Exploring Second Life

Matteo Varvello,Stefano Ferrari, Ernst Biersack, Christophe Diot

***Abstract*—Social virtual worlds such as Second Life are digital representations of the real world where human-controlled *avatars* evolve and interact through social activities. Understanding the characteristics of virtual worlds can be extremely valuable in order to optimize their design. In this work we perform an extensive analysis of Second Life (SL). We exploit standard avatar capabilities to monitor the virtual world, and we emulate avatar behaviors in order to evaluate user experience. We make several surprising observations. We find that 30% of the regions are never visited during the six day monitoring period, whereas less than 1% of the regions have large peak populations. Moreover, the vast majority of regions are static, i.e., objects are seldom created or destroyed. Interestingly, we show that avatars interact similarly to humans in real life, gathering in small groups of 2 to 10 avatars. We also show that user experience is poor. Most of the time avatars have an incorrect view of their neighbor avatars and inconsistency can last several seconds, impacting interactivity among avatars.**

***Index Terms*—Second Life, virtual worlds, measurement, peer-to-peer.**

I. Introduction

A Networked Virtual Environment (NVE) is a computer-generated world where users interact through the Internet [1]. NVEs were introduced in the 80s for military simulators, and then applied to on-line games. World of Warcraft [2] is the most popular on-line game with nearly 14 Millions subscribers. Social virtual worlds are a new type of NVE where people can meet, play, trade and even contribute to the development of the NVE. Second Life [3] (SL), launched in 2003 by Linden Lab, has become the most popular social virtual world, reaching 16 million registered users in June 2009.

SL consists of a virtual land, divided into fixed-size *regions*, where users interact via their digital representation called *avatars*. Avatars participate in the development of the virtual environment by creating *objects* such as cars, walls, trees, and buildings. SL has created a full-blown economy, attracting companies that have invested millions of dollars in order to build their own virtual products and advertisements.

Despite the widespread interest that SL has generated from both users and companies, very little is known about its characteristics. In fact, SL provides only limited statistics such as the total number of regions, registered accounts, and online avatars. Some authors have analyzed in details the SL client [4] and the network traffic it generates [5], [6], [7], while others have characterized avatar mobility [8], [9]. To the best of our knowledge, the work in [10] is the only large scale data collection performed in SL. Nevertheless, we are not aware of research work that has investigated yet the user Quality of Experience (QoE) in SL.

This work extends the results published in [10]. Our motivation is twofold: (1) understand avatar behavior and object characteristics in order to allow further improvements of the SL architecture, (2) understand user QoE in SL.

To conduct this study, we design and deploy a *crawler* and a *player*. Our crawler is an application that connects to SL servers and exploits standard avatar capabilities to collect information about the virtual world. Our player emulates the behavior of an avatar in SL, while collecting traces about avatars and objects encountered.

We use the crawler to explore SL at different time and space resolutions. (i) We monitor the object composition of around $13,000$ public regions during one month. (ii) We collect server statistics of the public regions over one week. (iii) We track avatar distribution in the entire virtual world (i.e., public and private regions) over one week.

We then use the player to explore user QoE. We execute several instances of the player over multiple Planetlab [11] machines in order to populate a SL region only with our controlled avatars. In this way, we have complete control on the local view of each avatar that interacts in the region as well as its behavior. We then study user QoE comparing the local view of each avatar with the "ground truth" defined by avatar behaviors. We replay avatar mobility traces [10] in order to make our measurements realistic. We conduct these measurements in three SL regions characterized by a different object composition.

We find that the number of objects per region is roughly constant over one month period. The active population at any point of time is between $30,000$ and $50,000$ avatars, i.e., about 0.3% of the registered avatars. Quite surprisingly, about 30% of the regions are never visited during six days. Avatars tend to organize in small groups of 2 to 10 avatars, suggesting that the human "attention budget" theory [12], [13] may also hold in social virtual worlds. Large groups of avatars are very rare, and are driven by the presence of events such as concerts and shows.

From a user perspective, we observe that the object composition of a region can have a negative impact on the QoE: in a region crowded with virtual objects, avatars have an inconsistent view of their neighbor avatars half of the time, i.e., either they do not see them or they see them at a wrong location. Moreover, in 50% of the cases this inconsistency lasts more than one second. Since acceptable values of interactivity

Manuscript received September 22, 2009; revised April 5, 2010. Approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor X. First published X X, X; current version published July X, 2010.

Matteo Varvello is with Alcatel-Lucent (matteo.varvello@alcatel-lucent.com). Stefano Ferrari is with Cisco System (fstefano@cisco.com). Ernst Biersack is with Eurecom (ernst.biersack@eurecom.fr). Christophe Diot is with Technicolor (christophe.diot@technicolor.com).

in on-line games varies between 300 ms and 1 sec [14], these results indicate poor user QoE under the current SL architecture.

Based on our observations, we identify several mechanisms to improve SL's design. Given the highly static object distribution and predictable nature of avatar behavior, caching and prefetching techniques could be particularly useful. In addition, we believe that Peer-to-Peer technologies could be an interesting addition to the current SL design. In the end of the paper, we discuss some results obtained via a modified SL client that allows direct communications between its end-users.

II. Related Work

Given the lack of information released by Linden Labs, recent work has focused on studying some of its aspects. Fernandes et al. [5] perform the first study related to SL. They collect the traffic exchanged between client and server in a typical SL session in order to measure bandwidth consumption, packet size and packet inter-arrival times. They conclude that SL requires much more network resources than existing applications for virtual worlds such as on-line games. Moreover, they show that the down-link traffic is strongly impacted by avatar actions: an avatar that simply stands in SL consumes about 20 Kbps in the down-link, whereas as soon as the avatar moves the down-link traffic grows up to 110 Kbps.

Kinicky et al. [6] extend the SL traffic analysis proposed in [5] by focusing on regions with different objects compositions. They reproduce some of the results obtained in [5] and they show that regions with high avatar and object density require a bandwidth 10 times larger than empty regions. In an extension [7], the same authors develop traffic models for SL.

Kumar et al. [4] analyze the CPU performance of a high-end desktop machine running the SL client. They find that sorting translucent objects and decompressing textures stored as JPEG are the most CPU expensive operations. Similarly to [5] they analyze the network traffic exchanged between client and server. Their results confirm the high bandwidth requirements of SL, and also underline the benefits of objects caching to reduce network traffic. Finally, they analyze server performance. They show that the management of a region with only $5,000$ rigid-body objects requires about 72% of the total server computational power. As SL-like virtual worlds are expected to become more complex and realistic, e.g., supporting object fracturing and deformation, several CPU cores will be required.

La and Michiardi [8] use a different approach and retrieve information directly from SL servers. They deploy a crawler application that monitors avatar movements for short periods of time. The analysis of their traces reveals that avatars tend to interact similarly to their human counterpart, e.g., the distribution of avatar contact-times is similar to that observed in real-world experiments.

Liang et al. [9] extend the analysis of avatar mobility proposed in [8]. They use a crawler application to collect mobility traces of $84,208$ avatars spanning 22 SL regions over two months. Based on their observations, the authors suggest an hybrid avatar mobility model that incorporates both random way-point mobility model (for regions that contain few objects) and pathway mobility model (for regions that contain many objects).

Although the methodology used in [8], [9] is similar to ours, these works focus only on studying avatar mobility. Conversely, our work is a comprehensive study of SL.

III. Second Life

The innovative features of SL are *world-building* and *virtual economy*. World-building is the possibility to participate in the deployment and modification of the virtual environment through the creation of user-generated objects. A virtual economy is the possibility to buy and sell objects, services and lands. These feature make SL very different to classic NVEs such as World of Warcraft [2]. The reader can find a detailed comparison of SL with other NVES in [5]. In the following, we describe in details the SL features that are of interest for the understanding of the paper.

A. Virtual World

The virtual world of SL is composed of *regions*, which are independent lands of size 256x256 meters. Each region has a maximum of four adjacent regions and can be either *public* or *private*. The public regions are owned by Linden Lab, while the private regions are purchased by individuals or companies. Owners of private regions have total control on their virtual land. They can, for instance, limit access to a selected set of avatars. Both types of regions run on Linden Lab servers, which are called "Simulators".

The appearance of a region is defined by the objects it contains. Each region has a specific policy on object creation and destruction. For instance, "Sandbox" regions are used by avatars to test new objects, which are automatically destroyed shortly after their creation.

SL provides a *map* of the virtual world, i.e., a compact visual representation of both public and private regions. The map shows the number of avatars connected to each region by displaying points located at the avatar coordinates. Avatar identities are not visible on the map.

B. Client/Server Architecture

The SL design is based on a Client/Server architecture: each region is managed by a dedicated server, and users run "thin" clients which simply perform the three-dimensional rendering of the virtual world and eventually cache the virtual objects located in recently visited regions [4]. The SL Client/Server protocol is not public, however reverse engineering efforts are underway, e.g., libsecondlife [15]. The project released a set of $C\#$ libraries that allow third party applications to interact with SL servers.

Since our crawler interacts with SL servers, we now shortly describe the server-side of SL. A more complete description of the SL architecture can be found in [5] and [7].

The *Login Server* is the entry point in SL, and handles username and password verifications. The Login Server is also

responsible of granting or denying access to the regions (e.g., access may be denied during servers failures or maintenance operations). It maintains the following statistics: number of connected users and number of logins in the last 24 hours.

Simulators are the servers responsible for SL regions. Each simulator maintains the state of a region and performs the *visibility computation* [4], i.e., identify for each avatar located in the region the information about objects, land features and avatars that need to be transmitted to the clients. It also manages chat among avatars located within the region. A Simulator handles a maximum of 100 avatars [16]. However, larger population are possible by mirroring the region server [4]. For clarity, we refer to simulators as *servers*.

We do not know the total number of servers in SL, nor whether SL employs load balancing techniques among servers. Moreover, SL does not mention protective measures against Denial of Services attacks and crawling operations.

C. Avatar Capabilities

A user creates an avatar by registering at the SL website [2]. This registration requires filling out an on-line form with personal information and a valid e-mail address. We now give a short description of the avatar capabilities that are used by our crawler to monitor SL.

A user entering SL must perform a login procedure. After authentication, its avatar joins the virtual world. The region where the avatar appears is either specified in the login request or derived from the avatar coordinates at its last connection. An avatar can *walk*,*run* and *fly* within a region, and also directly move to adjacent regions provided they are public. It is also possible to perform a *teleport* operation to rapidly cover large distances. The target destination of the teleport can be within either the same region, or any other region selected from the map.

Avatars have a limited visibility area called *Area of Interest* (AoI). This area corresponds roughly to a sphere with a radius of 35 meters. Avatars teleporting to a region are informed by the server about the locations and identifiers of all objects in the region. In the following, we refer to this event as "initialization phase". Finally, an avatar can request several region statistics to a server. A complete description of these statistics can be found at *wiki.secondlife.com*.

Automated avatars called *bots* are frequently used by region owners to show some activity in their regions or simply to welcome visitors. In order to prevent the usage of bots, SL disconnects avatars that have not moved during the last 15 minutes. Not surprisingly, simple scripts allow bots to perform repetitive actions, such as head movements, which is enough to circumvent SL's bot detection mechanism. Thus, we can assume that avatars that barely move and remain connected for a long time are most likely bots. We recognize that this strategy may incorrectly identify some human-controlled avatars as bots, but it still intuitively captures the presence of bots in SL.

D. Message Types

Second Life currently uses 473 different message types in the communication between servers and clients [4]. However, three main packet types can be identified: *control*, *region* and *avatar* packets.

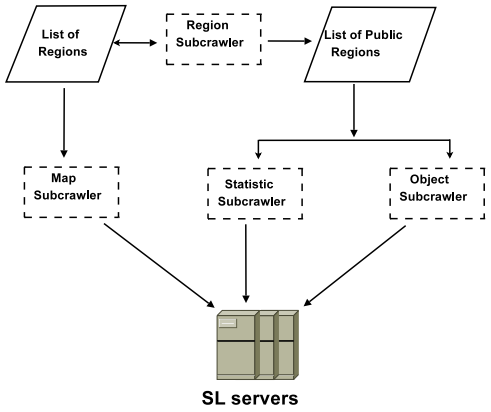


Fig. 1. Architecture of the crawler.

- The *control packets* are used to enforce security in SL and to check the state of the client connections.
- The *region packets* carry information about a region appearance and its object composition. The region packets received by a user contain the description of the portion of the region crossing its avatar AoI.
- The *avatar packets* carry information about the state, i.e., position and body appearance, of an avatar, and about the chat messages exchanged by avatars. The avatar packets received by a user refer to the state of the avatars located within its avatar AoI.

IV. Methodology

We use two different methodologies to study SL. First, we crawl all the information publicly available in SL. Second, we "play" SL and monitor the performance experienced by several controlled players. We first describe the *crawler* and the *player*, and then discuss the limitations and problems we encountered in our study.

A. Crawler

The main idea behind the crawler is to exploit standard avatar capabilities to obtain information about the virtual world. Our crawler is composed of multiple *subcrawlers*, each specialized in a different monitoring task (Figure 1). The reasons for this are twofold. First, different types of information can be collected using different crawling techniques. Second, splitting the crawling into different tasks allows us to control the temporal resolution of each type of information we collect. For instance, tracing the avatar distribution among SL regions should be done frequently (e.g., every 15 minutes), whereas determining the total number of objects in the system can be done much less often (e.g., once per day).

Each subcrawler is a modified SL client implemented using the libsecondlife [15] libraries. A subcrawler must be associated to an avatar registered on the SL website in order to be able to enter the virtual world. We use multiple instances of each subcrawler (associated to different avatar identities) in order to parallelize the crawling.

For each subcrawler, we describe its role, crawling technique and relationship with the other subcrawlers.

- The *Region subcrawler* monitors SL to maintain an up-to-date list of its regions. The region discovery is performed via a random walk among adjacent regions (we use a list of regions obtained at *http://stats.slbuzz.com/* to bootstrap). The Region subcrawler teleports to each region in the list to retrieve the set of adjacent regions. As new regions are discovered they are added to the list. In addition, region accessibility is verified to determine whether a region is public or private.
- The *Object subcrawler* tracks the evolution of objects in all public regions. It teleports to a public region and accomplishes the initialization phase, during which it is informed by the server of the coordinates and identifiers of all objects on the region. Then, it dumps this information and teleports to a new region.
- The *Statistics subcrawler* collects the statistics maintained by the servers of the public regions. It teleports to a public region, queries its server, dumps the results, and then moves to another public region. We collect the following server statistics: number of connected avatars, time dilation, which expresses the load on the server, and total number of packets going in and out from the server.
- The *Map subcrawler* monitors the location of avatars as shown on the SL map. For each public and private region, the subcrawler locates it on the map, and collects the coordinates of all the avatars currently connected to it. This task simply requires logging in to SL. Unfortunately, the Map subcrawler cannot identify the avatar identities as they are not shown on the map.

Note that the Statistics and Map subcrawlers collect partially redundant information. We exploit this redundancy to validate the correctness of our data.

B. Player

We reproduce avatar behaviors via controlled SL clients in order to evaluate user QoE. To do so, we use libsecondlife [15] to implement a *player* that emulates and automates avatar behavior while collecting traces related to user QoE.

The player performs the login of an avatar to a target region and moves the avatar in the region according to an input mobility pattern. Figure 2 shows the architecture of the player, which consists of the following components.

- The *libsecondlife* module handles the communication between an avatar and a server. It contains the API that allows an avatar to perform actions such as login or movements.
- The *movement engine* manages the movements of an avatar according to a given mobility pattern. It uses the libsecondlife module to inform a server of the avatar behavior, e.g., movements in the region.
- The *AoI table* is a data structure that contains positions and names of all avatars within an avatar AoI. The AoI table is updated according to the information received from the server. A snapshot of the AoI table is copied to the disk every 200 ms or when a modification of its content occurs.
- The *traffic analyzer* records all the incoming and outgoing packets exchanged between client and server. Subsequently, it parses each packet to extract information about its content, e.g., avatar, region or control traffic (Section III-D), and volume.

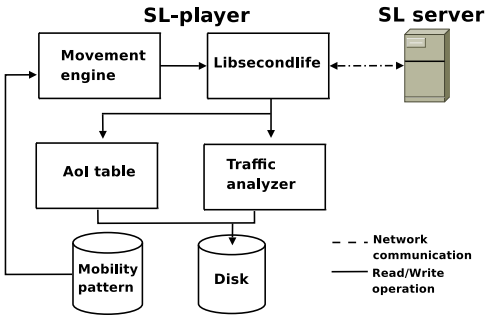


Fig. 2. Architecture of the player.

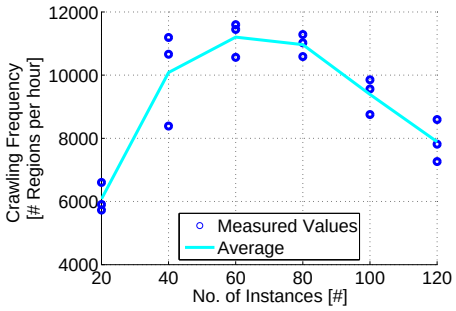


Fig. 3. Crawling Performance [Statistic subcrawler ; 18 hrs experience].

C. Problems and Limitations

To collect data in a scalable and accurate way, we had to solve several problems. We now discuss these problems as well as the limitations of our crawling strategies.

1) Crawling Performance: We refer to *crawling performance* as the number of regions a subcrawler monitors in a given time. As mentioned in Section IV-A, we run multiple instances of each subcrawler in parallel in order to increase the crawling performance. However, we observed that increasing the number of parallel instances does not necessarily improve performance. Figure 3 shows that the performance of the Statistics subcrawler degrades beyond 60 concurrent instances. We suspect that SL employs a rate-limiting policy against IP addresses that generate a large amount of traffic.

2) Experimental Challenges: Officially, SL does not mention any banning policies against avatars with unusual behaviors. However, many of the avatars associated to our subcrawlers were banned. The banning procedure consists of an exclusion due to "account verification". If a banned avatar attempts to login several times, its IP address is blacklisted. To avoid getting blacklisted, each of our subcrawler detects when it has been banned and automatically replaces the associated avatar identity with a new one.

Finally, we also observed a high degree of instability in SL. During a period of one month, the service was down multiple

times due to maintenance, server updates, or crashes [17]. While our short traces were mostly unaffected, outages had an impact on our long-term ones (see gap in Figure 6(b)).

3) Realism of the player: Libsecondlife implements a simplified Client/Server communication protocol compared to the official SL protocol [15]. Analyzing the incoming traffic at the player, we notice that many packets are ignored since they are unknown to the libsecondlife libraries. For this reason, the traffic volume exchanged between our player and the servers is generally smaller than what we would observe with the official client [5], [6]. However, a fine comparison of the libsecondlife and official SL client showed that the libsecondlife client can correctly handle all packets that refer to avatars and objects. Thus, we believe that the differences between the libsecondlife and the official SL client do not impact our measurements. Moreover, building a player with the official SL client would have the following limitations: (i) need of real users to control the avatars (ii), limited access to the data retrieved by the clients.

Finally, we measure user QoE "re-playing" real (monitored) avatar behaviors using bots. Intuitively, avatar behavior on a region changes according to factors such as the performance perceived by its user, user interest in the region and objects encountered. Our methodology cannot capture these factors. We also use bots that do not have any customization (e.g., fancy clothes) and that perform only simple movements. However, creating SL sessions that involve real players is a hard task, and does not allow a fair comparison of different regions.

V. A Global View of Second Life

We focus on the data collected by our crawler and on some data provided by SL through their website and Login Server in order to analyze SL-wide characteristics. When possible, we compare our results with these two sources to check their consistency. Finally, we use visual inspection to confirm some of our observations and interpretations.

A. Data Collection

Each subcrawler collects data at different time resolutions. In addition, some subcrawlers can traverse a set of regions much faster than others, according to the technique they use to collect information. We call *crawling frequency*, the frequency at which a subcrawler completely monitors a set of target regions. Finally, some subcrawlers require more resources than others (e.g., IP addresses, avatar identities), and are therefore executed for shorter periods of time.

We monitored the evolution of all regions and objects in SL during 28 days with a crawling frequency of 24 hours. We used three instances of the Region subcrawler and five instances of the Object subcrawler. Traces were collected between March 29, 2008 and April 25, 2008, except for April 4 and 5 when the SL service was down [17].

We ran 60 concurrent instances of the Statistic subcrawler, as this yields the highest crawling performance (Figure 3). With this configuration, the Statistic subcrawler can crawl about 11,000 regions in one hour. On March 29, 2008, the Region subcrawler identified 12,765 public regions, so we set the crawling frequency of the Statistic subcrawler to 90 minutes to be able to monitor all public regions with a safe time margin. Traces were collected for 6 days between March 29, 2008, and April 4, 2008.

We monitored the SL map with 40 instances of the Map subcrawler and a crawling frequency of 15 minutes. Traces were collected between April 18, 2008 and April 21, 2008. The traces refer to the total 17,526 regions identified by the Region subcrawler on April 18, 2008.

Table I summarizes each subcrawler configuration, trace length, and crawling frequency.

TABLE I
Second Life Crawling Summary.

Subcrawler	Instances	IP@s	Regions	Frequency	Days
Region	3	1	-	1/24 hrs	28
Object	5	1	-	1/24 hrs	28
Statistics	60	1	12,765	1/90 min	6
Map	40	1	17,526	1/15 min	3

B. Regions

Table II summarizes the total number of regions discovered by the Region subcrawler, as well as the official number reported by the SL website.

We observe that the Region subcrawler discovered a larger number of regions compared to official figures. These additional regions are not reachable and were discovered as adjacent of active ones. Therefore, they are probably a fraction of the virtual world reserved for future customers, and thus do not count in the official statistics.

TABLE II
Number of regions in SL (RS=Region subcrawler, SLW=Second Life website).

	March 29	April 18	April 25
Public regions (RS)	12,765	13,220	13,261
Total regions (RS)	17,280	17,526	17,573
Total regions (SLW)	13,693	N/A	14,150

The 6-day trace collected by the Statistics subcrawler shows that many regions experienced periods of unavailability. There are two possible causes for this: (i) the region server was down, or (ii) the maximum number of avatars per server has been reached.

We compute the region *availability* as the probability that a server accepts a connection from the Statistics subcrawler, i.e., the number of times it accepts a connection divided by the total number of connection attempts during 6 days. Figure 4 shows the Cumulative Distribution Function (CDF) of region availability. We observe that 90% of the regions have an availability of 0.9 or more, but only 1% show a high availability of 0.99 or higher. This is probably due to short maintenance operations or failures. The bottom 1% of the regions have an availability of 0.7 or lower. Our traces show that these highly unstable regions are not among the

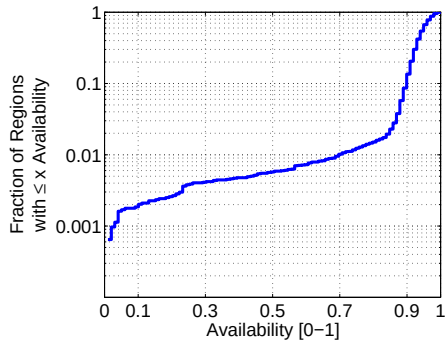


Fig. 4. CDF of Region availability [Statistics subcrawler].

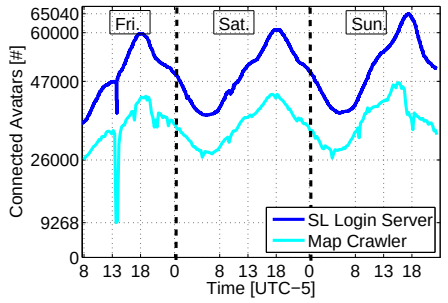


Fig. 5. Active population over time [Map subcrawler; SL Login Server; Coordinated Universal Time - 5].

most popular ones (Section V-F). This means that it is very unlikely that these region servers refuse the connection attempt from our crawler due to the 100 concurrent avatars limit. This suggests that the server unavailability more likely due to server failures.

C. Users

Figure 5 shows the evolution over time of the number of online users, as measured by the Map subcrawler and reported by the Login Server (this data is obtained by monitoring the Login Server [18]). Both curves exhibit the same daily cycle. However, the Login Server reports 10,000-20,000 more users than the Map subcrawler. This means that the data reported by the Login Server is not consistent with the data contained in the SL map. Moreover, during a major SL outage on Friday at 14:00, the Login Server reported a drop of 10,000 avatars, while our Map subcrawler observed a decrease by 20,000. Based on this observation, we conjecture that the values provided by the Login Server may be averaged over long time periods.

D. Server Traffic

The Statistics subcrawler collects information about the rate of outgoing server packets as reported by the SL servers. Although not shown for space reasons, the curve of the aggregate traffic generated by all servers shows a daily cycle ranging from 1.7 to 3.2 million packets per second. Moreover, the traffic's daily cycle closely follows that of Figure 5, which is to be expected for a Client/Server architecture. The correlation coefficient between the number of avatars in a region and the traffic volume is 0.8.

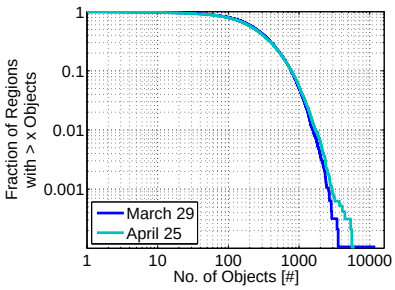
Knowing that the mean packet size in SL is 500 bytes [5], the aggregate traffic generated by all servers at peak time can be estimated to be around 13 $Gbps$. The peak number of users is $47,000$ measured on Sunday at 18:00, which yields an average bandwidth consumption of 280 $kbps$ per client. This confirms the results reported in [5] and shows the high bandwidth consumption of the SL service.

E. Object Distribution and Dynamics

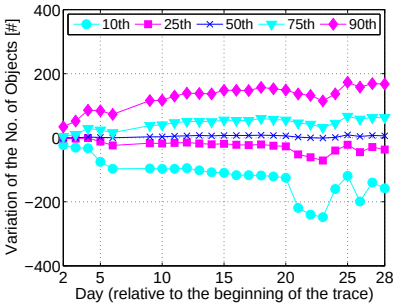
We analyze the 28-day trace collected by the Object subcrawler with a crawling frequency of 24 hours in order to understand SL object characteristics.

1) Object Distribution: We identified about 7 million unique user-generated objects across all public regions. Figure 6(a) shows the CCDF of the number of objects. Since the distribution did not significantly change over 4 weeks, we only plot the data for the first day (March 29, 2008) and for the last day (April 25, 2008). 30% of the regions are almost empty, containing less than 100 objects. Around 65% of the regions contain a relatively low object count, between 100 to $1,000$, while only 5% of the regions have $1,000$ objects or more. The richest region contains nearly $13,000$ objects. We would like to recall that the Object subcrawler cannot collect information about object sizes as this would take too much time and make the crawl operation very slow. Therefore, it is possible that some regions with a low object count actually have a more complex environment (e.g., bigger objects) compared to other regions with a larger object count [19].

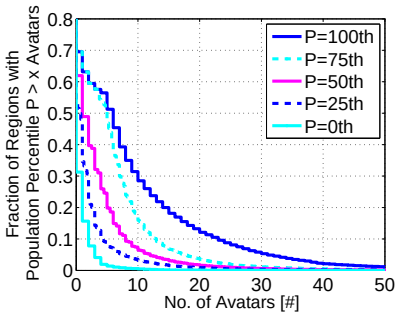
2) Object Dynamics: We now analyze the evolution of the number of objects over time. For each region, we compute the difference between the number of objects it contains at day i, and its initial object count observed at day 1, i.e., the first day of the monitoring (March 29, 2008). Figure 6(b) shows some significant percentiles of the distribution of these differences measured for all regions (the gap between days 6 and 9 is due to a SL outage). We observe that 50% of the regions (between the 25th and 75th percentiles) are almost completely static, showing a small variation between ± 50 objects after 28 days. The 10th and 90th percentiles remain between ± 250 objects, showing modest object variation rates in most regions. The median value is nearly zero, and the percentiles are almost symmetrical, indicating a similar object creation and destruction rate. In fact, our traces show that the total number of objects in SL remains approximately constant over time. Notice the presence of two drops between days 20-25 and 25-27. During these days, the SL website reported that their servers were being updated. Thus, we believe that these drops correspond to objects being lost during server updates and then slowly recovered. Finally, although not shown in Figure 6(b), we observed minimum and maximum variations close to ± 4000 objects. This implies that the regions at the bottom and top 10% of the distribution show a highly unstable behavior, with a large number of objects being continuously created and destroyed. These are mostly regions where users test their objects (e.g., sandbox regions), and which typically erase objects soon after they are created.



(a) CCDF of the Object distribution across regions ; [Object subcrawler]



(b) Percentiles of the variation of the no. of objects per region ; [Object subcrawler]



(c) CCDF of several percentiles of the population CDFs ; [Statistics subcrawler]

Fig. 6. Object and Avatar characteristics.

F. Region Popularity

We use the 6-day trace collected by the Statistic subcrawler in order to analyze the popularity of regions in terms of the number of avatars that visit them.

As the number of avatars in a given region is highly dynamic, we study for each region the evolution of the population with time. For each region we calculate the *population CDF*, i.e., the Cumulative Distribution Function (CDF) of the number of avatars observed during the 6-day period. Since we cannot plot the population CDFs for the $12,765$ monitored regions, we will take a few meaningful percentiles and plot their distribution among all regions.

Figure 6(c) shows the distribution among regions of the 0th, 25th, 50th, 75th, and 100th percentiles of the population CDFs. Note that the 100th and the 0th percentile correspond respectively to the maximum and minimum population observed for a given region. Similarly, the 75th percentile may be interpreted as a peak population, the 50th percentile as a median or typical population, and the 25th percentile as a residual population. Accordingly, Figure 6(c) shows that 30% of the regions are empty all the time, while around 45% of the regions have always less than 5 avatars. The 75th percentile curve overlaps with the 100th percentile one between 0 and 4 avatars. As a consequence, regions whose population is small most of the time have a small population all the time with no exception. The 0th percentile curve indicates that about 30% of the regions are never completely empty. However, the curve rapidly goes to zero, showing that regions with continuous activity are rare, e.g., less than 1% of the regions have a minimum population of 10 avatars. Focusing now on larger populations, we observe that around 5% of regions have at least 30 avatars as a maximum population, but that this number drops to 18 avatars for a peak population (75th percentile) and to 12 avatars for a typical population (50th percentile). Hence, although a non-negligible number of regions are occasionally densely populated, they usually contain few avatars. These results indicate that different SL regions have different population characteristics, and may necessitate different resource provisioning according to their popularity profile.

G. Virtual Groups

We are interested in determining to what degree avatars concentrate in *groups*, i.e., aggregation of avatars where each avatar is within visibility range from each other (35 meters as defined by SL). The rationale is that avatars located within such *virtual groups* are highly likely to interact with each other. The results we present in this Section are obtained from the analysis of the 3-day trace collected by the Map subcrawler with a crawling frequency of 15 minutes.

We estimate the number of virtual groups in a region by using the *k-means* clustering algorithm [20] to partition avatars in circles of radius $r \leq 35$ meters. We proceed as follows. Let n be the number of avatars in a region. The algorithm takes the avatar coordinates $a_i(t) = (x_i, y_i)$ at a time t with $1 \leq i \leq n$, and a number of target partitions k. It then clusters the avatars into k circular areas with center coordinates $c_j = (x_j, y_j)$ and radius r_j, where $1 \leq j \leq k$. We run the algorithm iteratively for increasing values of k until all circles have a radius $r_j \leq 35$ meters. The final value of k gives the number of virtual groups in the region, and c_j the coordinates of the group center.

We use the k-means clustering algorithm since it minimizes the distance of avatars from the center of the virtual group. Note that this algorithm does not track groups that move across the region. However, since avatars in SL tend to have a static behavior [10], this limitation only has a minor impact on our clustering scheme.

1) Virtual Group Sizes: Figure 7 shows the CDF of virtual group sizes across all regions. We observe that 50% of the avatars don't form groups. Surprisingly, 45% of the virtual groups are made of only 2-10 avatars. This could be explained by the *budget of attention* theory [12], which suggests that human beings can only focus their attention to a maximum of 5-9 entities at the same time. Finally, we observe a negligible number of virtual groups with more than 20 avatars. Note that large avatar crowds which extend beyond 35 meters may be split by our algorithm into smaller groups.

2) Points of Interest (POIs): Regions usually contain Points-of-Interest (POIs), i.e., spots which attract several avatars. In order to detect the presence of POIs, we look for virtual groups that are stable with respect to time and location. Therefore, we use a metric that we call the group's *spot lifetime*, which is defined as follows. For every new group, we record its initial center coordinates. Since the group may

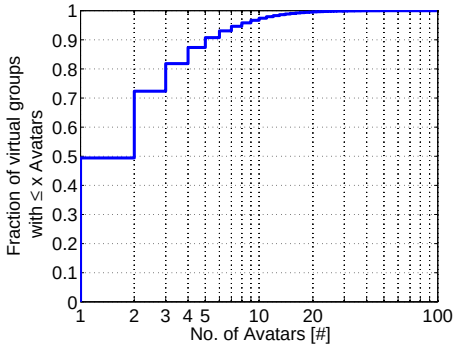


Fig. 7. CDF of virtual group sizes [Map subcrawler].

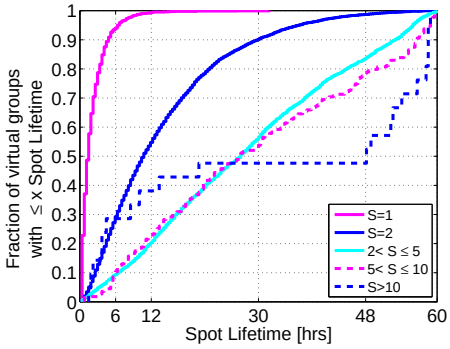


Fig. 8. CDF of spot lifetimes for different average group size S [Map subcrawler].

move or dissolve, we compute its spot lifetime as the time elapsed from its creation until we observe no virtual groups centered within 35 meters (the avatar visibility area) from its initial center coordinates. If the center of a group moves more than 35 meters from its original position, we consider that a new group has formed at the new center coordinates.

Figure 8 shows the CDF of spot lifetimes for all groups and for different ranges of S, the average group size. We notice that groups with large sizes tend to have a larger lifetime. This suggests the presence of POIs near the center of groups with high spot lifetimes. We also observe that 50% of the large virtual groups ($S > 10$) have a rather short spot lifetime. These groups can be event-driven groups, i.e., located near short-lived POIs.[1] Conversely, the remaining 50% have a very long spot lifetime. Intuitively, the area around popular POIs is unlikely to become empty, especially in popular regions, resulting in very long spot lifetimes.

Figure 8 also provides some interesting insight on isolated avatars. Around 40% of these avatars have a spot lifetime of a few minutes. These avatars are most likely exploring a region. Thus, the area traversed by these avatars are unlikely to be POIs. However, 10% have a lifetime between 5 and 32 hours, i.e., they stay at the same spot for a very long time without interacting with any other avatar. It is unlikely that this behavior is coming from human beings, so we suspect that these avatars are computer controlled, i.e., bots. Given that 50% of the virtual groups are composed by a single avatar (see Figure 7), we conjecture that at least 5% (i.e., the 10% with lifetime between 5 and 32 hours) of the entire SL population consists of bots. A similar result is described by Varvello and Voelker in [21] via a complete analysis of bot behaviors in SL.

VI. Playing Second Life

In the previous Section, we have passively measured SL characteristics. This methodology does not allow us to evaluate the Quality of Experience (QoE) perceived by SL users, i.e., how fast and correctly they obtain information about avatars and objects located in their avatar surroundings. In fact, passive measurement misses two fundamental information to capture user QoE: (1) avatar locations at any point in time constructed according to real user inputs (i.e., not filtered from the eye of the server), and (2) the local view of each avatar, i.e., the evolution over time of the avatar and object updates each user receives from the server. We now adopt an *active* measurement strategy in order to collect this information. In the reminder of this Section, we describe how the measurement is performed, and we introduce the metrics to evaluate user QoE.

A. Experimental Setting

Our QoE measurement methodology consists in: (1) collecting avatar traces in a popular SL region, and (2) replaying these traces in empty regions that we use as test-bed. Each time we replay a trace, we collect the local view of each avatar and compare it to the ground truth (i.e., the original trace). We use the player in order to automate the behavior of an avatar within a test-bed region while collecting information about avatars and objects that intersect its AoI. We launch the player from multiple Internet end-points (using Planetlab) and we teleport our controlled avatars into a target SL region. Note that we require that no external avatars, i.e., real SL users, interfere during our experiments to correctly evaluate user QoE. This is why we perform our controlled experiments in unused regions (remember from Section V-F that 30% of SL regions are unused).

We execute the SL player on several Planetlab [11] machines located worldwide in order to emulate realistic network conditions. We select stable Planetlab machines in terms of CPU load, free memory and network activity. Our reference trace is collected in the Japan Resort region which is one of the most popular SL region. Trace collection is described in [10]. We use the one hour period where we observe the maximum number of avatars, i.e., 84 concurrent avatars for a total of 207 different avatars. We build the trajectory of each avatar in the reference trace. Intermediate trajectory position are interpolated from measured positions (every 30 seconds) and avatar speed.

We run our experiments in three unused regions, i.e., generally empty of avatars, and that represent the diversity in object composition observed in SL (Figure 6(a)). We select respectively a *low density* (6 objects), a *normal density* (130 objects) and a *high density* (541 objects) region according to the number of objects they contain. During the experiments,

[1] We visually inspected some of the regions where we found these short-lived POIs, and we verified the presence of concerts and shows.

we also continuously check that no external user connects to the region and interfere with our measurements.

Note that the presence of objects in these test-bed regions may cause avatars to be blocked in their movements as avatar trajectories do not match object location in the testbed region. We solve this issue by making our avatars deviate their mobility pattern before coming back to their original trajectory. While this strategy avoids having avatars stuck in a wall, it also bias the avatar mobility pattern we are injecting in the regions. We analyze the impact of these modifications in avatar mobility when comparing user QoE in the three regions.

B. Metric Definition

The providers of existing Networked Virtual Environments (NVEs) often characterize user QoE by looking at the *cancellation rate*, i.e., the number of user accounts canceled during a given period of time, and/or Mean Opinion Score[2] which is based on user feedback. Given that we cannot compute these two metrics, we choose to compute instead three metrics that we formally define next: the *inconsistency*, the *inconsistency duration* and the *discovery latency*.

1) Inconsistency: An avatar Area of Interest (AoI)[3] is *inconsistent* when it contains wrong information about closeby avatars. Inconsistency can be caused by temporarily missing information as well as incorrect information. AoI inconsistency severely detracts from the user experience as it creates inconsistent views of the world among users.

We call $\mathcal{A}(t)$ the set of avatars connected to a region at time t. For an avatar i, we denote with $ID(i)$ its identity and with $a_i(t)$ its coordinates in the region at time t. Remember that we know the exact location of each avatar at any point in time. Therefore, we can determine $AoI(A,t)$, i.e., the set of avatars that should be included in the AoI of an avatar A at time t, for any A and t. We define $AoI(A,t)$ as follows:

$$AoI(A,t) = \{i \in \mathcal{A}(t)\ s.t.\ dist(i,A) \leq 35\ meters\} \quad (1)$$

Each player continuously logs the information about avatars in its surrounding as informed by the region server. We use this data to compute $SAoI(A,t)$, i.e., the set of avatars that intersect A's AoI at time t, for $\forall A$ and $\forall t$, given the information transmitted by the region server.

The information contained in the AoI of an avatar A at time t is correct if the perception of A's neighbor avatars given the data received by the server (i.e., $SAoI(A,t)$) is consistent with the ground truth given by the mobility traces (i.e., $AoI(A,t)$). This happens if the two sets $SAoI(A,t)$ and $AoI(A,t)$ are identical. Precisely, the following conditions need to hold: (i) $SAoI(A,t)$ and $AoI(A,t)$ have the same size, and (ii) each avatar that intersects A's AoI has the correct identifier and coordinates given the ground truth defined by $AoI(A,t)$. More formally:

$$\begin{array}{l}(i)\ |AoI(A,t)| = |SAoI(A,t)| \\ (ii)\ \forall i \in SAoI(A,t)\ \exists i^{'} \in AoI(A,t) \\ s.t.\ ID(i) = ID(i^{'}) \wedge a_i(t) = a_{i^{'}}(t)\end{array} \quad (2)$$

We say that an avatar AoI is *inconsistent* when Condition 2 is violated. Violation occurs if $AoI(A,t)$ and $SAoI(A,t)$ differ, i.e., when an avatar is in one set but not in the other one. Violation also occurs if an avatar is in both sets but not at the same position. Formally, we define the number of errors $Nerr(A,t)$ as:

$$\begin{array}{l}Nerr(A,t) =\mid \{(AoI(A,t) \cup SAoI(A,t))\ s.t. \\ (i \in AoI(A,t) \wedge i \notin SAoI(A,t)) \vee \\ (i \notin AoI(A,t) \wedge i \in SAoI(A,t)) \vee \\ (i \in AoI(A,t) \wedge i^{'} \in SAoI(A,t) \wedge ID(i) = ID(i^{'}) \wedge \\ a_i(t) \neq a_{i^{'}}(t))\} \mid\end{array} \quad (3)$$

Then, the *inconsistency* is computed as:

$$\frac{Nerr(A,t)}{|(AoI(A,t) \quad \cup \quad SAoI(A,t)|} \quad (4)$$

The *inconsistency* as defined in 4 takes values between 0 and 1 where 0 means that all the information contained in an avatar AoI is correct, and 1 means that all the information contained in an avatar AoI is wrong.

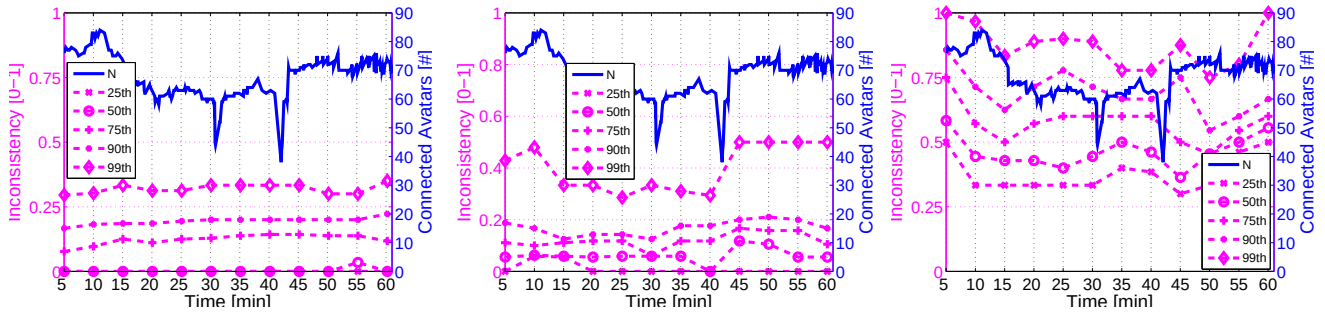
2) Inconsistency Duration: Avatars in NVEs require to be constantly updated about changes in the nearby avatar states. This is fundamental in order to guarantee NVE *interactivity* [14]. We introduce the *inconsistency duration* as a measure of interactivity. We define the inconsistency duration as the time an avatar needs to achieve a consistent view of the avatars in its AoI. We compute the inconsistency duration as follows: we start a timer whenever an inconsistency is detected in $SAoI(A,t)$ (i.e., inconsistency greater than 0) and we stop it as soon as $SAoI(A,t)$ becomes consistent again (i.e., inconsistency equals 0).

3) Discovery Latency: AoI changes as a consequence of avatar movements. Thus, new avatars and objects can enter or leave the AoI. In SL, the region servers transfer the virtual object descriptions over the Internet. Intuitively, this download time should be kept small in order to guarantee a smooth rendering of the virtual world. We call *discovery latency* the time an avatar needs to retrieve the virtual objects contained in its AoI.

We measure the discovery latency by parsing the traffic received by a player in order to isolate the packets that refer to the virtual objects. When a *region packet* (Section III-D) is received, we detect that the server is currently updating the avatar AoI and we start a timer. We stop this timer when no more region packets are received, indicating that all objects that intersect the avatar AoI have been correctly received. In fact, lost or corrupted packets trigger retransmissions at the SL servers. In case the avatar moves before it has correctly received a description of all virtual objects in its AoI, we simply stop the timer. Then, we re-start the timer as soon as the avatar stops at a new place. Thus, we do not estimate the discovery latency during the time an avatar moves across the region.

[2]http://en.wikipedia.org/wiki/Mean_opinion_score

[3]AoI=sphere with a 35 meters radius as defined in Section III-C.



(a) low density (b) normal density (c) high density

Fig. 9. Percentiles of the avatar inconsistency over time.

C. User QoE in SL

1) Inconsistency: We evaluate the inconsistency for each avatar every 200 ms or anytime a modification of the AoI occurs. Figure 9 plots the evolution over time of the 25th, 50th, 75th, 90th and 99th percentiles of the inconsistency distribution among avatars. Each plot also shows the evolution over time of the number of avatars connected to the region.

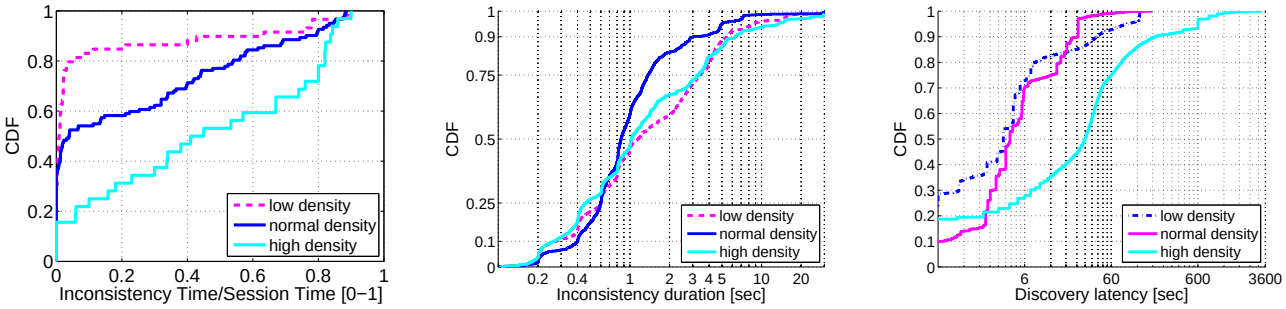
We observe in Figure 9 that the inconsistency values measured in the low density region are very stable over time: 99% of the time, the consistency of avatars is higher than 70%. This is probably because the server resources are not fully utilized and so user QoE is not impacted by the variation in the number of concurrent users. Conversely, in both the normal and high density regions the inconsistency curves vary significantly over time. SL users that interact in these two regions suffer from more inconsistency periods. For example, the median value of inconsistency reaches 0.4-0.5 in the high density region. As expected, the number of objects contained within a region has a significant impact on the SL server capability to maintain consistent information for all users. This is due to the effort required for a SL server to manage virtual objects [4], that leaves to a server only few free resources to perform the visibility computation (Section III-B) for each avatar.

The increase of inconsistency we measure in the high density region could also be due to the high density of objects that prevent avatar to move exactly as indicated by the mobility traces (Section VI-A). For example, larger avatar groups may be created, causing additional load at the server and making the comparison with other regions unfair. We compare the number of avatars intersecting each avatar AoI during the experiments. We find out that in the high density region avatars have in average less than 10% more avatars intersecting their AoIs than in the low and normal density region. We believe that such a small variation of the avatar mobility does not justify the increase in avatar inconsistency we measured in the normal and high density regions. Thus, the increase in inconsistency is not due to the variation of avatar mobility caused by the high density of objects.

We now want to understand how frequently inconsistency events affect an avatar during its journey in SL. Figure 10(a) plots the CCDFs of the ratio between the sum of the durations of an avatar inconsistency periods and the total time the avatar stays in a region. We observe again that in the high density region, avatars are more often inconsistent than in the other two regions. For example, in the high density region more than 90% of the avatars suffer from inconsistency, whereas this number is divided by three in both the low and normal density region. Interestingly, Figure 10(a) shows that avatars with an almost completely inconsistent SL experience are equally likely in the three regions, e.g., about 8% of the avatars have an inconsistent AoI during about 80-90% of their SL journey. The reason beyond this phenomenon is that the SL server spends a lot of time to correctly accomplish avatar login/logout as we will investigate next. Subsequently, avatars with very short session times have an inconsistent AoI most of the time.

2) Interactivity: We now analyze the inconsistency duration measured in the three regions in order to understand how fast SL servers react to avatar inconsistencies. Figure 10(b) shows the CDFs of the inconsistency duration values measured in the three regions. We notice that avatar inconsistency periods last for more than one second in 40%-50% of the cases. Even worse, 5% to 10% of the inconsistency periods last for more than 5 seconds. These inconsistency duration values are very long if we consider that acceptable latency values in virtual worlds vary between 300 ms and 1 sec [14], unveiling that SL servers provide poor interactivity to their avatars. Figure 10(b) also shows that the inconsistency duration measured in the three regions can reach up to 20 seconds. These extremely high values of inconsistency duration are measured in presence of *churn*, i.e., avatar login and logout operations, and they are due to the fact that SL servers spend several seconds to accomplish these operations. Surprisingly, Figure 10(b) shows that the number of objects contained within a region does not impact the inconsistency duration, e.g., 50% of the inconsistency values measured in the normal density region are lower than the values measured in the low density region. In the absence of official information from Linden Labs, we conjecture that SL servers choose, when over-loaded, to increase the number of inconsistencies but to shorter their durations.

3) Discovery Latency: Finally, we analyze how fast avatars retrieve the objects located in their surroundings. Figure 10(c) shows the CDFs of the discovery latency measured in the three regions. Not surprisingly, the higher the density of content in a region, the longer it takes for an avatar to reconstruct the virtual world. The median discovery latency grows from about 4 seconds in the low and normal density region to about 30 seconds in the high density region. In



(a) CCDFs of the fraction of time an avatar is inconsistent

(b) CDF of the duration of inconsistencies

(c) CDFs of the discovery latency

Fig. 10. Inconsistency Analysis.

our experiments, we verify that the user's network connection is not a bottleneck. Therefore, this result indicates that the longer discovery latency measured in the high density region is due to the fact that the server limits its outgoing traffic rate. However, Figure 10(c) also shows that the curves for the low and normal density region overlap for latency values larger than 20 seconds, i.e., high discovery latency values are more likely in the low density region than in the normal density region, which is counter-intuitive. This phenomenon is due to the fact that object density is not uniform in a region. Therefore, these high values of discovery latency measured in the low density region happen in portions of the region where the local density is much higher than the average object density of the normal density region.

Finally, Figure 10(c) shows that the discovery latency can reach extremely high values, e.g., a couple of minutes in both the low and normal density region and up to one hour in the high density region. This means that some avatars are never able to correctly render the virtual world in their surroundings during their entire SL journey.

VII. Enhancements to Second Life

This Section proposes several improvements to SL's design. Our suggestions are motivated by the observed characteristics of the virtual world, not by the specific implementation of the SL protocol. Therefore, they may be useful for other Networked Virtual Environments (NVEs) that show object and avatar characteristics similar to SL.

A. Object Management

Currently, SL servers transfer virtual objects to the users according to the location of their avatars in the virtual world. Moreover, users can activate a client side cache for recently visited places that greatly reduces network traffic [4].

The social aspect of SL encourages users to spend their time together: they organize in groups (Figure 7) and they make friends [10], [21]. This means that there is a high chance that every time an avatar connects to SL it will interact with a set of avatars it has repeatedly encountered before. This suggests that the SL caching system can be enhanced by taking into account the presence of highly synchronized avatars.

The main idea is to build a distributed cache using the information provided by the social network, i.e., that some avatars meet frequently in the virtual world. Therefore, avatars could first attempt to download data from their friends, and only resort to contacting the server when no friends are available. Moreover, having each friend store a different piece of content would reduce the size of the client cache.

Several Points-of-Interest (POIs) can be easily identified within each Region (Figure 8). These POIs are portions of the virtual world that avatars are most likely to visit. Thus, the objects located close to these POIs are very likely to be transmitted from the server to the clients.

The server could use the statistics about POIs to predict avatar behaviors. When an avatar enters a region, the server first transmits the data about its immediate surroundings. Whenever free server bandwidth is available, the server also transmits data about nearby POIs that the client can store in its cache. In this way, avatars moving toward a POI may have already downloaded the objects they need before reaching it, reducing the discover latency.

B. Avatar Management

The *avatar management* in NVEs consists of informing each avatar about the status of its neighbor avatars in real time. In the past few years, several *distributed avatar management* have been proposed [22], [23], [24] in which clients connect to other clients whose avatars are close in the virtual coordinate space. The Delaunay Network [22], [25] have received comparatively more attention from the research community. Interestingly, researchers show that a Delaunay-based avatar management performs poorly when large virtual groups are present, or when avatars move quickly as in multi-player games [26]. The nature of the SL virtual world favors avatar socialization rather than quick movements [10], [21]. Moreover, our analysis shows that avatars tend to organize in small groups of 2-10 avatars (see Figure 7). Thus, a distributed avatar management based on the Delaunay Network seems a promising approach to improve user QoE in SL and SL-like virtual worlds.

In order to verify our claim, we extend the player (Section IV-B) to allow direct communication among end-users organized as a Delaunay Network. To do so, we intercept the avatar state updates generated by the player and transmitted to the SL server and we duplicate them into a Delaunay Network constructed among active users. Then, we reproduce the experiment described in Section VI by focusing on a

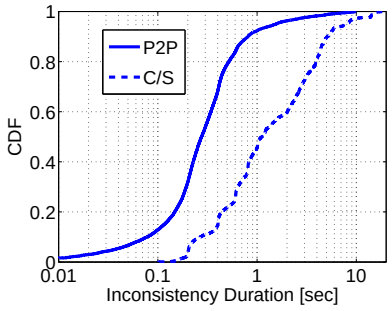


Fig. 11. CDF of the inconsistency duration; Client/Server (C/S) vs Peer-to-Peer (P2P) Second Life.

SL region empty of any objects. In this way, objects do not interfere with avatar mobility patterns. A complete description of the player modifications as well as of the experiments can be found in [27].

Figure 11 compares the inconsistency duration (Section VI-B2) measured in Peer-to-Peer (P2P) and Client/Server (C/S) Second Life. P2P clearly outperforms the current C/S design. About 90% of the time, inconsistency in P2P lasts less than 1 second, i.e., P2P is about 5 times faster than C/S. This result is very promising if we consider that acceptable values of interactivity in on-line games vary between 300 ms and 1 sec [14]. However, despite the fact that the Delaunay Network allows a direct communication among SL users Figure 11 shows that only 20% of the inconsistencies in P2P last less than 150 ms, i.e., a value comparable to common network latencies over the Internet [28]. Since avatars tend to form groups in SL (see Figure 7) the transmission of avatar state updates to all interested avatars require multiple hops in the Delaunay Network. This operation generates additional latencies that increase the inconsistency duration. This result suggests that further reduction of the inconsistency duration can be attained by introducing alternative strategies for the dissemination of avatar state updates within groups of avatars.

VIII. Conclusions and Future Work

Second Life (SL) has received a lot of press coverage and even some major companies and governments have created their own SL region. We have carried out a detailed evaluation of SL and made some interesting observations. Almost 30% of the regions do not attract any visitors, and only a few regions are popular. Moreover, the number of concurrent participants barely reaches $50,000$. So one is tempted to paraphrase the famous American comedian W. C. Fields saying "I went to Second Life and it was closed".

Our results also indicate that the user Quality of Experience (QoE) is generally poor: most of the time avatars have an inconsistent view of their surrounding, missing information about other avatars or visualizing them in an incorrect location. This inconsistency takes generally more than one second to be resolved, which impacts user *interactivity*. Finally, in a region crowded with virtual objects it generally takes more than one minute for a user to retrieve the description of the virtual world in its surroundings.

The static nature of most SL regions suggests that a distributed caching system as well as a prefetching algorithm for virtual objects can help reduce the server traffic and the discovery latency experienced at the clients. Moreover, a distributed avatar management could be highly efficient at increasing interactivity among avatars. As future work, we envisage to deploy an SL client that relies on Peer-to-Peer for both object and avatar management.

References

[1] S. Singhal and M. Zyda, *Networked Virtual Environments: Design and Implementation*. New York, NY, USA: ACM Press/Addison-Wesley Publishing Co., 1999.
[2] World of Warcraft, http://www.worldofwarcraft.com.
[3] Second Life, http://www.secondlife.com.
[4] S. Kumar, J. Chhugani, C. Kim, D. Kim, A. Nguyen, P. Dubey, C. Bienia, and Y. Kim, "Second life and the new generation of virtual worlds." *IEEE Computer*, vol. 41, no. 9, pp. 46–53, 2008.
[5] F. Stenio, K. Carlos, S. Djamel, M. Josilene, and A. Rafael, "Traffic Analysis Beyond This World: the Case of Second Life," in *Nossdav'07*, Urbana-Champaign, IL, USA, June 2007.
[6] K. James and C. Mark, "Traffic Analysis of Avatars in Second Life," in *Nossdav'08*, Braunschweig, Germany, May 2008.
[7] R. Antonello, S. Fernandes, J. Moreira, P. Cunha, C. Kamienski, and D. Sadok, "Traffic Analysis and Synthetic Models of Second Life," *Multimedia Systems*, vol. 15, no. 1, pp. 33–47, Feb. 2009.
[8] C.-A. La and P. Michiardi, "Characterizing User Mobility in Second Life," in *WOSN'08*, Seattle, USA, August 2008.
[9] H. Liang, R. N. D. Silva, W. T. Ooi, and M. Motani, "Avatar mobility in user-created networked virtual worlds: Measurements, analysis, and implications," *Multimedia Tools and Applications, Special Issue on Massively Multiplayer Online Gaming Systems and Applications*, vol. 45, no. 1-3, pp. 163–190, 2009.
[10] M. Varvello, F. Picconi, C. Diot, and E. Biersack, "Is There Life in Second Life?" in *Conext'08*, Madrid, Spain, Dec. 2008.
[11] Planetlab, https://www.planet-lab.org/.
[12] G. A. Miller, "The Magical Number Seven, Plus or Minus two: Some Limits on Our Capacity for Processing Information," *Psychological Review*, vol. 63, pp. 81–97, 1956.
[13] J. Pang, F. Uyeda, and J. R. Lorch, "Scaling Peer-to-Peer Games in Low-Bandwidth Environments," in *IPTPS'07*, Bellevue, Washington, USA, February 2007.
[14] M. Claypool and K. Claypool, "Latency and Player Actions in Online Games," *Commun. ACM*, vol. 49, no. 11, pp. 40–45, 2006.
[15] libsecondlife, http://www.libsecondlife.org/.
[16] Wikipedia, http://en.wikipedia.org/wiki/Real_estate_(Second_Life)(accessedJune15,2009).
[17] Second Life, http://blog.secondlife.com/ (accessed June 15, 2009).
[18] SL Login, http://secondlife.com/app/login/.
[19] H. Liang, M. Motani, and W. T. Ooi, "Textures in second life: Measurement and analysis," in *ICPADS '08*. Washington, DC, USA: IEEE Computer Society, 2008, pp. 823–828.
[20] K. Alsabti, S. Ranka, and V. Singh, "An Efficient K-Means Clustering Algorithm," 1997. [Online]. Available: http://www.cise.ufl.edu/~ranka/
[21] M. Varvello and G. M. Voelker, "Second Life: a Social Network of Humans and Bots," in *Nossdav'2010*, Amsterdam, Netherlands, June 2010.
[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] J. Keller and G. Simon, "SOLIPSIS: A Massively Multi-Participant Virtual World," in *Proc. of PDPTA*, Las Vegas, Nevada, USA, 2003.
[24] A. Bharambe, J. Pang, and S. Seshan, "Colyseus: A Distributed Architecture for Online Multiplayer Games," in *NSDI '06*, 2006. [Online]. Available: http://www.comp.nus.edu.sg/~bleong/hydra/related/bharambe06colyseus.pdf
[25] M. Ohnishi, R. Nishide, and S. Ueshima, "Incremental Construction of Delaunay Overlay Network for Virtual Collaborative Space," in *Proc of. C5*, Cambridge, MA, USA, January 2005.
[26] H. Backhaus and S. Krause, "Voronoi-Based Adaptive Scalable Transfer Revisited: Gain and Loss of a Voronoi-Based Peer-to-Peer Approach for MMOG," in *Proc. of Netgames*, Mellbourne, Australia, September 2007.
[27] M. Varvello, S. Ferrari, E. Biersack, and C. Diot, "A Distributed Avatar Management for Second Life," in *NETGAMES'09*, Paris, France, November 2009.
[28] J. Winick and S. Jamin, "Inet-3.0: Internet Topology Generator," University of Michigan, Tech. Rep. CSE-TR-456-02, 2002.

Matteo Varvello is a researcher at Alcatel-Lucent in Holmdel (New Jersey) since January 2010. In November 2006, he earned his MSc in Networking Engineering from Polytechnic of Turin (Italy), graduating cum laude. As part of his Master studies, he spent one year at Eurecom in Sophia Antipolis (France), where he obtained a Research Master in Network and Distributed System. In December 2009, he received a Doctoral degree in Computer Science from Telecom Paris (France). His current research interests include peer-to-peer, content-centric networking, cloud computing and energy efficiency.

Stefano Ferrari is a software engineer at Cisco System in Rolle (Switzerland) since December 2009. In April 2008, he received his MSc in Software Engineering from Polytechnic of Turin (Italy). His education was enriched with one year of study at EURECOM Institute (France) where he obtained a diploma in Network and Distributed Systems. His present main occupation is on design and development of advanced routing features for Internet core routers.

Ernst Biersack studied computer science at the Technische Universitt Mnchen and at the University of North Carolina at Chapel Hill. He received the Dipl. Inf. (M.S.) and Dr. rer. nat. (Ph.D.) degrees in computer science from the Technische Universitt Mnchen, Munich, Germany, and the Habilitation Diriger des Recherches from the University of Nice, France. From March 1989 to February 1992, he was a Member of Technical Staff with the Computer Communications Research Group of Bell Communications Research, Morristown, NJ. Since March 1992, he has been a Professor in telecommunications at Institut Eurecom, Sophia Antipolis, France. His current research is on peer-to-peer systems, network tomography of TCP connections, and LAS scheduling in edge routers.

Christophe Diot received a Ph.D. degree in Computer Science from INP Grenoble in 1991. He was with INRIA Sophia-Antipolis (Oct. 93-Sep.98), Sprint (Oct. 98-Apr. 03), and Intel Research in Cambridge, UK (May 03-Sep. 05). He joined Thomson in October 2005 to start and manage the Paris Research Lab (http://thlab.net). Diot has been a pioneer in multicast communication, DiffServ, Internet measurements, and more recently Pocket Switched Networks. Diot's research activities now focus on advanced p2p communication services. Diot is the Thomson Scientific Council chair and an ACM fellow.