Projeto Arquitetural do Sistema *Interactive Digital TV Show*

Marcel Nascimento Ferreira. *e-mail*: marcel_fib@hotmail.com
Tiago Mesquita de Araujo Cunha. *e-mail*: tiagomac@gmail.com
Instituto Federal da Bahia - IFBA.
Grupo de Pesquisa em Sistemas Distribuídos, Otimização, Redes e Tempo Real.

***Resumo*—This article show a possible architectural project for a Digital TV system.**

***Index Terms*—Projeto Arquitetural, Sistema de Tv Digital, UML, Estilo Arquitetural, Modelo Arquitetural.**

I. Introdução

O OBJETIVO deste artigo é apresentar uma solução prática utilizando as técnicas de projeto arquitetural, demonstradas na disciplina de Aspectos Avançados em Engenharia de Software. Os métodos foram aplicados para poder estruturar um modelo de arquitetura para *softwares* que poderão ser desenvolvidos para Sistemas de Televisão Digital Interativos, nomeados de *IDTVS* (*Interactive Digital TV Show*).

Possuindo como principal característica a interação com o usuário, o *IDTVS* consiste na disponibilização de recursos multimídia que facilitam o acesso a informação e permitem à oferta de serviços e aplicações ao telespectador, além de uma maior rede de canais. Os sistemas para *IDTVS* apresentam características particulares que tornam o desenvolvimento de software para este nicho, uma atividade bastante complexa. Todo o planejamento do projeto, desde a arquitetura deve ser bem implementado e seguido de tal forma que não ocorram erosões arquiteturais e os desvios de padrão sejam controlados mantendo sua integridade.

Afim de propor um cenário que possibilite a disponibilização de serviços de fácil uso baseados no cenário atual de conteúdo multimídia, como *YouTube* e *Netflix* por exemplo, são apresentados três projetos:

- Arquitetural - demonstrando as visões estrutural, de implantação e de concorrência;
- Detalhado - demonstrando o detalhamento do principal módulo da visão arquitetural, com os eventuais *design patterns* e princípios de projeto Orientado a Objetos;
- Implementação e Implantação - demonstrando os artefatos de implementação e os aspectos complexos da implantação.

II. Interative Digital TV Show

O principal objetivo do sistema *IDTVS* é prover para o usuário uma *interface* para consumo de conteúdo multimídia e serviços disponibilizados através de fornecedores cadastrados. Através do sistema o usuário poderá escolher uma programação que deseja assistir, além de interagir com o fornecedor dessa programação através de um conteúdo dinâmico, fornecido pelo mesmo.

O sistema *IDTVS* deve ser executado em diversas plataformas clientes, por isso sua *interface* de comunicação deve prover recursos que tornem possível o desenvolvimento de clientes multiplataformas, com possibilidade de se adicionar *plugins*, permitindo a adição de novas funcionalidades sem grandes impactos.

Abaixo são apresentados os requisitos funcionais e não funcionais para o sistema *IDTVS* proposto, requisitos como autogerenciamento, alta escalabilidade, qualidade do serviço e confiabilidade não são tratados neste artigo por fazerem parte de um contexto que poderá ser apresentado em um artigo futuro:

A. Requisitos funcionais

1) Prover recursos para o usuário consumir arquivos multimídia.
2) Disponibilizar *interface* de gerenciamento de conteúdo gravado e ao vivo para *IDTVS Provider*.
3) Exibir programação de *IDTVS Provider*.
4) Associar dispositivo externo de captura a programação.
5) Gerenciar conteúdo dinâmico (*IDTVS Objects*) das programações.
6) Permitir o gerenciamento de *IDTVS Provider* no *IDTVS Broadcaster*.
7) Prover *interface* de monitoramento para o IDTVS Broadcaster.

B. Requisitos não funcionais

1) Para conteúdo gravado o *IDTVS Provider* precisa controlar conflito de horário com o próximo conteúdo agendado para exibição e/ou gravação.
2) O sistema deverá ter alta eficiência, com capacidade para 10k+ usuários simultâneos.
3) Escalabilidade horizontal.
4) O sistema deverá manter uma base de dados.
5) O sistema deverá atender ás normas legais de direito áudio visual.

6) O sistema deverá prover mecanismo para que seu funcionamento seja possível em qualquer plataforma, desde que atenda seus requisitos de execução.

Nas seções a seguir desse trabalho é apresentada a proposta de arquitetura para atender os requisitos supracitados.

III. Projeto Arquitetural

O projeto arquitetural foi dividido em três partes onde são apresentadas as arquiteturas do *IDTVS Subscriber*, *IDTVS Broadcaster* e o *IDTVS Providers* de forma separada. O *IDTVS Subscriber* deve prover suporte para execução em multiplataforma de conteúdos multimídia obtidos do *IDTVS Broadcaster* que estará disponível na sua rede local. O *IDTVS Broadcaster* trata as formas referentes a obtenção do conteúdo multimídia a partir de *IDTVS Providers* específicos. Já os *IDTVS Providers* devem publicar conteúdo de *IDTVS* na rede, sendo estes responsáveis pela programação dos canais, bem como os serviços e recursos interativos disponíveis. Nas seções seguintes são apresentados os detalhamentos de cada parte da arquitetura proposta.

A. IDTVS Subscriber

O componente *IDTVS Subscriber* tem o papel de consumidor de conteúdo disponibilizado pelo *IDTVS Broadcaster*. Para o pleno funcionamento o *IDTVS Subscriber* possui uma *interface* que permite a descoberta de serviço de *IDTVS Broadcaster* em sua rede local, na arquitetura apresentada essa *interface* é apresentada na forma do componente *SubscriberDiscovery* usando um estilo arquitetural *peer-to-peer*, permitindo se comunicar com um *IDTVS Broadcaster* da sua rede local apresentando para o usuário opções de escolha.
A apresentação dos resultados assim como opções de utilização do aplicativo são realizadas pelo componente *SubscriberDisplay*, que atua em conjunto com o componente *SubscriberService* para se comunicar com demais componentes do aplicatívo.
A execução do conteúdo multimídia passa por um processo de fragmentação através do componente *SubscriberMixer*. Nesse processo de fragmentação o áudio e o vídeo são separados e permitem a comutação de filtros, obtidos através do componente *FilterFactory*. Esses filtros podem adicionar efeitos ou características novas ao resultado final. O trabalho gerado após a comutação dos filtros é processado pelo componente *SubscriberSink* que após realizar o processo e unificação dos dados recebidos disponibiliza uma saída *stream* para captação desse conteúdo através do componente *SubscriberService*.
O componente *SubscriberService* atua através de uma especificação proprietária do *IDTVS Provider*, provendo funções e recursos utilizáveis pelo usuário. O *SubscriberService* se comunica diretamente com o *SubscriberInterpreter*, camada responsável pelo tratamento das informações com o sistema operacional utilizado na plataforma.
Na Figura 1 é apresentado o diagrama estrutural do *IDTVS Subscriber*. Abaixo são descritas as funcionalidades dos componentes envolvidos.

Operating System: Sistema Operacional da plataforma utilizada.
SubscriberInterpreter: Segue os princípios de um *basic interpreter*, sendo responsável pela tradução da camada superior para a camada inferior. O componente faz utilização de recursos do Sistema Operacional e os torna transparente para a camada superior.
SubscriberService: Componente centralizador, *facade* dos recursos apresentados pelo cliente. Deve seguir especificação fornecida.
SubscriberMixer: Responsável pelo tratamento dos dados multimídia recebidos e fornecimento de saída adequada para execução dos filtros.
SubscriberDisplay: Componente responsável pela *interface* da aplicação, passível de alteração de *layout*, conforme a plataforma utilizada.
SubscriberDiscovery: Componente que trabalha de acordo com estilo *peer-to-peer*. Responsável pela descoberta de *IDTVS Broadcasters* na rede local.
SubscriberSink: Conteúdo responsável pelo tratamento e união dos dados multimídia gerados após a utilização dos filtros.
AudioDecoder: Componente responsável pela decodificação do áudio recebido do *SubscriberMixer*.
VideoDecoder: Componente responsável pela decodificação do vídeo recebido do *SubscriberMixer*.
VideoFilter1: Filtro de tratamento de vídeo.
AudioFilter1: Filtro de tratamento de áudio.

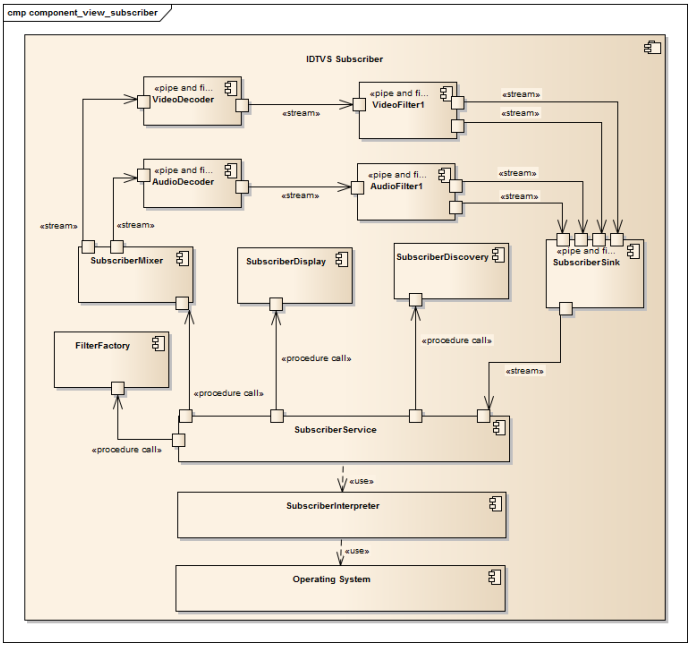

Figura 1. *Component View* do *IDTVS Subscriber*.

WebMonitorServer: Servidor HTTP para monitoramento das informações de programação dos *providers*.
WebMonitorDataBase: Base dados SQL para armazenar informações de configuração e monitoramento do *IDTVS Broadcaster*.

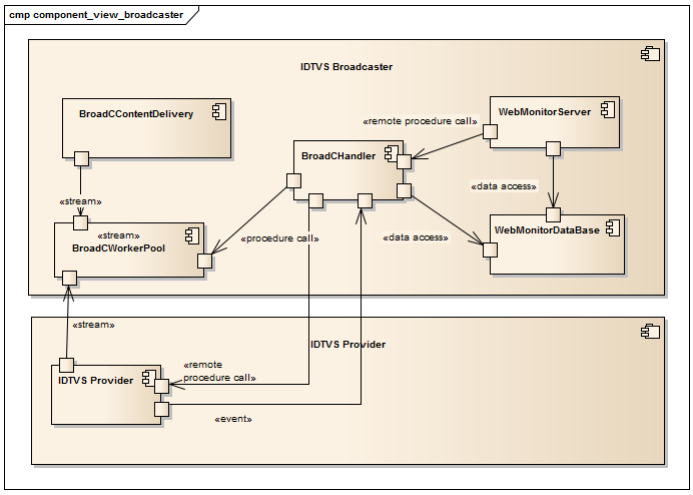

Figura 2. *Component View* do *IDTVS BroadCaster*.

B. IDTVS Broadcaster

O IDTVS Broadcaster é responsável por apresentar e repassar o conteúdo fornecido pelo *IDTVS Provider* para o *IDTVS Subscriber*. Para se comunicar com o *IDTVS Provider* o *IDTVS Broadcaster* utiliza através do componente *BroadCHandler* os conectores *remote procedure call* e *stream*. Para obter a grade de programação é utilizado o conector *remote procedure call*, que repassa o conteúdo multimídia para o *IDTVS Subscriber* através do componente *BroadCHandler* e dos conectores *stream BroadCWorkerPool* e *BroadCContentDelivery*. O *BroadCHandler* é o *facade* responsável pelas principais funções do *IDTVS Broadcaster*. O *IDTVS Broadcaster* conta com um servidor web para configurações internas de exibição para o usuário através do componente *WebMonitorServer*, que obtém as informações através do *WebMonitorDataBase*, atualizado pelo componente *BroadCHandler* na adição de um novo *IDTVS Provider* ou na atualização recebida através de um conector *event*.

Na Figura 2 é apresentado o diagrama estrutural do *IDTVS Broadcaster*. Abaixo são descritas as funcionalidades dos componentes envolvidos.

BroadCContentDelivery: Responsável por enviar o conteúdo obtido no *BroadCWorker*.
BroadCWorkerPool: *pool* de *workers* responsáveis por obter e armazenar em cache o conteúdo áudio visual em *live streaming* dos *providers* dos dados multimídia.
BroadCHandler: Tem como função: Notificar alterações ao *BroadCContentDisplay*; Atualizar banco de dados *WebMonitorDataBase*; Utiliza um *BroadCWorker* obtido do *pool* para se comunicar com o *IDTVS Provider* e obter o conteúdo multimídia que será encaminhado para o *IDTVS Subscriber*.

C. IDTVS Provider

O operador *IDTVS Provider* carrega o arquivo multimídia com sua melhor qualidade de áudio e vídeo disponível, juntamente com esse arquivo multimídia o operador cria um arquivo de configuração com a programação da grade e requisitos de exibição. Ao carregar o arquivo o componente *ProviderContentManager* fica responsabilizado em automaticamente criar versões do mesmo arquivo, com diferente tamanho e resolução, para atender a diferentes dispositivos e plataformas. O *ProviderContentManager* também realiza a checagem da programação por conflitos que impeçam a programação. Após a geração dos arquivos multimídia para diferentes dispositivos o *ProviderContentManager* armazena os dados gerados em uma CDN através do componente *ProviderCDN*. O componente *ProviderDriverNegotiator* é utilizado quando o operador deseja carregar o conteúdo de uma *webcam*, gravador, filmadora ou outro dispositivo de captura de audio e vídeo.

Na Figura 3 é apresentado o diagrama estrutural do *IDTVS Provider*. Abaixo são descritas as funcionalidades dos componentes envolvidos.

ProviderContentManager: Responsável pela geração de diferentes arquivos multimídia para diferentes plataformas.
ProviderCDN: Componente responsável pela comunicação e distribuição dos dados ao longo das *content delivery networks* disponíveis.
ProviderHandler: Responsável por enviar o conteúdo obtido no *BroadCWorker*.
ProviderWebManager: Componente responsável por disponibilizar recursos para o operador configurar o *IDTVS*

Provider.

ProviderWorkerPool: Responsável pelo armazenamento dos processos de tratamento do envio de conteúdo multimídia para o *IDTVS Broadcaster*.

ProviderDataBase: Componente responsável pelo armazenamento das configurações e dados de configuração do *IDTVS Object*.

ProviderDriverNegotiator: Componente responsável pelo tratamento de conteúdo provido de dispositivos periféricos como webcam, filmadoras, gravadores etc.

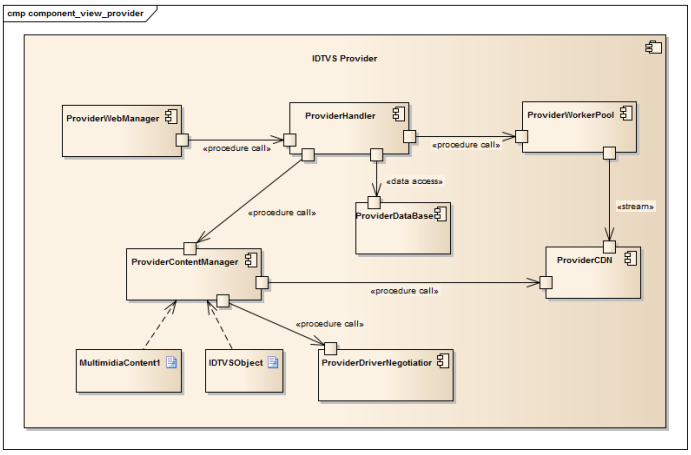

Figura 3. *Component View* do *IDTVS Provider*.

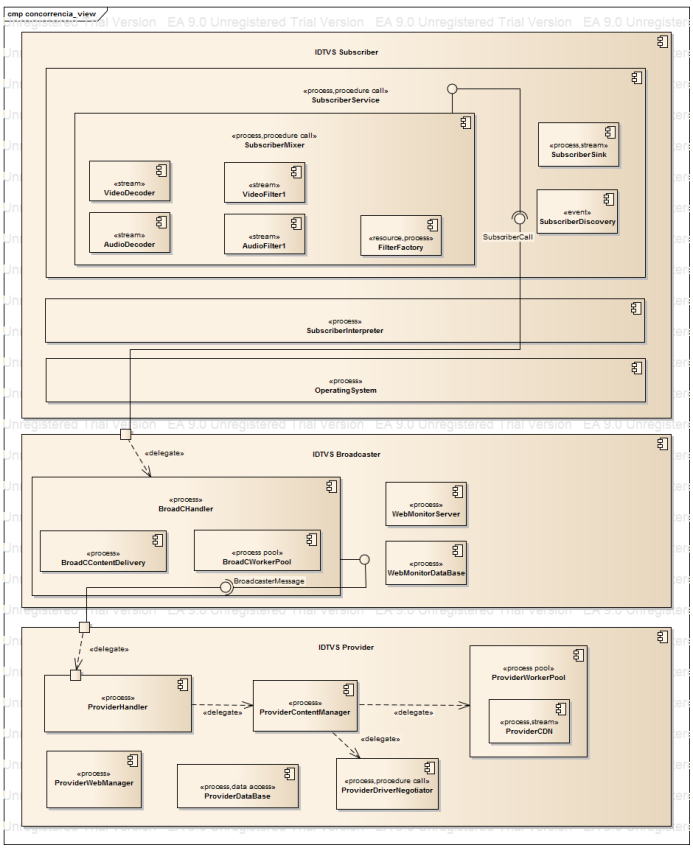

Figura 4. Visão de concorrência.

Na Figura 4 é apresentado o diagrama de concorrência do projeto arquitetural. Nesse diagrama os componentes e conectores são tratados como objetos ativos representando um modelo do projeto em um momento de sua execução.

IV. Projeto Detalhado

Nessa seção será detalhado o processo de atualização do *IDTVS Broadcaster* pelo *IDTVS Provider* através de *IDTVS Objects*. Os componentes *ProviderHandler* e *BroadCHandler* seguem o mesmo padrão e se comunicam através de conectores *remote procedure call* e *events*. No processo de atualização do *IDTVS Provider* os *IDTVS Broadcaster* cadastrados são notificados através de um conector *event*. Na proposta arquitetural foi escolhido o componente *ActiveMQ* por sua capacidade de integrar *events* em um ambiente Java e ser funcional no *container* Tomcat.

O processo de notificação de uma atualização no *IDTVS Provider* é iniciado através de uma atualização recebido no *IDTVS Provider* através do componente *ProviderWebManager*. Nesse momento é necessário que um operador realize a atualização de um conteúdo multimídia específico de um programa ou da grade de programação. No instante em que o *IDTVS Provider* é notificado dessa atualização o componente *ProviderHandler* inicia um novo *EventContext* através da chamada *lookup()*. Em seguida é instanciada uma nova conexão através do *ConnectionFactory*, responsável por gerar uma nova sessão. Um *Destination* é então gerado com base na sessão criada e um *MessageProducer* é encarregado de realizar o envio do objeto *IDTVS Object* que carrega as informações dinâmicas do conteúdo multimídia.

O recebimento de uma nova atualização de conteúdo pelo componente *BroadCHandler* ocorre através de um *TopicConnection* obtido em um *ConnectionFactory*. Através do *TopicConnection* um *MessageConsumer* dá origem a uma sessão que gera um *Message* contendo um objeto *IDTVS Object* da atualização realizada no *IDTVS Provider*.

Após o recebimento de um *IDTVS Object* no *IDTVS Broadcaster* o componente *BroadCHandler* atualiza o componente *WebMonitorDataBase* com a nova informação para disponibilizar ao *IDTVS Subscriber*.

V. Projeto de Implantação e Implementação

A proposta de projeto arquitetural apresentada divide a execução de seus componentes principais em diferentes dispositivos. O componente *IDTVS Subscriber* deve ser portável entre plataformas e sistemas operacionais distintos, mas deve atender para sua correta execução os requisitos mínimos de processamento de 1600Mhz e memória RAM equivalente ou superior a 1024mb.

O componente *IDTVS Broadcaster* deve ser executado preferencialmente em dois *devices* distintos. O *device* principal será responsável pelo gerenciamento do tráfego e disponibilização de uma *interface* de acesso às funções principais do sistema para serem utilizadas pelo *IDTVS Subscriber*. Um segundo *device* deverá ser utilizado para a execução do servidor web responsável por exibir e monitorar o tráfego do *IDTVS Broadcaster*. Ainda no segundo *device* deverá ser mantido o componente *WebMonitorDataBase* com as informações de configuração persistidas no sistema. O primeiro *device* deve dispor de um mínimo de 2200Mhz em capacidade de processamento e 1024mb de memória RAM. O segundo *device*, responsável pelo componente *web* do *IDTVS Broadcaster* deve dispor de um mínimo de 1800Mhz de processamento, 512mb de memória RAM e um espaço de armazenamento disponível de 512mb. Ambos os componentes devem executar em plataforma Linux ou Windows.
O componente *IDTVS Provider* responsável pelo fornecimento de conteúdo multimídia para o *IDTVS Broadcaster* deve dispor de dois *devices*, sendo o primeiro responsável pelos componentes principais do sistema e o segundo responsável pelos processos de distribuição do conteúdo multimídia fornecido. O primeiro *device* do *IDTVS Provider* deve dispor de um mínimo de 2600Mhz em capacidade de processamento e uma memória RAM mínima disponível de 2048mb. O segundo *device* deve dispor de um mínimo de 1800Mhz em processamento, 512mb de memória RAM e um espaço de armazenamento disponível de 512mb. Ambos os componentes devem ser executados em plataforma Linux.

Soluções COTS (*Commercial Off-The-Shelf*), que atendem aos requisitos do projeto foram adotadas para minimizar o esforço e possibilitar que o sistema mantenha um comportamento equivalente em diferentes ambientes de execução. A proposta em solução COTS adotada para o componente *IDTVS Subscriber* foi a utilização da plataforma Java. Dessa forma é possível utilizar a JVM como componente *SubscriberInterpreter* para comunicação com os recursos do sistema operacional. A escolha pela plataforma Java é justificada em sua capacidade de executar em diferentes diferentes *devices* com sistemas operacionais e arquitetura distintas.

O componente *SubscriberService* deve fazer uso extensivo da API JMF (*Java Media Framework*) que dispõe de recursos para comunicação com o sistema operacional que utiliza os componentes de *hardware* do dispositivo local para reprodução de conteúdo multimídia de modo transparente para o desenvolvedor e o usuário.
Devido a sua necessidade de escalabilidade e alto desempenho para entregar os dados fornecidos pelo(os) *IDTVS Provider(ers)* cadastrado(os) o componente *IDTVS Broadcaster* faz uso da linguagem de programação C++ nos componentes: *BroadCHandler*; *BroadCContentDelivery*; *BroadCWorkerPool*. O componente *WebMonitorServer* utiliza a plataforma J2EE com as seguintes soluções COTS: Tomcat 7; Hibernate 5; JSF 2.2. As soluções propostas atendem as necessidades de consulta de informações relacionadas ao *IDTVS Broadcaster* através da consulta direta ao componente *BroadCHandler*, onde podem ser obtidos os valores de utilização do serviço no momento. Para armazenar dados de configuração o componente *WebMonitorDataBase* faz uso da solução de armazenamento H2 devido a sua característica: Portabilidade; Suporte a *driver* ODBC; Escrito puramente em Java. Desse modo a solução proposta para o servidor Web do *IDTVS Broadcaster* se torna portável entre plataformas distintas.
O componente *IDTVS Provider* faz uso da plataforma Java em todos os seus componentes. Os componentes *ProviderWorkerPool* e *ProviderContentManager* fazem uso da especificação JNI para comunicação direta com algoritmos de processamento paralelo escritos em C++ para otimizar o desempenho de seus processos.
A Figura 5 abaixo apresenta o diagrama de implementação.

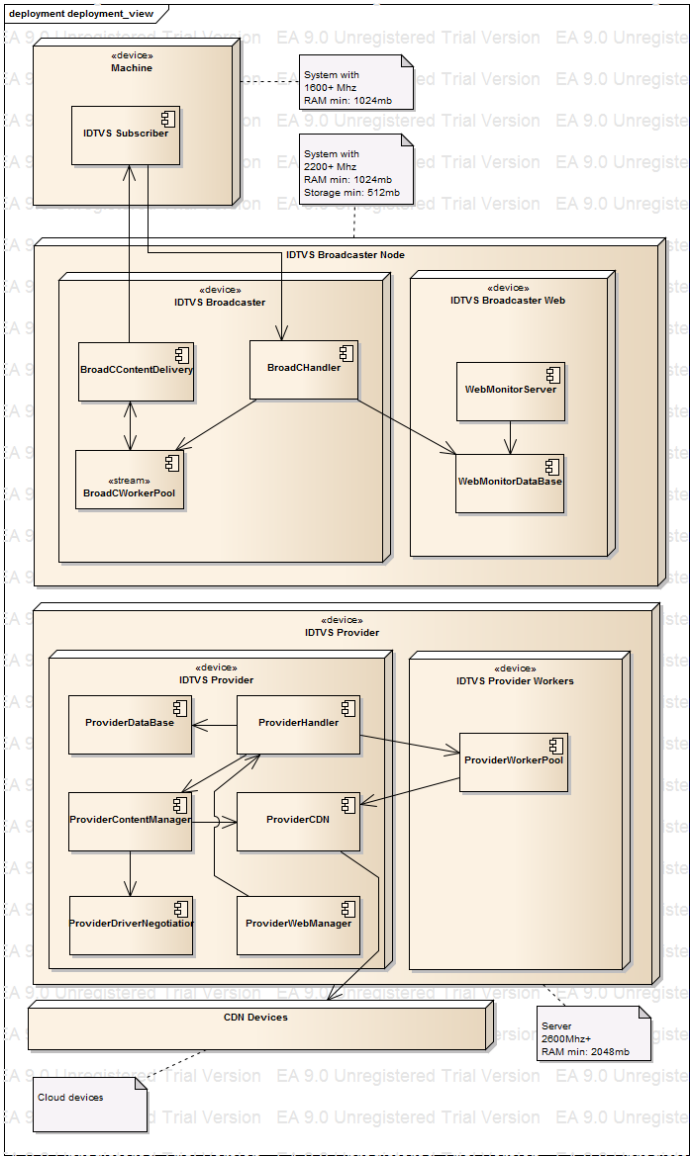

Figura 5. *Deployment view* do Sistema *IDTVS*.

VI. Discussão e Conclusões

Após a devida apresentação do principal objetivo deste artigo, nos capítulos seguintes foram apresentados aspectos técnicos da solução bem como algumas das possíveis visões arquiteturais para serem utilizadas no desenvolvimento de sistemas para *IDTVS*. Como discussões futuras se vê necessário avaliar questões como alta disponibilidade e auto-gerenciamento e adicionar novas *features* como mecanismos de controle de interrupção de sinal, interações offline, que poderão vir a trazer novas inclusões a arquitetura.

Alguns aspectos apresentam possibilidade de discussões sobre o âmbito de escalabilidade e performance, como por exemplo o componente *Subscribe Discovery* que pode atuar num estilo *Peer-to-Peer* hibrido, com um servidor controlando os possíveis *IDTVS Broadcaster* da rede local.

As propostas apresentadas neste artigo podem ser utilizadas como base para construção de um *software* para televisão digital interativa mas é necessário novas discussões, novas revisões com os interessados na arquitetura.

Referências

[1] Manoel Carvalho Marques Neto, Contribuição para modelagem de Aplicações Multimídia para TV Digital Interativa, Universidade Federal da Bahia, Salvador, 2011.

[2] Antonio Mendes da Silva Filho, Atributos em Arquitetura de Software, Revista Engenharia de Software, ed 22, 2009.

[3] Simone Diniz Junqueira Barbosa e Luiz Fernando Gomes Soares, TV digital interativa no Brasil sem faz com Ginga: Fundamentos, Padrões, Autoria Declarativa e Usabilidade. Em T. Kowaltowski e K.Breitman (orgs.) Atualizações em Informática 2008. Rio de Janeiro, RJ: Editora PUC-Rio, 2008. pp. 105-174.

[4] Hira, Celio, Arquimedia: uma proposta de arquitetura de software para terminais de acesso à TV digital interativa, ed. rev., São Paulo, SP: Dissertação de Mestrado - Escola Politécnica da Universidade de São Paulo, Departamento de Engenharia de Sistemas Eletrônicos. 2008.

[5] Jorge Fernandes, Guido Lemos e Gledson Elias, Introdução à Televisão Digital Interativa:Arquitetura, Protocolos, Padrões e Práticas, Apresentado na Jornada de Atualização em Informática do Congresso da Sociedade Brasileira de Computação, JAI-SBC, em Salvador – BA – Agosto de 2004. Anais do JAI—SBC, 2004.

[6] Roy Thomas Fielding, Architecture Styles and the Design of Network-Based Software Architectures, Dissertation Doctor of Philosophy in Information and Computer Science, University of California, Irvine, 2000.

[7] David Garlan e Mary Shaw: A Introduction to Software Architecture, School of Computer Science Carnegie Mellon University Pittsburgh, PA, 1994.