Suporte distribuído a jogos maciçamente multijogador em cenários com recursos limitados

Carlos Eduardo Benevides Bezerra[1], **Cláudio Fernando Resin Geyer**[1]

[1]Instituto de Informática – Universidade Federal do Rio Grande do Sul
Caixa Postal 15.064 – 91.501-970 – Porto Alegre – RS – Brasil

Abstract. *Traditionally, a central server is utilized to provide support to massively multiplayer games, where the number of participants is of the order of tens of thousands. In this work, it is proposed the utilization of a server system composed of geographically distributed lower-cost nodes, employing techniques to decrease the necessary bandwidth to these nodes. One of these techniques is a refinement of interest management algorithm, which obtained significant results in simulations. Other techniques still in preliminary phase of specification are: server nodes overlay network topology construction using knowledge of the real network topology, load balancing and hotspots dectection.*

Resumo. *Tradicionalmente, utiliza-se um servidor central para prover suporte a jogos maciçamente multijogador, onde o número de participantes é da ordem de dezenas de milhares. Neste trabalho, propõe-se utilizar um sistema servidor composto de nodos geograficamente distribuídos de menor custo, empregando-se técnicas para reduzir a largura de banda necessária para estes nodos. Uma das técnicas é um refinamento do gerenciamento de interesse, que obteve resultados significativos nas simulações realizadas. Outras técnicas ainda em fase preliminar de especificação são: construção da topologia de rede overlay de nodos servidores utilizando conhecimento da topologia real da rede, balanceamento de carga e detecção de aglomerados de jogadores (hotspots).*

1. Introdução

Atualmente, jogos eletrônicos têm se tornado bastante populares, especialmente os jogos maciçamente multijogador, onde há um número de participantes simultâneos da ordem de dezenas de milhares [Cecin et al. 2004]. Como exemplos, podemos citar World of Warcraft [Blizzard 2004], Lineage II [NCsoft 2003] e Guild Wars [ArenaNet 2005].

Usualmente, o suporte de rede para este tipo de aplicação consiste em um servidor central com recursos - capacidade de processamento e largura de banda para comunicação com os jogadores - super-dimensionados, ao qual se conectam as máquinas clientes. Cada jogador interage através de um destes clientes, que envia suas ações para o servidor, que as processa, verificando que alterações no jogo elas causam, e difunde o resultado para todos os clientes envolvidos. Em virtude do número de participantes simultâneos que este tipo de jogo costuma ter, percebe-se que tais tarefas demandam por uma quantidade de recursos significativa, no que tange a poder de processamento e, principalmente, largura de banda disponível para que sejam recebidas as ações dos jogadores e enviadas as atualizações de estado.

Nos últimos anos, têm-se pesquisado alternativas à abordagem com servidor centralizado. Uma delas é a distribuição, entre os próprios participantes, tanto da

simulação do jogo quanto da responsabilidade de atualizarem-se entre si quando realizam ações. A comunicação entre eles ocorre par-a-par, formando uma rede descentralizada [Schiele et al. 2007]. Esta abordagem seria o ideal, não fossem alguns problemas que lhe são inerentes. Por exemplo, como os jogadores participam do processamento da simulação, é necessário que eles entrem em acordo no que diz respeito ao estado da partida, sob pena de haver inconsistências caso isto não seja feito.

Outra questão se refere ao número de envios que cada participante tem que executar. No modelo cliente-servidor, basta que cada um envie suas ações para o servidor, que se encarrega de simular e difundir o novo estado para os outros jogadores. No caso do modelo par-a-par, cada par envolvido torna-se responsável por processar suas ações e enviar as atualizações de estado para os outros participantes. O problema disto reside no fato de que não se pode garantir que todos os jogadores possuam conexões de rede com largura de banda suficiente. Por fim, sem um servidor central, que poderia atuar como árbitro, o jogo torna-se dependente da simulação que os próprios jogadores executam, que pode ser desvirtuada de forma a chegar a um resultado inválido, que beneficie indevidamente determinado jogador ou mesmo que invalide a sessão de jogo.

Além do modelo par-a-par, existe também a alternativa de utilizar um servidor distribuído, em que diversos nodos conectados entre si dividem a tarefa de simular o jogo, como também de enviar as atualizações de estado aos jogadores [Assiotis and Tzanov 2006]. Tal abordagem possibilita o uso de computadores de menor custo para comporem o sistema distribuído servidor, barateando a infra-estrutura de suporte. Questões como consistência e vulnerabilidade a trapaça podem ser abstraídas, restringindo o conjunto de nodos servidores a computadores comprovadamente confiáveis, o que é plausível, levando em conta que o número de nodos servidores deverá ser algumas ordens de grandeza menor do que o número de jogadores. Além disso, não é necessário exigir que cada jogador envie atualizações de estado para todos os outros jogadores. Com menores exigências de largura de banda e processamento das máquinas clientes, o jogo torna-se acessível para um maior público.

No entanto, para evitar que o custo de manutenção do sistema distribuído servidor como um todo não se aproxime do custo de manutenção de um servidor central, é necessário realizar algumas otimizações com o intuito de reduzir a largura de banda necessária para cada um dos nodos. O presente trabalho, em desenvolvimento, propõe o uso e/ou investigação de algumas técnicas para reduzir o uso de largura de banda causado pelo tráfego do jogo entre os servidores e os clientes, diminuindo a quantidade de recursos necessários. Uma delas é um refinamento da técnica de gerenciamento de interesse [Boulanger et al. 2006] dos jogadores. O princípio básico desta técnica é que cada participante do jogo receba apenas atualizações de jogadores cujo estado lhes seja relevante. Foram realizadas simulações comparando a proposta deste trabalho com técnicas convencionais, obtendo resultados significativos. Além disso, propõe-se uma técnica que visa prover qualidade de serviço, adaptando a freqüência de atualizações enviadas pelo servidor à disponibilidade de recursos. Outra consiste em um heurística para detecção de aglomerados de jogadores - ou hotspots - de forma a otimizar o balanceamento de carga entre os servidores. Por fim, pode ser empregado conhecimento a respeito da topologia real da rede na criação da topologia da rede overlay do sistema servidor.

O artigo está dividido da seguinte maneira: na seção 2 são citados alguns trabalhos

relacionados; na seção 3, são apresentadas as definições de alguns conceitos utilizados ao longo do texto; na seção 4, é descrito o modelo de distribuição proposto, assim como algumas técnicas que se pretende implementar; na seção 5 é apresentada uma otimização proposta para reduzir o tráfego sem comprometer a qualidade do jogo; nas seções 6 e 7 é descrita a simulação realizada para validar a técnica proposta e os resultados obtidos, respectivamente e, na seção 8, são apresentadas as conclusões a que se chegou até então neste trabalho. O andamento desta pesquisa é descrito na seção 9.

2. Trabalhos relacionados

Como já foi dito, alguns trabalhos já foram feitos nos últimos anos visando distribuir o suporte a jogos maciçamente multijogador. Uma das abordagens é o modelo par-a-par [Crippa et al. , Hampel et al. 2006, Knutsson et al. 2004], que tem algumas dificuldades, no que se refere a consistência do estado do jogo nos diferentes pares participantes, vulnerabilidade a trapaça e uso eficiente de largura de banda. Alguns autores propõem abordagens cujo objetivo é minimizar estes problemas. Um destes trabalhos [Schiele et al. 2007] propõe a divisão do ambiente virtual simulado no jogo em regiões, e dentro de cada região é escolhido um par que será eleito coordenador daquela região. Sua função será a de gerenciar o interesse dos jogadores, verificando para quais pares cada atualização realmente precisa ser enviada. Dessa forma, reduz-se o uso de largura de banda de envio dos pares. No entanto, o uso de largura de banda de envio de cada participante ainda tende a ser significativamente superior àquele necessário quando utilizado o modelo cliente-servidor, pois neste é necessário apenas que cada jogador envie suas ações para um único destino. No modelo par-a-par, cada jogador deve atualizar, normalmente, mais de um outro jogador. Além disso, é necessário que o par escolhido para gerenciar o interesse naquela região seja confiável.

Outro trabalho voltado para o modelo par-a-par [Iimura et al. 2004] tem uma abordagem semelhante à de [Schiele et al. 2007], mas sugere que, para cada região do ambiente virtual, seja criada uma "federação de servidores", formada por pares escolhidos entre os participantes. A simulação torna-se mais confiável, já que diferentes nodos irão gerenciar aquele lugar no mundo do jogo e precisarão estar em acordo para que a simulação prossiga. Porém, o risco dos nodos escolhidos para gerenciarem aquela região cometerem trapaça de conluio [Yan and Randell 2005] não é eliminado. Além disso, o próprio acordo entre os nodos servidores, que provê maior confiabilidade na simulação, implica em grande quantidade de tráfego entre os nodos participantes, além de potencialmente atrasar cada passo da simulação.

Um grande problema das arquiteturas par-a-par, no que diz respeito à utilização de gerenciamento de interesse é que cada par é responsável por parte da simulação e por decidir para quem sua atualização de estado interessaria. Assumindo que haja apenas jogadores confiáveis, a técnica de gerenciamento de interesse pode ser útil. No entanto, supondo que determinado jogador seja malicioso, ele pode não enviar atualizações de seu próprio estado para algum outro jogador, que ficaria prejudicado no jogo. Sendo assim, o modelo de servidor distribuído [Assiotis and Tzanov 2006, Ng et al. 2002, Fiedler et al. 2002, Rieche et al. 2007] é considerado mais adequado para utilização de técnicas de gerenciamento de interesse dos jogadores.

Um exemplo deste tipo de modelo é descrito em [Assiotis and Tzanov 2006], onde

é proposta uma arquitetura distribuída para jogos maciçamente multijogador. Também é baseada na divisão do ambiente virtual do jogo em regiões, porém a cada uma destas estaria associado um nodo servidor. O jogador que estivesse situado em determinado lugar no mundo virtual deveria conectar-se ao servidor responsável por aquela região. Desta forma, cada servidor agruparia diferentes jogadores, baseado em sua localidade no ambiente do jogo. Para alcançar consistência entre os diferentes nodos servidores efetuando a simulação, é utilizado o conceito de travas. Quando um determinado nodo servidor precisa alterar o estado de uma entidade qualquer da partida, primeiro precisa obter acesso exclusivo àquela entidade. Para isso, ele negocia com os outros nodos servidores que possam também querer fazer alguma alteração, para somente então efetuar a mudança. Quando termina, o acesso é liberado, e os outros servidores são avisados através de mensagens.

A primeira grande restrição no trabalho de [Assiotis and Tzanov 2006], no entanto, é a premissa de que os nodos servidores estão conectados através de uma rede de alta velocidade e baixa latência, o que não pode ser assumido quando se trata de nodos de mais baixo custo geograficamente distribuídos. Outro problema é que a questão da escalabilidade é tratada através da pura e simples expansão do ambiente virtual, supondo que os jogadores se espalharão por ele. Por último, sugere-se resolver o problema de haver um grande número de jogadores no mesmo lugar através de sucessivos reparticionamentos recursivos das regiões, de forma a dividir os jogadores entre diferentes servidores. No entanto, existe um limite para o reparticionamento do ambiente virtual, e é deixado de lado o que fazer quando é atingido este limite.

No que se refere a gerenciamento de interesse, trabalhos como [Morse et al. 1996, Rak and Van Hook 1996, Zou et al. 2001, Morgan et al. 2005] podem ser citados. Em [Rak and Van Hook 1996], é proposto um esquema de gerenciamento de interesse baseado em um ambiente virtual divido em células de uma grade. A cada célula está associado um grupo de multicast. Cada participante da simulação se subscreve então no grupo de multicast da célula onde ele se encontra, assim como de células vizinhas se estiverem ao alcance de sua visão. Cada participante envia então suas atualizações de estado ao grupo de multicast da região onde se encontra.

Em [Zou et al. 2001], também são considerados esquemas de gerenciamento de interesse baseados em grupos de multicast. É feita uma comparação entre a formação de grupos baseado em célula, onde cada um está associado a uma célula de uma grade que compõe o ambiente virtual, e agrupamento baseado em objetos, onde para cada objeto existe um grupo multicast associado. Verificou-se que há uma compensação (tradeoff) entre o custo das mensagens de controle dos grupos de multicast e o custo das mensagens de atualização de estado propriamente ditas.

Um esquema de gerenciamento de interesse que utiliza um middleware orientado a mensagens é apresentado em [Morgan et al. 2005]. Dentre outros aspectos, este esquema faz uma predição do que será o interesse de determinado participante no futuro, baseado na posição e vetor velocidade do mesmo no ambiente virtual. Dessa forma, cada um começa a receber atualizações de estado de entidades que não estão ainda ao alcance de sua visão, mas que provavelmente estarão em um futuro próximo, tornando seus estados disponíveis assim que elas estiverem dentro do campo de visão.

Em [Morse et al. 1996], é feito um apanhado de sistemas que utilizam a técnica de gerenciamento de interesse, salientando quais critérios são utilizados por cada um. É apresentada então uma taxonomia de tais sistemas, classificando-os de acordo com: modelo de comunicação, foco da filtragem e domínios de responsabilidade. O modelo pode ser *unicast*, *multicast* ou *broadcast*. O foco da filtragem refere-se a que tipo de características são observadas de cada objeto para realizar esta filtragem: podem ser *intrínsecas*, como o valor de atributos do objeto (e.g. coordenadas exatas de sua localização), ou *intrínsecas*, como a qual grupo multicast ele está associado. Por fim, o domínio de responsabilidade atribuída a um gerenciador de interesse, que verifica para quem cada estado é relevante, pode ser *dinâmico* ou *estático*. Por exemplo, se cada gerenciador é designado para controlar uma área fixa do ambiente virtual, seu domínio de responsabilidade é estático, mas se ele controla uma área que possa aumentar ou diminuir de tamanho, seu domínio de responsabilidade é dinâmico.

Levando em consideração que o modelo de comunicação multicast não é amplamente suportado na Internet [El-Sayed 2004], tanto por razões técnicas quanto comerciais, neste trabalho optou-se por seguir o modelo unicast, considerando que cada broadcast consiste na verdadem em um conjunto de sucessivas transmissões unicast, uma para cada destino. Além disso, utiliza-se filtragem intrínseca e domínios de responsabilidade dinâmicos, para que haja maior precisão e, consequentemente, uma maior redução no tráfego de atualizações de estado.

3. Definições

Será utilizado o termo cliente para referir-se ao computador utilizado por cada jogador para conectar-se a um dos servidores do jogo, assim como o termo servidor fará referência a cada nodo integrante do sistema distribuído que estará servindo o jogo. É necessário descrever o modelo de suporte a jogos maciçamente jogador sobre o qual pretende-se utilizar o algoritmo de gerenciamente de interesse proposto. Ao longo do texto, serão utilizados alguns termos que precisam antes ser definidos:

Avatar é a representação do jogador no ambiente virtual. É através dele que o jogador interage com o mundo do jogo e com outros jogadores. Exemplos de avatar são os personagens controlados pelo jogador em jogos MMORPG, como World of Warcraft.

Entidades são as peças constituintes do mundo virtual. Exemplos de entidades são os próprios avatares dos jogadores, assim como avatares controlados por inteligência artificial do servidor - monstros dos MMORPGs, por exemplo - e dos objetos inanimados presentes no ambiente, tais como portas, armas e itens em geral com que os avatares possam interagir.

Estado é o conjunto de propriedades que podem ser observadas nas diferentes entidades do jogo. O estado global do mundo simulado é constituído dos estados individuais das diferentes entidades nele presentes.

Os jogadores interagem com o mundo do jogo através de **ações**. Uma ação é um comando do jogador como, por exemplo, mover seu avatar para determinada localização no mundo virtual, atacar outro jogador, tomar para si algum objeto disponível no ambiente e assim por diante. Em geral, ações modificam o estado de uma ou mais entidades presentes no jogo.

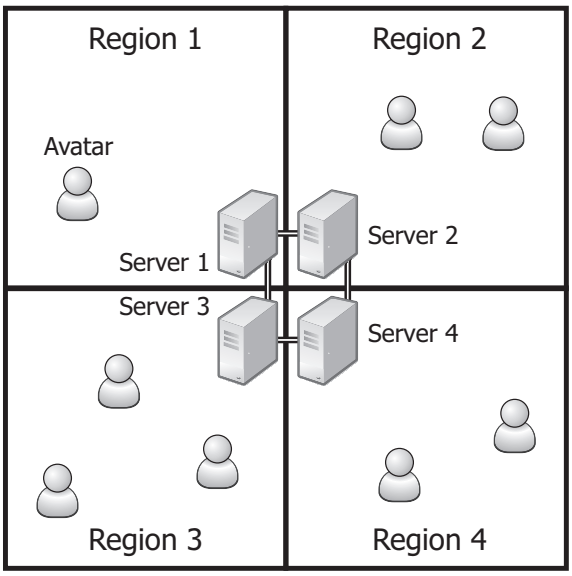

Figura 1. Modelo de distribuição

Região é um partição do ambiente virtual, sob responsabilidade de um único servidor. Dessa forma, jogadores cujos avatares estejam localizados na mesma região terão sua interação beneficiada, pois suas máquinas estarão conectadas ao mesmo servidor.

A **fronteira** entre duas regiões é a divisa entre as áreas que essas regiões ocupam. Quando um avatar está localizado próximo a uma fronteira, o servidor responsável pela região além desta fronteria é avisado a respeito da presença daquele avatar pelo servidor onde ele se encontra.

4. Modelo de distribuição

Este trabalho propõe um ambiente virtual particionado em regiões, cada uma gerenciada por um servidor. As regiões são contíguas, explorando a localidade dos avatares dos jogadores. Dessa forma, avatares próximos no jogo provavelmente estarão localizados na mesma região e, por conseguinte, os clientes dos jogadores a eles associados tenderão a estar conectados ao mesmo servidor, fazendo com que sua interação seja mais rápida (Figura 1). Em situações em que jogadores interagindo entre si estivessem conectados a diferentes servidores implicaria em maior tráfego, pois seria necessário algum tipo de negociação entre os servidores aos quais os diferentes jogadores estão conectados, para que os estados da simulação em ambos fossem idênticos. Além disso seria necessário que cada mensagem entre estes clientes desse mais saltos, passando por mais de um intermediário.

Uma questão que diz respeito a esse modelo de particionamento do ambiente virtual está relacionada às fronteiras entre as regiões. Se um avatar de um cliente conectado a um servidor está próximo à fronteira de uma região com outra, que está associada a um outro servidor, será necessário haver troca de informações entre os servidores. Essas informações consistirão em atualizações dos estados das entidades que estão interagindo entre si apesar de estarem situadas em regiões diferentes. Por exemplo, seja S_A o servidor responsável pela região R_A onde está situado o avatar do cliente C_A e S_B o servidor responsável por outra região, R_B, onde está situado o avatar do cliente C_B. Quando o avatar

de C_A aproxima-se da fronteira com R_B, S_A envia para S_B uma mensagem alertando a respeito da presença daquele avatar próximo da fronteira. Se o avatar de C_B aproximar-se da fronteira com R_A, S_B também avisa S_A a respeito, e começa a enviar atualizações de estado de C_B para S_A, que então encaminha para C_A, e vice-versa.

No que diz respeito à simulação das ações executadas por jogadores cujos avatares estão situados em regiões diferentes, deve-se decidir como será feita a simulação. Como o foco deste trabalho não é a simulação em si, mas sim a otimização do uso de largura de banda através de um novo algoritmo de gerenciamento de interesse, decidiu-se que a simulação será realizada pelo servidor ao qual o cliente daquele jogador está conectado. Dessa forma, se o jogador cujo avatar está em R_i executar uma ação próximo à fronteira, envolvendo entidades em R_j, será o servidor S_i quem decidirá o resultado destas ações, repassando a S_j apenas o novo estado já calculado. Os detalhes deste mecanismo não irão interferir no gerenciamento de interesse. Quando um jogador J com o avatar próximo à fronteira de determinada região executa ações cujo resultado precisa ser difundido para jogadores com os avatares em outras regiões, o servidor S responsável por J simula suas ações, calcula o estado resultante e simplesmente o envia para o servidor vizinho, como se estivesse enviando para seus próprios clientes. Isso acontece da mesma forma que aconteceria se os outros jogadores também estivessem conectados a S. Analogamente, quando S receber o estado resultante de uma ação de um jogador que está na região vizinha à sua, difunde-o para os jogadores a ele conectados como se um jogador dentro de sua própria região tivesse executado a ação.

Nas próximas seções serão descritos alguns mecanismos básicos do modelo de distribuição proposto, porém que estão ainda em desenvolvimento. Na seção 5, é apresentada uma parte já concluída deste trabalho, cujos resultados foram obtidos através de simulações.

4.1. Entrada e saída de nodos servidores

Pretende-se ter obter um sistema servidor dinâmico, no sentido em que poderão entrar e sair nodos servidores. Como idéia preliminar para bootstrap, poder-se-ia começar o sistema servidor com um nodo apenas, que convidaria outros nodos a unirem-se a ele. Estes nodos poderiam eles também convidar outros nodos e assim por diante. Quando um novo nodo passasse a integrar o sistema, parte da região de outro nodo que já estava presente é delegada a ele. Quando um nodo servidor saísse do sistema, a região pela qual ele estava responsável é entregue a outro nodo do sistema. O funcionamento exato dessa parte do sistema ainda não foi totalmente definido, sendo um trabalho em andamento.

4.2. Disseminação de informações

O objetivo deste modelo é ser o mais escalável possível, no que diz respeito à extensão do ambiente virtual e do número de nodos servidores. Sendo assim, deseja-se que um número maior de servidores não implique em um sobrecusto crescente, a ponto de inviabilizar a utilização deste modelo de suporte. Contudo, permanece a necessidade de disseminar tanto a lista de nodos servidores, quanto os estados dos diferentes avatares entre através do sistema. Caso houvesse obrigação de garantia de entrega de cada uma dessas informações a todos os nodos envolvidos, ter-se-ia uma complexidade alta, pois cada um dos n nodos servidores estaria enviando continuamente informações para $n-1$ outros servidores. Assim, o número de mensagens trocadas teria uma complexidade $O(n^2)$, levando

a uma saturação da banda de comunicação dos nodos à medida em que fosse aumentada a escala, tanto em número de nodos servidores, quanto de jogadores.

Para evitar isso, propõe-se a utilização de difusão probabilística destas informações, de maneira que não é necessário garantir a entrega de todas elas. Segundo [Kermarrec et al. 2003], é possível tornar esse tipo de difusão quase tão confiável quanto a difusão determinística, porém reduzindo sensivelmente o tráfego gerado pelas mensagens e confirmações.

A difusão do estado dos avatares ocorre da seguinte forma: como cada cliente está conectado a um servidor, este servidor terá o estado mais recente do avatar daquele jogador, que então é difundido para os servidores vizinhos, que repassa para seus vizinhos e assim por diante. Dessa forma, em algum momento todos servidores - ou boa parte deles - terão o estado daquele avatar. Além disso, quanto mais próximo o servidor estiver da região onde está o avatar, maior a probabilidade dele ter recebido o estado, assim como mais atualizado ele estará.

4.3. Jogadores e servidores

Cada servidor terá uma lista dos endereços e portas dos outros servidores, assim como a região a que cada um está associado. Quando um jogador deseja participar do jogo, deve conectar-se a um servidor qualquer. Se o jogador entrar com um avatar novo, é escolhida alguma região para ele começar a jogar, e o servidor a que ele se conectou irá redirecioná-lo para o servidor correto. O critério de escolha pode ser o número de jogadores em cada região, como também pode ser feita simplesmente uma seleção aleatória da região. Se já tiver jogado antes, será redirecionado para o servidor responsável pela área na qual ele parou de jogar da última vez. Desta forma, aumenta-se a probabilidade do jogador recomeçar a partida com as informações mais recentes a respeito de seu estado, que foi disseminado utilizando push-gossip.

5. Área de interesse gradual

Para que os diferentes jogadores interajam entre si e com as diversas entidades presentes no ambiente do jogo de maneira adequada, é necessário que disponham de réplicas locais destas entidades, cujo estado deve ser o mesmo para todos. A maneira mais simples de fazer isso seria difundir o estado de todas as entidades para todos os jogadores, mas isso geraria uma quantidade alta de tráfego, a depender do número de jogadores participando. Para economizar largura de banda, tanto dos jogadores, quanto dos servidores que os intermediam, é utilizada uma técnica conhecida como gerenciamento de interesse. Esta técnica reduz o número de atualizações que determinado jogador irá receber - e enviar, no caso de uma arquitetura par-a-par.

Em resumo, o gerenciamento de interesse funciona da seguinte forma: para cada mudança de estado de cada entidade, é calculado para quem ela será relevante. Por exemplo, se um avatar situa-se a quilômetros de distância de outro, sem nenhum tipo de vínculo (como grupo, guilda etc.) entre eles, é irrelevante para cada um deles o estado mais recente do outro. Assim, não é necessário que eles troquem suas informações de estado. Este princípio, de localidade, é utilizado como critério principal no gerenciamento de interesse.

O princípio por trás da nova abordagem proposta neste trabalho baseia-se no fato de que, quanto mais distante uma entidade se situar do avatar no ambiente virtual, menor será a exigência por rapidez nas suas atualizações, para aquele avatar. Sendo assim, pode-se receber atualizações de estado de entidades que estão mais distantes com maior intervalo entre elas. Por outro lado, se uma entidade está muito próxima, é desejável que o jogador disponha de seu estado mais recente assim que possível, para poder visualizar quaisquer mudanças rapidamente.

Para atingir este objetivo, é necessário definir alguns parâmetros:

Relevância - valor real entre 0 e 1, inclusive, que determina o quanto o estado de determinada entidade é relevante para um avatar.

Freqüência de atualização - quantidade de atualizações que cada avatar recebe de cada uma das entidades do ambiente virtual por unidade de tempo.

Intervalo normal de atualização - menor intervalo de tempo entre a chegada de duas atualizações de estado consecutivas de uma mesma entidade em um cliente, ou seja, quando a relevância daquela entidade para o avatar daquele cliente é 1. Assim sendo, o intervalo normal determina a freqüência máxima de atualização.

Alcance de visão - determina a que distância as entidades podem estar do avatar, no máximo, para que o jogador possa visualizá-las.

Distância crítica - é o raio do círculo, em torno do avatar, onde todas as entidades têm relevância igual a 1.

Para se enviar o estado de uma entidade para determinado cliente, verifica-se primeiro quando foi o último envio. O próximo instante de envio é então escalonado para ocorrer após um determinado intervalo de tempo. Se a relevância daquele estado for 1, será utilizado o intervalo normal de atualização. Se for menor que 1, divide-se o intervalo normal pela relevância. Por exemplo, seja um jogo em que o intervalo normal de atualização seja de 200 ms. Se o avatar A_i, que acabou de enviar uma atualização de estado para A_j, está a uma distância de A_j tal que sua relevância é 0.5, o próximo envio será depois de um intervalo de 200/0.5, ou seja, 400 ms. Apesar deste intervalo ainda ser uma fração de segundo, representa uma diminuição da freqüência de atualização do estado de A_i em 50%. Como estão a uma distância maior um do outro, e o intervalo foi aumentado de apenas 200 ms, esta variação deverá ser imperceptível para o jogador que controla A_j.

É importante perceber que a atenuação da freqüência de atualização das entidades é compatível com outras técnicas mais complexas de gerenciamento de interesse. Em [Boulanger et al. 2006], são descritos diversos algoritmos de gerenciamento de interesse que poderiam ser ainda melhorados se fosse agregada a idéia de diferentes intervalos de envio baseado na relevância destas atualizações. Geralmente o estado de cada entidade é classificado em um de apenas dois extremos: é relevante ou não é relevante, ignorando-se que há uma vasta gama de valores intermediários. A questão está em como definir esse valor de relevância para cada estado.

O algoritmo de gerenciamento de interesse proposto neste artigo, denominado A^3 (ângulo de visão com área próxima e atenuação de freqüência de atualização), leva em conta três fatores principais:

- Ângulo de visão do avatar, para determinar quais entidades o jogador tem que ser

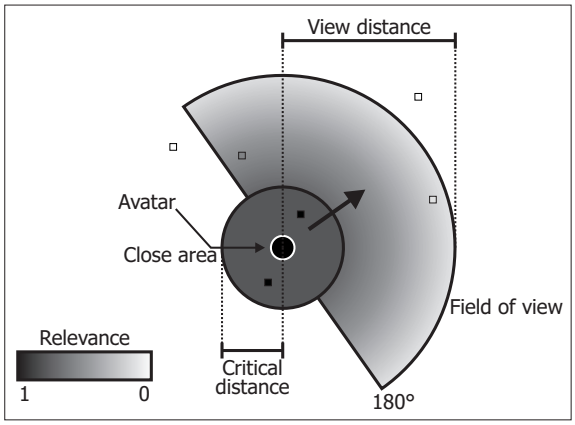

Figura 2. A^3 - área de interesse

capaz de perceber imediatamente, por estarem à sua frente, até a distância que seu alcance de visão permita;
- Área próxima, cujo objetivo é melhorar a qualidade do jogo no espaço mais perto do avatar. Seu raio é a distância crítica, definido anteriormente;
- Atenuação da freqüência de atualizações.

A área de interesse resultante então toma a forma de um setor de círculo, cuja origem é o centro de outro círculo, menor. Este círculo menor é a área próxima do avatar do jogador, que receberá atualizações de estado com intervalo normal de entidades que nela estiverem. Dessa forma, tem-se o estado mais atualizado possível do jogo na região próxima ao avatar. Isso favorece a interação com entidades que estejam perto dele. Mesmo que alguma delas esteja momentaneamente fora do campo de visão do jogador, ela estará disponível caso ele gire seu avatar repentinamente na direção oposta à que está voltado. Na Figura 2, é ilustrada a área de interesse que acaba de ser definida. O Algoritmo 1 define o funcionamento deste gerenciamento de interesse.

Algorithm 1 Calcular relevância da entidade E para o avatar A

$dist \leftarrow distância(A, E)$
if $dist \leq distância\_crítica$ **then**
  $relevância \leftarrow 1$
else
  if A tem E em seu campo de visão **then**
    $relevância \leftarrow 1 - \frac{dist - distância\_crítica}{alcance\_da\_visão - distância\_crítica}$
    if $relevância < 0$ **then**
      $relevância \leftarrow 0$
    end if
  else
    $relevância \leftarrow 0$
  end if
end if

As simulações e seus resultados são apresentados nas seções 6 e 7, respectivamente.

6. Simulação

Para efetuar a simulação do algoritmo proposto, foi necessário primeiro criar um modelo de ambiente virtual a simular, com diversos avatares presentes, pois o algoritmo é baseado nas informações de localização e ângulo de visão. O ambiente consiste em um espaço bidimensional, que corresponde à região gerenciada por um dos servidores. Nela, há diversos avatares presentes, cujo número varia de uma simulação para outra. Cada avatar escolhe aleatoriamente um ponto de destino no ambiente e segue até lá. Ao chegar no destino, permanece parado por um tempo aleatório, que pode ser zero, e então escolhe uma nova localização para se dirigir.

Foi utilizado o simulador de rede ns-2 [McCanne et al.]. Este simulador permite criar código específico da aplicação que será simulada. No caso, foi simulado um servidor, que deveria enviar atualizações de estado para um cliente, responsável por um dos avatares na região. Baseado na localização dos outros avatares e no algoritmo de gerenciamento de interesse escolhido, o servidor decidia quais outros avatares tinham um estado relevante para o cliente em questão. Com isso, obtém-se a ocupação de largura de banda de envio necessária para um único cliente. Não se julgou necessário simular simultaneamente todos os clientes conectados àquele servidor, pois todos os avatares têm o mesmo perfil. Para encontrar a carga total no servidor, basta multiplicar a banda de envio necessária para um cliente pelo número de clientes presentes na região.

Outra questão é que o consumo de largura de banda de envio do servidor é muito maior que o de recepção - se ele recebe n ações, cada uma oriunda de um dos n clientes, é necessário, no pior caso, enviar $O(n^2)$ atualizações de estado, pois cada jogador precisaria do estado de todos os outros. Assim sendo, foi necessário apenas medir a banda de transmissão utilizada.

Em trabalhos como [Yu et al. 2007], [Kim et al. 2005] e [Svoboda et al. 2007], é analisado o tráfego de rede gerado por jogos em larga escala. Baseado nestes trabalhos, e adotando uma postura conservadora, foram decididos os seguinte parâmetros para serem utilizados na simulação:

- Intervalo normal de atualização: 250 ms;
- Tamanho do pacote de atualização de estado de uma única entidade: 100 bytes;
- Duração de cada sessão de jogo simulada: 20 min;
- Área do ambiente virtual: 750 x 750 unidades de área;
- Alcance da visão: 120 unidades de comprimento;
- Distância crítica: 40 unidades de comprimento;
- Ângulo de visão: 180°.

Foram executadas diversas simulações, com o objetivo de comparar os algoritmos de gerenciamento de interesse apresentados. O número de avatares presentes no ambiente foi uma das variáves analisadas, para verificar a escalabilidade. Os algoritmos comparados foram os baseados em círculo, círculo com atenuação, ângulo de visão e o algoritmo proposto, A^3. Para demonstrar o quanto cada um destes reduz o tráfego, foram feitas simulações também em que não é empregado nenhum tipo de gerenciamento de interesse, e o servidor envia para o cliente atualizações de estado de todas as outras entidades do jogo.

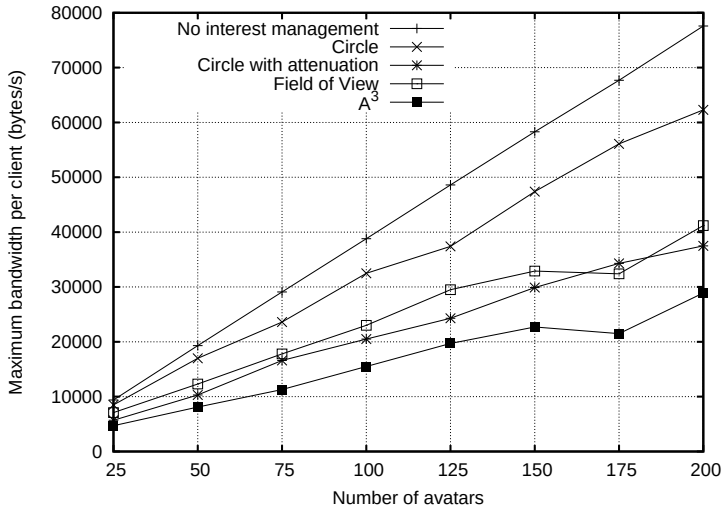

Figura 3. Resultados da simulação: uso máximo de largura de banda

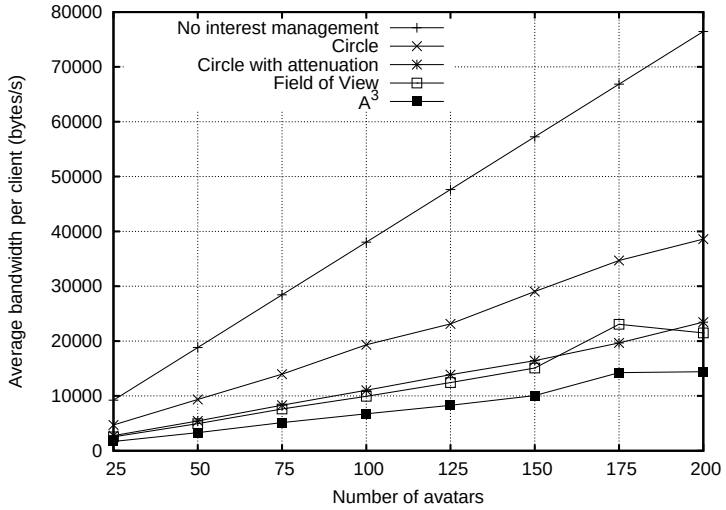

Figura 4. Resultados da simulação: uso médio de largura de banda

7. Resultados

Os resultados foram coletados da seguinte maneira: para encontrar a largura de banda utilizada em média para envio, foram somados todos os pacotes de cada sessão e dividido pelo tempo que foi simulado; para determinar a largura de banda máxima utilizada, foi verificado, segundo a segundo, quantos bytes foram enviados e foi selecionado o máximo.

Nos gráficos das Figuras 3 e 4, são apresentados os resultados obtidos de largura de banda máxima e média, respectivamente, utilizada com os quatro algoritmos simuladas - área em círculo (C), área em círculo com atenuação (C & A), área do campo de visão (FoV) e área do campo de visão mais área próxima mais atenuação (A^3) - além de mostrar quanto seria a largura de banda utilizada se nenhuma técnica fosse empregada (None). Os valores estão em bytes/s.

Apenas usando diferentes freqüências de atualização no gerenciamento de interesse baseado em círculo, reduziu-se em 41.59% a largura de banda de envio utilizada em

média pelo servidor por cliente. A utilização máxima de largura de banda também foi reduzida, em 36.19%. Estes valores representam a média de redução de uso de largura de banda para os diferentes números de clientes.

No que diz respeito ao algoritmo proposto A^3, obteve-se uma redução de uso médio de largura de banda de envio de 63.51% e 33.58%, comparado respectivamente com o algoritmo de área de interesse circular e baseado em ângulo de visão. Reduziu-se também o pico de utilização em 52.03% e 33.10%, comparado com os mesmos algoritmos. Na Tabela 1, são mostrados os percentuais médios de economia de largura de banda máxima e média com o algoritmo A^3, em relação aos outros algoritmos apresentados.

Tabela 1. Economia de largura de banda com o algoritmo A^3

Utilização	None	C	C & A	FoV
Máxima	60.10%	52.03%	24.81%	33.10%
Média	81.64%	63.51%	37.48%	33.58%

Observou-se também que os valores médio e máximo observados diferem, mesmo quando não é utilizado nenhum algoritmo de gerenciamento de interesse, ou seja, o cliente recebe atualizações de estado de todas as entidades presentes no jogo, com a freqüência normal. Além disso, com 200 avatares no ambiente, com estado de 100 bytes, cuja atualização é enviada a cada 250 ms, o servidor deveria alocar $199 \times 100 \times 4$ bytes/s para cada cliente, ou seja, 79600 bytes/s. No entanto, observou-se que a utilização máxima e média, com 200 avatares presentes e nenhum gerenciamento de interesse, foi de 77600 e 76458, respectivamente. Isso acontece porque o ns-2 é um simulador de eventos discreto, e o servidor simulado foi programado para checar o schedule de envios a cada 10 ms. Em conseqüência disto, cada atualização de estado pode ter tido seu intervalo aumentado em até 10 ms, o que explica os valores encontrados.

8. Conclusões

Foi proposto um modelo preliminar de distribuição do servidor para jogos maciçamente multijogador. Foram apresentadas também algumas técnicas que se pretende utilizar para economizar largura de banda dos nodos que fariam deste sistema servidor, assim como foi descrito um algoritmo desenvolvido para gerenciamento de interesse, o A^3, cuja idéia principal é adaptar a freqüência de atualização de estado das entidades do jogo de acordo com sua relevância para o cliente que as receberá. O formato da área de interesse utilizada pelo algoritmo A^3 consiste em um setor de círculo, correspondente ao campo de visão do jogador, mais um círculo de raio menor, que corresponde à área próxima ao avatar daquele jogador. O objetivo deste círculo menor é o de manter o estado naquela região, que é considerada crítica, o mais atualizado possível. Somando-se essas características, chegamos a um algoritmo que obteve redução da utilização máxima da banda de envio do servidor de 52.03% e 33.10%, comparados com o gerenciamento de interesse baseado em círculo e em campo de visão, respectivamente, e de 63.51% e 33.58% de utilização média, comparados com os mesmos algoritmos.

9. Andamento

No plano de estudos e pesquisa (PEP), foi apresentado um cronograma com as tarefas que se pretende realizar com o fim de concluir este trabalho. Daquelas tarefas, as seguintes

foram concluídas:

1. Levantamento bibliográfico do estado-da-arte das técnicas de distribuição do suporte a MMG;
2. Levantamento bibliográfico de técnicas de distribuição em geral, que possam ser aplicadas a jogos MMG;
3. Análise e categorização destas técnicas;
4. Elaboração de um modelo arquitetural de suporte distribuído a jogos MMG;
5. (em parte) Implementação de um protótipo com o fim de testes;
6. (em parte) Levantamento dos simuladores mais indicados para efetuar tais testes;
7. (em parte) Execução dos testes em um dos simuladores pesquisados;
8. (em parte) Análise dos resultados;
9. (iniciada) Escrita da dissertação;
10. (iniciada, alguns já submetidos) Escrita de artigos;

Referências

ArenaNet (2005). Guild wars. http://www.guildwars.com/.

Assiotis, M. and Tzanov, V. (2006). A distributed architecture for MMORPG. *Proceedings of 5th ACM SIGCOMM workshop on Network and system support for games*.

Blizzard (2004). World of warcraft. http://www.worldofwarcraft.com/.

Boulanger, J., Kienzle, J., and Verbrugge, C. (2006). Comparing interest management algorithms for massively multiplayer games. *Proceedings of 5th ACM SIGCOMM workshop on Network and system support for games*.

Cecin, F., Real, R., de Oliveira Jannone, R., Geyer, C., Martins, M., and Barbosa, J. (2004). FreeMMG: A Scalable and Cheat-Resistant Distribution Model for Internet Games. *IEEE Int. Sym. on Distributed Simulation and Real-Time Applications*, pages 83–90.

Crippa, M., Cecin, F., and Geyer, C. Peer-to-peer support for instance-based massively multiplayer games. Presented at 6th Brazilian Symposium on Computer Games and Digital Entertainment, São Leopoldo, Brazil, 2007.

El-Sayed, A. (2004). Application-Level Multicast Transmission Techniques over the Internet. *University of Paris*.

Fiedler, S., Wallner, M., and Weber, M. (2002). A communication architecture for massive multiplayer games. *Proceedings of the 1st workshop on Network and system support for games*, pages 14–22.

Hampel, T., Bopp, T., and Hinn, R. (2006). A peer-to-peer architecture for massive multiplayer online games. *Proceedings of 5th ACM SIGCOMM workshop on Network and system support for games*.

Iimura, T., Hazeyama, H., and Kadobayashi, Y. (2004). Zoned federation of game servers: a peer-to-peer approach to scalable multi-player online games. *Proceedings of ACM SIGCOMM 2004 workshops on NetGames' 04: Network and system support for games*, pages 116–120.

Kermarrec, A., Massoulié, L., and Ganesh, A. (2003). Probabilistic Reliable Dissemination in Large-Scale Systems. *IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS*, pages 248–258.

Kim, J., Choi, J., Chang, D., Kwon, T., Choi, Y., and Yuk, E. (2005). Traffic characteristics of a massively multi-player online role playing game. *Proceedings of 4th ACM SIGCOMM workshop on Network and system support for games*, pages 1–8.

Knutsson, B., Lu, H., Xu, W., and Hopkins, B. (2004). Peer-to-peer support for massively multiplayer games. *IEEE Infocom*.

McCanne, S., Floyd, S., et al. Network simulator ns-2. *The Vint project, available for download at http://www.isi.edu/nsnam/ns*.

Minson, R. and Theodoropoulos, G. (2005). An adaptive interest management scheme for distributed virtual environments. *Principles of Advanced and Distributed Simulation, 2005. PADS 2005. Workshop on*, pages 273–281.

Morgan, G., Lu, F., and Storey, K. (2005). Interest management middleware for networked games. *Symposium on Interactive 3D Graphics: Proceedings of the 2005 symposium on Interactive 3D graphics and games*, 3(06):57–64.

Morse, K. et al. (1996). Interest management in large-scale distributed simulations. *Technical Report ICS-TR-96-27, University of California, Irvine*.

NCsoft (2003). Lineage II. http://www.lineage2.com/.

Ng, B., Si, A., Lau, R., and Li, F. (2002). A multi-server architecture for distributed virtual walkthrough. *Proceedings of the ACM symposium on Virtual reality software and technology*, pages 163–170.

Rak, S. and Van Hook, D. (1996). Evaluation of grid-based relevance filtering for multicast group assignment. *Proc. of 14th DIS workshop*, pages 739–747.

Rieche, S., Fouquet, M., Niedermayer, H., Petrak, L., Wehrle, K., and Carle, G. (2007). Peer-to-Peer-based Infrastructure Support for Massively Multiplayer Online Games. *Consumer Communications and Networking Conference, 2007. CCNC 2007. 2007 4th IEEE*, pages 763–767.

Schiele, G., Suselbeck, R., Wacker, A., Hahner, J., Becker, C., and Weis, T. (2007). Requirements of Peer-to-Peer-based Massively Multiplayer Online Gaming. *Proceedings of the Seventh IEEE International Symposium on Cluster Computing and the Grid*, pages 773–782.

Svoboda, P., Karner, W., and Rupp, M. (2007). Traffic Analysis and Modeling for World of Warcraft. *Communications, 2007. ICC'07. IEEE International Conference on*, pages 1612–1617.

Yan, J. and Randell, B. (2005). A systematic classification of cheating in online games. *Proceedings of 4th ACM SIGCOMM workshop on Network and system support for games*, pages 1–9.

Yu, Y., Li, Z., Shi, L., Chen, Y., and Xu, H. (2007). Network-Aware State Update For Large Scale Mobile Games. *Computer Communications and Networks, 2007. ICCCN 2007. Proceedings of 16th International Conference on*, pages 563–568.

Zou, L., Ammar, M., and Diot, C. (2001). An evaluation of grouping techniques for state dissemination in networked multi-user games. *Modeling, Analysis and Simulation of Computer and Telecommunication Systems, 2001. Proceedings. Ninth International Symposium on*, pages 33–40.