UNIVERSIDADE FEDERAL DE JUIZ DE FORA
INSTITUTO DE CIÊNCIAS EXATAS
PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Marina Ivanov Pereira Josué

Gerenciamento da Transmissão de Aplicações Hipermídia em Modo Push

Juiz de Fora

2016

Marina Ivanov Pereira Josué

Gerenciamento da Transmissão de Aplicações Hipermídia em Modo Push

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação, do Instituto de Ciências Exatas da Universidade Federal de Juiz de Fora como requisito parcial para obtenção do título de Mestre em Ciência da Computação.

Orientador: Marcelo Ferreira Moreno

Juiz de Fora

2016

Ficha catalográfica elaborada através do programa de geração
automática da Biblioteca Universitária da UFJF,
com os dados fornecidos pelo(a) autor(a)

Pereira Josué, Marina Ivanov .
Gerenciamento da Transmissão de Aplicações Hipermídia em Modo Push / Marina Ivanov Pereira Josué. -- 2016.
76 f.

Orientador: Marcelo Ferreira Moreno
Dissertação (mestrado acadêmico) - Universidade Federal de Juiz de Fora, Instituto de Ciências Exatas. Programa de Pós Graduação em Ciência da Computação, 2016.

1. Analise de Conteúdo Hipermídia. 2. Gerenciamento de Transmissão . 3. Grafo Temporal. I. Ferreira Moreno, Marcelo, orient. II. Título.

Marina Ivanov Pereira Josué

Gerenciamento da Transmissão de Aplicações Hipermídia em Modo Push

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação, do Instituto de Ciências Exatas da Universidade Federal de Juiz de Fora como requisito parcial para obtenção do título de Mestre em Ciência da Computação.

Aprovada em 30 de Junho de 2016.

BANCA EXAMINADORA

Prof. D.Sc. Marcelo Ferreira Moreno - Orientador
Universidade Federal de Juiz de Fora

Prof. D.Sc. Eduardo Barrére
Universidade Federal de Juiz de Fora

Prof. D.Sc. Carlos de Salles Soares Neto
Universidade Federal do Maranhão

Prof. D.Sc. Romualdo Monteiro de Resende Costa
Universidade Federal de Juiz de Fora

AGRADECIMENTOS

Agradeço em primeiro lugar a Deus, pelo dom da vida e por iluminar meu caminho ao longo de toda minha trajetória.

Aos meus pais, Luciana e Sebastião, que com muito carinho e apoio, não mediram esforços para que eu chegasse até esta etapa da minha vida.

A meus familiares e amigos que tanto amo, pelo apoio e por me trazerem leveza e sorrisos em meio aos desafios enfrentados ao longo do mestrado.

Ao Washington, pelo companheirismo, incentivo e compreensão nos momentos em que não pude estar presente.

Aos colegas do Mestrado e do Laboratório de Aplicações e Inovação em Computação (LApIC), pelos bons momentos e ensinamentos compartilhados, em especial, Frâncila, Gustavo, Laura, Marcos, Rodrigo e Thomás.

Ao professor Marcelo Moreno, pela orientação, paciência, dedicação e pelo compartilhamento de seus conhecimentos ao longo desses anos. Obrigada por acreditar em mim para o desenvolvimento deste trabalho.

Ao professor Romualdo, pelas sugestões, dicas e apoio que enriqueceram este trabalho.

A todos os professores do Programa de Pós-Graduação em Ciência da Computação, que de alguma forma contribuíram para minha formação. E a CAPES, pelo suporte financeiro.

RESUMO

Atualmente o conteúdo hipermídia pode ser entregue utilizando diferentes tecnologias de rede, como a TV Digital terrestre por satélite, IPTV e Web. Por isso, as máquinas de apresentação hipermídia devem levar em conta as especificidades dessas redes suportadas, de modo a prover os níveis de QoS/QoE esperados pelo usuário do conteúdo. Máquinas de apresentação avançadas podem empregar também a análise de conteúdo para auxiliar na tarefa de manutenção dos níveis de QoE. De modo especial, quando o conteúdo hipermídia inclui dados enviados por pull, as máquinas de apresentação podem criar um Plano de Pré-Busca baseado no comportamento da apresentação extraído da especificação do conteúdo. Entretanto, quando o conteúdo hipermídia inclui dados enviados por push, a análise do conteúdo deve ser transferida para o lado do servidor e se basear na construção de um Plano de Transmissão de Conteúdo. O Plano de Transmissão de Conteúdo é uma estrutura de dados que prevê os instantes em que objetos de mídia devem ser transmitidos, e por quanto tempo, para otimizar o uso de recursos fim-a-fim como largura de banda e espaço de armazenamento nos receptores. Este trabalho propõe um framework para gerenciamento da entrega de conteúdo hipermídia em modo push. O framework é genérico o suficiente para ser adaptável a diferentes cenários de entrega de conteúdo que podem empregar diferentes protocolos e técnicas de gerenciamento. Alguns cenários de instanciação do framework e seus respectivos resultados são discutidos nesta dissertação.

Palavras-chave: Análise de conteúdo hipermídia. Gerenciamento de Transmissão. Grafo Temporal. Otimização de Recursos.

ABSTRACT

Nowadays hypermedia content may be delivered using different networking technologies, such as terrestrial broadcasting, satellite, IPTV and the Web. Therefore hypermedia presentation engines must be designed taking into consideration the specificities of their supported networks, in order to provide the expected QoS/QoE levels. Advanced presentation engines should also employ hypermedia content analysis to help on the task of maintaining QoE levels. Specifically, when the hypermedia content includes pulled data, presentation engines may create a Content Prefetching Plan based on the presentation behavior learned from the specification of that content. However, when the hypermedia content includes pushed data, this content analysis should be transferred to the server side and be taken as a basis for building the so-called Content Transmission Plan. The Content Transmission Plan is a data structure that predicts the time when media objects should start to be transmitted and for how long, in order to optimize end-to-end resource usage such as communication bandwidth and storage space in receivers. This work proposes a framework for managing the push-mode delivery of hypermedia content. The framework is generic enough to be instantiated over different content delivery scenarios that may employ different protocols and management techniques. Several instantiation scenarios and their respective results are also discussed in this dissertation.

Keywords: Hypermedia content analysis. Transmission Management. Temporal Graph. Resource usage optimization. Push delivery.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

HTG Hypermedia Temporal Graph

IPTV Internet Protocol Television

NCL Nested Context Language

FLUTE File Delivery over Unidirectional Transport

FDT File Delivery Table

TOI Transport Object Identifier

TSI Transport Session Identifier

RTP Real-time Transfert Protocolo

RTCP Real-time Transfert Control Protocol

QoS Quality of Service

QoE Quality of Experience

DSM-CC Digital storage media command and control

ALC Assynchronous Layered Coding

FEC Forward Error Correction

SUMÁRIO

1 INTRODUÇÃO

Os sistemas de entrega de conteúdo hipermídia dão suporte ao envio de objetos de diferentes tipos de mídias, como texto, imagem, áudio e vídeo. Tais objetos de mídia podem estar encadeados, sincronizados entre si, no tempo e no espaço, conforme definido pelo autor do conteúdo por meio de uma linguagem de especificação hipermídia. O conteúdo hipermídia pode ser visto como uma aplicação interativa, que, em essência, define um comportamento de apresentação dos objetos de mídia que a compõem.

Hoje em dia, a entrega de aplicações hipermídia pode ocorrer através de diferentes técnicas e tecnologias de rede, em plataformas como a TV Digital Terrestre por Satélite, IPTV ou a Web. As máquinas de apresentação hipermídia devem ser projetadas levando em consideração as especificidades das tecnologias de rede por elas suportadas, de forma a prover os níveis esperados de QoS/QoE aos seus usuários.

Primordialmente, as máquinas de apresentação buscam assegurar que os relacionamentos de sincronismo espaço-temporal entre os objetos de mídia mantenham-se conforme especificado pelo autor da aplicação. Porém, esses mesmos objetos de mídia são comumente entregues ao longo da apresentação da aplicação, o que exige técnicas avançadas de gerenciamento, que incluem a análise do comportamento temporal da própria aplicação.

Quando os instantes dos relacionamentos entre objetos de mídias são determinísticos, ou seja, conhecidos desde a autoria da aplicação, é possível recuperá-los em tempo de apresentação e, então, representar o sincronismo temporal entre os objetos em relação a intervalos absolutos de um eixo temporal. Para os casos de relacionamentos que envolvam eventos não-determinísticos, é possível também uma representação, de forma relativa, condicionada à ocorrência desses eventos.

O comportamento temporal de uma aplicação hipermídia deve então ser modelado por uma estrutura, como o HTG (Hypermedia Temporal Graph) (COSTA et al., 2008). Através do HTG, pode-se calcular os momentos ou intervalos de ocorrência dos eventos definidos na aplicação hipermídia. O HTG modela cadeias temporais determinísticas e não-determinísticas. Em diversos trabalhos, ele vem sendo aplicado como base para a construção dos Planos de Apresentação (COSTA et al., 2008) e de Pré-busca (FIGUEROA, 2014) de Conteúdo Hipermídia, os quais auxiliam a máquina de apresentação Ginga-NCL

(SOARES et al., 2010) na manutenção do sincronismo temporal.

De fato, se os objetos de mídia são entregues em modo pull, ou seja, são solicitados pela máquina de apresentação conforme são necessários na apresentação, um Plano de Pré-busca baseado no comportamento temporal da aplicação deve ser empregado para coordenar a chegada a tempo dos objetos de mídia.

No entanto, se os objetos de mídia são entregues em modo push, a responsabilidade por analisar o comportamento da aplicação para fins de otimização no uso dos recursos deve recair sobre o lado servidor. Tal análise deve servir de base para a construção de um novo plano, neste trabalho denominado como Plano de Transmissão de Conteúdo Hipermídia.

O Plano de Transmissão é um tipo de plano que também pode ser construído a partir do HTG, e contém informações sobre quando cada objeto de mídia entregue em modo push deve ser inserido ou removido do stream de dados. Nos métodos de trasmissão por broadcast ou multicast, esse fluxo de dados é normalmente implementado por meio de um protocolo de transmissão cíclica (ISO/IEC, 1998). O Plano de Transmissão deve levar em conta não somente o momento em que cada objeto de mídia será necessário na apresentação da aplicação, mas também o tempo a ser gasto para que os receptores possam obtê-lo no stream de dados.

A utilização do plano de transmissão tende a levar a uma economia de recursos de maneira fim-a-fim. Primeiramente, há uma redução na ocupação de banda passante pelo servidor de aplicações hipermídia, porque ao invés de enviar todos os objetos de mídia da aplicação a todo instante, o plano de transmissão faz com que o stream de dados seja composto apenas pelos objetos necessários à sua execução, em um dado intervalo de tempo. Outro recurso que tende a ser poupado é o espaço de armazenamento temporário nas caches dos receptores, uma vez que o stream de dados estará sempre atualizado com os objetos necessários, podendo ser removidas cópias locais dos objetos que não mais o são.

Este trabalho propõe um framework voltado à transmissão de aplicações hipermídia entregues em modo push que considera o comportamento temporal da aplicação para o gerenciamento da entrega de dados. O framework é genérico o suficiente para ser adaptável a diferentes cenários e tecnologias de rede, e busca otimizar a utilização de recursos fim-a-fim, como a banda passante necessária para o stream de dados e o espaço de armazenamento temporário nos receptores. O framework define estruturas de dados,

componentes gerenciadores e suas estratégias para a construção e aplicação do Plano de Transmissão de Conteúdo Hipermídia, promovendo uma coordenação dos instantes de envio dos objetos de mídia. O uso do framework é exemplificado em cenários de entrega de aplicações hipermídia especificadas na linguagem NCL (Nested Context Language) (SOARES, 2009).

Esta dissertação está estruturada da seguinte forma. O Capítulo 2 apresenta a fundamentação teórica, e descreve conceitos-base para a implementação do trabalho. Estes conceitos englobam a linguagem NCL, para especificação de aplicações hipermídia, e protocolos para entrega de conteúdo em modo push. O Capítulo 3 contém os principais trabalhos relacionados. O Capítulo 4 define o framework genérico proposto pelo trabalho com o objetivo de otimizar a entrega de aplicações hipermídia. O Capítulo 5 descreve cenários de instanciação do framework proposto, especificando alguns de seus componentes. O Capítulo 6 contém a avaliação da solução proposta e os resultados obtidos. Por fim, o Capítulo 7 apresenta as conclusões do trabalho, contribuições e os trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo apresenta os principais conceitos relacionados à proposta deste trabalho, como a linguagem NCL, para especificação de aplicações hipermídia, estudos relacionados a mecanismos para transmissão do conteúdo em modo push, como o carrossel DSM-CC e o protocolo FLUTE, e a estrutura HTG (COSTA et al., 2008) voltada para o controle da apresentação da aplicação.

2.1 NCL - NESTED CONTEXT LANGUAGE

A NCL é uma linguagem declarativa, que se baseia no modelo conceitual NCM (Nested Context Model) (SOARES; RODRIGUES, 2005), e é utilizada para definir como os objetos de mídia são estruturados e relacionados, no tempo e no espaço. A linguagem não restringe, nem prescreve os tipos de conteúdo dos objetos de mídia de uma aplicação e busca facilitar a autoria de aplicações através de recursos para reúso de bases (bases de regiões, descritores e conectores) e de conteúdo, além da importação de documentos. (SOARES; RODRIGUES, 2006)

Em NCL a sincronização é definida baseada em eventos, em que as ações associadas à apresentação dos objetos de mídia são executadas a partir da ocorrência de eventos. Quando a sincronização baseia-se em eventos, a especificação de uma aplicação pode ser completamente definida de forma relativa, sem referências absolutas de tempo. Um evento pode ser classificado como determinístico ou não-determinístico. O evento é determinístico quando é possível determinar o instante exato no tempo de sua ocorrência, e não-determinístico caso contrário.

Eventos não-determinísticos podem estar relacionados a ações de interação do usuário, adaptação de conteúdo, atribuição de valores a variáveis do ambiente, ou à existência ou não de múltiplos dispositivos conectados ao dispositivo base.

Com base nestas características da linguagem, como suas facilidades de uso e os vários meios de comunicação multimídia que a empregam, optou-se por utilizar a NCL como base de especificação de aplicações hipermídia, para alvo de análise conforme a solução proposta.

A estrutura do documento NCL é composta pelo cabeçalho e corpo do programa. O

cabeçalho contém as bases de regiões, descritores e conectores, e no corpo são especificados os nós de conteúdo, nós de contexto e links. A construção de uma aplicação hipermídia utilizando NCL se baseia em quatro questões: *O que exibir? Onde exibir? Como exibir? Quando exibir?*

O elemento **<media>** é usado para especificar os objetos de mídia (*O que?*) utilizados na aplicação, e seus atributos. Regiões (elemento **<region>**) definem onde as mídias (*Onde?*) ou nós de conteúdo devem ser exibidos. A forma de exibição (*Como?*) é definida pelos descritores (elemento **<descriptor>**), que podem especificar comportamentos adicionais para as mídias, além de associar as mídias às regiões previamente definidas no **<regionBase>**. As regiões e os descritores são definidos em uma base de regiões (elemento **<regionBase>**) e base de descritores (elemento **<descriptorBase>**), respectivamente.

Sobre cada objeto de mídia é possível criar âncoras(elemento **<area>**) que delimitam trechos (no tempo ou espaço) do conteúdo. Valores de propriedades de um objeto de mídia podem ser definidos pelo elemento **<property>**, como o volume de um objeto de mídia do tipo áudio, ou o espaço na tela em que uma mídia do tipo imagem será exibida. Âncoras e propriedades são generalizadas como interfaces de um objeto de mídia.

O sincronismo entre mídias (*Quando?*) é realizado por relações de causalidade definidas nos conectores (tag **<causalConnector>**), que fornecem a semântica da associação entre objetos de mídia. A partir de uma relação causal, representada por um elemento **<causalConnector>** na base de conectores, é possível definir elementos **<link>** que estabelecem relacionamentos entre objetos de mídia.

O elo preenche os papéis declarados em um conector associando-os a objetos de mídia da aplicação e suas interfaces. Os relacionamentos de sincronismo espacial e temporal são definidos de forma desacoplada à definição do conteúdo dos objetos de mídia relacionados.

Os objetos de mídia, contexto, elos e switches podem ser agrupados em contextos, onde o **<body>** é um contexto particular que representa a aplicação como um todo. Os contextos podem ser aninhados, por exemplo, para refletir a estrutura do documento e ajudar o autor a organizar os segmentos da aplicação. (SOARES, 2009)

Através de elementos da linguagem, a NCL dá suporte a certos requisitos para aplicações hipermídia, como suporte a sincronismo espaço-temporal, à adaptação de conteúdo e da forma como este é exibido, suporte à edição em tempo de execução da aplicação. A

linguagem é adotada para codificação de dados, no padrão ISDB-TB para TV Digital terrestre (ABNT, 2016) e Recomendação ITU-T H.761 (ITU-RECOMMENDATION, 2009) para serviços IPTV.

A NCL também permite que seja utilizado mais de um dispositivo para exibição do conteúdo da aplicação. Esses dispositivos podem ser classificados como ativos, caso sejam capazes de executar as funções de exibidores de mídia NCL, ou passivos, caso contrário. Para utilizar múltiplos dispositivos, deve-se especificar, para cada base de regiões, a que dispositivo as regiões definidas se referem, utilizando o atributo *device* do elemento **<regionBase>** (Figura 2.1).

```
<regionBase>
        <region id="screenReg" width="100%" height="100%" zIndex="1">
                <region id="frameReg" left="5%" top="6.7%" width="18.5%"
                                                height="18.5%" zIndex="2"/>
                <region id="iconReg" left="87.5%" top="11.7%" width="8.45%"
                                                height="6.7%" zIndex="2"/>
        </region>
</regionBase>

<regionBase device="systemScreen(1)">
        <region id="backgroundReg" width="100%" height="100%" zIndex="1">
                <region id="shoesReg" left="0" top="25%" width="50%"
                                                height="50%" zIndex="2"/>
                <region id="formReg" left="50%" top="0" width="50%"
                                                height="100%" zIndex="2"/>
        </region>
</regionBase>
```

Figura 2.1: Uso de múltiplos dispositivos de exibição

Uma aplicação pode ter seu comportamento ou sua forma de apresentação adaptados durante sua execução, através da avaliação de regras para seleção de objetos e descritores. As regras (elemento **<rule>**) são definidas no cabeçalho do documento, e são compostas por uma variável a ser avaliada, o operador de comparação e um valor a ser comparado.

As composições com nós alternativos (elementos **<switch>**), podem conter objetos de mídia, contextos ou outros switches. Estas composições possibilitam a adaptação pela escolha de qual nó será selecionado, através de regras de mapeamento, definidas por elementos **<bindRule>**.

Também é possível selecionar um descritor conforme uma regra utilizando o elemento **<descriptorSwitch>**. A Figura 2.2 ilustra a utilização de uma base de regras que avaliam o valor da propriedade "system.language" e um switch para a seleção de uma mídia de áudio conforme a regra em vigor.

```
<head>
        ...
        <ruleBase>
                <rule id="rEn" var="system.language" comparator="eq" value="en"/>
                <rule id="rPt" var="system.language" comparator="eq" value="pt"/>
        </ruleBase>

</head>
<body>
        <media id="noSettings" type="application/x-ncl-settings">
                <property name="system.language"/>
        </media>

        ...
        <switch id="switchIdioma">
                <bindRule rule="rEn" constituent="audioEn"/>
                <bindRule rule="rPt" constituent="audioPt"/>
                <media id="audioEn" src="media/audioEn.mp3" descriptor="dAudio"/>
                <media id="audioPt" src="media/audioPt.mp3" descriptor="dAudio"/>
        </switch>
        ...
</body>
```

Figura 2.2: Regras e switch para seleção de áudio de acordo com o idioma

2.2 HYPERMEDIA TEMPORAL GRAPH – HTG

Aplicações hipermídia podem ser especificadas e apresentadas a partir da sincronização baseada em eventos ou através de timeline. Quando a sincronização é baseada em eventos, o controle do tempo da apresentação e da transmissão dos dados torna-se mais difícil, uma vez que os momentos de ocorrência dos eventos nas aplicações não são explícitos. Neste contexto, uma estrutura com o propósito de modelar o comportamento temporal de aplicações hipermídia, o HTG (*Hypermedia Temporal Graph*) é definido em (COSTA, 2010).

O grafo temporal oferece suporte ao início ou à retomada síncrona da apresentação, a partir de um instante de tempo qualquer, e pode ser empregado nas decisões relacionadas à exibição e ao transporte dos objetos de mídia. Por exemplo, a partir da estrutura é possível prever a ocorrência de eventos ao longo da apresentação, reduzindo a necessidade de ajustes provocados por atrasos.

Os eventos que podem ocorrer sobre os conteúdos das mídias são representados no HTG por meio de vértices, enquanto os relacionamentos são representados por arestas dirigidas. A estrutura permite a representação dos mesmos eventos definidos pela linguagem NCL (apresentação, seleção e atribuição), cada qual controlado por uma máquina de estados.

As máquinas de estados são definidas para cada evento e devem ser controladas pelo formatador NCL (SOARES, 2009). As transições de estados de eventos de apresentação de conteúdo, estão relacionadas aos papéis de condição "*onBegin*", "*onEnd*", "*onAbort*",

"*onPause*"e "*onResume*", e também com os papéis de ação "*start*", "*stop*", "*abort*", "*pause*"e "*resume*", como mostra a Figura 2.3.

O grafo obtido deve preservar todos os relacionamentos entre os eventos (determinísticos e não-determinísticos), e o estado atual de cada máquina de estados associada a um evento. Ao controlar os estados de um evento, é possível determinar, com precisão, qual o comportamento esperado de um evento, considerando o seu estado atual e a transição realizada.

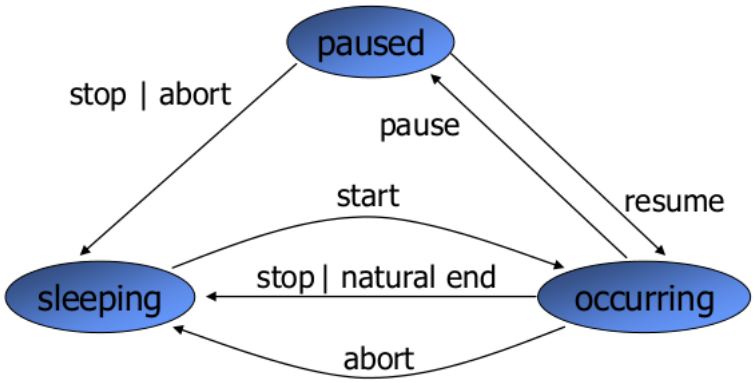

Figura 2.3: Máquina de estados de eventos (COSTA, 2010)

No HTG, os vértices contêm triplas, compostas pela transição da máquina de estado do evento, que pode ser identificada pela ação que dispara a transição (start, stop, abort, natural end, pause ou resume), pelo tipo de evento (apresentação, seleção, atribuição etc.) e pelo identificador da interface que define o evento. Uma interface pode representar o conteúdo de uma mídia, uma âncora do conteúdo ou até mesmo uma propriedade do objeto. Quando identifica uma propriedade, seu valor também é acoplado ao identificador da interface.

A definição do grafo contém o conjunto de vértices e arestas dirigidas, e maneiras de caminhamento no mesmo, que são condições (simples ou compostas) e prioridades para a verificação das condições associadas às arestas com origem em um mesmo vértice. As condições de uma aresta podem ser negadas, através do operador NOT, e operadores OR e AND podem ser utilizados para definir condições compostas.

Costa (2010) define algumas regras para a construção de um grafo que modela o comportamento temporal da aplicação hipermídia, especificada através de qualquer linguagem baseada em eventos. A Tabela 2.1 lista tais regras.

Deve-se destacar que todos os vértices podem ser origem e destino de arestas construí-

Regra	Descrição
1	Um vértice V do tipo (transition, event, anchorId) para eventos de apresentação ou seleção, qualquer que seja a transição (transition) e a âncora de conteúdo (anchorId), ou um vértice T do tipo (transition,attribution, objectId.propertyId) para eventos de atribuição, somente deve existir se V ou T for utilizado no controle da apresentação.
2	Um vértice V do tipo (stop, event, anchorId) no HTG pode ser equivalente a um vértice W do tipo (natural end, event, anchorId), para qualquer evento (event) e âncora de conteúdo (anchorId), caso a sintaxe de autoria não defina formas diferentes para o término de um evento (natural end e stop). Nesse caso, se necessários na apresentação (regra 1), um único vértice deve ser utilizado, contendo a transição stop.
3	Um vértice V do tipo (abort, event, anchorId) ou um vértice T do tipo (abort, event, objectId.propertyId) não pode ser vértice de origem de nenhum outro vértice, para qualquer evento (event) e âncora de conteúdo (anchorId) ou objeto (objectId) e sua propriedade (propertyId).
4	Um vértice T do tipo (start, attribution, objectId.propertyId) deve possuir, na própria tripla, o valor a ser atribuído à propriedade (propertyId). Se existirem diferentes valores a serem atribuídos a uma mesma propriedade, definida sobre um mesmo objeto (objectId), deve ser criado um vértice T n do tipo (start, attribution, objectId.propertyId = value) para cada valor (value) diferente.
5	Um vértice T do tipo (natural end, attribution, objectId.propertyId), se necessário no grafo (regra 1), deve ser o destino de todo vértice S do tipo (start, attribution, objectId.propertyId = value), para um mesmo objectId e propertyId, independente do valor a ser atribuído (value).
6	Um vértice T do tipo (start, selection, anchorId), se necessário no grafo (regra 1), deve ser o destino de um vértice origem S do tipo (start, presentation, anchorId), quando anchorId representa a mesma âncora de conteúdo para S e T.
7	Um vértice T do tipo (natural end, presentation, anchorId) deve ser o destino de um vértice origem S do tipo (start, presentation, anchorId), quando anchorId representa a mesma âncora de conteúdo para S e T, exceto quando a duração desse evento não tem relação com a sua transição de início.
8	Um vértice T do tipo (start, presentation, anchorIdT) deve ser o destino de um vértice origem S do tipo (start, presentation, anchorIdS), se anchorIdT é uma âncora equivalente a uma parte do conteúdo de anchorIdS.
9	Arestas do tipo ((start, presentation, anchorId), (natural end, presentation, anchorId)) para uma mesma anchorId, devem ter como condição a duração do evento de apresentação dessa âncora (anchorId) ou do objeto, caso anchorId seja uma parte do conteúdo desse objeto.
10	Arestas do tipo ((start, presentation, anchorIdS), (start, presentation,anchorIdT)) devem ter como condição parte da duração do evento de apresentação sobre anchorIdS, se anchorIdT é uma âncora equivalente a uma parte do conteúdo de anchorIdS.

Tabela 2.1: Regras para a construção do HTG (COSTA, 2010)

das para representar relacionamentos especificados na aplicação, exceto os vértices que representam transições do tipo “abort”(regra 4), e os vértices que representam o início do evento de seleção (regra 5).

A utilização do HTG para modelar o comportamento de uma aplicação hipermídia especificada em NCL, considera elementos da linguagem para a criação de novos vértices e arestas. A construção do HTG utilizando aplicações NCL é apresentada a seguir.

Para cada elemento **<media>** e âncora de conteúdo (elemento **<area>**), encontrado na especificação da aplicação, deve-se criar um vértice do tipo (start, presentation, idInterface). Caso essa mídia ou âncora possuam uma duração finita (explícita ou com fim natural), o vértice (stop, presentation, idInterface), é criado, juntamente com uma aresta que o liga ao vértice anterior. A aresta criada neste caso tem como condição de caminhamento a duração da mídia ou âncora de conteúdo.

O vértice que define o início da apresentação de âncoras que representam uma porção de um objeto deve ser o destino de uma aresta com origem no vértice correspondente ao início do todo da apresentação daquele objeto. Na Figura 2.4, o trecho do documento NCL especifica uma mídia do tipo vídeo que contém três áreas, nomeadas de aMap, aTicket, aQuiz. Estas áreas equivalem a trechos do conteúdo de vídeo principal, com instantes de início e fim demarcados.

```
<media id="video" src="media/video.mp4" descriptor="dVideo">
        <property name="bounds"/>
        <area id="aMap" begin="15s" end="30s"/>
        <area id="aTicket" begin="50s" end="60s"/>
        <area id="aQuiz" begin="85s" end="135s"/>
</media>
```

Figura 2.4: Definição de âncoras de conteúdo de um vídeo

Para o exemplo da Figura 2.4, serão criados seis novos vértices, três representando o início de cada âncora de conteúdo, e outros três para o seu fim. Além disso, são criadas arestas ligando os vértices de inicío e de fim de cada âncora, e que ligam o início de cada âncora ao início do conteúdo como um todo, como mostra a Figura 2.5.

A duração do evento de apresentação de uma âncora de conteúdo pode ter seus valores definidos de forma explícita, por exemplo, através da sintaxe de autoria, ou podem ser obtidos através da análise do conteúdo da mídia, para objetos como o áudio e o vídeo. A fim de simplificar o processo de análise da aplicação, considera-se um pré-processamento

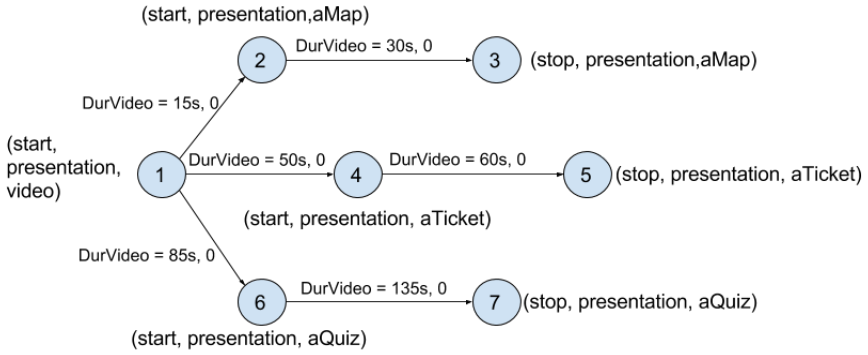

Figura 2.5: HTG para o trecho de aplicação apresentado na Figura 2.4

da aplicação para adicionar, explicitamente no documento NCL, a duração dos objetos de mídia, nos casos em que ela é definida de modo implícito.

No HTG, para cada elemento filho de um switch, devem ser construídos vértices representando as transições dos eventos que podem ocorrer sobre esse switch. Essas possíveis transições são definidas através de elos que contém o objeto switch associado a um de seus papéis.

Um relacionamento que tem como consequência o disparo de uma transição de evento sobre um switch leva à criação de uma aresta no HTG para cada um dos elementos filhos desse switch. E a condição associada a cada uma dessas arestas deve ser a mesma definida nas regras para a escolha de um dos componentes do switch.

A NCL oferece uma facilidade que é a possibilidade de reusar elementos, como os objetos de mídia e os contextos. O reúso em um contexto (elemento **<context>**) faz com que uma cópia do contexto referenciado (valor do atributo refer), incluindo todos os seus elementos filhos, seja realizada.

O reúso em um objeto de mídia (**<media>**) também gera uma cópia do objeto referenciado (indicado no atributo refer). No entanto, o objeto que realiza a referência pode definir novos elementos filhos (**<area>** e **<property>**).

O atributo *instance* do elemento **<media>**, que faz referência a outro objeto, pode ter os valores "*instSame*", "*new*" ou "*gradSame*". Cada um desses valores define uma variação na forma com que os objetos podem ser reusados, gerando arestas e vértices diferentes no HTG.

Quando o atributo instance do objeto que faz a referência tem o valor "*instSame*", o objeto que referencia é o mesmo objeto referenciado, desde a sua instanciação, incluindo todas as suas propriedades e âncoras de conteúdo. Por isso, no HTG, nenhum vértice

para este elemento é criado, e todos os relacionamentos que o envolvem, são definidos a partir do vértice com o objeto referenciado. Só serão adicionados novos vértices se novas âncoras de conteúdo forem especificadas.

Caso o atributo *instance* tenha o valor "*new*", uma cópia do objeto referenciado é realizada. Este valor é o default para o atributo instance, caso esse atributo não seja especificado. Para uma mídia que utilize esse valor, uma cópia dos vértices e das arestas associadas ao objeto referenciado deve ser realizada no HTG que representa a aplicação. Assim, um novo objeto é representado no grafo, e novos vértices e arestas são construídos para representar as transições dos eventos sobre esse novo objeto.

O outro valor possível para o atributo instance é "*gradSame*". Nesse outro caso o objeto que faz a referência e o objeto referenciado são os mesmos, porém, as incorporações no objeto que faz a referência serão realizadas à medida que eles sofrerem ações de start.

No reúso "*gradSame*", as transições de início para o evento de apresentação dos objetos (que faz a referência e o referenciado) podem ocorrer em instantes temporais distintos. Assim, no HTG que representa uma aplicação NCL, devem ser construídos vértices correspondentes ao início e ao fim do evento de apresentação para um objeto que utiliza o reúso "*gradSame*".

Elos (elemento **<link>**) e conectores (elemento **<causalConnector>** são utilizados para definir o sincronismo e, em particular, a interatividade entre os objetos de uma aplicação NCL. Na construção do HTG, elos da aplicação NCL que possuem uma condição e uma ação simples são representados por uma única aresta, unindo os vértices correspondentes à condição e à ação desse relacionamento.

O vértice que representa o início de um evento de seleção, caso esse evento seja necessário em uma aplicação, deve ser o destino de uma aresta com origem no vértice de início do evento de apresentação da âncora de conteúdo sobre a qual essa seleção pode ocorrer.

Um contexto agrupa objetos (de mídia, de contexto ou switch) e elos. O elemento **<body>** de uma aplicação NCL é um caso específico de contexto que representa a aplicação como um todo. Os demais contextos são definidos pelo elemento **<context>**, e podem ser aninhados para refletir uma estrutura do documento e auxiliar na organização dos segmentos da aplicação por parte do autor.

Para cada componente de um contexto, um vértice no HTG deve ser construído para representar as ações do tipo "*stop*" ou "*abort*" executadas diretamente sobre esse con-

texto. No caso das ações do tipo "*start*", devem ser construídos vértices apenas para os componentes mapeados através das suas portas.

O grafo considera tanto eventos determinísticos, quanto não-determinísticos. Informações ligadas ao tempo, como duração de um objeto de mídia, ou elos que contêm os papéis de condição **onBegin**, **onEnd**, **onPause** e **onResume** representam eventos determinísticos.

O elemento **switch**, e elos com papéis de condição **onSelection**, **onBeginAttribution** ou **onEndAttribution** definem o caráter não-determinístico de uma aplicação. Isto porque não é possível prever o momento em que o usuário irá interagir com a aplicação, ou o momento em que uma ação de atribuição inicia ou termina.

Um HTG pode ser dividido em sub-grafos, denominados cadeias temporais. Cada cadeia temporal é composta por vértices que representam transições de evento determinísticos passíveis de ocorrer devido a uma transição de evento inicial não-determinístico. Estas cadeias temporais buscam facilitar o controle da apresentação da aplicação.

A partir do HTG, cinco estruturas de dados podem ser construídas para o controle do sincronismo das apresentações hipermídia. 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) introduz estas estruturas, com especial detalhamento sobre a geração do plano de apresentação. O plano de distribuição introduzido naquele trabalho corresponde ao Plano de Transmissão de Conteúdo Hipermídia do presente trabalho.

A construção do plano de distribuição em (COSTA, 2010) considera que todas as transições de eventos não-determinísticos ocorrerão no menor instante de tempo possível, e as condições relacionadas ao contexto dos usuários, são sempre verdadeiras. Além disso, apenas os instantes temporais relativos às transições de início ("*start*") de um evento de apresentação fazem parte daquele plano. Isto porque são essas transições que indicam que um conteúdo deve estar presente no carrossel.

Atualmente a estrutura tem sido empregada apenas no lado do cliente, para a geração do plano de pré-busca (FIGUEROA, 2014) e para controle da apresentação. Para o desenvolvimento do presente trabalho, o HTG será utilizado no transmissor, de modo a gerar o plano de distribuição incomum em TV Digital. Porém, o plano proposto também considera instantes temporais relativos às transições de fim ("*stop*") de um evento de

apresentação.

Essas transições serão consideradas para definir quando um objeto de mídia deixa de ser necessário no fluxo de dados, e pode ser retirado. Além disso, em alguns cenários de instanciação do framework proposto, as condições relacionadas ao contexto do usuário poderão ter seus valores considerados de fato para a construção do plano de transmissão.

2.3 ENTREGA DE CONTEÚDO EM MODO PUSH

O usuário de um serviço de TV pode sintonizar em um canal a qualquer instante da distribuição de dados, por isso utiliza-se uma estrutura cíclica para enviá-los. Dessa forma, permite-se o recebimento desses dados de forma independente ao instante em que o receptor se junta ao serviço. Esse mecanismo é uma das ferramentas oferecidas pelo DSM-CC (ISO/IEC, 1998), e é denominado carrossel de dados. O DSM-CC (*Digital Storage Media Command and Control*), faz parte do conjunto de especificações MPEG-2, e padroniza um conjunto de protocolos para gerenciamento de fluxos de bits MPEG-2.

O protocolo DSM-CC especifica dois tipos de estruturas cíclicas para envio de dados, o carrossel de dados e o carrossel de objetos. No carrossel de dados, os dados são enviados ciclicamente em módulos, sem indicação a respeito do que eles se tratam. Já o carrossel de objetos é uma estrutura mais robusta, que fornece funcionalidades de um sistema de arquivos, e pode conter um ou mais arquivos e diretórios.

O receptor de um conteúdo transmitido por carrossel, caso não obtenha o dado desejado ao se juntar ao serviço, deve esperar por um novo ciclo do fluxo de dados, para obtê-lo, como mostra a Figura 2.6.

Os arquivos devem ser organizados no carrossel, a fim de obter o menor atraso para seu carregamento. E em alguns casos, arquivos imprescindíveis podem ser repetidos dentro do carrossel, para evitar que o usuário espere por toda sequência até que possa baixá-lo (PESSOA et al., 2008). Porém, essa decisão de construção do carrossel pode prejudicar a entrega de outros arquivos, com menor frequência e também aumenta o tamanho total do carrossel.

A disposição dos elementos em um carrossel pode seguir diferentes regras, dependendo da estrutura do conteúdo que será transmitido. Na transmissão de uma aplicação hipermídia com sincronismo baseado em eventos, a especificação da aplicação é enviada juntamente com os objetos de mídia que a compõem no carrossel DSM-CC (Figura 2.6).

Nesse tipo de aplicação é possível determinar, em tempo de apresentação, a relação entre as mídias e o momento em que são necessárias, com base na análise dos eventos determinísticos. Tais informações podem ser utilizadas para definir o momento em que a mídia deve ser inserida no carrossel, ou retirada, caso não seja mais necessária, otimizando o carrossel. Esta dissertação busca uma maneira de escalonar a transmissão dos objetos de mídia, otimizando a construção do carrossel e levando a um menor uso de recursos na transmissão de aplicações hipermídia em sistemas de comunicação multimídia por difusão (broadcast).

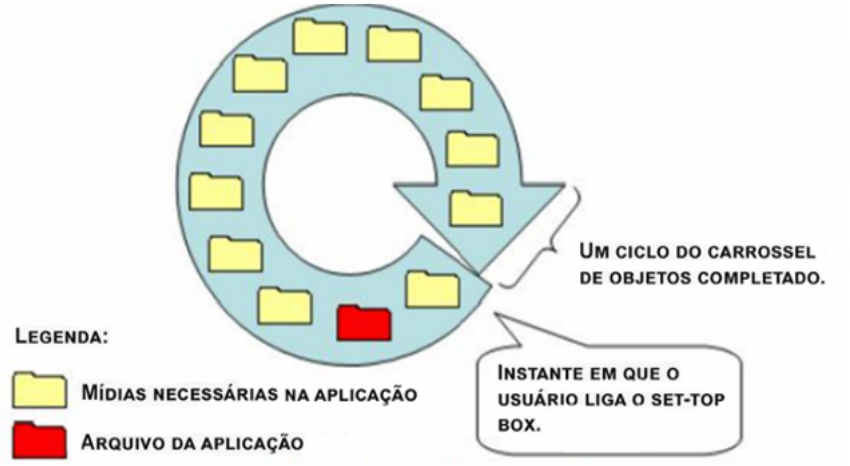

Figura 2.6: Carrossel de objetos contendo objetos de mídia de uma aplicação

FLUTE (PAILA et al., 2012) é um protocolo que também dá suporte à entrega de conteúdo hipermídia em modo push, para IP multicast ou unicast. O protocolo é unidirecional e utiliza estruturas de arquivos para entrega de objetos de mídia. FLUTE é definido com base no protocolo ALC (Asynchronous Layered Coding), que especifica o transporte de objetos binários. Para que os receptores processem o conteúdo recebido é necessário que eles saibam o que os objetos representam. Assim, o FLUTE define um mecanismo para sinalizar e mapear as propriedades de arquivos para os conceitos do ALC, a fim de permitir que os receptores atribuam parâmetros para cada objeto recebido.

Utilizado em diferentes padrões, como DVB-H (*Digital Video Broadcasting - Handheld*) (ETSI, 2009), DVB-IPTV(*DVB - Internet Protocol TV*)(ETSI, 2008) e MBMS (*Multimedia Broadcast Multicast Service*)(ETSI, 2013), o FLUTE utiliza codificação FEC (*Forward Error Correction*) para prover confiabilidade, e controle de congestionamento sem feedback.

Em FLUTE, os arquivos e suas propriedades são enviados em um mesmo canal. As informações sobre os arquivos são fornecidas por uma tabela, denominada FDT (*File Delivery Table*). A FDT permite especificar tanto atributos relacionados ao arquivo (e.g. identificador do arquivo, nome, tipo, tamanho e dados de segurança), quanto atributos ligados ao transporte do arquivo (e.g. identificador do objeto transportado – TOI e tamanho do objeto que contém o arquivo).

A sessão de entrega de arquivo consiste de múltiplos canais agrupados logicamente, com origem em um único emissor, que são usados em períodos de tempo, para o transporte de pacotes pertencentes à transmissão de um ou mais objetos de interesse dos receptores. Toda sessão deve conter uma FDT, onde cada entrada contém um TOI e a URI para o arquivo que descreve. O TOI é incluído em cada pacote de dados, e é a partir desta informação que o receptor determina qual arquivo está sendo enviado em cada pacote ALC. Um receptor se une a um canal para iniciar a recepção de pacotes de dados enviados pelo emissor através do canal, e se desassocia quando deseja interromper a recepção.

Os pacotes enviados em uma sessão contêm informações que identificam a qual sessão ele pertence, visto que um receptor pode receber dados de mais de uma sessão ao mesmo tempo. A identificação de uma sessão é feita pelo par (endereço IP da fonte, TSI), onde TSI é o identificador da sessão de transporte.

Um receptor necessita conhecer os parâmetros associados à sessão para que consiga iniciar a recepção de arquivos. De acordo com (LUBY et al., 2002), os parâmetros necessários para o início da recepção são: endereço IP da fonte, número de canais que compõem a sessão, endereço IP e porta para cada canal na sessão e identificador da sessão (TSI). Com o FLUTE é possível fornecer cinco tipos de sessões de entrega de arquivos:

- Sessão estática de entrega de arquivos: sessão que transporta um conjunto pré-definido de arquivos. A versão do arquivo pode mudar durante o ciclo de vida da sessão, mas apenas uma versão do arquivo é entregue em qualquer instante de tempo.

- Sessão de entrega de conteúdo fixo: tipo especial de sessão estática, onde o conjunto de arquivos e a versão do seu conteúdo não pode alterar durante a sessão.

- Sessão dinâmica de entrega de arquivos: nesta sessão, tanto o conjunto de arquivos que a compõe como os arquivos em si podem sofrer alterações.

- Carrossel estático de entrega de arquivos: é uma sessão sem duração definida, na qual um conjunto fixo de arquivos sem possibilidade de alteração é enviado continuamente.

- Carrossel dinâmico de entrega de arquivos: é uma sessão sem duração definida, que transporta um conjunto de arquivos de maneira contínua. Estes arquivos podem ser alterados, deletados e novos arquivos podem ser inseridos. Em um carrossel dinâmico, o receptor pode detectar a alteração no carrossel através de mudanças no número da instância FDT.

Como o protocolo FLUTE não depende do envio de nenhuma informação do receptor para o servidor, é possível que inúmeros dispositivos recebam concorrentemente um mesmo objeto, sem que haja maior sobrecarga na rede. Além disso, o protocolo permite que cada receptor inicie a recepção de um objeto de modo assíncrono. Essas características tornam o protocolo adequado à entrega de conteúdo multimídia interativo, em sistemas como o IPTV, que suporta na ordem de milhões de clientes, e possibilitam que receptores sintonizem em qualquer canal, a qualquer momento.

Aplicações hipermídia podem ser apresentadas de diferentes maneiras, de acordo com as interações realizadas pelo usuário, e também o contexto do dispositivo em que é executada. Por exemplo, caso um usuário opte por não interagir com a aplicação, todos os conteúdos relacionados a interação poderiam ser descartados pelo receptor. Desta forma, em dois dos cenários de instanciação do framework proposto neste trabalho será utilizado o mecanismo de envio de objetos em múltiplos canais, fornecido pelo FLUTE. Isto será especialmente útil para o envio de conjuntos de mídias relacionados a eventos não-determinísticos definidos pela aplicação, possibilitando que um receptor se junte a canais multicast que contém apenas objetos do interesse do usuário.

3 TRABALHOS RELACIONADOS

A qualidade da apresentação de um conteúdo hipermídia demanda o controle de diferentes recursos, para manter o sincronismo das aplicações. Estes recursos se relacionam não somente na fase de apresentação (instanciação de exibidores e pré-busca), mas também na fase de transmissão do conteúdo, conforme discutido anteriormente. Este capítulo aborda alguns mecanismos para gerenciar a transmissão de conteúdo multimídia/hipermídia, desempenhados pelo emisssor ou receptor.

3.1 GERENCIAMENTO DE TRANSMISSÃO DE APLICAÇÕES MULTIMÍDIA

Mahajan e Parashar (2003) propõe um controlador de largura de banda ciente do conteúdo que gerencia a qualidade de serviço para aplicações multimídia, em ambientes de serviços diferenciados (DiffServ). O controlador utiliza políticas baseadas na natureza e adaptabilidade do fluxo, e a parâmetros de rede como *delay* e *jitter*.

Em ambientes de serviços diferenciados, o tráfego de rede é classificado e os recursos são alocados para os fluxos com base em políticas de gerenciamento. As classes de serviço são criadas com diferentes garantias de QoS e os fluxos são atribuídos a estas classes. O controlador possibilita que a aplicação reduza os parâmetros de QoS, caso sua demanda exceda a disponibilidade de recursos.

O controlador se baseia na premissa de que aplicações multimídia são flexíveis, ou seja, que elas podem lidar com a degradação de um recurso, reduzindo seu desempenho, e também podem empregar recursos adicionais, para melhorar sua qualidade. Além disso, essas podem sacrificar a qualidade de alguns parâmetros, para favorecer parâmetros mais críticos.

As aplicações multimídia são enviadas através de fluxos, que são classificados com um índice de flexibilidade, baseado na sua tolerância às variações dos parâmetros de rede.

Quando uma aplicação solicita recursos de rede ao controlador, ela informa seus requisitos de QoS específicos e restrições. Se os recursos solicitados não puderem ser fornecidos, o controlador utiliza o número de flexibilidade do fluxo para minimizar os recursos solicitados. Se mesmo após a alteração dos requisitos da aplicação, os recursos ainda forem

escassos, o fluxo é rejeitado.

Ott e Mayer-Patel (2007) propõe uma arquitetura que dá suporte à troca de informações entre os vários fluxos pertencentes a uma mesma aplicação, como informação de estado da rede e informações específicas da aplicação. Para isto é definido um Protocolo de Coordenação (PC).

O protocolo é definido com o objetivo de tratar de maneira similar todos os fluxos que pertencem a uma mesma aplicação. Desse modo, o agregado de fluxos de uma aplicação terá uma visão uniforme das condições da rede, e consequentemente responderão a alterações igualmente. Esta coordenação também permite que fluxos de uma aplicação troquem informações para sincronizar o envio dos dados semanticamente relacionados, definidos na especificação da aplicação. A coordenação é realizada através de um cabeçalho. Este cabeçalho é inserido pela aplicação, em cada pacote, e localizado entre o cabeçalho da camada de rede e o da camada de transporte.

Em Fez et al. (2015), a entrega do conteúdo também é adaptável ao receptor, porém, esta adaptação é dada em relação ao contexto e ao perfil do usuário do dispositivo cliente. O trabalho apresenta uma arquitetura para entrega escalável de conteúdo multimídia personalizado em redes multicast sem fio.

O processo de adaptação proposto em (FEZ et al., 2015) ocorre em duas etapas, sendo uma no lado servidor e outra no cliente. O servidor gerencia e fornece os diferentes conteúdos e os metadados, utilizados para determinar o quanto um conteúdo é apropriado para o contexto de um usuário. A pré-filtragem realizada no servidor faz com que apenas arquivos relevantes sejam enviados aos usuários, sem sobrecarregar os canais de transmissão.

O cliente realiza uma segunda filtragem para apresentar ao usuário apenas os conteúdos de seu interesse. Isto é feito no cliente, pois o perfil do mesmo encontra-se armazenado no dispositivo do usuário.

3.2 MECANISMOS DE PRÉ-BUSCA

A qualidade da apresentação de aplicações hipermídia experimentada por usuários deste tipo de conteúdo é medida através de certos parâmetros, sendo um deles o tempo de espera na exibição de uma mídia. Além de degradar a qualidade da apresentação, atrasos na entrega do conteúdo também podem prejudicar a interatividade do usuário e a sincronização entre objetos de mídia. Algumas máquinas de apresentação de conteúdo hipermídia

utilizam mecanismos de pré-busca para lidar com este tipo de problema. A pré-busca consiste em tentar prever qual o próximo objeto do conteúdo a ser requisitado ao emissor, para armazená-lo antes em cache. As técnicas de pré-busca devem se antecipar às ações do sistema e são normalmente aplicadas em transferências em modo pull.

Rodrigues e Soares (2002) propõe uma arquitetura genérica adaptável para a especificação de formatadores, com o objetivo de facilitar a integração de mecanismos de pré-busca com outros componentes do sistema de apresentação e com formatadores já existentes, como QuickTime e RealOne Player. Formatadores são componentes de um sistema hipermídia responsáveis por adaptar e controlar a apresentação de documentos.

De acordo com Rodrigues e Soares (2002), agregar mecanismo de pré-busca a formatadores aumenta a probabilidade do conteúdo de mídia estar disponível nas ferramentas de apresentação, no momento em que deve ser exibido. Para isso, é preciso que um gerenciador de pré-busca calcule a duração esperada para obtenção da mídia, para cada objeto de mídia do documento. O instante para o gerenciador iniciar suas tarefas é definido pelo valor obtido neste processamento, junto com o tempo esperado de apresentação do objeto de mídia.

A partir do plano de pré-busca, um escalonador deve ser instanciado para informar aos componentes do sistema, o momento em que devem iniciar. O gerenciamento de pré-busca é dividido em duas fases: compilação e execução. A etapa de compilação tem como resultado o plano de pré-busca, e a etapa de execução, é responsável pelo escalonamento das ações do plano e monitoramento. A etapa de execução pode disparar ajustes na apresentação, caso ocorram erros.

Figueroa (2014) desenvolveu um plano de pré-busca com base em aplicações onde o sincronismo depende da ocorrência de eventos com duração variável, ou mesmo não-determinísticos no momento da autoria. A proposta define um plano com os tempos de início para recuperação antecipada das mídias, e determina como calcular o tamanho máximo necessário do buffer. O gerenciamento do buffer é importante para o caso de plataformas computacionais com recursos limitados.

Em sua proposta, Figueroa (2014) considera os relacionamentos causais entre os objetos de mídia para definir a ordem de recuperação dos conteúdos dos objetos de mídia, além dos intervalos absolutos de um eixo temporal. Os objetos de mídia que se relacionam entre si são agrupados, para facilitar a definição da prioridade de recuperação dos objetos.

O plano de pré-busca gerado por Figueroa (2014) considera a vazão do canal de comunicação estabelecido entre o servidor e o cliente, o espaço disponível no buffer para armazenar os conteúdos de mídia, o tamanho dos objetos a serem recuperados e o tempo máximo tolerável para a recuperação de um determinado objeto de mídia. O plano proposto é dinâmico, pois considera as variações que a largura de banda da rede e o tempo de acesso podem sofrer.

É importante salientar que o tempo necessário para a entrega de conteúdos de mídia é uma questão muito importante a ser considerada durante a construção do plano de pré-busca, assim como do plano de transmissão. Devem-se considerar os parâmetros de retardo de transmissão e tempo de acesso ao sistema de armazenamento do servidor, não somente no modo pull, mas também no modo push.

Eventos de seleção estão ligados à interatividade da apresentação hipermídia. Esses eventos se ligam a uma ou mais cadeias temporais secundárias derivadas do grafo hipermídia. Como as apresentações hipermídia são, por natureza, não-determinísticas, é impossível saber, a priori, se o usuário irá ou não interagir com a aplicação. Por isso, o plano de pré-busca considera que o evento de seleção sempre irá ocorrer, no momento em que se torna disponível. Caso a interação não ocorra, o plano de pré-busca deve ser recalculado.

A utilização de mecanismos de pré-busca é uma opção interessante que auxilia na redução do atraso na entrega do conteúdo hipermídia. Porém, esta abordagem pode despediçar recursos, em casos de aplicações que permitem a interação do usuário, e a escolha entre diferentes alternativas, ou seja, aplicações com eventos não-determinísticos.

Como é impossível saber antecipadamente a escolha do usuário, todas as cadeias temporais do HTG relativas às alternativas devem ser antecipadamente solicitadas pelo receptor, até que o usuário escolha uma opção. Nesse caso, devem ser realizadas reformulações do plano de pré-busca de acordo com a ocorrência dos eventos não-determinísticos no lado cliente.

Diferente dos mecanismos propostos por Figueroa (2014) e Rodrigues e Soares (2002), a solução apresentada por este trabalho busca gerenciar a entrega em modo push, realizando o controle da transmissão da aplicação no lado servidor, que é responsável por analisar o comportamento da aplicação para fins de otimização na utilização dos recursos.

3.3 COMPARAÇÃO DAS PROPOSTAS

Atualmente, os trabalhos relacionados que tem como objetivo gerenciar a entrega de conteúdo hipermídia no lado do servidor, desempenham tal tarefa sem considerar o comportamento temporal da aplicação. Essa técnica tem sido empregada apenas no lado do receptor, através da utilização de mecanismos de pré-busca. A Tabela 3.1 apresenta um resumo desses trabalhos, a fim de compará-los com a solução proposta nesta dissertação.

A comparação foi realizada com base em alguns parâmetros, como o local onde o gerenciamento da entrega ocorre (servidor ou cliente), se os trabalhos utilizam técnicas de análise da aplicação, se permitem adaptação da aplicação às condições da rede, adaptação da entrega ao usuário e o modo de entrega do conteúdo.

Tabela 3.1: Tabela comparativa dos Trabalhos Relacionados

Trabalho Relacionado	Local	Análise	Adaptável à rede	Adaptável ao usuário	Modo
Mahajan e Parashar (2003)	servidor	não	sim	não	push
Ott e Mayer-Patel (2007)	servidor	não	não	não	push
Fez et al. (2015)	servidor e cliente	não	não	sim	push
Rodrigues e Soares (2002)	cliente	sim	não	não	pull
Figueroa (2014)	cliente	sim	não	não	pull

O trabalho de (MAHAJAN; PARASHAR, 2003) propõe um mecanismo interessante, que possibilita à adaptação da aplicação às condições da rede, ao evitar a interrupção na transmissão em situações de congestionamento e permitir a redução de parâmetros de qualidade por parte da aplicação. Porém, a classificação de um fluxo, quanto à sua flexibilidade, não é feita de maneira automatizada, e os fluxos de mídias que compõem uma aplicação são tratados de modo independente, sem considerar as relações que a aplicação define sobre eles.

Já a proposta de Ott e Mayer-Patel (2007) permite relacionar os fluxos de acordo com a semântica definida na aplicação, mas não oferece nenhum meio para especificar e gerenciar os relacionamentos temporais entre os dados dos diferentes fluxos de mídia.

A arquitetura proposta por (FEZ et al., 2015) também tem o objetivo de adaptar a entrega ao usuário, porém ela não considera as relações de sincronismo definidas pela aplicação. Assim, mesmo que o conteúdo mais adequado seja entregue ao usuário, não existe nenhum mecanismo para prover QoE, conforme as relações de sincronismo definidas pelo autor, no momento da apresentação da aplicação.

Nesta dissertação, o gerenciamento da transmissão de aplicações hipermídia é realizado pelo servidor, para a entrega de conteúdo em modo push. Para isso, a proposta emprega a análise do comportamento da aplicação, possibilitando que ela se adapte tanto às condições da rede, quanto ao usuário que irá consumir o conteúdo.

4 FRAMEWORK DE TRANSMISSÃO DE APLICAÇÕES HIPERMÍDIA EM MODO PUSH

Este capítulo apresenta o framework proposto para gerenciamento de transmissão adaptável a diferentes técnicas e tecnologias de redes de comunicação multimídia. O framework define estruturas de dados, componentes gerenciadores e suas estratégias para a construção e aplicação do Plano de Transmissão de Conteúdo Hipermídia. A Seção 4.1 apresenta as estruturas de dados auxiliares definidas para modelar o comportamento temporal das aplicações hipermídia e especificar os instantes de apresentação dos objetos de mídia. Os componentes de análise e gerenciamento de transmissão, de entrega e de recepção são descritos na Seção 4.2.

Neste trabalho foi adotada uma abordagem de especificação orientada a classes e objetos para facilitar o entendimento de toda a estruturação do framework. Essa estruturação é genérica e deve ser especializada para cada cenário de provisão de serviços hipermídia.

4.1 ESTRUTURAS DE DADOS AUXILIARES

O comportamento temporal das aplicações hipermídia é modelado no framework através da estrutura de dados descrita anteriormente, o HTG.

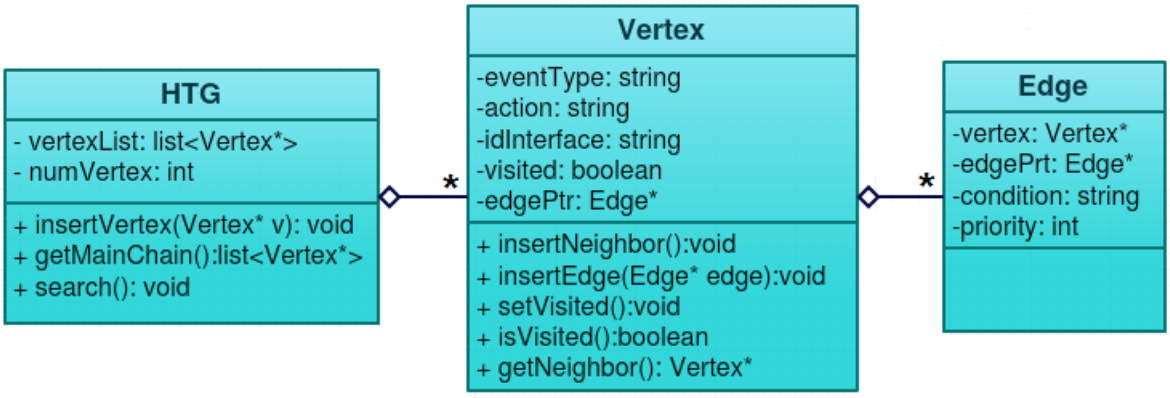

Figura 4.1: Definição do grafo temporal hipermídia

A classe HTG (Figura 4.1) contém a definição da estrutura do grafo, especificado nesta implementação através de uma lista de adjacências. As arestas contêm dois atributos, um com a condição de caminhamento, e outro com sua prioridade, em relação às outras

arestas que possuem o vértice fonte em comum. Operações de inserção e remoção de vértices podem ser realizadas sobre o grafo, além de caminhamento no grafo, a fim de obter as cadeias que o compõem.

Na construção do grafo, apenas as transições de eventos que provocam alterações na apresentação, serão representadas, para preservar a simplicidade do mesmo.

Ao percorrer o grafo, tendo como início o vértice que contém a mídia especificada como porta de entrada da aplicação, é possível construir uma versão incial do Plano de Transmissão (**TransmissionPlan**) que contém o comportamento da aplicação. O Plano de Transmissão é inicialmente preenchido com uma lista de mídias, com informações do instante em que cada uma é necessária (startRequiredTime) na aplicação, e o tempo em que deixam de ser utilizadas (endRequiredTime). A Figura 4.2 apresenta a estrutura do Plano de Transmissão.

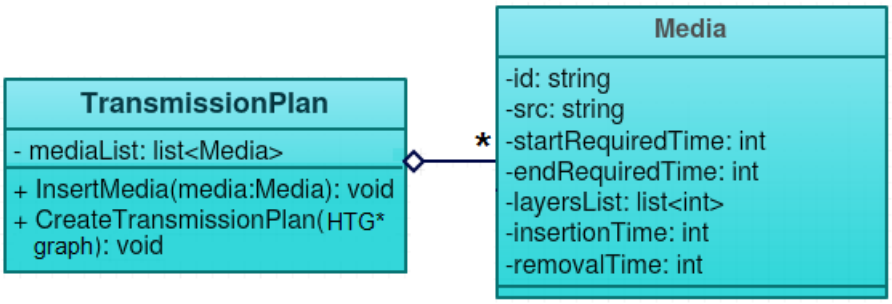

Figura 4.2: Definição do plano de transmissão

O algoritmo de preenchimento inicial do Plano de Transmissão (método *CreateTransmissionPlan*) percorre o grafo em profundidade, e para as arestas que possuem o tempo como condição de caminhamento, o instante de tempo é adicionado a uma variável que guarda o tempo de exibição da aplicação. O valor dessa variável é atribuído à informação de início de mídia, quando um vértice de start do objeto de mídia é encontrado, e à informação de fim da mídia, ao encontrar um vértice de stop da mesma.

Nas cadeias temporais do HTG, os eventos previsíveis não-determinísticos também ganham um tempo estimado de ocorrência. Esse tempo é aquele que será verdadeiro, caso a cadeia seja percorrida sem interrupção até o fim.

4.2 ELEMENTOS DO FRAMEWORK PARA GERENCIAMENTO DA TRANSMISSÃO

O gerenciamento de transmissão adaptável é realizado pelos componentes **ApplicationAnalyzer**, **TransmissionManager**, **TransmissionStrategy**, **DeliveryManager** e **DataSender**, instanciados no lado servidor. A Figura 4.3 apresenta a definição destes componentes, que são descritos a seguir.

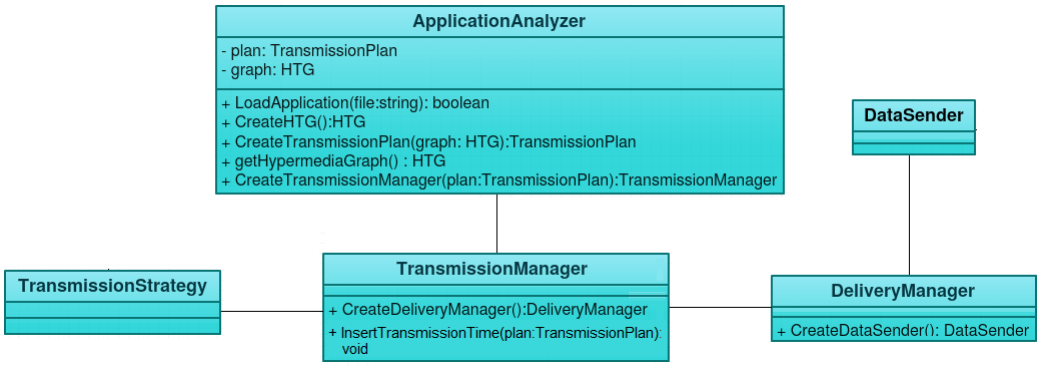

Figura 4.3: Elementos responsáveis pelo gerenciamento da transmissão

A classe **ApplicationAnalyzer** foi desenvolvida para carregar a especificação da aplicação (método *LoadApplication*) e extrair suas características, como o tempo de início e fim de cada objeto de mídia, e as relações temporais entre eles.

A análise de aplicações definidas por linguagem baseadas em eventos, como a linguagem NCL, possibilita a extração de informações sobre o comportamento de sua apresentação no dispositivo receptor. Estas informações também possibilitam escalonar a transmissão dos objetos de mídia, e com isso, otimizar a entrega do conteúdo interativo. Como resultado da análise da aplicação, especificada em **ApplicationAnalyzer**, obtém-se o HTG e o pré-plano de transmissão da aplicação.

O pré-plano representado por **TransmissionPlan**, inicialmente preenchido pelo método *CreateTransmissionPlan* da classe **ApplicationAnalyzer**, é complementado com base no cálculo de tempo de acesso às mídias do fluxo de dados, e nas estratégias de transmissão definidas em **TransmissionStrategy**. A classe **TransmissionManager** contém o método responsável por criar o Plano de Transmissão final, que irá coordenar a entrega do conteúdo hipermídia. A partir desta estrutura é possível visualizar as mídias necessárias em um intervalo de tempo, para que o conteúdo seja enviado no tempo correto de sua exibição nos clientes, sem que haja perda de conteúdo ou sincronização.

Como o framework dá suporte a diferentes cenários de provisão de serviços hipermídia, diferentes estratégias de transmissão podem ser empregadas, através da especialização do componente **TransmissionStrategy**.

O emissor de dados implementado pela classe **DataSender** comunica-se com o gerenciador de entrega (**DeliveryManager**), e implementa o protocolo de entrega de conteúdo. O gerenciador de entrega é responsável por manipular a estrutura utilizada pelo protocolo de entrega para envio dos dados, através da inserção e remoção de objetos de mídia.

4.3 ELEMENTOS DO FRAMEWORK PARA RECEPÇÃO DO CONTEÚDO

No lado cliente, o componente **DataReceiver** estabelece a conexão com o emissor de dados, para dar início ao recebimento de dados. É nesta classe que a comunicação com o servidor é gerenciada, de acordo com o protocolo de entrega de conteúdo.

ApplicationMiddleware representa o componente que controla a apresentação da aplicação multimídia, coordenando o carregamento de players de mídia, as interações do usuário com o conteúdo, entre outras atividades.

Um componente, definido pela classe **ReceptionManager**, foi acoplado ao middleware para possibilitar sua integração aos diferentes sistemas hipermídia, com características de entrega de conteúdo distintas. Este realiza um pré-processamento da aplicação recebida, para obter informações como o local de onde buscar os objetos de mídia, podendo, inclusive, alterar o documento recebido para maior adequação ao cenário de provisão do serviço. Esses componentes são exibidos na Figura 4.4, e o modo como se relacionam pode ser visto na Figura 4.5.

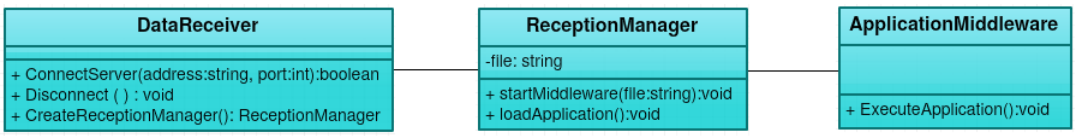

Figura 4.4: Elementos responsáveis pela recepção do conteúdo

O framework tem o propósito de ser genérico para se adaptar a diferentes plataformas de envio de conteúdo hipermídia por push e estratégias de envio, através da instanciação de certos pontos de flexibilização. Tal característica facilita o reúso do framework, que pode ser instanciado pelo preenchimento desses pontos, como apresentado no Capítulo 5.

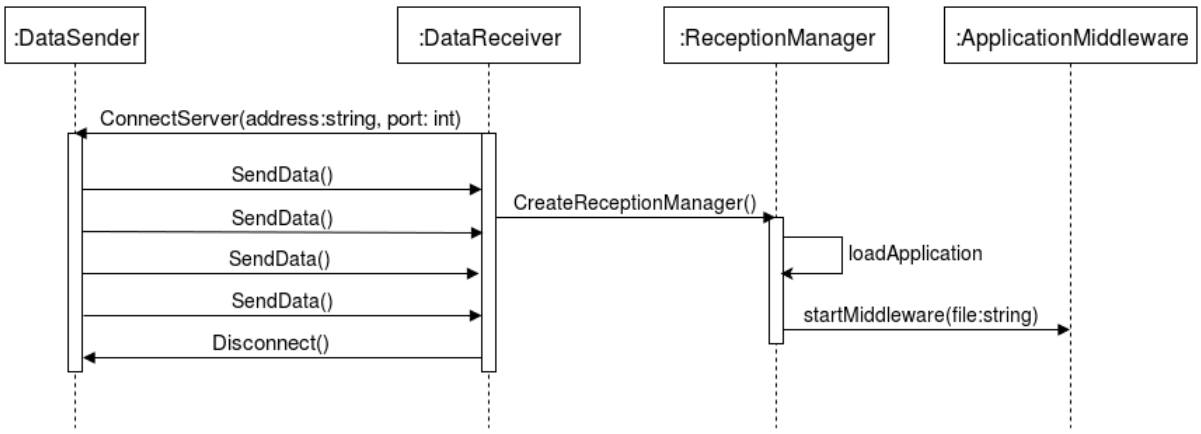

Figura 4.5: Diagrama de sequência do processo de recepção do conteúdo

5 CENÁRIOS DE INSTANCIAÇÃO DO FRAMEWORK

Como mencionado anteriormente, a transmissão de conteúdo hipermídia pode se dar em diferentes meios, seja por radiodifusão, redes gerenciadas, ou até mesmo através da Internet. Este capítulo descreve alguns cenários de instanciação do framework genérico e métodos de provisão de serviços hipermídia, a fim de demonstrar as características da solução proposta e avaliá-la.

Em aplicações hipermídia, é usual a existência de relacionamentos temporais entre os objetos de mídia que a compõem (eventos determinísticos). Além disso, elas permitem ao usuário, em certos instantes e para certos elementos, controlar como se dará a apresentação do conteúdo. Essa interação pode ser através da seleção de um objeto de mídia, por exemplo, ou através de funções para controlar a reprodução do conteúdo (eventos não-determinísticos).

Uma aplicação hipermídia linear tem sua execução em uma única linha do tempo possível, de duração pré-determinada, e com fim programado. Em geral, essas aplicações são não-interativas, como em programas regulares de TV aberta, mas também podem oferecer alguma interatividade. Para isso, é preciso que o autor do conteúdo restrinja as interações possíveis ao usuário, limitando-o à linha do tempo original, conduzindo-o a uma linearidade de raciocínio.

O conteúdo em que o usuário pode influenciar na apresentação, percorrendo uma linha de raciocínio específica é dito não-linear. Nestes casos, a reprodução do conteúdo pode seguir diferentes caminhos e durações de exibição, para diferentes usuários, ou para um mesmo usuário, em outros momentos de exibição do conteúdo.

Os diversos caminhos oferecidos ao usuário em uma aplicação não-linear podem estar relacionados a questões de adaptação do conteúdo a condições do ambiente de apresentação, ou ao perfil do usuário. Podem, ainda, estar relacionados a decisões do usuário de interagir ou não com o conteúdo exibido. Desse modo, dependendo do caminho que o usuário seguir, ele irá consumir um conjunto específico de mídias.

Por exemplo, considerando um filme que fornece um botão que permite ao usuário visualizar mais informações a respeito de um personagem, quando este aparece em cena.

Se o usuário selecionar o botão, novas mídias devem ser enviadas para o receptor, como um texto descritivo sobre o personagem e a imagem do mesmo. Caso contrário, estas mídias serão desnecessárias.

Considerando, então, os diferentes tipos de eventos determinísticos e não-determinísticos que caracterizam o conteúdo hipermídia e, ainda, considerando os diferentes métodos de transmissão, vislumbra-se possíveis instanciações do framework em diferentes cenários. Em um primeiro cenário, a transmissão do conteúdo se dá por meio em um único canal, em modo push, sem qualquer otimização, como ocorre atualmente nas transmissões de TV Digital terrestre (broadcast) e IPTV (multicast). No entanto, dadas as possibilidades decorrentes da análise do comportamento da aplicação, três novos cenários de instanciação são propostos neste trabalho: transmissão em uma única camada (concretizada por meio de um canal push), transmissão sobre múltiplas camadas independentes e transmissão sobre múltiplas camadas complementares. Para cada um desses cenários e cenários futuros, é esperada a definição de uma estratégia de transmissão específica (especialização de **TransmissionStrategy**). Assim, neste trabalho o framework foi instanciado para os seguintes cenários:

1. *Basic*: Os dados que compõem o conteúdo hipermídia são enviados através de um único canal, sem empregar nenhuma análise dos eventos definidos na aplicação para gerenciamento da transmissão. Neste cenário, enquadram-se transmissões em TV Digital terrestre (broadcast) e IPTV (multicast) como são realizadas atualmente.

2. *SingleLayer*: Os dados são enviados utilizando uma única camada, porém a análise da aplicação permite considerar os eventos determinísticos definidos pelo autor para o gerenciamento da transmissão.

3. *IndependentLayers*: Os dados são enviados utilizando múltiplas camadas, cujos conteúdos são independentes entre si, permitindo considerar também os eventos não-determinísticos.

4. *ComplementaryLayers*: Os dados são enviados utilizando múltiplas camadas, cujos conteúdos são complementares entre si, evitando redundâncias de objetos de mídia nas diferentes camadas.

Os Cenários *IndependentLayers* e *ComplementaryLayers* dependem da concretização de múltiplas camadas e, portanto, são próprios de métodos de transmissão que permitem

a criação de múltiplos canais. Assim, não são factíveis em transmissões broadcast de canal único, como a radiodifusão.

Conforme mencionado no capítulo anterior, um HTG é criado na análise da aplicação. Os vértices que podem ser atingidos pelo ponto de entrada da aplicação, a partir de arestas que têm apenas o tempo como condição associada, formam a cadeia principal. Uma cadeia secundária é construída quando uma aresta que possui como condição associada, ações externas ou variáveis não associadas ao tempo, é encontrada. Ou então, quando o vértice alcançado representa uma transição de evento imprevisível para o estado do seu evento.

No cenário de transmissão de múltiplas camadas independentes (*IndependentLayers*) cada camada será composta por objetos de mídia relacionados a uma opção de caminho a ser seguida ao longo da apresentação. Para cada cadeia secundária do HTG, deve-se criar uma nova camada de envio contendo as novas mídias contidas nos vértices que a compõem, além das mídias da cadeia principal do HTG.

Ao enviar conteúdo hipermídia em um cenário de múltiplas camadas complementares (*ComplementaryLayers*), um caminho de reprodução do conteúdo é dado pela junção de uma ou mais camadas contendo objetos de mídia. Nesse caso, busca-se minimizar a repetição de objetos entre as camadas. Para este cenário, os objetos de mídia definidos na cadeia principal do HTG são colocadas em uma camada base, e o conjunto de novos objetos de mídia relacionado a cada cadeia secundária do HTG, é disposto em novas camadas de envio.

5.1 INSTACIAÇÃO DE ESTRATÉGIAS DE TRANSMISSÃO

O gerenciador da transmissão (**TransmissionManager**) pode empregar diferentes estratégias para construção do plano de transmissão. Nesta discussão sobre a instanciação do framework, são consideradas quatro diferentes estratégias para construção do plano de transmissão: (i) básica (**TS_Basic**), (ii) envio em uma única camada considerando eventos determinísticos (**TS_SingleLayer**), (iii) envio em múltiplas camadas independentes (**TS_IndependentLayers**) e (iv) envio em múltiplas camadas complementares (**TS_-ComplementaryLayers**). Atualmente, apenas a estratégia (i) é utilizada para envio de conteúdo hipermídia. As estratégias (ii), (iii) e (iv) apresentam técnicas novas para a transmissão do conteúdo, e são propostas por este trabalho.

Nas estratégias de múltiplas camadas (**TS_MultipleLayers**), são analisados eventos

determinísticos e não-determinísticos e podem ser baseadas em camadas complementares (**TS_ComplementaryLayers**) ou independentes (**TS_IndependentLayers**). A Figura 5.1 ilustra tal especialização das estratégias junto ao framework. Nota-se que algumas estratégias podem ser aplicadas tanto para o método de transmissão por broadcast, quanto por multicast (especificamente, **TS_Basic** e **TS_SingleLayer**).

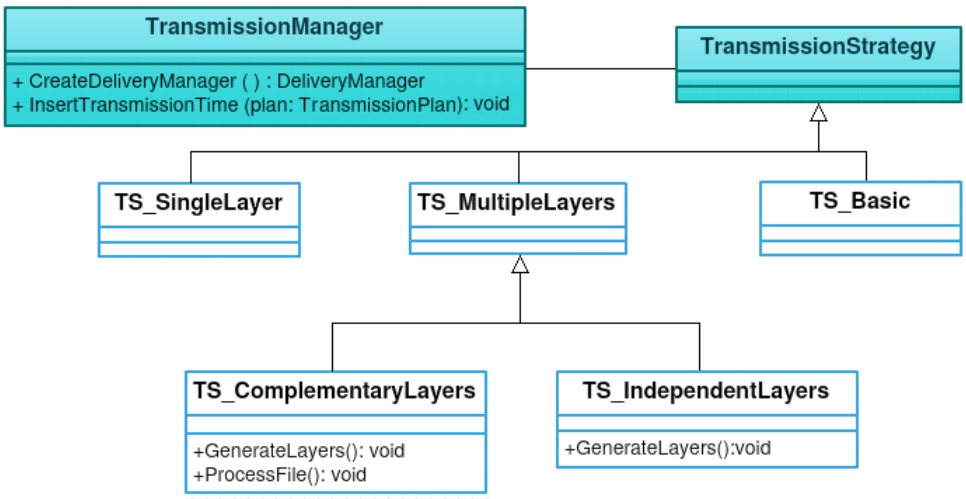

Figura 5.1: Instanciação do componente TransmissionStrategy

A estratégia **TS_Basic**, determina o envio de todos os objetos de mídia ao longo de toda a transmissão da aplicação, sem melhoria alguma no consumo de banda. Para isso, o Plano de Transmissão é configurado de modo que o tempo de envio de todas as mídias é "0" e o tempo de exclusão é o mesmo do fim da transmissão da aplicação. Na estratégia **TS_SingleLayer** a transmissão é otimizada através da análise de eventos determinísticos, e o plano é preenchido com os instantes de inserção e remoção de mídia do fluxo, calculados com base no momento em que são necessárias na apresentação do conteúdo, e o tempo necessário para transferí-la.

No método de envio por broadcast com um único canal, não é possível fazer nenhuma diferenciação entre os caminhos alternativos (não-determinismo) que uma aplicação pode seguir, uma vez que todos os objetos de mídia que os compõem devem ser transmitidos nesse mesmo canal. Por isso, apenas as estratégias (i) e (ii) serão aplicadas neste método. Entretanto, no envio por multicast utilizando múltiplos canais é possível utilizar técnicas de gerenciamento com envio em múltiplas camadas que consideram eventos não-determinísticos e determinísticos (**TS_ IndependentLayers** e **TS_ComplementaryLayers**).

A distribuição das mídias entre as camadas de conteúdo é dada pelo caminhamento

no HTG, considerando as cadeias secundárias do grafo. No caso de envio em camadas independentes (**TS_IndependentLayers**), cada camada é composta por objetos de mídia relacionados a uma opção de caminho a ser seguida ao longo da apresentação. Desse modo, cada caminho é associado a uma camada, que é enviada em um grupo multicast. A associação entre o grupo multicast e a camada de envio é feita pelo gerenciador da transmissão, e enviada ao receptor, para que este possa definir a qual grupo se conectar, conforme o comportamento de execução da aplicação.

No envio por camadas complementares (**TS_ComplementaryLayers**), um caminho de reprodução da aplicação é dado pela junção de uma ou mais camadas com objetos de mídia. Nesse caso, é possível estabelecer uma relação entre mídia e grupo multicast de envio, de modo que a localização de cada mídia na especificação da aplicação deve ser alterado para apontar para um endereço específico (*ProcessFile()*). Para este cenário, as mídias definidas na cadeia principal do HTG são colocadas em uma camada comum, e o conjunto de novas mídias relacionado à cada cadeia secundária é disposto em novas camadas de conteúdo.

5.2 INSTANCIAÇÃO DO FRAMEWORK PARA ENVIO POR CARROSSEL FLUTE

Nos cenários de instanciação do framework o protocolo FLUTE pode ser utilizado para envio dos objetos de mídia que compõem uma aplicação hipermídia. Esta foi inclusive a escolha para a obtenção dos resultados apresentados na próxima Seção, utilizando um carrossel dinâmico que permite diferentes versões de conjuntos de conteúdos a serem transmitidos. Dessa forma, a classe **DeliveryManager** deve ser especializada em **CarouselManager**, conforme ilustrado na Figura 5.2.

A classe **CarouselManager** especifica métodos para criação de uma ou mais versões de carrossel para entrega, de acordo com a estratégia de construção (**CarouselStrategy**) utilizada. Os objetos de mídias que são enviados no carrossel são descritos por uma ou mais tabelas de descrição de arquivo definidas pelo protocolo FLUTE.

Na estratégia **CS_Basic** apenas um carrossel composto por todos os objetos de mídia da aplicação interativa é criado. Mesmo que um objeto deixe de ser necessário na aplicação, ele continua sendo enviado, levando ao desperdício de banda, e podendo causar o

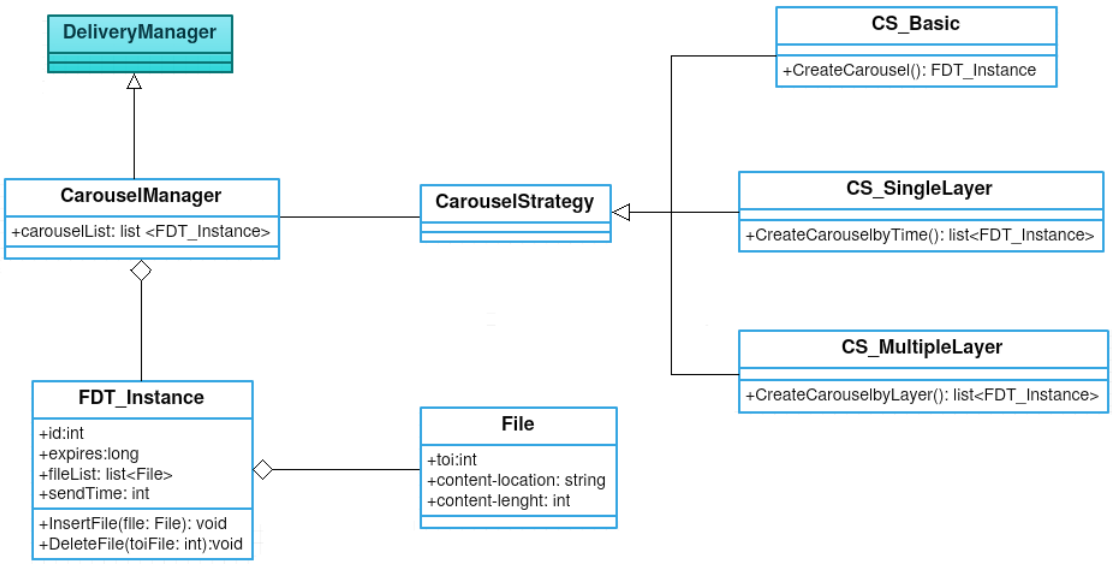

Figura 5.2: Instanciação do componente DeliveryManager

atraso na entrega de outras mídias. Tal mecanismo é o empregado atualmente no sistema brasileiro de TV digital terrestre, e representado pelo Cenário *Basic*.

O fluxo de transporte, distribuído por radiodifusão, pode ser capturado e repassado a usuários de serviços IPTV por multicast. Desse modo, a quantidade média de dados enviados, e o tamanho do carrossel ao longo do tempo, apresentam os mesmos valores tanto para entrega por radiodifusão, quanto por multicast. Assim, será analisado apenas um destes ambientes de entrega para a avaliação do framework proposto, tanto para o cenário atual, quanto para o cenário com otimização baseada em eventos determinísticos.

Ao aplicar a estratégia de transmissão em uma única camada considerando apenas eventos determinísticos (**TS_SingleLayer**) no envio por broadcast ou múlticast com um único canal, o gerenciamento do carrossel é definido pela classe **CS_SingleLayer**. Nessa classe mais de uma versão do carrossel pode ser criada, de acordo com os instantes de tempo definidos pelo Plano de Transmissão.

O Plano de Transmissão considera que todas as transições de eventos imprevisíveis irão ocorrer no menor instante no tempo em que suas ocorrências são possíveis. Para as mídias cujo fim de sua exibição está relacionado a eventos não-determinísticos, seu tempo de fim é dado pelo momento mais tardio em que o evento pode ocorrer. Além disso, todas as condições que dependem das características do ambiente de apresentação ou dos usuários também são consideradas como verdadeiras.

A partir das informações do plano, torna-se possível visualizar em um instante ou

intervalo de tempo quais mídias serão necessárias para que o conteúdo interativo seja apresentado ao espectador sem que haja perdas de dados. Isto porque o servidor de conteúdo pode estimar quais objetos devem ser colocados no carrossel, bem como determinar quais objetos podem ser removidos do carrossel.

Dessa forma, a transmissão dos objetos em si pode ser escalonada conforme a real necessidade de apresentação nos receptores. Obtém-se, portanto, uma redução no uso de banda passante e menor atraso na transmissão de aplicações interativas em sistemas de TV Digital por difusão.

A classe **CS_MultipleLayers** representa a estratégia para criação de carrosséis em cenários que empregam entrega de conteúdo em multicamadas (*IndependentLayers* e *ComplementaryLayers*). Como o conteúdo é enviado em mais de uma camada, mais de um carrossel é criado, podendo existir uma ou mais versões de cada um.

Na transmissão com camadas independentes, cada camada é enviada em um grupo multicast. O número de grupos multicast a serem criados deve ser igual ao número de caminhamentos possíveis no HTG que representa a aplicação. Já na transmissão utilizando camadas complementares, o documento que especifica a aplicação é enviado na íntegra em uma camada, denominada base, juntamente com as mídias relacionadas à cadeia principal do HTG. A segmentação do documento não foi empregada, visto que a dimensão deste conteúdo é irrelevante comparado aos outros objetos de mídia enviados no carrossel. E por isso, não altera o desempenho da entrega do conteúdo multimídia.

Apesar de não empregar a segmentação da aplicação, este trabalho levanta algumas possibilidades para fazê-lo nos casos em que a aplicação é especificada através da linguagem NCL. Esses métodos de segmentação do arquivo NCL e suas características serão descritos na Seção 5.4, e podem ser desenvolvidos em trabalhos futuros.

Além da geração das tabelas de informações, o servidor de aplicações multimídia calcula os instantes para enviar os conteúdos dos objetos de mídia. Na implementação proposta, diferentes versões do carrossl são criadas, de acordo com os intervalos de tempo em que a aplicação se mantém inalterada em relação ao conjunto de mídias em exibição. Para calcular o instante de envio de um novo carrossel, o servidor analisa o tempo necessário para transmistir as mídias que o compõem, e os instantes em que estas são necessárias na aplicação.

O tempo de transferência deve considerar fatores como a vazão do canal de comunica-

ção estabelecido entre o servidor e o receptor, e o tamanho dos objetos a serem enviados.

Um objeto que está sendo transmitido por meio de um carrossel tem seu tempo de transferência dado pela taxa de transmissão utilizada para transferir o carrossel, e o tempo necessário para acessar certo objeto dentro do carrossel. O período de um ciclo do carrossel determina o tempo máximo necessário para um receptor obter uma aplicação, ou um objeto especifíco. O tempo de acesso gera um retardo na entrega do conteúdo, e no pior caso, corresponde a um ciclo completo do carrossel de objetos.

Seja B_{size}, a largura de banda estimada do carrossel de objetos e T_{AC} o tempo de acesso ao carrossel. O tempo total para que um objeto de mídia M_i, seja transferido do servidor para o receptor, é dado pelo tempo gasto para sua transmissão pelo carrossel, acrescido do tempo de acesso a um determinado objeto. Assim, o tempo de transferência de um objeto de mídia, T_t, é dado pela Equação 5.1.

$$T_t = \frac{M_{i_{size}}}{B_{size}} + T_{ac} \tag{5.1}$$

O método responsável por definir o instante para envio de cada carrossel, obtém este valor subtraindo o tempo em que as mídias que o compõem são necessárias na aplicação (dado pelo plano de distribuição), do tempo de transferência da mídia de maior tamanho, que consequentemente, possui maior tempo de transferência. Seja T_{m_i}, o instante de exibição da mídia, dado pelo plano de distribuição, e T_{max_i}, o maior tempo de transferência de um objeto de mídia do carrossel, o instante $T_{carrossel_i}$ para envio de um carrossel, pode ser calculado pela Equação 5.2.

$$T_{carrossel_i} = T_{m_i} - T_{max_i} \tag{5.2}$$

O mecanismo de entrega do conteúdo através do carrossel FLUTE foi realizado através das bibliotecas MAD-ALCLIB, MAD-SDPLIB e MAD-FLUTELIB, desenvolvidas pela Tampere University of Technology (TUT) (TUT, 2004). Essas bibliotecas são responsáveis pela implementação do protocolo ALC/LCT, implementação de descritores de sessão FLUTE, e pela implementação do protocolo FLUTE, respectivamente.

5.3 INSTANCIAÇÃO DO FRAMEWORK PARA RECEPÇÃO DE CONTEÚDO

Os diferentes cenários de transmissão de aplicações implicam em modificações do formatador/exibidor do conteúdo. Para atender aos diferentes requisitos de cada cenário, a classe ReceptionManager pode ser especializada, como mostra a Figura 5.3.

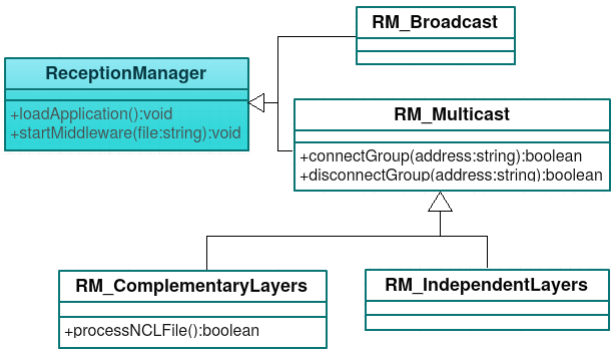

Figura 5.3: Instanciação do componente ReceptionManager

A classe **RM_Broadcast** implementa os mecanismos para permitir que o receptor execute uma aplicação multimídia que está sendo enviada por um único canal. Já o receptor de uma aplicação enviada por multicast, utilizando a estratégia de transmissão com camadas independentes, deve ser capaz de chavear entre camadas diferentes. Ele começa recebendo uma camada inicial, concretizada por meio de um grupo multicast, que representa a cadeia principal do HTG. Daí em diante, toda ação disparada por eventos não-determinísticos levará à dispensa da camada atualmente sendo recebida e à recepção de uma nova, ou seja, a disassociação e associação a grupos multicast.

O componente **RM_IndependentLayers** gera o HTG que representa a aplicação recebida, e monitora sua reprodução. A partir do monitoramento da aplicação, junto com o arquivo que relaciona cada cadeia secundária do HTG a um grupo multicast, o receptor obtém a informação de qual grupo precisa se conectar.

No envio por camadas complementares, o receptor (**RM_ComplementaryLayers**) deve ser capaz de receber mais de uma camada ao mesmo tempo. Na ocorrência de eventos não-determinísticos, o receptor deverá se associar a novos grupos multicast, para receber as novas camadas. Também é esperado que o receptor se desassocie de um grupo, caso a camada correspondente deixe de existir ou nela não exista mais nenhum objeto de mídia de interesse.

5.4 MÉTODOS DE ENTREGA DO ARQUIVO NCL

Um documento NCL especifica como os objetos de mídia são estruturados e relacionados, no tempo e no espaço. Os nós de composição da aplicação e seus relacionamentos podem ser agrupadas em contextos, que em alguns casos são autocontidos. Contextos autocontidos são aqueles que podem representar uma aplicação completa, e possuem sentido caso sejam reproduzidos isoladamente.

O uso de contextos na especificação de uma aplicação NCL pode ser empregado para definir sua estrutura, e também com foco no reúso. Caso o autor opte por estruturar a aplicação em contextos, esta pode ser fragmentada, em vários arquivos NCL, cada um contendo um contexto autocontido. Entretanto, se nenhuma estruturação é definida pelo autor, estes contextos podem ser obtidos a partir da análise das relações de sincronismo da aplicação.

Considerando que um contexto autocontido contém todos os objetos e elos necessários para sua reprodução independente, o arquivo NCL, poderia ser fragmentado em contextos, e enviados ao receptor, apenas quando fosse necessário. Os contextos resultantes da fragmentação, se tornariam novos documentos NCL, que se referenciam através de elementos **<media>**. Desse modo, um arquivo NCL daria início à exibição de outro arquivo, a partir da especificação de elos.

O envio do arquivo NCL em fragmentos é interessante por permitir que apenas os elementos necessários para a reprodução da aplicação em um intervalo de tempo estejam contidos na especificação. Isso traz uma certa economia na análise da aplicação, porém algumas questões devem ser consideradas, como a capacidade do middleware em reproduzir múltiplas aplicações NCL.

Outra solução para o envio do documento NCL é a utilização de comandos de edição. Comandos de edição possibilitam que documentos NCL sejam criados ou modificados em tempo de exibição. Estes comandos são recebidos e executados pelo Gerenciador de Base Privada, que compõe o núcleo da máquina de apresentação de documentos NCL, juntamente com o formatador NCL.

Comandos de edição NCL (SOARES et al., 2006) são envelopados em uma estrutura denominada descritor de eventos, e podem ser recebidos pela rede (carrossel DSM-CC ou por canal de interatividade), através de objetos imperativos embutidos no documento NCL, ou por aplicações externas residentes no receptor. O comando de edição, é um

evento imprevisível e não-determinístico, dificultando o gerenciamento de transmissão de aplicações que o empregam.

Um descritor de evento é composto por um id, que o identifica de maneira unívoca, uma referência de tempo, com o exato momento para disparar o evento, e um campo de dados privados. O campo de dados privados possibilita a identificação do comando e a definição de parâmetros do evento de edição.

A entrega do documento NCL utilizando comandos de edição, tem como princípio "quebrar" a aplicação em comandos de edição, que alteram um documento base. A NCL base contém as mídias e elos relacionados a eventos determinísticos. E para cada evento não-determinístico, um comando de edição é criado. Dessa forma, o documento NCL é alterado com base na ocorrência de eventos determinísticos ao longo da apresentação da aplicação.

O uso de comandos de edição também é dependente do middleware de reprodução, e em alguns casos, podem levar a problemas de inconsistência, resultando em uma aplicação diferente da especificada pelo autor.

As limitações impostas pelo middleware utilizado para a implementação da solução, o fato de apresentarem pouco ganho a nível de economia de banda, e a demanda por maior processamento no receptor, são algumas desvantagens dos mecanismos de segmentação do arquivo NCL. Portanto, optou-se por enviar o documento NCL integralmente na camada comum para o caso de envio em camadas complementares.

No envio de aplicações multimídia por camadas complementares, é possível relacionar cada mídia a um grupo multicast que corresponde à sua camada de envio. Assim, o servidor precisa alterar o documento NCL de modo que o atributo *src* do elemento **<media>** aponte para o endereço FLUTE correspondente ao canal que a mídia é transmitida. Ao receber o documento NCL, o receptor realiza um pré-processamento do mesmo, para obter informação de quais grupos multicast se conectar para obter as mídias.

Como mencionado anteriormente, no envio de aplicações por camadas independentes, um objeto de mídia pode ser distribuído em mais de uma camada, inviabilizando a definição de uma relação única entre camada e mídia, como ocorre com camadas complementares. Por isso, no cenário de envio por camadas independentes, o documento NCL é enviado conforme definido pelo autor da aplicação. Neste cenário, o receptor recebe um arquivo que relaciona cada cadeia alternativa do HTG a um canal de entrega de conteúdo.

O receptor é então responsável por monitorar a execução da aplicação interativa, e quando o telespectador segue por uma cadeia alternativa do grafo, ele deve se conectar ao canal relacionado à cadeia, para receber os conteúdos que a compõem. Considerando que o tempo para se conectar a um novo canal pode gerar um retardo na recepção das mídias e levar à falhas na exibição da aplicação, as mídias relacionadas ao evento não-determinístico que devem ser iniciadas em até t segundos após a ocorrência do evento devem ser colocadas na camada que representa a cadeia temporal do HTG disparada pelo evento, e na camada base.

O tempo t para inserção das mídias em uma nova camada de conteúdo, é calculado com base no tempo gasto de uma volta no carrossel definido para esta nova camada ($T_{carrossel_i}$), e pode ser obtido pela Equação 5.3.

$$t = \frac{Carrossel_{size}}{B_{size}} \tag{5.3}$$

Este método de antecipação das mídias em uma camada de conteúdo evita que a aplicação apresente atrasos na entrega dos objetos que devem ser exibidos logo no início de uma camada alternativa. Porém, caso essas mídias sejam do tipo contínua (vídeo e áudio, por exemplo) e com longa duração, enviá-las totalmente na camada base, não é interessante. Isto porque aumentaria a quantidade média de dados recebidas pelos usuários que não estão interessados nestas mídias ligadas ao evento não-determinístico, e que por isso, se mantém conectadas à camada base para recepção do restante do conteúdo.

Uma solução para o caso de mídias contínuas não-determinísticas, é a segmentação das mesmas, de modo que apenas um segmento inicial seja enviado na camada base, e o restante, enviado pela camada alternativa de conteúdo. Com base na análise do comportamento da aplicação, o framework pode calcular qual o tamanho necessário para o segmento inicial, e realizar a segmentação do objeto de mídia. Esta funcionalidade não está disponível na solução proposta, porém deve ser considerada em trabalhos futuros.

6 RESULTADOS

Este capítulo apresenta os resultados obtidos nos experimentos realizados, com o objetivo de avaliar as estratégias de transmissão definidas no trabalho de acordo com a eficiência na entrega do conteúdo hipermídia. Para isso, as estratégias foram aplicadas nos cenários de instanciação do framework proposto apresentados no Capítulo 5 . Os testes foram realizados utilizando duas aplicações desenvolvidas na linguagem NCL, que apresentam tanto eventos determinísticos, quanto não-determinísticos. Essas aplicações são descritas no Apêndice A.

Durante os testes, as aplicações hipermídia foram enviadas por push para um grupo de 10 receptores, com diferentes comportamentos ao longo da apresentação, quando permitido pela aplicação. A diversidade de históricos de apresentação nos receptores é importante para avaliar a quantidade média de dados recebida por cada um, nos diferentes cenários de instanciação. Como o conteúdo é enviado por broadcast e multicast, sem nenhum canal de comunicação entre o receptor e o servido, a quantidade de receptores analisados não altera o desempenho do gerenciamento de transmissão.

Outra métrica utilizada para avaliação da solução foi a ocupação média do carrossel, ou seja, a quantidade de dados transportados ao longo da transmissão da aplicação. A partir desses dados foi possível avaliar as melhorias na entrega do conteúdo hipermídia, em relação ao consumo de banda e utilização da rede.

Nas estratégias de transmissão que empregam o envio em multicamadas, como mais de um carrossel é enviado, existe um overhead relacionado ao transporte dos dados de definição da estrutura do protocolo. Entretanto, esse valor é da ordem de Kbytes, e normalmente o conteúdo hipermídia contém dados na ordem de Mbytes a Gbytes. Desse modo, tal valor não foi considerado no cálculo de ocupação média do carrossel, para análise dos cenários propostos.

É importante destacar, que os mecanismos de transmissão utilizando o conceito de camadas de conteúdo, dependem da capacidade do receptor de alternar entre grupos multicast, para o caso de multicast em camadas independentes, e de se associar a mais de um grupo simultaneamente, para o multicast em camadas complementares. Caso não exista suporte para tais condições, os mecanismos não poderão ser empregados.

A primeira aplicação analisada, nomeada Episódio3 é caracterizada por um grande número de relacionamentos de sincronismo temporal entre o fluxo audiovisual, e mídias que buscam adicionar informações ao conteúdo principal. Essas mídias são apresentadas através de textos e imagens, e estão dispostas no tempo, de acordo com a Figura 6.1.

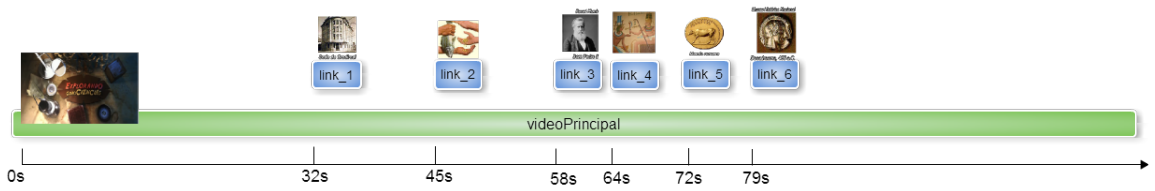

Figura 6.1: Disposição das mídias da aplicação 1 no eixo temporal

A segunda aplicação hipermídia (Episódio4), contém eventos determinísticos (sincronização de textos e imagens com o vídeo principal), e eventos não-determinísticos. O indeterminismo é dado por eventos de interação do usuário. A aplicação permite que o usuário decida se deseja ou não obter informaçõs adicionais ao conteúdo principal, além de possibilitar a escolha entre mais de um caminho de apresentação.

Dessa maneira, a apresentação da aplicação pode se dar de diferentes formas, conforme a interação do usuário com a mesma. Caso o usuário selecione o botão “Saiba Mais”, após um intervalo de 5s, um novo vídeo passa a ser exibido, relacionado ao conteúdo extra, além de um questionário sobre o assunto. Entretanto, se o usuário não selecionar tal opção apenas as mídias relacionados ao vídeo principal surgem ao longo de sua execução. A Figura 6.2 apresenta a disposição dos objetos de mídia da aplicação ao longo do tempo, onde as mídias apresentadas em vermelho estão relacionadas ao evento de seleção de um objeto, e por isso são não-determinísticas. Já as mídias em verde e azul, estão relacionadas aos eventos determinísticos definidos na aplicação (sincronismo temporal).

Ao receber as primeiras mídias necessárias à aplicação, o dispositivo receptor dá início à apresentação do conteúdo, e mantém-se conectado ao servidor para receber as mídias posteriores, que podem surgir ao longo do tempo. As aplicações foram transmitidas nos quatro cenários de transmissão de conteúdo hipermídia definidos anteriormente.

Como a aplicação 2 (Episódio4) pode ser reproduzida em dois caminhos diferentes pelos usuários, estes foram divididos em dois grupos. O Grupo 1 é composto por cinco usuários que optaram por não interagir com a aplicação, e o Grupo 2 contém os cinco restantes que selecionaram o botão “Saiba Mais” para obter mais informações sobre um assunto específico exibido no conteúdo. Os resultados dos testes são apresentados a seguir.

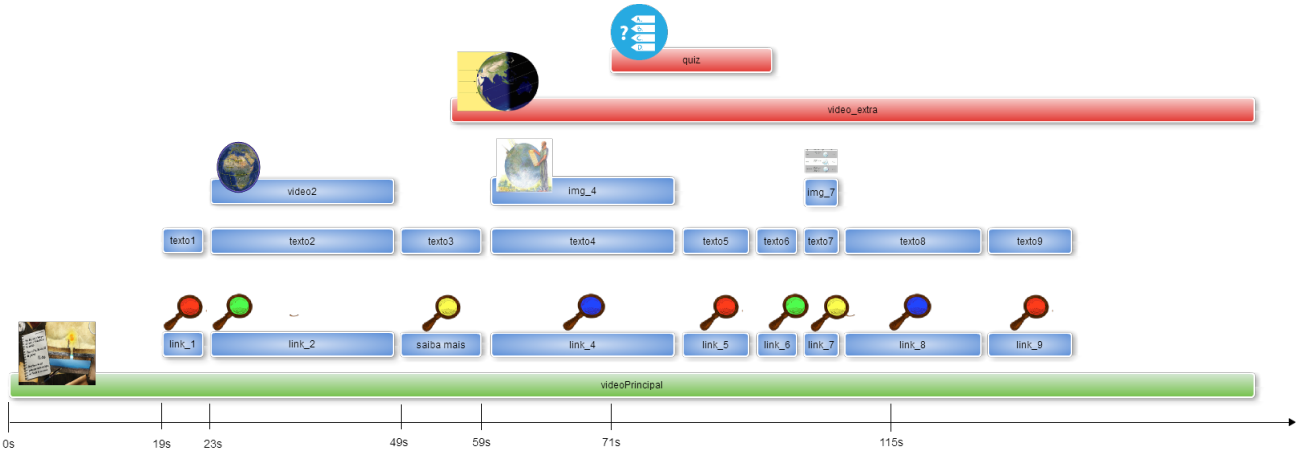

Figura 6.2: Disposição das mídias da aplicação 2 no eixo temporal

A aplicação “Episódio 3” apresenta apenas eventos determinísticos, e por isso, seu comportamento ao longo do tempo será sempre o mesmo para todos os usuários, e a quantidade de dados recebida por todos os receptores é igual ao tamanho total da aplicação. As mídias que a compõem totalizam aproximadamente 1,08Mbytes, e o conteúdo transportado pelo carrossel não inclui o fluxo audiovisual principal, que é transportado em um fluxo separado.

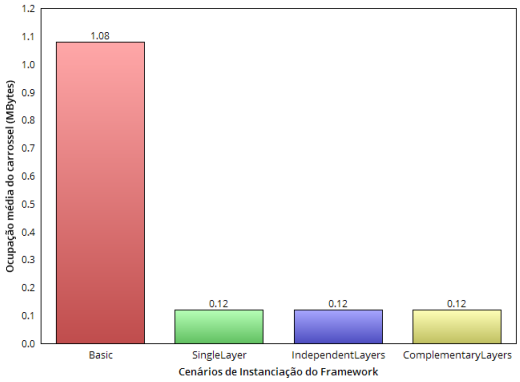

Figura 6.3: Ocupação média do carrossel para aplicação 1

A Figura 6.3 mostra ocupação média do carrossel para os métodos de transmissão por broadcast e multicast. Como a aplicação não apresenta nenhum evento não-determinístico, o desempenho dos Cenários *SingleLayer*, *IndependentLayers* e *ComplementaryLayers* tratando-se da ocupação do carrossel, é o mesmo. Isso porque nesses três cenários a construção do Plano de Transmissão se baseia na ocorrência de eventos determinísticos definidos na aplicação.

A aplicação “Episódio4” contém um conjunto de mídias que representam 30,33 Mby-

tes de dados transportados no carrossel. Essa aplicação é definida por eventos não-determinísticos e determinísticos. Sendo assim, entre os cenários de instanciação que utilizam o envio por broadcast em um único canal, o consumo de banda por parte do carrossel é reduzido no Cenário *SingleLayer*, como mostra a Figura 6.4.

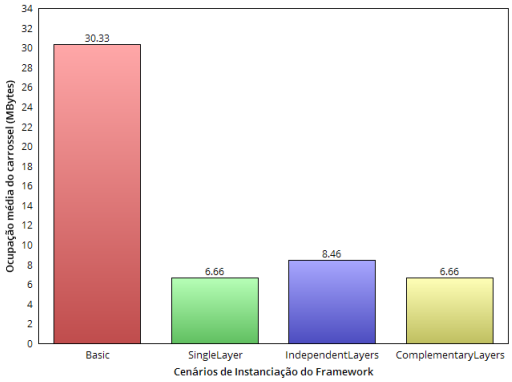

Figura 6.4: Ocupação média do carrossel para aplicação 2

Além da economia na ocupação do carrossel, devido à análise de eventos determinísticos, os cenários de instanciação que utilizam o envio por multicast com múltiplas camadas levam à redução na quantidade de dados recebida pelos usuários. Essa melhoria se deve à análise de eventos não-determinísticos, e pode ser vista na Figura 6.5, em que o grupo de usuários que não interagem com a aplicação, recebem um conjunto menor de dados.

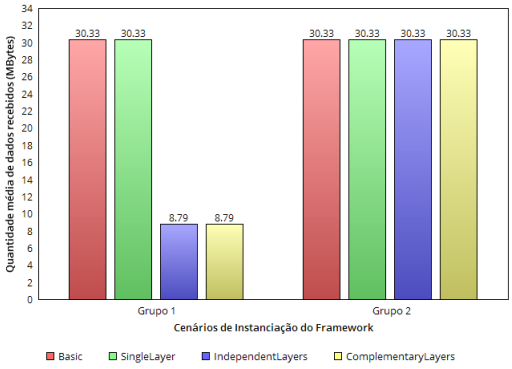

Figura 6.5: Quantidade de dados recebidos, por grupo de usuários, para aplicação 2

6.1 ANÁLISE DOS DADOS

No Cenário *Basic*, onde a aplicação é enviada por broadcast sem considerar os eventos definidos pela aplicação, o carrossel é estático, ou seja, seu tamanho não varia ao longo da

aplicação, independente de suas características de sincronismo temporal. Como a entrega é realizada por broadcast, em apenas um canal de comunicação entre o servidor e todos os receptores, nos Cenários *Basic* e *SingleLayer*, a quantidade de dados recebida pelos usuários permanece a mesma, independente de suas ações ao longo da apresentação.

Os testes demonstraram que o Cenário *SingleLayer* apresenta uma melhoria em relação ao anterior, considerando a ocupação média do carrossel. Como nesse cenário a construção do Plano de Transmissão se baseia na análise de eventos determinísticos, de modo que o servidor envia apenas os dados necessários ao momento de apresentação da aplicação, o tamanho médio do carrossel, em geral, é reduzido.

O método de transmissão por multicast, possibilita direcionar os dados em resposta às ações do usuário, uma vez que o conteúdo pode ser entregue em mais de um canal. Esse direcionamento pode ser dado de duas formas, seja por camadas independentes, onde o usuário pode alternar entre camadas de acordo com sua interação com a aplicação, ou por camadas complementares. Nas camadas complementares, o receptor se conecta a um ou mais grupos, de acordo com o que deseja receber da aplicação.

No Cenário *IndependentLayers*, cujo envio da aplicação é dado por multicast utilizando camadas independentes, a quantidade de dados recebidos pelo usuário corresponde apenas às mídias realmente consumidas. O mesmo ocorre no Cenário *ComplementaryLayers*, de envio por multicast com camadas complementares. Na transmissão de conteúdo utilizando multicast com camadas independentes, pode ocorrer repetição de mídias entre eles. Isso ocorre porque todas as camadas transmitem os objetos de mídia que compõem a cadeia principal do HTG. Essa característica pode ser observada pela diferença na quantidade média de dados enviadas no carrossel pela aplicação, entre os Cenários *IndependentLayers* e *ComplementaryLayers*. Aplicações com diferentes caminhos de reprodução possíveis, que apresentam um grande número de mídias em comum, levam ao maior consumo de banda no envio do carrossel.

7 CONCLUSÕES E TRABALHOS FUTUROS

Nesta dissertação foram estudados mecanismos para gerenciamento da transmissão de conteúdo entregues em diferentes cenários de provisão de serviços hipermídia. A fim de possibilitar o gerenciamento da entrega de conteúdo em modo push, foi proposto um framework genérico, cujos componentes atuam desde a análise do comportamento da aplicação, até o gerenciamento do streaming de dados e recepção do conteúdo. Além disso, o framework utiliza um grafo hipermídia temporal para modelar o comportamento temporal da aplicação e um Plano de Transmissão para escalonar o envio dos objetos de mídia que a compõe.

Cenários de uso do framework foram instanciados, considerando eventos determinísticos e não-determinísticos para minimizar o consumo de banda ao longo da transmissão do conteúdo, e o volume de dados armazenado pelo receptor. Um cenário para envio por broadcast com um único canal e outros dois para a entrega de aplicações em plataformas de serviços multimídia sobre IP, que dão suporte a multicast com múltiplos canais são propostas. No cenário de envio por broadcast com um único canal, é possível obter economia de recurcos da rede através da utilização de um Plano de Transmissão construído a partir da análise de eventos determinísticos. Em uma proposta de envio por multicast em múltiplos canais, a transmissão é separada em camadas independentes, onde cada camada equivale aos caminhos alternativos gerados por eventos não-determinísticos e é enviada em um grupo multicast diferente.

Para que o tempo de chaveamento entre grupos multicast não cause gaps na apresentação da aplicação por falta de mídias, é definido um cálculo para antecipação de mídias pertencentes a cadeias temporais secundárias do HTG.

Outra proposta é a associação de grupos multicast a camadas cujos conteúdos são complementares. As mídias relacionadas à cadeia principal do HTG são colocadas em uma camada base. E as mídias ligadas a eventos não-determinísticos são enviadas em grupos multicast específicos, que atuam como camadas de detalhamento, com as especificidades de cada cadeia temporal do HTG.

Neste trabalho também foi discutido como calcular o momento para adicionar uma mídia para envio, de acordo com o tamanho do carrossel, o tempo para sua transmissão

até o usuário e o momento definido no Plano de Transmissão.

A máquina de apresentação de conteúdo hipermídia precisa ser capaz de processar os dados recebidos referentes à aplicação, independente das técnicas e tecnologias de rede empregadas para entrega do conteúdo. Desse modo, um componente no receptor é apresentado, para processar a especificação da aplicação recebida, e em alguns casos, obter informações sobre quais endereços se conectar. Esse componente também monitora o comportamento da aplicação para definir a associação e desassociação de grupos ao longo de sua reprodução, para o cenário de multicast por camadas independentes.

É intuitivo que aplicações com poucos dados ou com duração curta praticamente não apresentarão ganho, e não irão impactar fortemente na rede. Assim, um trabalho futuro pode ser realizado, expandindo o framework para que este verifique se é valido gerenciar determinada aplicação, de acordo com suas características de duração e quantidade de dados. A extensão poderia verificar se o tamanho do carrossel realmente reduziu, para justificar o uso de algum mecanismo de gerência da transmissão.

O Plano de Transmissão gerado não considera alterações nas condições da rede durante a entrega da aplicação. Um mecanismo que altere o plano em tempo de execução em resposta a possíveis problemas na rede deve ser implementado.

O uso de camadas de envio pode ser definido pelo próprio autor da aplicação, que especifica a aplicação NCL em camadas, representadas por contextos. Dessa maneira, poderia existir uma camada com transmissão obrigatória e outras que poderiam ser enviadas separadamente, para o caso de baixa qualidade da transmissão. Para isso, é preciso definir um mapeamento que possibilite esta funcionalidade.

Outra funcionalidade deixada como trabalho futuro é a segmentação de objetos de mídia contínuos, para os casos em que estes são os primeiros dados a serem apresentados em uma camada independente. Com isso, apenas parte deste objeto será colocado antecipadamente na camada relacionada à cadeia principal do HTG. E receptores não interessados no conteúdo da camada independente, não precisarão armazenar todo o conteúdo dessa mídia contínua, apenas o segmento antecipado.

Apesar dos cenários de instanciação serem descritos para a transmissão de aplicações definidas em NCL, a utilização do framework não está associada à linguagem. Desse modo, o framework é capaz de gerenciar a transmissão de aplicações hipermídia especificadas em qualquer linguagem, desde que esta seja baseada em eventos.

REFERÊNCIAS

ABNT. **NBR 15606-2: 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**, 2016. ABNT, NBR 15606-2:2016. Versão corrigida.

COSTA, R. M. d. R.; MORENO, M. F.; SOARES, L. F. G. Intermedia synchronization management in dtv systems. In: ACM. **Proceedings of the eighth ACM symposium on Document engineering**, 2008. p. 289–297.

COSTA, R. M. de R. **Controle do Sincronismo Temporal de Aplicações Hipermídia**. Tese (Doutorado) — Programa de Pós-graduação em Informática da PUC-Rio, Agosto 2010.

ETSI. **Transport of MPEG-2 TS Based DVB Services over IP based Networks (and associated XML)**, 2008. V1.4.1.

ETSI. **Digital Video Broadcasting (DVB); IP Datacast over DVB-H: Content Delivery Protocols**, 2009. V 1.3.1.

ETSI. **Universal Mobile Telecommunications System (UMTS); LTE; Multimedia Broadcast/Multicast Service (MBMS); Protocols and codecs (release 10)**, 2013. V11.3.0.

FEZ, I. de; GIL, M.; FONS, J.; GUERRI, J. C.; PELECHANO, V. A personalized system for scalable distribution of multimedia content in multicast wireless networks. **Multimedia Tools and Applications**, Springer, v. 74, n. 21, p. 9595–9621, 2015.

FIGUEROA, A. A. M. **Pré-Busca de Conteúdo em Apresentações Multimídia**. Dissertao (Mestrado) — Programa de Pós-graduação em Informática da PUC-Rio, Março 2014.

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

ITU-RECOMMENDATION. Nested context language (ncl) and ginga-ncl for iptv services. April 2009.

LUBY, M.; GEMMELL, J.; VICISANO, L.; RIZZO, L.; J.CROWCROFT. **Asynchronous Layered Coding (ALC) Protocol Instantiation**, December 2002. RFC 3450.

MAHAJAN, M.; PARASHAR, M. Managing qos for multimedia applications in the differentiated services environment. **Journal of Network and Systems Management**, Springer, v. 11, n. 4, p. 469–498, 2003.

OTT, D. E.; MAYER-PATEL, K. An open architecture for transport-level protocol coordination in distributed multimedia applications. **ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)**, ACM, v. 3, n. 3, p. 17, 2007.

PAILA, T.; WALSH, R.; LUBY, M.; ROCA, V.; LEHTONEN, R. **FLUTE - File Delivery over Unidirectional Transport**, November 2012. RFC 6726.

PESSOA, B. J. d. S.; FILHO, G. L. de S.; CABRAL, L. d. A. F. Metaheurísticas aplicadas à geraçao de carrossel no sistema brasileiro de tv digital. In: ACM. **Proceedings of the 14th Brazilian Symposium on Multimedia and the Web**, 2008. p. 91–98.

RODRIGUES, R. F.; SOARES, L. F. G. A framework for prefetching mechanisms in hypermedia presentations. In: IEEE. **Multimedia Software Engineering, 2002. Proceedings. Fourth International Symposium on**, 2002. p. 278–285.

SOARES, L. F. G.; MORENO, M. F.; NETO, C. D. S. S.; MORENO, M. F. Ginga-ncl: declarative middleware for multimedia iptv services. **IEEE Communications Magazine**, IEEE, v. 48, n. 6, p. 74–81, 2010.

SOARES, L. F. G.; RODRIGUES, R. F. Nested context model 3.0: Part 1 - ncm core. **Monografias em Ciência da Computação do Departamento de Informática, PUC-Rio**, n. 18/05, 2005.

SOARES, L. F. G.; RODRIGUES, R. F. Nested context language 3.0 part 8 - ncl digital tv profiles. **Monografias em Ciência da Computação do Departamento de Informática da PUC-Rio**, v. 1200, n. 35, p. 06, 2006.

SOARES, L. F. G.; RODRIGUES, R. F.; COSTA, R. R.; MORENO, M. F. Nested context language 3.0: Part 9–ncl live editing commands. **Monografias em Ciência da Computação do Departamento de Informática, PUC-Rio**, v. 6, p. 36, 2006.

SOARES, L. F. G. S. **Programando em NCL 3.0: desenvolvimento de aplicaçoes para middleware Ginga: TV digital e Web**, 2009.

TUT. **MAD Project's**. 2004. `http://mad.cs.tut.fi/index.html`.

Apêndice A - APLICAÇÕES HIPERMÍDIA ESPECIFICADAS EM NCL

A.1 APLICAÇÃO EPISÓDIO 3

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<ncl id="episodio3" xmlns="http://www.ncl.org.br/NCL3.0/EDTVProfile">
<head>
<regionBase>
<region id="regVideo" width="100%" height="100%" zIndex="0"/>
<region id="regLupaBarra" left="10.9%" top="77.8%" width="14.6%"
height="22.2%" zIndex="1"/>
<region id="regBarra" left="18.75%" top="80.1%"
width="71.9%" height="16.7%" zIndex="2"/>
<region id="regSair" left="17.2%" top="82.4%"
width="4.2%" height="12%" zIndex="3"/>
<region id="regTexto" left="25.5%" top="81.5%"
width="59.4%" height="18.5%" zIndex="3"/>
<region id="regFoto" left="70.3%" top="67.6%"
width="17.2%" height="32.4%" zIndex="3"/>
<region id="regLupaInterativa" left="5%" top="67.6%"
width="29.2%" height="7.4%" zIndex="1"/>
<region id="regLupaVideo1" left="26.6%" top="42.6%"
width="4.1%" height="7.4%" zIndex="1"/>
</regionBase>
<descriptorBase>
    <descriptor id="descVideo" region="regVideo"/>
    <descriptor id="descLupaBarra" region="regLupaBarra"/>
    <descriptor id="descBarra" region="regBarra"/>
    <descriptor id="descSair" region="regSair"/>
    <descriptor id="descTexto" region="regTexto" explicitDur="5s"/>
```

```
    <descriptor id=‘‘descFoto’’ region=‘‘regFoto’’/>
    <descriptor id=‘‘descLupaVideo1’’ region=‘‘regLupaVideo1’’ explicitDur=‘‘1.5s’’/>
    <descriptor id=‘‘descAudioLupa’’/>
    <descriptor id=‘‘descLupaInterativa’’ region=‘‘regLupaInterativa’’/>
</descriptorBase>
<connectorBase>
    <importBase documentURI=‘‘causalConnBase.ncl’’ alias=‘‘conEx’’/>
</connectorBase>
</head>
<body>
    <port id=‘‘portaPrincipal’’ component=‘‘videoPrincipal’’/>
    <media id=‘‘videoPrincipal’’ src=‘‘../Episodio3s.mp4’’ descriptor=‘‘descVideo’’>
        <area id=‘‘ancora1’’ begin=‘‘32s’’ end=‘‘105s’’/>
        <area id=‘‘ancora2’’ begin=‘‘45s’’/>
        <area id=‘‘ancora3’’ begin=‘‘58s’’/>
        <area id=‘‘ancora4’’ begin=‘‘64s’’/>
        <area id=‘‘ancora5’’ begin=‘‘72s’’/>
        <area id=‘‘ancora6’’ begin=‘‘79s’’/>
    </media>
    <media id=‘‘imgTexto1’’ src=‘‘media/link_1_texto.png’’ descriptor=‘‘descTexto’’/>
    <media id=‘‘imgFoto1’’ src=‘‘media/link_1_foto.png’’ descriptor=‘‘descFoto’’/>
    <media id=‘‘imgTexto2’’ src=‘‘media/link_2_texto.png’’ descriptor=‘‘descTexto’’/>
    <media id=‘‘imgFoto2’’ src=‘‘media/link_2_foto.png’’ descriptor=‘‘descFoto’’/>
    <media id=‘‘imgTexto3’’ src=‘‘media/link_3_texto.png’’ descriptor=‘‘descTexto’’/>
    <media id=‘‘imgFoto3’’ src=‘‘media/link_3_foto.png’’ descriptor=‘‘descFoto’’/>
    <media id=‘‘imgTexto4’’ src=‘‘media/link_4_texto.png’’ descriptor=‘‘descTexto’’/>
    <media id=‘‘imgFoto4’’ src=‘‘media/link_4_foto.png’’ descriptor=‘‘descFoto’’/>
    <media id=‘‘imgTexto5’’ src=‘‘media/link_5_texto.png’’ descriptor=‘‘descTexto’’/>
    <media id=‘‘imgFoto5’’ src=‘‘media/link_5_foto.png’’ descriptor=‘‘descFoto’’/>
    <media id=‘‘imgTexto6’’ src=‘‘media/link_6_texto.png’’ descriptor=‘‘descTexto’’/>
    <media id=‘‘imgFoto6’’ src=‘‘media/link_6_foto.png’’ descriptor=‘‘descFoto’’/>
    <context id=‘‘ctxBarra’’>
```

```
    <port id=‘‘portaBarra’’ component=‘‘imgBarra’’/>
    <port id=‘‘portaLupaBarra’’ component=‘‘imgLupaBarra’’/>
    <media id=‘‘imgLupaBarra’’ src=‘‘media/link_lupa.png’’ descriptor=‘‘descLupaBarra’’/>
    <media id=‘‘imgBarra’’ src=‘‘media/link_faixa.png’’ descriptor=‘‘descBarra’’/>
</context>
<link id=‘‘link1’’ xconnector=‘‘conEx#onBegin1StartN’’>
    <bind role=‘‘onBegin’’ component=‘‘videoPrincipal’’ interface=‘‘ancora1’’/>
    <bind role=‘‘start’’ component=‘‘imgTexto1’’/>
    <bind role=‘‘start’’ component=‘‘imgFoto1’’/>
    <bind role=‘‘start’’ component=‘‘ctxBarra’’/>
</link>
<link id=‘‘link2’’ xconnector=‘‘conEx#onEnd1StopN’’>
<bind role=‘‘onEnd’’ component=‘‘imgTexto1’’/>
<bind role=‘‘stop’’ component=‘‘imgFoto1’’/>
<bind role=‘‘stop’’ component=‘‘ctxBarra’’/>
</link>
<link xconnector=‘‘conEx#onBegin1StartN’’>
    <bind role=‘‘onBegin’’ component=‘‘videoPrincipal’’ interface=‘‘ancora2’’/>
    <bind role=‘‘start’’ component=‘‘imgTexto2’’/>
    <bind role=‘‘start’’ component=‘‘imgFoto2’’/>
    <bind role=‘‘start’’ component=‘‘ctxBarra’’/>
</link>
<link xconnector=‘‘conEx#onEnd1StopN’’>
<bind role=‘‘onEnd’’ component=‘‘imgTexto2’’/>
<bind role=‘‘stop’’ component=‘‘imgFoto2’’/>
<bind role=‘‘stop’’ component=‘‘ctxBarra’’/>
</link>
<link xconnector=‘‘conEx#onBegin1StartN’’>
<bind role=‘‘onBegin’’ component=‘‘videoPrincipal’’ interface=‘‘ancora3’’/>
<bind role=‘‘start’’ component=‘‘imgTexto3’’/>
<bind role=‘‘start’’ component=‘‘imgFoto3’’/>
<bind role=‘‘start’’ component=‘‘ctxBarra’’/>
```

```
</link>
<link xconnector=“conEx#onEnd1StopN”>
<bind role=“onEnd” component=“imgTexto3”/>
<bind role=“stop” component=“imgFoto3”/>
<bind role=“stop” component=“ctxBarra”/>
</link>
<link xconnector=“conEx#onBegin1StartN”>
<bind role=“onBegin” component=“videoPrincipal” interface=“ancora4”/>
<bind role=“start” component=“imgTexto4”/>
<bind role=“start” component=“imgFoto4”/>
<bind role=“start” component=“ctxBarra”/>
</link>
<link xconnector=“conEx#onEnd1StopN”>
<bind role=“onEnd” component=“imgTexto4”/>
<bind role=“stop” component=“imgFoto4”/>
<bind role=“stop” component=“ctxBarra”/>
</link>
<link xconnector=“conEx#onBegin1StartN”>
<bind role=“onBegin” component=“videoPrincipal” interface=“ancora5”/>
<bind role=“start” component=“imgTexto5”/>
<bind role=“start” component=“imgFoto5”/>
<bind role=“start” component=“ctxBarra”/>
</link>
<link xconnector=“conEx#onEnd1StopN”>
<bind role=“onEnd” component=“imgTexto5”/>
<bind role=“stop” component=“imgFoto5”/>
<bind role=“stop” component=“ctxBarra”/>
</link>
<link xconnector=“conEx#onBegin1StartN”>
<bind role=“onBegin” component=“videoPrincipal” interface=“ancora6”/>
<bind role=“start” component=“imgTexto6”/>
<bind role=“start” component=“imgFoto6”/>
```

```
    <bind role=‘‘start’’ component=‘‘ctxBarra’’/>
    </link>
    <link xconnector=‘‘conEx#onEnd1StopN’’>
    <bind role=‘‘onEnd’’ component=‘‘imgTexto6’’/>
    <bind role=‘‘stop’’ component=‘‘imgFoto6’’/>
    <bind role=‘‘stop’’ component=‘‘ctxBarra’’/>
    </link>
</body>
</ncl>
```

A.2 APLICAÇÃO EPISÓDIO 4

```
<?xml version=‘‘1.0’’ encoding=‘‘ISO−8859−1’’?>
<ncl id=‘‘episodio4’’ xmlns=‘‘http://www.ncl.org.br/NCL3.0/EDTVProfile’’>
<head>
<regionBase>
<region id=‘‘regVideo’’ width=‘‘100%’’ height=‘‘100%’’ zIndex=‘‘0’’/>
<region id=‘‘regFundoQuiz’’ left=‘‘0’’ top=‘‘0’’ width=‘‘100%’’ height=‘‘100%’’ zIndex=‘‘0’’/>
<region id=‘‘regQuiz’’ left=‘‘17.5%’’ top=‘‘63.7%’’ width=‘‘76%’’ height=‘‘23.1%’’ zIndex=‘‘2’’/>
<region id=‘‘regBarra’’ left=‘‘12.5%’’ top=‘‘76.9%’’ width=‘‘76%’’
  height=‘‘23.1%’’ zIndex=‘‘2’’/>
<region id=‘‘regSair’’ left=‘‘16.4%’’ top=‘‘82.4%’’ width=‘‘3.9%’’ height=‘‘9.3%’’ zIndex=‘‘3’’/>
<region id=‘‘regTexto’’ left=‘‘21.4%’’ top=‘‘63%’’ width=‘‘65.1%’’ height=‘‘37%’’ zIndex=‘‘3’’/>
<region id=‘‘regLupaInterativa’’ left=‘‘2.1%’’ top=‘‘87%’’ width=‘‘20.3%’’
  height=‘‘7.4%’’ zIndex=‘‘2’’/>
<region id=‘‘regLupaVideo1’’ left=‘‘20.7%’’ top=‘‘43.1%’’ width=‘‘4.1%’’
  height=‘‘7.4%’’ zIndex=‘‘1’’/>
<region id=‘‘regPergaminho’’ left=‘‘12.5%’’ top=‘‘63%’’ width=‘‘76%’’
  height=‘‘33.1%’’ zIndex=‘‘1’’/>
<region id=‘‘regAba’’ left=‘‘0%’’ top=‘‘67.6%’’ width=‘‘2.86%’’ height=‘‘10.2%’’ zIndex=‘‘1’’/>
</regionBase>
<descriptorBase>
<descriptor id=‘‘descVideo’’ region=‘‘regVideo’’/>
```

```
<descriptor id=‘‘descFundoQuiz’’ region=‘‘regFundoQuiz’’/>
<descriptor id=‘‘descBarra’’ region=‘‘regBarra’’/>
<descriptor id=‘‘descSair’’ region=‘‘regSair’’/>
<descriptor id=‘‘descTexto’’ region=‘‘regTexto’’/>
<descriptor id=‘‘descPergaminho’’ region=‘‘regPergaminho’’ explicitDur=‘‘5s’’>
        <descriptorParam name=‘‘transparency’’ value=‘‘30%’’/>
</descriptor>
<descriptor id=‘‘descQuiz’’ region=‘‘regQuiz’’/>
<descriptor id=‘‘descAba’’ region=‘‘regAba’’/>
<descriptor id=‘‘descAudioLupa’’/>
<descriptor id=‘‘descLupaInterativa’’ region=‘‘regLupaInterativa’’/>
<descriptor id=‘‘descLupaVideo1’’ region=‘‘regLupaVideo1’’ explicitDur=‘‘1.5s’’/>
</descriptorBase>
<connectorBase>
        <importBase documentURI=‘‘causalConnBase.ncl’’ alias=‘‘conEx’’/>
</connectorBase>
</head>
<body>
        <port id=‘‘portaPrincipal’’ component=‘‘videoPrincipal’’/>
        <media id=‘‘videoPrincipal’’ src=‘‘../Episodio4s.mp4’’ descriptor=‘‘descVideo’’>
                <area id=‘‘ancora0’’ begin=‘‘0s’’ end=‘‘126s’’/>
                <area id=‘‘ancora1’’ begin=‘‘19s’’/>
                <area id=‘‘ancora2’’ begin=‘‘23s’’/>
                <area id=‘‘ancora3’’ begin=‘‘49s’’/>
                <area id=‘‘ancora4’’ begin=‘‘59s’’/>
                <area id=‘‘ancora5’’ begin=‘‘82s’’/>
                <area id=‘‘ancora6’’ begin=‘‘90s’’/>
                <area id=‘‘ancora7’’ begin=‘‘95s’’/>
                <area id=‘‘ancora8’’ begin=‘‘99s’’/>
                <area id=‘‘ancora9’’ begin=‘‘116s’’/>
        </media>
        <media id=‘‘imgSair’’ src=‘‘media/link_botao_sair.png’’ descriptor=‘‘descSair’’/>
```

```
<media id=‘‘video_extra’’ src=‘‘media/solsticio.mp4’’ descriptor=‘‘descVideo’’>
        <area id=‘‘ancoraQuiz’’ begin=‘‘115s’’/>
</media>
<media id=‘‘imgLink1’’ src=‘‘media/link_1.png’’ descriptor=‘‘descLupaInterativa’’/>
<media id=‘‘imgLink2’’ src=‘‘media/link_2.png’’ descriptor=‘‘descLupaInterativa’’/>
<media id=‘‘imgLink3’’ src=‘‘media/link_3_new.png’’ descriptor=‘‘descLupaInterativa’’/>
<media id=‘‘imgLink4’’ src=‘‘media/link_4.png’’ descriptor=‘‘descLupaInterativa’’/>
<media id=‘‘imgLink5’’ src=‘‘media/link_5.png’’ descriptor=‘‘descLupaInterativa’’/>
<media id=‘‘imgLink6’’ src=‘‘media/link_6.png’’ descriptor=‘‘descLupaInterativa’’/>
<media id=‘‘imgLink7’’ src=‘‘media/link_7.png’’ descriptor=‘‘descLupaInterativa’’/>
<media id=‘‘imgLink8’’ src=‘‘media/link_8.png’’ descriptor=‘‘descLupaInterativa’’/>
<media id=‘‘imgLink9’’ src=‘‘media/link_9.png’’ descriptor=‘‘descLupaInterativa’’/>
<media id=‘‘imgTexto1’’ src=‘‘media/link_1_texto.png’’ descriptor=‘‘descTexto’’/>
<media id=‘‘imgTexto2’’ src=‘‘media/link_2_texto_1.png’’ descriptor=‘‘descTexto’’/>
<media id=‘‘imgTexto3’’ src=‘‘media/link_3_texto_1.png’’ descriptor=‘‘descTexto’’/>
<media id=‘‘imgTexto4’’ src=‘‘media/link_4_texto.png’’ descriptor=‘‘descTexto’’/>
<media id=‘‘imgTexto5’’ src=‘‘media/link_5_texto.png’’ descriptor=‘‘descTexto’’/>
<media id=‘‘imgTexto6’’ src=‘‘media/link_6_texto.png’’ descriptor=‘‘descTexto’’/>
<media id=‘‘imgTexto7’’ src=‘‘media/link_7_texto.png’’ descriptor=‘‘descTexto’’/>
<media id=‘‘imgTexto8’’ src=‘‘media/link_8_texto.png’’ descriptor=‘‘descTexto’’/>
<media id=‘‘imgTexto9’’ src=‘‘media/link_9_texto.png’’ descriptor=‘‘descTexto’’/>
<media id=‘‘fundoQuiz’’ src=‘‘media/fundo_quiz.png’’ descriptor=‘‘descFundoQuiz’’/>
<media id=‘‘pergaminho’’ src=‘‘media/link_faixa.png’’ descriptor=‘‘descPergaminho’’/>
<media id=‘‘quiz’’ src=‘‘media/main.lua’’ descriptor=‘‘descQuiz’’>
        <area id=‘‘quest’’/>
        <area id=‘‘acerto’’/>
</media>
<media id=‘‘settings’’ type=‘‘application/x−ginga−settings’’>
        <property name=‘‘service.currentKeyMaster’’ value=‘‘’’/>
</media>
<context id=‘‘ctxBarra’’>
        <port id=‘‘portaBarra’’ component=‘‘imgBarra’’/>
```

```
        <port id=‘‘portaSair’’ component=‘‘imgSair’’/>
        <media id=‘‘imgBarra’’ src=‘‘media/link_faixa.png’’ descriptor=‘‘descBarra’’/>
        <media id=‘‘imgSairLocal’’ refer=‘‘imgSair’’ instance=‘‘instSame’’/>
        <link xconnector=‘‘conEx#onKeySelectionSetNStopNStartN’’>
                <bind role=‘‘onSelection’’ component=‘‘imgSairLocal’’>
                        <bindParam name=‘‘tecla’’ value=‘‘BACK’’/>
                </bind>
                <bind role=‘‘stop’’ component=‘‘ctxBarra’’/>
        </link>
</context>
<link xconnector=‘‘conEx#OnBeginStartNSetNStopN’’>
        <bind role=‘‘onBegin’’ component=‘‘videoPrincipal’’ interface=‘‘ancora1’’/>
        <bind role=‘‘start’’ component=‘‘imgLink1’’>
                <bindParam name=‘‘retardo’’ value=‘‘1.5s’’/>
        </bind>
</link>
<link xconnector=‘‘conEx#onKeySelectionSetNStopNStartN’’>
        <bind role=‘‘onSelection’’ component=‘‘imgLink1’’>
                <bindParam name=‘‘tecla’’ value=‘‘RED’’/>
        </bind>
        <bind role=‘‘stop’’ component=‘‘imgLink1’’/>
        <bind role=‘‘start’’ component=‘‘ctxBarra’’/>
        <bind role=‘‘start’’ component=‘‘imgTexto1’’/>
        <bind role=‘‘start’’ component=‘‘imgSair’’/>
</link>
<link xconnector=‘‘conEx#onKeySelectionSetNStopNStartN’’>
        <bind role=‘‘onSelection’’ component=‘‘imgSair’’>
                <bindParam name=‘‘tecla’’ value=‘‘BACK’’/>
        </bind>
        <bind role=‘‘stop’’ component=‘‘imgTexto1’’/>
</link>
<link xconnector=‘‘conEx#OnBeginStartNSetNStopN’’>
```

```
        <bind role=‘‘onBegin’’ component=‘‘videoPrincipal’’ interface=‘‘ancora2’’/>
        <bind role=‘‘start’’ component=‘‘imgLink2’’>
                <bindParam name=‘‘retardo’’ value=‘‘1.5s’’/>
        </bind>
        <bind role=‘‘stop’’ component=‘‘imgLink1’’/>
        <bind role=‘‘stop’’ component=‘‘imgTexto1’’/>
        <bind role=‘‘stop’’ component=‘‘ctxBarra’’/>
</link>
<link xconnector=‘‘conEx#onKeySelectionSetNStopNStartN’’>
        <bind role=‘‘onSelection’’ component=‘‘imgLink2’’>
                <bindParam name=‘‘tecla’’ value=‘‘GREEN’’/>
        </bind>
        <bind role=‘‘stop’’ component=‘‘imgLink2’’/>
        <bind role=‘‘start’’ component=‘‘ctxBarra’’/>
        <bind role=‘‘start’’ component=‘‘imgTexto2’’/>
        <bind role=‘‘start’’ component=‘‘imgSair’’/>
</link>
<link xconnector=‘‘conEx#onKeySelectionSetNStopNStartN’’>
        <bind role=‘‘onSelection’’ component=‘‘imgSair’’>
                <bindParam name=‘‘tecla’’ value=‘‘BACK’’/>
        </bind>
        <bind role=‘‘stop’’ component=‘‘imgTexto2’’/>
</link>
<link xconnector=‘‘conEx#OnBeginStartNSetNStopN’’>
        <bind role=‘‘onBegin’’ component=‘‘videoPrincipal’’ interface=‘‘ancora3’’/>
        <bind role=‘‘start’’ component=‘‘imgLink3’’>
                <bindParam name=‘‘retardo’’ value=‘‘1.5s’’/>
        </bind>
        <bind role=‘‘stop’’ component=‘‘imgLink2’’/>
        <bind role=‘‘stop’’ component=‘‘imgTexto2’’/>
        <bind role=‘‘stop’’ component=‘‘ctxBarra’’/>
</link>
```

```
<link xconnector=‘‘conEx#onKeySelectionSetNStopNStartN’’>
        <bind role=‘‘onSelection’’ component=‘‘imgLink3’’>
                <bindParam name=‘‘tecla’’ value=‘‘YELLOW’’/>
        </bind>
        <bind role=‘‘stop’’ component=‘‘imgLink3’’/>
        <bind role=‘‘start’’ component=‘‘ctxBarra’’/>
        <bind role=‘‘start’’ component=‘‘imgTexto3’’/>
</link>
<link xconnector=‘‘conEx#onKeySelectionSetNStopNStartN’’>
        <bind role=‘‘onSelection’’ component=‘‘imgSair’’>
                <bindParam name=‘‘tecla’’ value=‘‘BACK’’/>
        </bind>
        <bind role=‘‘stop’’ component=‘‘imgTexto3’’/>
</link>
<link xconnector=‘‘conEx#onBeginStart_delay’’>
        <bind role=‘‘onBegin’’ component=‘‘imgTexto3’’/>
        <bind role=‘‘stop’’ component=‘‘videoPrincipal’’>
                <bindParam name=‘‘delay’’ value=‘‘5s’’/>
        </bind>
        <bind role=‘‘stop’’ component=‘‘imgTexto3’’>
                <bindParam name=‘‘delay’’ value=‘‘5s’’/>
        </bind>
        <bind role=‘‘stop’’ component=‘‘ctxBarra’’>
                <bindParam name=‘‘delay’’ value=‘‘5s’’/>
        </bind>
        <bind role=‘‘start’’ component=‘‘video_extra’’>
                <bindParam name=‘‘delay’’ value=‘‘5s’’/>
        </bind>
</link>
<link xconnector=‘‘conEx#onBeginStart’’>
        <bind role=‘‘onBegin’’ component=‘‘video_extra’’ interface=‘‘ancoraQuiz’’/>
        <bind role=‘‘start’’ component=‘‘pergaminho’’/>
```

```
        <bind role=‘‘start’’ component=‘‘quiz’’ interface=‘‘quest’’/>
</link>
<link xconnector=‘‘conEx#onBeginSet’’>
        <bind role=‘‘onBegin’’ component=‘‘quiz’’ />
        <bind role=‘‘set’’ component=‘‘settings’’ interface=‘‘service.currentKeyMaster’’>
                <bindParam name=‘‘var’’ value=‘‘quiz’’/>
        </bind>
</link>
<link xconnector=‘‘conEx#onEndStopNSetN’’>
        <bind role=‘‘onEnd’’ component=‘‘pergaminho’’/>
        <bind role=‘‘set’’ component=‘‘settings’’
            interface=‘‘service.currentKeyMaster’’>
                <bindParam name=‘‘var’’ value=‘‘’’/>
        </bind>
        <bind role=‘‘stop’’ component=‘‘quiz’’ interface=‘‘quest’’/>
</link>
<link xconnector=‘‘conEx#OnBeginStartNSetNStopN’’>
        <bind role=‘‘onBegin’’ component=‘‘videoPrincipal’’ interface=‘‘ancora4’’/>
        <bind role=‘‘start’’ component=‘‘imgLink4’’>
                <bindParam name=‘‘retardo’’ value=‘‘1.5s’’/>
        </bind>
        <bind role=‘‘stop’’ component=‘‘imgLink3’’/>
</link>
<link xconnector=‘‘conEx#onKeySelectionSetNStopNStartN’’>
        <bind role=‘‘onSelection’’ component=‘‘imgLink4’’>
                <bindParam name=‘‘tecla’’ value=‘‘BLUE’’/>
        </bind>
        <bind role=‘‘stop’’ component=‘‘imgLink4’’/>
        <bind role=‘‘start’’ component=‘‘ctxBarra’’/>
        <bind role=‘‘start’’ component=‘‘imgTexto4’’/>
        <bind role=‘‘start’’ component=‘‘imgSair’’/>
</link>
```

```
<link xconnector=‘‘conEx#onKeySelectionSetNStopNStartN’’>
        <bind role=‘‘onSelection’’ component=‘‘imgSair’’>
                <bindParam name=‘‘tecla’’ value=‘‘BACK’’/>
        </bind>
        <bind role=‘‘stop’’ component=‘‘imgTexto4’’/>
</link>
<link xconnector=‘‘conEx#OnBeginStartNSetNStopN’’>
        <bind role=‘‘onBegin’’ component=‘‘videoPrincipal’’ interface=‘‘ancora5’’/>
        <bind role=‘‘start’’ component=‘‘imgLink5’’>
                <bindParam name=‘‘retardo’’ value=‘‘1.5s’’/>
        </bind>
        <bind role=‘‘stop’’ component=‘‘imgLink4’’/>
        <bind role=‘‘stop’’ component=‘‘imgTexto4’’/>
        <bind role=‘‘stop’’ component=‘‘ctxBarra’’/>
</link>
<link xconnector=‘‘conEx#onKeySelectionSetNStopNStartN’’>
        <bind role=‘‘onSelection’’ component=‘‘imgLink5’’>
                <bindParam name=‘‘tecla’’ value=‘‘RED’’/>
        </bind>
        <bind role=‘‘stop’’ component=‘‘imgLink5’’/>
        <bind role=‘‘start’’ component=‘‘ctxBarra’’/>
        <bind role=‘‘start’’ component=‘‘imgTexto5’’/>
        <bind role=‘‘start’’ component=‘‘imgSair’’/>
</link>
<link xconnector=‘‘conEx#onKeySelectionSetNStopNStartN’’>
        <bind role=‘‘onSelection’’ component=‘‘imgSair’’>
                <bindParam name=‘‘tecla’’ value=‘‘BACK’’/>
        </bind>
        <bind role=‘‘stop’’ component=‘‘imgTexto5’’/>
</link>
<link xconnector=‘‘conEx#OnBeginStartNSetNStopN’’>
        <bind role=‘‘onBegin’’ component=‘‘videoPrincipal’’ interface=‘‘ancora6’’/>
```

```
        <bind role=‘‘start’’ component=‘‘imgLink6’’>
                <bindParam name=‘‘retardo’’ value=‘‘1.5s’’/>
        </bind>
        <bind role=‘‘stop’’ component=‘‘imgLink5’’/>
        <bind role=‘‘stop’’ component=‘‘imgTexto5’’/>
        <bind role=‘‘stop’’ component=‘‘ctxBarra’’/>
</link>
<link xconnector=‘‘conEx#onKeySelectionSetNStopNStartN’’>
        <bind role=‘‘onSelection’’ component=‘‘imgLink6’’>
                <bindParam name=‘‘tecla’’ value=‘‘GREEN’’/>
        </bind>
        <bind role=‘‘stop’’ component=‘‘imgLink6’’/>
        <bind role=‘‘start’’ component=‘‘ctxBarra’’/>
        <bind role=‘‘start’’ component=‘‘imgTexto6’’/>
        <bind role=‘‘start’’ component=‘‘imgSair’’/>
</link>
<link xconnector=‘‘conEx#onKeySelectionSetNStopNStartN’’>
        <bind role=‘‘onSelection’’ component=‘‘imgSair’’>
                <bindParam name=‘‘tecla’’ value=‘‘BACK’’/>
        </bind>
        <bind role=‘‘stop’’ component=‘‘imgTexto6’’/>
</link>
<link xconnector=‘‘conEx#OnBeginStartNSetNStopN’’>
        <bind role=‘‘onBegin’’ component=‘‘videoPrincipal’’ interface=‘‘ancora7’’/>
        <bind role=‘‘start’’ component=‘‘imgLink7’’>
                <bindParam name=‘‘retardo’’ value=‘‘1.5s’’/>
        </bind>
        <bind role=‘‘stop’’ component=‘‘imgLink6’’/>
        <bind role=‘‘stop’’ component=‘‘imgTexto6’’/>
        <bind role=‘‘stop’’ component=‘‘ctxBarra’’/>
</link>
<link xconnector=‘‘conEx#onKeySelectionSetNStopNStartN’’>
```

```
        <bind role=‘‘onSelection’’ component=‘‘imgLink7’’>
                <bindParam name=‘‘tecla’’ value=‘‘YELLOW’’/>
        </bind>
        <bind role=‘‘stop’’ component=‘‘imgLink7’’/>
        <bind role=‘‘start’’ component=‘‘ctxBarra’’/>
        <bind role=‘‘start’’ component=‘‘imgTexto7’’/>
        <bind role=‘‘start’’ component=‘‘imgSair’’/>
</link>
<link xconnector=‘‘conEx#onKeySelectionSetNStopNStartN’’>
        <bind role=‘‘onSelection’’ component=‘‘imgSair’’>
                <bindParam name=‘‘tecla’’ value=‘‘BACK’’/>
        </bind>
        <bind role=‘‘stop’’ component=‘‘imgTexto7’’/>
</link>
<link xconnector=‘‘conEx#OnBeginStartNSetNStopN’’>
        <bind role=‘‘onBegin’’ component=‘‘videoPrincipal’’ interface=‘‘ancora8’’/>
        <bind role=‘‘start’’ component=‘‘imgLink8’’>
                <bindParam name=‘‘retardo’’ value=‘‘1.5s’’/>
        </bind>
        <bind role=‘‘stop’’ component=‘‘imgLink7’’/>
        <bind role=‘‘stop’’ component=‘‘imgTexto7’’/>
        <bind role=‘‘stop’’ component=‘‘ctxBarra’’/>
</link>
<link xconnector=‘‘conEx#onKeySelectionSetNStopNStartN’’>
        <bind role=‘‘onSelection’’ component=‘‘imgLink8’’>
                <bindParam name=‘‘tecla’’ value=‘‘BLUE’’/>
        </bind>
        <bind role=‘‘stop’’ component=‘‘imgLink8’’/>
        <bind role=‘‘start’’ component=‘‘ctxBarra’’/>
        <bind role=‘‘start’’ component=‘‘imgTexto8’’/>
        <bind role=‘‘start’’ component=‘‘imgSair’’/>
</link>
```

```
        <link xconnector=‘‘conEx#onKeySelectionSetNStopNStartN’’>
                <bind role=‘‘onSelection’’ component=‘‘imgSair’’>
                        <bindParam name=‘‘tecla’’ value=‘‘BACK’’/>
                </bind>
                <bind role=‘‘stop’’ component=‘‘imgTexto8’’/>
        </link>
        <link xconnector=‘‘conEx#OnBeginStartNSetNStopN’’>
                <bind role=‘‘onBegin’’ component=‘‘videoPrincipal’’ interface=‘‘ancora9’’/>
                <bind role=‘‘start’’ component=‘‘imgLink9’’>
                        <bindParam name=‘‘retardo’’ value=‘‘1.5s’’/>
                </bind>
                <bind role=‘‘stop’’ component=‘‘imgLink8’’/>
                <bind role=‘‘stop’’ component=‘‘imgTexto8’’/>
                <bind role=‘‘stop’’ component=‘‘ctxBarra’’/>
        </link>
        <link xconnector=‘‘conEx#onKeySelectionSetNStopNStartN’’>
                <bind role=‘‘onSelection’’ component=‘‘imgLink9’’>
                        <bindParam name=‘‘tecla’’ value=‘‘RED’’/>
                </bind>
                <bind role=‘‘stop’’ component=‘‘imgLink9’’/>
                <bind role=‘‘start’’ component=‘‘ctxBarra’’/>
                <bind role=‘‘start’’ component=‘‘imgTexto9’’/>
                <bind role=‘‘start’’ component=‘‘imgSair’’/>
        </link>
        <link xconnector=‘‘conEx#onKeySelectionSetNStopNStartN’’>
                <bind role=‘‘onSelection’’ component=‘‘imgSair’’>
                        <bindParam name=‘‘tecla’’ value=‘‘BACK’’/>
                </bind>
                <bind role=‘‘stop’’ component=‘‘imgTexto9’’/>
        </link>
</body>
</ncl>
```