3
Ajuste Elástico de Áudio no Fluxo Comprimido

Este capítulo descreve primeiramente uma análise resumida do algoritmo proposto e a seguir a modelagem do fluxo de áudio comprimido adotada, o mecanismo genérico de ajuste de áudio proposto e a instanciação do mecanismo para alguns formatos de áudio. Por fim, o capítulo descreve como o algoritmo de ajuste pode ser aplicado em um fluxo de sistemas.

3.1.
Análise Resumida do Ajuste Elástico Proposto: Requisitos do Algoritmo

O problema de realizar ajuste elástico em uma mídia com compressão pode ser resolvido pelo menos de três maneiras diferentes, como já discutido quando da apresentação das soluções existentes. A primeira opção é realizar um pré- e pós-processamento de modo a somente aplicar o algoritmo de ajuste na mídia sem compressão. Nesse cenário, é preciso primeiro decodificar o fluxo, realizar o ajuste e, em seguida, codificá-lo novamente, como ilustrado na Figura 13.

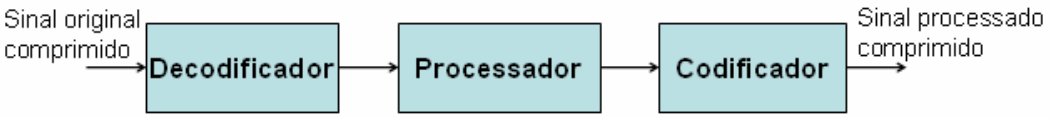

Figura 13 - Realização do ajuste em fluxo comprimido, utilizando decodificação prévia e nova codificação.

A vantagem dessa opção é que existem vários bons algoritmos de ajuste elástico para mídias sem compressão propostos na literatura (Lee et al., 2004). No entanto, essa estratégia é bastante custosa em termos de processamento, dificultando seu uso em tempo de exibição. Além disso, existe perda de qualidade inerente a uma nova compressão.

Outra possível solução é realizar o ajuste depois da descompressão (e antes da exibição), como ilustra a Figura 14. É fácil perceber que esse cenário é útil quando a recodificação não é mais necessária. A vantagem dessa opção é que o

algoritmo de ajuste ainda irá manipular um fluxo sem compressão. Por outro lado, essa opção precisa ser realizada de modo dependente do decodificador.

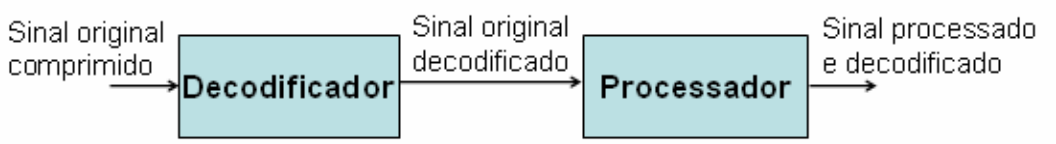

Figura 14 - Realização do ajuste em fluxo comprimido, utilizando decodificação prévia.

A última abordagem é realizar o ajuste antes de enviar o fluxo de dados para a descompressão, diretamente no fluxo de dados comprimido, como ilustra a Figura 15. A vantagem dessa opção é ter um ajuste computacionalmente mais barato e totalmente independente do decodificador. No entanto, o ajuste elástico deve ser realizado considerando as regras de formatação dos dados especificadas pelo padrão de compressão.

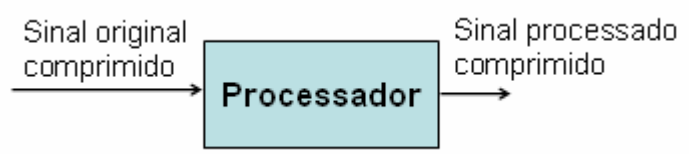

Figura 15 - Realização do ajuste diretamente no fluxo comprimido.

Considerando que o ajuste precisa ser realizado em tempo de exibição e independente do exibidor de conteúdo, a última abordagem foi adotada. Por isso, no decorrer deste trabalho será dito que o algoritmo de ajuste é aplicado diretamente no domínio comprimido.

O mecanismo de ajuste deste trabalho utiliza um fator de ajuste que pode ser modificado em tempo de exibição. Para preservar alta fidelidade em relação ao conteúdo original e atingir um processamento linear aceitável, o fator de ajuste deve variar em até *10%*, ou seja, entre o intervalo de *0.90* a *1.10*. É esperado que esse intervalo já seja suficiente para a manutenção do sincronismo entre mídias nas plataformas de exibição de documentos hipermídia atuais. Além disso, para dados audiovisuais, o vídeo correlacionado e multiplexado normalmente não tolera variação maiores que *5%*, sem perda de qualidade (Golubchik et al., 1995). Para comprovar que os requisitos de fidelidade e linearidade são atendidos, medidas subjetivas e objetivas de qualidade serão extraídas de mídias processadas (ver Seção 7.1).

Para atingir o requisito de independência do exibidor de conteúdo, o algoritmo de ajuste elástico deve preservar as mesmas características do fluxo

original, tanto em relação ao formato do fluxo como às características de codificação. Este trabalho considera importante, além da independência do exibidor, ser também independente do processo de decodificação, já que desse modo o mesmo algoritmo pode ser facilmente aplicado a vários formatos de mídia.

Como último requisito, o algoritmo deve ser desenvolvido como uma ferramenta de fácil acesso às aplicações clientes, chamada ferramenta de ajuste (ver Capítulo 4). Os clientes podem solicitar a realização do ajuste no local adequado, que pode ser no transmissor, no receptor ou em algum nó intermediário. A ferramenta de ajuste deve ser capaz de manipular arquivos locais, arquivos remotos e fluxos de mídia de dados "ao vivo", além de permitir monitorar marcações de tempo.

3.2. Modelagem do Fluxo de Áudio Com Compressão

Grande parte dos formatos atuais de áudio comprimido de alta qualidade define que um fluxo de mídia é constituído por um conjunto de quadros. O modelo adotado por este trabalho procura generalizar os formatos hoje existentes. Nele, cada quadro é univocamente identificado e possui um cabeçalho e um campo de dados. Além disso, cada quadro tem associado um conjunto de amostras referentes a um curto intervalo de tempo. O conjunto de amostras associado a um quadro, quando comprimidas, formam uma *unidade lógica de dados*.

A Figura 16 ilustra um exemplo de parte de fluxo de áudio com quadros divididos por grandes traços verticais. Os bits referentes ao cabeçalho são verticalmente listrados e se localizam sempre no início do quadro. Na figura, cada quadro transporta seu cabeçalho e sua unidade lógica de dados, e o campo de dados dos quadros sempre coincide com sua unidade lógica de dados. Entretanto, de modo mais geral, esses campos não são necessariamente iguais. O tamanho de uma unidade lógica de dados é diferente para cada quadro. Os dados de uma única unidade lógica associada a um quadro podem estar contidos no campo de dados do próprio quadro, ou também em campos de dados de quadros imediatamente anteriores. O conjunto de bits que armazenam amostras de quadros posteriores é denominado *reservatório de bits*, como ilustrado na Figura 17.

Figura 16 - Representação dos quadros que compõem um fluxo de áudio com unidade lógica de dados igual ao campo de dados.

Na Figura 17, quadros, como na figura anterior, são também divididos por grandes traços verticais e os bits referentes ao cabeçalho são verticalmente listrados e se localizam sempre no início do quadro. A unidade lógica do quadro *1*, entretanto, está ocupando parte do seu campo de dados e parte do campo de dados do quadro anterior. Os bits restantes no campo de dados do quadro *1* são suficientes para armazenar toda unidade lógica do quadro *2* e parte da unidade lógica do quadro *3*. O quadro *2* armazena somente a unidade lógica do quadro *3*. O quadro *3* armazena parte da sua unidade lógica e o início da unidade lógica do quadro *4*. Os bits iniciais e finais da figura não pertencem a nenhuma unidade lógica de dados e são representados apenas para indicar que o fluxo de mídia é maior do que os quatro quadros.

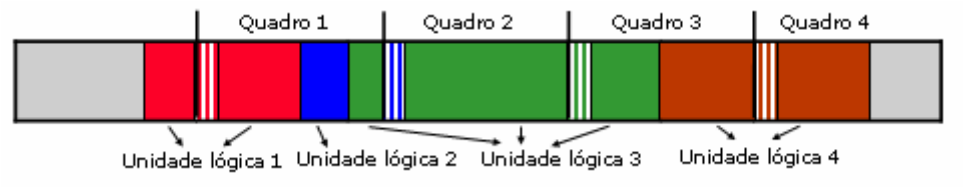

Figura 17 - Representação dos quadros que compõem um fluxo de áudio.

O modelo de fluxo de áudio comprimido ainda permite que no final de cada unidade lógica de dados existam dados de enchimento (PAD), ou seja, dados que não são provenientes da codificação das amostras daquele quadro. Esses dados podem representar algum metadado padronizado ou não. O tamanho do PAD não é limitado, mas os bits da unidade lógica do quadro nunca podem ocupar campo de dados de quadros posteriores. Por outro lado, podem existir bits de PAD em quadros anteriores, apenas se a unidade lógica de dados associada a esse quadro terminar antes mesmo do campo físico do quadro começar. A Figura 18 ilustra o mesmo fluxo de dados da Figura 17, mas apresenta que no final da unidade lógica de dados dos quadros *1* e *3* existem bits de PAD. Os bits de PAD estão ilustrados com a cor branca. Além dos dados de enchimento dentro das unidades lógicas de

dados, também podem existir bits antes de iniciar o primeiro quadro e depois de terminar o último quadro do fluxo de áudio.

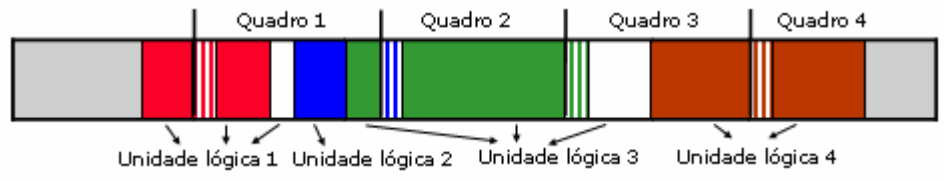

Figura 18 - Representação dos quadros que compõem um fluxo de áudio com bits de PAD no quadro 1 e 3.

O cabeçalho de um quadro sempre inicia com uma mesma seqüência de bits, chamada de *palavra de sincronismo*, para ajudar a manter o sincronismo. O primeiro quadro do fluxo inicia quando a primeira palavra de sincronismo for encontrada alinhada em bytes (ou seja, o primeiro bit da palavra de sincronismo deve ocorrer no primeiro bit de um byte do fluxo).

Um quadro sempre dispõe de informações suficientes para localizar sua unidade lógica de dados. Além disso, o tamanho do reservatório de bits não pode crescer indefinidamente, existindo um número máximo de bits que um quadro pode utilizar de quadros anteriores. Esse tamanho máximo varia entre diferentes formatos de áudio e depende do modo de localização da unidade lógica de dados do quadro e de restrições de codificação ou decodificação do formato.

Por fim, é válido destacar que a única base temporal do fluxo de mídia é a unidade lógica de dados. Como mencionado, cada quadro codifica um conjunto de amostras que representa um curto intervalo de tempo do sinal original. Tipicamente, esse intervalo é da ordem de poucas dezenas de milisegundos. Os bits de uma unidade lógica não necessariamente representam ordenadamente as amostras do sinal original no tempo, uma vez que as amostras originais são submetidas a várias transformações durante o processo de codificação.

3.3. Mecanismo de Ajuste Elástico de Áudio com Compressão

Considerando os requisitos de realizar o ajuste em tempo de exibição, alta fidelidade, pequena amplitude de variação do fator de escala e os formatos de áudio comprimidos estudados, a classe de algoritmos que funciona no domínio do tempo foi escolhida. Dentre os algoritmos dessa categoria, apenas o *Ajuste Regular* (ver Subseção 2.1.2) não realiza algum processamento envolvendo as

amostras do sinal. Esse foi o algoritmo escolhido como base para ser aplicado diretamente no formato de áudio comprimido porque os algoritmos mais complexos aumentariam muito o custo computacional e não atenderiam os requisitos de eficiência. Além disso, uma vez que esse trabalho foca em ajustes de pequena faixa, poucos defeitos são introduzidos pelo fato do algoritmo de ajuste ser simples e a qualidade do áudio resultante é aceitável, como apresentado nos testes de qualidade na Seção 7.1.

O algoritmo de ajuste é aplicado percorrendo o fluxo de áudio quadro a quadro e selecionando quadros a intervalos aproximadamente regulares para efetuar processamento. Processar quadros significa removê-los ou duplicá-los, caso se deseje, respectivamente, acelerar ou retardar a exibição de uma mídia. Considerando a modelagem do fluxo de áudio anteriormente descrita, a unidade de ajuste adotada no algoritmo de *Ajuste Regular* deve ser a unidade lógica de dados de um quadro.

Remover um quadro significa retirá-lo do fluxo de áudio, juntamente com sua unidade lógica de dados. No entanto, os bits do campo de dados do quadro removido que armazenam unidades lógicas de quadros posteriores devem permanecer no fluxo de dados, se espalhando pelos quadros anteriores não removidos. A Figura 19 ilustra o mesmo fluxo da Figura 17 após a simples remoção do quadro *3*. Nesse estágio, parte da unidade lógica do quadro *4* foi perdida. Para que isso não ocorra, os bits que estavam no campo de dados do quadro *3* que armazenavam amostras do quadro *4* precisam ser copiados para o quadro *2*, resultando no que está ilustrado na Figura 20. Como apresentado nas figuras, após a realização de operações de processamento de quadros, bits adicionais de PAD podem ter de ser inseridos no fluxo de dados.

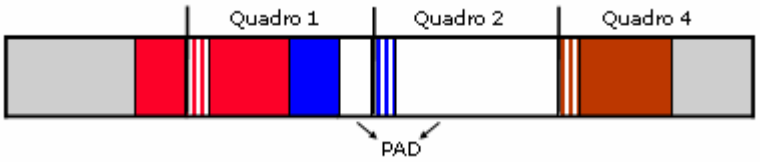

Figura 19 - Representação dos quadros no primeiro passo para remover o quadro 3.

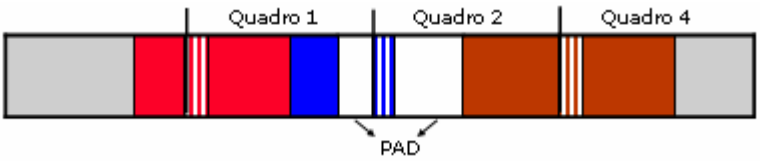

Figura 20 - Representação dos quadros após remoção do quadro 3.

De modo análogo, duplicar um quadro significa criar um novo quadro no fluxo de áudio e, possivelmente, modificar bits de quadros anteriores ao inserido no fluxo, com objetivo de inserir a unidade lógica de dados duplicada do quadro processado. A Figura 21 ilustra o fluxo de áudio após a simples duplicação do quadro *2* do fluxo de dados apresentado na Figura 17. Nesse estágio, os dados da unidade lógica do quadro *2* não foram duplicados. Para que isso ocorra, os bits dos quadros *1* e *2* precisam ser modificados, conforme ilustra a Figura 22.

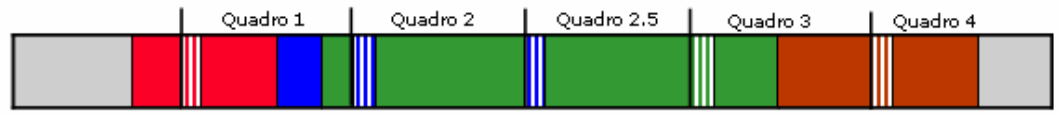

Figura 21 - Representação dos quadros no primeiro passo para duplicar o quadro 2.

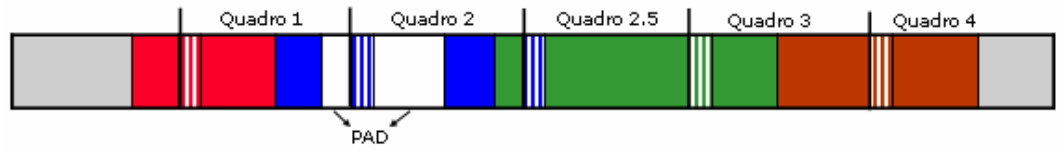

Figura 22 - Representação dos quadros após duplicação do quadro 2.

Nem todos os quadros têm um processamento fácil. No algoritmo proposto, a primeira decisão tomada foi considerar a diferença entre o tamanho do campo de dados e da unidade lógica de dados de um quadro durante o processamento. Assim, *a princípio*, nem todos os quadros podem ser processados (removidos ou duplicados). As regras consideradas foram as seguintes:

1. Somente os quadros que possuem o tamanho da unidade lógica de dados associada maior do que o tamanho do campo de dados podem ser removidos. Essa decisão evita a falta de espaço para dados de unidades lógicas de quadros posteriores.

2. Analogamente, apenas os quadros que possuem o tamanho da unidade lógica de dados associada menor do que o tamanho do campo de dados podem ser duplicados. Essa decisão evita que falte espaço para a unidade de dados lógica duplicada.

Os exemplos anteriores de remoção e duplicação respeitam essas regras. Na Figura 17, a unidade lógica do quadro *3* ocupa todo o campo físico do quadro *2* e parte dos campos físicos dos quadros *1* e *3*. Por isso, a remoção do quadro *3* pode ocorrer sem problemas, segundo a regra *1* acima. Por essa regra, os quadros *1* e *2* não podem ser removidos. De modo análogo, a unidade lógica do quadro *2* ocupa

apenas parte do campo físico do quadro *1*. O quadro *2* pode, então, ser duplicado segundo a regra *2* acima. Segundo a regra *2*, os quadros *3* e *4* não podem ser duplicados.

Sempre que ocorre uma operação de processamento respeitando as regras *1* e *2*, bits de PAD devem ser inseridos no espaço que restou no fluxo de áudio. Os bits de PAD podem ser transferidos entre os quadros do fluxo utilizando o seguinte algoritmo:

- Calcular o número *X* de bits do PAD que podem ser transferidos de um quadro para o próximo considerando que o tamanho máximo que o reservatório de bits pode atingir não pode ser ultrapassado.
- Deslocar em *X* bits a unidade lógica de dados do próximo quadro para ocupar parte do espaço atual do PAD e inserir novos bits de PAD ao final da unidade lógica transferida.

A Figura 23 ilustra o fluxo de áudio da Figura 22 após a transferência do PAD do final da unidade lógica do quadro *1* para o final da unidade lógica do quadro *4* (depois de ter passado pelos quadros *2*, *2.5* e *3*). A presença do PAD no final da unidade lógica de um quadro nunca invalida o fluxo, pois tais bytes são interpretados como uma continuação dos dados auxiliares do quadro.

Figura 23 - Representação dos quadros após transferência do PAD para quadro 4.

Desse modo, os bytes que formam o enchimento podem ser utilizados para refinar o algoritmo de ajuste elástico, anteriormente proposto, com as seguintes regras:

3. Quadros que não atendem à regra de remoção definida no item *1* podem ser retirados do fluxo, desde que o tamanho da unidade lógica de dados associada, somada aos enchimentos do quadro, seja maior do que o tamanho do campo de dados.

4. Quadros que não atendem à regra de duplicação definida no item *2* podem ser re-inseridos no fluxo se o tamanho da unidade lógica de dados associada for menor do que o tamanho do campo de dados, somado a enchimentos de quadros anteriores.

Com a utilização das novas regras, mais quadros podem ser processados. Por exemplo, seria possível duplicar o quadro *4* no fluxo de áudio da Figura 23. Para fazer uso de tais regras, um PAD está sendo constantemente transferido entre os quadros do fluxo. O número de bits do PAD aumenta quando um quadro que atende às regras *1* e *2* é processado e diminui quando um quadro que não atende a essas regras, mas satisfaz as regras *3* e *4*, é processado ou quando não é possível transferir o PAD completo para o quadro seguinte por esse ter atingido o tamanho máximo do reservatório de bits.

É válido destacar que o tamanho dos quadros não é necessariamente constante. O algoritmo extrator de quadros é responsável por reconhecer e criar quadros utilizando as seguintes regras:

a) Reconhecer e criar o primeiro quadro analisando os bits iniciais do fluxo byte a byte até encontrar a palavra de sincronismo.

b) Verificar se imediatamente depois do final do quadro reconhecido existe outra palavra de sincronismo.

- Se sim, reconhecer novo quadro e voltar ao passo (b).
- Se não, indicar que fluxo de quadros terminou.

Durante o processamento do áudio, cada quadro é analisado para verificar se deve ou não ser processado. Essa decisão é responsabilidade do algoritmo de seleção de quadros a processar, descrito a seguir:

a) Calcular o fator de ajuste efetivamente aplicado ao fluxo de áudio até o momento considerando o número de quadros já processados dividido pelo número de quadros já analisados.

b) Comparar o resultado do fator de ajuste efetivamente aplicado com o fator originalmente solicitado para decidir se o quadro em análise deve ou não ser processado.

- Se sim, verificar se o quadro pode ser processado considerando os tamanhos do PAD, do campo físico e da unidade lógica do quadro segundo as regras 1, 2, 3 e 4 citadas.

Caso ocorra uma solicitação de mudança no fator de ajuste, o algoritmo de seleção de quadros recalcula o fator de ajuste efetivo do momento da solicitação em diante, desconsiderando os ajustes anteriores.

Uma vez que nem todos os quadros podem ser processados, o conjunto de quadros escolhidos pelo algoritmo de seleção de quadros a processar não é regularmente distribuído. Uma medida de linearidade do algoritmo será discutida na Seção 7.1.

A não-linearidade do algoritmo aliada ao fato da granularidade da unidade de ajuste elástico ser discreta implica que os novos valores das âncoras (lembrando que uma âncora marca um instante de tempo em um fluxo audiovisual) existentes sobre o fluxo de áudio não podem ser simplesmente estimados através de uma relação linear do fator de ajuste aplicado (ver Figura 24). Em outras palavras, considerando que existia uma âncora no fluxo de áudio no tempo a_1 e que foi aplicado um fator de ajuste f, não é válido supor que o novo valor da âncora após o ajuste será $f*a_1$.

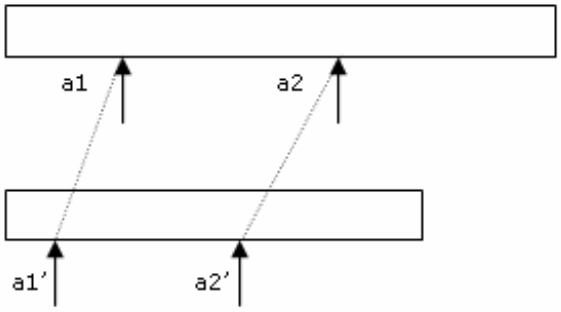

Figura 24 – Reposicionamento de âncoras em um fluxo de mídia devido ao ajuste.

Por essa razão, o mecanismo de ajuste elástico é capaz de indicar quais os novos valores de determinados instantes de tempo do fluxo original da mídia após o processamento. Isso pode ser feito depois da realização do ajuste ou durante sua execução. No primeiro caso, o programa que solicitou previamente a realização do ajuste pode perguntar qual os novos tempos dos instantes desejados. Quando realizado em tempo de execução, a chamada da realização do ajuste deve indicar quais instantes deseja monitorar. Nesse caso, o próprio algoritmo de ajuste

elástico dispara eventos ao encontrar o novo valor de uma âncora desejada. Sendo assim, o programa que solicita o ajuste pode ser avisado em tempo de exibição qual o novo valor de apresentação da âncora.

Por fim, cabe ressaltar que o algoritmo de ajuste proposto preserva bits do fluxo de áudio existentes antes de iniciar o primeiro quadro e depois de terminar o último quadro, bem como dados de enchimento dos quadros presentes no fluxo de áudio original. A preservação desses bits é muito importante uma vez que esses podem representar metadados importantes.

3.4. Ajuste Elástico em Fluxos de Áudio Suportados

O mecanismo de ajuste da seção anterior foi aplicado a alguns formatos de áudio comprimido. As subseções a seguir detalham as particularidades do mecanismo de ajuste proposto para MPEG-1 áudio, MPEG-2 áudio BC, MPEG-2 AAC, MPEG-4 AAC e AC-3.

3.4.1. MPEG-1 áudio

O padrão MPEG-1 áudio (ISO, 1993) define três camadas de codificação de áudio, denominadas MP1, MP2 e MP3. Quanto maior o número da camada, mais complexa é a sua codificação, sua sintaxe dos dados gerados e sua decodificação. Todas as camadas podem codificar áudio com um ou dois canais e taxa de amostragem de *32*, *44.1* ou *48*kHz. A Tabela 1 ilustra uma comparação entre as 3 camadas acima citadas em relação à taxa de bits por canal e taxa de compressão atingida.

Camadas	**Taxa de bits (kbp/s)**	**Taxa de compressão**
MP1	32 a 448	4:1
MP2	32 a 384	6:1 - 8:1
MP3	32 a 320	10:1 - 12:1

Tabela 1 - Comparação entre camadas do MPEG-1 - fonte: (Noll, 2000; Soares, 2005).

O formato de dados de todas as camadas é dividido em quadros. O tamanho de um quadro é determinado a partir de uma fórmula que relaciona a taxa de bits e

a de amostragem[5] que, na maioria dos casos, não se modifica dentro de um mesmo fluxo de áudio. As fórmulas a seguir calculam o tamanho de um quadro em fluxos MP3, MP2 e MP1.

$$Tamanho\_Quadro = 144 * \frac{taxa\_de\_bits}{taxa\_de\_amostragem} + (byte\_complementar)$$

Figura 25 - Cálculo do tamanho de um quadro MP2 ou MP3.

$$Tamanho\_Quadro = 12 * \frac{taxa\_de\_bits}{taxa\_de\_amostragem} + (byte\_complementar)$$

Figura 26 - Cálculo do tamanho de um quadro MP1.

Nas fórmulas acima, as taxas de bits e de amostragem podem assumir os valores já mencionados. A unidade a considerar da taxa de bits é de kbp/s e da taxa de amostragem é kHz. Se a divisão da taxa de bits pela taxa de amostragem não resultar em um número inteiro, o resultado é truncado e, nessas situações, o codificador irá adicionar um byte complementar em alguns quadros. Em todos os demais casos, o valor do byte complementar é sempre zero. Vale mencionar que caso a taxa de bits mude no decorrer do fluxo, indicando a mudança do tamanho dos quadros, diz-se que o fluxo é VBR (*variable bit rate*), mas muitos decodificadores não são capazes de manipular fluxos desse tipo.

O MPEG-1 não padroniza bits que possam existir antes ou depois dos quadros do fluxo, embora seja comum que alguns codificadores armazenem metadados nesses bits. Por exemplo, existe um padrão informal[6] chamado *ID3 tag* (Nilsson, 2003) que se aplica a arquivos que armazenam música em formato MP3. O padrão é bastante utilizado e convenciona utilizar os últimos *128* bytes do arquivo para adicionar informações relacionadas à música como: nome, artista, álbum, ano de lançamento, gênero, comentário e outros dados definidos pelo usuário.

Um quadro MPEG-1 áudio é composto pelos seguintes campos:

[5] Existe a possibilidade de um campo do cabeçalho desses formatos indicar que o tamanho do quadro não é determinável *a priori*, mas somente durante a interpretação individual dos campos do quadro. Esse formato é conhecido como livre e não foi tratado por este trabalho.

[6] Padrão informal quer dizer uma especificação elaborada por alguém, mas não regulamentada por nenhum órgão de padronização (como ISO e ITU).

- *Cabeçalho*: inicia com a palavra de sincronismo "*1111 1111 1111*" e contém dados de descrição do quadro, como presença ou não de CRC, taxa de bits e taxa de amostragem.
- *CRC*: campo opcional para detectar erros em partes críticas do quadro.
- *Dados de áudio*: contém todas as informações relativas às amostras de áudio desse quadro.
- *PAD*: campo opcional para bits de enchimento.

Ao empregar o algoritmo da seção anterior, os campos *cabeçalho* e *CRC* foram mapeados para o campo de *cabeçalho* do formato genérico e os campos *dados de áudio* e *PAD* foram mapeados para a *unidade lógica de dados*.

Um quadro MP1 contém informações de *384* amostras de áudio e quadros MP2 e MP3 contêm informações de *1152*. O tempo necessário para exibir as amostras associadas a um quadro de fluxos MPEG-1 é calculado pela divisão do número de amostras no quadro pela taxa de amostragem. A Tabela 2 ilustra esses valores em milisegundos.

Camada	**Taxa de amostragem em kHz**		
	32	**44,01**	**48**
I	12	8,73	8
II	36	26,12	24
III	36	26,12	24

Tabela 2 - Duração para exibir as amostras associadas a um quadro (em milisegundos).

Em fluxos MP1 e MP2 não existe reservatório de bits e, conseqüentemente, o campo físico é igual à unidade lógica de dados. Desse modo, qualquer quadro do fluxo pode ser processado pelo mecanismo de ajuste e nunca são inseridos ou transferidos bytes de PAD. Sendo assim, o mecanismo de ajuste proposto pode ser utilizado supondo que o número de bytes emprestados é sempre zero.

Em fluxos MP3 existe o conceito de reservatório de bits e o mecanismo de ajuste proposto pode ser utilizado em sua plenitude. Cada quadro possui um campo de *9* bits para armazenar o número de bytes emprestados de quadros anteriores (chamado *main_data_begin*). Sendo assim, um quadro MPEG-1 MP3 pode depender de, no máximo, *5* quadros anteriores.

O processamento de quadros MP3 durante a realização do ajuste deve alterar o valor do campo *main_data_begin* para alterar o tamanho do reservatório

de bits dos quadros. Como conseqüência, o campo CRC (caso exista) precisa ser recalculado.

O algoritmo de ajuste poderia aproveitar os bits de preenchimento do final da unidade lógica dos quadros para aumentar o número de bits do PAD e facilitar o processamento de quadros. No entanto, esse trabalho optou por preservar esses bits porque eles podem representar metadados importantes para alguma aplicação, como já mencionado anteriormente.

3.4.2. MPEG-2 áudio BC

O MPEG-2 áudio BC (*Backward Compatible*) (ISO, 1998) também possui três camadas de codificação que geram fluxos MP1, MP2 e MP3. Esse formato define dois novos padrões de áudio: LSF (*Low Sampling Frequencies*) e MC (*Multi Channel*).

O MPEG-2 áudio LSF define um formato de dados bastante semelhante ao MPEG-1 áudio. Algumas diferenças entre esses formatos relevantes para o contexto deste trabalho podem ser enumeradas.

1. As taxas de amostragem suportadas pelo novo padrão são *16*, *22.05* ou *24*kHz.
2. As taxas de bits também estão mais baixas. De 32 a 256kbit/s para fluxos MP1 e de 8 a 160kbit/s para fluxos MP2 e MP3.
3. O número de amostras associadas a um quadro no MP2 é *576*.
4. Alguns campos do formato foram modificados, por exemplo, o *main_data_begin* passou a ter *8* bits. Devido a essas mudanças, um quadro MP3 pode depender de até *10* quadros anteriores considerando o tamanho mínimo de um quadro e o tamanho máximo do reservatório de bits.

O mecanismo de ajuste elástico no MPEG-2 BC LSF é realizado de modo similar ao que foi descrito para o MPEG-1, não trazendo nenhuma informação adicional que mereça destaque.

O MPEG-2 áudio MC define um formato compatível com o MPEG-1 áudio e acrescenta, opcionalmente, mais *4* canais de *surround*, para possibilitar configuração 5.1, e até *7* canais com qualidade de voz. Um decodificador MPEG-

2 áudio MC é capaz de reproduzir áudio codificado como MPEG-1 áudio do mesmo modo que um decodificador MPEG-1 áudio. Porém, um decodificador MPEG-1 áudio reproduz áudio codificado como MPEG-2 áudio MC desconsiderando as extensões do formato de dados acrescentadas pelo novo padrão, ou seja, desconsiderando os novos canais. Isso ocorre porque os dados de tais canais são adicionados no campo de PAD dos quadros. A Figura 27 ilustra um codificador e decodificador MPEG-2 áudio MC, destacando a semelhança com codificadores e decodificadores MPEG-1.

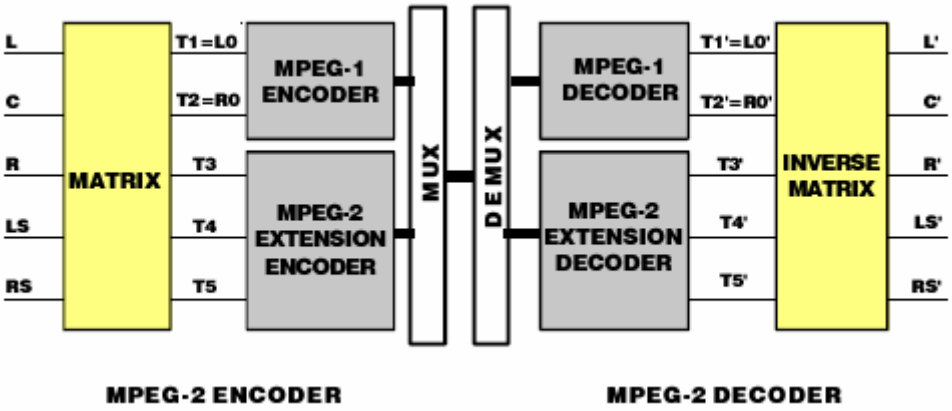

Figura 27 - Codificador e decodificador MPEG-1 e MPEG-2 áudio MC - fonte: (Noll, 1999).

O mecanismo de ajuste elástico no MPEG-2 BC MC também é realizado de modo similar ao que foi descrito para o MPEG-1. Vale comentar apenas que os canais de *surround* estão codificados nos quadros do sinal de áudio da mesma maneira que os outros canais. Portanto, o descarte e duplicação de quadros do algoritmo de ajuste também acontece com os canais de *surround*, sem modificar o algoritmo proposto.

3.4.3. MPEG-2 AAC

O AAC (*Advanced Audio Coding*) (ISO, 1997; ISO, 2004) é o padrão de áudio definido pelo MPEG-2 para aplicações que não precisam manter a compatibilidade com o formato MPEG-1 áudio.

Esse padrão suporta taxa de amostragem de *8* a *96*kHz e taxa de bits varia de *48* até *576*kbits/s por canal. Utilizando o AAC, é possível obter uma taxa de compressão de *16:1* mantendo a qualidade de um áudio de CD. O AAC pode

suportar até *48* canais, sendo, portanto, possível configurar o som em *5.1*, *7.1* ou mesmo *10.2* canais.

Dada a variedade grande de configurações possíveis para número de canais e taxas de amostragem e de bits, o AAC define um conjunto de perfis de configuração padrão. São eles:

- *Low-complexity* (LC): utiliza um conjunto de ferramentas de codificação para possibilitar a codificação e, principalmente, a decodificação com pequeno custo computacional.
- *Main*: inclui o perfil LC e adiciona novas ferramentas de codificação com objetivo de aumentar a taxa de compressão com o custo de aumentar a complexidade computacional.
- *Scalable Sampling Rate* (SSR): codifica fluxo em camadas complementares de modo que decodificadores diferentes possam extrair somente as camadas necessárias. A qualidade do fluxo obtido pode ser menor do que a do perfil *LC,* chegando até a do perfil *Main*.

O padrão AAC define que as amostras codificadas são dividas em blocos, cada qual com informações de *1024* amostras. Entretanto, como não existe um campo que determina o tamanho dos blocos, estes podem ser delimitados somente ao final da leitura e interpretação completa de seus bits.

Para permitir a correta identificação dos blocos, o padrão AAC sugere dois formatos de dados para protocolos de transporte. O formato ADIF (*Audio Data Interchange Format*) contém um cabeçalho inicial seguido por blocos. Desse modo, esse formato é adequado apenas para situações em que o fluxo será lido do inicio ao fim, e onde a possibilidade do decodificador perder o sincronismo é bastante remota, como a leitura de um arquivo armazenado localmente.

O formato ADTS (*Audio Data Transport Stream*) possui sintaxe similar ao MPEG áudio BC, sendo, por isso, conhecido como MP4. Um quadro ADTS é composto pelos seguintes campos:

- *Cabeçalho de tamanho fixo*: inicia com a palavra de sincronismo "*1111 1111 1111*" e contém dados de descrição do quadro, como presença ou não de CRC e taxa de amostragem.
- *Cabeçalho de tamanho variável*: contém informações de *copyright* (codificadas nos campos *copyright_identification_bit* e

copyright_identification_start), o tamanho do quadro (campo *frame_length*), o tamanho do reservatório de bits (campo *adts_buffer_length*) e o número de blocos no quadro (campo *number_of_raw_data_blocks_in_frame*).

- *CRC*: campo opcional para detectar erros em partes críticas do quadro.
- *Blocos de áudio*: contém um conjunto de blocos AAC.
- *PAD*: campo opcional com bits de enchimento.

Nesse formato, cada quadro pode conter informações de até *4* blocos (campo *number_of_raw_data_blocks_in_frame*), o que equivale a dizer que cada quadro possui informações de até *4*1024* amostras. A vantagem do codificador adicionar vários blocos em um quadro é diminuir o *overhead* de codificar os campos de cabeçalho e CRC no início de um quadro. Entretanto, é válido destacar que sempre é possível codificar um fluxo AAC utilizando quadros com um único bloco e que em todas as codificações pesquisadas durante o decorrer deste trabalho, essa foi a opção de compressão utilizada pelos codificadores.

Os quadros ADTS podem conter reservatório de bits cujo tamanho é indicado pelo campo *adts_buffer_fullness*. O tamanho máximo do reservatório de bits depende do número de canais do fluxo e da taxa de bits média.

O mecanismo de ajuste elástico proposto foi aplicado ao formato MPEG-2 AAC ADTS. Os campos *cabeçalho de tamanho fixo*, *cabeçalho de tamanho variável* e *CRC* do quadro ADTS foram mapeados para o campo de *cabeçalho* do formato genérico e os demais campos foram mapeados para a *unidade lógica de dados*.

O ADIF e ADTS são apenas exemplos de protocolo de transporte. Outros formatos podem ser utilizados para encapsular os blocos de dados AAC, como, por exemplo, o protocolo do MPEG-4 Sistemas (ISO, 2001a).

É importante ainda ressaltar que um novo problema surge nas regras de seleção e processamento de quadros em fluxos do perfil principal do AAC, pois existe uma ferramenta de codificação que permite que o codificador utilize valores de amostras de blocos anteriores para prever os valores de amostras do bloco atual. No pior caso, pode demorar até 240 quadros para o codificador gerar um quadro totalmente independente dos anteriores (Purnhagen, 2004). O algoritmo

proposto não funciona com tamanha dependência entre quadros, ficando a generalização do algoritmo para cobrir este caso proposta como trabalho futuro.

3.4.4. MPEG-4 AAC

O padrão MPEG-4 áudio (ISO, 2001b) codifica sinais de voz humana e áudio multicanal com alta qualidade e também oferece suporte a áudios naturais e sintéticos, como explicado na Subseção 2.2.8. Para os sinais naturais de áudio, existe suporte à codificação entre *2*kbit/s até *64*kbit/s, existindo *3* tipos de codificação, como ilustrado na Figura 28.

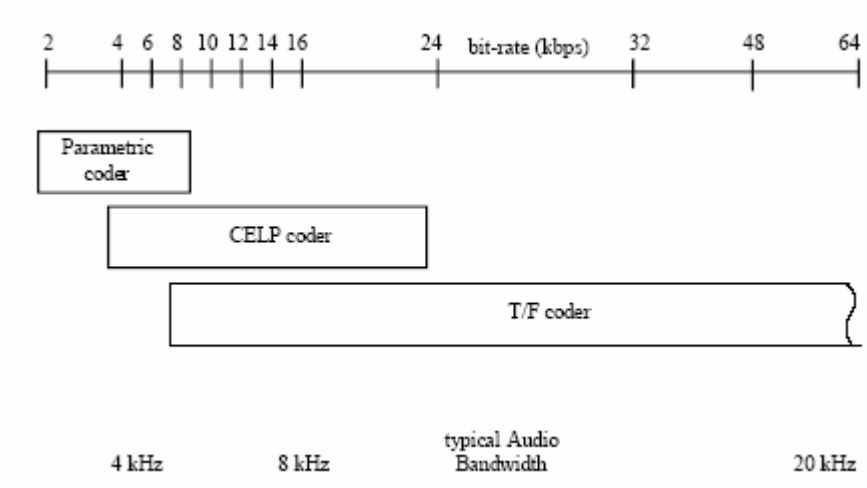

Figura 28 - Tipos de codificação para diferentes taxas de bits em MPEG-4 natural - fonte: (ISO, 2001b).

O codificador MPEG 4 utiliza técnicas paramétricas para sinais com taxas de bits entre *2* e *8*kbit/s. Essa técnica é utilizada principalmente para voz humana com taxa de amostragem *8*kHz. Para taxas de *4* a *24*kbit/s, o codificador utiliza técnicas CELP (*Coding Excited Linear Predictive*). Essa codificação pode ser utilizada para voz ou outros tipos de áudio com taxas de amostragem de *8*kHz e *16*kHz. Para taxas de bits mais altas (em geral, acima de *16*kbps) e taxas de amostragem a partir de *8*kHz, são utilizadas técnicas que se baseiam no MPEG-2 AAC.

O formato de dados do fluxo MPEG-4 AAC é similar ao MPEG-2 AAC, embora existam várias ferramentas de codificação novas. O MPEG-4 pode ser transmitido por diferentes protocolos de transporte, assim como o MPEG-2. O

mecanismo de ajuste elástico foi aplicado apenas aos fluxos MPEG-4 AAC ADTS.

3.4.5. AC-3

O *AC-3* (*Audio Code-3*) (ATSC, 1995) é um formato de áudio proprietário da *Dolby Laboratories Inc* conhecido também pelos seguintes nomes: Dolby Digital, DD e ATSC A/52.

O formato permite codificar de *1* a *5* canais de *20*Hz até *20*kHz e um canal de baixas freqüências de *20*Hz até *120*Hz. *Dolby Digital* utiliza taxas de dados de *32*kbps até *640*kbps, atingindo uma taxa de compressão de aproximadamente *13:1*. As taxas de amostragem possíveis são *32*, *44.1* e *48*kHz.

Um fluxo AC-3 é composto por uma seqüência de quadros. Um quadro AC-3, ilustrado na Figura 29, é formado pelos seguintes campos:

- *Synchronization Information* (SI): inicia com a palavra de sincronismo "*0000 1011 0111 0111*" e contém dados de descrição do quadro, como taxa de amostragem e tamanho do quadro.
- *Bit Stream Information* (BSI): contém parâmetros da codificação das amostras do quadro.
- *Audio blocks* (AB): Seis blocos de áudio, cada um contendo *256* amostras de áudio. Os blocos não são independentes.
- *Auxiliary data* (Aux): campo opcional que contém bits de PAD.
- *CRC*: campo obrigatório para detectar erros.

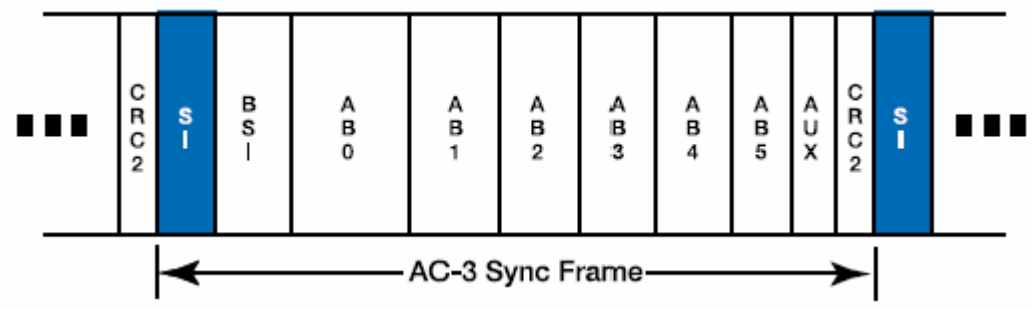

Figura 29 - Quadro do formato AC-3.

O campo *SI* foi mapeado para o campo de *cabeçalho* do formato genérico e os demais campos foram mapeados para a *unidade lógica de dados*.

Contando as amostras dos *6* blocos, cada quadro possui *1536* amostras. Isso significa que o tempo necessário para exibição de um quadro varia entre *32* e *48*ms.

Em um fluxo AC-3 não existe reservatório de bits nem qualquer dependência entre quadros. Conseqüentemente, o mecanismo de ajuste pode ser utilizado de modo similar ao usado no MP2.

3.5. Ajuste Elástico em Fluxos de Sistemas

Alguns formatos de mídia agrupam vários fluxos elementares juntamente com metadados no que se chama de *fluxo de sistemas*. Exemplos de fluxos elementares são fluxos de áudio, vídeo e dados; exemplos de metadados são marcas para manter o sincronismo temporal entre diferentes fluxos elementares.

Para realizar ajuste elástico em fluxos de sistemas pode ser necessário ajustar todos os fluxos elementares e recalcular alguns metadados. Para ajustar um fluxo elementar de áudio, o algoritmo proposto nas subseções anteriores pode ser plenamente utilizado. Para ajustar o vídeo, pode-se utilizar, por exemplo, o algoritmo proposto por Cavendish (Cavendish, 2005). O ajuste de fluxo de dados, quando necessário, é deixado como trabalho futuro.

O modelo do fluxo de sistemas proposto neste trabalho é composto por quadros chamados PACKETS. Um PACKET é formado por um cabeçalho seguido por um conjunto de bits de um fluxo elementar. No cabeçalho, podem existir marcas de tempo (*timestamp*) cujos valores foram atribuídos pelo codificador com base no valor de seu relógio, e outros campos dependentes do formato de mídia. O conjunto de bits de cada PACKET pode conter dados de um ou mais quadros de um fluxo elementar (comum em fluxos de áudio, que possuem quadros pequenos), podendo inclusive conter apenas parte dos dados de um quadro (caso comum para quadros grandes de vídeo). De modo análogo ao caso do fluxo de áudio, é válido destacar que o tamanho dos PACKETs não é necessariamente constante e seu cabeçalho sempre inicia com uma palavra de sincronismo.

As figuras abaixo ilustram exemplos de fluxos de sistemas, cada um com dois fluxos elementares, um de áudio e outro de vídeo. A Figura 30 contém PACKETs de áudio e vídeo uniformemente distribuídos. Entretanto, no caso

geral, existem vários PACKETs de vídeo para um PACKET de áudio, já que é comum que quadros de vídeo possuam muito mais bits do que quadros de áudio. A Figura 31 ilustra um fluxo típico de sistemas.

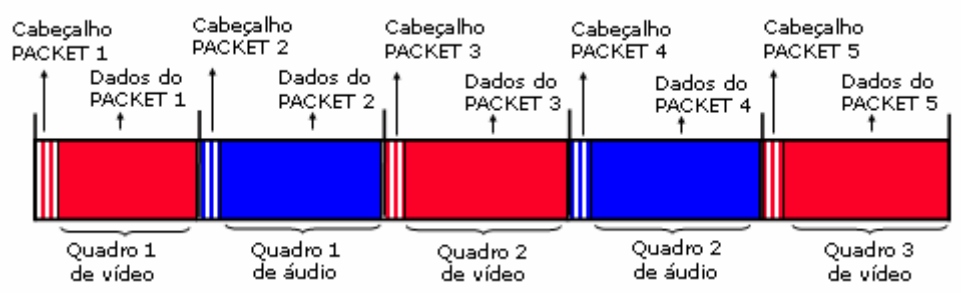

Figura 30 - Fluxo de sistemas com fluxos elementares de áudio e vídeo uniformemente distribuídos.

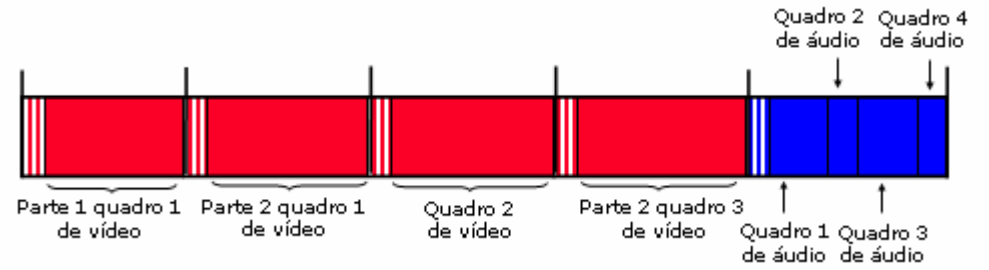

Figura 31 - Fluxo de sistemas com fluxos elementares de áudio e vídeo com distribuição não-uniforme.

Os metadados de sistemas, cabeçalhos dos PACKETS, podem ser tratados como novos atributos dos quadros dos fluxos elementares e devem ser atualizados por um algoritmo de resincronização, que é constituído pelos algoritmos extrator de estruturas de sistemas, de demultiplexação, de conversão de formatos, de controle de fator de ajuste, de verificação do sincronismo intermídia, de multiplexação e de serialização, todos apresentados na Figura 32 e detalhados no decorrer desta seção.

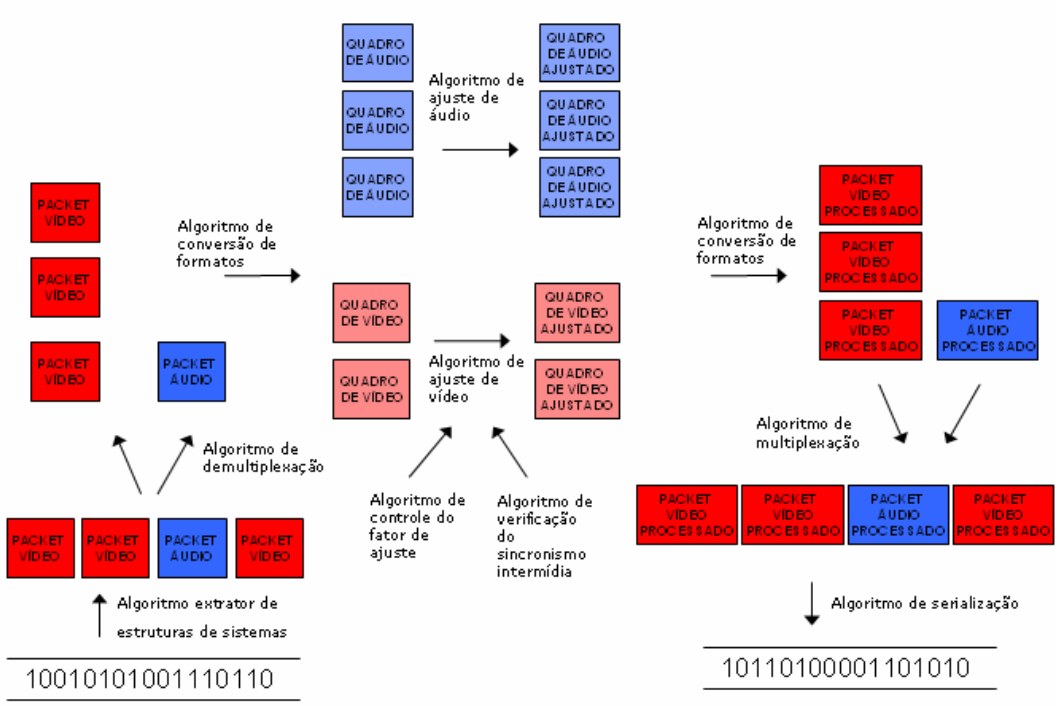

Figura 32 – Conjunto de algoritmos para realizar ajuste elástico em fluxos de sistemas.

O algoritmo extrator de estruturas de sistemas é responsável por reconhecer e criar os PACKETs. O primeiro PACKET do fluxo inicia quando a primeira palavra de sincronismo for encontrada, alinhada em bytes[7]. A partir de então, os bits seguintes ao final desse PACKET podem ser uma nova palavra de sincronismo, caso outro PACKET seja encontrado, ou não, quando o fluxo de sistemas termina. Durante a criação, os PACKETs reconhecidos são numerados com o objetivo de manter a informação da ordem em que estavam no fluxo original.

O próximo passo é chamado de algoritmo de demultiplexação. É necessário separar os PACKETs de cada fluxo para então aplicar o algoritmo de ajuste em cada fluxo elementar. A separação é baseada no fato de cada PACKET identificar univocamente um fluxo elementar.

Depois da demultiplexação, é necessário extrair os dados dos PACKETs para montar os quadros de cada fluxo elementar, sendo esses os dados esperados pelo algoritmo de ajuste do fluxo elementar. Para transformar o modelo de fluxo de sistema no novo modelo de fluxo de áudio - e no de vídeo proposto por

[7] É preciso que exista um tratamento de erros para verificar que a palavra de sincronismo encontrada é válida, ou seja, não faz parte dos dados transportados.

Cavendish (Cavendish, 2005) - é necessário aplicar um algoritmo de conversão de formatos.

A Figura 33 ilustra como o algoritmo deve montar quadros do fluxo elementar a partir dos dados contidos nas estruturas de sistemas. O algoritmo de montagem de quadros de áudio já foi descrito na Seção 3.3. A novidade é que também é necessário criar os metadados, que são os cabeçalhos dos PACKETs do fluxo de sistemas. Entretanto, nem todos os quadros gerados dos fluxos elementares precisam ter metadados, apenas aqueles que contêm o primeiro byte de um PACKET (ver quadro *2* do fluxo elementar na Figura 33). É valido destacar que, após a realização de ajuste, é preciso realizar a conversão inversa, ou seja, montar as estruturas de sistemas a partir dos quadros do fluxo elementar.

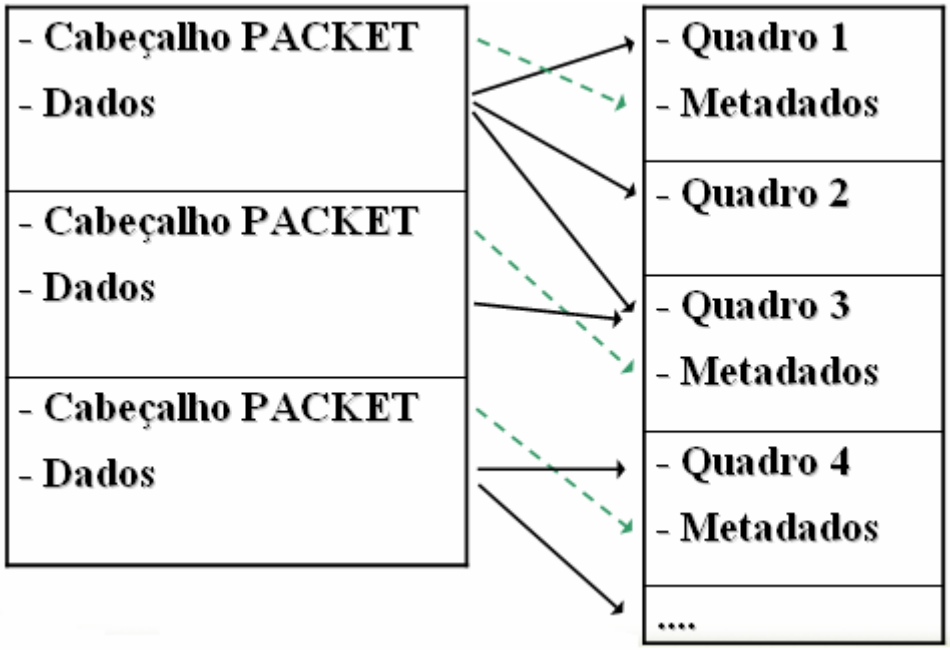

Figura 33 - Transformação de um fluxo de sistemas em um fluxo elementar para aplicar algoritmo de ajuste elástico.

Para o tratamento do fluxo de sistemas, algumas mudanças são necessárias nos algoritmos de ajuste dos fluxos elementares. Em primeiro lugar, a modelagem do fluxo elementar deve conter os metadados de sistemas, como já mencionado, o que afeta a modelagem do fluxo de áudio proposta na Seção 3.2. Cada quadro do fluxo elementar deve poder conter um conjunto de metadados. Além disso, também é necessário armazenar a posição em que os bits desses metadados precisam ser inseridos nos bits do quadro do fluxo elementar no momento da reconstrução dos quadros de sistemas.

Outras mudanças nos algoritmos de ajuste dos fluxos elementares com objetivo de manipular os metadados de sistemas são listadas abaixo:

- Durante a remoção de um quadro, pode ser necessário transferir os metadados contidos de um quadro para o seguinte;
- Durante a inserção de um quadro, pode ser necessário criar estruturas de sistemas (metadados) no novo quadro;
- Para todos os quadros do fluxo resultante é necessário recalcular marcas de tempo presentes nos metadados.

O recálculo das marcas de tempo merece ser mais detalhado. A inserção ou o descarte de quadros invalidam os cálculos realizados no codificador. O algoritmo de ajuste elástico precisa recalcular novos valores para as marcas de tempo do fluxo de mídia acrescentando ou diminuindo o valor da duração de quantos quadros já foram inseridos ou removidos, respectivamente. Somente com essa atualização, os valores das marcas de tempo continuam válidos e o sincronismo intra e intermídia pode ser mantido.

Um outro problema relacionado à transmissão de marcas de tempo é que deve existir um intervalo máximo de tempo entre o envio de duas amostras consecutivas da referência de relógio. Ao realizar ajuste elástico com fator $f>1$, o intervalo máximo, no entanto, pode ser ultrapassado, fazendo com que seja necessário determinar e inserir, no fluxo de sistemas, novas amostras de relógio, através de interpolação entre as existentes. Mais ainda, pode ser necessário criar novas estruturas de sistemas para transmitir as novas amostras de relógio.

A última mudança nos algoritmos de ajustes de fluxos elementares é necessária para viabilizar o algoritmo de controle do fator de ajuste, como explicado a seguir. No algoritmo de ajuste proposto, um fluxo de sistemas é processado por um pequeno trecho (o que equivale a um pequeno intervalo de tempo da mídia) e seguido por um algoritmo de controle de fator. Isso se repete de trecho em trecho até acabar o fluxo. Obviamente, no primeiro trecho o valor do fator de ajuste aplicado em cada fluxo elementar é igual ao fator originalmente aplicado no fluxo de sistemas.

Em cada trecho, os algoritmos de ajuste de fluxos elementares devem executar até acabarem os quadros de fluxos elementares criados a partir dos bits extraídos do fluxo de sistemas. Ao acabar a execução dos ajustes em todos os fluxos elementares presentes naquele trecho do fluxo de sistemas, o algoritmo de

controle de fator é executado. O algoritmo de controle deve analisar o fator obtido por cada fluxo elementar até o momento atual e decidir, sempre tentando aproximar o valor obtido do fator originalmente aplicado ao fluxo de sistemas, qual será o fator de ajuste a ser utilizado no próximo trecho em cada fluxo elementar. A Figura 34 ilustra um fluxo de sistemas sendo divido em fluxos elementares e a delimitação de trechos de processamento.

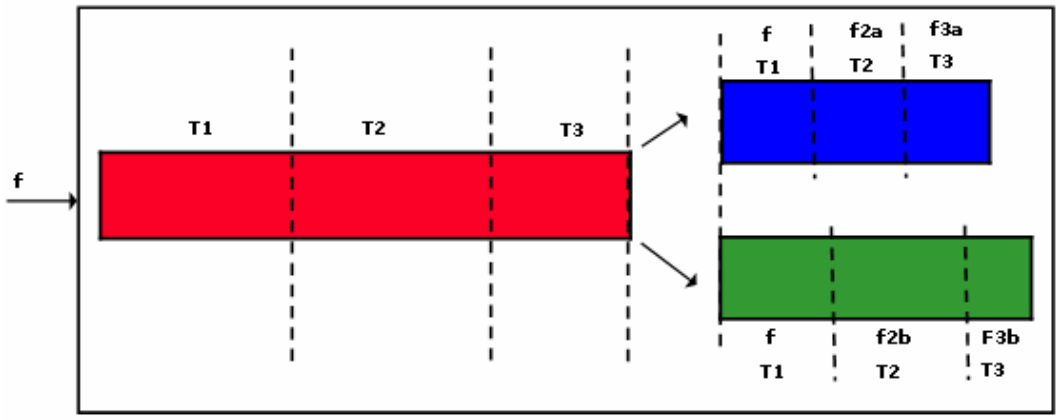

Figura 34 - Representação de divisão dos fluxos elementares em trechos para recálculo de fator de ajuste durante aplicação de ajuste elástico.

Sabendo-se que n é o número de trechos em que o ajuste já foi aplicado, f é o fator de ajuste originalmente aplicado ao fluxo de sistemas e f_{atual} é o fator de ajuste obtido pelo algoritmo de ajuste na parte do fluxo elementar já processada, a seguinte fórmula calcula o fator adequado para se aplicar no novo trecho de cada fluxo elementar com objetivo de convergir f_{atual} para f .

$$f_{i+1} = (n+1) \cdot f - n \cdot f_{atual}$$

Figura 35 - Cálculo do fator de ajuste a ser aplicado no trecho i+1 dos fluxos elementares componentes da mídia original.

Por exemplo, supondo que seja solicitado um ajuste com fator *0.9*, e que nos dois primeiros trechos de um fluxo elementar não tenha sido atingido o fator de *0.9*, mas apenas de *0.95*. Nesse caso, o valor do fator de ajuste calculado para o terceiro trecho desse fluxo elementar deve ser *3*0.9-2*0.95=0.8*. Utilizando um fator de *0.8* no terceiro trecho é possível compensar o erro de não ter atingido o fator *0.9* nos dois primeiros trechos.

O exemplo do parágrafo acima ilustra que, embora o algoritmo de ajuste tenha sido criado pensando em ajustar a mídia em até 10% pode ser necessário

ultrapassar esses valores em determinados trechos da mídia. Vale ainda mencionar que caso ocorra uma solicitação de mudança no fator de ajuste originalmente aplicado, deve-se recalcular o fator de ajuste efetivo do momento da solicitação em diante, desconsiderando os ajustes anteriores.

O algoritmo de controle de fator de ajuste acima, no entanto, não considera que a sincronização intermídia pode ser comprometida. Surge então a necessidade de verificar se o sincronismo intermídia está sendo mantido e, em caso negativo, tomar as providências necessárias. Um possível algoritmo é descrito a seguir:

a) Calcular o instante atingido por cada fluxo elementar até um determinado trecho;

b) Identificar qual o fluxo que está mais adiantado f_1 e qual está mais atrasado f_2 dentre todos os fluxos elementares. Calcular a diferença existente entre os instantes de f_1 e f_2 considerando que não existe ajuste (obtendo-se o valor *dmo*) e considerando que existe (obtendo-se o valor *dmp*). A seguir, calcular o seguinte valor: $dif=|dmo-dmp|$;

c) Verificar se o valor de *dif* é maior do que um determinado valor limite. Se sim, deve-se considerar que o fator aplicado ao fluxo de sistema é o fator do fluxo elementar que está mais atrasado (caso o $f<1$) ou adiantado (caso contrário).

O valor limite mencionado no passo (c) é estabelecido dependendo do quanto se deseja manter o sincronismo intermídia. Segundo Youssef (Aly & Youssef, 2002), diferenças superiores a *160*ms já são perceptíveis aos usuários. Se a condição do passo (c) for verdadeira em algum instante significa que o ajuste elástico não será realizado com o fator originalmente aplicado, mas sim com um fator cujo valor é um número entre o fator original e *1*. Vale ainda observar que é importante que os trechos de processamento sejam pequenos de modo que o dessincronismo intermídia seja pequeno.

Depois de realizar os ajustes nos fluxos elementares e de recriar as estruturas de sistemas, é necessário multiplexar os PACKETs de diferentes fluxos elementares com objetivo de recriar o fluxo de sistemas. A ordem em que os PACKETs devem ser multiplexados obedece ao número de ordem definido durante a criação da estrutura. Por fim, o algoritmo de serialização transforma os PACKETS nos dados binários do fluxo de sistemas ajustado.

3.5.1.
Ajuste Elástico em Fluxo MPEG-2 de Sistemas

Esta subseção descreve a instanciação do algoritmo genérico de ajuste elástico para fluxos de sistemas para a realidade do padrão MPEG-2 sistemas (ISO, 2000a).

A estrutura de um fluxo definida pelo padrão MPEG-2 pode ser visualizada na Figura 36 em dois formatos distintos: o Fluxo de Transporte (TS), que contém um ou mais programas e é apropriado para a transmissão e o armazenamento em ambientes ruidosos onde a ocorrência de erros é freqüente; e o Fluxo de Programa (PS), que contém apenas um programa e é adequado para uso em ambientes com baixas taxas de erros. Cada programa é definido como um conjunto de fluxos elementares que podem ou não ter algum relacionamento temporal entre si. A codificação dos elementos sincronizados entre si utiliza uma mesma base de tempo, ou referência de relógio. A proposta desta dissertação atualmente só manipula Fluxos de Programas, sendo o Fluxo de Transporte deixado como trabalho futuro. O texto a seguir refere-se, assim, sempre ao Fluxo de Programas.

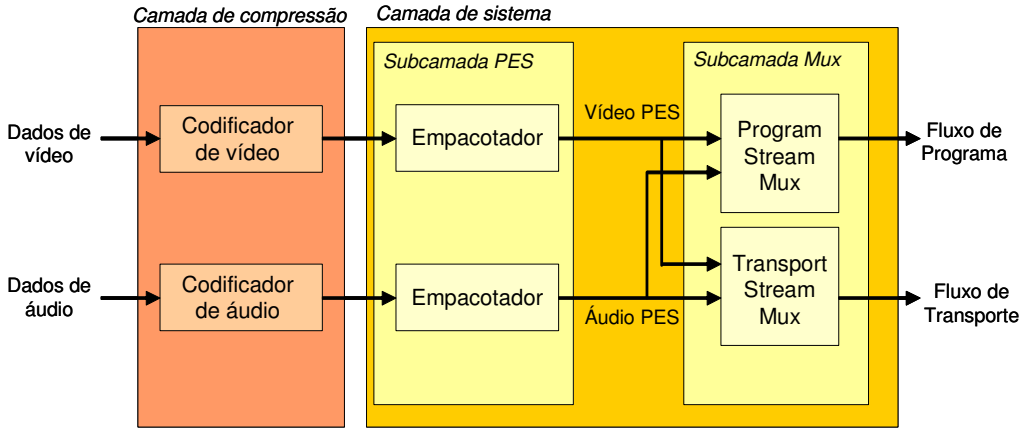

Figura 36 - Estrutura de fluxo MPEG-2 - fonte: (Cavendish, 2005).

Como ilustrado na Figura 36, a estrutura do MPEG-2 está dividida em duas camadas: a camada de compressão e a camada de sistema. A camada de sistema, definida no padrão MPEG-2 sistemas, é responsável pela divisão e encapsulamento de cada fluxo comprimido em pacotes; pela inserção de informações de sincronização entre fluxos de mídias diferentes; pela multiplexação dos fluxos encapsulados; e pelo transporte da informação de referência do relógio utilizado no codificador. A camada de compressão refere-se

à codificação de cada um dos dados audiovisuais, conforme especificado nos padrões MPEG-2 áudio e vídeo.

Os dados individuais de cada mídia, após sofrerem o processo de compressão, são denominados de fluxos elementares e são divididos em PACKETs na subcamada PES (*Packetized Elementary Stream*). O termo PACKET aqui deve ser interpretado como descrito na Seção 3.5. Os PACKETs são empacotados na subcamada de multiplexação em estruturas chamadas PACKs. Um PACK inicia com uma palavra de sincronismo, contém um cabeçalho e é seguido por zero ou mais PACKETS. A Figura 37 ilustra parte de um fluxo de sistemas contendo um PACK com quatro PACKETS e o cabeçalho do próximo PACK.

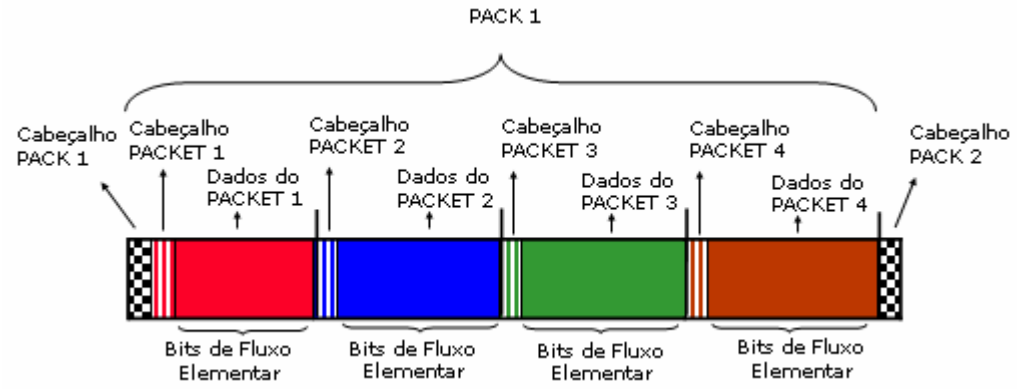

Figura 37 - Representação de parte de um fluxo de sistemas, com PACKs, PACKETs e bits de fluxos elementares.

O algoritmo extrator de estruturas de sistemas, descrito genericamente na Seção 3.5, é adaptado para contemplar a estrutura PACK, utilizando as seguintes regras:

a) Reconhecer e criar o cabeçalho do primeiro PACK. O primeiro PACK do fluxo inicia quando a primeira palavra de sincronismo for encontrada alinhada em bytes.

b) Analisar os bits seguintes ao atual

- Se for a palavra de sincronismo de outro PACK, voltar ao passo (a).
- Se for a palavra de sincronismo de um PACKET, reconhecer a estrutura e voltar ao passo (b).

- Se não for nenhum dos casos acima, indicar que fluxo de sistemas terminou.

Os principais campos do cabeçalho de um PACK para o contexto deste trabalho são: a palavra de sincronismo (cujo valor em hexadecimal é "*0x000001BA*") e o SCR (*System Clock Reference*). Os valores das marcas de tempo SCR significam o instante de tempo em que o último bit desses campos deve entrar no decodificador. Esse campo permite que o decodificador recupere a referência do relógio utilizado pelo codificador. O SCR é divido em duas partes, como indica a fórmula a seguir:

$$SCR(i) = SCR\_base(i) \cdot 300 + SCR\_ext(i)$$

Figura 38 - Cálculo do SCR em fluxos MPEG-2.

As duas partes são transmitidas no cabeçalho do PACK, sendo que a primeira delas (*SCR base*) contém *33* bits e a segunda (*SCR extension*) contém *9*. As fórmulas para calcular tais campos são apresentadas abaixo:

$$SCR\_base(i) = \left[\frac{((system\_clock\_frequency) \cdot t(i))}{300} \right] \% \; 2^{33}$$

Figura 39 - Cálculo do SCR base em fluxos MPEG-2.

$$SCR\_ext(i) = \left[(system\_clock\_frequency) \cdot t(i) \right] \% \; 300$$

Figura 40 - Cálculo do SCR extension em fluxos MPEG-2.

O valor $t(i)$ é um instante de tempo do relógio do codificador, o operador % representa o resto da divisão inteira (comum em linguagens de programação) e o valor do $system\_clock\_frequency$ é dado pela seguinte equação:

$$27000000 - 810 \leq system\_clock\_frequency \leq 27000000 + 810$$

Figura 41 - Intervalo de variação do *system_clock_frequency*.

Este trabalho assume que $system\_clock\_frequency = 27000000$.

Em relação à camada PES, os principais campos para o contexto deste trabalho são listados abaixo:

- *Packet_start_code_prefix*: palavra de sincronismo que possui o valor em hexadecimal igual a "*0x000001*".
- *Stream_id*: especifica o tipo e o número do fluxo elementar. Fluxos de áudio e vídeo possuem o número "*110xxxxx*" e "*1110xxxx*", sendo que a

seqüência de '*x*'s indica o número do fluxo elementar. Outros tipos de fluxos possuem outros números.

- *PES_packet_length*: indica o número de bytes contidos no PACKET contando a partir do final desse campo.
- *PTS_DTS_flags*: indica a presença ou não das marcas de tempo PTS e DTS, explicadas a seguir.
- *PTS (Presentation time stamp)*: marca de tempo de *33* bits que indica o tempo de apresentação de uma unidade de apresentação (imagens, amostras de áudio ou dados) no decodificador e, desse modo, mantém o sincronismo temporal entre diferentes fluxos elementares. O PTS é calculado utilizando a equação a seguir.

$$PTS(i) = \left[\frac{(system\_clock\_frequency) \cdot tp_n(i)}{300} \right] \% \, 2^{33}$$

Figura 42 - Cálculo do PTS em fluxos MPEG-2.

A fórmula acima é bastante parecida com a Figura 39 de calculo do *SCR base*. O *system_clock_frequency* é definido como antes e o $tp_n(i)$ é o instante de apresentação de uma unidade de apresentação presente dentro do PACKET.

- *DTS (Decoding time stamp)*: marca de tempo de *33* bits que indica o instante de decodificação de uma unidade de apresentação no decodificador e é calculada segundo a equação a seguir.

$$DTS(i) = \left[\frac{(system\_clock\_frequency) \cdot td_n(i)}{300} \right] \% \, 2^{33}$$

Figura 43 - Cálculo do DTS em fluxos MPEG-2.

Novamente, a fórmula acima é bastante parecida com a Figura 39 de calculo do *SCR base*. O *system_clock_frequency* é definido como antes e o $td_n(i)$ é o instante de decodificação de uma unidade de apresentação presente dentro do PACKET.

Como anteriormente mencionado, o algoritmo de ajuste elástico proposto deve recalcular as marcas de tempo sendo transmitidas, no caso SCR, PTS e DTS. Uma vez que os instantes de tempo são baseados no relógio do codificador, a

atualização desses valores se dá primeiramente extraindo o valor do instante de tempo original (dado por $t(i)$ no SCR, por $tp_n(i)$ no PTS e por $td_n(i)$ no DTS), acrescentando a este a variação de tempo decorrente do ajuste elástico para enfim gerar as novas marcas de tempo. O intervalo máximo permitido entre duas marcas do relógio consecutivas do SCR é de *0.7* segundos e por isto pode ser necessário inserir amostras de relógio quando o fator *f>1*. Um outro metadado que pode precisar ser alterado pelo algoritmo de ajuste é o campo *PES_packet_length.*

Por fim, vale mencionar que o tamanho das estruturas PACKs e PACKETs não são fixos (lembrando que esta subseção trata apenas de Fluxo de Programas), assim, teoricamente, nunca é preciso duplicar uma estrutura PACKET ao inserir um novo quadro do fluxo elementar. No entanto, no caso de quadros grandes, que são divididos em vários PACKETs, pode ser adequado criar novos PACKETs para transportar os quadros duplicados para não gerar um PACKET de tamanho muito diferente dos existentes no fluxo original.