1
Introdução

Jogos Distribuídos para jogadores em massa (*Massively Multiplayer Games*) são jogos onde milhares de jogadores se conectam pela Internet à mundos virtuais persistentes. Nesse estilo de jogo, o mundo continua a evoluir mesmo quando o jogador não está conectado. Os personagens utilizados pelos jogadores geralmente não perdem suas caraterísticas quando o jogador se desconecta, permitindo a evolução contínua do personagem.

Alguns dos jogos mais bem sucedidos desse tipo são: Everquest (fig. 1.1), Lineage (fig. 1.2) e World of Warcraft (fig. 1.3). O Brasil também possui alguns jogos desenvolvidos como FutSim e Erinia (fig. 1.4).

Algumas siglas geralmente utilizadas para nomear esse tipo de jogo são MMORPG (*Massive Multiplayer Online Role Playing Game*) e MMOG(*Massive Multiplayer Online Game*).

Para conseguir atender esse grande número de jogadores, os servidores precisam usar diversas técnicas de otimização da distribuição e tecnologias eficientes no processamento das mensagens.

Geralmente, o processamento do jogo é custoso demais para ser

Figura 1.1: Imagem do jogo Everquest

Figura 1.2: Imagem do jogo Lineage

Figura 1.3: Imagem do jogo World of Warcraft

Figura 1.4: Imagem do jogo Erinia

efetuado em uma única CPU. Sendo assim, normalmente, ou é utilizado um servidor com múltiplas CPUs para o processamento ou o processamento é quebrado entre diversos computadores conectados numa rede de alta velocidade.

Quando uma rede de computadores é utilizada para o processamento, cuidados devem ser tomados para que nenhum computador fique sobrecarregado. Uma maneira de equilibrar o processamento é fazer cada computador da rede tratar de um número equivalente ou próximo de clientes. Para conseguir isso, geralmente cada servidor é colocado como responsável por uma parte do mundo virtual. Todos os clientes que se encontram em uma região são tratados pelo servidor responsável. Quando um cliente se move para uma outra região, ele é redirecionado para o servidor responsável pela região de destino.

Nos primeiros jogos distribuídos para jogadores em massa, as regiões tratadas pelos servidores eram estáticas. Isso trazia problemas quando grandes grupos de jogadores resolviam se juntar em uma única região, sobrecarregando o servidor responsável.

Atualmente já existem tecnologias em distribuição que implementam regiões dinâmicas. À medida que os jogadores se movimentam no mundo, as fronteiras entre as regiões se movem de maneira a equilibrar o processamento

entre os servidores.

É importante que sejam enviadas para um cliente, apenas mensagens que o influenciam de alguma forma ou são perceptíveis pelo jogador. Uma maneira de se fazer isso, é enviar para um jogador apenas informações sobre acontecimentos próximos a ele. Para conseguir adquirir rapidamente essas informações, é necessário o uso de alguma estrutura especial, como árvores que dividem hierarquicamente o espaço (*Quadtrees*) ou um *grid*. Informações sobre outros tipos de estruturas podem ser encontradas em fontes relacionadas a computação gráfica [18].

Sistemas operacionais como o Windows e o Unix possuem tecnologias eficientes para tratar de grandes quantidades de clientes. No Windows, existe o modelo de processamento de dados chamado IOCP(Input/Ouput Completion Ports) que é sugerido como a forma mais eficiente para processar muitos sockets em Windows [31]. Em Unix, existem as funções poll() e kqueue() que também permitem boa eficiência ([30], [11]).

Geralmente, TCP e UDP não são os protocolos mais adequados para todos os tipos de informação que devem ser trocados em um sistema para jogadores em massa. Muitos jogos implementam pelo menos garantia de chegada por cima do protocolo UDP, pois algumas informações não precisam chegar ordenadas, isto é; importa apenas que elas cheguem.

A união dessas e outras técnicas, tecnologias e protocolos é o que faz os servidores serem capazes de possuir milhares de jogadores conectados, pois essa união permite otimizar o processamento de jogos multijogador em massa.

Esses sistemas de jogadores em massa têm também que lidar com um desafio inerente a ambientes virtuais distribuídos: como gerenciar o estado de jogo compartilhado. Para manter uma boa interatividade, é necessário que os clientes possuam alguma divergência no estado de jogo. Porém essa divergência está sempre sob o controle do servidor.

É possível manter a sincronia completa no estado de jogo dos clientes, usando um repositório central que armazena o estado do jogo. Neste caso, os jogadores acessam este repositório, em turnos, e consultam essa mesma base de dados a cada frame. Essa técnica tem um alto custo em envio de mensagens e reduz a interatividade.

Outras maneiras de gerenciar o estado de jogo, são (1) notificação frequente de atualização de estado(*frequent state update notification*); (2) *Dead-Reckoning*. Na primeira maneira, as atualizações são constantemente transmitidas sem trocas de confirmação. Um frequência alta dessas atualizações tende a deixar os estados de jogo dos clientes próximos uns dos

outros. Usando *Dead-Reckoning*, um cliente ou servidor só receberá uma retransmissão de estado quando não for capaz de prever o estado futuro de algum jogador ou objeto do ambiente. Estas duas técnicas de gerenciamento de estado compartilhado permitem o uso de bases de dados replicadas e descentralizadas.

A qualidade de um ambiente virtual distribuído também é determinada por sua escalabilidade, que é a sua capacidade de aumentar o número de componentes (objetos e jogadores) sem degradar o sistema. As técnicas de gerenciamento de estado anteriormente mencionadas devem ser usadas em conjunto com técnicas de gerenciamento de recursos para escalabilidade e desempenho. Singhal e Zyda [24] apresentam uma descrição destas técnicas de gerenciamento de escalabilidade e desempenho; tais como: compressão de pacotes; agregação de pacotes; filtragem de áreas de interesse; protocolos multicast; exploração dos limites de percepção humana (p.ex. percepção de nível de detalhe e percepção temporal); otimizações de arquitetura de sistema (p.ex. agrupamentos de servidores).

Um grande desafio na implementação de protocolos de comunicação em MMORPGs é como implementar um protocolo seguro contra ataques de hackers. Quando um "hacker"consegue quebrar uma regra de jogo, ele pode desbalancear o mundo e tirar a diversão para muitos jogadores honestos. Como princípio básico, não se deve confiar em nenhuma informação fornecida pelo cliente. Em geral a comunicação vai consistir em o cliente "pedindo"para que algo aconteça e o servidor decidindo se esse algo vai acontecer ou não.

Fraudes e ataques são bem comuns em jogos *online* [25] e há, na literatura, várias sugestões de como implementar protocolos específicos anti-fraude [2]. Os problemas de segurança junto com as questões de gerenciamento de estado e de recursos tornam os MMORPGs um das mais complexas e desafiadores aplicações em tempo real. Propostas práticas podem ser encontradas no trabalho de Feijó e Binder [4].

Técnicas eficientes para o processamento de muitas conexões e para o uso eficiente de sockets podem ser encontradas em algumas fontes, como por exemplo [26] e [9]. O mesmo pode ser dito sobre arquiteturas de ambientes virtuais distribuídos [24], técnicas de gerenciamento de estado [29] e técnicas de balanceamento e transmissão de informações espaciais e geométricas [10]. No entanto, informação sobre ambientes virtuais distribuídos para jogadores em massa (*Massively Multiplayer*) ainda é muito escassa, apesar de já existirem livros sobre o assunto [1] e alguns congressos que tratam da área [22], [5]. Exemplos práticos, sobre a maneira como todas essas técnicas

são usadas em conjunto para criar ambientes virtuais, são difíceis de serem encontrados.

É com base no contexto descrito que essa dissertação é desenvolvida. O objetivo dessa dissertação é analisar técnicas para a construção de ambientes virtuais distribuídos para jogadores em massa, suprindo uma lacuna na literatura, e propor uma ferramenta para a utilização conjunta de algumas das técnicas apresentadas.

A ferramenta proposta consiste numa biblioteca que pode ser incluída à um projeto que queira implementar um MMORPG.

A dissertação está organizada como se segue:

- O capítulo 2 trata de TCP,UDP e sockets. Este capítulo disserta sobre quando usar TCP e UDP, sobre otimizações para reduzir cópias de buffers quando se está usando sockets e sobre uma maneira de reduzir o custo no envio de mensagens UDP. É falado também sobre como implementar garantia de chegada sobre UDP.
- O capítulo 3 trata de maneiras de se processarem múltiplos clientes em um servidor. Inicialmente são explicadas as maneiras usuais para se fazer isso e suas limitações. A seguir são explicadas maneiras mais eficientes de se realizar esse processamento. Também são explicadas tecnologias específicas de sistemas operacionais, como o IOCP do Windows e o kqueue do Unix.
- O capítulo 4 trata de técnicas específicas para ambientes virtuais distribuídos. *Dead-reckoning* é explicado. A sincronização baseada em tempo de comando, uma sincronização complementar ao *Dead-Reckoning*, é detalhada. São explicadas também várias técnicas de otimização da distribuição.
- O capítulo 5 trata da ferramenta proposta e de seus resultados. Várias técnicas e tecnologias explicadas anteriormente são mostradas, na prática, na implementação de um sistema distribuído para jogadores em massa.
- O capítulo 6 trata das conclusões da dissertação, apresentando as contribuições e os trabalhos futuros.