OddCI-Ginga: Uma Plataforma para Computação de Alta Vazão baseada em Receptores de TV Digital

Diogo Henrique D. Bezerra[1], Rostand Costa[1, 2], Dênio Mariz Sousa[1]
Diénert A.Vieira[1], Guido Lemos de Souza Filho[1], Francisco Vilar Brasileiro[2]

[1]Digital Video Applications Lab – LAVID
Federal University of Paraíba (UFPB) - João Pessoa, PB – Brazil

[2] Distributed Systems Lab – LSD
Federal University of Campina Grande (UFCG) - Campina Grande, PB - Brazil

{rostand,diogohenrique,denio,dienert,guido}@lavid.ufpb.br,
fubica@dsc.ufcg.edu.br

Abstract. *OddCI is a new architecture for distributed computing that is both flexible and highly scalable. Previous works have demonstrated the theoretical feasibility of implementing the proposed architecture on a digital television system, but without taking into consideration any practical issues or details. This paper describes the construction of a proof of concept for the architecture and its implementation through modeling OddCI components over a testbed based on ISDB-T Digital TV receivers, called OddCI-Ginga. Performance tests using real broadcast transmission and the return channel demonstrate the feasibility of the model and its usefulness as a platform for efficient and scalable distributed computing.*

Resumo. *O OddCI é uma nova arquitetura para computação distribuída que é ao mesmo tempo flexível e altamente escalável. Trabalhos anteriores demonstraram a viabilidade teórica da implementação da arquitetura proposta sobre um sistema de televisão digital. Este trabalho descreve a construção de uma prova de conceito para a arquitetura através da modelagem e implementação dos componentes do OddCI sobre um testbed real baseado em receptores de TV Digital ISDB-T, denominado de OddCI-Ginga. Testes de desempenho usando transmissão real em broadcast e uso do canal de retorno demonstram a viabilidade do modelo e sua utilidade como uma plataforma de computação distribuída eficiente e escalável.*

1. Introdução

O cenário tecnológico atual é fortemente orientado para a convergência e marcado pelo surgimento de serviços e dispositivos que combinam tecnologias que surgiram inicialmente em contextos distintos. Celulares com capacidade de captura de imagens e vídeo, acesso às redes de telefonia, Internet e televisão, TVs conectadas à Internet, consoles de jogos conectados, receptores de TV digital, *tablets* – todos com processador, memória, sistema operacional e capacidade de executar aplicações – formam um vasto contingente distribuído de recursos computacionais.

Esta miríade de dispositivos digitais recentes ou tradicionais, computacionalmente capazes, virtualmente conectados e eventualmente ociosos, se devidamente coordenados e agrupados, podem representar um potencial de processamento sem precedentes para a construção de infraestruturas computacionais distribuídas de larga escala.

Neste trabalho consideramos um tipo particular de dispositivo com poder computacional relevante, os receptores de TV Digital, principalmente considerando que sua presença nas residências é uma tendência com a digitalização da televisão, a mais popular das mídias de massa. A TV Digital oferece recursos que vão desde a melhoria da qualidade da imagem à capacidade de interação com o conteúdo. Com essa nova modalidade de TV, o telespectador tem a possibilidade de exercer um papel mais ativo, interagindo com os programas de televisão, que além de áudio e vídeo, passam também a incorporar software de forma sincronizada [23]. Para tanto, o receptor de TV Digital conta com características típicas de um computador: possui memória, processador, sistema operacional e capacidade de se conectar em rede.

O grande alcance que a mídia televisiva apresenta com audiências que podem atingir bilhões de pessoas [7], a exemplo de transmissões de eventos globais como olimpíadas e copas do mundo, demonstra bem a escala associada com este segmento. Na Europa, onde a TV Digital aberta já se encontra disponível, milhões de receptores foram vendidos entre 2005 e 2007 [17]. A partir de dezembro de 2007, Sistema Brasileiro de TV Digital (SBTVD) [14] entrou em um processo de implantação paulatina, encontra-se em operação nas principais cidades brasileiras e foi adotado por praticamente toda a América do Sul. Adicionalmente, todos os países de grande população, incluindo China, EUA e Rússia já têm seus sistemas TVD em operação.

High Throughput Computing (HTC) [12][22] é um paradigma de computação que explora a idéia de usar muitos recursos computacionais, eventualmente distribuídos, para processar tarefas menores em paralelo, em oposição à idéia de processar uma tarefa grande em um recurso computacional único de maior capacidade. Há uma categoria de aplicações paralelas cujas tarefas são totalmente independentes umas das outras, definidas na literatura como *Embarrassingly Parallel Applications* ou *Bag-of-task Applications* (BoT) [9]. A vazão obtida quando se executa aplicações HTC em geral, e BoT em particular, sobre uma infraestrutura computacional distribuída depende diretamente da escala que a mesma permite. O tamanho do *pool* de processamento, definido como o número de processadores alocados, é o principal promotor de desempenho, enquanto que o esforço de coordenação envolvido é o principal fator de limitação. Para atingir uma vazão extremamente alta é necessário operar eficientemente em escala extremamente alta, assumindo que a distribuição de tarefas para os processadores disponíveis e o fornecimento de qualquer dado de entrada necessário ou coleta dos resultados gerados não sejam um gargalo.

O uso eficiente da infraestrutura por aplicações HTC com tarefas de curta duração (*short-lived*) requer a capacidade de instanciar um grande *pool* de recursos para uma aplicação em qualquer momento e somente pelo tempo que durar a execução da aplicação. Estes recursos podem ser depois designados novamente para aplicações diferentes. Além disso, para permitir a execução de um número ilimitado de aplicações de tipos diferentes, é essencial que a configuração da infraestrutura, inclusive a instalação de qualquer componente de software específico da aplicação, possa ser realizada de forma leve em termos de complexidade e ágil em termos de tempo. Tal premissa deve continuar válida até mesmo considerando-se que a escala desejada esteja na ordem de centenas de milhões de nós de processamento. Em outras palavras, o usuário deve ser capaz de facilmente e rapidamente personalizar a infraestrutura de processamento inteira de acordo com as suas necessidades.

Em um trabalho anterior, os autores apresentaram uma proposta de uma nova arquitetura para computação distribuída que é ao mesmo tempo flexível e escalável, chamada de *OddCI – On-Demand Distributed Computing Infrastructure* [11]. Trata-se de uma abordagem suportada pela existência potencial de milhões de dispositivos que, sob

demanda, podem ser acessados através de uma rede com suporte à transmissão em *broadcast*, através da qual, dispositivos podem ser alcançados simultaneamente.

Apesar de teoricamente viável sobre uma plataforma de TV Digital (TVD), a arquitetura OddCI ainda precisava ser posta em prática em uma plataforma de TV digital real, para aferição do seu desempenho e sua viabilidade prática. O objetivo deste trabalho é, portanto, aferir a viabilidade da arquitetura proposta, demonstrando como ela pode ser modelada e construída sobre um sistema TVD real. Para tanto, os componentes da arquitetura OddCI foram modelados, implementados e testados em receptores SBTVD-compatíveis. Nós chamamos esta implementação particular de OddCI-Ginga.

O restante do documento está organizado da seguinte forma. Na Seção 2 é descrita a arquitetura geral da plataforma OddCI para permitir o entendimento do restante do trabalho. Na Seção 3 é discutido o modelo de operação, a modelagem e implementação do OddCI-Ginga - uma instanciação OddCI sobre o Sistema de TV Digital Brasileiro (SBTVD). Na Seção 4 apresentamos uma avaliação do desempenho do OddCI-Ginga baseado em um *testbed* real, discutindo as métricas de interesse e, na Seção 5, discutimos os resultados obtidos. Na Seção 6 abordamos trabalhos relacionados e, finalmente, apresentamos as nossas conclusões finais na Seção 7.

2. OddCI – On-Demand Distributed Computing Infrastructure

A ideia de alocar uma enorme quantidade de recursos computacionais distribuídos e habilitá-los para o processamento distribuído de aplicações paralelas (centenas de milhares de computadores conectados via Internet, por exemplo), apesar de atrativa, representa um desafio não trivial. A questão principal é: *onde* encontrar uma grande quantidade de processadores disponíveis e *como* configurá-los para o uso em infraestruturas computacionais distribuídas (*distributed computing infrastructure*, DCI) dinâmicas voltadas para os requisitos de alta vazão de aplicações HTC? Além disso, como executar esta tarefa com um retardo mínimo?

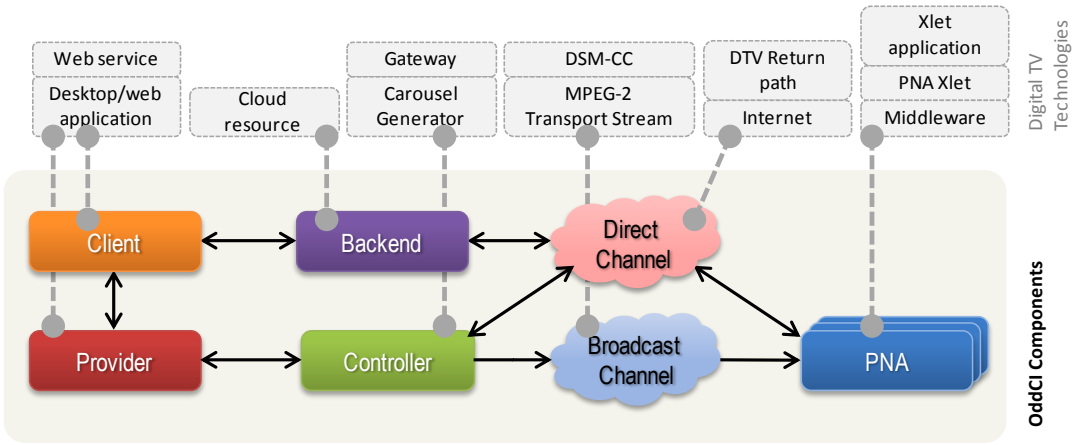

Figura 1 - Arquitetura OddCI, seus componentes e as tecnologias atuais disponíveis no sistema de TV Digital para sua implementação

Uma rede de *broadcast* tem o potencial de simultaneamente acessar todos os dispositivos a ela conectados, os quais podem ser coordenados para realizar uma determinada ação[1]. Neste trabalho considera-se uma rede de *broadcast* como uma rede

[1] Nas redes de difusão, como as de TV Digital, é possível usar mensagens de um-para-muitos efetivas, ou seja, que atingem múltiplos receptores com uma única transmissão. Isto permite a construção escalável de DCIs, posto que o custo de arregimentação de nós não é diretamente proporcional ao tamanho desejado para a infraestrutura.

composta por um transmissor digital de dados, um canal de *broadcast*, um conjunto de equipamentos receptores com capacidade de processamento de aplicações paralelas e a possibilidade de acesso a um canal de interação bidirecional, comumente uma conexão com a Internet.

Usando o conceito de redes de *broadcast*, Costa *et al.* [11] propuseram uma nova arquitetura para construir DCIs baseadas em recursos computacionais de alta granularidade, alta volatilidade e alta dispersão que é, ao mesmo tempo, flexível e altamente escalável, sendo aplicável para a execução eficiente de aplicações BoT de larga escala e curta duração. Com esta abordagem, um cliente que necessita de um grande número de unidades de processamento para executar uma aplicação BoT pode criar, sob demanda, uma instância de DCI grande o bastante para processar a sua aplicação de forma tão eficiente quanto possível. Após completar a execução, o cliente pode liberar a instância de DCI que foi criada. Por causa desta singularidade, a arquitetura é chamada de Infraestrutura Computacional Distribuída Sob Demanda (OddCI, do inglês *On-Demand Distributed Computing Infrastructure*) e a ativação de uma instância OddCI é chamada de *wakeup process* (WP).

A arquitetura OddCI é formada por um *Provider*, um *Backend*, uma ou mais redes de *broadcast*, cada uma contendo um *Controller*, um canal de *broadcast* e *Processing Node Agents* (PNA). Os PNA serão executados em cada um dos recursos computacionais acessíveis pelo *Controller* através da sua rede de *broadcast* correspondente. Além disso, é assumido que os recursos computacionais também são acessíveis através de um canal de comunicação individual, ponto-a-ponto e bidirecional, chamado canal direto ou de interação - o qual os conecta tanto com o *Backend* quanto com o seu respectivo *Controller* (Figura 1). Uma breve descrição[2] de cada um dos componentes previstos na arquitetura OddCI é apresentada na Tabela 1 (o mapeamento de tecnologias de TV digital mostrado na figura será discutido na seção 3).

Tabela 1 – Descrição dos componentes da arquitetura OddCI

Componente	Descrição
Provider	(provedor) é responsável por criar, gerenciar e destruir as instâncias OddCI de acordo com as solicitações dos clientes. O *Provider* é também responsável pela autenticação do cliente e pela verificação das suas credenciais para usar os recursos que estão sendo requisitados.
Controller	(controlador) é encarregado de configurar a infraestrutura, instruído pelo *Provider*. Formata e envia, via canal de *broadcast*, de mensagens de controle e imagens (executáveis) do PNA, necessárias para construir e manter as instâncias OddCI.
Backend	(retaguarda) é responsável pelo gerenciamento das atividades específicas de cada aplicação sendo executada, incluindo distribuição de tarefas (aplicações), o provisionamento de dados de entrada e a recepção e pós-processamento dos resultados gerados pela aplicação paralela.
Processing Node Agents - PNA	(agentes processadores) são responsáveis pelo gerenciamento da execução da aplicação do cliente no dispositivo computacional e comunicação com o *Controller* e *Backend*
Client	(cliente) é o usuário que deseja executar aplicações paralelas na infraestrutura OddCI. Ele deve fornecer as aplicações paralelas para o *Provider*, que as repassará para o *Backend*.
DirectChannel	O canal direto é uma rede de comunicação bidirecional que permite a comunicação entre todos os componentes da arquitetura, tal como a Internet.
BroadcastChannel	O canal de *broadcast* é um canal unidirecional para envio de dados do *Controller* para os dispositivos. Pode ser um canal de transmissão de TV Digital ou uma ERB de rede celular.

[2] Maiores detalhes sobre a arquitetura OddCI podem ser encontrados em [11].

Os dispositivos que executarão o PNA são descobertos e inicializados através da *wakeup message* (WM) transmitida pelo *Controller*. Esta mensagem contém, dentre outras coisas, o executável do PNA e a imagem da aplicação do cliente. Um PNA ativo envia regularmente sondas (*heartbeat messages*) para o *Controller* informando sobre o seu estado e indicando a instância que integra.

A próxima seção discute como uma implementação da arquitetura OddCI pode ser construída usando os recursos e tecnologias de uma rede de TV Digital.

3. OddCI-Ginga: Um Sistema OddCI sobre uma Rede de TV Digital

Para instanciar a arquitetura OddCI sobre uma rede de televisão digital, é necessário implementar os quatro componentes de software discutidos, isto é: o *Provider*, o *Controller*, o *Backend* e o PNA.

O papel do *Provider* é exercido por uma rede de TV que produz e transmite a programação nacional para diversas emissoras afiliadas. O *Controller* será um serviço exercido pela emissora ou repetidora local de TV digital que detém a concessão do canal de TV, através do qual enviará as mensagens de controle (dados) para os receptores sintonizados, junto com sua programação. O *Backend* pode ser montado como um conjunto de servidores sob controle do *Cliente* ou de um terceiro, possivelmente usando recursos distribuídos em nuvens. Cada PNA é uma aplicação que executará sobre o *middlewar*e de TVD, presente no receptor. O PNA usará a pilha TCP/IP e o canal de retorno (a Internet residencial) como canal direto para comunicação com o *Controller* e o *Backend*.

Na Figura 1 são identificadas as tecnologias atualmente disponíveis em um sistema de TV Digital que podem ser usadas e como elas estão associadas com os elementos da arquitetura OddCI genérica.

3.1. O Modelo de Operação do OddCI-Ginga

O OddCI-Ginga funciona da seguinte maneira. Inicialmente, o *Client* solicita ao *Provider* a criação de uma instância OddCI, fornecendo a aplicação, em um formato que permita que a mesma seja executada nos receptores de TV Digital. O *Provider* valida o *Client* e a imagem da aplicação e acata (ou não) o pedido baseando-se no histórico de audiência e em estimativas dos receptores conectados no momento. Em seguida, o *Controler* formata e encaminha uma mensagem de controle para ser transmitida pela emissora de TV, que inclui uma aplicação de PNA compatível com os receptores de TV Digital com a *flag* AUTOSTART acionada. A emissora, após validar o *Controller* e a mensagem de controle, usa o seu transmissor para enviá-la para todos os receptores sintonizados no seu canal.

O envio de dados usa o processo de distribuição e execução de aplicações interativas, conforme descrito no padrão Brasileiro de TV Digital e que ocorre da seguinte forma: inicialmente o conteúdo da imagem da aplicação é serializado na forma de um carrossel de objetos no padrão DSM-CC [21], onde os arquivos e pastas da aplicação são codificados em sessões e encapsulados em um fluxo *MPEG2 Transport Stream* (TS) [20]. Após a codificação dos dados, as propriedades da aplicação como nome, tipo, classe principal e outras características são definidas e estruturadas através da tabela AIT (*Application Information Table*) e encapsulados em pacotes TS. Terminada a preparação dos dados, ocorre a configuração da tabela PMT (*Program Map Table*) com o PID utilizado pelo TS de dados (*Object Carousel*) e o PID da AIT, além da adição dos descritores necessários para identificar a existência de um fluxo (*stream*) de dados para um determinado programa ou serviço. Por fim, o fluxo de dados é multiplexado com outros fluxos de áudio, vídeo e dados para então ser transmitido em *broadcast* pela emissora.

O receptor de TV Digital, ao receber a mensagem de controle, representada por uma aplicação com a *flag* AUTOSTART ligada, providencia a sua inicialização. Estando sintonizado na frequência da emissora, o receptor verifica a existência do fluxo de dados, e executa uma rotina de processamento desses dados, que é responsável por verificar a integridade do conteúdo recebido através do CRC de cada informação. Os dados são gravados obedecendo à estrutura de pastas e arquivos configurados na AIT. Ao término do processamento, o *middleware* é notificado da existência de uma nova aplicação passando informações sobre o nome, o tipo e o modo de execução da aplicação para o gerenciador de aplicações que seleciona o módulo de apresentação (*engine*) adequado ao tipo de aplicação: NCL/Lua [1] ou Java DTV [2], por exemplo.

A aplicação inicializada inicialmente é o PNA, que usa o canal de retorno do receptor para sinalizar ao *Controller* a sua disponibilidade para participar da instância e, caso seja aceito, carrega a aplicação do cliente propriamente dita. A partir deste ponto, a aplicação usa o canal de retorno para obter tarefas e enviar resultados para o *Backend* diretamente.

3.2. O Componente PNA - *Processing Node Agent*

Como o *Processing Node Agent* (PNA) é o componente da arquitetura OddCI que executa nos dispositivos finais (nós de processamento), precisou ser adaptado aos modelos de programação do *middleware* Ginga (Java e NCL) de forma a ser devidamente executado pelos receptores de TV Digital.

```
public class PNA {
    protected int state = PNAState.IDLE;
    protected String pna_id = "EMPTY";

    public void startXlet() throws XletStateChangeException  {
        while (true) {
            pna_id = control.sendHBI(pna_id, state);
            if (!pna_id.equals("EMPTY") && control.hasMessage()) {
                switch (state) {
                    case PNAState.IDLE:
                        if (control.get(PNAAttribute.MSGTYPE).equals("WAKEUP")) {
                            state = PNAState.BUSY;
                            vm = newVMThread(control.get(PNAAttribute.APPIMAGE));
                            vm.start();
                        }
                        break;
                    case PNAState.BUSY:
                        if (!vm.isAlive()) {
                            state = PNAState.IDLE;
                        } else {
                            if (control.get(PNAAttribute.MSGTYPE).equals("RESET")) {
                                if (vm.isAlive()) {
                                    vm.stopped = true;
                                }
                                if (finalize != null) {
                                    finalize.run();
                                }
                                state = PNAState.IDLE;  // the PNA is free again
                            }
                        }
                        break;
                }
            }
        }
    }
}
```

Figura 2 - Algoritmo Principal do PNA em Java DTV

De acordo com a arquitetura OddCI [11], um PNA ativo possui dois estados *Idle* e *Busy*. No estado *Idle*, o PNA não está integrando nenhuma instância OddCI mas fica monitorando o canal de *broadcast* permanentemente para o caso do *Controller* ter enviado alguma mensagem de controle do tipo WAKEUP convocando-o para integrar uma instância

nova ou para recompor uma instância em andamento. Neste momento, o PNA passa do estado *Idle* para o estado *Busy*, carrega e executa a imagem da aplicação recebida e guarda a identificação da instância que passou a integrar. Ele ficará neste estado até que um dos seguintes eventos ocorra: a) a aplicação finalize a sua execução; ou b) receba uma mensagem do tipo RESET do *Controller* com a identificação da sua instância. Neste momento, o PNA libera os recursos usados pela aplicação e retorna para o estado *Idle*, reiniciando o ciclo. Em ambos os estados, o PNA periodicamente avisa ao *Controller*, através de uma sonda, que continua ativo e a instância que integra.

Um trecho de código da versão do PNA em Java DTV que contém o seu algoritmo principal, aderente a transição de estados de um PNA é mostrado na Figura 2.

3.3. Os Componentes *Provider*, *Controller* e *Backend*

O *Controller* e o *Backend* também foram desenvolvidos de forma completa e plenamente funcional, com aderência a todos os eventos básicos descritos no diagrama de sequência da Figura 3. Isto permitiu uma observação de toda a dinâmica do sistema com o PNA interagindo com o *Controller* através da troca de mensagens de controle para criação e desmonte de instâncias, incluindo o envio da imagem da aplicação.

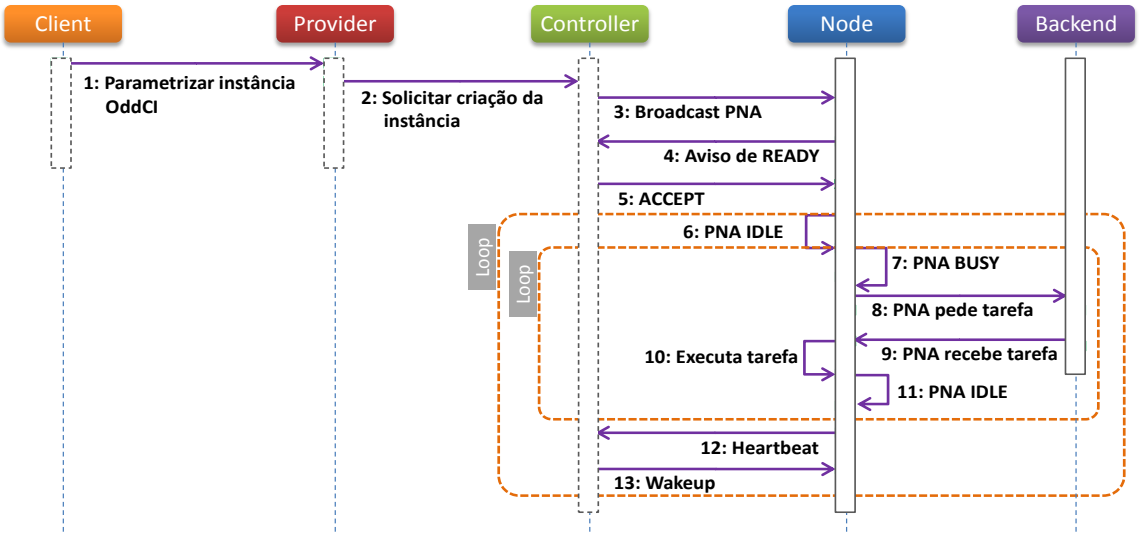

Figura 3 - Diagrama de Sequência do OddCI-Ginga

Para a validação do *Backend*, foi definida a aplicação paralela **Primos**[3] com dois módulos: o módulo cliente, desenvolvido em uma aplicação que executa no receptor de TV Digital, e um módulo servidor, que roda em um computador convencional. O objetivo da aplicação cliente é processar as tarefas que recebe do módulo servidor, que são caracterizadas por dois números representando um intervalo numérico discreto. O módulo cliente deve calcular todos os números primos existentes no intervalo e devolver o resultado para o módulo servidor. Neste ponto, solicita uma nova tarefa e o ciclo reinicia. O módulo servidor distribui tarefas para os PNA e recebe os resultados.

A aplicação **Primos** tem dois comportamentos possíveis: a) como aplicação BoT, no qual o módulo servidor distribui tarefas (intervalos de números) para os módulos clientes; e

[3] Em um trabalho anterior [11], nós já fizemos o porte bem sucedido de uma aplicação de bioinformática (NCBI BLAST) para um set-top-box. Embora a aplicação **Primos** também seja um exemplo real - fatoração de primos tem enorme uso na ciência em geral - ela é especialmente adequada ao objetivo do experimento: estressar a capacidade do receptor.

b) como aplicação paramétrica, na qual o próprio cliente seleciona o intervalo numérico a ser processado. Em ambos os casos, a carga de processamento do módulo cliente pode ser regulado pelo tamanho do intervalo numérico a ser investigado.

4. Avaliando o Desempenho do Sistema OddCI-Ginga

Com o objetivo de realizar um estudo preliminar do desempenho de um sistema OddCI-Ginga em receptores reais de TV Digital, foi construído um protótipo funcional que permitiu que todos os fluxos de comunicação fossem contemplados, como o fluxo entre o *Controller* e o PNA e a troca de informações entre a aplicação paralela e o seu respectivo *Backend*.

O ambiente montado para os testes envolvem um sistema completo de transmissão e recepção de TV Digital padrão SBTVD (TVD), disponível no LAVID/UFPB: gerador de carrossel, multiplexador, modulador, transmissor (de baixa potência para uso local) e alguns receptores TVD com uma versão do *Middleware* Ginga. Dois componentes receberam atenção especial: o *Controller* e o PNA. O *Controller*, porque é o *"gateway"* que interage com o sistema de TVD usando o gerador de carrossel para enviar dados e o PNA, porque é uma aplicação completa que roda no ambiente TVD, ou seja, no receptor, sobre o Ginga. Todo o ambiente utilizado é real, sem emuladores, de maneira que os resultados são parâmetros confiáveis para balizar o desempenho de uma plataforma em escala maior.

As subseções seguintes detalham quais as métricas que foram utilizadas na avaliação de desempenho, os experimentos planejados e executados bem como foi construído e configurado o ambiente usado nos testes.

4.1. Métricas de Desempenho

Para aferição da eficiência do sistema implementado foram consideradas três características específicas de um Sistema OddCI-Ginga: a) a velocidade do *Controller* para disparar comandos pelo canal de *broadcast*; b) a capacidade do canal de retorno para receber tarefas a serem processadas e transmitir os resultados obtidos; e, finalmente, c) o potencial dos receptores de TV Digital para o processamento de aplicações paralelas. Neste sentido, as seguintes métricas de desempenho foram observadas:

- **Tempo médio de preparação do PNA** (σ), que mede a agilidade do OddCI-Ginga para criar instâncias e considera o tempo envolvido na comunicação *Controller–PNA–Backend* para iniciar a execução da aplicação, calculado pela expressão:

  $$\sigma = w + d + r + a$$

  onde w é o tempo de preparação e transmissão da *wakeup message* (que contém a imagem executável do PNA) do *Controller* para o receptor usando o canal de *broadcast* (carrossel de dados), d é o tempo de processamento do carrossel de dados e carga da imagem do PNA no receptor, r é o tempo para envio da solicitação de dados do PNA para o *Backend* e a é o tempo para a resposta do *Backend* para o PNA.

- **Tempo Médio de Processamento** (δ) que mede o tempo médio de processamento de diversas tarefas de uma aplicação pelo receptor de TV Digital a partir do momento em que o PNA recebe a tarefa do *Backend* e inicia o seu processamento (P_1) até o momento da efetiva obtenção do resultado do processamento (P_2), imediatamente antes de iniciar a transmissão do resultado para o *Backend*. No caso da aplicação **Primos,** esta métrica é calculada dividindo o tamanho do intervalo numérico que constitui uma tarefa da aplicação ($\{I_I, I_F\}$) pelo tempo total de processamento:

$$\delta = \frac{I_F - I_I}{P_2 - P_1}$$

4.2. Planejamento dos Experimentos

Experimentos foram realizados sobre o ambiente de testes desenvolvido com o objetivo de avaliar o desempenho do Sistema OddCI-Ginga, os quais são descritos a seguir.

O primeiro experimento teve como objetivo medir o Tempo Médio de preparação do PNA (σ) com base em aplicações de diversos tamanhos. Neste sentido, foram formatadas oito *wakeup messages* com tamanhos de 119, 500, 1.000, 1500, 2.500, 3.500 e 7.500 Kb. Neste experimento, cada aplicação foi transmitida com exclusividade pelo carrossel de dados.

Foram ainda realizados dois outros experimentos adicionais para obtenção do Tempo Médio de Processamento (δ). O segundo foi a execução da aplicação **Primos** com intervalos limites de diversas magnitudes: 10^n, com n variando de 1 a 6. O terceiro experimento envolve uma aplicação que usa a pilha TCP/IP para buscar dados pelo canal de retorno. Foram realizados testes de acesso a páginas Web com 100, 500, 1.000, 1.500, 2.500, 3.500, 5.000, e 7.000 KB usando um acesso doméstico padrão de 1 Mbps.

Cada experimento foi realizado nas plataformas receptor TVD e PC de referência (vide Tabela 2). Cada experimento foi replicado tantas vezes quanto necessário para obter o número de amostras suficientes para definição de um intervalo de confiança com erro máximo de 5% e nível de confiança de 95%.

4.3. Configuração do Ambiente de Testes

Tabela 2 – Detalhes dos componentes do ambiente de testes do OddCI-Ginga

Componente	Descrição
Estação de TV	Modulador Linear ISMOD (ISDB-T Digital Modulator - Série ISCHIO) e Gerador de Carrossel e Multiplexador Linear/DommXstream (Instalado em um servidor Intel(R) Xeon(R) x3430 2.4 GHz com placa Dektec, Memória RAM de 3 GB, Placa de Rede Gigabit Ethernet, S.O. Ubuntu Server 32 bits - v. 10.04); Taxa máxima do carrossel de dados configurada para 1Mbps.
Receptores de TV Digital	Receptores (ou *set-top-boxes*) da marca Proview modelo XPS-1000 (firmware 1.6.70, middleware Ginga da RCASoft, com processador STMicroeletronics STi7001, Tri-core (audio, vídeo, dados) 266 MHz de clock, memória RAM de 256 MB DDR, memória flash de 32 MB, placa de rede Fast Ethernet (10/100) e Sistema Operacional adaptado do STLinux;
Processing Node Agent (PNA):	Versão A: em Ginga-NCL/Lua Script [19], imagem (executável) com 116,5 Kb.Versão B: em Ginga-J [18], imagem de 20,3 Kb.
Aplicação cliente	Aplicação "Primos", que implementa o algoritmo "crivo de Eratóstenes" para encontrar números primos até um valor limite [5]. Implementada em duas versões: Lua Script e Ginga-J, com tamanho do executável resultante em 2,6 Kb e 10,8 Kb, respectivamente;
Provider, *Controller* e *Backend*	*Provider*, *Controller* e *Backend* implementados como serviços de rede executando sobre o *middleware* Apache/Tomcatv6.0.33, protocolo HTTP para troca de mensagens, usando *framework* Web Grails/Groovy scripts, MySQL v.5.1 para o armazenamento de tarefas e resultados no *Backend*. No caso do *Provider*, foi criada uma interface Web para que clientes solicitem a criação de instâncias e a comunicação com o carrossel de dados. Estes componentes foram executados em um computador com processador Intel(R) Xeon(R) x3363 2.83 GHz, Memória RAM de 512 MB, Placa de Rede Gig*abit Eth*ernet e SO Ubuntu Server 32 bits v9.10.
Computador Pessoal de Referência	Para fins de comparação de desempenho com o receptor TVD foi usado um notebook com Processador Intel(R) Core(TM) i3-2310M 2.1 GHz, Memória 4 GB RAM, Placa de Rede Fast Ethernet e SOUbuntu 64 bits v11.10

Para a realização dos experimentos de avaliação de desempenho do Sistema OddCI-Ginga, um ambiente de testes (Tabela 2) foi montado com:

a) uma **estação de TV**, usada para as tarefas de geração do carrossel, multiplexação, modulação e transmissão das *wakeup messages* (WM) pelo *Controller*;
b) **receptores de TV digital**, para receber e processar as *wakeup messages* transmitidas pelo ar em *broadcast* pela estação de TV;
c) duas versões de **PNA** (uma em em Ginga-NCL/Lua Script [19] e outra em Ginga-J [18]) e refletem o comportamento descrito nas seções 3.1 e 3.2;
d) uma **aplicação cliente** em duas versões (Ginga-NCL/Lua e Ginga-J), que implementam o "crivo de Eratóstenes" [5] para encontrar números primos;
e) ***Provider***, ***Controller*** e ***Backend*** desenvolvidos como serviços de rede.

5. Análise dos Resultados

Nesta seção apresentamos os resultados das medições efetuadas nos experimentos realizados.

O primeiro teste envolve a avaliação do tempo de preparação do PNA. Os relógios do *Controller* e do *Backend* foram sincronizados usando o NTP (*Network Time Protocol*) e o processo de medição envolveu as seguintes etapas. O *Controller* envia uma *wakeup message* para receptor no tempo *T1*. Ao ser recebida pelo receptor, a mensagem é interpretada e o PNA é carregado e executado. No início da execução, o PNA marca o instante de tempo t_A e solicita dados ao *Backend* usando o canal de retorno. Ao receber a requisição, o *Backend* registra o instante de tempo t_B e envia t_B para o PNA na resposta da requisição. No recebimento da resposta, o PNA registra o instante de tempo t_C. Com os instantes t_A, t_B e t_C o PNA calcula o instante $T_2 = t_B - ((t_C - t_A)/2)$, baseando-se no algoritmo de Cristian [13], que assume o mesmo cenário de sincronização entre o *Controller* e o *Backend* já descrito. O tempo total da transmissão é então computado como $T = T_2 - T_1$.

O resultado das medições de tempo de preparação do PNA para vários tamanhos de imagens é mostrado na Figura 4. Observa-se que o tempo de preparação é linear em termos do tamanho da imagem, uma vez que o canal de *broadcast* tem banda constante de 1Mbit/s e, portanto tempo constante de transmissão. Pequenas variações aleatórias foram verificadas para os demais tempos envolvidos, que são explicados pela carga instantânea do *Controller*, do receptor, do *Backend* e do tráfego de retaguarda do canal de retorno. Esta análise permite concluir que o tempo de preparação pode ser estimado com segurança com grande dependência do tamanho da imagem e pouca dependência dos demais fatores envolvidos.

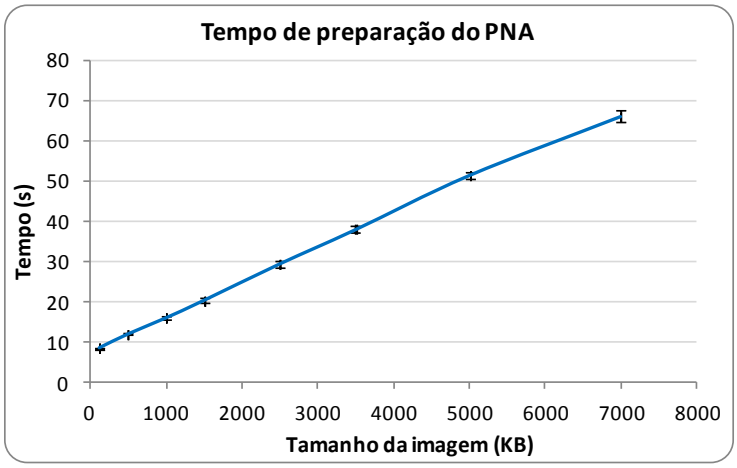

Figura 4 - Tempo de carga de uma imagem PNA

Para comparar a capacidade de processamento de um receptor com um computador pessoal de referência, o cliente da aplicação "**Primos**" foi executado em ambas as plataformas. O resultado apresentado na Figura 5 (escala logarítmica) demonstrou que o receptor é, em média, 27 vezes mais lento. Outra observação é que a aplicação no receptor estoura a memória quando tenta processar números acima de 10^6. Ou seja, é importante observar que aplicações para essa plataforma serão tão eficazes quanto menos dependentes do uso de memória.

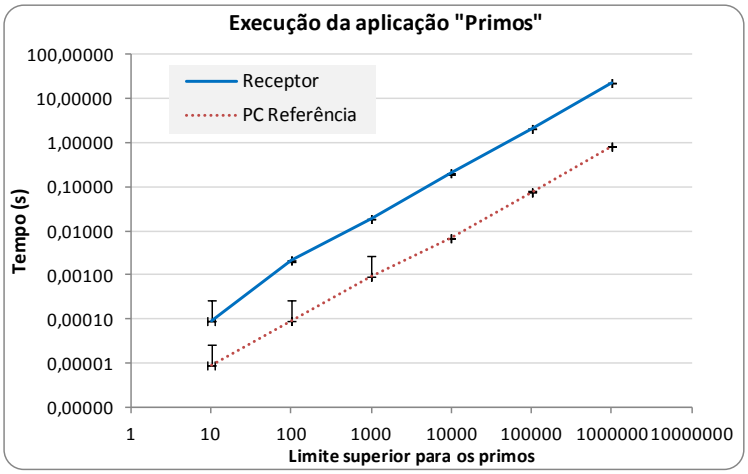

Figura 5 - Comparação do tempo de execução da aplicação "Primos"

O teste da eficiência do uso da pilha TCP/IP do receptor, comparado com o computador de referência é mostrado na Figura 6 (escala vertical logarítmica). O computador de referência acessou as diferentes páginas sem problemas, enquanto que a aplicação do receptor enfrentou problemas de memória com páginas a partir de 2.500kB. Assim, para comparação, calculamos o tempo projetado para páginas acima de 2.500kB no receptor, com uso de regressão linear. O tempo do receptor é em média 19 vezes maior do que o computador de referência. A diferença é menor do que no experimento anterior porque envolve o tempo de tráfego dos dados no enlace, constante para ambos.

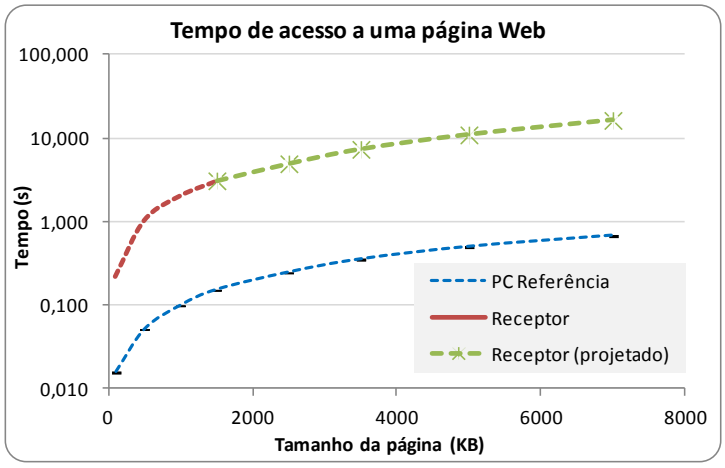

Figura 6 - Comparação do tempo de acesso a uma página da web

6. Trabalhos Relacionados

Considerando o uso de dispositivos não convencionais para a construção de infraestruturas para executar aplicações HTC, podemos destacar três sistemas. O sistema TVGrid [6], o projeto BOINCOID [8], e o projeto Folding@home [15][16].

O projeto BOINCOID foi criado em 2008 e também endereça o uso de dispositivos não convencionais para execução de aplicações HTC com foco em sistemas baseados no sistema operacional Android. O seu objetivo principal é o porte da plataforma BOINC [3] para o Android, através da tradução do código original em C++ para a linguagem Java, mantendo o comportamento original. Esta iniciativa habilita a participação de um enorme contingente de dispositivos baseados no Android em projetos de computação voluntária como o Seti@Home [4].

O Folding@home é um projeto de computação distribuída desenhado para realizar simulações moleculares para entender o dobramento de proteínas, más formações e doenças relacionadas. Iniciado em 2006, começou a usar o tempo ocioso de consoles de videogames conectados à Internet para obter um desempenho na escala de PetaFLOPs [15]. Essa experiência ratifica a tendência de usar dispositivos digitais emergentes e mostra a alta escalabilidade que tais dispositivos podem oferecer.

A proposta do TVGrid [6] tem por objetivo o aproveitamento, para computação em grade (do inglês, *grid computing*), de recursos que seriam desperdiçados em uma rede de TV Digital, como banda de transmissão do canal e capacidade de processamento do receptor de TV Digital. Ao adotar o conceito de escalonamento multi-nível, o OddCI-Ginga torna-se mais flexível que a abordagem TV Grid com relação à gama de aplicações suportadas. A separação do processo de provisionamento e controle de recursos, realizada pelo *Controller*, da distribuição de tarefas, realizada pelo *Backend*, permite que controles fim-a-fim específicos de cada aplicação, incluindo os relativos à segurança, possam ser implementados facilmente. Além disso, o OddCi-Ginga é mais transparente e requer uma menor participação da estação de TV na operacionalização de instâncias OddCI, o que pode se traduzir em maior facilidade para implantação.

7. Conclusões

O uso da capacidade ociosa de processamento de muitos recursos computacionais distribuídos, tais como os dos receptores de TV digital já havia sido demonstrada antes, na proposta do OddCI. A construção de uma prova de conceito com a implementação do sistema OddCI-Ginga sobre uma rede de TV Digital, a montagem de um *testbed* real e uma avaliação do seu desempenho mostraram não apenas a viabilidade dessa abordagem como também o fato de que é promissora.

Em particular, esta continuação da pesquisa buscou obter medições de campo sobre o potencial da TVD para sistemas OddCI. Assim, foi possível confirmar o comportamento linear na transmissão de mensagens de controle por radiodifusão, a adequação dos recursos de comunicação direta dos receptores para troca de tarefas/resultados e algumas das eventuais limitações de processamento dos dispositivos. Os resultados obtidos não se baseiam em simulações ou emulações, mas em um *testbed* real usando uma infraestrutura real, em escala menor, de TVD (mux, modulador, transmissor, receptor).

O canal de *broadcast* da TV Digital mostrou-se eficiente para os propósitos do OddCI-Ginga. Um canal SBTVD dispõe de uma banda total próxima de ~18 a ~21 Mbit/s, a depender de configuração [1][2]. A experiência mostra que emissoras podem dispor de uma banda residual de 1 a 4 Mbit/s para o carrossel de dados, descontada a vazão necessária para o fluxo de um vídeo FullHD codificado em H.264 e uma margem de segurança. Com 1Mbit/s, a carga usando um PNA e uma aplicação BoT típicos consome 10s. Com um PNA de ~7MB não leva mais do que 70s, tempo aceitável já que a carga do PNA é feita pelo receptor no primeiro recebimento e ignorado nos demais. Aplicações BoT paramétricas podem ter desempenho ainda melhor.

A avaliação da capacidade de processamento do receptor utilizado mostrou que ele é, em média, 27 vezes mais lento que um computador pessoal típico. Como os testes envolveram receptores de baixo custo, representando o pior caso, e a tendência observada é de melhoria da capacidade dos equipamentos, espera-se que esta relação possa ficar mais favorável. Entretanto, o fato do receptor ser mais lento não é necessariamente um problema, uma vez que a escala potencial da rede de TVD é na ordem de centenas de milhares ou milhões de vezes maior do que uma grade computacional tradicional.

A pesquisa obteve sucesso ao revelar a uniformidade da transmissão pelo ar e também as limitações do receptor, notadamente com relação à memória. Isso deve ser usado para definir o perfil das aplicações adequadas para instâncias OddCI. Como a filosofia das aplicações BoT é que elas podem ser muito pequenas, é perfeitamente viável encontrar aplicações cujos requisitos principais são de processamento. Há casos em que o uso de memória é pequeno e constante (que não aumenta a alocação com o tempo), a exemplo de aplicações que buscam padrões. Desta forma, ajustes na granularidade das tarefas da aplicação BoT podem permitir um adequado aproveitamento dessa infraestrutura.

Embora não tenha sido o foco deste trabalho, é sabido que alocar e manter ativos um *pool* de receptores em larga escala (milhares ou milhões) não é trivial. Primeiro, os dispositivos são sujeitos a falhas ou desconexões voluntárias e, portanto, são voláteis. Segundo, o canal de *broadcast* é eficiente, mas unidirecional, demandando o uso complementar, que deve ser otimizado, do canal direto para a operação de instâncias OddCI.

Neste sentido, o foco de outros trabalhos em andamento é a investigação de como os aspectos de imprevisibilidade e volatilidade envolvidos na coordenação e uso de recursos computacionais de redes de *broadcast*, adequadamente identificados e tratados, podem ser contornados com a aplicação de técnicas de predição e algoritmos compensatórios.

Referências

[1] ABNT. Televisão digital terrestre - Codificação de dados e especificações de transmissão para radiodifusão digital - Parte 2. NBR 15606-2¸ 2009.

[2] ABNT. Televisão digital terrestre – Codificação de dados e especificações de transmissão para radiodifusão digital - Parte 4. NBR 15606-4¸ 2009.

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

[4] Anderson, D. P.; Cobb, J.; Korpela, E.; Lebofsky, M.; Werthimer, D. (2002) *SETI@Home An Experiment in Public-Resource Computing*, Communications of the ACM Archive, vol. 45(11), pp. 56-61.

[5] ARM. (2011) *Sieve of Eratosthenes*. Disponível em: http://www.keil.com/benchmarks/sieve.asp. Acessado em 20 de Novembro de 2011.

[6] Batista, C. E. C. F.; de Araujo, T. M. U.; dos Anjos, D. O.; de Castro, M.; Brasileiro, F. & de Souza Filho., G. *TVGrid: A Grid Architecture to use the idle resources on a Digital TV network*. Proc. 7th IEEE International Symposium on Cluster Computing and the Grid (The Latin America Grid Workshop), 2007, 823-828

[7] Blog, B. O. Record 4.7 billion Television Viewers Watched Beijing Olympic Games 2008, 2008.

[8] Boincoid. *An Android Port of the Boinc Platform*, 2011. Acessado em 7 de Dezembro de 2011. Disponível em http://www.hatzlaha.co.il/150842/Boincoid

[9] Cirne, W.; Paranhos, D.; Costa, L.; Santos-Neto, E.; Brasileiro, F. E. A. (2003) *Running Bag-of-Tasks applications on computational grids: the MyGrid approach*. IEEE.

[10] Cooke, E.; Jahanian, F.; McPherson, D. (2009). *The Zombie Roundup: Understanding, Detecting, and Disrupting Botnets*. Proceedings of the 16th ACM conference on Computer and communications security.

[11] Costa, R.; Brasileiro, F.; Lemos, G; Mariz, D. (2009) *OddCI: On-Demand Distributed Computing Infrastructure*, Proceedings of the 2nd Workshop on Many-Task Computing on Grids and Supercomputers.

[12] Costa, R.; Brasileiro, F.; Lemos, G; Mariz, D. (2011) *Sobre a Amplitude da Elasticidade dos Provedores Atuais de Computação na Nuvem*, XXIX Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos.

[13] Cristian F. *Probabilistic clock syncronization*. Distributed Computing, v. 3, n. 3, p. 146-158, 1989.

[14] Eduardo, L.; Leite, C. & Rodrigues, R. F. *FlexTV Uma Proposta de Arquitetura de Middleware para o Sistema Brasileiro de TV Digital Revista de Engenharia de Computação e Sistemas Digitais*, Citeseer, 2005, 2, 29-4

[15] Folding@home. Petaflop Barrier Crossed, 2011.

[16] Folding@home. *PS3 FAQ*, 2011. Acessado em 7 de Dezembro de 2011. Disponível em http://folding.stanford.edu/English/FAQ-PS3.

[17] Freeman, J. &Lessiter, J. Masthof, J.; Griffiths, R. & Pemberton, L. (Eds.) *Using Attitude Based Segmentation to Better Understand Viewers' Usability Issues with Digital and Interactive TV* Proceedings of the 1st European Conference on Interactive Television: from Viewers to Actors?, 2003, 19-27

[18] Ginga. (2011) Ginga-J. Disponível em: http://www.ginga.org.br/. Acessado em 20 de Novembro de 2011.

[19] Ginga-NCL. (2011) Ginga-NCL. Disponível em: http://www.gingancl.org.br/. Acessado em 20 de Novembro de 2011.

[20] ISO/IEC. ISO/IEC 13818.2. MPEG Committee International Standard: *Generic Coding of Moving Pictures and Associated Audio Information: Video*. ISOMEG 1994

[21] ISO/IEC. ISO/IEC TR 13818.6. Information technology: Generic coding of moving pictures and associated audio information. Part 6: Extensions for DSM/CC 1998

[22] Litzkow, M.; Livny, M. &Mutka, M. Condor - *A Hunter of Idle Workstations* Proceedings of the 8th International Conference of Distributed Computing Systems, IEEE Comput. Soc. Press, 1988, 104-111

[23] Morris, S. & Chaigneau, A. S. Interactive TV Standards: A Guide to MHP, OCAP, and JavaTV Focal Press, 2005