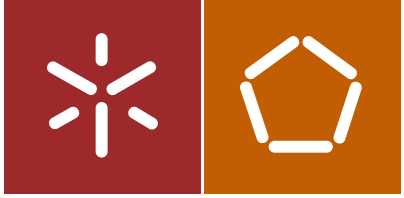
Universidade do Minho
Escola de Engenharia

Raquel Sofia Lima Pereira

Análise de desempenho e usabilidade em sistemas VoIP seguros

janeiro de 2015

Raquel Sofia Lima Pereira Análise de desempenho e usabilidade em sistemas VoIP seguros
UMinho | 2015



Universidade do Minho
Escola de Engenharia

Raquel Sofia Lima Pereira

Análise de desempenho e usabilidade em sistemas VoIP seguros

Dissertação de Mestrado
Ciclo de Estudos Integrados Conducentes ao Grau de
Mestre em Engenharia de Telecomunicações e Informática

Trabalho efetuado sob a orientação do
Professor Doutor António Costa

janeiro de 2015

DECLARAÇÃO

Nome: Raquel Sofia Lima Pereira

Endereço eletrónico: a56929@alunos.uminho.pt

Telefone: +351919047218

Número de Cartão de Cidadão: 13598667

Título da dissertação: Análise de desempenho e usabilidade em sistemas VoIP seguros

Ano de conclusão: 2015

Orientadores: Prof. Dr. António Costa

Designação do Mestrado: Mestrado Integrado em Engenharia de Telecomunicações e Informática

Departamento de Informática

Guimarães, __ /__/____

Assinatura:_____

Agradecimentos

Gostava de agradecer em primeiro lugar ao meu orientador, Professor António Costa, pela árdua tarefa que teve em me orientar. Pois mesmo com trabalho em demasia, aceitou o desafio e mostrou-se sempre disponível. Sei que este meu obrigada, nunca será suficiente, quando comparado a tudo que fez para me ajudar a atingir a difícil meta.

Como também não podia deixar de ser, gostaria de agradecer aos meus pais, pois sem eles nunca tinha chegado até aqui. Á minha mentora, a minha mãe, a quem dedico esta dissertação, quer pelo apoio incondicional que me deu, quer por nunca ter deixado de acreditar em mim. Agradeço também, a toda a família pelos bons conselhos.

Agradeço aos meus tios, Serafim e Alzira, que durante todo o meu percurso académico me acolheram em sua casa com muito amor e carinho.

Ao Luis, o meu namorado, pela difícil tarefa de me aturar, pois esteve sempre do meu lado, dando-me sempre força para continuar e superar todas as adversidades.

A todos os professores, que contribuíram para a minha formação profissional e pessoal.

Por fim, mas não menos importante, a todos o meus amigos. Foram eles, que nos dias mais árduos do meu percurso, me ajudaram a não desistir. Gostaria de lhes agradecer por todos o momentos em que me fizeram sorrir e pelo companheirismo. Obrigada por terem feito parte da minha vida, levo-vos no meu coração.

Resumo

A tecnologia *VoIP* (*Voice over Internet Protocol*) encontra-se num elevado crescimento, que não se encontra ligado à segurança, ou pelo menos são poucos os que se preocupam com isso. Para que seja garantida a confidencialidade, a autenticidade, a integridade e a disponibilidade, considerou-se que a aplicação dos protocolos *TLS* (*Transport Layer Security*) e *IPSec* (*IP Security*) são atualmente as principais soluções disponíveis. Pois o *TLS* implementa uma camada de transporte segura e o *IPSec*, em modo túnel, fornece segurança ao nível da camada de rede IP.

Nesta dissertação propõe-se um mecanismo de avaliação de desempenho, em ambiente real, de sistemas VoIP seguros, com monitorização do impacto nos parâmetros de qualidade de serviço. A arquitetura do sistema de avaliação é totalmente baseada em ferramentas de código aberto, cuja instalação e configuração se descreve de modo a que possa ser facilmente repetida. Das mais variadas ferramentas usou-se o Asterisk, que permitiu a criação de uma central telefónica. Numa fase inicial, foram usados os *softphones* que posteriormente foram substituídos pelo *software SIPp*. O CACTI permitiu visualizar graficamente, os comportamentos das máquinas. Para uma maior flexibilidade e realismo dos testes, foram adicionados, com a ajuda do *dummynet*, os parâmetros tradicionais da internet, no que diz respeito ao *delay*, *packet loss* e *bandwidth*. Para a captura dos pacotes, a serem analisados posteriormente, usou-se o *Wireshark*. Como também não podia deixar se ser, a tecnologia *WebRTC* também é aqui estudada e avaliada.

Os resultados obtidos em cenários típicos, mostram que a introdução de mecanismos de segurança dificulta não só a sua utilização pelo utilizador comum, normalmente confrontado com dados e terminologia especifica, mas dificulta também a instalação, monitorização e avaliação. O sistema proposto provou-se funcional e adaptável a novos cenários.

Abstract

The VoIP (Voice over Internet Protocol) is in a high growth, yet not connected to security, or at least there are few who care about it. In order to provide guarantees of confidentiality, authenticity, integrity and availability it was considered that the application of TLS (Transport Layer Security) and IPSec (IP Security) are currently the main solutions available. Since TLS implements a secure transport layer and the IPSec, in mode tunnel, provides security at the IP network layer.

This thesis proposes a performance evaluation mechanism, of VoIP secure systems, in a real environment, with integrated monitoring of the impact on service quality parameters. The architecture of the proposed evaluation system is entirely based on open source tools, whose installation and configuration is here described so that it can be easily repeated in other situations. Including Asterisk, which allowed the creation of a telephone exchange. Initially, softphones were used, later replaced by the SIPp software. The CACTI allowed to graphically visualize the behavior of machines. For greater flexibility and realism of the tests, the traditional parameters of the internet in terms of delay, packet loss and bandwidth, were added with the aid of the Dummynet tool. Wireshark was used for capturing the packets, for later analysis. WebRTC technology is also here studied and evaluated.

The results obtained in typical scenarios, show that the introduction of security mechanisms hinders not only their use by the average user, usually confronted with data and specific terminology, but also hinders the installation, monitoring and evaluation. The proposed system proved to be functional and adaptable to new scenarios.

Índice de conteúdos

Lista de figuras

Lista de tabelas

Lista de abreviaturas

3GPP	3rd Generation Partnership Project
AAA	Authentication, Authorization and Accounting
ACELP	Algebraic Code-Excited Linear Prediction
ACK	Acknowledgement
AES	Advanced Encryption Strandard
AH	Authentication Header
AMR	Adaptive Multi-Rate
API	Application Programming Interface
ATA	Analogue Terminal Adapter
AVP	Audio Video Profile
AVPF	Audio-Visual Profile with Feedback
CBC	Cipher Block Chaining
CODEC	Coder-Decoder or Compressor-Decompressor
CPU	Central Processing Unit
CS-ACELP	Conjugate Structure – Algebraic Code-Excited Linear Prediction
CSR	Certificate Signing Request
CSV	Comma Separated Values
CLC	Close Logical Channel
CLI	Command Line Interface
CRT	Certificate
DiffServ	Differentiated Services

DNS	Domain Name System
DoS	Denial of Service
DTLS	Datagram Transport Layer Security
ESC	End Session Command
ESP	Encapsulation Security Payload
EUA	United States of America
ETSI	European Telecommunications Standards Institute
FIFO	First IN First Out
GNU	General Public License
GSM-EFR	Global System for Mobile Communications – Enhanced Full Rate
GUI	Graphical User Interface
HTML	HyperText Markup Language
HTTP	Hyper Text Transfer Protocol
HTTPS	Hyper Text Transfer Protocol Secure
ICE	Interactive Connectivity Establishment
IETF	Internet Engineering Task Force
iLBC	internet Low Bitrate Codec
IntServ	Integrated Services
IP	Internet Protocol
IPComp	IP Payload Compression
IPDC	Internet Protocol Device Control
PESQ	Perceptual Evaluation of Speech Quality
IPSec	Internet Protocol Security
ISAKMP	Internet Security Association and Key Management Protocol
ISO	International Organization for Standardization

ITU-T	International Telecommunications Union - Telecommunication standardization sector
IKE	Internet Key Exchange
LPC	Linear Predictive Coding
MeGaCo	Media Gateway Control
MGCP	Media Gateway Control Protocol
MIB	Management Information Base
MKI	Master Key Identifier
MOS	Mean Opinion Score
MP-MLQ	Multi-Pulse - Maximum Likelihood Quantization
MR-ACELP	Multi-Rate – Algebraic Code-Excited Linear Prediction
NAT	Network Address Translation
OID	Objet Identifier
OLC	Open Logical Channel
OSI	Open Systems Interconnection
P2P	Peer-to-Peer
PBX	Private Branch Exchange
PCAP	Packet CAPture
PCM	Pulse Code Modulation
PEM	Privacy Enhanced Mail
PFS	Perfect Forward Security
PKCS	Public-Key Cryptography Standards
PSTN	Public Switched Telephone Network
PSK	Pre-Shared Key
PRNG	Pseudo-Random Number Generators

QoS	Quality of Service
RFC	Request For Comments
RTCP	Real-time Transport Control Protocol
RTP	Real-time Transport Protocol
RTSP	Real Time Streaming Protocol
S/MIME	Secure/Multipurpose Internet Mail Extensions
SA	Security Association
SAD	Security Association Database
SAP	Session Announcement Protocol
SAVP	Secure AVP
SBC	Session Border Controller
SCTP	Stream Control Transmission Protocol
SDP	Session Description Protocol
SGCP	Simple Gateway Control Protocol
SIP	Session Initiation Protocol
SLA	Service Level Agreements
SNMP	Simple Network Management Protocol
SPAM	Sending and Posting Advertisement in Mass
SPD	Security Policy Database
SPI	Security Parameter Index
SRTP	Secure Real-time Transport Protocol
SSH	Secure Shell
SSL	Secure Sockets Layer
STUN	Session Traversal Utilities for NAT
TCP	Transmission Control Protocol

TCS	Terminal Capability Set
TLS	Transport Layer Security
UA	User Agent
UAC	User Agent Client
UAS	User Agent Server
UDP	User Datagram Protocol
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
VoIP	Voice over Internet Protocol
VoIPSA	VoIP Security Alliance
VPN	Virtual Private Network
W3	World Wide Web
W3C	World Wide Web Consortium
WebRTC	Web Real-Time Communication
XML	Extensible Markup Language
XMPP	Extensible Messaging and Presence Protocol

1. Introdução

Esta dissertação pretende avaliar o desempenho e usabilidade dos sistemas VoIP em vários cenários, considerando diferentes mecanismos de segurança. Neste capítulo faz-se um pequeno enquadramento do tema (secção 1.1), apresentando-se de seguida os principais objetivos (secção 1.2). Serão também explicados os principais contributos (secção 1.3) e o capítulo termina com a descrição da estrutura da dissertação (seção 1.4).

1.1 Enquadramento

O VoIP (Voice over Internet Protocol), consiste em transmitir mensagens de voz através de uma rede usando o protocolo IP, sendo que esta é transportada sobe a forma de pacotes, ou seja, é transmitida no formato digital e não analógico. No serviço tradicional a mensagem é transportada através de uma rede de comutação de circuitos. Uma das principais razões pelas quais esta tecnologia se tornou muito popular é, devido à existência de software gratuito, pois permite dispensar os serviços de voz das companhias telefónicas, ou seja, da telefonia tradicional chamada de PSTN (*Public Switched Telephone Network*). Um outro factor importante para a popularidade desta tecnologia, é o crescimento do número de locais com acesso a banda larga. Nas chamadas tradicionais, os recursos são reservados durante a chamada e libertados no fim da mesma. Já nas chamadas VoIP, que recorre a uma infraestrutura da rede IP para a realização de chamadas, não há à priori reserva de recursos.

Soluções VoIP proprietárias, como o Skype ou o Google Hangouts, entre outros, popularizaram este serviço mesmo entre os utilizadores da Internet mais casuais. No que concerne aos standards, existem duas famílias de protocolos de referência: o H.323 [1] da ITU-T (*International Telecommunications Union - Telecommunication standardization sector*) e o SIP (*Session Initiation Protocol*) [2] do IETF (*Internet Engineering Task Force*). Embora seja de prever que ambos coexistam por muito tempo, o SIP tem tido um maior crescimento e aceitação e consequentemente maior popularidade. Ultimamente tem vindo a ser preparado no seio do W3C

(*World Wide Web Consortium*) um novo standard para o VoIP via Web, que se designa por *WebRTC* [3]. O objetivo do *WebRTC* é permitir chamadas voz e vídeo a partir de *browsers* Web, quer de *browser* para *browser*, quer de *browser* para telefones.

Um terminal de voz, seja ele de hardware ou software, SIP ou H.323, regista-se inicialmente junto dum servidor. O endereço IP e o nome/número ficam assim disponíveis, para serem localizados, só depois as chamadas podem ser reencaminhadas para os respectivos terminais. Este registo pode ou não exigir autenticação.

Num cenário de intranet, os endereços IP usados podem ser das gamas privadas sem conectividade global. Para que as conexões áudio/vídeo possam atravessar a firewall é necessário recorrer a outros protocolos. O mesmo se passa quando há requisitos de confidencialidade das chamadas e do padrão de chamadas. Um cenário VoIP real é pois mais complexo que a aparente simplicidade do SIP (e protocolos afins) possa efetivamente induzir. E como serviço fundamental numa organização, é necessário testar e monitorizar o seu desempenho.

A questão que se coloca como ponto de partida nesta proposta é avaliar o impacto em termos de desempenho de sistemas VoIP com e sem mecanismos de segurança normalmente adotados. O requisito da segurança tem influências no custo e na qualidade de serviço, e eventualmente também na usabilidade.

1.2 Objetivos

Os principais objetivos desta dissertação podem ser sintetizados nos seguintes pontos:

- Compreender o funcionamento do VoIP e estudo dos protocolos de sinalização mais utilizados, como o H.323 e o SIP;
- Avaliar os vários mecanismos de segurança, possíveis de serem usadas em cenários VoIP, como o HTTP *Digest Authentication*, TLS/DTLS, SRTP e IPSec;
- Avaliar os problemas de QoS, nomeadamente o Atraso, Variação do atraso (*Jitter*) e Largura de banda em diferentes cenários de rede;
- Implementação de um sistema de teste em ambiente real e/ou simulado;
- Conclusão e recomendações futuras para Sistemas VoIP seguros;

1.3 Principais contributos

Sendo o VoIP uma tecnologia muito popular, nos dias de hoje, será de elevado interesse solucionar os diversos problemas ou debilidades, em termos de segurança, que este apresenta. A introdução de mecanismos de segurança torna os serviços VoIP mais difíceis de usar e além de ter impacto no desempenho, são pois contraditórios, pois torna-se muito complicado se no meio de um conversa houver atrasos, ou outros problemas de qualidade na comunicação. O tema de QoS também foi estudado, tendo em conta os diversos mecanismos de segurança usados.

No final desta dissertação, apresentam-se os mecanismos de segurança mais indicados, para as situações em que se pretende ou não confidencialidade, para os níveis de desempenho que se encontrem numa gama aceitável pelo utilizador. A arquitetura do sistema de testes, é totalmente baseada em ferramentas de código aberto, cuja instalação e configuração se descreve de modo a que possa ser facilmente repetida.

1.4 Estrutura da dissertação

Esta dissertação é constituída por oito capítulos. No primeiro capítulo inclui-se uma pequena explicação do que foi feito, sendo apresentados os principais objetivos e explicando um pouco do que é o VoIP. No segundo capítulo faz-se um enquadramento teórico do tema em causa, sendo apresentado, um pouco mais ao detalhe, uma introdução ao VoIP, quais os protocolos de sinalização, de transporte e de controlo usados, codificação, segurança e questões de QoS. Ainda neste capítulo, apresenta-se uma breve explicação do *WebRTC* e alguns trabalhos relacionados com o tema em causa. No terceiro capítulo descrevem-se os procedimentos experimentais iniciais, e apresenta-se o cenário experimental base. O quarto capítulo, apresenta todos os procedimentos efetuados e quais as ferramentas adotadas. São também explicados os diversos componentes intervenientes em todos os testes. No capítulo cinco, incluem-se todos os testes efectuados e os resultados obtidos em cada cenário de teste. Os dados recolhidos são apresentados sob a forma de gráficos. No capítulo seis apresentam-se as principais conclusões e propostas de trabalho futuro. Por último são incluídas as referências utilizadas e um conjunto de apêndices.

2. Voz sobre IP

Este capítulo, está dividido em três secções. Na primeira encontra-se uma explicação da tecnologia VoIP, sendo explicado o seu funcionamento, os protolocos de sinalização de transporte de voz e os protocolos de controlo de *gateway* usados, a segurança e o QoS. A segunda seção, consiste numa breve explicação da tecnologia *WebRTC*. Na última secção serão apresentados e descritos alguns trabalhos relacionados.

2.1 Contexto

O primeiro software VoIP surgiu em 1995 [4], por parte da empresa *VocalTec*, em Israel. Em 1998, sugiram os primeiros sistemas que permitiram a integração do VoIP e telefones convencionais.

Trata-se de uma tecnologia que faz o encaminhamento da voz através de pacotes IP, tornando possível comunicações com fins de uso pessoal e/ou de negócios. Funciona como um sistema tradicional de realização de chamadas, mas a um custo muito mais reduzido. Possui uma infraestrutura composta por terminais, sendo eles os telefones (*hardware* ou *software*), e os nós *gateway* de comunicação, interligados por uma rede IP [5].

Esta tecnologia contém diversas vantagens, como custos reduzidos nas chamadas, a possibilidade de usar uma infraestrutura já existente, portabilidade (bastando apenas ter conectividade à internet), entre outras. Contém no entanto também algumas desvantagens, como uma arquitetura e serviços de rede complexos (devido aos diversos serviços existentes na mesma rede, como voz e dados), problemas de QoS, de segurança e de interação entre as diferentes aplicações [6].

2.1.1 Modo de funcionamento

No VoIP, o telefone (ou *softphone*) captura a voz e passa-a de sinal analógico para digital, que posteriormente é codificada e enviada sob a forma de pacotes, sendo remetidos através da rede IP. No recetor é feito o processo inverso, ou seja, é feito o desempacotamento, seguidamente a descompressão e finalmente a passagem do sinal digital para analógico, por forma a ser "entendido" pelo mesmo (ver Figura 2.1). Estabelece-se uma ligação entre um emissor e um recetor, através da rede IP. Os pacotes de dados são trocados em tempo real, nos dois sentidos, de forma bidirecional [7].

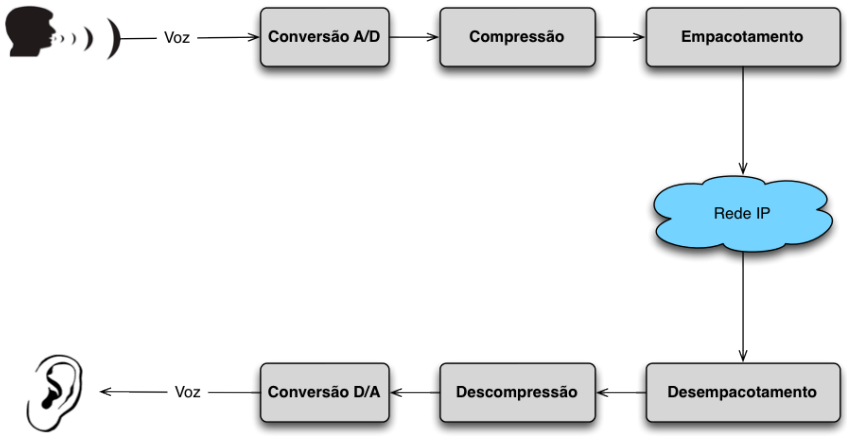


Figura 2.1. Funcionamento do VoIP [adaptado de [7]].

Uma infraestrutura VoIP [8] pode ser dividida em três partes, sendo elas os equipamentos terminais, os componentes da rede IP e o *gateway* VoIP (servidores intermediários) (ver Figura 2.2).

Os equipamentos terminais podem ser telefones ou computadores. De notar que a segurança nestes equipamentos depende também da forma como são instalados.

Os componentes de rede estão localizados na rede IP existente, que por consequência herda as suas vulnerabilidades. Estes elementos podem ser routers*, switches, firewalls, proxy*, entre outros. Cada um deles tem uma preocupação diferente com a segurança. A rede IP é constituída por diversos meios de comunicação muito heterogéneos, como ligações Ethernet, ponto-a-ponto em fibra ou sem fios (*Wireless*).

Por fim, temos o *gateway* VoIP que é o componente responsável pela interligação da rede IP e a rede PSTN. As suas principais funções são as de codificação e descodificação, controlo de sinalização, encaminhamento de chamadas entre as duas redes. Por outras palavras pode dizer-se que permite interoperabilidade entre duas entidades de domínios diferentes, como a rede VoIP e rede telefónica pública [8].

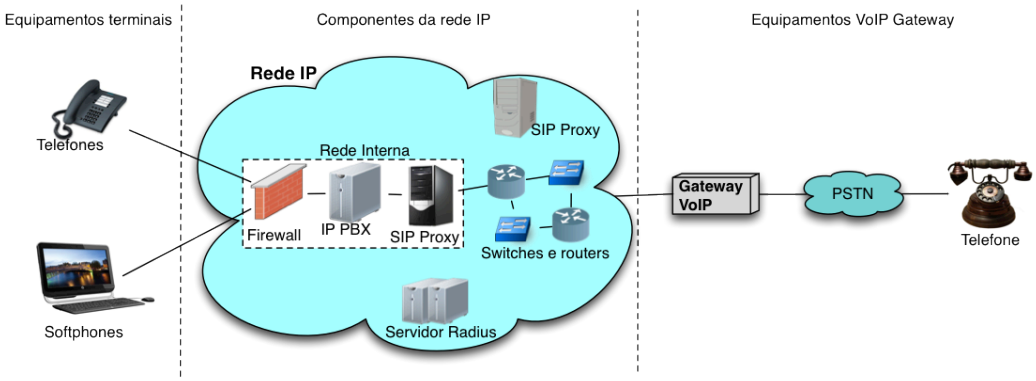


Figura 2.2. Estrutura da rede VoIP [adaptado de [8]].

Para se poder estabelecer uma chamada, é necessário obter um programa do tipo *softphone* em computadores (como o Skype e MSN), ou um telefone com ATA (Adaptador de Telefone Analógico, que interliga a rede IP à rede PSTN) ou um telefone IP (similar ao ATA, sendo aparentemente parecido a um telefone comum).

A tecnologia VoIP faz uso de diversos protocolos que atuam em conjunto (ver Figura 2.3), sendo eles os protocolos de sinalização, transporte e controlo de *Gateway*. Os protocolos de sinalização, são os responsáveis por localizar os utilizadores/terminais na rede, notificar o inicio de uma chamada ou de uma sessão, notificar a aceitação da mesma e da sua terminação. Já no caso dos protocolos de transporte, o objetivo é transportar os pacotes áudio e vídeo. Por fim, no protocolo de controlo de *Gateway* é feita a garantia do QoS e tradução de endereços IP, através do NAT (*Network Address Translator*) [9].

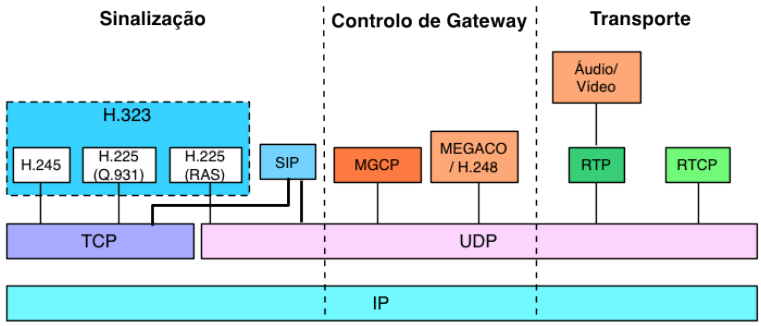


Figura 2.3. Protocolos VoIP [adaptado de [9]].

2.1.2 Codificação e descodificação

Para o envio e receção dos dados, o VoIP necessita de CODECs (*Coder-Decoder*), que podem ser dispositivos ou programas, permitindo a codificação e descodificação de um sinal. Por exemplo, transição do sinal de voz de analógico para digital no emissor, e de digital para analógico no recetor.

Uma das principais funções dos CODECs é a de fazer a conversão do sinal de áudio, por amostragem, milhares de vezes por segundo. Depois converte para digital cada amostra retirada e seguidamente comprime para ser enviado (ver Figura 2.4) [10]. Essa compressão pode ser com ou sem perdas, sendo que depois terá implicações no desempenho. Pois para uma compressão com perdas, a taxa de compressão é mais elevada, mas de notar que as perdas estão localizadas nos sons ou cores quase imperceptíveis, tentando que se consiga um entendimento aceitável. Dado que o ouvido humano só consegue ouvir sons entre os 20Hz e os 20kHz, os sons fora deste intervalo podem ser eliminados. Já a compressão sem perdas é raramente usada, só em casos em que os utilizadores queiram preservar o original, mas haverá um elevado consumo da largura de banda.

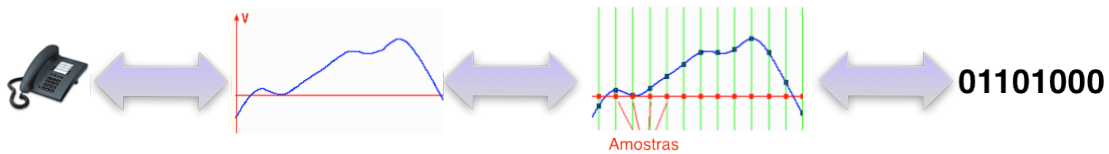


Figura 2.4. Funcionamentos dos *Codecs*.

Segundo [7], os *codecs* mais usados são o G.711, G729, G.723.1, AMR, GSM-EFR e iLBC. Na Tabela 2.1, é apresentada uma breve descrição de cada um deles, sendo referenciado por quem foi desenvolvido, que algoritmo usa, qual a taxa de transferência e por fim se possui uma licença livre ou paga.

Tabela 2.1. Codecs mais usados [7].

Codec	Desenvolvido por	Tipo de compressão	Taxa de transferência (Kbits/s)	Licença
G.711	ITU-T	A-law/μ-law PCM	64	Grátis
G.729	ITU-T	CS-ACELP	8	Paga
G.723.1	ITU-T	MP-MLQ	5.3	Paga
		ACELP	6.3	
AMR	3GPP	MR-ACELP	4.75 – 12.2	Paga
GSM-EFR	ETSI	ACELP	12.2	Paga
iLBC	Global IP Solutions	Block Independent	15.2	Grátis
		LPC	13.33	

Os codecs G.711, G.729 e G.723.1, foram desenvolvidos pelo ITU-T e usam o algoritmo A-law/μ-law PCM (*Pulse Code Modulation*), CS-ACELP (*Conjugate Structure – Algebraic Codebook Excited Linear Predication*) e MP-MLQ (*Multi-Pulse - Maximum Likelihood Quantization*)/ACELP, respectivamente. O G.711 requere uma taxa de transmissão de 64Kbits/seg e com licença grátis. Já o G.729 e o G.723.1 têm as licenças pagas, mas o G.729 exige 8Kbit/s de taxa de transmissão, enquanto o G.723.1 possui 5.3Kbits/s e 6.3Kbits/s, dependendo do algoritmo aplicado.

O AMR (*Adaptive Multi-Rate*), foi desenvolvido pelo 3GPP (*3ª Generation Partnership Project*), usa o algoritmo MR-ACELP (*Multi-Rate – Algebraic Code-Excited Linear Prediction*), a uma taxa de transmissão entre os 4.75Kbits/seg e 12.2Kbits/seg e é pago.

Por fim, existe o iLBC (*internet Low Bitrate Codec*) que foi desenvolvido pelo *Global IP Solutions*. Este, tal como o G.723.1, usa dois tipos de algoritmos, o *Block Independent*, que gera 15.2Kbits/seg e licença paga, e o LPC (*Linear Predictive Coding*), que gera 13.33KBits/seg e licença gratuita.

2.1.3 Protocolos de sinalização

No VoIP podem ser usados dois tipos de sinalização, o SIP e o H.323. Sendo que atualmente, devido à sua simples arquitetura, o mais usado é o SIP [11]. Ambos surgem no mesmo ano, mas foi o H.323 que apareceu primeiro. Estes protocolos serão explicados com mais detalhe nos pontos que se seguem.

2.1.3.1 SIP (Session Initiation Protocol)

Criado em 1996, com o propósito de proporcionar tráfico de voz sobre a rede IP [4], o SIP é um protocolo de sinalização da camada de aplicação, normalizado pelo IETF e está descrito na RFC 3261 [2]. Permite localizar, registar e autenticar os utilizadores. Este protocolo é similar, em termos de estrutura sintática das mensagens protocolares, ao protocolo HTTP, baseado em texto, com a arquitetura baseada no modelo cliente-servidor, onde em cada pedido é gerada pelo menos uma resposta e/ou ação.

Permite aos terminais, em cada sessão, negociar os tipos de dados multimédia, e também a modificação de parâmetros durante a mesma, por exemplo retirando ou acrescentando participantes. O SIP usa formatos de endereços parecidos aos do correio electrónico (*e-mail*), por exemplo "sip:raquel@uminho.pt", sendo o "uminho.pt" o domínio. Estes formatos ajudam á localização, pois com a auxílio do DNS (*Domain Name System*), obtém-se o endereço IP do servidor onde se encontra o utilizador, permitindo localizá-lo com mais facilidade.

Arquitetura

Na interação protocolar SIP é possível identificar cinco entidades distintas, o Agente do Utilizador (*User Agent (UA)*), o Servidor de Registo (*Register Server*), o Servidor de Redireccionamento (*Redirect Server*), o Servidor de Localização (*Location Server*) e o Servidor de Proxy (*Proxy Server*) (ver Figura 2.5).

O Agente do Utilizador é um terminal SIP, que se divide em Agente do Utilizador Cliente *(User Agent Client (UAC))* e em Agente do Utilizador Servidor *(User Agent Server (UAS))*, de acordo com o seu papel na comunicação. O Agente do Utilizador Cliente inicia a sessão,

enviando mensagens e pedidos, enquanto o Agente do Utilizador Servidor atende e responde a essas mesmas mensagens, numa lógica de cliente-servidor.

O Servidor de Registo é um servidor que aceita pedidos de registo de um cliente e atualiza a base dados local, que serve para armazenar informações de contacto. O Servidor de Redireccionamento, aceita pedidos SIP de um cliente, faz o mapeamento do endereço SIP dos destinatário da chamada e retorna o endereço para o cliente. Este servidor não encaminha pedidos para outros servidores. O Servidor de Localização é onde é armazenada a localização dos terminais, Servidor *Proxy* e Servidor de Redireccionamento. O Servidor *Proxy* trata pedidos SIP de fontes UA, mas também pode funcionar como um servidor ou um cliente, para efetuar pedidos em nome dos clientes. Os pedidos são atendidos localmente ou são transmitidos para um outro servidor [9].

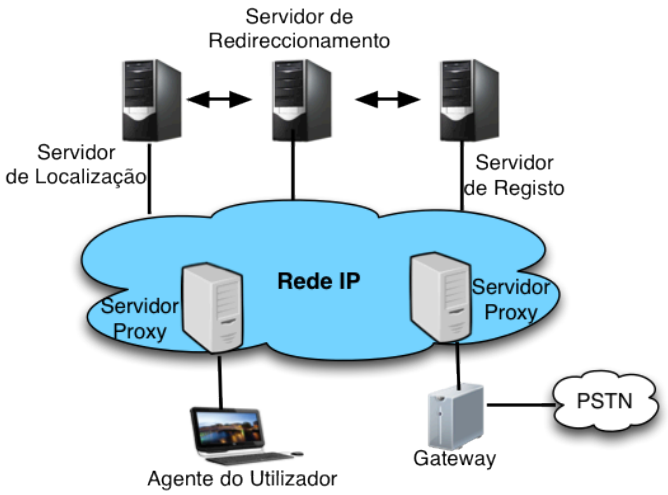


Figura 2.5. Arquitetura do SIP [adaptado de [12]**].**

Estabelecimento de chamadas

Durante o estabelecimento de uma chamada SIP, existe uma troca de mensagens que permite que os utilizadores se registem no Servidor *Proxy*, efetuem pedidos de chamadas, troquem dados e por fim que finalizem a sessão [12].

De acordo com a Figura 2.6, o utilizador *A* e o utilizador *B* registam-se previamente no Servidor *Proxy (REGISTER)*, confirmando aos mesmos que o registo foi efetuado com sucesso (*OK 200*). De seguida o utilizador *A* decide efetuar uma chamada ao utilizador *B*, mas tem de contactar primeiro o *Proxy* (*INVITE*). Por sua vez o *Proxy* confirma a receção da mensagem e

avisa que está a tentar estabelecer a sessão (*TRIYNG 100*). Posteriormente o *Proxy*, envia uma mensagem de convite (*INVITE*) ao utilizador *B*, a pedido do utilizador *A*. O utilizador *B* confirma ao *Proxy* a receção do pedido (*TRIYNG 100*).

Nesta fase o telefone do utilizador *B* começa a tocar e avisa o *Proxy* (*RINGING 180*). E o *Proxy* avisa o utilizador *A* de que o telefone do utilizador *B* já está a tocar (*RINGING 180*). No utilizador *A* é gerado um som de chamada em curso, indicando que tudo foi feito corretamente.

Assim que o utilizador *B* atender a chamada/telefone, será enviada uma mensagem do utilizador *B ao Proxy* (OK 200) e do *Proxy* ao utilizador *A* (OK 200), indicando que o utilizador *B* atendeu o telefone. Para que a chamada seja efetuada com sucesso, nas duas extremidades, será enviada uma mensagem de confirmação (ACK) do utilizador *A* para o *Proxy* e do *Proxy* para o utilizador *B,* indicando que a ligação foi estabelecida com êxito.

Na fase seguinte, serão trocados pacotes RTP/RTCP, tratando-se da troca de dados de voz e/ou vídeo entre as duas extremidades. Assim que um utilizador desejar finalizar a chamada será enviada uma mensagem "BYE" ao *Proxy*. Este por sua vez, envia a mesma mensagem ao utilizador *B*. Para que a finalização seja efetuada corretamente dos dois lados, é enviada uma mensagem "ACK" de utilizador *B* para o *Proxy* e do *Proxy* para o utilizador *A*. Em algumas aplicações, a mensagem de "ACK" depois do "BYE", não é aguardada ou nem sequer enviada [13].

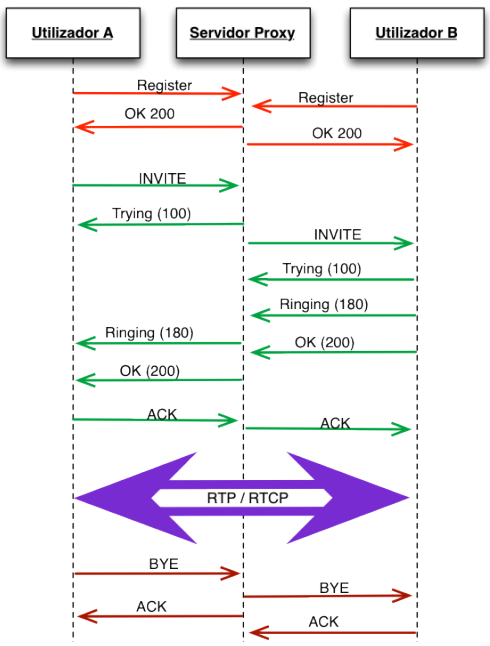


Figura 2.6. Criação de uma chamada SIP [adaptado de [12]].

Resumidamente, os diversos pedidos e as diversas respostas SIP estão sintetizados na Tabela 2.2 e na Tabela 2.3. Este conjunto de mensagens são fundamentais para o estabelecimento de uma ligação e/ou para, por exemplo, relatar problemas, como palavra-passe errada.

Tabela 2.2. Pedidos SIP.

Pedido	Descrição
INVITE	Inicia uma chamada
BYE	Termina uma sessão
ACK	*Acknowledge* (confirmação)
OPTIONS	Questiona o servidor sobre as suas capacidades
CANCEL	Usado para cancelar um dado pedido
REGISTER	Usado para registar informações de localização/autenticação de um utilizador

Tabela 2.3. Respostas SIP.

Código	Descrição	Exemplo
1xx	Informativo	*100 Trying* *180 Ringing*
2xx	Confirmação de sucesso	*200 OK*
3xx	Redireccionamento	*300 Multiple choices*
4xx	Erro de cliente	*400 Bad Request* *403 Forbidden*
5xx	Erro de servidor	*500 Server Internal Error* *501 Not Implemented*
6xx	Erro global	*600 Busy Everywhere* *603 Decline*

Depois de visto quais as mensagens trocadas entre os utilizadores, a Figura 2.7 apresenta como são localizados os utilizados, para posteriormente ser feita uma ligação.

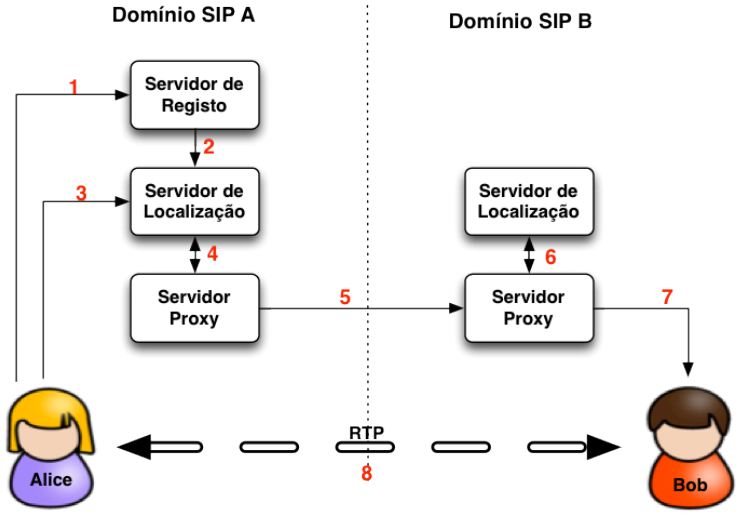


Figura 2.7. Interação entre os vários elementos SIP.

As interações ilustradas na Figura 2.7 ocorrem pela seguinte ordem:

1 - O utilizador Alice, regista-se com o seu respectivo dominio no "Servidor de Registo";

2 – O "Servidor de Localização" guarda as informações de Alice;

3 – Alice decide efetuar uma chamada e para tal liga-se ao "Servidor Proxy";

4 – O "Servidor Proxy" irá, de seguida, consultar o "Servidor de Localização" para encontar o utilizador Bob ;

5 – A chamada é encaminhada para outro "Servidor Proxy";

6 – Esse "Servidor Proxy" irá consultar o respectivo dominio do Bob, no "Servidor de Localização", antes de encaminhar a chamada para o Bob;

7 – O Servidor Proxy contacta o Bob;

8 - Após a negociação do SIP, é feita uma ligação directa ente a Alice e o Bob, para troca de dados;

Por simplicidade, não se mostrou a interação entre a Alice e o "Servidor de Redireccionamento", e entre o "Servidor de Redireccionamento" e o "Servidor de localização". Pois a sua função é, como já foi dito, mapear os endereços SIP dos destinatários da chamada e retornar um endereço para o cliente.

2.1.3.2 H.323

O padrão H.323 foi o primeiro protocolo a aparecer no mercado, pois foi criado em 1996 e faz parte da família de recomendações do ITU, pertencente á série H, que já vai na sétima versão, *H323v7*, com o título "*Audiovisual and Multimedia Systems*" [1]. Este padrão é um conjunto de vários protocolos que determinam procedimentos para a autenticação, contabilização, criação, controlo e encerramento de chamadas [12].

Arquitetura

Ao conjunto dos elementos *Terminal (Tn), Multipoint Control Unit (MCU), Gatekeeper (GK) e Gateway (GW),* que possibilitam a comunicação multimédia, pode ser chamado de *Zona H.323* (Figura 2.8). De referir que em cada "Zona" só poderá existir, e apenas só, um *Gatekeeper* [1].

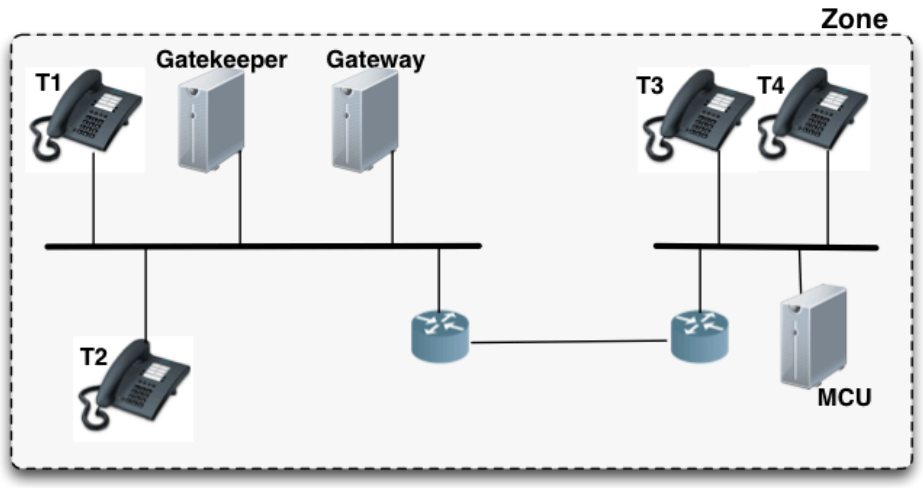


Figura 2.8. Arquitetura H.323 [adaptado de [1]].

O *Terminal* é um elemento básico, ou seja, é o intermediário entre o utilizador e a rede. Pode ser um programa instalado no PC ou apenas um telefone IP, que poderá ou não suportar vídeo.

O *Multipoint Control Unit*, é um elemento importante para conferências com mais de três terminais ou *gateways*, pois será o responsável pela gestão e controlo de voz e vídeo durante as conferências.

O *Gatekeeper* é o elemento mais importante, pois é o que garante a aceitação das chamadas na rede. Controla a realização das mesmas, sendo o responsável pela autenticação dos utilizadores, controlo de QoS e tradução de nomes e números em endereços IP, evitando o conhecimento dos endereços IP dos utilizadores.

Por fim, temos o *Gateway*, que é um elemento opcional em conferências H.323. Ele é o responsável pela interconexão de redes diferentes. Trata-se do local onde se dá a ligação entre a rede H.323 e as redes de telefonia, ou outras redes baseadas noutros protocolos VoIP [12].

Estabelecimento de chamadas

Para a realização de uma chamada em H.323, existem quatro fases distintas: a Inicialização (*Setup*), o Controlo de Sinalização (*Control Signalling*), Áudio e Finalização (*Release*) (ver Figura 2.9) [12].

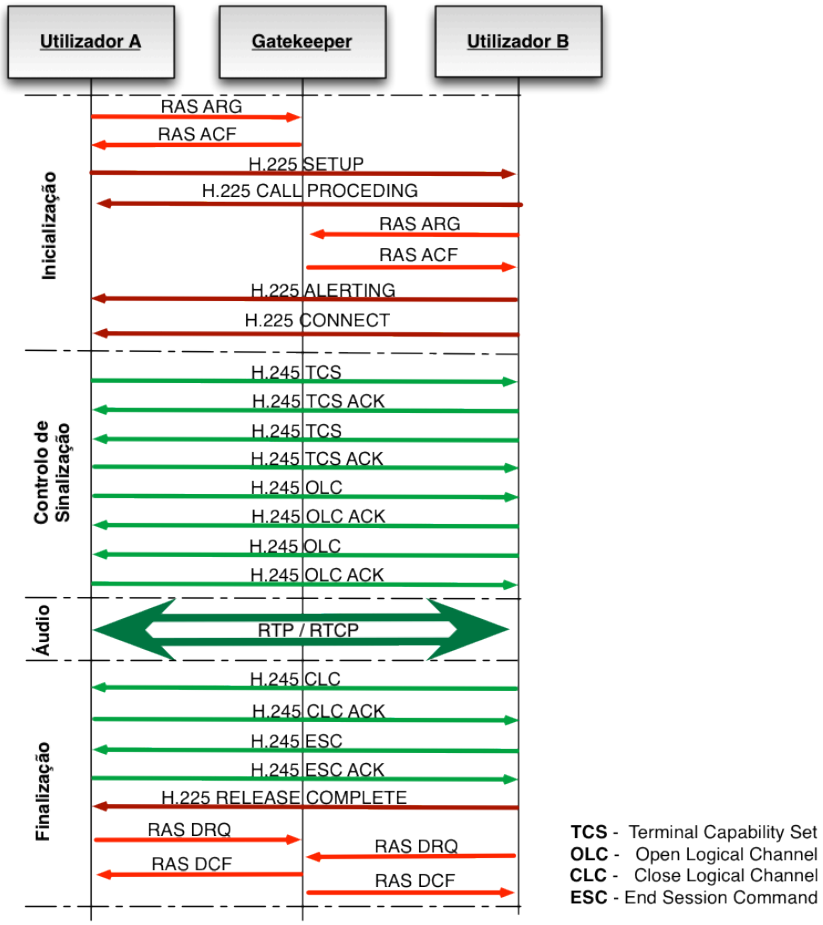


Figura 2.9. Criação de uma chamada H.323 [adaptado de [12].

Na fase de Inicialização, temos a etapa de pedidos de permissão ao Gatekeeper, pelos utilizadores, para efetuar chamadas através das mensagens *RAS ARG* e *REAS ACF*. Nesta fase estão também as mensagens de *H.225 SETUP*, *H.225 CALL PROCEDING*, *H.225 ALERTING* e *H.225 CONNECT*.

No Controlo de Sinalização, são trocadas mensagens que permitem a negociação de parâmetros e abertura de um canal em cada uma das direções, para a transmissão de tráfego de voz e de vídeo.

No Áudio, que só é executado após os canais estarem abertos, passará a existir tráfego entre os vários terminais, encapsulado através do protocolo RTP.

Na última fase, a finalização, temos a finalização da chamada, pois os recursos alocados têm de ser libertados de forma adequada. Nesta etapa temos as seguintes mensagens, a *H.245 CLC (Close Logical Channel)*, a *H.245 CLC ACK (Close Logical Channel ACK)*, a *H.245 ESC (End Session Command)*, a *H.245 ESC ACK (End Session Command ACK)*, a *H.225 RELEASE COMPLETE*, a RAS-DRQ e a RAS-DCF. As ligações a vermelho são fluxos TCP e as verdes são fluxos UDP [12].

2.1.3.3 H.323 versus SIP

De acordo com o que foi apresentado nos pontos acima, sobre o SIP e o H.323, pode-se concluir que o protocolo SIP é mais simples que o protocolo H.323 (ver Tabela 2.4). Isto porque no H.323 existe um elevado número de mensagens trocadas, quer no estabelecimento, quer na terminação de uma chamada, quando comparado com o SIP.

No entanto, o H.323 é mais maduro e confiável, embora não seja muito flexível. Já o SIP apesar de não conseguir prever todas as situações, é mais flexível, sendo compatível com várias aplicações desenvolvidas [12].

A negociação de parâmetros é feita nos dois, assim como o suporte de criptografia e os protocolos de transporte. O endereçamento no H.323 é feito através do número do *host* ou do número de telefone, já no SIP é através de um URL (*Uniform Resource Locator*), como por exemplo, “raquel@voip.uminho.pt”.

Tabela 2.4. Comparação entre H.323 e SIP.

	H.323	SIP
Projetado por	ITU-T	IETF (RFC 3261)
Arquitetura	Pouco flexível	Flexível
Negociação de parâmetros	Sim	Sim
Transporte de Informação	RTP/RTCP	RTP/RTCP
Endereçamento	Número de *host* ou telefone	URL
Segurança com criptografia	Sim	Sim
Implementação	Complexa	Moderada
Tamanho do documento	320 páginas	270 páginas
Estado atual	Pouco usado	Muito usado e em expansão.

2.1.4 SDP (Session Descriptor Protocol)

O Protocolo de Descrição de Sessão (SDP), é um protocolo essencial para o VoIP, sendo usado para descrever sessões multimédia, para efeitos de, por exemplo, anúncios e convites de sessão. Foi desenvolvido pelo IETF e está descrito no RFC 4566 [14]. Usado para anunciar uma sessão e convidar participantes. Não possui um mecanismo de transporte particular, mas está ajustado para utilizar os mecanismos de transporte de outros protocolos [4].

Inicialmente era um componente do SAP (*Session Announcement Protocol*), mas atualmente auxilia outros protocolos, como o RTP, RTSP (*Real Time Transport Streaming Protocol*) e o SIP. Em conjunto com o SIP, o SDP fornece um *backbone* para o controlo de sessão de áudio e vídeo. Os dois são usados em mensagens SIP, para a criação de sessões onde é descrito as capacidades do utilizador de origem. Essa descrição permite que os participantes acordem um conjunto de parâmetros, como o tipo de meio de comunicação, se áudio e/ou vídeo, endereço IP destino, protocolo de transporte (se RTP, UDP, IP, etc), número da porta do destino a ser usada.

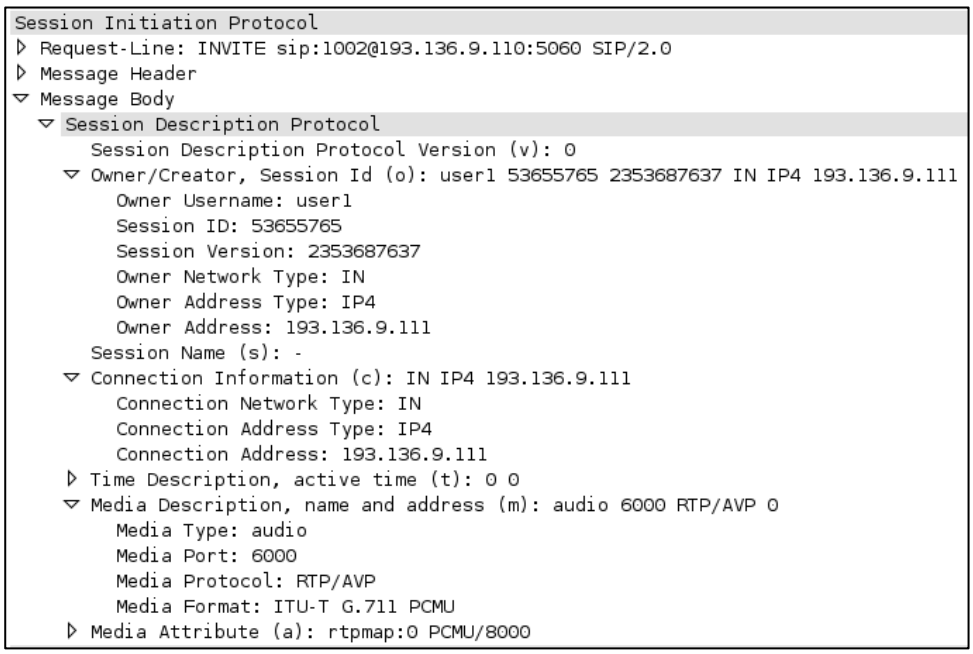


Figura 2.10. Exemplo de informações SDP contidas num pacote SIP.

Na Figura 2.10, é possível observar diversos parâmetros de descrição de sessão, entre os quais podemos retirar o tipo de média (que é áudio), o número da porta (que é 600), o tipo de protocolo (que é RTP/AVP (*Audio Video Profile*)), entre outros.

As mensagens SDP têm todas os seguinte formato *<código>=<valor>.* A Tabela 2.6 apresenta os códigos de carácter obrigatório, e na Tabela 2.5 os de carácter opcional.

Tabela 2.5. Códigos opcionais no SDP [15].

Código	Descrição	Exemplo
c	Informação da conexão	c=<tipo_rede> <tipo_endereço> <endereço_conexão> *c = IN IP4 193.136.9.112*
i	Informação da sessão	*i = <descrição textual>*
k	Chave de encriptação	*k = <método>:<chave _encriptação>* *k = <método>*
a	Atributos de sessão (0 ou mais linhas)	*a = <atributo><valor >* *a=rtpmap:0 PCMU/8000* *a = <atributo>* Estes <atributos> podem ser encontrados na Tabela 2.7

Tabela 2.6. Códigos obrigatórios no SDP [15].

Código	Descrição	Exemplo
v	Versão do protocolo	*v=a*
o	Proprietário e identificação da sessão, segue a seguinte sintaxe: *o=<nome_cliente> <id_sessão> <versão> <tipo_rede> <tipo_endereço> <endereço>*	*o=cliente_1002 56929 1821 IN IP4 192.136.9.112* **IN** – é a sigla para internet **IP4** – protocolo IP versão 4
s	Nome da sessão	*s=cliente_1002*
t	Intervalo de tempo em que a sessão está ativa, segue a seguinte sintaxe: t=<tempo_inicial> <tempo_final>. Sem significado para o SIP, pois não se consegue prever a duração de uma chamada	*t=0 0*
m	Tipo de media, formato e o endereço de transporte, segue a seguinte sintaxe: m=<media> <porta_rtp> <transporte> <lista_formatos>; <media> – pode ser áudio ou vídeo> <listas_formatos> - pode conter os seguintes valores: 0, 3, 8, 18, 101.	*m=audio 5060 RTP/AVP 0 8 101* *0 – G.711 uLaw ; 3 – GSM; 8 – G.711ALaw; 18 – G.729; 101 – outros*

A Tabela 2.7, possui os atributos de sessão usados nos códigos opcionais do SDP. Sendo que em [16] descrevem-se detalhadamente todos os atributos de sessão.

Tabela 2.7. Atributos de sessão [15].

Atributo	Descrição	Exemplo
rtpmap	Trata-se de um mapeamento de códigos RTP	a=rtpmap:<tipo_dados> <nome_codec>/<taxa_transferencia> [/<parametros_codificação>] *a = rtpmap:0 PCMU/8000*
sendrecv	Enviar e receber media	*a = sendrecv*
ptime	Intervalo de tempo (em milissegundos) realizado por um pacote RTP	*a = ptime:<tempo_pacote>*
fmtp	Parâmetros que são específicos de um determinado formato. O SDP não precisa entender os parâmetros, só os transporta	*a = fmtp: <formato><parametros>*

2.1.5 Protocolos de Transporte de Voz

Os protocolos de transporte são o RTP/RTCP, sendo que o RTP (secção 2.1.5.1) é usado para a transmissão de áudio e vídeo, enquanto o RTCP (secção 2.1.5.3) é usado para questões de qualidade de serviço (QoS).

2.1.5.1 RTP (Real-time Transport Protocol)

O Protocolo de Transporte em tempo Real (RTP), desenvolvido pelo IETF, permite transmitir os dados de áudio e vídeo em tempo real. Está descrito no RFC 3550 [17] e funciona sobre UDP, como pode ser visto na Figura 2.3. O uso do protocolo UDP, deve-se ao facto de as mensagens serem transmitidas em tempo real, sem preocupações de pacotes perdidos. Esses dados poderão ser complementados com o uso do protocolo RTCP. Devido à não garantia de QoS por parte do RTP, o RTCP fornece um mecanismo para informar ás aplicações qual é a qualidade das transmissões RTP [18]. O RTP permite uma distribuição *multicast* e *unicast*, ou seja, entrega a informação a múltiplos destinatários ou ponto-a-ponto, respectivamente [4].

A principal tarefa do RTP é a numeração dos pacotes IP, para que no destino se reconstrua a mensagem, de voz ou vídeo, mesmo que haja troca na ordem de receção dos pacotes, forçada pela rede.

A Figura 2.11, apresenta o cabeçalho RTP, a descrição de cada campo pode ser encontrada na Tabela 2.8.

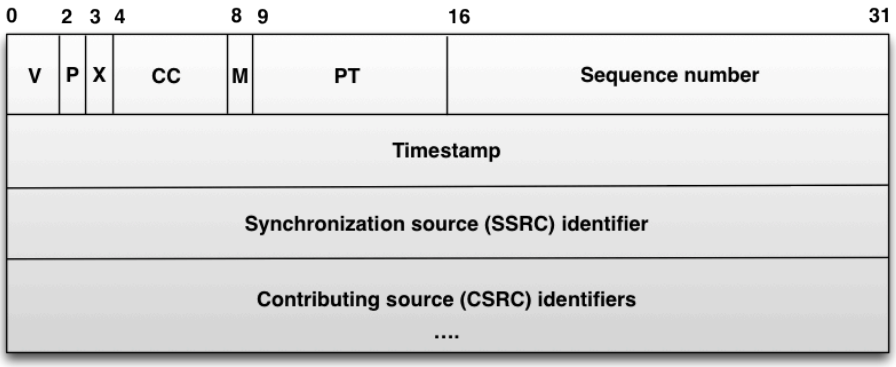


Figura 2.11. Cabeçalho RTP [adaptado de [17]].

Tabela 2.8. Descrição dos diversos campos do cabeçalho RTP.

Campo	Descrição do campo
V (version - 2 bits)	Número da versão (que é a número 2)
P (padding - 1 bit)	Indica se contém preenchimento (0 ou 1)
X (extension - 1 bit)	Indica se há extensão do cabeçalho (0 ou 1)
CC (CSRC Count - 4 bits)	Indica o número de identificadores CSRC
M (Marker - 1 bit)	Especifica o perfil do marcador
PT (Payload Type - 7 bits)	Indica o tipo de RTP, por exemplo, "G.721 Áudio"
Sequence number (16 bits)	Indica o número de sequência do pacote
Timestamp (32 bits)	Instante da amostragem do primeiro octeto de dados
SSRC (32 bits)	Indica a fonte de sincronização, o valor é escolhido de forma aleatória
CSRC (32 bits)	Identifica as fontes que contribuem para a carga contida no pacote.

De acordo com as informações da Figura 2.12, o pacote possui como versão, a número 2. No campo "*Padding*" (P) indica que não há octetos de preenchimento. Caso fosse "True", ou seja, o valor "1", o número de octetos de preenchimento a serem ignorados podem ser encontrados no ultimo octeto de preenchimento, incluindo o próprio. O "*extension"* (X), indica que não existe uma extensão variável no cabeçalho RTP. Já o campo "*CSRC count*" (CC), indica o número de identificadores CSRC (neste caso não existe nenhum). O "*Marker*" (M), destina-se a apresentar o perfil do marcador, permitindo que, por exemplo, sejam definidos os limites das "frames" no fluxo de pacotes, que neste caso não há. O "*Payload Type*" (PT), indica o tipo de dados a serem transmitidos, que neste caso é "ITU-T G.711 PCMU". O "*sequence number*" indica a ordem de sequência dos pacotes, a fim de que o recetor detete perda de pacotes e os ordene, neste exemplo o pacote é o número 9 035. O "*Timestamp*" indica o instante de amostragem, sendo o instante 23520, do primeiro octeto (primeiro byte), no pacote de dados RTP. O campo "SSRC" indica a fonte de sincronização, a 41, sendo que deve ser escolhido aleatoriamente, com o objetivo de não existirem duas fontes iguais dentro da mesma sessão

RTP. O "CSRC", indica as fontes que geram a carga contida em cada pacote e não consta no exemplo pois o campo "*CSRC count*" está a 0. E finalmente surgem os dados, com o valor= "48484948474644...".

```
Real-Time Transport Protocol
▷ [Stream setup by SDP (frame 504)]
  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: 9035
  [Extended sequence number: 74571]
  Timestamp: 23520
  Synchronization Source identifier: 0x00000029 (41)
  Payload: 484849484746444423f3e3e3e3e3e3d3d3d3d3c3c3d3f3e...
```

Figura 2.12. Exemplo de um pacote RTP.

2.1.5.2 SRTP (Security Real-time Transport Protocol)

O Protocolo de Transporte de Segurança em Tempo real (SRTP) é uma extensão do RTP e está descrito no RFC 3711 [19]. O SRTP pretende fornecer serviços de autenticação e integridade, combatendo os ataques de disfarce e roubo de identidade, serviços de confidencialidade dos dados transportados pelo protocolo RTP, e também minimizar os riscos de ataques por negação de serviço (*DoS*).

O uso do protocolo SRTP impossibilita a espionagem, a modificação dos dados e a repetição de pacotes. Minimiza o número de pares de chaves a partilhar, entre os diversos nós, e baseia-se na encriptação, pelo emissor, dos pacotes RTP. Depois estes viajam pela rede e são posteriormente desencriptados no recetor. A Figura 2.13, demonstra como é constituído um pacote SRTP e de que forma este complementa o pacote RTP.

Figura 2.13. Pacote SRTP.

No pacote SRTP (ver Figura 2.13) os dados vão encriptados, e estão localizados no *Payload*. Sendo o pacote composto por mais dois campos, o "*MKI* (*Master Key Identifier*)" e o campo de "*Authentication tag"*. O *MKI* é opcional e de comprimento configurável, onde identifica a chave mestre a partir da qual a chave de sessão foi derivada, que autentica e/ou encripta o pacote. Na caso do campo *Authentication*, também ele de comprimento configurável mas de uso recomendado, é usado para transportar os dados de autenticação das mensagens [19].

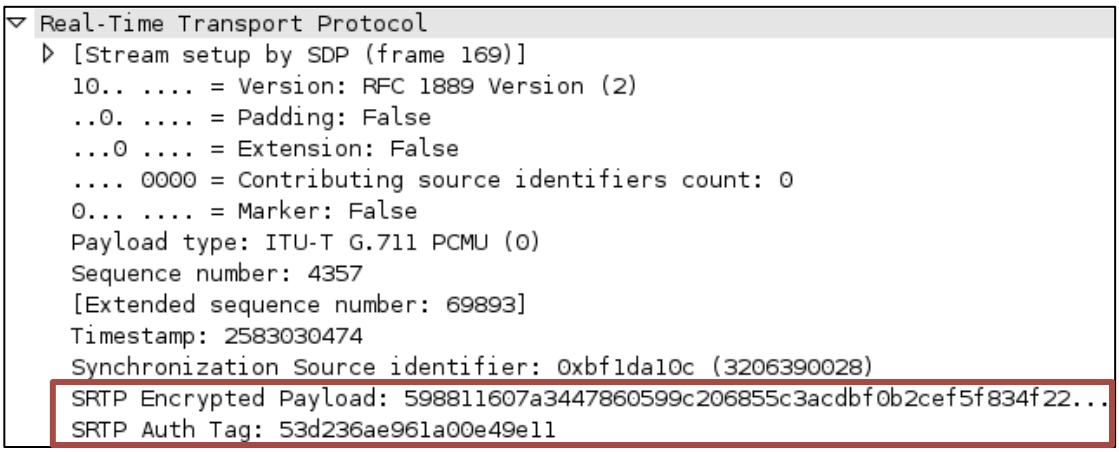


Figura 2.14. Exemplo de um pacote SRTP.

Comparado a Figura 2.14 com a Figura 2.12, deteta-se que no campo de dados do SRTP aparece "*SRTP Encrypted Payload*" e não "*Payload*", indicando que os dados estão encriptados. Deteta-se também a existência de um campo, o "*SRTP Auth tag*", que corresponde ao campo *"Authentication tag"* da Figura 2.14.

2.1.5.3 RTCP (Real-time Transporte Control Protocol)

O Protocolo de Controlo RTP foi desenvolvido pelo IETF e está descrito no mesmo RFC do RTP [17]. Trata-se de um protocolo muito utilizado em videoconferências, que se baseia em transmitir sincronamente pacotes de controlo a todos os intervenientes da sessão, permitindo fazer a gestão da qualidade de serviço, ou seja, monitoriza as sessões RTP, utilizando o mesmo mecanismo de distribuição, os pacotes de dados.

Este protocolo fornece monitorização ponto-a-ponto sobre o estado da qualidade de serviço (QoS), na entrega dos dados. Possui diversos funções: a de dar um *feedback* sobre o desempenho da aplicação e da rede; a correspondência e a sincronização de por exemplo, o áudio e o vídeo combinado; e por fim pode fornecer a identidade do remetente. Para reduzir a sobrecarga na rede, pode-se combinar e encapsular vários pacotes RTCP no mesmo datagrama UDP [18]. A Tabela 2.9, mostra alguns dos pacotes usados pelo RTCP [17].

Tabela 2.9. Pacotes RTCP.

Nome	Acrónimo	Descrição
200	SR "Sender Report"	Relatório do destinatário, para transmissão e receção de dados estatísticos de destinatários ativos, periodicamente
201	RR "Receiver Report"	Relatório do recetor com as estatísticas de receção dos participantes, que não são remetentes ativos, periodicamente.
202	SDES "Source Description"	Indicação de quem é o titular da fonte
203	BYE "Goodbye"	Indica o fim da participação
204	APP "Application-specific functions"	Destinado ao uso experimental de novas aplicações e novas funcionalidades

2.1.6 Protocolos de Controlo de *Gateway*

Dentro dos protocolos de Controlo de *Gateway,* podemos encontrar o protocolo MGCP (*Media Gateway Control Protocol*) e o MEGACO (*Media Gateway Control* ou H.248).

O MGCP foi desenvolvido pela Cisco, apresentado nos RFCs 3661 [20] e 5125 [21], e é utilizado para a criação, controlo e finalização das chamadas. Este protocolo é considerado um modelo H.323 *Gatekeeper* estendido e foi criado a partir de dois protocolos: o IPDC (*Internet Protocol Device Control*) e o SGCP (*Simple Gateway Control Protocol*). MGCP trabalha com o tráfego localizado entre o *Gateway* de media e o controlador [9].

O MEGACO ou H.248 foi desenvolvido pelo Grupo de Estudo 16, do ITU-T, e o grupo de trabalho MEGACO, do IETF. Está definido no RFC 5125 [22] e é similar ao MGCP, no que diz respeito à arquitetura e propósitos, mas apresenta algumas melhorias. Melhorias como o suporte de serviços multimédia e de videoconferência, permitindo o uso do protocolo UDP ou do protocolo TCP e por fim permite uma codificação em modo de texto ou binário. Sendo que o objetivo do seu desenvolvimento foi o de ser uma alternativa ao MGCP e outros protocolos similares.

2.1.7 Segurança

No que diz respeito à segurança no VoIP e segundo [8], existem diversos ataques de segurança, nomeadamente o Negação de Serviço (*Denial of Service (DoS)*), Falsificação e Disfarce dos Pacotes (*Packet Spoofing and Masquerading*), Espionagem (*Eavesdropping*), Mensagens Indesejáveis e Armadilha VoIP (*VoIP Spam and Phishing*) e Fraude de Tarifação (*Toll Fraud*).

O "*Denial of Service (DoS)*" é um tipo de ataque que torna os recursos de um dado sistema indisponíveis, pois o excesso de pedidos provocam uma diminuição da largura disponível e a exaustão dos recursos do sistema. O "*Packet Spoofing and Masquerading*" baseia-se em enganar (spoof) através dos pacotes IP, utilizando falsos endereços dos destinatários nos pacotes enviados, fazendo-se passar por outros, ameaçando os sistemas baseados em autenticação pelo endereço IP. O "*Eavesdropping*" é um tipo de ataque em que o atacante, se conecta com as vítimas de forma independente e retransmite as mensagens entre elas, sem que se apercebem que a conversa está a ser ouvida e controlada por terceiros, acreditando que a conversa é privada. Este tipo de ataque também pode ser chamado de Man-in-the-middle. O "*VoIP Spam and Phishing*" é causado por pessoas mal-intencionadas, que têm como objetivo comunicações não solicitadas e acima de tudo não desejadas. Este tipo de pessoas fazem-se passar por outras, como elementos de uma emprese em quem a vítima confia, e tentam adquirir dados pessoais, como *passwords*, dados financeiros, entre outros. O "Toll Fraud" é uma fraude de longa distância, ou seja, é o uso não autorizado de serviços pagos, como chamadas internacionais, sendo estas cobradas a uma entidade confiável.

Dito isto, e como o objetivo da dissertação é usar o protocolo RTP e o protocolo SIP, existem mais uma série de ataques (ver Tabela 2.10) associados especificamente a estes protocolos. Na tabela que se segue, é possível visualizar os ataques associados aos protocolos SIP e RTP, assim como as propriedades de segurança que são afectadas [23].

Tabela 2.10. Ataques ao SIP e ao RTP [23].

	Ataque	Confidencialidade	Integridade	Disponibilidade
SIP	Desvio de Registo *(Registration Hijacking)*	X	X	X
	Modificação de Mensagem *(Message Modification)*	X	X	
	Ataque de Cancelamento *(Cancel/Bye Attack)*			X
	Comando incorreto *(Malformed Command)*			X
	Redireccionamento *(Redirect)*	X		X
RTP	Dados RTP *(RTP Payload)*			X
	Adulteração RTP *(RTP Tampering)*	X	X	X

O *Registration Hijacking* ocorre quando um atacante se faz passar por um *UA* válido para o "Servidor de Registo", sendo que substitui o endereço do UA pelo seu próprio endereço. Depois disso todas as chamadas recebidas são enviadas para o UA registado pelo atacante.

O *Message Modification* baseia-se em modificar campos de uma mensagem, por exemplo, se o atacante conseguir alterar o campo "*Content-Length*" de 450bytes para 200bytes, o recetor da mensagem irá perder dados [24].

O *Cancel/Bye Attack* ocorre quando o atacante insere as mensagens de *Cancel* e *Bye* numa sessão, e estas provocam o cancelamento de um pedido em curso e o encerramento da sessão, respectivamente. Com o uso destas mensagens impede-se que um dado utilizador inicie a sua conversão ou que perca a conexão a meio da mesma.

O *Malformed Command* ocorre porque como o SIP baseia-se na linguagem HTML (*HyperText Markup Language*), o que faz dele um protocolo muito flexível e extensível para as funcionalidades VoIP, tornando-se difícil de testar e analisar cada entrada possível. Os atacantes aproveitam-se destas vulnerabilidades e geram pacotes mal formados e enviam-nos aos pontos mais sensíveis. O *Redirect* ocorre devido à fraca autenticação no SIP, como em outros ataques já mencionados. Este ataque baseia-se em atacar o Servidor de Redireccionamento, obrigando-o a redirecionar todas as chamadas de uma dada vítima para um endereço especificado pelo atacante. O *RTP Payload* baseia-se num atacante ter acesso ao fluxo RTP, que está a ser trocado entre dois pontos, permitindo-lhe inspecionar e modificar esse mesmo fluxo. Esta modificação pode causar ruído na conversa, podendo danificá-la ou mesmo torná-la impossível de interpretar. Isso é facilmente perceptível pelos intervenientes, sendo que a consequência imediata é a quebra da comunicação. O *RTP Tampering* ocorre devido à alteração dos campos "*Sequence number"* e "*Timestamp"* do pacote RTP, provocando uma reordenação ou mesmo inutilidade dos pacotes, podendo tornar a conversa ilegível [23].

2.1.8 QoS (Quality of Service)

Para além do baixo custo, os clientes esperam um bom QoS durante as chamadas VoIP. Isto porque os clientes habituaram-se à qualidade da conversação fornecida pela rede telefónica tradicional, como a PSTN. Porém as redes IP, não foram desenhadas para aplicações em tempo real, como é o caso do VoIP, pois não existem formas de monitorização central ou medição de desempenho que permita controlar ou manter o estado da rede [7].

Na verdade, o QoS não é garantido nos sistemas VoIP, sendo que estes sistemas, tal como outros aplicativos em tempo real, são sensíveis à largura de banda e ao atraso, estando muito propícios ao congestionamento e aos ataques maliciosos, como ataques DoS. No caso do congestionamento, os pacotes podem ser descartados ou até mesmo introduzir latências elevadas, degradando a qualidade da chamada. Dado que a disponibilidade das ligações dos diversos nós é dinâmica, o atraso (*delay*) irá variar ao longo do tempo. A variação do atraso é problemática para os serviços VoIP e é chamada de *jitter* [7].

De acordo com [25], o QoS em redes de dados por pacotes depende de parâmetros, como a largura de banda disponível, do *delay* (tempo decorrido desde a origem até ao recetor), do *jitter* (variação do *delay*) e do *packet loss* (perda de pacotes). As diferentes aplicações têm também diferentes requisitos e QoS. As aplicações de voz requerem pouca largura de banda, mas não toleram *delays* ou *jitters* elevados. As aplicações de imagem requerem mais largura de banda, mas não suportam altos *delays*, embora permitam algumas perdas. Uma das formas adotadas para o controlo de QoS, foi o desenvolvimento de protocolos para tal, como é o caso do RTCP. Este assunto, é uma área aberta e desafiadora.

Para que seja garantido o QoS, há três parâmetros que não devem ultrapassar os valores apresentados na Tabela 2.11, no caso da voz [25].

Tabela 2.11. Valores limite para a garantia de QoS [25].

Delay	Jitter	Packet Loss
$\leq$ 150 ms	$\leq$ 25 ms	$\leq 10^{-4}$ ou 0,1%

Na caso do *delay*, ele é aceitável por grande parte das aplicações se for $\leq$ 150 milissegundos (ms), mas de acordo com a fonte [26], ele pode ir até aos 400 ms e é aceitável pelo utilizador, estando este consciente do seu impacto. Valores de *delay* acima dos 400 ms já são inaceitáveis.

2.1.8.1 Modelos QoS

O modelo tradicional, designado por *Best-effort Service*, ou seja, serviço de melhor esforço, não era suficiente para satisfazer as necessidade das aplicações, portanto pensou-se em adicionar melhorias, como a inclusão de níveis de QoS diferenciados, capacidade de gestão e atribuição de recursos. Foi então que se criou o modelo de serviços Integrados, o *IntServ* (*Integrated Services*), e o modelo de Serviços diferenciados, o *DiffServ* (*Differentiated Services*).

O *IntServ* faz a reserva de recursos fim-a-fim, garantindo assim que os pacotes têm todos os recursos alocados, permitindo assim fazer uma videochamada sem problemas de falha. Este modelo não é escalável, pois é pouco provável que a rede possua os recursos necessários para as mais diversas aplicações.

O *DiffServ* não partilha da mesma ideia que o *IntServ*, pois pretende resolver o problema de escalonamento com grandes quantidades de dados. Este modelo subentende que as entidades envolventes, combinem sobre os recursos disponíveis. Estas combinações/negociações são chamadas de *SLAs* (*Service Level Agreements*), que traduzido significa acordos de nível de serviço.

Resumidamente, no *IntServ* os pacotes só são transmitidos se a rede possuir os recursos necessários, já no *DiffServ* os pacotes são tratados nó a nó, de acordo com a classe a que pertencem.

No que diz respeito aos *CODECs*, e tendo em conta a Tabela 2.1, existem dois parâmetros a considerar em cada um deles, que são o *delay* introduzido pelo algoritmo e o introduzido pelo *CODEC*, conforme se pode ver na Tabela 2.12.

Tabela 2.12. Parâmetros de QoS dos Codecs.[26]

CODEC	*Delay* algoritmo (ms)	*Delay* do CODEC (ms)
G.711	0,125	0,25
G.729	15	25
G.723..1	30	67,5
AMR-NB	25	45
GSM-EFR	20	40
iBLC (LPC)	40	60
iBLC (Block Independent)	25	40

2.2 WebRTC (*Web Real-Time Communications*)

WebRTC, também conhecido de *Web Real-Time Communications*, foi apresentada em 2012 pela Google. É uma tecnologia *open-source* e grátis, que está a ser desenvolvida com o apoio do W3C (*World Wide Web Consortium*) e do IETF. Esta tecnologia é suportada pelos *browsers Chrome*, *Firefox* e *Opera*, e pelas aplicações móveis *Android* e *iOS* [27].

Esta tecnologia soluciona o problema de incompatibilidade nas comunicações em tempo real utilizando apenas o *browser Web*, pois, atualmente, para realizar uma chamada de áudio ou vídeo, a partir de um computador, é necessária instalar *software's* ou *plugins*, e até criar contas. Por exemplo, o *Google Hangouts* usa o *plugin* do *Google Talk*. A instalação e atualização de *plugins* pode ser complexo e sujeito a erros, já para não falar de que muitas vezes, é complicado convencer os utilizadores a instalar *plugins*. O *WebRTC*, aproveita a tendência, de que o *browser* é a "aplicação", facilitando a comunicação entre os utilizadores e sem a necessidade de se instalar *software's* ou criar contas.

Segundo [28], a visão do *WebRTC* é a de, no mundo, telefones, televisões e computadores comuniquem todos numa plataforma comum. Possibilita um acesso à webcam e ao microfone, permitindo receber e enviar dados P2P (Peer-to-Peer) em *real-time* entre dois browsers sem *plugins*, ver Figura 2.15 [29].

Figura 2.15. Diagrama de funcionamento do *WebRTC*.

WebRTC não possui etapas complicadas ou instalações, pois usa recursos simples, ou seja, não requer *plugins*, *frameworks* ou instalação de aplicações, apenas faz uso do navegador que é usado diariamente. Uma das principais vantagens é a interoperabilidade da voz e do vídeo entre

aplicações *WebRTC*. É uma tecnologia que está a ganhar rapidamente terreno e irá revolucionar os padrões de comunicação [30]. A Figura 2.16, apresenta a arquitetura da tecnologia *WebRTC*.

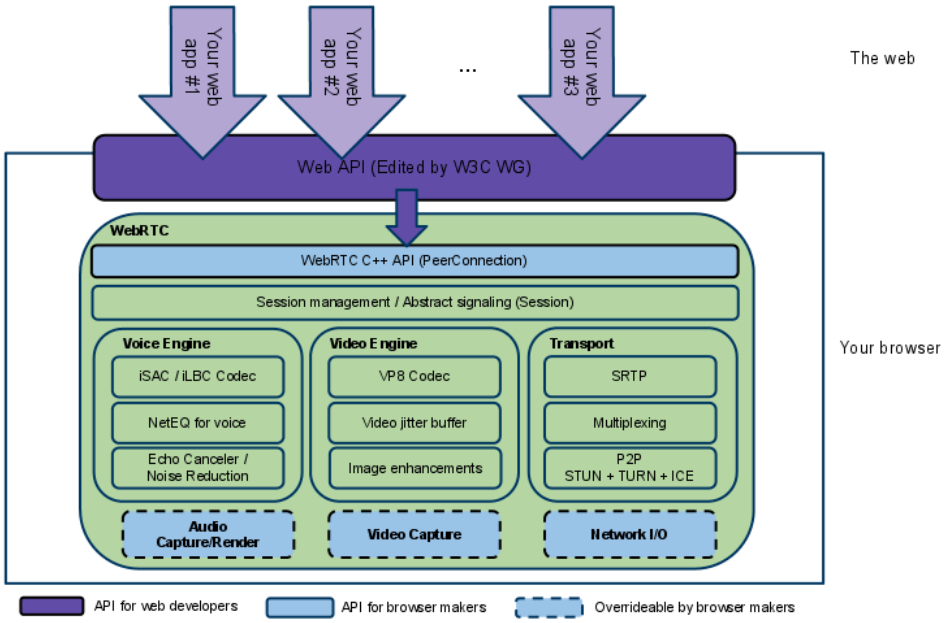


Figura 2.16. Arquitetura do *WebRTC* [27].

A Figura 2.16, mostra os componentes constituintes do *software WebRTC* de um *browser* ("*Your browser*"). O componente "*Your Web APP*", representa uma aplicação para terceiros, desenvolvido em Web (HTML5/*javascript*), baseada numa aplicação com capacidades de vídeo e áudio, suportados por uma "Web API" para comunicações em tempo real.

Esta tecnologia possui duas *APIs*, a "*Web API"* (usada por qualquer pessoa), que é uma API que pode ser usada para o desenvolvimento de aplicações de vídeo e *chat*, baseadas na Web. E a "*WebRTC C++ API"* (usada por programadores) refere-se a uma camada que permite aos fabricantes dos browsers, desenvolver *Web Browsers*.

No que diz respeito ao "*Transport/ Session"*, os componentes de sessão são construídos através da reutilização de componentes *libjingle*, sem usar ou exigir o protocolo XMPP/jingle. *Libjingle* é um conjunto de componentes fornecidos pela Google, para interagir com recursos *Google Talk peer-to-peer* e de voz. No "*Transport*", temos a RTP *Stack* e STUN/ICE. Sendo que o RTP Stack é uma pilha de rede para o RTP, ou seja, o Protocolo em tempo real. O STUN/ICE é um componente que permite chamadas que utilizem STUN (*Session Traversal Utilities for NAT*) e mecanismos ICE (*Interactive Connectivity Establishment*) para estabelecer conexões através de

firewalls. Por fim, no que diz respeito ao "*Session Management"*, este permite a configuração da chamada e da camada de gestão [27].

O *Voice Engine* é uma estrutura para serialização áudio, que vão da placa de som para a rede e vise versa. Por fim o *Video Engine* é também uma estrutura para serialização vídeo, que vão da camera para a rede e da rede para o ecrã. Faz uso do *codec* "*VP8*", devido à sua baixa latência. O "*Video Jitter Buffer"* camufla os efeitos do *jitter* e da perda de pacotes na qualidade de vídeo em geral. A *"Image Enhancements"* remove, por exemplo, o ruído do vídeo durante a captura de imagem, pela webcam [27].

2.3 Trabalhos relacionados

Nas secções que se seguem, serão apresentados trabalhos de investigação atuais, que de alguma forma se assemelham ao trabalho em curso.

2.3.1 Identificação de vulnerabilidades

O artigo [6], intitulado por "*VoIP Technology: Security Issues Analysis*", publicado em 2013, tem como finalidade a análise dos problemas de segurança associados ao VoIP. O seu objetivo é o de realizar uma análise profunda das preocupações de segurança associados à tecnologia VoIP.

O autor começa por enumerar as vantagens, como custos reduzidos, prestação de novos serviços de comunicação, portabilidade e integração com novas aplicações. Como obstáculos o autor definiu os seguintes, arquitetura complexa, problemas de interoperabilidade, questões de QoS e preocupações associadas à área da segurança. Realça ainda, que dentro desses problemas todos, a segurança é a mais grave, isto porque os dispositivos de segurança tradicional, protocolos e arquiteturas não conseguem proteger adequadamente os sistemas VoIP dos ataques inteligentes mais recentes.

Como introdução, é apresentada uma breve visão da tecnologia VoIP. Onde é dito que a tecnologia VoIP para além das vulnerabilidades associadas aos dispositivos VoIP, também herda as vulnerabilidades associadas à infraestrutura. Os ataques de segurança são classificados em quatro tipos, ataques contra a disponibilidade (colocando as ligações ou os recursos em baixo),

confidencialidade (não afecta a comunicação em si, mas afecta o sigilo), integridade (garante que o que chegou ao recetor não é o mesmo que saiu do emissor) e Engenharia social (manipular a comunicação, em que o invasor falsifica a comunicação, mostrando-se ser uma entidade confiável).

A arquitetura VoIP também é apresentada, é dito que esta, pode ser implementada usando a arquitetura distribuída ou centralizada, apesar de que a maioria dos sistemas VoIP são implementados segundo a arquitetura cliente/servidor centralizado. Depois da arquitetura, são descritos os diversos protocolos existente nesta tecnologia, como os protocolos de transporte de media (RTP e SRTP) e protocolos de sinalização (H.323, SIP e IAX). Por fim, são apresentadas as desvantagens VoIP, como o grau de dificuldade dos serviços e uma arquitetura pouco simplificada (pois integra de diferentes serviços como a voz, o vídeo e os dados, num só), problemas de interoperabilidade entre diferentes aplicações ou produtos e problemas de QoS.

De seguida o autor define as vulnerabilidades (falha ou fraqueza a ser explorada por um atacante), onde apresenta a vulnerabilidade herdada pela rede, pelo Sistema Operativo, pelo Servidores Web, etc. E as vulnerabilidades herdadas pelos protocolos e pelos dispositivos VoIP, tais como telefone IP, serviços media, etc.

São apresentados, de forma sistemática, os ataques de segurança VoIP contra a disponibilidade (como o ataque *DoS*, *Call flooding*, *Malformed Messages*, *Spoofed Messages*, *Call Hijacking* e QoS *Abuse*), contra a confidencialidade (como ataques *Media eavesdropping*, *Call Pattern Tracking* e *Data Mining*), contra a integridade (como ataques *Call Rerouting*, *Media injects* e *Media degrading*) e contra o contexto social (como *Spam*, *Phishing* e *Misrepresentation*). Ataques *Call flooding* refere-se à inundação da rede com tráfego válido e inválido, que podem ser de sinalização ou de media, colocando o sistemas em baixo. O *Malformed Messages* (é uma mensagem com sintaxe errada), podem ser enviadas para um servidor de destino ou para um cliente, que tem por finalidade a interrupção do serviço, podendo provocar *loop* infinito, *buffer overflow*, etc. O ataque de *Spoofed Messages*, baseia-se num atacante inserir mensagens falsas, numa sessão VoIP, podendo interromper o serviço ou roubar a sessão. *Call Hijacking* ocorre quando o atacante consegue apoderar-se das operações associadas entre uma extremidade VoIP e a rede, podendo interromper os serviços, desativar usuários legítimos, etc. QoS *Abuse* consiste na modificação de CODECs, do tipo de media, da

taxa de bits, etc, durante a negociação de parâmetros de uma sessão. *Media eavesdropping* é o acesso não autorizado aos pacotes media. *Call Pattern Tracking* é a análise não autorizada de tráfego VoIP de ou para qualquer nó ou rede. *Data Mining* é um ataque que recolhe dados, de forma não autorizada, como nome do usuário, número de telefone, *password*, endereço de *e-mail*, etc. *Call Rerouting* consiste na alteração não autorizada da direção da chamada, alterando as informações de encaminhamento. *Media injects* reside no atacante injetar media num canal media ativo, podendo causar ruído, silêncio ou até publicidade no meio de uma conversa. *Media degrading* baseia-se na redução da qualidade da comunicação (QoS), por exemplo, o atacante interceta os pacotes RTCP e altera-lhes o número de sequência. O *Spam* é definido como um conjunto de e-mails enviados em massa para um cliente, por forma a iniciar uma sessão e tentar estabelecer uma sessão de voz ou de vídeo. O *Phishing* é uma tentativa ilegal de obter informações pessoais, como *password*, número de conta bancária, número de cartão de crédito, etc, em que o atacante se faz passar por uma entidade confiável. Por fim, o *Misrepresentation* é um método de apresentações falsas, como nome falso, organização, endereço de e-mail, etc, enganando assim os utilizadores.

Uma vez enumeradas os ataques são apresentadas as capacidades de segurança associadas aos protocolos H.323, SIP, IAX e RTP/RTCP, onde o autor faz uma descrição de cada um dos protocolos.

Por fim, detalha-se a segurança associada aos dispositivos VoIP, como o *VoIP-aware firewall*, o NAT (*Network Address Translation*) e o *Session Border Controller* (SBC). O *VoIP-aware* protege a rede interna de ataques externos. Bloqueia alguns tipos de tráfego com base no endereço IP de origem e destino, no protocolo de transporte (TCP ou UDP), no número de porta de origem e/ou destino, etc. O NAT para além da tradução de endereços de redes internas para externas e *vice-versa*, também fornece proteção automática como se fosse uma *firewall*, sem qualquer configuração especial. O SBC é um dispositivo situado na fronteira entre duas sessões de rede. Uma sessão de rede pode ser uma rede de acesso, um núcleo de rede, etc.

O autor termina com um breve resumo e indica que num trabalho futuro irá abordar as capacidades, da tecnologia VoIP, de suportar restrições de QoS em aplicações de voz e vídeo.

O artigo [9] é intitulado por "*Security threats of VoIP*", publicado em 2014, discute as ameaças de segurança VoIP. É referido que as medidas de segurança no VoIP estão num estado inicial e de que atualmente existe muita investigação na área. Os autores mencionam como a velocidade da internet vai aumentando, no futuro novas ameaças surgirão. Atualmente, os atacantes/hackers ameaçam não só ao nível da rede, mas também ao nível do protocolo.

Os autores começam por apresentar o objetivo principal do artigo, que é o de discutir as ameaças na segurança do VoIP e propor métodos de segurança, de forma a prevenir essas ameaças. O artigo realça o protocolo SIP, discutindo as ameaças de segurança a ele associadas. São explicados os componentes VoIP, onde destaca os três componentes essenciais, em que o VoIP consiste, sendo o CODEC, *packetizer* e o *play out buffer*. É feita a observação de que o protocolo TCP é o mais usado na internet, mas que o VoIP prefere o UDP, pois neste não faz uso do esquema *acknowledgement* (ACK), como é o caso do TCP. Este esquema irá introduzir atrasos devido ao recetor ter de notificar o emissor, a receção de cada pacote.

Depois de uma introdução os autores apresentam as normas e os protocolos VoIP, sendo que nos protocolos os autores apenas se preocuparam com os mais comuns, como é o caso do RTP/RTCP, H.323, SIP, MGCP.

De seguida, os autores discutem as questões relativas às ameaças de segurança no VoIP. Onde é destacado o ataque *Malformed Message Threat*, *SIP Flooding Threat*, *Spoofing Threat*, *DoS*, *Call Hijacking and Intercetion*, *H.323-specific attacks* e *SIP-specific attacks*. O ataque *Malformed Message Threat* é realçado como um dos ataques mais representativos, que utiliza as vulnerabilidades dos protocolos baseados em texto. Sendo que o atacante ao intercetar as mensagens transferidas, entre o servidor e o cliente, pode obter a chave pública e posteriormente, obter mensagens enviados pelo cliente e decifra-las com a chave. Após a decifragem, o atacante pode modificar a mensagem e encaminha-la para o servidor.

Após uma discussão detalhada sobre a segurança SIP, são feitas algumas propostas, sendo elas o uso dos protocolos *HTTP Digest Authentication*, *IPSec*, *TLS* e *S/MIME* (*Secure/Multipurpose Internet Mail Extensions*). Como medidas os autores referem as seguintes: uso do protocolo SRTP, uso de assinatura digitais, uso de dispositivos seguros, uso de autenticação forte, habilitar a autenticação DIP, uso de VLAN com o padrão *802.1x* e desativar o uso de *telnet* na configuração do telefone. Os ataques SIP mencionados foram os seguintes:

Registration Hijacking, *Proxy Impersonation* (ocorre durante a comunicação com um *Proxy* desonesto), *Message Tempering*, *Session Tear Down* e *DoS*.

2.3.2 Desafios da investigação na segurança VoIP

O artigo [13] de 2012, intitulado por "*A Comprehensive Survey of Voice over IP Security Research*" faz um levantamento exaustivo da tecnologia VoIP, utilizando um conjunto de 245 publicações, classificadas de acordo com uma versão estendida da taxonomia de ameaça, definida pela organização VoIPSA (*VoIP Security Alliance).* O objetivo é o de fornecer um "mapa" aos investigadores que procuram perceber as capacidades e identificar as falhas do VoIP, relativamente às inúmeras ameaças e vulnerabilidades que o mesmo apresenta. É feita uma discussão sobre as implicações dos resultados apresentados, em relação ás vulnerabilidades apresentadas em uma variedades de produtos VoIP. O autor identifica duas áreas problemáticas, a de negação de serviço (DoS) e a de abuso de serviços.

O autor inicia com uma introdução à tecnologia VoIP, apresentando uma visão geral do protocolo SIP, onde o autor afirma ser o protocolo mais popular atualmente em uso. São apresentadas as interações SIP, as trocas de mensagens durante a configuração de uma chamada SIP e as mensagens HTTP *Digest Authentication* entre um utilizador e o *proxy*, que se trata de um protocolo de desafio-resposta simples.

Seguidamente faz a classificação das ameaças identificadas nos trabalhos pesquisados, usando a taxonomia fornecida pelo VoIPSA, onde descreve a VoIPSA como sendo um fornecedor neutro, uma organização sem fins lucrativos. A taxonomia de ameaça VoIPSA, tem como objetivo a definição das ameaças de segurança contra as implementações VoIP, serviços e utilizados finais. Os elementos chave da taxonomia são: *Social Threats*, *Eavesdropping*, *DoS*, *Service Abuse*, *Physical Access* e *Interruption of services*. O autor faz a classificação dos trabalhos tendo em conta só as quatro primeiras ameaças, pois ele considera que as duas últimas, em grande parte ocorrem fora do âmbito da segurança do computador. Sendo que mais tarde, apresenta mais elementos, para classificação dos trabalhos, sendo a classificação, o seguinte, trabalhos que resumam ataques e defesas em VoIP (*Overviews and Surveys)*, trabalhos que analisam software ou protocolos (*Field Studies and System/Protocol Analysis)*, trabalhos que meçam o impacto dos mecanismos de segurança (*Performance Analysis*), trabalhos que falem de

mecanismos de autenticação e algoritmos SIP (*Authentication Protocols*), trabalhos que definam abordagens para garantia de QoS (*Architecture*), trabalhos que descrevam novas arquiteturas ou mecanismos de *firewall* (*Middleboxes*), trabalhos que detetem anomalias e que não sejam facilmente classificados em nenhuma das categorias anteriores (*Intrusion Detetion*), ou trabalhos que não se enquadrem e nenhum dos anteriores (*Miscellaneous*). O autor estrutura todas estas classificações sob a forma de árvore, onde é apresentado quantos trabalhos se encontram classificados em cada categoria.

Como principal contributo deste trabalho, é apresentada a pesquisa efetuada sobre a segurança VoIP. Onde é apresentado o processo utilizado para a recolha dos diversos trabalhos, como por exemplo, pesquisas feitas em CiteSeer e IEEE Xplore, etc. Posteriormente é feita a classificação dos trabalhos, segundo a taxonomia da VoIPSA (*Social Threats*, *Eavesdropping*, *DoS*, *Service Abuse*), que foram 111 trabalhos, e as categorias adicionais (como *Overviews and Surveys*, *Field Studies and System/ Protocol Analysis*, etc), foram 134 trabalhos.

Na discussão dos resultados, onde o autor conclui que diversos tipos de ataques DoS são provenientes de vulnerabilidades descobertas, ou seja, 90% devido a problemas de implementação e 7% devido à configuração. Conclui também que a pesquisa sobre a criptografia parece ser razoavelmente confortável, propõe ajustes e pequenas melhorias aos mecanismos de autenticação básica sendo que a comunidade de sistemas parece, no entender do autor, contentar-se com a análise de desempenho de diferentes configurações de protocolos, como o TLS vs IPSec,

O autor termina com uma breve conclusão, onde realça que espera ter facilitando a tarefa na realização de pesquisa sobre a segurança VoIP.

2.3.3 QoS e conectividade global

O artigo [26], intitulado por "*VoIP: State of art for global connectivity – A critical review*" de 2014, analisa os problemas da qualidade do sinal num sistema VoIP.

Este artigo, começa com uma breve descrição da evolução da telefonia tradicional. Depois, apresenta os sucessos do VoIP em relação ao PSTN, como o baixo custo, serviços integrados,

updates escaláveis e fáceis, recuperação de desastres, fax sobre IP e recursos avançados e por fim a segurança.

Os autores enumeram ainda os factores que afectam a qualidade do sinal, sendo eles o *delay* (podendo ser do tipo *Propagation delay*, que é o tempo gasto de um terminal ao outro; *Compression delay*, atraso inserido pelos *CODECs*; *Packetization delay*, é tanto maior quanto maior for o pacote; e por fim o *Packet-switching delay*, é o tempo que um *router* ou *switch* demora a decidir por que interface o pacote irá sair), o *Jitter* e o *Packet Loss*.

De seguida é apresentado quais os problemas existentes na compressão de voz, onde realça os *CODECs* de banda estreita (sinal entre 200Hz e 3,4KHz) e de banda larga (sinal entre 50Hz e 7KHz). É apresentada uma tabela bastante interessante, onde apresenta os *CODECs* de baixa e alta largura de banda, com parâmetros como, o *delay* provocado pelo algoritmo de compressão e pelo *CODEC*, entre outros parâmetros. Neste ponto são apresentados os pontos positivos e negativos de cada um dos *CODECs*.

Posteriormente, discute-se as técnicas para medição da qualidade de voz, quer através da opinião dos seres humanos (através do MOS, *Mean Opinion Score*, que avalia a qualidade da chamada de acordo com a opinião do utilizador), quer através do PESQ (*Perceptual Evaluation of Speech Quality*, que produz estimativas de uma vasta gama de ruídos de fundo, como de carros, barulhos de rua, etc).

Reportam-se também os *software's* VoIP, conhecidos como *softphones*. Nesta seção é apresentada uma tabela muito interessante sobre os *softphones* (como Skype, GoogleTalk, X-lite, etc), onde consta que tipo de encriptação possuem, qual o protocolo de suporte, entre outros.

Finalmente é feita uma análise técnica do VoIP das últimas quatro décadas. Destaca-se, por exemplo, na década atual que "*Karapantazis and Pavlidou*" estudaram a possibilidade de efetuar ligações VoIP via satélite. Na conclusão destaca-se que o sistema VoIP foi considerado a melhor alternativa ao sistemas PSTN, no que diz respeito aos serviços de voz para os utilizadores.

2.3.4 Avaliação de desempenho

A fonte [31], intitulada por "*Secure VoIP Performance Measurement*" refere-se a uma tese de doutoramento, da universidade *Loughborough* de 2013, é talvez o trabalho que mais se assemelha ao que é aqui estudado. O trabalho tem por base a realização de experiências em diferentes condições de rede e de segurança, com diferentes *softphones*, como o *Google Talk*, *Express Talk* e o *Skype*. É feita a análise da qualidade da voz, através dos valores *MOS*, utilizando as ferramentas *PESQ* e *E-model*.

A autora começa por desvendar os aspectos negativos do VoIP em relação à rede *PSTN*. É dito que a qualidade de voz no VoIP ainda não é tão boa como a do *PSTN*, a fiabilidade do VoIP é inferior aos 99,99% da rede de telefonia tradicional, o VoIP está exposto a vulnerabilidades de segurança da internet, este consume largura de banda (reduzindo assim a largura de banda disponível para outras aplicações) e usa recursos, como *CPU*, memória e *buffers* dos utilizadores.

A escritora faz avaliações do desempenho dos serviços VoIP com diferentes configurações de segurança. Sendo utilizado como comparação os parâmetros a serem medidos, como o *delay*, o *jitter*, o *packet loss*, a largura de banda disponível, o CPU, a memoria do *host* e o tamanho do *buffer*.

É efetuada a monotorização do desempenho da rede VoIP, do ponto de vista de um gestor de rede. É feita a implementação de uma plataforma de testes em ambiente real, para controlar as experiências realizadas em vários serviços VoIP. A autora também faz uso da ferramenta de monitorização (como o CACTI), do intercetor de pacotes (o *dummynet*) e do simulador de clientes (com auxilio dos *softphones* já falados). São criadas scripts para automatizar a recolha de dados, e por fim é feita a recolha e a validação dos dados. Os dados recolhidos são apresentados sob a forma de gráfico e sempre com o objetivo de comparação, por exemplo a avaliação MOS vs PESQ-WB com três linhas de comparação. As linhas referem-se aos *softphones Skype*, *Google Talk* e *SIP-based*. Posteriormente, fixa o *softphone* e altera o protocolo de transporte, sendo o protocolo SIP, o TLS e o IPSec.

A pesquisa baseia-se mais em testar os *softphones* e não em testar o protocolo em si, como é o caso a dissertação descrita ao logo desde documento.

2.4 Resumo

Neste capítulo foram apresentadas as tecnologias de voz mais atuais e fez-se uma revisão bibliográfica nos aspectos de segurança e Qualidade de Serviço (QoS).

Pelo estudo efetuado e pela análise da literatura resulta, que é claro o esforço da investigação atual na segurança VoIP e na procura de mecanismos mais adequados às novas ameaças inteligentes. Por outro lado, a segurança tem implicações em termos de desempenho e também nas dificuldades que introduz na utilização, configuração e monitorização dos sistemas VoIP.

3. Arquitetura

Nesta secção, será apresentada a arquitetura do sistema de avaliação de desempenho de sistemas VoIP seguros, bem como o seu objetivo. Descrevem-se os principais componentes e os seus requisitos e mostram-se as funcionalidades esperadas no cenário operacional de testes.

3.1 Introdução

A arquitetura do sistema a desenvolver possuiu diversos intervenientes (ver Figura 3.1), entre os quais os emissores e os recetores das chamadas, que podem ser telefones, computadores com ferramentas adequadas (*softphones*) e geradores/recetores de chamadas (*SIPp*). No núcleo da arquitetura, existe o servidor Asterisk, o Servidor de monitorização ativa (CACTI) e o Servidor de análise de ameaças (*logs*). Para uma maior proximidade dos problemas existentes na internet comum, usou-se um condicionador de tráfego (*Dummynet*) e para a respectiva leitura dos impactos causados usou-se um analisador de tráfego (*Wireshark*).

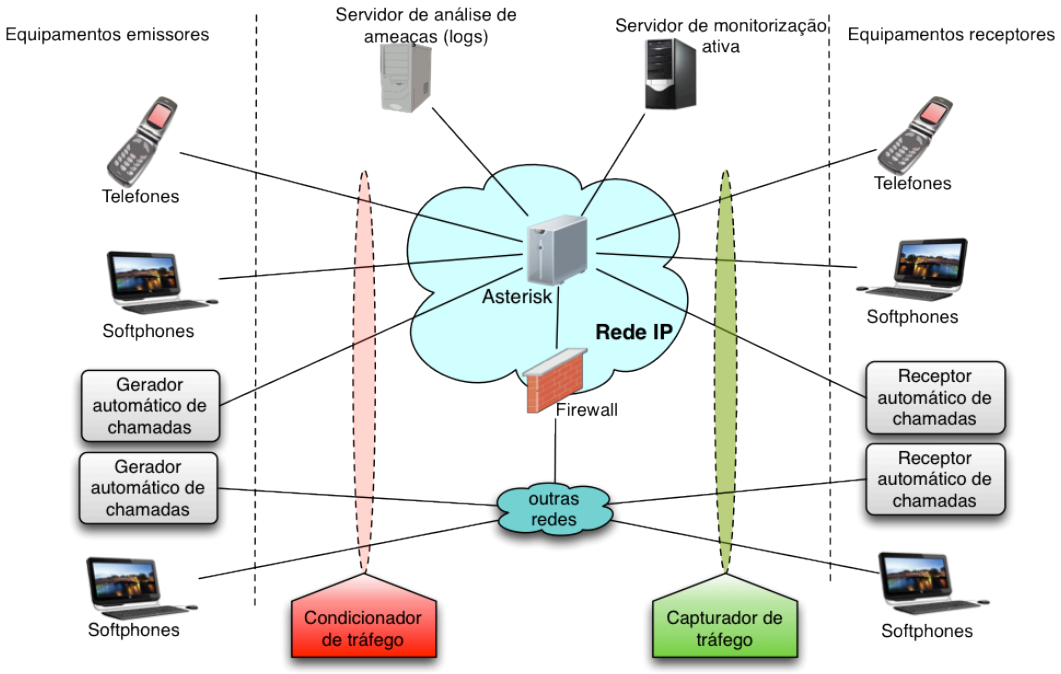


Figura 3.1. Arquitetura.

O esquema aqui apresentado é válido também quando considerados os vários mecanismos de segurança, como o IPSec e o TLS. É possível integrar os componentes num cenário experimental de investigação, mas também em cenários operacionais de monitorização contínua, sem interferir com o utilizador. Para uma maior comodidade dos testes, foi elaborado um conjunto de programas *bash*, que serão muito úteis a quem pretender repetir a experiência.

Os telefones ou softphones conjuntamente com o servidor Asterisk, são os componentes base do servidor VoIP. A rede de suporte é uma rede IP. Os restantes componentes foram acrescentados para construir um sistema de avaliação de desempenho:

- *Capturador de tráfego*: componente que possa ser colocado em pontos específicos da rede para capturar e analisar todo o tráfego VoIP (sinalização e dados) que possa ser gerado. Permitirá aferir as métricas (como o *delay*, *jitter*, *packet loss* e *bandwidth)* com grande rigor. As funcionalidades são a captura e filtragem de tráfego VoIP, análise de valores como o *delay*, o *jitter* e a largura de banda por fluxo.
- *Servidor de monitorização ativa*: componente capaz de obter informações do estado dos sistemas (por exemplo, o CPU) e das suas ligações à rede (por exemplo, estado das interfaces).
- *Servidor de análise de ameaças*: detetor de ameaças de segurança com base na verificação dos ficheiros de *logs*, do servidor Asterisk, detetando padrões de uso suspeitos e gerando alertas adequados (ver Figura 3.2).
- *Condicionador de tráfego*: componente que permite manipular as condições da rede, introduzindo atrasos, perdas ou restrições de largura de banda disponível, ajustando o cenário experimental a diferentes cenários de rede.
- *Gerador automático de chamadas*: componente capaz de gerar automaticamente chamadas de teste em grande número, se necessário, com possibilidade de configuração dos parâmetros de cada chamada.
- *Recetor automático de chamadas*: componente capaz de atender automaticamente chamadas de teste, tanto em termos de sinalização como de voz.

```
[2015-01-23 21:54:52] NOTICE[4643]: chan_sip.c:28062 handle_request_register: Registration from '"7002" <sip:7002@193.136.9.110:5060>' failed for '212.129.61.191:5083'
 - Wrong password
[2015-01-23 21:54:54] NOTICE[4643]: chan_sip.c:28062 handle_request_register: Registration from '"100" <sip:100@193.136.9.110:5060>' failed for '212.129.61.191:5066' -
 Wrong password
[2015-01-23 21:56:16] NOTICE[4643]: chan_sip.c:28062 handle_request_register: Registration from '"9999999" <sip:9999999@193.136.9.110:5060>' failed for '212.129.61.191
:5080' - Wrong password
[2015-01-23 21:56:24] NOTICE[4643]: chan_sip.c:28062 handle_request_register: Registration from '"401" <sip:401@193.136.9.110:5060>' failed for '212.129.61.191:5099' -
 Wrong password
[2015-01-23 21:56:27] NOTICE[4643]: chan_sip.c:28062 handle_request_register: Registration from '"331" <sip:331@193.136.9.110:5060>' failed for '212.129.61.191:5086' -
 Wrong password
[2015-01-23 21:57:18] NOTICE[4643]: chan_sip.c:28062 handle_request_register: Registration from '"205" <sip:205@193.136.9.110:5060>' failed for '212.129.61.191:5073' -
 Wrong password
[2015-01-23 21:57:45] NOTICE[4643]: chan_sip.c:28062 handle_request_register: Registration from '"1031" <sip:1031@193.136.9.110:5060>' failed for '212.129.61.191:5075'
 - Wrong password
[2015-01-23 21:57:50] NOTICE[4643]: chan_sip.c:28062 handle_request_register: Registration from '"774" <sip:774@193.136.9.110:5060>' failed for '212.129.61.191:5106' -
 Wrong password
[2015-01-27 16:09:59] NOTICE[13059]: chan_sip.c:28062 handle_request_register: Registration from '"6543210" <sip:6543210@193.136.9.110:5060>' failed for '94.23.12.154:
5067' - Wrong password
[2015-01-27 16:10:03] NOTICE[13059]: chan_sip.c:28062 handle_request_register: Registration from '"8000" <sip:8000@193.136.9.110:5060>' failed for '94.23.12.154:5167'
- Wrong password
[2015-01-27 16:10:32] NOTICE[13059]: chan_sip.c:28062 handle_request_register: Registration from '"8092" <sip:8092@193.136.9.110:5060>' failed for '94.23.12.154:5099'
- Wrong password
[2015-01-27 16:10:36] NOTICE[13059]: chan_sip.c:28062 handle_request_register: Registration from '"1820" <sip:1820@193.136.9.110:5060>' failed for '94.23.12.154:5119'
- Wrong password
[2015-01-27 16:10:42] NOTICE[13059]: chan_sip.c:28062 handle_request_register: Registration from '"7730" <sip:7730@193.136.9.110:5060>' failed for '94.23.12.154:5108'
- Wrong password
[2015-01-27 16:10:48] NOTICE[13059]: chan_sip.c:28062 handle_request_register: Registration from '"550" <sip:550@193.136.9.110:5060>' failed for '94.23.12.154:5089' -
Wrong password
[2015-01-27 16:11:06] NOTICE[13059]: chan_sip.c:28062 handle_request_register: Registration from '"0000110" <sip:0000110@193.136.9.110:5060>' failed for '94.23.12.154:
5086' - Wrong password
[2015-01-27 16:11:14] NOTICE[13059]: chan_sip.c:28062 handle_request_register: Registration from '"9720" <sip:9720@193.136.9.110:5060>' failed for '94.23.12.154:5155'
- Wrong password
```

Figura 3.2. Análise de ameaças (*logs*).

3.2 Metodologia

Como já foi explicado, pretende-se avaliar o desempenho dos serviços VoIP para os diferentes cenários de segurança. Para tal, será necessário, realizar testes reais e consequentemente medições de parâmetros em ambiente real.

Antes de iniciar qualquer ensaio, foi necessário identificar os diferentes cenários de teste, e seguidamente proceder às instalações das ferramentas de monitorização, como o Wireshark , o CACTI, o Asterisk e o Dummynet, definiram-se os parâmetros de QoS e posteriormente foram efectuadas as configurações dos serviços VoIP necessários, para cada cenário de teste definido. Por fim, foram efetuados os testes e analisados os resultados (ver Figura 3.3).

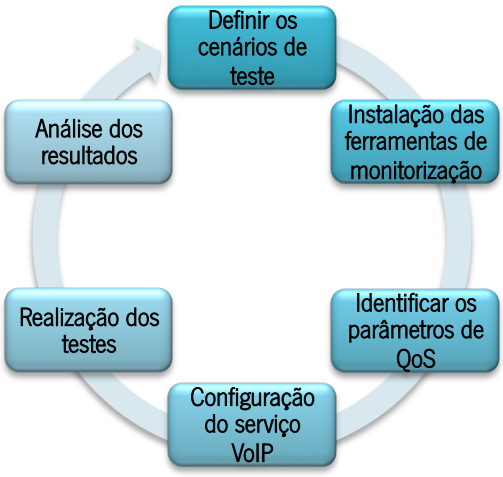


Figura 3.3. Metodologia da dissertação.

3.3 Cenário experimental base

Primeiramente foi realizada uma experiência em ambiente real (ver Figura 3.4), que inclui dois clientes (um com o softphone *Yate* e o outro com o *SFLphone*) e um servidor Asterisk com configurações básicas.

Figura 3.4. Diagramas de blocos da experiência.

Esta experiência tem como finalidade a interpretação das mensagens a serem trocadas. Como base sabe-se que ambos os clientes têm de se registar no Servidor Asterisk (ver Figura 3.5). Depois o cliente que pretenda realizar uma chamada, terá que contactar o Servidor Asterisk e anunciar quem pretende contactar. Seguidamente, o Servidor irá contactar esse mesmo cliente e se este aceitar a chamada, deverá avisar o Servidor de que aceita a chamada. E o servidor avisará o primeiro cliente, ou seja, o cliente que iniciou o processo de chamada. Posteriormente, os dois clientes trocam mensagens entre si. No fim de tudo, o cliente que pretender finalizar a chamada, deverá contactar o Servidor, e este avisará o segundo cliente de que a chamada foi finalizada.

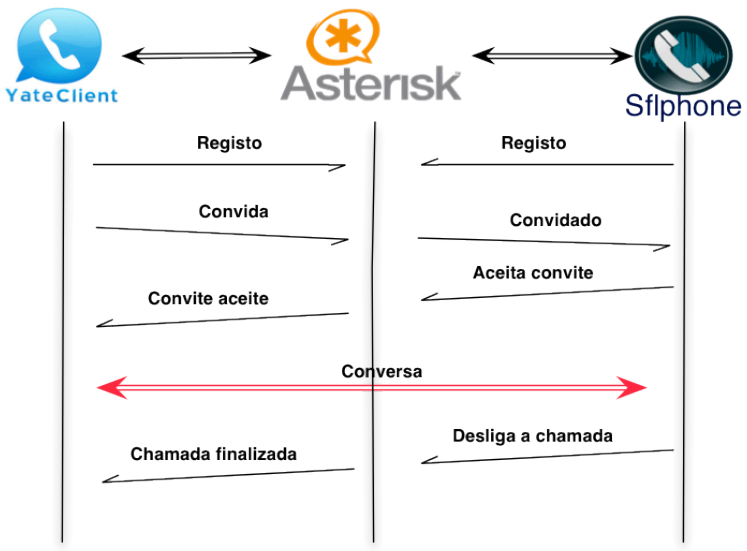


Figura 3.5. Diagramas de sequência de uma chamada.

Após o levantamento de todas as mensagens de sinalização e dos dados necessários, será deduzido um ficheiro em XML (*Extensible Markup Language*). O objetivo do ficheiro XML é o de criar um cenário, que será usado através da ferramenta de emulação SIPp [32]. Com esta ferramenta, será possível simular, uma ou várias, chamadas entre dois clientes.

Depois do XML criado, será capturado o tráfego gerado entre os dois clientes (ver Figura 3.6), afim de fazer a leitura dos parâmetros de QoS.

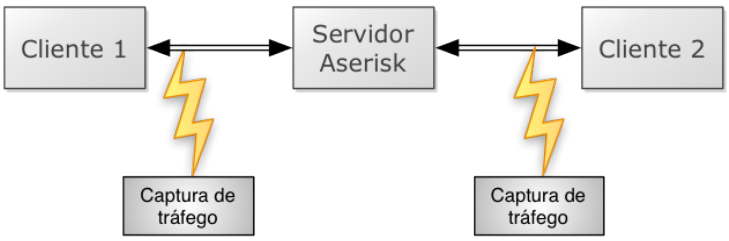


Figura 3.6. Captura de tráfego.

3.3.1 Cenários de teste

Os testes serão realizados com máquinas reais e o principal objetivo é o de efetuar, uma ou mais, chamadas. Para que os testes tenham a máxima fiabilidade, serão sujeitos ás condições normais existentes na internet comum, como atrasos e perda de pacotes.

Nos testes a serem realizados, serão inseridos mecanismos de segurança, sendo o objetivo perceber qual é o impacto causado em cada cenário de segurança (ver Tabela 3.1). No total serão quatro os cenários.

Para além da segurança, o *delay* da rede e o *packet loss* também será variado, recorrendo a uma ferramenta adequada, o *Dummynet* [33], que será falado mais á frente (ver secção 4.4.2).

Tabela 3.1. Cenários de Segurança.

	Sinalização	Dados
Com Segurança	TLS	SRTP
	IPSec	IPSec
	Firewall	*Firewall*
Sem Segurança	SIP	RTP

Com a alteração dos mecanismos de segurança, será esperado um impacto nos parâmetros de QoS, pois exigem as mensagens de controlo que dizem respeito ao mecanismo de segurança. Pois será aplicado um conjunto de critérios de segurança, que provocam atrasos, devido a um conjunto de mensagens e cabeçalhos importantes.

Os mecanismos de segurança, serão aplicados entre os clientes e o servidor (ver Figura 3.7), pois o Servidor é o intermediário. Mecanismos que, num primeiro teste serão aplicados *IPSec* e posteriormente *firewall* à sinalização e aos dados, e numa fase final, será aplicado o *TLS* à sinalização e *SRTP* aos dados.

Figura 3.7. Aplicação dos cenários de segurança.

Resumidamente, será mudado em cada teste:

- Mecanismos de segurança
  - Na sinalização: *TLS*, *IPSec, Firewall* e SIP
  - Nos dados: *SRTP*, *IPSec*, *Firewall* e RTP
- Condições da rede
  - Carga: largura de banda disponível
  - Perdas de pacotes
  - Delay
- Cenário de rede
  - Somente com *switch* em rede local
  - Rede alargada: máquinas em redes distintas

No caso das condições de rede, será usada uma ferramenta que se chama *Dummynet*. Esta ferramenta irá permitir alterar valores como a largura de banda, o atraso e a perda de pacotes.

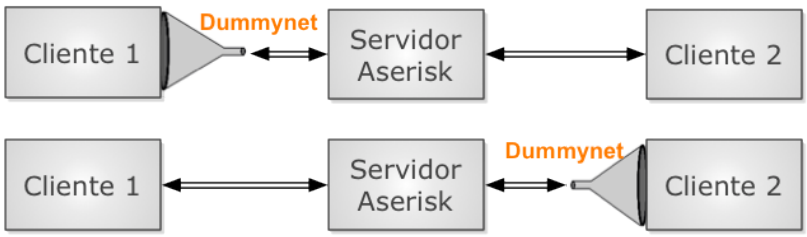


Figura 3.8. Aplicação da ferramenta *dummynet*.

O *dummynet* será aplicado num dos clientes, ou seja, a ferramenta pode ser instalada nos dois clientes mas só será configurada num dado cliente e não nos dois em simultâneo (ver Figura 3.8).

3.3.2 Métricas de avaliação

As métricas de avaliação, serão os valores que irão permitir uma comparação entre os diversos cenários, possibilitando tomar decisões de, por exemplo, o melhor mecanismo de segurança a adotar, face á rapidez *versus* segurança. O *delay* (atraso), *jitter* (variação do atraso), *packet loss* (perda de pacotes) e a *bandwidth* (largura de banda) serão as métricas de avaliação usadas nesta dissertação. Esta leitura será feita sempre do lado do cliente contactado (ver Figura 3.9).

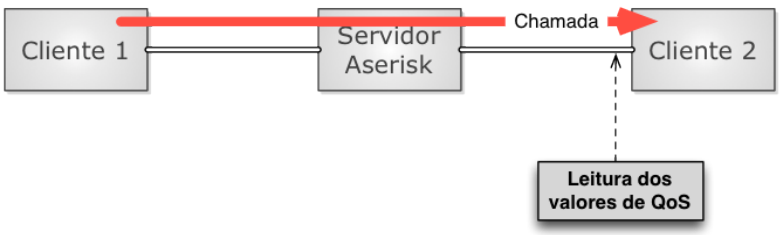


Figura 3.9. Leitura dos parâmetros de QoS.

Para uma melhor leitura visual, os valores obtidos serão apresentados em gráficos. Esses gráficos possuirão quatro colunas, que representam os quatro cenários de segurança adotados (ver secção 5).

3.3.3 Análise e Verificação

Para que o processo de análise/teste tenham a mesma quantidade de tráfego, pelo menos no que diz respeito aos dados RTP, será usado um ficheiro *.pcap (packet capture)* com um som já gravado (ver Figura 3.10), com a duração de 18,98 segundos.

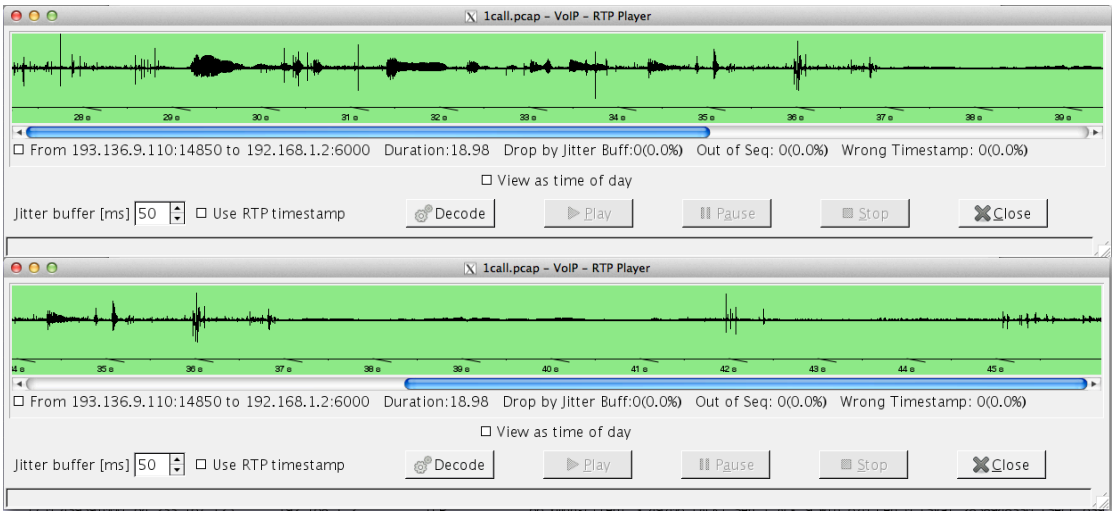


Figura 3.10. Som existente no ficheiro pcap.

Este ficheiro é composto por 949 pacotes UDP que possuem o som a transmitir (ver Figura 3.11).

1	0.000000	192.168.1.111	192.168.1.12	UDP	214	Source port: connected Destination port: telelpathstart
2	0.000046	192.168.1.111	192.168.1.12	UDP	214	Source port: connected Destination port: telelpathstart
3	0.061630	192.168.1.111	192.168.1.12	UDP	214	Source port: connected Destination port: telelpathstart
4	0.061671	192.168.1.111	192.168.1.12	UDP	214	Source port: connected Destination port: telelpathstart
5	0.061699	192.168.1.111	192.168.1.12	UDP	214	Source port: connected Destination port: telelpathstart
6	0.124110	192.168.1.111	192.168.1.12	UDP	214	Source port: connected Destination port: telelpathstart
7	0.124464	192.168.1.111	192.168.1.12	UDP	214	Source port: connected Destination port: telelpathstart
8	0.124500	192.168.1.111	192.168.1.12	UDP	214	Source port: connected Destination port: telelpathstart
9	0.186531	192.168.1.111	192.168.1.12	UDP	214	Source port: connected Destination port: telelpathstart
10	0.186573	192.168.1.111	192.168.1.12	UDP	214	Source port: connected Destination port: telelpathstart
11	0.186600	192.168.1.111	192.168.1.12	UDP	214	Source port: connected Destination port: telelpathstart
12	0.248687	192.168.1.111	192.168.1.12	UDP	214	Source port: connected Destination port: telelpathstart
13	0.248729	192.168.1.111	192.168.1.12	UDP	214	Source port: connected Destination port: telelpathstart
14	0.248757	192.168.1.111	192.168.1.12	UDP	214	Source port: connected Destination port: telelpathstart
15	0.248787	192.168.1.111	192.168.1.12	UDP	214	Source port: connected Destination port: telelpathstart
16	0.311524	192.168.1.111	192.168.1.12	UDP	214	Source port: connected Destination port: telelpathstart
....						
944	18.873749	192.168.1.111	192.168.1.12	UDP	214	Source port: connected Destination port: telelpathstart
945	18.873791	192.168.1.111	192.168.1.12	UDP	214	Source port: connected Destination port: telelpathstart
946	18.873817	192.168.1.111	192.168.1.12	UDP	214	Source port: connected Destination port: telelpathstart
947	18.936287	192.168.1.111	192.168.1.12	UDP	214	Source port: connected Destination port: telelpathstart
948	18.936332	192.168.1.111	192.168.1.12	UDP	214	Source port: connected Destination port: telelpathstart
949	18.936364	192.168.1.111	192.168.1.12	UDP	214	Source port: connected Destination port: telelpathstart

Figura 3.11. Pacotes existes dentro do ficheiro *pcap*.

Depois de todo o processo de estabelecimento da chamada, e segundo a configuração do XML, este som será enviado através do Protocolo RTP (ver Figura 3.12). Trata-se do mesmo conteúdo, mas com protocolo de transporte diferente.

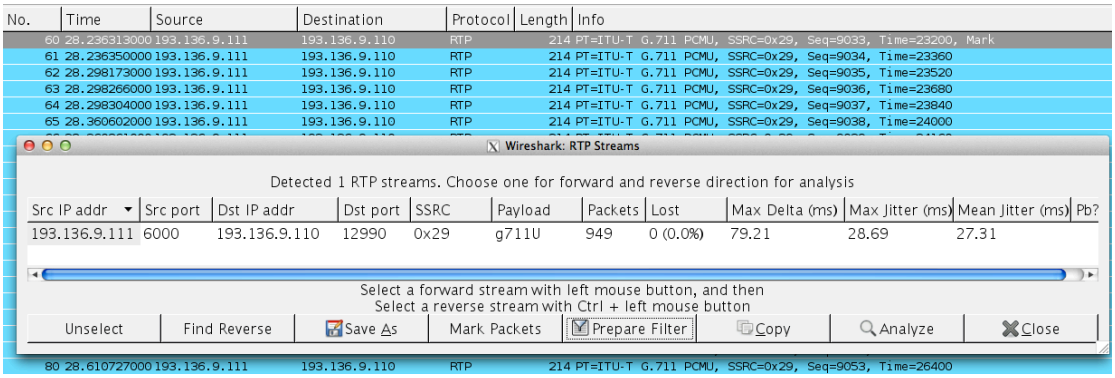


Figura 3.12. Transição para pacotes RTP.

3.4 Resumo

Neste capítulo propôs-se uma arquitetura para um sistema de avaliação do desempenho de sistemas VoIP seguros. Foi também identificada a metodologia de preparação dos diferentes cenários de teste, e as métricas a serem usadas na avaliação.

4. Implementação

Nesta secção será apresentada uma concretização do sistema de avaliação de desempenho de sistemas VoIP seguros, explicitando para cada componente da arquitetura quais as ferramentas escolhidas, bem como a forma como foram usadas e instaladas, no cenário de teste laboratorial adotado.

4.1 Introdução

Para o cenário de teste inicial e básico, foram usados três computadores, um com o *softphone Yate* [34], outro com o *softphone SFLphone* [35] e o terceiro com o *software AsteriskNOW* [36] instalado. Mais informações sobre as máquinas podem ser consultadas na Tabela 4.1.

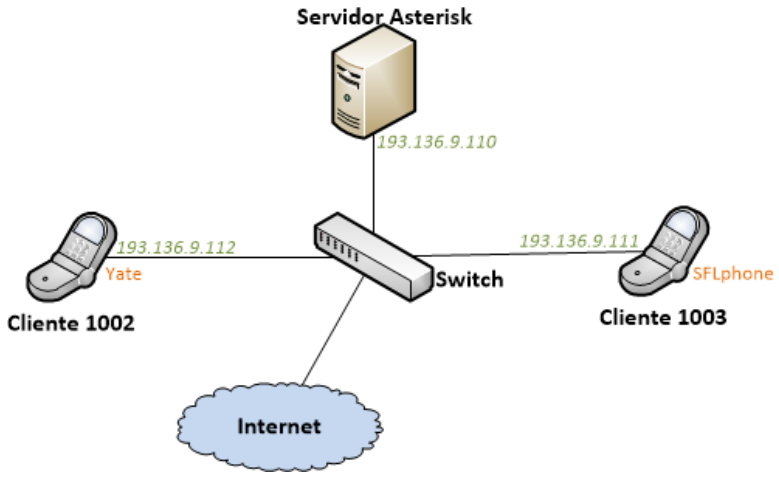


Figura 4.1. Cenário experimental.

Sugestão: Nos Sistemas Operativos (SO) Linux e CentOS as informações do SO podem ser obtidas através do comando "*lsb_release -a*", já no SO Mac OS X podem ser obtidas através do comando "*system_profiler SPSoftwareDataType*".

Tabela 4.1. Características das máquinas envolventes.

Identificador	Ferramenta	Sistema Operativo	Especificações do sistema
Cliente 1002	Yate 4.2.0	Mac OS X Versão 10.7.5 64 bits	*Processador*: 2,4GHz Intel Core i5 *Memória RAM:* 4GB 1333 MHz DDR3 *Disco rígido:* 750GB
Cliente 1003	SFLphone 1.3.0	Linux Mint 16 Petra Edição: Cinnamon 32 bits	*Processador:* 1.66GHz Intel Atom N280 *Memória RAM:* 1 GB *Disco rígido:* 250 GB
Servidor Asterisk	AsteriskNOW 3.0.1	CentOS 6.5 32 bits	*Processador:* 3.10GHz Intel Core i3-2100 *Memória RAM:* 2 GB *Disco rígido:* 40 GB

Esta experiência terá como objetivo inicial o de auxiliar a interpretação de quais as mensagens necessárias para a realização de uma chamada. A ordem e os dados contidos nas mensagens serão um auxílio para a criação do XML.

Depois de efetuado o teste entre os dois *softphones* e com o auxilio da ferramenta *Wireshark* [37], capturou-se uma chamada no lado do recetor (ver Figura 4.2) e outra do lado de emissor (ver Figura 4.3), e posteriormente aplicou-se um filtro, em cada um, para obter pacotes correspondentes a cada chamada. De notar, que por questões de simplicidade de leitura, as imagens apenas apresentam dois dos 949 pacotes RTP.

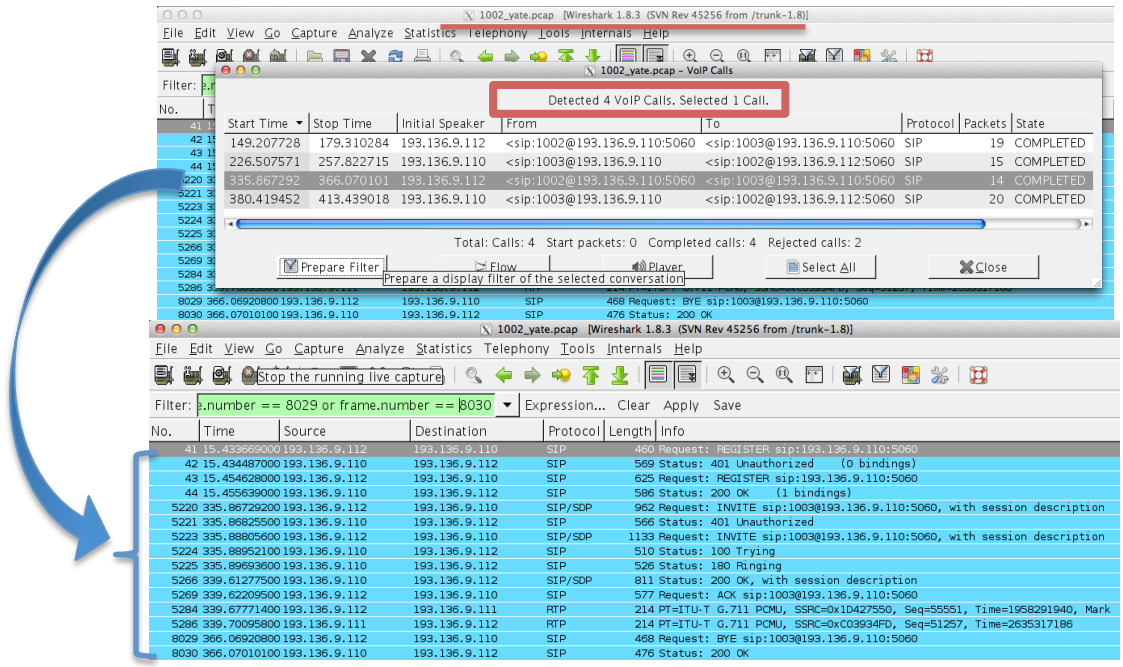


Figura 4.2. Pacotes existentes na chamada selecionada do cliente 1002.

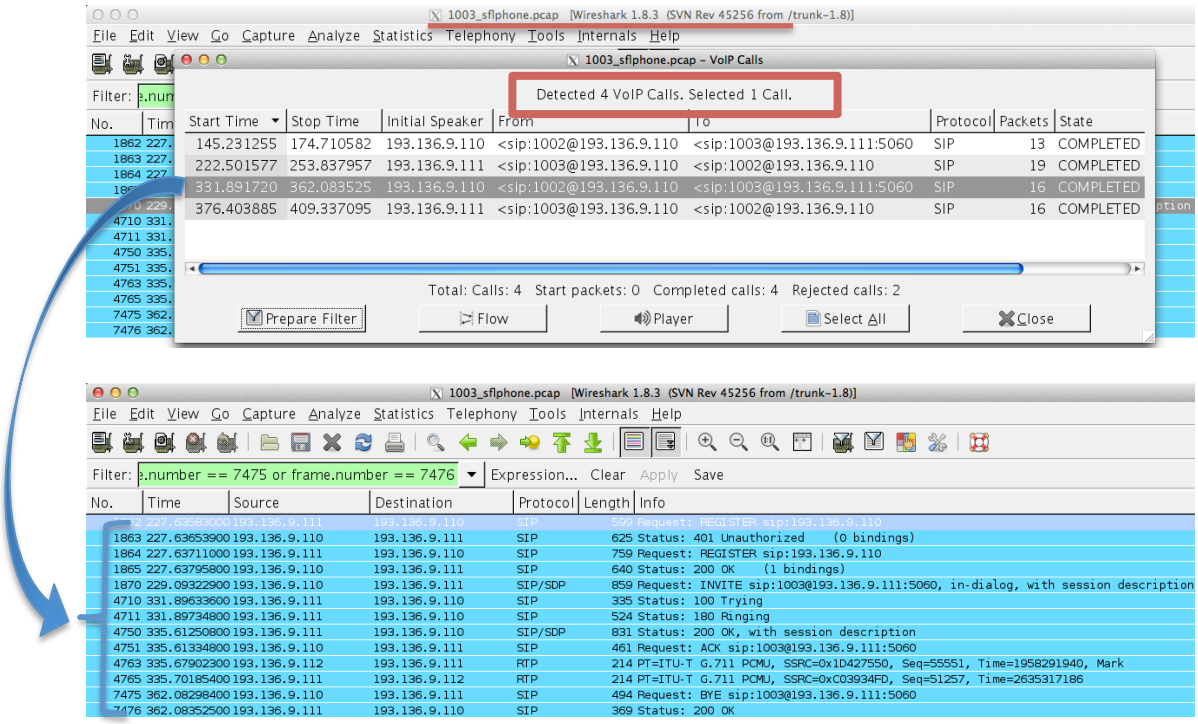


Figura 4.3. Pacotes existentes na chamada selecionada do cliente 1003.

Após alguns testes, foi possível visualizar que na configuração do Asterisk existem mais mensagens a serem trocadas do que esperado na secção 3.3, pois constatou-se que um cliente antes de efetuar ou receber qualquer chamada, tem de estar registado na base de dados do Servidor Asterisk e antes de efetuar um pedido de chamada, tem de se a autenticar.

Depois de uma análise cuidada, foi possível deduzir o diagrama de sequência apresentado na Figura 4.4, onde se pode observar a sequência de mensagens necessárias para uma chamada VoIP bem sucedida.

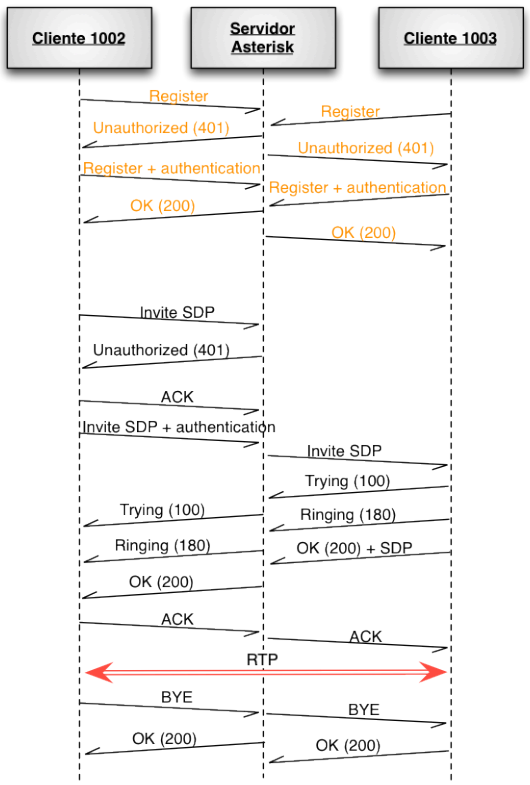


Figura 4.4. Mensagens necessárias numa chamada.

O registo inicial, de cada cliente, é feito através da mensagem "*Register*", que indica a intenção de registo. O Servidor Asterisk envia uma mensagem "*Unauthorized*", ao cliente em questão, indicando que não foi autorizado a registar-se e lança-lhe um desafio. O cliente envia outra mensagem de "*Register*", mas agora com autenticação. Se tudo correr bem o Servidor envia uma mensagem "*OK*".

Para se proceder ao estabelecimento de uma chamada existe um conjunto de mensagens que são necessárias, tais como a mensagem "*Invite*". Esta mensagem indica intenção de realização de chamada. O Servidor responderá com a mensagem "*Unauthorized*", mencionando que não foi autorizando a realizar a chamada e volta a lançar um desafio ao cliente. O cliente envia uma mensagem de "*ACK*" e logo a seguir uma mensagem de "*Invite*", onde contém a autenticação e informação *SDP* (Session Description Protocol). A informação *SDP* serve para negociar parâmetros de sessão, tais como descrição da sessão (versão do protocolo SDP, criador e identificador de sessão, etc), descrição do tempo de sessão, descrição media e quais os *codecs* disponíveis (ver o secção 2.1.4). Assim que o Servidor receba a mensagem "*Invite*", este irá reencaminha-la para o cliente (Cliente 1003) descrito na mensagem. O cliente contactado

enviará uma mensagem de "*Trying*", indicando que está a tentar estabelecer a chamada, e outra de "*Ringing*", indicando que o telefone está a tocar, estas duas mensagens são enviadas ao Servidor e deste para cliente que iniciou todo o processo. Numa fase quase final das mensagens de sinalização, o cliente contactado envia uma mensagem de "*OK*", com informações SDP, ao Servidor e o Servidor ao cliente 1002. Se tudo estiver correto o Cliente 1002 envia uma mensagem "*ACK*" ao Servidor e este ao cliente 1003. Finalmente os clientes passam a trocar pacotes RTP entre si, ou seja, a chamada é estabelecida e os cliente iniciam um diálogo.

Quando um dos clientes pretender finalizar a chamada, será enviada uma mensagem "*BYE*" ao servidor e este enviará um "*OK*", indicando que a chamada foi finalizada. O Servidor por sua vez contacta o outro cliente com a mensagem "*BYE*" e este responde com "*OK*".

4.2 Rede

O cenário de rede adotado para a realização dos testes, é semelhante ao cenário apresentado anteriormente. As máquinas que se fazem passar por clientes, neste cenário, contêm um emulador de tráfego, o SIPp. Este emulador irá permitir, com a ajuda do ficheiro XML, simular um estabelecimento de uma, ou várias, chamada(s) VoIP.

Adicionou-se uma máquina com o *software* CACTI, que é uma ferramenta para monitorização da rede, que será abordado na secção 4.4.1.2.

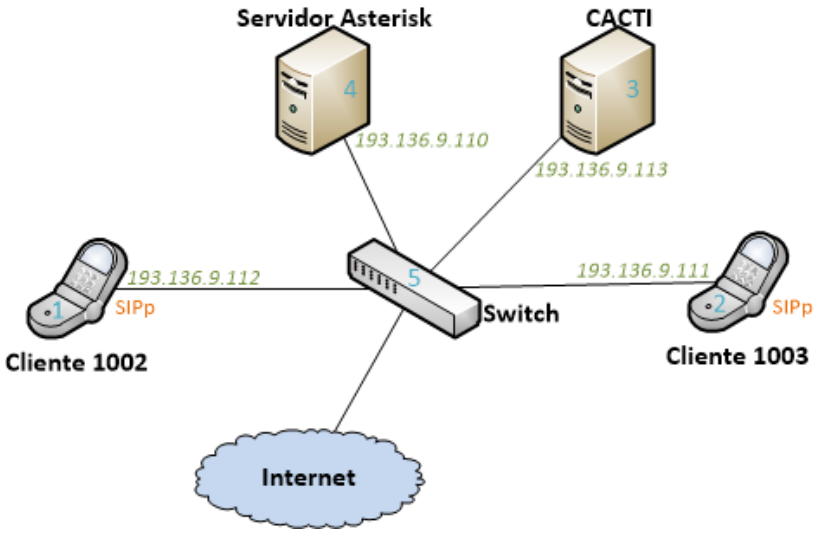


Figura 4.5. Cenário real com SIPp.

Todas as características de cada componente, do cenário descrito na Figura 4.5, poderão ser encontradas na Tabela 4.2.

Tabela 4.2. Características dos componentes envolventes.

Identificador	Descrição	Sistema Operativo	Especificações do sistema
1 (193.136.9.112)	Cliente 1002	Mac OS X Versão 10.7.5 64 bits	*Processador*: 2,4GHz Intel Core i5 *Memória RAM:* 4GB 1333 MHz DDR3 *Disco rígido:* 750GB
2 (193.136.9.111)	Cliente 1003	Linux Mint 16 Petra Edição: Cinnamon 32 bits	*Processador:* 1.66GHz Intel Atom N280 *Memória RAM:* 1 GB *Disco rígido:* 250 GB
3 (193.136.9.113)	CACTI	Linux Mint 16 Petra Edição: Cinnamon 32 bits	*Processador:* 2.80GHz Intel Celeron *Memória RAM:* 1 GB *Disco rígido:* 73 GB
4 (193.136.9.110)	Servidor Asterisk	CentOS 6.5 32 bits	*Processador:* 3.10GHz Intel Core i3-2100 *Memória RAM:* 2 GB *Disco rígido:* 40 GB
5	Switch	Cisco Catalyst 2950 (com 48 portas de 100Mbps e 2 portas de 1Gbps)	

Quanto à capacidade da rede, e segundo a ferramenta *iperf*, o cenário apresentado na Figura 4.5 possui uma capacidade de 94,1Mbits/s (ver Figura 4.6).

O *iperf* [38] é uma ferramenta de código aberto, desenvolvida em *C++,* que funciona sobre os Sistemas Operativos Linux e Windows e tem como objetivo medir o desempenho da rede, nomeadamente a largura de banda, o *jitter* e o *packet loss*. Possui funcionalidade de cliente/servidor, assim como faz as medições de forma unidirecional ou bidirecional (exemplo no servidor: **#iperf -s**; no cliente: **#iperf -c 193.136.9.110**).

```
raquelpereira@cliente 1002:~$ iperf -c 193.136.9.110 -p 5060
------------------------------------------------------------
Client connecting to 193.136.9.110, TCP port 5060
TCP window size:  129 KByte (default)
------------------------------------------------------------
[  4] local 193.136.9.112 port 49967 connected with 193.136.9.110 port 5060
[ ID] Interval       Transfer     Bandwidth
[  4]  0.0-10.0 sec   112 MBytes  94.1 Mbits/sec
```

Figura 4.6. Informação da rede.

4.3 Configurações de segurança

Neste tópico serão apresentadas todas as configurações de segurança usadas para o *TLS* e para o *IPSec*.

4.3.1 TLS (Transport Layer Security)

O protocolo *Transport Layer Security* (TLS) é baseado no protocolo *Secure Sockets Layer* (SSL), descrito no RFC 5246 [39]. Ambos os protocolos são protocolos criptográficos, que têm por finalidade a oferta de segurança em comunicações na Internet, tais como o e-mail, navegação em páginas, entre outros.

O principal objetivo do protocolo TLS é o de promover a confidencialidade e a integridade dos dados entre dois aplicativos que se comunicam. O protocolo é composto por duas camadas: a *Record* e a *Handshake* (ver Figura 4.7). A camada *Record* é utilizado para encriptar os dados da camada de Aplicação, por forma a não serem intercetadas por terceiros. A camada *Handshake* é divida em três protocolos, o *Handshake*, o *Change Cipher Spec* e o *Alert*. O *Handshake* faz negociações entre cliente e servidor, tais como a negociação de algoritmos criptográficos a serem usados e a troca de certificados. O *Change Cipher Spec* trata-se de uma mensagem, que notifica o recetor, de que as mensagens seguintes serão cifradas segundo as chaves acabadas de negociar. O *Alert* também é uma mensagem, mas pretende notificar uma ocorrência de erros ou uma ocorrência importante. A camada TLS a usar nesta dissertação é a *TLS Handshake Protocol* [40].

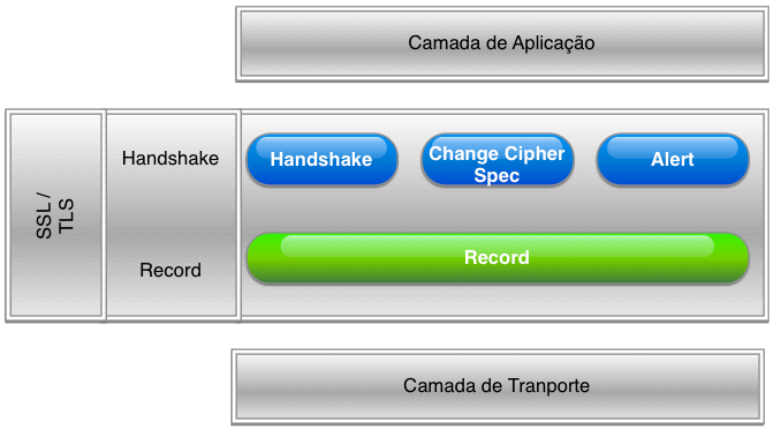


Figura 4.7. Camada SSL/TLS.

4.3.1.1 Certificados

Para a correta configuração do TLS, foi necessário gerar certificados, quer para o servidor Asterisk, quer para os clientes. Os procedimentos para a criação dos certificados começam por, primeiramente e por uma questão de organização, criar uma pasta com o nome "*keys*", segundo o comando "*mkdir keys*". No ponto intermédio, tem-se que descarregar a script "*ast_tls_cert*" que vai permitir a criação dos tais certificados, através do comando "*wget https://raw.github.com/rillian/asterisk-opus/master/contrib/scripts/ast_tls_cert*".

A script "*ast_tls_cert*" contém diversas formas de execução, sendo que tem diversos campos associados que podem ser passados como parâmetros na linha de comando. Com a ajuda do comando "**./ast_tls_cert – h**", obteve-se as informações descritas na Tabela 4.3.

Tabela 4.3. Campos disponíveis na script "*ast_tls_cert*".

Campo	Valores exemplo	Descrição do campo
-m	"client" ou "server"	Identifica se o certificado será para um cliente ou para um servidor, se nada for dito é assumido como servidor
-c	"keys/ca.crt"	Nome/caminho do certificado da Autoridade de Certificação
-k	"keys/ca.key"	Nome/caminho do ficheiro com a chave da Autoridade de Certificação. Isto só ocorre para os clientes.
-C	"193.136.9.110"	Pode ser o domínio ou o endereço IP da máquina, quer seja cliente ou servidor. Tem de ser único.
-O	"Uminho"	Nome da organização
-o	"cliente1002"	Nome para os ficheiros de saída
-d	"keys"	Diretoria onde serão guardados os ficheiros de saída

Depois de entendidos todos os campos existentes na script, os certificados usados na experiência, foram gerados através dos comandos, pela seguinte ordem:

1. ./ast_tls_cert -m server -C 193.136.9.110 -O "Servidor Asterisk" -o asterisk -d keys
2. ./ast_tls_cert -m client -C 193.136.9.111 -O "cliente1003" -o cliente1003 -d keys -c keys/ca.crt -k keys/ca.key

3. ./ast_tls_cert -m client -C 193.136.9.112 -O "cliente1002 -o cliente1002 " -d keys -c keys/ca.crt -k keys/ca.key

No primeiro ponto são criados os ficheiros da Autoridade de Certificação (CA) e do Servidor Asterisk. No segundo e no terceiro passo, com a invocação do ficheiros da CA, são criados os certificados dos clientes 1002 e 1003.

Após a execução dos comandos acima, obteve-se dezasseis ficheiros, apresentados na Figura 4.8.

```
[root@asterisk asterisk]# ls -lisa keys
total 72
1836797 4 drwxr-xr-x  2 root     root     4096 Set 23 21:34 .
1835731 4 drwxrwxr-x. 7 asterisk asterisk 4096 Set 23 21:35 ..
1836819 4 -rw-r--r--  1 root     root     1229 Set 23 21:32 asterisk.crt
1836818 4 -rw-r--r--  1 root     root      578 Set 23 21:32 asterisk.csr
1836817 4 -rw-r--r--  1 root     root      887 Set 23 21:32 asterisk.key
1836821 4 -rw-r--r--  1 root     root     2116 Set 23 21:32 asterisk.pem
1836803 4 -rw-r--r--  1 root     root      163 Set 23 21:31 ca.cfg
1836816 4 -rw-r--r--  1 root     root     1773 Set 23 21:32 ca.crt
1836814 4 -rw-r--r--  1 root     root     3311 Set 23 21:32 ca.key
1836825 4 -rw-r--r--  1 root     root     1220 Set 23 21:34 cliente1002.crt
1836823 4 -rw-r--r--  1 root     root      570 Set 23 21:34 cliente1002.csr
1836822 4 -rw-r--r--  1 root     root      887 Set 23 21:34 cliente1002.key
1836827 4 -rw-r--r--  1 root     root     2107 Set 23 21:34 cliente1002.pem
1842825 4 -rw-r--r--  1 root     root     1220 Set 23 21:34 cliente1003.crt
1842824 4 -rw-r--r--  1 root     root      570 Set 23 21:34 cliente1003.csr
1842823 4 -rw-r--r--  1 root     root      887 Set 23 21:34 cliente1003.key
1842826 4 -rw-r--r--  1 root     root     2107 Set 23 21:34 cliente1003.pem
1836799 4 -rw-------  1 root     root      120 Set 23 21:34 tmp.cfg
```

Figura 4.8. *Output* dos ficheiros criados.

Depois de todos estes procedimentos, será explicado, com o auxilio da Figura 4.9 e no ponto 4.3.1.2, qual o papel desempenhado por estes certificados criados.

De acordo com as extensões dos ficheiros apresentados na Figura 4.8, "pem" significa "*Privacy Enhanced Mail*", faz uso de criptografia de chave pública e inclui o certificado público. A extensão "crt" (*certificate*) é usada para os certificados, que é utilizado pelos browsers para verificar a autenticidade de um site e permite que os browsers se conectem com segurança, usando o SSL. A extensão "key", é usada tanto para PKCS#8 (*Public-Key Cryptography Standards*) pública ou privada. PKCS#8 define uma sintaxe padrão para armazenar informação da chave privada. A extensão "csr" (*Certificate Signing Request*), refere-se a um ficheiro que possui todos os detalhes do certificado requerido, como por exemplo, o utilizador, o país, a

organização, assim como a chave pública. Se este for assinado pela Autoridade de Certificação, será retornado um certificado público, não a chave [41].

4.3.1.2 Funcionamento

De acordo com a Figura 4.9, o "Cliente" deve contactar o "Servidor", mostrando interesse em comunicar com o mesmo. Posteriormente o "Servidor" entrega o seu certificado ao "Cliente". Seguidamente o "Cliente" procura obter um certificado da "Autoridade de Certificação" e valida o certificado que recebeu do "Servidor". Depois de este ser validado com sucesso o "Cliente" envia o seu certificado ao "Servidor", este após receber o certificado, irá executar o mesmo processo que o cliente, ou seja, pega no certificado do Cliente, obtém o certificado da "Autoridade de Certificação" e se a validade for confirmada o "Servidor" deverá notificar o "Cliente" do sucedido. Posto isto, o "Cliente" sabe com quem realmente está a falar e gera uma chave de sessão, encripta e envia-a ao "Servidor". O "Servidor" desencripta a chave e ambos passam a ter uma conversa encriptada segundo a chave de sessão previamente acordada.

De salientar que autenticação do cliente é opcional, ou seja, o cliente pode não ter necessidade de enviar o seu certificado, nem o servidor de o validar.

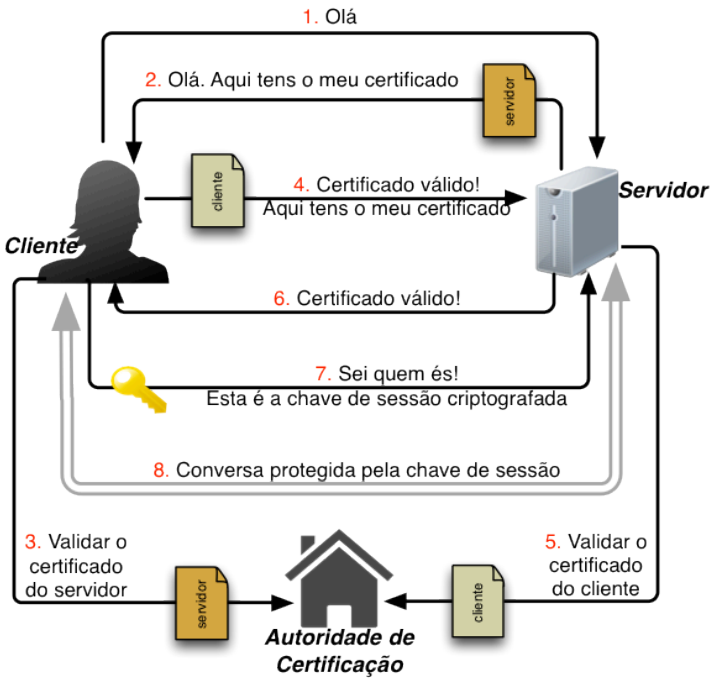


Figura 4.9. Ilustração do uso dos certificados entre as diferentes entidades.

Para uma melhor explicação dos procedimentos do TLS, esboçou-se um esquema que ilustra isso mesmo. Esse esquema (ver Figura 4.10) teve como base a troca de pacotes apresentados na Figura 4.11.

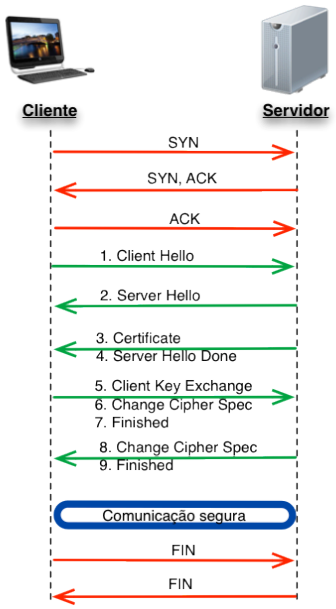


Figura 4.10. Esquema de mensagens trocadas numa conexão TLS.

No.	Time	Source	Destination	Protocol	Length	Info
26	0.000000000	193.136.9.111	193.136.9.110	TCP	74	57974 > sip-tls [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=1076604 TSecr=0 WS=1
27	0.000243000	193.136.9.110	193.136.9.111	TCP	74	sip-tls > 57974 [SYN, ACK] Seq=0 Ack=1 Win=14480 Len=0 MSS=1460 SACK_PERM=1 TSval=245216307
28	0.000068000	193.136.9.111	193.136.9.110	TCP	66	57974 > sip-tls [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSval=1076604 TSecr=245216307
29	0.005113000	193.136.9.111	193.136.9.110	TLSv1	247	Client Hello
31	0.000724000	193.136.9.110	193.136.9.111	TLSv1	1514	Server Hello
33	0.000079000	193.136.9.110	193.136.9.111	TLSv1	842	Certificate, Server Hello Done
35	0.019017000	193.136.9.111	193.136.9.110	TLSv1	264	Client Key Exchange, Change Cipher Spec, Finished
36	0.002351000	193.136.9.110	193.136.9.111	TLSv1	300	New Session Ticket, Change Cipher Spec, Finished
37	0.001219000	193.136.9.111	193.136.9.110	SIP	700	Request: REGISTER sips:193.136.9.110:5061;transport=TLS
160	33.528655000	193.136.9.111	193.136.9.110	SIP/SDP	1148	Request: INVITE sips:600@193.136.9.110:5061;transport=TLS, with session description
173	0.128608000	193.136.9.111	193.136.9.110	SRTP	224	PT=ITU-T G.711 PCMU, SSRC=0xBF1DA10C, Seq=4356, Time=2583030314
174	0.000871000	193.136.9.110	193.136.9.111	SRTP	224	PT=ITU-T G.711 PCMU, SSRC=0x29B0D7F7, Seq=15157, Time=160, Mark
175	0.008107000	193.136.9.111	193.136.9.110	SRTP	224	PT=ITU-T G.711 PCMU, SSRC=0xBF1DA10C, Seq=4357, Time=2583030474
333	1.516622000	193.136.9.111	193.136.9.110	TCP	66	57974 > sip-tls [FIN, ACK] Seq=7936 Ack=8597 Win=42368 Len=0 TSval=1085407 TSecr=245251499
336	0.000380000	193.136.9.110	193.136.9.111	TCP	66	sip-tls > 57974 [FIN, ACK] Seq=8634 Ack=7937 Win=41344 Len=0 TSval=245251519 TSecr=1085407

Figura 4.11. Mensagens trocadas numa conexão TLS.

A Figura 4.10, apresenta todas as mensagens necessárias para o estabelecimento de uma conexão TLS. Numa fase inicial, é necessário estabelecer uma ligação TCP (linhas a vermelho) e para tal são necessárias algumas mensagens como a SYN, SYN/ACK e ACK. A mensagem de SYN especifica o número de sequência inicial, ou seja, indica os números de sequência que devem ser sincronizados para iniciar a ligação, a de SYN/ACK especifica o tamanho do *buffer* de recepção e especifica o número de sequência inicial, por fim a mensagem de ACK indica que o

número de sequência de confirmação é válido. Depois de serem trocadas as mensagens TLS e ainda falando do TCP, existe uma outra mensagem, a FIN. Esta pretende indicar que a transmissão de envio de dados terminou.

Para uma breve interpretação das mensagens TLS (linhas verdes), pode ser consultada a Tabela 4.4, mas seguindo a ordem de ocorrência das mensagens na figura, o processo para estabelecimento de uma conexão TLS é descrita da seguinte forma [42]:

1. O cliente envia a versão do TLS que usa, uma lista de cifras suportados e uma sequência aleatória que o servidor irá precisar mais tarde;
2. O servidor envia também, a sua versão do TLS, a cifra que escolheu e uma *string* aleatória que o cliente irá usar mais tarde;
3. O servidor envia o seu certificado ao cliente, provando assim a sua identidade;
4. O servidor verifica que o certificado é válido e passa ao cliente alguns parâmetros;
5. O cliente envia um segredo que será usado por ambos, para gerar as chaves de sessão;
6. O cliente avisa de que todas as mensagens que se seguem serão encriptadas segunda a chave de sessão negociada;
7. O cliente envia um *hash* dos componentes do *handshake*, encriptado com a chave de sessão gerada. O servidor irá usar esse *hash* para verificar a integridade do processo *handshake*;
8. O servidor repete o processo descrito no ponto 6;
9. O servidor faz o mesmo processo que o do ponto 7.

Tabela 4.4. Designação de cada mensagem TLS.

Campo	Descrição
"Client Hello"	Envia a versão do TLS, a lista de cifras suportadas e uma *string* aleatória
"Server Hello"	Envia a versão do TLS, a cifra escolhida e uma *string* aleatória
"Certificate"	Envia o certificado
"Server Hello Done"	O certificado é válido e passa parâmetros
"Client Key Exchange""	Envia um segredo
"Change Chiper Spec"	Informa que todas as mensagens serão encriptadas segundo a chave de sessão gerada
"Finished"	Envia um *hash* dos componentes *handshake*, encriptado com a chave de sessão gerada

4.3.1.3 Configuração no Asterisk

Para ativar o protocolo TLS no servidor Asterisk (explicado na secção 4.4.3), foi necessário efetuar as configurações no ficheiro *extensions.conf* e *sip.conf* apresentadas na Figura 4.12 e na Figura 4.13, respectivamente.

```
[demo]
exten => 1003,1,Set(CHANNEL(secure_brigde_signaling)=1)
        same => n,Set(CHANNEL(secure_bridge_media)=1)
        same => n,Dial(SIP/1003@193.136.9.110)

exten => 1002,1,Set(CHANNEL(secure_brigde_signaling)=1)
        same =>n,Set(CHANNEL(secure_bridge_media)=1)
        same =>n,Dial(SIP/1002@193.136.9.110)
```

Figura 4.12. Configuração no ficheiro *extensions.conf*.

```
; aplicar a segurança TLS
tlsenable=yes
tlsbindaddr=0.0.0.0
tlscertfile=/etc/asterisk/keys/asterisk.pem
tlscafile=/etc/asterisk/keys/ca.crt
tlscipher=ALL
tlsclientmethod=tlsv1
transport=tls

; ********************  configuração dos utilizadores *********

[1003]
; pode fazer ou receber chamadas
type=friend
host=dynamic
secret=user1003voipuminho2014
port=5061
context=demo
disallow=all
allow=ulaw
allow=alaw
allow=g722
transport=tls
encryption=yes
username=1003 ;
canreinvite=yes

[1002]
; pode fazer ou receber chamadas
type=friend
host=dynamic
secret=user1002voipuminho2014
port=5061
context=demo
disallow=all
allow=ulaw
allow=alaw
allow=g722
transport=tls
encryption=yes
username=1002 ;
canreinvite=yes
```

Figura 4.13. Configuração no ficheiro *sip.conf*.

De acordo com as configurações apresentadas na Figura 4.13, destacam-se alguns "parâmetros". Comandos esses, que são acrescentados em relação à configuração normal e que permitem a ativação e configuração do protocolo TLS, a descrição de cada um deles pode ser encontrada na Tabela 4.5.

Tabela 4.5. *Flags* TLS usadas na configuração.

Campo	Descrição
tlsenable = yes	Ativa o servidor TLS de entrada, por defeito é "no"
tlsbindaddr = 0.0.0.0	Endereço IP do servidor TLS
tlscertfile = /etc/asterisk/keys/ asterisk.pem	Caminho até ao certificado do servidor, a usar nas conexões TLS
tlscafile = /etc/asterisk/keys/ ca.crt	Caminho até ao certificado que verifica a autenticidade dos certificados
tlscipher=ALL	A cifra "ALL" habilita todos os pacotes de criptografia, exceto as cifras eNULL, que devem ser explicitamente habilitados
tlsclientmethod=tlsv1	Especifica a versão do protocolo TLS, que pode ser tlsv1, sslv3 ou sslv2. Sendo o tlsv1 mais aconselhado por questões de segurança
transport=tls	Especificação do protocolo de transporte, podendo ser UDP, TCP ou TLS
port=5061	Especifica a porta para os pedidos TLS
encryption=yes	Habilita o uso do protocolo SRTP

4.3.2 IPSec (*IP Security*)

O *IPSec* (*IP Security*) encontra-se descrito no RFC 6071 [43] do IETF e opera na camada de rede do modelo OSI (*Open Systems Interconnection*). Possui diversos parâmetros e configurações possíveis, sendo muitas das vezes considerado complexo. A comunicação entre duas quaisquer entidades IP, pode incluir mecanismos de segurança IPSec, capazes de dar garantias de integridade, autenticação de origem e opcionalmente confidencialidade. Atua na camada de rede, sendo que os seus dois protocolos principais são, o ESP (*Encapsulation Security Payload*) e o AH (*Authentication Header*). Estes dois protocolos atuam de forma independente (podendo ser misturados, embora seja muito raro [44]) e têm tarefas muito importantes, pois o AH autentica e garante a integridade da ligação, enquanto o ESP encripta os dados da ligação e também autentica, certificando de que ninguém poderá visualizar os dados contidos no pacote e garante a integridade da ligação. Para além destes dois protocolos, o *IPSec*

possui ainda o IKE (*Internet Key Exchange*), que é usado para estabelecer uma associação segura (SA) com a qual se poderão negociar os restantes parâmetros.

Tabela 4.6. Protocolos e algoritmos suportados pelo *IPSec.*[45]

Protocolo	Serviço de segurança fornecido	Algoritmo suportado
ESP	Confidencialidade através de cifragem e, opcionalmente, integridade dos dados	AES (*Advanced Encryption Standard*) em modo CBC (*Cipher Block Chaining*) ou modo CTR (*Counter mode*)
AH	Integridade e autenticação de origem dos dados	HMAC-SHA1-96, AES-XCBC-MAC-96, HMAC-SHA-256
IKE	Negociação de parâmetros de conexão	Esquema Diffie-Hellman em grupos de 1024 bits ou 2048 bits e AES no modo PRNG (*Pseudo-Random Number Generators*)

AH (Authentication Header):

O AH descrito no RFC 4302 [46], é usado para autenticar em IPv4 e IPV6, não encripta, mas dá garantia de integridade, ou seja, dá garantia de com quem estamos realmente a falar. O cabeçalho deste protocolo encontra-se na Figura 4.14 e a descrição de cada uns dos campos encontra-se na Tabela 4.7.

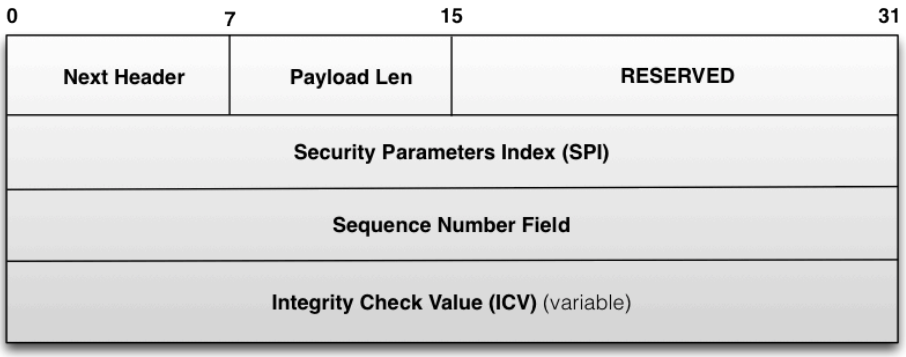


Figura 4.14. Cabeçalho do AH [adaptado de [46]]

Tabela 4.7. Descrição de cada campo do cabeçalho AH [46].

Campo	Definição
Next Header (8 bits)	Apresenta o tipo de protocolo usado no cabeçalho seguinte, por exemplo, "4" indica IPv4, "41" indica IPv6 e o "6" indica TCP.
Payload Len (8 bits)	Tamanho do cabeçalho, em palavras de 32 bits, menos duas.
RESERVED (16 bits)	Para uso futuro
Security Parameters Index (32 bits)	Valor arbitrário utilizado pelo recetor para identificar a SA (*Security Association*)
Sequence Number Field (32 bits)	Possui um valor que aumenta em uma unidade por cada pacote enviado. Permitindo ao recetor detetar repetição e/ou ordenação de pacotes
Integrity Check Value (variável e opcional)	Este campo verifica a integridade do valor calculado sobre o cabeçalho ESP, dados e campo "finalizador ESP". Só é usada se o serviço de integridade for selecionado.

ESP (Encapsulation Security Payload):

O ESP descrito no RFC 4303 [47], pode ser usado para fornecer autenticação ou confidencialidade. Este protocolo foi projetado para fornecer uma combinação de serviços de segurança em IPv4 e IPv6, sendo usado para garantir a confidencialidade, a autenticação de origem, a integridade *connectionless*, serviço anti-repetição e limitação de confidencialidade no fluxo de tráfego. Na Figura 4.15 pode-se encontrar a estrutura de um cabeçalho ESP e na Tabela 4.8 existe uma breve descrição de cada campo [47].

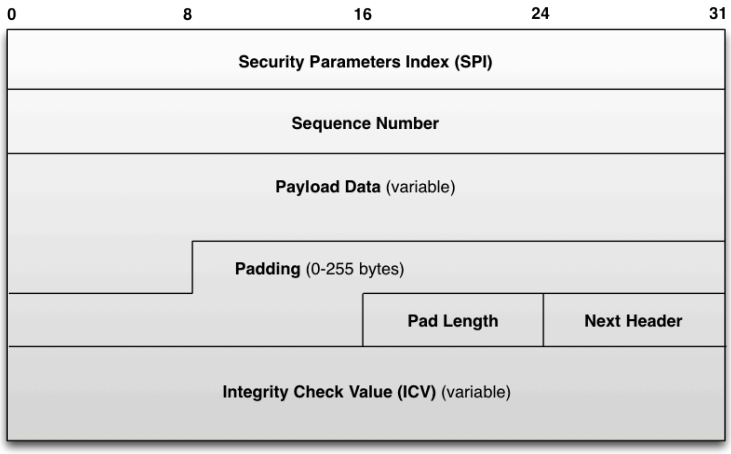


Figura 4.15. Cabeçalho do ESP [adaptado de [47]]

Tabela 4.8. Descrição de cada campo do cabeçalho ESP [47].

Campo	Definição
Security Parameters Index (32 bits)	Valor arbitrário utilizado pelo recetor para identificar a SA (*Security Association*)
Sequence Number (32 bits)	Possui um valor que aumenta em uma unidade por cada pacote enviado. Permitindo ao recetor ordenar os pacotes e/ou detetar repetição
Payload Data (varivável)	Possui dados descritos pelo campo "*Next Header*". É obrigatório e é um número inteiro de bytes.
Padding (0-255 bytes)	Usado para auxiliar um algoritmo de criptografia ou mesmo para garantir que o texto cifrado termine em um limite de 4 bytes. Usado também para esconder o comprimento real da carga útil
Pad Length (0-255 bytes)	É obrigatório e indica o número de bytes de preenchimento no campo "*Padding*". Que pode variar de 0-255, onde "0" indica que não há bytes de preenchimento
Next header (8 bits)	Apresenta o tipo de protocolo usado no cabeçalho seguinte, por exemplo, "4" indica IPv4, "41" indica IPv6 e o "6" indica TCP.
Integrity Check Value (variável)	Este campo verifica a integridade do valor calculado sobre todo o pacote.

4.3.2.1 Modos de funcionamento

O *IPSec* possui dois modos de funcionamento, o modo túnel e o modo transporte. De acordo com as imagens apresentadas abaixo, é possível visualizar na Figura 4.16 o pacote original, ou seja, um pacote IP que possui informações a ser entregues a uma dada máquina. De notar que no protocolo AH, tanto o cabeçalho como os dados são autenticados. Já no protocolo ESP o cabeçalho é autenticado e os dados para além de serem autenticados, também são encriptados.

Figura 4.16. Pacote de Dados IP original.

Modo Túnel:

O modo túnel pega no datagrama IP e encapsula-o dentro de um novo cabeçalho, este é normalmente usado para configurar as VPNs (*Virtual Private Network*). A Figura 4.17 e a Figura 4.18 demostram o datagrama resultante no modo túnel e com os protocolos AH e ESP.

Figura 4.17. Modo túnel com o AH [adaptado de [44]].

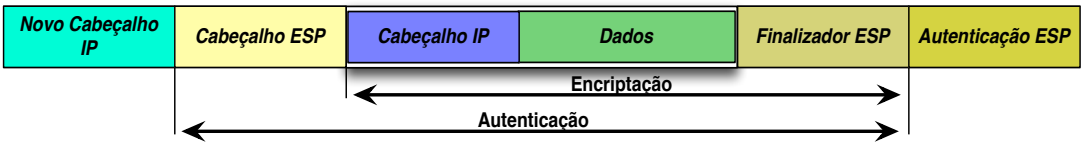


Figura 4.18. Modo túnel com o ESP [adaptado de [44]].

Modo Transporte:

O modo transporte preserva o cabeçalho IP, de acordo com a Figura 4.19 e a Figura 4.20, este apenas insere um novo cabeçalho, AH ou ESP, entre o "Cabeçalho IP" e os "Dados".

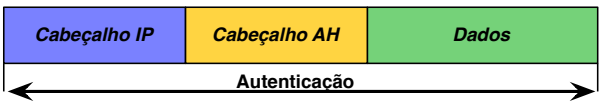


Figura 4.19. Modo transporte com o AH [adaptado de [44]].

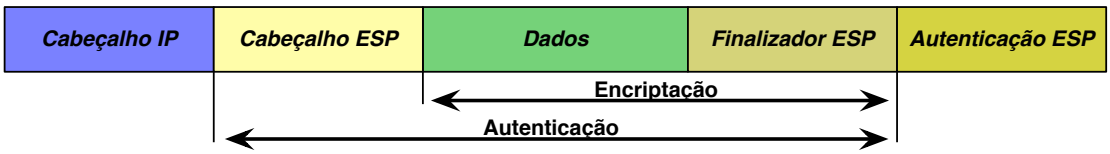


Figura 4.20. Modo transporte com o ESP [adaptado de [44]].

4.3.2.2 Configuração

A configuração a usar será em modo túnel, porque o pacote é protegido por inteiro, e com chave pré-partilhada, pois permite criar túneis confidenciais adicionando cabeçalhos ESP. O túnel será configurado entre duas máquinas (ver Figura 4.21).

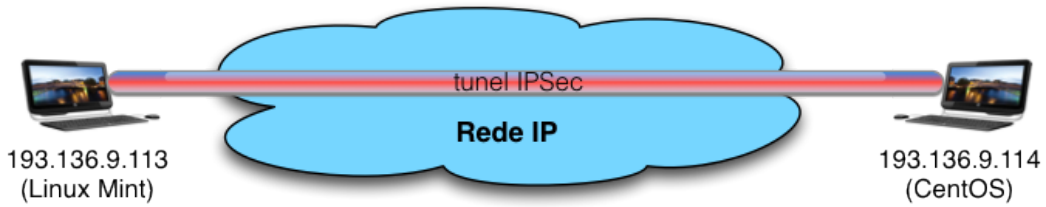


Figura 4.21. Ilustração da aplicação de um túnel *IPSec*.

As implementações *IPSec* disponíveis para Linux possuem sempre duas componentes distintas. A parte do *Kernel*, ou seja, do núcleo do Sistema Operativo, que é responsável por acrescentar os cabeçalhos AH e/ou ESP no envio e validá-los na receção. E a parte sobre o controlo do utilizador do sistema (*User Space*), que é responsável por lidar com as configurações, definir políticas de utilização *IPSec* (manipulando as tabelas de políticas) e gerindo as entradas nas tabelas de associações seguras. Na parte do utilizador inclui-se também o servidor IKE capaz de negociar associações seguras com outros sistemas, de forma automática.

A maioria das distribuições Linux atuais usa o NETKEY como implementação *IPSec* no *Kernel*. O NETKEY é parte integrante do código do *Kernel* e resulta de uma reescrita do código *IPSec* original do projeto KAME. O Projeto KAME foi um consórcio que disponibilizou uma implementação livre da *stack* IPv6 e *IPSec* para sistemas *Unix BSD*. A alternativa ao NETKEY é o KLIPS, que no entanto não faz parte integrante do *Kernel* e tem de ser compilado de raiz. O seu uso com NAT-T implica ainda a aplicação de um *patch*. A aplicação de regras de filtragem de tráfego (*firewall*) é mais fácil com o KLIPS porque as interfaces por onde flui o tráfego *IPSec* aparecem identificadas com *ipsecx* [48].

Quanto às ferramentas do utilizador, a utilizar com o *IPSec*, existem basicamente duas implementações *open source* muito utilizadas e com provas dadas (ver Figura 4.22): o *openswan*

e o *racoon* (contributo do projeto KAME, por isso também designado por KAME RACOON). A maioria das distribuições Linux ainda usa o *racoon* como base para o utilizador. Mas recentemente a RedHat (versão 6) decidiu adotar o *openswan*. Talvez outras distribuições lhe sigam a tendência, mas para já e para desespero de alguns administradores, tem de se viver com as duas implementações. Existir alternativa é bom, mas no caso do *IPSec* seria talvez desejável usar apenas uma das soluções. Claro que podemos sempre remover o *racoon* e instalar o *openswan* ou vice-versa, não sendo de todo desejável, pela confusão que gera ter os dois no mesmo sistema.

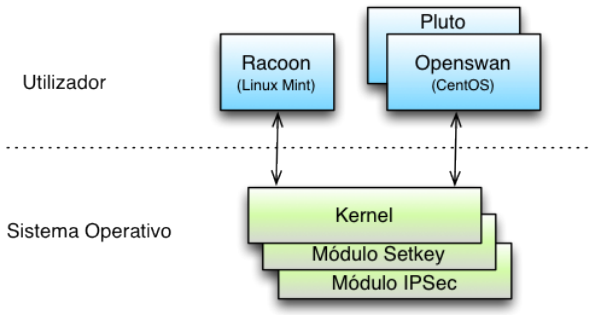


Figura 4.22. Diagrama de blocos dos componentes *IPSec.*

Configuração com RACOON

Esta configuração é o *standard* para os Sistemas Operativos Ubunto/Mint e seus derivados.

A instalação desta ferramenta começa com:

```
# sudo  apt-get install racoon ipsec-tools
```

Depois de executado o comando acima, há todo um conjunto de ficheiros criados. Entre eles o ficheiro *"/etc/default/racoon"*, que define o modo como a configuração é introduzida. Se diretamente no ficheiro *racoon.conf* ou se gerada indiretamente através da ferramenta *racoon-tool*. A variável CONFIG_MODE define o modo esperado, que por omissão será "*direct"*, mas neste caso será através da ferramenta (talvez seja o modo mais aconselhável, por ser mais simples).

```
# vim /etc/default/racoon
```

```
# Which configuration mode shall we use for racoon?
#       Should be either "direct" (edit racoon.conf by hand)
#       or "racoon-tool" (use this tool to do it).
#       Unknown values are treated as if "direct" was given.
CONFIG_MODE="racoon-tool"
# Arguments to pass to racoon (ignored when config mode is racoon-tool)
RACOON_ARGS=""
```

A configuração será gerada pela *racoon-tool*, ou seja, a configuração deve ser introduzida no ficheiro *"/etc/racoon/racoon-tool.conf"*. De cada vez que se iniciar o *racoon*, a ferramenta *"racoon-tool"* pega no ficheiro *"/etc/racoon/racoon-tool.conf"* e criar uma nova configuração, normalmente em *"/var/lib/racoon/racoon.conf"*.

Segue-se um exemplo do ficheiro *"/etc/racoon/racoon-tool.conf"*, onde é configurado um túnel entre a máquina 193.136.9.113 e a máquina 193.136.9.114.

```
global:
        # How to control the syslog level (notify, debug, debug2)
        log: notify
        # Turned this on for extra security, not enabled by default as
        # listen and path become only changeable by restart
        privsep: no

#
# Simple PSK - authentication defaults to pre_shared_key
#
connection(tunel-chave-114):
        src_range: 193.136.9.113/32
        dst_range: 193.136.9.114/32
        src_ip: 193.136.9.113
        dst_ip: 193.136.9.114
        admin_status: enabled
        compression: no
        pfs_group: modp2048
        authentication_algorithm: hmac_sha1
        encryption_algorithm: aes

peer(193.136.9.114):
        peers_identifier: address
        exchange_mode: main
        proposal_check: obey
        verify_cert: on
        passive: off
        verify_identifier: off
        dh_group[0]: modp2048
        hash_algorithm[0]: sha1
        encryption_algorithm[0]: aes
```

A descrição dos diversos campos existentes na configuração do ficheiro *"/etc/racoon/racoon-tool.conf"* podem ser encontradas na Tabela 4.9.

Tabela 4.9. Explicação dos campos usados no *racoon*.[49] [50]

Campo		Definição
connection(tunel-chave-114)		*Tag* da secção "*connection*", onde possui o nome "tunel-chave-114", que indica o nome da conexão
	src_range XXX *src_ip XXX*	Especifica o endereço IP de origem
	dst_range XXX *dst_ip XXX*	Especifica o endereço IP de destino
	admin_status: enabled	Especifica que o uso da conexão está habilitada
	compression: no	Especifica que o tráfego será compactado através da VPN
	pfs_group: modp2048	Define o grupo *Diffie Hellman* e indica a necessidade do uso de PFS (*Perfect Forward Security*)
	authentication_algorithm: hmac_sha1	Define o algoritmo de autenticação, neste caso é o *hmac_sha1*
	encryption_algorithm: aes	Define o algoritmo de encriptação, neste caso é o *aes*
peer(193.136.9.114)		*Tag* da secção "*peer*", onde possui o endereço IP entre parêntesis
	peers_identifier: address	Especifica o identificador do "peer" a ser recebido, neste caso é "address". Se não for definido, o racoon não vai verificar o identificador do "peer", mas uma vez definido, o seu comportamento depende da flag "verify_cert".
	exchange_mode: main	Define o modo de troca da fase 1, que pode ser *main*, *base*, *aggressive*
	proposal_check: obey	Especifica o tempo de vida da ação e o PFS da secção 2 do ficheiro *"/var/lib/racoon/racoon.conf"*. "*Obey*" significa que a resposta vai obedecer ao iniciador a qualquer momento.
	verify_cert: on	Habilita a verificação do certificado ou *PSK (Pre-Shared Key)*
	passive: off	Indica que irá iniciar a negociação
	verify_identifier: off	Define que a verificação do identificador, neste caso o IP, está desativo
	dh_group[0]: modp2048	Define o grupo *Diffie Hellman* corresponde ao número 14
	hash_algorithm[0]: sha1	Define como algoritmo de *hash* o *sha1*, para além deste existem mais algoritmos, como *md5*, *sha256*, *sha384* e o *sha512*
	encryption_algorithm[0]: aes	Define como algoritmo de encriptação o *aes*, para além desde existem mais algoritmos, como o *des*, *3des*, *blowfish* e o *castl128*

Arrancar os serviços, *setkey* e *racoon*:

```
root@voippc113 / # sudo service setkey start
 * Loading IPsec SA/SP database:
                                                    [ OK ]
root@voippc113 / # sudo service racoon start
Loading IPSEC/crypto modules...
IPSEC/crypto modules loaded.
Flushing SAD and SPD...
SAD and SPD flushed.
Loading SAD and SPD...
SAD and SPD loaded.
Configuring racoon...done.
Starting IKE (ISAKMP/Oakley) server: racoon already running - exiting.
```

Depois de iniciados os serviços, será gerado um ficheiro em *"/var/lib/racoon/racoon.conf"*, apresentado a seguinte configuração:

```
#
# Global items
#
path script "/etc/racoon/scripts";
path pre_shared_key "/etc/racoon/psk.txt";
path certificate "/etc/racoon/certs";
log notify;

#
# Connection tunel-chave-114
#
remote 193.136.9.114 {
        proposal {
                encryption_algorithm aes;
                hash_algorithm sha1;
                authentication_method pre_shared_key;
                dh_group modp2048;
        }

        passive off;
        verify_identifier off;
        proposal_check obey;
        verify_cert on;
        peers_identifier address;
        exchange_mode main;
}

sainfo address 193.136.9.113 any address 193.136.9.114 any {
        pfs_group modp2048;
        encryption_algorithm aes;
        authentication_algorithm hmac_sha1;
        compression_algorithm deflate;
}
```

Este ficheiro possui todas as configurações resultantes da configuração do ficheiro *"/etc/racoon/racoon-tool.conf"*. No caso da etiqueta "*remote 193.136.9.114*", que define os parâmetros da fase 1 do IKE, indica que as declarações serão somente aplicadas ao IP 193.136.9.114, caso fosse "*anonymous*" seriam aplicadas a todos os IPs. O "*proposal*" define as sub-instruções para a proposta. No caso da label "*sainfo address 193.136.9.113 any address 193.136.9.114 any",* define os parâmetros da fase 2 do *IKE* (*IPSec-SA*).

De salientar, que as chaves estão definidas no ficheiro "/etc/racoon/psk.txt" como indica no inicio do ficheiro *"/var/lib/racoon/racoon.conf"*. E esse ficheiro possui as seguintes chaves (as chaves foram geradas com o auxilio do comando *"dd if=/dev/random count=16 bs=1 | xxd -ps"):*

```
root@voippc113 ~ # vim /etc/racoon/psk.txt

193.136.9.110   55894d5aacb626d571b9389c7b53b20a
193.136.9.113   55894d5aacb626d571b9389c7b53b20a
193.136.9.114   55894d5aacb626d571b9389c7b53b20a
```

Por fim, são apresentadas as políticas de segurança que, são normalmente introduzidas à parte no ficheiro "*/etc/setkey.conf*":

```
#Configuration on 193.136.9.113 -> 114

# Flush SAD and SPD
 flush;
 spdflush;

 ####################################

 # Security Policies (SP)
 # do 113 -> 114
 spdadd 193.136.9.113 193.136.9.114 any -P out ipsec esp/transport//require;
 spdadd 193.136.9.113 193.136.9.114 ipv6-icmp -P out none;
 # do 114 -> 113
 spdadd 193.136.9.114 193.136.9.113 any -P in ipsec esp/transport//require;
 spdadd 193.136.9.114 193.136.9.113 ipv6-icmp -P in none;
```

Este ficheiro, nas primeiras duas linhas, permite limpar todas as SPD (*Security Policy Database*) e todas as SAD (*Security Association Database*), através dos comandos *"spdflush"* e "*flush",* respectivamente. Nas restantes linhas apenas pretende inserir novas SPDs, sendo "spdadd *X Y any –P out ipsec protocol/mode/src-dst/level"*, onde:

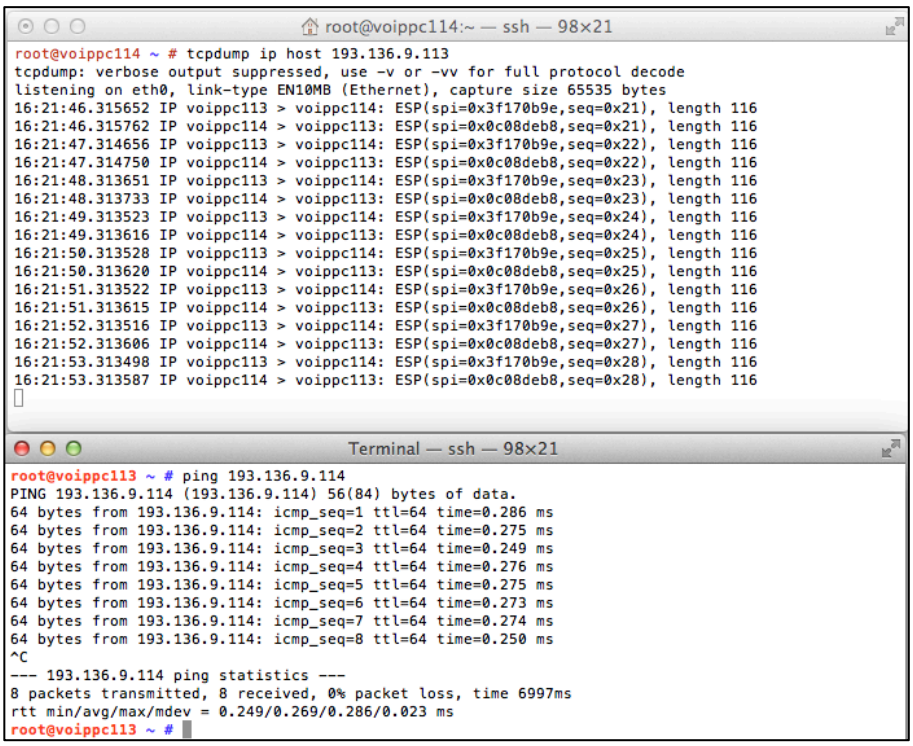
- *"X"* – é o IP de origem;
- *"Y"* – é o IP destino;
- *"any"* – indica que será aplicado sobre qualquer protocolo, podendo ainda usar icmp6 e ip4;
- *"-P out ipsec"* indica as politicas, sendo usado o *IPSec* à saída;
- *"protocol/mode/src-dst/level":*
  - como "*protocol*" é usado o *ESP*, mas ainda poderia usar o *AH* ou o *IPComp* (*IP Payload Compression Protocol*):

- como *"mode"* é usado o *transport*, mas ainda podia usar o *tunnel*, dependendo do modo de transporte defino
- o campo "src-dst" pode ser omitido;
- no *"level"* é usado o *require*, possuindo mais opções, como o *default*, *use* e *unique*.

Compilar as configurações existentes nos ficheiros de configuração:

```
root@voippc113 ~ # sudo racoon -f /etc/racoon/racoon-tool.conf
root@voippc113 ~ # sudo setkey -f /etc/setkey.conf
```

Pode-se testar todas as configurações aqui apresentadas, com o envio *pings* numa máquina e com a captura desses mesmos pacotes através do *tcpdump*, noutra máquina, como é apresentado abaixo:



Para verificar o conteúdo das tabelas de associações (SAD – *Security Association Database*) e de políticas (SPD – *Security Policy Database*) podem-se usar respetivamente os comandos:

```
root@voippc113 ~ # setkey -D
193.136.9.113 193.136.9.114
        esp mode=transport spi=1058474910(0x3f170b9e) reqid=0(0x00000000)
        E: aes-cbc  741f2f06 f988fa85 1db7a5be ca1fc178 8f132c1e 72eb3f2d daeb9e39 ca28dad8
        A: hmac-sha1  cebfb15c 698cca18 7421c190 438003f3 d03491ca
        seq=0x00000000 replay=4 flags=0x00000000 state=mature
        created: Nov 12 16:14:06 2014   current: Nov 12 16:14:41 2014
        diff: 35(s)     hard: 28800(s)  soft: 23040(s)
        last:                           hard: 0(s)      soft: 0(s)
        current: 0(bytes)       hard: 0(bytes)  soft: 0(bytes)
        allocated: 0    hard: 0 soft: 0
        sadb_seq=1 pid=8292 refcnt=0
193.136.9.114 193.136.9.113
        esp mode=transport spi=201907896(0x0c08deb8) reqid=0(0x00000000)
        E: aes-cbc  36881b7d 76193585 31758a69 d8812ee7 9fe371ef f4d7d3ed 27d14fee d7d9e929
        A: hmac-sha1  78f69ff6 6f99d98e c78207cd dadcc975 a465759e
        seq=0x00000000 replay=4 flags=0x00000000 state=mature
        created: Nov 12 16:14:06 2014   current: Nov 12 16:14:41 2014
        diff: 35(s)     hard: 28800(s)  soft: 23040(s)
        last:                           hard: 0(s)      soft: 0(s)
        current: 0(bytes)       hard: 0(bytes)  soft: 0(bytes)
        allocated: 0    hard: 0 soft: 0
        sadb_seq=2 pid=8292 refcnt=0
```

```
root@voippc113 ~ # sudo setkey -DP
193.136.9.114 193.136.9.113 ipv6-icmp
        fwd prio def none
        created: Nov 12 00:31:29 2014  lastused:
        lifetime: 0(s) validtime: 0(s)
        spid=1194 seq=7 pid=20291
        refcnt=1
193.136.9.114 193.136.9.113 ipv6-icmp
        in prio def none
        created: Nov 12 00:31:29 2014  lastused:
        lifetime: 0(s) validtime: 0(s)
        spid=1184 seq=8 pid=20291
        refcnt=1
193.136.9.114[any] 193.136.9.113[any] 255
        fwd prio def ipsec
        esp/transport//require
        created: Nov 12 00:31:29 2014  lastused:
        lifetime: 0(s) validtime: 0(s)
        spid=1178 seq=9 pid=20291
        refcnt=1
193.136.9.114[any] 193.136.9.113[any] 255
        in prio def ipsec
        esp/transport//require
        created: Nov 12 00:31:29 2014  lastused:
        lifetime: 0(s) validtime: 0(s)
        spid=1168 seq=10 pid=20291
        refcnt=1
193.136.9.113 193.136.9.114 ipv6-icmp
        out prio def none
        created: Nov 12 00:31:29 2014  lastused:
        lifetime: 0(s) validtime: 0(s)
        spid=1161 seq=11 pid=20291
        refcnt=1
193.136.9.113[any] 193.136.9.114[any] 255
        out prio def ipsec
        esp/transport//require
        created: Nov 12 00:31:29 2014  lastused:
        lifetime: 0(s) validtime: 0(s)
        spid=1153 seq=12 pid=20291
        refcnt=1
```

Configuração com OPENSWAN

Configuração adoptada para os Sistemas Operativos *RedHat/ CentOS* e seus derivados. No caso das distribuições *RedHat*, que já preferem o *openswan*, a instalação faz-se da seguinte forma:

```
# sudo apt-get install openswan
```

Toda configuração deve ser feita em torno de dois ficheiros, o "*/etc/ipsec.conf*" e o "*/etc/ipsec.secrets*". O primeiro determina as regras e o segundo as chaves pré-partilhadas (se for caso disso). Sendo que a configuração do ficheiro que determina as regras é:

```
root@voippc114 ~ # vim /etc/ipsec.conf

# Manual:     ipsec.conf.5
#
# Please place your own config files in /etc/ipsec.d/ ending in .conf

version 2.0     # conforms to second version of ipsec.conf specification

# basic configuration
config setup
        # Debug-logging controls:  "none" for (almost) none, "all" for lots.
        # klipsdebug=none
        # plutodebug="control parsing"
        # For Red Hat Enterprise Linux and Fedora, leave protostack=netkey
        protostack=netkey
        nat_traversal=yes
        virtual_private=%v4:10.0.0.0/8,%v4:192.168.0.0/16,%v4:172.16.0.0/16
        oe=off
        # Enable this if you see "failed to find any available worker"
        # nhelpers=0

#You may put your configuration (.conf) file in the "/etc/ipsec.d/" and uncomment this.
#include /etc/ipsec.d/*.conf

# configuracao com chave manual... (pre-shared secret)
conn tunel-chave-113
        authby=secret
        auto=start
        type=transport
        # this side of the connection...
        left=193.136.9.114
        leftid=voippc114.gcom.di.uminho.pt
        leftsubnet=193.136.9.114/32
        # the other side of the connection...
        right=193.136.9.113
        rightid=voippc113.gcom.di.uminho.pt
        rightsubnet=193.136.9.113/32
        # algo
        ike=aes256-sha1;modp2048
        phase2=esp
        phase2alg=aes256-sha1;modp2048
```

A descrição dos diversos campos existentes na configuração do ficheiro "*/etc/ipsec.conf*" encontram-se na .Tabela 4.10.

Tabela 4.10. Explicação dos campos do ficheiro *"/etc/ipsec.conf"* [51].

Campo		Definição
config setup	*Tag* da secção "*config*", onde contém informações usadas pelo o software, sempre que este é iniciado	
	nat_traversal=yes	Esta *flag* está ativada por se tratar do IKEv1, e serve para aceitar portas ISAKMP de origem diferentes da udp/500.
	virtual_private=%v4:10.0.0.0/8,%v4: 192.168.0.0/16,%v4:172.16.0.0/16	Define as redes privadas usando uma notação especial
conn tunel-chave-113	*"tunel-chave-113"* é o novo arbitrário da conexão, sendo o local onde é especificado a ligação	
	authby=secret	Define o tipo de autenticação, neste caso indica é para segredos pré-compartilhados
	auto=start	O funcionamento é feito automaticamente na inicialização do IPSec.
	type=transport	Define o tipo de ligação, a opção *"transport"* indica *host-to-host* modo transporte
	left=193.136.9.114 leftid=voippc114.gcom.di.uminho.pt leftsubnet=193.136.9.114/32	Identificação da máquina de origem, como o seu IP e o seu domínio
	right=193.136.9.113 rightid=voippc113.gcom.di.uminho.pt rightsubnet=193.136.9.113/32	Identificação da máquina de destino, como o seu IP e o seu domínio
	ike=aes256-sha1;modp2048	Especificação do algoritmo de encriptação (aes) e de autenticação (sha1)
	phase2=esp	Ativação do protocolo ESP
	phase2alg=aes256-sha1;modp2048	Especificação do algoritmos de autenticação e encriptação, assim como o grupo *Diffie Hellman*

O ficheiro de chaves pré-partilhadas, o "*ipsec.secrets",* (normalmente o *"/etc/ipsec.secrets*" inclui o "*/etc/ipsec.d/ipsec.secrets*") deve possuir a seguinte informação:

```
root@voippc114 ~ # vim /etc/ipsec.d/ipsec.secrets
```

```
193.136.9.114 193.136.9.110 : PSK "55894d5aacb626d571b9389c7b53b20a"
193.136.9.114 193.136.9.113 : PSK "55894d5aacb626d571b9389c7b53b20a"
```

Depois para arrancar o *IPSec* basta fazer:

```
[root@Servidor_Asterisk ~]# sudo service ipsec start
ipsec_setup: Starting Openswan IPsec U2.6.32/K2.6.32-431.el6.i686...
ipsec_setup: /usr/libexec/ipsec/addconn Non-fips mode set in /proc/sys/crypto/fips_enabled
ipsec_setup: /usr/libexec/ipsec/addconn Non-fips mode set in /proc/sys/crypto/fips_enabled
```

Verificar se o serviço está ativo:

```
[root@Servidor_Asterisk etc]# sudo service ipsec status
IPsec running  - pluto pid: 10184
pluto pid 10184
2 tunnels up
some eroutes exist
You have new mail in /var/spool/mail/root
```

Confirmar se as configurações de utilizador estão a "OK":

```
[root@Servidor_Asterisk etc]# ipsec verify
Checking your system to see if IPsec got installed and started correctly:
Version check and ipsec on-path                                 [OK]
Linux Openswan U2.6.32/K2.6.32-431.el6.i686 (netkey)
Checking for IPsec support in kernel                            [OK]
 SAref kernel support                                           [N/A]
 NETKEY:  Testing for disabled ICMP send_redirects              [OK]
NETKEY detected, testing for disabled ICMP accept_redirects     [OK]
Testing against enforced SElinux mode                           [OK]
Checking that pluto is running                                  [OK]
 Pluto listening for IKE on udp 500                             [OK]
 Pluto listening for NAT-T on udp 4500                          [OK]
Checking for 'ip' command                                       [OK]
Checking /bin/sh is not /bin/dash                               [OK]
Checking for 'iptables' command                                 [OK]
Opportunistic Encryption Support                                [DISABLED]
```

Verificar se tudo está correto com o IPsec, através do seguinte *output*:

```
[root@Servidor_Asterisk ~]# ipsec auto --status
000 using kernel interface: netkey
000 interface lo/lo ::1
000 interface eth0/eth0 fd78:2b3:e7b0:4:16da:e9ff:fe9a:c27f
000 interface eth0/eth0 2001:690:2280:20:16da:e9ff:fe9a:c27f
000 interface lo/lo 127.0.0.1
000 interface lo/lo 127.0.0.1
000 interface eth0/eth0 193.136.9.110
000 interface eth0/eth0 193.136.9.110
000 %myid = (none)
000 debug none
000
000 virtual_private (%priv):
000 - allowed 6 subnets: 10.0.0.0/8, 192.168.0.0/16, 172.16.0.0/12, 25.0.0.0/8, fd00::/8, fe80::/10
000 - disallowed 0 subnets:
000 WARNING: Disallowed subnets in virtual_private= is empty. If you have
000          private address space in internal use, it should be excluded!
000
000 algorithm ESP encrypt: id=3, name=ESP_3DES, ivlen=8, keysizemin=192, keysizemax=192
000 algorithm ESP encrypt: id=6, name=ESP_CAST, ivlen=8, keysizemin=128, keysizemax=128
000 algorithm ESP encrypt: id=7, name=ESP_BLOWFISH, ivlen=8, keysizemin=40, keysizemax=448
```

Verificar se o *Kernel* tem os dados corretos:

```
root@Servidor_Asterisk:~# ip xfrm policy
src 193.136.9.110/32 dst 193.136.9.114/32
        dir out priority 2080 ptype main
        tmpl src 0.0.0.0 dst 0.0.0.0
                proto esp reqid 16385 mode transport
src 193.136.9.110/32 dst 193.136.9.113/32
        dir out priority 2080 ptype main
        tmpl src 0.0.0.0 dst 0.0.0.0
                proto esp reqid 16389 mode transport
src 193.136.9.113/32 dst 193.136.9.110/32
        dir in priority 2080 ptype main
        tmpl src 0.0.0.0 dst 0.0.0.0
                proto esp reqid 16389 mode transport
src 193.136.9.114/32 dst 193.136.9.110/32
        dir in priority 2080 ptype main
        tmpl src 0.0.0.0 dst 0.0.0.0
                proto esp reqid 16385 mode transport
src ::/0 dst ::/0
        dir 4 priority 0 ptype main
```

E finalmente, verificar quais as conexões realizadas:

```
root@Servidor_Asterisk:~# ip xfrm state
src 193.136.9.110 dst 193.136.9.114
        proto esp spi 0xf5ad30a4 reqid 16385 mode transport
        replay-window 32
        auth hmac(sha1) 0x787d9e918faf03ed2f6ececaa34e55380666f710
        enc cbc(aes) 0x4d59734fd7dcaf4b4673a55197557bf8a0e9132ee2c6793c5925ba49c3ca28eb
        sel src 193.136.9.110/32 dst 193.136.9.114/32
src 193.136.9.114 dst 193.136.9.110
        proto esp spi 0x4c0f1c81 reqid 16385 mode transport
        replay-window 32
        auth hmac(sha1) 0xeb93a19fdd33b740cc3b0f8b7fdeda49686d5a62
        enc cbc(aes) 0xa7e3486747a5bd2c0d6985b562145eb75ee9c3877e45cbb89dbf15941e8b9966
        sel src 193.136.9.114/32 dst 193.136.9.110/32
src 193.136.9.113 dst 193.136.9.110
```

De igual modo, a configuração pode ser testada com *pings* e *tcpdump* (ver ponto anterior), verificando se está tudo a funcionar corretamente.

4.4 Ferramentas

O uso de ferramentas, para esta dissertação, é indispensável pois necessita-se do *Wireshark* para captura de tráfego, do *CACTI* para recolha de informações de estado das máquinas e do *Dummynet* para criação de um cenário de rede típico. Já os softphones *Yate* e *SFLphone* servem para simular clientes e o mais importante de todos, o Asterisk, serve como intermediário da chamada VoIP. Por fim temos o *SIPp*, que é uma ferramenta de simulação de chamadas, pois permitirá a criação de várias chamadas em simultâneo e a utilização de vários protocolos de transporte, como UDP e TLS.

Algumas das ferramentas foram adotadas por serem já familiares, poupando algum tempo na introdução, como é o caso do *Wireshark* e outras foram adotadas por serem mais simples.

4.4.1 Ferramentas de monitorização

As ferramentas de monitorização visam a recolha de dados durante os testes, para que posteriormente se possam retirar conclusões e apresentar resultados.

4.4.1.1 Wireshark

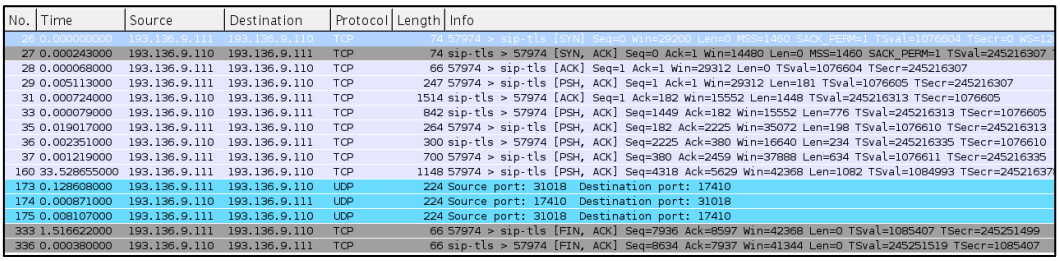
Wireshark [37] é uma ferramenta para análise de tráfego de rede (também conhecido como *sniffer*), que na verdade possui funcionalidades parecidas ao *tcpdump,* mas com uma interface GUI (*Graphical User Interface*). É possível visualizar todos os pacotes capturados em detalhe, podendo estes ser visualizados/organizados segundo os seus protocolos e é possível aplicar diversos filtros, facilitando a análise.

O *Wireshark* é uma ferramenta *open-source*, disponível para os Sistemas Operativos *Windows*, *Linux*, *Solaris*, *FreeBSD*, *NetBSD*, *OpenBSD* e *Mac OS X*. Com esta ferramenta é possível exportar informações para ficheiros de texto, para auxiliar a criação de gráficos e guardar todos os dados recolhidos pela ferramenta num ficheiro *.pcap (packet capture)*.

Funcionalidades TLS (*Transport Layer Security*)

O *Wireshark* possui diversas funcionalidades, algumas delas já foram até apresentadas em figuras anteriores, como é o caso da Figura 3.10, Figura 3.11, Figura 3.12, Figura 4.2, Figura 4.3 e da Figura 4.11.

No caso dos pacotes com o protocolo TLS, quem esteja a visualizar a visualizar o tráfego na rede, irá ver os pacotes de acordo com o que é apresentado na Figura 4.23.



No.	Time	Source	Destination	Protocol	Length	Info
26	0.000000000	193.136.9.111	193.136.9.110	TCP	74	57974 > sip-tls [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=1076604 TSecr=0 WS=12
27	0.000243000	193.136.9.110	193.136.9.111	TCP	74	sip-tls > 57974 [SYN, ACK] Seq=0 Ack=1 Win=14480 Len=0 MSS=1460 SACK_PERM=1 TSval=245216307
28	0.000068000	193.136.9.111	193.136.9.110	TCP	66	57974 > sip-tls [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSval=1076604 TSecr=245216307
29	0.005113000	193.136.9.111	193.136.9.110	TCP	247	57974 > sip-tls [PSH, ACK] Seq=1 Ack=1 Win=29312 Len=181 TSval=1076605 TSecr=245216307
31	0.000724000	193.136.9.110	193.136.9.111	TCP	1514	sip-tls > 57974 [ACK] Seq=1 Ack=182 Win=15552 Len=1448 TSval=245216313 TSecr=1076605
33	0.000079000	193.136.9.110	193.136.9.111	TCP	842	sip-tls > 57974 [PSH, ACK] Seq=1449 Ack=182 Win=15552 Len=776 TSval=245216313 TSecr=1076605
35	0.019017000	193.136.9.111	193.136.9.110	TCP	264	57974 > sip-tls [PSH, ACK] Seq=182 Ack=2225 Win=35072 Len=198 TSval=1076610 TSecr=245216313
36	0.002351000	193.136.9.110	193.136.9.111	TCP	300	sip-tls > 57974 [PSH, ACK] Seq=2225 Ack=380 Win=16640 Len=234 TSval=245216335 TSecr=1076610
37	0.001219000	193.136.9.111	193.136.9.110	TCP	700	57974 > sip-tls [PSH, ACK] Seq=380 Ack=2459 Win=37888 Len=634 TSval=1076611 TSecr=245216335
160	33.528655000	193.136.9.111	193.136.9.110	TCP	1148	57974 > sip-tls [PSH, ACK] Seq=4318 Ack=5629 Win=42368 Len=1082 TSval=1084993 TSecr=24521637
173	0.128608000	193.136.9.111	193.136.9.110	UDP	224	Source port: 31018 Destination port: 17410
174	0.000871000	193.136.9.110	193.136.9.111	UDP	224	Source port: 17410 Destination port: 31018
175	0.008107000	193.136.9.111	193.136.9.110	UDP	224	Source port: 31018 Destination port: 17410
333	1.516622000	193.136.9.111	193.136.9.110	TCP	66	57974 > sip-tls [FIN, ACK] Seq=7936 Ack=8597 Win=42368 Len=0 TSval=1085407 TSecr=245251499
336	0.000380000	193.136.9.110	193.136.9.111	TCP	66	sip-tls > 57974 [FIN, ACK] Seq=8634 Ack=7937 Win=41344 Len=0 TSval=245251519 TSecr=1085407

Figura 4.23. Pacotes do *Wireshark* sem configurações de TLS.

Para visualizar/avaliar devidamente o tráfego TLS, é necessário efetuar os seguintes passos: ir a "Edit → Preferences → Protocols → SSL" e repetir os passos apresentados na Figura 4.24. No exemplo apresentado, apenas está a ser inserido o IP "193.136.9.110" e o certificado do Servidor Asterisk, pois todos os pacotes só necessitam deste certificado. Todos os certificados aqui inseridos são os gerados no ponto 4.3.1.1.

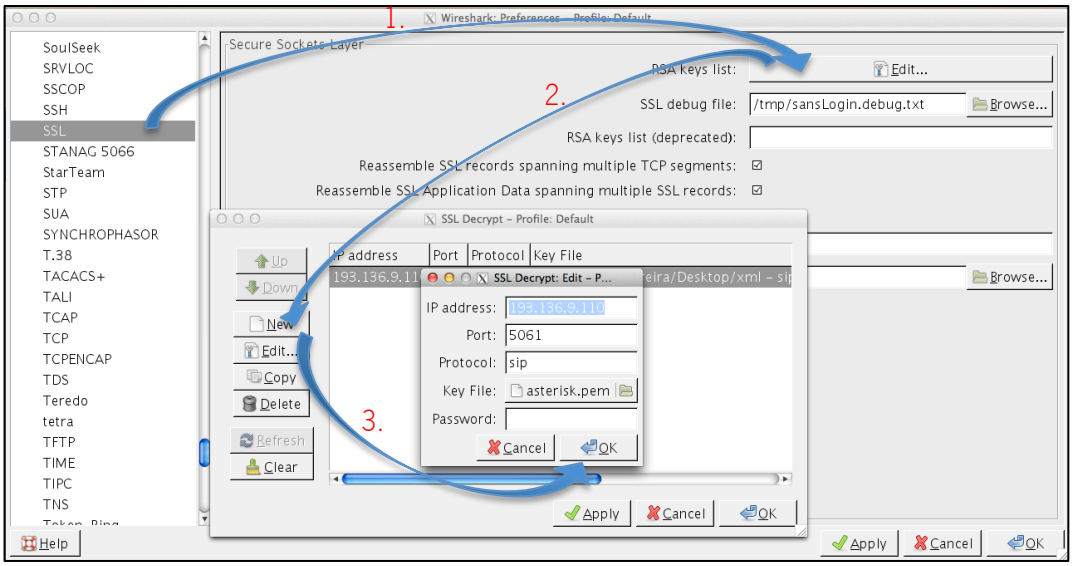


Figura 4.24. Inserção de certificados.

Para visualizar devidamente o tráfego SIP, é necessário ir a "Edit → Preferences → Protocols → SIP" e colocar as configurações segundo a Figura 4.25.

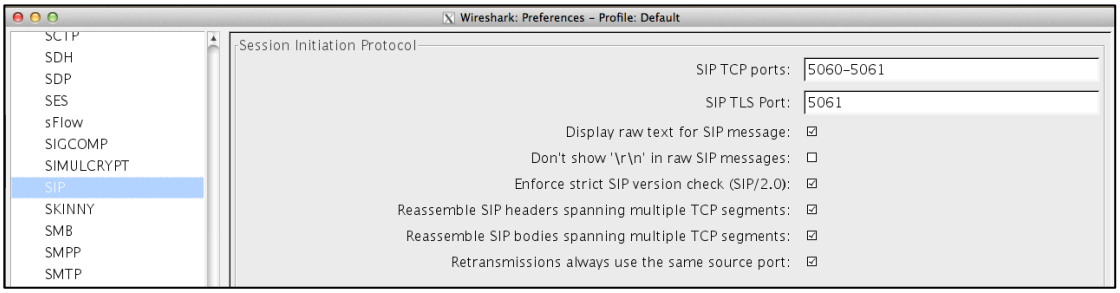


Figura 4.25. Preferências *Wireshark* para o protocolo SIP.

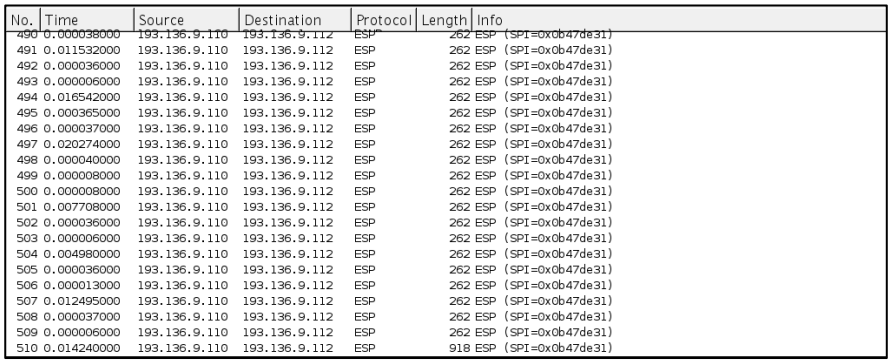
Depois de todas as configurações bem efetuadas, a Figura 4.23 traduz-se para na Figura 4.26, onde é possível comparar pelo número de sequência dos pacotes. Este tipo de "tradução" irá permitir fazer diversas leituras, que serão apresentadas mais à frente. É possível visualizar que os protocolos TCP e UDP, "escondem" segundo a configuração TLS, pacotes que fazem uso do protocolo SIP, TLS e SRTP.

No.	Time	Source	Destination	Protocol	Length	Info
26	0.000000000	193.136.9.111	193.136.9.110	TCP	74	57974 > sip-tls [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=1076604 TSecr=0 WS=1
27	0.000243000	193.136.9.110	193.136.9.111	TCP	74	sip-tls > 57974 [SYN, ACK] Seq=0 Ack=1 Win=14480 Len=0 MSS=1460 SACK_PERM=1 TSval=245216307
28	0.000068000	193.136.9.111	193.136.9.110	TCP	66	57974 > sip-tls [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSval=1076604 TSecr=245216307
29	0.005113000	193.136.9.111	193.136.9.110	TLSv1	247	Client Hello
31	0.000724000	193.136.9.110	193.136.9.111	TLSv1	1514	Server Hello
33	0.000079000	193.136.9.110	193.136.9.111	TLSv1	842	Certificate, Server Hello Done
35	0.019017000	193.136.9.111	193.136.9.110	TLSv1	264	Client Key Exchange, Change Cipher Spec, Finished
36	0.002351000	193.136.9.110	193.136.9.111	TLSv1	300	New Session Ticket, Change Cipher Spec, Finished
37	0.001219000	193.136.9.111	193.136.9.110	SIP	700	Request: REGISTER sips:193.136.9.110:5061;transport=TLS
160	33.528655000	193.136.9.111	193.136.9.110	SIP/SDP	1148	Request: INVITE sips:600@193.136.9.110:5061;transport=TLS, with session description
173	0.128608000	193.136.9.111	193.136.9.110	SRTP	224	PT=ITU-T G.711 PCMU, SSRC=0xBF1DA10C, Seq=4356, Time=2583030314
174	0.000871000	193.136.9.110	193.136.9.111	SRTP	224	PT=ITU-T G.711 PCMU, SSRC=0x29B0D7F7, Seq=15157, Time=160, Mark
175	0.008107000	193.136.9.111	193.136.9.110	SRTP	224	PT=ITU-T G.711 PCMU, SSRC=0xBF1DA10C, Seq=4357, Time=2583030474
333	1.516622000	193.136.9.111	193.136.9.110	TCP	66	57974 > sip-tls [FIN, ACK] Seq=7936 Ack=8597 Win=42368 Len=0 TSval=1085407 TSecr=245251499
336	0.000380000	193.136.9.110	193.136.9.111	TCP	66	sip-tls > 57974 [FIN, ACK] Seq=8634 Ack=7937 Win=41344 Len=0 TSval=245251519 TSecr=1085407

Figura 4.26. Pacotes do *Wireshark* com configurações de TLS.

Funcionalidades IPSec (*IP Security*)

No caso dos pacotes com o protocolo IPSec, quem esteja a visualizar o tráfego na rede, irá ver os pacotes da forma como são apresentados na Figura 4.27.



No.	Time	Source	Destination	Protocol	Length	Info
490	0.000038000	193.136.9.110	193.136.9.112	ESP	262	ESP (SPI=0x0b47de31)
491	0.011532000	193.136.9.110	193.136.9.112	ESP	262	ESP (SPI=0x0b47de31)
492	0.000036000	193.136.9.110	193.136.9.112	ESP	262	ESP (SPI=0x0b47de31)
493	0.000006000	193.136.9.110	193.136.9.112	ESP	262	ESP (SPI=0x0b47de31)
494	0.016542000	193.136.9.110	193.136.9.112	ESP	262	ESP (SPI=0x0b47de31)
495	0.000365000	193.136.9.110	193.136.9.112	ESP	262	ESP (SPI=0x0b47de31)
496	0.000037000	193.136.9.110	193.136.9.112	ESP	262	ESP (SPI=0x0b47de31)
497	0.020274000	193.136.9.110	193.136.9.112	ESP	262	ESP (SPI=0x0b47de31)
498	0.000040000	193.136.9.110	193.136.9.112	ESP	262	ESP (SPI=0x0b47de31)
499	0.000008000	193.136.9.110	193.136.9.112	ESP	262	ESP (SPI=0x0b47de31)
500	0.000008000	193.136.9.110	193.136.9.112	ESP	262	ESP (SPI=0x0b47de31)
501	0.007708000	193.136.9.110	193.136.9.112	ESP	262	ESP (SPI=0x0b47de31)
502	0.000036000	193.136.9.110	193.136.9.112	ESP	262	ESP (SPI=0x0b47de31)
503	0.000006000	193.136.9.110	193.136.9.112	ESP	262	ESP (SPI=0x0b47de31)
504	0.004980000	193.136.9.110	193.136.9.112	ESP	262	ESP (SPI=0x0b47de31)
505	0.000036000	193.136.9.110	193.136.9.112	ESP	262	ESP (SPI=0x0b47de31)
506	0.000013000	193.136.9.110	193.136.9.112	ESP	262	ESP (SPI=0x0b47de31)
507	0.012495000	193.136.9.110	193.136.9.112	ESP	262	ESP (SPI=0x0b47de31)
508	0.000037000	193.136.9.110	193.136.9.112	ESP	262	ESP (SPI=0x0b47de31)
509	0.000006000	193.136.9.110	193.136.9.112	ESP	262	ESP (SPI=0x0b47de31)
510	0.014240000	193.136.9.110	193.136.9.112	ESP	918	ESP (SPI=0x0b47de31)

Figura 4.27. Pacotes do *Wireshark* sem configurações de IPSec.

Para a configuração do IPSec no Wireshark, é necessário reunir alguns parâmetros da configuração durante a comunicação entre o cliente e o servidor. Parâmetros esses, que são destacados a vermelho nas tabelas de associações (ver Figura 4.28). Que é o SPI (*Security Parameter Index*), os algoritmos e as chaves de encriptação e autenticação, isto varia com o sentido da ligação.

```
root@cliente 1002:/Users/raquelpereira$ setkey -D
193.136.9.112 193.136.9.110
        esp mode=transport spi=2371004195(0x8d52a723) reqid=0(0x00000000)
        E: aes-cbc  7dc1ae27 40d21204 3c9b3bc5 549c1ff2 af04e293 044177c2 43cfeabd ff67188b
        A: hmac-sha1  54396123 a66a840b 5206f980 3c3c2f5e 11e39876
        seq=0x00000004 replay=4 flags=0x00000000 state=mature
        created: Jan 23 18:00:42 2015   current: Jan 23 18:01:18 2015
        diff: 36(s)     hard: 28800(s)  soft: 23040(s)
        last: Jan 23 18:01:01 2015      hard: 0(s)      soft: 0(s)
        current: 544(bytes)     hard: 0(bytes)  soft: 0(bytes)
        allocated: 4    hard: 0 soft: 0
        sadb_seq=1 pid=3664 refcnt=2
193.136.9.110 193.136.9.112
        esp mode=transport spi=189259313(0x0b47de31) reqid=0(0x00000000)
        E: aes-cbc  e45a4650 223e6717 3283e560 fffb8527 2e25e584 e0181c94 2e3e6d39 78339c5b
        A: hmac-sha1  445a7d22 40964ad7 e4b11847 f41cc1ce 10011144
        seq=0x00000004 replay=4 flags=0x00000000 state=mature
        created: Jan 23 18:00:42 2015   current: Jan 23 18:01:18 2015
        diff: 36(s)     hard: 28800(s)  soft: 23040(s)
        last: Jan 23 18:01:01 2015      hard: 0(s)      soft: 0(s)
        current: 336(bytes)     hard: 0(bytes)  soft: 0(bytes)
        allocated: 4    hard: 0 soft: 0
        sadb_seq=0 pid=3664 refcnt=2
```

Figura 4.28. Tabelas de associações (SAD).

Para além dos parâmetros já descritos como necessários para a configuração, também é necessário a chave partilhada entre as duas entidades, como é apresentado na Figura 4.29.

```
root@cliente 1002:/Users/raquelpereira$ vim /etc/racoon/psk.txt

193.136.9.110   55894d5aacb626d571b9389c7b53b20a
193.136.9.112   55894d5aacb626d571b9389c7b53b20a
```

Figura 4.29. Chaves partilhadas no IPSec.

Por fim, o último parâmetro a ser necessário é o *Initiator Cookie*, que pode encontrado nos pacotes ISAKMP (*Internet Security Association and Key Management Protocol*), inicialmente trocados no estabelecimento da ligação IPSec.

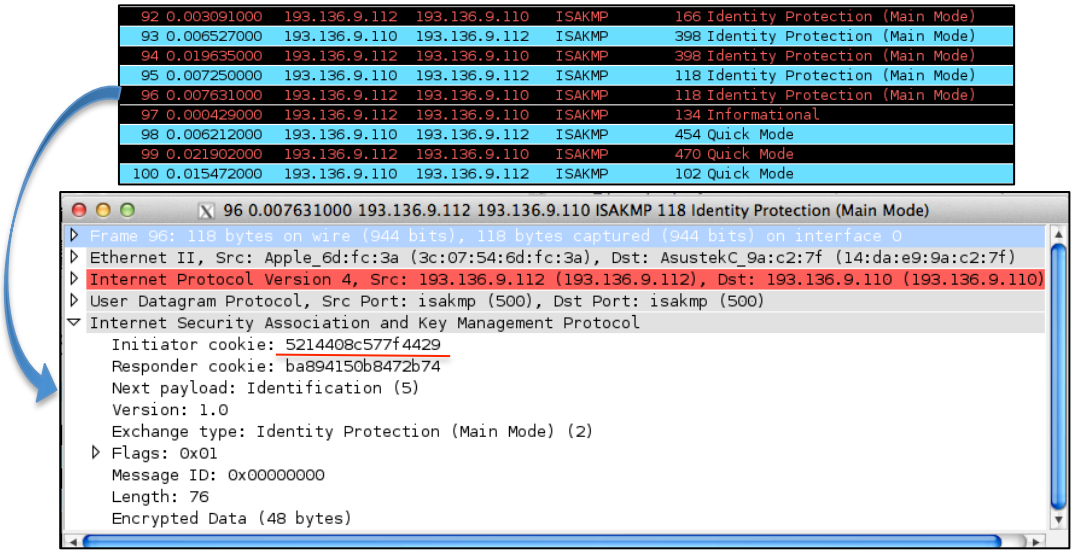


Figura 4.30. Localização do ICOOKIE num pacote ISAKMP.

Após a reunir todos este parâmetros, ir para "Edit → Preferences → Protocols → ISAKMP" e seguir os passos ilustrados na Figura 4.31. A "*INITITATOR COOKIE*" é a apresentada na Figura 4.30 e a "*Encryption Key*" é a chave partilhada apresentada na Figura 4.29.

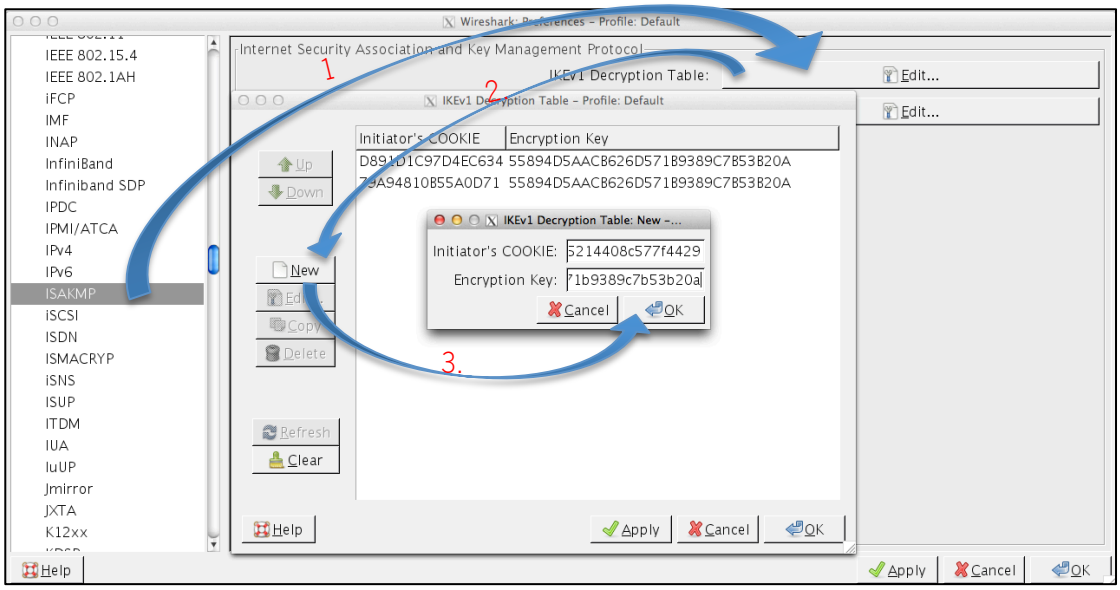


Figura 4.31. Configuração do ISAKMP no *Wireshark*.

A última configuração a fazer é, ir a "Edit → Preferences → Protocols → ESP" e repetir os passos descritos na Figura 4.32, todos os dados necessários estão apresentados na Figura 4.28. Para que seja possível visualizar todos pacotes trocados nos dois sentidos é necessário repetir o processo descrito para os dois sentidos.

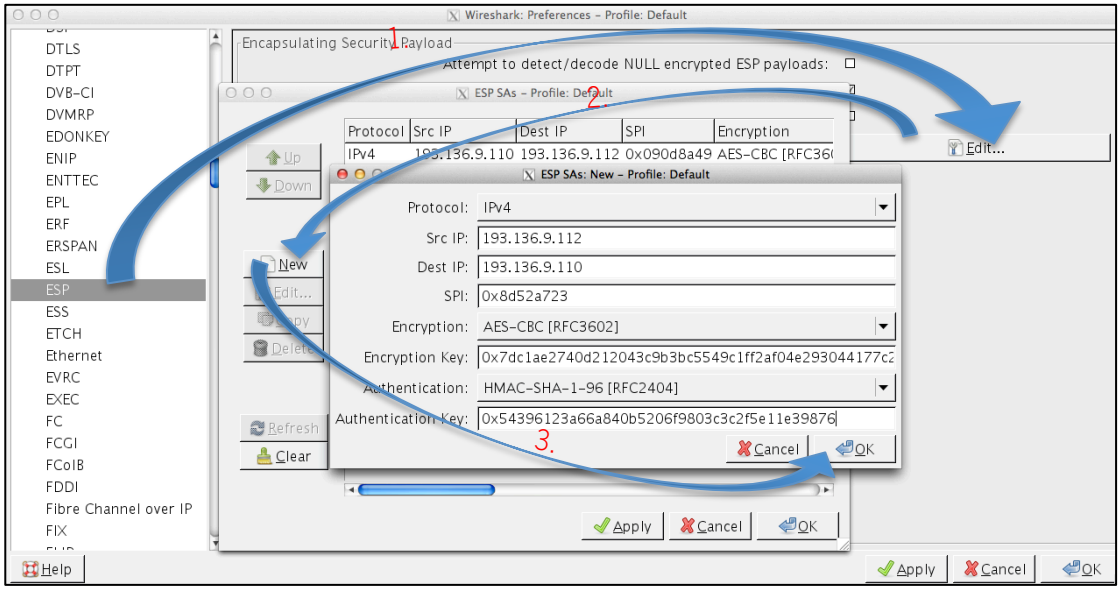
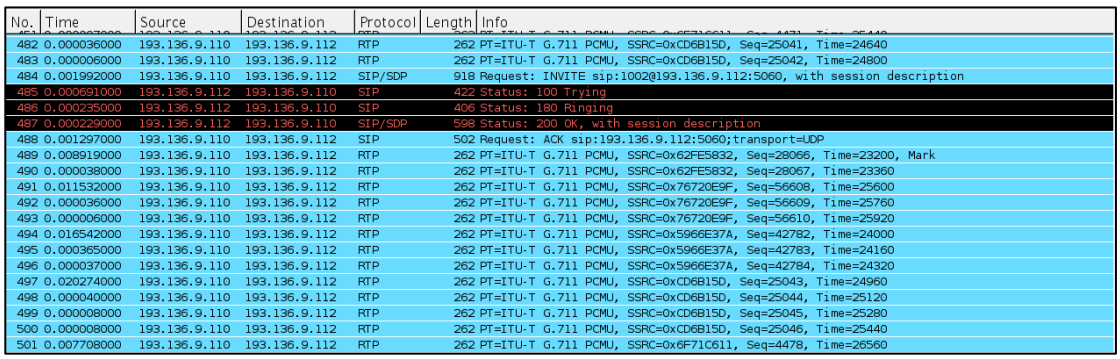


Figura 4.32. Configuração do ESP no *Wireshark*.

Se tudo foi realizado como o descrito, a Figura 4.27 converte-se na Figura 4.33, onde pode ser comparado pelo número de sequência dos pacotes. É possível visualizar que os pacotes com o protocolo ESP, "escondiam" segundo a configuração IPSec, pacotes que fazem uso do protocolo SIP, RTP, etc.



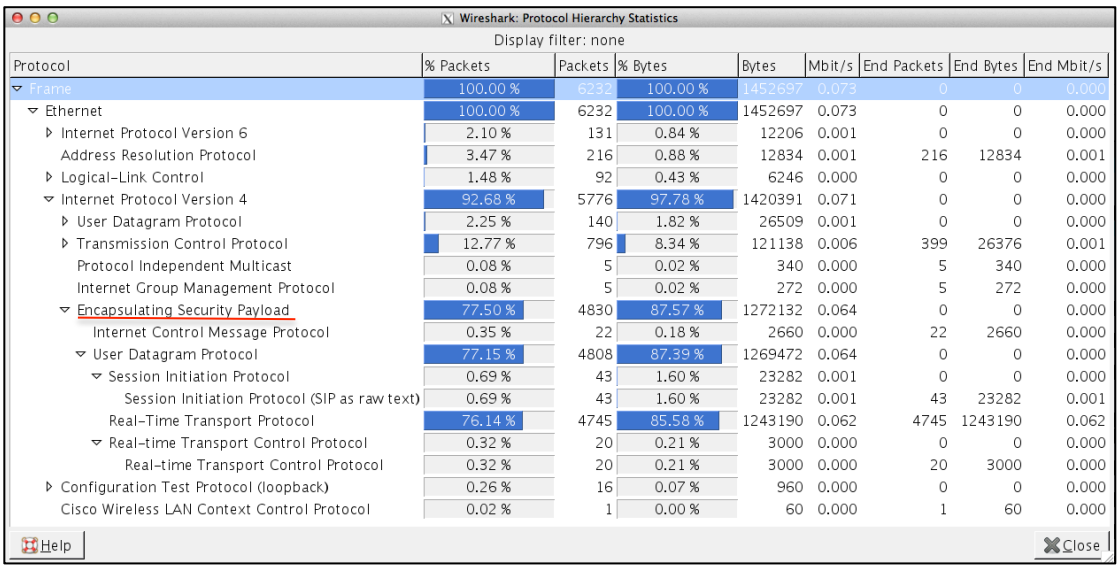
No.	Time	Source	Destination	Protocol	Length	Info
482	0.000036000	193.136.9.110	193.136.9.112	RTP	262	PT=ITU-T G.711 PCMU, SSRC=0xCD6B15D, Seq=25041, Time=24640
483	0.000006000	193.136.9.110	193.136.9.112	RTP	262	PT=ITU-T G.711 PCMU, SSRC=0xCD6B15D, Seq=25042, Time=24800
484	0.001992000	193.136.9.110	193.136.9.112	SIP/SDP	918	Request: INVITE sip:1002@193.136.9.112:5060, with session description
485	0.000691000	193.136.9.112	193.136.9.110	SIP	422	Status: 100 Trying
486	0.000235000	193.136.9.112	193.136.9.110	SIP	406	Status: 180 Ringing
487	0.000229000	193.136.9.112	193.136.9.110	SIP/SDP	598	Status: 200 OK, with session description
488	0.001297000	193.136.9.110	193.136.9.112	SIP	502	Request: ACK sip:193.136.9.112:5060;transport=UDP
489	0.008919000	193.136.9.110	193.136.9.112	RTP	262	PT=ITU-T G.711 PCMU, SSRC=0x62FE5832, Seq=28066, Time=23200, Mark
490	0.000038000	193.136.9.110	193.136.9.112	RTP	262	PT=ITU-T G.711 PCMU, SSRC=0x62FE5832, Seq=28067, Time=23360
491	0.011532000	193.136.9.110	193.136.9.112	RTP	262	PT=ITU-T G.711 PCMU, SSRC=0x76720E9F, Seq=56608, Time=25600
492	0.000036000	193.136.9.110	193.136.9.112	RTP	262	PT=ITU-T G.711 PCMU, SSRC=0x76720E9F, Seq=56609, Time=25760
493	0.000006000	193.136.9.110	193.136.9.112	RTP	262	PT=ITU-T G.711 PCMU, SSRC=0x76720E9F, Seq=56610, Time=25920
494	0.016542000	193.136.9.110	193.136.9.112	RTP	262	PT=ITU-T G.711 PCMU, SSRC=0x5966E37A, Seq=42782, Time=24000
495	0.000365000	193.136.9.110	193.136.9.112	RTP	262	PT=ITU-T G.711 PCMU, SSRC=0x5966E37A, Seq=42783, Time=24160
496	0.000037000	193.136.9.110	193.136.9.112	RTP	262	PT=ITU-T G.711 PCMU, SSRC=0x5966E37A, Seq=42784, Time=24320
497	0.020274000	193.136.9.110	193.136.9.112	RTP	262	PT=ITU-T G.711 PCMU, SSRC=0xCD6B15D, Seq=25043, Time=24960
498	0.000040000	193.136.9.110	193.136.9.112	RTP	262	PT=ITU-T G.711 PCMU, SSRC=0xCD6B15D, Seq=25044, Time=25120
499	0.000008000	193.136.9.110	193.136.9.112	RTP	262	PT=ITU-T G.711 PCMU, SSRC=0xCD6B15D, Seq=25045, Time=25280
500	0.000008000	193.136.9.110	193.136.9.112	RTP	262	PT=ITU-T G.711 PCMU, SSRC=0xCD6B15D, Seq=25046, Time=25440
501	0.007708000	193.136.9.110	193.136.9.112	RTP	262	PT=ITU-T G.711 PCMU, SSRC=0x6F71C611, Seq=4478, Time=26560

Figura 4.33. Pacotes do *Wireshark* com configurações de IPSec.

Leituras para apresentação de resultados

Para a apresentação de resultados, o *Wireshark* possui um conjunto de opções que auxiliam à recolha dos dados. Numa fase de testes a ferramenta é inicializada, para se proceder à captura de todos os pacotes existentes na rede. Após os testes terminarem, sendo uma única captura por teste, são usadas as opções "Telephony → RTP → show All Streams" (ver Figura 4.35), para exportação de valores de QoS, por cada chamada (como *delay*, *jitter*, *packet loss*). Também existe o "Statistics → Protocol Hierarchy" (ver Figura 4.34), que permite retirar o número de pacotes existente em cada protocolo. Esta última opção é útil para comparar, nos diveros mecanismo de segurança, o número de pacotes utilizados para o mesmo número de chamadas.

Na Figura 4.34, é possível visualizar que se tratou de um teste com a configuração de segurança IPSec ativa. Pois mais de 77% dos pacotes capturados são produzidos pelo ESP, sendo que desses 77%, mais de 76% é tráfego RTP.



Protocol	% Packets	Packets	% Bytes	Bytes	Mbit/s	End Packets	End Bytes	End Mbit/s
Frame	100.00 %	6232	100.00 %	1452697	0.073	0	0	0.000
Ethernet	100.00 %	6232	100.00 %	1452697	0.073	0	0	0.000
Internet Protocol Version 6	2.10 %	131	0.84 %	12206	0.001	0	0	0.000
Address Resolution Protocol	3.47 %	216	0.88 %	12834	0.001	216	12834	0.001
Logical-Link Control	1.48 %	92	0.43 %	6246	0.000	0	0	0.000
Internet Protocol Version 4	92.68 %	5776	97.78 %	1420391	0.071	0	0	0.000
User Datagram Protocol	2.25 %	140	1.82 %	26509	0.001	0	0	0.000
Transmission Control Protocol	12.77 %	796	8.34 %	121138	0.006	399	26376	0.001
Protocol Independent Multicast	0.08 %	5	0.02 %	340	0.000	5	340	0.000
Internet Group Management Protocol	0.08 %	5	0.02 %	272	0.000	5	272	0.000
Encapsulating Security Payload	77.50 %	4830	87.57 %	1272132	0.064	0	0	0.000
Internet Control Message Protocol	0.35 %	22	0.18 %	2660	0.000	22	2660	0.000
User Datagram Protocol	77.15 %	4808	87.39 %	1269472	0.064	0	0	0.000
Session Initiation Protocol	0.69 %	43	1.60 %	23282	0.001	0	0	0.000
Session Initiation Protocol (SIP as raw text)	0.69 %	43	1.60 %	23282	0.001	43	23282	0.001
Real-Time Transport Protocol	76.14 %	4745	85.58 %	1243190	0.062	4745	1243190	0.062
Real-time Transport Control Protocol	0.32 %	20	0.21 %	3000	0.000	0	0	0.000
Real-time Transport Control Protocol	0.32 %	20	0.21 %	3000	0.000	20	3000	0.000
Configuration Test Protocol (loopback)	0.26 %	16	0.07 %	960	0.000	0	0	0.000
Cisco Wireless LAN Context Control Protocol	0.02 %	1	0.00 %	60	0.000	1	60	0.000

Figura 4.34. *Protocol Hierarchy.*

A recolha dos dados de QoS, baseia-se em exportar os dados da Figura 4.35, para um ficheiro CSV (*Comma Separated Values*). Onde posteriormente são importados para Excel, sendo por fim criados os gráficos apresentados no Capítulo 5.

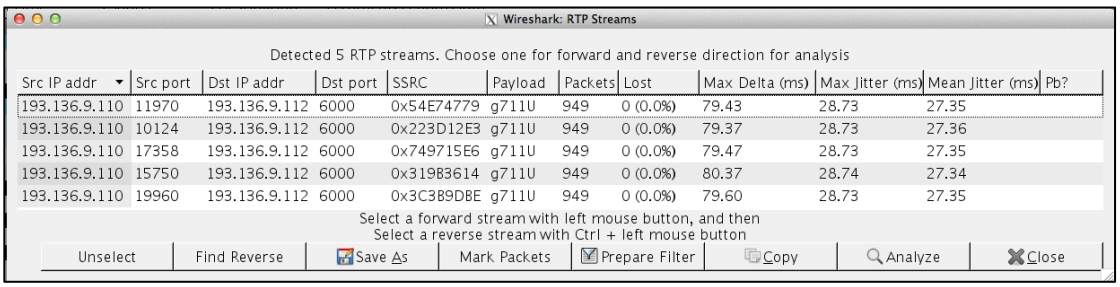


Figura 4.35. *RTP Streams.*

4.4.1.2 CACTI

CACTI [52] é uma interface completa que faz uso do RRDTool (*Round Robin Database Tool*). O RRDTool é um sistema de base de dados para armazenamento de dados relativos ao estado das redes de computadores.

CACTI é uma ferramenta *open-source* sob a licença da GNU (*General Public License*), ferramenta possibilita a criação de diversos gráficos para os diversos recursos que são monitorizados via SNMP. Permitindo assim a monitorização e a gestão de redes simples ou mesmo complexas. O uso do protocolo SNMP permite consultar informações nos diversos componentes da rede, desde que possuam o SNMP devidamente instalado e configurado. A ferramenta proporciona muitas opções para a criação de gráficos que se tornam disponíveis após a criação de dispositivos, ou seja, após a criação de "*Devices*".

SNMP (Simple Network Management Protocol)

SNMP significa, em Português, Protocolo Simples para Gestão de Redes, e como o próprio nome indica é utilizado para a gestão de redes e está descrito no RFC 3584 [53]. Não se trata de um sistema de monitorização, pois não apresenta dados sob a forma de tabelas, gráficos, entre outros, mas auxilia outras ferramentas que o poderão fazer, como é o caso do CACTI.

Este protocolo localiza-se na camada de aplicação, possuindo as operações GET (obtém um dado valor, existindo o GETNEXT para uso interativo que obtém uma sequência de valores ou o GETBULK que obtém dados de um grupo), SET (coloca valores) e TRAP (reporta notificações). Possui as versões SNMPv1, SNMPv2c e SNMPv3, sendo que a versão 2 é uma evolução da

versão 1, apresentando melhorias de performance, segurança, confidencialidade, entre outros. Já a versão 3 oferece privacidade e autenticação segura.

Para a instalação da ferramenta, no caso dos Sistemas Operativos *RedHat/ CentOS* e seus derivados, deve-se executar o seguinte comando:

```
# yum -y install net-snmp net-snmp-utils
```

Para os Sistemas Operativos Ubunto/Mint e seus derivados, executar o seguinte comando:

```
# sudo apt-get install snmp snmpd
```

O ficheiro de configuração, ou seja, o ficheiro "/etc/snmp/snmpd.conf" deve possuir uma configuração similar ao apresentado na Figura 4.36. De notar que a únicas alterações, relativas ao ficheiro original gerado automaticamente, estão sublinhados a vermelho.

```
# First, map the community name "public" into a "security name"

#       sec.name  source          community
com2sec notConfigUser  default       public

# Second, map the security name into a group name:

#       groupName      securityModel securityName
group   notConfigGroup v2c           notConfigUser

# Third, create a view for us to let the group have rights to:

# Make at least  snmpwalk -v 1 localhost -c public system fast again.
#       name           incl/excl     subtree         mask(optional)
view    systemview    included   .1.3.6.1


# Finally, grant the group read-only access to the systemview view.

#       group          context sec.model sec.level prefix read   write  notif
access  notConfigGroup ""      any       noauth    exact  systemview none none

###############################################################################
## System contact information
##
## It is also possible to set the sysContact and sysLocation system
## variables through the snmpd.conf file:
#
syslocation Uminho
syscontact a56929@alunos.uminho.pt
#
###############################################################################
## Logging
##
## We do not want annoying "Connection from UDP: " messages in syslog.
## If the following option is commented out, snmpd will print each incoming
## connection, which can be useful for debugging.
#
dontLogTCPWrappersConnects yes
#
###############################################################################
```

Figura 4.36. Configuração do ficheiro "/etc/snmp/snmpd.conf".

Os *OIDs* (*Objet IDentifier*) apresentados na configuração do ficheiro descrito na Figura 4.36, dizem respeito a objetos que possuem informações, como números *inteiros* ou *strings*. Essas informações/OIDs estão organizados sob a forma de uma árvore hierárquica (ver Figura 4.37). Assim o primeiro OID, de 1.3.6.1, é o "ISO", que se refere aos OIDs ISO (*International Organization for Standardization*) atribuídos. O nível seguinte, é o "org", e este identifica a organização, sendo ela a ISO. O próximo nível, o "dod", identifica o departamento de defesa dos EUA (*United States of America*). Posteriormente, vem o "internet", que é a "crista" de uma subárvore, onde contém todos os dados a ler da máquina. E para-se por aqui, pois é onde se situam todos os pedidos que o CACTI necessita.

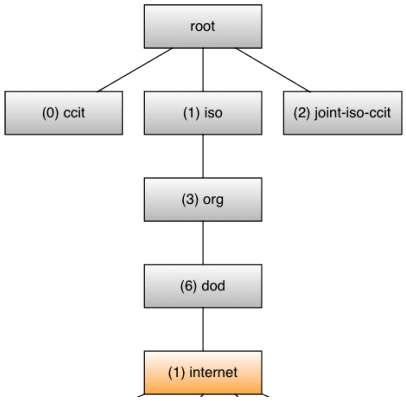


Figura 4.37. MIB de um sistema comum.

Depois de tudo devidamente configurado, procede-se à reiniciação dos serviços com:

```
# service snmpd restart
Stopping snmpd:                                            [  OK  ]
Starting snmpd:                                            [  OK  ]
```

Para testar se tudo está a funcionar corretamente, o seguinte comando deve executar sem problemas:

```
# snmpwalk -v 2c -c public -O e 127.0.0.1
SNMPv2-MIB::sysDescr.0 = STRING: Linux voippc114.gcom.di.uminho.pt 2.6.32-504.el
6.i686 #1 SMP Wed Oct 15 03:02:07 UTC 2014 i686
SNMPv2-MIB::sysObjectID.0 = OID: NET-SNMP-MIB::netSnmpAgentOIDs.10
DISMAN-EVENT-MIB::sysUpTimeInstance = Timeticks: (1635) 0:00:16.35
SNMPv2-MIB::sysContact.0 = STRING: a56929@alunos.uminho.pt
SNMPv2-MIB::sysName.0 = STRING: voippc114.gcom.di.uminho.pt
SNMPv2-MIB::sysLocation.0 = STRING: Uminho
```

Criação de dispositivos

Antes de criar qualquer gráfico, é necessário criar os dispositivos (ver Figura 4.38). Para tal, basta ir a "*Console*" → "*Devices*", depois ir ao canto superior direito e ir a "*add*".

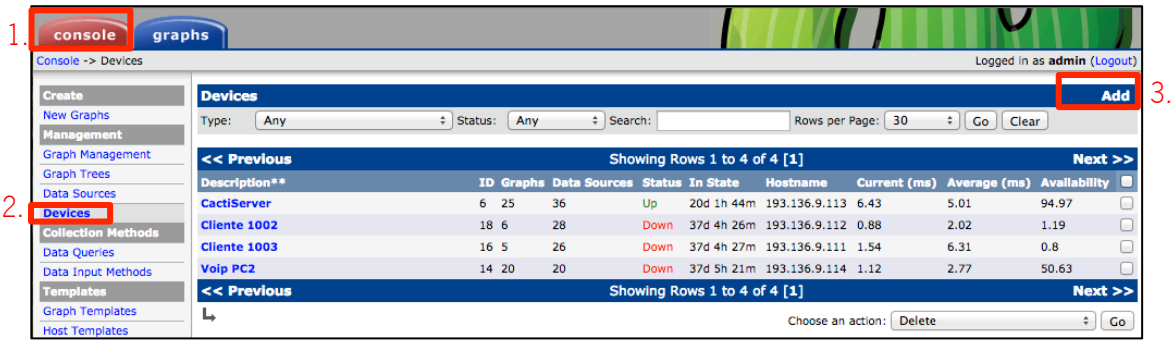


Figura 4.38. Criar dispositivos no CACTI.

Depois de carregar em "*add*", será apresentado um *output* com campos em branco, campos esses onde se irá colocar o nome a dar ao dispositivo, o seu IP, se a disponibilidade é determinada por *ping* ou *SNMP* e por fim qual a "*community*". Neste exemplo os campos foram preenchidos segundo a Figura 4.39 e depois carregou-se em "*create*".

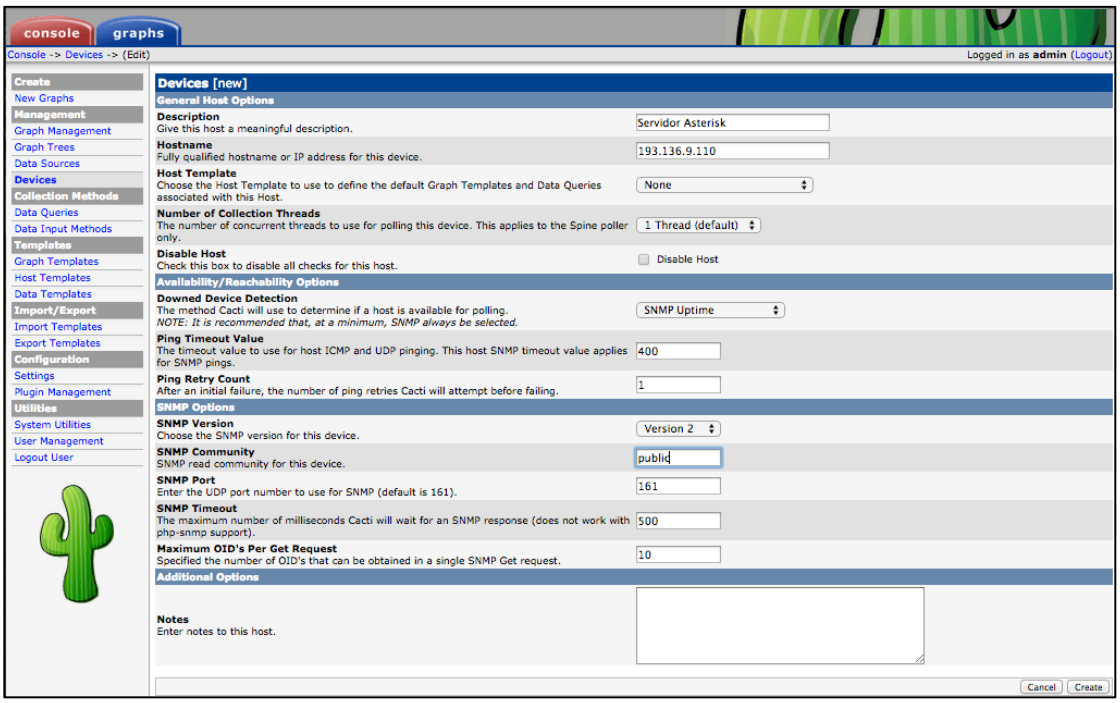


Figura 4.39. Configurar dispositivos no CACTI.

Depois de tudo devidamente preenchido e após se carregar em "*create*", será exibida a seguinte informação, onde indica que tudo funcionou corretamente (ver Figura 4.40).

Figura 4.40. Output de sucesso na criação de dispositivos no CACTI.

Criação de gráficos

Para a criação dos diversos gráficos, que serão apresentados mais à frente, foram necessários diversos passos. Para a realização desta dissertação, achou-se relevante apenas apresentar três gráficos distintos para cada dispositivo da rede. Os gráficos são o de uso de CPU (*Central Processing Unit*), uso de memória interna e por fim o tráfego de rede existente nas interfaces das máquinas.

Procedimentos para a criação do gráfico "uso de CPU":

1. Ir a *"Console"* → no separador *"Management"* clicar em *"Devices"* → Selecionar o *device* pretendido;
2. No fundo da janela no grupo *"Associated Graph Templates"*, em *"add Graph Template"* selecionar a opção *"ucd/net - CPU Usage"* e por fim carregar em *"add"*;
3. Mas a baixo, ou seja, na janela do grupo *"Associated Data Queries"*, em *"add Data Query"* selecionar a opção *"SNMP - Get Processor Information"* e por fim carregar em *"add"*;
4. Carregar no botão *"save"*; para salvar as alterações;
5. Ir ao canto superior direito da janela e selecionar *"Create Graphs for this Host"*;
6. Selecionar os gráficos a criar e carreguar em *"Create"*.

Procedimentos para a criação do gráfico "uso de memória":

1. Ir a *"Console"* → no separador *"Management"* clicar em *"Devices"* → Selecionar o *device* pretendido;
2. No fundo da janela do grupo *"Associated Graph Templates"*, em *"add Graph Template"* selecionar a opção *"ucd/net – Memory Usage"* e por fim carregar em *"add"*;
3. Mas a baixo, ou seja, na janela no grupo *"Associated Data Queries"*, em *"add Data Query"* selecionar a opção *"SNMP – Get Mounted Partitions"* e por fim carregar em *"add"*;
4. Carregar no botão *"save"*; para salvar as alterações;
5. Ir ao canto superior direito da janela e selecionar *"Create Graphs for this Host"*;
6. Selecionar os gráficos a criar e carreguar em *"Create".*

Procedimentos para a criação do gráfico "Tráfego na interface eth0":

1. Ir a *"Console"* → no separador *"Management"* clicar em *"Devices"* → Selecionar o *device* pretendido;
2. No fundo da janela no grupo *"Associated Data Queries"*, em *"add Data Query"* selecionar a opção *"SNMP – Interfaces Statistics"* e por fim carregar em *"add"*;
3. Carregar no botão *"save"*, para salvar as alterações;
4. Ir ao canto superior direito da janela e selecionar *"Create Graphs for this Host"*;
5. Selecionar os gráficos a criar e carregar em *"Create".*

Depois dos gráficos criados, a sua visualização encontra-se em *"graphs"*, no canto superior direito carregar no ultimo separador, ou seja, no *"Preview View"* e selecionar os gráficos, para ver com mais detalhe. No final, ficará algo parecido com o da Figura 4.41.

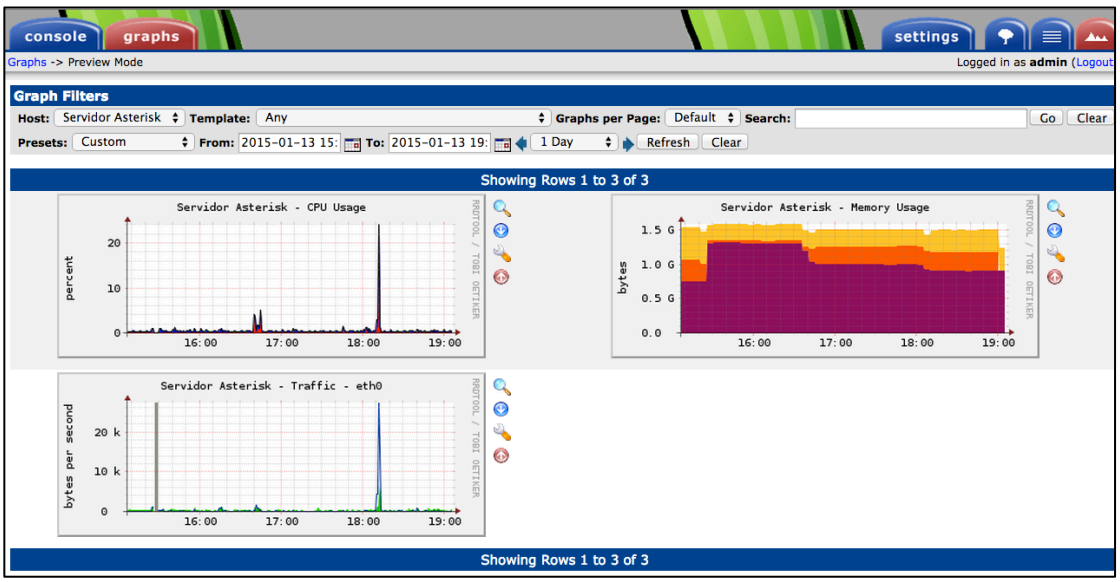


Figura 4.41. *Output* após a criação dos gráficos.

4.4.2 Ferramentas de Condicionamento de rede experimental

O *Dummynet* [33] é uma ferramenta *open-source* que emula parâmetros característicos da rede comum, como atrasos, larguras de banda, perda de pacotes, entre outros. Esta ferramenta funciona nos Sistemas Operativos FreeBSD, OSX, Linux e Windows.

O objetivo desta ferramenta é o de intercetar todo o tráfego segundo a regras definidas, e faz passar esses pacotes por um ou mais "*pipes*" ou "*queues*" com todos os recursos configuráveis, como é o caso da largura de banda, perda de pacotes, tamanho das filas, politicas de programação, entre outros. Os "*pipes*" são canais com largura de banda fixa, as "*queues*" são filas de pacotes associadas ao peso ("*weight*"), a Figura 4.42 ajuda à interpretação do funcionamento do *Dummynet*.

O *scheduler* é apoiado por dois mecanismos, a *queue* (que cria uma ou mais *queues* físicas) e um mecanismo que carrega e configura, em tempo de execução, algoritmos de ligação do "*scheduler*" [54].

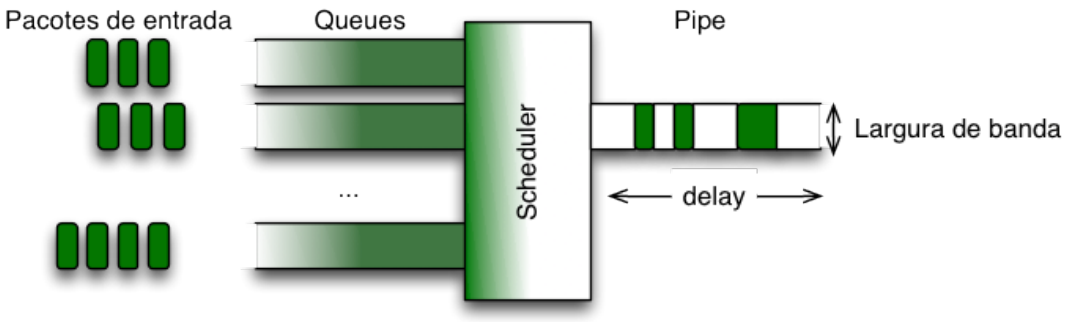


Figura 4.42. Funcionamento do *Dummynet* [Adaptado de [54]].

Resumidamente, o *dummynet* recebe uma série de pacotes, que de acordo com a configuração de cada *queue*, serão encaminhados para a *queue* correspondente, por exemplo, se tiverem como destino o IP 193.136.9.110. Depois dos pacotes estarem nas suas *queues*, o *scheduler* de acordo com o seu tipo de configuração, admitindo que é FIFO, irá pegar em todos os pacotes pela ordem de chegada e entregá-los aos *pipes* correspondentes, dada a configuração. Finalmente, os pacotes irão atravessar os *pipes* correspondentes e estes irão causar os "distúrbios" definidos na configuração do *pipe*, como limitações de largura de banda, *delay* e *packet loss*.

O tráfego é selecionado com precisão usando o *ipfw* (*ipfirewall*, é a *firewall* padrão do *FreeBSD*), sendo possível criar vários *pipes*, enviar tráfego para diferentes *pipes*, entre outros [33].

Para a inserção de regras basta seguir a seguinte sintaxe:

```
# ipfw add <número regra> <ação> <[opções]>
```

Analise o seguinte exemplo:

```
# ipfw add 100 prob 0.2 pipe 10 src-port 80 in
# ipfw add 101 prob 0.7 pipe 20 src-port 80 in
# ipfw add 102 pipe 30 src-port 80 in
```

O exemplo anterior dita que 20% do tráfego HTTP vai para o *pipe* 10, que 56% (que é 70% dos 80% restantes) do tráfego vai para o *pipe* 20 e o restante tráfego vai tudo para o pipe 30.

Para a inserção ou configuração da *queue* basta seguir a seguinte sintaxe:

```
# ipfw queue <núm_queue> sched <número_sched> weight <peso> <[opções]>
```

Analise o seguinte exemplo:

```
# ipfw sched 2 config type FIFO
# ipfw queue 10 config weight 20 sched 2
# ipfw add 100 queue 10 proto udp out
```

O exemplo acima apresenta, na primeira linha é criado um *scheduler* do tipo FIFO (*First In First Out*). Posteriormente é criada uma *queue* número 10, com um peso de 20 associada ao *scheduler* número 2. Por fim, é criada a regra número 100, que tem associada a *queue* número 10 e é aplicada aos pacotes na saída, cujo o protocolo de transporte seja o *UDP*.

Para a configuração simples e básica de um *pipe* basta seguir a seguinte sintaxe:

```
# ipfw pipe <núm_pipe> config <[opções]>
```

Analise o seguinte exemplo:

```
# ipfw pipe 1 config bw 1Mbit/s
# ipfw pipe 2 config delay 100ms
```

O exemplo acima apresenta, que na primeira linha, o *pipe* 1 está a ser configurado com 1Mbit/s de largura de banda e o *pipe* 2 com um *delay* de 100ms.

4.4.2.1 Instalação

A instalação desta ferramenta requereu uma série de passos, passo esses que serão seguidamente explicados.

Para uma correta instalação, foi descarregado o código fonte da ferramenta, através das seguintes linhas de comando:

```
$ sudo mkdir /tmp/dummynet
$ cd /tmp/dummynet/
$ wget http://info.iet.unipi.it/~luigi/doc/20120812-ipfw3.tgz
```

Posteriormente, extraiu-se os ficheiros e compilou-se o código:

```
$ tar -zxvf 20120812-ipfw3.tgz
$ cd /tmp/dummynet/ipfw3-2012
$  sudo make KERNELPATH=/usr/src/linux-headers-3.11.0-12-generic/
```

Depois carregou-se o módulo "*ipfw_mod.ko*":

```
$ cd /tmp/dummynet/ipfw3-2012/kipfw-mod/
$ sudo insmod ipfw_mod.ko
```

Por fim, moveu-se o comando *ipfw* para a diretoria de forma a ficar acessível, independentemente da localização:

```
$ cd /tmp/dummynet/ipfw3-2012/
$ cp ipfw/ipfw /usr/local/sbin
```

Depois da reiniciação do sistema, execute o comando "ipfw -h" e se funcionar, é porque tudo foi feito corretamente.

4.4.2.2 Configuração usada

A configuração a usar, pretende afectar três parâmetros de QoS, que são o *delay*, o *packet loss* e a largura de banda. Para a identificação dos limites dos parâmetros a usar, teve-se em consideração a Tabela 2.11.

Para os parâmetros *delay* e *packet loss*, decidiu-se três intervalos, já para a largura de banda apenas se pensou em limitar para metade (ver Tabela 4.11).

Tabela 4.11. Limites a usar nos parâmetros de QoS.

Largura de banda	Delay (ms)	Packet Loss
50 Mbits/seg	150	0,05 (5%)
	350	0,1 (10%)
	450	0,15 (15%)

Como comandos a usar, para a criação do *pipe*, foram os seguintes:

```
# ipfw add pipe 1 ip from 193.136.9.111 to 193.136.9.110 out
```

Este *pipe* abrange todo o tráfego IP à saída, que vá da máquina 193.136.9.111 para a máquina 193.136.9.110.

Os comandos usados, foram os seguintes:

```
# ipfw pipe 1 config bw 50Mbit/s
# ipfw pipe 1 config delay 150ms (350ms e 450ms)
# ipfw pipe 1 config plr 0,05 (0,1 e 0,15)
```

4.4.3 Servidor VoIP

O Asterisk é um *PBX* (*Private Branch Exchange*) IP completo, que traduzido ficaria Troca de Ramais Privados. Trata-se de um software *open-source* que possibilita a criação de uma central telefónica num simples computador. O *Digium* é a empresa responsável pelo Asterisk, mas atualmente é a comunidade de programadores que colaboram para os avanços da ferramenta [55]. É tanto utilizado em pequenas como em grandes empresas. O seu nome Asterisk, deriva do símbolo "*", que é muito comum no mundo da telefonia.

O Asterisk permite o uso de vários protocolos de comunicação e pode ser usado numa rede PSTN, com o auxílio de interfaces de telefone apropriadas. Suporta aplicações como o correio de voz, correio eletrónico, atendimento automático, conferências e muito mais. É um software com uma arquitetura simples, que suporta os protocolos IAX, SIP e H.323. Os CODECs de vídeo e de áudio, que este suporta, podem ser vistos na Figura 4.43 e na Figura 4.44. Esta informação foi obtida com o auxilio do comando "core show codecs video" e "core show codecs audio" na consola do Asterisk.

```
    NAME DESCRIPTION
-----------------------------------------------------------------------

    g723 (G.723.1)
     gsm (GSM)
    ulaw (G.711 u-law)
    alaw (G.711 A-law)
    g726 (G.726 RFC3551)
   adpcm (ADPCM)
    slin (16 bit Signed Linear PCM)
   lpc10 (LPC10)
    g729 (G.729A)
   speex (SpeeX)
 speex16 (SpeeX 16khz)
    ilbc (iLBC)
g726aal2 (G.726 AAL2)
    g722 (G722)
  slin16 (16 bit Signed Linear PCM (16kHz))
  siren7 (ITU G.722.1 (Siren7, licensed from Polycom))
 siren14 (ITU G.722.1 Annex C, (Siren14, licensed from Polycom))
 testlaw (G.711 test-law)
    g719 (ITU G.719)
 speex32 (SpeeX 32khz)
  slin12 (16 bit Signed Linear PCM (12kHz))
  slin24 (16 bit Signed Linear PCM (24kHz))
  slin32 (16 bit Signed Linear PCM (32kHz))
  slin44 (16 bit Signed Linear PCM (44kHz))
  slin48 (16 bit Signed Linear PCM (48kHz))
  slin96 (16 bit Signed Linear PCM (96kHz))
 slin192 (16 bit Signed Linear PCM (192kHz))
```

Figura 4.43. CODECs de áudio suportados pelo Asterisk.

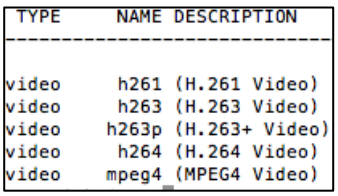


Figura 4.44. CODECs de vídeo suportados pelo Asterisk.

A instalação do software, Asterisk versão 11.13.1 (AsteriskNOW), foi feita como se fosse um Sistemas Operativo [56], mas também pode ser feita por linha de comandos (ver referência [57]). Esta central telefónica, foi instalada numa máquina e tem três diferentes configurações, a configuração básica (ver ponto 4.4.3.1), a configuração com TLS (apresentada no ponto 4.3.1.3) e a configuração com WebRTC (ver ponto 4.4.3.2).

4.4.3.1 Configuração básica

Para a plataforma de testes configurou-se o canal SIP, através da configuração dos ficheiros *sip.conf* e o *extensions.conf*. O primeiro ficheiro é onde são definidos os usuários SIP e no segundo é onde é definido os tipos de chamadas, se individual, se em conferência ou se de teste. Os pacotes SIP, são tipicamente transmitidos através da porta 5060, embora possa variar de acordo com os parâmetros de configuração. As portas para o tráfego de voz/vídeo são dinâmicas e negociadas via protocolo SDP.

Na Figura 4.45, encontra-se a configuração existente no ficheiro *sip.conf*.

```
[general]
context=default
bindport=5060
bindaddr=0.0.0.0

srvlookup=yes
alwaysauthreject=yes
allowguest=no
accept_outofcall_message=yes
auth_message_requests=no
outofcall_message_context=dpma_message_context
;sipdebug=yes
nat=yes

; ***  configuração dos utilizadores *********
[1003]
type=friend
host=dynamic
secret=user1003voipuminho2014
port=5060
context=demo
disallow=all
allow=ulaw
username=1003
canreinvite=yes

[1002]
type=friend
host=dynamic
secret=user1002voipuminho2014
port=5060
context=demo
disallow=all
allow=ulaw
username=1002
canreinvite=yes
```

Figura 4.45. Configuração básica do ficheiro sip.conf.

Na configuração dos clientes SIP, usou-se o CODEC G.711, devido a cumprir os requisitos necessários para uma boa comunicação, como baixo atraso e baixa distorção. Trata-se de um CODEC bastante popular que fornece alta qualidade no sinal de voz, apesar de ser tratar de um CODEC com elevado consumo de largura de banda (64 Kbits/seg) [26].

Cada uma das linhas apresentadas na Figura 4.45, tem uma dada interpretação, a Tabela 4.12 apresenta o significado de cada uma delas.

Tabela 4.12. Designação dos campos do *sip.conf*

Campo	Descrição do campo
[name]	Refere-se ao título da secção. Todas as linhas seguintes aos títulos de secção referem-se a esse mesmo titulo
type	Define o tipo de conexão se é do tipo "user", "peer" ou "friend", sendo que pode efetuar chamadas, receber chamadas ou pode receber e efetuar chamadas, respectivamente.
host	Define um endereço IP fixo, mas caso este seja variável pode ser definido com "*host=dynamic*"
secret	Especifica a palavra-passe a ser usada na autenticação do cliente
port	Porta onde o Asterisk deve esperar pela conexão
context	Encaminhar chamadas para a etiqueta especificada no ficheiro *extensions.conf*
disallow	Desabilita determinados *codecs*
allow	Habilita determinados *codecs*
username	Nome do cliente
canreinvite	Especifica que pode voltar a convidar
nat	Se o cliente está por de trás de uma *firewall* iguala-se a "*yes*", caso contrário a "*no*"

Na Figura 4.46, encontra-se a configuração existe no ficheiro *extensions.conf*. Este ficheiro possui várias linhas, sendo que cada linha tem uma prioridade diferente. Cada linha segue o seguinte formato: "*exten => <num cliente>,<prioridade>, <função>*". A Tabela 4.13 apresenta algumas das funções existentes.

```
[demo]
exten => 1002,1,Dial(SIP/1002,20)
exten => 1003,1,Dial(SIP/1003,20)

exten => 600,1,Answer()
exten => 600,n,Playback(/etc/asterisk/sounds/demo-echotest)
exten => 600,n,Hangup()

exten => 999,1,Answer()
exten => 999,n,Playback(/etc/asterisk/sounds/hello-world)
exten => 999,n,Hangup()
```

Figura 4.46. Configuração básica do ficheiro *extensions.conf*.

Tabela 4.13. Algumas das funções existentes em *extensions.conf*.

Campo	Descrição do campo
Answer()	Atender uma chamada
Dial(SIP/<id_cliente>, <timeout>)	Interliga dois canais de comunicação, o timeout é definido em segundos.
Playback(<caminho>)	Reproduz um ficheiro de áudio localizado na diretoria *<caminho>*
Voicemail(u<nome>)	Enviar a chamada por correio de voz
Hungup()	Finalizar uma chamada

A Figura 4.47, ilustra um cliente a efetuar o registo, sendo esse validado no *sip.conf*. Posteriormente ele efetua uma chamada, "saltando" para o ficheiro *extensions.conf*, onde se "localizará" o cliente a ser contactado, tendo ele, obrigatoriamente, de já estar registado, dando-se assim a chamada.

Figura 4.47. Ilustração da interação entre o clientes e os ficheiros de configuração

4.4.3.2 Configuração com *WebRTC*

A versão 11 do Asterisk, já possui suporte à tecnologia *WebRTC* e também o módulo *res_http_websocket*. Este módulo permite que os programadores de *JavaScript*, desenvolvam soluções onde há interação e comunicação com *WebRTC*. Para além disso, o *WebRTC*, tem como requisito o uso do protocolo *SRTP*, sendo que a biblioteca *libsrtp* já foi introduzida na versão 10 do Asterisk [29].

Para que tudo funcione corretamente e apesar de o Asterisk já possuir diversos módulos, é necessários verificar se os mesmos estão ativados.

Para o correto funcionamento do WebRTC, são necessárias uma série de configurações em diversos ficheiros. Todos os ficheiros estão na diretoria "etc/asterisk" e possuem as configurações apresentadas abaixo (configurações baseadas em [58]).

O ficheiro "http.conf", contém a seguinte configuração:

```
[general]
enabled=yes
bindaddr=0.0.0.0
bindport=8088
```

O ficheiro "sip.conf", contém a seguinte configuração:

```
 [general]
realm=193.136.9.110
udpbindaddr=193.136.9.110
transport=udp

[100]
type=friend
username=1000
host=dynamic
secret=password
encryption=yes
avpf=yes ;habilita o protocolo AVPF (Audio-Visual Profile with Feedback)
icesupport=yes ; habilita o ICE (Interactive Connectivity Establishment )
context=default
directmedia=no ; conexão através do Asterisk
transport=ws ; uso de WebSockets
force_avp=yes ;forçar o uso AVP
dtlsenable=yes ;uso de DTLS (Datagram Transport Layer Security)
```

```
dtlsverify=no ; não verificar certificados DTLS
dtlscertfile=/etc/asterisk/keys/asterisk.pem ; certificado DTLS
dtlsprivatekey=/etc/asterisk/keys/asterisk.pem ; chave privada DTLS
dtlssetup=actpass
```

O *Datagram Transport Layer Security* (DTLS) previne a privacidade das comunicações para os protocolos de datagramas. DTLS permite que aplicativos baseados em datagramas se comuniquem de forma a evitar a espionagem, adulteração ou falsificação de mensagens [59].

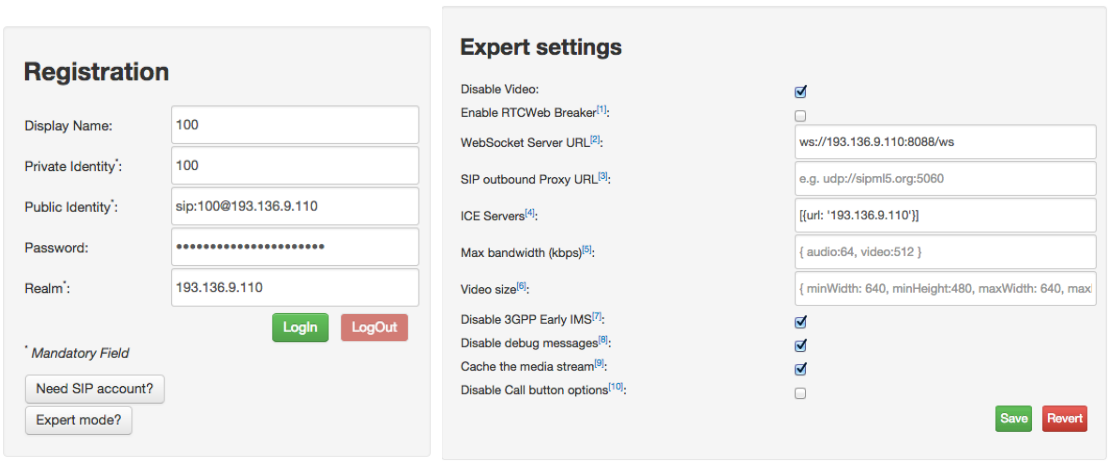
A norma *WebRTC* selecionou *Audio-Visual Profile with Feedback* (AVPF), como o perfil de áudio e vídeo a ser usado para fluxos de comunicação, por exemplo, "*m=audio 49170 RTP/AVPF*". *WebSockets* permitem uma comunicação bidirecional, entre clientes e servidores, sobre TCP.

O ficheiro "extensions.conf", contém a seguinte configuração:

```
[default]
exten => 100,1,Dial(SIP/100)
exten => 101,1,Dial(SIP/101)
```

Depois das configurações devidamente realizadas, deve-se proceder à reiniciação do Asterisk, através do comando "*service asterisk restart*".

Testar com sipML5 (ir a http://193.136.9.110/webrtc/call.htm):



4.4.3.3 Consola Asterisk

Depois de configurados todos os ficheiros necessários, é essencial efetuar a iniciação/reiniação do Asterisk. Esta consola possui uma elevada gama de comandos que acima de tudo auxiliam à verificação de estado da central telefónica e dos clientes.

Para iniciar o *daemon* do Asterisk, basta escrever na linha de comandos comum o comado "*service asterisk start*". Mas para entrar em modo consola, ou seja, no modo "*Command line interface* (*CLI*)", basta fazer "*asterisk –r*", ou "*asterisk –rvvvv*" para entrar em modo de notificação dos eventos, sendo o *output* apresentado na Figura 4.48.

```
[root@Servidor_Asterisk ~]# asterisk -r
Asterisk 11.13.1, Copyright (C) 1999 - 2013 Digium, Inc. and others.
Created by Mark Spencer <markster@digium.com>
Asterisk comes with ABSOLUTELY NO WARRANTY; type 'core show warranty' for details.
This is free software, with components licensed under the GNU General Public
License version 2 and other licenses; you are welcome to redistribute it under
certain conditions. Type 'core show license' for details.
=========================================================================
Connected to Asterisk 11.13.1 currently running on Servidor_Asterisk (pid = 7085)
Servidor_Asterisk*CLI>
```

Figura 4.48. Output após o comando "asterisk -r"

Depois de entrar em modo *CLI*, pode-se executar uma série de comandos. No entanto é possível executar esses mesmos comandos sem a necessidade de abrir a CLI, basta escrever "*asterisk –rx '<comando>*". A descrição de cada um desses comandos, encontram-se apresentados na Tabela 4.14, apesar de existir uma série de comandos, só se irá falar dos mais importantes no contexto desta dissertação.

Tabela 4.14. Comandos mais importantes da consola Asterisk.

Campo	Descrição do campo
core show help	Apresenta todos os comandos disponíveis
core show channels	Mostra as chamadas em curso
core show codecs [| audio | video]	Informa os codecs suportados [todos, áudio, vídeo]
core reload	Recarrega todos os ficheiros
core [restart | stop] now	Reinicia ou desliga o Asterisk imediatamente
core show channels	Apresenta informações das chamadas
core show channel <channel>	Apresenta informações de uma chamada especifica
channel request hangup SIP/100x – xxxxxx	Finaliza a chamada "SIP/100x – xxxxxx"
sip show channelstats	Apresenta estatísticas das chamadas SIP activas, como o "*jitter*" e "*packet loss*"
sip show channel <channel>	Apresenta informações de uma chamada SIP especifica
sip show users	Apresenta o nome, a palavra-passe e entre outros dados, relativos a cada utilizador existente no *sip.conf*
sip show peers	Dos utilizadores existentes no *sip.conf*, este comando indica os que estão ativos e qual o seu IP
sip reload	Recarrega as configurações existentes no *sip.conf*
sip set debug [on | off | ip | peer]	Habilita o modo *debug*
config list	Apresenta todos os ficheiros que foram carregados
config reload <name.conf>	Recarrega todos os ficheiros de configuração
dialplan show	Apresenta os diversos *dialplan* carregados
dialplan reload	Recarrega as configurações existentes no *extensions.conf*
module reload	Recarrega os módulos
module show like <*keyword*>	Verifica que um dado módulo foi carregado
exit	Sair do modo consola do Asterisk

4.5 Clientes VoIP

Os clientes VoIP são pontos terminais de todo o cenário, pois são eles que iniciam e finalizam todo o processo de inicialização e finalização das chamadas.

Um softphone é um software que oferece a funcionalidade de um telefone num dispositivo não-telefone, como um computador pessoal. Tem um aspecto muito similar a um telefone comum, como teclas e um pequeno ecrã, onde aparecem os dígitos que foram ou estão a ser marcados. Eles podem usar protocolos padrão, como o SIP ou o H.323. A sua principal função é a de fazer e receber comunicações de áudio e/ou vídeo.

4.5.1 Yate

O softphone *Yate* [34], escrito na linguagem C++, significa "*Yet Another Telephony Engine"*, que poderá ser traduzido por "Um outro mecanismo de telefonia" é *open-source* e está sob a licença do GNU GPLv2 (GNU *General Public Licence version 2)*.

Segundo [34], este softphone é considerado um motor da rede telefónica de última geração, sendo que atualmente está focado no VoIP e no PSTN. A mesma fonte identifica que o poder desde software reside na capacidade de ser facilmente estendido. *Yate* pode ser usado como um servidor VoIP, um cliente VoIP, um servidor de conferência, um *gateway* de VoIP para PSTN, entre outros. Pode ser instalado nos Sistemas Operativos *Windowns*, Linux e MAC OS. No que diz respeito à segurança este software não suporta nenhum mecanismo de segurança. No entanto tem como protocolos de suporte o SIP, H.323, IAX e o XMPP [60].

Neste caso, a ferramenta é usada como cliente VoIP comum. Depois de uma pequena pesquisa instalou-se o *YateClient* 4.2.0 que possui a interface apresentada na Figura 4.49.

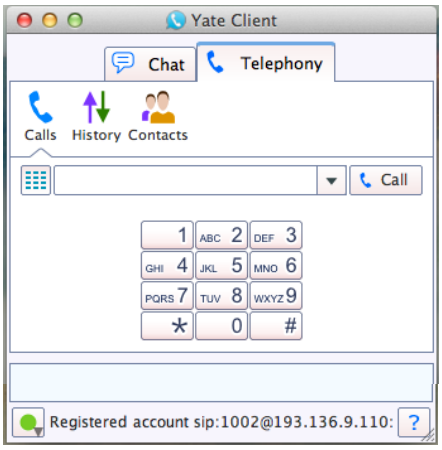


Figura 4.49. Output Yate.

Para o seu correto funcionamento, foram feitas as configurações apresentadas na Figura 4.50, onde é definido o protocolo de sinalização, o nome do utilizador e a palavra-passe para se registar no Servidor Asterisk e por fim o IP e a porta do Servidor Asterisk.

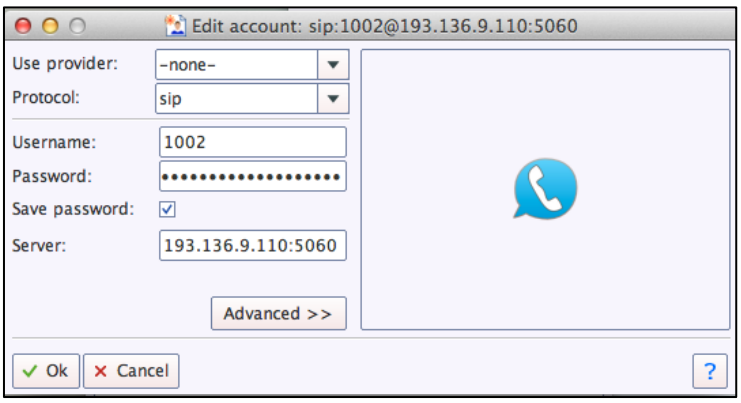


Figura 4.50. Configurações Yate.

4.5.2 SFLphone

O *softphone SFLphone* [35], desenvolvido na linguagem C e C++, é uma ferramenta *open-source* sob a licença *GNU GPLv3 (GNU General Public version 3),* usado nas distribuições Linux. Possui diversas funções de ligação, como número ilimitado de linhas, resposta automática, chamadas de vídeo, chamada em espera, gravação de chamadas, transferência de chamadas,

entre outras. Suporta segurança, nomeadamente SRTP, ZRTP, SDES, TLS e detém como protocolos de apoio o SIP e o IAX2.

Esta ferramenta possui a interface gráfica apresentada na Figura 4.51.

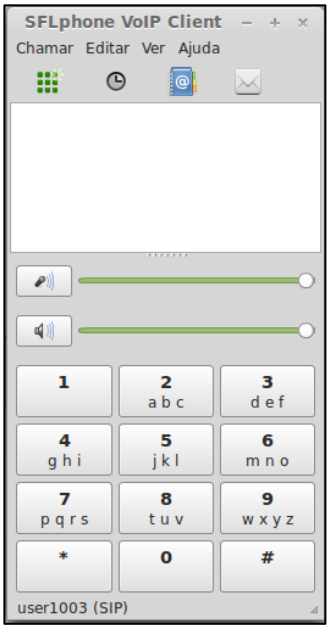


Figura 4.51. Output *SFLphone*.

Para se poder efetuar uma chamada, é necessário, configurar a ferramenta de acordo com o que é apresentado na Figura 4.52.

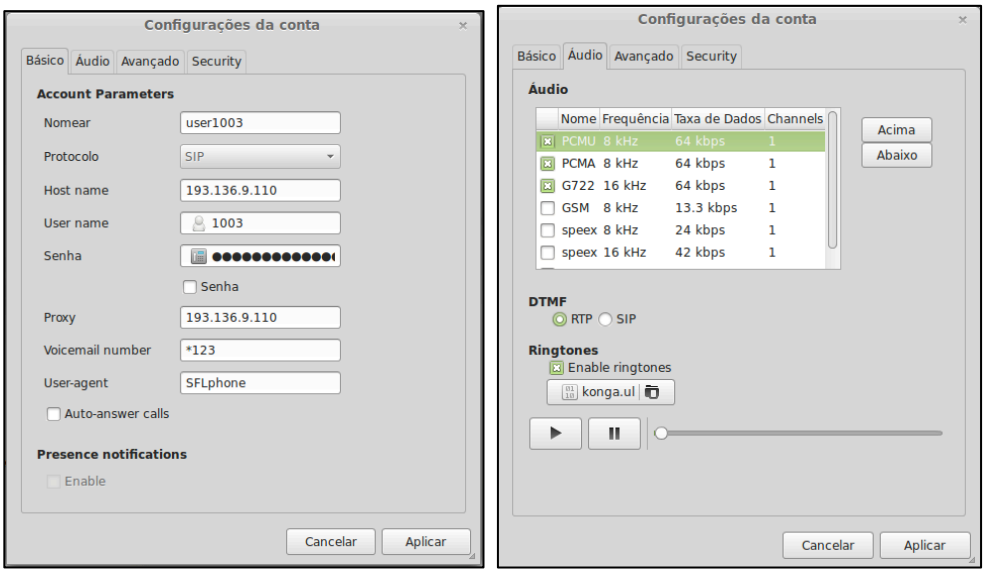


Figura 4.52. Configurações *SFLphone*.

De acordo com a Figura 4.52, a configuração desta ferramenta é semelhante à do Yate. No entanto a configuração adotada no separador "Básico", é, no campo "Nomear" coloca-se um nome à escolha, no "Protocolo" seleciona-se o protocolo pretendido, SIP ou IAX. No *"Host name"* coloca-se o domínio ou IP do servidor, neste caso é o 193.136.9.110. Nos campos *"User name"* e *"Senha"* são colocados o nome e a palavra-passe do utilizador, respectivamente. E por fim, no separador *"Áudio"*, selecionou-se os *Codecs* PCMU, PCMA e G722.

Para chamadas com segurança, para além das configurações apresentadas acima, são necessárias efetuar mais algumas configurações, como as apresentadas na Figura 4.53.

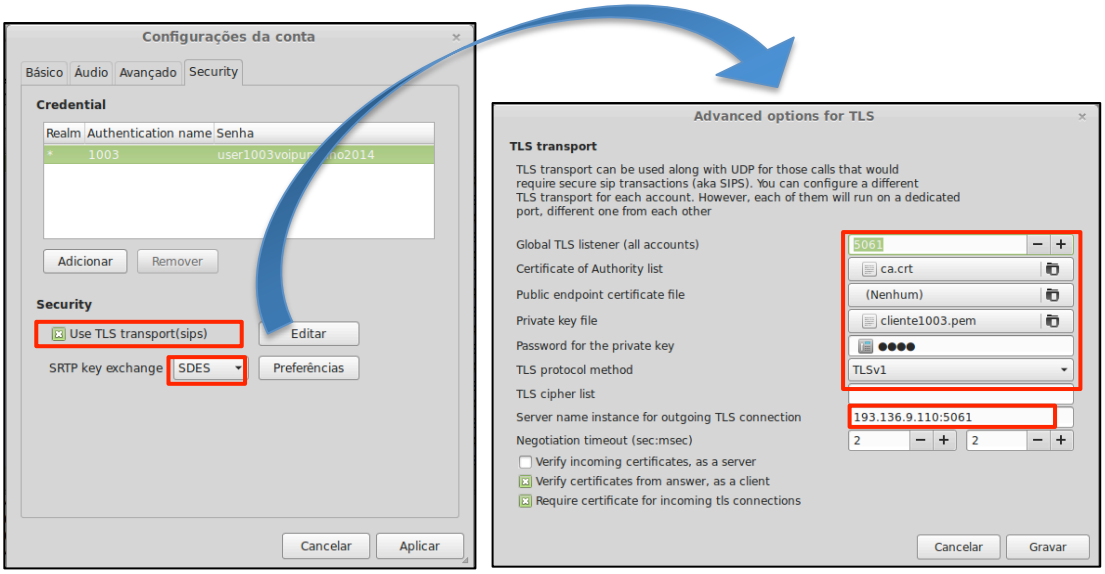


Figura 4.53. Configurações com segurança no *SFLphone*.

Estas configurações baseiam-se em, no separador *"Security"*, habilitar a opção *"Use TLS transport (sips)"* e a *"SDES"* na escolha da *"SRTP key exchange"*. Depois, selecionar a opção *"Editar"*, indo para *"Advanced options for TLS"*. Onde, por ordem, se deve definir a porta *TLS* para estar à escuta, o certificado da Autoridade de certificação e o certificado do cliente (ambos criados no ponto 4.3.1.1). Seguidamente terá de ser inserida a palavra-passe de segurança utilizada na criação dos certificados. Depois será definido o *"TLSv1"*, que define a versão do protocolo *TLS*. Por último, o nome ou IP do servidor *TLS*, assim como a porta.

4.5.3 SIPp

O SIPp [32] é uma ferramenta de emulação de tráfego *open source*, que simula/cria cenários SIP com o auxilio de ficheiros XML. Esta ferramenta possui alguns cenários básicos, como o UAC (*User Agent Client*) e o UAS (*User Agent Server*). Estes dois cenários interligam dois clientes, sem a necessidade de um servidor. Para um cenário mais completo, que comunique com o Servidor Asterisk, será criado um XML, onde existirá um cenário que registe, aguarde e receba chamadas.

No que diz respeito ao transporte, esta ferramenta, possui suporte para os protocolos TCP, UDP, TLS, SCTP (*Stream Control Transport Protocol*) e RTP. A cada execução do SIPp, são apresentados alguns valores estatísticos, como por exemplo, número de chamadas com e sem sucesso, tempo total das chamadas, entre outros (ver Figura 4.54).

```
------------------------------ Scenario Screen -------- [1-9]: Change Screen --
  Port   Total-time  Total-calls  Transport
  5060      50.11 s            1  UDP

  Call limit reached (-m 1), 0.000 s period  0 ms scheduler resolution
  0 calls                               Peak was 1 calls, after 10 s
  0 Running, 2 Paused, 0 Woken up
  0 dead call msg (discarded)
  1 open sockets

                                Messages  Retrans   Timeout   Unexpected-Msg
  ----------> INVITE             1         0         0         0

  <---------- 100                1         0
  <---------- 180                1         0
  <---------- 200                1         0         0
  ----------> ACK          E-RTD1 1         0         0         0

  ----------> INVITE             1         6         0         0

  ----------> BYE                1         3         0         0
  <---------- 200                1         3
  [   4000ms] Pause              1                             0
------------------------------ Test Terminated --------------------------------


------------------------------ Statistics Screen ------- [1-9]: Change Screen --
  Start Time              | 2014-07-10   17:55:24.361865 1405011324.361865
  Last Reset Time         | 2014-07-10   17:56:14.480649 1405011374.480649
  Current Time            | 2014-07-10   17:56:14.480729 1405011374.480729
--------------------------+---------------------------+---------------------------
  Counter Name            | Periodic value            | Cumulative value
--------------------------+---------------------------+---------------------------
  Elapsed Time            | 00:00:00:000000           | 00:00:50:118000
  Call Rate               |    0.000 cps              |    0.020 cps
--------------------------+---------------------------+---------------------------
  Incoming call created   |        0                  |        1
  OutGoing call created   |        0                  |        0
  Total Call created      |                           |        1
  Current Call            |        0                  |
--------------------------+---------------------------+---------------------------
  Successful call         |        0                  |        1
  Failed call             |        0                  |        0
--------------------------+---------------------------+---------------------------
  Response Time 1         | 00:00:00:000000           | 00:00:00:001000
  Call Length             | 00:00:00:000000           | 0--------------------------
---- Test Terminated ----------------------------
```

Figura 4.54. *Output* da ferramenta SIPp.

Durante a execução de uma ou mais chamadas é possível visualizar, a porta usada (5060), o número total de chamadas efetuadas (1), número de chamadas em curso (1), o número de *sockets* abertos (1), o número de retransmissões por mensagem, o protocolo de transporte usado (UDP), entre outros. Relativamente aos vários tipos de mensagens usadas, para o estabelecimento de uma chamada SIP, é apresentado por cada uma delas, o número de mensagens enviadas, assim como o número de vezes que foi retransmitida. No fim do teste terminar, serão apresentados o tempo inicial e o final da chamada, o número de chamadas de saída e de entrada, assim como o número total de chamadas efetuadas. Por fim, são apresentados os números de chamadas efetuadas com e sem sucesso.

4.5.3.1 Instalação

A instalação desta ferramenta é feita, seguindo os seguintes passos, que começa por descarregar o ficheiro de instalação:

```
root@voippc113 / # mkdir sipp
```

```
root@voippc113 / # cd sipp
```

```
root@voippc113 sipp # wget http://sipp.sourceforge.net/snapshots/sipp.2009-07-29.tar.gz
--2014-11-20 19:39:49--  http://sipp.sourceforge.net/snapshots/sipp.2009-07-29.tar.gz
A resolver sipp.sourceforge.net (sipp.sourceforge.net)... 216.34.181.96
A conectar sipp.sourceforge.net (sipp.sourceforge.net)|216.34.181.96|:80... conectado.
Pedido HTTP enviado, a aguardar resposta... 200 OK
Tamanho: 225486 (220K) [application/x-gzip]
Salvando em: "sipp.2009-07-29.tar.gz"

100%[======================================================================>] 225.486      127KB/s   em 1,7s

2014-11-20 19:39:51 (127 KB/s) - "sipp.2009-07-29.tar.gz" salvo [225486/225486]
```

Depois será descompactar o ficheiro de instalação

```
root@voippc113 sipp # tar zxvf sipp.2009-07-29.tar.gz
sipp.svn/actions.cpp
sipp.svn/actions.hpp
sipp.svn/auth.c
sipp.svn/call.cpp
sipp.svn/call.hpp
```

Para compilar o código fonte, é necessário executando o comando *"make pcapplay_ossl"*, que significa que vai instalar o *SIPp* com as opções do *pcap_play* o e o *openssl* ativas. Pois precisamos do *pcap_play* para o envio de pacotes RTP e do *openssl* para os cenários com segurança TLS:

```
root@voippc113 sipp # cd sipp.svn/
root@voippc113 sipp.svn # make pcapplay_ossl
make OSNAME=`uname|sed -e "s/CYGWIN.*/CYGWIN/"` MODELNAME=`uname -m|sed "s/Power Macintosh/ppc/"` OBJ_
S="auth.o sslinit.o sslthreadsafe.o  milenage.o rijndael.o" TLS_LIBS="-lssl -lcrypto" TLS="-D_USE_OPEN
L -DOPENSSL_NO_KRB5"  OBJ_PCAPPLAY="send_packets.o prepare_pcap.o" PCAPPLAY_LIBS="-lpcap `if test -f .
xt; then echo -L./ext/lib; fi;`" PCAPPLAY="-DPCAPPLAY `if test -f ./ext; then echo -I./ext/include; fi
" sipp
make[1]: Entrando no diretório `/programs/sipp/sipp.svn'
gcc   -D__LINUX -pthread  -DSVN_VERSION="\"unknown\"" -D_USE_OPENSSL -DOPENSSL_NO_KRB5 -DPCAPPLAY
I. -I/usr/include/openssl  -c -o auth.o auth.c
gcc   -D__LINUX -pthread  -DSVN_VERSION="\"unknown\"" -D_USE_OPENSSL -DOPENSSL_NO_KRB5 -DPCAPPLAY
I. -I/usr/include/openssl  -c -o sslinit.o sslinit.c
```

Por fim, copiar o ficheiro executável para a diretoria *"/usr/bin", estando assim acessível:*

```
root@voippc113 sipp.svn # sudo cp sipp /usr/bin
```

Para testar se tudo está correto, executar *"sipp -v"*:

```
root@voippc113 sipp.svn # cd /
root@voippc113 / # sipp -v

 SIPp v3.4-early-TLS-SCTP-PCAP-RTPSTREAM, version unknown, built Oct 22 2014, 12:25:57.

 This program is free software; you can redistribute it and/or
 modify it under the terms of the GNU General Public License as
 published by the Free Software Foundation; either version 2 of
 the License, or (at your option) any later version.

 This program is distributed in the hope that it will be useful,
 but WITHOUT ANY WARRANTY; without even the implied warranty of
 MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 GNU General Public License for more details.

 You should have received a copy of the GNU General Public
 License along with this program; if not, write to the
 Free Software Foundation, Inc.,
 59 Temple Place, Suite 330, Boston, MA  02111-1307 USA

 Author: see source files.
```

4.5.3.2 Funcionamento

Sipp é uma ferramenta de teste, que possui diversas opções de execução. O comando base é *"sipp IP_remoto[:porta_remota] [opções]"*, sendo que como opções principais (no contexto desta dissertação) possui as apresentadas na Tabela 4.15 e na Tabela 4.16.

Tabela 4.15. Opções do comando *sipp*.

Campo	Definição
-help	Apresenta os comandos disponíveis
-v	Apresenta a versão da ferramenta instalada
-i	Endereço IP local, este irá substituir os campos: *"Contact"*, *"Via"* e *"From"* do ficheiro XML
-inf	Adiciona um ficheiro csv (*Comma Separated Values)*, a ser usado nas chamadas.
-l	Define o número máximo de chamadas em simultâneo.
-m	Define o número máximo de chamadas a serem efetuadas, assim que esse limite seja atingido o teste é finalizado automaticamente
-p	Define o número da porta local. Se não for definido o sistema escolhe uma livre, de forma aleatória.
-rsa	Define o endereço IP remoto, para o envio das mensagens, no formato *"IP_remoto:porta_remota"*
-s	Define o nome do utilizador remoto, por defeito é "*service*"
-sd	Despeja cenários sip já incluídos no sipp
-sf	Carrega um cenário definido num ficheiro XML, para ajudar à criação destes XML's podem ser usados os cenários disponibilizados pelo "-sd uac", por exemplo: "sipp –sd uas".
-sn	Usa um dos cenários por defeito já incluídos no sipp, como por exemplo, o "uac" para usar como cliente e o "uas" para usar como recetor, entre outros.
-trace_stat	Define o caminho do ficheiro para onde devem ser descarregadas as estatísticas do teste
-fd	Define a frequência de leitura das estatística de cada teste

Tabela 4.16. Opções do comando *sipp* (cont.).

Campo	Definição
-ap	Define a *password* para os desafios de autenticação, por defeito é "password"
-t	Define o modo de transporte: "*u1*" – UDP com um *socket*; "*un*" – UDP com um *socket* por chamada; "*ui*" – UDP com um *socket* por cada endereço IP; "*t1*" – TCP com um *socket*; "*tn*" – TCP com um *socket* por chamada; "*l1*" – TLS com um *socket*; "*ln*" – TLS com um *socket* por chamada; "*c1*" – *u1* + compressão; "*cn*" – *un* + compressão (estes dois últimos funcionam se o *plugin* de compressão estiver carregado, pois o *sipp* não o oferece)
-tls_cert	Define o nome e o caminho para o certificado TLS, por defeito é "cacert.pem"
-tls_key	Define o nome e o caminho para o ficheiro de chave privada TLS, por defeito é "cakey.pem"

4.5.3.3 Ficheiros XML

Os ficheiros XML (*Extensible Markup Language*), criados para a execução dos testes com o *SIPp*, encontram-se nos pontos Apêndice A, Apêndice B, Apêndice C, Apêndice D, Apêndice E, Apêndice F, Apêndice G e Apêndice H. Estes ficheiros pretendem simular a criação e finalização de chamadas, mas para além disso vão permitir o uso do protocolo de segurança *TLS* e o envio de um ficheiro *.pcap*, que contém os pacotes de áudio.

Os arquivos XML, são usados em simultâneo com os arquivos CSV, estes ficheiros possuem uma estrutura, em que a primeira linha indica o tipo de leitura a fazer, ou seja, se é lido de forma sequencial (*SEQUENTIAL*), aleatória (*RANDOM*) ou na forma utilizador (*USER*), as restantes linhas possuem parâmetros importantes para o estabelecimento das chamadas. Cada uma dessas linhas corresponde a uma chamada e em cada linha, o diversos parâmetros são separados por ";", cada um desses campos é acedido por *[field0]* até *[fieldN]*.

De acordo com a imagem já apresentada no ponto 4.1, Figura 4.4, a configuração dos ficheiros XML segue a ordem dessas mensagens, apresentadas por essa figura. Por uma questão de organização, decidiu-se dividir essa mesma imagem em três partes distintas. Em que a primeira parte diz respeito ao registo dos clientes no servidor, a segunda diz respeito ao emissor da chamada e a terceira ao recetor da chamada.

Antes de apresentar os cenários, será apresentado muito rapidamente alguns dos campos necessários para a criação das mensagens SIP. Por exemplo, um pedido de INVITE possui os campos apresentados na Figura 4.55.

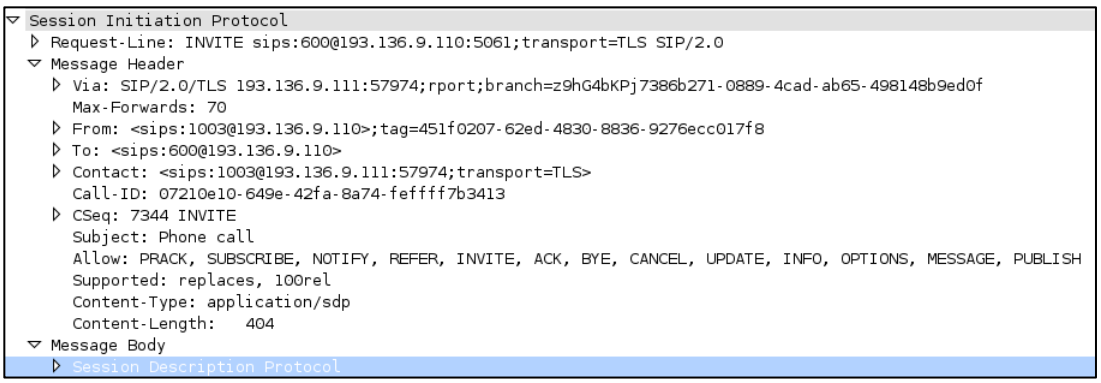


Figura 4.55. Informação SIP contida numa mensagem de INVITE.

Esta mensagem possui campos que são comuns a outras mensagens SIP. No entanto a descrição de cada campo encontra-se na Tabela 4.17.

Tabela 4.17. Explicação dos campos existentes numa mensagem SIP.

Campo	Definição
Via	Apresenta o endereço para onde o pedido deve ir
Max-Forwards	Limita o número de *proxies* ou *gateways* que podem enviar requisições para um servidor
From	Indica quem é o emitente da mensagem
To	Indica quem é o destinatário da mensagem
Contact	Fornece um URI (*Uniform Resource Identifier*), o significado desse URI depende do pedido ou da resposta. Pode possuir uma identificação do cliente
Call-ID	Identificador único por casa sessão
CSeq	*Command Sequence*. Atribui uma identificação única por cada chamada, diferenciando das outras mensagens
Content-Type	Descrição da mensagem, pode indicar se está a usar um softphone ou o SIPp
Content-length	Define o tamanho do corpo da mensagem

Após a descrição dos principais campos, é necessário perceber quais os comandos a usar nos ficheiros XML, ou seja, de que forma se pode aguardar ou enviar uma mensagem especifica, como por exemplo uma mensagem de INVITE, para tal basta analisar a Tabela 4.18.

Tabela 4.18. Principais comandos dos ficheiros XML.

Campo	Atributo	Definição
<send>	retrans	Serve para ajustar o temporizador. Por exemplo, <send retrans="500">, define o temporizar para 500ms
	clrf	Adiciona uma linha em branco no output do Sipp
<recv>	response	Especifica a mensagem SIP esperada. Exemplo: <recv response="100">, define que é esperada a mensagem 100, ou seja, a de *Trying*,
	request	Aguarda por mensagem de pedidos SIP. Exemplo: <recv request="INVITE">, indica que vai esperar pela mensagem de INVITE.
	optional	Especifica que mensagens são opcionais. Exemplo: <recv response="180" optional="true"> opcionais, indica que a receção da mensagem *Ringing* não é obrigatória.
	rrs	Significa *Record Route Set* e serve para que o SIP guarde o cabeçalho de registo da rota anterior. Exemplo: <recv response="100" rrs="true">, armazena a rota recebida, permitindo que esta seja acedida pela palavra chave "[routes]".
	timeout	Especifica um dado tempo de espera antes de cancelar a chamada.
	clrf	Adiciona uma linha em branco no output do Sipp
<pause>	milliseconds	Causa um dado atraso. Exemplo <pause milliseconds="20"/>, será inserido um atraso de 20s antes de iniciar a próxima tarefa

Posteriormente, existem alguns campos que são geridos pelo próprio SIPp e outros pelo utilizador através dos parâmetros passados pela linha de comandos, alguns desses comando ou palavras chave encontram-se descritos na Tabela 4.19.

Tabela 4.19. Palavras chaves existentes nos ficheiros XML.

Campo	Definição
[remote_ip]	Define IP remoto, passado na linha de comandos. Exemplo: sipp 193.136.9.110
[remote_port]	Define a porta remota, exemplo: sipp 193.136.9.110:5060
[transport]	Varia com o que for definido em –t, pode ser UDP (u1), TLS (l1) ou TCP (t1)
[len]	Calcula o comprimento da mensagem SIP, para definir em "Content-Lengh"
[call_number]	Inicia em 1 e é incrementado a cada mensagem
[cseq]	Valor automático, iniciado a 1
[call_id]	Identifica uma chamada e é gerado um novo a cada nova chamada Sipp

Cenário de registo

A Figura 4.56, apresenta as mensagens necessárias para se efetuar o registo de um cliente, sendo que o XML usado encontra-se no Apêndice B.

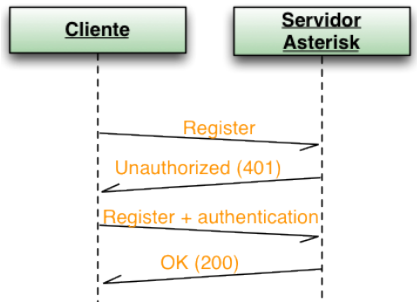


Figura 4.56. Registo dos clientes no Servidor Asterisk.

Para a execução do ficheiro XML, ou seja, do cenário de registo, basta executar a seguinte linha de comando:

```
$ sipp 193.136.9.110:5060 -sf registo.xml -inf 1002.csv -i 193.136.9.112 -p 5060 -m 1 -l 1
```

A linha de comandos apresentada destina-se a informar, que o IP do Servidor Asterisk é o 193.136.9.110 e a porta é a 5060. Indica que será usado um cenário XML, com o nome registo.xml (que se encontra na diretoria atual), e serão usados os parâmetros colocados no ficheiro 1002.csv. As opções seguintes indicam que será executado no máximo uma chamada, o IP local é o 193.136.9.112 e a porta local onde estará à escuta é a 5060.

A configuração no ficheiro *registo.xml* segue a ordem das mensagens apresentadas na Figura 4.56. Mensagens essas que, depois dos testes com os *Softphones* e com a ajuda do *Wireshark*, foram analisadas e replicadas para o ficheiro XML. Por exemplo, a Figura 4.57 apresenta a informação SIP necessária para a mensagem de "*REGISTER*".

Posteriormente, e por uma questão de lógica, implementou um cenário de *logout*. Cenário esse que é uma replicação do registo normal, apenas o campo "Expires" leva o valor 0, criando assim o *logout* do cliente (ver Apêndice E).

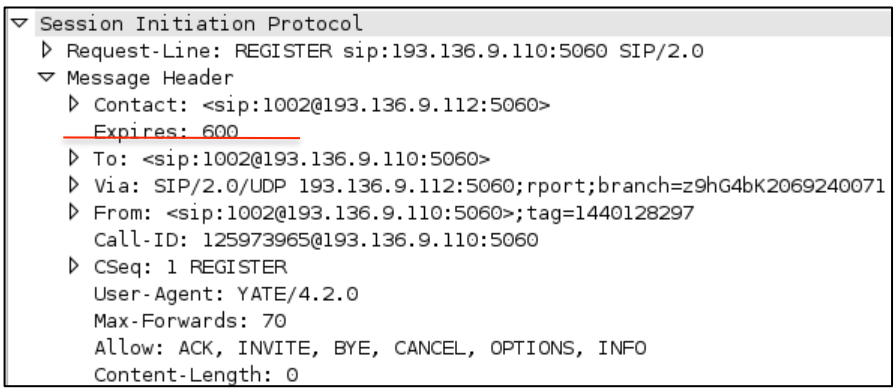


Figura 4.57. Informação SIP necessária para a mensagem de "REGISTER"

Na Figura 4.58 é apresentada uma replicação da Figura 4.57, trata-se de uma pequena parte da configuração existente no ficheiro registo.xml. É também explicado de que forma as variáveis existentes no ficheiro 1002.csv são substituídas no XML correspondente (ver na Tabela 4.15 o comando "*–inf*"), ou seja, em cada *[fieldN]* do *XML* é substituído pelo valor do ficheiro *CSV* na posição *N*.

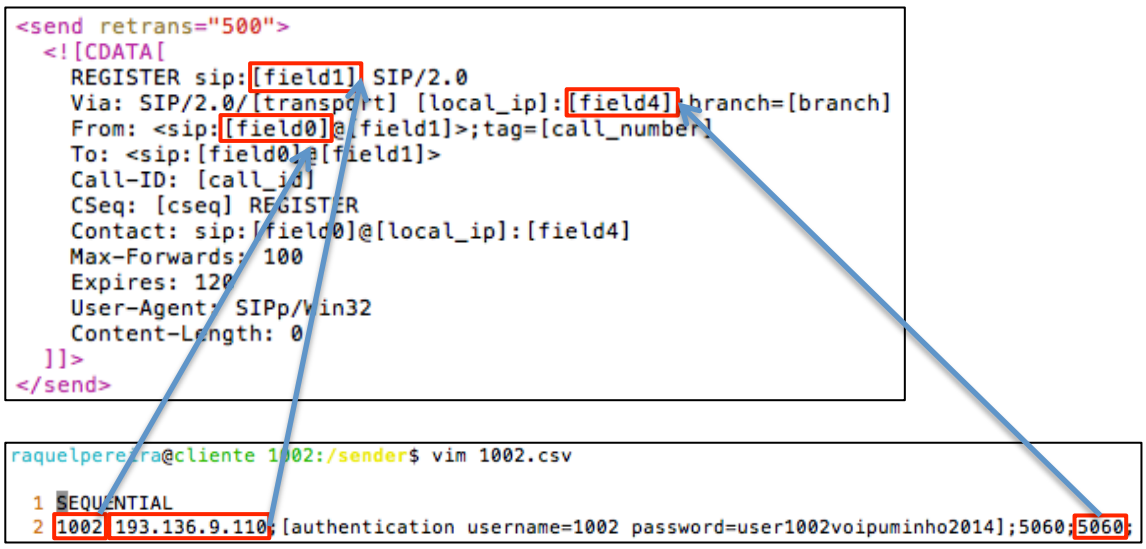


Figura 4.58. Exemplificação da interação ente o file *register.xml* e o *1002.csv*

Cenário de emissor da chamada

A Figura 4.59 apresenta as mensagens necessárias para efetuar uma chamada para um dado cliente, sendo que o XML usado encontra-se no Apêndice D.

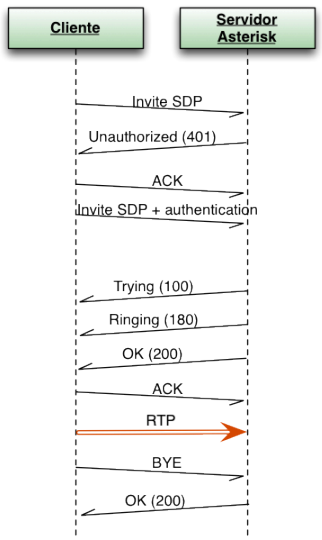


Figura 4.59. Envio de pedidos por parte dos clientes, para iniciação de uma chamada.

Para a execução do cenário de realização de chamadas para um dado cliente, na script desenvolvida no seio desta dissertação (ver Apêndice A) ficaria:

```
$ ./voipp –e 1002 UDP 1
```

Na linha de comandos seria:

```
$ sipp 193.136.9.110:5060 -sf emissor.xml -inf 1002.csv -i 193.136.9.112 -p 5060 -m 1 -trace_stat -fd 2
```

A linha de comandos apresentada acima, tem a mesma interpretação que a linha de comandos para o cenário de registo, apenas com a diferença nos ficheiros usados. Pois o cenário XML (*emissor.xml*) possui mensagens SIP diferentes.

Seguindo a mesma ordem de explicação, na Figura 4.60 encontra-se a informação SIP necessária para o envio, por exemplo, da mensagem de "*INVITE with authentication*".

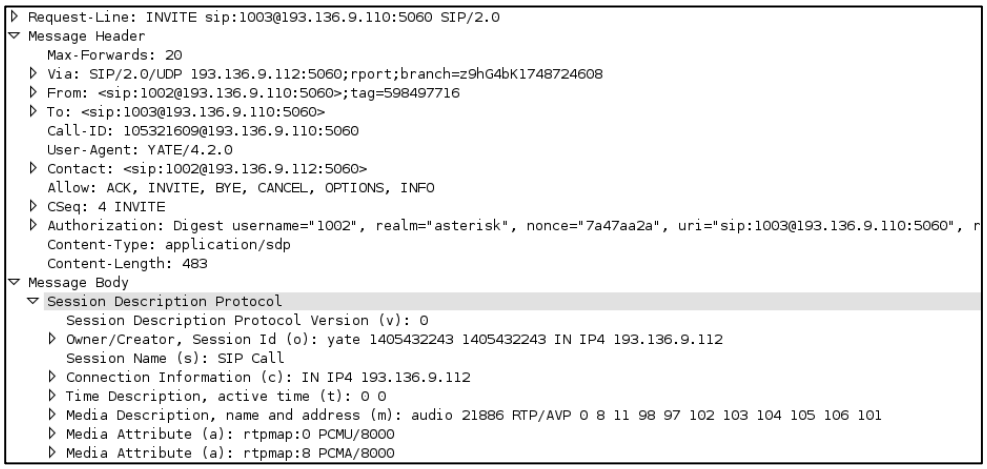


Figura 4.60. Informação SIP da "*INVITE with authentication*"

Na Figura 4.61 é apresentada uma replicação da Figura 4.60, tratando-se de uma pequena parte da configuração existente no ficheiro *emissor.xml*.

```
<send retrans="500">
  <![CDATA[
      INVITE sip:[field5]@[field1]:[field3] SIP/2.0
   Via: SIP/2.0/[transport] [local_ip]:[field4];branch=[branch]
   From: sipp <sip:[field0]@[local_ip]:[field4]>;tag=[pid]SIPpTag00[call_number]
   To: [field5] <sip:[field5]@[field1]:[field3]>
   Call-ID: [call_id]
   CSeq: 2 INVITE
   Contact: sip:[field0]@[local_ip]:[field4]
   [field2]
   Max-Forwards: 70
   Subject: Performance Test
   Content-Type: application/sdp
   Content-Length: [len]

   v=0
   o=user1 53655765 2353687637 IN IP[local_ip_type] [local_ip]
   s=-
   c=IN IP[media_ip_type] [media_ip]
   t=0 0
   m=audio [media_port] RTP/AVP 0
   a=rtpmap:0 PCMU/8000
   ]]>
</send>
```

Figura 4.61. Configuração da mensagem "INVITE + authentication" no *send.xml*

O ficheiro *CSV* que preenche o cenário acima possui os seguintes parâmetros:

```
1 SEQUENTIAL
2 1002;193.136.9.110;[authentication username=1002 password=user1002voipuminho2014];5060;5060;1003;
```

Este ficheiro *CSV*, possui mais um parâmetro em relação aos restantes. Parâmetro esse, que se encontra na última posição (field5) e indica o "nome" do cliente a ser contactado.

Cenário de recetor da chamada

A Figura 4.62 apresenta as mensagens necessárias para receber uma chamada de um dado cliente, sendo que o XML usado encontra-se no Apêndice C.

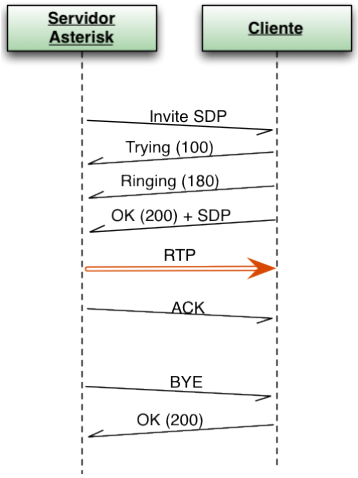


Figura 4.62. Registo dos clientes no Servidor Asterisk.

Para a execução do cenário de receção de chamadas de um dado cliente, na script ficaria:

```
$ ./voipp –r 1003 UDP 1
```

Na linha de comandos ficaria:

```
$ sipp 193.136.9.110:5060 -sf recetor.xml -inf 1003.csv -i 193.136.9.113 -p 5060 -m 1 -trace_stat -fd 2
```

A linha de comandos apresentada acima, tem a mesma interpretação que a linha de comandos para registo e iniciação da chamada, apenas com a diferença nos ficheiros usados. Pois o cenário XML (*recetor.xml*) possui mensagem SIP diferentes.

Seguindo a mesma ordem de explicação, na Figura 4.63 encontra-se, por exemplo, a informação SIP necessária para a receção da mensagem de *"INVITE"*. A configuração do ficheiro *recetor.xml* foi feita com o auxílio dos cenários já existentes no executável *sipp*, que podem ser importados através da linha de comandos "*sipp –sd uas*".

```
<recv request="INVITE" crlf="true">
</recv>
```

Figura 4.63. Informação SIP necessária para o envio da mensagem de "INVITE".

Cenário de registo com TLS

Os cenários com segurança diferem dos cenários anteriormente falados, em termos de protocolo e porta usados, que passou a ser TLS e 5061, respetivamente. Serão também necessários os certificados de segurança criados no ponto 4.3.1.1 e a configuração do ficheiro XML, que se encontra no Apêndice F

No caso concreto do cenário de registo, foi necessário efetuar a seguinte linha de comandos:

```
$ sipp 193.136.9.110:5061 -sf registoTLS.xml -inf 1002TLS.csv -i 193.136.9.112 -p 5061 -m 1 -trace_stat -fd 2 -tls_cert keys/cliente1002.pem -tls_key keys/cliente1002.key -t l1
```

A linha de comandos apresentada acima destina-se a informar, que o IP do Servidor Asterisk é o 193.136.9.110 e a porta é a 5061. Indica que será usado um cenário XML, com o nome registoTLS.xml (semelhante ao cenário de registo sem segurança, apenas se trocou as *labels* *sip* por *sips*), e serão usados os parâmetros colocados no ficheiro 1002TLS.csv. As opções seguintes indicam que será feita no máximo uma chamada, que o certificado TLS é o *cliente1002.pem*, já a chave encontra-se no ficheiro *cliente1002.key* e ambos encontram-se na diretoria *keys*. Por último é definido o uso do protocolo TLS (-t l1), sendo o IP local o 193.136.9.112 e a porta local é a 5061.

Para o preenchimento do ficheiro *registoTLS.xml*, foram usados os seguintes dados do ficheiro *1002TLS.csv* :

```
raquelpereira@cliente 1002:/sender$ vim 1002TLS.csv

  1 SEQUENTIAL
  2 1002;193.136.9.110;[authentication username=1002 password=user1002voipuminho2014];5061;5061;
```

Cenário de emissor da chamada com TLS

Este cenário segue as mesmas medidas que o cenário anterior, sendo a sua configuração apresentada no Apêndice H.

Para a execução do cenário de realização de chamadas para um dado cliente, na script desenvolvida, ficaria:

```
$ ./voipp –e 1003 TLS 1
```

Na linha de comandos ficaria (a interpretação desta linha é semelhante à do cenário anterior):

```
$ sipp 193.136.9.110:5061 -sf emissorTLS.xml -inf 1003TLS.csv -i 193.136.9.113 -p 5061 -m 1
-trace_stat -fd 2 -tls_cert keys/cliente1003.pem -tls_key keys/cliente1003.key -t l1
```

Na Figura 4.64 encontra-se a informação SIP necessária para o envio da mensagem de, por exemplo, *"INVITE with authentication"*, com o uso do protocolo de segurança TLS.

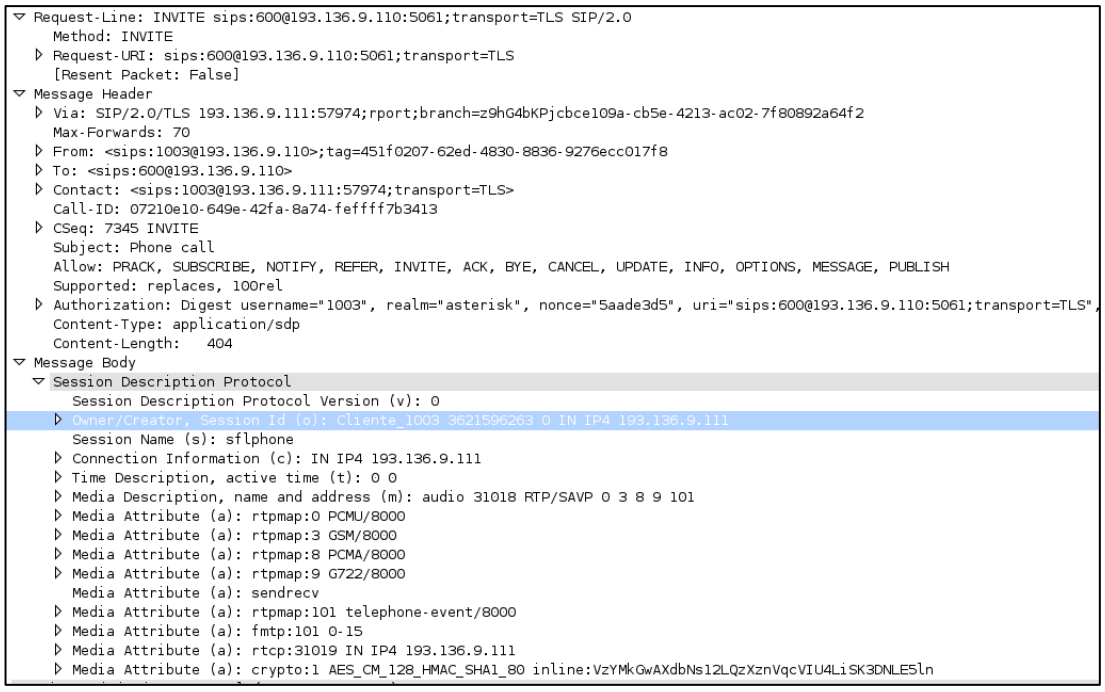


Figura 4.64. Informação SIP da mensagem de *"INVITE with authentication"* com TLS.

Na Figura 4.65 é apresentada uma replicação da Figura 4.64, trata-se de uma pequena parte da configuração existente no ficheiro *emissorTLS.xml*.

```
<send retrans="500">
    <![CDATA[

        INVITE sips:[field5]@[field1]:[field4];transport=[transport] SIP/2.0
        Via: SIP/2.0/[transport] [local_ip]:[field3];branch=[branch]
        From: <sips:[field0]@[local_ip]>;tag=[pid]SIPpTag00[call_number]
        To: <sips:[field5]@[field1]>
        Call-ID: [call_id]
        CSeq: 2 INVITE
        Contact: <sips:[field0]@[local_ip]:[field3];transport=[transport]>
        [field2]
        Max-Forwards: 70
        Content-Type: application/sdp
        Content-Length: [len]

        v=0
        o=[field0] 53655765 2353687637 IN IP[local_ip_type] [local_ip]
        s=-
        c=IN IP[media_ip_type] [media_ip]
        t=0 0
        m=audio [media_port] RTP/SAVP 0
        a=rtpmap:0 PCMU/8000
        a=crypto:1 AES_CM_128_HMAC_SHA1_80 inline:VzYMkGwAXdbNs12LQzXznVqcVIU4LiSK3DNLE5ln

    ]]>
</send>
```

Figura 4.65. Configuração da mensagem *"INVITE with authentication"*.

Para o preenchimento do ficheiro *emissorTLS.xml*, foram usados os seguintes dados do ficheiro *1003TLS.csv* :

```
raquelpereira@cliente 1002:/sender$ vim send_the_1003TLS.csv

 1 SEQUENTIAL
 2 1002;193.136.9.110;[authentication username=1002 password=user1002voipuminho2014];5061;5061;1003;
```

De acordo com a Figura 4.65 e com o auxilio de [16], a linha "a=crypto:1 AES_CM_128_HMAC_SHA1_80 inline: [...]" é um atributo de criptografia SDP para o transporte em causa, neste caso concreto é o transporte "RTP/SAVP", como é indicado no atributo "*m*". O atributo "a=crypto" segue a seguinte sintaxe "*a=crypto:<tag> <crypto-suite> <key-params> [<session-params>]*", que é apresentada em detalhe na Tabela 4.20.

Tabela 4.20.. Explicação do atributo "*a=crypto*" [61] e [62].

Campo	Definição
<tag>	Refere-se a um número decimal, que serve como identificador para um atributo criptográfico, nomeadamente para determinar qual dos atributos criptográficos oferecidos foi o escolhido pelo outro cliente. Este número deve ser único dentro dos vários atributos media.
<crypto-suite>	Especifica o algoritmo de criptografia e de autenticação a usar. Por exemplo, "AES_CM_128_HMAC_SHA1_32" indica que será usado o algoritmo criptográfico AES-CM com uma chave de 128 bits e também que será usado o algoritmo de autenticação HMAC-SHA1 de 32 bits.
<key-params>	Fornece um ou mais conjuntos de materiais chave para o "*crypto-suite*" usado. Este campo é composto pelo método, seguido de ":" e posteriormente informações de codificação real, por exemplo: *key-params = <key-method> ":" <key-info>*. De acordo com [61], o único método definido é o "*inline*", onde indica que o material de codificação se encontra no campo *<key-info>*. *<key-info>* é definido por uma *string* de octetos. É possível usar mais do que uma chave.
[<session-params>]	É opcional e especifica um determinado transporte.

Cenário de recetor da chamada com TLS

Este cenário segue as mesmas medidas que os dois cenários anteriores, sendo a sua configuração apresentada no Apêndice G.

Para a execução do cenário de receção de chamadas de dado cliente, na script ficaria:

```
$ ./voipp –r 1002 TLS 1
```

Na linha de comandos ficaria (tem a mesma interpretação dos cenários de registo e de iniciação de chamada):

```
$ sipp 193.136.9.110:5061 -sf recetorTLS.xml -inf 1002TLS.csv -i 193.136.9.112 -p 5061 -m 1 -trace_stat -fd 2 -tls_cert keys/cliente1002.pem -tls_key keys/cliente1002.key -t l1
```

4.6 Resumo

Neste capítulo apresentaram-se detalhadamente os componentes que formam a arquitetura proposta para o sistema de avaliação de desempenho para sistemas VoIP seguros, bem como as respectivas formas de utilização e configuração. Foram usadas apenas componentes de código aberto, de forma a que o sistema descrito possa ser facilmente reproduzido. Foram também desenvolvidas algumas scripts que ajudam a automatizar o processo.

5. Testes e resultados

Neste capítulo é apresentado, sob a forma de gráficos, todos os resultados dos testes realizados. Numa primeira fase (ponto 5.1), todos os testes foram executados com diferentes mecanismos de segurança. Sendo reunidos todos os dados necessários e apresentados todos os gráficos considerados importantes, de acordo com os resultados obtidos. Posteriormente, ponto 5.2, são acrescentados condicionantes, nomeadamente *delay*, *packet loss* e largura de banda. E por fim, no ponto 5.3 é apresentado um pequeno resumo do trabalho feito.

5.1 Cenário de referência

O cenário de referência adotado (ver Figura 5.1), inclui o Servidor Asterisk (máquina 1), o CACTI (máquina 2), o Cliente 1002 (máquina 4) e o Cliente 1003 (máquina 5). A máquina 3 foi usada como auxiliar para, por exemplo, testar algumas ferramentas.

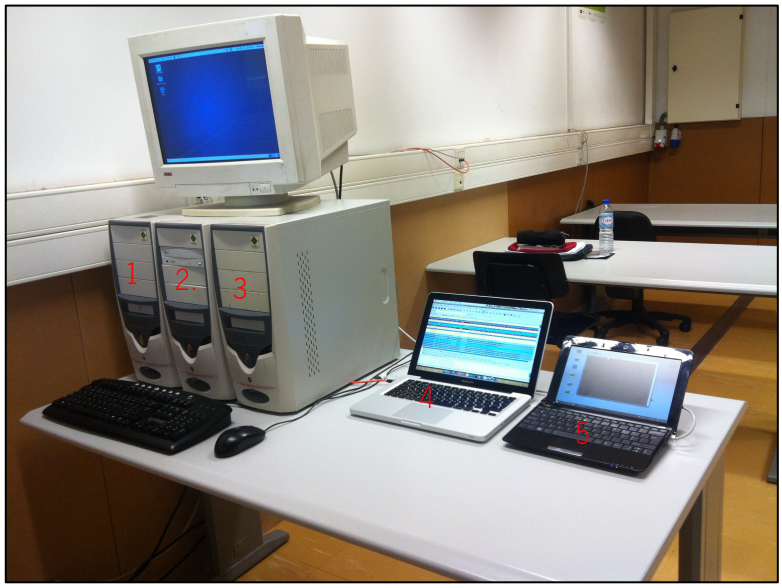


Figura 5.1. Bancada de testes.

O Gráfico 5.1, apresenta a gestão do SIPp face ao número de chamadas em simultâneo. Isto é relevante para identificar os processamentos internos das diversas máquinas envolvidas, ou seja, perceber qual a carga gerada nas máquinas para um elevado número de chamadas em simultâneo.

De acordo com o gráfico foram executadas 1600 chamadas, mas o SIPp só permitiu que fossem executadas aproximadamente 241 chamadas em simultâneo. Foram executadas 1600 chamadas, porque é aproximadamente o limite da largura de banda disponível (ver Figura 4.6), de acordo com o CODEC usado, ou seja, com aproximadamente uma largura de banda de 100Mbits/seg, podem ser executadas aproximadamente 1562 chamadas, com o CODEC de 64Kbits/s. O limite máximo de chamadas em simultâneo é de 241, porque é o limite de ficheiros abertos, estabelecido pelas máquina envolventes. Esse limite pode ser verificado com o uso do comando "*ulimit -n*".

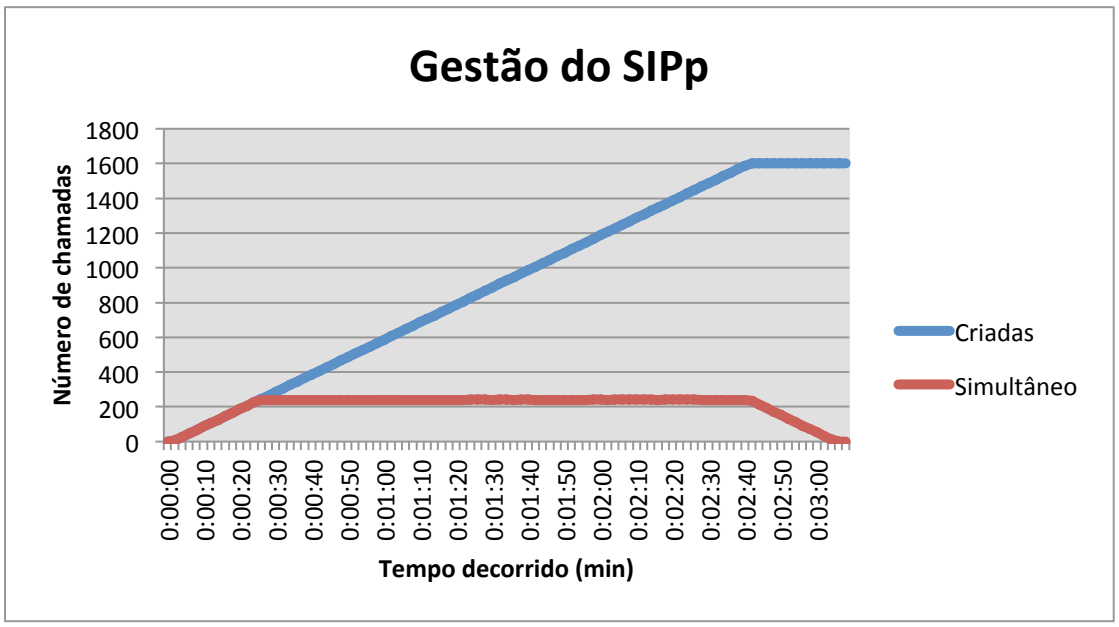


Gráfico 5.1. Gestão do número de chamadas pelo SIPp.

5.1.1 *Delay*

O *delay* obtido nos testes com diferentes mecanismos de segurança e com auxilio do *Wireshark*, pode ser consultado no Gráfico 5.2. Esperava-se que todos os testes com segurança efetuados, mantivessem uma distância significativa em relação ao teste sem qualquer tipo de segurança. No entanto o TLS superou, pois mais de 80% dos pacotes encontram-se numa gama de valores inferior a 88 milissegundos de *delay*. Tanto o *IPSec* como a *firewall* encontram-se quase a par, sendo que na última gama de valores o *IPSec* destaca-se. Face à análise feita, o melhor mecanismo seria o TLS e o pior seria o uso de *firewall*.

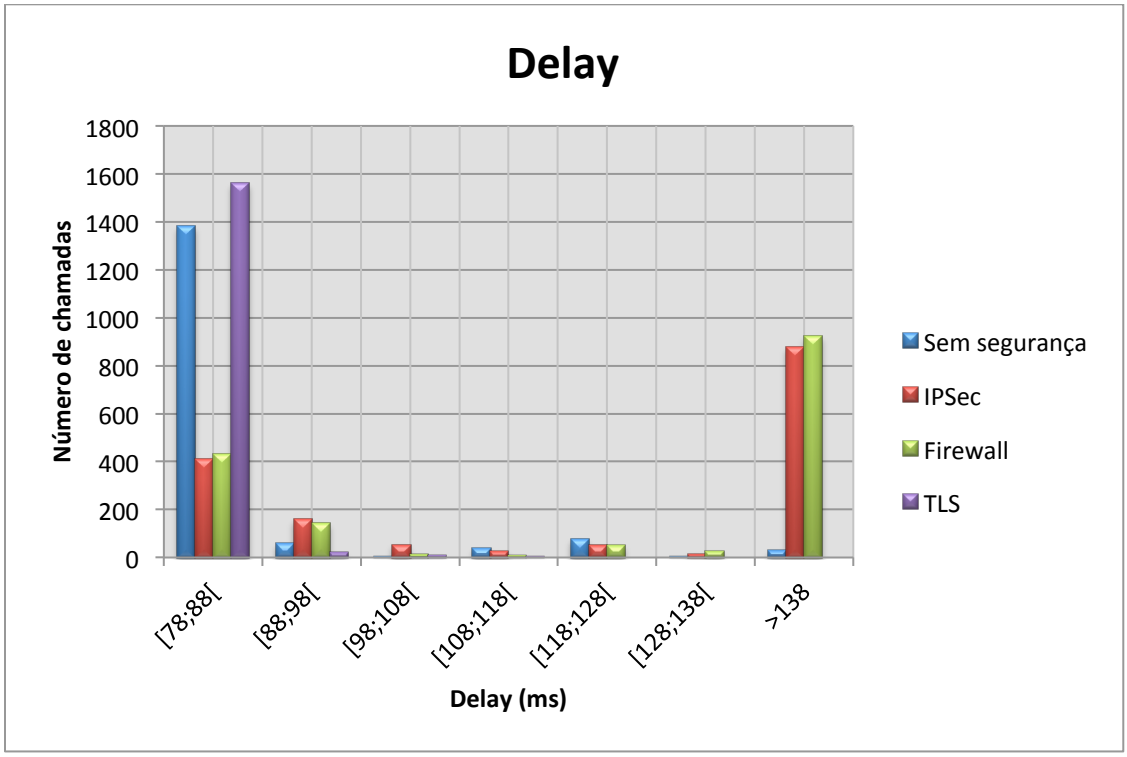


Gráfico 5.2. Delay resultante dos diferentes mecanismos de segurança.

5.1.2 Pacotes Perdidos

O Gráfico 5.3 foi construído com o auxilio do Wireshark e reforça a escolha feita anteriormente, ou seja, o TLS não perdeu um único pacote. De salientar, que o facto de o cenário com TLS possuir melhores resultados que o cenário sem segurança, pode dever-se a diversos factores que não estão ao nosso controlo, ou seja, a rede de testes usada é também utilizada por outras entidades, sendo recomendável a repetição de todos os testes em condições de carga da rede equiparáveis. O pior mecanismos de segurança continua a ser a *firewall.*

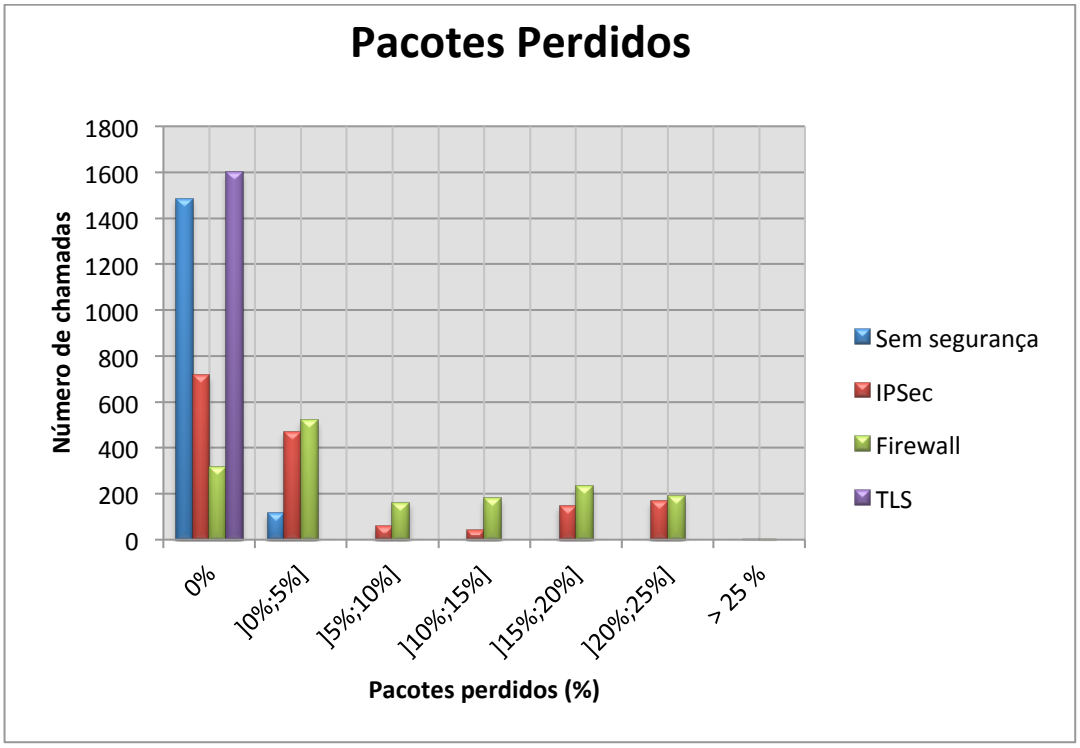


Gráfico 5.3. Pacotes Perdidos resultante dos diferentes mecanismos de segurança.

5.1.3 *Jitter*

O *jitter* médio é resultante da variação do *delay* e os seus resultados são apresentados no Gráfico 5.4, também ele construído com o auxilio do Wireshark. Sendo que todos os testes se concentram entre a gama de valores de 27 a 27,5 milissegundos, à exceção do TLS, que se encontra na gama de valores superior a 28,5 milissegundos, não apresenta uma diferença muito significativa dos restantes cenários.

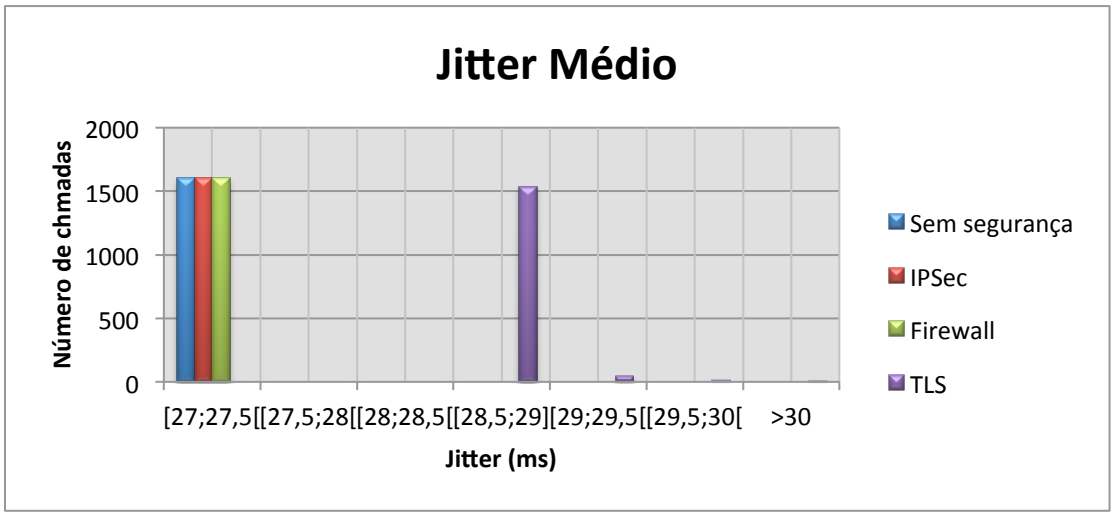


Gráfico 5.4. *Jitter* resultante dos diferentes mecanismos de segurança.

5.1.4 Pacotes

No que diz respeito ao número de pacotes SIP e RTP é apresentado o Gráfico 5.5 e o Gráfico 5.6, respetivamente. Estes gráficos foram construídos com o auxilio do wireshark e pretendem demonstrar, quando comparados com o teste sem segurança, qual o mecanismo de segurança mais adequado. Sendo que nos dois gráficos são retiradas conclusões muito diferentes, pois no que respeita aos pacotes de controlo o mecanismo de *IPSec* é o mais adequado, isto porque, até apresenta menos pacotes que o cenário sem segurança, podendo dever-se às retransmissões existentes. Já no que respeita aos pacotes RTP, o uso de *firewall* possui um número de pacotes inferior ao cenário sem segurança, tendo em conta que nos

pacotes RTP não existe retransmissões. Este gráfico pode ser justificado com o Gráfico 5.3, onde se observa que o uso de *firewall* é o que possui mais perda de pacotes.

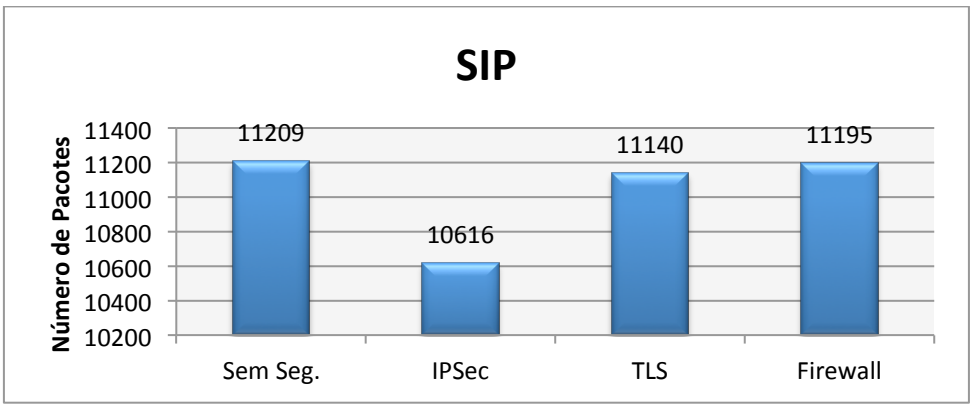


Gráfico 5.5. Pacotes *SIP* resultantes dos diferentes mecanismos de segurança.

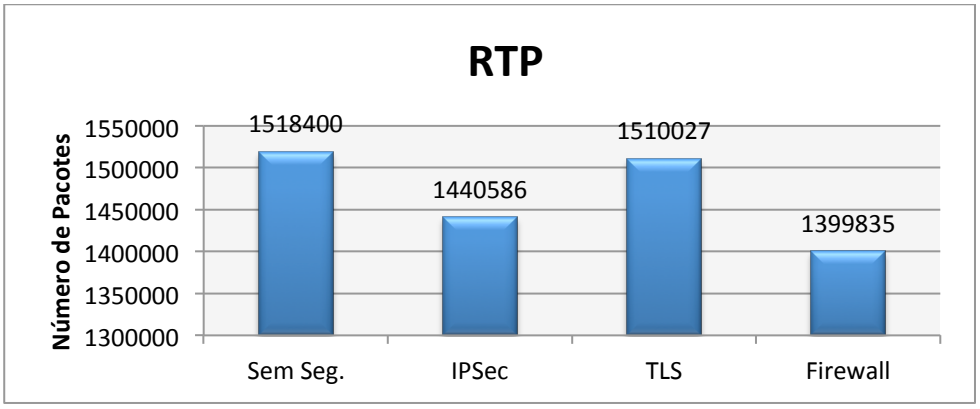


Gráfico 5.6. Pacotes *RTP* resultantes dos diferentes mecanismos de segurança.

5.1.5 Estado da máquinas

O estado da máquinas envolvidas é um aspecto importante a ter conta na escolha do mecanismo de segurança e tais valores foram obtidos através do CACTI. De acordo com o Gráfico 5.7, é possível visualizar que o *IPSec* é o protocolo de segurança que mais "trabalho" exerce nas máquinas e o TLS é o menos. O Gráfico 5.8 também confirma estes dados, ou seja, o *IPSec* é o que gera mais tráfego e o TLS é o que menos tráfego gera. De salientar, que a máquina 1002 estava a executar mais atividades para além dos testes. Será pois recomendável

a realização de mais testes, em condições semelhantes de carga de CPU. O fato de haver consumos de CPU superiores a 100%, nas máquinas 1003 e 1002, deve-se aos núcleos (reais ou físicos). O CACTI aumenta em 100% quando se faz uso de mais um núcleo. Neste caso o número máximo de núcleos usados foi dois.

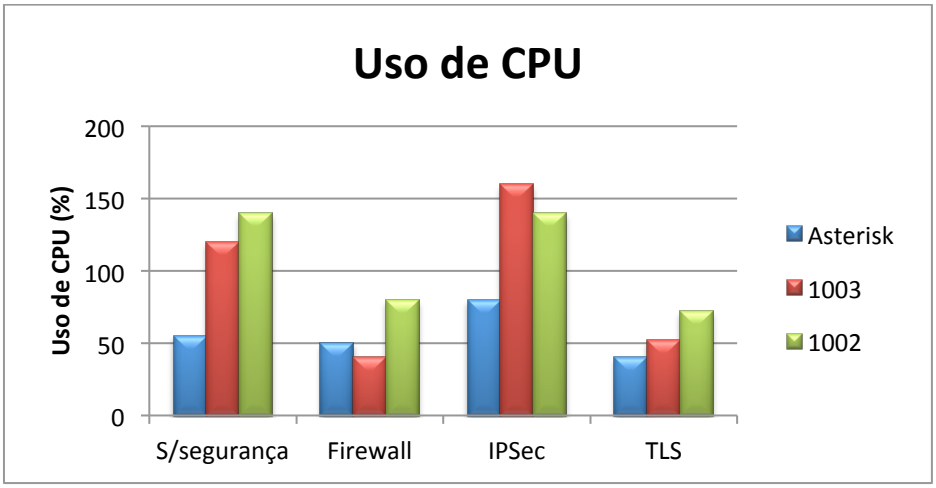


Gráfico 5.7. Uso de CPU com os vários mecanismos de segurança.

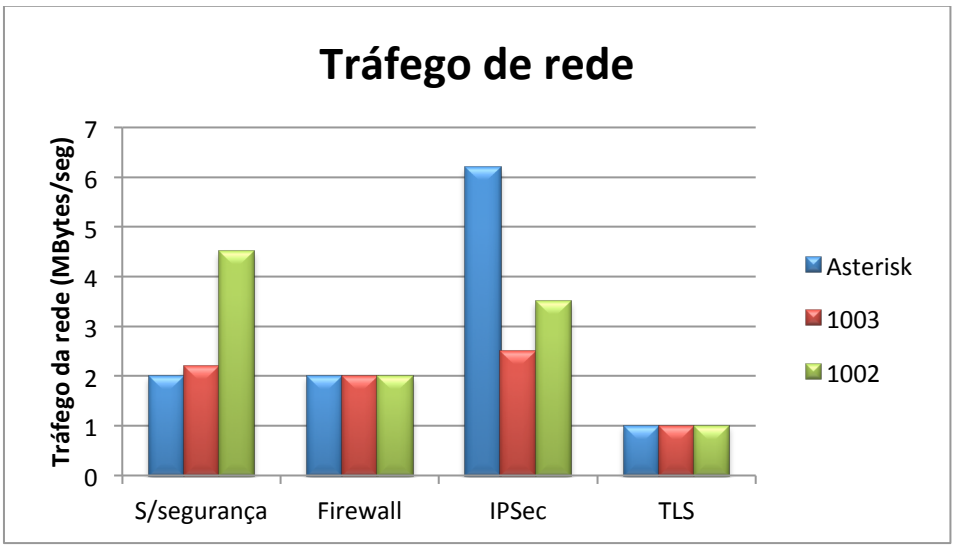


Gráfico 5.8. Tráfego gerado com os vários mecanismos de segurança.

Na Figura 5.2 e na Figura 5.3, são mostrados os comportamentos das máquinas face ao que já foi dito. Para que não fique muito detalhado apenas se achou relevante falar do cenário

básico (sem segurança) e do cenário que se achou mais adequado para este tipo de aplicações (com TLS). As duas figuras provam assim, todas as conclusões descritas.

Figura 5.2. Estado da máquinas sem segurança.

Figura 5.3. Estado da máquinas com TLS.

5.2 Cenário com condicionantes

Para este cenário, foram usadas restrições de *delay*, perda de pacotes e largura de banda. O objetivo é concluir sobre o comportamento das chamadas face a estas condicionantes. O número de chamadas foi fixado a 800, sendo adicionadas várias gamas de valores a cada tipo de restrição e cada teste possuirá apenas variação num único parâmetro de QoS (ver 3.3.1). Não se misturaram os diversos cenários de segurança com a variação dos parâmetros de QoS, apenas se usou o cenário sem segurança, pois o número de testes/gráficos iria ser muito elevado. Recomenda-se a realização de testes equivalentes nos vários cenários com segurança, de modo a verificar se o comportamento é semelhante. Será feita uma análise individual, face a cada valor de QoS usado.

5.2.1 Delay

O valores de *delay* resultantes de cada teste, encontram-se na gama do esperado, não sendo necessário a apresentação do seu gráfico. Ou seja, introduzindo uma variação de 450 ms ela observa-se como um acréscimo na chegada.

No que diz respeito aos pacotes perdidos (ver Gráfico 5.9), a variação introduzida no *delay* pouco ou nada afectou as perdas verificadas. Apenas se detetou, na utilização do delay de 450 milissegundos, duas chamadas perdidas, sendo apresentadas com 25% de pacotes perdidos.

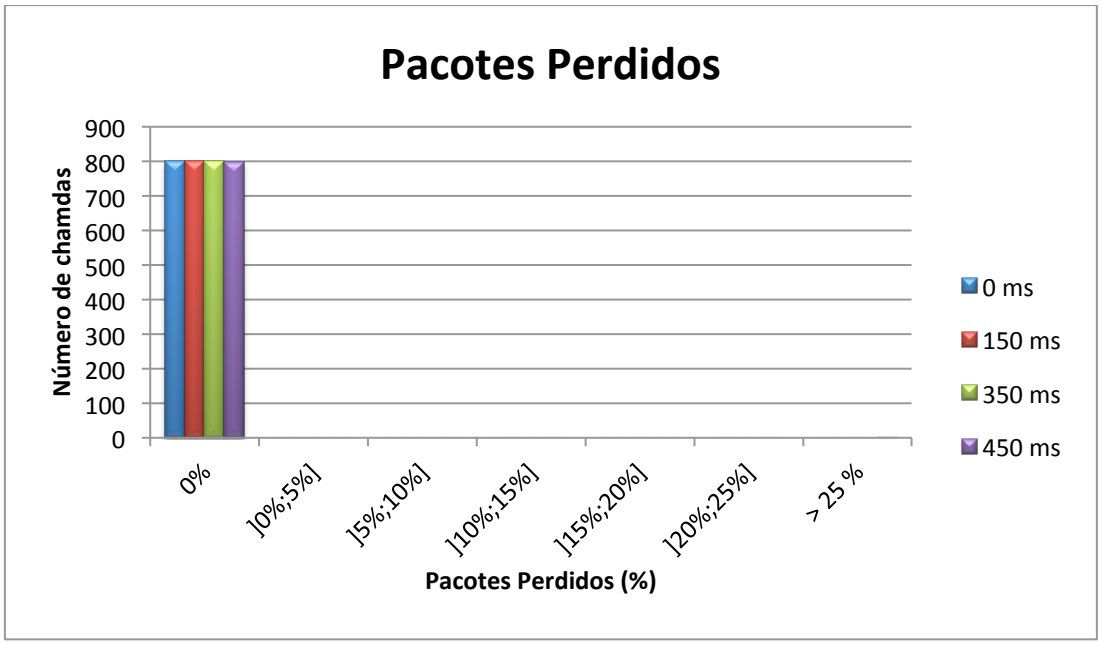


Gráfico 5.9. Pacotes perdidos resultante da variação do *delay*.

5.2.2 Pacotes Perdidos

O gráfico de pacotes perdidos resultante da inserção de perda de pacotes, é de acordo com o esperado, não se achando necessário a sua apresentação.

No Gráfico 5.10, é apresentado o *delay* resultante da perda de pacotes. Onde foi possível concluir que a perda de pacotes tem um impacto direto no *delay*, principalmente no que diz respeito aos pacotes de controlo. Sendo que a perda de pacotes RTP, vai complicar a ligação ou mesmo impossibilitar a sua comunicação.

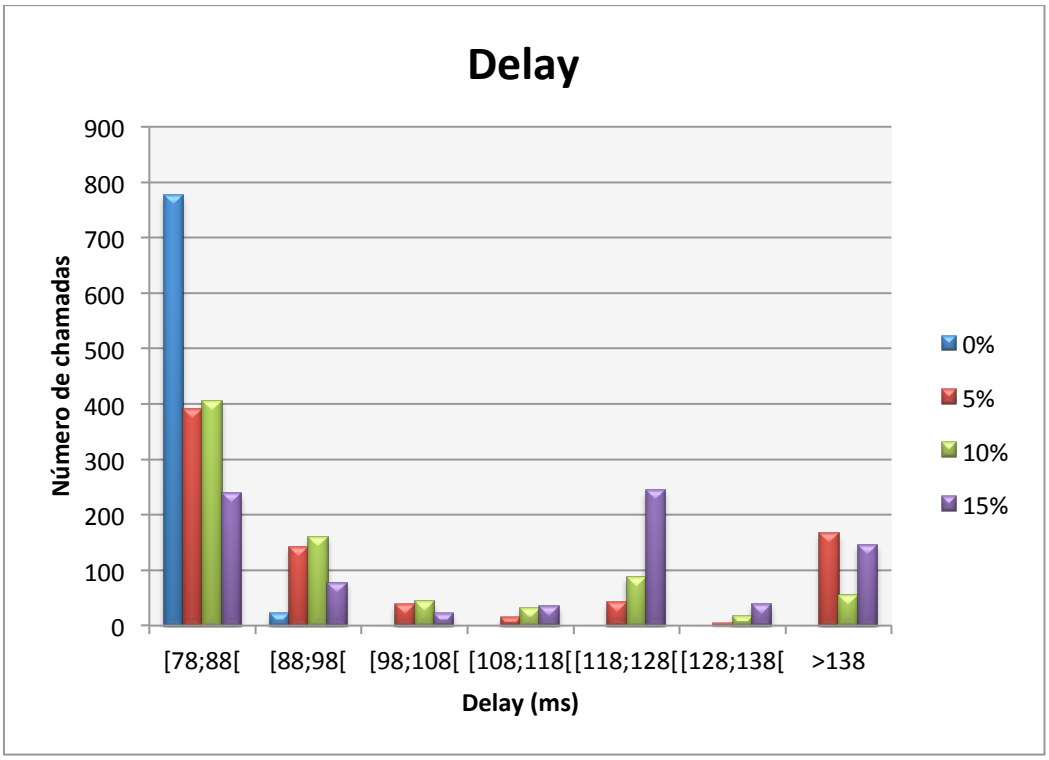


Gráfico 5.10. *Delay* resultante da variação dos pacotes perdidos.

5.2.3 Largura de banda

Para o estudo do impacto da largura de banda, apenas se restringiu a largura de banda para metade, ou seja, 50Mbits/seg.

O Gráfico 5.11, apresenta o impacto que a largura de banda disponível possui no *delay* das chamadas. Sendo que, como o esperado, o número de chamadas concentradas entre os 78 e 88 milissegundos, é mais do que reduzido para metade, quando comprado com o cenário com a largura de banda a 100Mbits/seg.

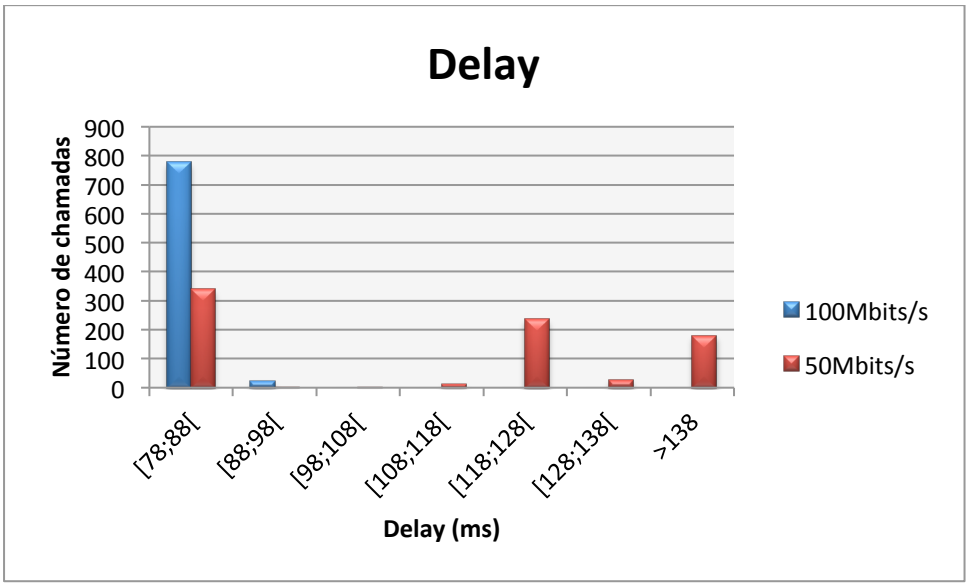


Gráfico 5.11. *Delay* resultante da alteração da largura de banda.

O Gráfico 5.12, apresenta o impacto, da restrição da largura de banda, na perda de pacotes. Sendo que o número de chamadas sem qualquer perda de pacotes é reduzido para cerca de um quarto, quando comparado com o cenário de largura de banda a 100Mbits/seg.

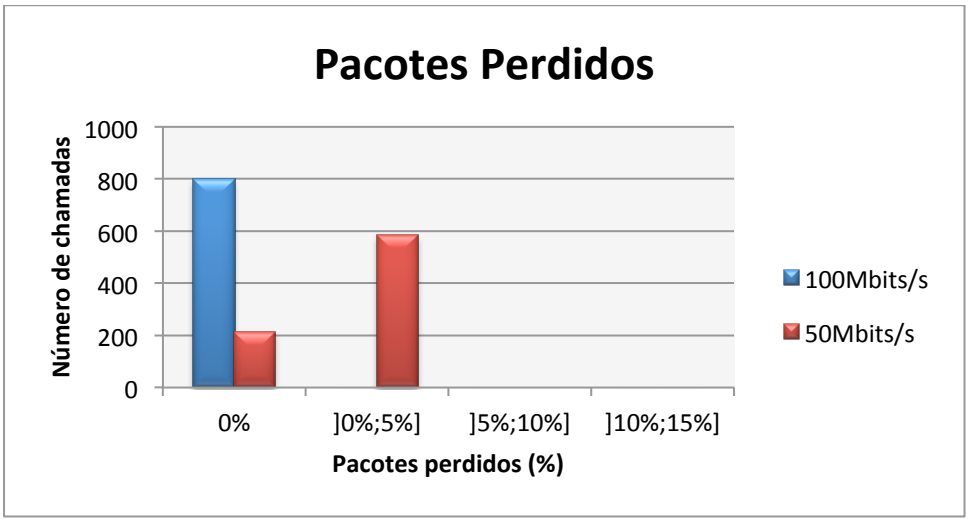


Gráfico 5.12. Pacotes perdidos resultante da alteração da largura de banda.

5.2.4 Estado da máquinas

A Figura 5.4, apresenta os vários comportamentos das várias máquinas. Sendo que a interpretação de cada gráfico existente nessa figura é semelhante, ou seja, cada "pico" corresponde a cada evento pela seguinte ordem, *delay:* 150ms, 350ms e 450ms, *packet loss*: 0.5%, 1% e 15% e por fim a largura de banda a 50Mbits/s.

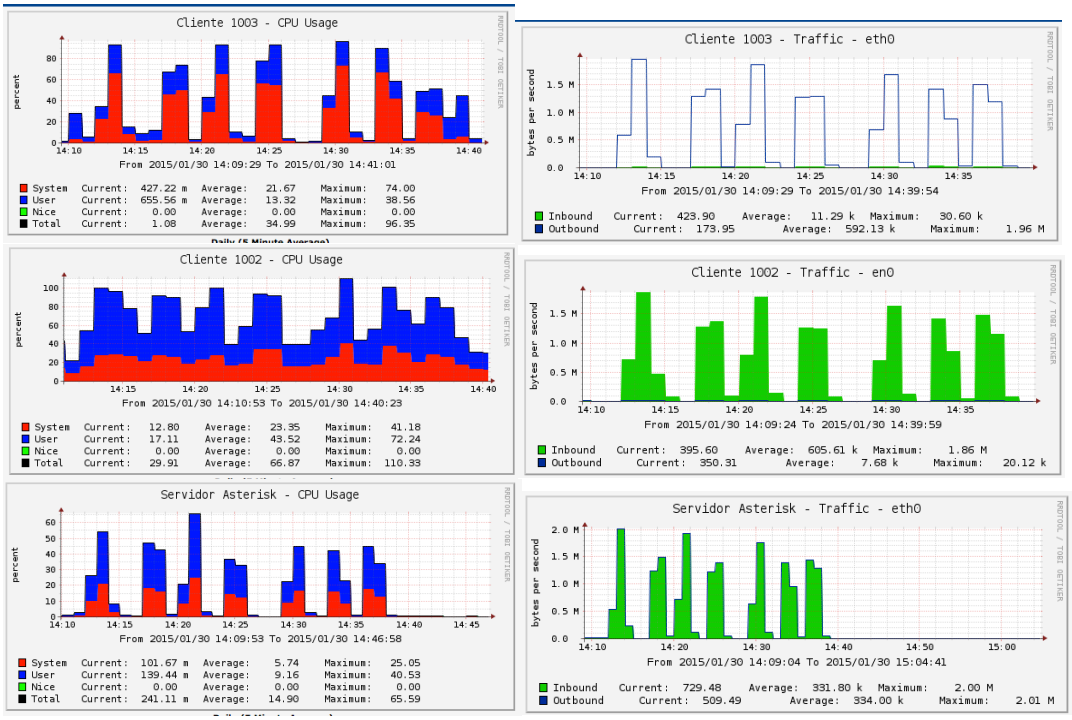


Figura 5.4. Estado da máquinas com as variações de QoS.

5.3 Resumo

O primeiro conjunto de testes permite avaliar o impacto dos diferentes mecanismos de segurança quer na sinalização quer no tráfego de voz. A análise global dos dados permite concluir que a opção TLS introduz menos *delay* e menos perdas na comunicação que as alternativas *IPSec*. O mesmo em relação à utilização de *firewalls* que degrada o *delay* e as perdas. Apenas no *jitter*, a opção pelo TLS mostrou um ligeiro agravamento, provavelmente não significativo, mas a confirmar em testes posteriores. De notar que em termos de consumo de

recursos nas máquinas (CPU, memória) também a opção TLS se mostrou mais leve que a alternativa *IPSec*.

Os testes com condicionantes, que pretendem mostrar qual o impacto que a alteração das condições da rede introduz na comunicação, mostram que as perdas e atrasos afectam linearmente as chamadas. A condicionante com maior impacto no QoS é a largura de banda. Será ainda necessário executar um conjunto adicional de testes, com planeamento adequado de experiências, dado o número de condicionantes que é possível alterar para cada solução de segurança e o impacto nas métricas de QoS.

Testes com a tecnologia WebRTC não foi possível implementar, porque não se conseguir efetuar mais de que uma chamada em simultâneo.

6. Conclusões e Trabalho Futuro

Nesta dissertação propôs-se um sistema de avaliação de desempenho de sistemas VoIP seguros, totalmente baseado em ferramentas de código aberto. O sistema foi implementado numa bancada de testes em ambiente real, usando o software Asterisk como central telefónica VoIP. O Asterisk foi escolhido por ser um dos pacotes de software mais completos e mais utilizados nos dias de hoje.

Em primeiro lugar fez-se um estudo completo do VoIP e das tecnologias relacionadas. Com destaque para o estudo dos principais ataques num serviço deste tipo, e dos mecanismos de segurança, métricas de qualidade de serviço e estratégias de avaliação de desempenho. O VoIP é uma tecnologia de voz sobre o protocolo da internet, complexa, que obriga a um elevado conhecimento na área, sistematizado no capítulo 2 deste documento. As experiências práticas conduzidas e descritas são também indispensáveis, para sedimentar os conhecimentos adquiridos.

A partir do estudo feito, percebe-se que a maioria das soluções em termos de segurança passa pela utilização de TLS/SSL na proteção da sinalização SIP e do protocolo SRTP na proteção dos dados de voz. Ou então, em alternativa, o uso de IPSec ao nível da rede IP. Os ataques e vulnerabilidades são muitas vezes categorizadas como ataques contra a disponibilidade (negação de serviço, inundação de chamadas, mensagens de texto erradas, desvio de mensagens, repetição indevida de mensagens, degradação e abuso do QoS disponível, etc), contra a confidencialidade (observação indevida de conversas, observação indevida do padrão de chamadas dos utilizadores, análise dos fluxos de dados, etc), contra a integridade (remoção ou inserção de pacotes de áudio, desvio de chamadas, etc) e contra o engenharia social (chamadas indesejadas, tentativas de roubo de identidade, disfarce, etc). As medidas de segurança a adotar interferem naturalmente com as métricas de QoS como o atraso, o *jitter* e estrangulamentos na largura de banda disponível.

Foi pois proposto um sistema completo, baseado em ferramentas de código aberto, capaz de ser integrado num sistema VoIP típico. Os componentes identificados são: gerador automático

de chamadas (em grande número), recetor e analisador automático de chamadas, condicionador de tráfego para ajuste dos parâmetros e condições da rede de suporte, capturador e analisador de tráfego à chegada, servidor de monitorização ativa em tempo real, detetor de ameaças nos *logs*. Para cada um os componentes foi selecionada uma ferramenta, depois instalada e testada na bancada de testes.

A montagem da bancada de testes, demorou mais tempo de que o previsto. Isto deve-se às máquinas serem um pouco antigas e ao uso de diferentes Sistemas Operativos. Dado que, por exemplo, o *Linux Mint* e o *CentOS* são duas distribuições *Linux*, mas no que diz respeito à implementação *IPSec*, o *Linux Mint* usa o a ferramenta *racoon* e o *CentOS* o *pluto*. No caso da recolha de dados o processo é razoavelmente simples, mas na análise e realização do gráficos é bastante demorado. O SIPp também é muito complexo, e exige muito tempo de adaptação para geração dos cenários.

Da experiência de montagem pode-se concluir que os mecanismos de segurança escolhidos não são assim tão fáceis de configurar como deveria ser. Será sempre uma tarefa a ser feita não pelo utilizador comum, mas por um técnico administrador dos sistemas. Ou com software mais amigável. O mesmo também se pode dizer da configuração de um softphone. Que também exige preencher alguns dados que só se identificam com conhecimentos de segurança.

As métricas de avaliação que foram escolhidas, permitem uma comparação entre os diversos cenários, possibilitando tomar decisões de, por exemplo, o melhor mecanismo de segurança a adotar.

Os resultados obtidos no conjunto de cenários mostram que é no cenário sem segurança que se conseguem realizar um maior número de chamadas, o que é um resultado esperado. Quanto aos cenários com segurança, a utilização de TLS mostrou em geral melhores resultados que a utilização do IPSec, para o conjunto de testes realizados. No entanto será necessário realizar mais testes, em mais cenários, para poder confirmar esta afirmação.

A implementação do sistema mostrou a sua viabilidade e a funcionalidade esperada. E também que pode ser adaptado a outros cenários, quer em ambientes operacionais ou apenas de testes.

Os testes realizados permitiram obter alguns dos resultados para o conjunto de experiências inicialmente planeando. É no entanto necessário fazer um conjunto de testes mais exaustivo. E também obter mais algumas métricas e/ou novos gráficos. Efetuar testes em cenários mais alargados, com diferentes softphones, verificar o impacto com diferentes CODECs e se possível medir a qualidade da voz, através do MOS (*Mean Opinion Score*).

Além disso, seria também importante continuar o desenvolvimento de alguns componentes. Foi desenvolvido um conjunto de scripts que permite simplificar alguns dos procedimentos. Esse conjunto de scripts formam um primeiro protótipo de novos componentes que pode ser melhorado. Um componente que seria importante desenvolver, e que foi equacionado, é a substituição do Wireshark por um componente Java mais específico construído à medida sobre as bibliotecas de suporte do Wireshark. De modo a que a obtenção de determinadas métricas, que envolvem tráfego cifrado, e para o qual se tem de introduzir manualmente os certificados, chaves e outra informação, possa ser feito de forma muito mais expedita. Melhorando significativamente o processo.

Um outro aspecto importante é melhorar os testes na componente WebRTC. É possível fazer chamadas de teste, dos browsers como por exemplo o Chrome, mas não foi ainda possível automatizar a geração de muitas chamadas e também a receção. Uma ferramenta com essas funções, construída sobre as bibliotecas usadas nos browsers, é importante, para um melhor estudo da tecnologia WebRTC. Pois trata-se de uma tecnologia "do futuro". Estudar formas de executar diversas chamadas em simultâneo.

Finalmente, para trabalho futuro, equacionar o alargamento do sistema de testes a ambientes de simulação e ou emulação, ou ambientes mistos, combinados, de componentes reais e simuladas. Seria interessante construir uma topologia de suporte no emulador CORE, ou no simulador NS-3, e analisar o desempenho por comparação com os resultados reais. O que permitiria também levar a avaliação para novas arquiteturas VoIP *peer-to-peer*, ou outras.

Referências

[1] ITU-T, "H.323," *Ser. H Audiov. Multimed. Syst.*, 2009.

[2] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler, "SIP: Session Initiation Protocol - RFC 3261," pp. 1–269, 2002.

[3] C. Perkins, M. Westerlund, and J. Ott, "Web Real-Time Communication (WebRTC): Media Transport and Use of RTP," 2013.

[4] N. Ribeiro and L. Mendes, "VoIP – Tecnologia de Voz sobre IP," 2008.

[5] M. Patel and B. V Buddhdev, "Analysis of Security Threats in Voice Over Internet Protocol (VOIP)," vol. 3, no. 5, pp. 30–38, 2013.

[6] A. Lazzez, "VoIP Technology: Security Issues Analysis," *Taif University*, vol. 2, no. 4, Kingdom of Saudi Arabia, pp. 333–341, 2013.

[7] Q. Gong, "Playout Buffering for Conversational Voice over IP," McGill University, 2012.

[8] R. Dantu, S. Fahmy, H. Schulzrinne, and J. Cangussu, "Issues and challenges in securing VoIP," *Computers & Security*, vol. 28, no. 8, pp. 743–753, Nov-2009.

[9] R. Sadiwala and M. Sharma, "Security Threats of VoIP," vol. 1, no. 1, pp. 7–18, 2014.

[10] H. B. Silva, "Qualidade de Serviço em chamada VoIP," Universidade do Porto, 2008.

[11] L. Tian, N. Dailly, Q. Qiao, J. Lu, J. Zhang, J. Guo, and J. Zhang, "Study of SIP protocol through VoIP solution of 'Asterisk,'" *Mob. Congr. (GMC), 2011 Glob.*, pp. 1–5, Oct. 2011.

[12] A. G. Robertson, "Implantando um serviço de Telefonia IP em Empresas de grande e médio porte," Nitéroi, RJ, 2010.

[13] A. D. Keromytis, "A Comprehensive Survey of Voice over IP Security Research," *IEEE Commun. Surv. TUTORIALS*, vol. 14, no. 2, pp. 514–537, 2012.

[14] M. Handley, V. Jacobson, and C. Perkins, "SDP: Session Description Protocol - RFC 4566," pp. 1–50, 2006.

[15] V. Toncar, “VoIP Protocols: SIP — Session Description Protocol.” [Online]. Available: http://toncar.cz/Tutorials/VoIP/VoIP_Protocols_SIP_Session_Description_Protocol.html. [Accessed: 10-Dec-2014].

[16] J. Santugini-Repiquet, “SDP Attributes,” 2005. [Online]. Available: http://www.in2eps.com/fo-abnf/tk-fo-abnf-sdpatt.html#r-3605. [Accessed: 11-Dec-2014].

[17] H. Schulzrinne, R. Frederick, and V. Jacobson, “RTP: A Transport Protocol for Real-Time Applications - RFC 3550,” pp. 1–89, 2003.

[18] G. Talaganov, “Green VoIP : A SIP Based Approach,” School of Information and Communication Technology, 2012.

[19] M. Baugher, D. Mcgrew, M. Naslund, E. Carrara, and K. Norrman, “The Secure Real-time Transport Protocol (SRTP) Status - RFC 3711,” 2004.

[20] F. Andreasen and B. Foster, “Media Gateway Control Protocol (MGCP) - RFC 3435,” 2003.

[21] C. Sivachelvan and B. Foster, “Media Gateway Control Protocol (MGCP) Return Code Usage - RFC 3661,” pp. 1–24, 2003.

[22] T. Taylor, “Reclassification of RFC 3525 to Historic - RFC 5125,” pp. 1–4, 2008.

[23] D. Butcher, X. Li, and J. Guo, “Security Challenge and Defense in VoIP Infrastructures,” *IEEE Trans. Syst. Man Cybern. Part C Appl. Rev.*, vol. 37, no. 6, pp. 1152–1162, Nov. 2007.

[24] D. Weerasinghe, *Information Security and Digital Forensics*. 2009.

[25] I. Pezelj, J. Ožegović, and L. Hrboka, “VoIP QoS on low speed links,” *12. Int. Conf. Software, Telecommun. Comput. Networks*, 2004.

[26] H. P. Singh, S. Singh, J. Singh, and S. A. Khan, “VoIP: State of art for global connectivity—A critical review,” *J. Netw. Comput. Appl.*, vol. 37, pp. 365–379, Jan. 2014.

[27] “WebRTC.” [Online]. Available: http://www.webrtc.org. [Accessed: 13-Mar-2014].

[28] S. Dutton, “Getting Started with WebRTC,” 2014. [Online]. Available: http://www.html5rocks.com/en/tutorials/webrtc/basics/. [Accessed: 03-Jan-2015].

[29] V. M. de F. Amaral, “Framework e Cliente WebRTC,” Universidade do Minho, 2013.

[30] 3CX Innovating Communications, “WebRTC VoIP.” [Online]. Available: http://www.3cx.com/pbx/webrtc/. [Accessed: 29-Oct-2014].

[31] A. Saad, “Secure VoIP Performance Measurement,” Loughborough University, 2013.

[32] R. Gayraud and O. Jacques, “Welcome to SIPp.” [Online]. Available: http://sipp.sourceforge.net/. [Accessed: 20-Sep-2014].

[33] L. Rizzo, “The dummynet project.” [Online]. Available: http://info.iet.unipi.it/~luigi/dummynet/. [Accessed: 01-Oct-2014].

[34] “Yate Client.” [Online]. Available: http://yateclient.yate.ro/. [Accessed: 20-Sep-2014].

[35] “SFLphone.” [Online]. Available: http://sflphone.org/. [Accessed: 20-Sep-2014].

[36] Digium, “AsteriskNOW.” [Online]. Available: http://www.asterisk.org/downloads/asterisknow. [Accessed: 20-Sep-2014].

[37] “Wireshark.” [Online]. Available: https://www.wireshark.org/. [Accessed: 20-Sep-2014].

[38] “Iperf.” [Online]. Available: https://iperf.fr/. [Accessed: 28-Jan-2015].

[39] T. Dierks and E. Rescorla, “The Transport Layer Security (TLS) Protocol - RFC 5246,” pp. 1–104, 2008.

[40] A. Lima, “Estudo da Viabilidade de um Serviço VPN assente numa Arquitectura Redundante,” Faculdade de Engenharia da Universidade do Porto, 2010.

[41] “Certificate File Formats,” 2014. [Online]. Available: http://support.sas.com/documentation/cdl/en/secref/67436/HTML/default/viewer.htm#n0z2rzwzxh81czn12ughc6fpoh1s.htm. [Accessed: 13-Jan-2015].

[42] S. Vandeven, “SSL/TLS: What’s Under the Hood Encrypted,” *SANS Inst.*, 2013.

[43] S. Frankel and S. Krishnan, “IP Security (IPsec) and Internet Key Exchange (IKE) Document Roadmap - RFC 6071,” *Internet Eng. Task Force*, pp. 1–63, 2011.

[44] M. Latosiewicz, “Crypto map based IPsec VPN fundamentals - negotiation and configuration,” *Cisco Support Community*, 2013. [Online]. Available: https://supportforums.cisco.com/document/12013476/crypto-map-based-ipsec-vpn-fundamentals-negotiation-and-configuration. [Accessed: 04-Nov-2014].

[45] A. Salman, M. Rogawski, and J.-P. Kaps, “Efficient Hardware Accelerator for IPSec Based on Partial Reconfiguration on Xilinx FPGAs,” *2011 Int. Conf. Reconfigurable Comput. FPGAs*, pp. 242–248, Nov. 2011.

[46] S. Kent, “IP Authentication Header - RFC 4302,” *IETF*, 2005.

[47] S. Kent, “IP Encapsulating Security Payload (ESP) - RFC 4303,” *IETF*, 2005.

[48] R. Rosen, “Creating VPNs with IPsec and SSL/TLS,” *Linux J.*, 2008.

[49] D. Kirkland, “Ubuntu manuals: racoon.conf(5).” [Online]. Available: http://manpages.ubuntu.com/manpages/lucid/man5/racoon.conf.5.html. [Accessed: 20-Nov-2014].

[50] OCAU Wiki, “DebIPsec.” [Online]. Available: http://www.overclockers.com.au/wiki/DebIPsec. [Accessed: 20-Nov-2014].

[51] Dustin Kirkland, “Ubuntu manuals : ipsec.conf.5.” [Online]. Available: http://manpages.ubuntu.com/manpages/lucid/en/man5/ipsec.conf.5.html. [Accessed: 20-Nov-2014].

[52] I. The Cacti Group, “CACTI.” [Online]. Available: http://www.cacti.net/. [Accessed: 13-Nov-2014].

[53] D. Levi, R. Frye, S. Routhier, and B. Wijnen, “Coexistence between Version 1, Version 2, and Version 3 of the Internet-standard Network Management Framework Status - RFC 3584,” 2003.

[54] M. Carbone and L. Rizzo, “Dummynet revisited,” *ACM SIGCOMM Comput. Commun. Rev.*, vol. 40, no. 2, pp. 13–20, Apr. 2010.

[55] P. Pinto, “Asterisk Now 3.0 – Monte você mesmo uma Central Telefónica,” *pplware*, 2013. [Online]. Available: http://pplware.sapo.pt/tutoriais/networking/asterisk-now-3-0-monte-voce-mesmo-uma-central-telefonica/. [Accessed: 12-Oct-2014].

[56] C. Hinkle, “Installing AsteriskNOW,” 2014. [Online]. Available: https://wiki.asterisk.org/wiki/display/AST/Installing+AsteriskNOW. [Accessed: 11-Dec-2014].

[57] P. Belanger, “Asterisk Packages.” [Online]. Available: https://wiki.asterisk.org/wiki/display/AST/Asterisk+Packages. [Accessed: 11-Dec-2014].

[58] W. Mitchell, J. Criscuolo, E. Green, and J. Frazier, “Configure Asterisk For WebRTC.” [Online]. Available: http://sipjs.com/guides/server-configuration/asterisk/. [Accessed: 04-Jan-2015].

[59] Wikipedia, “Datagram Transport Layer Security,” 2014. [Online]. Available: http://en.wikipedia.org/wiki/Datagram_Transport_Layer_Security. [Accessed: 04-Jan-2015].

[60] Wikipedia, “Comparison of VoIP software.” [Online]. Available: http://en.wikipedia.org/wiki/Comparison_of_VoIP_software. [Accessed: 20-Nov-2014].

[61] M. Baugher and D. Wing, “Session Description Protocol (SDP) Security Descriptions for Media Streams - RFC 4568,” *IETF*, pp. 1–44, 2006.

[62] Cisco, “SIP Configuration Guide , Cisco IOS Release 15M&T,” *Cisco Syst.*, no. 6387, 2014.

Apêndices

Apêndice A. Programa em bash (voipp)

```
#!/bin/bash

# voipp segue a seguinte ordem ./voipp.sh [emissor(-e) ou recetor(-r) ] [ID_user] [UDP; TLS; IPSEC] [N chamadas] {[-dl;-pl;-bw] [valor<ms;0-1;Mbits/s>]]}

#retirar o IP
IP='0'
SO=`uname -s`
asterisk='193.136.9.110'
portaUDP='5060'
portaTLS='5061'

cert1002='keys/cliente1002.crt'
cert1003='keys/cliente1003.crt'
key1002='keys/cliente1002.pem'
key1003='keys/cliente1003.pem'


#descobrir o SO, porque no SO MAC a interface ethernet chama-se en0 e no Linux é eth0
if [ $SO == "Linux" ]
then
#    status0=`ifconfig eth0 2>/dev/null |awk '/inet addr:/ {print $2}' |sed 's/addr://'`
     status0=`ifconfig eth0 2>/dev/null |awk '/inet end.:/ {print $3}' |sed 's/end.://'` #root
     echo "IP_eth0:" $status0

     IPvazio=''
     if [ "$status0" == "$IPvazio" ]
     then
          #ir buscar o IP da wlan0
          status1=`ifconfig wlan0 2>/dev/null |awk '/inet addr:/ {print $2}'|sed 's/addr://'`
          #echo "IP_wlan0:"$status1

          if [ "$status1" == "$IPvazio" ]
          then
               #echo "nenhum IP disponível neste "$SO
```

```
                IP='0'
            else
                #existe IP em wlan0
                IP=$status1
            fi
        else
            #IP eth0 existe
            IP=$status0
        fi

    else
        if [ $SO == "Darwin" ]
        then
            status0=`/sbin/ifconfig en0| grep 'status: ' | awk '{print $2}'`

            case "$status0" in
              active)  IP=`/sbin/ifconfig en0| grep 'inet ' | awk '{print $2}'`
                ;;

              inactive) # IP do wifi
                status1=`/sbin/ifconfig en1| grep 'status: ' | awk '{print $2}'`

                case "$status1" in
                  active)
                    IP=`/sbin/ifconfig en1| grep 'inet ' | awk '{print $2}'`
                    ;;

                  inactive) #avisar o user que não há conectividade
                    IP='0'
                    echo "Não há conectividade para continuar.."
                    ;;

                  *) #valor lido incorrecto
                    IP='0'
                    echo "Há problemas com conectividade, não pode continuar..    "
                    ;;

                esac
                ;;

              *) #valor lido incorrecto
                IP='0'
                echo "Há problemas com conectividade, não pode continuar.."
                ;;
            esac
        fi
    fi
```

```
# verificar o conteúdo das variáveis
echo "IP="$IP
echo "E/R-"$1"."
echo "User-"$2"."
echo "transpo-"$3"."
echo "calls-"$4"."

vazio=''

if [[ $4 != $vazio ]] && [[ $IP != "0" ]];
then
    # ver se é emissor ou recetor
    case $1 in
        "-e") #ver qual é o user
            case $2 in
                "1002") #ver qual é o tipo de protocolo a usar
                    case $3 in
                        "UDP") #comandos do 1002 com UDP
                            case $5 in
                                "-dl") # inserir restrições de delay
                                    ipfw add pipe 10 ip from $IP to $asterisk out
                                    ipfw pipe 10 config delay $6ms
                                    ;;
                                "-bw") # inserir restrições de largura de banda
                                    ipfw add pipe 10 ip from $IP to $asterisk out
                                    ipfw pipe 10 config bw $6Mbits/s
                                    ;;
                                "-pl") # inserir restrições de pacotes perdidos
                                    ipfw add pipe 10 ip from $IP to $asterisk out
                                    ipfw pipe 10 config plr $6
                                    ;;
                                *) # fazer a experiência sem restrições
                                    ;;
                            esac

                            # fazer sempre o processo normal
                            sipp $asterisk:$portaUDP -sf registo.xml -inf 1002.csv -i $IP -p
$portaUDP -m 1 -l 1
                            sipp $asterisk:$portaUDP -sf emissor.xml -inf 1002.csv -i $IP -p
$portaUDP -m $4 -trace_stat -fd 2
                            # faz logout da sessão
                            sipp $asterisk:$portaUDP -sf sair.xml -inf 1002.csv -i $IP -p $portaUDP -
m 1 -l 1
                            ;;

                        "TLS") #comandos do 1002 com TLS
                            case $5 in
```

```
                    "-dl") # inserir restrições de delay
                        ipfw add pipe 10 ip from $IP to $asterisk out
                        ipfw pipe 10 config delay $6ms
                        ;;
                    "-bw") # inserir restrições de largura de banda
                        ipfw add pipe 10 ip from $IP to $asterisk out
                        ipfw pipe 10 config bw $6Mbits/s
                        ;;
                    "-pl") # inserir restrições de pacotes perdidos
                        ipfw add pipe 10 ip from $IP to $asterisk out
                        ipfw pipe 10 config plr $6
                        ;;
                    *) # fazer a experiência sem restrições
                        ;;
                esac

                # fazer sempre o processo normal
                sipp $asterisk:$portaTLS -sf registoTLS.xml -inf 1002TLS.csv -tls_cert
$cert1002 -tls_key $key1002 -t l1 -i $IP -p $portaTLS -m 1 -l 1

                sipp $asterisk:$portaTLS -sf emissorTLS.xml -inf 1002TLS.csv -tls_cert
$cert1002 -tls_key $key1002 -t l1 -i $IP -p $portaTLS -m $4 -trace_stat -fd 2

                # faz logout da sessão
                sipp $asterisk:$portaTLS -sf sairTLS.xml -inf 1002TLS.csv -tls_cert
$cert1002 -tls_key $key1002 -t l1 -i $IP -p $portaTLS -m 1 -l 1
                ;;

            "IPSEC") #comandos do 1002 com IPSec
                case $5 in
                    "-dl") # inserir restrições de delay
                        ipfw add pipe 10 ip from $IP to $asterisk out
                        ipfw pipe 10 config delay $6ms
                        ;;
                    "-bw") # inserir restrições de largura de banda
                        ipfw add pipe 10 ip from $IP to $asterisk out
                        ipfw pipe 10 config bw $6Mbits/s
                        ;;
                    "-pl") # inserir restrições de pacotes perdidos
                        ipfw add pipe 10 ip from $IP to $asterisk out
                        ipfw pipe 10 config plr $6
                        ;;
                    *) # fazer a experiência sem restrições
                        ;;
                esac

                ###### fazer sempre o processo normal
```

```
                        #só por questões de iniciação do racoon
                        if [ $SO == "Linux" ]
                        then
                            racoon start
                            setkey -f /etc/setkey.conf
                        else
                            racoon -f /etc/racoon.conf
                            setkey -f /etc/setkey.conf
                        fi

                        sipp $asterisk:$portaTLS -sf registo.xml -inf 1002.csv -tls_cert
$cert1002 -tls_key $key1002 -t l1 -i $IP -p $portaTLS -m 1 -l 1

                        sipp $asterisk:$portaTLS -sf emissor.xml -inf 1002.csv -tls_cert
$cert1002 -tls_key $key1002 -t l1 -i $IP -p $portaTLS -m $4 -trace_stat -fd 2

                        # faz logout da sessão
                        sipp $asterisk:$portaTLS -sf sair.xml -inf 1002.csv -tls_cert $cert1002
-tls_key $key1002 -t l1 -i $IP -p $portaTLS -m 1 -l 1

                        #só por questões da iniciação do racoon
                        if [ $SO == "Linux" ]
                        then
                            # finaliza o racoon no Linux
                            racoon stop
                            setkey -F -P
                        else
                            # finaliza o racoon no Mac OS
                            e=`ps -ef | grep racoon | awk '{print $2}'`
                            kill $e
                            setkey -F -P
                        fi
                        ;;

                    *)  echo "Protocolo de transporte desconhecido..."
                        ;;
                esac
                ;;
            "1003") #ver qual é o tipo de protocolo a usar
                case $3 in
                    "UDP") #comandos do 1003 com UDP
                        sipp $asterisk:$portaUDP -sf registo.xml -inf 1003.csv -i $IP -p
$portaUDP -m 1 -l 1

                        sipp $asterisk:$portaUDP -sf emissor.xml -inf 1003.csv -i $IP -p
$portaUDP -m $4 -trace_stat -fd 2
```

```
                        # faz logout da sessão
                        sipp $asterisk:$portaUDP -sf sair.xml -inf 1003.csv -i $IP -p $portaUDP -
m 1 -l 1
                        ;;

                    "TLS") #comandos do 1003 com TLS
                        sipp $asterisk:$portaTLS -sf registoTLS.xml -inf 1003TLS.csv -tls_cert
$cert1003 -tls_key $key1003 -t l1 -i $IP -p $portaTLS -m 1 -l 1

                        sipp $asterisk:$portaTLS -sf emissorTLS.xml -inf 1003TLS.csv -tls_cert
$cert1003 -tls_key $key1003 -t l1 -i $IP -p $portaTLS -m $4 -trace_stat -fd 2

                        # faz logout da sessão
                        sipp $asterisk:$portaTLS -sf sairTLS.xml -inf 1003TLS.csv -tls_cert
$cert1003 -tls_key $key1003 -t l1 -i $IP -p $portaTLS -m 1 -l 1
                        ;;

                    "IPSEC") #comandos do 1003 com IPSec
                        case $5 in
                            "-dl") # inserir restrições de delay
                                ipfw add pipe 10 ip from $IP to $asterisk out
                                ipfw pipe 10 config delay $6ms
                                ;;
                            "-bw") # inserir restrições de largura de banda
                                ipfw add pipe 10 ip from $IP to $asterisk out
                                ipfw pipe 10 config bw $6Mbits/s
                                ;;
                            "-pl") # inserir restrições de pacotes perdidos
                                ipfw add pipe 10 ip from $IP to $asterisk out
                                ipfw pipe 10 config plr $6
                                ;;
                            *) # fazer a experiência sem restrições
                                ;;
                        esac

                        ###### acitivar o ipsec
                        #só por questões da iniciação do racoon
                        if [ $SO == "Linux" ]
                        then
                            racoon start
                            setkey -f /etc/setkey.conf
                        else
                            racoon -f /etc/racoon.conf
                            setkey -f /etc/setkey.conf
                        fi

                        ###### fazer  o processo normal
```

```
                        sipp $asterisk:$portaUDP -sf registo.xml -inf 1003.csv -i $IP -p
$portaUDP -m 1 -l 1
                        sipp $asterisk:$portaUDP -sf emissor.xml -inf 1003.csv -i $IP -p
$portaUDP -m $4 -trace_stat -fd 2
                        sipp $asterisk:$portaUDP -sf sair.xml -inf 1003.csv -i $IP -p $portaUDP
-m 1 -l 1

                        #só por questões da iniciação do racoon
                        if [ $SO == "Linux" ]
                        then
                            # finaliza o racoon no Linux
                            racoon stop
                            setkey -F -P
                        else
                            # finaliza o racoon no Mac OS
                            e=`ps -ef | grep racoon | awk '{print $2}'`
                            kill $e
                            setkey -F -P
                        fi
                    ;;

                    *) echo "Protocolo de transporte desconhecido..."
                       ;;

                    esac
                ;;

            *) echo "Utilizador não reconhecido..."
               ;;
        esac
        ;;

    "-r") #ver qual é o user
        case $2 in
            "1002") #ver qual é o tipo de protocolo a usar
                case $3 in
                    "UDP") #comandos a executar pelo user 1002 com UDP
                        sipp $asterisk:$portaUDP -sf registo.xml -inf 1002.csv -i $IP -p
$portaUDP -m 1 -l 1
                        sipp $asterisk:$portaUDP -sf recetor.xml -inf 1002.csv -i $IP -p
$portaUDP -m $4 -trace_stat -fd 2
                        #-fd 2 seg de intervalo entre leituras

                        # faz logout da sessão
                        sipp $asterisk:$portaUDP -sf sair.xml -inf 1002.csv -i $IP -p $portaUDP -
m 1 -l 1
                        ;;
```

```
                    "TLS") #comandos a executar pelo user 1002 com TLS
                        #echo "(recetor)executar registo 1002 TLS com IP="$IP". e
numCalls="$4

                        sipp $asterisk:$portaTLS -sf registoTLS.xml -inf 1002TLS.csv -tls_cert
$cert1002 -tls_key $key1002 -i $IP -p $portaTLS -m 1 -l 1 -t l1
                        sipp $asterisk:$portaTLS -sf recetorTLS.xml -inf 1002TLS.csv -tls_cert
$cert1002 -tls_key $key1002 -i $IP -p $portaTLS -m $4 -t l1 -trace_stat -fd 2

                        # faz logout da sessão
                        sipp $asterisk:$portaTLS -sf sairTLS.xml -inf 1002TLS.csv -tls_cert
$cert1002 -tls_key $key1002 -i $IP -p $portaTLS -m 1 -l 1 -t l1
                        ;;

                    "IPSEC") ###### acitivar o ipsec
                        #só por questões da iniciação do racoon
                        if [ $SO == "Linux" ]
                        then
                            racoon start
                            setkey -f /etc/setkey.conf
                        else
                            racoon -f /etc/racoon.conf
                            setkey -f /etc/setkey.conf
                        fi

                        ###### fazer  o processo normal
                        sipp $asterisk:$portaUDP -sf registo.xml -inf 1002.csv -i $IP -p
$portaUDP -m 1 -l 1
                        sipp $asterisk:$portaUDP -sf recetor.xml -inf 1002.csv -i $IP -p
$portaUDP -m $4 -trace_stat -fd 2
                        sipp $asterisk:$portaUDP -sf sair.xml -inf 1002.csv -i $IP -p $portaUDP
-m 1 -l 1

                        #só por questões da iniciação do racoon
                        if [ $SO == "Linux" ]
                        then
                            # finaliza o racoon no Linux
                            racoon stop
                            setkey -F -P
                        else
                            # finaliza o racoon no Mac OS
                            e=`ps -ef | grep racoon | awk '{print $2}'`
                            kill $e
                            setkey -F -P
                        fi
                    ;;
```

```
                *)  echo "Protocolo de transporte desconhecido..."
                    ;;
            esac
            ;;

        "1003") #ver qual é o tipo de protocolo a usar
            case $3 in
                "UDP") #comandos a executar pelo user 1003 com UDP
                    sipp $asterisk:$portaUDP -sf registo.xml -inf 1003.csv -i $IP -p
$portaUDP -m 1 -l 1
                    sipp $asterisk:$portaUDP -sf recetor.xml -inf 1003.csv -i $IP -p
$portaUDP -m $4 -trace_stat -fd 2
                    # faz logout da sessão
                    sipp $asterisk:$portaUDP -sf sair.xml -inf 1003.csv -i $IP -p $portaUDP -
m 1 -l 1
                    ;;

                "TLS") #comandos a executar pelo user 1003 com TLS
                    sipp $asterisk:$portaTLS -sf registoTLS.xml -inf 1003TLS.csv -tls_cert
$cert1003 -tls_key $key1003 -i $IP -p $portaTLS -m 1 -l 1 -t l1
                    sipp $asterisk:$portaTLS -sf recetorTLS.xml -inf 1003TLS.csv -tls_cert
$cert1003 -tls_key $key1003 -i $IP -p $portaTLS -m $4 -t l1 -trace_stat -fd 2

                    # faz logout da sessão
                    sipp $asterisk:$portaTLS -sf sairTLS.xml -inf 1003TLS.csv -tls_cert
$cert1003 -tls_key $key1003 -i $IP -p $portaTLS -m 1 -l 1 -t l1
                    ;;

                "IPSEC") ###### acitivar o ipsec
                    #só por questões da iniciação do racoon
                    if [ $SO == "Linux" ]
                    then
                        racoon start
                        setkey -f /etc/setkey.conf
                    else
                        racoon -f /etc/racoon.conf
                        setkey -f /etc/setkey.conf
                    fi

                    ###### fazer  o processo normal
                    sipp $asterisk:$portaUDP -sf registo.xml -inf 1003.csv -i $IP -p
$portaUDP -m 1 -l 1
                    sipp $asterisk:$portaUDP -sf recetor.xml -inf 1003.csv -i $IP -p
$portaUDP -m $4 -trace_stat -fd 2
                    sipp $asterisk:$portaUDP -sf sair.xml -inf 1003.csv -i $IP -p $portaUDP
-m 1 -l 1
```

```
                        #só por questões da iniciação do racoon
                        if [ $SO == "Linux" ]
                        then
                            # finaliza o racoon no Linux
                            racoon stop
                            setkey -F -P
                        else
                            # finaliza o racoon no Mac OS
                            e=`ps -ef | grep racoon | awk '{print $2}'`
                            kill $e
                            setkey -F -P
                        fi
                    ;;

                *)  echo "Protocolo de transporte desconhecido..."
                    ;;
            esac
            ;;
        *)  echo "Utilizador não reconhecido..."
            ;;
    esac
    ;;
*) echo "Não foi especificado nenhum tipo de ligação..."
    ;;
esac
else
    echo "Número de chamadas não aceite ou sem conectividade..."
fi
```

Apêndice B. registo.xml

```
<?xml version="1.0" encoding="iso-8859-2" ?>
<!DOCTYPE scenario SYSTEM "sipp.dtd">

<scenario name="registo">

  <send retrans="500">
    <![CDATA[

      REGISTER sip:[field1]:[field4] SIP/2.0
      Via: SIP/2.0/[transport] [local_ip]:[field3];branch=[branch]
      From: <sip:[field0]@[local_ip]:[field3]>;tag=[call_number]
      To: <sip:[field0]@[field1]:[field3]>
      Call-ID: [call_id]
```

```
        CSeq: [cseq] REGISTER
        Contact: <sip:[field0]@[local_ip]:[field3];transport=[transport]>
        Max-Forwards: 100
        Expires: 3600
        User-Agent: SIPp
        Content-Length: 0

    ]]>
</send>

<recv response="100" optional="true">
</recv>

<recv response="401" auth="true" rtd="true">
</recv>

<send retrans="500">
    <![CDATA[

        REGISTER sip:[field1]:[field4] SIP/2.0
        Via: SIP/2.0/[transport] [local_ip]:[field3];branch=[branch]
        From: <sip:[field0]@[local_ip]:[field3]>;tag=[call_number]
        To: <sip:[field0]@[field1]:[field4]>
        Call-ID: [call_id]
        CSeq: [cseq] REGISTER
        Contact: <sip:[field0]@[local_ip]:[field3];transport=[transport]>
        [field2]
        Max-Forwards: 100
        Expires: 3600
        User-Agent: SIPp
        Content-Length: 0

    ]]>
</send>

<recv response="100" optional="true">
</recv>

<recv response="200">
</recv>

<!– definition of the response time repartition table (unit is ms)   –>
<ResponseTimeRepartition value="10, 20, 30, 40, 50, 100, 150, 200"/>

<!– definition of the call length repartition table (unit is ms)     –>
<CallLengthRepartition value="10, 50, 100, 500, 1000, 5000, 10000"/>

  </scenario>
```

Apêndice C. recetor.xml

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE scenario SYSTEM "sipp.dtd">

<scenario name="recetor">

  <recv request="INVITE" crlf="true">
  </recv>

  <send>
    <![CDATA[

      SIP/2.0 100 Trying
      [last_Via:]
      [last_From:]
      [last_To:];tag=[pid]SIPpTag01[call_number]
      [last_Call-ID:]
      [last_CSeq:]
      Contact: <sip:[local_ip]:[field3];transport=[transport]>
      Content-Length: 0

    ]]>
  </send>

  <send>
    <![CDATA[

      SIP/2.0 180 Ringing
      [last_Via:]
      [last_From:]
      [last_To:];tag=[pid]SIPpTag01[call_number]
      [last_Call-ID:]
      [last_CSeq:]
      Contact: <sip:[local_ip]:[field4];transport=[transport]>
      Content-Length: 0

    ]]>
  </send>

  <send retrans="500">
    <![CDATA[

      SIP/2.0 200 OK
      [last_Via:]
      [last_From:]
      [last_To:];tag=[pid]SIPpTag01[call_number]
      [last_Call-ID:]
      [last_CSeq:]
      Contact: <sip:[local_ip]:[field3];transport=[transport]>
```

```
        Content-Type: application/sdp
        Content-Length: [len]

        v=0
        o=[field0] 53655765 2353687637 IN IP[local_ip_type] [local_ip]
        s=-
        c=IN IP[media_ip_type] [media_ip]
        t=0 0
        m=audio [media_port] RTP/AVP 0
        a=rtpmap:0 PCMU/8000

    ]]>
  </send>

  <recv request="ACK|UPDATE" regexp_match="true" rtd="true">
  </recv>

  <pause>
  </pause>

  <recv request="BYE">
  </recv>

  <send>
    <![CDATA[

        SIP/2.0 200 OK
        [last_Via:]
        [last_From:]
        [last_To:]
        [last_Call-ID:]
        [last_CSeq:]
        Contact: <sip:[local_ip]:[field3];transport=[transport]>
        Content-Length: 0

    ]]>
  </send>

  <!– para o caso de 200 se perder e se se receber um BYE enviar o OK  –>
  <timewait milliseconds="4000"/>

  <!– definition of the response time repartition table (unit is ms)  –>
  <ResponseTimeRepartition value="10, 20, 30, 40, 50, 100, 150, 200"/>

  <!– definition of the call length repartition table (unit is ms)  –>
  <CallLengthRepartition value="10, 50, 100, 500, 1000, 5000, 10000"/>

</scenario>
```

Apêndice D. emissor.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE scenario SYSTEM "sipp.dtd">

<scenario name="emissor">

  <send retrans="500">
    <![CDATA[

      INVITE sip:[field5]@[field1]:[field4] SIP/2.0
      Via: SIP/2.0/[transport] [local_ip]:[field3];branch=[branch]
      From: <sip:[field0]@[local_ip]:[field3]>;tag=[pid]SIPpTag00[call_number]
      To: <sip:[field5]@[field1]:[field4]>
      Call-ID: [call_id]
      CSeq: 1 INVITE
      Contact: <sip:[field0]@[local_ip]:[field3];transport=[transport]>
      Max-Forwards: 70
      Content-Type: application/sdp
      Content-Length: [len]

      v=0
      o=[field0] 53655765 2353687637 IN IP[local_ip_type] [local_ip]
      s=-
      c=IN IP[media_ip_type] [media_ip]
      t=0 0
      m=audio [media_port] RTP/AVP 0
      a=rtpmap:0 PCMU/8000

    ]]>
  </send>

  <!– aguarda pela mensagem "TRYING"–>
  <recv response="100" optional="true">
  </recv>

  <recv response="401" auth="true">
  </recv>

  <send>
    <![CDATA[

      ACK sip:[field5]@[field1]:[field4] SIP/2.0
      Via: SIP/2.0/[transport] [local_ip]:[field3];branch=[branch]
      From: <sip:[field0]@[local_ip]:[field4]>;tag=[pid]SIPpTag00[call_number]
      To: <sip:[field5]@[field1]:[field4]>[peer_tag_param]
      Call-ID: [call_id]
      CSeq: 1 ACK
      Contact: <sip:[field0]@[local_ip]:[field3];transport=[transport]>
      Max-Forwards: 70
```

```
        Content-Length: 0

    ]]>
</send>


<!– envia a mensagem "INVITE com authentication"–>
<send retrans="500">
    <![CDATA[

        INVITE sip:[field5]@[field1]:[field4] SIP/2.0
        Via: SIP/2.0/[transport] [local_ip]:[field3];branch=[branch]
        From: <sip:[field0]@[local_ip]:[field3]>;tag=[pid]SIPpTag00[call_number]
        To: <sip:[field5]@[field1]:[field4]>
        Call-ID: [call_id]
        CSeq: 2 INVITE
        Contact: <sip:[field0]@[local_ip]:[field3];transport=[transport]>
        [field2]
        Max-Forwards: 70
        Content-Type: application/sdp
        Content-Length: [len]

        v=0
        o=[field0] 53655765 2353687637 IN IP[local_ip_type] [local_ip]
        s=-
        c=IN IP[media_ip_type] [media_ip]
        t=0 0
        m=audio [media_port] RTP/AVP 0
        a=rtpmap:0 PCMU/8000

    ]]>
</send>

<!– aguarda pela mensagem "TRYING"–>
<recv response="100" optional="true">
</recv>

<!– aguarda pela mensagem "RINGING"–>
<recv response="180" optional="true">
</recv>

<!– aguarda pela mensagem "OK">–>
<recv response="200" rtd="true" crlf="true">
</recv>

<send>
    <![CDATA[

        ACK sip:[field5]@[field1]:[field3] SIP/2.0
        Via: SIP/2.0/[transport] [local_ip]:[field4];branch=[branch]
        From: <sip:[field0]@[local_ip]:[field3]>;tag=[pid]SIPpTag00[call_number]
```

```
        To: <sip:[field5]@[field1]:[field3]>[peer_tag_param]
        Call-ID: [call_id]
        CSeq: 2 ACK
        Contact: <sip:[field0]@[local_ip]:[field3];transport=[transport]>
        Max-Forwards: 70
        Content-Length: 0

    ]]>
</send>

<!– envio dos pacotes RTP - envio da mensagem –>
<nop>
    <action>
        <exec play_pcap_audio="/voipp/pcap/rtp.pcap" />
    </action>
</nop>

<!– tempo de cada chamada é 18,98seg, ficando os 20 seg–>
<pause milliseconds="21000"/>

<!– <pause milliseconds="1000"/> –>

<send retrans="500">
    <![CDATA[

        BYE sip:[field5]@[field1]:[field4] SIP/2.0
        Via: SIP/2.0/[transport] [local_ip]:[field4];branch=[branch]
        From: <sip:[field0]@[local_ip]:[field4]>;tag=[pid]SIPpTag00[call_number]
        To: <sip:[field5]@[field1]:[field4]>[peer_tag_param]
        Call-ID: [call_id]
        CSeq: 2 BYE
        Contact: <sip:[field0]@[local_ip]:[field3];transport=[transport]>
        Max-Forwards: 70
        Content-Length: 0

    ]]>
</send>

<!– aguarda pela mensagem "OK"–>
<recv response="200" crlf="true"> </recv>

<!– Estatisticas –>
<!– repartição em tempo de resposta–>
<ResponseTimeRepartition value="10, 20, 30, 40, 50, 100, 150, 200"/>

<!– repartição em comprimento de chamada–>
<CallLengthRepartition value="10, 50, 100, 500, 1000, 5000, 10000"/>

    </scenario>
```

Apêndice E. sair.xml

```
<?xml version="1.0" encoding="iso-8859-2" ?>
<!DOCTYPE scenario SYSTEM "sipp.dtd">

<scenario name="sair">

  <send retrans="500">
    <![CDATA[

      REGISTER sip:[field1]:[field4] SIP/2.0
      Via: SIP/2.0/[transport] [local_ip]:[field3];branch=[branch]
      From: <sip:[field0]@[local_ip]:[field3]>;tag=[call_number]
      To: <sip:[field0]@[field1]:[field3]>
      Call-ID: [call_id]
      CSeq: [cseq] REGISTER
      Contact: <sip:[field0]@[local_ip]:[field3];transport=[transport]>
      Max-Forwards: 100
      Expires: 0
      User-Agent: SIPp
      Content-Length: 0

    ]]>
  </send>

  <recv response="100" optional="true">
  </recv>

  <recv response="401" auth="true" rtd="true">
  </recv>

  <send retrans="500">
    <![CDATA[

      REGISTER sip:[field1]:[field4] SIP/2.0
      Via: SIP/2.0/[transport] [local_ip]:[field3];branch=[branch]
      From: <sip:[field0]@[local_ip]:[field3]>;tag=[call_number]
      To: <sip:[field0]@[field1]:[field4]>
      Call-ID: [call_id]
      CSeq: [cseq] REGISTER
      Contact: <sip:[field0]@[local_ip]:[field3];transport=[transport]>
      [field2]
      Max-Forwards: 100
      Expires: 0
      User-Agent: SIPp
      Content-Length: 0

    ]]>
  </send>
```

```
    <recv response="100" optional="true">
    </recv>

    <recv response="200">
    </recv>


    <!– definition of the response time repartition table (unit is ms)   –>
    <ResponseTimeRepartition value="10, 20, 30, 40, 50, 100, 150, 200"/>

    <!– definition of the call length repartition table (unit is ms)     –>
    <CallLengthRepartition value="10, 50, 100, 500, 1000, 5000, 10000"/>

      </scenario>
```

Apêndice F. registoTLS.xml

```
<?xml version="1.0" encoding="iso-8859-2" ?>
<!DOCTYPE scenario SYSTEM "sipp.dtd">

<scenario name="registo com TLS">

    <send retrans="500">
      <![CDATA[

          REGISTER sips:[field1]:[field4] SIP/2.0
          Via: SIP/2.0/[transport] [local_ip]:[field3];branch=[branch]
          From: <sips:[field0]@[local_ip]>;tag=[call_number]
          To: <sips:[field0]@[field1]>
          Call-ID: [call_id]
          CSeq: [cseq] REGISTER
          Contact: <sips:[field0]@[local_ip]:[field3];transport=[transport]>
          Max-Forwards: 70
          Expires: 3600
          User-Agent: SIPp
          Content-Length: 0

      ]]>
    </send>

    <recv response="100" optional="true">
    </recv>

    <recv response="401" auth="true" rtd="true">
    </recv>
```

```
  <send retrans="500">
    <![CDATA[

      REGISTER sips:[field1]:[field4] SIP/2.0
      Via: SIP/2.0/[transport] [local_ip]:[field3];branch=[branch]
      From: <sips:[field0]@[local_ip]>;tag=[call_number]
      To: <sips:[field0]@[field1]>
      Call-ID: [call_id]
      CSeq: [cseq] REGISTER
      Contact: <sips:[field0]@[local_ip]:[field3];transport=[transport]>
      [field2]
      Max-Forwards: 70
      Expires: 3600
      User-Agent: SIPp
      Content-Length: 0

    ]]>
  </send>

  <recv response="100" optional="true">
  </recv>

  <recv response="200">
  </recv>

  <!– definition of the response time repartition table (unit is ms)   –>
  <ResponseTimeRepartition value="10, 20, 30, 40, 50, 100, 150, 200"/>

  <!– definition of the call length repartition table (unit is ms)     –>
  <CallLengthRepartition value="10, 50, 100, 500, 1000, 5000, 10000"/>

</scenario>
```

Apêndice G. recetorTLS.xml

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE scenario SYSTEM "sipp.dtd">

<scenario name="recetor com TLS">

  <recv request="INVITE" crlf="true" rrs="true">
  </recv>

  <send>
    <![CDATA[
```

```
        SIP/2.0 100 Trying
        [last_Via:]
        [last_From:]
        [last_To:];tag=[pid]SIPpTag01[call_number]
        [last_Call-ID:]
        [last_CSeq:]
        Contact: <sip:[field5]@[local_ip]:[field3];transport=[transport]>
        Content-Length: 0

    ]]>
</send>

<send>
    <![CDATA[

        SIP/2.0 180 Ringing
        [last_Via:]
        [last_From:]
        [last_To:];tag=[pid]SIPpTag01[call_number]
        [last_Call-ID:]
        [last_CSeq:]
        Contact: <sips:[field0]@[local_ip]:[field4];transport=[transport]>
        Content-Length: 0

    ]]>
</send>

<send retrans="500">
    <![CDATA[

          SIP/2.0 200 OK
        [last_Via:]
        [last_From:]
        [last_To:];tag=[pid]SIPpTag01[call_number]
        [last_Call-ID:]
        [last_CSeq:]
        Contact: <sips:[field0]@[local_ip]:[field3];transport=[transport]>
        Content-Type: application/sdp
        Content-Length: [len]

        v=0
        o=[field0] 53655765 2353687637 IN IP[local_ip_type] [local_ip]
        s=-
        c=IN IP[media_ip_type] [media_ip]
        t=0 0
        m=audio [media_port] RTP/SAVP 0
        a=rtpmap:0 PCMU/8000
        a=sendrecv
        a=rtpmap:101 telephone-event/8000
        a=fmtp:101 0-15
        a=crypto:1 AES_CM_128_HMAC_SHA1_80
```

```
inline:mmolMni7tv32bHMwvVDJo9O/FkzgU4sBh87w7hlu

        ]]>
    </send>

    <recv request="ACK" rtd="true">
    </recv>


    <recv request="BYE">
    </recv>

    <send>
        <![CDATA[

            SIP/2.0 200 OK
            [last_Via:]
            [last_From:]
            [last_To:]
            [last_Call-ID:]
            [last_CSeq:]
            Contact: <sips:[field0]@[local_ip]:[field3];transport=[transport]>
            Content-Length: 0

        ]]>
    </send>

    <!– para o caso de 200 se perder e se se receber um BYE enviar o OK   –>
    <timewait milliseconds="4000"/>

    <!– definition of the response time repartition table (unit is ms)   –>
    <ResponseTimeRepartition value="10, 20, 30, 40, 50, 100, 150, 200"/>

    <!– definition of the call length repartition table (unit is ms)     –>
    <CallLengthRepartition value="10, 50, 100, 500, 1000, 5000, 10000"/>

</scenario>
```

Apêndice H. emissorTLS.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE scenario SYSTEM "sipp.dtd">

<scenario name="emissorTLS">

    <send retrans="500">
        <![CDATA[
```

```
        INVITE sips:[field5]@[field1]:[field4] SIP/2.0
        Via: SIP/2.0/[transport] [local_ip]:[field3];branch=[branch]
        From: <sips:[field0]@[local_ip]:[field3]>;tag=[pid]SIPpTag00[call_number]
        To: <sips:[field5]@[field1]:[field4]>
        Call-ID: [call_id]
        CSeq: 1 INVITE
        Contact: <sips:[field0]@[local_ip]:[field3];transport=[transport]>
        Max-Forwards: 70
        Content-Type: application/sdp
        Content-Length: [len]

        v=0
        o=[field0] 53655765 2353687637 IN IP[local_ip_type] [local_ip]
        s=-
        c=IN IP[media_ip_type] [media_ip]
        t=0 0
        m=audio [media_port] RTP/SAVP 0 101
        a=rtpmap:0 PCMU/8000
        a=sendrecv
        a=rtpmap:101 telephone-event/8000
        a=fmtp:101 0-15
        a=crypto:1 AES_CM_128_HMAC_SHA1_80
inline:mmolMni7tv32bHMwvVDJo9O/FkzgU4sBh87w7hlu

    ]]>
  </send>

  <!– aguarda pela mensagem "TRYING"–>
  <recv response="100" optional="true">
  </recv>

  <recv response="401" auth="true">
  </recv>

  <send>
    <![CDATA[

        ACK sips:[field5]@[field1]:[field4] SIP/2.0
        Via: SIP/2.0/[transport] [local_ip]:[field3];branch=[branch]
        From: <sips:[field0]@[local_ip]:[field4]>;tag=[pid]SIPpTag00[call_number]
        To: <sips:[field5]@[field1]:[field4]>[peer_tag_param]
        Call-ID: [call_id]
        CSeq: 1 ACK
        Contact: <sips:[field0]@[local_ip]:[field3];transport=[transport]>
        Max-Forwards: 70
        Content-Length: 0

    ]]>
  </send>
```

```
    <!– envia a mensagem "INVITE com authentication"–>
    <send retrans="500">
      <![CDATA[

        INVITE sips:[field5]@[field1]:[field4] SIP/2.0
        Via: SIP/2.0/[transport] [local_ip]:[field3];branch=[branch]
        From: <sips:[field0]@[local_ip]:[field3]>;tag=[pid]SIPpTag00[call_number]
        To: <sips:[field5]@[field1]:[field4]>
        Call-ID: [call_id]
        CSeq: 2 INVITE
        Contact: <sips:[field0]@[local_ip]:[field3];transport=[transport]>
        [field2]
        Max-Forwards: 70
        Content-Type: application/sdp
        Content-Length: [len]

        v=0
        o=[field0] 53655765 2353687637 IN IP[local_ip_type] [local_ip]
        s=-
        c=IN IP[media_ip_type] [media_ip]
        t=0 0
        m=audio [media_port] RTP/SAVP 0 101
        a=rtpmap:0 PCMU/8000
        a=sendrecv
        a=rtpmap:101 telephone-event/8000
        a=fmtp:101 0-15
        a=crypto:1 AES_CM_128_HMAC_SHA1_80
inline:mmolMni7tv32bHMwvVDJo9O/FkzgU4sBh87w7hlu

      ]]>
    </send>

    <!– aguarda pela mensagem "TRYING"–>
    <recv response="100" optional="true">
    </recv>

    <!– aguarda pela mensagem "RINGING"–>
    <recv response="180" optional="true">
    </recv>

    <!– aguarda pela mensagem "OK">–>
    <recv response="200" rtd="true" crlf="true">
    </recv>

    <send>
      <![CDATA[

        ACK sips:[field5]@[field1]:[field3] SIP/2.0
        Via: SIP/2.0/[transport] [local_ip]:[field4];branch=[branch]
        From: <sips:[field0]@[local_ip]:[field3]>;tag=[pid]SIPpTag00[call_number]
        To: <sips:[field5]@[field1]:[field3]>[peer_tag_param]
```

```
        Call-ID: [call_id]
        CSeq: 2 ACK
        Contact: <sips:[field0]@[local_ip]:[field3];transport=[transport]>
        Max-Forwards: 70
        Content-Length: 0

    ]]>
  </send>

  <!– envio dos pacotes RTP - envio da mensagem –>
  <nop>
    <action>
      <exec play_pcap_audio="/voipp/pcap/rtp.pcap" />
    </action>
  </nop>

  <!– tempo de cada chamada é 18,98seg, ficando os 20 seg–>
  <pause milliseconds="21000"/>

  <!– <pause milliseconds="1000"/> –>

  <send retrans="500">
    <![CDATA[

      BYE sips:[field5]@[field1]:[field4] SIP/2.0
      Via: SIP/2.0/[transport] [local_ip]:[field4];branch=[branch]
      From: <sips:[field0]@[local_ip]:[field4]>;tag=[pid]SIPpTag00[call_number]
      To: <sips:[field5]@[field1]:[field4]>[peer_tag_param]
      Call-ID: [call_id]
      CSeq: 2 BYE
      Contact: <sips:[field0]@[local_ip]:[field3];transport=[transport]>
      Max-Forwards: 70
      Content-Length: 0

    ]]>
  </send>

  <!– aguarda pela mensagem "OK"–>
  <recv response="200" crlf="true"> </recv>



  <!– Estatisticas –>
  <!– repartição em tempo de resposta–>
  <ResponseTimeRepartition value="10, 20, 30, 40, 50, 100, 150, 200"/>

  <!– repartição em comprimento de chamada–>
  <CallLengthRepartition value="10, 50, 100, 500, 1000, 5000, 10000"/>

</scenario>
```

Apêndice I. sairTLS.xml

```
<?xml version="1.0" encoding="iso-8859-2" ?>
<!DOCTYPE scenario SYSTEM "sipp.dtd">

<scenario name="sair com TLS">

  <send retrans="500">
    <![CDATA[

      REGISTER sips:[field1]:[field4] SIP/2.0
      Via: SIP/2.0/[transport] [local_ip]:[field3];branch=[branch]
      From: <sips:[field0]@[local_ip]:[field3]>;tag=[call_number]
      To: <sips:[field0]@[field1]:[field3]>
      Call-ID: [call_id]
      CSeq: [cseq] REGISTER
      Contact: <sips:[field0]@[local_ip]:[field3];transport=[transport]>
      Max-Forwards: 100
      Expires: 0
      User-Agent: SIPp
      Content-Length: 0

    ]]>
  </send>

  <recv response="100" optional="true">
  </recv>

  <recv response="401" auth="true" rtd="true">
  </recv>

  <send retrans="500">
    <![CDATA[

      REGISTER sips:[field1]:[field4] SIP/2.0
      Via: SIP/2.0/[transport] [local_ip]:[field3];branch=[branch]
      From: <sips:[field0]@[local_ip]:[field3]>;tag=[call_number]
      To: <sips:[field0]@[field1]:[field4]>
      Call-ID: [call_id]
      CSeq: [cseq] REGISTER
      Contact: <sips:[field0]@[local_ip]:[field3];transport=[transport]>
      [field2]
      Max-Forwards: 100
      Expires: 0
      User-Agent: SIPp
      Content-Length: 0

    ]]>
  </send>
```

```
<recv response="100" optional="true">
</recv>

<recv response="200">
</recv>


<!– definition of the response time repartition table (unit is ms)   –>
<ResponseTimeRepartition value="10, 20, 30, 40, 50, 100, 150, 200"/>

<!– definition of the call length repartition table (unit is ms)     –>
<CallLengthRepartition value="10, 50, 100, 500, 1000, 5000, 10000"/>

  </scenario>
```