Uma Proposta de *Middleware* de Comunicação para Jogos Multijogador Online com Enfoque no Projeto Subverse

Saulo Popov Zambiasi[1]
[1]UFSC – PPGEAS
Florianópolis, SC, Brasil

Fernando Costenaro Silva[2]
[2]UFSC – POSMEC
Florianópolis, SC, Brasil

Resumo

O Projeto Subverse é uma plataforma de aprendizado em constante evolução, para que alunos de cursos de computação possam trabalhar na prática diversas teorias apresentadas nas disciplinas. Este se apresenta como um mundo virtual para a implementação de agentes baseados no paradigma de jogos multijogador, necessitando de uma estrutura distribuída para múltiplas conexões de agentes remotos para interação entre eles. Contudo, pelo motivo dos agentes trabalharem remotamente, há a necessidade de um controle da troca dessa mensagens e das múltiplas conexões. Para facilitar o desenvolvimento desses agentes, sem que o desenvolvedor precise se preocupar com detalhes de comunicação, o desenvolvimento de um *middlewares* se apresenta como um recurso bastante interessante. Nesse contexto, este artigo apresenta uma proposta de *middleware* de comunicação para o Projeto Subverse, no estilo de jogos multijogador online.

Palavras-chave: *middleware*, comunicação, jogos digitais multijogador.

Contato dos Autores:
{saulopz[1],costenaro[2]}@gmail.com

1. Introdução

O Projeto Subverse é um ambiente de aprendizado com o objetivo de apresentar à alunos de cursos de computação não apenas um produto final, na forma de um jogo de computador online multijogador, em que o aluno implementa os comportamentos dos elementos do jogo, mas uma plataforma em constante evolução. Por meio dessa plataforma os alunos podem trabalhar na prática diversas teorias apresentadas em disciplinas de computação, tais como Inteligência Artificial, Algoritmos, Desenvolvimento de Jogos de Computador, Sistemas Distribuídos e outras [Zambiasi et al. 2010].

O Sistema Subverse, ou apenas Subverse, são as partes implementadas de um mundo virtual para a implementação de agentes baseados no paradigma de jogos multijogador que trabalham em rede [Benin 2008; Silva 2008]. Dessa forma, um conjunto de implementações de módulos do Subverse, e módulos não desenvolvidos, se mostram como desafios aos estudantes para a constante evolução, implementação e melhorias baseadas em teorias computacionais. O Projeto Subverse também sugere a utilização de tecnologias *opensource*, disponíveis de forma gratuita e com recursos bastante atuais [Zambiasi et al. 2009].

Em tempo, o Projeto Subverse sugere uma estrutura distribuída para múltiplas conexões de agentes remotos para interação entre eles no mundo virtual do Subverse [Zambiasi et al. 2009], tal como um MMOG (*Massive Multiplayer Online Game*). Contudo, no lugar de jogadores controlando avatares remotamente, os elementos que devem controlar os avatares são agentes desenvolvidos pelos usuários. Os usuários iniciam seus agentes e visualizam seu ciclo de vida no jogo por meio de uma interface gráfica que também se conecta remotamente ao jogo.

Em jogos MMOG, os clientes são utilizados pelos jogadores para se conectar ao servidor do jogo. Dessa forma, o jogador pode enviar informações da ação que deseja realizar e recebe informações atualizadas das atividades que estão ocorrendo à volta do seu personagem (avatar) no jogo. O servidor possui o estado geral do mundo do jogo e coordena a atividade dos jogadores [Balan et al. 2005]. Muitos desses jogos trabalham baseados no paradigma cliente-servidor. Neste estilo os clientes trocam mensagens com um servidor, que é a parte responsável pelo processamento do jogo [Dickey et al. 2004]. O servidor, em execução, abre um soquete de comunicação e fica aguardando clientes se conectarem. Um cliente estabelece uma conexão com o servidor por meio de um endereço IP (*Internet Protocol*) e uma porta, solicitando a conexão [Silberschatz et al. 2001].

A utilização de soquetes é uma ótima escolha para se trabalhar com a comunicação de processos por ser um método rápido e simples. Contudo, se por um lado, a sua utilização agiliza o processo da comunicação entre os processos, por outro, dificulta pela necessidade de se tratar as mensagens em um baixo nível. Sendo assim, para facilitar a utilização de soquetes, é interessante a utilização de um *middleware* de comunicação, a fim de promover a modularidade e maior facilidade na programação de sistemas distribuídos. Por meio desse recurso, *"o desenvolvedor não precisa se preocupar em escrever o código para gerenciar interações entre os processos"* [Silva 2008].

Além de facilitar ao desenvolvedor, os *middlewares* de comunicação buscam fornecer portabilidade, transparência, interoperabilidade aos sistemas distribuídos e ocultam detalhes de implementações [Silva 2008]. Para Coulouris et al. [2001], tal camada busca mascarar a heterogeneidade

dos sistemas, fornecendo um conjunto de funções para a troca de mensagens, dispensando-o da preocupação de como esta vai ser enviada ou recebida.

Pelo motivo dos agentes no Subverse trabalharem remotamente, há então a necessidade de um controle de troca de mensagens e uma forma de permitir múltiplas conexões. Desse modo, para facilitar o desenvolvimento desses agentes, sem que o desenvolvedor precise se preocupar com os detalhes da comunicação, este artigo apresenta uma proposta de *middleware* de comunicação para o Subverse. A proposta é baseada no paradigma cliente-servidor, e conexões via soquetes, com troca de mensagens. As mensagens são as representações das chamadas de métodos com uma estrutura de variáveis, com os nomes, tipos e valores, além do nome do método que está sendo chamado.

O presente artigo está organizado da seguinte forma: Na seção 2 são apresentados trabalhos relacionados ao assunto de jogos multijogadores onlines e *middlewares* de conexão. A seção 3 apresenta a proposta do *middleware* propriamente dita. Na seção 4 são apresentadas as considerações finais.

2. Trabalhos Relacionados

Feijó e Kozovitz [2003] apresentam uma proposta de comunicação para jogos multijogadores com conexões P2P (*peer-to-peer*) entre os jogadores utilizando multitarefa e baseado no servidor Web da Microsoft IIS .NET. Nesta proposta, uma instância do jogo foi implementada utilizando uma ferramenta baseada na arquitetura que eles propuseram. O sistema foi implementado em C/C++ e utilizaram OpenGL-glut para a renderização dos elementos do jogo. O Microsoft DirectPlay foi utilizado para trabalhar com as conexões P2P entre os jogadores e uma biblioteca integrada ao glut foi utilizada para encapsular a complexidade do DirectPlay.

Bessa et al. [2007] apresentam um motor de jogos 3D multiplataforma que trabalha com o protocolo UDP (*User Datagram Protocol*) para a troca de mensagens, apesar de não oferecer garantias de entrega. Eles implementam uma conexão virtual entre cliente e servidor em que eles mesmos tentam garantir a ordem das mensagens. A questão da utilização do TCP (*Transmission Control Protocol*), que garante a entrega, é que este impõe um custo maior de comunicação e processamento.

Em Bezerra et al. [2009] é apresentado o "FreeMMG 2" como uma arquitetura híbrida do P2P e do cliente-servidor para jogos massivos. Eles tratam o impacto no jogo com a utilização da trapaça (*state cheating*) como um elemento de grande impacto ao se lidar com economias virtuais. A arquitetura da proposta deles tenta evitar a trapaça de forma preventiva, sem a utilização de algoritmos que desfaça as alterações ilegais do jogo, reduzindo bastante o custo de processamento e utilização de banda da rede. Eles apresentam também um outro modelo baseado em um sistema distribuído de nodos servidores voluntários que lida com a escassez e não uniformidade dos recursos do sistema. A ideia desse segundo modelo foi adaptar a frequência de atualização de estado das entidades do jogo de acordo com a sua relevância para o cliente que recebe as atualizações. Ainda, no mesmo artigo, apresentaram um esquema de balanceamento de carga dinâmico para distribuir as cargas proporcionais à capacidade dos servidores.

Kielmann et al. [2009], de forma a acelerar o desenvolvimento de jogos em rede, desenvolveram um *middleware* de comunicação para jogos utilizando o paradigma de P2P, chamado MCommP2P. Neste, o *middleware* realiza tarefas necessárias aos jogos em rede e controla a consistência dos estados e de uma comunicação segura. Este oferece serviços de controle de sessão, criptografia de pacotes e monitoramento de fraudes, podendo eles serem configurados em XML (*eXtensible Marckup Language*). A utilização do paradigma P2P como base se dá de forma que que os envolvidos no jogo pudessem ser mais o independentes possíveis de um controlador central, reduzindo os custos de processamento do mesmo.

Fagá Jr. et al. [2009] apresentam um jogo estilo MMORPG (*Massive Multiplayer Online Role-Playing Game*) desenvolvido em linguagem de programação Python. Neste artigo, eles apresentam a linguagem Python como uma linguagem de sintaxe simplificada e escrita rápida, facilitando a aprendizagem. Além disso, mostram uma biblioteca para Python, chamada Pygame, com módulos para auxiliar na construção de jogos. Para a comunicação do jogo em rede, eles utilizam multitarefa para fazer conexão cliente-servidor com soquetes, para as trocas de mensagens. Não é criado nenhum *middleware* e apenas as funções específicas para conexões soquetes da linguagem Python são utilizadas. A intenção do artigo foi explorar alguns conceitos de desenvolvimento de jogos com a biblioteca Pygame e a plataforma Python.

Kubo [2006], em sua tese, propõe um *framework*, chamado FMMG (*Framework for Mobile Multiplayer Games*), que visa facilitar o desenvolvimento de jogos multijogadores para dispositivos móveis. Isso porque o desenvolvimento para essas plataformas normalmente são dificultadas devido as limitações de memória, recursos de processamento, etc. Ele apresenta o *framework* da forma mais genérica possível, facilitando a portabilidade de programas entre plataformas de desenvolvimento para dispositivos móveis e sob o paradigma de linguagens orientadas a objetos. O *framework* utiliza o padrão MVC (*Model View Controller*) para dividir as funcionalidades envolvidas. Neste *framework*, as principais funcionalidades oferecidas foram a apresentação de imagens do jogo por recursos de visualização, o fornecimento de um

conjunto de classes básicas para facilitar o desenvolvimento, funções para o controle de eventos gerados por diferentes dispositivos de entradas do usuário e uma estrutura de biblioteca para a criação de jogos em rede usando a arquitetura cliente-servidor.

3. *Middeware* Proposto

Apesar de existirem diversas propostas e implementações já prontas de *middlewares* de comunicação para jogos em rede, um dos objetivos do Projeto Subverse vai justamente no sentido de que os estudantes façam suas próprias implementações dos módulos. Contudo, baseando-se em teorias já existentes ou com adaptações, ou até novas ideias, se for o caso. Isso objetiva inserir os alunos ao aprendizado e à experimentação do "aprender fazendo".

Nesse sentido, Silva [2008], ex-aluno das Faculdades Barddal, em seu Trabalho de Conclusão do Curso de Sistemas de Informação, trabalhou em uma proposta de *middleware* para comunicação no Sistema Subverse. O *middleware* proposto foi aprimorado posteriormente e o estado atual do mesmo é mostrado no presente artigo. Contudo, antes de apresentar a proposta do *middleware* de comunicação do Subverse, é importante fazer uma breve apresentação do Sistema do Subverse, propriamente dito.

A estrutura do Subverse (Figura 1) é composta por alguns elementos gerais, organizados conforme o paradigma cliente-servidor. O Servidor Web e o Navegador Web são os elementos necessários para que uma pessoa possa criar um usuário no Subverse, criar seus agentes e gerenciá-los. Tais informações são armazenadas em um banco de dados, da qual o Mundo Virtual faz acesso para executar os elementos no ambiente virtual, de maneira a efetuar a interação entre eles e também para manter as informações de forma persistente. Os agentes, já cadastrados no servidor, são os elementos que o usuário deve desenvolver ou implementar o comportamento, no lado cliente, e que acessa o Mundo Virtual via um representante no lado servidor, o Avatar. Por fim, a Interface Visual é um programa, tal qual um jogo de computador, em que o usuário pode visualizar o que está ocorrendo com seus agentes no Mundo Virtual [Zambiasi et al. 2009].

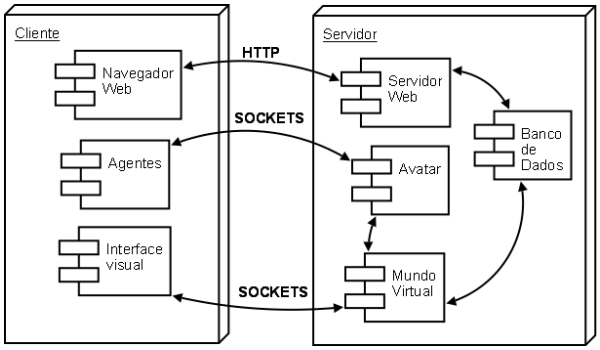

Figura 1: Estrutura do Subverse

De forma a facilitar o desenvolvimento do Subverse na comunicação entre o cliente e o servidor via soquetes, surgiu a necessidade do desenvolvimento de um *middleware* de comunicação. Assim, tendo como inspiração a forma de comunicação do RPC *(Remote Procuedure Call)*, visto em Coulouris et all. [2001], foi feita uma implementação de um *middleware* que fornece ao desenvolvedor um conjunto de métodos que, para ele, é como se estivesse acessando diretamente os métodos no próprio servidor. Esses métodos, são implementados na forma de criar uma mensagem com as informações do tipo do método que está sendo chamado e o conjunto de variáveis que estão sendo enviadas.

Conforme visto na Figura 2, o agente acessa o MAPA (Métodos de Acesso ao *Proxy* do Agente). Por sua vez, o *Proxy* do Agente se utiliza do MAA (Métodos de Acesso ao Agente) para se comunicar com o Agente. Da mesma forma se dá no lado do Visualizador com o MAPV (Métodos de Acesso ao *Proxy* do Visualizador) e o MAV (Métodos de Acesso ao Visualizador).

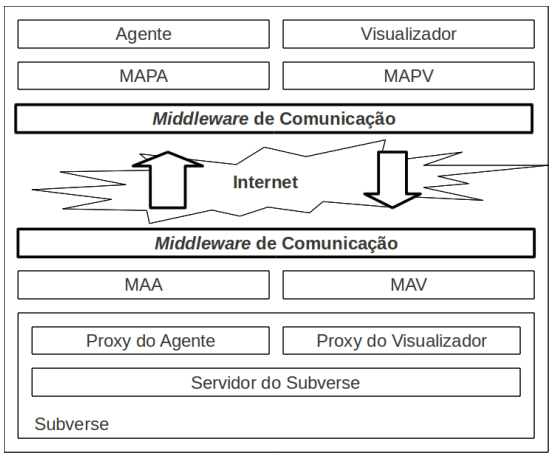

Figura 2: Comunicação do Subverse

O conjunto de métodos MAPA, MAPV, MAA e MAV, utilizam o *middleware* de comunicação do Subverse diretamente, criando mensagens com o nome do método a ser chamado e as variáveis envolvidas na chamada do método, tanto de entrada como de saída (retorno.

As chamadas dos métodos, no Subverse, se dão na forma de comunicação assíncrona, ou seja, um método é chamado para efetuar uma determinada operação e outros métodos são utilizados para saber o *status* da chamada desse método. Por exemplo, se um agente quiser fazer uma movimentação para frente, um método "mover" é chamado com o valor "frente" para a variável "direção". O desenvolvedor deve então utilizar o método "moveu" que retorna *null, false* ou *true* para saber se a operação foi executada com sucesso. Caso o retorno for *null*, então ainda não foi retornada uma resposta, *true* caso conseguiu mover e *false* caso a operação não foi possível.

Por sua vez, o *middleware* de comunicação do Subverse (Figura 3) é composto de alguns módulos principais. Os Métodos de Criação e Extração de Variáveis são responsáveis por criar uma variável com nome, tipo e valor, e métodos correspondentes para recuperar os nomes, tipos e valores das variáveis. Uma mensagem, que é a informação trocada entre o cliente e o servidor, é formada por uma ou mais variáveis, formando uma estrutura de variáveis. Quando o usuário termina a criação de uma estrutura, ele utiliza o método de empacotamento dessa estrutura, criando uma mensagem e enviando-a para a Fila de Mensagens para Envio. O Transmissor fica responsável por enviar essa mensagem ao servidor, quando possível. Cada mensagem possui um código, de forma a poder interpretar de qual requisição um determinado retorno se refere.

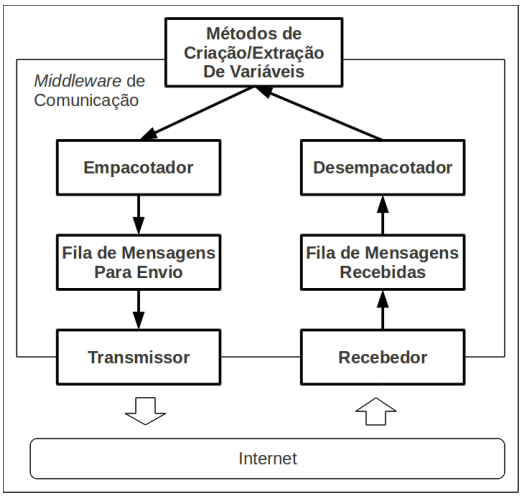

Figura 3: *Middleware* de Comunicação do Subverse

O recebimento de uma mensagem inicia o processo contrário. Quando o recebedor pega uma mensagem, a coloca em uma Fila de Mensagens Recebidas, que pode ser desempacotada e interpretada, para efetuar a chamada da qual a mensagem é responsável para executar.

Todo o processo de troca de mensagens funciona na forma de multitarefa, deixando tanto o cliente quanto o servidor livres para efetuarem as tarefas de processamento das quais são responsáveis, não necessitando o usuário desenvolvedor ficar preocupado com o processo de troca de mensagem. Além disso, o *middleware* do servidor aceita conexões de múltiplos clientes.

4. Considerações Finais

Este artigo apresentou uma proposta e implementação de um *middleware* de comunicação para o Projeto Subverse. Tal recurso objetiva facilitar a implementação do sistema, trabalhando com conexões dos agentes tal como um jogo multijogador online.

O *middleware* implementado foi testado com protótipos de programas que trocam mensagens com um servidor e cumpriu com o proposto. Incluindo as funcionalidades implementadas, estão a criação e extração das estruturas de variáveis, empacotamento e desempacotamento das mensagens, múltiplas conexões e troca de mensagens entre cliente e servidor de forma bastante efetiva.

No estado atual da implementação do Projeto Subverse, o sistema ainda não permite o armazenamento de dados persistentes, nem a conexão de clientes remotos. Dessa forma, os próximos passos do Projeto Subverse são a implementação de uma nova versão do Mundo Virtual com a utilização do *middleware* de comunicação, a modelagem e criação do banco de dados para permitir o armazenamento das informações de forma persistente, e a criação de uma aplicação para Web, em que o usuário pode gerenciar seus agentes via um navegador Web.

Referências

Balan, R., Ebling, M., Castro, P. and Misra, A., 2005. Matrix: adaptative middleware for distributed multiplayer games. *Journal Middleware*. Springer. 390-400.

Benin, M., 2007. Evolução de NPC's e adversários em jogos de computador usando algoritmos genéticos. *Monografia de graduação*. Faculdades Barddal. Florianópolis, SC.

Bessa, A., Sousa, C.T., Bezerra, C.E., Monteiro, I., Bandeira, H. and Souza, R., 2007. O desenvolvimento de um motor multiplataforma para jogos 3D. *SBGames*.

Bezerra, C.E.B., Cecin, F.R. and Geyer, C.F.R., 2009. Modelos de suporte distribuído para jogos online maciçamente multijogador. *Revista Diálogo*, La Salle.

Coulouris, G., Dollimore, J. and Kindberg, T.. 2001. Distributed system: concepts and design. *Addison-Wesley*.

Dickey, C.G, Zappala, D. and Lo, V., 2004. A Distributed Architecture for massively multiplayer online games. *ACM NetGames Workshop*.

Fagá Jr. R., Motti, V.G. and Pimentel, M.G., 2009. IX Escola Regional de Computação Bahia Alagoas Sergipe - ERBASE2009.

Feijo, B. and Kozovitz, L.E., 2003 Arquitetura para jogos massive multi-player. Artigo. *PUC*. Rio de Janeiro.

Kielmann, R., Costa, D.G. and Rocha-Junior, J.B., 2009. MCommP2P: um middleware p2p para jogos em rede. SBGames.

Kubo, M.M., 2006. FMMG: um framework para jogos multiplayer móveis. Tese. Universidade de São Paulo.

Silberschatz, A., Galvin, P., Gagne, G., 2001. Sistemas operacionais: conceitos e aplicações. *Editora Campus*. 6a.ed.

Silva, F.C., 2008. Middelware de comunicação para jogos multiplayer: um estudo de caso no Projeto Subverse. *Monografia de graduação*. Faculdades Barddal. Florianópolis, SC.

Zambiasi, S.P., Benin, M. and Silva, F.C., 2009. Projeto subverse, um mundo virtual baseado em jogos multijogadores como ferramenta de ensino multidisciplinar em cursos de tecnologia da informação. SCGames.