UNIVERSIDADE FEDERAL DA PARAÍBA
CENTRO DE CIÊNCIAS EXATAS E DA NATUREZA
DEPARTAMENTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

TVGrid
COMPUTAÇÃO EM GRID EM UMA REDE DE TV DIGITAL

Carlos Eduardo Coelho Freire Batista

João Pessoa – PB

Março de 2008

CARLOS EDUARDO COELHO FREIRE BATISTA

TVGrid
COMPUTAÇÃO EM GRID EM UMA REDE DE TV DIGITAL

DISSERTAÇÃO APRESENTADA AO CENTRO DE CIÊNCIAS EXATAS E DA NATUREZA DA UNIVERSIDADE FEDERAL DA PARAÍBA COMO REQUISITO PARCIAL PARA OBTENÇÃO DO TÍTULO DE MESTRE EM INFORMÁTICA (SISTEMAS DE COMPUTAÇÃO).

Orientador: Prof. Dr. Guido Lemos de Souza Filho
Co-Orientador: Francisco Vilar Brasileiro

BATISTA, Carlos Eduardo Coelho Freire

TVGrid: Computação em *grid* em uma rede de TV Digital. 2008.

66f.

Orientador: Guido Lemos de Souza Filho

Dissertação (mestrado) – Universidade Federal da Paraiba

Agradecimentos

Este espaço não comporta todos aos que devo agradecer por influenciarem positivamente o desenvolvimento desse trabalho. Os listados abaixo, porém, estão diretamente vinculados à viabilização deste trabalho.

Agradeço a:

Deus, pela realidade e todo o resto;

meus pais, por me trazerem para a realidade e todos os valores que me passaram;

minhas irmãs que tanto me ajudaram na minha educação e formação do meu caráter;

meu orientador e amigo, Guido Lemos, por me guiar na minha formação enquanto pesquisador e cientista, por me ensinar na prática que quem faz o que gosta vive de férias, e por toda sabedoria secular que me passou;

meu co-orientador, Francisco Brasileiro, pelo grande volume de conhecimento que me passou e por contribuições substanciais no trabalho que muito elevaram sua qualidade;

meus amigos, por serem meus amigos;

meus companheiros do LAViD, por serem meus amigos e por termos tido o melhor ambiente de trabalho possível;

aos professores do Departamento de Informática da Universidade Federal da Paraíba, por tudo que me passaram durante minha graduação e mestrado;

aos autores de trabalhos que serviram de base para o desenvolvimento deste;

a todos que utilizam o método científico em prol do desenvolvimento da humanidade, por me fazerem acreditar que a evolução é um processo contínuo do qual fazemos parte ativamente.

"*In science it often happens that scientists say, "You know that's a really good argument; my position is mistaken," and then they would actually change their minds and you never hear that old view from them again. They really do it. It doesn't happen as often as it should, because scientists are human and change is sometimes painful. But it happens every day. I cannot recall the last time something like that happened in politics or religion*."

Carl Sagan

Resumo

Plataformas de computação voluntária, tais como as utilizadas pelo projeto *SETI@home* comprovam que é possível se explorar volumes consideráveis de capacidade de processamento ociosa e banda de rede não utilizada de computadores conectados à Internet. Neste trabalho apresentamos a arquitetura do TVGrid, que explora essa idéia no contexto de uma rede de TV Digital.

Na arquitetura proposta, uma estação de TV é equipada com um escalonador de tarefas que utiliza a banda não utilizada do canal de TV Digital para transmissão de processos de uma aplicação paralela (tarefas) para os receptores, que poderão computá-las com seu poder de processamento ocioso. A saída do processamento realizado pelos receptores é posteriormente enviada para a estação de TV através de um canal de retorno – comumente uma conexão com a Internet.

A arquitetura apresentada contempla duas abordagens de escalonamento de tarefas, que são especificadas e analisadas de acordo com dados obtidos em testes representativos. Dessa forma o trabalho finalmente define um universo de aplicações para os quais a arquitetura do TVGrid é mais adequada.

Abstract

Voluntary computing platforms such as those deployed for the SETI@home project have proven that it is possible to harvest massive amounts of unused bandwidth and computing power available from computers connected to the Internet. In this work we present the TVGrid architecture to explore this idea in the context of a digital TV network.

In the proposed architecture, a TV station runs a scheduler that uses the unused bandwidth in the digital TV broadcast channel to send processes of a parallel (tasks) application to digital TV receivers, which may run them using their unused capacity. The output of the processing performed by the receivers are later sent back to the TV station through a return channel (a connection to the Internet).

The presented architecture comprises two approaches on task scheduling, which are specified and analyzed based on data collected on representative tests. The work thus aims to define a universe of applications for which TVGrid's architecture better fits.

Sumário

Índice de Figuras

Índice de Tabelas

Lista de Acrônimos e Siglas

ABNT – Associação Brasileira de Normas Técnicas
ACAP – *Advanced Common Application Platform*
AIT – *Application Information Table*
API – *Application programming interface*
ARIB – *Association of Radio Industries and Businesses*
ATSC – *Advanced Television Systems Committee*
BIOP – *Broadcast Inter-ORB Protocol*
BOINC – *Berkeley Open Infrastructure for Network Computing*
BoT – *Bag-of-Task*
CAT – *Conditional Access Table*
CORBA – *Common Object Request Broker Architecture*
CRC – *Cyclic redundancy check*
DAVIC – *Digital Audio Video Council*
DSM-CC – *Digital Storage Media Command and Control*
DVB – *Digital Vídeo Broadcasting*
ES – *Elementary Stream*
E/S – Entrada e Saída
EMM – *Entitlement Management Message*
GEM – *Globally Executable MHP*
HAVi – *Home Audio Video Interoperability*
HAN – *Home Area Network*
HDTV – *High Definition TV*
IEC – *International Electrotechnical Commission*
ISDB – *Integrated Services Digital Broadcasting*
ISO – *International Organization for Standardization*
ITU – *International Telecommunication Union*
ITU-T – *ITU Telecommunication Standardization Sector*
JCP – *Java Community Process*
JSR – *Java Specification Requests*
LGPL – *GNU Lesser General Public License*
MHP – *Multimedia Home Platform*
MPEG – *Moving Picture Experts Group*
MPEG2-TS – *MPEG-2 Transport Stream*
NIT – *Network Information Table*
OCAP – *OpenCable Application Platform*
PAT – *Program Association Table*
PCM – *Pulse-code modulation*
PES – *Packetised Elementary Stream*
PID – *Program Identification*
P2P – *Peer-to-peer*
PMT – *Program Map Table*
RPC – *Remote Procedure Call*
SBTVD – Sistema Brasileiro de TV Digital
SETI – *Search for Extra-Terrestrial Intelligence*
SI – *Service Information*
TS – *Transport Stream*
TVD – TV Digital
U-U – Usuário-para-Usuário
U-N – Usuário-para-Rede
WAN – *Wide area network*

1. Introdução

O cenário tecnológico mundial no qual este trabalho está contextualizado é de convergência entre tecnologias. O início do século XXI é marcado pelo surgimento de serviços e dispositivos que combinam tecnologias que surgiram em contextos distintos. Celulares com capacidade de captura de imagens e vídeo, provimento de serviços de telefonia, internet e televisão agregados, e vários outros produtos emergem no mercado. São vários os fatores que motivaram o surgimento dos mesmos, porém do ponto de vista técnico um fator se destaca: a digitalização das mídias.

Essa tendência teve como marco inicial o surgimento do CD (*Compact Disc*) no mercado em 1982, cuja finalidade original[1] era servir de armazenamento para música digital e substituiu os discos de vinil, popularmente conhecidos como LP – *Long playing* (seu mais popular formato) que armazenava música no formato analógico. A tendência chegou ao vídeo mais fortemente com o surgimento do DVD (*Digital Video Disc*), formato que já incorporava tecnologias de compressão de dados (o CD armazenava áudio no formato PCM – *Pulse-code Modulation*) [Maes, 2001][Blauert, 2005].

Com a popularização de serviços associados às tecnologias da informação, a digitalização da mais popular mídia de massa, a televisão, se deu através de sua sinergia com o computador. A TV Digital (TVD) oferece recursos que vão desde a melhora da qualidade da imagem à capacidade de interação com o conteúdo. O telespectador passa a ter um papel ativo, e os programas de televisão passarão a incorporar um *software*.

O receptor de TV Digital incorpora características de um computador – possui memória, processador e sistema operacional. Esse novo sistema de TV está desencadeando uma revolução tecnológica de enorme alcance, visto que uma importante fração da população mundial passa uma parte considerável do seu tempo assistindo televisão.

Na Europa a TV Digital aberta se encontra disponível para os telespectadores que possuem sistemas de recepção baseados nos padrões definidos pelo *Digital Video*

[1] Finalidade original visto que o *Compact Disc* pode armazenar qualquer tipo de mídia, possuindo também diferentes designações (CD-R e CD-RW por exemplo).

Broadcasting (DVB) [Freeman, 2004] – quatro milhões de receptores DVB foram vendidos na Itália entre 2005 e 2007[2]. A tendência é mundial e no Brasil desde 2005 oficialmente se desenvolve o padrão brasileiro de TV Digital aberta, através do projeto SBTVD (Sistema Brasileiro de TV Digital) [Leite, 2005].

Várias universidades e centros de pesquisa foram envolvidos na discussão do que deveria ser desenvolvido e o que deveria ser adaptado das normas internacionais já disponíveis. O LAVID (Laboratório de Aplicações de Vídeo Digital [3]), laboratório onde este trabalho foi desenvolvido, participou ativamente do projeto coordenando o desenvolvimento do padrão de *middleware* brasileiro, chamado de Ginga [Souza Filho, 2007]. As normas brasileiras de TV Digital foram publicadas pela ABNT (Associação Brasileira de Normas Técnicas[4]) e o sistema entrou oficialmente em operação no dia 2 de dezembro de 2007. Este trabalho é resultado indireto do desenvolvimento do *middleware* procedural brasileiro Ginga-J [Souza Filho, 2007].

1.1 Motivação

O uso de técnicas de modulação digital nos canais de TV (que possuem 6MHz de largura de banda no Brasil) torna possível a transmissão de dados com taxas na ordem de dezenas de megabits por segundo [Yamada, 2004] – no Brasil temos aproximadamente 20Mbps [Leite, 2005] e em sistemas de TV Digital como o japonês ISDB-S (sistema *Integrated Services Digital Broadcasting* para transmissão via satélite) utilizam canais com capacidade de transmissão de 52Mbps [Peng, 2002]. A transmissão de vídeos de alta definição (transmissão HDTV – *High Definition TV*), utilizando o padrão MPEG-2 de codificação (atualmente o mais utilizados pelos sistemas de TV Digital), demanda banda de transmissão de ordem entre 10 e 18 Mbps [Fox, 2002] e novos padrões de codificação como o padrão ITU (*International Telecommunication Union*) H.264 [Wiegand, 2003] demandam taxas ainda menores, na ordem de 8Mbps para vídeos de alta resolução (como na implementação *Quicktime*[5], da *Apple*®). As estações de TV Digital, então, muitas vezes dispõem de mais de 50% da sua banda de transmissão para multiplexação de dados com o fluxo de vídeo sendo transmitido. Essa "sobra" de banda pode ser utilizada para transmissão de informações adicionais relacionadas ao fluxo de vídeo transmitido (múltiplas legendas, opções de áudio) e

[2] http://www.dvb.org/about_dvb/dvb_worldwide/italy/index.xml
[3] http://www.lavid.ufpb.br
[4] http://www.abnt.org.br
[5] http://www.apple.com/quicktime/technologies/h264/

também aplicações para serem executadas (em um processador dedicado) no receptor de TV Digital – por conta desse aspecto técnico a TV Digital é dita interativa [Maior, 2002].

Porém, atualmente, nem todos os programas de TV utilizam as funcionalidades de interatividade oferecidas pela TV digital ou apenas utilizam-nas superficialmente [Maior, 2002], consumindo apenas uma pequena porção dos recursos disponíveis (banda de transmissão, capacidade de processamento). Por conta da natureza de alguns programas de TV, é bem provável que em muitos casos esses recursos nunca sejam utilizados em sua capacidade total. Esse fato motivou este trabalho, que propõe uma arquitetura para uma infra-estrutura de *grid* computacional, chamada de TVGrid, que é capaz de explorar os recursos não utilizados – particularmente a capacidade de processamento dos receptores – em uma rede de TV Digital.

Plataformas de "computação voluntária" tais como as desenvolvidas para o projeto SETI@home (*Search for Extra-Terrestrial Intelligence*) [Anderson, 2004] provaram que é possível se explorar quantidades substanciais de poder computacional ocioso e banda de rede não utilizada de usuários da Internet - a arquitetura do TVGrid, apresentada nesse trabalho, pretende adaptar essa idéia ao contexto de uma rede de TV Digital.

1.2 Objetivos

O objetivo geral do trabalho proposto é estabelecer uma arquitetura que permita a utilização de recursos ociosos de processamento dentro de uma rede de TV Digital. Para validar essa proposta foram definidos os seguintes objetivos específicos:

- Definição de uma arquitetura que modele um sistema de TV Digital como um tipo específico de *grid* computacional;
- Definição de um modelo que permita estimar a capacidade de processamento disponível em uma rede de TV Digital;
- Estudo de modelos de aplicações cuja execução seja possível na arquitetura proposta;
- Análise do modelo proposto com base em dados coletados durante a execução de testes representativos.

1.3 Estrutura da Dissertação

Esta dissertação está estruturada da seguinte maneira: seguindo esta introdução tem-se (no capítulo "2. Fundamentação Teórica") a base teórica relevante para este trabalho; adiante (no capítulo "3.") são apresentados alguns trabalhos cujas características são similares à desta proposta; no capítulo seguinte ("4. Arquitetura do TVGrid") a arquitetura do TVGrid é definida e analisada no capítulo "5. Análise de Desempenho da Arquitetura". Conclusões acerca do trabalho aqui apresentado são dispostas no capítulo "6. Conclusão".

2. Fundamentação Teórica

Este capítulo apresenta o contexto teórico utilizado para o desenvolvimento dos objetivos da presente dissertação. Foram definidos dois pilares teóricos fundamentais: as tecnologias de TV Digital (apresentada na Seção "2.1 TV Digital") e a teoria de computação em *grid* (Seção "2.2 Computação em *Grid*").

2.1 TV Digital

Um sistema básico de Televisão Digital (TVD) consiste de uma estação transmissora, um meio físico sobre o qual o sinal é transmitido, que pode ser o ar ou meios físicos guiados (cabo coaxial, fibra óptica etc.), e um Receptor responsável por receber o sinal transmitido, decodificá-lo e exibi-lo. É necessário que sejam estabelecidos padrões que normatizem todo o processo de captura, compressão, modulação e transmissão dos sinais de vídeo, além de todas as interfaces físicas entre os equipamentos envolvidos no processo, para garantir a compatibilidade entre os elementos envolvidos.

Como a transmissão é feita através de um fluxo de bits, há a possibilidade de se transmitir uma maior quantidade de informação multiplexada, em comparação ao sistema analógico[6]. Isso é possível principalmente graças ao desenvolvimento de técnicas de compressão, através das quais se podem produzir vídeos com taxas em bits de 1/4 a 1/10 do original puro (vídeo digital não comprimido) [Maior, 2002]. Graças a esta característica, os sistemas de TV Digital tendem a adotar padrões de codificação de vídeo que suportam resolução superior às disponíveis nos padrões de TV analógica, assim como padrões de codificação de áudio que suportam codificação de um maior número de canais de áudio.

A transmissão digital viabiliza também a transmissão de múltiplos fluxos de vídeo simultaneamente, permitindo a transmissão de mais de um programa de TV simultaneamente (multi-programação). As tecnologias de transporte permitem também que o fluxo carregue múltiplos formatos de vídeo, de forma que o conteúdo possa ser transmitido em diferentes resoluções, possibilitando a exibição em diferentes dispositivos.

[6] No formato analógico há transmissão de dados através da utilização do intervalo VBI (*Vertical blank interval*) para transmissão de *closed-caption* entre outras informações**Error! Reference source not found.**.

O mecanismo de transporte utilizado para transmissão de vídeo digital com dados agregados comumente utilizado nos sistemas de TV Digital é baseado no padrão MPEG-2 Sistemas [ISO/IEC, 2000]**Error! Reference source not found.**. Esse mecanismo define tabelas para inclusão de dados que podem ser informações acerca da programação transmitida, como também aplicações interativas. Alguns sistemas de TV Digital diferem justamente na forma da utilização destas tabelas.

Dentre os sistemas de TV Digital existentes, podem ser destacados [Morris, 2005][Souza Filho, 2007][Soares, 2007]:

- DVB (*Digital Vídeo Broadcasting*[7]) – sistema constituído por padrões mantidos pelo Projeto DVB (*DVB Project*), um consórcio composto por mais de 270 membros, e que são publicados por um comitê formado por ETSI (*European Telecommunications Standards Institute*[8]), CENELEC (*European Committee for Electrotechnical Standardization*[9]) e EBU (*European Broadcasting Union*[10]). Atualmente estima-se que mais de 170 milhões de receptores sejam compatíveis com o sistema DVB, que inclui definições para transmissão terrestre, via satélite, a cabo e por microondas.

- ATSC (*Advanced Television Systems Committee*[11]) – padrão desenvolvido pelo comitê homônimo americano, atualmente adotado por Estados Unidos, Canadá, México, Coréia do Sul, Argentina e Honduras.

- ISDB – (*Integrated Services Digital Broadcasting*) – conjunto de padrões desenvolvidos e mantidos pela organização japonesa ARIB (*Association of Radio Industries and Businesses*[12]), que incluem definições para rádio e TV Digital.

- SBTVD (Sistema Brasileiro de TV Digital) – sistema desenvolvido e mantido por um fórum homônimo criado pelo governo brasileiro, incorpora o padrão de transmissão terrestre japonês (ISDB-T) aliado à

[7] http://www.dvb.org
[8] http://www.etsi.org
[9] http://www.cenelec.org
[10] http://www.ebu.ch
[11] http://www.atsc.org
[12] http://www.arib.or.jp

tecnologia desenvolvida no Brasil – como uma API para desenvolvimento de aplicações que utilizem recursos de dispositivos (como celulares, PDAs, etc) em uma HAN (*Home Area Network*). Teve sua primeira transmissão oficial em 2 dezembro de 2007.

2.1.1 Uma rede de TV Digital

Entende-se uma rede de TV Digital como o conjunto de componentes necessários para transmissão e recepção de sinal de TV de acordo com um sistema de TV Digital [Yamada, 2004][Leite, 2005]. A Figura 1 ilustra uma rede de TV Digital representativa, cujos componentes serão descritos logo abaixo.

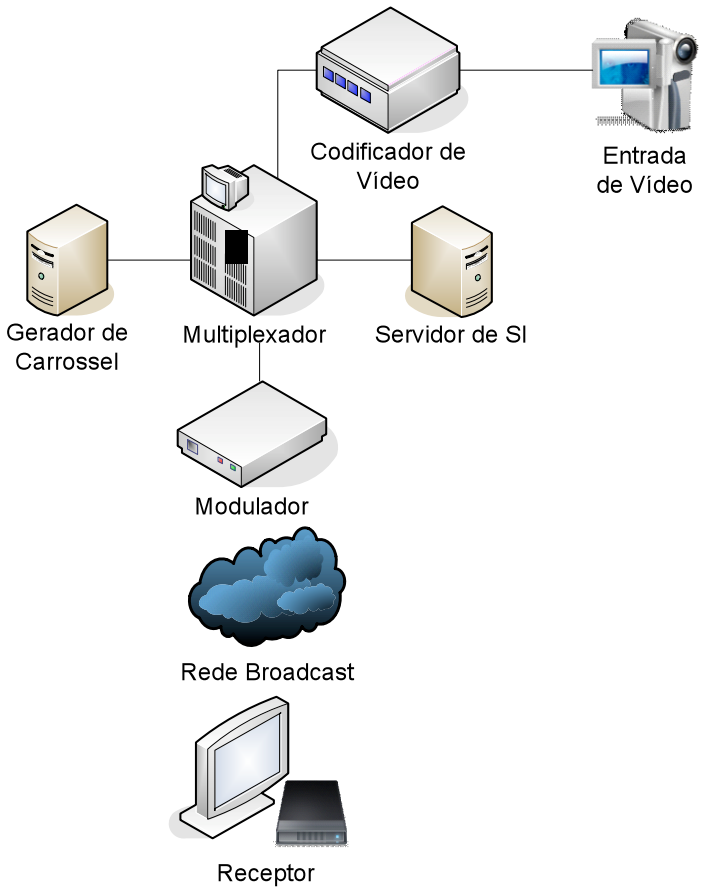

Figura 1. Componentes de uma rede de TV Digital

- *Codificador de Vídeo* – equipamento responsável por codificar uma entrada de vídeo em fluxo de vídeo digital, de acordo com um determinado padrão de codificação (MPEG 2 ou H.264, por exemplo).
- *Gerador de Carrossel* – módulo responsável por injetar no multiplexador um sistema de arquivos (comumente aplicações interativas e seus arquivos)

serializado, de maneira cíclica, produzindo o carrossel de dados. Para mais detalhes sobre o funcionamento deste componente, vide seção "2.1.4.2 DSM-CC".

- *Servidor de SI* (*Service Information*) – dispositivo responsável por gerenciar as tabelas que fornecem informações sobre o serviço oferecido em um canal de TV – tipicamente as informações acerca dos fluxos de áudio e vídeo transmitidos nesse canal.
- *Multiplexador* – equipamento responsável por unir (multiplexar) todos os fluxos que compõem o fluxo a ser transmitido. O formato adotado pela maioria dos sistemas é o MPEG-2 TS, que suporta multiplexação de múltiplas instâncias de vídeo, áudio e conjunto de dados (aplicações e seus arquivos).
- *Receptor* – dispositivo responsável por receber, interpretar e exibir o conteúdo do fluxo de transporte recebido das emissoras. Esse receptor, que na sua forma desacoplada é conhecido como *set-top box*, pode ser visto como um computador adaptado para as necessidades do ambiente televisivo, possuindo processador, memória, sistema operacional, etc (ver arquitetura de *software* na Figura 2). Nele é instalado um *middleware*, que é a instância de *software* responsável por abstrair características específicas de cada receptor, permitindo que uma mesma aplicação seja executada em *set-top boxes* de fabricantes e especificações diferentes [Souza Filho, 2007][Soares, 2007].

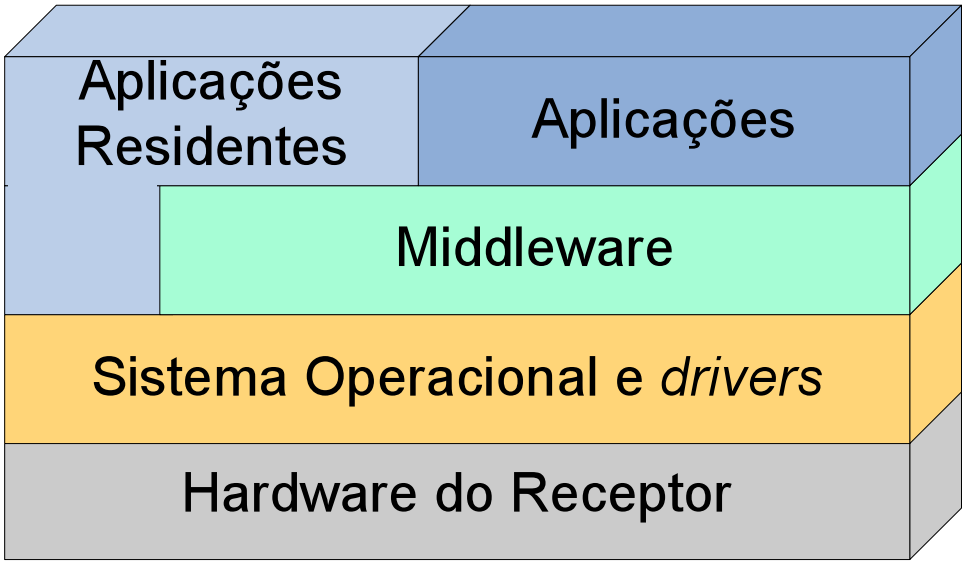

Figura 2. Arquitetura de *software* de um receptor de TV Digital

Uma rede de TV Digital pode incluir também um canal de retorno (rede de interação), de forma a permitir que os telespectadores possam enviar informações de volta à estação de TV. Uma representação visual de uma rede de TV Digital pode ser vista na Figura 3.

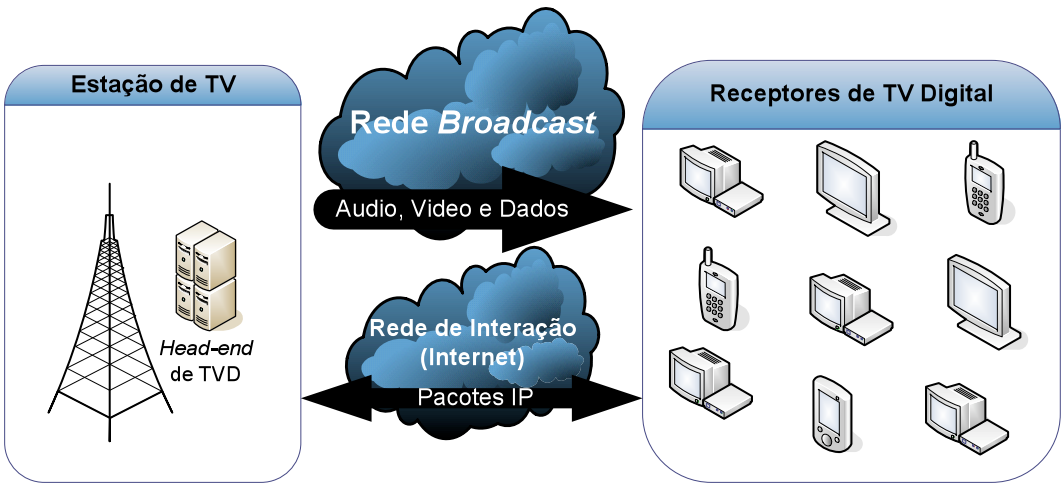

Figura 3. Visão de alto-nível de uma rede de TV Digital com canal de retorno.

2.1.2 *Middleware* do receptor de TV Digital

Atualmente os sistemas de Televisão Digital especificam uma camada de *software* chamada *middleware,* sobre a qual as aplicações interativas para TV podem ser executadas. *Middleware* é o neologismo criado para designar camadas de software que não constituem diretamente aplicações, mas que facilitam o uso de ambientes ricos em tecnologia da informação. A camada de *middleware* concentra serviços como identificação, autenticação, autorização, diretórios, certificados digitais e outras ferramentas para segurança [Leite, 2005].

No contexto de TV Digital, o *middleware* é o software que controla as principais facilidades disponíveis no receptor, tais como grade de programação, menus de opção e a possibilidade de execução de aplicações, fazendo com que a TV Digital possua caráter interativo. O *middleware* é um elemento capaz de fornecer uma abstração do sistema para as aplicações e os usuários, escondendo toda a complexidade dos mecanismos definidos por *hardware*, *software* e interfaces de comunicação do aparelho receptor do sinal de televisão digital. Dessa forma, a padronização de uma camada de *middleware* permite a construção de aplicações independentes do *hardware* e do sistema operacional, executáveis em qualquer plataforma de qualquer fabricante [Leite, 2005].

2.1.3 Compatibilidade entre Aplicações de Sistemas de TV Digital

A compatibilidade entre as aplicações baseadas nos diferentes padrões de TV Digital disponíveis, tais como SBTVD, DVB, ATSC e ISDB, se dá através das definições a cerca do *middleware* dos receptores. Eles utilizam a tecnologia Java[13] da *Sun* como parte da solução para a execução de aplicações nos seus receptores. A *Sun* disponibilizou a API (*Application programming interface*) Java TV para oferecer recursos específicos para o ambiente de televisão, e esta API está especificada na JSR 927[14] (*Java Specification Requests*) segundo os preceitos do JCP (*Java Community Process*). Além de Java e API relacionadas (de elementos gráficos, por exemplo), esses *middlewares* geralmente incluem suporte a linguagens declarativas (como XHTML) e de script. As aplicações Java executadas nos receptores são chamadas de *Xlets* [Morris, 2005].

Para tornar possível a harmonização das características dos *middlewares* dos diferentes sistemas tornando possível o desenvolvimento de aplicações Java (*Xlets*) que pudessem ser executadas em múltiplos sistemas foi elaborada uma norma desenvolvida com base no *middleware* DVB MHP (*Multimedia Home Platform*) denominada GEM (*Globally Executable MHP*) [DVB, 2007], que identifica e dá suporte a APIs independentes de sistema específico. Essa especificação é atualmente adotada pelos padrões de *middleware* procedural japonês (ARIB B.23) [ARIB, 2004], definições procedurais do *middleware* americano (ACAP – *Advanced Common Application Platform*) [ATSC, 2005], Brasileiro (Ginga) [Souza Filho, 2007] e, obviamente, o europeu (MHP) [DVB, 2003].

Seguindo a tendência de compatibilização, a ITU (*International Telecommunication Union*) publicou um conjunto de recomendações ITU-T: J.200 [ITU, 2001], J.201 [ITU, 2004] e J.202 [ITU, 2003]. Essas recomendações visam harmonizar os *middlewares* dos sistemas de TV Digital em diferentes níveis: arquitetura de *middleware* e suas definições para *middleware* declarativo e procedural, respectivamente [Morris, 2005].

[13] http://java.sun.com
[14] http://www.jcp.org/en/jsr/detail?id=927

2.1.3.1 Desenvolvimento de aplicações GEM

Aplicações procedurais de TV Digital, ou *Xlets*, devem se valer do modelo de aplicações proposto pela especificação Java TV da *Sun*. De forma bastante similar ao desenvolvimento de *applets* Java, a aplicação inicial deve implementar a interface apresentada abaixo [Morris, 2005]:

```
public interface Xlet {

public void initXlet(XletContext ctx) throws XletStateChangeException;

public void startXlet() throws XletStateChangeException;

public void pauseXlet();

public void destroyXlet(boolean unconditional) throws
XletStateChangeException;

}
```

Figura 4. Interface javax.xlet.Xlet [Morris, 2005]

Os métodos a serem implementados são utilizados para o controle do ciclo de vida das aplicações (Figura 5). Abaixo, cada método é detalhado:

- *initXlet* – invocado pelo *middleware* para inicializar o *Xlet*. O *middleware* passa à aplicação o contexto onde esta será executada, na forma de um objeto *XletContext*, que também é utilizado pela aplicação para sinalizar para o *middleware* que o *Xlet* mudou de estado. A inicialização do *Xlet* feita por esse método não deve utilizar nenhum recurso compartilhado. Quando há falha na chamada deste método, uma exceção *XletStateChangeException* é lançada.

- *startXlet* – utilizado para sinalizar que o *Xlet* está se tornando ativo, podendo utilizar os recursos necessários para realização de suas funções.

- *pauseXlet* – chamado quando há a necessidade que o *Xlet* tenha sua execução suspensa (para que outra aplicação fique ativa, por exemplo). Nesse método o *Xlet* deve liberar os recursos compartilhados que estiver utilizando para que possa ser considerado pausado (podendo ser ativado novamente pelo *middleware*).

- *destroyXlet* – chamada utilizada para liberar todos os recursos utilizados pelo *Xlet* e operações relacionadas à exclusão do *Xlet* (estado destruído). Se a exceção *XletStateChangeException* for lançada por esse método, o *Xlet* só será destruído se o atributo *unconditional* for definido como *true*.

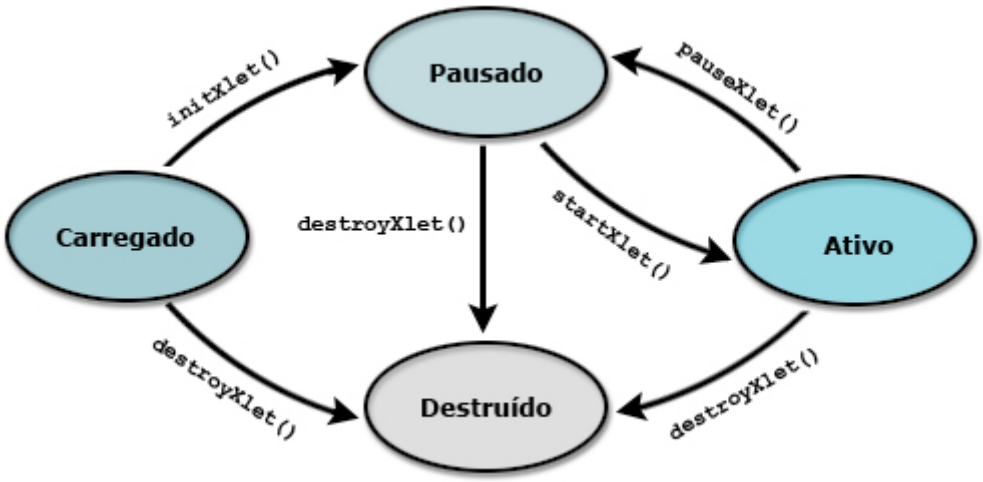

Figura 5. Ciclo de vida de um *Xlet* (aplicação Java TV) e os métodos utilizados para as transições de estado [Morris, 2005].

Para que uma aplicação *Xlet* esteja de acordo com as definições GEM ratificadas pelo padrão internacional ITU J.202 [ITU, 2003], os seguintes pacotes poderão ser utilizados durante o seu desenvolvimento:

- Pacotes JavaTV [Morris, 2005] – A API Java TV[15] é uma extensão para a plataforma Java para sustentar a produção de conteúdo interativo de forma procedural para a televisão digital, com base nas tecnologias Java. A principal finalidade da API Java TV é fornecer um conjunto de métodos, classes e interfaces para facilitar a construção de aplicativos destinados a serem executados através de plataformas de recepção de televisão digital independentes das tecnologias utilizadas na rede de transmissão.

- Pacotes DAVIC (*Digital Audio Video Council*) [DAVIC, 1998] – O sistema DAVIC é um conjunto de especificações que têm como principal objetivo sustentar uma interoperabilidade real de ponta a ponta para as plataformas

[15] http://java.sun.com/products/javatv/

envolvidas na execução de serviços de áudio e vídeo transmitidos via radiodifusão. Assim, esses padrões podem ser utilizados nos sistemas de televisão digital para fornecer conteúdo ao usuário final e também para permitir interatividade com o mesmo usuário. O sistema DAVIC tem suas próprias API, e portanto ele também pode ser considerado como um *middleware* de alto nível (apesar de ser comum para ele operar em conjunto com um *middleware* de nível mais baixo). DAVIC compreende os seguintes pacotes: org.davic.media, org.davic.resources, org.davic.mpeg, org.davic.mpeg.sections, org.davic.net, org.davic.net.dvb, org.davic.net.tuning

- Pacotes HAVi (*Home Audio Video Interoperability*) [HAVi, 2001] – As definições HAVi determinam uma rede doméstica padrão focada na interoperabilidade de dispositivos de áudio e vídeo, de forma que todos os dispositivos de áudio e vídeo componentes de uma rede HAVi possam interagir uns com os outros. O principal objetivo de uma interface de usuário em uma rede doméstica é oferecer um ambiente operacional fácil de usar. A arquitetura HAVi permite que os usuários controlem dispositivos de forma familiar, utilizando uma tela frontal ou através de um controle remoto. A interface de usuário HAVi Nível 2 API é um subconjunto Java AWT 1.1 com algumas extensões. Os seguintes pacotes são disponíveis: org.havi.ui, org.havi.ui.event.

- Pacotes DVB [Morris, 2005] – Ao desenvolver o *middleware* padrão MHP, o DVB incluiu alguns pacotes para estender as funcionalidades oferecidas por Java TV, HAVi e DAVIC. Essas funcionalidades incluíram API de Informações de Serviço, Intercomunicação entre *Xlets*, persistência etc. Os seguintes pacotes são incluídos nessa categoria: org.dvb.application, org.dvb.dsmcc, org.dvb.event, org.dvb.io.ixc, org.dvb.io.persistent, org.dvb.lang, org.dvb.media, org.dvb.net, org.dvb.net.tuning, org.dvb.net.rc, org.dvb.test, org.dvb.ui, org.dvb.user.

2.1.4 Sistemas de Transmissão de dados através de um canal de TV Digital

A seguir serão apresentadas as tecnologias de multiplexação utilizadas para transmissão de dados e aplicações interativas em sistemas de TV Digital.

2.1.4.1 MPEG-2 *Systems*

MPEG-2 é a designação dada a um conjunto de padrões e definições acerca de codificação e compressão para áudio e vídeo, publicado pela ISO/IEC (*International Organization for Standardization* / *International Electrotechnical Commission*) como o padrão internacional 13818 [ISO/IEC, 2000]. A especificação é dividida em 10 partes, e a primeira, MPEG-2 *Systems* [ISO/IEC, 2000], versa sobre a camada de sistema da codificação: sincronização de áudio e vídeo e multiplexação de áudio, vídeo e dados. Dentre os componentes definidos pelo processo de transmissão do MPEG-2, destacam-se o *Elementary Stream* (ES) e o *Packetised Elementary Stream* (PES).

O *Elementary Stream* (ES) pode ser encontrado em várias formas, descritos como: *Digital Control Data* (Dados de Controle Digital), *Digital Vídeo* (Vídeo Digital), *Digital Audio* (Áudio Digital) e *Digital Data* (Dados Digitais). No caso de áudio e vídeo, os dados podem ser organizados em unidades de acesso que representam por sua vez uma unidade fundamental de codificação.

O *Packetised Elementary Stream* (PES) é o resultado do empacotamento de um fluxo de bits elementar. Cada pacote PES tem um cabeçalho que contém tamanho do fluxo de bits, identificador do fluxo de bits, controle de criptografia, marcadores de tempo, etc. Os fluxos de bits PES para áudio, vídeo e possivelmente dados são multiplexados juntos em um único fluxo de bits de saída para transmissão.

MPEG-2 *Transport Stream* (MPEG2-TS) é uma forma simples de empacotar e multiplexar vídeo, áudio e fluxo de dados de aplicações para uma transmissão em uma determinada rede. Um pacote MPEG2-TS é a unidade elementar de dados em um fluxo TS, e é formado por 188 bytes, sendo divididos em quatro bytes de cabeçalho e mais 184 bytes de dados (*payload*).

O cabeçalho do pacote contém um campo de 13 bits que atua como identificador de programas (PID – *Program Identification*) responsável por separar o tipo de dados contido no *payload* do pacote sendo ele um vídeo, um áudio ou dados de uma aplicação, visto que cada um desses tipos possui um PID diferente. Ocorre ainda a utilização de pacotes com identificadores padronizados, como o caso de pacotes nulos, que utilizam o PID “0x1FFF”.

Os fluxos elementares são inseridos na entrada do modulador pelo multiplexador respeitando a ordem de chegada; não havendo pacotes de dados a ser inseridos, o multiplexador insere pacotes nulos com o objetivo de manter a taxa de transmissão constante. Para um usuário receber um determinado fluxo elementar multiplexado em um fluxo transporte (*Transport Stream* – TS), o mesmo deve selecionar um identificador e filtrar os pacotes caracterizados com o PID selecionado. Para isso, existem tabelas de sinalização que informam a descrição de cada programa transmitido. Estas tabelas são transmitidas junto com os fluxos elementares em pacotes MPEG-2 TS.

As tabelas de sinalização contêm as descrições dos fluxos elementares que precisam ser combinados para formar um programa, como também a descrição do programa transmitido. Cada tabela possui um conjunto de sessões de tamanho variável que são protegidas por um código de verificação de integridade CRC (*Cyclic redundancy check*).

Abaixo um conjunto representativo de tabelas de sinalização:

- PAT – *Program Association Table* (Tabela de Associação de Programas): contém um lista de PID das tabelas que descrevem cada programa. A PAT é enviada com um PID fixo e com valor "0x000".

- CAT – *Conditional Access Table* (Tabela de Acesso Condicional): define o tipo de embaralhamento (*scrambling*) usado e o valor do PID de cada fluxo (*stream*) de transporte que contém o gerenciador de acesso condicional e informações para o acesso ao fluxo protegido (EMM – *Entitlement Management Message*). A CAT é enviada com um PID conhecido e com valor "0x001".

- PMT – *Program Map Table* (Tabela de Mapa de Programas): define o conjunto de PID associados a um programa, como por exemplo PID de áudio e vídeo.

- NIT – *Network Information Table* (Tabela de Informações da Rede): contém detalhes sobre o canal físico utilizado para transmitir o fluxo MPEG incluindo a freqüência utilizada, por exemplo. A NIT possui um PID conhecido e de valor "0x010".

2.1.4.1.1 AIT – *Application Information Table*

Para sinalizar ao receptor que aplicações estão multiplexadas com o fluxo de vídeo, os *middlewares* aderentes ao GEM [DVB, 2007] definem uma tabela de serviço de informação (SI – *Service Information*) [Morris, 2005] denominada *Application Information Table* – AIT [Morris, 2005][DVB, 2007]. A AIT contém todas as informações que o receptor de TV Digital necessita para executar uma aplicação que recebe multiplexada. Estas informações podem ser: o nome da aplicação, o identificador da mesma, controles do ciclo de vida das aplicações (Figura 5, página 20), etc.

A identificação da aplicação consiste de duas partes: o identificador da organização, um valor global único identificando a organização responsável pela aplicação e o identificador da aplicação em si. O controle de ciclo de vida de aplicações é sinalizado através do campo *application_control_code* da AIT, permitido que seja sinalizado ao receptor de TV Digital o que deve ser feito com relação ao ciclo de vida da aplicação. Os valores possíveis para o *application_control_code* definidos pelo GEM são:

- ***autostart* (0x01)** – indicando que a aplicação deve ser iniciada assim que for carregada;
- ***present* (0x02)** – aplicação a ser iniciada pelo usuário, ficando disponível na lista de execução de aplicações.
- ***destroy* (0x03)** – quando o código de controle de uma aplicação é modificado de *autostart* ou *present* para *destroy* uma aplicação procedural, o *middleware* é invocado para que a aplicação seja finalizada (o método *destroyXlet*() será invocado com o valor *false*).
- ***kill* (0x04)** – quando o código de controle de uma aplicação é modificado de *autostart* ou *present* para *kill* uma aplicação procedural é então incondicionalmente finalizada (o método *destroyXlet*() será invocado com o valor *true*).
- ***prefetch* (0x05)** – código de controle utilizado para aplicações declarativas (DVB-HTML ou OCAP-HTML [CableLabs, 2005] por exemplo). Os elementos utilizados por aplicações com esse código de

controle são previamente carregados e disponibilizados para a máquina de execução declarativa.

- ***remote* (0x06)** – A aplicação não está disponível no canal corrente, porém em um outro canal que, quando selecionado, iniciará a execução da aplicação.

2.1.4.2 DSM-CC

O *Digital Storage Media Command and Control* (DSM-CC) é um conjunto de protocolos que fornecem operações e funções de controle para gerenciamento de *bitstreams* especificado nos padrões ISO/IEC 11172 (MPEG-1) e ISO/IEC 13818 (MPEG-2) [ISO/IEC, 1998]. O DSM-CC também pode ser definido como um modelo de referência funcional no qual as entidades Cliente e Servidor (também denotadas de Usuários) usam uma rede para se comunicar [ISO/IEC, 1999] [ETSI, 2003].

O DSM-CC pode ser dividido em duas categorias funcionais: Usuário-para-Usuário (U-U) e Usuário-para-Rede (U-N) [ISO/IEC, 1998].

No DSM-CC U-U, o cliente pode acessar remotamente objetos no servidor através de chamadas de procedimento remoto (*Remote Procedure Call* – RPC). Isso permite a execução de uma larga faixa de aplicações multimídia, como vídeo sob demanda, jogos e ensino a distância, sobre um sistema de distribuição MPEG [ETSI, 2003].

O DSM-CC U-N é usado para estabelecer uma sessão entre um Cliente e um Servidor e alocar os recursos de rede necessários para suportar o nível de interação do serviço associado entre os usuários finais [ISO/IEC, 1998].

2.1.4.2.1 Carrossel de dados

Carrossel de dados é o mecanismo de transporte DSM-CC U-N [ETSI, 2003], que permite a um servidor transmitir um grande volume de dados para um conjunto de receptores, repetindo ciclicamente a transmissão do seu conteúdo. Os dados desse carrossel são logicamente divididos em módulos que, por sua vez, são divididos em blocos de mesmo tamanho, com exceção do último, que pode possuir um tamanho menor. A divisão dos módulos em blocos torna mais fácil a transmissão por difusão [ETSI, 2003]**Error! Reference source not found.**. Se uma aplicação no receptor deseja

acessar um módulo específico do carrossel de dados, a mesma deve esperar pela próxima retransmissão dos dados. A Figura 6 apresenta a transmissão cíclica de um carrossel com três módulos (M2, M3 e M8).

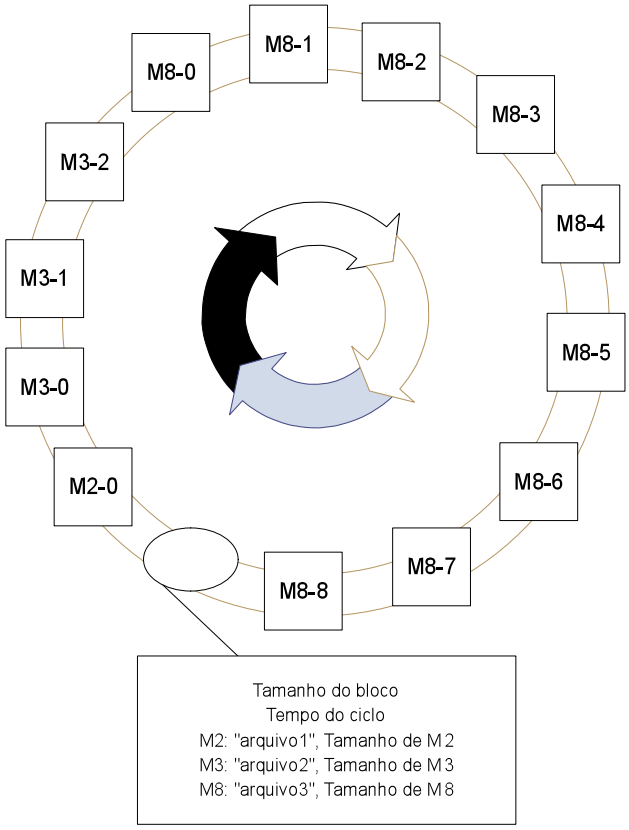

Figura 6. Transmissão cíclica de módulos no carrossel de dados [ETSI, 2003]

Na Figura 6, cada bloco do carrossel de dados é representado por um retângulo com notação MX-Y, onde Y é o índice do bloco correspondente ao módulo de índice X. M3-1, por exemplo, equivale ao bloco 1 do módulo 3. A quantidade de blocos dentro de um módulo depende do tamanho do módulo e do tamanho máximo do *payload* de cada bloco [ETSI, 2003]. É possível atualizar dinamicamente o carrossel que está sendo transmitido, adicionando, removendo ou alterando seus arquivos, bastando para isso criar uma nova versão para os módulos que contêm os arquivos a serem atualizados.

Analisando a Figura 6 também se observa que cada módulo foi inserido no carrossel de dados apenas uma vez e os blocos de um determinado módulo foram inseridos em seqüência. No entanto, não existe nenhuma restrição sobre o número de vezes e a posição em que os módulos são inseridos no carrossel de dados. Isso permite

criar um carrossel de dados que melhor se adapte a uma aplicação específica [ETSI, 2003].

O padrão internacional ISO/IEC 13818-6 [ISO/IEC, 1998] define que cada bloco de um determinado módulo é representado por uma mensagem DSM-CC *DownloadDataBlock* e as informações de controle sobre os mesmos, como quantidade e tamanho dos blocos, são definidas pelas mensagens DSM-CC *DownloadInfoIndication* e *DownloadServerInitiate* [ISO/IEC, 1998].

2.1.4.2.2 Carrossel de objetos

Um carrossel de objetos, pertencente à classe DSM-CC U-U, permite a transmissão de um grupo estruturado de objetos dentro de um carrossel de dados DSM-CC, uma vez que, os dados transmitidos neste último não possuem nenhum significado específico. Para transmitir esses dados de forma estruturada, um carrossel de objetos é transmitido usando o protocolo BIOP (*Broadcast Inter-ORB Protocol*), definido pelo modelo CORBA (*Common Object Request Broker Architecture*) [ETSI, 2003].

O carrossel de objetos utiliza objetos do tipo arquivo, diretório e fluxo (*stream*) permitindo que um carrossel possa transmitir um sistema de arquivos (Figura 7). O sistema de arquivos transmitido deve ser carregado pelo receptor de acordo com a implementação do *middleware*, visto que o mesmo poderá possuir restrições de *hardware* (espaço de armazenamento). Isso implica que, basicamente, deverão ser carregados os elementos principais da aplicação e, de acordo com a demanda da mesma (quando recursos são requisitados e carregados), mais elementos são copiados do carrossel para o receptor. Os objetos do tipo arquivo contêm os dados propriamente ditos; os objetos do tipo diretório contêm a localização (diretório) dos objetos dentro do carrossel e os objetos do tipo *stream* referenciam fluxos elementares (*elementary streams*) dentro do *bitstream* MPEG [ETSI, 2003].

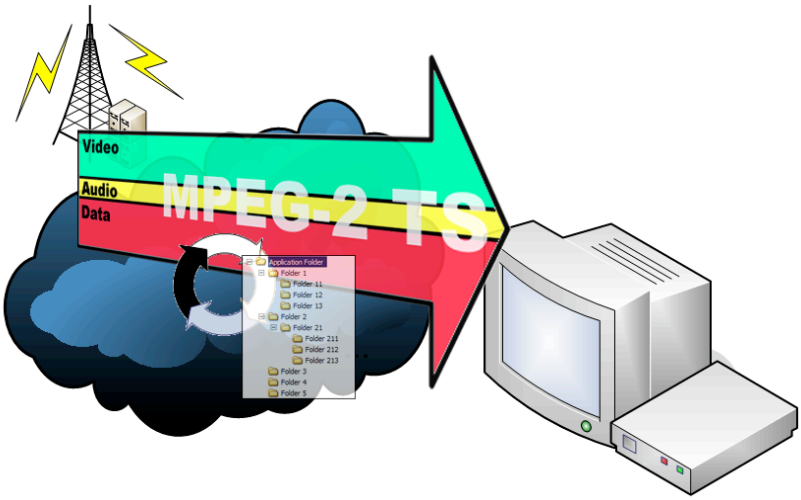

Figura 7. Transmissão DSM-CC de um sistema de arquivos.

2.2 Computação em Grid

Para uma melhor contextualização acerca da teoria referente à computação em *grid*, esta seção foi dividida em duas, versando a primeira sobre *Computação Paralela* e a segunda sobre *Grids Computacionais*.

2.2.1 Computação Paralela

Computação paralela [Wood, 1995][Theys, 2001] consiste na execução simultânea de uma mesma aplicação (dividida e especialmente adaptada) em múltiplos processadores de forma a obter resultados mais rápidos. A idéia é baseada no fato de que o processo de resolução de um problema usualmente pode ser dividido em problemas menores, que podem ser simultaneamente processadas com alguma coordenação. Deste modo, é possível solucionar problemas com maior rapidez, como também é viabilizado o tratamento de problemas de complexidade mais elevada (problemas grandes).

Computação paralela pode existir em diferentes cenários que, basicamente, se distinguem pela forma de interconexão entre processadores e entre processadores e memória, de acordo com a taxonomia de Michael J. Flynn [Tanenbaum, 2006], que também leva em consideração o fato de os processadores estarem ou não executando as mesmas instruções.

De acordo com a lei de Gene Amdahl [Amdahl, 1967][Annavaram, 2005], o fato de a maioria dos algoritmos ser essencialmente seqüencial faz com que o aumento da

velocidade (diminuição do tempo) na resolução de um problema não seja diretamente proporcional ao aumento do número de processadores envolvidos. Em algum ponto, um determinado trecho de algoritmo vai ter que ser executado em um processador apenas. Portanto, muitos algoritmos devem ser remodelados para que façam uso efetivo de arquiteturas paralelas.

Um algoritmo paralelo é aquele que pode ser executado um pedaço de cada vez em múltiplos processadores, com intuito de se obter um resultado geral a partir da junção do conjunto de resultados gerados. Alguns algoritmos são facilmente paralelizados. Por exemplo, paralelizar o trabalho de verificar quais números de 0 a N são primos, pode ser facilmente dividido entre múltiplos processadores atribuindo-se um subconjunto desses números a cada um dos processadores envolvidos, para depois reunir os resultados gerados.

2.2.2 *Grids* computacionais

Grids computacionais surgiram como promessa para computação paralela em recursos geograficamente dispersos e pertencentes a múltiplas organizações. Assim, permitiram plataformas de execução de aplicações distribuídas muito mais baratas que os supercomputadores de poder de processamento equivalente. Além disso, uma gama de aplicações de escala de execução impossível em um simples supercomputador tornaram-se viáveis através dos *grids*[16] computacionais [Cirne Filho, 2006][Cirne Filho, 2005].

Os *grids* computacionais surgiram da idéia de se utilizar computadores independentes e geograficamente dispersos como plataforma de execução de aplicações paralelas. A idéia é prover uma infra-estrutura que viabilize serviços sob demanda, permitindo uma maior colaboração entre diferentes instituições, através do compartilhamento de recursos computacionais, para diferentes finalidades. Algumas características distinguem os *grids* de outras plataformas de execução paralela, conferindo-lhes um maior grau de complexidade. São elas:

- Heterogeneidade – componentes que formam a infra-estrutura possuem diferentes características de *software* e *hardware*;

[16] O termo *Grid* significa "grade" em inglês. Apesar de o termo "grade computacional" também ser difundido, no decorrer desta dissertação a será utilizado apenas o termo *grid* em inglês, com função exclusiva de "grade computacional".

- Alta dispersão geográfica – *grids* agregam serviços localizados em diversas partes do planeta;
- Compartilhamento – *grids* não podem ser dedicados a uma aplicação de forma exclusiva;
- Múltiplos domínios administrativos – utilização de recursos com diferentes políticas de acesso e uso;
- Controle distribuído – como reflexo da dispersão dos componentes, tipicamente não há uma entidade que tenha poder sobre todo o *grid*.

Como *grids* são definidos conceitualmente, uma plataforma que possua todas essas características listadas anteriormente certamente é chamada de *grid*, porém, a ausência de uma dessas características não desclassifica a denominação de uma plataforma como *grid* [Cirne Filho, 2005] [Cirne Filho, 2006].

3. Abordagens Consideradas

Neste capítulo são apresentados alguns trabalhos que apresentam características similares às propostas neste trabalho, e que serviram de inspiração para delinear alguns de seus aspectos. Na seção 3.1 o projeto BOINC (*Berkeley Open Infrastructure for Network Computing*), que é uma plataforma para desenvolvimento de aplicações paralelas com abordagem de "computação voluntária", é brevemente descrito – o modelo de "computação voluntária" e algumas características implementadas pelos projetos que utilizam o BOINC serviram de inspiração para a abordagem inicial dessa dissertação. Na seção 3.2, é apresentado o projeto *OurGrid*, que consiste numa solução para criação de *grids* computacionais na Internet e será utilizado como base para comparação e análise da arquitetura proposta neste trabalho na seção "5. Análise de Desempenho da Arquitetura".

3.1 *BOINC*

O BOINC (*Berkeley Open Infrastructure for Network Computing*) [Anderson, 2004] é um *middleware* para "computação voluntária", cujo desenvolvimento deu-se inicialmente para suporte da aplicação SETI@home[17], mas que dá suporte ao desenvolvimento de aplicações em áreas tão diversas quanto matemática, medicina, biologia molecular e astrofísica. A intenção do BOINC é tornar possível a exploração de recursos ociosos de processamento disponíveis em computadores ao redor do mundo. BOINC é suportado por múltiplas plataformas (*Windows*, *Linux* e *Mac OS*) e licenciado sob a licença LGPL (*GNU Lesser General Public License*)[18]. BOINC foi modelado para ser uma estrutura livre para qualquer um que deseje iniciar um projeto de computação voluntária. A maioria dos projetos que atualmente utilizam esta plataforma não visam lucro e se valem basicamente de voluntários para sua execução – esses projetos são independentes, operando em servidores diferentes.

Projetos que se beneficiam de recursos oferecidos voluntariamente geralmente possuem requisitos adicionais. O primeiro a se analisar é o apelo público, ou seja, a motivação para que um usuário ofereça gratuitamente os seus recursos. Em seguida a minimização do ônus por parte dos usuários, de forma que as aplicações a serem

[17] http://setiathome.berkeley.edu/
[18] http://www.gnu.org/copyleft/lesser.html

utilizadas necessitem trafegar baixas quantidades de dados – todo o tráfego é feito através da Internet, cuja conexão pode incorrer em ônus adicionais para o usuário no caso de geração excessiva de tráfego.

BOINC foi desenvolvido para suportar aplicações que possuam requisitos altos de processamento, armazenamento ou ambos. O requisito básico para tal aplicação é que ela possa ser dividida em um grande número (milhares ou milhões) de tarefas que possam ser executadas independentemente.

BOINC pode ser entendido como um software que visa utilizar ciclos de CPU ociosos durante o uso de um computador convencional, de forma que os mesmos sejam utilizados para executar computação científica. O sistema é composto por um sistema servidor e um software cliente que se comunicam de forma a distribuir, processar e devolver unidades de trabalho (ou tarefas).

A retaguarda de servidores é uma parte vital do sistema BOINC. O servidor pode ser executado em um ou em muitos computadores, de forma que o sistema é facilmente escalável para projetos de vários tamanhos. O software servidor foi desenvolvido para a plataforma Linux, e tem como requisitos de software o servidor HTTP Apache[19], suporte à linguagem de script PHP[20] e o banco de dados MySQL[21]. A computação científica é realizada no computador dos participantes voluntários e os resultados são submetidos aos servidores, que os validam quando do recebimento.

Os servidores BOINC oferecem as seguintes funcionalidades:

- Redundância homogênea – envio de tarefas (*work units*) para computadores que compartilham uma mesma plataforma;
- Distribuição de tarefas gradual – possibilitando o envio de informações ao servidor antes da conclusão de uma tarefa (*work unit*);
- Escalonamento local – envio de tarefas (*work units*) para computadores que já possuam os arquivos necessários para seu processamento;

[19] http://httpd.apache.org/
[20] http://www.php.net
[21] http://www.mysql.com

- Distribuição de tarefas baseadas em parâmetros locais – tarefas (*work units*) que requerem características específicas só serão enviadas para computadores que as atendam.

Os projetos que utilizam o *middleware* BOINC contabilizam aproximadamente 680.000 participantes em 245 países (o que leva a um número aproximado de 1 milhão de computadores).

3.2 OurGrid

O *OurGrid* [Cirne Filho, 2006] constitui uma solução de computação distribuída, para criação de *grids* computacionais. O projeto foi concebido com o intuito de proporcionar, de uma forma simples, o compartilhamento de recursos computacionais ociosos, inicialmente entre laboratórios de pesquisa, seguindo a política do "melhor esforço".

A solução *OurGrid* funciona como uma rede *peer-to-peer* (P2P) onde cada *site*, interessado no compartilhamento de recursos, representa um nó da rede. Para tratar da troca de recursos computacionais entre os *sites*, o *OurGrid* implementa um sistema de créditos, denominado de Rede de Favores. Nesta abordagem, a credibilidade de um *site* é mensurada pela quantidade de tempo que um recurso esteve disponível. Nela a prioridade para um dado *site* A que deseja obter recursos, seja atendido por um *site* B que possui recursos atualmente ociosos, será determinada pelos créditos de A para com B. Nesse exemplo, o tempo de processamento que A já concedeu a B menos o tempo que B já concedeu a A, anteriormente. Deste modo, havendo mais de um *site* requerendo recursos, será atendido aquele que tiver mais créditos.

Embora tal mecanismo de créditos não seja o desejado para utilização em *grids* comerciais, onde modelos econômicos são estudados, a Rede de Favores se mostra de uso adequado em uma comunidade, pois estimulam que os *sites* doem recursos. Isso evita os assim chamados *freerides*: pares (*peers*) que apenas consomem, sem contribuir com a comunidade.

A arquitetura do *OurGrid* (Figura 8) é dividida em 3 componentes: o *MyGrid Broker*, o *OurGrid Peer* e o SWAN.

O *OurGrid Peer* é responsável por controlar os recursos de um site. Ele que concede (ou não) os recursos de seu site a outros pares que os requisitem. O módulo *Peer* também é responsável por buscar mais recursos, quando os de seu site não forem capazes de suprir uma demanda interna.

O *MyGrid* funciona como a interface entre o usuário que necessita de recursos, com o *grid* propriamente dito. Ele provê abstrações para ocultar do usuário a heterogeneidade do *grid*, como também é responsável pelo escalonamento das aplicações.

O SWAN é uma solução de *sandboxing*, responsável por executar as tarefas nas máquinas finais do *grid*, porém com a preocupação de proteger o recurso local de códigos desconhecidos. Execução remota traz claramente problemas de segurança, portanto uma solução que isole a aplicação, de modo a proteger o recurso de aplicações maliciosas, se faz necessária.

O foco do *OurGrid* são as aplicações *Bag-of-Task* (BoT). Aplicações BoT são aquelas nas quais é possível dividir o problemas em tarefas independentes, onde a ordem em que elas são executadas não interfere na resolução do problema. Tais aplicações se adequam melhor a dinamicidade, heterogeneidade e ampla distribuição de um *grid*.

Aplicações onde as tarefas requeiram uma alta conectividade, normalmente paralelizadas para execução em clusters, não são tratadas pelo *OurGrid*. Entretanto é possível que os pares (*peers*) possuam clusters em seus *sites*, e estes sejam compartilhados usando o *grid*.

A limitação do escopo às aplicações BoT restringe o leque de possibilidades do *OurGrid*. Contudo, tais aplicações já englobam uma série de problemas importantes, tais como *data mining*, processamento de genoma, simulações de Monte Carlo, processamento de imagens, dentre outras.

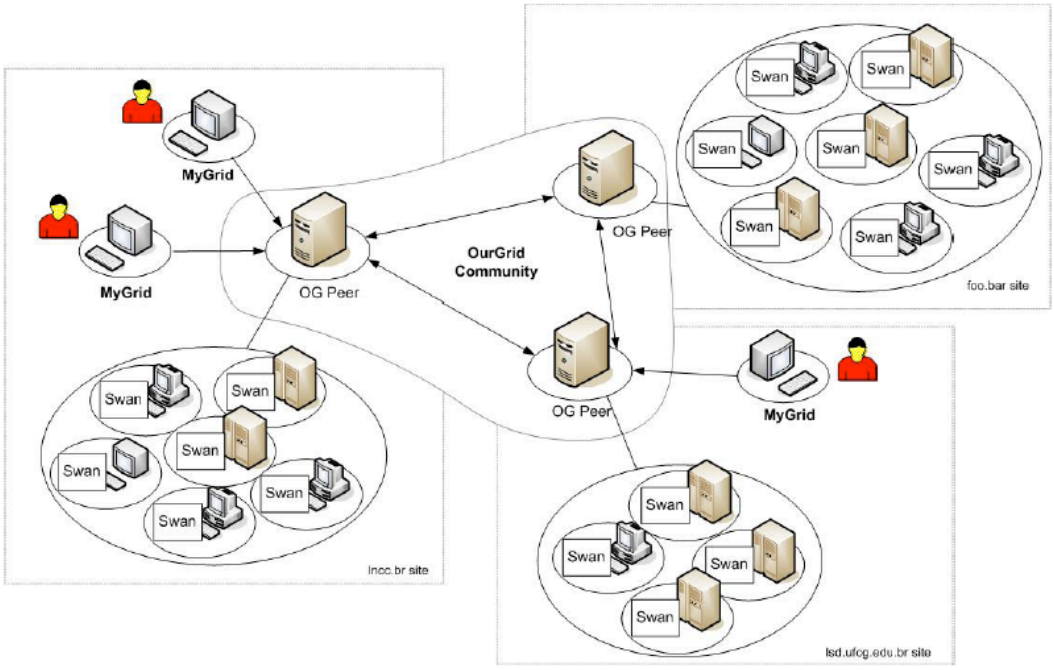

Figura 8. Visão de alto nível dos componentes do OurGrid[22]

[22] http://www.ourgrid.org/twiki-public/pub/OG/ManualOverview3_3/component_ourgrid2.png

4. Arquitetura do TVGrid

A motivação para o desenvolvimento do TVGrid é bastante similar às plataformas de computação voluntária – como o projeto pioneiro SETI@home [Anderson, 2004]. A arquitetura do TVGrid viabiliza a utilização de recursos que seriam desperdiçados: banda de transmissão do canal e capacidade de processamento do receptor de TV Digital. A arquitetura proposta nessa dissertação é baseada na patente de utilidade MU8600875-7 [Batista, 2006] e apresentada em "*TVGrid: A Grid Architecture to use the idle resources on a Digital TV network*" [Batista, 2007].

Nesta seção será detalhada a camada de *software* que deve ser incorporada à uma rede de TV Digital de forma que, utilizando as atuais tecnologias de TV Digital – particularmente as tecnologias de *middleware* incorporadas pelos padrões ITU-T J.200, J.201 e J.202 – seja possível utilizar a infra-estrutura disponível para computação em *grid*.

4.1 Elementos da Arquitetura

Para melhor entender a arquitetura proposta para o TVGrid, uma analogia entre a arquitetura de uma rede de TV Digital e uma arquitetura para computação paralela (vista graficamente na Figura 9).

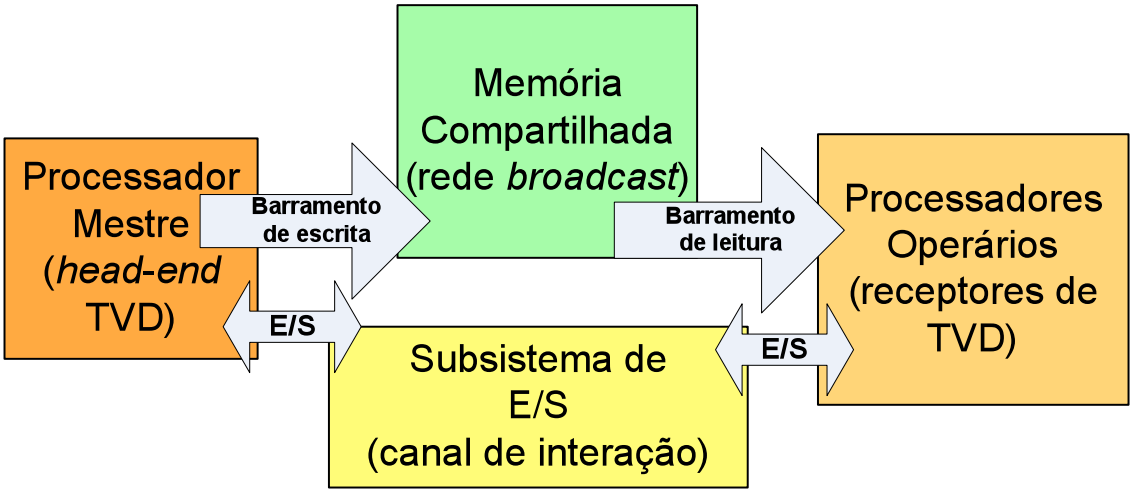

Figura 9. Os componentes de uma arquitetura de computação paralela representados como componentes de uma rede de TV Digital.

Como pode ser visto na Figura 9, é possível modelar um sistema de televisão digital como um computador paralelo com quatro classes de elementos: as unidades de processamentos, a memória compartilhada, o sistema de Entrada e Saída e os barramentos que conectam esses elementos. Serão levados em consideração dois tipos de processadores: os mestres e os operários. Os processadores mestres só poderão escrever na memória compartilhada, enquanto que os processadores operários só poderão ler da memória compartilhada. Nesta arquitetura o processador mestre é responsável por escrever na memória compartilhada as aplicações e os dados a serem processados pelos processadores operários. Os processadores operários acessam a memória compartilhada, lêem as aplicações e as executam. Qualquer dado necessário para a execução da aplicação será também lido da memória compartilhada. A saída do processamento é escrita no sistema de Entrada e Saída pelos processadores operários e lidas de lá pelo processador mestre.

Os seguintes componentes integram a arquitetura do TVGrid: uma estação de TV equipada com um escalonador de tarefas[23], que atua como o processador mestre; uma rede *broadcast* (terrestre, satélite ou cabo) para transmissão de sinal de TV Digital, que atua como memória compartilhada com o suporte da abstração de carrossel de objetos; uma rede de interação (que pode ser a Internet, por exemplo) que atua como o sistema de Entrada e Saída; e finalmente um conjunto de receptores de TV Digital, que atuam como processadores operários.

4.1.1 Processador Mestre

O processador mestre é o componente responsável por distribuir as tarefas, através da rede de broadcast, para que os processadores operários as executem. Para que uma estação de TV Digital possa atuar como o processador mestre da arquitetura do TVGrid, a mesma deve possuir um Escalonador de Tarefas conectada ao seu Gerador de Carrossel (vide Figura 1 e Figura 3), disponibilizando para o mesmo os arquivos que compõem a tarefa: a aplicação *Xlet* e outros arquivos necessários para execução da tarefa. Esses arquivos serão então serializados pelo Gerador de Carrossel, que os injetará no multiplexador para que a tarefa possa ser transmitida junto com a programação do canal em questão.

[23] Que são, no caso, aplicações de TV Digital.

4.1.2 Memória compartilhada

A memória compartilhada da arquitetura proposta é o meio físico de comunicação utilizado pela estação de TV para transmissão do sinal digital. Apenas a estação de TV possui acesso de escrita nesse meio, de tal forma que a programação do canal de TV (conteúdo audiovisual) multiplexada com dados (aplicações, informações de SI, etc) será escrita na memória para que seja lida pelos receptores de TV Digital – configurando assim uma transmissão *broadcast* (na qual o meio é compartilhado e a comunicação se dá de um para todos).

4.1.3 Sistema de Entrada e Saída

O sistema de entrada e saída da arquitetura proposta é caracterizado pelo canal de interação *full-duplex* – comumente uma conexão com a Internet – que liga a estação de TV (processador mestre) e os receptores (processadores operários). Este canal de interação é utilizado para a transmissão do resultado processado pelos receptores para a estação de TV, para que o Escalonador de Tarefas faça registro.

4.1.4 Processadores operários

Os processadores operários são os receptores de TV Digital capazes de executar as aplicações interativas multiplexadas junto à programação do canal – na nossa arquitetura aplicações *Xlet* compatíveis com GEM. Esses receptores devem estar conectados ao canal de interação (sistema de Entrada e Saída), para que possam enviar à Estação de TV, ao término do processamento, o resultado de uma tarefa.

4.2 Escalonando tarefas no TVGrid

Por conta de características particulares dos canais de comunicação que conectam os componentes de um sistema de TV Digital, e do fato de que os receptores em alguns casos não podem se comunicar uns com os outros, a arquitetura do TVGrid se adequa melhor para executar aplicações paralelas cujas aplicações não façam uso desse tipo de comunicação. Essas aplicações são definidas na literatura como *Embarrassingly Parallel Applications* ou *Bag-of-tasks* (BoT) *applications* [Wood, 1995][Theys, 2001]. Ademais, como a rede de interação pode possuir uma largura de banda que ofereça uma capacidade de transmissão limitada e seu uso excessivo pode impor custos adicionais ao modelo, um sistema TVGrid deve ser preferencialmente aplicado na resolução de problemas cuja saída (os dados que compõem a resolução da tarefa) não é um grande volume de dados.

O escalonador de tarefas instalado na Estação de TV é responsável por controlar o uso da infra-estrutura do TVGrid. Nesta seção serão discutidas duas lógicas possíveis para a implementação do escalonador de tarefas: a primeira é bastante simplista e deve ser utilizada com aplicações paramétricas e a segunda é um pouco mais complexa e deve ser utilizada com aplicações BoT.

4.2.1 O Escalonador de Aplicações Paramétricas

Uma aplicação paramétrica é uma aplicação paralela onde todas as tarefas realizam a mesma computação, apenas diferindo nos valores de entrada (parâmetros), que são gerados pela própria aplicação (ou seja, não há arquivos de entrada). As Simulações de Monte Carlo [Metropolis, 1949] são exemplos bastante conhecidos desse tipo de aplicação, visto que são aplicações que requerem múltiplas execuções de um mesmo conjunto de instruções para estimação e testes de hipóteses existentes.

Na maioria dos casos, uma aplicação paramétrica deve ser implementada de forma que ela não necessite ler nenhum dado de entrada. Cada tarefa da aplicação primeiro gera sua própria entrada de maneira aleatória e executa as instruções da tarefa considerando os parâmetros gerados. Cada tarefa é composta por um laço (*loop*) infinito no qual são efetuadas as seguintes instruções:

1. geração aleatória dos parâmetros de processamento;
2. processamento da tarefa utilizando os valores gerados;
3. o resultado do processamento é enviado de volta ao escalonador na estação de TV Digital, através da rede de interação, junto com os valores gerados.

Esses três passos são repetidos indefinidamente enquanto o receptor de TV Digital estiver sintonizado no canal que estiver sendo utilizado pelo escalonador paramétrico, visto que a troca de canal pode ocasionar o fim da aplicação em alguns sistemas de TV Digital.

Dessa forma, o escalonador de tarefas que for ser executado em uma estação de TV é muito simples. Esta aplicação basicamente incluirá a aplicação paralela no carrossel de objetos de forma que o receptor de TV Digital sintonizado no canal utilizado pelo escalonador de tarefas identifique que há uma aplicação multiplexada e

que a mesma deve ser executada – para isso o receptor utiliza informações constantes na AIT (descrita na seção "2.1.4.1.1 AIT – *Application Information Table*").

É importante ressaltar que, como o DSM-CC se vale de um mecanismo que envia os mesmos dados repetidamente, qualquer receptor de TV Digital que sintonizar em um canal contendo aplicações disponíveis as irá carregar e executar, independente de quando ocorra a sintonia.

Sempre que o escalonador receber um resultado através da rede de interação, o mesmo deverá checar se os valores utilizados como entrada para o processamento enviado é diferente de todos os valores utilizados em tarefas já executadas. Se essa condição for atendida, o resultado é armazenado como parte da saída geral da aplicação, caso contrário esse resultado é descartado.

Quando um número suficiente de tarefas são completadas, o escalonador pode iniciar a execução de outra aplicação utilizando a mesma estratégia, atualizando a aplicação no carrossel de objetos.

4.2.2 O Escalonador de aplicações BoT

Aplicações BoT são compostas de um conjunto de tarefas independentes. Cada tarefa define uma aplicação que normalmente lê dados de um arquivo de entrada, processa-os, e escreve o resultado do processamento em um arquivo de saída.

Em um sistema implementando a arquitetura TVGrid, uma tarefa é executada com o suporte de uma aplicação chamada de *trigger* (gatilho). A aplicação *trigger* é escrita no carrossel de objetos e carregada por todos os receptores que sintonizarem o canal utilizado pelo escalonador durante a transmissão do carrossel. Depois de ser carregada, a aplicação *trigger* é responsável pela execução de tarefas de uma aplicação BoT. A aplicação *trigger* copia os dados referentes à tarefa a ser executada do carrossel de objetos para a memória local e não-volátil do receptor de TV Digital e finalmente executa a aplicação, armazenando o resultado também na sua memória local e não-volátil. Quando o programa finaliza sua execução, a aplicação *trigger* executa as seguintes instruções:

1. envia o resultado do processamento da tarefa para a estação de TV utilizando a rede de interação;

2. remove da memória local todos os arquivos de entradas e saídas relacionados à execução da tarefa anterior;

3. inicia o processo de execução de uma nova tarefa, copiando os dados de entrada novamente do carrossel de objetos para a memória local.

Para que as tarefas de uma aplicação BoT sejam executadas, o escalonador de tarefas deve escrever no carrossel de objetos tanto a aplicação *trigger* quanto os arquivos de entrada e elementos de *software* a serem executadas por cada tarefa que compõe a aplicação (como visto graficamente na Figura 10). Cada tarefa é transmitida em um *slot* que pode ser representado por um diretório em um sistema de arquivos. Assim, a aplicação *trigger* pode escolher a tarefa simplesmente escolhendo um *slot* aleatoriamente e executando sua tarefa correspondente.

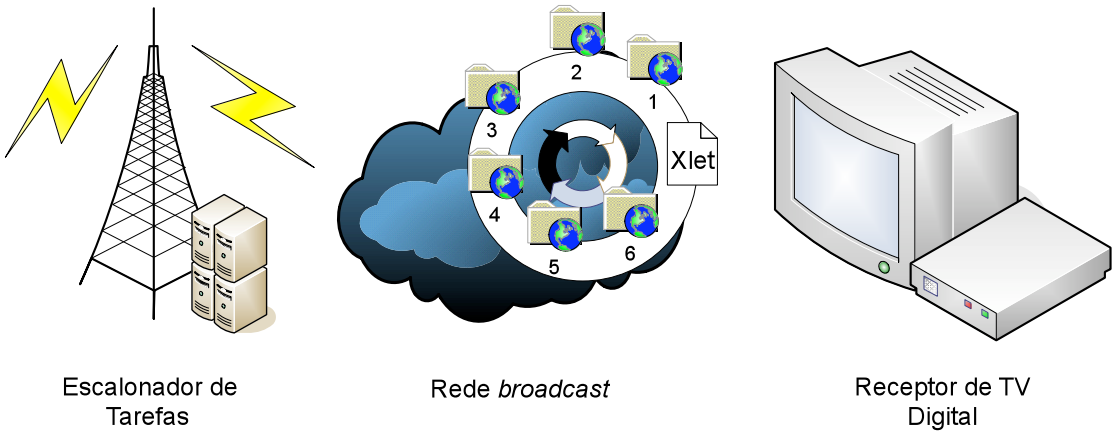

Figura 10. Escalonador de tarefas multiplexando um carrossel de objetos no fluxo de vídeo com a aplicação *trigger* (um *Xlet*) e as 6 tarefas em 6 *slots* (os diretórios) para o receptor de TV Digital.

A mesma tarefa pode ser executada em paralelo por mais de um receptor. O escalonador de tarefas deve ser responsável por identificar o recebimento do processamento replicado de tarefas. Essa redundância é necessária para garantir que todas as tarefas sejam realizadas (possibilidade estatística), apesar das possíveis falhas nos receptores. Como previamente mencionado, uma estação de TV não possui controle sobre os receptores de TV Digital – o telespectador pode trocar de canal ou desliga-lo a qualquer momento, o que causaria a finalização de aplicações procedurais em *middlewares* aderentes ao GEM [DVB, 2007]. Os middlewares ATSC OCAP [CableLabs, 2005] e ATSC ACAP [ATSC, 2005] incorporam uma nova tabela de sinalização de aplicações, chamada XAIT, que viabiliza a execução de aplicações

descontextualizadas, ditas *unbounded*. Estas aplicações não são finalizadas quando da troca de canal. O *middleware* brasileiro Ginga não utiliza a XAIT, porém possui uma classe de aplicações definida em sua AIT que possui comportamento similar, aplicações ditas desacopladas (*DETACHED*).

O escalonador de tarefas registra quais tarefas foram executadas e para isso utiliza uma estrutura de dados chamada de *bag* (saco). Todas as tarefas a serem agendadas são armazenadas neste *bag* e cada vez que o carrossel de objetos é montado para ser transmitido no canal de TV Digital, o escalonador seleciona tarefas aleatórias do *bag* e preenche os *slots* definidos para serem transmitidos no carrossel de objetos. Quando uma tarefa é concluída (seu resultado é recebido pela estação de TV através da rede interação) o escalonador de tarefas remove-a do *bag*. A execução da aplicação se encerra quando o *bag* se esvazia, ou seja, todas as suas tarefas são concluídas.

A latência da leitura dos dados de um carrossel é proporcional ao seu tamanho. Assim sendo, nem sempre é interessante que o escalonador de tarefas reserve muitos *slots* no carrossel de objetos (tantos *slots* quanto tarefas, por exemplo). Como o carrossel de objetos é transmitido em ciclos, um carrossel muito grande pode dificultar o recebimento da aplicação *trigger* por parte dos receptores de TV Digital – quando um receptor sintoniza o canal logo após a transmissão da aplicação *trigger* o mesmo deve esperar que o carrossel inteiro seja transmitido para, enfim, poder carregar o *Xlet*.

A estratégia do escalonador de tarefas é de manter um número reduzido de *slots* no carrossel a ser transmitido. Na sua implementação mais simples, o escalonador de tarefas preenche os *slots* do carrossel com as tarefas em seqüência. Dessa forma, uma aplicação com $\boldsymbol{s}$ tarefas é completamente transmitida utilizando-se um carrossel de objeto com $\boldsymbol{c}$ *slots* após $\lceil s/c \rceil$ "rodadas" do carrossel. Obviamente, no caso de uma ou mais tarefas não concluídas (tarefas cujos resultados não foram recebidos pela estação de TV Digital), rodadas extras se farão necessárias – uma tarefa pode não ser concluída se, por exemplo, todos os receptores de TV Digital que a requisitaram foram desligados antes de sua conclusão.

Caso o número de tarefas seja pequeno[24], um carrossel pode conter *slots* com tarefas repetidas, aumentando o grau de redundância e a probabilidade de conclusão das tarefas.

A aplicação *trigger* carrega os índices (o caminho de todos os diretórios disponíveis, por exemplo) dos *slots* disponíveis no carrossel, escolhendo um *slot* aleatório para processar sua respectiva tarefa. Entretanto, como é provável que o número de receptores executando as tarefas de uma determinada aplicação seja bem maior que o número de *slots* disponibilizados pelo escalonador de tarefas, o grau de replicação talvez seja maior que o necessário.

Uma abordagem para diminuir o nível de redundância é fazer com que a aplicação *trigger* tenha um fator de latência aleatório, definindo o teto deste fator como o tempo médio de execução de uma tarefa, de forma que alguns receptores percam algumas rodadas do carrossel quando, por exemplo, tiverem finalizado uma tarefa. Essa abordagem faz com que esses receptores tenham acesso a tarefas que certamente não foram concluídas, aumentando o paralelismo do sistema TVGrid.

O número de *slots* definido pelo escalonador de tarefas dependerá do tamanho médio das tarefas de uma aplicação em particular, na redundância necessária e no tempo médio que um telespectador permanece sintonizado em um determinado canal (que aproximadamente define o número de tarefas um determinado receptor processa em média).

A Figura 11 representa graficamente o procedimento descrito abaixo:

1) Aplicação *trigger* (*flag autostart*) é carregada pelo receptor;
2) receptor gera o valor do índice da tarefa a ser carregada;
3) receptor carrega tarefa (de acordo com valor gerado);
4) tarefa é processada e resultado é enviado de volta para o escalonador (estação de TV), que remove-a do "saco de tarefas";
5) uma nova tarefa é inserida no *slot* da tarefa processada no carrossel transmitido.

[24] Uma quantidade "pequena" de tarefas é aquela que pode ser transmitida para um receptor de TV Digital em um tempo consideravelmente menor que a média de permanência de sintonia em um determinado canal de TV.

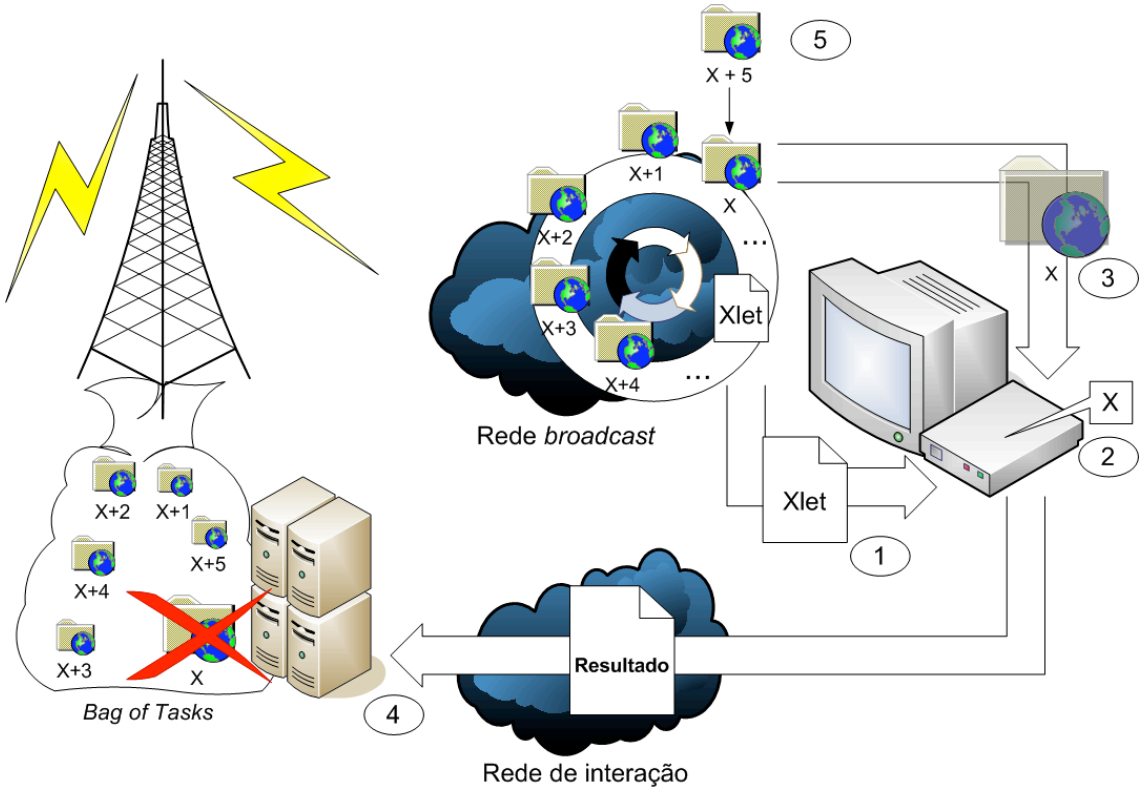

Figura 11. Processamento de tarefas no TVGrid

A utilização de mais de um canal de TV Digital para transmitir uma aplicação é certamente uma estratégia que aumenta o desempenho do sistema, visto que a tendência é que existam mais receptores sintonizados e é fato que o escalonador de tarefas disporá de mais *slots* para transmitir tarefas. Esta estratégia pode ser otimizada, visto que, utilizando canais diferentes, o escalonador de tarefas terá de considerar também parâmetros diferentes (o tempo médio de sintonia por telespectador, por exemplo) e logo pode transmitir uma quantidade diferente de *slots* em cada canal. Cada canal terá uma aplicação *trigger* diferente, visto que a quantidade de índices (referentes aos *slots*) será diferente em cada canal – este modelo é mais facilmente aplicável em redes de TV por assinatura onde uma estação controla a transmissão de múltiplos canais de TV em um mesmo meio.

5. Análise de Desempenho da Arquitetura

Neste capítulo a arquitetura proposta para o TVGrid é analisada, de forma a definir o universo de aplicações onde a mesma possui bom desempenho. A análise foi conduzida de acordo com características da arquitetura e de valores obtidos em testes representativos.

5.1 Modelagem do Sistema

Como previamente estabelecido, um sistema TVGrid é composto por uma estação de TV Digital, um canal *broadcast* com uma capacidade de transmissão de β bps, N receptores de TV Digital e um canal de retorno com capacidade de transmissão de ρ bps. É definido α como a fração dos receptores de TV Digital disponíveis em um dado momento, ou seja, os receptores sintonizados em um determinado canal a ser usado por um escalonador de tarefas.

Uma aplicação TVGrid paramétrica pode ser definida com uma tupla $A = (sz,pt)$,onde $A.sz$ é o tamanho em bits da aplicação e $A.pt$ é o tempo necessário para A ser processada em um receptor de TV Digital de referência. Similarmente uma aplicação TVGrid BoT (também definida como *job*) pode ser definida como uma tupla (A, T), onde A é aplicação *trigger* e T é um conjunto de tarefas, $T = \{ t_1, t_2, \ldots, t_s\}$, que serão executadas por A. Cada tarefa t em T é definida por uma tupla $t = (sz, pt)$, onde $t.sz$ é o seu tamanho em bits (tamanho da entrada para a aplicação *trigger*) e $t.pt$ é o tempo em milissegundos necessário para se concluir t em um receptor de TV Digital de referência.

5.2 Métricas de Desempenho

Para medir o desempenho da arquitetura definida para o TVGrid, será apresentado um levantamento acerca de qual é o tempo médio de execução (*makespan*) das aplicações TVGrid. O cenário considerado possui receptores de TV Digital homogêneos, ou seja: todos os receptores possuem a mesma capacidade de processamento – como previamente estabelecido, considera-se que todos os receptores são análogos a um receptor de referência. Neste cenário também, o tráfego gerado no canal de retorno (rede de interação) é pequeno o suficiente para que o tempo de transferência do resultado do processamento seja desconsiderado. O "dado momento"

que é considerado quando α é definido para este cenário, é o tempo necessário para executar todas as tarefas, ou seja, o tempo para concluir a aplicação *A*.

Definido *s* como sendo o número de tarefas em uma aplicação paralela – no caso das aplicações TVGrid paramétricas, *s* é o número de execuções diferentes necessárias. Considerando que todas as tarefas possuam o mesmo tamanho em bits o mesmo tempo médio de processamento, o valor médio do *makespan* para aplicações TVGrid paramétricas (M_p) é dado pela equação:

$$M_p = \frac{A.sz}{\beta} + \left\lceil \frac{s}{N \times \alpha} \right\rceil \times A.pt \quad (1)$$

Como no cenário considerado estabelece-se que existem $N \times \alpha$ receptores de TV Digital disponíveis para processamento em *grid* no momento que a estação de TV inicia a transmissão da aplicação, e cada um carregará a aplicação paramétrica *A* em $A.sz/\beta$ segundos. Cada receptor disponível executará, então, $\lceil s/(N \times \alpha) \rceil$ tarefas (número de execuções).

Para aplicações TVGrid BoT, considerando que cada tarefa só será transmitida no carrossel uma vez para ser concluída (N >> c), a seguinte fórmula define o *makespan* médio (M_{BoT} para *s* tarefas e *c slots* no carrossel):

$$M_{BoT} = \left\lceil \frac{s}{c} \right\rceil \times \left(\frac{A.sz + c \times t.sz}{\beta}\right) + t.pt \quad (2)$$

Esse *makespan* corresponde ao tempo de sequencialmente escrever a aplicação *trigger* e todas as tarefas da aplicação *A* para transmissão no carrossel de objetos (em $\lceil s/c \rceil$ rodadas do carrossel) mais o tempo de processamento das tarefas escritas na última rodada do carrossel – as tarefas anteriores são executadas em paralelo à escrita de novas tarefas no carrossel.

Como especificado, cada tarefa será concluída com sucesso na primeira vez que for transmitida. Para que isso seja possível, é necessário que o tempo de processamento de uma tarefa (*t.pt*) seja:

$$t.pt \leq \left(\left\lfloor \frac{N \times p_{\alpha}}{c} \right\rfloor - 1 \right) \times \left(\frac{A.sz + c \times t.sz}{\beta} \right) \quad (3)$$

Como forma de avaliação da arquitetura proposta, serão apresentadas comparações do TVGrid com uma arquitetura de computação em *grid* baseada no paradigma *peer-to-peer* (p2p) [Cirne Filho, 2006], escolhida por conta de sua característica de livre acesso (*free-to-join*), que faz com que esse tipo de *grid* geralmente agregue muito mais recursos que *grids* baseados no conceito de organizações virtuais (*virtual enterprises*).

Em um *grid peer-to-peer* é assumido que todos os dados relacionados à execução de uma determinada aplicação são armazenados no dispositivo (geralmente um computador pessoal) que o usuário do *grid* usa para acessá-lo. Definindo N_{p2p} e β_{p2p} como, respectivamente, o número de processadores operários que estão disponíveis para uma determinada aplicação (e que permanecerão disponíveis durante toda sua execução) e a banda média disponível da conexão que liga a máquina do usuário aos processadores operários do *grid*. Considerando λ (onde $0 < \lambda \leq 1$) como o fator de *speed up* de um operário no *grid peer-to-peer* quando comparado a um receptor de TV Digital de referência – um processamento que requer *p* unidades de tempo para ser executado em um receptor de TV Digital requer $p \times \lambda$ unidades de tempo em um processador operário de referência.

Executar uma aplicação paramétrica em um *grid peer-to-peer* obviamente requer que cada processador operário a possua. Assim, o *makespan* médio para uma aplicação paramétrica em um *grid peer-to-peer* (M_p`) é:

$$M_p' = \frac{\min\{s, N_{p2p}\} \times A.sz}{\beta_{p2p}} + \left\lceil \frac{s}{N_{p2p}} \right\rceil \times A.pt \times \lambda \quad (4)$$

Para aplicações BoT, cada uma das tarefas deve ser distribuída entre os processadores operários a executar a aplicação paralela. Assim, leva-se em consideração o custo de envio das tarefas para os processadores operários mais o tempo de processar essas tarefas. Em qualquer instante do tempo tem-se N_{p2p} tarefas sendo executadas em

paralelo, e o *makespan* médio para uma aplicação BoT em um *grid peer-to-peer* (M_{BoT}`) é:

$$M_{BoT}' = \frac{s \times t.sz}{\beta_{p2p}} + t.pt \times \lambda \quad (5)$$

O escalonador do *grid peer-to-peer* só envia uma nova tarefa para os processadores operários quando o mesmo já tiver concluído qualquer tarefa que tenha recebido. Isso acontece se:

$$t.pt \leq \left(N_{p2p} - 1\right) \times \left(\frac{t.sz}{\lambda \times \beta}\right) \quad (6)$$

5.3 Testes

Testes foram conduzidos para a obtenção de dados que embasassem uma análise comparativa baseada na modelagem apresentada na seção anterior. A seguir detalharemos como foram conduzidos os testes que viabilizaram um cálculo mais preciso do valor médio do *makespan* para aplicações TVGrid paramétricas e para aplicações TVGrid BoT.

A infra-estrutura utilizada para os testes é visualmente apresentada na Figura 12 e descrita abaixo:

- *Head-end* da estação (Processador mestre)
  - Codificador MPEG-2 Tandberg® ES5780
  - Encapsulador IP Tandberg® TT6120
  - Modulador QAM Tandberg® TT6120
  - PC Dell® Optiplex GX 620 (Intel® Pentium® 4 3GHz, 1Gb de RAM e 1024Kb de *cache*) com um escalonador de tarefas e software remultiplexador instalado.
- Rede *broadcast* (memória compartilhada)
  - Cabo coaxial conectando o modulador QAM e o receptor.

- Rede de interação (sistema de entrada e saída – E/S)
    - *Switch* Gigabit Ethernet Dell PowerConnect 5012.
- Receptor de TV Digital (processador operário)
    - *Set-top box* de referência Intel® Kalaheo (Mobile Intel® Celeron™ CPU 1266MHz, 256Mb de RAM e 256Kb de *cache*) com uma placa DVB-C Cable2PC B2C2 FlexCop e o *middleware* Ginga (compatível com GEM) instalado.

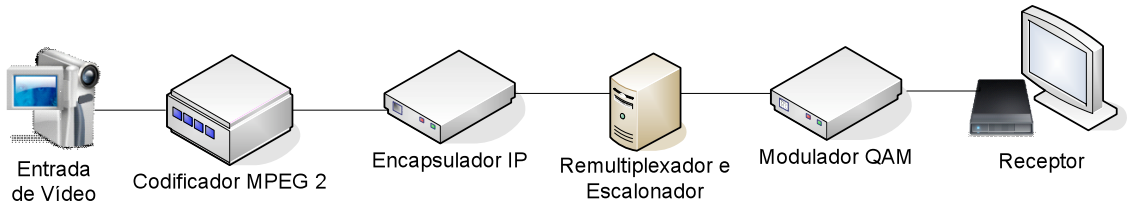

Figura 12. Componentes da infra-estrutura utilizada para os testes.

O procedimento de testes, em ambos os casos, foram conduzidos de forma que o codificador de vídeo enviasse o fluxo de vídeo codificado para o encapsulador IP, que retransmite esse fluxo via UDP com taxa constante (CBR – *Constant bit rate)* para o remultiplexador para o PC com o software remultiplexador e Escalonador de Tarefas instalado; o vídeo é então multiplexado com o carrossel contendo os dados referentes às tarefas a serem executadas, e enviado para o modulador QAM, que transmite o fluxo resultante para o receptor de TV Digital, que finalmente executa a tarefa, enviando ao escalonador o resultado do processamento e o tempo decorrido.

Para estimar o valor do fator de *speed up* de um operário em um *grid peer-to-peer* (λ) para as duas categorias de problemas, todas as tarefas processadas por um receptor de TV Digital foram também executadas por um computador convencional (Intel® Pentium® 4 CPU 3.00GHz, com memória RAM de 512Mb e *cache* de 1024 KB), com resultados armazenados para análise comparativa do modelo proposto.

5.3.1 Testes com aplicação paramétrica

Os testes foram feitos com base em uma implementação (sem saída gráfica) do método de Monte Carlo, que é uma categoria de algoritmos que se vale de repetições de um mesmo conjunto de instruções com parâmetros gerados aleatoriamente. A aplicação desenvolvida é um *Xlet* que calcula o valor aproximado de π, usando o método de

Monte Carlo. O algoritmo utilizado para o cálculo do valor estimado de π, que recebe como entrada a quantidade de iterações a serem realizadas e um valor *seed* para geração do número aleatório, é apresentado a seguir:

1. **para** P_{total}= 1 até n **faça**
2. x ← número aleatório entre 0 e 1
3. y ← número aleatório entre 0 e 1
4. **se** ($x^2+y^2 \leq 1$) **então**
5. $P_{in} \leftarrow P_{in} + 1$
6. **fim se**
7. **fim para**
8. $\pi \leftarrow 4 * P_{in} / P_{total}$

A execução dos testes se deu com base na definição de cinco tarefas representativas a serem executadas múltiplas vezes no receptor de TV Digital e no PC convencional – foram executados dez conjuntos de dez execuções (apresentadas como iterações) para cada tarefa. Cada tarefa é definida com base nos valores de entrada para a aplicação (quantidade de execuções e *seed*), para que a carga de processamento seja proporcional por tarefa. Os dados produzidos são apresentados nas tabelas a seguir.

Tabela 1. Tempo de execução (em milissegundos) das tarefas de uma aplicação paramétrica executada em um receptor de TV Digital.

Iteração	Tarefa 1	Tarefa 2	Tarefa 3	Tarefa 4	Tarefa 5
1	3099	340	12380	1257	1235
2	3100	344	12379	1266	1227
3	3099	337	12306	1257	1234
4	3099	337	12380	1258	1467
5	3098	338	12306	1266	1235
6	3098	339	12379	1308	1235
7	3099	338	12380	1266	1234
8	3099	337	13947	1258	1227
9	3100	337	12380	1266	1228
10	3101	343	12306	1265	1234
Maior valor	3101	344	13947	1308	1467
Média	3099,2	339	12514,3	1266,7	1255,6
Mediana	3099	338	12379,5	1265,5	1234
Desvio Padrão	0,918936583	2,581988897	504,595999	15,0926767	74,35679749

Variância	0,844444444	6,666666667	254617,1222	227,788889	5528,933333

Tabela 2. Tempo de execução (em milissegundos) das tarefas de uma aplicação paramétrica executadas em um PC convencional.

Iteração	Tarefa 1	Tarefa 2	Tarefa 3	Tarefa 4	Tarefa 5
1	3008	309	11984	1207	1174
2	3007	309	11974	1208	1183
3	3005	311	11990	1206	1174
4	3011	309	11997	1205	1182
5	3008	309	11974	1209	1180
6	2998	308	11592	1206	1177
7	3005	308	11965	1202	1174
8	3014	309	11998	1209	1178
9	3003	309	11965	1203	1172
10	3016	308	11994	1207	1176
Maior valor	3016	311	11994	1209	1183
Média	3007,5	308,9	11943,3	1206,2	1177
Mediana	3007,5	309	11979	1206,5	1176,5
Desvio Padrão	5,275731	0,875595	124,0619	2,347576	3,711843
Variância	27,83333	0,766667	15391,34	5,511111	13,77778

Com base nos valores apresentados, podemos definir que para aplicações paramétricas o valor do fator de *speed up* de um operário em um *grid peer-to-peer* (λ_p) como 0,95.

5.3.2 Testes com aplicação BoT

Para o teste com aplicação BoT foi utilizado o renderizador de imagens *Sunflow*[25], que é um software livre desenvolvido na linguagem de programação Java e projetado para processamento seqüencial. A aplicação BoT desenvolvida com base na arquitetura TVGrid é composta um *Xlet* que possui uma instância do *Sunflow* alterada, de forma que a renderização fosse realizada apenas sobre uma fração da imagem, além de uma imagem a ser renderizada dividida em frações – cada fração representa uma tarefa e o *Xlet* é a aplicação *trigger*.

A imagem foi dividida em dez frações, e cada fração processada dez vezes. O resultado dos testes é apresentado a seguir:

Tabela 3. Tempo de execução (em segundos) das tarefas de uma aplicação BoT em um receptor de TV Digital.

iteração	Tarefa 1	Tarefa 2	Tarefa 3	Tarefa 4	Tarefa 5	Tarefa 6	Tarefa 7	Tarefa 8	Tarefa 9	Tarefa 10
1	7,72	23,76	88,44	120,31	114,1	94,99	45,24	25,34	11,2	10
2	10,26	23,91	88,73	121,08	115,07	95	45,29	25,38	11,21	10,93

[25] http://sunflow.sourceforge.net/

3	10,88	24,77	89,63	121,15	114,37	95,15	45,29	26,48	11,29	11
4	10,82	23,76	88,81	121,17	115,35	95,96	45,23	26,04	11,2	10,98
5	10,77	23,74	87,6	121,1	113,45	95,96	45,35	26,11	11,54	11
6	10,99	23,74	88,8	121,41	113,94	96,17	45,23	25,98	11,18	10,97
7	10,86	23,76	88,6	121,04	114,22	95,88	45,25	26,2	11,19	10,87
8	10,18	22,86	88,77	121,02	114,16	96,14	45,24	25,98	11,29	11,01
9	10,87	22,89	88,52	120,25	114,29	96,08	45,58	25,18	11,3	10,89
10	11,67	22,78	88,44	121,25	114,02	96,09	45,31	26,12	10,45	11
Maior valor	11,67	24,77	89,63	121,41	115,35	96,17	45,58	26,48	11,54	11,01
Média	10,502	23,597	88,634	120,978	114,297	95,742	45,301	25,881	11,185	10,865
Mediana	10,84	23,75	88,665	121,09	114,19	95,96	45,27	26,01	11,205	10,975
Desvio Padrão	1,05848	0,6059	0,49853	0,38501	0,54739	0,48962	0,10577	0,42818	0,27941	0,30786
Variância	1,12039	0,3672	0,24853	0,14824	0,29964	0,23972	0,01118	0,18334	0,07807	0,09478

Tabela 4. Tempo de execução (em segundos) das tarefas de uma aplicação BoT em um PC convencional.

iteração	Tarefa 1	Tarefa 2	Tarefa 3	Tarefa 4	Tarefa 5	Tarefa 6	Tarefa 7	Tarefa 8	Tarefa 9	Tarefa 10
1	4,32	11,40	42,34	59,10	53,36	45,59	22,16	13,22	4,12	3,73
2	5,48	10,55	42,92	61,28	53,58	48,58	21,17	11,52	4,08	3,70
3	5,13	11,49	42,23	60,34	55,71	46,56	21,73	13,07	4,04	3,64
4	5,01	9,87	42,96	58,36	57,52	47,56	22,01	12,68	4,09	5,26
5	5,14	10,25	42,27	59,66	61,53	46,66	21,76	12,73	4,85	5,81
6	3,66	11,35	42,59	58,45	53,35	49,17	20,78	13,08	5,72	3,65
7	3,33	10,72	42,22	63,66	53,23	48,26	22,79	13,61	4,53	3,73
8	5,35	9,71	42,95	64,55	54,29	49,22	22,22	13,73	6,05	5,49
9	5,40	11,52	41,89	61,21	54,42	48,21	21,39	11,77	3,80	3,69
10	3,71	9,70	42,12	58,54	56,37	48,10	21,25	13,38	4,01	3,74
Maior valor	5,475	11,521	42,961	64,548	61,534	49,224	22,785	13,73	6,054	5,808
Média	4,6506	10,6559	42,4478	60,5146	55,3345	47,7897	21,7256	12,8797	4,5276	4,2418
Mediana	5,065	10,632	42,306	60,0015	54,351	48,155	21,7445	13,0775	4,102	3,727
Desvio Padrão	0,820764	0,751769	0,38218	2,18136	2,61825	1,19172	0,59466	0,73492	0,77759	0,891716
Variância	0,673654	0,565157	0,14606	4,75833	6,85523	1,42021	0,35362	0,54012	0,60465	0,795158

Para aplicações BoT definiremos então, com base nos tempos de execução das Tarefas 3 e 7 (as duas menores variâncias conjuntas) e a média geral de tempo de execução das tarefas nos dois cenários, o valor do fator de *speed up* de um operário em um *grid peer-to-peer* (λ_{BoT}) como 0,48.

5.4 Análise

Para analisar o quão boa é o desempenho do TVGrid em comparação com a de um *grid peer-to-peer*, as seguintes relações serão consideradas: M_p'/M_p e M_{BoT}'/M_{BoT}.

5.4.1 Aplicações paramétricas

Primeiro será apresentada a análise acerca da relação M_p'/M_p, ou seja, a relação do *makespan* para execução de aplicações paramétricas em um *grid peer-to-peer* e o *makespan* para a execução do mesmo tipo de aplicações em um sistema TVGrid. Considerando uma rede de TV Digital padrão, é seguro definir que β é pelo menos 1 Mbps [Morris, 2005]. Em uma WAN (*Wide area network*), atualmente é incomum haver valores de β_{p2p} maiores que 1 Mbps [Hazlett, 2001]. Nivelando por baixo a relação M_p'/M_p, considera-se que $\beta=\beta_{p2p}$. Definindo $\theta_p = A.sz/(\beta \times A.pt)$ como o grau de adequação de uma aplicação paramétrica para execução em um *grid* – quanto maior for o valor de θ_p, mais adequada para execução em um *grid* é a aplicação. Assim sendo, e assumindo o caso mais interessante, de quando $s>N_{p2p}$ das equações (1) e (4), tem-se:

$$M_p'/M_p \geq \frac{\theta_p \times N_{p2p} + \frac{s}{N_{p2p}} \times \lambda_p}{\theta_p + 1 + \frac{s}{N \times \alpha}} \qquad (7)$$

Na Tabela 5 é apresentada a relação M_p'/M_p para vários cenários. Nesses cenários foi considerado um *grid peer-to-peer* com 10.000 *desktops* disponíveis para executar uma aplicação paramétrica com 100.000 tarefas. Foi considerado também um TVGrid com 80 milhões de receptores (cenário esperado para o Brasil quando da total adoção da TV Digital, como dito pelo Ministro das Comunicações brasileiro em 2006[26]). Conforme calculado na seção anterior, λ_p assumirá o valor 0,95. A Tabela 5 foi produzida através da variação da fração de receptores disponíveis (α) de 0,0125% (10.000 receptores) a 0,2% (160.000 receptores) e o valor de θ_p de 1 a 0,0001.

[26] "Agência Brasil – TV digital deve aumentar em 80 milhões o número de aparelhos no país" - http://www.agenciabrasil.gov.br/noticias/2006/07/06/materia.2006-07-06.4998754189/view

Tabela 5. Desempenho relativo do TVGrid e de um *grid peer-to-peer*, para aplicações paramétricas.

θp	1	0,1	0,01	0,001	0,0001	α
M_p'/M_p	4.762,0	476,6	9,9	1,0	1,0	**0,0125%**
M_p'/M_p	4.878,1	625,6	18,2	2,1	1,9	**0,0250%**
M_p'/M_p	4.938,3	741,4	31,2	4,0	3,8	**0,0500%**
M_p'/M_p	4.969,0	817,1	48,5	7,8	7,5	**0,1000%**
M_p'/M_p	4.984,5	861,0	67,0	14,5	15,0	**0,2000%**

Como pode ser observado na tabela acima, para aplicações paramétricas o TVGrid tem um desempenho consideravelmente superior à dos *grids peer-to-peer* na maioria dos cenários considerados. Quando θ_p possui um valor alto, o desempenho do TVGrid é extremamente superior que a de um *grid peer-to-peer*, mesmo com valores para α baixos.

5.4.2 Aplicações BoT

Agora a análise da relação M_{BoT}'/M_{BoT} será apresentada, considerando o mesmo cenário da análise anterior e assumindo que o tamanho da aplicação *trigger* não é maior que o de uma tarefa média. Define-se $\theta_{BoT} = t.sz/(\beta \times t.pt)$ como sendo o fator de adequação de uma aplicação BoT para execução em um *grid*, e das equações (2) e (5), tem-se:

$$M_{BoT}' \Big/ M_{BoT} \geq \frac{\theta_{BoT} \times s + \lambda_{BoT}}{\theta_{BoT} \times (s + c + 1) + 1} \quad (8)$$

Considerando uma aplicação BoT com 100.000 tarefas e $\lambda_{BoT} = 0{,}48$, e se variando a quantidade de tarefas disponibilizadas através do carrossel (c), podemos visualizar os valores estimados para M_{BoT}'/M_{BoT} na Tabela 6.

Tabela 6. Desempenho relativo do TVGrid e de um *grid peer-to-peer*, para aplicações BoT.

θp	1	0,1	0,01	0,001	0,0001	*c*
Mp'/Mp	0,9999	0,9998	0,9994	0,9947	0,9526	**10**
Mp'/Mp	0,9990	0,9989	0,9985	0,9939	0,9519	**100**
Mp'/Mp	0,9901	0,9900	0,9896	0,9851	0,9441	**1000**
Mp'/Mp	0,9091	0,9090	0,9087	0,9052	0,8733	**10000**
Mp'/Mp	0,7692	0,7692	0,7690	0,7670	0,7486	**30000**

Analisando os resultados obtidos, estabelece-se que o desempenho do TVGrid para execução de tarefas BoT é crescente com o grau de adequação das tarefas (θp), e

também depende da quantidade de *slots* disponíveis para essas tarefas no carrossel (*c*). Harmonizando-se esses parâmetros, o desempenho do TVGrid situa-se no mesmo patamar da dos *grids peer-to-peer*.

É importante ressaltar que essa análise foi feita considerando-se que a estação de TV Digital utiliza apenas um canal de transmissão – se a estação de TV tiver capacidade de transmitir a mesma aplicação BoT em mais de um canal simultaneamente, teremos um desempenho melhor que a de um *grid peer-to-peer*, na proporção direta da quantidade de canais usada simultaneamente para transmissão da mesma aplicação BoT (considerando que a probabilidade de um receptor de TV Digital estar sintonizado pode ser calculada para esses múltiplos canais).

6. Conclusão

Neste trabalho foi discutida a viabilidade de se utilizar a infra-estrutura de uma rede de TV Digital para realização de computação em *grid*. Para tal, após apresentação do estado da arte e trabalhos relacionados, foi proposta uma arquitetura de sistema que efetivamente realizasse a execução de aplicações paralelas em uma rede de TV digital.

6.1 Resultados e Contribuições

A arquitetura proposta foi formalmente modelada, testada e analisada. Até a conclusão deste trabalho nenhuma outra abordagem encontrada na literatura procurou aproveitar recursos ociosos de processamento de uma rede de TV Digital. A análise da proposta forneceu resultados através dos quais foi possível estabelecer-se o universo de aplicações para as quais a arquitetura do TVGrid possui um bom desempenho em comparação com uma proposta já estabelecida (*OurGrid*). Foram produzidos dados suficientes para estabelecer que, para aplicações definidas como paramétricas, a arquitetura do TVGrid viabiliza que uma rede de TV Digital possua um potencial de processamento substancial, de forma que a solução de problemas cuja solução se valha dessas técnicas possa ser acelerada, em comparação às abordagens encontradas na literatura.

Para aplicações do tipo *Bag-of-Tasks*, o potencial da arquitetura proposta neste trabalho também é inegável. A redundância para execução de tarefas simultâneas oferecida por redes de TV Digital, quando da transmissão de programas de alta audiência onde milhões de receptores estão sintonizados em um mesmo canal, também propicia uma carga de processamento cujo aproveitamento seria indiscutivelmente positivo para aplicações escalonadas utilizando-se a abordagem de saco de tarefas.

O escopo de problemas cuja solução pode ser agilmente encontrada através de seu processamento dentro de uma rede de TV Digital é suficientemente grande para justificar uma revisão acerca de seu desperdício. A atual corrente de digitalização das redes de TV pode vir a trazer significativos resultados em campos que necessitam da execução de tarefas cujo perfil foi definido durante este trabalho.

6.2 Trabalhos Futuros

6.2.1 Modelo de negócio

O trabalho proposto expõe todos os aspectos técnicos necessários para se conseguir explorar a capacidade de processamento ociosa dos receptores em uma rede de TV Digital, porém para que esta arquitetura seja posta em prática, faz-se necessário um modelo de negócio para a rede de TV onde a arquitetura será aplicada seja desenvolvido. No escopo deste trabalho esse aspecto não foi analisado por conta de características díspares encontradas nas redes de TV que adotam padrões digitais para transmissão do seu sinal.

Por exemplo, existem casos de redes de TV fechadas (a cabo, satélite ou microondas) onde as empresas difusoras são proprietárias dos aparelhos receptores – neste caso a adoção da arquitetura do TVGrid não implicaria em maiores complicações jurídicas visto que a estação transmissora estaria utilizando recursos de sua propriedade. Em todo caso, um estudo de impacto econômico deve ser base de um eventual modelo de negócio para a adoção da arquitetura aqui proposta, visto que a execução de aplicações resulta em um gasto maior de energia por parte do receptor de TV – de alguma forma o modelo deve absorver o ônus que cabe ao usuário.

6.2.2 *Middleware* preparado para processamento em *grid*

Um desdobramento natural desse trabalho pode ser também o acoplamento, junto ao *middleware* do receptor, de bibliotecas e elementos de software que propiciem um melhor conjunto de recursos para o desenvolvimento de aplicações para execução em *grid*. Algumas características encontradas em trabalhos como o *OurGrid* e também no *middleware* BOINC (apresentados na seção "3.") poderiam ser adicionadas a implementações de *middleware* como o Ginga, possibilitando o desenvolvimento de uma gama de aplicações ainda maior.

O escalonamento de tarefas também poderia se valer de diferentes estratégias, visto que cada receptor estaria apto a participar mais ativamente do processo de escalonamento. Tal recurso viabiliza o desenvolvimento de aplicações que utilizam estratégias como hierarquização (prioridade) e armazenamento para execução posterior.

A exploração de outros recursos também poderia ser viabilizada através da utilização de um componente interno do *middleware*, de forma que não só o

processamento ocioso fosse aproveitado, mas também recursos como o de armazenamento, por exemplo.

6.2.3 *Middleware* para aproveitamento de recursos de uma HAN

O *middleware* brasileiro Ginga oferece recursos para o desenvolvimento de aplicações cuja execução se dá em múltiplos dispositivos em uma HAN (*Home Area Network*) [Souza Filho, 2007]. É provável que no futuro o receptor de TV Digital tenha papel central na distribuição de mídias dentro das HANs, logo o potencial de processamento oferecido por redes de TV Digital poderá contar também com os recursos ociosos disponíveis em outros dispositivos conectados ao receptor de TV Digital.

Em tal cenário um receptor de TV Digital receberia um conjunto de tarefas e as distribuiria para outros dispositivos (como celulares, PDAs, computadores, etc) para que todos, em conjunto, processem uma quantidade maior de tarefas simultaneamente. O alcance da arquitetura aqui proposta aumentaria substancialmente com a adoção dessa estratégia, bem como a carga de processamento oferecida pela rede como um todo.

6.3 Considerações Finais

Algumas das características restritivas levadas em considerações quando da elaboração deste trabalho poderão vir a não existir em um futuro próximo. É provável que o dispositivo que no futuro será responsável por receber e decodificar sinal de TV Digital possua capacidade de processamento superior à disponível para processamento convencional.

A tendência é que as redes domésticas tenham dispositivos com abundância de recursos – com o tempo é natural que os documentos multimídia se tornem cada vez mais compostos (presença de mais canais de áudio ou de elementos que atuem em sentidos diferentes da visão e audição, por exemplo) e que carreguem uma variedade maior de informação. A recepção e a decodificação do sinal de TV muito provavelmente serão realizadas por um dispositivo em um ambiente muito mais propício ao desenvolvimento de aplicações paralelas.

Programas de TV de grande audiência podem dispor de milhões de receptores: na grande São Paulo, na semana de 28/03 a 03/04/2005, o IBOPE estimou que mais de

2,6 milhões de receptores estavam sintonizados no programa *Big Brother* Brasil[27]; eventos como o *SuperBowl* nos Estados Unidos atingem anualmente a ordem de 40 milhões de receptores simultaneamente[28]. Este trabalho expôs que, com a digitalização das redes, indicadores de audiência podem ser interpretados também como de capacidade de processamento disponível – a arquitetura apresentada neste trabalho propôs uma maneira de se aproveitar esta capacidade para executar um universo significativo de aplicações.

[27] http://www.almanaqueibope.com.br

[28] "TV Basics Top 50 Sports Telecasts of All Time" - http://www.tvb.org/rcentral/mediatrendstrack/tvbasics/17_Top50SportsTelecasts.asp

7. Referências Bibliográficas

[Amdahl, 1967] AMDAHL, G.M., ***Validity of single-processor approach to achieving large-scale computing capability***, In *AFIPS Conference*, Reston, p. 483-485. 1967.

[Anderson, 2004] ANDERSON, D.P. ***BOINC: A System for Public-Resource Computing and Storage***. In *Proceedings of the Fifth IEEE/ACM International Workshop on Grid Computing* (GRID'04), p. 4-10, 2004.

[Annavaram, 2005] ANNAVARAM, M., GROCHOWSKI, E., SHEN, J., ***Mitigating Amdahl's Law through EPI Throttling***. In *Proceedings Int'l Symp. Computer Architecture*, IEEE CS Press, p. 298-309, 2005.

[ARIB, 2004] *Association of Radio Industries and Businesses*. ***ARIB STD-B23 Version 1.1: Application Execution Engine Platform for Digital Broadcasting (English Translation)***. ARIB, 2004.

[ATSC, 2005] *Advanced Television Systems Committee*. ***ATSC Standard, Advanced Common Application Platform*** **(ACAP)**, ATSC, 2005.

[Batista, 2006] BATISTA, C.E.C.F., SOUZA FILHO, G. L. de, **Sistema para Execução de Aplicações Paralisadas utilizando uma Grade de Aparelhos Receptores de Redes de Televisão**. Patente de Modelo de Utilidade MMU8600875-7, 2006.

[Batista, 2007] BATISTA, C.E.C.F. et al., ***TVGrid: A Grid Architecture to use the idle resources on a Digital TV network***. *In Proceedings of the Seventh IEEE International Symposium on Cluster Computing and the Grid (The Latin America Grid Workshop)* – Rio de Janeiro, p. 823-828, 2007.

[Blauert, 2005] BLAUERT, J. ***The Evolution of Digital Audio Technology***. Springer Berlin Heidelberg. ISBN 978-3-540-22162-3. p. 299-319, 2005.

[CableLabs, 2005] *Cable Television Laboratories*. ***OpenCable Application Platform Specification – OCAP 1.0 Profile***. CableLabs, 2005.

[Cirne Filho, 2005] CIRNE FILHO, W. et al, **Grids Computacionais: Da Computação de Alto Desempenho a Serviços sob Demanda**. In: Mini-cursos do 23º Simpósio Brasileiro de Redes de Computadores – Porto Alegre – Francisco Vilar Brasileiro (Org.). Sociedade Brasileira de Computação, v. 23, p. 15-65, 2005.

[Cirne Filho, 2006] CIRNE FILHO, W. et al., ***Labs of the World, Unite!!!***. In *Journal of Grid Computing – Holland*, v. 4(3), p. 225-246, 2006.

[DAVIC, 1998] *Digital Audio Video Council*. ***Part 2 – DAVIC Specification Reference Models and Scenarios 1.4***, DAVIC. Disponível em <http://www.davic.org>, acesso em Novembro de 2007.

[DVB, 2003] *Digital Vídeo Broadcasting* (DVB), ***Digital video broadcasting (DVB) multimedia home platform (MHP)***. Padrão ETSI TS 102 812, ETSI, 2003.

[DVB, 2007] Digital Vídeo Broadcasting (DVB). ***Globally Executable MHP (GEM) Specification 1.1***. *DVB Document* A103, Rev. 1. Disponível em <http://www.mhp.org/mhp_technology/gem/a103r1.tm3567r1.GEM1.1.1.pdf>. Acesso em Outubro de 2007.

[ETSI, 2003] *European Telecommunications Standards Institute*. ***Digital Video Broadcasting (DVB), Implementation guidelines for Data Broadcasting***. *Technical Report* ETSI TR 101 202, ETSI, v. 1.2.1., 2003.

[Fox, 2002] FOX, B.. ***Digital TV Rollout*** IEEE Spectrum, v. 38, n. 2, p. 65-67, 2002.

[Freeman, 2004] FREEMAN, J., LESSITER, J. ***Using Attitude Based Segmentation to Better Understand Viewers' Usability Issues with Digital and Interactive TV***. In *Proceedings of the 1st European Conference on Interactive Television: from Viewers to Actors? – Brighton*. p.19-27, 2004

[HAVi, 2001] *Home Audio Video Interoperability*. ***Level 2 Graphical User-Interface:2006, Specification of the Home Audio/Video Interoperability (HAVi) Architecture***. HAVi, Inc. 2001. Disponível em <http://www.havi.org>, acesso em Maio de 2007.

[Hazlett, 2001] HAZLETT,T. W., ***The U.S. Digital TV Transition: Time to Toss the Negroponte Switch***. *AEI-Brookings Joint Center Working Paper No. 01-15 – 2001*, Disponível em <http://ssrn.com/abstract=292655>, acesso em Novembro de 2007.

[ISO/IEC, 1998] ISO/IEC. ***TR 13818-6 – Information technology — Generic coding of moving pictures and associated audio information — Part 6: Extensions for DSM-CC***, ISO/IEC, 1998.

[ISO/IEC, 1999] ISO/IEC. ***TR 13818-10 – Information technology — Generic coding of moving pictures and associated audio information — Part 10: Conformance extensions for Digital Storage Media Command and Control (DSM-CC)***, ISO/IEC, 1999.

[ISO/IEC, 2000] ISO/IEC. ***TR 13818-1 – Information technology — Generic coding of moving pictures and associated audio information: Part 1: Systems***, ISO/IEC, 2000.

[ITU, 2001] *International Telecommunication Union*. ***ITU Recommendation J.200 – Worldwide common core – Application environment for digital interactive television services***, ITU, 2001.

[ITU, 2003] *International Telecommunication Union*. ***ITU Recommendation J.202 – Harmonization of procedural content formats for interactive TV applications***, ITU, 2003.

[ITU, 2004] *International Telecommunication Union*. ***ITU Recommendation J.201 – Harmonization of declarative content format for interactive television applications***, ITU, 2004.

[Leite, 2005] LEITE, L. E. C., et al. **FlexTV – Uma Proposta de Arquitetura de Middleware para o Sistema Brasileiro de TV Digital**. In Revista de Engenharia de Computação e Sistemas Digitais, v. 2, p 29-50, 2005.

[Lin, 1998] LIN, J., ERBAR, M. YAN, W. **A digital high-performance multi-standard video data slicer.** In *IEEE Transactions on Consumer Electronics*. v. 44, n. 3, p. 1103-1106, 1998.

[Maes, 2001] MAES, J., VERCAMMEN, M. ***Digital Audio Technology*** (ISBN 978-0240516547). 4.ed., 360p, Focal Press, 2001.

[Maior, 2002] MAIOR, M. S., **TV interativa e seus caminhos**. Dissertação (Mestrado profissional em Engenharia de Computação), Instituto de Computação, Universidade Estadual de Campinas, 2002.

[Metropolis, 1949] METROPOLIS, N., ULAM, S. ***The Monte Carlo Method***. In *Journal of the American Statistical Association*, v. 44, n. 247, p. 335-341, 1949.

[Morris, 2005] MORRIS, S., SMITH-CHAIGNEAU, A. ***Interactive TV Standards – A Guide to MHP, OCAP and JavaTV*** (ISBN 978-0240806662). Elsevier, Focal Press, 2005.

[Peng, 2002] PENG, C. ***Digital Television Applications***. Tese de Doutorado (*Doctor of Science in Technology*), *Department of Computer Science and Engineering*, *Helsinki University of Technology* (ISBN: 951-22-6171-5). Finlândia, 2002.

[Soares, 2007] SOARES, L. F. G.; RODRIGUES, R. F.; MORENO, M. F. ***Ginga-NCL: the Declarative Environment of the Brazilian Digital TV System***. In: *Journal of the Brazilian Computer Society* (ISSN: 0104-6500). n. 4, v. 13. p.37-46, 2007.

[Souza Filho, 2007] SOUZA FILHO, G. L. de; LEITE, L. E. C.; BATISTA, C. E. C. F. ***Ginga-J: The Procedural Middleware for the Brazilian Digital TV System***. In: *Journal of the Brazilian Computer Society* (ISSN: 0104-6500). n. 4, v. 13. p.47-56, 2007.

[Tanenbaum, 2006] TANENBAUM, A. S., **Organização Estruturada de Computadores** (8576050676), v. 5, 464p, Prentice-Hall, 2006.

[Theys, 2001] THEYS, M. D., et al, ***What Are the Top Ten Most Influential Parallel and Distributed Processing Concepts of the Past Millenium?***. In *Journal of Parallel and Distributed Computing*, v. 61, n. 12, p. 1827-1841, Dezembro 2001.

[Wiegand, 2003] WIEGAND, T., et al. ***Overview of the H.264/AVC Video Coding Standard***. In *IEEE Transactions on Circuits and Systems for Video Technology*, v. 13, n. 7, 2003.

[Wood, 1995] WOOD, D. A., HILL, M. D. ***Cost-effective parallel computing***. In *IEEE Computer*, 28(2) p.69-72, 1995.

[Yamada, 2004] YAMADA, F. et al, **Sistema de TV Digital**. In Revista Mackenzie de Engenharia e Computação – n. 5. Editora Mackenzie, 2004.