P2P Second Life: experimental validation using Kad

Matteo Varvello[†⋆], Christophe Diot[†], Ernst Biersack[⋆]
[†] Thomson, Paris, France
[⋆] Eurecom, Sophia-Antopolis, France
{matteo.varvello,christophe.diot}@thomson.net, ernst.biersack@eurecom.fr

***Abstract*—Applications such as Second Life require massive deployment of servers worldwide to support a large number of users. We investigate experimentally how Peer-to-Peer (P2P) communication could help cut the deployment cost and increase the scalability of Social Virtual Worlds such as Second Life. We design and build a communication infrastructure that distributes the management of the virtual world among user resources using a structured P2P network. Our communication infrastructure is implemented on the top of Kad, the P2P network that supports millions of eMule users. We then use avatar and object traces collected on Second Life to perform a realistic emulation of P2P Second Life over the Internet. We show that, despite using a standard P2P solution, P2P Second Life is mostly consistent, persistent and scalable. However, the latency avatars experience to recover from an inconsistent view of the virtual world can become disturbing for very large numbers of participants and objects. We analyze and discuss this limitation and give recommendation on how to design P2P Social Virtual Worlds.**

I. Introduction

Social Virtual Worlds (SVWs) are networked virtual environments where people can invent and emulate a new social life. Second Life[1] (SL), launched in 2003 by Linden Lab, has become the most popular SVW, reaching 14 million users in June 2008. SL consists of virtual regions where users interact via their digital representation called *avatar*. The main innovative feature of SL is that avatars can participate in the design of the virtual environment by creating *objects* such as cars, trees, and buildings. Thus, SL differs from on line games where the virtual world is mostly static.

SVWs must exhibit three major properties. First, all users must have a *consistent* view of the world. Second, no object should be lost; we call this property *persistency* of the virtual world. Third, the application must be *scalable*, i.e., the performance of the SVW must not be affected by the number of avatars and objects.

Today, SVWs are implemented using a client/server architecture. A server is used to maintain a unique state of the virtual world and to distribute it to the users. Client/server architectures are naturally consistent and persistent when communication is reliable. However, scalability is an issue as it depends on the number of servers deployed to support an unpredictable number of avatars and objects. In SL, the scalability issue is addressed by dividing the virtual world into *regions* that are each associated to a dedicated server. The number of avatars per region is limited to 100 and in case of server overload, the virtual world clock is slowed down. This strategy turns to be very inefficient as the popularity of regions is highly variable [15].

A scalable alternative to client/server is to use a Peer-to-Peer (P2P) approach, where the virtual world is maintained in a distributed way using SVW user own resources. If we make the assumption that each user joins with enough resources to maintain its "share" of the virtual world, scalability should come "for free". A P2P SVW only requires a small number of servers for access control, security and bootstrap. These servers do not need to be located close to the users as they do not play any active role in the virtual world simulation. However, object consistency and persistency may suffer from the unpredictable behavior of peers joining and leaving at any point in time. Security might also be a issue [1]; we do not address it in this paper.

In this work, we show experimentally that it is possible to guarantee a very high level of *object* consistency and persistency in Second Life using a standard P2P architecture already deployed over the Internet. The management of the interaction among avatars is beyond the scope of this paper and left for future work.

Specifically, we design a new communication infrastructure for SVWs where users (i.e., peers) are organized via a structured P2P network (Section 2). This P2P architecture is inspired from CAN [11] and PHT [10]. We integrate this P2P architecture on the top of Kad, the P2P network formed by eMule clients[2]. Thanks to Kad, we can perform a realistic emulation of SL on the Internet using the resources provided by eMule users. We emulate P2P SL using avatar and object traces collected in a region of SL (Section 3).

Our methodology is original and we believe that it is an important contribution of this work. To the best of our knowledge, this is the first work to directly exploit Kad as an experimental platform. However, the major contribution is the experimental evaluation of a P2P architecture for SVWs on the Internet, using SL object and avatar traces.

We show (Section 4) that it is possible to construct a consistent, persistent, and scalable P2P version of SL on top of a structured P2P network. However, in case of large number of objects, avatars can experience a long latency to recover from an inconsistent view of the virtual world. Moreover, search inefficiency in Kad introduces additional latency. Based on these observations, we discuss how to design a P2P overlay that would eliminate the limitations of the current architecture.

[1]http://www.secondlife.com/

[2]http://www.emule.com/

II. A PEER-TO-PEER INFRASTRUCTURE FOR SVW

In this Section, we design a P2P communication infrastructure for SVWs. First, we describe the mechanisms we use to share the virtual world among participants using a structured P2P network. Then, we explain how we adapt our architecture to work on Kad as there is no possibility to modify the Kad routing algorithm.

Note that we do not claim that this P2P communication infrastructure is completely innovative. It has been designed to allow realistic experiments on the Internet and to help us understand how to design an efficient P2P architecture for SVWs. The innovation lies in the way we adapt Kad to support a P2P SVW on the Internet.

A. Adaptive Cell-Based Management

We introduce here an *adaptive cell-based* partition of the virtual world. This simple mechanism is inspired from the scheme used by CAN [11] to dynamically share the load in the hash space. In the context of SVWs, we use a similar approach to dynamically adapt the virtual world division to the objects distribution.

We call a *cell*, denoted by C_i^l, a portion of the SVW. We call $N_o(C_i^l, t)$ the number of objects contained in C_i^l at time t. To start with, the virtual space is identified by a single cell $C_0^0 = \Omega$. Then, C_0^0 is successively divided into multiple cells such that $\Omega = \bigcup_{\forall(i,l)} \{C_i^l\}$. Cells are split when the number of objects they contain exceeds a given threshold D_{max}. Therefore, D_{max} is the maximum number of objects that can be present in a cell. We call D_{min} the minimum number of objects contained within two adjacent cells.

Whenever a user notices that $N_o(C_i^l, t) \geq D_{max}$, it performs a *split* operation that creates cells $C_{(2i+1)}^{(l+1)}$ and $C_{(2i+2)}^{(l+1)}$. In the same way, when $(N_o(C_{(2i+1)}^{(l+1)}, t) + N_o(C_{(2i+2)}^{(l+1)}, t)) \leq D_{min}$, a *merge* operation is performed originating cell C_i^l.

Split and merge operations are accomplished according to a well defined order. Assuming a two dimensional space, a cell is first split on the vertical dimension, then on the horizontal, and so on.

Figure 1 shows an example of the evolution of a cell-based SVW with $D_{max} = 3$. In Figure 1(a), two objects are created in the original cell. Then, in Figure 1(b) two other objects appear. Given $D_{max} = 3$, the first split operation is performed. In Figures 1(c) and 1(d), we show how the adaptive cell-based management incrementally partitions the virtual world according to the distribution of objects.

B. Virtual Space and Distributed key-space

Structured P2P networks provide a key-based routing (KBR) layer to their peers [4]. Peers and content are each identified by a key in a large identifier space, e.g., the key-space. The KBR layer allows to store/retrieve $< key, value >$ pairs. According to the specific protocol, different techniques are used to maintain the consistency and persistency over time of the $< key, value >$ pairs within the network.

Each cell is assigned a unique identifier in the key-space (called *cell-ID*) to distribute the virtual world responsibilities among peers of the structured P2P network. We now describe how cell-IDs are associated to cells of the virtual world. The mechanism we describe is inspired by PHT [10], a distributed data structure that enables sophisticated queries over Distributed Hash Tables.

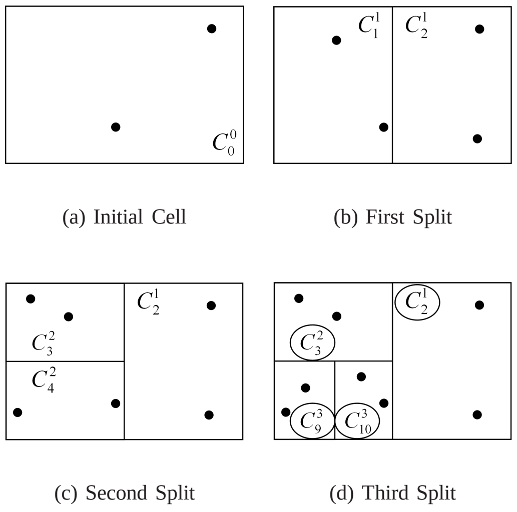


Fig. 1. Adaptive cell-based partition with $D_{max} = 3$

We call k_i^l the cell-ID associated to cell C_i^l; m is the maximum number of cell-IDs in the system. The cell-IDs are organized into a l-level binary tree with $0 \leq l \leq (log(m) - 1)$. Whenever a virtual cell C_i^l is split, two active cell-IDs $k_{(2i+1)}^{(l+1)}$ and $k_{(2i+2)}^{(l+1)}$ are derived as a function of k_i^l and associated to the cells $C_{(2i+1)}^{(l+1)}$ and $C_{(2i+2)}^{(l+1)}$. We associate $k_{(2i+1)}^{(l+1)}$ to west (vertical) or north (horizontal) originated cells and $k_{(2i+2)}^{(l+1)}$ to east (vertical) or south (horizontal) originated cells.

Given a cell-ID k_i^l, we derive $k_{(2i+1)}^{(l+1)}$ and $k_{(2i+2)}^{(l+1)}$ as follows. Let $\mathcal{H}$ be a generic hash function. For convention, $k_{(2i+1)}^{(l+1)} = \mathcal{H}(k_i^l)$ and $k_{(2i+2)}^{(l+1)} = \mathcal{H}(NOT(k_i^l))$. All the peers agree in a unique root for the tree, e.g., k_0^0, and on the hash function, e.g., $\mathcal{H}$=MD4 [3].

The hash function distributes cell-IDs uniformly in the key-space. In this way, any distribution of objects and cells in the SVW is mapped to a uniform distribution of cell-IDs in the key-space, achieving good load balancing among peers. The tree organization of the cell-IDs allows to always identify a set of cell-IDs to represent the cells organization in the SVW. We simply need to activate/deactivate the corresponding branches of the tree.

In order to maintain the evolution of the SVW, the cell-IDs are divided into *active* and *control* cell-IDs. We call an active cell-ID the identifier of a cell representing an active portion of the world, i.e., a leaf in the tree. We call a control cell-ID, the identifier used to retrieve information on the cells organization of the virtual world, i.e., an inner node of the tree. In Figure 2, we show the tree organization of the cell-IDs associated to the evolution of the SVW presented in Figure 1.

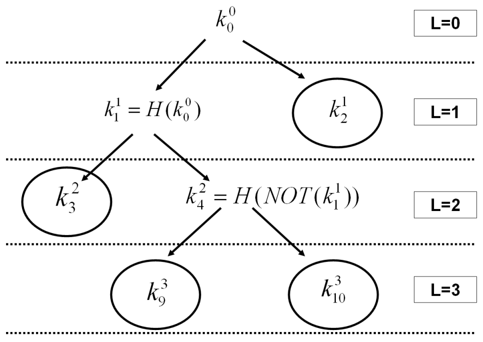


Fig. 2. Cell-IDs organization

C. Object Storage and Search

The KBR layer allows to identify for each cell-ID a set of peers in the P2P network who are responsible for this key. We call these peers *coordinators* as they are the peers who manage an object located within a cell of the SVW. For each cell within the SVW, R coordinators are selected, i.e., R copies of each object in a cell are stored on R different peers.

In a SVW, an avatar at time t is interested only in a portion of the entire virtual world. We call this area the *Area of Interest* (AoI). In order to ensure a consistent view of the world, users of the SVW subscribe to the R coordinators for the cells intersecting their AoI. In this way, they are informed of every modification of the world in their surroundings.

When an avatar creates an object in the virtual world, we say that the object is *published* in the P2P network. Object publication is done by mapping object coordinates (x, y) to the correct cell. An object $O_i(x, y)$ is published under the key k_i^l associated to C_i^l such that $(x, y) \in C_i^l$. The publication is done by informing the coordinators for cell C_i^l of the new created object.

A special object called *pointer* is published in cell C_i^l to inform other users that a merge/split operation was performed. The pointer contains an explicit notification of the split. In this way, other users are notified that k_i^l is a control cell-ID.

Whenever a participant wants to move toward a generic location $(x, y) \in \Omega$, it contacts the coordinators for cell C_i^l such that $(x, y) \in C_i^l$. If C_i^l is a control cell-ID, it is used as the entry point in the tree. The current active cell-ID containing (x, y) is retrieved by going through the tree. We call this operation the *discovery phase*. Note that the discovery phase is performed when an avatar joins the SVW and every time it moves to a cell which has been previously partitioned.

In each hop of the discovery phase, the cell coordinators must be found. Generally, this operation is logarithmic with the number of users N. In the worst case, a newcomer has to go through the entire tree, e.g., $log(m)$ hops are required when m is the maximum number of cell-IDs. Therefore, the number of hops for the discovery phase is $O(log(N) * log(m))$.

D. Implementation over Kad

Kad is the P2P network used for publish and search operations by eMule[2] users. It is derived from the original Kademlia protocol specifications [9]. In Kad, each peer is identified by a 128-bits Kad-ID and routing is based on prefix matching, i.e., the smallest XOR-distance. Kad divides keys in two types. A *source* key identifies the content of a file, and a *keyword* key classifies the content of a file.

We use the Kad API [3] to integrate our P2P infrastructure on top of Kad. The Kad *keyword* keys are used as control and active cell-IDs. Objects and pointers information are stored within the field of the metadata associated to a keyword. A unique identifier is used to distinguish between objects and pointers belonging to our system and content inserted into Kad by eMule clients.

The publication scheme in Kad differs from the original Kademlia protocol, since the XOR minimum distance is not really applied [12]. Keys are published on 10 different peers whose Kad-IDs agree at least in the first 8-bits with the key: this fraction of the key-space is called the *tolerance zone*. Since the Kad network is composed of millions of users, at any point in time there are about 10,000 nodes which fall in the tolerance zone of a given key [12]. This means that the consistency of our SVW could be impacted by the publication scheme deployed in Kad. For this reason, we simply modified the publication scheme to use the XOR minimum distance rule. Whenever a user subscribes to a cell C_i^l, it initially derives the set of peers in the tolerance zone of k_i^l. Then, it extracts the R coordinators for cell C_i^l by selecting the ones which identifiers are the closest to k_i^l according to the XOR distance.

The original Kademlia protocol recommends the usage of a re-publication scheme to maintain the consistency and persistency of a <*key,value*> pair during the publishing-searching life-cycle [9]. In order to ensure consistency, whenever a node w observes a new node u which is closer to some of w's <*key,value*> pairs, w replicates these pairs on u. A simple re-publication of the <*key,value*> pairs every hour is used to maintain persistency.

As for the publication scheme, the republication scheme in Kad differs from the original Kademlia. Keys are simply periodically republished by its owner, e.g., source keys every 5 hours and keyword keys every 24 hours. Moreover, a node responsible for a given key does not republish the key to a new node identified as closest to the key. Since Kad nodes may join and leave at any time, the set of R peers closest to a given cell-ID will change over time. This implementation choice adopted by Kad could negatively impact the consistency of our SVW.

We must assure that at any point in time, the R closest peers to a given cell-ID have the same information about the objects located in the cell. For this purpose, we modify the Kad republication scheme. We introduce a generalization of the Kademlia solution that we call the *delta publication*: whenever a peer p realizes that a peer Q is one of the R closest to k_i^l, p (re-)publishes on Q any objects $O_i(x, y) \in C_i^l$ it notices Q is missing.

III. Experimental Methodology

We now perform a realistic experimental evaluation of our P2P communication infrastructure for SVWs in two steps. Initially, we design a client prototype that implements the functionalities described in the previous Section, i.e., Kad routing, cell management, avatar movement across the virtual world, and object insertion and lookup. Then, we use object and avatar traces collected in SL to emulate the behavior of a P2P version of SL. We reduce the SL objects to simple (name,coordinates) pairs, i.e., we do not consider any additional object attribute such as textures, in order to store them within the fields of the metadata associated to a Kad keyword.

We use the libsecondlife libraries[3] to build a crawler application that monitors regions in SL. The crawler connects to a SL server and continuously asks the positions of avatars and objects within the region. In addition, it queries the region server about its load. A more complete description of the crawler can be found in [15].

Traces are collected in the *Money Tree Island* region for a period of 10 hours. In our traces, we observe about 500 different avatars, with at most 90 concurrent avatars and at least 20 avatars in this region. 90% of the avatars barely move and visit less than 13% of the region. The most adventurous avatar traverses 65% of the region extension. We found a constant number of about 586 objects in the region. While collecting the traces, we measured that these conditions force the SL region server to slow down the simulation in about 50% of the times.

We emulate the activity of the *Money Tree Island* region by replaying its traces using our P2P client. Each avatar is emulated on a cluster of machines and it is associated to a node within Kad. In this way, the objects are stored on Kad nodes and searched through the Internet. We also introduce a generic user acting as a sniffer that we refer to as the *monitor*. The monitor can see the entire region. It measures the region performance, without performing any operation.

One issue with this methodology is that we do not know the history of objects creation in the Money Tree Island prior to the monitoring period. Therefore, we create an *initialization phase* of duration T_i where avatars randomly insert new objects. Objects coordinates are extracted from the traces in order to be real. The initialization phase lasts 20 min. Once the initialization phase is completed, there is no object creation or deletion in the region. Therefore, all split operations are performed during the initialization phase and they do not impact our performance evaluation. This strategy is representative of the evolution of the object composition of most SL regions [15]. The maximum number of objects per cell (D_{max}) is set to 20. The number of coordinators per cell varies from 5 to 20 (default value is 20). We approximate the AoI of each avatar as a circle centered on the avatar coordinates, and we variate its radius AoI_r between 5 and 35 units (default value is 35). With these parameters, we distribute the 586 objects of the Money Tree Island region across 44 cells and a 9-levels tree.

[3]http://www.libsecondlife.org/

The numbers above are (1) derived from observations in multiple SL regions [15] and (2) chosen to make the experimental analysis tractable. It is not our intention to generalize to other SVWs, but to perform a realistic evaluation of a P2P Second Life.

IV. Performance Analysis

We now formally define the three performance metrics that we will evaluate. Then, we present the performance results. Please note that it was not possible to compare the performance of P2P SL to the real SL as we cannot perform similar measurements on SL. The goal of this work is to show the feasibility of a P2P SVW, and to give us insights on how to design a dedicated P2P infrastructure. In addition, SVWs requirements are different from the ones of on-line games. Therefore, comparison to previously published P2P on-line games is out of the scope of this work.

A. Parameters Definition

For simplicity, we consider a two dimensional region, i.e., a Cartesian space $\Omega = [0, X_{max}] \times [0, Y_{max}]$, where X_{max}, Y_{max} are the maximum extension of the region along the x,y dimension. However, extension to three dimensional virtual worlds is straightforward. An *object* is a piece of content identified by its name and coordinates. We denote it with $O_i(x, y)$. We call $\mathcal{O}(t)$ the set of objects created within the region before time t. To simplify the persistency evaluation, we assume that objects never disappear from the world. We call A a generic finite portion of the region. We call *state*, i.e., $S(t, A)$, the set of objects contained in an area A of the region at time t. We define $S(t, A)$ as follows:

$$S(t, A) = \{O_i(x, y) \in \mathcal{O}(t) \ s.t. \ (x, y) \in A\} \qquad (1)$$

We call $AoI_i(t)$ the AoI of an avatar at time t. We call $V_i(t, A)$ the set of objects seen by a user i at time t within an area A. Note that $V_i(t, AoI_i(t)) \subseteq S(t, AoI_i(t))$, i.e., an avatar may not see all the objects in its AoI due to inconsistency in the system.

Consistency: A SVW is consistent if at time t all the active avatars $\mathcal{N}(t)$ see the same set of objects (whether they exist or not). In order to define the consistency, we call $SAoI_i(t)$, or shared $AoI_i(t)$, the portion of $AoI_i(t)$ contained in at least another $AoI_j(t)$ at time t. We define $SAoI_i(t)$ as follows:

$$SAoI_i(t) = \bigcup_{j \neq i} \{AoI_i(t) \cap AoI_j(t)\} \qquad (2)$$

For a generic user i at time t we define the consistency of a SVW as follows:

$$K_i(t) = \frac{|\ (V_i(t, SAoI_i(t)) \cap \bigcup_{j \neq i} \{V_j(t, SAoI_i(t))\}\ |}{|\ \bigcup_{j \neq i} \{V_j(t, SAoI_i(t))\}\ |} \qquad (3)$$

In case that $SAoI_i(t) = \emptyset$, we consider that $K_i(t) = 1$. Note that the consistency is a user-dependent value.

Persistency: A SVW is persistent if no object gets lost during the evolution of the virtual world. Therefore, the persistency is defined by the following property:

$$S(t, \Omega) \subseteq S(t^{'}, \Omega) \quad t^{'} > t \tag{4}$$

Scalability: the consistency and persistency of a SVW must not be affected by the number of concurrent users and objects. Therefore, we have identified two approaches to establish the scalability of a SVW. First, prove that the P2P layer managing all communications is itself scalable. Second, analyze how increasing object density impacts the consistency and the persistency of the SVW. Given we use Kad, a well know P2P network that supports millions of users, we decided to focus the scalability analysis on the second issue.

B. Consistency

1) Region Consistency: Figure 3 shows the complementary cumulative distribution function (CCDF) of the consistency experienced by an avatar for different values of the AoI radius. For AoI radius up to 20 units, the region is perfectly consistent in 96% of the cases. For a more realistic value of 35 units, each avatar has a consistent view of the region 90% of the time. Intuitively, a larger AoI radius increases the discovery time and the likelihood of inconsistency.

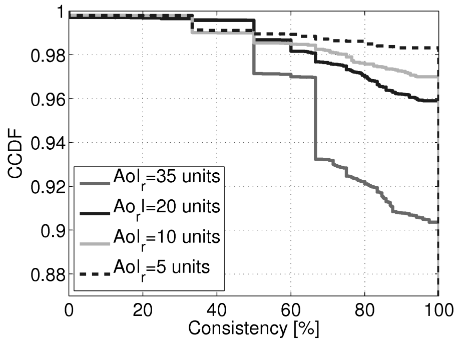


Fig. 3. CCDF of region consistency; $R = 20$; $AoI_r = [35; 20; 10; 5]$ $units$

We identify three origins of avatars inconsistency: (1) avatar movements among cells, (2) avatars joining the region, (3) P2P hazards, i.e., Internet latency, nodes churn and failures. Among the inconsistency values, avatar movements are cause of inconsistency in 45% of the cases, and they are responsible of consistency reduction from 65 to 99 percent (see Figure 3). In the same range of consistency values, accessing data in a real and large P2P network is a cause of inconsistency in 35% of the cases. Finally, the most dramatic inconsistency values (between 0% and 65%) are obtained when an avatar joins the region. These join operations cause 20% of the inconsistency values. In all cases, inconsistency is temporary and all avatars always succeed to reach 100% consistency after a while. The time it takes to come back to a consistent view of the region is discussed later in this Section.

Figure 3 shows three vertical steps respectively at 35, 50 and 65% of consistency. These steps indicate popular levels of inconsistency. When an avatar joins the region, it performs the *discovery phase* (see Section II-C). During this period, it only has a partial view of the region. Given that avatars tend to join a region always at the same locations [15], they miss more or less the same objects. A low level of consistency in the discovery phase is not really a problem as the effective join can be delayed until the view is consistent.

Inconsistency can either affect isolated avatars, or sets of avatars. We now measure the probability that a fraction of the avatars is inconsistent at the same time (see Figure 4). We notice that all avatars are perfectly consistent in about 55% of the cases. In 35% of the cases, a maximum of 10% of the avatars is simultaneously inconsistent. Finally, the number of inconsistent avatars is always lower than half of the avatars. These results demonstrate that inconsistency is never related to the P2P architecture, as it only affects a small subset of the region population.

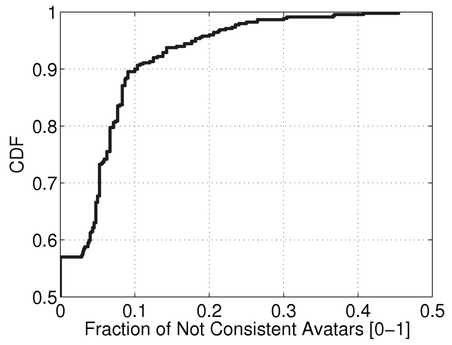


Fig. 4. CDF of the fraction of not consistent avatars ; $R = 20$; $AoI_r = 35$ $units$

2) Replica Consistency: We now analyze the impact of the number of coordinators R on the consistency. By default each user selects 20 coordinators per cell, i.e., each object is replicated 20 times. In order to simulate several values of R during a single emulation, we perform the emulation with $R = 20$ and then reconstruct the view of the region for each avatar using different subsets of the number of coordinators.

Figure 5 shows an evaluation of the impact of R on region consistency. We notice that we achieve comparable consistency levels for 10 to 20 replicas per object, and 5 replicas is clearly not enough to maintain the region consistency. In order to investigate deeper the impact of the number of replicas on consistency, we compute the probability that avatars access different coordinator sets for a given cell. We observe that avatars contact the same set of coordinators only 50% of the time. This is due to churn in Kad, obsolete information in some nodes routing tables and avatars joining a cell (i.e., selecting the set of coordinators) at different times. In addition, we observe that for $R = 5$, there is a non negligible probability that all avatars in a cell get the objects from a totally disjoint set of coordinators. This explains the reduction of consistency observed in Figure 5. These results are clearly impacted by the Kad users session characteristics and could be different with P2P nodes being maintained by SL users.

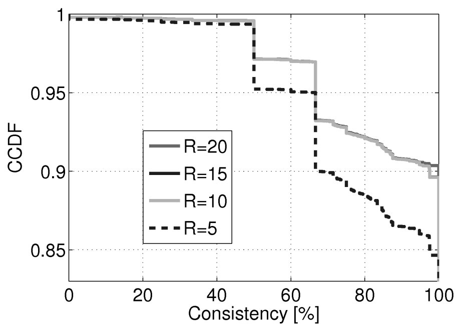


Fig. 5. CCDF of region consistency ; $R = [20; 15; 10; 5]$; $AoI_r = 35\ units$

3) Recovery Latency: We now investigate the time avatars spend in an inconsistent state. Figure 6 plots the cumulative distribution function (CDF) of the time it takes to an avatar to recover from an inconsistent view of the region. In the following, we refer to this time as the *recovery latency*. The recovery latency is smaller than 1 second around 35% of the times. 40% of the times, it takes to an avatar between 4 and 16 seconds to re-establish a consistent view of the region. About 20% of the times this latency can be longer of 16 seconds and reach couple of minutes. We recall that this latency only accounts for the discovery of objects in a region, and it does not refer to avatar interactions.

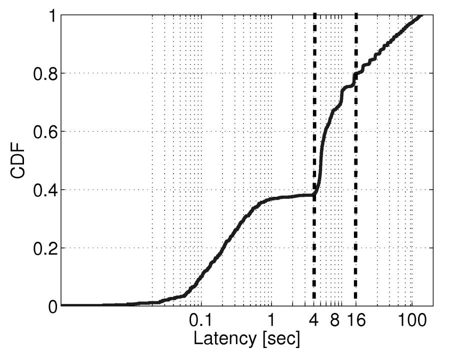


Fig. 6. CDF of the recovery latency ; $R = 20$; $AoI_r = 35\ units$

In Table I, we assign the recovery latency to each cause of inconsistency identified earlier. Latency values lower than one second correspond to normal network conditions we observe in Kad. Latency values between 4 and 16 seconds are experienced by avatars that move as they often have to perform a lookup operation in Kad. We observe that this lookup time in Kad is generally close to 4 seconds, as also observed by Steiner et al [13]. During the lookup, the objects contained in the new cell falling in the avatar AoI are not yet visible to the avatar. In addition, in case the new intersecting cell is a control cell, a discovery phase has to be initiated. Therefore, multiple lookups in Kad are performed, and the recovery latency can reach 16 seconds. Finally, values between 16 and 100 seconds are observed by avatars who join the region. Ideally, we should never observe delays larger than 40 seconds as the cells tree organization in the emulation has a maximum depth equal to 9. However, in some cases, avatars join the region and immediately start to move across cells, e.g., they travel the region to reach some friends. This behavior causes avatars to change cell even before they obtain a consistent view of the current cell, with a dramatic impact on the recovery latency.

Cause	Percentage	Recovery Latency (sec)
Moving	45	4-16
P2P Hazards	35	≤ 1
Joining	20	16-100

TABLE I
CAUSES OF INCONSISTENCY

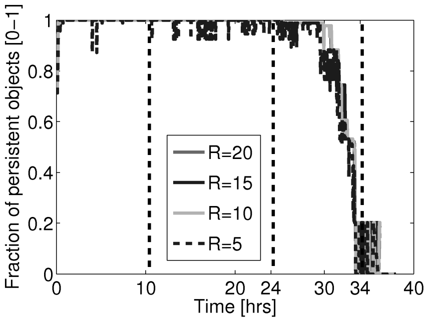
C. Persistency

Persistency characterizes the ability of our P2P architecture not to loose objects over time. Remember that after the initialization period where all objects are created, no more object will appear. A keyword in Kad is removed after 24 hours if not republished. Since we exploit Kad keywords to store SL objects, all objects should be discoverable by any avatar 24 hours after the last time they are re-published.

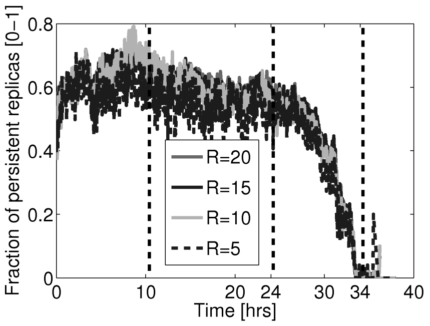
We measure region persistency through four monitors which systematically access all cells in the region and report statistics on the objects they contain. In order to evaluate the impact of a different number of coordinators R, each monitor contacts respectively 20,15,10 and 5 coordinators per cell.

In Figure 7, we plot the time evolution of respectively the fraction of persistent objects (Figure 7(a)) and the average fraction of persistent object replicas (Figure 7(b)). Note that an object is persistent when at least one of its replica is available in Kad. The first 20 minutes correspond to the initialization phase. During this period, the cell organization changes frequently as new objects are created. This explains why the persistency grows from 0.7 to 1 in Figure 7(a). Once the initialization phase is completed, SL is perfectly persistent during the entire emulation (i.e., 10 hours) for $R = [20; 15; 10]$. For $R = 5$, we notice two glitches in the curves of both Figure 7(a) and Figure 7(b) at $t = 5\ hrs$ and $t = 6\ hrs$. At these times, the monitors could not find some objects in Kad as all their replicas had disappeared from the P2P network. The cause of this phenomenon is the difficulty to constantly maintain a set of consistent coordinators under the presence of churn, failures, etc. Anyway, in both cases the persistency goes back to 1 at the next measure performed by the monitors. This is because, as soon as an avatar observes that some objects have disappeared, it immediately performs a (delta) publication and persistency is recovered.

The emulation ends after $10\ hrs$, which means that objects are not republished anymore for $t > 10\ hrs$. Figure 7(a) shows that for $t > 10\ hrs$ the glitches in the curve with $R = 5$ happen more frequently than in the first 10 hours. In addition, we observe a continuous decrease of the average fraction of persistent object replicas in Figure 7(b). This is due to the



(a) Time evolution of the fraction of persistent objects



(b) Time evolution of the average fraction of persistent object replicas

Fig. 7. Persistency analysis ; $R = [20; 15; 10; 5]$

absence of the delta publication, which goal is to actively maintain a target number of replicas in the network. However, until $t = 24\ hrs$ the region remains perfectly persistent for $R > 5$. In addition, even for $R = 5$ SL persistency goes back to 1 from time to time. This behavior is explained by the presence of churn in Kad: even if objects are not more constantly republished, old coordinators which temporarily left the Kad network eventually come back restoring the persistency of the region.

The delta publication algorithm seems to be too conservative when the number of object replicas in the P2P network is larger than $R = 5$. In order to study how our P2P SL would behave without the delta publication, we study persistency in the $[10, 34]$ hours period where the delta publication is not active anymore. For this reason, we now analyze the behavior of the Kad nodes selected at least once to be coordinators in the $[0, 10]$ hours period. We compute respectively their continuous on-line time and availability to serve SL objects in the time interval $[10, 34]$ hours, i.e., the likelihood to be selected as coordinators. For space limitations we do not plot any figure.

We observe that 80% of the coordinators have a continuous on-line time smaller than $3\ hrs$, and only 1% of the coordinators is on line during the entire time interval $[10, 34]$ hours. In one hand, this is due to the Kad user session characteristics [14]. In the other hand, it depends from the high level of churn in Kad [14] which causes a coordinator to often go out from the set of Kad nodes at minimum XOR distance from a cell-ID.

The strong presence of churn in Kad causes frequent changes in the sets of cell coordinators. 50% of the coordinators serve SL objects only for 10% of the time interval $[10, 34]$ hours. However, 15% of the coordinators serve objects for more than 60% of the time. These nodes guarantee persistency despite the presence of many unstable nodes.

Finally, we analyze the time interval $[24, 34]$ hours. Since objects in Kad are removed after 24 hours if not republished, in this time period we expect all objects to be deleted and the persistency of our P2P SL to go to zero. As expected, the average number of objects replicas decreases quickly (see Figure 7(b)). However, for $t \geq 34\ hrs$, 20% of the objects created are surprisingly still present in Kad. The reason is that some eMule clients increase the lifetime of Kad keywords, i.e., affecting our SL objects, to reduce the volume of publishing/republishing operations[4].

The persistency results we have presented are impacted by the behavior of peers in the Kad network. We expect to observe a different behavior of peers in a structured P2P network maintained among avatars of a SVW. However, these results underline two strong features of our P2P architecture. First, it can support a very high churn in the P2P network. Second, it is enough to have a low number of stable peers to maintain excellent persistency.

D. Scalability

We need to discuss two aspects of scalability. First, we must prove that our P2P SL scales with the number of users. This is easy as our system inherits its scalability from Kad. Therefore, we decided not to perform any emulation with large number of participants as the scalability of Kad and Kademlia has been shown already [12][14].

Second, we evaluate the impact of the number of objects (which in turn impacts the number of cells) on the scalability of our P2P architecture. To do so, we perform four emulations varying the number of objects in the region, namely $N = [29; 82; 112; 586]$. Object locations are randomly chosen from the objects population of the Money Tree Island. We chose several subsets of the Money Tree Island objects composition to analyze whether our P2P architecture becomes perfectly scalable for lower number of objects. The different configurations of cells and objects for the emulations are summarized in Table II.

We analyze first the impact of the number of objects on the consistency perceived by the avatars. Figure 8(a) plots the CCDF of the consistency for different values of N[5]. We observe that for $N = 29$, the consistency is nearly always equal to 100%. This is due to the fact that the limited depth of the cell tree organization reduces the impact of the join

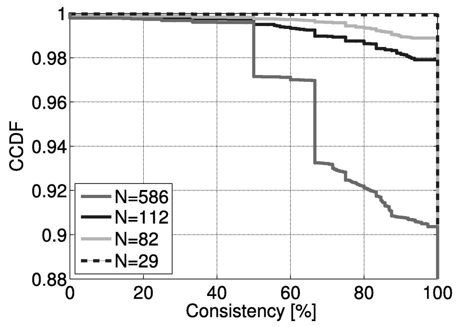
[4]http://www.emule-project.net/

[5]Note that the curve with $N = 586$ is the same as the curve with $AoI_r = 35\ units$ in Figure 3 and the curve with $R = 20$ in Figure 5

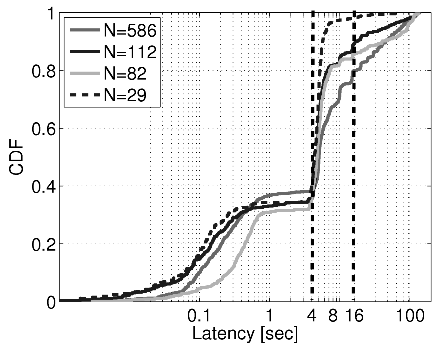
objects	active cell-IDs	max tree depth
586	44	9
112	15	5
82	6	4
29	3	2

TABLE II
CELLS CONFIGURATION

operation and consequently of the discovery phase. Moreover, the number of movements among cells is reduced as cell sizes are very big. Finally, we see that increasing the number of objects has a sublinear impact on the consistency; increasing by one order of magnitude the number of objects causes a reduction of consistency by less than 10%.



(a) CCDF of region consistency



(b) CDF of the recovery latency

Fig. 8. Scalability Analysis; $R = 20$; $AoI_r = 35\ units$; $N = [586; 112; 82; 29]$;

In Figure 8(b) we plot the CDF of the recovery latency for different values of N[6]. There is not clear impact of the cells organization on latency values smaller than 1 second. In fact, these values only depends on the Kad nodes involved in the emulations and on the network conditions. All curves meet at 4 seconds, which corresponds to the discovery latency of one cell as we have seen in Figure 6. For recovery latency values larger than 4 seconds, we observe a clear impact of the different complexity of the cell organization in the four emulations. For $N = 29$, 90% of the recovery latency values are smaller than $8\ sec$. Since the maximum depth of the tree is two, 8 seconds correspond to the time duration of a discovery phase of length two in the tree. Then, the depth of the tree increases with N. This in turn increases the probability to experience a larger recovery latency. In particular, we notice that when the recovery latency is larger than 4 seconds, increasing the number of objects of one order of magnitude causes a latency increase of about 30%. This result shows that the duration of lookup operations in Kad limits the scalability of our P2P SL.

[6]Note that the curve with $N = 586$ is the same curve of Figure 6

V. Related Work

We are not aware of research work on Peer-to-Peer (P2P) Social Virtual Worlds (SVWs). However, several authors have proposed P2P architectures for on-line games. MiMaze [5] was the first serverless on-line game, relying on IP multicast. More recent work includes SimMud [8] and Colyseus [2], both of which employ a Distributed Hash Table (DHT). Solipsis [7] adopts an unstructured approach where virtual neighbors collaborate to reciprocally monitor the state of the virtual world. A similar approach is proposed by S.-Y. Hu et al. [6]. They introduce the usage of a Voronoi network to manage collaboration among peers.

Albeit SVWs and on-line games are two applications of Networked Virtual Environments (NVEs), they exhibit several differences. In particular, they differ in the amount of user-generated content: while in on-line games the virtual world is mostly predefined and static, SVWs are predominantly user-generated. Thus, SVWs require significantly higher amounts of persistent networked storage and data transfers.

This paper differs from previous work on P2P NVEs in several ways. First, we focus on the research challenges involved in P2P NVEs with strong persistency needs, i.e., Social Virtual Worlds. Second, we propose an innovative approach as we implement our P2P architecture for SVWs on top of a widely deployed P2P network (Kad), and study the issues that arise when employing such an existing network. Finally, we perform a realistic evaluation employing traces of real Second Life user behavior.

The P2P communication infrastructure we deployed is inspired from CAN [11], a structured P2P network which provides hash table-like functionality. CAN introduces a relationship between a logical Cartesian space and the key-space. At any point in time, the Cartesian space can be dynamically split to achieve a uniform partitioning of the key-space among peers. The mechanism we use to index the virtual world is inspired from the Prefix Hash Tree (PHT) [10]. PHT is a distributed data structure that enhances DHTs to the management of more complex queries, such as *range queries*. PHT organizes keys in the DHT as a binary trie, and uses the DHT lookup operation to handle range queries.

We integrate our architecture on the top of Kad to perform a realistic emulation of Second Life on the Internet. In a study unrelated to NVEs, Steiner et al. [14] conducted a comprehensive analysis of Kad. They developed a crawler application to explore Kad and monitored a fraction of the Kad key-space for about six months. They identify two classes of peers: long-lived peers that participate in Kad for weeks and short-lived peers that remain in Kad no more than few days. Moreover, they observe that the distribution of peers arrivals and departures is well described by a Negative Binomial distribution.

VI. Lesson Learned and Future Work

This paper presents an experimental study of a Peer-to-Peer (P2P) architecture for Social Virtual Worlds (SVWs) such as Second Life (SL). Our communication infrastructure dynamically partitions the virtual world into cells, and assigns responsibility for those cells to peers named *coordinators*. The deployment of our P2P SVW is performed over Kad, a widely deployed structured P2P network based on the Kademlia routing protocol. We collect traces of user activity (i.e., movement, churn and object creation) in Second Life in order to perform a realistic emulation of a P2P SVW on the Internet.

Our evaluation shows that the architecture we have designed achieves acceptable levels of consistency, persistency and scalability. Inconsistency is temporary and limited to avatars entering new cells or joining the P2P SVW. Persistency is excellent for the whole duration of the emulation. Finally, the architecture can scale up to the number of objects contained in a typical SL region.

The real deployment and evaluation we performed over Kad underlines two interesting features of our architecture. First, region persistency can be maintained even with a very low number of stable peers; this result suggests that automated avatars in SL [15], which are nearly always connected, could be exploited to improve SL performance. Second, it exists a trade-off between the rate of object re-publication and the number of object replicas. This suggests the usage of an object re-publication rate which dynamically adapts to the number of currently available object replicas.

Our P2P SL suffers from the latency avatars experience to recover from an inconsistent view of the world. The causes of these inconsistencies are mainly: (1) avatars joining the P2P architecture, (2) avatar movements across cells. Our results show that, for large number of objects, the recovery delay can become so long that avatars would change cell before they even have a consistent view of the current cell.

The recovery latency due to newcomers joining our P2P SL can be easily addressed. The tree based organization of the cells could be cached at the client and re-used across different sessions. During the initial connection to SL or in case the cell organization has changed between two sessions, the effective join of an avatar could be easily delayed.

Reducing the recovery latency due to avatar movements is a more complex problem to solve. Pre-locating the coordinators for all adjacent cells can help reduce boundary-crossing latencies. However, this solution would come at the expense of increased traffic and load on the coordinators. We believe that this recovery latency can be nearly removed by mapping cells that are close in a SL region to cell-IDs which are close themselves in the XOR address space. This would allow a faster navigation in the virtual world with no additional cost on the coordinators. Finally, a caching or prefetching of objects can give significant benefits given the predictability of avatar movement patterns [15].

In the future, we plan to design a key-based-routing algorithm which implements the proposed technique to make the object-search cost independent from the size of the P2P network. We also plan to study how to integrate the proposed architecture with a Delaunay-based overlay topology in order to manage communication between avatars.

Acknowledgements

The authors would like to thank Moritz Steiner for the precious help to integrate our P2P infrastructure on top of Kad. We also would like to thank Fabio Picconi for his insightful comments and suggestions.

References

[1] N. E. Baughman, M. Liberatore, and B. N. Levine. Cheat-Proof Playout for Centralized and Peer-to-Peer Gaming. *IEEE/ACM Transactions on Networking*, pages 1–13, February 2007.
[2] A. Bharambe, J. Pang, and S. Seshan. Colyseus: A Distributed Architecture For Online Multiplayer Games. In *NSDI'06*, Berkeley, CA, USA, May 2006.
[3] R. Brunner and E. Biersack. A Performance Evaluation of the Kad Protocol. Technical report, Eurecom, 2006.
[4] F. Dabek, B. Zhao, P. Druschel, and I. Stoica. *Towards A Common API For Structured Peer-to-Peer Overlays*, pages 33–44. Number Volume 2735/2003 in Lecture Notes in Computer Science. 2003.
[5] L. Gautier and C. Diot. Design and Evaluation of MiMaze, a Multi-Player Game on the Internet. In *International Conference on Multimedia Computing and Systems*, pages 233–236, 1998.
[6] S.-Y. Hu, J.-F. Chen, and T.-H. Chen. VON: A Scalable Peer-to-Peer Network for Virtual Environments. *Network, IEEE*, 20(4):22–31, 2006.
[7] J. Keller and G. Simon. SOLIPSIS: A Massively Multi-Participant Virtual World. In *PDPTA*, pages 262–268, 2003.
[8] B. Knutsson et al. Peer-to-Peer Support for Massively Multiplayer Games. In *Infocom*, Hong Kong, China, March 2004.
[9] P. Maymounkov and D. Mazieres. Kademlia: A Peer-to-Peer Information System Based on the XOR Metric. In *IPTPS*, Cambridge, MA, USA, March 2002.
[10] S. Ramabhadran, S. Ratnasamy, J. M. Hellerstein, and S. Shenker. Brief Announcement: Prefix Hash Tree. In *PODC*, page 368, 2004.
[11] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Schenker. A Scalable Content-Addressable Network. In *SIGCOMM*, San Diego, USA, October 2001.
[12] M. Steiner, E. W. Biersack, and T. En Najjary. Actively Monitoring Peers in KAD. In *IPTPS*, Bellevue, USA, February 2007.
[13] M. Steiner, D. Carra, and E. W. Biersack. Faster Content Access in KAD. In *P2P*, Aachen, Germany, September 2008.
[14] M. Steiner, T. En Najjary, and E. W. Biersack. A Global View of KAD. In *IMC*, San Diego, USA, October 2007.
[15] M. Varvello, F. Picconi, C. Diot, and E. Biersack. Is There Life in Second Life? In *Conext*, Madrid, Spain, Dec. 2008.