# 2
# Trabalhos Relacionados

Este capítulo é dividido em seções focadas em cada assunto específico levantado no Capítulo 1, apresentando os trabalhos relacionados de relevância no desenvolvimento de um middleware para TVD, embora alguns deles não sejam diretamente propostos para esse fim. Neste capítulo os diversos trabalhos relacionados são apenas apresentados, deixando para o Capítulo 9 uma análise comparativa com as soluções propostas nesta tese, onde ficam evidenciadas as contribuições alcançadas.



## 2.1.
## Gerência de Recursos e Evolução Dinâmica

Diversos trabalhos na literatura têm apresentado soluções baseadas em componentes de software para prover um alto grau de adaptabilidade e um maior controle do uso de recursos computacionais.

Coulson et al. (Coulson, 2008) apresentam o OpenCom como um modelo de componentes de software para o desenvolvimento de sistemas de mais baixo nível, tais como sistemas de middleware, procurando prover as mesmas facilidades de reconfiguração que outras infra-estruturas de componentes provêem no nível das aplicações finais. Esse modelo de componentes já foi utilizado no desenvolvimento de sistemas de middleware para diferentes domínios de aplicação, tais como processadores de redes programáveis, sistemas de middleware de comunicação reflexivos e sistemas de roteamento em redes móveis *ad-hoc* (Ramdhany, 2009; Gomes, 2005).

Similarmente, o modelo de componentes Fractal (Bruneton, 2006) também tem sido utilizado na construção de sistemas de middleware. Por exemplo, Layaida et al. (Layaïda, 2005) apresentam um arcabouço baseado no modelo Fractal para a construção de aplicações multimídia, denominado PLASMA. Os autores mostram que sua arquitetura baseada em componentes pode prover

adaptabilidade em tempo de execução com um baixo impacto de desempenho, mesmo em dispositivos móveis com recursos computacionais limitados.

Finalmente, Filho et al. (Filho, 2007) especificam o modelo FlexCM, que possui como principal objetivo a composição automática de arquiteturas baseadas em componentes, através da representação explícita das conexões entre cada componente, utilizando um arquivo XML para a descrição arquitetural. Nessa descrição, as interfaces de componentes são identificadas através de GUIDs (*Globally Unique Identifiers*), que consistem em identificadores de 128 bits gerados de forma a garantir uma unicidade global. Cada componente pode declarar os GUIDs de suas interfaces providas e requeridas. Assim, um ambiente de execução conforme o modelo FlexCM poderá compor a arquitetura através da descrição XML. O middleware imperativo para sistemas TVD, denominado FlexTV (Filho, 2007), segue uma arquitetura conforme o modelo FlexCM.

Os middlewares declarativos para TVD não são discutidos nesta seção, pois, até onde sabemos, este é o primeiro trabalho que reporta a aplicação de uma arquitetura baseada em componentes de software na construção de um middleware declarativo para TVD, embora existam relatos de resultados no ambiente imperativo apresentado no parágrafo anterior (Filho, 2007).

O Capítulo 3 discute a arquitetura orientada a componentes especificada para o middleware Ginga-NCL e o modelo de componentes criado, bem como alguns detalhes de implementação do middleware.



## 2.2. Plano de Recuperação

Diversos trabalhos na literatura têm apresentado soluções para prover mecanismos de recuperação a falhas em seus sistemas, voltados para diferentes domínios.

Huang et al. (Huang, 1995) propõem um mecanismo de recuperação proativa, denominado rejuvenescimento de software, que consiste em periodicamente finalizar um sistema para imediatamente reiniciá-lo. Essa técnica parte do pressuposto que o estado inicial de um sistema é o estado mais correto e consistente em que ele se encontrará durante seu ciclo de vida. Trabalhos decorrentes do rejuvenescimento de software têm se concentrado na especificação

de políticas de rejuvenescimento, de forma a aumentar a disponibilidade do sistema e reduzir o custo do rejuvenescimento (Garg, 1998; Yujuan, 2003).

Técnicas de recuperação proativa são também utilizadas em sistemas distribuídos confiáveis (Castro, 2002; Souza, 2007). Castro e Liskov (Castro, 2002) descrevem um sistema de recuperação proativa para tolerância a falhas bizantinas. Nesse sistema, réplicas de servidores são periodicamente rejuvenescidas para eliminar os efeitos de ataques maliciosos e falhas do sistema. Sousa et al. (Souza, 2007) propõem uma abordagem complementar, que combina as técnicas de recuperação proativa com serviços que permitem que réplicas em estado normal atuem de forma reativa à detecção de falhas, recuperando réplicas comprometidas.

A solução proposta por Fugini e Mussi (Fugini, 2006) também combina técnicas de recuperação proativa com técnicas de recuperação reativa. Nessa solução é apresentada uma arquitetura para prover resiliência a serviços Web, utilizando mecanismos de detecção e recuperação de falhas ocorridas nesses serviços. Um ponto interessante nesse trabalho é que inconsistências nos dados de entrada (dados inseridos através de uma aplicação usuária dos serviços Web) são também consideradas falhas. Os mecanismos de recuperação a falhas definidos concentram-se na substituição de serviços Web e na recuperação da qualidade de dados de entrada.

Até onde sabemos o trabalho descrito nesta tese é o primeiro que reporta a aplicação de técnicas de recuperação a falhas para tornar resilientes apresentações de aplicações interativas em middlewares de TVD.

O Capítulo 4 apresenta uma arquitetura para recuperação a falhas, discute sua integração à arquitetura Ginga-NCL e as técnicas de recuperação aplicadas ao Ginga-NCL.

## 2.3. Comandos de Edição

Os principais sistemas de TVD permitem que aplicações sejam modificadas enquanto são exibidas, o que chamamos de edição ao vivo. Esses sistemas, o europeu DVB (*Digital Video Broadcast*), o americano ATSC (*Advanced Television System Committee*), o japonês ARIB (*Association of Radio Industries*

*and Businesses*) e o brasileiro ISDB-$T_B$, definem para suporte à autoria de aplicações interativas tanto linguagens declarativas quanto imperativas. Mais especificamente, todos os sistemas mencionados definem a linguagem Java para o suporte à autoria de aplicações imperativas e, com exceção do ISDB-$T_B$, uma linguagem baseada nos padrões XHTML, CSS e DOM para o suporte à autoria de aplicações declarativas, que podem utilizar ainda a expressividade da linguagem imperativa ECMAScript.

Nos três primeiros sistemas citados (europeu, americano e japonês), ao receber e interpretar uma aplicação declarativa, a máquina de apresentação constrói uma árvore DOM, cujos nós representam os elementos especificados em XHTML. O modelo DOM permite acesso dinâmico ao conteúdo dos nós, bem como atualizações de estilo, do conteúdo e da estrutura da árvore. Assim, alterações na aplicação podem ser feitas através de comandos externos, utilizando as funcionalidades do modelo DOM. Esses comandos podem modificar a estrutura da árvore DOM, preservando as intenções do autor para modificações na apresentação da aplicação e na sua estrutura lógica, uma vez que existe uma relação "um para um" entre os nós de uma árvore DOM e os elementos XHTML especificados na aplicação.

A edição ao vivo pode ser realizada através de eventos de sincronismo DSM-CC (detalhes sobre eventos de sincronismo podem ser encontrados no Anexo I), que podem ser mapeados em eventos DOM, desencadeando modificações na aplicação em um dado instante do tempo, em relação ao vídeo principal. No entanto, nesse caso, a relação criada através da edição é especificada entre o instante de tempo do vídeo principal e as entidades alteradas pela edição. Se a intenção do autor é estabelecer uma relação com qualquer outro objeto além do objeto de vídeo principal, essa semântica é perdida.

Diferente dos outros sistemas mencionados, o sistema nipo-brasileiro ISDB-$T_B$ tem como linguagem declarativa a linguagem NCL (*Nested Context Language*) (Soares, 2006). Em sua definição inicial (Costa, 2006b), os comandos de edição NCL foram modelados para realizar modificações nas estruturas de controle da apresentação derivadas da estrutura HTG[2] (Costa, 2006a), sem se preocupar com estruturas de exibição em si, como, por exemplo, com a identificação dos recursos

[2] HTG (*Hypermedia Temporal Graph*) é uma estrutura formada por dígrafos capaz de controlar o comportamento temporal das aplicações durante sua execução.

referenciados pela aplicação interativa ou indiretamente pelos próprios comandos de edição, como descrito na próxima seção.

O Capítulo 5 apresenta uma evolução da proposta inicial para os comandos de edição NCL. Essa evolução considera estruturas de exibição e a maturidade adquirida através do uso dos comandos de edição na implementação de referência Ginga-NCL.

## 2.4. Identificação de Recursos

Os sistemas de TVD (Morris, 2005), geralmente utilizam como formato para a multiplexação dos fluxos de áudio principal, vídeo principal e outros dados de um programa, definidos pelas aplicações de TVD, o fluxo de transporte MPEG-2, especificado no padrão MPEG-2 Sistemas (ISO, 2007). O Anexo I – Conceitos MPEG-2 – pode ser uma leitura complementar para um melhor entendimento dos trabalhos relacionados discutidos neste capítulo.

Na aplicação cliente (receptor de TVD) – servidor (provedor de conteúdo, por exemplo, estação radiodifusora) definida por um sistema de TVD, uma questão importante é a definição de um mecanismo para a identificação e localização dos recursos no ambiente do cliente, no ambiente do servidor e nas estruturas transmitidas. É desejável que tal mecanismo livre os autores das aplicações de conhecer como as estruturas de dados serão transmitidas e armazenadas no cliente, onde as aplicações serão exibidas, oferecendo uma abstração independente da localização dos seguintes recursos: conteúdo transportado em um carrossel de objetos, discutidos na Seção 2.4.1; fluxos elementares, discutidos na Seção 2.4.2; e bases temporais, discutidas na Seção 2.4.3.

Os sistemas DVB, ATSC e ARIB, mencionados na seção anterior, definem um esquema, um espaço de nomes e descritores, para que as aplicações de TVD identifiquem, de forma apropriada, os recursos recebidos nos dispositivos receptores, como identificados nas sub-seções seguintes.

### 2.4.1. Identificação de Recursos do Carrossel de Objetos

Inicialmente, na fase de concepção da aplicação, os recursos (arquivos de vídeo, de imagem, de texto, de áudio, de scripts etc.) utilizados para compor as aplicações de TVD estão localizados na máquina em que é feita a autoria, ou em sistemas de armazenamento (provedores de conteúdo) distribuídos em uma rede à qual o ambiente de autoria encontra-se conectado. Para identificar cada um dos recursos, geralmente são utilizadas URLs (*Universal Resource Locators*), cujos esquemas (RFC, 2005) identificam os recursos com base em suas localizações.

Para a transmissão assíncrona das aplicações, um protocolo, denominado carrossel de objetos (ISO, 1998), é usualmente utilizado. Uma das principais características do carrossel de objetos é permitir o envio cíclico de um sistema de arquivos, do provedor de conteúdo (ambiente de autoria) para os receptores (ambiente de apresentação). Assim, ao sintonizar um determinado canal, o receptor deve ser capaz de decodificar os dados recebidos e colocá-los em uma área de memória para que possam ser utilizados, preservando a estrutura de arquivos e diretórios enviada. Cabe observar que mais de um carrossel pode ser transmitido simultaneamente, e que um carrossel de objetos pode fazer uso de recursos (arquivos, diretórios e outros objetos (ISO, 1998)) que estejam sendo transferidos em outros carrosséis. Mais ainda, aplicações transferidas em arquivos de um carrossel podem fazer referência a recursos presentes no mesmo carrossel ou a recursos de outros carrosséis. Por exemplo, uma página HTML transmitida em um carrossel pode referenciar uma imagem cujo conteúdo é transmitido em outro carrossel de objetos.

Ao serem recebidos, os recursos precisam ter suas identificações reconhecidas pelas aplicações, já que eles possuirão uma nova localização no servidor, diferente daquela referenciada inicialmente pelas aplicações. Considerando também o fato de que um carrossel de objetos e aplicações transmitidas nesse carrossel podem referenciar recursos transmitidos em outros carrosséis, é preciso que haja uma forma de traduzir as identificações dos recursos no ambiente de autoria em identificações dos recursos no sistema de transmissão, e desse, para o ambiente de apresentação no cliente receptor. Como o carrossel de objetos preserva a estrutura do sistema de arquivos enviada, tal tradução pode ser

realizada, desde que o receptor saiba sob que diretório pai deve colocar a raiz do diretório recebido no carrossel. Sem informações adicionais, além daquelas enviadas no carrossel, tal diretório pai é desconhecido.

O sistema europeu DVB define um mecanismo denominado DVB *Locator* para que as aplicações imperativas (DVB-J) ou declarativas (DVB-HTML) possam identificar seus recursos, inclusive aqueles obtidos através do carrossel de objetos. Para esse tipo de identificação, a especificação do DVB *Locator* é realizada através do esquema de URL[3] "dvb" (ETSI, 2010).

Para que uma aplicação utilize recursos transportados no carrossel de objetos, a parte específica do esquema da URL deve conter os seguintes identificadores: rede de onde se origina o fluxo de transporte; fluxo de transporte que está sendo recebido através do canal sintonizado; serviço sintonizado; e, por fim, o nome do recurso desejado, nesse caso, o nome de um arquivo presente no carrossel de objetos. Por exemplo, a string "dvb://0x1A.0x2B.0x2C/imagens/logo.gif" identifica um arquivo chamado "logo.gif", presente no diretório "imagens", que é a raiz de um carrossel de objetos.

Sem outros expedientes adicionais, o autor de uma aplicação deveria referenciar todos os recursos não pela sua localização no ambiente de autoria, mas pela sua localização no sistema onde serão transportados, para que o receptor possa também entender essa localização. Como isso seria uma tarefa extremamente de baixo nível, árdua e susceptível a erros, para o autor de uma aplicação, outra construção é introduzida.

Para desobrigar o autor das aplicações interativas de conhecer a forma de representação que o DVB *Locator* especifica para os identificadores envolvidos no carrossel de objetos, o padrão do sistema europeu especifica um mecanismo que permite a descrição de forma simplificada das referências aos recursos utilizados nas aplicações interativas. No entanto, esse mecanismo está restrito à identificação de recursos de forma relativa a uma única raiz. Para isso, foram especificados um descritor para definir a localização de aplicações imperativas (DVB-J), o *dvb_j_application_location_descriptor*, e outro descritor para definir a

[3] A sintaxe básica de uma URL é: <esquema>:<parte específica do esquema> (RFC, 2005).

localização de aplicações declarativas (DVB-HTML), denominado *dvb_html_application_location_descriptor* (ETSI, 2010).

Entre os campos presentes no descritor *dvb_j_application_location_descriptor*, três destacam-se: *base_directory*, *classpath_extension* e *initial_class*. O primeiro deve conter uma string especificando um nome de diretório, sempre a partir da única raiz "/", que servirá como diretório base para caminhos relativos. Esse diretório é automaticamente inserido no caminho de busca por classes Java das aplicações interativas a serem chamadas. Caso o diretório base seja o diretório raiz, o campo deve conter apenas o valor "/". O segundo campo mencionado (*classpath_extension*), quando presente, deve possuir uma string especificando um nome de diretório relativo ao diretório base, para ser inserido no caminho de busca por classes Java das aplicações a serem chamadas. Finalmente, o campo *initial_class* deve conter uma string especificando a classe Java por onde a aplicação deve ser iniciada.

Os principais campos do descritor *dvb_html_application_location_descriptor* são: *physical_root* e *initial_path*. O campo *physical_root* deve conter uma string definindo o diretório raiz da aplicação. De forma análoga ao descritor para aplicações imperativas, caso o diretório base da aplicação seja o diretório raiz, o campo deve conter apenas o valor "/". O campo *initial_path* deve conter uma string definindo o caminho, relativo à raiz especificada em *physical_root*, para o documento declarativo que deverá iniciar a aplicação interativa. Por exemplo, suponha uma transmissão que possui um diretório "/aplicação/", contendo dois sub-diretórios: "imagens/" e "principal/". O sub-diretório "principal/", por sua vez, possui um documento "index.htm" e o sub-diretório "imagens" possui as imagens que servem de recursos para a aplicação declarativa. Nesse caso, o campo *physical_root* possui como conteúdo "/aplicação/" e o campo *initial_path* possui como conteúdo "principal/index.htm".

O sistema americano ATSC também possui uma forma para identificação de recursos nas especificações ACAP (ATSC, 2009). A URL definida é bem similar à do DVB *Locator*: "ocap://<sourceId>.<pid>[/<path>]". Note que o esquema é definido pela string "ocap", pois o padrão ACAP utiliza a forma de identificação definida nas especificações OCAP (CableLabs, 2005).

Na URL, “sourceId” identifica a rede de origem do fluxo de transporte, “pid” identifica o fluxo elementar em que se deseja identificar.

A parte final (“/path”) da URL é utilizada apenas quando se deseja identificar um recurso do carrossel de objetos. Além disso, o padrão ACAP (ATSC, 2009) define a mesma sintaxe e semântica do padrão europeu para os descritores responsáveis por localizar as aplicações imperativas e declarativas (*acap_j_application_location_descriptor* e *acap_x_application_location_descriptor*, respectivamente).

O sistema japonês ARIB especifica a identificação de forma idêntica ao padrão do sistema europeu, através de um ARIB *Locator*, com a mesma sintaxe e semântica. A única diferença fica por conta do esquema “arib-dc” para recursos presentes no carrossel de dados japonês (ARIB, 2004b). Para desobrigar o autor das aplicações interativas de conhecer os identificadores envolvidos no ARIB *Locator*, o padrão do sistema ARIB especifica o descritor *arib_j_application_location_descriptor* (ARIB, 2004a) que possui a mesma sintaxe e semântica do descritor *dvb_j_application_location_descriptor*. O padrão do sistema japonês, no entanto, não especifica esse tipo de descritor para aplicações declarativas. Assim, para identificar os recursos obtidos através do carrossel de objetos, as aplicações declarativas devem utilizar a URL conforme especificação do ARIB *Locator* (ARIB, 2004b).

Os mecanismos especificados nos três sistemas definem como identificar recursos recebidos pelo carrossel DSM-CC nos receptores (ambiente de apresentação), no entanto, com uma série de limitações. A partir dos mecanismos especificados, são duas as abordagens existentes para traduzir a identificação dos recursos no ambiente de autoria em uma identificação dos recursos no ambiente de apresentação.

A primeira abordagem é fazer uso apenas dos *Locators*, devendo os autores de aplicações, como já mencionado, aprender os detalhes do mecanismo de identificação de recursos no carrossel de objetos e detalhes do protocolo DSM-CC, para referenciarem os recursos nas estruturas do sistema de transporte para que, após, as referências possam ser traduzidas nas URLs adequadas do sistema de armazenamento utilizadas no ambiente de apresentação do sistema de TVD específico. Ou, alternativamente, que todas as URLs especificadas em uma aplicação no ambiente de autoria sejam automaticamente traduzidas para URLs de

*Locators* identificando recursos nos carrosséis de objetos por algum processo executado durante a geração do carrossel. Além de ser um processo de baixo nível de abstração e sujeito a erros, note que, caso as aplicações sejam armazenadas no cliente (receptor), para possível exibição futura, um novo sistema de arquivos deve ser gerado (nesse caso, um sistema de arquivos criado no dispositivo de armazenamento do receptor), que não tem relação alguma com o sistema de arquivos original, pois esse relacionamento foi perdido quando da tradução para as URLs dos carrosséis de objetos. Assim, uma atualização futura desse conteúdo pelo servidor seria muito difícil de ser realizada ou mesmo impossível.

Uma segunda abordagem é baseada no uso adicional do mecanismo de descritores, definidos nos padrões dos três sistemas, conforme anteriormente descritos. Descritores gerados, por exemplo, com auxílio de ferramentas gráficas, podem ser multiplexados ao fluxo de transporte, permitindo manter no cliente as mesmas referências utilizadas no servidor, uma vez que os descritores poderiam servir de ponte para uma tradução.

No entanto, ao definir nos descritores um diretório base para caminhos relativos e especificar que seu valor deve ser "/", caso seja o diretório raiz, as especificações estão impossibilitando o uso de referências absolutas para objetos transmitidos no carrossel. Por exemplo, sem alterações nos identificadores de recursos, o mecanismo não contempla a transmissão de aplicações armazenadas no ambiente de autoria que referenciem conteúdos dispostos em dispositivos de armazenamento compartilhados da rede; ou recursos dispostos em discos ou partições diferentes de uma mesma máquina, mesmo que esses recursos estejam sendo transmitidos em outro carrossel de objetos.

Como é natural no ambiente de autoria as aplicações referenciarem recursos de forma absoluta, para que o autor não se preocupasse com URLs como as definidas pelo DVB *Locator*, seria necessário sobrecarregar o ambiente de autoria de forma a permitir que o mesmo fosse capaz de traduzir as URLs absolutas em URLs relativas, de acordo com as definições dos descritores. Para tanto, uma nova versão da aplicação interativa deveria ser criada. Uma versão que possui identificadores de recursos que só fazem sentido no ambiente de apresentação. Por sobrecarregar o ambiente de autoria e depender do autor para que as informações dos descritores sejam preenchidas, essa abordagem pode introduzir um retardo no processo de geração do carrossel de objetos, o que pode prejudicar o

relacionamento temporal das aplicações interativas, principalmente aqueles com conteúdos gerados ao vivo.

Uma gerência de localização de recursos mais apropriada é definida no protocolo FLUTE (RFC, 2004), comumente utilizado em sistemas IPTV. Nele, uma tabela denominada FDT (*File Description Table*), especificada normalmente por meio de um arquivo XML, provê toda a informação necessária para a identificação, localização e recuperação de arquivos. A localização se dá por meio de URI's, que podem ser usadas diretamente no receptor cliente para acesso ao sistema de arquivos montado. A FDT deve ser enviada antes dos arquivos, porém, suas informações não precisam estar completas e podem ser modificadas de forma incremental. Apesar de não prover abstrações mais estruturadas que arquivos, FLUTE permite que hierarquias de sistemas de arquivos sejam montadas, ao definir que as referências a arquivos sejam feitas por URIs. Diferente dos sistemas de TVD por difusão anteriormente citados, diferentes raízes podem ser adotadas, com o uso de URIs absolutas. A proposta de solução apresenta nesta tese e adotada no Ginga-NCL, aproxima-se da solução adotada em FLUTE, no entanto, apresentando um maior refinamento na gerência, a possibilidade de uso de diferentes estruturas de transporte e a possibilidade de uso de diferentes sistemas de transporte, incluindo o próprio protocolo FLUTE.



### 2.4.2. Identificação de Fluxos Elementares

Para permitir a uma aplicação identificar fluxos elementares (áudio, vídeo ou texto) multiplexados no fluxo de transporte a parte específica do esquema deve ser especificada com os seguintes identificadores: rede de onde se origina o fluxo de transporte; fluxo de transporte que está sendo recebido através do canal sintonizado; serviço sintonizado (para informações, veja Anexo I); e, por fim, o identificador de componente (*component_tag*) atribuído por um descritor de componentes (ETSI, 2010) ao fluxo elementar desejado. Por exemplo, a URL "dvb://0x1A.0x2B.0x2C.0x87" referencia o fluxo elementar identificado com a *component_tag* 0x87, que faz parte do serviço 0x2C, presente no fluxo de transporte 0x2B da rede 0x1A. Como pode ser observado na URL, os

identificadores são separados por ponto e devem ser especificados como valores hexadecimais (ETSI, 2010).

Com o objetivo desobrigar o autor das aplicações interativas de conhecer os detalhes de representação que o DVB *Locator* especifica para os fluxos elementares multiplexados no fluxo de transporte, o padrão do sistema europeu especifica algumas palavras reservadas que permite a descrição de forma simplificada das referências aos fluxos elementares, no que o padrão chama de TV *Locators*. No entanto, as palavras reservadas foram especificadas apenas para alguns recursos, como apresentado na Tabela 1.

| DVB Locator | Semântica |
|---|---|
| dvb://current | Serviço atual. Pode ter sido sintonizado pela aplicação. |
| dvb://current.av | Conteúdo audiovisual principal sendo apresentado pelo dispositivo receptor. |
| dvb://current.audio | Áudio principal sendo apresentado pelo dispositivo receptor. |
| dvb://current.video | Vídeo principal sendo apresentado pelo dispositivo receptor. |
| dvb://original | Serviço que estava sintonizado quando a apresentação da aplicação foi iniciada |

Tabela 1 – TV Locators especificados pelo padrão DVB (ETSI, 2010)

Por exemplo, considere que o fluxo elementar identificado com a *component_tag* 0x87 do exemplo acima seja do vídeo principal sendo apresentado. Nesse caso, o autor da aplicação interativa poderia utilizar o TV *Locator* "dvb://current.video", sem precisar conhecer os identificadores da estrutura de transporte.

As mesmas palavras reservadas, com suas respectivas semânticas, apresentadas na Tabela 1, foram definidas nas especificações ACAP para facilitar a identificação de fluxos elementares.

As palavras oferecidas ao autor possuem importantes restrições como não poder referenciar fluxos elementares de texto e nem mesmo poder referenciar outros serviços ao especificar URLs com as palavras reservadas. A Seção 6.2 apresenta a solução proposta para o Ginga-NCL.

### 2.4.3. Identificação de Bases Temporais

Uma aplicação interativa pode conter objetos de mídia de diferentes tipos, como texto, imagem, vídeo, áudio etc. Em comum, esses objetos devem obedecer às relações de sincronismo especificadas pelo autor, tanto em relação ao tempo de exibição, quanto ao espaço de exibição nos dispositivos utilizados para apresentação.

A especificação do sincronismo entre os diferentes objetos de mídia, usualmente denominado sincronismo intermídia, pode ser realizada em relação a intervalos temporais absolutos de um eixo temporal (ISO, 2007). Porém, essa abordagem só é adequada quando os instantes temporais do sincronismo entre as mídias são deterministicamente conhecidos em tempo de especificação da aplicação. Esse é o caso, por exemplo, do sincronismo entre o áudio e o vídeo que compõem o conteúdo audiovisual de um fluxo de transporte MPEG-2 (MPEG-2 *Transport Stream* (ISO, 2007)).

Nas aplicações onde o sincronismo depende da ocorrência de eventos [4] com duração variável ou mesmo imprevisível no momento da especificação, é imperativo que essa especificação seja realizada de forma relativa à ocorrência desses eventos, independente do momento temporal em que eles ocorrem e se de fato eles irão ocorrer. Esse é o caso, por exemplo, das aplicações, onde é impossível prever o momento exato da interação do usuário.

Durante a execução das aplicações, os exibidores devem reportar todos os eventos, incluindo os não determinísticos, para que outros eventos relacionados possam ser disparados. O tempo de exibição (*media time*) individual de cada objeto de mídia deverá ser controlado, uma vez que os eventos podem estar associados a trechos específicos de seu conteúdo. Esses trechos (âncoras de conteúdo) podem ser definidos sobre intervalos temporais dos objetos. Como exemplo, uma aplicação pode especificar que quando o usuário interagir em um momento temporal específico da exibição de um filme (âncora temporal), uma propaganda, por exemplo outro vídeo, será exibida.

Para os conteúdos de mídia que estejam previamente disponíveis junto aos exibidores, o controle do tempo de exibição pode ser realizado simplesmente

[4] Nesta tese um evento corresponde a qualquer ocorrência no tempo instantânea, como o clique em um botão do controle remoto, ou com uma duração, como a apresentação de um objeto de vídeo.

verificando o tempo transcorrido a partir do início de sua exibição. Além disso, estando os conteúdos previamente disponíveis, seu tempo total de duração (*total media time*) pode ser trivialmente calculado. Ao contrário, para conteúdos entregues em tempo de exibição através de fluxos contínuos (*streaming*), determinar o instante inicial do conteúdo, ou seu tempo total de exibição, exige que o receptor esteja sintonizado desde o início do fluxo, ou então, que mecanismos adicionais sejam disponibilizados.

Referências temporais absolutas, que vão de zero ao "infinito" (valor muito grande), correspondendo, respectivamente, ao tempo inicial e final de um fluxo de mídia contínua, são usualmente multiplexadas ao fluxo. Como exemplo, podem ser citados o *Presentation Time Stamps* (PTS) e o *Presentation Clock Reference* (PCR) definidos no padrão MPEG-2 sistemas (ISO, 2007)[5]. No entanto, esse tipo de referência temporal não é suficiente como mecanismo adicional mencionado no parágrafo anterior. A razão é que um conteúdo já codificado pode ser editado, causando descontinuidade nas referências temporais. Mais ainda, ao longo do tempo, um mesmo fluxo pode conter inúmeros conteúdos distintos, muitas vezes até entrelaçados. Ou seja, o conteúdo do fluxo contínuo pode estar semanticamente relacionado a objetos de mídia diferentes com o passar do tempo. Como exemplo, pode ser citado o fluxo de um canal de uma emissora qualquer em um sistema de TVD por difusão terrestre. Durante o transcorrer da programação, diferentes conteúdos (diferentes objetos de mídia), que correspondem a diferentes programas da emissora, são transmitidos sem interrupção do fluxo audiovisual principal, e sem alterações das referências temporais citadas, que possam indicar o fim de um conteúdo específico ou que um novo conteúdo está iniciando sua transmissão. Além disso, usualmente, um conteúdo específico (programa) é interrompido diversas vezes para a exibição de comerciais, que do ponto de vista do sincronismo das aplicações hipermídia, podem ser considerados novos conteúdos (novos objetos de mídia).

O controle do tempo individual de conteúdos distintos, transmitidos através de um mesmo fluxo contínuo, exige o uso de diferentes bases temporais, cada uma correspondente ao tempo de exibição individual de cada conteúdo. Cada base

[5] Enquanto o PTS é utilizado para ajustar o relógio temporal dos clientes em relação ao relógio das estações transmissoras, preservando o sincronismo do conteúdo audiovisual quando ele é decodificado, o PCR é utilizado para controlar a taxa de decodificação do conteúdo, evitando *underflow* e *overflow* no *buffer* dos transmissores e receptores.

temporal deve ser mantida (multiplexada) junto ao fluxo e deve ser possível associar cada base ao conteúdo específico a que ela se refere. Quando mais de um conteúdo é transmitido de forma intercalada, operações de pausa e retomada sobre os objetos de mídia que representam esses conteúdos devem ser realizadas, tendo como referência a troca da base temporal. Pausar um objeto de mídia corresponde a pausar sua base temporal e vice-versa. Assim, uma base temporal pode ser pausada e, posteriormente, reiniciada a partir do instante em que foi pausada. Eventualmente, a pausa em um conteúdo e a sua retomada pode acontecer em algum instante temporal diferente. Como exemplo, quando o usuário troca de canal e em seguida retorna ao canal anterior, a exibição do conteúdo deve ser retomada a partir de um instante temporal posterior àquele em que a apresentação foi interrompida.

O uso de bases temporais permite que o tempo de exibição (*media time*) de cada um dos diferentes conteúdos existentes em um fluxo contínuo seja controlado. No entanto, é necessário definir como as aplicações fazem referência a esses conteúdos (ou a trechos desse conteúdo), que do ponto de vista do autor da aplicação são observados como um único fluxo (por exemplo, o vídeo principal de um canal de TVD).

Nos sistemas DVB, ATSC, ARIB e SBTVD as bases temporais dos diversos conteúdos de um fluxo elementar são multiplexadas, de forma idêntica à multiplexação dos conteúdos que compõem o fluxo, compondo o que se denomina o NPT (*Normal Play Time*). Esses valores são transportados em estruturas de dados denominadas descritores de referência NPT (NPT *Reference Descriptor*) (ISO, 1998). Um descritor NPT contém um campo *contentId* identificando a que conteúdo de um fluxo ele se refere e o valor de sua base temporal no momento.

Utilizando o NPT, o controle do sincronismo nesses sistemas pode ser realizado de duas formas principais:

- enviando aos receptores eventos de sincronismo DSM-CC (ISO, 1998);
- monitorando constantemente os valores NPT recebidos.

No primeiro caso, os objetos do carrossel de objetos são utilizados para atrelar os eventos de sincronismo à base temporal. Um carrossel de objetos pode transportar um objeto do tipo evento de fluxo (*ste*) (ISO, 1998). Esse objeto é utilizado para definir tipos de eventos DSM-CC possíveis de serem descritos no

fluxo de transporte. Para isso, o objeto relaciona identificadores de descritores de eventos DSM-CC a uma string (*event_name*). Assim, a aplicação pode requisitar ao receptor ser notificada sobre a ocorrência de tipos de eventos específicos, enviados pelo provedor de conteúdo. O objeto do tipo evento de fluxo é fundamental na definição da associação de um evento DSM-CC a uma base temporal (*contentId*), por meio de uma estrutura chamada *Tap* (ISO, 1998). A partir da associação do evento DSM-CC à base temporal, o provedor de conteúdo pode enviar eventos DSM-CC com instantes NPT para a ocorrência desses eventos.

Nos sistemas DVB, ATSC e ARIB, a identificação da base temporal, nesse caso, não é realizada diretamente na aplicação. O acesso à base temporal é realizado no tratamento do evento DSM-CC, realizado pelo middleware. Para especificar o sincronismo entre a aplicação e a base temporal, o autor precisa especificar mecanismos para monitorar os eventos DSM-CC. A ocorrência desses eventos é que acarreta nas mudanças de comportamento desejadas na aplicação.

O recebimento dos eventos DSM-CC pelas aplicações XHTML, especificadas nos três sistemas mencionados no parágrafo anterior, só é possível através de mapeamento de eventos DSM-CC em eventos DOM (ETSI, 2010; ATSC, 2009; ARIB, 2004b). A partir de um evento DOM, funções ECMAScript devem ser utilizadas para realizar as mudanças de comportamento desejadas. Para as aplicações Java, algumas APIs (ETSI, 2010; ATSC, 2009; ARIB, 2004a) são definidas para o acesso aos eventos DSM-CC. De forma similar às aplicações declarativas, a ocorrência dos eventos DSM-CC podem também disparar procedimentos imperativos para alterações de comportamento das aplicações, sincronizando assim as aplicações com as bases temporais recebidas.

No segundo caso, as aplicações se cadastram como “ouvintes” dos valores NPT. Isso envolve o processamento do fluxo de transporte, com o intuito de extrair os descritores NPT.

Nos três sistemas (europeu, americano e japonês), são especificadas APIs Java para que os valores da base temporal sejam obtidos, através de constantes consultas aos descritores NPT recebidos. O descritor recebido pela aplicação deve ter o campo *contentId* consultado para que a aplicação consiga distinguir as diferentes bases temporais multiplexadas no fluxo NPT.

Note que nenhum suporte declarativo é oferecido para o acesso às bases temporais, nem mesmo para a especificação do sincronismo das aplicações.

O Capítulo 6 apresenta alternativas para localização de recursos, incluindo identificação de recursos do carrossel de objetos, fluxos elementares e bases temporais.

## 2.5. Ciclo de Vida das Aplicações NCL

Nos sistemas DVB, ATSC e ARIB, o controle do ciclo de vida das aplicações é realizado através de comandos enviados em uma estrutura denominada AIT (*Association Information Table*) (Morris, 2005).

Aplicações podem ser iniciadas, paradas ou abortadas, dependendo do comando enviado na AIT. A Tabela 2 apresenta os comandos de controle que podem ser enviados em uma tabela AIT para aplicações declarativas DVB-HTML.

| Comando | Semântica |
|---|---|
| AUTOSTART | A aplicação deve ser interpretada e ter sua apresentação iniciada imediatamente. |
| PRESENT | Indica que existe uma aplicação no serviço sintonizado, mas essa aplicação não será iniciada automaticamente. Ela será disponibilizada em uma lista de aplicações que podem ser iniciadas pelo usuário. |
| DESTROY | Quando o comando é alterado de AUTOSTART ou PRESENT para DESTROY, a máquina de apresentação deve destruir a aplicação e ir para o estado Destroyed (ETSI, 2010). |
| KILL | Quando o comando é alterado de AUTOSTART ou PRESENT para KILL, a máquina de apresentação deve finalizar a aplicação e ir para o estado Killed (ETSI, 2010). |
| PREFETCH | Possui o mesmo comportamento que AUTOSTART. No entanto, neste comando, a máquina de apresentação aguarda uma instrução para iniciar a apresentação da |

| | |
|---|---|
| | aplicação. |
| REMOTE | Indica uma aplicação remota que só pode ter sua apresentação iniciada após a sintonia do serviço que ela é transmitida. |

Tabela 2 – Comandos para controle do ciclo de vida de aplicações DVB-HTML (ETSI, 2010)

O instante em que a apresentação é iniciada ao receber um comando AUTOSTART depende da plataforma de recepção (ETSI, 2010). Antes de iniciar a apresentação de uma aplicação, o gerente de aplicações definido (ETSI, 2010) certifica-se que nenhuma aplicação com o mesmo identificador de aplicação (informação enviada pela AIT) já esteja instanciada. Caso essa operação seja bem sucedida, o gerente de aplicações cria um novo *user agent* para apresentar a aplicação.

A principal diferença entre os comandos KILL e DESTROY é que o *user agent* criado e todos os recursos da aplicação são destruídos quando o comando KILL é recebido, o que não ocorre para o comando DESTROY. Quando o comando DESTROY é recebido, os recursos da aplicação são liberados, mas ainda é possível apresentar a aplicação com o *cache* dos recursos realizado pelo *user agent* (ETSI, 2010).

Quando o comando PREFETCH é recebido, o gerente de aplicações aguarda outro comando para iniciar a apresentação da aplicação. Esse comando é recebido através de um evento de sincronismo DSM-CC, denominado *trigger* (ETSI, 2010), que define o instante de tempo que a apresentação deve ser iniciada.

Note que uma aplicação não pode ser pausada e retomada por um comando da AIT. Essa limitação pode ser resolvida através de comandos de edição (baseados em eventos DSM-CC, como o *trigger*) propostos nesta tese, que permitem maior acuidade no controle das aplicações, como discutido no Capítulo 5.

No Capítulo 7 é discutido como os comandos de edição realizam o controle do ciclo de vida das aplicações NCL

## 2.6. Sistemas de Transporte

Conforme discutido na Seção 2.4 para o formato de transporte de uma aplicação (sua especificação e os conteúdos por ela referenciados), os sistemas DVB, ATSC e ARIB fazem uso de carrosséis DSM-CC, além dos fluxos elementares associados aos serviços de um canal. O padrão DSM-CC especifica dois tipos de carrosséis: o carrossel de dados, usado pelo ARIB, e o carrossel de objetos, utilizado pelos sistemas DVB e ATSC.

Alguns problemas sobre identificação dos recursos do carrossel de objetos foram discutidos na Seção 2.4. No entanto, outra questão que merece atenção é o overhead computacional e de transmissão do carrossel de objetos. O envelopamento de arquivos e diretórios é realizado em várias camadas e mensagens auxiliares são especificadas para informar detalhes do carrossel como, por exemplo, tamanho e versão das estruturas transportadas (ISO, 1998).

Ainda outro problema relacionado aos carrosséis é o controle de versão das estruturas transportadas. Uma incoerência pode ocorrer quando uma estrutura for atualizada ao mesmo tempo em que um cliente receptor está com uma montagem do carrossel em andamento. Se o cliente detectar que a versão de uma determinada estrutura não corresponde com a versão informada nas mensagens auxiliares, a montagem deve ser abortada (ISO, 1998). Outras incoerências podem ocorrer, porém com maior grau de dificuldade de detecção, devido a referências distribuídas, possivelmente presentes em uma aplicação, fazendo com que atualizações no carrossel de dados, bem como no carrossel de objetos, sejam consideradas tarefas com nível de dificuldade elevado.

O sistema ARIB define uma alternativa ao carrossel de objetos, especificando uma abstração própria para o carrossel de dados DSM-CC. Nessa abstração, os descritores de localidades são enviados em mensagens auxiliares do carrossel de dados, que transportam também informações sobre o sistema de arquivos a ser criado no receptor cliente. Porém, além de apresentar o mesmo ponto crítico anteriormente mencionado dos descritores de localidades do carrossel de objetos, e a dificuldade de realizar atualizações, herdada do carrossel de dados, existe a restrição de no máximo quatro níveis na hierarquia de diretórios na abstração especificada pelo sistema japonês (ARIB, 2004b).

Sistemas IPTV podem utilizar a multiplexação MPEG-2 para o fluxo de transporte, incluindo a codificação de dados DSM-CC. No entanto, redes de distribuição IPTV são tipicamente *multicast* e apresentam características diferenciadas, como o gerenciamento de qualidade de serviço, a possibilidade de ocorrência de congestionamentos e de erros de transmissão.

Em sistemas IPTV é comum a adoção do protocolo de transporte RTP (RFC, 2003). Nesse protocolo, fluxos de áudio e vídeo podem ser enviados em diferentes sessões RTP, cada qual possuindo suas marcações de tempo, usadas para sincronizar a apresentação no receptor.

Para o transporte de dados sob demanda em serviços assíncronos (a especificação da aplicação e seus conteúdos), pode-se adotar qualquer protocolo de aplicação baseado em IP, sendo o HTTP o caso mais comum. Na transmissão de dados sem solicitação (*pushed*) é necessário o uso de um protocolo cíclico, que pode ser o carrossel de objetos DSM-CC, ou outro protocolo mais direcionado às características da rede IPTV, como o protocolo FLUTE (RFC, 2004).

O protocolo FLUTE oferece transporte de arquivos em modo *push*, tanto em *multicast* quanto *unicast*, e é muito usado em sistemas IPTV, sendo também adotado pelo sistema DVB-H (ETSI, 2009) e 3GPP MBMS (ARIB, 2009). FLUTE suporta correção de erros baseada em objetos FEC (*Forward Error Correction*) e controle de congestionamento baseado no transporte de objetos do protocolo ALC (*Asynchronous Layered Coding*). Como discutido na Seção 2.4, FLUTE utiliza a tabela FDT para prover toda a informação necessária para a identificação, localização e recuperação de arquivos. Para prover repetições como um carrossel, a FDT (e os arquivos que refere) é enviada periodicamente em uma mesma sessão FLUTE, especificando os mesmos arquivos. Quando é necessária a atualização, modificação ou criação de um novo carrossel, uma nova FDT é enviada pela mesma sessão FLUTE, com as características do novo carrossel.

Conforme discutido na Seção 2.3, o suporte à edição ao vivo de aplicações é baseado em eventos DSM-CC. Os eventos DSM-CC são estruturas que contêm um identificador e um valor de referência temporal para sua ocorrência, que pode ser o do momento de sua recepção (eventos conhecidos como *do-it-now*).

Além do identificador e da referência temporal, a estrutura de um evento DSM-CC possui um campo para dados privados que podem ser usados de acordo com uma semântica definida pela aplicação tratadora do evento. Esse campo pode

atingir o tamanho máximo de 255 bytes, restringindo assim a possibilidade da definição de semânticas mais elaboradas. Outro detalhe interessante do uso de eventos DSM-CC nesses sistemas está no fato do ambiente de recepção não garantir a recepção de todos os eventos recebidos (Morris, 2005; ISO, 1998). Para evitar que um evento DSM-CC seja perdido, ele é enviado diversas vezes pelo provedor de conteúdo, cabendo ao cliente receptor interpretar a possível duplicação de recepção.

FLUTE, usado em sistemas IPTV como já mencionado, não oferece a abstração de estruturas, como os eventos DSM-CC, que possa ser usada para o encapsulamento de comandos de edição. Obviamente, em sistemas IPTV, uma representação de estruturas auxiliares pode ser definida de forma proprietária, como conteúdo de um dos arquivos do carrossel FLUTE.

Estruturas de dados mais leves para transporte de comandos de edição, de arquivos de dados e de metadados para suporte à identificação de recursos são apresentadas no Capítulo 8.