A Survey of State Persistency in Peer-to-Peer Massively Multiplayer Online Games

John S. Gilmore, *Student Member*, *IEEE*, and Herman A. Engelbrecht, *Member*, *IEEE*

Abstract—Recently, there has been significant research focus on Peer-to-Peer (P2P) Massively Multiplayer Online Games (MMOGs). A number of architectures have been presented in the literature to implement the P2P approach. One aspect that has not received sufficient attention in these architectures is state persistency in P2P MMOGs. This survey presents an overview of the current challenges present in P2P MMOGs, followed by an overview of classic state consistency models used in C/S MMOGs. The survey then classifies the state persistency techniques currently used in P2P MMOGs into super peer storage, overlay storage, hybrid storage, and distance-based storage. Key characteristics, namely scalability, fairness, reliability, responsiveness, and security are then defined. Each state persistency technique is evaluated according to these characteristics and recommendations are then made of possible future areas of research into the different storage types.

Index Terms—Distributed systems, distributed applications, internet applications, games.

1 Introduction

Peer-to-peer (P2P) Massively Multiplayer Online Games (MMOGs) have received significant attention from the research community, since the first publication on the subject by Knutssonn et al. in 2004 [1]. P2P MMOGs promise to solve many issues prevalent in today's Client/Server (C/S)-based MMOGs. Some key issues have to be solved before P2P MMOGs can be implemented commercially. Over the past few years, researchers have been addressing these challenges.

The focus of this paper is exclusively on one of the key challenges present in P2P MMOGs, namely: state persistency. The implementation of state persistency in P2P MMOGs allows for the storage of game data. The fact that game data must now be distributed among various peers in the network creates challenges not usually present in classic C/S MMOGs.

The paper classifies the techniques used in documented P2P MMOG architectures and discusses the advantages and disadvantages of the different types of storage. The paper also shows that no current storage type is well suited to P2P MMOGs. As the field of P2P MMOGs is a growing one, it is believed that a comprehensive survey of persistency techniques is required to act as a basis for further research into the field.

To the best of our knowledge, such a classification, where a focus is placed specifically on state persistency in P2P MMOGs, has not been undertaken in the literature. Other surveys, relating to P2P MMOGs and P2P overlays in general are discussed in Section 4. For each included survey, the differences between this paper and the included survey are discussed.

The remainder of this paper is structured as follows: the field of P2P MMOGs is introduced in Section 2, which contains an introduction to P2P overlays, the major advantages of P2P MMOGs and the key challenges that P2P MMOGs face. An introduction to some classic consistency models, currently used in computer games, are introduced in Section 3. Section 4 discusses some related surveys that have been completed in the field of P2P MMOGs and P2P applications in general. Section 5 contains an analysis of different types of distributed storage for MMOGs. The section defines characteristics against which all storage schemes are evaluated. The section concludes with recommendations to implementers as to the applicability of the different types of storage to different types of games. Section 6 concludes the paper by providing a brief summary and suggesting a number of areas for future work.

2 Peer-to-Peer MMOG Network Architectures

In 2004, an architecture using the peer-to-peer networking model to host MMOGs was proposed by Knutsson et al. [1]. This revealed a new research field, which attempts to establish the P2P model as a viable alternative to the classic C/S and Client/MultiServer (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 2.3, 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 host itself. This also means that all functions of the server in the classic C/S model are distributed among all peers. There are many areas where the P2P model can improve on the classic C/S mode, as will be discussed in Section 2.2.

• *The authors are with the MIH Media Laboratory, Department of Electrical and Electronic Engineering, Stellenbosch University, Electronic Engineering building, Cnr of Banhoek rd. and Joubert str., Stellenbosch 7600, South Africa. E-mail: jgilmore@ml.sun.ac.za, hebrecht@sun.ac.za.*

Manuscript received 11 Mar. 2011; revised 9 June 2011; accepted 10 July 2011; published online 21 July 2011.
Recommended for acceptance by J.C.S. Lui.
For information on obtaining reprints of this article, please send e-mail to: tpds@computer.org, and reference IEEECS Log Number TPDS-2011-03-0144.
Digital Object Identifier no. 10.1109/TPDS.2011.210.

2.1 Structured and Unstructured P2P Overlay Networks

P2P networks are created and maintained in the application layer of the Open Systems Interconnection (OSI) model protocol stack. This application layer network is called the P2P overlay. Peers in an overlay network might have neighbors that have no relationship to their physical position in the underlying network. Overlays can broadly be classified into structured and unstructured types. The classification is mostly based on the differing methods of routing and content retrieval in the network. This section only provides a brief comparison between structured and unstructured overlays. For a detailed comparison between the two types that also deals with many of the myths of structured overlays, please refer to [2].

With unstructured approaches, one is never assured that a data item will be retrieved, even if that data item is present in the network. If many duplicates of a data item are contained in the network, this becomes less of a problem, since it is assumed that the request will be routed to some set of nodes that do possess the item.

An unstructured architecture works well for content sharing and Voice over Internet Protocol (VoIP) networks, for example: P2P TV, BitTorrent, Gnutella, and Skype. The reason for this is the high level of duplication in these networks, especially for popular content. It is also easier to perform keyword searches in unstructured networks and the overlay requires less maintenance.

Because there is no assurance that a data item might be retrieved from an unstructured network, especially when that item is scarce, unstructured overlays are not considered adequate as a basis for P2P MMOGs, where all data items must be available at all times.

Structured overlays have been proposed that provide for efficient routing and reliable retrieval of data items. Some of these well known overlays are: CAN [3], Chord [4], Tapestry [5], and Pastry [6].

The basic idea of a structured overlay is that all nodes are identified by unique identifiers (IDs). A popular method to create the IDs is to use 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. An accurate comparison is that unstructured overlays are good at finding "hay," while structured overlays are good at finding "needles" [7].

2.2 Advantages

There are various advantages to moving from C/S to P2P in MMOGs. These include: increased robustness, improved scalability, lower operator costs, improved handling of transient player load, and lower latencies.

The P2P system is 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 behavior makes game down-time extremely unlikely.

Furthermore, because every peer hosts itself, the system is scalable. Another advantage is that no extra costs are incurred from an operator perspective, when more peers join 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, since peers already possess their required resources.

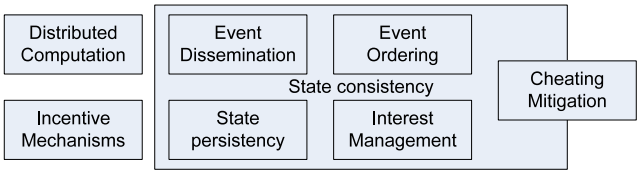

Fig. 1. VEN diagram showing the relationship between different characteristics.

P2P architectures also create a lot of opportunity for independent developers, because a large initial investment is no longer required to purchase the expensive server hardware. Not only are hardware costs reduced, but running costs are also reduced. The bandwidth required by the game server is now shared among users, which means that very little bandwidth costs will be incurred by the provider.

Latency is also improved, because it is now possible to directly communicate between peers and it is not necessary to communicate via a server. There is also no single server that has to process game events. Game events need only be processed by other peers who find the specific events of interest. The distribution of the load as well as direct communication will further reduce latency. Game events are defined in Section 3.1.

2.3 Key Challenges

A key challenge with any networked game is how to maintain state consistency between root and replica objects. A root object is the authoritive version of an object and the replica object is usually a local nonauthoritive version. Root objects are usually found on the server and replica objects are found in the local object cache of clients. The method by which the states between root and replica objects are updated is called the consistency model. Solving the state consistency problem for P2P MMOGs is one of the major development challenges.

A recent article has identified six key challenges of P2P systems: Interest Management, Game Event Dissemination, Nonplayer Character (NPC) Host Allocation, Game State Persistency, Cheating Mitigation, and Incentive Mechanisms [8]. A brief overview of these challenges will be introduced below.

As shown in Fig. 1, of the six challenges mentioned, Interest Management, Game Event Dissemination, and Game State Persistency all form part of State Consistency, with some aspects of Cheating Mitigation also a part of State Consistency. Also part of state consistency is event ordering, which deals with how to ensure that the system remains causal [9].

Another challenge for P2P systems is the required peer bandwidth. In a paper by Miller and Crowcroft, a packet simulator was created to determine the required bandwidth and effective latency, if a game such as World of Warcraft were to be implemented using P2P technologies [10]. Their simulation results indicate that today's networks are not able to host P2P MMOGs, with the required bandwidth and latency constraints. Such a significant result requires verification, but at the least, it shows that reducing

bandwidth and latencies for P2P MMOGs should be a primary design requirement.

It should be noted that the overview presented here is only an overview of the different techniques and general trends present in the different areas of peer-to-peer MMOGs. This overview does not presume to present an exhaustive list of papers in these areas, rather to place the topic of state persistency in context; to show readers how state persistency fits into the context of peer-to-peer games and to allow readers to distinguish between, for example, the topics of state persistency, interest management, and event dissemination.

2.3.1 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 players. In consistency terms, it provides a means to determine which replica objects require updates of the root object. The idea is not specific to P2P MMOGs and was already formally suggested in [11] and later with greater focus on a distributed environment in [12].

The main idea is that a player has a limited visual range and 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 in magnitude.

Extensive research has been done into solving AoI problems and a comparison of techniques can be found in [13] and [14].

2.3.2 Event Dissemination

Event dissemination deals with how information is sent to peers after interest management determines which information should be sent. In consistency terms, it determines how events and updates are distributed in the network. The first application of event dissemination for online games can be found in [15]. Recently, Application Layer Multicast (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 [16].

2.3.3 Cheating Mitigation

Cheating mitigation has been identified as a major issue for P2P systems [1], [17], [18]. The challenges reside in the fact that peers are not under the control of the game producer. Since all server data are distributed among 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 single peer has more authority than another.

There are various security issues that are usually classified according to the level in the protocol stack where they occur. The areas identified by GauthierDickey et al. [18] and expanded upon by Webb et al. [19] are: game level, application level, protocol level, and infrastructure level. This is consistent with the generally used layered security model [20].

As with all taxonomies, all cheats may not cleanly fit into one of 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 [21]. 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 [22]. These issues are the same issues facing P2P MMOGs, with the exception of the game and application layer issues.

2.3.4 Incentive Mechanisms

P2P schemes require all players to share resources in order to ensure correct functionality. The issue with this is that players 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. For example, Bittorrent systems use the tit-for-tat protocol to ensure that all people downloading data are also contributing data [23]. Such mechanisms are also required with P2P MMOGs. 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.

2.3.5 Distributed Computation

Nonplayer Characters are characters that are not controlled by any human player, but are rather controlled by some artificial intelligence routine or script executing on some host machine. These characters represent the traders and monsters in MMOGs and usually contain sets of rules that determine how they should interact with Player Characters (PCs) as well as their own state information. An NPC's state can be how much money and items it has to trade or how much health it still has after being attacked by a player.

In the original NPC host allocation classification by Fan, both NPC state and computational routines are combined into a single category [24]. In the classification presented below, NPC state forms part of normal game state persistency, since NPC objects are game objects like any other. The NPC routines requiring computational power are grouped under the heading of distributed computation. This heading is meant to include the distribution of all in game computational elements.

Some game objects require computational power to function. An example of this is the Artificial Intelligence routines of NPCs or the computation of physics effects on in-game objects. Some architectures assume that the computational requirements will be fulfilled where the object state is hosted [25], but other schemes exist that allow for the CPU power to be distributed among peers. One such scheme makes use of a "job board" like mechanism, where

tasks are advertised on specialized super peers. Other peers monitor these super peers and may elect to perform the advertised tasks [26].

2.3.6 Game State Persistency

Game state persistency involves the storage of game objects, either in primary or secondary storage. In a recently completed PhD on the subject of P2P MMOGs, Fan had this to say about state persistency: "Game state persistency is a major challenge for P2P MMOGs as existing P2P storage infrastructures are designed to support file sharing, and seldom fulfill the performance and security requirements of a MMOG. ... the persistency area is still immature with many problems waiting to be investigated." [24].

State persistency is treated as a subdomain of state consistency, in that state persistency models define where and how the root or authoritive objects are stored. It is assumed that replica objects are always stored in the primary memory of the clients that immediately require the information contained in the root object.

The issue of game state persistency in P2P MMOGs is the focus of this paper and the remainder of the paper will deal exclusively with this subject. However, before the different storage types are reviewed, some classic C/S models are presented for comparison with the fully distributed model.

3 Classic Consistency Models

A game object exists in two forms: the root object and the replica object. Root objects are the objects traditionally housed on the Server in the C/S-based MMOGs. All clients obtain replicas of these objects and duplicate them locally in order to perform low-latency computations. An example would be an NPC monster. When players perceive the monster in the virtual world, a duplicate of the NPC objects is sent to the user's computer for display purposes. When another player attacks the NPC, the change in health will be computed at the server and an update is then sent in order to ensure consistency between root and replica objects.

3.1 Terminology

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.

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 fully distributed model, also called the event-based model [27], and the C/S-based model, also called update-based model [28].

3.2 Event Based (Fully Distributed)

Fig. 2a shows the fully distributed model. The state persistency model for this consistency model is that 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. The event-based model works well for strategy games, and was implemented in Age of Empires [27] and Starcraft [29].

The order in which events 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 [30]. 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 [18]. This, however, makes it impossible for a player with a high latency to play the game with anyone other than from her own continent. When latency issues are not present and all players possess reasonable latencies, the event-based model can provide for a high degree of responsiveness, because of no extra latency being added by a server and no extra server hop required for communications.

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 traffic increases with a factor of N^2. The security issues of the fully distributed network 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.

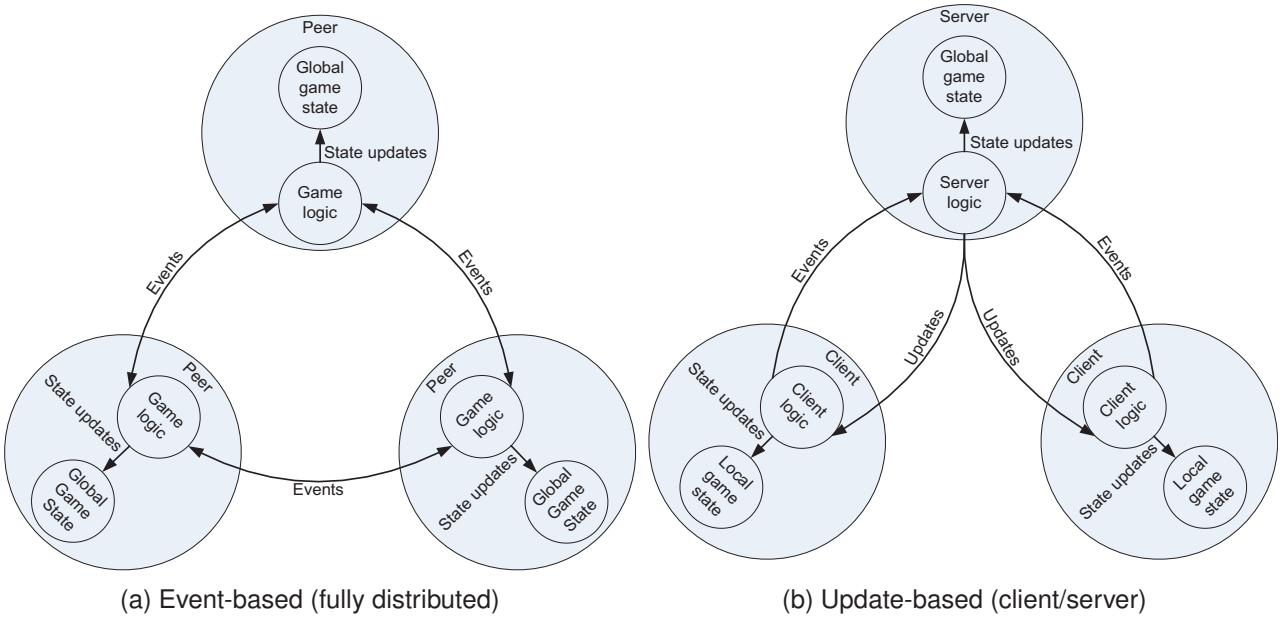

Fig. 2. Consistency models.

3.3 Update-Based (C/S)

An alternative to the event-based model is the update-based model, shown in Fig. 2b. This model is based on the C/S network model. The persistency model here is that an authoritative global game state is housed on the server and a nonauthoritative 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 security advantages of the C/S model also apply to this consistency model. Another reason why the update based model is successful is because it is more scalable then the fully distributed model. More hardware can be used to build a more powerful server, which can handle more clients. Computer clusters and large server are, however, costly to obtain and maintain.

The update-based model is used in many, if not all, MMOGs currently in operation. This includes games like World of Warcraft, Eve Online, and Ultima Online, to name just a few.

3.4 Client/Multiserver 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 [31].

3.4.1 Sharding

The consistency model that is exemplified by having a state persistency model where the game state (world) is duplicated over multiple servers, with players connecting to one of these servers is termed "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 its capacity. In the past, this has caused unhappiness among 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. For all practical purposes, this approach is still merely a C/S approach, with players forced into a specific C/S environment.

The benefit of sharding is its ease of implementation and the reduction of content designer load. Because no inter-server communication is required and no server migration is supported, this method greatly simplifies the server design process. Another benefit of sharding is that it allows for a relatively small game world to support many players because of the duplication of the worlds. This reduces the load on level designers and content designers, who now have to populate a much smaller world with content.

3.4.2 Replication-Based

The replication-based model is 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 [32]) or through a load distribution algorithm to a server (proxy servers [33]). 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.

3.4.3 Object-Based

The state persistency model of the object-based consistency model equally distributes all in-game objects among the servers [34], [35], [36]. For an MMOG, most of these objects are expected to be player 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 among all servers. This allows for more accurate prediction and provisioning of resources, but still does not handle transient loads well.

Another issue is interserver communications for this architecture. The interserver communications are random and also more than the interserver communications for a region-based system. The reason for this is that the number of player interactions increase with a decrease in the distance between the players. Players playing together move together, chat, and interact with NPCs together. For a region-based model, all player-neighbor interactions remain local to the server.

3.4.4 Zone/Region-Based

The state persistency model of the zone-based consistency model divides the virtual world into zones or regions, which are hosted on different servers [37], [38]. A well-known example of this model is Eve Online. 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 [37].

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 behavior happens quite regularly and is known as flocking [39]. When players find something of interest in a region, many players will flock to that region. The solution to flocking has been overprovisioning of resources to handle peak loads, which suffers from the disadvantages discussed above. Also, if the load changes, the server has to be brought offline 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 [38]. 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.

4 Related Work

In this section, we discuss related surveys on P2P MMOGs. This will set the context for the discussion on state persistency for P2P MMOGs.

In 2007 Schiele et al. published the paper: "Requirements of Peer-to-Peer-based Massively Multiplayer Online Gaming" [40]. This paper presents a broad overview of some key requirements that P2P MMOGs should possess, to function correctly under any load for any period of time. These are: distribution, consistency, self-organization, persistency, availability, interactivity, scalability, security, efficiency, and maintainability. The focus of this paper is on the persistency requirement identified in the paper by Schiele.

In 2005, Hasan et al. published the manuscript: "A Survey of Peer-to-Peer Storage Techniques for Distributed File Systems" [41]. The manuscript describes different techniques used to store data in a distributed fashion. The difference between the paper by Hasan et al. and this paper is that this paper focuses on storing data for gaming applications, which have other requirements than normal file storage. The contents of the paper by Hasan et al. is also encapsulated in the overlay storage section of this survey.

Krause presented "A Case for Mutual Notification: A survey on P2P protocols for Massively Multiplayer Online Games." [14]. The protocols discussed in this survey focused on the areas of interest management and event dissemination. Three protocols were presented: "Application Layer Multicast (ALM)-based protocols," "Supernode-based protocols," and "Mutual notification-based protocols." The first two protocols deal with region-based interest management techniques that employ supernodes, also called super peers, and ALM to achieve state consistency. The third protocol, which is presented as an alternative to region-based techniques, is a distance-based technique, making use of Voronoi diagrams to achieve state consistency.

While the survey by Krause is also in the area of P2P MMOGs, it deals with the topics of interest management and event dissemination and not with the topic of state persistency. In other words, it explains how updates to objects may be sent to earlier versions of an object, but not how these objects may be stored.

Webb and Soh presented "A Survey on Network Game Cheats and P2P Solutions" [21]. The paper introduces a cheat classification scheme, defining different "levels" of cheating, along with some examples of cheats in each level. For each example given, the authors also discuss possible solutions to these cheats. The difference between this paper and the survey paper presented by Webb and Soh, is their paper deals with securing the information stored in objects as well as securing updates made to objects, while this paper deals with storing those objects.

The previously mentioned articles deal with other issues present in P2P MMOGs, namely interest management, event dissemination, and security. The paper by Amoretti: "A Survey of Peer-to-Peer Overlay Schemes: Effectiveness, Efficiency, and Security," provides details of the broader area of P2P overlay schemes [42]. The paper focuses on security issues present in P2P overlay schemes, while also introducing hybrid, unstructured and structured overlays, and provides an extensive list of applications of the different schemes in different areas.

Three areas present in Amoretti's survey and also related to this survey are: "Content sharing," "distributed storage," and "gaming." The content sharing technologies described in Amoretti's work are considered forms of overlay storage, further discussed in Section 5.3. While Amoretti's survey gives a broad description of P2P gaming in general and why it should be a viable alternative to a C/S system, this survey deals specifically with the area of P2P MMOGs and more specifically, with state persistency in P2P MMOGs.

TABLE 1
Differences between Storage Mechanisms

Storage type	Reliability	Responsiveness	Security	Fairness	Complexity	Examples
Super Peer	Medium	High	Low	Low	Very Simple	[1]
Overlay	High	Low	Medium	High	Simple	[51], [52], [24], [53]
Hybrid	High	High	Medium	Low	Complex	[54], [55]
Distance-based	Medium	High	Low	Medium	Very Complex	[56], [31], [57], [25]

5 Peer-to-Peer MMOG State Persistency Models

This section identifies state persistency techniques used in P2P MMOGs. To achieve state persistency in P2P MMOGs, a type of distributed storage is required. Two classic distributed storage architectures are the Network File System (NFS) [43] and Coda [44]. There are, however, some major differences between the requirements of a P2P state persistency model and the classic distributed file storage model.

First, NFS and Coda still require servers. One of the main advantages of P2P MMOGs is that servers are, for the most part, not required. The other requirement is low-latency file storage and retrieval, which NFS and Coda also do not address. Conflicts are also an issue with the Coda system, which allows multiple nodes to modify the same object. Despite the many differences, the divergent applications share some of the same requirements, including: scalability, reliability, security, and responsiveness.

The requirement of disconnected operation is generally not applicable to most interactive online games, where a player has to stay connected to the world, to be able to interact with it. One area where disconnected operation might be of interest to game state persistency is in the area of mobile gaming. A major challenge for mobile games is the large variances in network latency. Such networks require games that are resistant to network jitter.

One approach has been proposed in [45], where every game client serializes its game state and distributes this game state to all other clients. Each client then creates some target game state from all received game states as well as its own state. The game client then attempts to manipulate all NPCs in the game in order to achieve the target game state. The target game state is constantly updated and thus remains a moving target. This is a new type of consistency model with many research opportunities.

Four approaches have been identified by which state persistency is achieved in P2P MMOGs: *super peer storage*, *overlay storage*, *distance-based storage*, and *hybrid storage*. These four storage architectures are detailed in Sections 5.2, 5.3, 5.4, and 5.5.

Two other types of storage sometimes described in P2P MMOGs papers are *centralized storage* and *individual storage*. Centralized storage is storage in a centralized database, the same as for a C/S MMOG [46], [47], [48]. Centralized storage for an MMOG requires the same large expensive servers and high bandwidth as required by a classic C/S architecture and therefore does not fit into the P2P MMOG paradigm. For this reason, centralized storage will not be evaluated in this paper.

With individual storage, player data are stored on a player's own computer [49], [50]. These architectures do not address the storage of NPC state or mutable objects. These objects cannot be as easily mapped to a single node in the network as player state can and therefore require a mapping mechanism to decide where to host these objects. Individual storage will not be explicitly discussed in this survey, however, individual storage can be regarded as a subset of distance-based storage, which will be discussed in detail in Section 5.5.

Table 1 presents a characterization of current storage systems according to the characteristics defined in Section 5.1. Table 1 also provides some references that act as examples of the different storage types mentioned. These example architectures will be discussed in detail in the sections to follow. The purpose of the complexity category in Table 1 is to provide a measure of how difficult it will be for an application developer to use any of the storage types and is discussed in Section 5.6.

5.1 Characteristics

The key challenges related to P2P MMOG persistency models identified during this literature study were: scalability, reliability, responsiveness, security, and fairness. All state persistency models will be reviewed with these characteristics in mind. In order to evaluate any persistency model, metrics have to be defined to measure the key characteristics of a storage system. This will allow for different persistency models to be compared and provide a measure of the applicability of any persistency model to P2P MMOGs.

5.1.1 Scalability

Scalability underpins all evaluation criteria. This implies that for a system to be scalable, all other evaluation criteria should be satisfied for large numbers of nodes and data. For this reason, scalability will not be explicitly reviewed in the following storage types. Rather, all other evaluation criteria will be evaluated for a large number of nodes, thereby taking into account scalability.

The question of what constitutes a large number of nodes arises. To establish what an adequate number of nodes is, current MMOG architectures can be used for inspiration. It is proposed that to classify a system as *sufficiently scalable*, the smallest number of peers that should be used is approximately 3,000. This is the number of players per server, currently supported by most active C/S MMOGs. For a system to be classified as *truly scalable*, it is believed that the architecture should support 60,000 concurrent users, 20 times more than a sufficiently scalable system. This is the number of peak concurrent users (PCUs) currently supported by the super computer

used to host Eve Online [58]. These two measures will ensure that a system is as scalable as other currently available architectures.

For systems that will support the MMOGs of the future, it is believed that a target of 1 million nodes should be used. This number is 16 times that of a truly scalable system and the peak concurrent player count of World of Warcraft in China in 2008 [59]. Systems that will support these numbers can be classified as *highly scalable*. It is important to note that no current systems support such a large PCU count on a single server cluster. The PCU count presented for WoW is the PCU count over all server clusters hosting WoW in China.

5.1.2 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.

Fairness can be evaluated by evaluating the distribution of game state among all nodes in the P2P network. 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.

5.1.3 Reliability

For the storage to be reliable, it must be impossible for data to be lost, and stored data should always be available when a node requests it. Reliability encompasses both robustness and availability. Robustness means that the data should be resilient to network churn and availability means that data should be available to any node in the network, with the correct permissions.

5.1.4 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 data retrieval times should be small. 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.

5.1.5 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 a malicious way. This also adds the requirement that nodes should be authenticated in the storage system and that only authorized nodes should be able to alter data. Security is the combination of a number of objectives: Authentication, Authorization, Data Integrity, Confidentiality, Availability, Trust, Privacy, and Identity Management [20].

For a state persistency model to address the security objectives of authentication, authorization, confidentiality, trust, privacy, and identity management, a certification scheme with public and private key encryption is required. Such a scheme allows for the identification of users, and by having users sign any storage interactions, every change made to the storage system can be tracked. This is a major differentiating factor from classic distributed storage systems such as Freenet, where a primary objective is anonymity. If all operations are logged and all users have to be identifiable for a secure system, no users can truly be anonymous.

5.2 Super Peer Storage

Super peer storage relies on the super peer storing all information that is in its domain. A domain is usually created by segmenting the world 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 Fig. 3. One can see that this approach is modeled 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 game logic. Clients in the region only house copies of the regional objects and some client logic to update the local copies of objects. 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.

5.2.1 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 specialized 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, while other players never contribute.

In a system with lower fairness, the individual user load is also higher for those users that do have to contribute resources. This means that in an unfair system, the users that do have to contribute, have to contribute more than what they would have, were it a fair system.

5.2.2 Reliability

In a P2P system, with a high rate of churn, players are expected to constantly leave and join the network. Because

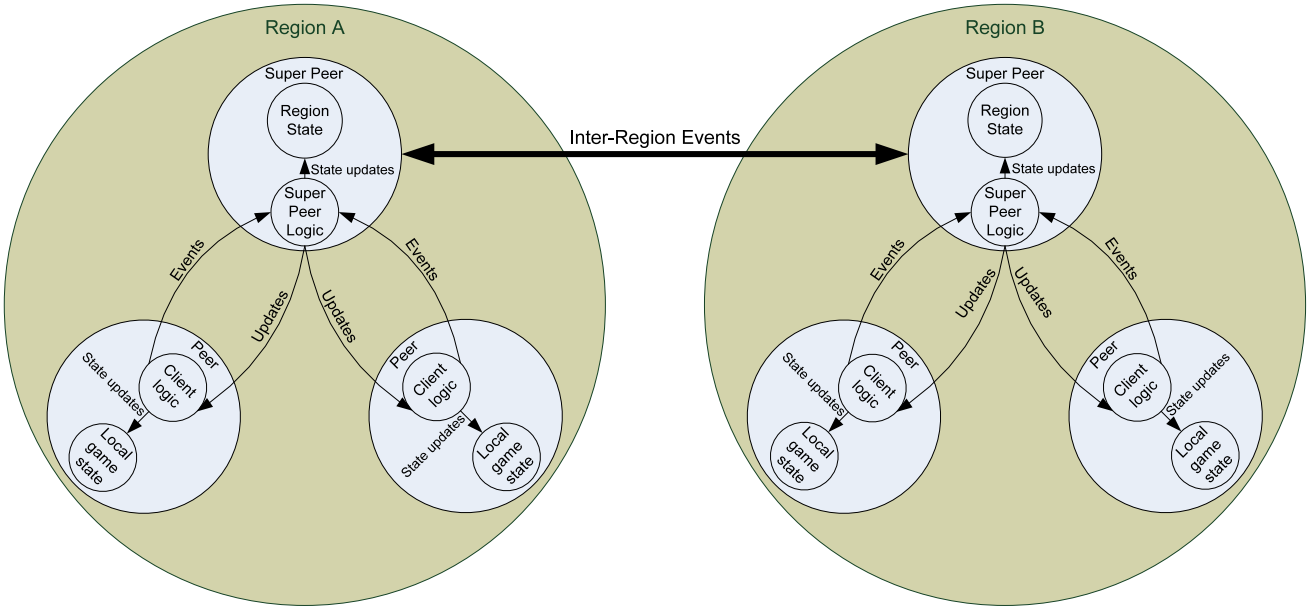

Fig. 3. Region-based Client/Server consistency model.

of this reality, redundancy mechanisms have to be developed that would ensure state data are always available, even when a super peer leaves the network. It is possible to solve these issues by having redundant super peers in each region that take over hosting responsibility when the main super peer leaves.

It is important that the main and backup super peers always possess 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 [26].

5.2.3 Security

Probably the 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 maliciously modify the data. The issue is not only that modification of the data might be possible, but also that it would be impossible for the cheating to be detected, because of no centralized logging. Locally obtaining access to data will circumvent the protections created by a certification system, which would then pose a threat to all security objectives protected by the certification system as mentioned in Section 5.1.5.

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 [53]. 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 have been received correctly. A hash is not the state update itself, so will be smaller, but the events that have to be sent to all super peers will increase traffic in the network and bandwidth usage by peers.

5.2.4 Responsiveness

There are 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 high speeds, making the system very responsive. Data retrieval from such a storage is 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.

5.2.5 Existing Architectures

Knuttson et al. [1] employ regional super peers called "coordinators," to host all shared object states. The coordinator is chosen as the node whose ID is closest to that of the region ID. The region ID is a SHA-1 hash of the region's textual name [60]. This mapping makes it unlikely that the coordinator will be a member of the region. The advantages of such a selection scheme is that the opportunities for cheating are reduced, because the data are hosted on a peer that has no or little interest in the data, as described in Section 5.5.3.

Coordinator handoffs also occur less than if the coordinator was an elected member of the region. If this is the case, a new coordinator has to be chosen every time the current coordinator leaves the region. In this scheme, handoffs only occur as a result of network churn, which is far lower than the number of players moving from one region to another.

Reliability is achieved by maintaining backup coordinators as the Distributed Hash Table (DHT) neighbors of each region coordinator. One method by which redundant region coordinators are maintained in [1], 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. This method is similar to how

reliability is achieved in overlay storage as described in Section 5.3.1.

5.3 Overlay Storage

Overlay storage is classified as using any type of structured P2P overlay to store data in a distributed fashion. This is a very broad definition, which basically encompasses any P2P distributed storage currently in use. Some examples and a comparison of different distributed storage techniques can be found in [41]. The reasoning is that any P2P distributed storage can be used to only store game files. Therefore, distributed storage is used as a distributed database for game files. Whether this is an optimal solution will be discussed in the remainder of the section.

5.3.1 Reliability

Overlay storage can be made reliable, using redundancy. One method used to achieve high reliability in structured overlay storage is to store k replicas for any file stored, in order to ensure the availability of the file. These replicas are stored at the neighboring nodes of the node containing the original file. Neighbors are the k nodes whose IDs are closest to that of the root node. 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 neighboring node, which stores a duplicate. This technique ensures high availability of data and the number of duplicates can be chosen according to the reliability of the network.

5.3.2 Responsiveness

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, an average of $O(\log(N))$ hops are required to retrieve or store a data item [61]. Although this is a sufficient order complexity for a routing algorithm in a large network, it is not sufficient to support a real-time application. For responsive MMOGs, a distributed file system is required that allows for real-time file storage and retrieval.

The mechanism by which churn is handled, described in Section 5.3.1, also improves responsiveness. This is achieved, because IDs created by the random hash function ensures that a node's neighbors are distributed randomly throughout the P2P overlay. This random distribution ensures that file replicas, stored at a node's neighbors, are uniformly distributed throughout the P2P overlay network.

5.3.3 Security

The overlay storage model is more secure than the super peer storage model as data are distributed among all peers and redundancy and quorum techniques can be implemented to ensure that files are retrieved with a high level of security.

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 additional network overhead as well as additional load on nodes to serve as file copies.

The network overhead can, however, be reduced by having file replica nodes only send hashes of the files, which may then be compared at the requesting node. Hashes require less bandwidth, while still allowing a requesting node to check update validity by hashing the received update and comparing with the received hashes.

5.3.4 Fairness

Overlay storage is fair, as all nodes share file data and requests equally. The system might be made fairer by taking into account the heterogeneity of peers. Peers do not all possess the same resources, something which a truly fair system should take into account. The difficulty with using such a scheme is that peers can be made to report incorrect resource information in order to reduce their resource donation requirement. This is where incentive mechanisms have to be investigated as well as ways to ensure correct resource reporting.

5.3.5 Existing Architectures

In 2004, Merabti and El Rhalibi mention the issue of "Data Storage" [52]. It is recognized that a distributed storage scheme is required and that such a scheme "... requires careful designing ..." They propose the use of a data storage architecture based on the Freenet project [62]. Freenet is a distributed storage facility that uses a Darknet to ensure user anonymity when distributing files. A system such as Freenet is designed for general file sharing, which means that no focus is placed on achieving the high levels of responsiveness required for MMOGs. While the need for state persistency is briefly mentioned in the 2004 paper by Marabti and El Rhalibi, Douglas et al. implement a workable solution for state persistency in 2005.

Douglas et al. designed a P2P MMOG architecture in 2005 [51] and implemented state persistency using a distributed storage implementation, which they developed in 2003 [63]. The storage system allows for the manipulation of spatial data, while also implementing range queries. This enables the system to store and retrieve data that exist in a certain area of the game world. In the MMOG architecture they developed, state persistency is implemented by the "Spatial Data Service" (SDS), which is a distributed storage architecture that uses the Chord P2P overlay for routing [4].

PAST has become a popular way to implement state persistency. This is the approach proposed by both Hampel et al. in 2006 [53] as well as Fan in 2009 [24]. In these publications, it is said that PAST is used to store the global game state, but never is detailed what is stored and how regularly it is stored. Player information is supposed to be stored as game state, but from the papers it is unclear how position updates are handled. It is not clear whether the last position of a player is stored at all or how regularly it is stored. It is important to know how position updates are handled in the game, since position updates are the most common type of update [1].

PAST has not gained much commercial adoption, because of the lack of support for keyword searches, which is a requirement of most distributed storage networks, where users constantly search for content. Keyword searches are, however, not required by the storage mechanism of an MMOG. The IDs of the items stored in the

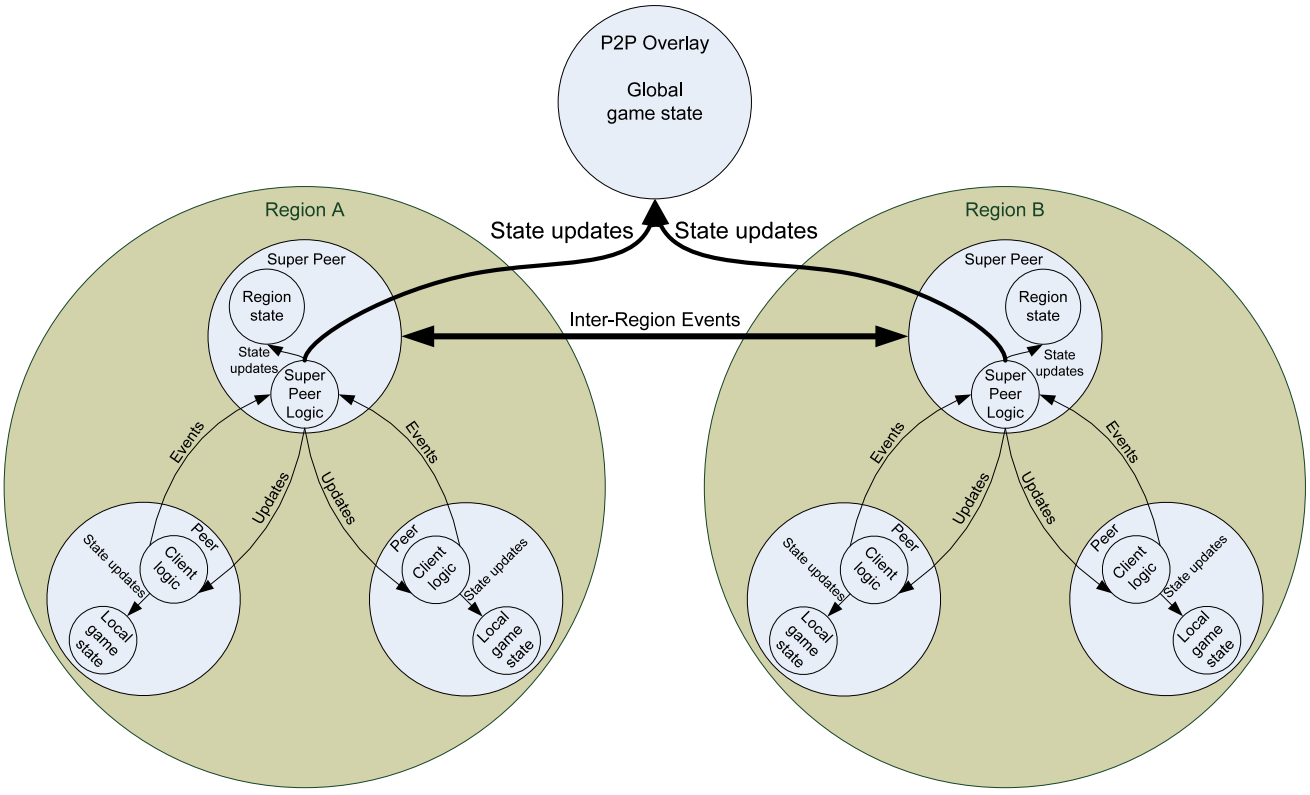

Fig. 4. Hybrid region-based super peer storage with backup overlay storage consistency model.

network will be known. A player's inventory can, for example, always be called `(player_name)_inventory`. A hash of this file name will find the correct file. This, and the use of Scribe, is why PAST has become popular with researchers of P2P MMOGs.

PAST [64] 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 [60]. The file, along with the ID are 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 wish to retrieve the file again, it only requires the file hash. A "get" message can be sent to the overlay, where the overlay will route the message to where the file is situated and retrieve the file.

In an effort to increase responsiveness, PAST also employs caching techniques [61]. If a node forward many queries for a file, that node can elect to cache the file to improve the responsiveness of the system. This caching can only occur if the node has space available. If a node has cached a file and another file is explicitly inserted into the node, it can elect to remove the cached file in order to free up space. This means that the success of the caching mechanism is directly related to the level of storage utilization. Higher utilization will prevent files from being cached.

The main difference between the implementation by Douglas et al. and the PAST implementations, is that the former supports range queries on spatial data, which allows for a set of objects to be returned, queried by their virtual in-game position. With PAST, the exact ID of an object is required before it may be retrieved. Using PAST is the simplest means by which game state persistency may be implemented, but PAST is not necessarily the application best suited to game state persistency. There exists a need for more research into appropriate state persistency mechanisms for P2P MMOGs.

5.4 Hybrid Region-Based Storage

Fig. 4 shows a type of Super Peer/Overlay hybrid storage implemented in [54]. The model depicted in Fig. 4 uses an overlay storage, managed by regional super peers. The world is divided into regions, with each region controlled by a super peer. The complete region state is cached at every super peer, the same as with super peer storage. There also exists a backup overlay storage architecture, to which data may be backed up for long term, redundant and secure storage. The hybrid region-based overlay storage contains many improvements over pure overlay storage as will be discussed in the following sections.

5.4.1 Reliability

Because of the use of overlay storage for backup, the hybrid region-based storage is almost as reliable as a pure overlay storage. It is classified as almost as reliable, because there is a delay between when data change and when it is updated in the overlay. If a super peer fails during this time and the data were not backed-up to the overlay, that data could be lost. Backup super peers can, however, be implemented as described in Section 5.2.2 to improve the reliability of the hybrid storage model.

5.4.2 Responsiveness

Because all regional files are cached at super peers, the system is as responsive as super peer storage.

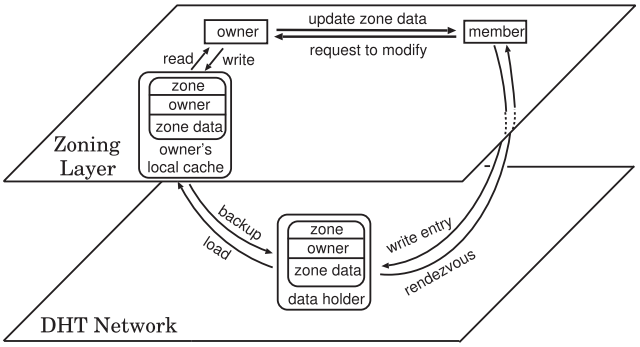

Fig. 5. Zoned Federation Model [54].

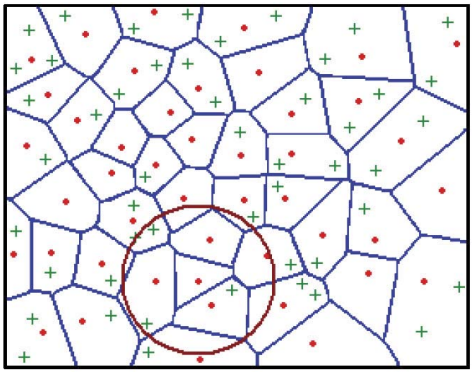

Fig. 6. Voronoi Diagram [56].

5.4.3 Security

Security in hybrid storage is still an issue, because of the inherent problems of the super peer storage model. Although it is more difficult for nodes to access and manipulate data stored in the overlay, a malicious node promoted to super peer status may manipulate the region data it controls.

It does seem possible to achieve higher levels of security, by checking data received from super peers, against the data stored in the overlay. Care should be taken with such a scheme, because the rate of change of data at the super peers might be higher than the rate that data are submitted to the overlay. Another issue is the time delay between data received from a super peer and data received from the overlay.

5.4.4 Fairness

The issue of unfairness is also still present in hybrid storage. The system is fairer in that all nodes share the load of the overlay storage, but there still exists the unfairness of the super peer storage. As all data exists in both the overlay storage as well as the super peer storage, the system is as unfair as super peer storage, because the same quantity of data as in super peer storage is not being distributed evenly among all nodes.

5.4.5 Existing Architectures

The first hybrid state persistency model for P2P MMOGs was proposed by Iimura et al. in 2004 [54] and called the "Zoned Federation Model," shown in Fig. 5. The regional super peers are called "Zone Owners," which handle all events by clients in their zone or region. In the Zoned Federation model, a Zone Owner acts as the primary storage medium for all object states in the zone or region. As shown in Fig. 4, this is analogous to an update-based model, divided into zones. The difference here is that the game state of all zone owners are regularly backed-up to overlay storage. The zoned federation model can thus be seen as a super peer/overlay storage hybrid. The super peers storage provides for low-latency data storage and the overlay storage provides security and reliability.

An extended abstract, published by GauthierDickey et al. in 2004 [55], proposed to distinguish between permanent and ephemeral data. Permanent data are described as data that should exist at all times and ephemeral data are described as data that need only exist for as long as its owner is in the game. An item, being dropped by a dispatched NPC, can be considered as ephemeral. When the peer on which the data are hosted leaves the area, that item can disappear. An example of permanent data are a player's inventory contents, which can further be classified as participatory data or a player's house, which can be classified as existential data. Participatory data are data that need only be available when a specific player is in the game and existential data are data that should be available, even when a certain player is not present in the game.

Categorizing data by how long and under which circumstances the data should exist, may assist in the design of the storage model. Since ephemeral data do not have to exist after the player has left the game, it may be stored in primary memory. Participatory data might also be stored on the player's computer, but security issues will have to be kept in mind. Existential data will have to be stored somewhere other than on the player's computer, since other players will require the data, even in the absence of the player that might have left the game. GauthierDickey et al. did not explore how their data classification scheme might be translated into a state persistency model.

5.5 Distance-Based Storage

Distance-based approaches, such as the Voronoi storage approaches [56], [31] and some more general approaches [57], [25], 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.

Given a set of points, the Voronoi diagram of the set of points is the partition of the plane, which associates a region around every point in such a way that all other points contained in the region are closer to the center point than any other point in the set. Fig. 6 shows a Voronoi diagram, where the lines define the region boundaries, the dots define the players, which make up the set of points for which the diagram was calculated, the plus signs represent mutable objects and the circle represents the AoI of a central point in the set.

For the Voronoi approaches, described in more detail in Section 5.5.5, 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.

5.5.1 Responsiveness

An issue with the above reasoning is that multiple nodes are usually 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.

Multiple player interactions are, however, not as big an issue as others have suggested [8]. 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. In 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 C/S or super peer application.

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. Because object transfer introduces overhead into the system, how regularly an object has to be transferred and whether the number of transferals produce sufficiently low overhead to implement a real-time game, still have to be investigated.

Voronoi-based storage schemes also become unresponsive when communications are no longer between neighbors, but between two arbitrary nodes in the Voronoi overlay. When such communications occur, the average required time to route a message is $O(N^{1/2})$ for a two-dimensional configuration. Advancements have been made that suggest augmenting the Voronoi overlay with additional links to far off nodes to create a small world network. This reduces the average routing time to $O(\log(N))$, the same as for overlay storage [65].

5.5.2 Reliability

Reliability, because of network churn, is still an issue. Nodes will leave the network whenever a player stops playing the game, which makes it a common occurrence. When nodes leave the network, the objects that are stored on that node should still be accessible to other players. This will require transferring all objects contained in the leaving node to another object, still present in the network. No papers have yet dealt with the issue of reliability in a distance-based storage network.

The same solution that is used for overlay storage, namely the presence of redundant peers, might also be implemented for distance storage. Another structured overlay might be used to implement this redundancy in exactly the same way it is done with hybrid storage mentioned in Section 5.4.1.

5.5.3 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 modifications 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.

This security issue is similar to that of super peer storage, in that local data can be accessed, thus circumventing the certification system. With this circumvention, the objectives of authentication, authorization, confidentiality, trust, privacy, and identity management are all compromised.

5.5.4 Fairness

Distance-based storage is relatively fair as all objects are distributed among 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 might 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 neighboring peers. The use of aggregators as a proposed solution to this problem is discussed in Section 5.5.5.

5.5.5 Existing Architectures

Bharambe et al. created the Colyseus architecture in 2006 [57]. The architecture is designed to support First Person Shooter (FPS) games and implemented to function with Quake II. Mutable game objects are stored on the peer that is nearest to the object in the game world. An "object placer" component is mentioned, but the details of the placement algorithm are left for future work. The architecture also does not implement nonvolatile state persistency, since this is not required for normal FPS games, where object states need only exist to the end of a round and where players generally do not leave before the end of the round. This means that object states are only stored in primary memory, until the end of a game round.

The Solipsis architecture was created by Frey et al. in 2008 [25]. The architecture uses Voronoi diagrams to create virtual regions. Stationary objects are maintained by site nodes until a player picks up an object. When an object is picked up, control of that object is transferred to the player that picked up the object. The Solipsis architecture focuses on distributed physics computation and when a player gains control of an object, that player is responsible for the object's physics computations. That player should also save all object state until a new player takes the object, at which time control is transferred to her. Control can also be transferred back to a site node if an object remains stationary for some time.

What differentiates the Solipsis distance-based storage from the other architectures presented in this section, is that object states are only handled by peers as long as those peers directly use an object. At other times, those objects are handled by site nodes. This differs from other distance-based storage techniques, where all objects that are nearest to a player in the virtual world are controlled by that player.

In 2008, papers were published by Buyukkaya and Abdallah [56], and Hu et al. [31], proposing to use Voronoi diagrams [66] to implement distance-based storage. Voronoi state management schemes host the mutable objects on the peer in which region the object exists. As peers move around in the virtual world, the Voronoi diagram has to be constantly recalculated and objects have to be moved to new owner peers as the regions in which they fall change. The significant advantage of the Voronoi approach is that peers only require connections with their neighbors, and peers within their AoI.

A thesis by Chang also describes the Voronoi approach in more detail and how to achieve game state consistency among all nodes in a Voronoi network [67]. What distinguishes this work from the others is the implementation of a load balancing mechanism. When peers get overloaded, another, more powerful peer is chosen as an Aggregator. The Aggregator assumes responsibility for a larger area that encompasses multiple peers. This scheme will reduce the load on peers with minimal resources, but it is uncertain how this would reduce load when peers with an average number of resources in an area become overloaded.

The works by Buyukkaya and Abdallah, Hu et al., and Chang form part of the VAST project, which is being created to be a fully functional P2P overlay architecture, using Voronoi diagrams as its basis [68]. The reason why state persistency in Voronoi-based P2P MMOGs architectures are mostly distance-based approaches, is because the Voronoi diagram immediately identifies which objects are closest to a particular peer, and therefore, which objects that peer has to host. Architectures not making use of Voronoi diagrams still require some other mechanism to identify which objects which peers should host.

5.6 Summary and Recommendations

Supper peer storage can be seen as a C/S type storage implementation, where every region has a super peer that stores all data for that region. Overlay storage is a fully distributed, P2P approach to storage, where every node stores data as part of a P2P overlay network. Hybrid region-based storage combines super peer and overlay storage to improve the overall storage performance. Distance-based storage is a different paradigm that stores game objects on the nearest node to that game object. This type of storage is usually characterized by the use of Voronoi diagrams to determine which nodes are nearest to which objects.

Super peer storage is characterized by its high level of responsiveness and ease of implementation. Overlay storage is characterized by its high level of fairness and reliability. Hybrid region-based storage combines the two previous schemes and has high levels of reliability and responsiveness. Distance-based storage is also very responsive and can be made both reliable and fair.

Security is still an issue for all storage types. Super peer storage has the issues that are usually present in a centralized system, namely low fairness, security and also not being reliable and not resistant to failure. The main issue with overlay storage is its unresponsiveness, because of the routing delay in the P2P overlay. Hybrid storage suffers from the disadvantages of super peer storage, namely low fairness and security, due to all files still stored on super peers. The main issue of distance-based storage is security, because players that have the most incentive to alter object states own those objects.

The different storage types vary greatly in implementation complexity as also shown in Table 1. The simplest type of storage to use is super peer storage, but because of the many issues present, this is not recommended. Only when one is sure that the super peers will not be overloaded should this storage method be used. There might still be some customers who are unhappy that a few have to serve data to the many.

No recent mention could be found of researchers implementing super peer storage for P2P MMOGs. This could mean that there has been very little evolution in super peer storage and that super peer storage is currently less mature than other storage schemes. Super peer storage might still be used for small independent casual games, such as online board games, where a group of friends can implicitly trust each other or where the risk of cheating is at a minimum. The reason why this storage method is recommended for games from independent developers, is because the scheme is easy to implement and will not require many resources when used for small games.

For developers who are looking for a simple yet fairly robust storage system to start using in their project, overlay storage is recommended. Overlay storage can be used for games that do not require low-latency data access and would allow for lazy updating of the database. These games will be lightweight casual and social games. But, depending on the game implementation itself, this storage type might also be used for smaller hardcore games. Overlay storage will place some restrictions on certain game mechanics, but it is possible that games could be designed around these restrictions. The primary restriction being that data cannot be immediately retrieved or stored from or to storage.

Examples of overlay storage types that might be used are: PAST, Freenet, and Oceanstore and any other overlay storage that is based on a DHT overlay. Storage based on DHT overlays is still a viable research field and future implementations will improve on the present systems.

Hybrid region-based storage has all the advantages of super peer storage, but with the additional reliability of overlay storage. For this reason, it is recommended that any game presently requiring high levels of reliability and responsiveness use hybrid storage, instead of super peer or overlay storage. This storage type currently seems to be the only type applicable to games that require responsive and reliable storage.

That said, no existing implementations could be found that implement this type of storage for public use. The consequence is that someone who wishes to use this storage will mostly have to implement it. This can be done by starting out with super peer storage and using a publicly available overlay storage to augment the super peer storage.

Distance-based storage is still fairly immature, but shows a lot of promise. No publicly available implementation could be found for this type of storage. Where this type of storage was used, it was explained as distance-based storage, but most of the details were left out. This storage is, therefore, not a candidate as a storage technique for

developers that are merely looking for a storage system to use presently. This type of storage does, however, provide a lot of opportunities for research.

6 Conclusion

6.1 Summary

After providing an overview of the classic C/S and C/MS state persistency techniques, this survey classified P2P MMOG state persistency techniques into super peer based, overlay based, distance based, and hybrid storage. The advantages and disadvantages of each method were discussed after identifying key challenges that state persistency techniques have to solve. These challenges are: Reliability, Security, Fairness, and Responsiveness.

We conclude that there exists no single state persistency architecture, currently in use, that is suited to P2P MMOGs. None of the storage techniques reviewed meet the requirements of a real-time distributed application, such as an MMOG. What is required is a state persistency architecture, specifically geared toward data persistency in P2P MMOGs, that meet all the challenges of the application.

This survey was written, because of an identified need for a concise summary of the field of P2P MMOG game state persistency. Many techniques used in the past were used because of the ease with which they could be integrated into a P2P MMOG. The purpose of this survey is to identify those techniques and to stimulate further research, using empirical methods to compare the different storage techniques used.

6.2 Prospective Research Directions

One of three paths may be followed in order to create a storage type that is suitable to modern hardcore MMOGs. The first would be to take a look at the deficiencies mentioned in the different storage types and to then improve one of these types. The other path is what was seen with hybrid storage, where multiple types of storage are combined in order to form a new and improved storage type. The third path would be to create a new storage type, based on some novel insight into the ways in which games are designed.

What follows is a discussion on what work is required in order to improve the current storage types and attempts to define the gaps present that should still be solved by future researchers.

For super peer storage to become viable, a means is required to secure the data stored in the super peers and ensure that no super peer is able to access the data stored. If this issue is solved, the fairness issue still exists, but might be allowable depending on the type of application.

Overlay storage is a popular storage method in the literature. This is believed to be more as a consequence of the use of Scribe [69] than any inherit benefit to P2P MMOGs [53], [24]. The reason for using overlay storage in so many implementations seem to be merely the availability of PAST, after having already decided to use Scribe. In other words, researchers use Scribe for ALM, Scribe runs on Pastry, and PAST is then a readily available storage implementation which also runs on Pastry.

The applicability of overlay storage to MMOGs is still unknown and further research is required. This paper showed that while PAST is regularly used in P2P MMOGs, it is not because of the applicability of PAST to P2P MMOGs.

One type of hybrid storage was investigated in this paper, namely overlay/super peer storage, since this is a hybrid type currently seen in P2P MMOGs. Other hybrid types should also be investigated, for example, a distance-based/overlay storage hybrid might improve the reliability of distance-based storage. Multitiered storage might also be of interest, where players or areas are grouped in some way. One type of storage might then be implemented among the members of a group, with another storage type implemented over all groups. Such a storage type might improve responsiveness among group members, while adding reliability because of an intergroup backup mechanism.

Distance-based storage still seems immature, with many open questions. Currently, distance-based storage is based on Voronoi diagrams, which seem promising. But Voronoi diagrams only provide for a means to identify which objects should be under a specific node's control. Object migration, which will occur frequently in this type of storage should still be addressed. The number of migrations should be kept to a minimum to ensure responsive access to all stored objects.

The storage is also not yet reliable, but it could be possible to add overlay storage as a backup to the system and thereby add reliability. If voting techniques can be used to update game state, some of the security issues might also be solved. With these two issues addressed, the high degree of fairness of distance-based storage makes it an excellent candidate to power the P2P MMOGs of the future.

An empirical comparison of the performances of the different storage types is also required. It is believed that the metrics presented in this paper could provide a common basis to use in comparing the different storage types.

Research is still required into the characteristics of data stored by MMOGs. This includes how regularly game objects are stored as well as the sizes of these objects. It is also expected that different read and write patterns exist for different types of game objects. These patterns should be explored in order to determine what the required performance is for P2P MMOGs storage mechanisms.

It is recommended that C/S MMOG storage patterns initially be investigated to provide a benchmark of the required storage performance for a mature MMOGs. The storage patterns for a C/S compared to P2P MMOGs might be different. The storage patterns among different MMOGs might also differ greatly, but for an immature field any data are better than no data.

In conclusion, this paper describes a possible new research field with many open questions still to be answered.

Acknowledgments

The financial assistance of MIH and the National Research Foundation (NRF) toward this research is hereby acknowledged. Opinions expressed and conclusions arrived at, are those of the author and are not necessarily to be attributed to MIH or the NRF.

REFERENCES

[1] B. Knutsson, H. Lu, W. Xu, and B. Hopkins, "Peer-to-Peer Support for Massively Multiplayer Games," *Proc. IEEE INFOCOM,* vol. 1, p. 107, 2004.

[2] M. Castro, M. Costa, and A. Rowstron, "Debunking Some Myths about Structured and Unstructured Overlays," *Proc. Second Symp. Networked Systems Design and Implementation (NSDI '05),* vol. 2, pp. 85-98, 2005.

[3] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, "A Scalable Content-Addressable Network," *Proc. SIGCOMM,* pp. 161-172, 2001.

[4] I. Stoica, R. Morris, D. Karger, M.F. Kaashoek, and H. Balakrishnan, "Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications," *SIGCOMM Computer Comm. Rev.,* vol. 31, no. 4, pp. 149-160, 2001.

[5] B.Y. Zhao, J.D. Kubiatowicz, and A.D. Joseph, "Tapestry: An Infrastructure for Fault-Tolerant Wide-Area Location and Routing," technical report, Univ. of California at Berkeley, 2001.

[6] A. Rawstron and P. Druschel, "Pastry: Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems," *Proc. 18th IFIP/ACM Int'l Conf. Distributed Systems Platforms (Middleware '01),* 2001.

[7] R. Rodrigues and P. Druschel, "Peer-to-Peer Systems," *Comm. ACM,* vol. 53, pp. 72-82, 2010.

[8] L. Fan, P. Trinder, and H. Taylor, "Design Issues for Peer-to-Peer Massively Multiplayer Online Games," *Int'l J. Advanced Media Comm.,* vol. 4, no. 2, pp. 108-125, 2010.

[9] C. GauthierDickey, D. Zappala, V. Lo, and J. Marr, "Low Latency and Cheat-Proof Event Ordering for Peer-to-Peer Games," *Proc. 14th Int'l Workshop Network and Operating Systems Support for Digital Audio and Video,* pp. 134-139, http://doi.acm.org/10.1145/1005847.1005877, 2004.

[10] J.L. Miller and J. Crowcroft, "The Near-Term Feasibility of P2P MMOG's," *Proc. Ninth Ann. Workshop Network and Systems Support for Games (NetGames),* pp. 1-6, 2010.

[11] 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.

[12] L. Wang, S. Turner, and F. Wang, "Interest Management in Agent-Based Distributed Simulations," *Proc. Seventh IEEE Int'l Symp. Distributed Simulation and Real-Time Applications,* pp. 20-27, 2003.

[13] J.-S. Boulanger, J. Kienzle, and C. Verbrugge, "Comparing Interest Management Algorithms for Massively Multiplayer Games," *Proc. Fifth ACM SIGCOMM Workshop Network and System Support for Games (NetGames),* p. 6, 2006.

[14] S. Krause, "A Case for Mutual Notification: A Survey of P2P Protocols for Massively Multiplayer Online Games," *Proc. Seventh ACM SIGCOMM Workshop Network and System Support for Games (NetGames),* pp. 28-33, 2008.

[15] S. Fiedler, M. Wallner, and M. Weber, "A Communication Architecture for Massive Multiplayer Games," *Proc. First ACM SIGCOMM Workshop Network and System Support for Games (NetGames),* pp. 14-22, 2002.

[16] C. Diot, B. Levine, B. Lyles, H. Kassem, and D. Balensiefen, "Deployment Issues for the IP Multicast Service and Architecture," *IEEE Network,* vol. 14, no. 1, pp. 78-88, Jan./Feb. 2000.

[17] C. Neumann, N. Prigent, M. Varvello, and K. Suh, "Challenges in Peer-to-Peer Gaming," *SIGCOMM Computer Comm. Rev.,* vol. 37, no. 1, pp. 79-82, 2007.

[18] C. GauthierDickey, D. Zappala, V. Lo, and J. Marr, "Low Latency and Cheat-Proof Event Ordering for Peer-to-Peer Games," *Proc. 14th Int'l Workshop Network and Operating Systems Support for Digital Audio and Video (NOSSDAV),* pp. 134-139, 2004.

[19] S.D. Webb, S. Soh, and W. Lau, "RACS: A Referee Anti-Cheat Scheme for P2P Gaming," *Proc. Int'l Workshop Network and Operating Systems Support for Digital Audio and Video (NOSSDAV),* pp. 34-42, 2007.

[20] A. Belapurkar, A. Chakrabarti, H. Ponnapalli, N. Varadarajan, S. Padmanabhuni, and S. Sundarrajan, *Distributed Systems Security: Issues, Processes and Solutions.* Wiley, 2009.

[21] S.D. Webb and S. Soh, "A Survey on Network Game Cheats and P2P Solutions," *Australian J. Intelligent Information Processing Systems,* vol. 9, no. 4, pp. 34-43, 2007.

[22] D.S. Wallach, *Software Security—Theories and Systems,* A Survey of Peer-to-Peer Security Issues, pp. 253-258. Springer, 2003.

[23] B. Cohen, "Incentives Build Robustness in Bittorrent," *Proc. First Workshop Economics of Peer-to-Peer Systems,* 2003.

[24] L. Fan, "Solving Key Design Issues for Massively Multiplayer Online Games on Peer-to-Peer Architectures," PhD dissertation, School of Math. and Computer Sciences—Heriot-Watt Univ., 2009.

[25] D. Frey, J. Royan, R. Piegay, A.-M. Kermarrec, E. Anceaume, and F.L. Fessant, "Solipsis: A Decentralized Architecture for Virtual Environments," *Proc. First Int'l Workshop Massively Multiuser Virtual Environments (MMVE),* pp. 29-33, 2008.

[26] L. Fan, H. Taylor, and P. Trinder, "Mediator: A Design Framework for P2P MMOGS," *Proc. Sixth ACM SIGCOMM Workshop Network and System Support for Games (NetGames)* pp. 43-48, 2007.

[27] P. Bettner and M. Terrano, "1500 Archers on a 28.8: Network Programming in Age of Empires and Beyond," *Proc. Game Developers Conf. (GDC '01),* 2001.

[28] T. Sweeney, "Unreal Networking Architecture," Epic MegaGames, technical report, http://unreal.epicgames.com/Network.htm, 1999.

[29] A. Dainotti, A. Pescape, and G. Ventre, "A Packet-Level Traffic Model of Starcraft," *Proc. Second Int'l Workshop Hot Topics in Peer-to-Peer Systems (HOT-P2P '05),* pp. 33-42, 2005.

[30] N. Baughman and B. Levine, "Cheat-Proof Playout for Centralized and Distributed Online Games," *Proc. IEEE INFOCOM,* vol. 1, pp. 104-113, 2001.

[31] S.-Y. Hu, S.-C. Chang, and J.-R. Jiang, "Voronoi State Management for Peer-to-Peer Massively Multiplayer Online Games," *Proc. IEEE Fifth Consumer Comm. and Networking Conf. (CCNC),* pp. 1134-1138, 2008.

[32] E. Cronin, B. Filstrup, A.R. Kurc, and S. Jamin, "An Efficient Synchronization Mechanism for Mirrored Game Architectures," *Proc. First Workshop Network and System Support for Games (NetGames),* pp. 67-73, 2002.

[33] J. Müller and S. Gorlatch, "Rokkatan: Scaling an RTS Game Design to the Massively Multiplayer Realm," *Computers in Entertainment,* vol. 4, no. 3, p. 11, 2006.

[34] F. Lu, S. Parkin, and G. Morgan, "Load Balancing for Massively Multiplayer Online Games," *Proc. Fifth Workshop Network and System Support for Games (NetGames),* p. 1, 2006.

[35] J.C.S. Lui and M.F. Chan, "An Efficient Partitioning Algorithm for Distributed Virtual Environment Systems," *IEEE Trans. Parallel and Distributed Systems,* vol. 13, no. 3, pp. 193-211, Mar. 2002.

[36] P. Morillo, J.M. Orduña, M. Fernández, and J. Duato, "An Adaptive Load Balancing Technique for Distributed Virtual Environment Systems," *Proc. 15th IASTED Int'l Conf. Parallel and Distributed Computing and Systems (PDCS '03),* pp. 256-261, 2003.

[37] M. Assiotis and V. Tzanov, "A Distributed Architecture for MMORPG," *Proc. Fifth ACM SIGCOMM Workshop Network and System Support for Games (NetGames),* p. 4, 2006.

[38] R. Chertov and S. Fahmy, "Optimistic Load Balancing in a Distributed Virtual Environment," *Proc. Int'l Workshop Network and Operating Systems Support for Digital Audio and Video (NOSSDAV),* pp. 1-6, 2006.

[39] J. Chen, B. Wu, M. Delap, B. Knutsson, H. Lu, and C. Amza, "Locality Aware Dynamic Load Management for Massively Multiplayer Games," *Proc. 10th ACM SIGPLAN Symp. Principles and Practice of Parallel Programming (PPoPP '05),* pp. 289-300, 2005.

[40] G. Schiele, R. Suselbeck, A. Wacker, J. Hahner, C. Becker, and T. Weis, "Requirements of Peer-to-Peer-Based Massively Multiplayer Online Gaming," *Proc. Seventh IEEE Int'l Symp. Cluster Computing and the Grid (CCGRID),* pp. 773-782, 2007.

[41] R. Hasan, Z. Anwar, W. Yurcik, L. Brumbaugh, and R. Campbell, "A Survey of Peer-to-Peer Storage Techniques for Distributed File Systems," *Proc. Int'l Conf. Information Technology: Coding and Computing (ITCC),* vol. 2, pp. 205-213, 2005.

[42] M. Amoretti, "A Survey of Peer-to-Peer Overlay Schemes: Effectiveness, Efficiency and Security," *Recent Patents on Computer Science,* vol. 2, no. 3, pp. 195-213, 2009.

[43] S. Shepler, B. Callaghan, D. Robinson, R. Thurlow, C. Beame, M. Eisler, and D. Noveck, *Network File System (NFS) Version 4 Protocol,* Sun Microsystems, Hummingbird and Network Appliance IETF RFC 3530, http://tools.ietf.org/html/rfc3530, 2003.

[44] J.J. Kistler and M. Satyanarayanan, "Disconnected Operation in the Coda File System," *ACM Trans. Computer Systems,* vol. 10, pp. 3-25, 1992.

[45] A. Chandler and J. Finney, "On the Effects of Loose Causal Consistency in Mobile Multiplayer Games," *Proc. Fourth ACM SIGCOMM Workshop Network and System Support for Games (NetGames),* pp. 1-11, 2005.

[46] S. Kulkarni, "Badumna Network Suite: A Decentralized Network Engine for Massively Multiplayer Online Applications," *Proc. IEEE Ninth Int'l Conf. Peer-to-Peer Computing (P2P),* pp. 178-183, 2009.
[47] S. Rooney, D. Bauer, and R. Deydier, "A Federated Peer-to-Peer Network Game Architecture," *IEEE Comm. Magazine,* vol. 42, no. 5, pp. 114-122, May 2004.
[48] J. Jardine and D. Zappala, "A Hybrid Architecture for Massively Multiplayer Online Games," *Proc. Seventh ACM SIGCOMM Workshop Network and System Support for Games (NetGames),* pp. 60-65, 2008.
[49] F. Chen and V. Kalogeraki, "Adaptive Real-Time Update Dissemination in Distributed Virtual Simulation Environments," *Proc. IEEE Eighth Int'l Symp. Object-Oriented Real-Time Distribute Computing (ISORC),* pp. 233-236, 2005.
[50] N.E. Baughman, M. Liberatore, and B.N. Levine, "Cheat-Proof Playout for Centralized and Peer-to-Peer Gaming," *IEEE/ACM Trans. Networking,* vol. 15, no. 1, pp. 1-13, Feb. 2007.
[51] S. Douglas, E. Tanin, A. Harwood, and S. Karunasekera, "Enabling Massively Multi-Player Online Gaming Applications on a P2P Architecture," *Proc. IEEE Int'l Conf. Information and Automation (ICIA),* pp. 7-12, 2005.
[52] M. Merabti and A. El Rhalibi, "Peer-to-Peer Architecture and Protocol for a Massively Multiplayer Online Game," *Proc. IEEE Global Telecomm. Conf. Workshops (GlobeCom Workshops),* pp. 519-528, 2004.
[53] T. Hampel, T. Bopp, and R. Hinn, "A Peer-to-Peer Architecture for Massive Multiplayer Online Games," *Proc. Fifth ACM SIGCOMM Workshop Network and System Support for Games (NetGames),* p. 48, 2006.
[54] T. Iimura, H. Hazeyama, and Y. Kadobayashi, "Zoned Federation of Game Servers: A Peer-to-Peer Approach to Scalable Multi-Player Online Games," *Proc. Third ACM SIGCOMM Workshop Network and System Support for Games (NetGames),* pp. 116-120, 2004.
[55] C. Gauthier Dickey, D. Zappala, and V. Lo, "A Fully Distributed Architecture for Massively Multiplayer Online Games," *Proc. Third ACM SIGCOMM Workshop Network and System Support for Games (NetGames),* pp. 171-171, 2004.
[56] E. Buyukkaya and M. Abdallah, "Data Management in Voronoi-Based P2P Gaming," *Proc. IEEE Fifth Consumer Comm. and Networking Conf. (CCNC),* pp. 1050-1053, 2008.
[57] A. Bharambe, J. Pang, and S. Seshan, "Colyseus: A Distributed Architecture for Online Multiplayer Games," *Proc. Symp. Networked Systems Design and Implementation (NSDI),* 2006.
[58] CCP Shadow. 60,453 Pilots: the New Eve PCU Record. CCP Games, http://www.eveonline.com/news.asp?a=single&nid=3934&tid=1, June 2010.
[59] The9, "The9 Limited Reports Fourth Quarter and Fiscal Year 2008 Unaudited Financial Results," The9, Earnings Release, http://www.the9.com/en/pdf/The9_4Q08_ER_FINAL.pdf, 2009.
[60] *Secure Hash Signature Standard,* Nat'l Inst. of Standards and Technology Std., http://csrc.nist.gov/publications/fips/fips180-2/fips180-2withchangenot ice.pdf, Aug. 2002.
[61] A. Rowstron and P. Druschel, "Storage Management and Caching in PAST, a Large-Scale, Persistent Peer-to-Peer Storage Utility," *SIGOPS Operating Systems Rev.,* vol. 35, no. 5, pp. 188-201, 2001.
[62] I. Clarke, O. Sandberg, B. Wiley, and T.W. Hong, "Freenet: A Distributed Anonymous Information Storage and Retrieval System," *Proc. Int'l Workshop Designing Privacy Enhancing Technologies,* pp. 46-66, 2001.
[63] A. Harwood and E. Tanin, "Hashing Spatial Content over Peer-to-Peer Networks," *Proc. Australian Telecomm., Networks, and Applications Conf. (ATNAC),* pp. 1-5, 2003.
[64] P. Druschel and A. Rawstron, "PAST: A Large-Scale, Persistent Peer-to-Peer Storage Utility," *Proc. Eighth Workshop Hot Topics in Oerating Systems (HotOS VIII),* 2001.
[65] M. Steiner and E.W. Biersack, "Shortcuts in a Virtual World," *Proc. ACM CoNEXT Conf.,* p. 1, 2006.
[66] F. Aurenhammer, "Voronoi Diagrams—A Survey of a Fundamental Geometric Data Structure," *ACM Computing Surveys,* vol. 23, no. 3, pp. 345-405, 1991.
[67] S.-C. Chang, "Voronoi Diagram Based State Management for Peer-to-Peer Virtual Environments," master's thesis, Nat'l Central Univ., 2008.
[68] VAST Development Team. VAST. http://vast.sourceforge.net, Dec. 2010.
[69] M. Castro, P. Druschel, A.-M. Kermarrec, and A. Rowstron, "Scribe: A Large-Scale and Decentralized Application-Level Multicast Infrastructure," *IEEE J. Selected Areas in Comm.,* vol. 20, no. 8, pp. 1489-1499, Oct. 2002.

John S. Gilmore received the bachelor's and master's degrees in electrical and electronic engineering with computer science from Stellenbosch University in 2007 and 2010, respectively. He is currently working toward the PhD degree at the MIH Media Lab at Stellenbosch University. His research interests include computer networks, distributed systems, and communication protocols. He is a student member of the IEEE.

Herman A. Engelbrecht received the PhD degree in electronic engineering from the University of Stellenbosch in 2007. He joined the Department of Electrical and Electronic Engineering at the University of Stellenbosch in 2003. His research interests include pattern Recognition, computer vision, and electronic media. He is a member of the IEEE.