Distributed Avatar Management for Second Life

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

***Abstract*—Second Life (SL) is currently the most popular *social virtual world*, i.e., a digitalization of the real world where *avatars* can meet, socialize and trade. SL is managed through a Client/Server (C/S) architecture with a very high cost and limited scalability. A scalable and cheap alternative to C/S is to use a Peer-to-Peer (P2P) approach, where SL users rely only on their own resources (storage, CPU and bandwidth) to run the virtual world. We develop a SL client that allows its users to take advantage of a P2P network structured as a *Delaunay* overlay. We compare the performance of a P2P and C/S architecture for Second Life, executing several instances of our client over Planetlab and populating a SL region with our controlled avatars. Avatar mobility traces collected in SL are used to drive avatar behaviors. The results show that P2P improves user experience by about 20% compared to C/S (measured in term of consistency). Avatar interactivity is also 5 times faster in P2P than in C/S.**

I. INTRODUCTION

Social virtual worlds are interactive environments accessible from the Internet. Multiple participants interact in social virtual worlds through an *avatar*, i.e., the digital representation of a user. The virtual world consists of several lands called *regions* where avatars can create *objects* such as a car, a tree or a building. Second Life (SL), launched in 2003 by Linden Lab, is the most famous social virtual world counting more than 16 Millions registered users in September 2009.[1]

SL leverages a Client/Server (C/S) architecture where each of the 18,000 region is managed by a dedicated server [10]. Users run "thin" clients that simply perform the three-dimensional rendering of the virtual world and cache the virtual objects located in recently visited regions. The main tasks of SL servers are: *avatar management* and *object management*. The avatar management consists in updating each avatar about the status of its neighbor avatars in real time. The object management consists in maintaining the user-created objects over time, and informing each avatar about the objects in their visibility area.

Researchers have recently shown that SL suffers from scalability problems. Servers get easily overloaded with as little as 20 concurrent avatars [20], and are forced to significantly slow down their region's virtual time to artificially reduce avatar activity. Nevertheless, SL makes also an intensive use of network resources [17][7].

A scalable and cheap alternative to C/S is to use a Peer-to-Peer (P2P) approach, where the virtual world is managed in a distributed way leveraging end-user resources. Several P2P architectures for virtual worlds have been proposed [6][3][8]. These P2P architectures mainly address the problem of a distributed avatar management, i.e., neighbor avatars discovery and avatar state updates dissemination. For the object management, these solutions simply assume that the virtual world object composition is mostly stored at the clients.

In this work, we design and evaluate a *distributed avatar management* for Second Life. Object management is out of the scope of this work[2]. Specifically, we develop a P2P-SL client that includes the current C/S avatar management and a distributed one. To do so, we add to the SL client a P2P module that leverages the *Delaunay Network* [6][14] in order to disseminate and receive avatar state updates. We choose the Delaunay Network as it is a well-known P2P strategy for avatar management and it holds desirable properties for SL-like environments [18].

We compare P2P versus C/S Second Life by focusing on the Quality of Experience (QoE) perceived by multiple SL users. To do so, we execute several instances of the P2P-SL client over Planetlab[3] and we populate a SL region with our controlled avatars. Avatar mobility traces extracted from SL [20] are used to reproduce real avatar behaviors.

We show that a distributed avatar management for SL always outperforms the current C/S design. P2P Second Life produces a gain of correctness in the user experience of about 20% compared to C/S. Nevertheless, 90% of the times inconsistency in P2P is solved in less than one second, i.e., 5 times faster than in C/S. This result is very promising since acceptable values of interactivity in on-line games vary between 300 ms and 1 sec [5]. However, as already observed in [1][18], we confirm experimentally that the Delaunay Network suffers from avatar groups, fast movements and churn.

II. RELATED WORK

The success of Second Life (SL) has recently attracted the attention of the research community. Fernandes et al. [17] propose the first study related to SL. They collect the traffic exchanged in a C/S session, measuring bandwidth consumption, packet size and packet inter-arrival times. They show that SL makes an intensive use of network resources, much more than other 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 stands in SL consumes about 20 Kbps in the down-link, whereas as

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

[2] The reader can found an experimental evaluation of a distributed object management for SL in [19].

[3] https://www.planet-lab.org/

978-1-4244-5605-5/09/$26.00 ©2009 IEEE

soon as the avatar moves the down-link traffic grows up to 110 Kbps.

Kumar et al. [10] 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 [17] they also analyze the network traffic exchanged between client and server. Their results confirm the high bandwidth cost 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 server computational power. As SL-like virtual worlds are expected to become more complex and realistic several CPU cores will be required.

Varvello et al. [20] perform a large scale data collection in SL. They design and deploy a *crawler* application that collects traces of avatar behaviors, object characteristics and server performances in the public SL regions. The analysis of their traces show that about 30% of the regions do not attract any visitor, and that in the few popular regions servers are overloaded most of the time. Interestingly, they show that avatars behave similarly to humans, visiting a few popular places where they meet friends.

Liang et al. [11] use a similar approach to [20] and collect mobility traces of 84,208 avatars spanning 22 SL regions over two months. Based on their analysis of avatar mobility, the authors suggest an hybrid avatar mobility model that incorporates both random way-point mobility model (for regions poor of objects) and pathway mobility model (for regions rich of objects).

Our work is different from the previous work conducted on SL as we propose an enhancement to its architecture, rather than study its characteristics. Specifically, we use the Delaunay Network to build a distributed avatar management for SL. A similar solution was originally introduced by Hu et al. [6] in VON. VON is a P2P architecture for virtual worlds based on a Voronoi diagram, i.e., the dual of the Delaunay triangulation. In VON, each user locally contributes to the construction of a Voronoi graph based on a spatial metric in the virtual world. The authors show by simulation that VON is a scalable design for virtual worlds.

Backus and Krause [1] discuss benefits and problems of using Voronoi diagrams in P2P virtual worlds. They look at bandwidth usage, scalability and consistency of Voronoi-based virtual worlds. They show that while Voronoi-based virtual worlds perform quite good when avatars move according to a Random Waypoint model[4], consistency is greatly reduced when groups of avatars get close and avatar speed exceeds the mean distance among avatars.

Our work is complementary to [6] and [1] as we extend the evaluation of the Delaunay Network through real experiments performed in SL. We also analyze strengths and weaknesses of the Delaunay Network when dealing with user QoE. Furthermore, we develop a concrete enhancement to the current SL architecture.

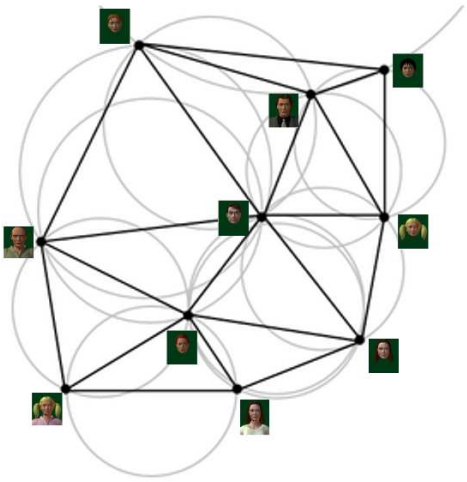


Fig. 1. Distributed avatar management leveraging the Delaunay Network.

III. BACKGROUND

A. Second Life

The virtual world of SL is divided into lands called regions that have a fixed size of 256x256 meters. Each region can have up to four neighbor regions that are independents among them. The region appearance is defined by the land features and the user-generated objects.

Users join the SL community by registering an avatar at the SL website[1]. This registration requires filling out an on-line form with private information and a valid e-mail address. Afterwards, a user explores SL by controlling its avatar movements, e.g., it can *walk*, *run*, *fly* and *teleport*.[5] Avatars have a limited visibility area called Area of Interest (AoI). The AoI consists of a sphere centered on the avatar coordinates with 35 meters radius.

The C/S architecture of SL leverages a cluster of servers each dedicated to a different task, e.g., login and region simulation. Since we perform experiments directly on a *region server* we now shortly describe it. A complete description of the SL Client/Server architecture can be found in [10][17].

A region server is responsible for a SL region. The main roles of a region server are: (i) maintain the state of its region (e.g., user-created objects and land features) over time, (ii) perform the *visibility computation* [10], i.e., identify for each avatar located in the region the information about objects, land features and avatars that need to be transmitted, (iii) manage chat among avatars located within the region. A region server handles a maximum of 100 concurrent avatars. However, larger population are possible by mirroring the region server [10].

[4]http://ica1www.epfl.ch/RandomTrip

[5]The teleport operation allows avatars to rapidly cover large distances.

B. Delaunay Network

The *Delaunay Network* [4] is an overlay network whose topology is defined by a Delaunay triangulation. In the following, we give a formal definition of the Delaunay triangulation:

Definition 1: The Delaunay triangulation of a set of N points in $\Re^2$ is a triangulation of points $DT(N)$ such that no point p lies inside the circumcircle of any triangle in $DT(N)$.

The coordinates of avatars in the virtual world are used to generate the Delaunay triangulation, and consequently build a Delaunay Network among the respective end-users [14][6]. In this way, the interactions among avatars on a proximity metric basis are reflected into Internet connections among their end-users. Figure 1 shows an example of a Delaunay triangulation constructed among avatars in a virtual world.

An avatar that participates to the Delaunay Network continually monitors the position of its one hop neighbors in order to maintain the triangulation over time. As avatars move, Delaunay links are added and removed to maintain a valid and consistent triangulation. A more detailed description of the Delaunay Network can be found in [14].

Researchers have shown that the performance of the Delaunay Network may deteriorate in presence of large avatar groups, or when avatars move very fast [1][18]. Avatars in SL are mostly static, and tend to organize in small groups of 2-10 avatars [20][11]. Thus, the Delaunay Network seems a promising approach for the distributed avatar management in SL and SL-like virtual worlds.

IV. P2P-SL CLIENT

SL servers currently perform the *avatar management* by constantly updating each avatar about its nearby avatar states. In this Section, we design a P2P-SL client that performs a distributed avatar management.

A. Design

We use the libsecondlife[6] libraries to deploy a P2P-SL client. The P2P-SL client includes the fundamental features of the official SL client, e.g., login/logout operations and avatar movements, while removing the CPU intensive operations, e.g., the three dimensional rendering of the virtual world.

The P2P-SL client does not require to be human-controlled. Avatar traces, e.g., movement and churn, can be used to automate the client operations. The innovative feature of the P2P-SL client is the possibility to directly communicate with other P2P-SL clients without the need of a server.

In order to permit direct communications among SL users, the P2P-SL client implements the Delaunay Network protocol using HyperCast.[7] HyperCast is a set of Java libraries that allows to build several overlays such as the Delaunay Network [4] and Pastry [16]. HyperCast provides to the P2P-SL client a *Neighborhood table* that contains the routing information towards the one-hop Delaunay neighbors of a peer. A complete description of HyperCast can be found in [12].

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

[7]http://www.hypercast.org.

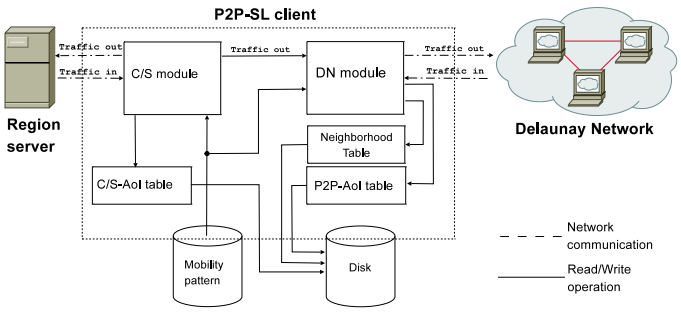


Fig. 2. The P2P-SL client design.

We build a distributed avatar management for SL on top of the Delaunay Network constructed among P2P-SL clients. To do so, we intercept the avatar state updates generated by the client and transmitted to the SL server and we duplicate them into the Delaunay Network. Note that the avatar state updates sent to the server are now redundant as each avatar already manages its state updates via the Delaunay Network. However, we experienced that suppressing or reducing this traffic causes two main problems: (i) the server continually queries the P2P-SL client about its avatar state, (ii) the server can label our avatars as "misbehaving" and temporarily exclude them from its region. Nevertheless, this strategy is extremely useful to perform a fair comparison between P2P and C/S Second Life (Section VI).

We use UDP as transport layer protocol for the dissemination of the avatar state updates over the Delaunay Network. The official SL client also uses UDP communication to transport the avatar traffic. Similarly to the SL design choice, we opted for UDP since the avatar state updates are transmitted at a constant rate, and do not need delivery guarantees.

We now propose a detailed description of the P2P-SL client (Figure 2). In the following, we call *node* the representation of an avatar in the Delaunay triangulation.

- The *C/S module* is the core of the P2P-SL client. It manages the communication between the P2P-SL client and a SL server, i.e., avatar, object and land discovery. It receives as input the avatar mobility pattern that it uses to emulate a realistic avatar behavior on a SL region. Most importantly, it duplicates the traffic dedicated to the avatar state management and forward it to the Delaunay Network module.
- The *C/S-AoI table* is the data structure that contains up-to-date avatar state information for the avatars located within an avatar AoI. This data structure is constantly updated using the avatar traffic received from the SL server. A snapshot of the C/S-AoI table is copied to the disk every 200 ms or when a modification of its content occurs.
- The *Neighborhood table* is the data structure maintained by the HyperCast libraries. It contains routing information towards the Delaunay one-hop neighbors of a peer.

A snapshot of the Neighborhood table is copied to the disk every 200 ms or when a modification of its content occurs.

- The *Delaunay Network module* manages the Delaunay Network, and the transmission/reception of avatar state updates. It receives as input the avatar mobility pattern that it uses to update the coordinates of the respective node in the Delaunay triangulation. This information is propagated to the nodes contained in the node's Neighborhood table via heartbeat messages at a fast or slow rate. The fast rate (1 message every 200 ms) is used during the join of a new node and in case of unstable neighborhood, e.g., when a node changes position or a newcomer joins the network. The slow rate (1 message every sec) is used when the neighborhood is stable. Rate values are chosen according to [12]. The Delaunay Network module receives as input the traffic generated by the C/S module for the avatar state updates. This traffic is flooded into the Delaunay Network with *AoI filtering*, i.e., packets are not forwarded farther than an avatar AoI (35 meters as default in SL). Local forwarding decisions at nodes are made using *compass routing*, e.g., among three nodes A, B and C that are all one-hop Delaunay neighbors of a node D, the node that forwards the avatar state update received by a node R is the node that minimizes the angle it forms with R and D [9].
- The *P2P-AoI table* is the data structure that contains up-to-date avatar states information for the avatars located within an avatar AoI. This data structure is constantly updated using the avatar traffic received from the Delaunay Network module. A snapshot of the P2P-AoI table is copied to the disk every 200 ms or when a modification of its content occurs.

V. EXPERIMENTAL METHODOLOGY

We compare a P2P versus a C/S architecture for Second Life by focusing on the user Quality of Experience (QoE). Measuring user QoE in a virtual world is a challenging task. Currently, game providers 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[8] that is based on user feedback. Given that we cannot rely on these two techniques, we choose to study the user QoE as perceived by multiple SL users. More precisely, we compute two metrics that we define next: the *inconsistency* and the *inconsistency duration*.

A. Metric Definition

- *Inconsistency* - In order to have meaningful interactions among avatars, each avatar needs to have correct information about the avatars contained in its AoI. Temporarily missing information as well as incorrect information make an avatar AoI inconsistent. Inconsistency is measured as the fraction of wrong avatar information contained within an avatar AoI, i.e., additional/missing avatars and avatar located at wrong coordinates. The inconsistency 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.
- *Inconsistency Duration* - User experience in virtual worlds is positive when avatars perceive quickly enough changes in the nearby avatar states [5]. This guarantees interactivity among avatars. The inconsistency duration is the time an avatar needs to achieve a consistent view of the avatars in its AoI. We compute the inconsistency duration by starting a timer whenever an inconsistency is detected in an avatar AoI and stopping it as soon as the avatar's AoI becomes consistent again.

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

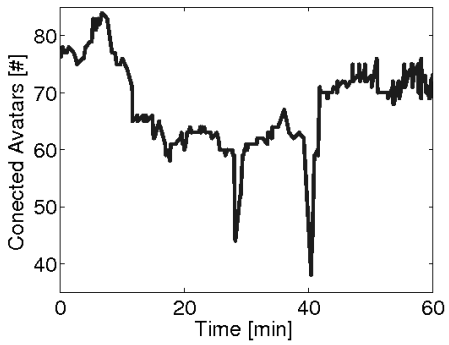


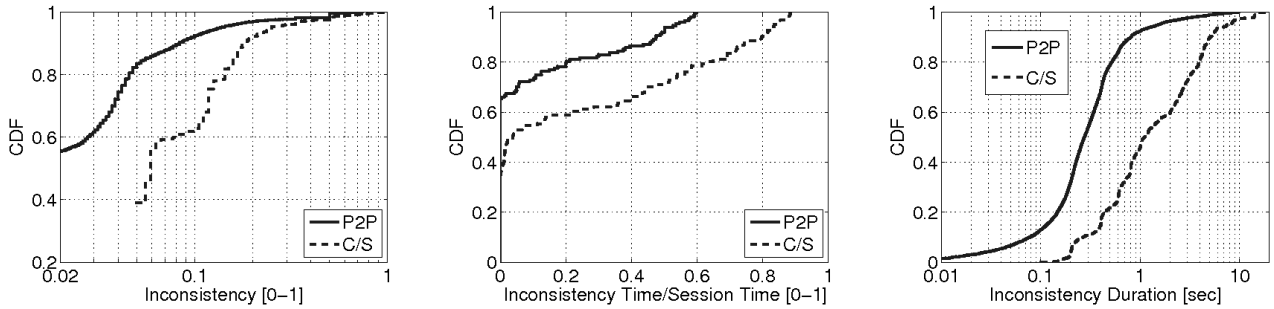
Fig. 3. Evolution over time of the no. of connected avatars during the experiment.

B. Experimental Settings

We use the P2P-SL player in order to automate the behavior of an avatar within a SL region while collecting information about avatars intersecting its AoI as indicated from the SL server and the P2P network. We launch the player from multiple points on the Internet and we teleport our controlled avatars into a target SL region. In this way, we have direct access to the local view of each avatar through the traces collected by each player. Moreover, we can easily build the "ground truth" of avatar locations at any point in time since we control all avatars that interact in the region. Note that we require that no external users, i.e., real SL users, interfere during our experiments in order to correctly evaluate user QoE.

We execute the P2P-SL player on several Planetlab[3] 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 automated avatars reproduce real avatar behaviors using a *mobility trace*, i.e., a trace that contains avatar coordinates at any given time as well as the avatar session durations. We extract the mobility traces from the avatar traces collected by Varvello et al. [20] in the Japan Resort region, i.e., one of the most popular SL region. We take from the avatar traces the one hour period where we observe the maximum number of



(a) CDF of the inconsistency.

(b) CDF of the fraction of time an avatar AoI is inconsistent.

(c) CDF of the inconsistency duration.

Fig. 4. Quality of Experience analysis ; Client/Server (C/S) vs Peer-to-Peer (P2P) Second Life

avatars, i.e., 84 concurrent avatars for a total of 207 different avatars during one hour, and for each avatar we generate its mobility trace. Figure 3 shows the evolution over time of the avatar population extracted from the Japan Resort region and that we reproduce in SL. Due to a crawling frequency of about 30 seconds [20], we only have few samples of avatar movements. In order to reproduce fluid avatar movements in the region, whenever an avatar changes coordinates between two successive crawling snapshots we compute the avatar speed and we interpolate its trajectory.

We run our experiments in a region that contains only six small objects in order not to interfere with the avatar mobility patterns, e.g., avatars could be blocked behind a virtual building. In the rare cases where an avatar meets an object on its path, the avatar simply deviates its mobility pattern and gets around the object. Finally, the region we choose for the experiments is an un-popular SL region generally empty of avatars. In this way, we reduce the chance that external users interfere. Nevertheless, during the experiments we continuously verify that no real SL user connects to the region we use as a test-bed.

VI. EVALUATION: P2P VS C/S

A. Correctness

We start by looking at the probability to have inconsistencies in the avatar AoIs. We evaluate the inconsistency for each avatar AoI every $200\ ms$ or anytime a modification of the AoI occurs. Figure 4(a) shows the Cumulative Distribution Function (CDF) of the inconsistency values computed on both C/S and P2P Second Life. Since we plot the inconsistency values in logarithmic scale (x-axis in Figure 4(a)), the two curves are truncated respectively for inconsistency values equal to 0.02 for P2P and 0.05 for C/S, i.e., the smallest non-zero values measured during the experiments.

Figure 4(a) shows that P2P achieves higher consistency than C/S. Avatars have a perfect view of their AoIs, i.e., inconsistency equals 0, in about 55% of the cases compared to 40% of the cases in C/S. The distance between the two curves is roughly constant for inconsistency values smaller than 0.2, indicating that P2P produces a gain of correctness in the user experience of about 20%. For inconsistency values larger than 0.3-0.4 the two curves nearly overlap. These high inconsistency values happen in presence of churn (i.e., login/logout operations) and fast avatar movements (e.g., teleport). While the SL server suffers these events due to an increase on its load [20], the P2P overlay suffers due to the difficulty in maintaining a consistent Delaunay triangulation [1][18]. This result indicates still an open issue with the Delaunay Network.

We now want to understand how frequently inconsistency events affect an avatar during its SL journey. Figure 4(b) plots the CDF 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 C/S suffers more from avatar inconsistency than P2P. In C/S, about 35% of the avatars do not see any inconsistency event, whereas this number nearly doubles in P2P. Interestingly, inconsistency in P2P never lasts more than 60% of the time an avatar spends in a region, whereas in C/S 10% of the avatars have an inconsistent view of their neighbor avatars during about 80-90% of their SL journeys. The reason behind this phenomenon is that the SL server spends a lot of time to correctly accomplish avatar login/logout as we investigate in Section VI-B. Subsequently, avatars with very short session times have an inconsistent AoI most of the time.

B. Interactivity

We now analyze the inconsistency duration in P2P and C/S in order to understand which architecture faster solves avatar inconsistencies (Figure 4(c)). As for the inconsistency results, 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 [5]. Conversely, Figure 4(c) unveils unacceptable inconsistency duration values under the current C/S architecture, e.g., 40% of the inconsistencies last for more than 2 seconds.

Figure 4(c) shows another interesting result. SL avatars can

experience a very long inconsistency duration both under a P2P and a C/S architecture, e.g., about 8 seconds in P2P and 15 seconds in C/S. Similarly to what we observed in Figure 4(b), churn (i.e., login/logout operations) and fast avatar movements are the causes of these high values of inconsistency duration.

Finally, despite the fact that the Delaunay Network allows a direct communication among SL users Figure 4(c) 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 [21]. Since avatars tend to form groups in SL [20] the dissemination of avatar state updates require multiple hops in the Delaunay Network before to reach all the interested avatars. This operation generates additional latencies that increase the inconsistency duration. However, according to the results presented in [15][2] and the budget of attention theory [13] these long inconsistency durations may be tolerated. In fact, these inconsistencies affect avatars located far away from the avatar that generates the state update, and so most likely not interested in the reception of the update. However, as a future work we envisage to optimize the Delaunay Network to further improve its performance in presence of avatar groups.

VII. CONCLUSIONS AND FUTURE WORK

Second Life (SL) is currently the most famous social virtual world counting more than 16 Millions registered users in September 2009. Despite that, the Quality of Experience (QoE) perceived by its users is poor. In this work, we investigate experimentally the benefits of Peer-to-Peer (P2P) to improve user QoE in SL.

We deploy a SL client that leverages the Delaunay Network, a very well-know design for P2P virtual worlds, to manage interactions among avatars. We use our client to evaluate P2P versus C/S Second Life. To do so, we execute several instances of our client over multiple Planetlab machines and we populate a SL region with controlled avatars. Avatar mobility traces extracted from SL are used to emulate real avatar behaviors. We show that a distributed avatar management for SL greatly improves its users QoE, making avatar experience more *correct* and *interactive*. However, we also unveil some weaknesses of the Delaunay Network in presence of churn, fast avatar movements and groups.

As a future work, we envisage to optimize the Delaunay Network to further improve its performance. Such an optimization is necessary as next generation virtual worlds will most likely require an higher level of interactivity among avatars [2]. Another avenue for future work consists in further improving the P2P-SL client. We aim at integrating in our client a distributed management for virtual objects [19]. Avatars first attempt to retrieve virtual objects from the P2P network, and only resort to contacting the server when no replica of a virtual object is available.

ACKNOWLEDGEMENTS

The authors would like to thank Fabio Picconi for his insightful comments and suggestions. This research was supported by the EU FP7 “Nano Data Centers” project.

REFERENCES

[1] 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'07*, Mellbourne, Australia, September 2007.

[2] A. Bharambe, J. R. Douceur, J. R. Lorch, T. Moscibroda, J. Pang, S. Seshan, and X. Zhuang. Donnybrook: enabling large-scale, high-speed, peer-to-peer games. In *Proc. of SIGCOMM'08*, Seattle, WA, USA, August 2008.

[3] A. Bharambe, J. Pang, and S. Seshan. Colyseus: A Distributed Architecture for Online Multiplayer Games. In *Proc. of NSDI'06*, San Jose, CA, May 2006.

[4] A. Bowyer. Computing Dirichlet Tessellations. *Computer Journal*, 24(2):162–166, 1981.

[5] M. Claypool and K. Claypool. Latency and Player Actions in Online Games. *Commun. ACM*, 49(11):40–45, 2006.

[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] K. James and C. Mark. Traffic Analysis of Avatars in Second Life. In *Proc. of NOSSDAV'08*, Braunschweig, Germany, May 2008.

[8] J. Keller and G. Simon. SOLIPSIS: A Massively Multi-Participant Virtual World. In *Proc. of PDPT'03*, Las Vegas, NV, USA, 2003.

[9] E. Kranakis, H. Singh, and J. Urrutia. Compass routing on geometric networks. In *CCCG*, pages 51–54, Vancouver, August 1999.

[10] 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*, 41(9):46–53, 2008.

[11] H. Liang, R. N. D. Silva, W. T. Ooi, and M. Motani. Avatar mobility in user-created networked virtual worlds: Measurements, analysis, and implications. *To Appear in Multimedia Tools and Applications, Special Issue on Massively Multiplayer Online Gaming Systems and Applications*, 2009.

[12] J. Liebeherr, M. Nahas, and W. Si. Application-layer multicasting with delaunay triangulation overlays. *Selected Areas in Communications, IEEE Journal*, 20(8):1472–1488, 2002.

[13] G. A. Miller. The Magical Number Seven, Plus or Minus two: Some Limits on Our Capacity for Processing Information. *Psychological Review*, 63:81–97, 1956.

[14] M. Ohnishi, R. Nishide, and S. Ueshima. Incremental Construction of Delaunay Overlay Network for Virtual Collaborative Space. In *Proc. of C5'05*, Cambridge, MA, USA, January 2005.

[15] J. Pang, F. Uyeda, and J. R. Lorch. Scaling Peer-to-Peer Games in Low-Bandwidth Environments. In *Proc. of IPTPS'07*, Bellevue, WA, USA, February 2007.

[16] A. Rowstron and P. Druschel. Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer systems. In *Lecture Notes in Computer Science*, pages 329–350, 2001.

[17] F. Stenio, K. Carlos, S. Djamel, M. Josilene, and A. Rafael. Traffic Analysis Beyond This World: the Case of Second Life. In *Proc. of NOSSDAV'07*, Urbana-Champaign, IL, USA, June 2007.

[18] M. Varvello, E. Biersack, and C. Diot. Dynamic Clustering in Delaunay-Based P2P Networked Virtual Environments. In *Proc. of NETGAMES'07*, Mellbourne, Australia, September 2007.

[19] M. Varvello, C. Diot, and E. Biersack. P2P Second Life: experimental validation using Kad. In *Proc. of INFOCOM'09*, Rio De Janeiro, Brazil, April 2009.

[20] M. Varvello, F. Picconi, C. Diot, and E. Biersack. Is There Life in Second Life? In *Proc. of CONEXT'08*, Madrid, Spain, Dec. 2008.

[21] J. Winick and S. Jamin. Inet-3.0: Internet Topology Generator. Technical Report CSE-TR-456-02, University of Michigan, 2002.