8
Sistemas de Transporte

Aplicações de TVD (com seus objetos de mídia relacionados) e comandos de edição de aplicações ao vivo (com seus parâmetros associados) são transportados em estruturas de dados de serviços assíncronos definidas em cada sistema de TVD específico. Este capítulo apresenta várias alternativas de transportes para o middleware Ginga-NCL, salientando as estruturas de dados que permitem não apenas o controle do ciclo de vida das aplicações e o mapeamento de referências do ambiente de autoria na sintaxe de transferência sem a intervenção, ou conhecimento, do autor, mas também um maior desempenho na transmissão e processamento de aplicações. Embora propostas para o middleware Ginga-NCL, focando tanto a TVD por difusão terrestre quanto sistemas de IPTV, as opções podem ser estendidas a outros middlewares.

8.1.
Comandos de Edição e Aplicações no Ginga-NCL

Esta seção apresenta quatro estruturas necessárias para a exibição de uma aplicação de TVD: descritor de evento, mapa-de-eventos, arquivo de dados e metadados.

8.1.1.
Descritor de Evento

Os Comandos de Edição NCL (definidos no Capítulo 5) consistem em eventos (ocorrências no tempo) envelopados em uma estrutura chamada descritor de evento: a primeira estrutura de interesse deste capítulo.

Cada descritor de evento (de edição) tem uma estrutura composta basicamente por um id, uma referência de tempo e um campo de dados privados. A identificação define univocamente o evento como sendo de edição (e não cada tipo de comando). No middleware Ginga, outros tipos de evento além dos de

edição NCL podem ser enviados. A referência de tempo (campo *eventNPT*) indica o exato momento de ocorrência (disparo) do evento. Um tempo de referência igual a zero informa que o evento de edição deve ser disparado imediatamente após ser recebido. Se diferente de zero, o tempo de disparo é no momento de ocorrência do valor NPT (*Normal Play Time*) (ISO, 1998) especificado, associado à base temporal identificada pelo objeto *Tap* discutido na Seção 2.3. O campo de dados privados oferece suporte para a identificação do comando e a definição de parâmetros do evento de edição, como apresentado na Tabela 10.

Diferente dos outros sistemas discutidos na Seção 2.3, qualquer alteração de comportamento de uma aplicação NCL é feita de forma declarativa, evitando erros e efeitos colaterais discutidos naquela seção. Assim, os comandos de edição NCL devem ter uma sintaxe padrão bem definida, como discutido em (Costa, 2006b). Na Tabela 10, o campo *commandTag* identifica univocamente os tipos de comandos de edição.

Sintaxe	Número de Bits
`EventDescriptor() {`	
`eventId`	15
`eventNPT`	33
`privateDataLength`	8
`commandTag`	8
`sequenceNumber`	7
`finalFlag`	1
`privateDataPayload`	8 a 1928
`FCS`	8
`}`	

Tabela 10 – Descritor de evento para comandos de edição NCL.

Ainda na Tabela 10, para permitir o envio de um comando completo, com tamanho superior a 241 bytes, em mais de uma estrutura de descritor de evento, todos os descritores de um mesmo comando devem ser numerados e enviados em seqüência (isto é, não pode ser multiplexado com outros comandos de edição com o mesmo *commandTag*), com o *finalFlag* igual a 0, exceto para o último descritor, que deve ter o campo *finalFlag* igual a 1. O campo *privateDataPayload* contém os parâmetros do comando de edição. Finalmente, o campo FCS contém um *checksum* de todo o campo *privateData*, inclusive o *privateDataLength*.

Como discutido no Capítulo 5, os comandos *add* têm entidades NCL como seus parâmetros (parâmetros de comando especificados em XML). A consistência do documento é mantida pelo Formatador NCL, quer a entidade especificada já exista ou não. As entidades são definidas utilizando uma notação sintática idêntica à usada pelos esquemas NCL (veja Capítulo 5). Se o parâmetro de comando baseado em XML for curto o suficiente, ele pode ser transportado diretamente no *payload* dos descritores de evento. Senão, o *privateDataPayload* transporta um conjunto de pares de referência {cUri, cId}, para a identificação dos recursos, diferentemente dos sistemas apresentados na Seção 2.6, como descrito a seguir.

No caso de arquivos recebidos pelo canal de difusão (documentos ou nós NCL enviados sem solicitação), cada par relaciona um caminho de arquivo ou diretório de arquivos identificado pelo sistema de autoria e sua respectiva localização no sistema de transporte. Não é necessária a inclusão de um par {cUri, cId} no comando para cada arquivo enviado por difusão. Mas é necessário que a partir dos pares {cUri, cId} incluídos no comando, todo arquivo recebido possa ter o seu caminho absoluto deduzido a partir de metadados também enviados ao receptor, conforme discutido na Seção 8.1.3. Isso equivale a dizer que, a partir dos metadados recebidos e dos pares {cUri, cId} dos comandos, será possível o sistema receptor mapear os conteúdos dos objetos de mídia referenciados no documento NCL na sua localização dentro da base privada que ele gerencia. No caso de arquivos recebidos sob demanda pelo canal de interatividade ou localizados no próprio receptor, nenhum par de referências necessita ser enviado, exceto se o arquivo for o da especificação do documento NCL ou da especificação XML do nó (objeto) NCL que deverá ser adicionado, segundo o comando de adição (*add*) correspondente. Nesse caso, o par {cUri, "null"} deve ser enviado, especificando o caminho do arquivo a ser buscado.

Considere, por exemplo, o sistema de arquivos da aplicação NCL, denominada "*primeiroJoao"* (Soares, 2009a), apresentado na Figura 10, enviada por meio de um carrossel de objetos.

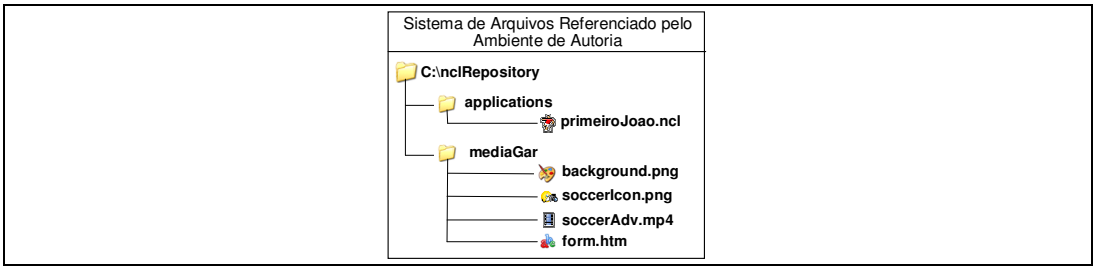

Figura 10 – Sistema de Arquivos da Aplicação *primeiroJoao*.

A adição da aplicação na base privada "TV GINGA" seria realizada pelo comando:

```
addDocument("TV GINGA",
    {"C:\nclRepository\applications", "0x1, 0x1, 0x2"})
```

Note que, nesse caso, o comando de edição faz referência apenas ao caminho do diretório onde está o documento NCL e onde ele será transportado (no caso, em um carrossel de objetos: *Service Domain* "0x1", *module* "0x1" e *object* "0x2", conforme discutido na Seção 8.2.3.2). Com o auxílio de metadados, também recebidos pelo sistema de transporte, todos os arquivos poderão ter seus caminhos absolutos resolvidos a partir desse relacionamento, conforme também discutido na Seção 8.2.3.2. A Tabela 11 apresenta os campos do descritor de evento do comando. Note que o comando determina a adição imediata do documento na base.

Campo	Valor
eventId	Identificador de 15 bits
eventNPT	0
privateDataLength	Comprimento do restante do comando
commandTag	0x05
sequenceNumber	0x00
finalFlag	1
privateDataPayload	"TV GINGA", {"C:\nclRepository\application", "0x1,0x1,0x2"}
FCS	8 bits de *checksum*

Tabela 11 – Descritor de evento para o comando addDocument.

8.1.2. Mapa-de-Eventos

Três tipos de estrutura de dados são definidos para dar suporte à transmissão de parâmetros dos comandos de edição NCL, além da estrutura de descritor de

evento já definida: mapa, metadados e arquivo de dados. As duas últimas são assuntos da próxima subseção.

Para estruturas de mapa (*mappingStructure*), o campo *mappingType* identifica o tipo do mapa. Se o valor de *mappingType* for igual a "0x01" ("events"), um mapa-de-eventos é caracterizado. Nesse caso, depois do campo *mappingType*, vem uma lista de identificadores de eventos, como ilustrado na Tabela 12. Outros valores para o campo *mappingType* podem ser definidos, mas não são relevantes para a discussão desta tese.

Sintaxe	Número de Bits
`mappingStructure() {`	
`    mappingType`	`8`
`    for (i=1; i<N; i++) {`	
`        eventId`	`16`
`        eventNameLength`	`8`
`        eventName`	`8 to 255`
`    }`	
`}`	

Tabela 12 – Lista de identificadores de eventos definidos pela estrutura de mapa.

Mapas-de-eventos são usados para mapear nomes de eventos em *eventIds* dos descritores de evento. Mapas-de-eventos são usados para indicar quais eventos devem ser recebidos. Nomes de eventos permitem especificar tipos de eventos, oferecendo maior nível de abstração às aplicações. Assim, quando for necessário enviar um comando de edição NCL, deve-se criar um mapa-de-eventos, mapeando a string "*nclEditingCommand*" em um *eventId*, previamente selecionado, de um descritor de evento. O Gerente de Bases Privadas de um sistema receptor deve se registrar como ouvinte de um evento "*nclEditingCommand*" para ser notificado da chegada desse tipo de evento.

8.1.3. Arquivos de Dados e Metadados

Cada estrutura arquivo de dados é de fato o conteúdo de um arquivo que compõe uma aplicação NCL ou um nó NCL. Isto é, um arquivo contendo a especificação da aplicação, ou um arquivo com a especificação XML de um nó NCL, ou um dos arquivos com conteúdo de mídia de um objeto de mídia qualquer (vídeo, áudio, texto, imagem, etc.).

Para permitir ao sistema receptor a completa independência entre a localização dos dados de uma aplicação em seu sistema de armazenamento local e a localização dos dados como referenciado pelo documento de especificação da aplicação, uma estrutura de metadados é definida, conforme o *schema* XML *NCLMetadataFile* apresentados no Quadro 6. Para todo arquivo de dado entregue sem solicitação (*pushed file*), uma associação entre sua localização no sistema de transporte (identificação do sistema de transporte – atributo *component_tag* – e a identificação do arquivo no sistema de transporte – atributo *structureId*) e seu identificador de recurso universal (atributo *uri*), conforme especificado no documento da aplicação, é definida.

```
<schema xmlns="http://www.w3.org/2001/XMLSchema"
 xmlns:NCLMetadataFile="http://www.ncl.org.br/NCLMetadataFile"
 targetNamespace="http://www.ncl.org.br/NCL3.0/NCLMetadataFile"
 elementFormDefault="qualified" attributeFormDefault="unqualified">

 <complexType name="NCLMetadataType">
  <sequence>
   <sequence>
    <element ref="NCLMetadataFile:baseData"
                minOccurs="0" maxOccurs="unbounded"/>
   </sequence>
   <element ref="NCLMetadataFile:pushedRoot"
                minOccurs="0" maxOccurs="1"/>
   <sequence>
    <element ref="NCLMetadataFile:pushedData"
                minOccurs="0" maxOccurs="unbounded"/>
   </sequence>
  </sequence>
  <attribute name="name" type="string" use="optional"/>
  <attribute name="size" type="positiveInteger" use="optional"/>
 </complexType>

 <complexType name="baseDataType">
  <sequence>
   <element ref="NCLMetadataFile:pushedRoot"
                        minOccurs="0" maxOccurs="1"/>
   <sequence>
    <element ref="NCLMetadataFile:pushedData"
                minOccurs="0" maxOccurs="unbounded"/>
   </sequence>
  </sequence>
  <attribute name="uri" type="anyURI" use="required"/>
 </complexType>

 <complexType name="pushedRootType">
  <attribute name="component_tag"
                type="positiveInteger" use="optional"/>
  <attribute name="structureId" type="string" use="required"/>
  <attribute name="uri" type="anyURI" use="required"/>
  <attribute name="size" type="positiveInteger" use="optional"/>
 </complexType>

 <complexType name="pushedDataType">
  <attribute name="component_tag"
                type="positiveInteger" use="optional"/>
  <attribute name="structureId" type="string" use="required"/>
  <attribute name="uri" type="anyURI" use="required"/>
  <attribute name="size" type="positiveInteger" use="optional"/>
 </complexType>

 <!-- declare global elements in this module -->
 <element name="metadata" type="NCLMetadataFile:NCLMetadataType"/>
 <element name="baseData" type="NCLMetadataFile:baseDataType"/>
 <element name="pushedRoot" type="NCLMetadataFile:pushedRootType"/>
 <element name="pushedData" type="NCLMetadataFile:pushedDataType"/>

</schema>
```

Quadro 6 – Schema XML NCLMetadataFile.

Para cada arquivo de documento NCL ou arquivo de documento XML, usados nos comandos de edição *addDocument* e *addNode*, pelo menos um arquivo de metadados deve ser definido. Apenas um arquivo de aplicação NCL ou um arquivo de documento XML representando um nó NCL a ser inserido pode ser definido por uma estrutura de metadados (pelo elemento *<pushedRoot>*). Mais precisamente, pode haver apenas um elemento *<pushedRoot>* em um arquivo de metadados. Contudo, uma aplicação NCL (e seus arquivos de conteúdo) ou um documento de especificação de um nó NCL (e seus arquivos de conteúdo) podem

se estender por mais de um arquivo de metadados. Mais ainda, podem existir arquivos de metadados sem qualquer aplicação NCL ou documento XML descritos em seus elementos <*pushedRoot*> e <*pushedData*>.

```
<metadata name="primeiroJoao" size="50kb">
     <baseData uri="file://c:/nclRepository/applications/">
       <pushedRoot component_tag="0x01.0x09" structureId="0x0A"
         uri="primeiroJoao.ncl" size="10KB"/>
       <pushedData component_tag="0x01.0x09" structureId="0x09"
          uri="../mediaGar/background.png" size="10KB"/>
       <pushedData component_tag="0x01.0x09" structureId="0x08"
           uri="../mediaGar/soccerIcon.png" size="10KB"/>
       <pushedData component_tag="0x01.0x09" structureId="0x07"
          uri="../mediaGar/soccerAdv.mp4" size="10KB"/>
       <pushedData component_tag="0x01.0x09" structureId="0x06"
          uri="../mediaGar/form.htm" size="10KB"/>
     </baseData>
</metadata>
```

Quadro 7 – Arquivo de metadados do exemplo primeiroJoao.

Retomando o exemplo da Figura 10, o arquivo de metadados para o sistema de arquivos da aplicação NCL é aquele apresentado no Quadro 7. Nesse exemplo, todos os dados estão sendo transmitidos em um fluxo de transporte identificado como componente 0x09 do serviço 0x01. Dentro desse fluxo os vários arquivos são identificados pelo campo *structureId*. As URIs definidas são sempre relativas à URI definida no elemento <*baseData*>, podendo haver mais de um em metadado.

8.2. Sistema de Transporte

As quatro estruturas definidas: arquivo de dados, metadados, mapa-de-eventos e descritores de evento, além dos conteúdos definidos em fluxos elementares de bits, compõem todas as estruturas necessárias para a exibição de uma aplicação de TVD. As quatro estruturas podem ser envelopadas em estruturas do sistema de transporte, mas podem também ter seu conteúdo distribuído, principalmente as estruturas de metadados, como discutido nas opções de transporte da Seção 8.2.2.

8.2.1. Transporte de Descritores de Evento

Comandos de edição NCL podem ser transportados usando descritores de evento DSM-CC (ISO, 1998). Os descritores de evento DSM-CC têm uma

estrutura muito parecida com a dos descritores de evento NCL apresentados na Seção 8.1.1. Na verdade, a estrutura dos descritores de evento NCL foi definida para tornar o mais fácil possível seu envelopamento em descritores de evento de fluxo DSM-CC.

Descritores de evento de fluxo DSM-CC devem ser enviados mais de uma vez, antes do momento desejado para o disparo do comando, para garantir sua recepção, uma vez que não há garantia de entrega.

O uso de descritores de evento de fluxo DSM-CC foi a solução adotada pelo SBTVD (ABNT, 2008b). Qualquer outro protocolo para envio de dados sem solicitação poderia, no entanto, ter sido adotado, incluindo o FLUTE.

8.2.2. Transporte de Metadados

Metadados podem ser transmitidos usando o mesmo protocolo utilizado no transporte de arquivos de dados e mapa-de-eventos, ou ainda usando um protocolo diferente. Nesta seção discutiremos esse último caso, deixando para a Seção 8.2.3 a discussão da transmissão integrada.

8.2.2.1. Metadados em Descritores de Evento

Uma alternativa para o transporte das estruturas de metadados é tratá-las como parâmetros dos comandos de edição NCL *addDocument* e *addNode*, a serem transportados no campo *privateDataPayload* dos descritores de evento.

Nesse caso, o conjunto de pares {cUri, cId} dos comandos *addDocument* e *addNode* é substituído por parâmetros da estrutura de metadados, que, por sua vez, definem, como apresentado na Quadro 7, um conjunto de pares {“uri”, “*component_tag.structureId*”} para cada arquivo transmitido sem solicitação (*pushed file*).

Voltando ao exemplo da Figura 10, o par {cUri; cId} na Tabela 11 seria substituído pela estrutura de metadados serializada do Quadro 7. Na estrutura de metadados, os atributos *component_tag* dos elementos <*pushedRoot*> e <*pushedData*> devem, nesse caso, ser definidos obrigatoriamente, uma vez que a

estrutura não é mais transportada no mesmo fluxo que transporta os arquivos de dados da aplicação NCL.

Devido ao baixo overhead envolvido, a transmissão de metadados em descritores de evento provavelmente é a solução mais adequada para sistemas de IPTV que não utilizam Seções MPEG-2 no transporte de dados.

8.2.2.2. Metadados em Seções de Metadados MPEG-2

Outra alternativa para o transporte de estruturas de metadados é encapsulá-las em Seções de Metadados MPEG-2, transportadas em fluxos MPEG-2 do tipo "0x16" (ISO, 2007). Cada Seção de Metadados MPEG-2 poderá conter dados de apenas uma estrutura de metadados. Contudo, uma estrutura de metadados pode se estender por várias Seções de Metadados. Para tanto, um campo da seção (*section_fragment_indication*) é usado, indicando se a estrutura de metadados se estende por mais de uma seção, e, no caso de se estender, se a seção carrega a primeira parte da estrutura, ou a última parte, ou uma parte intermediária.

No campo *privateDataPayload* do descritor de evento dos comandos de edição NCL *addDocument* e *addNode* (vide Tabela 10), os pares de referência {cUri, cId} devem ter os parâmetros cUri com o valor "null". O parâmetro cId do primeiro par deve identificar o fluxo elementar TS (ISO, 2007) do tipo= "0x16" e a estrutura de metadados (campo "*structureId*" da seção de metadados) que carrega o caminho absoluto do documento NCL ou da especificação do nó NCL (o caminho no servidor de dados). Se outras estruturas de metadados forem usadas para relacionar arquivos presentes no documento NCL, ou na especificação do nó NCL, a fim de completar os comandos *addDocument*, ou *addNode*, com conteúdos de mídia, outros pares de referência {cUri, cId} devem ser definidos no comando. Nesse caso, o parâmetro cUri deve ter o valor "null" e o parâmetro cId correspondente no par deve referir-se ao *component_tag* de um serviço (ISO, 2007) e ao *structureId* do metadado correspondente.

Caso sejam enviadas sem a solicitação do receptor, Seções de Metadados MPEG-2 devem ser repetidas periodicamente para garantir que as estruturas de metadados sejam recebidas pelo cliente, independente do momento da sintonização do serviço que demanda a execução da aplicação de TVD.

8.2.3.
Transporte Conjunto das Estruturas

Como já mencionado, as estruturas de metadados, arquivo de dados e mapa-de-eventos podem ser transmitidas usando o mesmo protocolo. As soluções propostas pelo middleware Ginga-NCL são discutidas nesta seção. Elas se diferenciam principalmente pelo overhead imposto pelas várias camadas de encapsulamento das estruturas.

8.2.3.1.
Carrossel de Dados

Para transmissão das estruturas, o carrossel de dados DSM-CC pode ser utilizado (ISO, 1998). Carrossel de dados é a forma mais simples de transmissão cíclica de dados DSM-CC. Nele não existe qualquer indicação sobre o que consistem os dados. As especificações ATSC (ATSC, 2000) e ARIB (ARIB, 2004b) fazem uso dessa modalidade de transmissão.

Um carrossel de dados consiste de uma sequência de módulos, que define um ciclo. Módulos são transmitidos um após o outro até que todos os módulos de um ciclo tenham sido transmitidos, quando todo o processo recomeça. Um módulo pode ser inserido mais de uma vez em um mesmo ciclo do carrossel.

Não existe nenhuma estruturação de mais alto nível acima do módulo definida pelo padrão DSM-CC. Entretanto, o Ginga-NCL pode dar essa semântica de mais alto nível, transportando nos módulos do carrossel suas estruturas de metadados, arquivo de dados e mapa-de-eventos, em uma solução bem mais expressiva que os outros padrões mencionados nesta seção.

Cada módulo pode conter várias estruturas, desde que o tamanho seja menor que 64 KBytes. Não é possível dividir um arquivo de dados em mais de um módulo. Dessa forma, arquivos com tamanho maior que 64 KBytes devem ser transportados em um único módulo. Esse é o único caso em que um módulo pode exceder o tamanho de 64 KBytes. Arquivos em um módulo podem vir de qualquer parte do sistema de diretórios, eles não têm a necessidade de pertencerem ao mesmo diretório. Cada módulo é quebrado, por sua vez, em blocos, como mostra a Figura 11, que são então transportados em Seções Privadas MPEG-2 (ISO, 2007). Note, na figura, que as estruturas de metadados, mapa-de-eventos e arquivo

de dados precisam ser de alguma forma encapsuladas em mensagens. No Ginga-NCL esse encapsulamento será o de mensagens BIOP (ISO, 1998), o mesmo usado nos carrosséis de objetos apresentados na próxima seção, ou em Seções NCL, conforme discutido na Seção 8.2.3.3.

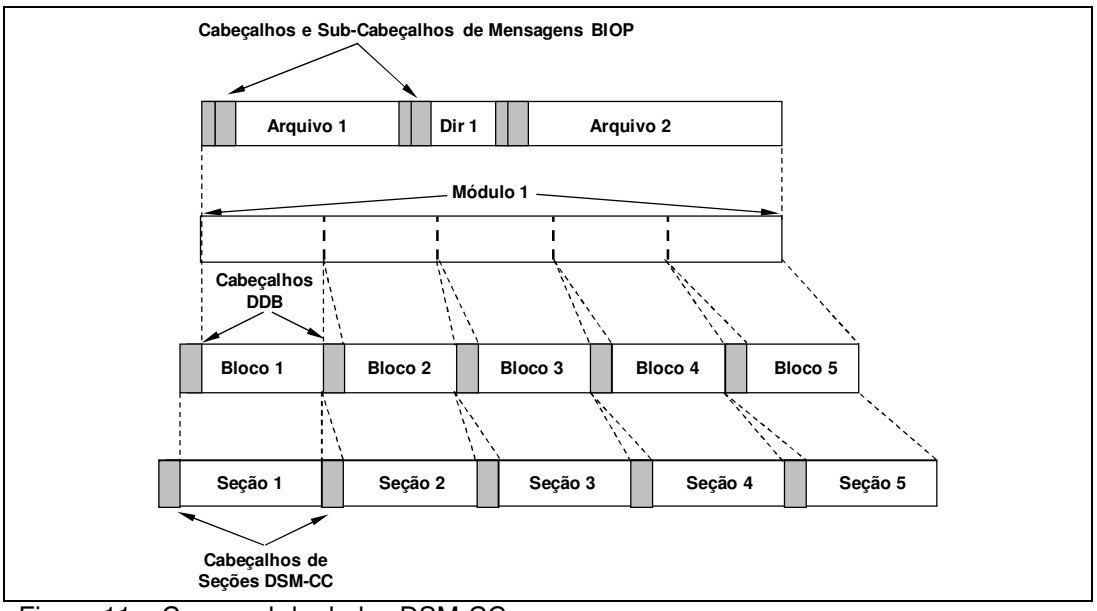

Figura 11 – Carrossel de dados DSM-CC.

8.2.3.2. Carrossel de Objetos

O carrossel de objetos DSM-CC (ISO, 1998) pode ser usado como alternativa para transmissão das estruturas. Essa foi a solução escolhida quando da adoção do Ginga-NCL pelo SBTVD (ABNT, 2008b).

O protocolo de carrossel de objetos DSM-CC permite a transmissão cíclica de objetos de eventos de fluxo e sistemas de arquivos.

Objetos de eventos de fluxo são utilizados para mapear nomes de eventos de fluxo em *ids* de eventos de fluxo, definidos pelos descritores de evento, cumprindo o mesmo papel das estruturas mapa-de-eventos. Assim, quando um comando de edição NCL precisa ser enviado, um objeto de eventos de fluxo DSM-CC deve ser criado, mapeando a string "*nclEditingCommand*" em uma id de evento de fluxo. O objeto é então colocado em um carrossel de objetos DSM-CC e enviado em um fluxo elementar MPEG-2 TS (ISO, 2007). Um ou mais descritores de evento de fluxo DSM-CC com o *id* previamente selecionado podem ser então criados e enviados em outro fluxo elementar MPEG-2 TS.

Carrosséis de objetos DSM-CC são também usados para o transporte de arquivos organizados em diretórios. Um demultiplexador DSM-CC é responsável por montar a estrutura recebida no dispositivo receptor. Parâmetros dos comandos de edição, especificados como documentos XML (documentos NCL ou nós NCL que se quer adicionar) podem, assim, ser organizados em sistemas de arquivos a serem transportados nos carrosséis.

Cada carrossel de objetos contém uma árvore de diretórios, que é quebrada em uma série de módulos, que podem conter um ou mais arquivos (objetos de arquivo) ou diretórios (objetos de diretórios e objetos *Service Gateway*). Um diretório contém o nome dos seus componentes filhos e ponteiros para localização do conteúdo dos mesmos no carrossel. Objetos *Service Gateway* (srg) representam um conceito similar a um diretório. A principal diferença é que um objeto *Service Gateway* identifica o diretório raiz da árvore de diretórios de um carrossel de objetos. Isso significa que existirá um e apenas um objeto *Service Gateway* em um carrossel de objetos. A localização do *Service Gateway* é transmitida em uma mensagem *DownloadServerInitiate* (DSI) (ISO, 1998) ao sistema receptor.

Note assim que a própria estrutura do carrossel de objetos representa a mesma estrutura de metadados (com a diferença única que a estrutura de metadados não perde a referência da raiz do diretório) definida na Seção 3.3, embora não a envelope literalmente. O conjunto de pares de referência transportado pelo descritor de evento de fluxo completa toda informação necessária para que o receptor consiga mapear as referências do documento da especificação XML (aplicação NCL ou especificação de um nó NCL) em seu sistema de armazenamento local. Nesse caso, a especificação XML deve ser enviada no mesmo carrossel de objetos que carrega o objeto de eventos de fluxo. O parâmetro cUri do primeiro par de referências deve ter o caminho absoluto da especificação XML (o caminho no servidor de dados). O parâmetro cId correspondente no par deve fazer referência ao IOR (endereço) da especificação XML (carouselId, moduleId, objectKey) (ISO, 1998) no carrossel de objetos. Se outros sistemas de arquivos precisarem ser transmitidos usando outros carrosséis de objeto a fim de completar o comando de edição (como usual nos comandos *addDocument* ou *addNode*) com conteúdo de mídia, outros pares {cUri, cId} devem estar presentes no comando. Nesse caso, o parâmetro cUri deve ter o caminho absoluto da raiz do sistema de arquivos (o caminho no servidor de

transmissão de dados) e o respectivo parâmetro cId no par deve fazer referência ao IOR (carouselId, moduleId, objectKey) de qualquer arquivo ou diretório filho da raiz no carrossel de objetos (o *service gateway* do carrossel).

Voltando ao exemplo da Figura 10, a transmissão dos arquivos de dados, dos metadados e do mapa-de-eventos (objeto de eventos de fluxos) da aplicação NCL "O Primeiro João" seria realizada pelo carrossel da Figura 12.

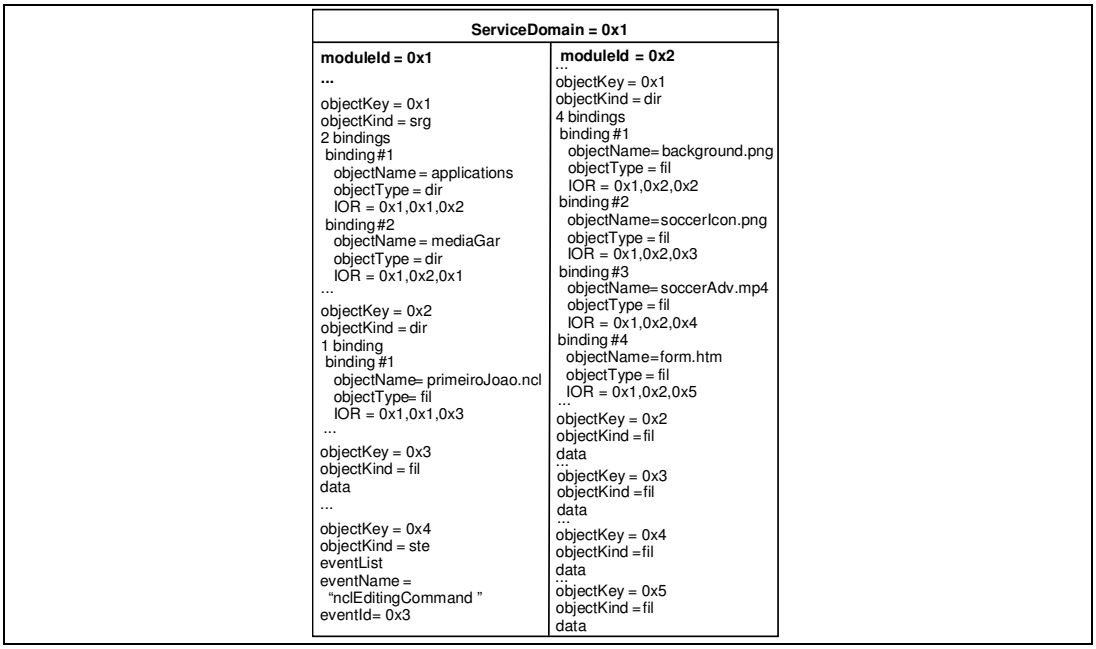

Figura 12 – Carrossel de objetos para Figura 10.

No carrossel, dois módulos são gerados e identificados com valores "0x1" e "0x2". O identificador de cada objeto é apresentado através do campo "objectKey". O objeto (do tipo "srg") que representa o *Service Gateway* é identificado com o valor "0x1" e é encapsulado no Módulo "0x1". Como conseqüência, sua IOR deve ser transmitida na mensagem *DownloadServerInitiate* definida como "0x1,0x1,0x1" (*Service Domain*=0x1, id do módulo=0x1 e id do objeto=0x1). Os objetos que representam o diretório "applications" e o arquivo "primeiroJoao.ncl" são identificados pelos valores "0x2" e "0x3", respectivamente, e também são encapsulados no Módulo "0x1". Já os objetos que representam o diretório "mediaGar" e os arquivos com os conteúdos de seus filhos são identificados com os valores "0x1", "0x2", "0x3", "0x4"e "0x5", respectivamente, mas encapsulados no Módulo "0x2". Ainda no Módulo 1, um objeto de eventos de fluxo (na Figura 12 o objeto do tipo "ste") mapeia a string "*nclEditingCommand*" no identificador "0x3" de evento.

Ainda no exemplo, um descritor de evento de fluxo deve ser transmitido com o valor de *eventId* apropriado, no exemplo "0x3", e o valor "0x05" no *commandTag*, indicando um comando *addDocument*. O parâmetro cUri conterá o esquema (no caso do SBTVD, "x-sbtvd", indicando, opcionalmente, o recebimento no carrossel) e o caminho absoluto do documento NCL ("C:\nclRepository\applications", de acordo com a Figura 10). Finalmente, a IOR do documento NCL no carrossel de objetos é transportada no parâmetro cId (carouselId = 0x1, moduleId = 0x1, objectKey = 0x3).

8.2.3.3. Seções NCL

Seções NCL nos permitem transmitir as três estruturas de dados anteriormente definidas: mapas, metadados e arquivo de dados. Cada Seção NCL pode conter os dados de apenas uma estrutura. Contudo, uma estrutura pode se estender por várias Seções. Estruturas de dados podem ser transmitidas em qualquer ordem e quantas vezes forem necessárias.

Seções NCL podem ser encapsuladas em carrosséis de dados, como indica a Figura 11. Nesse caso, o carrossel é usado apenas para a transmissão cíclica das estruturas, a um custo grande de overhead de encapsulamento.

Seções NCL podem, alternativamente, ser encapsuladas diretamente em um tipo específico de Seção MPEG-2 (identificado por um valor do campo *table_id* de uma seção privada (ISO, 2007)), diminuindo o overhead de encapsulamento. Cada Seção MPEG-2 pode conter apenas uma Seção NCL.

Seções NCL podem também ser encapsuladas em outras estruturas de dados. Por exemplo, o encapsulamento MPE (*Multi-Protocol Encapsulation*) (ATSC, 2000) pode ser usado. Nesse caso, Seções NCL seriam Seções MPEG-2 de datagrama. Elas podem ainda ser envelopadas em outro formato de dados de protocolo que não MPEG-2 System, como, por exemplo, pacotes FLUTE (RFC, 2004).

O primeiro byte do cabeçalho de uma Seção NCL identifica o tipo da estrutura transportada (0x01 para metadatos; 0x02 para arquivos de dados, e 0x03 para mapa-de-eventos). O segundo byte carrega um identificador único da estrutura (*structureId*) no fluxo de transporte. O fluxo e o identificador da

estrutura são aqueles que devem ser associados pela estrutura de metadados, através dos atributos *component_tag* de um serviço e *structureId* dos elementos <pushedRoot> e <pushedData>, a localizadores de arquivos (URL).

Depois do segundo byte, vem uma estrutura de dados serializada, que pode ser a *mappingStructure* (como ilustrado pela Tabela 12), ou a estrutura de matadados (um documento XML, conforme *Schema* XML introduzido na Seção 8.1.3), ou uma estrutura de arquivo de dados (um conteúdo de arquivo serializado). O demultiplexador de Seções NCL é responsável por montar toda a estrutura da aplicação no dispositivo receptor.

Quando Seções NCL são utilizadas, o campo privateDataPayload do descritor de evento (vide Tabela 10) deve carregar o par de referências {cUri, cId}. No entanto, nesse caso, os parâmetros cUri têm sempre o valor "null". No caso dos comandos *addDocument* e *addNode*, o parâmetro cId do primeiro par deve identificar o fluxo elementar ("*component_tag*") de um serviço e a estrutura de metadados que ele transporta ("*structureId*"). A estrutura de metadados, por sua vez, contém o caminho absoluto do documento NCL ou da especificação do nó NCL (o caminho no servidor de dados) e a estrutura arquivos de dados relacionada ("*structureId*") transportada em Seções NCL do mesmo fluxo elementar. Se outras estruturas de metadados adicionais forem necessárias para completar os comandos de edição *addDocument* ou *addNode*, outros pares {cUri, cId} devem se fazer presentes no comando. Nesse caso, os parâmetros cUri devem também ter o valor "null" e os parâmetros cId correspondentes devem referir-se ao *component_tag* e à estrutura de metadados transportada (*structureId*) correspondente.

No caso de se usar Seções privadas MPEG-2 para transporte direto, o começo da Seção será delimitado pelo campo *payload_unit_start_indicator* de um pacote TS (ISO, 2007). Depois dos quatro bytes do cabeçalho TS, a carga (*payload*) do pacote TS começa, com um campo ponteiro de um byte indicando o início de uma Seção NCL (ISO, 2007). No mesmo fluxo elementar que carrega a especificação XML (o arquivo do documento NCL ou o documento XML de especificação de um nó NCL) é recomendado que uma estrutura mapa-de-eventos seja transmitida, para que seja mapeado o nome "*nclEditingCommand*" no *eventId* do descritor de evento que transportará os comandos de edição.

A Figura 13 ilustra o descritor de eventos e o mapa-de-eventos para a aplicação NCL *primeiroJoao*, transportados por meio de Seções NCL. A estrutura de metadados enviada é a mesma do Quadro 7. Pela figura, um fluxo elementar MPEG-2 do serviço 0x01 (*component_tag* = "0x01.0x09") é gerado, carregando todo o sistema de arquivos do programa interativo (o arquivo NCL e todos os arquivos de conteúdo de mídia, conforme a Figura 10). O mapa-de-eventos criado (*structureType*="0x03"; *structureId*="0x0C", na Figura 13), mapeia o nome "*nclEditingCommand*" ao valor de *eventId* (valor "0x03", Figura 13). O descritor de evento define para *eventId* o valor "0x03", e para o *commandTag* o valor "0x05", que indica um comando *addDocument*. O parâmetro cUri tem o valor "null" e o parâmetro cId o valor (*component_tag*= "0x01.0x09", *structureId*= "0x0A").

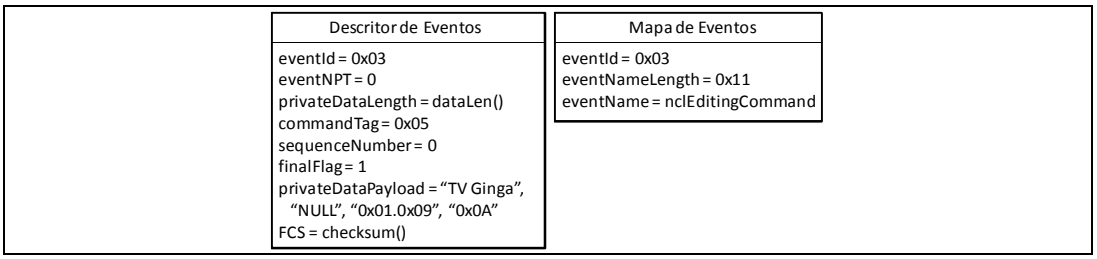

Figura 13 – O primeiroJoao em Seções MPEG-2.