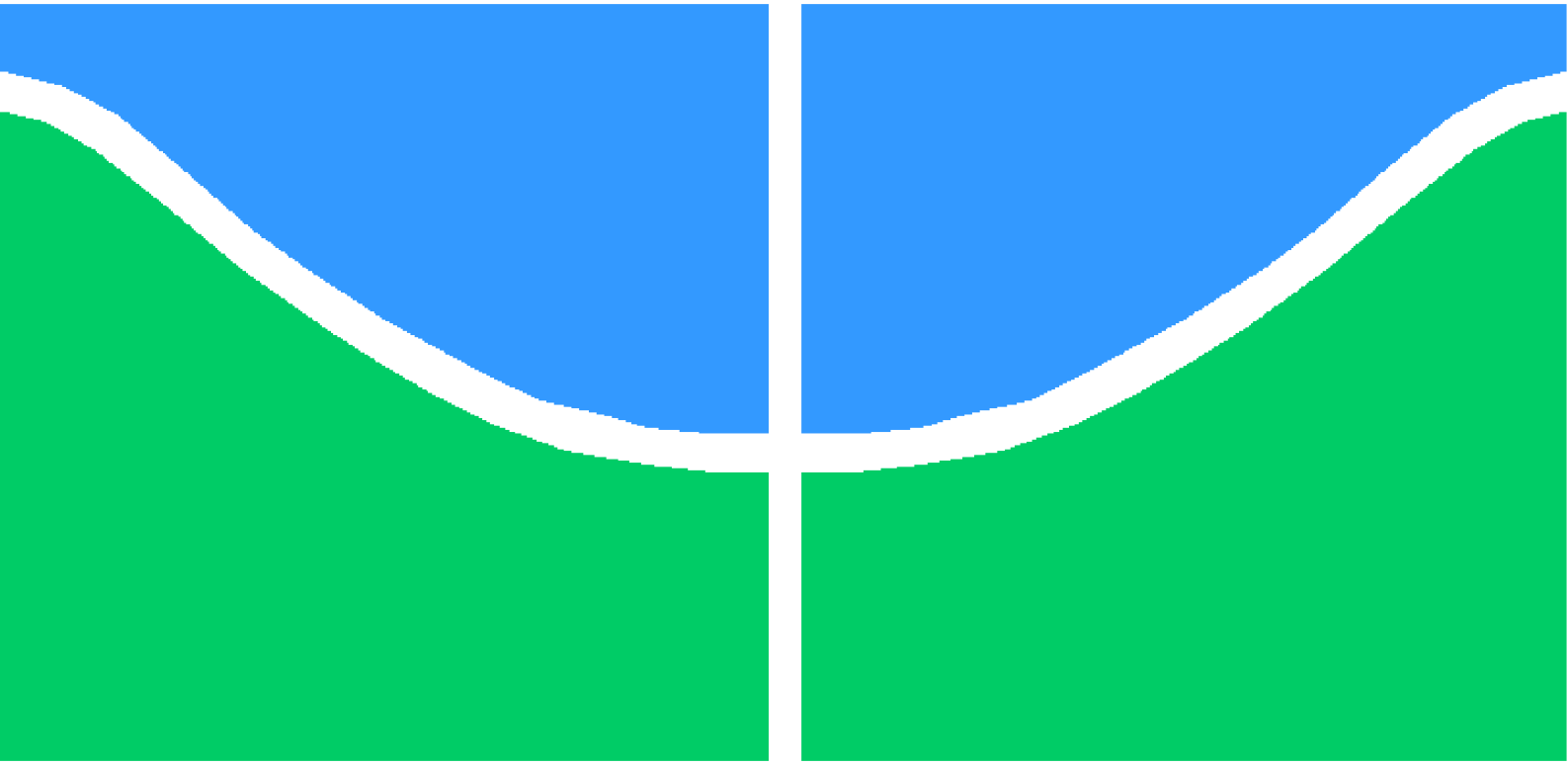

Universidade de Brasília - UnB

Faculdade UnB Gama - FGA

Engenharia de Software

Evolução da especificação de um formato de arquivo open source para audiobooks com suporte a marcações de conteúdo

Autor: Bryan de Holanda Fernandes

Orientador: Prof. Dr. Edson Alves da Costa Júnior

Brasília, DF

2014

Bryan de Holanda Fernandes

Evolução da especificação de um formato de arquivo open source para audiobooks com suporte a marcações de conteúdo

Monografia submetida ao curso de graduação em *Engenharia de Software* da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Software.

Universidade de Brasília - UnB

Faculdade UnB Gama - FGA

Orientador: Prof. Dr. Edson Alves da Costa Júnior

Brasília, DF

2014

Bryan de Holanda Fernandes

Evolução da especificação de um formato de arquivo open source para audiobooks com suporte a marcações de conteúdo/ Bryan de Holanda Fernandes. – Brasília, DF, 2014-

61 p. : il. (algumas color.) ; 30 cm.

Orientador: Prof. Dr. Edson Alves da Costa Júnior

Trabalho de Conclusão de Curso – Universidade de Brasília - UnB
Faculdade UnB Gama - FGA , 2014.

1. Audiobooks. 2. Vorbis Ogg. I. Prof. Dr. Edson Alves da Costa Júnior. II. Universidade de Brasília. III. Faculdade UnB Gama. IV. Evolução da especificação de um formato de arquivo open source para audiobooks com suporte a marcações de conteúdo

CDU 02:141:005.6

Bryan de Holanda Fernandes

Evolução da especificação de um formato de arquivo open source para audiobooks com suporte a marcações de conteúdo

Monografia submetida ao curso de graduação em *Engenharia de Software* da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Software.

Trabalho aprovado. Brasília, DF, 01 de junho de 2013:

Prof. Dr. Edson Alves da Costa Júnior
Orientador

Convidado 1

Convidado 2

Brasília, DF
2014

Agradecimentos

A Deus acima de tudo e por tudo que tem feito em minha vida. Pelas oportunidades que Ele tem concedido a mim. Agradeço a minha família e amigos pelo apoio, carinho e consideração.

"Nada façais por contenda ou por vanglória,
mas por humildade;
cada um considere os outos superiores a si mesmo.
Não atente cada um para o que é propriamente seu,
mas cada qual também para o que é dos outros.
(Bíblia Sagrada, Filipenses 2, 3-4)

Resumo

Neste trabalho é apresentada um proposta de evolução do trabalho proposto por (REIS, 2013) e, para tal, será desenvolvido uma especificação de um formato livre e *open source* com o objetivo de oferecer suporte de marcação de conteúdo para *audiobooks*. Um Editor é desenvolvido no qual será responsável por codificar e decodicar os metadados de um arquivo do tipo Ogg Vorbis bem como inserir um novo pacote contendo as marcações de conteúdo. Tal proposta foi construída em cima das especificações do formato Ogg Vorbis e nas bibliotecas fornecidas pela fundação Xiph.org.

Palavras-chaves: *Audiobooks*. Marcação de conteúdo. Ogg Vorbis. *Open Source*.

Abstract

This paper presents a proposal for development of work proposed by (REIS, 2013), and for this, will be developed a specification of a free and open source format with the goal of providing support for marking content for audiobooks. An Editor is developed in which will be responsible for encode and decode the metadata of a file type Ogg Vorbis and insert a new package containing the markings content. This proposal was built on the specifications of Ogg Vorbis and libraries provided by the foundation Xiph.org.

Key-words: *Audiobooks*. Bookmarking Content. Ogg Vorbis. *Open Source*.

Lista de ilustrações

Lista de tabelas

Sumário

1 Introdução

Os *audiobooks* tem crescido no mercado e cada vez mais sendo usado pelas pessoas ao redor do mundo. Eles são amplamente difundidos pela internet e em sua maioria no formato MP3. Este formato tem sido difundido por armazenar áudio de alta qualidade e ocupar pouco espaço em memória devido ao uso de uma técnica de compressão de dados. Entretanto, com desvantagem, quaisquer marcações são ligadas diretamente ao tocador e não ao arquivo. Outro ponto é que não existem formatos *open source* que ofereçam suporte para marcação de conteúdo e os que possuem são protegidos por leis de propriedade intelectual. Estas marcações são marcadores de posição que representam a estrutura lógica de um livro físico o que vem a ser capítulos, seções, parágrafos, versículos, entre outros. Estas são as informações levantadas por (REIS, 2013) e a motivação de seu trabalho foi desenvolver uma especificação de um formato aberto para *audiobooks* com suporte a marcadores de conteúdo pois não existe um formato de arquivo livre e de código aberto que tenha suporte a estes marcadores. Os objetivos foram alcançados mas melhorias foram sugeridas pois os arquivos gerados ocupavam muito espaço em memória.

Este trabalho visa reproduzir o trabalho feito por (REIS, 2013) mas em um outro formato de arquivo que faz compressão de dados ao ponto de gerar arquivos menores viabilizando a proposta incialmente sugerida. Portanto, neste trabalho foi desenvolvida a especificação de um formato aberto para *audiobooks* com suporte a marcadores de conteúdo e que faz uso da técnica de compressão de dados para o armazenamento dos dados.

O trabalho está organizado em capítulos. No próximo capítulo, é apresentado os objetivos pretendidos para o qual este trabalho foi motivado.

No capítulo 3, é apresentada toda a revisão bibliográfica onde foram revistos monografias, dissertações, livros, publicações em *websites* e especificações necessários para o entendimento e desenvolvimento do projeto.

No capítulo 4, será restringida a proposta do trabalho.

No capítulo 5, serão apresentados todas as etapas realizadas para a especificação do formato e para o desenvolvimento do Editor bem como as ferramentas utilizadas com suporte no processo de desenvolvimento e pesquisa.

No capítulo 6, serão apresentados os resultados obtidos no projeto onde será mostrado o arquivo gerado pelo Editor.

E, por fim, no capítulo 7, será apresentado um cronograma onde estão definidas as atividades executadas durante o desenvolvimento deste trabalho.

2 Objetivo

Nesta seção serão descritos os objetivos que se pretende alcançar com este trabalho e estão divididos em objetivo geral e objetivos específicos.

2.1 Objetivo Geral

Evoluir o trabalho proposto por (REIS, 2013) que especifica um formato de arquivo open source para audiobooks com suporte a marcação de conteúdo e construção de um tocador para o novo formato especificado.

2.2 Objetivo Específico 1

Analisar todo o trabalho desenvolvido com o intuito de identificar problemas encontrados no formato de arquivo especificado e propor melhorias para uma melhor aceitação no ambiente computacional atual.

2.3 Objetivo Específico 2

Construir um tocador capaz de decodificar e executar o novo formato especificado ou melhorado sem interrupção ou perda de informação sendo possível inserir marcações de conteúdo de áudio e pular a execução para os pontos marcados.

3 Fundamentação Teórica

Neste capítulo estão as pequisas e todo o embasamento teórica referente a literatura acerca de trabalhos já desenvolvidos que dão suporte e direcionam o tema deste trabalho. As seções contidas neste capítulo abordam temas referentes a *audiobooks*, formato de áudio, algoritmos de compressão e também traz uma contextualização sobre a viabilidade e aceitação do tema proposto.

3.1 Audiobooks

Podemos dizer que *audiobook* é a versão em áudio de livros impressos. Também conhecido como "livro falado", o *audiobook* tem sido conhecido aos poucos aqui no Brasil e sido utilizado como suporte para que pessoas que não possuem o hábito de ler passam a conhecer obras literárias. O audiobook é uma forma inovadora de acesso à leitura (SOUZA; CELVA; HELVADJIAN, 2006). Mesmo para aquelas pessoas que possuem o hábito de leitura, este hábito vem se perdendo devido ao ritmo acelerado das grandes cidades do mundo de hoje. Por falta de tempo, as pessoas tem feito uso de *audiobooks* para estarem "lendo" enquanto passam horas dirigindo em extensos engarrafamentos ou enquanto fazem exercícios físicos (PALLETA; WATANABE; PENILHA, 2008). Isso é possível pois a portabilidade do audiobook em comparação aos livros impressos é muito maior, pois o local não parece importar.

Desde 1950, a apreciação pela literatura falada vem crescendo e se tornando tradição nos Estados Unidos. Mas foi em 1980 que realmente os audiobooks ganharam corpo. Segundo (TEIXEIRA, 2006) havia, em 2006, cerca de 30 mil títulos de audiobooks nos Estados Unidos apenas no site da Audible, companhia da Amazon. Neste ano, a companhia possui mais de 150 mil títulos disponíveis para download (AUDIBLE, 2014). Este número mostra como o mercado de *audiobooks* vem crescendo no mercado norte-americano.

O crescimento do uso de *audibooks* não tem ocorrido apenas nos Estados Unidos. Na Europa, os livros falados também tem ganhado força, sendo mais popular na Alemanha. Neste país, o hábito virou moda na década de 90. Podemos citar o Instituto Goethe que é um instituição sem fins lucrativos que visa a disseminação do idioma e cultura alemã. Eles possuem *streaming* de diversas obras literárias alemãs disponíveis em seu site. Dois eventos populares na Alemanha, a grande Feira Internacional de Livros de Leipzig e o Festival Internacional de Literatura de Berlim, contam com estandes de venda voltados à apresentação de *audiobooks* (DW, 2004).

Oscar Niemeyer, antes mesmo do uso dos CDs, disse "Quando quero ler, eu ouço.

Pago uma pessoa para gravar os livros em fitas e depois, quando sinto vontade, as coloco para tocar" (PALLETA; WATANABE; PENILHA, 2008). Aqui no Brasil, o uso de *audiobooks* começou um pouco mais tarde e em um ritmo um pouco mais lento se comparado com Estados Unidos e Europa. No entanto, nos últimos anos os livros falados tem sido cada vez mais utilizados pela população brasileira. O mercado brasileiro tem correspondido a esta nova disseminação cultural (FARIAS, 2012). Isto mostra que a cultura de livros falados no Brasil veio para ficar. Plugme e Audiolivro Editora sao editoras que tem investido em audiolivros no Brasil. Em 2008, quando o mercado ainda era considerado tímido, a Plugme registrou uma venda de setecetnos livros por mês contra mil livros da Audiolivro (SOUZA; CELVA; HELVADJIAN, 2006). A Audiolivro conta com mais de 900 títulos. Voltado para a literatura infantil, a editora RHJ reúne mais de 200 publicações e premiações recebidas no exterior tais como *White RAvens* na Alemanha, *Octogone* na França, *BIB Plaque* na Eslováquia, *The Noma Concours* no Japão e *The Ibby Honour List Diploma* no Canadá. A editora RHJ também possui premiações no Brasil (RHJ, 2009).

A Universidade Falada é um portal de iniciativa privada que visa difundir cultura pelo Brasil com distribuição de conteúdo em áudio viabilizado pela *Editora Alyá* com preços acessíveis e facilidade para a aquisição. Este projeto conta com mais de mil e trezentos audiolivros e estimam mais de cinco mil horas de áudio disponíveis em formato mp3. (FALADA, 2004).

> O projeto Universidade Falada® pretende democratizar a cultura. Facilitar o acesso, em formato audio e audiolivro, a grandes obras da literatura nacional e internacional à população mais afastada dos grandes centros culturais do país. Nossa missão é ajudar pessoas, oferecer conhecimento e cultura. Discutir temas velhos e novos, ensinar e filosofar. Agregar valor ao ser humano. É isso que nós editores, autores e palestrantes desejamos desta empreitada. Nossos preços permitem que qualquer brasileiro com acesso a internet possa adquirir nossos produtos. Sem exceção (FALADA, 2004).

Em outubro deste ano, a *PublishNews* publicou uma matéria informando a chegada de duas plataformas de audiolivros para smartphones e tablets no mercado brasileiro. A UBook é uma delas e já está disponível para plataforma iOS e Android. Adotando o serviço de subscrição, os usuários possuem acesso ilimitado a mais de mil obras. Segundo os desenvolvedores, cinco dias após seu lançamento, a plataforma já possuia mais de vinte mil usuários. Uma outra alternativa é a TocaLivros que optou pela venda unitária do audiolivro em formato digital (NETO, 2014). Esta pretende lançar sua plataforma em algumas horas como mostra a Figura 1.

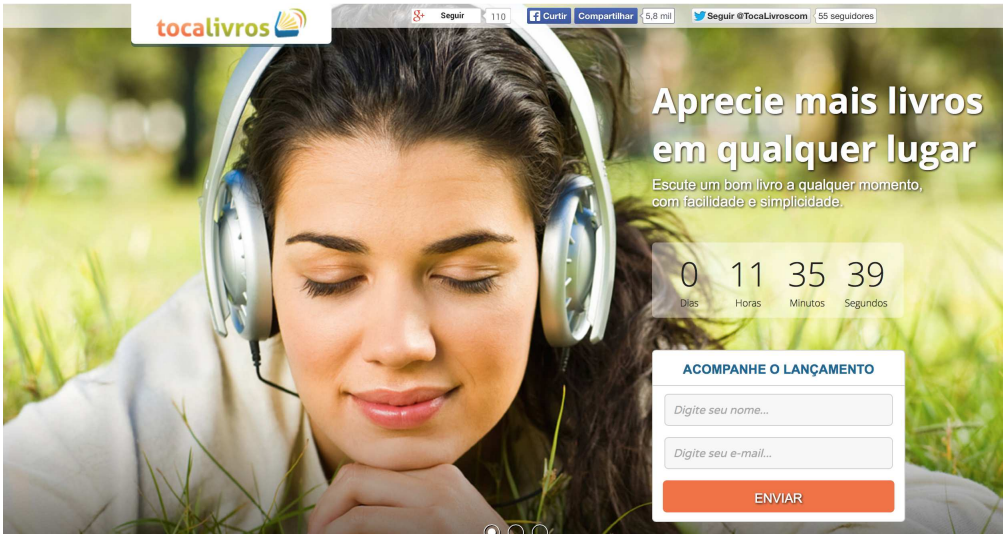

Figura 1 – Captura de tela tirada às 23:24:04 do dia 11/11/14 (TOCALIVROS, 2014)

3.2 Formatos de Áudio Digital

O áudio digital refere-se a representação do som (captação por microfone) ou áudio analógico (representado por um vinil ou fita cassete) que é armazenado na forma de um arquivo digital em uma unidade de computador ou outra mídia digital (como *Compact Disc*). O som é contínuo no tempo e para conseguir converter este sinal analógico em sinal de áudio digital, o conversor analógico-digital consiste em "tirar fotos" do sinal de áudio em intervalos constantes. Portanto, estas "fotos" são uma sequência de amostras que é representado por um sinal de tempo discreto originado da conversão de uma onda de som que, por sua vez, é um sinal contínuo. Assim, definimos **amostragem** como sendo a redução de um sinal contínuo em um sinal discreto. Em cada uma destas amostras é medido a intensidade do sinal que pode ser representada por 0s e 1s. Ou seja, um áudio digital é a representação binária de um som que é traduzida por um computador ou leitor de CD. Estes são capazes de reproduzir o som original.

Como vimos, o formato de áudio é uma forma de representar digitalmente um som ou um sinal de áudio. Algumas propriedades gerais de representação de som digital são:

***bitrate* ou taxa de bits:** que é o número de bits por segundos necessários para representar o sinal. O *bitrate* descreve, então, a velocidade com que o som é reproduzido (por exemplo, 320 000 bits por segundos ou 320 Kbps);

***number of channels* ou número de canais:** usa-se, tradicionalmente, um canal (som mono) ou dois canais (som estéreo);

***sampling frequency* ou frequência de amostragem:** define o número de amostras que são coletadas em um tempo pré determinado e a unidade de medição é dada

em *Hertz*. As amostras são medidas em intervalos fixos.

Com o avanço e popularidade da digitalização do som, o áudio digital tem sido largamente usado. Um dos principais motivos é a facilidade no uso deste tipo de arquivo, pois trabalhar com um sinal digital é muito mais simples se comparado ao sinal analógico. Os formatos são geralmente conhecidos por sua extensão e não pelo seu nome. Existem diversos formatos de áudio digital, mas primeiro vamos falar sobre o PCM.

3.2.1 PCM

Com seu surgimento na década de 30 por Alec Reeves, o PCM (*do inglês, Pulse Code Modulation - Modulação por Código de Pulso*) é a forma mais antiga de digitalização do som e o mais utilizado devido à reprodução fiel do som, que é armazenado. Isso significa que os dados foram levados diretamente a partir da entrada, digitalizada e armazenada sem transformação. Não havia compressão de dados na época pois o poder de computação era escasso (WAGGENER, 1994).

Como explanado no início desta seção, a digitalização não é contínua, ou seja, não é constante como o som. Por possuir valores discretos (descontínuos), a digitalização sonora envolve dois parâmetros básicos: a frequência de amostragem e a profundidade de bit. A **profundidade de bit** (*bit depth*) trata da quantidade de bits de computador que são usados para representar cada amostra. Uma representação com um bit recebe apenas dois valores (0 e 1). Se tenho uma representação com 4 bits é possível receber 16 valores diferentes onde a amplitude de cada amostra é um dos 16 valores possíveis (WAGGENER, 1994). A taxa de amostragem e a amplitude influenciam diretamente na qualidade do áudio. Sony e Philips desenvolveram a tecnologia PCM e criaram o *Compact Disc*. O CD possui 44100 amostras por segundo e uma amplitude de 16 bits o que torna a qualidade alta.

3.2.2 Formatos não comprimidos

Os formatos não comprimidos possuem qualidade máxima pois não há alteração dos bits do ficheiro de áudio. Isto faz dos formatos não comprimidos útil para aplicações profissionais. O áudio digital é armazenado sem realizar a compressão dos dados e, portanto, demandam grande espaço em memória.

3.2.3 Formato WAV

O formato *Waveform Audio Format* ou WAVE (os ficheiros deste formato utilizam a extensão *.wav) possui o seu áudio codificado em PCM. Criado pela Microsoft, este formato de áudio digital é nativo do sistema operativo Windows e é compatível com

quase todos os tocadores atuais. Um ponto negativo é o tamanho de seu arquivo que chega a possuir uma média de 10 MB por minuto (SERRA, 2002).

O WAV é derivado do padrão IFF criado pela *Eletronic Arts Interchange File Format* onde seu conteúdo é divido em blocos denominados *chunks*. Existem diversos tipos de *chunks* em um arquivo WAV, mas apenas dois são obrigatórios (além do chunk *header* que especifica o formato do áudio).

fmt chunk : define algumas informações sobre o formato do arquivo tais como número de canais, sua taxa de amostragem e o tamanho de seus *samples*;

data *chunk* : responsável por armazenar a sequência de *samples*.

A Figura 2 ilustra o que foi explicado sobre a estrutura geral de um formato WAVE.

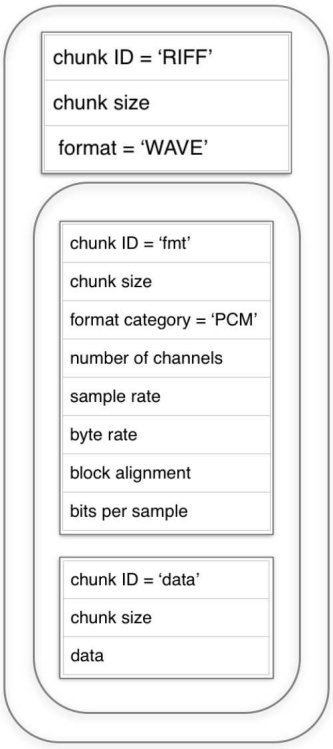

Figura 2 – Diagrama Geral do formato WAVE.

3.2.4 Formato AIFF

O formato *Audio Interchange File Format* (a extensão destes ficheiros pode ser *.aiff ou *.aif) é bastante similar ao WAV. Foi criado pela Apple e portanto é bastante popular e nativo em seu sistema operacional. O AIFF possui áudio em formato PCM porém não é tão difundido quanto o formato WAV. Como o nome sugere, o AIFF também usa o método IFF para armazenar seus dados e também possui dois *chunks* obrigatórios:

o *common chunk* e o *sound data chunk* (MURATNKONAR, 2014). A Figura 3 ilustra, de forma geral, como um formato AIFF de áudio é estruturado.

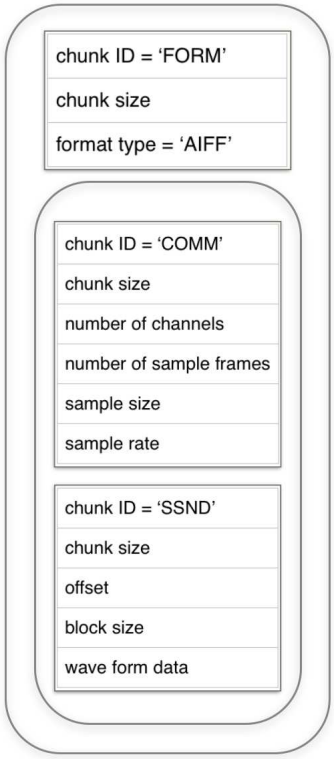

Figura 3 – Diagrama Geral do formato AIFF.

***common chunk*:** define algumas informações fundamentais do formato do como visto na Figura 3;

sound data *chunk*: responsável por armazenar a sequência de *sample frames*.

3.2.5 Formatos comprimidos

Os formatos comprimidos possuem qualidade inferior aos formatos não comprimidos. Todavia, dependendo da compressão feita, mesmo com um pouco da perda da qualidade, a alteração é imperceptível. Por conta da compressão é possível obter um ficheiro de áudio com qualidade boa e com o espaço ocupado em memória reduzido.

3.2.6 Formato MP3

O MPEG-1 Layer 3 (popularmente conhecido com MP3) é a primeira versão do codec MPEG que é utilizada, atualmente, para codificar áudio. O MPEG, do inglês *Moving Picture Experts Group* é um padrão que define técnicas de compressão de áudio e vídeo. A Camadas (*Layers*) diferem-se em termos de qualidade, sendo as primeiras dotadas de

qualidades superiores e consequentemente voltadas para uso profissional de áudio como em estúdios de gravação. A Camada 3 foi adotada para o consumidor final por ser capaz de gerar arquivos com boa taxa de compressão e áudio de boa qualidade. O MP3 é construído a partir de pequenas partes chamadas *frames*. Cada frame, por sua vez, é composto por um *header* e um bloco de dados (ERROR, 1999). A Figura 4 melhor define esta estrutura.

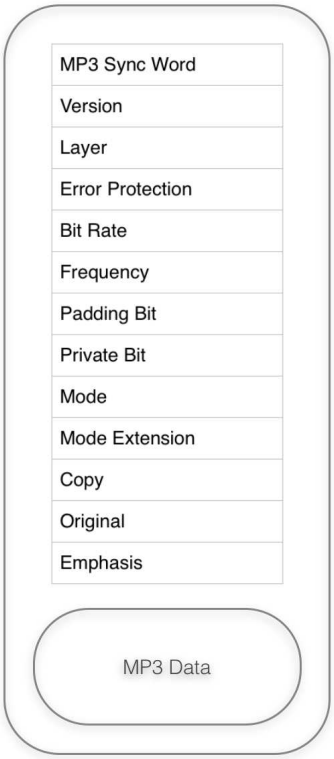

Figura 4 – Diagrama Geral representando um *frame* do formato MP3.

O MP3 se popularizou através de um software criado por Shawn Fanning chamado Napster. Apesar da rede Napster ter tido uma vida curta (cerca de 2 anos), o programa fez bastante sucesso sendo o percursor do compartilhamento de arquivos de áudio em formato MP3. Por conta da dificuldade em encontrar áudios em formato MP3, Fanning criou o programa para compartilhar estes ficheiros de áudio entre amigos. Usando da tecnologia P2P (*peer to peer*) e pela popularidade entre os amigos, o programa foi copiado para diversas outras pessoas alcançando pessoas ao redor do mundo (ALECRIM, 2006). Dentre os formatos de áudio, o MP3 é o mais popular. O grande problema dele é sua patente.

3.2.7 Formato Ogg Vorbis

Será visto na seção 3.4.

3.3 Compressão de Dados

A compressão de dados é uma técnica utilizada para que um arquivo de dados ocupe um menor espaço em memória física, reduzindo assim a dimensão física de blocos de informação. Esta compressão se dá pela diminuição da quantidade de bits ou até mesmo de bytes que representam um dado. Todos os dados computacionais como texto, imagens, músicas e vídeos são compostos por uma série de bits, sendo o bit uma unidade atômica de armazenamento, que podem ser representados por bytes que consistitui-se do conjunto de oito bits (TENENBAUM et al., 2004). Um dado pode ser comprimido sem nenhuma perda, por meio de algoritmos de compressão, retirando-se informações redundantes. É percepitível o motivo de compressão para economizar espaço em dispositivos de armazenamento. No entanto, os dados também são comprimidos para um ganho de desempenho em transmissões. Sayood, atual professor da Universidade de Nebraska-Lincoln, escreveu:

> Compressão, redução da taxa de bits e redução de dados têm basicamente o mesmo significado, ou seja, quer se obter a mesma informação porém utilizando menor quantidade de dados. Há vários motivos para que a compressão se torne popular. Podemos citar a diminuição da quantidade de memória necessária para o armazenamento (hardware menor), o menor custo devido a uma menor largura de banda requerida para a transmissão da mesma quantidade de dados e a facilidade nas transmissões em tempo real (SAYOOD, 2012).

Além da redundância, um outro fator que pode ser levado em consideração é a irrelevância da informação. Diferente do caso anterior, a irrelevância necessariamente ocasionará em perda de informação. Existem diversos métodos de compressão de dados. Apesar da existência de outros, a forma mais conhecida de se classificar métodos de compressão de dados é pela ocorrência ou não de perda de dados durante o processo. Apesar do computador manipular todas as informações por bits, o método de compressão depende intrinsecamente do tipo de dado a ser comprimido (ASSIS, 2010).

A compressão sem perda de dados tira partido da redundância. Este tipo de compressão deve ser capaz de reconstruir, após a descompressão por meio de algoritmos de compressão, a informação original sem nenhuma perda de informação. Podemos citar como exemplo os ficheiros executáveis. Este tipo de dado deve ser recuperado sendo necessário a conservação da sua integridade para que funcione corretamente. Ou seja, este método de compressão é reversível sendo possível reconstruir o dado mantendo sua integridade e a mesma qualidade.

A compressão com perda de dados tira partido da redundância e da irrelevância. Este tipo de compressão não é capaz de reconstruir, após a descompressão por meio de algoritmos de compressão, a informação original. Porém, é suficientemente parecida para que seja útil de alguma forma ou sua diferença imperceptível. Este tipo de método de compressão é comumente utilizado para arquivos de áudio e vídeo. Os dados multimídia

podem tolerar um certo nível de degradação sem que a percepção humana, como olho e tímpano, distingam uma degradação significativa. Para exemplificar, as frequencias de som em que a audição humana não é capaz de captar é eliminado por este tipo de compressão. O arquivo passa a ter uma qualidade reduzida porém uma alta taxa de compressão.

3.3.1 Compressão de áudio

Como explanado nos tópicos anteriores, é de suma importância comprimir dados e com o áudio não é diferente. Com a compressão de áudio há uma melhoria no fluxo de dados minimizando os esforços computacionais nas tarefas de transmissão de dados e armazenamento. Isto facilita a execução de serviços de comunicação em tempo real e gera um aumento de desempenho das aplicações (BRAGA, 2003).

Geralmente, a compressão de áudio é feita com perda de dados. Numa música, pode ocorrer uma redundância de informação a partir do momento em que longos períodos de amostras de som possuem o mesmo valor. Para eliminar a repetição desnecessária de informação, por exemplo, os trechos das amostras de som de mesmo valor poderiam ser substituídos por um pequeno código dizendo que a mesma frequência deve ser repetida um número determinado de vezes. Outro tipo de redundância é os momentos de silêncio num sinal de áudio. Até certo ponto é possível comprimir o som sem nenhuma perda de qualidade, mas, para que possamos comprimir ainda mais um ficheiro de áudio, podemos abrir mão de uma pouco da qualidade de áudio para geração de arquivos ainda menores. No entanto, além das técnicas habituais de compressão, aproveita-se o conhecimento das imperfeições ou limitações na audição. Usando da irrelevância da informação e com prévio conhecimento das características da audição humana, podemos eliminar certas informações sem afetar o que ouvimos. Segundo (BRAGA, 2003) “Porém, biologicamente, podemos afirmar que o ouvido humano somente responde a certas faixas de frequência”. Ou seja, existe uma pequena perda da qualidade do áudio mas são inaudíveis para o ser humano. Assim podemos afirmar que a qualidade é mantida, pois “Por definição, o som de qualidade de um codificador só pode ser determinado pelo ouvido humano.” (BRAGA, 2003).

3.3.2 Porque comprimir os dados?

Com o avanço tecnológico, a manipulação de dados tem exigido cada vez mais poder de processamento e capacidade de armazenamento. Com o crescente desenvolvimento das aplicações que manipulam dados multimídia para entretenimento dos usuários, há um aumento da quantidade de informação que é passada através das redes. Estes tipos de dados, tais como som, imagem e vídeo, ocupam grande espaço em disco o que dificulta o uso das redes para compartilhamento destes dados. Considere um áudio codificado a uma taxa de amostragem de 44.1Hz, 16 bits por amostra e estéreo. Este áudio possui

qualidade de CD. Vamos definir a largura de banda necessária para a transmissão deste áudio. Para tal, basta multiplicar os valores destas três informações (FOROUZAN, 2006). Se multiplicarmos a frequência de amostragem pelos bits de cada amostra, nós teremos o número de bits por segundo. No entanto, ainda devemos multiplicar este valor por dois, pois o áudio está no modo estéreo. Este modo utiliza dois canais de som sincronizados no tempo. Assim, teremos o valor de 1.411.200 bits por segundo. Isto significa que para transmitir tal arquivo por uma rede, é necessária uma largura de banda de 1,41 Mbits/s. Para sabermos qual o espaço utilizado por este áudio para armazená-lo em um computador precisamos, primeiramente, saber qual a sua duração. Se dissermos que o áudio possui 180 segundos de duração e temos 1.411.200 bits por segundo, estimasse que este áudio possui 254.016.000 bits o que é equivalente a um pouco mais de 30MB (FOROUZAN, 2006). Este dois exemplos dão uma ideia da importância da compressão de áudio.

Além dos fatores espaço de armazenamento e compartilhamento de dados através das redes, estes dados também tem grande influência no que tange o desempenho das aplicações. O poder de processamento tem crescido mais rapidamente do que as capacidades de armazenamento e desempenho no uso da memória. Em 1965, Gordon Earl Moore previu que o poder de processamento dos computadores dobrariam a cada 18 meses. Essa observação ficou conhecida como Lei de Moore. (DISCO; MEULEN, 2012) e (TANENBAUM, 2007). A Figura 5 informa dados pontuais onde é possível observar a evolução da quantidade de componentes por chip para os microprocessadores da Intel entre 1970 e 2005.

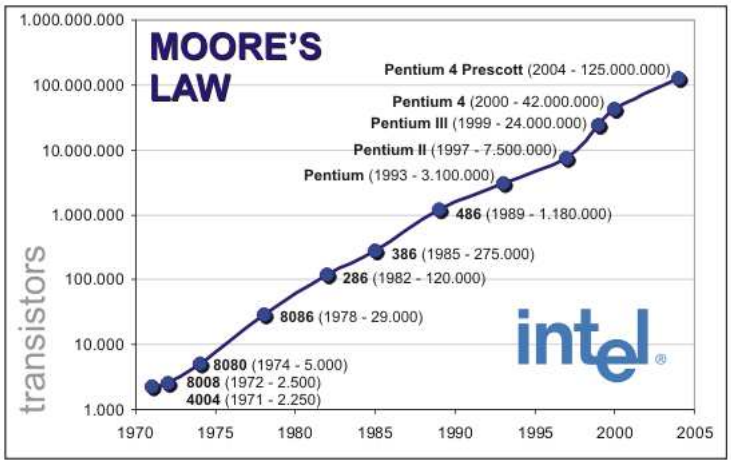

Figura 5 – Lei de Moore em ação para microprocessadores Intel (VICTOR, 2014)

Apesar do poder de processamento ter crescido exponencialmente, ele não é o único fator que influencia no desempenho de uma aplicação. Um computador, basicamente, trabalha com dados e para manipulá-los o processador faz requisições a memória, seja para gravar algum dado em disco ou obter dele. Estas requisições feitas ao disco leva um tempo considerado altíssimo e isto reduz o desempenho de uma aplicação apenas pelo

acesso feito a memória (TANENBAUM, 2007). Ou seja, o processador acaba ficando ocioso esperando as instruções ou operandos que estão sendo buscados na memória durante a execução de instruções. A capacidade de armazenmento também está ligada a taxa de transferência de dados. Quanto maior a capacidade, maior será o tempo de acesso. Outro fator limitante para o uso de memórias com capacidades cada vez maiores é o seu custo. Para que um sistema seja comercialmente viável, o custo da memória deve ser compatível com os demais componentes (TANENBAUM, 2007).

De acordo com (FAGUNDES, 2014), entre os anos 2012 e 2017 os recursos por mais processamento, armazenamento de dados e comunicações poderá triplicar. Este crescimento de dados foi atribuído, representando 50% do aumento, ao crescimento de streaming de vídeo e áudio. Em horários de pico, 50% do tráfego de Internet são dados trafegados do Netflix e Youtube (FAGUNDES, 2014).

A compressão dos dados é importante para a redução de dados como áudio, imagens e vídeos em situações como transmissão de dados ou armazenamento, pois é ideal a diminuição do tempo de latência. Desta forma, o custo para transmissão dos arquivos na rede ou para a taxa de transfência para a/da memória será menor. O problema de economia de memória, apesar das memórias terem se tornado maiores se comparadas em épocas passadas, ele ainda permanece nos dias de hoje. Como vimos, os custos para acessar os dados em memória ainda é alto, ocasionando na perda de desempenho (TANENBAUM, 2007). A compressão de dados é um dos fatores que mais contribuiu para o grande cescimento das tecnologias da informação e da comunicação. Sem compressão, a maioria dos produtos tecnológicos de consumo e entretenimento nunca teria chegado a existir como, por exemplo, as câmeras fotográficas digitais, DVDs, leitores de MP3, o Youtube e o streaming de vídeo, as redes sem fio e a televisão digital. De fato, as tecnologias de compressão multimídia permitem representar a informação de uma forma mais eficiente, reduzindo os grandes volumes de espaço de armazenamento que ocupa e, portanto, a largura de banda que consome para se transmitir nas redes e na Internet (RIBEIRO; TORRES, 2009).

3.4 Formato Ogg Vorbis

Conhecido por seu método de compressão de áudio digital e por ser livre de patentes, o codec de áudio Ogg Vorbis surgiu a partir da junção de dois projetos desenvolvidos pela Xiph.org. O projeto Ogg é um formato recipiente que armazena qualquer tipo de conteúdo multimédia digital. O contêiner multimédia (assim chamado pela Xiph.org) pode conter informações de textos, áudio e vídeo e tem sido usado para encapsular dados comprimidos de outros codecs (RFC_3533, 2003). O projeto Vorbis, por outro lado, é um *codec* responsável pela compressão e descompressão de áudio e foi projetado para ser contido em um ficheiro Ogg (XIPH.ORG, 2012). O *codec* é o termo utilizado para algoritmos

que realizam codificação e decodificação de um fluxo de dados digital. Este dois projetos podem ser usados separadamente ou em conjunto com outros formatos e *codecs*.

O Ogg Vorbis visa substituir completamente todos os formatos patenteados e proprietários. Ele foi criado para concorrer, principalmente, com os codecs MP3 e o AAC. O formato foi iniciado logo após a organização Fraunhofer Gesellschaft (responsável por projetar o algortimo de compressão do formato MP3 patenteado pela Fraunhofer IIS) anunciar os seus planos de cobrar licenças de uso para o formato MP3.

Apesar de ser um formato não proprietário, o Ogg Vorbis consegue melhores taxas de compressão e qualidade de áudio superior que o MP3. Uma desvantagem do Ogg Vorbis é sua compressão ser quase duas vezes mais lenta, mas isso pode ser resolvido no futuro. O Ogg pode não ser tão popular quanto o MP3, todavia, tem sido cada vez mais conhecido e melhorado. No ramo de jogos, o Ogg tem sido consideravelmente usado.

3.4.1 Formato Ogg

O resultado de um encapsulamento Ogg é chamado de ***Pyshical Ogg Bitstream***. O Ogg encapsula dados comprimidos criado por codecs. Os dados comprimidos são fluxos de bits de mídia que é chamado de ***logical bitstreams*** e pode ser representados por um fluxo simples de áudio Vorbis, múltiplos fluxos de áudio ou fluxos de áudio e vídeo multiplexados. Um *logical bitstream* é estruturado, ou seja, ele é dividido em uma sequência de ***packets***. Os pacotes são criados pelo codificador de *logical bitstream* e estes pacotes apenas tem representação para o codificador: “Please note that the term packet is not used in this document to signify entities for transport over a network” (RFC_3533, 2003).

O Ogg oferece suporte para o transporte do fluxo de bits lógicos como enquadramento e intercalação para diferentes fluxos. O formato também é capaz de detectar corrupção e recuperar-se após algum erro de análise. Marcos de posição é suportado e, dessa forma, torna-se possível o acesso aleatório direto de posições arbitrárias na *bistream*. Outra forte característica é a capacidade de streaming, onde não é necessário a construção completa do *bitstream* para a transmissão do mesmo. Outros suporte, e não menos importante, é a pequena sobrecarga referente a largura de banda do bitstream.

O *Pyshical Ogg Bitstream* consiste de vários *logical bitstreams* intercalados em ***Pages***. O *logical bitstream* são indentificados por um número de série único no cabeçalho de cada *page*. As *pages* são intercaladas simultaneamente e não precisam seguir uma ordem regular, mas precisam ser consecutivos dentro de um *logical bitstream*. A desmultiplexação Ogg é capaz de reconstruir o *logical bitstream* original. Cada *page* contém apenas um tipo de dado, uma vez que pertence a um único *logical bitstream*. As *pages* possuem tamanhos variáveis e tem um cabeçalho contendo encapsulamento e recuperação de erros de informação. Cada *logical bitstream* em um *physical Ogg bitstream* começa com uma

páginal inicial especial **bos** (*beginning of stream*) e termina com uma página especial **eos** (*end of stream*) (RFC_3533, 2003).

Uma página bos contém informações para identificar o tipo de codec e pode conter informações para configurar o processo de decodificação. Ela também deve conter informações sobre os meios de comunicação codificada. Para exemplificar, uma página bos, para um áudio, deve conter a taxa de amostragem e o número de canais. Por convenção, os primeiros bytes de uma página bos contém informações necessárias para identificação do *codec*. O Ogg também permite mas não obriga pacotes de cabeçalhos (*header packets*) secundários após a página bos para o *logical bitstream* e estes devem preceder quaisquer pacotes de dados (*data packtes*) em qualquer *logical bitstream*. Este pacotes de cabeçalhos são enquadrados em um número integral de páginas e estas, por sua vez, não contém quaisquer pacotes de dados. Portanto, conclui-se que um *physcial Ogg bitstream* começa com as páginas de todos os *logical bitstreams* contendo um pacote de cabeçalho inicial por página, seguido por pacotes de cabeçalhos subsidiários de todos os fluxos e, por fim, seguido pos páginas contendo pacotes de dados (RFC_3533, 2003).

A especificação de encapsulação de um ou mais *logical bitstreams* é chamado de ***media mapping*** ou mapeamento de mídia. A (RFC_3533, 2003) exemplifica um mapeamento de mídia citando o "Ogg Vorbis", que usa a estrutura Ogg Vorbis para encapsular os dados de áudio codificados para armazenamento baseado em arquivo e transmissão baseada em fluxo. O Ogg Vorbis fornece o nome do *codec* e algumas informações referentes ao áudio codificado. Ele também possui duas páginas adicionais de cabeçalho. A página bos do Ogg Vorbis começa com o byte 0x01, seguido por "Vorbis" totalizando 7 bytes de identificador. O *codec* Vorbis será visto na seção 2.4.2.

3.4.1.1 Mutiplexagem

O formato possui dois tipos de multiplexagem: *grouping* (agrupamento) e *chaining* (encadeamento). No ***grouping*** a multiplexação é simultânea onde vários *logical bitstreams* são intercalados em um mesmo *physical bitstream* como, por exemplo, a intercalação de um fluxo de vídeo com diversas faixas de áudio. É importante ressaltar que na multiplexagem *gouping* todas as páginas bos de todos os *logical bitstreams* devem aparecer juntas no início do *physical Ogg bitstream*. Por outro lado, as páginas eos não precisam estar lado a lado. A multiplexagem ***chaining***, por sua vez, foi definido para proporcionar um mecanismo simples de concatenação do *physical Ogg Vorbis* onde os *logical bitstreams* completos são concatenados. O *chaining* são frequentemente usados em aplicações de transmissão. Não há sobreposição dos *bitstreams* pois a página eos de um dado *logical bitstream* é imediatamente seguido pela página do próximo bos (RFC_3533, 2003). Os dois tipos de multiplexagem podem ser usados em conjunto. A Figura 6 é um diagrama que representa os dois tipos de multiplexagem.

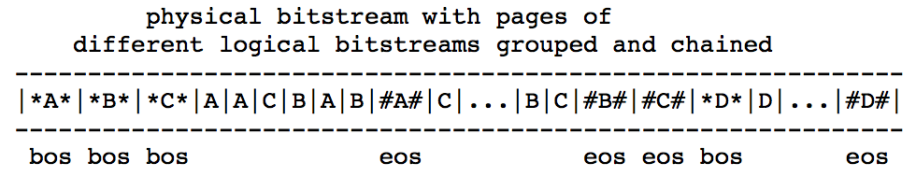

Figura 6 – Diagrama exemplificando as multiplexagens (RFC_3533, 2003)

O Ogg não sabe nada sobre os dados do *codec* que ele encapsula e é, portanto, independente de qualquer codec de mídia. A exceção para aquilo que o Ogg conhece é que cada *logical bitstream* pertence a um *codec* diferente, os dados do *codec* vem em ordem e possuem marcadores de posição (são *granule positions* que serão abordados mais a frente). O recipiente Ogg não possui conceito de tempo. No entanto, o Ogg sabe sobre o aumento sequencial dos marcadores de posição. Isto faz do Ogg uma estrutura genérica mas que realiza o encapsulamento de fluxos de bits de tempo contínuo.

O processo de multiplexação ocorre no nível de página. Porém, os *codecs* fornecem fluxos de bits que são entregues ao Ogg em pacotes de tamanho arbitrário e não limitado. As páginas, em contra partida, possuem tamanho máximo de 64 Kbytes. Desta forma, os pacotes, na maiora das vezes, precisam ser distribuídos ao longo de várias páginas. Para realizar essa distribuição e facilitar o processo, o Ogg quebra um pacote em **segmentos** de 255 bytes sendo o último segmento mais curto. Os segmentos são apenas uma construção lógica e não tem um cabeçalho para si. Uma *flag* no cabeçalho da página informa se uma página contém informações de um pacote de continuação da página anterior (RFC_3533, 2003). Para um melhor entendimento, a Figura 7 é um diagrama que exemplifica a relação entre páginas, pacotes e seguimentos.

Um cabeçalho de página contém todas as informações necessárias para a desmultiplexação dos fluxos de bits lógicos e para executar a recuperação de erros básicos e marcos para a procura. Cada página é uma entidade auto-suficiente de modo que o mecanismo de página de decodificação pode reconhecer, verificar e lidar com páginas simples em um momento sem ser necessário todo o fluxo de bits. A Figura 8 define o formato para um cabeçalho de página.

Segundo (RFC_3533, 2003), os campos da estrutura da Figura 8 possuem os seguintes significados:

- ***capture_pattern***: cada página começa com uma *string* de quatro bytes "OggS". As letras "O" e "S" devem ser maiúsculas e suas representações na tabela ASCII dadas em hexadecimal são 0x4f e 0x53, respectivamente. A letra minúscula "g" é representada por 0x67. Se a sincronização for perdida, este campo ajuda o decodificador na recuperação da sincronização;

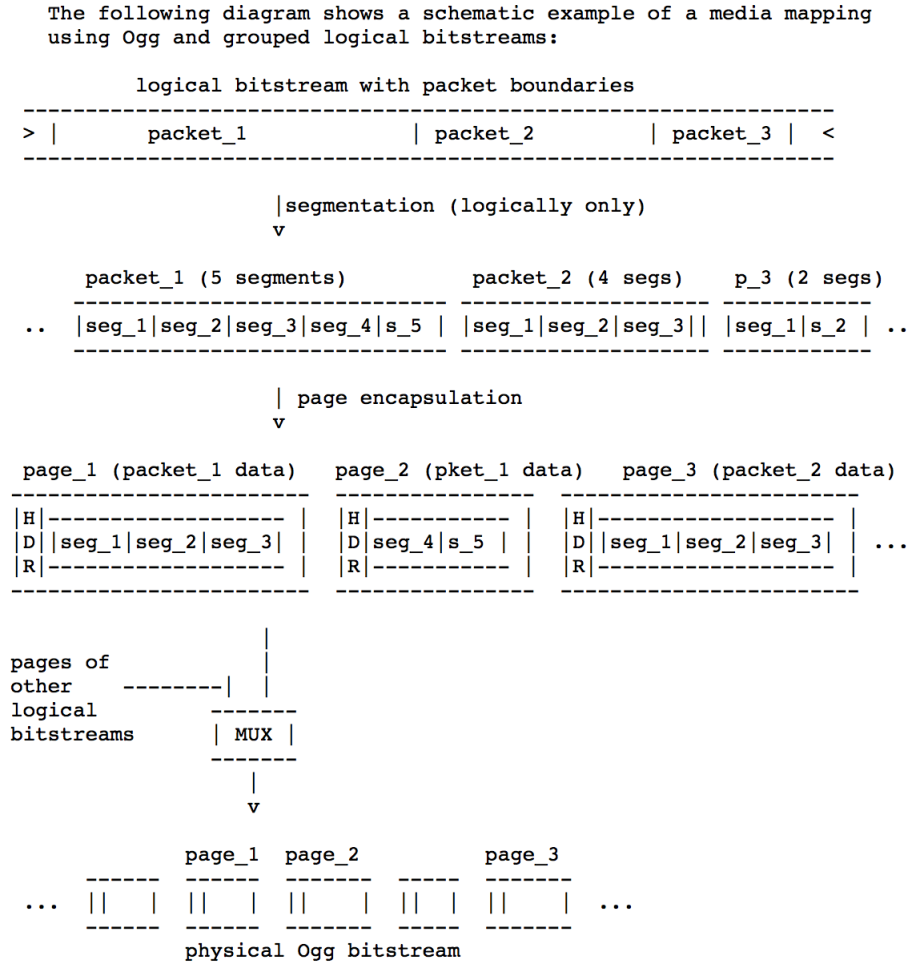

Figura 7 – Diagrama exemplifica a forma como são postos os segmentos (RFC_3533, 2003)

- ***version***: especifica a versão do arquivo Ogg e possui 1 byte para sua representação;
- ***header_type***: Este campo possui apenas um byte. No entanto, os bits identificam o tipo específico de cada página:

  bit 0x01: se este bit for igual a 1, significa dizer que a página contém dados de um pacote de continuação da página anterior. Caso seja 0, a página está iniciando um novo pacote;

  bit 0x02: se o bit estiver setado como um, então é uma página bos. Se o bit for 0, significa que não é uma primeira página;

  bit 0x04: se o bit estiver setado como um, então é uma página eos. Se o bit for 0, significa que não é a última página.

- ***granule_position***: Os 8 bytes (ou 64 bits) que representa este campo, contém informações referente a posição. O Ogg não contém informações de tempo, porém o tempo é mapeado referente a posição de um dado digital em uma *stream*. O significado real deste campo em relação ao tempo depende de cada *codec*;

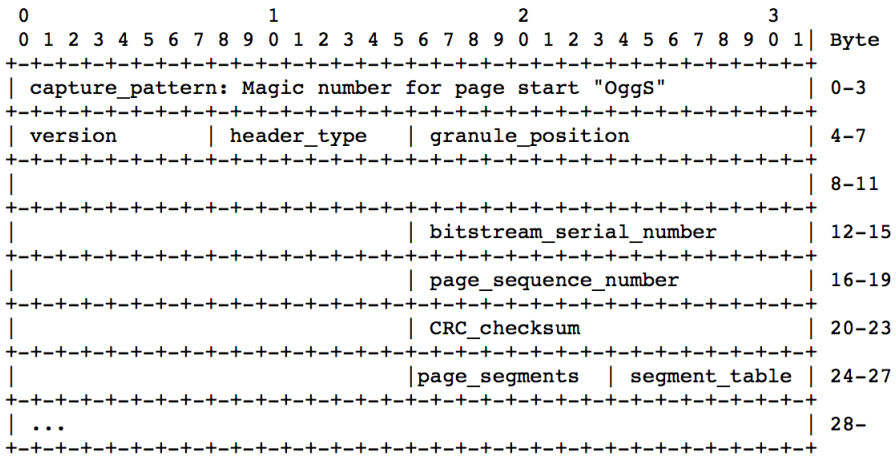

Figura 8 – Formato de cabeçalho de uma página (RFC_3533, 2003)

- ***serial_number***: o *logical bitstream* é identificado com um único número de série neste campo de 4 bytes;
- ***sequence_number***: se o campo anterior identifica um *logical bitstream* este campo contém o número de sequência da página que é incremetnado separadamente. O campo também possui 4 bytes e através dele o decodificador consegue identificar a perda de página;
- ***CRC_checksum***: contém CRC checksum de 32 bits cujo polinômio gerador é 0x04c11db7;
- ***page_segment***: este campo possui um 1 byte de tamanho e fornece o número de entradas de segmentos codificados na tabela de segmentos;
- ***segment_table***: contém informações de *lacing values* de todos os seguimentos contido na página. Cada byte deste campo contém um valor de laço e, portanto, o número de bytes deste campo é igual a quantidade de segmentos existentes na página.

3.4.2 *Codec* Vorbis

O *codec* Vorbis codifica blocos PCM de curta duração em pacotes de bits de dados comprimidos. Ele não fornece nenhum tipo próprio de enquadramento, sincronização ou proteção contra erros. Para isso, Vorbis utiliza do formato de fluxos de dados do Ogg para disponibilizar sincronização, recaptura de sincronização após erros, marcadores durante a procura e informação suficiente para separar corretamente os pacotes de dados codificados para obter o pacote original. Segundo sua especificação (XIPH.ORG, 2012), Vorbis é apenas um método de aceitação de entrada de áudio, dividindo este áudio em *frames* (quadros) individuais comprimidos que são pacotes não formatados. Basicamente,

o decodificador aceita estes pacotes em sequência, decodifica-os e os sintetiza no fluxo do áudio original.

Para obter arquivos cada vez menores e com uma qualidade constante, o Vorbis utiliza uma codificação de débito binário variável. O **VBR** (*Variable Bit Rate*) é um algoritmo que define qual a melhor taxa de bit para cada frame da música variando, assim, a quantidade de transferência de bits por segundo mas mantendo a qualidade constante (BECKER..., 2014). Ou seja, O Vorbis é um formato de taxa variável e seus pacotes não possuem tamanho fixo e depende do parâmetro de qualidade. O nível de qualidade varia de 0 a 10 com mudanças feitas de 1 em 1. A qualidade 0 corresponde a 64 Kbps, 5 por volta de 160 Kbps e 10 cerca de 400 Kbps. O nível cinco de qualidade já é suficiente para se aproximar de uma qualidade de áudio de CD. Portanto, ao usar o VBR, o Vorbis consegue atingir qualidade de som semelhante ao original e com melhor compressão. A taxa de bits varia de 16 Kbps a 500 Kbps por canal. O número de canais de áudio independentes suportados podem chegar a 255 (XIPH.ORG, 2012). No entanto, o nível 3 é mais utilizado por conseguir, a partir de 110 Kbps, gerar ficheiros de áudio menores e qualidade superior a ficheiros MP3 com 128 Kbps.

3.4.2.1 Algortimo de compressão

Os métodos usados para a compressão de áudio geralemente pertencem a duas categorias: métodos que trabalham no domínio do tempo e métodos que trabalham no domínio da frequência. Todos os formatos conhecidos que trabalham no domínio da frequência usa a transformada discreta do cosseno modificada. O **MDCT** (*Modified Discrete Cosine Transform*) realiza a tranformação para o domínio da frequência a partir do domínio do tempo. O ***window shape*** que contém amostras PCM com duas variações de comprimentos especificados em 2048 ou 512 amostras. O Vorbis usa o MDCT para aplicar transformação com janelas sobrepostas (XIPH.ORG, 2012). A função MDCT é dada por:

$$X(k) = \sum_{n=0}^{N-1} x(n) \cos\left[\left(n + \frac{M+1}{2}\right) + \left(k + \frac{1}{2}\right)\frac{\pi}{M}\right] \tag{3.1}$$

, onde N = 2M é o tamanho da janela, M é o número do coeficiente de transformação e o k variando de 0, 1, ..., M - 1 (NIKOLAJEVIC; FETTEWIS, 2003). A inversa do MDCT (a saber, IMDCT) é dada por

$$x(k) = \sum_{n=0}^{M-1} X(n) \cos\left[\left(n + \frac{M+1}{2}\right) + \left(k + \frac{1}{2}\right)\frac{\pi}{M}\right] \tag{3.2}$$

, em que o k varia de 0, 1, ..., N - 1 (NIKOLAJEVIC; FETTEWIS, 2003). Como dito anteriormente, cada janela pode ter um de dois tamanhos possíveis indicados no

cabeçalho da *stream*. Os tamanhos admissíveis (em amostras) são todas dadas em potência de 2 entre 64 e 8192. Todas as janelas de Vorbis utilizam a função de inclinação:

$$y = \sin\left(0.5 * \pi \sin^2\left(\frac{x + 0.5}{n} * \pi\right)\right) \tag{3.3}$$

, onde n indica o tamanho das amostras (XIPH.ORG, 2012). Após a transformação de domínio de frequência do sinal, o próximo passo é realizar uma análise pelo modelo psicoacústico onde a parte inaudível de um *spectrum* pela audição humanda é removido. O modelo **psicoacústico** baseia-se no limite absoluto da audiência humana e, assim, indica quais frequências podem ser "sacrificadas" durante a compressão sem uma grande perda audível de qualidade. Em seguida, o vetor ***floor*** é gerado para cada um dos canais e é usado para obter uma aproximação de baixa resolução do *spectrum* de áudio para o canal fornecido no *frame* atual. O *floor* obtido será então subtraído do *spectrum* gerando os ***residues***. Os vetores de resíduos de ambos os canais são transformados da representação cartesiana para a representação polar e esse processo recebe o nome de ***channel coupling*** e, a partir disso, é feita uma compressão dos resíduos com **VQ** (Vector Quantization). Posteriormente, os resultados são codificados por meio do algorimto de **Huffman** para eliminar ainda mais a redundância. A saída final do processo gera o pacote Vorbis. Estes pacotes, por fim, são encapsulados em um recipiente Ogg universal (SVíTEK, 2006) e (XIPH.ORG, 2012). A firgura 9 é um diagrama que representa todo o processo descrito nesta seção.

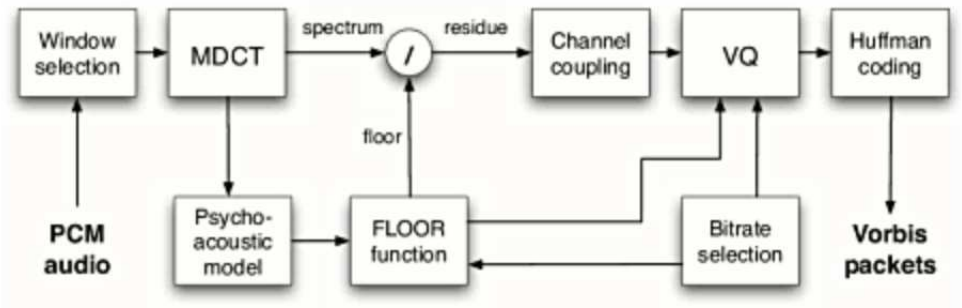

Figura 9 – Diagrama do algoritmo de compressão da Vorbis I (SVíTEK, 2006).

3.4.2.2 Estruturação

O Vorbis é estruturado em pacotes e utiliza quatro tipos diferentes. O três primeiros tipos marcam três tipos diferentes de cabeçalho e devem estar dispostos na seguinte ordem: cabeçalho de identificação, cabeçalho de comentário e cabeçalho de configuração. Após os três pacotes de cabeçalho, todos os pacotes subsequentes são pacotes de áudio. E estrutura, de uma forma geral, é ilustrada na Figura 10.

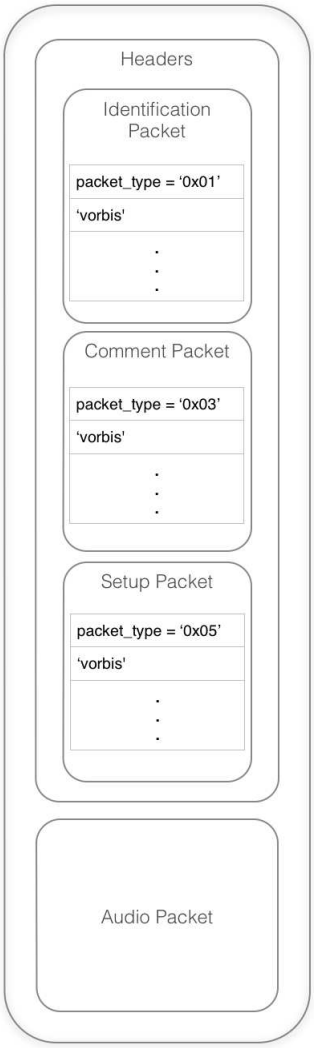

Figura 10 – Diagrama Geral representando uma estrutura Vorbis.

Todos os pacotes cabeçalhos começam com um byte para identificação do tipo de cabeçalho e mais seis bytes para identificação da *string* "vorbis". O campos seguintes são específicos de cada pacote. O ***identification packet*** identifica o fluxo de bits como Vorbis e contém informações sobre a versão do Vorbis usado, o número de canais de áudio e a taxa de amostragem. O segundo pacote é o ***comment header*** e contém metadados que são comentários de usuários referentes ao pacote de áudio tais como nome da música, nome dos artistas ou o nome da gravadora. Este campo não é necessário para a decodificação do áudio, mas é um pacote obrigatório e deve ser identificado e ignorado adequamente para não corromper o arquivo. O Vorbis sugere alguns nomes de campos mas eles não são obrigatórios e outros também podem ser usados. A lista contém as seguintes sugestões: TITLE, VERSION, ALBUM, TRACKNUMBER, ARTIST, PERFORMER, COPYRIGHT, LICENSE, ORGANIZATION, DESCRIPTION, GENRE, DATE, LOCATION, CONTACT e ISRC. A maioria das informações necessárias para inicializar o decodificador estão contigas no ***setup header***. Ele contém informações de configuração

do *codec* e *codebooks* contendo o VQ completo e o código de Huffman. Por último, temos o ***audio packet*** que contém os dados de áudio do arquivo (XIPH.ORG, 2012).

No processo de decodificação, a primeira etapa da decodificação de pacotes de áudio é para ler e verificar o tipo de pacote. Um pacote não-audio quando um pacote de áudio é esperado indica corrupção de fluxo. O decodificador deve ignorar o pacote e não tentar decodificá-lo para áudio. Os pacotes de áudio começam com um único bit que precisa sempre ser 0 (XIPH.ORG, 2012).

3.4.3 Licença de uso

É intenção dos desenvolvedores Ogg Vorbis que o formato seja utilizável sem preocupações de propriedade intelectual e, por isso, são de domínio público. “Ogg Vorbis is a fully open, non-proprietary, patent-and-royalty-free (...)” (XIPH.ORG, 2000). O Ogg Vorbis está sob licença BSD e os termos de licença estão descritos e acessíveis em (XIPH.ORG, 1994)

4 Delimitação do Assunto

A cultura de audiolivros, como explanado anteriormente, tem ganhado força no Brasil. Os formatos de áudio, até hoje, não oferecem suporte para marcação de conteúdo. Assim como um livro possui um sumário para facilitar e nortear o acesso a informação contido no livro, em um áudiolivro isso se torna ainda mais necessário. (REIS, 2013) inicou um trabalho visando uma melhor suporte para o uso de *audiobooks*. A solução proposta por (REIS, 2013) soluciona o problema e os resultados foram satisfatórios. O trabalho proposto fez uso do formato WAVE onde dois novos blocos de dados foram inseridos originando o formato RAB. Estes blocos contém informações a respeito do áudio armazenado e marcações de conteúdo.

No entanto, percebeu-se que o tamanho do arquivo RAB crescia de forma linear em função do tempo. Um arquivo RAB, em média, chega a possuir mais de 200 megabytes de tamanho. Isso ocorre porque o formato WAVE (sobre o qual o RAB foi derivado) não usa compressão de dados. Considerando toda a fundamentação teórica apresentada neste trabalho principalmente no que tange o rumo em que o avanço tecnológico tem tomado, a necessidade das pessoas em cada vez mais compartilhar dados em redes de transmissão e o desempenho das aplicações em processar estes dados, futuramente, o formato RAB poderá não ser útil por conta do tamanho dos seus ficheiros de áudio.

A solução para este problema seria ou fazer a compressão do formato RAB ou propor o uso de um novo formato, open source e que já faz o uso de compressão de dados. Partindo deste pressuposto e de toda a pesquisa realizada, o formato não proprietário Ogg Vorbis fornece um melhor suporte para a especificação de um formato de áudio não ocasionando o mesmo problema do formato RAB.

5 Metodologia

5.1 Levantamento Bibliográfico

A pesquisa bibliográfica foi feita, basicamente, por assunto, por autor e por título. A pesquisa realizada por assunto foi a mais utilizada e termos adequados foram usados para se obter um pesquisa mais efetiva para o entendimento e desenvolvimento do trabalho tais como *audiobooks*, ogg vorbis, livorbisfile, libvorbis, libvorbisenc, compressão de áudio, MDCT, formatos de áudio, WAV, AIFF, MP3, PCM, entre outros. Também foram feitas pesquisas por assunto a respeito de ferramentas necessárias para o desenvolvimento do trabalho tais como formulas latext, player ogg, sox play, dump ogg, convert png to eps, bibtex models example, entre outros. Para a pesquisa feita por autor e título é necessário que já se saiba qual autor ou obra são relevantes para o tema escolhido como, por exemplo, a pesquisa pelo autor Tanenbaum. Levantamento por assunto foi bastante utilizado em pesquisa na internet usando catálogos e mecanismos de busca em sites como o Google Acadêmico (GOOGLE, 2014), Google com pesquisa *web* e com filtro para livros, *ACM Digital Library* (ACM, 2014) e Scielo (SCIELO, 2014). Ideias e dicas dadas pelo orientador prof. Dr. Edson Júnior deste trabalho foram de suma importância principalmente para uma determinação de "um ponto de partida".

Através do levantamento bibliográfica foi possível listar e consolidar citações de trabalhos fundamentais para o tema ou algo similar ao que foi proposto neste trabalho.

5.2 Ferramentas utilizadas

As ferramentas utilizadas para o desenvolvimento do trabalho são descritas, em poucas palavras, para qual propósito cada uma foi usada e qual a versão utilizada.

Por fornecer ferramentas nativas que ofereceram grande suporte para o desenvolvimento deste trabalho, pelo formato Ogg Vorbis fornecer API de fácil instalação e uso em distribuições Unix e pelo conhecimento prévio do sistema operacional o Ubuntu 14.04 LTS foi adotado para o ambiente de desenvolvimento. O compilador utilizado foi o gcc versão 4.8.2.

O editor de texto utilizado para o desenvolvimento dos códigos fonte em linguagem de programação C foi o Sublime Text 2. Ele foi escolhido por possuir uma interface limpa, por fornecer uma prévia visualização de todo o documento e, principalmente, pelos comandos que ele oferece para facilitar a codificação. Podemos citar como exemplo a troca de linhas sendo possível mover uma linha ou um bloco inteiro de código, cursores múltiplos

possibilitando a escrita em diversas parte do código simultaneamente, busca por palavras de mesmo nome e a buscar e substituição de palavras específicas com uma ou múltiplas seleções.

O Latex foi utilizado para desenvolver o trabalho escrito e escolhido por gerar, como saída, um pdf com alta qualidade tipográfica e totalmente formatado. A versão utilizada foi pdfTeX versão 3.1415926-2.5-1.40.14.

Inicialmente foi utilizada a ferramenta hexdump nativa do sistema operacional Unix para mostras os dados binários em hexadecimal com o intenção de facilitar o entedimento do formato de áudio e validar os dados inseridos no formato. Entretanto, a feramenta hexdump foi substituída pela oggz versão 1.1.1.

A versão v.14.4.1 da ferramenta sox foi utilizada para executar os arquivos de áudio *.ogg com o intuito de verificar se o arquivo não foi corrompido devido as constantes modificações do formato para inserção do pacote de marcação de conteúdo e da inserção dos metadados no cabeçalho de comentários.

5.3 Proposta anterior

Como este trabalho é uma evolução do trabalho realizado por (REIS, 2013), a primeira coisa a ser feita foi entender a sua proposta de trabalho e, para isso, foi feito um estudo em cima de sua monografia. Esta proposta da evolução considerou os problemas levantados e traçou o objetivo de reproduzir o trabalho não originando o mesmo problema e então melhorar o trabalho anterior.

5.4 Entendendo o Ogg Vorbis

O segundo passo foi o estudo de um novo formato: Ogg Vorbis desenvolvido pela fundação Xiph.org. Foi feito um estudo minucioso em cima do documento de especificação do formato Ogg Vorbis para conhecer a sua estrutura e o tipo de suporte que ele oferece. A ferramenta *hexdump* utilizada por (REIS, 2013) foi usada como suporte para um melhor entendimento do formato *.ogg. A Figura 11 mostra o resultado da execução do hexdump em um arquivo *.ogg cujo comando é: $ hexdump -C <file_name>.

No entanto, os dados ainda ficaram muito confusos e de difícil interpretação. Após uma pesquisa verificou-se a existência de uma outra ferramenta também nativa no sistema operacional Ubuntu e voltada para os arquivos com extensão *.ogg e possui a mesma finalidade da ferramenta hexdump. A diferença entre elas está na forma com que os dados são apresentados. Podemos verificar na Figura 12 como o oggz-dump estrutura os dados. O comando para execução da ferramenta é dado no terminal e possui o seguinte formato: $ oggz-dump <file_name>.

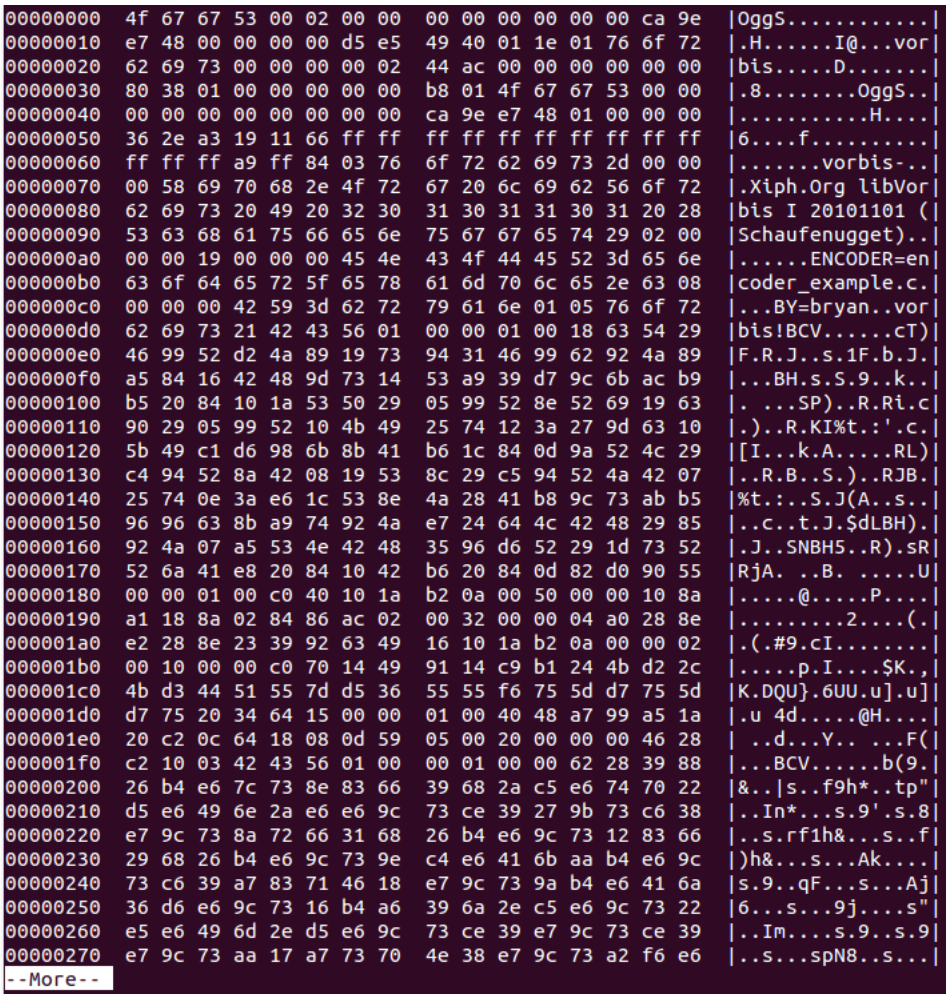

Figura 11 – Execução da ferramenta hexdump.

É notável a diferença e a facilidade com que a ferramenta oggz informa sobre os pacotes contidos em um arquivo *.ogg. Após o comando é possível verificar os pacotes separadamente bem como sua informações tais como número do pacote, informações de grânulo, o “tempo” em que aquele pacote é lido, o seu tamanho, entre outras informações. Além das informações do pacote também é possível visualizar seu conteúdo e logo percebemos que os pacotes de número 0, 1 e 2 são os pacotes cabeçalhos. A identificação dos pacotes cabeçalhos pode ser percebida pois após o primeiro byte do pacote, os 6 bytes subsequentes contém a string “vorbis” onde cada byte representa uma letra.

5.5 Construção do codificador

O estudo e o processo acima foi realizado para entender a estrutura do Ogg Vorbis e onde inserir os metadados e as marcações de conteúdo validando, assim, a possibilidade do uso do formato para a solução do problema. Para dar continuidade no estudo de viabilidade do formato Ogg Vorbis, foi desenvolvido um codificador em linguagem C. Como suporte, foram utilizadas as bibliotecas *libogg*, *libvorbis* e *libvorbisenc*. O *libvorbisenc*

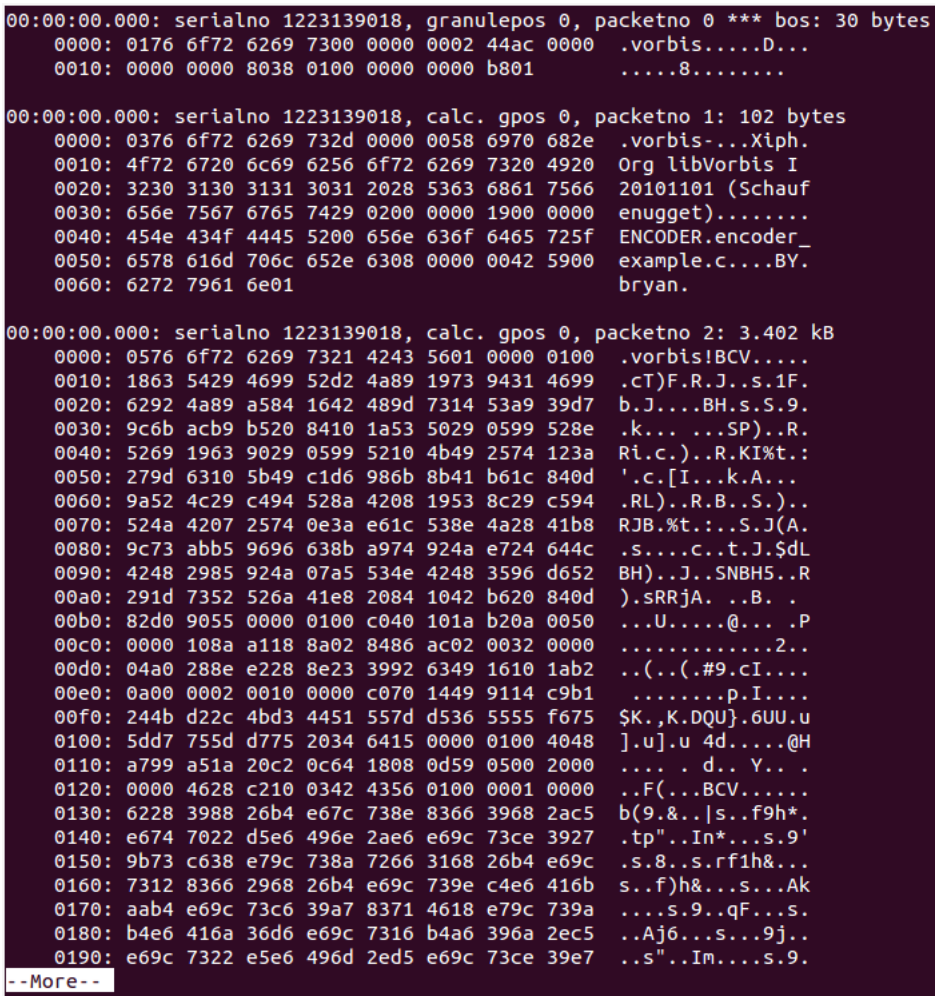

Figura 12 – Execução da ferramenta oggz-dump.

é responsável pela codificação. Para compilar o arquivo é necessário utilizar o seguinte comando:

$ gcc -o <nome_para_o_executável> <código_fonte> -logg -lvorbis -lvorbisenc

Para garantir que o processo de codificação realmente funcionasse, foi utilizado outro formato no processão de codificação de um formato Ogg Vorbis. O código desenvolvido pega o conteúdo sonoro de um formato WAVE e o codifica em Ogg Vorbis, com os seus dados comprimidos. Em outras palavras, o PCM do formato WAVE é decodificado e posto em memória e, em seguida, os pacotes cabeçalhos do Ogg Vorbis são construídos. O PCM passa a ser inserido dentro do pacote de áudio finalizando o processo de decodificação. A ferramenta *oggz-dump* foi executada no arquivo gerado e após análise, os pacotes cabeçalhos, em tese, foram codificados corretamente. Para verificar se a integridade do arquivo não foi corrompida, utilizou-se o player Sound Exchange licenciado sob a GNU General Public License e distribuído por Chris Bagwell através (SOX, 2014). Este player possui uma interface de linha de comando e, ao utilizá-lo, era possível executar o som, este agora no formato Ogg, sem interrupção.

5.5.1 Inserção dos metadados

Como fundamentado teoricamente no item mais acima, o arquivo *.ogg possui um pacote onde é possível inserir comentários. O *comment packet* é o segundo pacote de cabeçalho da sequência de três que o Ogg Vorbis utiliza. Na Figura 12 ele aparece como o pacote número 1. O próximo passo então foi inserir, comentários referentes ao arquivo de áudio. Logo, o pacote de comentários do formato Ogg Vorbis foi utilizado para inserção dos metadados e, comparando-o ao trabalho realizado (REIS, 2013), corresponde a estrutura META. Para este fim, o código desenvolvido em linguagem de programação C para a codificação foi modificado e este agora, além de pegar o conteúdo sonoro de um formato WAVE e o codificar em Ogg Vorbis com os dados comprimidos, ele também insere metadados no pacote. Para verificação da integridade do arquivo neste ponto do desenvolvimento, os mesmos passos utilizados no processo de codificação foram seguidos e as ferramentas *oggz-dump* e *sox* foram utilizadas.

5.5.2 Construção do pacote LGMK

Referente a estrutura LGMK (REIS, 2013), o formato Ogg Vorbis não possui suporte e se fez necessário a alteração de sua estrutura. Isso deveria ser feito, obviamente, sem que o arquivo fosse corrompido possibilitando sua execução em players comuns. Para tal finalidade, um novo pacote foi definido. A Figura 16 mostra quais são os campos que compõe o pacote.

Figura 13 – Formato do pacote LGMK.

O campo referente ao **user_lgmks** é um array que armazena todas as marcações fornecidas pelo usuário e seu tamanho é ilimitado. O **lgmks_lengths** é um array responsável por armazenar o tamanho de cada marcação. A quantidade de marcações contidas no campo user_lgmks é armazenada no **lgmks**. Por último, o campo **user** contém informações sobre o usuário que fez as marcações. Uma nova estrutura foi definida e inserida dentro do formato sem, obviamente, corrompê-lo. Foi possível codificar e também decodificar as marcações de conteúdo com integridade.

Os três *header packets* definidos pelo Vorbis devem seguir a ordem disposta na Figura 14 ou o arquivo será corrompido. O pacote LGMK não é reconhecido pelo Vorbis pelos seguintes motivos:

1. O pacote possui um tipo diferente dos três tipos de cabeçalhos do Vorbis, desta forma ocorre erro no processo de decodificação.
2. A string de identificação do pacote é "lgmks" e o Vorbis não irá decodificar este pacote;
3. Quando o Vorbis for decodificar o pacote de áudio, o pacote LGMK SERÁ ignorado por ser um pacote não-áudio.

No entanto, a decodificação não é corrompida por conta da forma que foi estruturado o pacote e do local onde foi inserido. O pacote LGMK foi construído como sendo um pacote de cabeçalho semelhante aos *header packets* do *codec* Vorbis. Ou seja, ele possui um byte para a identificação do tipo de pacote de cabeçalho. A diferença está nos seis bytes subsequentes, onde a string que representa a identificação do pacote é "lgmks". O pacote foi inserido direto no formato recipiente Ogg logo após o terceiro pacote de cabeçalho do Voribs. A Figura 14 mostra como ficou organizado a estruturação do formato Ogg vorbis após inserção do pacote referente a marcação de conteúdo.

Como foi posto após o cabeçalho de configuração do Ogg Vorbis, no processo de decodificação padrão, o pacote LGMK tentará ser codificado como pacote de áudio e então será ignorado. Portanto, ao executar um player o arquivo Ogg Vorbis gerado é tocado normalmente.

5.6 Construção do decodificador

Para o desenvolvimento do código referente ao processo de decodificação do arquivo *.ogg foram utilizadas as bibliotecas *libvorbisfile*, *libvorbis* e *libogg*. A *libvorbisfile* oferece o suporte necessário voltado para a decodificação do arquivo tornando o processo mais simples. Para compilar o arquivo é necessário utilizar o seguinte comando:

$ gcc -o <nome_para_o_executável> <código_fonte> -logg -lvorbis -lvorbisfile

O código implementado decodifica os dados de cabeçalho e os imprime no terminal. Os dados PCM contidos do pacote de áudio são direcionados para um arquivo de saída.

5.6.1 Decodificação dos metadados

O metadados, uma vez codificados, precisavam ser decodificados e seu conteúdo recuperado em mémoria sem perda de dados. Para este fim, foi feito o uso da API *libvor-*

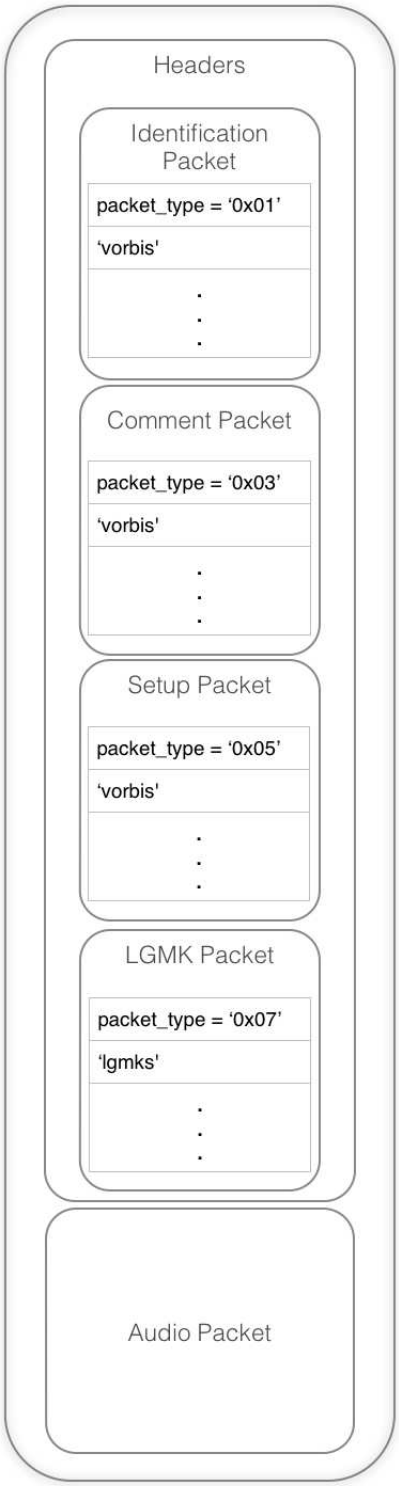

Figura 14 – Estrutura Ogg Vorbis com o pacote LGMK inserido.

bisfile onde foi possível recuperar os metadados corretamente.

5.6.2 Decodificação do pacote LGMK

Como dito, o pacote é inserido diretamente no contêiner Ogg, ou seja, em nenhum momento é feito uso da estrutura vorbis para a inserção do pacote. Para o processo de decodificação do pacote LGMK, e por não fazer parte da estrutura Vorbis, a biblioteca *libvorbisfile* não oferece nenhum suporte e foi desconsiderada no processo de decodificação a partir deste ponto do projeto. Para alcançar o objetivo de decodificação, o uso das bibliotecas *libvorbis* e *logg* foram ainda mais efetivos. A *libvorbis* ainda foi utilizada para decodificar a estrutura Vorbis, porém de uma forma mais "manual". Para o pacote LGMK foi utilizada apenas a biblioteca *logg*. Ao final, todo o arquivo foi decodificado corretamente.

6 Resultados Alcançados

O Editor de *audiobooks* gera com sucesso um formato Ogg Vorbis com todos os metadados inseridos no pacote Vorbis de comentários e as marcações de conteúdo contidas no pacote LGMK construído. Mesmo com sua estrutura alterada, o formato *.ogg pode ser executado em players com suporte a arquivos como o *sox* ou Rhythmbox.

A efeito de comparação ao (REIS, 2013), também foi utilizado o Hino Nacional Brasileiro para simular um *audiobook*. O arquivo orignal no formato WAV possui 3 minutos e 41 segundos de duração e ocupa 42,5 MB de espaço em memória. Este arquivo foi carregado pelo Editor e em seguida foram inseridos os mesmos metadados inseridos por (REIS, 2013) conforme mostra Tabela 1.

Tabela 1 – Metadados inseridos no *comment header*.

Nome	**Valor**
Título	Hino Nacional Brasileiro
Autor	Joaquim Osório & Francisco Manuel
Idioma	Português
Editora	
Local de Criação	Brasil
Número de Páginas	0
Ano de Criação	1822

Para verificarmos os metadados inseridos no pacote de comentários do formato Ogg, a Figura 15

O pacote de comentários é o pacote número 1 e pode-se perceber que os metadados foram inseridos corretamente. Relacinado as marcações de conteúdo, a Tabela 2 mostram as informações que serão inseridas no pacote LGMK. As marcações também foram as mesmas utilizadas por (REIS, 2013).

O verificação da inserção dos marcadores de conteúdo no pacote LGMK pode ser visualizado usando a ferramenta oggz-dump. A Figura 16 mostra o pacote LGMK que pode ser identificado pelo número de pacote 3 e os marcadores de conteúdo contidos dentro dele.

O pacote Ogg Vorbis gerado possui apenas 1,84MB e não mais 42,5MB como o arquivo original. Houve uma redução considerável devido ao algoritmo de compressão do *codec* Vorbis. O arquivo gerado é executado corretamente do início ao fim sem interrupções. Também foi feito um teste para verificar se o pacote LGMK não iria corromper o arquivo Ogg Voribs inserido 22 milhões de marcações. O arquivo final ficou com 340MB de tamanho, no entanto, o player conseguiu executá-lo normalmente.

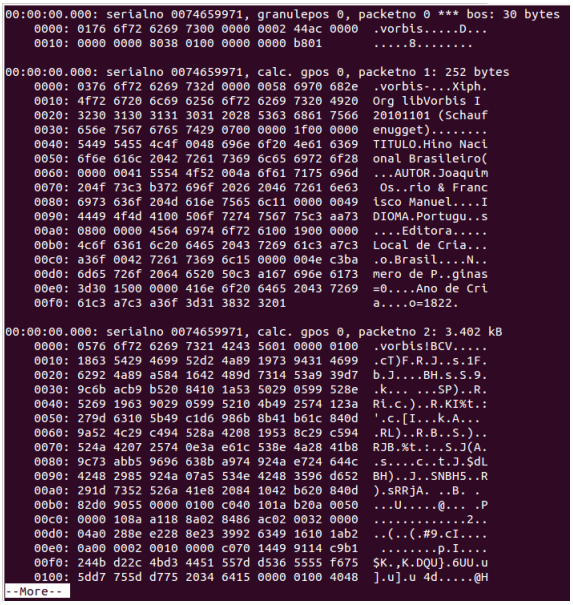

Figura 15 – Utilização do oggz-dump no formato Ogg Vorbis - pacote de comentários.

Tabela 2 – Marcações inseridas no *LGMK header*.

Nome da Marcação	**Tempo(s)**
Parte 1	0
Parte 2	103
Introdução 1	0
Primeira Estrofe	28
Segunda Estrofe	44
Terceira Estrofe	60
Quarta Estrofe	64
Quinta Estrofe	79
Sexta Estrofe	91
Introdução 2	103
Sétima Estrofe	132
Oitava Estrofe	148
Nona Estrofe	164
Décima Estrofe	168
Décima Primeira Estrofe	183
Décima Segunda Estrofe	195
Décima Terceira Estrofe	201
Final	206

Ao utilizarmos a ferramenta oggz-dump podemos visualizar o pacote LGMK inserido

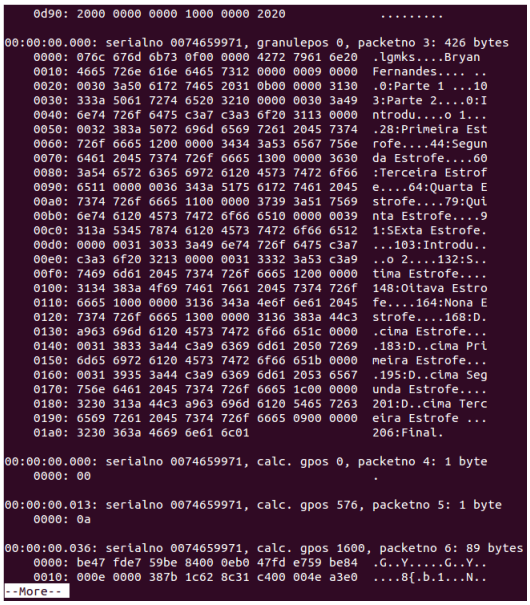

Figura 16 – Utilização do oggz-dump no formato Ogg Vorbis - pacote LGMK.

7 Cronograma de Desenvolvimento

Para evidenciar o cronograma aproximado, segue o gráfico de *gantt* representado pela Figura 17 mostrando o desenvolvimento do projeto que teve sua primeira semana iniciada no dia 16 de setembro de 2014. As entregas foram planejadas semanalmente. O gráfico mostra um aproximação pois algumas adaptações foram feitas no período de prototipagem.

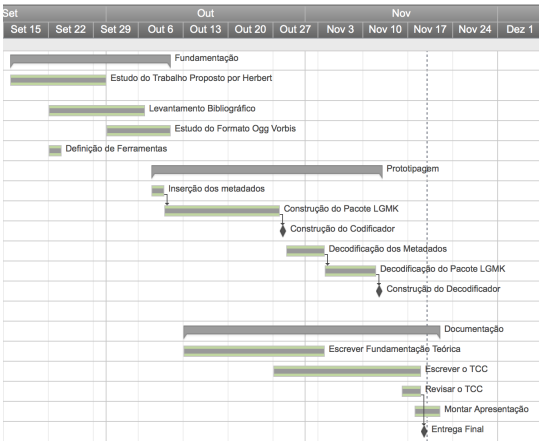

Figura 17 – Gráfico de *gantt* referente ao cronograma.

7.1 Trabalhos Futuros

Neste trabalho houve a tentiva de criar um tocador para ser executado mesmo no terminal, mas não foi possível concluí-lo a tempo. Para continuação do trabalho, o próximo passo é implementar um player para o formato especificado em plataforma iOS. O aplicativo deverá ser capaz de executar o áudio, marcar conteúdo e salvá-lo no formato especificado e inserir metadados no mesmo. A plataforma iOS possui duas linguagens capazes de desenvolver os aplicativos: Objecitve-C e Swift. Será feita uma análise para verificar qual delas melhor encaixa no contexto do trabalho.

Referências

ACM. *ACM Digital Library*: Site. 2014. Disponível em: <http://dl.acm.org>. Acesso em: 12 nov. 2014. Citado na página 45.

ALECRIM, E. Mp3. Info Wester, 2006. Disponível em: <http://www.infowester.com/histomptres.php>. Acesso em: 05 nov. 2014. Citado na página 29.

ASSIS, A. F. *Proposta de Middleware para Compressão Adaptativa de Dados em Ambientes Android*. Monografia (Especialização) — Universidade Federal de Outro Preto, 2010. Citado na página 30.

AUDIBLE. *Home page*: Site. 2014. Disponível em: <http://www.audible.com>. Acesso em: 27 out. 2014. Citado na página 23.

BECKER, F.: Entendendo o mp3 (cbr, abr, vbr). 2014. Disponível em: <http://www.midideejay.com/2009/03/tutorial-mp3-cbr-abr-vbr.html>. Acesso em: 15 nov. 2014. Citado na página 38.

BRAGA, E. G. C. *Compressão de Áudio*. 76 p. Monografia (Projeto Final de Graduação) — Departamento de Engenharia Elétrica, Universidade de Brasília, 2003. Citado na página 31.

DISCO, C.; MEULEN, B. v. d. *Getting New Technologies Together: Studies in Making Sociotechnical Order*. Berlin, Boston: De Gruyter, 2012. Citado na página 32.

DW. Audiolivro vira moda na alemanha. DW, Deutsche Welle, 2004. Disponível em: <http://dw.de/p/4wvJ>. Acesso em: 27 out. 2014. Citado na página 23.

ERROR. *MPEG Audio Compression Basics*: Site. 1999. Disponível em: <http://www.mpgedit.org/mpgedit/mpeg_format/mpeghdr.htm>. Acesso em: 03 nov. 2014. Citado na página 28.

FAGUNDES, E. M. O desafio do consumo de energia dos data centers no brasil. 2014. Disponível em: <http://efagundes.com/itgov/o-desafio-do-consumo-de-energia-dos-data-centers-no-brasil>. Acesso em: 29 set. 2014. Citado 2 vezes nas páginas 32 e 33.

FALADA, U. *Home page*: Site. 2004. Disponível em: <http://www.universidadefalada.com.br>. Acesso em: 02 nov. 2014. Citado na página 24.

FARIAS, S. C. O audiolivro e sua contribuição no processo de disseminação de informações e na inclusão social. Revista Digital de Biblioteconomia e Ciência da Informação, 2012. Disponível em: <http://www.sbu.unicamp.br/seer/ojs/index.php/rbci/article/view/529>. Citado na página 24.

FOROUZAN, B. A. *Comunicação de Dados e Redes de Computadores*. [S.l.]: Mcgraw Hill Brasil, 2006. Citado 2 vezes nas páginas 31 e 32.

GOOGLE. *Google Acadêmico*: Site. 2014. Disponível em: <http://scholar.google.com.br>. Acesso em: 15 nov. 2014. Citado na página 45.

MURATNKONAR. *Audio Interchange File Format*: Site. 2014. Disponível em: <http://www.muratnkonar.com/aiff/index.html>. Acesso em: 03 nov. 2014. Citado na página 27.

NETO, L. Livros para ouvir. PublishNews, 2014. Disponível em: <http://www.publishnews.com.br/telas/noticias/detalhes.aspx?id=79097>. Acesso em: 11 nov. 2014. Citado na página 24.

NIKOLAJEVIC, V.; FETTEWIS, G. Computation of forward and inverse mdct using clenshaw's recurrence formula. IEEE Transactions on Signal Precessing, 2003. Disponível em: <http://www.infowester.com/histomptres.php>. Acesso em: 13 nov. 2014. Citado na página 39.

PALLETA, F. A. C.; WATANABE, E. T. Y.; PENILHA, D. F. Audiolivro: inovações tecnológicas, tendências e divulgação. 2008. Citado 2 vezes nas páginas 23 e 24.

REIS, H. C. *Especificação de um formato de arquivo open source para audiobooks com suporte a marcações de conteúdo*. 87 p. Monografia (Especialização) — Universidade de Brasília, 2013. Citado 7 vezes nas páginas 9, 19, 21, 43, 46, 49 e 53.

RFC_3533. *The Ogg Encapsulation Format Version 0*: Documento técnico. 2003. Disponível em: <http://xiph.org/ogg/doc/rfc3533.txt>. Acesso em: 03 nov. 2014. Citado 6 vezes nas páginas 13, 33, 34, 35, 36 e 37.

RHJ. *Home page*: Site. 2009. Disponível em: <http://www.editorarhj.com.br>. Acesso em: 29 out. 2014. Citado na página 24.

RIBEIRO, N.; TORRES, J. *Tecnologias de Compressão Multimédia*. [S.l.]: FCA, 2009. Citado na página 33.

SAYOOD, K. *Introduction to data compression*. [S.l.]: Newnes, 2012. Citado na página 30.

SCIELO. *Scientific Electronic Library Online*: Site. 2014. Disponível em: <http://www.scielo.org/php/index.php>. Acesso em: 14 nov. 2014. Citado na página 45.

SERRA, F. *Áudio Digital: A tecnologia aplicada à música e ao tratamento de som*. [S.l.]: Editora Ciência Moderna Ltda, 2002. Citado na página 26.

SOUZA, M. S. D.; CELVA, R. A.; HELVADJIAN, V. Audiolivro: um suporte para a educação literária. 2006. Disponível em: <http://ltp.emnuvens.com.br/ltp/article/viewFile/69/66>. Acesso em: 22 out. 2014. Citado 2 vezes nas páginas 23 e 24.

SOX. *Sound eXchande*: Site. 2014. Disponível em: <http://www.sox.sourceforge.net>. Acesso em: 20 set. 2014. Citado na página 48.

SVíTEK, J. Ogg vorbis: Subjective assessment of sound quality at very low bit rates. Cesnet Archiv, 2006. Disponível em: <http://archiv.cesnet.cz/doc/techzpravy/2006/vorbis/#Vor00>. Acesso em: 12 nov. 2014. Citado 2 vezes nas páginas 13 e 40.

TANENBAUM, A. S. *Organização Estruturada de Computadores*. 5 edição. ed. [S.l.]: Pearson Prentice Hall, 2007. Citado 2 vezes nas páginas 32 e 33.

TEIXEIRA, J. Leitura de ouvido. Veja, p. 135, n. 35, ano 39, 2006. Citado na página 23.

TENENBAUM, A. M. et al. *Estrutura de dados usando C*. [S.l.]: Pearson Makron Books, 2004. Citado na página 29.

TOCALIVROS: Site. 2014. Disponível em: <http://www.tocalivros.com>. Acesso em: 11 nov. 2014. Citado 2 vezes nas páginas 13 e 25.

VICTOR. *Phone Arena*: Site. 2014. Disponível em: <http://www.phonearena.com/news/Moores-Law-is-coming-to-an-end_id54127>. Acesso em: 24 set. 2014. Citado 2 vezes nas páginas 13 e 32.

WAGGENER, B. *Pulse Code Modulation Techniques*. [S.l.]: Springer, 1994. Citado na página 26.

XIPH.ORG. *Vorbis Audio Compression*: Site. 1994. Disponível em: <http://www.xiph.org/licenses/bsd/>. Acesso em: 15 nov. 2014. Citado na página 41.

XIPH.ORG. *Vorbis Audio Compression*: Site. 2000. Disponível em: <http://www.xiph.org/vorbis/>. Acesso em: 15 nov. 2014. Citado na página 41.

XIPH.ORG. *Vorbis I specification*: Site. 2012. Disponível em: <http://wiki.xiph.org/vorbis/doc/Vorbis_I_spec.html>. Acesso em: 03 nov. 2014. Citado 5 vezes nas páginas 33, 38, 39, 40 e 41.