A Scalable and Fair Peer-to-Peer MMOG Architecture with Novel State Consistency

John S. Gilmore
Department of Electronic Engineering
Stellenbosch University
Stellenbosch, South Africa
Email: jgilmore@ieee.org

Abstract

Recently, there has been a drive to move away from a Client/Server approach to Massively Multiplayer Online Games to a Peer-to-Peer approach. This can been seen when looking at the number of articles published on Peer-to-Peer systems compared to those published on Client/Server systems. This proposal discusses the different architectures currently in use and discusses the key challenges that a move to Peer-to-Peer presents. From the discussion, a novel architecture based on groups of players is proposed in an attempts to exploit the flocking behaviour of players. Focus is also placed on state persistency and a hybrid state consistency model is presented to deal with persistency in games. The model distinguishes between ephemeral and persistent data in an attempt to improve the responsiveness of the storage. The model makes use of both overlay storage as well as distance based storage to achieve these ends.

June, 2010

I. Introduction

With the advent of broadband Internet, Massively Multiplayer Online Games (MMOGs) have gained significant popularity over the course of the past few years. Figure 1 shows the total number of active MMOG subscriptions over time for the period 1997 to 2008. From here, the accelerating growth of the MMOG market is evident.

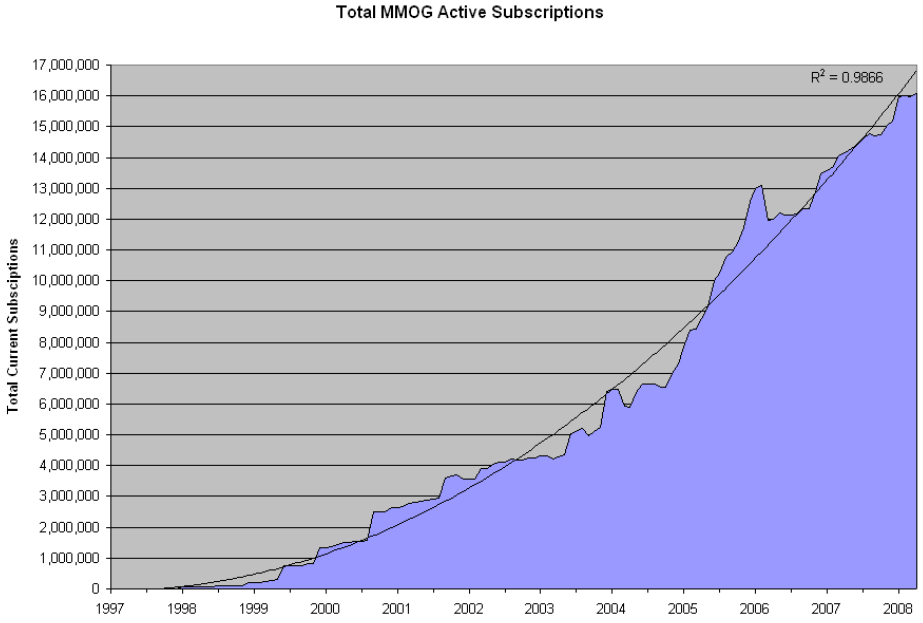


Fig. 1. Total number of active MMOG player subscriptions over time [1].

MMOGs are characterised by expansive worlds, where a large number of players interact online with each other and the virtual environment to achieve certain goals through collaboration and teamwork. Throughout the development of MMOGs, role play has been tightly coupled to this type of game. This is perhaps due to the exploration and player interaction aspects. Role play allows players to fully immerse themselves in the game world and might, therefore, provide for a more compelling experience. Because of this tight coupling, the terms Massively Multiplayer Online Role Playing Game (MMORPG) and MMOG have almost become synonymous. Throughout this work, a distinction will, however, be made between the two.

MMOGs hold great commercial as well as academic value. From a commercial perspective, the growing number of active subscriptions shown in Figure 1 translates to a growing MMOG market. Figure 2 shows the online games market forecast by DFC Intelligence, a company specialising in game market forecasts for various sectors.

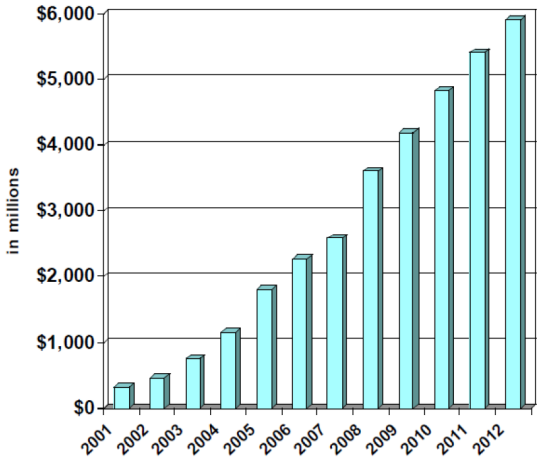


Fig. 2. DFC Intelligence MMOG market forecast '08 [2].

From an academic perspective, MMOGs also hold great value. An MMOG is a complex networked application, with clients requiring reliable real-time feedback on actions taken. The design of an MMOG requires in-depth knowledge of server architectures and network design. The design of a server architecture determines how may players the game will be able to host and what the user experience will be in terms of quality of service. The server architecture must be able to handle thousands of requests, store large amounts of data, update the game state and send responses back to all clients in real time.

MMOGs, therefore, hold great academic as well as commercial value. Recently, significant research has been done in order to develop Peer-to-Peer (P2P) MMOGs. These P2P MMOGs hold many advantages over current systems, as discussed in Section IV-A. These P2P MMOGs, however, still have many issues that need to be resolved before they can become commercially viable, as discussed in Section IV-C.

Part of an MMOG is ensuring low latency, consistent data persistency of all game data. Player data should be stored when players log off from the game. In-game object states should also be stored as well as the states of computer characters in the game. The issue that will be focused on in this study will be ensuring data integrity and persistency in these P2P MMOGs. The question will also be asked which is the better architecture for MMOGs? The classic architecture will be compared to the P2P architecture in order to answer this question. While doing this, a novel architecture will also be developed to improve upon the current P2P architectures by using what was learned from this study.

This study has three expected outcomes:

1) The main outcome will be a state consistency model that is provably better than currently used models for P2P MMOGs.
2) The second part of the contribution is developing a P2P MMOG architecture, which uses the new state consistency model to achieve better performance than current architectures.
3) The third contribution will be to compare different architectures and consistency models and determine which of the other proposed architectures in the literature work better with a P2P MMOG.

Section II presents a history of MMOGs in an effort to show the advancement and major contributors in the field over the past few years. Section III discusses classic networking models used in current MMOGs. The models are described and the advantages and disadvantages of each are compared. Section IV introduces the P2P MMOG networking model currently in use. It discusses all aspects of the model and compared different approaches employed. Section V proposes a novel P2P MMOG architecture, which makes use of knowledge gained from the development of previous architectures. Section VI compares the proposed architectures to related ones and discusses the differences of the architectures and the novelty of the proposed architecture. Section VII details how file storage and consistency are handled in a range of systems currently employed. Section VIII details how file storage and consistency are employed in P2P MMOGs. The three different approaches are discussed in detail. Section IX proposes a novel state consistency model, to support the proposed architecture and improve storage performance in MMOGs. Section X compares the proposed consistency model to a currently used model that is most related. Section XI describes how the system will be evaluated. Including a discussion of the testing of the architecture and simulation and testing of the consistency model.

II. A BRIEF HISTORY OF MMORPGS

MMOGs have a history stretching from 1978 to the present. A complete history of MMOGs and the histories of the games and boardgames that they originate from can be found in [3] and Chapter 1 of [4]. The first games that could be called

MMOGs were Multi-User Dungeons (MUDs) (1978) [5]. MUDs are entirely text-based, with players exploring areas by receiving descriptions of what they were "seeing" and typing commands to move and interact with objects or players. MUDs contain many of the elements that today are still central to the MMOG concept. These elements include exploration, large worlds, multiplayer, social interaction and progression.

After MUDs, there were many MMORPGs that acted as building blocks for what is recognised today as being an MMORPG. The first graphical MMORPG was Neverwinter Nights (1991) [6]. Neverwinter Nights was not an Internet based game, it was hosted on what today is the AOL network. Meridian 59 (1994) was the first MMOG to have featured a monthly subscription fee, receive wide media coverage and most importantly, the first MMOG to feature a 3D engine [7].

The first MMORPG to be commercially successful and largely credited with popularising the genre was Ultima Online (1997). Ultima Online used existing intellectual property from the Ultima universe as well as an aggressive marketing campaign by game publisher EA, to quickly gain 100,000 subscribers. NCsoft's Lineage (1998) looked similar to Ultima Online, but was more Player-versus-Player (PvP) oriented and with an added castle siege mechanism, became very popular in South Korea. Lineage had more than three million subscribers at one point, most of them from South Korea [8]. Everquest (1999) is credited for bringing MMORPGs into mainstream Western Culture. It featured a large persistent 3D environment that was capable of hosting up to 15000 players per server [9].

MMORPGs in the first millennium are considered to be of the first generation. These games provided blueprints for all MMORPGs to follow in the second generation. While there are many more games in the second generation, these games are characterised by little innovation in the genre and focus more on improving graphics and ease-of-use [3]. Notable games in this generation are: Final Fantasy XI (2000) (console based), Dark Age of Camelot (2001) (realm vs. realm combat) and Anarchy Online (2001) (instancing).

Eve Online (2003), developed by CCP Games in Iceland, brought many new innovations to the MMORPG. It was the first successful MMORPG to feature a science fiction theme. It was the first MMOG to have a single distributed server architecture. In 2006, CCP Games launched the largest supercomputer in the gaming industry to upgrade their existing infrastructure and enable Eve to support more than 50,000 concurrent users [10]. This number was surpassed in 2009 with 54,181 concurrent users in-game [11].

Another innovation of Eve was the in-game economy. CCP games appointed Dr. Eyjólfur Guõmundsson as chief economist of Eve online in 2006 [12]. His duties were to monitor and predict market trends in the game world and produce detailed quarterly economic reports [13]. The economy is based on an open market system, ruled by supply and demand. No other game has implemented an in-game economy in such a rigourous fashion.

Blizzards's World of Warcraft (WoW) (2004) is the most successful MMORPG to date. After six years it still has by far the most subscribers of any MMOG, totaling 11,5 million, each paying $15 per month subscription [14]. In 2008 it was estimated that WoW holds more than 60% of the MMORPG subscription market [15]. From the first generation of games, a steady growth has been seen in the MMOG space, but before the run away success of WoW, no one had estimated that the gaming market could be this large [16]. It should be noted, however, that the growth seen in the MMOG market, is mostly due to growth in the Asian markets and that the size of these markets are much larger than that of Western markets.

The success of WoW has largely been attributed to the overall quality and finish of the game [17]. It is interesting to note that WoW is not attributed with many innovations. Most games that came before it implemented most of the features in WoW. What WoW did do, is combine all previous innovations into a package that was accessible to a large number of people. Players also don't just play WoW to experience the game content, they play the game to meet up with friends and socialise. Guilds are also an integral part of WoW. Guilds are collections of players that choose to play together to achieve some common goal.

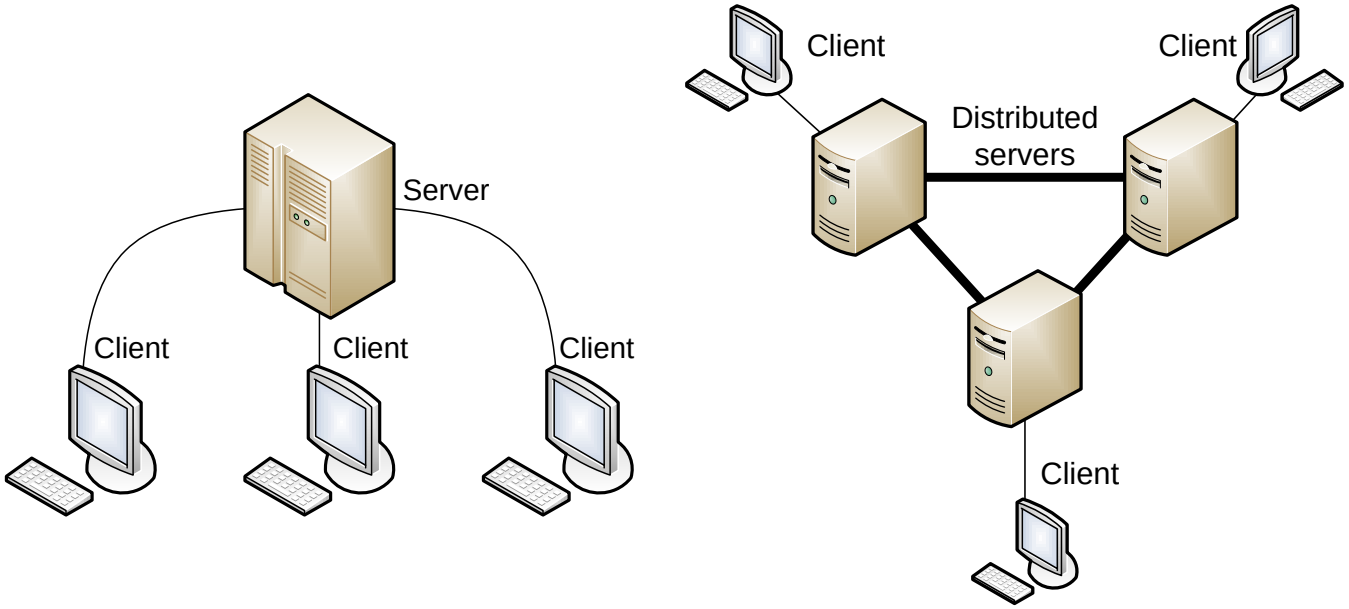
All games in the history of MMOGs can be thought of as a collection of architectures that together make up the game. These architectures include the graphics architecture, sound architecture and networking architecture. This proposal will focus on the networking architecture of the game.

III. Classic MMOG network models

The network architecture, also called the network model of a system, specifies how the different entities in the network communicate, where authority lies and whether a centralised or distributed approach is followed. The networking model used, determines how the different nodes in the network interact and what the roles of these different nodes are. The consistency model used, is based on the networking model, but determines how data are stored on the system as well as how data are moved from one node to another. Throughout this work, data persistency is handled as a subset of data consistency.

Currently, all MMOGs being operated run on a Client/Server (C/S) networking model or a distributed C/S networking model, also called a Client/Multi-Server (C/MS) model. The simplest form of a network model used in MMOGs is the pure C/S model. Other games, such as strategy games and first person shooter games regularly employ the peer-to-peer networking model to achieve greater system responsiveness, as discussed in Section VII-A.

Figure 3a shows the C/S model. The server is the entity on which the MMOG is hosted and is controlled by the game producer or the game operator. Clients are computers operated by players, that connect to the server to play the game. The server is responsible for handling all queries from clients. Clients never communicate with other clients; they send their actions to the server and receive the updated states of other players from the server.



(a) Client/Server (b) Client/Multi-Server

Fig. 3. Network architectures

The C/S architecture has two main advantages that made it the architecture of choice for all MMOG developers. Because of the centralised approach of the architecture, both administration and security are greatly simplified. Administration is simplified, because the game producer has full control over the server, server data and code. Efficient logging is also supported, because the server is able to not only log all server actions, but also all client actions.

Security is a significant issue in MMOGs, since some players sell in-game currency for real-world currency [18]. This makes the MMOG a platform that is capable of producing income, which increases the incentive of players to be able to gain an unfair advantage over others. The more popular an MMOG, the greater the security threat. Because the producer has full control over the server code and is never required to furnish the client with the server code, a potential attacker never has any knowledge of the server architecture and code. Because clients are never allowed to communicate, all malicious users can be filtered out of the network by the server when detected and even banned from the network.

Producers are able to ban players, since these games usually require a game account, which is linked to a copy of the game as well as some payment method. This introduces a large cost to players whose accounts are banned. The server or cluster is also housed in a secured location, where access can be controlled.

These factors simplify the security of the C/S model by allowing the developers to place all intelligence in the server and move all sensitive computations to the server. This provides the C/S model with a high level of security.

The C/S architecture however does have some disadvantages. These are: weak robustness, weak scalability, high cost to the provider, high latency, high amount of required server bandwidth and weak handling of transient loads. These disadvantages are described below.

The robustness of the system is weak because it is a single point of failure. If the server fails or goes down for maintenance, the game is off-line and players are unable to play.

This system is also not scalable, since a single server cannot easily be extended with more resources. Even if an off-line approach is used, where hardware is upgraded after the system is taken down for maintenance, the hardware required to support a game hosting more than 3000 players, become prohibitively expensive. In 2004, it was estimated that an MMOG that supports approximately 30,000 players, requires 2 to 3 years to develop and costs more than $10 mil [19], [20]. With newer games using high definition graphics and the general drive in the market to improve on previous titles, the cost of MMOG games can only be expected to increase.

The server hardware should be able to support peak system loads, which means that sufficient resources should always be provisioned to support these peak load. This is not an economically viable solution, because resources to handle peak loads are not used most of the time. This translates to producers paying for the provisioning of resources, without having active players that pay for these resources.

This disadvantage also leads to a high cost for the provider, which translates to high costs for clients or reduced profit. The cost is due to the hardware that is required to host the system as well as maintenance and running costs. Running costs include bandwidth provisioning and IT personnel. Maintenance costs include replacement of malfunctioning or outdated hardware. It

has been estimated that maintenance and running costs consume approximately 80% of game revenue during the lifetime of the game [19].

Because no clients are allowed to communicate with other clients, every change that is made to the game world by a client, first had to be communicated to the server, which in turns relays this message to all clients after applying game logic and artificial intelligence (AI) algorithms. This two hop path, with the additional time for computation added by the server as well as possible buffering at the server when many clients communicate, significantly increases the latency of the system compared to a system where direct communication is used.

In an effort to address some of these issues, the distributed C/S, also called C/MS, was introduced. In a distributed C/S model, the server functions are distributed amongst multiple machines in an effort to distribute the server load. The server functions can be distributed by means of redundancy or specialisation. In a redundant system, server functions are duplicated. In a specialisation system, different server functions are handled by different distributed servers.

In general, the issues addressed and improved by the C/MS architecture are robustness, scalability, and peak load handling. The system is more robust, because the failure of one server will not necessarily lead to the failure of the whole system for certain system designs. The system is more scalable, because many less powerful servers may be used, which allows for the hosting of more players than what is currently possible with single server hardware. It also handles transient loads better, because, for cases where loads can be predicted, resources can by shifter between servers to improve the user experience.

The disadvantages of this system is that the administration complexity is greatly increased. Such systems, although capable of handling many more users than a single server, is also much more expensive. These disadvantages are, however, not technical problems and so it is assumed for current games, that these systems are what is required if a game is to be hosted for a large number of players.

IV. Peer-to-Peer MMOG network models

A. Overview

Recently, an architecture making use of the peer-to-peer networking model to host MMOGs, has gained popularity. This was first formally published in [21]. A new research field has been opened up, which is attempting to make the peer-to-peer model a viable alternative to the classic C/S and C/MS architectures. This architecture does, however, still have a few major issues that need to be solved before MMOGs can be developed that use it. If these issues, discussed in Section IV-C, can be solved, a P2P architecture holds some powerful advantages over a C/S system.

The core idea of the P2P model is that each peer contributes sufficient resources to the network to be able to host itself. This also means that all functions of the server in the classic C/S model are distributed amongst all peers. There are many areas where the P2P model can improve on the classic C/S model. These areas are robustness, scalability, provider cost, latency, server bandwidth and handling of peak transient load.

The system is very robust, because there is no server that can fail, only individual peers. Individual peers failing will not affect any other peers other than the peer that failed. This behaviour makes game down-time extremely unlikely.

Furthermore, because every peer is expected to add sufficient resources to the system to host itself, this makes the system very scalable with no extra costs being incurred from a provider viewpoint for any peer that joins the network. This will also allow for efficient handling of transient loads. If many players suddenly enter the game, no resource provisioning issues will arise, as peers already possess their required resources. It should also be noted that it is not at all unreasonable that a peer will have sufficient resources to host itself. Peer computers are very powerful systems these days, with multi-core CPUs, multiple gigahertz of clock speeds, multiple gigabytes of memory and secondary storage space in the terabyte range. The graphics cards in gaming machines have also become immensely powerful.

P2P architectures also create a lot of opportunity for independent developers, because a large initial investment is now no longer required to purchase the expensive server hardware. Not just are hardware costs greatly reduced, but running costs are also greatly reduced. The bandwidth required by the game server is now shared amongst users. Which means that no bandwidth costs will be incurred by the provider. Also, the amount of required bandwidth per user is usually not high, which means that users will not be expected to use a much greater amount of bandwidth.

Latency is also improved, because it is now possible to directly communicate between peers and not necessary to go through a server for communications. There is also no single server that has to process peer actions. Peer actions need only be processed by other peers who find the specific peer actions of interest. The distribution of the load as well as direct communication will reduce latency.

There are, however issues with administration and security. From an administrative perspective, it is more complex to administer a decentralised system than a centralised one. For a peer-to-peer system, the server is the collection of all peers. The game producer does not have direct access to peers and so controlling these nodes become significantly more complex. P2P security issues also stem from the decentralised nature of the system and the fact that an attacker has full access to all system code. These issues are discussed in more detail Section IV-C.

Table I summarises the differences between the three architectures as discussed thus far. From this architecture and the precious discussion, it can be seen that the P2P architecture has a lot of advantages over a classic C/S architecture, but that

there are issues that have to be dealt with. These include ways to administer the system, when there is no centralised form of control and security issues arising from the decentralised nature of the architecture.

Property	Client/Server	Client/Multi-Server	Peer-to-Peer
Administration	Low	Medium	High
Security	High	High	Low
Robustness	Low	Medium	High
Scalability	Low	Medium	High
Provider cost	High	Very high	Low
Latency	High	High	Low
Server bandwidth	High	High	None
Peer bandwidth	Low	Low	Medium
Peak transient load handling	Bad	Medium	Good

TABLE I
DIFFERENCES BETWEEN CLIENT/SERVER, CLIENT/MULTI-SERVER AND PEER-TO-PEER ARCHITECTURES

B. Structured P2P overlay networks

P2P architectures can be divided into structured and unstructured types. The classification is mostly based on routing and content retrieval in the network. Unstructured approaches generally make use of flooding techniques to obtain data items.

In flooding, a query is send out by one node to all of its neighbours. If these neighbours do not posses the item, they send the request to all of their respective neighbours. Various issues have been identified with flooding [22]. Flooding is unscalable as the number of queries grows exponentially with the number of nodes in the network. Not just is it unscalable, but the node originating the query is not assured that an item residing on the network will be found.

These attributes of flooding make it an unreliable option for use with P2P MMOGs. Structured overlays have been proposed that provide for efficient routing. Some of these well known overlays are: CAN [23], Chord [24], Tapestry [25] and Pastry [26]. The overlay most used in P2P MMOG systems is Pastry, as Scribe [27], which implements Application Layer Multicast, runs on top of Pastry.

The basic idea of an overlay is that all nodes are identified by unique IDs, which are hashes to a circular key space. Any node in the overlay network is then able to efficiently route a query with a given ID, to a node with an ID closest to the given ID.

Clustered DHTs have also been developed, to be more resistent to network churn and reduce reorganisation in the network [28]. Such an overlay consists of groups of peers. The network is constructed at a group level, rather than a node level. This reduces the reorganisation in the network as nodes join groups and groups themselves remain more or less constant. There might also be some applications for clustered DHT in a tiered state consistency model as described in Section V.

Since the introduction of overlays, they have been employed for various tasks. One task is simple message routing, but on top of the routing layer, many other services have been implemented. These include Application Level Multicast (ALM), distributed storage [29] and indexing. ALM requires the presence of a structured tree to send messages over. Implementations such as Scribe use the Pastry overlay to form a multicast tree over the structured network overlay.

PAST [29] uses Pastry to implement a distributed storage system. Files that have to be stored are given IDs, by using some hash function, for example SHA-1 [30]. The file, along with the ID is sent as a message over the overlay. The messages is then routed to the node whose ID is a closest match of the file ID, where the file is stored. If any nodes wishes to retrieve the file again, it only required the file hash. It can send a "get" message to the overlay, which will route the message to where the file is situated and retrieve the file. A great advantage of this storage technique is that the hash function is random, which ensures a good first order distribution of files over the complete key space.

Both PAST and Scribe are widely used in P2P MMOGs as will be shown throughout this proposal.

C. Key challenges

A recent article has identified six key challenges of P2P systems: Interest Management, Game Event Dissemination, NPC Host Allocation, Game State Persistency, Cheating Mitigation and Incentive Mechanisms [31]. Currently, these are the most pressing issues to be addressed in the design of a P2P game. Each of these challenges will be described below, except NPC host allocation. NPC host allocation is handled as a specialised form of state persistency management in this work.

1) Interest Management: Interest management is used to determine the smallest amount of information that a peer requires, in order to present an accurate representation of the world to each player. The idea is not specific to P2P MMOGs and was already formally put forward in [32] and later with greater focus on a distributed environment in [33]. The main idea is that a player has a limited visual range and a limited area around the player in which it can interact with objects. The player requires update information of all objects in this area, called the player's Area of Interest (AoI). AoI calculations also rely on the fact the a player's direction and velocity of movement cannot change instantaneously and are bounded. Extensive research has been

done into solving AoI problems and a comparison of techniques can be found in [34]. The solutions range from aura/nimbus [35] to publish/subscribe [36] to Voronoi based models [37], [38] to hybrid models [39], [40], [41].

Generally, interest management solutions can be divided into coarsely or finely grained solutions, although the hybrid models, especially MOPAR, seem to have gained greater popularity because they seem to have all the benefits of the two solutions and little of the drawbacks [40]. MOPAR has been shown to perform better than either a finely grained technique or a coarsely grained technique.

Coarsely grained solutions usually divide the game world into multiple regions and when a player enters a region, it subscribes to that region's events. This is called the region-based publish subscribe model [31]. All players in the region then receive the region's events, even for players not in their AoI. Finely grained techniques create groups of players from their AoIs. Groups of interacting players directly exchange information, so all players only receive information that is relevant to them. This has been termed the spatial model [31]. The grain of the solution in turn determines the type of event dissemination that should be used, as described later in this section.

MOPAR partitions the game world into hexagonal regions and appoints "home" nodes to act as bootstrap nodes for each region. A home node of a region is that node whose ID is the closest match the the region ID. This allows any node to find the home node for a region. A master node is then selected for every region, whose function it is to inform slave nodes of new neighbours. All slaves nodes in a region register at their region's master node. Slave nodes send direction and velocity updates to their masters. Masters communicate directly with other masters if a node is about to enter their region. Masters inform their slaves of a new neighbour. Slaves communicate directly with each other, once identified by a master.

2) Event Dissemination: Event dissemination deals with how information should be sent to peers after interest management has determined which information should be sent. The first application of event dissemination for online games can be found in [42]. Recently, ALM and unicast techniques of event dissemination have become popular, depending on the grain of the event dissemination. ALM is used, instead of router level multicast, because of a lack of general support for this technology at the router level [43].

ALM is used for coarsely grained interest management techniques, while unicast is used for finely grained techniques. Unicast is not used for coarsely grained techniques, because it is not scalable. For a network with N nodes, N^2 messages are exchanges for every player action. ALM, however, significantly increases the message latency in the system, because messages first have to be routed over a structured overlay network. ALM is however preferred over unicast for large numbers of messages, because of the weak scalability of unicast.

3) Cheating Mitigation: Cheating mitigation has been identified as a major issue for P2P systems [21], [44], [45]. The challenges reside in the fact that peers are not under the control of the game producer. Since all server data are distributed amongst peers, all peers have access to sections of the server data. Peers also have access to the distributed server code. One advantage that can be exploited to prevent cheating is that no peer contains all server data and no one peer has more authority than another.

There are various security issues and these are usually divided according to the level of the protocol stack where they occur. The areas that have been identified by [45] and expanded upon by [46] are: game level, application level, protocol level and infrastructure level. This is consistent with the generally used layered security model [47]. Game level cheats are ways in which a malicious player may gain an unfair advantage over other players, within the confines of the game. These cheats are usually because of software bugs and some examples are duplication and teleport cheats.

Application level cheats are where malicious players alter the game software to gain an unfair advantage. This is usually done by gaining access to the game state to which they should not have access at the current time. An example of this is map reveal cheats in strategy games. Where the fog of war is removed and the player can observe all the opponent's movements. Other cheats are sometimes used that augment the player's User Interface (UI) with extra information that allows the player to make more informed decisions. It is debatable whether these additions are cheats. They are, however, considered almost essential for competitive WoW play.

Protocol level cheats are cheats based on the different methods of communicating data across the system. These usually concern dropping, delaying of modifying IP packets to achieve certain outcomes in the game. Infrastructure level cheats concern exploiting the underlying infrastructure on which the games are built. These include hacking the hardware or P2P overlay.

As with all taxonomies, all cheats may not cleanly fit into one if these boxes, some cheats may occur over multiple levels or a cheat with a specific outcome can be implemented differently on different levels. The field of P2P security has recently received more attention than in the past and has started to bear fruit [48]. This is, however, an ongoing research field with many issues still open. For an in-depth review of the security issues facing peer-to-peer system in general, refer to [49]. These issues are the same issues facing P2P MMOGs, with the exception of the game and application layer issues.

4) Incentive Mechanisms: P2P schemes require all players to share resources in order to ensure that the system functions correctly. The issue with this is that players are people who might not want to share their resources, but still benefit from the resources of others. This is where incentive mechanisms become important. The function of these mechanisms is to ensure that all players contribute resources, by incentivised contribution.

All distributed resource sharing models require incentive mechanisms. Bittorrent systems for example use the tit-for-tat protocol [50] to ensure that all people downloading data are also contributing data. Such mechanisms are also required with

P2P MMOGs. Peers playing a game should be incentivised to contribute resources to the system. One advantage in designing an incentive algorithm for a P2P MMOG is that players can be made to contribute resources for the duration of play. The issues with file sharing systems are not present where a peer, after downloading a file, has no more incentive to contribute. When a peer plays a game, incentive can be created to provide resources for the duration of the game.

Some incentive schemes proposed increase a player's reputation when resources are provided [51] [52]. This might create a type of meta game, where players try to gain as much reputation as possible. It can however be argued that this scheme does not really enforce the provisioning of resources. A player who does not want to provide resources might not see a higher reputation as sufficient incentive to provide resources.

Other issues with incentive schemes is that sometimes players have limited or no resources. Such players should be hosted with other player with sufficient resources and not be disallowed to play the game. When limited resources are taken into account, the issue of reporting a false amount of available resources becomes a problem. A peer that has sufficient resources, might report insufficient resources, to not be penalised.

The field of incentive mechanisms is an ongoing one.

5) Game State Persistency: The issue of game state persistency will be dealt with in detail in Sections VII, VIII and IX. Game state persistency will form the focus of this work. All that might be said at this stage is what was stated by Lu Fan in Section 3.5.3 of a recently completed PhD on the topic of P2P MMOGs: "Game state persistency is a major challenge for P2P MMOGs as existing P2P storage infrastructures are designed to support file sharing, and seldom fulfil the performance and security requirements of a MMOG. . . . the persistency area is still immature with many problems waiting to be investigated." [2].

V. Proposed P2P MMOG architecture

A. Initial implementation and test framework

Initially, the Badumna P2P MMOG architecture, which is described in Section VI-C, will be deployed. The deployment of this existing architecture will assist with the understanding of P2P MMOG architectures in general. It would also allow for thorough testing of such an architecture, to determine what the expected values of the runtime parameters are.

The parameters or metrics that will be used to test the architecture as a whole will be those metrics that mostly influence game play from a player's perspective. These metrics can then be used to judge the performance of the architecture. The metrics that were identified as most relevant to a player, which are affected by the network architecture, were: *latency* and *bandwidth*.

Bandwidth, also called bit rate or throughput, which is measured in bits per second (bps), is a measure of how much information can be transmitted in a certain amount of time. This metric is important, because it determines what the Internet speed requirement is on the player's end and also the costs involved.

Latency, measured in seconds, is a measure of how low an information packet takes to travel from the source to the destination and back. Here, the term is defined to include both the propagation delay as well as the processing delay. In computer games, latency is of key importance as it affects the real-time nature of the game. There is a certain delay threshold after which the user experience starts to degrade. This perceived delay is termed "lag". The threshold is different for different genres of games. For First Person Shooter (FPS) games, for example Quake 3, this threshold seems to be around 150 to 180 ms [53]. For RPGs and Real-Time Strategy Games (RTSs), delays of up to a second are still acceptable [54].

Different P2P MMOG architectures will process game data differently, or require different amounts of processing power. All these differences will affect the latency of the game. A combination of techniques should be used in the architecture, so as to minimise the latency as well as the bandwidth usage of the game.

It is proposed that the architecture will be deployed onto nodes in the cloud. This is the only practical way in which the architecture can be tested with thousands of nodes. It is proposed that the Amazon Elastic Compute Cloud (Amazon EC2) be used, as the EC2 cloud provides one with low level root access as well as a web based management interface to any virtual machine created.

After the creation of a virtual machine and the loading of the Badumna architecture onto that machine, testing can commence. Latency and bandwidth usage can be monitored by running network monitoring software, such as Wireshark, on every virtual node.

The testing of the Badumna architecture will initially only test the API functions of the architecture. This includes creating game objects, updating object states and synchronising states with replica objects. After Badumna has been implemented and tested, a novel architecture will be developed as described in the next section.

B. Implementation using grouping

We propose to develop a truly distributed P2P MMOG architecture, using hybrid techniques to improve upon current performance. The architecture will be developed to solve the six issues of P2P systems discussed in Section IV-C. Existing solutions to each of the six issues will be combined into a novel architecture, with the exception of state persistency. For state persistency, a completely novel approach will be followed in line with the design of the overall architecture.

At the core of the architecture will be how peers are grouped. The architecture will not use regions to group peers, but rather, make use of the flocking behaviour of players to dynamically group players into flocks or clusters [55]. The main idea of flocking is that players move around in groups, rather than randomly on their own. These clusters can then be used as the nodes in a clustered Distributed Hash Table (DHT). Hybrid mechanisms can then be used to either interact with players at the individual level or the cluster level. Because of the flocking behaviour of players, dynamic groups or regions that move with groups of players may be a better fit than static or even dynamic regions that operate on areas of the virtual world and not on how groups of players actually cluster.

For Interest Management, a hybrid approach will be followed such as MOPAR. A hybrid interest management technique will fit well with the proposed architecture. The only difference is: where regions were previously used, groups will now be used. MOPAR's direct communication between peers reduces bandwidth usage in the network, because at this level, a finely grained interest management technique is used. The structured overlay does however assist in ensuring that all peers remain connected or able to reconnect if they have become disconnected by use of the home node.

We believe that a scheme where the regions or groups move with the groups of players will enhance the functionality of the interest management scheme. If players are grouped more intelligently, less traffic will have to flow between groups, which will reduce the number of queries to the DHT, which in turn will improve the latency of the overall system. Using groups or flocks would, therefore, compliment this technique. The issue with such a system would be how to uniquely identify a group and how the identification would be applied when groups merge or split. The identification is required in order to ensure that disconnected nodes are able to reconnect, once they have joined a group.

As Game Event Dissemination is closely coupled to the Interest Management technique, unicast will be employed. Unicast is used between all master nodes as well as between all slave nodes in MOPAR. Unicast holds many advantages over ALM, if the number of neighbouring nodes are bound. ALM introduces significant delays into the communications system, by the use of a network overlay as discussed in Section VIII-D.

Security is also of great importance to the design of an MMOG architecture and so security issues will be kept in mind when designing the MMOG architecture. Recent advances and surveys can be used to improve game security, by improving the security of currently used peer-to-peer overlays. Examples include using secure node ID assignments, by making use of a Certification Authority, or designing the system in such a way that a peer cannot select or report it's own node ID. All aspects of security will be investigated, this includes Authentication, Authorisation, Data Integrity, Confidentiality, Availability, Trust, Privacy and Identity Management [47].

Incentive Mechanisms are also of key importance in a P2P system. If peers are not required to share resources, this will lead to system degradation. Two factors should however be addressed: How will resource provisioning be incentivised and more importantly, how will this scheme be made secure. In other words, the scheme should not only ensure that resource provisioning be incentivised, it should also ensure that peers are not able to cheat the incentive mechanism, by providing false information to other peers.

The focus of this work will, however, be on game state persistency. This is an area that has not received much attention in the P2P gaming field. Significant focus has been placed on Interest Management, Event Dissemination and, to a lesser degree, on Cheating Mitigation. The proposed state persistency mechanism is discussed in detail in Section IX. As an overview, a distinction will be made between ephemeral data and persistent data. A peer-to-peer overlay will be used to securely store persistent data, while a distance based approach will be used to store ephemeral data in a group or flock of peers. This will allow for high speed game state updates for peers within the same group, but the structured approach will still ensure the availability of the game state to other flocks of players, to a lesser state of consistency. It is also argued that players not near to each other, do not require a perfectly consistent view of the complete world. As different flocks near each-other, the degree of state consistency will become higher until the groups merge and the states are consistent.

It is argued that NPC hosting is a specialised form of state persistency. NPCs are seen as objects with logic and state, both of which are data that has to be stored. This merely requires for the definition of game state to be expanded, as elaborated upon in Section VII. Object logic is both something that has to be stored as well as executed. The question of how and where object logic should be executed should also be investigated.

VI. Related P2P MMOG architectures

As the field is still in its infancy, few, if any, complete P2P MMOG architectures have been produced. Some papers describe their work as being a full architecture, but then goes on to only describe one aspect of the implementation. In those circumstances, source code is also not available to help determine whether a fully working architecture has been implemented.

Many of the papers reviewed only describe a partial implementation a P2P MMOG, even if presenting the solution as a complete one. Some articles describe sate persistency, as discussed in Section VIII-B, others describe interest management as presented in Section IV-C1, still other describe security issues as presented in Section IV-C3.

Few implementations could be found that attempt to implement a complete P2P MMOG implementation. These implementations are Mediator (2007) [2], VAST (2007) [56] and Badumna (2009) [57].

A. VAST

VAST uses Voronoi-based interest management techniques with unicast event dissemination. Unicast is usable in this case because of the small neighbour sets present in the Voronoi-based approach. It uses a distance-based NPC hosting mechanism, but centralised state persistency.

B. Mediator

The Mediator framework employs MOPAR-like hybrid interest management with unicast event dissemination. It implements a task sharing approach to NPC hosting, where NPCs are represented as tasks that require computing power. These tasks are then advertised by a specialised super peer. Tasks can be undertaken by peers with sufficient computing power. It implements state persistency by using PAST overlay storage.

C. Badumna

The recently developed Badumna uses a three tiered interest management technique. The three tiers are called: "Cell", "Dynamic Bounded Region" and "Gossip". These are techniques based on the aura/nimbus design. Different techniques are used for different network conditions. As player density increases, interest management moves from cell, to dynamic bounded region, to gossip.

In the cell protocol, where all peers start out, the virtual world is divided into regions, with region manager super peers. These super peers maintain lists of nodes in their regions and provide these lists to nodes entering the region. When too many auras intersect, dynamically bounded regions are formed. This is done by grouping peers in some way that is not explained and adding these groups into the regional cells. Another tier is effectively created, with each group having a super peer and a group also being an entity in the virtual world. A group with an area of interest (AoI) is then inserted into a region and when the AoIs of groups intersect, the super peers in the groups are informed. These group super peers then inform the other peers in the group. This approach is used in an effort to decrease network traffic. The gossip protocol attempts to avoid overloading the regional super peer by reducing the rate at which peers query the super peers and allowing normal peer nodes to broadcast their information to their neighbours. It is not clear how this tiered scheme performs to other schemes as this is never compared. Only the different tiers are compared.

In [57], where Badumna is described, it sometimes seems that Interest Management, Event Dissemination and State Consistency are confused for the same thing. Interest management techniques are described as being multicast-based or DHT-based. Apart from the fact that multicast-based schemes usually are also DHT-based, the paper seems to describe DHT-based schemes as state persistency schemes, rather than interest management schemes. Where the example objects (cat, dog, and cat and dog) are stored somewhere on the network. No mention is made here of actual interest management, which is how to determine what is of interest to which peers in the network.

It is also said that multicast schemes are not suitable for interest management, because of the high transmission latency, which is true. But the paper than goes on to state that DHTs are very suitable for interest management, because of the lookup features. No mention is made here that DHTs posses the same latency as multicast, mostly due to that fact the high multicast latency stems from the use of DHTs.

What Badumna does provide, is a usable implementation of a P2P MMOG network suite that is currently being implemented for commercial use in Vastpark, is integrated with the Unity game development tool and OpenSim, and has been used to create a working technology showcase, called "Troll Basher" [58]. Badumna does not seem to be a complete implementation of a P2P MMOG architecture, but it allows developers to use it as a base to develop a complete P2P MMOG architecture on.

Issues of state persistency seem to be left up to the developers. Badumna provides functions to create game objects and to ensure that these objects are synchronised with other duplicate and non-authoritive objects. Where the authoritive objects are stored, is left up to the developer. This system could thus provide the base to implement a P2P MMOG architecture on, using novel state persistency and consistency mechanisms. Which is exactly the novel contribution of the proposed work.

None of the architectures reviewed have investigated dynamically grouping peers in a natural way, in an attempt to exploit the flocking behaviour of peers.

VII. Classic consistency models

A consistency model is based on the network architecture, but deals with how data are stored and distributed throughout the system. The consistency model specifies on what type of nodes what types of data are stored and also how data consistency is ensured with multiple node requests.

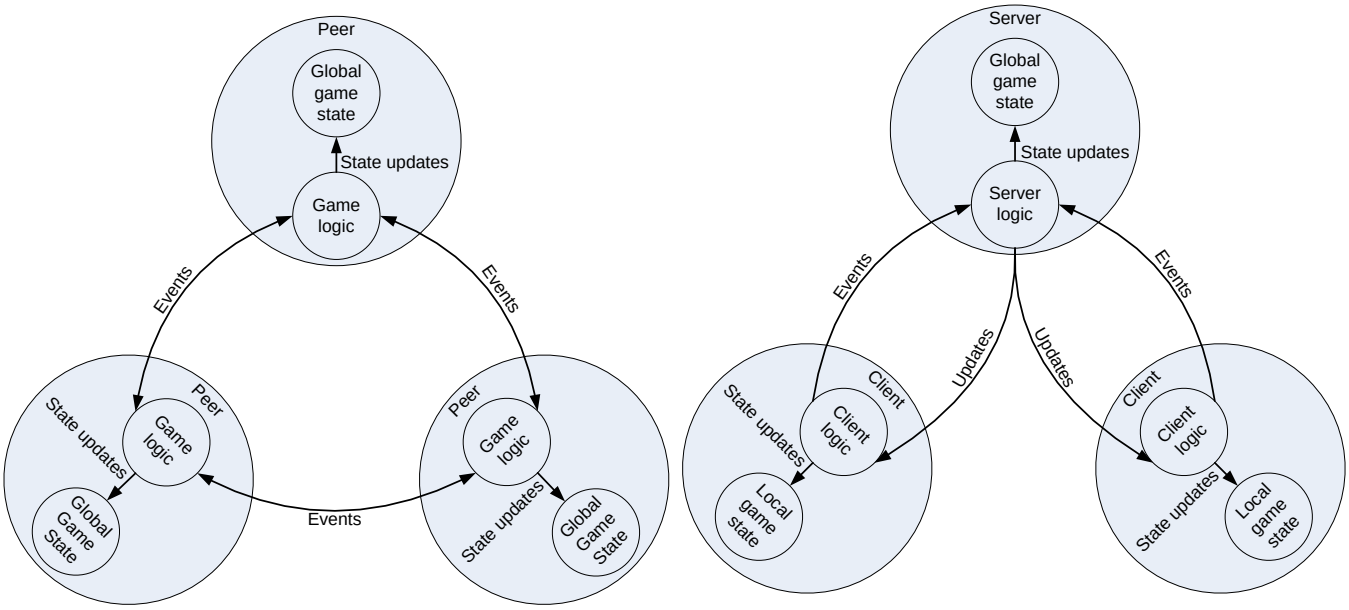
To understand consistency models, some basic terms should first be understood. These terms are: "event", "update", "game state", "game logic" and "game object".

Events — Events are generated by players and can be thought of as actions taken by players. These include casting a spell, using an item or walking.

Game Logic	Game logic is applied to events to determine what updates should be applied to the game state. Game logic is thus a "think" function, which determines how the world should change as a result of an event. Another way to think about game logic is to see it as the game rules. A player casting a spell might cause another player's health to be reduced, her own health to be increased or a monster to spawn. When a player is walking, the logic will cause the player's position to update at the player's walking speed.
Update	Game logic communicates how the world should change via game updates. Game updates are the incremental changes that specify how the game state should change.
Game State	The state of the game is the positions, health and all other attributes of all players, NPCs and game objects in the game world. Game state consists of a collection of game objects. An NPC as well as an immutable plant are both examples of game objects that together make up the game state.
Game objects	When discussing how to segment game state, it is sometimes easier to speak in terms of game objects, since they are separable. For the purposes of this work, game objects are objects with both state and logic, which means they consume both storage space, as well as CPU power. Game objects can also produce events, which should be sent to other objects. When this definition is used, NPC objects may be classified as a specific type of a game object, which forms part of the global game state. The question of NPC hosting then also becomes a question of state persistency.

A. P2P and C/S consistency models

As an introduction to consistency models, an overview of the two common models, currently used in computer games will be described. The models used in P2P MMOGs are all permutations of these two basic models. The two models are based on the two different network models. These are the P2P-based model, also called the event-based model [59], and the Client/Server-based model, also called update-based model [60].



(a) Update-based (peer-to-peer)

(b) Event-based (client/server)

Fig. 4. Consistency models

1) Event-based: Figure 4a shows the P2P model. In this model the complete game state is stored on each peer. Any event that a peer generates is sent to all other peers. These events are used as inputs to the game logic, which creates updates, which are then used to update the global game state at each peer.

It is at this level that consistency becomes important. The order in which updates are received should be the same for all peers, otherwise the game states of different peers may become inconsistent. Usually some kind of lockstep technique is used to solve this issue [61]. The issue with lockstep is that it reduces the latency to twice that of the peer with the highest latency. Various techniques have been proposed that improves the latency by introducing some deadline before which all events should be submitted [45]. This, however, makes it impossible for a player with a high latency to play the game with anyone other than from her own continent.

The issue with the event-based model is that it is not scalable, since all peers should connect to all other peers and every event is transmitted to everyone. This means that as N, the number of peers in the network, increases, the amount of traffic

increases with a factor of N^2. The security issues of the P2P networking model, on which this consistency model is based, are also present. Slowdown is also experienced by all players if one player's latency is below par, since the lockstep mechanism has to wait for all events to be received for that round to conclude.

2) Update-based: An alternative to the event-based model is the update-based model, shown in Figure 4b. This model is based on the Client/Server networking model. An authoritative global game state is housed on the server and a non-authoritative local game state is housed on all clients for display purposes. No real game logic is housed at the clients, only on the server. All clients send events to the server, which applies the game logic and sends updates to the clients, while also updating its own game state.

This approach greatly assists with security, as clients cannot influence the state of any other clients and every client's state depends on updates received from the server. The server state is also termed authoritative, because if there is a conflict, the server state is always the state to which the system is expected to return. All the security advantages of the C/S model also apply to this consistency model. Another reason why the event based model is successful is because it is more scalable then the pure P2P model. More hardware can be used to build a more powerful server, which can handle more clients. This is however very costly as discussed in Section III.

B. Client/Multi-Server consistency models

Apart from the two classic models, there are also models based on the C/MS network model, which are: shard-based, replication-based, object-based and zone-based [37].

1) Sharding: Sony introduced the first consistency method for a C/MS network in Everquest, where copies of the game world ran on different servers and players connected to one of these servers [62]. Sony termed this method: "Sharding". Clients are not able to interact or communicate with players on other shards, which reduces game immersion. This method does, however, allow for a more scalable system as maximum load is fixed. Players are not able to enter a shard if that shard has reached it capacity. This has in the past caused unhappiness amongst players, since popular shards could be difficult to log in to. Players are also reluctant to move to a new shard, because a lot of time is invested in their characters in their "home" shard. Sharding doesn't allocate resources efficiently, as one shard may be overpopulated while another is underpopulated. There is no way to dynamically distribute the available resources from one shard to another. For all practical purposes, this approach is still merely a C/S approach, with players forced into a specific C/S environment.

2) Replication-based: The replication-based model is very similar to sharding, with the difference that all servers share the same duplicated game state. Each server contains the global game state and clients connect to any one of these servers (mirror-servers [63]) or through a load distribution algorithm to a server (proxy-servers [64]). Each server handles all actions from clients and updates its own database. The servers in turn send updates to each other over a high quality link, such as fibre, to maintain database consistency at high speeds. The problem with this system is that the world is never truly consistent and that there are no optimally chosen inconsistency obfuscation boundaries. In other words, two players standing next to each other in the virtual world, might be on different servers and, therefore, experience two slightly different worlds.

3) Object-based: The object-based method distributes all in-game objects amongst the servers [65], [66], [67]. For an MMOG, most of these objects are expected to be players objects. The advantage of this method is that the system load is fixed for a certain player population and that the load is equally distributed amongst all servers. This allows for more accurate prediction and provisioning of required resources, but still does not handle transient loads well. Another issue is inter-server communications for this architecture. The inter-server communications are random and also much more than the inter-server communications for a region based system. The reason for this is that the amount of player interaction increases with a decrease in the distance between the players. Players playing together move together, chat and interact with Non-Player Characters (NPCs) together. For a region based model, all player-neighbour interactions remain local to the server.

4) Zone/Region-based: The zone-based method divides the virtual world into zones or regions, which are hosted on different servers [68], [69]. Busy regions are hosted on their own servers, while multiple quiet regions are hosted on a single server. This is termed the static region approach [68]. The issue of the static region approach is that it does not scale well when one region is suddenly populated with players. This type of behaviour happens quite regularly and is known as flocking [55]. When players find something of interest in a region, many players will flock to that region. In-game events and festivals are also becoming popular and these events also cause flocking to the region where the event is held. The solution to these effects have been over provisioning of resources to handle peak loads, which suffers from the disadvantages discussed above. Also, if the load changes, the server has to be brought off-line in order to balance the regions. Dynamic regions are being investigated, where regions can be dynamically shifted from one server to another, in order to balance load [69]. This approach adds overhead and significant complexity with regards to the migration of the data and the handling of player actions while the data are in transit.

VIII. P2P MMOG CONSISTENCY MODELS

A. Key Challenges

The key challenges related to P2P MMOG consistency models identified during this literature study were: reliability, responsiveness, security, fairness and consistency.

Reliability — For the storage to be reliable, it must not be possible for data to be lost, and stored data should always be available when a node requests it.

Responsiveness — To ensure system responsiveness, data must be stored or retrieved in real-time. With real-time, it is meant that data should be available within a certain time frame that would ensure correct functionality of the MMOG requiring it. The variance in times when data become available should also be small.

Security — The storing system should store data securely. It should not be possible for data to be altered in ways that are inconsistent with the game rules. It should also be possible to identify nodes that alter the data in this malicious way. This also adds the requirement that nodes should be authenticated in the storage system and that only authorised nodes should be able to alter data.

Fairness — Ensuring fairness in the system means distributing load evenly according to the abilities of individual nodes. This ensures that not only a small number of nodes provide all system resources required for the system to function, but that all nodes contribute what they can, in order to support the system.

Consistency — The consistency requirement specifies that nodes should perceive the game world as the same. This is intentionally a vague requirement, as it is believed that the stricter requirement of the game world actually being the same is too strong. When two players are playing together, they should perceive the world exactly the same way. If, for example, one player perceives a monster and starts to attack it, while the other player sees nothing, the game experience of both will degrade. As another example, if there is a damsel in distress in a tower for a player that visits the tower, but no damsel for a player that does not enter the tower, the game experience would not degrade. This is because the player not in the tower does not have to know about the damsel. From these two examples, the reason for state persistency based on perception, rather than reality is shown.

All state persistency models will be reviewed with these issues in mind.

B. Overview of approaches

Very little work has been done on state persistency for P2P MMOGs. Generally three approaches have been identified for state persistency: super peer storage [21], overlay storage [70], [71], [2], [29] and distance-based storage [38], [37], [72]. Hybrid techniques, distinguishing permanent data from ephemeral data have also been proposed [73], [74].

Storage type	Reliability	Responsiveness	Security	Fairness	Consistency
Region-based Super Peer	Medium	High	Low	Low	High
Overlay	High	Low	Medium	High	Low
Region-based Super Peer/Overlay	High	High	Low	Low	High
Distance-based	Low	High	Low	High	High

TABLE II
DIFFERENCES BETWEEN STORAGE MECHANISMS

Table II presents a summary of how current storage systems handle the five key issues identified. From this table it can be seen that no one storage mechanism has fully addressed all the identified issues. What is also shown is that architectures that differentiate between different types of data, are theoretically better suited to the MMOG application. The example of such an architecture is the hybrid architecture, which also seems to fare best, when compared to other storage techniques.

C. Super peer storage

Super peer storage relies on the super peer storing all relevant information in its domain. An example of this is in [21], where the world is segmented into regions and super peers act as regional servers to all peers in their region. Each super peer handles all game logic and distributes updates to all peers in its region. The super peer also handles state persistency for its region, hosting NPCs, objects and persistent player data.

The consistency model for this approach is depicted in Figure 5. One can see that this approach is modelled on the update based model, but segmented into separate regions. The role of the server is here fulfilled by a super peer, which is a peer that is selected in some logical way, from the available set of peers and then promoted. Server selection in itself is a complex topic that has to deal with determining whether a peer has sufficient resources available and also whether the peer is trustworthy.

Each super peer in this model houses the complete region state as shown. Super peers also house the real game logic. Clients in the region only house copies of the regional objects and some client logic to update the local copies of objects housed. Like the C/S model, clients only send events to super peers, where super peers apply the game logic and send state updates to clients.

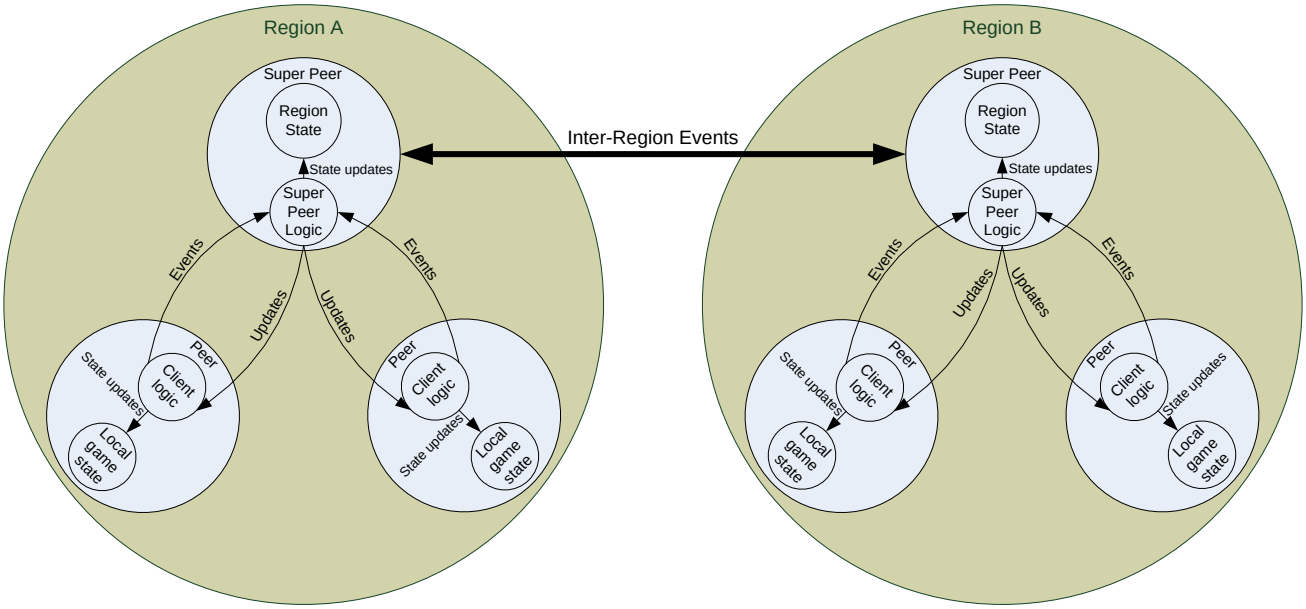


Fig. 5. Region-based Client/Server consistency model

1) Fairness: The super peer storage model has many potential issues. Overloading of the super peer is one. A super peer could be relatively easily overloaded if a region becomes too crowded, since a super peer is merely the computer of some player in the game and not a specialised server machine. The question of fairness also arises. The idea of a P2P MMOG model is that all peers share resources. With this model, peers with extra resources are expected to donate these resources for the good of all. Players might consider it unfair, when they are constantly expected to donate resources, some of which they might have to pay for.

2) Reliability: Another issue is reliability. In a P2P system, with a high rate of churn, players are expected to constantly leave and join the network. Because of this reality, redundancy mechanisms have to be developed that would ensure state data are always available, even when a super node leaves the network. It is possible to solve these issues by having redundant super peers in each region, that take over hosting responsibility if the main super peer leaves. One method by which redundant region coordinators are maintained in [21], is to create backup coordinators on peers with IDs closest to the current coordinator. This means that if the main coordinator fails, all data will automatically be routed to the backup, because of the feature of DHTs. It is important that the main and backup super peers always posses consistent states, even during a transition from main to backup. Other schemes to support improved reliability deal with reputation mechanisms for super peers. Super peers that have more resources and stay in the network longer are preferred during super peer selection, using reputation mechanisms [41].

3) Security: The third, and probably most important issue is that of security. If a single peer is allowed to house the player information of a large group of players, it might become possible for such a peer to modify the data to suit his own ends. The issue is not only that modification of the data might be possible, but also that it would not be possible for the cheating to be detected, because of no centralised logging. A scheme that would improve the reliability of this systems has been proposed, where every event is also sent to the backup super peer of the region [29]. The main super peer responds with the update and the backup super peer responds with a hash of the update. A peer can then check whether the hashes match to determine whether the data has been received correctly. A hash is not the state update itself, so will be much smaller, but the events that have to be sent to all super peers will increase traffic in the network and bandwidth usage by peers.

4) Responsiveness: There are, however, also advantages to super peer storage. All data are stored on the super peer, which means that storing data is a low latency operation. The regional state can be stored and retrieved at very high speeds, making the system very responsive. Data retrieval from such a storage is also relatively fast; as fast as data retrieval from a server. Peers can request data from a super peer and the data can be returned to the peer in one hop after transmission of the request. Super peers may, however, become overloaded with requests and thereby increase the latency of the system.

5) Consistency: Super peer storage is always consistent, because event ordering is handled by a centralised super peer for each region. If events occur across region boundaries, the super peer of the region in which the event was triggered will send a message to the super peer in the region affected by the event. This is also an event that can be ordered, as the one super peer is handled as a normal peer for the consistency calculation.

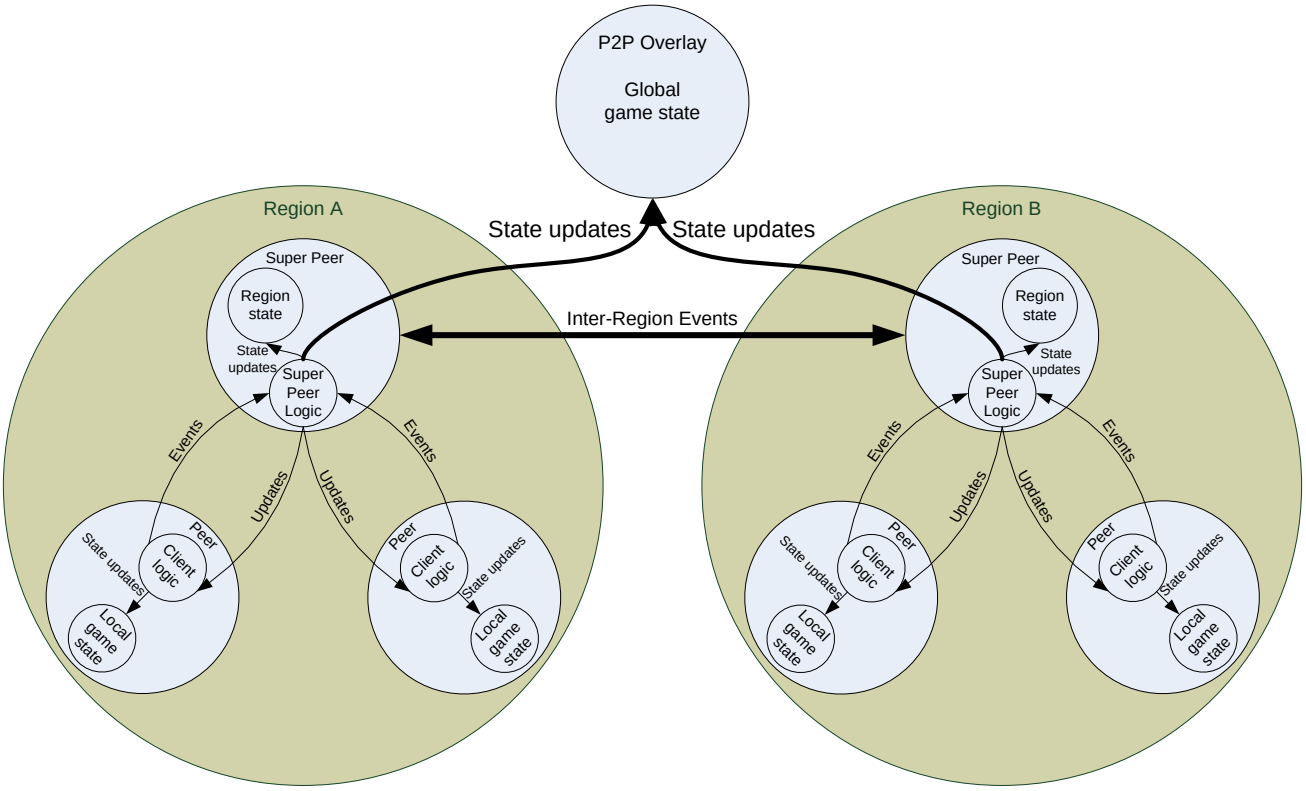


Fig. 6. Region-based Client/Server with overlay consistency model

D. Overlay storage

Overlay storage, shown in Figure 6 entails using a distributed file storage system, which is based on a P2P overlay to host objects. Figure 6 actually shows a type of Super Peer/Overlay hybrid storage implemented in [73], but the basic principles remain the same. The model depicted in Figure 6, uses an overlay storage managed by regional super peers. The reason for this management is to achieve state consistency. Although game state is always consistent in overlay storage, because of the inherent consistency of DHTs, issues of serialisability can still occur [73]. In order for game states to remain consistent, events should be serialisable. In the context of state consistency, this means that the order in which events are received at any peer, should be the same. This cannot be ensured in this case, because there is no central entity that orders events and an update-based model is not used. If the order in which events are processed are not the same, state inconsistencies may arise.

The world is divided into regions and so data stored for one region will have no effect on the sate of other regions. This can of course be debated, but for global specific data a global "region" can be created. Super peers then act as servers that use their section of the overlay as their own storage databases. Data consistency may be assured as all access to a region's data is controlled by that region's super peer. The Zoned Federation model further uses local caching at super peers to achieve real-time access to data. Overlay storage is used as a backup mechanism to achieve data persistency and redundancy.

Overlay storage is a very popular storage method, currently used by most P2P MMOG architectures. This is believed to be more as a consequence of the use of Scribe than any inherit benefit to P2P MMOGs [29]. This is also the motivation used in Chapter 4 of [2], where the Mediator architecture is described. Scribe is an implementation of ALM that uses the Pastry overlay as the structure to send multicast messages over [27]. PAST is a distributed persistent storage implemented to use Pastry for routing data and requests for data. The reason for using overlay storage in so many implementations seem to be merely the availability after using Scribe. The implementation of state persistency, however, does not have to be linked with the event dissemination scheme. A P2P overlay may be used for event dissemination, and another method can be used to ensure state persistency.

1) Responsiveness: An evaluation of overlay storage also shows some issues when used for state persistency in MMOGs. The main issue with this mode of storage is summed up by the creators of PAST: "Finally, PAST is intended as an archival storage and content distribution utility and not as a general-purpose filesystem. It is assumed that users interact primarily with a conventional filesystem, which acts as a local cache for files stored in PAST." [75]. This is not how the file system interaction occurs in MMOGs. For responsive MMOGs, a distributed file system is required that allows for real-time file storage and retrieval.

The most significant issue with overlay storage is the delay incurred when storing and retrieving data. As data can be stored

anywhere on the network and the network is not fully connected, it takes $O(\log_{2^4}(N))$ hops on average to retrieve or store a data item using Pastry [75]. Although this is a good order complexity for a routing algorithm in a large network, it is not sufficient to support a real-time application. This latency issue is the same issue that is present in ALM as discussed in Section V.

2) Consistency: Different types of overlays have different advantages and disadvantages. A pure overlay-based storage scheme performs very badly. The scheme is inconsistent, because there is no way to synchronise the order of events amongst nodes, which leads to inconsistent states.

3) Security: This model does have better security than the super peer storage model as data are distributed amongst all peers and redundancy and quorum techniques can be implemented to ensure that files are retrieved with a high level of security. Why the security is rated as average is not because a real lack of security, but because of the network overhead, which a secure system introduces.

To ensure a secure system, copies of files have to be saved at different locations. If a file is retrieved, all copies must be queried and received. All received copies then have to be compared to ensure that the contents are correct. This introduces significant additional network overhead as well as additional load on nodes to serve as copies of files.

4) Fairness: Pure overlay storage is very fair, as all nodes share file data and requests equally. Is is also reliable when adequate redundancy is built into the system.

5) Reliability: Overlay storage can also be made very reliable, using redundancy. One method used to achieve high reliability is to duplicate a data item and store the duplicate at the neighbour of the node where the original item is stored. The neighbour is the node whose ID is closest to the node where the original data are stored. By the characteristics of DHT distance-based routing, if the node with the original data leaves the network, packets will automatically be routed to the neighbouring node, where the duplicate will then be. This technique ensures high availability of data and the number of duplicates can be chosen according to the reliability of the network.

6) Hybrid region-based: The hybrid region-based overlay storage contains many improvements over pure overlay storage. It is just as reliable as pure overlay storage, but that is where the similarities end. Because all regional files are cached at super peers, the system is very responsive. The use of super peers also allows for strict event ordering to be implemented, which ensures data consistency. The only two issues are fairness and security, which are the same as for the super peer storage model.

E. Distance-based storage

Distance based approaches, such as the Voronoi storage approaches [38], [37] and also some more general approaches [72], store object data on the peer closest to the object in the virtual world. Some distance metric is used to determine on which node an object should be stored. For the Voronoi approaches, a node controls and hosts all objects within its Voronoi region. The reasoning is that there is a high probability that the player closest to the object is also the player using the object. Examples of this are where a player is trading or fighting with an NPC. The only problem with this reasoning is that usually multiple nodes are interacting with a single object. The examples of the NPC monster and trader are again relevant. Usually many players interact with a trader NPC and usually players attack monster NPCs in groups.

1) Responsiveness: Multiple player interaction is, however, not really an issue as others have stated [31]. In the best case, the object being used by a player is also hosted on that player's node. If another player requires use of a remotely hosted object, that player may still interact with the object, where the host node is now acting as a server to that player. This means that every player hosting an object becomes a server for that object. For the case where a player interacts with an object hosted locally, there is no object latency. In the case where a player accesses a remotely hosted object, there is only one hop latency, the same as with a client server application. The advantage however is that the total server load for all objects is distributed amongst all peer, which means that each peer should have to handle much less queries than with the super peer storage approach. This might protect peers from becoming overloaded and improve latency.

Issues with this approach stem from the fact the players are constantly moving. When players move, the objects in their regions change. Objects, therefore, have to be constantly handed over from one peer to another, which might cause significant network traffic. An object in transit might also delay interaction with that object.

2) Reliability: Reliability is also an issue, because of network churn. When nodes leave the network, the objects that they controlled should still be accessible. Redundancy and added overlay storage can be implemented here to ensure reliability.

3) Security: The main issue with the distance based scheme is security. Nodes that have the most interest in an object also have the most interest to manipulate that object in ways inconsistent with the game rules. When objects are hosted on nodes that have the most interest in them, there will be a strong drive to try and manipulate these objects. Because these modification are all local, it is also not possible to log the alterations and detect cheating. Means by which local objects can be secured have to be found or distance based algorithms with quorum need to be investigated.

4) Fairness: Distance-based storage is relatively fair as all objects are distributed amongst all nodes. Where the system becomes somewhat unfair is when a node is nearest to a large number of objects. For Voronoi regioning approaches, this is when a peer's Voronoi region contains many more objects than the average number of objects hosted by other peers. This will

not be a major issue, depending on how long the objects have to be hosted on the overloaded peer. This will depend on the movement of the overloaded peer as well as that of neighbouring peers.

5) Consistency: As with super peer storage, peers using distance based storage act as the hosts of the objects in their area. This means that all events affecting a certain object should be sent to that object's host. This ensures that for any object, there is only one authoritative host, which ensures serialisability. Issues may, however, occur when an object is being transferred from one peer to another.

IX. Proposed consistency model

A. Model design

A novel state persistency architecture is proposed in this work, specifically designed to meet the real-time requirements of P2P MMOGs.

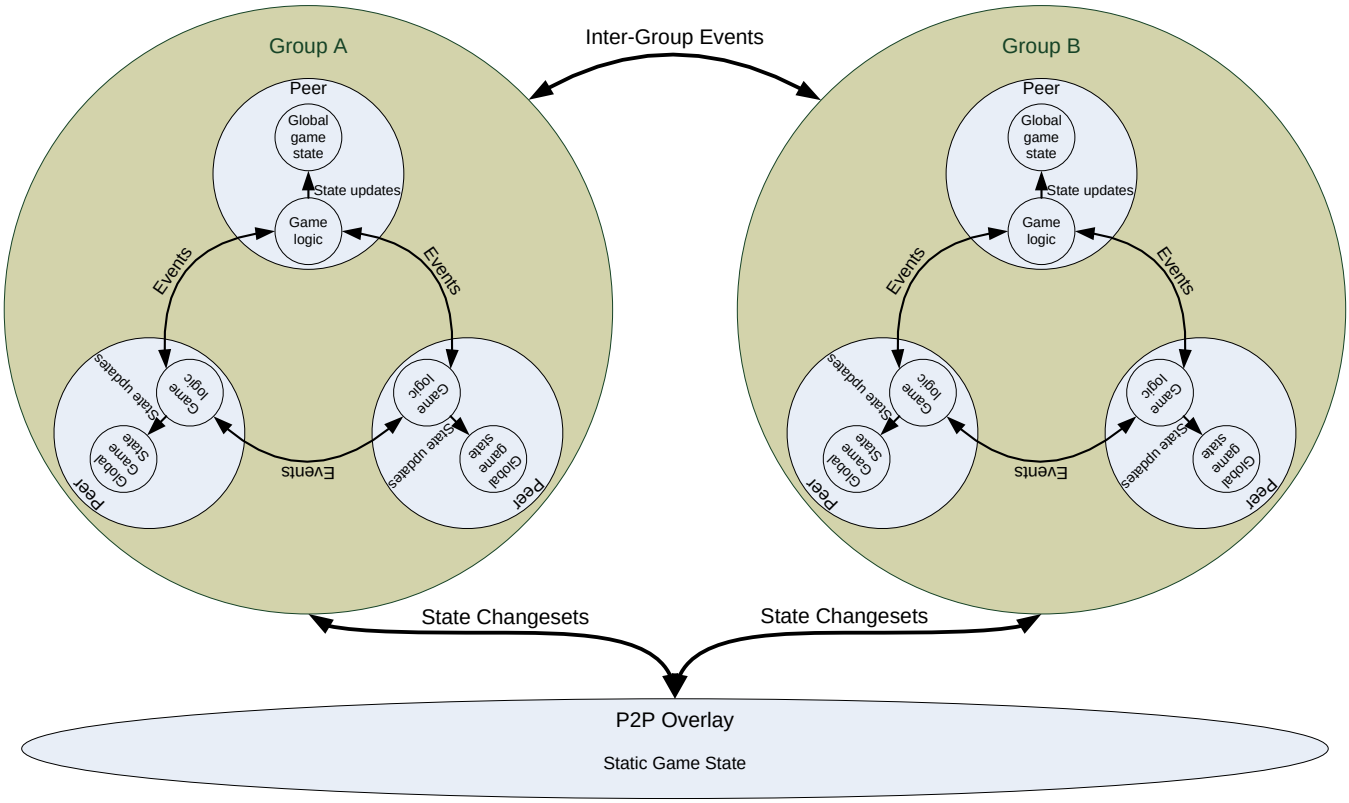


Fig. 7. Group-based Peer-to-Peer with overlay consistency model

Figure 7 shows the initial proposed state consistency model for an MMOG. As described in Section V, this persistency model is based on groups of players, rather than regions. An architecture is proposed that uses a hybrid model of storage, similar to the Zoned Federation model proposed in [73].

A distinction will be made between ephemeral and permanent data. Where ephemeral data is considered data that is only valid for a short time. An example of this is player movement, which changes frequently, so long term storage is unnecessary. Permanent data are data that remains constant for a significant amount of time. An example if this would be a player's attributes or inventory contents.

All players will form part of a specific group. The hierarchical state persistency model on one level operates within groups and on the higher level, amongst groups. The proposal is that ephemeral data will only exist within groups, with a high level of responsiveness. Unicast connections will be used amongst groups that would enable fast retrieval of ephemeral data such as player position updates. It is also proposed that a fully connected P2P network model is maintained within groups of players. This would allow for high responsiveness.

Ephemeral player data might be stored on a second level overlay storage, which only exists amongst the members of every group. Group data can thus be distributed amongst players to achieve maximum fairness. It is also proposed that no central peer be used to apply game logic to peer events as is the case with super peers.

It should be explored whether an event-based or update-based consistency model should be used within groups. If groups sizes could be bound, the disadvantages named in Section VII-A will not be an issue. This would allow for all peers to apply game logic and for validity checking to be done on updates sent from peers. This would increase the security of the system

and would prevent any malicious super peer from hijacking the game. How a limited group size will affect the game will be investigated to determine whether it is feasible.

It is then proposed that the overlay acts as a slow, but reliable storage medium. With data being constantly backed up to this medium. If data are lost before it could be backed up, the game should still be able to return to a previously stable state for the players that logged out, after logging back in.

The main purpose of the proposed consistency model is to develop a low latency, highly reliable, consistent, fair and secure storage network. It is proposed that the use of super peers be avoided as much as possible, in order to obtain all benefits of a P2P model.

B. Testing and evaluation

1) Evaluation Criteria: In order to evaluate any consistency model, metrics have to be defined to measure the key consistency challenges, as described in Section VIII-A.

To evaluate *consistency* is to evaluate how well the game state remains consistent between peers. For the game state to be consistent, all game objects should be consistent. What will be measured is the differences between root and replica objects. Root objects are those objects that represent part of the global game state and replica objects are local objects used to display the game state at the client side. When an update is sent to a root object, the changes to the root object should be reflected in all replica objects. The appearance of a root object should also be the same for all peers. Any peer querying a root object should receive the same object. This can be tested by querying root objects and updating these objects as they are queried. What should be tested is how long it takes for an object's appearance to update, after an update has been submitted by some node.

To measure *fairness*, the distribution of game state should be measured. This can be measured at a file level, i.e. what is the variance of the number of files contained on each node, or on byte level, i.e. what is the variance of the number of bytes stored on each node. A lower variance will point to a fairer data persistency scheme.

Reliability encompasses both robustness and availability. Robustness means that the data should be resilient to nodes leaving the network and availability means that data should be available to any node in the network, with the correct permissions. To measure robustness, nodes have to leave the network at different rates and the loss of storage for different rates of churn have to be measured. It can then be determined what the churn threshold is for the network to maintain all data at a certain rate of churn. This will be a function of the number of redundant objects in the storage system.

Responsiveness can be measured by the time it takes for an object to be available for reading, anywhere in the network, after having been written. How long it takes to read or write data to the storage network can also be measured.

Security is the combination of a number of objectives as described in Section V-B. These are: Authentication, Authorisation, Data Integrity, Confidentiality, Availability, Trust, Privacy and Identity Management. Initially, a focus will be placed on data integrity as a failure here will result in the game data being directly compromised. Data integrity can be tested by ensuring the storage system can withstand the dropping, delay or, possibly, the modification of data packets by malicious nodes in the network. Nodes can be programmed to randomly drop packets and the availability of the data under these circumstances can be measured by the time it takes to rebuild a file under different rates of packet drops or delays.

2) Testing methodology: Initially an evaluation by simulation approach will be followed for the proposed consistency model. For this evaluation there are a three possible options which should be investigated. The first option is to use a generic Discrete Event Simulator (DES), such as DESMO-J, the second option is to use a network simulator, such as OMNeT++ or ns-2, and the third option is to use a higher level simulator running on an existing network simulator, such as OverSim [76].

If the consistency model is not too complex for simulation, a DES can be used to simulate the model from scratch. DESMO-J, for example, is a Java based simulator that provides the tester with basic queues and probability distributions and allows her to create objects. DESMO-J also tracks the sizes of all queues over time and is capable of producing graphs of queue lengths over time as well as a detailed report. Java objects are created in DESMO-J, each with a particular behaviour. This could entail sampling from a probability distribution to simulate processing time, adding itself or other objects onto queues or producing other objects. Objects can also easily communicate using the simulation environment.

This environment allows for the creation objects with specific behaviour. Objects can be made to model anything in the virtual world. It is easy to learn the environment and one can quickly start to produce results. It does however lack the depth and extensive library support of the network simulators. If no network specific protocols need to be simulated for the consistency model, DESMO-J might be the best choice.

Network simulators posses large libraries that allows a tester to easily simulate a complete protocol stack. It would then be possible to implement the consistency model in the application layer and simulate the functionality of the system from the ground up. This will allow for all network parameters to be taken into account for the simulation. The question is whether it is necessary to take into account all these parameters.

Another strength of the more well known open network simulators is that users have implemented their own simulators on top of these network simulators and made these simulators available to the public. OverSim, which is based on OMNeT, allows for the simulation of an overlay network in the network simulator. It is a working simulation of an overlay network

that allows users to add nodes with certain behaviours into this overlay simulation. This would allow for the simulation of the consistency model on an overlay and could greatly decrease the time required to set up the simulation. The viability of this solution will depend on the availability of features to support the simulation of the consistency models.

Grouping algorithms first have to be investigated on which the consistency model and network architecture will be built. A consistency model will then be developed using the grouping algorithm. This model will then be tested using the metrics identified in Section IX-B1. Other consistency models will also be simulated in order to compare their performances with the proposed model. Player movement data is required for this simulation. If possible, real game traces will be obtained and used to test the consistency model performance with actual players movements. If this is not possible, player movement would have to be simulated, which could become complex as the concept of player flocking should also then be simulated.

X. Related consistency models

There have been very few papers that deal directly with state persistency in MMOGs. Most papers merely assume that reliable consistent storage is available and sometimes go as far as to propose PAST to be used with Scribe. State persistency has thus far been regarded as a basis to be used to build other MMOG functions on. The few papers that have addresses state persistency directly have been discussed in Section VIII.

The existing model that is nearest to the proposed state persistency model would be the Zoned Federation model [73]. The main difference between the Zoned Federation model and the proposed model is that the Zoned Federation model uses super peers for data caching, instead of all nodes in the network. This use of super peers has all the issues of super peer storage, discussed in Section VIII. It is believed that a fair, more secure approach, where all users contribute in accordance with their means, should rather be investigated. This and grouping are exactly the novelty of the proposed persistency model.

XI. Proposed evaluation techniques

What became evident while performing the literature study, was that there are no quantitative studies, comparing different P2P MMOG architectures or P2P MMOG architectures to C/S MMOG architectures. The reason for this is believed to be the incompleteness of most P2P MMOG architectures and also the difficulties with a test setup.

The difficulties with a test setup that would compare P2P to C/S is that the same application layer data should be compared. In other words, the same game should be used to test both architectures. The reason for this being that different games have implemented different networking architectures, as well as different communications protocols. The sizes and number of packets will differ. Some games may have implemented a more efficient communications protocol with less network overhead. To be able to accurately test the two architectures then, the same communications protocol should execute over these architectures. The main issue with this is that communications protocols differ from C/S to P2P. As each of these architectures have different advantages and disadvantages, communication protocols executing over them will have to implement different mechanisms to cope with each architecture. The security mechanisms, for example, will look very different from one architecture to the other.

In an effort to compare P2P to C/S, a test setup is proposed that would theoretically allow the two architectures to be used for the same game. It is proposed that an existing open source C/S MMOG server be adopted for P2P. Every player will have both a client and server installed on her computer. The existing game client will connect to the modified game server on the same host. The game servers on all hosts will then form a P2P network and function as the P2P MMOG.

This approach will allow for the same game to be compared with the two different architectures. It will also allow for comparison between different P2P architectures if the server alterations are implemented as an extra middleware layer, with which architectures may interact over a standard interface.

XII. Conclusion

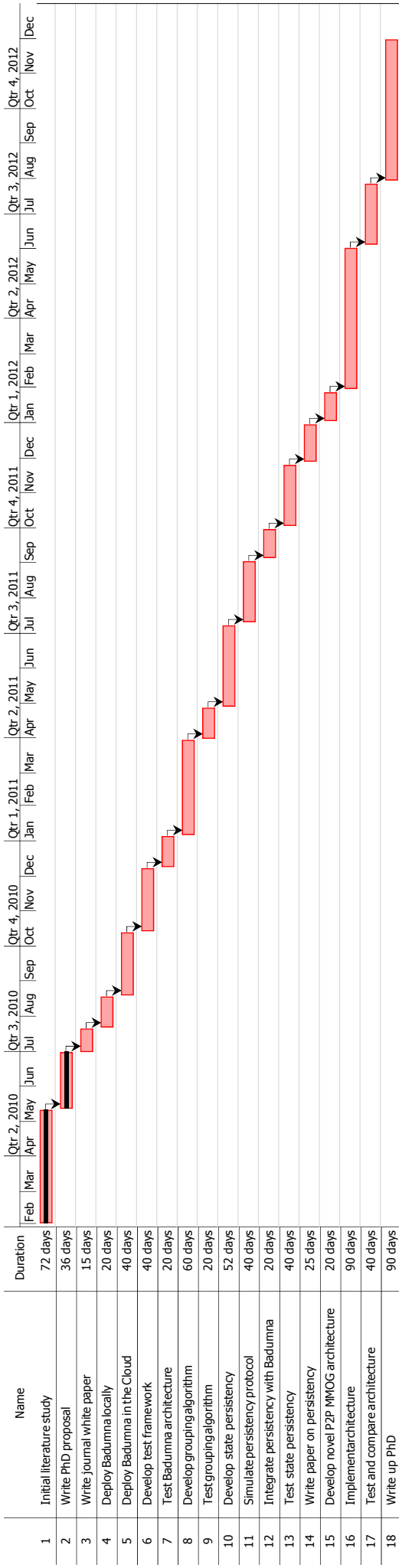
This proposal presents an extensive literature study into what the state of the art is of P2P MMOGs. The P2P MMOG architecture is compared to more classic architectures, both in terms of the overall network architecture, as well as in terms of the consistency models employed. After a thorough investigation into the issues of P2P MMOG, it was determined that state persistency and consistency are areas that still require much attention. With this in mind a novel networking architecture as well as a consistency model were proposed that hopes to address some of the issues related to P2P MMOGs.

The state consistency model distinguishes between ephemeral and persistency game state and proposes a hybrid state consistency model that employs overlay storage techniques as backups to achieve low latency state persistency.

Finally, possible testing techniques and a testing framework were presented. The framework will be used to test all contributions and enable a rigourous evaluation of all contributions.

A Gantt Chart for the project is provided in the appendix, to provide an estimate of the different activities and the timescale of each. This will be reviewed throughout the lifetime of the project, to ensure that it remains possible to complete the project at the end of 2012.

Appendix



	Name	Duration
1	Initial literature study	72 days
2	Write PhD proposal	36 days
3	Write journal white paper	15 days
4	Deploy Badumna locally	20 days
5	Deploy Badumna in the Cloud	40 days
6	Develop test framework	40 days
7	Test Badumna architecture	20 days
8	Develop grouping algorithm	60 days
9	Test grouping algorithm	20 days
10	Develop state persistency	52 days
11	Simulate persistency protocol	40 days
12	Integrate persistency with Badumna	20 days
13	Test state persistency	40 days
14	Write paper on persistency	25 days
15	Develop novel P2P MMOG architecture	20 days
16	Implementarchitecture	90 days
17	Test and compare architecture	40 days
18	Write up PhD	90 days

REFERENCES

[1] B. S. Woodcock. (2008) Total MMOG active subscriptions. mmogchart.com. [Online]. Available: http://www.mmogchart.com/Chart4.html
[2] L. Fan, "Solving key design issues for massively multiplayer online games on peer-to-peer architectures," Ph.D. dissertation, School of Mathematical and Computer Sciences – Heriot-Watt University, 2009.
[3] L. Achterbosch, R. Pierce, and G. Simmons, "Massively multiplayer online role-playing games: the past, present, and future," *Comput. Entertain.*, vol. 5, no. 4, pp. 1–33, 2007.
[4] R. Bartle, *Designing Virtual Worlds*. New Riders Games, 2003.
[5] R. Bartle, "Hearts, clubs, diamonds, spades: Players who suit MUDs," *Journal of MUD Research*, vol. 1, pp. 1–25, 1996.
[6] Medar. (2004, October) The history of Neverwinter Nights. [Online]. Available: http://www.bladekeep.com/nwn/index2.htm
[7] A. Kirmse. (2000, May) History of Meridian 59, 1994-2000. [Online]. Available: http://sites.google.com/site/meridian59history/
[8] B. S. Woodcock. (2008) MMOG active subscriptions 200,000+. mmogchart.com. [Online]. Available: http://www.mmogchart.com/Chart1.html
[9] T. Fritsch, H. Ritter, and J. Schiller, "The effect of latency and network limitations on MMORPGs: a field study of Everquest 2," in *NetGames*, 2005, pp. 1–9.
[10] M. Bergsson and J. Alberni. (2006) Eve online launches largest supercomputer in the gaming industry running on IBM server technology. CCP Games and IBM. [Online]. Available: http://www.eveonline.com/pressreleases/default.asp?pressReleaseID=25
[11] CCP Shadow. (2010, June) 60,453 pilots: the new Eve PCU record. CCP Games. [Online]. Available: http://www.eveonline.com/news.asp?a=single&nid=3934&tid=1
[12] V. Massey. (2007, June) Eve online appoints in-world economist. CCP Games. [Online]. Available: http://www.eveonline.com/pressreleases/default.asp?pressReleaseID=34
[13] E. Guōmundsson (edt.). (2010, February) Eve Online fourth quarter economic newsletter for 2009. CCP Games. [Online]. Available: http://ccp.vo.llnwd.net/o2/community/QEN/QEN_Q4-2009.pdf
[14] B. Entertainment. (2008, December) World of Warcraft subscriber base reaches 11.5 million worldwide. Blizzard Entertainment. [Online]. Available: http://us.blizzard.com/en-us/company/press/pressreleases.html?081121
[15] B. S. Woodcock. (2008, April) MMOG subscriptions market share. mmogchart.com. [Online]. Available: http://www.mmogchart.com/Chart7.html
[16] B. S. Woodcock. (2008) An analysis of MMOG subscription growth version 23.0. mmogchart.com. [Online]. Available: http://www.mmogchart.com/analysis-and-conclusions/
[17] N. Ducheneaut, N. Yee, E. Nickell, and R. J. Moore, "Building an MMO with mass appeal: A look at gameplay in World of Warcraft," *Games and Culture*, vol. 1, pp. 281–317, 2006.
[18] J. Dibbell. (2007, June) The life of the Chinese gold farmer. New York Times. [Online]. Available: http://www.nytimes.com/2007/06/17/magazine/17lootfarmers-t.html?ei=5090&en=1676d344608cb590&ex=1339732800
[19] J. Kesselman, "Server architectures for massivly multiplayer online games," in *JavaOne conference – Session TS-1351*. Sun Microsystems, 2004.
[20] D. James and G. Walton, "2004 persistent worlds whitepaper," IGDA Online Games SIG, White Paper, 2004.
[21] B. Knutsson, H. Lu, W. Xu, and B. Hopkins, "Peer-to-peer support for massively multiplayer games," in *INFOCOM*, vol. 1, 2004, p. 107.
[22] D. Doval and D. O'Mahony, "Overlay networks: A scalable alternative for P2P," *Internet Computing, IEEE*, vol. 7, no. 4, pp. 79 – 82, 2003.
[23] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, "A scalable content-addressable network," in *SIGCOMM*, 2001, pp. 161–172.
[24] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for internet applications," *SIGCOMM Comput. Commun. Rev.*, vol. 31, no. 4, pp. 149–160, 2001.
[25] B. Y. Zhao, J. D. Kubiatowicz, and A. D. Joseph, "Tapestry: An infrastructure for fault-tolerant wide-area location and routing," University of California at Berkeley, Tech. Rep., 2001.
[26] A. Rawstron and P. Druschel, "Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems," in *Proc. of the 18th IFIP/ACM International Conference on Distributed Systems Platforms (Middleware 2001)*, 2001.
[27] M. Castro, P. Druschel, A.-M. Kermarrec, and A. Rowstron, "Scribe: a large-scale and decentralized application-level multicast infrastructure," *Selected Areas in Communications, IEEE Journal on*, vol. 20, no. 8, pp. 1489 – 1499, 2002.
[28] W. Ye, A. Khan, and E. Kendall, "Cdht: the design of a clustered dht routing scheme for serverless (p2p) networks," in *ICON*, vol. 1, 2004, pp. 351 – 356 vol.1.
[29] T. Hampel, T. Bopp, and R. Hinn, "A peer-to-peer architecture for massive multiplayer online games," in *NetGames*, 2006, p. 48.
[30] *Secure Hash Signature Standard*, National Institute of Standards and Technology Std., August 2002. [Online]. Available: http://csrc.nist.gov/publications/fips/fips180-2/fips180-2withchangenotice.pdf
[31] L. Fan, P. Trinder, and H. Taylor, "Design issues for peer-to-peer massively multiplayer online games," *Int. J. Adv. Media Commun.*, vol. 4, no. 2, pp. 108–125, 2010.
[32] K. L. Morse, L. Bic, and M. Dillencourt, "Interest management in large-scale virtual environments," *Presence: Teleoperators and Virtual Environments*, vol. 9, pp. 52–68, 2000.
[33] L. Wang, S. Turner, and F. Wang, "Interest management in agent-based distributed simulations," in *Distributed Simulation and Real-Time Applications, 2003. Proceedings. Seventh IEEE International Symposium on*, 2003, pp. 20 – 27.
[34] J.-S. Boulanger, J. Kienzle, and C. Verbrugge, "Comparing interest management algorithms for massively multiplayer games," in *NetGames*, 2006, p. 6.
[35] S. Benford and L. Fahlén, "A spatial model of interaction in large virtual environments," in *ECSCW*, 1993, pp. 109–124.
[36] A. R. Bharambe, S. Rao, and S. Seshan, "Mercury: a scalable publish-subscribe system for internet games," in *NetGames*, 2002, pp. 3–9.
[37] S.-Y. Hu, S.-C. Chang, and J.-R. Jiang, "Voronoi state management for peer-to-peer massively multiplayer online games," in *CCNC*, 2008, pp. 1134 –1138.
[38] E. Buyukkaya and M. Abdallah, "Data management in Voronoi-based P2P gaming," in *CCNC*, 2008, pp. 1050 –1053.
[39] K. Pan, W. Cai, X. Tang, S. Zhou, and S. J. Turner, "A hybrid interest management mechanism for peer-to-peer networked virtual environments," in *IPDPS*, 2010, pp. 1 –12.
[40] A. P. Yu and S. T. Vuong, "MOPAR: a mobile peer-to-peer overlay architecture for interest management of massively multiplayer online games," in *NOSSDAV*, 2005, pp. 99–104.
[41] L. Fan, H. Taylor, and P. Trinder, "Mediator: a design framework for p2p mmogs," in *NetGames*, 2007, pp. 43–48.
[42] S. Fiedler, M. Wallner, and M. Weber, "A communication architecture for massive multiplayer games," in *NetGames*, 2002, pp. 14–22.
[43] C. Diot, B. Levine, B. Lyles, H. Kassem, and D. Balensiefen, "Deployment issues for the IP multicast service and architecture," *Network, IEEE*, vol. 14, no. 1, pp. 78–88, 2000.
[44] C. Neumann, N. Prigent, M. Varvello, and K. Suh, "Challenges in peer-to-peer gaming," *SIGCOMM Comput. Commun. Rev.*, vol. 37, no. 1, pp. 79–82, 2007.
[45] C. GauthierDickey, D. Zappala, V. Lo, and J. Marr, "Low latency and cheat-proof event ordering for peer-to-peer games," in *NOSSDAV*, 2004, pp. 134–139.
[46] S. D. Webb, S. Soh, and W. Lau, "RACS: A referee anti-cheat scheme for P2P gaming," in *NOSSDAV*, 2007, pp. 34–42.
[47] A. Belapurkar, A. Chakrabarti, H. Ponnapalli, N. Varadarajan, S. Padmanabhuni, and S. Sundarrajan, *Distributed Systems Security: Issues, Processes and Solutions*. Wiley, 2009.

[48] S. D. Webb and S. Soh, "A survey on network game cheats and P2P solutions," *Australian Journal of Intelligent Information Processing Systems*, vol. 9, no. 4, pp. 34–43, 2007.
[49] D. S. Wallach, *Software Security – Theories and Systems*. Springer, 2003, vol. 2609/2003, ch. A Survey of Peer-to-Peer Security Issues, pp. 253–258.
[50] B. Cohen, "Incentives build robustness in bittorrent," in *1st Workshop on Economics of Peer-to-Peer Systems*, 2003.
[51] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina, "The eigentrust algorithm for reputation management in p2p networks," in *WWW*, 2003, pp. 640–651.
[52] G. Swamynathan, B. Y. Zhao, and K. C. Almeroth, "Exploring the feasibility of proactive reputations," *Concurr. Comput. : Pract. Exper.*, vol. 20, no. 2, pp. 155–166, 2008.
[53] G. Armitage, "An experimental estimation of latency sensitivity in multiplayer quake 3," in *ICON*, 2003, pp. 137 – 141.
[54] N. Sheldon, E. Girard, S. Borg, M. Claypool, and E. Agu, "The effect of latency on user performance in warcraft iii," in *NetGames*, 2003, pp. 3–14.
[55] J. Chen, B. Wu, M. Delap, B. Knutsson, H. Lu, and C. Amza, "Locality aware dynamic load management for massively multiplayer games," in *PPoPP*, 2005, pp. 289–300.
[56] S.-Y. Hu, J.-F. Chen, and T.-H. Chen, "Von: a scalable peer-to-peer network for virtual environments," *Network, IEEE*, vol. 20, no. 4, pp. 22 –31, 2006.
[57] S. Kulkarni, "Badumna network suite: A decentralized network engine for massively multiplayer online applications," in *P2P*, 2009, pp. 178 –183.
[58] S. Kulkarni. (2010) Badumna network suite showcase. Scalify. [Online]. Available: http://www.badumna.com/badumna/showcase.html
[59] P. Bettner and M. Terrano, "1500 archers on a 28.8: Network programming in Age of Empires and beyond," in *Proc. GDC2001*. Ensemble Studios, 2001.
[60] T. Sweeney, "Unreal networking architecture," Epic MegaGames, Tech. Rep., 1999. [Online]. Available: http://unreal.epicgames.com/Network.htm
[61] N. Baughman and B. Levine, "Cheat-proof playout for centralized and distributed online games," in *INFOCOM*, vol. 1, 2001, pp. 104 –113 vol.1.
[62] D. Kushner, "Engineering EverQuest: online gaming demands heavyweight data centers," *Spectrum, IEEE*, vol. 42, no. 7, pp. 34 – 39, 2005.
[63] E. Cronin, B. Filstrup, A. R. Kurc, and S. Jamin, "An efficient synchronization mechanism for mirrored game architectures," in *NetGames*, 2002, pp. 67–73.
[64] J. Müller and S. Gorlatch, "Rokkatan: scaling an RTS game design to the massively multiplayer realm," *Comput. Entertain.*, vol. 4, no. 3, p. 11, 2006.
[65] F. Lu, S. Parkin, and G. Morgan, "Load balancing for massively multiplayer online games," in *NetGames*, 2006, p. 1.
[66] J. C. S. Lui and M. F. Chan, "An efficient partitioning algorithm for distributed virtual environment systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 13, no. 3, pp. 193–211, 2002.
[67] P. Morillo, J. M. Ordua, M. Fernndez, and J. Duato, "An adaptive load balancing technique for distributed virtual environment systems," in *Proceedings of the 15th IASTED international PDCS-03*, 2003, pp. 256–261.
[68] M. Assiotis and V. Tzanov, "A distributed architecture for mmorpg," in *NetGames*, 2006, p. 4.
[69] R. Chertov and S. Fahmy, "Optimistic load balancing in a distributed virtual environment," in *NOSSDAV*, 2006, pp. 1–6.
[70] S. Douglas, E. Tanin, A. Harwood, and S. Karunasekera, "Enabling massively multi-player online gaming applications on a P2P architecture," in *ICIA*, 2005, pp. 7–12.
[71] M. Merabti and A. El Rhalibi, "Peer-to-peer architecture and protocol for a massively multiplayer online game," in *GlobeCom Workshops*, 2004, pp. 519 – 528.
[72] A. Bharambe, J. Pang, and S. Seshan, "Colyseus: A distributed architecture for online multiplayer games," in *Proc. of NSDI*, 2006.
[73] T. Iimura, H. Hazeyama, and Y. Kadobayashi, "Zoned federation of game servers: a peer-to-peer approach to scalable multi-player online games," in *NetGames*, 2004, pp. 116–120.
[74] C. Gauthier Dickey, D. Zappala, and V. Lo, "A fully distributed architecture for massively multiplayer online games," in *NetGames*, 2004, pp. 171–171.
[75] A. Rowstron and P. Druschel, "Storage management and caching in PAST, a large-scale, persistent peer-to-peer storage utility," *SIGOPS Oper. Syst. Rev.*, vol. 35, no. 5, pp. 188–201, 2001.
[76] I. Baumgart, B. Heep, and S. Krause, "OverSim: A Flexible Overlay Network Simulation Framework," in *GI in conjunction with INFOCOM*, 2007, pp. 79–84.