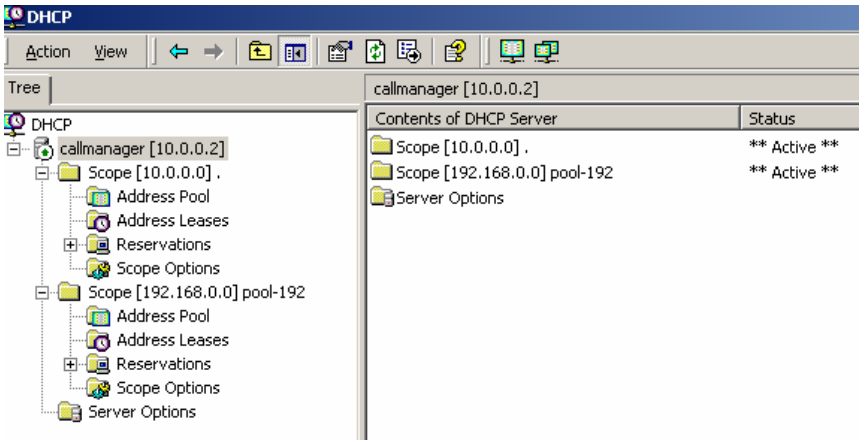


Figura 122 DHCP no CallManager (duas pools 10.0.0.2/8 e 192.168.0.0/24)

No router2 para redundância DHCP deve configurar os seguintes comandos no Router2 (nota este comandos foram explica no ponto 8.10.2.1):

ip dhcp excluded-address 192.168.0.1

ip dhcp pool pool-192

network 192.168.0.0 255.255.255.0

option 150 ip 10.0.0.2

default-router 192.168.0.1

Quando se configure a *pool* no CCM, tem se indicar o *default router* ou *gateway* e a *option* 150. Assim o telefone 7960 poderá então obter um endereço IP mesmo ligação WAN que a ligação esteja indisponível, e. Para o telefone analógico ligado a interface FXS do router, foi necessário criar um *gateway*.

Configurações no Router1

```
dial-peer voice 1 voip
!Caso o número seja para 800
 destination-pattern 800
!vai pelo o endereço 192.168.0.1
 session target ipv4:192.168.0.1
```

Configurações no Router2

```
! ordem de preferência de codecs
voice class codec 1
 codec preference 1 g729r8
 codec preference 2 g711ulaw

dial-peer voice 100 voip
!qualquer seja o número de três digitos
 destination-pattern ...
! feito a referência à class codec a utilizar
 voice-class codec 1
! Todos as chamadas devem ser estabelecidas pelo o CCM
 session target ipv4:10.0.0.2
! (Opcional) Este comando serve para que o telefone IP consiga  ver o número de telefone
!do telefone analógico (100) ou o CalledID
translate-outgoing called 100
! (Opcional) encaminha os tons DTMF utilizando quando o utilizador digita os caracteres
! alfanuméricos H.245 de 0 a 9, *, #, e de A a D.
dtmf-relay h245-alphanumeric

dial-peer voice 800 pots
!número de telefone do telefone analógico
 destination-pattern 800
 port 1/0/1
```

- Criou-se um *gateway* H.323 no CCM com o endereço IP do Router2
- Criou-se uma *route-pattern* (número 800) no CCM apontar para o Router2

Para saber cada comando em detalhe ver o *command reference* do IOS da Cisco[37]. Para encontrar mais exemplos de cenários que incluem telefones analógico, é só visitar o

[37]http://www.cisco.com/univercd/cc/td/doc/product/software/ios124/124cr/hvr_r/index.htm

seguinte *link*[38]. Na figura seguinte é mostrado a configuração do *gateway* H.323 no CallManager.

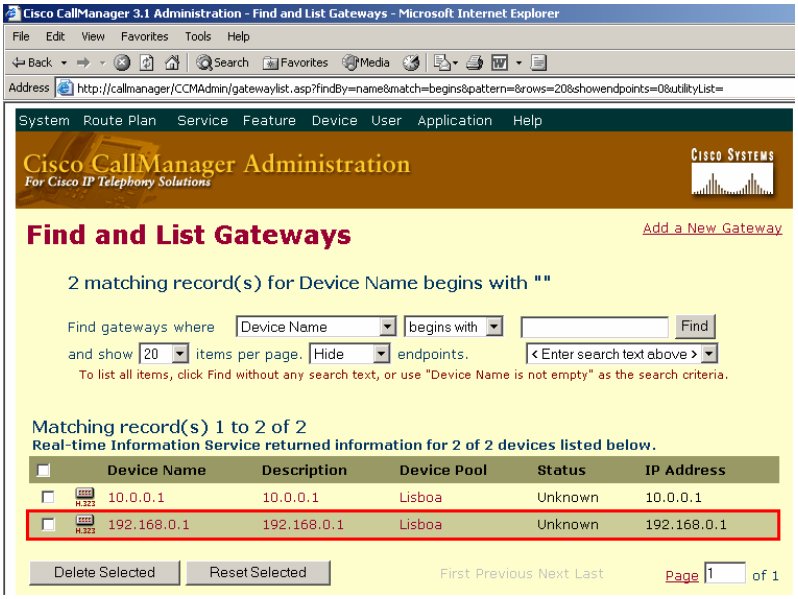


Figura 123 Configuração do *gateway* H.323 com o endereço do Rotuer2 192.168.0.1

Verificação no Ethereal do processo de Encapsulamento GRE nos protocolos ICMP, skinny e RTP

O primeiro teste realizado foi com um *ping* do CCM (com o endereço 10.0.0.2) para um PC (com o endereço IP 192.168.0.4) (onde devia estar o telefone IP 7960). Como se pode verificar pelas figuras seguintes, o protocolo IPv4 está encapsulado dentro do GRE e o GRE está no extension header do IPv6.

A figura seguinte mostra um pedido ICMP capturado no Ethereal.

[38]http://www.cisco.com/univercd/cc/td/doc/product/software/ios123/123cgcr/vvfax_c/int_c/dpeer_c/dp_app_a.htm ou
Dial Peer Configuration Examples

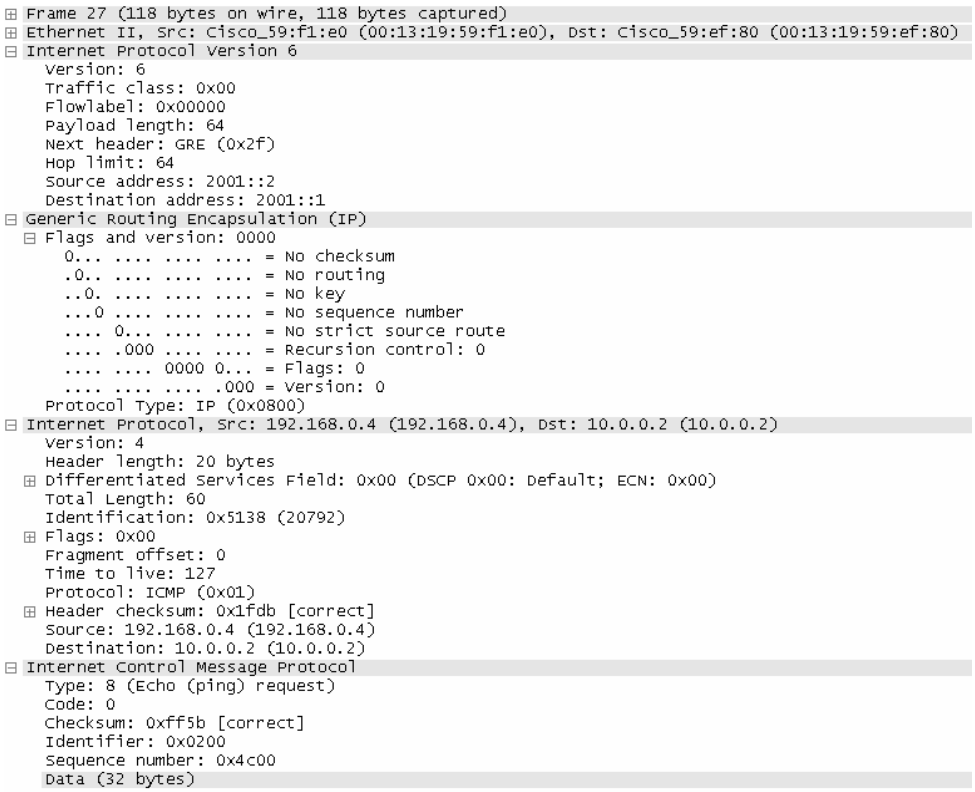


Figura 124 ICMPv6 Request capturado no Túnel GRE

A figura seguinte mostra a resposta ICMP capturado no Ethereal.

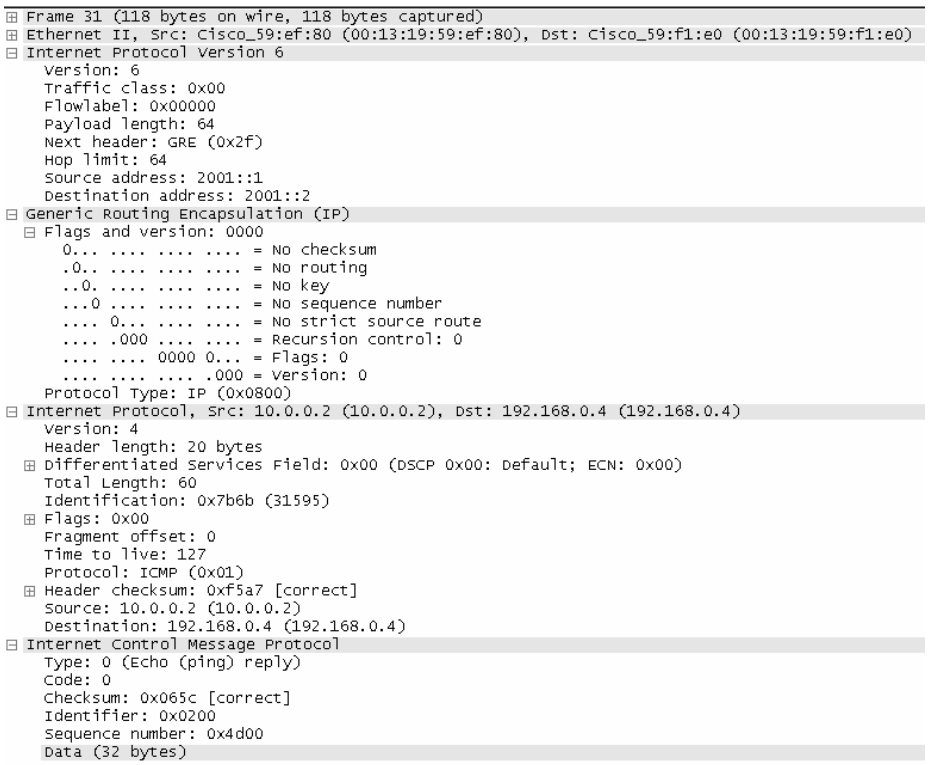


Figura 125 Pacote ICMP IPv6 reply, capturado a meio do túnel GRE

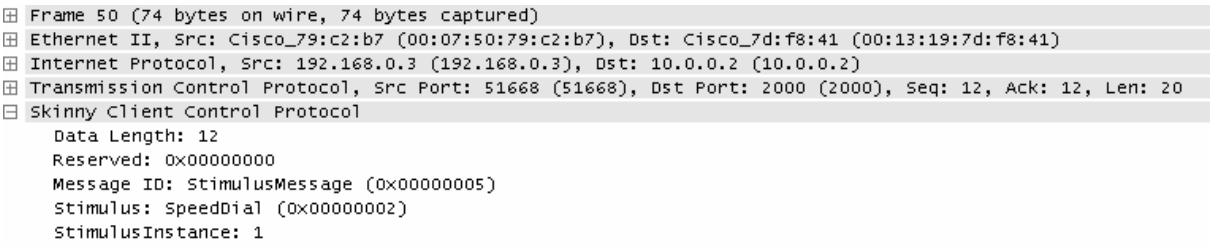


Figura 126 Mensagem StimulusMessage IPv4 para a inicialização da chamada

No dentro do túnel

O mesmo pacote que foi capturado fora do túnel aparece agora encapsulado no GRE e por sua vez no IPv6 de acordo com a figura seguinte.

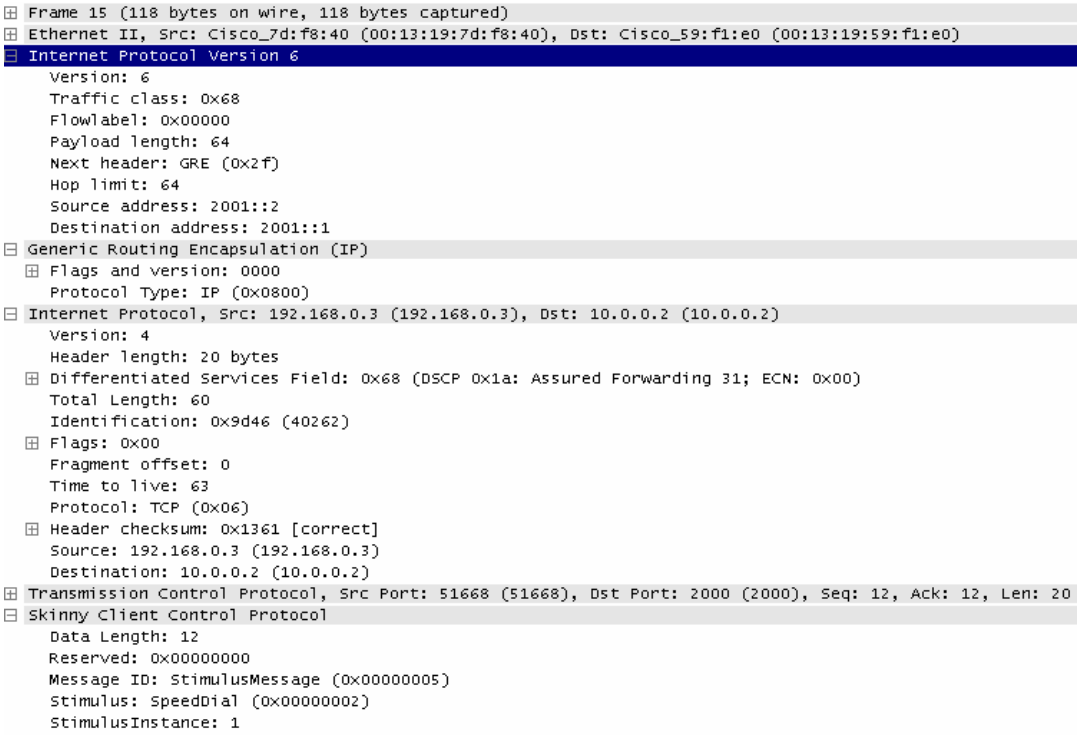


Figura 127 pacote Skinny IPv6

A informação sobre o número telefone de *calling party e called party* está na mensagem Skinny chamado CallMessageInfo (ver figura seguinte). O *calling party* é quem está a chamar e o *called party* é quem recebe ou atende a chamada.

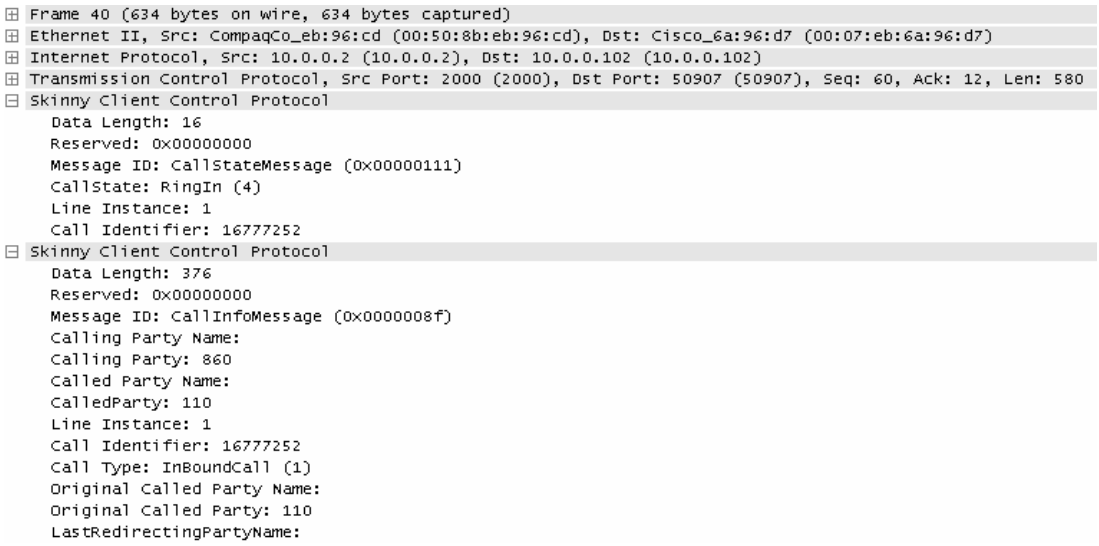


Figura 128 Mensagem com a informação do *Called Party* e *Calling party* em IPv4

Pacote RTP Fora do túnel e dentro do túnel

Verifica-se que a mensagem RTP é ponto a ponto. E tem a seguinte forma em IPv4, de acordo com a figura.

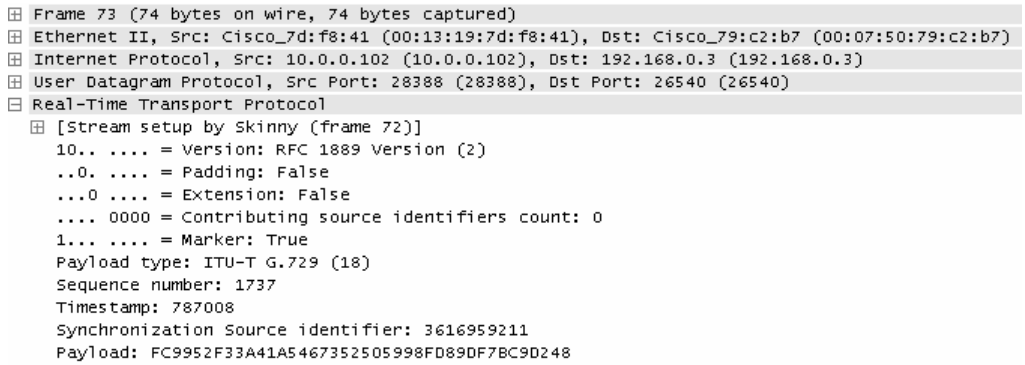


Figura 129 Pacote RTP IPv4, verifica-se que é sobre UDP

Quando o pacote está a passar pelo o túnel, tem a seguinte estrutura de acordo com a figura seguinte:

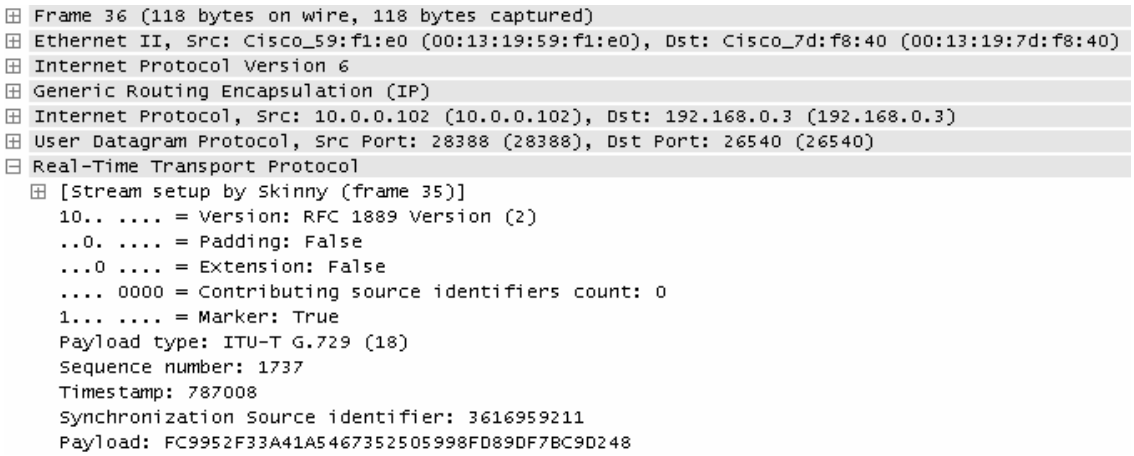


Figura 130 Pacote RTP no túnel IPv6

8.12. Configurações para os Teste QoS

8.12.1. Instalação do Chariot da NetIQ

O Chariot é uma ferramenta que serve para testar a performance da rede através da geração de tráfego. A escolha incidiu nesta ferramenta, uma vez que foi utilizado na cadeira de GIRS Gestão Inteligente de Redes e Serviços. O Chariot tem dois componentes o gerador de tráfego e os seus agentes. Como foi utilizado o Windows XP foi necessário activar o Type of Service (TOS) uma vez que fica desactivada por defeito.

Instalação e configuração do agente[39]

Nos agents é necessário no regedit realizar os seguintes passos:

- Entra-se no Regedit do Windows (Inicia>Executar>Regedit)
- Depois localiza-se a pasta HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\Tcpip\Parameters
- Depois no menu **Edit** cria-se numa nova variável do tipo **REG_DWORD** com o nome *DisableUserTOSSetting ver* Figura 131.

[39] Ver a ajuda do chariot ou o seguinte link: http://wani.bbsnavi.net/wc/support.microsoft.com?kbid=329704&product=wms

- Na **prompt** box coloca-se o valor zero.
- Fechar o editor **Registry,** e reiniciar o computador.

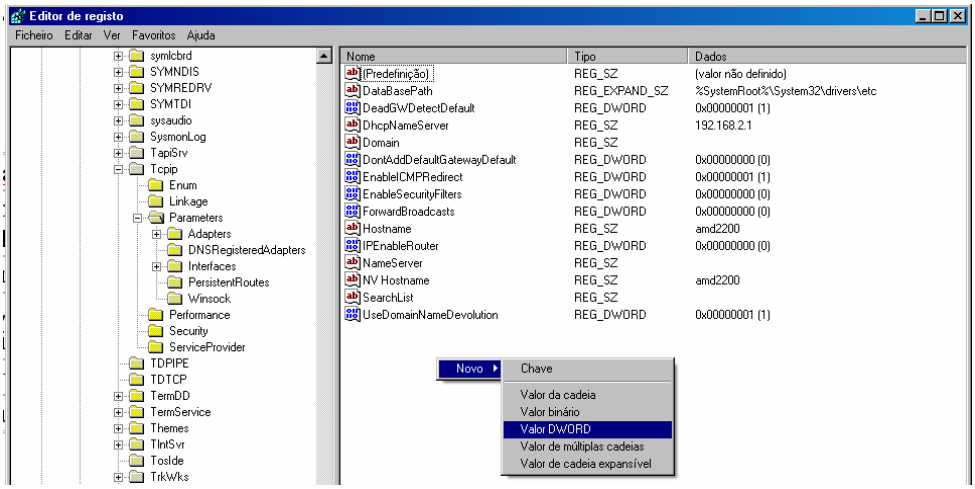


Figura 131 Novo DWORD chamado DisableUserTOSSetting

8.12.2. MGEN[40]

O múltiplo gerador (*MGEN, Multi-Generator*) é uma aplicação de código aberto desenvolvida pelo grupo de trabalho PROTEAN *(PROTocol Engineering Advanced Networking)* pertencente ao laboratório de pesquisa e desenvolvimento da marinha Norte Americana. Foi necessário recorrer a esta aplicação uma vez que ele suporta IPv6 e o Chariot não suporta. O MGEN ocorre em várias plataformas. Foi utilizado um executável para Windows. Para executar a aplicação é necessário um *script*.

Código do *script* que foi utilizado:

TXBUFFER 80000000
5.0 ON 1 UDP DST [2003::3]/5000 PERIODIC [256 512]
120.0 OFF 1

O script indica quantos fluxos ou streams irá ser gerados, o destino e o débito. Assim é gerado 1 Mbps para a máquina com o endereço IPv6 2003::3 durante 2 minutos. Verificou-se no destino que o valor do *throughput* (sem outro tráfego adicional do Chariot) que chegavam cerca de 1Mbps. Comando necessário executar na linha de comando do Windows:

C:\>mgen ipv6 input script.txt txlog

[40] http://downloads.pf.itd.nrl.navy.mil/mgen/

Nas máquinas PC3 e PC4 é necessário instalar o IPv6, basta executar o comando **ipv6 install.** Configurar os PC3 e PC4 com endereços ipv6, é necessário primeiro verificar qual o número ou identificador da interface que pertence à área local, executando o seguinte comando:

Netsh>interface>ipv6>sh addr

E ver qual o número da ligação da área local **N**

Para atribuir um endereço ipv6 é necessário executar o seguinte comando:

Netsh>interface>ipv6>add addr *N* 2000::3

Para eliminar:

Netsh>interface>ipv6>delete addr *N* 2000::3

A seguir é mostrado um printscreen do MGEN a gerar tráfego IPv6:

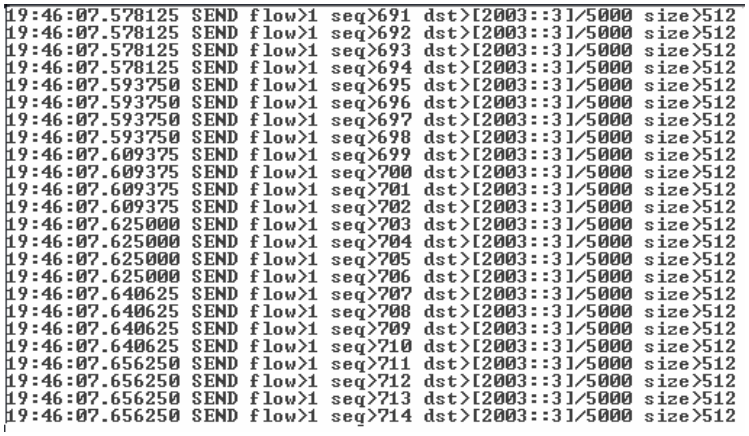


Figura 132 MGEN a gerar tráfego ipv6

No receptor do tráfego MGEN, procurou-se determinar as perdas, para isso é necessário o seguinte comando:

mgen ipv6 port 5000 output log.x

Verificou-se aplicação não conseguia capturar e armazenar a informação no ficheiro output.txt. Para saber mais sobre o modo de funcionamento do MGEN ver em anexo 8.13.

MGEN

Nº de bytes da *frame* 574 bytes no *ethereal.*

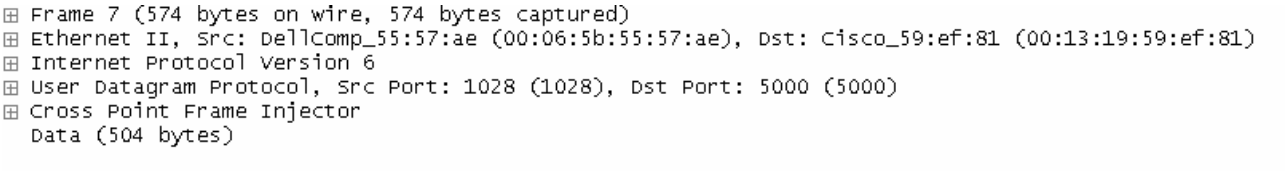


Figura 133 Pacote UDP gerado do MGEN

Alterações no *script* do MGEN para cada teste realizado

64kbps no script: 5.0 ON 1 UDP DST [2003::3]/5000 PERIODIC [**16** 512]

128kbps no script: 5.0 ON 1 UDP DST [2003::3]/5000 PERIODIC [**33** 512]

256kbps no script: 5.0 ON 1 UDP DST [2003::3]/5000 PERIODIC [**66** 512]

512kbps: 5.0 ON 1 UDP DST [2003::3]/5000 PERIODIC **[131** 512]

Nota: Os cálculos realizados foram baseados num ficheiro Excel fornecido pelo aluno Rui Bernardo de projecto I.

Não foi possível obter as perdas do MGEN uma vez que aplicação no Windows e Linux não funcionam e por falta de tempo para fazer *troubleshooting*

mgen port 5000,5004-5006 output log.x

8.12.3. Atribuição do valor DSCP no Chariot para marcar os Pacotes

O Chariot marca dos pacotes RTP com o valor **cs5**. No caso dos telefones a Cisco marcam os pacotes com outro valor 46, ou seja 101110 ou **ef**, ou seja *Expedited Forwarding* EF. Assim foi necessário alterar o valor no Chariot. No Chariot foi necessário alterar o valor do DSCP para a marcação dos pacotes para EF. Que tem o valor binário de 101110.

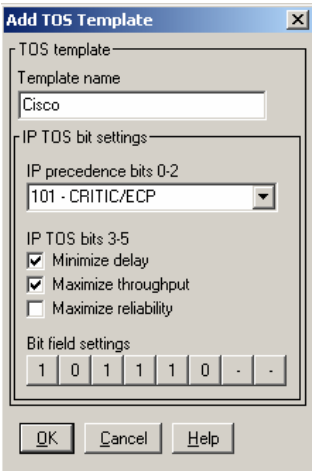


Figura 134 Marcação dos pacotes com o valor 46 (DEC) mesmo dos telefones

```
⊞ Frame 14 (74 bytes on wire, 74 bytes captured)
⊞ Ethernet II, Src: DellComp_55:4e:5d (00:06:5b:55:4e:5d), Dst: Cisco_59:ef:80 (00:13:19:59:ef:80)
⊟ Internet Protocol, Src: 10.0.0.2 (10.0.0.2), Dst: 192.168.0.2 (192.168.0.2)
    Version: 4
    Header length: 20 bytes
  ⊞ Differentiated Services Field: 0xb8 (DSCP 0x2e: Expedited Forwarding; ECN: 0x00)
    Total Length: 60
    Identification: 0xb1b1 (45489)
  ⊞ Flags: 0x00
    Fragment offset: 0
    Time to live: 128
    Protocol: UDP (0x11)
    Header checksum: 0xbd9b [correct]
    Source: 10.0.0.2 (10.0.0.2)
    Destination: 192.168.0.2 (192.168.0.2)
⊟ User Datagram Protocol, Src Port: 4302 (4302), Dst Port: 16440 (16440)
    Source port: 4302 (4302)
    Destination port: 16440 (16440)
    Length: 40
    Checksum: 0xec31 [correct]
  Data (32 bytes)
```

Figura 135 Pacote gerado pelo o Chariot

8.12.4. Alteração da stream RTP do Chariot

O Chariot permite gerar *streams* RTP, no entanto verificou-se no Ethereal, que as *frames* geradas pelo o Chariot (98 bytes) não tinha o mesmo tamanho que a *frame (*74 bytes*)* dos pacotes RTP do telefone da Cisco. Alterou-se então editou-se o *script* de modo a obter a *stream* pretendida, tanto no tamanho como no *throughput*.

No Chariot alterou-se o send_buffer_size para 20 de modo a ter as mesmas características dos da norma G.729 dos telefones IP da Cisco para dar uma *frame* de 74 bytes. É necessário alterar o send_buffer_size para 20 e o send_data_rate a 8 kbps.

Line	Endpoint 1	Endpoint 2
1	RTP_PAYLOAD_TYPE	
2	type = G729	
3	SLEEP	
4	time = initial_delay (0)	
5	CONNECT_INITIATE	CONNECT_ACCEPT
6	port = source_port (AUTO)	port = destination_port (AUTO)
7	LOOP	LOOP
8	count = number_of_timing_records (100)	count = number_of_timing_records (100)
9		START_TIMER
10	SEND	RECEIVE
11	size = file_size (1760)	size = file_size (1760)
12	buffer = send_buffer_size (20)	buffer = receive_buffer_size (DEFAULT)
13	type = send_datatype (NOCOMPRESS)	
14	rate = send_data_rate (8 kbps)	
15		END_TIMER
16	END_LOOP	END_LOOP
17	DISCONNECT	DISCONNECT
18	type = close_type (Reset)	type = close_type (Reset)

Figura 136 Alteração do script no Chariot para dar uma frame de 74 bytes para 29.6kbps

No menu *statistics summay* do Ethereal é possível verificar as estatísticas da *stream* RTP gerado pelo o Chariot, como o número de pacotes, o débito médio em Mbit/sec. Para o caso do RTP o importante é verificar o número de pacotes enviados por segundo (50pps), e a frame ter 74bytes.

Traffic	Captured	Displayed
Between first and last packet	6,436 sec	6,436 sec
Packets	346	322
Avg. packets/sec	53,758	50,029
Avg. packet size	74,000 bytes	74,000 bytes
Bytes	25604	23828
Avg. bytes/sec	3978,076	3702,140
Avg. MBit/sec	0,032	0,030

Figura 137 *Stream* RTP gerado pelo o Chariot (32kbps)

De uma *stream* RTP entre telefones:

Traffic	Captured	Displayed
Between first and last packet	31,564 sec	19,428 sec
Packets	1375	1241
Avg. packets/sec	43,563	63,878
Avg. packet size	76,000 bytes	74,000 bytes
Bytes	104596	91834
Avg. bytes/sec	3313,827	4726,942
Avg. MBit/sec	0,027	0,038

Figura 138 *Stream* RTP dos telefones da Cisco (27 kbps)

Como o *throughput* varia, e de facto o Ethereal apenas se baseia nos pacotes capturados, nunca irá fica exactamente igual, mas estão está correcto, é possível verificar que estão com o mesmo tamanho de 74 bytes, e também enviar 50pps.

8.12.5. Valor possíveis para o dscp no Router

Router1(config-cmap)# match dscp ?
<0-63> Differentiated services codepoint value
af11 Match packets with AF11 dscp (001010)
af12 Match packets with AF12 dscp (001100)
af13 Match packets with AF13 dscp (001110)
af21 Match packets with AF21 dscp (010010)
af22 Match packets with AF22 dscp (010100)
af23 Match packets with AF23 dscp (010110)
af31 Match packets with AF31 dscp (011010)→Para o protocolo Skinny
af32 Match packets with AF32 dscp (011100)
af33 Match packets with AF33 dscp (011110)
af41 Match packets with AF41 dscp (100010)
af42 Match packets with AF42 dscp (100100)
af43 Match packets with AF43 dscp (100110)
cs1 Match packets with CS1(precedence 1) dscp (001000)
cs2 Match packets with CS2(precedence 2) dscp (010000)
cs3 Match packets with CS3(precedence 3) dscp (011000)
cs4 Match packets with CS4(precedence 4) dscp (100000)
cs5 Match packets with CS5(precedence 5) dscp (101000)
→do chariot a norma G729 HEX 28→binário dá 101000
cs6 Match packets with CS6(precedence 6) dscp (110000)
cs7 Match packets with CS7(precedence 7) dscp (111000)
default Match packets with default dscp (000000)
ef Match packets with EF dscp (101110)→utilizado pelos os telefones Cisco.

8.13. MGEN 4.2 [40]

O múltiplo gerador (MGEN, Multi-Generator) é uma aplicação de código aberto desenvolvida pelo grupo de trabalho PROTEAN (PROTocol Engineering Advanced Networking) pertencente ao laboratório de pesquisa e desenvolvimento da marinha Norte Americana [21]. Este laboratório tem o nome de U. S. Naval Research Laborary (NRL) e é responsável pelo largo programa de investigação e desenvolvimento avançado de tecnologias. Está inserido na marinha e foi criado no ano de 1924, sendo uma instituição de renome, com vastas provas dadas, continua rumo ao futuro na tentativa de criar / inovar as tecnologias existentes.

O NRL é constituído por vários departamentos, a secção de redes e sistemas de comunicação (Networks and Communication Systems Branch) uma das principais, sendo constituída por diversas secções distintas mas que se completam entre si, cada uma responsável por uma determinada área, quer seja no desenvolvimento de determinados projectos inovadores, investigação de engenharia de protocolos avançados ou na integração de sistemas e instrumentação.

Enquadramento

O NRL PROTEAN Research Group tenta desenvolver comunicações versáteis e adaptativas, protocolos de rede e, soluções tecnológicas para ambos os meios, com e sem fios. Actualmente, estão a decorrer vários projectos, destacando-se o NTP (Network Traffic Management) e o MANET (Mobile Ad-Hoc Networking). O primeiro permite a análise e investigação na área da gestão avançada de tráfego e qualidade de serviço, o segundo trata-se também de uma investigação, em conjunto com o IETF, na área de Wireless Routing e serviços de rede. Além destes, existe já um vasto leque de produtos lançados, uns descontinuados e outros em constante melhoramento. No último caso englobase o MGEN, um versátil gerador de tráfego para medir o desempenho da rede.

De seguida são apresentadas, de forma sintetizada, algumas das várias ferramentas existentes criadas pelo PROTEAN Research Group:

_ TRPR - acrónimo de Tcpdump Rate Plot Real Time, trata-se de uma ferramenta para análise gráfica de tráfego, quer este seja em tempo real ou não. Permite capturar a informação gerada pela ferramenta tcpdump e MGEN, e inclui vários tipos de opções de filtragem, possibilitando também a integração com outras ferramentas de análise gráfica

_ GPSLogger - ferramenta para monitorizar a informação criada pelos dispositivos GPS,

_ SDT - de Scripted Display Tool. Entre outras funcionalidades, esta ferramenta permite a visualização em tempo real de determinadas informações de redes móveis.

CMAP - em conjunto com o SDT, fornece uma ferramenta flexível para visualização móvel. Possibilita a leitura dos ficheiros de log criados pelo MGEN, OLSR, MNE, etc. e formata-os para poderem ser utilizados na ferramenta SDT.

_ MNE - Scripts – fornece uma grande variedade de scripts para serem utilizados no MNE (Mobile Network Emulator), TRPR e MGEN. Estes scripts correm os cenários de testes MANET com vários modelos de movimento.

8.13.1. Características da ferramenta

A ferramenta MGEN é capaz de gerar diversos tipos de tráfego em tempo real para simular uma rede. O tráfego gerado pode ser analisado e colocado num ficheiro para análise posterior com a ferramenta adequada. Presentemente está disponível, a versão 4.2 que tem a capacidade de gerar tráfego UDP mas prevê-se em futuros lançamentos, a capacidade de gerar também tráfego usando o protocolo TCP.

Através do uso de scripts é possível definir os parâmetros de análise que se pretendem testar. Estes permitem à aplicação gerar diversos tipos de modelos de tráfego, dos quais se destacam a capacidade de gerar tráfego periódico (em stream) e em rajadas. A aplicação requer que, tanto o transmissor como o receptor a executem em simultâneo. Isto é, esta encapsula todas as suas funcionalidades num único ficheiro que corre do lado do cliente e

do receptor. São os scripts que definem, se esta se comporta como transmissor ou como receptor.

A grande característica do MGEN consiste em medir diversos parâmetros de qualidade de serviço:

- Atrasos
- Perdas
- Throughputs

A ferramenta requer, devido ao formato dos pacotes gerados, um cuidado acrescido na altura da realização dos testes. Efectivamente, como é possível constatar na figura A1.1, os pacotes gerados pela aplicação "transportam" a hora, os minutos, e os segundos (com uma precisão de micro segundos) que a máquina transmissora tem na altura que está a gerar os pacotes. Quando a estação receptora, a que se encontra à escuta num determinado porto, recebe os pacotes, utiliza estes valores para calcular os atrasos. Facilmente se compreende que as máquinas devem estar sincronizadas entre si, pois se isso não acontecer, os resultados não são validos. O protocolo NTP (Network Time Protocol) é geralmente utilizado para a sincronização das máquinas

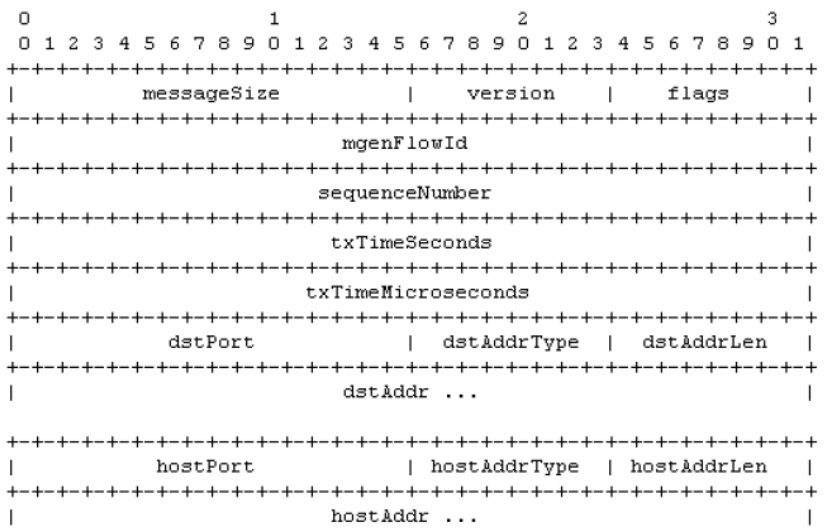


Figura 139 formato do pacote MGEN

8.13.2. Opções da linha de comandos

A versão actual do MGEN necessita de ser "lançada" através da linha de comandos. Num futuro próximo, a equipa de desenvolvimento pretende criar um ambiente gráfico para facilitar a configuração e análise dos resultados. Ao executar o programa são definidas opções, podendo, grande parte delas, ser incluídas no script de configuração. De seguida são apresentadas algumas das opções usadas:

[ipv4][ipv6]

Os comandos permitem definir qual o protocolo que irá ser utilizado [ipv4 ou ipv6].

[input <scriptFile>]

Indica o nome do ficheiro que contém os scripts de teste.

[output <logFile>][log <logFile>]

O primeiro parâmetro indica ao MGEN o nome do ficheiro, usado para armazenar o resultado das medições, dado que este por omissão é apresentado no ecrã. O segundo parâmetro deve ser usado quando se pretende manter os resultados anteriores, isto é, permite a actualização do ficheiro sem perda dos resultados anteriores.

[binary][convert <binaryLog>]

O primeiro parâmetro armazena o resultado dos testes em formato binário, este deve ser colocado antes dos comandos output e log. O segundo parâmetro faz a conversão de um ficheiro binário para texto e apresenta o resultado no ecrã.

[txlog]

É usado para visualizar no ecrã os fluxos que estão a ser enviadas para o receptor.

[hostAddr {on|off}]

Se esta opção for utilizada, então o MGEN coloca nas mensagens enviadas o IP da máquina que gerou os pacotes, permitindo identificar donde estes foram enviados. Esta característica é especialmente útil quando os pacotes atravessam um servidor de NAT.

[event "<mgen event>"]

É usado para configurar os fluxos que se pretendem gerar, sem ser necessário a criação de um ficheiro de script propriamente dito.

[port <recvPortList>]

Permite definir o(s) porto(s) onde a aplicação irá ficar à escuta.

[interface <interfaceName>]

É usado em testes Multicast para definir o endereço IP da máquina que deseja entrar no "grupo".

[tos <typeOfService>]

Permite atribuir aos pacotes um valor da precedência IP.

[txbuffer <txSocketBuffersize>][rxbuffer <rxSocketBuffersize>]

Permite definir o tamanho do *buffer* de transmissão/recepção associado aos *sockets*. Se o valor definido for superior ao máximo suportado pelo sistema operativo, a opção é ignorada.

[start <hr:min:sec>[GMT]][offset <sec>]

Permite definir uma hora a partir da qual a aplicação inicia o teste. O valor de offset pode ser usado para atrasar <seg> o início da execução do script.

precise {on|off}]

Quando esta opção está activa, o MGEN utiliza o máximo de tempo de processador para gerar os pacotes. Esta só deve ser usada quando se pretende realizar testes com uma grande quantidade de dados transmitidos.

[txcheck][rxcheck]

Permite adicionar às mensagens criadas pelo MGEN uma flag de 4 bytes, usada para garantir que os pacotes corruptos não sejam levados em conta para as medições.

8.13.3. Exemplo de utilização

Para iniciar a aplicação basta introduzir, por exemplo, o nome do executável e o nome do script que contém os parâmetros do teste. O segundo parâmetro é usado para apresentar do lado do transmissor a quantidade de fluxos enviada a cada momento para o receptor.

mgen input example.mgn txlog

No outro extremo, a aplicação está à escuta nos portos 5000, 5004, 5005 e 5006 à espera de receber os fluxos de tráfego UDP. Estes vão ficar armazenados no ficheiro log.x

mgen port 5000,5004-5006 output log.x

Formato dos scripts usados pelo MGEN

Grande parte das opções atrás apresentadas pode e deve ser incluída no script de configuração. Estes, não são mais do que simples ficheiros de texto que contêm uma sequência de comandos e eventos, descrevendo o tipo de teste a realizar, portas e grupos Multicast, entre outras opções. Caso seja necessário usar várias linhas para a definição do mesmo fluxo é obrigatório colocar no fim de cada uma destas uma barra ("\").

Antes de se definirem os diversos parâmetros para os scripts, é necessário usar, na primeira linha destes, o comando “TXBUFFER”, responsável pelo tamanho do buffer do transmissor. Os scripts devem obrigatoriamente respeitar o seguinte formato:

[<eventTime>] <eventType> <parameters ...> [<options ...>]

O "eventTime" é usado pelo MGEN para determinar quando é que começa a transmitir os fluxos. Este campo requer o uso dum valor inteiro com uma casa decimal, que geralmente é colocada a zero. Se este campo não for configurado, então a ferramenta inicia automaticamente a transmissão dos fluxos após o executável ter sido lançado.

O "eventType" define e caracteriza o tipo de tráfego que irá ser gerado. Dependendo da configuração, a aplicação pode enviar dados para diversos destinatários. Os fluxos gerados contêm um identificador que é usado pelo receptor para analisar as perdas. Existem dois tipos de eventos disponíveis, transmissão e recepção, no entanto este documento apenas abordará o primeiro caso. Assim, são apresentadas de seguida as opções existentes para o referido tipo de transmissão:

<eventTime> OFF <flowId>

<eventTime> ON <flowId><protocol> [SRC <port>] DST <addr>/<port>
<patternType> [parameters ...]

<eventTime> OFF <flowId>

O último evento permite terminar, aos "eventTime" segundos, a transmissão de um fluxo definido por "flowId".

<eventTime> ON <flowId><protocol>

A presença de "ON" indica à aplicação para gerar um fluxo, identificado pelo "flowId", depois de terem decorrido "eventTime" segundos. No campo "protocol" é definido qual o protocolo usado para transportar os fluxos, actualmente só o UDP é permitido.

[SRC <port>] DST <addr>/<port>

É necessário definir o endereço e o porto da estação que irá receber a informação gerada. O campo "SRC" é opcional e permite definir o porto que a aplicação utiliza para transmitir os fluxos.

<patternType> [parameters ...]

O tráfego gerado pelo MGEN consiste numa série de mensagens numeradas, enviadas em sequência. Estas podem variar no tamanho ou na frequência de transmissão para emular uma série de cenários possíveis. Actualmente, estão definidos três tipos de testes "PERIODIC", “POISSON" e "BURST". O primeiro é usado para enviar uma quantidade constante de dados para a rede, o segundo usa uma distribuição estatística para gerar o tráfego e por fim o último permite gerar picos de tráfego.

PERIODIC [<rate> <size>]

Este modelo usa o parâmetro <size> para gerar mensagens de tamanho fixo, em bytes, a uma taxa constante de mensagens por segundo (<rate>). O segundo parâmetro necessita de ser superior ou igual ao tamanho mínimo dos pacotes gerados pelo MGEN, no entanto, não pode ultrapassar o limite máximo dos pacotes UDP, que é de 8192 bytes.

POISSON [<aveRate (msg/sec)> <size (bytes)>]

Permite gerar mensagens de tamanho fixo em intervalos estatísticos a uma média de <rate> de mensagens por segundo. O tamanho das referidas mensagens é configurado em <size (bytes)>.

BURST [REGULAR|RANDOM <aveInterval (sec)> <patternType> [<patternParams>]

FIXED|EXPONENTIAL <aveDuration (sec)>]

Este teste usa um dos dois modelos apresentados anteriormente para gerar picos de tráfego, durante um determinado intervalo de tempo. O primeiro parâmetro pode ser configurado para criar picos em intervalo de tempo regulares ou aleatórios (<aveInterval (sec)>). De seguida, é necessário definir o tráfego que se pretende usar para criar os burts, recorrendo-se aos modelos existentes. Por fim, resta um último parâmetro que define a duração de cada pico de tráfego.

8.13.4. Exemplo de utilização

Exemplo da criação de um fluxo com 1024 bytes que é enviado à taxa de 10 mensagens por segundo, para a máquina com o endereço de 192.168.234.244 que está à escuta no porto 5000. Passados 15 segundos, este fluxo é terminado.

0.0 ON 2 UDP SRC 5001 DST 192.168.234.244/5000 PERIODIC [10.0 1024]

10.0 OFF 2

armazenadas no ficheiro de log, enquanto que, para medir os atrasos compara-se o tempo de chegada, medido no receptor, com o tempo de transmissão. Como os fluxos estão numerados, é fácil visualizar as perdas existentes no canal.

Cada linha no ficheiro de logs corresponde a um único fluxo. Como já foi mencionado, existem vários eventos possíveis, no entanto nem todos serão abordados. O ficheiro de log, após a recepção dum evento RECV (chegada dum fluxo), apresenta a seguinte forma:Envia um "burst" a cada 10s com o tamanho de 1Mbits (128Hz <-> 1024Bytes), sendo a duração

deste, de 6 segundos. A aplicação começa a enviar os dados após terem decorrido 10 segundos.

15.0 ON 2 UDP DST 192.168.200.1/5005 BURST [REGULAR 10.0 PERIODIC [128.0 1024] FIXED 6.0]

8.13.5. Formato dos ficheiros de log criados pelo MGEN

As mensagens MGEN contêm informação útil e facilitam as medições de performance de uma rede. É através da análise do ficheiro de log que alguns parâmetros podem ser analisados. Para medir o débito efectivo são comparados os tamanhos e os tempos das mensagens

<eventTime> RECV flow><flowId> seq><sequenceNumber> src><addr> /<port> dst><addr>/<port> sent><txTime> size><bytes> [host><addr>/<port>] [gps><status>,<lat>,<long>,<alt>] [data><len>:<data>]

onde:

<eventTime> :: corresponde ao tempo de chegada do fluxo (medido no receptor).
RECV :: permite ver que o fluxo em causa entrou no equipamento (chegada dum fluxo) .
<flowId> :: identificação do fluxo.
seq><sequenceNumber> :: número de sequência do fluxo.
src><addr> /<port> :: endereço e porto de origem da máquina emissora.
dst><addr>/<port> :: endereço e porto de destino da máquina emissora.
sent><txTime> :: tempo existente na máquina transmissora quando esta gera o fluxo.
size><bytes> :: tamanho da mensagem definida no emissor.
host><addr>/<port>] :: endereço da máquina receptora e porto onde a aplicação está a "correr".
[gps><status>,<lat>,<long>,<alt>] [data><len>:<data>]:: estes parâmetros são usados apenas em redes móveis, em redes cabladas estes não contêm informação útil.

8.14. Configurações dos Routers cenário QoS

8.14.1. No Router1

version 12.4
service timestamps debug datetime msec
service timestamps log datetime msec
no service password-encryption
!
hostname Router1
!
boot-start-marker
boot-end-marker
!
resource policy
!
no aaa new-model
memory-size iomem 10
no network-clock-participate slot 1
no network-clock-participate wic 0
!
ip cef
ipv6 unicast-routing
!
class-map match-all skinny
match dscp cs3 af31
match protocol ipv6
class-map match-all voz
match dscp ef
match protocol ipv6

```
!
policy-map pri500
 class voz
  priority 500
 class skinny
  bandwidth 10
!
interface Tunnel0
 ip unnumbered FastEthernet0/0
 tunnel source Serial0/0
 tunnel destination 2001::2
 tunnel mode gre ipv6
!
interface FastEthernet0/0
 ip address 10.0.0.1 255.0.0.0
 duplex auto
 speed auto
!
interface Serial0/0
 no ip address
 ipv6 address 2001::1/64
 clock rate 1000000
 no dce-terminal-timing-enable
 service-policy output pri500
!
interface FastEthernet0/1
 no ip address
 duplex auto
 speed auto
 ipv6 address 2000::1/64
!
interface Serial0/1
 no ip address
 shutdown
```

```
 no dce-terminal-timing-enable
!
interface Serial0/2
 no ip address
 shutdown
 no dce-terminal-timing-enable
!
interface Serial0/3
 no ip address
 shutdown
 no dce-terminal-timing-enable
!
ip route 192.168.0.0 255.255.255.0 Tunnel0
!
ip http server
no ip http secure-server
!
ipv6 route ::/0 Serial0/0
!
control-plane
!
dial-peer voice 1 voip
 destination-pattern 800
 session target ipv4:192.168.0.1
!
line con 0
 logging synchronous
line aux 0
line vty 0 4
 login
!
end
```

8.14.2. No Router2

```
version 12.4
service timestamps debug datetime msec
service timestamps log datetime msec
no service password-encryption
!
hostname Router2
!
boot-start-marker
boot-end-marker
!
resource policy
!
no aaa new-model
memory-size iomem 10
no network-clock-participate slot 1
no network-clock-participate wic 0
!
ip cef
ipv6 unicast-routing
!
voice class codec 1
 codec preference 1 g729r8
 codec preference 2 g711ulaw
!
class-map match-all skinny
 match  dscp cs3  af31
 match protocol ipv6
class-map match-all voz
 match  dscp ef
```

```
 match protocol ipv6
!
!
policy-map pri500
 class voz
  priority 500
 class skinny
  bandwidth 10
!
interface Tunnel0
 ip unnumbered FastEthernet0/0
 tunnel source Serial0/0
 tunnel destination 2001::1
 tunnel mode gre ipv6
!
interface FastEthernet0/0
 ip address 192.168.0.1 255.255.255.0
 ip helper-address 10.0.0.2
 duplex auto
 speed auto
!
interface Serial0/0
 no ip address
 ipv6 address 2001::2/64
 no dce-terminal-timing-enable
 service-policy output pri500
!
interface BRI0/0
 no ip address
 encapsulation hdlc
 shutdown
!
interface FastEthernet0/1
 no ip address
```

```
 duplex auto
 speed auto
 ipv6 address 2003::1/64
!
interface Serial0/1
 no ip address
 shutdown
 no dce-terminal-timing-enable
!
ip route 10.0.0.0 255.0.0.0 Tunnel0
!
ip http server
no ip http secure-server
!
ipv6 route ::/0 Serial0/0
!
control-plane
!
voice-port 1/0/0
voice-port 1/0/1
!
dial-peer voice 100 voip
 destination-pattern
 voice-class codec 1
 session target ipv4:10.0.0.2
 dtmf-relay h245-alphanumeric
 no vad
!
dial-peer voice 800 pots
 destination-pattern 800
 port 1/0/1
!
line con 0
 logging synchronous
```

```
line aux 0
line vty 0 4
 login
end
```

Para ter redundância é possível ter as seguintes configurações no Router2 (para além do comando **ip helper-address 10.0.0.2**)

```
ip dhcp excluded-address 192.168.0.1
!
ip dhcp pool pool-192
   network 192.168.0.0 255.255.255.0
   option 150 ip 10.0.0.2
   default-router 192.168.0.1
```

Assim no caso da WAN estar em baixo, o telefone consegue obter um endereço IP.

8.15. Mensagens SIPv6

Estas mensagen foram capturas no Cenário definido na Figura 68, com a seguinte sequência:

win2 regista-se no ser
win3 regista-se no ser
win2 telefona ao win3
win3 atende fala um pouco e desliga
win2 manda um olá para win3
win3 responde um olá ao win2
win3 desliga o *softphone*
win2 desliga o *softphone*

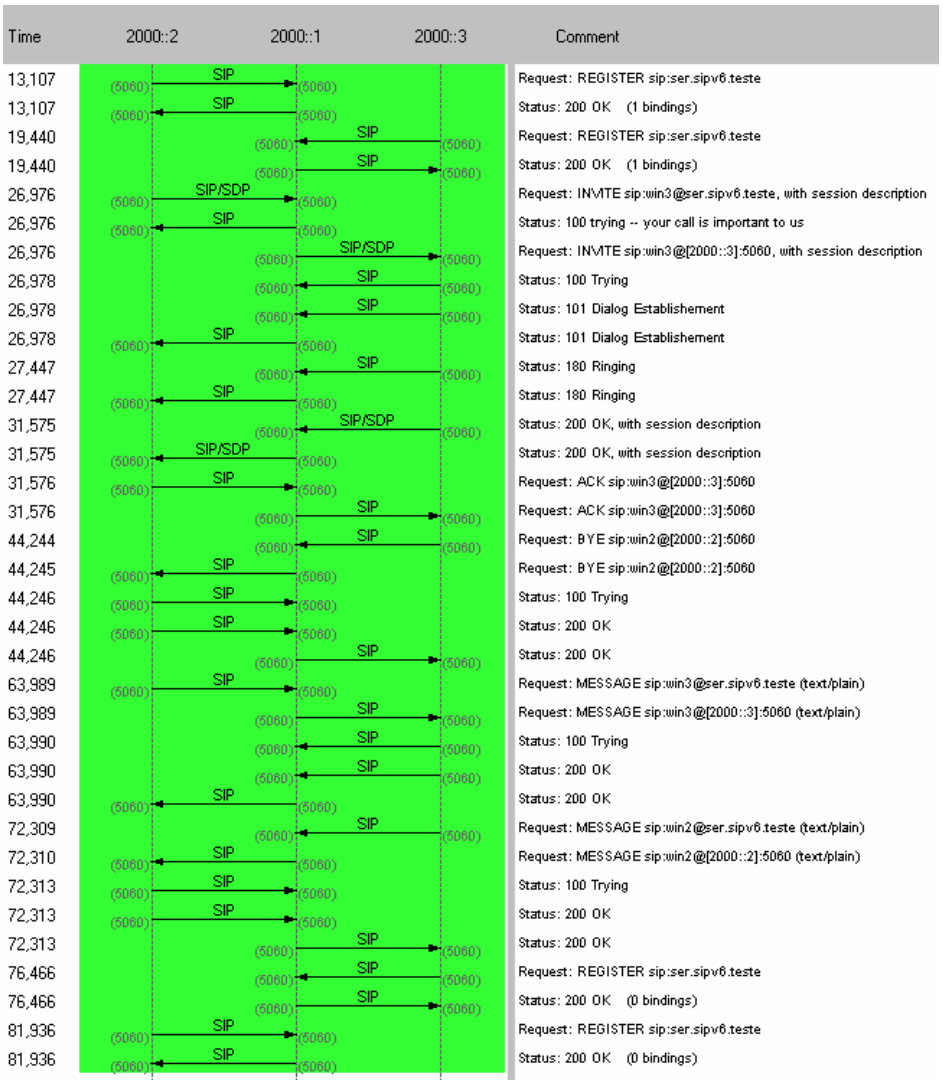


Figura 140 Todas as Mensagens capturadas no servidor SER da IPTEL

8.16. Manual de instalação do SER da IPTEL

Passos e comandos no *linux*:

-verificar se está instalado a versão > 9.2.5 do bind
[root@ser ~]# rpm -q bind
bind-9.2.5-3

-installar o DNS
instalar o yum
export http_proxy=http://proxy.student.estg.ipleiria.pt:8080/
/etc/rc.d/init.d/yum start
yum install bind-chroot.i386
yum list | grep bind

-installar o servidor ser
fazer download do site http://ftp.iptel.org/pub/ser/latest/bin/
ou http://www.iptel.org/ser/download/
gunzip ser-0.9.4_linux_i386.tar.gz
tar xvf ser-0.9.4_linux_i386.tar
para a directoria /usr/local/sbin/

-vi /usr/local/etc/ser/ser.cfg
editar o ficheiro acrescentar as seguintes linhas:
listen=2000::1
listen=192.168.0.1

---------- global configuration parameters ------------------------

#debug=3 # debug level (cmd line: -ddddddddddd)
#fork=yes
#log_stderror=no # (cmd line: -E)

/* Uncomment these lines to enter debugging mode
fork=no
log_stderror=yes
*/

check_via=no # (cmd. line: -v)
dns=no # (cmd. line: -r)
rev_dns=no # (cmd. line: -R)
#port=5060
#children=4
fifo="/tmp/ser_fifo"
listen=2000::1
listen=192.168.0.1
----------------- module loading ----------------------------------

-colocar os ficheiro DNS nas directorias respectivas:
cp -R var-named-chroot-etc/ /var/named/chroot/etc/
cp -R var-named-chroot-var-named/ /var/named/chroot/var/named/

-verificar qual a placa de rede pretendida
-atribuir o endereço IP à interface:
/sbin/ip -6 addr add 2000::1/64 dev eth0
Configurar o endereço ipv4 (ou pelo o comando setup):
/sbin/ip addr add 192.168.0.2/24 dev eth0

-reiniciar o serviço de rede
/etc/init.d/network restart

-coloca em baixo eth1 (se não necessário, é da outra placa de rede)
/sbin/ifconfig eth1 down

-fazer ping6
ping6 2000::1

-iniciar o servidor SER
/usr/local/sbin/ser

-iniciar o serviço DNS
service named restart ou
/etc/rc.d/init.d/named restart

-testar o serviço DNSv6
[root@ser ~]# host -t AAAA ser.sipv6.teste
ser.sipv6.teste has AAAA address 2000::1

-desactivar as firewalls, para fins de teste

8.17. Ficheiros para o DNSv6

os ficheiro que devem fica na pasta var/named/chroot/etc

named.conf

```
//
// named.conf for Red Hat caching-nameserver
//

options {
        directory "/var/named";
        dump-file "/var/named/data/cache_dump.db";
        statistics-file "/var/named/data/named_stats.txt";

        listen-on-v6 { any; };

        /*
         * If there is a firewall between you and nameservers you want
         * to talk to, you might need to uncomment the query-source
         * directive below.  Previous versions of BIND always asked
         * questions using port 53, but BIND 8.1 uses an unprivileged
         * port by default.
         */
         // query-source address * port 53;
};

//
// a caching only nameserver config
//
controls {
        inet 127.0.0.1 allow { localhost; } keys { rndckey; };
};


zone "0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.2.IP6.ARPA." IN {
        type master;
        file "2000::.db";
};
zone "0.168.192.IN-ADDR.ARPA." IN {
        type master;
        file "192.168.0.db";
};
zone "sipv6.teste." IN {
        type master;
        file "sipv6.teste.db";
```

```
};
zone "." IN {
        type hint;
        file "named.ca";
};

zone "localdomain" IN {
        type master;
        file "localdomain.zone";
        allow-update { none; };
};

zone "localhost" IN {
        type master;
        file "localhost.zone";
        allow-update { none; };
};

zone "0.0.127.in-addr.arpa" IN {
        type master;
        file "named.local";
        allow-update { none; };
};

zone "0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.ip6.arpa" IN {
      type master;
        file "named.ip6.local";
        allow-update { none; };
};

zone "255.in-addr.arpa" IN {
        type master;
        file "named.broadcast";
        allow-update { none; };
};

zone "0.in-addr.arpa" IN {
        type master;
        file "named.zero";
        allow-update { none; };
};

include "/etc/rndc.key";
```

Ficheiros que devem ficar na directoria /var/named/chroot/var/named/

2000::.db

```
$TTL 1H
0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.2.ip6.arpa.    SOA    ser.sipv6.teste.
        root.ser.sipv6.teste. (  1
                                                        3H
                                                        1H
                                                        1W
                                                        1H )
                                                        NS      ser.sipv6.teste.
1.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0IN      1H      PTR     ser.sipv6.teste.
2.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0IN      1H      PTR     win2.sipv6.teste.
3.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0IN      1H      PTR     win3.sipv6.teste.
```

192.168.0.db

```
$TTL 1H
@       SOA     ser.sipv6.teste.        root.ser.sipv6.teste. (  1
                                                        3H
                                                        1H
                                                        1W
                                                        1H )
                NS      ser.sipv6.teste.
1       IN      1H      PTR     ser.sipv6.teste.
2       IN      1H      PTR     win2.sipv6.teste.
3       IN      1H      PTR     win3.sipv6.teste.
```

sipv6.teste.db

```
$TTL 1H
@       SOA     ser     root.ser.sipv6.teste. (  1
                                                        3H
                                                        1H
                                                        1W
                                                        1H )
                        NS      ser
ser     IN      1H      A       192.168.0.1
        IN      1H      AAAA    2000::1
win2    IN      1H      A       192.168.0.2
        IN      1H      AAAA    2000::2
win3    IN      1H      A       192.168.0.3
        IN      1H      AAAA    2000::3
```

8.18. **Manual de instalação do Linphone**

Passos e comandos no linux:

-Fazer *download* do site
http://simon.morlat.free.fr/download/1.2.x/source/
ou http://www.linphone.org/

Dos seguintes ficheiros

ilbc-rfc3951.tar.gz
libosip2-2.2.2.tar.gz
linphone-1.2.0.tar.gz

outro ficheiro necessário o speex versão 1.0.5
http://www.speex.org/download/speex-1.0.5.tar.gz

-criar uma pasta linphone_src
[root@localhost ~]# cd linphone_src/
[root@localhost linphone_src]# ls
ilbc-rfc3951.tar.gz linphone-1.2.0.tar.gz
libosip2-2.2.2.tar.gz linphone-plugin-ilbc-1.2.0.tar.gz
[root@localhost linphone_src]# gunzip *
-fazer o tar xvf cada tar

Para cada directoria é necessário compliar e instalar cada biblioteca **ilbc-rfc3951, libosip2-2.2.2,** e o programa **linphone-1.2.0.**
para isso necessário executar os comandos em cada pasta.
./configure
make
make install

-Ao compliar o linphone-1.2.0 poderá aparecer o seguinte erro:
Checking for speex_encode_int in -lspeex... no
configure: error: Could not find a libspeex version that have the speex_encode_int() function. Please install libspeex=1.0.5 or libspeex>=1.1.6

-Se acontecer seguir os seguintes passos:
/usr/bin/updatedb
locate speex
poderão ser necessário eliminar o ficheiro /lib/speex.so. antigo
ou fazer rpm -e speex
rm /usr/local/lib/libspeex.*
rm /usr/lib/libspeex.*

-Compliar e instalar o speex
speex-1.0.5.tar

cd speex-1.0.5
./configure
make
make install

atribuir o endereço IP,
ter o cuidado de verificar no caso de existirem mais do que uma placa de rede
/sbin/ip -6 addr add 2000::2/64 dev eth1
/sbin/ip addr add 192.168.0.2/24 dev eth1
/etc/init.d/network restart

-testar o ping6 2000::1

[root@localhost speex-1.0.5]# ping6 2000::1
PING 2000::1(2000::1) 56 data bytes
64 bytes from 2000::1: icmp_seq=0 ttl=64 time=1.20 ms
64 bytes from 2000::1: icmp_seq=1 ttl=64 time=0.264 ms

-testar o ping 192.168.0.1

[root@localhost speex-1.0.5]# ping 192.168.0.1
PING 192.168.0.1 (192.168.0.1) 56(84) bytes of data.
64 bytes from 192.168.0.1: icmp_seq=0 ttl=64 time=0.192 ms
64 bytes from 192.168.0.1: icmp_seq=1 ttl=64 time=0.281 ms

-testar DNS

[root@localhost speex-1.0.5]# host -t AAAA ser.sipv6.teste
ser.sipv6.teste has AAAA address 2000::1

-correr o linphone
/root/linphone_src/linphone-1.2.0/gnome
./linphone

Há a opção de instalar o linphone para linha de comando, para o caso de não ter o gnome instalado (libgnomeui-dev). Chamado o **linphonec.**

Em vez de fazer ./configure é necessário indicar os parâmetros

Comando para disactivar ter a versão linha de comando do linphone é a seguinte
./configure --disable-glib --disable-gnome_ui

Configuração do linphone na interface GUI

O linphone irá detectar automática uma rede IPv6, no entanto é necessário indicar a utilização do linphone para IPv6 por meio de uma checkbox, como indicada na figura seguinte:

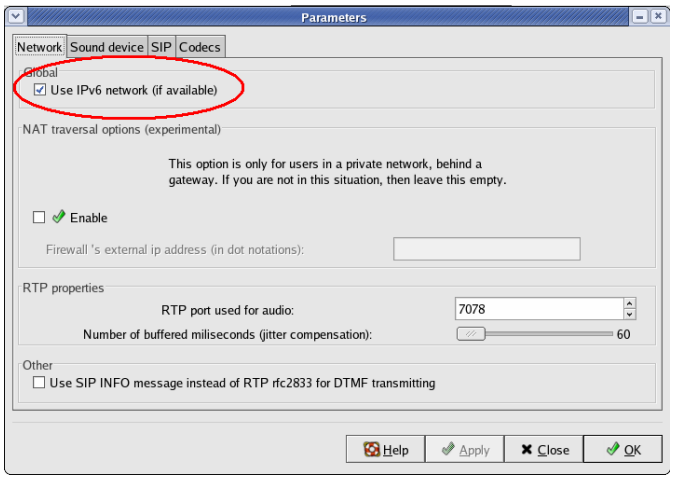


Figura 141 Indicação para utilizar o protocolo IPv6

Depois é necessário indicar o SIP URI e o endereço do servidor
por exemplo: sip:ser.sipv6.teste.

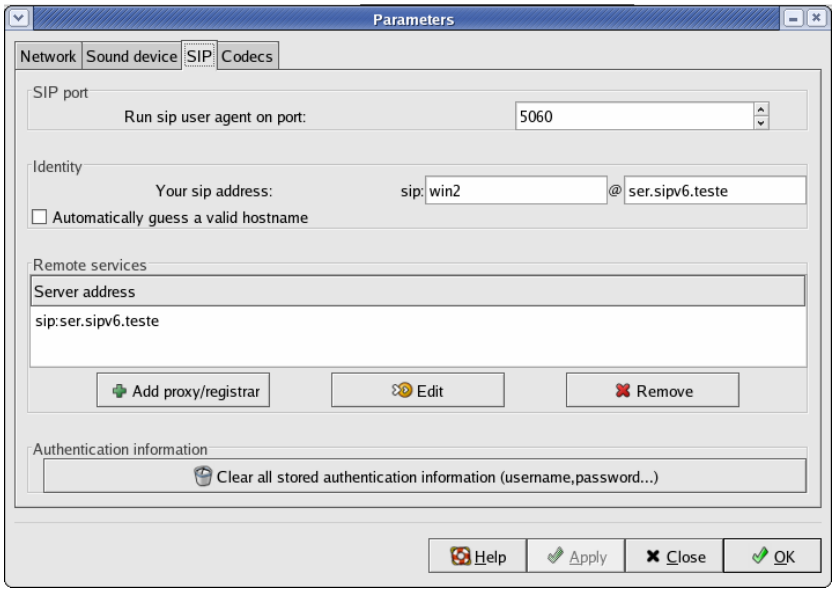


Figura 142 Configuração do endereço do servidor SIP e o SIP URL

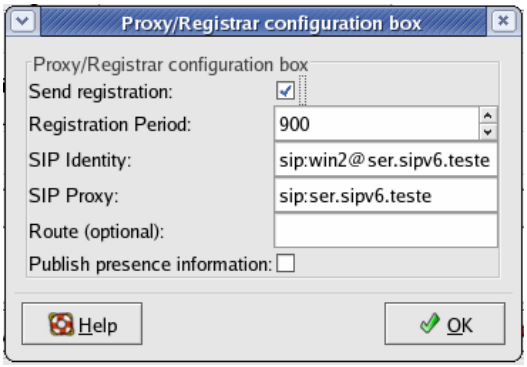


Figura 143 Configuração do URL do utilizado e do Servidor SIP

Para a realização duma chamada é necessário o linphone esteja registado com sucesso.

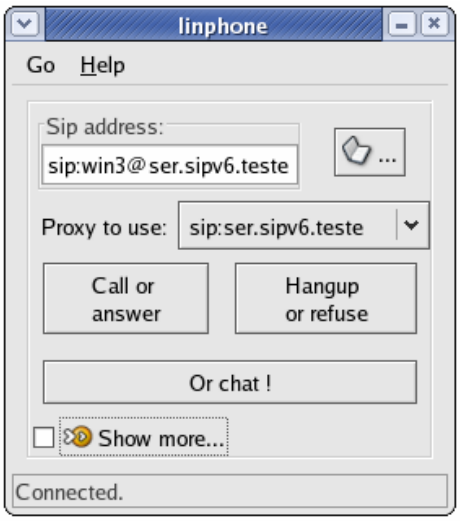


Figura 144 Realização de uma chamada para utilizador com o URL win3@ser.sipv6.teste

O Linphone também permite Chat, como é mostrado na figura seguinte:

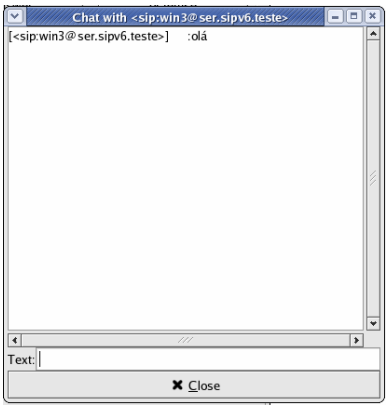


Figura 145 Janela de *Chat*

8.19. Registo do softphone no SER

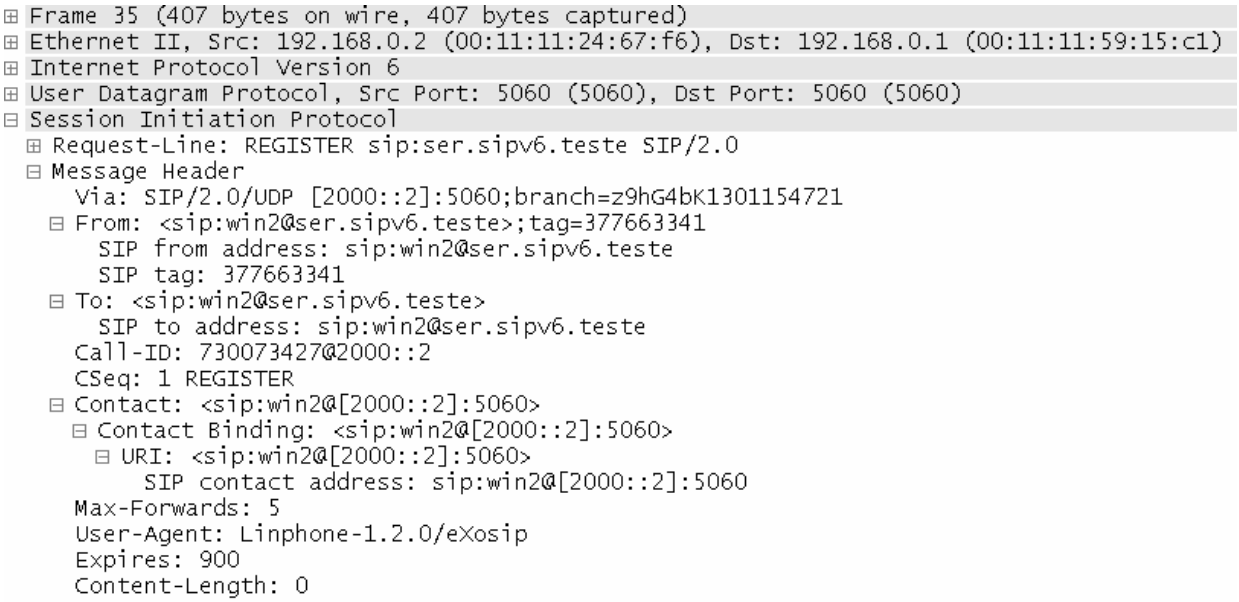


Figura 146 Pedido de registo ao servidor

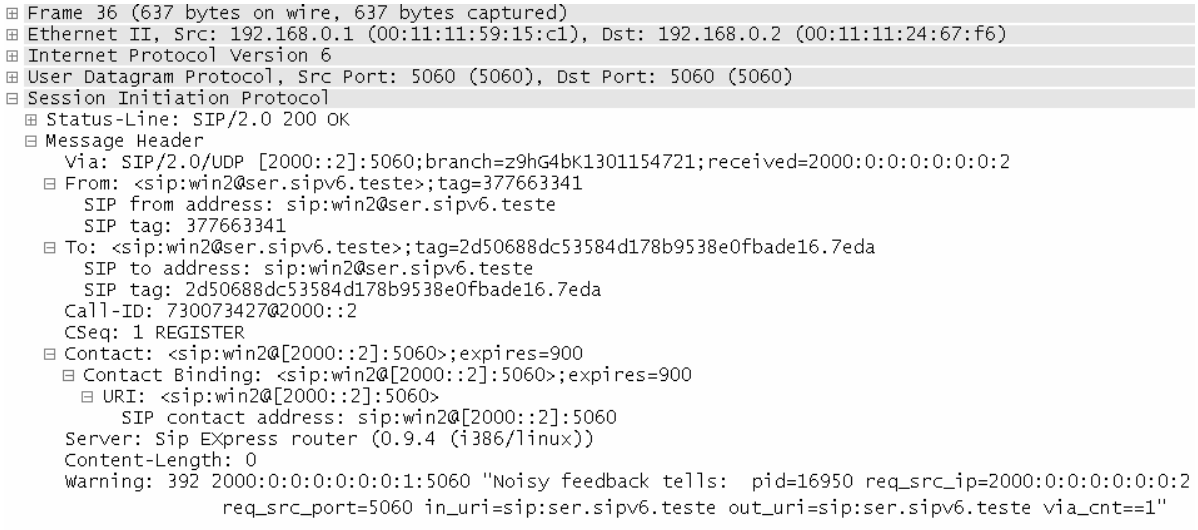


Figura 147 Resposta do servidor

8.20. Terminação da sessão com o SER

Figura 148 Mensagem REGISTER, para termina a sessão com o SER

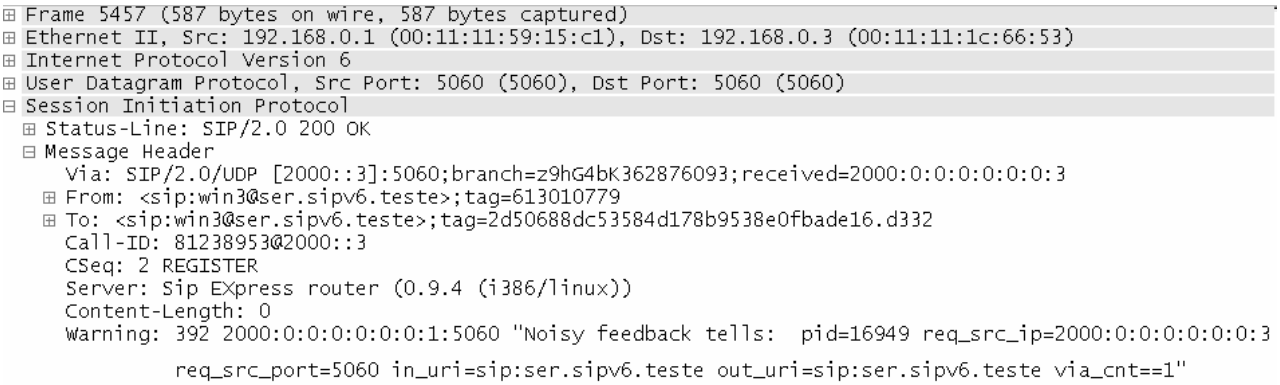
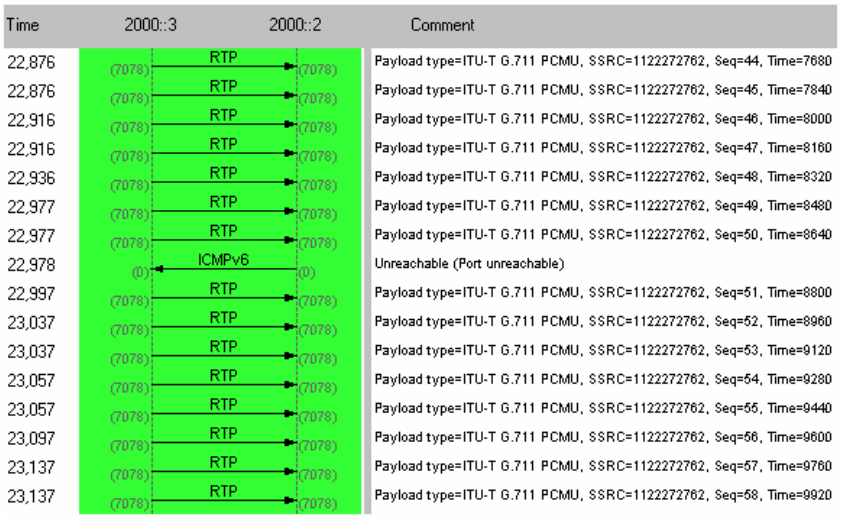


Figura 149 Mensagem SIP para aceitar a terminação da sessão do win3

8.20.1. Problemas anteriores com o Linphone versão 1.1

Na versão 1.1 do Linphone não era possível a comunicação em ambos os sentidos, ou seja, havia uma comunicação em *half-duplex*. Concluiu-se que era um *bug* na camada de aplicação uma vez que o ping funcionou. Em redes a maior parte dos problemas encontram-se na camada física, rede e aplicação.



Time	2000::3	2000::2	Comment
22,876	(7078) RTP	(7078)	Payload type=ITU-T G.711 PCMU, SSRC=1122272762, Seq=44, Time=7680
22,876	(7078) RTP	(7078)	Payload type=ITU-T G.711 PCMU, SSRC=1122272762, Seq=45, Time=7840
22,916	(7078) RTP	(7078)	Payload type=ITU-T G.711 PCMU, SSRC=1122272762, Seq=46, Time=8000
22,916	(7078) RTP	(7078)	Payload type=ITU-T G.711 PCMU, SSRC=1122272762, Seq=47, Time=8160
22,936	(7078) RTP	(7078)	Payload type=ITU-T G.711 PCMU, SSRC=1122272762, Seq=48, Time=8320
22,977	(7078) RTP	(7078)	Payload type=ITU-T G.711 PCMU, SSRC=1122272762, Seq=49, Time=8480
22,977	(7078) RTP	(7078)	Payload type=ITU-T G.711 PCMU, SSRC=1122272762, Seq=50, Time=8640
22,978	(0) ICMPv6	(0)	Unreachable (Port unreachable)
22,997	(7078) RTP	(7078)	Payload type=ITU-T G.711 PCMU, SSRC=1122272762, Seq=51, Time=8800
23,037	(7078) RTP	(7078)	Payload type=ITU-T G.711 PCMU, SSRC=1122272762, Seq=52, Time=8960
23,037	(7078) RTP	(7078)	Payload type=ITU-T G.711 PCMU, SSRC=1122272762, Seq=53, Time=9120
23,057	(7078) RTP	(7078)	Payload type=ITU-T G.711 PCMU, SSRC=1122272762, Seq=54, Time=9280
23,057	(7078) RTP	(7078)	Payload type=ITU-T G.711 PCMU, SSRC=1122272762, Seq=55, Time=9440
23,097	(7078) RTP	(7078)	Payload type=ITU-T G.711 PCMU, SSRC=1122272762, Seq=56, Time=9600
23,137	(7078) RTP	(7078)	Payload type=ITU-T G.711 PCMU, SSRC=1122272762, Seq=57, Time=9760
23,137	(7078) RTP	(7078)	Payload type=ITU-T G.711 PCMU, SSRC=1122272762, Seq=58, Time=9920

Figura 150 *Stream* RTP apenas num sentido na versão Linphone 1.1

Dava um erro de ICMPv6 port unreachable.

```
⊞ Frame 203 (282 bytes on wire, 282 bytes captured)
⊞ Ethernet II, Src: 192.168.0.2 (00:11:11:24:67:f6), Dst: Intel_58:f1:c0 (00:11:11:58:f1:c0)
⊟ Internet Protocol Version 6
    Version: 6
    Traffic class: 0x00
    Flowlabel: 0x00000
    Payload length: 228
    Next header: ICMPv6 (0x3a)
    Hop limit: 64
    Source address: 2000::2
    Destination address: 2000::3
⊟ Internet Control Message Protocol v6
    Type: 1 (Unreachable)
    Code: 4 (Port unreachable)
    Checksum: 0x4ca9 [correct]
  ⊟ Internet Protocol Version 6
      Version: 6
      Traffic class: 0x00
      Flowlabel: 0x00000
      Payload length: 180
      Next header: UDP (0x11)
      Hop limit: 64
      Source address: 2000::3
      Destination address: 2000::2
  ⊟ User Datagram Protocol, Src Port: 7078 (7078), Dst Port: 7078 (7078)
      Source port: 7078 (7078)
      Destination port: 7078 (7078)
      Length: 180
      Checksum: 0xe2e5 [correct]
  ⊟ Real-Time Transport Protocol
    ⊞ [Stream setup by SDP (frame 82)]
      10.. .... = Version: RFC 1889 Version (2)
      ..0. .... = Padding: False
      ...0 .... = Extension: False
      .... 0000 = Contributing source identifiers count: 0
      0... .... = Marker: False
      Payload type: ITU-T G.711 PCMU (0)
      Sequence number: 100
      Timestamp: 16640
      Synchronization Source identifier: 1122272762
      Payload: 7C7C7C7C7C7C7C7E7CF47C757E7B7C7B7C7C7C7C7B7C7B7C...
```

Figura 151 ICMPv6 Port unreachable

Depois ao utilizar a versão 1.2 do Linphone funcionou à primeira, dando uma comunicação *full-duplex*.

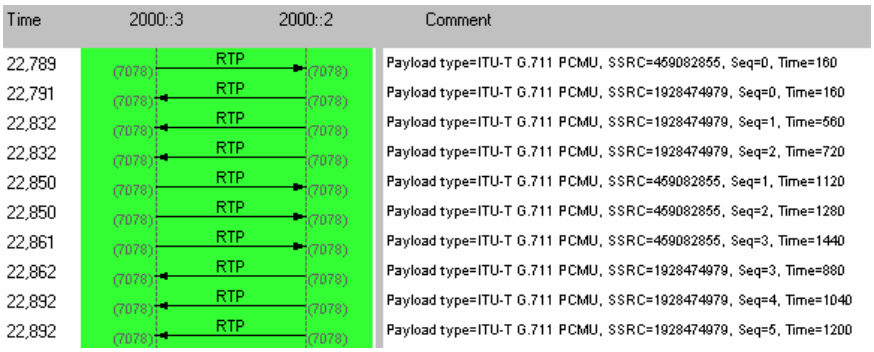


Figura 152 *Streams* RTP em ambos sentidos WIN2 na versão Linphone 1.2

8.20.2. Mensagens DNSv6

A seguir é mostrado o pedido e resposta DNS para IPv6.

```
⊞ Frame 6 (75 bytes on wire, 75 bytes captured)
⊞ Ethernet II, Src: 192.168.0.2 (00:11:11:24:67:f6), Dst: 192.168.0.1 (00:11:11:59:15:c1)
⊞ Internet Protocol, Src: 192.168.0.2 (192.168.0.2), Dst: 192.168.0.1 (192.168.0.1)
⊟ User Datagram Protocol, Src Port: 32773 (32773), Dst Port: domain (53)
    Source port: 32773 (32773)
    Destination port: domain (53)
    Length: 41
    Checksum: 0x818e [incorrect, should be 0x3e44]
⊟ Domain Name System (query)
    Transaction ID: 0x747c
  ⊞ Flags: 0x0100 (Standard query)
    Questions: 1
    Answer RRs: 0
    Authority RRs: 0
    Additional RRs: 0
  ⊟ Queries
    ⊟ ser.sipv6.teste: type AAAA, class IN
        Name: ser.sipv6.teste
        Type: AAAA (IPv6 address)
        Class: IN (0x0001)
```

Figura 153 Pedido DNSv6 do nome ser.sipv6.teste

O servidor DNS responde com o endereço IPv6 2000::1.

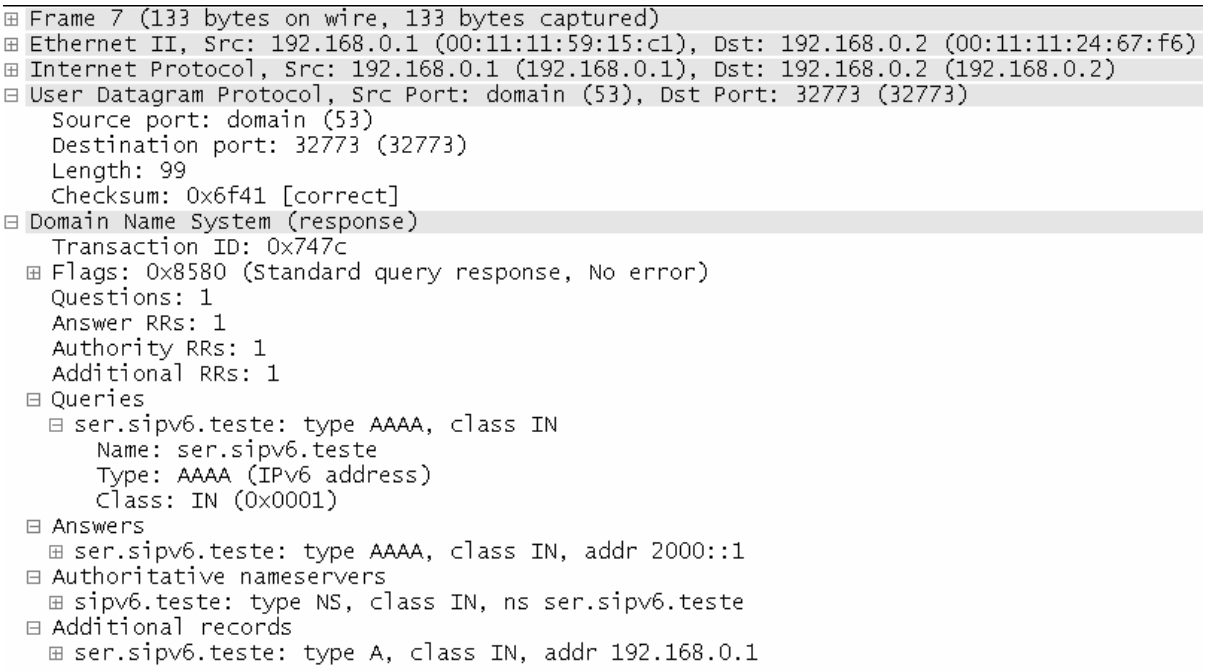


Figura 154 Resposta com o endereço IPv6 2000::1

Na figura seguinte é possível verifica a mensagem SDP.

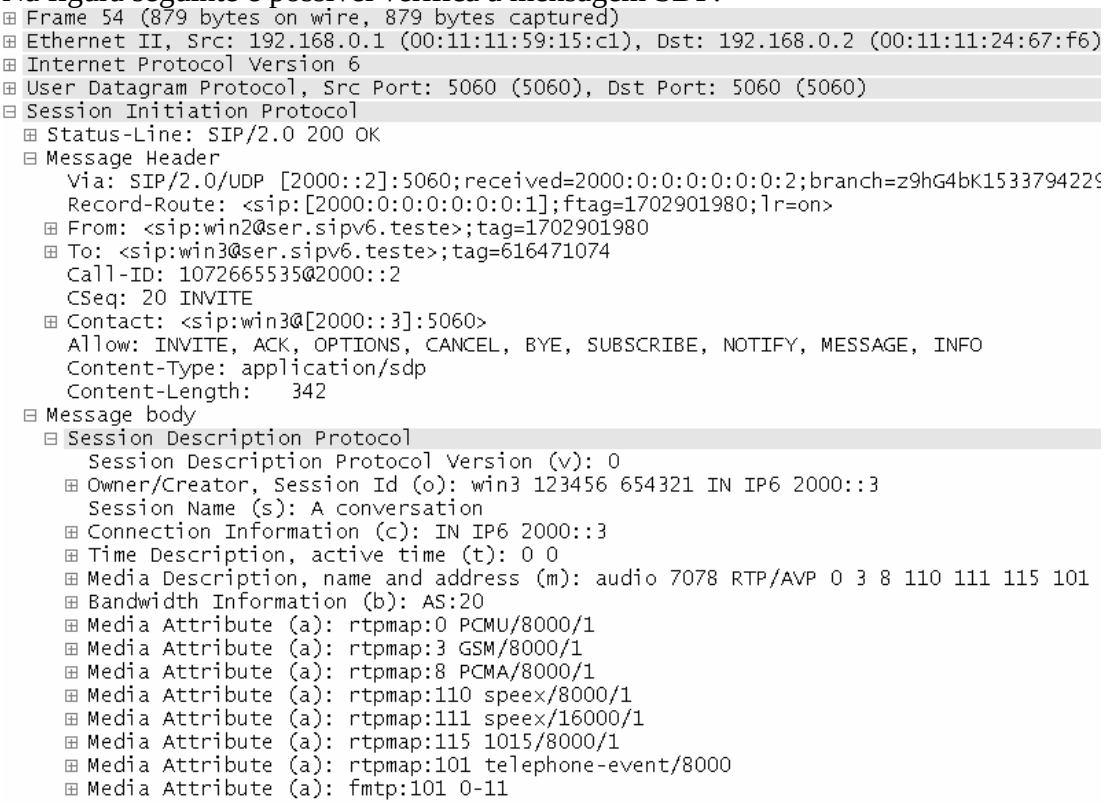


Figura 155 Mensagem SIP/SDP

Verifica-se que os pacotes não estão marcados pelo o linphone, e que é utilizado o codec G.711.

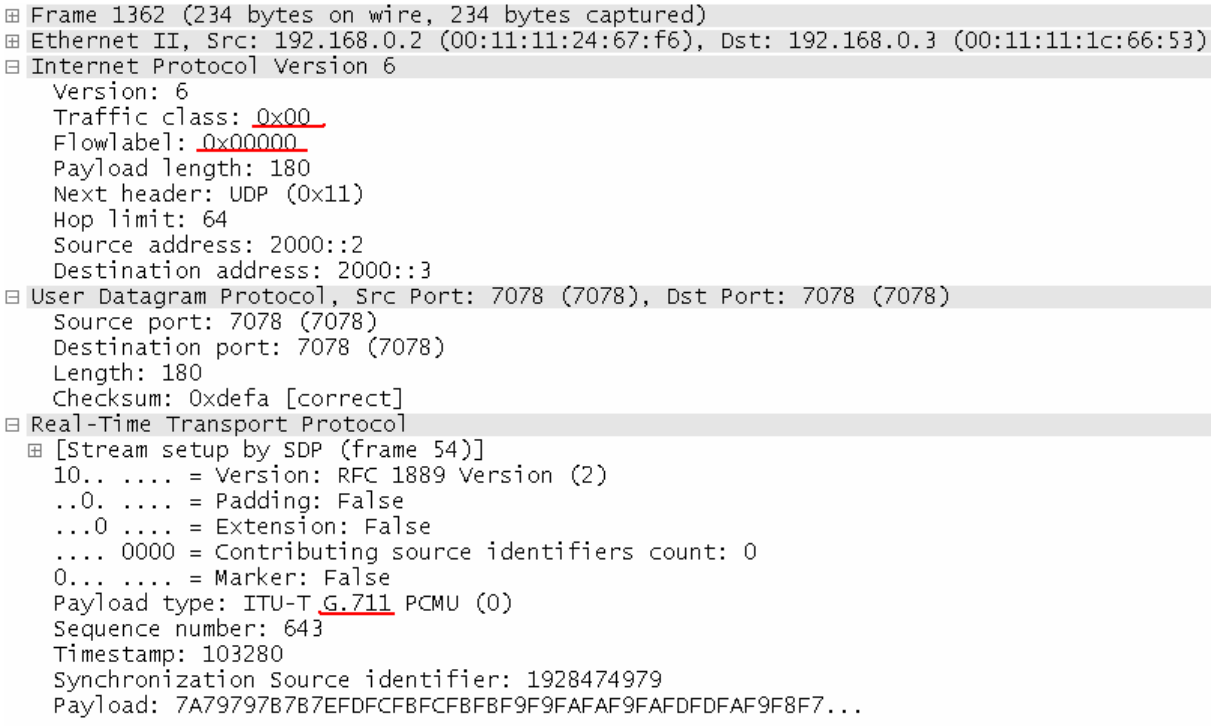


Figura 156 Pacote RTP capturado no Ethreal

9. Bibliografia

[1] http://www.voipreview.org/news.details.aspx?nid=51

[2] http://www.numberingplans.com

[3] http://www.icp.pt/template25.jsp?categoryId=38944

[4] http://www.protocols.com/pbook/h323.htm#RTP

[5] Currículo FWL da Cisco

[6] http://www.protocols.com/pbook/h323.htm#H245

[7] http://www.javvin.com/

[8] http://www.itu.int/itudoc/itu-t/rec/h/h245.html

[9] http://www.protocols.com/pbook/Skinny

[10] http://www.cisco.com/univercd/cc/td/doc/product/voice/c_callmg/4_1/srnd4_1/ipt4_1/41nstrct.pdf

[11] http://www.cisco.com/en/US/products/sw/voicesw/ps556/products_tech_note09186a0080129d92.shtml

[12] http://www.ipv6.rennes.enst-bretagne.fr/dstm/

[13] http://staff.csc.fi/~psavola/residential.html

[14] livro Internet Performance Survival Guide by Geoff Huston, John Wiley & Sons, published February 2000. http://www.potaroo.net/books/index.html

[15] http://www.circleid.com/article/246_0_1_0_C/

[16] [Brian E. Carpenter and Kathleen Nichols, "Differentiated Services in the Internet", Proceedings of the IEEE, Vol. 90, No. 9, September 2002, pp. 1480] http://dynamo.ecn.purdue.edu/~cath/ee647/notes.html

[17] http://www.cisco.com/warp/public/cc/pd/iosw/ioft/iofwft/prodlit/difse_wp.htm

[18] http://www.ripe.net/rs/ (exemplos: ftp://ftp.ripe.net/pub/stats/ripencc/membership/alloclist.txt)

[19] David Luís Santos Serafim,Vítor André Cordeiro dos Santos, IPv6@ESTG-Leiria, Instalação de uma Rede Piloto, ESTG, Leiria, Julho de 2005

[20] Trabalho de sistemas multimedia 4ºano 2ºsemestre de rui Manuel Ferreira Bernardo, Junho de 2005, VoIP

[21] http://playground.sun.com/pub/ipng/html/ipng-implementations.html

[22] RAJAHALME, J.; CONTA, A.; CARPENTER, B.; DEERING, S. – *IPv6 Flow Label Specification (RFC3697)*, 2004, ftp://ftp.rfc-editor.org/in-notes/rfc3697.txt.

[23] http://www.cisco.com/univercd/cc/td/doc/product/software/ios123/123cgcr/ipv6_c/sa_qosv6.htm

[24] http://facweb.cs.depaul.edu/econfer/ect481/Class%20Material/IPv4%20vs%20IPv6.pdf

[25] http://www.cisco.com/en/US/products/ps6610/products_white_paper09186a00800a3e2f.shtml

[26] http://www.ietf.org/internet-drafts/draft-ietf-ipv6-flow-label-07.txt

[27] http://www.circleid.com/article/246_0_1_0_C/

[28] Tipos de mensagens Skinny http://www.cisco.com/univercd/cc/td/doc/product/software/ios123/123newft/123_1/ftskinny.htm#wp1040520

[29] http://en.wikipedia.org/wiki/Session_Initiation_Protocol

[30] http://www.bertolinux.com/voip/english/VoIP-HOWTO.html

[31] http://www.openh323.org/docs/diagrams.html

[32] http://www.cisco.com/en/US/tech/tk652/tk701/technologies_white_paper09186a008009294d.shtml

[33] Foruns da Cisco http://forum.cisco.com/eforum/servlet/NetProf?page=main

[34] Forum sobre IP telefonia IPv4: http://www.iptnetworkers.com/

[35] Notícias http://www.ipv6tf.org/search.php

[36] Realisation of IPv6/IPv4 VoIP Integration Scenarios da 6NET, P.O'Hanlon (UCL), S.Varakliotis (UCL), R.Ruppelt (FhG), J.Fiedler (FhG) 30th January 2005 ,http://www.6net.org/

[37] http://www.bt.com/ipv6/

[38] http://www.eurescom.de/~public-webspace/P1000-series/P1009/doc3_1.html

[39] http://www.voip.nce.ufrj.br/leandro/download/slides-Mestrado2005.pdf

[40] http://www.newport-networks.com/whitepapers/voip-bandwidth1.html

[41] IP *voice solutions*, Paul Hanna (http://www.infj.ulst.ac.uk/~bcs/voip.pdf

[42] http://www.cisco.com/en/US/tech/tk652/tk698/technologies_tech_note09186a0080094ae2.shtml

[43] Relatório de Projecto Informático I, Eng.ª Informática e Comunicações, 3º ano 1º semestre, Implementação de Mecanismos de Diferenciação de Tráfego na Rede da ESTG de Rui Manuel Ferreira Bernardo, Setembro de 2004 (a parte do MGEN)

[44] Naval Research Laboratory (NRL): *The Multi-Generator (MGEN) Toolset*., http://manimac.itd.nrl.navy.mil/MGEN/.

[45] IP Quality of Service, IP Quality of Service Srinivas Vegesna Copyright© 2001 Cisco Press Cisco Press logo is a trademark of Cisco Systems, Inc.

[46] http://www.cisco.com/en/US/tech/tk543/tk757/technologies_tech_note09186a0080103eae.shtml

[47] http://www.okena.com/en/US/products/sw/iosswrel/ps1830/products_feature_guide09186a0080087b13.html#wp9890

[48] http://www.cisco.com/en/US/products/sw/iosswrel/ps1835/products_configuration_guide_chapter09186a00800b75a9.html#46744

[49] http://www.cisco.com/en/US/products/sw/iosswrel/ps1835/products_configuration_guide_chapter09186a00800b75a9.html#1003189

[50] http://www.secretwall.com/Globe/VoiceQoS/qos/content/module3/5_llq.htm

[51] http://en.wikipedia.org/

[52] http://www.cisco.com/en/US/products/sw/voicesw/ps556/products_administration_guide_chapter09186a00800c4ba5.html

[53] http://www.cisco.com/warp/public/126/saa.html

[54] http://ngn.pr-group.ru/library/Applications/VoIP%20Testing.pdf

[55] http://voip.ipb.pt/

[56] http://www.asterisk.org/

[57] http://www.voip.nce.ufrj.br/index_tutorial_openh323_pt_1.htm

[58] http://www.cisco.com/warp/public/105/ipv6tunnel.html

[59] IP Telephony Cookbook, Dr. Margit Brandl, Dimitris Daskopoulos, Erik Dobbelsteijn, Dr. Rosario Giuseppe Garroppo, Jan Janak, Jiri Kuthan, Saverio Niccolini, Dr. J ¨org Ott, Stefan Prelle, Dr. Sven Ubik, Egon Verharen, 9 March 9 2004, http://www.informatik.uni-bremen.de/~prelle/terena/cookbook/main/

[60] Universidade de Trás-os-Montes e Alto Douro,Mestrado em Tecnologias das Engenharias,sIPtel - Um sistema de IPtel com suporte para vídeo,utilizando o protocolo SIP,João Paulo Pereira de Sousa,UTAD, 2003, http://www.ipb.pt/~jpaulo/documentos/Thesis-Joao-Paulo.pdf

[61] Experimental Study of SIP-based VoIPv6 in the Stockholmopen Network, Chengxin Li, 2003-03-03

[62] Vovida Open Communication Application Library, User's Guide,Software Version 1.5.0, Guide Revision 2

[63] Practical VoIP using VOCAL, Luan Dang, Cullen Jennings e David Kelly, O'Reilly, capítulo 14 (sample chapter), http://www.oreilly.com/catalog/voip/

[64] http://www.iptel.org/ser/doc/sip_intro/sip_introduction.html

[65] http://www.cisco.com/warp/public/788/voip/delay-details.html

Documentação para o Cisco Call Manager CCM

[66] http://www.cisco.com/univercd/cc/td/doc/product/voice/c_callmg/4_1/srnd4_1/ipt4_1/41cac.htm

[67] http://www.cisco.com/en/US/products/sw/voicesw/ps556/products_implementation_design_guide_chapter09186a0080447513.html

[68] http://www.cisco.com/univercd/cc/td/doc/product/software/ios120/120newft/120t/120t5/cbwfq.htm

[69] http://www.cisco.com/en/US/tech/tk652/tk698/technologies_tech_note09186a0080094ae2.shtml

Artigos:

- Mohamed A. El-Gendy, Abhijit Bose, and Kang G. Shin, "Evolution of the Internet QoS Support for Soft Real-Time Applications", Proceeding of the IEEE, Vol. 91, No. 7, July 2003, pp. 1086-1104.
- Multimedia Networks and Communication, S. Khanvilkar, F. Bashir, D. Schonfeld, and A. Khokhar, University of Illinois at Chicago. May 2003.
- Review of IPv6 Transition Scenarios for European Academic Networks, Tim Chown, Ming Feng, Mike Saywell, Electronics and Computer Science Department,
- The George Washington University, Department of Computer Science, Fall 2002 - Computer Networking VoIP and Packet Voice Protocols , Seyed Ali Ahmadi,

Livros:

- Administering Cisco Qos in IP networks, Benoit Durand, Jerry Sommerville, Mark Buchmann, Ron Fuller, Syngress
- Integrating Voice and data Networks, Cisco press, Scott Keagy

RFCs relevantes:

- **[RFC 1809]** *Using the Flow Label Field in IPv6*, C. Partridge, June 1995
- **[RFC 2460]** *Internet Protocol, Version 6 (IPv6) Specification*, S. Deering, R. Hinden, December 1998
- **[RFC 2474]** *Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers*, S. Blake, F. Baker, D. Black, December 1998
- **[RFC 2475]** *An Architecture for Differentiated Services*, S. Blake, D. Black, M. Carlson E. Davies, Z. Wang, W. Weiss, December 1998
- **[RFC 3311]** The Session Initiation Protocol (SIP) UPDATE Method, J. Rosenberg September 2002
- **[RFC 3697]** *IPv6 Flow Label Specification*, J. Rajahalme, A. Conta, B. Carpenter, S. Deering, March 2004
- **[RFC1701]** *Generic Routing Encapsulation (GRE)* S. Hanks, T. Li, D. Farinacci, P. Traina, October 1994
- **[RFC2396]** *Uniform Resource Identifiers (URI): Generic Syntax*, T. Berners-Lee, R. Fielding, U.C. Irvine, L. Masinter
- **[RFC2784]** *Generic Routing Encapsulation* D. Farinacci, T. Li, S. Hanks, D. Meyer, P. Traina, March 2000
- **[RFC2893]** *Transition Mechanisms for IPv4 Hosts and Routers*, R. Gilligan, E. Nordmark, IETF, August 2000
- **[RFC2976]** *The SIP INFO Method*, S. Donovan, October 2000,
- **[RFC3056]** *Connection of IPv6 Domains via IPv4 Clouds*. IETF, B. Carpenter, K. Moore, February 2001

- **[RFC3261]** *SIP: Session Initiation Protocol*, J. Rosenberg, H. Schulzrinne,G. Camarillo,A. Johnston,J. Peterson,R. Sparks,M. Handley,E. Schooler,June 2002
- **[RFC3489]** STUN - *Simple Traversal of User Datagram Protocol (UDP) Through Network Address Translators (NATs),* J. Rosenberg, J. Weinberger,C. Huitema,R. Mahy, March 2003
- **[RFC3508]** *H.323 Uniform Resource Locator (URL) Scheme Registration*, O. Levin, April 2003
- **[RFC3665**] *Session Initiation Protocol (SIP) Basic Call Flow Examples*, A. Johnston S., Donovan, C. Cunningham, K. Summers, December 2003
- **[RFC3880]** Call *Processing Language (CPL): A Language for User Control of Internet Telephony Services*, J. Lennox, X. Wu, H. Schulzrinne, October 2004

Para a pesquisa dos RFC foi utilizado o seguinte *link:*

- http://www.rfc-editor.org/rfcsearch.html