Universidade Federal de Pernambuco

Centro de Informática

Pós-graduação em Ciência da Computação

MECANISMOS PARA MELHORIA DA TRANSFERÊNCIA DE AUDIO EM REDES PEER-TO-PEER

Márcio Leal de Melo Dahia

TESE DE DOUTORADO

Recife

Janeiro de 2009

Universidade Federal de Pernambuco
Centro de Informática

Márcio Leal de Melo Dahia

MECANISMOS PARA MELHORIA DA TRANSFERÊNCIA DE AUDIO EM REDES PEER-TO-PEER

Trabalho apresentado ao Programa de Pós-graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco como requisito parcial para obtenção do grau de Doutor em Ciência da Computação

Orientador: *Geber Lisboa Ramalho*

Recife

Janeiro de 2009

RESUMO

Dentre os principais problemas observados nas redes *peer-to-peer* de compartilhamento de conteúdo (RCAs) atuais como, por exemplo, *BitTorrent* e *eDonkey*, estão o download inadvertido de conteúdo diferente do desejado e a grande espera por conteúdo considerado infrequente na rede.

Esses problemas são, em parte, reflexos da concepção atual de dois processos fundamentais para o funcionamento dessas redes: a indexação, responsável por identificar e localizar o conteúdo na rede, e a segmentação, que divide os arquivos em pedaços de forma a possibilitar a transferência a partir de várias fontes simultaneamente. Nas RCAs atuais, esses processos são iguais independentemente das características (tipo de mídia e formato de arquivo, por exemplo) do conteúdo a ser transmitido, o que acaba limitando o real potencial da rede em alguns casos, como por exemplo, na transmissão de conteúdo fonográfico.

Este trabalho propõe novos processos de indexação e segmentação para RCAs que levam em consideração características estruturais e semânticas dos arquivos de áudio a fim de tornar o processo de download desse tipo de conteúdo mais eficiente e eficaz.

O novo processo de segmentação, ao contrário do atual, que divide os arquivos em partes de tamanhos iguais, separa o arquivo em partes inerentes de sua própria estrutura. Para o caso de áudio, portanto, duas segmentações são possíveis: a segmentação por tempo, em que um fonograma é divido em intervalos de tempo, independentemente de seu tamanho, e a segmentação por significância perceptiva, em que as partes resultantes correspondem a níveis crescentes de fidelidade perceptiva ao original.

O novo processo de indexação despreza diferenças pouco relevantes à percepção da música, como, por exemplo, diferenças em metadados e até algumas diferenças de qualidade, permitindo que fonogramas ligeiramente diferentes possam ser identificados como equivalentes, aumentando assim o número potencial de fontes disponíveis download.

Como ilustração de utilização dos novos processos, podemos citar um cenário em que o usuário pode escolher baixar uma música desconhecida em baixa qualidade, porém rapidamente, a fim de assegurar que ela é de seu agrado. E, em caso positivo, pode, posteriormente, baixar apenas os segmentos de significância perceptiva necessários para atingir o nível que satisfaça suas necessidades, dispondo, para isso, de múltiplas fontes, algumas delas não exatamente iguais.

Testamos os novos processos para o formato MP3. Os resultados obtidos indicam seu melhor desempenho, mantendo o consumo de recurso níveis admissíveis, o que aponta a possibilidade de aplicação desses conceitos para outros formatos e mídias, como imagens e vídeos.

Palavras-chaves: download, redes de compartilhamento de arquivos, indexação, segmentação.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

CAPÍTULO 1

INTRODUÇÃO

Cenário 1: *Jó, um homem justo e temente a Deus, navegando em seu blog favorito sobre música, leu ótimas críticas a respeito da recém-lançada canção* ***Pirata? Eu não!****, da banda Bucaneiro Sorridente, publicada sob licença Creative Commons. Curioso para ouvi-la, ele entra na sua Rede de Compartilhamento de Arquivos favorita e faz uma pesquisa pela música usando as palavras-chaves: "pirata", "eu" e "não". Na lista de resultados, aparecem várias ocorrências. Ele escolhe a que parece mais apropriada e aguarda pacientemente o fim do download. Na hora de ouvir, que decepção! Ele acabou baixando*[1] *a detestável música* ***Pirata eu! Não?****, lançada não faz muito tempo pela banda RIAA Enquanto Pode. Jó resigna-se e tenta a sorte em outra ocorrência da lista de resultados.*

Cenário 2: *Em outra ocasião, Jó lembra-se de procurar* ***Aquela****, música bastante antiga, uma de suas favoritas, e já fora de catálogo, que há muito vem procurando pelas lojas e meios convencionais. Decide, portanto, tentar a sorte nas Redes de Compartilhamento de Arquivos. Logo na primeira busca a resposta é animadora: dezenas de ocorrências de Aquela. "Boas novas, enfim...", pensa ele, pouco antes de se dar conta da realidade. Para cada ocorrência disponível nos resultados, há apenas 1 pessoa compartilhando o arquivo e isso, ele já sabe, é prenuncio de uma longa espera. Ele escolhe aleatoriamente uma delas e pensa consigo mesmo que vai precisar de*

[1] *"Baixar: transferir (software ou dados provenientes de um computador) para o computador que está sendo operado pelo usuário, estando os dois computadores conectados por linha telefônica ou por outro canal de telecomunicações"* (Houaiss 2007)

bastante sorte, perseverança, tempo, e, por que não dizer, fé para conseguir Aquela... Jó tem tempo, é perseverante e é fiel. Mas sorte nunca foi o forte de Jó...

Os cenários descritos anteriormente, longe de ser apenas ficcionais, apresentam situações com as quais já se deparou a maioria dos usuários das Redes de Compartilhamento de Arquivos (a partir desse ponto, RCAs), e com uma freqüência muito maior que a admissível.

A real gravidade dessa situação torna-se mais clara com a percepção da grande demanda de utilização dessas redes. Examinando-se os principais trabalhos sobre análise de tráfego da Internet dos últimos anos (CacheLogic 2004; 2005b; 2005a; Ellacoya 2007; Ipoque 2007; Sandine 2008), é possível chegar às conclusões listadas a seguir.

- Há sete anos, registra-se que a maior parte do tráfego da Internet (mais de 44% em dados de 2008) é composta por pacotes originados de demandas de RCAs[2];
- as duas RCAs mais utilizadas, *BitTorrent* e *eDonkey*, são responsáveis por mais de 80% desse tipo de tráfego;
- mais de 80% do tráfego em RCAs é composto de arquivos de vídeo e áudio.

A partir disso, duas hipóteses parecem bastante plausíveis apesar da falta de estudos empíricos para ratificá-las. A primeira é que o tráfego de conteúdo indesejável de originados das RCAs é não desprezível. Já a segunda diz que o tráfego não é maior hoje em razão da grande espera a que o usuário é submetido quando deseja um conteúdo encontrado em menor frequência na rede.

Sem dúvida, grandes avanços foram alcançados nas pesquisas em *peer-to-peer*, porém poucos tratam dos problemas descritos nos parágrafos anteriores (Christin, Weigend et

[2] Esse quadro vem mudando desde 2006 com a utilização maciça de streaming de vídeo. Apenas o site *Youtube* é responsável por 10% de todo o tráfego *Web* (Ellacoya 2007). Alguns estudos já apontam vantagem no tráfego *HTTP* (Sandine 2008). Em números absolutos, contundo, o uso das RCAs continua a crescer (Ipoque 2007; Sandine 2008).

al. 2005; Costa and Almeida 2007). Ainda assim, a adoção desses avanços em uma RCA existente hoje, implicaria adaptações de tal forma radicais que inviabilizariam a compatibilidade com a infraestrutura existente até o momento, o que significaria potencial perda massiva de usuário.

No escopo deste trabalho, analisemos duas das principais limitações atribuídas às RCAs atuais do ponto de vista do gerenciamento de conteúdo.

A primeira refere-se ao processo de recuperação[3] do conteúdo desejado, em que uma ou mais palavras consultadas são comparadas a palavras-chaves extraídas do nome do arquivo disponibilizado (Izal, Keller et al. 2004; Kulbak and Bickson 2005). Essa abordagem torna o processo muito vulnerável a erros de digitação, ortografia, homografia ou mesmo alteração proposital dos dados (Kubiatowicz 2003). A descoberta tardia de que o arquivo baixado é diferente daquilo que se desejava, como acontece com *Jó* no cenário descrito no primeiro parágrafo, ocorre com grande frequência.

A segunda limitação está relacionada à identificação do conteúdo a ser transferido, processo conhecido como indexação. A fim de tornar o processo de transferência mais eficiente, as RCAs utilizam a técnica de download de múltiplas fontes. Nesta técnica, um arquivo é dividido em segmentos contíguos (etapa chamada de segmentação). Segmentos diferentes podem ser requisitados a *peers* distintos simultaneamente, aumentando assim a banda disponível.

Apesar de já representar um grande avanço em relação ao download sequencial tradicional, esta técnica é muito sensível a mudanças no conteúdo devido a forma como foi construída. Arquivos com diferença em apenas um bit, por exemplo, são considerados distintos, fato que impede que possam ser utilizados como fonte de download entre si (Chun, Wu et al. 2006).

[3] Na acepção semelhante à da palavra *retrieval* da língua inglesa

Podemos concluir, portanto, que parte das deficiências das RCAs nos processos de recuperação e transferência decorrem principalmente da interpretação dos arquivos como meros repositórios de bits, ignorando suas características conceituais e estruturais.

1.1 OBJETIVOS DO TRABALHO

Este trabalho apresenta uma proposta de melhoria da experiência de transferência de conteúdo fonográfico em RCAs. Tal proposta fundamenta-se na hipótese de que levar em consideração as particularidades do conteúdo a ser transmitido pode tornar o processo de transferência mais eficiente e eficaz e pode, ainda, minimizar consideravelmente os efeitos de eventuais erros no processo de pesquisa.

Em relação a áudio, devemos observar três características principais. A primeira é a existência da dimensão tempo em qualquer entidade sonora. A discretização do tempo característica da digitalização do som cria unidades naturais de segmentação.

A segunda característica se refere ao fato de que fonogramas com representação binária distinta (por exemplo, o mesmo fonograma codificado em formatos diferentes) podem ser considerados iguais, ou suficientemente similares, do ponto de vista acústico e psicoacústico e, por isso, poderiam ser usados como fonte de transferência uns para os outros. *Jó*, por exemplo, poderia ser beneficiado, no cenário 2, caso algumas ocorrências de ***Aquela*** encontradas fossem agrupadas (o que implicaria mais de uma pessoa compartilhando ***Aquela***).

A terceira, também associada percepção da música é a chamada significância perceptiva. Segundo estudos (Wang, Ojanpera et al. 2001a; Wang, Huang et al. 2004a), algumas faixas de freqüência são estritamente necessárias para o “entendimento” do som (alta significância perceptiva) enquanto outras meramente “aumentam sua qualidade” (baixa significância perceptiva).

Este trabalho propõe novos processos de indexação e segmentação para RCAs que levam em consideração características estruturais e semânticas dos arquivos de áudio a fim de tornar o processo de download desse tipo de conteúdo mais eficiente e eficaz.

Os novos processos de segmentação propostos, ao contrário do atual, que divide os arquivos em partes de tamanhos iguais, separam o arquivo em partes inerentes de sua própria estrutura. Duas técnicas de segmentação são definidas. A primeira é denominada Segmentação em Intervalo de Tempo, em que um fonograma é divido em partes correspondentes a, por exemplo, 10 segundos, independentemente de seu tamanho. Dessa forma, dois segmentos contíguos estarão sempre sincronizados.

A segunda chama-se Segmentação por Significância Perceptiva. Com ela, cada segmento corresponde a um nível de fidelidade perceptiva ao arquivo original. O primeiro segmento corresponde ao maior nível de significância perceptiva, o segundo, ao segundo maior e assim sucessivamente. Com essa segmentação, é possível baixar apenas o primeiro segmento de um arquivo para identificá-lo, para depois baixar o restante.

Os novos processos de indexação desprezam diferenças pouco relevantes à percepção da música, permitindo que fonogramas ligeiramente diferentes em suas representações binárias possam ser identificados como equivalentes, aumentando assim o número de fontes disponíveis download. Para isso 3 técnicas foram criadas e são utilizadas em conjunto. A primeira, chamada Indexação Bit-A-Bit, é similar a indexação encontradas nas RCAs atuais, ela serve para a identificação do arquivo e verificação de integridade da transferência. A segunda, Indexação Sensível ao Conteúdo, ignora informações irrelevantes ao sinal musical, propriamente dito, como, metadados, por exemplo. E, a terceira, chamada Indexação de Frankenstein, desconsidera a representação binária, observando exclusivamente as características do sinal codificado (Wang, Huang et al. 2004b). Dessa forma, é possível identificar um dado fonograma codificado em formato MP3, por exemplo, com diferentes parâmetros ou codificadores.

Como estudo de caso, implementamos as indexações e segmentações para arquivos no formato de *MPEG Layer-3* (MP3), que permanece, contrariando todas as previsões, o formato mais utilizado para codificações de áudio com perdas.

1.2 PRINCIPAIS DIFICULDADES

Algumas questões são essenciais para determinar a viabilidade dessa proposta e, por isso, este trabalho visa respondê-las. São elas:

- Qual o custo computacional das novas soluções? É um custo viável?
- Qual o esforço para implantá-las nas RCAs atuais e qual o custo disso?
- O quanto é possível generalizar essas ideias para outros formatos de áudio?
- E outros formatos de arquivo?
- Como comparar as soluções atuais com as propostas?

1.3 ORGANIZAÇÃO DO DOCUMENTO

No próximo capítulo, discorreremos sobre os principais problemas relacionados ao estado da prática das RCAs e como ele influencia a transferência de arquivos de áudio. No capítulo 3, apresentaremos os principais trabalhos que visam suplantar as dificuldades atuais, bem como os principais caminhos que, em nossa opinião, nos levam em direção a transferências de arquivos de áudio mais eficientes e eficazes. No capítulo 4, apresentaremos algumas características do formato MP3 essenciais a uma melhor compreensão da proposta, apresentada no capítulo 5, seguinte. No capítulo 6, apresentaremos os principais resultados obtidos e, por fim, as conclusões no capítulo 7

CAPÍTULO 2

LIMITAÇÕES DAS REDES DE COMPARTILHAMENTO DE ARQUIVOS

Iniciaremos esse capítulo com a elucidação de algumas características esperadas em um sistema de distribuição de áudio. Para isso será necessária uma pequena introdução às principais características das redes *peer-to-peer*. Embora os conceitos de rede *peer-to-peer* e compartilhamento de arquivo sejam independentes, podendo existir tanto RCAs que não sejam *peer-to-peer* e vice-versa, é utilizando os conceitos da primeira que a segunda encontra sua utilização plena.

Em seguida, descreveremos o funcionamento das duas RCAs mais utilizadas no momento, *BitTorrent* e *eDonkey*, enfatizando os processos de recuperação e transferência de arquivos e suas respectivas limitações em atender às propriedades descritas inicialmente.

2.1 REDES PEER-TO-PEER

Milojicic (Milojicic, Kalogeraki et al. 2002), faz a seguinte definição do termo *peer-to-peer*:

> O termo peer-to-peer (P2P) refere-se à classe de sistemas e aplicações que empregam recursos distribuídos para executar uma função crítica de maneira descentralizada. Com a distribuição pervasiva de computadores, P2P está cada vez mais recebendo atenção nos círculos de pesquisas, desenvolvimento de produtos e investimentos. O interesse abrange desde entusiasmo passando pelo exagero, até a descrença em seu potencial. Alguns benefícios da abordagem P2P incluem: melhora na escalabilidade evitando a dependência de pontos centralizados; a eliminação da necessidade de infraestrutura cara ao permitir comunicação direta entre clientes e a possibilidade da agregação de recursos. (Milojicic, Kalogeraki et al. 2002)

Esse conceito é corroborado por Rocha (Rocha, Domingues et al. 2004), onde é possível encontrar (no trabalho, um *peer* conectado à rede é denominado nó):

> Redes peer-to-peer são redes virtuais que funcionam na Internet com o objetivo de compartilhar recursos entre os participantes, sendo que por princípio não há diferenciação entre os participantes. Em geral, é aceito pela comunidade que sistemas peer-to-peer devem suportar os seguintes requisitos:
>
> - Nós que podem estar localizados nas bordas da rede;
> - Nós com conectividade variável ou temporária e endereços também variáveis;
> - A capacidade de lidar com diferentes taxas de transmissão entre nós;
> - Nós com autonomia parcial ou total em relação a um servidor centralizado;
> - Assegurar que os nós possuem capacidades iguais de fornecer e consumir recursos de seus peers;
> - A rede deve ser escalável;
> - A capacidade dos nós se comunicarem diretamente uns com os outros.
>
> Tendo todas estas características, uma rede pode ser dita peer-to-peer, mesmo que algumas das funções de controle da rede estejam localizadas em um servidor central (ponto de falha). (Rocha, Domingues et al. 2004)

De fato, a grande importância das redes *peer-to-peer* está no fato de que ela proporciona uma opção ao convencional paradigma centralizado da Internet. Naquelas, os usuários comuns passam a ter papel fundamental (e não mais apenas coadjuvantes) na publicação e disponibilização de conteúdo. Por isso, um *peer* ou nó, é também, por vezes, denominando *Servent*, palavra-valise originada a partir de *server* e *client*, pelo fato de poder se comportar simultaneamente como servidor e como cliente (Meyers 2004).

2.1.1 Arquitetura das redes peer-to-peer

Segundo, quanto à organização arquitetural, as redes podem ser divididas em três categorias: descentralizadas, hibridas ou baseadas em supernós e centralizadas, sua principal diferença está no tráfego de mensagens de controle (mensagens que permitem

a localização de outros *peers* ou os recursos a serem usados) (Rocha, Domingues et al. 2004). Abaixo, uma explicação mais detalhada de cada uma delas.

DESCENTRALIZADAS

São redes onde toda e qualquer informação trafega entre os *peers*. Neste modelo, os nós são autônomos e todos são responsáveis tanto por compartilhamento de recursos, quanto por transmissão de informações de controle (como a localização de um *peer* ou um recurso). Os nós podem se comunicar de maneira direta ou através de vizinhos comuns (Rocha, Domingues et al. 2004). A Figura 1(a) apresenta um diagrama deste tipo de rede. Exemplos de redes com essa arquitetura são *freenode*[4] e *gnutella*[5].

HÍBRIDAS OU BASEADAS EM SUPERNÓ

São redes onde há diferença de papeis entre os nós. Nelas existem *peers* chamados supernós (Figura 1, subfigura c) a quem é delegada a responsabilidade do transporte de informações de controle. Para ingressarem na rede, os *peers* convencionais, isto é, aqueles que trocam apenas as mensagens sobre a troca de recursos, precisam estar conectados a algum supernó, o que cria um ponto de falha na rede (caso o supernó venha a se desconectar, todos os *peers* a ele conectados são desconectados da rede, por consequência) (Rocha, Domingues et al. 2004). Um exemplo desse tipo de rede é a RCA *Kazaa*[6].

CENTRALIZADAS

Nestas redes, existem um ou vários pontos centrais, responsáveis exclusivamente pelo transporte de informações de controle, como pode ser visto na Figura 1 (b). Para ingressar na rede, os *peers* devem se conectar a esses pontos centrais (denominados

[4] http://freenode.net/

[5] http://www.limewire.com/

[6] http://www.kazaa.com/

servidores, concentradores ou *brokers*). Estão nessa classe as RCAs *BitTorrent* e *eDonkey*, que serão examinadas neste capítulo (Rocha, Domingues et al. 2004).

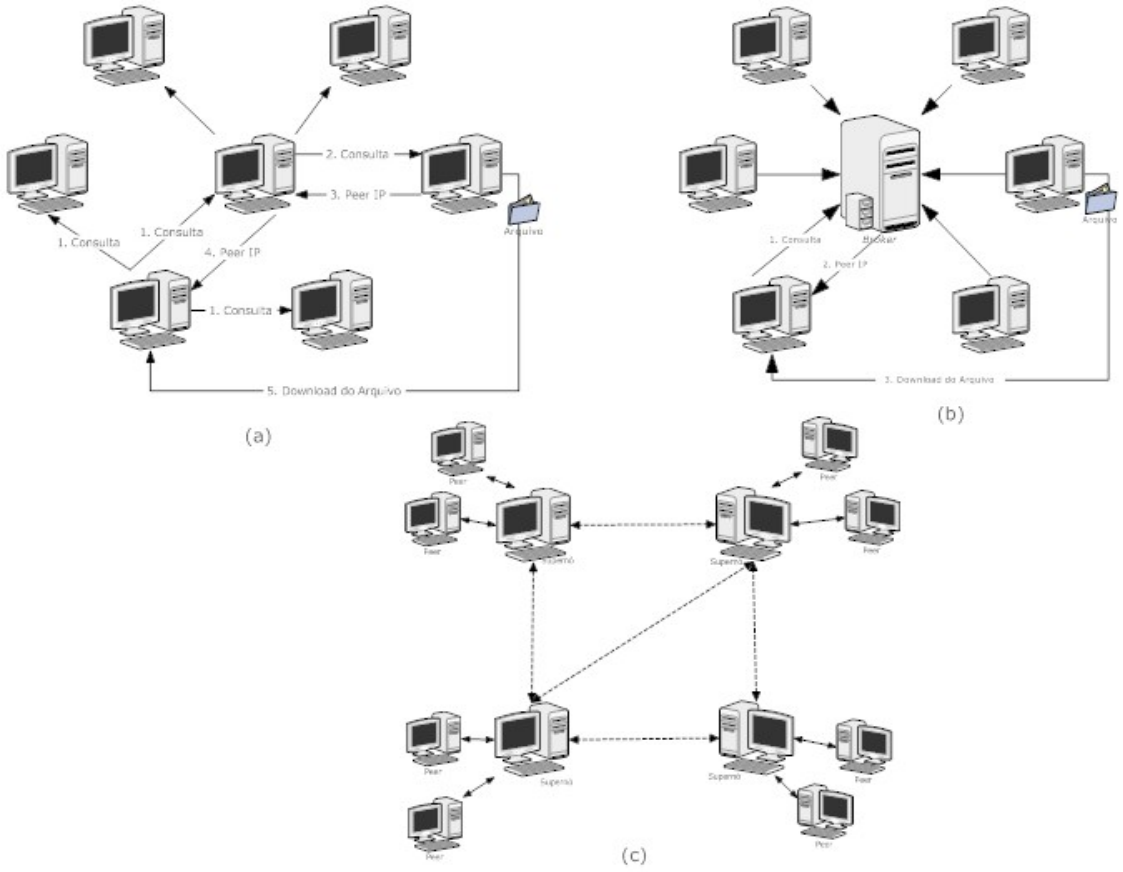

Figura 1 – Tipos de arquiteturas de redes *peer-to-peer*: (a) descentralizada, (b) centralizada e (c) híbridas.

Vale a pena frisar que existe compromisso entre descentralização e facilidade de encontrar um recurso. Quanto mais descentralizada é a rede, maior sua autonomia, porém também aumenta a dificuldade de encontrar recursos na rede.

2.2 REQUISITOS NÃO-FUNCIONAIS DE UMA REDE DE COMPARTILHAMENTO DE CONTEÚDO DE ÁUDIO

Podemos dividir os requisitos dessa classe de sistemas em dois grupos: as que são gerais para toda rede de compartilhamento de conteúdo e aquelas que são especificas ao

compartilhamento de áudio. A seguir, discutiremos separadamente sobre cada uma delas.

2.2.1 Requisitos compartilhados por todas as RCAs

Daswani (Daswani, Garcia-Molina et al. 2002) apresenta em seu trabalho, uma série de requisitos desejáveis em RCAs. Segue comentários sobre cada um deles.

ROBUSTEZ

É a habilidade do sistema em se manter funcional apesar da presença de falhas. No caso específico das RCAs, equivale a manter os processos de busca e transferência ativos mesmo em situações de saída abrupta de *peers* ou do servidor de busca, caso exista (Daswani, Garcia-Molina et al. 2002).

AUTONOMIA

Pode ser vista como a liberdade para que cada *peer* armazene e compartilhe o que quiser e com quem quiser, podendo negar-se a atender requisições pelo motivo que achar apropriado. O *peer* também tem total liberdade para tentar estabelecer conexões com outro *peers* e sair da rede quando desejar.

ALTA CONECTIVIDADE

Os *peers* geralmente não possuem endereço IP fixo e com frequência utilizam dispositivos como NAT[7] e *Firewall*, o que dificulta o estabelecimento de conexões iniciadas por uma entidade externa. A alta conectividade refere-se à capacidade de suplantar tais dificuldades para permitir que o maior número possível de *peers* possa utilizar a rede (RCA) de forma plena, podendo iniciar e receber pedidos de conexão para ou de qualquer outro *peer*.

[7] *Network Address Translator* é um dispositivo que fornece, temporariamente, a maquinas com endereço de rede local, endereços IP válidos, para que estas possam utilizar a Internet. O NAT é usado quando o número de endereços IP válidos disponíveis é menor que o número de máquinas que deles precisam.

CONFIABILIDADE

Uma RCA confiável é aquela em que há garantias de equivalência entre o que está sendo baixado e o que o usuário acredita que está baixando. A replicação de conteúdo, razão principal do sucesso das RCAs, e a freqüente disponibilização de conteúdo protegido por direitos autorais resultaram no surgimento dos chamados *digital decoys*. Trata-se de arquivos defeituosos ou preparados intencionalmente com metadados incompatíveis com o seu conteúdo, cujo objetivo é dificultar o acesso do usuário ao conteúdo correto (Christin, Weigend et al. 2005).

JUSTIÇA

Geralmente, a demanda por arquivos de um determinado *peer* é bem maior que sua capacidade de supri-la. Por esse motivo, há a necessidade de criação de filas de espera. A justiça se refere à ordenação dos usuários nesta fila. Um escalonamento justo numa RCA deve levar em consideração, além do tempo de espera, a quantidade de dados que cada membro da fila já "serviu" a outros usuários, prevenindo a existência de *peers* que apenas baixam e nunca servem arquivos (chamados de *free riders*).

DISPONIBILIDADE DE CONTEÚDO

A natureza volátil dos *peers* tem, como conseqüência, a incerteza sobre a preservação de conteúdos pouco populares na rede. No caso crítico, em que apenas um usuário possui determinado arquivo, os interessados em tal conteúdo nada podem fazer além de aguardar sua entrada na rede. Nesse caso, aumentar a disponibilidade (o número de fontes) do conteúdo é o que pode ser feito para comprovadamente aumentar a eficiência da rede.

PRECISÃO NA TRANSFERÊNCIA

A precisão é o resultado da soma de uma transferência Eficiente e Eficaz simultaneamente.

A eficiência na transferência é obtida com a utilização do canal de comunicação em toa sua capacidade. Em RCAs, a eficiência está muito relacionada a outro requisito: disponibilidade. Quanto maior a quantidade de fontes, maior a probabilidade de se utilizar eficientemente o canal de comunicação.

Já a eficácia é obtida com a escolha correta do que baixar, evitando a ocupação do canal de comunicação inutilmente com conteúdo diferente do desejado pelo usuário. A eficácia está bastante relacionada com o requisito da confiabilidade. Quando mais confiável a rede, menos tempo será perdido em downloads indesejáveis.

SEGURANÇA

O software deve prezar pelos cinco princípios da segurança de informação: disponibilidade, utilização, integridade, autenticidade e confidencialidade (Pfleeger and Pfleeger 2003).

DESEMPENHO

De todo *software*, espera-se que este desempenhe sua função utilizando o mínimo possível de recursos (memória, espaço de armazenamento e tempo de processador). Uma característica peculiar em um *software* de acesso a redes RCAs que agrava essa necessidade é o fato de que ele permanece a maior parte do tempo em *background* (uma vez que o uso da RCA não é uma atividade fim), sendo comum o usuário utilizar a máquina simultaneamente para outros propósitos.

2.2.2 Requisitos peculiares a uma RCA de áudio

EFICÁCIA NAS BUSCAS

Embora seja um requisito desejado em qualquer RCAs, há certas particularidades associadas ao tipo de conteúdo a ser tratado (Pachet 2003), algumas delas são:

- Nem sempre o usuário sabe exatamente o que deseja. É comum o interesse por uma música cujo nome não é conhecido.

- Em algumas ocasiões, não se deseja um objeto especifico. Amantes da música costumam buscar por novas sonoridades, estilos e/ou artistas não conhecidos.
- Determinados metadados como, por exemplo, estilo ou nome da banda são, algumas vezes, mais relevantes que o nome da música.

A partir dessas informações podemos concluir que a precisão absoluta dos resultados não implica uma busca eficaz.

ADAPTABILIDADE

Há vários fatores que influem nas necessidades do usuário por determinado conteúdo. Como exemplo, podemos citar três deles.

- Capacidade de armazenamento, processamento ou resolução do dispositivo onde o arquivo será reproduzido (por exemplo, celular monofônico *versus* sistema de som de alta definição).
- As características ou limitações de percepção de sons por parte do usuário (por exemplo, compressão transparente[8] a uma determinada taxa de bits).
- Finalidade de uso do conteúdo (por exemplo, se o usuário pretende descobrir se gosta de determinada música *versus* usá-la para fazer uma composição).

Nem sempre é possível encontrar em uma RCA um objeto que satisfaça todas as características exigidas pelo usuário. A adaptação no conteúdo traria como conseqüência, um aumento significativo na eficácia na transferência. Voltando ao cenário 1 apresentado no primeiro capítulo, a RCA usada por *Jó* poderia permitir a possibilidade de baixar de ***Pirata? Eu não!*** apenas as partes de alta significância perceptiva, a fim de permitir que *Jó* confirmasse tratar-se da musica desejada. Dessa forma, *Jô* poderia detectar mais cedo o fato de estar baixando a música errada.

[8] Um arquivo comprimido está em compressão transparente quando não é possível distingui-lo do seu original. Evidentemente, trata-se de um conceito subjetivo.

Na seção 2.3 , discutiremos o funcionamento das RCAs sob a ótica de alguns dos requisitos apresentados até o momento e que são tratados diretamente na nossa proposta: confiabilidade, eficácia na transferência, eficácia nas buscas e adaptabilidade.

2.3 FUNCIONAMENTO DE UMA RCA

Como foi dito anteriormente, analisaremos apenas as redes *eDonkey* e *BitTorrent*, por representarem a imensa maioria do que é transmitido em RCAs.

Buscando objetividade, aprofundaremos a discussão apenas sobre os requisitos citados na seção 2.2 que têm influência direta no tema desta proposta, a saber: confiabilidade, disponibilidade de conteúdo, precisão na transferência, e adaptabilidade. Quanto aos outros requisitos, podemos fazer o seguinte resumo: RCAs possuem pouca robustez e autonomia, porém permitem alta conectividade (já que é possível utilizar o servidor de busca como intermediador de conexões que não possam ser feitas diretamente). As características de eficácia na busca e justiça serão citadas brevemente no decorrer do texto, quando oportuno. Os requisitos de desempenho e segurança estão fora do escopo deste trabalho.

Em linhas gerais, as duas RCAs funcionam de forma bastante similar (ver figura 2), variando apenas em algumas características específicas. Discutiremos o funcionamento da rede dividindo-o em duas partes: a primeira, que chamaremos de **etapa de comunicação *peer*-servidor**, envolve a entrada do *peer* na rede, a disponibilização dos arquivos (segmentação e indexação) e a busca pelo conteúdo desejado. Em redes centralizadas, nessa etapa, não há comunicação entre os *peers*. A segunda parte envolve o processo de transferência propriamente dito. Chamá-la-emos de **etapa comunicação *peer-peer***, que envolve a seleção dos *peers* a quem serão requisitados os segmentos e a checagem de integridade além da transferência.

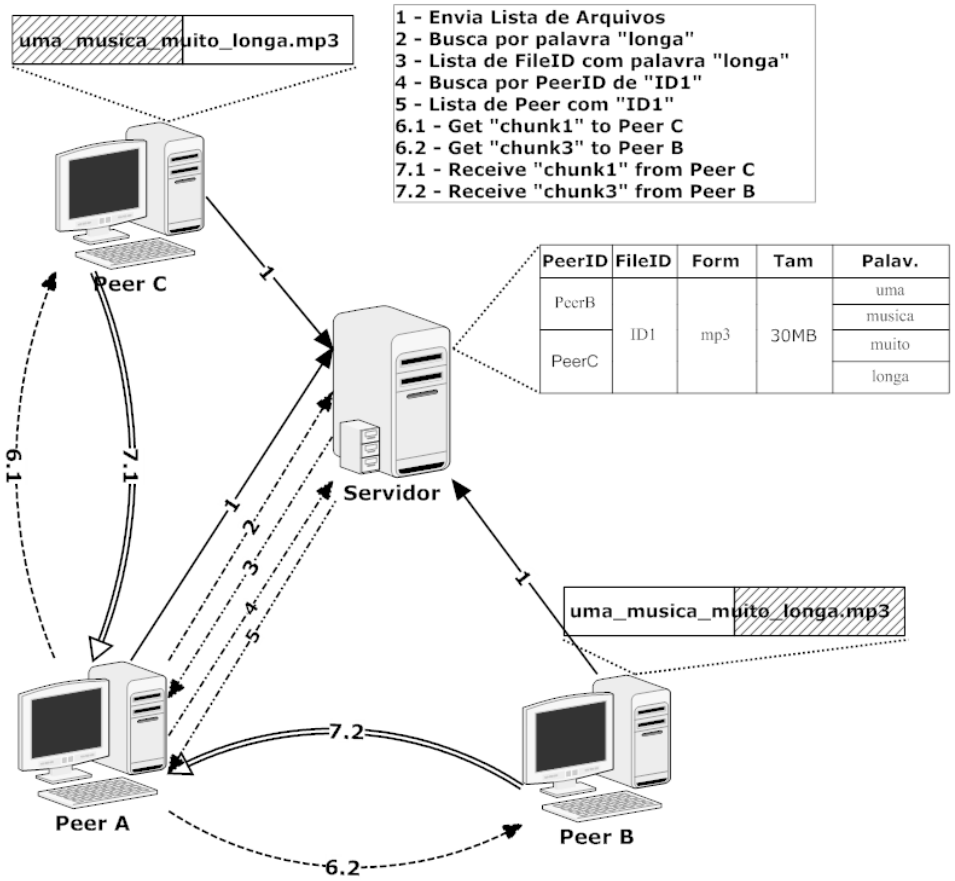

Figura 2 – Funcionamento de uma RCA com download de múltiplas fontes.

Nas seções seguintes, apresentaremos as etapas de busca e transferência para cada uma das redes citadas anteriormente (*eDonkey* e *BitTorrent*), identificando as diferenças e similaridades entre elas.

2.3.1 Comunicação PEER-SERVIDOR na rede eDonkey

Para entrar numa rede *eDonkey*, antes de tudo é necessário conhecer o IP de, pelo menos, um dos servidores de busca da rede. Existem milhares de servidores espalhados pelo mundo e geralmente os softwares *Servent*, como o *eMule*[9], costumam trazer uma lista extensa deles que é atualizada quando da primeira conexão bem sucedida. Nessa

[9] http://www.emule-project.net/

fase, é definido o *peerID*, número que identifica unicamente cada usuário na rede em um dado momento (Kulbak and Bickson 2005).

Uma vez conectado, o *peer* envia ao servidor a lista de arquivos que deseja compartilhar. Nesse processo de indexação do conteúdo, para cada arquivo, o *peer* processa cinco itens (figura 3):

- O primeiro é o conjunto formado por todas as palavras-chaves extraídas do nome do arquivo;
- O segundo é a terminação do arquivo que indica o seu formato;
- O terceiro é o tamanho do arquivo em Megabytes;
- O quarto é o conjunto de *chunkIDs*. Cada elemento do conjunto (um *chunkID*) é o resultado da aplicação do algoritmo de *hash* MD4 (Rivest 1992) para cada segmento contíguo de 9.28MB do arquivo. O último segmento pode ser menor que os outros (Kulbak and Bickson 2005). O resultado são elementos de 128 bits (resultado do MD4 para cada *chunkID*);
- O quinto e último é o *fileID*, que é o resultado da aplicação do MD4 no conjunto de *chunkIDs*.

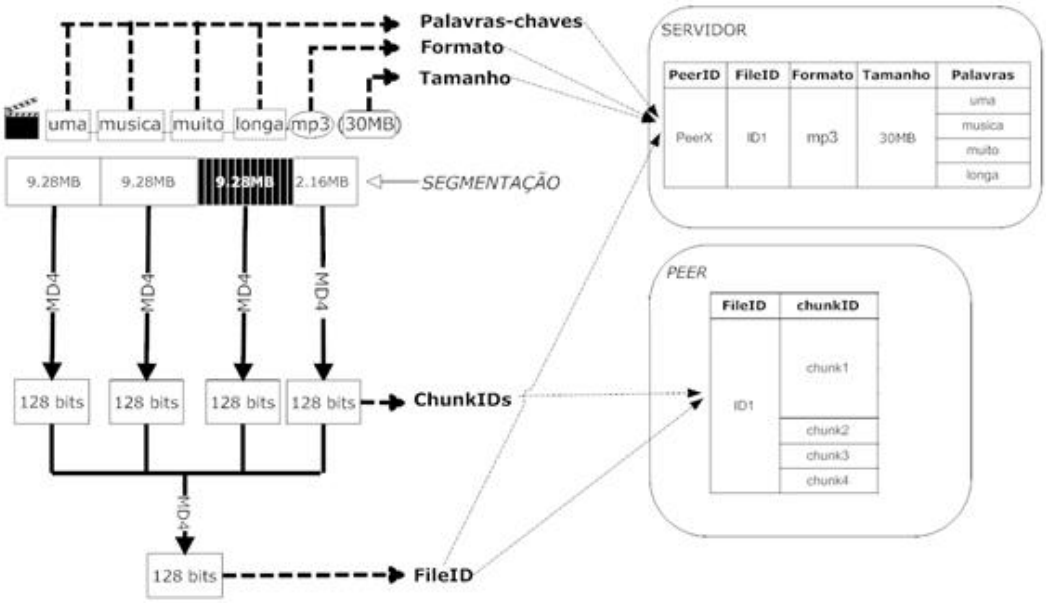

Figura 3 – Processo de indexação e segmentação de arquivos no eDonkey.

Os objetivos da aplicação do MD4 nos arquivos e nos segmentos são dois. O primeiro é criar um identificador único na rede para cada um dos elementos, com pequena probabilidade de colisão. Com isso, é possível, não só localizar arquivos facilmente entre os *peers*, como também os seus segmentos. O segundo objetivo é verificar a integridade dos segmentos e arquivos baixados, aplicando o MD4 no conteúdo baixado e comparando-o com os valores dos *chunkIDs* e *fileID* de referência para certificar que foram baixados corretamente.

As palavras-chaves, o tamanho, o formato e o *fileID* são enviados ao servidor, que, por sua vez, os insere em uma tabela global, chamada **tabela de busca.** A associação entre *fileID*, *peerID* e *chunkID* é mantida no próprio *peer* e será discutida na seção 2.3.2 .

Quando um *peer* deseja encontrar um arquivo, consultando a palavra-chave "longa", como na figura 2 (passo 2), o servidor devolve todas as ocorrências de "longa" encontradas na tabela de busca (passo 3). Ao decidir qual o arquivo que deseja baixar, no nosso exemplo, "`uma_musica_muito_longa.mp3`", o *peer* envia uma mensagem requisitando informações sobre quais os *peers* que possuem algum segmento do arquivo cujo *fileID* é "ID1" (passo 4). O servidor, finalmente, retorna todas as ocorrências de "ID1" na tabela de segmentos. A partir desse momento, inicia-se a comunicação entre os *peers*.

2.3.2 Etapa de comunicação PEER-PEER na rede eDonkey

Uma vez escolhido o arquivo a ser baixado e de posse da lista dos *peers* que possuem o arquivo desejado, ou segmentos dele, inicia-se o processo de requisição dos segmentos a esses *peers* (passos 6.1 e 6.2 na figura 2).

Após se certificar de que os *peers* constantes da lista enviada pelo servidor de fato estão conectados, o *peer* solicitante requisita uma sessão de download a cada um dos *peers* da lista. Estes, por sua vez, inserem o pedido em suas filas de requisições. A prioridade de cada cliente é calculada com base no tempo de permanência na fila e por um modificador de prioridade, seguindo a seguinte formula:

```
pontuação = (modificador * segundos_na_fila)/100
```

Quanto maior o valor de `pontuação`, mais próximo do topo da fila. O modificador recebe valor 100 como padrão e tem seu valor a influenciado pelas seguintes informações:

- crédito (quantidade de bytes já servidos) possuído pelo solicitante, que pode assumir valores de 1 a 10;
- prioridade estabelecida pelo solicitante , cujo valor vai de 0.2 a 1.8;

Ao atingir o topo da fila, o solicitante recebe uma mensagem do solicitado, avisando da situação. O solicitante pode então cancelar o processo, caso tenha já tenha recebido o arquivo de outros *peers*, ou requisitar um segmento identificado pelo *chunkID*.

Ao fim do processo de transferência, o *peer* solicitante checa a integridade do segmento recebido aplicando-se o algoritmo MD4 no segmento recebido e comparando-o com o valor *chunkID*. Caso os valores sejam diferentes, deduz-se que houve uma falha de durante a transferência. Na rede *eDonkey* tradicional, o segmento é eliminado e o processo de solicitação de segmentos inicia-se. O software *eMule*, porém, permite uma extensão a essa fase. Trata-se de um segundo nível de segmentação, desta vez com sub-segmentos de 180KB e utilizando um algoritmo de *hash* mais robusto, o SHA-1 (Eastlake and Jones 2001), cujo resultado é dado em 160 bits. Os sub-segmentos são requisitados um a um (aplicando-se a mesma técnica de checagem de integridade de segmento citada anteriormente) até que o segmento como um todo passe no teste de integridade.

Assim que o primeiro segmento está disponível, o *peer* atualiza sua lista de arquivos disponíveis no servidor, podendo imediatamente se tornar fonte para outros *peers* que desejem o arquivo.

Além dessa extensão já citada, o *eMule* permite que os segmentos sejam compactados no formato ZIP antes de serem enviados, aumentando, em alguns casos, a eficácia de utilização da largura de banda.

Os softwares clientes das redes *eDonkey* permitem que se configure o número máximo de conexões simultâneas para o recebimento e envio de arquivos, o que configurado corretamente pode aumentar a eficiência.

2.3.3 Etapa de comunicação PEER-SERVIDOR na rede BitTorrent

Diferentemente da rede *eDonkey*, a rede *BitTorrent* não abrange a etapa de busca. Todo o processo é realizado por entidades externas, geralmente, servidores *Web* de terceiros. O trabalho dos servidores da rede *BitTorrent*, chamados de *trackers*, consiste em manter uma lista atualizada dos *peers* ativos e de seus arquivos ou segmentos de arquivo.

Para disponibilizar conteúdo na rede, o usuário deve criar um arquivo especial de metadados (usualmente com terminação `.torrent)` e em seguida cadastrá-lo em um servidor Web especializado, associando-o a palavras-chaves apropriadas. Esses arquivos contêm as seguintes informações:

- O nome e o tamanho do arquivo ou lista de nomes e tamanhos quando se tratar de mais de um arquivo;
- O endereço do *tracker* ou lista de *trackers* que mantém a identificação dos usuários que possuem o arquivo ou a lista de arquivos;
- O tamanho de cada segmento que é determinado pela quantidade de segmentos que serão gerados. Tipicamente, o tamanho do segmento é de 256 KB;
- O *hash* SHA-1 de cada segmento, para identificação e verificação de integridade (da mesma forma que o *chunkID* do *eDonkey*). Em ambos os casos a intenção é criar um identificador único que possibilite a verificação da integridade após o *download* ;

2.3.4 Etapa de comunicação PEER-PEER na rede BitTorrent

De posse do arquivo *torrent* correspondente ao conteúdo desejado, o cliente *BitTorrent* requisita ao *tracker* (ou *trackers*, no caso de existir mais de um) a lista dos *peers* que possuem tal conteúdo. Essa lista geralmente contém o endereço de 50 *peers* escolhidos aleatoriamente do rol de todos os *peers* que possuem o conteúdo inteiramente (chamados de *seeds*) ou dos que possuem apenas parte dele (chamados de *leechers*).

Nesse momento, o solicitante escolhe que segmento deseja baixar usando a estratégia a requisição dos segmentos que possuem menor quantidade de fontes (mais raros) primeiro, e depois envia o pedido a todos os *peers* que possuem tal segmento. Da mesma forma que a rede *eDonkey*, os *peers* da rede *BitTorrent* também mantêm uma fila de espera, porém o algoritmo de *ranking* funciona de forma diferente (Legout 2006):

1. A cada 10 segundos há um re-ordenamento na fila de acordo com o tempo cedido por cada *peer* a transferência de segmentos do arquivo em questão. Os três mais bem colocados iniciam o processo de transferência.
2. A cada 30 segundos, um *peer* da lista escolhido aleatoriamente é liberado para efetuar o *download* do segmento desejado.

Os softwares clientes para redes *BitTorrent* permitem ainda configurar o número total de conexões simultâneas de recebimento de arquivo, o que configurado corretamente otimiza a utilização da banda.

2.4 DISCUSSÃO

As redes apresentadas neste capítulo representam os modelos mais difundidos em transferência de conteúdo, sendo responsáveis, no Brasil, por 85% do tráfego das RCAs (Silvestre, Kamienski et al. 2006) . Apesar disso, ambas falham em atender alguns dos requisitos esperados tanto em RCAs convencionais, quanto em uma hipotética rede de transferência de áudio. Resumidamente, as falhas estão em (i) não prover meios

eficientes de identificar inequivocamente o conteúdo antes de sua transferência (falta de confiabilidade e ineficácia nas buscas), (ii) não prover adaptabilidade de conteúdo visando às necessidades do usuário e (iii) não prover meios de aumentar a disponibilidade (encontrar mais fontes por arquivos). Por conseqüência, falham em prover um processo de transferência mais eficaz. A Tabela 1 compila os requisitos mais relevantes e como as RCAs apresentadas lidam com o assunto. No capítulo seguinte, discutiremos alguns esforços para melhorar as RCAs existentes e tecnologias que podem auxiliar no atendimento a alguns requisitos discutidos aqui.

Requisito	***eDonkey***	***BitTorrent***
Adaptabilidade	Cliente *eMule* permite comprimir um segmento antes de enviá-lo . A compressão não traz benefícios na transferência de arquivos de áudio já comprimidos.	Não provê
Precisão na transferência	A compressão aumenta a eficácia em alguns casos, porém o processo de escolha dos *peers* e o algoritmo de *ranking* levam a downloads lentos.	O algoritmo dinâmico de *ranking* e a transferência dos segmentos mais raros primeiro tornam o processo download relativamente rápido.
Confiabilidade	O uso de combinado de palavras-chaves com *hash* não é suficiente para garantir a confiabilidade.	Mesmo problema ocorre para redes *BitTorrent*.

Disponibilidade	Pode requisitar segmentos mais raros primeiro, mas, na prática, essa funcionalidade esbarra na longa espera nas filas.	O processo de segmentos mais raros primeiro e a escolha periódica de servir *peers* sem méritos, aumentam o número de *peers* com segmentos.

Tabela 1 – Associação entre os requisitos e a realidade das RCAs *eDonkey* e *BitTorrent.*

CAPÍTULO 3

AVANÇOS TECNOLÓGICOS PARA EXPERIÊNCIA DE DOWNLOAD EM RCAS

Nesse capítulo, apresentaremos algumas tecnologias que influenciam a melhoria dos processos de busca e transferência de conteúdo. Algumas delas já são usadas hoje, enquanto outras são apresentadas como possíveis fontes de melhoria.

Esse capítulo está dividido em duas seções. Na 3.1 , será apresentada a abordagem mais efetiva visando uma maior eficiência no processo de transferência e com grande importância para o trabalho que será desenvolvido. Na seção 3.2 , apresentaremos tecnologias que, em nossa opinião, podem vir a ser utilizadas para melhorar tanto o processo de busca quanto o de transferência em redes RCAs.

3.1 SET (SIMILARITY-ENHANCED TRANSFER)

Essa nova técnica ultrapassou a barreira acadêmica e foi veiculada por órgãos internacionais de notícia (BBCNews 2007) por permitir uma redução no tempo de download de até 500% em relação ao método de transferência tradicional do *BitTorrent*.

A idéia principal por trás do SET é aprimorar o método tradicional de download de múltiplas fontes das RCAs identificando segmentos iguais, ou seja, com a mesma representação binária, em arquivos diferentes e, assim, usar o segmento como fonte no processo de download de qualquer arquivo que o possua. Com isso, há um aumento significativo no número de fontes, aumentando, por conseqüência, a probabilidade de um download mais rápido (Pucha, Andersen et al. 2007a).

Essa abordagem é sustentada por resultados de testes empíricos sobre 1.7 *terabytes* de dados baixados de diversas RCAs onde se constata que:

- A metade dos arquivos MP3 e MPEG com similaridade diferente de zero entre si têm similaridade maior ou igual a 99,6%. Na maioria dos casos, as diferenças estão apenas nos metadados armazenados nos arquivos.

Em seu artigo, Pucha argumenta que a alta similaridade nos arquivos MP3 ocorre em arquivos que possuem o mesmo conteúdo sonoro, porém, com mudanças simples nos metadados. No caso dos arquivos de vídeo, a justificativa de alta similaridade está nos seguintes fatos: (i) filmes e seus respectivos trailers compartilham sequências inteiras de conteúdo audiovisual e (ii) dublagem para diversas línguas de um mesmo vídeo, têm em comum todo o canal de vídeo

O grande trunfo desse trabalho está na criação do método de indexação batizado de *Handprinting*, que permite a localização de segmentos similares em complexidade constante no tempo (*O(1)*). O processo de criação do *handprint* de um arquivo consiste de duas etapas. A primeira é definir uma segmentação que, além de eficiente, seja tolerante a edições nos limites dos segmentos, processo chamado de sincronização (Manber 1994). Dois arquivos estão dessincronizados quando ambos possuem um ou mais segmentos em comum, porém em posições iniciais diferentes. A estratégia utilizada pela grande maioria das RCAs de usar segmentos iniciando na mesma posição, portanto, não é viável para o novo método.

Uma segmentação eficiente e sensível a sincronização é alcançada com o uso do algoritmo de Rabin (Broder 1993). Em linhas gerais, esse algoritmo recebe como entrada *string* de bits (um arquivo, por exemplo) e gera um identificador único resultante da aplicação da função matemática de módulo de um polinômio irredutível a essa entrada. Da mesma maneira que as funções de *hash* citadas no decorrer do capítulo anterior (MD-4 e SHA-1), a aplicação algoritmo de Rabin a diferentes entradas tem altíssima probabilidade de resultar em identificadores distintos.

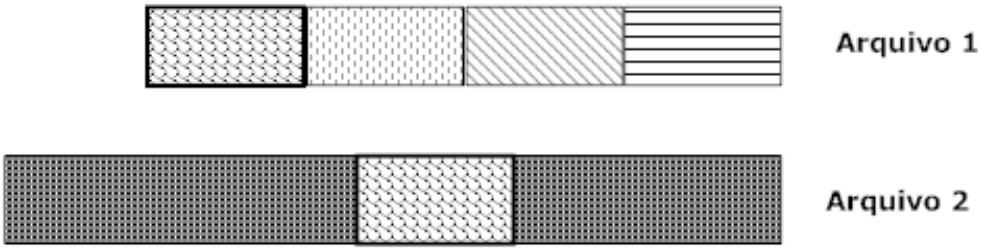

Figura 4 - Exemplo de sincronização usando Rabin para determinação dos limites dos segmentos.

A exemplo de aplicações como sistemas de arquivos distribuídos (Kubiatowicz, Bindel et al. 2000; Muthitacharoen, Chen et al. 2001), no SET, o código Rabin é utilizado para definir os limites de cada segmento, sendo aplicado sucessivamente a uma janela deslizante de 16KB do arquivo até atingir um valor prefixado. A cada um dos segmentos resultantes, é aplicado o algoritmo de *hash* SHA-1 para obter a identificação do segmento (Pucha, Andersen et al. 2007a).

A segunda etapa consiste em colocar os valores dos *hashs* de cada segmento em ordem numérica crescente. O *handprint* é constituído pelos *k* primeiros *hash* de cada arquivo. Quando maior o *handprint*, maior a probabilidade de descoberta de arquivos com segmentos similares, porém com maior número de *hashs* de segmentos a serem indexados pela tabela de índice global. O algoritmo detecta que dois arquivos são similares caso compartilhem pelo menos um dos *k* segmentos (cf. figura 5). Com *k*=30, é possível achar 99% dos arquivos que compartilhem pelo menos 10% de seus segmentos.

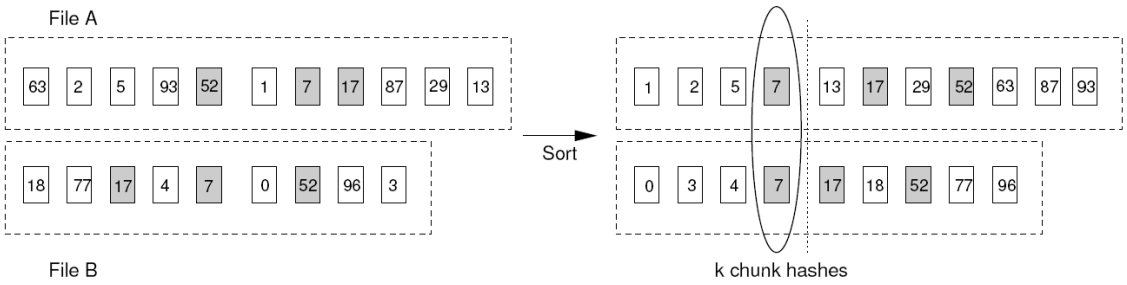

Figura 5 – Ilustração do funcionamento do Handprinting. A ordenação do hash facilita a identificação de segmentos compartilhados. Retirado de (Pucha, Andersen et al. 2007a).

A localização das fontes dos segmentos é mostrada na figura 6. A tabela de busca global contém dois níveis de mapeamentos: o primeiro, que leva o *hash* dos segmentos ao identificador do arquivo (*OID*), e o segundo, que leva o OID ao endereço do(s) *peer(s)* que contém o arquivo.

Uma vez localizadas as fontes de download, é utilizado o processo de download de múltiplas fontes tradicional, com várias semelhanças ao processo usado no *BitTorrent*, dando-se prioridade a segmentos mais raros e atualizando-se a tabela de índice global quando um *peer* adquire um novo segmento.

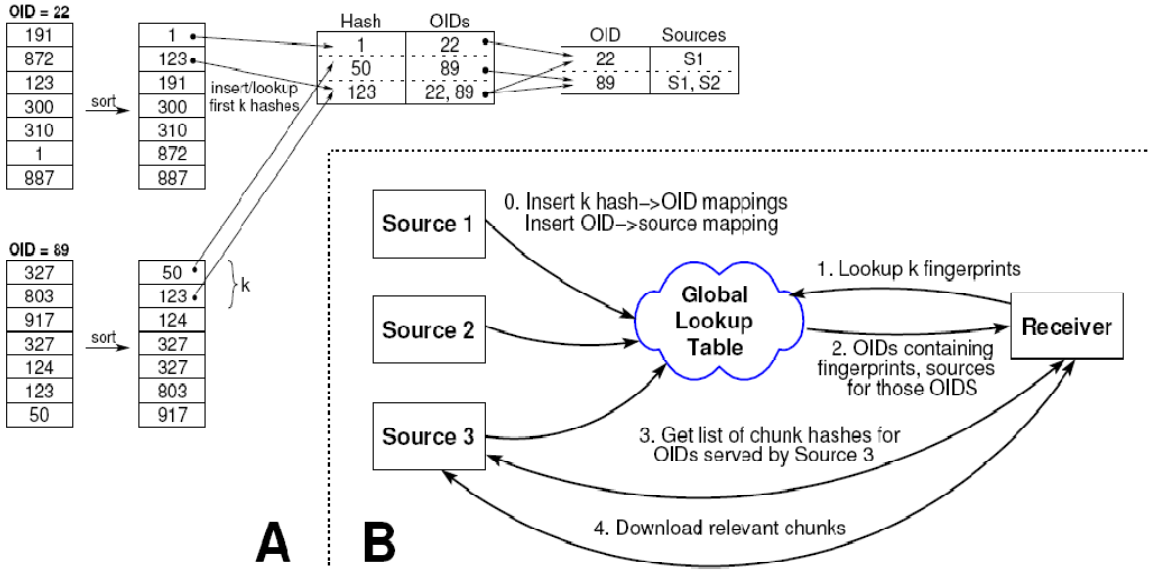

Figura 6 – Funcionamento do SET. Retirado de (Pucha, Andersen et al. 2007a).

Segundo Pucha (Pucha, Andersen et al. 2007a), foram realizados testes de desempenho no simulador *Emulab* (White, Lepreau et al. 2002) em diversos cenários. O objetivo era fazer uma análise de desempenho entre um cliente *BitTorrent* comum e as quatro configurações do SET descritas a seguir:

1. Permite download apenas de arquivos estritamente iguais (SET ou identificador 2, como identificado na Figura 7);

2. Adiciona uma fonte com 10% de similaridade, i.e., uma fonte que compartilha 10% de seus segmentos com o arquivo desejado (SET10 ou identificador 3, idem);
3. Adiciona três fontes com 15% de similaridade (SET15 ou identificador 4, *idem*);
4. Adiciona três fontes com 90% de similaridade (SET90 ou identificador 5, *idem*).

A figura 7 indica o resultado de tempo de transferência de um arquivo de 50MB para todas as configurações em 4 diferentes tipos de conexão de rede. Destacamos o fato de que quanto mais lento o canal de comunicação, maior o beneficio da utilização de um número maior de fontes, resultando em uma diferença de mais de 70% na velocidade download entre *BitTorrent* (1) e SET com 90% de similaridade (5).

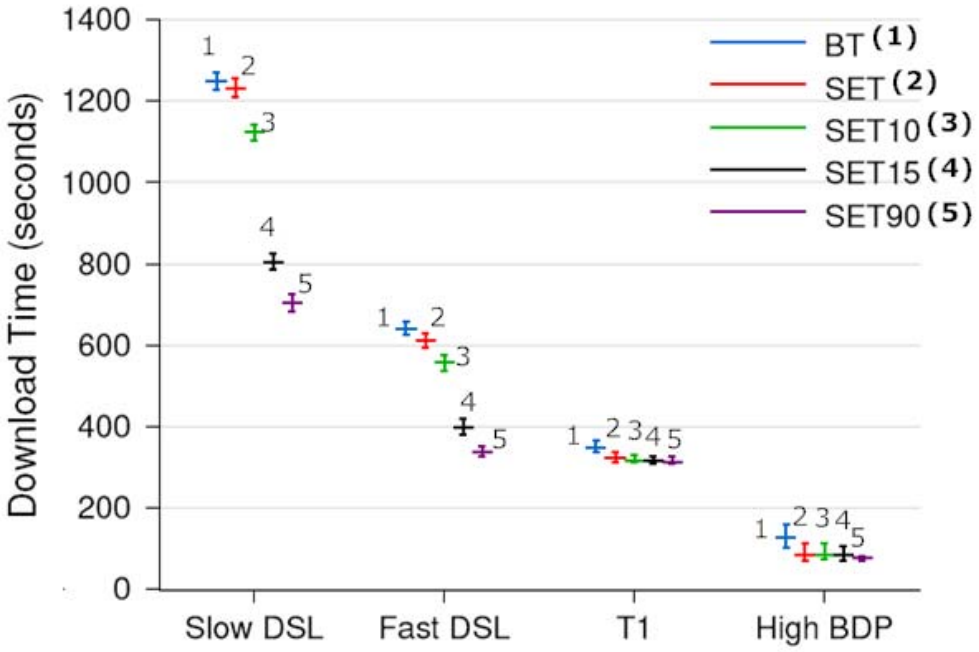

Figura 7 – Resultados experimentais para tempo de download de arquivos de 50MB para várias configurações de SET e BitTorrent comum. (Pucha, Andersen et al. 2007a).

Apesar dos excelentes resultados obtidos, uma crítica a ser feita ao trabalho é que, apesar da comprovação da influência positiva do aumento de fontes para a velocidade

de download, o uso dos *handprints* ainda explora uma parcela reduzida da similaridade entre arquivos. Por exemplo, caso dois arquivos possuam o mesmo segmento com uma defasagem de 1 byte, possivelmente, o processo não identificaria tal semelhança, pois a probabilidade de criação de um segmento 1 byte é extremamente baixa. O trabalho é motivado principalmente por semelhanças exploráveis em arquivos de áudio e vídeo, porém, não leva em consideração características como, por exemplo, formato ou capacidade de adaptação, peculiares desse tipo de conteúdo.

3.2 CAMINHOS PARA MELHORIA DE DOWNLOAD DE ÁUDIO

Na seção anterior, apresentamos o SET, que é considerado o grande avanço em busca de melhorias no processo de transferência em RCAs. Nesta seção, apresentamos algumas técnicas que, embora não tenham sido utilizadas no contexto das RCAs (ou, quando usadas, de forma bastante tímida), tem, em nossa opinião, potencial tanto para o processo de transferência quanto o de busca.

3.2.1 Adaptação de Conteúdo

A natureza, de um lado, heterogênea e, de outro, democrática da Internet sempre fomentou a discussão de como permitir que usuários com recursos distintos possam usufruir das mesmas informações, mesmo que de forma limitada em alguns casos.

Com a sofisticação de dispositivos como, por exemplo, aparelhos celulares e *PDAs* e, consequentemente, os primeiros passos para a chamada computação pervasiva (Satyanarayanan 2001), a preocupação com adaptação de conteúdo aumentou consideravelmente, recebendo a designação de *Universal Media Access* (UMA) (Mohan, Smith et al. 1999). Neste novo cenário, segundo Lei (Lei and Georganas 2001), as adaptações podem ser classificadas de acordo com 4 pontos de vista .

DO CONTEXTO-ALVO

- Adaptação à infra-estrutura técnica: quando a transformação de dados leva em conta apenas fatores relacionados aos recursos, como capacidade de

armazenamento, processamento, velocidade de conexão, capacidade de reprodução de conteúdo do dispositivo, duração da bateria e natureza da conexão (*wi-fi*, *bluetooth*, etc.);

- Adaptação às preferências do usuário: quando o usuário tem diferentes requisitos em relação à apresentação de dados;

DO MOMENTO DA CRIAÇÃO DO CONTEÚDO

- Adaptação estática: quando os dados são pré-processados e armazenados em diferentes versões, de acordo com a qualidade de apresentação ou dados relevantes. A partir do contexto do usuário, a versão mais adequada é disponibilizada ao usuário;
- Adaptação dinâmica: as informações são processadas e entregues sob demanda.

DOS TIPOS DE MÍDIAS

- Adaptação para mesma mídia: quando os tipos de mídia do conteúdo original e conteúdo adaptado são os mesmos. Por exemplo, a transcodificação de uma imagem padrão *bitmap* BMP, para o padrão GIF;
- Adaptação para mídia diferente: Quando um tipo de mídia é transformado em outro mais adequado a um dispositivo particular. Por exemplo, transformação de voz no texto equivalente.

DA COMPLEXIDADE DA ADAPTAÇÃO

- Adaptação física: é definida como a aplicação de um ou mais processos simples como, por exemplo, conversão de dados e filtragens de determinadas informações guiadas pelas características do formato (sintaxe) do conteúdo.
- Adaptação semântica: é um processo de seleção em que componentes específicos do conteúdo têm um significado conhecido e bem definido. A inclusão ou retirada desses componentes influencia na apresentação do conteúdo. Um exemplo de uma adaptação semântica seria o envio, a partir da

mesma música, apenas do acompanhamento para o uso em um karaokê ou apenas da voz para uma aula de acompanhamento.

Para o nosso trabalho, estamos interessados em adaptações que permitam transmissões mais eficientes de áudio em RCAs, que levem em consideração às preferências do usuário e sem o comprometimento severo dos recursos computacionais para esse fim. Nosso interesse, portanto está em adaptações com a seguinte classificação: de acordo com a preferência do usuário, dinâmica, na mesma mídia e física.

3.2.2 Áudio escalável

Um conteúdo de áudio é definido como escalável quando é possível reduzir ou aumentar seu *bitrate*, geralmente em tempo real, de acordo com as restrições de largura de banda do usuário que o deseja consumir (Dunn 2006).

Essa técnica se faz muito presente nos serviços de *streaming* de áudio, pois é uma forma de servir a clientes com características diferentes (maior ou menor largura de banda) sem a necessidade de armazenar o mesmo conteúdo em arquivos codificados em *bitrates* diferentes.

Existem formatos inerentemente escaláveis como, por exemplo, o EAC (Li 2002), o MPEG-4 *GA* e *SSR*[10] (Herre and Grill 2000). Neles, a organização sintática do formato é elaborada para permitir escalabilidade. Como resultado, o processo de redução de *bitrate* é eficiente e os resultados obtidos, de qualidade. Já em outros, como o MP3, a escalabilidade é atingida via um processo não trivial que requer grande conhecimento da estrutura do formato e o do seu processo de codificação (Wang, Huang et al. 2004c).

De forma geral, é possível distinguir dois caminhos para se atingir a redução de *bitrate*: através da re-compressão do conteúdo ou através da eliminação de certas partes do conteúdo que possuem menor relevância.

[10] *GA* (*General Audio*) e *SSR* (*Scalable Sampling Rate*) são modos de compressão do padrão MPEG-4 desenvolvidos para atender a necessidades específicas (Herre and Grill 2000).

O primeiro caminho possui duas técnicas principais: a primeira, que, embora bastante ineficiente, é usada na grande maioria dos servidores de *streaming* consiste na decodificação do conteúdo para o formato de onda sonora seguida de nova compressão com os parâmetros escolhidos especialmente para atingir o *bitrate* desejado. A segunda, menos usada, tenta eliminar possíveis redundâncias deixadas pelo processo de codificação utilizando técnicas mais precisas de compressão de dados. O alvo freqüente desse caminho são os arquivos MP3, cujo método de compressão não retira toda a redundância de dados entre os *frames*. Wang (Wang, Ojanpera et al. 2001b) e Golchin (Golchin and Paliwal 2000) propõem, ambos, o uso combinado da função de predição com uma técnica de codificação sem perdas (codificação diferencial, no primeiro, e codificação aritmética, no segundo). O resultado para ambos é um ganho de codificação de pouco mais de 10%. Apesar do resultado expressivo em termos de compressão, é um resultado pouco útil em termos de escalabilidade.

O segundo caminho, parte da mesma premissa das técnicas de compressão perceptivas: de que, psicoacusticamente, existem algumas informações mais relevantes e outras menos, que podem ser sacrificadas em detrimento da redução do espaço de armazenamento (Painter and Spanias 2000). A retirada das informações pode ocorrer no domínio do tempo ou no domínio da freqüência. No domínio do tempo a técnica mais utilizada é a Eliminação de *frames* (Seo, Kwon et al. 2003; Schwarz, Marpe et al. 2004). Trata-se de retirar uma quantidade de *frames* tal que o *bitrate* desejado seja atingido. Os *frames* a serem retirados são aqueles que possuem maior grau de semelhança com seus vizinhos. O *gap* de tempo que surge com a retirada dos *frames* é eliminado com a adição dos elementos perdidos na máquina do receptor usando técnicas como a repetição do *frame* anterior ou do próximo ou a interpolação do elemento. A necessidade de processamento extra no cliente, muitas vezes inviabiliza a utilização desta técnica.

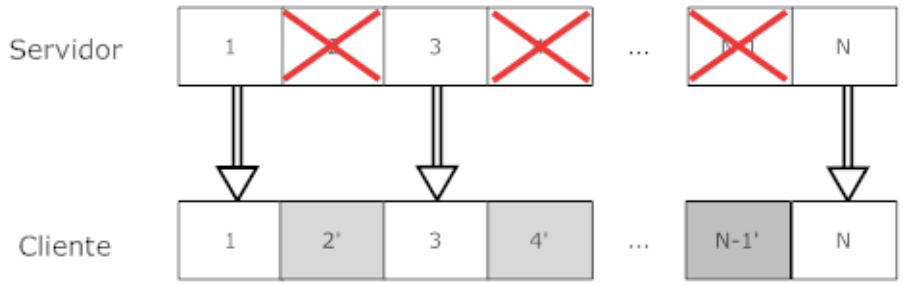

Figura 8 – Técnica de eliminação de frames. Redução de bitrate de 50%. Blocos em cinza representam frames criados no cliente.

No domínio da freqüência, duas são as técnicas mais utilizadas: a retirada de faixas de freqüências inteiras e a eliminação de *bit-planes*. A primeira é baseada na resposta em freqüência do ouvido humano. As freqüências mais baixas têm maior influência na identificação dos sons, enquanto as mais altas são responsáveis pelo aumento de qualidade (Roads 1996). A técnica consiste em organizar as freqüências em uma estrutura de camadas, onde as primeiras, chamadas camadas de base, são compostas pelas freqüências de maior significância perceptiva e, por isso, indispensáveis para identificação do conteúdo. As camadas seguintes, chamadas de camadas de melhorias, podem ter seu envio dispensado, a depender das restrições impostas pelo *bitrate*.

Cada implementação dessa técnica possui seu próprio mapeamento de freqüências em camada (Wang, Huang et al. 2004c; Huang, Wang et al. 2005). Destacamos na Tabela 2 a estrutura sugerida por Wang, composta por duas camadas básicas (BL1 e BL2) e três camadas de melhoria (EL1, EL2 eEL3). As camadas EL1 e EL2, ainda são subdivididas para que seja possível atender a uma maior variação de *bitrates*.

Camadas	**Intervalo de Freq. (Hz)**
BL1	269-1493
BL2	1494-3330
EL1_1	3331-5168
EL1_2	5169-7618

EL2_1	7619-10068
EL2_2	0-268 & 10069-12211
EL3	12212-22050

Tabela 2 – Estrutura de camadas proposta por Wang e suas freqüências correspondentes (Wang et al. 2004).

A técnica de eliminação de *bit-planes* é uma evolução psicoacústica da retirada de faixas de freqüências inteiras. Trata-se não só de considerar a precedência de certas faixas de freqüência, como também a quantidade de bits usada para representá-las. Ao invés das camadas possuírem exclusivamente representantes de uma determinada faixa de freqüência, a proposta é que possuam certa quantidade de bits das outras faixas (Li 2002; Raad, Mertins et al. 2003; Dunn 2006). Pode-se imaginar, portanto, cada camada como "fatias" de bits de freqüências. As camadas básicas (fatias mais importantes) ficam com os bits mais significativos e as camadas de melhoria, com os bits menos significativos, como se pode visualizar na figura 9. Embora nenhum dos trabalhos sobre a eliminação de *bit-planes* defina efetivamente a divisão das camadas, é bastante plausível supor que o número de bits por camada seja definido por alguma função logarítmica, simulando-se, assim, o comportamento do ouvido humano (Roads 1996).

É importante notar que a imensa maioria dos trabalhos apresenta a escalabilidade exclusivamente como redução do *bitrate*. Para o nosso trabalho será importante analisarmos o caso inverso: como partir de *bitrate* menores para maiores. Discutiremos esse ponto no capítulo 4.

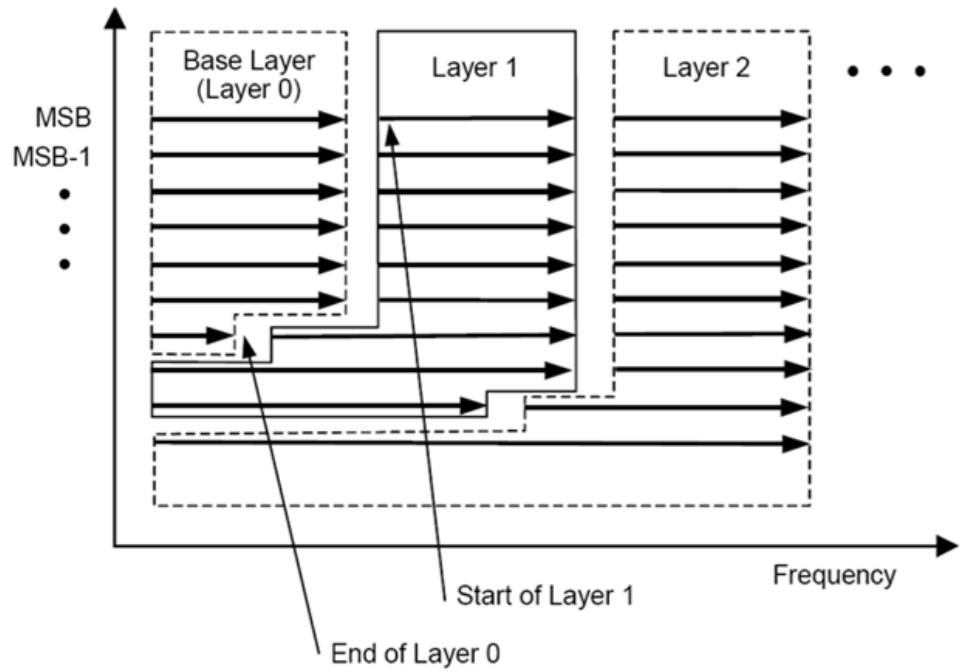

Figura 9 – Ilustração da estrutura de camadas em bit-planes. Retirado de (Dunn 2006).

3.2.3 *Fingerprinting* de áudio

O termo *fingerprinting* de áudio designa o processo de identificar e extrair características acusticamente relevantes de um elemento sonoro que o identifique única e exclusivamente (Cano 2006). Conceitualmente, portanto, não difere das funções de *hash*, como MD4 e SHA-1, comentadas e citadas em seções anteriores. Porém, um processo especial para áudio faz-se necessário porque a representação binária de áudio, tanto em formato de onda, quanto em formato comprimido, diz muito pouco sobre as suas características psicoacústicas. A partir daí, conclui-se que a principal diferença entre o *hash* comum e o *fingerprinting* está no fato de que o primeiro atua na representação estrutural (sintaxe) do objeto, enquanto o segundo, na sua semântica.

Segundo Kalker (Kalker 2001), um sistema de *fingerprinting* deve preencher quatro requisitos principais. A decisão sobre como cada um deles será atendido depende das (será influenciadas pelas) particularidades e propósitos do sistema a ser desenvolvido. Os requisitos são os seguintes:

ACURÁCIA

Estabelece a relação entre o número de identificações corretas, incorretas e não realizadas. Sistemas de alta precisão costumam consumir uma grande quantidade de recursos computacionais.

ROBUSTEZ

A habilidade de identificar um item, independentemente do formato, taxa de compressão ou alguma degradação do sinal, como ruído, por exemplo. É importante notar que algumas aplicações, como os sistemas de verificação de integridade de conteúdo (Gomez, Cano et al. 2002), podem necessitar que o sistema seja sensível o suficiente para perceber e sinalizar modificações mínimas no conteúdo.

ESCALABILIDADE

A capacidade de responder a consultas com eficiência mesmo na presença de uma grande base de confronto. Pode referir-se também a quantidade de consultas realizadas simultaneamente.

COMPLEXIDADE

Refere-se ao custo computacional da extração do *fingerprint*, ao tamanho do *fingerprint*, à complexidade da busca, ao método de comparação dos elementos, ao custo de adicionar um item à base de dados, etc. Também, à semelhança das funções de *hash*, o resultado do processo de *fingerprinting* sobre um determinado conteúdo é um identificador único (*fingerprint*) e pequeno de tal forma que sons diferentes têm probabilidade muito reduzida de possuírem o mesmo *fingerprint*.

Embora existam outras aplicações para o *fingerprinting* como, por exemplo, a já citada verificação de integridade ou o suporte à marcação digital, nesse trabalho será discutido exclusivamente o seu uso para identificação de áudio baseada em conteúdo, certamente sua aplicação mais utilizada.

Nesse tipo de aplicação, o processo de *fingerprinting* é composto por duas partes, como ilustrado na Figura 10. A primeira consiste na criação do *fingerprint* de um sinal de áudio e no seu armazenamento junto aos respectivos metadados. A segunda consiste na recuperação do sinal ou dos metadados a partir do *fingerprint*.

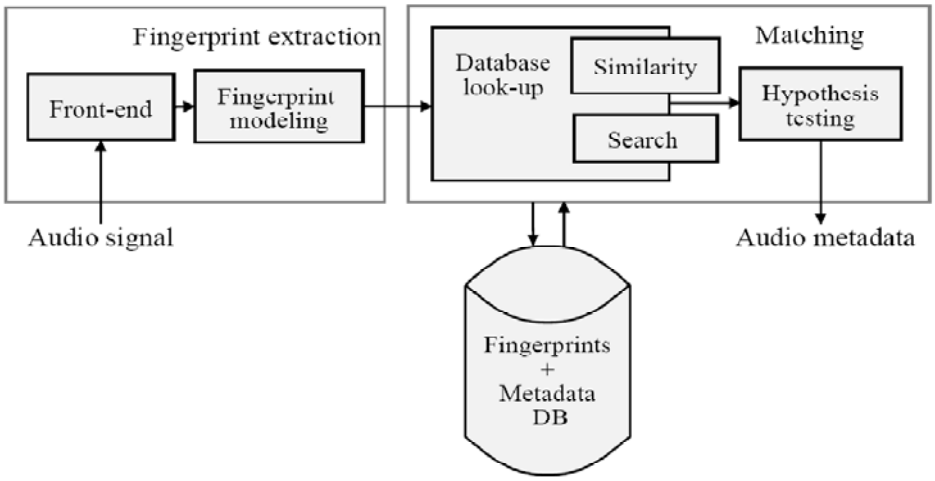

Figura 10 – Identificação de áudio baseada em conteúdo. Retirado de (Cano 2006).

Na figura são ilustradas as etapas do processo de *fingerprinting* que são descritas a seguir.

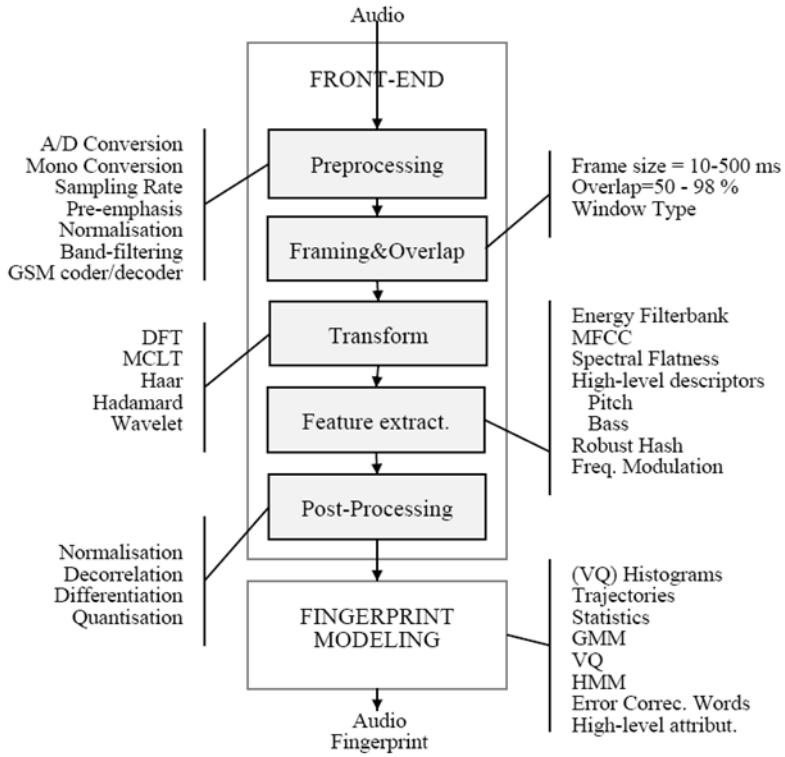

Figura 11 – Etapas do processo de criação do fingerprint. Retirado de (Cano 2006).

PRE-PROCESSAMENTO

Neste passo, há uma normalização da representação do conteúdo a ser usado no processo de *fingerprinting* com o objetivo de evitar distorções causadas por valores em escala diferente, por exemplo. Geralmente o formato adotado é PCM. São normalizados também elementos como a quantidade de bits por amostra, a taxa de amostragem, a amplitude máxima e mínima e a faixa de freqüência de atuação do sinal.

ENQUADRAMENTO E SOBREPOSIÇÃO

A suposição chave na medição de uma característica é que o sinal possa ser analisado de forma estacionaria em um intervalo de tempo pequeno (milisegundos). Para isso o sinal é dividido em *frames* (etapa chamada enquadramento) cujo tamanho está associado à velocidade dos eventos que se deseja observar. Uma janela de sobreposição

é aplicada a cada *frame* para minimizar a descontinuidade nas bordas do frame no processo seguinte. Há um compromisso a ser analisado entre a alta sensibilidade a mudanças, representados por *frames* pequenos e o tempo de processamento nas fases seguintes.

TRANSFORMADA

O objetivo dessa etapa é promover a mudança do ponto de vista de análise dos dados recebidos. A depender das características a serem analisadas, várias funções de transformada podem ser realizadas, as mais utilizadas são as que promovem o sinal do domínio do tempo para o domínio da freqüência, com uso habitual da Transformada Rápida de Fourier (Roads 1996) (sigla em inglês, FFT), da Transformada Discreta de Cosseno (Painter and Spanias 2000) (sigla em inglês, DCT) e dos *Wavelets* (Kudumakis and Sandler 1996).

EXTRAÇÃO DE CARACTERÍSTICAS

Uma vez no domínio da freqüência, transformações adicionais são aplicadas com o objetivo de reduzir a dimensionalidade dos dados, ao retirar informações perceptualmente irrelevantes e, ao mesmo tempo, aumentar a tolerância a distorções no sinal. Grande parte dos sistemas aplica métodos de divisão do espectro em bandas críticas[11] como, por exemplo, escala Bark, *Mel-Frequency Cepstrum Coefficients* (Cano 2006), *Spectral Flatness* (Haitsma, Kalker et al. 2001), etc .

Além desses elementos, são extraídas também características que comprovadamente são úteis na comparação de sons como, por exemplo, *harmonicity*, envoltória e *Zero-crossing rate* (Gouyon, Pachet et al. 2000)

[11] Banda-crítica são pequenas faixas de freqüência organizadas de tal forma que, dois sinais distintos com freqüência dentro da faixa não são percebidos separadamente. Na realidade, o que é percebido é o som anômalo resultante chamado batimento (Roads 1996).

PÓS-PROCESSAMENTO

A maioria das características das fases anteriores são medidas absolutas extraídas do sinal. Para uma melhor caracterização das variações temporais do sinal, a imensa maioria dos sistemas aplica, às características extraídas, derivada no domínio do tempo (Haitsma, Kalker et al. 2001).

É bastante comum, também, a diminuição de dimensionalidade das características com a redução na quantidade de bits usados para representá-las. Ao eliminar os bits menos significativos, ganha-se um pouco de robustez contra pequenas variações do sinal.

MODELAGEM DO *FINGERPRINT*

Essa etapa recebe como entrada uma seqüência de vetores de características calculados para cada *frame*, e tem por objetivo criar uma representação concisa e de fácil recuperação das informações desses vetores. As escolhas estabelecidas aqui influenciarão diretamente o algoritmo de localização e indexação dos *fingerprints*.

Uma forma trivial de se obter um *fingerprint* é simplesmente concatenar as informações dos vetores. O problema dessa abordagem é ainda a grande dimensionalidade, que pode ser proibitiva em alguns ambientes.

Uma outra forma simples é resumir as informações dos vetores condensando-os em agrupamentos que representem partes específicas do sinal, como, por exemplo, conjuntos contíguos de *frames*. Funções como média e variância podem ser utilizadas para a obtenção dos novos valores. O grande problema dessa abordagem é a perda da capacidade de recuperar uma música quando se tem apenas um fragmento dela, já que com essa abordagem perde-se a capacidade de conduzir de forma apropriada o alinhamento do fragmento dentro da música (Cano 2006).

Outra forma bastante comum é utilizar alguma heurística para extrair propriedades específicas dos vetores de características que sejam as mais representativas do sinal ou, contrariamente, eliminar propriedades que não sejam representativas do sinal (Cano, Batlle et al. 2002).

A segunda etapa do processo de identificação de áudio baseada em conteúdo consiste em estabelecer um método de comparar o *fingerprint* extraído do áudio a ser identificado daqueles armazenados na base de *fingerprints* de músicas conhecidas.

Uma abordagem bastante intuitiva é o uso de métricas de distância em espaços multidimensionais (Mitchell 1997). Nesse caso, o áudio em análise é identificado como aquele que possui a menor distância para todos os elementos da base de *fingerprint*, numa área previamente definida. As distâncias mais usadas são a Euclidiana, a de *Manhattan* e a *Hamming* (que mede a distância pela diferença de bits dos dois *fingerprint*).

3.3 DISCUSSÃO

Neste capítulo, apresentamos o SET como o estado da arte na direção a um processo de transferência mais eficiente em RCAs. A principal crítica em relação a esse trabalho é a aparente contradição entre o esforço aplicado na busca de segmentos semelhantes em arquivos diferentes sem, contudo, levar em consideração as características estruturais (principalmente o formato) do conteúdo sendo comparado.

CAPÍTULO 4

FORMATOS DE ÁUDIO COM PERDAS: MP3

A fim de expor com mais facilidade os mecanismos propostos neste trabalho, é importante destacar algumas características dos formatos de áudio com perdas, em especial o formato MP3 que será usado como estudo de caso neste trabalho.

É importante ressaltar, contudo, que a escolha de um formato específico não acarreta em perda de generalidade para o trabalho, pois, embora haja grandes diferenças entre os diversos formatos, principalmente na relação qualidade versus tamanho, apenas três características são fundamentais para que um formato possa ser adaptado à nossa proposta.

A primeira é que possua o conceito de *frame*, que é a representação do sinal em um instante de tempo. A segunda é que haja certa independência de codificação entre os *frames*, em outras palavras, o formato deve poder ser usado em *streaming*. E, por fim, a terceira é que cada *frame* esteja codificado no domínio da frequência.

Por fim, é importante ressaltar que apesar da sua idade e deficiências técnicas, o formato ainda é, de longe, o mais utilizado hoje em dia, como foi dito no capítulo 1. Embora sua morte já tenha sido anunciada por várias vezes, o formato resiste principalmente por causa de sua onipresença na vida das pessoas, seja por meio dos *players*, ou da enorme base legada de músicas.

4.1 INTRODUÇÃO AO MP3

O surgimento do padrão MPEG-1 *Layer* 3, comumente chamado de MP3, definido pela ISO em 1993, foi um divisor de águas na história da música. Embora seja considerado

pouco eficiente comparado aos diversos formatos que o sucederam, ainda é de longe, o formato mais utilizado para compressão de áudio no mundo. Pouco menos de 65% de conteúdo de áudio nas RCAs está em formato MP3 (CacheLogic 2005a).

Como todo padrão de codificação de áudio, apenas a estrutura do arquivo e o processo de decodificação são padronizados. Do codificador, apenas as etapas obrigatórias do processo estão definidas. Isso ocorre para que o processo de codificação possa evoluir livremente, absorvendo avanços técnicos relevantes que venham a ocorrer na área. Um codificador MP3, por exemplo, é definido como um processo que cria um arquivo capaz de ser compreendido pelo decodificador padrão (ISO/IEC 1993).

Da mesma maneira que todos os outros métodos de codificação com perdas, o MP3 vale-se da constatação de que uma parte considerável do sinal produzido não é percebida pelo ouvido humano, num processo chamado mascaramento de sons (Roads 1996).

Como o processo de compressão de MP3 é relativamente complexo (a figura 12 ilustra-o por completo), nos concentraremos a apresentar os fatos relevantes para a compreensão dessa proposta.

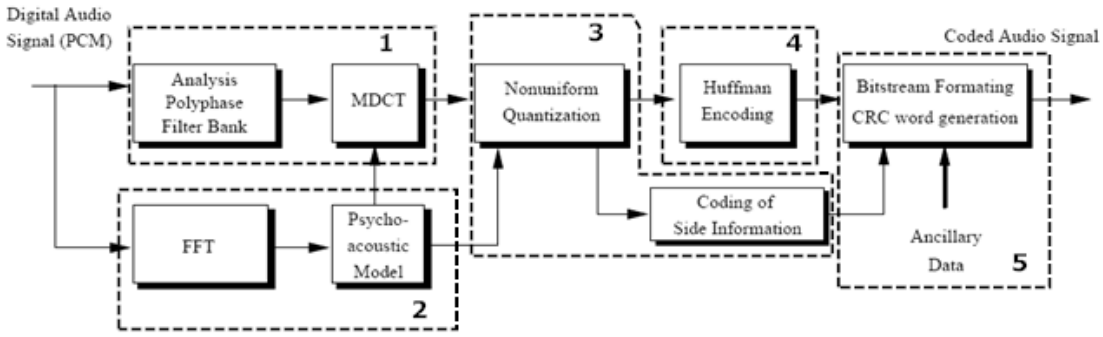

Figura 12 – Etapas (numeradas) e passos do processo de codificação MP3. Retirado de (Salomonsen, Sogaard et al. 1997).

A codificação supõe a entrada de um sinal de áudio em codificação PCM. Esse sinal é divido em partes de 1152 amostras. Cada uma dessas partes será codificada separadamente e resultará em um *frame*. Cada *frame*, portanto, codifica um intervalo de

tempo fixo. Por exemplo, para um sinal PCM com taxa de amostragem igual a 44.1KHz (a mesma usada no CD), cada *frame* MP3 corresponderá a fragmentos de 26,12 milissegundos.

A codificação do *frame* ocorre em cinco etapas: a primeira (número 1, figura 12) consiste em converter as amostras do domínio do tempo para o domínio da freqüência. Ela consiste de dois passos: o primeiro passo consiste de um conjunto de 32 filtros passa-banda de igual largura que separam o conjunto de amostras em 36 sub-amostras para cada uma das 32 faixas de freqüência (*Analysis Polyphase Filter Bank*). Essa etapa tem o objetivo de reduzir a complexidade do passo seguinte, a Transformada Modificada Discreta do Cosseno (MDCT), que efetivamente faz a conversão.

A segunda etapa (número 2, figura 12), acontece paralelamente à primeira. Seu objetivo é sinalizar quais elementos são psicoacusticamente irrelevantes (ou de pouca relevância) no sinal. O primeiro passo consiste na Transformada Rápida de Fourier (FFT), enquanto o segundo, na sinalização dos elementos mascarados (*Psychoacustic Model*).

O primeiro passo (*Nonuniform Quantization*) da terceira etapa (número 3, figura 12), consiste em limitar o número de bits para representar cada faixa de freqüência resultante da etapa 1. Isso é determinado, principalmente, pelas configurações de codificação escolhidas pelo usuário (*bitrate* e número de canais) e pela sinalização recebida da etapa 2. O segundo passo (*Coding of Side Information*) codifica as informações que foram geradas no passo anterior que serão úteis para formatação do arquivo, como por exemplo, o número de bits necessários para representar cada faixa de freqüência, para cada canal.

A quarta etapa consiste em uma compressão sem perdas dos valores das faixas de freqüência quantizadas pela etapa anterior. Trata-se da aplicação da tradicional codificação de *Huffman* (Salomon 2004).

Por fim, na quinta e última etapa, todas as informações importantes são formatadas em um *frame* que pode ainda receber informações externas ao processo de codificação (*ancillary data*).

Figura 13 – Ilustração de um *frame* MP3. Retirado de (Salomonsen, Sogaard et al. 1997).

Na figura 13, vemos a ilustração das informações contidas no *frame* MP3 cujo significado pode ser visto a seguir.

- O cabeçalho contém, além de uma palavra de sincronização, que identifica o seu início, informações sobre o *bitrate* usado na codificação, número de canais, taxa de amostragem do sinal original, indicador da presença do CRC no frame, entre outras. É importante enfatizar a existência de dois campos, ambos com 1 *bit* cada, que não tem influência na decodificação do *frame*. Eles são *copyright*, que sinaliza se o arquivo em questão é protegido por *copyright*, e *original*, que indica se o arquivo é original ou cópia.
- CRC é um campo opcional. Quando existe, armazena o resultado do algoritmo CRC-16 (Salomon 2004), usado para checagem de consistência do *frame*.
- Informação colateral, como já descrita anteriormente, guarda um ponteiro para a posição inicial e o número de bits usados dos dados principais do *frame*.
- Os dados principais são compostos pela informação das faixas de freqüência resultantes da MDCT codificadas utilizando o algoritmo de Huffman;
- Os dados opcionais (*ancillary data*) não fazem parte da codificação MP3. Esse espaço pode ser utilizado, por exemplo, para transportar informações de uma aplicação específica.

Um arquivo MP3, nada mais é que a concatenação de vários *frames* opcionalmente precedidos e/ou sucedidos por uma área de metadados. Na realidade, a área de

metadados não faz parte padrão MP3. Ela foi criada posteriormente, explorando as lacunas existentes nos arquivos. O ID3[12], por exemplo, padrão de fato de edição de metadados, cria sua área de metadados no início (antes do primeiro *frame*) ou no fim (depois do último *frame*) do arquivo, áreas que serão ignoradas pelo decodificador.

[12] http://www.id3.org/

CAPÍTULO 5

MAIS DISPONIBILIDADE E PRECISÃO NA TRANSFERÊNCIA DE ÁUDIO

No capítulo 2, apresentamos o estado da prática das RCAs usando como exemplo as redes *eDonkey* e *BitTorrent* e as expomos suas principais deficiências em prover requisitos importantes como, disponibilidade e confiabilidade.

No capítulo seguinte, apresentamos o SET, trabalho que se propõe a aumentar a eficiência da transferência de dados, criando um novo mecanismo de indexação denominado *handprinting* cujo principal objetivo é viabilizar a descoberta de segmentos iguais em arquivos apenas semelhantes e, assim, permitir um aumento da disponibilidade desses segmentos. Indicamos também, algumas tecnologias com potencial de ajudar a evolução das RCAs.

Neste capítulo, utilizaremos os conceitos descritos até aqui para apresentarmos nossa proposta.

5.1 AUMENTANDO A DISPONIBILIDADE EM ARQUIVOS DE ÁUDIO

Apesar do grande aumento de disponibilidade promovido pelo SET, percebem-se algumas limitações deste trabalho. O principal deles é a impossibilidade de usá-lo em uma RCA atual, sem provocar mudanças significantes nos servidores de busca. Isto porque a procura por fontes similares ocorre por busca aproximada do *handprint* e não

por busca exata. De fato, consideramos essa restrição como um dos maiores problemas na evolução das RCAs.

Como foi dito anteriormente, dada a imprevisibilidade da entrada e da permanência dos *peers* na rede, muito pouco pode ser feito para melhorar a eficiência da transferência diretamente. Protocolos melhores e mais desempenho do canal de comunicação só são úteis quando aquilo que se deseja transmitir está presente na rede.

Estudos demonstram que a única solução efetiva possível visando o aumento da eficiência é aumentar a disponibilidade do conteúdo (Chun, Wu et al. 2006; Pucha, Andersen et al. 2007b).

Em se tratando especificamente de conteúdo fonográfico, constata-se facilmente que tanto as RCAs atuais quanto o SET falham em prover uma descoberta eficiente de fontes similares. Descrevemos, abaixo 5 casos em que arquivos originários do mesmo fonograma podem ser agrupados em uma mesma fonte de transferência.

É importante reiterar que, como estamos lidando com arquivos comprimidos no mesmo formato, MP3, (que sempre codifica 1152 amostras em cada *frame*), arquivos originados do mesmo fonograma (mesma faixa de CD, por exemplo) sempre estão sincronizados.

CASO 1: METADADOS DIFERENTES

Como foi constado pelos autores do SET (Pucha, Andersen et al. 2007a), é bastante frequente encontrar muitos fonogramas codificados exatamente da mesma forma (mesmo formato, mesmo compressor, mesmo *bitrate*), porém, com metadados diferentes. Essas diferenças podem ser ortográficas, como o uso ou não de acentos ou letras maiúsculas nas palavras ou a presença ou ausência de determinadas dados menos conhecidos, como por exemplo, ano de gravação, gravadora, etc.

CASO 2: DIFERENÇAS EM PARTES IRRELEVANTES DO FRAME

O segundo caso é representado pela existência de arquivos MP3 com *frame*s iguais, exceto pelos campos CRC e de dados opcionais (*ancillary data*) que não exercem

qualquer influência nas informações relevantes da música. Da mesma forma, os campos *original* e *copyright*, presentes no cabeçalho dos *frames*, podem diferir sem que haja modificações no conteúdo.

CASO 3: PROCESSOS DE CODIFICAÇÃO DIFERENTES

O caso três decorre do fato de que o processo de compressão de áudio não é padronizado, o que resulta no fato de codificadores diferentes (porém do mesmo formato) gerarem arquivos com conteúdo e/ou tamanho diferentes, como pode ser visto na Tabela 3, porém com muita semelhança perceptiva.

Bitrate (kbps)	Codec.	Tamanho (bytes)	MD5 (hexadecimal)
128	*Lame*	3.353.704	16572038CB6F550B13A91C47D9D1806E
	WMP	3.353.704	93DA6195110EA5A9106F49D394A556CE
	iTunes	3.354.123	25FC6366187CC9E00C6CF4F2E98116A1
160	*Lame*	4.192.131	32219331678A42EAE73E27422F5FDF33
	WMP	4.192.131	C783A707E76F3775CEC36B593FE0A61D
	iTunes	4.192.653	4AD418AA627D34DF9DBC4738298C613B
192	*Lame*	5.030.556	5416507D8970DAD23CE1C19182AE7BEE
	WMP	5.030.557	86E8389A1069A4480AD428AC33446C18
	iTunes	5.031.184	7DD14DD09BCF74C07B27826A1657A000

Tabela 3 – Resultado da codificação MP3 da mesma música em diferentes codecs para três *bitrates* diferentes.

CASO 4: *BITRATES* DIFERENTES

Arquivos codificados utilizando *bitrates* diferentes, também têm muito em comum, principalmente nas faixas de frequência mais baixas (Wang, Huang et al. 2004a).

CASO 5: FORMATOS DIFERENTES

Codificadores diferentes assumem diferentes pressupostos no processo de codificação, porém a semelhança perceptiva indica que há o que explorar mesmo nessas situações. Não abordaremos esse caso por se tratar de um procedimento que, caso seja factível, demandará muito poder de processamento, principalmente porque há poucas chances de que os arquivos estejam sincronizados.

Neste trabalho, propomos um método de explorar os quatro primeiros casos de subutilização de fontes descritos nos parágrafos anteriores. Este método baseia-se no fato de que o conhecimento sobre o formato do arquivo e o método de compressão utilizado pode ser empregado para encontrar elementos semelhantes com facilidade, bem como desprezar informações irrelevantes quando adequado. Para tanto, são necessárias novas técnicas de segmentação e indexação específicas para conteúdo fonográfico.

5.2 INDEXAÇÃO VISANDO AUMENTO DA DISPONIBILIDADE DOS ARQUIVOS DE AUDIO

A técnica tradicionalmente utilizada para a indexação de arquivos e seus segmentos, embora inegavelmente útil e eficiente, não é capaz de tirar proveito de todas as potenciais fontes disponíveis quando se trata domínio de áudio. O *Handprint* (indexação) utilizado no SET, que permite o encontro de segmentos iguais em arquivos diferentes com alta probabilidade, é útil para resolver o primeiro caso (metadados diferentes), mas não é capaz de lidar com os outros casos.

Nossa proposta de indexação é composta por três tipos de índices distintos: o Índice Bit-a-Bit, o Índice Sensível ao Conteúdo e o Índice de *Frankenstein*. O primeiro, que tem o mesmo propósito que os índices encontrados nas RCAs, é o resultado da aplicação de um algoritmo de *hash* SHA-1 em todo o conteúdo do segmento, esse índice serve para identificar exclusivamente um arquivo, quando necessário, bem como

permitir a verificação de integridade após a transmissão. Os dois índices restantes serão discutidos a seguir.

5.2.1 Índice Sensível ao Conteúdo

Este índice foi criado para contemplar arquivos na situação descrita nos casos 1 e 2.

O índice é criado como resultado da aplicação da função de *hash* SHA-1 em todo arquivo excetuando: metadados, campos CRC, *ancillary data* e campos *copyright* e *original* presentes no cabeçalho de cada *frame*, como pode ser visto na Figura 14.

Esse método seria capaz de identificar as semelhanças contidas nos casos 1 (metadados diferentes) e 2 (CRC e dados opcionais diferentes) descritos anteriormente. Por exemplo, seria possível identificar que dois arquivos são semelhantes mesmo que tenham metadados, CRCs e dados opcionais diferentes.

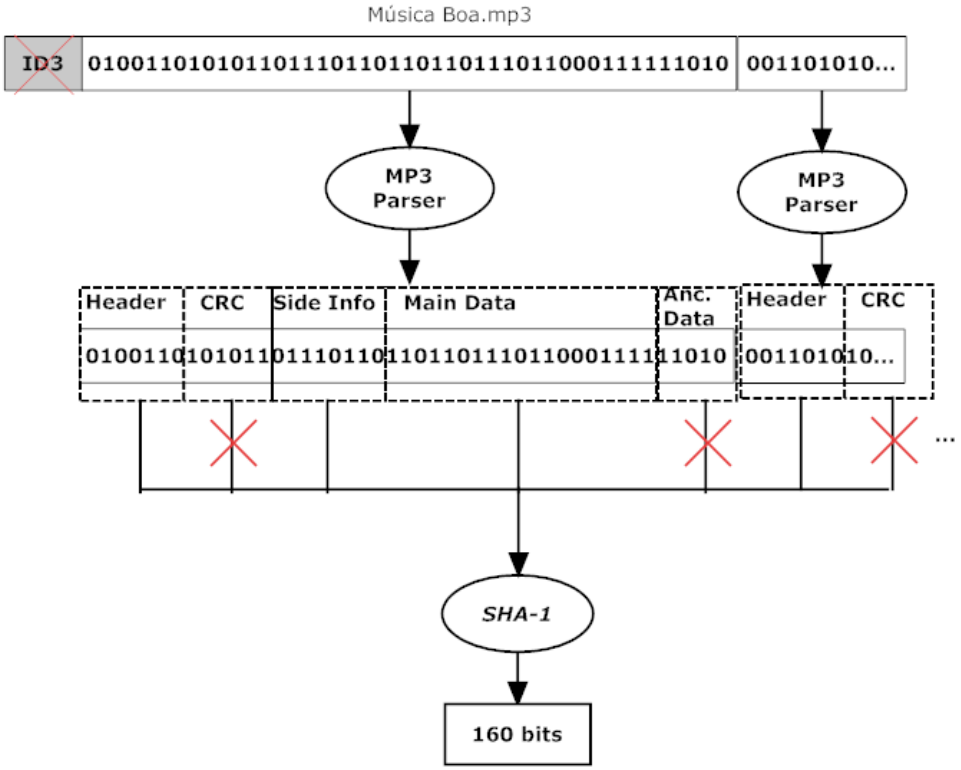

Figura 14 - Técnica de indexação sensível ao conteúdo. Metadados e alguns campos dos *frames* são ignorados.

Para viabilizar essa solução, se faz necessária a criação de um *parser* MP3 capaz de identificar a área de metadados, bem como todos os campos de um *frame* MP3.

5.2.2 Índice de Frankenstein

O objetivo de Índice de Frankenstein (a partir daqui, IF) é permitir a identificação da semelhança em fonogramas na situação descrita nos casos 3 e 4, em que a comparação de representações binárias é irremediavelmente falha. A similaridade para esses casos está no sinal propriamente dito.

Com a utilização dessa indexação, surge a possibilidade de juntar segmentos de origens distintas a fim de criar um novo arquivo. Ela recebe o nome de Frankenstein em homenagem a personagem do médico que, analogamente, criou um novo ser a partir de partes distintas de vários cadáveres no famoso romance de Mary Shelley (Shelley 1994).

Pretendia-se, a princípio, que o IF implementasse técnicas similares àquelas encontradas nas ferramentas de *audio fingerprinting*, mais precisamente utilizar características que pudessem ser extraídas diretamente do MP3, ou domínio comprimido, segundo o jargão da área (Tzanetakis and Cook 2000) e, com isso, obter bom desempenho de (Doets and Lagendijk 2005).

Para uma melhor compreensão do processo de concepção desse índice, discutiremos os requisitos com base nos passos necessários para a construção de um sistema de *fingerprinting* como estabelecido no capítulo 3 (observar seção 3.2.3 e Figura 11).

Como nosso objetivo consiste em apenas um subconjunto daquilo a que se propõe um sistema de *fingerprinting* (queremos encontrar apenas os fonogramas similares e com *frames* sincronizados), as etapas de pré-processamento e de enquadramento e sobreposição não são necessárias, já que estamos procurando arquivos com a mesma origem (o CD) e por isso, mesma taxa de amostragem e resolução e, reiterando, sincronizados no tempo.

A etapa de transformada também pode ser ignorada, visto que os arquivos MP3 já estão no domínio da frequencia.

A etapa seguinte é a extração de características. Grande parte das características sugeridas nos artigos sobre o tema foi testada, e todas se mostraram muito pouco eficazes em promover uma indexação de qualidade. As características extraídas foram:

- Energia[13], por subbanda de frequência e suas variantes, como por exemplo, o quadrado da energia e raiz quadrada da média quadrada ou, em inglês, *root mean square- RMS* (Tzanetakis and Cook 2000);
- Energia por *frame* e suas variantes (Cano 2006);
- Variação da energia por banda de frequência; (Doets and Lagendijk 2005)
- Variação da energia por *frame* (Cano 2006);
- *Spectral Flatness Measure* (Cano 2006);
- Algoritmo de *fingerprinting* proposto pela Philips (Haitsma, Kalker et al. 2001).

Na etapa de pós-processamento, os valores obtidos tiveram seus valores arredondados e/ou truncados a fim de reduzir a dimensionalidade. Nenhuma das características extraídas conseguiu resultados satisfatórios (gerar índices iguais para arquivos na situação descrita nos casos 3 e 4).

Foi necessário, então, abdicar da velocidade de processamento na extração das características no domínio comprimido em detrimento da procura por características mais precisas obtidas a partir da codificação do arquivo para o domínio do tempo.

No domínio do tempo, a característica que se mostrou mais confiável, e que foi efetivamente utilizada neste trabalho, foi a detecção do andamento da música utilizando um algoritmo *BeatRoot* (Dixon 2007). O processo de indexação acontece da seguinte forma: um segmento qualquer de um arquivo MP3 é descomprimido, gerando seu

[13] A Energia é calculada como a soma das intensidades das frequências. A energia pode ser agrupada em faixas de frequencia. Dessa forma a energia de um *frame* é a soma das intensidades de todas as freqüências nele contidas (Wang, Divakaran et al. 2003).

correspondente em formato WAV. Como pode ser visto na Figura 15, o arquivo descomprimido é passado como parâmetro para o algoritmo *BeatRoot*, que retorna uma lista contendo o andamento (lista de batidas[14]) daquele segmento. A lista então é analisada, obtendo-se o andamento mais frequente. Esse valor é processado pela função de *hash* SHA-1, criando-se um índice de 160 *bits*.

Da mesma forma que os demais indexadores, o índice do arquivo completo é o resultado da aplicação da mesma função SHA-1 à concatenação dos índices dos seus segmentos.

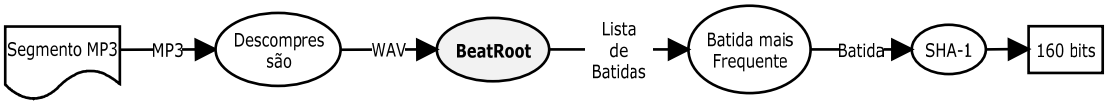

Figura 15 - Descrição do processo de Indexação de Frankenstein.

Apesar de necessitar de muito poder de processamento, a IF conseguiu, nos testes, aumentar a disponibilidade significantemente em alguns casos. Os resultados serão detalhados no capítulo posterior.

5.3 SEGMENTAÇÃO VISANDO AUMENTO DA DISPONIBILIDADE DE ÁUDIO

A indexação proposta na seção 1.2, visa identificar, como iguais, arquivos que possuem certas diferenças entre si. Se aliado a isso, utilizássemos a segmentação usualmente encontrada nas RCAs atuais, isto é, segmentação baseada em tamanho do arquivo, fatalmente criaríamos arquivos MP3 malformados. Há, portanto, a necessidade de se criar um novo método de segmentação para viabilizar a indexação proposta.

[14] A batida é cada unidade perceptiva do ritmo. O intervalo entre duas batidas consecutivas é chamado andamento.

5.3.1 Segmentação em intervalo de tempo (SIT)

Como foi visto no capítulo anterior, o MP3, assim como todo formato de codificação de áudio é composto por uma sequência de *frames* que armazenam ordenadamente pequenas frações iguais de tempo (correspondentes a 1152 amostras) do fonograma original.

Portanto, todos os arquivos MP3 codificados a partir do mesmo fonograma, terão a mesma quantidade de *frames* independentemente dos parâmetros de codificação usados.

Cada *frame* representa, portanto, um intervalo de tempo determinado definido no projeto do codificador (um *frame* MP3, por exemplo, possui 26,12 ms, para codificação de áudio em qualidade de CD).

Ao tentarmos concatenar arquivos MP3 distintos em pedaços baseados em tamanho fixo, como *eDonkey* e *BitTorrent* , ou tamanhos mutável mas sem ligação com a estrutura do arquivo isso implicará arquivos mal formados

Se usarmos como critério de segmentação, ao invés de tamanho em *bytes*, intervalos de tempo, ou números inteiros de frames, garantimos que dois segmentos podem ser concatenados resultando em um arquivo MP3 bem formado como pode ser visto na Figura 16.

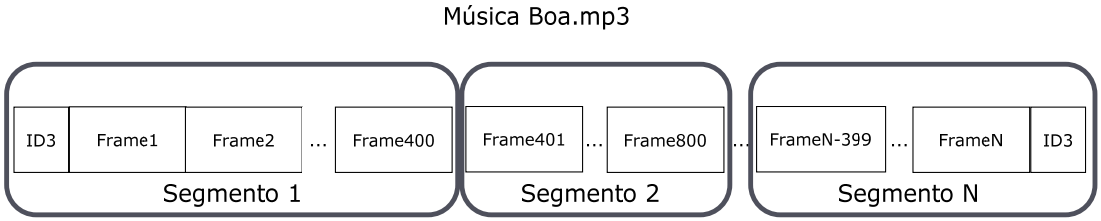

Figura 16 - Segmentação em Intervalo de Tempo (400 frames).

A segmentação por tempo do MP3 deve conter obrigatoriamente um número inteiro de frames calculado para manter um tamanho em bytes semelhante ao dos segmentos usados na rede *BitTorrent*. Escolhemos a segmentação a cada 400 *frames* (aproximadamente 10,4 segundos) cujo tamanho, em KB, vemos na Tabela 4.

Número de *frames*	Tempo (em segundos)	*Bitrate* (kbps)	Tam. Aprox. (KB)
100	2,6	128	41,6
		160	52
		192	62,4
400	10,4	128	166,4
		160	208
		192	249,6
1149	30	128	480
		160	600
		192	720

Tabela 4 – Tamanho aproximado, em KB, para 3 casos distintos.

A grande vantagem desse tipo de segmentação é a capacidade de promover a concatenação de segmentos com *bitrate* diferentes em um mesmo processo de download preservando a consistência do conteúdo baixado, uma vez que segmentos consecutivos podem ser perfeitamente encadeados independentemente do seu *bitrate*.

5.4 AUMENTANDO A EFICÁCIA DA TRANSFERÊNCIA

Apesar de sua importância, a eficácia é um requisito pouco discutido na literatura. Em linhas gerais, uma RCA eficaz teria que obedecer a duas regras: a primeira é evitar o download de arquivos indesejados e a segunda consiste em dar mais controle de decisão ao usuário sobre o que deve ser baixado.

Na realidade, o download de arquivos indesejados é conseqüência da falta de outro requisito fundamental: a confiabilidade. As propostas de possíveis soluções passam inevitavelmente pela tentativa de determinar se o *peer* que está fornecendo o conteúdo é

confiável ou não (Christin, Weigend et al. 2005), algo que nem sempre é possível estabelecer (como classificar um *peer* que nunca entrou na rede antes?).

Nossa proposta para o aumento de eficácia consiste em uma nova técnica de segmentação, chamada de Segmentação por Significância Perceptiva, que permite ao usuário decidir em que nível de significância perceptiva ele deseja baixar um determinado arquivo de áudio. Ao escolher baixar apenas o segmento de maior significância (que é consideravelmente menor que o arquivo completo), o usuário pode detectar possíveis downloads indesejados mais cedo, economizando, assim, tempo e largura de banda. Na seção seguinte, discutiremos a segmentação em mais detalhes.

5.4.1 Segmentação por Significância Perceptiva (SSP)

Segmentação por Significância Perceptiva (SSP) utiliza os conceitos de escalabilidade de áudio para aumentar a eficácia na transferência de segmentos. Dessa forma, o usuário assume o controle do compromisso qualidade *versus* velocidade de download.

Implementamos a técnica de retirada de faixas de freqüência, com a estrutura de sete camadas de qualidade proposta por Wang (Wang, Huang et al. 2004c), descrita no capítulo 3. Um dos benefícios de utilizar essa técnica é poder atuar diretamente nos arquivos MP3, sem a necessidade de decodificação.

A segmentação foi estabelecida visando evitar segmentos muito pequenos e que alguns segmentos fossem muito menores ou maiores que os outros. Baseando-se nessa premissa, foram estabelecidos 4 segmentos:

- No segmento 1 são transportados os dados referentes a metadados do arquivo, cabeçalhos, CRCs, *side information* e as frequências relativas à camada BL1. Esse segmento tem, em média, 16% do tamanho total do arquivo;
- No segmento 2 são transportadas as freqüências relativas às camadas BL2, EL1_1 e EL1_2, Esse segmento possui, em média, 10% do tamanho total do arquivo;

- No segmento 3 são transportadas as freqüências relativas às camadas EL2_1 e EL2_2, esse é o maior segmento e corresponde a 62%, em média, do tamanho total;
- No segmento 4 são transportadas as freqüências relativas à camada EL3 e os *ancillary data*, o que corresponde a, em média, 13% do total.

O primeiro segmento possui todas as informações necessárias para que o arquivo seja decodificado sem erros, bem como as freqüências de maior significância perceptiva. Cada segmento restante corresponde ao incremento de qualidade na percepção do som produzido no primeiro.

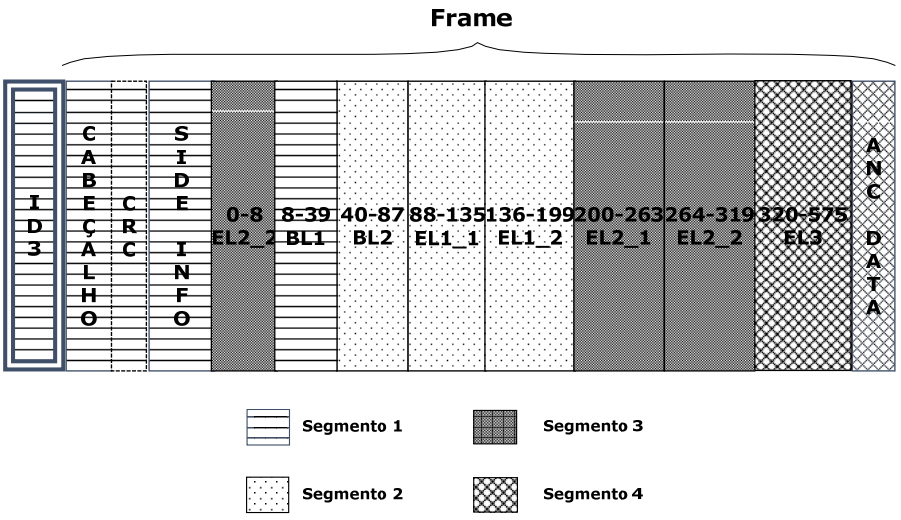

Figura 17 - Segmentação por Significância Perceptiva. Cada padrão de preenchimento corresponde a um segmento.

É importante notar que nenhum segmento gerado pela SSP é um arquivo MP3 bem formado. É necessário, no software do cliente, a implementação de um procedimento para transformar o(s) segmento(s) em arquivo MP3. O software se comporta da seguinte forma:

- Caso seja segmento 1, para cada *frame*, adiciona-se bits de valor zero nas frequências ausentes/

- Caso seja outro segmento com o segmento 1 presente, para cada *frame*, substituem-se as frequências com bits de valor zero pelos valores corretos trazidos pelo segmento;
- Caso seja outro segmento com o segmento 1 ausente, simplesmente, armazena-se o segmento e aguarda-se a chegada do segmento 1.

Vale salientar ainda 2 pontos importantes. O primeiro é que para a que se propõe a SSP, a indexação tem aqui o papel de verificação de integridade e identificação única do arquivo, portanto, propomos o emprego do mesmo Índice Bit-a-Bit em cada segmento. O índice do arquivo todo, como de costume, é o resultado da aplicação do SHA-1 na concatenação nos índices de todos os segmentos do arquivo.

O segundo ponto é que como a SSP atua separando os bits dentro de cada *frame*, é esperado que tenhamos um *overhead* do tamanho dos segmentos em relação ao tamanho do arquivo segmentado. Discutiremos esse custo no capítulo 6 (seção 6.3.1).

5.5 ADAPTANDO E-DONKEY E BITTORRENT

Os mecanismos propostos neste trabalho concentram-se exclusivamente no software utilizado no *peer* e, por esse motivo, é possível adicionar tais recursos tanto à rede *BitTorrent* quanto à rede *eDonkey*, e assim, fazer uso de seus protocolos e estratégias de seleção de fontes.

Os novos mecanismos podem também funcionar como uma espécie de *plug-in* numa RCA. Nas seções seguintes, apresentaremos como seriam a comunicação entre *peer* e servidor nas redes *BitTorrent* e *eDonkey*.

5.5.1 Comunicação PEER-SERVIDOR no eDonkey

A maneira mais simples é enviar para o servidor as informações do arquivo e seus respectivos segmentos para cada par segmentação-indexação. Para a proposta apresentada, isso significaria enviar, para cada arquivo na base de compartilhamento: Segmentação em Intervalo de Tempo (SIT) com índice Bit-a-Bit, SIT com Índice

Sensível ao Conteúdo, SIT com Índice de Frankenstein e SSP com Índice Bit-a-Bit. A indicação de qual par segmentação-indexação se trata, poderia ser incluída em um dos metadados disponíveis ou até mesmo no nome do arquivo.

5.5.2 Comunicação PEER-SERVIDOR no BitTorrent

Da mesma forma como ocorre no *eDonkey*, a maneira mais simples seria criar um arquivo *.torrent* diferente para cada par segmentação-indexação. Como o arquivo *.torrent* é mais flexível quanto aos metadados que podem ser colocados, é possível indicar claramente o par segmentação-indexação utilizado.

Além de não exigir adaptação por parte das RCAs existentes, os mecanismos propostos podem funcionar como uma espécie de plug-in de uma rede já existente. Quando um *peer* tradicional inicia uma comunicação, o seu interlocutor responde seguindo o protocolo tradicional, porém, quando um *peer* dotado das novas indexações e segmentações, estabelece-se o modo avançado de comunicação (Figura 18).

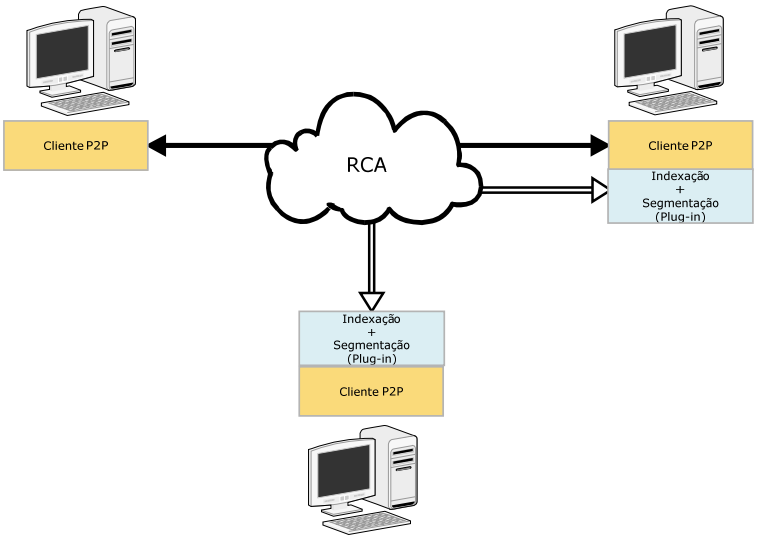

Figura 18 - Ilustração da Adaptação para as RCAs existentes. Indexação e Segmentação como plug-ins.

5.6 INTEGRANDO AS TÉCNICAS

O funcionamento da nova RCA resultante da incorporação desses mecanismos propostos é praticamente igual ao da rede subjacente, com exceção do fato de que na primeira, o *peer* tem mais controle em decidir qual a estratégia de download que melhor se adapta às suas necessidades (escolhas pessoais quanto à qualidade do áudio e eficácia da transferência e quanto à disponibilidade versus grau de miscelânea do arquivo). Para facilitar a exposição de alguns cenários de utilização, imagine que Jó tem a sua disposição uma rede onde ele pode manipular os seguintes parâmetros:

- Número de fontes: pode receber três valores, **baixo**, que agrupa as fontes pelo método utilizado nas redes convencionais (*hash* tradicional), **normal**, que agrupa as fontes utilizando o ISC e **alto**, que agrupa as fontes utilizando o IF;
- Limites de *bitrate*: determina qual o limite máximo e mínimo de bitrate das fontes quando o número de fontes estiver definido como alto;
- O grau de significância perceptiva da música: determina qual ou quais os segmentos da SSP. O nível mínimo corresponde ao primeiro segmento, enquanto o nível máximo corresponde ao arquivo completo;

Nas seções seguintes apresentaremos cenários de download e como as técnicas propostas influenciam nesses cenários.

5.6.1 Cenário 1: download clássico

Suponha para este cenário que Jó deseja baixar uma música previamente conhecida (chamada ***Aquela***), na mais alta qualidade possível. Ele define os parâmetros com os seguintes valores:

- Número de fontes: **baixo**
- Limites de *bitrate*: **desabilitado (não importa neste caso)**;

Após fazer a busca, aparecem nos resultados 6 ocorrências da música organizadas da seguinte forma:

Nome	*Bitrate* (kbps)	Codificador	Fontes	Metadados
Aquela	128	A	2	Valor1
Aquela	128	B	1	Valor1
Aquela	160	A	3	Valor2
Aquela	192	A	1	Valor2
Aquela	192	A	2	Valor3
Aquela	192	A	2	Valor 4

Tabela 5 – Resultado da busca. Número de fontes=baixo.

Jó percebe que há ocorrências da música em alta qualidade, porém com metadados diferente. Ele, então, modifica o parâmetro número de fontes para **normal**. Os resultados são reagrupados:

Nome	*Bitrate* (kbps)	Codificador	Fontes
Aquela	128	A	2
Aquela	128	B	1
Aquela	160	A	3
Aquela	192	A	5

Tabela 6 – Resultado da busca. Número de fontes=normal.

Com o número de fontes maior que anteriormente, o usuário decide iniciar o processo de download da música a 192kbps.

5.6.2 Cenário 2: música desconhecida

Neste segundo cenário, Jó está interessado em experimentar uma nova música que lhe foi recomendada, cujo nome é ***Pirata? Eu não!***. Ele decide então experimentar a

música antes de baixá-la em qualidade alta e configura os parâmetros da seguinte forma:

- O grau de significância: **1**;

Após fazer a busca, aparecem nos resultados 1 ocorrência da música organizada da seguinte forma:

Nome	***Bitrate* (kbps)**	**Fontes**
Pirata? Eu não!.	128, 160	6

Tabela 7 – Resultado da busca. Número de fontes=alto, limite de *bitrate*=128-160.

Ao fim do processo de transferência, depois de aprovar a música, Jó decide incrementar a qualidade do que já foi baixado, modificando o parâmetro grau de significância para máximo (5). O sistema automaticamente iniciará o processo de download dos segmentos de qualidade restantes utilizando como fonte o *bitrate* de cada segmento.

5.7 DISCUSSÃO

No próximo capítulo apresentaremos os resultados dos testes realizados e o desempenho dos pares indexação-segmentação.

CAPÍTULO 6

AVALIAÇÃO E RESULTADOS

Neste capítulo, testaremos os mecanismos de indexação e segmentação discutidos até agora. O nosso objetivo é descobrir em que medida tais mecanismos são viáveis do ponto de vista de desempenho e se de fato, são capazes de promover eficácia e aumentar disponibilidade dos recursos, como foi defendido até aqui.

É preciso reiterar que a avaliação feita aqui não envolverá em momento algum, testes em redes *peer-to-peer* reais ou simulação de tais redes, uma vez que o este trabalho, não trata de *como* ocorre o transporte de dados e sim, de *o que* é transportado.

6.1 METODOLOGIA DE TESTES

Serão testados neste capítulo todos os métodos de segmentação e indexação desenvolvidos: são eles a Segmentação em Intervalo de Tempo (SIT), a Segmentação por Significância Perceptiva (SSP), a Indexação Bit-a-Bit (BB), a Indexação Sensível ao Conteúdo(ISC) e a Indexação de Frankenstein (IF).

Além deles, mais dois pares de indexação e segmentação serão utilizados para integrar o ambiente de teste: o *eDonkey*, como representante das RCAs atuais e o SET, como representante do estado da arte na área.

Todas as técnicas de indexação e segmentação envolvidas no teste foram implementadas em Java, utilizando sempre que possível as mesmas interfaces, classes e métodos para evitar que alguma delas obtivesse qualquer tipo de vantagem competitiva.

Os testes ocorreram da seguinte forma: cada solução de segmentação e indexação é executada 10 vezes e intercaladamente para todos os arquivos da base de testes, criando uma base de compartilhamento (explicada na seção 6.2.1), durante o teste o tempo é cronometrado.

O testes foram efetuados numa máquina Pentium IV 2.8Mhz com 768Mb de memória RAM, com sistema operacional Ubuntu Linux 8.04. Os código implementado em Java foram executados em Plataforma Eclipse 3.4

6.1.1 Base de Testes

Para a avaliação experimental, optamos pela criação de uma base de fonogramas constituída a partir de amostras coletadas de duas origens distintas: de arquivos disponíveis em RCAs (mais especificamente *eDonkey*) e da conversão de CDs para o formato MP3. Com isso, abrangemos tanto exemplos capazes de refletir a realidade do conteúdo encontrado nas RCAs em aspectos importantes como, por exemplo, diversidade, tamanho, *bitrates* e metadados, quantos outros capazes de promover o teste integral dos mecanismos desenvolvidos.

Os arquivos copiados das RCAs foram selecionados da seguinte forma: foram requisitados *todos* os fonogramas listados nas respostas (e, portanto, considerados distintos) de sete consultas (ver Tabela 8) elaboradas com o intuito de encontrar músicas específicas.

As quatro primeiras foram escolhidas por representarem o comportamento de músicas que podem ser encontradas nas RCAs, mas com frequencia baixa ou mediana, enquanto as outras foram os três *singles* mais ouvidos na primeira semana de 2009, segundo a revista Billboard[15]. Representam o comportamento das músicas com grande quantidade de fontes.

[15] http://www.billboard.com/bbcom/index.jsp

O tempo de espera para que todos os fonogramas fossem baixados foi de cinco meses para as quatro primeiras consultas e dois meses, as restantes. Apesar disso, o número de *downloads* não concluídos foi de quase 20% do total de respostas.

String de Consulta	Fonogramas	Tamanho (MB)
`as time goes by chet baker mp3`	8	48,5
`autumn leaves cannonball feat mp3`	21	243
`jackson do pandeiro sebastiana mp3`	16	42.3
`perlman itzhak concerto vivaldi mp3`	7	114
`my life would suck without you mp3`	34	196
`lady gaga featuring colby mp3`	13	101
`single ladies mp3`	125	810
Total	224	1550

Tabela 8 - Fonogramas baixados da RCA *eDonkey*.

Amostras da segunda origem foram geradas a partir de 22 CDs de diversos estilos escolhidos aleatoriamente do acervo do autor da tese, totalizando 314 faixas. Cada uma delas foi codificada de 13 formas diferentes. Foram utilizados, para isso, os três codificadores mais populares do mercado: o codificador presente no *Windows Media Player* (WMP)[16], que acompanha os sistemas operacionais *Windows*, o codificador presente no *iTunes*[17], da *Apple*, o mais famoso gerenciador de acervo de música atualmente e o *Lame*[18], codificador de código aberto que está presente na imensa maioria dos programas gratuitos.

[16] http://www.microsoft.com/windows/windowsmedia/br/

[17] http://www.apple.com/br/itunes/

[18] http://lame.sourceforge.net/

Para cada codificador (e para cada fonograma) foram gerados arquivos MP3 com parâmetros distintos de *bitrate*, três de *bitrate* constante (CBR) com taxas de 128, 160 e 192 kbps e dois de *bitrate* variável (VBR) com qualidade intermediária e alta (exceto o WMP, que não é capaz de gerar arquivos VBR), como pode ser visto na Tabela 9. Foram criados, portanto, um total de 4082 arquivos, com 18,4 GB de espaço em disco.

Codificador	CBR			VBR	
	128	160	192	média	alta
WMP	x	x	x		
iTunes	x	x	x	x	x
Lame	x	x	x	x	x

Tabela 9 - Todas as codificações geradas.

Com a junção dos dois, obtemos uma base de testes de aproximadamente 20 GB de conteúdo distribuídos em 4306 arquivos.

6.2 EXPERIMENTOS

Os experimentos criados visam analisar o desempenho dos mecanismos propostos em relação ao que existe disponível.

6.2.1 Avaliando a utilização de recursos computacionais

O primeiro ponto importante é saber se a solução sugerida se comporta do ponto de vista do consumo de recursos. Para tanta utilizaremos duas medidas:

TAMANHO DA BASE DE COMPARTILHAMENTO

A base de compartilhamento é o resultado do processamento do par indexação-segmentação. Ela contém, para cada arquivo processado, o índice gerado do arquivo e o índice para cada segmento desse arquivo. Essas informações são armazenadas no *peer*, para que não seja necessário o reprocessamento, e enviadas ao servidor de busca, no

momento da conexão. Portanto, o tamanho da base de compartilhamento deve ser a menor possível. Na Figura 19, são apresentados os resultados do tamanho da base de compartilhamento para o SET, o *eDonkey* e para nossa proposta identificada como IBB+ISC+IF (contração para índice Bit-a-Bit, sensível ao conteúdo e de Frankenstein). Optamos por estabelecer como medida para o tamanho da base de compartilhamento, a proporção entre tamanho da base de compartilhamento (em *kilobytes*) e o tamanho da base de testes (em *megabytes*). Essa medida foi porque é independente do tamanho da base de testes e está em uma boa escala de observação.

Como pode ser visto na figura, apesar de possuir 2 índices a mais, IBB+ISC+IF é cerca de 1/3 menor que a base de compartilhamento gerada pelo SET. Isso se deve ao fato de que os segmentos o SET tem segmentos de tamanho que variam de 16KB a 64KB, o que é consideravelmente menor que os segmentos do SIT (250KB, em média) e *eDonkey* (9,28MB) apesar disso, nem a base de compartilhamento do SET chega a causar algum transtorno. Apesar da pequena quantidade de segmentos, na base gerada por SSP, cada segmento precisa guardar as informações de onde localizar cada camada de qualidade em cada *frame*. Na Figura 20, temos os valores separados para cada índice um dos três índices do IBB+ISC+IF.

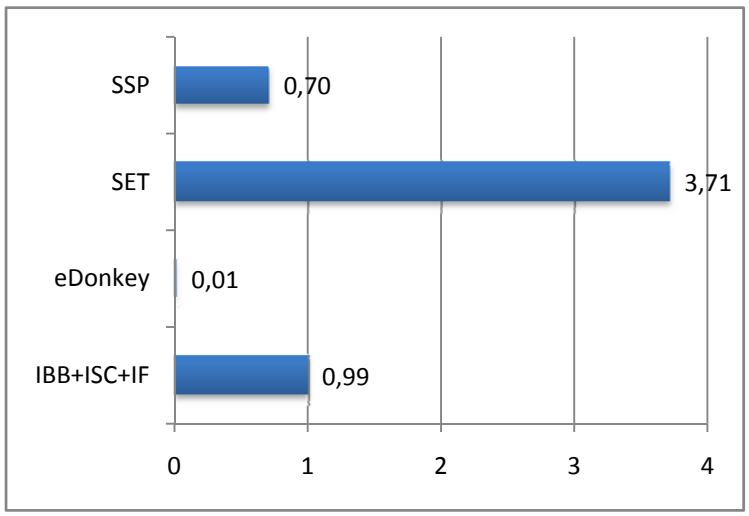

Figura 19 - Tamanho da base de compartilhamento (KB/MB). Índices agrupados.

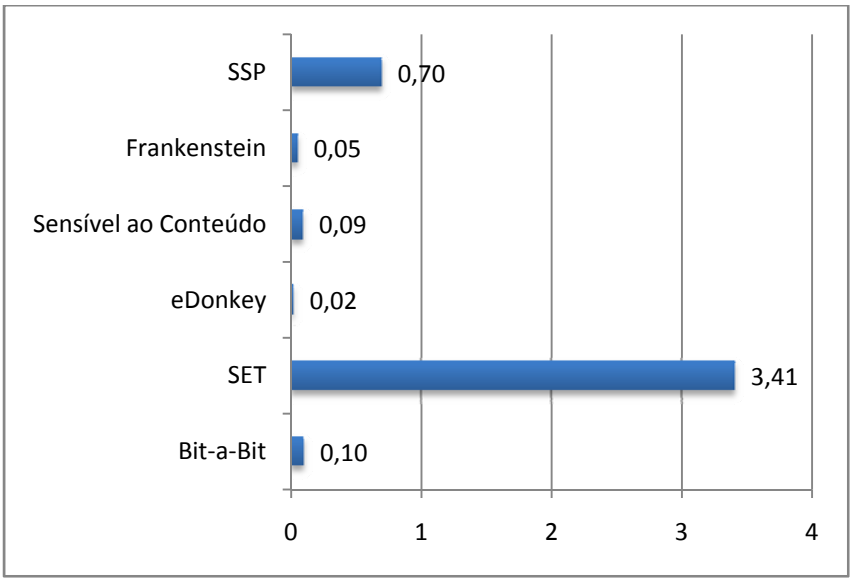

Figura 20 - Tamanho da base de compartilhamento (KB/MB).

VELOCIDADE DE CRIAÇÃO DA BASE DE COMPARTILHAMENTO

A velocidade de criação da base de compartilhamento indica em quanto tempo o arquivo leva para um arquivo é ser segmentado e indexado. Esta medida é expressa em MB/s (*megabytes* por segundo). Como pode ser visto na Figura 21, a velocidade de criação da base de compartilhamento tanto para o nosso método é menor a que a dos concorrentes. Em especial, o conjunto IBB+ISC+IF tem velocidade de criação da base duas ordens de magnitude menor que a dos outros. Em ambos os casos isso é de certa forma esperado, pois as segmentações e algumas das indexações propostas são bem mais complexas que a dos concorrentes. . O caso do SSP é menos grave, pois, na prática, à velocidade por volta de 650 KB/s não chega a, de fato, ser um transtorno ao usuário. O caso do IBB+ISC+IF, ao contrário, inspira uma análise mais cuidadosa.

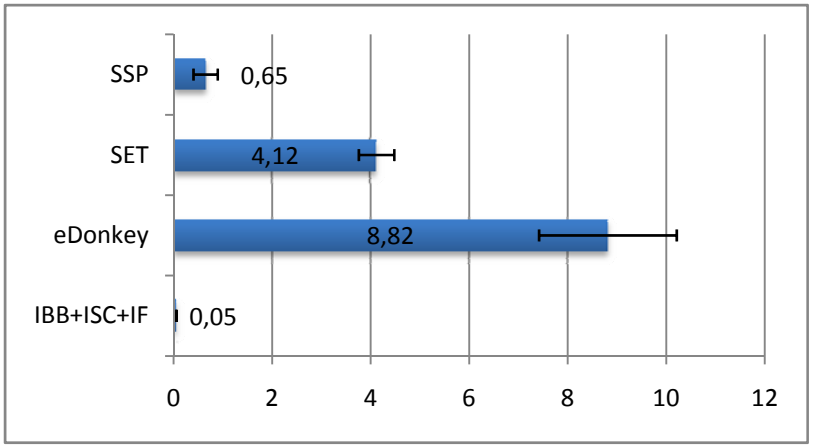

Figura 21 - Velocidade média de criação da base de compartilhamentos (MB/s).

Na Figura 22, fica claro que o índice de Frankenstein foi responsável pela baixa pontuação geral dos índices combinados. É possível perceber, por exemplo, que o índice Bit-a-Bit divide o primeiro lugar de velocidade com o *eDonkey* enquanto o ISC tem tecnicamente a mesma velocidade que o SET.

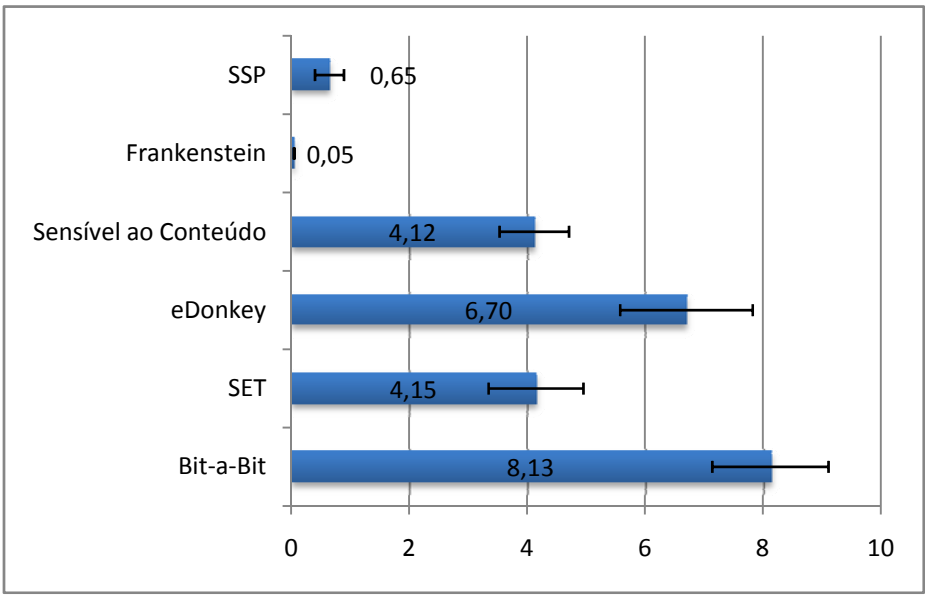

Figura 22 - Velocidade média de criação da base de compartilhamento (MB/s).

6.2.2 Avaliando o aumento de disponibilidade

Para avaliarmos como se comporta a variável disponibilidade, iremos analisar duas medidas. Para essas medidas iremos avaliar cada índice separadamente. Neste este requisito não incluiremos os resultados para SSP, uma vez que ela não trata desse requisito.

PERCENTUAL DE ARQUIVOS DISTINTOS

O percentual de arquivos distintos é dado pela razão entre o número de arquivos com índices distintos e o número total de arquivos na base de testes. Devemos interpretar essa medida como a dissimilaridade da base de testes. Quanto menor esse número, portanto, maior a será disponibilidade total da base de testes, uma vez que o risco de colisão, arquivos diferentes com o mesmo índice é muito baixo para os algoritmos de *hash* utilizados.

A Figura 23 indica que, apesar de sua lentidão, o IF consegue obter a maior disponibilidade na base de testes, seguida de perto do ISC. Os outros três índices encontram uma base 100% dissimilar, ou, de outra maneira, não conseguem aumentar a disponibilidade em nada, o que é já era esperado, inclusive para o SET, que encontra semelhanças em segmentos.

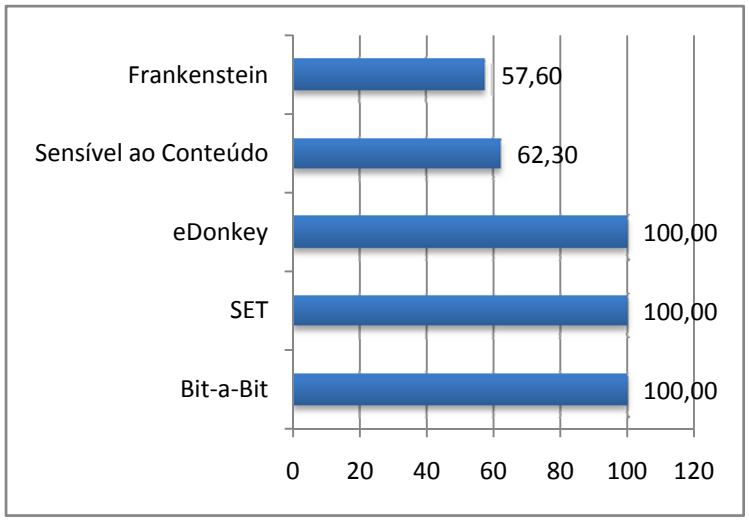

Figura 23 - Percentagem de arquivos distintos.

PERCENTUAL DE SEGMENTOS DISTINTOS

Diferentemente do que acontece com os arquivos, um (Daswani, Garcia-Molina et al. 2002)baixo percentual de segmentos distintos implica o aumento de disponibilidade, uma vez que as RCAs atuais não são capazes de usufruir dessa situação. Apesar disso, o percentual de segmentos distintos é um bom indicador do grau de semelhança entre os arquivos na base de testes. Na Figura 24 é possível perceber que o IF encontra só encontra 38,30% de arquivos dissimilares, um forte indicador de possíveis colisões. Outra informação digna de nota é são os resultados bastante similares entre ISC e o SET. Contudo, vale lembrar que a quantidade de segmentos produzida pela técnica de segmentação do SET é mais de 10 vezes maior que o que prova distorção nos dados.

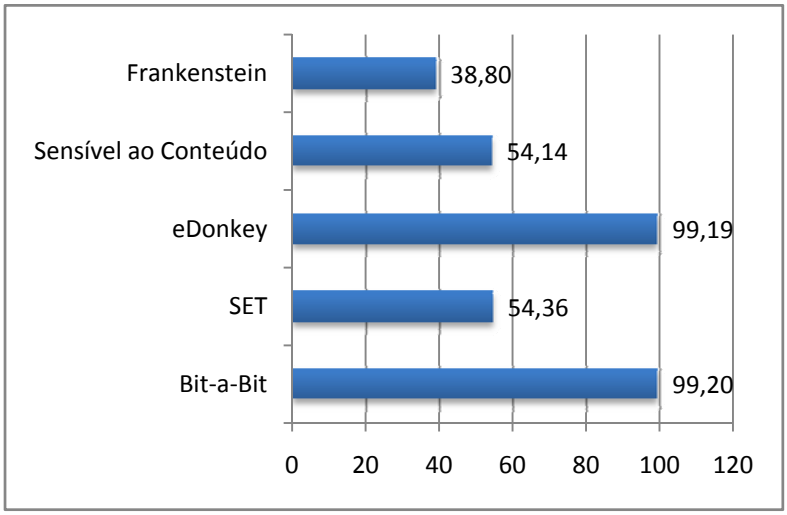

Figura 24 - Percentagem de segmentos distintos.

6.3 EXAMINANDO UM SUBCONJUNTO DOS DADOS

Visando uma melhor compreensão dos resultados obtidos com a base de testes, apresentamos a seguir, nas Figura 25, Figura 26 e Figura 27 o comportamento de um

subconjunto da base de testes. Trata-se de 15 arquivos diferentes com nome contendo a palavra Sebastiana baixados na RCA *eDonkey* conforme explicado na seção 6.1.1 . É interessante notar que na realidade foram baixados 16 arquivos nessa categoria, porém, um deles era sequer um arquivo MP3. O que nos alertou para isso foi o fato de o segmentador em intervalo de tempo, ter rejeitado o arquivo durante o processamento.

A Figura 25 mostra o resultado da Indexação Bit-a-Bit nos 15 arquivos. Nenhum agrupamento foi feito, o que significa que não foi possível encontrar mais de um arquivo com o mesmo índice e, portanto, cada arquivo possui apenas uma fonte, como pode ser visto na segunda coluna da figura. Na terceira coluna, chamada “Semelhantes” aparece o número de quantos dos segmentos do arquivo exposto na linha compartilham índice com algum outro segmento na base.

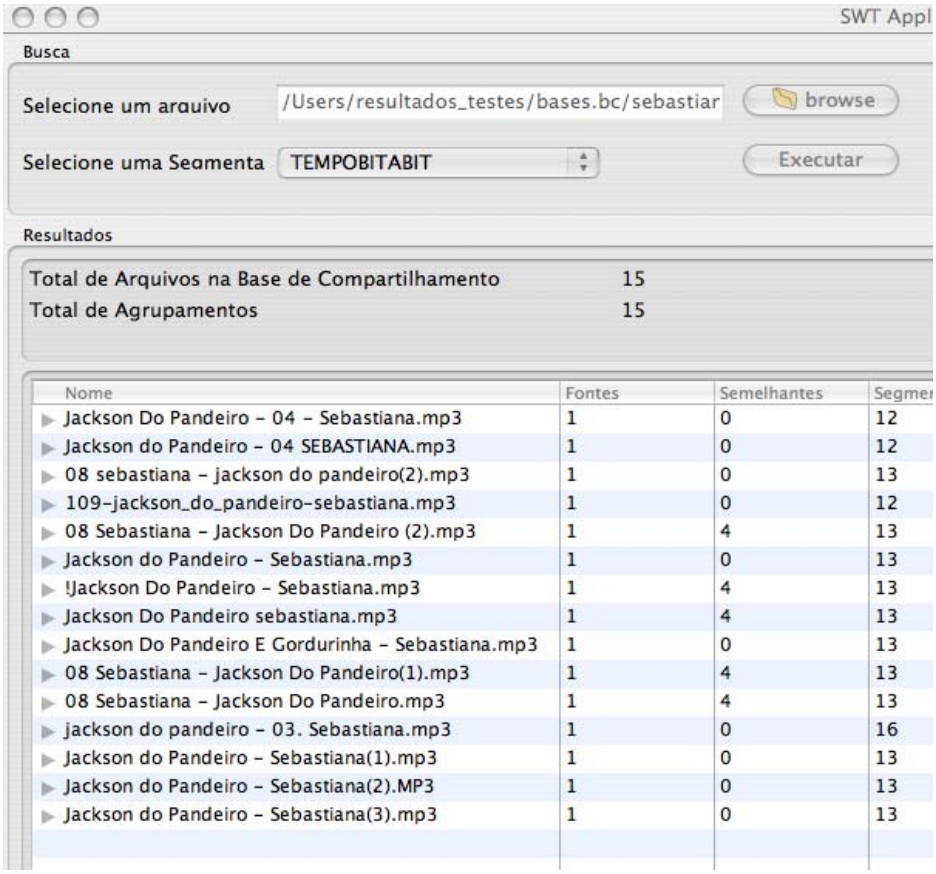

Figura 25 - Número de fontes disponíveis para música Sebastiana – IBB.

Na Figura 26, vemos o resultado da Indexação Sensível ao Conteúdo para os mesmo 15 arquivos. Percebemos aqui, o número de agrupamentos diminuiu, o que pode ser comprovado pelo número de fontes para o arquivo posicionado na ultima linha da tabela. O número de segmentos semelhantes também aumentou consideravelmente.

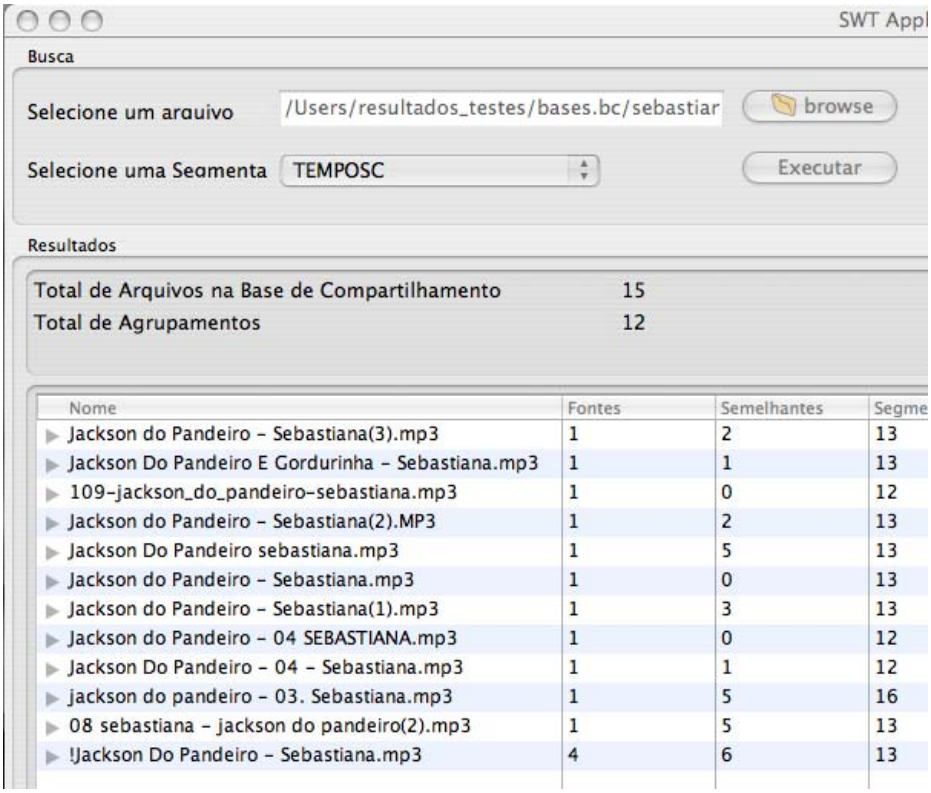

Figura 26 – Número de fontes disponíveis para música Sebastiana – ISC.

A Figura 27 exibe os resultados da Indexação Frankenstein para os 15 arquivos processados. Mais uma vez, houve redução no número de agrupamentos (de 12 no ISC para 10) com conseqüente aumento da disponibilidade para 2 arquivos da lista. O número de segmentos semelhantes subiu para em todos os arquivos, o que não se configura em erro obrigatoriamente, uma vez que pode se tratar dos mesmos 14 segmentos similares.

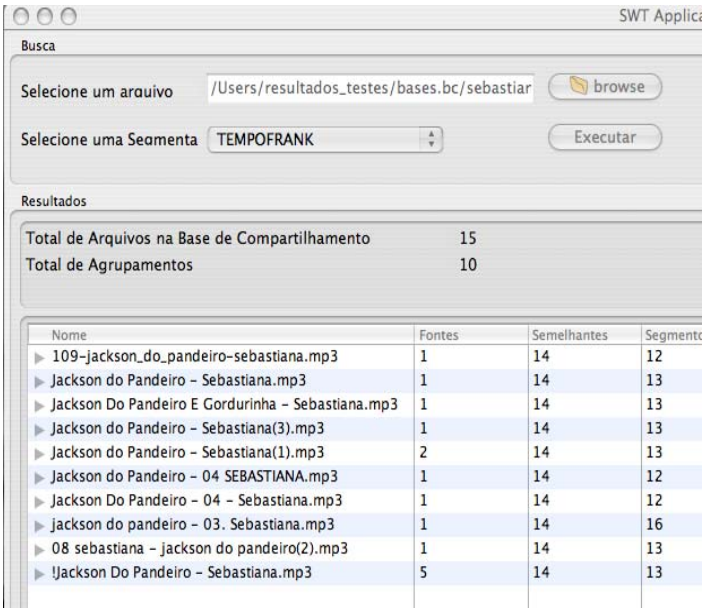

Figura 27 – Número de fontes disponíveis para música Sebastiana – IF.

6.3.1 Avaliando a Eficácia da Transferência

A primeira medida importante a se avaliar nesse caso é o *overhead* para o transporte dos segmentos. Esse valor é fixo e está associado à quantidade de *frame* do arquivo. Mais precisamente, ele é obtido pela necessidade de, no transporte, identificarmos cada camadas de qualidade de Wang (Wang, Huang et al. 2004c) para cada *frame*. A Tabela 10 indica o *overhead* para cada segmento e a média de *overhead* na base de testes, ou seja a média da razão entre a quantidade de bytes totais a serem transmitidos e quantidade de bytes por úteis para cada segmento.

Segmento	*Overhead* (bytes/*frame*)	Overhead (em %)
Segmento 1	5	6,94 ± 3,61
Segmento 2	3	5,90 ± 2,70
Segmento 3	6	1,92 ± 1,14
Segmento 4	6	8,90 ± 3,38

Tabela 10 - *Overhead* por frame para cada segmento

CAPÍTULO 7

CONCLUSÃO

Esse trabalho apresentou um esforço rumo à melhoria da experiência de download de áudio em redes de compartilhamento de arquivo. Ele se justifica pelo fato de que as RCAs atuais não são capazes de prover níveis aceitáveis de disponibilidade de conteúdo e eficácia na transferência.

Para lidar com o problema da disponibilidade, propomos novos mecanismos de indexação e de segmentação dos arquivos de áudio. A nova indexação é composta por três índices, O primeiro, chamado Índice Bit-a-Bit, similar ao índice usado tradicionalmente nas RCAs. O segundo, é o Índice Sensível ao Conteúdo, que aplica os algoritmo de *hash* exclusivamente nas informações que tem algum valor perceptivo, ignorando informações como: metadados e informações descartáveis nos *frames*. Com isso, arquivos considerados diferentes por terem apenas metadados distintos, por exemplo, passam a ser agrupados juntos, podendo ser utilizados como fonte mutuamente. O terceiro, chamado Índice de Frankenstein, consiste na aplicação de técnicas de *fingerprinting* para agrupar cópias do mesmo fonograma, porém em *bitrate* e codificações diferentes, por exemplo.

Para dar suporte a essa heterogenia, foi concebido um novo processo de segmentação, chamado Segmentação em Intervalo de Tempo, que divide o arquivo em segmentos com 400 *frames*, independentemente do tamanho em *bytes*. Os segmentos criados pela SIT, são em si, arquivos MP3 bem formados, o que permite a junção entre quaisquer 2 segmentos.

Para lidar com o problema da eficácia na transferência, foi desenvolvida a Segmentação por Significância Perceptiva, que utiliza técnica de cortes de frequencia de Wang para separar o arquivo em segmentos com diferentes níveis de importância perceptiva. Com a SSP é possível baixar o primeiro segmento (com grande significância perceptiva) para certificar-se da música, antes de baixá-la totalmente.

Os mecanismos de indexação e segmentação desenvolvidos foram comparados com os principais concorrentes, e demonstraram desempenho superior aos concorrentes em variáveis como número de segmentos distintos e numero de arquivos distintos na base de testes. No quesito desempenho de processamento, porém, o Índice de Frankenstein obteve a pior colocação, sendo quase 100 vezes mais lento que o penúltimo colocado.

Teoricamente, qualquer arquivo que possa ser separado e, ainda assim, decodificado é passível de receber uma segmentação por tempo. Todos os arquivos que podem ser usados em *streaming* encaixam-se nesse perfil, já que foram concebidos tendo em mente a possibilidade de perda de pacotes durante a transmissão.

7.1 CONTRIBUIÇÕES DO TRABALHO

A principal contribuição desse trabalho é fornecer mecanismos que tornam possível a transferência mais eficiente de, virtualmente, qualquer tipo de arquivo, bastando apenas que sejam elaboradas técnicas específicas de indexação e segmentações. A concepção de segmentadores e indexadores para diferentes formatos inspirado no conceito de plug-ins permite que sejam criados quantos indexadores e segmentadores forem desejados.

Outra contribuição é a técnica de baixar sempre o primeiro segmento da SSP que não só minimiza consideravelmente os efeitos de um download indesejado, como permite “testar” o conteúdo desconhecido.

Os resultados obtidos com o índice de Frankenstein, embora insipientes, apontam para a possibilidade de utilizar busca exata na identificação dos arquivos em detrimento da busca aproximada, que é consideravelmente mais lenta.

A utilização de um *parser* MP3 na geração do índice sensível ao conteúdo acabou levando ao efeito colateral de detecção e rejeição de arquivos mal formados. Dessa forma, arquivos mal formados são impedidos de ingressarem na rede, aumentando sua confiabilidade.

7.2 MELHORIAS NECESSÁRIAS NO TRABALHO ATUAL

Muito ainda deve ser feito para o enriquecimento deste trabalho. Alguns trabalhos em andamento são:

- Executar mais testes, com bases maiores e mais heterogêneas;
- Acrescentar algumas variáveis ainda não analisadas como, por exemplo, o consumo de memória durante a execução.
- Refinamento e otimização de código, principalmente para o índice de Frankenstein;
- Estudo e implementação de outras características que possam ser utilizadas no índice de Frankenstein;
- Diminuir o overhead da base de compartilhamento, que é um elemento critico no caso da segmentação por significância perceptiva.

7.3 TRABALHOS FUTUROS

A primeira tarefa a ser feita, evidentemente é a implementação efetiva dos mecanismos em uma RCA. Em nossa opinião, a rede *eDonkey*, com seu software cliente *eMule* parece ser a mais apropriada para isso. Outra tarefa importante é testar a generalidade dos mecanismos de segmentação e indexação implementando-os em outros formatos de áudio como *vorbis* e *AAC*. A princípio, excluindo o esforço natural de tal empreitada, não parece haver qualquer empecilho teórico.

Durante o decorrer do trabalho, várias ideias foram surgindo e não puderam ser implementadas por restrições de tempo.

A primeira ideia é que a segmentação em intervalo de tempo e a segmentação por significância perceptiva foram descritas e empregadas separadamente. Porém, para atingir a maior flexibilidade, é possível imaginar o uso combinado das duas da seguinte forma: primeiramente seria aplicada a segmentação por tempo, dividindo o arquivo em conjuntos de *frames*, em seguida, a cada segmento, seria aplicada a técnica de segmentação por qualidade. Obter-se-ia daí, quatro segmentos SSP para cada segmento SIT.

A segunda ideia é a concepção de segmentação e indexação para vídeos. É bastante comum encontrar um filme com áudio em diversas línguas, porém com o vídeo rigorosamente igual. Podemos imaginar uma segmentação que separa o vídeo do áudio, dessa forma, aumentar a disponibilidade do vídeo.

Por fim, uma terceira proposta de segmentação é tentar extrair de arquivos de áudio sem perda como o FLAC[19], uma segmentação no mesmo molde da SSP.

7.4 LIÇÕES APRENDIDAS

Os *papers* acadêmicos na área de codificação de áudio, raramente são confiáveis. Isso porque nunca mostram a real dificuldade de implementação de um determinado código. Parece ser unanimidade acadêmica nesta área retratar apenas a característica relevante associada ao problema a ser tratado e ignorar as outras, supondo-as “um esforço adicional irrelevante”. Há casos, porém, que esse esforço é muito maior que o necessário para desenvolver aquilo de que se trata o trabalho. Sugerimos, caso o leitor deseje fazer pesquisa na área, procurar listas de discussão e fórum na Internet, nesses locais, curiosamente, as perguntas difíceis são respondidas.

[19] http://flac.sourceforge.net/

REFERENCIAS BIBLIOGRÁFICAS

BBCNews. (2007). "Speed boost plan for file-sharing." http://news.bbc.co.uk/2/hi/technology/6544919.stm Acessado em 15/07/2007.

Broder, A. (1993). Some applications of Rabin's fingerprinting method. In Proceedings of the Sequences II: Methods in Communications, Security, and Computer Science, New York, USA, Springer-Verlag.

CacheLogic. (2004). "True Picture of P2P Filesharing." http://www.cachelogic.com/home/pages/studies/2004_01.php Acessado em 16/07/2007.

CacheLogic. (2005a). "Peer-to-Peer Filetype Study." http://www.cachelogic.com/home/pages/research/filetypestudy.php Acessado em 16/07/2007.

CacheLogic. (2005b). "Peer-to-Peer in 2005." http://www.cachelogic.com/home/pages/studies/2005_01.php Acessado em 16/07/2007.

Cano, P. (2006). Content-based audio search: from fingerprinting to semantic audio retrieval Departament de Tecnologia. Barcelona, Universitat Pompeu Fabra. **PhD Thesis:** 212.

Cano, P., Batlle, E., et al. (2002). Robust sound modeling for song detection in broadcast audio. In presentation at the 112th Audio Engineering Society. Munich, AES.

Christin, N., Weigend, A., et al. (2005). Content availability, pollution and poisoning in file sharing peer-to-peer networks. In Proceedings of the 6th ACM conference on Electronic commerce, ACM.

Chun, B., Wu, P., et al. (2006). ChunkCast: An anycast service for large content distribution. In Proceedings of the 5th International Workshop on Peer-to-Peer System (IPTPS 2006), Santa Barbara.

Costa, C. and Almeida, J. (2007). Reputation Systems for Fighting Pollution in Peer-to-Peer File Sharing Systems. In Proc. of the Seventh IEEE International Conference on Peer-to-Peer Computing (P2P 2007)

Daswani, N., Garcia-Molina, H., et al. (2002). Open problems in data-sharing peer-to-peer systems. In Proceedings of the 9th International Conference on Database Theory (ICDT '03), Springer-Verlag.

Dixon, S. (2007). "Evaluation of the Audio Beat Tracking System BeatRoot." Journal of New Music Research **36**(1): 39-50.

Doets, P. and Lagendijk, R. (2005). Extracting Quality Parameters for Compressed Audio from Fingerprints. In Proceedings of the 6th International Conference on Music Information Retrieval (ISMIR 2005). London, UK.

Dunn, C. (2006). Scalable Bitplane Runlength Coding. In presentation at the 120th Audio Engineer Society Convention, Paris, FR, AES.

Eastlake, D. and Jones, P. (2001). US Secure Hash Algorithm 1 (SHA1). Internet Engineer Task Force. http://www.ietf.org/rfc/rfc3174.txt Acessado em.

Ellacoya. (2007). Ellacoya Data Shows Web Traffic Overtakes Peer-to-Peer (P2P) as Largest Percentage of Bandwidth on the Network. Ellacoya Networks. Acessado em.

Golchin, F. and Paliwal, K. (2000). Lossless coding of MPEG-1 Layer III encoded audio streams. In Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing Istambul, TR, IEEE.

Gomez, E., Cano, P., et al. (2002). Mixed watermarking-fingerprinting approach for integrity verification of audio recordings. In Proceedings of the International Telecommunications Symposium, Natal, BR.

Gouyon, F., Pachet, F., et al. (2000). On the Use of Zero-Crossing Rate for an Application of Classification of Percussive Sounds. In Proceedings of the Conference on Digital Audio Effects, Verona, IT.

Haitsma, J., Kalker, T., et al. (2001). Robust audio hashing for content identification. in Proceedings of the Content-based multimedia indexing, Firenze, IT.

Herre, J. and Grill, B. (2000). Overview of MPEG-4 audio and its applications in mobile communications. In Proceedings of the 5th International Conference on Signal Processing, Istambul, TR, IEEE.

Houaiss, A. e. a. (2007). Dicionário eletrônico Houaiss da língua portuguesa. I. A. Houaiss, Editora Objetiva.

Huang, W., Wang, Y., et al. (2005). Power-aware bandwidth and stereo-image scalable audio decoding. In Proceedings of the 13th Annual ACM international Conference on Multimedia, Hilton, SG, ACM Press.

Ipoque. (2007). Internet Study 2007. Ipoque GmbH. Acessado em.

ISO/IEC. (1993). Information technology - Coding of moving pictures and associated audio for digital storage media at up to about 1,5 Mbit/s Part 3: Audio. Acessado em.

Izal, M., Keller, U., et al. (2004). Dissecting BitTorrent: Five months in a torrent's lifetime. In Proceedings of the 5th Passive and Active Measurement Workshop.

Kalker, T. (2001). Applications and challenges for audio fingerprinting. In presentation at the 111th Audio Engineering Society Convention, New York, EUA, AES.

Kubiatowicz, J. (2003). "Extracting guarantees from chaos." Communications ACM **46**(2): 33-38.

Kubiatowicz, J., Bindel, D., et al. (2000). OceanStore: an architecture for global-scale persistent storage. In Proceedings of the ninth international conference on Architectural support for programming languages and operating systems, ACM.

Kudumakis, P. and Sandler, M. (1996). Wavelet packet based scalable audio coding. In Proceedings of the IEEE International Symposium on Circuits and Systems, Espoo, FL.

Kulbak, Y. and Bickson, D. (2005). The eMule Protocol Specification. The Hebrew University of Jerusalem. Acessado em.

Legout, A. (2006). Rarest First and Choke Algorithms Are Enough. In Proceedings of the Internet Measurement Conference. Rio de Janeiro, ACM Press.

Lei, Z. and Georganas, N. (2001). Context-based media adaptation in pervasive computing. In Proceedings of the Canadian Conference on Electrical and Computer Engineering (CCECE). Toronto, CA.

Li, J. (2002). Embedded audio coding (EAC) with implicit auditory masking. In Proceedings of the 10th ACM International conference on Multimedia. Juan-les-Pins, France, ACM Press.

Manber, U. (1994). Finding similar files in a large file system. In Proceedings of the USENIX Winter 1994 Technical Conference.

Meyers, D. (2004). A Multi-Source Download Capable Gnutella Servent. University of Lancaster. http://www.lancs.ac.uk/~meyersd/fyp/report/fyp_report_daniel_meyers.pdf Acessado em 20/11/2008.

Milojicic, D., Kalogeraki, V., et al. (2002). Peer-to-peer computing. HP Laboratories. http://www.hpl.hp.com/techreports/2002/HPL-2002-57R1.pdf Acessado em.

Mitchell, T. (1997). Machine Learning, McGraw-Hill Education (ISE Editions).

Mohan, R., Smith, J. R., et al. (1999). "Adapting Multimedia Internet Content for Universal Access." IEEE Transactions on Multimedia **1**(1): 104-114.

Muthitacharoen, A., Chen, B., et al. (2001). A Low-Bandwidth Network File System. In Proceedings of the 18th Symposium on Operating Systems Principles**:** 174-187.

Pachet, F. (2003). "Content management for electronic music distribution." Communications of the ACM **46**(4): 71-75.

Painter, T. and Spanias, A. (2000). "Perceptual coding of digital audio." Proceedings of the IEEE **88**(4): 451-515.

Pfleeger, C. and Pfleeger, S. (2003). Security in computing, Pearson Education.

Pucha, H., Andersen, D. G., et al. (2007a). Exploiting similarity for multi-source downloads using file handprints. Proceedings of the 4th USENIX NSDI, Cambridge, MA.

Pucha, H., Andersen, D. G., et al. (2007b). Exploiting similarity for multi-source downloads using file handprints. In Proceedings of the 4th USENIX NSDI, Cambridge, MA.

Raad, M., Mertins, A., et al. (2003). Scalable to lossless audio compression based on perceptual set partitioning in hierarchical trees (PSPIHT). In Proceedings of the IEEE International Conference on Multimedia and Expo, IEEE.

Rivest, R. (1992). The MD4 Message-Digest Algorithm. Internet Engineer Task Force. http://www.ietf.org/rfc/rfc1320.txt Acessado em.

Roads, C. (1996). The Computer Music Tutorial, The MIT Press.

Rocha, J., Domingues, M., et al. (2004). Peer-to-Peer: Computação Colaborativa na Internet. Tutorial SBRC 2004. www.cin.ufpe.br/~cak/publications/sbrc2004_minicurso_p2p.pdf Acessado em 20/11/2008.

Salomon, D. (2004). Data Compression : The Complete Reference, Springer.

Salomonsen, K., Sogaard, S., et al. (1997). Design and Implementation of an MPEG Audio Layer III Bitstream Processor. Institute of Electronic Systems, Aalborg University. Acessado em.

Sandine. (2008). 2008 Global Broadband Phenomena. Sandine Incorporated. http://www.sandvine.com/general/documents/2008%20Global%20Broadband%20Phenomena%20-%20Executive%20Summary.pdf Acessado em.

Satyanarayanan, M. (2001). "Pervasive computing: Vision and challenges." IEEE Personal Communications ACM **8**(4): 10–17.

Schwarz, H., Marpe, D., et al. (2004). SNR-scalable extension of H.264/AVC. In Proceedings of the International Conference on Image Processing, Cingapura, IEEE.

Seo, K.-d., Kwon, S.-k., et al. (2003). Dynamic bit-rate reduction based on frame-skipping and requantization for MPEG-1 to MPEG-4 transcoder. In Proceedings of the IEEE International Symposium on Circuits and Systems, IEEE.

Shelley, M. W. (1994). Frankenstein. São Paulo, Cia. das Letras.

Silvestre, G., Kamienski, C., et al. (2006). Análise de tráfego peer-to-peer baseada na carga útil dos pacotes. II Workshop de Peer-to-Peer (WP2P 2006). Curitiba.

Tzanetakis, G. and Cook, P. (2000). Sound Analysis Using MPEG Compressed Audio. In Proceedings of the 2000 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP 2000).

Wang, H., Divakaran, A., et al. (2003). "Survey of Compressed-Domain Features used in Audio-Visual Indexing and and Analisys." Journal of Visual Communication & Image Representation(14): 150-183.

Wang, Y., Huang, W., et al. (2004a). A framework for robust and scalable audio streaming. In Proceedings of the 12th annual ACM international conference on Multimedia (MULTIMEDIA '04), ACM.

Wang, Y., Huang, W., et al. (2004b). A framework for robust and scalable audio streaming. Proceedings of the 12th annual ACM international conference on Multimedia (MULTIMEDIA '04), ACM.

Wang, Y., Huang, W., et al. (2004c). A framework for robust and scalable audio streaming. In Proceedings of the 12th annual ACM international conference on Multimedia, New York, EUA, ACM Press.

Wang, Y., Ojanpera, J., et al. (2001a). Schemes for Re-compressing MP3 Audio Bitstreams. In Proceedings of the 111th Audio Engineering Society Convention New York, EUA.

Wang, Y., Ojanpera, J., et al. (2001b). Schemes for Re-compressing MP3 Audio Bitstreams. In Proceedings of the 111th Audio Engineering Society Convention New York, EUA.

White, B., Lepreau, J., et al. (2002). An Integrated Experimental Environment for Distributed Systems and Networks. In Proceedings of the 5th USENIX OSDI. Boston**:** 255-270.