UNIVERSIDADE FEDERAL DE SÃO CARLOS

CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

UMA ARQUITETURA PARA PROMOVER O USO DE DISPOSITIVOS COM LIMITAÇÕES COMPUTACIONAIS NA INTERAÇÃO COM MÍDIAS SINTETIZADAS REMOTAMENTE

ARTHUR PEDRO DE GODOY

ORIENTADOR: PROF. DR. CESAR AUGUSTO CAMILLO TEIXEIRA

São Carlos – SP
Maio/2014

UNIVERSIDADE FEDERAL DE SÃO CARLOS

CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

UMA ARQUITETURA PARA PROMOVER O USO DE DISPOSITIVOS COM LIMITAÇÕES COMPUTACIONAIS NA INTERAÇÃO COM MÍDIAS SINTETIZADAS REMOTAMENTE

ARTHUR PEDRO DE GODOY

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de São Carlos, como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação, área de concentração: Sistemas Distribuídos e Redes
Orientador: Dr. Cesar Augusto Camillo Teixeira

São Carlos - SP
Maio/2014

Ficha catalográfica elaborada pelo DePT da Biblioteca Comunitária da UFSCar

G589ap

Godoy, Arthur Pedro de.
Uma arquitetura para promover o uso de dispositivos com limitações computacionais na interação com mídias sintetizadas remotamente / Arthur Pedro de Godoy. -- São Carlos : UFSCar, 2014.
80 f.

Dissertação (Mestrado) -- Universidade Federal de São Carlos, 2014.

1. Computação gráfica. 2. Sistemas multimídia. 3. Plataforma móvel. 4. Sistemas distribuídos. I. Título.

CDD: 006.6 (20ª)

Universidade Federal de São Carlos

Centro de Ciências Exatas e de Tecnologia

Programa de Pós-Graduação em Ciência da Computação

"Viabilização de apresentações ricas em eletrônicos de consumo computacionais com mídias interativas sintetizadas remotamente"

Arthur Pedro de Godoy

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de São Carlos, como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação

Membros da Banca:

Prof. Dr. Cesar Augusto Camillo Teixeira
(Orientador - DC/UFSCar)

Profa. Dra. Regina Borges de Araújo
(DC/UFSCar)

Prof. Dr. Valter Roesler
(UFRGS)

São Carlos
Maio/2014

Dedico este trabalho a minha esposa Kamilla e aos meus familiares que sempre me incentivaram e apoiaram em cada momento, principalmente os difíceis.

AGRADECIMENTO

Agradeço primeiramente ao Professor Cesar, meu orientador, por ter se empenhado em prover todo suporte necessário à realização deste trabalho e por ter acreditado em mim desde o início das minhas pesquisas.

Também agradeço aos meus amigos do laboratório Lince e do Departamento de Computação pelo apoio, opiniões e sugestões tão valiosas.

Agradeço à minha esposa Kamilla por ter sempre me auxiliado e apoiado na concretização dos meus objetivos com carinho e compreensão.

Aos meus familiares, agradeço o suporte e incentivos presentes desde o início dos meus estudos.

"Querem que vos ensine o modo de chegar à ciência verdadeira?
Aquilo que se sabe, saber que se sabe; aquilo que não se sabe, saber que
não se sabe; na verdade é este o saber."

Confúcio.

RESUMO

As tecnologias de computação gráfica e realidade virtual permitem experiências visuais e auditivas com alto nível de realismo sensorial, como alcançado em jogos eletrônicos e no cinema. A sintetização de mídia com alto grau de realismo visual e auditivo demanda sistemas especializados com capacidade computacional de alto desempenho. A profusão na última década de eletrônicos de consumo computacionais pessoais, como dispositivos móveis e portáteis, desperta o interesse em tornar possível, também nesses dispositivos, aplicações visuais com alto nível de realismo. Devido às limitações da capacidade computacional intrínsecas dos dispositivos móveis, decorrentes de suas características físicas, como dimensões, restrições de consumo e dissipação térmica, não é possível a execução direta de mídia interativa com o mesmo grau de realismo encontrado em sistemas especializados. Este trabalho sugere a exploração de uma solução tradicional: o processamento e a interação remota com esse tipo de mídia. Desafios decorrentes dessa solução são identificados e estudados. Soluções são propostas, analisadas e formalizadas em uma arquitetura de referência. Como prova de conceito da arquitetura explorou-se o desenvolvimento de aplicações em dois cenários: um em rede local com mídia de baixa tolerância a atrasos no tempo de resposta às interações e outra através da Internet com mídia de maior tolerância a atrasos. Mostrou-se também a flexibilidade da arquitetura desenvolvida com a integração do componente cliente a um sistema com múltiplas mídias, em que a mídia remota relaciona-se integralmente com o contexto multimídia.

Palavras-chave: computação gráfica, eletrônicos de consumo, mídia de alto desempenho, sistemas multimídia, renderização remota, plataforma móvel, cloud gaming, desktop remota.

ABSTRACT

Computer graphics and virtual reality technologies allow audiovisual experiences with high level of realism, as the ones achieved in video games and movies. The synthetization of media with high degree of audiovisual realism demands specialized systems with high performance computing capacity. The profusion in the last decade of personal computing consumer electronics such as mobile and portable devices, encourages the interest in making possible, also in these devices, applications with high level of visual realism. However, due to the intrinsic limitations of the computing capability of mobile and portable devices, regarding to the physical characteristics of these devices, such as dimensions, electrical consume and heat dissipation, it's not possible directly process media with the same level of visual realism that is found in specialized systems. This research suggests the exploration of a traditional solution: the remote interaction with these applications. Challenges resulting from this approach are identified and studied. Solutions are proposed, analyzed and formalized in a architecture for reference. As a proof of concept the architecture is used in the development of applications on two scenarios: one in a local network with a media characterized by its low tolerance in delay of interaction response and another with a high tolerance media through the Internet. Also is showed the flexibility of the proposed architecture by integrating one of the applications developed with a multimedia context.

Keywords: graphics computing, consumer electronics, high performance media, multimedia systems, remote rendering, mobile platform, cloud gaming, remote desktop.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

4G – *Fourth Generation of mobile telecommunications*

4K – *3840x2160 pixels of resolution*

8K – 7680*x4320 pixels of resolution*

AAC – *Advanced Audio Coding*

API – *Application Programing Interface*

BYOD – Bring Your Own Device

CE – *Consumer Electronics*

dB – *Decibels*

FPS – Frames Per Second

FullHD – *1920x1080 pixels of resolution*

GDI – *Graphics Device Interface*

GFLOPS – Giga Float Operations per Second

GPGPU – General Purpose computing on a Graphics Processing Unit

GPS – *Global Positioning System*

GPU – Graphics Processing Unit

GUI – *Graphical User Interface*

HEVC – *High Efficiency Video Coding*

HLS – *Hypertext transfer protocol Live-Streaming*

HPC – *High Performance Computing*

HPM – *High Performance Media*

HTML – *HyperText Markup Language*

IPC – *Inter Process Communication*

ISP – *Internet Service Provider*

LAN – *Local Area Network*

Mbps – *Megabits per Second*

MPEG – *Moving Picture Experts Group*

NCL – *Nested Context Language*

PC – *Personal Computer*

PSNR – *Peak Signal to Noise Ratio*

RDP – *Remote Desktop Protocol*

RFB – *Remote FrameBuffer*

RTMP – *RealTime Message Protocol*

RTP – *Realtime Transfer Protocol*

RTSP – *RealTime Streaming Protocol*

SIMD – *Single Instruction, Multiple Data*

SSE – *Streaming SIMD Extensions*

SSIM – *Structural Similarity*

STB – *Set-Top Box*

TCP – *Transmission Control Protocol*

TVDi – *TV Digital Interativa*

UDP – *User Datagram Protocol*

VOD – *Video On Demand*

VPN – *Virtual Private Network*

WAN – *Wide Area Network*

WebRTC – *Web Real Time Communication*

WiFi – *Wireless Fidelity*

WLAN – *Wireless Local Area Network*

XML – *eXtensible Markup Language*

SUMÁRIO

Capítulo 1

INTRODUÇÃO

A principal questão investigada nesta pesquisa é como propiciar a utilização de mídias com alto nível de realismo audiovisual em dispositivos com limitações computacionais. Com essa questão, este trabalho se propõe a ilustrar as diversas vantagens e oportunidades que o uso de uma experiência com alto grau de realismo pode trazer para aplicações de consumo pessoal, entretenimento e serviços gerais, formalizando a proposta em uma arquitetura de referência. Essa arquitetura pode ser utilizada como diretrizes para implementações que buscam solucionar esse problema especificamente com uma abordagem remota, onde a aplicação que gera a mídia encontra-se em um contexto remoto ao dispositivo que a exibe, e esse dispositivo é responsável por encaminhar as interações do usuário para a aplicação.

1.1 Contexto

A capacidade de processamento gráfico de sistemas especializados torna possível experiências visuais e auditivas com alto grau de realismo. No aspecto visual, gráficos foto realistas tornam-se cada vez mais comuns em mídias pré-renderizadas, como animações e efeitos de computação gráfica em filmes e outros materiais audiovisuais, e também em mídias sintetizadas em tempo real, como jogos eletrônicos e efeitos visuais em transmissões ao vivo. Quando esse realismo visual e auditivo é aliado à interatividade permitida pela sintetização em tempo real, proporciona-se uma experiência imersiva e interessante para o usuário. Porém,

essas mídias demandam alto desempenho computacional e alta especialização em processamento gráfico do sistema.

Para melhor caracterizar esse tipo de mídias, utiliza-se neste trabalho o termo HPM (High Performance Media). Como definido em "*Multimedia presentation integrating interactive media produced in real time with high performance processing"* [1], HPM é uma mídia sintética interativa que necessita alto desempenho de processamento para ser produzida, que é consumida por seres humanos, e que promove a sensação de realismo, principalmente nos sentidos de visão e audição.

Outro fator importante para o contexto desta pesquisa é a disseminação, na última década, de eletrônicos de consumo computacionais como dispositivos móveis e portáteis, dentre eles os laptops, smartphones, tablets e até mesmo smart tvs. Esses dispositivos, referenciados neste trabalho apenas como CE (Consumer Electronics), permitem à computação pessoal ser utilizada de forma abrangente e até mesmo ubíqua. Assim, ao considerarmos o uso de aplicativos que se adequam ao cotidiano dos usuários, os dispositivos CE tornam-se indispensáveis. Essa profusão também desperta o interesse científico em pesquisar a viabilidade de promover aos usuários de CE a experiência encontrada com HPM. Entretanto, tais dispositivos possuem limitações computacionais intrínsecas, referentes as características físicas, como dimensões, consumo elétrico e dissipação térmica. Essas limitações inviabilizam a execução direta de HPMs em CE.

A busca de uma solução para transpor a barreira tecnológica de HPMs em CE é o aborda nesta pesquisa por meio da proposta de uma arquitetura independente de tecnologias e avaliada por meio de duas provas de conceito. Cada prova de conceito foi implementada com o objetivo de solucionar as necessidades de um contexto específico: um em rede local com mídia de baixa tolerância a atrasos no tempo de resposta às interações e outra por meio da Internet com uma mídia de maior tolerância a atrasos.

Figura 1. Representação do problema de pesquisa abordado neste trabalho. Onde A representa a aplicação remota que gera a mídia interativa a qual é exibida e controlada por B.

Na **Figura 1** é apresentada uma representação da questão abordada nesta pesquisa: a exibição e interação remota com mídias de alto realismo audiovisual. Nela são representadas a aplicação geradora da mídia remota em **A**, a qual é exibida e controlada pelos dispositivos com limitações computacionais representados em **B**.

1.2 Motivação

A motivação para este projeto pode ser dividida em três tópicos, detalhados a seguir:

○ Profusão do uso dos dispositivos CE computacionais: os recursos intuitivos de integração com o cotidiano de seus usuários, tais como, geolocalização, câmeras de alta resolução, acelerômetro, tela sensível ao toque, recursos multimídia, permitem a exploração de aspectos de usabilidade interessantes e promovem a criação de novas formas de interatividade;

○ Aplicações gráficas foto realistas: com as gerações de usuários atuais usando programas com gráficos cada vez mais foto realistas, cria-se a expectativa de que as aplicações tenham alta qualidade gráfica e realismo. O atendimento dessa necessidade torna-se um tema interessante para pesquisa, principalmente quando essas aplicações são focadas na interatividade com o usuário;

○ Viabilização de experiências inovadoras: permitindo a interação de HPMs por dispositivos CE possibilita-se a criação de aplicações com experiências novas para os usuários, aliando-se o alto grau de realismo com a facilidade de uso dos CE. Alguns exemplos de aplicações inovadoras são:

- *TVDi*: na área de TV Digital Interativa, pode-se imaginar uma aplicação em que o programa televisivo seja sincronizado a uma HPM, representando um ambiente virtual, ou até mesmo com interação com outros usuários. Por exemplo, um documentário em que o telespectador interage diretamente na TV com uma versão virtual do ambiente em que se passa o documentário. Isso lhe permitiria realizar um tour virtual contextualizado com o documentário; Outra aplicação interessante na área de TVDi, seria a interação de múltiplos usuários em um programa televisivo de auditório em que, por meio de uma HPM, os telespectadores poderiam interagir como se estivessem participando do auditório, como por exemplo em uma gincana.

- *Comunicação de projetos*: outra aplicação, poderia ser o desenvolvimento de uma solução para visualização, interação e colaboração de projetos entre equipes com conhecimentos técnicos heterogêneos. Essa aplicação poderia permitir um melhor entendimento de projetos imobiliários, automotivos, navais, etc. Por meio de um website, os clientes ou interessados poderiam interagir com a aplicação executando remotamente nos servidores.

- *Marketing e publicidade de produtos*: suprindo a demanda por interatividade em campanhas publicitárias e mesmo em *hotsites* de produtos que já utilizam alguma forma de realidade virtual, o uso de HPMs remotas traria um alto nível de realismo à essas aplicações. Esse uso poderia aumentar o envolvimento do usuário com o produto ou marca por meio da combinação de elementos como: fidelidade visual, foto realismo e imersão sensorial, viabilizando até mesmo o uso de renderização estereoscópica se o dispositivo CE possuir o recurso para apresentação.

- *Educação*: também é possível imaginar aplicações na área de educação, nas quais os alunos teriam a possibilidade de visualizar conteúdos muito mais interativos e reais diretamente em seus dispositivos. Essa visualização poderia facilitar o entendimento de conceitos abstratos, teóricos e/ou complexos.

As aplicações detalhadas acima são algumas das inovações que motivam esta pesquisa. Acredita-se também, que as contribuições deste trabalho possam ser utilizadas em aplicações já tradicionais, como:

- *Desktop Remota*: em uma intranet ou VPN corporativa, a solução desenvolvida poderia ser utilizada para garantir performance computacional para as aplicações com alta demanda sem necessidade de dispor um hardware especializado para cada usuário. Além disso, poderia viabilizar de forma facilitada o acesso a informação por meio da implantação do conceito *BYOD* (Bring Your Own Device), no qual os funcionários levam e utilizam seus próprios laptops e smartphones no ambiente de trabalho, sem comprometer o nível de segurança, pois as informações nunca saem da *cloud* privada da empresa, apenas o *streaming* de vídeo da tela da aplicação remota é transmitido de forma segura até os dispositivos clientes;
- *Cloud Gaming*: no setor de entretenimento poderia ser utilizada como uma solução de *Cloud Gaming*, conceito empregado gloabalmente em [2], na qual um usuário pode acessar e jogar instantaneamente por meio de uma conexão com Internet banda larga e um navegador ou aplicação cliente suportada pelo serviço. Dessa forma, todo o processamento ocorre nos servidores da nuvem e apenas a interação é coletada no software cliente no dispositivo CE. Esse tipo de aplicação poderia ser enriquecida com softwares clientes integrados em contexto multimídia e TVDi, como TVs digitais e programas televisivos;

1.3 Mídias de Alto Desempenho

Com os trabalhos desenvolvidos anteriormente [1], iniciamos a exploração de mídias sintéticas interativas que necessitam de alto desempenho de processamento para serem produzidas e são consumidas por seres humanos, promovendo a

sensação de realismo, principalmente nos sentidos visuais e auditivos. Essa mídia de alto desempenho é intitulada HPM (High-Performance Media), que se pode definir como uma mídia que inclui animações com alto grau de realismo e complexidade computacional, que têm informações visuais e auditivas manipuladas por meio da interação do usuário, requerendo processamento em tempo real e de alto desempenho.

Como essa característica de alto desempenho não possui uma delimitação imutável, por ser dependente das tecnologias e recursos disponíveis em determinada época (um conceito temporal), o mesmo se aplica à definição de HPM. Para definir tal delimitação, é utilizada a capacidade usual e atual de processamento em sistemas computacionais especializados em computação gráfica.

1.3.1 Características do processamento gráfico e hardware especializado

Para o processamento gráfico de HPM, assim como de qualquer aplicação 3D sintética em tempo real, é necessário o processamento especializado gráfico. Para essa demanda geralmente é utilizada uma GPU (Graphics Processing Unit).

A GPU tem como principal função realizar o processamento especializado na geração das imagens de saída visual (framebuffer) que são apresentadas ao usuário por meio da tela de um monitor ou outra forma de display (um projetor, por exemplo) [6]. Porém, devido a sua arquitetura paralela e sua capacidade intrínseca de processamento altamente escalável tem sido também utilizada para processamento de propósito geral (GPGPU – General Purpose Computing on GPU) [7].

No desenvolvimento de GPUs a tendência é a busca por uma melhor eficiência energética, comumente quantificada pela relação GFLOPS / W (Giga Float Operations per Second / Watt), a qual é fator decisivo em otimização de performance com o mesmo consumo energético. Essa diretriz de projeto ficou mais evidente em 2012, com o lançamento da arquitetura Kepler pela Nvidia, com essa nova arquitetura, as GPUs desenvolvidas obtiveram aumento de performance consumindo a mesma quantidade ou menos de energia quando comparadas a linha anterior dessa mesma empresa. Em outras palavras, a arquitetura Nvidia Kepler teve um aumento expressivo na relação GFLOPS / W. Em seguida, o mesmo foi realizado pelas principais empresas de GPU.

. Para a obtenção de melhores resultados no consumo energético sem diminuir a capacidade de desempenho computacional, as GPUs estão tendo suas dimensões e peso expandidos a cada nova geração. Isso se deve também ao aumento de transístores, pela tendência de dual e quad-gpus, onde em um mesmo invólucro físico colocam-se duas ou até mesmo quatro GPUs compartilhando a mesma estrutura energética e memória. Com isso a necessidade de maximizar a dissipação de calor é evidente.

Nos gráficos a seguir constam os dados de capacidade computacional (**Figura 2**), consumo energético (**Figura 3**) e a relação de eficiência energética (**Figura 4**), ao longo dos últimos anos (últimas 5 gerações de GPU Nvidia):

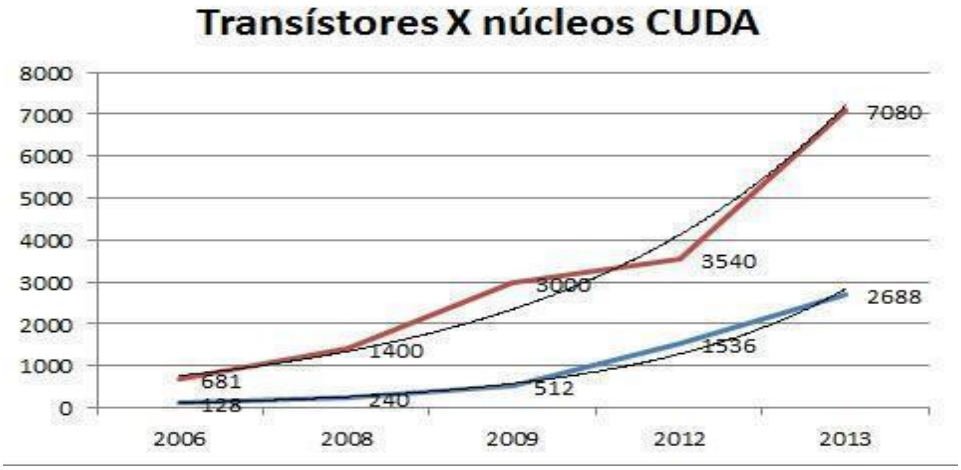

Figura 2. Avaliação da evolução na quantidade de núcleos de processamento e transístores em GPUs

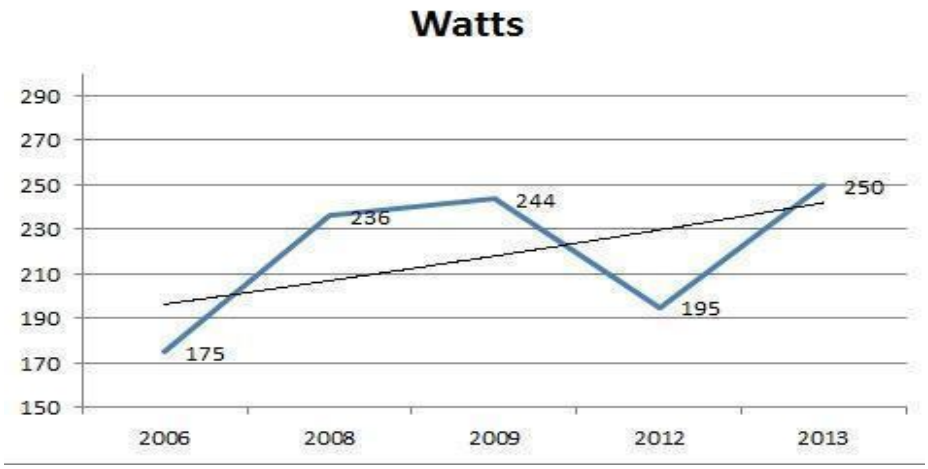

Figura 3. Avaliação da evolução do consumo energético em GPUs

Figura 4. Avaliação da evolução na eficiência energética das GPUs

Mesmo com um aumento de cerca de 30% no desempenho computacional entre as gerações de GPUs, a demanda computacional crescente dos softwares de animação 3D, em sua maioria, cresce ainda mais. A seguir, essa discrepância é apresentada na **Figura 5**, representando a quantidade de quadros (renderizações) por segundo (FPS - Frames Per Second), que as últimas gerações de GPU Nvidia obtiveram em alguns dos principais frameworks de desenvolvimento de jogos (game engines):

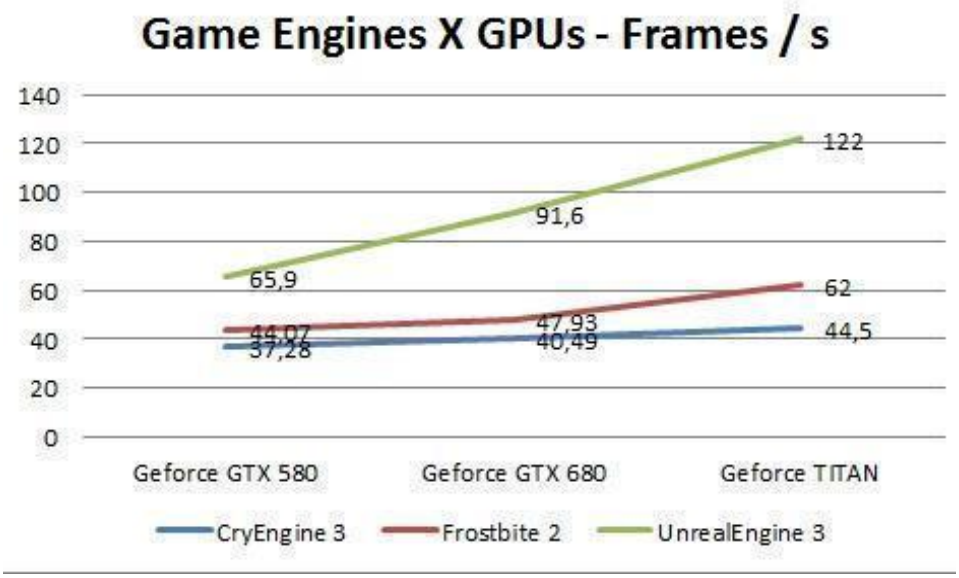

Figura 5. Avaliação da evolução na demanda de GPUs com capacidades computacionais superiores por três game engines utilizadas em grandes empresas de desenvolvimento de jogos.

Com as características físicas e de software expostas observa-se que a execução de HPMs é um processo altamente especializado, e que mesmo com a crescente eficiência das GPUs certos aspectos físicos continuam relevantes para o design desses sistemas computacionais especializados, como dissipação térmica e consumo elétrico, não encontrados em sistemas comuns ou não especializados em computação gráfica. Outro fator de impacto na demanda dos softwares por mais recursos computacionais, considerando outras tendências de display que são diretamente relacionadas a essa demanda de processamento, como monitores 3D estereoscópicos, os quais necessitam a geração de no mínimo duas vezes a mesma quantidade de quadros por segundo, ou os novos padrões de alta definição com resoluções de 4k e 8k, aumentando respectivamente em quatro e dezesseis vezes a quantidade de pixels renderizados (em relação à resolução FullHD), a restrição do uso de HPMs em sistemas especializados fica ainda mais evidente, mesmo em um futuro próximo.

1.4 Dispositivos com Limitações Computacionais

Entende-se por Eletrônicos de Consumo (Consumer Electronics - CE) dispositivos eletrônicos que são utilizados como equipamentos de uso diário, geralmente para comunicação entre pessoas, trabalhos de escritório e entretenimento. Esses são produzidos em larga escala e em sua maioria projetados para uso pessoal e/ou doméstico. Devido às grandes quantidades produzidas e à especialização dos processos de produção, uma característica comumente compartilhada entre os dispositivos CE é o preço, geralmente com uma tendência decrescente de uma determinada versão ou tipo de CE ao longo do tempo.

Podem ser considerados como exemplos de dispositivo CE, o receptor de radiodifusão, players de dvd e blu-ray, relógio digital, televisores, computadores pessoais (PC), celulares, laptops e mais recentemente smart TVs, smartphones e tablets. Para esta pesquisa considera-se a categoria de CE que possuem capacidade computacional, como os smartphones, tablets e notebooks.

1.4.1 Uso e repercussão atual

Os principais usos para os dispositivos CE são no entretenimento, na comunicação e no trabalho de escritório e comércio. Como exemplos do setor de entretenimento, pode-se citar consoles video-game e tocadores de vídeo, como blu-ray e dvd players. Na comunicação, são utilizados dispositivos CE como televisores, para difusão em massa, e celulares, para comunicação ponto-a-ponto. Porém, alguns CE tem seu uso não delimitado ou generalizado, como computadores pessoais, que inicialmente eram utilizados especificamente para os trabalhos de escritório, mas com sua disseminação nos anos 80, e com o surgimento dos laptops e o desenvolvimento acelerado da Internet nos anos 90, teve sua capacidade de uso estendida. Atualmente, os PCs auxiliam os mais diversos tipos de atividades, por exemplo, no entretenimento, comunicação, experimentos científicos, além do uso doméstico e profissional.

Essa capacidade de generalização foi aplicada em outros dispositivos CE, como celulares e televisores, por exemplos, na última década, foram desenvolvidos os smartphones e smart TVs. Também na área móvel disseminou-se um dispositivo CE que está conquistando tantos usuários quanto o PC, o tablet. Em 2012, segundo [8] foram vendidos aproximadamente 821 milhões de smartphones e tablets, cerca de 70% do total de dispositivos CE vendidos no mesmo ano.

Dispositivos CE costumam compartilhar certas características, como baixo consumo energético, tamanho e peso reduzido. Essas são características que originaram-se da forma de utilização pelos seus usuários, como por exemplo, a alta disponibilidade que originou a funcionalidade de standby, na qual o dispositivo desliga a maioria de seus componentes, mas continua ligado em modo de espera, aguardando o usuário utilizá-lo. Segundo [9], cerca de 75% da energia utilizada por dispositivos CE em uma residência é consumida no modo standby. Isso se deve pela necessidade do dispositivo estar sempre disponível. Outra funcionalidade originada por essa necessidade de uso e que também impacta na característica de baixo consumo energético é a autonomia de uso. Como exemplo, os smartphones ou os tablets têm a capacidade energética de sua bateria limitada devido às restrições do peso e do tamanho, o que influencia em sua capacidade computacional, inclusive gráfica, como demonstrado em "*An analysis of power consumption in a smartphone*"[10].

Com o advento dos smartphones e tablets, aliados à Internet móvel, vive-se o quinto ciclo tecnológico que foi precedido pelos ciclos: mainframe, mini computadores, computação pessoal e computação desktop com Internet. Esse ciclo, intitulado computação móvel, será de longe o de maior impacto e adoção na sociedade [11].

A computação móvel apresenta duas características principais: A primeira é a conectividade, na qual o dispositivo está conectado a uma rede móvel, como EDGE e 3G, ou WiFi, a maior parte do tempo. Já a segunda é o processamento embarcado, empregado para realizar as tarefas computacionais, como executar aplicações, utilizar as câmeras do dispositivo, controlar conexões com outros dispositivos e, usualmente, realizar várias dessas tarefas simultaneamente.

Atualmente estão sendo inseridas no mercado, porém ainda em desenvolvimento, as tecnologias de conexão móvel 4G (quarta geração), que podem alcançar 50 Mbps ou mais [12]. Com essas tecnologias será possível ter uma qualidade de conexão à Internet em um dispositivo móvel antes só alcançada em conexões cabeadas.

Aliados às tecnologias de conectividade, os processadores embarcados estão aumentando rapidamente sua eficiência e capacidade computacional. Entretanto, mesmo utilizando 4 ou mais núcleos de até 2,5GHz e GPUs integradas cada vez mais sofisticadas [13], a capacidade computacional dos dispositivos móveis mais modernos ainda é comparável aos sistemas desktop de vários anos atrás. Essa limitação se deve à relação direta entre a capacidade computacional com a eficiência energética, que no caso de dispositivos móveis, deve-se ao tamanho, peso e a capacidade de sua bateria.

A seguir são comparadas as características entre componentes de hardware de dispositivos CE, como laptops commodities e smartphones, com os componentes de um computador especializado em processamento gráfico de HPM:

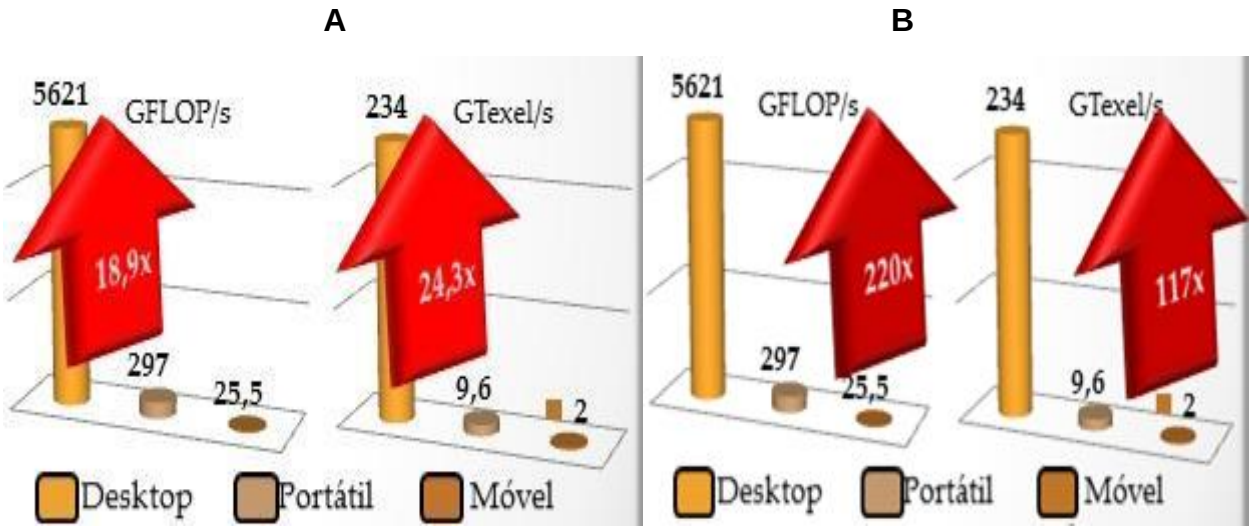

Figura 6. Comparativo entre o desempenho computacional das plataformas Desktop, Portátil e Móvel quanto a sua capacidade de operações de ponto flutuante (FLOPS) e mapeamento de texturas gráficas (Texels/S)

Na **Figura 6** é ilustrada a diferença de capacidade computacional gráfica entre as plataformas Desktop, Portátil (Laptop) e Móvel (Smartphone). Comparando-se os resultados entre exemplares mais atuais de cada plataforma nos seus requisitos de desempenho gráfico como operações de ponto flutuante por segundo (FLOPS) e operações de mapeamento de unidades de textura por segundo (Texels/S).

Ao analisar os resultados comparativos entre a plataforma Desktop e a Portátil (**Figura 6-A**) percebe-se que a diferença em capacidade computacional gráfica é grande. A plataforma Desktop tem um desempenho cerca de 20 vezes maior que a Portátil. Essa diferença torna-se cerca de 10 vezes maior quando comparada a plataforma Móvel com a Desktop (**Figura 6-B**).

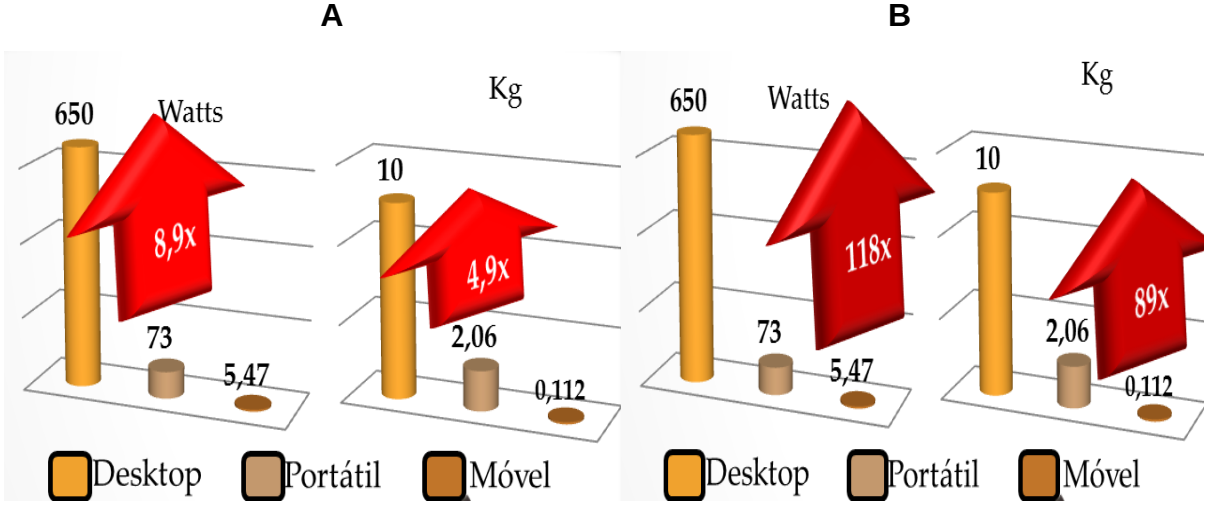

Figura 7. Comparativo entre as características físicas de C, Portátil (Laptop) e Móvel (Smartphone), e um Desktop especializado para processamento gráfico.

Já na **Figura 7** pode-se observar o motivo da diferença entre desempenho computacional gráfico das plataformas de dispositivos CE (portátil e móvel) e da desktop ser tão expressiva. Nela são apresentadas as características físicas dessas três plataformas, como a diferença entre potência elétrica (em watts) e massa (em quilogramas). Por meio do comparativo (**Figura 7-A**) é possível perceber que a plataforma desktop apresenta um consumo energético maior, cerca de 9 vezes a mais do que o laptop (plataforma portátil) e dispõe mais espaço e peso para dissipar o calor produzido (5 vezes). Na comparação com smartphone (plataforma móvel) essa diferença é ainda mais expressiva (**Figura 7-B**).

Portanto, a possibilidade de acessar por meio de um dispositivo CE os recursos gráficos, com fidelidade visual e auditiva, e alto nível de interação de uma HPM, não é diretamente possível, devido às características limitadas as suas dimensões e os requisitos computacionais necessários para o processamento de HPMs.

1.5 Objetivos

Este trabalho tem como objetivo geral estudar os problemas relacionados à viabilização da apresentação e interação de HPMs em dispositivos CE, buscando soluções adequadas a esses problemas e as formalizando em uma arquitetura para referência. O escopo dos objetivos deste projeto foi dividido em dois domínios de HPMs: (a) um de alta tolerância a latência no tempo de resposta, como passeios virtuais e visualização arquitetônica, e outro (b) de baixa tolerância no tempo de resposta, como jogos eletrônicos de ação. Assim, como objetivo principal desta pesquisa define-se:

- Demonstrar a viabilidade da utilização dessas aplicações remotamente em eletrônicos de consumo computacionais, dentro de contextos específicos para cada domínio: (a) utilizando uma conexão WAN (banda larga, 4G, etc.) entre os pontos (cliente e cloud); (b) e por meio de uma rede WLAN / LAN (WiFi, Ethernet, etc.).

Como objetivos específicos da pesquisa apresentam-se:

- Experimentar e analisar tecnologias e técnicas para solucionar os desafios e limitações tecnológicas da integração de HPM em CE;
- Projetar uma solução com arquitetura flexível que permita o desenvolvimento de aplicações desacopladas entre os artefatos de software cliente e as HPMs.
- Demonstrar a flexibilidade da arquitetura componentizada desenvolvida integrando o componente cliente dentro de um contexto multimídia.

Por meio de uma revisão sistemática, a qual possui questões de pesquisa definidas, focadas em um ponto específico da área de estudo, foram selecionados os trabalhos e pesquisas relacionados [3]. Com essa seleção foi possível conhecer os conceitos envolvidos e listar as tecnologias já testadas e aplicadas ao tema, tanto do ponto de vista acadêmico como de mercado. Além de prover o embasamento teórico para o desenvolvimento dessa pesquisa, esse levantamento permitiu a aquisição dos dados e parâmetros necessários para a realização dos experimentos e testes das tecnologias que foram abordadas.

Para cada teste das técnicas e tecnologias necessárias a composição da solução desenvolvida foram realizadas as quatro etapas do processo de experimentação: definição, planejamento, operação e interpretação [4]. Os experimentos foram compostos por meio de prototipação dos componentes isoladamente para cada operação analisada. Assim, simulando a entrada proveniente do componente anterior foram interpretados os dados obtidos quanto ao desempenho da operação e qualidade do resultado, baseando-se nas métricas que possuem uma maior relevância a percepção de qualidade da interação pelo usuário em sistemas de realidade virtual: latência na resposta da interação e qualidade visual da experiência.

Na avaliação da prova de conceito da solução desenvolvida, os experimentos foram realizados de forma geral e específica. A análise geral tinha por objetivo comparar a qualidade visual e responsividade da mesma HPM no cenário local e na prova de conceito remota. Assim, essa análise foi composta pela comparação de HPMs sendo utilizadas por meio da prova de conceito e das mesmas HPMs sendo utilizadas localmente. Utilizou-se o mesmo tipo de cliente, um laptop comum com GPU integrada (baixo consumo energético), para a realização de ambos os tipos de

experimentos. Também foram utilizados dois cenários de testes para a versão remota, um por meio de uma WAN (alta latência e baixa largura de banda) e outro em LAN (baixa latência e alta largura de banda).

Para as análises específicas da prova de conceito o objetivo da análise foi avaliar em cenários opostos o desempenho de cada componente da prova de conceito. Para isso, utilizou-se a solução implementada em dois cenários: (a) um em rede local sem fio (largura de banda até 54mbps e latência inferior a 5ms) com uma mídia possuindo baixa tolerância a atrasos no tempo de resposta das interações (representando um jogo eletrônico de rápida interatividade) e outro (b) por meio de WAN (largura de banda até 10mbps e latência até 50ms) com uma mídia que possui alta tolerância à latência de interação (representando uma mídia contemplativa, onde o usuário interage de forma menos contínua). Para a mídia “b” foi utilizada uma visualização arquitetônica (maquete virtual interativa) [5] e a mídia “a” utilizou-se um jogo de demonstração do framework de desenvolvimento de jogos UDK[1]. Como dispositivos clientes desses experimentos foram utilizados um laptop comum com GPU integrada de baixo consumo, um smartphone com o sistema operacional Google Android 4.0.3 e um tablet com o sistema Apple iOS 5.1.1.

1.6 Organização do texto

No Capítulo 2 são discutidos os desafios na relação HPM com CE, descrevendo os conceitos envolvidos na solução de interação remota e os trabalhos relacionados com suas correlações com o projeto desenvolvido. No Capítulo 3 são detalhadas as definições da arquitetura proposta bem como as soluções aplicadas na implementação das provas de conceito, e no Capítulo 4 os resultados obtidos com essa implementação, as limitações encontradas e suas análises. No Capítulo 5 as conclusões do trabalho desenvolvido são apresentadas e também são expostos os trabalhos futuros para a continuidade desta pesquisa.

[1] https://www.unrealengine.com/products/udk/

Capítulo 2

INTERAÇÃO REMOTA COM MÍDIAS DE ALTO DESEMPENHO

Uma solução clássica para execução de aplicações que possuem limitações de execução em determinados sistemas computacionais, tanto por características de hardware como software, é sua execução remota. Mesmo sendo um conceito simples de arquitetura cliente-servidor, as especificidades da implementação em um ambiente real de uso, em que múltiplas aplicações com requisitos diferentes de software devem ser suportadas e com dispositivos clientes heterogêneos, com plataformas e hardwares completamente diferentes, criam desafios tecnológicos interessantes para a pesquisa. Nas seções seguintes são apresentados esses desafios.

2.1 Renderização Remota em Tempo Real

2.1.1 Conceito e Aplicações

O conceito de renderização remota em tempo real se refere à necessidade de baixa latência entre o momento em que houve a alteração no estado da aplicação gráfica e o momento em que a renderização do novo estado é finalizada apresentando a resposta visual. Esse conceito já era aplicado em empresas e instituições de pesquisa na década de 80 com os *thin clients*, terminais computacionais muito simplificados apenas para visualização e interação remota com o computador central que realizava todas as operações [14]. Com a descentralização dos recursos computacionais das décadas seguintes, impulsionada em grande parte pela Internet, a renderização remota se concentrou no setor de pesquisa que utilizam clusters HPC (High Performance Computing) e necessitam de visualização em tempo real das simulações gráficas geradas [15].

Já nos últimos anos houve uma nova disseminação da aplicação dos conceitos de renderização remota no setor empresarial, com a utilização de acesso remoto a máquinas virtuais para *heavy users* (máquinas virtuais capazes de atender as necessidades de processamento HPC). Também no setor de entretenimento, com a utilização em plataformas de Cloud Gaming, esses conceito vêm sendo aplicados.

2.1.2 Tecnologias e Soluções

Devido à utilização e desenvolvimento inicial dos conceitos de renderização remota para aplicações de desktop remota, como os *thin clients*, as soluções para renderização remota estão muito relacionadas às soluções de desktop remota. Porém, essas tecnologias de renderização remota não se restringem às soluções de desktop remota, como por exemplo, as tecnologias utilizadas em cloud gaming.

Destacam-se as seguintes tecnologias e soluções para renderização remota:

- Microsoft RemoteFX:

Por meio de um driver genérico e dos recursos de virtualização de GPU, o gerenciador de virtualização Microsoft Hyper-V permite a aceleração gráfica 3D dentro de máquinas virtuais, desde que sejam utilizadas GPUs homologadas para o sistema;

- Microsoft RemoteApp:

  Utiliza o Microsoft RemoteFX para disponibilizar apenas uma aplicação remotamente, podendo ser uma aplicação com aceleração gráfica 3D;

- Citrix HDX 3D Pro Graphics:

  Utilizando GPUs homologadas para essa aplicação, permite a aceleração gráfica 3D dentro da máquina virtual. Entretanto não permite a virtualização de GPUs, o que limita a escalabilidade do sistema à quantidade de GPUs físicas, como explanado na **Figura 8**.

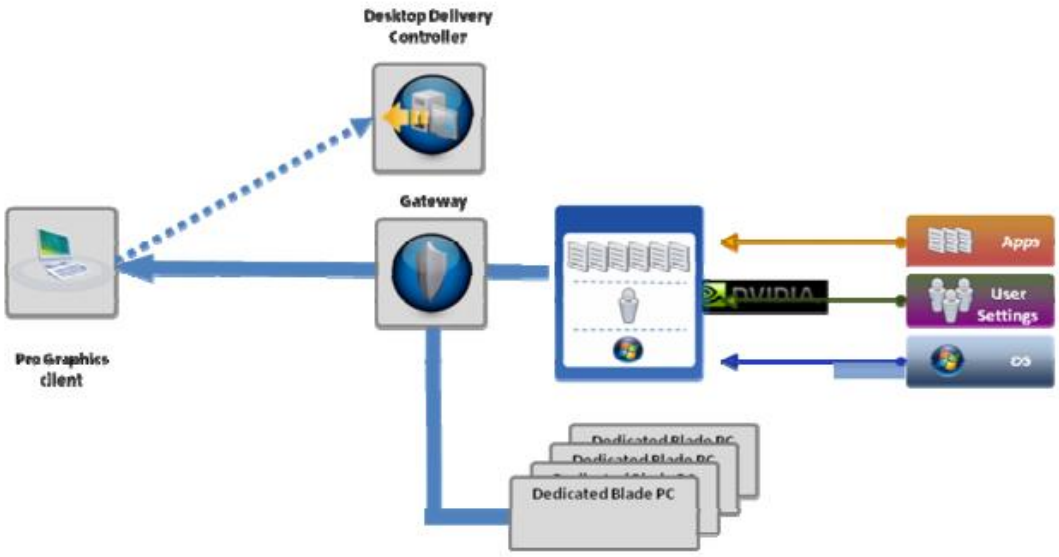

Figura 8. Fluxograma demonstrando o funcionamento do Citrix HDX 3D Pro [16], em que um gateway atua como uma camada de abstração entre o cluster (*blade servers*) que possuem GPUs, os quais executam os sistemas virtualizados, e as aplicações dos usuários.

- VMware Tools com aceleração 3D:

  Tecnologia que permite a aceleração gráfica 3D dentro de máquinas virtuais do gerenciador de virtualização VMware vSphere. Porém, assim como o Citrix HDX não permite a virtualização de GPUs;

- Nvidia GRID:

  Primeira e única solução de virtualização de GPU em hardware existente até o momento. Permite o compartilhamento de uma GPU física por múltiplos usuários por meio de GPUs virtuais. As camadas de

abstração do sistema de virtualização são apresentadas na **Figura 9.** Nela é possível observar as interfaces existentes desde a aplicação rodando na máquina virtual até o hypervisor da tecnologia GRID.

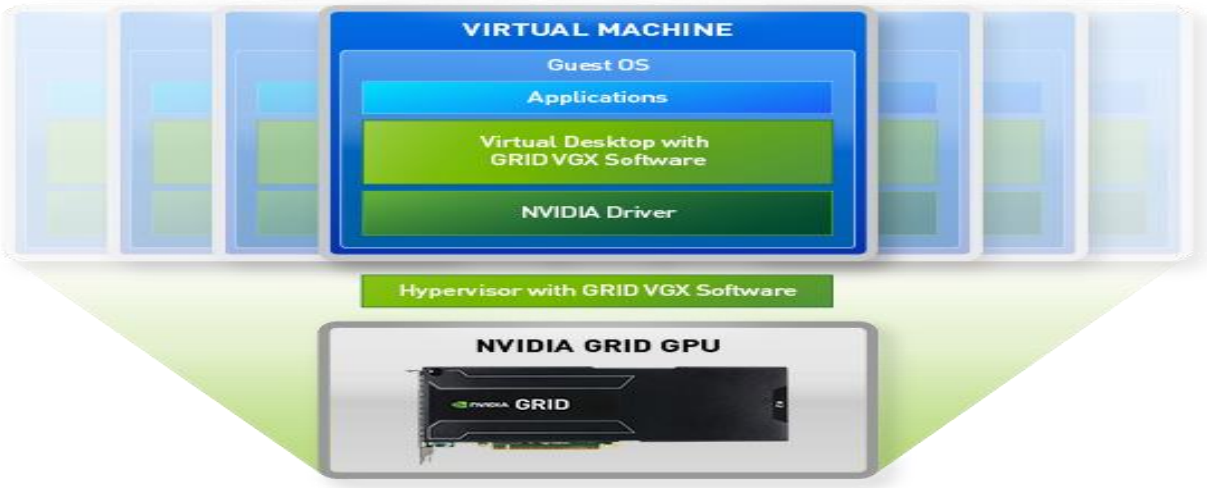

Figura 9. Diagrama demonstrando as camadas de abstração do funcionamento da Nvidia VGX em uma GPU Nvidia GRID [17].

2.1.3 Desafios Tecnológicos

No escopo de aplicações das técnicas de renderização remota destacam-se dois desafios tecnológicos principais, comuns a qualquer das tecnologias e soluções, que interferem na qualidade e usabilidade do sistema pelos usuários: a latência entre o tempo de interação e a resposta visual, e a quantidade de banda da conexão necessária para transferir a resposta visual ao usuário. Essas limitações tecnológicas são detalhadas a seguir:

- Latência:

  Definida pelo tempo entre o momento de interação e o momento de apresentação da renderização, é uma característica que interfere diretamente na usabilidade de uma renderização remota em tempo real.

  Um usuário utilizando uma aplicação com aceleração gráfica 3D de alto realismo (como um jogo moderno, por exemplo) sujeita-se a uma latência de aproximadamente 160ms [17]. Já em um outro estudo sobre jogos multiplayers online esse valor é no máximo 150ms [18]. Esse valor caracteriza a latência aceitável de interação e resposta visual, não havendo prejuízo para a usabilidade do sistema estudado.

- Largura de banda:

  É a taxa de bits por segundo necessária na conexão para transportar uma quantidade suficiente de dados visuais (quadros de um vídeo ou bits de uma imagem) para que não haja a sensação de travamento, ocasionada pela apresentação mais lenta do que a geração da mídia (fluxo gerado maior que a largura de banda disponível). Essa característica limita a capacidade mínima da conexão que usuário deve ter para obter uma qualidade satisfatória na visualização da renderização remota em tempo real (resolução idêntica à gerada e sem necessidade de um *buffer* para aguardar a transferência do fluxo de bits). As configurações e padrões de codificação, assim como o tempo disposto para o processamento (comparação interframes), interferem diretamente na compressão do vídeo e por consequência na largura de banda necessária para o transporte sem necessidade de utilização de um *buffer* para o pré-carregamento. Essa característica da conexão pode eventualmente impedir o usuário de utilizar a solução remota, caso não tenha a largura de banda suficiente.

2.2 Captura da Renderização Gráfica em Tempo Real

2.2.1 Contexto e Aplicações

São utilizadas técnicas de captura de renderizações gráficas em tempo real em diversos tipos de aplicações, desde captura de softwares de apresentações, principalmente quando utilizam animações gráficas, captura da interação do usuário com o sistema (para a confecção de um vídeo tutorial, por exemplo), captura da saída visual de vídeo games (para transmissão ao vivo de torneios na Internet), entre outras aplicações.

A forma de captura é definida por meio da tecnologia utilizada para renderização, quando a solução de captura é implementada apenas em software, ou

com a obtenção de componentes de hardware especializados quando a solução é independente das tecnologias de software utilizadas na renderização. As principais características de desempenho do processo de captura de renderizações gráficas em tempo real são: latência entre o momento em que foi finalizada a renderização e o momento do término da cópia (captura) da saída gráfica produzida, perceptível por meio da comparação entre a quantidade de renderizações (quadros) por segundo que são produzidos com a quantidade que é capturada; redução (impacto) no desempenho do sistema ou da aplicação que está sendo capturada, mensurável por meio da comparação entre a quantidade de quadros por segundo que o sistema é capaz de produzir sem a captura com a quantidade obtida enquanto a captura está sendo executada.

2.2.2 Soluções e Desafios Técnicos

Existem quatro principais técnicas de captura de renderizações em tempo real: captura por meio da API de gerenciamento gráfico de janelas, captura por meio da API de aceleração gráfica 2D/3D, captura por meio de *mirror drivers* e captura por meio de componentes de hardware. Essas técnicas e soluções são detalhadas a seguir:

- API de gerenciamento gráfico de janelas:
  Por meio da captura das operações de redesenho do gerenciadores de gráficos de janelas (2D), GDI e X11, respectivamente nos sistemas operacionais Microsoft Windows e sistemas baseados em GNU/Linux;
- API de aceleração gráfica 2D/3D:
  Realiza a captura do processo de renderização diretamente nas APIs de aceleração gráfica 3D como Microsoft DirectX e OpenGL;
- Mirror Drivers:
  Muito utilizados em clientes VNC (Virtual Networking Computing) para capturar a tela e interagir no sistema remotamente. Porém, como um *mirror driver* não é o driver padrão de saída visual (apenas o driver conectado ao monitor/GPU) ocorre uma latência entre a geração da renderização na saída visual padrão e a cópia para o *mirror driver*.
- Componentes de Hardware:

Utilizam uma solução de hardware para capturar fisicamente a saída visual como uma GPU faria ao enviá-la ao monitor. Depois aplica-se algum tipo de compressão e empacotamento para o envio pela rede. Essa é a solução que gera a menor latência, entretanto, em sua maioria, impõem ao usuário o uso de hardware especializado.

2.3 Pixel Streaming em Tempo Real

2.3.1 Conceito e considerações iniciais

A necessidade de acessar conteúdos audiovisuais diretamente no navegador web estimulou a criação de vários serviços de streaming de vídeo, também chamados de pixel streaming. De maneira simplificada esses serviços podem ser divididos em: vídeo sob demanda (Video on Demand - VOD) e streaming ao vivo (Live Streaming).

A principal diferença entre esses dois tipos de serviço de pixel streaming é que no VOD a mídia está pronta (já codificada e armazenada) no servidor, diferente do Live Streaming em que a mídia está sendo codificada e enviada ao mesmo tempo em que está sendo produzida. Como exemplos de aplicação de um serviço de VOD temos sites de compartilhamento e divulgação de vídeos ou serviços de acesso a filmes, seriados e documentários. Já para um Live Streaming pode-se citar o streaming de atividades esportivas e programação televisiva, como exemplos de pixel streaming em tempo real com baixa restrição de latência, já que alguns segundos de diferença do momento de produção da mídia podem ser tolerados. Entretanto, para um pixel streaming em tempo real com alta restrição de latência, como, por exemplo, o compartilhamento de tela e interação remota entre dois ou mais dispositivos, ou a renderização de aplicações interativas (jogos, simulações, aplicações de realidade virtual, etc.) remotamente, exigem no máximo alguns centésimos de segundo de diferença.

2.3.2 Limitações tecnológicas e Desafios

Três etapas importantes do streaming são: codificação, streaming e playback. A codificação se refere aos padrões utilizados para codificar áudio, vídeo e a multiplexação de ambos em uma mídia. Já o streaming é o momento em que a mídia é formatada e segmentada em pacotes, segundo um padrão de protocolo de transmissão de vídeo. Por fim, o playback consiste no recebimento e agrupamento dos pacotes, demultiplexação e decodificação do vídeo e áudio, e a apresentação do resultado ao usuário. Essa etapa exige uma compatibilidade direta com o protocolo de streaming e o padrão de codificação.

Em serviços VOD a codificação é realizada com antecedência à disponibilização dos vídeos. Usualmente se utiliza os *codecs* (padrões de codificação e decodificação de vídeo) H264 e VP8, sendo comumente multiplexados em MPEG4 e WebM, respectivamente.

Para realizar o streaming, os protocolos mais utilizados são o RTMP (Realtime Message Protocol) e o HLS (HTTP Live Streaming). Entretanto, este último se caracteriza como um protocolo de download progressivo, por enviar pacotes contendo pedaços do arquivo da mídia e não quadros do vídeo. Essa técnica de download progressivo também é utilizada para realizar a entrega das mídias multiplexadas em WebM.

O playback de VOD costuma ser *bufferizado*. O software do usuário armazena em um *buffer* alguns segundos do vídeo antes de apresentá-los. Essa técnica possibilita que usuários com restrições em sua conexão com a Internet, por exemplo, com baixa largura de banda ou inconsistência da conectividade, possam assistir o streaming de vídeo com pouco ou nenhum travamento. Tolerando-se alguns segundos de diferença entre a geração e visualização da mídia, essa técnica pode ser utilizada também para Live Streaming em que haja baixa restrição de latência.

De maneira geral, a etapa que mais demanda recursos, incluindo tempo de processamento computacional, é a codificação de vídeo. Nessa etapa evidencia-se a relação direta entre a compressão do vídeo e a latência entre o momento de geração da mídia original e a finalização da codificação. Com uma maior latência é possível obter uma melhor compressão, devido à quantidade maior de tempo para análise dos quadros e codificação. Portanto, um grande desafio em especificar uma

codificação para pixel streaming em tempo real com alta restrição de latência é gerenciar uma compressão boa suficiente para o perfil de conexão do usuário desejado com baixa latência e alta qualidade visual.

Outro desafio técnico é a compatibilidade entre playback e a codificação e o protocolo de streaming utilizados. Como existe uma dependência direta entre essas etapas, cria-se uma barreira em utilizar um mesmo conjunto de tecnologias para atingir o maior número possível de usuários, pois existem muitos dispositivos e tecnologias que apenas são compatíveis com determinadas combinações de padrões de codificação e streaming (por exemplo, o sistema operacional móvel iOS). Essa compatibilização com dispositivos sem tecnologias convergentes para playback de pixel streaming é uma barreira importante no projeto de um sistema flexível de pixel streaming em tempo real com alta restrição à latência.

2.4 Interação Remota

2.4.1 Contexto

Para realizar a interação em aplicações sendo executadas remotamente existem basicamente três etapas a serem cumpridas: captura local, comunicação e interação remota. Na captura local é necessário capturar os eventos de interação local desejados e realizar o mapeamento desses para os eventos que se deseja inserir na aplicação remota. Por exemplo, a partir de um dispositivo móvel, como um smartphone, seria possível mapear o toque na tela do dispositivo para um clique de um mouse ou pressionar de uma tecla na aplicação remota que suporta apenas interações tradicionais (mouse e teclado), permitindo um tipo de interação para o qual não foi projetada (*touchscreen*).

Já a etapa de comunicação refere-se aos processos de formatação do evento local em uma mensagem comum ao interpretador remoto, transmissão dessa mensagem por um canal seguro comum a ambos e recebimento pelo interpretador remoto. A generalização e flexibilidade do protocolo utilizado por ambos podem

definir a capacidade de inserir eventos específicos da aplicação remota. Um exemplo seria disparar a execução de uma função interna à aplicação.

Por fim, a etapa de interação remota é o processo de interpretar a mensagem de evento recebida e inserir o evento correspondente na aplicação. Essa inserção do evento pode ser feita por simulação de um evento comum ao sistema, como, por exemplo, a simulação de movimentação do mouse, ou por interação direta com a aplicação, quando essa já foi preparada para receber e gerar eventos diretamente.

2.4.2 Escalabilidade e acesso múltiplo

A principal característica para a escalabilidade de um sistema de interação remota é a capacidade de ter a mesma coleção de recursos computacionais (hardware e software) sendo acessada por múltiplos usuários simultâneos. Entretanto, devido às restrições tecnológicas na simulação correta de interações na pilha de eventos do sistema operacional, principalmente quando consideradas as decisões de prioridade na interação, ou eventos de interação personalizados por frameworks utilizados na confecção da aplicação desejada, tornam essa tarefa não trivial quando se busca uma solução com alto grau de compatibilidade.

Uma forma interessante de compartilhar recursos computacionais garantindo compatibilidade de interação remota satisfatória seria por meio da utilização de máquinas virtuais para executar as aplicações remotas. Com a virtualização dos recursos de hardware e isolamento do software entre usuários é possível promover a escalabilidade de aplicações remotas. Porém, HPMs e aplicações que utilizam aceleração gráfica 3D necessitam de acesso especializado à GPU. Mas, a virtualização de GPU é apenas oferecida por hardware altamente especializado, como explanado na **Seção 6.1.2**, impossibilitando a flexibilidade de personalização nos protocolos de comunicação e o desacoplamento entre a HPM e o software local do usuário.

Outra forma satisfatória e mais flexível de alcançar o acesso múltiplo de usuários seria pela emulação de múltiplos dispositivos de entrada no sistema operacional, por exemplo, múltiplos mouses e teclados. Com tal capacidade seria possível compartilhar os recursos de hardware, bem como os recursos de software, evitando armazenamento duplicado de arquivos e recursos das HPMs.

2.5 Estado da Arte em Interação Remota

Por meio dos desafios detalhados neste capítulo é possível observar a importância dos estudos necessários desenvolvidos na pesquisa buscando solucionar ou mitigar esses problemas. Cada desafio técnico é um obstáculo crítico para o sucesso da solução, com potencialidade de impacto direto na usabilidade do sistema, interferindo principalmente na qualidade e interatividade perceptível pelos usuários. Tal impacto pode comprometer a principal característica buscada nesta pesquisa: manter a fidelidade visual e de interação das HPMs remotas em CE.

Devido a essa importância, no Capítulo 6 são propostas soluções para esses desafios com o objetivo de viabilizar o acesso de usuários de dispositivos CE à HPMs.

O conceito de interação remota com HPM é abordado em Viel et al [19], sendo apresentado um framework que utiliza uma estratégia baseada em streaming de vídeo. Dessa forma, torna-se possível que aplicações que requeiram alto grau de processamento, como é o caso da renderização de HPMs, possam ser acessadas em dispositivos com restrições de hardware.

Nesse trabalho, utilizaram-se aplicações HPM que foram alteradas ou desenvolvidas especialmente para os testes. Tais alterações foram necessárias para o funcionamento do framework proposto, devido ao acoplamento entre a forma de captura e interação utilizados, impossibilitando que aplicações sem essas implementações pudessem usufruir do framework. Além disso, foi desenvolvido com um software cliente específico em foco, não podendo ser acessadas de outra maneira.

Na pesquisa de Jurgelionis et al [20] é demonstrada uma forma de captura capaz de acessar a saída visual de aplicações 3D, sem a necessidade de alterá-las. Também é exposto que a partir da inserção de bibliotecas dinâmicas (DLL) no processo da aplicação é possível capturar as chamadas e inserir novas chamadas à API 3D (no caso MS DirectX). Entretanto os procedimentos de envio dos resultados visuais e interação são específicos a essa API 3D, não havendo possibilidade de captura de uma aplicação OpenGL, por exemplo.

O trabalho de Karachristos et al [21] apresenta um serviço de GoD (Game on Demand), em que os jogos são acessados sob demanda, assim como vídeos em um

serviço de VoD (Video on Demand). O sistema proposto também utiliza técnicas de inserção de DLL para a captura da saída de vídeo da aplicação desejada e codificação de vídeo para realizar o streaming até o usuário, em uma arquitetura cliente-servidor. Porém, o trabalho restringe-se a uma aplicação específica para aceitar o tipo de streaming utilizado e enviar as interações remotas, funcionando apenas em clientes Microsoft Windows.

Em Yu et al [22] é proposto um sistema para interação remota com aplicações baseado em dispositivos móveis como thin clients. O sistema utiliza uma codificação de vídeo para realizar o streaming da tela ou parte da tela (janelas), por meio do protocolo RFB (Remote Framebuffer) utilizado para transmitir remotamente o framebuffer do dispositivo gerenciador de interface gráfica (Graphics Device Interface - GDI) do sistema operacional Microsoft Windows. Para a captura da saída visual é realizada a cópia do buffer da janela selecionada por meio do GDI. Porém, esse método de captura não possibilita o acesso eficiente às telas de aplicações gráficas 3D que utilizam a API Microsoft DirectX.

No estudo de Zhao et al [23] é explorada uma arquitetura de Cloud Gaming. Servidores especificamente configurados para virtualização com suporte a GPU atendem às interações remotas dos clientes em três cenários testados: tablet, PC e smartphone. Para a comunicação, tanto do streaming de vídeo quanto das mensagens de interação e controle, foi utilizado um protocolo genérico de comunicação (UDP). Porém, para utilização da solução é necessário que a aplicação modificada seja desenvolvida especialmente para o sistema, como foi realizado nos experimentos do trabalho.

Torterolo et al [24] apresentam um sistema de cloud computing 3D com o objetivo de acessar aplicações gráficas 3D, da área de ciências biológicas, de forma remota e colaborativa. No sistema proposto é utilizada também uma técnica de inserção de bibliotecas dinâmicas no processo da aplicação. Já a visualização e interação são realizadas por meio de um streaming de vídeo e um plugin proprietário, o qual necessita ser instalado no navegador web para seu funcionamento. Entretanto, o ambiente de execução e captura é restrito ao hypervisor de virtualização utilizado, funcionando apenas com aplicações que utilizam OpenGL, API de renderização comum em sistemas científicos de representação gráfica.

Em Huang et al [25], foi proposto um sistema de cloud gaming open source, provendo um framework com as ferramentas necessárias para o setup de servidores do portal de seleção de jogos, servidores de execução dos jogos e a aplicação cliente. Entretanto, nesse sistema há limitações impostas pelos protocolos de streaming de vídeo escolhidos, tipo de interação remota e aplicação cliente com dependências de plataforma, inviabilizando a integração de clientes multiplataforma ou mesmo HTML5.

Na **Tabela 1** é apresentado um panorama acadêmico e tecnológico da última década sobre o conceito de interação remota com HPMs:

Tabela 1 – Panorama tecnológico ( ) e acadêmico ( ) com acontecimentos e trabalhos relevantes ao escopo desta pesquisa na última década. Os tópicos são separados entre publicações científicas e tecnologias.

Ano	Tipo	Acontecimento
2008		Karachristos et al [21] propõem um serviço em que jogos são acessados sob demanda, analogamente a vídeos em um serviço de VoD (Video on Demand)
2008		Amino, uma empresa de STB, lança uma parceria com a G-cluster, pioneira em Cloud Gaming desde 2004, para estudar um serviço integrado aos seus set-top boxes
2008		Microsoft começa os experimentos com vGPU, tecnologia que resultara no lançamento posterior da extensão do protocolo RDP para gráficos 3D chamada RemoteFX
2009		Jurgelionis et al [20] expõe uma proposta de plataforma distribuída para entrega interna do streaming e interação com os jogos em um ambiente doméstico para múltiplos dispositivos
2009		OnLive anuncia que irá lançar o primeiro serviço de cloud gaming disponível no EUA, em parceria com a provedora de infraestrutura de comunicação AT&T
2009		Gaikai anuncia que irá lançar um serviço de demonstração de games por cloud gaming. Posteriormente esse serviço foi adquirido pela Sony Computer Entertainment

2010		Microsoft lança o RemoteFX na atualização do Windows Server 2008 SP1
2010		Playcast Media Systems e SFR lançam seus serviços de cloud gaming em operadoras de IPTV na Europa
2010		OnLive lança seu serviço
2010		Inicia-se o trabalho de conclusão de curso, do autor deste trabalho, intitulado “Captura, Compactação e Streaming de vídeo para controle em Ambiente Remoto”, o qual originou esta pesquisa
2011		Gaikai lança seu serviço de demonstração de jogos
2011		Yu et al [22] desenvolve um sistema de desktop remota utilizando dispositivos móveis como *thin clients*.
2011		Viel et al [19] apresentam um framework que utiliza uma estratégia baseada em streaming de vídeo para a interação remota com HPMs
2011		Ubitus GameCloud lança seu serviço em parceria com operadora de 4G LTE no Japão
2012		Durante a GPU Technology Conference (GTC) 2012 a NVIDIA anuncia a sua plataforma de GPUs com suporte virtualização em hardware chamada GRID
2012		Iniciam-se as atividades desta pesquisa, com a integração de um módulo cliente da solução proposta de HPM remota em contexto multimídia [1]
2012		Torterolo et al [24] demonstra um sistema com o objetivo de acessar aplicações gráficas 3D de ciências biológicas remotamente e de forma colaborativa

2012		Zhao et al [23] apresentam-se testes de um sistema de cloud gaming composto por servidores virtuais e com GPUs dedicadas (não virtuais)
2013		NVIDIA GRID é lançado em parceria com empresas de servidores, como DELL e HP
2013		Amazon anuncia parceria com NVIDIA para incorporar ao seu serviço AWS as GPUs GRID
2013		Huang et al [REF GA 2013] expõe a primeira plataforma para cloud gaming open source
2014		Sony lança o serviço de cloud gaming da Gaikai para a sua nova geração de consoles PS4
2014		Amazon AWS lança o novo serviço AppStream, baseado na infraestrutura de GPUs GRID

Como se visualiza no panorama da **Tabela 1**, ao longo dos últimos anos, a academia e o mercado estão consolidando a tendência estudada, desde o início desta pesquisa, sobre as possibilidades interessantes do uso do conceito de HPM remota. Iniciativas como as realizadas pela Nvidia, com a tecnologia VGX e GRID, possibilitando serviços com renderização e interação remotas em HPMs como os lançados recentemente pela Amazon e SONY (AppStream e Gaikai) evidenciam a demanda por mais pesquisas nessa área.

Este trabalho busca esclarecer quais desafios tecnológicos são decorrentes da técnica de interação remota com HPMs, bem como, as soluções para os problemas inerentes da implementação de uma solução que contemple o contexto heterogêneo de dispositivos CE, dependências de software e hardware de HPMs e com uma arquitetura modular e flexível para adequações à tecnologia futuras. Essas são características que não são contempladas em sua totalidade nos trabalhos até então existentes na academia e no mercado.

A solução proposta e seus componentes são detalhados no Capítulo 6.

Capítulo 3

ARQUITETURA PROPOSTA

3.1 Visão Geral da Proposta

A estratégia de solução proposta neste projeto de pesquisa é a execução remota de HPMs. Para alcançar esse objetivo é projetada uma arquitetura que oferecerá as diretrizes necessárias para o desenvolvimento dos componentes de forma modular, possibilitando a substituição flexível e adequação às futuras tecnologias.

Uma visão geral da proposta é detalhada a seguir:

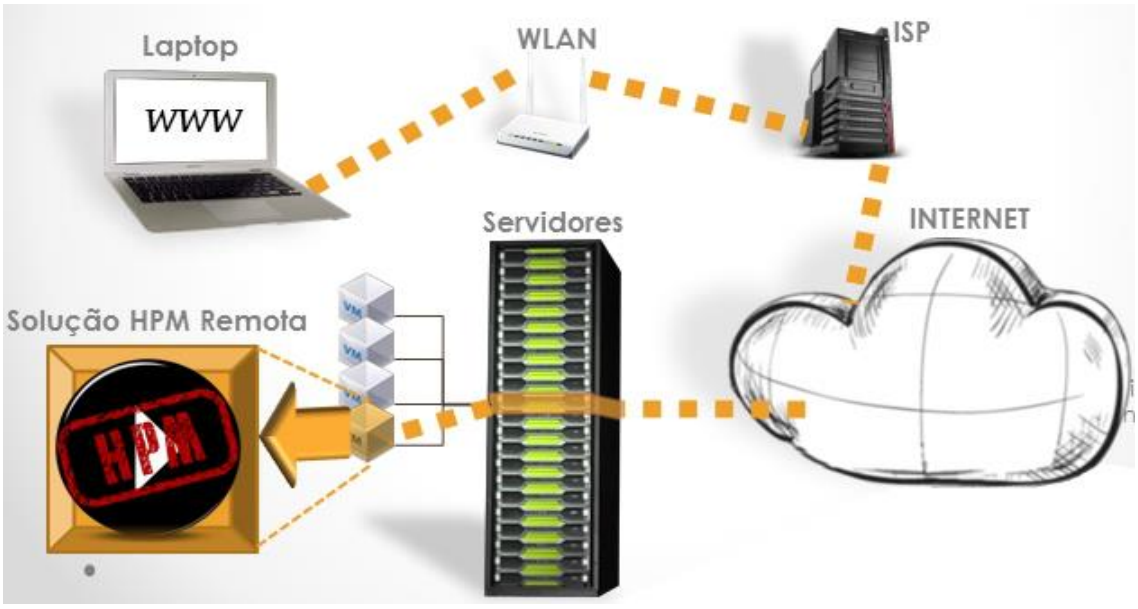

Figura 10. Representação do sistema de execução remota de uma HPM: requisição do acesso remoto pelo usuário.

Como representado na **Figura 10** (sentido horário), o usuário acessa um website, cria uma requisição de acesso à HPM remota, passa pela rede local (WLAN) e provedor de Internet (ISP) até chegar, por meio da Internet, em um dos servidores. Nesse servidor é selecionada uma máquina virtual que esteja totalmente disponível ou com baixa utilização para atender a nova demanda. Por fim, a solução de HPM remota instancia a HPM requerida para o usuário que criou a requisição.

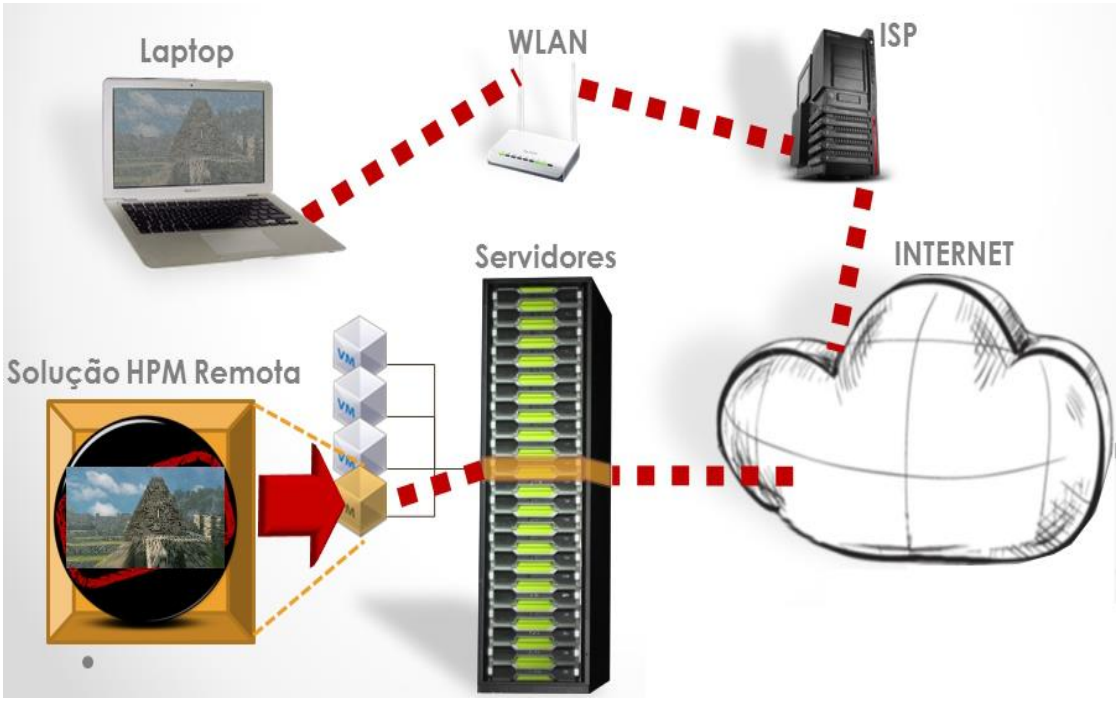

Figura 11. Representação da resposta da requisição por meio de um pixel streaming até o usuário.

Na **Figura 11** (sentido anti-horário) é apresentado o pixel streaming sendo enviado ao usuário que requisitou o acesso. Por meio do website acessado pelo usuário é realizado o *playback* do pixel streaming recebido e a interceptação das interações (mouse, teclado, touchscreen, etc.), enviando-as até a solução de HPM remota. Dessa forma, completa-se o ciclo de interação e resposta audiovisual, no momento em que a interação remota do usuário ocasiona uma alteração no estado da HPM, que por sua vez, gera uma nova resposta sensorial ao usuário.

Nessa visão geral, semelhante ao realizado em [UBITUS 4G], considera-se o ISP devido à possibilidade de se posicionar nós da cloud nas centrais de ultima milha dos provedores de conexão. Tal possibilidade se torna interessante pelo baixíssimo atraso no tempo de resposta entre clientes desse provedor e as centrais regionais. Entretanto,

o aumento no custo da infraestrutura dessa distribuição dos nós da *cloud* deve ser levado em consideração.

3.2 Requisitos

Por meio dos estudos realizados, nos desafios e problemas específicos, foi possível elencar a existência dos seguintes requisitos principais para a arquitetura, nos quais foram considerados os dois cenários base: (a) HPM de alta iteratividade em uma LAN e (b) uma HPM com alta tolerância a atraso de interação em WAN.

a. Um atraso da ordem de 160ms é considerado aceitável entre a interação de um usuário e o *display* da imagem no monitor em um sistema comum (renderização e interação realizadas na mesma máquina, localmente)[17]. Para um outro estudo sobre a percepção da qualidade em *streamings* o tempo de resposta médio humano (tolerância a atrasos) é apresentado como cerca de 200ms [26]. Com esses dados é definido que para o cenário (a) o atraso máximo da plataforma em seu tempo de resposta ao usuário deve ser inferior a 200ms, preferencialmente no máximo por volta de 150 ms. Para o cenário (b), devido a sua característica de maior tolerância e baseado em estudos prévios realizados, define-se o tempo máximo de resposta como até 500ms [1] [19] [27].
b. Devido às características dos dispositivos CE, os quais possuem plataformas bastante diferentes na sua maioria, observa-se uma necessidade de realizar o desenvolvimento do componente cliente de forma *cross-platform*. Porém, não há restrição na arquitetura projetada para um determinado tipo de implementação do módulo cliente. Assim, a aplicação HTML5 desenvolvida neste projeto, além de seu funcionamento em dispositivos portáteis e móveis, pode ser utilizada como referência para implementações do módulo cliente em outras plataformas.

c. Para possibilitar a flexibilidade no processo de captura e inserção nas HPMs, define-se a o requisito de não intrusão ou alteração da HPM para o uso na plataforma.
d. Como requisitos mínimos para utilização de uma HPM na plataforma é utilizada a forma de renderização implementada pelo módulo de captura, neste caso, deve ser uma aplicação Microsoft Windows utilizando a API DirectX versões 9c ou superior.
e. Para efetivar as interações enviadas pelo componente cliente com a HPM, é necessário evitar dependências de virtualização de hardware. Com isso espera-se diminuir o acoplamento a plataforma utilizada nos servidores e maximizar a compatibilidade com diversos tipos de HPM.
f. Restrição de largura de banda do streaming de vídeo foram fixadas, para os codecs descritos na **Seção 6.3**, para cada cenário em: (a) 10Mbps em resolução de 1280x720 pixels e 15Mbps em 1920x1080; Em (b) 4Mbps na resolução de 854x480 e 6Mbps na 1024x600 pixels. Essas restrições baseiam-se na popularização de telas *FullHD*, para o cenário (a), e da grande disponibilidade de conexão banda larga, mesmo para redes móveis como 3G+ e 4G.
g. Como requisito do protocolo de streaming duas características são desejadas: empacotamento/desempacotamento em tempo real e compatibilidade com os requisitos da aplicação cliente. A primeira característica mantém uma dependência direta com a capacidade de processamento do hardware cliente.
h. Para a codificação de vídeo/áudio o fator mais importante é seu impacto no tempo de resposta e decodificação do *streaming*. Influenciando diretamente no tipo de *streaming* a ser codificado, transmitido e decodificado, podendo acrescentar limitações quanto a tecnologias, compatibilidade de hardware e software, e desempenho nas demais partes do sistema.

3.3 Componentes de Software e Implementação

A proposta apresentada neste projeto de pesquisa para uma solução de interação remota com HPM é composta pelos seguintes módulos:

- **Módulo de Captura**:

  Módulo responsável por realizar todas as operações relativas à captura da tela e áudio da aplicação HPM. Em sua implementação foi definido o uso da técnica de inserção de biblioteca dinâmica no dispositivo de renderização 3D. Por meio de observação e protótipos realizados pelo autor, foi possível observar que essa técnica tem a melhor eficiência e menor impacto no desempenho da HPM. A outras técnicas avaliadas foram a utilização de *mirror driver* embutido no Microsoft Windows SDK, a partir da versão 8, e a técnica de captura do dispositivo de renderização e gerenciamento de janelas (GDI) da plataforma Microsoft Windows.

  Utilizando o framework de inserção dinâmica de biblioteca EasyHook[2] implementou-se esse componente de captura gráfica do dispositivo Microsoft DirectX nas versões 9c[3] e 11[4]. O componente foi desenvolvido integrando-se a DLL dos frameworks/bibliotecas com um código otimizado em C++. Partes importantes do processo de captura, como transcodificação da imagem capturada, foram implementadas com suporte a subsets SIMD como SSE2 e SSE4.

- **Módulo de Codificação**:

  Realiza a preparação das imagens capturadas em frames e as codifica no padrão escolhido. Foram realizadas observações com os codificadores FFMPEG[5] e x264[6] por meio de uma comunicação IPC entre o codificador e o módulo de codificação. A seleção de codecs avaliados é composta por: MPEG-2[7], H.264, e

[2] http://easyhook.codeplex.com/
[3] http://www.microsoft.com/downloads/details.aspx?FamilyID=2da43d38-db71-4c1b-bc6a-9b6652cd92a3
[4] http://msdn.microsoft.com/en-us/library/windows/desktop/hh404562%28v=vs.85%29.aspx
[5] http://www.ffmpeg.org/
[6] http://www.videolan.org/developers/x264.html
[7] http://www.comp.nus.edu.sg/~cs5248/l03/IEC-13818-2_Specs.pdf

VP8[8]. Entretanto, devido à flexibilidade da arquitetura componentizada, o módulo de codificação não está restringido a esses padrões.

- **Módulo de Streaming**:

  Recebe o fluxo de frames codificados, empacotando-os em um streaming de vídeo. Para sua implementação realizou-se testes com o intuito de selecionar qual protocolo possui maior compatibilidade entre os CE, viabilizando o streaming e interação com a responsividade necessária. Assim, os protocolos de streaming de dados e vídeo avaliados por observação foram: RTMP[9], RTSP[10], HLS[11] e WebSOCKET[12]. Outro aspecto buscado foi a compatibilidade entre esses protocolos e os padrões de codificação de vídeo e áudio que possuam as características de qualidade visual e performance em tempo real. Esse módulo foi desenvolvido integrando as bibliotecas necessárias para cada protocolo de streaming utilizado nas provas de conceito, tais como: RTMP com a biblioteca cRTMP[13], RTP, RTSP e HLS com a biblioteca Live555[14]. Também foram desenvolvidas versões com o streaming de vídeo em WebSocket e WebRTC[15], utilizando frameworks NodeJS[16] e dotNet[17] para avaliação.

- **Módulo de Playback**:

  Módulo responsável por apresentar o streaming de vídeo recebido na aplicação cliente. Foi implementado um componente *player* de referência (servindo como base para *players* de aplicações clientes em outras plataformas) em HTML5 (com as plataformas alvo sendo dispositivos portáteis e móveis). O *player* é dividido em dois módulos: um desempacotador do protocolo de streaming e o *decoder* de video/áudio. Para interpretar os pacotes de streaming e

[8] http://datatracker.ietf.org/doc/rfc6386/
[9] http://www.adobe.com/devnet/rtmp.html
[10] http://www.ietf.org/rfc/rfc2326.txt
[11] http://tools.ietf.org/html/draft-pantos-http-live-streaming-12
[12] http://tools.ietf.org/html/rfc6455
[13] http://www.rtmpd.com/
[14] http://www.live555.com/liveMedia/
[15] http://www.w3.org/TR/webrtc/
[16] http://nodejs.org/
[17] http://www.microsoft.com/net

realizar *playback* do próprio vídeo utilizou-se funcionalidades já presentes nos navegadores[18] que suportam o HTML5 e bibliotecas de mídia em javascript.

- **Módulo de Controle**:

  Intercepta as interações do usuário com a aplicação cliente, enviando os comandos para o módulo de interação remota. Esse componente é responsável por capturar as interações comuns ao cliente em HTML5 (mouse e tela touchscreen). Um sistema de eventos foi implementado, realizando o encaminhamento de eventos do mouse/touchscreen para o componente de broker. Também realiza o remapeamento de eventos da GUI no dispositivo cliente (tais como mouse e toques na tela touchscreen) para os eventos de mouse e teclado esperados pela aplicação. Utilizou-se as bibliotecas javascript jQuery[19] e jQueryUI[20] para o desenvolvimento da aplicação cliente HTML5 e do sistema de eventos, considerando eventos comuns às plataformas alvo (portátil e móvel), como o arrastar do mouse ou o *swipe* no touchscreen.

- **Módulo de Interação Remota**:

  Módulo responsável por interpretar os comandos em simulações de interações, efetivando a interação inicialmente realizada pelo usuário no componente cliente com a HPM. Implementou-se o componente de simulação de interação com duas técnicas: mensagem de input do sistema e dispositivo virtual de interação. Em ambas as técnicas o objetivo da implementação é permitir a comunicação direta da interação na aplicação, não envolvendo outras aplicações ou dispositivos do sistema. Como o tempo de resposta é um fator crucial, o mesmo se aplica ao tempo necessário para a sincronização entre as interações realizadas no dispositivo cliente e as efetivadas na HPM remota. Qualquer atraso nessa operação de sincronia aumenta significantemente a latência do sistema. Assim, foi necessária a otimização desse processo com a configuração do *broker* de mensagens para que não haja uma perda de sincronismo (efetivação local da

[18] http://caniuse.com/#cats=HTML5
[19] http://jquery.com/
[20] https://jqueryui.com/

interação realizada pelo dispositivo cliente) ou inversão de inputs, com uma interação fora da mesma sequência realizada pelo usuário, entre a aplicação cliente e a aplicação no servidor.

Esses módulos e suas interações com o sistema proposto são explanados a seguir:

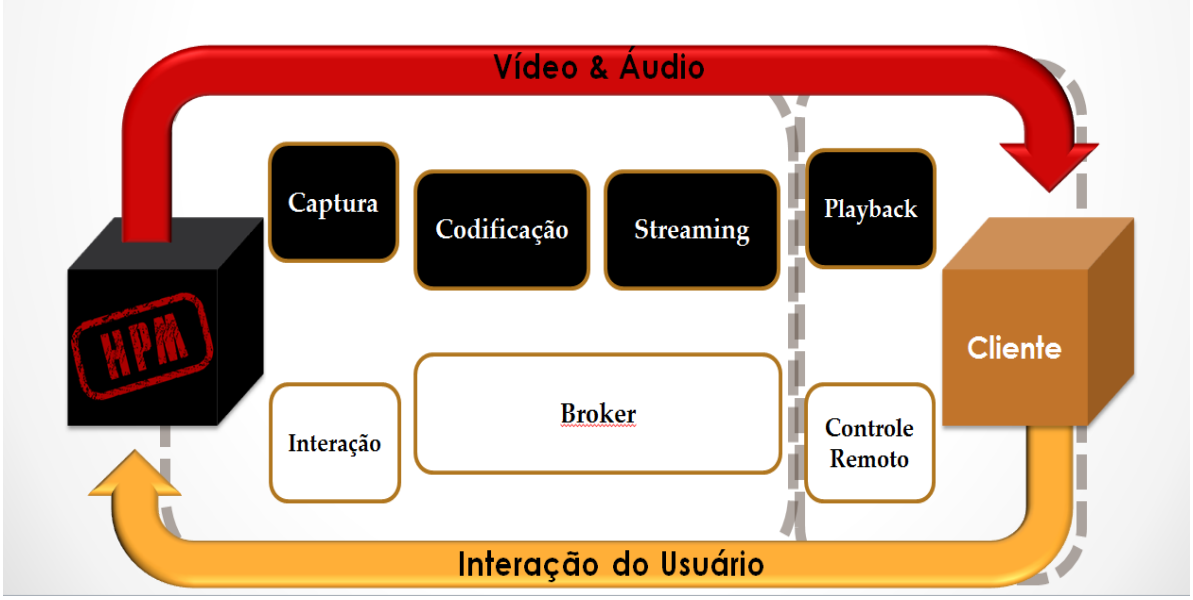

Figura 12. Diagrama dos componentes da solução em interação remota com HPM.

No diagrama apresentado na **Figura 12**, observa-se os módulos componentes da solução de interação remota com HPM, como também, suas duas principais funcionalidades: o envio da renderização capturada da HPM na forma de um pixel streaming (seta superior); e recebimento das interações remotas do usuário (seta inferior).

Já na **Figura 13**, é apresentado o fluxo entre as etapas para realizar a funcionalidade de pixel streaming da renderização capturada da HPM.

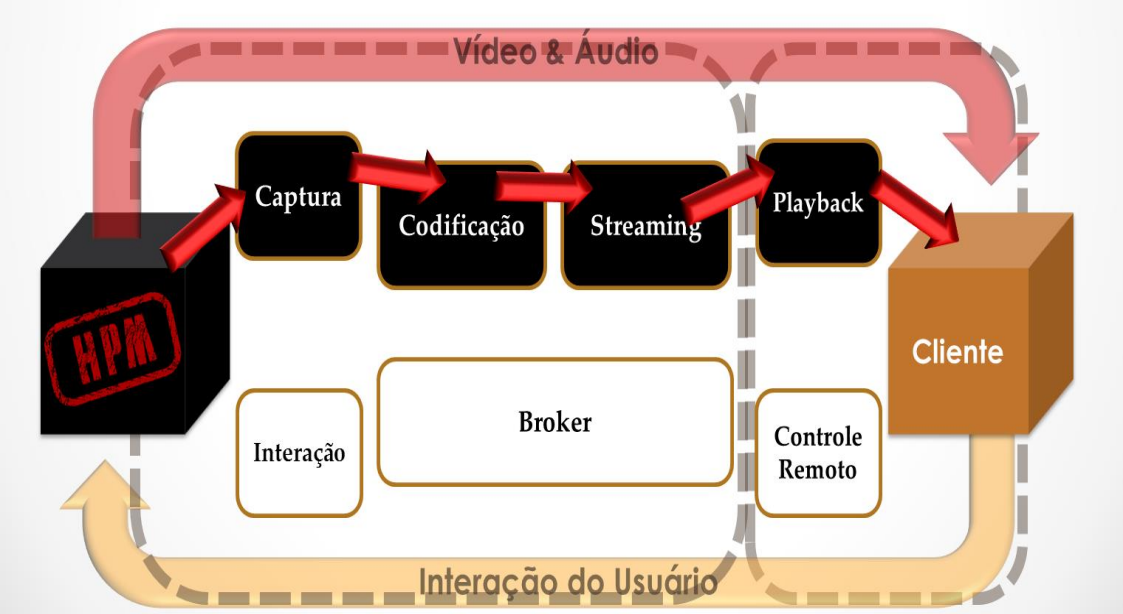

Figura 13. Fluxo entre as etapas de captura da tela da HPM e streaming de vídeo até o usuário.

Na **Figura 13**, a primeira etapa representa a inserção da biblioteca dinâmica no processo da HPM (seta clara superior). Depois, o módulo de captura começa a receber os bits da renderização capturada da HPM e prepara esses dados em uma imagem (bitmap), enviando-os ao módulo de codificação. O módulo de codificação prepara as imagens em frames, codifica-os no padrão de vídeo selecionado e encaminha o fluxo multiplexado de frames ao módulo de streaming,que empacota o fluxo em um pixel streaming. Por fim, o pixel streaming é recebido e apresentado pelo módulo de playback.

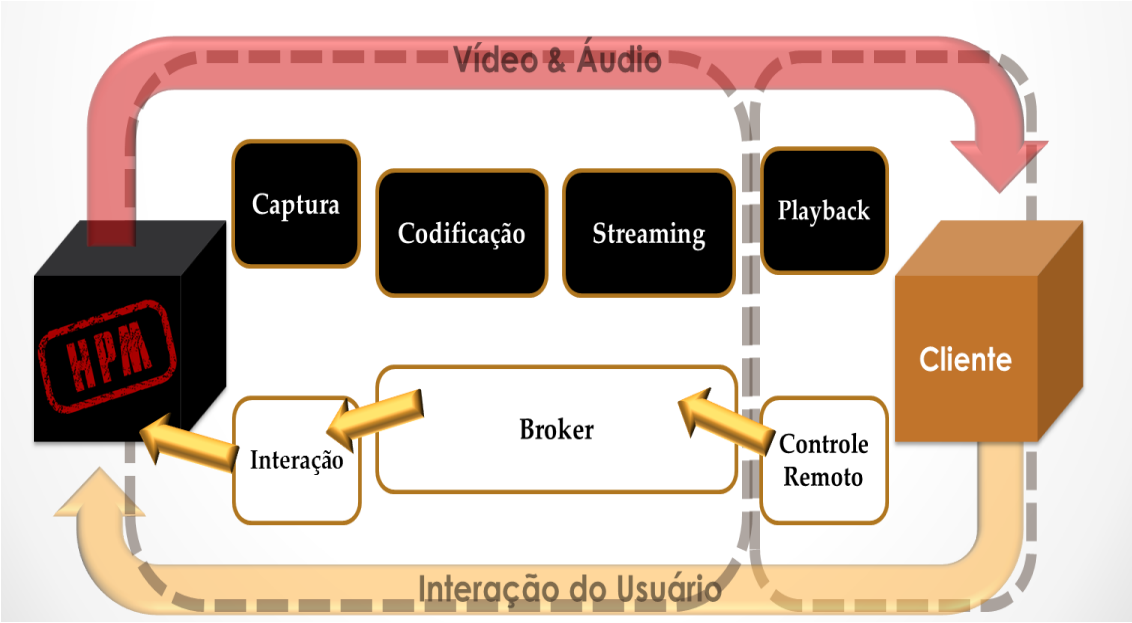

Figura 14. Etapas desde a interceptação da interação do usuário até o controle da HPM remota.

Na **Figura 14** são apresentadas as etapas para realizar a interação remota com a HPM. Primeiramente, o módulo de Controle intercepta as interações do usuário (mouse, teclado, touchscreen, etc.). Após isso, ele envia por meio de uma conexão pré-estabelecida (utilizando um *broker* de mensagens) esses comandos ao módulo de interação, com a formatação de mensagem do sistema de eventos. Esse meio de comunicação flexibiliza e desacopla o módulo de Interação e o módulo de Controle. Concluindo, o módulo de Interação interpreta esses eventos em interações diretamente na HPM.

3.4 Integração com Contexto Multimídia / TVDi

Como prova de conceito da flexibilidade da arquitetura componentizada na solução proposta foi implementado um módulo cliente em um contexto multimídia [27].

O desenvolvimento referente à incorporação de uma HPM em uma máquina de apresentação multimídia foi realizado com a participação nos trabalhos realizados em "*Multimedia presentation integrating interactive media produced in real time with high performance processing*"[1] e "*Multimedia Presentation with Virtual 3D Realistic Environment Produced in Real Time with High Performance Processing*"[27]. As características que foram exploradas e implementadas são: incorporação de uma HPM em um contexto multimídia, para o qual foi criado um interpretador de objetos HPM em uma máquina de apresentação que utiliza a linguagem declarativa NCL (Nested Context Language), e parte do sistema de interação remota, no qual se utiliza um manifesto que descreve as capacidades e atributos da HPM por meio de um arquivo de configuração na linguagem XML (eXtensible Markup Language), com o objetivo de criar uma comunicação e interação desacoplada entre a máquina de apresentação multimídia e a HPM remota.

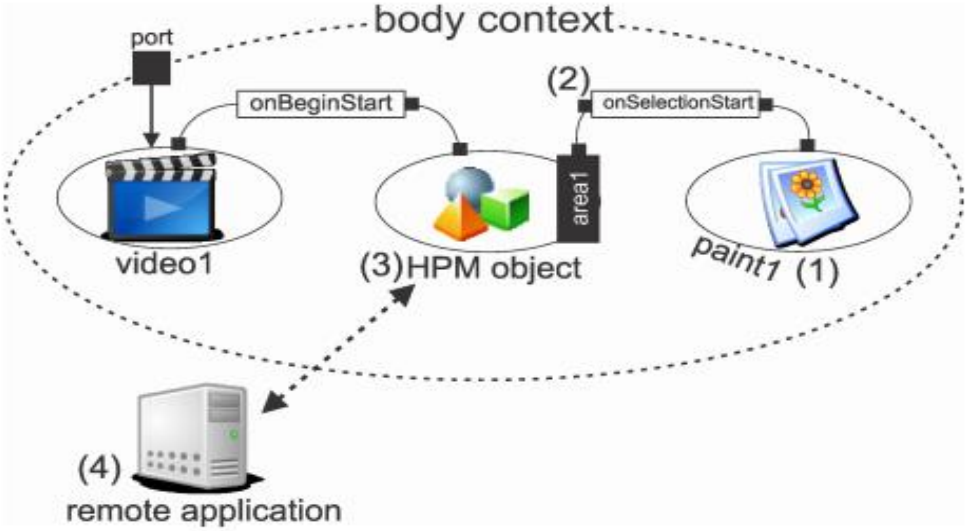

Figura 15. Etapas HPM no contexto multimídia. Retirado de [1]

Na **Figura 15** é apresentado um esquema que demonstra a como a inserção de uma HPM foi projetada [1]. O item 1 é representa uma imagem que se relaciona com a HPM no item 2, que ao disparar um evento na HPM irá ocasionar a apresentação da mídia no item 1. Essa relação demonstra a integração da HPM no contexto multimídia, onde se relaciona e interage com outras mídias, por exemplo, o item 3 demonstra a divisão do vídeo da HPM em uma nova mídia que interage com o objeto HPM e por sua vez com a HPM remota.

As avaliações modulares e da solução proposta são apresentadas no **Capítulo 5**.

Capítulo 4

PROVAS DE CONCEITO E RESULTADOS

4.1 Contexto das Avaliações

Os resultados apresentados neste capítulo foram obtidos por meio da execução de duas HPMs com características opostas: a primeira é uma visualização arquitetônica (maquete virtual) desenvolvida em [5] e a segunda é um jogo de demonstração do framework de desenvolvimento de jogos UDK[21]. A escolha dessas HPMs se deve às suas características contrastantes quanto a tolerância no atraso de tempo de resposta. Na maquete virtual, o usuário interage de forma menos contínua devido ao seu caráter contemplativo, nela alguns centésimos de segundo de atraso não anulam a experiência de visualização do ambiente virtual (desde que a qualidade visual esteja em um nível de alto realismo). Já a HPM que representa jogos eletrônicos de ação em primeira pessoa (*First Person Shooter*), possui uma taxa de interações do usuário por minuto bastante alta e cada décimo de segundo na latência da interação impacta no objetivo da experiência.

[21] https://www.unrealengine.com/products/udk/

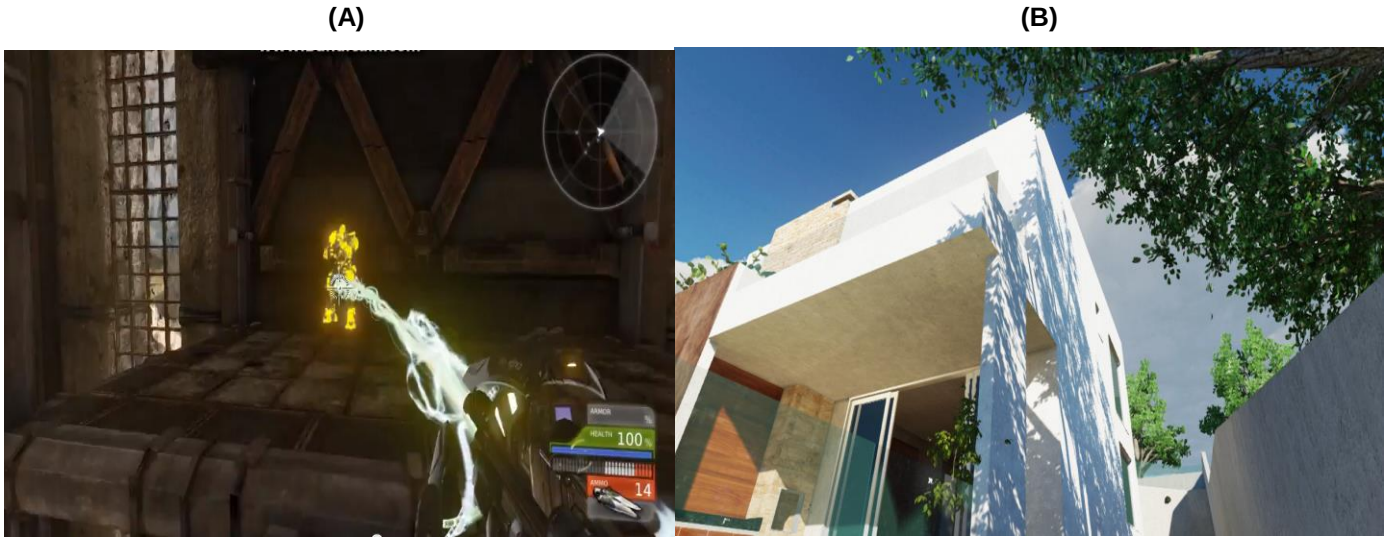

Figura 16. Telas das HPMs utilizadas nos experimentos. HPM "A" representa jogos de ação e a "B" a maquete virtual interativa.

Na **Figura 16** são apresentadas telas das duas HPMs, onde o item A é a tela do jogo eletrônico e o item B a tela da maquete virtual. Cada HPM foi executada no cenários: a visualização arquitetônica no cenário (a) de WAN e o jogo de ação em (b) uma LAN, como definidos no **Capítulo 1**. Em cada teste realizado utilizou-se os seguintes dispositivos clientes: Ultrabook com GPU integrada de baixo consumo energético e um Smarpthone Android 4.3. Esses dispositivos são representantes das plataformas portátil e móvel, respectivamente. Para os experimentos com plataforma móvel também verificou-se a compatibilidade *cross-platform* com um tablet executando o sistema operacional iOS 5.1.1. Como servidor utilizou-se um computador com processador de 4 núcleos em 3,0GHz, 8GB de memória RAM e uma GPU Nvidia GTX 760 com vRAM de 2GB DDR5.

Para a obtenção dos dados de latência de rede, largura de banda utilizada e outras características de comunicação em rede foi utilizado um *sniffer* de pacotes chamado WireShark[22]. Já para as medições entre módulos locais (que não se comunicam por rede), utilizou-se o *timestamp* local, pois um mesmo servidor os estava executando.

Nas medições de atraso no tempo de resposta dos módulos de captura, codificação e streaming utilizou-se uma técnica de marcação de frames, semelhante a

[22] http://www.wireshark.org/

utilizada na ferramenta NVIDIA FCAT[23]. O conceito utilizado consiste em realizar uma marcação no frame renderizado com a identificação temporal, baseando-se no *timestamp* do servidor, e compará-lo com o tempo obtido, na mesma máquina, para obter a latência de cada módulo interno (captura, codificação e streaming).

Para o teste da decodificação compara-se o momento do registro no módulo de streaming com a marcação do frame apresentado no dispositivo cliente. Para realizar esse registro independente de sincronização entre o servidor e o dispositivo cliente, utiliza-se um terceiro dispositivo (câmera digital, filmadora, etc) para gravação da apresentação dos frames marcados em cada dispositivo no mesmo momento.

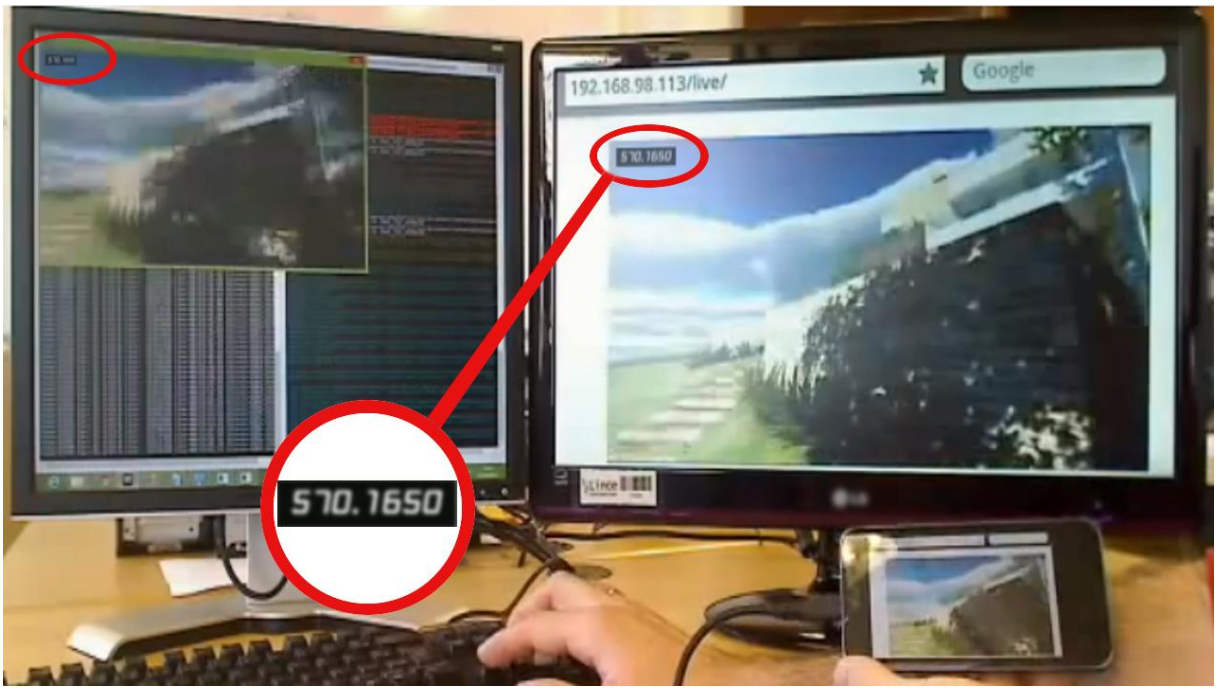

Figura 17. Exemplo de gravação registrando os frames marcados no servidor (esquerda) e no dispositivo móvel (direita), o qual está conectado a um monitor pela conexão HDMI.

Na **Figura 17** um exemplo da gravação utilizada para registro da latência do módulo de *playback* é apresentada, destacando-se a marcação temporal dos frames. Para tratar e obter os dados coletados foi utilizada em cada amostra (imagem registrada) uma técnica de corte automático dos quadrantes onde se encontram as marcações dos frames e, posteriormente, o reconhecimento de caracteres de cada marcação.

Como métricas quantitativas de qualidade visual de vídeo foram utilizadas a PSNR (*Peak Signal-to-Noise Ratio*) e a SSIM (*Structural Similarity*). Para obter esses

[23] http://www.geforce.com/hardware/technology/fcat/technology

índices a ferramenta utilizada foi a MSU Video Quality Measurement Tool[24], na versão 3.0 64-bit.

Figura 18. Telas da aplicação TVDi com o módulo cliente integrado. O item A mostra a tela de navegação e o item B a tela com outras mídias (imagens) que interagem com a HPM.

A **Figura 18** apresenta-se duas telas do trabalho desenvolvido para integração do módulo cliente em um ambiente multimídia [27]. No item A é visualizado a tela de nevegação da aplicação TVDi cliente para interação com a HPM remota. O item B mostra a tela de sincronização entre outras mídias (imagens de pinturas dentro do ambiente virtual) com a HPM, sendo disparadas por eventos do contexto multimídia (âncoras).

As seções seguintes apresentam os resultados obtidos nos módulos e na plataforma como um todo.

4.2 Latência

Para a obtenção dos resultados referentes a latência identificou-se o momento de uso em cada HPM que ocasiona o maior número de interações e que exige a resposta mais rápida do usuário, dentro das características da HPM utilizada. Para o jogo o momento de interação mais intensa é a situação em que há vários inimigos na área de visão (cerca de 6 a 8 personagens controlados pelo computador). Na maquete virtual selecionou-se o momento em que o usuário interage com os elementos de mobília ao

[24] http://compression.ru/video/quality_measure/video_measurement_tool_en.html

mesmo tempo em que observa as animações da vegetação. Esses momentos também utilizados para os testes de latência da codificação por atualizarem rapidamente vários elementos da tela, aumentando a diferença entre quadros (*interframes*).

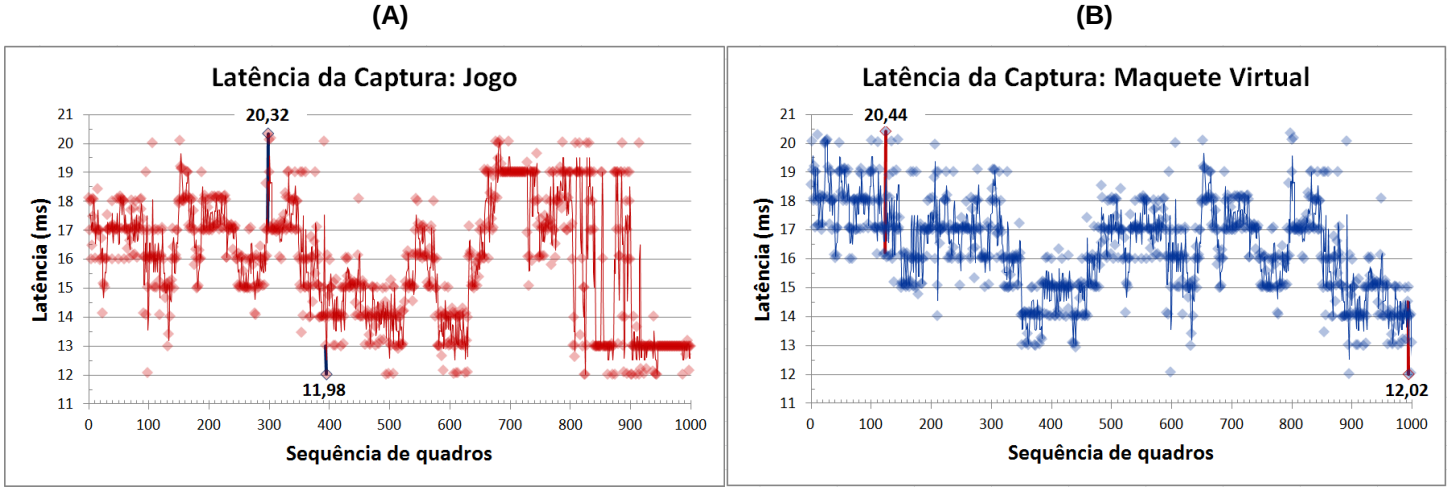

Figura 19. No gráfico A é apresentada a latência de captura para cada quadro em uma amostragem de 1000 quadros para do jogo. No B são mostradas as latências de captura para os quadros da maquete virtual.

Na **Figura 19** apresenta-se as latências obtidas para uma amostragem de 1000 quadros em cada HPM, capturados nos momentos correspondentes. No gráfico A percebe-se que houve uma variabilidade maior em relação ao B, principalmente no trecho final. Essa variabilidade pode ser decorrente da espera do módulo de captura pela renderização ser finalizada. No experimento do gráfico A houve uma variação muito maior do desempenho da renderização no sistema (34 a 56 quadros por segundo), devido à grande quantidade de animações e polígonos no campo de visão, do que a presenciada no experimento da maquete virtual, o qual se manteve estável a 60 quadros por segundo (limite fixado para essa HPM).

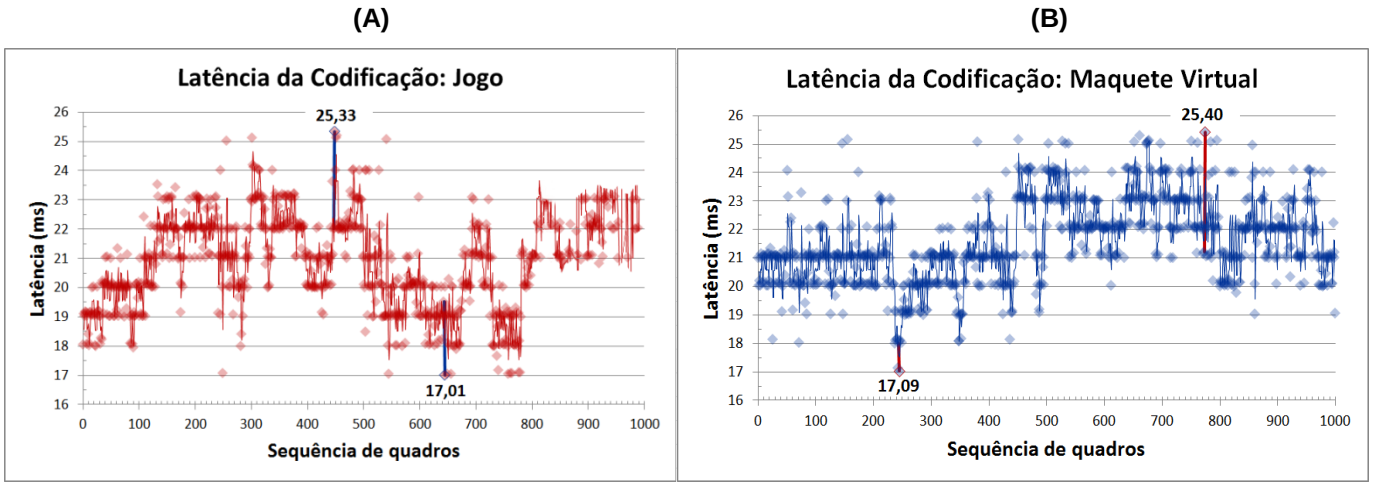

Figura 20. O gráfico A é apresenta a latência de codificação em cada quadro do momento escolhido do jogo. No B são mostradas as latências para a maquete virtual.

A **Figura 20** apresenta as latências de codificação dos quadros para as duas HPMs. Observa-se que mesmo a variação ao longo do tempo ter sido grande, devido ao tipo e quantidade de interações diferentes, a latência máxima e mínima foram similares. A latência média apresentada no gráfico B (22,3ms) é relativamente superior a apresentada no gráfico A (20,1ms), possivelmente, devido a quantidade alterada da tela durante as movimentações (giro da visão) e animações (árvores, vegetação, etc) da maquete virtual.

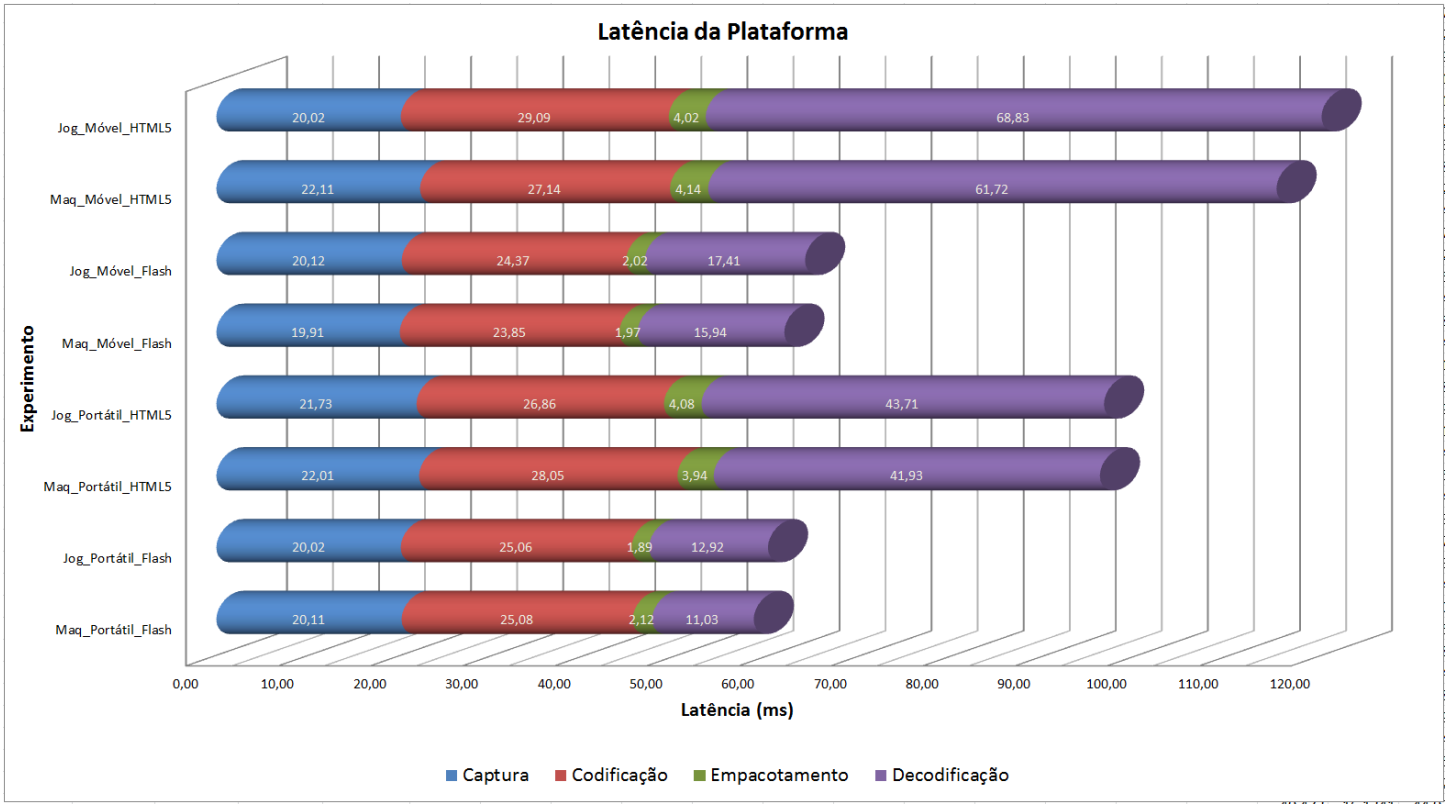

Figura 21. Representação da latência acumulada por cada módulo, resultando na latência total da solução para cada experimento realizado. Demonstra-se os resultados para as cominações de HPM (jogo ou maquete virtual), plataforma (móvel ou portátil) e módulo cliente (HTML5 ou flash).

Na **Figura 21** são apresentados os valores de latência para cada módulo, bem como os valores acumulados para a plataforma em cada experimento realizado (combinação do tipo de dispositivo CE, HPM e implementação do módulo cliente). Para os resultados do módulo de *playback* considerou-se o tempo de desempacotamento e decodificação. Não se considerou o tempo de rede para isolar os resultados da plataforma.

Analisando esse gráfico é possível perceber que o impacto para a latência da plataforma do módulo de *streaming* é mínimo, não ultrapassando 4ms para o

empacotamento HTML5 e 2ms para a versão em flash. Também se percebe que o maior impacto para a latência, com exceção do dispositivo móvel com HTML5, é do conjunto de operações dos módulos de captura e codificação. Outro ponto interessante, é a comparação entre a decodificação em HTML5 usada (MPEG2 diretamente em javascript), a qual não tem suporte de hardware, com a versão em flash (H.264) com suporte nativo (aceleração de hardware), demonstrando um atraso 4 vezes maior para a decodificação em javascript (experimental).

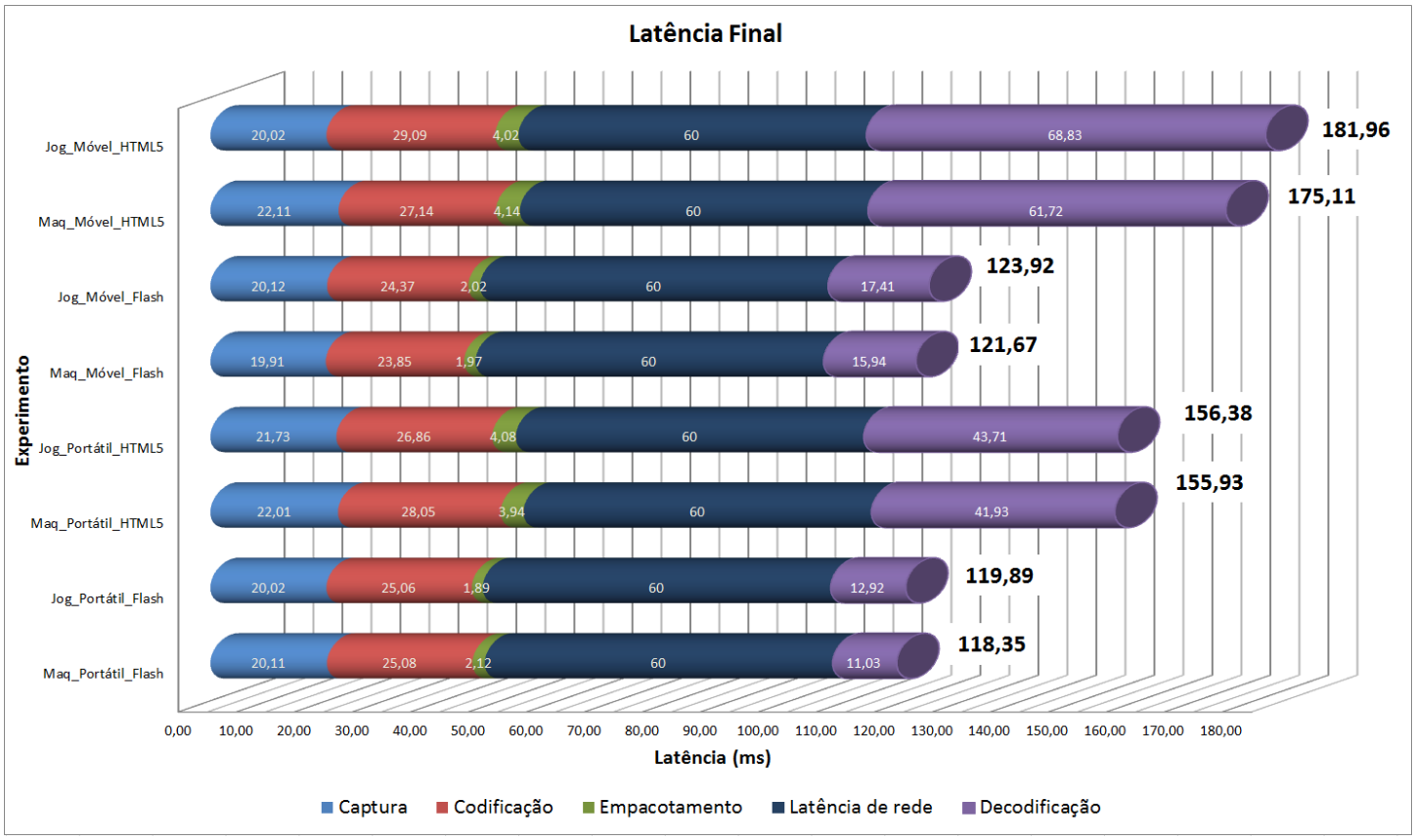

Figura 22. Gráfico do acumulado de latência da plataforma considerando o atraso de rede médio encontrado na infraestrutura de Internet banda larga do Brasil.

No gráfico da **Figura 22** a latência da plataforma com o tempo de atraso de rede é considerado. Esse atraso de 60ms inserido como latência de rede corresponde a média das conexões banda larga do Brasil [28]. Também se considera parte desse atraso a envio das interações interceptadas no módulo cliente.

Com esses resultados, observa-se que, mesmo considerando uma latência de rede WAN, apenas as versões com o módulo cliente em HTML5 ficaram com a latência total próxima ou acima da ideal (160ms).

4.3 Largura de Banda

A largura de banda média das conexões de banda larga no Brasil é de cerca de 2,7Mbps. Entretanto nos grandes centros urbanos esse valor, dependendo da região, chega a 10Mbps [28]. Por meio de testes com diferentes configurações de compressão e largura de banda, observou-se que limitando a largura de banda dos *streamings*, nas resoluções 720p (1280x720) e 480p (854x480), para 9Mbps e 6Mbps respectivamente, foram obtidos resultados com qualidade visual satisfatória mantendo a baixa latência na codificação.

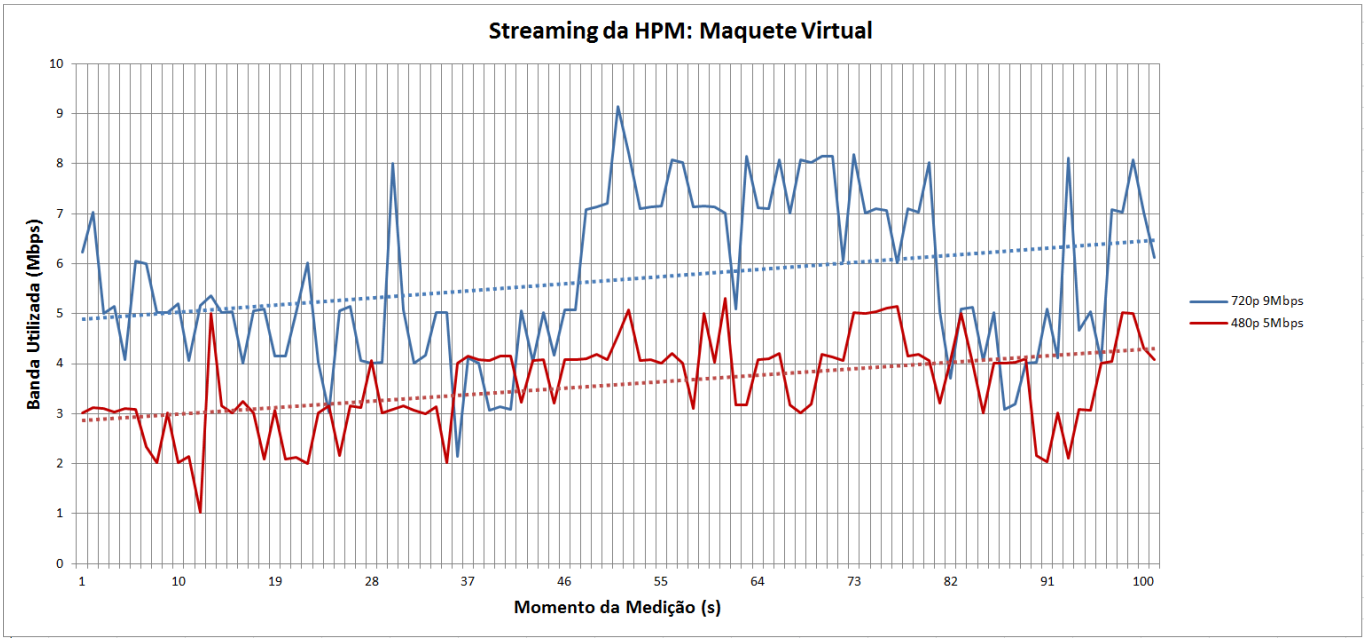

Figura 23. Apresentação dos dados obtidos da largura de banda utilizada para duas configurações de streaming: primeiro em resolução 720p (1280x720) limitado a 9Mbps e o segundo com uma resolução de 480p (854x480) até 6Mbps.

Na **Figura 23** o uso da largura de banda pelos streamings (720p e 480p) utilizando a implementação em flash com o protocolo RTMP são apresentados para um período de 100 segundos. Não foram considerados os resultados da implementação em HTML5 por ser uma técnica considerada *pseudo-streaming* (onde envia-se porções do arquivo codificado em tempo real ao invés de pacotes reais). Observa-se que com a largura de banda necessária os streamings utilizaram apenas 7Mbps e 4Mbps em média dos totais disponíveis.

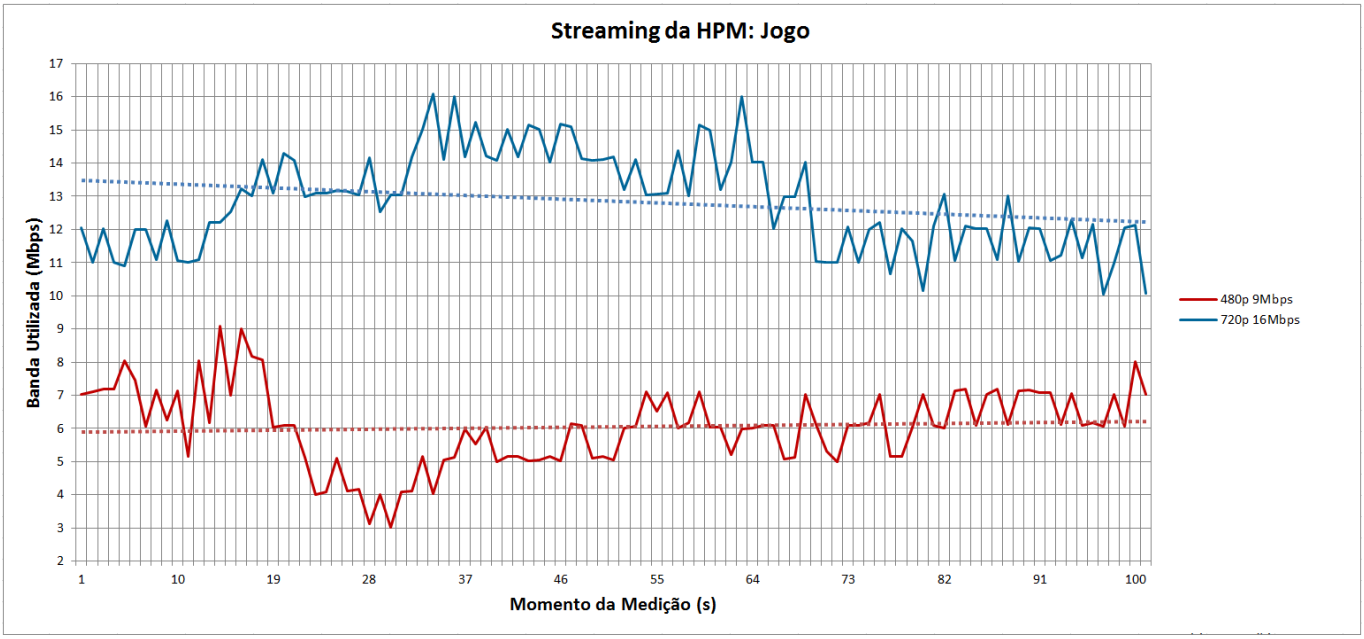

Figura 24. Taxa de bits utilizada para o streaming em dois casos: 720p com até 16Mbps de largura de banda e 480p com 9Mbps.

No gráfico da **Figura 24** a largura de banda média utilizada para simular os cenários em LAN para as duas resoluções avaliadas também foram em inferiores aos limites, atingindo 12,5Mbps para o teste em 720p e 6,5Mbps para o teste em 480p.

Percebe-se com a avaliação da largura de banda necessária que em todos os casos os limites foram respeitados, permitindo a usuários dos dois cenários de teste (WLAN com 54Mbps e WAN com 10Mbps de largura de banda) utilizar a solução satisfatoriamente.

4.4 Qualidade Visual

As métricas utilizadas para a análise objetiva da qualidade visual são a taxa em decibéis do PSNR e o índice de similaridade SSIM. O primeiro mensura a distorção ocorrida do quadro codificado em relação ao original. Já o SSIM, mensura a semelhança entre o quadro codificado e a sua referência original, onde 1 é o valor máximo do índice.

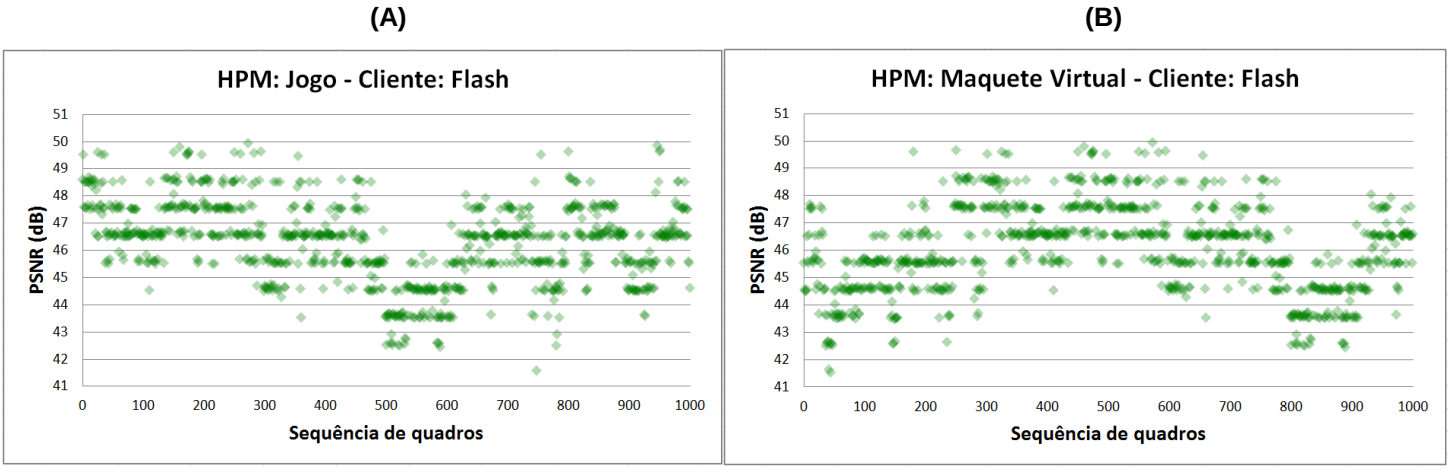

Figura 25. Representação gráfica dos índices PSNR, em decibéis, obtidos para (A) jogo e (B) maquete virtual, ambos com a versão em flash do módulo cliente, com uma amostragem de 1000 quadros.

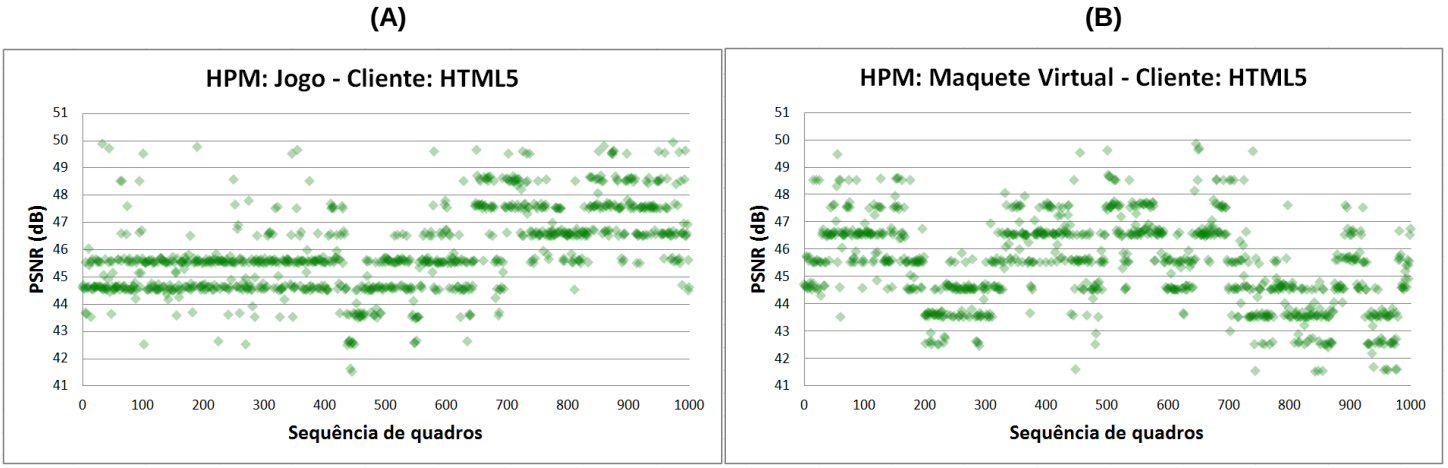

Figura 26. Taxa PSNR (dB) para cada quadro analisado com as HPMs (A) jogo e (B) maquete virtual, ambos com o módulo cliente implementado em HTML5 e uma amostragem de 1000 quadros.

Os dados apresentados nos gráficos da **Figura 25** e da **Figura 26** foram obtidos comparando-se o quadro capturado da HPM com o quadro resultante do módulo de codificação.

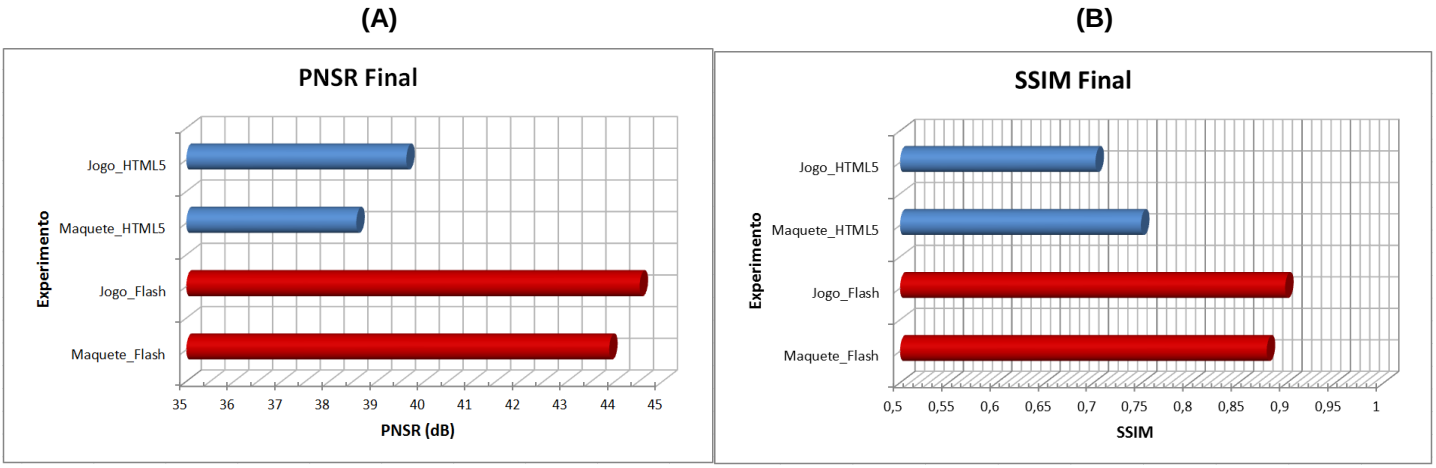

Figura 27. Taxas médias (A) PSNR e (B) SSIM obtidas para cada HPM (jogo e maquete virtual) testados com implementações do módulo cliente em flash e HTML5.

Nos gráficos da **Figura 27** o resultado médio das métricas são apresentados para cada HPM (jogo e maquete virtual) em cada implementação do módulo cliente (flash e javascript). Observa-se com a diferença entre os resultados da implementação em javascript, que utilizou um *codec* mais antigo e menos otimizado (MPEG2), e a implementação em flash, utilizando o H.264, o resultado visual do *streaming* é dependente do padrão de codificação utilizado.

Para a métrica PSNR valores entre 30 e 40 são geralmente satisfatórios, acima de 45 apresentam uma quantidade de ruído imperceptível [29]. As médias obtidas com a implementação HTML5 foram 39,5dB e 38,5dB (jogo e maquete virtual, respectivamente). Com a implementação em flash os resultados foram 44,3dB e 43,6dB, respectivamente.

Utilizando a métrica SSIM, índices entre 0,7 e 0,9 são comumente satisfatórios, qualquer valor acima de 0,9 possui diferenças irrelevantes [30]. Na **Figura 27 B** observa-se que as médias resultantes para os experimentos em HTML5 foram 0,695 para o jogo e 0,74 para a maquete virtual. Para a implementação em flash os resultados foram 0,89 e 0,87 para jogo e maquete virtual, respectivamente. Mais uma vez demonstrando a dependência da qualidade com o *codec* utilizado.

A partir desses resultados é possível afirmar que os experimentos desempenharam um fator de qualidade visual aceitável, principalmente com a implementação em flash que também obteve o melhor uso da largura de banda.

4.5 Limitações

Para a prova de conceito realizada no contexto de uma rede local com uma aplicação com baixa tolerância a atrasos no tempo de resposta das interações do usuário utilizou-se um roteador comum WiFi no padrão 802.11B/G/N (com adaptação dependendo dos dispositivos conectados). Como a latência na entrega de pacotes em tempo real é sensível ao desempenho de roteamento, os resultados de latência podem sofrer influência da capacidade de processamento e da sobrecarga da rede. Esse fator pode influenciar também a etapa final (já no contexto local) da prova de conceito realizada em WAN.

Outra limitação dos resultados é devido a técnica de captura dos quadros renderizados, a qual é dependente (de forma síncrona e direta) ao processo de rendeização da GPU. Assim, caso haja uma sobrecarga da GPU haverá um atraso e perda de quadros no processo de captura, influenciando tanto na latência quanto na qualidade da experiência perceptível (fluidez do vídeo).

A fluidez do streaming de vídeo recebido no dispositivo cliente pode ser também afetada por alguma sobrecarga no processamento do dispositivo. Assim, caso haja um excesso de aplicações utilizando o processamento da CPU integrada a qualidade da experiência bem como a latência podem ser afetadas.

Capítulo 5

CONCLUSÕES E TRABALHOS FUTUROS

As características físicas e de software expostas ao longo deste trabalho demonstram que a execução de HPMs é um processo altamente especializado e que os recursos computacionais de um dispositivo CE não possuem esse nível de processamento especializado. Assim, não é diretamente possível a execução de HPMs em dispositivos CE. Com essa limitação como problema de pesquisa, este trabalho propõe a exploração de uma solução tradicional: a interação remota com essas aplicações. Entretanto, os desafios decorrentes da implementação dessa solução, com toda a diversidade de dispositivos e aplicações envolvidas, não são triviais.

Nesse contexto, apresentou-se uma possível solução para a apresentação remota de HPMs em dispositivos CE computacionais, como dispositivos das plataformas portátil e móvel. Por meio dos resultados obtidos é possível afirmar que para os dois cenários alvo propostos, um possuindo uma HPM com baixa tolerância a atrasos no tempo de resposta das interações, em um contexto de rede local, e outro com uma HPM tolerante a atrasos maiores no tempos de resposta, há indícios

de que a solução desenvolvida atendeu os requisitos definidos, dentre eles a diretriz de não interferir na percepção de qualidade visual e interação original do uso da HPM.

Mesmo utilizando os dados de latência da plataforma com os maiores valores, somando-se a média de latência da Internet no Brasil, apenas a combinação dos experimentos em que envolviam a plataforma móvel (smartphone) e o decodificador em HTML5 ultrapassaram o valor de 160ms (o cenário da visualização arquitetônica com ~175ms e o cenário do jogo eletrônico com ~181ms). Assim, mantendo a qualidade visual com a largura de banda necessária (~13Mbps para o cenário do jogo em LAN e ~7Mbps para o cenário da visualização arquitetônica), a percepção de interação e qualidade visual para o usuário seria a mesma ou muito próxima da original (com a HPM executada localmente).

A prova de conceito, por meio do protótipo implementado, demonstrou a capacidade da arquitetura projetada de solucionar o problema de pesquisa também para um contexto multimídia, como demonstrado no trabalho [1]. Essa implementação, em um contexto diferente do desenvolvido originalmente, demonstra a flexibilidade da solução proposta.

5.1 Avaliações e Contribuições

Os processos de experimentação de tecnologias envolvidas com a solução proposta, realizados neste trabalho, e suas avaliações, permitem que os resultados e técnicas abordadas sejam utilizados em diversas áreas de pesquisas correlacionadas. A seguir são descritos os principais pontos de contribuição desta pesquisa:

a. *Captura de renderização gráfica com aceleração 3D de hardware*: a técnica de inserção de DLL e captura do *frame buffer*, utilizada neste trabalho, demonstrou ter o menor tempo de captura, cerca de 20ms em média, e ainda existindo possiblidade de maior otimização;

b. *Flexibilidade de integração de componentes a uma arquitetura modular***:** também houve a contribuição da arquitetura componentizada

proposta, na qual foi demonstrada a flexibilidade de alterações dos componentes de mídia com a integração de uma HPM em um contexto multimídia [1] [27];

Também podem ser consideradas possíveis contribuições os estudos realizado nos seguintes tópicos:

a. *Codificação e streaming em tempo real*: os testes realizados demonstraram que a codificação é a etapa com maior latência dentro do sistema, sendo assim, possui a maior necessidade de otimização do desempenho. Com a utilização do padrão de codificação H.264 e o protocolo de streaming RTMP foi possível obter a menor latência, cerca de 30ms. Porém, com essa escolha haveria uma limitação nos possíveis clientes, devido ao suporte restritivo a esse protocolo de streaming (sistemas móveis como o iOS e o Android não suportam mais nativamente esse protocolo). Portanto, também foi avaliada a utilização de técnicas de streaming com suporte direto nos browsers das plataformas portátil e móvel.

b. *Playback de baixa latência do live streaming multiplataforma*: considerando a grande diversidade de dispositivos clientes, nas plataformas portátil e móvel, e o acoplamento entre *codecs* suportados por hardware com o desempenho de apresentação, encontrar uma combinação de tecnologias é o um desafio. Por isso, foram separados em duas opções: uma com o melhor desempenho (H.264 e RTMP) de qualidade visual, largura de banda utilizada e latência, mas com um restrição de suporte no software/hardware cliente; e outra com amplo suporte (MPEG2 e HTML5/javascript) entre os dispositivos CE, mas com desempenho inferior;

c. *Mapeamento da interação em dispositivos por toque para aplicações tradicionais*: além do aspecto de usabilidade, o mapeamento de interações com dispositivos *touchscreen* para o uso de aplicações convencionais (mouse/teclado) em tempo real também é um desafio

técnico. A técnica utilizada deve interpretar os eventos de interação no dispositivo cliente em mensagens que serão transmitidas e consumidas em tempo real no servidor remoto, convertendo de forma imperceptível ao usuário as operações de toque em movimentação do cursor remoto. Para isso, observou-se que métodos diferentes eram necessários para operações diferentes, por exemplo: um toque deve ser interpretado como um clique de forma absoluta (utilizando a posição absoluta do cursor remoto), para ocorrer o evento do mouse na mesma área remota que o usuário visualizou em seu dispositivo; já a movimentação do toque sobre a tela precisa ser interpretado de forma relativa (a partir da posição relativa à diferença entre o estado inicial do cursor e a distância movimentada), ocorrendo uma movimentação incremental do cursor remoto correspondente a direção e intensidade do evento no dispositivo cliente;

d. *Protocolo de comunicação em tempo real multiplataforma*: basicamente, existem duas formas de comunicação em tempo real em uma arquitetura cliente-servidor, direta e indireta. A primeira possui vantagem quanto a responsividade e latência da comunicação, por ser o mais direta possível, porém, não possui grande escalabilidade, compatibilidade e limita outros tipos de comunicação não um-para-um. A forma indireta, com um intermediário, possui a vantagem de possuir a maior compatibilidade e escalabilidade possível, além de prover a possibilidade de comunicação um-para-muitos e muitos-para-muitos. A técnica indireta utilizada (com um *broker* intermediário), mostrou-se ser possível mitigar o problema da responsividade utilizando-se uma comunicação direta entre as pontas e o intermediário (*socket* entre o servidor e o broker, *websocket* entre o *broker* e o dispositivo cliente), sem perder a característica de compatibilidade multiplataforma;

e. *Simulação de interação para aplicações de terceiros (sem alteração da aplicação)*: a criação das interações mapeadas do dispositivo cliente no servidor é um desafio técnico por ser um problema dependente da forma como o sistema operacional utilizado gerencia e, ao mesmo

tempo, depende do tipo de tratamento realizado pela HPM que se deseja controlar. A técnica utilizada para solucionar esse problema foi a criação de eventos de hardware do cursor compatíveis com as esperadas pelo sistema operacional Microsoft Windows;

Com os experimentos realizados para as contribuições descritas acima foram testados conceitos e técnicas que indicam a capacidade de solucionar os desafios tecnológicos e limitações inerentes à relação HPM / CE.

5.2 Trabalhos Futuros

Como trabalhos futuros, no âmbito das atividades de empreendedorismo iniciadas em conjunto com esta pesquisa, seria interessante realizar adequações aos novos padrões de codificação, VP9[25] e HEVC[26], testes com protocolos de streaming de vídeo experimentais, como o mpeg-dash[27], e codecs como o Mozilla Daala[28].

Para a continuidade da pesquisa científica vislumbra-se o estudo de novas formas de captura da renderização gráfica da HPM, como a utilização de uma técnica de z-buffer ao invés da captura do front-buffer, utilizada neste trabalho. Essa técnica consiste em capturar as alterações interframes das renderizações, compondo assim um quantidade menor de dados para codificação e streaming. Esse estudo aliado a pesquisa de padrões de codificação otimizados para *live streaming* em HTML5, espera-se estender as soluções encontradas nesta pesquisa para os dois tópicos mais importantes do tema: desempenho e compatibilidade.

Outros estudos interessantes para a continuidade desta pesquisa seriam a otimização da escalabilidade, por meio da integração com um hypervisor de

[25] http://www.webmproject.org/vp9/
[26] http://www.mpegla.com/main/PID/HEVC/default.aspx
[27] http://standards.iso.org/ittf/PubliclyAvailableStandards/c057623_ISO_IEC_23009-1_2012.zip
[28] https://www.xiph.org/daala/

virtualização com suporte a vGPU, como o open source XEN[29] ou o proprietário VMware Sphere[30], os quais possuem esse requisito.

[29] http://www.xenproject.org/
[30] *www.vmware.com/br/vSphere*

Referências

[1] VIEL, C. C; MELO, E. L.; GODOY, A. P.; DIAS, D. R. C.; TREVELIN, L. C.; TEIXEIRA, C. A. C. 2012. Multimedia presentation integrating interactive media produced in real time with high performance processing. In: ***Proceedings of the 18th Brazilian symposium on Multimedia and the web*** *(WebMedia '12)*. ACM, New York, NY, USA, 115-122. DOI=10.1145/2382636.2382664.

[2] Onlive, documentos de suporte. http://www.onlive.com/, acessado em 29/03/2014.

[3] KITCHENHAM, B. A.; CHARTERS, S. Guidelines for performing Systematic Literature Reviews in Software Engineering - EBSE Technical Report - No. EBSE-2007-01. (2007).

[4] BASILI, V. R.; SELBY, R. W.; HUTCHENS, D. H. Experimentation in software engineering. Software Engineering, IEEE Transactions on, v. SE-12, n. 7, p.733-743, jul. 1986.

[5] RABELLO, G. P. Transformação de Ambiente Arquitetônico Virtual de Alto Realismo em Objeto HPM, 2013. http:// lince.dc.ufscar.br/monografias/tcc-guilherme-rabelo.pdf. Acessado em 29/03/2014.

[6] LUEBKE, D.; HUMPHREYS, G., How GPUs Work, *Computer*, v. 40, n. 2, p. 96-100, Feb. 2007 DOI=10.1109/MC.2007.59.

[7] NICKOLLS, J.; DALLY, W. J., The GPU Computing Era, *Micro,* ***IEEE***, v. 30, n. 2, p. 56-69, mar./abr. 2010 DOI=10.1109/MM.2010.41.

[8] MEULEN, ROB VAN DER. Purchases of Tablets by Businesses Will Triple by 2016, **Gartner**, November 2012, http://www.gartner.com/newsroom/id/2227215, acessado em 29/03/2014.

[9] USA Government, http://energy.gov/consumption, acessado em 29/03/2013.

[10] CARROL, A.; HEISER, G. An analysis of power consumption in a smartphone, 2010. In: **Proceedings of the 2010 USENIX conference on USENIX annual technical conference** (USENIXATC'10). USENIX Association, Berkeley, CA, USA, 21-21.

[11] MICROSTRATEGY. The New Era of Mobile Intelligence. 2010. http://www.microstrategy.com/mobile/platform/The_Convergence_of_Mobile_Technology_and_Mobile_Intelligence.pdf, acessado em 29/03/2014.

[12] VARSHNEY, U. 4G Wireless Networks. IT Professional , v. 14, n. 5, p. 34-39, 2012. DOI=10.1109/MITP.2012.71.

[13] Nvidia, Tegra 4 Specifications. 2013. http://www.nvidia.com/object/tegra-4-processor.html, acessado em 29/03/2014.

[14] NIEH, J.; YANG, S. J.; NOVIK, N. A Comparison of Thin-Client Computing Architectures. 2000. **NoMachine documentation**. http://www.nomachine.com/documentation/pdf/cucs-022-00.pdf, acessado em 29/03/2014.

[15] NAGELLA, S.; SASTRY, L.; FOWLER, R.; O'BRIEN, J. Remote Rendering on Visualization Cluster using VirtualGL and Chromium. 2008. **VizNet documentation.** http://www.viznet.ac.uk/files/VIZNET-WP4-16-STFC-UsingVizClusters-web.pdf, acessado em 29/03/2013.

[16] Citrix. HDX for Professional Graphics. http://support.citrix.com/servlet/KbServlet/download/23011-102-666023/HDX_3D_Pro_Graphics_1.1_-_Administrator's_Guide_0070.pdf, acessado em 29/03/2014.

[17] Nvidia VCA, http://www.nvidia.com/object/visual-computing-appliance.html, acessado em 29/03/2014.

[18] LAMPE, U.; WU, Q.; HANS, R.; MIEDE, A.; STEINMETZ, R. **To Frag Or To Be Fragged**, 2013. ftp://130.83.198.178/papers/LWH+13.pdf, acessado em 29/03/2014.

[19] VIEL, C. C.; MELO, E. L.; TREVELIN, L. C.; TEIXEIRA, C. A. C. Rv-mtv: Framework para interação multimodal com aplicações de realidade virtual em tv digital e dispositivos móveis. In: **WebMedia** 2011, 2011.

[20] JURGELIONIS, A.; FECHTELER, P.; EISERT, P.; BELLOTTI, F.; DAVID, H.; J. P. LAULAJAINEN, J. P.; R. CARMICHAEL, R.; V. POULOPOULOS, V.; LAIKARI, A.; PER¨AL¨A, P.; DE GLORIA, A.; BOURAS, C. Platform for distributed 3d gaming.Int. J. Comput. **Games Technol**., 2009:1:1–1:15, jan. 2009. DOI=10.1155/2009/231863.

[21] KARACHRISTOS, T.; APOSTOLATOS, D.; METAFAS, D. 2008. A real-time streaming games-on-demand system. In: **Proceedings of the 3rd international conference on Digital Interactive Media in Entertainment and Arts** (DIMEA '08). ACM, New York, NY, USA, 51-56. DOI=10.1145/1413634.1413648.

[22] WEIREN YU; JIANXIN LI; CHUNMING HU; LIANG ZHONG. Muse: a multimedia streaming enabled remote interactivity system for mobile devices, 2011. In: **Proceedings of the 10th International Conference on Mobile and Ubiquitous Multimedia** (MUM '11). ACM, New York, NY, USA, 216-225. DOI=10.1145/2107596.2107624.

[23] ZHOU ZHAO; KAI HWANG; VILLETA, J. Game cloud design with virtualized CPU/GPU servers and initial performance results, 2012. In: **Proceedings of the 3rd workshop on Scientific Cloud Computing Date** (ScienceCloud '12). ACM, New York, NY, USA, 23-30. DOI=10.1145/2287036.2287042.

[24] TORTEROLO, L.; PAPALEO, G.; SCAGLIONE, S.; RUFFINO, F.; AIELLO, M. “3D Cloud” in Life Sciences: An innovative framework for remote 2D/3D visualization and collaboration, **Computer-Based Medical Systems (CBMS)**, 2012 25th International Symposium on, p. 1, 6, 20-22. Jun. 2012. DOI=10.1109/CBMS.2012.6266403.

[25] CHUN-YING HUANG; CHENG-HSIN HSU; YU-CHUN CHANG; KUAN-TA CHEN. GamingAnywhere: an open cloud gaming system. 2013. In: ***Proceedings of the 4th ACM Multimedia Systems Conference*** (MMSys '13). ACM, New York, NY, USA, p. 36-47. DOI=10.1145/2483977.2483981 http://doi.acm.org/10.1145/2483977.

[26] MCCARTHY, J. D.; SASSE, M. A.; MIRAS, D. Sharp or smooth?: comparing the effects of quantization vs. frame rate for streamed video. 2004. In: **Proceedings of the SIGCHI Conference on Human Factors in Computing Systems** (CHI '04). ACM, New York, NY, USA, 535-542. DOI=10.1145/985692.985760.

[27] GODOY, A. P.; VIEL, C. C.; MELO, L. E.; DIAS, D. R. C.; TREVELIN, L. C.; TEIXEIRA, C. A. C. Multimedia Presentation Integrating Media with Virtual 3D Realistic Environment Produced in Real Time with High Performance Processing, **SBC Journal of Interactive Systems** (JIS). 2014 - em processo de publicação.

[28] Comitê Gestor da Internet no Brasil CGI – NIC – SIMET: Sistema de Medição de Tráfego da Internet, http://simet.nic.br/mapas/, acessado em 23/03/2014.

[29] WANG, Y.; OSTERMANN, J.; ZHANG, Y. Video Processing and Communications. Prentice Hall, 2001.

[30] Z. WANG L. LU; BOVIK, A. Video quality assessment based on structural distortion measurement. **SIgnal Processing**: Image Communication, v. 19, n. 2: p. 121-132, fev. 2004.