UNIVERSIDADE DA BEIRA INTERIOR
Faculdade de Engenharia
Departamento de Informática

Escalabilidade de Jogos *Online*
Comparação de uma versão Cliente-Servidor com uma versão *Peer-to-Peer* para o jogo *WebRun*

Rui Manuel Ferreira da Cunha

Dissertação para obtenção do Grau de Mestre em
Engenharia Informática
(2º ciclo de estudos)

Orientador: Prof. Doutora Paula Prata

Covilhã, Outubro de 2014

Dedicatória

Dedico esta dissertação a todas as pessoas importantes da minha vida, por tudo o que aprendi com elas e também pelo apoio incondicional com que sempre me abraçaram.

Dedico esta vitória, em especial, aos meus pais, irmão e família por todo o carinho, força e apoio que fizeram com que terminasse, com sucesso, esta etapa.

À Daniela por toda a dedicação e por nunca me ter deixado sozinho nesta caminhada.

Agradecimentos

À minha orientadora, Professora Doutora Paula Prata, por toda a paciência, compreensão e encorajamento, sem ela esta dissertação não seria uma realidade. Agradeço-lhe também todos os ensinamentos e amizade que sempre demonstrou para comigo.

Ao Tiago Dias e ao José Castanheira pela disponibilização do código do protótipo da versão inicial do jogo *WebRun*.

A todos os meus amigos e colegas que sempre me apoiaram e me acompanharam neste percurso.

A todos os professores que marcaram com os seus ensinamentos e amizade este meu percurso.

Muito obrigado a todos.

Resumo

O negócio dos jogos *online* está em constante expansão. Se pensarmos que neste preciso momento se encontram ligados milhões de jogadores a um qualquer jogo *online*, rapidamente percebemos a dimensão e potencialidade deste mercado. São jogos cada vez mais complexos e elaborados, de maneira fidelizar os jogadores atuais, mantendo-os motivados, e a angariar potenciais jogadores. Essa motivação tem a ver não só com a atratividade do jogo mas também com a qualidade de serviço que o jogo oferece. A maioria dos jogos online segue um modelo cliente-servidor, uma arquitetura que facilita o desenvolvimento e o controlo do jogo, nomeadamente em termos de evitar a utilização indevida por jogadores mal-intencionados. Esta arquitetura tem, no entanto, problemas em termos de escalabilidade. Com o aumento do número de utilizadores, a qualidade do serviço diminui do ponto de vista dos jogadores.

Nesta dissertação propomos como alternativa a utilização de uma arquitetura peer-to-peer para um jogo que permite promover a atividade física e que inicialmente foi desenvolvido como cliente-servidor: o jogo *WebRun*. Apresenta-se um estudo do desempenho das duas versões do jogo quando o número de jogadores aumenta. Os resultados obtidos mostram que a arquitetura peer-to-peer permite aumentar significativamente a escalabilidade do jogo, sendo uma alternativa promissora para áreas em que o problema da fraude no jogo não seja crítico.

Palavras-chave

Cliente-servidor, escalabilidade, exergames, jogos online, peer-to-peer, *WebRun*.

Abstract

The business of online gaming is constantly expanding. If we think that in this precise moment there are millions of players linked to any online game, we quickly realized the power of this market. These games are more and more complex and have elaborated sceneries, in order to keep motivated the current players and captivate new potential players. This motivation has to do with not only the attractiveness of the game but also with the quality of service that the game offers. Most online games follow a Client/Server architecture that facilitates the development and control of the game, for example, in terms of preventing improper use by malicious players. This architecture has, however, problems in terms of scalability. When the number of users increases, decreases the quality of service from the viewpoint of the players.

In this thesis we propose as alternative the use of Peer-to-Peer architecture for a game that promotes physical activity and that was initially developed as Client/Server: the *WebRun* game. It presents a study of the performance of two versions of the game when the number of players increases. The results show that the Peer-to-Peer architecture significantly increases the scalability of the game, being a promising alternative for areas where the fraud problem is not critical.

Keywords

Client-server, scalability, exergames, online games, peer-to-peer, *WebRun*.

Índice

Lista de Figuras

Lista de Acrónimos

ALM	Application-Level Multicast
AoI	Area of Interest
BD	Base de Dados
C/S	Cliente/Servidor
DHT	Tabela de Hash Distribuida
FPS	First Person Shooter
IM	Interest Managment
IP	Internet Protocol
MMOG	Massively Multiplayer Online Games
NPC	Non-Player Character
P2P	Peer-to-Peer
QoS	Qualidade do Serviço
RFC	Request for Comments
RPG	Role-Playing Games
RTS	Real-Time Strategy
WWW	World Wide Web

Capítulo 1

Introdução

O negócio dos jogos *online* não pára de crescer, devido ao facto de comercialmente ser um mercado muito rentável. Atualmente milhões de jogadores estão ligados entre si em jogos *online*. Casos de sucesso como o *World of Warcraft* (lançado na Europa em 2005 pela Blizzard Entertainment) que, segundo a Activision Blizzard em Agosto de 2014 registava 6.8 milhões de jogadores ou o *League of Legendes* (lançado na Europa em 2009 pela Riot Games) que em Janeiro de 2014 registava 7.5 milhões de jogadores, segundo a Riot Games, apresentam números que demonstram o universo que este negócio representa. As perspetivas apontam, para que estes números continuem a aumentar. (Neuman et al. 2007)

Os ambientes virtuais são cada vez mais complexos, autênticos universos com milhões de objetos e personagens não controladas pelo utilizador, os chamados *Non-Player Character* (NPC). Este tipo de complexidade torna difícil uma gestão eficiente por um único servidor, mesmo com arquiteturas *multi-core*. A este já populoso universo temos ainda de incluir os clientes (isto é, os jogadores) que podem facilmente ultrapassar os milhares em simultâneo (Chen et al. 2005).

A qualidade de serviço (QoS) destas aplicações é um fator muito importante para a aceitação por parte do cliente (Robles et al. 2008; Chambers et al. 2012). Para que um jogo tenha sucesso não basta que seja interessante como jogo, é necessário que o jogo seja também fiável, quer em termos de disponibilidade quer em termos de funcionamento sem erros. Mas também é necessário que seja seguro, isto é, sem possibilidades de fraude. Finalmente, é ainda necessário que o jogo seja escalável, isto é, que com o aumento do número de jogadores a qualidade de serviço não seja afetada, nomeadamente que evite os atrasos significativos no tempo de resposta a cada jogador

Há atualmente 3 tipos de abordagens para estudar o problema da escalabilidade: usar *cluster* de servidores; usar servidores distribuídos; e estruturar o jogo segundo uma arquitetura *Peer-to-Peer* (P2P) (Diot e Gauter 1999; Lui e Chan 2002; Ng et al. 2005).

A mais adotada normalmente é o uso de *cluster* de servidores embora seja também a mais custosa monetariamente. Esta abordagem garante melhor escalabilidade quando conjugada com protocolos de encaminhamento, planeamento de tráfego e nós *gateway* (Bauer et al. 2002; Knutsson et al. 2004; Che net al. 2005).

Atualmente, a maioria dos jogos *online* usa por norma uma arquitetura centralizada, tipo Cliente-Servidor (C/S), com um servidor central ou um *cluster* de servidores. Estas

arquiteturas permitem uma fácil gestão de toda atividade dos jogadores. No entanto, esta arquitetura centralizada tem algumas desvantagens: por um lado o servidor constitui um ponto único de falha; e por outro lado não é escalável, tendo sempre de ser definido um número máximo esperado de jogadores (Neumann et al. 2007).

A primeira proposta para o uso da arquitetura P2P em jogos *online* com milhares de utilizadores, os chamados *Massively Multiplayer Online Games* (MMOGS), foi publicada em 2004 por Knutssonn (Knutsson et al. 2004). Nesse modelo, a ideia base é distribuir a carga de processamento por todos os intervenientes do jogo, isto é, pelos vários *peers*, tendo cada *peer* recursos para se manter a si próprio. Isso significa que as operações, que no modelo C/S estão centralizadas no servidor, vão agora ser divididas pelos clientes do jogo. Esta divisão torna o jogo escalável, pois ter mais jogadores implica automaticamente ter mais recursos, e torna o jogo mais tolerante a falhas pois deixa de haver um ponto único de falha. Mas o modelo também levanta questões difíceis de resolver, nomeadamente como gerir, manter consistente e armazenar de forma persistente o estado do jogo (Boulanger et al. 2006; (Gilmore e Engebrecht 2012). Ou ainda, como proteger o jogo contra jogadores mal-intencionados numa arquitetura em que cada jogador também pode ser servidor do jogo e, portanto, ter um controlo sobre parte do estado do jogo (Neumann et al. 2007).

1.1 Motivação e Objetivos

O jogo *WebRun* consiste em usar uma bicicleta de manutenção integrada com um computador ligado à internet. Este jogo pretende promover a atividade física, no conforto da própria casa, tornando-a um divertimento (Castanheira 2012; Araújo et al. 2011). O jogador que propõe uma corrida escolhe num mapa o percurso que pretende simular e convida outros jogadores a participar na corrida. À medida que esta decorre, os vários jogadores podem visualizar a evolução no percurso dos outros jogadores do grupo.

A versão inicial deste jogo segue uma arquitetura C/S, que nos testes efetuados mostrou ser um jogo que se degrada muito rapidamente ao nível da qualidade do jogo com o aumento de jogadores.

O objetivo deste trabalho é propor uma nova arquitetura para o jogo e avaliar a sua escalabilidade.

1.2 Abordagem

A partir do protótipo do jogo *WebRun* disponibilizado pelos seus autores, foram implementadas alterações na aplicação e na sua base de dados (BD). Através dessas

alterações foi possível monitorizar a resposta do servidor aos clientes em jogo. Os dados recebidos foram analisados e foi desenhada uma nova estrutura para o jogo.

A versão P2P implementada neste trabalho é uma versão híbrida: o jogo usa a arquitetura C/S na fase de autenticação mas, mais tarde, durante o jogo, o servidor de uma nova corrida passará a ser o jogador que propôs a corrida.

Finalmente, foi estudado o comportamento de cada uma das versões do jogo quando o número de utilizadores aumenta.

1.3 Contribuições

O trabalho desenvolvido para esta dissertação deu origem às seguintes contribuições para a área dos jogos *online*:

- Apresenta uma síntese do estado da arte em arquiteturas de jogos *online*.
- Propõe uma nova arquitetura para o jogo *WebRun*.
- Apresenta um estudo do comportamento da versão C/S do jogo *WebRun*
- Foi construído um protótipo da versão P2P do jogo *WebRun*
- Apresenta um estudo do comportamento da versão P2P do jogo *WebRun*
- Preparou um artigo científico com os resultados obtidos (em fase de submissão).

1.4 Organização da Dissertação

Esta dissertação encontra-se dividida em seis capítulos. No primeiro capítulo faz-se uma pequena introdução ao tema da dissertação, referindo as motivações que levaram à sua escolha, os objetivos que se pretendem alcançar e a abordagem escolhida ao longo do seu desenvolvimento. O Capítulo 2 apresenta a revisão da literatura existente sobre arquiteturas de jogos *online*, vantagens e problemas de cada uma. No capítulo 3 é descrito o jogo *WebRun* na versão C/S que serve de base a este trabalho. No capítulo 4 descrevem-se as alterações desenvolvidas, apresentando-se a arquitetura P2P da nova versão do jogo. No Capítulo 5 descreve-se a avaliação da escalabilidade das duas versões do jogo, quando o número de jogos/jogadores aumenta. Finalmente, no Capítulo 6 apresentam-se as conclusões, bem como a descrição do que eventualmente poderá ser acrescentado a este trabalho.

Capítulo 2

Revisão da Literatura

Neste capítulo é feita uma introdução aos jogos *online*, em especial os chamados *exergames*, e a duas principais arquiteturas com que normalmente são desenvolvidos: a arquitetura C/S e a Arquitetura P2P.

2.1 Jogos *Online*

Nos dias que correm os jogos *online* têm uma legião cada vez maior de adeptos, o que faz com que este mercado esteja já bastante evoluído, mas mesmo assim possuí ainda muita margem de progressão. (Robles et al. 2008; Chambers et al. 2010)

Existem vários tipos de jogos *online*, estando definidos em diferentes categorias. A principal categoria são os MMOGs, jogos que permitem ligações em simultâneo com milhões de jogadores. Basicamente, um jogo *multiplayer* consiste num tema, personagens e objetos dinâmicos (desde alimentos, armas, etc.). Em relação às personagens, existem as que são controladas pelo jogador e as que não são controladas, designadas por NPC e que são controladas por algoritmos automatizados. Neste tipo de jogos, o jogador torna-se parte integrante de um mundo virtual, onde por norma as personagens pertencem a variadas raças, desde simples humanos até aos míticos elfos (Pinho 2009).

Na sua maioria, os MMOGs dividem-se nas seguintes categorias: *Role-Playing Games* (RPG), *First Person Shooter* (FPS) e *Real-Time Strategy* (RTS). Os RPG são jogos onde o jogador se assume como personagem de uma aventura, como acontece no jogo *Dungeons & Dragons*, *World of Warcraft*, *League of Legends,* entre outros. Por sua vez, nos FPS o jogo é criado segundo a perspetiva visual do jogador. Jogos como o *Quake*, *Counter-Stike*, entre outros. E, por último, os RTS, onde o jogador se posiciona e evolui estrategicamente de forma a dominar uma área do mapa, onde um dos jogos mais conhecido é o *Warcraft III* (Pinho 2009).

O estado do jogador, isto é, a informação importante a ter sempre disponível para os jogadores, é a sua posição e a interação que o jogador pode realizar com o mundo, com os objetos do jogo ou com outros jogadores. Como, por norma, o universo de jogo é muito grande, esse universo é dividido em regiões que são ligadas entre si estaticamente.

Na sua generalidade, a implementação de um jogo baseia-se na partição do mapa, onde cada uma dessas partições, ou regiões, está associada a um servidor (Chen et al. 2005). Os jogadores vagueiam pelos vários servidores à medida que vão percorrendo o mapa de jogo.

Na Figura 2.1 é possível observar, no *World of Warcraft*, um jogo muito popular atualmente, o nível de complexidade a que um jogo pode chegar. Contém uma quantidade muito elevada de itens, que um jogador pode utilizar a qualquer momento.

Figura 2.1 – Screenshot do jogo *World of Warcraft*.[1]

Podemos também classificar os jogos segundo a forma como os clientes se ligam: os jogos baseados no *browser* e os jogos baseados numa aplicação cliente. Os jogos baseados no *browser* requerem apenas uma ligação à internet, e a sua principal vantagem é não haver necessidade de instalar qualquer aplicação para se poder jogar, dando ao cliente liberdade para aceder de qualquer localização desde que possua um computador e acesso à internet. Por sua vez, os jogos baseados no cliente são mais próximos dos jogos de computador tradicionais, em que é necessário a instalação duma aplicação que permitia ao cliente entrar no jogo. A vantagem deste modelo é uma experiência gráfica muito superior (Pinho 2009).

2.2 Exergames e Exergaming

O termo *exergame*s é utilizado para definir jogos de vídeo que promovem a atividade física, nomeadamente os jogos com controlo remoto e sensores de movimento que implicam o

[1] Fonte: http://www.sea-of-memes.com/LetsCode3/LetsCode3.html

movimento dos próprios jogadores. Estes jogos, com a entrada das grandes empresas de videojogos no mercado, tornaram-se muito populares (Brox et al. 2011).

Yoonsin Oh, no seu artigo de 2010, vai um pouco mais longe na definição de *exergame* e considera-o como uma combinação de jogo de vídeo com atividades de esforço, equilíbrio e flexibilidade. Atualmente existe uma preocupação crescente na sociedade para com o problema da obesidade entre crianças e adolescentes, e, neste contexto jogos deste tipo tornam-se aliados importante na promoção da atividade física (Oh e Yang 2010).

Figura 2.2 – Idoso a jogar um *Exergame*.[2]

Outra preocupação atual é a de tornar os idosos mais ativos e combater o isolamento social nas faixas etárias mais avançadas. Há várias dificuldades em fazer com que os idosos participem em sessões de treino fora de suas casas; Outra dificuldade é a motivacional. Na Figura 2.2 podemos observar um idoso a praticar exercícios no conforto da sua casa. Se os jogos tiverem uma componente *online*, permitindo a interação com outros participantes, torna-se também numa ajuda ao combate ao isolamento social (Brox et al. 2011). Um estudo feito por Lieberman, em 2010, concluiu que os *exergames* tornaram a atividade física mais

[2] Fonte: http://joininproject.wordpress.com/2014/02/14/8-user-requirements-of-the-seniors-for-exergames

apelativa, conseguindo com que indivíduos que normalmente não manifestam interesse por atividade física a pratiquem com regularidade (Lieberman et al. 2011).

Num estudo, feito em 2007 por Sinclair et al., sobre o *design* de *exergames*, chegou-se à conclusão que o sucesso de um *exergame* está associado a dois aspetos importantes: o da sua capacidade de proporcionar exercício físico e o da sua atratividade. Um *exergame online*, associado à sua parte social, pode tornar-se mais atrativo e um aliado no combate à solidão dos idosos (Sinclair et al. 2007).

Os *exergame*s mudaram o paradigma da atividade nos jogos de vídeo, de uma atividade passiva para ativa. Empresas de renome como a Nintendo, a Microsoft ou a Sony já exploram este mercado, aliando as suas consolas ao exercício físico. Estes jogos mostram que a tecnologia pode ser uma grande aliada da saúde na luta contra o sedentarismo em indivíduos de todas as idades. A atividade física torna-se mais atrativa quando aliada a uma narrativa, a um sistema de recompensas ou aos gráficos. Com estes ingredientes um individuo perde a noção do tempo e tem prazer enquanto desenvolve uma atividade física (Smallwood et al. 2012).

Tanaka no seu artigo de 2010, analisa os vários tipos de sistemas usados em *exergames*, abordando sobretudo as suas vantagens e desvantagens. Nesse artigo conclui que há vários pontos em comum entre as consolas, como a capacidade de reconhecimento dos movimentos. Revela também que um dos grandes problemas que atualmente impede a maioria dos possíveis utilizadores destas consolas é o seu preço e das licenças dos respetivos jogos (Tanaka 210).

Num estudo desenvolvido por Smallwood, em 2012, foi avaliada a resposta fisiológica e energética durante a jogabilidade usando a *Kinect* para a *Xbox 360*. Neste estudo participaram 18 crianças entre os 11 e os 15 anos, divididos em 10 rapazes e 8 raparigas. Foram postos à prova jogando um jogo sedentário tradicional e dois *exergames*. Cada jogo tinha uma duração de 15 minutos. Nos gráficos apresentados na Figura 2.3 podem ver-se os resultados obtidos durante a experiência, entre os quais a frequência cardíaca, o consumo de oxigénio e o gasto energético (Smallwood et al. 2012).

Os valores analisados para estes parâmetros durante os *exergames* foram considerados bastante positivos em relação ao descanso e ao jogo tradicional. Os *exergames* aumentaram o gasto de energia numa ordem de 150% no jogo *Dance Central* e 263% no jogo *Kinet Sports Boxing*, ambos lançados em 1010 na Europa pela *Microsoft Game Studios*, em relação ao descanso, e de 103% e 194% em relação ao jogo tradicional.

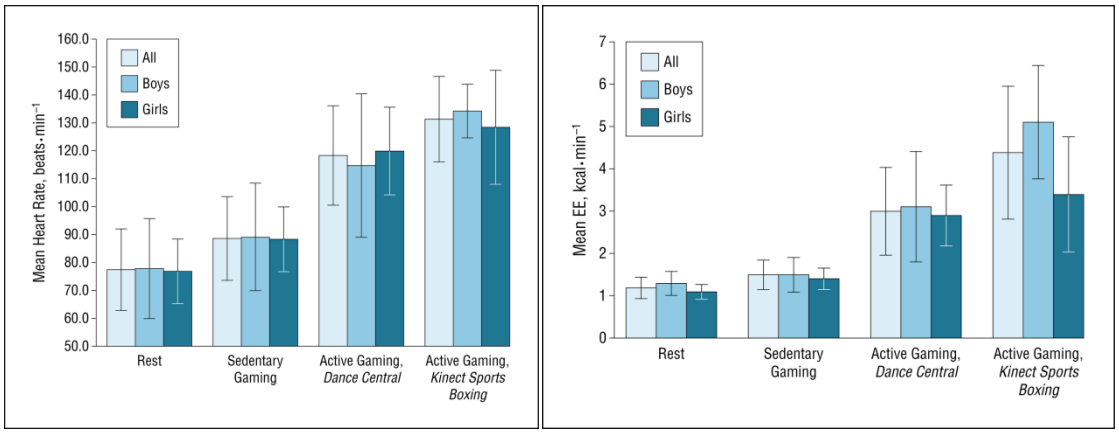

Figura 2.3 – Gráficos dos dados registados da experiência de Smallwood 2012.[3]

2.3 WebRun

O jogo *WebRun* é um *exergame online*, de arquitetura C/S, que consiste na integração de uma bicicleta de manutenção como meio de interação e jogabilidade de um jogo de computador. Ao praticarmos na bicicleta, o jogo permite-nos observar o percurso no *Google Maps*. A principal motivação deste jogo é fazer com que os seus utilizadores estejam motivados enquanto praticam exercício físico em casa. Esta motivação surge da opção de jogar contra outros jogadores e de se desenrolar em percursos reais (Araújo et al. 2011).

O jogo foi desenvolvido a pensar em todas as pessoas que possuam força e destreza para serem capazes de pedalar, podendo fazê-lo da forma mais tradicional ou usando as mãos para mover os pedais. Pessoas com escassez de tempo, utilizadores de ginásios e sem motivação para a prática de exercício, são os potenciais utilizadores deste jogo, que pretende motivá-los com a ideia de transformar a prática de exercício físico numa forma de diversão. Outra motivação no seu desenvolvimento foi o combate ao sedentarismo e à solidão dos mais idosos, de forma a fomentar a prática de exercício físico de forma regular (Castanheira 2012).

2.4 Arquitetura Cliente/Servidor

A arquitetura C/S é um modelo usado no contexto de sistemas distribuídos (Consel e Réveillèu 2003). Esta arquitetura permite a integração de dados e de serviços, isolando os clientes da complexidade inerente a esses mesmos serviços (Oracle 1996).

São três os componentes do modelo C/S: o Cliente, o Servidor e a Rede. O Cliente normalmente é uma aplicação executada num computador, que interage com um periférico de entrada. Não tem responsabilidade no processamento de dados, apenas faz um pedido ao

[3] Fonte: http://www.exergamelab.org/2012/09/kinect-boxing-and-dance-central.html

servidor e apresenta os dados recebidos do servidor. O Servidor é uma máquina que é responsável por um conjunto de processos e funções. A Rede permite o acesso remoto entre os clientes e o servidor (Oracle 1996).

Num servidor são implementados serviços e, através de uma ligação de rede, os clientes fazem pedidos a esses serviços. Uma vez processados esses pedidos, o servidor envia a resposta ao cliente. Para que esta comunicação seja possível é definido um protocolo para coordenar as mensagens enviadas entre cliente e servidor (Consel e Réveillèu 2003). Na Figura 2.4 podemos ver uma esquematização do processo atrás referido.

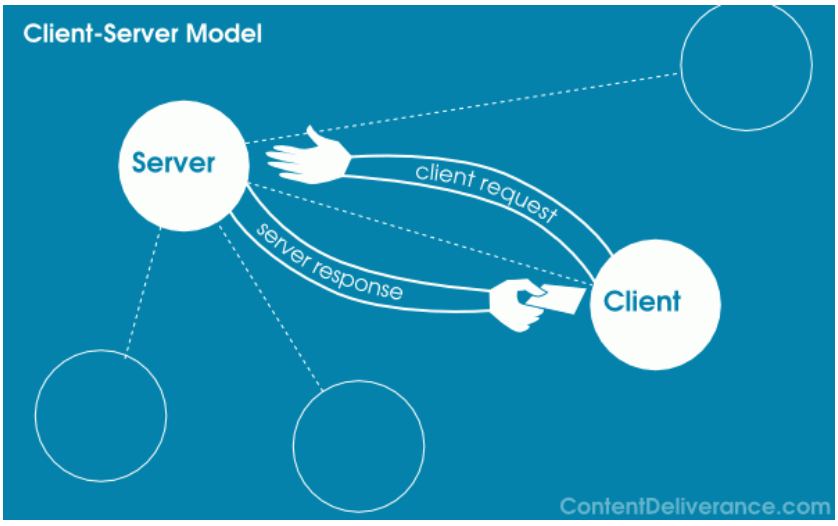

Figura 2.4 - Modelo Cliente/Servidor.[4]

Algumas vantagens deste modelo é a capacidade da aplicação Cliente fazer o pedido ao Servidor e com os dados recebidos poder analisá-los e apresentá-los da forma adequada aos seus recursos. A aplicação Cliente pode também ser desenhada sem dependências fortes do servidor, continuando a funcionar mesmo com algumas alterações no servidor, desde que o protocolo de comunicação não seja alterado (Oracle 1996). A segurança neste modelo é também elevada, pois o servidor é capaz de validar cada pedido enviado pelo cliente (Fan et al. 2010).

Esta arquitetura é a mais usada na implementação dos MMOGs, sobretudo devido à sua fácil implementação e garantia de segurança (Fan et al. 2010). O cliente liga-se a um servidor central, através de uma aplicação própria, e é do servidor a responsabilidade de autenticação e gestão das contas de cliente e da distribuição do estado do jogo pelos mesmos (Knutsson et al. 2004).

[4] Fonte: http://contentdeliverance.com/2011/client-server-architecture

Mas, devido à rápida expansão e graças à popularidade cada vez maior dos MMOGs, o número de jogadores tem crescido drasticamente (Woodcock et al. 2005). Com este aumento de utilizadores, esta arquitetura apresenta algumas desvantagens. Desvantagens que vão desde as de natureza comercial às técnicas, especialmente na área da confiança e dos custos inerentes aos servidores, à largura de banda, ao alojamento, ao arrefecimento, aos sistemas UPS e à manutenção dedicada. Estes fatores têm motivado fortemente a investigação para outra arquitetura como o P2P que a seguir se apresenta (Douglas et al. 2005; Hampel et al. 2006).

2.5 Arquitetura *Peer-to-Peer*

A arquitetura P2P é considerada a terceira geração de Internet, a seguir à própria Internet e à World Wide Web (WWW), pois trouxe de volta o poder para o utilizador comum, permitindo-lhe partilhar os recursos e dados da sua máquina sem necessidade de uma autoridade central (Lee 2003). É comumente caracterizada como um serviço que interage com outros sistemas sem um servidor central, embora esta definição exclua algumas aplicações consideradas P2P, porque usam servidores como diretórios de arquivos centralizados, como o *Naspster* por exemplo (Lee 2003). O *Napster* foi o primeiro serviço P2P de partilha de ficheiros (Waldfogel 2011). Hoje em dia parece emergir um consenso na definição de P2P: não pela tecnologia comum mas pela sua função, são aplicações que exploram os recursos disponíveis nas extremidades da rede, os peers (Lee 2003).

A noção de P2P foi estabelecida pela primeira vez em 1969, no primeiro Request for Comments (RFC). O RFC requer uma ligação P2P, mas apenas em 1979 foi implementada a primeira rede P2P, a USENET. Na USENET, embora os *peers* ainda acedessem aos recursos através de servidores, os servidores comunicavam entre si enviando mensagens sem um servidor central (Li 2007).

Desde finais da década de 90 surgiram várias aplicações P2P, maioritariamente de partilha de ficheiros. Alguns dos mais populares são o *Freenet*, o *Napster*, o *Gnutella*, o *eDonkey2000* e o *BitTorrent* (Li 2007).

Um dos problemas com os sistemas P2P é o mapeamento dos *peers* na internet. Uma vez que não há servidores fixos, os *peers* têm de contar com métodos para localizar outros *peers*. Um dos principais métodos é um sistema de diretórios centralizado (ver Figura 2.5), onde os recursos são indexados num servidor central e os *peers* fazem pedidos ao servidor para localizar os *peers* com os pedidos. Outro método é o *Query Flooding*, que envia para a rede o pedido e os *peers* que possuem esse recurso respondem (Li 2007). Outro método popular passa pelo uso das Tabelas de Hash Distribuídas (DHT), que foram introduzidas em 2001. É, essencialmente, uma tabela de *hash* com um valor-chave de pesquisa com o índice distribuído entre os *peers* da rede.

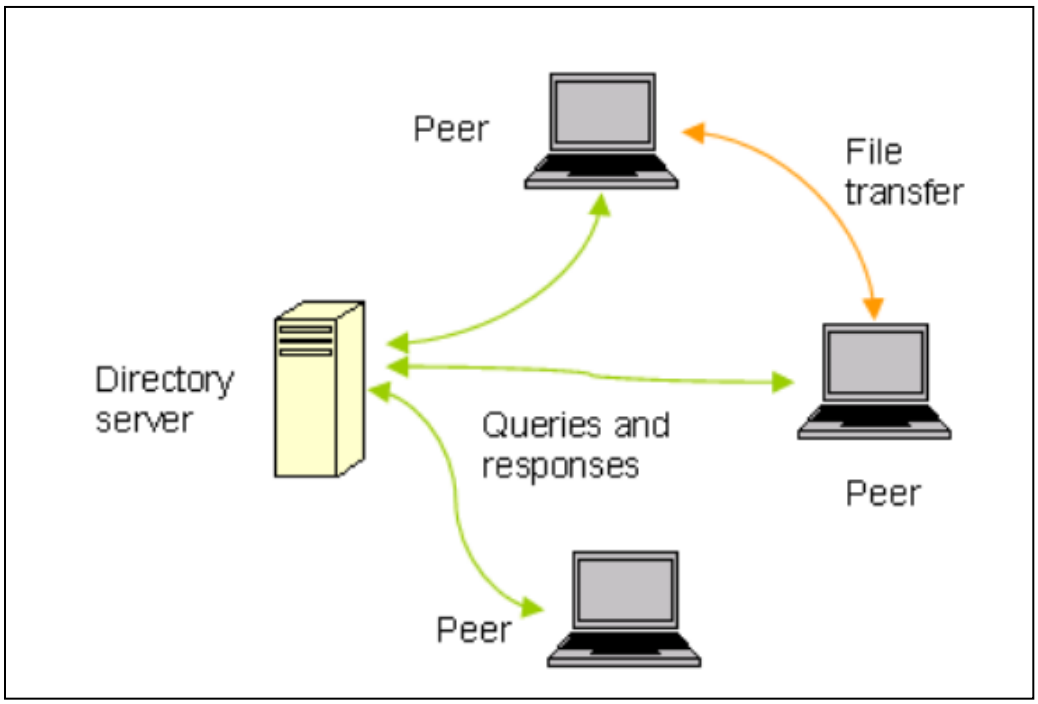

Figura 2.5 – Sistema de Diretórios Centralizado.[5]

É um modelo que pretende resolver muitos dos problemas existentes na arquitetura C/S. A ideia essencial é que cada processo contribua com os seus recursos para um melhor funcionamento da aplicação. Neste tipo de arquitetura todas as funções do servidor no modelo clássico C/S estão distribuídas por todos os *peers*. Os *peers* são um conjunto de computadores ligados entre si através da internet (Gilmore e Engebrecht 2012; Rachanc 2012). Podemos ver um exemplo da arquitetura na Figura 2.6.

[5] (Li 2007)

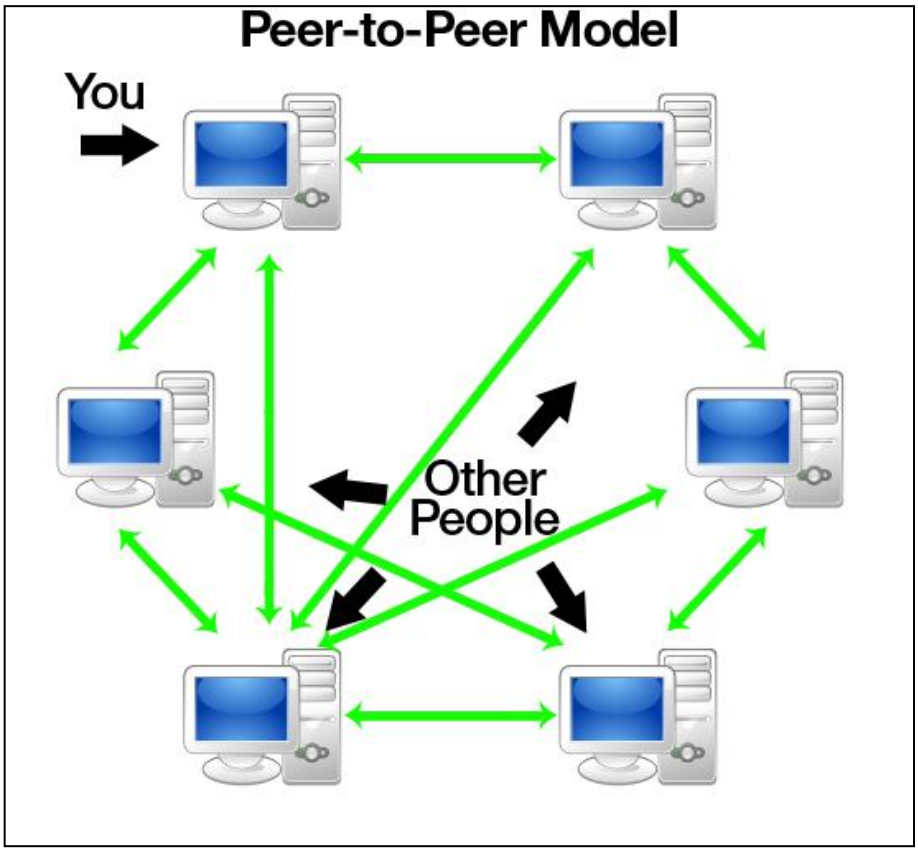

Figura 2.6 - Modelo Peer-to-Peer.[6]

2.6 Desafios da adaptação de Cliente/Servidor para *Peer-to-Peer*

A adaptação dos jogos da arquitetura de C/S para P2P levanta várias questões, entre elas estão o *Interest Management*, *Game Event Dissemination*, *Non-Player Character Host Allocation*, *Game State Persistency*, *Cheating Mitigation* e *Incentive Mechanisms* (Fan et al. 2010).

2.6.1 *Interest Management*

O *Interest Management* (IM) é um tema de pesquisa clássico que foi inicialmente introduzido em meados da década de 1990: Conseguir que um simples jogador não tenha necessidade de ter consciência de todo o universo de jogo e de tudo o que nele se passa. Apenas os acontecimentos que possam afetar um determinado jogador têm importância relevante para que ele tenha conhecimento. É neste determinar a menor informação necessária, que um *peer* necessita para se manter a par da informação relevante ao seu estado atual, que se encontra o desafio do IM. A visão do mundo pelo jogador pode-se limitar à sua *Area of*

[6] Fonte: http://www.codeproject.com/KB/WCF/614028/pto1.png

Interest (AoI), e apenas precisa de ter conhecimento do que se passa dentro dessa área (Macedonia et al. 1994; Fan et al. 2010).

A AoI é a área do jogo a que o jogador, a cada preciso momento, tem pelo menos um dos seus sentidos com capacidades para avaliar alguma alteração. Podemos comparar, por exemplo, ao sentido da visão nos humanos, em que a AoI seria o horizonte percetível a olho nu. Os esquemas de IM podem ser classificados em três tipos distintos: Modelo *Spacial*, Modelo *Region-Based Publish/Subscribe* e um Modelo Híbrido de Comunicação (Fan et al. 2010; Gilmore e Engebrecht 2012).

2.6.1.1 Modelo *Spacial*

Este modelo usa as propriedades do espaço como base para mediar a interação entre os objetos do jogo. Este modelo também é referido como o modelo *aura-nimbus* por causa destas duas abstrações, *aura* e *nimbus*. A abstração *aura* representa a área que limita a presença de um objeto no espaço, esquanto o *nimbus* representa a consciência de presença de um objeto de jogo nas proximidades. Um exemplo simples é comparar este modelo com um humano: como *aura* podemos considerar o corpo desse humano e o *nimbus* os seus sentidos, com os quais se apercebe da presença de algo na sua área (Benford e Fahlen 1993; Morgan et al. 2005; Boulanger et al. 2006).

A vantagem, deste modelo é que permite ao IM um bom refinamento, em que um jogador só recebe as mensagens relevantes. Contudo, uma desvantagem importante deste modelo é a necessidade de que todos os objetos têm de trocar informação sobre a sua posição de forma a saber quando poderão estar dentro de uma zona AoI de um jogador. Pode ser usado um diagrama de *Voronoi*, um método matemático para dividir uma região em várias partes (Aurenhammer e Klein 2000), para ajudar a encontrar a vizinhança de um jogador. É necessário que cada *peer* construa e mantenha o diagrama de *Voronoi* sempre atualizado. Assim, um *peer* apenas precisa de manter ligação com os objetos na sua vizinhança. As desvantagens deste modelo prendem-se com os seguintes fatores: a vulnerabilidade do diagrama *Voronoi* para o problema *circular line-up*, que no pior dos casos seria quando um *peer* tivesse *n-1* vizinhos. Outra desvantagem é que um *peer* continua a ter necessidade de receber e processar mensagem de fora da sua AoI. E, por fim, os custos de computação são muito elevados (Hu e Liao 2004; Buyukkaya e Abdallah 2008).

2.6.1.2 Modelo *Region-Based Publish/Subscribe*

Este modelo usa um refinamento mais grosseiro do IM, particionando o universo de jogo em regiões. Cada jogador receberá informação apenas de quem se encontrar na sua região de jogo. A principal tarefa deste mecanismo é determinar que regiões estão na AoI do jogador. Neste modelo, as comunicações podem ser enviadas usando o *multicast,* devido à

possibilidade de criar grupos de *multicast* com os objetos e jogadores nessa região (Morgan et al. 2005).

As desvantagens deste modelo prendem-se com a dificuldade em determinar o tamanho das regiões. Têm de ter o tamanho suficiente para que um objeto consiga enviar mensagens a todos os objetos antes de mudar de região, de forma a sair do grupo de *multicast*. Outro problema manifesta-se quando os objetos não estão igualmente distribuídos (Morgan et al. 2005).

2.6.1.3 Modelo Hibrido de Comunicação

O Modelo Hibrido de Comunicação é uma mistura do modelo *Spacial* e o modelo *Region-Based Publish/Subscribe*: o universo é particionado em regiões e um *super-peer* é responsável por cada região. Um *super-peer* é um *nó* numa rede P2P que opera de ambas as formas, como servidor e como cliente (Yang e Garcia-Molina 2003; Matsumoto et al. 2005).

Este modelo tira vantagens de ambos os modelos, aproveitando a mais-valia do modelo *Spacial* para permitir ao IM um melhor refinamento em cada região do modelo *Region-Based Publich/Subscribe* e reduz a sobrecarga de comunicação para os jogadores. É relativamente mais simples de implementar e mais eficiente que o modelo *Spacial* puro. As desvantagens são as de impor elevados graus de computação e comunicação aos *super-peer* de cada região. Esse *super-peer* é uma potencial falha na região, o que faz com que sejam necessários mecanismos de tolerância a falhas (Fan et al. 2010).

2.6.2 Game Event Dissemination

Enquanto o IM se foca em encontrar que informação é relevante para cada cliente, o *Game Event Dissemination* preocupa-se em permitir que essa informação relevante seja entregue a cada cliente. Esta escolha está inerente à escolha do mecanismo de IM.

O *Internet Protocol* (IP) *multicast* foi inicialmente proposto como um mecanismo eficiente de comunicação para grupos. Contudo, devido a alguns obstáculos tecnológicos, o IP *multicast* não está amplamente disponível na internet. Como alternativa foi proposto o *Application-Level Multicast* (ALM), com funcionalidades semelhantes, mas apenas como aplicação e não como serviço de rede (Diot et al. 2000; Fieldler et al. 2002; El-Sayed et al. 2003).

2.6.2.1 Unicast vs Multicast

O modelo *Spatial* é um modelo com um bom refinamento do IM, porque alerta um pequeno número de objetos que podem ter de interagir com um jogador. Desta forma, o jogador pode entrar diretamente em contacto com o objeto e a comunicação usada pode ser o *unicast*.

Os *super-peers* no modelo híbrido de comunicação também fornecem um bom refinamento do IM, graças a este tirar vantagem do modelo *Spatial*. Neste caso a comunicação por *unicast* também é a melhor opção.

Contudo, com um refinamento mais grosseiro, em que apenas se define que eventos um jogador de dada região pode subscrever, e havendo a forte possibilidade de nesta região estarem muito utilizadores, o *unicast* é inviabilizado. Em alternativa, cada região pode representar um grupo *multicast* para disseminar por esse grupo os eventos dessa região (Yu e Vuong 2005; Rhalibi e Merabti 2006; Hu et al. 2008).

2.6.2.2 Problemas com o Application-Level Multicast

Um problema significante no ALM é o potencial problema de latência: um evento de jogo pode passar por vários *routers* antes de chegar ao destino final. Para contornar este problema foi sugerido que fosse construída uma *multicast tree* de acordo com a proximidade dos *peers* no jogo em vez da proximidade na rede (Fan et al. 2010).

2.6.3 Non-Player Characters Host Allocation

Para além dos jogadores normais, controlados por uma pessoa através da aplicação cliente, pode haver no jogo vários *Artificial Intelligence-Controlled* NPCs. O jogo, neste caso, tem de garantir a presença destes personagens de jogo, que tradicionalmente são geridos através de um servidor central, consumindo significante poder de processamento e largura de banda. Uma das soluções para este problema é conseguir alojar estes NPCs usando recursos disponíveis nas mais comuns máquinas dos jogadores (Fan et al. 2010).

2.6.3.1 Region Based Approach

A *Region Based Approach*, após dividir o universo em múltiplas regiões, atribui cada uma dessas regiões a um *super-peer*. Na proposta sugerida por Knutsson et al., um *peer* activo cujo *peerId* seja numericamente próximo do *regionID* é escolhido como o coordenador da região. Por sua vez, na proposta de Iimura et al. é escolhido o primeiro *peer* a juntar-se a uma região como *zone owner* (Knutsson et al. 2004; Iimura et al. 2004).

Este modelo tem de levar em consideração as seguintes desvantagens: como apenas um *super-peer* tomará conta de uma região, a excessiva computação e comunicação pode bloquear esse *super-peer*. O critério de seleção é demasiado simples, pois não tem em consideração os recursos disponíveis do *peer* a escolher (Fan et al. 2010).

2.6.3.2 Virtual Distance Based Approach

A ideia principal deste modelo é atribuir um NPC ao jogador mais próximo, devido à grande possibilidade deste interagir com ele. Comparando com o *Region Based Approach*, este modelo é o melhor a usar os recursos de computação, uma vez que usa mais máquinas de

jogadores. Contudo, existem fortes desvantagens: partindo do princípio que o jogador mais próximo interage mesmo com esse NPC, nada impede que outros jogadores o façam. É até usual um grupo de jogadores interagir com o mesmo NPC. Os custos de computação tornam-se elevados, e a constante mudança do jogador no terreno de jogo motiva também a mudança do jogador a controlar um NPC, e isto acontece com muita frequência. Fazer batota pode também tornar-se mais fácil, os jogadores podem aproveitar-se do facto de alojarem o NPC para tirarem vantagens disso (Yonekura et al. 2004; Hu et al. 2008; Bharambe 2006).

2.6.3.3 Heterogeneous Task Sharing

Este mecanismo distribui NPCs pelos jogadores participantes de acordo com os recursos de processamento que podem providenciar. O modelo envolve três partes: uma *work source*, que é responsável por gerar tarefas para os NPCs; os *resource providers*, que são participantes do jogo que têm recursos disponíveis; e, finalmente, *matchmakers*, que vai distribuir as tarefas pelos recursos disponíveis (Fan et al. 2007).

2.7 Game State Persistency

Um MMOG é também usualmente referido como um mundo persistente devido ao facto de estar sempre disponível para os utilizadores, mesmo quando alguns não estão de momento a jogar. O MMOG tem de ter a capacidade de armazenamento entre sessões. É um grande desafio para o P2P MMOG ter uma estrutura de armazenamento desenhada para suportar partilha de ficheiros, pois as existentes raramente cumprem o desempenho e a segurança necessária (James e Walton 2004).

2.8 Cheating Mitigation

Este tópico é um enorme desafio para o P2P MMOG pois é no facto de os *peers* não estarem sob o controlo do produtor da aplicação que reside o grande problema. Algumas das propostas vão ao encontro de forçar uma ordem de eventos ou apenas detetar inconsistências nos pedidos que possam levar a uma suspeição e correção dos resultados (Fan et al. 2010).

2.9 Mecanismos de Incentivo

Por natureza, as aplicações P2P são sistemas voluntários de partilha. Os benefícios deste sistema têm por base a cooperação. Foram criados mecanismos de incentivo para mitigar comportamentos anti-cooperação. O problema prende-se com o fato de o cliente não querer partilhar recursos, mas mesmo assim beneficiar dos recursos dos outros. É aqui que estes mecanismos se tornam necessários, de modo a convencer os participantes a contribuir com os seus recursos (Zghaibeh e Anagnostakis 2007).

2.10 Vantagens e Desvantagens do modelo Peer-to-Peer

Algumas vantagens do modelo P2P são: a sua robustez, a maior escalabilidade, o menor custo operacional e a baixa latência. Em relação à robustez, esta resulta do facto de não haver um servidor que possa falhar, mas um *peer*. A falha desse *peer* não afeta mais nenhum *peer* a não ser o seu. Esta característica permite que o *down-time* da aplicação seja quase impossível. Como cada *peer* se hospeda a si próprio, torna o sistema escalável, não havendo custos inerentes ao aumento de *peers* na rede. A latência também melhora, devido à possibilidade de se comunicar diretamente com um *peer* desejado (Gilmore e Engebrecht 2012).

As desvantagens prendem-se no facto do *peer* ser desconhecido, e poder não ser de confiança. O *peer* pode também não possuir poder de processamento ou largura de banda necessária para este tipo de aplicações.

Capítulo 3

WebRun - versão Cliente/Servidor

O *WebRun* é um jogo, que se insere no mercado emergente dos *exergames*. Um *exergame* é um jogo que insere uma vertente de exercício físico na sua jogabilidade (Oh e Yang 2010; Stach 2012). Para se jogar o *WebRun* é necessário que cada jogador disponha de acesso a uma bicicleta de manutenção que deverá dispor de um sistema de comunicação, designado por controlador, com um computador. O computador, onde previamente se instalou a aplicação Cliente do *WebRun*, tem de se encontrar preparado para receber informação do controlador. Através de uma ligação à aplicação Servidor do *WebRun*, que poderá ser na rede local ou na internet, o jogador interage com outros jogadores. Através da interação da aplicação Cliente com o *Google Maps* obtém-se um mapa relativo ao percurso, com o caminho a percorrer assinalado. Sempre que se encontre disponível o percurso no *Google Street View* é possível visualizar a nível do solo imagens da região que se está a percorrer. O *Google Street View* disponibiliza imagens panorâmicas de 360º na horizontal e 290º na vertical. Na Figura 3.1 é esquematizada a arquitetura do jogo.

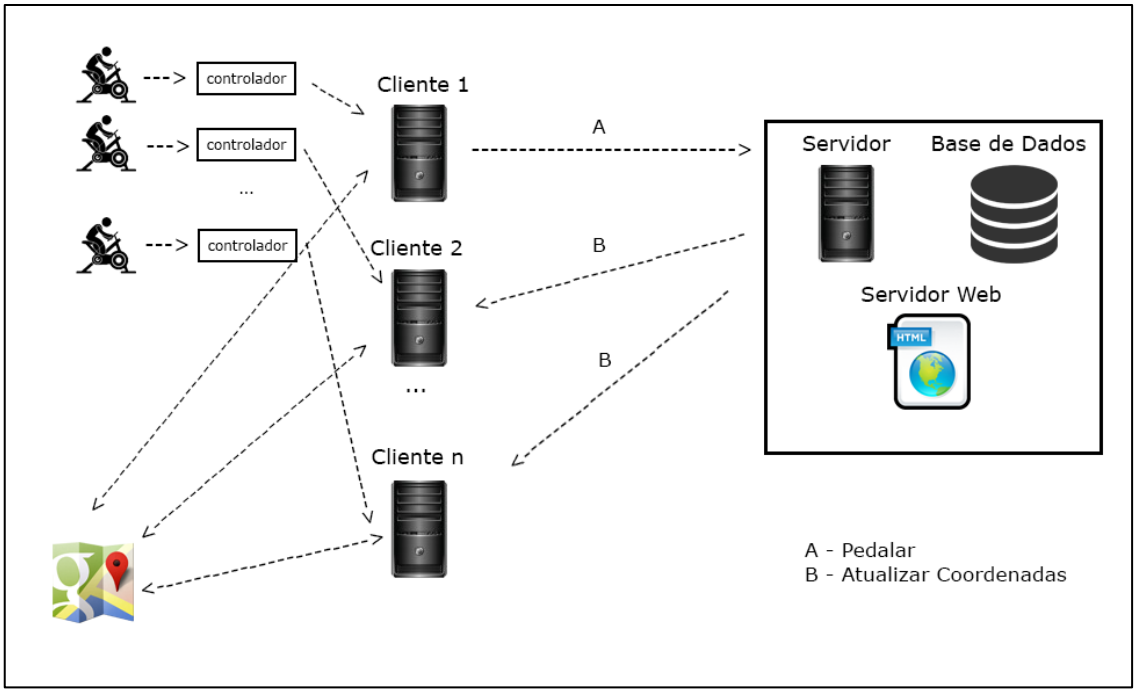

Figura 3.1 - Arquitetura C/S do jogo *WebRun*.

A aplicação interage com uma BD, onde possui informação referente aos utilizadores, às corridas e aos treinos. As corridas consistem num jogo realizado entre vários jogadores, que interagem entre si vendo o percurso do grupo no ecrã da sua aplicação. Um treino é um jogo realizado por um jogador sozinho, para praticar quando não tem adversários para convidar, ou simplesmente porque quer disfrutar do jogo sem concorrência. O esquema dessa BD é apresentado na Figura 3.2.

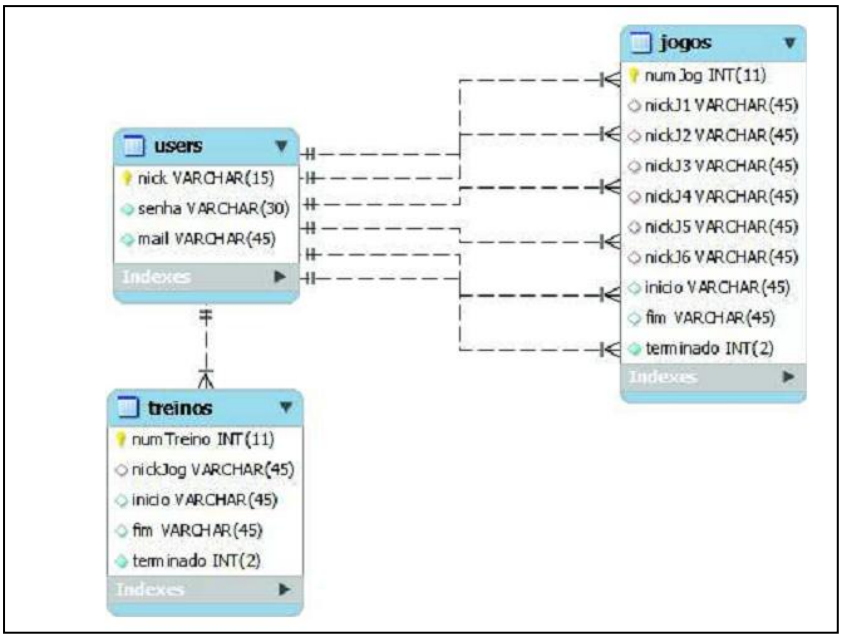

Figura 3.2 - Esquema da Base de Dados do *WebRun* - versão Cliente/Servidor.[7]

3.1 Aplicação *WebRun* - versão Cliente/Servidor

A linguagem com que a aplicação foi desenvolvida foi a linguagem *C#*, HTML e Java Script. Em relação à BD foi usado o *MySQL* e usada a linguagem SQL.

Esta aplicação destina-se a proporcionar a qualquer interessado uma experiência de jogo no *WebRun*. A aplicação funciona apenas em sistemas operativos *Windows*, na sua versão *XP* e posteriores (Castanheira 2012).

3.1.1 Aplicação Cliente

Após feito o *download* da aplicação Cliente, para iniciar o jogo é necessário fazer um registo. Os jogadores são identificados por um *nickname* e uma *password*, necessários para aceder à aplicação Cliente. Ao efetuar o *login* é estabelecida uma ligação com o servidor, e após um

[7] (Castanheira 2012)

processo de autenticação realizado pelo servidor é-nos apresentada a janela principal do jogo *WebRun* (ver Figura 3.3). Nesta janela é permitido ao jogador ver a lista de utilizadores ligados ao *WebRun*, é possível também ver para cada um o estado em que se encontra. O estado de cada jogador será um de quatro possíveis (Castanheira 2012):

- Amarelo: em espera de iniciar uma corrida - representa um jogador que aceitou entrar numa corrida e está à espera que a mesma seja iniciada.
- Vermelho: a treinar - representa um jogador que está a praticar um treino.
- Verde: a participar numa corrida - representa um jogador que está a participar numa corrida com outros utilizadores.
- Preto: em espera - representa um jogador que não está pendente de nenhuma ação, podendo ser convidado para uma corrida.

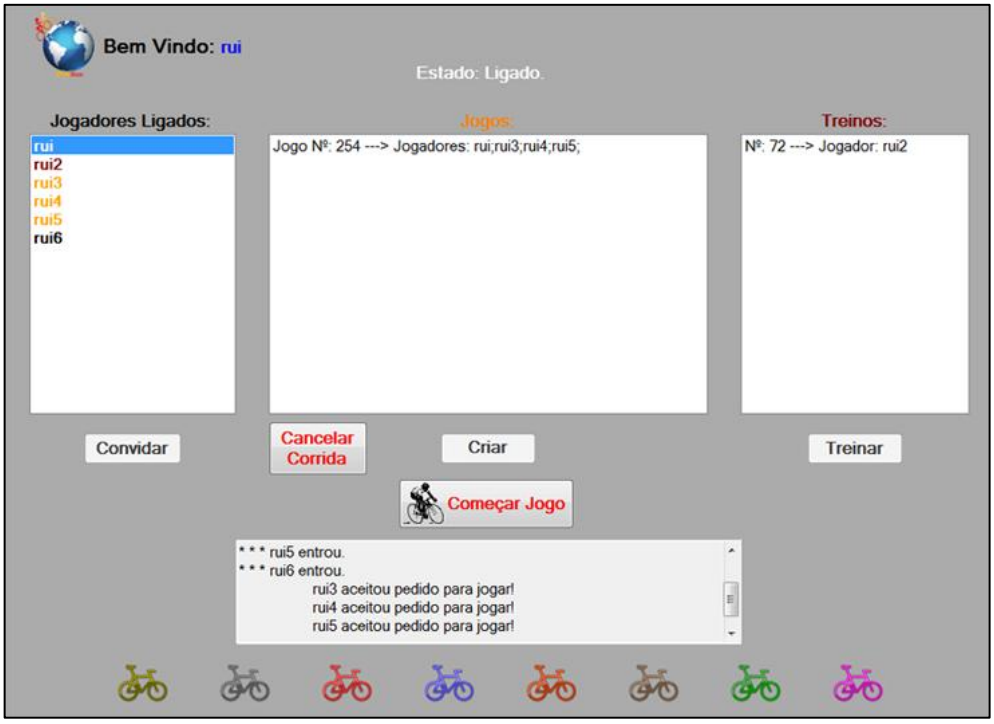

Figura 3.3 - Janela principal do jogo *WebRun* (aplicação Cliente).

É também nesta janela que é permitido ao jogador criar um treino ou uma corrida, convidar outros jogadores para entrar num jogo. Ao escolher criar uma corrida (ou treino) é enviada uma mensagem ao servidor com o percurso a realizar e com quantos adversários será realizado (em caso de treino será sem adversários, no caso de corrida poderá ter entre 1 a 5 adversários). Após a criação da corrida, na janela principal da aplicação é ativada a opção

para convidar os adversários e é enviada uma mensagem ao servidor com o identificador do adversário a convidar e a respetiva corrida. Se o adversário aceitar o desfio, é adicionado à corrida pelo servidor que informa o criador do jogo dessa opção.

Depois de todos os adversários aceitarem juntar-se à corrida, a mesma é iniciada pelo jogador que procedeu à sua criação. O servidor comunica aos jogadores o início da corrida e cada jogador carrega a janela de jogo (ver Figura 3.4). Esta janela possui, para além de alguns botões para controlo das imagens obtidas do *Google Maps*, cinco *frames* que se descrevem de seguida e que na Figura 3.4 se encontram numeradas de i) a v).

Figura 3.4 - Janela do jogo.

i) *Frame* Jogadores: mostra a lista de todos os jogadores na corrida (não necessário para o treino, caso em que apenas aparece o próprio jogador);

ii) *Frame* Finalistas: vai mostrando os jogadores que já terminaram a corrida;

iii) *Frame* Principal: apresenta uma secção do mapa do percurso visto de cima, com o percurso assinalado e as bicicletas nele posicionadas consoante o percurso já efetuado. Cada jogador é identificado por uma bicicleta com a sua cor;

iv) *Frame Street View*: quando possível, mostra o *Google Street View* da zona onde o jogador se encontra;

v) *Frame* Percurso: mostra o mapa do percurso com a totalidade do caminho a percorrer assinalado;

Depois de carregada a janela do jogo, cada jogador pode começar a pedalar. Sempre que um jogador pedala, a sua posição no mapa é atualizada e é enviada para o servidor uma

mensagem com o *nickname* do jogador, a identificação da corrida em que participa e as coordenadas da sua posição.

Ao receber essa mensagem, o servidor vai reenviá-la para todos os seus adversários da corrida. Assim, cada jogador (isto é, cada aplicação cliente) da mesma corrida irá receber uma atualização da posição dos jogadores correspondentes, acedendo ao servidor do Google para redesenhar o mapa.

Quando o primeiro jogador chega às coordenadas finais é anunciado o vencedor na *frame* Finalistas. Sempre que um novo jogador termina, é adicionado o seu *nickname* à mesma *frame*. Terminado o jogo, todos os jogadores da corrida voltam para a janela principal do *WebRun*.

3.1.2 Aplicação Servidor

É esta aplicação que contém o motor do jogo. É o Servidor que regista os jogadores na BD, faz a sua autenticação quando acedem ao jogo e faz a gestão de cada corrida (ou treino) criada. Ao ser executada, a aplicação Servidor apresenta uma janela de administração (ver Figura 3.5) com um botão para iniciar e outro para desligar a aplicação. A janela contém informação sobre a data e hora em que o servidor foi ligado, quais os jogadores que se encontram ligados, quais os jogos e quais os treinos em curso. Contém ainda uma *frame* que funciona como *log* de todas as ações realizadas no jogo.

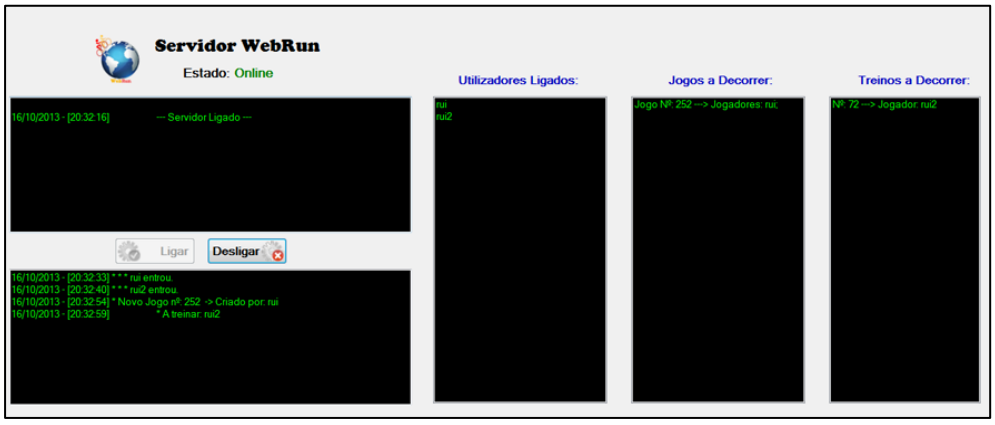

Figura 3.5 - Janela principal do jogo *WebRun* (aplicação Servidor).

O Servidor contém uma *thread* principal que espera por ligações de clientes. Quando um jogador registado efetua o login é criada uma *thread* para comunicar com esse jogador. A comunicação entre as aplicações Cliente e Servidor é centralizada no Servidor e todo o tráfego tem obrigatoriamente de ser encaminhado pelo servidor para os clientes. Se o número de jogadores ligados em simultâneo aumentar consideravelmente, o Servidor irá ser um ponto

de estrangulamento do jogo, diminuindo o tempo de resposta aos Clientes. A comunicação com o *Google Maps,* para a visualização da posição no mapa, é a única que é efetuada diretamente pela aplicação Cliente. Por fim, a comunicação entre a bicicleta e a aplicação Cliente é feita através de um controlador.

Capítulo 4

WebRun - versão *Peer-to-Peer*

Foi necessário estudar cuidadosamente a versão C/S do *WebRun* para perceber que alterações seriam necessárias efetuar para que nos fosse possível desenvolver na aplicação a arquitetura P2P. Neste capítulo iremos descrever essas alterações e a forma como afetaram a aplicação. Antes de qualquer mudança no tipo de comunicação, começamos por retirar as coordenadas de um percurso para efeitos de teste. Todas as coordenadas calculadas pela aplicação foram enviadas para um ficheiro.

4.1 Aplicação *WebRun* – versão *Peer-to-Peer*

Nesta versão do jogo, o registo de novos jogadores mantém-se na aplicação Servidor. O mesmo acontece com a operação de *login* e com o processo da criação de uma corrida. É a partir do momento em que uma corrida é criada que surgem as alterações: o jogador que criou essa corrida torna-se o Servidor da mesa. No caso dos treinos nada altera, mantendo-se o esquema da versão C/S. Repare-se que nos treinos não existe comunicação entre as aplicações Cliente e Servidor, pois é a aplicação Cliente que comunica com o *Google Maps* e atualiza o seu mapa do percurso.

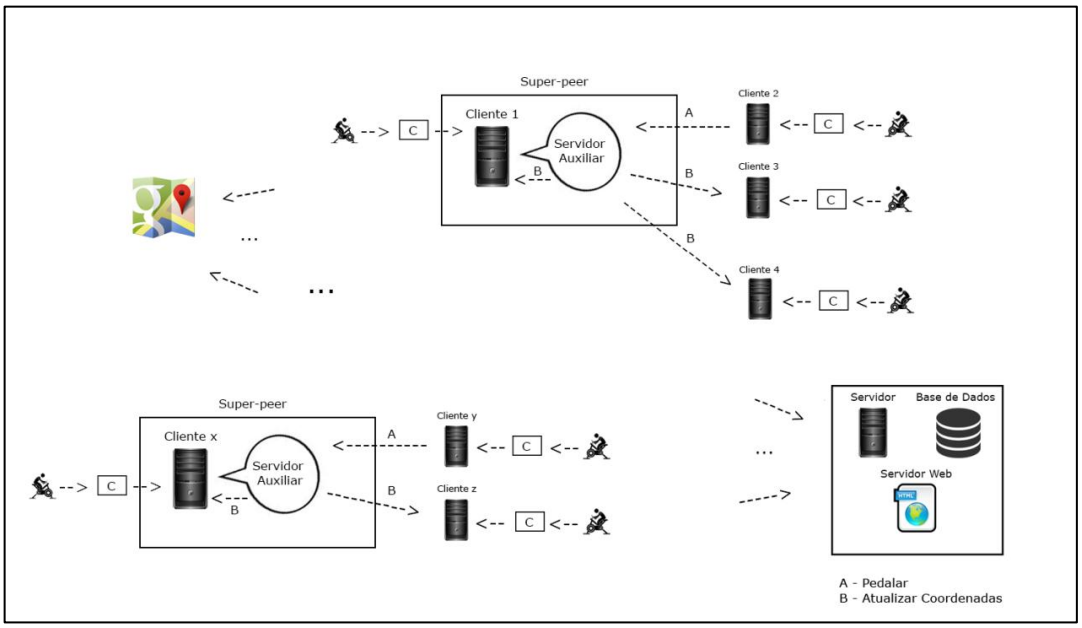

Figura 4.1 - Arquitetura da versão P2P do *WebRun*.

No caso de uma corrida, o jogador responsável pela sua criação funcionará como um *super-peer* e será da sua responsabilidade receber e disseminar as posições de cada jogador da corrida para os adversários, de forma a ser possível a atualização da posição dos jogadores no mapa do percurso. Terminada a corrida, é anunciado o vencedor pelo *super-peer*. Após terminar a corrida todos os jogadores voltam para a janela principal, passando novamente a ser controlados pelo Servidor central. A arquitetura da nova versão está esquematizada na Figura 4.1.

4.1.1 Nova Aplicação Cliente

A interface da aplicação para o jogador não vai ter nenhuma diferença percetível em comparação com a versão C/S. As mudanças são apenas estruturais na forma como a comunicação é feita com a aplicação servidor.

Na nova aplicação Cliente da máquina do jogador que cria uma nova corrida será instanciada uma nova *Thread* que ficará responsável pela gestão dessa corrida. Essa nova *Thread*, a que chamamos "gestora da corrida", irá desempenhar o papel de Servidor para a nova corrida e a aplicação Cliente passa a desempenhar um papel de *super-peer*, isto é, será Cliente e Servidor ao mesmo tempo. A *Thread* gestora da corrida ficará à espera das ligações dos jogadores que depois de convidados aceitem jogar essa corrida. Para cada jogador será criada uma *Thread* para a comunicação entre a aplicação Cliente e a *Thread* gestora da corrida. Finalmente, a *Thread* gestora da corrida será responsável por receber "o pedalar" de cada jogador da corrida e disseminar pelos seus adversários, que irão atualizar nos seus mapas quer a sua posição como as dos seus adversários. À medida que os jogadores vão chegando ao fim da corrida, os seus *nickname* aparecem na *frame* Finalistas da janela do jogo de cada um dos jogadores.

4.1.2 Nova Aplicação Servidor

Como já foi referido, o Servidor vai continuar a ser o responsável pelo registo, autenticação e criação de corridas, mas vai perder algumas das suas funções. A partir do momento em que todos os convidados para uma corrida a aceitam, a função de gerir o pedalar dos vários concorrentes dessa corrida passa para o *super-peer* do jogador responsável pela criação da mesma. No momento em que a aplicação Servidor permite que o criador do jogo dê início à corrida, envia na mensagem, para todos os jogadores da corrida, o endereço (IP e porto) do cliente que se tornou o *super-peer* do jogo e que assumirá a partir desse momento a gestão do jogo.

Uma vez terminada a corrida, a responsabilidade volta para o Servidor, de forma a gerir os jogadores que continuam ligados à aplicação.

4.1.3 Nova Comunicação entra a Aplicação Cliente e a Aplicação Servidor

Nesta nova arquitetura a comunicação deixa de estar centralizada no Servidor, mas a autenticação continua a ser da responsabilidade do Servidor, nomeadamente a criação de jogos e os convites para os mesmos. A partir do início de um jogo, o criador do mesmo torna-se um *super-peer* e vai controlar essa região, ou jogo. Desta forma, o Servidor deixa de ser entupido por comunicações desnecessárias. Toda a comunicação durante jogo passa a ser canalizada pelo *super-peer*.

A comunicação efetuada entre Cliente e o Servidor é feita da mesma forma até ao inicio do jogo. Aí, e uma vez iniciada a janela do jogo em que todos os concorrentes podem começar a pedalar, cada vez que pedalam é enviada uma mensagem ao *super-peer*, que a dissemina por todos os participantes desse jogo para que em todos os jogos seja atualizada a posição do jogador. Terminando a corrida, é anunciado o vencedor pelo *super-peer* e, terminando a ligação, voltam para a janela principal passando novamente a ser controlados pelo servidor.

Capítulo 5

Avaliação Experimental

Para estudar a escalabilidade das duas versões do jogo foi simulada a execução simultânea de várias corridas em cada versão. Nessas simulações foram usados 26 computadores ligados em rede, com o sistema operativo *Windows XP Professional* com o *Service Pack 3*, um processador *Intel Core2 Quad CPU Q6600 @ 2.40GHz*, com 3 *GB* de memória *RAM* e com uma placa de rede *Atheros L1 Gigabit Ethernet 10/100/1000Base-T*.

5.1 Metodologia

Para simular a execução dos jogos, foi usado um ficheiro onde previamente se armazenaram as coordenadas do percurso de uma corrida exemplo. Esse ficheiro foi colocado em todas as máquinas que contêm a aplicação Cliente e nesta foi criado um *Timer* com uma frequência de 200 milissegundos, para simular 5 pedaladas por segundo. De cada vez que o *Timer* expira é lido um novo par de coordenadas do ficheiro que contém o percurso e que depois é enviado ao Servidor.

O Servidor, ao receber a mensagem com as coordenadas, e identificando o jogador que a enviou, vai disseminar a nova posição do jogador aos seus adversários. Na versão P2P, o Servidor que recebe a mensagem é a *Thread* "gestora da corrida" do *super-peer* que criou a corrida. Esta mensagem é equivalente à mensagem enviada pelo controlador aquando do pedalar do jogador.

As mensagens para registo, login de utilizador, convites para a corrida e aceitação de convites não foram consideradas para a medição dos tempos de resposta do jogo. Isto é, supôs-se que os jogos estavam iniciados e mediu-se o tempo de resposta de cada jogador durante o processo em que apenas se estavam a realizar corridas e nenhum treino.

Para obter o tempo médio de resposta do Servidor a cada jogador foram guardados no Cliente, para cada mensagem recebida, o identificador do jogador que pedalou e o *timestamp* da máquina ao receber a mensagem. Os dados foram processados de modo a obter, em primeiro lugar, o tempo médio de resposta para cada jogador, considerando todas as mensagens recebidas durante a simulação da corrida.

No caso da versão C/S, calculou-se depois o valor médio dos tempos de resposta considerando todos os jogadores em jogo, em todas as corridas, durante a simulação.

Para a versão P2P, as mensagens que cada jogador recebe durante a fase de "pedalar" vêm do seu *super-peer*. Assim, calcular o tempo médio de resposta a cada jogador é calcular o tempo médio de resposta a todos os jogadores da mesma corrida.

Para ambas as versões do jogo foi simulado 1 corrida com 2 jogadores, e *i* corridas com 5 jogadores cada, fazendo variar *i* de 1 até 5. Foram assim obtidos os tempos médios de resposta aos processos Clientes quando considerados 2, 5, 10, 15, 20 e 25 jogadores. O processo Servidor do jogo foi executado numa máquina dedicada e cada Cliente foi executado numa das outras máquinas disponíveis para a simulação.

5.2 Resultados

O gráfico apresentado na Figura 5.1, mostra os tempos médios de resposta aos jogadores do jogo *WebRun*, durante a fase em que os jogadores pedalam, para as duas versões do jogo, C/S e P2P, considerando o número total de jogadores: 2 (1 corrida * 2 jogadores), 5 (1 corrida * 5 jogadores), 10 (2 corridas * 5 jogadores), 15 (3 corridas * 5 jogadores), 20 (4 corridas * 5 jogadores), 25 (5 corridas * 5 jogadores).

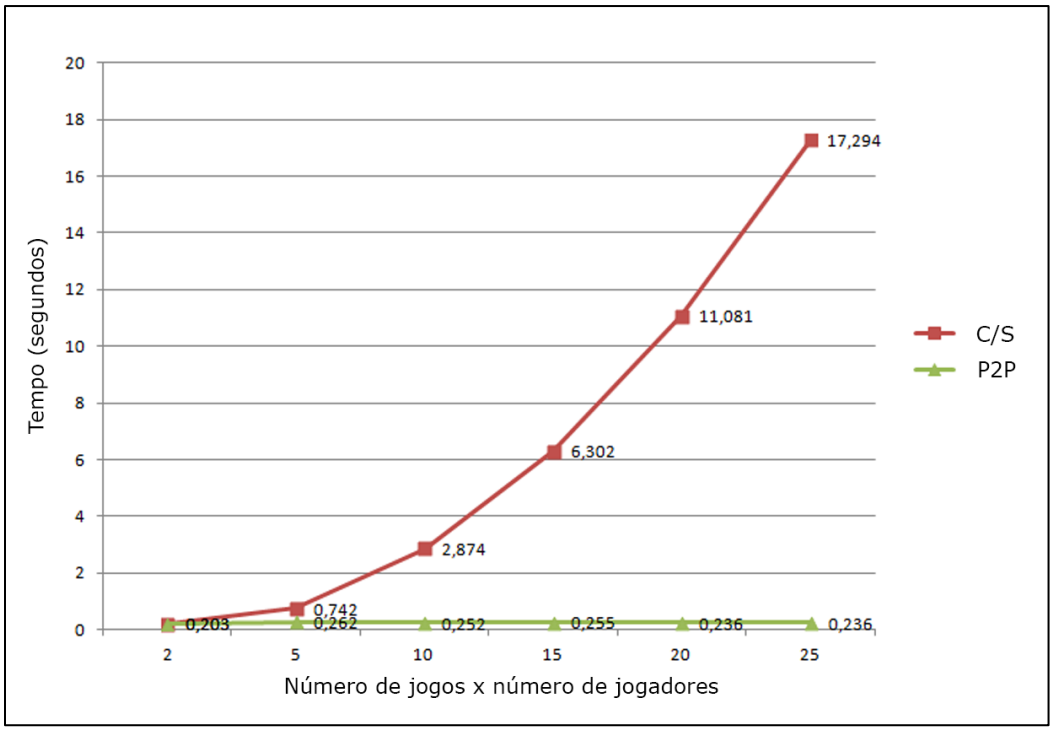

Figura 5.1 - Gráfico da média dos tempos de resposta aos clientes nas arquiteturas C/S e P2P, quando todos os jogadores pedalam, variando o número de jogadores de 2 a 25.

Observando os valores obtidos para a versão C/S, podemos concluir que o tempo de resposta ao jogador cresce exponencialmente com o aumento do número de jogadores. O tempo

obtido para 5 corridas (25 jogadores), cerca de 17 segundos, mostra que dificilmente o jogo seria praticável, para um número razoável de corridas, mesmo que o servidor fosse uma máquina mais potente.

Para a versão P2P, podemos observar que o tempo de resposta é independente do número de jogadores, permanecendo perto de 0,2 segundos. Podemos concluir que o jogo é perfeitamente escalável em relação ao aumento do número de jogadores, quando se ignora o tempo de inicialização das corridas.

Para melhor se perceber as diferenças entre as duas arquiteturas analisámos as mensagens trocadas entre jogadores e Servidor para a versão C/S e entre jogadores e *super-peer* para a versão P2P. Mais uma vez, não são contabilizadas as mensagens supondo que cada jogador pedala uma única vez.

Na versão C/S, considerado uma corrida com 2 jogadores (sejam jogador "x" e jogador "y"), cada jogador envia uma mensagem "pedalar" ao Servidor, e o Servidor envia para "x" a mensagem de que "y" pedalou e envia para "y" a mensagem de que "x" pedalou. Portanto, com 2 jogadores, o Servidor recebe 2e envia outras 2 mensagens. Com 5 jogadores, o servidor recebe 5 mensagens, uma de cada jogador, e por cada uma delas, vai enviar 4, isto é, atualizando as coordenadas, dos jogadores adversários. Assim, com 5 jogadores, o Servidor recebe 5 e envia 20 mensagens. O gráfico da Figura 5.2 mostra o número de mensagens recebidas e enviadas pelo Servidor quando o número de jogadores varia entre 2 e 25, como descrito anteriormente, e quando cada jogador pedala uma única vez. Como se pode ver, o número total de mensagens recebidas mais as enviadas passa de 4 com 2 jogadores para 125 com 25 jogadores. Mesmo considerando que uma pedalada a cada 200 milissegundos é muito rápido, e que numa situação real duas pedaladas por segundo será uma velocidade razoável, o número de mensagens trocadas entre servidor e jogadores durante uma corrida será enorme.

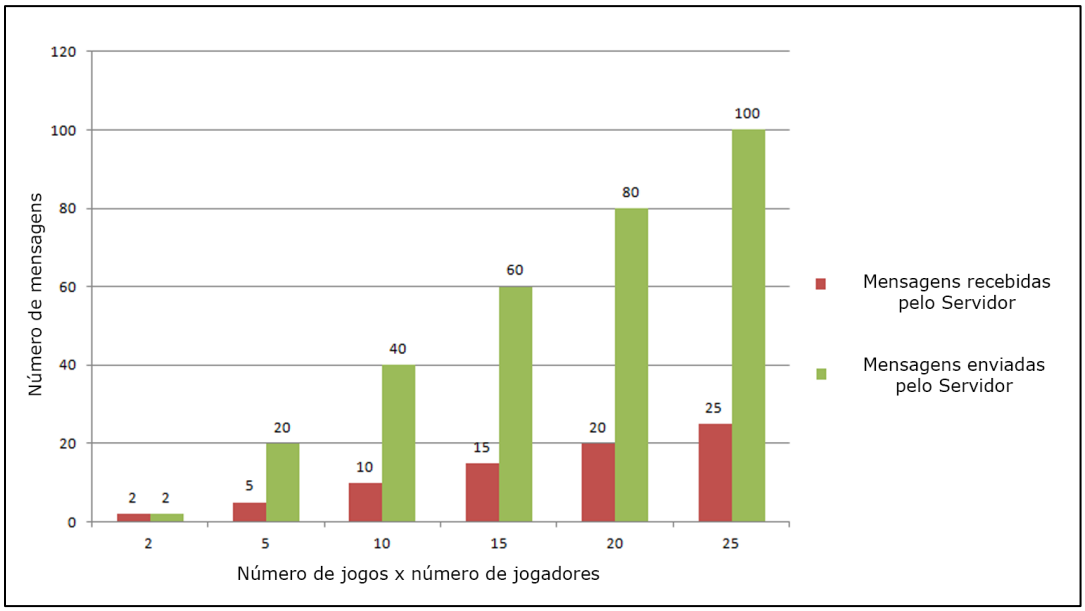

Figura 5.2 - Gráfico com o número de mensagens enviadas e recebidas pelo servidor na arquitetura C/S considerando apenas jogadores em jogo a pedalar uma única vez e variando o número de jogadores de 2 até 25.

Por sua vez, na versão P2P, o número de mensagens recebidas e enviadas por um *super-peer* apenas depende do número de jogadores da corrida controlada pelo *super-peer*. Executar em simultâneo várias corridas apenas vai ter impacto nas mensagens trocadas com o Servidor principal na fase inicial das corridas. Na fase de pedalar, o número de mensagens trocadas com o *super-peer* é independente do número de corridas. Analisando o número de mensagens recebidas e enviadas pelo *super-peer*, quando aumenta o número de jogadores numa corrida, e considerando também apenas se pedala uma vez, observa-se o seguinte: numa corrida com 2 jogadores o *super-peer* recebe 2 mensagens "pedalar" e envia outras 2, uma para cada jogador. Num jogo com 3 jogadores, o *super-peer* recebe 3 mensagens pedalar, e por cada uma que recebe, envia 2 para os jogadores adversários, isto é, envia 6 mensagens. No gráfico da Figura 5.3 podemos observar que quando o número de jogadores de uma corrida aumenta de 2 para 6, o número total de mensagens (o total das recebidas e das enviadas) pelo *super-peer* varia de 4 para 35, o que, como vimos, é aceitável em termos de tempo de resposta que o jogador pode esperar obter, independentemente do número de corridas que venham a decorrer em simultâneo.

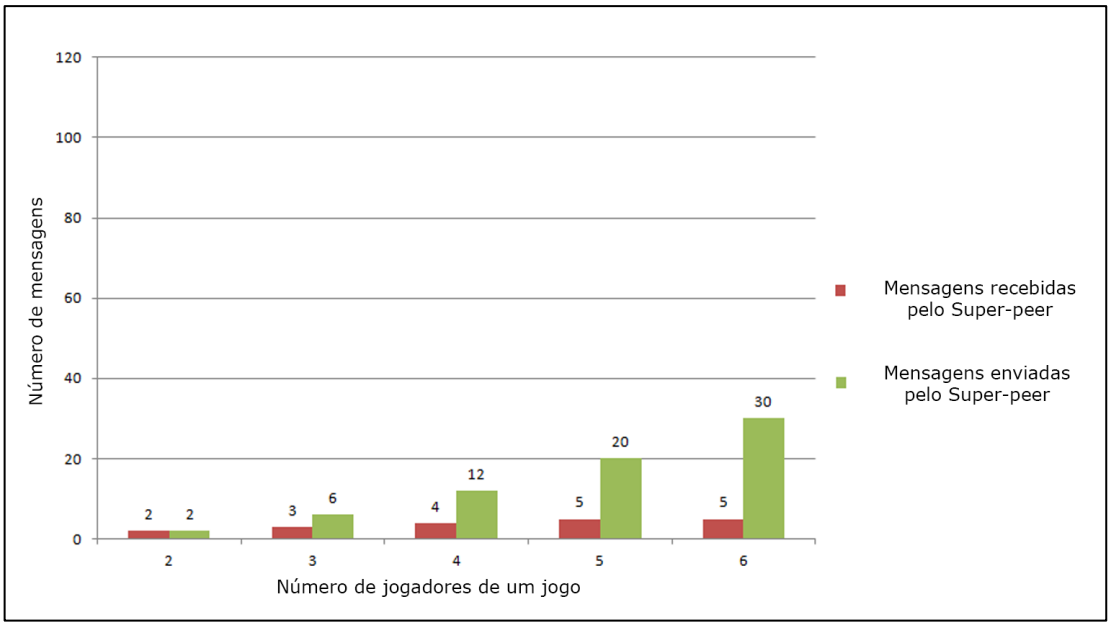

Figura 5.3 – Gráfico com o número de mensagens trocadas pelo Super-peer, na arquitetura P2P, quando cada jogador pedala uma vez, variando o número de jogadores do jogo de 2 a 6.

Capítulo 6

Conclusões e Trabalho Futuro

Neste capítulo da dissertação apresentamos as conclusões retiradas do estudo da escalabilidade das duas arquiteturas, C/S e P2P, para o caso do jogo *WebRun*.

É também nesta secção que vamos apresentar algumas ideias a explorar para a evolução da aplicação, e para a exploração do problema da escalabilidade em jogos *online*.

6.1. Conclusões

Nesta dissertação foi proposta a arquitetura P2P para um jogo *online* que promove a atividade física. Partindo do protótipo do jogo *WebRun*, construído segundo a arquitetura C/S, foi desenvolvida uma versão P2P para o mesmo jogo. Sendo cada corrida do *WebRun* jogada por pequenos grupos de jogadores (2 a 6 elementos), foi proposto passar o controlo do estado de cada corrida, ou jogo, do servidor central para a máquina do jogador que cria o jogo, criando um *super-peer*. A avaliação do tempo de resposta a cada processo Cliente mostrou que a nova arquitetura (P2P) é completamente escalável em relação ao número de corridas que decorrem em simultâneo. A mesma avaliação mostrou que a versão C/S não é apropriada para um número elevado de jogadores. Um dos pontos fracos da arquitetura P2P para jogos multiutilizador é a maior possibilidade de fraude. Mantendo centralizado o processo de registo e autenticação, a segurança no acesso ao jogo é preservado. Num tipo de jogo em que o ganho é essencialmente o bem-estar físico e psicológico, a hipótese de fraude no decorrer do jogo não é fator crítico.

6.2. Trabalho Futuro

Como trabalho futuro pretende-se estudar outros modelos de gestão do estado do jogo, por exemplo, considerando a proximidade geográfica dos percursos de diferentes corridas.

Outra área a explorar será a distribuição do estado persistente do jogo pelos *super-peers*, de forma a descentralizar a processo de registo e autenticação dos jogadores.

Referências

Araújo, P.; Prata, P.; Dias, T.; Barroso, J.; Castanheira, J. (2011) “WebRun - Promoting Healthy Sport Activity Through Interactive Online Games”, in Proceedings of CISTI, IEEE Computer and Communications Societies.

Aurenhammer F.; Klein, R. (2000) "Voronoi Diagrams”, in Handbook of Computational Geometry, Elsevier Science Publisher, pp. 201-290.

Bauer, D.; Rooney, S.; Scotton, P. (2002) “Network Infrastructure for Massively Distributed Games”, in Proceedings of NetGames, Association for Computer Machinery, pp. 36-43.

Benford, S.; Fahlen, L. E. (1993) “A Spatial Model of Interaction in Large Virtual Environments”, in Proceedings of ECSCW, IEEE Computer and Communications Societies, pp. 107-123.

Bharambe, A. (2006) “Colyseus: A Distributed Architecture for Online Multiplayer Games”, in Proceedings of NSDI, USENIX, pp. 3-6.

Boulanger, J.-S.; Kienzle, J.; Verbrugge, C. (2006) “Comparing Interest Management Algorithms for Massively Multiplayer Games”, in Proceedings of NetGames, Association for Computer Machinery.

Brox, E.; Luque, L.F.; Evertsen, G. J.; Hernandes, J. E. G. (2011) “Exergames for elderly: Social exergames to persuade seniors to increase physical activity” in 5th International Conference on Pervasive Computing Technologies for Healthcare (PervasiveHealth), pp.546-549.

Buyukkaya, E.; Abdallah, M. (2008) “Data Management in Voronoi-Based P2P Gaming”, in Proceedings of CCNC, IEEE Computer and Communications Societies, pp. 1050-1053.

Castanheira, J. (2012) “WebRun: Jogo Web interactivo para promoção da prática desportiva”. Dissertação de Mestrado em Engenharia Informática, Universidade da Beira Interior – Departamento de Informática.

Chambers, C.; Feng, W.-C.; Sahu, S.; Saha, D.; Brandt, D. (2010) “Characterizing Online Games”, in IEEE/ACM Transactions on Networking (TON), vol. 18, n. 3, pp. 899-910.

Chen, J.; Wu, B.; Delap, M.; Knutsson, B.; Lu, H.; Amza, C. (2005) “Locality Aware Dynamic Load Management for Massively Multiplayer Games” in Proceedings of PPoPP, Association for Computer Machinery, pp. 289-300.

Consel, C.; Réveillère, L. (2003) “A Programmable Client-Server Model: Robust Extensibility via DSLs”, in Proceedings of ASE, IEEE Computer and Communications Societies, pp. 70-79.

Diot C.; Gautier, L. (1999) “A Distributed Architecture for Multiplayer Interactive Applications on the Internet” in Network, IEEE Computer and Communications Societies, vol. 13, pp. 6-15.

Diot, C.; Levine, B. N.; Lules, B.; Kassem, H.; Balensifen, D. (2000) “Deployment Issues for the IP Multicast Service and Architecture”, Network, IEEE Computer and Communications Societies, vol. 14, pp. 78-88.

Douglas, S.; Tanin, E.; Hardwood, A. (2005) “Enabling Massively Multi-Player Online Gaming Applications on a P2P Architecture”, in Proceedings of ICIA, IEEE Computer and Communications Societies, pp. 7-12.

El-Sayed, A.; Roca, V.; Mathy, L. (2003) “A survey of Proposals for an Alternative Group Communication Service”, Network, IEEE Computer and Communications Societies, vol. 17, n. 1, pp. 46-51.

Fan, L.; Taylor, H.; Tinder, P. (2007) “Mediator: A Design Framework for P2P MMOGs”, in Proceedings of NetGames, Association for Computer Machinery, pp.43-48.

Fan, L.; Taylor, H.; Trinder, P. (2010) “Design Issues for Peer-to-Peer Massively Multiplayer Online Games”, International Journal of Advanced Media and Communication, vol. 4, n. 2, pp. 108-125.

Fieldler, S.; Wallner, M.; Weber, M. (2002) “A Communication Architecture for Massive Multiplayer Games”, in Proceedings of NetGames, Association for Computer Machinery, pp. 14-22.

Gilmore, J. S.; Engebrecht, H. A. (2012) “A Survey of State Persistency in Peer-to-Peer Massively Multiplayer Online Games”, Parallel and Distributed Systems, IEEE Transactions, vol. 23, n. 5, pp 818-834.

Hampel, T.; Bopp T.; Hinn, R. (2006) “A P2P Architecture for Massive Multiplayer Online Games” in Proceedings of NetGames, Association for Computer Machinery, Nova Iorque, USA.

Hu, S.-Y.; Liao, G.-M. (2004) “Scalable Peer-to-Peer Networked Virtual Environments”, in Proceedings of NetGames, Association for Computer Machinery, pp. 129-133.

Hu, S.-Y.; Chang, S.-C.; Jiang, J.-R. (2008) “Voronoi State Management for Peer-to-Peer Massively Multiplayer Online Games”, in Proceedings of CCNC, IEEE Computer and Communications Societies, pp. 1134-1138.

Iimura, T.; Hazeyama,H.; Kadobayashi, Y. (2004) “Zoned Federation of Game Servers – a Peer-to-Peer Approach”, in Proceedings of NetGames, Association for Computer Machinery, pp. 116-120.

James, D.; Walton, G. (2004) “2004 Persistent Worlds Whitepepar”, IGDA Online Games SIG, www.igda.org.

Knutsson, B.; Lu, H.; Xu, W; Hopkins, B. (2004) “Peer-to-peer support for massively multiplayer games”, INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies, vol.1, pp.107-118.

Lee, J. (2003) “Peer to Peer File Sharing System: What Matters to the End User?” in Communications of the Association for Computer Machinery, v. 46, n. 2, pp 49-53.

Li, J. (2007) “A Survey of Peer-to-Peer Network Security Issues”, acedido em Agosto de 2014 em http://www.cs.wustl.edu/~jain/cse571-07/ftp/p2p.

Lieberman, D. A.; Chamberlin, B.; Medina, E.; Franklin, B. A.; Sanner, B. M.; Vafiadis, D. K. (2011) “The Power of Play: Innovations in Getting Active Summit 2011: A Science Panel Proceedings Report From the American Heart Association”, in Circulation, Dalas.

Lui, J. C. S.; Chan, M. F. (2002) “An Efficient Partitioning Algorithm for Distributed Virtual Environment Systems” in IEEE Transactions on Parallel and Distributed Systems, vol. 13, n. 3, pp. 193-211.

Macedonia, M.; Zyda, M.; Pratt, D.; Barham, P. Zeswitz, S. (1994) “NPSNET: A Network Software Architecture for Large Scale Virtual Environments”, Presence, vol. 3, no. 4, pp. 265-287.

Matsumoto, N.; Kawahara, Y.; Morikawa, K.; Aoyama, T. (2005) “A Scalable and Low Delay Communication Scheme for Networked Virtual Environments”, in Proceedings of GLOBECOM, IEEE Computer and Communications Societies, pp. 529-535.

Morgan, G.; Lu, F.; Storey, K. (2005) “Interest Management Middleware for Networked Games”, in Proceedings of i3D’05, Association for Computer Machinery, Nova Iorque, USA, pp. 57-64.

Neumann, C.; Prigent, N.; Varvello, M.; Suh, K. (2007) “Challenges in Peer-to-PeerGaming”, in Computer Communication Review, ACM SIGCOMM, vol. 37, n. 2, pp. 79-827.

Ng, B.; Lau, R. W. H.; Si, A.; Li, F. W. B. (2005) “Multiserver Support for Large-Scale Distributed Virtual Envioronments”, in IEEE Transactions Multimedia, vol. 7, n. 6, pp. 1054-1065.

Oh, Y.; Yang, S. (2010) “Defining exergames and exergaming”, in Proceedings of Meaningful Play, Michigan State University.

Oracle (1996) “*Oracle7 Server Distributed Systems Manual”,* Oracle Corporations, vol. 1. Oracle Corporation.

Pinho, E. (2009) “Tolerância a Falhas em Jogos On-line”. Dissertação de Mestrado em Engenharia Informática, Universidade da Beira Interior – Departamento de Informática.

Rhalibi, A. E.; Merabti, M. (2006) “Interest Management & Scalability Issues in P2P MMOG”, in Proceedings of CCNC, IEEE Computer and Communications Societies, pp. 1188-1192.

Robles, R. J.; Yeo, S.-S; Moon, Y.-D.; Park, G.; Kim, S. (2008) “Online Games and Security Issues”, in Proceedings of FGCN, IEEE Computer and Communications Societies, 2008.

Sinclair, J.; Hingston, P.; Masek, M. (2007) “Considerations for the design of exergames. In Proceedings of the 5th international conference on Computer graphics and interactive techniques” in Australia and Southeast Asia (GRAPHITE '07), Association for Computer Machinery , pp. 289-295.

Smallwood, S. R.; Morris, M. M.; Fallows, S. J.; Buckley, J. P. (2012) “Physiologic Responses and Energy Expenditure of Kinect Active Video Game Paly in Schoolchildren” in Archives of Pediatrics & Adolescent Medicine, pp 1-5, 2012.

Stach, T. B. (2012) “Heat Rate Ballancing for Multiplayer Exergames”. Tese de Doutoramento em Filosofia, Queen’s University.

Waldfogel, J. (2011) “Is the Sky Falling? The quality of new recorder music since Napster”, in Vox - Research-based policy analysis and commentary from leading economists, acedido em Agosto de 2014 em http://www.voxeu.org/article/was-napster-day-music-died.

Woodcock, B. S. (2005) “An analysis of MMOG Subscription Growth”, www.mmogchart.com.

Yang, B. B.; Garcia-Molina, H. (2003) “Designing a Super-Peer Network”, in Proceedings of ICDE, IEEE Computer and Communications Societies, pp. 49-60.

Yonekura, T.; Kawano, Y.; Hanawa, D. (2004) “Peer-to-Peer networked field-type virtual environment by using atoz”, in Proceedings of Cyberworlds, IEEE Computer and Communications Societies, pp. 241-248.

Yu, A. P.; Vuong, S. T. (2005) “MOPAR: a Mobile Peer-to-Peer Overlay Architecture for Interest Management of Massively Multiplayer online Games”, in Proceedings of NOSSDAV, Association for Computer Machinery, pp. 99-104.

Zghaibeh, M.; Anagnostakis, K. (2007) “On the Impact of P2P Incentive Mechanisms on User Nehavior”, in Proceedings of NetEcon’07, Association for Computer Machinery.