MCommP2P: Um Middleware P2P para Jogos em Rede

Roberio Kielmann Daniel G. Costa João B. Rocha-Junior

Departamento de Tecnologia
Universidade Estadual de Feira de Santana, Bahia, Brasil

Abstract

Nowadays, most of the electronic games communicate using the client-server paradigm, needing specific centralized machines acting as servers. P2P paradigm brings advantages in load distribution, dispensing the use of servers. In such context, a P2P middleware for game developing is created. Such middleware addresses the communications features related with networked games, such as state control and secure data exchange, bringing a new flexible and pluggable solution for the development of games.

Keywords: P2P communication, networked games, middleware.

Resumo

Atualmente, a maior parte dos jogos em rede se comunica utilizando o paradigma cliente-servidor, demandando processamento centralizado em máquinas específicas. O uso do paradigma P2P traz vantagens na distribuição de carga, dispensando o uso de servidores. Nesse contexto, foi desenvolvido um middleware de comunicação para jogos utilizando este paradigma. Esse middleware realiza diversas tarefas necessárias aos jogos em rede, como controle de consistência de estados e comunicação segura, se apresentado como uma solução flexível e fácil de usar no desenvolvimento de jogos.

Palavras-chave: Comunicação P2P, jogos em rede, middleware.

Contato dos autores:
{roberio,daniel,joao}@ecomp.uefs.br

1. Introdução

A indústria nacional de jogos ainda está em um ciclo inicial de amadurecimento, resultando em uma grande distância entre os produtos nacionais e estrangeiros. Isto faz com que motores de jogos (*game engines*), ferramentas RAD (Rapid Application Development) e middlewares especializados tornem-se essenciais para o desenvolvimento de jogos, aumentando seu realismo, jogabilidade e competitividade junto ao mercado [Bittencourt e Osório 2006].

Os jogos eletrônicos em rede destacam-se como uma forte tendência dentro do mercado de entretenimento, apesar de possuírem potencialmente maior custo e complexidade de desenvolvimento. Isto ocorre devido, entre outras coisas, à popularização de tecnologias como a *web*, a difusão do uso de computadores e o barateamento do acesso à Internet.

Atualmente, a maior parte dos jogos que fornecem suporte a comunicação em rede utilizam o paradigma cliente-servidor, devido, entre outros fatores, à menor complexidade e custo de desenvolvimento em relação ao paradigma P2P (peer to peer) [Azambuja 2006]. Contudo, comunicações baseadas no paradigma cliente-servidor tornam necessária a utilização de máquinas específicas e geralmente com maior largura de banda para atuarem como servidor devido à centralização de informações. Isso pode ser tornar um problema para pequenas empresas desenvolvedoras de jogos, devido ao investimento e manutenção desta estrutura, e jogadores, que precisarão definir máquinas para atuarem como servidor [Jábali 2004].

O uso do paradigma P2P no desenvolvimento de jogos *multiplayer*, apesar de ter sua implementação potencialmente mais complexa devido a questões de consistência de estados e monitoramento de fraudes, gera vantagens na distribuição de carga, visto que todas as máquinas dividem as tarefas do servidor. Este paradigma, portanto, dispensa o uso de uma máquina específica centralizando os dados.

Nesse contexto, foi desenvolvido um middleware de comunicação para jogos utilizando este paradigma, o MCommP2P, visando simplificar e reduzir tempo e custos de desenvolvimento de jogos por pequenas empresas e grupos de estudo na área. Esse middleware trata questões específicas dos jogos em rede, como consistência de estados, segurança dos dados que trafegam na rede, busca por sessões de jogos a iniciar, entre outros. A idéia é que esse middleware possa ser utilizado para facilitar o desenvolvimento de jogos em rede, podendo ser utilizado diretamente em jogos desse tipo ou ser incorporado como um módulo em motores de jogos que possuam uma arquitetura que favoreça essa modularização.

Este artigo está organizado da seguinte forma. Na segunda seção é apresentado o desenvolvimento de jogos em rede. Em seguida, os detalhes de especificação e implementação do middleware MCommP2P são apresentados. Além disso, são apresentados nessa seção os testes e resultados do

middleware, mostrando como foi feita sua integração com os protótipos de jogos desenvolvidos, além da discussão de resultados. Por ultimo encontram-se as considerações finais e referências do trabalho.

2. Desenvolvendo jogos em rede

Há algumas décadas, os jogos que possibilitavam a comunicação em rede destacavam-se dos demais, visto que tinham esta opção como um diferencial. Atualmente, oferecer a possibilidade de diversos jogadores participarem de uma mesma partida em um jogo sem que estejam reunidos presencialmente ou interconectados no mesmo dispositivo é um requisito comum [Bittencourt e Osório 2006]. Dentro da grande quantidade de jogos em rede disponíveis no mercado, destacam-se quatro grandes grupos: First Person Shooters (FPS), Real Time Strategy (RTS), Massively Multiplayer Online Game (MMOG) e Non Real Time (NRT) [Anibolete 2006].

Alguns desafios são comuns a todos os jogos em rede, devendo haver um maior esforço para tentar resolve-los ou minimizá-los durante o desenvolvimento de jogos deste tipo. Entre os desafios mais comuns nessa área estão a consistência de estados, o controle de fraudes e a segurança na comunicação.

Manter a consistência de estados significa garantir que todos os jogadores possuem o mesmo estado do jogo e que concordam com cada acontecimento desse jogo. Esta funcionalidade é considerada a mais complexa, visto que se torna difícil garantir a consistência de estado sem causar perdas de desempenho da aplicação. Para a sincronização do estado entre todos os computadores envolvidos na partida, diversos algoritmos distribuídos otimistas e pessimistas são propostos. Os algoritmos distribuídos pessimistas são projetados para evitar a ocorrência de inconsistências, não estando preparados para corrigi-las, enquanto que os otimistas permitem a ocorrência de inconsistências, realizando a correção assim que identificadas [Anibolete 2006]. Alguns dos principais algoritmos distribuídos são o stop-and-wait, o bucket synchronization e o Trailing State Synchronization, sendo o primeiro pessimista e os demais otimistas.

Para dificultar a ação de jogadores mal intencionados é necessária a utilização de mecanismos de criptografia dos pacotes transmitidos na rede, já que capturar pacotes para análise e geração de outros, que possam dar vantagens ao jogador, é considerada uma prática comum [Kozovits 2003]. Os algoritmos simétricos, que utilizam a mesma chave para criptografar e decriptografar, são considerados a melhor opção de criptografia de pacotes, pois possuem uma menor complexidade temporal quando comparados aos assimétricos. Algoritmos assimétricos têm sua complexidade aumentada por utilizarem um par de chaves, uma pública e outra privada, usadas na criptografia e decriptografia, respectivamente [Hinz 2000]. Neste contexto, os algoritmos assimétricos podem ser usados para garantir o compartilhamento da chave simétrica de forma segura.

Em jogos com número elevado de jogadores, não se pode assumir que todos estejam agindo corretamente, visto que é comum existirem jogadores que tentam obter vantagens irregulares durante o jogo. Sendo assim, torna-se necessário o monitoramento e combate às fraudes que ocorrem durante o jogo, visando minimizar o número de vitórias fraudulentas e restringir ou desestimular a quantidade de tentativas de violação [Baughman e Levine 2007]. Uma das alternativas nesse sentido é guardar um arquivo de *log* contendo a data e a hora de inicio e fim da seção, bem como ocorrências suspeitas. A partir da análise estatística destes dados é possível identificar o nível de confiabilidade de algum jogador, que caso seja baixo pode ter suas atividades restringidas ou bloqueadas. Em casos de ocorrências de atividades suspeitas, podem ser enviadas mensagens de notificação ao jogador de forma não determinística para que seja despertada a dúvida se está ou não sendo monitorado [Baughman e Levine 2007]. Todas essas funcionalidades, que devem ser levadas em consideração no desenvolvimento de jogos em rede, estão presentes no middleware MCommP2P.

3. O Middleware MCommP2P

Visando facilitar o desenvolvimento de jogos em rede, foi criado um middleware de comunicação que já implementa diversos serviços relacionados aos jogos desse tipo. Esses serviços são disponibilizados seguindo o paradigma P2P, não sendo obrigatório, nesse caso, o uso de servidores centralizando o controle das informações. A idéia foi criar um recurso que facilite e acelere o desenvolvimento de jogos, apoiando diretamente pequenos desenvolvedores.

O middleware desenvolvido oferece serviços relacionados à comunicação de jogos em rede, endereçando problemas como consistência de estados, controle de sessão, criptografia de pacotes e monitoramento de fraudes. Desta forma, estes serviços podem ser abstraídos pelo usuário, que poderá escolher quais funcionalidades oferecidas serão utilizadas, tornando o desenvolvimento de jogos em rede mais simples. Todos esses serviços podem ser configurados através de um arquivo de definições em XML.

Outra característica do software desenvolvido está relacionada ao paradigma de comunicação P2P, utilizado durante a dinâmica do jogo. Este paradigma foi escolhido para que os jogos desenvolvidos pudessem ser os mais independentes possíveis de um controlador central, reduzindo assim os custos. Algumas das funcionalidades oferecidas ainda possuem dependência com um servidor, como a autenticação de usuários, devido à dificuldade de persistência de dados em redes P2P. Contudo,

características como estas podem ser desabilitadas pelo desenvolvedor, caso não sejam necessárias.

A consistência de estados foi implementada no middleware seguindo o algoritmo stop-and-wait. Este algoritmo limita-se a bloquear o envio de mensagens até que todos tenham enviado no turno anterior, funcionando como uma barreira cíclica. A quantidade de mensagens que deve ser recebida para que o envio esteja liberado é definido após o início da sessão.

Para evitar que esta forma de sincronização possa atrapalhar a integração com algum jogo, e permitir aos desenvolvedores a opção de realizar sua própria sincronização através de outros algoritmos, é oferecida a opção de desativação da barreira no arquivo de configuração do middleware. Com a barreira desativada, as mensagens são enviadas imediatamente, devendo a consistência de estado ser garantida, nesse caso, pelo desenvolvedor do jogo. Após passar pela barreira, a mensagem é enviada a cada um dos *hosts* de uma sessão, sendo para isto serializada, criptografada, quando a criptografia está ativa, e enviada através do canal de comunicação entre os *hosts*. Os algoritmos de criptografia utilizados na implementação inicial do middleware foram o RSA e o AES.

Para monitorar fraudes, o middleware MCommP2P valida os pacotes recebidos através da captura de erros durante a deserialização dos objetos transmitidos. A ocorrência de erros neste processo indica a alteração de conteúdo do pacote, sendo a suspeita encaminhada ao módulo Monitor (pertencente ao middleware). Ao receber uma notificação, o módulo Monitor pode realizar dois procedimentos: notificar o jogador suspeito, sendo isto feito de forma não determinística para despertar a dúvida sobre seu monitoramento, ou notificar um servidor informado no arquivo de configuração, para que este possa analisar estatisticamente as ocorrências. A figura 1 apresenta o digrama de classes do middleware.

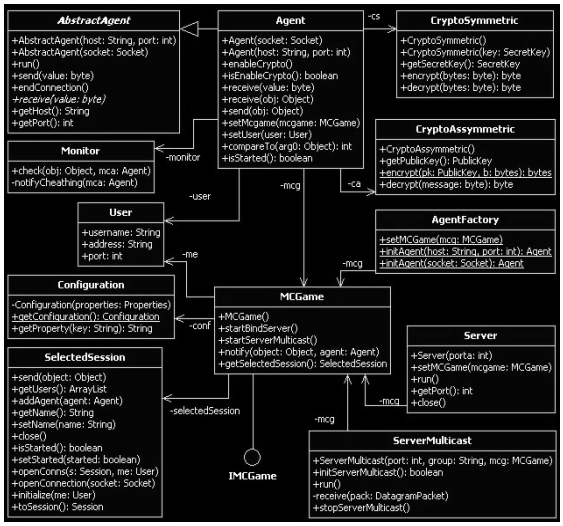

Figura 1: Diagrama de classes do middleware.

3.1 Interfaces de integração

O middleware MCommP2P foi idealizado para ser integrado facilmente a qualquer jogo em rede que esteja preparado para isso. Essa integração pode ser feita diretamente, com chamadas explícitas vindas do código funcional do jogo. Para isto, duas interfaces foram especificadas no middleware: Observer, que deve ser implementada pelo jogo, e IMCGame, que corresponde aos serviços de comunicação do MCommP2P. A Figura 2 apresenta um digrama de componentes dessa integração.

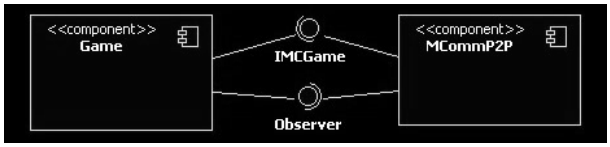

Figura 2: Modelo de integração do middleware.

Outra opção de integração é através de uma incorporação a um dos motores de jogos disponíveis. De fato, o middleware foi idealizado para utilização em jogos com o padrão arquitetural semelhante ao do motor de jogos JFROG [Bittencout et al. 2003], cujo modelo dá ênfase a modularização do jogo. Nestes tipos de jogos a parte de comunicação é separada do restante do jogo, fazendo com que o middleware possa ser integrado com este através dos serviços disponibilizados e de uma interface para notificação de eventos ao módulo de controle do jogo. Apesar disso, o MCommP2P pode ser adicionado a qualquer arquitetura de jogo, desde que o programador trate eventuais necessidades de compatibilização. A Figura 3 apresenta uma idéia de como o middleware MCommP2P poderia ser integrado ao modelo arquitetural do motor de jogos JFROG.

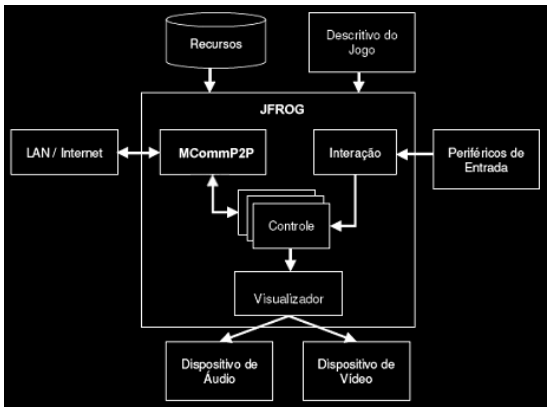

Figura 3: JFROG com o middleware MCommP2P.

3.2 Casos de testes

Para verificar a usabilidade do middleware no desenvolvimento de jogos em rede, foram desenvolvidos dois protótipos de jogos que ilustram as possibilidades da aplicação: Jogo da Velha e Space Invaders (jogo clássico na vertical de naves espaciais, onde o objetivo do jogo em rede é uma nave acertar a

outra). Para o desenvolvimento do Jogo da Velha, foi criada uma única classe contendo tanto a lógica do jogo quanto sua interface. Este jogo teve como objetivo principal realizar os primeiros testes e ajustes do middleware, não sendo analisadas características relacionadas à performance. Ao final dos ajustes, diversas funcionalidades já estavam validadas, como a inicialização do jogo (gerência de usuários e sessões) e a troca de mensagens.

O desenvolvimento do jogo Space Invaders necessitou de mais planejamento, visto que sua finalidade foi testar não somente o funcionamento, mas também o desempenho do middleware desenvolvido, além de requisitos mais complexos como a sincronização de estados do jogo. O modelo estrutural do motor JFROG foi utilizado como base para a definição arquitetural do jogo, como apresenta a Figura 4. Neste jogo foram definidas três classes principais, sendo elas: CaptureKeyboard, responsável pela captura de entradas do teclado, Canvas, responsável pela renderização gráfica do jogo, e Game, que realiza a consistência e dinâmica de estados do jogo.

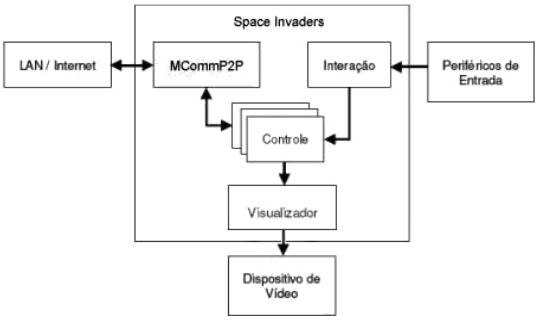

Figura 4: Modelo conceitual do jogo, com o MCommP2P.

Para realização dos testes iniciais de desempenho, o middleware foi configurado com as opções de sincronização de estados e criptografia ativas, e o jogo teve sua taxa de atualização fixada em 30 vezes por segundo. Esta taxa indica que o estado do jogo é atualizado 30 vezes a cada segundo, implicando na sua alteração a partir da captura de eventos do teclado e mensagens da rede, e renderização para apresentação ao jogador. Com essa configuração apresentada o jogo manteve-se estável durante os testes, sem ocorrência de erros, a uma taxa de atualização de quadros em 30 frames por segundo (fps) e um consumo de banda médio de 24 Kbps.

Para os jogos utilizados nos testes, a união das técnicas de sincronização, serialização e criptografia não geraram atrasos visivelmente perceptíveis em ambos os casos. Em relação ao tamanho dos pacotes, a técnica de criptografia gerou um aumento médio de 10%. Já na serialização, o aumento médio foi de 15% mais 40 bytes, referente ao cabeçalho de identificação da classe. Para pacotes muito pequenos é aconselhado o uso de arrays de bytes, pois neste caso a serialização gera somente um aumento de 25 bytes referente ao cabeçalho de identificação.

4. Considerações finais

O trabalho desenvolvido apresentou um middleware P2P para suporte ao desenvolvimento de jogos em rede. Esse middleware possui grande potencial para suporte ao desenvolvimento de jogos em rede, sobretudo quando consideramos pequenos desenvolvedores. Os testes iniciais mostraram que o MCommP2P funciona como esperado, podendo ser utilizado em projetos pilotos.

Como trabalhos futuros, pretende-se implementar o middleware sobre plataformas avançadas, como o XPastry e JXTA (MCommP2P foi montado sobre sockets TCP). Além disso, é esperada a integração do middleware com jogos mais complexos, visando a realização de testes mais conclusivos sobre a usabilidade comercial da solução. Por fim, testes formais de desempenho serão realizados, buscando a comparação do MCommP2P com outras soluções.

Referências

Bittencourt, J. R. e Osório, F. S. 2006. “Motores para Criação de Jogos Digitais: Gráficos, Áudio, Interação, Rede Inteligência Artificial e Física”. Disponível em: http://osorio.wait4.org/oldsite/iajogos/artigos/bittencourt-osorio-eri-mg2006.pdf. Porto Alegre.

Azambuja, J. 2006. “Peer to Peer: Modelos, Middlewares e Aplicações”. Disponível em: http://www.inf.ufrgs.br/~asc/sodr/pdf2006-02/SODRJoseAzambuja.pdf. Porto Alegre.

Jábali, S. 2004. “Introdução ao Projeto Peers”. Disponível em: http://www.lsd.ic.unicamp.br/projetos/peers/arquivos/PEERS_Intro.pdf. Laboratório de Sistemas Distribuídos, Unicamp. Campinas.

Anibolete, T. J. 2006. “Algoritmos distribuídos em Jogos Multi-usuário”. Disponível em: http://www-di.inf.puc-rio.br/~endler/courses/DA/Monografias/06/MultiUserGames-Tulio-Mono.pdf. PUC-RIO. Rio de Janeiro.

Kozovits, L. E. 2003. “Arquiteturas para Jogos Massive Multiplayer”. Disponível em: ftp://ftp.inf.puc-rio.br/pub/docs/techreports/03_36_kozovits.pdf. Rio de Janeiro.

Hinz, M. A. M. 2000. “Um estudo descritivo de novos algoritmos de criptografia”. Disponível em: http://www.ufpel.tche.br/prg/sisbi/bibct/acervo/info/2000/Mono-MarcoAntonio.pdf. Pelotas.

Baughman, N. E, e Levine, B. N. 2001. *Cheat-proof playout for centralized and distributed online game*. In: Procedures of IEEE Infocom, San Francisco.

Bittencourt, J. R., Giraffa, L. M. M. e Santos, R. C. 2003. *Criando Jogos Computadorizados Multiplataforma com Amphibian*. In: II Congresso Internacional de Tecnologia e Inovação em Jogos Computadorizados. Curitiba.