TV 3.0: A Ginga-NCL and Common Core Webservices Extension for Multidevice Support

Karen S. S. Oliveira
Paulo R. M. de Macedo
GPMM - Cefet/RJ
Rio de Janeiro, Brazil
karen.oliveira@aluno.cefet-rj.br
paulo.macedo@aluno.cefet-rj.br

Marina I. P. Josué
Débora C. Muchaluat Saade
Laboratório MídiaCom - UFF
Rio de Janeiro, Brazil
marinaivanov@midiacom.uff.br
debora@midiacom.uff.br

Joel A. F. dos Santos
GPMM - Cefet/RJ
Rio de Janeiro, Brazil
joel.santos@cefet-rj.br

Abstract

Different studies explore the use of second screen devices associated with the content presented on TV. This functionality has been present in the Nested Context Language (NCL) since its proposal as a standard for specifying interactive applications in the Brazilian digital TV system. Despite language support for multiple devices connected to the TV, there is still a lack of a clear definition of protocols for discovery, registration and communication with remote devices. This is the focus of the Guaraná proposal, accepted in the TV 3.0 call. This article extends the Guaraná proposal in order to allow multiple users with their respective HMDs to run the same application. It also generalize the form of communication between Head-Mounted Displays (HMDs) and the middleware Ginga, allowing its reuse by other device types.

Keywords: Ginga, Multidevice, Guaraná

1 Introdução

O uso de dispositivos de segunda tela em um contexto de TV [13] e em particular em aplicações NCL[1] (*Nested Context Language*) não é uma funcionalidade nova [3, 4]. Esta funcionalidade começa agora a atingir os televisores disponíveis no mercado com a implementação do perfil D da norma ABNT 15606-11 [1]. Nas propostas existentes, a indicação de execução de uma mídia em um dispositivo remoto é feita através do atributo *device* em elementos `regionBase` ou, de maneira equivalente, através da propriedade *deviceClass* de uma mídia NCL.

Apesar da existência desta funcionalidade, ainda há uma carência de uma definição clara de protocolos para descoberta, registro e comunicação com dispositivos remotos. Visando suprir essa demanda, Souza *et al.* [14] apresentam uma proposta de comunicação entre o *middleware* Ginga e dispositivos remotos executando o *player* Guaraná.

O *player* Guaraná é uma aplicação para dispositivos HMD (*Head-Mounted Display*), possibilitando a imersão do usuário numa experiência interativa. Ele é projetado para renderizar cenas 360 que são compostas por um vídeo 360^o enriquecido com mídias 2D, objetos 3D e áudio 3D. A descrição de uma cena 360 é feita com uma linguagem baseada em NCL chamada *NCL360*.

O controle da sincronização entre o conteúdo do programa e a cena 360 é feito pelo dispositivo principal, no caso, um receptor de TV com o *middleware* Ginga[2]. O *Guaraná* interage com o *middleware* Ginga recebendo uma cena 360 a ser renderizada, bem como comandos para ativação dos elementos de cena (*e.g.*, iniciar um vídeo), e notificando ao Ginga transições de eventos (*e.g.*, interação do usuário) que tenham ocorrido durante a apresentação do conteúdo 360 no HMD.

Este artigo apresenta uma extensão da proposta intitulada Guaraná [14] visando permitir que múltiplos usuários com seus respectivos HMDs possam executar uma mesma aplicação. Ainda, pretende-se generalizar a forma de comunicação entre HMDs e o *middleware* Ginga, permitindo seu reúso por outros tipos de dispositivos.

O restante deste artigo está organizado como segue. A Seção 2 apresenta trabalhos relacionados. A Seção 3 detalha a extensão da proposta Guaraná, foco deste artigo. A Seção 4 conclui este artigo e apresenta trabalhos futuros.

2 Trabalhos Relacionados

A proposta de suporte a múltiplos dispositivos em NCL, consequentemente no *middleware* Ginga, não é nova. Tal funcionalidade de NCL é apresentada ou utilizada em aplicações em [6, 7, 9]. Costa *et al.* [5] definem duas classes bases nas quais os dispositivos podem se registrar. Na classe ativa os dispositivos controlam o conteúdo sendo apresentado. Já na classe passiva, apenas recebem as amostras de áudio e vídeo prontas para serem exibidas. Os autores em [5] propõem uma arquitetura para comunicação com dispositivos remotos na qual a comunicação do *middleware* Ginga para dispositivos se dá por endereços *multicast* e, no sentido contrário, usando

[1] http://www.ncl.org.br

[2] http://www.ginga.org.br

In: Workshop Futuro da TV Digital Interativa, Ribeirão Preto, Brasil. Anais Estendidos do Simpósio Brasileiro de Sistemas Multimídia e Web (WebMedia). Porto Alegre: Sociedade Brasileira de Computação, 2023.

ISSN 2596-1683

o endereço *unicast* do receptor. Dentre as mensagens enviadas pelo *middleware* estão mensagens de anúncio das classes suportadas, do conteúdo a ser apresentado por uma classe de dispositivos, eventos a serem executados ou mensagens para manutenção da sincronização. Cabe ressaltar que, para cada classe de dispositivo, um IP *multicast* é utilizado. Batista *et al.* [3] estendem a proposta de [5] especificando detalhes das mensagens trocadas entre dispositivos e apresentam um módulo Lua para acesso a essa funcionalidade. Os autores também definem uma arquitetura para implementação de um componente *DeviceIntegration* responsável pelo registro e comunicação entre o Ginga e dispositivos.

Num contexto mais amplo, diversos trabalhos focam na sincronização do conteúdo apresentado num dispositivo remoto com a TV [10–12, 15, 16]. Tal sincronização é garantida seja com a inclusão de metadados comuns no fluxo de transporte *broadcast* e *broadband* [11], com o uso de marcas sonoras inaudíves por seres humanos [10], utilizando amostras de PCR (*Program Clock Reference*) do fluxo de transporte MPEG-2 [12], por uma aplicação web executada simultaneamente em diversos dispositivos [16] ou por protocolos específicos, como é o caso do padrão DVB CSS [15]. Em geral, esses trabalhos consideram que a comunicação entre TV e dispositivos remotos se dá na rede local, com [11, 12] explicitando que tal comunicação é feita via *multicast*.

Diferente das propostas acima que focam em dispositivos remotos de segunda tela, Gómez *et al.* [8] apresentam uma proposta multitela utilizando dispositivos HMDs para execução de conteúdo alternativo ao apresentado na TV. Nesta proposta, metadados multiplexados num fluxo MPEG-DASH são utilizados para sinalizar descoberta de mídias, associação, orquestração, execução e interação do usuário.

3 Proposta

Em Souza *et al.* [14], foi apresentada uma proposta para integração de receptores de TV com o *middleware* Ginga e dispositivos HMD que executam o *player* Guaraná. Esta proposta, aprovada na chamada do Projeto TV 3.0 do Fórum SBTVD, estende o subsistema Ginga *Commom Core Web-Services* (CCWS) criando uma nova API para registro de dispositivos remotos. Aquela proposta também define que o envio de mensagens de sincronização do *middleware* Ginga se dá por chamadas HTTP para o dispositivo remoto.

A proposta apresentada neste artigo avança aquela apresentada em [14] em três frentes: *(i)* modificação da forma de comunicação entre dispositivos; *(ii)* suporte ao uso de múltiplos dispositivos de um mesmo tipo em uma mesma aplicação; e *(iii)* permite a adaptação de conteúdo conforme o usuário de cada dispositivo.

3.1 Registro e Comunicação

A comunicação de um dispositivo remoto com o *middleware* Ginga é estabelecida via *Ginga CCWS*. Assim, após a fase de descoberta SSDP [1], um dispositivo se registra como dispositivo remoto via API *remote-device*. A API de registro é acessada por uma requisição `POST` na rota `http(s)://<host>/dtv/remote-device`.

```
{
    "deviceClass" : <class>,
    "supportedFormats" : [ <format>, ...],
    "recognizableEvents" : [ <event>, ...]
}
```

Listagem 1. Corpo da mensagem de registro do dispositivo remoto.

O corpo da requisição é um objeto JSON[3], de acordo com a estrutura apresentada na Listagem 1. Onde "deviceClass" é um identificador da classe do dispositivo se registrando como dispositivo remoto. O vetor "supportedFormats" indica os formatos de conteúdo que o dispositivo remoto é capaz de executar, enquanto o vetor "recognizableEvents" indica os eventos NCL que esse dispositivo é capaz de reconhecer.

Registrado um dispositivo remoto, o *Ginga CCWS* cria um *WebSocket server* em uma porta atribuída dinamicamente. O retorno é um objeto JSON, de acordo com a estrutura apresentada na Listagem 2.

```
{
     "handle": <handle>,
     "url": <entry-point>
}
```

Listagem 2. Corpo do retorno do registro do dispositivo remoto.

Na mensagem de retorno, "handle" é um identificador gerado pela implementação do *Ginga CCWS*, que deve ser usado para posterior liberação deste recurso através da API de cancelamento de registro. O cancelamento de registro é feito por uma requisição `DELETE` na rota `http(s)://<host>/dtv/remote-device/<handle>`.

Ao conectar-se no *entry-point* do *WebSocket* indicado em "url", a aplicação será capaz de comunicar eventos e receber ações do Ginga-NCL para sincronização do conteúdo apresentado no dispositivo. Essa comunicação é feita por meio de objetos JSON conforme apresentado nos parágrafos a seguir.

Um dispositivo remoto pode requisitar a identificação do usuário utilizando-o num dado momento[4]. Feita a identificação do usuário, o dispositivo envia para o Ginga essa informação em um objeto JSON com a estrutura apresentada na Listagem 3.

```
{
   "user" : <userId>
}
```

[3] https://www.json.org

[4] A lista de usuários registrados na TV é acessada via API específica que foge ao escopo deste artigo.

Listagem 3. Mensagem de identificação do usuário usando o dispositivo remoto.

Cabe destacar que a identificação do usuário poderá ser alterada a qualquer momento. Refeita a identificação, a estrutura apresentada na Listagem 3 é reenviada.

Uma vez iniciada a aplicação NCL e identificada a existência de conteúdo a ser executado em um dispositivo remoto, o *middleware* Ginga faz o envio de um objeto JSON com a estrutura apresentada na Listagem 4.

```
{
   "nodeId" : <nodeId>,
   "nodeSrc" : <URL>,
   "appId" : <appId>,
   "type" : <mimeType>,
   "notifyEvents" : [<event>, ...],
   "properties" : [
      { "name" : <propName>, "value" : <
    propValue> }
   ]
}
```

Listagem 4. Mensagem de envio das informações sobre a mídia a ser executada no dispositivo remoto.

Na mensagem de envio de informações da mídia ou efeito sensorial, "nodeId" é o identificador do nó cujo conteúdo será executado no dispositivo remoto. O atributo "nodeSrc" indica o local onde obter esse conteúdo, se for o caso. O atributo "appId" apresenta o identificador da aplicação a ser utilizado nas rotas para obtenção de arquivos enviados junto da aplicação NCL. O atributo "type" indica o tipo do conteúdo a ser executado. O vetor "notifyEvents" indica os eventos que o dispositivo deve notificar ao longo da execução da aplicação. O vetor "properties" carrega propriedades do nó NCL, conforme definidos na aplicação.

Cabe ressaltar que, diferentes tipos de URLs podem ser indicadas no atributo "nodeSrc". No caso de URLs que não definam um protocolo e apresentam uma URL relativa, o dispositivo remoto deverá utilizar a API do CCWS para obtenção de conteúdos enviados junto com a aplicação. Caso contrário, o conteúdo será acessado conforme o protocolo especificado, caso suportado pelo dispositivo remoto.

A partir do recebimento das informações do nó NCL, o dispositivo remoto já é capaz de carregar todas as informações necessárias para sua apresentação. Sempre que for necessário realizar uma ação em um dos conteúdos sendo apresentados no dispositivo, a comunicação de ações é feita pelo envio de um objeto JSON com a estrutura apresentada na Listagem 5

```
{
   "node" : <nodeId>,
   "eventType" : <presentation|attribution|
    preparation|...>,
   "action" : <start|stop|pause|resume>,
   "delay" : <value>
}
```

Listagem 5. Mensagem de envio de ações para o dispositivo remoto.

Na mensagem de envio de ações, "nodeId" é o identificador do nó sobre o qual a ação deve ser executada, os atributos "eventType" e "action", em conjunto, identificam a ação a ser executada e "delay" é um atributo opcional que indica se a ação deve ser executada com algum atraso após recebida. O valor padrão do atributo "delay" é zero, o que indica que a ação é executada imediatamente.

Por fim, sempre que necessário reportar a ocorrência de uma transição de estado de evento, a comunicação das transições é feita pelo envio de um objeto JSON com a estrutura apresentada na Listagem 6.

```
{
   "node" : <nodeId>,
   "eventType" : <presentation|selection|
    view|...>,
   "transition" : <starts|stops|aborts|
    pauses|resumes>,
   "value" : <value>,
   "user" : <userId>
}
```

Listagem 6. Mensagem de notificação de transição de evento para o *middleware* Ginga.

Na mensagem de notificação de transições, "nodeId" é o identificador do nó sobre a transição do evento que ocorreu, "eventType" identifica o tipo de evento e "transition" a transição. O atributo "value" é um atributo opcional para o caso de eventos do tipo atribuição. O atributo "user" é utilizado para o caso em que seja necessário identificar o usuário que realiza a interação sendo notificada.

3.2 Gerenciamento de Dispositivos

Um exemplo de uso de múltiplos dispositivos conectados ao receptor Ginga é para visualização de uma cena 360 através do Guaraná. A Figura 1 apresenta a arquitetura atualizada do *middleware* Ginga para suporte a dispositivos Guaraná. Cabe ressaltar, entretanto, que a proposta pode ser explorada para conexão com dispositivos de outros tipos, conforme será discutido na Seção 3.4.

Os componentes de Ginga-NCL identificados na figura por *Interaction Manager*, *Context Manager* e *XML Parser* seguem as especificações definidas por NCL 4.0 [2]. O *Interaction Manager* é o componente responsável por gerenciar todas as interações ocorridas durante a execução de um documento. As interações gerenciadas por esse componente podem ocorrer na própria TV ou via dispositivos reconhecedores externos, conectados à TV. O *Context Manager* é o componente responsável por gerenciar o contexto de execução da aplicação. Para fins de harmonização com o Guaraná, considera-se

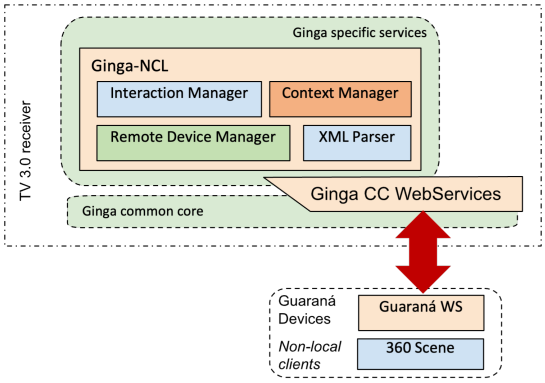

Figura 1. Arquitetura simplificada do receptor de TV 3.0 considerando os componentes necessários para suporte a múltiplos dispositivos.

o gerenciamento de múltiplos usuários realizado por esse componente durante a execução de uma aplicação. Por fim, o componente *XML Parser* carrega um documento NCL para sua execução.

Além de estender os componentes *Interaction Manager* e *XML Parser* para suporte ao evento de *View*, o *XML Parser* é estendido para o tratamento diferenciado de uma mídia com um conteúdo NCL 360 [14]. O player Guaraná, executando em um HMD, gera transições do evento de *View* para cada mídia executando numa cena 360. Esse evento segue a máquina de estados de evento de NCL conforme apresentado na Tabela 1.

Transição	Nome do Papel	Motivo
sleeping → occurring	onEnterView	Objeto entrou no campo de visão
occurring → sleeping	onExitView	Objeto saiu do campo de visão

Tabela 1. Comportamento do Evento de *View*.

O gerenciamento dos dispositivos remotos é feito pelo componente *Remote Device Manager* (RDM). O *RDM* mantém a lista de dispositivos conectados ao *middleware* Ginga, bem como as informações recebidas durante seu registro, como apresentado na Figura 2.

Toda mensagem recebida na comunicação entre dispositivos, conforme apresentado na Seção 3.1, é repassada pelo *CCWS* para o *RDM*. As mensagens vêm acompanhadas do *handle* do dispositivo remetente. Com base nessa informação, o *RDM* identifica o nó NCL correspondente para atualização do seu estado na aplicação. No caso da desconexão do dispositivo remoto, o *RDM* dispara uma transição *aborts* para o nó correspondente ao dispositivo desconectado. a partir desse ponto, quaisquer ações sobre o nó são ignoradas.

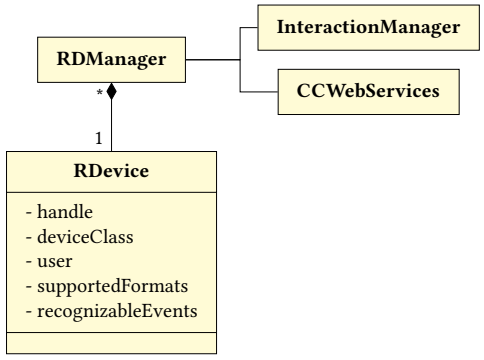

Figura 2. Diagrama de classes do *Remote Device Manager*.

No caso de um evento de *View*, o evento é repassado para o *Interaction Manager* para o tratamento necessário. Cabe ressaltar que, quando a própria notificação do evento não vem acompanhada da informação do usuário, o *RDM* incluirá essa informação ao notificar o *Interaction Manager*, caso a tenha.

No caminho contrário, *i.e.*, quando uma ação é executada sobre um nó, a ação é repassada ao *RDM*. A ação é então encaminhada ao *CCWS* junto com o *handle* do dispositivo para que a mensagem correspondente seja enviada pelo *CCWS*. A Tabela 2 resume o comportamento dos eventos associados a um nó executado em um dispositivo remoto.

3.3 Extensão do XML Parser para Cenas 360

Conforme proposto em [14], uma cena 360 a ser executada em um dispositivo Guaraná, é representada em NCL como uma mídia do tipo *application/x-ncl360* ou com a extensão *.ncl360* no *src* da mídia. Um exemplo de especificação desse tipo de mídia é apresentado na Listagem 7.

```
1 <media id="scene" src="*.ncl360">
2   <area id="anchorId" label="sceneElmId"/>
3 </media>
```

Listagem 7. Especificação de Cena 360 a ser executada em dispositigo Guaraná em uma aplicação NCL.

A associação entre os elementos NCL e NCL360 se dá da seguinte forma. A mídia NCL identificada por *scene* na Listagem 7 corresponde a toda a cena 360 a ser executada num dispositivo Guaraná. A mídia NCL deve definir uma âncora para cada elemento da cena 360 que deseja controlar. A associação de cada âncora com um elemento da cena é feita pelo atributo *label* que indica o identificador do elemento da cena 360 correspondente. Ações executadas sobre a âncora são comunicadas ao dispositivo Guaraná para que sejam feitas sobre o elemento da cena 360 correspondente, enquanto eventos de interação que ocorram com o elemento da cena 360 são notificados pelo Guaraná e tratados em NCL como tendo ocorridos com a âncora correspondente.

Transição	Tipo do Evento	Descrição
sleeping → occurring (starts)	Preparação	Início da preparação, com envio do conteúdo, se for um nó de mídia, ou propriedades de um efeito sensorial.
occurring → sleeping (stops)	Preparação	Fim da preparação, com carregamento do conteúdo terminado, se for um nó de mídia, ou configuração de atuadores, para o caso de um efeito sensorial.
sleeping → occurring (starts)	Apresentação	Início da execução.
occurring → sleeping (stops)	Apresentação	Fim da execução.
occurring → sleeping (aborts)	Apresentação	Perda de conexão com o dispositivo. A partir desse ponto quaisquer ações sobre o nó são ignoradas.
occurring → paused (pauses)	Apresentação	Dispositivo remoto entrou em modo de *stand-by*. A partir desse ponto quaisquer ações sobre o nó são armazenadas pelo *RDM*.
paused → occurring (resumes)	Apresentação	Dispositivo remoto saiu do modo de *stand-by*. Ações armazenadas pelo *RDM* são enviadas ao dispositivo.

Tabela 2. Comportamento dos Eventos de um mídia.

Uma vez identificada uma mídia NCL360, o parser se comunica com o *RDM* para obter a lista de dispositivos capazes de executar mídias desse tipo. A associação de uma mídia NCL360 com um dispositivo remoto da classe Guaraná se dá de 3 formas. Resumidamente, *(i)* sem qualquer identificação de dispositivo, *(ii)* identificando explicitamente um dispositivo e *(iii)* identificando dispositivos pelo usuário que o utiliza.

3.3.1 Sem Identificação Explícita do Dispositivo.

Nesta forma de associação, o autor da aplicação especifica apenas a mídia sem qualquer informação específica de dispositivo, conforme ilustrado na Listagem 8.

```
<media id="scene" src="cena360.ncl360">
   <area id="vdBand" label="bandVideo360"/>
</media>

<link>
   <bind role="onEnd" .../>
   <bind role="start" component="scene"
    interface="vdBand"/>
</link>
```

Listagem 8. Associação sem identificação explícita de dispositivo.

Nesse caso, a mídia *scene* e os elos nos quais ela participa serão replicados para cada dispositivo conectado. A informação sobre em qual dispositivo cada cena é executada será preenchida automaticamente pelo parser, de acordo com a lista de dispositivos fornecida pelo *RDM*. A Listagem 9 apresenta o resultado da replicação do trecho apresentado na Listagem 8 para dois dispositivos.

```
<media id="sceneRD0" src="cena360.ncl360">
   <property name="deviceClass" value="
    Guarana(0)"/>
   <area id="vdBandRD0" label="bandVideo360"
    />
</media>
<media id="sceneRD1" src="cena360.ncl360">
   <property name="deviceClass" value="
    Guarana(1)"/>
   <area id="vdBandRD1" label="bandVideo360"
    />
</media>

<link>
   <bind role="onEnd" .../>
   <bind role="start" component="sceneRD0"
    interface="vdBandRD0"/>
</link>
<link>
   <bind role="onEnd" .../>
   <bind role="start" component="sceneRD1"
    interface="vdBandRD1"/>
</link>
```

Listagem 9. Resultado da replicação do trecho apresentado na Listagem 8.

3.3.2 Identificação Explícita do Dispositivo.

Nesta forma de associação, o autor da aplicação especifica explicitamente o dispositivo em que a mídia deve ser executada, conforme ilustrado na Listagem 10.

```
<media id="scene1" src="cena360.ncl360">
   <property name="deviceClass" value="
    Guarana(0)"/>
   <area id="vdBand1" label="bandVideo360"/>
</media>

<media id="scene2" src="cena360.ncl360">
   <property name="deviceClass" value="
    Guarana(1)"/>
```

```
    <area id="vdBand2" label="bandVideo360"/>
</media>

<link>
    <bind role="onEnd" ... />
    <bind role="start" component="scene1"
     interface="vdBand1"/>
    <bind role="start" component="scene2"
     interface="vdBand2"/>
</link>
```

Listagem 10. Associação com identificação explícita de dispositivo.

Nesse caso, não há necessidade de o parser replicar as mídias *sceneX* para cada dispositivo conectado pois elas já explicitam o dispositivo em que irão ser executadas. No caso, em dispositivos da classe Guaraná e conforme a ordem de conexão desses dispositivos.

Note que, nesse exemplo, a aplicação prevê a possíbilidade de apenas dois dispositivos. Caso o *RDM* possua um número menor de dispositivos conectados, as mídias em excesso ficam no estado *sleeping* e o formatador ignorará qualquer ação de elos realizada sobre a mídia. Caso o *RDM* possua um número maior de dispositivos conectados, os dispositivos em excesso não apresentarão nenhum conteúdo.

3.3.3 Identificação Implícita do Dispositivo.

Nesta forma de associação, o autor da aplicação especifica implicitamente o dispositivo em que a mídia deve ser executada por meio do usuário nele conectado. Esse tipo de associação permite ainda o uso da funcionalidade multiusuário de NCL 4.0 [2] para adaptação do conteúdo.

Para exemplificar essa forma de associação serão definidos dois perfis de usuário: *adult* e *child*. Os usuários são associados a estes perfis de acordo com sua idade. No caso, maiores ou menores de 18 anos, respectivamente.

O trecho de código da Listagem 11 estabelece um comportamento alternativo do conteúdo a ser apresentado, conforme o perfil do usuário. Note que, nesse caso, a identificação do usuário está sendo feita a partir da identificação do usuário que realiza a interação de *View* com uma imagem apresentada na cena 360.

```
<media id="scene" src="cena360.ncl360">
    <area id="vdBand" label="bandVideo360"/>
    <area id="vdPlay" label="playVideo360"/>
    <area id="imWelc" label="welcomeImage"/>
</media>

<link>
    <bind role="onEnterView" component="scene
     " interface="imWelc">
        <bindParam name="user" value="adult"/>
    </bind>
    <bind role="start" component="scene"
     interface="vdBand"/>
</link>

<link>
    <bind role="onEnterView" component="scene
     " interface="imWelc">
        <bindParam name="user" value="child"/>
    </bind>
    <bind role="start" component="scene"
     interface="vdPlay"/>
</link>
```

Listagem 11. Associação com identificação implícita de dispositivo.

Nesse caso, a mídia *scene* e os elos nos quais ela participa serão replicados para cada dispositivo conectado. Após essa primeira etapa de replicação, o código da aplicação ainda identificará os usuários pelos perfis. Assim, a etapa seguinte é a replicação dos elos com base nos usuários cadastrados na TV, de acordo com a lista fornecida pelo *Context Manager*. Como apenas um usuário pode usar um dos HMDs por vez, a identificação do usuário interagindo garante que o conteúdo sendo entregue corresponde ao seu perfil.

3.4 Extensão do Parser para Outras Classes

Diferente da proposta apresentada na Seção 3.3, a execução de uma mídia em outros tipos de dispositivos requer a identificação explícita da classe do dispositivo. A proposta aqui apresentada não restringe os nomes das classes, permitindo que cada tipo de dispositivo possa definir seu nome de classe. A Listagem 12 apresenta um exemplo de execução de uma mídia em um dispositivo de uma classe identificada por *SecondScreen*.

```
<media id="M1" src="video.mp4">
    <property name="deviceClass" value="
     SecondScreen"/>
</media>
```

Listagem 12. Associação com outra classe de dispositivo.

Nesse caso, a mídia *M1* e os elos nos quais ela participa serão replicados para cada dispositivo conectado que pertença à classe *SecondScreen*, com base na lista mantida pelo *RDM*. Note que, no exemplo da Listagem 12, apenas o nome da classe é utilizado. Caso o autor especifique também o índice do dispositivo - *e.g.*, *SecondScreen(0)* -, a etapa de replicação não é realizada e a mídia será executada apenas no primeiro dispositivo desta classe conectado ao *middleware* Ginga.

Conforme apresentado na Seção 3.1, a mensagem de envio do conteúdo a ser apresentado contém as propriedades da mídia *M1*, permitindo assim que sua apresentação no dispositivo remoto possa ser configurada conforme especificado pelo autor da aplicação NCL.

4 Conclusão

Este artigo apresentou uma proposta de extensão do *middleware* Ginga, em seus subcomponentes Ginga-NCL e Ginga *Common Core WebServices* para suporte a múltiplos dispositivos. Apensar de tal funcionalidade já estar presente na linguagem NCL desde sua adoção como padrão para especificação de aplicações interativas no Sistema Brasileiro de TV digital, especificidades da descoberta, registro e comunicação entre TV e dispositivos remotos forma deixadas em aberto.

A proposta aqui apresentada se baseou naquelas apresentadas em [14] e [2], aceitas na chamada de TV 3.0, harmonizando as propostas do Guaraná e NCL 4.0 para execução de cenas 360 em dispositivos HMD conectados à TV. As novidades apresentadas permitem que múltiplos usuários com seus respectivos HMDs possam executar uma mesma aplicação, bem como uma generalização da forma de comunicação entre HMDs e o *middleware* Ginga, permitindo seu reúso por outros tipos de dispositivos.

Referências

[1] ABNT. 2018. Televisão digital terrestre - Codificação de dados e especificações de transmissão para radiodifusão digital Parte 11: Ginga CC WebServices - Especificação de WebServices do Ginga Common Core. ABNT NBR 15606-11:2018 standard.

[2] Fábio Barreto, Raphael S. de Abreu, Marina I. P. Josué, Eyre Brasil B. Montevecchi, Pedro Alves Valentim, and Débora C. Muchaluat-Saade. 2023. Providing multimodal and multi-user interactions for digital tv applications. *Multimedia Tools and Applications* 82, 4 (2023). https://doi.org/10.1007/s11042-021-11847-3

[3] Carlos Eduardo CF Batista, Luiz Fernando Gomes Soares, and Guido Lemos de Souza Filho. 2010. Estendendo o uso das classes de dispositivos Ginga-NCL. In *Anais Principais do XVI Simpósio Brasileiro de Sistemas Multimídia e Web*. SBC, 27–34.

[4] Pablo Cesar, Ishan Vaishnavi, Ralf Kernchen, Stefan Meissner, Cristian Hesselman, Matthieu Boussard, Antonietta Spedalieri, Dick C.A. Bulterman, and Bo Gao. 2008. Multimedia Adaptation in Ubiquitous Environments: Benefits of Structured Multimedia Documents. In *Proceedings of the Eighth ACM Symposium on Document Engineering*. ACM, New York, NY, USA, 275–284. https://doi.org/10.1145/1410140.1410201

[5] Romualdo Monteiro de Resende Costa, Marcio Ferreira Moreno, and Luiz Fernando Gomes Soares. 2009. Ginga-NCL: Supporting Multiple Devices. In *Proceedings of the XV Brazilian Symposium on Multimedia and the Web* (Fortaleza, Ceará, Brazil) *(WebMedia '09)*. Association for Computing Machinery, New York, NY, USA, Article 6, 8 pages. https://doi.org/10.1145/1858477.1858483

[6] Marcio Ferreira Moreno, Romualdo M. de R. Costa, and Marcelo F. Moreno. 2018. *Specifying Intermedia Synchronization with a Domain-Specific Language: The Nested Context Language (NCL)*. Springer International Publishing, Cham, 387–410. https://doi.org/10.1007/978-3-319-65840-7_14

[7] Luiz Fernando Gomes Soares, Marcio Ferreira Moreno, Carlos De Salles Soares Neto, and Marcelo Ferreira Moreno. 2010. Ginga-NCL: Declarative middleware for multimedia IPTV services. *IEEE Communications Magazine* 48, 6 (2010), 74–81. https://doi.org/10.1109/MCOM.2010.5473867

[8] David Gómez, Juan A. Núñez, Mario Montagud, and Sergi Fernández. 2018. ImmersiaTV: Enabling Customizable and Immersive Multi-Screen TV Experiences. In *Proceedings of the 9th ACM Multimedia Systems Conference* (Amsterdam, Netherlands) *(MMSys '18)*. Association for Computing Machinery, New York, NY, USA, 506–508. https://doi.org/10.1145/3204949.3209620

[9] Alan Livio Vasconcelos Guedes, Luís Feliphe Silva Costa, Fernando Santos De Mattos Brito, Ana Paula Nunes Guimaraes, José Ivan Bezerra Vilarouca Filho, Carlos Eduardo Coelho Freire Batista, and Guido Lemos De Souza Filho. 2013. GingaSpace: A Solution to Execute Multidevice Applications on Broadband TV Systems *(WebMedia '13)*. Association for Computing Machinery, New York, NY, USA, 305–308. https://doi.org/10.1145/2526188.2526239

[10] Michael E. Holmes, Sheree Josephson, and Ryan E. Carney. 2012. Visual Attention to Television Programs with a Second-Screen Application *(ETRA '12)*. Association for Computing Machinery, New York, NY, USA, 397–400. https://doi.org/10.1145/2168556.2168646

[11] Christopher Howson, Eric Gautier, Philippe Gilberton, Anthony Laurent, and Yvon Legallais. 2011. Second screen TV synchronization. In *2011 IEEE International Conference on Consumer Electronics -Berlin (ICCE-Berlin)*. 361–365. https://doi.org/10.1109/ICCE-Berlin.2011.6031815

[12] Francisco Pedro Luque, Iris Galloso, Claudio Feijoo, Carlos Alberto Martín, and Guillermo Cisneros. 2014. Integration of Multisensorial Stimuli and Multimodal Interaction in a Hybrid 3DTV System. *ACM Trans. Multimedia Comput. Commun. Appl.* 11, 1s, Article 16 (Oct. 2014), 22 pages. https://doi.org/10.1145/2617992

[13] Timothy Neate, Matt Jones, and Michael Evans. 2017. Cross-device media: a review of second screening and multi-device television. *Personal and Ubiquitous Computing* 21, 2 (2017), 391–405. https://doi.org/10.1007/s00779-017-1016-2

[14] Gabriel Souza, Daniel Silva, Matheus Delgado, Renato Rodrigues, Paulo R. C. Mendes, Glauco Fiorott Amorim, Alan L. V. Guedes, and Joel dos Santos. 2020. Interactive 360-Degree Videos in Ginga-NCL Using Head-Mounted-Displays as Second Screen Devices. In *Proceedings of the Brazilian Symposium on Multimedia and the Web* (São Luís, Brazil) *(WebMedia '20)*. Association for Computing Machinery, New York, NY, USA, 289–296. https://doi.org/10.1145/3428658.3430972

[15] M. Oskar van Deventer, Hans Stokking, Matt Hammond, Jean Le Feuvre, and Pablo Cesar. 2016. Standards for multi-stream and multi-device media synchronization. *IEEE Communications Magazine* 54, 3 (2016), 16–21. https://doi.org/10.1109/MCOM.2016.7432166

[16] Mikel Zorrilla, Njal Borch, François Daoust, Alexander Erk, Julián Flórez, and Alberto Lafuente. 2015. A Web-based distributed architecture for multi-device adaptation in media applications. *Personal and Ubiquitous Computing* 19, 5 (2015), 803–820. https://doi.org/10.1007/s00779-015-0864-x