MINISTÉRIO DA DEFESA
EXÉRCITO BRASILEIRO
DEPARTAMENTO DE CIÊNCIA E TECNOLOGIA
INSTITUTO MILITAR DE ENGENHARIA
CURSO DE GRADUAÇÃO EM ENGENHARIA DE COMUNICAÇÕES

CAP CLAISSO PIRES AZZOLIN
1º Ten CAIO SILVA MARTINS DE OLIVEIRA

DESENVOLVIMENTO DE APLICATIVO PARA TRANSMISSÃO DE VOZ ATRAVÉS DE DISPOSITIVOS MOVÉIS

Rio de Janeiro
2016

INSTITUTO MILITAR DE ENGENHARIA

CAP CLAISSO PIRES AZZOLIN
1º Ten CAIO SILVA MARTINS DE OLIVEIRA

DESENVOLVIMENTO DE APLICATIVO PARA TRANSMISSÃO DE VOZ ATRAVÉS DE DISPOSITIVOS MOVÉIS

Relatório de Projeto de Final de Curso apresentado ao Curso de Graduação em Engenharia de Comunicações do Instituto Militar de Engenharia.

Orientador: Maj Vítor Gouvêa Andrezo Carneiro – D.C.
Co-orientador: Maj André Luis Souza de Araújo - M.C.

Rio de Janeiro
2016

2016

INSTITUTO MILITAR DE ENGENHARIA

Praça General Tibúrcio, 80 – Praia Vermelha

Rio de Janeiro – RJ CEP: 22290-270

004.6 Azzolin, Claisso Pires

A999d Desenvolvimento de aplicativo para transmissão de voz através de dispositivos móveis / Claisso Pires Azzolin, Caio Silva Martins de Oliveira; orientados por Vitor Gouvêa Andrezo Carneiro; André Luis Souza de Araújo – Rio de Janeiro: Instituto Militar de Engenharia, 2016.

52p. : il.

Projeto de Fim de Curso (PROFIC) – Instituto Militar de Engenharia, Rio de Janeiro, 2016.

1. Curso de Engenharia de Comunicações – Projeto de Fim de Curso. 2. Etransmissão de voz. 3. Dispositivos móveis. I.Oliveira, Caio Silva Martins de. II. Carneiro, Vitor Couvêa Andrezo. III. Araújo, André Luis Souza de. IV. Título. V. Instituto Militar de Engenharia.

INSTITUTO MILITAR DE ENGENHARIA

Cap CLAISSO PIRES AZZOLIN
1º Ten CAIO SILVA MARTINS DE OLIVEIRA

DESENVOLVIMENTO DE APLICATIVO PARA TRANSMISSÃO DE VOZ ATRAVÉS DE DISPOSITIVOS MÓVEIS

Projeto de Final de Curso apresentado ao Curso de Graduação em Engenharia de Comunicações do Instituto Militar de Engenharia.

Orientador: Maj Vítor Gouvêa Andrezo Carneiro – D.C

Co-orientador: Maj André Luis Souza de Araújo – M.C.

Aprovada em 7 de Outubro de 2016 pela seguinte Banca Examinadora:

Vitor Gouvêa Andrezo Carneiro

Prof. Vítor Gouvêa Andrezo Carneiro – D.C. do IME

Prof. André Luis Souza de Araújo – M.C. do IME

Alexandre Amorim P Júnior

Prof. Alexandre Amorim Pereira Júnior – M.C. do IME

Rio de Janeiro
2016

SUMÁRIO

LISTA DE FIGURAS

LISTA DE SIGLAS

3G	3ª Geração
ACK	Acknowledgement
ALSA	Advanced Linux Sound Architecture
API	Application Programming Interface
ARPA	Advanced Research Projects Agency
ATA	Adaptador Telefônico Analógico
CoDec	Codificador/Decodificador
DHCP	Dynamic Host Configuration Protocol
DNS	Domain Name System
DVD	Digital Video Disc
EUA	Estados Unidos da América
GCC	GNU Compiler Collection
HD	Hard Disc
HTTP	Hyper Text Transfer Protocol
iOS	iPhone Operating System
MATLAB	Matrix Laboratory
OSI	Open Systems Interconnection
P2P	Peer-to-Peer
RSA	Rivest, Shamir e Adleman
SIP	Session Initiation Protocol
SMTP	Simple Mail Transfer Protocol
SNMP	Simple Network Management Protocol
SNR	Signal-to-Noise Ratio

SO	Sistema Operacional
SSL	Secure Sockets Layer
SVA	Serviço de Valor Adicionado
SYN	Synchronize
TCP/IP	Transmission Control Protocol/Internet Protocol
TFTP	Trivial File Transfer Protocol
TLS	Transport Layer Security
UA	User Agent
UAC	User Agent Client
UAS	User Agent Server
UDP	User Datagram Protocol
VDI	VirtualBox Disk Image
VoIP	Voice over Internet Protocol
WWW	World Wide Web

RESUMO

Os meios de comunicação são imprescindíveis, sejam eles internet, rádio, televisão, jornal ou qualquer um dos outros meios. Através deles que pode-se obter informações em tempo real sobre acontecimentos no mundo todo, independente da distância, além de possibilitar a comunicação entre pessoas.

A internet é o meio com maior destaque atualmente devido a sua ampla área de abrangência. Além de ser usada para transmitir informação, seja em jornais online, blogs e outros, ou para entretenimento com filmes, jogos, canais de televisão, ela também vem sendo cada vez mais usada para comunicação entre usuários. A internet permite comunicação através de troca de mensagens instantâneas, fotos, áudios e, o mais recente, realizar ligações através de aplicativos que utilizam *Voice over Internet Protocol.*

A tecnologia VoIP, apesar de nova, século XXI, vem ganhando mercado e substituindo a telefonia convencional devido as suas várias vantagens. Dentre essas a mais impactante é a economia de recursos, paga-se muito menos para realizar ligações. Não substituiu completamente a rede de telefonia convencional, pois além do legado deixado, a mesma continua mais confiável em relação à estabilidade. O VoIP depende de uma boa qualidade nos serviços de internet para funcionar corretamente.

Visando estudar o VoIP, este projeto tem por finalidade desenvolver um programa em linguagem C/C++ para estabelecer comunicação entre 2 computadores pertencentes a uma mesma rede, seja ponto-a-ponto ou estruturada. Foi feito um estudo inicial sobre os aplicativos e programas que utilizam VoIP para se comunicar a fim de entender um pouco desta técnica. Também foi usado como base um trabalho já feito neste Instituto, o qual foi referenciado, e a partir disso foi desenvolvido um código para que fosse possível gravar e enviar um arquivo de áudio durante a execução do programa. Foram feitos testes, gravando e enviando vários arquivos de áudio, além de testes comparando os arquivos de áudio com e sem criptografia, antes do mesmo ser enviado, e os resultados foram expostos ao longo deste trabalho.

Palavras-chave: VoIP, Internet, Cliente, Servidor, Socket, Redes.

1. Introdução

Esse projeto tem como objetivo o desenvolvimento de um aplicativo para transmissão de voz através de computadores, utilizando programação em linguagem C/C++, produzido e testado na plataforma Linux. Para isso, serão apresentados conceitos e exemplos como VoIP, *Transmission Control Protocol/Internet Protocol, User Datagram Protocol*, codificação, criptografia, programas já consolidados os quais utilizam VoIP, além de apresentar o código, em linguagem C, desenvolvido durante o projeto.

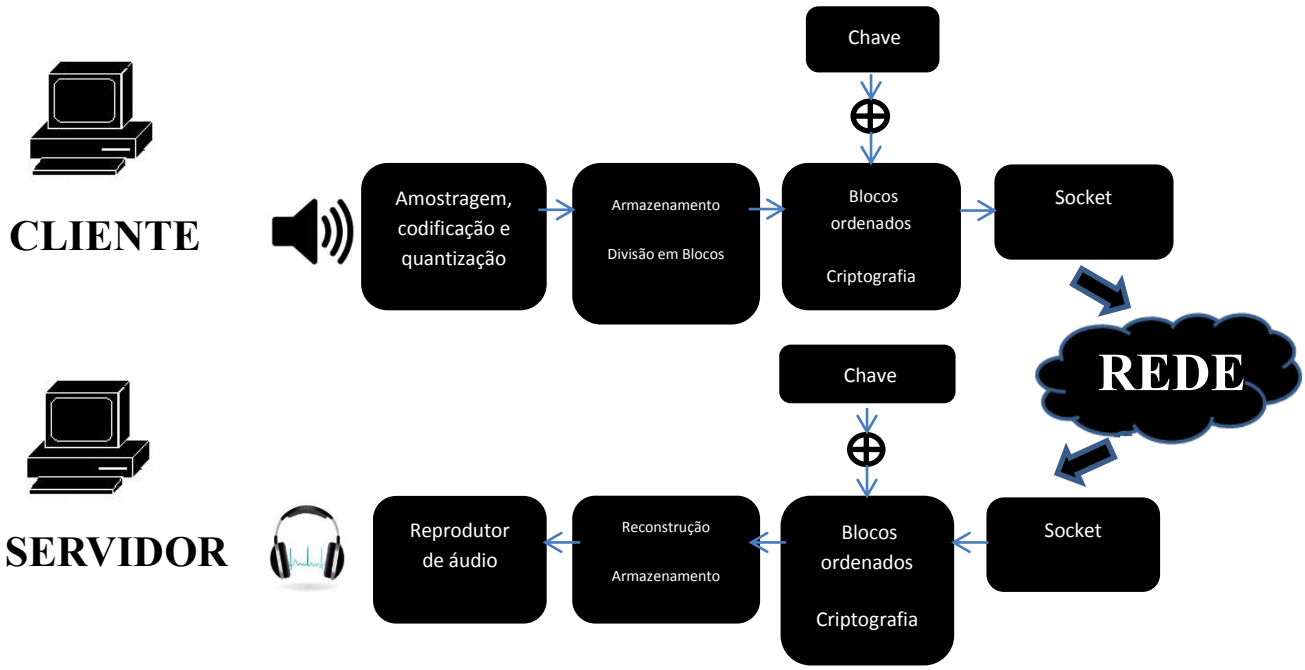

Figura 1.1 – Concepção geral do programa

A internet, meio essencial para comunicação VoIP, utiliza o protocolo TCP/IP, que apesar de não ser um protocolo muito rápido mostrou-se o mais adequado, devido às suas características tais como interoperabilidade entre diferentes sistemas, segurança na entrega de dados e a certeza de entrega, onde o pacote será entregue

mesmo que ocasione atrasos. Este acaba sendo um ponto negativo do TCP/IP, ser orientado à conexão e gerar atrasos. Para os aplicativos VoIP, transmissão em tempo real, convém-se usar UDP, que apesar de permitir perdas não apresenta o atraso do TCP/IP.

Para minimizar as perdas, esses aplicativos VoIP apresentam codificações, as quais não são reveladas. Codificar é modificar o sinal, informação, para que este se torne mais robusto e apropriado para ser transmitido.

Porém, tudo isso só foi possível com o avanço da internet, aumento das taxas e diminuição dos custos, e o fato de estar cada vez mais disseminada, estimulando a contínua melhora das tecnologias nessa área. O VoIP, foco deste trabalho, o qual era pouco pesquisado e estimulado anos atrás devido à baixa qualidade nos serviços de internet, ganhou muita força com a melhora na qualidade desta. Muitos aplicativos e programas fazem uso do VoIP como Skype, Whatsapp, Facebook, Viber e muitos outros. Alguns destes foram estudados e serão apresentados ao longo deste trabalho.

Atualmente, é possível, através da internet, o envio de mensagens, fotos, vídeos, áudio e realização de ligações em tempo real, incluindo vídeo conferências, devido ao grande avanço na capacidade da internet. Entender as tecnologias utilizadas pelos aplicativos e programas VoIP foram motivação para a realização deste projeto, afim de tentar produzir um programa que consiga fazer, reservadas as proporções, parte do que os aplicativos consagrados fazem.

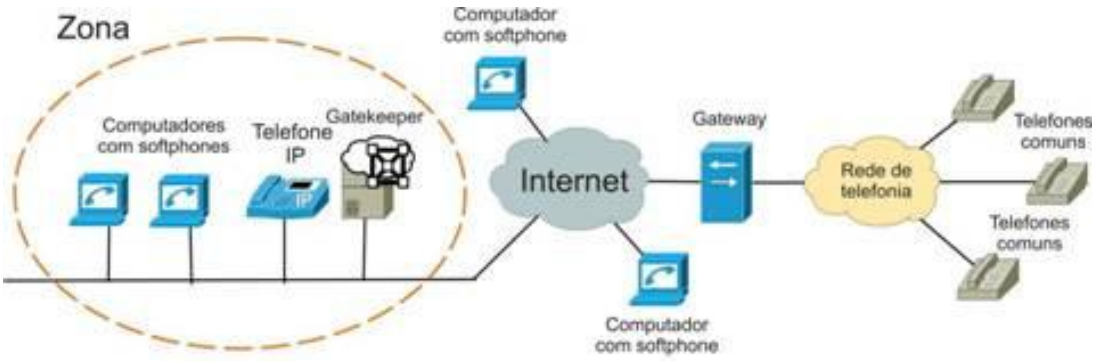

Figura 1.2 – Arquitetura títica de rede com VoIP

Fonte: http://www.teleco.com.br/tutoriais/tutorialvoipconv/pagina_4.asp

2. Internet

A internet, tão popular nos dias de hoje, onde a maioria da população tem acesso por diversos dispositivos, fixos e portáteis, foi criada inicialmente com objetivos militares. Durante a Guerra Fria, a busca por evoluções tecnológicas, a fim de superar o adversário, motivou os EUA a buscar uma opção para manter suas comunicações em caso de ataque Soviético.

> Era preciso, portanto, criar uma rede que ligasse os diferentes "nós" de forma redundante. Essa rede seria a ARPANET, criada pela Advanced Research Projects Agency.
> Em 1962, J.C.R. Licklider, do MIT, já falava na criação de uma Rede Intergalática de Computadores (Intergalactic Computer Network) (HEITLINGER, 2012).

O ARPA, do Departamento de Defesa dos EUA, contratou J.C.R. Licklider para liderar as novas iniciativas. Um dos sonhos de Licklider, que ficou no projeto até 1964, era uma rede de computadores que integrasse pessoas geograficamentes distantes. A ARPA, em 1967, contratou Lawrence Roberts para tornar a ideia uma realidade. O primeiro teste da rede ocorreu em 1970, por quatro universidades americanas, e em 1972 foi feita a primeira demonstração pública (SIMON, 1997).

Com a adoção dos protocolos TCP/IP, em 1983, surgiu a verdadeira internet onde se separou a utilização civil e militar, necessitando de um enorme aumento nas estruturas de coordenação e cooperação (HEITLINGER, 2012).

Em 1988 foi liberada a comercialização, inicializando a popularização da internet, seguida do “boom” em 1992 com a criação do World Wide Web, www, pelo cientista Tim Berners-Lee (BARROS, 2013).

3. VoIP

É uma tecnologia que utiliza uma Rede IP para transmitir voz e estabelecer chamadas telefônicas pela rede de dados. O fato de usar a internet faz com que os custos sejam reduzidos, pois os planos de internet muitas vezes se apresentam mais baratos comparados aos custos de ligações telefônicas internacionais, por exemplo.

A utilização da comutação por pacotes ao invés da comutação de circuitos, como na telefonia convencional na qual cada conexão tinha que existir fisicamente, faz com que o VoIP seja ainda mais econômico e eficiente, pois, além de economizar recursos físicos, minimiza o disperdício de banda.

Existe também a Telefonia IP que por utilizar o VoIP é comumente confundida com o mesmo, porém é apenas uma tecnologia que utiliza o VoIP dentre muitas outras. A Telefonia IP ao usar o VoIP, abrange uma gama de funcionalidades de integração nas comunicações possibilitando diversos serviços, como serviços de mensagem, vídeo/áudio conferência, vídeo-chamada, etc.

3.1. Histórico

O conceito de VoIP existe há mais de 25 anos, com a implementação das primeiras redes de computador, sendo possível graças à internet, a qual é um avanço do que Alexander Graham Bell iniciou ao inventar o telefone, gerando meios de interligar pessoas fisicamente distantes. A consolidação do VoIP se deu com o surgimento do primeiro software comercial por volta de 1990, o Internet Phone da VocalTec Communications, possibilitando a troca de pacotes de voz por IP (TELECO, 2016).

> Mais tarde, em 1998, começaram a surgir os primeiros "gateways", equipamentos capazes de interligar aparelhos telefônicos convencionais ou centrais telefônicas de empresas (PABX's) à rede de dados para comunicação entre estes sistemas com sistemas VoIP. Mais recentemente, surgiram "gateways" especializados e dispositivos denominados ATA (Analog Telephone Adapter - Adaptador para Telefone Analógico), para interligar dois sistemas convencionais e/ou PABX's utilizando como meio de transmissão redes IP. Tanto os "gateways" como os ATA's dispensam o uso de microcomputadores nas conexões (ZAPPAROLI, [200-?]).

Outras tecnologias foram surgindo, ou foram melhoradas, aumentando a qualidade do sinal e de transmissão como o uso de codificadores, buffers, filtros, dentre outros, fazendo com que mais informação pudesse ser transmitida além de melhorar a

qualidade da mesma. O que inicialmente eram pacotes de áudio, evoluiu para chamadas e já é possível realizar ligações com vídeo e com múltiplas pessoas, vídeo conferências.

3.2. Tecnologia

O VoIP transforma áudios, sinais analógicos, em sinais digitais para que os mesmos sejam transmitidos via protocolo TCP/UDP na internet. Logo, é necessário obter algum programa, software, que realize essa conversão.

A Agência Reguladora Brasileira, Anatel, não regulamenta o uso de tecnologias mas sim de serviços. Sendo assim, há duas frentes de utilização: a primeira e mais utilizada refere-se a comunicação entre 2(dois) dispositivos que possuam programa que utilizem apenas a internet, considerado Serviço de Valor Adicionado; e a segunda refere-se à comunicação entre um dispositivo que utilize programa para transmitir via internet e outro dispositivo que utilize a telefonia convencional, constituindo-se de um Serviço de Telecomunicações (ANATEL, 2016).

Para estabelecer a comunicação entre VoIP e telefonia convencional é necessário o uso de um Adaptador Telefônico Analógico, um conversor analógico-digital.

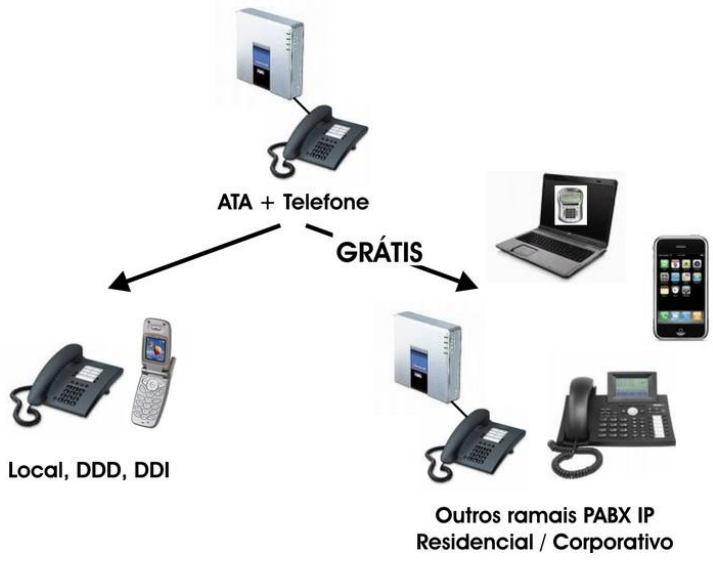

Figura 3.1 – Estrutura de Comunicação com uso do ATA

Fonte: http://savacom.com.br/ata.html

3.3. Vantagens e Desvantagens

As vantagens do VoIP são amplas, porém o fator preponderante para a popularização é o custo reduzido desta tecnologia, a qual apresenta economia na utilização do serviço e melhoria no aproveitamento da banda, pois não necessita da banda reservada exclusivamente a uma ligação.

Por utilizar a internet não é necessário criar nenhuma estrutura, aproveita uma já pronta, além disso, permite a integração dos dispositivos com outros serviços como vídeo, compartilhamento de arquivos, dentre outros. O aplicativo WhatsApp é um exemplo claro disso, o qual faz ligações, envia áudios, imagens, etc.

Pode-se citar também a versatilidade, pois a ligação feita de um celular(smartphone) pode ser direcionada para outro celular bem como para computadores, tablets e outros dispositivos que apresentem funcionalidade para executar ligações, VoIP ou não.

> Além de baratearem o custo das ligações telefônicas, o VoIP ainda permite que várias chamadas ocupam o mesmo espeço que é preenchido somente por um na telefonia tradicional. Na rede convencional, 10 minutos de ligação consumiam 10 minutos de transmissão a uma taxa de 128 Kbp/s, enquanto com VoIP a mesma ligação pode ocupar somente 3,5 minutos do tempo de transmissão a 64 kbp/s, deixando o restante livre para outras chamadas (PIXININE, 2015).

Apesar de trazer vários benefícios, é a internet que ocasiona as principais desvantangens. A qualidade das ligações VoIP está diretamente ligada à qualidade dos serviços de internet contratado, que varia por diversos fatores, dentre eles a banda contratada e o local a se utilizar os serviços. Devido à baixas velocidades ou problemas na recepção do sinal, pode-se ocorrer geração de ecos, cortes nas ligações, ou seja, uma redução considerável na qualidade da ligação.

Nos serviços VoIP, a identificação de usuário nas ligações é mais complicada, se comparada com a telefonia convenciona. É possível alterar o endereço de IP, fazendo com que a identificação torne-se realmente complicada. A internet é um mar de possibilidades, inclusive para fazer o mal. Isso ocasiona insegurança, pois possibilita crimes nos quais a polícia tem mais dificuldade em identificar os criminosos e as vezes não os identifica, dependendo dos recursos utilizados.

Apesar das desvantagens o VoIP populariza-se cada vez mais, porém as falhas que podem ocorrer ainda são razão para que as empresas não substituam completamente a telefonia convencional, pois esta ainda continua mais estável e confiável.

4. Protocolos

Os protocolos utilizados nas comunicações pela internet abrangem as camadas de Rede, Transporte e Aplicação, onde cada uma apresenta os próprios protocolos e são responsáveis por determinada tarefa.

Na camada de Enlace a comunicação é possível devido ao fato de o transmissor e receptor estarem na mesma rede. Já a camada de Rede faz a comunicação de roteador à roteador, entre redes distintas, e decide qual o melhor caminho para os dados. Ela converte o endereço físico em lógico, endereço IP, e é a camada responsável por intermediar a camada de Enlace com a de Transporte.

A camada de Aplicação, no TCP/IP, é a interface usuário-rede, faz as requisições na rede ou recebe as informações da mesma. Oferece diversos serviços ao usuário, gerencia os sistemas e ferramentas e disponibiliza uma interface mais simples, de maneira que facilite a interação. Engloba as camadas de Aplicação, Apresentação e Sessão, distintas no modelo Open Systems Interconnection. Dentre seus protocolos estão o Hyper Text Transfer Protocol, responsável pela ligação de recursos da World Wide Web, o Simple Mail Transfer Protocol, para enviar emails, e o Domain Name System, para identificar nome de domínios como sites.

Já a camada de Transporte, a qual será abordada com mais detalhes, é responsável pela entrega dos dados, podendo ser ou não orientado a conexão. No caso de orientado a conexão, TCP/IP, os dados são reenviados em caso de erros mesmo que isso comprometa a velocidade, garantindo a certeza de entrega. Já no caso de não orientado, UDP, a velocidade é mantida mesmo que informação seja perdida, usado principalmente em transmissão em tempo real e nas conexões com os servidores de DNS. Esses são os protocolos mais usados pelas tecnologias atuais, apesar de não serem os únicos existentes.

4.1. Transmission Control Protocol/Internet Protocol - TCP/IP

O TCP/IP não representa apenas um protocolo, seu nome por exemplo refere-se a dois protocolos, TCP e IP. Como citado anteriormente, é orientado a conexão, para que possa ter o controle dos pacotes enviados. Nele, ao se enviar um pacote, espera-se uma resposta de confirmação para garantir que o pacote chegou ao destino. Caso

isso não aconteça o pacote é retransmitido até que a resposta de confirmação seja recebida.

Além disso, apresenta mecanismos para evitar que sejam recebidos pacotes corrompidos, ou seja, ao receber-se um pacote é possível determinar se o mesmo está ou não corrompido e, caso esteja, o pacote é retransmitido. Por isso é muito raro que um site apresente problemas, como um link que não seja carregado, ou que um download venha corrompido, mesmo quando a conexão com a internet é bastante oscilante.

Os pacotes enviados, por TCP/IP, apresentam uma numeração própria, que serve para ordenação, e devem chegar ao destino na ordem determinada, ocasionando retransmissão em caso de chegada desordenada. Outros mecanismos presentes no TCP/IP são o Controle de Congestionamento e o Controle de Fluxo, buffers e janelas deslizantes.

A conexão é feita em 3 etapas, primeiramente o cliente inicia a ligação enviando um pacote com a flag SYN ativa, então o servidor para aceitar a conexão envia um pacote com as flags SYN e ACK ativas, caso o cliente não receba a resposta de confirmação após o timeout o mesmo envia novamente o pacote SYN. A ligação é concluída com o envio, pelo cliente, do pacote com a flag ACK. A conexão somente é encerrada após o recebimento de todos os dados pelo cliente, sendo essa a principal característica do TCP/IP, a confiabilidade.

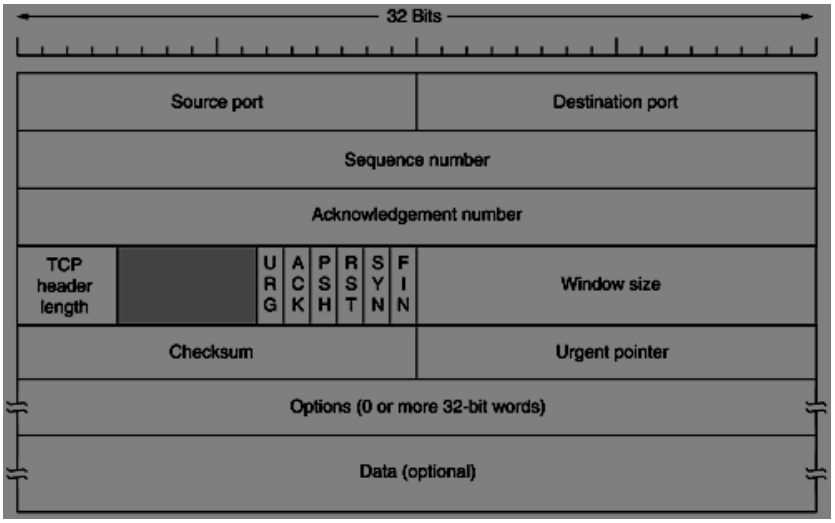

Figura 4.1.1 – Cabeçalho TCP

Fonte: Tanenbaum, 2003

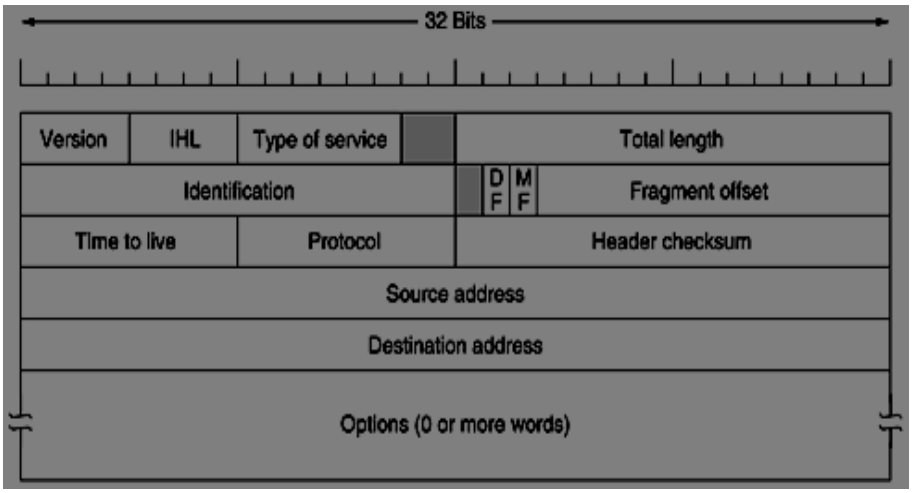

Figura 4.1.2 – Cabeçalho IP

Fonte: Tanenbaum, 2003

4.2. User Datagram Protocol - UDP

Diferentemente do TCP/IP, o protocolo UDP não é orientado a conexão. Como principais características estão a não garantia de entrega, falta de correção de erros, o não reenvio do pacote em caso do mesmo ser recebido corrompido e a ausência de controle de fluxo. Todas essas desvantagens, em relação ao TCP/IP, deve-se ao fato de ser mais simples e rápido. Por não se importar com a chegada dos pacotes, o protocolo UDP consegue transmitir em uma taxa mais alta que o TCP/IP.

Esse protocolo é muito útil em pequenos serviços como: Trivial File Transfer Protocol, Simple Network Management Protocol, Dynamic Host Configuration Protocol, DNS, além de transmitir dados pouco sensíveis, streaming de áudio e vídeo, e transmissões em tempo real.

A vulnerabilidade desse sistema é também um ponto importante, pois a não orientação a conexão faz com que seja possível invasões, resultando em obtenção e até mesmo na alteração de dados, bastando que o *hacker* utilize também o UDP, podendo falsificar o seu IP de maneira que torne difícil ou impossível a determinação de sua real identidade. Em contra partida, isso pode ser corrigido com o uso de um *Firewall* na rede. Em geral, como já citado, o UDP é utilizado apenas em serviços específicos, sendo o TCP/IP o mais utilizado, como é feito na Internet.

5. Programas que utilizam VoIP

Apesar da tecnologia VoIP ter sido criada, desenvolvida, na década de 90 somente popularizou-se nos anos 2000 e ganhou força em 2003 com o lançamento do Skype, o qual tornou-se um dos mais populares aplicativos de comunicação VoIP. Desde então muitos outros programas, *softwares*, foram sendo lançados. Dentre eles o Facebook Messenger, Gizmo, Viber, WhatsApp, Google Hangouts e muitos outros, sendo mais populares os gratuitos. O Skype e o Facebook Messenger foram utilizados como ferramenta de estudo durante este projeto e portanto terão um enfoque maior nos tópicos a seguir.

5.1. Skype

O Skype é baseado em comunicação Peer to Peer, porém não é totalmente pura, isto é, a infraestrutura conta com um servidor central de autenticação para validação de usuários na rede e atualizações de software que são assinados digitalmente pela chave privada RSA1 embutida em seus executáveis. Utiliza TCP e UDP como protocolos de transporte e faz uso de diversos codificadores/decodificadores (codecs), escolhendo qual o melhor de acordo com a banda e condições de tráfego.

Foi fundado em 2003, em Luxemburgo, por Niklas Zennström e Janus Friis com o simples ideia de oferecer comunicação via internet a qualquer um em qualquer parte do planeta. Foi uma ideia revolucionária, pois não somente permite a comunicação entre dois possuidores do *software*, mas também vídeo conferência, além da comunicação entre o software e a telefonia convencional, fixa e móvel.

Foi vendido em 2005 para o eBay por US$ 2,6 bilhões e possuía por volta de 50 milhões de usuários, número que passou pra 100 milhões em 2006, ano em que iniciou-se as videochamadas. Em 2009 foi lançado para Android e iOS, aumentando ainda mais o número de usuários, e foi nesse ano também que as ações da empresa foram divididas entre eBay, Silver Lake e Andreessen-Horowitz.

Algumas novidades ainda foram lançadas antes de sua última venda, que ocorreu em 2011 para a Microsoft por 8,5 bilhões, tais como a videochamada em grupos de até 4 pessoas, incorporação do Skype à Smart TVs e o funcionamento utilizando internet móvel, 3G.

Após a compra a Microsoft informou que não pretendia, a princípio, fundir o Skype com nenhum *software* existente, o objetivo era possibilitar a integração com seus produtos, *softwares*, já existentes como Xbox, Kinect, entre outros. O Skype, segundo a última pesquisa em 2016, já possui em torno de 300 milhões de usuários.

5.2. Facebook Messenger

O Facebook foi fundado em fevereiro de 2004, com objetivos iniciais de interligar os alunos de Havard, por Mark Zuckerberg e seus colegas de quarto, Chris Hughes, Dustin Moskovitz e Eduardo Saverin.

Devido ao sucesso do site, foi rapidamente expandido para outras faculdades dos EUA e em sequência para faculdades de outros países. Em pouco tempo deixou de ser algo para faculdades e abriu-se para o mundo. Em 2006, para aumentar ainda mais o número de usuários que já chegava a 5,5 milhões, a rede social passou a aceitar qualquer pessoa com 13 anos ou mais. Foi quando começaram a surgir as maiores mudanças, muitas das quais continuam até hoje, como o *feed* de notícias, o lançamento do aplicativo para celulares, e muitos outras. Em relação ao aplicativo, rapidamente o número de usuários conectados por aparelhos portáteis superou o número de usuários conectados por aparelhos fixos.

Uma das ferramentas mais usadas é o *chat*, a possibilidade de conversar em privado com outros usuários. Devido a sua grande demanda, foi lançado o Facebook Messenger, aplicativo com a função exclusiva de conversação. O que inicialmente era somente mensagem foi aprimorado e atualmente é possível fazer ligações e chamadas de vídeo.

O Facebook Messenger utiliza o *Session Initiation Protocol* para iniciar sessões de comunicação interativa entre usuários. O SIP é um protocolo de Aplicação e apresenta arquitetura similar ao HTTP, cliente e servidor. A arquitetura é composta por *User Agent, User Agent Client, User Agent Server, Multimedia Session e Server*.

Entre os serviços disponíveis estão: Localização de usuário, localizar o usuário dentro da rede devido à sua movimentação na mesma, capacidade do usuário, determinar quais mídias podem ser usadas por determinado usuário, controle da chamada, usuário gerencia as chamadas, disponibilidade do usuário, determinada se

em dado momento o usuário está disponível ou não para realizar uma chamada, e configuração de chamada, define os parametros a serem usados na chamada.

6. Codificação

Codificar é transformar uma informação em uma mensagem, em determinado formato baseado em regras, para que a informação consiga ser transmitida e entendida. As comunicações sempre envolvem incerteza, por isso as informações são codificadas antes de serem enviadas, podendo ser por uma codificação de fonte, com o intuito de tornar a mensagem adequada ao sistema de digital de comunicação, como no caso da digitalização da voz e vídeo, ou então por uma codificação de canal para tornar a mensagem mais robusta ao canal, diminuir a interferência do canal na mensagem recebida para que a mesma apresente um menor erro, de acordo com as determinações do projeto previamente estabelecidos.

Os fatores que causam incerteza nas comunicações são diversos, como interferência entre símbolos, ruído, atenuação do sinal devido a distância, variações no canal de transmissão, entre outros, então deve-se analisar, para cada caso, qual técnica de codificação será mais adequada, dentro das diversas opções tanto de fonte como de canal.

A codificação de fonte visa reduzir ao máximo as redundâncias em uma sequência de informação gerada no transmissor, aumentando a eficiência espectral. Os símbolos são mapeados, de acordo com o código utilizado, em uma sequência de dígitos binários. De acordo com o 1° Teorema de Shannon, dada uma fonte discreta e sem memória de informação cuja entropia é H, o comprimento médio de um código sem distorção para essa fonte será sempre maior ou igual que H (HAYKIN, 2001). Em outras palavras, o comprimento médio de um código sempre será maior que a entropia da fonte de informação. Para os sistemas reais busca-se obter valores muito próximos de H, o que pode ser obtido com o popular código de Huffman.

Uma codificação bastante empregada em sistemas de comunicação que lidam com sinais de voz é o *Pulse Code Modulation*. No PCM, um sinal analógico é amostrado e quantizado de acordo com sua amplitude. Cada nível de quantização será representado por um código binário que pode ser diretamente convertido para um número decimal proporcional a amplitude do sinal original.

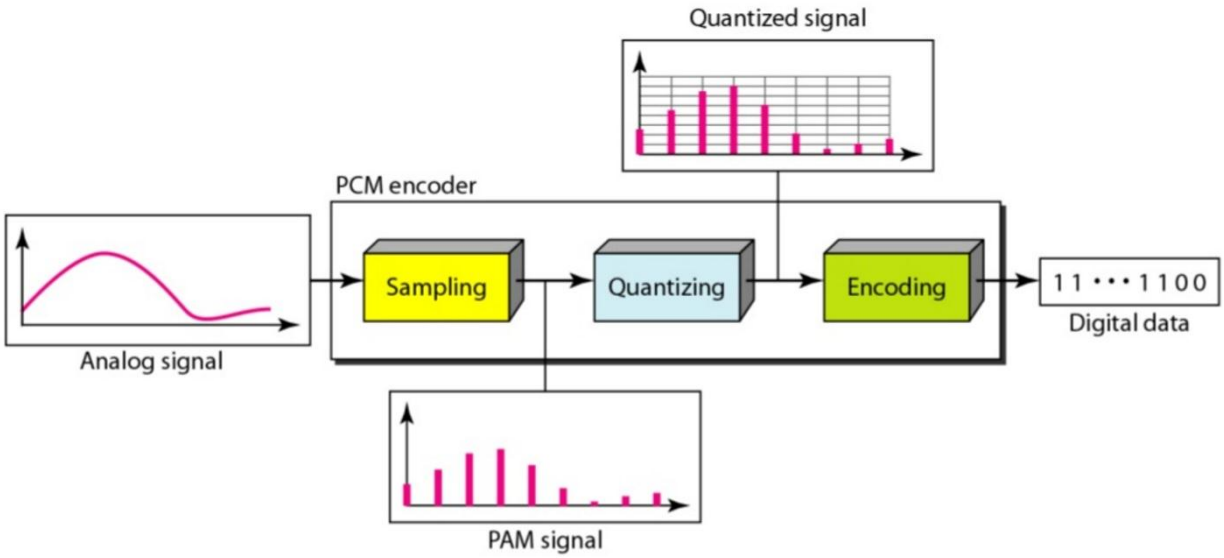

Figura 6.1 – Processo de codificação PCM

Fonte: http://pt.slideshare.net/vishnudharan11/pulse-code-modulation-pcm

7. Criptografia

Criptografia é a técnica de codificar a mensagem de maneira que não possa ser entendida por terceiros, apenas o emissor e receptor os quais devem possuir a chave para decifrar, ou seja, consiste em transformar uma informação legível em uma combinação de caracteres ininteligíveis.

O nome vem do grego, junção de *Kriptós* (Secreto) com *Gráphein* (Escrita). Apesar de muito estudada na atualidade a criptografia é uma técnica antiga, já era utilizada pelos egípcios, romanos, hebreus, espartanos e muitos outros povos, para esconder segredos e trocar informações importantes.

Uma das técnicas clássicas mais conhecidas é a Cifra de César, que foi usada por César para se comunicar com suas tropas durante as guerras. Ela consiste, no caso da chave ser 3, em substituir cada letra do alfabeto por 3 letras a frente, ou seja, a letra A seria D, a letra B seria E e assim por diante. Por apresentar apenas 26 variações, 26 possibilidades de chave, é considerado inseguro e de fácil quebra. Um modo de melhorar é utilizando permutação, no qual obtem-se 26! (26 fatorial) opções de chave.

O mais conhecido avanço do século passado foi a Máquina Enigma, desenvolvida em 1918, e que despertou grande interesse alemão, sendo adquirida em 1926 e melhorada, desenvolvida uma versão própria pelos alemães, em 1928 passando a ser usada por todo o exército alemão, o qual trocava diariamente sua chave. A mensagem criptografada pela maquina elétrico-mecânica era considerada de impossível descriptografia, devido à necessidade de alto poder computacional para o mesmo.

Somente foi possível decifrar os códigos, apontam a maioria dos pesquisadores, após o roubo de uma dessas máquinas. Através da engenharia reversa um grupo de engenheiros e matemáticos, poloneses e ingleses, foi capaz de decifrar, sendo Alan Turing, o pai da computação moderna, o mais conhecido.

A mesma chave era usada criptografar e descriptografar as informações, o que é conhecido por Criptografia por chave Simétrica. Com a evolução das tecnologias e dos estudos na área foram desenvolvidas outras técnicas como a Criptografia por chave Assimétrica e Criptografia Hash.

Pode-se também utilizar ambos os métodos, com e sem chave, como feito pelos protocolos mais usados na internet, o Transport Layer Security e o Secure Sockets Layer. Nestes protocolos são utilizadas chaves públicas e privadas, que se complementam, resolvendo o problema de compartilhar as chaves pela internet. Entretetando o uso de chaves assimétricas eficientes exigem bastante capacidade de processamento, o que pode reduzir o desempenho de um sistema mais simples.

8. PRÁTICA

8. 1. MATERIAL UTILIZADO

8. 1. 1. COMPUTADORES

Usaremos computadores pessoais para o desenvolvimento deste trabalho. Nesses computadores serão instalados os *softwares*, desenvolvidos e testados os códigos. É recomendado que os computadores sejam formatados para instalar os *softwares* a se utilizar. É possível a utilização em conjunto com outro sistema operacional utilizando máquina virtual, por exemplo, porém isso deixa o sistema operacional emulado muito lento. Dessa forma, recomenda-se a instalação exclusiva ou a utilização de Dual Boot.

Foram utilizados 3 computadores: em um deles foi instalado exclusivamente o Ubuntu; no segundo usou-se um *dual boot* de Ubuntu com Windows 7; e no terceiro, que possuía Windows 10, foi instalada a Máquina Virtual Ubuntu. Esses dois primeiros foram os computadores utilizados em praticamente todo o projeto, enquanto o terceiro foi utilizado somente para pequenos testes quando um dos dois primeiros computadores não estava disponível para uso.

8. 2. INSTALAÇÃO DOS SOFTWARES

8. 2. 1. UBUNTU 14.04.4 LTS

É recomendado, como já comentado, que os computadores sejam formatados antes da instalação . Faça o *download* do Ubuntu 14.04.4 lts e a partir de um DVD ou Pendrive rode o programa e clique em Install Ubuntu-14-04-LTS, após siga os seguinte passos:

- Selecione o idioma desejado;
- Certifique-se de ter espaço suficiente em disco e estar conectado à Internet e a uma fonte;
- Selecione Opção Avançada, em caso de particionamento prévio do HD;
- Selecione onde pretende instalar o Ubuntu e clique em Change;
- Clique em Instalar agora;

- Selecione a sua Localização;
- Selecione o Layout do teclado;
- Defina um nome para o Computador;
- Defina um nome de Usuário;
- Defina uma Senha.

Após clicar em continuar a instalação será concluída e então deve-se reiniciar o computador para começar a rodar o Ubuntu.

8. 2. 2. MÁQUINA VIRTUAL

Inicialmente deve-se baixar e instalar o VirtualBox, para depois criar a Máquina Virtual. Após o *download* do VirtualBox, clique no ícone e siga os passos:

- Clique em Next na primeira tela que aparecer;
- Configure, caso deseje, as opções de instalação, o recomendado é que mantenha-se como aparece, e então clique em *Next*;
- Marque as opções desejadas de ícones e clique em *Next*;
- As conexões de rede deverão ser reiniciadas, clique em *Yes* para prosseguir.

Agora basta clicar no botão *Install* e em seguida no botão *Finish* e estará pronto para uso. Basta agora criar a Máquina Virtual, para isso abra o VirtualBox e siga os passos:

- Clique no botão Novo;
- Defina o nome da Máquina e escolha o Sistema Operacional, nos campos Tipo e Versão, e clique no botão Próximo(N);
- Defina a quantidade de memória RAM a ser utilizada na Máquina Virtual, recomenda-se apenas um quarto (1/4) do total, e clique no botão Próximo(N);
- Selecione a opção “Criar um disco rígido virtual agora” e clique no botão Criar;
- Selecione a opção “VDI (VirtualBox Disk Image)” e clique no botão Próximo(N);

- Selecione a opção “Dinamicamente alocado” e clique no botão Próximo(N);
- Defina o tamanho máximo que o HD virtual ocupará em seu HD real e clique no botão Criar.

A Máquina Virtual está criada, bastando agora instalar o Sistema Operacional. Abra o VirtualBox e inicie a Máquina Virtual criada, para a instalação do Ubuntu siga os mesmos passos já citados anteriormente.

8. 3. CÓDIGO

O trabalho inicial foi a implementação e modificação do código de um trabalho já pronto, o IP, Iniciação a Pesquisa, do Major André Luis Souza de Araújo e 2 colegas de turma (SOUZA, 1999). Foram feitos 2 programas, o Cliente e o Servidor. A ideia era rodar ambos os programas nos 2 computadores, utilizando o programa Cliente para fazer a transmissão e o programa Servidor para a recepção.

Os programas são compilados utilizando o GNU Compiler Collection no Prompt de comando do Ubuntu e a comunicação entre os computadores é feita através de uma entidade lógica denominada *socket*. Esta entidade é uma interface entre as camadas de Aplicação e de Transporte do modelo TCP/IP. A camada de Transporte oferece um serviço para a camada de aplicação através dessa interface, que é caracterizada pela Porta (*gate*). Junto com a porta, o endereço IP do servidor também faz parte da identificação do serviço fornecido pela camada de Transporte.

A primeira versão dos códigos era bem simples, permitia apenas o envio de arquivos de áudio previamente gravados e com tamanhos determinados, ou seja, era necessário fazer a gravação do áudio fora do programa e com tamanhos fixados para após isso rodar os programas e realizar o envio.

Essa versão foi melhorada para que fosse enviado um áudio de tamanho qualquer sem a necessidade de saber previamente o tamanho do mesmo.

O próximo passo foi permitir a gravação do áudio durante a execução do programa. Para tanto, recorreu-se ao uso de *Application Programming Interface* do Ubuntu, onde o programa solicita ao Sistema Operacional recursos para utilizar a placa de som. A execução do comando “arecord” diretamente sobre o *prompt* aciona drivers

para controlar a placa de som. Esses drivers fazem parte de um pacote chamado de *Advanced Linux Sound Architecture*. Ao rodar o programa era pedido para apertar a tecla Enter para iniciar a gravação e as teclas Ctrl e C, ao mesmo tempo, para encerrar a gravação.

A última modificação no código, antes do final do projeto, foi a implementação de embaralhamento e criptografia antes de se enviar o áudio. A criptografia funcionou bem para chaves de até 8 bytes, porém o embaralhamento apresentava eventualmente alguns erros cujas causas não foram identificadas. Assim, o módulo criptográfico implementa uma codificação com uma chave de 8 bytes e opcionalmente pode embaralhar os pacotes em 4 sub-pacotes. O embaralhamento e criptografia foram feitos dentro do *loop* de envio, o programa envia o áudio de 32 em 32 bytes através de um loop.

Portanto o projeto final é capaz de iniciar a comunicação entre 2 computadores, gravar um áudio no computador que estiver executando o programa Cliente e enviá-lo para o computador com o Servidor. Este procedimento pode ser executado simultaneamente no caminho inverso configurando uma comunicação *full-duplex*. Um sistema de criptografia simples ajuda a proteger os dados transmitidos.

A seguir, os códigos desenvolvidos para este trabalho serão comentados e explicados. Primeiramente iremos analisar o código do programa cliente e na sequência o código um pouco mais complexo relativo ao programa servidor.

8. 3. 1. PROGRAMA CLIENTE

Como todo programa C, começa declarando variáveis e designando bibliotecas. A maioria das bibliotecas chamadas nesse código são típicas de redes de computadores e necessárias para a criação dos *sockets*. As duas variáveis globais mais importantes do programa são declaradas neste escopo. Tem-se a variável buffer[32], que trata-se de um vetor de 32 bytes que representará um pacote de voz digitalizada que transitará pela rede IP. Aqui também é declarada a variável senha[8], que trata-se da chave secreta usada para manter a troca de informações segura durante a transmissão. A chave foi fixada para as demonstrações mas basta que seja alterada nesta linha para usar uma senha diferente. O tamanho de 8 bytes para a senha foi

escolhida por ser eficaz em esconder a informação e ao mesmo tempo não comprometer o desempenho do programa.

```
//PROGRAMA CLIENTE PARA COMUNICACAO VOIP – PFC IME 2016
TELECOMUNICACOES
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <stdio.h>
#include <sys/time.h>
#include <time.h>
#include <stdlib.h>
#include <strings.h>

struct tm *t;
time_t lt;
int primeira=0;


int audio_leitura;
unsigned char buffer[32], auxbuffer[32];
unsigned char senha[8]={0xAF, 0x12, 0x6F, 0xAB, 0x5B, 0x90, 0x55, 0x4A};
```

O próximo passo é a declaração da função principal. Observe que a função exige a passagem de parâmetros através do ponteiro argv. Esses parâmetros são o endereço IP e a porta do computador servidor, respectivamente. Ao executar esse programa, o usuário deve entrar com o nome do programa seguido do IP e da porta do servidor para assim alimentar a função main.

O programa segue declarando algumas variáveis locais e cria um arquivo no HD, onde será gravado o áudio. Por fim, o programa avisa ao usuário que está pronto para gravar e espera os comandos do usuário. Para realizar a gravação, o programa solicita ao sistema operacional, através de uma API, recursos para usar a placa de som. Nesse ponto, optou-se por utilizar as facilidades que o Sistema Ubuntu oferece enviando comandos diretamente ao kernel do Sistema Operacional. Os drivers usados

para controlar a placa de som são os da ALSA, que permitem ao usuário controlar diversos aspectos da gravação, como a período de amostragem, número de canais, resolução, etc. Nesse programa optou-se por fazer 8000 amostras por segundo, canal MONO e resolução de 8 bits por amostra, o que se mostrou adequado para aplicações em telefonia.

```
main(argc, argv)
int argc;
char *argv[];
{
while(1)

{
        int aux=0;
        int n, m;
        char nomeArquivo[100]; // nome do arquivo
        FILE *arquivo; // ponteiro para o arquivo
        long tamanho;

        printf("Para gravar um audio tecle ENTER \n");
        printf("Ao final da gravacao, tecle CRTL+C \n");
        getchar();
        system("arecord teste.wav");
```

Com o áudio já gravado através de uma codificação PCM e armazenado no HD, o programa passa a se preocupar com a conexão ao servidor. Nesse momento algumas variáveis extras relativas ao *socket* são declaradas e algumas funções típicas relativas a abertura de *socket* e estabelecimento de conexão são invocadas. Se não houverem erros, o programa se conectará ao programa servidor.

```
int sock;
        unsigned short int i,j,k;
        short int dif=0;
        struct sockaddr_in server;
        struct hostent *hp, *gethostbyname();

        /*Cria o socket*/
        sock=socket(PF_INET, SOCK_STREAM, 0);
        if(sock < 0)
        {
                perror("abertura de socket");
                exit(0);
        }
        server.sin_family = PF_INET;
        hp = gethostbyname(argv[1]);
        if(hp==0)
        {
                fprintf(stderr, "%s: host desconhecido.", argv[1]);
                exit(2);
        }
        bcopy((char *)hp->h_addr, (char *)&server.sin_addr, hp->h_length);
        server.sin_port = htons(atoi(argv[2]));

        if (connect(sock, (struct sockaddr *)&server, sizeof server) < 0)
        {
                perror(" conexao de socket");
                exit(1);
        }
```

Agora que tem-se a conexão aberta e o audio já gravado no HD, basta tratar de enviá-lo ao *socket*. Para isso, o programa verifica o tamanho o exato do arquivo de áudio, assim ele saberá em quantos blocos de 32 bytes ele terá que ser dividido. Essa informação é guardada na variável tamanho.

```
        arquivo = fopen("teste.wav", "r");
        // verifica se o arquivo foi aberto com sucesso
        if (arquivo != NULL)
        // movimenta a posicao corrente de leitura no arquivo para o seu fim
        fseek(arquivo, 0, SEEK_END);
// copia a posicao corrente de leitura no arquivo
        tamanho = ftell(arquivo);
        audio_leitura = open("teste.wav", O_RDONLY);
```

O programa entra num *loop*, de maneira que trate todos os pacotes de 32 bytes em ordem. Essa rotina tem a função de extrair os próximos 32 bytes do arquivo original e colocá-lo no vetor buffer[32], embaralhar o vetor, criptografar o vetor e enviá-lo ao *socket*. Após fazer isso com todos os blocos, o programa encerra a conexão, volta ao início e aguarda uma nova gravação de áudio. A parte de embaralhamento foi comentada e foi deixada como opcional, pois provocou distorções não esperadas na decodificação de 60% dos testes realizados, ocassionando ruídos no áudio recebido.

```
        while (aux<(tamanho/32))
        {
                read(audio_leitura, buffer, 32);
                /*for (m=0; m<32; m++)     //embaralhamento
                        auxbuffer[m]=buffer[m];

                for (m=0; m<8; m++)
                {
                        buffer[m]=auxbuffer[m+24];
                        buffer[m+8]=auxbuffer[m];
                        buffer[m+16]=auxbuffer[m+8];
                        buffer[m+24]=auxbuffer[m+16];
                }*/
                for (n=0; n<8; n++)      //criptografia
                {
                        buffer[n]=buffer[n]^senha[n];
                        buffer[n+8]=buffer[n+8]^senha[n];
                        buffer[n+24]=buffer[n+24]^senha[n];
                        buffer[n+16]=buffer[n+16]^senha[n];

                }
                if(write(sock, buffer, 32) < 0)   //escrita no socket
                        perror(" escrita no socket");
                aux++;
        }

        close(sock);
}exit(0);}
```

8. 3. 2. PROGRAMA SERVIDOR

O programa servidor é um pouco mais complexo, possuindo chamadas de funções típicas de servidor e alguns recursos multitarefas. Toda a primeira parte é análoga ao cliente, com declarações de variáveis, bibliotecas, função main e definição da senha. Entretando, para o servidor só existe um argumento argv para a main, que é a porta oferecida para a conexão.

```
//PROGRAMA SERVIDOR PARA COMUNICACAO VOIP – PFC IME 2016
TELECOMUNICACOES
#include <sys/stat.h>
#include <fcntl.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>
#include <stdio.h>
#include <sys/time.h>
#include <time.h>
#include <stdlib.h>
#include <strings.h>

struct tm *t,*t2;
time_t It,It2;
int contatempo=0;
int primeira=0;

main (argc, argv)
int argc;
char *argv[];
{
while(1)
{
        int sock,i,j;
        struct sockaddr_in server;
        int msgsock;
        unsigned char buf[32], auxbuf[32];
        unsigned char senha[8]={0xAF, 0x12, 0x6F, 0xAB, 0x5B, 0x90, 0x55, 0x4A};
```

Também analogamente ao programa cliente, o aplicativo irá criar um arquivo de extensão .wav no HD da máquina, onde o áudio recebido será gravado e regravado toda vez que uma transmissão foi concluída.

```
int n;
int rval;
int length;
int audio_escrita;

FILE *f = fopen("teste.wav","w");
audio_escrita = open("teste.wav", O_WRONLY);
```

O próximo passo são as rotinas relativas a *socket* em um servidor TCP. Tem-se a criação do *socket*, que devolve um inteiro como descritor do mesmo, seguido da função bind(), que relaciona esse descritor com um endereço na memória e, não havendo nenhum erro, o *socket* entra no modo listen(), aguardando conexões. A porta escolhida, argumento da função main, é associada com este *socket*.

```
/*Cria um socket*/
sock = socket(PF_INET, SOCK_STREAM, 0);
if (sock < 0)
{
        perror ("Abertura de socket errado. ");
        exit (1);
}

/*Nomeia socket usando wildcards*/
server.sin_family = PF_INET;
server.sin_addr.s_addr = INADDR_ANY;
server.sin_port = htons(atoi(argv[1]));

if(bind (sock, (struct sockaddr *) &server, sizeof server) < 0)
{
        perror ( "ligacao (bind) no socket errado");
        exit (1);
}

/*encontra o numero da porta associado e o imprime*/
length = sizeof server;
```

```
if(getsockname (sock, (struct sockaddr *) &server, &length) < 0)
{
        perror ("obtencao do nome do socket errado");
        exit (1);
}
printf ("Socket port %d\n\r", ntohs (server.sin_port));

/*Inicia aceitacao das conexoes*/
listen (sock, 5);
msgsock = accept (sock, (struct sockaddr *) 0, (int *) 0);
if (msgsock == -1)
        perror ("Accept com erro");
else
{
```

Enquanto não chegam solicitações de conexão, o programa segue congelado em modo *listen*. Assim que se estabelecer a primeira conexão TCP, o programa está pronto para verificar o *socket* criado em busca dos pacotes transmitidos. Da mesma forma que no programa cliente, pacotes de 32 bytes irão chegar pelo *socket* e serão gravados em um vetor unsigned char buf[32].

```
do
        {
                i=0;j=0;
                bzero (buf, sizeof buf);
                if ((rval = read (msgsock, buf, 32)) > 0)
                        perror (" Lendo mensagem...");
                if (rval == 0)
                        printf ("terminando conexao...\n\r");
```

Como é utilizado o protocolo TCP, a entrega dos pacotes é garantida e em ordem. Portanto, espera-se que estejam armazenados no vetor buf[] exatamente os mesmos 32 bytes que saíram do programa cliente. Caso isso não ocorra, as funções de criptografia e desembaralhamento iriaõ distorcer mais ainda o sinal. Para descriptografar, basta usar a mesma senha do programa cliente. Novamente, a função de desembaralhamento fica como opcional e segue comentada no corpo do código.

```
        for ( n=0; n<8; n++)                    //descriptografia
                        {
                                buf[n]=buf[n]^senha[n];
                                buf[n+8]=buf[n+8]^senha[n];
                                buf[n+16]=buf[n+16]^senha[n];
                                buf[n+24]=buf[n+24]^senha[n];
                        }

/*                      for ( n=0; n<32; n++)     //desembaralhamento

                        {
                                auxbuf[n]=buf[n];
                                buf[n]=0;
                        }

                        for ( n=0; n<8; n++)
                        {
                                buf[n]=auxbuf[n+8];
                                buf[n+8]=auxbuf[n+16];
                                buf[n+16]=auxbuf[n+24];
                                buf[n+24]=auxbuf[n];
                        }*/
```

A última parte do código trata da gravação do vetor buf[] no arquivo teste.wav armazenado no HD da máquina. O laço é repetido até o último pacote de 32 bytes ser gravado no arquivo. Quando o programa percebe o fim dos pacotes, devido ao encerramento da conexão, ele encerra o laço e solicita ao sistema operacional que toque o áudio recebido para que o usuário escute a mensagem enviada.

```
                        write(audio_escrita, buf, 32);

                }while (rval !=0);
                close (msgsock);
                close(audio_escrita);
                close(sock);
                system ("aplay teste.wav");

        }
        exit(0);}
```

Todas as rotinas que utilizam APIs, chamadas ao sistema operacional ou uso de drivers devem ser adaptadas se o programa for utilizado em outros sistemas. Por exemplo, se for desejado implementar esse programa numa placa de desenvolvimento, comandos relativos ao controle de um conversor analógico-digital devem substituir os comandos que usam o driver ALSA.

8. 4. TESTES

Apesar dos sinais enviados e recebidos soarem perfeitamente iguais, tentou-se realizar uma análise dos sinais, sem criptografia, com o uso do MATLAB, através da correlação dos sinais enviado e recebido. Porém ao tentar-se abrir o arquivo recebido no MatLab ocorreu uma mensagem de erro :"*Truncated chunk header found – possibly not a WAV file*", logo não foi possível realizar uma comparação mais sofisticada. O que pode-se assegurar é que os dois arquivos tem exatamente o mesmo tamanho e que não foi possível identificar diferenças através do aparelho auditivo humano. Também deve-se salientar que não houve erro ao executar nenhum dos arquivos em um *player* mais sofisticado, como o Windows Media Player.

O sistema criptográfico implementado foi bastante simples. A partir de uma chave secreta de um tamanho múltiplo dos pacotes que seriam enviados, faz-se uma operação binária de ou exclusivo (XOR) entre os dados e a chave. Esta chave é repetida até cifrar todo o pacote de dados. A operação XOR é perfeita para criptografias por chave simétrica, pois basta realizar outra operação XOR com a mesma chave no texto cifrado, que recupera-se o texto original.

Esse tipo de criptografia é suficiente para esconder informações presentes num texto mesmo usando uma chave de apenas um byte. Por exemplo, se trocar cada letra de um texto pela 4ª letra consecutiva no alfabeto (xor com 0b00000100), o texto já fica ininteligível e necessita uma análise para ser decodificado sem o conhecimento da senha.

Por outro lado, este trabalho ajudou a verificar que isso não se aplica com um sinal de áudio amostrado e codificado por amplitudes. Como a percepção do áudio depende também das variações do áudio e das características do sistema auditivo

humano, esse tipo de criptografia fraca não é suficiente. Alguns testes foram realizados com chaves de 1 ou 2 bytes e ainda foi possível escutar parte da mensagem.

Evidentemente quanto maior a chave, mais segurança tem o método criptográfico. Entretando chaves muito longas exigem maior processamento e isso pode denegrir o desempenho de aplicações que funcionam em tempo real. Alguns testes com chaves mais longas causaram uma distorção no áudio decodificado provavelmente devido a atrasos no processamento dos pacotes.

Um desempenho melhor foi conseguido utilizando uma chave de 8 bytes. Esta chave não alterou o desempenho do programa e conseguiu esconder a informação transmitida também no campo acústico. Para ilustrar o sistema criptográfico implementado, foi criada uma rotina em MATLAB que pode ser consultada no Apêndice A deste trabalho. Esta rotina consiste na gravação de um áudio nas mesmas condições de amostragem e quantização do programa e na criptofonia deste áudio usando a mesma chave de 8 bytes usada no programa. Seguem os gráficos obtidos comparando os dois sinais gerados. A frase usada neste exemplo foi: “Alô, teste. Bom dia. Som”. A frase teve uma duração de 2 segundos.

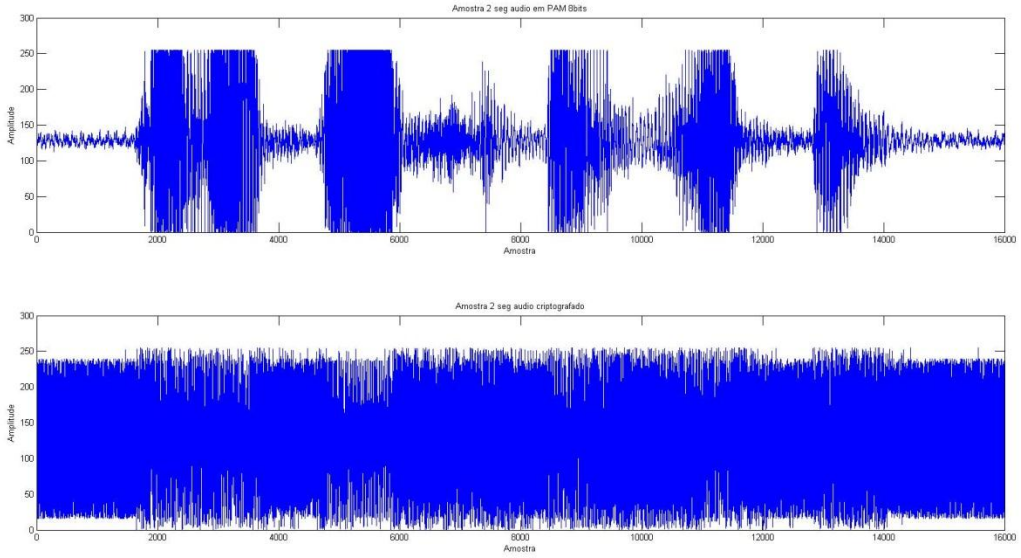

Figura 8.4.1 – Compação dos sinais criptografado e não criptografado

Um coeficiente de correlação próximo a 1 representa uma forte dependência linear entra as variáveis, enquanto coeficientes próximo a 0 significa que as duas

variáveis não dependem linearmente uma da outra. Aproveitando um pouco mais os recursos do MATLAB, foi calculado o coeficiente da correlação entre os dois sinais e chegou-se ao valor de 0.1315 para este teste, o que demonstra o poder dissimulativo de um método criptográfico tão simples.

9. CONCLUSÃO

Para a realização deste trabalho, buscou-se o desenvolvimento de um aplicativo que englobasse diversas áreas do curso de Engenharia de Comunicações e que servisse como um coroamento aos 5 anos de graduação. A ideia era que fosse algo simples, rápido, de fácil implementação e leve. Assim, além de programação, poderíamos estudar diversas áreas do interesse da engenharia no seu desenvolvimento, tal como Redes de Computadores, Redes de Acesso, Processamento Digital de Sinais, Princípios de Comunições Analógicas e Digitais, Codificação de Fonte e Canal, Teoria da Informação e Criptografia.

O programa desenvolvido cumpre a finalidade de servir como um VoIP em redes *adhoc* ou estruturadas e ao mesmo tempo é bastante leve e simples. Essas últimas características são necessárias se houver o interesse em adaptá-lo para um microcontrolador ou alguma outra interface. Acredita-se que futuramente seja interessante tentar adaptar o programa a uma placa de desenvolvimento do tipo Raspberry Pi ou similar para utilizá-lo como plataforma de um pequeno rádio baseado em Voip

De maneira geral, nosso objetivo foi atingido quando conseguimos estabelecer a primeira troca de mensagens entre dois computadores distanciados entre si. A partir de então, a ideia foi implementar melhorias, tal como criptografia, embaralhamento dos pacotes, modificar o programa para que funcione em tempo real, desenvolver uma interface gráfica, etc. Entretanto, no tempo que tivemos, conseguimos implementar com qualidade apenas o processo criptográfico usando chaves de até 8 bytes. Portanto, ainda há bastante possibilidade de trabalho para que se dê continuidade a esse programa.

10. REFERÊNCIAS BIBLIOGRÁFICAS

ARAÚJO, A.; ESTEVES, L.; ABREU, L. Transmissão de sinal de voz comprimido e criptografado, 1999. Trabalho de Iniciação a Pesquisa – Instituto Militar de Engenharia, Rio de Janeiro, 1999. [Orientadores: Cel/R1 Roberto M. Filho, Cap Ronaldo M. Salles].

BARROS, Thiago. Internet completa 44 anos; relembre a história da web. Disponível em: <http://www.techtudo.com.br/artigos/noticia/2013/04/internet-completa-44-anos-relembre-historia-da-web.html>. Acesso em 08 de setembro de 2016.

BEZERRA, Romildo. A Camada de Transporte. Disponível em: <http://www2.ufba.br/~romildo/downloads/ifba/transporte.pdf>. Acesso em 13 de setembro de 2016.

CASTELLÓ, T. VAZ, V. História da Criptografia. Disponível em: <http://www.gta.ufrj.br/grad/07_1/ass-dig/HistriadaCriptografia.html>. Acesso em 14 de setembro de 2016.

CAYONI, Paula. Uma breve história sobre Criptografia. Disponível em: <https://cryptoid.com.br/banco-de-noticias/a-historia-da-criptografia/>. Acesso em 14 de setembro de 2016.

DAMAS, Luís. Linguagem C. 10ª edição. Rio de Janeiro, 2007.

HAYKIN, Simon. Sistemas de Comunicações Analógicas e Digitais. 4. ed., Bookman, Porto Alegre, 2004.

HEITLINGER, Paulo. A evolução da rede chamada internet. Disponível em: <http://www.tipografos.net/internet/internet-evolucao.html> Acesso em 08 de setembro de 2016.

IMRE, Simon. A arpanet. Disponível em: <https://www.ime.usp.br/~is/abc/abc/node20.html>. Acesso em 08 de setembro de 2016.

KUROSE, J. F. e Ross, K. - Redes de Computadores e a Internet - 5ª Ed., Pearson, 2010.

LOPES, Petter. Camada de Aplicação TCP/IP. Disponível em: <https://petterlopes.wordpress.com/2011/07/18/camada-de-aplicacao-modelo-tcpip/>. Acesso em 12 de setembro de 2016.

PISA, Pedro. O que é criptografia?. Disponível em: <http://www.techtudo.com.br/artigos/noticia/2012/06/o-que-e-criptografia.html>. Acesso em 14 de setembro de 2016.

PIXININE, Juliana. Entenda o VoIP, tecnologia que permite apps ligarem pela Internet. Disponível em: <http://www.techtudo.com.br/noticias/noticia/2015/03/entenda-o-voip-tecnologia-que-permite-apps-ligarem-pela-internet.html>. Acesso em 12 de setembro de 2016.

RAPPAPORT, Theodore S. Comunicações sem fio: príncipios e práticas. 3a Ed. São Paulo: Pearson Prentice Hall, 2009.

SOUZA, Fabiano. Monitoração de Desempenho de Voz sobre IP (VoIP). Disponível em: <http://www.teleco.com.br/pdfs/tutorialmondesvoip.pdf>. Acesso em 07 de agosto de 2016.

SOUZA, Wendley. VoIP. Disponível em: <http://brasilescola.uol.com.br/informatica/voip.html>. Acesso em 07 de agosto de 2016.

STATISTA. Leading social networks worldwide as of September, ranked by number of active users (in millions). Disponível em: <http://www.statista.com/statistics/272014/global-social-networks-ranked-by-number-of-users/>. Acesso em 13 de setembro de 2016.

TANENBAUM, Andrew S. Redes de Computadores. 4ª Ed. Rio de Janeiro: Campus (Elsevier), 2003.

TELECO. Codificação. Disponível em: <http://www.teleco.com.br/Curso/Cbcod/default.asp>. Acesso em 14 de setembro de 2016.

TELECO. Internet: Protocolos. Disponível em: <http://www.teleco.com.br/tutoriais/tutorialinter/pagina_2.asp>. Acesso em 12 de setembro de 2016.

TELECO. PTT no Celular II: Protocolos SIP e RTP. Disponível em: <http://www.teleco.com.br/tutoriais/tutorialpushtotalk2/pagina_3.asp>. Acesso em 16 de setembro de 2016.

TELECO. Telefonia IP: Introdução. Disponível em: <http://www.teleco.com.br/tutoriais/tutorialtelefoniaip/pagina_1.asp>. Acesso em 10 de setembro de 2016.

TELECO. Voz sobre IP I: Tecnologia e Protocolos. Disponível em: <http://www.teleco.com.br/tutoriais/tutorialvoipconv/pagina_4.asp>. Acesso em 10 de setembro de 2016.

WILHELMS, Brittany. Problems with VoIP Phone Systems. Disponível em: <http://www.callforwarding.com/blog/problems-with-voip-phone-systems/>. Acesso em 10 de agosto de 2016.

ZAPPAROLI, Agenor. A história. Disponível em: <http://www.agenorzapparoli.com.br/ahistoria.html>. Acesso em 07 de agosto de 2016.

APÊNDICE A – Rotina MATLAB

ROTINA:

```
clear all
Fs = 8000;
Nr=2*Fs;
y  = wavrecord(Nr, Fs, 'uint8');
x = 1:Nr;

senha(1)=hex2dec('AF');
senha(2)=hex2dec('12');
senha(3)=hex2dec('6F');
senha(4)=hex2dec('AB');
senha(5)=hex2dec('5B');
senha(6)=hex2dec('90');
senha(7)=hex2dec('55');
senha(8)=hex2dec('4A');

crip = [senha];
for n=1:(Nr/8-1)
    crip = [crip senha];
end

crip = uint8(crip');

c = bitxor(y,crip);

subplot(2,1,1)
plot(x,y);
title ('Amostra 2 seg audio em PAM 8bits')
xlabel ('Amostra')
ylabel ('Amplitude')

subplot(2,1,2)
plot(x,c);
title ('Amostra 2 seg audio criptografado')
xlabel ('Amostra')
ylabel ('Amplitude')
Y= double(y);
C= double(c);
Y= (Y-128)/128;
C= (C-128)/128;

correlacao= corr2(Y,C)

sound(Y);
sound(C);
```

APÊNDICE B – Programa Cliente

```
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <stdio.h>
#include <sys/time.h>
#include <time.h>
#include <stdlib.h>
#include <strings.h>

struct tm *t;
time_t It;
int primeira=0;

int audio_leitura;
unsigned char buffer[32], auxbuffer[32];
unsigned char senha[8]={0xAF, 0x12, 0x6F, 0xAB, 0x5B, 0x90, 0x55, 0x4A};

main(argc, argv)
int argc;
char *argv[];
{
while(1)
{
        int aux=0;
        int n, m;
        char nomeArquivo[100];
        FILE *arquivo;
        long tamanho;

        printf("Para gravar um audio tecle ENTER \n");
        printf("Ao final da gravacao, tecle CRTL+C \n");
        getchar();
        system("arecord teste.wav");

        int sock;
        unsigned short int i,j,k;
        short int dif=0;
        struct sockaddr_in server;
        struct hostent *hp, *gethostbyname();

        sock=socket(PF_INET, SOCK_STREAM, 0);
        if(sock < 0)
        {
                perror("abertura de socket");
                exit(0);
        }

        server.sin_family = PF_INET;
        hp = gethostbyname(argv[1]);
        if(hp==0)
        {
                fprintf(stderr, "%s: host desconhecido.", argv[1]);
                exit(2);
        }
        bcopy((char *)hp->h_addr, (char *)&server.sin_addr, hp->h_length);
        server.sin_port = htons(atoi(argv[2]));
        if (connect(sock, (struct sockaddr *)&server, sizeof server) < 0)
        {
                perror(" conexao de socket");
                exit(1);
```

```
	}

	arquivo = fopen("teste.wav", "r");

	if (arquivo != NULL)
	fseek(arquivo, 0, SEEK_END);

	tamanho = ftell(arquivo);
	audio_leitura = open("teste.wav", O_RDONLY);

	while (aux<(tamanho/32))
	{
		read(audio_leitura, buffer, 32);
		for (m=0; m<32; m++)
			auxbuffer[m]=buffer[m];
		for (m=0; m<8; m++)
		{
			buffer[m]=auxbuffer[m+24];
			buffer[m+8]=auxbuffer[m];
			buffer[m+16]=auxbuffer[m+8];
			buffer[m+24]=auxbuffer[m+16];
		}
		for (n=0; n<8; n++)
		{
			buffer[n]=buffer[n]^senha[n];
			buffer[n+8]=buffer[n+8]^senha[n];
			buffer[n+24]=buffer[n+24]^senha[n];
			buffer[n+16]=buffer[n+16]^senha[n];
		}
		if(write(sock, buffer, 32) < 0)
			perror(" escrita no socket");
		aux++;
	}

	close(sock);

}exit(0);}
```

APÊNDICE C – Programa Servidor

```
#include <sys/stat.h>
#include <fcntl.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>
#include <stdio.h>
#include <sys/time.h>
#include <time.h>
#include <stdlib.h>
#include <strings.h>

struct tm *t,*t2;
time_t lt,lt2;
int contatempo=0;
int primeira=0;

main (argc, argv)
int argc;
char *argv[];
{
while(1)
{
        int sock,i,j;
        struct sockaddr_in server;
        int msgsock;
        unsigned char buf[32], auxbuf[32];
        unsigned char senha[8]={0xAF, 0x12, 0x6F, 0xAB, 0x5B, 0x90, 0x55, 0x4A};
        int n;
        int rval;
        int length;
        int audio_escrita;

        FILE *f = fopen("teste.wav","w");
        audio_escrita = open("teste.wav", O_WRONLY);

        sock = socket(PF_INET, SOCK_STREAM, 0);
        if (sock < 0)
        {
                perror ("Abertura de socket errado. ");
                exit (1);
        }
        server.sin_family = PF_INET;
        server.sin_addr.s_addr = INADDR_ANY;
        server.sin_port = htons(atoi(argv[1]));

        if(bind (sock, (struct sockaddr *) &server, sizeof server) < 0)
        {
                perror ( "ligacao (bind) no socket errado");
                exit (1);
        }

        length = sizeof server;
        if(getsockname (sock, (struct sockaddr *) &server, &length) < 0)
        {
                perror ("obtencao do nome do socket errado");
                exit (1);
        }
        printf ("Socket port %d\n\r", ntohs (server.sin_port));

        listen (sock, 5);
        msgsock = accept (sock, (struct sockaddr *) 0, (int *) 0);
        if (msgsock == -1)
```

```
                perror ("Accept com erro");
        else
        {
                do
                {
                        i=0;j=0;
                        bzero (buf, sizeof buf);
                        if ((rval = read (msgsock, buf, 32)) > 0)
                                perror (" Lendo mensagem...");
                        if (rval == 0)
                                printf ("terminando conexao...\n\r");
                        for ( n=0; n<8; n++)
                        {
                                buf[n]=buf[n]^senha[n];
                                buf[n+8]=buf[n+8]^senha[n];
                                buf[n+16]=buf[n+16]^senha[n];
                                buf[n+24]=buf[n+24]^senha[n];
                        }
                        write(audio_escrita, buf, 32);
                }while (rval !=0);
                close (msgsock);
                close(audio_escrita);
                close(sock);
                system ("aplay teste.wav");
        }
}
exit(0);}
```