Proposta de uma Arquitetura Híbrida para Massively Multiplayer Online Games

Mitchel Soni Felske* e Diana F. Adamatti*

Abstract— In this present work it is proposed a hybrid architecture for be used in development of Massively Multiplayer Online Games (MMOGs). This model derived from the server Cluster architecture and aims to reduce deployment and maintenance costs of the physical infrastructure needed to support the game. Thus, this model seeks to withdraw a portion of the load on the servers, in that it updates only a few players, which are responsible for passing on this new information to the others. To validate the solution an experiment on the SIM3D platform was proposed. Comparing the responsiveness times between the proposed architecture with the traditional Client-Server, it was found that them were similar for the players in both models.

***Index Terms* — Distributed Systems, MMOGs.**

***Resumo* — Neste trabalho é proposta uma arquitetura híbrida para ser utilizada no desenvolvimento dos Massively Multiplayer Online Games (MMOGs). Esse modelo é derivado da arquitetura Cluster de servidores e visa reduzir os custos de implantação e manutenção da infraestrutura física necessária para suporte ao jogo. Para isso, ele visa retirar uma parcela da carga dos servidores, na medida em que ele atualiza somente alguns jogadores, sendo esses os responsáveis por repassar essas novas informações para os demais. Para validar a solução, foi realizado um experimento sobre a plataforma SIM3D. Comparando os tempos de resposta obtidos entre a arquitetura proposta e a tradicional Cliente-Servidor, foi constatado que esses foram semelhantes para os jogadores em ambos os modelos.**

***Palavras chave* — Sistemas Distribuídos, MMOGs.**

I. INTRODUÇÃO

Um Massively Multiplayer Online Game (MMOG) pode ser definido como um jogo de simulação online distribuída e de tempo real. Uma de suas principais características é a interação proporcionada aos jogadores no mundo virtual, permitindo que esses desfrutem de diversas oportunidades de interação social, bem como de atividades competitivas, como combate, estratégia, economia, gestão, entre outros. Além disso, são jogos de longa duração e larga escala, suportando uma grande quantidade de jogadores simultâneos [5, 8, 11].

A popularidade desses jogos vem crescendo vertiginosamente nos últimos anos. Em abril de 2008, pelo menos 16 milhões de pessoas ao redor do mundo estavam jogando MMOGs [10]. O sucesso desses jogos fundamenta-se em alguns fatores fundamentais que são:

Os jogadores competem entre si tornando a disputa mais atraente e desafiadora do que o enfrentamento de oponentes virtuais controlados pelo computador. Em jogos monousuários, por mais que as técnicas de Inteligência Artificial empregadas para controlar os personagens virtuais sejam aprimoradas, o comportamento humano simulado diverge em relação à realidade [6];

A interação entre os participantes promove o aparecimento de relações sociais interessantes, as quais não estão presentes em jogos monousuários;

Enquanto que os jogos offline possuem uma duração de algumas horas, um MMOG pode proporcionar aos jogadores experiências que podem durar de alguns meses até vários anos.

Grande parte dos MMOGs faz uso de uma arquitetura do tipo Cliente-Servidor (C-S), na qual o jogador (cliente) é sincronizado com um servidor confiável. Esse servidor normalmente está sob resguardo da empresa desenvolvedora do MMOG para garantir a segurança e a confiabilidade dos dados.

Com base nos dados recebidos dos jogadores, o servidor tem a obrigação de simular o mundo virtual, o que gera uma grande sobrecarga de processamento. Outra limitação imposta por essa arquitetura é a exigência de uma largura de banda elevada no servidor, para que ele possa repassar as atualizações para os clientes em um tempo aceitável para a aplicação.

Assim, para reduzir essas restrições, tem-se estabelecido como uma forte corrente de pesquisa o uso de arquiteturas híbridas – união do C-S com Peer to Peer (P2P), visando descentralizar parcialmente o controle das ações do jogo. Contudo, essa nova abordagem traz consigo novos desafios. Como será realizada a sincronização entre os jogadores a fim de permitir que esses visualizem um único mundo virtual compartilhado? Como ocorrerá a simulação do mundo virtual? Quais tipos de mecanismos serão utilizados a fim de se evitar trapaças?

Apesar do surgimento de inúmeras arquiteturas híbridas, nenhuma delas se estabeleceu como definitiva para ser adotada pelos desenvolvedores de MMOGs para substituir a tradicional arquitetura C-S na produção de seus jogos. Dessa

* Mitchel Soni Felske é engenheiro de Computação pela Universidade Federal de Rio Grande (FURG). Atualmente, é mestrando na Universidade Federal de Santa Catarina (UFSC). E-mail: msfelske@hotmail.com
Diana F. Adamatti é professora doutora do Centro de Ciências Computacionais e do Programa de Pós-Graduação em Modelagem Computacional da Universidade Federal do Rio Grande (C3/PPGMC/FURG). E-mail: dianaadamatti@furg.br

forma, optou-se neste trabalho por desenvolver um novo modelo híbrido derivado da arquitetura Cluster de servidores, tendo como principal objetivo a redução dos custos relacionados com a implantação e manutenção da infraestrutura física necessária para disponibilizar e manter essas aplicações.

Esse modelo visa reduzir a largura de banda necessária nos servidores, na medida em que diminui o número de mensagens que esses devem enviar para repassar uma atualização. Diferentemente da arquitetura C-S, onde os servidores têm a função de repassar as atualizações para todos os jogadores, nesse novo modelo eles atualizam apenas uma parcela desses jogadores, sendo que esses é que deverão atualizar os demais.

O trabalho foi estruturado da seguinte maneira: primeiramente foi realizado um estudo acerca dos MMOGs, com o intuito de compreender melhor o funcionamento dessas aplicações distribuídas, o qual é apresentado na Seção 2. Após, pesquisou-se como as arquiteturas de comunicação C-S, P2P e Híbrída se enquadravam no contexto dos MMOGs. Essa análise é mostrada na Seção 3. Com base nessas informações foi definido o modelo híbrido, o qual foi implementado em um experimento realizado sobre a plataforma SIM3D (Seções 4 e 5). Essa plataforma consiste em um ambiente virtual compartilhado que simula oficinas reais do polo naval.

Por fim, foram definidos testes para comprovar a sua eficácia, os quais são explicados na Seção 6. Os casos de teste visavam comparar os tempos entre o envio de requisições por parte dos jogadores e as suas respostas, entre o modelo proposto e a arquitetura C-S. O objetivo era verificar se esse novo modelo mantinha um tempo de resposta similar ao da arquitetura C-S.

Como foi constatado, esse tempo é semelhante e, portanto, compatível com a aplicação. Desse modo, conforme detalha a Seção 7, essa nova arquitetura tende a ser uma boa alternativa como modelo de comunicação a ser adotado pelos MMOGs. Além disso, essa seção apresenta alguns temas que devem ser explorados em trabalhos futuros.

II. MMOG

Um MMOG é um jogo onde milhares de participantes interagem uns com os outros, em um ambiente específico, através de uma conexão com a Internet entre seus dispositivos computacionais, que são geralmente computadores pessoais ou consoles. Essa interação é ilustrada na Figura 1. Esse ambiente é caracterizado pela simulação distribuída de um mundo virtual persistente em tempo real [5, 11].

Ambientes virtuais são espaços distribuídos, imersivos e realistas, gerados por computador, que permitem a interação simultânea de vários participantes em tempo real [8]. No caso específico dos MMOGs, esses ambientes representam o espaço compartilhado entre os jogadores, ou seja, é a plataforma onde o jogo é realizado.

Figura 1. Interação entre os jogadores de um MMOG [10].

Os MMOGs são caracterizados como um sistema de tempo real, pois os jogadores enviam comandos de forma independente à passagem do tempo de simulação [5, 11]. Por ser uma aplicação sobre um Sistema Distribuído (SD), esses jogos permitem a manipulação concorrente sobre dados compartilhados (objetos e itens que compõem o cenário do jogo).

Dessa forma, é fundamental que o sistema mantenha o estado global consistente entre os jogadores. Assim, toda a ação executada por cada avatar (personagem virtual controlado por um jogador) deve ser processada e repassada para os demais de forma confiável e rápida, possibilitando o cálculo do novo estado do jogo. Essa tarefa é chamada de sincronização. Para realizá-la, os sistemas devem seguir regras que governam a relação existente entre os processos e o repositório de dados a fim de garantir a integridade da informação manipulada.

Essa simulação está em constante execução, independentemente da presença de jogadores. Portanto, diz-se que o mundo virtual é persistente, pois o estado do jogo é gravado e esse pode continuar sem uma perspectiva de encerramento [6,9]. Assim, as alterações no ambiente de jogo são mantidas para todos os jogadores, inclusive aqueles que não estão conectados no presente momento [5].

Ao contrário do que ocorre com a grande maioria das aplicações de simulação de ambientes virtuais compartilhados, cuja interação entre os usuários é colaborativa, o foco dos MMOGs é a competição entre os jogadores. Desse modo, para que um MMOG seja confiável, ele deve ser intolerante a trapaças, ou seja, o sistema de suporte deve prover mecanismos de forma a garantir que as regras do jogo não possam ser burladas por jogadores desonestos para obtenção de vantagens ilegais [6,9].

II-A Elementos de um MMOG

Com o objetivo de facilitar o desenvolvimento de um MMOG, o seu projeto envolve duas etapas. A primeira consiste em determinar os componentes de hardware e redes que irão compor o sistema. Esses fazem parte do que se denomina plataforma (ou infraestrutura) física. A segunda fase consiste na definição de uma arquitetura de comunicação e

outra de controle, as quais constituem a plataforma lógica [6].

A arquitetura de comunicação se caracteriza por definir qual será a topologia de comunicação existente entre os elementos do sistema (computadores dos usuários, servidores, etc.), comumente denominados nodos, permitindo que esses interajam através da troca de mensagens. É por meio dessa relação que ocorrerão as trocas de informações necessárias para a atualização do mundo virtual. Por outro lado, a arquitetura de controle é responsável por ordenar os eventos oriundos dos jogadores, permitindo o cálculo do novo estado do jogo. Além disso, é ela que determina como irá ocorrer o processo de sincronização entre os nodos.

Os problemas que surgem na criação dos MMOGs estão diretamente relacionados com ambas as plataformas, bem como com algumas características próprias dos SD: desempenho, segurança, escalabilidade, confiabilidade e consistência. A preocupação com a consistência, o desempenho e a segurança existem em todos os ambientes virtuais multijogador. Por outro lado, outros entraves vinculados com a escalabilidade, a persistência e a confiabilidade, proveem exclusivamente de aplicações massivas [8], devido ao fato dessas possuírem um elevado número de participantes e o ambiente virtual estar em constante simulação.

A largura de banda dos jogadores e servidores que compõem um MMOG influi diretamente em sua escalabilidade. Conforme ocorre o aumento do número de jogadores simultâneos, o número de mensagens trocadas para que todos eles possam ter o estado do jogo atualizado também se eleva. Assim, o sistema deve possuir uma largura de banda adequada de forma que a aplicação possa incorporar mais jogadores sem uma perda significativa de desempenho.

Contudo, por mais que se utilizem equipamentos de última geração, sempre existirão restrições para compor a plataforma física, as quais não podem ser totalmente eliminadas devido à limitação tecnológica existente. Portanto, deve-se possuir uma preocupação maior com a escolha da plataforma lógica, visto que é através dela que se pode atenuar os efeitos prejudiciais que a plataforma física acarreta e, também, obter um ganho de desempenho considerável para a aplicação [6, 9].

III. Análise Comparativa entre as Arquiteturas de Comunicação

A Tabela 1 traça um comparativo entre as principais arquiteturas de comunicação possíveis de serem utilizadas no desenvolvimento de um MMOG. Como pode ser observado, a arquitetura P2P é a que apresenta o menor custo para implementação e manutenção, devido à simulação ser realizada pelos próprios jogadores. Contudo, esse mesmo fator diminui a escalabilidade e a confiabilidade do sistema e aumenta a complexidade do mecanismo de sincronização.

Por outro lado, o modelo híbrido, por possuir um sistema de replicação de servidores, apresenta uma confiabilidade maior que o C-S, cujo único servidor existente torna-se um único ponto de falha, deixando o sistema mais vulnerável.

	C-S	P2P	Híbrida Cluster de Servidores
Escalabilidade	Média	Baixa	Média
Confiabilidade	Baixa	Baixa	Alta
Custo Implementação / Manutenção	Alto	Baixo	Alto
Mecanismo de Sincronização	Simples	Complexo	Mediano

Tabela 1. Nível de desempenho atingido pelas arquiteturas de comunicação com relação aos requisitos que devem estar presente em qualquer SD.

Além de avaliar as arquiteturas de comunicação quanto a sua eficiência sob os aspectos acima mencionados, é imprescindível analisá-las com relação a algumas características peculiares dessa aplicação, as quais estão resumidas na Tabela 2.

	C-S	P2P	Híbrida Cluster de Servidores
Funcionalidades Administrativas	Baixo	Alto	Baixo
Controle de Trapaças	Baixo	Alto	Baixo
Persistência do Ambiente	Baixo	Alto	Baixo

Tabela 2. Grau de dificuldade que a arquitetura de comunicação adotada impõe para a implementação das funcionalidades de um MMOG [1, 5, 9].

Analisando esses dados, nota-se que a arquitetura P2P, devido a seu caráter descentralizado, possui uma alta complexidade para tratar o gerenciamento das funcionalidades mencionadas na Tabela 2, pois a simulação do jogo é realizada pelos próprios jogadores. Por outro lado, tanto a arquitetura C-S como a híbrida, devido a possuírem um elemento centralizador (servidor) conseguem implementá-las de maneira mais simples.

Assim, pode-se constatar que a grande vantagem do modelo C-S é o baixo nível de dificuldade que ele gera para a implementação do MMOG, enquanto que o potencial de uma arquitetura P2P é o baixo custo de implantação e manutenção da plataforma física. Porém, o primeiro possui baixa confiabilidade devido ao fato de possuir um único ponto de falha e o segundo é muito complexo para ser implantado em jogos de larga escala.

Portanto, pode-se inferir que um modelo híbrido, que visa agregar as potencialidades de ambas as arquiteturas, pode vir a ser uma alternativa para substituir o tradicional modelo C-S como componente da plataforma lógica de um MMOG.

Na seção a seguir é apresentada a arquitetura de comunicação desenvolvida a partir do modelo híbrido Cluster de Servidores.

IV. ARQUITETURA PROPOSTA

A arquitetura híbrida proposta é derivada do modelo Cluster de servidores. Ela possui como principal objetivo diminuir o gasto excessivo relacionado com a implantação e, principalmente, com a manutenção dos componentes de hardware que constituem a plataforma física, o qual é a principal deficiência desse modelo. Essa redução permite que pequenas empresas possam entrar nesse ramo de aplicações para competir em condições de igualdade com as organizações que já dominam esse mercado. Além disso, essa competição promoverá melhorias cada vez maiores nos MMOGs, os quais beneficiarão os jogadores.

Existem duas alternativas que permitem atingi-lo: diminuir a carga de processamento dos servidores, fazendo uso de elementos descentralizadores da arquitetura P2P para auxiliar na simulação, ou reduzir os seus custos relacionados com os recursos de rede.

Este trabalho foca a segunda alternativa. Para diminuir a carga de rede espera-se reduzir o número de mensagens que o servidor deve enviar para os jogadores quando se fizer necessário repassar uma atualização. A ideia geral é fazer com que alguns jogadores se tornem responsáveis por atualizar um conjunto de outros jogadores, isentando o servidor dessa responsabilidade.

A arquitetura proposta pode ser dividida em três camadas, conforme exposto na Tabela 3.

.

1	PEERS SERVIDORES
2	CLIENTE-SERVIDOR
3	PEERS CLIENTES

Tabela 3. Camadas da arquitetura de comunicação proposta.

A primeira camada representa a comunicação P2P entre os servidores do jogo. É através dela que ocorre a troca de informações entre os servidores para a atualização das entidades. O segundo nível representa a comunicação entre jogadores e servidores, a qual é feita segundo o clássico C-S. Os jogadores são capazes de enviar requisições de atualização somente para os servidores os quais estão conectados. Contudo, após o servidor realizar o processamento do novo estado do jogo, ele enviará as informações atualizadas somente para alguns jogadores, os quais são denominados ClientPeers. Eles constituem a terceira camada e foram assim designados com o propósito de indicar que além de clientes, esses jogadores atuam como peers, na medida em que eles são responsáveis por repassar as atualizações, através de uma arquitetura P2P, para os demais usuários que não as receberam diretamente do seu servidor base. Esses usuários, por atuarem apenas como clientes, ora enviando requisições para o seu servidor, ora recebendo atualizações dos ClientPeers, são sugestivamente chamados de Clients.

Esse modelo pode ser melhor compreendido observando-se a sua representação na Figura 2. Nela, os servidores são representados por círculos com a inscrição Sy, onde y é o número que identifica cada servidor. Esse número pode variar de 1 até a quantidade máxima de m, a qual é definida de acordo com a demanda de usuários. Os jogadores, por sua vez, são divididos em dois grupos, conforme explicado anteriormente: os ClientPeers (CPx) e os Clients (Cx), onde x representa uma identificação única para cada jogador. Cada servidor pode ter um número máximo n de usuários conectados a ele, o qual é definido de acordo com a sua capacidade.

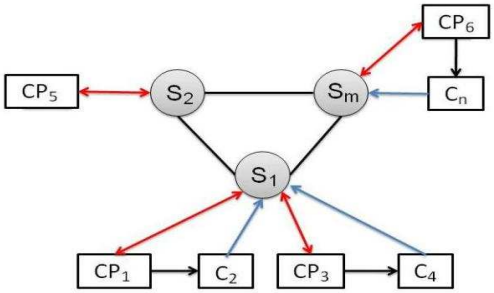

Figura 2. Representação hipotética da arquitetura híbrida proposta.

O número máximo de ligações com Clients que um jogador do tipo ClientPeer pode realizar depende da capacidade de hardware da máquina que esse usuário está utilizando, bem como da sua largura de banda disponível. Contudo, deve-se ter cuidado para não sobrecarregar os recursos desse usuário, evitando-se que a qualidade do jogo seja comprometida. Assim, o sistema gerenciador do MMOG deve possuir um mecanismo para detectar quais jogadores possuem maior potencial para se tornarem ClientPeers de acordo com a disponibilidade de seus recursos de hardware.

IV-A Algoritmo de Sincronização

A singularidade desse modelo traz consigo a necessidade de elaboração de um algoritmo de sincronização para manter o ambiente virtual consistente. O algoritmo proposto leva em conta a existência de dois tipos de bancos de dados: um Banco de Dados Global (BDG) e um Banco de Dados Local (BDL). O BDG é gerenciado pelos servidores e contém todas as entidades do jogo. Ele sempre conterá a cópia mais atual de cada entidade presente no mundo virtual, sendo, desse modo, consultado pelos jogadores quando esses realizarem pedidos de atualização para os servidores. Por outro lado, o BDL contém uma cópia local das entidades visíveis por cada participante no seu ambiente virtual. Dessa forma, existe um único BDG e várias instâncias do BDL, uma para cada jogador.

Além disso, cada entidade do BDG tem associado a ela um Contador Global (CG). Da mesma forma, cada cópia do objeto nas respectivas BDLs possui um Contador Local (CL). Esses contadores determinam o tempo lógico, ou seja, a versão de cada entidade do jogo. Para determinada entidade, valores

de CL no BDL de cada cliente iguais ao do CG do BDG do servidor, indicam que esses jogadores possuem a versão mais atual daquela entidade. Valores abaixo do CG, determinam que esses usuários ainda estão esperando pelo recebimento das cópias mais atualizadas daquela entidade.

O algoritmo funciona da seguinte maneira: quando um usuário necessitar atualizar um objeto, como pegar uma arma que estava no chão, ele deve enviar uma consulta para o seu servidor a fim de verificar se essa ação pode ser realizada. Caso o servidor constate que essa operação é válida, ou seja, que a CL do jogador é igual a CG do BDG no servidor, ele realiza o processamento do novo estado e o repassa para os demais servidores os quais ele está conectado. Esses servidores, por sua vez, repassarão a entidade atualizada para os ClientPeers os quais eles estão associados. Por fim, esses ClientPeers atualizarão os Clients que estão sob seu encargo. Assim, o jogador que solicitou e todos os demais que necessitam dessa informação irão recebê-la e atualizarão as suas respectivas BDLs.

Por outro lado, se aquela ação não puder ser realizada, o servidor envia uma mensagem para o jogador informando que o pedido foi rejeitado. Isso irá ocorrer quando o CL do jogador for inferior ao CG do referido objeto. Isso significa que a cópia local estava desatualizada no momento da tentativa de atualização, de forma que o jogador ainda não possui a cópia com a última modificação realizada naquele objeto e, portanto, não pode alterá-lo.

Isso garante que o usuário não irá manipular uma entidade que já foi alterada por outro jogador sem que ele receba essa atualização, impedindo, por exemplo, que dois usuários tenham a posse do mesmo objeto concomitantemente, o que claramente representaria uma inconsistência. Voltando ao exemplo anterior, isso evita que um jogador pegue a arma que estava no chão na sua visão do cenário, mas que na verdade já havia sido obtida por um adversário.

Esse algoritmo assegura a coerência do mundo virtual quando cada jogador tenta atualizar uma entidade por vez. Contudo, ele não garante a consistência quando vários usuários tentam atualizar um conjunto de entidades simultaneamente. Para resolver este problema, deve-se inserir um mecanismo de controle de concorrência, garantindo que as transações dos usuários sejam executadas de uma forma segura e seguindo as propriedades ACID (Atomicidade, Consistência, Isolamento e Durabilidade) [4]. Além disso, deve-se possuir um mecanismo para garantir a consistência entre os servidores, o que também não foi implementado.

Visto que o objetivo principal do trabalho é realizar uma análise da viabilidade de uso da arquitetura elaborada em MMOGs, a solução desses desafios não será proposta neste trabalho. Primeiramente, optou-se por verificar o seu desempenho em condições ideais, ou seja, admitindo que não haja conflitos entre os jogadores. Caso os resultados sejam positivos, essas e outras adversidades explicitadas mais adiante, poderão ser tratadas.

V. IMPLEMENTAÇÃO

V-A A Plataforma SIM3D

O projeto SIM3D, desenvolvido pelos alunos do Centro de Ciências Computacionais (C3) da Universidade Federal do Rio Grande – FURG – em parceria com a Universidade Federal do Rio de Janeiro (UFRJ), surgiu com o objetivo de permitir a visualização de plantas e processos de manufatura industrial, mais especificamente do segmento naval, em realidade virtual, através de diferentes meios, por exemplo, Cave Automatic Virtual Enviroment (Cave), computadores, TV 3D e celulares [3].

Com isso, oficinas reais podem ter uma representação digital, que visa reproduzir fielmente as condições encontradas no ambiente real [Figura 3]. Cada usuário possui um personagem virtual que pode se movimentar livremente sobre esse cenário digital, podendo observá-lo sob qualquer ângulo desejado, além de poder realizar alterações sobre as entidades ali representadas. Esses cenários são compartilhados e podem ser modificados cooperativamente, permitindo que os participantes, durante a navegação, troquem informações com o objetivo de resolver os problemas ali observados [2].

Baseando-se nessas características, pode-se constatar que a plataforma SIM3D se assemelha muito aos MMOGs por se tratar de uma aplicação que simula um ambiente virtual instanciado para vários usuários geograficamente distribuídos. Da mesma forma que nesses jogos, essas instâncias dos cenários virtuais devem ser mantidas sincronizadas, pois essa interação concorrente traz consigo problemas para a manutenção da consistência global das informações. Assim, devem existir mecanismos para tratar da sincronização dos dados entre as diferentes instâncias da aplicação, visando garantir a coerência dos dados do mundo virtual.

Essa similaridade entre o princípio de funcionamento de ambos, torna possível a utilização do SIM3D como base para a elaboração de um experimento que visa implementar, nessa plataforma, a arquitetura híbrida proposta na Seção 4 e analisar a sua performance.

A grande diferença entre ambas as aplicações está no fato de que os MMOGs são jogos onde a interação entre os usuários possui um caráter competitivo, enquanto o SIM3D visa à colaboração entre os participantes. Essa característica dos MMOGs acarreta em um aumento da complexidade para gerenciar a sincronização entre os usuários, conforme citado no final da Seção 4.1. Contudo, como o escopo do trabalho é analisar o comportamento e o desempenho da arquitetura de comunicação proposta para esses jogos, essa peculiaridade não interfere no experimento proposto.

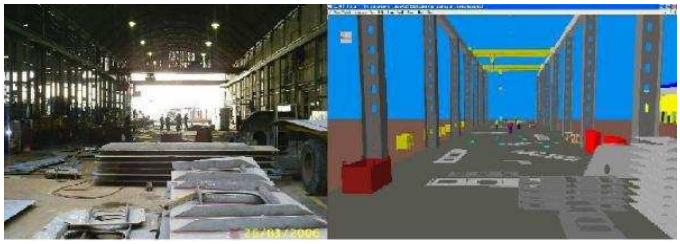

Figura 3. Oficina real e oficina digital [2].

A arquitetura da plataforma pode ser dividida em três grandes etapas: modelagem, conversão e visualização. Carvalho e Oliveira (2009, pág. 17) definem que "na modelagem ocorre a concepção do modelo a ser visualizado, a conversão executa ações a fim de preparar o modelo, compatibilizando-o e otimizando-o para a visualização e, por fim, a visualização se encarrega dos aspectos de renderização, multiprojeção, interatividade e simulação física."

Para o experimento proposto, o processo de visualização é o mais importante, pois em jogos a interatividade e a simulação física são vitais. A interatividade é o processo de troca de informações entre o usuário e o sistema. É necessário saber que tipos de dados vão ser manipulados entre os dispositivos dos usuários e a aplicação. Conforme Carvalho e Oliveira (2009), de posse dessas informações, a simulação física, que tem por finalidade dar realismo ao ambiente virtual, detecta e soluciona possíveis colisões entre as entidades modificadas, de acordo com a política de sincronização adotada.

V-B Experimento realizado sobre a plataforma SIM3D

A linguagem de programação C++ foi adotada para a realização deste experimento visando facilitar a comunicação com os demais componentes do SIM3D, visto que essa plataforma foi desenvolvida nessa linguagem. O diagrama da Figura 4 apresenta o modelo conceitual da plataforma SIM3D. Nela podem ser visualizadas as classes que compõe o sistema implementado, suas generalizações e especializações, bem como o processo de interação entre elas. O foco será mantido na classe "Synchronism", onde será desenvolvido o código referente ao estudo de caso.

Para realizar essa tarefa, foi necessário aprofundar o conhecimento sobre o funcionamento geral da plataforma e, mais especificamente, do funcionamento da classe "EntityState". Isso se fez necessário, pois é nessa classe que está definida a estrutura de representação dos objetos do cenário, bem como os métodos para manipulá-los, os quais serão utilizados pelo mecanismo de sincronização desenvolvido para garantir a coerência entre as diversas instâncias desses objetos. Além disso, adicionou-se nessa classe o atributo "versão", o qual corresponde ao CG para os servidores e ao CL para os jogadores.

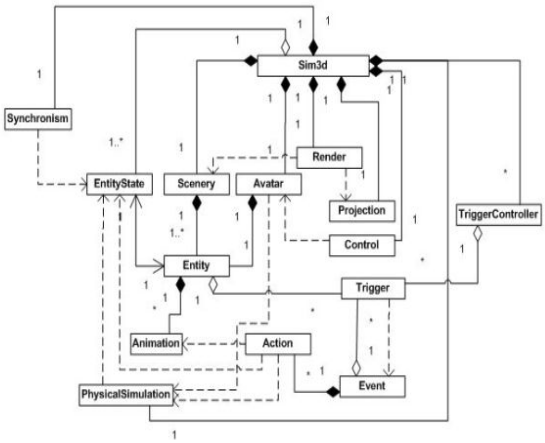

Figura 4. Modelo Conceitual da Plataforma SIM3D.

Um grande desafio encontrado para realizar essa implementação foi definir como iria ocorrer a troca de informações entre os elementos do sistema. A maneira mais fácil de fazê-lo é serializando os dados em uma sequência de bytes, os quais pudessem ser decodificados no receptor. Essa tarefa, complexa de ser realizada, foi resolvida utilizando o Protocol Buffers [7]. Essa API, desenvolvida pelo Google, é uma linguagem de descrição de interface que permite a serialização de diferentes estruturas de dados. A partir da definição prévia das estruturas de dados necessárias para aplicação, o Protocol Buffers gera automaticamente todas as funções que permitem serializar e, posteriormente, recuperar esses dados, automatizando e facilitando esse processo.

VI . TESTES E RESULTADOS

Como resultado deste experimento, espera-se uma diminuição do uso da largura de banda dos servidores, bem como a manutenção do tempo de resposta em um nível que não comprometa a aplicação. Dessa forma, para avaliar o desempenho da arquitetura híbrida proposta, foram desenvolvidos dois casos de teste que visam comparar os tempos de atualização entre os jogadores.

Outra maneira utilizada para verificar a performance desse modelo foi confrontá-lo com o tradicional C-S. Para viabilizar essa comparação, o experimento realizado foi adaptado de forma que, além da arquitetura proposta, ele implementasse o modelo C-S. Caso os tempos de atualização entre os jogadores sejam similares em ambas as arquiteturas, fica comprovado a viabilidade do uso desse protocolo de comunicação híbrido para o desenvolvimento de MMOGs. Porém, outros fatores relacionados à aplicação distribuída, como tolerância a falhas, segurança e consistência, deverão ser tratados para garantir que a aplicação seja executada corretamente, propiciando uma experiência agradável aos jogadores.

Para ambos os casos de teste adotou-se uma máquina servidora, localizada em uma sub-rede, com as seguintes configurações: processador Intel Core2Duo T8100 de 2.10 GHz, 4 GB de memória RAM e placa de rede Realtek

RTL8102E Fast Ethernet. Além disso, foram usados outros três computadores, lotados em outra sub-rede, que possuem um processador AMD Phenom II X4 945 AM3 3.0 GHz, 4 GB de memória RAM, os quais simulam os usuários que estão rodando a plataforma SIM3D sobre o mesmo ambiente virtual.

Conforme Bezerra (2009), o intervalo médio de atualização dos jogadores em um MMOG, ou seja, o tempo entre requisições para atualização de uma entidade, é de cerca de 250 ms. Assim, para tornar o teste mais verossímil com os jogos reais, adotou-se esse mesmo intervalo de tempo para o envio dos pedidos de atualização de cada cliente. Em ambos os testes desenvolvidos, cada participante enviou 1000 requisições de forma intermitente de acordo com esse intervalo estipulado. Para cada pedido foi armazenado, em um arquivo de log, o tempo em que ele foi enviado, bem como o tempo de chegada da mensagem com a entidade atualizada.

O cenário 1 representa um teste com um servidor e três jogadores seguindo o modelo híbrido proposto. Conforme ilustrado na Figura 5, o servidor, denotado por S, é responsável por atualizar o jogador CP1, que por sua vez deve repassar as atualizações para o usuário C1 e C2.

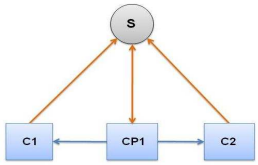

Figura 5. Caso de teste 1.

O gráfico da Figura 6 mostra o tempo decorrente desde o envio de um pedido de atualização até a resposta para cada requisição dos jogadores CP1, C1 e C2, bem como a média desses tempos. A Tabela 4, a qual complementa essas informações, exibe o número de atualizações efetuadas por cada jogador, o tempo médio de atualização de uma entidade, bem como o desvio padrão obtidos como resultado desse teste.

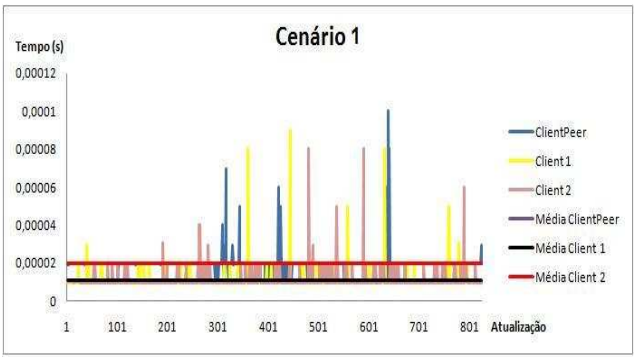

Figura 6. Relação entre o tempo de requisição e de atualização dos jogadores CP1, C1 e C2.

	Quantidade de atualizações efetuadas	Média	Desvio Padrão
CP1	991	20 µs	20 µs
C1	878	10 µs	10 µs
C2	976	20 µs	20 µs

Tabela 4. Resultados obtidos para o teste 1.

Com base no gráfico da Figura 6, é possível constatar que todos os clientes tiveram alguns picos com relação ao tempo de resposta. Essa oscilação é normal, podendo ser causada seja por um congestionamento da rede ou por outros fatores externos, como o hardware do usuário. Além disso, o tempo médio do CP1 não aparece visualmente no gráfico, pois ele foi sobreposto pelo tempo do C2 que foi exatamente igual.

Fazendo uma análise dos dados da Tabela 4, nota-se que a o tempo de atualização dos jogadores se manteve baixo ao longo das atualizações, comprovado pelo baixo desvio padrão apresentado por todos eles. Isso pode ter sido ocasionado por um momento de estabilidade da rede no período de testes, corroborando para esses resultados. Isso impossibilita uma avaliação mais precisa da performance da arquitetura híbrida, pois, nesse caso, era esperado que o jogador CP1 tivesse um tempo médio de atualização menor que os jogadores C1 e C2, uma vez que ele recebe os dados diretamente do servidor, enquanto que os demais jogadores recebem somente do CP1.

Uma informação importante que pode ser extraída da Tabela 4 diz respeito ao número de atualizações efetuadas pelos jogadores. Todos tiveram êxitos similares para concluírem suas requisições, o que era esperado, pois todos tiveram tempos médios de atualização semelhantes. Desse modo, quando o pedido de cada jogador chegou para o servidor, ele já possuía a versão mais atual daquele objeto e, portanto, de acordo com a política de sincronização adotada na Seção 4.1, ele está apto para atualizar aquela entidade.

O segundo caso de teste também representa um cenário com um servidor e três jogadores, porém foi utilizada a topologia de comunicação C-S. Nesse caso, conforme pode ser observado na Figura 7, o servidor S recebe os pedidos de atualizações de todos os jogadores (C1, C2 e C3), realiza a simulação do mundo virtual e repassa o novo estado do jogo para todos participantes.

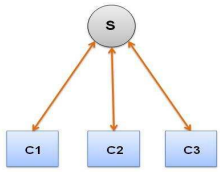

Figura 7. Caso de teste 2.

O gráfico da Figura 8 mostra a variação de tempo existente no decorrer das atualizações dos clientes. Como pode ser observado, o jogador C2 teve uma oscilação muito maior que os demais jogadores, de forma que o tempo para tratar as suas requisições foi maior do que para os demais clientes praticamente durante todo o tempo da medição. Assim, observando o gráfico, fica difícil de visualizar os tempos para

os demais jogadores, visto que eles estão sobrepostos pela curva do jogador mais lento.

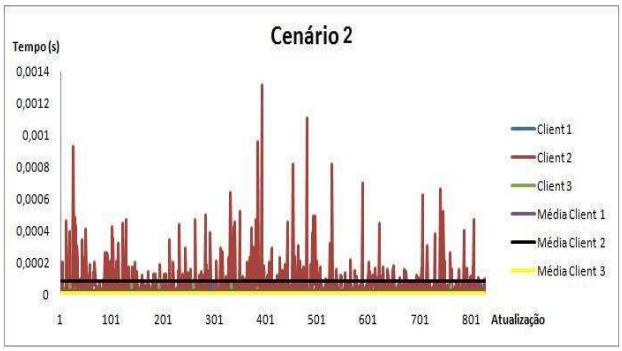

Figura 8. Relação entre o tempo de requisição e de atualização dos jogadores C1, C2 e C3.

	Quantidade de atualizações efetuadas	Média	Desvio Padrão
C1	991	20 µs	10 µs
C2	829	83 µs	129 µs
C3	995	10 µs	6 µs

Tabela 5. Resultados obtidos para o teste 2.

Esse fato fica comprovado pelo alto desvio padrão do jogador C2, diferentemente do que ocorreu com os jogadores C1 e C3, os quais foram bem baixos, conforme mostrado na Tabela 5. Essa discrepância entre esses clientes pode ter sido ocasionada por uma instabilidade da rede ou por um fator externo relacionado, por exemplo, ao computador do cliente C2.

Além disso, essa média elevada para o jogador C2 impossibilitou que ele realizasse uma quantidade de atualizações similares aos jogadores C1 e C3. Por esse cliente ser mais lento, algumas requisições enviadas por ele foram rejeitadas pelo servidor, visto que ele ainda não havia recebido a última versão atualizada daquela entidade e, portanto, não estava apto a realizar aquela operação.

VII. Conclusões e Trabalhos Futuros

Nos últimos anos, os MMOGs têm se consolidado no mercado de games. O número de adeptos para esses jogos tem aumentado consideravelmente, o que impulsiona as empresas desse ramo a aumentarem os investimentos nessa área. Isso beneficia os jogadores, visto que essa maior concorrência obriga os desenvolvedores a se dedicarem em fazer jogos cada vez melhores para manterem seus clientes.

Assim, neste trabalho foi realizado um estudo sobre o uso das arquiteturas de comunicação no desenvolvimento de MMOGs. A arquitetura de comunicação C-S, comumente utilizada no desenvolvimento dessas aplicações, possui suas limitações. O grande fator que pesa contra a sua utilização está relacionado com o custo para a implantação e, principalmente, manutenção de sua plataforma física.

Com base no estudo realizado neste trabalho sobre os modelos de comunicação utilizados no desenvolvimento de MMOGs, conclui-se que o modelo híbrido tende a ser uma boa alternativa a ser adotada na criação desses jogos. Ele pode beneficiar tanto os jogadores, melhorando aspectos relacionados com o fato de a aplicação ser distribuída, como as empresas, reduzindo os custos que essas possuem para implantá-los.

Após essa constatação, optou-se por desenvolver uma arquitetura de comunicação híbrida derivada do modelo Cluster de servidores, visando, principalmente, a redução dos custos de implantação e manutenção da plataforma física. O objetivo principal foi buscar um mecanismo para reduzir a carga de rede dos servidores. A solução proposta consiste em utilizar os clientes para auxiliar os servidores na tarefa de repasse das atualizações.

A conclusão a que se chegou foi que a arquitetura híbrida proposta parece não prejudicar os jogadores em relação ao tempo de recebimento das atualizações. Conforme os testes realizados, os tempos de resposta dos clientes ficaram próximos aos dos jogadores que utilizaram o modelo de comunicação C-S, confirmando as expectativas. Porém, jogadores que levaram maior tempo para serem atualizados, tiveram maior dificuldade para conseguir concretizar as suas requisições, devido à política de sincronização adotada. Essas restrições podem ser contornadas através de modificações no algoritmo de sincronização.

Apesar das respostas positivas, os testes realizados possuem limitações. Não foi utilizado nenhum mecanismo para medir a carga de rede do servidor, o que apenas nos permite inferir que essa carga foi reduzida, uma vez que alguns jogadores ficaram responsáveis por repassar as mensagens de atualização, auxiliando o servidor nessa tarefa. Também não é possível garantir que o modelo híbrido proposto é viável, pois não foi realizado nenhum teste com um número de jogadores compatível com uma situação real.

Assim, para trabalhos futuros são propostos mais testes com o objetivo de se chegar a um argumento mais irrefutável sobre a capacidade desse modelo apresentado servir como arquitetura base no desenvolvimento de MMOGs. Para isso, pretende-se utilizar um software de simulação, onde possa ser colocado no ambiente de jogo um número de jogadores compatível com uma situação real e até mesmo inserir interferências na rede, permitindo medir e verificar a carga do servidor e o desempenho da aplicação nos clientes. Isso auxiliaria na obtenção de resultados melhores e mais confiáveis.

Referências

[1] Bezerra, C. E. B. (2009). “Lidando com recursos escassos e heterogêneos em um sistema geograficamente distribuído atuando como servidor de MMOG.” Dissertação de Mestrado, Programa de Pós-Graduação em Computação da Universidade Federal do Rio Grande do Sul – UFRGS, Porto Alegre – RS.

[2] Carvalho, J. T., Filho, N. L. D., de Botelho Marcos, P., de Queiroz Maffei, R., Oliveira, R. R., Botelho, S. C., e Hax, V. A. (2009). “An automated plataform for immersive and collaborative visualization of industrial models.” International Conference Engineering of Complex Computer Systems – ICECCS, p. 258-264.

[3] Carvalho, J. T. e Oliveira, R. R. (2009). “Uma plataforma para visualização de cenários 3D animados em realidade virtual.” Trabalho de Graduação, Universidade Federal do Rio Grande – RS

[4] Chapple, M. (2011). “The ACID Model”. Disponível em: http://databases.about.com/od/specificproducts/a/acid.htm Acesso em Dezembro de 2011

[5] Cecin, F. R. (2009). “Peer-to-Peer and cheat-resistant support for Massively Multiplayer Online Games.” Tese de Doutorado, Programa de Pós-Graduação em Computação da Universidade Federal do Rio Grande do Sul – UFRGS, Porto Alegre – RS.

[6] Cecin, F. R. e Trinta, F. (2007). “Jogos Multiusuários Distribuídos.” SBGames: Simpósio anual da Comissão Especial de Jogos e Entretenimento Digital da SBC.

[7] Google (2011). “Google Protocol Buffers.” Disponível em: http://code.google.com/p/protobuf/. Acesso em Dezembro de 2011.

[8] Hu, S.-Y. e Liao, G.-M. (2004). “Scalable Peer-to-Peer networked virtual enviroment.” 3rd ACM SIGCOMM workshop on Network and system support for games, p. 129-133.

[9] Rabello, S. A., Cecin, F. R., Barbosa, J. L. V., e Geyer, C. F. R. (2006). “Modelo de comunicação híbrido para jogos massivos multijogador.” SBGAMES 2006.

[10] World of Craft (2010). Disponível em: http://www.worldofwarcraft.com. Acesso em Dezembro 2010.

[11] Woodcock, B. S. (2008). “An analysis of MMOG subscription growth.” Technical report, Disponível em mmogchart.com. Acesso em Dezembro de 2011.