6
Identificando Recursos nas Aplicações NCL

Como acontece com qualquer sistema distribuído, em um sistema de TVD, os equipamentos dos provedores de conteúdo e os equipamentos receptores devem compartilhar de protocolos bem definidos para que o intercâmbio de informações e a provisão de serviços sejam possíveis. Este capítulo discute os mecanismos de identificação de recursos para que as aplicações de TVD especificadas para o ambiente declarativo do middleware Ginga identifiquem, de forma apropriada, os recursos recebidos nos dispositivos receptores. Esse mecanismo é capaz de livrar 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 para a localização dos recursos.

6.1.
Carrossel de Objetos

O mecanismo proposto para localização de recursos em um carrossel de objetos será explicado nesta seção tomando como base a implementação de referência do ambiente declarativo do SBTVD, bem como a implementação de um servidor *datacasting* para esse mesmo sistema.

Na implementação do servidor *datacasting*, as informações de identificação de recursos das aplicações declarativas são transmitidas em estruturas (detalhes sobre todas as estruturas de transporte necessárias são discutidos no Capítulo 8) contendo comandos de edição NCL. Como apresentado no Capítulo 5, entre esses comandos, foram definidos um comando para carregar a aplicação, denominado *addDocument*; e um comando para adicionar estruturas lógicas (que podem possuir referências para recursos a serem apresentados pela aplicação) a uma aplicação já carregada, denominado *addNode*. Essas estruturas são especificadas através de um documento XML (veja Capítulo 5). São exatamente esses os comandos usados para dar suporte ao mapeamento das estruturas de dados usadas no servidor e no cliente receptor.

Nesses comandos, um conjunto de um ou mais pares {*uri*, *id*} é transmitido como parâmetro. No primeiro par, a *uri* possui o esquema "x-sbtvd", definido para o padrão brasileiro de TVD, seguido do caminho (no ambiente de autoria) absoluto de uma aplicação interativa ou de um documento XML (contendo especificações de estruturas lógicas a serem adicionadas a uma aplicação já carregada), caso o recurso seja transmitido por carrossel. Nesse caso, um *id* deve referenciar o identificador do objeto (*ior*, veja Capítulo 8) que transporta o recurso no carrossel de objetos. Caso contrário, a *uri* possui um esquema qualquer que não seja "x-sbtvd" e o *id* igual a "NULL".

Para contemplar todas as referências utilizadas pela aplicação (caso do comando *addDocument*) ou pelo documento XML (caso do comando *addNode*), sistemas de arquivos adicionais podem ter de ser transmitidos em outros carrosséis de objetos. Nesse caso, outros pares {*uri*, *id*} devem ser adicionados ao conjunto transmitido como parâmetro do comando de edição especifico. A *uri* deve então possuir o esquema "x-sbtvd", seguido do caminho (no ambiente de autoria) absoluto da raiz do sistema de arquivos e o *id* deve referenciar o identificador (*ior*) do objeto do carrossel de objetos que representa o conteúdo desse recurso. Normalmente, essa necessidade existe quando a estrutura dos sistemas de arquivos que representa a aplicação não pode ser convenientemente representada por uma única árvore.

Todo esse processo é realizado automaticamente pelo servidor *datacasting*, sem necessidade da intervenção do autor da aplicação, a quem cabe apenas o papel de criá-la e especificar o seu sistema de arquivos (a ser transmitido por carrosséis de objetos) através de referências a serviços de armazenamento localmente conhecidas.

Nos receptores, o middleware, ao receber o comando, percorre o conjunto de pares {*uri*, *id*} e guarda em uma estrutura de dados cada *uri* transmitida, relacionando-a com a *uri* da área de memória onde o objeto de *id* (*ior*) relacionado foi montado. A partir daí, para resolver as *uris* existentes na aplicação interativa, o middleware consulta essa estrutura de dados, fazendo o mapeamento necessário para encontrar os recursos na área de memória do dispositivo. Note assim que as *uris* presentes na aplicação do lado cliente são idênticas àquelas presentes do lado servidor, sendo automaticamente traduzidas (mapeadas) quando qualquer referência é realizada. Note também que em um comando *addDocument*, se todos

os recursos da aplicação estiverem sob uma mesma raiz no sistema de arquivos, o parâmetro id do par {*uri, id*} pode ser omitido.

O mecanismo discutido pode ser estendido para outras alternativas de transporte de aplicações, como discorrido no Capítulo 8, que apresenta ainda exemplos de uso desse mecanismo. O Capítulo 7 evidencia como o mecanismo proposto resolve os problemas encontrados nos outros mecanismos discutidos no Capítulo 2.

6.2. Fluxos Elementares

Uma aplicação NCL pode identificar fluxos elementares por meio da seguinte uri: sbtvd-ts://*program_number.component_tag*. Na uri, o *program_number* identifica o serviço do fluxo de transporte que está sendo recebido através do canal sintonizado e o *component_tag* identifica o fluxo elementar do serviço por meio de uma *tag* atribuída por um descritor de componentes (ABNT, 2008a).

Um ponto importante na atribuição dessas *tags* é que intervalos de valores são reservados para cada tipo de fluxo elementar (vídeo, áudio ou texto). Além disso, a menor *tag* atribuída a cada tipo deve ser definida para o fluxo *default* (ABNT, 2008a). Por exemplo, no SBTVD (ABNT, 2008a) para os fluxos elementares de vídeo, o intervalo de possíveis valores para *tags* é de 0x00 a 0x0F. O fluxo elementar de vídeo identificado com a menor *tag* será o decodificado pelo receptor quando o usuário sintonizar o canal de TVD.

Com o objetivo desobrigar o autor das aplicações interativas de conhecer os detalhes de representação para os fluxos elementares multiplexados no fluxo de transporte, as palavras reservadas, apresentadas na Tabela 7, foram definidas.

URL	Semântica
sbtvd-ts://i.video	Fluxo elementar de vídeo de menor *component_tag* no i-ésimo menor serviço. Caso i não seja especificado, o serviço consiste o mesmo em que a aplicação foi multiplexada.
sbtvd-ts://i.audio	Fluxo elementar de áudio de menor *component_tag* no i-ésimo menor serviço. Caso i não seja especificado, o serviço consiste o mesmo em que a aplicação foi multiplexada.
sbtvd-ts://i.text	Fluxo elementar de texto de menor *component_tag* no i-ésimo menor serviço. Caso i não seja especificado, o serviço consiste o mesmo em que a aplicação foi multiplexada.
sbtvd-ts://i.video(j)	Fluxo elementar de vídeo que possui o j-ésimo menor *component_tag* no i-ésimo menor serviço. Caso i não seja especificado, o serviço consiste o mesmo em que a aplicação foi multiplexada. O valor de j deve ser diferente de 0 (valor 0 é reservado para sbtvd-ts://i.video).
sbtvd-ts://i.audio(j)	Fluxo elementar de áudio que possui o i-ésimo menor *component_tag* no i-ésimo menor serviço. Caso i não seja especificado, o serviço consiste o mesmo em que a aplicação foi multiplexada. O valor de j deve ser diferente de 0 (valor 0 é reservado para sbtvd-ts://i.audio)
sbtvd-ts://i.text(j)	Fluxo elementar de texto que possui o i-ésimo menor *component_tag* no i-ésimo menor serviço. Caso i não seja especificado, serviço consiste o mesmo em que a aplicação foi multiplexada. O valor de j deve ser diferente de 0 (valor 0 é reservado para sbtvd-ts://i.text)

Tabela 7 – URLs para Identificação de Fluxos Elementares

A partir das URLs apresentadas na Tabela 7, uma aplicação NCL pode especificar de forma simples a navegação entre fluxos elementares, como exemplificado no trecho da aplicação NCL apresentado no Quadro 4.

```
                                    ...
1:  <media id="a1" src="sbtvd-ts://audio">
2:    <area id="a1synch1" first="5npt" last="15npt"/>
3:    <property name="soundLevel" value="1"/>
4:  </media>
5:  <media id="a2" src="sbtvd-ts://audio(1)">
6:    <area id="a2synch1" first="5npt" last="15npt"/>
7:    <property name="soundLevel" value="0"/>
8:  </media>
                                    ...
```

Quadro 4 – Trecho de Aplicação NCL com Identificadores de Fluxos Elementares.

No Quadro 4, dois elementos <*media*> são apresentados (linhas 1 a 8). O primeiro elemento (linhas 1 a 4) especifica uma URL que referencia o fluxo elementar *default* do serviço da aplicação. O segundo elemento (linhas 5 a 8) especifica uma URL que referencia um fluxo secundário (ABNT, 2008a) de menor *tag*. Por meio de atribuição de valores a propriedade *soundLevel* (ABNT, 2009; ITU-T, 2009) de cada um desses dois elementos (linha 3 para o primeiro elemento e linha 7 para o segundo elemento), o autor pode especificar relações para permitir que o usuário escolha o fluxo elementar a ser apresentado. Por exemplo, escolha do idioma através do controle remoto. O mesmo pode ser realizado com a propriedade *visible* (ABNT, 2009; ITU-T, 2009), mas para fluxos do tipo de vídeo (por exemplo, escolha entre câmeras diferentes de um evento esportivo) ou do tipo de texto (por exemplo, escolha entre *closed captions* (ABNT, 2008a) em idiomas diferentes).

As âncoras “a1synch1” (linha 2) e “a2synch1” (linha 6) dos dois elementos de media são discutidas na próxima seção.

6.3. Bases Temporais

Em um fluxo elementar multiplexado em um fluxo de transporte podem existir vários conteúdos em sequência. Por exemplo, em um fluxo elementar que carrega os vídeos (principal) de programas, vários vídeos (vários programas) são colocados em sequência representando a programação de uma emissora. Todos esses vídeos foram gerados (tiveram suas amostras geradas) seguindo uma base temporal, que pode diferenciar de um vídeo para outro. Para a decodificação desses vídeos o conhecimento da base temporal se faz necessária e, portanto, precisam ser enviadas ao dispositivo receptor. Um fluxo elementar, denominado NPT (Normal Play Time) é responsável por esse envio. Em um fluxo elementar

NPT, diversas bases temporais podem ser definidas, em sequência, para cada serviço do fluxo de transporte. Note que essa sequência determina que, em um dado instante, apenas uma base temporal, entre as definidas para o serviço, pode progredir (as outras bases devem estar no estado de pausa).

O conteúdo de um serviço de TVD pode, no entanto, não ser contínuo: o conteúdo de um serviço pode ser entrelaçado com outros conteúdos. Por exemplo, no fluxo elementar transportando os vídeos, mencionado no parágrafo anterior, pode ser que um programa de TV (e consequentemente seu vídeo) seja interrompido para a inserção de um vídeo de propaganda, e depois seja retomado. Como conseqüência, o fluxo elementar NPT pode carregar bases temporais entrelaçadas. Por serem entrelaçadas, essas bases precisam ser identificadas e são, através de um campo, denominado *contentId*, do descritor NPT.

Uma propriedade, também denominada *contentId*, foi especificada para objetos de mídia de uma aplicação NCL cujo conteúdo refere-se a um fluxo elementar, como os discutidos na seção anterior. Inicialmente, esses objetos de mídia não possuem uma base temporal associada, tendo sua propriedade *contentId* com o valor "null". Assim que um objeto que referencia um fluxo elementar é iniciado (ABNT, 2009), o middleware associa ao objeto a base temporal que estiver progredindo naquele instante, considerando as bases temporais definidas para o serviço referenciado pelo objeto de mídia pelo campo *program_number* do esquema que define seu conteúdo (caso o program_number seja omitido, é considerado o serviço em que a aplicação foi multiplexada). Nesse instante, a propriedade *contentId*, passa a ter o valor do identificador dessa base temporal. O exibidor instanciado para esse objeto de mídia deve ser capaz de tratar o valor da base temporal passado pelos descritores NPT com o mesmo *contentId*.

Como mencionado, uma base temporal pode ter uma transição em seu estado, mudando de progredindo para pausado ou de pausado para progredindo, quando entrelaçada com outra base temporal. Transições também existem entre bases temporais, obviamente. Para tratar essas transições e refletir a mudança de base nas apresentações dos objetos de mídia, outra propriedade definida para os objetos de mídia que referenciam fluxos elementares, denominada *standby*, assume o valor "*true*" enquanto o objeto estiver com sua base temporal no estado pausado e "*false*" enquanto estiver no estado progredindo.

A Figura 9 apresenta um exemplo de como as bases temporais podem estar entrelaçadas. Nessa figura, um filme chamado "Garrincha" (áudio e vídeo codificados) é transmitido através de um fluxo contínuo. Durante a transmissão são inseridos dois outros conteúdos a esse fluxo, relativos a duas propagandas ("Propag.1" e "Propag.2"). Quatro segmentos são definidos:

- S1: primeiro segmento da grade de programação, com o filme "Garrincha", possui a base temporal "cid1" no estado progredindo (entre os valores 0 e 30), e deve ter a aplicação "App1" em exibição, considerando seu tempo de início em 5 NPT_{cid1};
- S2: segundo segmento da grade, com o filme "Propag.1", possui a base temporal "cid2" no estado progredindo (valores de 0 a 10), a base temporal "cid1" no estado pausado (valor constante 30) e deve ter a aplicação "App2" em exibição;
- S3: terceiro segmento da grade, com o filme da "Propag.2", possui a base temporal "cid3" no estado progredindo (valores de 0 a 15), a base temporal "cid1" no estado pausado (valor constante 30) e deve ter a aplicação "App3" em exibição;
- S4: quarto segmento da grade, com o filme "Garrincha", possui a base temporal "cid1" no estado progredindo (valores de 30 a 60) e deve ter a aplicação "App1" em exibição, considerando seu tempo de início em 5 NPT_{cid1}.

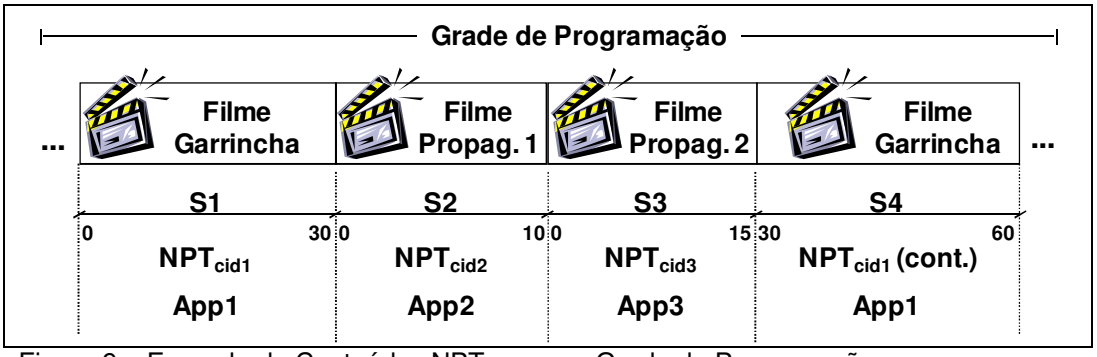

Figura 9 – Exemplo de Conteúdos NPT em uma Grade de Programação.

Nesse exemplo, que também será utilizado no Capítulo 7, as aplicações NCL e o conteúdo dos objetos de mídia são transmitidos através de carrosséis de objetos, durante os segmentos definidos na grade de programação do exemplo.

Para especificar âncoras de conteúdo a serem usadas nas relações de sincronismo, basta o autor das aplicações definir os instantes temporais das

âncoras do objeto que referencia o fluxo elementar, como apresentado nas linhas 2 e 6 do Quadro 4, referenciando a base temporal do objeto. Assim fazendo, não corre o risco de nenhum problema de sincronismo, mesmo se o trecho definido pela âncora for entrelaçado por outro conteúdo.

Note que o objeto "a1" pode ter associada a ele uma base temporal diferente do objeto "a2", dependendo do instante em que cada objeto é iniciado. Note também que o autor das aplicações não precisa se preocupar com os detalhes de acesso às bases temporais e toda a complexidade da manipulação de descritores NPT (ISO, 1998), nem mesmo se preocupar com o envio de eventos de sincronismo para alterar o comportamento da aplicação, como discutido na Seção 2.4.3. O mecanismo proposto permite não só a associação, e identificação, de bases temporais, como também a identificação do estado da base temporal associada ao objeto, que pode ser usado em ações de sincronismo, como descrito a seguir.

Por meio da propriedade *standby*, o autor pode definir relações de sincronismo no caso da ocorrência de transições entre os estados (progredindo ou em pausa) da base temporal associada ao objeto. O Quadro 5 apresenta como exemplo, uma aplicação NCL em que a propriedade *standby* é usada para pausar e "esconder" uma aplicação quando o objeto que referencia um fluxo elementar que corresponde ao vídeo principal de um programa de TVD é temporariamente interrompido por uma propaganda. A mesma propriedade é usada para retomar a aplicação. Neste caso, as ações de sincronismo atuam diretamente no ciclo de vida das aplicações.

No Quadro 5, o código NCL define, inicialmente, a base de conectores do documento "connectors.ncl" (linhas 5 e 7), que é importada para o documento "nptControl". O conteúdo exato de uma base de conectores é especificado por um perfil de linguagem (ABNT, 2009) que utiliza as facilidades oferecidas pelos conectores. Entretanto, como a definição de conectores não é facilmente realizada por usuários inexperientes, a idéia é ter usuários experientes definindo os conectores, armazenando-os em bibliotecas (bases de conectores) que possam ser importadas, tornando-as disponíveis a outros usuários para a criação de elos. O Anexo C da norma brasileira (ABNT, 2009) fornece um exemplo de definições de conectores que podem ser importadas.

As linhas 11 e 12 indicam que o documento inicia pelos nós “v1” e “v2” (linhas 12 a 15). Em assumindo que o documento foi enviado pelo canal de difusão, o atributo *src* igual a “sbtvd-ts://video” do nó “v1” significa que esse nó de mídia se refere ao vídeo principal da emissora. Já o nó “v2” se refere a um recurso transmitido no carrossel de objetos e mapeado em um endereço do ambiente de recepção.

```
1:  <?xml version="1.0" encoding="UTF-8"?>
2:  <ncl id="nptControl"
3:  xmlns="http://www.ncl.org.br/NCL3.0/EDTVProfile">
4:   <head>
5:    <connectorBase>
6:      <importBase alias="conn" documentURI="connectors.ncl"/>
7:    </connectorBase>
8:   </head>
9:   <body id="appBody">
10:   <property name="visible"/>
11:   <port id="p1" component="v1"/>
12:   <port id="p2" component="v2"/>
13:   <media id="v1" src="sbtvd-ts://video">
14:     <property name="standby"/>
15:   </media>
16:   <media id="v2" src="media/anime.mpg"/>
17:   <link id="l1" xconnector="conn#onEndAttribTestSetPause">
18:     <bind component="v1" interface="standby"
19:           role="onEndAttribution"/>
20:     <bind component="v1" interface="standby" role="attTest">
21:       <bindParam name="testVar" value="true"/>
22:     </bind>
23:     <bind component="appBody" interface="visible" role="set">
24:       <bindParam name="setVar" value="false"/>
25:     </bind>
26:     <bind component="appBody" role="pause"/>
27:   </link>
28:   <link id="l2" xconnector="conn#onEndAttribTestSetResume">
29:     <bind component="v1" interface="standby"
30:           role="onEndAttribution"/>
31:     <bind component="v1" interface="standby" role="attTest">
32:       <bindParam name="testVar" value="false"/>
33:     </bind>
34:     <bind component="appBody" interface="visible" role="set">
35:       <bindParam name="setVar" value="true"/>
36:     </bind>
37:     <bind component="appBody" role="resume"/>
38:   </link>
39:   <link id="l3" xconnector="conn#onEndStop">
40:     <bind component="v1" role="onEnd"/>
41:     <bind component="v2" role="stop"/>
42:   </link>
43:  </body>
44: </ncl>
```

Quadro 5 – Aplicação NCL com Associação a Bases Temporais.

Ainda com relação ao Quadro 5, o elo “l1” (linhas 16 a 26) descreve o relacionamento entre a propriedade “standby” do nó “v1” (linha 13), a propriedade “visible” do nó de contexto “appBody” (linha 10) e o próprio nó de contexto “appBody”. Tal relacionamento utiliza a relação “onEndAttibTestSetPause”, expressa no documento “connectors.ncl” e especifica

que ao terminar uma atribuição na propriedade “standby” do nó “v1” (linhas 18 e 19), o valor dessa mesma propriedade deve ser comparado com o valor “true” (linhas 20 a 22). Caso a comparação seja bem sucedida, duas ações são disparadas: uma ação de atribuição do valor “false” à propriedade “visible” do nó de contexto “appBody” (linhas 23 a 25); e uma ação de pausa ao nó de contexto “appBody” (linha 26).

A propriedade *visible* também pode ser associada a um elemento <*context*> e <*body*>. Nesses casos, quando a propriedade tiver seu valor igual a “true”, vale a especificação de *visible* de cada nó filho. Quando tiver o seu valor igual a “false”, todos os elementos da composição são exibidos de forma invisível. Em particular, quando um documento tem seu elemento <*body*> com a propriedade *visible* = “false” e seu evento de apresentação no estado= “paused”, diz-se que a aplicação está em “espera”. Quando uma aplicação entra em “espera”, o vídeo principal do serviço volta a ocupar 100 % da dimensão da tela em que é exibido e o áudio principal a 100 % de seu volume.

O elo “l2” especifica um relacionamento similar ao elo “l1”. No entanto, com efeito de retomar a aplicação. Ao terminar uma atribuição da propriedade “standby” do nó “v1” (linhas 29 e 30), o valor dessa mesma propriedade deve ser comparado com o valor “false” (linhas 31 a 33). Caso a comparação seja bem sucedida, duas ações são disparadas: uma ação de atribuição do valor “true” à propriedade “visible” do nó de contexto “appBody” (linhas 34 a 36); e uma ação para retomar o nó de contexto “appBody” (linha 37).

Finalmente, no Quadro 5, o elo “l3” (linhas 39 a 42) especifica um relacionamento entre o nó “v1” e “v2”. Ao terminar o nó “v1” (linha 40), uma ação para terminar o nó “v2” será executada e a aplicação chegará ao seu fim.

A condição especificada, de ‘ao terminar o nó “v1”’, refere-se, na verdade, ao término da base temporal[18] à qual “v1” está associado e não ao fim do fluxo elementar em si. Isso permite ao autor especificar, como no exemplo, que quando o filme do Garrincha terminar, a aplicação também deve ser finalizada.

[18] Uma base temporal pode ser finalizada pelo provedor de conteúdo de duas formas: através da transmissão de um descritor denominado *NPTEndPointDescriptor* (ISO, 1998); não enviar descritores NPT com o *contentId* da base temporal a ser finalizada por um período de tempo (ISO, 1998).