UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

CARLOS EDUARDO BENEVIDES BEZERRA

<PEGAR DO PLANODOC NO NOTE>

Dissertação apresentada como requisito parcial
para a obtenção do grau de
Mestre em Ciência da Computação

Prof. Dr. Cláudio Fernando Resin Geyer
Orientador

Porto Alegre, janeiro de 2009

CIP – CATALOGAÇÃO NA PUBLICAÇÃO

Bezerra, Carlos Eduardo Benevides

<PEGAR DO PLANODOC NO NOTE> / Carlos Eduardo Benevides Bezerra. – Porto Alegre: PPGC da UFRGS, 2009.

39 f.: il.

Dissertação (mestrado) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR–RS, 2009. Orientador: Cláudio Fernando Resin Geyer.

1. Jogos online maciçamente multijogador. 2. MMOG. 3. Simulação interativa distribuída. I. Geyer, Cláudio Fernando Resin. II. Título.

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
Reitor: Prof. José Carlos Ferraz Hennemann
Pró-Reitor de Coordenação Acadêmica: Prof. Pedro Cezar Dutra Fonseca
Pró-Reitora de Pós-Graduação: Profª. Valquíria Linck Bassani
Diretor do Instituto de Informática: Prof. Philippe Olivier Alexandre Navaux
Coordenador do PPGC: Prof. Carlos Alberto Heuser
Bibliotecária-chefe do Instituto de Informática: Beatriz Regina Bastos Haro

AGRADECIMENTOS

<TO-DO>

SUMÁRIO

LISTA DE ABREVIATURAS E SIGLAS

ADI	*Área de Interesse*
API	*Application Programming Interface*
CVE	*Collaborative Virtual Environment*
CPU	*Central Processing Unit*
DHT	*Distributed Hash Table*
DIS	*Distributed Interactive Simulation*
FPS	*First-Person Shooter*
IP	*Internet Protocol*
J2SE	*Java 2 Standard Edition*
MMG	*Massively Multiplayer Game*
MMOFPS	*Massively Multiplayer Online First-Person Shooter*
MMORPG	*Massively Multiplayer Online Role-Playing Game*
MMORTS	*Massively Multiplayer Online Real-Time Strategy*
NVE	*Networked Virtual Environment*
RTP	*Real-time Transport Protocol*
RTS	*Real-Time Strategy*
TCP	*Transmission Control Protocol*
TSS	*Trailing State Synchronization*
UDP	*Unreliable Datagram Protocol*
URL	*Uniform Resource Locator*

LISTA DE FIGURAS

RESUMO

<lalalalalalalalá ... fazer resumo lalalalalalá....>

A popularização das tecnologias de acesso à Internet por "banda larga" suportam a crescente proliferação de aplicações do tipo "par-a-par" (peer-to-peer), onde o usuário doméstico, tipicamente consumidor de informação, passa também a atuar como provedor. De forma simultânea, há uma popularização crescente dos jogos em rede, especialmente dos "jogos maciçamente multijogador" (MMG ou *massively multiplayer game*) onde milhares de jogadores interagem, em tempo real, em mundos virtuais de estado persistente. Os MMGs disponíveis atualmente, como *EverQuest* e *Ultima Online*, são implementados como sistemas centralizados, que realizam toda a simulação do jogo no "lado servidor". Este modelo propicia controle de acesso ao jogo pelo servidor, além de ser muito resistente a jogadores trapaceiros. Porém, a abordagem cliente-servidor não é suficientemente escalável, especialmente para pequenas empresas ou projetos de pesquisa que não podem pagar os altos custos de processamento e comunicação dos servidores de MMGs centralizados. Este trabalho propõe o FreeMMG, um modelo híbrido, cliente-servidor e par-a-par, de suporte a jogos maciçamente multijogador de estratégia em tempo real (MMORTS ou *massively multiplayer online real-time strategy*). O servidor FreeMMG é escalável pois delega a maior parte da tarefa de simulação do jogo para uma rede par-a-par, formada pelos clientes. É demonstrado que o FreeMMG é resistente a certos tipos de trapaças, pois cada segmento da simulação distribuída é replicado em vários clientes. Como protótipo do modelo, foi implementado o jogo *FreeMMG Wizards*, que foi utilizado para gerar testes de escalabilidade com até 300 clientes simulados e conectados simultaneamente no mesmo servidor. Os resultados de escalabilidade obtidos são promissores, pois mostram que o tráfego gerado em uma rede FreeMMG, entre servidor e clientes, é significativamente menor se comparado com uma alternativa puramente cliente-servidor, especialmente se for considerado o suporte a jogos maciçamente multijogador de estratégia em tempo real.

Palavras-chave: Jogos online maciçamente multijogador, MMOG, simulação interativa distribuída.

FreeMMG: A Client-Server and Peer-to-Peer Support Architecture for Massively Distributed Games

ABSTRACT

The increasing adoption of "broadband" Internet access technology fosters the increasing development of networked "peer-to-peer" applications, where the end-user, usually the consumer of information, begins to act also as a producer of information. Simultaneously, there is a growing popularity of networked games, especially "massively multiplayer games" (MMGs) where thousands of players interact, in real time, in persistent-state virtual worlds. State-of-the-art massively multiplayer games such as EverQuest and Ultima Online are currently implemented as centralized, client-server systems, which run the entire game simulation on the "server-side". Although this approach allows the development of commercially viable MMG services, with game access control and player cheating prevention, the processing and communications costs associated with running a MMG server are often too high for small companies or research projects. This dissertation proposes FreeMMG, a mixed peer-to-peer and client-server approach to the distribution aspect of MMGs. It is argued that the FreeMMG model supports scalable, cheat-resistant, massively multiplayer real-time strategy games (MMORTS games) using a lightweight server that delegates the bulk of the game simulation to a peer-to-peer network of game clients. It is shown that FreeMMG is resistant to certain kinds of cheating attempts because each segment of the distributed simulation is replicated by several clients. A working prototype game called FreeMMG Wizards is presented, together with scalability test results featuring up to 300 simulated game clients connected to a FreeMMG server. The obtained results are promising, by showing that the generated server traffic in a FreeMMG network, between the server and the clients, is substantially lower if compared with purely client-server alternatives, especially if the support for massively multiplayer real-time strategy games is considered.

Palavras-chave: massively multiplayer games, distributed interactive simulation, peer-to-peer systems, real-time strategy games.

1 INTRODUÇÃO

2 CONTEXTO, ESTADO DA ARTE E MOTIVAÇÃO

3 MODELO BASE

4 A^3: UM ALGORITMO DE OTIMIZAÇÃO POR ÁREA DE INTERESSE

Traditionally, a central server is used to provide support to MMGs (massively multiplayer games), where the number of participants is in the order of tens of thousands. Much work has been done trying to create a fully peer-to-peer model to support this kind of application, in order to minimize the maintenance cost of its infra-structure, but critical questions remain. Examples of the problems relative to peer-to-peer MMG support systems are: vulnerability to cheating, overload of the upload links of the peers and difficulty to maintain consistency of the simulation among the participants. In this work, it is proposed the utilization of geographically distributed lower-cost nodes, working as a distributed server to the game. The distribution model and some related works are also presented. To address the communication cost imposed to the servers, we specify a novell refinement to the area of interest technique, significantly reducing the necessary bandwidth. Simulations have been made with ns-2, comparing different area of interest algorithms. The results show that our approach achieves the least bandwidth utilization, with a 33.10% maximum traffic reduction and 33.58% average traffic reduction, when compared to other area of interest algorithms.

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 (**?**).

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; 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 propõe uma técnica para reduzir o consumo de largura de banda causado pelo tráfego do jogo entre os servidores e os clientes, diminuindo a quantidade de recursos necessários, através de um refinamento da técnica de gerenciamento de interesse (**?**) 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.

O artigo está dividido da seguinte maneira: na seção 4.1 são citados alguns trabalhos relacionados onde buscou-se distribuir o servidor do jogo; na seção 4.2, são apresentadas as definições de alguns conceitos utilizados ao longo do texto; na seção 4.3, é descrito o modelo de distribuição proposto; na seção 4.4 é apresentada a otimização proposta para reduzir o tráfego sem comprometer a qualidade do jogo; nas seções 4.6 e 4.7 é descrita a simulação realizada para validar a técnica proposta e os resultados obtidos, respectivamente e, na seção 4.8, são apresentadas as conclusões a que se chegou neste trabalho.

4.1 Related Work

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, 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; HAZEYAMA; KADOBAYASHI, 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 (**?**) 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 é considerado mais adequado para utilização de técnicas de gerenciamento de interesse dos jogadores.

Um exemplo deste tipo de modelo é descrito em (ASSIOTIS; 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; 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 (**???**MORGAN; LU; STOREY, 2005) e (MINSON; THEODOROPOULOS, 2005) podem ser citados. Em (**?**), 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 (**?**), 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; LU; STOREY, 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 (**?**), 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 (**?**), 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 (**?**).

4.2 Definitions

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.

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.

4.3 Distribution Model

Este trabalho baseia-se em 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 (Fig. 4.1). 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

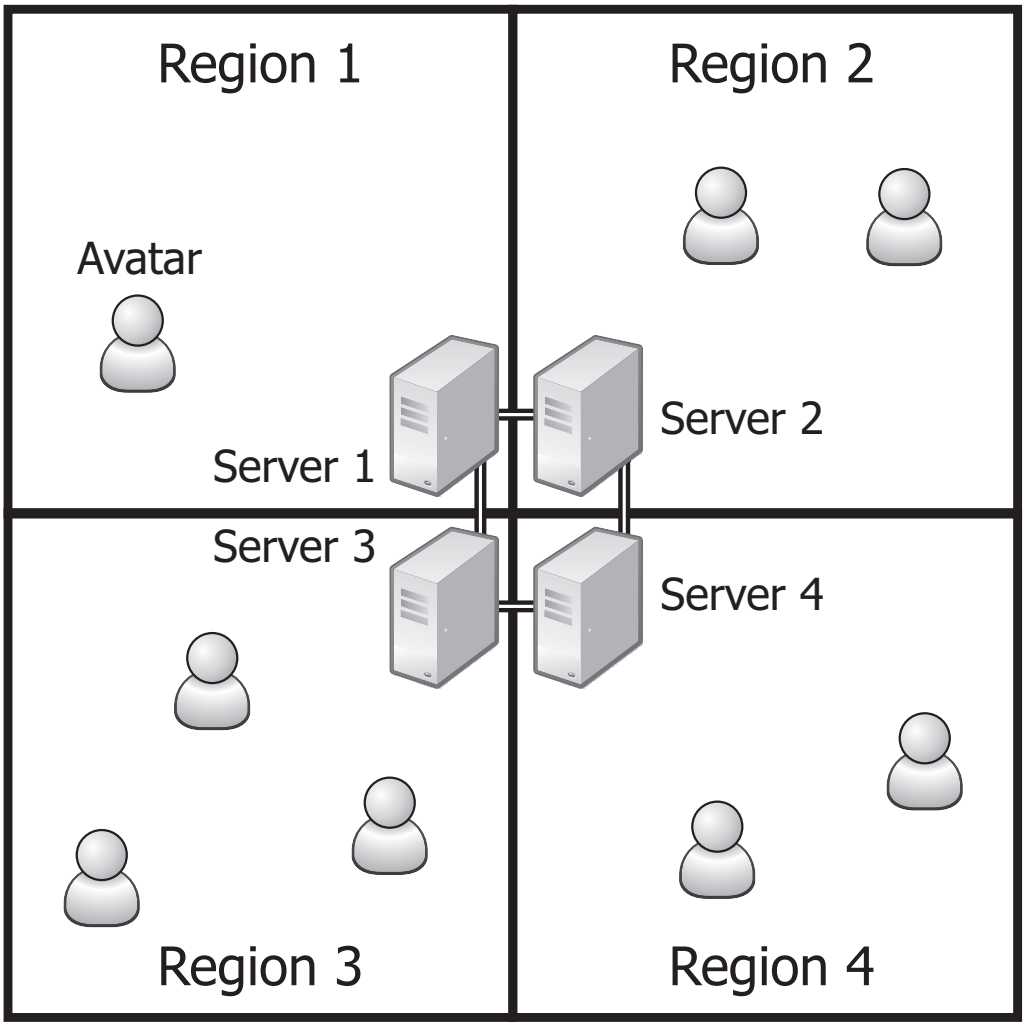


Figura 4.1: Distribution model

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.

Desta maneira, os detalhes deste mecanismo não irão implicar em mudanças rele-

vantes para o 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.

4.4 Area of Interest Algorithms

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.

Os algoritmos descritos nas próximas seções baseiam-se, entre outras coisas, na distância euclidiana entre cada avatar e todas as outras entidades presentes no ambiente virtual. Isso poderia gerar um problema de escalabilidade, porém está sendo suposta uma arquitetura distribuída, onde tal processamento poderá e deverá ser paralelizado. No modelo de distribuição definido anteriormente, cada servidor controla uma região do mapa. Por conseguinte, cada um deles gerencia apenas um subconjunto das entidades do jogo, verificando somente as distâncias entre cada par delas, além das entidades que estiverem em uma região vizinha, próximos à sua fronteira.

Nas sessões seguintes, serão descritos algumas versões desta técnica, tais como gerenciamento de interesse baseado em área circular e em ângulo de visão do avatar. Na seção 4.5 é introduzida a abordagem de atenuação da freqüência de atualizações, onde será descrita em detalhes o algoritmo proposto.

4.4.1 Área circular

A forma mais simples de executar gerenciamento de interesse consiste em definir uma área em forma de círculo, cujo centro é definido pelas coordenadas da localização do avatar no ambiente virtual. Após isso, é calculada a distância euclidiana entre cada avatar e cada uma das outras entidades presentes no mundo do jogo. Seja o avatar A_i, cuja área de interesse é um círculo de raio rad_i. Se o avatar A_j estiver a uma distância menor que rad_i de A_i, então suas atualizações de estado serão relevantes. A_i não receberá

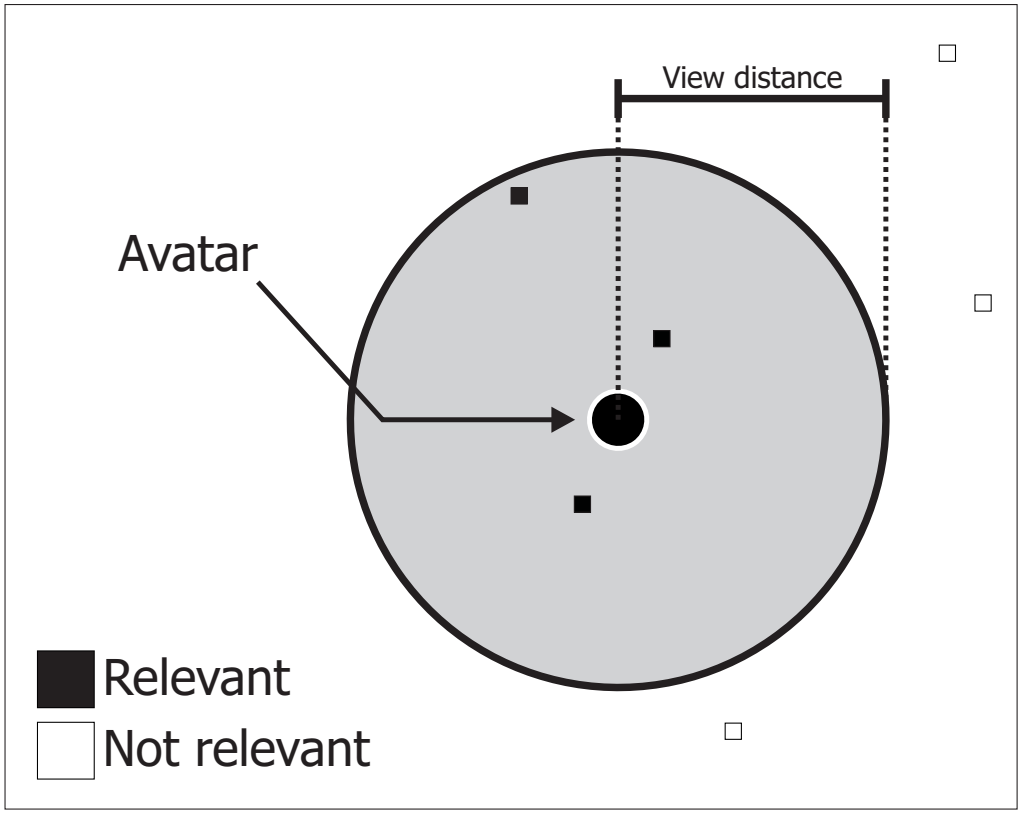


Figura 4.2: Circular area of interest

atualizações de estado de entidades que estejam a uma distância maior. A figura 4.2 ilustra este tipo de área de interesse.

4.4.2 Ângulo de visão

Outra maneira, um pouco mais refinada, de gerenciar o interesse dos avatares consiste em levar em conta o que o jogador pode visualizar, ou seja, seu ângulo de visão. A área dentro de onde esse jogador perceberá mudanças relevantes pode ser definida como um setor de círculo. É similar à área em formato de círculo definida anteriormente, porém leva em consideração que o jogador só pode visualizar objetos que estão situados à frente de seu avatar.

Uma questão a ser considerada, no entanto, é que o jogador não irá receber atualizações de estado de entidades imediatamente atrás de seu avatar, podendo comprometer o jogo. Se o avatar girar 180° em torno de seu próprio eixo rapidamente, pode ser que não veja determinada entidade que deveria estar ali, necessitando de certo tempo para receber o estado dela. Isto acontece porque, apesar desta entidade ter estado próximo do avatar, ele não recebeu suas informações ainda pois antes ela estava atrás dele, fora do seu campo de visão. Na figura 4.3 é ilustrado como seria uma área de interesse que levaria em consideração o campo de visão do jogador.

4.5 Graded Area of Interest

O princípio por trás da abordagem proposta aqui 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 atual-

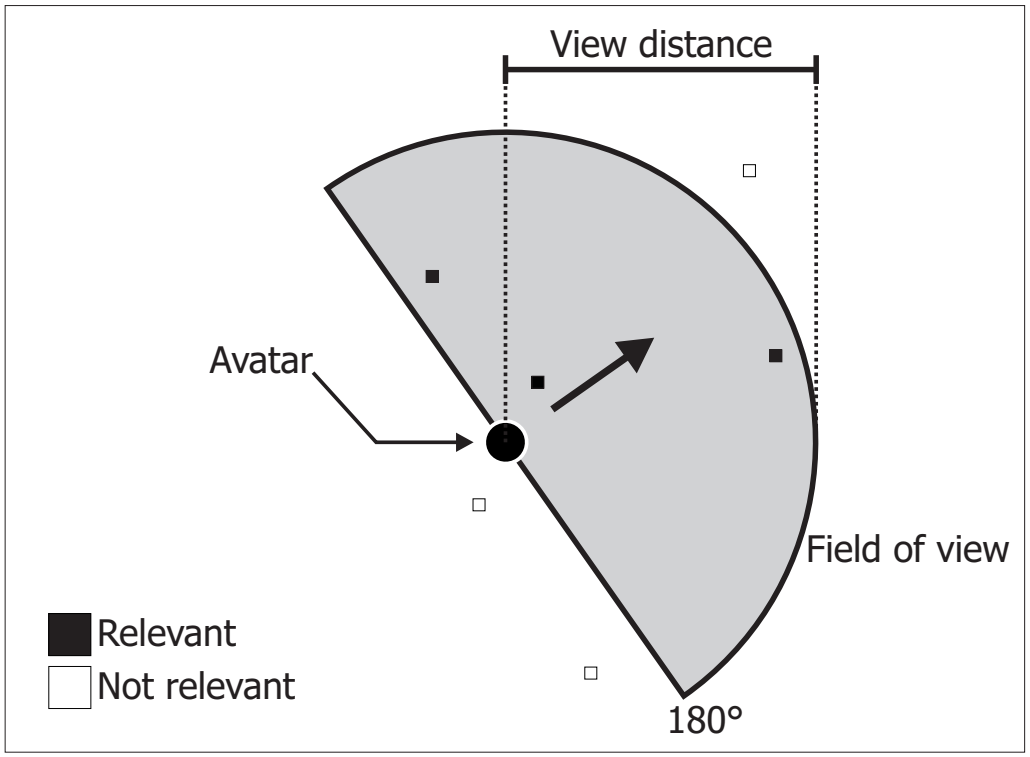


Figura 4.3: Field of view based area of interest

izações 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 se-

gundo, 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 (**?**), 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. Nas seções seguintes, serão apresentados dois exemplos de algoritmos que definem o método de se obter esse valor, assim como o tipo de área utilizado. Na seção 4.5.2, é especificado o A^3, que é um algoritmo original de gerenciamento de interesse, que, dentre outros princípios, emprega a atenuação da freqüência de atualização. As simulações e seus resultados são apresentados nas seções 4.6 e 4.7, respectivamente.

4.5.1 Círculo com atenuação

Um exemplo simples de utilização de intervalos variados de atualização baseado em relevância seria utilizando a área de interesse em formato de círculo. Para obter-se a relevância de uma entidade em relação a um avatar, pode-se fazer com que seu valor seja 1 quando a entidade estiver na mesma posição do avatar e ir diminuindo gradualmente à medida em que se afasta, até chegar a 0, quando para de diminuir não importa o quanto mais se afaste. Essa é uma maneira que, apesar de simples, demonstrou uma significante redução no tráfego entre clientes e servidores. Na figura 4.4, é ilustrado como seria a área de interesse com atenuação gradual da freqüência de atualização das entidades para um determinado avatar.

4.5.2 Algoritmo A^3

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 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 4.5, é ilustrada a área de interesse que acaba de ser definida.

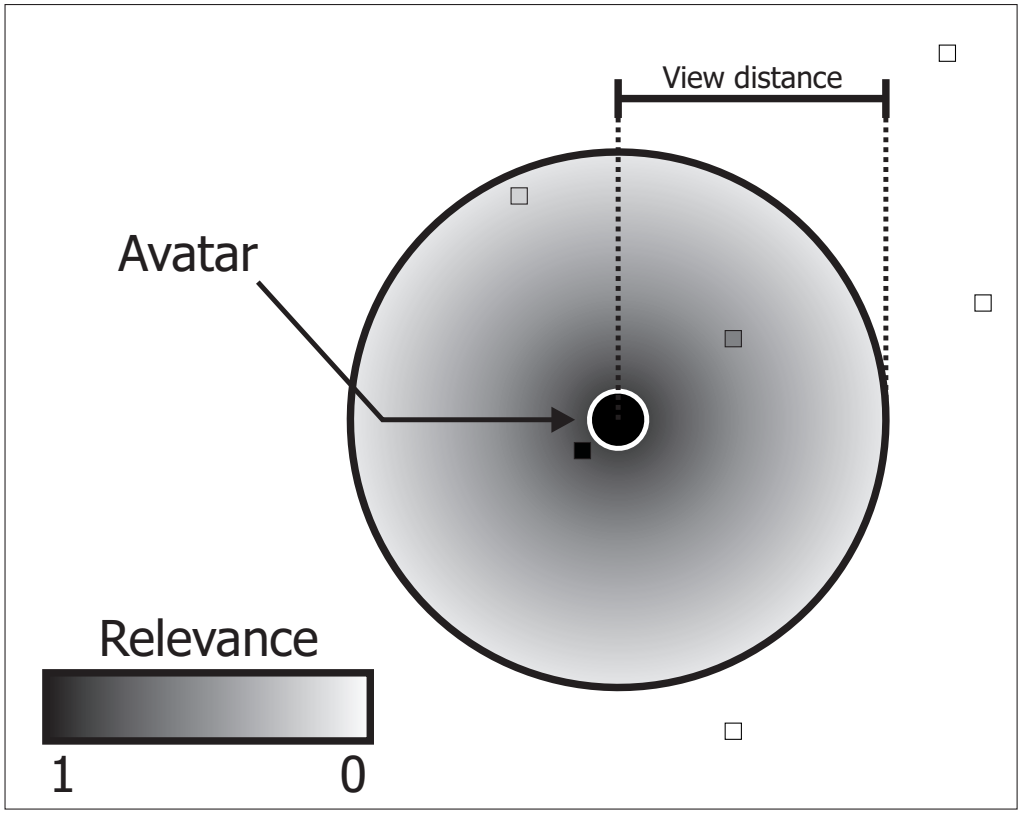


Figura 4.4: Circular area of interest with update frequency atenuation

Quanto às entidades que estiverem fora da área próxima, mas ainda dentro do ângulo de visão, será calculada sua relevância. Propõe-se que a relevância de cada entidade diminua gradualmente de acordo com a distância entre ela e o avatar do jogador em questão. Quanto mais longe, menos freqüentes serão as atualizações de estado. Isso é possível porque mesmo que o intervalo de atualização seja duplicado, ainda será de uma fração de segundo, o que será dificilmente perceptível por um jogador cujo avatar está situado a uma grande distância da entidade em questão. Além disso, pequenos atrasos entre a chegada das atualizações de estado podem ser facilmente mascarados através de técnicas de interpolação, como dead-reckoning (**?**). O algoritmo 1 define o funcionamento deste gerenciamento de interesse.

4.6 Simulation

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; FLOYD et al., 2006). 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

Algorithm 1 Calculate relevance of entity E to avatar A

$dist \leftarrow distance(A, E)$
if $dist \leq distância\_crítica$ **then**
  $relevance \leftarrow 1$
else
  if A can see E in its field of view **then**
    $relevance \leftarrow 1 - \frac{dist - distância\_crítica}{alcance\_da\_visão - distância\_crítica}$
    if $relevance < 0$ **then**
      $relevance \leftarrow 0$
    end if
  else
    $relevance \leftarrow 0$
  end if
end if

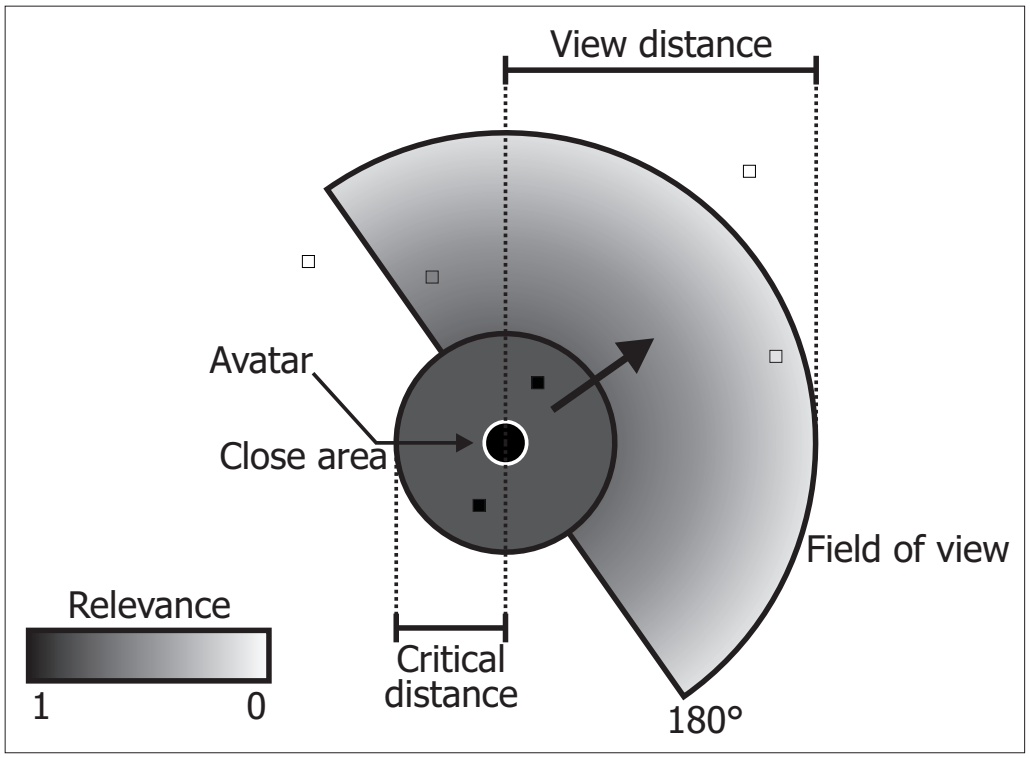


Figura 4.5: A^3 area of interest

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 precis-

aria do estado de todos os outros. Assim sendo, foi necessário apenas medir a banda de transmissão utilizada.

Em trabalhos como (**?**), (**?**) e (**?**), 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.

4.7 Results

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.

Nas tabelas 4.1 e 4.2, são apresentados os dados coletados 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. Nas figuras 4.6 e 4.7 são mostrados os gráficos correspondentes.

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 4.1: Largura de banda máxima utilizada

Avatars	None	C	C & A	FoV	A^3
25	9400	8500	5700	7100	4700
50	19300	17000	10300	12300	8100
75	29100	23600	16600	17800	11300
100	38800	32500	20500	23000	15500
125	48600	37400	24300	29500	19700
150	58300	47400	29900	32900	22700
175	67700	56100	34300	32400	21500
200	77600	62300	37500	41200	28900

Tabela 4.2: Largura de banda utilizada em média

Avatars	None	C	C & A	FoV	A^3
25	9221	4715	2759	2534	1700
50	18826	9350	5442	4949	3303
75	28432	13963	8315	7619	5137
100	38037	19324	11029	9928	6739
125	47642	23138	13871	12434	8290
150	57247	29031	16432	15085	10062
175	66853	34697	19661	23060	14250
200	76458	38600	23450	21491	14413

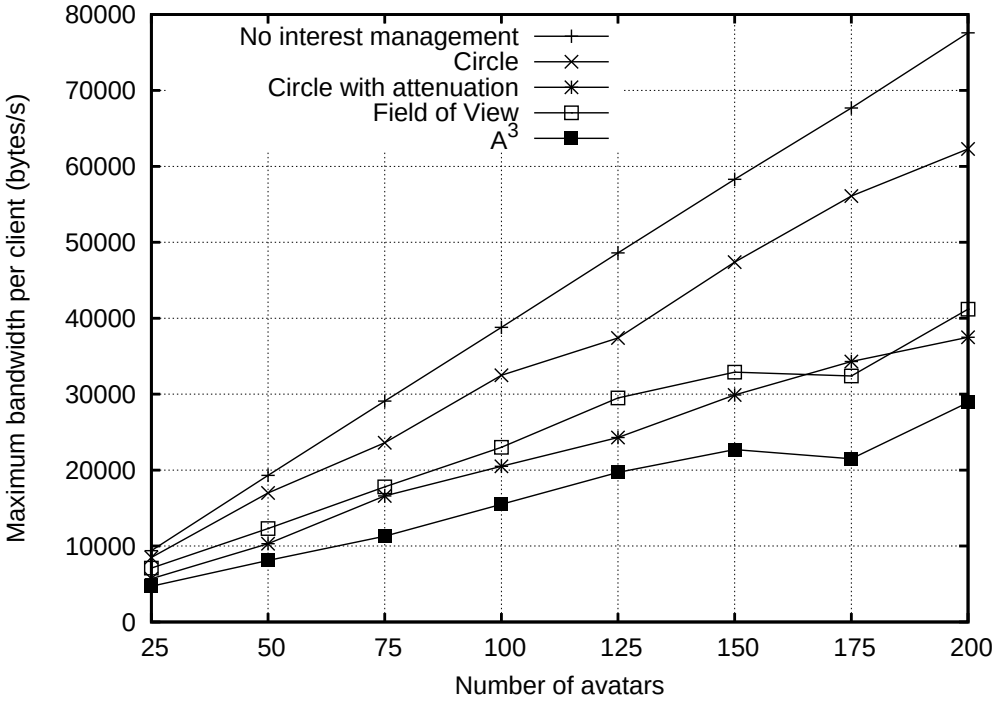


Figura 4.6: Simulation Results: maximum bandwidth usage

tabela 4.3, 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.

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

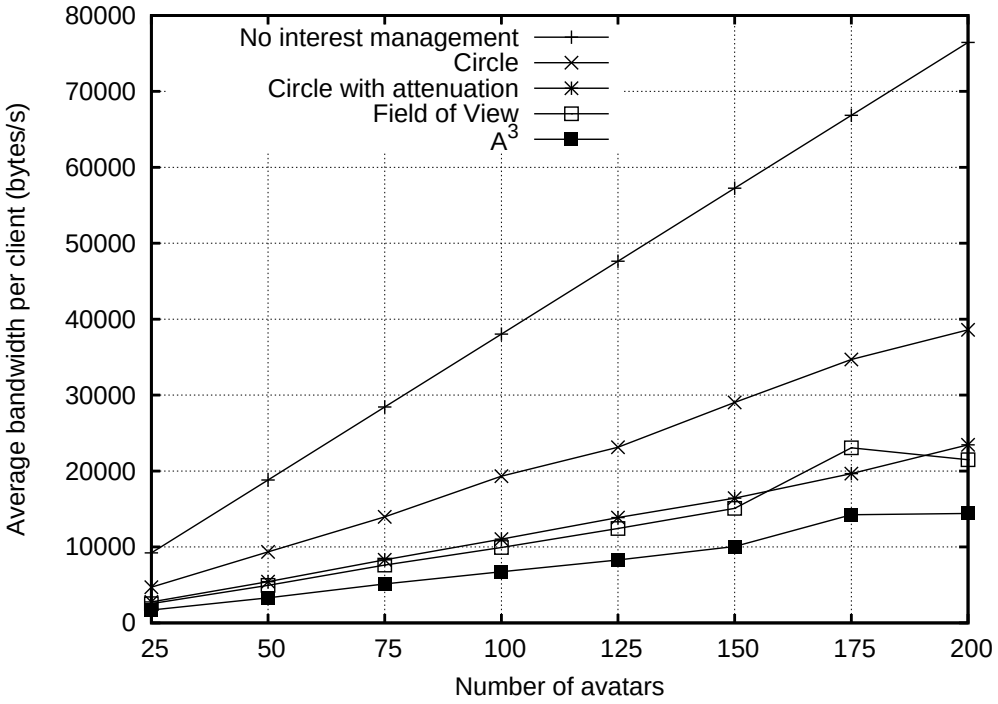


Figura 4.7: Simulation Results: average bandwidth usage

Tabela 4.3: 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%

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.

4.8 Conclusion

Foi apresentado um algoritmo de 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 receberá as atualizações. 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.

[TODO:cap/lista de termos e definições]

5 BALANCEAMENTO DE CARGA

A principal característica dos jogos maciçamente multijogador é a grande quantidade de jogadores, chegando a ter dezenas ou centenas de milhares de participantes simultaneamente [TODO:ref]. Essa grande quantidade de jogadores interagindo entre si gera um tráfego na rede de suporte que tem crescimento quadrático em relação ao número de jogadores, no pior caso [TODO:referenciar/provar].

Quando se utiliza uma arquitetura cliente-servidor, é necessário que o servidor intermedie a comunicação entre cada par de jogadores – supondo que se pretende prover ao jogo garantias de consistência e resistência a trapaça. Obviamente, esse servidor terá uma grande carga de comunicação e, conseqüentemente, deverá ter recursos (largura de banda disponível) proporcional à demanda do jogo.

[TODO:largura de banda X CPU]

A questão posta aqui é que, quando se faz uso de um servidor distribuído, principalmente com recursos escassos, que é a proposta deste trabalho, é necessário otimizar o uso destes recursos, atribuindo a cada servidor uma carga que ele seja capaz de suportar. Dessa forma, não importando a qual servidor cada jogador estiver conectado, sua experiência de jogo será semelhante, no que diz respeito ao tempo de resposta para suas ações e o tempo que leva para ser notificado de ações de outros jogadores, assim como de mudanças de estado no ambiente virtual do jogo.

[aqui já fala do n ao quadrado, por cima]

Uma idéia inicial poderia ser a de distribuir os jogadores entre servidores, de maneira que o número de jogadores em cada servidor fosse proporcional à largura de banda daquele servidor. No entanto, essa distribuição não funcionaria, pelo fato de que a carga causada pelos jogadores depende também do quanto os jogadores estão interagindo entre si. Por exemplo, se os avatares de dois jogadores estiverem muito distantes um do outro, provavelmente não haverá interação entre eles e, portanto, o servidor precisará apenas atualizar cada um a respeito de suas próprias ações. No entanto, se estes avatares estiverem próximos, cada jogador deverá ser atualizado não apenas a respeito de suas próprias ações, como também das ações do outro jogador.

Percebe-se, então, que quando os avatares estão distantes uns dos outros, o tráfego cresce linearmente com o número de jogadores. Porém, se eles estão próximos uns dos outros, o tráfego cresce quadraticamente. Por fim, ambas as funções de crescimento do número de mensagens podem estar presentes no mesmo jogo se, em alguns lugares do ambiente virtual, os avatares estiverem próximos e, em outros lugares, eles estiverem distantes.

A grande maioria dos jogos multijogador apresenta a característica de localidade, no que diz respeito à distribuição dos jogadores no ambiente virtual. Existem alguma exceções, como GuildWars [TODO:ref], em que apenas grupos com um número limitado

de jogadores podem iniciar uma partida. Este tipo de jogo é baseado no modelo de instâncias, onde todos os avatares dos jogadores se encontram em um espaço social, de interação limitada, menos dinâmica e, portanto, com tráfego de rede algumas ordens de grandeza menor [TODO:ref/prove]. Quando pretende-se iniciar uma partida "real", os jogadores requisitam ao servidor que seja criado um grupo de ação. Dessa forma, impede-se que um número teoricamente ilimitado de jogadores interajam entre si, sobrecarregando o servidor.

Normalmente, porém, os jogadores podem mover seus avatares livremente através do mundo do jogo. Isso torna possível a formação de pontos de interesse – também conhecidos como *hotspots* [TODO:ref] – ao redor dos quais os jogadores se concentram mais do que em outras regiões do ambiente virtual. Aliás, muitos jogos de RPG online maciçamente multijogador não só permitem como também estimulam, até certo ponto, a formação destes pontos de interesse. Nestes mundos dos MMORPGs, existem cidades inteiras, onde os jogadores se encontram para conversar, trocar mercadorias virtuais do jogo e/ou duelar, assim como existem também zonas desérticas, sem muitos atrativos para os jogadores, e onde o número de avatares presentes é relativamente pequeno, se comparado com os outros lugares no jogo.

[TODO:screenshot(s)]

Por esta razão, não é suficiente simplesmente dividir os jogadores entre os servidores, mesmo que proporcionalmente aos recursos de cada um destes. Primeiro, em alguns lugares o consumo de largura de banda do servidor é quadrático ao número de jogadores, enquanto é linear em outros. Essa razão por si só já é suficiente para buscar outro critério para o balanceamento de carga. Além disso, surge outra questão importante: a existência de pontos de interesse. Esta última característica motiva à criação de um esquema de balanceamento de carga para jogos que impeça que a presença de hotspots degrade a qualidade do jogo além do tolerável.

5.1 Trabalhos relacionados

Como foi dito, quando existe um número considerável de avatares em um mesmo ponto de interesse, é gerado um tráfego proporcional ao quadrado do número de avatares ali presentes. Também foi mostrado que os servidores recebem as ações enviadas pelos jogadores, calculam seu resultado e o enviam para todos os jogadores interessados, que são, geralmente, aqueles cujos avatares estiverem próximos do avatar do primeiro jogador. Se esses jogadores forem divididos entre diferentes servidores, cada um destes precisará não apenas enviar o estado do mundo resultante das ações para os jogadores controlados por ele, como também deverá enviá-lo para o servidor ao qual os outros jogadores estão conectados. Este, por sua vez, encaminhará este resultado para seus jogadores.

[por figura do envio de estados através de servidores diferentes. duas figuras: no mesmo servidor, e através de diferentes servidores.]

Percebe-se, então, que cada estado deverá ser enviado duas vezes, para cada par de jogadores que se comunicam através de servidores diferentes. Esse overhead não apenas causa o desperdício de recursos dos servidores, como também aumenta o atraso para atualização de estado das réplicas do jogo nas máquinas dos jogadores. Isto faz com que o tempo entre o envio de uma ação por um jogador conectado a um servidor e o recebimento do estado resultante por outro jogador, conectado a outro servidor, seja maior, prejudicando a interação entre eles.

Assim sendo, jogadores que estão interagindo entre si devem, idealmente, estar conec-

tados ao mesmo servidor. Contudo, é possível que todos os jogadores estejam ligados entre si através de relações de interação. Por exemplo, dois avatares, de dois jogadores diferentes, podem estar distantes, porém ambos interagindo com um terceiro avatar, entre os dois. Esse tipo de situação faz com que todos os jogadores estejam relacionados de alguma forma. Portanto, é necessário um critério para decidir quando dois jogadores estarão conectados ao mesmo servidor, e quando não estarão.

5.1.1 Microcélulas e macrocélulas

O balanceamento de carga entre servidores de jogos online maciçamente multijogador é fortemente dependente da distribuição dos avatares dos jogadores através do ambiente virtual. Além disso, a depender da concentração de avatares, pode-se alternar entre uma função de crescimento de tráfego linear e uma função quadrática. Sendo assim, é necessário lidar com a localidade dos avatares, de maneira a otimizar o uso de largura de banda do sistema servidor, minimizando, tanto quanto possível, o overhead causado pela comunicação entre jogadores ligados a servidores diferentes.

Uma maneira de fazer uso da localidade dos jogadores é agrupá-los de acordo com a posição ocupada por seus avatares no ambiente virtual. A questão seria a de como formar estes grupos. Uma maneira de fazer isto seria dividindo o mundo do jogo em várias células conectadas entre si. Cada célula consiste em uma parte do mundo, com conteúdo e características próprios, sendo delegada a um nodo servidor. A forma mais simples de fazer isto é com uma grade de células de mesmo tamanho e formato, porém o formato e a disposição destas células irá influenciar no tráfego gerado pelas mesmas entre os servidores. Por exemplo, um ambiente bidimensional poderia ser dividido em uma grade de células quadradas. Neste caso, cada uma destas células teria oito vizinhos, em média – células nas bordas do mapa poderiam ter cinco ou três vizinhos apenas. Quanto mais servidores vizinhos, maior o tráfego entre servidores, e maior o overhead causado por esta comunicação. A distribuição ideal, então, seria utilizando células hexagonais, cada uma com seis vizinhos. Estudos comprovam que esta é a divisão em células iguais que permite o menor número de vizinhos por célula. Outra possibilidade seria utilizando fileiras alternadas de células quadradas, onde cada fileira seria deslocada o equivalente à metade do comprimento de uma célula.

[TODO:figura com diferentes tipos de divisão]

Uma questão importante deste design é que o conceito de célula é transparente para os jogadores. Estes visualizam um mundo vasto, único e não fragmentado, mesmo que estejam cruzando repetidamente as fronteiras entre diferentes células. Assim, é possível para eles moverem-se livremente através do ambiente virutal, independente de como é feita a distribuição. Obviamente, issso exige que as células se comuniquem, de maneira a atualizarem-se mutuamente e notificarem-se a respeito de eventos que ocorrem próximo à fronteira entre cada par de células, assim como a respeito da migração de jogadores entre uma e outra.

Embora essa abordagem com células distribua a carga entre diversos servidores, não há garantias de que essa distribuição será uniforme, devido à grande mobilidade dos jogadores e à existência de pontos de interesse. Uma das idéias propostas na literatura (DE VLEESCHAUWER et al., 2005) também segue o princípio de dividir o ambiente virtual em células de tamanho e posição fixos, porém essas células são relativamente pequenas – ou **microcélulas** – e podem ser agrupadas, formando um espaço contínuo chamado de **macrocélula**. Cada macrocélula é então designada a um diferente servidor, que administra então não apenas uma grande célula, mas um conjunto variável de peque-

nas células. Estas microcélulas podem então ser transferidas dinamicamente entre diferentes macrocélulas, de maneira a manter a carga em cada um dos diferentes servidores abaixo do limite por ele suportado.

Obviamente, as microcélulas designadas ao mesmo nodo servidor não gerarão tráfego adicional para sincronizarem-se entre si, porém torna-se imprevisível o overhead de sincronização entre diferentes macrocélulas, pois o número de outras macrocélulas vizinhas é imprevisível, tal como o seu formato. Porém, demonstrou-se (DE VLEESCHAUWER et al., 2005) que esse overhead é compensado pela melhor distribuição da carga do jogo entre os servidores.

[TODO: figura com microcélulas e macrocélulas - fazer vetorial ou copiar? depende do tempo que tiver depois de terminar o texto]

5.1.2 <dewan ahmed>

save for later... find some holes in it...

5.1.3 Balanceamento local

No trabalho de (LEE; LEE, 2003), é proposto um esquema dinâmico de distribuição de carga para os servidores de um sistema multi-servidor de ambiente virtual, levando em conta que os usuários podem estar distribuídos através deste mundo de maneira não uniforme. De acordo com o esquema proposto, um servidor sobrecarregado inicia a distribuição de carga selecionando um um conjunto de outros servidores para fazerem parte da distribuição, adptando-se dinamicamente ao nível de carga destes servidores selecionados. Após completar a seleção de servidores, o servidor que iniciou o processo reparte as regiões dedicadas ao grupo resultante de servidores utilizando um algoritmo de particionamento de grafo, de forma que os servidores envolvidos tenham carga final de trabalho semelhante. Após decidir-se quem fica com que trabalho, os servidores envolvidos migram suas cargas entre si de maneira par-a-par.

A principal questão que os autores consideraram para resolver este problema foi a da escolha de utilização de informações locais (servidor e seus vizinhos) ou globais (todos os servidores envolvem-se no balanceamento). A primeira apresenta pouco overhead, mas pode não resolver o problema de maneira eficiente em poucos passos, já que servidores sobrecarregados tendem a estar adjacentes. Já a abordagem global é capaz de dividir a carga de trabalho da forma mais equilibrada possível, mas sua complexidade cresce exponencialmente com o número de servidores envolvidos. A solução apontada no trabalho então é de o balanceamento de carga envolver apenas um subconjunto de servidores, sendo que sua cardinalidade varia de acordo com a necessidade (se os vizinhos do servidor que disparou o balanceamento de carga estiverem também sobrecarregados, são selecionados mais servidores). Dessa forma, tem-se um pouco mais de informação do que a abordagem local, mas sem o problema da complexidade inerente à abordagem global.

[TODO:parei aqui]

Para atacar o problema, os autores consideram um sistema multi-servidor de ambiente virtual distribuído, consistindo de Ns servidores: S_1, S_2, ..., S_{Ns}. O ambiente virtual que o sistema mantém é subdividido em Nc células retangulares: C_1, C_2, ..., C_{Nc}, sendo que Ns $<<$ Nc. As células são agrupadas em Nr regiões e cada região é gerenciada por um servidor (i.e., Ns = Nr). Cada servidor mantém atualizadas informações de estado dos usuários e lida com as interações entre usuário na região a ele dedicada. Cada usuário envia e recebe atualizações de estado através do servidor que gerencia a região na qual ele está jogando. A Figura 5.1 ilustra o modelo de um sistema multi-servidor consistindo

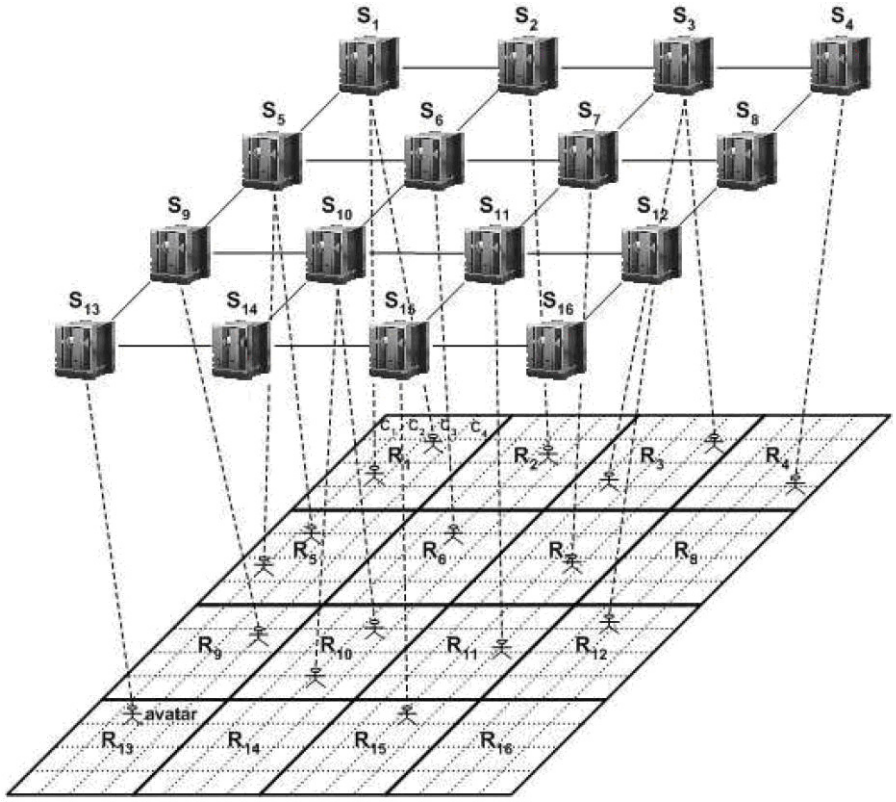


Figura 5.1: Um modelo multi-servidor para ambiente virtual distribuído

de 16 servidores. O ambiente virtual é dividido em 256 células, que são agrupadas em 16 regiões.

Duas células são ditas adjacentes (ou vizinhas) se elas compartilham uma fronteira em comum. Analogamente, duas regiões (e seus respectivos servidores designados) são ditos adjacentes se uma região contém uma célula que pertença à outra região. Um usuário representado por um avatar circula livremente de região a região e interage com outros usuários no ambiente virtual.

Define-se a carga de trabalho de uma célula, denotada por w(C), como o número de usuários presentes naquela célula. Assumindo que todos usuários atualizam seus estados na mesma frequência, a carga de processamento (computação e comunicação) que uma célula impõe a um servidor é proporcional ao número de usuários naquela célula. Similarmente, a carga de trabalho de uma região e seu servidor designado, denotada por w(R), é definida como a soma das cargas de trabalho das células que compõem aquela região.

Cada servidor periodicamente avalia sua carga de trabalho e troca informações de carga com os servidores vizinhos. Assume-se que estes servidores estejam conectados através de uma rede de alta velocidade. Dessa forma, o sobrecusto de trocar informações de sobrecarga entre vizinhos é limitada e considerada negligenciável, se comparada com outros custos da distribuição de carga.

A capacidade de processamento de um servidor é limitada; isto é, uma grande quantidade de carga, além da capacidade de um servidor aumenta o tempo de processamento das mensagens de atualizações de estado dos usuários. Define-se como capacidade de um servidor, representada por CP, o máximo número de usuários que o servidor pode suportar sem prejudicar a performance da interação entre os usuários. Considera-se que todos os servidores têm a mesma capacidade.

5.1.4 O esquema de distribuição de carga dinâmico proposto

Um servidor inicia a distribuição de carga quando sua carga excede sua capacidade. O servidor iniciador primeiro seleciona um conjunto de servidores para se envolver com a distribuição. Após completar a seleção dos servidores, o servidor que iniciou reparte as regiões que eram dedicadas a ele entre os servidores envolvidos, de forma que eles terão aproximadamente a mesma carga. Então, os servidores envolvidos migram suas celulas e usuarios entre si de maneira par-a-par, de acordo com o resultado do reparticionamento. Nas sessoes seguintes, será descrito em detalhe como isso é realizado.

5.1.4.1 Seleção adaptativa de servidor

Diferente dos esquemas global e local, um servidor iniciador no esquema proposto seleciona um conjunto de servidores para se envolver com a distribuição de carga dinamicamente se adptando ao status dos outros servidores. O servidor iniciaddor primeiro escolhe o menos carregado dentre seus vizinhos e pede que ele participe da distribuição. O vizinho escolhido rejeita o pedido se ele já participa ou executa outra distribuição de carga; caso contrário, ele participa da distribuição de carga respondendo ao servidor iniciador com a informação de carga de seus servidores vizinhos. Se o servidor vizinho que está partcipando não for capaz de absorver a carga de trabalho excedente do servidor iniciador, a seleção é executada novamente entre os servidores vizinhos de não apenas o servidor sobrecarregado, como também os vizinhos do vizinho escolhido na primeira fase. A seleção continua até que a carga de trabalho excedente do primeiro servidor possa ser absorvida - isto é, a carga de trabalho de todos os servidores selecionados torna-se menor que o limite. Pode-se definir este limite como 90% do CP, de forma a evitar o imediato reinício da distribuição de carga.

O procedimento para determinar o conjunto de servidores envolvidos é descrito da seguinte forma:

1. Um servidor iniciador é inserido a SELECIONADOS, que é o conjunto de servidores envolvidos na distribuição de carga. Os servidores vizinhos do iniciador são adicionados como CANDIDATOS, que é o conjunto formado pelos servidores candidadtos a seleção.

2. De CANDIDATOS, é selecionado o servidor com menor carga de trabalho; então, o servidor iniciador faz um pedido a ele para participar na distribuição de carga

   (a) Se o servidor escolhido não está envolvido em outra distribuição de carga, ele responde ao servidor iniciador com a carga de trabalho de seus vizinhos. Quando o servidor iniciador recebe esta resposta, o servidor escolhido é inserido em SELECIONADOS e seus vizinhos são inseridos em CANDIDATOS, se eles já não estiverem dentro de SELECIONADOS ou de CANDIDATOS.

   (b) Se o servidor escolhido já está participando de outra distribuição de carga, ele rejeita o pedido e é removido do conjunto CANDIDATOS.

3. O passo 2 é repetido até que a carga de trabalho média dos servidores selecionados se torne menor que um limite: 0,9 x CP.

Para exemplificar o funcionamento do algoritmo, pode-se observar a Figura 5.2. Todos os servidores têm a mesma capacidade de 100 - isto é, CP = 100. Primeiro, o servidor

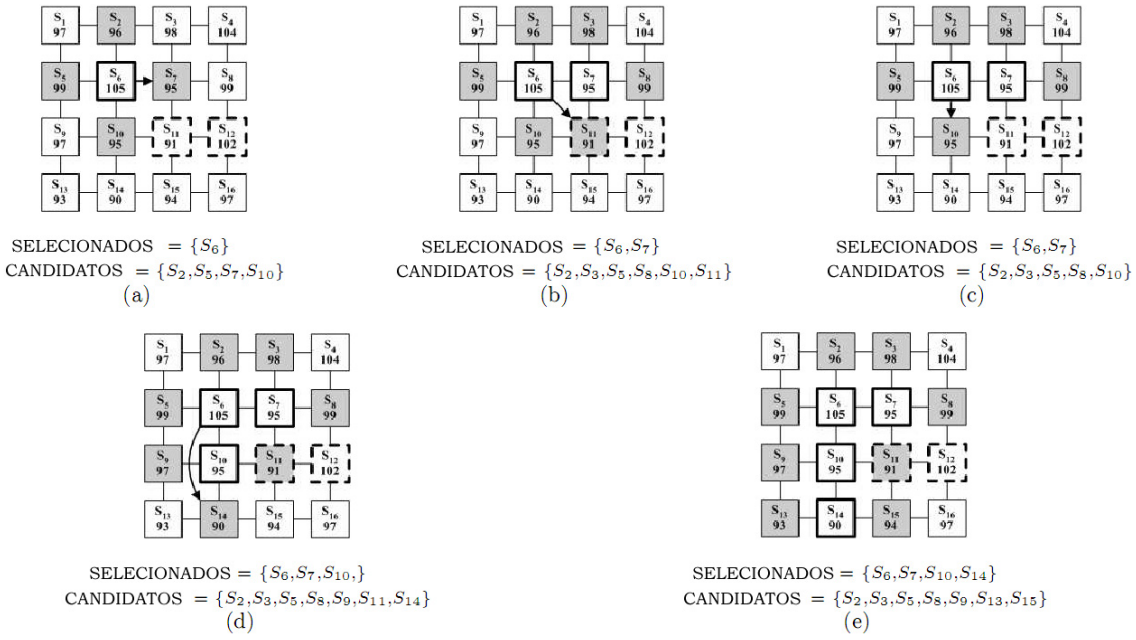


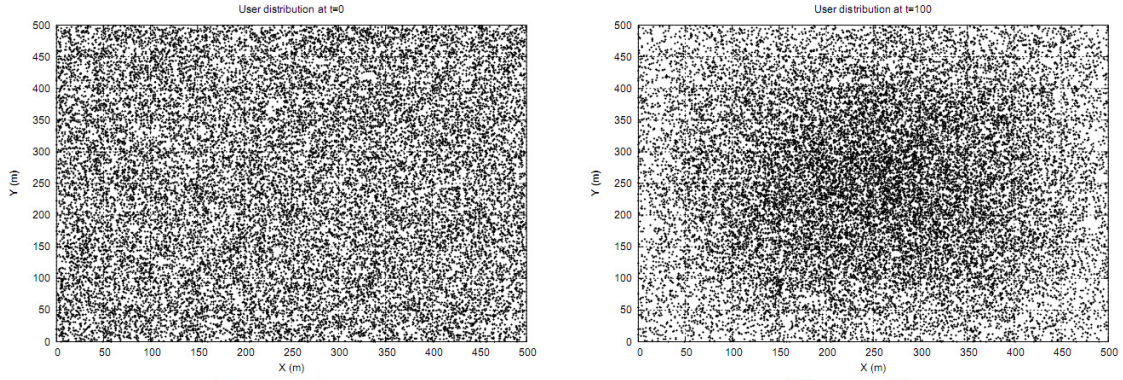
Figura 5.2: Um exemplo do algoritmo proposto para seleção de servidor

iniciador, S6, é inserido em SELECIONADOS e seus vizinhos (S2, S5, S7 e S10) são adicionados a CANDIDATOS (Figura 5.2(a)). Então, S7, que têm a menor carga de trabalho dentre os servidore em CANDIDATOS, é selecionado e convidado a participar da distribuição de carga. Quando S7 responde a S6 com a informação de carga de seus vizinhos (S3, S6, S8 e S11), S7 é inserido em SELECIONADOS e seus vizinhos, exceto S6, são adicionados a CANDIDATOS (Figura 5.2(b)). Agora, S11, que tem a menor carga de trabalho dentre os servidores em CANDIDATOS, é selecionado e convidado a participar da distribuição de carga. Porém, S11 rejeita o convite, pois já está envolvido em outra distribuição, iniciada por S12. Assim, S11 é removido de CANDIDATOS e S10 é selecionado porque tem agora a menor carga de trabalho dentre os servidores em CANDIDATOS (Figura 5.2(c)). Até que a carga de trabalho média dos servidores em SELECIONADOS se tornar menor que 0,9xCP = 90, o procedimento acima continua (Figura 5.2(d) e Figura 5.2(e)).

5.1.4.2 Particionamento de região

Uma vez que o servidor iniciador seleciona um conjunto de servidores para se envolverem na distribuição de carga, ele reparte as regiões a ele dedicadas com os servidores envolvidos. Depois de reparticionar as regiões, todos os servidores envolvidos terão aproximadamente a mesma carga de trabalho. É utilizada a técnica de particionamento de grafos, que é extensivamente utilizada em computação paralela e de alta performance.

Um grafo G = (V, E, P) é construído utilizando informações detalhadas, isto é, a carga de trabalho de cada célula. Um vértice Vi pertencente a V representa a célula Ci. O peso do vértice Vi é ajustado como a carga de trabalho da célula Ci - isto é, w(Ci). Uma aresta Eij representa que duas células Ci e Cj são adjacentes. Os vértices são agrupados em partições $P = \{P1, P2, ...P|selected|\}$ que representam as regiões selecionadas. O grafo construído é reparticionado utilizando um algoritmo de particionamento de grafo, de forma que cada partição tem um subconjunto de vértices de peso total aproximadamente igual - ou seja, cada servidor terá aproximadamente a mesma carga de trabalho.



(a) $t = 0$. (b) $t = 100$.

Figura 5.3: Distribuição dos usuários com o modelo de movimentação proposto

5.1.4.3 Migração de usuário e célula

Com o término do reparticionamento da região, o servidor iniciador dissemina o resultado do reparticionamento para os servidores envolvidos. O resultado inclue a informação de que células devem migrar para quais servidores. Os servidores que recebem o resultado executam a migração de células e usuários uns com os outros de maneira par-a-par.

Primeiro um servidor replica a informação atualizada de usuários nas células migrantes para os servidores que as receberão. A informação que deve ser replicada varia de acordo com as características da apliação. Por exemplo, pode incluir apenas a localização dos usuários; ou pode conter informações mais detalhadas, como qual é o modelo 3D do avatar do usuário. Depois de completar a replicação, o servidor notifica os usuários nas células migrantes a respeito do novo servidor. Os usuários, então migram para o novo servidor; eles simplesmente deixam o servidor antigo e juntam-se ao novo.

5.2 Padrão de movimentação do usuário

Este artigo também propõe um modelo de movimentação de usuário, que baseia-se no *Random Waypoint Mobility Model*, que é extensivamente utilizado para avaliação de performance em redes sem fio ad-hoc. Usuários estão aleatoriamente distribuídos no ambiente virtual. Eles têm suas próprias localizações de destino traçadas, que são escolhidas aleatoriamente no mundo virtual. A cada passo t no tempo, cada usuário se move em direção ao seu destino através de uma linha reta com velocidade escolhida aleatoriamente entre 0 e max_speed. A velocidade máxima dos usuários, max_speed, é escolhida de forma que um usuário possa ir da esquerda para a direita do ambiente virtual em 100 passos de tempo. Por exemplo, se o tamanho do ambiente virtual é de $500x500(m^2)$, a velocidade máxima dos usuários é de 500/100 = 5 (m/passo de tempo). Quando um usuário chega à sua posição de destino, ele começa a se mover novamente de acordo com a mesma regra. Figura 5.3(a) mostra a distribuição inicial de usuários no ambiente virtual no instante t=0 e a Figura 5.3(b) mostra a distribuição de usuários no instante t=100. A distribuição de usuários mostrada nas figuras está de acordo com os resultados da distribuição espacial do *Random Waypoint Mobility Model*: os usuários tendem a se aglomerar na área central, ao invés da área das bordas.

5.3 Avaliação do trabalho

Tal como os trabalhos vistos anteriormente, (LEE; LEE, 2003) contém uma proposta de suporte a ambientes virtuais distribuídos (onde se encaixam jogos maciçamente multijogador), baseada na distribuição do servidor do ambiente, porém, considerando uma rede local e que não há atraso significativo na comunicação entre os nodos servidores.

De qualquer forma, este trabalho traz um algoritmo interessante para reparticionamento do ambiente virtual do jogo, de maneira dinâmica e adaptativa, buscando repartir a carga de trabalho com servidores menos sobrecarregados, seguindo uma heurística definida pelos autores.

Além disso, é proposto um modelo simples de movimentação dos usuários, baseado no *Random Waypoint Mobility Model*, que pode vir a ser útil numa futura simulação em algum trabalho futuro.

5.4 Esquema proposto

5.5 Implementação

5.6 Simulações e resultados

6 CONSIDERAÇÕES FINAIS E TRABALHOS FUTUROS

REFERÊNCIAS

ASSIOTIS, M.; TZANOV, V. A distributed architecture for MMORPG. **Proceedings of 5th ACM SIGCOMM workshop on Network and system support for games**, [S.l.], 2006.

BLIZZARD. **World of Warcraft**. http://www.worldofwarcraft.com/.

CECIN, F.; REAL, R.; OLIVEIRA JANNONE, R. de; RESIN GEYER, C.; MARTINS, M.; VICTORIA BARBOSA, J. **FreeMMG: a scalable and cheat-resistant distribution model for internet games**. 2004. 83–90p.

DE VLEESCHAUWER, B.; VAN DEN BOSSCHE, B.; VERDICKT, T.; DE TURCK, F.; DHOEDT, B.; DEMEESTER, P. Dynamic microcell assignment for massively multiplayer online gaming. **Proceedings of 4th ACM SIGCOMM workshop on Network and system support for games**, [S.l.], p.1–7, 2005.

IIMURA, T.; HAZEYAMA, H.; KADOBAYASHI, Y. 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**, [S.l.], p.116–120, 2004.

LEE, K.; LEE, D. A scalable dynamic load distribution scheme for multi-server distributed virtual environment systems with highly-skewed user distribution. **Proceedings of the ACM symposium on Virtual reality software and technology**, [S.l.], p.160–168, 2003.

MCCANNE, S.; FLOYD, S. et al. Network simulator ns-2. **Available for download at http://www.isi.edu/nsnam/ns**, [S.l.], 2006.

MINSON, R.; THEODOROPOULOS, G. An adaptive interest management scheme for distributed virtual environments. **Proc. of PADS '05**, [S.l.], p.273–281, 2005.

MORGAN, G.; LU, F.; STOREY, K. Interest management middleware for networked games. **Proc. of SIGGRAPH 2005**, [S.l.], v.3, n.06, p.57–64, 2005.

NCSOFT. **Lineage II**. http://www.lineage2.com/.

SCHIELE, G.; SUSELBECK, R.; WACKER, A.; HAHNER, J.; BECKER, C.; WEIS, T. Requirements of Peer-to-Peer-based Massively Multiplayer Online Gaming. **Proceedings of the Seventh IEEE International Symposium on Cluster Computing and the Grid**, [S.l.], p.773–782, 2007.