Pithos: A State Persistency Architecture for Peer-to-Peer Massively Multiuser Virtual Environments

John S. Gilmore and Herman A. Engelbrecht
MIH Media Lab, Electrical and Electronic Engineering Department
University of Stellenbosch, Stellenbosch, South Africa
mail: jgilmore@ml.sun.ac.za and hebrecht@sun.ac.za

***Abstract*—An aspect of peer-to-peer (P2P) massively multiuser virtual environments (MMVEs) that has not received significant attention thus far, is how game objects can be stored in a distributed fashion, taking into account the unique aspects of MMVEs. The field is termed P2P MMVE state persistency. This paper explores the different models of state persistency for P2P MMVEs, namely super peer storage, overlay storage, hybrid storage and distance-based storage. All storage types are found to be lacking in some respects and therefore a novel storage system called Pithos is introduced. Pithos provides low latency storage for peers within the same group, while enabling higher latency, but reliable backup to a storage overlay. This is achieved by a two-tiered hybrid architecture making use of player grouping.**

I. Introduction

Peer-to-Peer (P2P) Massively Multiuser Virtual Environments (MMVEs) have received significant attention from the research community, since the first publication on the subject by Knutssonn et al. in 2004 [1]. P2P MMVEs promise to solve many issues prevalent in today's Client/Server (C/S) based MMVEs.

Recently, six key challenges of P2P systems have been identified: *interest management*, *game event dissemination*, *non-player character (NPC) host allocation*, *game state persistency*, *cheating mitigation* and *incentive mechanisms* [2]. Most of the challenges mentioned have received significant attention from the research community, with the exception of state persistency.

State persistency defines how object states should be stored. Objects states can be anything from a user's position to the state of the virtual market in an MMVE. For a P2P MMVE, game data must be distributed amongst various peers in the network. This creates challenges not usually present in classic C/S MMVEs. We've identified the main requirements of P2P MMVE state persistency in [3] as scalability, reliability, fairness, responsiveness and security.

In [3] we argue that none of the current approaches to state persistency satisfy all identified requirements. The focus of this paper is, therefore, exclusively on state persistency in P2P MMVEs. It presents a design that satisfies all the identified requirements, along with some implementation details and preliminary results. Pithos, a novel hybrid multi-tiered state persistency architecture is proposed. The novelty of Pithos lies in its support for both a responsive and a fair storage system, while also taking into account security aspects of distributed storage. To the best of our knowledge, there are some storage systems that provide responsive or fair storage, but none that provide both. No storage system, designed specifically for P2P MMVEs, have taken security into account.

If Pithos is incorporated into an existing P2P MMVE network architecture, it will add the ability to reliability share the storage load of long and short term game state. The addition of a robust state persistency mechanism, specifically designed for P2P MMVEs, will bring us one step closer to the creation of a complete P2P MMVE architecture.

Section II describes the different storage models found in the literature and describes the advantages and disadvantages of each group. Section III presents the key design characteristics the define Pithos. Section IV describes the implementation of the simulated Pithos and continues to evaluate the responsiveness and fairness of the simulation. Section V concludes with a summary of the paper and a discussion on future work.

II. Distributed state persistency models

After reviewing various storage architectures, the reviewed architectures were categorised into four broad types by which state persistency is currently achieved in P2P MMVEs [3]. The four identified storage types are: *super peer storage*, *overlay storage*, *distance-based storage* and *hybrid storage*. In order to evaluate the different storage types, the following requirements were identified:

Scalability: For an MMVE state persistency architecture to be scalable, it should be able to support thousands of users. In the paper, scalability it not handled as some separate entity, but rather all other requirements are evaluated in terms of a system of thousands of users.

Reliability: Reliability is defined to mean that a file in the storage system may neither be lost, nor be unavailable when a user requests it.

Fairness: For a system to be fair, the responsibility of storage should be equally shared amongst all users according to their available resources. This ensures that costs due to bandwidth and storage are shared amongst all.

Responsiveness: Responsiveness is a requirement that has not really been part of file storage systems in the past. For MMVEs, it is believed that responsive object storage is a

key requirement to promote responsive game play and robust recovery mechanisms.

Security Data security is a major issue for distributed storage, because data are stored on users' machines who should not necessarily be able to access and alter the data stored.

The following sub-sections briefly describe the different storage types and lists the advantages and disadvantages of each. For a thorough evaluation of the different storage types, the reader is referred to [3].

A. Super peer storage

Super peer storage uses super peers to store the game state [1]. The game world is usually divided into multiple regions, with the root objects of each region stored on different super peers. Each super peer stores the authoritive objects for its region. All peers in a region access their regional super peer to store and retrieve regional object states. In this way, super peer storage can be seen as a C/S type of storage, with super peers acting as regional servers. To improve reliability, multiple super peers per region are sometimes used, where each super peer in the same region stores the same objects as all other super peers in that region [4].

The advantage of super peer storage is responsiveness. The disadvantages of super peer storage stem from the per-region centralised approach, where super peers have access to all region data and they are the only entities in the network that store data. This leads to the main issues with super peer storage being lack of fairness and security. It is also difficult to achieve high levels of reliability.

B. Overlay storage

Overlay storage distributes the game state across a P2P overlay [5], [6], [7], [8]. Any available P2P overlay can be used, but a structured overlay is preferred because all objects are always accessible, which is not always the case in a unstructured overlay. Various distributed storage techniques are compared in [9].

Overlay storage is characterised by all nodes sharing all objects as well as multiple replicas of those objects. The storage time is $O(\log(N))$, where N is the total number of nodes in the network. Therefore, the main advantages of overlay storage are reliability, fairness and reasonable security, with the main disadvantage being lack of responsiveness.

C. Super peer-overlay hybrid storage

Hybrid storage describes any combination of the other three storage types. Only one type of hybrid storage is, however, currently found in the literature [10], [11], which is the combination of super peer and overlay storage.

Super peer-overlay hybrid storage divides the world into regions, as with super peer storage. Super peers are assigned region states to store, as with super peer storage, but the complete region state is also stored using a form of overlay storage.

Super peer-overlay hybrid storage combines some of the features present in overlay and super peer storage to have the advantages of high reliability from overlay storage and high responsiveness from super peer storage. Hybrid storage, does however, still suffer from the same security issues and lack of fairness present in super peer storage.

D. Distance-based storage

Distance-based approaches distribute objects to the closest peers in the virtual world [12], [13]. The hypothesis is that peers are interested in objects that are close to them, so the object access latency can be reduced by storing objects on nodes that are interested in them. Some storage approaches only consider player characters (PCs) for storage and do not consider non-player characters (NPCs) [14], [15]. These storage types are considered sub-sets of distance-based storage. Some specialised distance-based approaches exist that partition the virtual world into a Voronoi diagram to determine which objects are closest to which peers [16], [17].

Distance-based storage exploits the assumptions that objects close to each other in the virtual world are more likely to interact with one another and that the number of objects stored per nodes depends on the distribution of nodes in the virtual world. This leads to responsiveness being the main advantage of distance-based storage. The disadvantages are a lack of reliability and security, and that it can suffer from some fairness issues.

E. Summary and comparison

From the above discussion, it is evident that none of the mentioned storage types in their current forms are appropriate for data storage in MMVEs. Super peer storage is not fair nor secure, overlay storage is not responsive, hybrid overlay/super peer storage is not fair and distance-based storage is not secure and not yet reliable.

III. Architecture characteristics

In this section, "Pithos", the proposed P2P MMVE state persistency architecture design is described. The inspiration for this architecture come from two observations:

1) One can combine multiple storage models and arrive at a model which possesses fewer disadvantages than any of the models used.
2) Responsiveness is greatly increased in a fully distributed model, where there is no intermediate server that relays all information. However, fully distributed architectures are not scalable because the number of messages scaling by $O(N^2)$, where N is the number of nodes in the network.

Figure 1 shows the Pithos architecture. The figure shows groups of fully connected peers (light blue and dark red), where all groups are connected to each other in an P2P overlay through super peers (red).

Pithos groups peers to form a two tiered storage model. The first tier is a storage model at group level and the second is a storage model over all groups. On the first tier, which is the intra-group level, a fully distributed storage system is used to allow for highly responsive read and write operations within

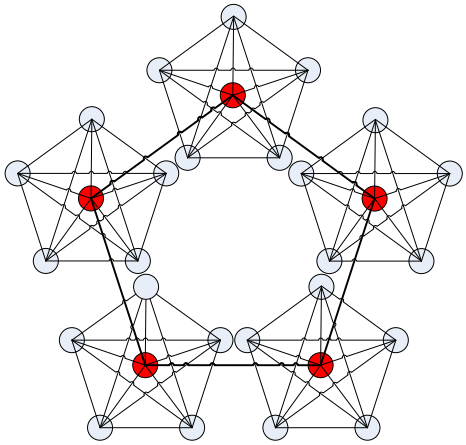

Fig. 1. Layout of the Pithos storage architecture

the group. On the second tier, which is the inter-group level, a P2P overlay is used to store data between groups.

According to categorisation of II, Pithos is a type of hybrid storage, that incorporates overlay storage and distance-based storage. Responsiveness is achieved by constructing fully connected networks amongst groups of players and then storing objects that are mostly used by the group within the group, as described in Sections III-A, III-B and III-C. Reliability is achieved by making use of replication and migration mechanisms as described in Section III-B. Security is achieved by using a certification authority to assign node IDs and signing any storage and retrieve request with the requesting node's certificate, as described in Section III-D. Fairness is achieved by having all nodes store objects, as described in Sections III-B and III-C.

A. Grouping

At the core of the architecture is the peer clustering mechanism. Two approaches are being evaluated: distributed clustering techniques (for example affinity propagation [18]) and dynamic regioning techniques (for example self-organising spatial publish subscribe (SOSPS) [19]).

Distributed peer clustering techniques: make use of the flocking behaviour of players to dynamically group players into flocks or clusters [20]. The main idea of flocking is that players move around in groups, rather than randomly on their own. It is desirable that user density within groups should remain constant, because a fully distributed architecture is not scalable. This means that groups should merge or split as the user density within them change.

Affinity propagation clusters nodes using a similarity matrix to find similar nodes. The similarity matrix may contain user positions. In this case, affinity propagation will group nodes depending on their location in a virtual world. This algorithm is ideally suited to P2P applications, since it is a distributed clustering algorithm based on message passing.

Dynamic regioning: divides the virtual world into regions that can be resized or further divided to maintain constant player densities across regions. SOSPS creates dynamic regions based on a Voronoi overlay network [21]. Near constant user density is achieved by increasing and decreasing the area sizes. This system is based on VON, a distributed Voronoi overlay network designed for MMVEs [22].

B. Replication

When storing objects in Pithos, replication is used to increase object availability under network churn and for security in the presence of malicious nodes [23]. For every object that is stored in Pithos, k object replicas are also stored. The number of replicas (k) depends on the degree of network churn as well as the number of expected malicious users in the network. If the network churn is high, more replicas are required to avoid the situation where all k peers hosting an object leaves the network before any object migration can be done.

If a node leaves the network and stops to transmit "keep alive" messages, the migration mechanism will detect this and replicate the file on another node. Replication exists intra- as well as inter-group and is useful in ensuring that if a nodes leaves the network, the data are not lost. All object requests are routed to the peer with the next closest ID if the root peer leaves, because of how overly routing functions. The new destination peers will possess the stored files, since Pithos stores overlay replicas at overlay neighbours.

Another reason to replicate game objects is to make the system more secure. If it is known that a certain percentage of users are malicious, it is advantages to have more replicas than malicious users. This will allow for a secure system where object hashes can be compared to determine which nodes are malicious and what version of an object is accurate.

C. Distance-based storage

For Pithos to succeed as an MMVE storage architecture, intra-group data requests should be preferred to inter-group data requests. This requirement, combined with the fact that the grouping algorithm geographically groups players in the virtual world, lends Pithos to a storage system based on distance-based storage. Similar to interest management, the assumption is that players have a limited area of interest and require interaction with a limited number of objects within range.

Therefore, distance-based storage is implemented on a group level rather than an individual level. This means that objects are stored on the nearest group of players, rather than the nearest user. It is assumed that such an approach will alleviate the security and reliability challenges present in distance-based storage [3].

With group-based distance-based storage, it is assumed that because peers now store objects closest to the group, the objects that they are interested in will most likely be stored within their own group. Therefore, most data requests should be intra-group requests. The overlay storage component ensures that nodes that require data, which are not stored within their group, are still able to access requested data.

D. Secure storage and node ID assignments

In order to design a secure distributed storage system, one requirement for the P2P overlay is that nodes should not be able to select their own IDs or it will not be possible to secure the system against attack. Node IDs should rather be assigned securely by some certification authority [24].

To meet this requirement, Pithos implements its own certification authority to assign node IDs securely and promote security in the P2P overlay. A certification server exists that handle ID requests from nodes. The server assigns IDs to nodes and provides the node with a signed certificate that it may use to store data.

Whenever an object is stored or updated in the storage network, nodes have to sign the object to enable the tracking of object changes throughout the life of the object. This system is very different from classic distributed file storage designs that advocate anonymity in storage. The fact that all changes can be tracked to a specific node will simplify the task of eliminating user cheating.

IV. Architecture Evaluation

Pithos is currently still a work in progress, and although the design presented in the previous section is meant to address all identified requirements, only preliminary results for responsiveness and fairness is presented in this section.

A. Simulation model

The proposed multi-tiered model is currently being implemented in Oversim [25], a P2P simulation environment based in Omnet++, which allows for the measurement of identified requirements. Furthermore, it allows for the comparison of the current model with other state persistency models. Initial results are promising, with the implemented model functioning as expected. The simulated system is very responsive when storing data within a group and as responsive as storing data in an overlay, when storing data between groups.

Pithos is designed to form part of a complete P2P MMVE network solution. It is assumed that there exists some intelligence that drives Pithos and has to determine when objects have to be stored and retrieved on each peer. This driving intelligence is termed the game layer.

In the results shown, Pastry was used as the P2P overlay and Pithos was driven by a *Game* module developed for this purpose. After a node has joined a group, the game module starts to generate store and retrieve requests at a rate of 10 objects per second and a size of 1024 bytes. The size of 1024 byte objects was chosen to be much larger than Quake 3 game objects without delta encoding, used in [26]. Pithos is designed for the low latency storage of small game objects.

For the results shown, 14499 peers, 500 super peers and a single directory server are created at the start of the simulation. The directory server publishes super peer information, which allows a peer to join the group nearest to it. Because of the way Pithos is structured, each super peer node is also a peer node, which gives a total of 15000 Oversim nodes.

To be able to simulate Pithos for 15000 nodes, it runs on the Oversim simple underlay network [27], where node latencies are determined by the distance between nodes placed in an n-dimensional Euclidean space. The positions of the nodes are chosen to match the latencies of the CAIDA/Skitter project. Different nodes are also assigned different bandwidth and jitter parameters to simulate a heterogenous network.

B. Responsiveness

To exactly compare Pithos with overlay storage, the probability that a message is routed within a group ($P(g)$) should first be known. It is expected that $P(g)$ will be different for MMVEs with different defining mechanics. It should be possible to determine $P(g)$ experimentally for a specific type of game, but this will require access to the game client of an already implemented P2P MMVE. The exact measurement of $P(g)$ is left for future work, but the responsiveness can be calculated as a function of $P(g)$. Working with a function in $P(g)$ also allows for the implementation of various dynamic strategies that can adapt to various values of $P(g)$.

We define $P(o) = 1 - P(g)$ as the probability that a message is routed within the overlay. We define T_{group} as the root and replica message distribution and T_{overlay} as the overlay message distribution, both shown in Figure 2. The expected value of the overall system response time ($E[T_{\text{resp}}]$) can then be presented as a weighted average of the expected values of both the group and overlay storage distributions, as follows:

$$\begin{aligned} E[T_{\text{resp}}] &= P(g)\left(E\left[T_{\text{group}}\right]\right) + P(o)\left(E\left[T_{\text{overlay}}\right]\right) \\ &= P(g)\left(E\left[T_{\text{group}}\right]\right) + \left[1 - P(g)\right]\left(E\left[T_{\text{overlay}}\right]\right). \end{aligned} \quad (1)$$

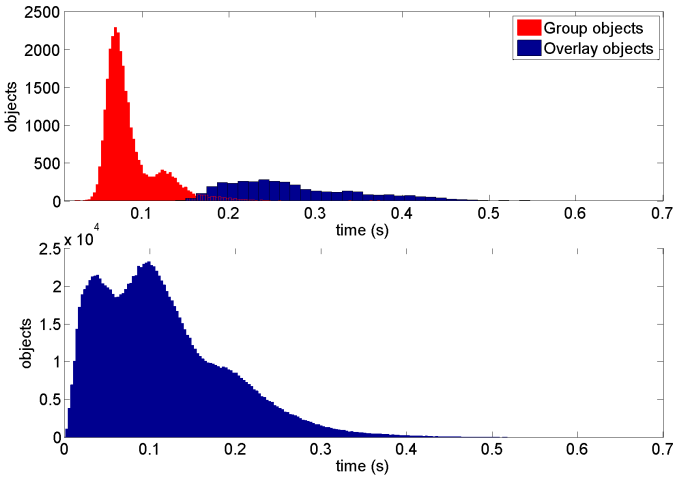

Fig. 2. (top) Time distribution of overlay and root/replica objects, (bottom) time distribution of Pastry objects.

Figure 2 (top) shows the distribution of mean storage request times over all nodes in the Pithos network for the different storage types. One can see that the intra-group root and replica objects are stored much faster ($E\left[T_{\text{group}}\right] = 0.0878s$) than the overlay objects in the network ($E\left[T_{\text{overlay}}\right] = 0.328s$). Figure 2 (bottom) presents the responsiveness of a pure Pastry network

of 14999 nodes in Oversim. The simulated Pastry network was found to have a mean routing time of $0.12s$.

The responsiveness of Pithos will depend on the responsiveness of Pastry, where the expected number of Pastry hops are given by: [23]:

$$E[H_{\text{pastry}}] = \log_{2^b}(N), \tag{2}$$

where b is a network parameter that is usually chosen as $b = 4$. From this, it is possible to calculate a theoretical performance for Pithos and compare that with a theoretical performance of overlay storage.

When using a weighted hop average, as with Equation (3), the expected number of Pithos hops is given by:

$$E[H_{\text{pithos}}] = P(g)\left(E\left[H_{\text{group}}\right]\right) + P(o)\left(E\left[H_{\text{overlay}}\right]\right), \tag{3}$$

where $E\left[H_{\text{group}}\right]$ is the expected number of group hops and $E\left[H_{\text{overlay}}\right]$ is the expected number of overly hops. In Pithos, $E\left[H_{\text{group}}\right] = 1$, because in a fully connected group any node is always one hop away from any other node.

To find the value of $E\left[H_{\text{overlay}}\right]$, one has to consider how many hops an overlay message requires in Pithos. One hop is required to send a store request from a group peer to its super peer. The super peer then forwards the message to another super peer in $\log_{16}(M)$ hops, from Equation (2), where M is the number of super peers in the network. From the destination super peer, another hop is required to send the message to the destination group peer. This gives:

$$E\left[H_{\text{overlay}}\right] = 1 + \log_{2^b}(M) + 1. \tag{4}$$

Equation (3) then becomes:

$$\begin{aligned} E[H_{\text{pithos}}] &= P(g) + [1 - P(g)]\left[2 + \log_{16}(M)\right] \\ &= 1 + [1 - P(g)] + [1 - P(g)]\left[\log_{16}(M)\right] \\ &= 1 + [1 - P(g)]\left[1 + \log_{16}(M)\right] \\ &= 1 + P(o)\left[1 + \log_{16}(M)\right]. \end{aligned} \tag{5}$$

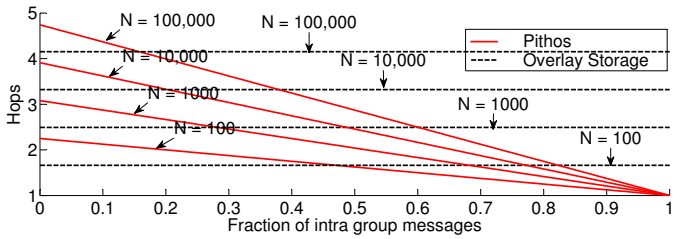

Fig. 3. Expected number of Pithos hops, compared to the expected number of overlay hops, as a function of $P(g)$ for various values of N

Figure 3 compares the expected number of Pithos hops with the expected number of overlay hops as a function of intra-group probability ($P(g)$) for various numbers of nodes (N). The overlay hops were calculated from Equation (2), while the Pithos hops were calculated from Equation (5). For the Pithos graphs, an average number of 50 peers per group was used to determine the number of super peers.

Figure 3 shows that for a low value of $P(g)$, overlay storage performs better than Pithos because of the additional two hops present in Pithos. High values for $P(g)$ are expected, because of the distance-based design of Pithos that attempts to maximise the value of $P(g)$. This should have Pithos perform better than overlay storage.

C. Fairness

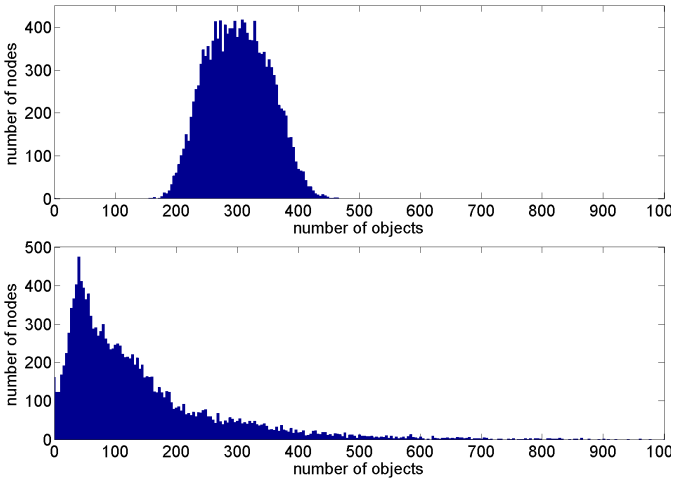

Fig. 4. (top) Root/Replica object number distribution, (bottom) overlay number distribution.

To evaluate the fairness, we evaluate the standard deviation of the number of objects stored per peer. Figure 4 (top) shows the distribution of group objects over nodes in the network. The figure shows how many nodes store how many objects. The distribution has a mean and standard deviation of 302 and 51 objects per node respectively.

Figure 4 (bottom) shows the distribution of overlay objects in Pithos with a mean and standard deviation of 153 and 189 objects per node respectively. Comparing the standard deviations of group storage to overlay storage, it appears that group storage is much fairer than overlay storage. This shows that by designing a hybrid system which prefers group storage to overlay storage, one is also designing a fairer system than overlay storage.

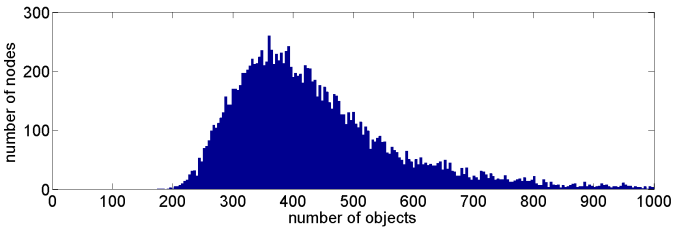

Fig. 5. Combined object number distribution

Figure 5 shows the combined object distribution of Pithos, with a mean and standard deviation of 453 and 200 objects per node respectively. This shows that the fairness of Pithos is currently dominated by the fairness of Pastry and that Pithos is as fair as overlay storage.

V. Conclusion

A. Summary

In this paper, it was found that none of the storage types reviewed satisfied all identified requirements of P2P MMVE storage systems. A novel storage architecture called Pithos was presented to satisfy all the identified requirements. The key characteristics of Pithos is that is groups peers and thereby forms a two-tiered storage architecture to promote low latency interactions amongst group peers. It employs storage replication to ensure reliable storage. It uses distance-based storage to determine on which groups objects will be stored. This is to ensure that objects stored within a group are of interest to the group. Pithos also makes use of secure ID assignments and request signing to promote storage security. For preliminary results, the requirements of responsiveness and reliability were compared to those of overlay storage and Pithos was found to be more responsive than overlay storage and as fair.

B. Future work

We plan to complete the Pithos architecture simulation and compare all requirements with all identified storage types. The simulation does not yet support network churn. A migration mechanism should still be implemented before testing under churn can be done. Another improvement is adding all peers to the storage overlay in situations where there are relatively few peers per group. This will improve responsiveness by removing the two extra hops present in Equation .

A key aspect of completing Pithos will be how players are grouped. Future research will focus on user behaviour, clustering algorithms and dynamic regioning approaches. The different clustering techniques should be compared and the most applicable one will be chosen and improved to drive Pithos.

A model of data storage and retrieval requests is also being developed. This includes sizes of objects stored, how regularly these objects are stored and what latency requirements exist for object retrieval. It is assumed that these values will depend on the specific MMVE and therefore different storage parameters should also be identified.

Acknowledgment

The financial assistance of MIH and the National Research Foundation (NRF) towards 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," in *INFOCOM*, vol. 1, 2004, p. 107.

[2] L. Fan, P. Trinder, and H. Taylor, "Design issues for peer-to-peer massively multiplayer online games," *Int. J. Adv. Media Commun.*, vol. 4, no. 2, pp. 108–125, 2010.

[3] J. S. Gilmore and H. A. Engelbrecht, "A survey of state persistency in peer-to-peer massively multiplayer online games," *Parallel and Distributed Systems, IEEE Transactions on*, vol. PP, no. 99, p. 1, 2011. [Online]. Available: http://dx.doi.org/10.1109/TPDS.2011.210

[4] M. Varvello, C. Diout, and E. Biersack, "P2p second life: Experimental validation using kad," in *INFOCOM 2009, IEEE*, april 2009, pp. 1161 –1169.

[5] S. Douglas, E. Tanin, A. Harwood, and S. Karunasekera, "Enabling massively multi-player online gaming applications on a P2P architecture," in *ICIA*, 2005, pp. 7–12.

[6] M. Merabti and A. El Rhalibi, "Peer-to-peer architecture and protocol for a massively multiplayer online game," in *GlobeCom Workshops*, 2004, pp. 519 – 528.

[7] L. Fan, "Solving key design issues for massively multiplayer online games on peer-to-peer architectures," Ph.D. dissertation, School of Mathematical and Computer Sciences – Heriot-Watt University, 2009.

[8] T. Hampel, T. Bopp, and R. Hinn, "A peer-to-peer architecture for massive multiplayer online games," in *NetGames*, 2006, p. 48.

[9] R. Hasan, Z. Anwar, W. Yurcik, L. Brumbaugh, and R. Campbell, "A survey of peer-to-peer storage techniques for distributed file systems," in *ITCC*, vol. 2, 2005, pp. 205–213.

[10] T. Iimura, H. Hazeyama, and Y. Kadobayashi, "Zoned federation of game servers: a peer-to-peer approach to scalable multi-player online games," in *NetGames*, 2004, pp. 116–120.

[11] C. Gauthier Dickey, D. Zappala, and V. Lo, "A fully distributed architecture for massively multiplayer online games," in *NetGames*, 2004, pp. 171–171.

[12] A. Bharambe, J. Pang, and S. Seshan, "Colyseus: A distributed architecture for online multiplayer games," in *Proc. of NSDI*, 2006.

[13] D. Frey, J. Royan, R. Piegay, A.-M. Kermarrec, E. Anceaume, and F. L. Fessant, "Solipsis: A decentralized architecture for virtual environments," in *MMVE*, 2008, pp. 29–33.

[14] F. Chen and V. Kalogeraki, "Adaptive real-time update dissemination in distributed virtual simulation environments," in *ISORC*, 2005, pp. 233 – 236.

[15] N. E. Baughman, M. Liberatore, and B. N. Levine, "Cheat-proof playout for centralized and peer-to-peer gaming," *Networking, IEEE/ACM Transactions on*, vol. 15, no. 1, pp. 1 –13, 2007.

[16] E. Buyukkaya and M. Abdallah, "Data management in Voronoi-based P2P gaming," in *CCNC*, 2008, pp. 1050 –1053.

[17] S.-Y. Hu, S.-C. Chang, and J.-R. Jiang, "Voronoi state management for peer-to-peer massively multiplayer online games," in *CCNC*, 2008, pp. 1134 –1138.

[18] B. J. Frey and D. Dueck, "Clustering by passing messages between data points," *Science*, vol. 315, no. 5814, pp. 972–976, 2007.

[19] S.-Y. Hu and K.-T. Chen, "Self-organizing spatial publish subscribe," in *ICAC*. ACM, 2011, pp. 171–172.

[20] J. Chen, B. Wu, M. Delap, B. Knutsson, H. Lu, and C. Amza, "Locality aware dynamic load management for massively multiplayer games," in *PPoPP*, 2005, pp. 289–300.

[21] F. Aurenhammer, "Voronoi diagrams—a survey of a fundamental geometric data structure," *ACM Comput. Surv.*, vol. 23, no. 3, pp. 345–405, 1991.

[22] S.-Y. Hu, J.-F. Chen, and T.-H. Chen, "Von: a scalable peer-to-peer network for virtual environments," *Network, IEEE*, vol. 20, no. 4, pp. 22 –31, 2006.

[23] A. Rowstron and P. Druschel, "Storage management and caching in PAST, a large-scale, persistent peer-to-peer storage utility," *SIGOPS Oper. Syst. Rev.*, vol. 35, no. 5, pp. 188–201, 2001.

[24] M. Castro, P. Druschel, A. Ganesh, A. Rowstron, and D. S. Wallach, "Secure routing for structured peer-to-peer overlay networks," *SIGOPS Oper. Syst. Rev.*, vol. 36, pp. 299–314, 2002.

[25] I. Baumgart, B. Heep, and S. Krause, "OverSim: A Flexible Overlay Network Simulation Framework," in *GI in conjunction with INFOCOM*, 2007, pp. 79–84.

[26] A. Bharambe, J. R. Douceur, J. R. Lorch, T. Moscibroda, J. Pang, S. Seshan, and X. Zhuang, "Donnybrook: enabling large-scale, high-speed, peer-to-peer games," *SIGCOMM Comput. Commun. Rev.*, vol. 38, pp. 389–400, August 2008.

[27] I. Baumgart, B. Heep, and S. Krause, "Oversim: A scalable and flexible overlay framework for simulation and real network applications," in *Peer-to-Peer Computing, 2009. P2P '09. IEEE Ninth International Conference on*, sept. 2009, pp. 87 –88.