Um Ambiente de Simulação para Aplicações Multimídia Complexas

Manoel C. Marques Neto, Felipe Machado, Thiago Rodrigues
GSORT - IFBA
Rua Emídio Santos, s/n,
Salvador - Ba
manoelnetom@ifba.edu.br

Celso A. S. Santos
DI-CT-UFES
Av. Fernando Ferrari, 514,
Vitória - ES
saibel@inf.ufes.br

Raoni Kulesza
CI/UFPB
Cidade Universitária, Campus I, João Pessoa – PB
raoni@lavid.ufpb.br

ABSTRACT
This paper aims to present a simulation environment that can be used to run interactive applications developed in JavaDTV and NCL, focused on digital television. The proposed environment differs from others because it allows the simulation of both the digital TV environment sides: 1) the broadcaster which is responsible for transmission of both main and extra content, and 2) the receiver which is responsible for receiving all contents and executing applications to allow content presentation.

Categories and Subject Descriptors
H.5.4 [**Information Interfaces and Presentation**]: Hypertext/Hypermedia Architectures Navigation Theory Multimedia Information Systems

General Terms
Algorithms, Design, Languages

Keywords
TV Digital; Multimídia; Simuladores

1. INTRODUÇÃO
Os "programas de TV Digital Interativos" (TVDI) criam a possibilidade de veiculação de softwares interativos e personalizados junto com qualquer outro conteúdo da grade de uma emissor [1] [2]. Este cenário tem impactos diretos tanto na recepção de conteúdo quanto na forma como ele é produzido. O novo processo (que agora engloba softwares) é não linear, iterativo, ágil, multidisciplinar, heterogêneo e convergente. Essas características são semelhantes com aquelas encontradas em sistemas ubíquos [3] onde softwares diferentes podem rodar em plataformas diferentes e muitas vezes não convencionais (ex.: TV). Um dos pontos fundamentais para desenvolvimento desses sistemas é a existência de um ambiente que permita simular o seu funcionamento antes da sua implantação no ambiente real [4].

Os simuladores [4] de programas de TV digital interativos comumente utilizados em ambientes de desenvolvimento são o XletView [5], Emulador Ginga-J [6] e o Ginga-NCL Virtual Box [7]. Esses ambientes permitem executar aplicações interativas de TV digital a partir do uso de computadores simulando o receptor de TV. Por outro lado, eles não contemplam o ambiente onde ocorre a geração / transmissão de conteúdo: o *broadcaster*. A simulação do *broadcaster* é ponto fundamental para permitir a execução de uma categoria de aplicações interativas onde seu conteúdo tem três características importantes: (1) possuem relação com a semântica do conteúdo principal ; (2) são transmitidos via fluxos alternativos; e (3) são exibidos de forma sincronizada [8]. Neste artigo serão chamadas de aplicações multimídia complexas (AMC) as aplicações pertencentes a esta categoria.

O objetivo deste trabalho é apresentar uma infraestrutura de simulação completa (*broadcasters* e receptores) para permitir a execução de AMC para TV Digital nas suas duas modalidades de interatividade (aberta e conectada) [2]. Para isso o ambiente construído possibilita a simulação tanto da transmissão / recepção de conteúdos (oriundos de provedores diferentes) quanto a execução das aplicações interativas. É importante ressaltar que este ambiente não contempla a simulação da fase de produção de conteúdos.

Para demonstrar as principais funcionalidades do simulador foram desenvolvidas duas aplicações. Elas permitem apresentar durante a transmissão de uma corrida de F1 replays enviados pelo *broadcaster* e a previsão do tempo obtida em um provedor de conteúdos na Internet. Os vídeos com estas aplicações em execução podem ser vistos no YouTube[1].

2. ARQUITETURA E IMPLEMENTAÇÃO
A arquitetura do simulador está dividida em três módulos denominados *Broadcaster*, *Provider* e *Subscriber* conforme pode ser visto na Figura 1. O primeiro representa um emissor que é responsável pelo envio tanto conteúdo principal ("Gerador de Conteúdo Principal") como do conteúdo extra ("Gerador de Conteúdo Extra"), ambos em modo *broad-*

[1]http://youtu.be/lJ6ZPxgf-CI
http://youtu.be/VRg6149XesU

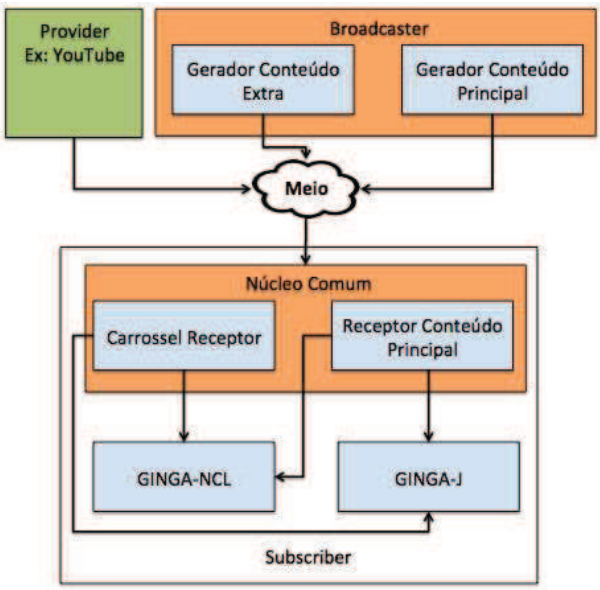

Figura 1: Arquitetura do simulador

cast. O segundo representa um provedor de conteúdos extras diferente do *Broadcaster* (ex.: portal de vídeos na Internet como o YouTube). O terceiro representa um receptor que permite sintonizar um canal (endereço de rede) e consumir o conteúdo enviado por um *Broadcaster* ou obtido (baixado) de um *Provider*. O *Subscriber* é ainda dividido em três sub-módulos: (i) núcleo comum responsável por processar o conteúdo enviado e disponibilizá-lo para os demais sub-módulos, (ii) simulador da máquina de apresentação Ginga-NCL que é responsável por executar aplicações NCL e (iii) simulador da máquina de execução Ginga-J que permite a execução de aplicações Java.

Para transmissão de conteúdo principal armazenado ou ao-vivo, o módulo *Broadcaster* usou a biblioteca *libvlc* incluída no projeto Vídeo LAN media player (VLC) e uma API denominada VLCj que permite manipular as funcionalidades de transmissão, recepção e codificação de mídias a partir de código escrito em Java [9]. Além do conteúdo principal, este módulo provê uma interface para o envio de conteúdo extra de qualquer tipo (vídeo, aplicação Java, NCL, etc.). Este envio ocorre de forma semelhante ao padrão DSM-CC que é conhecido como carrossel de dados [10]. O carrossel implementado controla o fluxo de envio dos diversos tipos de conteúdos extras. Para isso, cada conteúdo extra, chamado de objeto do carrossel, é armazenado em uma fila circular, segmentado em pacotes e transmitido usando intervalos de tempo de tamanho fixo. A implementação do carrossel utilizou bibliotecas Java para transmissão de dados via rede (ex.: *Socket*, *DatagramSocket*, etc.). A fim de manter a similaridade com o DSM-CC e permitir uma entrega confiável sem uso do TCP, foi definido e implementado um protocolo de verificação de integridade e de tipo de objetos. Neste protocolo os tipos de conteúdo foram identificados e isso permitiu diferenciar o tratamento dado a cada um deles. Por exemplo, objetos que representem classes Java devem ser armazenados e executados pelo receptor ou ainda, objetos que representem um vídeo ou uma imagem devem ser armazenados em diretórios no receptor e posteriormente consumidos por uma aplicação.

O núcleo comum do módulo *Subscriber* tem como objetivo receber todo o conteúdo transmitido (ou baixado de um *Provider*) e fazer a demultiplexação para disponibilizá-lo aos demais sub-módulos do *Subscriber*. Este módulo usa as mesmas bibliotecas do *Broadcaster* (*libvlc* e VLCj) e está dividido em duas partes. A primeira delas ("Receptor de Conteúdo Principal") é responsável pelo recebimento do conteúdo principal. Para isso ela usa uma instância da classe *Player* (que pertence a biblioteca VLCj), criada através do método *Manager.createRealizedPlayer(URL)*, onde a *URL* representa o endereço para o fluxo do conteúdo principal (ex.: udp://@:5555). Uma vez criada, essa instância é embutida em um contêiner Java e depois é apresentada ao usuário para simular a tela de uma TV real. O conteúdo extra é tratado através da parte do núcleo comum que implementa o recebimento do fluxo compartilhado chamada de "Carrossel Receptor". Ela tem a responsabilidade de separar, identificar e persistir em disco os pacotes recebidos. Caso um pacote já tenha sido persistido anteriormente, ele será ignorado. Após a conclusão de cada objeto (recebimento de todos os pacotes), o núcleo comum dispara um evento de notificação para os outros sub-módulos do *Subscriber* com quatro informações relacionadas ao objeto recebido: (i) seu identificador, (ii) sua localização física no disco para que os demais sub-módulos possam acessar o objeto e consumir o seu conteúdo, (iii) a versão do objeto e (iv) o tipo do objeto que pode ser: um diretório, arquivo ou aplicação.

Os outros dois módulos do *Subscriber* são as máquinas de apresentação Ginga-NCL e GINGA-J. A primeira estende o subsistema de mesmo nome e tem como objetivo permitir a apresentação de aplicações declarativas escritas em NCL/Lua. A segunda é uma extensão do Emulador Ginga-J e permite a execução de aplicações desenvolvidas em Java (*Xlets*). Esses módulos, são capazes de consumir conteúdos enviados pelo módulo *Broadcaster* e / ou baixados de um *Provider*.

3. PRINCIPAIS FUNCIONALIDADES

A Figura 2 ilustra a tela de controle do *Broadcaster* que permite: i) transmitir conteúdo principal e conteúdo extra; ii) iniciar, parar e reiniciar o carrossel de dados; iii) adicionar e remover objetos do carrossel e iv) atualizar a versão de um objeto. Para transmissão do conteúdo principal, o operador deve selecionar um arquivo de vídeo (ou uma URL) através do botão "Enviar conteúdo principal".

Para transmitir conteúdo extra, o operador deve inicialmente usar o botão "Adicionar Conteúdo" para selecionar e adicionar arquivos com os dados que serão enviados. Depois disso, o simulador exibe os nomes dos arquivos selecionados na tabela localizada na centro da tela. Em seguida, para dar início a transmissão, o operador deve clicar no botão "Iniciar Carrossel". O conteúdo é enviado de forma cíclica e cada objeto ocupa uma janela de tempo. Assim, cada vez que o tempo de uma janela se esgota o próximo objeto é transmitido e isso é repetido sucessivamente até o último objeto do carrossel. Uma vez que a última janela tenha se esgotado, o carrossel seleciona novamente o primeiro objeto e repete o ci-

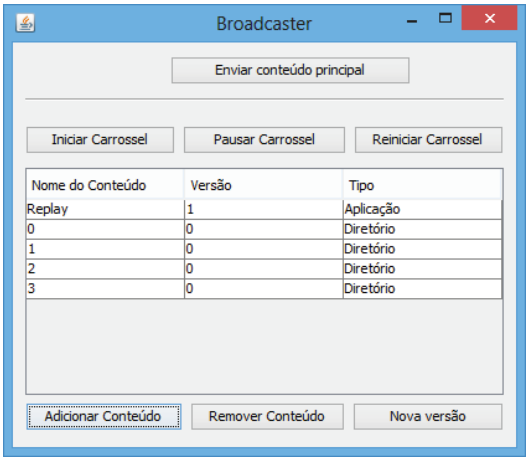

Figura 2: Tela do Broadcaster: envio de conteúdo principal e extra

clo de envio. O carrossel só para de enviar conteúdo quando o operador remove os objetos ou quando o operador pausa / para o carrossel manualmente (botões "Pausar e Reiniciar"). As operações de adição, remoção e substituição de objetos podem ser feitas em qualquer instante, inclusive durante a transmissão, sem a necessidade de parar a execução do carrossel. Isso é feito a partir dos botões "Adicionar Conteúdo", "Remover Conteúdo" e "Nova Versão".

Para recepção de conteúdos (principal e extra) é necessário que pelo menos um *Subscriber* esteja sintonizado com o *Broadcaster*. Para sintonizar um canal, o usuário deve executar uma das máquinas de apresentação disponíveis e definir a URL do serviço envio de conteúdos (endereço de broadcast) disponível na rede. A Figura 3 apresenta a tela que simula a sintonização de um canal de TV usando o módulo GINGA-J do *Subscriber*. Neste módulo, a configuração do serviço é feita no menu de configuração localizado no topo da tela. Assim que um canal é sintonizado, os conteúdos principal e extra começam a ser recebidos automaticamente. Os objetos são recebidos em pacotes e armazenados em uma área temporária do disco. Para cada objeto do carrossel, é criada uma pasta nomeada com o respectivo identificador, e nela são armazenados os pacotes identificados com um número de sequência e extensão pkt.

4. EXEMPLOS DE USO

Para demonstrar as principais funcionalidades do simulador foram desenvolvidos dois exemplos de uso. O primeiro deles permite apresentar, durante a transmissão de uma corrida de F1[2], replays enviados pelo *Broadcaster*. A Figura 3 exibe a tela desse exemplo de uso. Nesta aplicação, o conteúdo principal selecionado foi um vídeo de uma corrida previamente gravado (*F1.avi*). Os objetos com os conteúdos extras selecionados foram:

1. *Replay.class*: objeto do tipo aplicação (*Xlet*) cujo objetivo é apresentar um Menu com a lista dos replays

[2]http://youtu.be/lJ6ZPxgf-CI

Figura 3: Tela do Subscriber: recebimento de conteúdo principal e extra

disponíveis. Ele está localizado no canto superior esquerdo da tela e é atualizado em tempo de apresentação com itens selecionáveis. Cada item selecionado dispara a apresentação de uma reprise cujo conteúdo representa trechos importantes de uma corrida de F1 (largada, ultrapassagens, parada nos boxes, etc.);

2. *Pastas*: objetos do tipo diretório que contêm um vídeo (reprise) e um arquivo XML semelhante ao trecho de código a seguir:

```
<replays>
 <replay>
  <description>Largada</description>
  <sequence>4</sequence>
 </replay>
</replays>
```

Este arquivo é processado pela aplicação *Replay* e os seus dados fornecem o nome do replay (*tag description*) e a posição na qual ele aparecer no menu (*tag sequence*). Por exemplo, na Figura 3 a reprise da "Largada" é o quarto item do menu.

O segundo exemplo de uso permite apresentar durante a mesma transmissão de uma corrida de F1 do exemplo anterior uma previsão climática[3] (temperatura em duas localidades diferentes). Os dados desta previsão podem ser obtidos em um provedor de conteúdos na Internet (*Provider*) ou enviados pelo *Broadcaster*. A Figura 4 exibe a captura da tela deste exemplo de uso em execução. Nele os conteúdos extras utilizados foram:

1. *ClimaXlet.class*: objeto do tipo aplicação (*Xlet*) cujo objetivo é apresentar, no canto superior esquerdo da tela, uma tabela atualizável em tempo de apresentação que contém previsão de temperatura para duas cidades.

2. *clima.xml*: objeto do tipo arquivo que contém código escrito em XML semelhante ao trecho a seguir:

[3]http://youtu.be/VRg6149XesU

```
<climas>
 <clima>
  <local>Salvador</local>
  <temperatura>29</temperatura>
 </clima>
 <clima>
  <local>Rio de Janeiro</local>
  <temperatura>40</temperatura>
 </clima>
</climas>
```

A aplicação recebe duas instâncias de arquivos *XML*. A primeira é enviada pelo *Broadcaster* e o outra é baixada de um *Provider*. Ambas são processadas pela aplicação *ClimaXlet* e os seus dados fornecem a nome da cidade (*tag local*) e a sua respectiva temperatura (*tag temperatura*).

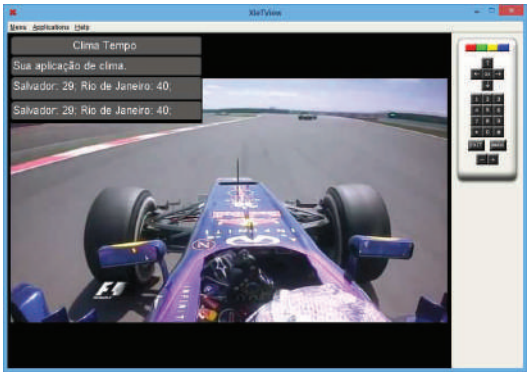

Figura 4: Tela do Subscriber: Aplicação de previsão do tempo

Na Figura 4, as linhas da tabela apresentam: i) o título "Clima Tempo"; ii) a mensagem "Sua aplicação de clima"; iii) a previsão, enviada pelo *Broadcaster*, para Salvador (29 graus) e Rio de Janeiro (40 graus); e iv) a mesma previsão obtida em um provedor de conteúdos.

5. CONCLUSÃO

Este artigo apresentou uma ferramenta que consiste em uma infraestrutura de simulação completa para TV Digital e TV Conectada. As principais contribuições do simulador são:

1. permitir a execução de AMC usando computadores conectados em rede;
2. aumentar fidelidade dos testes para esta categoria de aplicações; e
3. a sua arquitetura modular que permite estender os artefatos de software produzidos de forma a adaptar o simulador para outros contextos.

O simulador permite a execução das aplicações interativas de TVD mas não está limitado a esta plataforma. Este diferencial permite que outras aplicações interativas possam usar do simulador construído como, por exemplo, aquelas que são voltadas para IPTV uma vez que o simulador está estruturado em redes TCP/IP. Não foram avaliadas as restrições relacionadas com a rede de comunicação como a largura de banda ou a qualidade de vídeo para conteúdo principal. Este é um dos tópicos que será tratado nos próximos passos da evolução do sistema. A construção do simulador foi a primeira etapa de um projeto que visa apontar soluções para problemas apontados em [11] [8] [12] e que são relacionados às aplicações multimídia complexas.

6. REFERÊNCIAS

[1] L. E. C. Leite, G. L. de Souza Filho, S. R. de Lemos Meira, P. C. T. de Arújo, J. F. de A. Lima, and S. M. Filho, "A component model proposal for embedded systems and its use to add reconfiguration capabilities to the flextv middleware," in *WebMedia '06*. New York, NY, USA: ACM, 2006, pp. 203–212.

[2] S. Soursos and N. Doulamis, "Connected tv and beyond," in *Consumer Communications and Networking Conference (CCNC), 2012 IEEE*, jan. 2012, pp. 582 –586.

[3] W. I. Grosky, C. Zhang, and S.-C. Chen, "Intelligent and pervasive multimedia systems," *MultiMedia, IEEE*, vol. 16, no. 1, pp. 14 –15, jan.-march 2009.

[4] M. Laureano, *Maquinas Viruais e Emuladores Conceitos, Tecnicas e Aplicacoes*. Novatec, 2006.

[5] A. F. Vilas, R. P. Díaz Redondo, M. R. Cabrer, J. J. Pazos Arias, A. G. Solla, J. G. Duque, M. L. Nores, and Y. B. Fernández, "Mhp-osgi convergence: a new model for open residential gateways," *Softw. Pract. Exper.*, vol. 36, no. 13, pp. 1421–1442, Nov. 2006.

[6] R. Kulesza, C. Santos, T. Tavares, M. Neto, and G. de Souza Filho, "Desenvolvimento de aplicações imperativas para tv digital no middleware ginga com java."

[7] Telemídia. (2012, Jun.) Ginga-ncl virtual stb. [Online]. Available: http://www.ncl.org.br

[8] M. C. Maques Neto and C. A. Santos, "An approach based on events for treating the late tuning problem in interactive live tv shows," in *Proceedings of the 1st ACM international workshop on Events in multimedia*. ACM, 2009, pp. 41–48.

[9] D. Terra, N. Kumar, N. Lourenco, L. N. Alves, and R. L. Aguiar, "Design, development and performance analysis of dsss-based transceiver for vlc," in *EUROCON'11*, 2011, pp. 1–4.

[10] ISO/IEC, "ISO/IEC 13818-6:1998/Cor 1:1999," Padrão, 1999.

[11] M. C. Marques Neto and C. A. Santos, "An event-based model for interactive live tv shows," in *Proceedings of the 16th ACM international conference on Multimedia*, ser. MM '08. New York, NY, USA: ACM, 2008, pp. 845–848. [Online]. Available: http://doi.acm.org/10.1145/1459359.1459502

[12] M. C. Marques Neto and C. A. S. Santos, "Storytocode: a model based on components for specifying interactive digital tv convergent applications," in *Proceedings of the XV Brazilian Symposium on Multimedia and the Web*, ser. WebMedia '09. New York, NY, USA: ACM, 2009, pp. 8:1–8:8. [Online]. Available: http://doi.acm.org/10.1145/1858477.1858485