Integration of P2P and Clouds to Support Massively Multiuser Virtual Environments

Emanuele Carlini
IMT, Lucca, Italy and
CNR-ISTI, Pisa, Italy
Email: emanuele.carlini@isti.cnr.it

Massimo Coppola
Institute of Information Science
and Technologies CNR-ISTI, Pisa, Italy
Email: massimo.coppola@isti.cnr.it

Laura Ricci
University of Pisa
Pisa, Italy
Email: ricci@di.unipi.it

***Abstract*—Massively Multiuser Virtual Environments (MMVEs) are attracting millions of players from all over the world. Currently used Client/Server infrastructures and technologies are reaching their limits of flexibility and scalability.**

We propose an approach that combines the technological advantages of two different paradigms, namely P2P networking and Cloud Computing. Our proposal leverages known P2P techniques like Virtual Nodes and consistent hashing, as well as separate overlays for different purposes, e.g. interest and object management.

We propose combining these techniques with the definition of a specific role in the overlay for Cloud-based, trusted resources. This enables the distribution of the MMVE on top of a mix of Cloud and user resources. The solution outlined allows building key-value distributed storage systems that can resize at run-time (elasticity) and provide scalability and load balancing features to the MMVE platform.

I. Introduction

A Massively Multi-User Virtual Environment (MMVE, [1]) is a synthetic world where multiple participants share the same virtual environment and interact with it and among themselves via *avatars*, virtual representations of the users.

Most of the available MMVEs as of today rely on a Client/Server (C/S) architecture. A cluster of high-end servers can provide a lot of computational power and data centres aggregate a large amount of network traffic. Such infrastructures easily manage some MMVE tasks, such as user login, state management, synchronization between players and billing. However, as the number of simultaneous players keeps growing, C/S technologies show their scalability and applicability limits. For instance, server clusters need to be bought and operated to withstand service peaks, balancing computational and electrical power constraints. A cluster-based C/S architecture concentrates all communication bandwidth needs at the data centre.

These drawbacks are incentives to investigate two orthogonal approaches: on-demand (or *Cloud*) computing and Peer-to-Peer (P2P) networks.

The exploitation of Cloud computing in MMVE [2], as in other service provisioning infrastructures, allows on-demand resource gathering according to the current load of the platform. Typically, this represents a huge saving for the service provider, as its economical plan need to account for hardware management costs which are flexible and almost proportional to the actual load, with very low fixed expenses. For MMVEs, besides server time, bandwidth costs represent a major expense when renting on-demand resources. As the MMVE scales up, these costs can significantly affect the feasibility of the on-demand approach in providing computing power. As reported in [3], if we consider an average traffic of 27TB per 12 hours and the prices of Amazon EC2 (i.e. 0.08$ per GB), the monthly fee for bandwidth is about 130000$. This kind of economic efforts can render the Cloud approach to MMVEs too costly, and prevent some game providers from joining the market.

In order to exploit on-demand computing in a MMVE, ideas from P2P-based MMVE architectures [4] can yeld concrete advantages. P2P infrastructures are inherently scalable, and may relieve the load on the MMVE servers by exploiting the capacity of the peers. If an host fails, the network is able to self-repair and reorganize, providing robustness to the MMVE. Network traffic is distributed among the users involved. These properties pair with little costs for the MMVE operator.

However, with the application of P2P techniques a number of problems arise in maintaining all of the basic structural requirements of MMVEs [5]. Mechanisms have to be added to known P2P architectures to ensure persistence of the game state, because when users leave the system they remove both their resources and the data they have managed so far, unduly stressing the self-repair features of the overlay. The lack of a central authority makes it quite hard to enforce security and soundness of updates to the system state at any time. Moreover, users machines typically have strict constraints on computational power and network capability, which may make some of them harder to exploit. Solving these problems with an acceptable overhead requires a very specific P2P approach to suit MMVEs.

In this paper we propose to employ several existing and new P2P techniques together in order to allow synergistic exploitation of P2P and Cloud features. The main ideas can be summarize as follows.

- An MMVE architecture based on P2P distributed memory spaces, used to store data items by the MMVE; the concept of Virtual Nodes is exploited in order to ease load balancing, fault tolerance and system robustness.
- Decoupling of interest management and object management over two distinct data services on top of the P2P overlay. By separating the effort due to object movement from the actual management of the object state, we increase the

performance and scalability of a distributed MMVE.

- The proposed mechanism is aware of different classes of resources, and is flexible enough to allow heterogeneous nodes to join the P2P network, by selectively exploiting end-user provided resources and Cloud-based ones.

Sect. II recalls main MMVE concepts and requirements, and Sect. III discusses closely related works. In Sect. IV we describe the high-level architecture, whose possible implementation and deployment are further detailed in Sect. V.

II. Background

In the following we include player avatars, NPCs and all items in the virtual environment in the set of *game objects*, which are characterized within a 2D virtual world by a pair of coordinates and an object state. The simulation is an iterative process, divided in time intervals of a fixed length. During each interval the game server(s) typically (1) receive events from the connected users, (2) process these events, computing a new state for some objects and (3) broadcasts the new state of the simulation to the connected users.

Currently, the C/S architecture is by far the most common architecture for commercial MMVEs. In order to run the whole range of services provided by commercial MMVEs, a C/S architecture can be extended to exploit a set of servers running on top of federated machines, such as a dedicated server cluster or Cloud-based resources.

Depending on the type of the MMVE, communication among servers may be required. *Distributed server architectures* exploit communication among servers to provide the players with the ability to share a vast virtual world, such as in Role-Playing Games (RPG). *Replicated servers architectures* replicate the virtual world for different groups of users and require no communications among distinct servers, easing to achieve shorter response times e.g. in First Person Shooter (FPS) and Real Time Strategy (RTS) games.

P2P architectures share the workload of the simulation among the peers, which usually act as servers of small virtual world regions. *Hybrid* P2P architectures exploit only a subset of the user resource (i.e. the *super peers*) to compute the simulation. Conversely in *pure* P2P architectures all the peers perform a role in the computation of the simulation.

All MMVEs share a common set of basic requirements which we quickly summarize in the following.

Scalability By scalability we consider the ability of a MMVE to react to an increasing load without jeopardizing the quality of the interactive experience. In a MMVE the load is generated by the following factors [6] : (i) the size of the virtual world, (ii) the number of entities, (iii) the density of the objects, and (iv) the amount of interactions.
The first three factors are related to the placement of the objects into the virtual world. C/S architectures usually divide the virtual world into *regions*. In particular, techniques as *zoning*, *mirroring*, and *instancing* aim at balancing the workload between servers, in order to increment scalability (see [6] for a detailed description of these methods).
In P2P architectures most solutions exploit a distribution of the load that takes in account heterogeneity of the peers in terms of bandwidth and computational capabilities. Also because of the unpredictable pattern of join and leave of the peers, robust fault tolerance mechanisms should be considered to guarantee persistence and availability.
The amount of interactions among players depends on the kind of the simulation genre [6]. For instance, fast-paced games as FPS are likely to have higher amount of interactions than RPGs, where the interactions are scattered. A proper selection of the virtual world objects from which receiving updates helps to reduce the amount of interactions.

Cheating Mitigation Cheating mitigation is an important aspect in those MMVEs where a certain degree of competition among users is expected. The MMVEs businesses model is strictly related to the number of the users, thus maintaining a secure and fair (thus enjoyable) user experience is a core task. In C/S all the servers are trustworthy, hence cheating mitigation mechanisms are applied in a straightforward way. Conversely, P2P-based architectures incur in the additional problem of the individuation of the trustworthy peers that take the role of the servers.

Persistence and Availability Players pay regular fees to access a commercial MMVE. Discontinuity in the service, such as servers failures and login problems are perceived as annoying problems by the users. Persistence and availability are implicitly guaranteed in C/S as all the data are stored in central and accessible databases. In P2P-based architectures, data can be stored using a distributed storage infrastructure, for example by indexing the content by the geographical area in the virtual world to ease the retrieve. However, even if the design of such infrastructures does exist, it is not clear if they are able to support frequent reading and writing which are typical of MMVEs.

Interest management Interest Management (IM) [7] is a well studied topic in research on distributed environments. Any single entity or player does not need to know the whole world state, and is instead only interested in those events that happen inside her Area Of Interest (AOI). IM means defining the shape and content of each entity's AOI. As relevant information has to be gathered and transmitted to the entities, IM can deeply influence the MMVE architecture. C/S system are less affected, as IM is managed internally, powerful servers and ad-hoc connections are available. In this case, IM is mainly relevant to reduce the bandwidth toward the users.
These assumptions are unrealistic for P2P-based architecture, where the two common approaches to IM are the region-based one and the spatial-based one. Spatial-based methods require that each peer maintains a connection with each avatar's peer in the AOI. This model permits fine-grained filtering of the entities, but assumes each peer is able to compute the AOI by itself. Region-based methods are simpler. IM for each region is delegated to the region server. However, an avatar's AOI may overlap with multiple regions, requiring either the peer to contact a variable number of region servers or a communication among the region servers involved.

Object allocation The problem of objects allocation is to find the proper match between the virtual world's objects and the servers in distributed architectures (P2P-based and distributed server).

Object allocation strategies are affected by the IM system used. Again, common solutions can be classified into region-based and spatial-based.

Region-based solutions rely on the concept of region server, which owns all the objects in a region. These approaches are widespread in distributed servers and hybrid P2P architectures, and are easy to implement as they closely match region-based IM. The main drawback is load balancing, as regions with a high density of objects may saturate the bandwidth capability of a server or peer. In P2P approaches, the issue is mitigated by a proper selection of the region servers.

Spatial models are usually exploited in pure P2P approaches. The asymmetry between entities, which need state updates, and objects, which need to be stored somewhere, distinguishes spatial IM and spatial object allocation. Each object is managed by its closest peer, leading to a spatial partitioning of the world among the peers (e.g. Voronoi diagrams). P2P spatial approaches are vulnerable to overheads caused by non-uniform object distributions and ownership changes.

A. Virtual Nodes and DHT approaches

Virtual Nodes [8] (VNs, also called virtual servers) were introduced with Chord [9] as a mechanism to improve load balancing in Distributed Hash Table (DHT) approaches.

In a DHT P2P network of N nodes, the DHT address space is partitioned among the nodes. The topology of inter-node connections allows to reach any node within $O(logN)$ hops, and limits the amount of connections per peer to $O(logN)$. When nodes are physical, adding and removing nodes requires some repartitioning of the address space among the nodes.

When using VNs, each one is in charge of an address range as if it were an ordinary DHT node. However, VNs are not permanently associated with physical resources, and each physical node can host several VNs. A physical node entering the system becomes responsible of a set of non contiguous VNs (see also §IV-C) which migrate to the new peer.

All standard techniques as node replication can be applied to VN approaches. There is a tradeoff in implementation complexity, as nodes need to manage multiple VNs, but the address space partitioning among VNs may be kept static. The VNs approach has some evident advantages related to static and dynamic load balancing. **(A)** More powerful hosts may receive an higher number of VNs than less powerful ones. **(B)** Heavy loaded nodes may trade VNs with unloaded ones. **(C)** In the case of a physical node failure, its VNs are possibly transferred/reassigned to different, unloaded physical nodes, reducing the risk of hot spots and overloaded nodes.

As it will be important in the following, load balancing in DHTs is also affected by the choice of the hash function mapping the objects into the DHT address space. Random hash functions can uniformly spread the objects, while locality sensitive and locality preserving hash functions increase locality, speeding up related queries at the cost of possibly unbalancing the object distribution.

III. Related Work

Many MMVE architectures have been introduced to overtake the inherent limitations of C/S infrastructures. Pure P2P approaches [10], [5], [11], [12], [13] rely on the distribution of the state of the virtual world on user's node.

In particular, VoroGame [13] uses two different logical structures for the interest management and objects allocation. VoroGame considers a pure P2P network where the peers belong to two different overlays: a structured overlay (DHT) for the data distribution and a Voronoi-based overlay for the IM. Data distribution in the DHT is done randomly, all the peers managing some part of the world state. Locality-based filtering on the objects is computed by the Voronoi overlay, which exploits a spatial division of the world. The approach suffers the problems typically related to pure P2P architectures, namely heterogeneity and unpredictability of the peers.

Due to these problems, several approaches provide solution where servers are paired with user's resources [14], [15], [16], [17]. One of the earlier approach is HYMS [14] that combines peer and server resources to manage a MMVE simulation. It divides the world into square regions which are possibly assigned to peers running on end-user resources. HYMS explicitly manages peer heterogeneity. The computational overhead for a region is given to the first peer with enough computational power and bandwidth capabilities that enters into a region. A fixed number of peers which afterwards enter the region may act as secondary replicas, in order to increase failure robustness. The peer managing a region is interested in the region itself, and this may a be security problem, as "user" peers can exploit their "manager" role to gain unfair advantages. Also, in HYMS the size of the regions is fixed, which can limit the effectiveness of the load balancing.

The concept of VN is currently exploited in commercial applications such Dynamo [18]. In Dynamo, data are distributed into an address space by using consistent hashing and address space is then assigned to a set of Virtual Nodes. Each object is paired with a *key* belonging to the address space, through the application of a hashing function. The address space is structured as a ring, and keys are assigned to the first VN encountered by a clockwise walking of the ring (i.e. the successor VN), such as in the Chord DHT. Dynamo's data partitioning schema proved to be flexible enough to provide scalability and robustness, which are two basic requirements for a MMVE architecture. However, replicas are managed according the *eventual consistency* model, which is unsuitable for MMVE, given its fast evolving nature.

IV. Architecture Description

The architecture we describe combines several previously known ideas, stemming from the choice of adopting Virtual Nodes. As we combine these approaches, we extend them

	Cloud Resources	User Resources
performance	reliable	unreliable
security	controlled environment	open environment, vulnerabilities
scalability price	costly	no cost
availability	reliable	high churn

TABLE I
CLOUD RESOURCES VERSUS USER RESOURCES.

(§IV-B, §IV-C) in order to meet the specific requirements of MMVEs.

A. Scalability and Load Balancing

In order to achieve load balancing we have to define a load measure for VNs (see §V-B) as well as a strategy for moving VNs among nodes (initial VN mapping may be a special case of this). The chosen number of VNs in the system, if we assume it to be constant, affects the load balancing and scalability features of the system. Load balancing can be ineffective if the number of VNs is too close to the number of physical nodes (also impairing the ability of the MMVE system to recruit more resources), and finding a new mapping for a VN without violating any constraint (see §IV-C) can be difficult if all physical nodes manage too many VNs. The scalability of the system is thus bounded between these two phenomena.

The amount of VNs per node (i.e. the node load) dynamically varies at run time according to factors like hardware capabilities and resource trust. Thus the overall number of VNs controls the grain of the resource load. With more VNs and a smaller grain, we achieve finer load balancing (and higher VN overhead). We are not considering dynamic approaches that can split VNs, as the management overhead seems to be even higher.

B. Backup Virtual Nodes

The mapping strategies for VN will be different for trusted and untrusted nodes. Table I summarizes the main differences between cloud and user resources. In order to cope with the user-provided resource limitations of trust and performance, we propose to identify two special types of VNs.

Any VN that is assigned to a user resource will be classified as an *unreliable Virtual Node* (uVN). An uVN is always specially replicated (see Figure 1). We call the special replica called *backup Virtual Node* (bVN). A bVN is always assigned to a trusted resource, either an on-demand resource or a provider's own resource. The uVN, hence the user resource, performs several tasks: (i) it maintains the topology connections, (ii) it replies to the requests of the users, and (iii) it sends a periodic copy of the objects' state to its bVN.

While some of the VN will still reside on trusted resources, the presence of uVN and bVN provides some concrete advantages and opportunities. First, backup data are managed by a trusted resource, to allow recovering from failures of untrusted resources. Second, the bVN can assume the role of the referee to handle cheating. Using bVNs for cheating

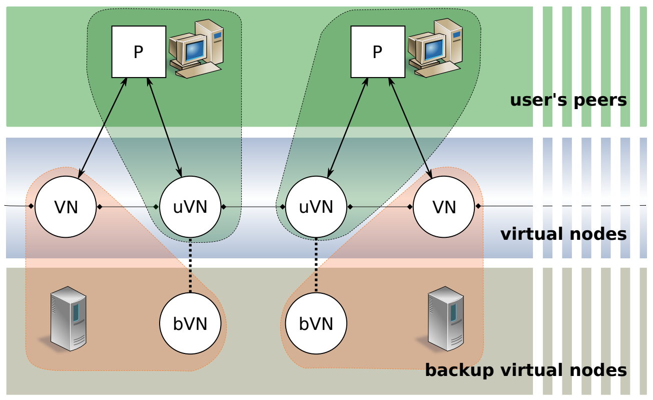

Fig. 1. High-level overview of the architecture. Each user's peer P connects to VNs (or uVNs) to access the MMVE content. The bVN layer provides fault resilience and security.

mitigation requires that the workload imposed by the role of referee is less than the workload generated from the full-blown server (VN) role. This assumption will often be true, as many referee-based strategies only perform random-sampling checks.

C. Fault tolerance

The proposed architecture provides two layers of replication. A fine-grain data replication mechanism copies each object assigned to a VN in several other VNs that follow the first one in the object address space. A coarse grain VN replication strategy copies the whole VN state.

Fine-grain. Most of the DHTs use data replication to increase failure tolerance [19]. When a node N_A leaves the network, the DHT forwards the request for objects managed by N_A to the node N_B that maintains a replica of the objects of N_A. However data replication shows some limits when the departure rate of nodes is high [20]. As N_B handles an extra load, the departure of N_A may prevent N_B from complying with the interactivity requirements of the MMVE.

Coarse-grain. VN replication copes with the unreliability of untrusted resources, preventing unexpected peaks of load due to user nodes leaving the network. With reference to Figure 1, when a node N_A that manages an uVN departs from the network, the users connected to N_A are assigned to the node that manages the correspondent bVN, N_C. Two entities may realize about the involuntary departure of N_A: N_C, and the neighbour of N_B in the address space, which maintains replicas. Each of such entities is able to inform the user peers to forward their requests at N_C.

In VN-based data partitioning it is important that no physical node manages an object as well as some of its replicas which reside on separate VNs. We follow the object replication model used by Chord, exploiting successive VNs, as it makes easy both finding the VN handling object replicas, and generate incompatibility constraints among VNs in order to guide VN (re)mapping. E.g. with replication k in the DHT, hosting VNs which are at least k positions apart is a sufficient constraint to ensure that no physical node is able to manage any object twice.

V. Implementation Issues

This section describes how to deploy **object** and **interest management services** on top of the architecture presented in the previous section. Following the discussion in §II, the objects of the MMVE are mapped onto two different logical address spaces, possibly on separate DHTs.

The Object Space (OS) maps the objects into the DHT using an uniform hash function (i.e. SHA1). Each node acts as a server for the objects in its address space, resolving conflicts and broadcasting state updates. Each peer maintains a connection with the nodes of the DHT managing objects in its AOI. Since the object positions in the OS are unrelated to their position in the virtual world, their movement in the MMVE requires no transfer between nodes and introduces no overhead. On the other hand, interest management may require a huge amount of accesses to the DHT, since objects in the AOI of a peer may be potentially spread on a large amount of DHT nodes.

To address this issue we define a further address space, the Interest Management Space (IMS). IMS stores only the virtual world coordinates of the MMVE objects, and is exploited to perform interest management. The objects are mapped to the IMS space by using a locality-preserving hash function, which with high probability maps close objects to the same DHT node. Since the position of each object O determines its address in the IMS space, a movement of O may require the migration of the object between nodes. The overhead is minimized as only the object's coordinates in the MMVE have to be transmitted.

Let us now describe in more details the operations on the DHTs. When a peer P enters the simulation, P sends its initial position to the IMS space in order to find the objects in its AOI. Afterwards the identifiers of the objects are sent to the OS space to gather their state. A parallel search of the objects within the OS space may reduce the delay of retrieving their state. The main problem of this approach may be the latency of the object transfers. Even if a delay is acceptable during the login phase, the delay for accessing the two DHT during game-play may jeopardize the quality of the simulation. For this reason, the definition of a set of prefetching techniques should be defined. Each peer should prefetch the objects located beyond its AOI before they actually enter in the AOI.

A. Locality-aware mapping of VNs

As we pointed out in the previous section, mapping a VN onto an user resource (in the following we refer to user resources as URs) is no trivial task. The problem can be specified as follow: there exists a list $l_{ur} = \{UR_1 \dots UR_n\}$ of URs candidates to act as servers, and a list of peers $l_p = \{P_1..P_m\}$ that access to the objects managed by a specific VN, v.

We look for a mapping of v on a specific UR_x which satisfies a set of constrains related to the hardware capabilities of the physical nodes (processing power to perform the MMVE computation associated to the VNs, and bandwidth to receive and broadcast all the updates, the security level they offer, and to the latency and bandwidth they can provide with respect to the peer).

In particular, our aim is to minimize the latency between a request from a peer and the reply from UR_x. Measuring the latencies among all pairs of peers and URs is of course unpractical. Network Coordinates Systems (NCSs) [21] create a two-dimensional representation where a point is associated to each peer, and the distance between two points is roughly proportional to the latency between the peers. NCSs allow to predict network latencies without performing direct measurements and to optimize network usage due to a minimal monitoring effort.

Even knowing all the inter-peer latencies, finding the best mapping is still a difficult task to perform in real-time. A way to approximate the problem is to create a Voronoi Diagram from the NCS two-dimensional area. A Voronoi diagram [22] (also Voronoi tessellation) is a decomposition of a metric space determined by the distances of the points of the space from a specified discrete set of objects in the space, i.e. the *sites*. Given a set of N sites on a two-dimensional euclidean plane, a Voronoi diagram partitions the plane into N non-overlapping regions, each one a convex polytope containing all the points closer to that region site than to any other one.

A Voronoi tessellation built on NCS locations permits to obtain easily the closer UR for a set of peers. In [22] the functionalities of the *VAST* library have been shown to support the construction and management of Voronoi Diagrams, and to incur in overheads compatible with the real-time application like MMVEs.

Voronoi graphs allow us to choose among different strategies when selecting the proper UR to map a VN. The main examples are *most crowded* (the chosen UR is the site with the larger amount of peers in its area) and *average closeness* (the chosen UR minimizes the latencies with the peer in its area).

B. Virtual Nodes Load Measurement

We consider a VN receiving a number of writes for an object o from a set $l_p = \{P_1..P_n\}$ of peers. The VN resolves the conflicts, it computes the new state for o and it sends the updated version of the object at each P. The node that manages the VN needs to support a load in terms of incoming and outgoing bandwidth.

The modification rate for the object o (stored in the VN) depends on the number of peers that write o during a simulation interval T. For the sake of simplicity, we assume the following: (i) each peer modifies o once per T and (ii) the peers that can modify an object are also the only that receive updates for such object. Following these assumptions the average incoming and outcoming bandwidth requirement for each interval is:

$$B_{in}(o) = B_{out}(o) = s(o) \times |l_p| \tag{1}$$

where $s(o)$ is the size of the object's state. With $s(o) = 100$ bytes and $|N| = 10$ nodes and $T = 100ms$ the bandwidth requirement is $80Kbps$ just for a single objects. In order to manage consistency of the objects' state between the server and the users' replicas, we consider Vector-Field Consistency (VFC) [23]. The VFC exploits a 3-dimensional consistency vector. Each dimension bounds the maximum divergence with respect to a view of an object: (i) the *time* an object can withstand without being refreshed with its last value, (ii) the *sequence*, as the maximum amount of tolerated lost updates, and (ii) the *value*, as the maximum allowed difference between the replicas' value.

The VFC decreases the bandwidth requirement for the objects that are at the border of the avatar AOI, by a factor that depends on the values assigned to the VFC:

$$B_{out}(o) = \sum_{i=0}^{n} f(VFC_{n,o})s(o) \tag{2}$$

where $0 \leq f(VFC_{n,o}) \leq 1$ is the bandwidth coefficient reduction given a VFC at node n for the object o.

VI. Conclusion

We believe that the combination of cloud computing and P2P resources is eventually needed to realize MMVEs that are scalable to larger and larger set of users, and to allow dynamic resizing of the MMVE servicing platform.

In this paper we proposed a basic architecture for a flexible and scalable management of MMVE simulation, by composing previous techniques from the P2P field, and by specializing those techniques to allow exploiting a mix of heterogeneous resources in such time-critical applications.

There are still many open issues in the framework we devised. Designing and testing the Virtual Nodes allocation and monitoring policies, with respect to the scalability and tune-ability of the resulting platform, is a new direction for MMVEs, and it is also strongly affected by our introduction of special "backup Virtual Nodes". These bVNs are the key to exploiting trusted, reliable resources from the Cloud to harness and control the transient and unreliable user resources that we want to exploit for simulation management. The ultimate goal is to amortize the cost of on-demand resources by leveraging the large set of resources volunteered by the end-users. We intend to investigate on the optimal trade-off between the direct costs of renting resources and the indirect costs of the monitoring and back up activities required by the use of unreliable resources.

Acknowledgment

Authors acknowledge the support of projects XtreemOS, "Building and Promoting a Linux-based Operating System to Support Virtual Organizations for Next Generation Grids", grant FP6-033576, and CONTRAIL, "Open Computing Infrastructure for Elastic Services", grant FP7-257438, and the anonymous reviewers for useful comments.

References

[1] S. Hu, "Spatial Publish Subscribe," *Proc. of IEEE Virtual Reality workshop, Massively Multiuser Virtual Environment (MMVE09)*, 2009.

[2] A. Shaikh, S. Sahu, M.-C. Rosu, M. Shea, and D. Saha, "On demand platform for online games," *IBM Systems Journal*, vol. 45, no. 1, pp. 7–19, 2006.

[3] K. Chen, P. Huang, and C. Lei, "Game traffic analysis: An MMORPG perspective," *Computer Networks*, vol. 50, no. 16, pp. 3002–3023, 2006.

[4] A. Chen and R. Muntz, "Peer clustering: a hybrid approach to distributed virtual environments," in *Proc. of 5th ACM SIGCOMM workshop on Network and system support for games*. ACM, 2006, p. 11.

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

[6] R. Prodan and V. Nae, "Prediction-based real-time resource provisioning for massively multiplayer online games," *Future Generation Computer Systems*, vol. 25, no. 7, pp. 785–793, Jul. 2009.

[7] J.-S. Boulanger, J. Kienzle, and C. Verbrugge, "Comparing interest management algorithms for massively multiplayer games," *Proceedings of 5th ACM SIGCOMM workshop on Network and system support for games - NetGames '06*, p. 6, 2006.

[8] A. Rao, K. Lakshminarayanan, S. Surana, R. Karp, and I. Stoica, "Load Balancing in Structured P2P Systems," in *Peer-to-Peer Systems II*, ser. LNCS. Springer, 2003, vol. 2735, pp. 68–79.

[9] I. Stoica, R. Morris, D. Liben-nowell, D. R. Karger, M. F. Kaashoek, F. Dabek, and H. Balakrishnan, "Chord: A Scalable Peer-to-Peer Lookup Protocol for Internet Applications," *IEEE/ACM Transactions on Networking (TON)*, vol. 11, no. 1, pp. 17–32, 2003.

[10] D. Frey, J. Royan, R. Piegay, A. Kermarrec, E. Anceaume, and F. Le Fessant, "Solipsis: A decentralized architecture for virtual environments," *Proc. of 1st Intn.l Workshop on Massively Multiuser Virtual Environments*, pp. 29–33, 2008.

[11] B. Knutsson, H. Lu, W. Xu, and B. Hopkins, "Peer-to-peer support for massively multiplayer games," *IEEE INFOCOM*, vol. 1, pp. 96–107, 2004.

[12] A. Bharambe, J. Douceur, J. Lorch, T. Moscibroda, J. Pang, S. Seshan, and X. Zhuang, "Donnybrook: Enabling large-scale, high-speed, peer-to-peer games," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 4, pp. 389–400, 2008.

[13] E. Buyukkaya, M. Abdallah, and R. Cavagna, "VoroGame: A Hybrid P2P Architecture for Massively Multiplayer Games," *6th IEEE Consumer Communications and Networking Conf.*, pp. 1–5, 2009.

[14] K. Kim, I. Yeom, and J. Lee, "HYMS: A Hybrid MMOG server architecture," *IEICE Transactions on Information and Systems*, vol. E87, pp. 2706–2713, 2004.

[15] J. Jardine and D. Zappala, "A hybrid architecture for massively multiplayer online games," *Proc. of the 7th ACM SIGCOMM Workshop on Network and System Support for Games - NetGames '08*, p. 60, 2008.

[16] T. Iimura, H. Hazeyama, and Y. Kadobayashi, "Zoned federation of game servers: a peer-to-peer approach to scalable multi-player online games," in *Proceedings of 3rd ACM SIGCOMM workshop on Network and system support for games*. ACM, 2004, pp. 116–120.

[17] S. Rooney, D. Bauer, and R. Deydier, "A federated peer-to-peer network game architecture," *IEEE Communications Magazine*, vol. 42, no. 5, pp. 114–122, 2004.

[18] G. DeCandia, D. Hastorun, M. Jampani, and G, "Dynamo: Amazon's highly available key-value store," *ACM SIGOPS*, pp. 205–220, 2007.

[19] S. Ktari, M. Zoubert, A. Hecker, and H. Labiod, "Performance evaluation of replication strategies in DHTs under churn," in *MUM '07: Proc. of the 6th Intn.l Conf. on Mobile and Ubiquitous Multimedia*. New York, NY, USA: ACM, 2007, pp. 90–97.

[20] S. Rhea, D. Geels, T. Roscoe, J. Kubiatowicz, and I. Research, "Handling Churn in a DHT," in *ATEC'04: Proceedings of USENIX Annual Technical Conference*. USENIX Association, 2004, p. 10.

[21] B. Donnet, B. Gueye, and M. A. Kaafar, "A Survey on Network Coordinates Systems, Design, and Security," *IEEE Comm. Surveys & Tutorials*, pp. 1–16, 2010.

[22] L. Genovali, "A Voronoi Based Framework for the Definition of P2P Distributed Virtual Environments," in *Proc. of IEEE ICUMT*, 2009.

[23] N. Santos, L. Veiga, and P. Ferreira, "Vector-field consistency for ad-hoc gaming," *Lecture Notes in Computer Science*, vol. 4834, p. 80, 2007.