DOCTORATE STUDY PLAN

CARLOS EDUARDO BENEVIDES BEZERRA

A decentralized network support middleware for MMOGs

Porto Alegre, September 2009

TABLE OF CONTENTS

ABSTRACT

This documente describes the full doctorate study plan whose part is intended to be executed in the University of Lugano, Switzerland. It proposes the research and creation of a novel support model for MMOGs (massively multiplayer online games), by using large-scale systems similar to those created for sharing of computing resources, such as file sharing peer-to-peer networks. Following this model, a distributed system could be formed to act as a server for an MMOG. Then, a portion of the game's communication and processing load would be delegated to each node of the server system – more precisely, the game's virtual environment would be partitioned into regions, each of which managed by one or more server nodes. Such a system would be perceived by the players as an unique central servel to which they connect via a gateway. Upon connected, each player is seamlessly redirected to the appropriate server node. However, for such server system to work as expected, some issues must be addressed, such as maintaining the game state consistent among the different game server nodes, so each player has the same vision; management and balancing of the load assigned to the different server nodes; what to do when a node crashes or leaves the server system and, finally, how to broadcast and store the game state to the players in a distributed manner so that its most recent state can be restored even in the presence of system failures.

Keywords: Massively multiplayer online games, MMOG, distributed server, distributed interactive simulation.

1 INTRODUCTION

Massively multiplayer online games (MMOGs) are the theme of this study plan. Usually a client-server infrastructure is used to support this kind of games. This plan proposes the investigation and definition of a novel distribution model to be used for MMOGs, reducing the maintenance cost of the game server. In the next sections, the state of the art regarding this area will be presented.

1.1 Multiplayer games

In the past few decades, since the creation of the personal computer, computer games have enjoyed an evergrowing popularity. After the acess to the Internet became widely available to the general public, with increasingly fast and cheap domestic connections, multiplayer online games have become an important market. To illustrate this, a study from 2006 estimated that the online games market would grow from US$ 3.4 billion in 2005 to US$ 13 billion in 2011 (DFC Intelligence, 2006). The main characteristic of the MMOGs is the large number of concurrent players interacting with one another through the Internet. This kind of game may be divided into several categories, such as:

- FPS: *first-person shooting*, in which each player controls a character who has one or more weapons, with which he fights enemies that, in turn, may be controlled by other players or by artificial intelligence. These games are characterized by fast action, including shooting, moving, dodging etc.;
- RTS: *real-time strategy*, where each player commands an army, similarly to a chess game, except that the game play is not divided in turns, i.e. it's a real-time game and the player who thinks faster than the others may have an advantage. Nevertheless, the action is usually slower than in an FPS game;
- RPG: *role-playing game*. In this kind of game, each player has a characters who evolves with time, acquiring power and accumulating wealth. These games are characterized by not having a predetermined end of the "match", creating a virtual world with persistent state. The player may, then, connect to this world, play for some time with his character, disconnect and, later, reconnect and continue from the point he was when he disconnected for the last time, for his state is stored in the game server.

As examples of well-known online multiplayer games, we may cite *Quake* (id Software, 1996) as a classic FPS game; *Starcraft: Brood War* (BLIZZARD, 1998) a classic RTS game still played nowadays and Diablo II (BLIZZARD, 2000) as an RPG example.

1.2 Massively multiplayer online games (MMOGs)

After single-player computer games evolved to multiplayer online games with the advent of the Internet, MMOGs emerged as a new trend because of the decrease of the domestic Internet connection cost and the increase of its average bandwidth. This kind of games have become very popular in the latest years because they allow the online interaction of a massive number of players at the same time. In the most succesful cases, such as World of Warcraft (BLIZZARD, 2004), Lineage II (NCSOFT, 2003) and EVE Online (CCP, 2003), for example, tens of thousands of players are supported simultaneously (CHEN; MUNTZ, 2006). These players can, then, interact with one another and with the virtual environment of the game.

One of the main features of these games, present in almost all commercial MMOGs, is that the matches are very long. In some MMORTS (real-time strategy MMOGs), such as Travian (TRAVIAN, 2004), the games come to last from six months to one year. MMORPGs (MMOGs in RPG style), as is the case of World of Warcraft, the games have no given time to end. The worlds of these games are so vast and populous – as regards the number of players – that they may evolve and modify their state independently of some players being playing it or not, because there is always someone connected so the game never stops. For this reason, it is expected that each player can disconnect and continue playing later, without having to start from scratch. Moreover, it is expected that changes made on the game world persist, no matter how many times each player enters and leaves.

Generally, each player controls an entity called *avatar*, which is nothing more than his representation in the virtual environment – a character who executes the orders given by the player, therefore interfering in the history and in the outcome of the game. For the game to remain consistent for the different participants, each action performed by each avatar must be processed by the server, which calculates the game state resulting from such action. This new state is then broadcast to other players. Other changes in the game world, such as destruction or creation of objects in the environment, and events such as actions of characters controlled by the server must also be sent to the players involved. It is then through these messages sent by the player to the server and in the opposite direction that players can interact with each other and with the simulated world of the game.

However, it is perceived that such a mechanism can easily saturate the bandwidth and overload the cpu(s) of the server. What is usually done is build an infrastructure with a large central server, usually a cluster with its nodes connected through a local high-speed and low latency network, with an Internet connection of some hundreds or thousands of MBps. The greatest cost comes from maintaining this connection, whose main purpose is to maintain every player as up-to-date as possible about the game state – naturally, tolerating some delay, which should not be over a given threshold.

2 MOTIVATION AND STATE OF THE ART

As already mentioned, MMOGs are characterized by a large number of players. To support this great number of participants, a client-server infrastructure is generally used, and all interaction is done through the server machine. In this paradigm, the server computer has the official copy of the game state and it is responsible for computing its change over time, while the game client is responsible for presenting to the user the game state sent by the server. The server is also responsible for updating the clients, in real time, on the changes that occur in the virtual environment of the game. Because of the high number of participants, the centralization on the game server makes the maintanance cost very high. Generally, it is required that the MMOG server has a connection to the Internet with several GBps of bandwidth and enough processing power to run the simulation of the virtual world (FENG, 2007). It becomes impossible, for example, the maintenance of massively multiplayer online games by small or independent groups on a limited budget.

Because of this high cost, alternative solutions have been sought in order to distribute somehow the communication and processing loads originated from the game. Generally, it is proposed the use of peer-to-peer networks – or even computer GRIDS (IBM, 2003; WANG; WANG; LAU, 2004). However, an MMOG is a simulation of a dynamic virtual environment, with a maximum tolerable delay between sending the player's actions and receiving the resulting state. MMOGs have also other requirements that must be observed carefully so that the players may have a good game experience. For example, as in a fully decentralized peer-to-peer MMOG the peers may have to calculate themselves the outcome of the players' actions, a malicious player could try to hack the software running in his computer in order to forge game states that provide him unfair advantages over the other players. To distribute the network support structure among different machines connected via the Internet – which is the case of the peer-to-peer networks – while meeting the requirements of an MMOG becomes a challenging task that has motivated many researchers in recent years.

In the following sections, we introduce some requirements of the MMOGs in addition to the state of the art as regards the distribution of the infrastructure to support this kind of game. Finally, it will be presented some research questions that are still open, and which can be addressed in a doctoral project.

2.1 Requirements of the MMOGs

Based on the current state of the art and in previous work – such as in (BEZERRA; GEYER, 2009a; BEZERRA; CECIN; GEYER, 2008; VILANOVA et al., 2008; CECIN et al., 2006, 2004) –, we can summarize a few requirements that we find critical for the game network support infrastructure to be considered adequate, and which will be dealt

with in the doctoral course:

- **Persistence** - MMOG games generally consist of a virtual world where players interact with each other and with the environment, changing the state of the world and of their own avatars. It is expected that each player can leave the game and return some time later without these changes having been undone. For this reason, it is necessary to support the persistence of the states of each avatar and of the virtual world as a whole;

- **Availability** - In general, MMOGs require that the players make some kind of payment for account maintenance and, because of that, they tend to be demanding when it comes to availability. Thus, it is desired the system is always available;

- **Consistency** - Multiplayer games consist of a virtual environment shared among the players participating in the same match. It is therefore necessary that the environment is represented in the same way on the machines of the various players so that they can interact with each other and with the world in an appropriate manner;

- **Security** - In a system where there are tens of thousands of participants, it is unreasonable to rely on all of them, requiring some technique to provide resistance against faulty or malicious behaviour. Security here refers not only to the authentication of registered users, but also to check whether the states or actions that the different players submit are valid. Cheaters, for example, can change the software running on their machines so it may send states or actions that give them an advantage in the game;

- **Scalability** - Taking into account the number of participants in an MMOG, it is necessary that the system that supports it is able to maintain the quality of service despite the increasing processing and communication load required for this.

2.2 Related work

The **client-server architecture** meets various aspects necessary for the satisfactory implementation of games such as MMOGs, what includes a high level of control over the system as a whole, which facilitates authentication, persistence and security. But the maintenance of a traditional MMOG server is expensive, as already mentioned, besides making the server a possible bottleneck. In order to minimize this problem some alternatives have been proposed. One of them is to use cluster computing, where a cluster, rather than a single computer, plays the role of the server. This approach has a significant increase in processing power, but does not solve all the problems of massively multiplayer online games. It should also provide the bandwidth needed to support the traffic between the server and the players.

Another approach is the **peer-to-peer architecture**, where the simulation is divided among the computers involved. There can be a system without any server, where the peers, which are the machines of the players, come into some kind of agreement for the various steps of the simulation, calculating the game state resulting of each action of the participants. Regarding scalability, this approach is not optimal, since ensuring such "agreement" is costly in terms of message exchange (LAMPORT; SHOSTAK; PEASE, 1982). Even if one of the peers is elected to decide the course of the simulation, still

persists the problem of all peers potentially needing to exchange messages with one another. Having n peers, there is a complexity of O(n^2) message exchanges for each step of the simulation, in the worst case. Clearly, this approach is not as scalable as should be a system to support a massively multiplayer online game. Moreover, it would be necessary to distribute the storage and retrieval of the states of the game.

Some studies – such as (SCHIELE et al., 2007; RIECHE et al., 2007; HAMPEL; BOPP; HINN, 2006; EL RHALIBI; MERABTI, 2005; IIMURA; HAZEYAMA; KADOBAYASHI, 2004; KNUTSSON et al., 2004) – have been done to make games on peer-to-peer networks more scalable. For example, to reduce traffic between the peers, each peer can send state updates only to whom the state changes are relevant. To achieve this goal, (SCHIELE et al., 2007) suggests the division of the virtual environment into regions. The purpose of this is for each region to act as an independent set of smaller scale. Players whose avatars were in a given region could form a small peer-to-peer group and their own computers would decide the outcome of the players' actions in that area of the virtual environment. When an avatar moved from one region to another, it would leave the first group and join the group of its new region. Since the environment is considered contiguous, i.e. the avatars could move freely across adjacent regions, some of the peers in a group would keep connections to peers from groups of neighboring regions and, through these connections between adjacent groups, the necessary information for transferring a player could be obtained.

Still according to the work of (SCHIELE et al., 2007), each region has a coordinator. The coordinator is elected among the peers which were present in the region and becomes responsible, only, to decide to whom each state update is relevant – it does *not* intermediate message exchanges between peers, whose communication is direct. However, this approach is based on the fact that the coordinator is reliable, which can not be guaranteed, since the software used by the player may have been modified to behave incorrectly in order to provide invalid benefits – what is also known as *cheating*. Another approach would be to elect multiple coordinators per region, but that would involve the implementation of a voting mechanism, and depend on the availability of a certain number of peers to manage the region. Moreover, the communication load remains quadratic (in the worst case), since every peer must send its state to several other peers.

There are also proposals for **hybrid architectures** (VILANOVA et al., 2008; CHEN; MUNTZ, 2006), using servers and peer-to-peer networks at the same time. According to these works, peers and server share the simulation of the game world. In (VILANOVA et al., 2008), the game world is divided into social space and action spaces. The first is aimed at social interactions, such as chat, exchange of in-game items, forming groups, etc., which are actions that do not impose a heavy burden on the server. However, if the players want, for example, fight – what is the main objective in most MMOGs – they have to request to the server the creation of an action space within which the interaction is faster and with little tolerance for communication delays. This space is an *instance* of the game, isolated from the rest of the game world, within which a limited number of players form a small peer-to-peer group, being themselves responsible for performing the simulation and for updating his companions on the game state. To resolve inconsistencies in the simulation due to lack of message ordering, one player is chosen as a *super-peer* and is responsible for ordering the events that are "heavily coupled", whose order can significantly interfere in the outcome of the game. This approach presents some problems, such as the fact that there cannot be a fight, for example, between a large number of players. Moreover, nothing guarantees that the peer chosen to be the super-peer is trustable. If it

belongs to a dishonest player, it may put the events in an order that provides an invalid advantage over the other players. In addition, the problem of overloading the bandwidth of peers is addressed by simply setting a limit to the number of participants in each action group, when in fact it is desirable that in an MMOGs there is a lot of players interacting with each other, not only in social circumstances, but also in more dynamic situations, like fast-paced fights.

In (CHEN; MUNTZ, 2006), the virtual environment is divided into regions, each managed by a peer, which acts as a sub-server. Each one of the other participants in the region sends state updates to this sub-server, which forwards the messages to the other peers in the region to which the state changes may be relevant. To provide fault tolerance, the authors suggest the use of peer clusters, consisting of the region's sub-server and backup peers. These backup peers receive the players' actions from the region manager and process them, creating a backup of the game state. In addition to providing robustness to the system in case the manager crashes, these backup nodes may also detect faults in the simulation and perform recovery procedures. However, this approach does not solve the problem of the peer elected to be the region manager not being reliable: to prevent the backup nodes from detecting an invalid simulation, the region manager may send invalid actions to them, instead of forwarding the players' real actions. Finally, if a region manager gets overloaded, it may simply return to the server part of the load assigned to it, keeping a dependence on a centralized infrastructure.

Another hybrid architecture is that of FreeMMG (CECIN et al., 2004). In it, the players are organized in a peer-to-peer manner in every region of the virtual environment and the server mediates the communication between different regions. Again, there is the problem of security: as the peers within a region control the simulation that occurs there, they can also subvert it. The author proposes a probabilistic approach, using a randomly selected peer to verify the simulation in that region. It is expected that, if the peers in a given region try to subvert the game there, at least the randomly inserted node detects the invalid actions and reports to the server. However, besides not completely guaranteeing security, persists the problem that there may be many peers communicating with many, which may compromise the quality of the game.

Some models that suggest the use of clients connected to a distributed server have been proposed in the literature. In (ASSIOTIS; TZANOV, 2006), for example, it is proposed to divide the virtual world into smaller regions, each assigned to a different node of the server system. The authors address the consistency problem by using a lock mechanism – each entity in the game is associated with a lock that must be obtained before any of the servers making any changes. In addition, they tried to address the problem of *hotspots* (clusters of players in a relatively small area of the virtual environment) through the recursive partitioning of the overpopulated area until the overload is eliminated or there are no more servers available. However, there is a practical limit to this partitioning, for very small areas cause avatars to migrate between regions more often, which implies more network traffic between servers.

Finally, the authors of (ASSIOTIS; TZANOV, 2006), as well as several others with similar proposals (NG et al., 2002; CHERTOV; FAHMY, 2006; LEE; LEE, 2003), consider that the nodes are servers connected through a high speed and low latency network, what makes their solutions only partially applicable in a scenario with highly dynamic and volatile resources, such as the peer-to-peer networks formed with volunteer resources. Such networks have some inherent problems: nodes with low availability and dependability, low bandwidth and low processing power compared to current dedicated MMOG

servers maintained by large game companies.

2.3 Research questions

In a completely decentralized MMOG deployed on a peer-to-peer network, each peer is responsible for deciding the outcome of the actions of its player. Although this makes the system less dependent on a single point of failure – if a player is disconnected from the game, only the state regarding its avatar will be lost – and to make the game independent of central servers, this approach proved to be very vulnerable to cheating precisely because at least part of the simulation is relied on the peers. Besides, this approach is prone to saturation of the upload bandwidth of the participants, who must send state updates to their fellow players.

This doctorate plan considers the use of a server system consisting of geographically distributed volunteer nodes. This system would be heterogeneous, where each node could be anything from a home computer to a cluster belonging to some company that had interest in serving the game. Some minimum processing power and communication bandwidth would be required for being part of the server, and it would be necessary to ensure somehow that each node is trusted. However, taking into account that the number of server nodes would be much smaller than the number of players, it would be more feasible to ensure its reliability regarding the possibility of cheating than in a fully decentralized peer-to-peer architecture. Furthermore, the server nodes do not necessarily belong to players. They could be small local Internet service providers (ISPs), for example, who would profit by putting their resources available for a game. That would be interesting to these companies for the resources – at least communication bandwidth – of the ISP would be used anyway, for all the traffic from players who were also its subscribers would pass over there regardless of which server each player was connected to.

Nevertheless, as the simulation would be distributed among the server nodes, it is not necessary that all they have the same processing power or communication bandwidth owned by the huge central servers commonly used for MMOGs. To perform this distribution, the game world could be divided into regions, each managed by at least one of these server nodes. More than one server node should be assigned to each region, so that the simulation would be distributed on two levels: first into regions, then within each one of them among the nodes managing it. Thus, each player would send his actions to one of these nodes, which would process them, then synchronize with other nodes in the same region and, finally, make the resulting game state persistent in a distributed way and send it to the players to whom it is relevant. Thus, it would also be provided a better availability, as each region of the virtual world would be served by a set of servers holding a synchronized game state. If any of these servers fail, one of the others could take on the players who were connected to the server that is no longer available.

To achieve the goal of distributing an MMOG server on a dynamic network, with few and volatile resources and less dependability, as are the peer-to-peer networks formed by volunteer nodes, some questions that either have been the subject of research in recent years or that are still unresolved arise. Some of them are:

- Assign to each server node a portion of the virtual world, so that the load is dynamically balanced, taking into account the existence of *hotspots* (clusters of avatars in a relatively small area);

- Maintain the simulation consistent among the different regions and among servers

managing the same region;

- Coordinate the entry, exit and the crash detection of nodes of the system, providing fault tolerance;
- Distribute and store in a persistent manner the state of the world and of the players, so that they can be retrieved when necessary;
- Optimize the communication between clients and servers in order to save the maximum possible bandwidth, without this being perceptible by the players.

It is intended, during the doctoral course, to research and develop new solutions to these problems, seeking those that are most appropriate in a context of massively multiplayer online games. With these issues resolved, it may be provided to MMOGs availability and fault tolerance – due to the replication of server nodes in the same region – and scalability, while maintaining the communication delay within an acceptable range for the game, all this for a low cost for the player, compared to traditional MMOG servers.

These solutions will be wrapped together in a middleware, which will provide them as services. These services will be accessible via an API (application programming interface) to allow an easier development of a massively multiplayer game. The game programmer will be able to use these services without needing to have a deep understanding of the distribution model used by the middleware. To the application developer, a high-level interface will be provided, while the middleware will take care of the tasks performed in lower levels. The communication abstraction provided to the game developer may be: publish-subscribe, as in PADRES (FIDLER et al., 2005), distributed shared memory, as in Sinfonia (AGUILERA et al., 2007), message passing or any other that best suits the development of an MMOG.

3 OBJECTIVE

The main objective of this doctoral plan is to create a decentralized architectural model to support massively multiplayer online games. Following this model, it should be possible to form a system of geographically distributed server nodes, using the Internet as a means of connection between them. This distribution should be transparent to the players, to whom the system will be seen as a single MMOG server, which maintains the game world consistent, available and with a good quality of service. As a final product, a middleware for MMOG development with a high-level interface will be created.

3.1 Detailed objectives

To achieve this general objective, some goals were established:

1. Develop a mechanism to deal with entry, exit and crash of server nodes, providing fault tolerance, while trying to keep the nodes which are geographically close to each other adjacent in the overlay network of the server system;

2. Create an algorithm for balancing the load of the game, maintaining the load on each server proportional to its capacity, while avoiding the splitting of a hotspot into different regions, as this would cause a strong dependence between these regions (BEZERRA; GEYER, 2009a);

3. Define a scheme for synchronizing the simulation state among different server nodes assigned to the same region of the gaming world, as well as synchronize periodically, or when necessary, the states in different regions, in order to maintain global consistency;

4. Build a distributed persistence mechanism to store the states of the players and of the virtual game world as a whole, so they are always available for any server node, even if some of these become unexpectedly unavailable;

5. Design techniques and algorithms that aim to reduce inter-server and server-client communication, in order to save resources of the server nodes, reduce delays in the communication between them and expand the audience of the game due to the reduction in the bandwidth required to play it and, besides, reducing the requirements for a computer to be part of the server system, without undermining the game experience of the players;

6. Create a middleware containing the desired services, along with a high-level API to make them accessible to the game developer.

4 PRELIMINARY MODEL: COSMMUS

In a previous work (BEZERRA; GEYER, 2009b), it was set up a preliminary model – which aimed to partition the virtual environment into regions, each assigned to a single server – on which a load balancing scheme was defined. The Cosmmus – Cooperative System for Massive Multiplayer Service – is a new model to be defined during the doctorate course, which will also uses the principle of partitioning the virtual environment of the game into regions, but addressing the most critical problems (as referred in section 2.3), which were left out so far.

Cosmmus should include solutions to the major aspects of MMOGs, such as task distribution, synchronization of the simulation, distributed persistence and so on. Although this model is not defined yet, there are already some ideas on how it could be and what kind of approach it should use, based on related research. In the following sections it will be presented these initial ideas, which will be better developed during the doctoral course.

4.1 Load distribution

Some studies have been done concerning the partitioning of the virtual environment of the game, as (DE VLEESCHAUWER et al., 2005; MORILLO; FERNÁNDEZ; ORDUÑA, 2006; LUI; CHAN, 2002). For Cosmmus, it is intended to make the load distribution among the game servers in two levels:

1. It will be used the idea of regions (portions of the virtual world), whose size will be adjusted dynamically depending on how the avatars of the players are more or less concentrated in certain places. This should be done transparently to the player, who will see a large contiguous world through which he can move freely;

2. Each region can be assigned to more than one server node, with two main objectives: first, to provide fault tolerance through replication and, secondly, to allow regions that are already heavily used to be managed by more than one server node.

If a region get heavily crowded, one could think about splitting it in two or more new regions, but this could be counterproductive in certain situations, as in the case of a region that contains a hotspot: its division would result in two regions overly dependent on one another. Because of this kind of situation, it should be also created an algorithm, heuristic or metric to determine whether a region should or should not be subdivided into new regions.

It has been proposed, during the master course, a dynamic load balancing scheme based on graph partitioning and refinement of its partitions (KERNIGHAN; LIN, 1970; KARYPIS et al., 1999; KARYPIS; KUMAR, 1998; DUTT, 1993). The world was divided

into cells of equal size and fixed position, each represented by a vertex in the graph. The interaction (measured in kilobytes per second) between the cells are represented by the edges of the vertices. The algorithm consists of partitioning the graph, where each partition represents a region to be assigned to a server. This balance takes into account different levels of resources among different servers and takes into account the needs of MMOGs, considering the bandwidth usage as the load to be balanced among the server nodes.

It will also be created a novel load balancing scheme which is intended to follow a few principles of the scheme proposed in the master's thesis. However, unlike the model suggested previously, there will be not only one server node per region, but a group of server nodes. In this case, it will be investigated ways to balance the load of the system, considering the distribution in two levels. Besides, other environment partitioning techniques could be used instead of using cell grids, such as space partitioning trees, as hinted in (BEZERRA; COMBA; GEYER, 2009).

4.2 Formation of the server system overlay network

Regarding the formation of the overlay network on which the server system will be deployed, there are some proposals, such as in (TA et al., 2008; RATNASAMY et al., 2002; HAN; WATSON; JAHANIAN, 2005; ZHANG et al., 2004), which try to build a network so that nodes with good network connection between them are adjacent in the overlay topology. It was proposed, for example, grouping the nodes according to the network delay between them. The goal is to use these techniques so that certain server nodes – nodes which are serving adjacent regions and, especially, serving the same region – belong to the same group. Adaptations are possible, including criteria other than delay, as the data transmission rate between them, in bits per second.

The purpose of mounting the overlay network following this strategy would be to minimize the delay in interaction between players. This delay in the interaction would be the time elapsed since the sending of an action from a player, processing by the server and receiving the resulting state by another player. If any two players are interacting with each other, but each one of them connected to a different server node, it will be required that each of their interactions is mediated by both servers. So, if the connection between these two servers has a low delay, the interaction of the players will be less harmed by the addition of a second intermediary.

4.3 State synchronization

One of the key issues that will be handled by Cosmmus is maintaining the consistency of the simulation among the different server nodes. Even if each player sees just a part of the virtual world, this world must be the same for all servers. Thus, if two players are with their avatars in the same place of the virtual environment, they should see the same scene. Another important issue is the ordering of events. Assuming that two players are fighting in the game, the avatar of one of them shoots with a gun and the other avatar moves, the order of these events would be crucial to decide whether the latter player had his avatar dead, only wounded or completely avoided the bullet.

In the event that both players are connected to the same server, as it is traditionally done by game developing companies, this task would be trivial, by simply checking which actions come first. Even if a mechanism for delay compensation – the *timestamp* of the

action of each player would be equal to the current time minus the player's connection latency – as it is done in Half-Life (VALVE, 1998), for example, it would not be too problematic to sort these events. In the case of a distributed server, it would require the server nodes to enter into some kind of agreement or synchronization regarding this event ordering, whose complexity would grow with the increase of the number of servers involved.

Taking into consideration that there are several regions forming a contiguous world, and that in each one of them there is one or more servers, these issues to maintain the consistency of the simulation becomes even more difficult. It is expected that the division of the environment is seamless to the player. Then, when approaching a region boundary, the player should be able to see whatever is beyond it and to move freely across the border. For this, it will be necessary that the server to which he is connected to communicate with the one that serves the region beyond the border which is being viewed by the player. It will provide information such as the state of the environment and of avatars who are nearby and of other objects in the game.

In (CECIN et al., 2006), it was proposed a state synchronization scheme that consists of two simulators: one conservative and one optimistic. The optimistic one processes each player action immediately after it is received and its resulting state becomes available to the player and something is show on his screen. The conservative simulator, however, divides the time in turns and only performs the actions of the players after performing a synchronization. At each turn, it waits for the arrival of the actions of all players involved – or some explicit notification that they did nothing – which ared then sorted and processed. Only after this step the conservative simulator advances to the next turn. In the case of the optimistic simulation deviate beyond a tolerable limit from the conservative one, the latter overwrites its state on the first, which continues from there. However, this scheme was proposed for a fully decentralized architecture, being necessary to analyze it and make necessary adjustments so it can be used in a server system distributed in two levels, as intended for Cosmmus. There are also other works that aim to maintain a consistent simulation, such as (ZHOU; SHEN, 2007; MOON et al., 2006; MÜLLER; GÖSSLING; GORLATCH, 2006), which will also be studied and possibly some of its principles will be part of the solution for consistency designed for the final model.

4.4 Distributed persistence

Regarding the storage of the state of the virtual world and the players, it will be devised a way to do this in a distributed manner, since there will not be an unique central server. It might be simpler to choose one of the server nodes to save this information. However, this could cause the game state to depended on the availability and reliability of the chosen node and, even if these two conditions were satisfied, the node's connection could be easily saturated by the great number of requests, slowing down the recovery of the states stored in it.

Cosmmus should then use a distributed storage scheme, where portions of the state information are stored in different places. Furthermore, it should replication, so that the state becomes available even if some nodes of the system disconnect or crash unexpectedly. DHT-based overlay networks, such as Pastry (ROWSTRON; DRUSCHEL, 2001), Tapestry (ZHAO et al., 2004), Chord (STOICA et al., 2001) and Koord (KAASHOEK; KARGER, 2003) seem to be possible solutions to provide the kind of distributed and redundant storage desired. It is intended, therefore, to investigate how suitable these net-

works really are for this purpose. There are simulators, such as OverSim (BAUMGART; HEEP; KRAUSE, 2007), allowing the simulation of overlay networks, such as those mentioned, and through simulation it be will assessed the behavior of the system when using some of these logical networks.

A problem, however, is that the computational cost of using those DHT-based networks may become too high, for it would be necessary to maintain the structure of a network such as Pastry on the server system, retrieving and storing the game state with application level multicast and replication, added to the own cost that the server nodes handle to perform the game world simulation itself, receiving players' actions and returning them a resulting state. To avoid overloading the system with traffic related only to persistence of the game state, some parameters could be adjusted, like the state update frequency and, at the application level, the level of detail of the state to be persisted. For example, instead of storing the location, direction and action being performed by each avatar in an area of the virtual environment, it could be stored only its location.

4.5 Bandwidth usage optimization

As the server nodes are, ideally, of low cost, it is necessary to save their resources as much as possible, but without harming the game quality. For example, it can be made some interest-based filtering, being sent to each player only what he can see, in order to save as much bandwidth between clients and servers, and between the servers. Following this same principle, servers from two different regions may have an old version of the state of each other, if there is no interaction between players whose avatars are in different regions. Thus, these servers do not need to update each other so often.

Periodically, or when it is needed (e.g. two players from different regions interacting near the border), the states of the regions may be exchanged between the servers managing them. The granularity of this information can be even thinner, and each server could send to the other only the most essential data. For example, each server would send to the other only the state of the virtual environment near the border between the regions, rather than the state of the entire region.

Several works have been made in order to make filtering interest-based message filtering (BEZERRA; CECIN; GEYER, 2008; AHMED; SHIRMOHAMMADI, 2008; MORGAN; LU; STOREY, 2005; MINSON; THEODOROPOULOS, 2005). In addition to studying which one best fits the proposed model, and evaluate the possible adjustments that are necessary, other ways to save the resources of the server system will be investigated.

4.6 High-level application programming interface

In order to offer to application (game) developers an easy-to-use set of services, a high-level application programming interface must be created for Cosmmus. This set of services will compose the Cosmmus middleware, which will encapsulate all the low level operations regarding the distribution of the game. There are some options to define this distribution model – and, also, the abstraction provided to the game developer –, like distributed shared memory, publish/subscribe, message passing and others. Each one of these presents both advantages and disadvantages, but the publish/subscribe seems to be the most suited for the development of MMOGs, since the entities (avatars and objects) present in the game's virtual world imply state update messages, in which every player

may be interested or not.

4.7 Philosophical questions

Although generally it is needed a powerful, centralized and expensive (FENG, 2007) infrastructure to support an MMOG, a system that follows the approach outlined here would allow the resources of smaller businesses and home users (with the required minimum of processing power and connection bandwidth, which will depend on the particular game) to be used in order to form a system to serve a massively multiplayer online game with quality comparable to the most popular MMOGs.

The fundamental idea behind Cosmmus is to form a server system on a peer-to-peer network, rather than indiscriminately distributing the simulation among all players, as are several proposals surveyed. With such separation, where only the server nodes could decide the progress of the game, it could be formed communities whose sole purpose would be to keep MMOGs up and running, organizing who would be allowed to participate in the system as a server node, with which permissions, and so on. There could be a scheme similar to that used in various free software projects, in Wikipedia (ADLER; ALFARO, 2007) and others, where the individual – or company – who had the initial idea gradually delegates authority to community members, according to how much each one cooperated, creating a hierarchy, with different levels of permission, but with all participating in a common effort.

In addition to this philosophical question, to an MMOG server, which acts as referee and where the actual simulation is performed, it is easier to detect and prevent cheating than in an MMOG with a completely decentralized simulation, where each player's computer decides in part what will happen as a result of players' actions in the game. However, even in a collaborative network as the one idealized in this study plan, there is no guarantee that some of the members will not try to subvert the rules of the game, providing unfair advantages to some players, or even just destroying part of the game state. However, a reputation system is imagined. When entering the server system, each participant will have a record, stored in a trusted location, of how he has performed the simulation of the game.

This reputation system would work as follows: at first, it would be assigned to the new server a portion of the world which is considered of minimal importance, so that attempts to subvert that part of the game environment will cause very little damage to the game, if any. Both players and servers may report such cases in order to punish or even banish the node that acted maliciously. Honest members of the server system will gain reputation points with time, and they will manage increasingly important portions of the game world. It is believed to be enough demotivating the idea of destroying a good reputation, gained after a long period of honest collaboration with the game because of an cheating attempt.

This way, many MMOGs could arise and become popular more easily, opening this market to small companies and independent game developers. Every game would be associated with a self-regulating community. The level of justice or tampering of each one of them would depend on who were responsible. A well organized community could make a game as fair as those managed by large companies with expensive and powerful central servers, for example.

5 WORKPLAN AND SCHEDULE

This section describes the workplan, classes to attend and the intended schedule to perform the activities described in the previous sections. Section 5.1 describes the activities to execute and the intermediate goals to fulfill the study plan; section 5.2 lists the classes to attend and section 5.3 describes shortly the intended international cooperation in Lugano.

5.1 Methodology

The steps and intermediate goals to achieve in the doctorate course are as follows:

1. **Doctoral course requirements**, as imposed, by the postgraduate program in computer science of UFRGS, to the PhD candidate:

   (a) Validation of the 24 credits obtained during the Master;

   (b) Obtainment of the remaining credits required for the doctorate course;

   (c) Realization of teaching activity;

   (d) Presentation of the research stated during the academic week of UFRGS;

   (e) Obtaining of approval in a broad knowledge examination;

   (f) Presentation of the thesis proposal.

2. **Literature review**: aims to update and expand the domain of the state of the art related to the distribution of MMOGs:

   (a) Study overlay networks formation technology and check its compatibility with the intended type of management of entry, exit and crash of server nodes;

   (b) Search for papers offering techniques aimed at allocating the servers on the network overlay, taking into account the quality of communication between them (i.e. servers geographically close – and with a good connection between them – tend to be adjacent in the overlay network);

   (c) Analysis of load balancing algorithms, with regard to its applicability on a distributed server for MMOGs, in order to determine what adaptations are needed to the context of games;

   (d) Search works that are intended for the discovery and management of hotspots, so that their presence does not degrade the quality of service of the game;

(e) Review of techniques to synchronize the state of the game among different servers dynamically, without generating excessive traffic on the server system;

(f) Search mechanisms for distributed, consistent and fault-tolerant persistent, which can be used to store the states of the players and of the virtual environment;

(g) Study on existing techniques to reduce the traffic between clients and servers and between servers, so as to reduce costs without harming the game quality.

3. **Model definition**: aims to define the components necessary for the architecture to be able to provide the desired service:

   (a) Definition of the kind of topology that the overlay network should have and what technology will be used/adapted for the management of this network;

   (b) Definition of criteria used to perform load balancing and how it will be done;

   (c) Definition of the synchronization scheme among the server nodes;

   (d) Definition of the persistence model used to store in a distributed and fault-tolerant manner the state of the virtual world and the players;

   (e) Integration of the model defined with techniques for optimizing the use of bandwidth and reducing delay in communications;

   (f) International cooperation with another research group, via sandwich or co-tuition (more details in section 5.3).

4. **Implementation of the model**, which is aimed to create a prototype that can be used both in simulations and in actual implementations of the proposed model:

   (a) Implementation of the layer for managing the network connections between node servers and between servers and clients;

   (b) Implementation of the module responsible for load distribution and hotspots management;

   (c) Implementation layer of simulation synchronization and state persistence;

   (d) Creation and integration of module responsible for monitoring the node's resource usage and adapting the quality of the game to the capacity of each server.

5. **Simulation and tests**, whose objective is to validate the proposed model and verify whether the defined objectives have been attained:

   (a) Analysis and comparison of network and large-scale systems simulators, one of which shall be chosen to perform the simulations of the proposed model to support distributed MMOGs;

   (b) Adaptation of the prototype to run on the chosen simulator;

   (c) Performing of tests with respect to: quality of communication between adjacent nodes; efficiency of load balancing and hotspots detection; consistency of the simulation among different server nodes; availability of the game and of the states of the players, even in the presence of faults;

(d) Test the prototype on a real large scale system, such as Grid5000 (BOLZE et al., 2006) and analyze the results.

6. **Thesis' writing**: documentation of the work as a PhD thesis, and publish it, among with conference and journal papers.

   (a) Thesis writing;

   (b) Writing of articles for national (SBGames) and international (DS-RT, NetGames) publication, besides international journals, such as *Multimedia Tools and Applications*, for example.

5.2 Courses to attend and credits to obtain

In order to fulfill the requirements of the doctorate course in UFRGS, it is intended to:

1. Validate the 24 credits obtained during the Master course in Computer Science, in UFRGS. The attended courses are listed in table 5.1;

2. Obtain 12 additional credits, by attending the courses CMP134, CMP189 and CMP112;

3. Obtain 3 more credits, corresponding to a written work and the realization of teaching activity (Table 5.2).

Table 5.1: Credits, from Master, to validate

Code	Subject	Credits	Grade
CMP139	Fault-tolerance in distributed systems	4	A
CMP255	Parallel programming for HPC	4	A
CMP167	Distributed objects programming	4	A
CMP256	Design of parallel programs for HPC	4	A
CMP157	Parallel and distributed programming	4	A
CMP250	Conceptual aspects of fault-tolerance	1	B
CMP401	Written work I	2	A
CMP410	Teaching activity I	1	A

Table 5.2: Credits obtained during the doctorate course

Code	Subject	Credits
CMP112	Distributed information systems	4
CMP134	Introduction to parallel and distributed processing	4
CMP189	Geometric algorithms	4
CMP402	Written work II	2
CMP411	Teaching activity II	1

5.3 International cooperation

There has been several contacts with professor Fernando Pedone, from University of Lugano (USI), Switzerland, who works with distributed systems and distributed data management, with the possibility of an internship abroad at the Swiss university, for what a great interest has been demonstrated by the researchers from both institutions (UFRGS and USI). The purpose of the period in a foreign university would be to exchange experiences between the research groups in order to better define a final model for Cosmmus, utilizing expertise of others researchers in the field of distributed simulation and, more specifically, MMOGs.

5.4 Schedule

Table 5.3 presents the schedule for the activities depicted in the previous sections, as listed in section 5.1.

Table 5.3: Work schedule

Task	2009/1	2009/2	2010/1	2010/2	2011/1	2011/2	2012/1	2012/2
1a	X							
1b	X	X						
1c		X						
1d			X		X		X	
1e		X						
1f				X				
2a	X							
2b	X							
2c	X							
2d	X							
2e		X						
2f		X						
2g		X	X					
3a			X					
3b		X						
3c			X					
3d			X					
3e			X					
3f				X	X			
4a				X				
4b				X				
4c					X			
4d					X			
5a					X			
5b					X			
5c						X		
5d						X	X	
6a							X	X
6b			X	X	X	X		

6 RESEARCH EXPERIENCE

Before entering the Master course, the candidate has already worked with research and development, in the independent group *indigente* (INDIGENTE, 2004), being one of the mentors and developers of the game engine InGE (ROCHA et al., 2007) and the sole programmer of its network module. Next, he was one of the developers of the game *We Are The Champignons*, which obtained the 3rd prize in a national indie games contest – the "Game Festival", part of the 5th Brazilian Symposium on Computer Games and Digital Entertainment, in 2006.

One of the objectives of the doctoral course is to continue the research started during the Master's course, during which a technique for bandwidth usage optimization have been proposed: the A^3 algorithm (BEZERRA; CECIN; GEYER, 2008), which consists of a fine-tuned interest management, based on different relevance levels for each pair of entities in the game. Another work done during the Master's was the definition of a distributed server load balancing scheme (BEZERRA; GEYER, 2009a) which takes into consideration several points that have been considered critical: first, the distributed MMOG server system is not necessarilyh homogeneous, requiring the load balancing scheme to consider the different amounts of resources; second, the players may move their avatars to certain positions with a higher probability, leading to the formation of densely populated areas (hotspots) in the virtual environment; and, finally, the algorithms proposed tried to minimize the inter-server communication by using a graph partitioning approach.

Additionally, the candidate participated of the P2PSE project (VILANOVA et al., 2008), as one of the developers responsible for the multi-server support in the library. The programming language used was C++. Besides, he set up and executed the network tests with the ns-2 simulator (MCCANNE; FLOYD et al., 2006), whose results have been considered satisfying.

For the doctoral course, the candidate intends to create a more detailed model to cover the requirements defined for the distributed server system, resulting in a MMOG network support middleware. During the research period in Switzerland, the solutions for distributed persistence and consistency management, at least, will be researched, along with the definition of a high-level application programming interface for the MMOG developers to use the services provided by the middleware seamlessly, minimizing their concern with distribution issues. Besides, fault-tolerance and further bandwidth usage optimizations will be provided by the intended middleware.

REFERENCES

ADLER, B.; ALFARO, L. de. **A Content-Driven Reputation System for the Wikipedia**. 2007. 261–270p. v.7.

AGUILERA, M.; MERCHANT, A.; SHAH, M.; VEITCH, A.; KARAMANOLIS, C. Sinfonia: a new paradigm for building scalable distributed systems. **ACM SIGOPS Operating Systems Review**, [S.l.], v.41, n.6, p.159–174, 2007.

AHMED, D. T.; SHIRMOHAMMADI, S. A Dynamic Area of Interest Management and Collaboration Model for P2P MMOGs. **Proceedings of the 12th IEEE International Symposium on Distributed Simulation and Real Time Applications, 2008, Vancouver, BC, Canada**, [S.l.], p.27–34, 2008.

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.

BAUMGART, I.; HEEP, B.; KRAUSE, S. **OverSim: a flexible overlay network simulation framework**. 2007. 79–84p.

BEZERRA, C. E. B.; CECIN, F. R.; GEYER, C. F. R. A3: a novel interest management algorithm for distributed simulations of mmogs. **Proceedings of the 12th IEEE International Symposium on Distributed Simulation and Real Time Applications, 2008, Vancouver, BC, Canada**, [S.l.], p.35–42, 2008.

BEZERRA, C. E. B.; COMBA, J. L. D.; GEYER, C. F. R. A fine granularity load balancing technique for MMOG servers using a kd-tree to partition the space. **VIII Brazilian Symposium on Computer Games and Digital Entertainment - Computing Track, 2009, Rio de Janeiro, RJ, Brazil**, [S.l.], 2009.

BEZERRA, C. E. B.; GEYER, C. F. R. A load balancing scheme for massively multiplayer online games. **Massively Multiuser Online Gaming Systems and Applications, Special Issue of Springer's Journal of Multimedia Tools and Applications**, [S.l.], 2009.

BEZERRA, C. E. B.; GEYER, C. F. R. **Lidando com Recursos Escassos e Heterogêneos em um Sistema Distribuído Atuando como Servidor de MMOG**. 2009. Dissertação (Mestrado em Ciência da Computação) — Universidade Federal do Rio Grande do Sul, Av. Bento Gonçalves, 9500, Porto Alegre, RS, Brazil.

BLIZZARD. **Starcraft**: brood war. http://www.blizzard.com/broodwar/.

BLIZZARD. **Diablo 2**. http://www.idsoftware.com/games/quake/.

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

BOLZE, R.; CAPPELLO, F.; CARON, E.; DAYDE, M.; DESPREZ, F.; JEANNOT, E.; JEGOU, Y.; LANTERI, S.; LEDUC, J.; MELAB, N. et al. Grid'5000: a large scale and highly reconfigurable experimental grid testbed. **International Journal of High Performance Computing Applications**, [S.l.], v.20, n.4, p.481, 2006.

CCP. **EVE Online**. http://www.eve-online.com/.

CECIN, F.; GEYER, C.; RABELLO, S.; BARBOSA, J. A peer-to-peer simulation technique for instanced massively multiplayer games. **Proceedings of the Tenth IEEE International Symposium on Distributed Simulation and Real-Time Applications (DS-RT'06)-Volume 00**, [S.l.], p.43–50, 2006.

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.

CHEN, A.; MUNTZ, R. Peer clustering: a hybrid approach to distributed virtual environments. **Proceedings of 5th ACM SIGCOMM workshop on Network and system support for games**, [S.l.], 2006.

CHERTOV, R.; FAHMY, S. Optimistic Load Balancing in a Distributed Virtual Environment. **Proceedings of the 16th ACM International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV)**, [S.l.], 2006.

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.

DFC Intelligence. Online Game Market forecasted to reach $13 billion by 2011. **Pressemitteilung**, [S.l.], v.6, p.2006, 2006.

DUTT, S. **New faster Kernighan-Lin-type graph-partitioning algorithms**. 1993. 370–377p.

EL RHALIBI, A.; MERABTI, M. Agents-based modeling for a peer-to-peer MMOG architecture. **Computers in Entertainment (CIE)**, [S.l.], v.3, n.2, p.3–3, 2005.

FENG, W. What's Next for Networked Games? **Sixth annual Workshop on Network and Systems Support for Games (NetGames)**, [S.l.], 2007.

FIDLER, E.; JACOBSEN, H.; LI, G.; MANKOVSKI, S. The PADRES distributed publish/subscribe system. **Feature Interactions in Telecommunications and Software Systems VIII**, [S.l.], p.12, 2005.

HAMPEL, T.; BOPP, T.; HINN, R. A peer-to-peer architecture for massive multiplayer online games. **Proceedings of 5th ACM SIGCOMM workshop on Network and system support for games**, [S.l.], 2006.

HAN, J.; WATSON, D.; JAHANIAN, F. **Topology aware overlay networks**. 2005. v.4.

IBM. **Butterfly.net: powering next-generation gaming with on-demand computing**. 2003.

id Software. **Quake**. http://www.idsoftware.com/games/quake/.

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.

INDIGENTE. **Interactive Digital Entertainment**. Grupo de pesquisa na área de jogos da UFBA, formado em abril de 2004. http://indigente.dcc.ufba.br/.

KAASHOEK, M.; KARGER, D. Koorde: a simple degree-optimal distributed hash table. **LECTURE NOTES IN COMPUTER SCIENCE**, [S.l.], p.98–107, 2003.

KARYPIS, G.; KUMAR, V. **Multilevel algorithms for multi-constraint graph partitioning**. 1998. 1–13p.

KARYPIS, G.; KUMAR, V.; CENTER, A. H. P. C. R.; MINNESOTA, U. of. Parallel multilevel k-way partitioning scheme for irregular graphs. **SIAM Review**, [S.l.], v.41, n.2, p.278–300, 1999.

KERNIGHAN, B.; LIN, S. An efficient heuristic procedure for partitioning graphs. **Bell System Technical Journal**, [S.l.], v.49, n.2, p.291–307, 1970.

KNUTSSON, B.; LU, H.; XU, W.; HOPKINS, B. Peer-to-peer support for massively multiplayer games. **IEEE Infocom**, [S.l.], 2004.

LAMPORT, L.; SHOSTAK, R.; PEASE, M. The Byzantine Generals Problem. **ACM Transactions on Programming Languages and Systems (TOPLAS)**, [S.l.], v.4, n.3, p.382–401, 1982.

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.

LUI, J.; CHAN, M. An Efficient Partitioning Algorithm for Distributed Virtual Environment Systems. **IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS**, [S.l.], p.193–211, 2002.

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.

MOON, K.; MUTHUKKUMARASAMY, V.; NGUYEN, A.; PAN, Z.; AYLETT, R.; DIENER, H.; JIN, X.; GOBEL, S.; LI, L. Efficiently Maintaining Consistency Using Tree-Based P2P Network System in Distributed Network Games. **LECTURE NOTES IN COMPUTER SCIENCE**, [S.l.], v.3942, p.648, 2006.

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.

MORILLO, P.; FERNÁNDEZ, M.; ORDUÑA, J. M. Solving the Partitioning Problem in Distributed Virtual Environment Systems Using Evolutive Algorithms. **Handbook of bioinspired algorithms**, [S.l.], p.531–554, 2006.

MÜLLER, J.; GÖSSLING, A.; GORLATCH, S. **On correctness of scalable multi-server state replication in online games**. 2006.

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

NG, B.; SI, A.; LAU, R.; LI, F. A multi-server architecture for distributed virtual walk-through. **Proceedings of the ACM symposium on Virtual reality software and technology**, [S.l.], p.163–170, 2002.

RATNASAMY, S.; HANDLEY, M.; KARP, R.; SHENKER, S. Topologicallyaware overlay construction and server selection. **Proceedings of IEEE INFOCOM'02, New York, NY, June 2002.**, [S.l.], 2002.

RIECHE, S.; FOUQUET, M.; NIEDERMAYER, H.; PETRAK, L.; WEHRLE, K.; CARLE, G. Peer-to-Peer-based Infrastructure Support for Massively Multiplayer Online Games. **Consumer Communications and Networking Conference, 2007. CCNC 2007. 2007 4th IEEE**, [S.l.], p.763–767, 2007.

ROCHA, R.; BESSA, A.; BEZERRA, C. E. B.; MEDEIROS, I.; OLIVEIRA, C.; BANDEIRA, H. Desenvolvendo um Motor Multiplataforma para Jogos 3D. **VI Brazilian Symposium on Computer Games and Digital Entertainment - Computing Track, 2008, Belo Horizonte, MG, Brazil**, [S.l.], 2007.

ROWSTRON, A.; DRUSCHEL, P. **Pastry: scalable, distributed object location and routing for large-scale peer-to-peer systems**. 2001.

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.

STOICA, I.; MORRIS, R.; KARGER, D.; KAASHOEK, M.; BALAKRISHNAN, H. **Chord: a scalable peer-to-peer lookup service for internet applications**. 2001. 149–160p.

TA, D.; ZHOU, S.; CAI, W.; TANG, X.; AYANI, R. Network-Aware Server Placement for Highly Interactive Distributed Virtual Environments. **Proceedings of the 12th IEEE International Symposium on Distributed Simulation and Real Time Applications, 2008, Vancouver, BC, Canada**, [S.l.], p.95–102, 2008.

TRAVIAN. **Travian**. http://www.travian.com/.

VALVE. **Half-Life**. http://www.half-life.com/.

VILANOVA, F. J.; BEZERRA, C. E. B.; CRIPPA, M. R.; CECIN, F. R.; GEYER, C. F. R. P2PSE - A Peer-to-Peer Support for Multiplayer Games. **VII Brazilian Symposium on Computer Games and Digital Entertainment - Computing Track, 2008, Belo Horizonte, MG, Brazil**, [S.l.], p.47–53, 2008.

WANG, T.; WANG, C.; LAU, F. **A Grid-enabled Multi-server Network Game Architecture**. 2004. 26–27p.

ZHANG, X.; ZHANG, Q.; ZHANG, Z.; SONG, G.; ZHU, W. A construction of locality-aware overlay network: moverlay and its performance. **Selected Areas in Communications, IEEE Journal on**, [S.l.], v.22, n.1, p.18–28, 2004.

ZHAO, B.; HUANG, L.; STRIBLING, J.; RHEA, S.; JOSEPH, A.; KUBIATOWICZ, J. Tapestry: a resilient global-scale overlay for service deployment. **Selected Areas in Communications, IEEE Journal on**, [S.l.], v.22, n.1, p.41–53, 2004.

ZHOU, S.; SHEN, H. **A Consistency Model for Highly Interactive Multi-player Online Games**. 2007. 318–323p. v.26, n.28.