Pontifícia Universidade Católica
do Rio de Janeiro

Amparito Alexandra Morales Figueroa

Pré-Busca de Conteúdo em Apresentações Multimídia

Dissertação de Mestrado

Dissertação apresentada como requisito parcial para obtenção do título de Mestre pelo Programa de Pós-Graduação em Informática da PUC-Rio.

Orientador: Prof. Luiz Fernando Gomes Soares

Rio de Janeiro

Março de 2014

Amparito Alexandra Morales Figueroa

Pré-Busca de Conteúdo em Apresentações Multimídia

Dissertação apresentada como requisito parcial para a obtenção do grau de Mestre pelo Programa de Pós-graduação em Informática do Departamento de Informática do Centro Técnico Científico da PUC-Rio. Aprovada pela Comissão Examinadora abaixo assinada.

Prof. Luiz Fernando Gomes Soares
Orientador
Departamento de Informática – PUC-Rio

Prof. Marcio Ferreira Moreno
Departamento de Informática – PUC-Rio

Prof. Sérgio Colcher
Departamento de Informática – PUC-Rio

Prof. José Eugenio Leal
Coordenador Setorial do Centro
Técnico Científico – PUC-Rio

Rio de Janeiro, 21 de Março de 2014

Amparito Alexandra Morales Figueroa

Graduou-se em Engenharia em Eletrônica e Telecomunicações na Escola Politécnica do Exército (ESPE - Equador), em dezembro de 2010. Ingressou no programa de mestrado do Departamento de Informática em 2011.

Ficha Catalográfica

Figueroa, Amparito Alexandra Morales

Pré-Busca de Conteúdo em Apresentações Multimídia / Amparito Alexandra Morales Figueroa; orientador: Luiz Fernando Gomes Soares. – 2014.

121 f: il. (color) ; 30 cm

Dissertação (mestrado) – Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Informática, 2014.

Inclui referências bibliográficas.

1. Informática – Teses. 2. Mecanismos e Algoritmos de Pré-busca de conteúdo. 3. Aplicações multimídia. 4. NCL. I. Soares, Luiz Fernando Gomes. II. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Informática. III. Título.

CDD: 004

Agradecimentos

Agradeço a Deus por me dar a força e sabedoria para conseguir com sucesso os objetivos estabelecidos. Agradeço aos meus pais, Amparito e Ricardo, pelo amor e suporte incondicional, por ser essa luz que sempre tem guiado o meu caminho e porque graças a eles sou o que sou e tenho conseguido chegar até aqui. Agradeço ao meu irmão Pablito por ser a alegria da minha vida, por estar sempre ao meu lado me incentivando a continuar e por ter sempre um sorriso e uma palavra de ânimo quando mais preciso. Agradeço a minha amada avó Inés (+) pelo imenso amor que sempre teve para mim, porque ela sempre acreditou em mim e foi a força e o motivo para não desmaiar e culminar essa etapa da minha vida. Agradeço ao meu namorado Gianni Cristian pelo suporte, ânimos e por todo o amor e felicidade proporcionada em minha vida.

Agradeço ao meu orientador, professor Luiz Fernando Gomes Soares, por toda a paciência, dedicação e pelo compartilhamento de sus conhecimentos durante a realização deste trabalho e durante todo o período do mestrado.

Agradeço a todos os membros do laboratório TeleMídia, pela amizade e pela valiosa ajuda prestada para a realização deste trabalho. Agradeço ao Coordenador Márcio Moreno pelo conhecimento que pude adquirir e pelas contribuições essências realizadas durante o desenvolvimento da dissertação.

Agradeço a todos os meus amigos do Departamento de Informática e todos aqueles que tive a oportunidade de conhecer durante esses dois anos, pela amizade sincera e verdadeira que pode encontrar em eles, por todo o apoio oferecido, por compartilhar comigo momentos felizes e também por estar comigo em um dos momentos mais tristes e difíceis da minha vida.

Agradeço à PUC-Rio, pelo fornecimento de recursos e infraestrutura que viabilizaram a construção deste trabalho. Finalmente, agradeço à CAPES pelo apoio financeiro concedido.

Resumo

Figueroa, Amparito Alexandra Morales; Soares, Luiz Fernando Gomes. **Pré-Busca de Conteúdo em Apresentações Multimídia.** Rio de Janeiro, 2014. 121p. Dissertação de Mestrado - Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

Quando entregamos e apresentamos aplicações multimídia por meio de uma rede de comunicação, a latência de exibição pode representar um fator central e crítico que afeta a qualidade da apresentação multimídia. Na entrega de uma apresentação multimídia de boa qualidade o sincronismo é prevalecido, consequentemente, os conteúdos são exibidos de forma contínua, conforme as especificações do autor da aplicação. Nesta tese, um plano de pré-busca de conteúdos multimídia é proposto com o intuito de reduzir a latência de exibição e garantir o sincronismo entre os objetos de mídia que fazem parte da apresentação multimídia. O mecanismo proposto considera as aplicações multimídia desenvolvidas na linguagem declarativa NCL e utiliza a vantagem do sincronismo estar baseado em eventos, na determinação da ordem adequada de recuperação dos diferentes objetos de mídia e no cálculo dos seus tempos de início de recuperação. Aspectos importantes a serem considerados em um ambiente de pré-busca são levantados e os diferentes algoritmos que compõem o plano de pré-busca são desenvolvidos.

Palavras-chave

Apresentações multimídia; algoritmos de pré-busca; recuperação de conteúdo; sincronismo; NCL; algoritmos heurísticos; latência de exibição; ordenamento de objetos; agendamento de objetos.

Abstract

Figueroa, Amparito Alexandra Morales; Soares, Luiz Fernando Gomes (Advisor). **Prefetching Content in Multimedia Presentations.** Rio de Janeiro, 2014. 121p. MSc. Dissertation- Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

When delivering and presenting multimedia applications through a communication network, the presentation lag could be a major and critical factor affecting the multimedia presentation quality. In a good quality presentation the synchronism is always preserved, hence all the contents are presented in a continue way according to the authoring specifications. In this dissertation, a multimedia content prefetching plan is proposed in order to minimize the presentation lag and guarantee the synchronism between the media objects, which constitute the multimedia application. The proposed mechanism regards the multimedia applications developed using the NCL declarative language and it uses the events based synchronism advantage to determinate the ideal retrieval order of the media objects and to calculate their start retrieval times. Furthermore, important issues to be considered in a prefetch ambient are raised and the different algorithms that belong to the prefetching plan are developed.

Keywords

Multimedia applications; prefetching algorithms; multimedia content retrieval; synchronism; NCL; heuristic algorithms; presentation lags; ordering objects; scheduling objects.

Sumário

Lista de figuras

Lista de tabelas

Lista de algoritmos

1
Introdução

A preservação do sincronismo em apresentações multimídia está entre os principais requisitos de QoS (*Quality of Service*) de um sistema multimídia. Uma aplicação multimídia pode conter objetos de mídia de diferentes tipos, como texto, imagem, vídeo, áudio, etc. Em comum, esses objetos devem obedecer às relações de sincronismo especificadas pelo autor, tanto em relação ao tempo de exibição, quanto ao espaço de exibição nos dispositivos utilizados para apresentação (Blakowski & Steinmetz 1996).

Naturalmente, para realizar a apresentação de uma aplicação multimídia, os conteúdos dos objetos de mídia devem ser recuperados a partir do dispositivo em que estão armazenados. A localização desses dispositivos pode ser remota ou local. No entanto, o tempo de recuperação dos conteúdos em discussão não varia apenas em função da localização e tipo do dispositivo de armazenamento, mas também de acordo com o tipo do conteúdo e seu tamanho. Uma vez que os instantes de apresentação dos objetos de mídia são definidos, o tempo de recuperação dos objetos podem produzir latências de exibição[1], que causam a perda do sincronismo especificado na aplicação. Nesse caso, existe não só a possibilidade de distorcer o significado da apresentação, originalmente vislumbrada pelo autor, como, também, de causar o desinteresse do usuário da aplicação, que pode chegar a experimentar tempos excessivos de espera para a apresentação de um conteúdo (Rodrigues & Gomes Soares 2002).

A solução comumente utilizada para o problema de perda de sincronismo, devido a retardos inseridos pelo processo de recuperação de conteúdo, consiste na aplicação de algoritmos de pré-busca nos sistemas de apresentação de documentos multimídia. Os algoritmos de pré-busca são, geralmente, encarregados de determinar um tempo próximo ao tempo ótimo (heurístico) para iniciar a recuperação antecipada dos conteúdos dos objetos que compõem uma aplicação,

[1] Latência de exibição refere-se ao intervalo de tempo entre o instante em que o conteúdo de um objeto deve ser exibido e o instante em que o mesmo é efetivamente apresentado.

com o intuito de tentar preservar o sincronismo especificado na aplicação.

Os principais trabalhos existentes na literatura, que serão discutidos no Capítulo 2, incluem, em suas soluções de pré-busca, algumas técnicas de ajuste elástico do tempo de exibição de conteúdo, que podem ser usadas para reduzir as interrupções de exibição produzidas durante a apresentação da aplicação. Ao definir os instantes de recuperação de cada conteúdo dos objetos, alguns trabalhos realizam ainda um procedimento para reportar qual é a capacidade de armazenamento necessária à plataforma.

Um ponto crítico, detectado nos trabalhos mencionados, consiste na forma em que o plano para realizar a recuperação de conteúdo é projetado (plano de pré-busca). São levados em consideração apenas intervalos absolutos em um eixo temporal (ISO/IEC 2000) para especificar o sincronismo espaço-temporal dos diferentes objetos de mídia que compõem uma aplicação, o que limita seu escopo, principalmente na ocorrência de eventos imprevisíveis. Com uma abordagem diferente, este trabalho propõe a construção de um plano de pré-busca a partir de aplicações onde o sincronismo depende da ocorrência de eventos com duração variável ou mesmo imprevisível no momento da autoria. Nessas aplicações, é imperativo que a especificação do sincronismo seja realizada de forma relativa à ocorrência desses eventos (independente do momento temporal em que eles ocorrem, e se de fato eles irão ocorrer). Esse é o caso, por exemplo, das aplicações interativas, onde é impossível prever o momento exato da interação do usuário. Quando o sincronismo é especificado de forma relativa, os instantes temporais de ocorrência dos eventos somente serão conhecidos na fase de execução da aplicação (Moreno 2008).

Como prova de conceito, o plano proposto será gerado a partir de aplicações especificadas na linguagem NCL[2]. Na verdade, serão utilizados como dados de entrada uma compilação de aplicações NCL. Ou seja, a partir de um documento NCL será construída uma estrutura de dados contendo, além dos relacionamentos temporais entre os objetos de mídia, as suas especificações de apresentação necessárias para projetar o plano de pré-busca.

Para validar as propostas apresentadas, serão realizadas medições que indicam os benefícios dos algoritmos desenvolvidos em comparação aos

[2] Guia de Referência Rápida NCL Perfil EDTV, *Handbook*, http://handbook.ncl.org.br.

existentes.

Um algoritmo trivial de pré-busca consiste na recuperação prévia de todo o conteúdo referenciado pela aplicação multimídia. Além de poder causar retardo para o início da apresentação da aplicação, essa abordagem considera que a capacidade de armazenamento da plataforma é infinita. Por sua parte, este trabalho propõe o desenvolvimento de algoritmos de pré-busca que, além de reduzir a latência de exibição em aplicações multimídia, também calculam a capacidade máxima de armazenamento requerida pela aplicação.

1.1 Motivação

Em um sistema multimídia, a sincronização entre os diferentes objetos de mídia, que compõem uma aplicação, é um fator que determina a sua qualidade de apresentação. A apresentação dos conteúdos de uma aplicação de maneira contínua, sem a presença de interrupções ou se isso não for possível, sempre visando a entrega de uma apresentação com a menor quantidade de interrupções, é o critério/objetivo de qualidade.

Para que esse objetivo seja alcançado, é fundamental conhecer os requisitos de sincronização do usuário, isso é a percepção do usuário frente aos tempos de espera para a exibição do conteúdo. A qualidade da percepção do usuário final no contexto de aplicações multimídia é frequentemente caracterizada pelos tempos de espera inseridos antes ou durante a exibição do conteúdo na tela.

Em particular, em um ambiente, onde as condições da rede de comunicação não são boas e a plataforma de exibição possui limitações referentes à capacidade de armazenamento, é imprescindível utilizar mecanismos de pré-busca para a recuperação prévia do conteúdo, a fim de reduzir os tempos de espera para a sua exibição. Esses tempos de espera podem ocorrer: (a) antes da apresentação multimídia, denotado por *retardo inicial*, neste trabalho, e (b) durante a apresentação multimídia (Hossfeld et al. 2012). A latência na exibição é um dos parâmetros para medir a qualidade de experiência (Quality of Experience – QoE)[3] percebida pelo usuário final em uma aplicação multimídia. Em (Hossfeld et al. 2012), foi concluído que os usuários são extremamente sensíveis ás interrupções e

[3] Qualidade de Experiência (QoE) é uma medida subjetiva da experiência do usuário com relação a algum serviço (navegador web, ligação telefónica, transmissão de TV, IPTV, etc.).

que essas devem ser evitadas em qualquer caso, mesmo à custa de aumentar o *retardo inicial* para recuperar a informação e colocá-la no *buffer*.

No entanto, no contexto de sistemas de aplicações multimídia interativas, os tempos de resposta ou tempos de espera contribuem na perda de qualidade percebida pelo usuário e acrescentam uma sensação de não naturalidade. Além disso, esses tempos não podem ser muito altos para assim evitar que o usuário desista da apresentação. Em conclusão, a finalidade é manter uma boa qualidade nas apresentações multimídia, e, para obter essa qualidade, as latências de exibição devem ser controladas e reduzidas.

1.2 Objetivos

O objetivo deste trabalho é construir um plano de pré-busca a fim de reduzir a latência de exibição em apresentações multimídia. Esse plano contém os tempos de início para a recuperação antecipada dos conteúdos dos objetos de mídia, com o intuito de apresentar esses conteúdos nos instantes de tempo especificados na autoria.

O foco principal do trabalho é considerar aplicações multimídia onde o sincronismo está baseado na ocorrência de eventos. Somente são consideradas as aplicações que possuem eventos determinísticos. Os eventos determinísticos são aqueles onde o desenvolvedor da aplicação tem o controle da apresentação dos objetos multimídia (Rodrigues & Gomes Soares 2002).

Em plataformas computacionais com recursos limitados, torna-se importante o gerenciamento dos recursos do sistema. Este trabalho tem o objetivo de determinar como o cálculo do tamanho máximo necessário do *buffer* de armazenamento (para armazenar os conteúdos dos objetos de mídia que vão ser adquiridos) deve ser realizado.

1.3 Organização

Esta dissertação está estruturada da seguinte forma. O Capítulo 2 apresenta os principais trabalhos relacionados, destacando suas limitações e soluções. O Capítulo 3 analisa as características de um ambiente de pré-busca e, baseado

nessas características, é descrito um cenário em que a utilização de algoritmos de pré-busca torna-se importante. O Capítulo 4 discorre sobre os parâmetros de entrada para construção do plano de pré-busca. O Capítulo 5 analisa e descreve o algoritmo de pré-busca desenvolvido. O Capítulo 6 propõe uma estratégia para realizar a pré-busca com eventos não determinísticos. Finalmente, o Capítulo 7 apresenta as conclusões obtidas e os trabalhos futuros.

2
Trabalhos Relacionados

S.W. Kim et al. (Kim et al. 2001) desenvolveram um mecanismo de pré-busca em que as apresentações multimídia são gerenciadas por um módulo denominado MPC (*Multimedia Presentation Coordinator*). O MPC determina um tempo máximo de recuperação (MRT – *Maximum Retrieval Time*) para cada objeto de mídia especificado na aplicação multimídia. O MRT consiste no tempo máximo necessário para carregar o conteúdo (ou um segmento do conteúdo) de um objeto de mídia em um *buffer*.

O trabalho de Kim considera que os conteúdos dos objetos de mídia encontram-se localizados em um único dispositivo de armazenamento local, propondo um método para calcular os MRTs de acordo com as características desse dispositivo e de acordo com um atraso máximo tolerado. Esse atraso máximo depende se o conteúdo em questão consiste em uma mídia contínua ou discreta (Candan et al. 1998; Steinmetz 1996).

O MPC é responsável pela construção de um plano de pré-busca para que a recuperação do conteúdo de cada objeto seja concluída antes de sua respectiva exibição. A prioridade do MPC é garantir que a apresentação dos objetos de mídia seja realizada sem interrupções (*gaps*). Para isso, o instante de início de apresentação (PST – *Presentation Start Time*) de cada objeto e a duração de sua apresentação (PT – *Presentation Time*) podem ser alterados. Além disso, o algoritmo para construção do plano de pré-busca pode retardar o início da apresentação da aplicação multimídia para evitar a ocorrência de gaps.

A estrutura de dados utilizada como entrada para a construção do plano de pré-busca, leva em consideração a divisão do conteúdo dos objetos de mídia em segmentos mínimos que satisfazem, entre si, a condição de igualdade (*equals*) de Allen[4] (Allen 1983).

[4] Conforme as relações temporais definidas por Allen (1983), uma relação de igualdade entre dois objetos de mídia existe se a apresentação de ambos é iniciada no mesmo instante.

Esses segmentos são agrupados, dependendo da relação temporal entre eles, em blocos chamados blocos de sincronização, como é mostrado na Figura 1.

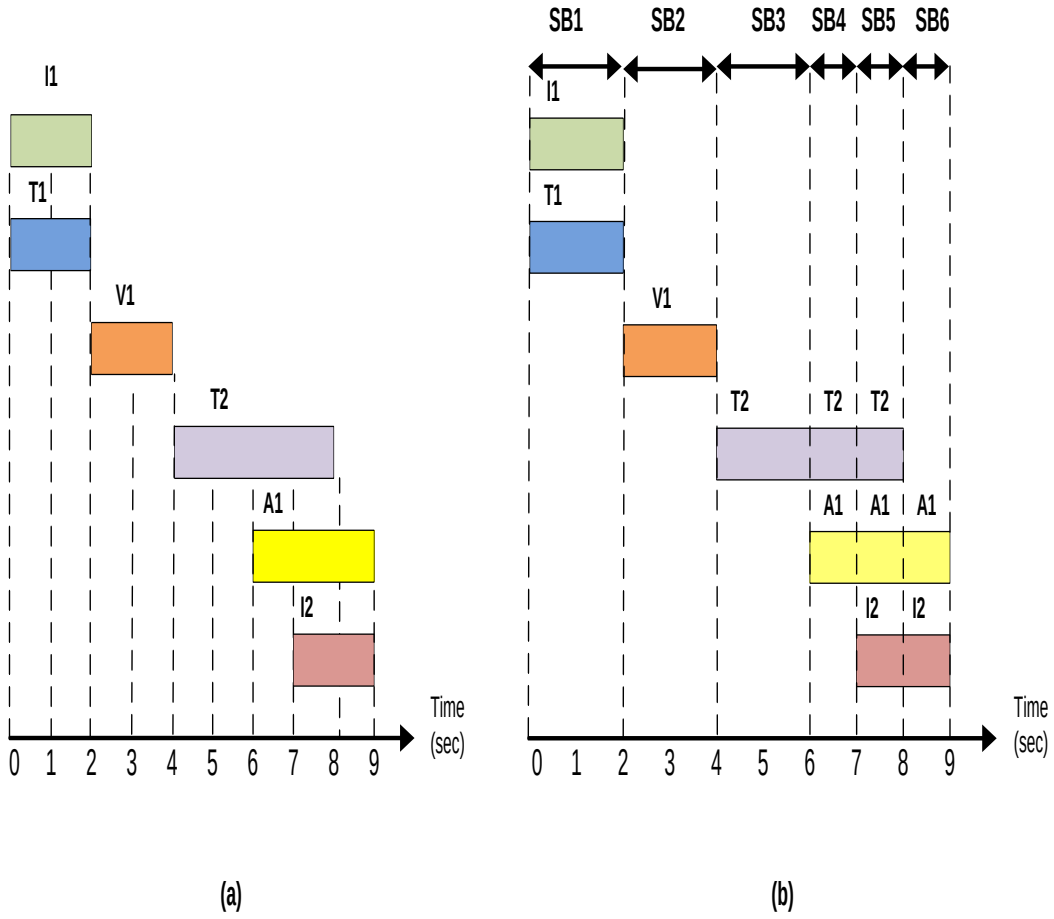


Figura 1 – Expressão absoluta. (a) dos objetos de mídia (b) dos blocos de sincronização.

Cada bloco contém a informação correspondente ao seu PST e PT, assim como os segmentos de mídia que o compõem. Essa estrutura é denominada Tabela de Sincronização de Blocos (Tabela 1).

Resumidamente, no algoritmo usado para calcular os instantes de tempo ótimos para iniciar a recuperação (ORSTs – *Optimal Retrieval Start Time*) dos conteúdos dos objetos de mídia, um ORST inicial é definido com o valor 0 (instante inicial de processamento). A soma do valor do ORST atual com o MRT de um objeto de mídia é atribuída ao ORST do próximo objeto de mídia. Cabe notar que se um objeto de mídia contínua for dividido, o seu respectivo MRT é novamente calculado (Figura 2).

Synchronization Block	PST (s)	PT (s)	Media Objects
SB1	0,0	2,0	T1 and T1
SB2	2,0	2,0	V1
SB3	4,0	2,0	T2
SB4	6,0	1,0	T2 and A1
SB5	7,0	2,0	T2, A1 and I2
SB6	8,0	1,0	A1 and I2

Tabela 1 - Tabela de Sincronização de Blocos.

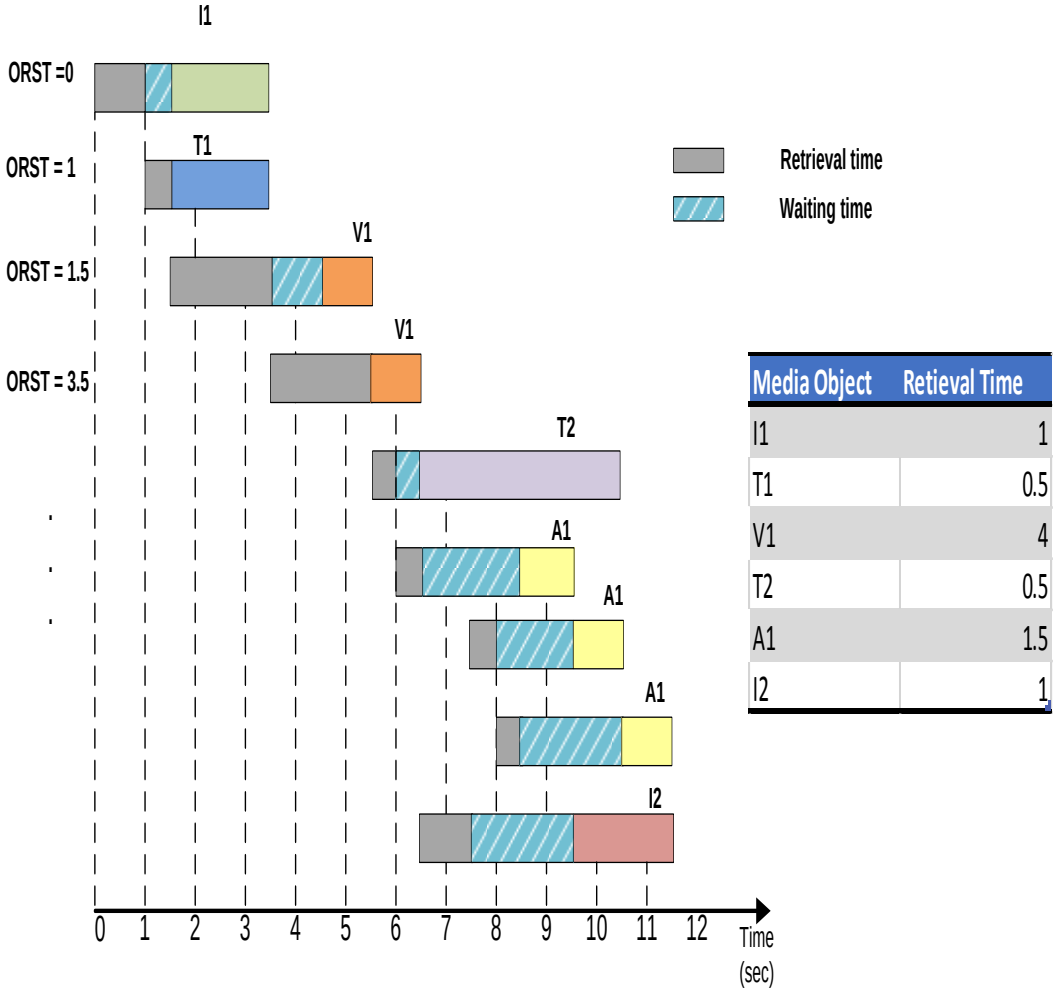
A segmentação de conteúdo dos objetos de mídia contínua torna desnecessária a recuperação de todo seu conteúdo para que sua exibição seja iniciada. Em consequência a recuperação dos objetos de mídia discreta possui prioridade maior sobre os objetos de mídia contínua.

Por outro lado, devido ao fato que os objetos de mídia contínua diferem dos discretos em suas características de saída, como a taxa de consumo, que é definida como a quantidade de dados que estão sendo exibidos por segundo, (Kim et al. 2001), existe a possibilidade de quebrar os objetos de mídia contínua, que não se encontrem sobrepostos no tempo a nenhum outro objeto, em vários objetos de mídia independentes, conforme suas taxas de consumo, a fim de reduzir mais ainda sua latência de exibição.

Cabe mencionar que a quebra dos objetos de mídia contínua antes mencionada nem sempre pode ser feita em qualquer ponto, como é proposto naquele trabalho. Deve-se levar em conta que alguns protocolos definem como um arquivo pode ser dividido, para que o cliente escolha a partir de qual segmento o conteúdo será adquirido.

Durante a execução, existe um monitor que compara o tempo de pré-busca de cada objeto com o tempo de pré-busca previsto no plano. Se esse tempo ultrapassa o limite previsto, o MPC executa um mecanismo de agendamento imediato para recalcular os tempos de duração de apresentação dos conteúdos a fim de manter a sincronização dos segmentos. Quando necessário, o algoritmo seleciona um bloco de sincronização para ser sacrificado. Uma vez que o usuário é mais sensível ao ajuste de tempos nos objetos de mídia contínua do que nas mídias discretas, o primeiro bloco de sincronização que contiver unicamente conteúdos de mídias estáticas será escolhido, tendo as durações desses conteúdos

reduzidas. Cabe notar que, para o cálculo do MRT, é considerado um parâmetro de QoS que corresponde ao máximo atraso *end-to-end* permitido conforme o tipo de mídia. Isso significa que é considerado o pior caso. Assim, conforme mencionado acima, esse mecanismo de agendamento imediato é desnecessário.



Media Object	Retieval Time
I1	1
T1	0.5
V1	4
T2	0.5
A1	1.5
I2	1

Figura 2 - Apresentação multimídia com mecanismo de pré-busca.

Como é muito difícil calcular a taxa de consumo atual do *buffer* para os objetos de mídia contínua, devido aos vários modos existentes de armazenar os dados e às diferentes taxas de compressão dos mesmos, esses dados devem ser guardados no *buffer* até que toda sua apresentação seja completada. S.W. Kim et al. sugerem determinar os *"Buffer Releasing Times"* (BRTs), indicando os tempos para a liberação dos dados.

A proposta de contar com o MPC para a orquestração e controle das aplicações multimídia é uma solução interessante, porque apresenta de maneira

prática vários requisitos que devem ser considerados na elaboração de uma estratégia de pré-busca. No entanto, o trabalho de Kim somente considera a recuperação dos objetos de mídia a partir de um único dispositivo de armazenamento e de maneira contígua, não sendo abordados aspectos referentes a transmissões na rede.

Além disso, existem algumas deficiências nos mecanismos de ajuste de tempo de exibição de conteúdo, utilizados tanto em tempo de compilação como em tempo de exibição. Em tempo de compilação, o ajuste dos tempos das mídias estática não levam em conta o futuro da apresentação. Em tempo de apresentação, um mecanismo é disparado quando o MPC detecta que o MRT de um objeto é maior do que o esperado. No entanto, esse mecanismo corrige o plano de apresentação, mas não altera o plano de pré-busca.

Quando um objeto de mídia é recuperado antes do MRT planejado, os ORSTs de todos os objetos de mídia são reorganizados e trazidos para frente. Dessa forma, o plano de apresentação é modificado, diminuindo os *gaps*. Porém, esse tempo ganho poderia ser considerado como uma reserva, em caso de um atraso futuro, ou seja, nesse caso, seria conveniente não modificar o plano. Isso se justificaria principalmente quando não há *gaps* na apresentação.

In-Ho Lin et al. (In-Ho Lin & Wu 2001) estabelecem um modelo orientado a objetos que especifica e controla as relações temporais de todos os objetos de mídia na plataforma de apresentação. Além disso, o trabalho discute os problemas relacionados com a alocação de recursos do sistema. A finalidade principal desse trabalho é solucionar os possíveis problemas relacionados com o retardo produzido no nível de apresentação, investigando os requisitos mínimos de *buffer*, assim como os tempos mínimos para a pré-busca dos conteúdos.

Da mesma forma que Kim (Kim et al. 2001), para facilitar a sincronização intermídia, o trabalho propõe a divisão da aplicação multimídia em *grupos de apresentação*. Esses grupos são construídos respeitando as relações temporais de Allen (Allen 1983) e considerando a sobreposição temporal dos conteúdos dos objetos de mídia. Quando um objeto de mídia discreto se sobrepõe no tempo a um objeto de mídia contínuo, se diz que existe um ponto de separação (SP), que facilita a partição de um objeto de mídia discreto sem causar nenhuma alteração na apresentação. Pontos de separação delimitam os grupos de apresentação encontrados na aplicação, onde cada grupo possui um tempo de início e fim de

grupo.

Com a finalidade de conseguir uma sincronização temporal entre grupos, o tempo de início de reprodução de um grupo pode ser tratado como um ponto de sincronização com outros grupos. A solução define que todos os conteúdos dos objetos de mídia pertencentes a um grupo devem estar disponíveis, de alguma forma, para serem apresentados, antes que o tempo de início de grupo seja ativado. Assim, um grupo pode ser tratado como uma unidade independente de apresentação.

É importante considerar que o tamanho dos grupos de apresentação tem relação com a utilização dos recursos do sistema (tamanho do *buffer*, banda passante etc.). Em consequência, um grupo de apresentação pequeno precisará de menor quantidade de recursos que um grupo maior, e a qualidade da apresentação também será melhor.

In-Ho Lin et al. também apresentam os requisitos básicos para a ótima recuperação dos conteúdos dos objetos de mídia, a fim de facilitar sua reprodução com um certo nível de QoS. Principalmente, calcula a capacidade mínima do *buffer* que será requerida para o armazenamento dos conteúdos de um determinado grupo de sincronização em um instante de tempo específico. Consideram que os conteúdos são recuperados a través de uma banda passante constante previamente alocada, além de assumirem que a taxa de consumo dos dados é maior do que a banda passante. Em consequência, deve-se garantir que o conteúdo que se encontra armazenado no *buffer* seja sempre suficiente, para evitar um *underflow*.

Para realizar os cálculos mencionados, In-Ho Lin et al. definem uma série de fórmulas matemáticas para encontrar o tempo mínimo e o tempo máximo de início da recuperação do conteúdo, levando em consideração tanto a quantidade total de dados que são transmitidos, como a quantidade de dados que são consumidos na plataforma de apresentação em um determinado instante de tempo. A capacidade total do *buffer* que está sendo ocupada nesse instante de tempo é dada pela diferença dos dados transmitidos e os dados consumidos.

Em particular, os resultados alcançados naquele trabalho mostram que uma rede que apresente uma banda passante menor precisa uma maior capacidade no *buffer* e os tempos de recuperação dos conteúdos devem iniciar o mais cedo possível. No entanto, não é definido um método para a construção de um plano de

pré-busca; apenas são analisados os requisitos do sistema que deveriam ser levados em conta para construí-lo.

Rodrigues et al. (Rodrigues & Gomes Soares 2002) levantam os requisitos para a implementação de mecanismos de pré-busca em formatadores hipermídia[5], e a partir desses requisitos, propõem uma arquitetura genérica que possa ser reusada no desenvolvimento de formatadores específicos. Esses requisitos consideram: aplicações multimídia que possuem objetos de mídia contínua, os quais encontram-se armazenados de maneira distribuída. Essas aplicações podem conter tanto eventos determinísticos como não determinísticos. Além disso, considera que os recursos da plataforma de exibição, tais como banda passante da rede, CPU e memória, são limitados, variam ao longo do tempo e são compartilhados estatisticamente.

Esse trabalho propõe a construção de um plano de pré-busca formado por duas estruturas principais: O Compilador do Orquestrador de Pré-Busca (OPBC) e O Executor do Orquestrador de Pré-Busca (OPBE). O OPBC é o encarregado de decidir a ordem/ prioridade das buscas e o percentual de cada objeto a ser trazido antes do início da exibição. Uma vez executado o OPBC, o OPBE consulta o plano e dispara nos instantes programados requisições aos controladores de exibição das ferramentas de exibição para iniciar a preparação para exibição de cada um dos objetos.

Com o intuito de diminuir a latência na apresentação dos objetos não determinísticos, é importante que o compilador da apresentação (ou mesmo o executor) informe também as possibilidades de exibição dos objetos. Desse modo, sempre que houver espaço em memória (ou mesmo em um disco local) e capacidade de processamento disponíveis no cliente, o plano de pré-busca pode tentar trazer o conteúdo dos objetos com eventos não determinísticos de maior probabilidade.

O gerenciamento do *buffer* é outra responsabilidade que é atribuída ao mecanismo de pré-busca. Para manter a exibição contínua do conteúdo multimídia deve ser evitada a ausência de dados no buffer (*buffer underflow*). Portanto, as taxas de aquisição e consumo de cada um dos objetos de mídia precisam ser conhecidas. Evidentemente, pode ser que os dados estejam comprimidos e a taxa

[5] Nome dado aos sistemas responsáveis por controlar a exibição de documentos em clientes hipermídia.

de consumo não seja constante, além disso a taxa de aquisição pode variar dependendo das condições da rede de comunicação.

Com o intuito de conseguir uma alocação eficiente dos recursos, algumas informações importantes referentes aos objetos de mídia devem ser passadas ao módulo de pré-busca. Essas informações estão relacionadas o tamanho (ou a duração) dos objetos, a quantidade mínima de dados que deve existir no *buffer* antes do início da apresentação e a latência máxima permitida para a sua exibição.

Além das informações a respeito de cada objeto, é fundamental que o mecanismo de pré-busca tenha acesso à descrição da plataforma de exibição. É importante que essa descrição contenha dados tais como banda passante disponível, espaço livre para o armazenamento dos objetos e outros parâmetros de desempenho de QoS.

Embora foi proposto um framework para auxiliar no projeto e na implementação de mecanismos de pré-busca em sistemas de apresentação hipermídia, aspectos relacionados com as técnicas e algoritmos heurísticos para a construção do plano de pré-busca em si, não foram tratados.

Feng-Cheng Lin et al. (Lin et al. 2008) apresentam algumas das técnicas utilizadas para a redução da latência de exibição em documentos multimídia gerados dinamicamente a partir de objetos de mídia recuperados de livrarias digitais multimídia. A tecnologia de *streaming* de mídia é um exemplo de pré-busca, e consiste em executar a pré-busca de segmentos mínimos dos objetos e colocá-los em um *buffer* que se encontra no lado do cliente. Assim que o *buffer* estiver cheio, o objeto de mídia inicia a sua reprodução. Enquanto o objeto de mídia está sendo apresentado, mais dados são recuperados.

Em um serviço de *media streaming* a través de uma rede com baixo retardo, o único tempo de espera é o retardo inicial do objeto de mídia, chamado retardo de *"pre-roll"*, onde a latência de exibição é próxima a zero. Contudo, tem que ser considerado o fato que nem todos os objetos podem ser apresentados usando a tecnologia de *streaming*. Em contrapartida, a ordem de recuperação dos objetos de mídia tem um impacto importante no atraso total inserido na apresentação. (Lin et al. 2008) apresentam diferentes algoritmos que otimizam uma sequência de objetos de mídia, dependendo do tipo de restrição que esses possuem.

Existem vários critérios gulosos para estabelecer a ordem na qual os objetos de mídia podem ser recuperados, entre eles encontram-se: "*Earliest Due Date*

First" (EDD) que considera os objetos de mídia em ordem crescente com respeito ao seu tempo de início de apresentação, "*Shortest Processing Time First*" que considera os objetos de mídia em ordem crescente com respeito a seu tempo de processamento (tempo de apresentação) e o "*Shortest Retrieval Time First*", que considera os objetos de mídia em ordem crescente com respeito a seu tempo de recuperação. Os métodos gulosos acima mencionados foram propostos por Kleinberg & Tardos (2005).

No trabalho apresentado por Lin é considerado um cenário onde a banda passante é constante e não existe nenhuma limitação em relação à capacidade do *buffer* do cliente para armazenar os conteúdos dos objetos de mídia recuperados. Assim, a fim de encontrar uma sequência de objetos de mídia que minimize o tempo total de espera para a reprodução dos dados, é possível aplicar a regra de Johnson apresentada em S.M. Johnson (Johnson 1954). De forma intuitiva, para reduzir o tempo de espera, o objeto de mídia que possui o menor tempo de transmissão é escolhido para ser recuperado primeiro.

A regra de Johnson pode ser aplicada diretamente em problemas que não tem nenhuma classe de restrição em relação à ordem de exibição dos objetos de mídia, mas em cenários mais complexos essa regra não pode ser aplicada de modo direito e a utilização de algoritmos mais sofisticados é requerida.

Quando as aplicações multimídia apresentam várias restrições como relações temporais entre os objetos de mídia, prioridade dos objetos, tempo de termino dos objetos, limitações na capacidade do buffer do cliente etc., o problema de agendamento dos objetos de mídia pode ser modelado como um problema do tipo "*two-machine flowshop*". Em um problema de *"two-machine flowshop",* um conjunto de trabalhos independentes são agendados para serem processados em duas máquinas diferentes. Esse problema é tratado em (Reisman et al. 1997), (Hejazi*y & Saghafianz 2004), e (Pinedo 2002). O problema de "*two-machine flowshop*", modelado considerando essas restrições mencionadas acima, é apresentado em (Lin et al. 2013; Lin, Hong, et al. 2009). Usualmente esses problemas têm uma complexidade NP-Difícil. Como a taxa de transmissão é constante, então o tempo de recuperação de um objeto de mídia é proporcional a seu tamanho. Portanto, o número de objetos permitidos dentro do *buffer* depende do tempo de processamento. Está comprovado que esse problema é NP-Difícil.

Em conclusão, Feng-Cheng Lin et al. (Lin et al. 2008) apresentam um

método para obter a sequência ótima para a recuperação dos objetos de mídia que pertencem uma aplicação multimídia a fim de reduzir a latência de exibição. Quando esses objetos não possuem restrições temporais e o *buffer* é considerado como infinito, é possível resolver o problema em tempo polinomial. Caso contrário, o problema torna-se NP-Difícil, onde é necessário o uso de algoritmos heurísticos para obter uma solução próxima da ótima, como é mostrado em (Lin, Lai, et al. 2009).

2.1
Observações sobre os trabalhos relacionados

1. A quebra de um objeto de mídia contínua nem sempre pode ser feita em qualquer ponto, o protocolo utilizado para a entrega do conteúdo é o encarregado de definir como esse processo será realizado.

A transmissão de conteúdo em redes de computadores, principalmente de conteúdo do tipo contínuo (vídeo e áudio), representa um desafio importante na garantia de entrega oportuna de grandes quantidades de dados. Com o intuito de facilitar a transmissão de conteúdo de áudio e vídeo, o conteúdo desses objetos de mídia são quebrados em pacotes. Com a evolução das tecnologias de redes, a entrega de áudio e vídeo através de pequenos pacotes tem diminuído consideravelmente. Em seu lugar, o conteúdo multimídia pode ser entregue eficientemente através de grandes segmentos, usando o protocolo HTTP (*Hypertext Transfer Protocol*). HTTP *streaming* permite o gerenciamento do fluxo sem precisar manter uma sessão com o servidor. Isso viabiliza a entrega de conteúdo para milhões de usuários com um custo viável.

HTTP *streaming* é uma abordagem muito popular nos desenvolvimentos comerciais. Por exemplo, plataformas *streaming* como *Apple's HTTP Live Streaming* (Pantos & May 2011), *Microsoft's Smooth Streaming* (Microsoft 2009), e *Adobe's HTTP Dynamic Streaming* [6], utilizam HTTP *streaming* como o seu método base para a entrega de conteúdo. Não obstante, cada uma dessas implementações proprietárias possui um sistema fechado, utilizando diferentes

[6] http://www.adobe.com/products/httpdynamicstreaming

formatos para especificar a informação relacionada com a apresentação multimídia, assim como a forma de segmentar o conteúdo. Em consequência, para receber o conteúdo a partir de cada servidor, o cliente deve suportar o protocolo usado especificamente por esse servidor.

Com o intuito de criar um formato universal para a entrega de conteúdo multimídia através de *streaming*, o grupo MPEG tem recentemente concentrado esforços em um novo padrão para *streaming* multimídia, conhecido como MPEG *Dynamic Adaptive Streaming over HTTP*, ou MPEG-DASH. Esse padrão incorpora os melhores elementos das três principais soluções proprietárias mencionadas acima. MPEG-DASH procura proporcionar a tão procurada interoperabilidade entre os diferentes servidores de rede e os diferentes dispositivos eletrônicos de consumo.

Com a introdução do *streaming* adaptativo sobre HTTP, o conteúdo multimídia pode ser entregue com mais facilidade do que antigamente. Em particular, o *streaming* adaptativo possui duas caraterísticas importantes. A primeira refere-se à quebra de conteúdo de vídeo em pequenos pedaços para facilitar a sua apresentação. Por exemplo, o protocolo *Apple's HTTP Live Streaming* (HLS) geralmente quebra o conteúdo de vídeo em segmentos que possuem uma duração de 10 segundos, enquanto o protocolo *Microsoft's Smooth Streaming* (MSS) e *Adobe's HTTP Dynamic Streaming* (HDS) usualmente quebram o vídeo em pedaços ainda menores: cinco segundos ou menos.

A segunda característica define a possibilidade de codificar o conteúdo de vídeo utilizando várias taxas de bits (*bitrates*), assim como diferentes resoluções (Cada uma dessas codificações é denominada perfil). Esse mecanismo permite que o cliente faça a escolha entre os vários perfis disponíveis e, em seguida, adapte automaticamente pedaços pequenos ou grandes dependendo das condições da rede de comunicação (Salo 2012).

O protocolo MPEG-DASH é abordado com mais detalhe no Apêndice A, onde também é apresentada uma tabela comparativa de todos os protocolos *streaming* acima mencionados.

2. *Para definir a ordem de recuperação dos objetos de mídia, além de considerar a sua disposição temporal, é importante levar em conta os relacionamentos causais que existem entre eles a fim de diminuir a latência de*

exibição.

(Kim et al. 2001) estabelece que os objetos de mídia discretos possuem uma prioridade de recuperação sobre os objetos de mídia contínua dentro de um bloco de sincronização. Essa prioridade é atribuída devido a que, no caso de acontecer um retardo na recuperação dos conteúdos dos objetos de mídia contínua, eles podem ser apresentados usando um carregamento parcial do conteúdo ou diminuída a sua qualidade de apresentação. Enquanto que um retardo na recuperação dos conteúdos dos objetos de tipo discreto provoca um aumento na duração do tempo total da apresentação.

Além disso, Kim considera mais importante manter a sincronização intra-blocos que a sincronização inter-blocos. Essa afirmação nem sempre é válida, devido a que poderia dar-se o caso em que os conteúdos, sejam eles contínuos ou discretos, pertencentes a um bloco de sincronização se estenderem aos blocos seguintes, ou possuírem uma relação temporal rígida com os conteúdos que fazem parte dos blocos seguintes. Nesse caso, a prioridade de recuperação dos objetos de mídia deve ser analisada, levando em consideração os relacionamentos causais entre os objetos de mídia que compõem a apresentação multimídia, a fim de evitar a presença de *gaps* entre os blocos de sincronização.

As relações causais, também conhecidas como relações baseadas em eventos, estão baseadas em condições e ações. Essas relações definem ações que devem ser disparadas quando algumas condições são satisfeitas. Um exemplo de relação causal é: “Quando a apresentação do vídeo *A* acabar, inicia a apresentação do áudio *B*”.

Nesta dissertação utilizamos aplicações multimídia desenvolvidas na linguagem declarativa NCL (*Nested Context Language*). Os relacionamentos dos objetos de mídia nessa linguagem são baseados em eventos. Um evento NCL é uma ocorrência no tempo que pode ser instantânea ou ter um tempo de duração. Cada um desses eventos são definidos por uma máquina de estados que é controlada pelo *player* NCL (Figura 3).

Em uma aplicação multimídia onde a sincronização dos seus objetos de mídia está baseada em eventos, torna-se importante considerar esses relacionamentos causais no momento de definir a ordem na qual os objetos de mídia serão recuperados. Para analisar e discutir essa situação, um exemplo é

apresentado a seguir. Nesse exemplo o objeto de mídia é recuperado por completo. Além disso, só é permitida a recuperação de um objeto por vez.

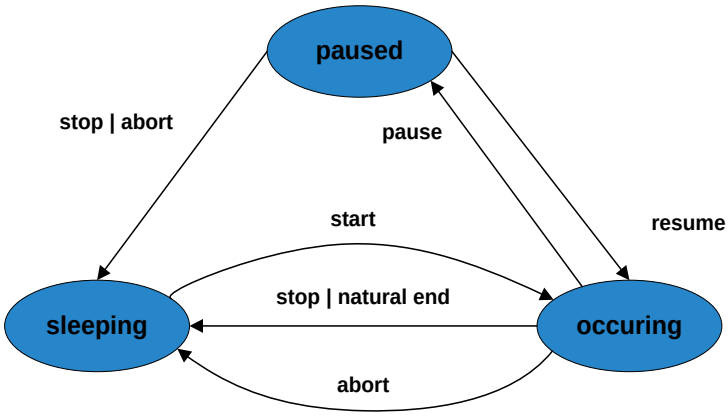


Figura 3 - Máquina de estado de um evento (Soares & Barbosa 2012)**.**

Ressaltamos que os trabalhos relacionados apenas tratam transições[7] do tipo "*Start*". As ações do tipo "*Stop*", "*Abort*", "*Pause*" e "*Resume*" não são consideradas e isso altera completamente a pré-busca do conteúdo. Essas ações executadas em objetos de mídia contínua podem modificar o plano de pré-busca

Exemplo 1:

Tem-se uma aplicação multimídia conformada por duas imagens (I1, I2), dois vídeos (V1 e V2), um objeto de áudio (A1) e um objeto de texto (T1). Todos esses objetos são dispostos temporalmente e apresentados como é mostrado na Figura 4. O objeto de mídia I1 possui um relacionamento do tipo: Ao terminar I1 inicia V1 e T1. O objeto de mídia V1 possui um relacionamento do seguinte tipo: Ao terminar V1 inicia V2 e A1. Finalmente, I2 é iniciado quando V2 acabar.

[7] Refere-se a transição na máquina de estados. Pode assumir os valores: "*start*", "*stop*", "*abort*", "*pause*" ou "*resume*", conforme a máquina de eventos de um objeto.

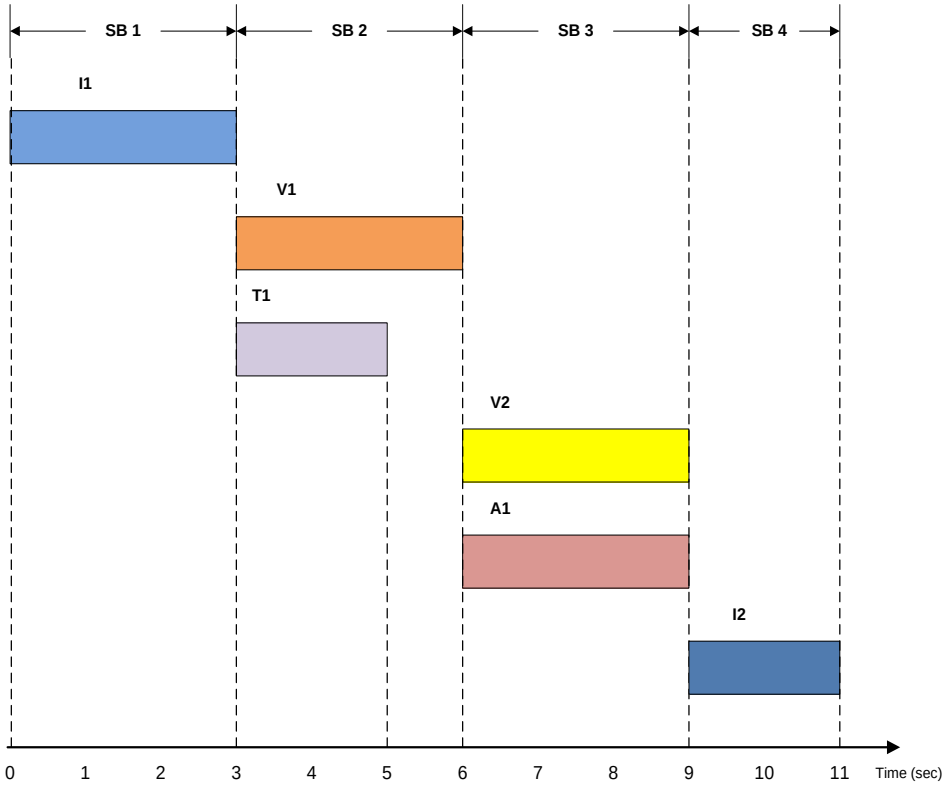


Figura 4 - Representação temporal do Exemplo 1.

Conforme estabelecido por (Kim et al. 2001), a aplicação multimídia seria quebrada em blocos de sincronização e a recuperação dos objetos de mídia realizada de maneira sequencial seguindo a ordem dos blocos. Mais ainda, dentro de um bloco de sincronização os objetos de mídia discretos possuem uma prioridade de recuperação sobre os objetos de mídia contínua. Considerando que os tempos máximos para a recuperação dos objetos de mídia são: 1, 3, 0.5, 3, 2 e 1 segundos para I1, V1, T1, V2, A1 e I2 respectivamente, a pré-busca, segundo Kim, da aplicação apresentada na Figura 4 é mostrada na Figura 5.

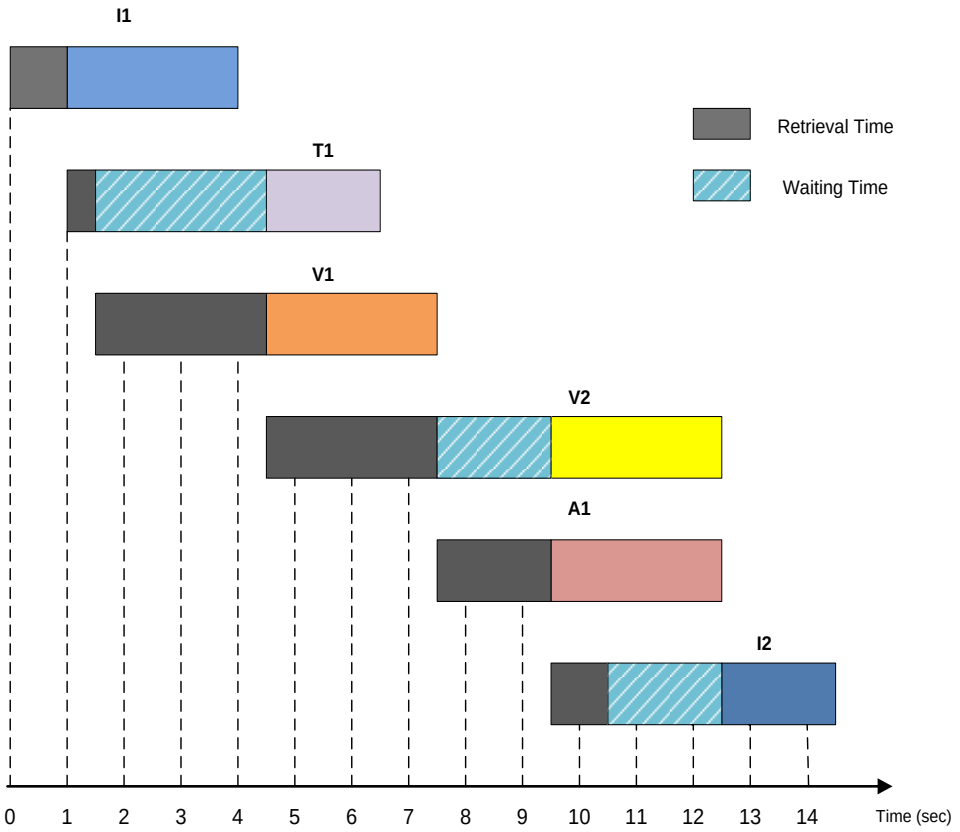


Figura 5 - Pré-busca do Exemplo 1 conforme o algoritmo de Kim (Kim et al. 2001**).**

Nesse caso o tempo total de apresentação é de 14,5 segundos e um *gap* de 0,5 segundos é inserido entre SB1 e SB2; um outro *gap* de 2 segundos é inserido entre SB 2 e SB 3. Se os relacionamentos causais entre os objetos de mídia fossem levados em consideração no momento da recuperação dos objetos, os tempos de duração desses *gaps* poderiam ser diminuídos ou evitados.

NCL, em sua versão 3.1, permite a definição de um atributo que é o encarregado de especificar o tipo de sincronismo ligado a uma determinada ação. Esse atributo define se as ações possuem um sincronismo do tipo *"Synchronous"* ou *"Plesiochronous"*. "*Synchronous"* (*Sync*) estabelece que tanto o objeto de mídia ligado ao papel como os objetos de mídia ligados a essa ação devem ser apresentados no mesmo instante de tempo e de forma sincronizada (sincronismo rígido). Por outro lado, *"Plesiochronous"* (*PleSync*) define que um máximo esforço deve ser realizado para que os objetos sejam apresentados no mesmo instante de tempo (sincronismo fraco).

Em geral, um retardo na exibição de um ou mais objetos de mídia que

possuem um relacionamento do tipo *Sync* prejudica o entendimento da aplicação multimídia. Enquanto que, um retardo na apresentação dos objetos de mídia que possuem um relacionamento *PleSync* não afetaria a compreensão nem a continuidade da aplicação como tal. O ideal é conseguir satisfazer tanto os relacionamentos *Sync* como *PleSync.* Porém, devido a diversos fatores, como o tempo de chegada da aplicação multimídia à plataforma de exibição e retardos inseridos na recuperação dos objetos causados pelas flutuações na rede de comunicação, nem sempre isso pode ser conseguido. Por esse motivo, o principal objetivo torna-se preservar a maior quantidade de relacionamentos *Sync*.

Note que Kim considera que todos os objetos de mídia que pertencem a um bloco de sincronização possuem um relacionamento *Sync* entre eles e, portanto, guardam uma restrição em relação ao instante de tempo no qual esses objetos devem ser apresentados. Todos os objetos de mídia pertencentes a um bloco de sincronização devem ser recuperados para que a sua apresentação possa ser iniciada.

Voltando ao Exemplo 1, suponha que os objetos de mídia V1 e T1 estão ligados a uma ação *Start_PleSync*, que é disparada quando I1 acaba de ser apresentado. Nesse caso, tanto V1 como T1 possuem a mesma prioridade de recuperação porque suas ações são do tipo *PleSync*. Suponha também que o objeto de mídia V2 está ligado a uma ação *Start_Sync* enquanto que A1 está ligado a uma ação *Start_PleSync*. Finalmente, suponha que I2 está ligado a uma ação *Start_Sync* que é disparada quando a apresentação de V2 chega ao seu fim.

Para determinar se os relacionamentos causais influenciam na ordem de recuperação dos objetos de mídia e em consequência reduzem a latência total de exibição[8], todas as possibilidades devem ser avaliadas. A Figuras 6 apresenta a recuperação dos objetos de mídia dando prioridade à recuperação de T1. A Figura 8 apresenta a recuperação dos objetos de mídia dando prioridade à recuperação de V1. Em ambos casos V2 é recuperado antes do que A1.

[8] Latência total de exibição refere-se à soma das latências de exibição de todos os objetos de mídia que compõem uma apresentação multimídia.

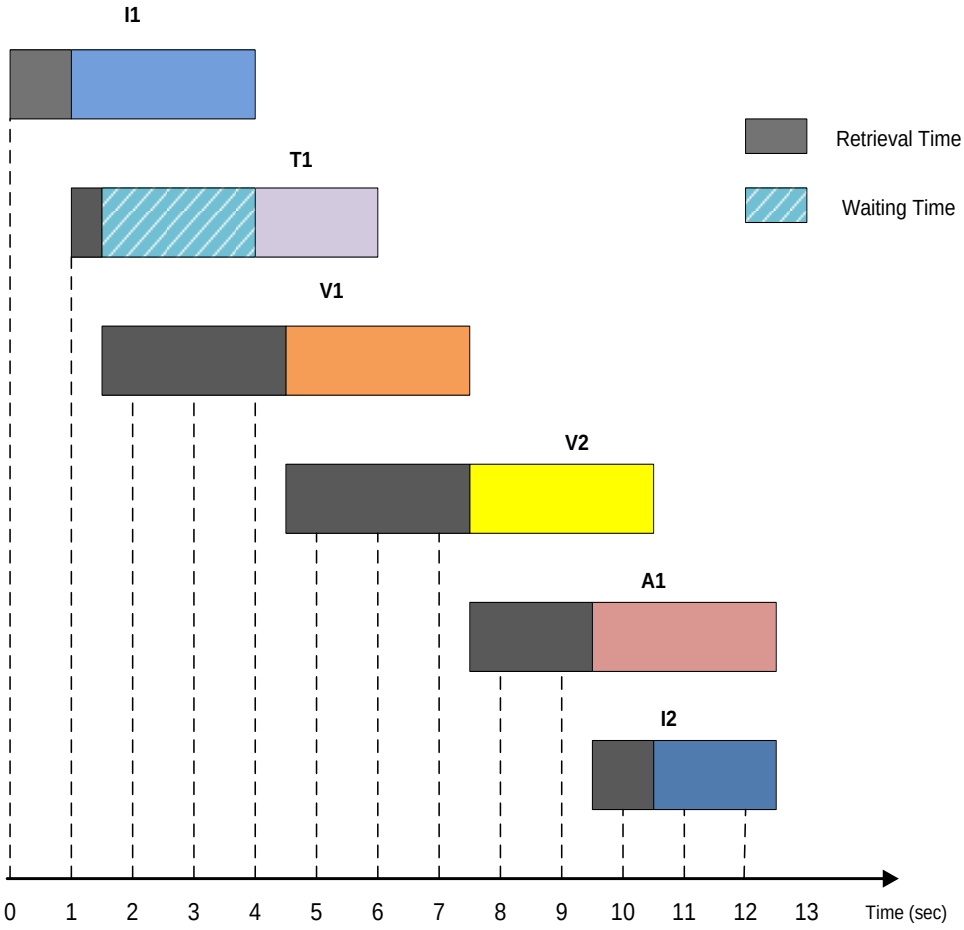


Figura 6 - Pré-busca do exemplo 1, considerando os relacionamentos causais entre os objetos de mídia. T1 e V2 possuem prioridade de recuperação.

Note que na Figura 6, o tempo total da apresentação é o mesmo que o obtido aplicando o algoritmo de Kim, com a diferença que nenhum *gap* foi inserido durante a apresentação, isso deve-se ao fato que os relacionamentos causais foram considerados no momento da recuperação. Cabe ressaltar que no algoritmo proposto por Kim (Kim et al. 2001) todos os objetos de mídia que pertencem a um bloco de sincronização devem ser recuperados para que a sua apresentação seja iniciada. Ao contrário do que acontece quando consideramos os relacionamentos relativos entre os objetos de mídia. Por exemplo, o relacionamento "*OnEnd"* I1 *"Start_PleSync"* V1 e *"Start_PleSync"* T1 estabelece que uma vez terminada a apresentação de I1, a exibição de V1 e T1 pode ser iniciada, no entanto não existe uma restrição que defina que tanto V1 como T1 devem iniciar a sua apresentação no mesmo instante de tempo, uma vez que o sincronismo especificado é do tipo *PleSync*. Pode existir, no entanto, o caso no qual um relacionamento defina que

dois ou mais objetos ligados por um sincronismo *Sync*. Nessa situação, se o tempo disponível para a recuperação de todos esses objetos não for suficiente, será inevitável a presença de *gaps.*

Por outro lado, note na Figura 7 que, se o objeto A1 fosse recuperado antes que V2, o objeto I2 teria que esperar a finalização de V2 para ser apresentado conforme ao relacionamento "*OnEndStart_Sync"* estabelecido entre esses dois últimos objetos de mídia. Além disso, o relacionamento *Sync* entre V1 e V2 não seria satisfeito.

Conforme mencionado anteriormente, V2 deve ser recuperado antes do que A1 e dessa forma é possível diminuir o tempo de apresentação de 14,5 segundos para 12,5 segundos. Nesse exemplo específico não faz diferença se T1 ou V1 é recuperado um antes do que o outro, porém, cabe ressaltar que o fato de um objeto de mídia estar ligado a uma transição do tipo *PleSync* evita ou diminui a inserção de *gaps* causados pela espera da recuperação de um objeto de mídia.

A Figura 8 mostra como a recuperação dos objetos de mídia do Exemplo 1 é realizada levando em consideração todas as premissas levantadas anteriormente.

Como podemos observar, nessa aplicação a prioridade de recuperação de V1 e T1 é a mesma, isto é, não interfere no aumento do tempo total de apresentação. Enquanto que a escolha de recuperar primeiro V2 ou A1 é decisiva no aumento ou diminuição da latência total de exibição e, consequentemente, do tempo total de apresentação. A latência total de exibição resultante com a abordagem de Kim é igual a 3,5 segundos, enquanto que se os relacionamentos causais são considerados, a latência total de exibição é igual a 1 segundo (2,5 menos que Kim), tempo equivalente ao atraso inicial.

Nota-se assim que é importante considerar os relacionamentos causais existentes entre os objetos de mídia no momento de definir a ordem na qual os objetos serão recuperados e não simplesmente considerar a sua disposição no eixo temporal.

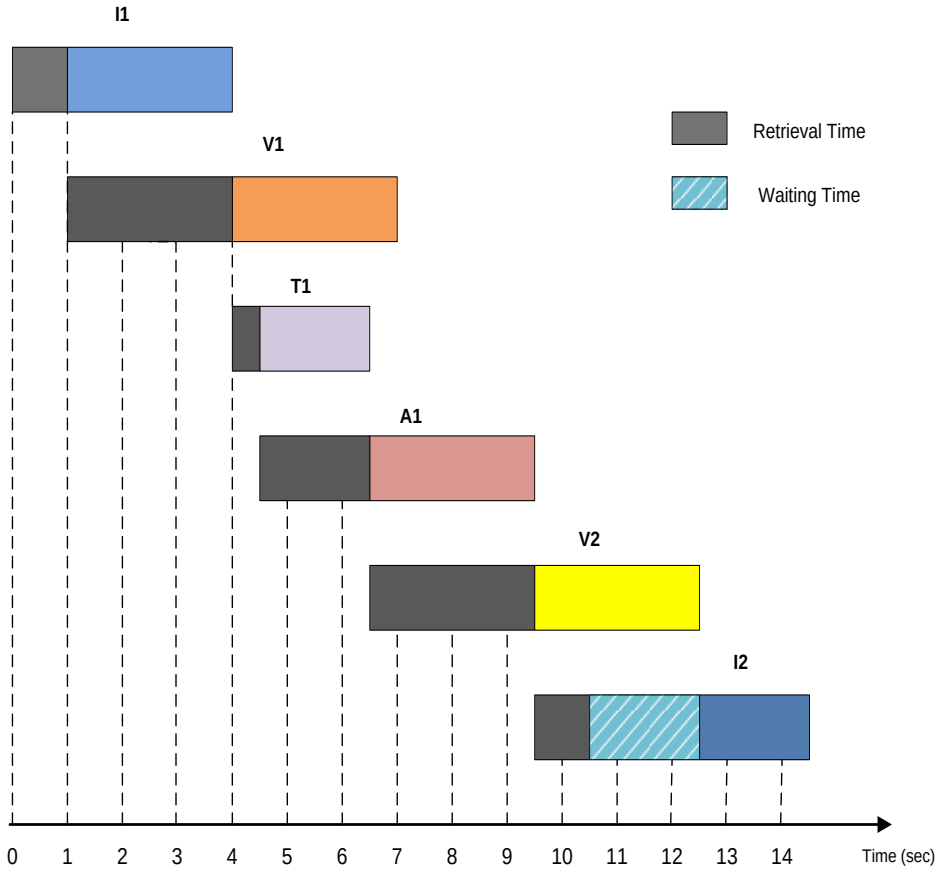


Figura 7 - Pré-busca do exemplo 1, considerando os relacionamentos entre os objetos de mídia. V1 e A1 possuem prioridade de recuperação.

Basicamente os relacionamentos causais devem ser considerados por três motivos principais: primeiro, as especificações do autor relacionadas com a apresentação de uma aplicação multimídia devem ser respeitadas, se possível, em sua maioria. Segundo, existem conteúdos que necessariamente devem ser apresentados de forma simultânea porque possuem uma relação forte de sincronismo entre eles. Terceiro, ao considerar os relacionamentos causais entre os objetos de mídia é factível definir quais objetos possuem uma restrição de apresentação e quais não, priorizando a recuperação de objetos e evitando a recuperação desnecessária de outros, evitando aumentar a latência de exibição.

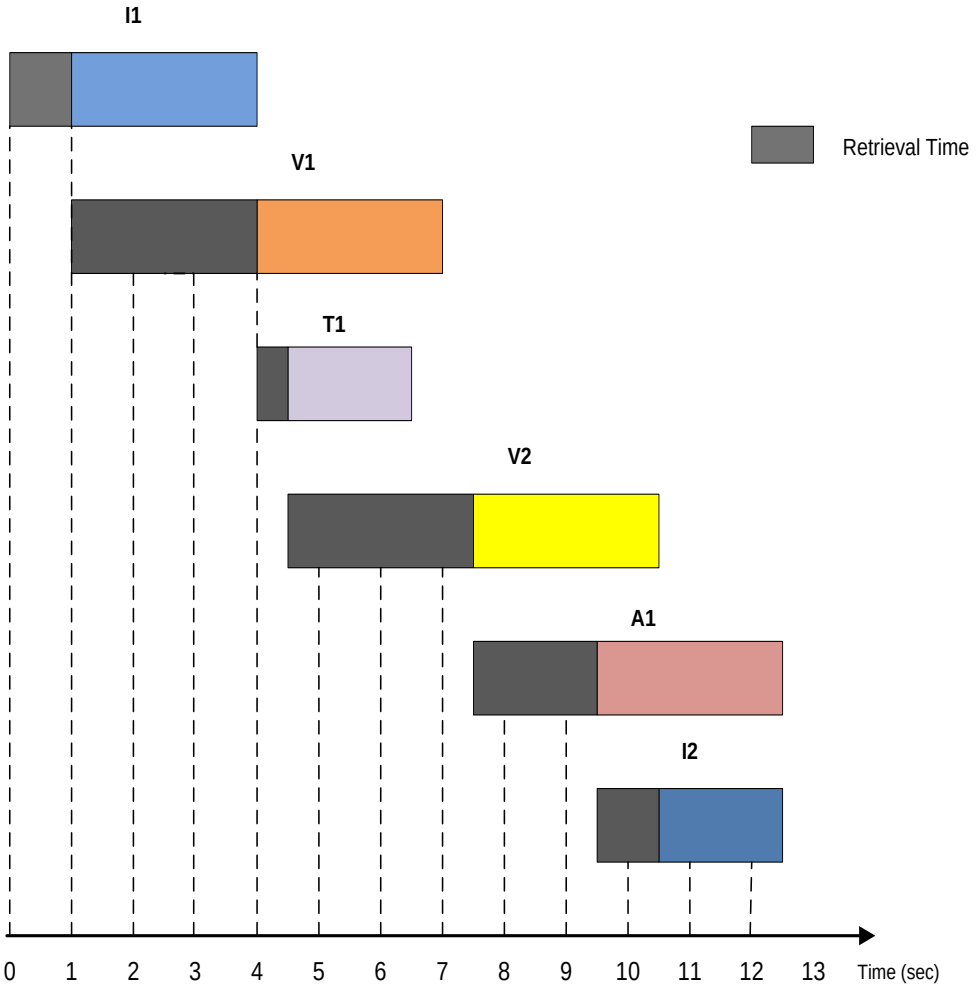


Figura 8 - Pré-busca do exemplo 1, considerando os relacionamentos entre os objetos de mídia. V1 e V2 possuem prioridade de recuperação.

3
Características de um ambiente de pré-busca

Este capítulo apresenta, em sua Seção 3.1, as principais caraterísticas que devem ser consideradas num ambiente de pré-busca de aplicações. O objetivo é, com base nessas caraterísticas, definir os aspectos que serão considerados no escopo desta dissertação. Em seguida, o cenário para a construção do plano de pré-busca é detalhado na Seção 3.5.

3.1
Relações temporais entre os objetos de mídia

A apresentação de uma aplicação multimídia inclui vários tipos de relacionamentos temporais entre os objetos de mídia que a conformam. Para a exibição desses objetos de maneira coordenada é fundamental que esses relacionamentos temporais, os quais são especificados na etapa de autoria, sejam considerados.

Nesse contexto, Feng-Cheng Lin et al. estabelecem que os relacionamentos temporais comumente encontrados incluem simultaneidade, contiguidade e precedência. Uma relação de contiguidade é estabelecida entre dois objetos de mídia quando o segundo objeto é apresentado consecutivamente ao primeiro objeto e não existe nenhum outro objeto de mídia entre eles. Por outro lado, em uma relação de simultaneidade entre dois objetos de mídia, é preciso que o segundo objeto seja iniciado e apresentado de forma simultânea com o primeiro objeto. Quando esses objetos são entregados por meio de uma rede de comunicação, os objetos devem ser recuperados antes do seu tempo de exibição para que a sua sincronização seja garantida. Finalmente, em uma relação de precedência entre dois objetos de mídia a e b, onde a precede b $(m_a \rightarrow m_b)$ o segundo objeto tem que ser apresentado depois do primeiro objeto, mas não necessariamente de forma imediata.

Conforme ao mencionado no Capítulo 1, para a construção de um plano de pré-busca é necessário especificar o sincronismo espaço-temporal dos diferentes

objetos de mídia que compõem uma aplicação multimídia. Nos principais trabalhos existentes na literatura, a especificação desse sincronismo considera apenas os intervalos absolutos de um eixo temporal sem levar em conta os relacionamentos causais entre os objetos de mídia.

Com o objetivo de reduzir os *gaps* produzidos em uma apresentação multimídia, os relacionamentos causais entre os objetos de mídia devem ser considerados no momento de definir a ordem de recuperação dos conteúdos dos objetos de mídia.

Determinar a prioridade de recuperação dos objetos de mídia em apresentações multimídia, onde vários tipos de relações temporais são especificadas entre os seus objetos de mídia, é um problema muito complexo devido a que cada aplicação multimídia possui uma lógica diferente. Por esse motivo, é preferível agrupar os objetos de mídia que estejam estreitamente relacionados para serem apresentados como um grupo.

3.2
Retardo produzido na recuperação dos conteúdos multimídia

O retardo produzido durante o processo de recuperação dos conteúdos de objetos de mídia é um parâmetro muito importante a ser levado em conta no momento da construção do plano de pré-busca. Esse tempo de recuperação pode ser considerado como o tempo necessário para colocar os conteúdos dentro do *buffer* de armazenamento.

Para a apresentação de uma aplicação multimídia, os conteúdos dos objetos de mídia podem ser adquiridos de duas formas: sob demanda (modelo *pull* de serviço) ou/e sem a solicitação do sistema de apresentação (modelo *push* de serviço). No modelo tipo *pull* os dados encontram-se armazenados de forma local ou remota em vários servidores diferentes, cada um deles com as suas características próprias. Nesse caso, para o cálculo total do tempo de retardo produzido pela recuperação do conteúdo deve ser considerado o retardo de transmissão da rede mais o retardo produzido pelo servidor (provedor de conteúdo). Por outro lado, no modelo tipo *push*, é comumente utilizado um mecanismo para o envio cíclico da aplicação multimídia, denominado carrossel de objetos, utilizado principalmente em sistemas de TV Digital (S & A 2005). No

modelo *push*, o registro de um cliente em um serviço pode ser realizado em qualquer instante da distribuição de dados. O envio cíclico da aplicação torna-se então vantajoso, uma vez que permite o recebimento de desses dados de forma independente do instante em que o registro no serviço é realizado.

Se os conteúdos dos objetos de mídia a serem apresentados encontram-se dentro de um carrossel de objetos, o retardo total inserido, nesse caso, é igual à taxa de transmissão do carrossel de objetos mais o tempo de acesso de um objeto de mídia específico no carrossel. Esse tempo de acesso refere-se ao tempo que o usuário tem de esperar para acessar um determinado objeto do carrossel. Esse tempo varia de acordo com o número de vezes que o objeto foi inserido em um ciclo do carrossel de objetos e do tamanho do carrossel. Um objeto pode ser transmitido mais de uma vez no carrossel a fim de diminuir seu tempo de acesso, aumentando, contudo, o tamanho do carrossel e o tempo de acesso aos outros arquivos. Aumentar o tamanho do carrossel significa diminuir a banda passante para outros conteúdos.

Resumindo, para a recuperação prévia dos conteúdos de mídia armazenados tanto em forma local, remota, ou dentro de um carrossel de objetos, os parâmetros a serem considerados são o retardo de transmissão e o tempo de acesso ao sistema de armazenamento ou ao carrossel de objetos respectivamente. Usualmente, na recuperação de conteúdos remotos o retardo de transmissão da rede é mais relevante que o tempo de acesso ao sistema de armazenamento do servidor. Já para conteúdos no carrossel de objetos o tempo de acesso é usualmente mais significativo que a taxa de transmissão do carrossel.

3.3
Capacidade do Buffer de Armazenamento

Na pré-busca os conteúdos recuperados são armazenados em um *buffer* no lado do cliente até que a sua reprodução seja concluída (Assume-se que o *player* de mídia prove o serviço de controle do *buffer*).

É comum, principalmente em receptores de baixo custo (*low-end*) ter-se plataformas com capacidade de armazenamento limitada. Nesse caso, para a construção do plano de pré-busca, é necessário levar em conta o tamanho máximo do *buffer* disponível.

A construção de um plano de pré-busca considerando a existência de uma restrição na capacidade de armazenamento do *buffer* é um problema NP-Difícil (Lin et al. 2008; Papadimitriou & Kanellakis 1980). Uma alternativa adotada em muitos algoritmos propostos na literatura tem sido tratar a capacidade do *buffer* como infinita e, a partir dessa premissa, calcular o tamanho máximo do *buffer* requerido.

Conforme ao apresentado em (In-Ho Lin & Wu 2001), o tamanho do *buffer* requerido por um conjunto de segmentos correspondentes aos objetos de mídia que fazem parte da aplicação multimídia, em um instante de tempo definido, pode ser calculado a partir da taxa de consumo dos dados, a taxa de transmissão e os tempos de duração dos conteúdos de mídia a serem apresentados.

3.4
Existência de eventos não determinísticos

Uma aplicação multimídia interativa é formada por uma cadeia temporal[9] principal, formada por uma sequência de eventos determinísticos, que começa com o próprio evento de início da apresentação de uma aplicação. Cada evento não determinístico (por exemplo, uma interação do usuário) inicia uma nova cadeia secundária de eventos determinísticos que se acopla à cadeia principal quando da ocorrência do evento (Rodrigues & Gomes Soares 2002).

Existem alguns aspectos importantes quanto ao instante de início da interação (evento não determinístico). Pode ser que se deseje a exibição do conteúdo interativo de forma imediata; ou pode ser que exista um intervalo de tempo, definido na autoria, o qual tem que ser respeitado antes que o resultado da interação seja apresentado.

Nesta dissertação somente são consideradas as aplicações multimídia que possuem eventos determinísticos. No entanto, a análise de como a pré-busca de aplicações multimídia que possuem tanto eventos determinísticos como não determinísticos (aplicações interativas) deveria ser realizada é apresentada no Capítulo 6.

[9] Uma cadeia temporal corresponde a uma sequência de eventos (ocorrências no tempo), iniciada pelo evento que corresponde ao início da apresentação do objeto hipermídia declarativo.

3.5
Descrição do cenário foco desta dissertação

Conforme as características levantadas na seção anterior, a Figura 9 apresenta uma representação gráfica do cenário utilizado neste trabalho.

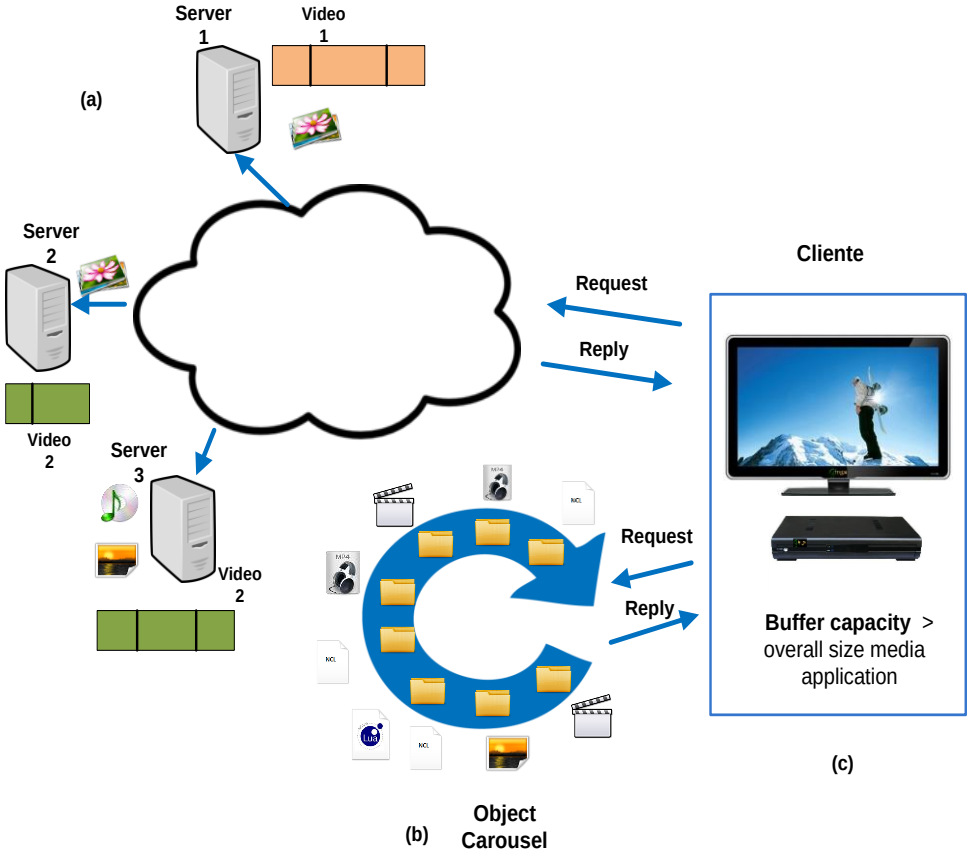


Figura 9 - Cenário Levantado.

- A Figura 9 (a), mostra que os conteúdos dos objetos de mídia que fazem parte da aplicação multimídia podem estar armazenados localmente, em dispositivos de armazenamento remoto (servidores).
- Além disso, os conteúdos podem estar também armazenados em carrosséis de objetos (Figura 9 (b)).
- Para a recuperação dos conteúdos, são considerados os parâmetros referentes à rede de comunicação utilizada, os tempos de acesso ao sistema de armazenamento, e os tempos de acesso ao carrossel.
- A plataforma de exibição possui um *buffer* de armazenamento com capacidade "infinita", isto é, o tamanho do *buffer* é maior do que o tamanho total requerido pela aplicação (Figura 9 (c)).

- As aplicações multimídia são desenvolvidas utilizando a linguagem declarativa NCL.
- Por utilizar NCL, é possível determinar, antes do início da aplicação, os tempos de exibição relativos, de cada um de seus objetos de mídia.

4
Parâmetros de Entrada para Construção do Plano de Pré-busca

Uma aplicação multimídia consiste de diferentes tipos de objetos de mídia, os quais possuem tempos de início de apresentação e tempos de duração diferentes. Essas instâncias temporais e os tempos de duração dos objetos de mídia podem ser especificados tanto de maneira rígida como de maneira flexível (Candan et al. 1998). Na especificação temporal rígida os tempos antes mencionados são fixos, a diferença do que acontece quando a aplicação multimídia possui uma especificação flexível, na qual esses tempos podem variar dentro de um grupo de valores definidos. Note-se que se as especificações temporais são rígidas, os tempos agendados para a apresentação dos objetos de mídia devem ser os mesmos que foram definidos na fase de autoria. Cabe ressaltar que no presente trabalho somente serão consideradas as aplicações multimídia especificadas de forma rígida.

Em uma apresentação multimídia distribuída os objetos que a compõem encontram-se dispersos em vários dispositivos de armazenamento remotos e/ou locais. Durante o processo de recuperação do conteúdo, esses dispositivos atuam como *servidores* e o sistema de apresentação atua como *cliente*. A recuperação é iniciada pelo cliente e é dada pelos seguintes passos:

- Identificar o *plano de apresentação* da aplicação multimídia. Em (Costa 2010) é proposto e construído um *plano de apresentação* para orquestrar a apresentação de uma aplicação multimídia. Na Seção 4.1, é explicado de forma detalhada em que consiste esse plano e com ele é gerado.
- Identificar o *plano de pré-busca* da aplicação multimídia. Esse plano especifica os instantes de tempo nos quais as requisições ao servidor devem ser feitas por parte do cliente, com intuito de apresentar os conteúdos dos objetos de mídia nos tempos definidos no *plano de*

apresentação.

Basicamente, a especificação dos instantes de tempo para a recuperação dos conteúdos dos objetos de mídia é feita a partir do cálculo do tempo requerido para transferir um objeto de mídia a partir do servidor para o cliente. Por tanto, o *plano de pré-busca* considera os seguintes fatores:

- A vazão (ou a largura de banda) do canal de comunicação estabelecido entre o servidor e o cliente.
- O espaço disponível no *buffer* para armazenar os conteúdos dos objetos de mídia recuperados.
- O tamanho dos objetos a serem recuperados a partir do servidor.
- O tempo máximo tolerável (fator de QoS) para a recuperação de um determinado tipo de objeto de mídia (como é discutido no Capítulo 5, Seção 5.3).

A vazão do canal de comunicação e os recursos do *buffer* são dependentes da plataforma de exibição. Além disso, a vazão disponível pode variar de acordo com o tipo de rede e da carga que ela possui. Enquanto que o tamanho dos objetos de mídia e o tempo máximo permitido para a sua recuperação dependem exclusivamente da aplicação.

Os objetos de mídia a serem recuperados são classificados como: objetos atómicos e objetos tipo *stream* (Candan et al. 1998).

Os objetos do tipo atômico precisam ser recuperados por completo e armazenados do lado do cliente antes da sua apresentação. Um exemplo desse tipo de objetos são as imagens. Contudo, objetos de mídia tipo *stream* podem ser apresentados assim que alguns segmentos pertencentes ao objeto de mídia são recebidos. Geralmente, os objetos de mídia de tipo vídeo e áudio são considerados de tipo *stream*. Entretanto, em sistemas nos quais não é possível efetuar a operação de apresentação e recuperação em paralelo, os objetos de mídia devem ser recuperados por completo.

4.1
Plano de apresentação

Na sincronização baseada em eventos, ao contrário da sincronização *timeline*, os relacionamentos em uma aplicação são definidos de forma relativa, sem nenhuma menção ao tempo absoluto de suas ocorrências. Aplicações hipermídia podem ser completamente especificadas e apresentadas através da sincronização baseada em eventos. No entanto, essa forma de sincronização não oferece facilidade para o controle do tempo da apresentação e da transmissão dos dados, uma vez que os momentos de ocorrência dos eventos nas aplicações são desconhecidos (Costa 2010).

Com intuito de controlar o comportamento temporal das aplicações durante a sua execução, em (Costa 2010), uma estrutura denominada HTG (*Hypermedia Temporal Graph*) é construída. O HTG consiste em um grafo temporal de dependências que oferece suporte ao início ou à retomada síncrona da apresentação a partir de um instante de tempo qualquer.

No HTG, os relacionamentos são representados através de arestas dirigidas, enquanto os vértices representam os eventos (ocorrências no tempo) que podem acontecer sobre os conteúdos das mídias. Os eventos representados nessa estrutura são os mesmos definidos na linguagem NCL, ou seja, os eventos de apresentação, seleção e atribuição, cada um controlado por uma máquina de estados (Soares & Barbosa 2012).

O grafo temporal obtido deve preservar todos os relacionamentos entre os eventos, sejam eles determinísticos ou não determinísticos, bem como o estado atual de cada máquina de estados associada a um evento.

Tendo como base o HTG, cinco estruturas de dados podem ser derivadas para o controle do sincronismo das apresentações multimídia. Essas estruturas são: plano de apresentação, plano de carregamento de exibidores, plano de distribuição, plano de pré-busca e plano de QoS. O trabalho de (Costa 2010) foca-se na construção do plano de apresentação a partir do HTG gerado.

Os planos mencionados anteriormente encontram-se detalhados em (Costa 2010). Com respeito ao plano de pré-busca, ele não é gerado apenas é mencionado como ele deveria ser tratado é construído. Para a sua construção é definido que somente devem ser considerados os instantes temporais relativos às transições de

início (“*Start*”) do evento de apresentação. Além disso, adota-se um cenário conservador o qual, no caso extremo, consiste em aguardar o recebimento de todos os conteúdos para que uma apresentação seja iniciada.

Antes do início da apresentação, o plano de apresentação é formado apenas pelas especificações associadas à cadeia principal. Durante a apresentação, o plano de apresentação deve ser atualizado, caso ocorra uma transição de evento imprevisível, origem de uma cadeia. Antes, porém, as especificações temporais calculadas para essa cadeia devem ser atualizadas.

Para o desenvolvimento do presente trabalho, com base no HTG e no *plano de apresentação* proposto em (Costa 2010), é construída uma estrutura de dados que contém informações específicas sobre os atributos dos objetos de mídia, que compõem uma determinada aplicação multimídia. Fundamentalmente, essas informações são: identificador de objeto de mídia, especificações temporais dos objetos de mídia (tempo de início e tempo de fim de apresentação), tamanho, tipo de objeto (contínuo ou discreto), papéis e transições ligadas a cada objeto de mídia, tempo máximo de recuperação e a localização dos conteúdos associados a cada objeto de mídia.

4.2 Segmentação da aplicação multimídia

Uma aplicação multimídia está formada por vários objetos de mídia, tanto do tipo discreto como contínuo. Esses objetos encontram-se organizados em um eixo temporal. Com o intuito de facilitar a recuperação dos objetos de mídia que fazem parte de uma aplicação multimídia, essa pode ser dividida em grupos menores.

Além de facilitar a recuperação dos conteúdos dos objetos de mídia, destacam-se duas principais vantagens de quebrar uma aplicação multimídia. A primeira vantagem é diminuir as interrupções inseridas durante a apresentação devido à recuperação de conteúdo desnecessário em um determinado instante de tempo. Assim, uma apresentação mais contínua e suavizada pode ser alcançada com uma maior granularidade da aplicação. A segunda vantagem tem relação com a diminuição do *overhead* na alocação de recursos na plataforma de apresentação.

Os instantes de tempo de início e fim da apresentação de cada objeto de

mídia são considerados como pontos de quebra da aplicação que, por sua vez, delimitam os grupos. É importante destacar que, essa divisão em grupos menores, na qual os objetos de mídia, tanto contínuos como discretos são divididos em vários pontos, nem sempre significa que os objetos de mídia são quebrados fisicamente nesses pontos. Essa divisão em grupos é uma divisão "virtual" ou de referência, que tem com propósito determinar os conteúdos de mídia que devem ser apresentados em um determinado intervalo de tempo, e assim, recuperar somente os conteúdos estritamente necessários.

Se um objeto de mídia do tipo discreto é dividido em duas ou mais partes, não significa que o objeto de mídia será recuperado dessa forma. Pelo contrário, deve ser recuperado por completo para ser apresentado. Em contrapartida, os objetos de mídia contínuos não precisam ser recuperados por completo para que a sua apresentação seja iniciada. Aproveitando esse atributo, é possível recuperar um objeto de mídia por partes, assim a sua latência de exibição é diminuída.

4.3
Cálculo dos tempos de recuperação dos objetos de mídia

4.3.1
Pulled Data

Existem dois fatores principais que devem ser considerados para o cálculo dos instantes de tempo nos quais os conteúdos dos objetos de mídia devem ser solicitados: (1) o tempo de início de apresentação de um objeto e (2) o tempo total requerido para a recuperação de um objeto através da rede. O tempo de início de apresentação de cada objeto depende dos relacionamentos temporais entre os objetos de mídia que compõem a apresentação, definidos na fase de autoria, e do instante de tempo no qual o usuário interage, no caso de apresentações multimídia interativas. Por sua vez, o tempo total requerido para recuperar um objeto de mídia depende do tamanho do objeto e da largura de banda da rede de comunicação utilizada para transferi-lo até o cliente.

Levando em conta os aspectos mencionados acima, o instante de tempo que um objeto deve ser solicitado ser estimado pela diferença entre o seu tempo de início de apresentação e o seu tempo de recuperação.

Seja um objeto de mídia Mo_i como tamanho Mo_{size}, sendo a largura de

banda estimada para sua transmissão igual a $BandaEst_{Mo_i}$ e o tempo de acesso ao sistema de armazenamento igual a $AcessoServidor_{Mo_i}$. O tempo total requerido para a recuperação desse objeto, chamado "*Retrieval Time*" (RT), é dado pela Equação 1.

$$RT = \left(Mo_{i_{size}} / BandaEst_{Mo_i}\right) + AcessoServidor_{Mo_i} \qquad (1)$$

4.3.2 Pushed Data

Para calcular o tempo de recuperação de um determinado objeto de mídia que está sendo transmitido por meio de um carrossel, dois parâmetros básicos devem ser considerados: (1) a taxa de transmissão utilizada para transferir o carrossel, e (2) o tempo requerido para acessar um determinado objeto dentro do carrossel. No pior caso, esse tempo de acesso corresponde a um ciclo completo do carrossel de objetos.

Seja a largura de banda estimada do carrossel de objetos é $BandaCarrossel_{Mo_i}$ e o tempo de acesso ao carrossel é $AcessoCarrossel_{Mo_i}$. O tempo total requerido para a recuperação de um objeto de mídia Mo_i é o resultado da adição do retardo produzido pelo tempo de acesso a um determinado objeto e o tempo gastado para a transmissão desse objeto de mídia a partir do carrossel até o cliente. Logo, esse tempo de recuperação (RT) é dado pela Equação 2.

$$RT = \left(Mo_{i_{size}} / BandaCarrossel_{Mo_i}\right) + AcessoCarrossel_{Mo_i} \qquad (2)$$

Cabe ressaltar que tanto no modelo *pulled data* como no *pushed data*, a largura de banda da rede de comunicação está sujeita a variações, bem como o tempo de acesso. Em consequência, o tempo de recuperação não é fixo e sendo esse um dado de entrada do plano de pré-busca, esse plano torna-se dinâmico. Em geral, supõe-se o pior caso e vai se adaptando o plano no decorrer do processo.

4.4
Retardo máximo tolerável em apresentações multimídia (Fator de QoE)

Como foi mencionado na Seção 1.1, os usuários de uma aplicação multimídia são extremamente sensíveis às interrupções que acontecem durante sua apresentação. Assim, essas interrupções, ou *gaps*, devem ser evitados, mesmo à custa de aumentar o *retardo inicial* da aplicação, ou de cadeias secundárias que a compõem.

Devido ao fato de que o tempo de *retardo inicial* é um parâmetro que define a QoE (*Quality of Experience*) percebida pelo usuário, é importante definir qual é o máximo *retardo inicial* tolerável que pode ser inserido antes de se iniciar a apresentação da aplicação multimídia, ou de uma de suas cadeias secundárias. No contexto de aplicações multimídia e de forma mais específica, aplicações multimídia interativas, esses tempos não somente contribuem para a qualidade percebida pelo usuário do sistema, como também adicionam uma sensação de naturalidade e continuidade. (Fiedler 2004) define três limites importantes para os tempos de resposta subjetivos, como por exemplo tempos de espera e a sua relação com respeito à interatividade percebida:

- 0,1 segundos é o tempo limite para conseguir, no usuário, uma sensação de que o sistema reage de forma instantânea.
- 1,0 segundos é o tempo limite para que o usuário se mantenha atento à aplicação, mesmo percebendo que um atraso tem acontecido.
- 10 segundos é o tempo limite para ainda manter alguma atenção do usuário.

O *retardo inicial* da aplicação, ou de uma cadeia secundária da aplicação após a interação é chamado de *tempo de tolerância* nos capítulos seguintes.

5
Construção do plano de pré-busca

Os eventos de apresentação, seleção e atribuição interferem no momento de estabelecer as prioridades de recuperação dos conteúdos dos objetos de mídia que compõem uma apresentação multimídia interativa. Nesta dissertação, para a construção do plano de pré-busca, são tratadas as aplicações multimídia que só possuem eventos determinísticos, ou seja, são considerados os eventos de apresentação e atribuição.

Para a construção do plano de pré-busca é necessário considerar as transições relativas ao início ("*Start*"), continuação ("*Resume*"), pausa ("*Pause*"), fim ("*Stop*") e interrupção (*"Abort"*) do evento de apresentação sobre os objetos de mídia. Ressaltamos que os trabalhos relacionados apenas tratam transições do tipo "*Start*". As outras transições, que normalmente ocorrem em uma aplicação multimídia, como as ações do tipo "*Stop*", "*Abort*", "*Pause*" e "*Resume*", não são consideradas para a construção do plano de pré-busca. Além disso, a diferença dos trabalhos relacionados que só consideram o sincronismo rígido ("*Sync*") entre os objetos de mídia, neste trabalho é considerado tanto o sincronismo rígido ("*Sync*") como o sincronismo fraco ("*PleSync*").

É importante notar que na construção do plano de pré-busca, as prioridades de recuperação dos objetos de mídia estabelecidas no plano podem sofrer alterações quando acontece uma transição de evento. Esse é o caso, por exemplo, quando uma transição do tipo "*Pause*" pausa a apresentação de um objeto de mídia contínua. Nesse caso, a recuperação desse objeto é pausada, se ele ainda não foi totalmente recuperado, dando prioridade à recuperação de outros objetos de mídia que não dependem do relacionamento com o objeto pausado. Alterações similares, acontecem com as transições do tipo "*Stop*" e "*Abort"*, onde a recuperação de um objeto de mídia contínua é parada ou interrompida, dependendo da transição executada. Uma vez que uma dessas transições é disparada, a recuperação do conteúdo desse objeto, a partir do instante de tempo de início da transição em diante, torna-se desnecessária, e o plano de pré-busca

deve ser atualizado. Isso não acontece quando são consideradas apenas transições do tipo "*Start*", em que a recuperação dos conteúdos pertencentes aos objetos de mídia que foram pausados, parados ou abortados é levada até o fim, causando assim uma latência de exibição maior nos próximos objetos de mídia.

É importante que o plano de pré-busca considere todas as transições mencionadas acima para evitar a recuperação desnecessária de conteúdo, causando um aumento na latência de exibição total e um maior uso do *buffer* de armazenamento, que poderia ser utilizado por outros objetos de mídia que devem ser apresentados.

A Tabela 2 apresenta a forma como deveria ser tratada a recuperação de conteúdo, frente as diferentes transições de evento. Cabe mencionar que essas transições de evento podem estar ligadas aos papeis: "*OnBegin*", "*OnEnd*", "*OnAbort*", "*OnStop*", "*OnPause*" e "*OnResume*".

Transição	**Estado de Recuperação**	**Descrição**
Start_Sync	Iniciar recuperação	Tanto os objetos de mídia ligados a essa transição como os objetos ligados a um determinado papel devem ser recuperados previamente para que a sua apresentação seja iniciada simultaneamente, porque possuem um relacionamento de sincronismo rígido ("*Sync*") entre eles.
Start_PleSync	Iniciar recuperação	Os objetos de mídia ligados a essa transição não necessariamente tem que ser apresentados de forma simultânea ao objeto ligado a um determinado papel, embora seja desejável. Somente é estabelecida uma relação de

		precedência entre o objeto ligado a esse papel e os objetos ligados a essa transição. Embora seja tolerável uma certa assincronia, o objeto deve ser apresentado o mais cedo possível.
Stop_Sync Stop_PleSync	Finalizar recuperação	Esse relacionamento não influencia no início da recuperação de um objeto. Entretanto, faz com que a recuperação de um objeto contínuo ligado à transição "*Stop*" termine. A parada de exibição de um objeto contínuo pode afetar (acelerar) todo o plano de pré-busca a partir do ponto de parada.
Abort_Sync Abort_PleSync	Finalizar recuperação	Esse relacionamento não influencia o início da recuperação de um objeto. Entretanto, faz com que a recuperação de um objeto contínuo ligado à transição "*Abort*" termine. A parada de exibição de um objeto contínuo pode afetar (acelerar) todo o plano de pré-busca a partir do ponto de parada.
Pause_Sync Pause_PleSync	Pausar recuperação (se necessário)	Esse relacionamento não influencia o início da recuperação de um objeto. Entretanto, faz com que a recuperação de um objeto contínuo ligado à transição "*Pause*" possa ser suspensa. A pausa de exibição de um objeto

		pode afetar (retardar) todo o plano de pré-busca a partir do ponto de pausa.
Resume_Sync	Retomar/Continuar recuperação, se ela foi pausada	A transição "*Resume*" corresponde a um "*Start*" com a diferença que a recuperação de um objeto contínuo (ou objetos ligados a essa transição) é realizada a partir do ponto que esse objeto foi pausado. Os objetos ligados a esse papel devem ser recuperados previamente para que a sua apresentação seja reiniciada simultaneamente, uma vez que possuem um relacionamento de sincronismo rígido ("*Sync*").
Resume_PleSync	Retomar/Continuar recuperação do objeto, se ela foi pausada	Os objetos de mídia ligados a essa transição são recuperados a partir do ponto que foram pausados. Porque esses objetos possuem um relacionamento de sincronismo fraco (*"PleSync"*) entre eles, a apresentação dos objetos ligados a esse papel não necessariamente tem que ser realizada de forma simultânea, mas é desejado que seja.

Tabela 2 - Recuperação do conteúdo frente as transições de evento.

O papel "*OnResume*" é tratado da mesma forma que o papel "*OnBegin*", com a diferença que no primeiro a recuperação dos conteúdos dos objetos de mídia ligados a esse papel é retomada após uma pausa, ao invés de recuperar

desde o início. De forma similar, o papel "*OnPause*" é tratado como o papel "*OnAbort*" com a diferença que no caso de "*OnPause*" a recuperação dos conteúdos de mídia ligados a esse papel é pausada, em lugar de ser abortada de forma definitiva.

Os eventos de atribuição também devem ser considerados na construção do plano de pré-busca porque esses eventos definem as propriedades do conteúdo audiovisual que deve estar sendo exibido.

Em relação aos eventos de seleção, o instante exato de suas ocorrências somente será conhecido durante a fase de execução. A recuperação prévia dos objetos de mídia ligados a essa transição deve ser prevista, com o intuito de evitar tempos de resposta muito grandes. Com a finalidade de tratar eventos não determinísticos o papel *"OnSelection"* poderia ser tratado de igual forma que o papel *"OnEnd"* considerando que em todos os casos a probabilidade de interação do usuário é igual a 1, isso é, o usuário sempre interage.

O evento *"OnSelection"* acontece quando uma tecla (a ser especificada) for pressionada enquanto o objeto ligado a esse papel estiver sendo apresentado. Porque o instante de tempo de ocorrência da seleção é imprevisível, neste trabalho assumimos que o instante de seleção acontece ao termino do objeto ligado a esse papel.

Conforme ao mencionado na Seção 2.1, os relacionamentos entre os objetos de mídia que compõem uma aplicação multimídia devem ser considerados no momento de definir a sua ordem de recuperação, para, dessa forma, diminuir a ocorrência de interrupções (*gaps*) durante a apresentação. O mecanismo de pré-busca proposto nesta dissertação considera uma aplicação multimídia formada tanto por eventos determinísticos como não determinístico, onde para definir a ordem de recuperação dos objetos de mídia, assim como os instantes de tempo de início para a recuperação dos conteúdos, são levados em conta os relacionamentos causais existentes entre os objetos de mídia.

Com o intuito de garantir a ausência de *gaps* na apresentação multimídia é fundamental calcular o retardo inicial que precisa ser inserido antes do início da apresentação para ter uma exibição contínua do conteúdo. Dependendo do tempo de chegada da aplicação à plataforma de apresentação, podem ser definidas as ações a serem tomadas para a entrega de uma apresentação de qualidade, visando sempre diminuir as interrupções, na medida possível. Esse retardo inicial é

denominado **Tret**, no presente trabalho, e é explicado na seção a seguir.

5.1
Retardo inicial para o início de uma aplicação multimídia (Tret)

Seja **Tret** o tempo que uma aplicação recebida demora para começar a sua apresentação sem que aconteçam *gaps*. Então, dependendo do instante de tempo no qual a aplicação é recebida na plataforma de apresentação, podem acontecer quatro casos:

1. A especificação da aplicação é enviada em qualquer momento antes do seu pedido de início, ou seja, a aplicação é recebida **Tret** segundos antes do seu pedido de início:
   - Se a aplicação é do tipo *Video on Demand* (VoD), o **Tret** pode ser alcançado inserindo uma propaganda, com um tempo de duração equivalente a **Tret**, no início.
   - Se a aplicação é de Televisão digital (TVD), a aplicação pode ser enviada mediante *push* com um tempo de antecedência igual a **Tret**.

Nos dois casos anteriores, o tempo **Tret** é calculado e a aplicação só começa a ser apresentada quando **Tret** é concluído.

2. A aplicação é recebida e o **Tret** é maior que o tempo de início da apresentação, porém menor que um *tempo de tolerância*, onde esse tempo de tolerância corresponde ao tempo máximo que o usuário pode aguardar pelo início de uma aplicação. Nesse caso, o **Tret** é calculado e o atraso de chegada da aplicação é acrescentado ao **Tret**. A aplicação só é apresentada depois de **Tret**. Neste trabalho, o tempo de tolerância possui um valor igual a 10 segundos, conforme ao explicado na Seção 4.4.

3. A aplicação é recebida e o **Tret**+tempo de tolerância é maior que o tempo de início de apresentação da aplicação. Nesse caso, o **Tret** é calculado e um tempo igual à tolerância máxima permitida é inserido, em seguida a apresentação é iniciada. Porque o atraso na chegada da aplicação à plataforma de exibição ultrapassou o tempo permitido sem

que haja *gaps*, as seguintes ações devem ser tomadas:

i) Tentar resolver com um ajuste elástico em tempo de compilação.

ii) Tentar resolver o *gap* "na sorte", isto é, esperando que algum retardo de recuperação seja menor que o esperado (note que se o tempo esperado for calculado como o pior caso, é bem provável que isso aconteça).

iii) Aplicar ajuste elástico em tempo de execução.

4. A aplicação tem que ser exibida em um tempo anterior à recepção de sua especificação. Nesse caso, a aplicação deve ser acelerada para então cair em um dos três casos mencionados acima. Nesta dissertação não trataremos deste caso.

Os três primeiros casos são mostrados na Figura 10, onde: t_{rec} é o instante de tempo no qual a aplicação é recebida na plataforma de exibição; t_{disp} é o instante de tempo em que a aplicação deveria ser disparada; t_{start} refere-se ao instante de tempo no qual a apresentação da aplicação é realmente iniciada e, finalmente, *tol* corresponde ao *tempo de tolerância*.

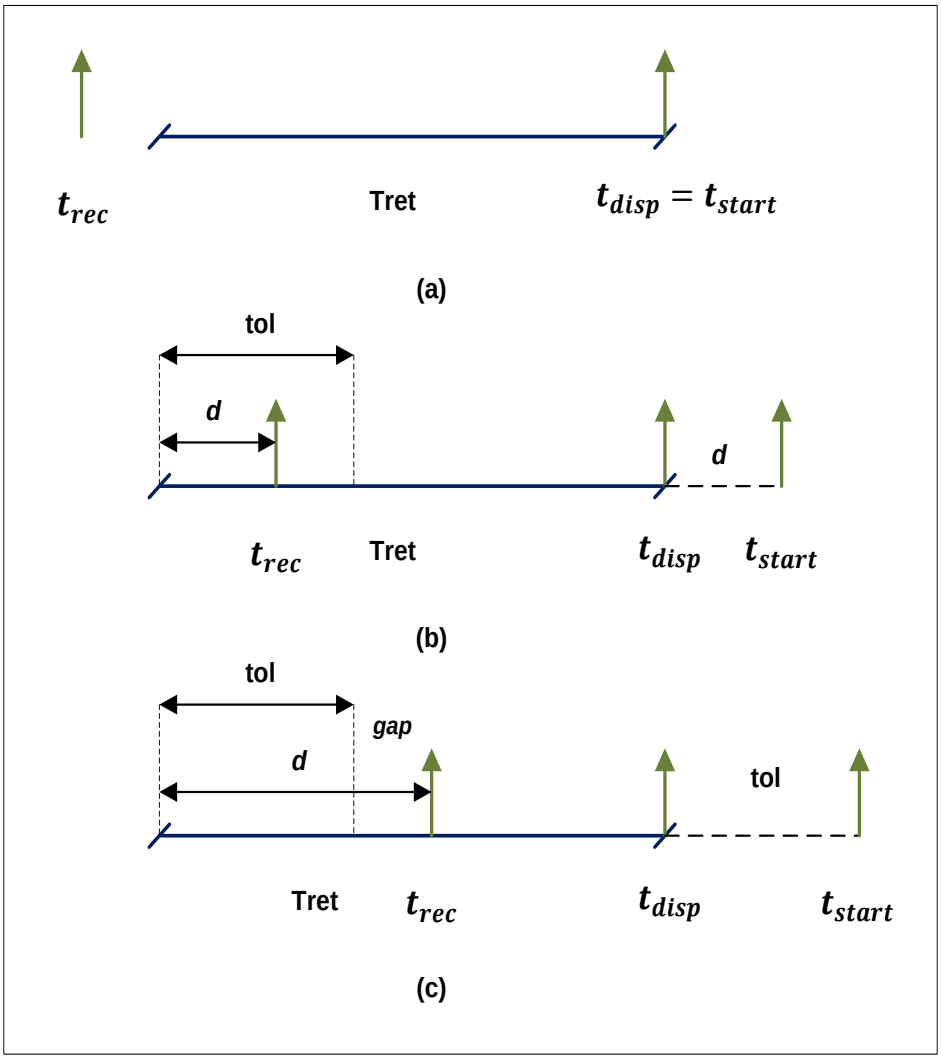


Figura 10 - Tempo de chegada da aplicação multimídia à plataforma de exibição: (a) $t_{disp} = t_{start}$ (a apresentação multimídia não possui gaps); (b) $t_{start} = t_{disp} + d$ e $d \leq tolerância$ (a apresentação multimídia não possui gaps, porém um retardo igual a d é inserido no início); (c) $t_{start} = t_{disp} + tol$ (um gap total equivalente a $t_{rec} - tol$ é inserido na apresentação).

5.2
Heurística de pré-busca

Os mecanismos de pré-busca são aplicados com o intuito de reduzir a latência total de exibição nas apresentações multimídia e recuperar os conteúdos dos objetos de mídia antes de seus tempos de apresentação, eliminando, assim, *gaps*. É importante ressaltar que a ordem de recuperação dos objetos possui um

papel determinante no aumento ou redução da latência de exibição dos objetos de mídia, assim como também influencia na redução do tempo de duração total da apresentação multimídia. Como já visto, em apresentações multimídia onde o sincronismo é baseado em eventos, eles devem ser considerados na definição da sequência de agendamento de pré-busca dos objetos de mídia. Os relacionamentos causais entre os objetos de mídia, o tipo de sincronismo e o critério "*Shortest Retrieval Time First*" são utilizados para construir uma heurística que permite determinar uma sequência adequada de recuperação

Nas heurísticas propostas nesta dissertação, antes de definir a sequência dos objetos de mídia e calcular os instantes de início de pré-busca de cada um deles, um passo é realizado. Esse passo consiste em dividir a apresentação multimídia em grupos menores, denominados ***Blocos de Sincronização,*** conceito herdado de (Kim et al. 2001). Cada Bloco de Sincronização (SB) é delimitado pelos instantes de início e fim dos objetos de mídia. Se um objeto de mídia contínua é dividido em segmentos, previamente definidos por um protocolo específico, cada um desses segmentos deve ser considerado como um objeto de mídia independente. A quebra dos objetos de mídia está ligada ao tipo do objeto e ao protocolo utilizado pelo servidor para a entrega de conteúdo ao cliente. A Figura 11 apresenta uma aplicação multimídia, Exemplo 2 anterior, formada por 12 objetos de mídia, tanto do tipo contínuo como discreto, na qual os objetos do tipo contínuo estão compostos por segmentos de mídia de tamanhos iguais.

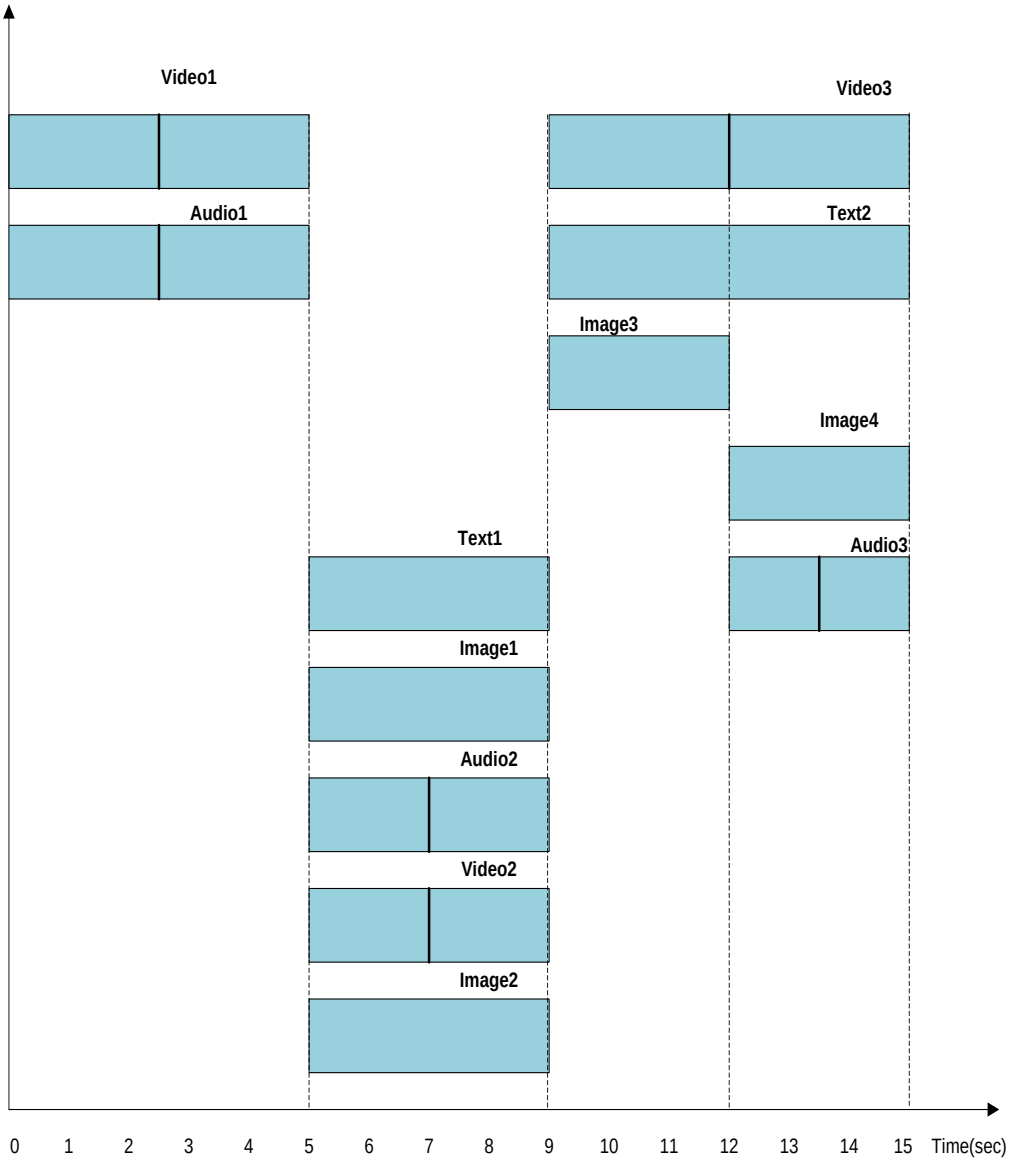


Figura 11 - Representação da apresentação no tempo do Exemplo2.

Os blocos de sincronização são definidos utilizando uma estrutura de dados que contém a informação referente às especificações temporais da apresentação multimídia. Essa estrutura é denominada ***Plano de Apresentação*** e possui os campos a seguir: transição (ação que executada), objeto de mídia, tempo de início, tempo de fim, tipo de mídia (1 - objeto de mídia contínua, 0 - objeto de mídia discreto, 2 - objeto de mídia contínua segmentado) e tempo de recuperação (Tabela 3). O Exemplo 2 corresponde a uma aplicação multimídia que só possui transições do tipo "*Start*". Posteriormente serão tratadas aplicações as quais possuem além da transição "*Start*", transições do tipo "*Stop*", "*Abort*", "*Pause*" e "*Resume*".

Transição	Objeto de mídia	Tempo de início (s)	Duração(s)	Tipo de mídia	Tempo de recuperação (s)
Start	Video1Seg1	0,00	2,50	2	2,00
Start	Audio1Seg1	0,00	2,50	2	1,50
Start	Video1Seg2	2,50	2,50	2	2,00
Start	Audio1Seg2	2,50	2,50	2	1,50
Start	Text1	5,00	4,00	0	1,00
Start	Image1	5,00	4,00	0	2,00
Start	Image2	5,00	4,00	0	3,00
Start	Video2Seg1	5,00	2,00	2	2,00
Start	Audio2Seg1	5,00	2,00	2	1,50
Start	Video2Seg2	7,00	2,00	2	2,00
Start	Audio2Seg2	7,00	2,00	2	1,50
Start	Video3Seg1	9,00	3,00	2	2,50
Start	Text2	9,00	6,00	0	2,00
Start	Image3	9,00	3,00	0	1,00
Start	Video3Seg2	12,00	3,00	2	2,50
Start	Image4	12,00	3,00	0	1,00
Start	Audio3Seg1	12,00	1,50	2	1,00
Start	Audio3Seg2	13,50	1,50	2	1,00

Tabela 3 - Plano de apresentação do Exemplo 2.

Nesse exemplo, a apresentação multimídia é dividida em 6 blocos de sincronização, como é mostrado na Figura 12. Uma vez encontrados os blocos de sincronização é possível estabelecer a ordem de recuperação adequada visando sempre reduzir os *gaps* inseridos durante a apresentação multimídia, assim como o tempo total de apresentação.

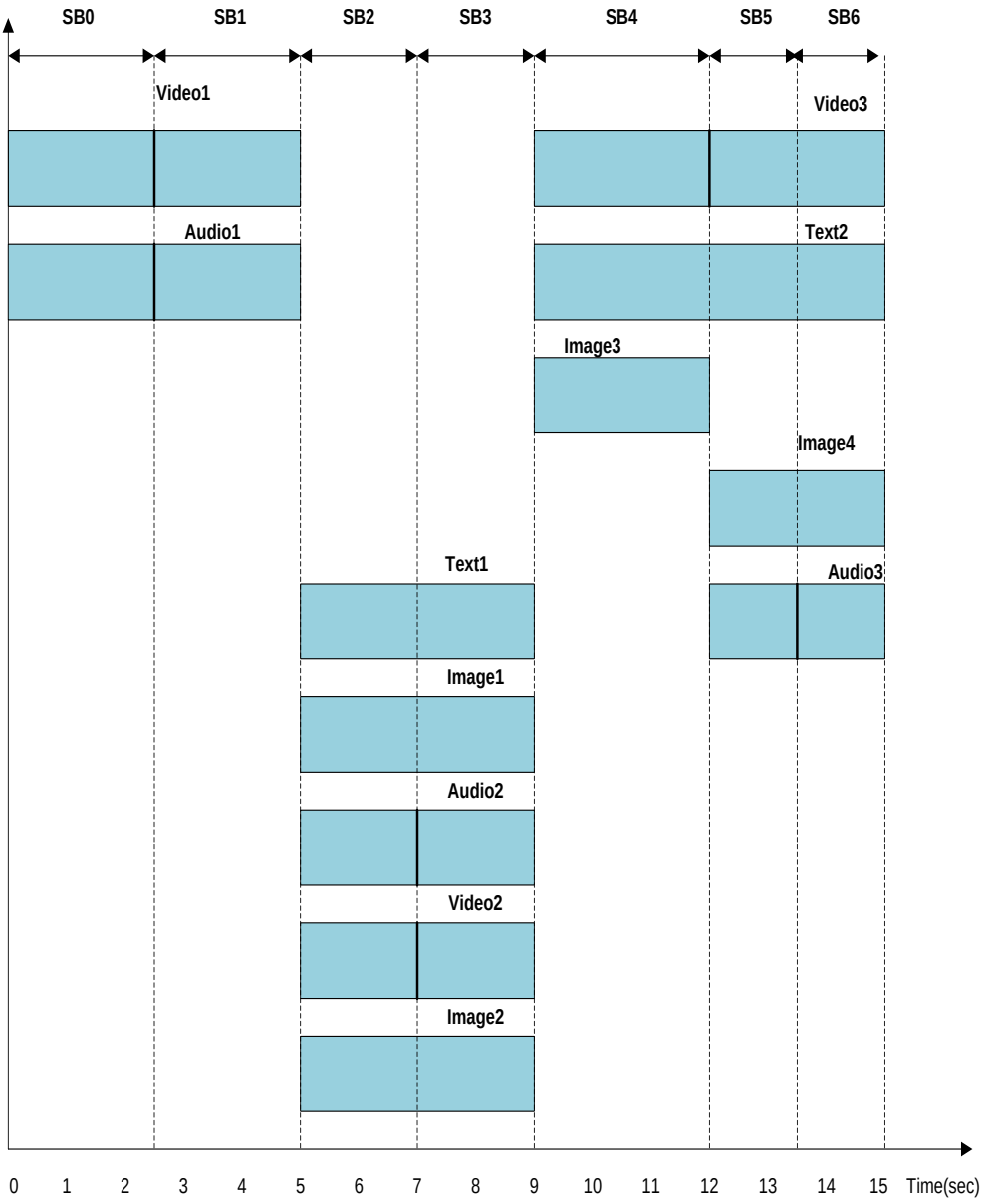


Figura 12 - Blocos de Sincronização do Exemplo 2.

Para definir a ordem dos objetos de mídia são levados em consideração os relacionamentos causais. Primeiro todos os objetos pertencentes a um mesmo bloco de sincronização são agrupados em ***Itens***. Um item é um objeto atômico constituído por um conjunto de objetos que possuem sincronismo **"*Sync*"** entre eles. Note que um item pode conter um ou mais objetos. Dentro de um Bloco de Sincronização, os itens têm um sincronismo fraco (*"PleSync"*) entre si.

O processo de pré-busca é formado por três etapas fundamentais: Encontrar os itens dos blocos de sincronização, ordená-los dentro de cada bloco para definir a sua ordem de recuperação, e calcular os instantes de tempo de início de recuperação para cada item que compõe a apresentação multimídia. O algoritmo

PrefetchStartTimes, apresentado a seguir, executa os algoritmos relacionados com as etapas mencionadas acima, na seguinte ordem: *FindItems*, *SortItems*, e *RetrievalTimes*.

Algorithm 1: PrefetchStartTimes

```
/* Calculate the optimal retrieval start times to all media
   objects of the multimedia presentation.                    */
Variables:
numOfBlocks: int: The number of Synchronization Blocks;
mo={MO_1, MO_2, ..., MO_n} : vector < string >:
The media objects of the multimedia presentation;
objectsOfSB[k]={MO_1, MO_2, ..., MO_n} : vector < string >:
The media objects j which belong to the SB[k];
ItemsOfSB[k]={}: vector< string >: The items of the SB[k];
ItemsInOrder={}: vector< string >: The items of the multimedia
presentation in retrieval order;
i:int;
1 begin
    /* Find the items of each SB                               */
2   for i=0 to numOfBlocks do
3     ItemsOfSB[i]:=FindItems(objectsOfSB[i]);
      /* Sort partially the items that are into each SB        */
4     ItemsInOrder=SortItems(ItemsOfSB[i],i);
5   end
    /* Calculate the retrieval start times to all the items    */
6   RetrievalTimes(ItemsInOrder);
7 end
```

Algoritmo 1 - PrefetchStartTimes.

Na linha 3 os itens de cada bloco de sincronização são encontrados; na linha 4 esses itens são ordenados. A função *SortItems* define uma ordem adequada para a recuperação dos itens. Uma vez que ordem de recuperação é estabelecida, a função *RetrievalTimes* calcula os instantes de tempo de início para a recuperação de cada um dos itens que compõem a apresentação multimídia. A complexidade matemática do algoritmo *PrefetchStartTimes* é igual a:

$$numOfBlocks * Complexidade\ Max\ (FindItems, SortItems) + Complexidade\ RetrievalStartTimes \quad (3)$$

5.2.1
Algoritmo *FindItems*

O algoritmo ***FindItems*** é encarregado de encontrar os itens de todos os bloco de sincronização. Para isso, o algoritmo utiliza uma estrutura de dados, denominada ***Plano de eventos,*** que contém a informação referente a todos os relacionamentos existentes entre os objetos de mídia que fazem parte da apresentação. A heurística proposta nesta dissertação considera que os segmentos que compõem um mesmo objeto de mídia possuem relacionamentos ***"Sync"*** entre si. A Tabela 4 apresenta o plano de eventos correspondente ao Exemplo 2.

Mídia - Papel	Papel	Transição	Mídia - Transição
Video1Seg1	OnBegin	Start_Sync	Audio1Seg1
Video1Seg1	OnEnd	Start_Sync	Video1Seg2
Audio1Seg1	OnEnd	Start_Sync	Audio1Seg2
Video1Seg2	OnEnd	Start_PleSync	Video2Seg1
Video2Seg1	OnBegin	Start_Sync	Audio2Seg1
Audio2Seg1	OnEnd	Start_Sync	Audio2Seg2
Video2Seg1	OnEnd	Start_Sync	Video2Seg2
Video1Seg2	OnEnd	Start_PleSync	Text1
Video1Seg2	OnEnd	Start_PleSync	Image1
Video1Seg2	OnEnd	Start_PleSync	Image2
Video2Seg2	OnEnd	Start_Sync	Video3Seg1
Video3Seg1	OnBegin	Start_Sync	Text2
Video3Seg1	OnEnd	Start_Sync	Video3Seg2
Video2Seg2	OnEnd	Start_PleSync	Image3
Image3	OnEnd	Start_PleSync	Image4
Image4	OnBegin	Start_Sync	Audio3Seg1
Audio3Seg1	OnEnd	Start_Sync	Audio3Seg2

Tabela 4 - Plano de eventos do Exemplo 2.

Algorithm 2: FindItems

```
/* Find the Items of the multimedia presentation.  An Item is
   defined as an atomic object that groups the media objects
   of a Synchronization Block which have a ''Sync" relation
   with each other.  All the items belong a Synchronization
   Block are asynchronous between them.                        */
1  FindItems : vector: < string > (objectsOfSB[i]: vector< string >)
   Variables:
   ItemsOfSB[k]:={}: The items of the SB[k];
2  begin
3      it =1;
4      for media object j ∈ ObjectsOfSB[i] do
5          visit[j]:= false;
6      end
       /* Find the items of SB[i]                                 */
7      for each media object j ∈ objectsOfSB[i] do
8          if visit[j] is false then
9              Item[it]:=mo(j);                  /* mo = media object */
10             visit[j]:=true;
11             for each media object p ∈ objectsOfSB[i] AND p ≠j do
12                 while media object j has an "OnBegin" role and a
                   "Start_Sync"' action with media object p do
13                     item[it]:=item[it] ∪ media object p;
14                     visit[p]:=true;
15                 end
16                 item[it] is added to ItemsOfSB[i];
17             end
18             it=it+1;
19         end
20     end
21     return [itemsOfSB[i]];
22 end
```

Algoritmo 2 - FindItems (Encontra os items de cada SB).

Dentro de um bloco de sincronização, um item é definido como um objeto atômico que é composto pelos objetos de mídia que possuem um relacionamento rígido (“*Sync*”) entre eles. Dentro de um item, esses objetos de mídia são chamados de elementos.

Cabe ressaltar que se entre os segmentos pertencentes de dois ou mais objetos de mídia contínua diferentes existe um sincronismo rígido, por inferência também existe um sincronismo rígido entre os segmentos sucessivos correspondentes a esses objetos.

A linha 8 verifica se um objeto de mídia *j* já foi visitado. Se esse objeto ainda não foi visitado, ele é colocado dentro de um item. Depois, procura-se se

existem outros objetos de mídia, dentro desse SB, que possuam um relacionamento "*Sync*" com ele. Todos os objetos de mídia que possuam um relacionamento "*Sync*" entre si são agrupados dentro de um mesmo item. Esse processo é realizado entre as linhas 11 e 17. Finalmente, na linha 21 o algoritmo ***FindItems*** retorna todos os itens de um bloco de sincronização determinado. A Tabela 7 mostra os itens do Exemplo 2.

A complexidade matemática do algoritmo *FindItems* é $O(n^3)$. Porém segundo (Kim et al. 2001) devido às limitações relacionadas com a percepção humana para reconhecer os objetos de mídia apesentados de forma simultânea, o número de objetos de mídia *n* deve ser pequeno ($n < 10$) na prática. De modo que no pior caso teremos 1000 iterações.

Bloco de Sincronização	Item	Elementos de Item
SB[0]	Item0:	Video1Seg1
		Audio1Seg1
SB[1]	Item0:	Video1Seg2
		Audio1Seg2
SB[2]	Item0:	Video2Seg1
		Audio2Seg1
	Item1:	Image1
	Item2:	Image2
	Item3:	Text1
SB[3]	Item0:	Video2Seg2
		Audio2Seg2
SB[4]	Item0:	Video3Seg1
		Text2
	Item1:	Image3
SB[5]	Item0:	Video3Seg2
	Item1:	Image4
		Audio3Seg1
SB[6]	Item0:	Audio3Seg2

Tabela 5 - Itens do Exemplo 2.

5.2.2
Algoritmo *SortItems*

Uma vez que os itens de cada bloco de sincronização são encontrados, eles são ordenados considerando: os relacionamentos causais, o tempo de recuperação e o número de elementos que possui cada item. O algoritmo heurístico ***SortItems*** é apresentado a seguir.

Algorithm 3: SortItems

```
/* Sort the items of each Synchronization Block. Firstly,
   all the items are grouped together regarding their
   relationships with the past and the future of the
   multimedia application.Then, the items of each group are
   sorted.                                                  */
1 SortItems: vector: < string > (ItemsOfSB[i]: vector< string >,i:int)
  Variables:
  numOfBlocks: int: The number of Synchronization Blocks;
  ItemsGroup1[k][j]:= {item_1, item_2, ..., item_n} :
  The items j of the SB[k] that belong to Group 1;
  ItemsGroup2[k][j]:= {item_1, item_2, ..., item_n} :
  The items j of the SB[k] that belong to Group 2;
  ItemsGroup3[k][j]:= {item_1, item_2, ..., item_n} :
  The items j of the SB[k] that belong to Group 3;
  ItemsGroup4[k][j]:= {item_1, item_2, ..., item_n} :
  The items j of the SB[k] that belong to Group 4;
  ItemsGroup5[k][j]:= {item_1, item_2, ..., item_n} :
  The items j of the SB[k] that belong to Group 5;
  RtOfItems(j): The retrieval times of item j;
  ItemsInOrder={}: vector< string >: The items of the multimedia
  presentation in retrieval order;
2 Function Sort (ItemsOfGroup[i]);
  Variables:
  ItemsEqualRt:={}: Subset of items that have the same retrieval time in a
  Synchronization Block;
3 begin
4   Sort all items j ∈ ItemsOfGroup[i] in a ascending order to the
    retrieval time value;
5   if ItemsEqualRt ≠ ∅ then
6     Sort ItemsEqualRt in descending order to the number of
      elements that are contained in each item k ∈ ItemsEqualRt;
7   end
8 end
```

```
begin
    /* Define if item j belongs to G1,G2,G3,G4 or G5          */
    /* First synchronization block                            */
    if i==0 then
        for each item j ∈ ItemsOfSB[i] do
            if item j has a "Sync" relation with other item p ≠ j in the
            future then
                Add item j into ItemsGroup1[i][j];
            end
            else if item j has a "PleSync" relation with other item p ≠
            j in the future then
                Add item j into ItemsGroup2[i][j];
            end
            else
                Add item j into ItemsGroup5[i][j];
            end
        end
        Sort (ItemsGroup1[i]);
        Sort (ItemsGroup2[i]);
        Sort (ItemsGroup5[i]);
        ItemsGroup1[i], ItemsGroup2[i] and ItemsGroup5[i] are added to
        ItemsInOrder;
    end
    else
        for each item j ∈ ItemsofSB[i] do
            if item j has a "Sync" relation with other item p ≠ j in the
            past then
                Add item j into ItemsGroup3[i][j];
            end
            else if item j has a "Sync" relation with other item p ≠ j in
            the future then
                Add item j into ItemsGroup1[i][j];
            end
            else if item j has a "PleSync" relation with other item p ≠
            j in the past AND item j does not have a "Sync" relation
            with other item q ≠ j in the future then
                Add item j into ItemsGroup4[i][j];
            end
            else if item j has a "PleSync" relation with other item p ≠
            j in the future then
                Add item j into ItemsGroup2[i][j];
            end
            else
                Add item j into ItemsGroup5[i][j];
            end
        end
        Sort (ItemsGroup3[i]);
        Sort (ItemsGroup1[i]);
        Sort (ItemsGroup4[i]);
        Sort (ItemsGroup2[i]);
        Sort (ItemsGroup5[i]);
        ItemsGroup3[i],ItemsGroup1[i],ItemsGroup4[i], ItemsGroup2[i],
        and ItemsGroup5[i] are added to ItemsInOrder;
    end
    return [ItemsInOrder];
end
```

Algoritmo 3 - SortItems (Define a ordem dos itens de um SB).

Como é tão importante manter o sincronismo com o passado quanto com o futuro, o algoritmo ***SortItems*** reúne, em uma primeira etapa, os itens pertencentes a um SB em cinco grupos: G1, G2, G3, G4 e G5. Onde G1 e G2 contêm os itens que possuem alguma relação com o futuro, enquanto que G3 e G4 agrupam os itens que possuem relação com o passado da apresentação multimídia. Finalmente, em G5 encontram-se os itens que não possuem relação nem com o passado nem com o futuro, isto é, possuem apenas uma relação com algum item pertencente ao bloco de sincronização atual.

Todos os itens que possuem um relacionamento ***"Sync"*** com algum item que pertence a um bloco de sincronização futuro fazem parte de G1 e, em consequência, são agrupados no vetor **ItemsGroup1**. Os itens que possuem um relacionamento ***"PleSync"*** com algum item que pertence a um SB futuro fazem parte de G2, e são agrupados no vetor **ItemsGroup2**. Por outro lado, os itens que possuem um relacionamento ***"Sync"*** com algum item que pertence a um bloco de sincronização passado fazem parte de G3 e são agrupados no vetor **ItemsGroup3**. Os itens que possuem um relacionamento "***PleSync"*** com algum item que pertence a um SB passado fazem parte de G4 e são agrupados no vetor **ItemsGroup4**. Por último, os itens que não possuem nenhuma relação com o futuro nem com o passado são agrupados no vetor **ItemsGroup5**.

Uma vez que os itens são agrupados de acordo com os seus relacionamentos causais, é preciso definir a ordem de recuperação de cada um desses grupos.

Como os itens pertencentes ao primeiro bloco de sincronização não possuem passado, eles não podem ser ordenados usando esse critério. Em consequência, para a ordenação desses itens, só é considerada a sua relação com o futuro (linhas 10 - 26). Os grupos pertencentes ao primeiro bloco de sincronização são recuperados como se segue:

1) Grupo 1 (Itens que possuem uma relação "***Sync***" com algum item no futuro).
2) Grupo 2 (Itens que possuem uma relação "***PleSync***" com algum item no futuro).
3) Grupo 5 (Itens que possuem alguma relação com algum item que pertence ao SB atual).

O Grupo 1 é recuperado primeiro porque dessa forma é possível diminuir a latência de exibição que pode ser inserida entre os itens ligados a esse sincronismo

rígido.

A ordenação dos itens pertencentes aos demais blocos de sincronização SB[i], onde $i \neq 0$, considera tanto o passado quanto o futuro (linhas 27 - 51). A ordem de recuperação dos grupos pertencentes a esses blocos é como segue:

1) Grupo 3 (Itens que possuem uma relação "***Sync***" com o passado).
2) Grupo 1 (Itens que possuem uma relação "***Sync***" com o futuro).
3) Grupo 4 (Itens que possuem uma relação "***PleSync***" com o passado).
4) Grupo 2 (Itens que possuem uma relação "***PleSync***" com o futuro).
5) Grupo 5 (Itens que possuem relação com algum item que pertence ao SB atual).

Os itens que possuem um sincronismo rígido com o passado possuem uma prioridade de recuperação sobre os demais pois, dessa forma, a latência de exibição entre itens do tipo "***Sync***" pode ser reduzida ou evitada. Porém, logo após a recuperação desses itens, os itens que possuem um sincronismo fraco como o passado são recuperados. Embora o sincronismo fraco ofereça maior flexibilidade, a apresentação dos itens que possuem esse tipo de sincronismo também deve se aproximar, na medida do possível, das especificações do autor da aplicação. Por outro lado, é interessante mencionar que, se um item possui um relacionamento do tipo "***PleSync***" com o passado e um relacionamento "***Sync***" com o futuro, é mais importante manter o seu sincronismo com o futuro que com o passado, porque o sincronismo com o futuro é do tipo rígido (linha 35).

A prioridade de recuperação nos diferentes grupos que fazem parte de um SB é definida no momento de ordenar os itens que pertencem a cada um desses grupos. A função ***Sort*** ordena os itens de cada grupo considerando dois critérios: tempo de recuperação do item e número de objetos de mídia que compõem um item.

Como definimos na seção anterior, um item pode ser composto de 1 ou mais objetos de mídia que possuem um relacionamento ***"Sync"*** entre si. Primeiramente, todos os itens pertencentes a um grupo são ordenados em ordem crescente com relação a seu tempo de recuperação: os itens que possuem menor tempo de recuperação serão recuperados primeiro (linha 4). No caso de existir empate entre dois ou mais itens, isto é, possuírem o mesmo tempo de recuperação, eles são agrupados em um vetor auxiliar chamado ***ItemsEqualRt.*** Esses itens são ordenados em forma decrescente ao número de objetos de mídia que possui: o

item que está composto pelo maior número de objetos de mídia será recuperado primeiro (linha 6).

Finalmente, a ordem dos itens de um bloco de sincronização é equivalente a:

- Para $SB[i]\ onde\ i = 0$,a ordem dos itens é : $\boldsymbol{G1} \cup \boldsymbol{G2} \cup \boldsymbol{G5}$.
- Para $SB[i]\ onde\ i \neq 0$, a ordem dos itens é: $\boldsymbol{G3} \cup \boldsymbol{G1} \cup \boldsymbol{G4} \cup \boldsymbol{G2} \cup \boldsymbol{G5}$.

O algoritmo ***SortItems*** retorna o vetor ***ItemsInOrder,*** que contém todos os itens que fazem parte da apresentação multimídia em sua ordem de recuperação.

A complexidade do algoritmo *SortItems* é equivalente à de ordenar os itens pertencentes a cada grupo. Se utilizarmos o algoritmo *Counting Sort*, no melhor caso os itens são ordenados em complexidade O(n) e no pior caso são ordenados em O(n log n).

Conforme ao mencionado na seção anterior, o número máximo de objetos de mídia por bloco de sincronização é 10. Ou seja, é possível ordenar os itens em tempo lineal. Logo, a complexidade do algoritmo *SortItems* é igual a O(n).

A próxima seção discute, em mais detalhes, as considerações e critérios para a ordenação dos itens.

5.2.2.1 Considerações sobre a ordenação dos Itens de um mesmo grupo

Para análise do algoritmo ***SortItems,*** vamos utilizar o cenário apresentado na Figura 13. O objetivo é agendar um conjunto de itens que pertencem a um bloco de sincronização *k.* Vamos supor que todos esses itens encontram-se dentro do grupo **G1**, isto é, possuem um relacionamento ***"Sync"*** com algum ou alguns itens pertencentes a um SB futuro. Além disso, vamos assumir que existe um intervalo de tempo equivalente a 6 segundos para a recuperação desses itens, correspondente ao tempo de apresentação do item anterior, na Figura 13, o item 1.

O primeiro passo é selecionar o maior número de itens que possam ser recuperados no intervalo de tempo disponível. Como observamos na Figura 13, existem quatro itens que devem ser satisfeitos. Para satisfazer o maior número de itens, o tempo de recuperação deve ser otimizado.

Denota-se ***Trecuperação_disp*** como o tempo disponível para a recuperação

dos itens de um bloco de sincronização *k*. Esse tempo é definido como a diferença entre o tempo de início de $SB[k]$ e o tempo em que o último item do bloco anterior (k-1) é recuperado. Seja $\sum_{i=1}^{n} TR_{item_i}$ a somatória de todos os tempos de recuperação pertencentes aos objetos de mídia de $SB[k]$. Dois casos são possíveis:

1) $\sum_{i=1}^{n} TR_{item_i} \leq Trecuperação\_disp$

2) $\sum_{i=1}^{n} TR_{item_i} > Trecuperação\_disp$

No primeiro caso, todos os itens podem ser recuperados antes do tempo de início de $SB[k]$, garantindo, assim, a ausência de *gaps*.

No segundo caso, o tempo de recuperação disponível não é suficiente para recuperar todos os itens antes do tempo de início de $SB[k]$. Nesse caso, uma heurística é proposta com a finalidade de recuperar o maior número de itens possíveis. Para isto, os itens são ordenados em forma crescente com relação a seu tempo de recuperação. Dessa forma, o maior número de itens podem ser recuperados até que ***Trecuperação_disp*** seja ocupado. No caso em que dois ou mais itens possuam o mesmo tempo de recuperação, eles são ordenados de forma descendente com relaçõ ao número de seus elementos. Isso é, já que os tempos de recuperação são iguais, é preferível recuperar o item que possui o maior número de elementos.

Os itens que conseguirem ser recuperados em ***Trecuperação_disp*** são apresentados no instante de tempo certo $(gap = 0)$, nos outros casos, é inserido um atraso na sua apresentação $(gap \neq 0)$.

Assim, se um $item_i$ possui um relacionamento ***"Sync"*** com o passado ou com o futuro, procura-se reduzir o atraso na sua apresentação. Por outro lado, se um item for **"*PleSync",*** não é preciso considerar esse atraso. Contudo, com o intuito de evitar atrasos muito grandes na apresentação de itens do tipo **"*PleSync",*** um tempo limite de apresentação deve ser definido para que a sua exibição não seja adiada de tal forma que o atraso de exibição chegue a ser tão grande que não valha mais a pena a sua apresentação. Esse tempo limite não é definido nem levado em conta neste trabalho, porém deveria ser considerado em trabalhos futuros.

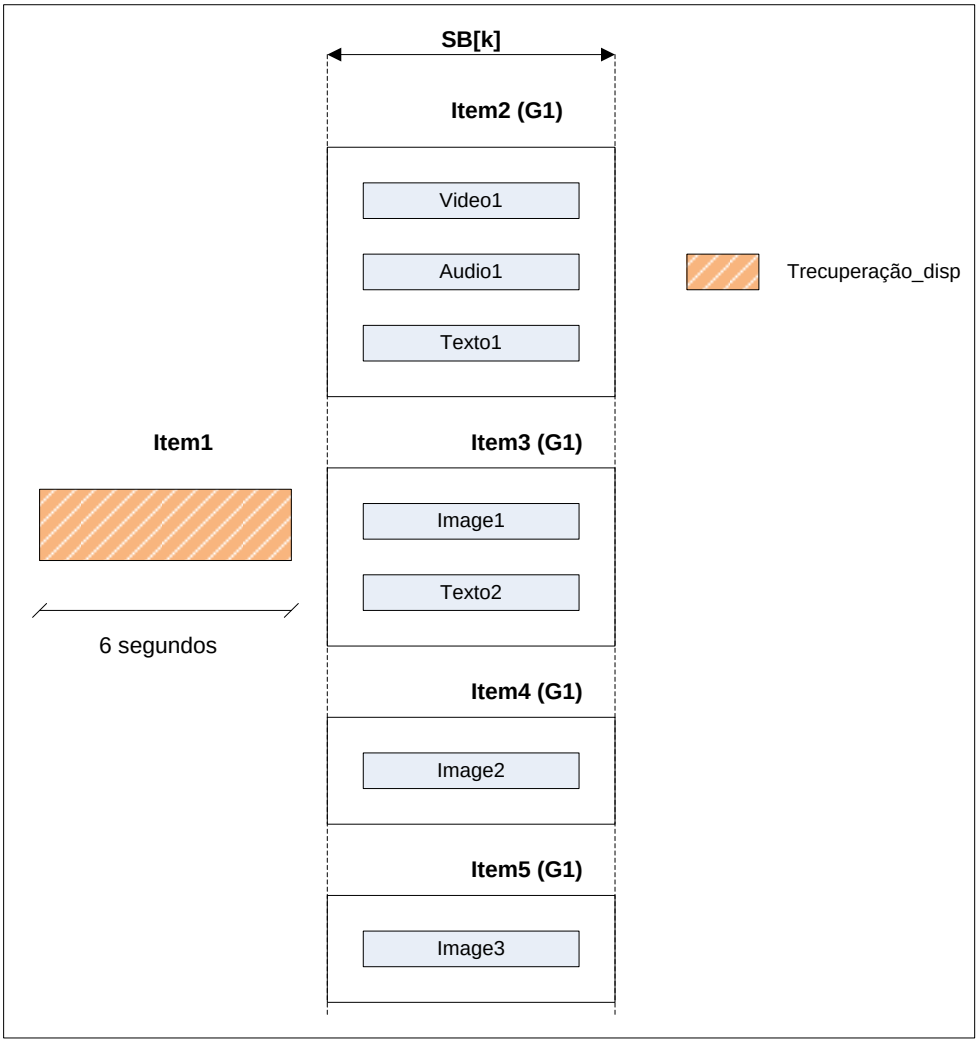


Figura 13 - Definir a ordem dos Itens de SB[k].

É importante ressaltar que o critério para ocupar de forma adequada ***Trecuperação_disp*** é considerar as $n!$ combinações dos itens que pertencem a um bloco de sincronização. A Figura 14 mostra todas as possíveis combinações dos itens do $SB[k]$, onde só quatro combinações podem ser recuperadas em ***Trecuperação_disp.*** A combinação mais adequada é aquela que encontra-se com a borda de cor azul porque, além de reduzir o tempo de recuperação, aumenta o número de itens que podem ser recuperados no intervalo de tempo. Aplicando a heurística ***SortItems*** no exemplo mostrado na Figura 14, é possível obter a sequência de itens a seguir: item3, item2, item4 e item1. Note que a combinação item3 – item2 é uma aproximação válida, uma vez que faz parte de uma das soluções adequadas.

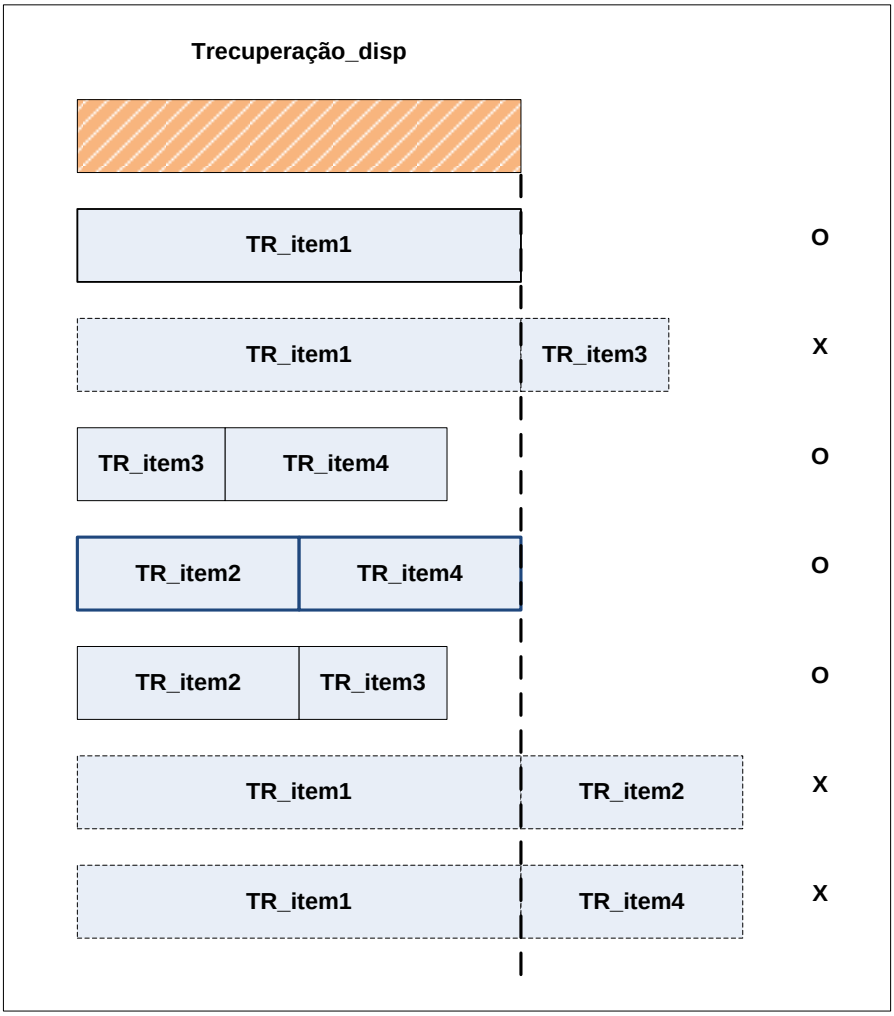


Figura 14 - Cálculo do número máximo de itens que podem ser recuperados em Trecuperação_disp.

5.2.3
Algoritmo *RetrievalTimes*

Estabelecida a sequência de recuperação dos itens, o algoritmo ***RetrievalTimes*** é aplicado para determinar os instantes de início de recuperação de cada um dos itens que fazem parte da apresentação multimídia.

Algorithm 4: RetrievalTimes

```
   /* Calculate the retrieval start time to each item that
      belongs to the multimedia presentation.                  */
1  Void RetrievalTimes (ItemsInOrder);
   Variables:
   RetrievalStartTimes[j]: The retrieval start time of item j;
   RtOfItems[j]: The retrieval time of item j;
2  begin
3  |  RtOflastItem:= 0;
4  |  for each item j ∈ ItemsInOrder do
5  |  |  RetrievalStartTimes[j]:= RtOflastItem;
6  |  |  RtOflastItem:=RtOflastItem + RtOfItem[j];
7  |  end
8  end
```

Algoritmo 4 - RetrievalTimes (Calcula os tempos de início de recuperação dos itens).

Os tempos de início de recuperação de cada item são armazenados no vetor ***RetrievalStartTimes***. O primeiro item é recuperado no instante de tempo igual a zero. A soma do tempo de início de recuperação do item anterior (*RtOflastItem*) e o seu tempo de recuperação (*RtOfItem*) é atribuída ao instante de recuperação do próximo item.

Para alcançar uma apresentação contínua e a fim de respeitar as especificações do autor da aplicação, tempos de espera são inseridos entre o intervalo de tempo compreendido pelo instante no qual um item termina de ser recuperado e o instante que ele deve ser apresentado.

A complexidade do algoritmo *RetrievalStartTimes* é O(n). Substituindo as complexidades das funções que compõem o algoritmo *PrefetchStartTimes* na equação 3 temos:

$$numOfBlocks * Max\ (O(n^3), O(n \log n)\ ) + O(n)$$

$$numOfBlocks * O(n^3) + O(n)$$

Seja m=*numOfBlocks*, logo a complexidade do algoritmo *PrefetchStartTimes* é $O(mn^3)$.

O resultado de aplicar a técnica de pré-busca empregando o algoritmo ***PrefetchStartTimes*** no Exemplo 2 (Figura 11) é apresentado na Figura 15. A ordem de recuperação dos itens e os seus tempos de início de recuperação são apresentados na Tabela 6.

Item	Tempo de início de recuperação
Video1Seg1 Audio1Seg1	0,00
Audio1Seg2	3,50
Video1Seg2	5,00
Video2Seg1 Audio2Seg1	7,00
Text1	10,50
Image1	11,50
Image2	13,50
Audio2Seg2	16,50
Video2Seg2	18,00
Video3Seg1 Text2	20,00
Image3	24,50
Video3Seg2	25,50
Image4 Audio3Seg1	28,00
Audio3Seg2	30,00

Tabela 6 - Tempos de início de recuperação dos itens do Exemplo 2.

Diferente do algoritmo desenvolvido em (Kim et al. 2001), o algoritmo de pré-busca apresentado no presente trabalho diminui a latência de exibição por levar em consideração o sincronismo está baseado em eventos definido em NCL e considerar a existência tanto de sincronismo rígido (***"Sync"***) como de sincronismo fraco (***"PleSync"***) no momento de definir a ordem de recuperação dos objetos de mídia que compõem uma aplicação. No Exemplo 2 a latência de exibição é igual a 4.5 segundos, 13.5 segundos menos que o resultado obtido aplicando o algoritmo de Kim, onde a latência resultante é de 18 segundos. A Figura 16 apresenta o resultado de aplicar o algoritmo de Kim no Exemplo2.

Essa redução na latência de exibição é alcançada devido a dois fatores fundamentais: a existência de eventos **"*PleSync*"** na apresentação multimídia e porque uma prioridade de recuperação é estabelecida para o conteúdo de mídia que possui um relacionamento **"*Sync*"** com o futuro ou com o passado da

aplicação. No caso em que uma aplicação multimídia esteja composta unicamente por eventos **"*Sync*"**, isto é, os itens pertencentes a um mesmo bloco de sincronização possuem um relacionamento **"*Sync*"** entre si e os blocos de sincronização também são **"*Sync*"** um em relação ao outro, tanto a latência de apresentação como a duração da apresentação multimídia resultam iguais tanto usando o algoritmo de Kim quanto o algoritmo apresentado neste trabalho. Isto é devido ao fato de todos os itens serem **"*Sync*"** e possuírem a mesma prioridade de recuperação, além de que itens de um mesmo SB deverem ser apresentados de forma simultânea. Em consequência, para que a sua apresentação seja iniciada todos os itens devem ser recuperados previamente, inserindo inevitavelmente um *gap* entre os blocos de sincronização.

Note-se, na Figura 15, que ao terminar a apresentação de Video1Seg2, cinco objeto de mídia devem ser apresentados: Text1, Image1, Image2, Video2Seg1 e Audio2Seg1. Todos esses objetos possuem uma relação ***"PleSync"*** com Video1Seg2, isto é, esses objetos devem ser exibidos quando a apresentação de Video1Seg2 termina, porém não necessariamente imediatamente após do seu termino. Além disso, esses objetos não precisam ser apresentados de forma simultânea. Eles são apresentados assim que forem recuperados. Como o objeto Video2Seg1 e Audio2Seg1 possuem uma relação ***"Sync"*** com o futuro, esses dois objetos são recuperados primeiro, para evitar que uma latência de exibição seja inserida com objetos de mídia que possuem sincronismo rígido. O *gap* inserido entre Video1Seg2 e esses dois últimos objetos não é considerado, porque o autor da aplicação estabeleceu um sincronismo fraco entre eles.

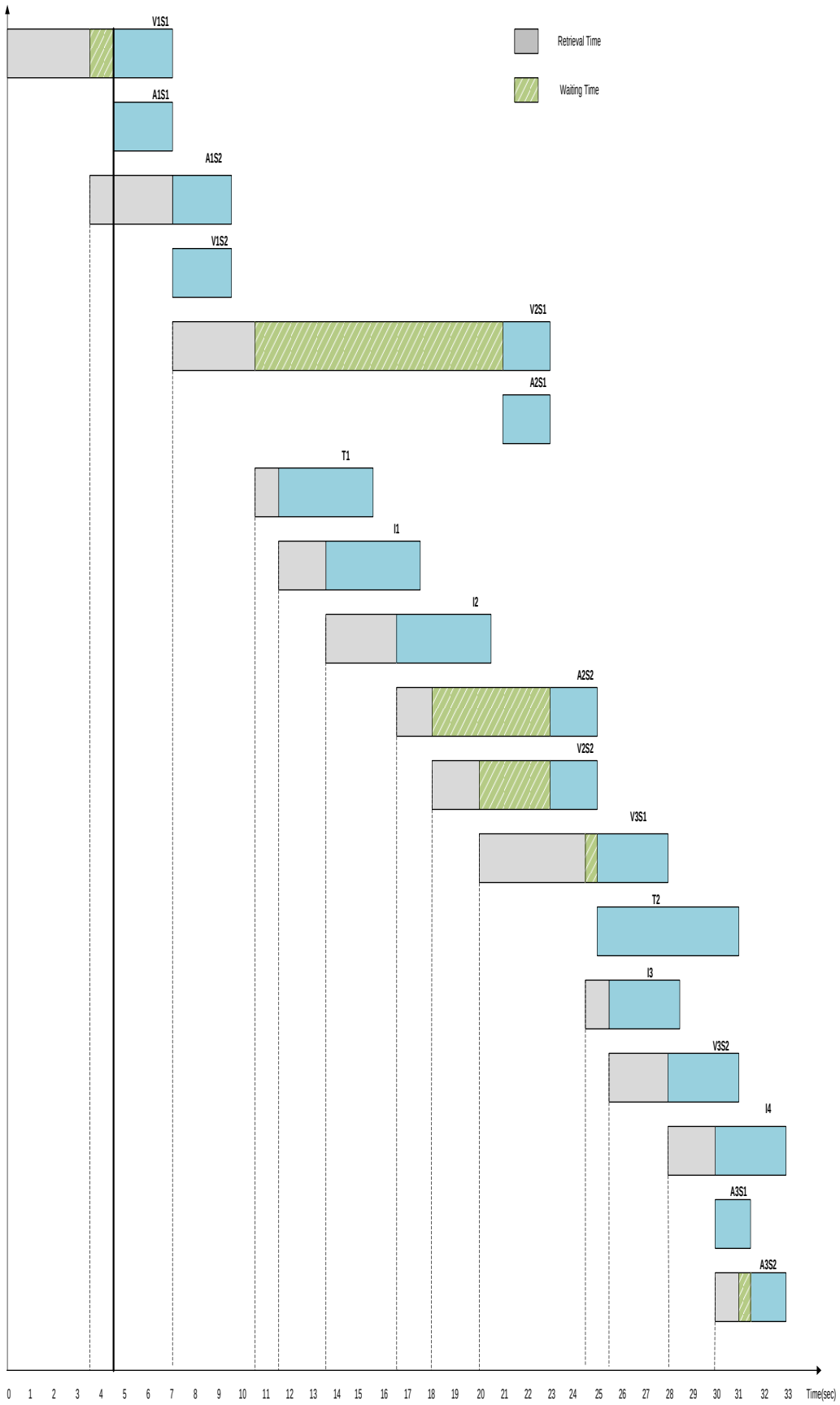


Figura 15 - Resultado de aplicar o algoritmo *PrefetchStartTimes* no Exemplo 2.

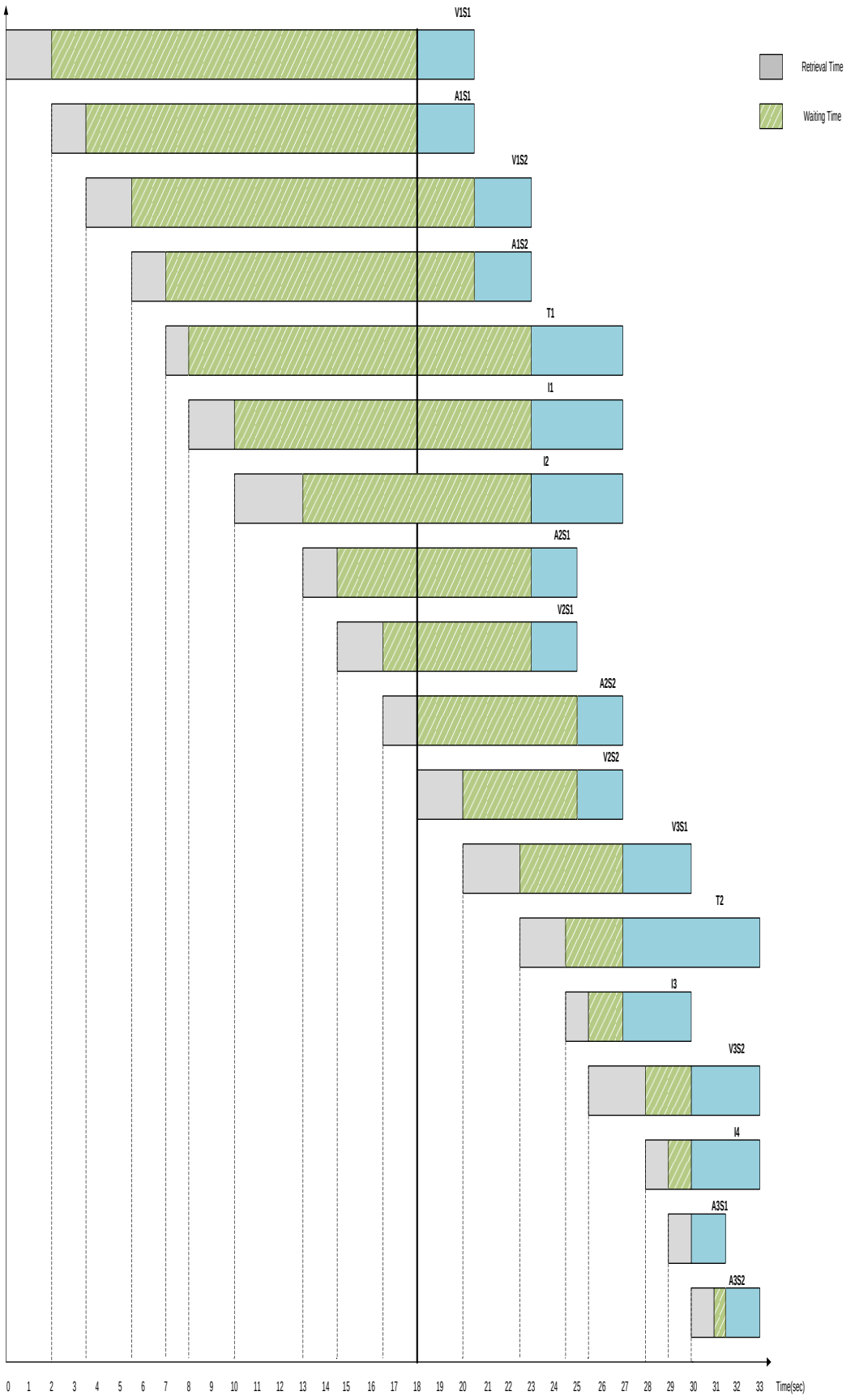


Figura 16 - Pré-busca do Exemplo 2 conforme ao algoritmo de Kim (Kim et al. 2001).

A Figura 17 apresenta outra aplicação, Exemplo 3, formada por 13 objetos de mídia, tanto do tipo contínuo como discreto. A Tabela 7 e 8 mostram o Plano de Apresentação e o Plano de Eventos dessa aplicação, respectivamente. A Figura 18 apresenta a aplicação dividida em 9 blocos de sincronização.

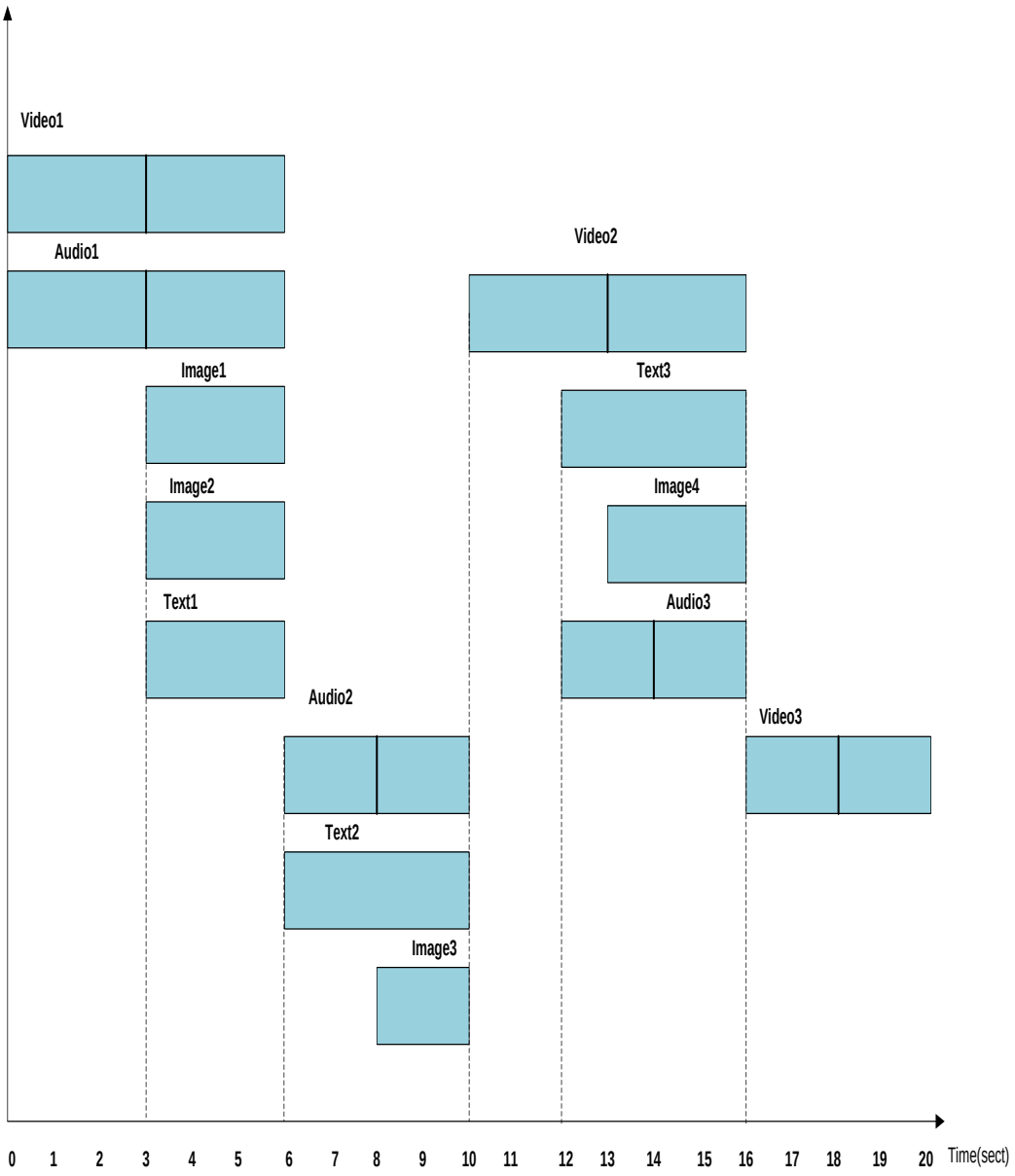


Figura 17 - Representação absoluta do Exemplo3.

Transição	Objeto de Mídia	Tempo de início (s)	Duração(s)	Tipo de mídia	Tempo de recuperação (s)
Start	Video1Seg1	0,00	3,00	2	2,50
Start	Audio1Seg1	0,00	3,00	2	2,00
Start	Image2	3,00	3,00	0	2,00
Start	Text1	3,00	3,00	0	1,50
Start	Image1	3,00	3,00	0	1,00
Start	Audio1Seg2	3,00	3,00	2	2,00
Start	Video1Seg2	3,00	3,00	2	2,50
Start	Audio2Seg1	6,00	2,00	2	1,50
Start	Video2Seg1	10,00	3,00	2	2,50
Start	Text3	12,00	4,00	0	0,50
Start	Image4	13,00	3,00	0	1,00
Start	Audio3Seg1	12,00	2,00	2	2,00
Start	Audio3Seg2	14,00	2,00	2	2,00
Start	Video2Seg2	13,00	3,00	2	2,50
Start	Video3Seg1	16,00	2,00	2	1,50
Start	Video3Seg2	18,00	2,00	2	1,50

Tabela 7 - Plano de apresentação do Exemplo 3.

Mídia - Papel	Papel	Transição	Mídia - Transição
Video1Seg1	OnBegin	Start_Sync	Audio1Seg1
Video1Seg1	OnBegin	Start_Sync	Image1(delay 3s)
Image1	OnBegin	Start_PleSync	Text1
Image1	OnBegin	Start_PleSync	Image2
Video1Seg1	OnEnd	Start_Sync	Video1Seg2
Audio1Seg1	OnEnd	Start_Sync	Audio1Seg2
Video1Seg2	OnEnd	Start_PleSync	Audio2Seg1
Audio2Seg1	OnBegin	Start_Sync	Text2

Audio2Seg1	OnEnd	Start_Sync	Audio2Seg2
Text2	OnBegin	Start_PleSync	Image3 (delay 2s)
Image3	OnEnd	Start_Sync	Video2Seg1
Video2Seg1	OnBegin	Start_Sync	Text3(delay 2s)
Text3	OnBegin	Start_PleSync	Image4 (delay 1s)
Video2Seg1	OnBegin	Start_Sync	Audio3Seg1
Video2Seg1	OnEnd	Start_Sync	Video2Seg2
Audio3Seg1	OnEnd	Start_Sync	Audio3Seg2
Image4	OnEnd	Start_Sync	Video3Seg1
Video3Seg1	OnEnd	Start_Sync	Video3Seg2

Tabela 8 - Plano de Eventos do Exemplo 3.

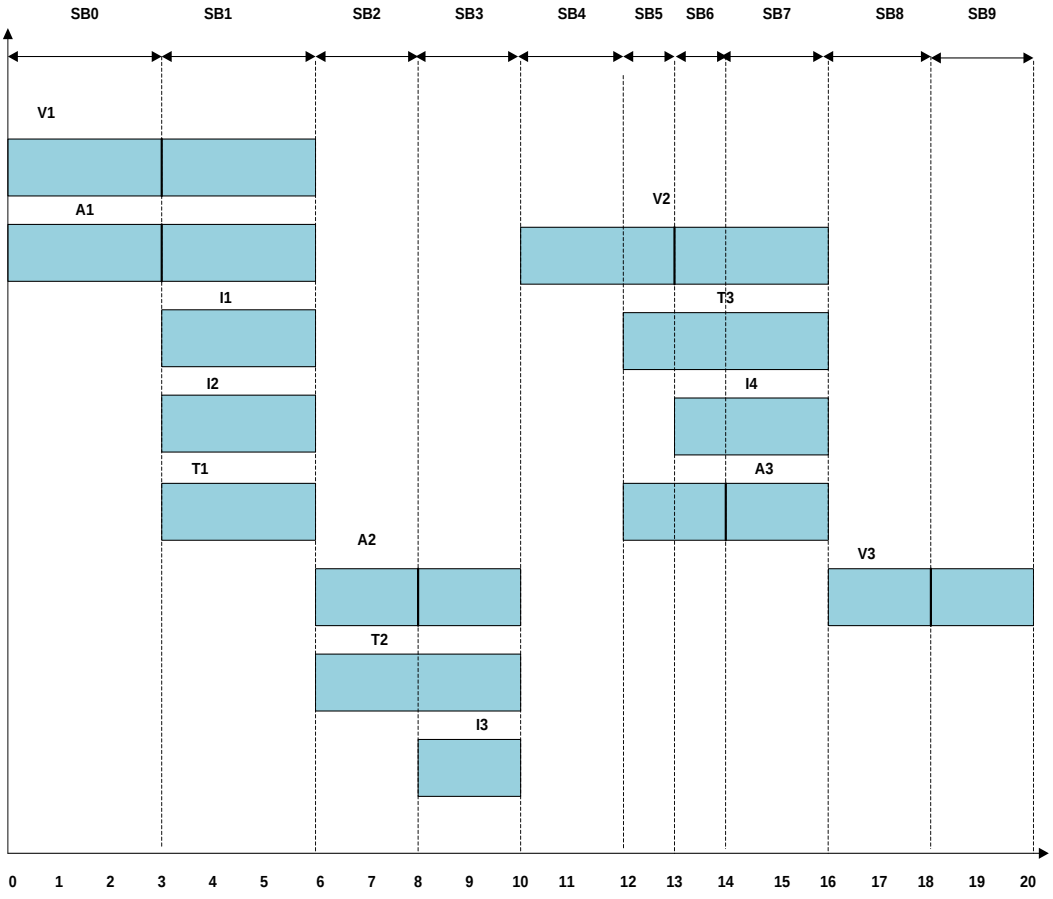


Figura 18 - Blocos de Sincronização do Exemplo 3.

Aplicando o algoritmo desenvolvido em Kim a latência de exibição é avaliada como 15 segundos, enquanto que aplicando o algoritmo de pré-busca desenvolvido no presente trabalho a latência de exibição resultou em 7 segundos, 8 segundos menos que Kim. A Tabela 9 apresenta os itens que fazem parte do Exemplo 3 na sua ordem de recuperação adequada. Finalmente, na Figura 19 é apresentado o resultado da aplicação do algoritmo de Kim no Exemplo 3, e na Figura 20 o resultado da aplicação do algoritmo ***PrefetchStartTimes***.

Bloco de Sincronização	**Item**	**Elementos de Item**	**Grupo**	**Ordem de recuperação**
SB[0]	Item0:	Video1Seg1	G1	Video1Seg1
		Audio1Seg1		Audio1Seg1
SB[1]	Item0:	Video1Seg2	G3	Image1
		Audio1Seg2		Audio1Seg2
	Item2:	Image1	G3	Video1Seg2
	Item3:	Image2	G5	Text1
	Item4:	Text1	G5	Image2
SB[2]	Item0:	Audio2Seg1	G1	Audio2Seg1
		Text2		Text2
SB[3]	Item0:	Image3	G1	Audio2Seg2
	Item1:	Audio2Seg2	G3	Image3
SB[4]	Item0:	Video2Seg1	G3	Video2Seg1
SB[5]	Item0:	Text3	G3	Text3
	Item1:	Audio3Seg1	G3	Audio3Seg1
SB[6]	Item0:	Image4	G1	Video2Seg2
	Item1:	Video2Seg2	G3	Image4
SB[7]	Item0:	Audio3Seg2	G3	Audio2Seg2
SB[8]	Item0:	Video3Seg1	G3	Video3Seg1
SB[9]	Item0:	Video3Seg2	G3	Video3seg2

Tabela 9 - Itens do Exemplo 3.

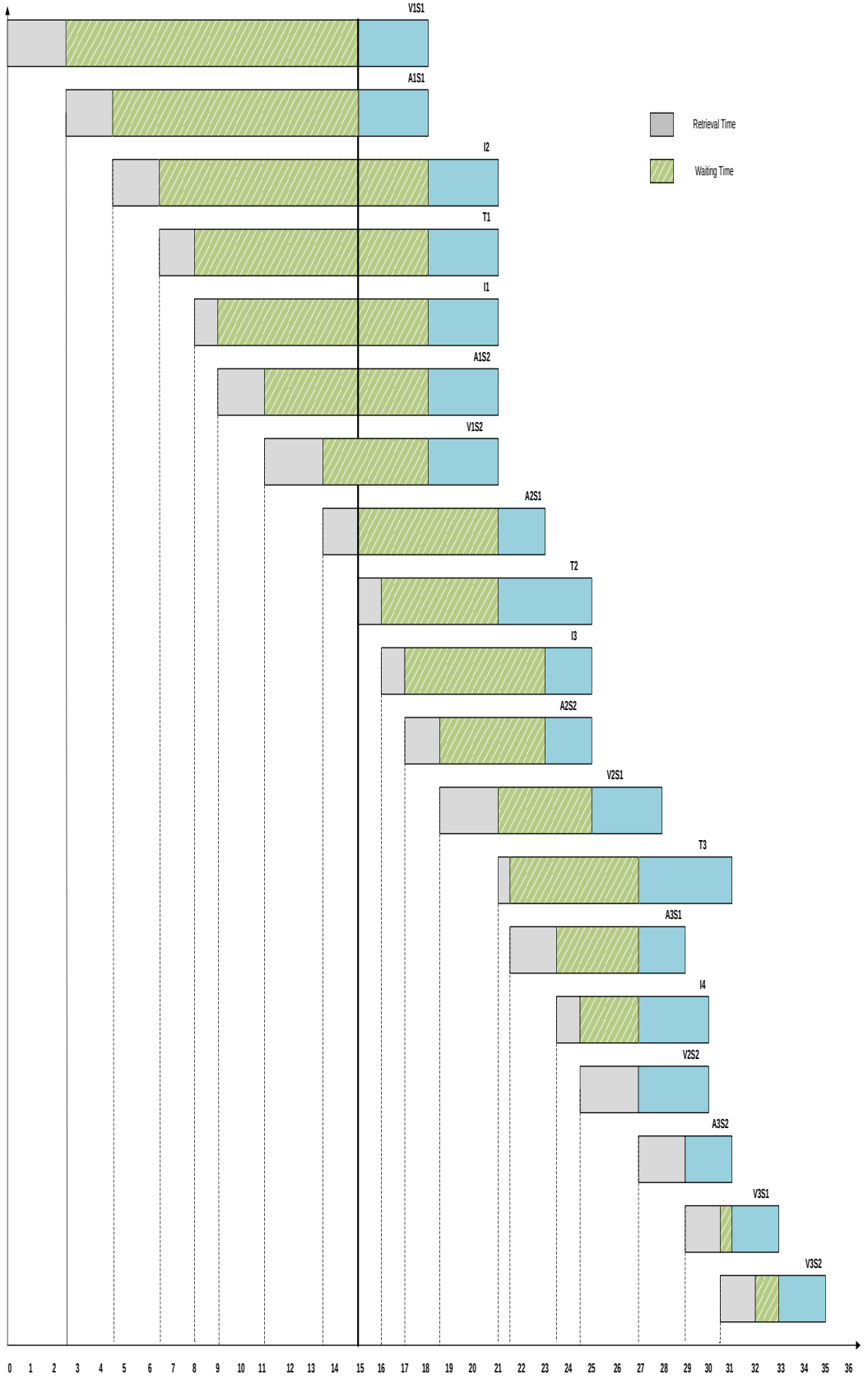


Figura 19 - Pré-busca do Exemplo 3 conforme ao algoritmo de Kim (Kim et al. 2001).

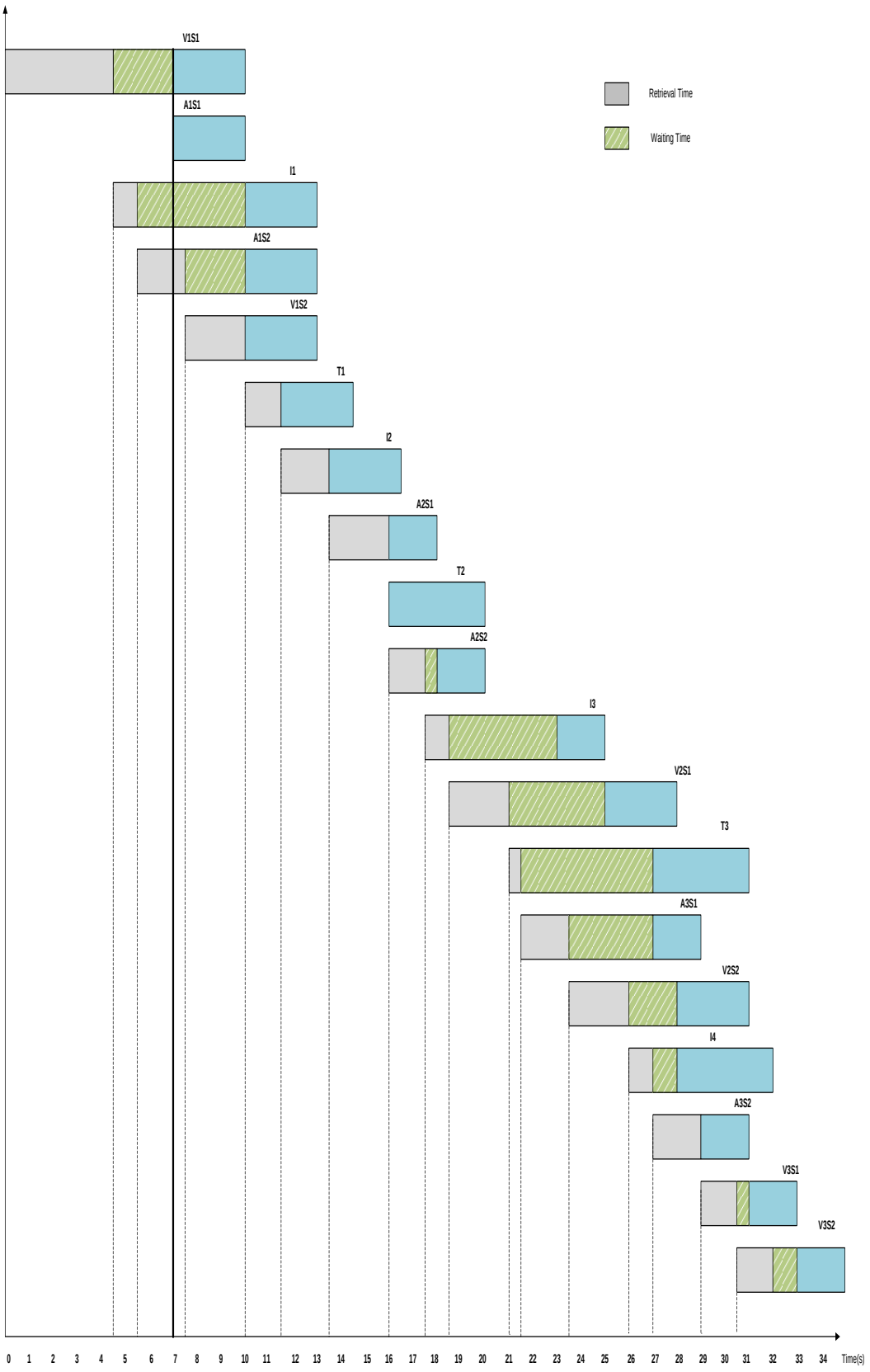


Figura 20 - Resultado de aplicar o algoritmo PrefetchStartTimes no Exemplo 3.

Para realizar a recuperação dos conteúdos pertencentes aos diferentes objetos de mídia que compõem a apresentação, o algoritmo ***RetrievalTimes*** trata, além das transições do tipo "*Start*", as transições de tipo "*Pause*", "*Stop*", "*Abort*" e "*Resume*", especificadas no plano de apresentação. A Tabela 10 apresenta o plano de apresentação que corresponde ao Exemplo 4 e a Tabela 11 apresenta o plano de eventos. Devido ao fato de que estamos trabalhando com eventos determinísticos, é possível conhecer os instantes de tempo exatos nos quais as transições deverão ser realizadas. Dessa forma, usando o plano de apresentação, a representação temporal do Exemplo 4 pode ser construída (Figura 21).

Transição	**Objeto de mídia**	**Tempo de início (s)**	**Duração (s)**	**Tipo de mídia**	**Tempo de recuperação (s)**
start	Video1Seg1	0,00	1,50	2	2,00
start	Audio1Seg1	0,00	1,50	2	1,50
pause	Video1Seg1	1,50	-	2	-
pause	Audio1Seg1	1,50	-	2	-
start	Video3Seg1	1,50	3,00	2	2,00
start	Video3Seg2	4,50	3,00	2	2,00
start	Text1	6,00	3,00	0	1,50
start	Video3Seg3	7,50	1,50	2	2,00
stop	Video3Seg3	9,00	-	2	-
resume	Video1Seg1	9,00	1,50	2	2,00
resume	Audio1Seg1	9,00	1,50	2	1,50
start	Video1Seg2	10,50	3,00	2	1,50
start	Audio1Seg2	10,50	3,00	2	1,50
start	Image2	10,50	3,00	0	2,00
start	Text2	10,50	6,00	0	0,50
start	Video1Seg3	13,50	3,00	2	2,00
start	Audio1Seg3	13,50	3,00	2	1,50
start	Image3	16,50	3,00	0	2,00
start	Text3	16,50	3,00	0	0,50
start	Image4	16,50	3,00	0	2,50

start	Video2Seg1	19,50	3,00	2	2,50
start	Audio2Seg1	19,50	3,00	2	2,50
start	Video2Seg2	22,50	3,00	2	2,50
start	Image5	22,50	3,00	0	2,00

Tabela 10 - Plano de Apresentação do Exemplo 4.

Mídia - Papel	Papel	Transição	Mídia - Transição
Video1Seg1	OnBegin	Start_Sync	Audio1Seg1
Video1Seg1	OnBegin	Start_Sync	Image1 (delay 1,5s)
Image1	OnBegin	Start_Sync	Video3Seg1
Video1Seg1	OnEnd	Start_PleSync	Image2
Video1Seg1	OnEnd	Start_PleSync	Text2
Video3Seg1	OnEnd	Start_Sync	Video3Seg2
Video3Seg1	OnBegin	Start_Sync	Text1(delay 4,5s)
Video3Seg2	OnEnd	Start_Sync	Video3Seg3
Text2	OnEnd	Resume_Sync	Video1Seg1
Text2	OnEnd	Resume_Sync	Audio1Seg1
Video1Seg1	OnEnd	Start_Sync	Video1Seg2
Audio1Seg1	OnEnd	Start_Sync	Audio1Seg2
Video1Seg2	OnEnd	Start_Sync	Video1Seg3
Audio1Seg2	OnEnd	Start_Sync	Audio1Seg3
Video1Seg3	OnEnd	Start_PleSync	Image3
Video1Seg3	OnEnd	Start_PleSync	Image4
Image3	OnBegin	Start_Sync	Text3
Image3	OnEnd	Start_Sync	Video2Seg1
Video2Seg1	OnEnd	Start_Sync	Video2Seg2

Video2Seg1	OnBegin	Start_Sync	Audio2Seg1
Video2Seg1	OnEnd	Start_PleSync	Image5

Tabela 11 - Plano de Eventos do Exemplo 4.

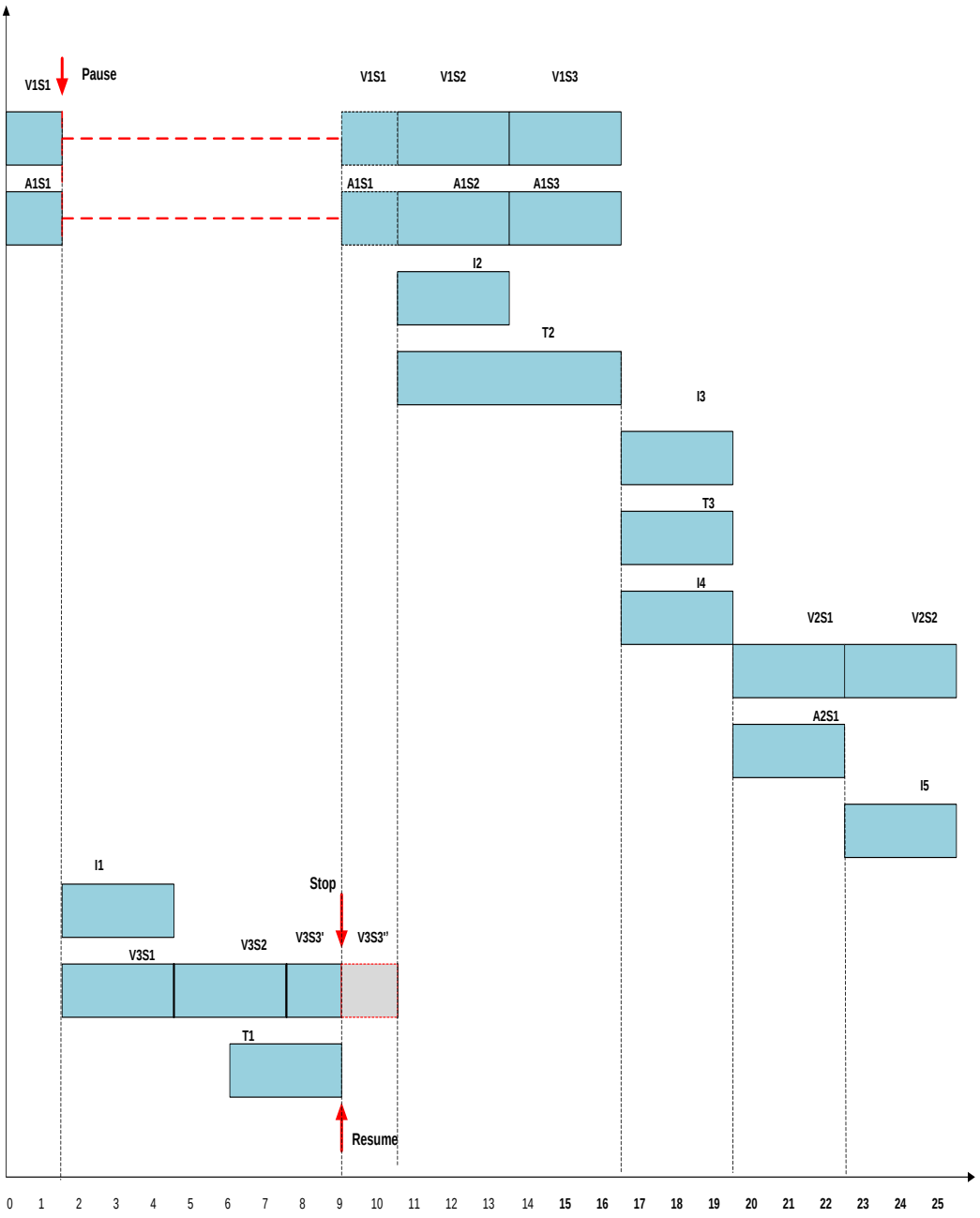


Figura 21 - Representação temporal do Exemplo 4.

Ao tratar as transições "*Pause*" e "*Resume*", é muito importante atentar para o fato de que os objetos de mídia contínua encontram-se divididos em segmentos (divisão que é realizada pelo protocolo usado para a entrega do conteúdo). Pode acontecer o caso em que o instante de início da transição aconteça durante o intervalo de tempo compreendido entre o início e o fim de um segmento pertencente a um objeto de mídia contínua, como acontece com o Video1Seg1 e Audio1Seg1 no Exemplo 4 (em 1,50 segundos a apresentação do Video1 e Audio1 é pausada, sendo retomada somente aos 9 segundos). Nesse caso, no momento da transição "***Resume***" ser executada sobre os objetos de mídia ligados a essa transição, deve-se levar em consideração que esses segmentos (Video1Seg1 e Audio1Seg1) já foram recuperados por completo e encontram-se no *buffer* da plataforma de exibição, prontos para serem apresentados. Isto acontece porque, como foi explicado na seção 4.2, cada segmento pertencente a um objeto de mídia contínua é sempre recuperado por completo. Dado isso, no momento da transição "***Resume***" ser executada, esses segmentos devem apenas ser apresentados a partir do instante de tempo no qual a pausa foi realizada.

Por outro lado, aos 9 segundos de iniciada a apresentação, o Video3 é parado. Dessa forma, todos os segmentos ligados a esse objeto a partir de nove segundos são descartados e não são considerados no plano de apresentação. Da mesma forma, são tratados os eventos de transição do tipo "*Abort*".

Uma vez que o Exemplo 4 é representado no eixo temporal, os blocos de sincronização podem ser encontrados. A Figura 22 mostra o Exemplo 4 dividido em 10 blocos de sincronização.

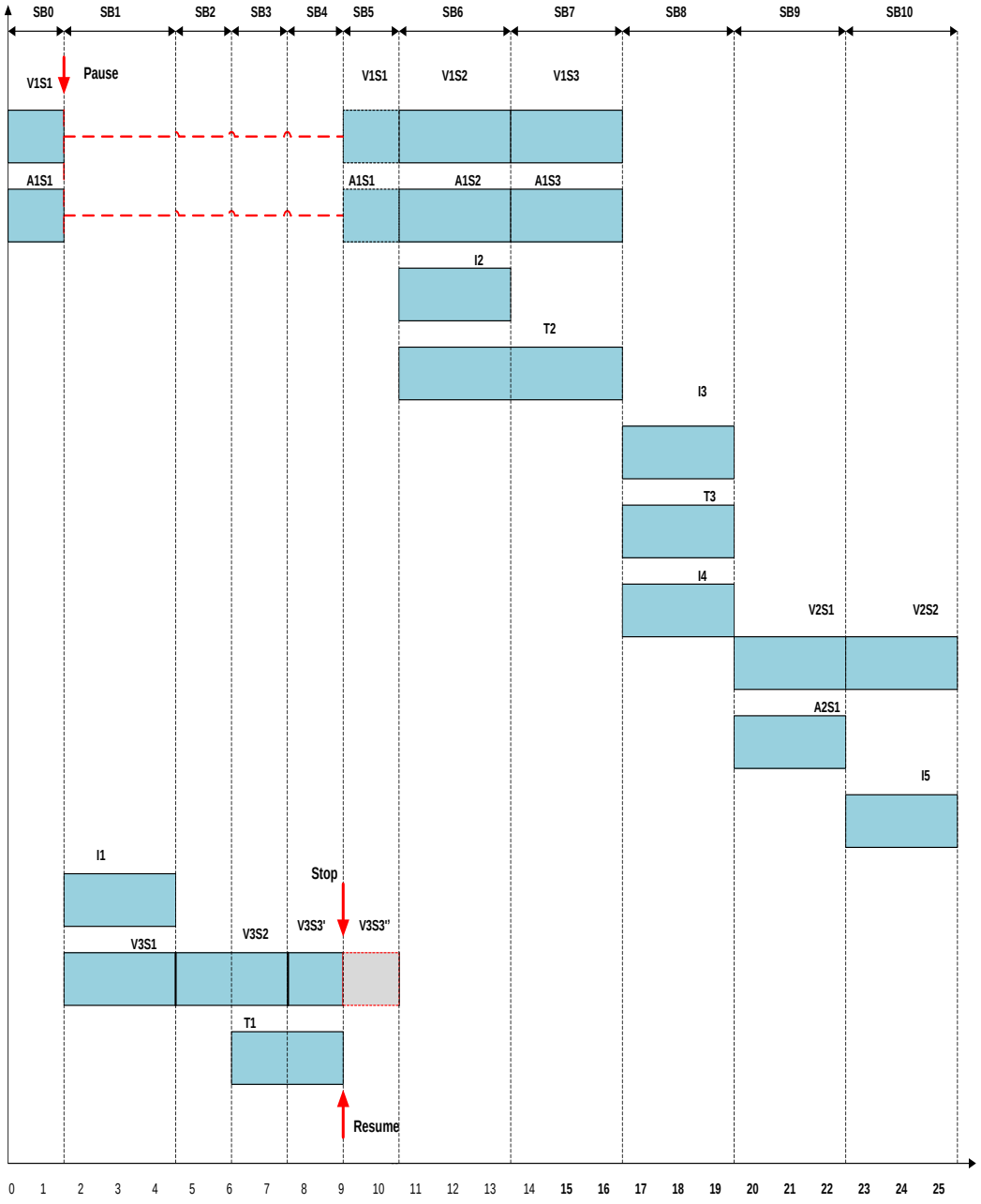


Figura 22 - Blocos de Sincronização do Exemplo 4.

Estabelecido os blocos de sincronização, os itens que compõem a apresentação multimídia são encontrados aplicando o algoritmo ***FindItems.*** Um passo prévio à execução do algoritmo ***FindItems*** é transformar os papeis "*OnResume*" em "*OnBegin*" e as transições "*Resume*" em "*Start*". Conforme explicado na Tabela 2, ao contrário das transições "*Start*" e "*Resume*", as transições do tipo "*Stop*", "*Abort*" e "*Pause*" não influenciam na recuperação do conteúdo. Depois disso, a ordem de recuperação dos itens é estabelecida, usando

o algoritmo ***SortItems.***

Para o Exemplo 4, os itens e a sua ordem de recuperação é apresentada na Tabela 12. Finalmente, a Figura 23 apresenta o resultado da aplicação do algoritmo ***PrefetchStartTimes*** no exemplo.

Bloco de Sincronização	**Item**	**Elementos de Item**	**Grupo**	**Ordem de recuperação**
SB[0]	Item0:	Video1Seg1	G1	Video1Seg1
		Audio1Seg1		Audio1Seg1
SB[1]	Item0:	Image1	G3	Image1
		Video3Seg1		Video3Seg1
SB[2]	Item0:	Video3Seg2	G3	Video3Seg2
SB[3]	Item0:	Text1	G3	Text1
SB[4]	Item0:	Video3Seg3	G3	Video3Seg3
SB[5]	-	-	-	-
SB[6]	Item0:	Video1Seg2	G3	Audio1Seg2
		Audio1Seg2		Video1Seg2
	Item2:	Image2	G4	Text1
	Item3:	Text2	G4	Image2
SB[7]	Item0:	Video1Seg3	G3	Audio1Seg3
		Audio1Seg3		Video1Seg3
SB[8]	Item0:	Image4	G4	Image3
	Item1:	Image3	G1	Text3
		Text3		Image4
SB[9]	Item0:	Video2Seg1	G3	Video2Seg1
		Audio2Seg1		Audio2Seg1
SB[10]	Item0:	Video2Seg2	G3	Video2Seg2
	Item1:	Image5	G4	Image5

Tabela 12 - Itens do Exemplo 4.

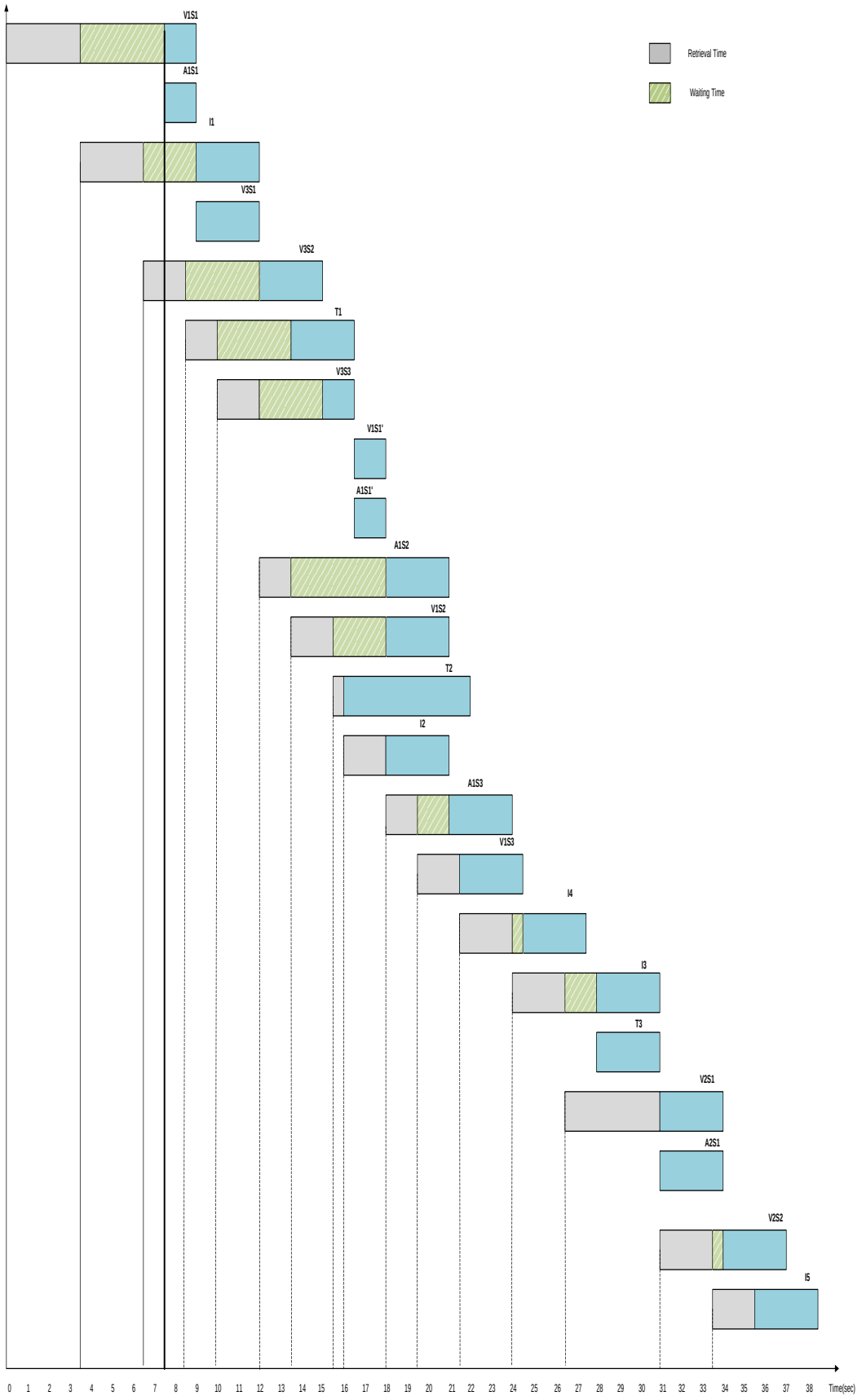


Figura 23 - Resultado de aplicar o algoritmo *PrefetchStartTimes* no Exemplo 4.

Outra forma de tratar as transições "*Pause*" e "*Resume*" é considerar que um objeto de mídia pode ser subdividido em objetos independentes, e cada "sub-objeto" pode ser segmentado de forma independente. Com os objetos de mídia não segmentados e com base no plano de apresentação, é possível considerar que uma mídia M1 que receba um "*Start*", "*Pause*", "*Resume*" e "*Stop*" (nessa ordem, e em momentos diferentes) é, na verdade, dois objetos de mídia M1' e M1'' que recebe um "*Start*" e "*Stop*" em M1' e, posteriormente, um "*Start*" e um "*Stop*" em M1''. Dessa forma, a transição "*Pause*" é transformada em "*Stop*" e a transição "*Resume*" em "*Start*". Ao tratar "*Pause*" e "*Resume*" como "*Start*" e "*Stop*" em objetos de mídia independentes é possível evitar o problema anterior de recuperar todo o segmento do objeto de mídia contínua sendo que só uma parte daquele segmento é necessária para ser apresentada naquele momento.

5.3
Medições do algoritmo de pré-busca

Em relação à implementação real, o algoritmo *PrefetchStartTimes* foi implementado em C++. Esse algoritmo tem como dados de entrada: o plano de apresentação e o plano de eventos que correspondem a uma aplicação multimídia desenvolvida na linguagem declarativa NCL. A saída do algoritmo são os tempos de início de recuperação dos objetos de mídia que compõem a apresentação, além da latência de exibição inserida.

Foram realizadas algumas medições com o objetivo de comparar o mecanismo de pré-busca desenvolvido neste trabalho e os mecanismos apresentados nos trabalhos relacionados, em particular o algoritmo proposto em (Kim et al. 2001). A Tabela 12 mostra os resultados obtidos com 7 aplicações multimídia. Essas aplicações contém eventos determinísticos e variam pela porcentagem de relacionamentos do tipo "*Sync*" e "*PleSync*" que existem entre os objetos de mídia.

Aplicação	No Objetos	% R. Sync	% R. PleSync	Latência de exibição (s)	
				Kim	PrefetchStartTimes
1	9	36,36%	63,63%	5,5	3,00
2	13	72,22%	27,77%	15,00	7,00
3	12	64,7%	35%	18,00	4,5
4	13	66,66%	33,33%	25,5	14,5
5	13	100%	0%	25,5	25,5
6	6	40%	60%	5,00	1,00
7	10	100%	0%	6,32	6,32

Tabela 13 - Medições do algoritmo PrefetchStartTimes.

Os resultados apresentados na Tabela 13 mostram que uma redução considerável na latência de exibição foi alcançada aplicando o algoritmo desenvolvido neste trabalho. A latência de exibição foi reduzida quase à metade nas aplicações 1, 2 e 4. Nas aplicações 3 e 6 a latência foi reduzida em 75% e 80% respectivamente. Essa redução na latência de exibição depende da flexibilidade que possui a apresentação multimídia com respeito ao instante de tempo no qual os seus conteúdos devem ser apresentados.

Conforme definido na seção anterior, a redução da latência de exibição é conseguida devido à presença de objetos de mídia que possuem um sincronismo rígido e também devido à prioridade de recuperação estabelecida nos objetos de mídia que possuem um relacionamento com o passado ou futuro da aplicação. Cabe ressaltar que melhores resultados são obtidos quando existem relacionamentos "*PleSync*" entre os objetos de mídia pertencentes a blocos de sincronização diferentes.

5.4
Cálculo da capacidade do *buffer* requerida

Conforme ao definido na Seção 3.1.3, neste trabalho a capacidade do *buffer* é considerada como infinita. Pode-se, no entanto, calcular a capacidade máxima necessária.

Para calcular a capacidade do *buffer* requerida para cada instante de tempo de início de pré-busca, é necessário considerar tanto a taxa de consumo Ω dos

conteúdos que encontram-se no *buffer* como a banda passante Bw em um instante de tempo determinado. A Figura 24 mostra o comportamento de um objeto de mídia na plataforma de apresentação. O objeto possui um tempo de início de apresentação t_{cs}, um intervalo de execução ou exibição igual a T_c e um tempo de fim de apresentação igual a t_{ce}. O ciclo de pré-busca dos dados deve terminar antes do tempo t_{ce}, isto é, $t_{pe} \leq t_{ce}$, onde t_{pe} é o tempo de fim de pré-busca dos conteúdos. Em consequência, o tempo de início de pré-busca de um determinado objeto de mídia não pode ser maior do que:

$$\bar{t}_{ps} = t_{ce} - \Omega.\left(\frac{T_c}{Bw}\right) \qquad (4)$$

Onde, $\bar{t}_{ps}$ é o tempo máximo para o início da pré-busca. Além disso, T_{pf} é o intervalo de tempo que existe entre o tempo de início da pré-busca t_{ps} e o tempo de início da apresentação do objeto de mídia t_{cs}, isto é $T_{pf} = t_{cs} - t_{ps}$. Então, o mínimo intervalo de tempo para a pré-busca é definido como segue:

$$\bar{T}_{pf} = t_{cs} - \bar{t}_{ps} \qquad (5)$$

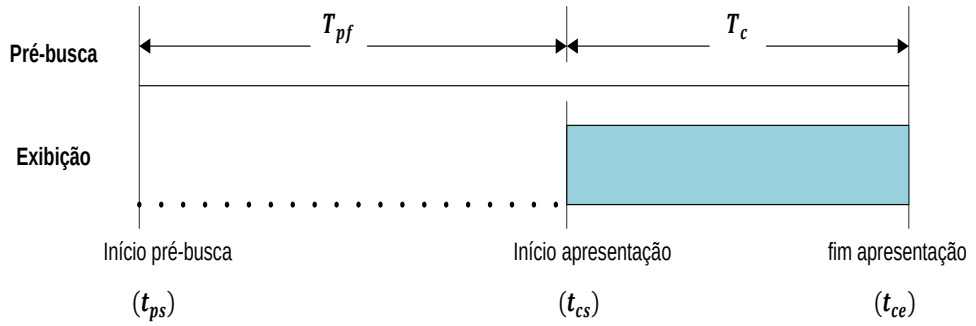


Figura 24 - Comportamento da pré-busca e apresentação de um objeto de mídia.

Para a recuperação dos conteúdos dos objetos de mídia a partir da rede de comunicação, os dados são procurados no servidor no instante de tempo igual a t_{ps}. Seja $P_{(t_{ps},t)}$ a função de produção no tempo t, indicando a quantidade total de dados transmitidos para a plataforma de apresentação. Essa função é expressa pela

equação 6:

$$P(t_{ps}, t) = \begin{cases} Bw \cdot (t - t_{ps}) & if\ t_{ps} < t \leq t_{pe} \\ 0 & otherwise. \end{cases} \quad (6)$$

Por outro lado, o número total de dados consumidos a partir do *buffer* para reprodução no tempo t é definido por uma função de consumo $C(t_{cs}, t)$, representada pela equação 7:

$$C(t_{cs}, t) = \begin{cases} \Omega \cdot (t - t_{cs}) & if\ t_{cs} < t \leq t_{ce}, \\ 0 & otherwise \end{cases} \quad (7)$$

Considerando que os dados armazenados no *buffer B* estão definidos pela função de produção $P(t_{ps}, t)$ e consumidos pela função de consumo $C(t_{cs}, t)$, a quantidade total do *buffer* que é ocupado pelos conteúdos de mídia no tempo *t* é definida por uma função de ocupação do *buffer* $B(t_{ps}, t_{cs}, t)$ dada pela equação 8:

$$B(t_{ps}, t_{cs}, t) = P(t_{ps}, t) - C(t_{cs}, t) \quad (8)$$

Com a finalidade de evitar a carência de dados durante a apresentação, a função de ocupação do *buffer* deve ser sempre positiva durante a sessão de exibição. Isto é, $t_{cs} \leq t \leq t_{ce}$.

Durante o processo de pré-busca os dados são armazenados dentro do *buffer* de forma incremental, até o seu tempo de início de apresentação. Uma vez que a apresentação dos conteúdos é iniciada, os dados são consumidos de tal forma que a seu término o *buffer* seja liberado.

O *buffer* alcança a sua capacidade máxima B_{max} quando *t* é igual a t_{cs}. Porém, isso nem sempre acontece, porque poderia se dar o caso em que a recuperação dos conteúdos inicie antes de $\bar{t}_{ps}$, no instante de tempo igual a $\bar{\bar{t}}_{ps}$, onde $\bar{\bar{t}}_{ps} = t_{cs} - \bar{\bar{T}}_{pf}$ ou equivalentemente $\bar{\bar{T}}_{pf} = \Omega . \frac{Tc}{Bw}$. Assim, é evidente que nenhum tempo de pré-busca menor do que $\bar{\bar{t}}_{ps}$ vai precisar requisitos de *buffer* maiores do que $B_{max}(\bar{\bar{T}}_{pf}) = \Omega . T_c$. Por conseguinte, $B_{max}(\bar{\bar{T}}_{pf})$ é o pior caso.

A Figura 25 mostra a variação do requisito máximo de *buffer* para cada

instante de tempo de início de pré-busca, considerando sempre que no instante de tempo t_{cs} o requerimento do *buffer* é máximo. Os requisitos do *buffer* associados com o tempo de pré-busca T_{pf} são definidos como:

$$B_{req}(T_{pf}) = B_{max}(T_{pf}) = Min(Bw \cdot T_{pf}, \Omega \cdot T_c) \qquad (9)$$
$$for\ T_{pf} \geq (t_{cs} - \bar{t}_{ps})$$

Assim, os requisitos do *buffer* são dados pela equação 10:

$$B_{req}(Bw) = (\Omega - Bw) \cdot T_c \qquad (10)$$

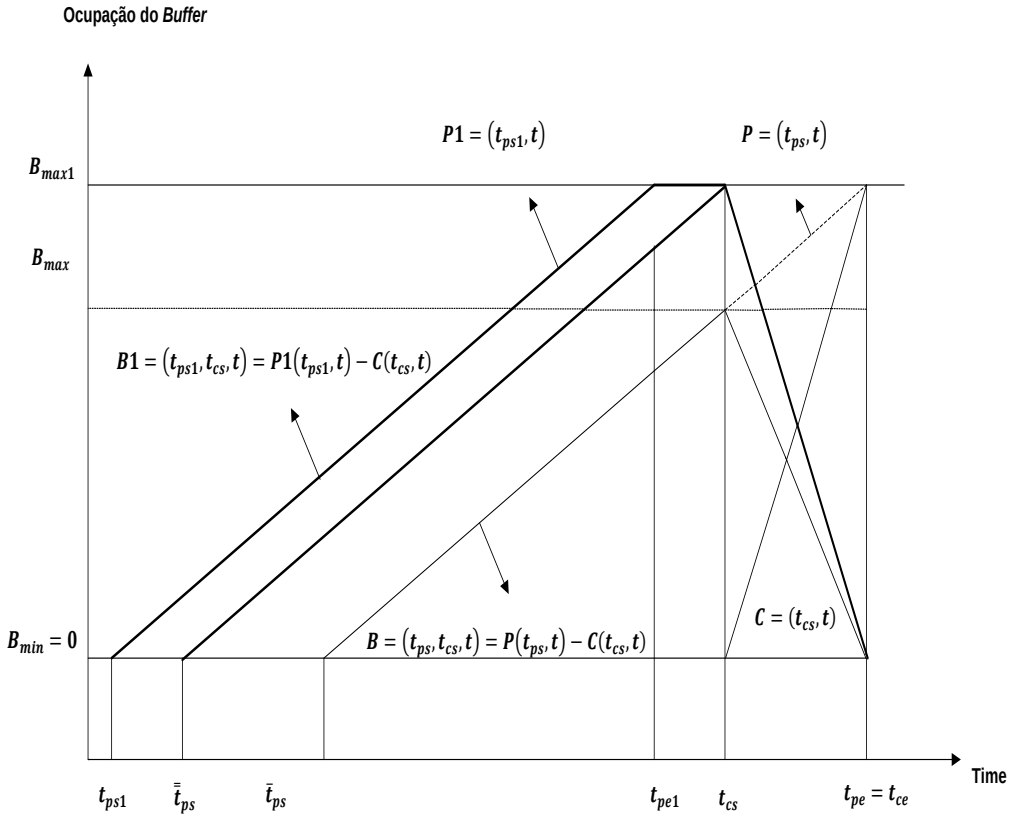


Figura 25 - Relação entre a Função de Produção $P(t_{ps}, t)$, a Função de Consumo $C(t_{cs}, t)$ e a Função de ocupação do Buffer $B(t_{ps}, t_{cs}, t)$ de um objeto de mídia em particular.

Cabe mencionar que o controle de *buffer* explicado acima não têm

contribuição além do Kim (Kim et al. 2001). Porém esse mecanismo pode ser aplicado no algoritmo de pré-busca desenvolvido neste trabalho.

5.5
Mecanismos para a compensação de *gaps*

Mesmo que os mecanismos de pré-busca tenham sido aplicados e os instantes de tempo de início de recuperação dos objetos de mídia que fazem parte da apresentação multimídia tenham sido calculados, nem sempre a ausência de interrupções ou *gaps* pode ser garantida. Como foi explicado na Seção 5.1, a ausência de *gaps* possui uma relação estreita entre o instante de tempo no qual a aplicação multimídia é recebida na plataforma de apresentação (trec) e o Tret necessário para que apresentação esteja livre de *gaps.*

Com o intuito de reduzir os *gaps* inseridos na apresentação multimídia, mecanismos de ajuste elástico podem ser empregados, tanto em tempo de compilação, quando o plano de apresentação é construído; quanto em tempo de execução, quando é necessário realizar ajustes durante a apresentação. Basicamente, os mecanismos de ajuste elástico permitem aumentar ou diminuir o tempo de apresentação de alguns objetos de mídia a fim de manter a consistência espaço-temporal da apresentação, melhorando assim a sua qualidade. Além disso, ajustes de tempo elástico podem ser necessários se um sincronismo do tipo rígido é exigido.

Michele Y. Kim e Junehwa Song (Kim & Song 1995) descrevem o "*elastic time model*" para documentos multimídia, onde cada objeto de mídia possui uma tolerância referente a suas durações de apresentação, tal como acontece no sistema Firefly (Buchanan & Zellweger 1993). Nesse sistema foi aplicada uma técnica de otimização baseada em programação linear que determina um agendamento ótimo que minimiza o número total de aumentos ou diminuições dos objetos, sem alterar as contruções temporais inter-objetos dadas.

Baseado no "*elastic time model*" mencionado acima, (Bachelet et al. 2007) propõem um algoritmo ainda mais eficiente para resolver o problema de ajuste elástico baseado no problema de "*minimum-cost flow problem*" (Fulkerson 1961) aplicado em grafos temporais.

6
Pré-Busca de apresentações multimídia interativas

Como mencionado no início do Capítulo 5, os eventos de seleção estão associados à interatividade da apresentação multimídia. Esses eventos podem estar ligados a uma ou mais cadeias temporais secundarias e, dependendo da escolha realizada pelo usuário, uma dessas cadeias é apresentada. Devido à natureza não determinística das apresentações multimídia interativas é impossível saber previamente se o usuário irá ou não interagir com a aplicação.

O algoritmo ***PrefetchStartTimes*** pode também ser empregado em apresentações multimídia que possuem eventos não determinísticos com tipo "***OnSelection***". Conforme mencionado acima, o resultado de um evento de seleção unicamente é conhecido em tempo de execução. Então, se o usuário interagir, o plano de pré-busca permanece igual, caso contrário esse plano é recalculado. O plano de pré-busca é calculado considerando o pior caso, ou seja, considera-se que existe um evento de seleção e seu resultado é realizado no final da apresentação da mídia. Desse modo, todos os papeis ***"OnSelection"*** existentes na aplicação multimídia são transformados em ***"OnEnd"***, assim os objetos ligados a esse papel são tratados como é explicado na Tabela 2. Cabe ressaltar, que ao realizar essa transformação, os objetos de mídia ligados à transição "***Start***" somente serão apresentados uma vez que o objeto de mídia ligado a esse papel termine. Isso é, não existe interatividade instantânea.

No caso de existir vários caminhos ou escolhas, como acontece nas Narrativas Interativas, onde o usuário deve escolher entre várias opções para continuar a história, não é factível determinar previamente qual desses caminhos será selecionado. Essa informação é conhecida unicamente durante o processo de execução. Quando existem vários caminhos onde o usuário deve escolher um, o plano de pré-busca é calculado considerando todos esses caminhos e quando o resultado da interação é conhecido o plano deve ser recalculado imediatamente.

Por outro lado, a interatividade pode possuir uma resposta instantânea ou possuir um tempo de interatividade. No primeiro caso, os objetos de mídia têm

que estar disponíveis desde o início da interatividade e o plano de pré-busca tem que ser recalculado a partir do momento que se conhece o instante do evento imprevisível, isso porque o plano foi construído baseado no pior caso. No segundo caso é estabelecido um intervalo de tempo de interatividade, então os objetos de mídia têm que estar disponíveis no fim desse tempo. Para determinar o tempo de interatividade ideal são considerados os tempos de resposta propostos em (Fiedler 2004), onde o tempo máximo em aplicações interativas é de 10 segundos.

Nesse último caso, possuímos uma flexibilidade da ocorrência de interatividade porque a interatividade somente será disparada ao terminar o tempo de interatividade definido. Dessa forma é possível utilizar esse tempo de interatividade para recuperar a maior quantidade de conteúdo possível para assim evitar interrupções durante a sua apresentação.

Assim, nas Narrativas Interativas, devido à existência de caminhos alternativos, o plano de pré-busca deve ser calculado considerando que todos os caminhos possuem a mesma probabilidade de ocorrência. Quando a escolha for realizada o plano deve ser reajustado, isto é, recalculado a partir do caminho selecionado. É importante destacar que como todos os caminhos foram calculados previamente, no instante de tempo que o caminho escolhido pelo usuário é conhecido, os conteúdos correspondentes a esse caminho (que já foram recuperados) devem ser aproveitados e o plano deve ser recalculado considerando esses conteúdos.

A Figura 26 representa uma narrativa interativa onde o vídeo 1 apresenta 3 opções de caminhos diferentes para continuar a história. Cada um desses caminhos corresponde a uma cadeia temporal secundária dentro da aplicação multimídia interativa.

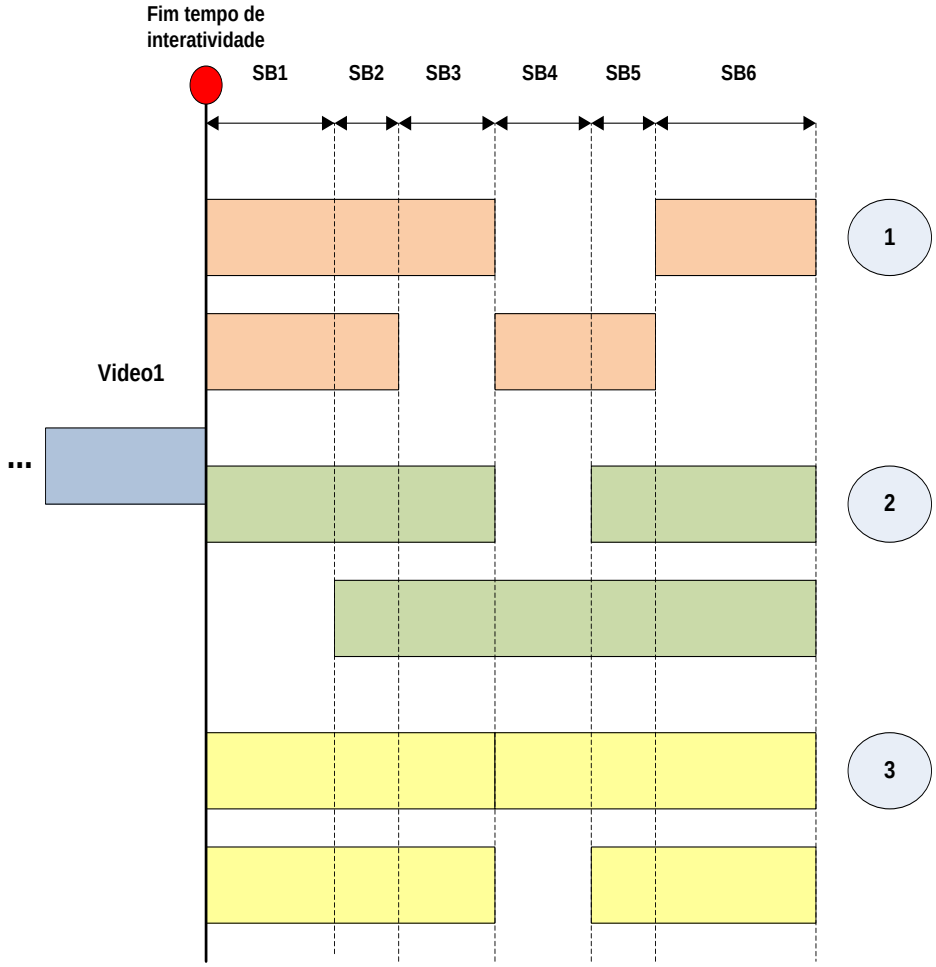


Figura 26 - Exemplo de Narrativa Interativa com 3 caminhos alternativos.

Devido ao fato que é impossível conhecer previamente qual alternativa será escolhida pelo usuário, todos os conteúdos correspondentes a esses caminhos devem ser recuperados por igual. Para isso, todas as cadeias temporais correspondentes aos três caminhos alternativos são divididas em blocos de sincronização. Os blocos são recuperados em ordem sequencial até que a escolha do usuário seja conhecida.

Uma vez que a escolha do usuário é conhecida, os conteúdos de mídia que já foram recuperados e não correspondem à alternativa selecionada pelo usuário são descartados. Por outro lado, os conteúdos que correspondem à alternativa selecionada e que já foram recuperados são mantidos no *buffer* de armazenamento para serem aproveitados. Considerando o último bloco de sincronização que foi recuperado, é realizada uma nova divisão de blocos, que unicamente considera a cadeia temporal do caminho escolhido. Por exemplo, com base na Figura 26, suponhamos que quatro segundos antes do termino do tempo de interatividade, o

usuário realiza a escolha da alternativa número três. Durante os primeiros seis segundos foi possível recuperar todos os conteúdos de mídia, de todos os caminhos alternativos, até o Bloco de Sincronização 3. Isso significaria que, a partir do segundo 4 até o fim do tempo de interatividade, somente seriam recuperados os objetos de mídia que fazem parte da escolha número três.

Um aspecto importante que deve ser considerado é que o tempo de interatividade, definido neste trabalho como 10 segundos, pode ser empregado para recuperar a maior quantidade de conteúdo possível até o seu término. Porém, esse intervalo de tempo poderia ser utilizado de uma melhor forma se a escolha do usuário for conhecida quando o tempo de interatividade é iniciado, por exemplo, nos três primeiros segundos. Isto significaria que se disporia de 7 segundos para recuperar exclusivamente os objetos de mídia que fazem parte do caminho alternativo selecionado. O pior caso acontece quando a escolha do usuário é conhecida um pouco antes do fim do tempo de interatividade. Nesse caso o tempo de interatividade foi compartilhado para a recuperação de todos os caminhos existentes.

7
Conclusões

Esta dissertação apresentou aspectos relacionados aos mecanismos de pré-busca com o fim de reduzir a latência de exibição em apresentações multimídia. Entre vários aspectos, foi destacada a importância de considerar os relacionamentos causais, existentes entre os objetos de mídia que compõem uma apresentação multimídia, na definição da ordem de recuperação dos objetos e no cálculo dos instantes de tempo para o início de recuperação.

Embora o presente trabalho tenha apresentado um plano de pré-busca que primordialmente considera aplicações multimídia que possuem eventos determinísticos, aspectos importantes para a pré-busca de conteúdo em aplicações que têm eventos não-determinísticos foram levantados e analisados. Outro aspecto importante discutido na dissertação foi que, embora o *buffer* teórico utilizado para armazenar os conteúdos dos objetos de mídia que são adquiridos tenha sido inicialmente considerado infinito, o cálculo do tamanho máximo do *buffer* requerido pela apresentação multimídia foi realizado.

As seções a seguir detalham as principais contribuições desta dissertação (Seção 7.1) e os principais trabalhos futuros que podem dela decorrer (Seção 7.2).

7.1
Contribuições da dissertação

Ao longo desta dissertação foi desenvolvido um algoritmo de pré-busca com o intuito de reduzir a latência de exibição em apresentações multimídia, denominado ***PrefetchStartTimes***. Fundamentalmente, são três as etapas que compõem o processo de pré-busca: encontrar os blocos de sincronização (SBs) da apresentação multimídia, encontrar os itens pertencentes a cada SB, ordená-los de forma adequada e, finalmente, calcular os tempos de início de recuperação de cada item.

No algoritmo de pré-busca apresentado neste trabalho, diferente de outros mecanismos de pré-busca que somente consideram a distribuição dos objetos de

mídia no eixo temporal, os relacionamentos causais entre os objetos são considerados, assim como o tipo de sincronismo (definido pelo autor da aplicação): "*Synchronous*" ou "*Plesiochronous*". Devido ao fato que sincronismo do tipo "*Plesiochronous*" proporciona maior flexibilidade, o propósito é satisfazer o maior número de relacionamentos que possuam um sincronismo do tipo "*Synchronous*".

Outra contribuição vem do fato que os mecanismos de pré-busca apresentados em trabalhos relacionados apenas tratam transições do tipo "*Start*". O mecanismo desenvolvido neste trabalho trata, além das transições "*Start*", transições do tipo "*Stop*", "*Pause*", "*Resume*" e "*Abort*", que são comumente utilizadas nas apresentações multimídia.

Com o intuito de diminuir ou evitar a inserção desnecessária de *gaps* que afetam a qualidade da apresentação, a ordem de recuperação adequada dos objetos de mídia é estabelecida considerando as relações existentes entre eles. Tanto a sua relação com o passado quanto com o futuro é considerada.

Esta dissertação também discutiu como pode ser calculada a capacidade máxima do *buffer* de armazenamento requerida por uma apresentação multimídia determinada.

7.2 Trabalhos futuros

Os trabalhos realizados nesta dissertação abrem várias possibilidades para o desenvolvimento de trabalhos futuros. Em relação ao algoritmo de pré-busca desenvolvido, o algoritmo *PrefetchStartTimes* calcula os tempos de início adequados para a recuperação dos conteúdos dos diferentes objetos de mídia que fazem parte da apresentação multimídia. No entanto, em tempo de exibição, podem existir fatores, como o desempenho da rede de comunicação, o tempo de acesso aos servidores onde encontram-se armazenados os objetos de mídia etc., que podem alterar o plano de pré-busca e, nesse caso, é necessário que o plano seja recalculado em tempo de execução.

O algoritmo de pré-busca proposto (***PrefetchStartTimes***) trata unicamente eventos determinísticos. Eventos de seleção foram transformados em eventos de apresentação. No entanto, quando um evento de seleção acontece, o plano de pré-

busca deve ser recalculado em tempo de exibição.

Os trabalhos desta dissertação não incluem algoritmos de pré-busca para apresentações multimídia que possuam conteúdo interativo. Porém, o algoritmo de pré-busca apresentado neste trabalho serve como base para o desenvolvimento de algoritmos de pré-busca que considerem eventos não-determinísticos.

Para a recuperação dos objetos de mídia que possuem um sincronismo fraco é necessário definir um tempo limite de recuperação porque caso contrário a recuperação desses objetos poderia ser adiada até um instante de tempo que não faça mais sentido a sua apresentação.

Em relação ao *buffer* de armazenamento, este trabalho considerou a capacidade do *buffer* infinita. Contudo, considerando o fato de que os recursos da plataforma de exibição são limitados, o ideal é trabalhar com uma capacidade do *buffer* definida e limitada, onde o seu gerenciamento torna-se necessário. Nesse caso é importante levar em consideração alguns aspectos como: o custo de manter no *buffer* os conteúdos que poderiam ser utilizados no futuro ou recuperá-los novamente quando esses são requeridos; definir os instantes de tempo nos quais os conteúdos devem ser liberados do *buffer*; e, principalmente, otimizar a capacidade de armazenamento.

Por fim, é importante mencionar que o desenvolvimento e a utilização de algoritmos de pré-busca em apresentações multimídia é fundamental. Estudos e aperfeiçoamentos nessa área favorecem a entrega de apresentações multimídia sincronizadas e sem a presença de interrupções durante a sua apresentação, isto é, com uma melhor qualidade de experiência.

8
Referências Bibliográficas

ABNT - Associação Brasileira de Normas e Técnicas**. Televisão digital terrestre – Codificação de dados e especificações de transmissão para radiodifusão digital – Parte 2: Ginga-NCL para receptores fixos e móveis – Linguagem de aplicação XML para codificação de aplicações**. ABNT, NBR 15606-2:2007. Versão corrigida. 2008.

Allen, J.F. **Maintaining knowledge about temporal intervals.** *Commun. ACM*, 26, pp.832–843, 1983.

Bachelet, B. et al. **Elastic time computation in QoS-driven hypermedia presentations.** *Multimedia Systems*, 12, pp.461–478, 2007.

Begen, A.C., Akgul, T. & Baugher, M. **Watching Video over the Web: Part 1: Streaming Protocols**. *Internet Computing, IEEE*, 15, pp.54–63, 2011.

Blakowski, G. & Steinmetz, R. **A media synchronization survey: reference model, specification, and case studies.** *Selected Areas in Communications, IEEE Journal on*, 14, pp.5–35, 1996.

Boronat, F., Lloret, J. & Garcıa, M. **Multimedia group and inter-stream synchronization techniques: A comparative study**. *Inf. Syst.*, 34, pp.108–131, 2009.

Buchanan, M.C. & Zellweger, P. **Automatically generating consistent schedules for multimedia documents.** *Multimedia Systems*, 1, pp.55–67,1993.
Bulterman, D. & Rutledge, L. ***SMIL 2.0: Interactive Multimedia for Web and Mobile Devices***, Springer, 2004.

Candan, K.S., Prabhakaran, B. & Subrahmanian, V.S. **Retrieval schedules based on resource availability and flexible presentation specifications.** *Multimedia Systems*, 6, pp.232–250, 1998.

Costa, R.M.R. **Controle do Sincronismo Temporal de Aplicações Hipermídia.** Tese de doutorado do Programa de Pós-Graduação em Informatica da Puc-Rio, 2010.

Costa, R.M.R. & Soares, L.F.G. **Modelo Temporal Hipermídia para Suporte a Apresentações em Ambientes Interativos.** *XIII Simpósio Brasileiro de Sistemas Multimídia e Web – WebMedia 2007, 2007.*

Egger, S. et al. **Waiting times in quality of experience for web based services**. In *Quality of Multimedia Experience (QoMEX), 2012 Fourth International*

Workshop on. pp. 86–96, 2012.

Fiedler, M. EuroNGI Deliverable D. WP. **JRA. 6.1. 1: State-of-the art with regards to user-perceived Quality of Service and quality feedback**, 2004.

Fulkerson, D. **An out-of-kilter method for minimal cost flow problems.** *SIAM J. Appl.Math*, 9, pp.18–27, 1961.

Geyer, W., Bernhardt, C. & Biersack, E. **A Synchronization Scheme for Stored Multimedia Streams**. In Interactive Distributed Multimedia Systems and Services (European Workshop IDMS'96. Springer Verlag, pp. 277–295, 1996.

Hejazi*y, S.R. & Saghafianz, S. **Flowshop-scheduling problems with makespan criterion: a review**, 2004**.**

Hossfeld, T. et al. **Initial delay vs. interruptions: Between the devil and the deep blue sea.** In *Quality of Multimedia Experience (QoMEX), 2012 Fourth International Workshop on*. pp. 1–6, 2012.

In-Ho Lin, B.-H.L. & Wu, C.-C. **A Synchronization Model for Presentation of Multimedia Objects.** *Journal of the Chinese Institute of Engineers*, 24, pp.173–186, 2001.

Ishibashi, Y. & Tasaka, S. **A comparative survey of synchronization algorithms for continuous media in network environments**. In *Local Computer Networks, 2000. LCN 2000. Proceedings. 25th Annual IEEE Conference on*. pp. 337–348, 2000.

ISO/IEC 13818-1. International Organization for Standarization. **Information technology –Generic coding of moving pictures and associated audio information**: Systems, ISO/IEC, 2000.

ISO/IEC 13818-6. International Organization for Standarization. **Information technology- Generic coding of moving pictures and associated audio information – Part 6 Extensions for DSM-CC**. 1998.

ITU-T Recommendation H.761, 2009. **Nested Context Language (NCL) and Ginga-NCL for IPTV Services**. Geneva, April, 2009.

Johnson, S.M. **Optimal two-and three-stage production schedules with setup time included.** *Naval Research Logistics Quarterly*, 1, pp.61–68 , 1954.

Kim, M.Y. & Song, J. **Multimedia Documents with Elastic Time.** In *Proceedings of the Third ACM International Conference on Multimedia*. MULTIMEDIA '95. San Francisco, California, USA: ACM, pp. 143–154, 1995.

Kim, S.-W. et al., 2001. **The optimal retrieval start times of media objects for the multimedia presentation.** *Information and Software Technology*, 43, pp.219–229.

Kleinberg, J. & Tardos, E. ***Algorithm Design***, Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc, 2005.

Kononov, A. et al. **Quantity-based buffer-constrained two-machine flowshop problem: active and passive prefetch models for multimedia applications**. *J. of Scheduling*, 15, pp.487–497, 2012.

KURISU, T. **TWO-MACHINE SCHEDULING UNDER REQUIRED PRECEDENCE AMONG JOBS.** *Journal of the Operations Research Society of Japan*, 19, pp.1–13, 1976.

Lazic, K. et al. **One Implementation of adaptive streaming over HTTP on Android DTV platform.** In *Consumer Electronics - Berlin (ICCE-Berlin), 2012 IEEE International Conference on*. pp. 282–284, 2012.

Lin, F.-C., Hong, J.-S. & Lin, B.M.T. **A two-machine flowshop problem with processing time-dependent buffer constraints-An application in multimedia presentations.** *Comput. Oper. Res.*, 36, pp.1158–1175, 2009.

Lin, F.-C., Hong, J.-S. & Lin, B.M.T. **Sequence optimization for media objects with due date constraints in multimedia presentations from digital libraries.** *Inf. Syst.*, 38, pp.82–96, 2013.

Lin, F.-C., Lai, C.-Y. & Hong, J.-S. **Heuristic algorithms for ordering media objects to reduce presentation lags in auto-assembled multimedia presentations from digital libraries.** *The Electronic Library*, 27, pp.134–148, 2009.

Lin, F.-C., Lai, C.-Y. & Hong, J.-S. **Minimize presentation lag by sequencing media objects for auto-assembled presentations from digital libraries.** *Data Knowl. Eng.*, 66, pp.382–401, 2008.

Lippens, A.F. ***A Review of HTTP Live Streaming***, 2010.

Little, T.D.C. & Ghafoor, A. **Synchronization and storage models for multimedia objects**. *Selected Areas in Communications, IEEE Journal on*, 8, pp.413–427, 1990.

Liu, H. & Zarki, M.E. **A synchronization control scheme for real-time streaming multimedia applications**. *Proceedings of the 13th Packet Video Workshop*, 2003.

Manvi, S.S. & Venkataram, P. **An agent based synchronization scheme for multimedia applications**. *J. Syst. Softw.*, 79, pp.701–713, 2006.

Microsoft. **IIS Smooth Streaming Transport Protocol**, 2009.

Mitten, L.G. **Sequencing n Jobs on Two Machines with Arbitrary Time Lags.** *Management Science*, 5, pp.293–298, 1959.

Monma, C.L. **Sequencing to Minimize the Maximum Job Cost.** *Operations Research*, 28, p.942, 1980.

Moreno, M.F. **Sincronismo entre fluxos de mídia contínua e aplicações multimídia em redes por difusão.** In *Proceedings of the 14th Brazilian Symposium on Multimedia and the Web***.** WebMedia '08. Vila Velha, Brazil: ACM, pp. 202–209, 2008.

Pantos, R. & May, E.W. **HTTP Live Streaming**. *IETF Internet draf, work in progress*, 2011.

Papadimitriou, C.H. & Kanellakis, P.C. **Flowshop scheduling with limited temporary storage**. *J. ACM*, 27, pp.533–549, 1980.

Pinedo, M. ***Scheduling Theory, Algorithms, and Systems***. P. Hall, ed., 2002.

Reisman, A., Kumar, A. & Motwani, J. **Flowshop scheduling/sequencing research: a statistical review of the literature**, 1952-1994. *Engineering Management, IEEE Transactions on*, 44, pp.316–329., 1997.

RFC 3550, Standard 64, **RTP: A Transport Protocol for Real-Time Applications**.

Rodrigues, R.F. & Gomes Soares, L.F. **A framework for prefetching mechanisms in hypermedia presentations**. In *Multimedia Software Engineering, 2002. Proceedings. Fourth International Symposium on*. pp. 278–285, 2002.

Rodrigues, R.F. & Soares, L.F.G. **Produção de Conteúdo Declarativo para TV Digital**. *XXVI Congresso da SBC*, pp.286–300, 2006.

Rodrigues, R.F. & Gomes Soares, L.F. **A framework for event-driven hypermedia presentation systems**. *Multimedia Modeling Conference*, pp.169–185, 2001.

S, M. & A, S.-C. ***Interactive TV Standards: A Guide to MHP, OCAP, and JavaTV***, F. Press, ed., 2005.

Salo, A. **MPEG DASH: A Technical Deep Dive and Look at What´s Next**. *2012 Cable Connection Spring Technical Forum Conference Proceedings*, 2012.

Soares, L.F.G. & Barbosa, S.D. ***Programando em NCL 3.0***, Pontifícia Universidade Católica do Rio de Janeiro (PUC-Rio), 2012.

Sodagar, I. **The MPEG-DASH Standard for Multimedia Streaming Over the Internet**. *MultiMedia, IEEE*, 18, pp.62–67, 2011.

Steinmetz, R. **Human perception of jitter and media synchronization**. *Selected Areas in Communications, IEEE Journal on*, 14, pp.61–72, 1996.

Stockhammer, T. **Dynamic adaptive streaming over HTTP –: standards and**

design principles. In *Proceedings of the second annual ACM conference on Multimedia systems*. MMSys '11. San Jose, CA, USA: ACM, pp. 133–144, 2011.

Tsai, W.-J. & Lee, S.-Y. **Real-time scheduling of multimedia data retrieval to minimize buffer requirement**. *SIGOPS Oper. Syst. Rev.*, 30, pp.67–80, 1996.

Vazirgiannis, M. et al. **Interactive Multimedia Documents: A Modeling, Authoring and Rendering Approach**. *Multimedia Tools Appl.*, 12, pp.145–188, 2000.

Wu, K.-L., Yu, P.S. & Wolf, J.L. **Segmentation of multimedia streams for proxy caching**. *Multimedia, IEEE Transactions on*, 6, pp.770–780, 2004.

Yang, C.-C. & Yang, Y.-Z. **Design and implementation of the just-in-time retrieving policy for schedule-based distributed multimedia presentations**. *J. Syst. Softw.*, 71, pp.49–63, 2004.

Apêndice A
Protocolos de *Streaming*

Antigamente, a maioria dos sistemas de *live streaming* utilizavam os protocolos RTSP (*Real Time Streaming Protocol*) e RTP (*Real time Protocol).* Os programas utilizados para reproduzir os conteúdos (*players*) usavam RTSP para fazer o pedido de *streaming* ao servidor e, após o pedido ser aceito, entravam em ciclo para os enviar, utilizando o protocolo RTP sobre UDP.

Neste modelo de comunicação RTSP/RTP, é o servidor quem decide quais segmentos de conteúdo devem ser enviados no próximo momento, baseando-se nos *acknolegements* e informação de erros enviadas pelo cliente. O papel do cliente é apenas receber os dados enviados pelo servidor e reproduzi-los.

Posteriormente, começou a ser adoptado o modelo HTTP *live streaming*. Nesse modelo é o *player* do cliente que controla quando e quais segmentos de conteúdo devem ser enviados por parte do servidor.

Atualmente, os sistemas baseados no modelo *streaming* dispõem de *adaptive streaming*, isto é, o player é o encarregado de escolher o *bit rate* dos segmentos a serem solicitados ao servidor. Dessa forma, o servidor tem apenas duas tarefas, codificar o conteúdo a transmitir em tempo real com diferentes qualidades e enviar os fragmentos solicitados pelo cliente. Quando *player* do cliente inicia uma sessão *streaming* por primeira vez, o servidor envia um ficheiro metadata (*manifest*) contendo a lista com todos os segmentos disponíveis no servidor. Existem três protocolos comerciais baseados nesse modelo: HLS (HTTP *Live Streaming*) da Apple, HDS (HTTP *Dynamic Streaming*) da Adobe e *Smooth Streaming* da Microsoft.

	Microsoft IIS Smooth Streaming	Adobe Flash Dynamic Streaming	Apple HTTP Adaptive Bitrate Streaming
On-demand & Live Streaming	✓	✓	✓
Live Streaming DVR	✓	Pause & Seek	
Streaming Protocol	HTTP	RTMP	HTTP
Scalability via HTTP Edge Caches	✓		✓
Stateless Server Connection	✓		✓
Supported Platforms	Silverlight, Xbox 360, other Smooth Streaming-compatible players, and iPhone OS 3.0	Flash Player 10, AIR	iPhone OS 3.0, devices running QuickTime X
DRM Support for Live, VOD	PlayReady	None	None
Interoperable DRM(DECE Approved)	✓		
Real-time Client and Server Logging	✓		
Programmable Client Side Switching Logic	✓	✓	
Live In-Stream Ad Integration	✓		

Built-in Analytics Framework	✓		
Delivery to Mobile Devices	✓		✓
Native 64-bit Server Support	✓		✓
Media Container	MPEG 4 – Part 12 (Fragmented MP4)	MPEG 4 – Part 12 (MP4), FLV	MPEG-2 TS
Supported Video Codecs	Codec Agnostic (currently supports VC-1 Advanced Profile & H.264 Baseline, Main and High)	H.264 Baseline, Main, and High; VP6	H.264 Baseline Level 3.0
Supported Audio Codecs	Codec Agnostic (currently supports WMA & AAC)	AAC, MP3	MP3, HE-AAC, AAC-LC
Maximum Bit Rate	No limit	No limit	1.6 Mbps
Default Fragments Length	2 seconds	n/a	10 seconds
End-to-End Latency	As low as 1.5 seconds (configurable)	6 seconds	30 seconds
File type on server	Contiguous	Contiguous	Fragmented
Client Programming Platform	Microsoft .NET Framework	Adobe ActionScript	Objective-C

Tabela A.1 - Tabela comparativa dos protocolos de streaming adaptativo.

A.1
Microsoft Smooth Streaming

É semelhante ao Apple HLS, mas possui diferenças chave. Enquanto o HLS cada vez que tem um segmento novo disponível envia um *manifest* atualizado para o cliente, o *Smooth Streaming* utiliza códigos de tempo nas solicitações dos segmentos. Assim, o cliente não tem que fazer repetidamente *dowload* do *manifest*. Isto faz com que a duração recomendada para os segmentos em *Smooth Streaming* seja de 2 segundos, mas em HLS seja de 10 segundos, de forma a diminuir o envio de *manifest*.

Enquanto o HLS utiliza ficheiros do tipo MPEG-2 *Transport Stream* (MPEG-TS), o *Smooth Streaming* utiliza ficheiros do tipo ISO Base Media File Format (ISO BMFF).

A.2
HTTP Dynamic Streaming

Mas parecido com o *Smooth Streaming* do que com o HLS. Utiliza ficheiros fragmentados do tipo ISO BMFF como o *Smooth Streaming*. A principal diferença entre HDS, HLS e *Smooth Streaming* reside no envio do *manifest*, o HDS utiliza sequências de números nos pedidos dos segmentos, para que o cliente não tenha que fazer repetidamente *download* do *manifest*. Isso faz com que a duração recomendada para os fragmentos em HDS seja entre 2 a 5 segundos.

A.3
HTTP Live Streaming

O HLS permite que o usuário faça *on-demand* ou *live streaming* de áudio e vídeo de um servidor para qualquer dispositivo que tenha pelo menos o iOS 3.0 (iPhone, iPad e Apple TV) ou o Safari 4.0 instalado (Computador). O HLS possui três partes fundamentais: componente do servidor, componente de distribuição e software do cliente.

Uma aplicação multimídia é especificada através de uma URI (*Uniform Resource Identifier*)[10] que aponta para um arquivo que contém uma lista de reprodução (*Playlist*), a qual contém uma lista ordenada de URIs de mídia, além

[10] URI é uma sequência compacta de caracteres que identificam um recurso abstrato ou físico.

de etiquetas de informação. Essas URIs e as suas etiquetas associadas especificam uma série de segmentos de mídia (Pantos & May 2011).

Para tocar um *stream* de mídia, primeiro o cliente obtém o arquivo de *Playlist* e em seguida, adquire e reproduz cada segmento de mídia que encontra-se nesse arquivo. Basicamente, o arquivo de *Playlist* é um arquivo de texto com formato M3U[11] estendido, que contem URIs que identificam os segmentos de mídia ou a sua vez apontam para um arquivo *Playlist* diferente. A duração do arquivo de *Playlist* é igual à soma das durações dos segmentos de mídia que pertencem a ele.

A.4 MPEG-DASH

Com o intuito de normalizar o HTTP *adaptive streaming* nasce o MPEG-DASH (*Dynamic Adaptive Streaming over HTTP*). Esse padrão pode ser considerado como uma combinação dos protocolos Apple HLS, Adobe HDS e *Microsoft Smooth Streaming.* As principais características de MPEG-DASH são:

- O *manifest* é chamado de *Media Presentation Description* (MPD) e é criado no formato XML.
- Permite a utilização tanto de MPEG-TS como de ISO BMFF, facilitando dessa forma a migração para DASH.
- Há uma independência de *codecs*, isto é, existe liberdade para a escolha dos *codecs* a serem utilizados na codificação do conteúdo de áudio e vídeo.
- *Common Encryption* permite a utilização de vários métodos de encriptação.
- Vários perfis podem ser criados. Em cada perfil pode-se definir restrições para formatos multimídia, *codecs*, formatos de proteção, *bit rates*, resoluções, entre outros aspectos relacionados com o conteúdo.

A Figura A.1 apresenta um exemplo simples de um *streaming* sobre demanda, dinâmico e adaptativo (Sodagar 2011). Esse exemplo consiste de componentes de áudio e vídeo. O conteúdo de vídeo está codificado a três *bitrates* diferentes: 5Mbytes, 2Mbytes, e 500 Kbits por segundo. Por outro lado o

[11] M3U (MP3 URL) é um formato de arquivo de computador que armazena listas de arquivos multimídia.

conteúdo de áudio está disponível em dois idiomas: o áudio 1 corresponde ao áudio em inglês enquanto que o áudio 2 é a versão do áudio original que está em francês. O áudio possui duas alternativas de codificação: AAC 128 Kbytes e 48 Kbytes.

Assume-se que, inicialmente, o dispositivo realiza o *streaming* do conteúdo solicitando os segmentos pertencentes ao *bitstream* de vídeo que possuem a qualidade mais alta disponível (5Mbytes) e o áudio em inglês a 128 Kbytes AAC (etiqueta 1 na Figura A.1). Depois de realizar o *streaming* dos primeiros segmentos de vídeo e áudio, e uma vez detectada a largura de banda efetiva (vazão), o dispositivo percebe que a largura de banda atual é menor do que 5 Mbps. Então, no próximo ponto de troca disponível (*key frame*), o vídeo é trocado para 2 Mbps para realizar o *streaming* dos próximos segmentos, enquanto que o *streaming* de áudio continua sendo o mesmo (etiqueta 2 na Figura A.1). O *player* controla constantemente a largura de banda atual e percebe que a largura de banda tem diminuído mais ainda, para um valor mais baixo que 2 Mbps. Portanto, para manter uma reprodução contínua do conteúdo, o dispositivo escolhe os *streams* de vídeo que possuem uma *bitrate* de 500Kbps e os *streams* de áudio de 48 Kbps (etiqueta 3 na Figura A.1). O conteúdo continua tocando até que a largura de banda da rede incrementa e o vídeo troca para a *bitrate* de 2Mbytes (etiqueta 4 na Figura A.1). Depois de um tempo, o usuário decide pausar o vídeo e retrocede-lo. Nesse ponto, para conseguir tocar o vídeo na ordem reversa, o dispositivo inicia o *streaming* a partir do *"trick-mode*[12]*"*, enquanto que o áudio é silenciado (etiqueta 5 na Figura A.1). Quando o usuário chega ao ponto desejado do vídeo, dá um clique para reproduzir o conteúdo com o áudio original em francês. Nesse momento, o dispositivo retoma o *streaming* de vídeo a partir da maior qualidade (5 Mbytes) e o áudio em francês a 128 Kbytes (etiqueta 6 na Figura A.1). O exemplo aqui apresentado é um dos casos de uso mais simples, porém existem casos mais avançados que incluem a troca entre múltiplas visões de câmeras, *streaming* de conteúdo multimídia 3D, vídeo *streams* com subtítulos e muitos outros.

[12] Corresponde a um *bitstream* composto somente por *frames I* que possuem um baixo *frame rate.*

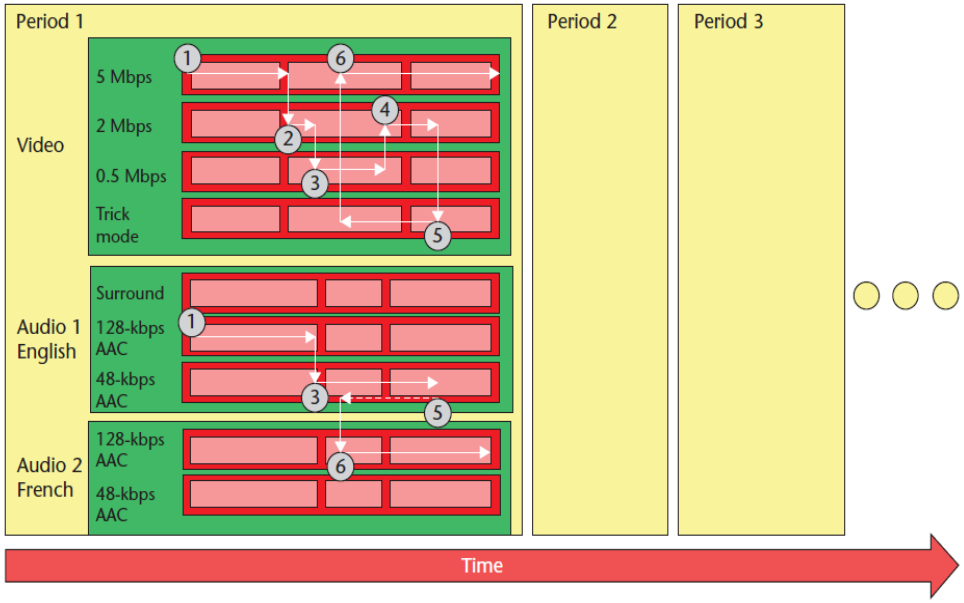


Figura A.1 - Exemplo simples de streaming dinâmico adaptativo (Sodagar 2011)**.**

O conteúdo multimídia está composto por vários componentes (objetos) de mídia (áudio, vídeo, texto, etc.), onde cada um deles possuem diferentes características, as quais, conforme o padrão MPEG-DASH são descritas através de um documento XML chamado *Media Presentation Description* (MPD).

Basicamente, o MPD consiste em uma sequência de um ou mais períodos consecutivos. Cada período contém uma ou mais representações a partir do mesmo conteúdo de mídia. Por sua parte, cada representação consiste de um ou mais segmentos, onde um segmento é definido como um pedaço (*chunk*) de um *stream* de mídia. Cada segmento possui uma URI que corresponde a sua localização no servidor, o qual permite que os segmentos sejam baixados utilizando HTTP GET especificando uma serie de bytes.

Além disso, cada período possui um tempo de início, um tempo de duração, e contém um ou múltiplos conjuntos de adaptação. Um conjunto de adaptação prove informação sobre um ou mais objetos de mídia e as suas várias alternativas de codificação (representações). Por exemplo, um conjunto de adaptação pode conter as diferentes *bitrates* de um objeto de vídeo. Outro conjunto de adaptação pode conter as diferentes *bitrates* de uma mídia de áudio (baixa qualidade *stereo* e alta qualidade *surround sound*). Note-se que cada conjunto de adaptação usualmente inclui múltiplas representações, onde uma representação é uma alternativa de codificação para um objeto de mídia. Os parâmetros que podem

variar de uma representação para outra podem ser: a taxa de bits, a resolução, o número de canais, ou outras características. Na Figura A.2 é apresentado o modelo hierárquico do MPD.

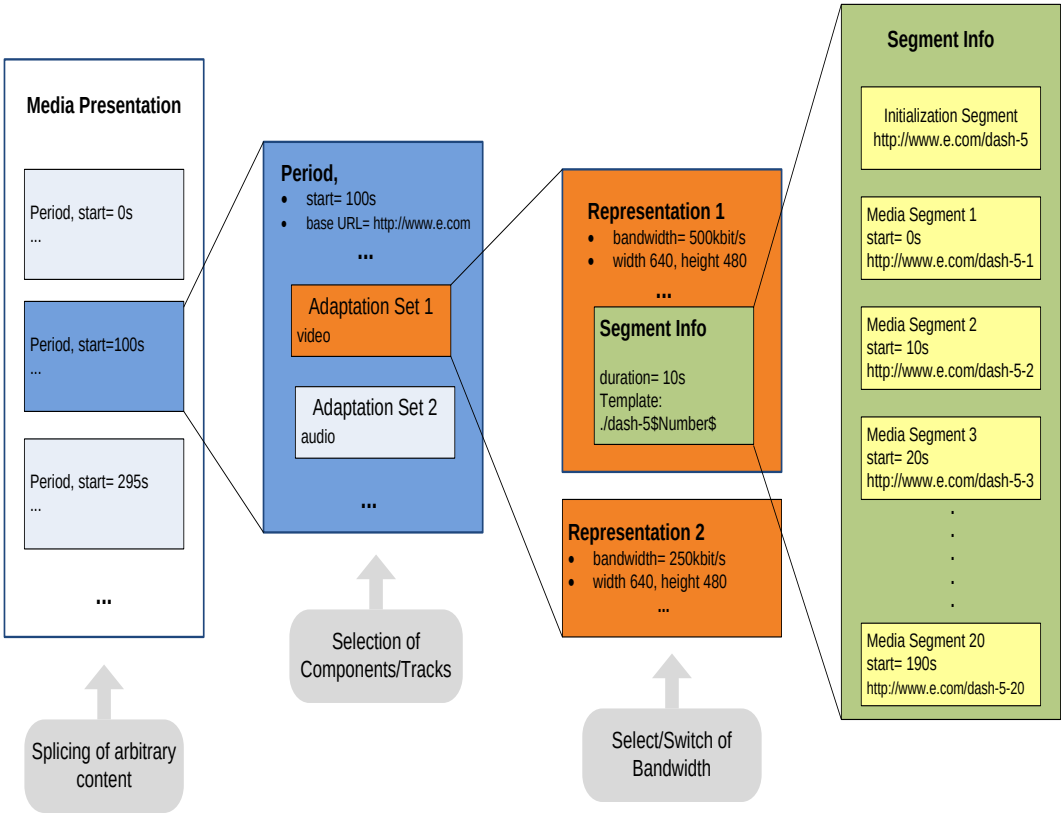


Figura A.2 - Modelo de dados hierárquico da Descrição da Apresentação Multimídia.

Para utilizar esse modelo de dados MPEG-DASH, primeiro o cliente DASH recebe o documento XML que corresponde ao MPD da aplicação. Em seguida, analisa e aprende sobre o conteúdo multimídia disponível, isto é, os tipos de mídias, as resoluções, as larguras de banda mínimas e máximas, a existência de várias alternativas de codificação dos componentes multimídia, as localizações dos conteúdos, os aspectos relacionados com acessibilidade e gerenciamento de direitos digitais (DRM) e outras características. Utilizando a informação descrita anteriormente, o cliente DASH seleciona a alternativa de codificação adequada e inicia o *streaming* do conteúdo através de solicitações HTTP GET. Por outra parte, o cliente encontra-se constantemente monitorando as condições da rede de comunicação para adaptar o conteúdo às diferentes *bitrates* disponíveis (Sodagar 2011). A Figura A.3 representa a comunicação, descrita acima, entre o servidor HTTP e o cliente DASH.

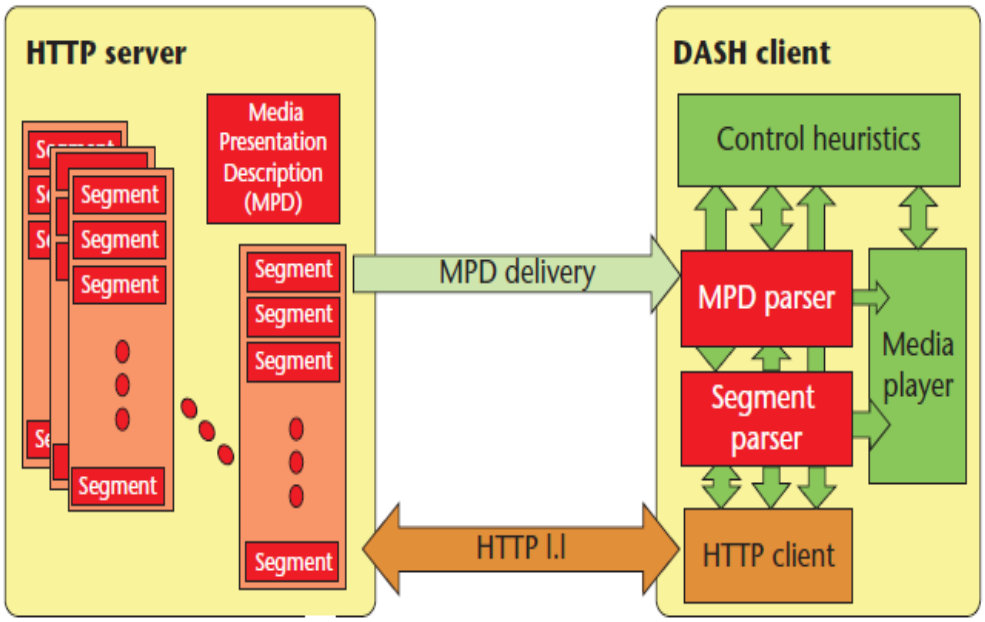


Figura A.3 - Padrão MPEG-DASH.

A.4.1
Formato dos segmentos MPEG-DASH

O conteúdo multimídia pode ser acessado como uma coleção de segmentos. Um segmento é definido como a entidade recebida em resposta a uma solicitação HTTP GET feita pelo cliente DASH.

Como mencionado anteriormente, um componente de mídia é codificado e dividido em múltiplos segmentos consecutivos. O primeiro desses segmentos é um segmento de inicialização, que fornece ao cliente a *metadata* que descreve o conteúdo multimídia, mas não inclui nenhum dado de mídia. Enquanto que os segmentos seguintes contém informação multimídia e cada um deles possui uma única URL, um índice, e um tempo de início de duração explícito ou implícito.

Uma representação pode possuir um segmento de inicialização o qual está seguido de uma sequência de segmentos ou diretamente possuir um segmento de mídia que tenha auto inicialização. Para permitir a recuperação dos segmentos em várias partes, a especificação de MPEG-DASH define um método para sinalizar subsegmentos através de um índice. Esse índice descreve os subsegmentos e os pontos de acesso de *stream* (*Start Access Point* –SAP) para cada segmento. Os SAPs correspondem a pontos de acesso randômico dentro de um *stream* de mídia e permitem a troca de uma representação para outra.

A indexação de segmentos é um conceito importante que permite acessar uma serie de bytes pertencente a um subconjunto de segmentos. Isso permite o rápido acesso ás subestruturas dentro do segmento e conseguir uma rápida comutação entre perfis, o simples acesso randômico, etc. Cada segmento inicia com um índice de segmento ‘sidx’ que define um ou mais fragmentos consecutivos de tipo *movie* (Figura A.4). O ‘sidx’ contém a informação relacionada com as especificações temporais de cada segmento, assim pode ser situado na línea temporal global da apresentação multimídia. Além disso, o ‘sidx’ permite a rápida navegação entre os segmentos, possibilitando a formulação de solicitações partir de uma série de bytes para reduzir a recuperação de conteúdo durante o processo de busca. Ainda assim, o índice de segmento localiza a posição dos pontos de acesso randómico dentro dos segmentos, evitando assim a recuperação desnecessária de informação que esteja antes de um ponto de acesso.

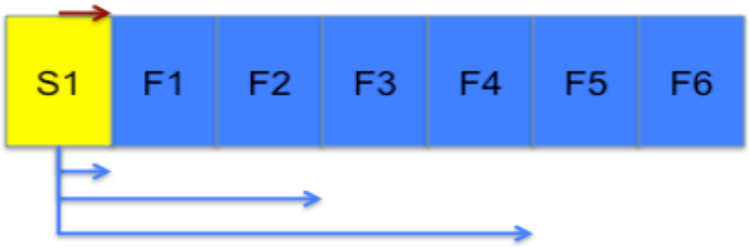


Figura A.4 - Exemplo simples de um segmento (o bloco amarelo representa o índice do segmento S1; Os blocos azuis são os fragmentos movie que contêm os dados; A seta vermelha acima de S1 representa a sequência dos fragmentos no primeiro loop; As setas azuis mostram os ponteiros do índice no segundo loop) (Stockhammer 2011)**.**

A partir do fato que os segmentos podem possuir diferentes tamanhos, o primeiro ‘sidx’ poderia ou não descrever todos os detalhes dos seguintes fragmentos que encontram-se dentro do segmento. Com isso, para evitar um ‘sidx’ longo, isto é, no caso de ter segmentos longos, a informação poderia ser fornecida de uma maneira aninhada, de tal forma que o ‘sidx’ faça referência não só ao fragmento inicial senão também a outros ‘sidx’. Na Figura A.5, encontram-se representados três métodos para indexar informação de segmentos em forma aninhada. Esses métodos são: hierárquico, cadeia e híbrido.

No modo hierárquico, o primeiro *loop* para S1 aponta para o primeiro

fragmento *movie*, porém o primeiro ponteiro no segundo *loop* para S1 faz referência a um ‘sidx’. De maneira análoga, qualquer outro ponteiro no segundo loop para S1 faz referência a um segmento ‘sidx’. Somente no segundo nível (S2/3/4), os ponteiros apontam diretamente para os fragmentos *movie*. Por consequência, com a recuperação de ‘sdix’ de S1, é possível realizar uma rápida navegação através dos segmentos disponíveis. No caso do método cadeia ou “*daisy chain*”, S1 e S3 apontam para os seus fragmentos *movie* e para outros ‘sidx’. Essa estrutura permite a navegação rápida dos fragmentos iniciais dos segmentos, onde os últimos fragmentos podem necessitar de algum tipo de resolução sequencial (Stockhammer 2011).

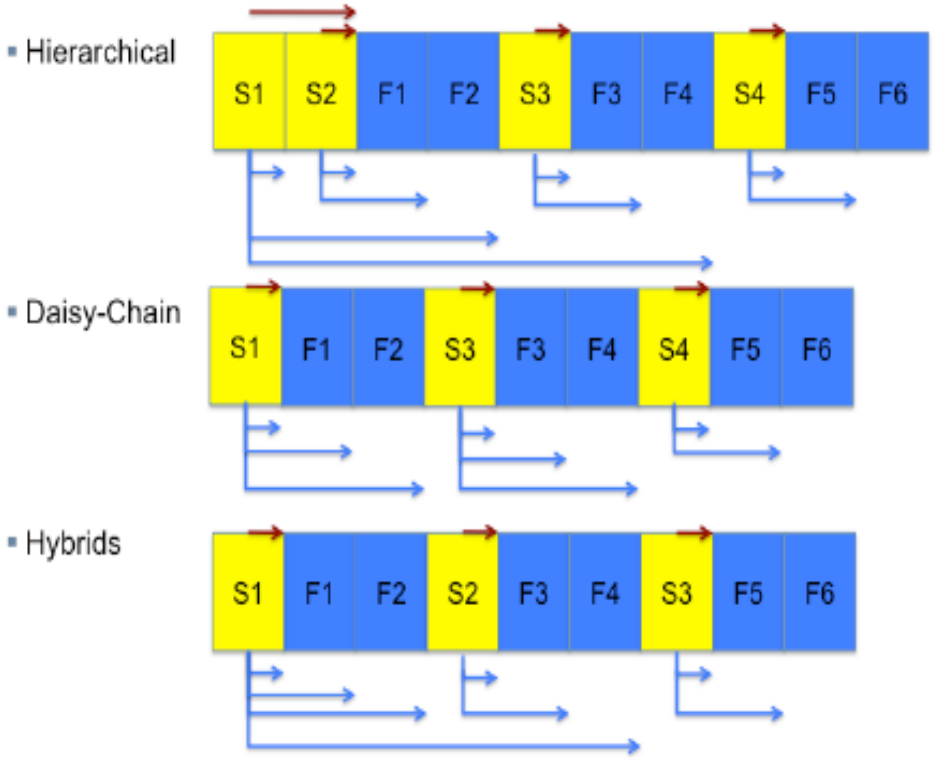


Figura A.5 - Índices de Segmentos Aninhados (Stockhammer 2011)**.**

Apêndice B
O sincronismo em NCL

Em NCL o sincronismo não é feito por marcas de tempo (*timestamps*), senão por relações de causalidade definidas nos conectores (*connectors*)[13]. Um elemento <causalConnector> representa uma relação causal que pode ser utilizada por elementos <link> na definição de relacionamentos entre objetos. Em uma relação causal, uma condição deve ser satisfeita para que ações possam ser disparadas. A NCL, em sua versão 3.0, define quatro tipos de eventos: eventos de apresentação, seleção, atribuição e composição. A Tabela B.1 apresenta os papéis e as condições definidos em NCL.

Os papéis de condição "*onBegin*", "*onEnd*", "*onAbort*", "*onPause*", e "*onResume*", assim como os papéis de ação "*Start*", "*Stop*", "*Abort*", "*Pause*" e "*Resume*" estão relacionados às possíveis transições de estados de eventos de apresentação, essas transições são definidas na máquina de estado NCL (Figura 4).

Por sua vez, os papéis de condição "*onSelection*", "*onBeginSelection*", "*onEndSelection*" estão relacionados às possíveis transições de estados de eventos de seleção de âncoras de conteúdo. Eles são ligados à interatividade, realizada por médio de dispositivos de entrada. Em contrapartida os papéis de condição "*onBeginAttribution*", "*onEndAttribution*", "*onAbortAttribution*", "*pauseAttributtion*", "*stopAttribution*", "*abortAttribution*", "*pauseAttribution*" e "*resumeAttribution*", estão relacionadas aos eventos de atribuição, isto é, à manipulação de valores de propriedades.

[13] Bases de conectores, conectores e seus atributos são especificados nos módulos ConnectorBase, ConnectorCommonPart, CausalConnector, CausalConnectorFunctionality, ConnectorCausal-Expression, ConnectorAssementExpression e ConnectorTransitionAssessment [ABNT, NBR 15606-2,2011;ITU-T, H.761,2011].

Papel (*role*)	Valor de Transição	Tipo de Evento
onBegin	Start	apresentação
onEnd	Stop	apresentação
onAbort	Abort	apresentação
onPause	Pause	apresentação
onResume	Resume	apresentação
onSelection	Stop	seleção
onBeginSelection	Start	seleção
onEndSelection	Stop	seleção
onAbortSelection	Abort	seleção
onPauseSelection	Pause	seleção
onResumeSelection	Resume	seleção
onBeginAttribution	Start	atribuição
onEndAttribution	Stop	atribuição
onPauseAttribution	Pause	atribuição
onResumeAttribution	Resume	atribuição
onAbortAttribution	Abort	atribuição

Tabela B.1 - Valores reservados para papéis utilizados como condição em NCL.