Estendendo o uso das classes de dispositivos Ginga-NCL

Carlos Eduardo C. F. Batista
Departamento de Informática
PUC-Rio
Rio de Janeiro, RJ, Brasil
cbatista@inf.puc-rio.br

Luiz Fernando Gomes Soares
Departamento de Informática
PUC-Rio
Rio de Janeiro, RJ, Brasil
lfgs@inf.puc-rio.br

Guido Lemos de Souza Filho
Departamento de Informática
Universidade Federal da Paraíba
João Pessoa, PB, Brasil
guido@di.ufpb.br

ABSTRACT
Digital TV applications are able to use the resources made available by the devices connected to the Digital TV receiver through a Home Area Network (HAN). That brings advanced features to the Digital TV environment, since such devices may offer media display resources, multiple interaction mechanisms, amongst other features. NCL (Nested Context Language), which is the core language of Ginga-NCL Digital TV middleware, is provided with mechanisms for the distributed reproduction of its applications, so that multiuser applications can be developed, redefining the single user/remote control relationship. Using a hierarchical model for the application distribution and an abstraction for grouping devices (called device classes), it is possible to orchestrate the usage of the resources provided by the connected devices. This work aims to extend the original model proposed for the Ginga-NCL middleware, so that it can be made compatible with the different technologies used for device integration, such as UPnP and OSGi.

RESUMO
Aplicações de TV Digital podem utilizar recursos disponibilizados por dispositivos em redes domésticas (conhecidas como HAN - *Home Area Network*). Funcionalidades avançadas são trazidas para o ambiente de TV, visto que tais dispositivos podem oferecer recursos de visualização de objetos de mídia, múltiplos mecanismos de interação, entre outras funcionalidades. A linguagem NCL (*Nested Context Language*), que é a linguagem núcleo do *middleware* Ginga-NCL, possui mecanismos para a reprodução distribuída de suas aplicações, de forma que aplicações multiusuário possam ser desenvolvidas, eliminando assim o legado monousuário imposto pelo controle remoto. Através de um modelo hierárquico para distribuição de partes de uma aplicação e uma abstração para agrupar os dispositivos (as chamadas classes de dispositivos), é possível orquestrar a utilização de recursos providos pelos dispositivos conectados. Esse trabalho visa estender o modelo original proposto para o *middleware* Ginga-NCL, de forma a compatibilizá-lo com tecnologias usadas para integração de dispositivos, tais como UPnP e OSGi.

Categories and Subject Descriptors
I.7.2 [**Document Preparation**]: Languages and systems, Hypertext/hypermedia, Markup languages, Standards.

General Terms
Management, Design, Standardization, Languages.

Keywords
NCL, Ginga, TV Digital, múltiplos dispositivos, HAN.

1. INTRODUÇÃO
Com a popularização dos dispositivos computacionais portáteis e a evolução das tecnologias de comunicação em rede, surgem também protocolos e plataformas que têm por objetivo integrar os recursos abundantes em redes que possuem muitos dispositivos conectados. Um conjunto representativo de tais tecnologias está relacionado ao desenvolvimento de aplicações de multimídia distribuída [1] para redes de dispositivos domésticos (as HAN - *Home Area Network*).

Tecnologias como UPnP [2], OSGi [3] e Bluetooth [4] já são realidade comercial e, em linhas gerais, integram dispositivos que podem oferecer recursos para exibição de mídias (dispositivos de apresentação/saída: telas, celulares com suporte à reprodução audiovisual etc.) ou que podem oferecer interfaces para interação (dispositivos de interação/entrada: *joysticks*, acelerômetros, teclados).

No âmbito da TV Digital, existem esforços para a definição de plataformas de integração de dispositivos com funcionalidades específicas, como as oferecidas pelas especificações de *middleware* ARIB (*Association of Radio Industries and Businesses*) [5], do sistema japonês ISDB (*Integrated Services Digital Broadcasting*), e pelo *middleware* Ginga [6][7], do Sistema Brasileiro de TV Digital (SBTVD ou ISDB-Tb). O *middleware* Ginga dá suporte à integração com múltiplos dispositivos através de seus dois ambientes: um imperativo, chamado Ginga-J [6] e outro declarativo, chamado Ginga-NCL [7]. Este trabalho tem como foco o ambiente Ginga-NCL.

Através de um modelo hierárquico suportado pela linguagem NCL (*Nested Context Language*) [8], é possível orquestrar o uso dos recursos providos por dispositivos registrados junto ao receptor de TV Digital (em uma *Home Are Network – HAN* – IP, por exemplo), necessários na execução de uma aplicação. NCL oferece mecanismos para utilização desses recursos de acordo com a lógica de uma aplicação multimídia, sendo possível tratar, de forma declarativa, eventos de interação, sincronismo espaço

• *Extending the usage of Ginga-NCL device classes*

temporal entre mídias em reprodução e adaptação (de conteúdo e apresentação) de tal modo que possam ser desenvolvidas aplicações que individualizam a experiência de interatividade na TV Digital, superando a barreira mono-usuário imposta pelo controle remoto.

Nas especificações da NCL [8], os recursos dos dispositivos são acessíveis através do uso de *classes de dispositivos* (mecanismo que será detalhado na Seção 3), que podem ser associadas a objetos de mídia de um documento NCL [9]. A especificação de tais classes, porém, fica a cargo da implementação, bem como o registro de dispositivos dinamicamente nessas classes, por exemplo, através do uso de scripts em Lua (linguagem de script nativa do Ginga-NCL). A proposta aqui apresentada visa preencher essa lacuna, oferecendo uma alternativa inter-operável para o mecanismo de definição de classes de dispositivos Ginga-NCL, e visando aumentar a gama de possibilidades em aplicações de TV digital multi-dispositivos.

Assim, o objeto deste trabalho é um módulo para o *middleware* Ginga que agregue funcionalidades para o suporte à execução de aplicações NCL que utilizam recursos de múltiplos dispositivos secundários conectados a um dispositivo base. Para tanto, propõe-se um modelo para descrição de classes de dispositivos, que permita a associação dos recursos descritos a serviços (oferecidos por dispositivos secundários) nos quais o dispositivo base pode se tornar usuário. O modelo é não restritivo, no sentido que as classes definidas possam incorporar dispositivos compatíveis com as mais variadas plataformas (como as já mencionadas UPnP e Bluetooth, por exemplo). Os dispositivos secundários podem se registrar junto ao dispositivo base utilizando diferentes mecanismos e, de acordo com suas capacidades (serviços oferecidos), serem associados em uma ou mais classes de dispositivos [9].

A seguinte estrutura é utilizada no restante deste artigo: a segunda seção contextualiza o uso de múltiplos dispositivos no Ginga-NCL; a terceira seção discute algumas abordagens de multimídia distribuídas consideradas na elaboração do presente trabalho; na quarta seção são descritos os mecanismos de definição de classes de dispositivos propostos para o Ginga-NCL; a quinta seção apresenta detalhes da implementação do módulo; na sexta seção são apresentadas informações referentes à execução de cenários de validação; e, finalmente a sétima seção discorre sobre as pesquisas relacionadas em andamento.

2. MÚLTIPLOS DISPOSITIVOS EM TV DIGITAL

O objetivo desta seção é estabelecer, em linhas gerais, quais são as funcionalidades atualmente oferecidas para os desenvolvedores de aplicações interativas multi-dispositivos nas diferentes plataformas de TV Digital, de forma a evidenciar as novas funcionalidades possíveis na extensão de classes de dispositivos propostas para o *middleware* Ginga-NCL.

Os ambientes imperativos dos *middlewares* Ginga (Ginga-J) [6], e ISDB ARIB (ARIB-J STD-B23) [5] oferecem algum suporte a múltiplos dispositivos. Já o *middleware* do sistema DVB (*Digital Vídeo Broadcast*), o MHP (*Multimedia Home Platform*) [11], possui trabalhos relacionados visando a integração com plataformas para aplicações multi-dispositivo, porém suas especificações não oferecem funcionalidades específicas para esse contexto.

A relação entre *middlewares* de TV Digital com plataformas para integração de dispositivos foi desenvolvida entre o *middleware* europeu DVB/MHP e o *framework* para aplicações em redes residenciais OSGi (*Open Service Gateway initiative*) [3], de forma que aplicações possam usar recursos associados a serviços dinamicamente registrados. O *framework* OSGi utiliza também a plataforma Java (compartilhando sua máquina virtual com o DVB/MHP) e oferece funcionalidades voltadas para automação residencial. As abordagens [12] que utilizaram o OSGi não possuem foco em apresentação e sincronização de mídias distribuídas. Elas basicamente harmonizam os modelos de ciclo de vida de aplicações OSGi e os *Xlets* MHP [12].

Uma API é definida pela especificação ARIB STD-B23 (*Application Execution Engine Platform for Digital Broadcasting*) para a integração de dispositivos conectados com o receptor de TV Digital (dispositivo base). O pacote núcleo dessa API é denominado *jp.or.arib.tv.peripheral* [5]. O pacote é composto de classes e interfaces com funcionalidades para descoberta e registro de dispositivos, obtenção de propriedades e estado dos dispositivos, além de funcionalidades de leitura e escrita para comunicação. Possui ainda suporte a mecanismos do UPnP (pacote *jp.or.arib.tv.peripheral.protocol*), e registro de dispositivos através de Bluetooth e interfaces USB. O UPnP [2] foi criado por um Fórum de empresas da área de dispositivos portáteis e tem por objetivo a definição de uma plataforma para interoperabilidade entre dispositivos, utilizando padrões industriais. Em uma rede doméstica IP, UPnP usa padrões como HTTP, HTML, XML e SOAP para descoberta, descrição, apresentação e controle de dispositivos.

A especificação da API para integração de múltiplos dispositivos do ambiente imperativo Ginga-J [10] do *middleware* Ginga oferece uma classe que atua como gerente de dispositivos (*GRemoteDeviceManager*) para recuperação da lista de dispositivos (*GRemoteDevice*) conectados e registrados junto ao receptor de TV Digital (dispositivo base). O objeto definido pela classe *GRemoteDevice* é uma representação abstrata de um dispositivo de interação, oferecendo métodos que possibilitam a recuperação de informações acerca dos dispositivos registrados (tipo do dispositivo, funcionalidades disponíveis etc.), bem como explorar suas funcionalidades disponíveis (recursos de gravação de áudio, de vídeo, captura de imagens). Cada dispositivo possui uma classe contêiner, da API gráfica associada (classe *DTVContainer* na API JavaDTV) que pode ser utilizada para compor interfaces a serem exibidas de forma transparente nos dispositivos. Ouvintes (*GRemoteDeviceActionListener*) podem ser registrados nos dispositivos para notificação de eventos de dados e de interação do usuário. Eventos podem encapsular dados de áudio e vídeo, requisições de certos tipos de dados ou códigos de teclas pressionadas.

Diferente de todos os outros *middlewares* mencionados, Ginga-NCL oferece suporte declarativo a múltiplos dispositivos. A arquitetura do *middleware* Ginga define que seus dois ambientes de execução (Ginga-NCL e Ginga-J) de aplicações devem utilizar os recursos de um núcleo comum (*Ginga-CC*) [6][7]. Assim, a especificação de uma API em Lua, parte da proposta objeto deste artigo, permite que funcionalidades equivalentes às oferecidas pela API Ginga-J também sejam oferecidas pelo Ginga-NCL. Dessa forma, os cenários viabilizados pela plataforma de integração de múltiplos dispositivos do *middleware* Ginga serão suportados em sua totalidade pelo ambiente Ginga-NCL sem a necessidade de utilização da ponte com o ambiente Ginga-J [6].

Adicionalmente, Ginga-NCL oferece facilidades para tratamento declarativo de múltiplos dispositivos e funcionalidades não existentes no Ginga-J, como o tratamento de dispositivos em classes.

3. MÚLTIPLOS DISPOSITIVOS NA NCL

O *middleware* declarativo Ginga-NCL [9] define dois tipos de classes de dispositivos secundários de exibição para a apresentação de aplicações NCL distribuídas. No primeiro tipo, um mesmo conteúdo é apresentado nos dispositivos registrados na classe, sobre controle de navegação único; o segundo tipo permite que os dispositivos registrados na classe controlem a apresentação do conteúdo de forma independente, permitindo um controle de navegação completamente individualizado. O primeiro tipo define uma classe chamada passiva, enquanto o segundo define uma classe ativa. Subclasses podem ser definidas, com base nos recursos exigidos dos dispositivos, como extensões das classes ativa e passiva.

Dispositivos se registram em classes junto a um dispositivo base (que é único) ou dispositivos descendentes (recursivamente registrados) do dispositivo base, e formam um domínio. É chamado de dispositivo pai do dispositivo registrado, aquele onde o registro é efetuado. No modelo de controle hierárquico da NCL, independente do tipo de classe, dispositivos em uma classe só podem exibir conteúdos de objetos de mídia vindos do mesmo dispositivo pai. Ou seja, uma classe não pode ter mais de um dispositivo pai por vez. Adicionalmente, o dispositivo base não pode se registrar em nenhum outro dispositivo do domínio. Portanto, não é possível ter um dispositivo como ascendente ou descendente de si mesmo.

O dispositivo base orquestra a exibição da aplicação utilizando um modelo hierárquico. Um dispositivo pai deve transmitir amostras de áudio e/ou vídeo para serem exibidas pelos dispositivos que gerencia e estão registrados em classes passivas, de forma que todos esses dispositivos apresentem sempre o mesmo conteúdo. No caso de classes ativas, o dispositivo pai deverá transmitir para os dispositivos que gerencia (dispositivos filhos) partes da aplicação (objetos de mídia, podendo enviar inclusive objetos de mídia com código NCL) para que sejam decodificadas, apresentadas e localmente controladas. É importante ressaltar que o modelo permite a criação de subdomínios a partir de aplicações em dispositivos ativos, de forma que as mesmas possam ser distribuídas novamente entre outros dispositivos (ativos e passivos) recorrentemente.

As funcionalidades da NCL relacionadas à utilização de recursos de dispositivos secundários são suportadas por módulos específicos na arquitetura da implementação de referência do Ginga-NCL (Figura 1) [9]. Os módulos que compõem o componente de integração com múltiplos dispositivos são responsáveis pelo registro de dispositivos e pela comunicação entre dispositivos, para apresentação de um documento multimídia distribuído. O *Controlador de Serviços* controla os serviços dos dispositivos em um domínio, que é definido pelo conjunto dos dispositivos em suas classes. O *Gerenciador de Dispositivos* [9] controla o registro de dispositivos junto a um conjunto de classes de dispositivo pré-definidas: as classes “0”, “1” e “2”. A classe "0” é associada apenas ao dispositivo base pai, que executa o código NCL inicial. A classe passiva "1" agrega dispositivos registrados que devem ser capazes de receber fluxos de mídia vídeo e áudio decodificados, prontos para serem exibidos. A classe ativa "2" registra dispositivos capazes de receber *qualquer* objeto de mídia NCL especificado pelo padrão Ginga-NCL [9] (incluindo objetos de mídia com código NCL), para apresentação de forma independente. Outras classes podem ser definidas, com base nos recursos exigidos dos dispositivos, restringindo, por exemplo, os tipos de objetos de mídia que os dispositivos registrados devem dar suporte de exibição.

Dispositivos registram-se dinamicamente nas classes e a definição dos serviços exigidos para registro em uma classe pode também ser feito dinamicamente, por exemplo, a partir do uso de scripts Lua [9], porém, a API Lua com esse intuito não é definida pela implementação de referência do Ginga-NCL atual. O modelo de descrição de classes não é alvo das especificações do Ginga-NCL, podendo ser definido pelo fabricante do dispositivo base, sendo atrelado a uma implementação específica do Ginga-NCL.

O *Gerenciador de Dispositivos* guarda todas as informações relativas à configuração dos dispositivos para todas as classes registradas, bem como a associação dos dispositivos às classes. Informações sobre os recursos dos dispositivos que serão utilizados pelas aplicações NCL (ex. tamanho de tela, número de canais de áudio etc.) estão associados a um conjunto de variáveis de ambiente do Ginga-NCL, que podem ser acessadas através do seu "nó de *settings*" (nó de definições) [9]. Essas informações podem ser utilizadas para adaptação de aplicações às características de uma determinada classe de dispositivos [9].

O *Controlador de Serviços* [9] é responsável por realizar as ações necessárias para que objetos de mídia possam ser apresentados em dispositivos secundários, registrados nas classes ativas e passivas. Para as classes passivas, o controlador deverá acessar o conteúdo já decodificado do objeto de mídia a ser apresentado remotamente (o mapa de memória de vídeo para um objeto de mídia visual, por exemplo) e utilizar o componente de *Transporte* para entrega do objeto decodificado. Já para classes ativas, o controlador envia aos dispositivos da classe objetos de mídia por completo, em sua codificação original (uma aplicação NCL e suas mídias associadas, por exemplo).

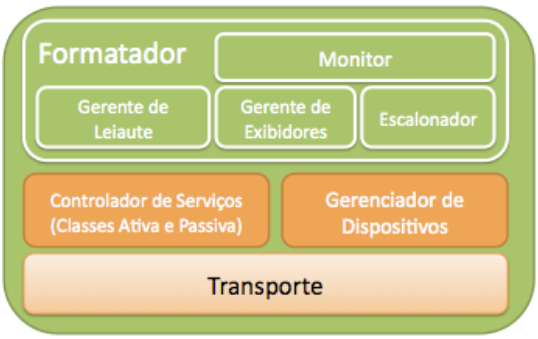

Figura 1. Módulos utilizados para suporte a apresentações através de múltiplos dispositivos [9].

A implementação de referência atual do Ginga-NCL [9] define mecanismos não flexíveis para registro de dispositivos. A implementação utiliza um mecanismo de descoberta de serviços de acordo apenas com as duas classes pré-definidas. Embora feito para ser estendido, o mecanismo não viabiliza a definição dinâmica de outros tipos de classe, e o uso de outras técnicas de registro de dispositivo e busca de serviços em ambientes móveis e

pervasivos, como proporcionado, por exemplo, pelo UPnP e Bluetooth. Através da atual implementação, o dispositivo pai recebe apenas notificações dos eventos para o formatador do Ginga-NCL (eventos de apresentação, de seleção e de atribuição), ou seja, os dispositivos secundários filhos atuam como exibidores Ginga-NCL remotos. Não é possível, por exemplo, reproduzir no dispositivo base pai um fluxo de vídeo ou um trecho de áudio capturado por um dispositivo secundário filho, apesar desse recurso ser viável através da linguagem NCL (identificando o fluxo através do atributo *src* do nó media ou utilizando os recursos oferecido pelo Ginga-J). Adicionalmente, não é possível identificar a origem dos eventos de seleção e atribuição (identificador único de dispositivo), diminuindo sensivelmente as possibilidades para desenvolvimento de aplicações multiusuário. Cabe ressaltar que tais limitações não são do *middleware* Ginga-NCL, mas sim de sua implementação de referência atual. Implementações comerciais podem apresentar soluções próprias para os problemas aqui mencionados.

A ausência de tais funcionalidades motivaram o desenvolvimento do presente trabalho, para que o módulo de comunicação com múltiplos dispositivos da implementação de referência do *middleware* Ginga-NCL permita explorar a linguagem NCL em todo seu potencial.

4. DEFINIÇÃO DE CLASSES DE DISPOSITIVOS

O modelo de definição de classes de dispositivos deve ser genérico o suficiente para acomodar os diferentes parâmetros relacionados com as diversas plataformas de integração de dispositivos conectados. A abordagem adotada para a descrição de classes de dispositivos, proposta para o Ginga-NCL, possui os requisitos apresentados abaixo:

- Possibilidade de estabelecer um limite de quantidade de dispositivos (máximo e mínimo) para uma classe, associada a um mecanismo de identificação único para cada dispositivo da classe.
- Provisão de mecanismos para a descrição das capacidades específicas que os dispositivos filhos devem possuir, para poderem ser associados pelo dispositivo pai a uma classe. Essas capacidades podem utilizar semântica de descrição própria a uma plataforma específica, sendo que a requisição por tais informações deve ser suportada pela plataforma.
- Possibilidade de associação de parâmetros e estados às classes, de forma que o registro a uma classe possa ser condicionado a um parâmetro arbitrário disponibilizado pelo dispositivo (por exemplo, a versão de um dos exibidores suportados, o valor de uma assinatura digital, a posição geográfica do dispositivo etc.). Novamente, a semântica de descrição dos parâmetros poderá ser própria a uma plataforma específica.

A possibilidade de limitar a quantidade de dispositivos registrados é importante para aplicações que só façam sentido com um limite máximo ou mínimo de participantes. Assim, ações relacionadas a um objeto de mídia só serão notificadas pelo dispositivo pai aos dispositivos filhos quando a classe contiver uma quantidade de dispositivos compatíveis com sua definição (por exemplo, usuários de uma classe recebem a sinalização para inicialização de um objeto de mídia relacionado só quando o número mínimo de dispositivos conectados é alcançado). A identificação única de dispositivos poderá ser utilizada através da API Lua (apresentada na Seção 5.1), e também na definição de elos (elemento *link*) contendo objetos de mídia de uma classe registrada, através do elemento *bindParam* com o atributo *name* contendo *"device_id"*, e o atributo *value* a identificação do dispositivo. O objetivo da identificação é ser possível limitar a notificação de eventos apenas para um dispositivo secundário filho, dentre os registrados na classe. Cada dispositivo em uma classe possuirá, então, um identificador único, acessível de acordo com sua ordem de registro (exemplo: a variável *ClasseAtivaNCL.device*(1) contém o identificador único para o primeiro dispositivo da classe *ClasseAtivaNCL*).

A descrição das capacidades de uma classe é utilizada pelo dispositivo base pai quando da distribuição exibição de uma aplicação, através de chamadas para os serviços do dispositivo filho. Note que a descrição das capacidades de uma classe deve ser comparada com a descrição das capacidades (serviços oferecidos) do dispositivo quando do registro em uma classe. O uso de mesma semântica e sintaxe para essas especificações simplifica muito a comparação.

Tecnologias de descrição de capacidades de dispositivos estão comumente atreladas às plataformas para integração de dispositivos, sendo utilizadas para descoberta e entrega de serviços de forma automática, em ambientes de execução para aplicações móveis e pervasivas [13]. Plataformas como Jini [14], OSGi [3] e UPnP [2] oferecem mecanismos para que seus serviços sejam descritos, utilizando semântica própria e especializada para seu contexto de uso [13].

UAProf (*User Agent Profile*) [15] é uma implementação concreta do *framework* W3C CC/PP (*Composite Capabilities/Preference Profiles*) [16], que utiliza RDF (*Resource Description Framework*) para descrição de capacidades de dispositivos móveis genéricos, e que pode ser também utilizada na descrição de classes. A descrição é composta de informações como: tamanho de tela, capacidades multimídia exigida (exibidores e geradores de mídia), conjunto de caracteres que devem ser suportados, entre outras. As especificações foram originalmente concebidas para que as informações do perfil do dispositivo fossem enviadas junto ao cabeçalho de suas requisições HTTP, de forma que o conteúdo a ser recebido pudesse ser adaptado pelo servidor respondendo à requisição. No entanto, o modelo CC/PP também é suficiente para definição de parâmetros e estados associados a classes de dispositivos. A associação de parâmetros dinâmicos em ontologias baseadas no *framework* CC/PP já foi proposta em trabalhos relacionados à adaptação de conteúdo sensível a contexto [17].

Neste trabalho, uma extensão da especificação CC/PP é definida para descrição das classes de dispositivos NCL, na implementação *UAProf* [15]. A extensão contempla o acréscimo de alguns elementos específicos (Tabela 1) do modelo de múltiplos dispositivos proposto para o Ginga-NCL, de forma a incorporar outras semânticas de descrição de capacidades (como a descrição UPnP [2]). Para a definição das classes de dispositivos NCL são utilizados *HardwarePlatform*, *SoftwarePlatform* e *NetworkCharacteristics* do UAProf (que são subclasses do componente genérico definido pelo CC/PP), e uma nova classe *DeviceGroup*. Um novo *namespace* é usado para a definição da nova classe, bem como os novos parâmetros específicos do Ginga-NCL e atributos genéricos para comportar definições de capacidades com semântica associada a outro modelo de descrição diferente do UAProf.

A Tabela 1 sumariza as propriedades adicionadas aos elementos definidos pelo UAProf. Na tabela seguinte (Tabela 2) são apresentados os atributos que um elemento *DeviceGroup* suporta.

Tabela 1. Atributos acrescentados às classes definidas pelo UAProf

Classe: SoftwarePlatform		
Atributo	**Tipo**	**Exemplo**
gncl:supportedNCLProfiles	rdf:Bag	"NCL3.0/EDTV", "NCL3.0/BDTV"
gncl:supportedServices	rdf:Bag	"UPnP MediaRenderer", "NCL Active Class Service"
gncl:softwareParams	rdf:Bag	"KEY=123456"
Classe: NetworkCharacteristics		
gncl:registrationMethod	Literal	"UPnP", "Bluetooth"
gncl:supportedProtocols	rdf:Bag	"HTTP", "HTTPS", "UDP", "RTP"
gncl:networkParams	rdf:Bag	"IP=10.0.1.1"

Tabela 2. Definição da classe DeviceGroup

Classe: DeviceGroup	
Atributo	**Tipo**
gncl:maxDevices	Número
gncl:minDevices	Número
gncl:Hardware	prf:HardwarePlatform
gncl:Software	prf:SoftwarePlatform
gncl:Network	prf:NetworkCharacteristics

A classe *DeviceGroup* possui os atributos que determinam os limites máximo e mínimo para quantidade de dispositivos que poderão se registrar em uma classe, sendo responsável por encapsular os três elementos da descrição de perfil UAProf. O componente *SoftwarePlatform* (Tabela 1) foi acrescido de três novos atributos. O atributo *supportedNCLProfiles* deve conter os perfis NCL [8] que os dispositivos da classe devem suportar; o atributo *supportedServices* agrega identificadores para os serviços que devem ser providos pelos dispositivos filhos da classe que o dispositivo base pai pode usar; e o atributo *softwareParams* especifica parâmetros arbitrários para a utilização desses serviços. O componente *NetworkCharacteristics* também recebeu três novos atributos: o atributo *registrationMethod,* que armazena o identificador do mecanismo de registro utilizado pelo dispositivo secundário filho junto ao dispositivo base pai; o atributo *supportedProtocols* define quais são os protocolos que podem ser utilizados para a comunicação com o dispositivo secundário filho; finalmente, o atributo *networkParams* define parâmetros arbitrários para o uso dos serviços de rede dos dispositivos secundários filhos.

5. IMPLEMENTAÇÃO DA EXTENSÃO

Esta seção apresenta o componente de software a ser incorporado junto ao Ginga-NCL. Esse componente torna o Ginga-NCL capaz de utilizar as definições de classe de dispositivos apresentadas na seção anterior, aumentando as possibilidades para o desenvolvimento de aplicações NCL para exibição em múltiplos dispositivos de exibição, e ainda com a possibilidade de compatibilização com as diferentes plataformas para aplicações distribuídas já mencionadas.

Os requisitos de alto nível definidos para o módulo de comunicação com os múltiplos dispositivos, proposto para a implementação de referência do *middleware* Ginga-NCL, são os seguintes:

- Suporte a definição de classes de dispositivos a partir de um modelo de descrição genérico, extensível e representativo (conforme apresentado na seção anterior).
- Possibilidade de associação dinâmica de dispositivos secundários filhos junto às classes definidas no dispositivo pai, através de um mecanismo genérico de registro.
- Suporte a utilização de serviços e canais de comunicação diferentes, para registro e controle dos dispositivos secundários filhos.

De acordo com os requisitos acima, foram elaboradas as definições arquiteturais para um novo módulo de suporte à comunicação com múltiplos dispositivos para a implementação de referência do *middleware* Ginga-NCL. Na Figura 2 é apresentado um diagrama de segmentação em camadas contendo o novo módulo de integração com múltiplos dispositivos do Ginga-NCL, compreendendo as principais entidades funcionais relacionadas.

Figura 2. Arquitetura de software do módulo de integração com múltiplos dispositivos com suporte à extensão de classes

Os componentes do módulo de integração com múltiplos dispositivos devem ser responsáveis por realizar as etapas necessárias para descoberta, registro e utilização dos serviços oferecidos pelos dispositivos filhos, de diferentes plataformas, de forma que tais serviços possam ser utilizados por uma aplicação NCL. Os novos elementos concebidos neste trabalho (apresentados na Figura 2) são apresentados a seguir.

– *Serviço de Dispositivo* – componente que oferece uma abstração através da qual os serviços disponibilizados pelos dispositivos secundários filhos já conectados ao dispositivo pai podem ser utilizados, para a exibição de objetos de mídia

da aplicação NCL. Para tal, deve-se estabelecer uma conexão (através do *Canal de Comunicação*) com o dispositivo filho e também alocar recursos do dispositivo pai (através do componente *Proxy de Recurso*), que serão utilizados pelo serviço durante a comunicação com o dispositivo filho.

- *Canal de Comunicação* – entidade responsável por oferecer mecanismos para o envio e recebimento de dados entre o dispositivo pai e os dispositivo filhos.
- *Proxy de Recurso* – elemento responsável por prover uma abstração para acesso a recursos locais do dispositivo pai, que serão utilizados durante a comunicação com dispositivos secundários filhos. Exemplos de tais recursos: arquivos armazenados localmente, acesso a captura de fluxos (*streams*) de vídeo etc.
- *Interface de Registro* – interface responsável por oferecer mecanismos genéricos de registro de dispositivos filhos e de seus perfis capacidades. Tal interface deverá ser implementada por componentes que adaptarão a semântica e os mecanismos de descrição utilizados pelas diferentes plataformas de integração com múltiplos dispositivos, compatibilizando com a extensão ao modelo UAProf proposta neste artigo. Os dispositivos registrados, de acordo com seu perfil de capacidades, poderão ser agrupados, pelo *Gerente de Classes de Dispositivos,* nas classes associadas a uma aplicação NCL em execução.
- *Gerente de Classes de Dispositivos* – módulo responsável por gerenciar as classes de dispositivos disponíveis, a partir de uma descrição RDF[1] das classes associadas a uma aplicação NCL e do registro dinâmico de classes (oferecidos pela *API Lua*). O módulo realiza a associação de dispositivos filhos a uma ou mais classes.
- *Monitor de Dispositivos* – componente responsável pela comunicação com o formatador do *middleware* Ginga-NCL. É o componente responsável por traduzir os eventos do formatador para sinais de controle dos serviços oferecidos pelos dispositivos secundários filhos.

O dispositivo secundário registra seu perfil de capacidades junto ao dispositivo base, através da "Interface de Serviço". O dispositivo base (através do componente o "Gerente de Classes de Dispositivos") associa o agora dispositivo filho às classes em que as capacidades descritas no seu perfil são iguais ou superiores com as capacidades associadas às classes suportadas pelo dispositivo base pai. Um dispositivo pode pertencer a mais de uma classe simultaneamente. A implementação apresentada nesta seção permite também que um dispositivo possa selecionar quais, dentre as disponíveis classes, ele irá associar-se. Um serviço específico deve ser suportado pelo dispositivo (identificador "NCLClassSelection"), no qual o dispositivo pai envia ao dispositivo filho, através de um canal de comunicação TCP, os metadados das classes acessíveis, e o dispositivo filho responde com os identificadores das classes selecionadas. O dispositivo irá criar (através do "Gerente de Classes de Dispositivos") e controlar (a partir do "Monitor de Dispositivos") os componentes de serviços de dispositivos, de acordo com as classes de dispositivos em operação e com os seus dispositivos registrados.

[1] Manipulação RDF implementada através das bibliotecas *Redland RDF* [21].

A camada de software do núcleo comum do Ginga utiliza a linguagem C++, linguagem também utilizada no desenvolvimento do componente objeto deste artigo. As definições arquiteturais aqui descritas são genéricas o suficiente para que também possam ser utilizadas para o suporte à API de integração com dispositivos do *middleware* Ginga-J [10], com acesso às funcionalidades do componente através do framework JNI (*Java Native Interface*). Por se tratar de uma linguagem de extensão, Lua pode acessar naturalmente código C/C++ através da sua pilha de comunicação global [18], o que viabiliza a elaboração de uma API Lua para controle imperativo, conforme descrito na subseção seguinte.

5.1 API Lua

Uma API Lua foi definida para oferecer suporte à criação dinâmica de classes e oferecer outras funcionalidades relacionadas ao modelo de integração de dispositivos do Ginga-NCL. A API é parte das especificações LuaTV [19], sendo composta por dois módulos, chamados de *devicemanager* e *deviceservice*, além de utilizar uma nova classe de eventos NCLua [20], chamada *device*.

O módulo *devicemanager* é responsável pelo registro de classes através de informações RDF serializadas (função *devicemanager:registerClass*), pelo acesso aos metadados das classes de dispositivos registradas (função *devicemanager:classMetadata*) e dos seus dispositivos filhos (função *devicemanager:deviceMetadata*). Ela também é responsável por notificar eventos de entrada e saída de dispositivos em classes – eventos da classe "*device*", do tipo "*join_class*" e "*leave_class*", carregando o identificador único do dispositivo e da classe.

Através do módulo *devicemanager* é possível instanciar um elemento que oferece acesso, com alto grau de abstração, aos serviços oferecidos pelos dispositivos secundários filhos. O módulo *deviceservice* oferece uma interface genérica para submissão de requisições (função *deviceservice:request*) e recebimento de dados junto aos serviços dos dispositivos secundários filhos. O recebimento de informações do serviço é feito de forma assíncrona, utilizando a notificação de eventos da classe "*device*" com tipo "*data*", contendo o identificador único do dispositivo, os dados recebidos e o identificador do serviço associado.

6. CENÁRIOS DE USO E TESTES

Para os testes foram planejados dois cenários de aplicações, utilizando duas definições de classes distintas. Os cenários foram elaborados de forma a validar os requisitos definidos para a proposta deste trabalho. As classes são acessadas pelo componente *Gerente de Classes de Dispositivos* a partir de um arquivo de definições RDF. Os identificadores das classes (atributo *ID* do elemento de tipo RDF *DeviceGroup*) são associados ao elemento *regionBase* (parâmetro *device*), que contém as regiões que agregam os objetos de mídia a serem apresentados nos dispositivos secundários filhos registrados na classe.

O primeiro cenário considera a descrição da classe de dispositivos ativa do Ginga-NCL [9], na qual o dispositivo filho recebe uma aplicação NCL, parte de uma aplicação em execução no dispositivo base pai. O dispositivo pai controla a execução da aplicação no dispositivo filho, enquanto o dispositivo filho notifica os eventos de transição, seleção e atribuição do objeto de mídia NCL que controla, e que são relacionados pelo dispositivo pai. No segundo cenário, o dispositivo base pai atua como um

UPnP MediaServer [22], para compartilhamento em rede de objetos de mídia, e aceita registro de dispositivos *UPnP MediaServer Control Point* [22]. As subseções seguintes detalham a implementação dos testes.

Em todos os testes, foi utilizado, como dispositivo base, um *MiniPC* AOpen (Linux) e, como dispositivos secundários, dois celulares HTC Magic (com a API *Android* 1.5) e um computador portátil *MacBook* (Mac OS X). O primeiro cenário utiliza uma classe de dispositivos que engloba apenas os celulares; a classe utilizada para o segundo cenário inclui serviços suportados pelos celulares e pelo computador portátil.

6.1 Classe Ginga-NCL Ativa

Para a elaboração desse cenário foram desenvolvidos componentes adicionais compatíveis com a arquitetura apresentada na seção anterior, que reproduzem funcionalmente o modelo de classes ativas proposto originalmente para o Ginga-NCL [9].

A classe de dispositivos descrita e carregada junto ao *Gerente de Classes de Dispositivos* associa um serviço à execução de documentos NCL nos dispositivos secundário filhos (vide Figura 2). Os componentes especializados que podem ser utilizados pelo módulo de integração com múltiplos dispositivos foram os seguintes:

- *Registro Ginga-NCL* – implementa a *Interface de Registro*, e registra as características e propriedades disponibilizadas pelos dispositivos secundário filhos, junto ao dispositivo base pai, de acordo com o procedimento originalmente proposto para a implementação de referência Ginga-NCL [9].
- *Serviço Ginga-NCL Ativo* – módulo responsável por tratar as requisições vindas do *Monitor de Dispositivos*, a partir de chamadas padronizadas advindas do formatador. Esse serviço faz uso de dois componentes internos: o *Canal TCP*, que é uma implementação da interface *Canal de Comunicação* utilizando o protocolo TCP para envio e recepção de eventos, e também para envio dos objetos de mídia serializados (usando *Base64*); e o *Proxy Contêiner NCL*, que permite o acesso ao sistema de arquivos que contém os documentos NCL e seus objetos de mídia associados, para que sejam enviados para os dispositivos secundário filhos.

A Figura 3 ilustra o cenário de execução da aplicação, destacando os componentes utilizados pelo módulo de integração com múltiplos dispositivos.

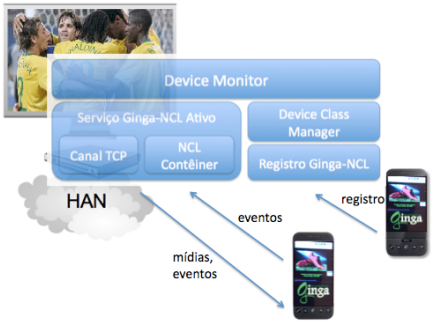

Figura 3. Cenário de execução da aplicação "Pênalti interativo".

O serviço Classe Ativa Ginga-NCL é oferecido por uma implementação Ginga-NCL para a plataforma *Android*, desenvolvida em conjunto com alunos do Departamento de Informática da UFES a partir da versão Java do Emulador Ginga-NCL [23], e foi instalado em celulares HTC Magic.

Dentre as aplicações NCL utilizadas para testar o cenário, uma, chamada de "Pênalti Interativo", possui um conjunto representativo de funcionalidades. Após o registro e a entrega dos objetos de mídia relacionados, o dispositivo base pai inicia nos dispositivos secundários filhos um objeto de mídia NCL que oferece, durante um período fixo de tempo, a possibilidade de escolha de qual a direção do chute e para qual a direção pulará goleiro em uma simulação de pênalti. A escolha dessas opções é feita através de eventos de atribuição em variáveis globais, que são notificados para o dispositivo base pai – há claramente a necessidade de limitação de usuários e de identificação unívoca dos dispositivos, para que cada um exerça um papel distinto (batedor e goleiro). Uma animação com o resultado das escolhas (de acordo com a direção do chute e do pulo do goleiro) é então exibida no dispositivo base pai, utilizando um elemento *switch* contendo as animações prováveis, associadas a regras definidas a partir das variáveis globais alteradas pelos dispositivos filhos.

6.2 Classe UPnP MediaServer Control Point

A classe de dispositivos descrita com identificador *UPnP_MSCP*, é carregada junto ao Gerente de Classes de Dispositivos, e deve ser composta por dispositivos que provêem o serviço *UPnP MediaServer Control Point*. Os componentes associados ao módulo de integração[2]. com múltiplos dispositivos foram os seguintes:

- *Registro UPnP* – registra as características dos *MediaServer Control Point* registrados através da *Interface de Registro*
- *Serviço UPnP* (*MediaServer*) – oferece um serviço de compartilhamento de objetos de mídia UPnP *MediaServer*. Tal serviço faz uso de dois componentes internos: o *Proxy de Mídias*, que dá acesso ao sistema de arquivos no qual se encontram as mídias a serem compartilhadas pelo dispositivo pai; e o *Canal UPnP*, através do qual são compartilhadas as mídias com os dispositivos filhos. Exemplo: o serviço irá adicionar um vídeo na lista de compartilhados, quando o nó de mídia que o referenciar na aplicação em execução no dispositivo base for inicializado (transição de estado de evento *start*). O compartilhamento é encerrado quando houver a transição de estado de evento NCL *stop*.

Dois clientes UPnP *MediaServer Control Point* foram utilizados nos testes: o AndroMote [24] para a plataforma Android (celulares HTC) e o VLC (*Videolan Client*) [25] para a plataforma Mac OS X (MacBook). A aplicação de teste para o serviço proposto compartilha um conjunto de vídeos dos melhores momentos de um evento esportivo. Os nós de mídia dos vídeos e seus elos podem ser adicionados, no caso de um evento ao vivo, através de comandos NCLEdit [27].

7. PESQUISAS EM ANDAMENTO E CONCLUSÃO

Este artigo propõe um módulo para a implementação de referência do *middleware* Ginga, que estende as funcionalidades para o

[2] Implementados através das bibliotecas Platinum UPnP SDK [26]

suporte à execução de aplicações NCL que utilizem recursos de múltiplos dispositivos secundários filhos conectados a um dispositivo pai. O modelo de descrição de classes de dispositivos, baseado no UAProf, é detalhado, bem como as definições arquiteturais consideradas para o componente responsável. Uma API Lua, parte das definições LuaTV [19], também é apresentada. A realização dos cenários de aplicação como casos de testes para os requisitos inicialmente considerados, exemplificaram o uso das novas funcionalidades mais representativas incorporadas pelo novo componente.

Aplicações multimídia distribuídas são ainda pouco exploradas no contexto de TV Digital, apesar da pluralidade de plataformas que lhes dão suporte. O módulo proposto oferece funcionalidades que vão além das oferecidas por outras plataformas, e é flexível o suficiente para se adaptar ao surgimento de plataformas futuras, também em outros contextos de execução de aplicações multimídia diferentes de TV Digital. O trabalho possui grande potencial para evolução, e a alguns tópicos estão sendo levados em consideração para as pesquisas em andamento, como a definição de mecanismos para comunicação entre dispositivos de uma mesma classe, e funcionalidades relacionadas ao mecanismo de edição NCL (*NCLEdit*) [27].

8. REFERÊNCIAS

[1] MILLER B. A., et al. *Home Networking with Universal Plug and Play*. IEEE Communications Magazine, pp 105-109, 2001.

[2] *Universal Plug and Play Forum*. Disponível em http://www.upnp.org (Acesso em Abril de 2010).

[3] *Open Service Gateway initiative Alliance*. Disponível em http://www.osgi.org (Acesso em Abril de 2010).

[4] *Bluetooth Forum*. Disponível em http://www.bluetooth.com (Acesso em Abril de 2010).

[5] *Association of Radio Industries and Businesses*. ARIB STD-B23 Version 1.2: *Application Execution Engine Platform for Digital Broadcasting*. 2004.

[6] SOUZA FILHO G. L. de, LEITE L. E. C., BATISTA C. E. C. F. 2007. *Ginga-J: The Procedural Middleware for the Brazilian Digital. Journal of the Brazilian Computer Society*, v. 12, p. 47-56, 2007.

[7] SOARES L. F. G. et al, 2007. *Ginga-NCL: the Declarative Environment of the Brazilian Digital TV System. Journal of the Brazilian Computer Society*, v. 12, p. 37-46, 2007.

[8] ITU. ITU-T Recommendation H.761: *Nested Context Language (NCL) and Ginga-NCL for IPTV services*. (2009).

[9] SOARES L. F. G. et al, 2009. Ginga-NCL: Suporte a Múltiplos Dispositivos. *Brazilian Symposium on Multimedia and the Web – WebMedia*. 2009.

[10] SILVA L. D. N. e, et al. Suporte para desenvolvimento de aplicações multiusuário e multidispositivo para TV Digital com Ginga. T&C Amazônia Magazine. N. 12, ISSN 1678-3824, pp 75-84. Manaus, AM : s.n., 2007.

[11] DVB. 2003. *Digital Vídeo Broadcasting* (DVB). *Digital video broadcasting* (DVB) *Multimedia Home Platform* (MHP). Padrão ETSI TS 102 812, ETSI, 2003.

[12] LIN C.-L. et al.: *A Wrapper and Broker Model for Collaboration between a Set-Top Box and Home Service Gateway*. IEEE *Transactions on Consumer Electronics*, Vol. 54, No. 3, pp 1123-1128, 2008.

[13] HELAL S. *Standards for Service Discovery and Delivery. IEEE Pervasive Computing* archive. Volume 1 , Issue 3, pp 95-100 2002.

[14] *Apache River*. Disponível em http://incubator.apache.org/river/RIVER/ (Acesso em Abril de 2010).

[15] *WAP Forum, WAG UAProf. Technical Report* WAP-248-UAPROF-20011020-a, 2001.

[16] KLYNE G., REYNOLDS F., WOODROW C., OHTO H., HJELM J., BUTLER M. H., TRAN L. *Composite Capability/Preference Profiles (CC/PP): Structure and vocabularies. W3C working draft*, 2004.

[17] VITERBO J., CASANOVA M.A., RUBINSZTEJN H.K., ENDLER M. *An ontology based on the CC/PP framework to support content adaptation in context-aware systems*, *Proc. of WOMSDE 2006, in conjunction with SBES*, Florianópolis, pp. 1-10, ISBN 85-7669-086-1, 2006.

[18] IERUSALIMSCHY R. 2006. *Programming in Lua*. 2. ed. Rio de Janeiro. s.l. : Lua.org, p. 308, 2006.

[19] BRANDÃO R. R. de M., et al. *Extended features for the Ginga-NCL environment - Introducing the LuaTV API. Proceedings of the 19th International Conference on Computer Communication Networks (2nd Workshop on Multimedia Computing and Communications*), 2010 (*To Appear*).

[20] SANT'ANNA, F., et al. 2008. NCLua - Objetos Imperativos Lua na Linguagem Declarativa NCL. *Brazilian Symposium on Multimedia and the Web – WebMedia*, 2008.

[21] *Redland RDF Libraries*. Disponível em http://librdf.org/ (Acesso em Abril de 2010).

[22] KIM J., OH Y., LEE H., PAIK E., PARK K. *Implementation of the DLNA Proxy System for Sharing Home Media Contents*, IEEE *Transactions on Consumer Electronics*, Vol. 53, No. 1, 2007.

[23] *Ginga-NCL - Declarative DTV Middleware*. Disponível em http://www.gingancl.org.br/ (Acesso em Abril de 2010).

[24] *Andromote - An Android UPnP Remote Control*. Disponível em http://www.andromote.de/ (Acesso em Abril de 2010).

[25] *VideoLAN*. Disponível em http://www.videolan.org (Acesso em Abril de 2010).

[26] *Platinum UPnP*. Disponível em http://sourceforge.net/projects/platinum/ (Acesso em Abril de 2010).

[27] COSTA R.M.R., MORENO M.F., RODRIGUES R.F., SOARES L.F.G. 2006. *Live Editing of Hypermedia Documents. In Proceedings of ACM Symposium on Document Engineering* (Amsterdam, Holanda, 2006). DocEng 2006.